Embodied, in-medium design of VR game motion controls using interactive supervised learning

Carlos Gonzalez Diaz

Doctor of Philosophy

University of York
Computer Science
July 2023

Abstract

In recent years, motion controls have become a prevalent input modality for virtual reality (VR) games, offering players richer ways to interact with virtual environments. However, designing nuanced motion controls remains a challenging task for game creators. Interactive machine learning (IML) has been proposed as a solution to better support the iterative embodied process of movement interaction design. However, its application to VR game motion control design poses unique challenges. The goal of this thesis is to investigate the role of a novel IML design process and tool for VR embodied motion control design that addresses these challenges and supports game creators in their design process.

To achieve this goal, we first investigate considerations for integrating IML into game development pipelines, focusing on seamless integration with game engines. We then propose a novel IML process and tool, InteractML, that addresses these challenges, offering native in-engine integration.

We investigate the embodied design process with two studies involving game creators designing motion controls for VR games throughout ideation, implementation and evaluation. Results show that in-medium, embodied interaction design with IML enables game creators to bodily explore, prototype, and evaluate VR movement interactions with richer nuance. We find that designing for VR is better done in VR, that there is a social embodied cognitive dimension to embodied interaction design, and that there exists a dissonance between the creator's first-person embodied understanding of movement and the machine's third-person computational representation of movement.

In conclusion, this thesis presents a contribution to the field of VR game motion control design, embodied interaction design and human-centred IML. Our work has the potential improve the richness of motion control design in VR games and to inspire further work in the broader area of IML for creative applications.

Acknowledgements

I would like to express my sincere gratitude to my supervisors, Prof. Sebastian Deterding and Prof. Marco Gillies, for their invaluable guidance, support, and encouragement throughout my PhD journey. Their expertise, insights, feedback (and patience!) have been instrumental in shaping my research and helping me achieve my academic goals.

During my study I have been lucky to collaborate with Dr. Nicola Plant, Clarice Hilton, Dr. Ruth Gibson, Bruno Martelli, Dr. Michael Zbyszyński, Phoenix Perry and Prof. Rebecca Fiebrink. I am deeply grateful for the rich perspectives that these collaborations brought me, and for being considered to work alongside such great minds.

I would like to extend my heartfelt thanks to my parents, my brother, and my sister-in-law, for their unwavering support, love, and encouragement throughout my academic journey. Their belief in me and their unconditional love have been a constant source of motivation and refuge, and I will be eternally grateful for enjoying such privilege.

I am also deeply grateful to my partner, whose unwavering love, patience, and understanding have been a constant source of strength and inspiration. Her support and encouragement have been essential in helping me navigate the challenges of the PhD process and the COVID-19 pandemic. *Mulţumesc foarte mult!*

I would also like to thank the IGGI PhD community for their support, camaraderie, and friendship. The opportunity to be part of such a vibrant and supportive community has been a privilege, and I am grateful for the many friendships and collaborations that have emerged from this experience.

Finally, I would like to express my gratitude to the UK research culture for providing a rich and stimulating environment for academic inquiry. When I arrived to the country after finishing my MSc in Sweden and my BSc in Spain, I did not realise the richness of the opportunities for collaboration, networking, and professional development. This experience has been invaluable, and I am grateful for the many ways in which this culture has enriched my academic journey.

Declaration of Authorship

I declare that this thesis is a presentation of original work and I am the sole author. This work has not previously been presented for an award at this, or any other, University. All sources are acknowledged as References. I use *nosism* throughout this thesis for stylistic purposes, i.e., I use the pronoun ‘we´ to refer to myself. Instances of ‘we´ should therefore here not be read as (acknowledged or unacknowledged) collective authorship.

The list of works published as part of the completion of this thesis is as follows:

1. Carlos Gonzalez Diaz, Phoenix Perry, and Rebecca Fiebrink. “Interactive Machine Learning for More Expressive Game Interactions”. In: *2019 IEEE Conference on Games (CoG)*. 2019, pp. 1–2. DOI: 10.1109/CIG.2019.8848007
2. Nicola Plant, Ruth Gibson, Carlos Gonzalez Diaz, Bruno Martelli, Michael Zbyszyński, Rebecca Fiebrink, Marco Gillies, Clarice Hilton, and Phoenix Perry. “Movement interaction design for immersive media using interactive machine learning”. In: *ACM International Conference Proceeding Series*. New York, NY, USA: Association for Computing Machinery, July 2020, pp. 1–2. ISBN: 9781450375054. DOI: 10.1145/3401956.3404252
3. Nicola Plant, Clarice Hilton Goldsmiths, Marco Gillies, Michael Zbyszyński, Rebecca Fiebrink, Phoenix Perry, Carlos Gonzalez Diaz, Ruth Gibson, Studio Gibson, and / Martelli. “Programming by Moving: Interactive Machine Learning for Embodied Interaction Design”. In: *Workshop "The UX of Interactive Machine Learning" at NordiCHI ’20*. 2020. DOI: 10.1145/3359852.3359869
4. Carlos Gonzalez Diaz, Nicola Plant, Clarice Hilton, Michael Zbyszyński, Rebecca Fiebrink, Phoenix Perry, Ruth Gibson, Bruno Martelli, Sebastian Deterding, and Marco Gilles. “Bodystorming in SocialVR to Support Collaborative Embodied Ideation”. In: *CHI 2021 Workshop on Social VR*. New York, NY, USA: ACM, 2021, p. 3
5. Nicola Plant, Clarice Hilton, Marco Gillies, Rebecca Fiebrink, Phoenix Perry, Carlos González DÍaz, Ruth Gibson, Bruno Martelli, and Michael Zbyszyński. “Interactive Machine Learning for Embodied Interaction Design: A tool and methodology”. In: *TEI 2021 - Proceedings of the 15th International Conference on Tangible, Embedded, and Embodied Interaction* (Feb. 2021). DOI: 10.1145/3430524.3442703
6. Clarice Hilton, Nicola Plant, Carlos Gonzalez Diaz, Phoenix Perry, Ruth Gibson, Bruno Martelli, Michael Zbyszyński, Rebecca Fiebrink, and Marco Gillies. “InteractML: Making machine learning accessible for creative practitioners working with movement interaction in immersive media”. In: *Proceedings of the 27th ACM Symposium on Virtual Reality Software and Technology*. NY, USA: ACM, 2021. ISBN: 9781450390927. DOI: 10.1145/3489849

InteractML was developed from 2019 until 2023, and it went through several iterations following a Google-funded project that was led by Phoenix Perry and Professor Rebecca Fiebrink. In 2019 a publication introducing the tool at the IEEE Conference on Games was published together with Phoenix Perry and Professor Rebecca Fiebrink, where I acted as the main author of the publication and only programmer [57]. As such, I programmed the architectural foundations of the tool, its visual scripting interface and C++ to C# backend integration.

In 2020 the 4i project started, which was a 2-year multi-university project focused on VR creative practitioners using InteractML. The authors involved were Dr. Nicola Plant, Carlos Gonzalez Diaz, Clarice Hilton, Dr. Ruth Gibson, Bruno Martelli, Dr. Michael Zbyszyński, Phoenix Perry, Prof. Rebecca Fiebrink and Prof. Marco Gillies. As part of the project, Plant and colleagues [219, 220, 221] and Hilton and colleagues [113] published about how InteractML was used by technology artists as part of an embodied design methodology using machine learning. In these publications, I acted as a co-author and lead programmer of InteractML. I worked on the refactoring of the tool improving its stability and integration with custom game scripts in C# via a drag-and-drop functionality into the visual scripting graph. Nicola Plant programmed the re-skin of the nodes and designed the studies for publication, and Clarice Hilton programmed different example scenes and the event system that tackles controller input. We all contributed bud fixes and general system stability. Thus, with the exception of the visual re-skin of nodes and the event system for input handling, all changes on InteractML as part of this project were authored by me in dialogue with the project team.

Lastly, in 2022, I conducted a 1-year workshop-as-a-study project on game prototyping as the main author and single programmer, which is presented in chapter 5. As the single programmer, I extended InteractML's original architecture to support modules and added a panel-based 3D VR interface as an independent module, together with the explicit testing interface module. In sum, while InteractML was funded from various project sources and created in dialogue with other project members, the InteractML tool presented in this thesis is substantively my work.

The tool and methodology were presented in the following venues over the course of the PhD:

- 2019:
  - Games Developer Conference (GDC) 2019 [216]
  - Develop:Brighton 2019 [99]
  - IEEE Conference on Games (CoG) 2019 [57]
  - IGGI Conference 2019 [100]
- 2020:
  - Exhibition at Tate Modern London [222]
  - Games Week Berlin 2020 [101]

  - ACM Conference on Movement and Computing (MOCO) 2020 [219]
  - ACM Nordic Conferene on Human-Computer Interaction (NordiCHI) 2020 [220]

- 2021:
  - ACM Computer-Human Interaction (CHI) 2021, Workshop on SocialVR [102]
  - ACM Conference on Tangible, Embedded and Embodied Interaction (TEI) 2021 [221]
  - ACM Symposium on Virtual Reality Software and Technology (VRST) 2021 [113]

Contents

List of Figures

List of Tables

1 Introduction

Game interfaces have traditionally been limited to controllers such as keyboards, mice, and gamepads, often operating within the WIMP (Windows, Icons, Menus, Pointer) paradigm [304]. However, in the last two decades, motion controllers have emerged as a novel and increasingly popular input modality for games. Mass-market motion controllers like the Nintendo Wii [201], Playstation Move [272] and the Microsoft Kinect [187] have allowed players to interact with games using natural body movements. With the renaissance and consumer market push of virtual reality (VR) from the mid-2010s onward, motion controllers have become the default input modality for VR games. This is because VR aims to create a sense of immersion and presence with high degrees of movement freedom, which motion controls are seen to drive. As a result, motion controls have become an important design space for games, promising different, possibly more intuitive and enjoyable ways for players to interact with the virtual environment.

However, there aren't standard motion control design methodologies or tools for games, and designing or implementing precise and nuanced motion controls in video games remains a challenging task, particularly when using conventional game engine tools. Challenges include a design space limited by the sole reliance on colliders or triggers [104] that limit nuance and richness in movement; adapting movements to different body types [89]; and abstracting movement into meaningful computational rules for games. The growing popularity and market penetration of VR increase the urgent need for game engine tools that address these challenges and allow for (a) faster exploration of the design space, (b) simpler movement interaction prototyping, and (c) increased nuance in game motion control design and implementation.

Unpacking why these challenges persist, Gillies [93] explains that designing movement interactions with traditional approaches (as found in current game design practice and tools) is held back by the usage of existing interface metaphors, representing movement through rules, and interpreting movement via object interactions. Movement is a complex and multifaceted phenomenon and there are many different types of movements, each with their own unique characteristics and challenges. For example, designing a gesture-based interface for a mobile device requires different considerations than designing a full-body motion capture system for virtual reality.

In addition, movement interactions often rely on embodied movement knowledge that cannot be fully communicated explicitly – it is a tacit understanding of how to move one's body in a particular way, which is often acquired through practice and experience rather than explicit instruction [145, 175, 82, 93]. To better computationally capture such tacit knowledge, Françoise and colleagues [82] argue that there is an embodied cognitive relationship between design and implementation, where designers might realise how their movements don't *feel* as expected once implemented and require iterative loops of design and implementation to computationally express their intent. Therefore, Gillies [93] reasons that interactive machine learning (IML) can be such a design and implementation solution to better capture the iterative embodied cognitive process behind movement interaction design.

Machine learning (ML) techniques can also be a solution because of their ability to process nuanced movement data [93]. ML techniques have been shown to detect fine-grained differences in movement with great results by identifying patterns in motion capture data [307]. Break-throughs in motion tracking via ML techniques are commonplace, with a fundamental shift away from algorithmic rule representations to compute features and movement [151]. However, implementing standard ML techniques still requires significant expertise not common among game designers and other creative practitioners interested in using movement interactions, such as musicians [74], dancers [172] or technology artists [113, 221]. To support these non-ML expert users to engage in movement interaction design exploration and iteration, IML interventions have been proposed, developed, and validated as working approaches, with promising results emerging via GUIs that guide non-experts in performing ML tasks [63]. IML approaches present a so-called model steering loop, where users can iteratively train a machine learning model by (a) selecting or modifying features, (b) generating or revising pairs of human movement to model output examples as training data, and (c) selecting or modifying model parameters [63]. Studies investigating the IML iterative approach found that non-experts were able to generate ML models to correctly perform the diversity of tasks they desired, from recognising pictures to producing music [73, 6].

This iterative implementation with IML can be a solution for non-ML expert game creators designing motion controls for games. Still, IML-based game motion control design poses its own set of challenges that must be addressed. Firstly, designers must work with game development tools and pipelines to ensure that the motion controls are integrated seamlessly into the game, and currently no off-the-shelf IML solution offers native integration with game engines [74, 6, 80, 63], which is a necessity as part of the game development process [217]. Secondly, designers must be able to evaluate and steer the design process towards the desired player experience. This requires a way to feed game creator evaluation into the steering loop to ensure that the design meets the desired player experience goals. Evaluation of IML remains a challenging task because of the subjective nature of game creation practice, which yields standard ML evaluation metrics (i.e. accuracy, F-score, squared error) insufficient and many times requires support from qualitative approaches [73, 276]. Quantitative measures can investigate and describe algorithm behaviour (algorithm-centred evaluation), while qualitative measures can inform about user behaviour (user-centred evaluation) [26]. Hence, IML game development practices requires game creators take into account intended player experience outcomes, which adds further complexity and specificity to the evaluation methodology because current IML quantitative metrics don’t capture player experience [106]. Additionally, and as mentioned earlier, creative practitioners are not necessarily experts in machine learning, which requires a simplified approach to tackle the adoption of IML in game design.

1.1 Our IML tool: InteractML

In short, there is an opportunity for enhancing game development processes via IML for movement interactions. IML stands out as a promising approach since it allows users to directly act out their tacit embodied knowledge [93]. InteractML, the tool we developed for this end, is an interactive supervised learning solution integrated into the Unity 3D game engine in the form of a plugin [57]. Plant et al. [221] and Hilton et al. [113] investigated how creative practitioners understood and used InteractML to create a series of digital art pieces. Game development shares similarities as a creative field with digital art, but it is also differentiable by its heavy industrialisation and product end-goals. Videogame production manifests itself in technology that allows game designers to quickly prototype and iterate on game ideas and mechanics, usually simplifying logic interaction into reusable game engine components through a graphical editor [217]. Game development occurs within in-house or third party game engines that any new process, such our IML solution, needs to interface or integrate with [217]. Hence, any new system for a game engine should ideally exist as a no-frills integration that supports the modular and iterative industrial nature of game production, meant for digital manufacture of games on a time efficient schedule [138]. Videogame design goals usually attempt to elicit positive player experiences that will induce players to continue playing and differentiate the end-product from the competition [250]. Game designers usually perform personal qualitative experiential assessments while implementing interactions before engaging in external passes of quality assurance with players and testers [16]. How movements are implemented, iterated on, and recognised to ensure such a positive end-user experience is therefore relevant in making a successful game [19, 126].

Designing game movement interactions with InteractML differs from other IML processes studied in the literature because of the above-mentioned industrial nature of game development requirements and design goals. InteractML, as a toolkit, supports modularised and iterative prototyping of interactions that are integrated directly into the game engine, and can be incorporated into game development processes by interfacing with game engine components and scripts in the IML Graph editor [57, 113]. Such a no-frills integration has the intention of reducing friction and iteration time in a creative process beset by time consuming challenges [138, 217]. The interaction loop of InteractML aims at rapid iterative implementation and direct evaluation of movements inside of the game scene, where designers can experiment with movements that are not only usable or efficient, but also pursue a specific game feel [55] or aesthetic [154], and that are engaging [160], immersive [314], natural [180], or enjoyable [265] to perform. Furthermore, InteractML offers both a desktop visual scripting interface and a VR model steering interface to facilitate embodied interaction design processes. The supervised learning paradigm used by InteractML requires raw data to be processed into meaningful features that the algorithms can be trained on. The process of synthesising such features from data is known as feature extraction [236]. InteractML offers a selection of relevant movement feature extractors through its node interface that users can then select from to generate their training datasets [57, 113]. The process of selecting a reduced significant representation set among the feature extractors is known as feature selection, and the selected feature sets will have a substantial effect in the discrimination ability of the learning algorithms [178]. Iterative feature selection is a relevant and challenging stage in the interactive supervised learning workflow when implementing and evaluating motion data [73, 178], and it has not been previously investigated with InteractML.

Therefore, there remains open questions in the literature. Firstly, there are no clear standard design methodologies for motion controls in games which are currently the standard input modality in VR gaming [103, 89, 251, 15, 98]. Secondly, there is no tooling that marries the design and implementation processes to make use of the tacit embodied knowledge occurring from motion control design [82, 93]. Thirdly, even though IML has been previously investigated on other creative domains [74, 221, 172], it has not been investigated in motion control creation for games where design, implementation and evaluation stages are iteratively followed [251, 305, 16, 98] inside of game engines [217, 301, 83].

1.2 Research Questions and Thesis Overview

The goal of this thesis is to investigate the role of a novel in-medium design process and IML tool for VR embodied motion control design. Consequently, in this thesis we investigate the following research questions:

1. *What are the design and implementation considerations for an IML tool for virtual reality VR game motion control design?*
2. *What is a functional IML tool that addresses these considerations?*
3. *What are the opportunities and challenges that game creators encounter when ideating VR game motion controls in-medium?*
4. *What are the opportunities and challenges that game creators encounter in prototyping and evaluating game motion controls with an IML tool?*

In order to answer the research questions, we will review the relevant literature in chapter 2 on motion controls, design and development methodologies, positive experiential qualities of motion controls, embodied interaction design, and interactive machine learning. We will look into answering research question 1 in chapter 3, where InteractML will be described in detail as a tool for game motion control design inside a game engine. To do this, we will expand on prior published work by the author of this thesis [57, 219, 220, 221, 113]. Gonzalez Diaz and colleagues [57] published for the first time about InteractML at IEEE CoG 2019, where the author of this thesis acted as the main author for the publication and the main programmer of the tool under the supervision of Professor Rebecca Fiebrink and Phoenix Perry.

Plant and colleagues (2020, 2021) [219, 220, 221] and Hilton and colleagues (2021) [113] published about how InteractML was used by technology artists as part of an embodied design methodology using machine learning at ACM NordiCHI'20, ACM MOCO'20, ACM TEI'21 and ACM VRST'21. In these publications the author of this thesis acted as a co-author and lead programmer of InteractML.

We will answer research question 3 in Chapter 4, where we will investigate how VR game creators ideate and design VR movement interactions in-medium following an embodied ideation methodology, drawing influences from bodystorming [246, 23], embodied sketching [174] and somaesthetics interaction design [116]. The chapter qualitatively compares game creators involvement with the methodology from an in-medium and out-of-medium perspective. The chapter expands in depth prior work published at the ACM CHI'21 Workshop on SocialVR, where Carlos was the main author [102].

We will answer research question 4 in Chapter 5, where we will investigate how VR game creators prototype VR motion controlled games and implement movement interactions using InteractML in the Unity3D game engine. In this chapter, we investigated creator's iterative development of the movement interactions from a games-focused IML perspective, which involved studying (a) how they define and train IML models, (b) how do they subjectively assess the IML models' behaviour, and (c) how do they reflect on their assessment to debug/steer/integrate the IML models as part of their game scenes.

Finally, in chapter 6 we reflect on the findings from the thesis overall and their implications for games development, motion control design, interactive machine learning and movement technology. We suggest a series of potential promising research directions based on our reflections and conclude the thesis with a summary of findings.

2 Related Literature

2.1 Introduction

This chapter aims to familiarise readers with the terms, methods and essential concepts behind the creation of motion controls in video games, embodied interaction design, and interactive machine learning. In this chapter, we aim to identify the state of the art and open questions that motivate the thesis. To achieve this, we will conduct a comprehensive review of the literature on game motion controls in VR, motion control design and development practices, experiential qualities of motion controls, embodied interaction design methodologies and interactive machine learning.

Section 2.2 describes forms of motion controls in games and their renaissance in VR gaming. Next, section 2.3 introduces and describes literature on game motion control design and development. The section highlights how while there is prior literature discussing design guidelines, the topic of interaction design methodologies has received less attention in game design literature. As the aesthetic experience or 'game feel' of controls matters greatly in video games, we next cover what positive experiential qualities motion controls can deliver to players in movement-based games in section 2.4. Since the topic of movement interaction design has received less attention in game design, as per section 2.3, there is a need to explore the broader movement interaction design literature in the human-computer interaction (HCI) field to gain a better understanding of how to design effective and engaging movement interactions in VR games. Therefore, section 2.5 introduces embodied interaction design design principles and methods to discuss what challenges exist when designing and implementing movement interactions from a creator perspective. Finally, section 2.6 introduces the interactive machine learning (IML) methodology as a methodology to implement movement-focused interactions, and review existing IML systems for creative use in the literature and evaluation methodologies of IML systems. At the end of the chapter, we will summarise the state of the art and identify the design and implementation considerations for an IML tool for virtual reality VR game motion control design.

2.2 Motion Controls in Games

After first being introduced to the mass market with the Nintendo Wii and Microsoft Kinect, motion controllers have experienced a renaissance in virtual reality (VR) gaming. All current popular VR platforms – HTC Vive, Valve Index, Meta Quest or PlayStation VR – by default support motion control and offer bespoke controllers, presumably to increase immersion and presence in the virtual world. We define motion controls as computer interactions that require some form of physical movement to perform them, and motion controllers as the hardware input device that tracks user physical movements into computer signals.

There are various methods for tracking motion: sensing the whole body, using one or both hands, or tracking the head. Notably, all VR platforms support and most published VR games today require two-handed motion control with head tracking. However, due to its novelty, there are few if any best practices for two-handed motion control schemes for games, particularly when it comes to its effect on player experience. Due to this, we consider important to review current forms of motion controllers and design dimensions.

2.2.1 Characteristics and variants

To understand possible best practices for VR motion control design, we first need to understand motion control characteristics. Motion controls have a series of characteristics that describe how well they track user motion and position in space, namely accuracy, update rate, latency, drift and jitter [39]. Below we provide the definitions from Burdea and Coiffet (2003) [39] for each technical characteristic of a tracker:

- **Tracker accuracy:** "the difference between the object's actual 3D position and that reported by tracker measurements".
- **Tracker update rate:** "the number of measurements (datasets) that the tracker reports every second".
- **Tracker latency (or lag):** "the time delay between action and result. In the case of the 3D tracker, latency is the time between the change in object position/orientation and the time the sensor detects this change".
- **Tracker drift:** "the steady increase in tracker error with time"
- **Tracker jitter (or noise):** "the change in tracker output when the tracked object is stationary".

Depending on how many spatial axes the sensors track, a motion controller can have different degrees of freedom, with 6 degrees of freedom representing complete three-dimensional movement and rotation in space [39]. Sensors differ as well. The major distinction here is whether they measure position *indirectly* with accelerometers, gyroscopes or magnetometers; or *directly* with external visual, sonic or magnetic trackers [39]. Nowadays mass-marketed game motion controllers mix direct trackers with indirect sensors to calculate six degrees of freedom [114]. In addition, input devices can differ in how many communication channels enable users to interact with a computer. Devices can be *unimodal* if they are based on one modality, or *multimodal* if they combine many [140]. Moreover, input can be analog if it allow infinite amount of input values, or digital if it is a finite and discrete number [155].

The relation between input and output in a motion controller is commonly described with a transfer function, which is a mathematical transformation of input data to the desired output [114]. The most relevant transfer function to this thesis is the control-to-display (C:D) ratio or gain function, the ratio between the movement of the input device and the corresponding movement of the controlled virtual object [114]. This ratio is well known in the context of traditional 2D mouse interfaces [21], but it is kept as 1:1 in VR to ensure a sense of presence and embodiment [142].

The output or feedback that the controller can give back to the user can be of different nature. Feedback can be proprioceptive, which relates to the position and motion of one own's motion; haptic for vibro-tactile or temperature information; and visual, when the output can be perceived with the user's eyes [39].

However, none of this address how it feels to hold a controller in one hands. The physicality of the controller can be described in terms of ergonomics, gripping, button disposition, shape, specialization, aesthetics, weight or the building material used [206]. The resistance of the device can be framed as elastic, viscous or inertial, depending if it increases with displacement, movement, or acceleration [60].

2.2.2 Current common forms in VR games

Once we have understood motion control characteristics, we need to have an overview of currently available VR motion controls for games. This will allow us to frame our understanding of current best practices later on. There are a number of different VR setups available to the mass-consumer market. The most popular ones are the HTC Vive [119], Meta Quest [186], Valve Index [10] and PlayStation VR (PSVR) and PSVR 2 [273, 274]. These VR system offer 6 degrees of freedom head and controller tracking by combining optical trackers with magnetic ones inside the headset and controllers. The optical trackers can be external in the case of the HTC Vive, Valve Index or PSVR1 [119, 114, 273], or internal on the headset in a 'inside-out' configuration in the Meta Quest and PSVR2 [10, 274].

In addition, all of these systems rely on the use of two-handed motion controllers to track player hand movement. The 'simplest' controller, the PSVR1 PlayStation Move [272], has the shape of a wand, with four buttons following the classical PlayStation aesthetic, a fifth wide *move* button that is positioned in the front-centre of the device and a last trigger linear-digital button in the back of the controller (Fig. 1). The controller embeds an accelerometer, a gyroscope, a magnetometer and a LED sphere to be tracked by an external visual optical tracker [272]. The controller is built with plastic and offers haptic feedback with vibrations.

The HTC Vive controllers are as well wand-shaped, but they display a more specialised button where the *Move* button was in the previous controller (Fig. 2). This specialised button is a haptic trackpad that can detect where the user is touching and pressing on its surface. Instead of having the general purposed buttons at the

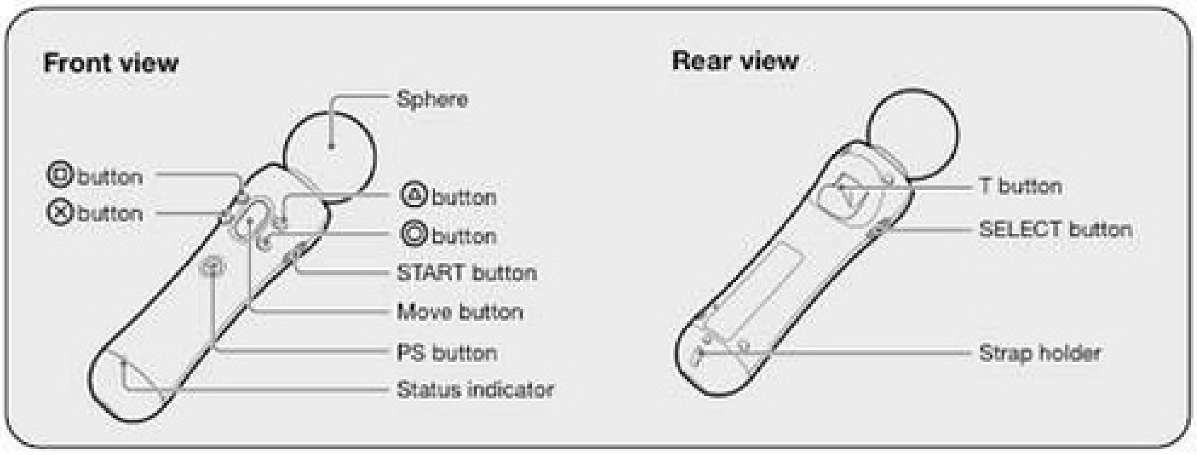


Figure 1: PlayStation Move Button Layout [272]

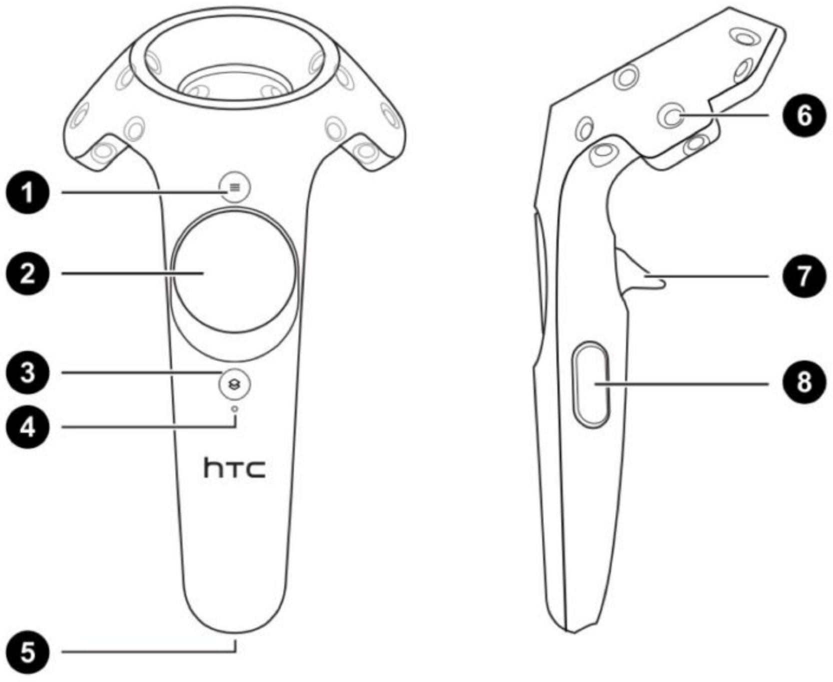
front, they are disposed in the back in the form of two grip buttons, together with a trigger with similar characteristics to the PlayStation Move trigger button. This controller is equipped with 24 sensors to be tracked by HTC Vive's optical tracker *lighthouse* [119].

The Valve Index offers an iteration of the controller originally called Valve 'Knuckles' [303] but nowadays referred to as 'Index Controllers' [10], which offer a different disposition of the same buttons, reallocating the grip buttons to the front and modifying the original wand shape to adjust the gripping to the hand (Fig. 3). It also includes haptic sensors for all the fingers, so the system knows whether the player is just having the fingers over the button or pressing it, to emulate the feeling of a hand in-game [10]. The controller includes a strap to adjust it to the palm of the hand to have the controller attached to the hand even with an open hand (Fig. 3).

The other popular family of mass-marketed motion control designs are the Meta Quest Touch [186] and PSVR2 controllers [274], and they display a shape reminiscent of a gun grip, with a button disposition similar to the Index Controllers, but without the strap (Fig. 4). The controller is also equipped with haptic sensors on each button to know when the finger is on the button but not pressing it. Instead of a big trackpad, it offers a centred joystick with two buttons below it and two triggers in the back [186]. The grip of the controller is designed to fit a half-opened hand.

Additionally, it is important to notice the existence of less popular motion controls in the field of bespoke motion controls for VR games. A very good recent example is the PlayStation VR Aim Controller [271], which is designed to play *First-Person Shooter* Games exclusively. The controller resembles the shape of a plastic two-handed fire gun, but is equipped with the same sensors as the PlayStation Move and displays two triggers and several more buttons (Fig. 5)

Lastly, many of these VR headsets support optical markerless hand-tracking via the



1	Menu button
2	Trackpad
3	System button
4	Status light
5	USB charging adapter
6	Tracking sensor
7	Trigger
8	Grip button

Figure 2: HTC Vive Controller [119]

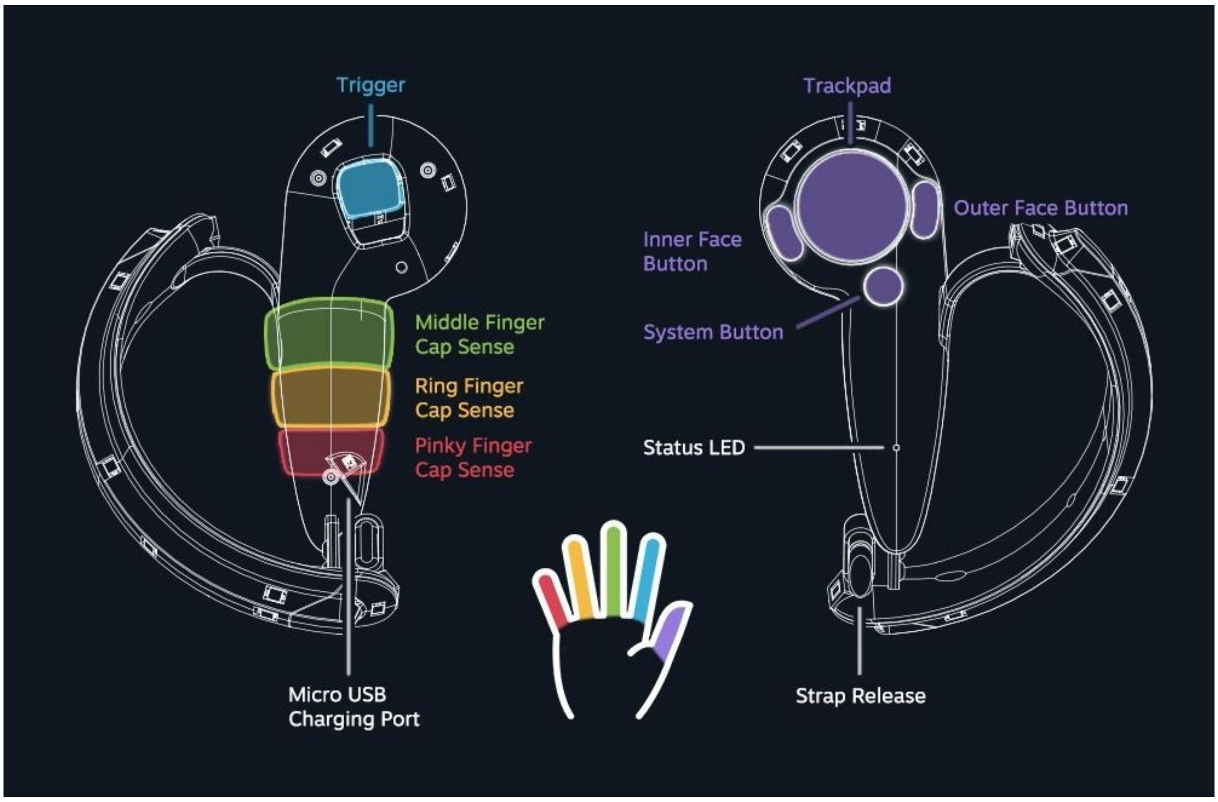


Figure 3: Valve Index Controller [303]

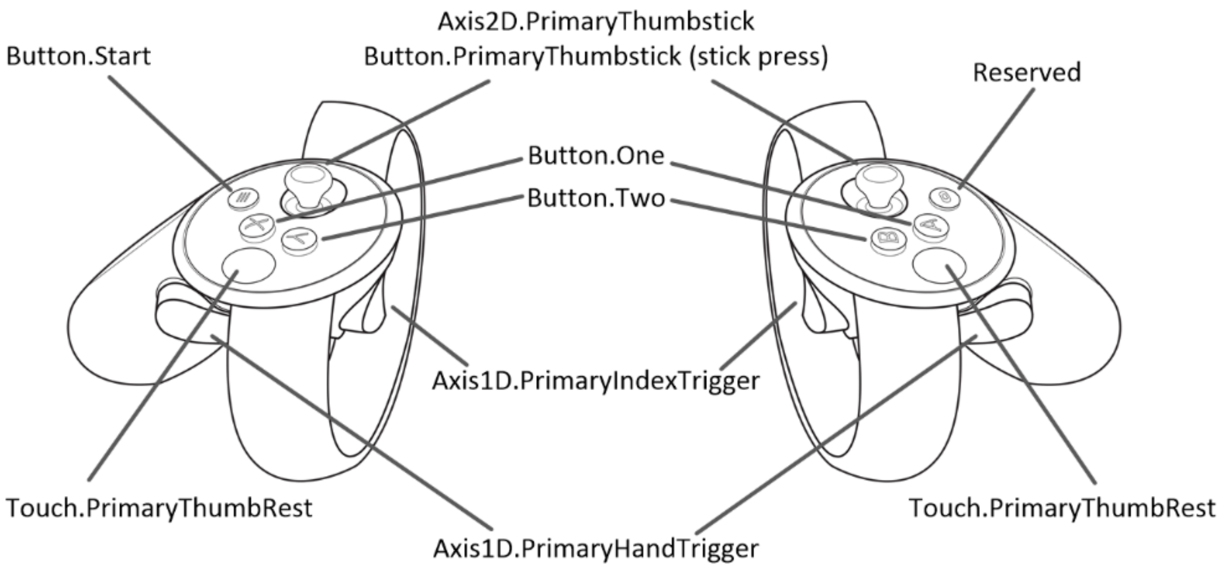


Figure 4: Oculus Touch [114]

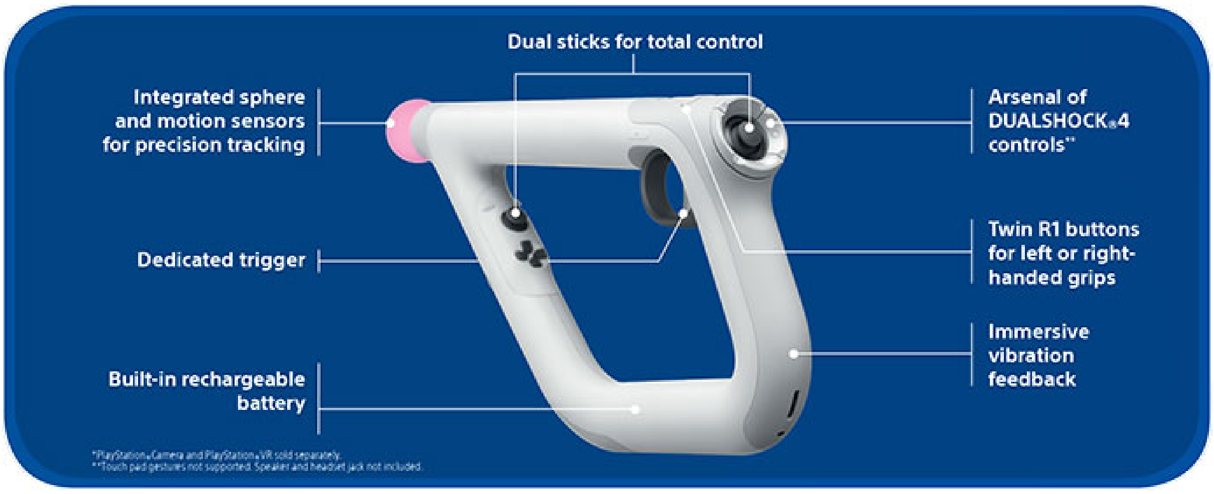

Dual sticks for total control
Integrated sphere and motion sensors for precision tracking
Arsenal of DUALSHOCK®4 controls**
Twin R1 buttons for left or right-handed grips
Dedicated trigger
Immersive vibration feedback
Built-in rechargeable battery
*PlayStation®Camera and PlayStation®VR sold separately.
**Touch pad gestures not supported. Speaker and headset jack not included.

Figure 5: PlayStation VR Aim Controller [271]

internal optical trackers on the 'inside-out' VR tracking configuration [10, 186]. With this input modality, users can perform mid-air interactions with their hands. Still, the two handed and 6 degrees of freedom paradigm still holds, hence the focus from this thesis on supporting such kind of interactions.

2.2.3 Summary

This section introduced and described current motion controllers in games, and shown how VR is the field were motion controls had a renaissance after the demise of previous commercial consumer entertainment systems in the early 2010s. The section described the vocabulary to technically describe tracking qualities of motion controllers, form factors, button dispositions, and the prevalence of 6 degrees of freedom in current VR systems.

2.3 Game Motion Control Design & Development

2.3.1 Introduction

In section 2.2 we introduced what current game motion controllers exist and how motion controls can be technically described. However, since the focus of this thesis is on game creators, it is important to understand how motion controls can be designed and implemented.

2.3.2 Game motion controls design principles

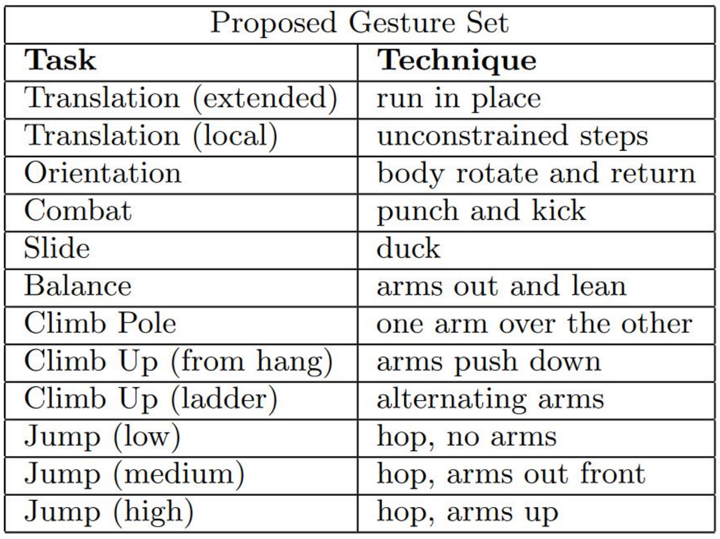
VR is the medium where motion controls are currently commonplace in videogames, but creators worked with motion controllers before with more limited sensing capacities (see section 2.2 for reference). Here, prior literature has looked into design guidelines and suggestions to incorporate movement into games.

Norton and colleagues (2010) [205] explored strategies and guidelines for developing full body videogame interfaces. The authors followed a Wizard-of-Oz methodology with the game Mirror's Edge to explore patterns in participants spontaneous movements in front of the screen while an operator played the game for them interpreting their movement. The authors argued that it was important to differentiate a set of "constrained interactions", which they defined as "those performable in a confined space, such as a living room with a game console setup". Their synthesised guidelines are as follows:

- "Design the environment to quickly challenge natural locomotion and then support the switch to compensated locomotion"
- "Expect running in place to be the compensating locomotion technique"
- "Retain the ability for natural locomotion in short travel tasks"
- "Design environments to challenge participant's steering early but unlike locomotion, expect to have to guide them to a particular technique"
- "Enable a means for players to recall in-game interaction possibilities for a full body interface"
- "Use head orientation to control user's gaze and the body's orientation to control steering"
- "Full body gestures may have cross interference therefore care should be taken in assigning functions"
- "Full body interfaces may have more gestures, but participants readily understand them and can use them"
- "Predicting possible player control gestures may be achieved by Pierce's assumption breaking methodology [218]"
- "Fatigue needs to be managed as a part of the design of a full body video game"

The authors also listed a gesture set to play first-person games like Mirror's Edge, which can be found in table 1.

Gerling and colleagues (2012) [89] suggested seven guidelines for the design of full-body interaction for older adults, based on two user studies with older adults



Proposed Gesture Set	
Task	**Technique**
Translation (extended)	run in place
Translation (local)	unconstrained steps
Orientation	body rotate and return
Combat	punch and kick
Slide	duck
Balance	arms out and lean
Climb Pole	one arm over the other
Climb Up (from hang)	arms push down
Climb Up (ladder)	alternating arms
Jump (low)	hop, no arms
Jump (medium)	hop, arms out front
Jump (high)	hop, arms up

Table 1: Gesture sets synthesised from Norton and colleagues (2010) [205]

(i.e. 60 to 90 years old) using the Kinect sensor, which uses computer vision to track full-body motion. The authors argue that their design guidelines relate to the guidelines from Norton and colleagues (2010) [205] in terms of avoiding too many gestures and taking fatigue into account. In addition, the authors highlight the potential benefit of configurable gestures to ease the cognitive cost of learning new gestures and increase the comfort for individuals. Their synthesised guidelines are as follows:

- “Create inclusive games by embracing age-related physical and cognitive impairments”
- “Create interaction paradigms that adapt to individual differences in player range of motion”
- “Provide fatigue management and prevent overexertion by appropriate game pacing”
- “Offer difficulty adjustments between players and individually scale challenges”
- “Provide natural mappings and clear instructions that support gesture recall to empower players”
- “Integrate continuous tutorials and player prompting to facilitate gesture learning and interaction”
- “Implement easy menus, startup and shutdown routines to encourage independent play”

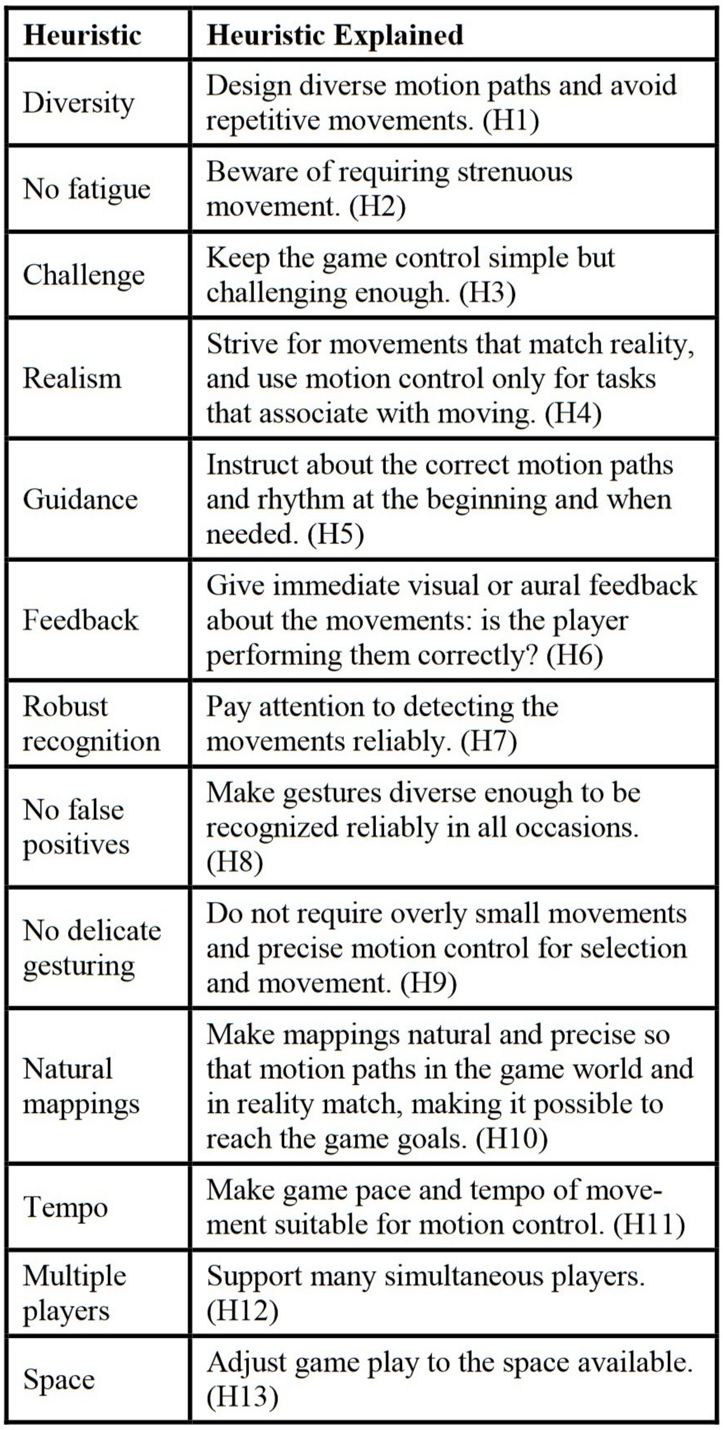
Hara and Ovaska (2014) [107] synthesised a set of design heuristics based on an extensive list of player problems arising from a systematic study of non-academic game reviews for the Kinect and PS Move. The authors grouped their heuristics into two broad themes: User-Centered Design of Movements (i.e. H1, H2, H3, H4, H5, H6, H11) , and Technical and Spatial Aspects in Motion Control (i.e. H7, H8, H9, H10, H12, H13). The complete list of heuristics can be found in table 2.

Silpasuwanchai and Ren (2015) [263] investigated how to several gestures could be performed simultaneously in gaming, similarly to how players use several buttons at the same time in a gamepad. They performed three user studies following the elicitation approach, where players spontaneously come up with gestures for given actions that happen at the same time in a videogame (Fig. 6). The authors synthesised a set of guidelines to assist creators in designing simultaneous full-body gestural interactions. The guidelines are as follows:

- “Prioritize events”
- “Prioritize immersion over playability”
- “Use the hand moderately”
- “Exploit transferability between leg and hand and right and left body parts”
- “Accommodate high tolerance for recognizing gestures”
- “Gesture reuse”
- “Design multiple gestures for one event, when needed”
- “Reducing fatigue by a small amount can have great impact”
- “Design kinetically feasible combined gestures”

Additionally, industry game designers also published design guidelines extending from direct experience releasing motion controlled games to the consumer market, albeit not in academic journals and conferences. For instance, Jack (2011) [129] dissects motion controls from his industry experience. He reasons a series of design guidelines meant to reduce misconceptions and mistakes that game designers transitioning from designing regular gamepad control schemes to motion control schemes. His rules are as follows:

- “Motion Control Games are not like Controller Games”
- “There are actually only so many motions”
- “It’s all about depth”
- “Computers don’t interpret people, they interpret points”
- “Fatigue is a big freakińfactor!”



Heuristic	Heuristic Explained
Diversity	Design diverse motion paths and avoid repetitive movements. (H1)
No fatigue	Beware of requiring strenuous movement. (H2)
Challenge	Keep the game control simple but challenging enough. (H3)
Realism	Strive for movements that match reality, and use motion control only for tasks that associate with moving. (H4)
Guidance	Instruct about the correct motion paths and rhythm at the beginning and when needed. (H5)
Feedback	Give immediate visual or aural feedback about the movements: is the player performing them correctly? (H6)
Robust recognition	Pay attention to detecting the movements reliably. (H7)
No false positives	Make gestures diverse enough to be recognized reliably in all occasions. (H8)
No delicate gesturing	Do not require overly small movements and precise motion control for selection and movement. (H9)
Natural mappings	Make mappings natural and precise so that motion paths in the game world and in reality match, making it possible to reach the game goals. (H10)
Tempo	Make game pace and tempo of movement suitable for motion control. (H11)
Multiple players	Support many simultaneous players. (H12)
Space	Adjust game play to the space available. (H13)

Table 2: Design heuristics for motion control in games from Hara and Ovaska (2014) [107]

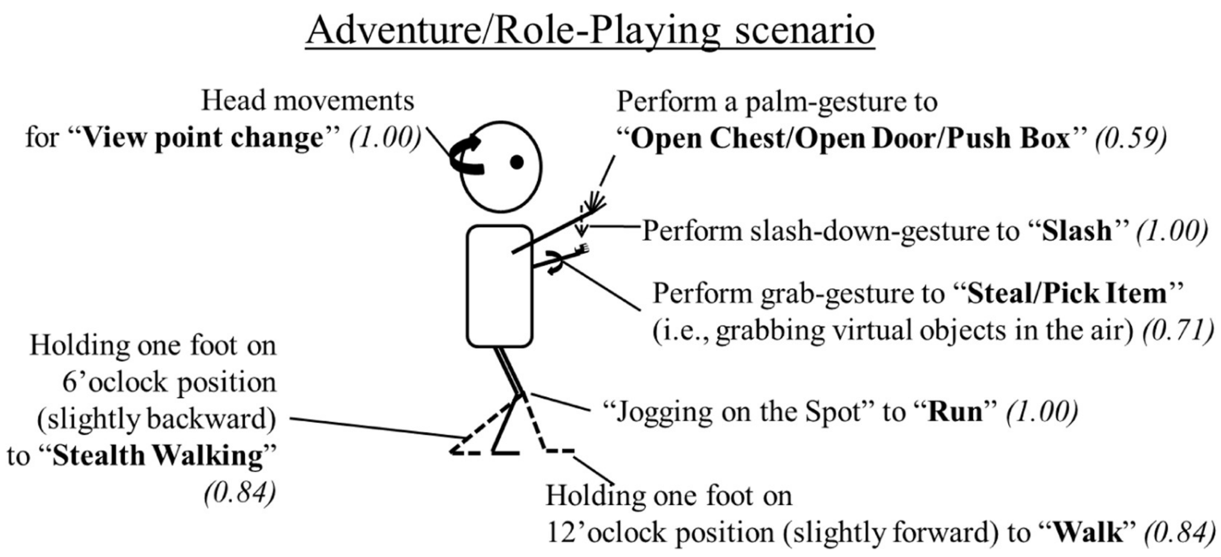

Adventure/Role-Playing scenario
Head movements for "View point change" (1.00)
Perform a palm-gesture to "Open Chest/Open Door/Push Box" (0.59)
Perform slash-down-gesture to "Slash" (1.00)
Perform grab-gesture to "Steal/Pick Item" (i.e., grabbing virtual objects in the air) (0.71)
Holding one foot on 6'oclock position (slightly backward) to "Stealth Walking" (0.84)
"Jogging on the Spot" to "Run" (1.00)
Holding one foot on 12'oclock position (slightly forward) to "Walk" (0.84)

Figure 6: A consensus of an user-define simultaneous gesture set for an adventure game [263]

- "Resistance and feedback are more important than ever"
- "Controls can (and maybe should) be interpretive"
- "Be wary of why conventions exist"

Additionally, Rogers (2014) [238] discusses game controls from his experience as a game designer, and synthesises the following design guidelines for motion controls:

- Gameplay breaks can avoid fatigue
- Recognition latency can cause player frustration
- Broad body movement can be better recognised than subtle motions
- Gestures should match their "real-world counterparts" where possible
- Players will use movements once learned, avoid introducing variation to gesture sets
- Graphic should guide player motion
- Constant audiovisual feedback is required to reduce player confusion
- Mid-air shape drawing can be misrecognised if the shapes are not simple
- Motion controls can be mixed with traditional button inputs to avoid saturating players with motion

If we compare these sets of design guidelines from both industry [129, 238] and academia [205, 89, 263], we can see that there are shared common themes. All design guidelines point out the importance of avoiding fatiguing the player and avoiding challenging movements because of the diverse body abilities of players. They also point out limitations with the physical space that players might have available at home, and how that might constraint player movement and gesture choice, with Norton and colleagues (2010) explicitly naming them as “constrained interactions” [205]. Similarly, all authors reason that there can be computer recognition limitations, and how to tackle that from a player perspective (i.e. give constant feedback to the player when they perform a movement) and a machine perspective (i.e. avoid very similar gestures or gestures that require precise movements). However, these design guidelines were proposed for motion controls that worked in front of a flat screen with limited degrees of freedom, and modern VR scenarios have a very different spatial disposition. In modern VR systems, the player has the freedom to move in the entirety of the available space, instead of the ‘cone’ that originates from the TV outwards. Furthermore, the VR controllers track 6 degrees of freedom, and there is no need to make approximations from signals detected from accelerometers or external cameras. To our knowledge, there is very limited research looking into design guidelines for VR game motion controls. Çatak and colleagues (2020) [46] synthesised a series of guidelines from five successful VR games published between 2016 to 2018. The authors divide the guidelines between perceptional, interactional and navigational design guidelines. Since the focus of this thesis is on interaction design, we list here their interactional design guidelines for VR games. The authors stress the importance of visual signifiers and interactional constraints. They define signifiers as visual cues that “inform the user to perceive affordances and figure out how things work [...] A signifier can be known as an existed experience from the player’s mind or can be learned in the game”. They define interactional constraints as “imitations of actions and behaviors, which make interaction design feasible and simplified . [...] Constraints are mostly used to limit the number of dimensions for virtual reality interactions”. Their synthesised guidelines are as follows:

- “Intuitive interaction designs easily fit the mental models of players. For example, shooting gun interaction in most VR games uses the trigger button to simulate real gun shooting”
- “Interaction methods must be consistent throughout levels. Consistency is necessary for all games, but it is essential for VR games. It is because VR is a full-body experience. If things keep changing between levels, players may lose orientation.”
- “Signifiers should be easily perceivable by users, but too many signifiers can be confusing. Therefore, they must be used in the proper amount.”
- “Constraints must be used where appropriate. If constraints are not used appropriately, they make it harder to complete interactions.”

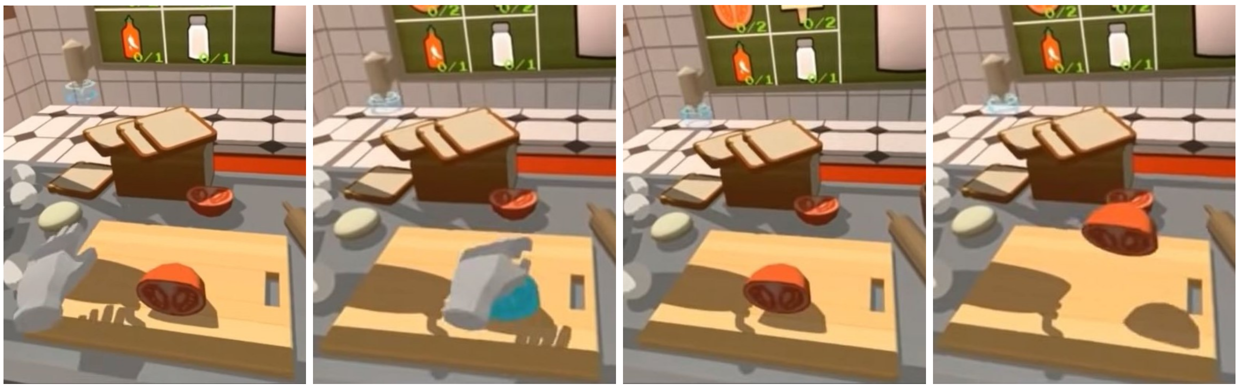


Figure 7: The “Tomato Presence” VR design concept [251]. Game designers from the Owlchemy Labs game studio found in 2017 via playtesting that players responded more positively to the object grabbed acting as a stand-in for the hand.

- “Constraints can be both realistic and unrealistic.”
- “Feedbacks are essential for effective interactions. However, designers should be careful not to use too much feedback which would overwhelm the player.”

Unlike of the lack of academic design guidelines for VR motion controls, industry venues have seen an abundance of game designers sharing insights they learned from their direct work experience releasing VR games to the consumer market. For instance, the Games Developer Conference (GDC) is an international annual conference known for its applicable talks from game studios into all topics behind crafting a game. In their 2017 GDC talk, Alexander Schwartz and Devin Reimer from the game studio Owlchemy Labs discussed design guidelines synthesised from their experience developing the VR game hit Job Simulator [251]. The guidelines are as follows:

- Adding physics to every object
- Near-field manipulation so that objects are in player range
- Account for players can have pre-conceived expectations when interacting with objects
- Every object needs a purpose
- *“Tomato Presence.”* (Fig. 7). The object grabbed in the hand acts as a stand-in for hand presence.

Similarly, game designers Dave Bennet and Patrick Jalbert from the game studio Schell Games discussed their experience of developing the VR sword fighting game Until You Fall at GDC 2020 [15]. However, they moved away from general object interactions because their game focused on sword swinging. They prompted the player to focus on sword movements by actually removing object interactivity from any other entity that wasn't a sword or an enemy. Their design guidelines are as follows:

- Traditional non-VR game feel designs can work in VR. For instance, using "hit pause" where the animation of the enemy pauses before hitting the player, screen flashes on hits, or camera shakes with powerful swings.
- Secondary VR interactions can distract the focus of the game. For instance, allowing to grab any object introduces design complexity instead of focusing on the swordplay. The designers brought that to the extreme where the player couldn't even drop the sword.
- Supporting the "player fantasy". For instance, prompting heroic poses and postures, or requiring big and exaggerated sword swings. The designers prompted different movement styles by rewarding each style (i.e. quick waggles would increase the "Rogue" power, whereas big swings would increase "Bruiser" power).

Design guidelines can play a crucial role in the creation of motion controls, as they provide a set of principles that guide designers in exploring the design space. The industry VR game interaction guidelines presented in this review highlight the tension between designing general object-focused interactions and the richness of movement. While general object interactions are concerned with manipulating objects within a virtual environment in a playful but physically plausible way, such as in Job Simulator (2017) [251], specific object interactions highlight the challenges of designing and creating a rich and diverse set of movements with that particular object, as in the sword swinging in Until You Fall (2020) [15]. Çatak and colleagues (2020) [46] synthesised such VR interaction breadth-or-depth tension in VR via their interactional constraint guideline, in which actually constraining the object interaction set allows designers to focus on interaction depth rather than interaction breadth. Interestingly, once comparing conclusions from non-VR motion control design guidelines to VR ones, we see that constraining interactions is suggested as a design trait to avoid [202], while in VR is suggested as a solution to the breadth-or-depth tension [46]. Additionally, non-VR motion control guidelines make a stronger focus on the amount of gestures [263] instead of on the richness of the gestures [15]. Fatigue is a dimension that isn't covered in VR guidelines, while the object focus isn't present in non-VR guidelines, when presumably players can get fatigued in VR and players would use objects in non-VR motion controls. On the other hand, both 'families' of guidelines highlight the importance of constant movement feedback [129, 89, 238, 46, 15] and visual cues or "signifiers" [205, 89, 238, 46, 15].

Nevertheless, previous work has focused on synthesizing design guidelines, but has not provided detailed methodologies for their implementation. Gomes and colleagues (2020) [98] suggested an iterative design toolkit for VR experiences. Their design toolkit consisted of the following phases:

1. **Start:** a set of reflective questions about the motivation of the project to answer by the creators.

2. **Empathy:** test, watch and learn from existing VR applications, as well as establishing a ground vocabulary to use in the project.

3. **Define:** frame the project and list potential challenges.

4. **Generate:** conception of novel or existing solutions for identified challenges.

5. **Select and refine:** work on improving the best candidates to identify points of tension in the design of the solution or the development of the solution.

6. **Prototype:** transform best candidate with less tension points into a digital prototype.

7. **Evaluation:** run user studies to gather data about the usability of the prototype.

We can see that Gomes and colleagues (2020) [98] motivate the need for such methodologies, but did not delve into the specifics of their design steps, which could be considered the (4) generate and (5) select and refine stages. To address this gap, it is necessary to review the literature on movement interaction design from an HCI perspective to gain a better understanding of how motion controls can be effectively designed. We will review such literature in more detail in section 2.5. Still, Gomes and colleagues (2020) explicit mention the iterative synergy between the design stages, and the prototype and evaluation stages. This is inline with Vanderdonckt and Vatatu's (2018) [305] suggested break-down of gestural user interfaces of (1) design, (2) engineering and (3) evaluation.

Therefore, in order to have more complete view of design methodologies, it is essential to understand how motion controls are implemented during the game development process. Game engines are common pieces of software used for implementing motion interactions in VR environments. Thus, the next section will explore in more detail the concept of game engines and discuss game development as a process.

2.3.3 Game development and game engines

Gomes and colleagues (2020) mention an iterative synergy between the design and prototyping stages in game creation [98]. Therefore, here we focus on how to create games and game motion controls using the standard software implementation tool: the game engine. The concept of a game engine has been previously defined as "a framework comprised of a collection of different tools, utilities, and interfaces that hide the low-level details of the various tasks that make up a video game" [260]. Nonetheless, the idea of what a game engine is can sometimes be dependent on the genre of the game at hand (i.e. a racing game engine, or a first-person shooter game engine) [9]. Modern game engines are considerably more generic [104]. One of the most popular game engines is Unity3D [300], which universities regularly use for teaching and research [318, 156, 7], and which is widely used in the games industry [137].

Petrillo and colleagues (2009) [217] show how game development is a working field beset by a plethora of industrial challenges, including management, communication and technological problems. They analysed videogame postmortems, which can be described as post-release reports of what went well and what went wrong with the development process. They found that, even though most problems were reported to be managerial, technological problems with game engine tools were present in 60% of games studied, with 35% of games reporting specifically problems with their tools. They illustrate such problems with a quote from the postmortem of the videogame *Diablo II*, where the developers expressed how important was to have tools that operated within their game engine [217]:

> “The greatest deficiency of our tools was that they did not operate within our game engine. We should have made tools that let us create content within the game engine” [217]

Kanode and Haddad (2009) [138] also argued for the need to optimise tools and pipelines to ease game development processes. Wang and Nordmark (2015) [310] studied what game developers considerered important when working with software architectures and creative processes in game engines. Because of the interdisciplinary nature of game development, where programmers work together with designers and artists, it is important to provide scripting solutions to support the creative side of the team (i.e. designers and artists) where the underlying technical complexity is simplified or hidden. Additionnaly, the authors found that scripting languages are preferred as they can help rapid prototyping of levels, scenarios or behaviours.

Aleem and colleagues (2016) [5] investigated the critical success factors in game development, and found more evidence on how industrial and management requirements beset the game development process. Technological problems can be entangled with management challenges, hence, streamlining and simplifying game engine and asset technologies are of great importance to reduce friction points during the development lyfecycle.

Marklund and colleagues (2019) [16] performed a systematic literature review on game development processes and found that there is a lack on agreement on common practices and definitions on what an“agile” development process constitutes. They highlight the agreement that, while a formally defined process doesn’t exists, there seems to be a clear pattern where the development process is non-formally highly iterative with strong reliance on regular playtesting loops. Newell and colleagues (2021) [199] stressed how iterative game development processes are still prevalent and empirically showed how regular and detailed play-testing was correlated with better game quality and ‘healthier’ working processes.

Therefore, there is consensus in the literature that game development is a process challenged by its industrial nature [217, 138, 5] where iterative development processes are predominant [16, 199]. A solution to help reducing industrial challenges and help iteration time is to streamline tools and technology to work within the game engine [217, 138] in the form of scripting languages for rapid prototyping [310].

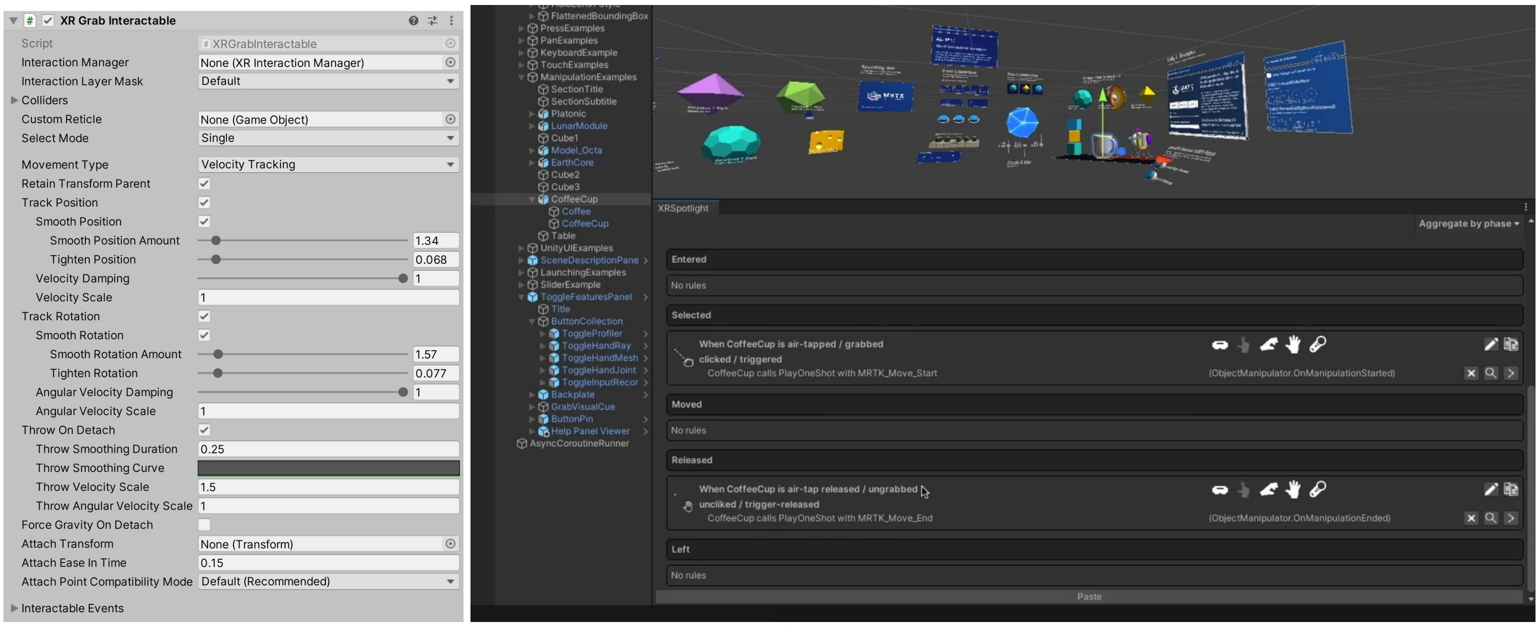
2.3.4 Motion Control Toolkits

To understand potential best practices in motion control implementation, we now look at what motion control toolkits exist for game engines. Before (in the past with Wii, Kinect, PSMove) motion control toolkits that recognised user movement were in-house, and unfortunately there we don't have available information of how such systems looked like in editor, as current mass-adopted game engines display systems on two categories: (1) Interaction Toolkits and (2) gesture recognisers. This subsection focuses on the Unity3D game engine [300] tools, since it is representative of other engines currently available (Unreal Engine, Godot Engine) because of its matured and establised full VR development support.

In the field of VR game motion controls development, a number of in-engine software toolkits appeared since 2015 to simplify the implementation and prototyping of motion controlled interactions [1, 64, 188, 301]. We will refer to these toolkits as *interaction toolkits*, as they focused on interactions between the user and the VR environment. We briefly describe each of them below:

- **NewtonVR [1]:** an interaction toolkit that allows to "pick up, drop, throw, and use held objects". NewtonVR focuses on offering a more believable object interaction set than what non-VR games previously used
- **Virtual Reality Toolkit (VRTK) [64]:** an interaction toolkit offering a collection of modular and generalised components to specify object and player interactions in VR, such as locomotion, grabbing or interacting with virtual menus
- **Mixed Reality Toolkit (MRTK) [188]:** an interaction toolkit offering several features and ready to use components to prototype interactions with objects (e.g. grab, pick, throw) or menus (e.g. sliders, panels, buttons, selections).
- **XR Interaction Toolkit (XRI) [301]:** an interaction toolkit developed directly by Unity, which aims to unify support between VR hardware vendors and offer modular components to specify object and player interactions, akin to VRTK.

All of the abovementioned interaction toolkits offer interactions based on how an object will interpret them, something these toolkits called an "interactable" [301].



Script	XRGrabInteractable
Interaction Manager	None (XR Interaction Manager)
Interaction Layer Mask	Default
Colliders	
Custom Reticle	None (Game Object)
Select Mode	Single
Movement Type	Velocity Tracking
Retain Transform Parent	✓
Track Position	✓
Smooth Position	✓
Smooth Position Amount	1.34
Tighten Position	0.068
Velocity Damping	1
Velocity Scale	1
Track Rotation	✓
Smooth Rotation	✓
Smooth Rotation Amount	1.57
Tighten Rotation	0.077
Angular Velocity Damping	1
Angular Velocity Scale	1
Throw On Detach	✓
Throw Smoothing Duration	0.25
Throw Smoothing Curve	
Throw Velocity Scale	1.5
Throw Angular Velocity Scale	1
Force Gravity On Detach	
Attach Transform	None (Transform)
Attach Ease In Time	0.15
Attach Point Compatibility Mode	Default (Recommended)
Interactable Events	

Entered
No rules
Selected
When CoffeeCup is air-tapped / grabbed
clicked / triggered
CoffeeCup calls PlayOneShot with MRTK_Move_Start (ObjectManipulator.OnManipulationStarted)
Moved
No rules
Released
When CoffeeCup is air-tap released / ungrabbed
unclicked / trigger-released
CoffeeCup calls PlayOneShot with MRTK_Move_End (ObjectManipulator.OnManipulationEnded)
Left
No rules

(a) (b)

Figure 8: (a) A screenshot of the 'Grab' Interactable component on the Unity3D game engine editor. (b) The XR Spotlight window displaying the rules distilled from the 'Coffee Mug' interactable object [83]

Therefore, any object, menu, button or entity in the virtual scene needs to be considered an interactable to model its behaviour. The behaviour is usually mapped in the form of pairs of triggers-responses, where usually triggers involve collision volumes in the engine known as *colliders* [103]. For instance, an interactable virtual ball that the player can grab can be triggered via a collision between the player controller and the collider volume of the virtual ball together with a physical press of the controller button. Once triggered, the response is 'to grab' the ball via moving it to the player hand. Another example can be pressing a button, where the trigger is the collision at a certain depth between the player controller and the collider volume of the button, thus triggering the response 'button pressed'. Interactables are meant to be represented as modular components that can be attached to any virtual object with a collider, and their behaviour can be scripted visually via dropdowns, sliders and drag&drop boxes on the Unity3D game engine editor (Fig. 8.a).

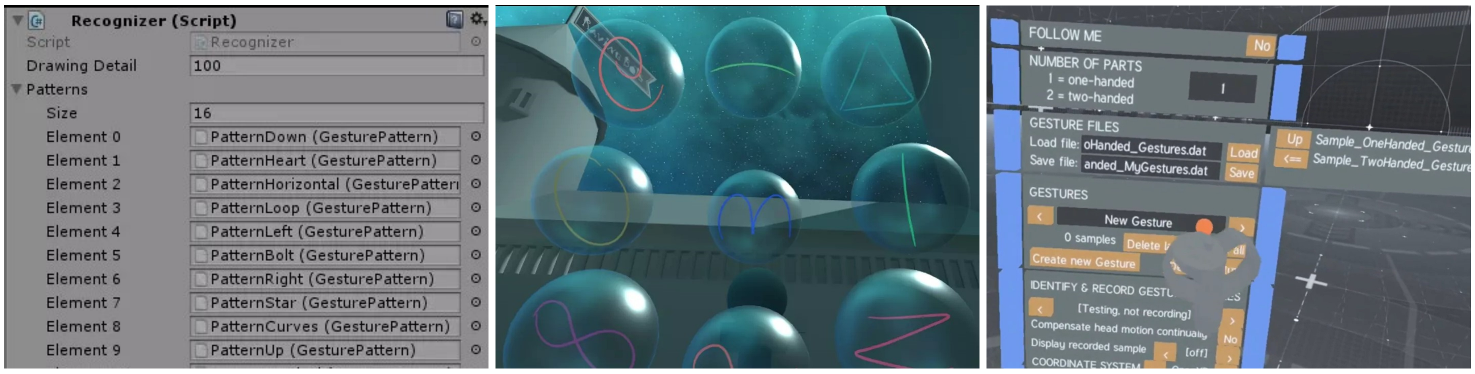
Vittoria and colleagues (2023) [83] argue that these interaction toolkits aren't novice-friendly and propose a solution to simplify the workflow process of creating interactable objects in Unity3D with their *XRSpotlight* system. Their solution follows an example-based approach to distill rules of interactables of any toolkit that are visually displayed in a new editor window on the game engine (Fig. 8.b). For instance, a novice user can use XRSpotlight to adapt an example interactable available in a example scene of MRTK into their own project and modify its behaviour by providing such existing interactable as an example to the system.

Still, these toolkits focus solely on object-focused interactions and there are more kinds of known interactions that are currently non supported out of the box via these toolkits. For instance, any interaction that is movement-focused, such as ‘wiggles’, ‘shakes’ or swings cannot be prototyped with the tools offered by these interaction toolkits, and could only be approximated via rules and colliders without incorporating additional plugins. For instance, the ‘wiggle’ motion could be mapped onto a pair of triggers-actions onto two colliders on opposite ends of an invisible bounded box around the controller. The rule could be described as the player moving the controller left-to-right in quick succession between the two colliders. However, how far apart do the two collider need to be from each other? And how quickly in seconds does the player need to move between colliders? What happens if the player wants to wiggle top-to-bottom the controller instead of left-to-right? We can see that a movement-focused approximation can easily become cumbersome and complicated to describe with these approaches. This means that game creators must write scripts that can extract meaningful signals from motion trackers to approximate gestural detection.

However, if game creators desire to incorporate some degree of movement recognition beyond these interaction toolkits, there exist third-party plugins for Unity to perform gestural recognition[54, 231, 232, 302, 176]. Gesture recognition, as a paradigm, involves the classification of defined gestures onto discrete classes. For instance, Vatavu and colleagues (2012) [306] developed a widely used 2D gesture recogniser named “$P”, which was later translated into an Unity plugin [54]. The “$P” gesture recogniser computes a cloud of points on a plane and searches for similarities across a dataset of labelled gestures. Users are able to define additional gestures as long as the label is consistent (e.g. draw additional circles on the screen to recognise the letter ‘O’).

Still, these gesture recognisers require a beginning and an end, and particularly the “$P” recogniser suffers the limitation of computing ‘flat’ gestures instead of 3D movements. There are more recent plugins supporting 3D gestures in VR that follow a similar paradigm of (a) providing lablled gestural data, and (b) evaluating the recogniser [231, 232, 302, 176]. Most of these gestural plugins offer a game engine editor interface with limited configurability, where game creators can add more samples to a gesture dataset and bodily evaluate how their recognition works [231, 232] (Fig. 9.a). Plugins such as “VR Magic Gestures AI”[232] and “MiVRy” [176], offer a graphical user interface in VR where the user can visualise labels for their gestures in 3D (Fig. 9.b), or modify configurations of their gestural recognition (Fig. 9.c).

Nonetheless, all of these gestural recognisers are centered around the mediums main input method (i.e. the mouse for desktop and the controller for VR). These gestures recognisers aren’t prepared to support creators when wanting to recognise a gesture from their head movement, or take into account their body as whole. This is because the classification algorithms have a set of movement features preset and can’t be reconfigured. The interface of “MiVRy” [176] allows to compute the movement from both controllers simultaneously, but it still falls short on the configurability required to fully support motion control creation in a game engine.


Recognizer (Script)

Script: Recognizer

Drawing Detail: 100

Patterns

Size: 16

Element 0: PatternDown (GesturePattern)

Element 1: PatternHeart (GesturePattern)

Element 2: PatternHorizontal (GesturePattern

Element 3: PatternLoop (GesturePattern)

Element 4: PatternLeft (GesturePattern)

Element 5: PatternBolt (GesturePattern)

Element 6: PatternRight (GesturePattern)

Element 7: PatternStar (GesturePattern)

Element 8: PatternCurves (GesturePattern)

Element 9: PatternUp (GesturePattern)

FOLLOW ME: No

NUMBER OF PARTS (1 = one-handed, 2 = two-handed): 1

GESTURE FILES

Load file: oHanded_Gestures.dat — Load

Save file: anded_MyGestures.dat — Save

Up Sample_OneHanded_Gestur

<== Sample_TwoHanded_Gestur

GESTURES

New Gesture

0 samples — Delete

Create new Gesture

IDENTIFY & RECORD GESTU

[Testing, not recording]

Compensate head motion continuall — No

Display recorded sample — [off]

COORDINATE SYSTEM

(a) (b) (c)

Figure 9: Desktop editor and VR interfaces of different gesture recognisers. (a) Unity editor interface of “Gesture Recogniser Asset”. (b) VR interface of “VR Magic Gestures Asset”. (c) VR interface of “MiVRy” plugin.

Furthermore, gesture recognisers require explicit marking the beginning and the end of a gesture during recognition by holding a button. This can limit expressivity as players wouldn’t want to hold a button press every time they perform a movement. Imagine a situation where a player would want to do sword fighting while swinging their controller, and required them to hold a button while moving. What would be the point of bodily performing the swing if you are already pressing a button?

Therefore, there is a need for an in-engine software solution that would support VR movement-focused interactions in a more flexible and general manner. This would allow to incorporate more nuanced motion controls and reduce technical challenges of game studios that are already limited by industrial time and management difficulties [217]. This issue will be explored in more detail in Chapter 3, where we will describe computational movement features and discuss our solution to implement motion interactions in VR. However, there is still a need to understand what kinds of experiential qualities motion controls can elicit on players. Therefore, we will review the literature of positive experiential qualities related to motion controls in section 2.4 to better frame the design of our in-engine solution for VR movement-focused interactions.

2.3.5 Summary

In this section we have reviewed motion game design and development principles, tools, and processes. We found that there aren’t a set of standardised motion game design guidelines applicable to VR motion controls; similarly, there is little work on how game motion design as a process or method actually or ideally operate; we therefore will reviewing broader HCI literature on motion control design in section 2.5. Regarding game development, we found the a field dominated by industrial challenges like integration into complex production processes and tooling pipelines, particularly the need for any tooling and process to integrate with the given game engine. A review of popular industry tools for creating interaction in VR showed that existing solutions tackle interactions from an object-centric perspective in pairs of triggers-responses, or from a gestural perspective requiring explicit movement segmentation and only taking into account the main input source (i.e. the VR controller). There is room and need for an in-engine movement-focused solution to prototype motion controls for games flexibly and we suggest a solution in chapter 3. Still, an understanding of what experiential qualities motion controls elicit in players is important to frame the design of such prototyping solution. We therefore review positive experiential qualities related to game motion controls in the next section.

2.4 Positive Experiential Qualities of Motion Controls

2.4.1 Introduction

To frame the design of our movement-focused motion control solution, we need to understand what experiential qualities game motion controls can elicit. Since entertainment game design is concerned with affording positive, enjoyable *player experience* (PX), a crucial aspect of game controls and control schemes is their *game feel* – the PX of how it feels to operate them [288]. In many games, the core interaction of manipulating the game state already presents a major source of positive PX – be it moving jumping Mario or swinging Spider-Man through space via your game controller, or swapping candies with a finger swipe in *Candy Crush*. Thus, good game feel or "enjoyable gestures" are similarly crucial design considerations for motion controls and control schemes. Just like PX is a broad construct that has been conceptualised and decomposed into may other constructs without a strong consensus model, motion control PX has been variously connected to PX constructs like Flow and Immersion [196]; Challenge, Fun, Frustration, Boredom, Predictability and Anxiety [215]; Immersion, Enjoyment and Natural Controls [247]; Immersion and Controller Naturalness [181]; Engagement [19]; or Presence, Enjoyment, Interactivity and Realism [77].

This subsection follows each of the relevant positive experiential qualities reviewed in the literature. We will begin by introducing engagement and immersion, and how both constructs might be related from a player perspective. Then, we will introduce the area of aesthetics of interaction and game feel, where the literature argues how movement can be described and how visuals, movement and experience might be entangled in games. Later, we review literature on enjoyment, to better understand how such a critical construct for games can be framed and understood from a media, player and controls perspective. Finally, we review theories of presence, embodiment and embodied cognition that of high relevance for VR and movement interaction design.

2.4.2 Engagement

Engagement is the first relevant experiential construct we cover in our review to frame the design of our motion control creation tool. There is substantial debate and literature around the definition of the term engagement. The definition and understanding of the concept, varies between the context of the study and the interaction that wants to be measured. Even though the focus of this thesis is on games, engagement can be studied in many different domains. For example, in [165] engagement is studied from the perspective of the employee in a company, being understood as a multidimensional set of constructs being named state, behavioural and trait engagement [165] . In the domain of customer engagement, understanding the concept as an state that can be produced because of the interaction of a customer, contemplating its nature as multidimensional process as well [33]. Student engagement is studied acknowledging its multidimensional nature as a state [11]. We can look at engagement from a motivational and emotional point of view, defining it as “the behavioural intensity, emotional quality, and personal investment in another person’s involvement during an activity.” [234] and dividing the extent of engagement into behavioural engagement, emotional engagement, cognitive engagement, and voice. Reeve (2008) understands engagement as a state that you can enter because of a physiological or psychological response to an antecedent condition, being able to measure the level of engagement thought different psychometrical factors, such as blood pressure or eye focus. All of the perspectives share the understanding of engagement as a multidimensional state, although it can be understood as well as a process [33].

Regarding a more technological field, user engagement with technology has been defined by O’Brien and Tom (2008) as “the process sustained when users are able to maintain their attention and interest in the application, and is characterized by positive emotions”, understanding the concept as a process with different stages. They propose their Model of Engagement as a quality of user experience divided in several phases: Point of Engagement, Period of Engagement, Disengagement and Re-Engagement, with multiple attributes of varying level of intensity on each stage [209]. In his more recent work, O’Brien further discusses the concept and defines it as a process that measures “the depth of the actor’s investment in the action”. He discusses how it can be analysed during and after the interaction through a multifaceted perspective, with a need to vary the unity of analysis depending on the interaction, based on his work on the Process Model of Engagement [208, 209]. In comparison to the studies in the previous paragraph, engagement is understood as a process alone and while other authors acknowledge its multidimensionality as a set of constructs or factors [11, 33, 165, 234], O’Brien calls them attributes.

When related to digital games, engagement has been discussed to be a quantifiable construct, with the development of questionnaires such as the Game Engagement Questionnaire [32], where engagement is used as a “generic indicator of game involvement”, relating it to other concepts such as immersion, presence and flow. Another study performs a systematic review over 55 papers on engagement in digital entertainment games, concluding how complex and multi-factorial the term is understood [27]. They discuss that engagement can be analysed as a behavioural outcome of playing computer games or as a process with different stages[27]. Cairns extends from O'Brien's work understanding engagement as a process and studying how the Process Model of Engagement adapts into digital games [40, 209]. He discusses how there are several components to engagement named Control, Challenge, Feedback, Aesthetic and sensory appeal, Attention, Awareness, Interactivity, Novelty, Interest and Positive Affect. These attributes are a series of aspects of the process with varying levels of intensity that helps to measure the level of engagement [40, 209]. Cairns conclusion shows that engagement can be cycled as a process several times in one single game-play session, or one cycle may need several sessions, as well as the importance of the external factors into the engagement and the possibility of engagement outside of the game. He points out the necessity of adapting the measurement method of engagement depending on the genre of the game [40], as suggested by more previous studies [234] with a different view of engagement. A different study shows that player engagement is studied as a process, developing a framework and four main components consisting of objectives, activities, accomplishments and affects [249] .

During the review, we have seen that engagement is a very complex and multifaceted term studied through different fields and views. Engagement can be understood as a multidimensional psychological state [11, 33, 165, 234], as a behavioural outcome [27] and as a process modelled with different stages, attributes and components [33, 40, 208, 209, 249]. The subcomponents of engagement vary depending on the study, but is possible to agree on the multidimensional aspect of engagement, with concepts such as immersion, flow, involvement, affect and interactivity related. In regards of the view of engagement for this thesis, we will understand the concept as a multidimensional process with different stages and attributes following the view of Cairns (2016) and O'Brien and Tom (2018) [40, 209, 249] acknowledging that each stage of the process can be seen as a state [33, 234]. Because of that, we could also understand engagement as a behavioural outcome when the change of stages is produced while cycling the process [27], due to an antecedent condition that could trigger the process.

Despite of this being just a fraction of the research available, we can see that most of the research is focused on gameplay and game design and not much in interaction and control schemes. E. A. Boyle and colleagues (2012) and P. Cairns (2016) [27, 40] brings the topic of interactivity in their studies, mentioning how naturalness and embodied interaction might affect engagement in digital games. However, the connection with enjoyment seems unclear.

Another of the constructs that is related to positive experiential qualities of game controls and, in a repetitive fashion to VR, is immersion. In the next section we will look at how immersion is portrayed in literature to better understand how it can connect with enjoyment in VR games.

2.4.3 Immersion

Immersion is the second relevant experiential quality we review to frame the design of our motion control creation tool. Additionally, immersion is a highly relevant construct for games [35]. Yet, immersion is a complicated construct with different meanings and definitions depending on the field of study and the author that studies it, very similarly to the engagement literature. The definitions can easily be contradictory and can make the research on the field difficult, due to the complexity in understanding all the different meanings the construct gathers. Nonetheless, immersion is a well-known and referred term both inside and outside the academic environment, proving to be useful the time and effort spent understanding its different meanings.

Brown and Cairns (2004) [35] describe immersion as a “scale of involvement with a game” where exist a division into three different levels named Engagement, Engrossment and Total Immersion; depending on how strong the sense of immersion is. Engagement is the first and lowest stage of immersion, and to be engaged, the player needs to invest time, effort, and attention. However, firstly is necessary to pass the barrier of access, which is related to game controls and how the player becomes an expert interfacing with the game. The second stage is named Engrossment, where there is a high level of emotional investment in the game, having a strong effect reaching this point visuals, interesting tasks to perform, and plot. Total Immersion is the name given to the maximum level capable of being experienced. This level is as well referred to as presence, introducing the barriers of empathy and atmosphere to reach it. Here, empathy is the “growth of attachment” and atmosphere is the “development of game construction”. In this sense, immersion is understood as a state with different levels of depth.

The above definition [35] has been used in later studies [131, 252] to better understand the concept of immersion in terms of addiction and subjective and objective measures. Seah and Cairns (2008) [252] study the potential similarities between being immersed and being addicted to a video game, showing that addiction seems to be an extreme form of engagement and immersion. They define immersion as “an internal measure, or psychological measure, of engagement in playing videogames and the engagement/addiction factor as an external measure, or behavioural measure”. They show that immersion can be “a mix of psychological factors, specifically the cognitive and emotional involvement in the game, and sense of dissociation from the real world, as well as game challenge and game control”, comparing the term with flow [52], cognitive absorption [3] as well as GameFlow [287]. It is shown that immersion can be a precursor to flow, that can happen even if the game fails to provide flow, or that it overlaps with three factors of GameFlow. However, an important difference with the previous study is that it separates Immersion from Presence, defining Presence as “the sense of being present in some virtual world as opposed to the real world” and calling it “perceptual immersion” too, while differentiating it from “psychological immersion” [48] which is the one we are focusing on here. This way, by using a virtual reality headset together with a motion tracking technology to explore a virtual world, one can feel perceptually immersed; and as well one can feel psychologically immersed when playing a puzzle game where there is no clear environment to be immersed into [131]. It is important to remark that they are already starting to contemplate the fact that immersion can be produced far from a positive effect, when a game is frustrating for example.

Jennett and colleagues (2008) [131] show that Immersion can be measured both subjectively, through questionnaires, and objectively, through task completion time and eye movements. They ran three different experiments. The first experiment was related to task completion time, and showed that “the more immersed a person was when playing a game, the longer it took them to complete a tangram task afterwards that was not related to the game”. The second experiment was related to eye movement frequency, and showed that “participant’s eye movements in the immersive condition significantly decreased over time”. The third experiment was related to pace of interaction, and showed that “emotional involvement appears to be a key factor on immersion”. These findings support and build from the studies by other authors [35, 252], but still can’t tell if other objective measures, such a biosensors and body motion analysis, can be used to measure Immersion as defined above.

Nevertheless, Calleja (2011) understand differently the term immersion and offers an alternative naming, Incorporation, that in his opinion suits better what the current literature defines as immersion or presence. He makes a definition review of immersion as a term, both outside and inside the game research literature, identifying two meanings [42]. The first one he names Immersion as Absorption, which refers to the fact of being immersed in some condition, action, interest, etc.; and the second meaning which he calls Immersion as Transportation, refers to the fact of being transported to the virtual world. Immersion as Absorption would be equivalent to the previous definition of Psychological Immersion, and Immersion as Transportation to Perceptual Immersion [48, 131]. Following this idea, he disagrees with the concept that there is only an unidirectional relation between the player and the virtual environment, introducing his term Incorporation, as an “experiential phenomenon that accounts for the simultaneous assimilation into consciousness of the virtual world and the systemic acknowledgement of the player’s location and existence therein” [43]. Consequently, the term can only be used in ergodic media, due to the fact that there is requirement from the medium to specifically acknowledge the player’s presence and agency within the virtual world.

In order to facilitate research into this area, Calleja (2011) explains in the Player Involvement Model, a framework intended to describe the phenomena afforded by involving gameplay [43]. The model introduces two temporal phases. The first one is “the macro” and represents the offline involvement, and the second one is “the micro”, representing moment-to-moment involvement during gameplay. There are as well six dimensions of player involvement, named kinaesthetic, spatial, shared, narrative, affective, and ludic.

All of these theories could help to build the bridge between motion controls and player experience in VR games, but there isn't enough research relating immersion directly to engagement or enjoyment. It might be that a broader qualitative perspective might help describe relevant experiential factors from movement interactions. The next section will introduce the field of aesthetics of interaction and how it could help creators to verbalise their motion controlled ideas.

2.4.4 Aesthetics of Interaction

Since a broader qualitative perspective might help frame relevant VR motion controls experiential factors, we review the field of aesthetics of interactions in this subsection. When interacting with motion controlled VR games, creators and players constantly perform motion, resembling the 'natural' interaction they would be performing if they were present in that virtual world. Nevertheless, it is hard to categorise what is the meaning of player's gestures if we cannot refer to them in a qualitative way.

The same way we can describe several different mediums with qualities, we could use several qualities to describe an interaction. The term aesthetics of interaction was first mentioned in the first decade of the XXI century [154], conceiving 'aesthetic' as a system of values an interaction can have [154].

In a study conducted in 2007, the authors discuss how intelligent products cannot just have a beautiful appearance but a beautiful interaction as well. They discuss how an interaction can have both task-oriented meaning as well as aesthetic meaning, as the actions themselves may be aesthetically rewarding [59]. They as well point out the correspondence in the quality of movement between the input and the output in an interaction, connecting the richness of actions that a product allows for, the degrees of freedom that product component have, the sensitivity in the input, and the smooth movement in the output; naming this phenomena symmetry of input and output qualities [59]. Löwgren (2009) extends this work and introduces four concepts trying to answer some of the questions when describing an interaction: *pliability, rhythm, dramaturgical structure, and fluency* [162].

In a literature synthesis, they review a total of 19 papers regarding this topic [154]. A first inspection revealed that the total of 151 attributes studied can be divided into two main groups, one for describing experiential interactions and another one for describing physical interactions. However, after studying in more detail the definition for each of the attributes, the authors proposed a division into 3 main sub-categories: prescriptive (describing the goal of the user with the interaction), descriptive/motor-level (describing the physical experience of performing the interaction) and descriptive/be-level (describing the psychological experience and meaning of performing the interaction). For both descriptive categories a further subcategorization was assigned, dividing the interaction in the be-level between the psychological needs of stimulation, security, competence, autonomy, relatedness, meaning and popularity; and the motor-level was divided into the temporal, spatial, action-reaction, presentation, forces and meta sub-categories [154]. Additionally, one of the most relevant qualities for game controls is game feel, which we will review in the following subsection.

2.4.5 Game Feel

Aesthetics of interaction offers a form of qualitative description and framong of movement. Related to aesthetics of interaction in digital games is the term ‘Game Feel’, which is defined as a “Real-time control of virtual objects in a simulated space, with interactions emphasized by polish” [288]. There are a total of five experiences for a game to convey in order to feel good: aesthetics sensation of control, pleasure of learning and mastering a skill, extension of senses, extension of identity and interaction with a unique physical reality within the game[288]. Swink presents Game Feel as a universal concept, but it can be narrowed down to specific in-game actions. In the study of Fasterholdt and colleagues (2016), the authors study the feel of the jump action in 2D platformer digital games, and how it relates to the physical implementation through game controllers, developing a framework to quantify different parameters in the feeling [70]. A similar study was conducted by Dahl & Kraus (2015) [55], but understanding how players feel a bespoke 2D platformer through a series of questionnaires, where words such a “twitchy”, “fluid”, “difficult” or “enjoyable” were used to describe the interaction, following some of the terminology in the interaction of aesthetics literature [154]. Nevertheless, the authors of the study conclude that players understand Game Feel and the different attributes to described their experience in different ways [55].

In the quest to label interactions, another study by Isbister and colleagues (2011) [127] looked at how much players rated their motion interaction with a set of Wii games in terms of fun, happiness, frustration, and energy levels. Their conclusions show that there is no apparent relationship in the amount of movement and how much players understand their interaction to be funnier, happier or more frustrating; although it shows that their self-reported energy level was higher when playing games with more movements. Following their quest to understand players’ feelings when performing movements, Isbister et al. (2011) developed a series of experiments with movement games to see whether movement adds emotional impact and increases social connectedness. Results showed that vigorous movement led to reports of higher arousal but not of more positive emotional valence [125, 128].

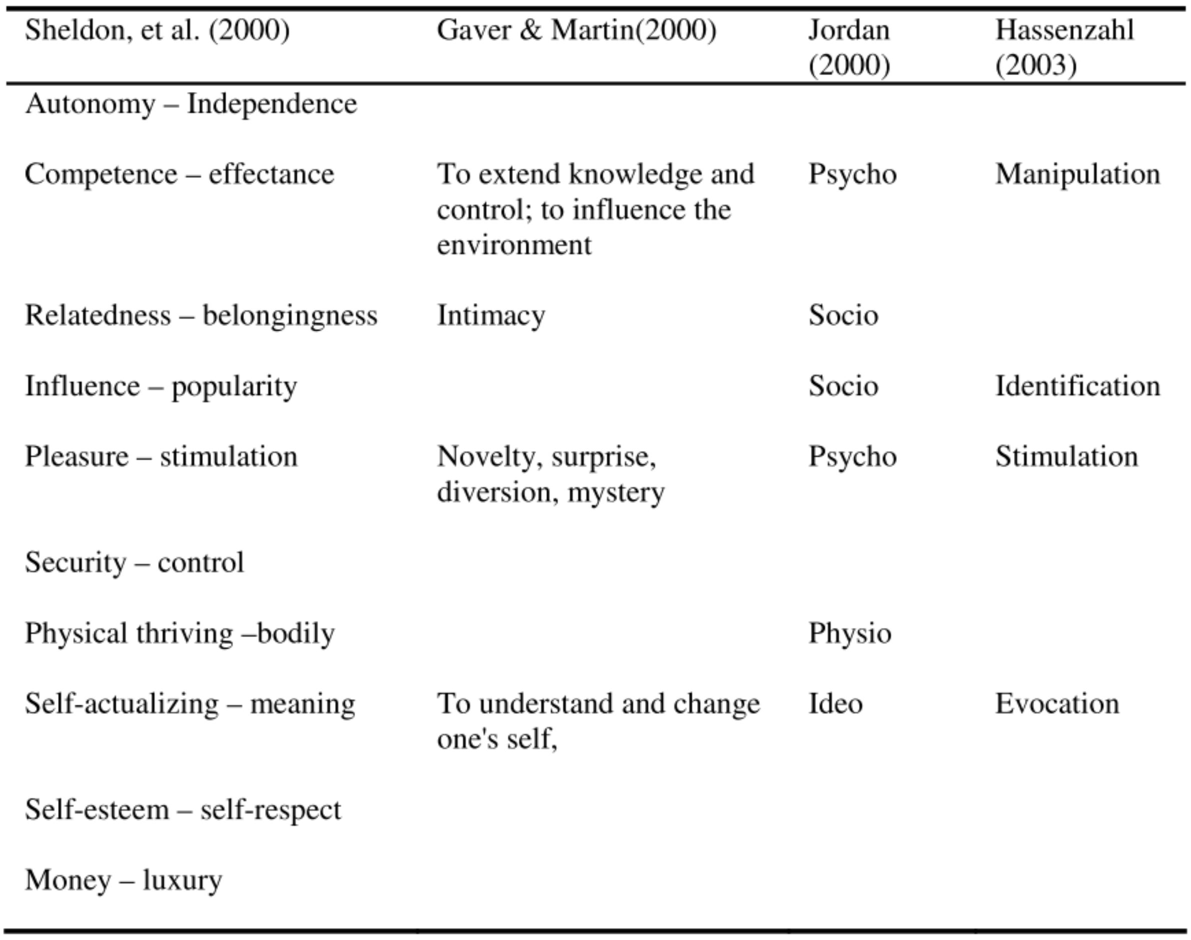
After reviewing both aesthetics of interaction and game feel, we have seen that both are terms with different understandings often confused among users [55] or even by experts [154]. While the field of aesthetics of interaction tries to describe and frame the physical, psychological and motivational aspects of an interaction with a computer [59, 162, 154], the field of game feel focuses specifically in the psychological experience when interacting with a game, either broadly [288] or game mechanic focused [55, 70]. Relating to motion controls, Isbister and colleagues performed a series of studies with the goal to better understand how the amount of movement in motion controlled digital games affected emotional qualities and social connectedness, with results linking movement with arousal [125, 127, 128]. Still, none of these constructs frame the experiential ethos of game playing as an enjoyable one, which could easily be understood as one of the key goals of most videogames. Therefore, the next section reviews prior work on conceptualising enjoyment, from a broad understanding of need satisfaction to a multi-layered game experience.

2.4.6 Enjoyment

We have so far reviewed how engagement, immersion aesthetics of interaction and game feel help describe and frame game motion controls. Still, enjoyment could be assumed to be a key constructs for game creators. It is possible to argue that one of the main reasons why game creators might want to design and prototype motion controls is to create a more enjoyable experience with the game. However, what constitute an enjoyable gesture? And, how do we define what is enjoyment? The latter question needs a broader understanding of how enjoyment can be defined. In this section different understanding of enjoyment as a psychological construct will be reviewed, as it could be viewed as the satisfaction of needs, a result of the flow experience, or as a multifactorial and multi-layered game experience.

2.4.6.1 Enjoyment as the Satisfaction of User Experience (UX) Needs One of the general ways to conceptualise enjoyable experiences is via need satisfaction theories. Hassenzahl (2003) [109] proposed a model of user experience based on product character attributes, which can be pragmatic or hedonic, and reflect the designer's intention and the user's goals and well-being. Vorderer, Klimmt and Ritterfeld (2004) [309] proposed a model of media entertainment based on user and media prerequisites, which influence the user's enjoyment and its effects. Their model can be related to Hassenzahl's (2003) theory of user experience in terms of hedonic and pragmatic attributes [109]. Kim and colleagues (2011) [143] proposed a framework and a method for generating design concepts based on human needs satisfaction, which they derived from previous literature. They compared their needs list with other models, such as Hassenzahl's (2003) theory of user experience (Fig. 10).

Tamborini and colleagues (2010) [289] investigated the role of hedonic and nonhedonic needs in videogame enjoyment, using arousal, absorption, competence and autonomy as measures. They found that both types of needs contributed to the variance in enjoyment in a singleplayer game. Tamborini and colleagues (2011) [290] conducted a second study to examine the effects of hedonic and nonhedonic needs on videogame enjoyment, using arousal, affect, competence and autonomy as measures. They found that hedonic and nonhedonic needs varied depending on the interactivity of the medium. They compared their approach with other models of entertainment that differentiate between enjoyment and appreciation [109, 309]



Sheldon, et al. (2000)	Gaver & Martin(2000)	Jordan (2000)	Hassenzahl (2003)
Autonomy – Independence			
Competence – effectance	To extend knowledge and control; to influence the environment	Psycho	Manipulation
Relatedness – belongingness	Intimacy	Socio	
Influence – popularity		Socio	Identification
Pleasure – stimulation	Novelty, surprise, diversion, mystery	Psycho	Stimulation
Security – control			
Physical thriving –bodily		Physio	
Self-actualizing – meaning	To understand and change one's self,	Ideo	Evocation
Self-esteem – self-respect			
Money – luxury			

Figure 10: Comparison of Need Satisfaction Theories [143]

2.4.6.2 Enjoyment Explained by the Flow Experience In contrast with the need satisfaction approach, Sherry (2004) [261] proposed a theory of media enjoyment based on the flow experience, which occurs when the user's cognitive abilities match the media content message. Sherry (2004) applied the flow theory to media enjoyment, arguing that it depends on the balance between challenge and skills. He built on his previous work on videogame uses and gratifications, and discussed the role of individual and contextual factors in flow. He concluded that flow provides a theoretical framework for understanding enjoyment across different media.

Sweetser and Wyeth (2005) [287] proposed a model of game enjoyment based on flow, which consists of eight elements: concentration, challenge, skills, control, clear goals, feedback, immersion and social interaction. They validated their model on two RTS games and recognised its limitations. They later revised their model for different game genres and noted the difficulty of assessing enjoyment and immersion from observations of the game exclusively.

The GameFlow model is more complex than the previous theory from Sherry (2004), as it takes more elements into account that the skill and challenge from the game. Even though Sherry (2004) select because of allowing concentration, allowing to adjust the challenge to our skills, having concrete goals and providing feedback, the GameFlow model [287] accounts for Immersion and Social Interaction as key elements of the experience. Nevertheless, Sherry's (2004) study offered solely a theoretical explanation for media enjoyment, while Sweetser and Wyeth (2017) offered an applicable framework for expert reviewers.

Still, it is possible to observe that the viewpoint of enjoyment as a result of the flow experience [261, 287] shared certain elements with the need satisfaction literature presented earlier in the text [109, 143, 265, 309]. In both approaches, it is possible to see elements relating to the intrinsic motivation needs of Autonomy, Competence and Relatedness [143, 243, 259, 290] Like the elements of Skills, Controls, Clear Goals and Social Interaction from the GameFlow model [287]. After all, Sherry (2003) studied a selection of uses and gratification in games that led him to apply the flow theory to explain enjoyment a year after [261].

2.4.6.3 Enjoyment as a Multifactorial Game Experience Still, the literature until this point showed several levels of overlap and different categorisations of the experience that might lead to think of enjoyment as a more complex construct with different dimensions. The following literature understands enjoyment as a multifactorial game experience, where intrinsic needs, hedonic needs, flow and different psychological constructs mix.

Poels, de Kort and Ijsselsteijn (2007) [224] proposed a categorisation of digital game experience based on focus groups and existing literature on flow, immersion, enjoyment, social context and negative experience. They identified nine dimensions of game experience, but did not explore their interrelations or variations. They introduced the Game Experience Questionnaire as a tool to measure game experience. Ijsselsteijn and colleagues (2007) [124] further examined the concepts of flow and immersion, and argued for a multi-method, multi-measure approach to evaluate gameplay. They also mentioned the Game Experience Questionnaire as a first step to assess player experience.

To frame the research around the GExpQ, it was important to understand the different ways to empirically measure enjoyment. That is the reason why in a following study, Ijsselsteijn and colleagues (2008) [123] investigated how to measure Player Experience (PX), with enjoyment as one aspect of that measure. They strived to measure PX with a multi-method, multi-measure approach, following the multidimensionality of PX presented in the study of Poels, de Kort and Ijsselsteijn (2007) [224].

In addition, Ijsselsteijn and colleagues (2008) [123] expanded the dimensions of their player experience categorisation to include positive affect as well as negative affect. Similarly, psychophysiological measures might be useful in distinguishing player emotions along the dimensions of arousal and valence. Arousal dimension distinguishes between exciting and dull, and the valence dimension distinguishes between positive and negative emotions, as presented in the well-known research from Russell (1980) [241].

The work from Poels, de Kort and Ijsselsteijn (2007) [224], and Ijsselstein and colleagues (2007, 2008) [124, 123] expanded the conceptualisation of game experience as a multidimensional construct, where the dimensions can be fitted into the categories of enjoyment, flow, immersion, intrinsic motivation and hedonic need satisfaction. This is a combination from the previous research on need satisfaction and flow explained in this thesis, with the addition of immersion. However and in spite of all the work published by Poels, de Kort and Ijsselsteijn (2007), and Ijsselstein and colleagues (2007, 2008), there was not enough empirical data published around the development of the measures and the list of the different multi-measures selected.

Following the critique around the lack of data published, Norman (2013) [204] reviewed two of the papers introducing self-reported measures for PX using digital games: the Game Experience Questionnaire [124] - labelled as GExpQ by Norman (2013) - and the Game Engagement Questionnaire [32] – labelled GEngQ by Norman (2013). The GExpQ measures PX as a game evaluation characteristic, but there is not empirical evidence in its multidimensionality, as was pointed out before. The GEngQ measures engagement as a unidimensional player characteristic and it is backed up by empirical evidence [32]. Nevertheless, the unidimensionality of the GEngQ might be due to the factors selected in its conceptualisation, and unfortunately its usage is associated with anti-game groups [8].

Norman (2013) [204] followed his argument by explaining that the individual characteristics of the player or the game might affect the game experience but none of the measures studied attempt to reconcile the interaction between the two. Therefore, Norman (2013) proposed two new measurement candidates, studying both player and game using immersion as a pathway, named the Immersability of individuals (IAI) and the Imersiveness of Games (ING). The idea of taking into account specific qualities of the game when measuring PX was as well contemplated in the enjoyment research from Sweetser et al. (2017) [286], where the GameFlow measure was proposed to be adapted depending of the genre of the game studied.

As it was presented up until this point in this review the understanding of game enjoyment and game experience is quite dissimilar in the different dimensions that comprises the experience. In order to gain a broader assessment, a systematic review of 87 quantitative studies on the enjoyment of digital entertainment games was revised [183]. The motivation of Mekler and colleagues (2014) [183] started with the lack of a common terminology for discussing and measuring PX and game enjoyment, in a similar manner to the motivation of the work of Poels, de Kort and Ijsselsteijn (2007) [224], and Ijsselstein and colleagues (2007, 2008) [124, 123]. Therefore, Mekler and colleagues (2014) reviewed 87 studies that involved a quantitative measure of enjoyment of some sort. In addition, different terms that do not cover the exact same meaning to enjoyment were discarded, such as *fun, liking or preference* [22, 194]. The review excluded serious games, pervasive games and augmented reality games as well, since they might require their own criteria for modelling enjoyment [130]. Mekler and colleagues (2014) grouped then determinants that potentially predict enjoyment found in the literature into game system, player and context as suggested by the model from Nacke and Drachen (2011) [195]. In addition, the separation of PX into different layers accounting the player and the game is partly aligned with the previous research from Norman (2013) [204] and Sweetser and colleagues (2017) [286].

Interestingly for the purpose of this thesis, intuitive control schemes were found as one of the determinants to predict game enjoyment, since they facilitated feelings of being in-control and self-efficacy [157, 247, 295]. Players enjoyed easy to control interfaces that allowed best performance [36, 166]. The psychological outcomes of feeling of being in-control, self-efficacy and need satisfaction were found as determinant predictors of game enjoyment as well [228, 289, 290].

Mekler and colleagues (2014) then analysed enjoyment in relation to other PX components. Different views on how flow relates to enjoyment were found in the literature. Some authors use the words flow and enjoyment as synonyms [134, 146] or use specifically the GameFlow model when studying enjoyment [287]. Other studies entrusting the GExpQ argue than flow is a dimension of player involvement instead of player enjoyment [84]. Finally, some stated that enjoyment results from the flow experience [314]. Although several studies associated the flow aspects of focused attention [134] and the balance of skill and challenge [146] to explain enjoyment, Shim, Srivastava and Hsu (2011) identified that the balance of skill and challenge accounts partially for game enjoyment. Conversely, Limperos and colleagues (2011) discovered that another aspect of flow –the experience of control – was related to player enjoyment, but not to other characteristics of flow.

In addition, different studies found a positive connection between presence and

Table 3: Game Enjoyment Determinants found by Mekler and colleagues (2014) [183]

Game System	**Player**	**Context**
• Challenge • Player Skills • Motives • Game Outcome (e.g. win/lose. Mixed results) • Intuitive control schemes • Fantasy • Narrative • Avatar resemblance • Identification with the avatar and other playable characters • Sound • Music • Violence	• Player types and motives • Personality traits – Sensation seeking – Self-forgetfulness • Player Gender • Psychological outcomes – Feelings of in-control – Feelings of self-efficacy – Feelings of need satisfaction – Winning, ego-enhancement – Mood repair, recovery experience – Feelings of guilt	• Social Interaction – Presence of other players – Communication between players • Location

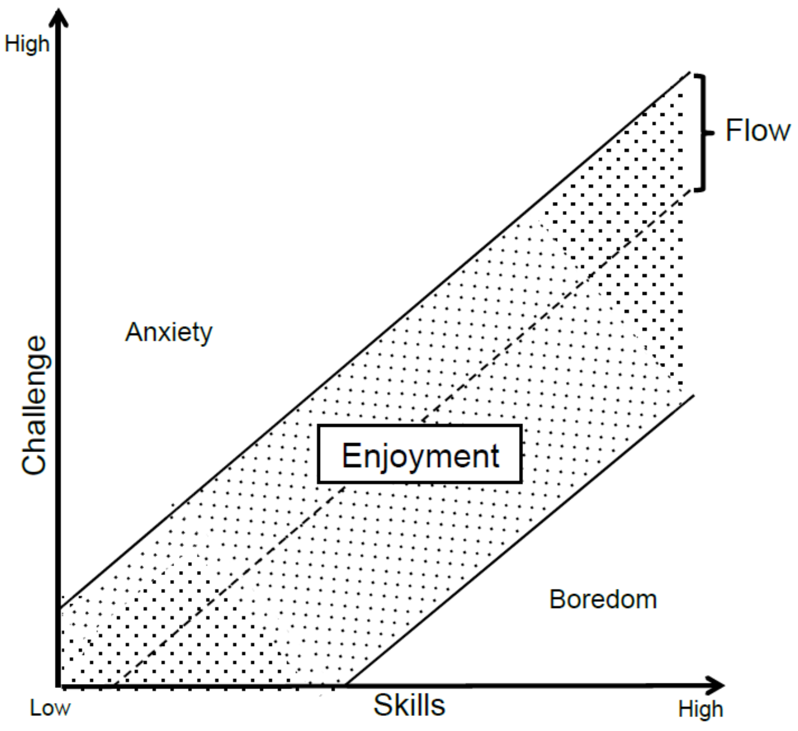


Figure 11: Flow and Enjoyment are similar but not the same. Less demanding gameplay can be perceived as enjoyable [183]

flow [134, 197]. Presence might cause enjoyment indirectly through flow because of the facilitation of cognitive involvement [247, 314] . Rewording, flow encompasses both enjoyment and involvement, but players can experience enjoyment separately from flow (Fig. 11) when skills exceed challenges posed by the game [157, 262]. Immersion was found as well relevant to a good gaming experience, but not to how it relates to cognitive and emotional involvement [131]. Immersion is as well associated with positive affect and competence, most likely due to spatial presence [197].

Mekler and colleagues (2014) [183] then conceptualised game enjoyment based on all the papers reviewed. Enjoyment was often understood as fun, interest and the opposite of boredom, especially when seen as intrinsic motivation [228, 248]. Enjoyment was also conceptualised as positive affect [223], or as a construct with affective and cognitive aspects differentiated between them [69]. Since it is possible to have an overall enjoyable experience while playing a frustrating game, frustration was also proposed as one important concept related to enjoyment and not considering it the antipode of enjoyment [61]. Additionally, enjoyment was often associated with feeling of being in-control [247], competence [133], and improved mood because of need satisfaction [235]. At the same time, feelings of guilt were found to negatively correlate with enjoyment [97, 108, 158]. Mekler and colleagues (2014) summarise their conceptualisation of enjoyment as the combination of positive cognitive and affective appraisal of the game [69], the psychological outcomes of absence of guilt [97, 108, 158] and need satisfaction [235].

Additionally, Mekler and colleagues (2014) uncovered different methodological challenges in their review. Few studies used standard measures and many of the studies with custom scales omitted item descriptions and information on the reliability and validity on the scales. The most used standardised questionnaires were the Intrinsic Motivation Inventory (IMI) [242, 297], the Game Experience Questionnaire (GExpQ) [124] and the self-assessment manikin scale (SAM) [115, 223].

Finally, Mekler and colleagues (2014) called attention to the fact that even though the context surrounding the player experience is important [111, 195], scarce consideration has been paid to how contextual aspects affect game enjoyment (Table 3). Furthermore, Lazarro's 4 keys to fun model [153] linked player, game system and context with emotions such as challenge and *fiero* (pride). Mekler and colleagues (2014) discussed the importance of further studying how game system, player and context evoke these emotions and this impacts game enjoyment. Likewise, research on non-interactive media revealed that negative experiences can often be experienced as enjoyable [194], indicating the necessity of a better understanding in how negative affective experience might impact game enjoyment, and how they relate to need satisfaction and player values.

2.4.6.4 Enjoyment and Game Controls The three prior forms of conceptualising enjoyment (i.e. needs satisfaction, flow or multi-factorial game experience) do not delve deeply into controls. Consequently, we here review literature specifically conceptualising enjoyment from a game controls perspective. Following the idea of how game controls affect the game experience of enjoyment, a study performed in 2011 compared if the naturalness of the controller type affected spatial presence, and if that was related to player enjoyment [265]. The authors compared the experience of the players across a golf game and a racing game; using a gamepad, a motion control, a keyboard, a joystick and a steering wheel. Results did not only find that the more natural the controller was, the higher spatial presence , but also that it is a predictor of enjoyment. Although, is a different study the authors conclude that active videogames can be less enjoyable that more sedentary ones [164]. Nevertheless, another study determines that the more the natural the controller is, the higher the social interaction is and the more enjoyable is playing with a collaborator [19]. For this last study, they compared a gamepad, a motion control, and a bespoke motion control for a music game.

Shafer and colleagues (2011) [257] investigated how motion controls compared to traditional controls in terms of enjoyment, and found that higher levels of interactivity of motion controls were a predictor for enjoyment. Shafer and colleagues performed a follow-up study in 2014 [256] where they included 'props' for certain movement tasks (such as "table tennnis" or "lightsaber dueling") and found that those were the biggest predictors for enjoyment, albeit movement interactivity remained the most important predictor all conditions.

This might indicate that including motion controls ‘for the sake of moving’ might not be such an important design factor as incorporating ‘nuance’ into the movement to benefit interactivity during game creation. Shafer and colleagues (2011, 2014) also looked at how enjoyment and presence were connected, and found that both constructs correlated with movement and interactivity. Since VR it is the current medium for motion controlled games, presence can be a dimension highly affected by the stereoscopic visual effect that VR offers, hence in the following subsection we will look into what is presence and how VR affects presence.

2.4.7 Presence

We have reviewed so far five main experiential qualities: engagement, immersion, aesthetics of interaction, game feel, and enjoyment – to better frame the design of our VR motion control creation tool. Prior work related enjoyment with presence, which is also one of the key constructs in VR. In their review paper, Felton and Jackson (2021) [71] defined presence as “the extent to which something (environment, person, object, or any other stimulus) appears to exist in the same physical world as the observer” [71]. They argue that their rationale goes beyond technology and is framed as a psychological phenomenon disentangled from its cause. In their understanding, presence exists in the psychology of the user, regardless of whether in VR or not, and they give examples of how someone can feel present in a dream, through religious experiences or even while consuming media (ie. movies, books). Felton and Jackson (2021) specifically refer to presence “elicited in response to VR technology” as *virtual presence* [71]. They synthesised two main dimensions from the literature, named spatial presence and social presence. They defined spatial presence as “the subjective sense that one is physically located within the perceived environment and subject to any physical consequences therein” [71] and social presence as “the degree to which another animate entity appears to coexist in the same environment as the user” [71].

For instance, spatial presence would be understood as feeling as if one is really in a different place, and virtual presence would be understood as feeling that someone is there with you. Felton and Jackson (2021) also review the determinants of virtual presence, with head-tracking and haptic cues as the most relevant ones for this thesis. The define head-tracking as “the change in a visual display that corresponds with changes in the location of the user” and haptic cues as “the sense of touch, including the perception of pressure, temperature, vibration, and limb position in space”. There are technological limitations to simulate pressure, temperature, vibration in VR [311], but limb position is simulated via controller tracking [39], hence supporting head and controller tracking are critical to the experience of virtual presence [71].

Additionally, prior research from Slater and colleagues (1998) suggested that body movement was positively associated with virtual presence [268]. Participants that physically crouched and moved in the virtual space prompted by the height of virtual trees reported higher levels of presence than those in the control group. However, later research didn't find significant differences on levels on body movement and virtual presence [12, 269]. Still, it is safe to assume that, depending on the movements performed by users, presence might be affected, hence game creators might want to ensure high subjective feelings of virtual presence with their motion controls.

Skarbez and colleagues (2017) informed their review of presence by Slater's place and plausibility illusions [267, 266]. Mel Slater theorised that what is understood as presence can be represented by two orthogonal illusions: (a) the place illusion which refers to how "real" the environment feels, and (b) the plausibility illusion which refers to how believable the situation is [267]. Skarbez and colleagues (2017) argued that framing the feeling of presence as 'illusions' was benefitial, and actually suggested the alternative term *placeness* for what Felton and Jackson (2021) described as virtual presence [266, 71]. Furthermore, Skarbez and colleagues (2017) also take into accoutn models of deeply related concepts to presence or placeness, such as embodiment.

2.4.8 Embodiment and Embodied Cognition

Related to presence, another highly relevant psychological concept in VR is the sense of embodiment, which is defined as the "representation of a user (also known as *avatar*) withing a mediated or virtual environment" [266]. Kilteni and colleagues (2012) frame the sense of embodiment on "one's own biological body", which grounds the user feelings on their physical body [142]. They further define three subcomponents of embodiments, named self-location, agency and body ownership [142]. Self-location refers to the "relationship between one's self and one's body", agency refers to the sense of having "global motor control" and body ownership refers to "self-attribution of a body... and it implies that the body is the source of experienced sensations" [142]. For instance, a user feeling that they are placed inside a body would be self-location, accepting that body as theirs would be body ownership, and feeling that they can control the body they are in and have accepted is agency.

But prior research showed that users can do more than feel with their bodies, they also *think* with their bodies [148] via processes of embodied cognition [148]. Klemmer and colleagues (2006) discussed how physical bodies shape how users experience, understand and interact in the world and synthesised five themes relevant for interaction design: thinking through doing, performance, visibility, risk and thick practice [148]. In their thinking through doing theme, Klemmer and colleagues discuss how previous research has shown "the importance of physical action as an active component of our cognition" via a shared link of "perception, cognition and action". According to them, users can learn about complex physical or mathematical concepts via direct embodied interaction with tangible systems, such as the Montessori blocks for learning numbers [191] or the tangible illuminating light workbench for learning optical physics [298]. Furthermore, Klemmer and colleagues differentiate between pragmatic and epistemic actions to argue that users think via direct interaction. They use the example of Tetris piece movement, in which pieces can be move pieces pragmatically to align them, or epistemically to understand how different options would work [169]. Therefore, they discuss that the iterative epistemic prototyping process allows creators to think via prototyping, and reason that the form of thinking via doing helps in cognitive processes that could not be produced without "producing a concrete manifestation of her ideas" [148]. Then they introduce their performance theme, which synthesises the notion that artifacts are incorporated into peoples embodied perceptions, "acting through it rather than on it" and expressing tacit knowledge [148]. They argue that there is evidence about the existence of kinesthetic memory, such as bodily remembering how to ride a bicycle or how to play a piano but not being able to reflectively articulate how it is done [279, 148]. They then claim that reflective reasoning is too slow compared to experiential cognition, in which users are able to quickly bodily think in interactive scenarios that make use of body interactions "in the loop" [203, 148] and state that one of the most commercially successful interfaces to leverage embodied cognition are game controllers (beskope action ones, such as flight joysticks or steering wheels).

Kirsh (2013) extended the embodied cognition theory with the insight that bodies can be used as "simulation devices to physically model things" after finding that dancers were able to better learn a complex dance phrase by bodily rehearsing an incorrect performance than by mentally reflecting on the correct performance [145]. He argues that his findings occur because of the ability of people to cognitively "project the structure or idea" via physical movements. He additionally accounts for how objects can be brought into the cognitive process since they can be "absorbed" into the body, and thus enrich interaction design [145].

Therefore, the untackled benefits of embodied cognition inspired further research into embodied interaction design. In the next section, we introduce previous literature on embodied interaction design and discuss the most relevant work to our thesis.

2.4.9 Summary

This section of the background showed how diverse the conceptualisation of experiential qualitites of movement are, and open questions of better understanding how to frame and describe motion interactions in digital games, as well as how changes in these interactions affects the game feel of specific motion game mechanics. In addition, game creators might prioritise other constructs such as flow, immersion, engagement or enjoyment to affect the "feeling" of the motion controlled game and the different aesthetic qualities they might want to deliver to the player.

Since motion controls are predominantly used in VR gaming nowadays, presence is another key construct to consider when creating motion controlled videogames, as body movement might affect the sense of presence in the virtual environment.

Furthermore, we introduced theories of embodiment that tackle how users ground their feelings in their bodies. Finally, we covered theories of embodied cognition that argue that users not only *feel* with their bodies, but are also able to *think* with their bodies.

We have reviewed how motion controls can elicit a diverse set of experiential qualities grounded in movement [288, 40, 154, 125, 257] and the body [267, 71]. Therefore, any VR motion control design solution should accommodate the expression of such rich intent. Furthermore, we found how there aren't standard game motion control design principles [202, 263, 46, 98], and existing game engine implementation solutions are focused on object-based interactions [301] or inflexibly support movement [54]. Therefore, in the following section we will review broader HCI design methods focused on movement with a rich and bodily grounded experiential perspective.

2.5 Embodied Interaction Design

2.5.1 Introduction

We have previously seen how there is a lack of movement-focused standard design methods for game motion controls. We have also seen how rich and bodily grounded motion control experiential qualities can be. In this section, we review HCI literature investigating embodied interaction design, which is an approach that suits our motion control design problem. Embodied interaction is an approach that emphasizes the role of the human body and its interactions with the environment in supporting bodily processes of interaction design [175]. This design methodology is grounded in the theory of embodied cognition, which posits that human cognitive processes are deeply rooted in the body [145]. Embodied interaction design aims to create more natural, intuitive, and engaging interfaces by incorporating principles from embodied cognition theory into the design of human-computer interaction systems [175].

2.5.2 Creators need to move to design

One of the key ides to understand embodied interaction design is that movement is itself is part of the design process. Klooster and Overbeeke (2005) [149] and Hummels and colleagues (2007) [121] suggested design tools to support expressiveness of designers, users and products based on the principle that designers should master movement knowledge and sensibilities to actually design. Klooster and Overbeeke (2005) [149] developed the choreography of interaction framework, which described movement as a "trinity" of entangled pivots (physical involvement, dynamic quality and expressed meaning). Reflecting on the choreography of interaction and other methods, Hummels and colleagues (2007) [121] synthesised seven guiding principles for movement-based interactions: (1) meaning through interaction, (2) richness of interaction, (3) design by moving, (4) support for movement, (5) research by doing, (6) educate through and for movement, and (7) design for diversity.

2.5.3 Describing movement

Prior work has focused on framing the movement interactions between users and computers. The *tangible interaction* framework presents a set of four overarching themes (tangible manipulation, spatial interaction, embodied facilitation, expressive representation) that integrates the body, its movements, and tangible interfaces within physical and social contexts to describe interactive social spaces and artifacts [118]. The *multimodal interaction space* framework has a focus on levels, modes and senses to describe movement mediated by technology [24]. The framework developed by Eriksson and colleagues (2007) [67] uses three main concepts (space, relation and feedback) to articulate movement interactions encompassing body parts, shape, position, and orientation to articulate and describe movement within camera-tracking applications [67]. The *kinesthetic interaction* framework described movement interactions via three overarching design themes (Development, Means and Disorder) and seven design parameters (engagement, sociality, movability, explicit/implicit motivation, expressive meaning and kinesthetic empathy) [78]. For instance, in fig. 12 we can see that the Nintendo Wii Tennis game, where users play tennis by physically swinging the Wii controller as if it was a tennis racket, can be described by the theme kinesthetic means (i.e. playful kinesthetic activity) and the kinesthetic parameters engagement (i.e. it is an engaging activity), sociality (i.e. fosters social play), explicit motivation (i.e. the game has explicit rules) and expressive meaning (i.e. the physical-virtual mapping is a congruent fit).

2.5.4 Mapping movement interactions

How movement interactions can be mapped onto user inputs and computer outputs is another field in which frameworks have been developed. The *interaction frogger* framework examines the interplay between the actions of individuals and the functional capabilities of products, leveraging both inherent and augmented information, with the goal of enhancing the range of actions with perceptual motor abilities of individuals in tangible interaction [316]. The interaction frogger framework can be seen in figure 13 is constructed around coupling actions with types of information that computers can offer (i.e. inherent, augmented or functional), and each coupling can six aspects between input/output (time, location, direction, dynamics, modality and expression). Benford and colleagues (2005) [14] created a framework around the interplay and tensions of three reflective principles (i.e. expected movements, sensed movements and desired movements) when describing movement inputs and computer outputs [14]. The *sense-making experience* framework was developed to tackle the design of sensor-based movement interactions via

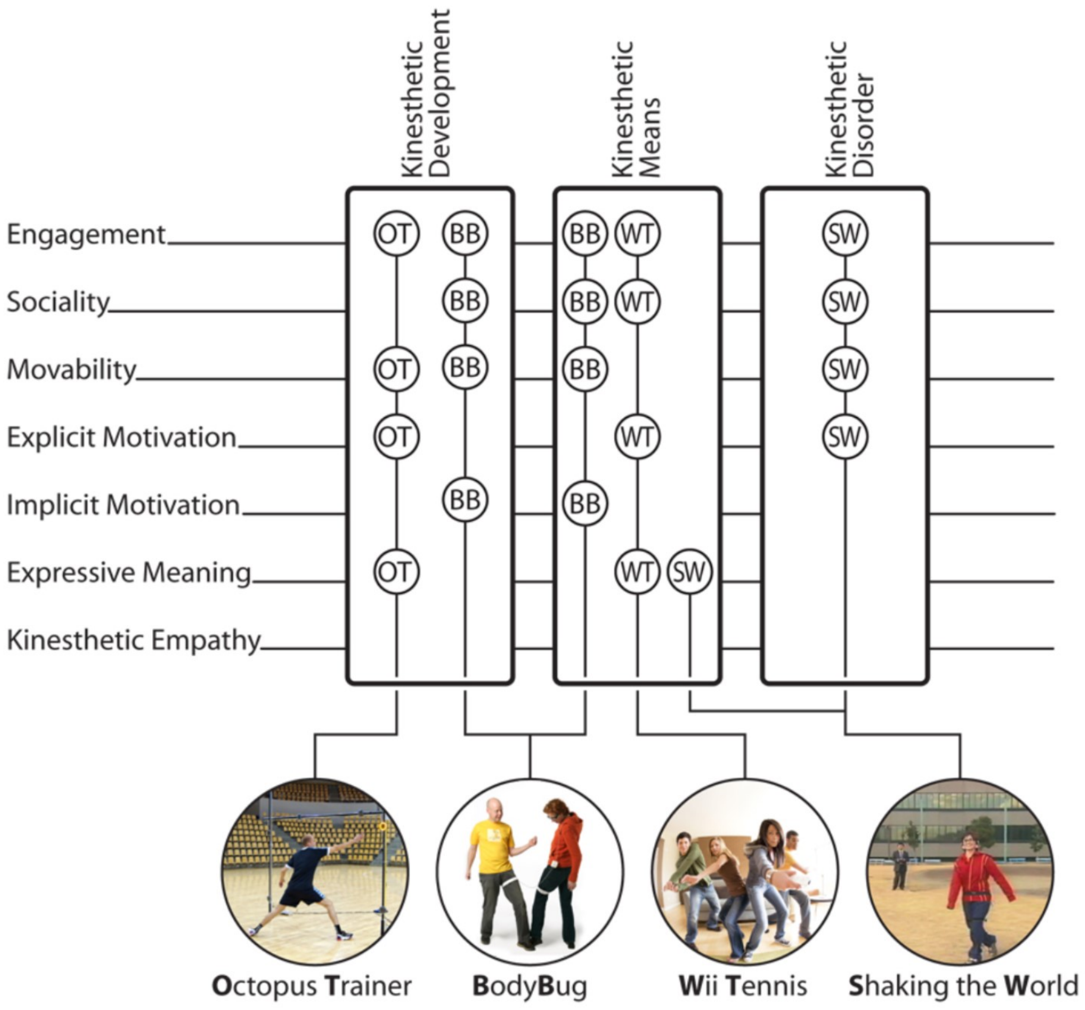


Figure 12: The Kinesthetic interaction conceptual framework from Fogtmann et al. (2008) [78]

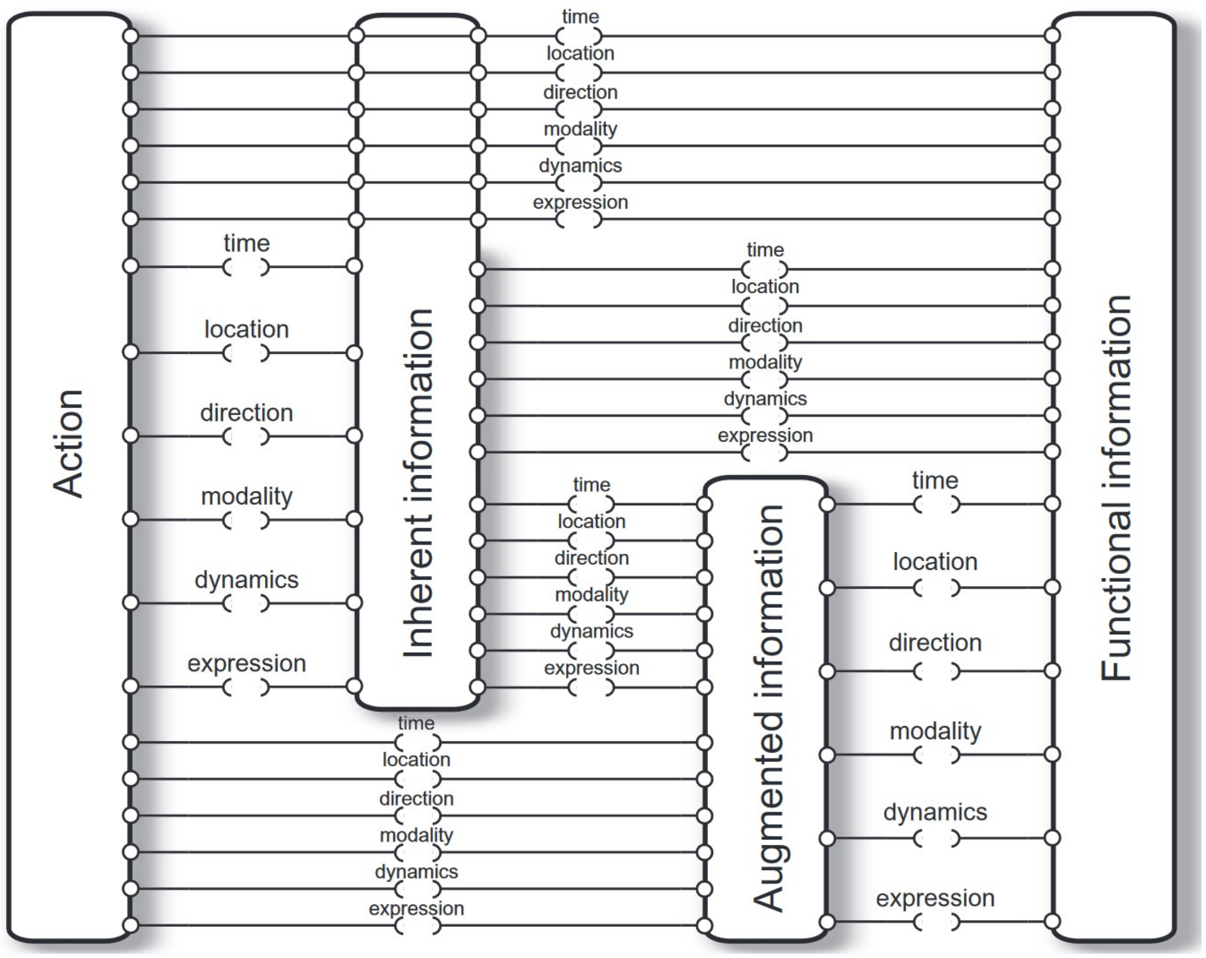


Figure 13: Interaction Frogger framework describing couples of human actions and information offered by a computer [316]

transforms, which are the authors terms for describing pairs of human input and world outputs [239]. Rogers and Muller (2006) describe their sense-making experience framework from the lenses of transforms, and refine three user cognitive processes that tackle inputs/outputs (i.e. perceiving, understanding and reflecting). They argue that novel transforms can be created by considering uncertainty and unexpectedness as parameters of the movement interaction [239]. They also reason that there are three sensor properties when coupling human movements with computer outputs (i.e. discrete-continuous, precision, explicit-implicit) [239].

2.5.5 The first-person subjective body

All of the previous frameworks aim to help design more intuitive and natural embodied interactions. However, Norman (2010) claimed that the term "natural" has been overused when referring to the benefits of embodied interactions [202]. He firstly stated that gestural interfaces weren't new to the decade of the 2010's (shortly after the appearance of the mass-marketed Nintendo Wii, Playstation Move and

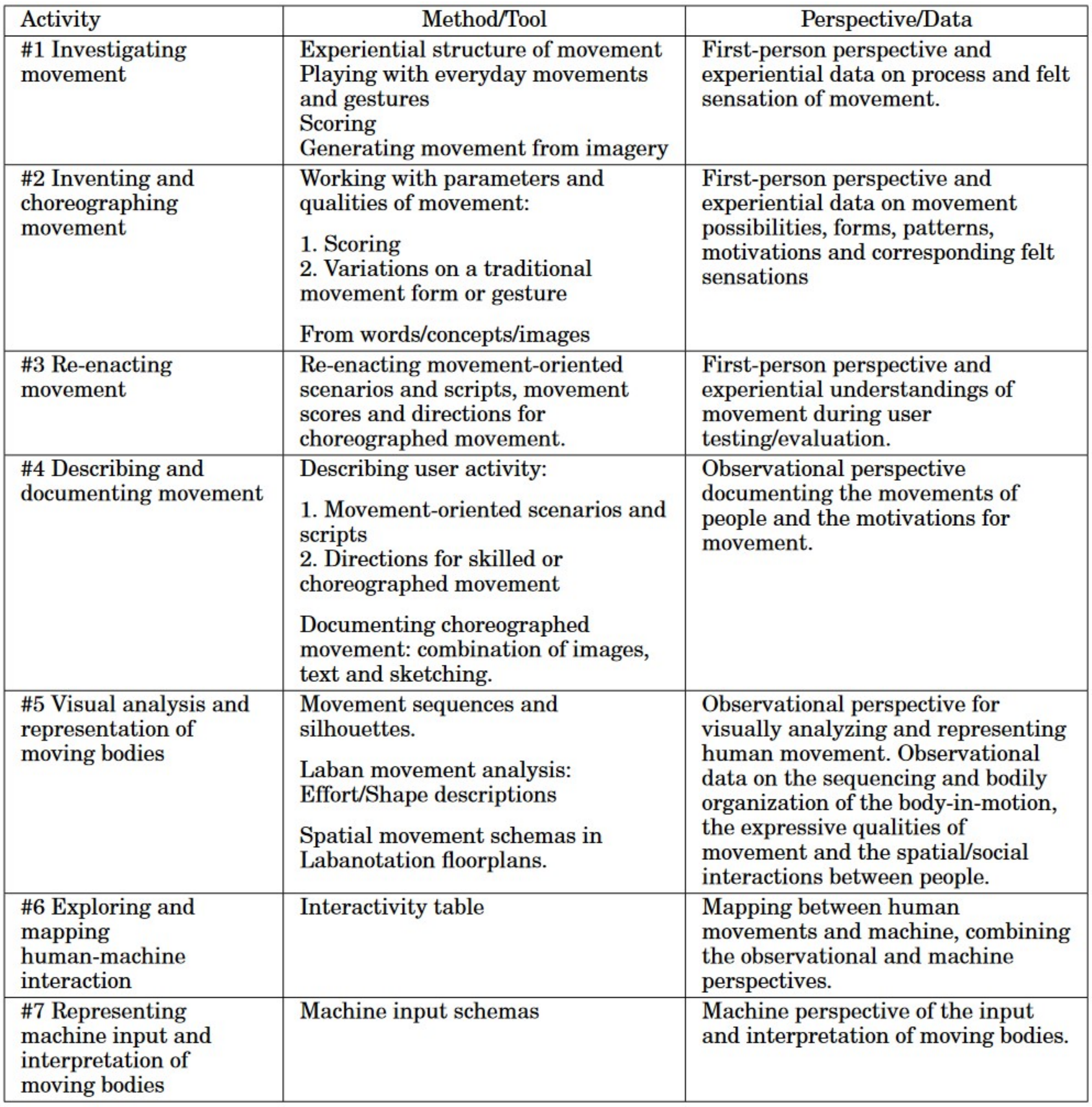
Kinect Controllers [272, 265]) and that known problems with gestural interactions prevailed back then. He illustrated such "naturalness" problem via the Wii bowling game, in which the natural interaction was swinging the motion controller forward and release gripping pressure to throw a virtual bowling ball, but this caused the unintended problem of some controllers flying away from users' hands and damaging their TV displays. He made the point that while gestural purists would claim that the problem was in the presence of the controller or the strap instead of in the gesture, he saw very challenging to map the "release grip" input without the presence of any controller [202]. He reasoned that gestures and interactions are not "natural" and instead they are "learned" via social conventions, and illustrated this with the difference of nodding between western and indian cultures or the apparent naturalness or pinching to zoom on a screen [202].

Gillies and Kleinsmith (2014) [94] reasoned from Norman's argument that bodily interfaces and traditional interfaces exhibit a distinct contrast. The authors reasoned from the reality-based interaction point of view that Jacob and colleagues (2008) introduced, where bodily interfaces and other emerging forms of interaction are effective by utilising a different range of our inherent skills than traditional WIMP-based GUIs. While a GUI draws on our skills in manipulating external visual and symbolic representations, bodily interfaces employ skills related to body and environmental awareness. Therefore, Gillies and Kleinsmith (2014) [94] reason that this difference occurs because of the non-representational, sensorimotor nature of bodily interactions, which contrasts with the inherent symbolic characteristics of GUIs. Hence, natural user interfaces would only be "natural" if they account for such non-presentational and sensorimotor grounding.

O'hara and colleagues (2013) extended Norman's argument by differentiating between representational concerns away from interactional concerns [207]. With representational, O'hara and colleagues (2013) refer to the positivist view that has been prevalent on HCI work that motion interfaces are inherently more natural because they are closer to the "real-world" and hence can distill some universal gestural vocabulary that, with enough research, could be represented and modeled by technology. Instead, they position themselves away from the representational concern to adopt Merleau-Ponty's (1968) phenomenological perspective in which actions could be performed from the *objective* body (i.e. third-person, abstracted and representable) or from the *lived* body (i.e. first-person, subjective and personal) [185, 184, 207]. Furthermore, they reason from a Wittgensteinian (1969) and ethnomethodological perspective that the *lived* body is affected by the social activities that construct one's individual experiences [317, 207]. Therefore, O'hara and colleagues claim that separating what is really "natural" about a motion interface is not how it is engineered to be used, but rather who uses it and the way the it is used in context [207].

Loke and Robertson (2013) [161] introduce a movement-based design methodology that is directly inspired by prior phenomenological work from O'hara et al. [207] and Svanaes [284]. Loke and Robertson criticised the third-person focus of prior embodied interaction conceptual design frameworks from the 2000 to 2010 decade[118, 24, 67, 78, 240] and argued that, in order to innovate in movement-based interaction design a first-person perspective focused on “the how of working with the moving body” should also be taken into consideration [161]. They take inspiration from principles of “defamiliarisation” of the subjective lived experience into free variations, a technique known as “making the familiar strange” or “making strange” [161]. Hence, Loke and Robertson suggests the methodology *moving and making strange*, which is structured around three core perspectives: the observer (i.e. third person *human* perspective), the mover (i.e. first person *human* perspective), and the machine (i.e. third person *computer* perspective). They then suggests that these three core perspectives are to be support the execution of seven activities during embodied interaction design (see Table 4).

Svanaes (2013) [281] suggests a different vocabulary when studying embodied interactions, grounded in three concepts that he has been researching on since 1993 from a phenomenological perspective: the *feel* dimension of embodied interactions [284], the *interaction gestalts* [282], and the *kinaesthetic thinking* [283]. The feel dimension encompasses the overall “sum of stimuli in the visual, auditory, tactile, and olfactory sense modalities” when performing a particular action with a particular interface [284]. The interaction gestalts are the “atomic percepts in the kinaesthetic sense modality” [282] and Svanaes stresses that gestalts are complete behaviours instead of symbolic action/reaction representations of interaction [281]. He argues that the gestalts are understood by users in the feel dimension of embodied interactions via kinaesthetic thinking, which represent the “experiential wholes” and “direct cognitive operations” with the physical environment [283, 281]. Kinaesthetic thinking allows users to compare interaction gestalts between them, akin to how visual gestalts of faces can be compared via visual thinking [281]. The author illustrates his terminology with a driving metapor: driving a car is experienced via the feel dimension, and kinaesthetic thinking allows users to understand the changing gear gestalt [281]. Svanaes argues that while kinaesthetic thinking can be used to describe embodied reasoning processes, users can also have the ability to abstract movements away from concrete gestalts. Therefore, he introduces a new term: *kinaesthetic creativity*, which he defines as “the active use of the body through abstract movements to explore possible futures” and discusses that body-based interaction design can be used for both concrete reasoning and alternative futures explorations [281]. He illustrated how kinaesthetic creativity can be useful in embodied interactions design via two design-through-enactment studies [285], where participants mixed their lived and abstracted bodies with role-playing exercises to generate possible futures. The first study presented in his 2013 paper [281] involved a workshop in a hospital setting to ideate useful interfaces that didn’t exist (in this case, a tablet GUI to read tests results and medical history); the second study presented involved a participatory workshop using disabled Wii remotes with physiotherapists, to ideate movement-based serious games that did not exist (Fig. 14) [281].



Activity	Method/Tool	Perspective/Data
#1 Investigating movement	Experiential structure of movement Playing with everyday movements and gestures Scoring Generating movement from imagery	First-person perspective and experiential data on process and felt sensation of movement.
#2 Inventing and choreographing movement	Working with parameters and qualities of movement: 1. Scoring 2. Variations on a traditional movement form or gesture From words/concepts/images	First-person perspective and experiential data on movement possibilities, forms, patterns, motivations and corresponding felt sensations
#3 Re-enacting movement	Re-enacting movement-oriented scenarios and scripts, movement scores and directions for choreographed movement.	First-person perspective and experiential understandings of movement during user testing/evaluation.
#4 Describing and documenting movement	Describing user activity: 1. Movement-oriented scenarios and scripts 2. Directions for skilled or choreographed movement Documenting choreographed movement: combination of images, text and sketching.	Observational perspective documenting the movements of people and the motivations for movement.
#5 Visual analysis and representation of moving bodies	Movement sequences and silhouettes. Laban movement analysis: Effort/Shape descriptions Spatial movement schemas in Labanotation floorplans.	Observational perspective for visually analyzing and representing human movement. Observational data on the sequencing and bodily organization of the body-in-motion, the expressive qualities of movement and the spatial/social interactions between people.
#6 Exploring and mapping human-machine interaction	Interactivity table	Mapping between human movements and machine, combining the observational and machine perspectives.
#7 Representing machine input and interpretation of moving bodies	Machine input schemas	Machine perspective of the input and interpretation of moving bodies.

Table 4: Summary of activities and perspectives in the moving and making strange methodology [161]

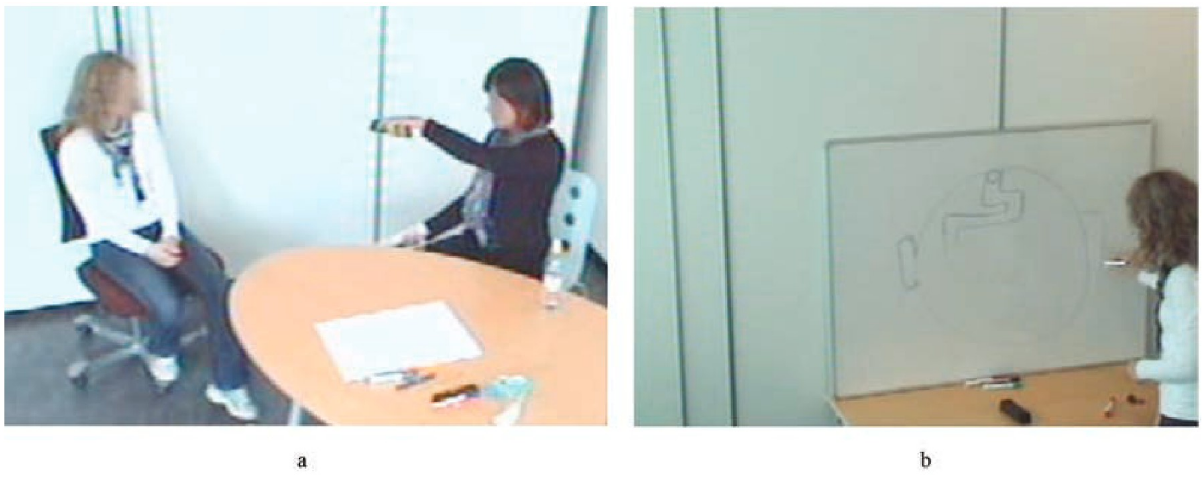


Figure 14: A design-through-enactment workshop from Svanaes (2013) [285, 281]. (a) Participants use kinaesthetic creativity to ideate alternative Wii games via direct first-person lived movement-experience. (b) Participants abstract their lived experience into a whiteboard sketch describing their concrete idea, which is what they "saw" while enacting physically.

2.5.6 Methods and techniques to design

Other authors studied 'bodystorming' methods to explore creativity methods from a first-person lived body experience [212, 246, 174, 23, 116]. With bodystorming, we refer to a creativity-through-enactment activity in which users explore ideas from a whole body first-person perspective. Oulasvirta and colleagues (2003) [212] decided to explore bodystorming as an creative ideation method for ubiquitous computing. In ubiquitous computing the context of where the device is located and how it is on-site plays a key role, hence why bodystorming becomes a relevant method. Bodystorming in this paper is understood as "brainstorming on a context-aware physical location with potentially some acting-out activities when required" [212]. In the motivation, the authors compare bodystorming with brainstorming as exemplars of user-centered design processes (althought they briefly mentioned Contextual Design, Scenario-Based Design, FACE UI Design and others). There is a common flow among all these design methodologies involving three steps: "(1) observation of user activities; (2) documentation of the observations; and (3) design based on the documentation". Authors motivate how stages (1) and (2) are solely meant to produce a document that can be later on complicated and biased, therefore failing to communicate its findings during the design stage. Bodystorming might help since it could allow participants to better 'experience' the context. Authors hypothesise that bodystorming could potentially (1) take less time to produce outputs, (2) result in better designs, (3) provide better understanding of contextual factors and (4) give immediate feedback on-site. These assumptions are motivated by previous research showing how externalising representations and contextual cues can reduce cognitive load, better recall ideas and memories, and direct attention to relevant features. In their study, Oulasvirta and colleagues (2003) compared 4 bodystorming case studies with different settings between them and between another 4 brainstorming case studies with similar settings by looking at how the variations affected the “session and generated innovations”, “the quality and quantity of generated ideas” and “comparing expert sociologist’s comments of the acquired results”. The authors found out that the novelty of bodystorming technique required participants to be trained during several sessions for participants to grasp the methodology, and after comparing all case studies they didn’t found significant qualitative differences between traditional brainstorming and the bodystorming technique they used. Yet, they especulated that, from their observations, bodystormed experiences might be better remembered and could prove useful in the long term, thus “inspiring researchers to get familiar with new contexts” [212]. Schleicher and colleagues (2010) [246] argued that a rework of the bodystorming method needs to be done, addressing the problem that the end-focus shouldn’t be on the final idea generated but rather on the physical experience behind the enactment of a situation. Their goal is to focus on tacit knowledge expression via participatory physical level of experience. They name their variation of bodystorming “embodied storming”, which focuses on “scenarios” more than enacted personas. Schleicher and colleague’s (2010) reason that embodied storming helps creating themes via tacit knowledge translation into rapid communication and social “generate-do-learn” cycles [246]. They also consider the usage of props and suggest that they can have “feelings, thoughts, and the ability to speak” [246]. Höök and colleagues (2018) [116] researched *somaesthetic interaction design*, a design methodology focused on “on making people more aware of their felt bodily experiences”, with term “soma” refers to the “ the unity of mind and body, intellect and experience”, while “aesthetics” refers to subjective appreciation. The authors argue that it is challenging to incorporate aesthetically pleasing designs into products that are meant to be used with the human body, with many poorly designed products it is difficult to create a good bodily experience. Additionally, it is difficult to articulate what constitutes a good or bad bodily experience, which poses methodological challenges. Höök and colleagues (2018) synthesise three main insights from their somaesthetic interaction design journey: (1) attaining somaesthethic skills (i.e. mastery of lived bodily practices), (2) somaesthetic brainstorming (i.e. the methodology affected the way ideas are generated), and (3) the materials in the design process (i.e. iteratively exploring digital and physical materials). Márquez Segura and colleagues (2016) [174] proposed a new method called *embodied sketching*, which has the purpose of designing enjoyable movement interactions for a product based on play, even before the product is firstly prototyped. The authors reason that motion-tracking technology should be bodily sketched to avoid negative applications and instead come up with positive applications before creating the technology. They illustrate their position with motion-controlled game experiences, arguing that they could fail engage players long-term as gesture interactions could not offer enough play after the initial novelty phase, because of treating the motion-control design problem an engineering one. Márquez Segura and colleagues (2016) cleverly illustrate this problem with the a quote on the Playstation Move controller: “[. . .] great tech, probably not so great applications so far [. . .]” [174]. The authors synthesise five principles from their embodied sketching method: “i) an activity-centred embodied approach to ideation, ii) the use of the

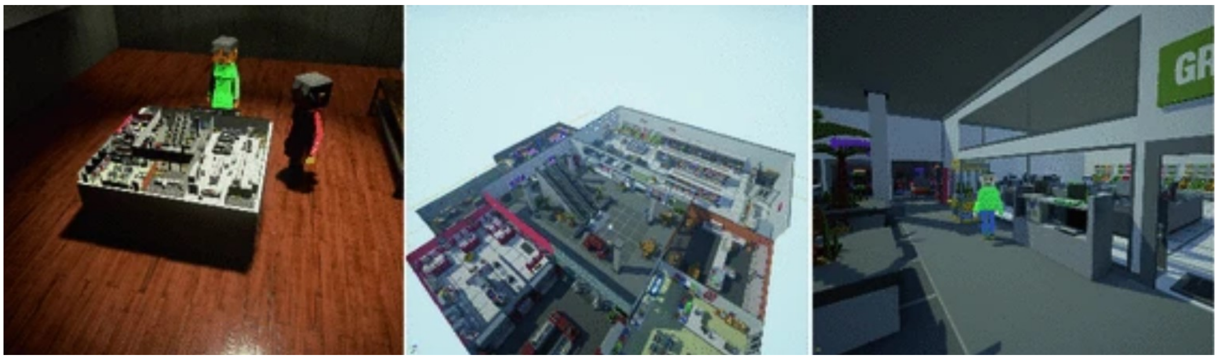


Figure 15: Prototype screenshots from the virtual bodystorming study [23]

complete setting as a design resource, iii) the physical and hands-on engagement of designers with a non-scripted activity, iv) the use of movement and play both as method and goal, and v) the facilitation of a sensitizing and design conducive space" [174]. Boletsis and colleagues (2017) explored a VR-centric bodystorming methodology, named *virtual bodystorming* [23]. They motivate their work from the perspective of traditional design ideation methods, which inform about specifities of interactions with users and service in a limited manner. Therefore, there is a need for service design ideation methods that can convey the information better. The authors hypothesise that bodystorming in VR might help overcome the limitations between the service environment and its prototype, thanks to the capacity of recreating physical scenarios in 3D and simulating acted-out interactions in a collaborative fashion (Fig. 15). These immersive and networked features could help the service designer in extracting and communicating relevant user feedback [23].

More recently, the challenges behind the COVID pandemic stimulated embodied interaction research on remote scenarios. Ferran Altarriba and colleagues (2022) [18] introduce their *designerly tele-experiences* method that focuses on iterative remote co-design between designers and stakeholders. The designerly tele-experiences allowed participants that were co-located in the same physical space act as co-designers by interacting with playable digital prototypes that were remotely controlled by the researchers (Fig. 16)

Weijdom (2022) [315] proposed the *performative prototyping* method for collaborative mixed reality environments. The authors explored how bodystorming techniques in a socialVR scenario could benefit from a digital puppeteering (i.e. blending physical and virtual props, see Fig. 17) and Wizard of Oz [56] techniques, which "asks for an *embodied awareness* by the operator of its *mediated performance* of objects and events in the MR environment in response to other participants"[315]. Weijdom (2022) explains that the performative prototyping technique is done both from an *inside-out* phenomenological appreciation and an *outside-in* somaesthetic evaluation [315]. The author chose to study his methodology in socialVR because "the capability to do the designing, coding, prototyping, and sharing from within its VR environment allows for a continuous flow of embodied design strategies and experiential learning", and additionally it allows "to scale oneself in relation to a

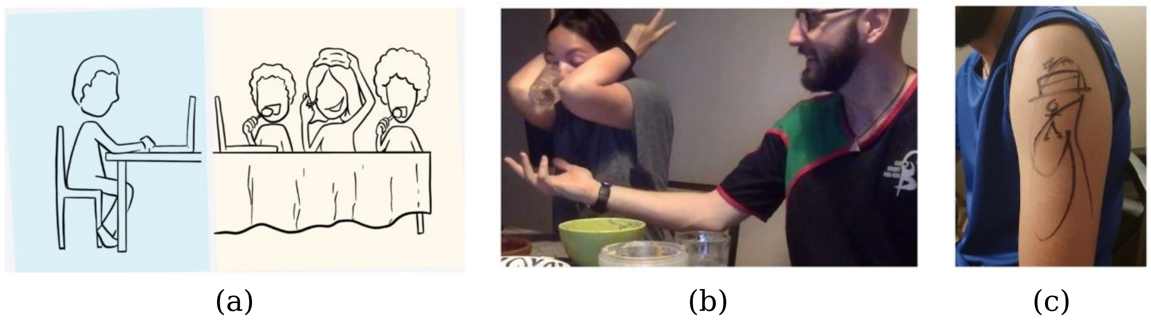


Figure 16: The designerly tele-experiences method [18]. (a) Sketch of the remote methodology, in which the researcher controls a prototype remotely and remote participants bodily co-design while co-located. (b) Participants playfully co-designing with props in response to one of the prototypes. (c) Participants marking body parts of other co-located peers as ideated effects from an interaction with the prototype.

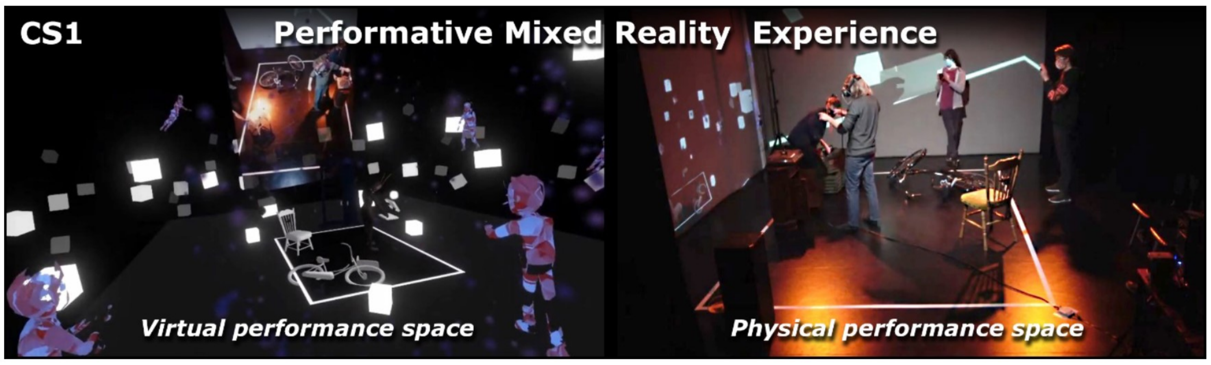

CS1
Performative Mixed Reality Experience
Virtual performance space
Physical performance space

Figure 17: In performative prototyping, participants bodystorm and merge physical props with virtual representations that are controlled via puppeteering or wizard of oz techniques [315]

virtual environment". The performative prototyping method follows the stages of ideation, development, and testing and suggests that bodystorming, Wizard of Oz, puppeteering or role-play can be used in any of the stages as designers see fit [315].

Furthermore, Françoise and colleagues (2017) [82] focuses on the importance of technology-driven kinesthetic awareness in embodied interaction design. The authors explored how their interactive sound installation that "generates a continuously evolving sound environment, in response to participants' micro-movements and muscular activity". The authors emphasize that, while methodologies like embodied sketching [174] or somaesthetics interaction design [116] are powerful to generate embodied interaction designs, implementation of such ideas is essential for continuous interaction. They claim that "moving *in interaction* is fundamentally different from *sketching* or *simulating* interactions, as it positions the user withing a singular action-perception coupling" [82]. The authors reason the importance of designing "inside the feedback loop" of the technological intervention to explore variations in implementation and mapping. They acknowledge that there is often no single criterion for finding the optimal design, and designers must embrace explo-

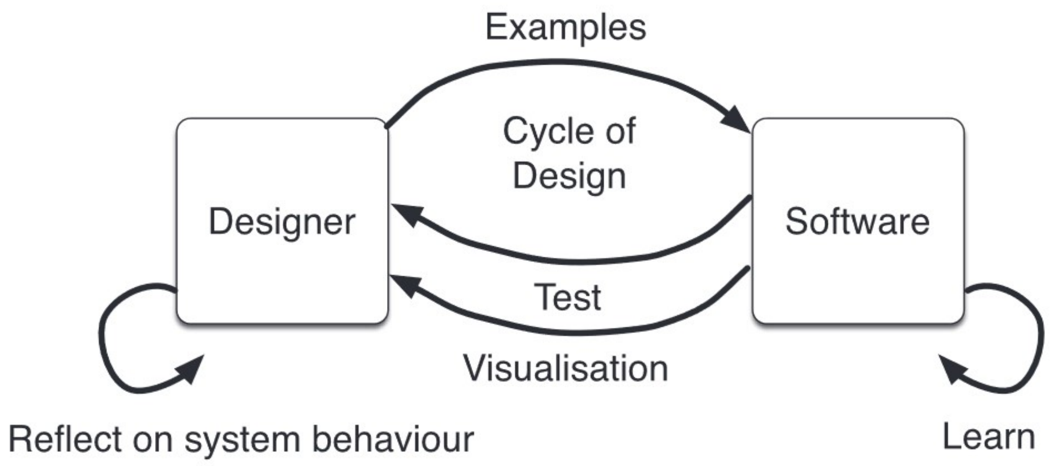


Figure 18: The reflective design loop model for embodied interaction design and implementation of interactive machine learning systems [93].

ration and deep attention, citing qualitatively perceived “Aha moments” to uncover singular experiences within the design space.

Gillies (2019) [93] agrees with Francoise and colleagues (2017) [82], and reasons there isn’t a similar experience to a paper prototype that could translate the feeling of enacting movement interactions and receiving immediate computer feedback. He suggests that movement interactions can be classified in three major styles: (1) object focused (i.e. tangible user interfaces in which the focus is on the object rather than on the movement), (2) direct mapping (i.e. virtual tangible interfaces where a virtual action can be tackled via different forms of undefined movements, such as pressing a virtual button), and (3) movement focused (i.e. interaction design around specific body movements that don’t rely on objects). Gillies (2019) argues that machine learning techniques can be applied to successfully tackle movement-focused interaction designs following an interactive human-in-the-loop approach. He reasons that fast interactive loops of movement-to-machine-feedback allow for embodied reflection that positively affect the interaction being designed and implemented [93]. In this manner, he suggests that a movement interaction designer can follow iterative loops of performance, machine feedback, human reflection, and machine learning refinement (Fig. 18).

2.5.7 Summary

This section of the background reviewed embodied interaction design theories and methodologies grounded in the body, where designers benefit from thinking with their bodies to design. We reviewed theories of the first-person *lived* body discussing how the body can be used for both reasoning and creativity, and then introduced embodied ideation methodologies, such as bodystorming [246], embodied sketching [174] and somaesthetic interaction design [116]. Finally, we reviewed literature arguing that embodied interaction design processes for interactive technology should be accompanied by implementation of the ideas generated to foster embodied creative reflection [82, 93].

Therefore, we have seen that in this section different methods of embodied interaction design and reasons that suggest a deep integration between embodied ideation, design and implementation in an iterative fashion. Furthermore, interactive machine learning seems to be posed as a solution to explore for game motion controls interaction design. In the next section, we will introduce the concept of interactive machine learning and prior research in the field.

2.6 Interactive Machine Learning

2.6.1 Introduction

As described previously in the thesis, Interactive Machine Learning (IML) can be a potential solution to support game creators in designing and implementing more nuanced motion controls. There is room for improving the player experience of virtual reality games because of the undesirable tracking quality of motion controllers [258] and the limited interactions that existing tools in game engines allow [1]. These issues can create friction during motion control scheme development and lead to a poorer player experience.

Traditional gamepads and keyboards are as well limited in the interaction range they offer, but they still allow for a discrete remapping of control schemes when creators find that their controls aren't working well. Joysticks can support a somewhat richer finger movement space [114], but they can't be compared with the richness of human body motion [91].

Additionally, previous research has highlighted how existing methods of embodied interaction design, such as bodystorming [23], embodied sketching [174] or somaesthetics interaction design [116] require marrying design and implementation loops to benefit from embodied reflection [82, 93].

Hence, there is a need for better embodied motion control prototyping pipelines that might might potentially improve the player experience by broadening the richness of interactions and making use of cycles of design and implementation. Interactive Machine Learning then offers a solution to these problems by allowing game creators to prototype gestural interactions via direct human movement examples to tailor the experience to their needs, therefore better representing gestural intent with current technologies [63, 93]

2.6.2 Related IML Work

The field of Interactive Machine Learning (IML from now on) is relatively new. The term was first introduced in 2003 by Fails and Olsen (2003) [68]. Dudley and colleagues (2018) [63] define IML as the "interaction paradigm in which a user or user group iteratively builds and refines a mathematical model to describe a concept through iterative cycles of input and review". Importantly, not only does user input shape the model and its subsequent behaviour: users modify their behaviour in response to the system output as well [68]. IML thus provides an interaction loop where the user is heavily involved, steering the feature space, learner model or training and evaluation data sets in a desired direction after every iteration [6, 63].

Amershi and colleagues (2014) [6] performed a literature review that collected several IML case studies that show how some first attempts fail to account for the user and how there is a need for new learning systems to interact with the end-users. IML introduces "rapid, focused and incremental model updates" [6], that allows users to perform small changes on a particular aspect of the model and immediately obverse the effect of the update (Fig. 19). These characteristic of the IML workflow allows non-expert users to explore the model space and drive the system towards an intended behaviour, reducing the need of expert supervision [6]. Amershi and colleagues (2014) also comment that there exists a tight coupling between the user and the system because of the rapid iterative IML workflow. Amershi and colleagues illustrates as well the fact that studying user interaction can lead to an improvement of classical machine learning algorithms and novel IML systems [6]. This is in line with later research from [92] that points to different traditions in the evaluation of models and learners between the HCI and machine learning communities. HCI research can benefit from the use of representative datasets and structured tasks, as it already happens in traditional machine learning experimentation; and machine learning research can improve by performing user studies in the style of HCI investigation [6, 63].

Videogames then become an interesting test environment where it is possible to explore how the user and the learner affect each other and how tight their coupling is during the experience. It is as well a potential powerful answer to the points raised by previous research regarding the need of user studies in machine learning research and regarding the use of standardised methods – with the right set of structured tasks [6, 92, 63].

Based on a literature review by Dudley and colleagues (2018) [63] , a common IML workflow involves six steps: (1) feature selection, (2) model selection, (3) model steering, (4) quality assessment, (5) termination assessment, and (6) transfer. In Dudley and Kistensson's (2018) opinion, the model steering task is the core activity of the IML workflow, and the one where the user will most likely spend most of the time seeking to steer the model behaviour to a desired one. Fiebrink and colleagues (2011) [73] comment that model steering can as well entail defining the

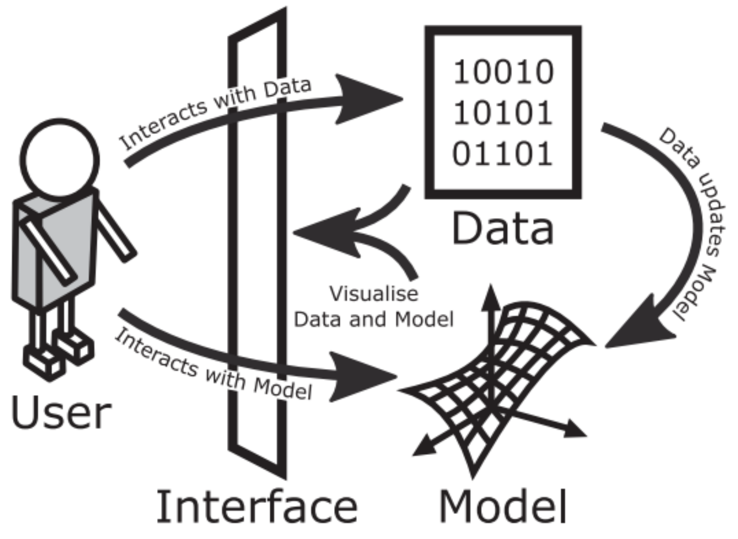
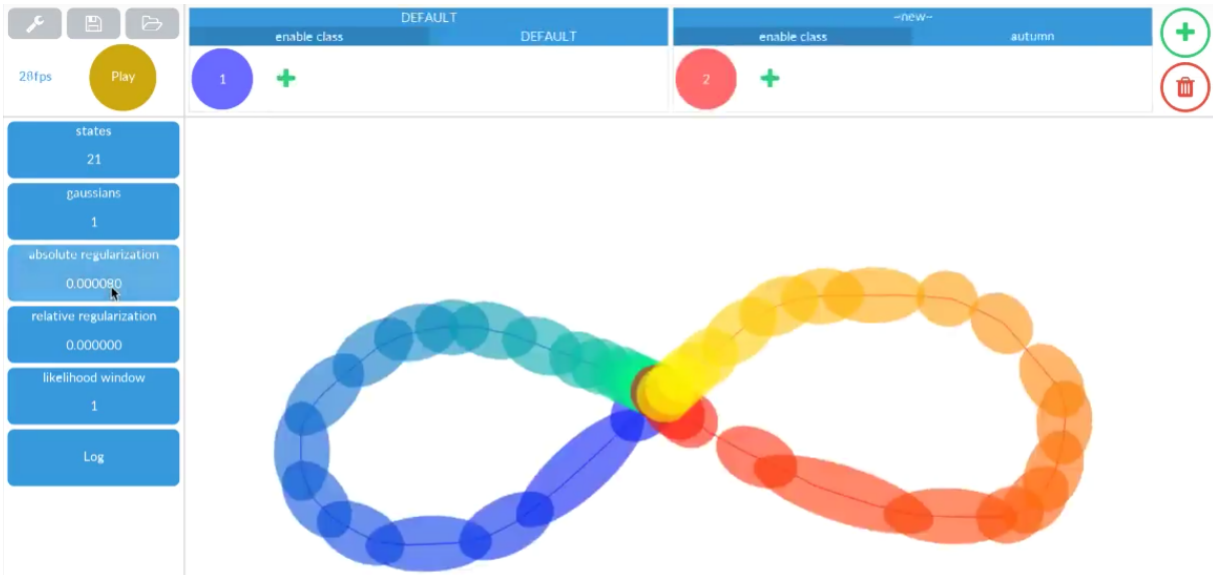


Figure 19: Structural breakdown of a generic IML system [63]

right decision boundaries of a classifier. It is presumably the most relevant aspect in the IML workflow since it is where the most effort might be spent [63]. During the model steering, there will be an exchange of information between the model and the user that can manifest in different ways, for example as a potential concept drift [41, 150] or as situations where the users cannot correctly express their intent [226, 163]. [226] describe this exchange of information as the *training dialog* and the specific features selected as the *training vocabulary*.

Still, all the tasks impact model performance and user engagement with the IML system in a variety of ways and deserve focus. Explicit feature selection can be beneficial for generalised interactions as it can be translated into efficiency [230] and quality gains [34]. When selecting the model, allowing the user to accurately specify the model to use or make a comparative analysis and adjust the model parameters can be favourable for the users to deliver their intent [63]. For that reason, Francoise and colleagues (2016) [80] developed GaussBox, an IML tool for inspecting hidden markov models trained to recognise mouse gestures. The idea behind GaussBox (Fig. 20) is to improve users understanding of machine learning's elemental mechanisms by increasing the transparency of the interactive representation of gestures [80]. In Fig. 20 it is possible to see the representation of the likelihood of the markov model as the user performs the gesture with the mouse, aiming to help designers create efficient gesture sets [80].

In the field of gestural based computer interaction, Fiebrink and colleagues (2011) [73] explains that the shape and smoothness of the decision boundaries of a model can be of higher relevance to the user than the where those boundaries lay in the feature space. To reach that conclusion, Fiebrink and colleagues (2011) developed a tool to allow training supervised learning model for gestural interaction in music, called the Wekinator [73, 74]. In their study, the authors found that as users

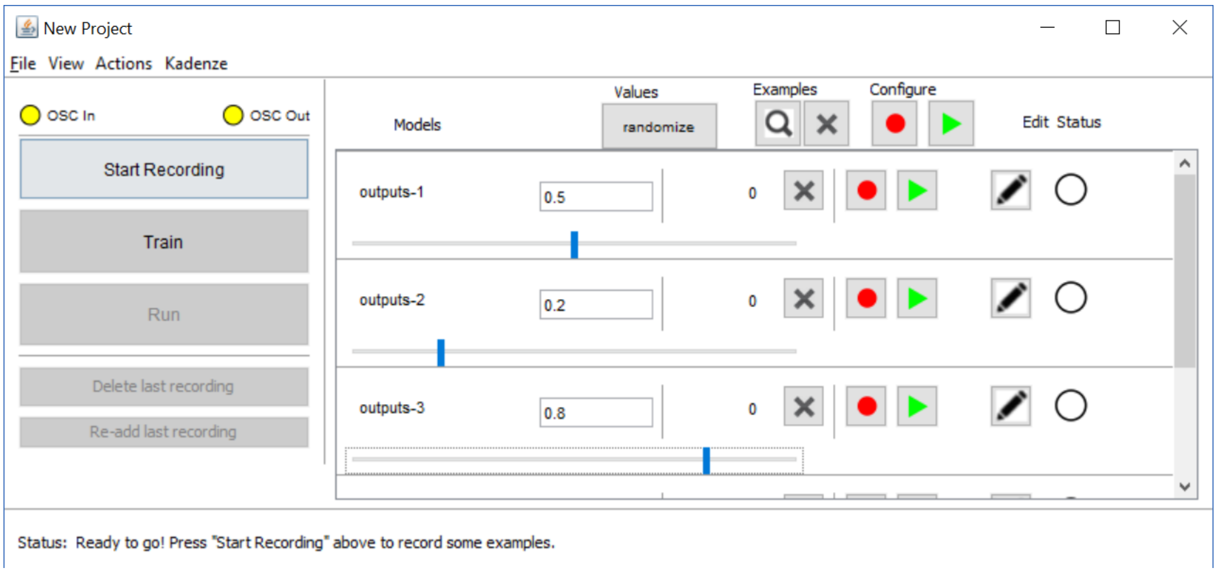


states 21
gaussians 1
absolute regularization 0.000080
relative regularization 0.000000
likelihood window 1

Figure 20: GaussBox Interface [80]

learned how to steer the model in the desired direction, the model taught them back how to recognise noise in the training dataset and how to adjust goals to match observed capabilities of the learning system. Interestingly enough, even a professional cellist identified flaws in her technique by using Wekinator to train a motion gesture recogniser [73]. The Wekinator follows an IML workflow by letting users select which type of model they want to use and sample paired gestures with sounds to perform regression or classification [74]. The Wekinator is a highly configurable tool where users can sample data from different sources, by sending OSC messages to the Wekinator through the computer's virtual ports system, and then experimenting with which model users feel they are expressing their intent better [74]. The Wekinator interface can be seen in Fig. 21.

As the literature points out, sometimes end-users might not have a clear idea of how exactly they want their interaction to be [226], and this is no exception when creating videogame interactions. It might happen that players might expect to have a full-fledged sword fighting combat system, but when allowed to freely encounter an enemy they might not know how to physically react. Or it might be that a game designer wants to create such sword-fighting system, but not have a clear idea of how it will manifest in its final form. Lü and colleagues (2014) [163] therefore tackled this intent problem during the training dialog by creating Gesture Script, an application where the users can describe the fundamental structure of a 2D gesture by providing new samples to improve gesture classification. In the Gesture Script example [163], the model steering activity is comprised of several sub-tasks in which users engage in the training dialog by either generating new samples for a gesture, selecting generated ones from the interface, or defining the underlying structure of a gesture by coding simple steps. The approach of Lü and colleagues (2014) creates an interesting solution to tackle the problem of user intent by creating a highly configurable system that doesn't penalize users when their intent or direction is not clear. Tsandilas and colleagues (2009) [296] studied as well gestural



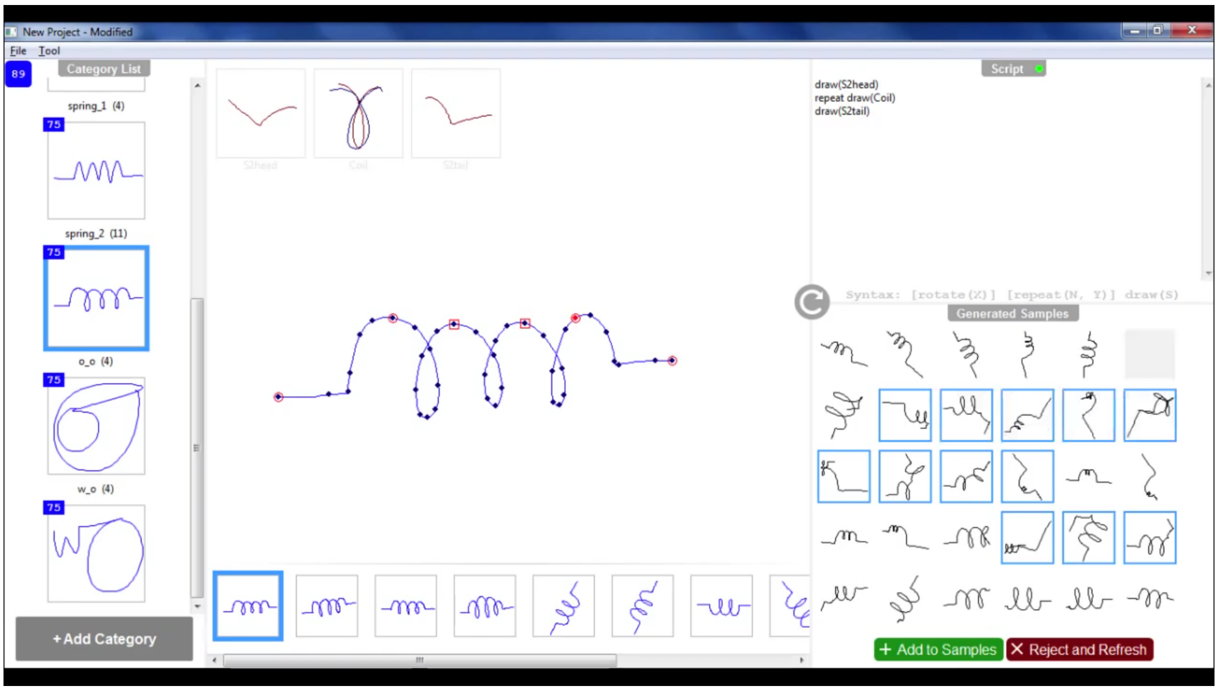
Status: Ready to go! Press "Start Recording" above to record some examples.

Figure 21: The Wekinator interface [74, 73]

interaction, although in music content research, and approached users intent drift by allowing users to sample gestures without explicit meaning and giving users the option to give meaning to pre-sampled gestures when the concept has solidified in their mind. The authors called their strategy "semi-structured delayed interpretation of gestures" [296].

Even when the user believes that they have a clear direction, there is always a potential risk of intent drift without users noticing as [63] extracted from the literature. [150] tackled the intent drift problem by developing a more structured sampling method during model steering. [41] answered the same issue by designing a series of heuristics and text guidance for the user to follow while the user is engaged in the training dialog. The users would encounter explicit text instructions while sampling to avoid drifting their intent, for example "When you show examples of an Angry face vary them as much as possible" [41].

These solutions for the interaction challenges that the training dialog offers can be very useful in the context that this thesis presents. As users interact with the system, having a very structured sampling workflow and explicit guidance can manifest as potential benefits on intent drift and delivering the intended player experience. Nevertheless, as Kleinsmith and Gillies (2013) [147] discovered, sometimes users will not follow an iterative strategy during model steering. Kleinsmith and Gillies (2013) study is of special relevance since it is following a very similar approach to the proposed in this thesis, by using an IML system to let users customize the behaviour of in-game avatars in a Kinect motion-controlled game (Fig. 23). Therefore, the customisation game interface should encourage an iterative strategy through the interaction and visual feedback presented [147], very much in line with the non-game related studies from Cakmak and Thomaz (2014) [41] and Kulesza and colleagues (2014) [150].Gillies (2016) [92] show as well the potential of involving an user in the IML process and the opportunities to create embodied



```
draw(S2head)
repeat draw(Coil)
draw(S2tail)
```

```
Syntax: [rotate(X)] [repeat(N, Y)] draw(S)
```

Figure 22: Gesture Script Interface [163]

virtual interactions aiming to immerse the user in a variety of scenarios [91, 92, 4].

Another of the characteristics of the IML workflow is that it can potentially relieve prohibitive challenges by making more accessible motion-controlled interactions. Katan and colleagues (2015) [141] studied how the IML workflow could be used to customise gestured-based instruments with people with mental disabilities and found that disabled and expert users shared the same goals and practices. Other studies have looked at increasing the accessibility of motion-based videogames for players with physical disabilities [29, 88], but have relied on pre-existing frameworks that required the development time from experts. Therefore, the same way customisable game controllers can improve the accessibility for regular game-pad interactions [122], even leading to the later release of mass-marketed products [85], IML systems offers the possibility to create adaptable motion-based interactions for video-games.

2.6.3 Summary

In this section of the background we introduced what interactive machine learning (IML) is and reviewed relevant IML work. We saw that the IML process comprises of fast and focused iterations of feature, data or model changes in the model steering loop. To perform such loop, users interact with GUIs displaying relevant information about the ML task at hand where they can perform model steering tasks. We then reviewed relevant IML systems from the literature that work with movement data, and showed that, while each IML system provided a well-performing solu-

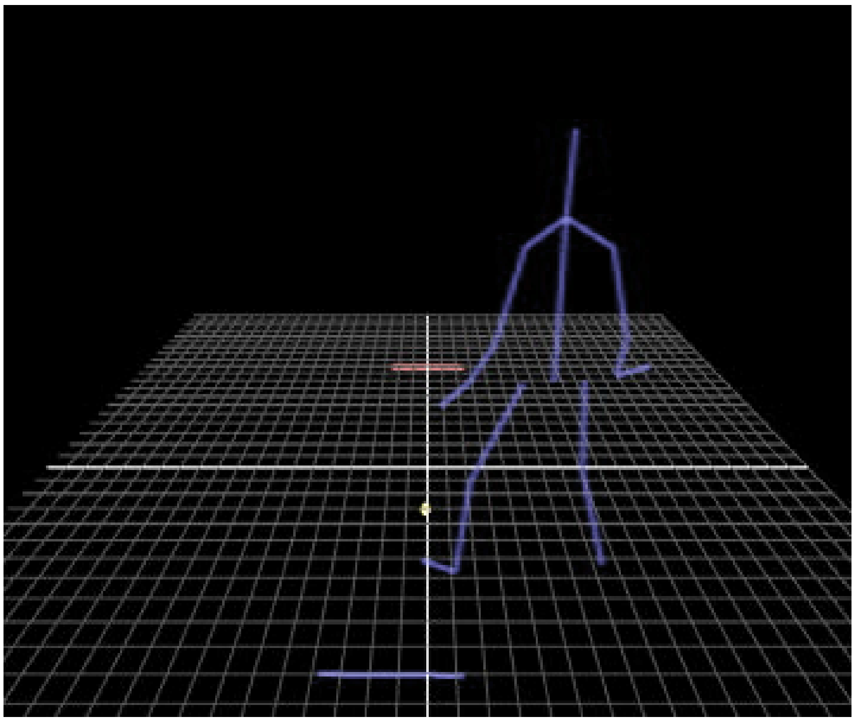


Figure 23: Player avatar in motion-controlled game [147]

tion to a particular movement-based problem, none of the current systems respond to the specific requirements of game creation practice and game motion control design and implementation.

2.7 Summary and Conclusions

In this chapter we have reviewed the literature about game motion controls in modern VR systems, and how IML can be a solution to support the embodied design and implementation of more positively perceived motion controls. There isn't a standard methodology for games motion control design [202, 263, 46, 98], and current game motion controls implementation solutions are either too focused on object interactions [301, 83], or too inflexible in their support for movement [306, 54]. Additionally, motion controls can elicit a rich and diverse set of experiential qualities grounded in movement [288, 40, 154, 125, 257] and the body [267, 71], thus any solution should accommodate the expression of such rich intent. Embodied interaction design methodologies ground their process in the body [121, 281], and propose methodologies of creative embodied design and reflection [174, 117] that suit our problem. IML is then suggested as a movement-focused implementation solution for the embodied interaction design creative reflection process [82, 93].

Nevertheless, and due to the particularities of the IML work pipeline, where the user is constantly iterating and steering the model [6, 63], there can be unexpected behaviours in motion-controlled games [147]. Furthermore, current IML solutions

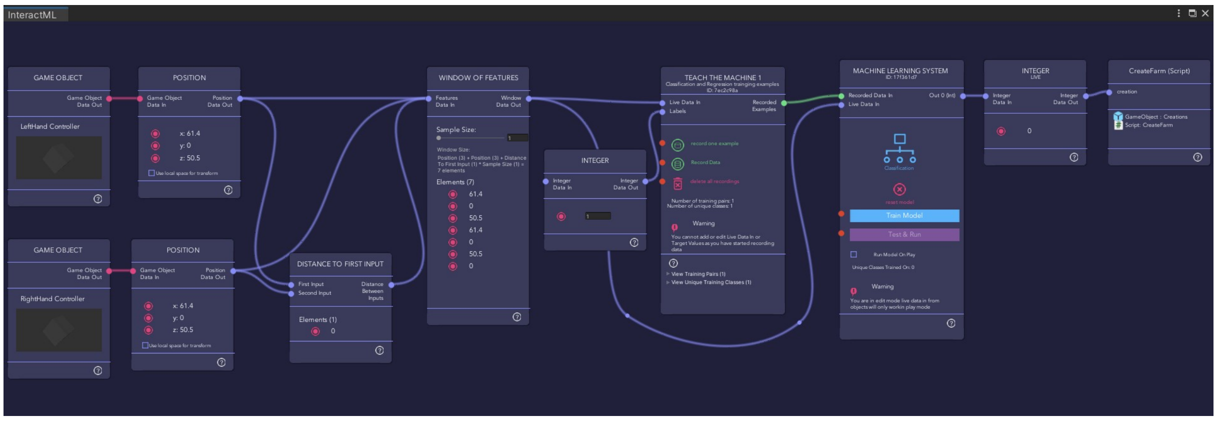


Figure 24: Screenshot of InteractML, our custom developed IML system for Unity 3D.

don't accommodate game creators needs of systems working in-engine [217]. There is therefore a research need for a new IML system built to support game creators in their embodied design and implementation practice.

The IML system to use should follow a series of requisites from the literature. The proposed IML system in this thesis should provide a way to configure input, output and model to use for creators [6, 63]. The system should allow for an easy way to visualise and de-bug the sampled-data by performing modifications on the dataset, as well as guiding the training dialog to avoid potential intent drift [226, 163, 150, 41]. Finally, the proposed model should fully account for an IML workflow allowing rapid iterations by being as easy to use by end-users as possible [6, 63].

Is for the above-mentioned reasons that we decided to develop a custom IML solution in the form of a visually-scripted game engine plugin, called InteractML (Fig. 24). The proposed plugin wraps most of the functionality of the C++ RAPID-MIX API [50], which contains the requirements needed to allow the IML workflow in terms of model selection and training. The plugin works as an integrated component of Unity 3D [300], offering a visual scripting interface displayed as an additional window in the game engine (Fig. 24). Game creators can configure inputs, outputs, model selected and providing the tools to debug the feature space and sampled data, both in desktop and VR mode.

In chapter 3 the system will be introduced, and its functionalities, features and machine learning algorithms described in detail.

3 InteractML: Our IML Framework

3.1 Introduction

Videogames are increasingly incorporating a diverse variety of sensors. Examples include DIY hardware games, VR sensors, smartphones, AR systems, smart watches and more. Despite decades of research on using sensors as game controllers, there are still no standard practices for how to design sensor-based interactions. It can be difficult for developers to implement accurate and robust movement analysis when sensors are noisy or high-dimensional, or when the goal is to sense complex motions or actions in games. Furthermore, the motion-controlled game industry transitioned from low-level IMU or camera systems with 3 degrees of freedom (DoF) [201, 272, 187] to current 'room-scale' virtual reality (VR) systems with 6 DoF [119, 211]. Unfortunately, there isn't an agreed set of motion control design methodologies for VR games [205, 89, 263, 46], and current in-engine VR solutions for motion control design are overwhelmingly focused on object-interactions [188, 301, 83] while leaving unattended the movement-focused richness arising from plenty of unsupported interactions [93].

Moreover, previous generations of motion-controlled systems were constructed on the same visualisation principle as traditional computer work: users are meant to move always facing the screen to receive visual feedback. However, VR's 6 DoF displays break that principle by displaying information anywhere in the virtual scene, since the interaction and spatial perception of the stereoscopic visual illusion strongly resembles that of real-life, where users move and interact with their whole bodies. Hence, game creators 'hide' the main display of the computer when they place the VR headset on during VR development. Since their view of the keyboard and mouse is also overriden by the stereoscopic display, current VR systems show a virtual representation of the two controllers (in the case of controller-tracking) or the two hands (in the case of hand-tracking). This creates the need for IML tools for games to also have a 'presence' in the VR display, both for information display and system control.

Previous literature on end-user motion authoring explored interactive machine learning (IML) workflows on creative domains [74, 6]. In an IML workflow, end-users follow iterative loops of human teaching and machine learning of model steering iterations [63]. Via iterative model steering, end-users author machine learning models that are able to process user input and transform it into meaningful outputs. For instance, the creative domains that showed promising results include music motion research [74, 73] or dance [81], where body movement of musicians and dancers was processed to create music, visuals or a combination of both. Nonetheless, none of the previous domains included motion-controlled videogames or IML toolkits specifically built to support game creation processes.

In this chapter, we describe InteractML, which is an IML system designed to facilitate the design and prototyping of motion-controlled games to capture rich information about player movements and actions. This work is informed by the success of IML in facilitating the design of new gestural musical interfaces for professional musicians [76, 75] as well as children and adults with disabilities [213, 141]. InteractML has been created in the form of a no-frills plugin for the widely-used Unity3D game engine. The design principles behind InteractML were (a) to maintain game editor interaction metaphors game creators are already familiar with (i.e. drag&drops, gameobjects, scripts) and (b) to facilitate non-expert work via a visual-scripting interface for ML tasks [173, 63].

Additionally, InteractML supports in-VR development by displaying IML system state on a movable virtual panel. The IML information displayed includes input features values, model steering stage (i.e. recording data, training, running) and live ML model output. There are also VR specific nodes in the game editor node-graph interface to control any other node behaviour with VR controllers, thus allowing to change labels, record data, train or run ML models.

The chapter begins in section 3.2 by introducing an overview of what is interactive supervised learning and what are the stages from a user-centred process, followed by our design rationale and process to implement such stages in section 3.3. Then, section 3.4 details the core contribution that this chapter presents: InteractML's user-centred in-engine visual and embodied model steering. Afterwards, the system architecture is explained in section 3.5, and a detailed explanation of each node is done in section 3.6. Finally, the chapter discusses how our visual and embodied model steering process in the game engine editor compares to prior IML literature.

3.2 Interactive Supervised Learning

Supervised learning algorithms are capable of building new recognisers or control systems from examples, rather than requiring a developer to write code analysing sensor data and specifying how an application (e.g., a game) should respond. Specifically, an algorithm learns from examples of human gestures or actions, each paired with the desired response. For instance, a developer (or player) could provide a few examples of an accelerometer being tilted right and left, along with information about how the colour of an on-screen game object should change in response to each tilt. A supervised learning algorithm can then build a mathematical model capable of choosing a new colour (or any other property) change in response to each new tilt observed during gameplay.

In interactive machine learning, a user (e.g., a developer or player) iteratively builds and refines the model through "cycles of input and review" [63]. IML systems for building new gestural controllers for music typically allow users to create new gesture examples on the fly, and to evaluate models by experimenting with the new controllers in realtime [76, 90]. Users can iteratively modify the training examples, sensor, and learning algorithm to improve performance. A similar approach has been used to customise virtual game characters' behaviour via physical player

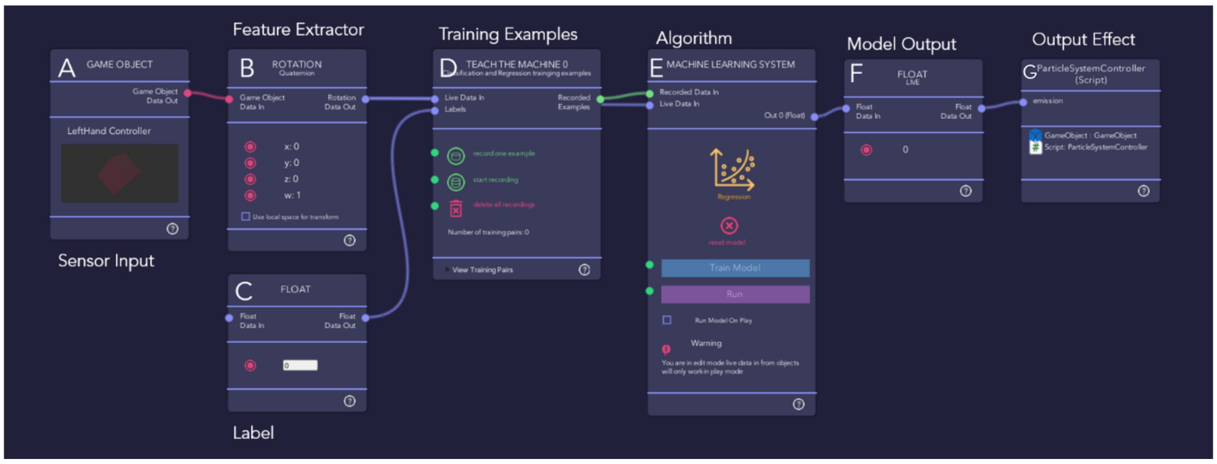


Figure 25: InteractML, the proposed interactive machine learning solution. On the left of the image are the nodes extracting features from gameobjects in the scene. The node “Rotation” is a “Feature Extractor” that sends data from a gameobject forward in the graph. In the center of the image is the “Training Examples” node where users can iteratively collect pairs of features extracted (positions, rotations, velocities, etc) and game outputs (sounds, colours, particles, etc). To the right is the “IML Configuration” node, which holds the properties to build the ML model, specifying the type of supervised learning algorithm (classification, regression or time series analysis), the training dataset and which gameobject will feed features during gameplay to perform the real-time ML analysis. Game creators can iteratively customise the training dataset, the features extracted or the properties affected by the ML model outputs until they reach the ML configuration that best expresses their intent (wave a hand, fly a dragon, play an instrument, etc).

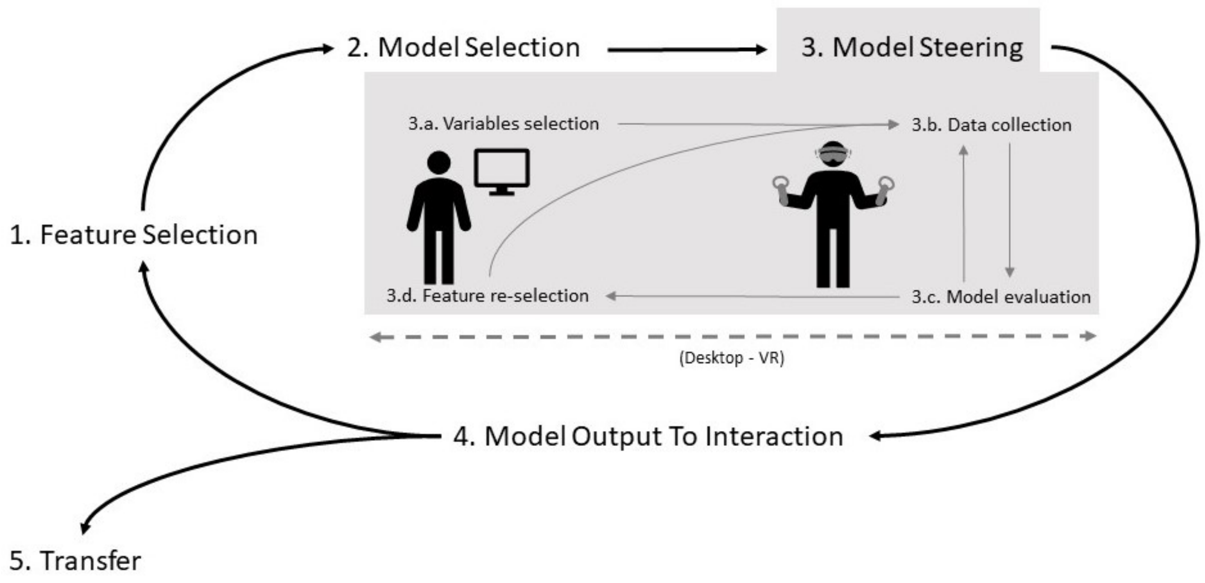


Figure 26: IML stages as present in InteractML

movements [147].

Several standalone software tools have been created to support IML creation of gestural control interfaces, such as Wekinator [76], GRT [90], and GVF [45]. Occasionally these have been used in games. For example, Schedel and Perry [244] used Wekinator to create a musical game in Unity3D controlled using a Cello K-bow and a Microsoft Kinect [244]. However, the implementation of this system was difficult and needed the writing of substantial new code to enable communication between Wekinator, Unity and the audio engine. Further, the final game could not be released as a standalone application, as it required multiple software programs (Unity3D, Wekinator, and a tailored application to extract data from sensors) running in parallel. The complexity of this toolchain limits the utility of such an IML solution for game developers that are used to different subsystems in the game engine editor window with seamless integration between them (i.e. the scene viewer with the light system, the shader viewer with the behaviour tree system, etc.).

3.2.1 Overview of Interactive Supervised Learning Stages

Interactive supervised learning is the IML paradigm that InteractML follows. It relies on iterative human-in-the-loop cycles where labelled training data is recorded, an ML model is trained and evaluated. Dudley and colleagues [63] generalised an IML workflow divided into four main stages: (1) feature selection, where the user chooses meaningful symbolic representations of input data to generate training data; (2) model selection, where the user chooses a suitable algorithm to tackle the machine learning problem; (3) model steering, where the user engages in iterative loops of data collection, model evaluation and/or feature re-selection; and (4) transfer, where the user packages and deploys the trained model onto a target application. InteractML follows the four main stages described in Dudley and colleagues [63], but additionally requires users to think about (a) how the model output is mapped onto a game interaction, and (b) which variables and data types should be chosen to represent the mapping. Hence, a summary of InteractML IML stages would be:

1. Feature selection: the user chooses meaningful symbolic representations of the VR input controllers to record training data. These representations are known as features, and the user selects them by connecting the appropriate feature node into the VR controller node (Fig. 25.B, Fig. 26.1). The features we implemented for InteractML are position, rotation, velocity, distance and window of features.

2. Model selection: the user chooses one of the algorithms offered to do classification or regression via creating an 'ML System' node of the appropriate type (Fig. 25.E, Fig. 26.2). The algorithms offered for classification are k-nearest neighbour (KNN) and dynamic time warping (DTW), and for regression a multilayer perceptron (MLP).

3. Model steering: the user follows iterative loops of (a) variables and data types selection, (b) data collection, (c) model evaluation and, depending on assessment, (d) feature re-selection and/or (e) further data collection (Fig. 26.3). A summary of each substage would be:

   (a) Variables and data types selection: the user select variable nodes in the IML visual scripting graph to define and connect different types of data to pins for labelling training data or visualising model outputs (Fig. 25.C, Fig. 26.3.a).

   (b) Data collection: the user collects training data via creating a node called 'Teach the Machine Node' and recording their own movement data as they exemplify movements. Each data recordings is paired with a label to produce a labelled dataset that then can be connected to a 'ML System Node' (Fig. 25.D, Fig. 26.3.b).

   (c) Model evaluation: the user performs a qualitative direct evaluation over the model [73], where they bodily *feel* how the model recognises their movements. Models output their inferred result onto a variable node (Fig. 25.F, Fig. 26.3.c).

   (d) Feature re-selection: the user might decide to change the features selected after an unsatisfactory evaluation. This step will require the user to collect further data as the previous dataset didn't include the correct features (Fig. 25.B, Fig. 26.3.d).

   (e) Further data collection: the user might decide that there is a need to record more training data after an unsatisfactory model evaluation. More data can be recorded following sub-step (b) data collection (Fig. 25.D, Fig. 26.3.b).

4. Model output to interaction: the user will map the model output – which is usually an integer – into a game interaction. To achieve this, the user drags and drop a game script from the main game engine editor view onto the IML visual scripting graph, thus creating a 'Script Node'. These 'Script Nodes' respond to the logic written in them, and show as many input or output pins as specified in the variable declaration (Fig. 25.G, Fig. 26.4).

5. Transfer: the user can export the trained models by packaging and loading training data and models as JSON files (Fig. 26.5). This process is automated following Unity3D packaging process.

Below we will describe our design rationale and design process to create an InteractML.

3.3 Design Rationale of InteractML

The design goals of InteractML were grounded by the limitations posed by prior IML tools [76, 45, 90] in respect to modern game motion controls design practice [217, 244, 83]. We aimed at designing a system that (1) is fully integrated in the game engine as an additional editor subsystem, (2) reduces the amount of written programming code required to use, and (3) follows a visually scripted approach to configure the IML system. To tackle the no-frills integration into the game engine, since its very inception we designed InteractML as a native game engine plugin, where users download and install and additional package into Unity 3D, which displays an additional editor window with the system reusing engine interaction metaphors via supporting drag and drops of scene objects. We wanted to design a system to reduce written code with a visually scripted interface. Therefore, we decided to use a node-based visually scripted interface for users to define feature extraction from dropped game objects onto the node graph window, training data collection, model training and inference via dedicated nodes.

The decision to employ a node-based system for InteractML was influenced by the popularity of such solutions in game development, namely Unreal Engine's Blueprints [66, 313] and Unity's ShaderGraph [299]. Additionnally, node-based systems for visual programming has been shown to simplify adoption and programming outside of games development [173]. Hence, by following a similar node-based approach, InteractML aligns with familiar visual scripting paradigms in game development, potentially reducing the learning curve for creators and promoting adoption of IML tools.

Additionally, InteractML's data workflow is strongly inspired by previous IML systems, especially the workflow present in the Wekinator [74]. Wekinator functions as a Java application based on the Weka machine learning framework, and offers a GUI relying on dropdowns, buttons, sliders and tables. Such a design requires game creators to (1) write code to extract features from objects in their game, (2) write additional Open Sound Control (OSC) network code to send such features onto Wekinator, (3) use Wekinator’s GUI to set up a Wekinator project, record data and configure an ML model, (4) write code to receive and process model outputs from Wekinator. Additionally, once deploying their game to players, game creators need to ensure that Wekinator is also installed and open on players machines, which complicates deployment. Wekinator additionally relies on ’projects’ as a way to define input and output parameters, requiring users to change projects if users want to modify the number of inputs or outputs in the machine learning model.

InteractML attempts to simplify this process by allowing users to visually map model inputs and outputs to game script variables directly within the game engine. Our design requires game creators to use InteractML’s node window to (1) visually select features from game objects in the game scene, (2) visually record data and configure one or several ML models, and (3) visually map model outputs to game script variables. The user will only need to write code to process the variables receiving the model outputs in a game script as they would do if the variables received the information from another game subsystem (i.e. no network code required). Additionally, InteractML’s also offers an in-VR console displaying the node-graph status, which creators can use to follow the IML process in their practice fully in-medium, which is an interface that creators would need to construct themselves if using Wekinator. Finally, models are automatically exported to players when game creators deploy their games in an ’invisible’ manner, similarly to how the rest of the game engine deploy systems to a player build (e.g. the player isn’t actively aware what makes the lighting or physics work). Therefore, our design facilitates a more accessible and user-friendly experience, particularly for non-technical creators, by enabling them to incorporate the IML workflow to their process with fewer steps.

3.3.1 Design Process

The development of InteractML followed an iterative process with expert feedback, where the design methodology was characterized by a series of progressive refinements without the direct involvement of end-user testing or evaluation. This approach leverages iterative cycles of planning, implementation, feature testing, discussion with specialists and subjective personal evaluation, where each phase builds upon the insights gained from the preceding one. Decisions regarding feature implementation or modification were informed from 2019 until 2022 by Prof. Rebecca Fiebrink and Phoenix Perry, that acted as experts, and allowed us to build on from their IML experience for creative practice [76, 73] and games [244].

The initial purpose of the project in 2019 was to expand on the shortcomings from Fiebrink’s and Perry’s prior work, resulting from decades of user feedback and their personal specialist introspection during their own IML research. From 2020 until 2022 we collaborated with Prof. Marco Gillies, Dr. Nicola Plant and Clarice Hilton as part of the 4i project, which investigated how InteractML was used by tech-

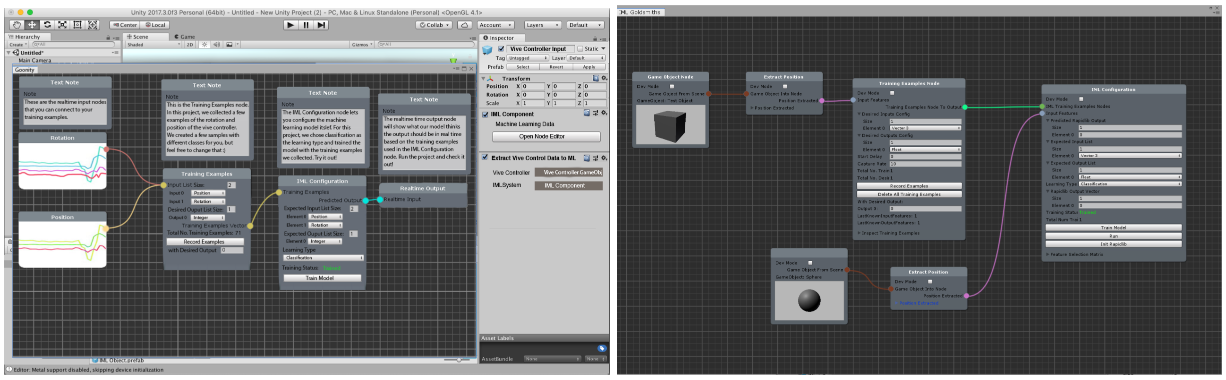


Figure 27: InteractML Design Iterations. (Left) The original mock-up of InteractML, showing the node-based system with data flow with feature extraction, data collection, and model configuration as dedicated nodes. (Right) The first implemented iteration of InteractML displaying the initial implementation of the mockup design to include objects from the scene and nodes for feature, training data and model configuration.

nology artists as part of an embodied design methodology using machine learning [219, 220, 221, 113]. During this period, and after each iteration, we gathered expert feedback and discussed our improvement proposals informally within the team of collaborators. These discussions went from tackling general IML principles (e.g. "How is model steering performed now?") to focused discussions tackling pilot studies performed as part of the 4i project (e.g. "How are users going to implement and evaluate movements with both their VR controllers in irregular circles?"). Hence, changes were decided in discussions between the main author of this thesis and his collaborators, taking into account the expert feedback from Prof. Rebecca Fiebrink and Phoenix Perry. Changes consisted of usability enhancements (e.g. changing the text or button layouts in nodes), functionality expansions (e.g. introducing new nodes), and performance optimizations (e.g. optimizing the update loop of the model node). Between iterations, we developed small prototypes aiming to assess how it is to place InteractML into controlled implementation scenarios. The original mock-up of the interface, together with its first iteration can be observed in figure 27, which is visibly different from the final form of the tool in figure 26. The last period of design work of InteractML, from 2022 until 2023, was conducted by the main author of this thesis. This period focused on designing how users provide explicit testing examples of their own movement, and followed a similar iterative process with the supervisors of the thesis providing feedback based on prior experience (see section 3.6 for more information).

The original mockup of the tool (Fig. 27 (Left)) laid the foundations of InteractML's data workflow, in which (1) features nodes are fed onto a 'Training Examples' node to build a dataset, (2) the training dataset from the 'Training Examples' node is fed onto a model in the 'IML Configuration' node, and (3) the model prediction is fed onto a 'Realtime Output' node. Such a design attempts to implement the basic steps in the IML process to perform feature and model selection, model steering and model output to interaction.

The first working implementation of InteractML (Fig. 27 (Right)) included game object nodes and feature extraction nodes, the ‘Training Examples’ node and the ‘IML Configuration’ node from the original mockup design. One of the changes from the mockup design to the first implementation was the separation of input data coming onto the ‘Training Examples’ node to the ‘IML Configuration’ node, aiming to decouple input for training datasets from the input for models to encourage flexibility. The configuration of the training dataset and the model was done via fields and dropdowns on both the ‘Training Examples’ and ‘IML Configuration’ node, following a similar set of options to that of Wekinator’s GUI. In this initial implementation, the model (or models) output was mapped onto game scripts interactions via referencing an entire graph and reading an array with all model outputs sequentially.

Nevertheless, after iterations of personal prototype implementation and system refinement, we identified several points of friction with the initial implementation. One of the first problems to arise was processing the array of model outputs from an InteractML node graph reference. We initially followed the same approach as the Wekinator, where an array of outputs is returned and the user needs write code to segment and parse the data onto variables that then can be used for game logic. We found that this ended in an unnecessary amount of code written that could be simplified directly in the node graph. We introduced ‘Script Nodes’ that allowed to visually map model outputs to specific game script variables on the node graph editor, reducing the amount of written code for game creators. The additional advantage of such design is that users could then train several models in an exploratory fashion, but not connect all of them onto the game script. The second point of friction we identified was the configuration of the ‘Training Examples’ and ‘IML Configuration’ via fields and dropdowns. Such interface process made exploration on feature selection and model steering time consuming because the process required two redundant steps: (1) create and connect feature nodes, (2) specify via dropdowns what features are expected to be connected. We decided to simplify the configuration by removing the fields and dropdowns, and instead let users select and connect features and variable nodes on the ‘Training Examples’ and ‘IML Configuration’ node. Furthermore, we decided to split the ‘IML Configuration’ node into specific model nodes to reduce the amount of clicks required for model selection and configuration. Finally, we decided to overhaul the look of nodes and colour palette to hide information that we considered redundant or confusing, and attempted to highlight when variables changed on nodes to visually highlight the data flow. The visual look overhaul also included changing the names of the ‘Training Examples’ node into ‘Teach the Machine’ node to signify that users should spend effort ‘teaching’ movements to this node, and that this node would be used to ‘teach’ a model once connected. The final look of InteractML is the one that we elaborate on during the rest of the chapter, and it was kept consistent for the VR interface as well (see core contribution in subsection 3.4).

Below we will describe more in detail how the core contribution of InteractML –an in-engine and in-medium visual and embodied user-centred process for interactive supervised learning.

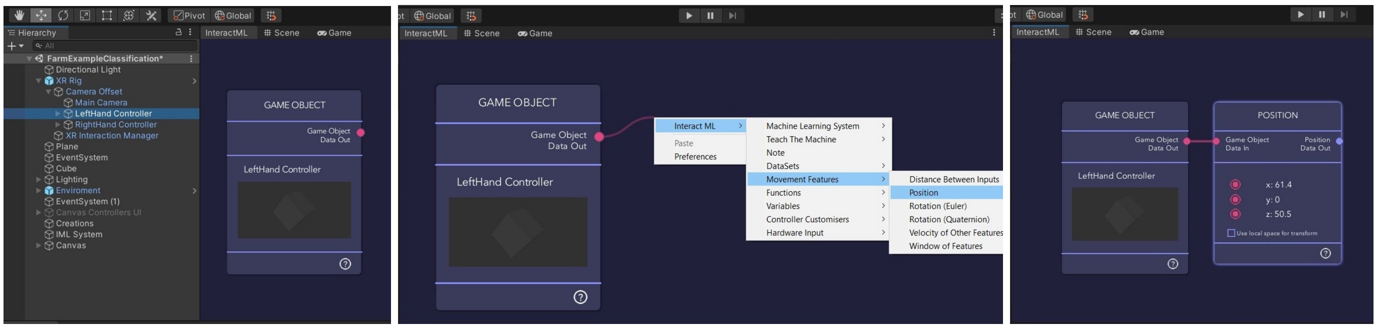


Figure 28: Position selection of the left VR controller. (Left) The user drags and drops the ‘LeftHand Controller’ over to the IML Graph to create a ‘Game Object Node’ with a reference to the controller. (Middle) The user clicks on the output pin and selects the ‘Position’ movement feature from the contextual menu (Right) The Position Node showing real-time position changes.

3.4 Core Contribution: Visual and Embodied Interactive Supervised Learning

3.4.1 Visual Feature Selection

One of the novel contributions to the field is the visual and user-centred design process of our tool. InteractML is a visually scripted tool directly in the game engine scene, and as such users will perform feature selection on the IML graph.

Firstly, users drag and drop any gameobjects that they want to extract movement features from. For instance, given a Unity scene with an XR Rig, which is a hierarchical representation of gameobjects of a physical VR system, the user can drag and drop the ‘LeftHand Controller’ onto the IML Graph to select the position feature (Fig. 28). The feature selection process in the example provided on figure 28 can be reused to import more gameobjects and select further features. For examples, in figure 45 both controllers have been imported and the user has followed the visual feature selection process to select the position of both VR controllers, then further select the distance between both controllers and select a window of features were all features are combined together in a window of size 1.

3.4.2 Visual Model Selection

In InteractML, the user can select a model via right-clicking anywhere on the IML Graph and selecting one of the available moodel nodes from the dropdown menu (Fig 29). The user can create as many model nodes as desired on the IML graph, and any of them can tick the ‘Run Model On Play’ checkbox (Fig. 47) that will allow a model to automatically run once the game starts.

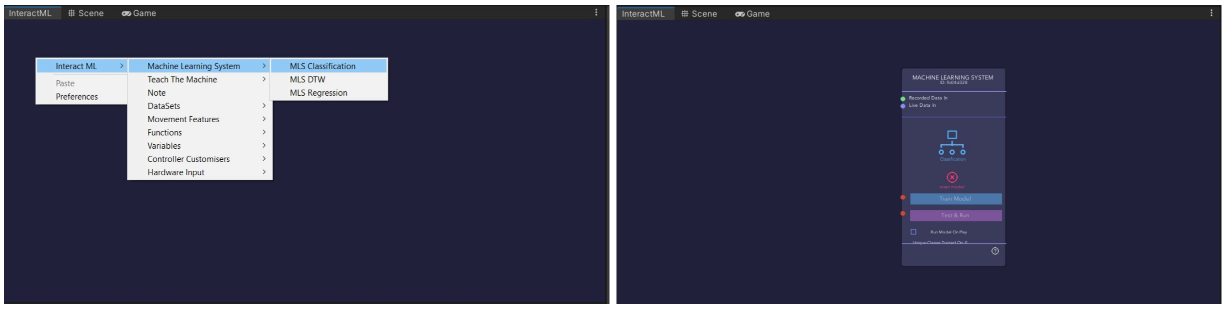


Figure 29: The user selects a classification model on the IML Graph. (Left) The user right clicks on the IML Graph to display the contextual menu. (Right) The user selected the 'Classification MLS' option to create the 'Classificataion Machine Learning System Node'.

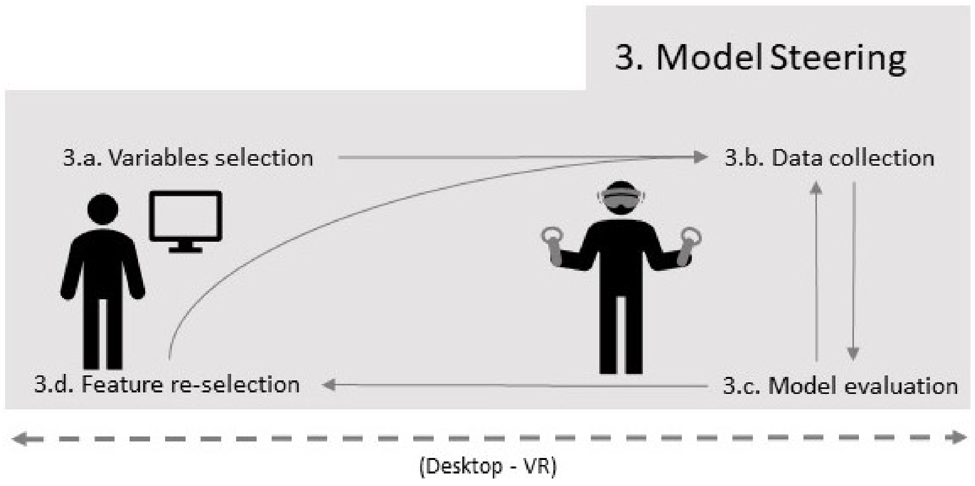


Figure 30: The model steering iterative task. The user can perform the task on desktop or VR depending on the sub-stage they are working on.

3.4.3 Visual and Embodied Model Steering

One of the main novel contributions of this chapter is the no-frills visual and embodied model steering capabilities of InteractML, both in desktop and VR, extending beyond the user-centred functionalities of prior IML systems [74, 90, 80].

In figure 30 we can see an overview of how the model steering task consists of four sub-tasks in InteractML. Before jumping to a detailed description of each higher-level IML task, we wanted to explain our core contribution in detail since it can be better understood explained as a whole process.

Visual variable selection and control: Users can create and connect variable nodes to specify labels or to visualise model outputs. However, in order to allow a better embodied variable control while the user is in VR, users need to be able to change the value of a variable with VR controller input. In order to do that, we implemented a node that serve as a simple mathematical function that adds or removes a value for any variable (Fig. 31).

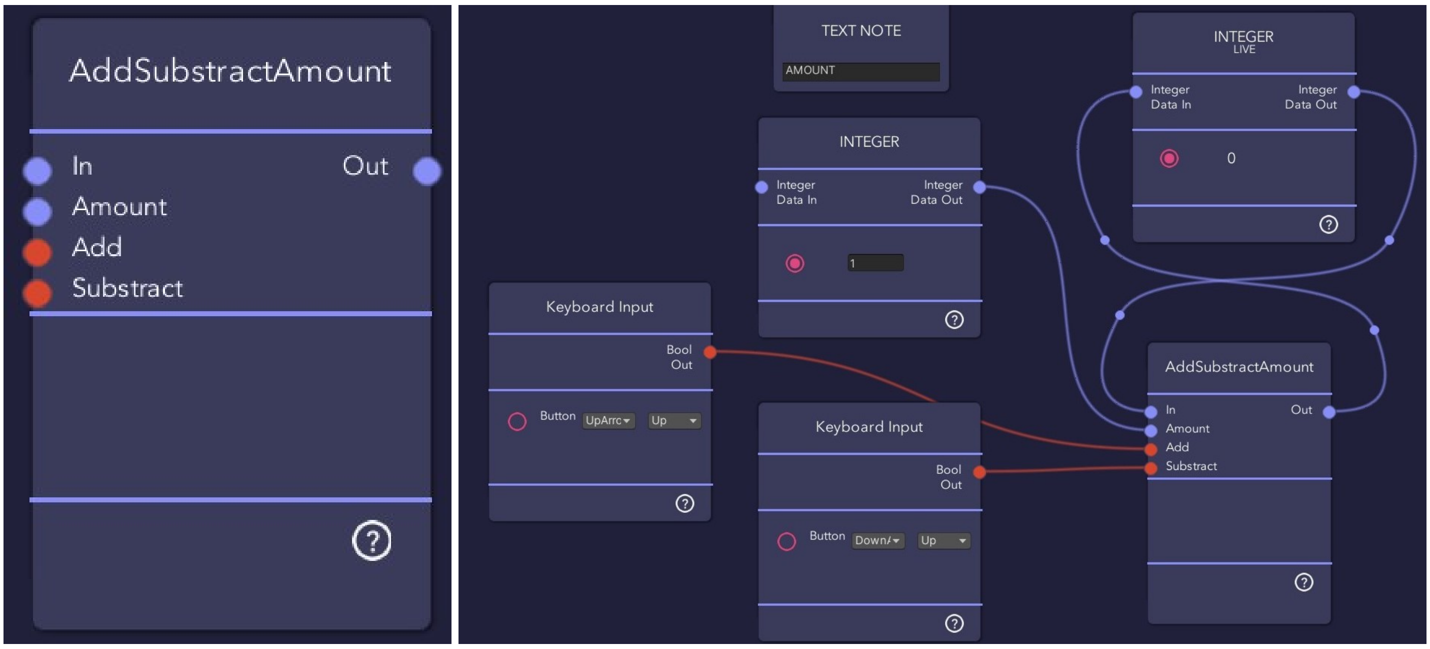


Figure 31: (Left) The node 'AddSubstractAmount'. (Right) The node visually programmed to add or remove an amount of 1 to a variable triggered via up or down keyboard arrows.

Embodied Data collection: The user can perform the entire data collection stage in VR by visually scripting the control of a 'Teach the Machine' node with VR controller buttons. In VR, the user visualises the current label and state of data recording on the VR console in 3D. The user will record pairs of performed human movements and labels while in VR.

Embodied Model evaluation: The user can bodily *feel* how the model recognises their movements, and observe the inferred output of the model on the VR console and, additionally, on the audiovisual in-game effect they aim to control.

Visual Feature re-selection: After an unsatisfactory evaluation, the user might decide to change the features selected. For this step, they will need to remove the VR headset and visually select features on the desktop IML Graph. This step will require the user to collect further data as the previous dataset didn't include the correct features, but it can be done in VR alltogether once features are selected.

Further embodied data collection: If there is a need to record more training data after an unsatisfactory model evaluation, the user can do so fully in VR, hence ensuring that, if there is no need to change features, the entire model steering process can be performed in-medium in VR. In this manner, the user will perform fully embodied in-medium loops of data collection and model evaluation, which are the most performed subtasks in model steering [63].

Therefore, by using InteractML, game creators can visually and bodily create and iteratively train interactive supervised learning models without needing expert ML knowledge. For example, a game creator that is used to create VR games for Ocu-

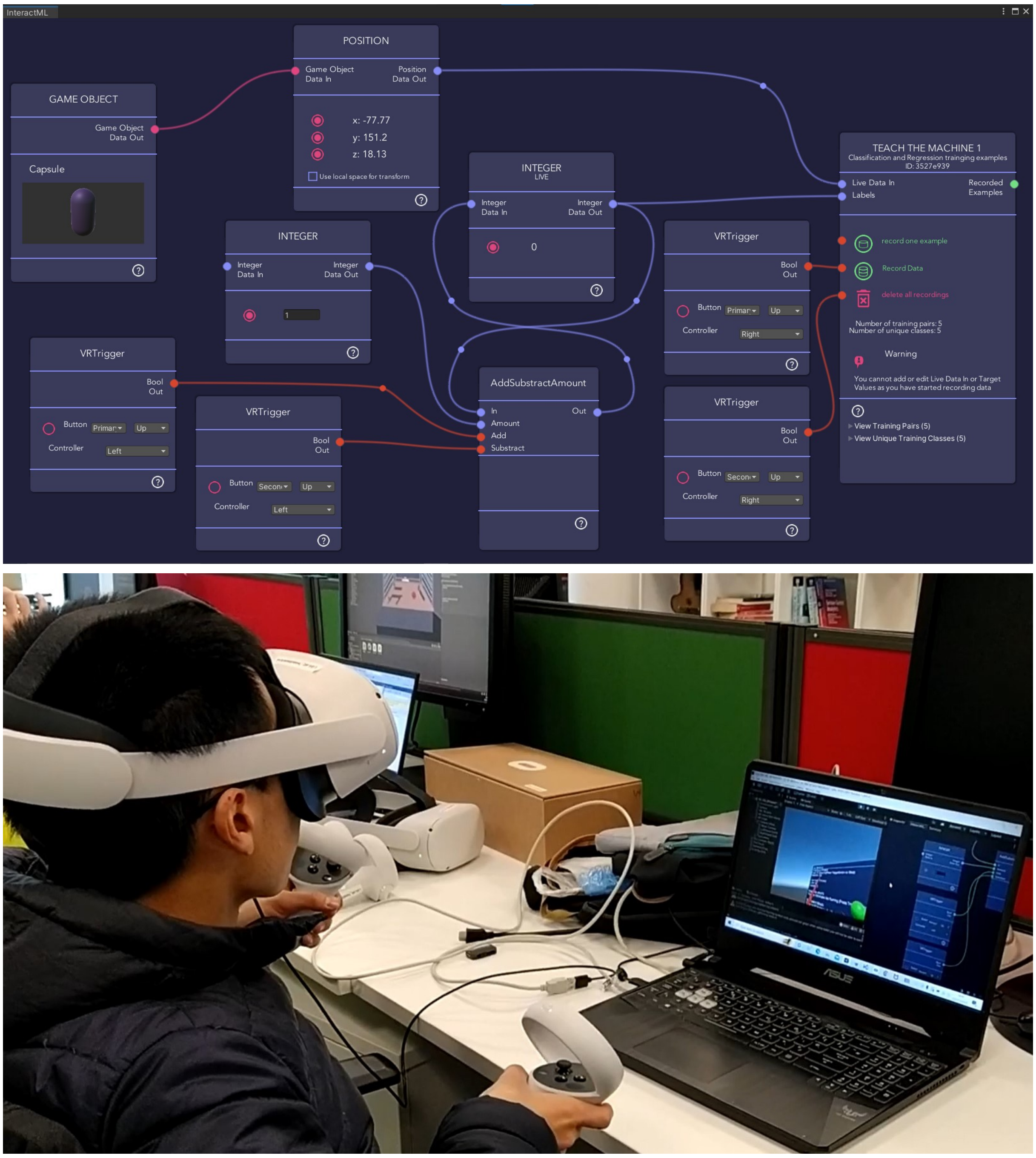


Figure 32: (Top) A 'Teach the Machine Node' programmed to perform embodied data recording via VR controller input. On the left side of the graph, two 'VRTrigger' nodes control that the dataset label changes with the left controller primary and secondary buttons. On the right side of the graph, the 'VRTrigger' nodes connected to the 'Teach the Machine' node control that the primary and secondary buttons of the right controller trigger data recording or delete the dataset in case of the user is unsatisfied. (Bottom) User performing embodied data collection. On his laptop screen it can be observed how he looks at the VR console ingame while controlling the change of label in his IML Graph with VR controllers.

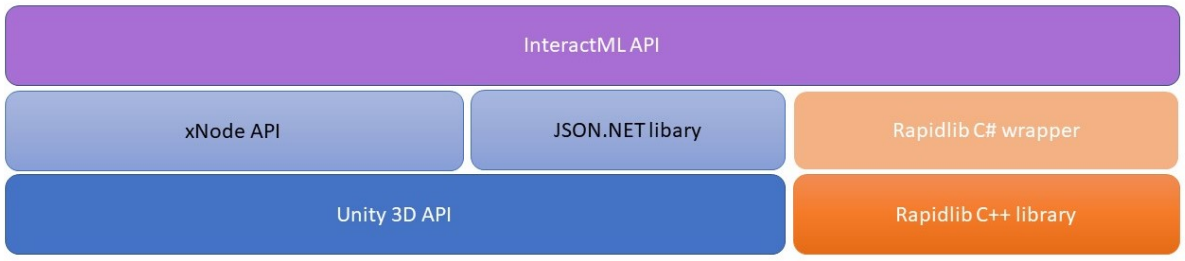


Figure 33: The dependency diagram of InteractML.

lus headsets with Unity 3D, wants to explore how to introduce dancing capabilities to their game. They can download and integrate InteractML from the github repository, and immediately start experimenting on a scene that they already have in place for their preivous game. They can drag and drop to the IML graph their XR rig objects and experiment with different features and movements all within the engine and the virtual scene they are familiar with. We believe these are much needed requirements to translate the knowledge generated in prior IML work onto such a practical and industrialised field as game development. Furthermore, we are advancing the state of the art in IML tooling by supporting a fully visually programmable IML loop, and a fully embodied model steering activity.

Still, there is value in understanding how InteractML supports our visual and embodied user-centred design process from a software standpoint. In the following section we will describe in detail the software architecture of InteractML, how data flows through the system, its most important classes and how InteractML integrates with the Unity 3D game engine.

3.5 System Architecture

In this subsection, we describe InteractML's underlying software architecture as a Unity3D game engine plugin. Firsly, we will describe the dependencies the system relies on to function. Secondly, we will introduce how data flows into InteractML. Thirdly, we will introduce an overview of all the relevant classes in detail. Lastly, we will describe in detail the most relevant classes in charge of data flow management, IML graph and node display, input handling and virtual reality support.

3.5.1 Dependencies

InteractML, as a game engine plugin, relies on a series of dependencies:

1. **Unity3D Game Engine API:** The first major dependency is the game engine itself. InteractML is developed as a Unity3D ([300]) and will not function outside of it. We don't consider this a limitation as this is an intended behaviour

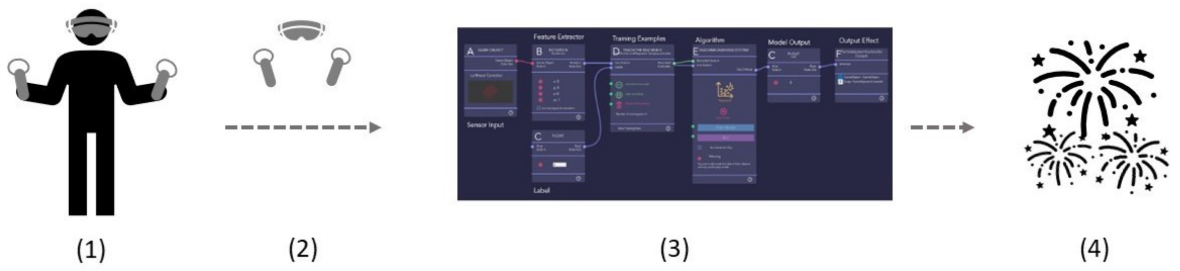


Figure 34: The movement data flow of InteractML. (1) User interacts with VR devices. (2) Movement data is piped onto IML Graph. (3) Movement data is (a) broken onto features, (b) processed by an IML model, and (c) piped onto a game script. (4) IML Graph output is mapped onto a custom game interaction in-game.

and one of the systems strengths. InteractML works on any Unity version from Unity 5 onwards.

2. **xNode framework:** InteractML's node graphical capabilities are built on top of an Unity open-source node framework called xNode [31]. xNode is a generic framework for node-graphs specifications and drawings. Graphs and nodes are scriptable objects (i.e. a Unity specific data structure) that can be accessed at run-time. Every InteractML node class and the IML graph class base implementation are provided via xNode. InteractML extends xNode to account for (1) custom data flow, (2) two update loops (in editor time and in 'play' time), (3) a full re-skin of the UI, and (4) custom library of IML nodes and functions.

3. **Rapidlib libary:** InteractML's uses Rapidlib as its machine learning backend [50, 320] via a C# to C++ interoperability layer. Rapidlib is a C++ All machine learning algorithms and training data structures extend from rapidlib's definitions.

4. **Newtonsoft JSON.NET library:** InteractML relies on Newtonsoft JSON.NET library to serialise the training datasets and system states into JSON files [200].

3.5.2 Data Flow

Because of the movement-focused nature of InteractML, data flows from the user onto a game action as depicted in figure 34.

3.5.3 Overview of Classes

`IMLComponent.cs`: InteractML has a main starting point to execute logic on the `IMLComponent.cs` class. This class has a reference to an IML graph and implements the main data flow update method, which (1) pulls information from every referenced gameobject

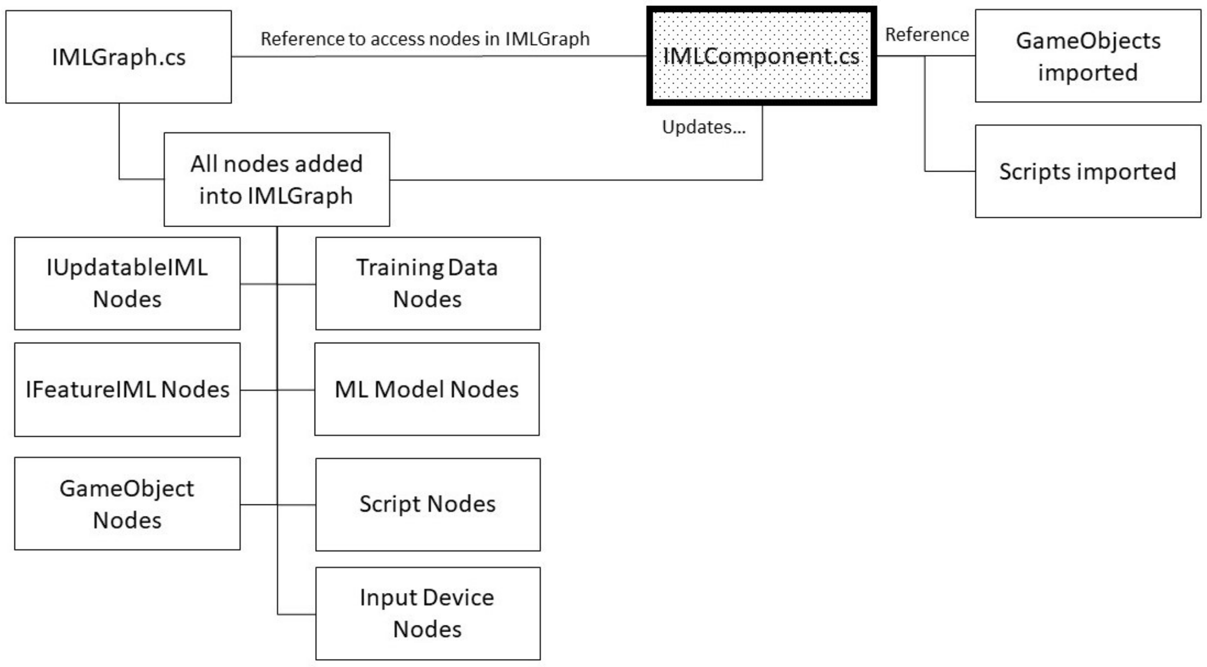


Figure 35: Relationship diagram of core classes of InteractML. The class `IMLComponent.cs` links gameobjects and scripts from the virtual scene with the nodes contained in an `IMLGraph.cs` instance.

into the graph and (2) updates the state of every node implementing the interface `IFeatureIML` or `IUpdatableIML`. The `IMLComponent` can be added as a component to any game object in the scene, but we provide an editor menu option to create an 'IML System' game object in the scene with an `IMLComponent` already added to it.

`IMLGraph.cs`: The IML Graph class contains a list of all nodes and specifies how the graph is drawn on the game engine editor.

`IUpdatableIML.cs`: An interface that marks any node class implementing it as updatable by `IMLComponent.cs`, and requires the implementation of the methods `Update()` and `LateUpdate()`. See code listing 1 for the core interface definition.

Listing 1: IUpdatableIML.cs interface

```
namespace InteractML
{
    /// Allows a node to implement the Update and LateUpdate methods
    public interface IUpdatableIML
    {
        ...
        /// Function to call to run code to update class
        void Update();
        /// Called after update has finished
        void LateUpdate();
    }
}
```

IFeatureIML.cs: An interface that marks any node class implementing it as a feature from a machine learning perspective, which allows the node to be considered for training data collection and machine learning inference input. This interface requires the implementation of the FeatureValues variable and the UpdateFeature() method. See code listing 2 for the core interface definition.

Listing 2: IFeatureIML.cs interface

```
namespace InteractML
{
    /// Implement this interface when a class will act as an input/output feature
        for the IML system
    public interface IFeatureIML
    {
        /// Values returned by this feauture
        IMLBaseDataType FeatureValues { get; }
        ...
        /// Function to call to run code to update feature values
        object UpdateFeature();
    }
}
```

IMLBaseDataType.cs: A generic class that identifies a variable as an acceptable input or output into a rapidlib model. Because rapidlib algorithms and training datasets expect all data to be parsed as double precision point numbers, we need to wrap any variable into an IMLBaseDataType class to support variables of diverse types on the IML Graph while complying with rapidlib's data requirements.

3.5.4 Game Engine Editor Time and 'Play' Time Loops

As shown in the class diagram in figure 35, InteractML updates every frame all nodes that require it via the IMLComponent class. To do so, a loop needs to be implemented and the main update method called in it. Because InteractML is a plugin of a game engine system, it needs to update both during editor time and 'play' time. Editor time is the period of work that the user spend on the game engine editor itself, without launching the game and compiling all scene scripts. In Unity, editor time stops once the 'Play' button on the top of the editor is clicked. Once that button is clicked, all game scripts are compiled and 'play' time begins, executing routines and code in the same order as during execution time on a built game.

Therefore, InteractML implements two different update loops –one for editor time and one for 'play time'. Each of them is handled by a different class:

IMLEditorManager.cs: Manages InteractML's update logic during editor time. The class offers public subscription events to subscribe methods from any class to run in editor time. On project start, all IMLComponent are found and subscribed. Additionally, when a new IMLComponent is instantiated its main update loop is also subscribed to the IMLEditorManager class.

`MonoBehaviour.cs`: The `MonoBehaviour` class is provided by the Unity API, and allows any class inheriting from it to automatically subscribe to the game engine update loops in ‘play’ time. Our `IMLComponent` class inherits from `MonoBehaviour` to call InteractML’s main update loop in ‘play’ time. Additionally, and thanks to the inheritance from the `MonoBehaviour` class, an `IMLComponent` instance can be created via the game engine editor interface when adding such class as a component to any game object in the scene.

3.5.5 IML Graph Display

In order to draw the IML Graph as a node-based window in the Unity3D engine, InteractML inherits from the generic node editor framework xNode [31]. The main class that draws the IML Graph editor in InteractML is `IMLGraph.cs`, which inherit from the class `NodeGraph.cs`. Our `IMLGraph` class implements the methods `AddNode()` and `RemoveNode()`, which allows create and destroy nodes in the graph (see code listing 3). The graph is structure as an Unity ‘Scriptable Object’, which facilitates many data persistence benefits that don’t need to be reimplemented. However, to support the persistence of scene game objects, scene scripts, training datasets and ML models, we needed to write custom serialisation methods exposed in the static class `IMLDataSerialization.cs`.

Listing 3: IMLGraph.cs class definition excerpt

```
using System;
using UnityEngine;
using XNode;

namespace InteractML
{
    /// Defines an example nodegraph IML Graph that can be created as an asset in
        the Project window.
    [CreateAssetMenu(fileName = "New IML Graph", menuName = "InteractML/IML Graph
        ")]
    public class IMLGraph : NodeGraph
    {
        ...
        /// Override addNode to account for custom adding logic
        public override Node AddNode(Type type) {...}
        /// Override removeNode to account for custom removal logic
        public override void RemoveNode(Node node) {...}
        ...
    }
}
```

3.5.6 IML Nodes

Node classes are defined by the `IMLNode.cs` class. Every node class in InteractML (e.g. models, training datasets, variables, etc.) inherits from it in order to correctly function as part of the `IMLGraph` class and be updated by the `IMLComponent` main update loop. The `IMLNode.cs` class offers methods to initialise nodes on instantiation and automatic assignment of unique IDs. Below is a list of the main node classes in InteractML which functionality is explained more in detailed in section 3.2.1:

1. `BaseMovementFeatureNode.cs`: the base class for all feature nodes and implements the `IFeatureIML` interface.
2. `TrainingExamplesNode.cs`: the base class for the 'Teach the Machine Node' where training data recording is performed. It implements the interface `IDataSetIML` to hold training data.
3. `MLSystem.cs`: the base class for all the different algorithms offered by InteractML to train and infer with.
4. `ScriptNode.cs`: the node that allows to drag and drop scripts onto the IML Graph. It shows information about a script and pipes data into our out of the script via input or output pins.

3.5.7 Input Handling

Input is handled in the main update loop of the `IMLComponent` class, which updates the information from any node implementing the `IInputType` interface. The `IInputType` interface includes the methods `OnTriggerChange()` and `OnButtonChange()` to detect when a button or trigger has been pressed on input devices.

The class `KeyboardInput` handles input coming from a keyboard device, and such input is exposed in the node class `KeyboardPress` which inherits from the `CustomController` parent node class. The user can select which keyboard key will be detected and can connect the output pin of the node into any clickable button on an IML Node (Fig. 36).

3.5.8 VR Support

Until now, we have described the 'InteractML Core' classes and functionalities. However, InteractML can be extended with additional modules and nodes. We created the VR module that adds additional functionality of top of InteractML to perform interactive machine learning loops in virtual reality. The module is divided in two main components: (1) VR display of relevant IML Graph information, and (2) VR control of IML Graph nodes.

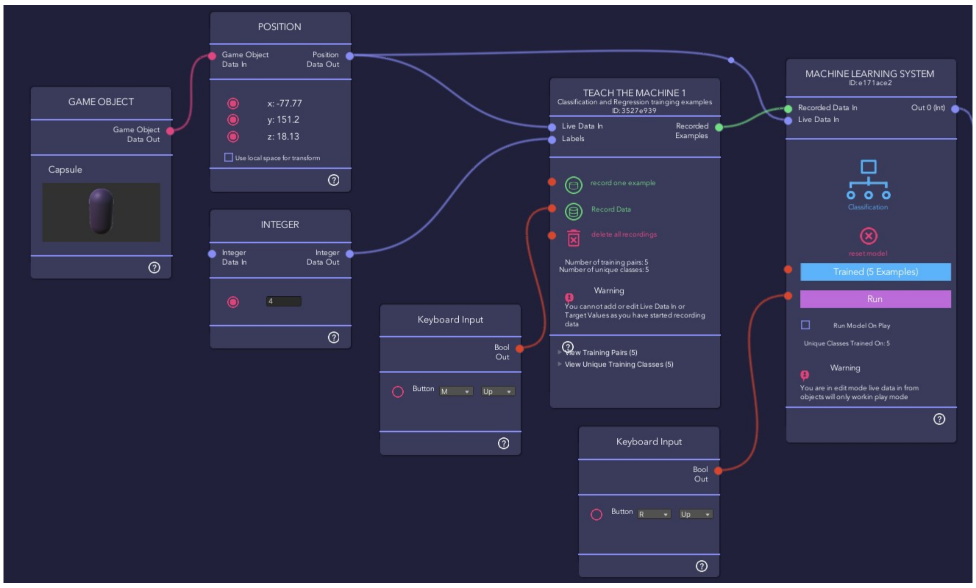


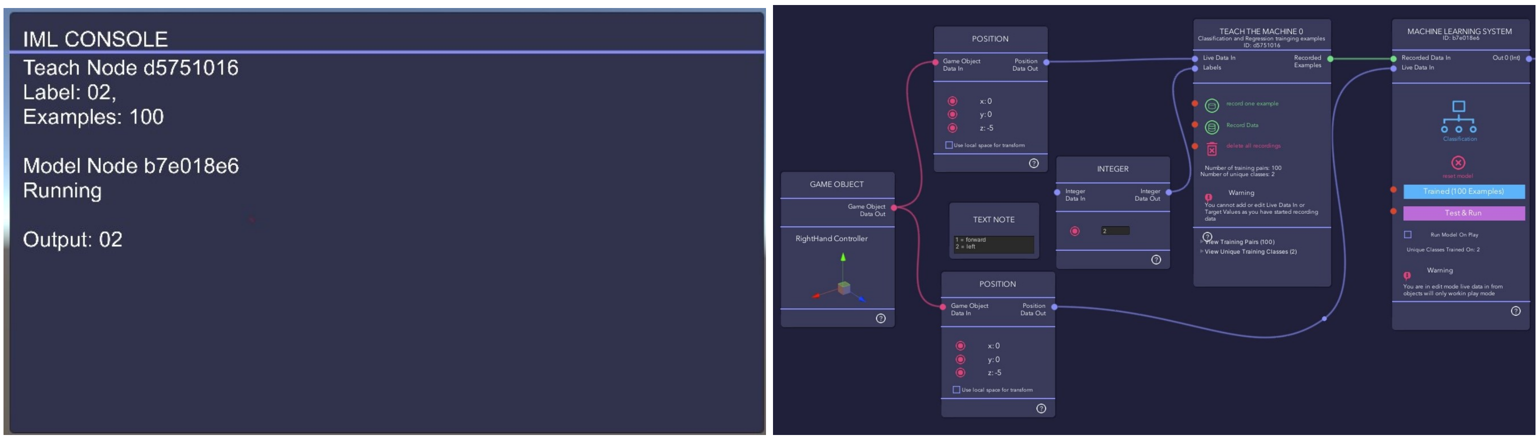
Figure 36: Two keyboard input nodes are connected to the 'Record Data' and 'Run' buttons, thus programming that the key press 'M' wil trigger data collection, and key press 'R' will trigger the model to run inference.

VR Display. Because InteractML is built around the Unity 3D game engine editor, which is created for desktop usage, we needed to create a solution to display information about the IML Graphs that users create in VR. To simplify the amount of learning that users would require, we decided to follow a simple console approach that displays information about an individual IML Graph on a virtual panel (Fig. 37). The console displays the ID of each 'Teach the Machine' node or 'ML System' node in the graph. From the 'Teach the Machine', the console displays the current label connected to the node and how many examples are stored in it, together with the state of the data collection (i.e. whether is currently recording data or not). From the 'ML System', the console displays the total number of training data trained on together with the state of the model (i.e. training, running) and the inferred output of the model. This way, the user can visualise core system information while in VR.

VR Input Handling. The VR module adds a new kind of input device node specific for VR controller buttons named `VRTrigger`, which inherits from InteractML's core input handling parent class `CustomController`. Once a `VRTrigger` is connected to a 'Teach the Machine Node' or 'ML System Node', an instruction is displayed on the VR console with information about which button to press on the VR controller to trigger a node action (Fig. 38). This way, the user can perform InteractML's core model steering tasks from within VR without the need to remove the headset.

3.6 Feature Selection in InteractML

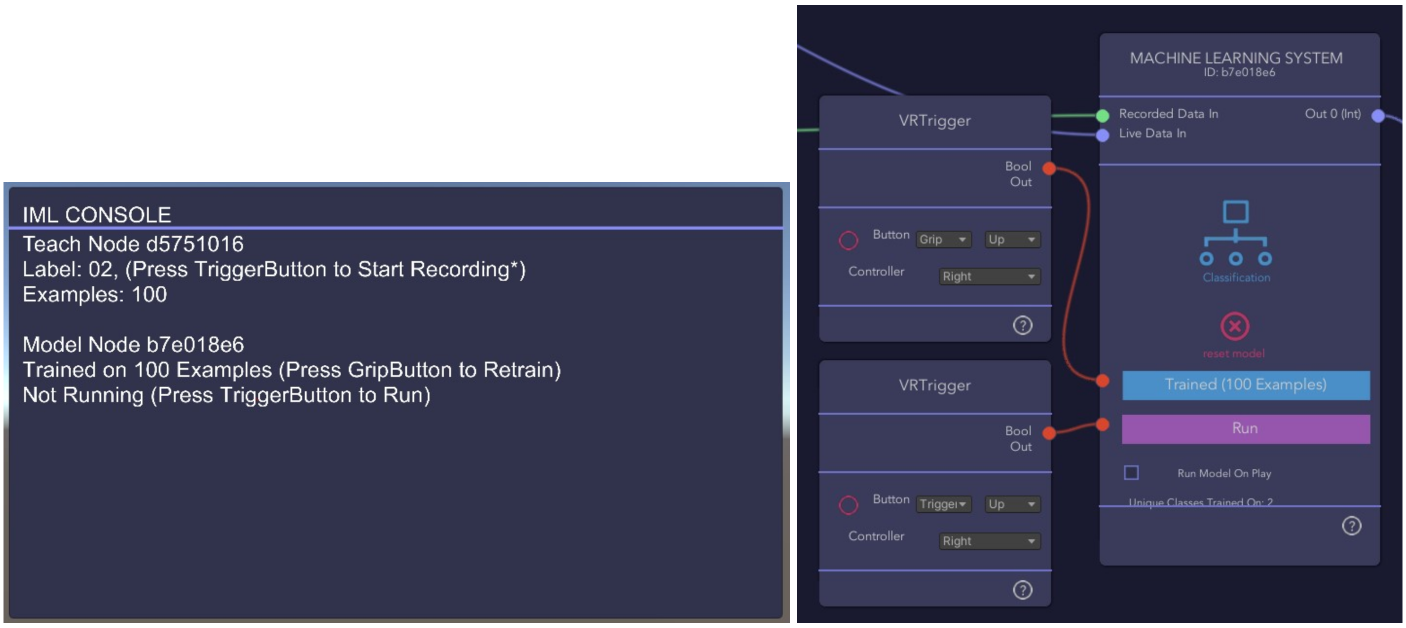
One of the game engine components that provides support for the programming of movement is the 'transform' in the virtual scene. Transforms are engine compo-


IML CONSOLE
Teach Node d5751016
Label: 02,
Examples: 100

Model Node b7e018e6
Running

Output: 02

Figure 37: InteractML's VR Console. (Left) The in-game console displaying the state of training data recordings and ML model of an IML Graph. (Right) The corresponding IML Graph.


IML CONSOLE
Teach Node d5751016
Label: 02, (Press TriggerButton to Start Recording*)
Examples: 100

Model Node b7e018e6
Trained on 100 Examples (Press GripButton to Retrain)
Not Running (Press TriggerButton to Run)

Figure 38: InteractML's VR Console displaying VR input details. (Left) The in-game console displaying the state of training data recordings and ML model of an IML Graph, together with which buttons from the VR controller will trigger node actions. (Right) The corresponding 'ML System' node with two 'VRTrigger' nodes connected to its main buttons for 'Train' and 'Run' the model.

nents that change the position, orientation, and scale of an object in a coordinate system. Transforms can be applied to any object in a virtual scene, such as models, cameras, lights, etc. In Unity, they are the basic component of any game object [300].

Game engines usually provide functions and tools that allow developers to manipulate transforms easily and efficiently. This manipulation can be done via code or via the editor. Therefore, transforms are a fundamental concept in game engines that enable the programming of movement in virtual scenes.

Transforms can be utilised by supervised learning algorithms as the basis for features describing movement. In supervised learning, a feature is an abstraction that pre-processes data into a meaningful numerical value to improve machine learning results. Features are usually derived from existing human knowledge, such as knowledge extracted from audio signal processing to derive fourier transforms [30], or from computer vision research representing motion as the positive absolute difference between an object position in space over time [190].

Since game creators would use transforms to describe movement in their games, it makes sense in the context of supervised learning features to use transforms as the basis for our features. InteractML offers five core movement features: position, rotation, velocity, distance between features, and a window of features.

3.6.1 Position and Rotation

The position feature exposes the field `GameObject.Transform.Position` that returns a vector with x, y, and z coordinates, each specifying the spatial location of the object. The rotation feature exposes the field `GameObject.Transform.Rotation` that returns a quaternion vector x, y, z, and w. Quaternions are mathematical representations of a rotation that can be difficult to understand for game creators, that is why InteractML also offers a rotation feature in Euler angles [58] that returns a vector in x, y and z coordinates, in which every dimension is in the range [0, 360] that could map better to known rotation degrees values. However, we disencouraged the use of Euler angles since they suffer from gimbal lock, which means that one degree of freedom is lost when two rotation axes align. This can cause sudden and unpredictable changes in the orientation of an object. This can cause confusion and inconsistency in the representation of rotations. Both the position and rotation features can be calculated in global space (i.e. the coordinate origin is the virtual world space) or in local space (i.e. the coordinates of origin are relative to the parent gameobject in the engine hierarchy). We decided to offer the position, the quaternion rotation and the euler rotation as nodes for feature selection (Fig. 39 because creators might want to experiment with features and they match the symbolic representation of the objects in the Unity editor (Fig. 40).

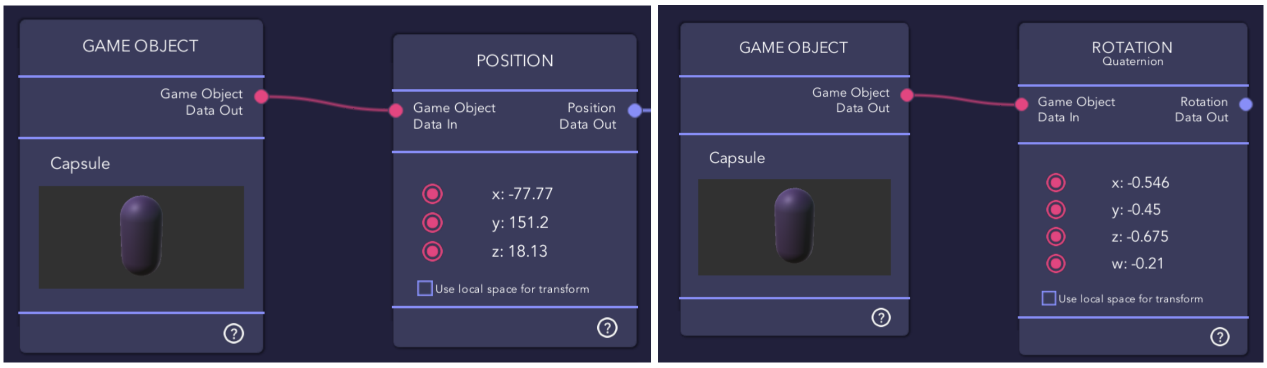


Figure 39: Position and Rotation Nodes in InteractML node interface

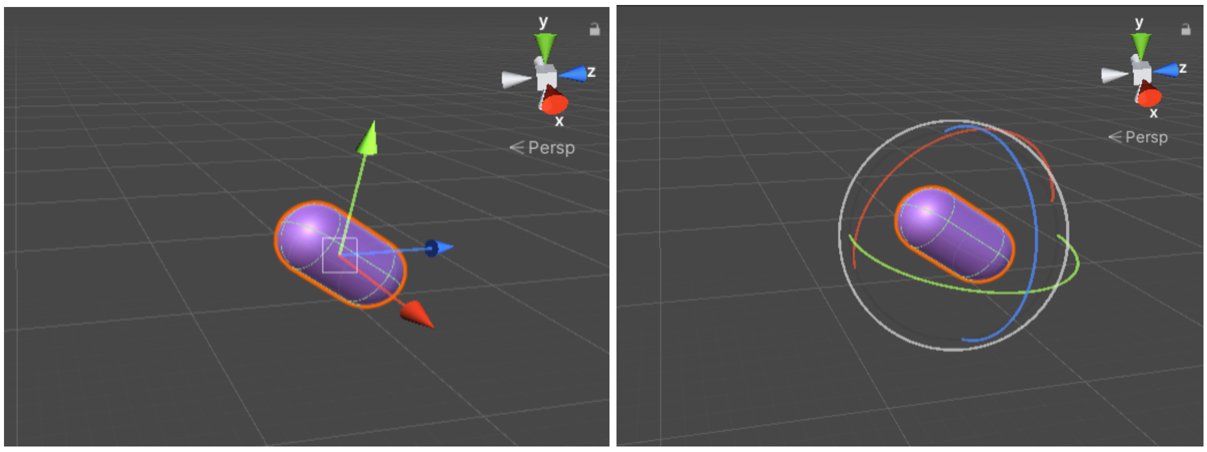


Figure 40: (Left) Position and (Right) Rotation as visually represented in the game engine coordinate space

3.6.2 Velocity of other features

The velocity feature calculates the first order derivative of whichever feature is used as an input. The first order derivative in regards to movement is the rate of change of one variable with respect to another variable, following the formula $\vec{v} = \frac{x_f - x_0}{\Delta t}$, where velocity is a vector quantity that describes the rate of change of value of a feature with respect to time, where $\vec{v}$ is the velocity vector, x_f is the final position, x_0 is the initial position, and Δt is the time interval in between two frames.

For example, if the variable is the position of an object, then the first order derivative is the velocity of the object, which tells how fast and in what direction the object is moving. If the variable is the velocity of an object, then the first order derivative is the acceleration of the object, which tells how fast and in what direction the velocity is changing. Hence, participants can use this feature node with any other feature node.

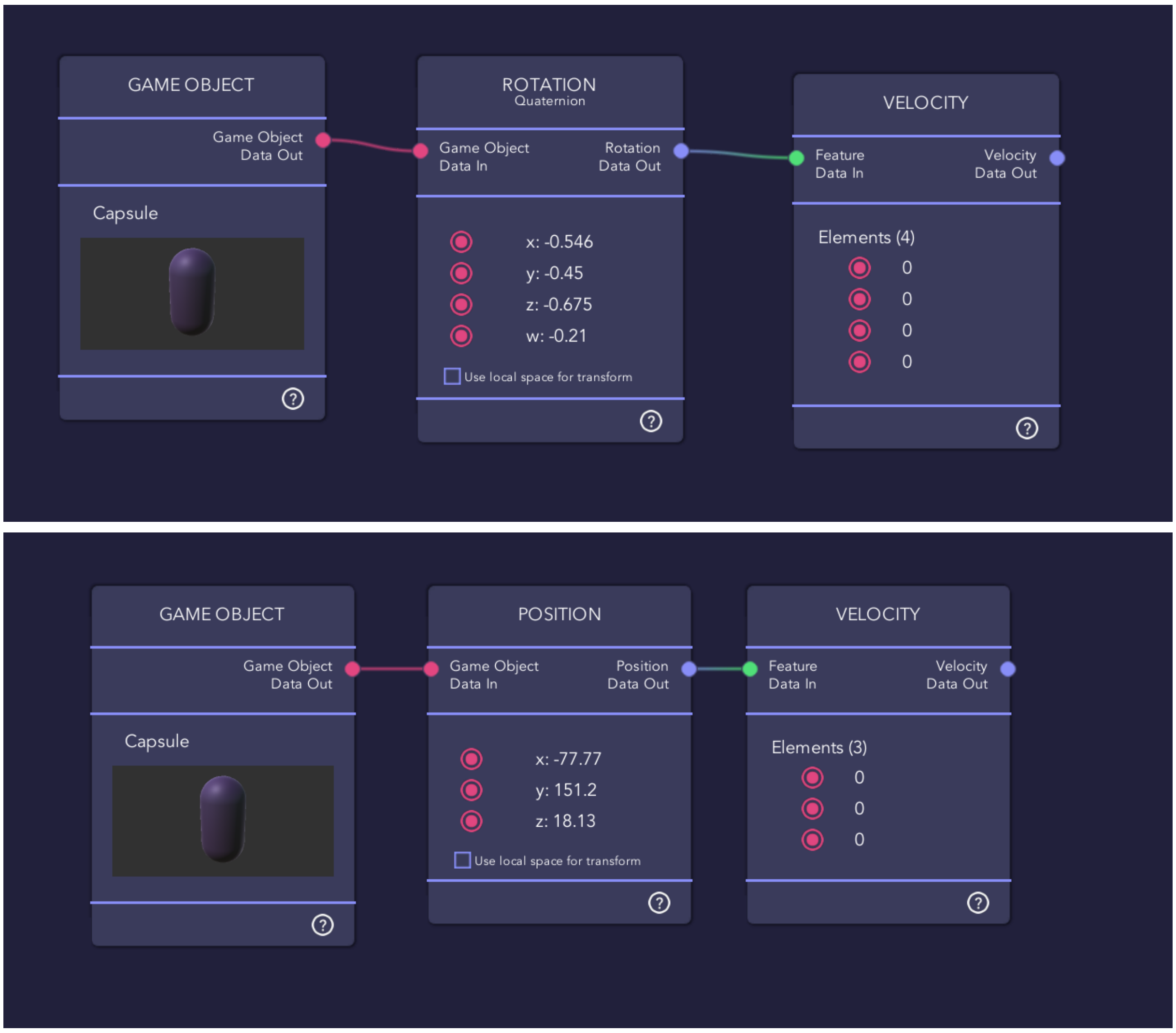
We decided to offer the velocity node as the first order derivative of any other feature to simplify feature selection, because the alternative would have been to offer specific nodes for the 'Velocity of the Position', 'Velocity of the Euler Rotation' and 'Velocity of the Quaternion Rotation' as three separate nodes, whereas now it can all be selected with the same node (Fig. 41). Similarly, if the user wants to calculate a the acceleration of a feature, which is the second order derivative, the user can join two velocity nodes together instead of selecting a specific 'Acceleration' set of nodes.

3.6.3 Distance between features

The distance between features calculates the euclidean distance between two features, following the formula $d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}$, where where d is the distance, (x_1, y_1, z_1) and (x_2, y_2, z_2) are the coordinates of the two points.

In the case of using more than two features, the first feature is used as the origin when calculating each euclidean distance. For example, using the distance function is possible to calculate the openness of the arms by having both hand positions as inputs. Similarly, one can calculate the distance between the head and the hands by having the head as the first input, and each hand as the second and third inputs. In this case, the node outputs two euclidean distances beginning at the head and ending at each hand (Fig. 44).

We decided to offer a configurable euclidean distance node to simplify feature selection, because the user can use one node to visually calculate relative features to one another. For instance, this node could be used to visually calculate a distance relationship between the head and the hands (Fig. 44), as well as a distance relationship between the palm of the hand and the fingers for hand-tracking gesturing.


GAME OBJECT
Game Object Data Out
Capsule

ROTATION
Quaternion
Game Object Data In
Rotation Data Out
x: -0.546
y: -0.45
z: -0.675
w: -0.21
☐ Use local space for transform

VELOCITY
Feature Data In
Velocity Data Out
Elements (4)
0
0
0
0

GAME OBJECT
Game Object Data Out
Capsule

POSITION
Game Object Data In
Position Data Out
x: -77.77
y: 151.2
z: 18.13
☐ Use local space for transform

VELOCITY
Feature Data In
Velocity Data Out
Elements (3)
0
0
0

Figure 41: Velocity of Position and Velocity of Rotation Nodes

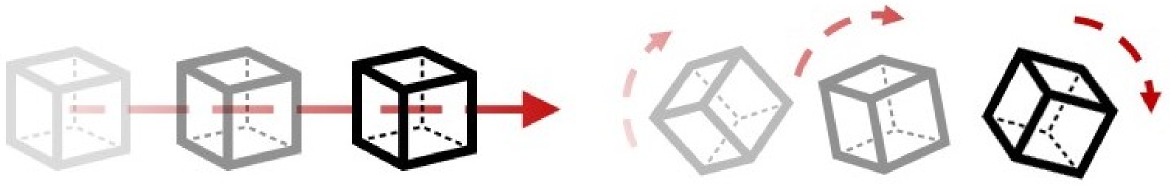


Figure 42: (Left) Representation of the velocity of a position and (Right) velocity of a rotation. The information of the initial or final position or rotation is discarded, thus only the velocity vector coloured in red is returned by the velocity node.

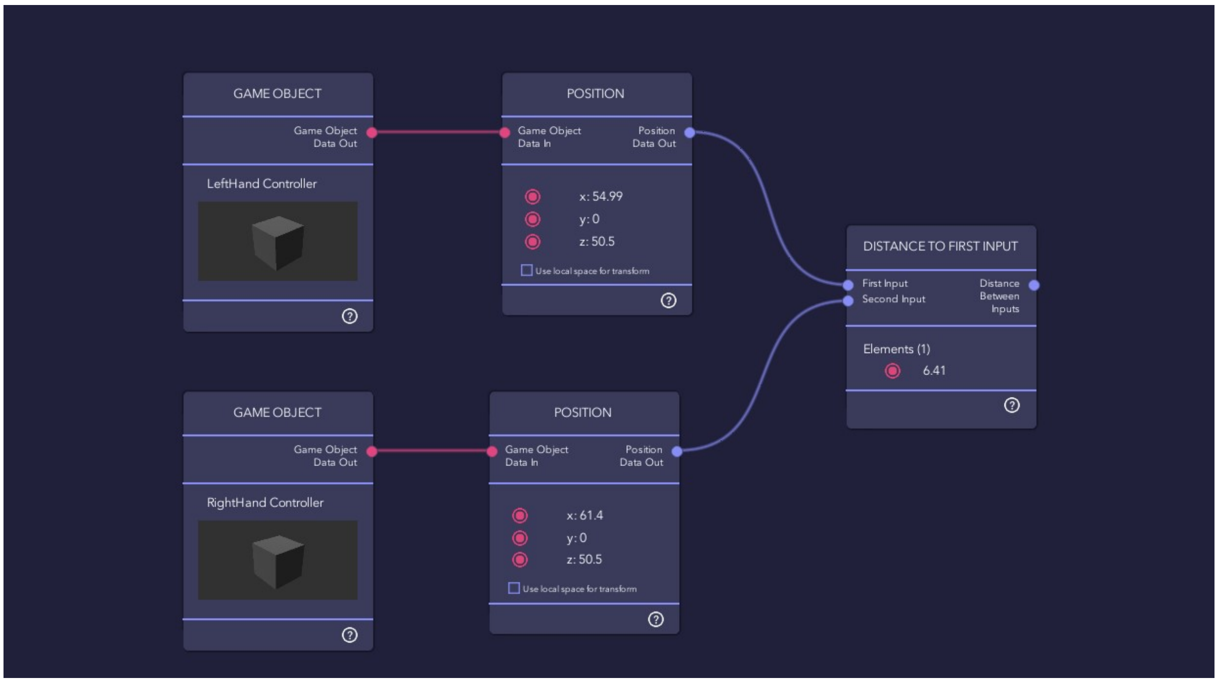


Figure 43: Distance node between two positions nodes

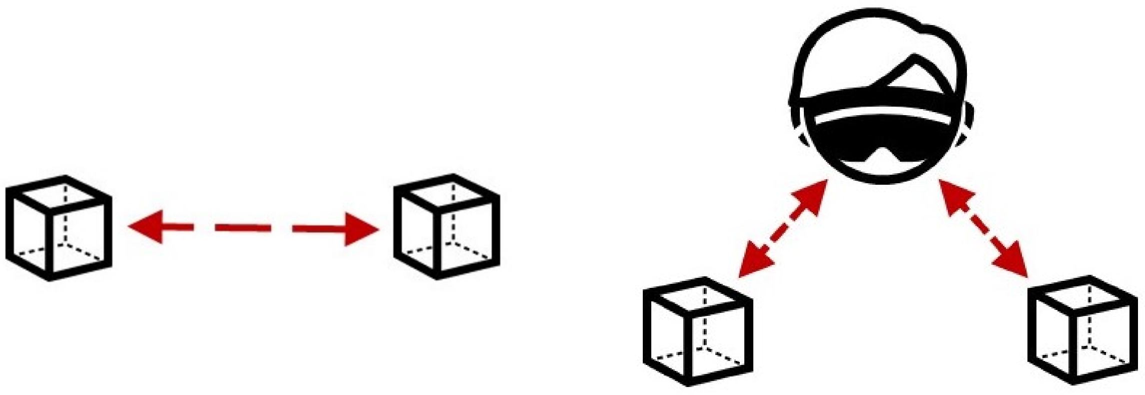


Figure 44: (Left) Representation of the distance between two positions and (Right) two cubes to the head. The information of the initial or final position is discarded, thus only the distance amount coloured in red is returned by the distance node.

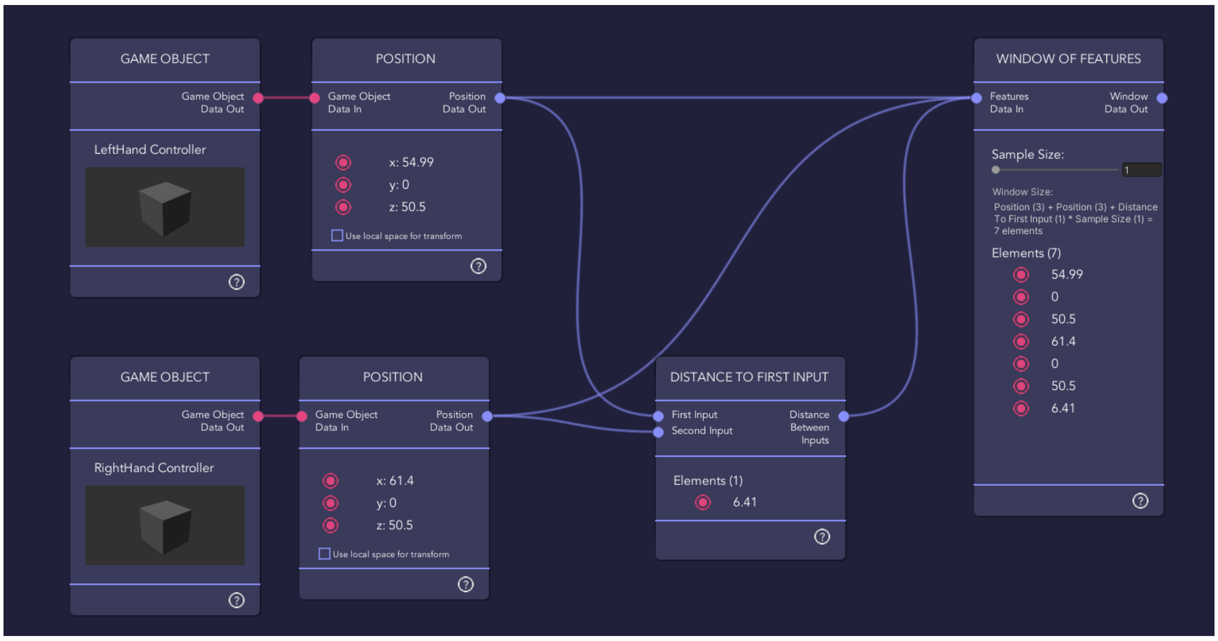


Figure 45: A Window of Features Node including the position of the left and right hands, and the distance between the two hands. The window is configured for one sample only, hence only a frame of time is taken into account.

3.6.4 Window of Features

The window of features creates new features from existing ones by using past values of the data. In this sense, movement data will be computed as a form of time series data, where the window of features can help capture the patterns in the movement over time. For instance, with the window of features it is possible to create a time series of positional data from a VR controller in the past second, minute, hour, or any other time interval. The way to do this is by specifying a sample size on the node itself. For reference, the sampling rate of InteractML is one sample per frame, with the update rate fixed at 16.7 milliseconds (60 frames per second refresh rate). Hence, a sample size of 60 is equivalent to a second of time. This can help to distinguish different types of movements, such as walking, running, jumping, etc.

We decided to design this node as a key component of time segmentation for the classification and regression algorithms. Windowed-time segmentation is a challenging problem to automate in machine learning reasearch [293], and any given movement can be segmented in many different ways (Fig. 46). Thus, we decided that the best judges for the quality of the recognition are the creators and attempting to automate this process could negatively affect (a) the quality of the creator's interaction, (b) the sense of control during feature selection, and (c) the mental model of the creator as it learns about ML while using the tool [73].

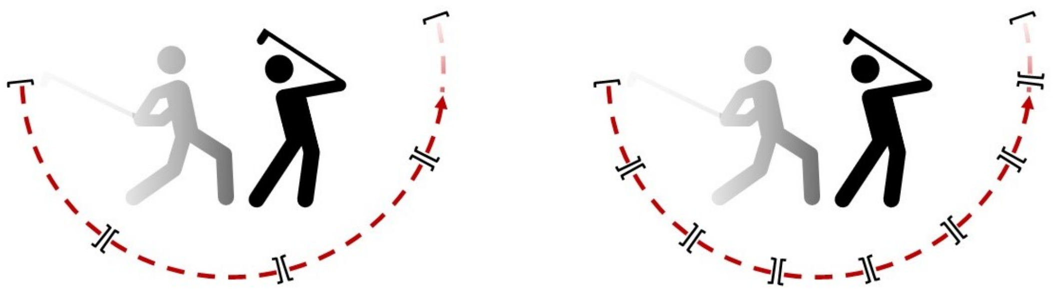


Figure 46: Red lines are single data samples. Black bars are the boundaries of the windows. (Left) Representation of a window of features of sample size 6 and (Right) sample size 3 for a swinging motion. The sample size determines the size of the window in regards to time. The higher the sample size of the window, the more quantity of information is stored in each window, and the less windows needed to describe a movement

3.7 Model Selection in InteractML

By combining features, users can symbolically describe what is important about their movements to better teach their machine learning models. However, it is also important to select a suitable machine learning algorithm that will correctly learn from the input features.

InteractML implements the following learning algorithms: (1) classification with k-nearest neighbour [53], (2) regression with a multilayer perceptron [86], and (3) time series analysis with dynamic time warping [17]. Each of these learning algorithms can be selected in the form of a 'Machine Learning System' node on the graph window. Every algorithm is implemented from Rapidlib, a C++ machine learning library developed as part of the European project Rapidmix with contributions from researches at Goldsmiths, University of London [50]. Rapidlib was developed to prototype light-weight interactive supervised learning models with a variety of sensors in a diversity of platforms via porting of the C++ library into the target system [320]. For InteractML, we compiled the C++ code into a dynamic link library (.dll) file and wrote a C# wrapper compatible with Unity3D.

Each algorithm node is design in a similar way with two input pins, variable output pins depending on training data labels, and three buttons to control the model state (Fig. 47). The reason we chose this design is to facilitate experimentation with feature selection and training data recording. In terms of feature selection, user might decide to record data with a set of features pulling information from one gameobject, but want to run inference on the same set of features with a different gameobject. In terms of training data collection, the user might prefer to record different movements on different 'Teach the Machine Nodes', and connect all of them or a subset to the ML System node to train. This flexibility is one of the key benefits of a visually scripted ML workflow such as the one we designed for InteractML.

3.7.1 KNN Classification

The k-nearest neighbour (kNN) algorithm is a supervised learning classifier that uses proximity to make classifications or predictions about the grouping of an individual data point [53]. The kNN algorithm assumes that similar data points are close to each other in the feature space. It works by calculating the distance between a query point and all the training samples, then selecting the k closest training samples (i.e. nearest neighbours) based on the calculated distances between them. The algorithm assigns the query point to the class that has the majority of the nearest neighbours.

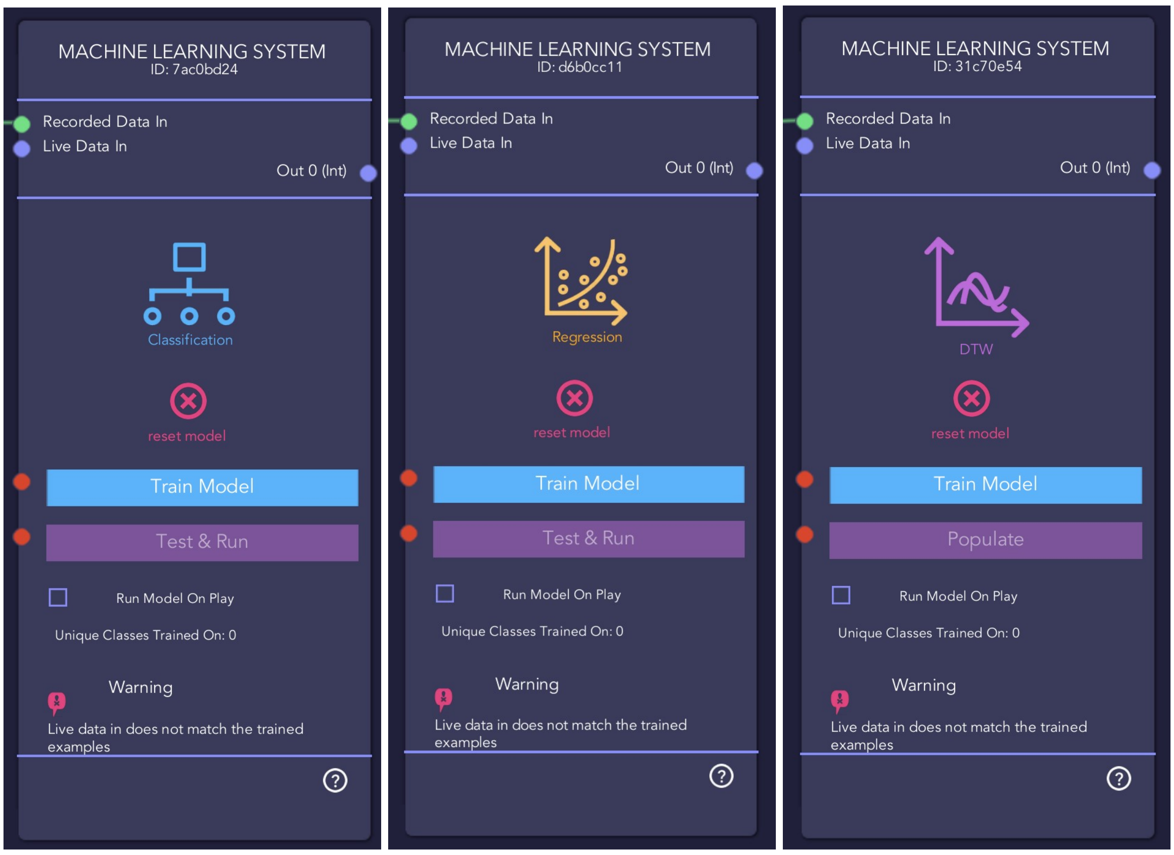
For example, consider a problem where we want to classify human hand movements as either moving to the left or to the right. We can represent each movement by a set of features, such as the position and velocity of the hand. Given a new hand movement with an unknown direction, the kNN algorithm would calculate the distances between this movement and all the known movements in the training set. It would then select the k nearest movements and assign the new movement to the class (left or right) that has the majority of the nearest neighbours.

The decision boundaries in the kNN algorithm are determined by the local geometry of the distribution of the data in the feature space and their relative distance measures [53]. As a result, the decision boundaries can be nonlinear and non-smooth. The smoothness of the decision boundaries can be influenced by the choice of the number of neighbouring data points in the parameter k. This process is known as 'parameter tuning', and as k is modified, the decision boundaries could become smoother, thus potentially reducing overfitting and improving generalisation [53]. To simplify user interaction, InteractML hides any parameter tuning options, and instead has a default value of '3' for the k parameter, which is the same default value offered by the rapidlid library.

In terms of training time, kNN requires no training time. KNN is a lazy learning model that does not generalise the data in advance. For inference time, it scans the entire dataset to predict the class of a test sample by finding the closest class based on a distance metric [53]. The time complexity of the kNN algorithm is O(nd), where n is the total number of data points in the training data, and d is the total number of features in the dataset [53]. Hence, kNN tends to be slow with large datasets because it scans the whole dataset to predict and it is more appropriate for small datasets to reduce friction in the IML loop.

3.7.2 Multilayer Perceptron Regression

The regression node in InteractML implements a multilayer perceptron. The multilayer perceptron (MLP) is a type of feedforward artificial neural network (ANN) that consists of multiple layers of nodes, including an input layer, one or more hidden layers, and an output layer [86]. Each node in a layer is connected to every



(a) (b) (c)

Figure 47: ML System nodes. (a) Classification Node, (b) Regression Node, (c) Dynamic Time Warping (DTW) Node.

node in the adjacent layers, making it a fully connected network. MLPs are used for various tasks, such as classification and regression problems, and can model complex non-linear relationships between inputs and outputs. In InteractML, the MLP algorithm is only used for regression tasks, where the output of the algorithm is continuous instead of discrete, unlike in classification.

For example, consider the problem of processing human hand movements when moving to the left or to the right. Given a new hand movement with an unknown direction, the MLP algorithm would infer a new output based on the trajectory of the training dataset, instead of assigning it to a known class as kNN does. So, if we assume a linear trajectory in a training dataset where the movements hand to the left, centre and right are labelled as '-1', '0' and '1', moving the hand further down to the left that what was known would output '-1.5' Similarly, moving the hand mid-way between the known 'left' and 'centre' would output '-0.5'.

In an MLP, the input layer receives the input data, and the output layer produces the final predictions. The hidden layers in between perform transformations on the input data using activation functions. The most common activation function used in MLPs is the sigmoid function [86], thus that's the one implemented in rapidlib. The learning process in an MLP involves adjusting the weights of the connections between nodes to minimise an error function, which measures the difference between the predicted outputs and the actual target outputs. This is typically achieved using the backpropagation algorithm, which calculates the gradient of the error function with respect to each weight and updates the weights accordingly [86, 110]. MLPs have a theoretical universal approximation capability, meaning that MLPs with a sufficient number of hidden nodes can approximate any continuous function [86].

Selecting the number of hidden layers or the activation functions would affect the curvature of the decision boundaries, and it is a form of parameter tuning. To simplify user interaction with InteractML, a default neural network with one hidden layer and the sigmoid activation function is created for every regression node placed on the IML graph.

Training a neural network using backpropagation involves a forward pass and a backward pass, and the training time complexity increases considerably with each layer and node on the network because of the fully connected relationship between layers [86]. Therefore, MLP is more time-consuming to train than kNN, especially for large datasets. While training an MLP is rather time-intensive, inference time can be relatively 'cheap' since all weights have been calculated and neural network inference time is less sensitive to the amount of training data and feature dimensionality than the kNN algorithm.

3.7.3 Dynamic Time Warping Classification

Dynamic Time Warping (DTW) is an algorithm used for measuring the similarity between two temporal sequences [17]. DTW works by finding the optimal alignment between two time series, that may have different lengths or speed via non-linear stretching or shrinking along the time axis, hence the name ‘warping’ [17].

Unlike kNN and the MLP, DTW inherently processes data from a ‘time’ perspective. That is, every single data point is actually a series of points. For example, five seconds of the position of a hand would be processed as five seconds of independent points with kNN and the classifier wouldn’t be able to process time without a window of features node. However, DTW wouldn’t need the window of features as five seconds of data would be understood as a time series to compare. And because of its time ‘warping’ capabilities, a gesture drawing a circle in mid-air that takes 5 seconds can be compared with another slower sample of drawing the same circle mid-air that takes 15 seconds. Despite DTW’s inherent time representation usefulness, it has limitations, because while the algorithm can compare gestures of different time lengths, it also requires explicit segmentation of when the gesture begins and ends. This segmentation problem is a well documented problem with time series analysis algorithms [293]. This problem isn’t present when a kNN is used in conjunction with a window of features, as the algorithm can run in the background processing movement data without the user explicitly signalling the beginning and end of a movement.

Additionally, DTW can be computationally expensive in training and inference time, especially when dealing with large datasets or long time series. The time complexity of DTW is O(m * n), where m and n represent the length of each sequence [17]. Therefore, large training datasets with lengthy sequences may highly impact users using DTW in their IML loops.

Yet developers do not need to know anything about the algorithms themselves to begin using InteractML; they must only understand (1) the interaction flow required to build a model (i.e., record data, train a mode, run the model) and (2) which type of machine learning algorithm (classification, regression, time series analysis) they wish to use. InteractML comes with numerous examples and tutorials to aid developers with these tasks.

3.8 Model Steering in InteractML

3.8.1 Variables and Data Types Selection

Users can create variable nodes in the IML graph of the following types: Boolean, Integer, Float, Vector2, Vector3, Vector4 and Array (Fi.g 48. Variable nodes can be connected to any pin that accept the corresponding data type, for example to specify a label when recording training data (Fig. 49 Left) or visualising the inferred output from a machine learning model (Fig. 49 Right).

We designed the variable nodes in this manner to allow for as much flexibility as possible when selecting labels or controlling buttons with variable output. Users

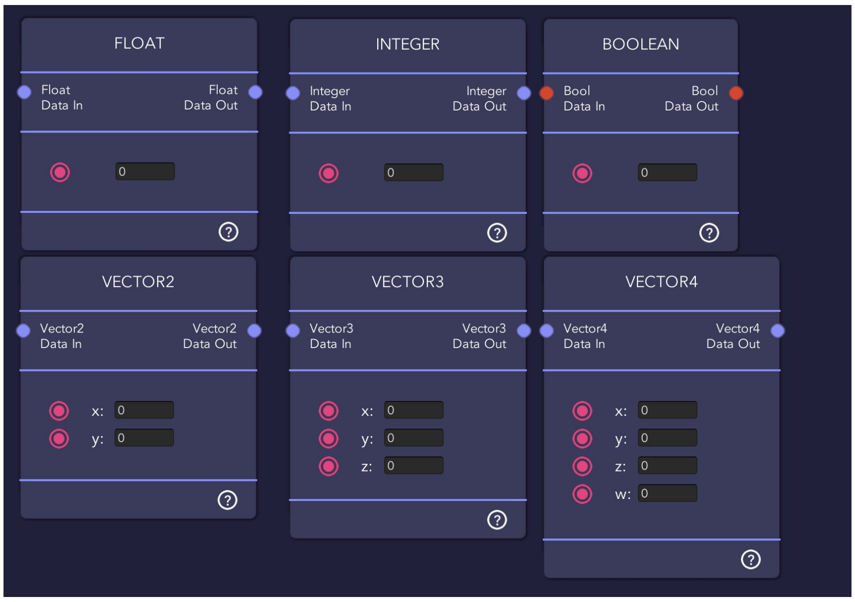


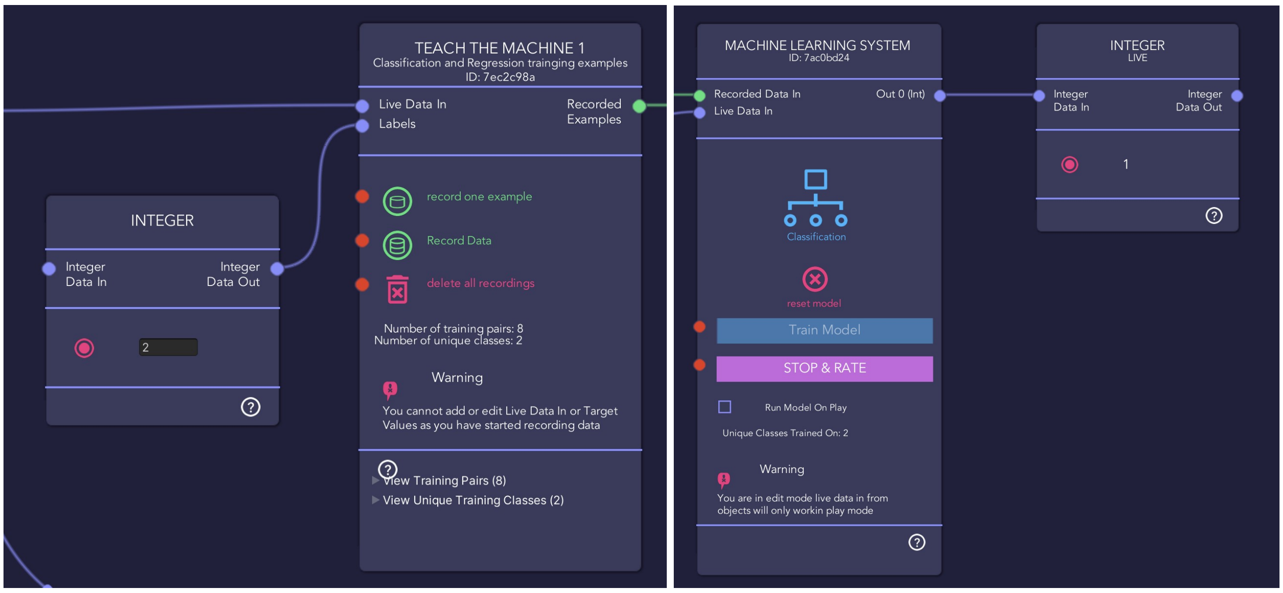
Figure 48: Variable Nodes available in InteractML

can create a variable node and modify its value, and as soon as they connect it to a suitable node the value of the variable node will start 'flowing' onto the other node. Users can even record training data using variables as training samples instead of movement features. This is because all movement feature nodes outputs the same data structures that variables output. For instance, a position node outputs the same kind of data structure that a Vector3 node outputs. On the visualisation side, to ensure that the user understands when a variable is 'reading' a value instead of 'writing' a value, we removed the darkened background that indicates modifiable capibilities when the variable node receives and input. For example, on the right part of figure 49, an integer node receives the output of the model as an input and, consequently, visually indicates that the integer value contained in the node cannot be modified by the user by changing the darkened background –as opposed to the left side picture.

3.8.2 Training Data Collection

To record training data, InteractML offers a 'Teach the Machine Node' with two input pins: one for features and one for labels (Fig. 50 Left). Via this node, the user can record data into training pairs of feature data and labels. For example, the feature 'Position' of the hand gameobject can be paired with the integer label '1'. Then the user can click on the button 'Start Recording' to collect example pairs of positions with label '1'. The number of collected examples is displayed on the node and the user can stop collecting data by pressing the button 'Stop Recording'. The recorded data can then be viewed by opening a scrollable layout at the bottom of the node (Fig. 50 Right).

We designed this node to offer flexibility during data recording. The node offers a



Number of training pairs: 8
Number of unique classes: 2

You cannot add or edit Live Data In or Target Values as you have started recording data

Unique Classes Trained On: 2

You are in edit mode live data in from objects will only workin play mode

Figure 49: Variables being used in conjuction with other nodes. (Left) An integer variable node is used to specify the label of the training data to record. (Right) An integer variable node is used to display the inferred output of a model.

button for recording one sample of data, because users might want to control very carefully which poses or points in space they want to record onto the dataset. The node also offers a button to record one example every frame, and this way the user can record as much movement nuance as they desire and stop recording when they are satisfied. We also didn't want to constrain creators with the amount of features or labels they choose. Users can experiment with as many 'Teach the Machine Nodes', movement feature nodes and variables as they steer their model with their own data.

3.8.3 Model Training and Running States

Before evaluating a model, the user needs to train the model with the data that has been collected. All of the 'ML System Nodes' display buttons to (1) train, (2) re-train if needed, (3) run the model, and (4) reset the model (Fig. 47). We describe each of the model states below:

- **Untrained:** Models that have been just created or reset are untrained, hence the user cannot run them –the 'Run' button is greyed out and the user can't click on it (Fig. 47). The user can train model once a 'Teach the Machine Node' is connected to it (Fig. 51).
- **Training:** Models that are training with very large datasets will display this state as is training on a different task. The model cannot run nor be retrained.
- **Trained:** Models that have been trained can now run and will display the

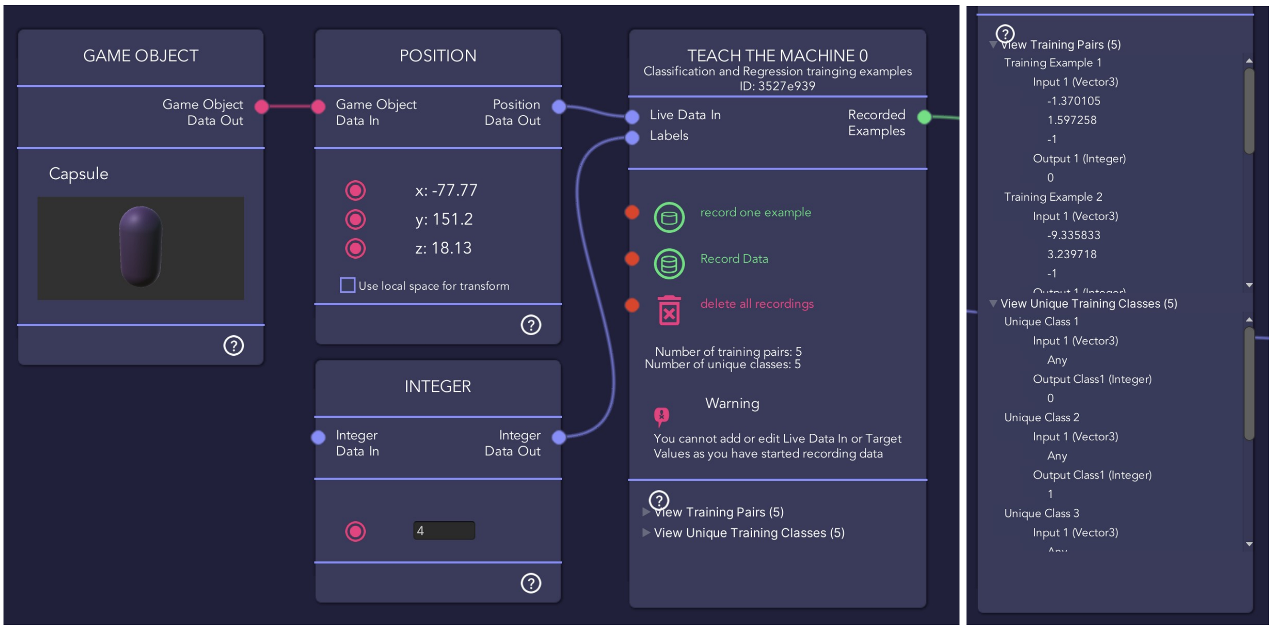


Figure 50: Training Data Collection Node, called 'Teach the Machine Node'. (Left) The node receiving input from a position feature with an integer label, and displaying all the buttons that the user can press. The user can record one example, trigger data recording until the user wants to stop, or delete all recorded examples. (Right) The detail dropdowns for the training data recorded and for the unique classes recorded in the node.

correct number of output pins based on the label structure of the training dataset (Fig. 51).

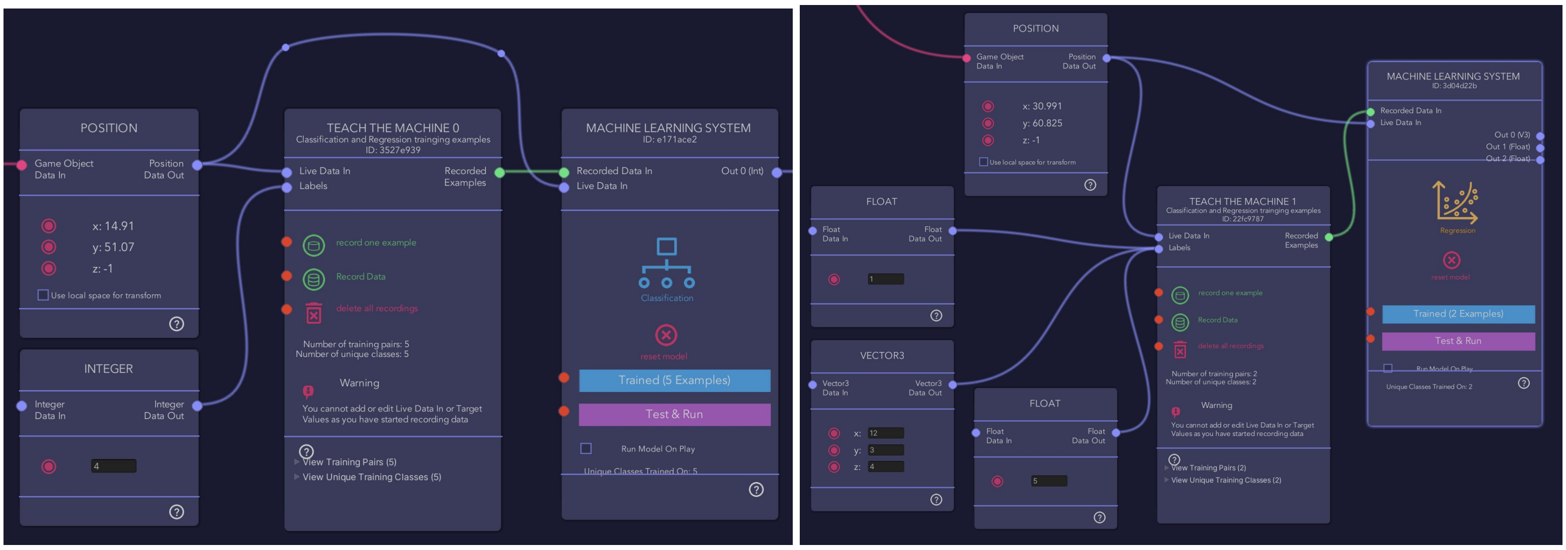
- **Running:** Models that are running infer in real-time an output that can be displayed via variable nodes (Fig. 52). In this state, models can be stopped or reset, but cannot be retrained.

And the node actions are as follows:

- **Train:** Enters the model into the 'Training' state and 'Trained' states.
- **Re-train:** Enters the model again into the 'Training' and 'Trained' states if the node detect changes to the 'Teach the Machine Nodes' connected to it.
- **Run:** Enters the model into the 'Running' state.
- **Reset:** Re-enters the model into the 'Untrained' state via re-instantiating the internal rapidlib model.

3.8.4 Model Evaluation in InteractML

How well a model learns can be calculated via the accuracy of inference. Inference is the process of applying a trained ML model to a dataset and producing an output



(a) (b)

Figure 51: Training model nodes. (a) A ‘Teach the Machine Node’ with a training dataset recorded is connected to a Classification model node that is trained displaying the one output pin according to the training dataset label. (b) A ‘Teach the Machine Node’ with a training dataset recorded with one Vector3 and two Float variables is connected to a trained Regression model node displaying more output pins because of the higher number of labels.

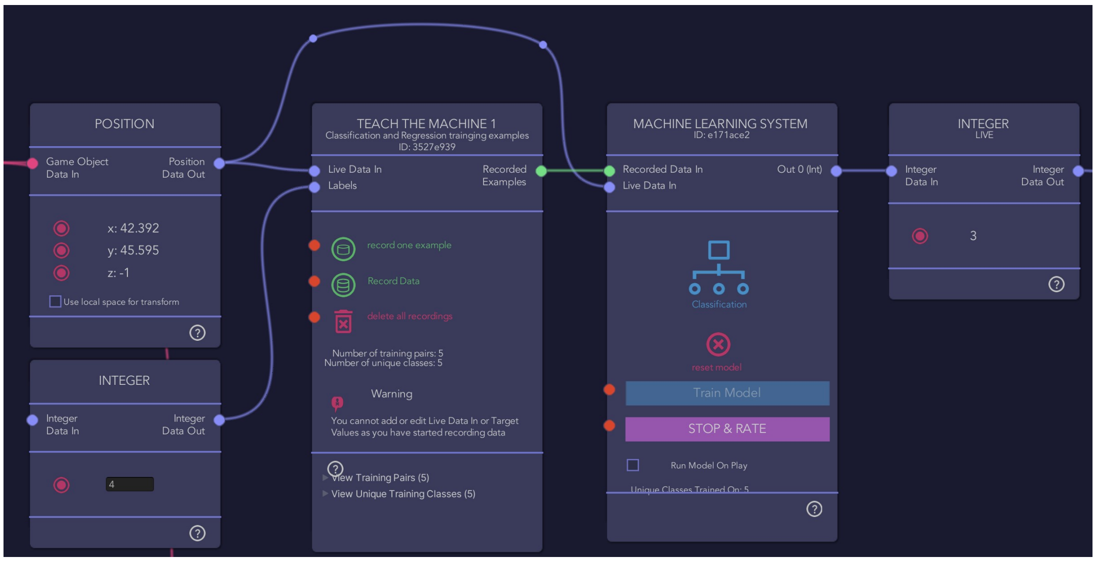


Figure 52: A classification model node is running displaying real-time inference via a variable node.

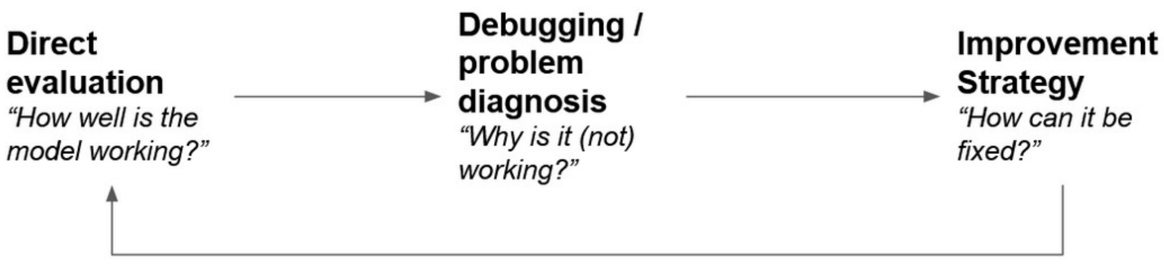

Direct evaluation
"How well is the model working?"
Debugging / problem diagnosis
"Why is it (not) working?"
Improvement Strategy
"How can it be fixed?"

Figure 53: Model evaluation can be broken down into three sub-stages: (1) direct evaluation, (2) debugging or problem diagnosis, and (3) improvement strategy [264, 95]

or prediction. In InteractML's case, the inference dataset will always be the live input data coming from the selected features, and the inference output is specified by the labels recorded in the training datasets. For example, if the user recorded a training dataset in which the position 'left hand up' is labelled as '1', and 'left hand down' is labelled as '2', the model will infer the output '1' when the hand is up, and '2' when hand is down. Accuracy is a metric for evaluating the performance of a classification model. It is the fraction of inferences that the model got right out of all the ones it made. For example, if a model correctly infers 90 out of 100 examples, its accuracy is 0.9 or 90%. Accuracy is a simple and intuitive way to measure how well a model can identify the correct class for a given example. However, given the qualitative nature of this thesis, we did not investigate the quantifiable value of accuracy, but rather its qualitatively perceived value [73]. Because game designers use their own qualitative assessment to prototype controls before engaging in playtesting or quality assurance processes, we consider that InteractML should support their existing way of assessing game feel via direct qualitative assessment, something we refer to throughout the thesis as direct evaluation.

User evaluation of each iteration in the model steering loop can be divided into three further sub-stages which form a sub-loop by themselves. The model evaluation sub-stages are (see Fig. 53):

1. Direct evaluation [73]: During direct evaluation, the user 'runs' the model while performing movements to directly observe the numerical model output to assess whether the performance is satisfactory [73]. In InteractML, we prompt a class by class evaluation structure via interface prompts (Fig. 54).

2. Debugging or problem diagnosis [95]: After an unsatisfactory direct evaluation, a debugging or problem diagnosis phase is entered where the user attempts to assess what causes are affecting the model numerical output during live inference [95]. For instance, users might find that their training dataset is lacking a label or more examples on a certain label, or that the selected features don't describe well enough the movement and a different set features is selected.

3. Improvement strategy [264]: once the problem has been diagnosed, an im-

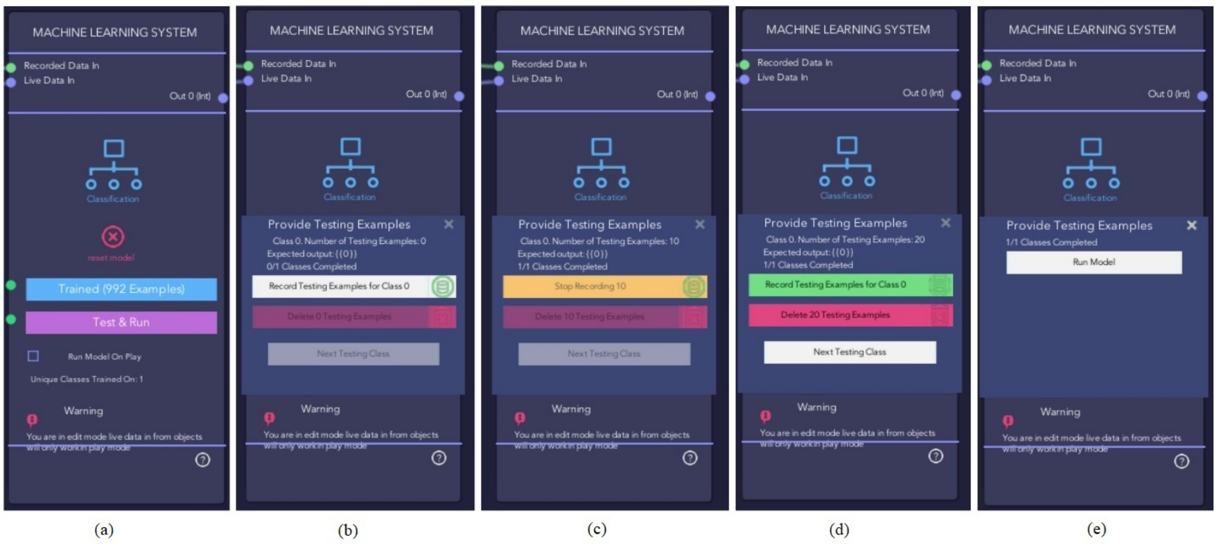

MACHINE LEARNING SYSTEM
Recorded Data In
Live Data In
Out 0 (Int)
Classification
reset model
Trained (992 Examples)
Test & Run
Run Model On Play
Unique Classes Trained On: 1
Warning
You are in edit mode live data in from objects will only work in play mode
Provide Testing Examples
Class 0. Number of Testing Examples: 0
Expected output: {(0)}
0/1 Classes Completed
Record Testing Examples for Class 0
Delete 0 Testing Examples
Next Testing Class
Class 0. Number of Testing Examples: 10
1/1 Classes Completed
Stop Recording 10
Delete 10 Testing Examples
Class 0. Number of Testing Examples: 20
Delete 20 Testing Examples
Run Model
(a)
(b)
(c)
(d)
(e)

Figure 54: Explicit testing data collection interface on model node. (a) Classification node default view. (b) The user clicks ‘RUN & TEST’ after training the model, therefore entering a structured class per class explicit testing procedure visualised as a pop-up window over the node. (c) The user clicked ‘Record Testing Examples’, which started a testing data collection in which the user movements are recorded. The user is also able to see what is the expected numerical output for that class and the live inferred output from the trained model (i.e. input: hands-up, output: stop taxi). (d) The user clicked ‘Stop Recording’ and a screen displaying how many testing examples were collected and an opportunity to delete and re-record the testing examples is given. (e) Once the user has collected testing examples for all classes, the user can exit the explicit testing interface to freely and directly evaluate the model without structured prompts. They can stop or re-train the model at any time from this moment.

provement strategy is followed to steer the model into the desired behaviour (i.e. the user might need to select a different set of features, or to provide additional training examples for a new or existing class) [264].

This evaluation loop can be performed for as many times as the user requires to achieve a subjective degree of satisfaction with the model behaviour.

The IML Workshops chapter (Chapter 5) limited InteractML’s ML algorithms to only offer the k-Nearest Neighbour (kNN) classifier[53] to both simplify quantitative comparisons during future post-hoc analysis and reduce cognitive load from participants. The kNN classifier was represented as the classification node in the IML Graph (Fig. 54.a), which accepted nodes containing training examples (called ‘Recorded Data’ on the classification node) and features expressing live movement data from the VR gameobjects (head and hand controllers).

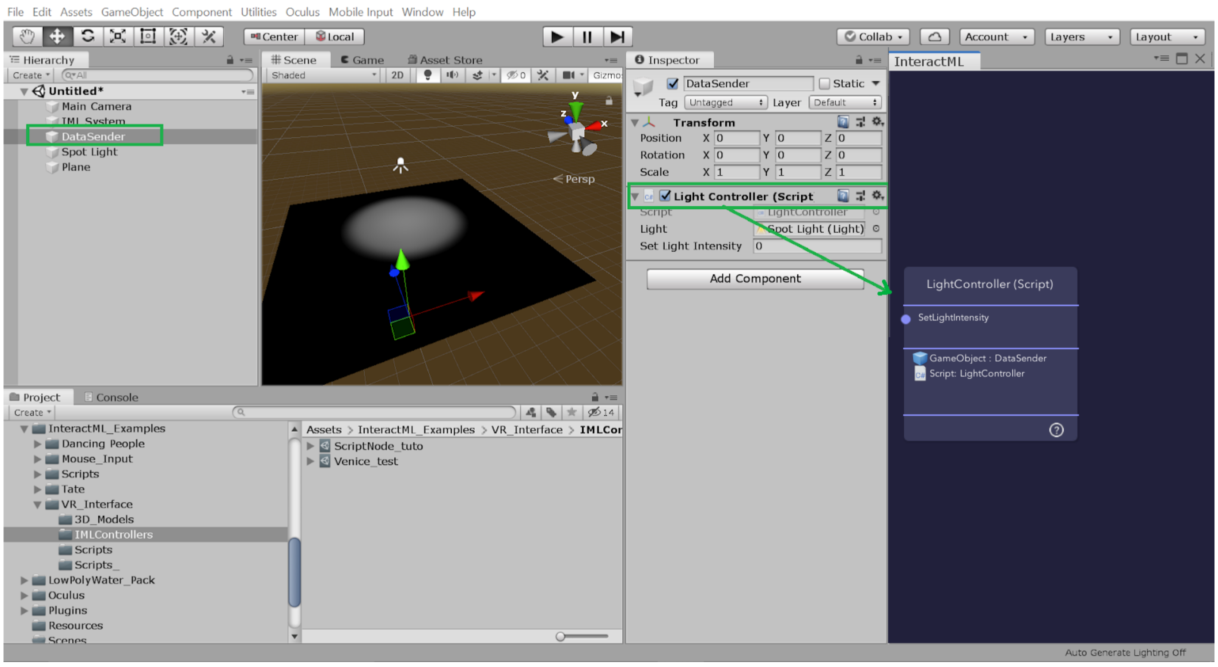
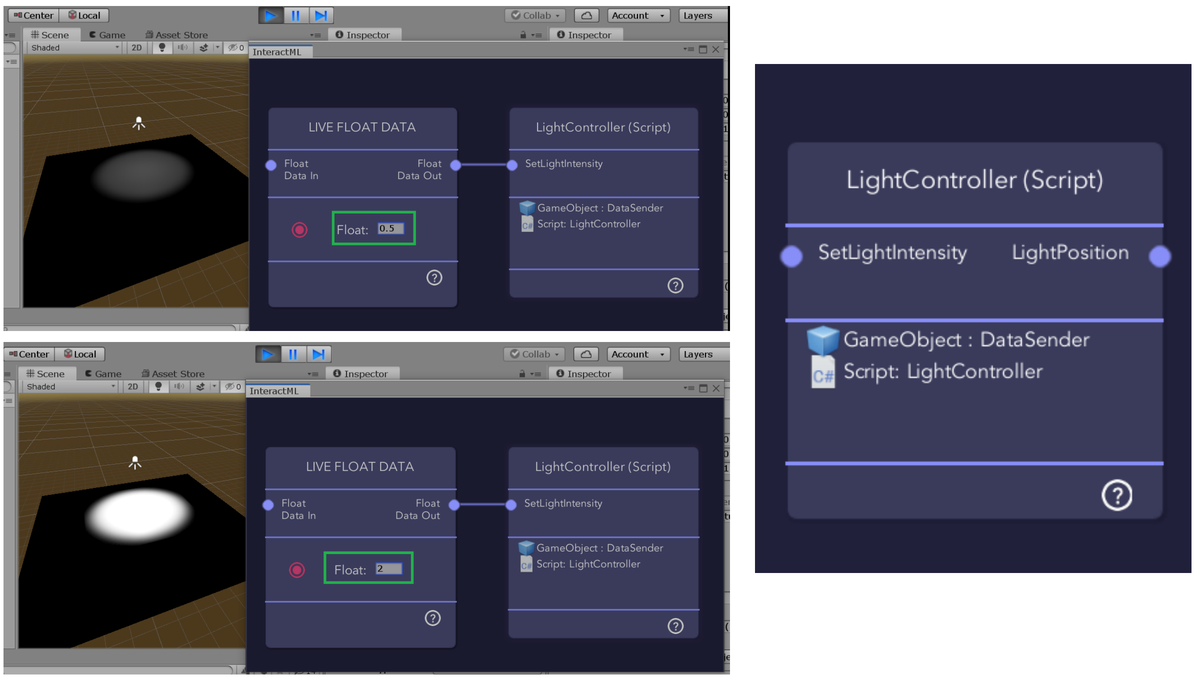


Figure 55: The user drags and drops the 'LightController.cs' script onto the IML graph to create a 'ScriptNode'



(a) (b)

Figure 56: (a) A float variable node is used to affect the light intensity of the 'LightController.cs' script in IML graph. (b) A script node with display input and output pins following the code on the LightController.cs Listing 4.

3.9 Model Output to Interaction in InteractML

The final section of any IML graph is to pipe the model output into a custom script in the game scene in order to translate model outputs into game interactions. In InteractML, this is done via the 'Script Node' which represents an existing script. For example, the user can drag and drop a Unity C# script that expects a float value to control the intensity of a light into their IML graph as a 'Script Node', that will display information about the script (i.e. name and gameobject holder) together with input pins to connect the a variable node (Fig. 55. Then, given an interaction where the movement 'hand to the left' outputs '0' and 'hand to the right' outputs '2', the user (Fig. 56.a).

Additionally, users can use 'Script Nodes' to output information from the scene into their IML graphs. In the previous 'LightController.cs' example, the user can display an output pin from the node with the intensity of the light when it is on. In order to configure how many input and output pins are displayed from a script, the user can decorate any public variable in the C# code with the attribute [PullFromIMLGraph] to display an input pin, or [SendToIMLGraph] to display an output pin (see code listing 4 and Fig. 56.b). Pins are displayed with the name of the variable together with its data type following a code reflection process in C# [189].

We designed the script node to reflect the existing way of working in Unity, in which creators would write simple scripts to prototype game behavior. Thanks to that, we allow creators to reuse existing scripts they have in place by adapting them to work with the IML Graph processing. Additionally, scripts are not a new metaphor to learn, since users already understand that in order to have an effect on a gameobject you need a script, and the API to affect a gameobject is still the Unity API.

Listing 4: LightController.cs script with an input and output pin

```
using UnityEngine;
using InteractML; // This import is needed to use InteractML instructions

public class Lightcontroller: MonoBehaviour
{
    // Displays input pin
    [PullFromIMLGraph]
    public float SetLightIntensity;

    // Displays output pin
    [SendToIMLController]
    public Vector3 LightPosition;

    // Update is called once per frame
    void Update()
    {
        // Pull data from IML Graph
        light.intensity = SetLightIntensity;

        // Send data to IML Graph
        LightPosition = light.transform.position;
    }
}
```

3.10 Model Transfer in InteractML

Since InteractML is integrated as a game engine plugin, models are transferred to exported games via Unity's regular build procedure [294]. InteractML automatically packages training data and machine learning models as JSON files and exports them into the file structure of the packaged game in the folder 'InteractML/Data'. This is then loaded into the game on run-time allowing for real-time inference on an executable. This process is automated and doesn't require any intervention from the user side, ensuring functionality parity between the game engine editor and the built game.

For instance, an excerpt of an example JSON file containing training data can be found in code listing 5 and an excerpt of a trained model can be found in code listing 6. We decided to use JSON as the serialisation format for data because it is human-readable. As it can be seen in code listing 5, user can modify their training dataset by writing or deleting examples from the dataset file. They can also choose to merge datasets together by copying and pasting them together from one file to another.

Listing 5: JSON excerpt from a file containing training data

```
[ {
    "Inputs": [ {
        "InputType": 0,
        "InputData": {
          "Values": [0.0, 0.0, -5.0],
          "DataType": 3
        }
      } ],
    "Outputs": [ {
        "OutputType": 0,
        "OutputData": {
          "Values": [3.0],
          "DataType": 1
        } } ] },
  ...
} ]
```

Listing 6: JSON excerpt from a file containing a trained classification model

```
{...
"modelSet" : [ {
```

```
          "examples" : [
             {
                "class" : 1,
                "features" : [ 61.400001525878906, 0, 50.5, 61.40000
                   1525878906, 0, 50.5, 0 ]
             },
             ...
             {
                "class" : 2,
                "features" : [ 61.400001525878906, 0, 50.5, 61.40000
                   1525878906, 0, 50.5, 0 ]
             },
             ...
          ],
          "inputNames" : ["inputs-1", "inputs-2", "inputs-3", "
             inputs-4","inputs-5",
             "inputs-6", "inputs-7", "inputs-1", "inputs-2", "inputs
                -3", "inputs-4",
             "inputs-5", "inputs-6", "inputs-7"],
          "k" : 1,
          "modelType" : "kNN Classificiation",
          "numInputs" : 7,
          "whichInputs" : [ 0, 1, 2, 3, 4, 5, 6 ]
       }]
 }
```

3.11 Discussion

Here, we discuss how the design features and interactions supported by InteractML relate to existing tools for game motion control creation, and to out-of-engine interactive machine learning tools.

3.11.1 Advances on Game Motion Control Toolkits

In supporting in-engine, visual and embodied motion control creation, InteractML makes several contributions on interaction toolkits [301, 83] and gesture recognisers [306, 176]. Unlike existing VR interaction toolkits, InteractML moves away from object-focused interactions and embraces a full movement-focused approach: the source of an action is not the object "interactable" [83], as in a VR interaction toolkit, but rather the movement itself. Additionally, InteractML can be used alongside a VR interaction toolkit.

For instance, holding an interactable ball can set to `True`, a Boolean variable in the

IML graph that will in turn run an ML System node to process user movement. We could get creative and, similarly to how input is handled currently via 'VR Trigger' nodes, creators could integrate virtual interactables to control the entire model steering process. We could imagine a basketball game with superpowers, where creators could build an interface based around throwing the basketball into a particular hoop to change the training data recording label, or to grab the basketball to begin recording movement data and throw the basketball to stop recording. This example brings more *nuance* to object-focused interactions as the player movement begins to matter when there is a direct consequence on how the player is moving and how the player can control their own movement recognition.

And since the entire IML Graph is visually scripted, our system pushes forward the state of the art of configurability and flexibility of movement-focused interaction toolkits.

The basketball example also brings one of the key differences between InteractML and current VR gesture recognition plugins–sensitivity to more than a single, prespecified controller through the ability to flexibly configure what the recognition algorithm is 'observing'. InteractML doesn't assume that there is a single 'correct' input device. Rather, it accepts any game object from the scene as the source to select movement features from –which could range from motion controls to webcams, microphones or wearables. This is a key difference from gesture recognition plugins such as "MiVRy" [176], where the VR controller is the only input method accepted.

InteractML also doesn't 'lock' the user onto an opaque set of features or a single processing algorithm, but rather offers a range of transparent features and algorithms to better serve creative needs. Transparent, flexible feature selection increases user agency, control, and potentially ML learnability, since non-experts creators can discover how different feature sets affect the quality of their machine learning inference.

InteractML allows users to perform the core of the tool interaction (i.e. the model steering process) in an in-medium embodied manner. Users can use their body movements to configure how their game interaction functions solely within VR. This encourages users to indeed bodily think, instead of clicking sliders or typing numbers on a GUI. Additionally, a fully embodied process is more direct than rule-based movement abstractions [83] or simplified gesture recognisers [306, 176]. In short, InteractML goes beyond existing systems that require users to constantly switch between mediums to tweak variables in the game engine inspector. This is done by offering a full embodied model steering process: users can bodily think about the system as a whole.

Finally, regarding processing algorithms, InteractML allows for both classification and regression tasks to be tackled. For instance, none of the gesture recognition plugins available would allow to run a regression to control the intensity of an in-game light or the strength of an attack; and none of them would allow 'free moving' as InteractML does. With the window of features node in combination with the kNN classification or the MLP regression node, InteractML users aren't required to carefully segment their movements during inference and support 'unbounded' movements such as wiggles, shakes or continuous movements such as walking or dancing. And, since InteractML also offers a DTW algorithm node for time series classification, users get the best of both worlds and can choose to 'bound' their movements to explicit segmentations with the DTW algorithm.

3.11.2 Advances on Interactive Supervised Learning

The interactive supervised workflow supported by InteractML builds on existing systems and approaches, especially the Wekinator [74]. The Wekinator affords the same stages of interactive supervised learning that InteractML, and it has been a clear influence to InteractML's ethos [74, 73]. In both systems, users can (1) select features from sensors without clear default input devices, (2) select models, and (3) perform model steering.

At the same time, InteractML deviates from and advances upon this predecessor in significant ways. Firstly, InteractML is a fully integrated into a game engine, whereas Wekinator acts as a companion app that receives and sends data to other computer programs. Secondly, InteractML's interface is visually scripted, whereas Wekinator's interface follows a traditional interface based on buttons and sliders. Thirdly, feature selection in InteractML is performed fully in-engine visually, whereas Wekinator requires users to (a) write programming code in their language of choice to select features from sensors, and (b) send the information via a network protocol known as OSC. This is one of the strongest barriers of usage for non-ML expert practitioners using Wekinator: coding features can prove difficult to creators without signal processing or OSC experience. Thus, InteractML improves Wekinator's feature selection methodology considerably because of its visually scripted and in-engine nature.

Model selection is done differently in InteractML. In Wekinator, users select a model from a dropdown, whereas in InteractML users do so visually. But most importantly, model steering has some clear differences between both systems. In Wekinator, users record their entire training dataset in the same interface and file. If the user wants to change their feature set, Wekinator requires them to delete their entire data recording or "change session". In InteractML, users can create as many 'Teach the Machine' nodes as they desire, each of them with the same or different feature configuration and file. Of course, users can delete all their prior training examples in a 'Teach the Machine' node if they wish to, but they aren't required to by the interface and can keep all their training data on one graph. Thus, InteractML offers a far more flexible design to record training data, because users can visually configure a set of different data sets, and even different feature configurations, without the need to delete any prior recordings.

Both the Wekinator and InteractML offer functionality to trigger each of the model steering sub-tasks via input devices. However, Wekinator requires users to write programming code to send a signal via the OSC network protocol to invoke the different buttons that trigger each model steering sub-stage. On the other hand, InteractML offers visual scripting nodes to handle the equivalent functionality, what is (a) more readable from the user perspective, because the user can visually see where each input device node connects to; and (b) easier to modify since users can simply disconnect and reconnect input device nodes onto other nodes to control when to record data or to start running or stopping a model.

Additionally, InteractML's core contribution of performing in-medium model steering isn't offered by Wekinator 'out of the box'. In InteractML, users can perform model steering fully in VR with the nodes that the system already provides, however in Wekinator the user would need to program all the behaviour from scratch (i.e. reading data from Wekinator via OSC code, creating a VR console and input handling of model steering stages). This contribution, together with its visually scripted nature, make InteractML stand out from prior IML systems, not only the wekinator, because to our knowledge no other IML system is fully integrated into VR and into a game engine simultaneously.

Finally, InteractML's model output to interaction is also different from Wekinator. In Wekinator, the user would need to write OSC network code in their programming language of choice to receive the output of the model as an array, and further code to pipe that array into a meaningful variable. In InteractML, the user only needs to add an attribute on their game script variable and pipe the model output visually on the IML Graph. This process is (a) more readable for the user and (b) more accessible for non-technical creators that can modify the behaviour of existing scripts in less steps than writing the equivalent code in Wekinator.

3.11.3 Limitations of InteractML

Nonetheless, InteractML, in its current state, suffers from certain limitations when compared to prior interaction toolkits, gestural recognisers and IML solutions.

Regarding VR Interaction Toolkits, the fact that behaviours are configured via rules can also offer advantages compared to a data-drive solution such as InteractML, because the rule-based model can be more transparent and visually understandable to users. Additionally, the limitation of VR Interaction Toolkits is also their strenght, since they are better suited to model object-focused interactions. That is why we suggest that mixing a system like InteractML with VR interaction toolkits could offer more nuance to game creators working with object-focused interactions.

Furthermore, currently training data in InteractML is not visualised beyond the numerical dropdowns from the 'Teach the Machine' node or the JSON files where the training data is stored. The gestural recognition plugins that we introduced in the background section 2.3 usually offer a very concrete shape to be drawn and provide examples against (e.g. a geometrical shape, a stroke, a letter or a symbol), and gestures on the training dataset are usually displayed to the user in 2D or 3D. Gestural recognisers can do this ‘easily’, since all gestures are clearly segmented and can be represented as a continuous plot of points, whereas the freedom that InteractML affords increases the complexity of the data visualisations. It is very different how the system should display a movement computed as the velocities of rotations compared to a movement computed as the distance between the hands and the head. This is also true for the labels, since the goal of the creator might not be to represent a clearly defined shape with their movement, but rather a nuanced and highly dimensional action. Hence, one of the clear weaknesses of InteractML is how the training data is currently displayed. Still, creators can annotate their training datasets with ‘Note’ nodes to keep track of their intentions when recording movements and constructing datasets during the model steering stage.

Moreover, InteractML’s training data modification functionality is not as user-friendly as some prior IML tools. For instance, Wekinator allows users to modify their training data directly on the tool [74], GestureScript and similar gesture IML system display a symbol representation per gestural class to detect on screen [296, 163], and Kleinsmith and Gillies system displayed recorded user movements as 3D motion capture skeletons [147]. InteractML is limited in this sense, since it requires users to modify the JSON files with training data, which is prone to errors and not really user-friendly, or to record further movement samples on a different ‘Teach the Machine’ node. Furthermore, the Wekinator offers a ‘Feature’ matrix where features can be muted, reducing the need of recording again data in scenarios where the user follows a reductive feature selection strategy. In InteractML, users that want to ‘mute’ features from an existing data can’t do so, and are forced to create a new ‘Teach the Machine’ node and record again their movements. Finally, InteractML can’t be used out of the box with any other software except Unity, although it could be used in a similar form to Wekinator, implementing a layer of OSC code to handle information into and out of Unity. Still, Wekinator already has the integration done on the ML side, whereas that is an additional layer that should be written for InteractML.

3.11.4 Future Work

Despite the limitations of InteractML, there are several avenues for future work that could address these limitations and improve the tool’s functionality.

Regarding the limitations of InteractML in regards to VR Interaction Toolkits, future work could explore ways to integrate InteractML with these toolkits to create more nuanced and complex object-focused interactions. This could involve developing new nodes or modules that allow for seamless integration between the two systems, or creating new visual scripting tools that allow for more complex rule-based models to be created within InteractML.

To address the limitations of InteractML in regards to gestural recognisers, new data visualisation tools could be developed to allow for more nuanced and rich visual data representations to be displayed within InteractML, by taking into account the complexity of each feature. Such data visualisations could be displayed both in 2D or 3D in VR, to facilitate data exploration in either medium.

Furthermore, InteractML suffer from some limitations compared with prior IML tools. Future work could focus on developing new training data modification tools that are more user-friendly and accessible to non-technical creators. This could involve developing new visual scripting tools that allow for more seamless modification of training data within InteractML, or creating new features that allow for more fine-grained control over the training data. For instance, we could implement visual functionalities to support incremental or reductionist feature selection strategies that save the user from recording data again after re-selecting features. The feature matrix from the Wekinator [74] could be a good starting point, although the interface doesn't have a lot of room for embodied exploration. Alternatively, any form of movement data could be displayed based on the gameobject that originates from, and movement can be displayed as a recording of the movement itself regardless of the feature. Then, once the user decides to increment or reduce their feature selection, a break down of further visualisations of the same gameobject could be displayed in a new pop-up window. For instance, for the rotation feature, the object would be displayed only rotating on intself, and the velocity feature would display the velocity vector based on the movement recorded. Such an idea is still limited, since it is impossible to predict all the gameobjects that the user might want to extract features from, nor every possible feature selection once features are chained (i.e. the velocity of the distance between rotations of the head in relation to the controller). Therefore, such a challenging task on generalisable training data visualisation remains as a challenging open field of research.

Overall, there are many exciting avenues for future work in the field of interactive machine learning for motion-controlled videogames, and InteractML is well positioned to be at the forefront of these developments. By addressing the limitations of the tool and continuing to innovate and develop new features, InteractML has the potential to become an indispensable tool for game creators looking to create complex and nuanced motion-controlled interactions.

3.12 Conclusion

InteractML offers a simple and accessible tool to develop sensor interactions, using an in-engine visual node IML workflow that does not require prior expertise with ML techniques. The tool also includes a VR module to perform in-VR model steering loops, making use of the movement tracking sensors of the VR system. Thus, InteractML supports game creators in more easily developing movement interactions with movement sensors, and in creating more nuanced embodied experiences for players. This chapter contributes to the body of literature of interactive machine learning by pushing the state of the art with a games-specific IML system built into a game-engine, and visually programmed to facilitate non-expert interaction. Furthermore, it contributes to motion control and embodied interaction design methodologies by incorporating IML implementation and design principles into motion controls design methodologies.

InteractML enables the design methodology we study in this thesis, therefore the following chapters will focus on the in-depth investigation of each of the embodeid design stages: ideation, implementation and evaluation. Chapter 4 will begin by studying the embodied ideation process, and chapter 5 will focus on the embodied implementation and evaluation of the process using InteractML.

4 In-Medium Motion Control Embodied Ideation within an IML context

4.1 Introduction

This thesis investigates a novel design process and IML tool for VR embodied game control design. Prior research showed that embodied interaction design processes benefit from the implementation of the ideas generated. In order to fully understand such design process, we need to investigate the three stages of embodied motion control design: namely embodied ideation, embodied implementation and embodied evalation. Therefore, while Chapter 3 deal with the description of our tool for VR motion control creation, this chapter investigates the role of embodiment in the first step ideation for VR motion controls.

Traditional ideation methods used in interaction design commonly fail to explore and capture the somaesthetic dimension of movement interactions. As a result, designed interactions may be usable and responsive but still lack noteworthy applications [174]. This has led researchers to develop a range of novel embodied ideation and design methodologies that explicitly target articulating and affording particular kinds of bodily experience, such as bodystorming [246], somaesthetic design [117], or embodied sketching [174].

These embodied methods assume bodily co-presence: as participants bodily explore and act out interactions, multiple participants might play the roles of different agents or parts (and thus need to physically interact with each other), and participants fluidly build on each other's ideas by observing and mirroring each other's movements. However, creative collaboration is increasingly shifting toward remote arrangements [62] without immediate bodily co-presence. Accelerated by the COVID-19 pandemic [37], businesses have been fast to explore and adopt combinations of video conferencing with new mass-market remote design collaboration tools like *Figma* or *Miro* to conduct common non-embodied ideation methods and workshop formats. However, these usually focus on and afford shared ground in the form of a focal object or display space, such as a virtual whiteboard with and sticky notes [170]. They do not afford bodily co-presence.

Previous research explored how other physically co-located ideation methods can be adapted to a digital medium [132]. Boletsis, Karahasanovic and Fjuk (2017) [23] explored how to translate a well-known embodied design methodology, bodystorming, into virtual reality through 'Virtual Bodystorming' for service design. Weijdom (2022) [315] explored how bodystorming techniques blending physical and virtual props in a 'Mixed Reality' scenario could affect designing performative experiences.

Additionally, previous literature has suggested that, while embodied interaction design offers many documented benefits grounded on embodied experience and cognition [121, 78, 281], there is a lack of practical embodied reflection on the embodied design loop, because designers should have focused loops of design and implementation to better reflect on their ideas [82]. Consequently, IML has been identified as a promising technical solution to such embodied design reflection loop [93]. Our IML tool, InteractML, contributes to the literature its visual and embodied IML workflow. However, studying embodied implementation without investigating embodied ideation would be incomplete, since ideation is an essential first step in the design process, and there is limited work in how in-medium embodied ideation is performed and what is the role of embodiment in such scenario.

We therefore investigate in this chapter the opportunities and challenges when ideating VR game motion controls in-medium. We will follow a comparative study in which game creators bodystorm in-medium against out of medium in an idealised IML scenario without constraints (i.e. any input can be mapped onto any output).

However, this study was performed during the first wave of the worldwide COVID-19 pandemic, therefore our control group was performed in a video-call scenario instead of an in-person scenario. Nevertheless, this allow us to delve into relevant insights about technology and embodiment, and the impact it has on motion control design. It also contributes to the understanding of computer-mediated embodied remote work practices, something that it is likely to become even more important given the impending need to reduce carbon emissions from worker mobility for sustainable international creative collaborations [225].

Therefore, in this chapter we explore two different mediums suitable for remote embodied ideation: video-calls and virtual reality online spaces. In order to compare the two mediums, we prepared a series of structured embodied design workshop inspired by Márquez Segura and colleagues' (2016) [174] Embodied Sketching work. Each workshop consisted of a total of seven different activities to sensitise, bodystorm and ideate with designers in one of the two mediums.

4.2 Method

We chose a qualitative study design to investigate how in-medium embodied design affect the ideation of game motion controls as the first step of an embodied design process. Our qualitative approach follows a comparative between-subject design. This qualitative approach is necessary because of the constructive and non-quantifiable nature of our analysis framing, where we are interested in investigating qualities of an embodied ideation process. Similar qualitative analyses have previously been performed in studies of embodied interaction ideation [246, 174] and design [116]. To achieve an understanding of how in-medium embodied ideation works, we opted for a workshop structure where participants were initially bodily sensitised [174], introduced into movement ideation principles, and then proceeded to group collaborative ideation. As a group, participants ideated either on a private video-call or in a private SocialVR room. Specific topics for ideation were counterbalanced to avoid any potential carry-over effects between sessions. We targeted a sample size of 12 to 20 to ensure good code and meaning saturation [112]. We chose to video record participants' interactions and collect focus group data to better qualitatively capture the embodied interactional phenomena of participants throughout their creativity process. For analysis, we followed a video-based interaction analysis of the interactional phenomena recorded and, on the basis of the coded interaction events, we performed an inductive thematic analysis. There were three workshops per medium, where participants selected how many workshops did they take part in. On each medium, participants were tasked to ideate in three different workshops, with the possibility of selecting how many workshops they took part in. Workshops lasted no longer than 90 minutes each.

4.2.1 Participants

A total of seventeen participants (13 male-identifying, 2 female-identifying, 2 gender variant/non-conforming) took part in the study to ideate in a total of 6 remote workshops. The participants were split in two groups: one for video-call ideation (out-of-medium) and another for virtual reality ideation (in-medium). Six participants (3 male, 1 female, 2 gender variant/non-conforming) ideated in the videocall group and eleven participants (10 male, 1 female) ideated in VR. On each medium, participants were tasked to ideate in three different workshops, with the possibility of selecting how many workshops they took part in. In the video-call group, two participants took part in all three workshops, two participants took part in two workshops and two participants only took part in one workshop. In the VR group, one participant took part in all three workshops, two participants took part in two workshops and eight participants took part in one workshop. The average amount of participants on the video-call workshops was four, and the average amount of participants in the VR workshops was five. It is important to note that one of the participants in the video-calls group required the assistance of two British sign language (BSL) interpreters during the ideation workshops, raising the average number of people in the video-calls from four to six and it was likely to impact communication between participants.

The participants were recruited through social media or email, with the requirement that they had previous experience in videogame creation and virtual reality. The experience requirement was a deliberate choice to assure that participants were familiar with the ideation topics chosen for our case study. After participants filled the consent form, they were invited to a private texting server on the Discord platform to communicate outside of the workshops.

4.2.2 Procedure and Data Collection

Before working on the embodied collaborative ideation tasks, participants were briefed about the purpose of the study and introduced themselves to each others. The core part of the study involved collaboratively ideating movement interactions

Table 5: Magical Interactions Workshop Phase Lengths

MAGICAL INTERACTIONS	video-call	VR
(i) Body Sensitising	3 mins	3 mins
(ii) Performance of movement interactions ideated at home	14 mins	28 mins
(iii) Breaking down interactions between actuators/inputs and verbs/outputs	2 mins	8 mins
(iv) Force-pairing actuators/inputs and verb/outputs to generate new movement interactions	6 mins	23 mins
(v) Performance of movement interactions ideated on-site	8 mins	9 mins
(vi) Labelling of interactions according to implementation difficulty with current game engines	14 mins	8 mins

focused on three topics in two media; one in the videocall platform Zoom and the other in the socialVR platform Rec Room. Each topic focused on eliciting different kinds of interactions for VR videogames where movement can be used: everyday interactions, magical interactions and standard game interactions. The rationale behind these three topics was to offer participants a diverse range of movements to work with, as movements that are performed on a daily basis tend to be qualitatively different from movements that are performed on films, books or games, many times exaggerating the performance for visual impact. For each topic, participants were tasked to think of at least 3 different movement interactions at home before the start of each workshop.

Regardless of the topic, each workshop involved 6 phases: (i) body sensitising with designers, (ii) performance of movement interactions ideated at home, (iii) breaking down interactions between actuators/inputs and verbs/outputs (i.e. move your hands in a circle (actuator/input) to create a portal (verb/output)), (iv) force-pairing actuators/inputs and verb/outputs to generate new movement interactions, (v) performance of movement interactions ideated on-site, and (vi) labelling of interactions according to implementation difficulty with current game engines. Whenever the group was big enough (4 participants or more), participants were split up in two sub-groups from phase iv onward. The exception was the video-call magical interactions workshop, since the sub-group policy was introduced afterwards.

In order to have similar conditions among the groups, the 6 phases were timed and participants were asked to continue with the next embodied ideation phase when the time was up. To ease participants' creative processes, we did not require groups to continue when they wanted to move on to the next phase, and they were allowed to finish discussions and embodied ideation that were initiated before the phase finished. Still, after each workshop we adapted the timings to ensure participants could finish the creative tasks in the next workshop. Timings for each workshop and medium can be found on tables 5, 6 and 7. In total, including the introductions, the creative work and a follow-up debriefing interview, each workshop lasted 90 to 120 minutes.

Table 6: Everyday Interactions Workshop Phase Lengths

EVERYDAY INTERACTIONS	video-call	VR
(i) Body Sensitising	4 mins	4 mins
(ii) Performance of movement interactions ideated at home	15 mins	18 mins
(iii) Breaking down interactions between actuators/inputs and verbs/outputs	13 mins	13 mins
(iv) Force-pairing actuators/inputs and verb/outputs to generate new movement interactions	19 mins	24 mins
(v) Performance of movement interactions ideated on-site	8 mins	7 mins
(vi) Labelling of interactions according to implementation difficulty with current game engines	11 mins	19 mins

Table 7: Game Interactions Workshop Phase Lengths

GAME INTERACTIONS	video-call	VR
(i) Body Sensitising	4 mins	5 mins
(ii) Performance of movement interactions ideated at home	11 mins	21 mins
(iii) Breaking down interactions between actuators/inputs and verbs/outputs	28 mins	11 mins
(iv) Force-pairing actuators/inputs and verb/outputs to generate new movement interactions	13 mins	18 mins
(v) Performance of movement interactions ideated on-site	10 mins	5 mins
(vi) Labelling of interactions according to implementation difficulty with current game engines	10 mins	15 mins

Consent to video record the workshops was gathered from participants. In the video-call setup, we recorded the researcher's screen showing all participants webcams in a "gallery view", including the ones with the webcams off. In the VR setup, we placed two virtual cameras to capture participant's interactions. One camera was controlled by the main author and the other by an assistant, attempting to capture the entire group at all times (e.g. how participants interacted with each other, their movements from two different angles where possible and their writings on the virtual whiteboards). Whenever participants were divided in two sub-groups, each virtual camera was assigned to each of the two sub-groups. At the end of the study, participants joined back together to take part in a group debriefing interview about their experience in the workshop and rationale behind interaction labelling choices.

4.2.3 Analysis

The approach used in our research draws from qualitative ethnomethodology and focuses on the interactional and sequential organisation of verbal and nonverbal behaviour. Participant's interaction and collaboration while working on the embodied ideation tasks was analysed using video-based interaction analysis as outlined by [136]. To that end, interactional phenomena such as turn at talk, communication practice, clarification, repairs, and movement-based exemplifications of ideas were coded. Furthermore, on the basis of the coded interaction events, we performed an inductive thematic analysis [28].

With reference to previous experience that our group had with pilot face-to-face [219] and remote workshops, we anticipated that we would observe different communication behaviours between the video-call and the VR groups. Then, we explored the influence on collaboration and ideation. We thus applied a deductive approach to our interaction analysis. We followed an up-to-bottom approach and started by looking at entire pairs of videos based on each workshop topic: the pair of videos about magical interactions, the pair of videos about everyday interactions, and the pair of videos about game interactions. We then broke each video into smaller sections and coded them. Once we had an initial pool of candidate codes, we then had group-viewing sessions on video snippets to further iterate on categories. In the coding process, attention was focused on when participants interacted with each other as well as how they communicated their ideas with their bodies on both the video-call and VR mediums. Additionally, we looked at how the impact of participant's environment (either physical or virtual) and the presence of the researcher in the space influenced participants' interactions. Identified codes were then checked against other sequences across pairs of videos. To code the verbal communication between participants, movement ideation, and the location of participants, we used the qualitative video analysis software ELAN [65]. We coded the videos using 3 schemes, each focusing on a different aspect. First, we coded the participant's interactions in the workshop, asking; where is the workshop going poorly? How did the presence and interactions of the researcher influenced the workshop? Second, we coded participant's movements, asking; what is the nature of movement produced on each medium? What is the relationships between the movements and the interactions? Third, we coded the use of participants' movements in communication, asking; how is movement used in collaboration? How is movement used in individual thinking? Finally, by combining these insights, we also asked; does the remote medium impact the embodied ideation techniques used by the participants? With these questions in hand, we conducted our interaction analysis. An inductive, bottom-up thematic analysis [28] was later used to make sense of the codes generated.

4.3 Results

4.3.1 Types of User Interactions

In total, 9 hours of video and 110 ideas were analysed. A total of 28 ideas were generated in the Magical Interactions workshop, 47 ideas in the Everyday Interactions workshop and 35 ideas in the Game Interactions workshop (Figure 57). Based on participants' verbal and non-verbal behaviour with the researcher and between themselves, video snippets were selected and coded by the main author, and preliminary codes were discussed with the other authors by collaboratively viewing selected video fragments and discussing the interactional phenomena observed [136]. In the end, the codes were divided into two broad categories of events: Interactional (Embodied Demonstration, Embodied Exploration, Play) and Conversational (Request Clarification, Embodied Clarification, Laughter, Joke). Furthermore, since some of the video snippets provided richer insights that couldn't be strictly categorised, we provided a "Special Insight" category for annotation (i.e. a participant is bodily demonstrating, but while doing so they interact in an unusual but interesting manner with another participant/virtual prop). In addition, less common events such as pointing and joined attention where coded under the "Special Insight" category with a contextual annotation. We used all three categories to code the videos for further analysis. As this study focuses on the participants movements, interaction, communication, and creativity in their respective remote medium, video sections with no observable phenomena were excluded from the analysis. Finally, the 110 ideas collected from the videos were coded into 37 low-level codes and synthesised into 11 high-level codes. Some of the ideas 110 ideas were coded with more than one of the 37 low-level codes, or with no suitable code for one for the themes, resulting in the distribution of codes per theme presented in figures 58, 68, 73, and 82. An enumeration of the most relevant interaction/communication codes can be found in Table 8 and an enumeration of the low-level and hight-level codes for the generated ideas can be found in Table 9.

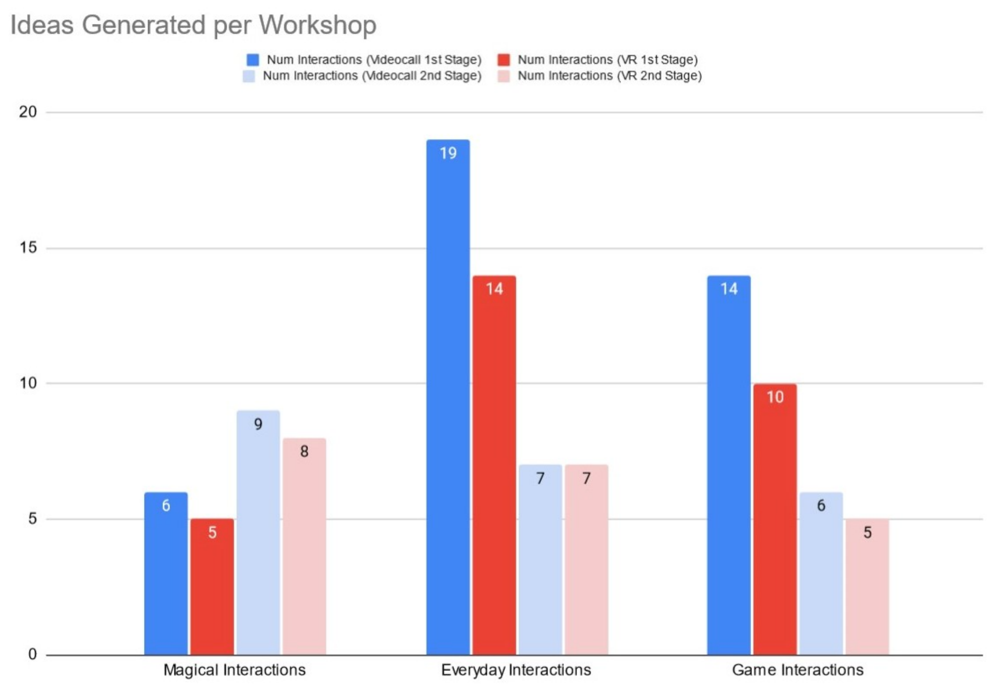


Figure 57: Total amount of interactions genereated per workshop.

Table 8: Interaction Events Codes

Interaction Events	Communication Events	Turn-taking events
Embodied Demonstration	Clarification Request	Successful Turn-taking
Embodied Exploration	Embodied Clarification	Challenging Turn-taking
Play	Laughter/Joke	

Table 9: Low-level codes for ideas generated in each workshop.

Magical Interactions	Everyday Interactions	Game Interactions
Upper body	Upper body	Sequence
Object creation	Object quality	Attack Action
Player movement	Object creation	Upper body
Object movement	Agent Movement	Object destruction
Sequence	Object Movement	Lower body
Attack Action	Hands	Player quality
Defense Action	Agent Quality	Hands
Equip Item Action	Mouth	Communication Action
Not present in whiteboard	Unconventional Input	Object quality
Voice	Head	Object movement
Object quality	Object Selection	Eyes
Audio Creation	Eye	Object selection
Hands	Lower body	Player movement
Lower body	Player Quality	Full body
Not demonstrated to group	Player Movement	World quality
Breath	Not present in whiteboard	Head
Unconventional Input	Object destruction	Agent quality
Pose	Full body	Agent destruction
	Sequence	Equip Item Action
	Multiplayer action	Audio Creation
	Cooperation	

4.3.2 Thematic Analysis

On the basis of the codes and the insights gathered during the first up-to-bottom interaction analysis, we performed an inductive thematic analysis [28] to make sense of the data. We looked at themes following a bottom-up approach by looking at the collection of coded video snippets and the high-level coded generated ideas. We looked for patterns across the snippets, with a focus on (a) movement, (b) creativity, and (c) communication. Since this follows a comparative and qualitative nature, we compared snippets where similar interactional phenomena was being observed both in the video-calls and in SocialVR. Additionally, instances that could not be compared because of the particular affordances of each medium were also taken into account during the analysis.

Candidate themes were synthesised from the coded video snippets and coded generated ideas by the main author and discussed with the other authors using slide presentations and example video fragments for further iteration. After enough iterations the following themes were synthesised, where all codes and relevant interaction instances could fall in place. Themes were divided into 4 main overarching categories depending on the relationship between input and ouputs of their interactions, how the interactions were ideated during the workshop, and how participants communicated. Themes are: (1) Output Space, (2) Input Space, (3) Expressive Ideation Space and (4) Communication Space. Each theme has two to three subthemes. Subthemes for theme (1) are (1.a) Effect on Virtual Elements and (1.b) Social Interactivity. Subthemes for theme (2) are (2.a) Bodily Expressions and (2.b) Expressive Input Space. Subthemes for theme (3) are (3.a) Exploration of affordances, (3.b) Co-located Playfulness, and (3.c) Challenges in Remote Ideation. Subthemes for theme (4) are (4.a) Support for Non-Verbal Cues, (4.b) Embodied Repair Strategies, and (4.c) Remote Turn-Taking.

Table 10: Themes and subthemes from the thematic analysis

THEMES	**SUBTHEMES**
(1) Output Space	(1.a) Effect on Virtual Elements (1.b) Social Interactivity
(2) Input Space	(2.a) Bodily Expressions (2.b) Expressive Input Space
(3) Expressive Ideation Space	(3.a) Exploration of Affordances (3.b) Co-located Playfulness (3.c) Challenges in Remote Ideation
(4) Communication Space	(4.a) Support for Non-Verbal Cues (4.b) Embodied Repair Strategies (4.c) Remote Turn-Taking

Themes and codes will be discussed in the following subsections in depth, providing example video-frames to illustrate our findings.

4.3.2.1 Output Space: Effect on Virtual Elements

Having an effect on virtual elements was the most common form of output in participants' interactions. We understand virtual elements as the fundamental objects in a virtual environment, like characters, props and scenery. From all videocall sessions, 90% of outputs referred to virtual elements. And from all SocialVR sessions, 66% of outputs referred to virtual elements. By looking more in depth into the kinds of virtual elements participants wanted to interact with, we found four main categories of virtual elements: (1) objects, (2) the player, (3) other agents, and (4) the virtual world itself.

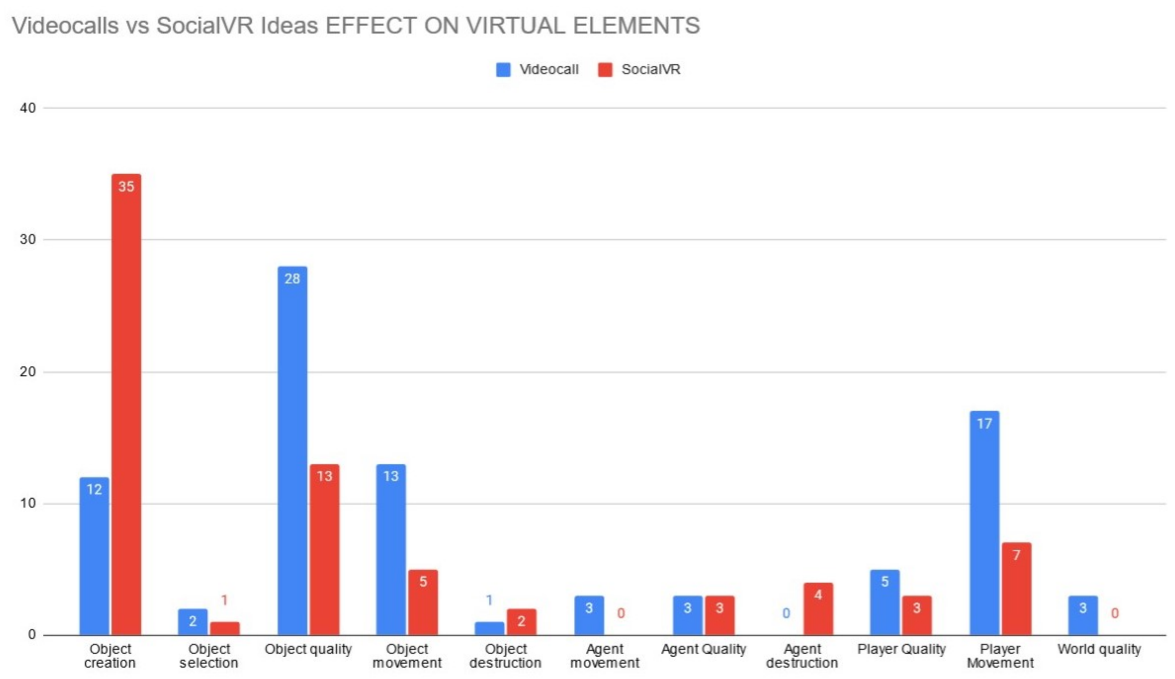


Figure 58: Comparison of low-level codes for generated ideas under the theme Effect on Virtual Elements (a generated idea could have more than one code assigned)

A break-down comparing generated ideas between each medium can be found in figure 58 and explanations per category can be found below.

4.3.2.2 Object Interactions Object interactions are one of the first high level codes synthesised from the generated ideas, and the most predominant form of outputs (24% of all codes refer to object interactions, with a total of 224 interactions). In this context, an object can be understood as any visible virtual element that can't be embodied but rather controlled or affected by players actions (i.e. an energy ball, a protective bubble or a tomato). Participants ideated interactions to create, destroy, select, move and alter qualities of objects. For instance, in SocialVR, a group of participants was bodily exploring a stretching movement with their arms when they had the idea of affecting objects with that movement (Figure 59). After a brief conversation and performing the movement among them, the participants decided that it made sense to stretch the arms to multiply objects. Participants even bodily explored how slow movement repetitions could reduce the multiplication effect. Another instance where participants ideated an object interaction was during one of the videocall workshops when participants explored what the "Pulling a rope" movement could do (Figure 60). After performing the movement a few times each and discussing about it, they decided that it made sense that by pulling they could change the size of an object, similarly to how you could zoom/pan with the fingers on a touch-screen.

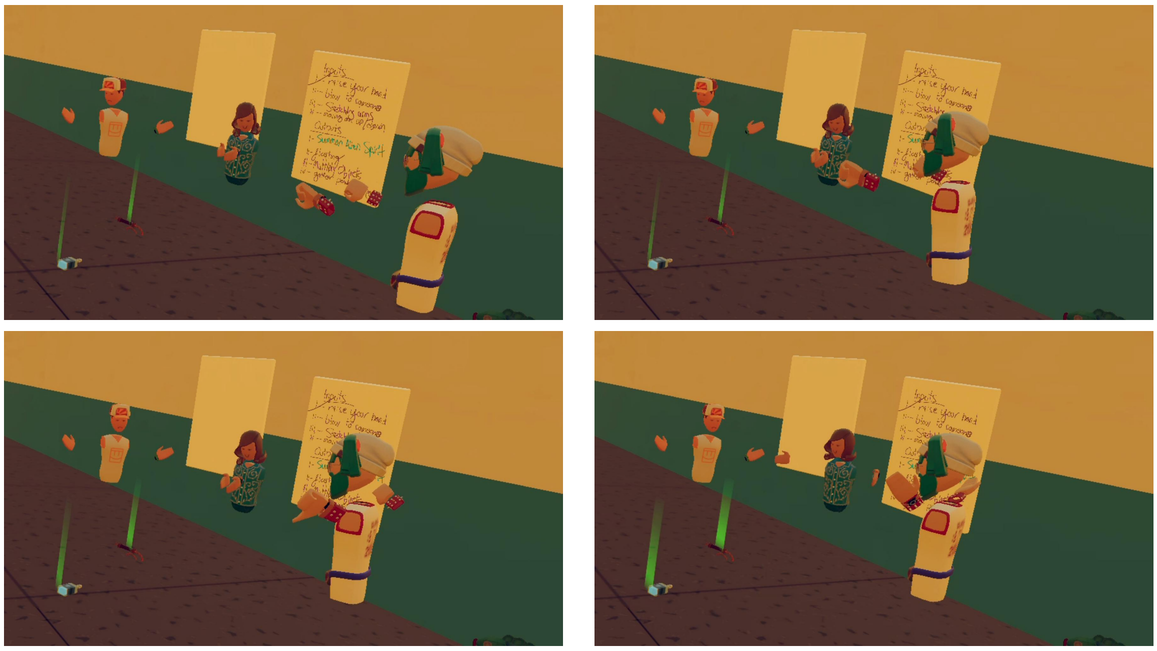


Figure 59: Stretch arms to multiply objects Interaction. Participants ideated movement in which they needed to stretch their arms two multiply objects . As participants were performing the movement, they started to experiment with the idea of slowly performing the repetitions to affect the object multiplication by reducing the amount of objects created. In addition, participants also accompanied their movements with sounds they were making with their own voices, imitating the sound of pop-corn cooking in a microwave as they imagined objects being multiplied in front of them.

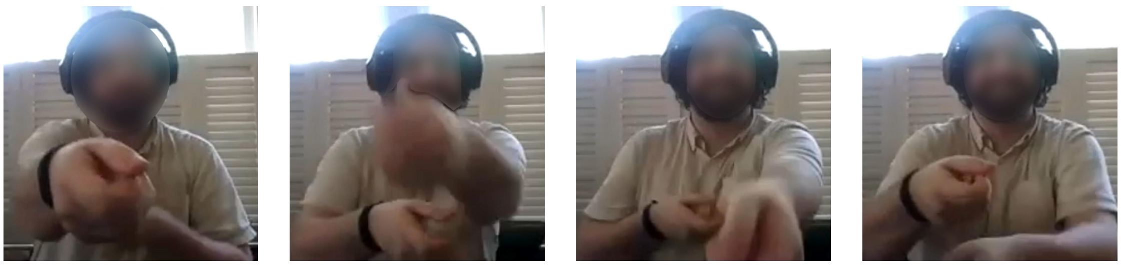


Figure 60: Pulling rope to change object size. In this example a participant is bodily exemplifying a "pulling rope" movement that makes an object smaller. However, after performing the movement in front of the camera, he explained that he didn't like the feeling of pushing rope forward to make the object bigger, hence he only accepted the pulling mechanism as a valid movement.

Between the two mediums, movements that had an effect on the creation of objects were more predominant in SocialVR whereas movements that had an effect on qualities of an object were more predominant in the videocalls (Object creation interactions: 12 videocall vs 35 SocialVR; Object quality interactions: 28 videocall vs 13 SocialVR; see Figure 58) This could potentially be related to how each medium affords embodied ideation. In a videocall, participants are surrounded by objects in their real spaces, and by having a quick look around, participants might be influenced by each of these objects in their space. For instance, one of the interactions that was ideated in the video call was controlling how the curtains opened or how a door opened. Both elements might have been physically present in front of the participant that suggested those movements. By comparison, in SocialVR, participants do not have such diversity of objects. We filled the virtual room with a selection of virtual props that rec room offered, but it did not equal the amount of objects that are found in a real room: office material, house furniture , electricity sockets, boxes, windows ... (Figure 61) Therefore, it might be that the lack of virtual objects might prompt participants to explore ideas and movements that would allow them to fill the space so that it looks closer to its real counterpart.

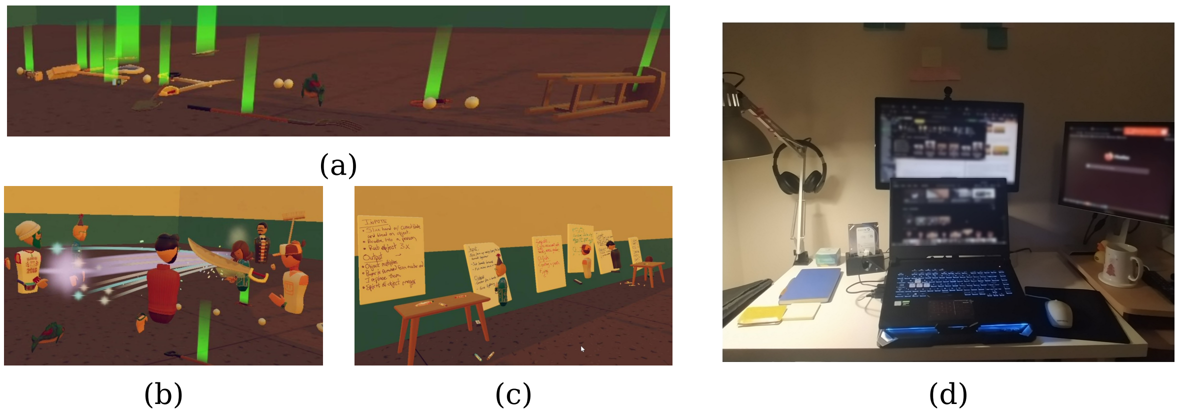


Figure 61: Virtual vs physical room appliances that might have biased the ideation of object interactions. (a) Collection of virtual props present in virtual room, from left to right: magic wand, brush, bow, shields, swords, shovel, garden fork, snowballs, toy fish, wip, wooden stool. (b) Participants playfully interact with the virtual props betwen them. (c) Participants use virtual markers provided on virtual tables to fill a virtual canvas with the ideas they created. (d) An example of a potential home physical environment from which participants might have joined the videocall workshops.

4.3.2.3 Player Interactions Since the case study of this paper is focused on VR game interactions, participants ideated interactions having an effect on the player. The player in this context is understood as the agent that the participant would embody in a videogame. Player interactions mostly generated ideas to alter qualities of the player or to move the player; counting 31 player interactions in total.

For example, some of the ideas were related to player qualities (5 interactions in videocalls and 3 interactions on SocialVR workshops). In one of the videocall workshops, one participant created an idea where “Crossing your arms as if you are resting” would make the player to rest (Figure 62.a). The idea required to have a body posture and facial expression that would be perceived by other as clearly “resting”, since the participant initially bodily demonstrated the idea by sitting on a chair, which seemed like potentially not specific enough after demonstrating it. The idea was not picked-up in the force-pairing stage. Another player quality example was ideated in SocialVR when a participant demonstrated the movement to wash their hands with an avatar to specifically clean COVID-19. (Figure 62.b).

The most common kind of player interactions was be focused on the movement on the player (17 in videocall workshops and 7 in SocialVR workshops) . In SocialVR, a group of participants forced-paired the movement "Pedal hands" with the "Run" player action. The participants were playfully exploring the movement with different outputs like "Going up the stairs" or "Searching a pile of objects", but finally settled on "Run" as their favourite outcome (Figure 62.c).

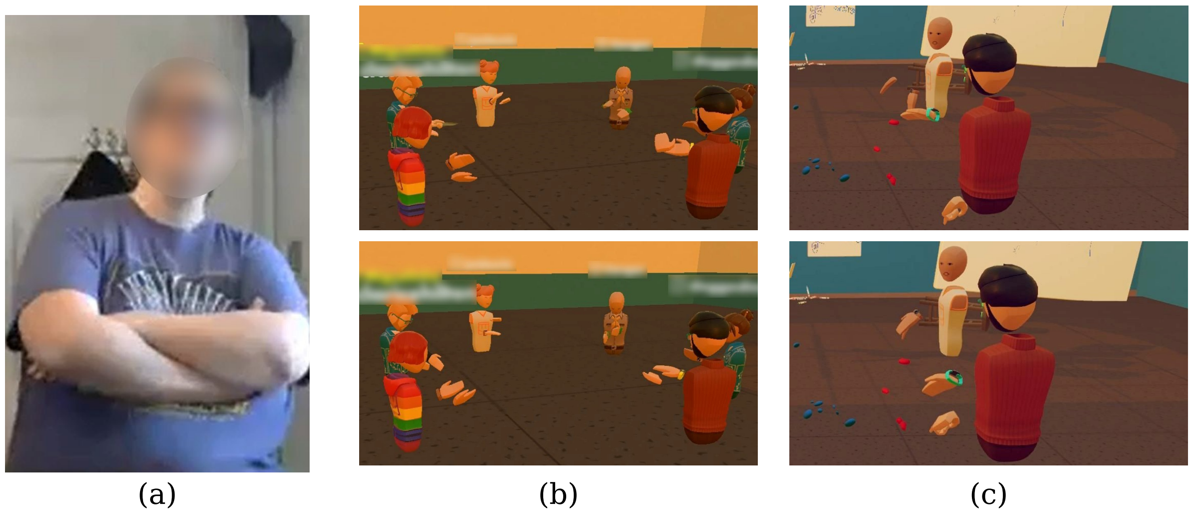


Figure 62: Player Interactions Demonstrations. (a) Crossing arms in resting posture to rest the player. (b) Wash hands to eliminate the COVID-19 virus. (c) Pedal with hands to make the player run.

4.3.2.4 Agent Interactions Similarly to what happened with object interactions and the player interactions, participants generated ideas that had an effect on agents other than the player. We can understand agents in this context as any kind of virtual entitity, represented by an avatar, that is controlled either by remote users (i.e. other players) or artificial intelligences (i.e. non-playable charaters, enemies) or a mixture of both (i.e. a remote user controlling a group of enemies). The generated ideas focused on the movement, qualities and destruction of the agents; with 13 ideas generated in total.

For example, one of the interactions that affected the movement of an agent was ideated in a videocall workshop. A participant paired the "Chewing" mouth movement as an input, with "Leading dog on lead" as an output. The participant thought it was fun to control with his jaw the direction of where his dog would move next to follow his orders. This particular interaction also shows one of the strengths of the videocall format, which is the higher expressivity for facial movement interactions. (Figure 63).

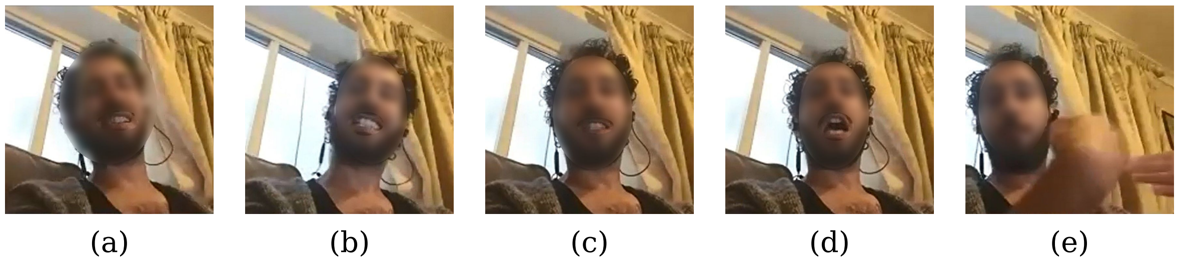


Figure 63: Chewing to lead dog. In this agent interaction example a participant is bodily exemplifying a “Chewing” face gesture that allows to lead a dog. The participant didn’t stand up to do this movement, but rather showed on the camera the different variations of the gesture to control the different axis of movement. (a) Chewing stronger on the left leads the dog to the left. (b) Chewing stronger on the right leads the dog to the right. (c) Chewing equally with both sides of the jaw leads the dog to the right. (d) Chewing while opening the jaw slightly leads the dog backwards. (e) Participants explains to the rest of the group how his previous movements would lead the dog to wherever he wanted.

A second example that illustrates an idea affecting a player quality came from the Game Mechanics SocialVR workshop, where a participant paired the “Touch person on top of the head” gesture as an input with the “steal an item from that person” output (Figure 64). The SocialVR medium allowed participants to explore close-contact interactions movements because of the virtual co-llocation of other player avatars. Similarly, in the same workshop an interaction for destroying agents was produced. Participants ideated an interaction where hugging an agent would destroy it (Figure 65). Furthermore, the same group of participants further ideated a variation of the movement. when bodily demonstrating their interaction to the rest of the workshops. The variation consisted in pairing the hugging gesture as an input with ingesting big pieces of food as the output. The variation resulted from a playful group demonstration where participants were laughing and virtually hugging each others.

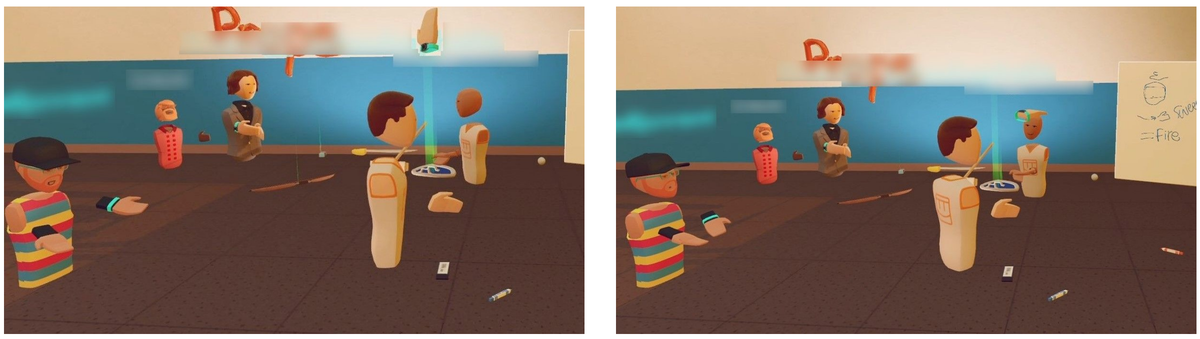


Figure 64: Touch head to steal item from agent. In this agent interaction example, participants bodily demonstrated to others how the “touching head” gesture would work in SocialVR. The rest of the group obeserved at the demonstration and followed afterwards.

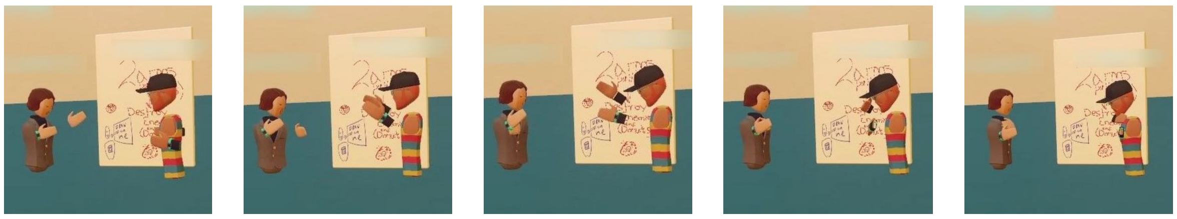


Figure 65: Hugging to destroy enemy. In this agent interaction example, participants are bodily demonstrating how they would hug, or pull their arms towards their chest, to destroy an agent. The participants also collaboratively came out with a variant where hugging would result in eating human-size pieces of food.

4.3.2.5 World Interactions Finally, the last kind of virtual element mentioned among the ideas generated was the virtual world itself. We can understand the virtual world as the environment where all virtual elements live and the systems and rules that control them. This category only came up in the videocall workshop about game mechanics, with just two ideas generated. Participants in the videocall played with the idea of accelerating and slowing down time. Their final idea was inspired by SuperHot VR [280], which is a very popular VR video game that has a direct relationship between how slow time is based and how much the player moves. However, participants created two interesting twists from the idea by force-pairing different inputs. One group of participants paired the "shooting arrow" input to accelerate time (Figure 66), whereas the other group paired their eye movement as an input to slow-down time (Figure 67)

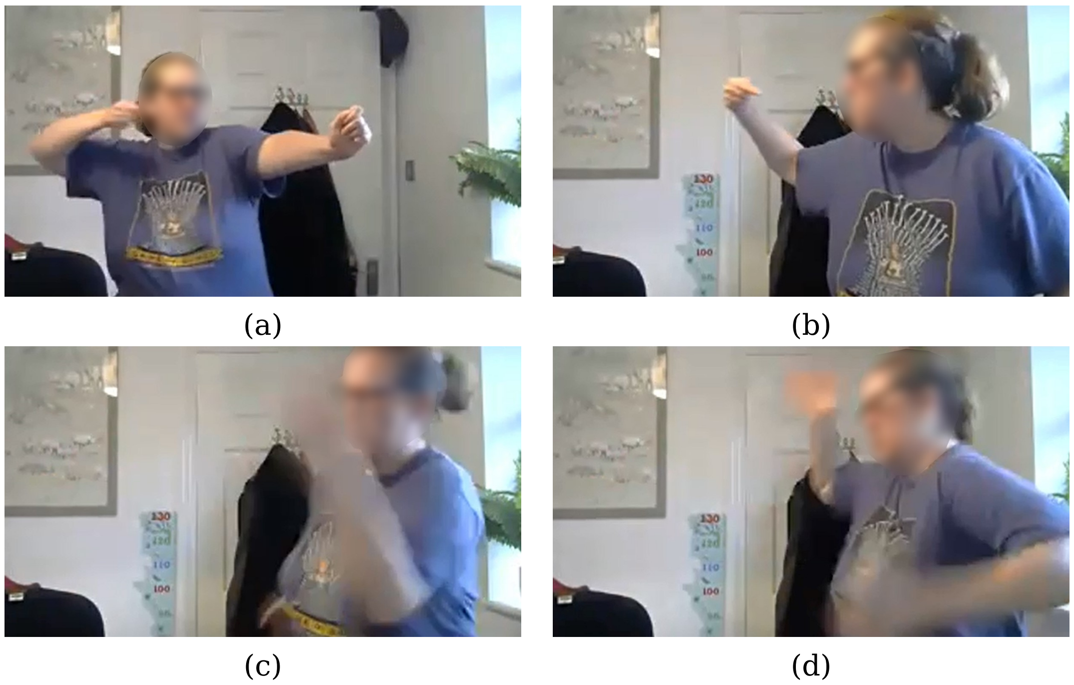


Figure 66: World Interactions Demonstrations - Shoot arrow to accelerate or slowdown down time. (a)-(b) The participant bodily demonstrates the input of shooting an arrow. (c)-(d) The participant bodily demonstrates the output of "accelerating" time.

Figure 67: World Interactions Demonstrations - Use eyes to accelerate or slowdown time. (a)-(b) The participant bodily demonstrates the input of gazing with their eyes in a particular direction. (c)-(d) The participant bodily demonstrates the output of altering time speed to the group.

4.3.2.6 Output Space: Social Interactivity

When looking at all the ideas generated, we identified 44 ideas focused on the interaction between several virtually co-located players. We grouped all those ideas under the theme Social Interactivity, and 79.5% of them were generated in SocialVR (35 out of 44). We provide a breakdown of codes on Figure 68.

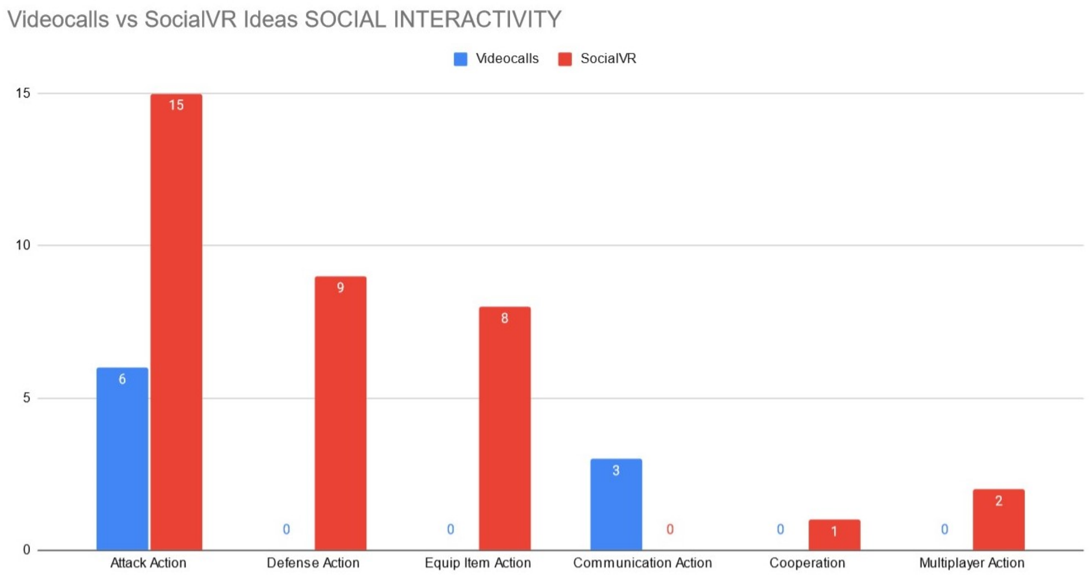
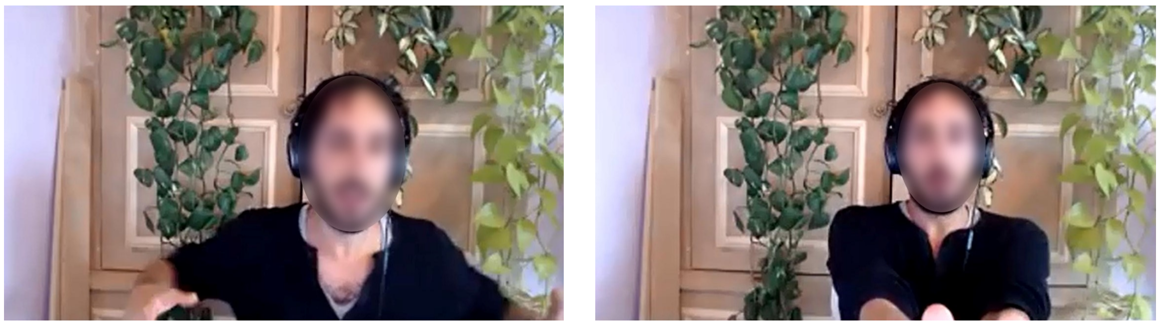


Figure 68: Comparison of low-level codes for generated ideas under the theme Social Interactivity (a generated idea could have no suitable code assigned for this theme)

Because of our particular case-study, 86% of the ideas are focused on common game-like interactions, such as attacking or defending from another player (38 out of 44). As an example of an attack interaction, one of the participants in the video-call group for magical interactions thought of the classical “Hadouken” movement from the Street Fighter game’s franchise [44]. In this interaction, the participant showed in front of the camera how moving his arms forward with the hands shape in a certain way would produce an energy ball thrown at another player (Figure 69). In contrast to this interaction, a participant from the SocialVR workshop for magical interactions ideated a sequence of movements to defend another player with a protective dome (Figure 70).



(a) (b)

Figure 69: Social Interactions Demonstrations - Move arms forward to throw energy ball. The participant in the videocall showed how to start from the movement by opening his arms in (a) and then pushing forward with the hards opened in a semicircle in (b).

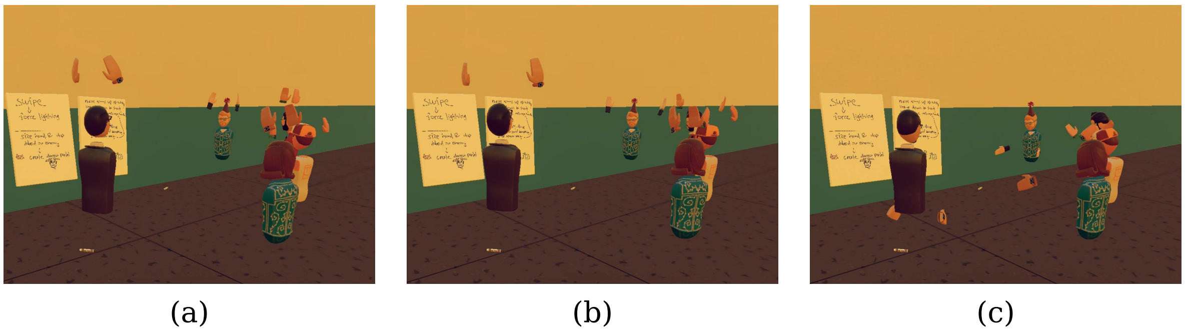


Figure 70: Social Interactions Demonstrations - Create a protective dome for another player. This is an interaction that requires a sequence of movements as an input before the dome can be placed on a player as the output. (a) Participants move their arms up at sides to create a protective dome. (b) Participants set distance between their hands to change the protective dome's size. (c) Participants vigorously move down theirs arms to finally shield another player.

Nonetheless, more general social interactions were also explored, such as communication or cooperation between players. In the videocall workshop for everyday interactions, a group of participants force-paired the "chopping with axe" as an input with the "talk to other player" output (Figure 71), producing an interesting and unconvential result. Furthermore, in the SocialVR workshop for everyday interactions, a group of participants force-paired the output "make friends" with two different playful inputs: wash each others hands (Figure 72 (a)-(b)) and dance with each other (Figure 72 (c)-(d)).

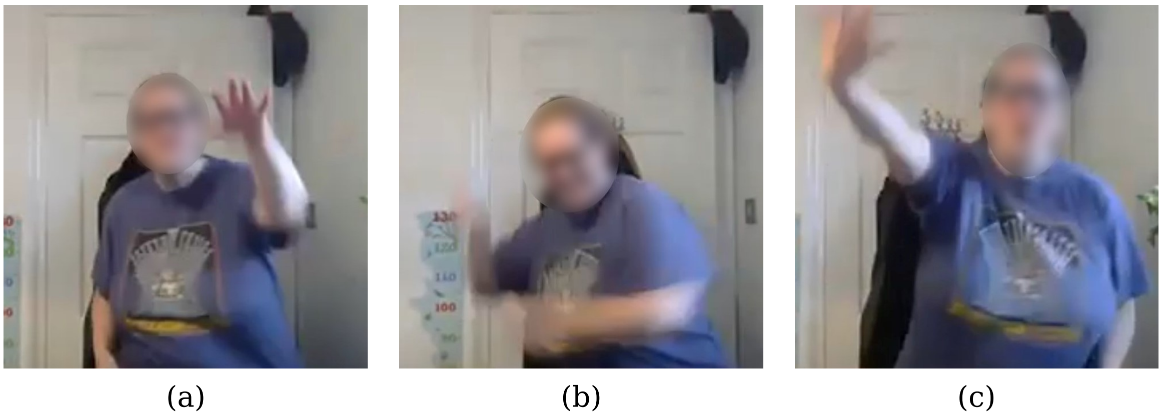


Figure 71: Social Interactions Demonstrations - Hit with an axe to another player to talk. (a) The participant selects which player to talk with . (b) The participant move their arms vigorously pretending to hit another player with an axe. (c) The participant engages in a conversation with the other player.

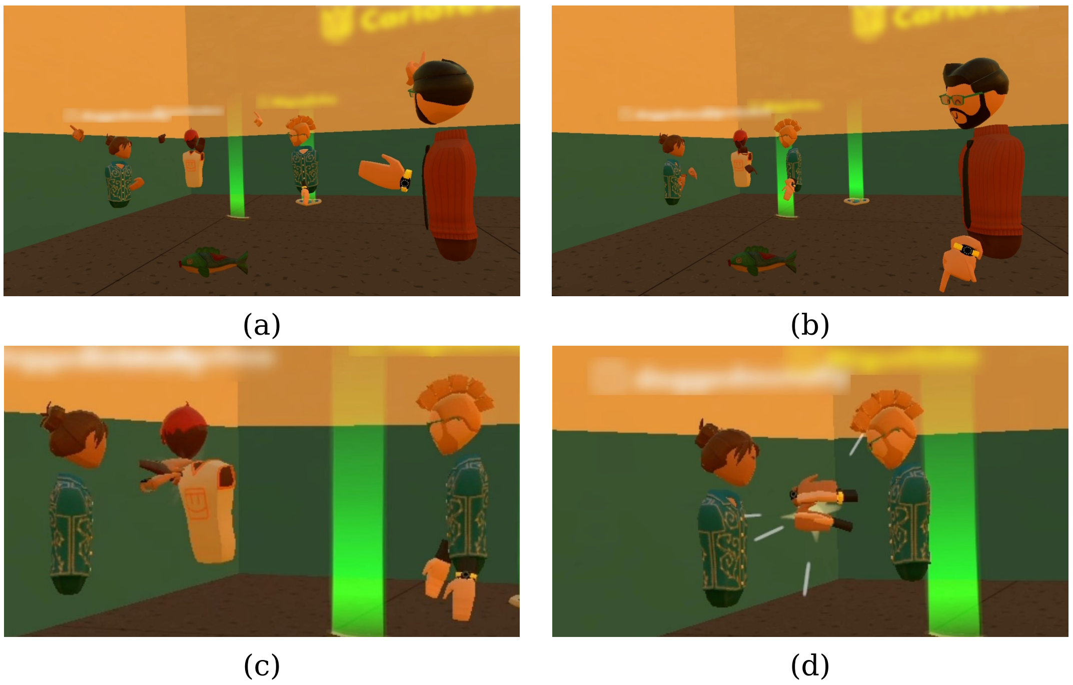


Figure 72: Social Interactions Demonstrations - Cooperative actions. (a)-(b) Participants dance with each other to add each other to their friends list. (c)-(d) Participants wash each others hands to add each others to their friends list.

4.3.2.7 Input Space: Bodily Expressions

When analysing the data from an input standpoint, we could see how diverse and rich the movements and input that participants used in the ideation process were. Participants used the hands, head, upper body, full body, lower body... but not only. Participants also used other forms of input that are more unconventional such as their voice, their mouth, lips or jaw movement; their eyes, their breath, or even a multi-modal combination of several of the previous inputs.

A total of 208 codes were annotated for this theme, out of which 85% (177 of 208) belong to regular limb motion (I.e. head, hands, arms or legs). 5 ideas used eye gaze or eye lid motion. 5 ideas used voice as an input, 2 ideas used mouth movements as inputs and 4 ideas used breath as an input. A complete breakdown of the codes can be found in figure 73.

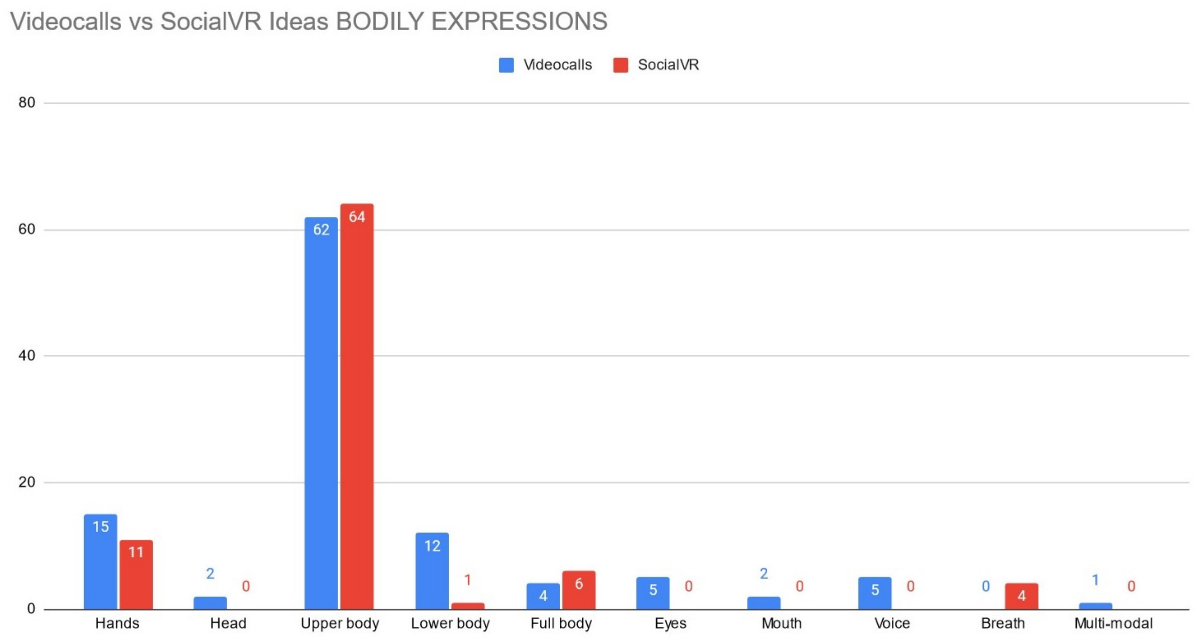


Figure 73: Comparison of low-level codes for generated ideas under the theme Bodily Expressions (a generated idea could have more than one code assigned)

An example of an interaction using hands in the videocall comes from the workshop for everyday interactions, where a participant force-paired an idea that connected "Picking and dropping tea" as an input with "opens and closes curtains" as an output (Figure 74). The participant made sure to show on camera how he envisions the "picking" gesture, and explained the rest of the group how that would work. Figure 72 (c)-(d) shows how a group of participants ideated a movement interaction requiring to "wash each other hands" to "make friends" in SocialVR. They enacted the movement with the limitations of their avatars in SocialVR, but it was picked up by the entire group quickly afterwards.

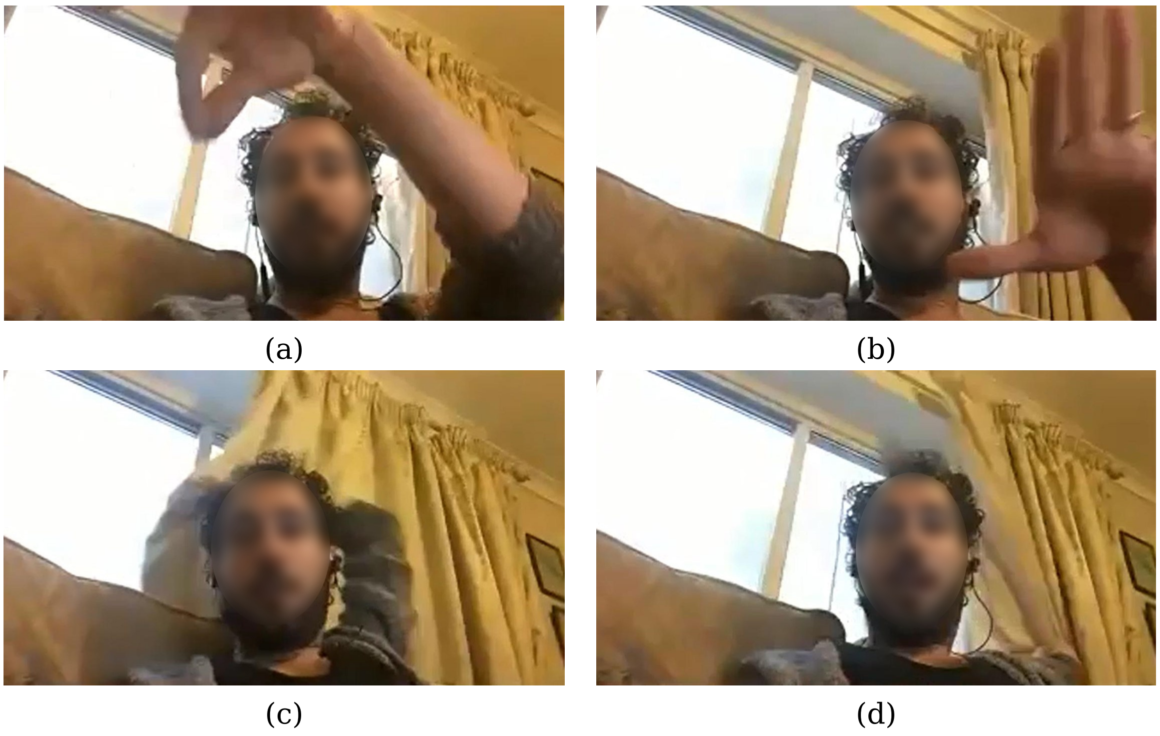


Figure 74: Bodily Expressions - Hand Interaction demonstration in a videocall. (a)-(b) The participant shows to the group how he envisions a fine-grained gesture of picking and dropping a tea bag by pinching with his fingers and releasing on the air. (c)-(d) the participant shows on camera how the gesture would open or close the curtains.

We can only report interactions that used head or eye gaze from the videocall workshop. In the videocall workshop for game interactions a participant thought of an interaction requiring them to look in a particular direction to accelerate or slow-down time (Figure 67). They enacted this interaction on camera by gazing with their head left or right in their room (Figure 67 (a)-(b)). In the same workshop, the participants also ideated movements that required fine-grained eye movements only, requiring to look and focus in order to spread “butter on bread” (Figure 75).

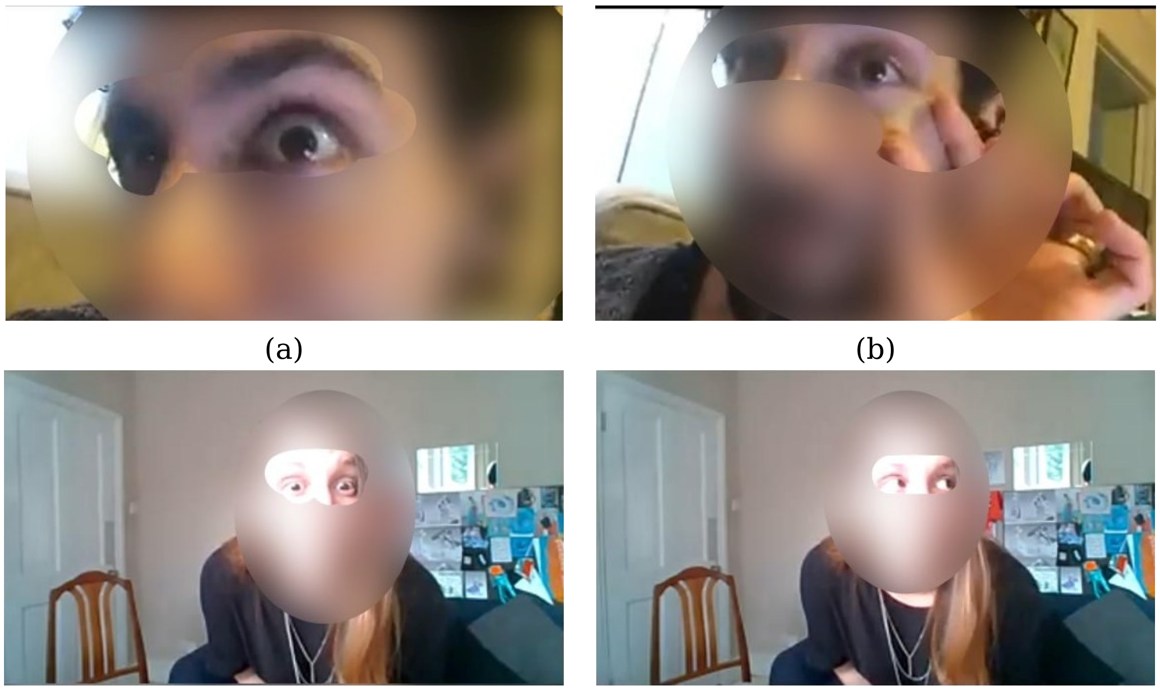

(a) (b) (c) (d)

Figure 75: Showing eye movement on webcam. (a)-(b): The participant bodily demonstrating a movement interaction idea to show their "focused" movement and eye movement. The participant gets closer to his webcam and shows the eye movement required for his interaction. (c)-(d): The participant shows a variation of the eye movement doing an eye roll.

Upper body interactions were the most common type of interactions in both mediums with 126 instances coded out of the 208 in total. For instance, an upper body interaction was ideated in SocialVR when participants were bodily exploring what would be a good fit to the "moving arms up and down" input. Participants ended up force-pairing the "moving arms up and down" input with the "generate energy". (Figure 76). An example of an upper body interaction in the videocall medium was demonstrated by a participant "swinging an axe" as an input gesture to talk with other players (Figure 71).

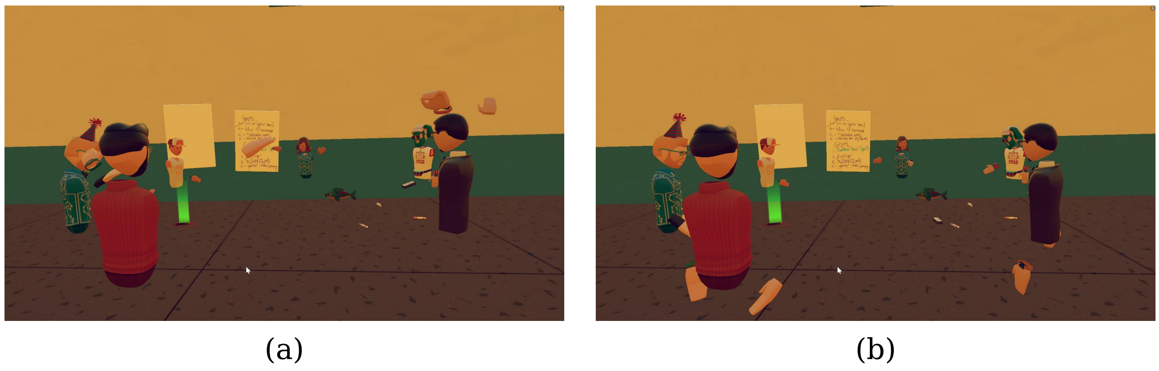


Figure 76: Moving arms up and down to generate energy. The entire group in the workshop mimics the interaction while the participant that ideated it explains how to generate energy from it, since they are "releasing" their own inner energy by doing vigorous upper body movement.

Lower body interactions were mostly produced in the videocalls workshop, with only one lower body interaction produced on SocialVR (12 out of 13 lower body interactions were produced in the videocall workshops). For instance, an example of a videocall lower body interaction was produced in the workshop for magical interactions, where a participant a force-paired the input "move body up and down" with the output "change object size". The only example from SocialVR only refers to "walking around" as an input for the output "walk".

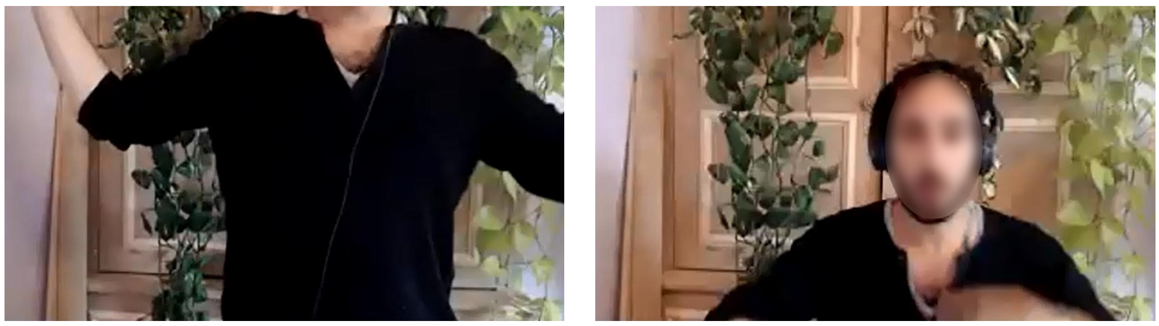


Figure 77: Lower body interactions: Moving body up or down in a videocall to open or close doors.

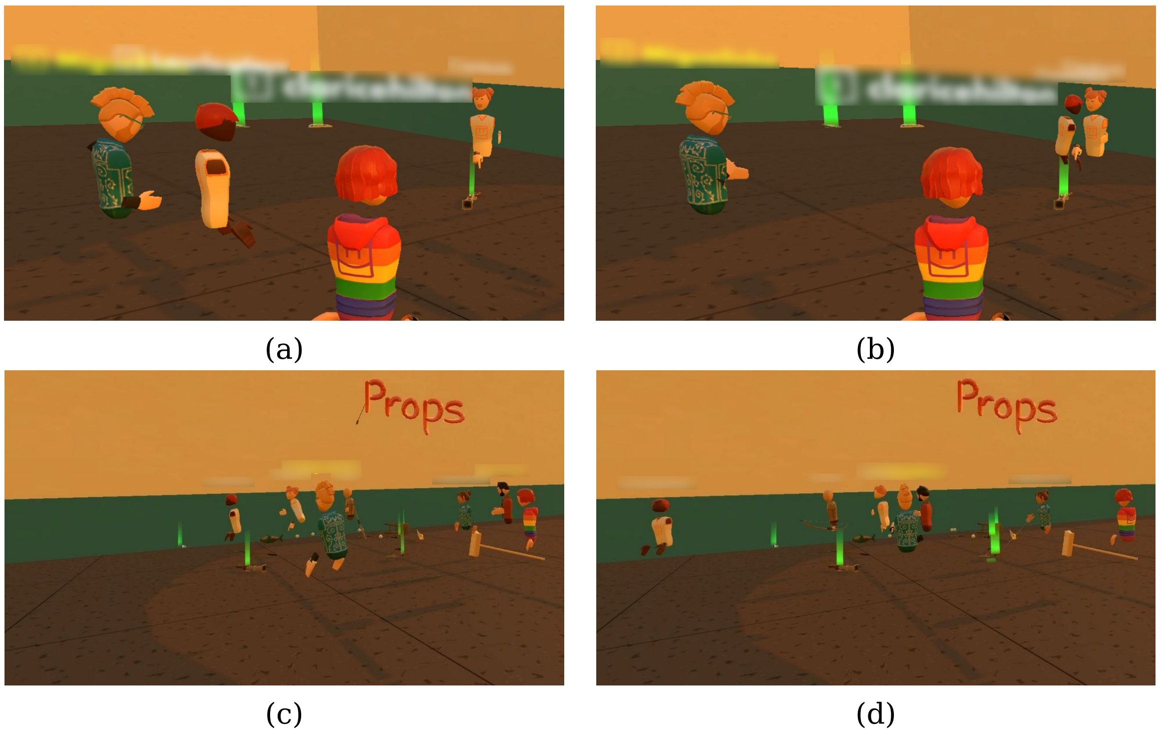


Figure 78: Lower body interactions. (a)-(b) Participant walking around in SocialVR to bodily demonstrate how it looks in the medium. (c)-(d) The rest of the group starts to follow the participant bodily demonstrating the walking interaction

Interactions that used explicitly the mouth or voice were only ideated in the videocall medium. For instance, a participant force-paired the input "chewing" with the leading dog "leading dog", generating an idea where he could control the different directions of the dog's movement by chewing differently (Figure 63). An example of an interaction that used the voice as an input comes from the videocall workshop in magical interactions, where a participant generated an idea requiring to scream a positive afirmation to make an object bigger (Figure 79)

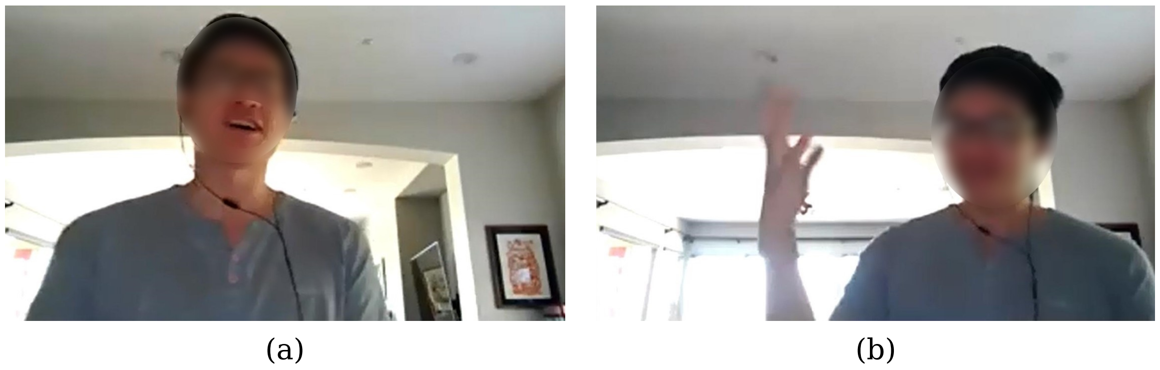


Figure 79: Voice interaction. (a) A participant explains how to scream a positive affirmation at an object by saying "You are wonderful" or "I really enjoy my time around you". (b) The participant explains how the voiced positivity will make the object grow after each phrase.

On the contrary, interactions that required breathing as an input were only generated in SocialVR. For instance, a group of participants ideated an interaction that required “breathing in or out” as an input to trigger the “fly up or down” output (Figure 80). They used their microphones to make the “breathing in” or “breathing out” sounds and accompanied their breathing with body movements inwards (Figure 80.(a)) or outwards (Figure 80.(b)) from their chest.

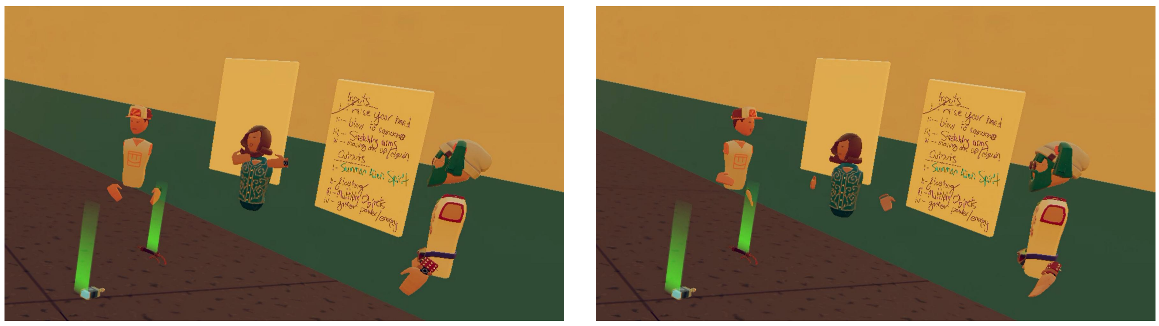


Figure 80: Breathing interaction in SocialVR to fly up or down. (a) The participants breaths in to fly up while moving her arms inwards to her chest. (b) The participants breaths out to fly down while moving her arms outwards from her chest.

The only interaction coded as multi-modal was produced in the videocall workshop for magical interactions, since it used the full-body as well as the voice. The interaction consisted of a sequence of inputs where they needed to (a) “start out sitting in a ball” (Figure 81.a), (b) “explode up and outward” and (c) “scream a positive affirmation for someone” (Figure 81.b) to trigger the output “teleport to that person” (Figure 81.c).

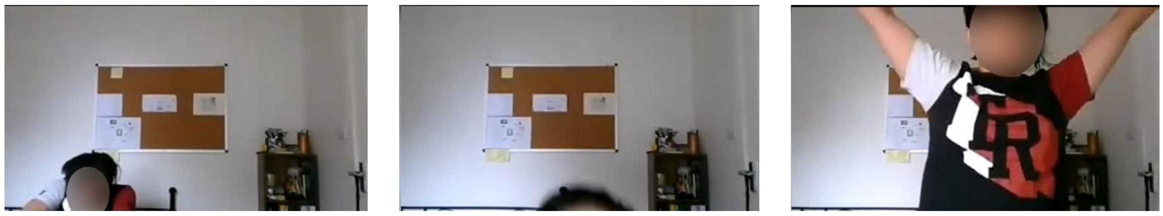


Figure 81: Multimodal interaction in a videocall. (a) The participant starts by sitting down forming a ball with their body. (b) The participant continues to jump as if “exploding up and outward”. (c) The participant voices a positive affirmation about someone to teleport to that person.

4.3.2.8 Input Space: Expressive Input Space

From the above we can see that participants expressed their interactions with a diverse range of inputs. Furthermore, the nature of this inputs is relevant to our case study. Approximately 93% of the inputs participants used in their interactions are gestures (100 gesture codes out of 107 expressive input space codes). We understand a gesture as any movement that requires time to be performed. That leaves the door open to interpret two sub types of gestures: repetitive gestures and sequential gestures (Figure 82). A repetitive gesture is any movement that needs to be repeated over time without a particular order in order to produce its output (I.e., pedal hands to play musical tones). A sequential gesture is any collection of movements or poses that needs to be performed in a specific order.

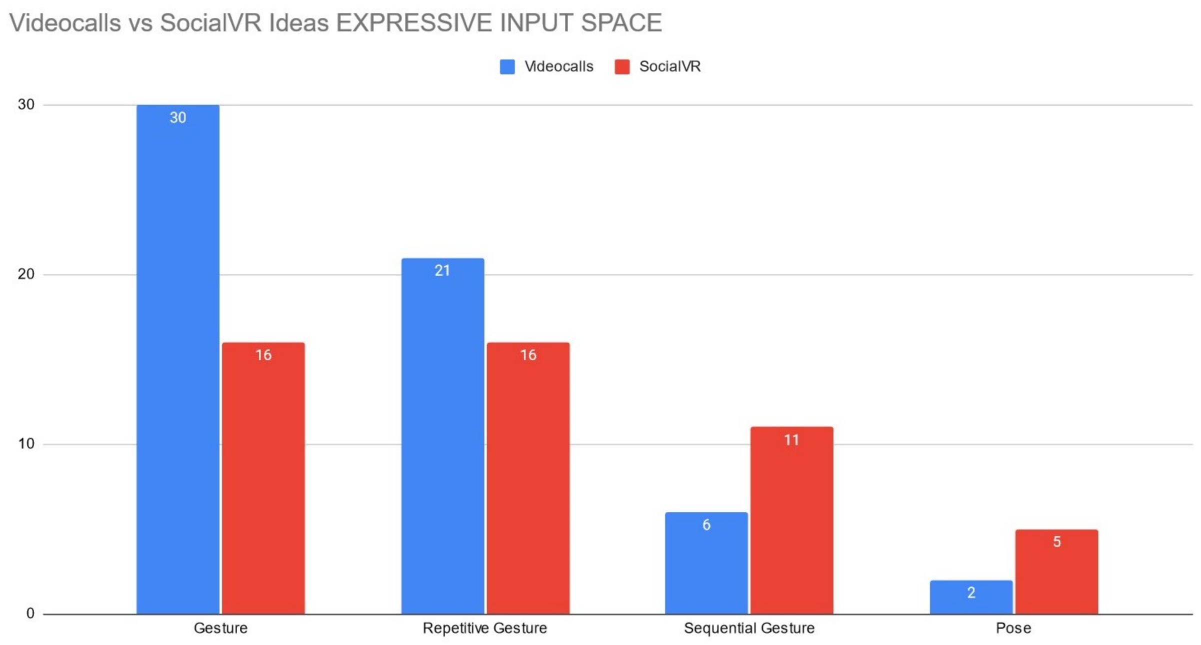


Figure 82: Comparison of low-level codes for generated ideas under the theme Expressive Input Space (a generated idea could have more than one code assigned, or no suitable code assigned)

Examples of simple gestures can be found in figure 65 for SocialVR or in figure 71 for videocalls, where participants ideated the "hug an enemy to destroy it" and "swing axe to talk" interactions. Both movements required the participant to perform a single movement with their body in order to trigger their respective output. However, for the case of repetitive gestures, an example that illustrates this be found in figure 83 for SocialVR and figure 63 for videocalls. In the case of these repetitive gestures, participants performed the movement continuously, pedaling the hands or constantly chewing, in order to affect the output. In contrast, sequential gestures use a combination of gestures or poses in a specific order, such as in figure 84 where participants generated an idea in SocialVR for a horror game where the player would need to first slice their hand's skin, then place their hand above an area to summon a demon. An example of a sequential gesture ideated in a videocall can be found in figure81 that mixes movement and audio sequentially to allow the player to teleport.

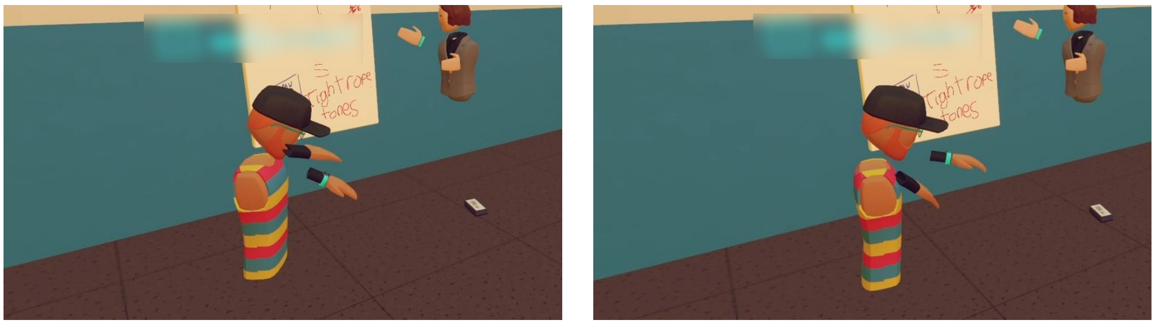


Figure 83: Expressive Input Space - Repetitive Gesture in SocialVR. Participants ideated the gesture "Pedal hands to play musical tones". The speed in the gesture's repetition would control how fast the track plays, similarly to how a mechanical musical box would work.

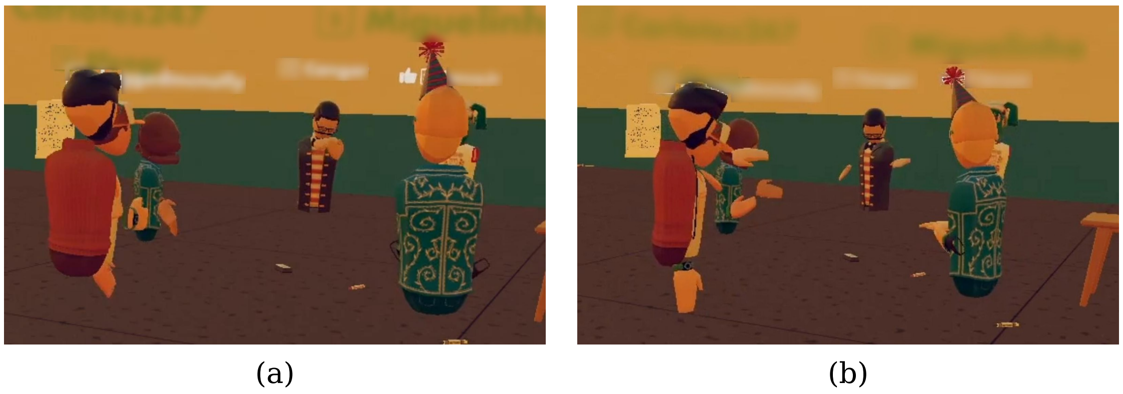


Figure 84: Expressive Input Space - Sequential Gesture in SocialVR. (a) First, participants need to "slice their hand" to open a wound. (b) Then, participants need to let the blood drip on floor to open a demon portal.

On the other hand, only 7% were poses (7 out of 107 expressive input space codes). An example from SocialVR can be found in figure 85 where participants ideated the "arms up" pose to immediately create a protective bubble in an archery game. An example from a pose ideated in the videocalls workshops can be seen in figure 62.a, where a participant demonstrated how their idea of a resting pose would look like on camera.

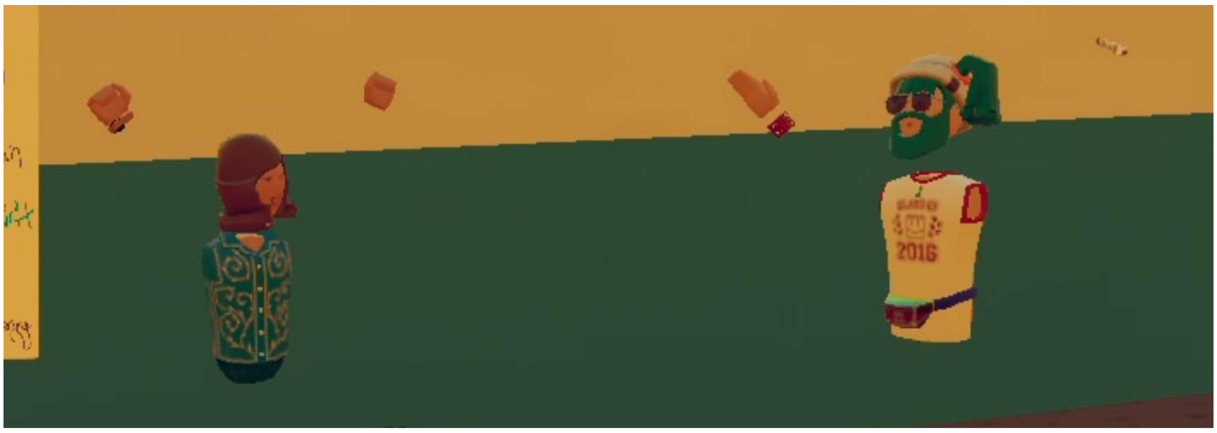


Figure 85: Expressive Input Space - Pose ideated in SocialVR. Participants raise their arms in an open body pose to immediately create a protective shield in an archery game.

4.3.2.9 Expressive Ideation Space: Exploration of Affordances

In both mediums, the exploration of the possibilities of the medium was a consistent finding. We coded this instances as either "Embodied Demonstration" or "Embodied Exploration". An embodied demonstration instance is a bodily enactment of any of the interactions participants ideated during the workshops (i.e. Figure 65 or 75). On the other hand, in an embodied exploration instance participants would explore up to what extent they can make use of their bodies and use of the software itself to express their ideas to the rest of the group. For instance, a participant might move the camera or explore the software options to see how they can show a movement or an element from their physical space in a videocall (Figure 86 b-c). Additionally, participants would make use of background filters that show video clip of a video game from which they extract some of the movement interactions that they were discussing during the ideation process (Figure 86.c).

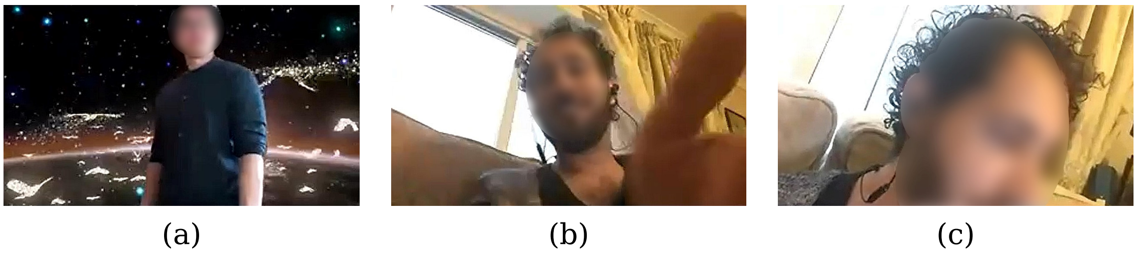
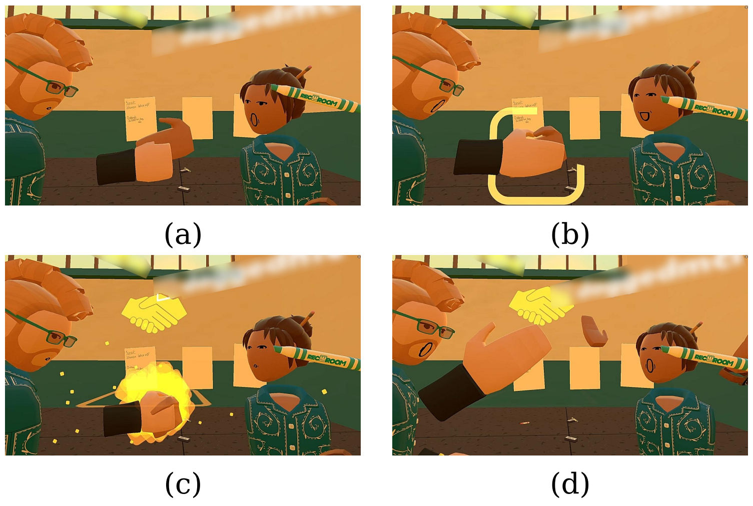


Figure 86: Expressive Ideation Space - Exploration of affordances in a videocall. (a) A participant changed their videocall background to better reflect the movements and mechanics from a game they enjoyed. (b)-(c) A participant moves his webcam trying to find a good viewpoint to demonstrate his movements on camera.

In social VR, the exploration of affordances was even more clearly observable during analysis. The platform used for embodied ideation, Rec Room, has a particular way of affording virtual avatar control via VR controller input mappings (i.e. press the VR controller trigger to open/close virtual hands). As this mapping is not standardise, because it is still too early for social VR apps to converge into a common controller-avatar mapping, participants would collaboratively explore what kind of interactions rec room affords. For example, we could observe how participants would gaze at each other, and at their own virtual body, to perform movements and interactions in a curious and exploratory way to understand what were they able to do with their virtual bodies. Furthermore, there would be verbalizations of this phenomena to request this kind of knowledge from their co-located peers (Figure 87).



R.1 (speaks): "... How dow you, how do you [do a handshake]... **(frame a)**"

L.1 has his fist closed because of gripping his physical controller, but R.1 doesn't know how to close her avatar fist.

L.1 (speaks): "If we... if we both do it... I think you... I can't remember if you have to grip. Yeah, yeah, you need to grip"

R.1 closes her avatar fist by gripping her physical controller.

R.1 (speaks): "Oh yeah I see now" **(frame b)**

L.1 and R.1 both close their avatar fists on top of each other which then triggers an scripted "add friend" audiovisual animation **(frame c)**.

R.1 (speaks): "Friended! **(frame d)**"

Figure 87: Expressive Ideation Space - Exploration of affordances in SocialVR. This example illustrate how participants in SocialVR jointly explored the affordances of the space and their avatars. Participant R.1 (to the right of every image) wants to do a handshake, which is a supported scripted animation inside Rec Room. Participant L.1 (to the left of every image) explains her how to do the handshake, but he doesn't exactly remember how. By moving and talking they both explore the affordance of their controller-avatar mapping and finally manages to do a handshake.

4.3.2.10 Expressive Ideation Space: Co-located Playfulness

Another observation we made of remote ideation was how playful participants could be in both mediums and how participants were making use of a shared space. Playfulness was understood as any moment in which participants behaved lightheartedly with a focus on fun. The codes used to instantiate different playfulness manifestations were "Play", for when a participant or group playfully interacting with each other outside of the workshop requirements but during the workshop time; "Joke", for whenever a participant made a fun comment intended for the rest of the group; or "Laughter", for whenever participants would overtly express laughter audiovisually. Playfulness was present in both mediums, but we observed less limitations for it in SocialVR because of participant co-location in the same virtual space.

Firstly, participants explored the virtual space with curiosity and used the virtual props playfully, interacting joyfully among themselves. We coded this instances as "Play" during our video analysis. (Figure 61.b). Furthermore, participants would use props in the virtual space playfully, and make use of their co-location by jointly interacting with them (Figure 88.a) Participants would as well sporadically play with the virtual props whenever they would finish a task or there was a moment of breakdown (i.e. a participant was disconnected or there was an audio issue with a participant). Play many times would be inviting for other participants near the ones that were playing, or even for participants that were still in the middle of a task but where inclined to join this sporadic play.

Later on, during the body sensitising and embodied demonstration sessions, participants could form group structures such as circles so that they could gaze each other and communicate naturally (Figure 62.b). Moreover, participants would approach each other in the virtual space and jointly and playfully bodily explore interactions that would require co-located cooperation to be performed (Figure 88.b)

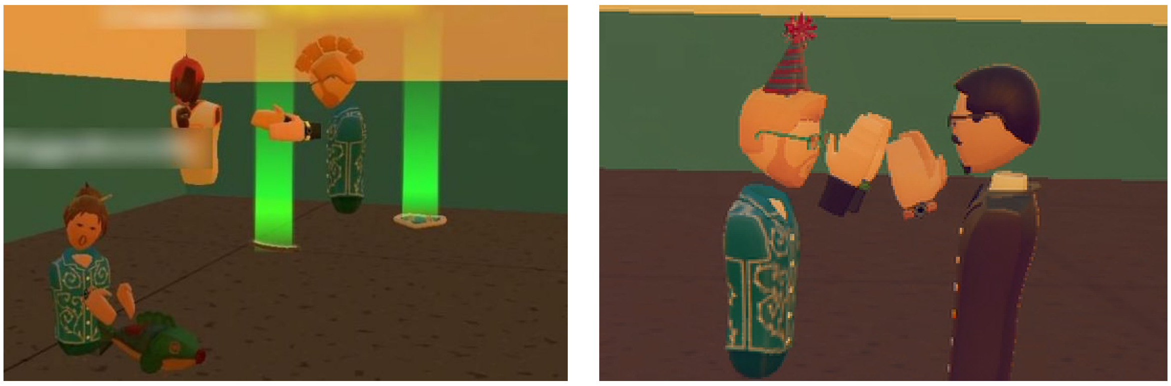

(a) (b)

Figure 88: Expressive Ideation Space - Co-located playfulness. (a) Participants use a virtual fish prop playfully to collaboratively demonstrate how to pet a dog with a particular movement. (b) Awareness of co-located peers while bodily demonstrating as participants collaboratively perform movements requiring their co-location and with two people in a specific pose.

4.3.2.11 Expressive Ideation Space: Challenges in Remote Ideation

Even though participants can verbally and non-verbally express themselves during a videocall or a group meeting in SocialVR, there are a series of limitations on expressivity inherent to each medium.

The observed challenges on the ideation space in videocalls are as follows: (1) the field of view of a webcam is many times limited, not allowing participants to fully gesturally express themselves when they are communicating through video (Figure 89); (2) the resolution of a webcam can greatly differ between participants, which can affect how group members perceive an individual (i.e. gesturing of the eyes, eyebrows, and facial expressions in general, can be hard to perceive in low resolution cameras); (3) the number of frames per seconds that a webcam streams can be a limitation during non-verbal communication in a video feed, for instance a participant might want to bodily demonstrate a quick gesture, such as a jump or sitting down promptly on the chair , hence if the video feed skips several frames the rest of the group might not fully capture the nuance of the movement (Figure 90); (4) there can be connection issues during a remote group video call, such as for online bandwidth, audio or video problems from the video call software or hardware(i.e. participant's webcam might disconnect because of a battery loss or general compatibility issues between hardware and software); (5) participants are still using their regular working machines, therefore there is room for them to get distracted by notifications from their operating system, or for participants to follow up on a question or idea by browsing the Internet, which can cause distractions.

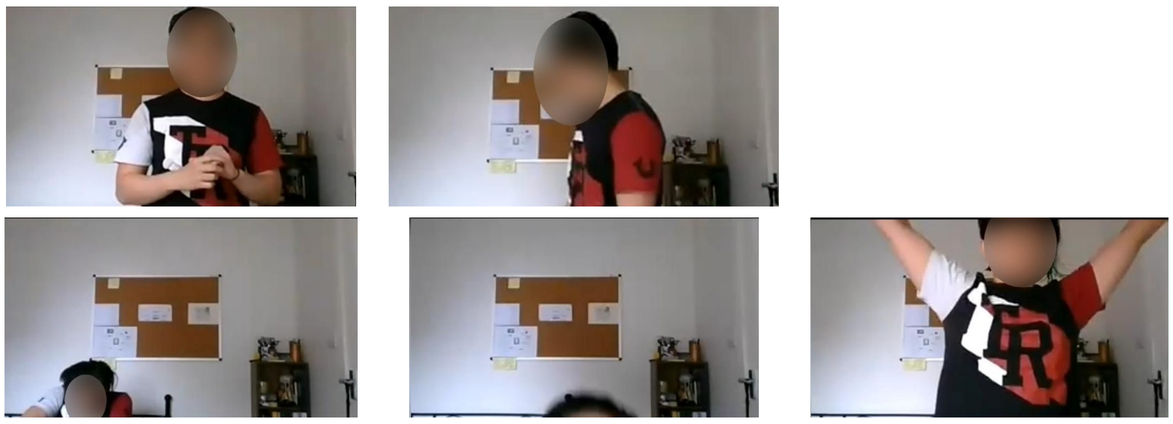


Figure 89: Limited FOV of webcam (participant getting out of view while performing movement). The participant wants to show a movement that requires them to lay down on their bed, but they can't do it easily without moving the webcam. The participant decides to perform the movement in real life and not show to the camera the entire sequence.

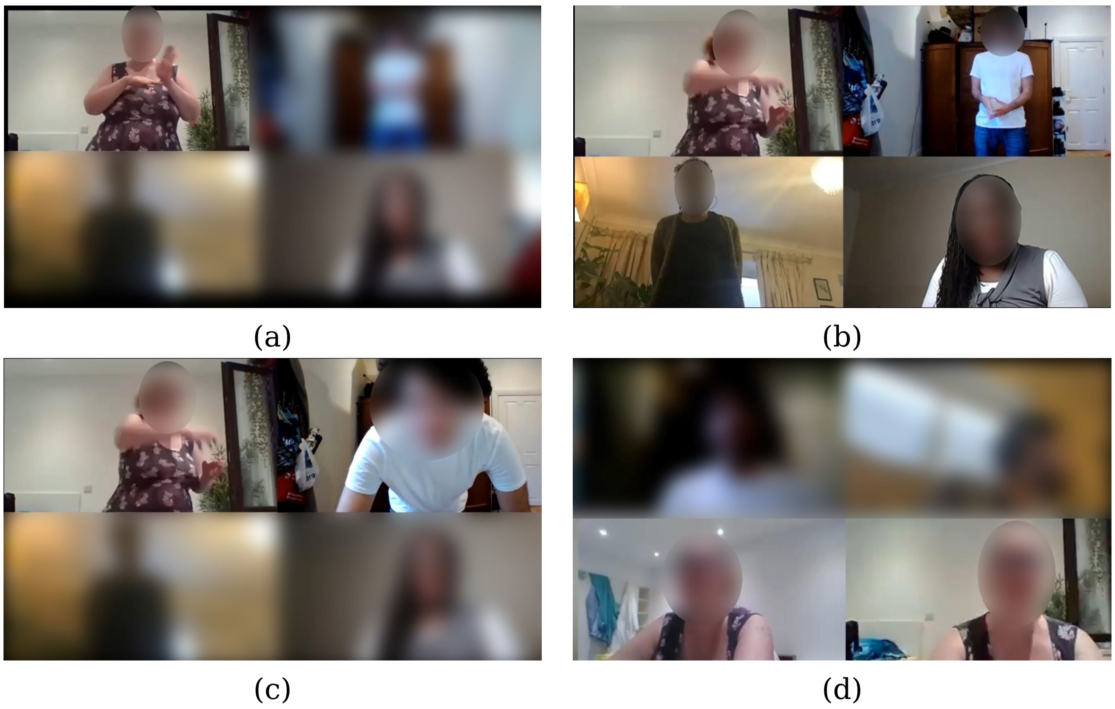


Figure 90: Connection problems in videocall (the frame freezes). (a) A participant is demonstrating a movement to the rest of the group. (b) The webcam of the participant performing the movement freezes the video-feed and there is a moment of break-down. (c) Another participant approaches his computer trying to see what happened and how to help the other participant without video. (d) Later on, the participant finds a solution by joining from two devices, but their engagement in the embodied ideation process gets negatively affected.

In a social VR setting, the affordances of the medium for ideation are different. We found a series of challenges behind ideation particular to SocialVR: (1) stand alone virtual reality headsets that don't require a PC to be used (i.e. Oculus Quest) require a battery to work, therefore participants need to make sure that they have a full battery when joining a workshop or otherwise they might not be able to ideate as the rest of the group; (2) motion sickness is a known effect that can appear when participants use virtual reality, hence the researcher needed to account for this effect by regularly checking on participants how they felt and inviting them to stop using VR for a short period of time and re-join the workshop when they feel better if they wish (Figure 91). Motion sickness can negatively impact non-verbal communication and creativity as participants experience strong moments of breakdown from the communication and ideation in the space; (3) the relationship between the virtual avatar body representation and participant body in real life play an important role in non-verbal communication in SocialVR. For instance, a participant might want to perform movements that make use of their legs, and because of the lack of leg representation in Rec Room's virtual avatars, participants cannot really express these movements properly. Nonetheless, participants can infer these movements from the virtual avatar torso movement(i.e when the torso of the avatar moves down, participants can understand that the body of the corresponding real-life person is moving down by crouching as seen in Figure 97). This is a common problem with other virtual avatar representations in social VR spaces, as current mass marketed VR hardware only physically tracks head and hand movement from users, and the virtual avatar representations many times decide not to represent virtual legs and choose a “head-torso-hands” representation (Figure 95). (4) the affordances and the mapping between VR controllers and virtual actions can be a challenge when not mapped correctly. For instance, participants might use a new kind of headset that uses VR controllers that don’t require buttons to grab virtual objects and, by mistake, they grab a virtual whiteboard while bodily demonstrating and idea, therefore experiencing a moment of breakdown and frustration. (5) Facial expressions of virtual avatars don’t match facial expressions of participants in real life, and that stops participants from conveying nuanced meaning to their communication and ideation. (i.e. Rec room uses a series of premade and playful expressions that are generated based on participants speech as seen in Figure 92).

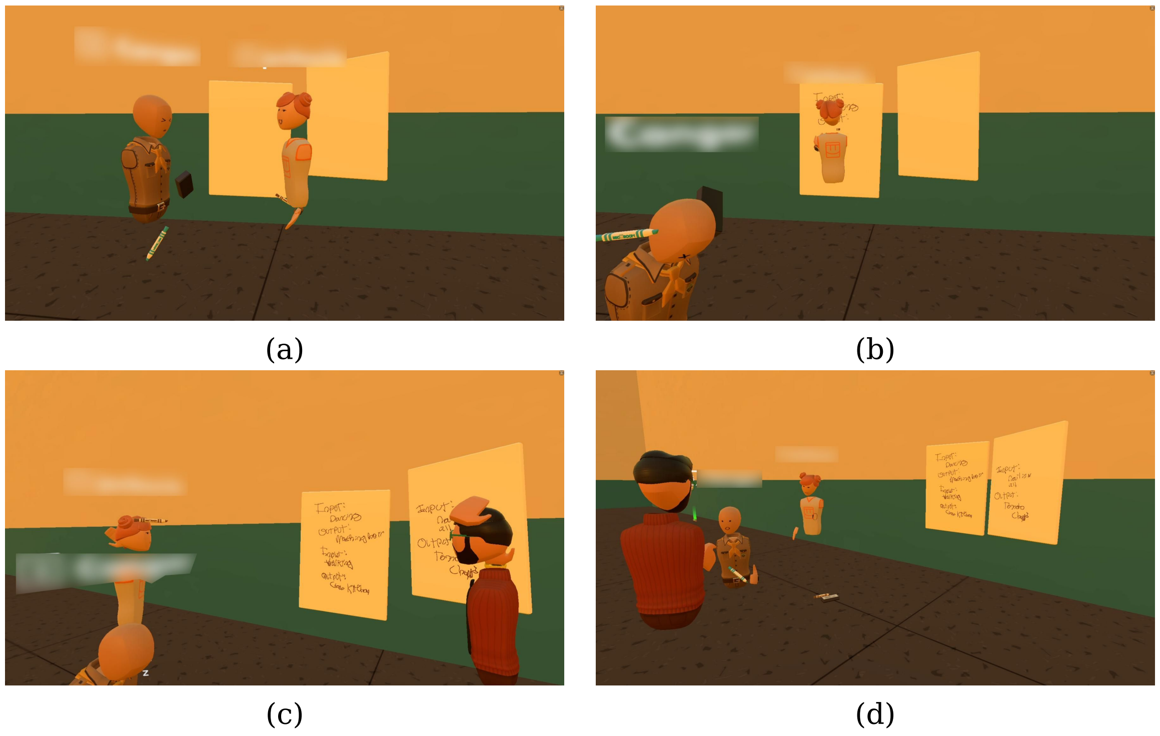


Figure 91: Motion sickness. (a)-(b) A participant tells his ideation partner that he needs to pause his participation temporarily due to motion sickness (c) The researcher takes over his participation and joins while the other participant gets back from his pause, but the creativity process gets negatively affected as the researcher is likely to introduce bias (d) The participant that experienced motion sickness comes back after a short period of time (approximately 5 minutes) and decides to participate in a seated position (as can be indirectly inferred by the lower height of his virtual avatar).

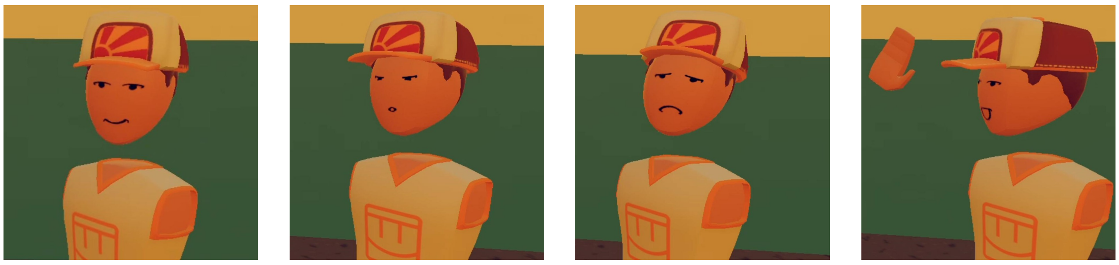


Figure 92: Facial expressions Rec Room. The chosen SocialVR platform, Rec Room, represents facial expressions in their virtual avatars through a combination of playful cartoon-like animations. The above expressions are all rendered in different frames while the participant is quiet and listening. They don't directly represent the real life expressions participants and are a barrier for full non-verbal communication, as they limit participant expression capacity. However, as long as a participant doesn't want to control their facial expressions directly, Rec Room's facial expression approach seems to be accepted by participants and no clear complaint against it was observed. Positive and playful comments were made by participants of each other facial expressions during moments of embodied demonstration or discussion.

4.3.2.12 Communication Space: Support for Non-Verbal Cues

Both the videocall and SocialVR workshops supported standard non-verbal cues. Non-verbal cues are a key component in group communication. However, each remote medium affords non-verbal cues in a different way. The observed non-verbal interactions made use of extensive gesturing to convey ideas to other participants. The nature of the non-verbal interactions was multifold in each medium.

For instance, during the videocall workshops, participants can use their fine-grained full body expressions to support the delivery of an idea (Figure 93), as well as make use of facial gesturing during conversation (i.e. a participant can show how happy or sad character can be with a body pose, or a participant can facially express how much they like or dislike a particular idea from a different participant).

Secondly, a video-call format offers the opportunity for deaf participants to communicate with the rest of the group through text chat and with sign language interpreters, thanks to the camera capturing fine-grained details from hands, arms and facial gesturing (Figure 94).

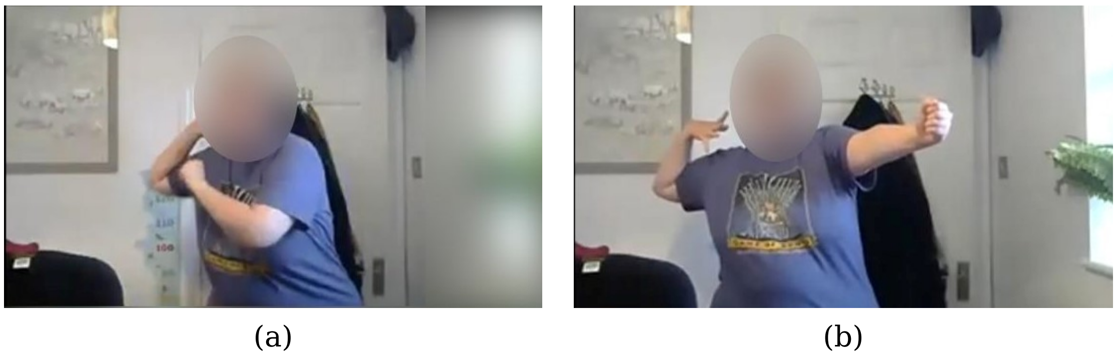


Figure 93: Showing full-body pose on webcam. The participant bodily demonstrate a full body idea in front of the camera, by bending the knees (not visible because of the limited FOV) and moving the body to the cut a tree (a) and shoot an arrow with a bow (b)

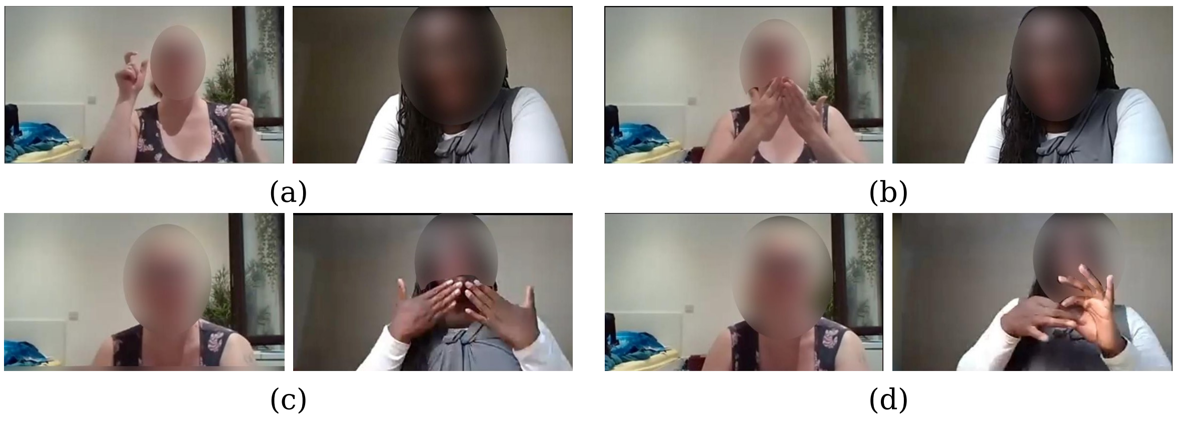


Figure 94: Communication of deaf participant with British Sign Language (BSL) interpreters. (a)-(b) A deaf participant is able to communicate with the rest of the group through a BSL interpreter translating their signing into spoken English language. (b)-(c) The BSL interpreter translates what another participant says to the deaf participant signing in front of the camera. Communication deaf participant - BSL interpreter - group is possible thanks to the webcam support for a complete range of non-verbal gesturing and high-resolution facial expressions.

In SocialVR the affordances for non-verbal cues were different due to the spatial nature of the medium. Firstly, head gaze and pointing are well supported by current implementation of virtual avatars. Current virtual avatars are limited in their body representation in Rec Room (Figure 95.a), but since they provide fully tracked head movements and hand movements gaze and pointing where extensively used by participants during communication and ideation (Figure 96). Furthermore, movements that in principle couldn't be understood due to the lack of avatar body limbs, such as crouches, could be inferred by participants because of the smooth tracking of the VR systems (Figure 97). This allowed participants in SocialVR to further non-verbally express their movements.

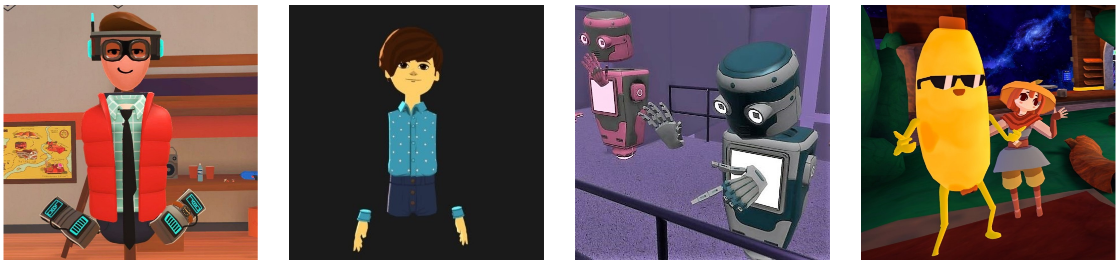

(a) (b) (c) (d)

Figure 95: Avatars in sampled SocialVR platforms. (a) - (c) The “head-torso-hands” representation is consistent among most SocialVR platforms trialed during the early stages of the study. From left to right: Rec Room, AltspaceVR and Mozilla Hubs. (d) Other SocialVR platforms, such as VRChat, support virtual avatars with full body representation, although they approximate the movement of non-directly tracked body parts through an algorithmical “inverse kinematics” approach, which results in an inaccurate representation of lower body movement.

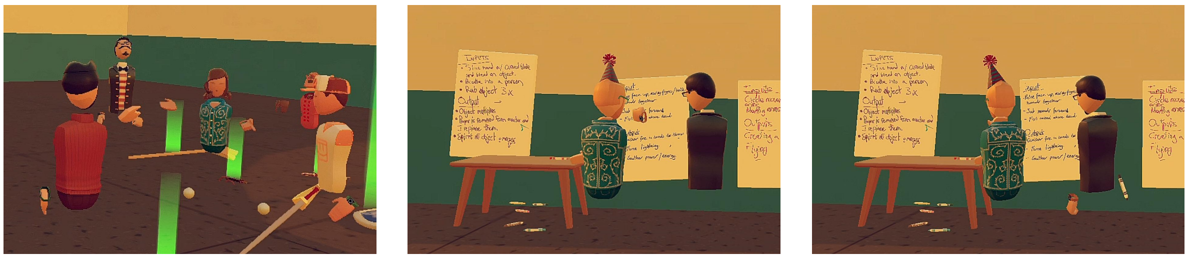

(a) (b) (c)

Figure 96: Gaze and pointing SocialVR. (a) Participants gaze and point naturally at each other during embodied ideation. (b)-(c) A participant can be seen pointing towards an idea at a virtual whiteboard for his partner to gaze there.

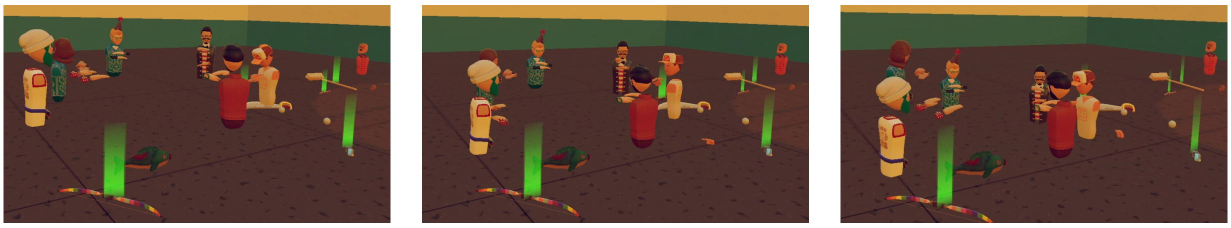
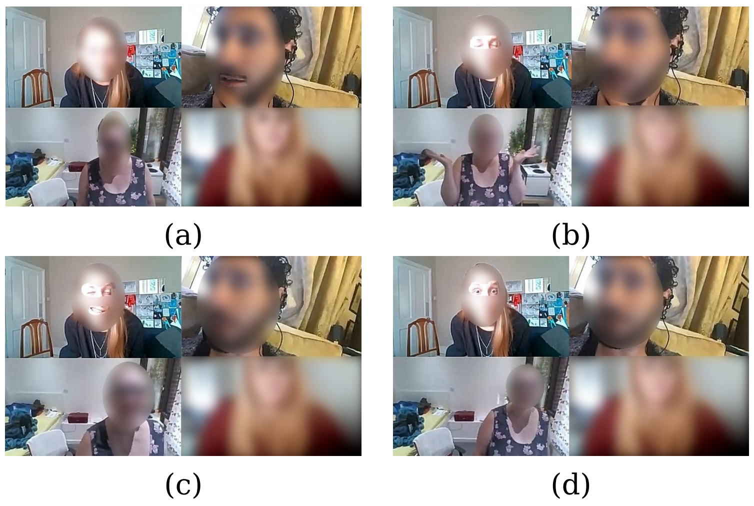


Figure 97: Crouches on VR (imgs). Participants can infer real life movement that is not directly supported by Rec Room’s virtual avatars thanks to indirect virtual cues. In this example, participants can infer the movement “crouching” because the virtual torso of the avatar moves up and down following the crouching motion discussed in the group.

4.3.2.13 Communication Space: Embodied Repair Strategies

When analysing the video data from both workshops from a communication perspective, we coded instances where participants made clarification requests about the ideas being generated (coded as “Clarification Request”), and to how movement and non-verbal cues were used by the participant clarifying (coded as “Embodied Clarification”). Repair strategies in videocalls made use of the non-verbal language presented in the previous section, with face and upper body gesturing used to bodily clarify ideas an even to finish repair sections (Figure 98.d). Repair strategies in SocialVR equally made use of the non-verbal cues inherent to the medium, with head gaze and pointing being instrumental in clarification requests and embodied repair instances (Figure 99).



R.2-top (speaks): “Do you have any other ideas on it [using eyes to focus]? **(frame a)**”

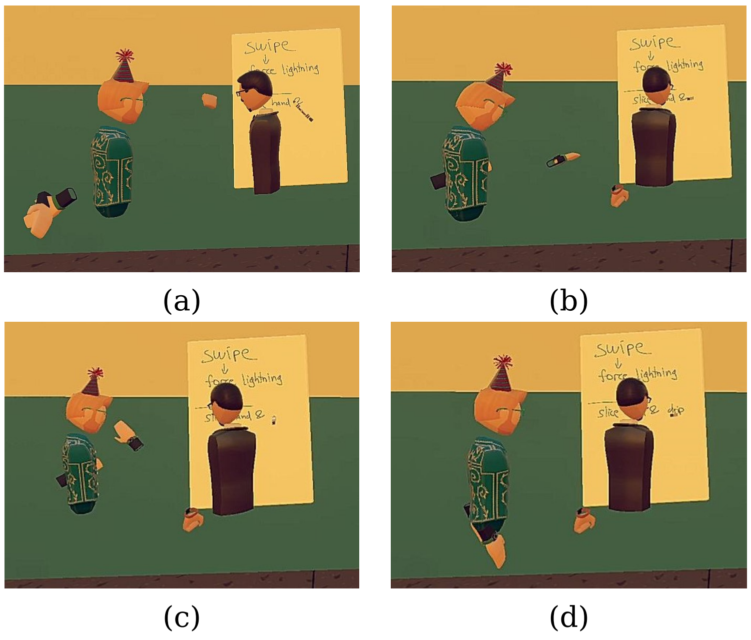
L.2-bottom shrug their shoulders to express confusion **(frame b)**.

L.2-top (speaks): “I mean, can’t you use the eyelids, so then, you could just like obviously communicate with your eyelids in that sense. So you want to execute something, maybe you can do...” L.2 gesticulates with her eyes while clarifying **(frame b-c)**

R.2-top (interrupts): “Is that like mouse-click but with blinking?”

L.2-top opens her eyes and blinks in an affirmative manner, and concludes the clarification non-verbally without verbally answering **(frame d)**

Figure 98: Communication Space - Embodied Repair Strategies in videocalls. This example illustrates how participants in the videocall verbally and non-verbally repaired requests for embodied clarifications. Participant R.2-top (to the top-right of every image) wants to clarify the details of an interaction utilising the eyes to focus. L-2-bottom (to the bottom-left of every image) shrug their arms shoulders as they don’t know what to answer. Participant L.2-top (to the top-left of every image) explains him how to do the movement, both verbally and non-verbally by gesticulating with her eyes while talking. In the end, L.2-top ends the clarification only by blinking and nodding non-verbally after another question from R-2-top.



R.3 (speaks): "And, drip blood on an enemy or do you also want to do this circular movement **(frame a)**"

R.3 performs the circular motion with his hand and gazes to L.3 while requesting a clarification for a movement.

L.3 (speaks): "Hmm, I don't know you would almost drip it on the floor or wipe somewhere and that is making your portal? **(frame b-c)**

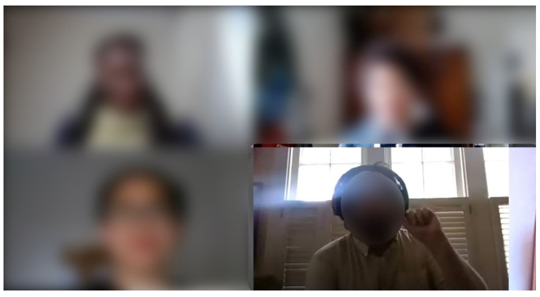
L.3 responds non-verbally by performing the dripping and wiping movements while head gazing towards R.3

R.3 writes down the interaction on the virtual whiteboard **(frame d)**

Figure 99: Communication Space - Embodied Repair Strategies in SocialVR. This example illustrate how participants in SocialVR verbally and non-verbally repaired requests for embodied clarifications. Participant R.3 (to the right of every image) wants to clarify the details of the sequence of movements to use as the input for an interaction they are generating. Participant L.3 (to the left of every image) explains him how to do the sequence, both verbally and non-verbally by gesticulating while talking. Extensive use of head gaze is done in this example.

4.3.2.14 Communication Space: Remote Turn-Taking

As part of the communication observations made during analysis was how turn-taking differed between both mediums. Turn-taking was observably more challenging on a video-call compared to socialVR. Participants would take a turn that abruptly or not indicate well when their turn finished or started (Figure 100). However, the use of names or pauses by participants could make turn-taking smoother (Figure 101). In contrast, turn-taking was observably smoother in SocialVR, as seen in Figure 102 where participants could take turns without exaggerated pauses between them.


(a)

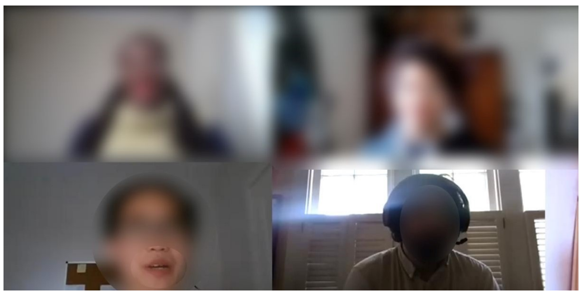
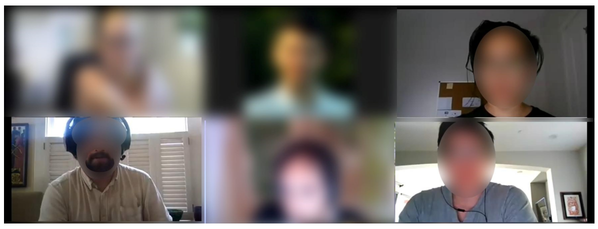
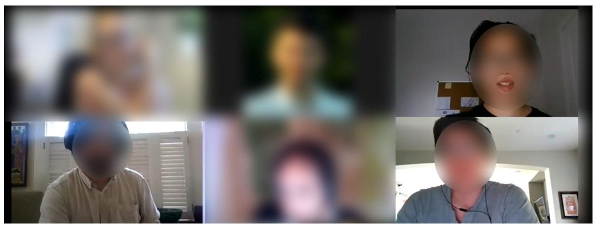

(b)

Figure 100: Confusing turn-taking on video-call. (a) Participant on the bottom-right makes a visual indication to take a turn by pulling his headset's microphone down and unmuting himself. (b) Participant on the bottom-left starts talking and takes the turn that the bottom-right participant intended to take.


(a)


(b)

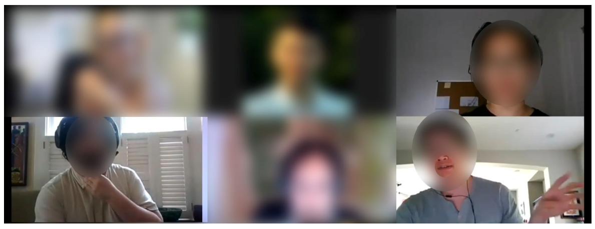

(c)

Figure 101: Successful turn-taking on video-call). (a) Bottom-left participant talks about the task and gives his opinion on a movement, then pauses. (b) Following a pause, the top-right participant says a short comment and agree with the previous statement, then pauses. (c) Following the previous pause, the bottom-right participant gives his opinion about the same topic, ending as well in a pause.

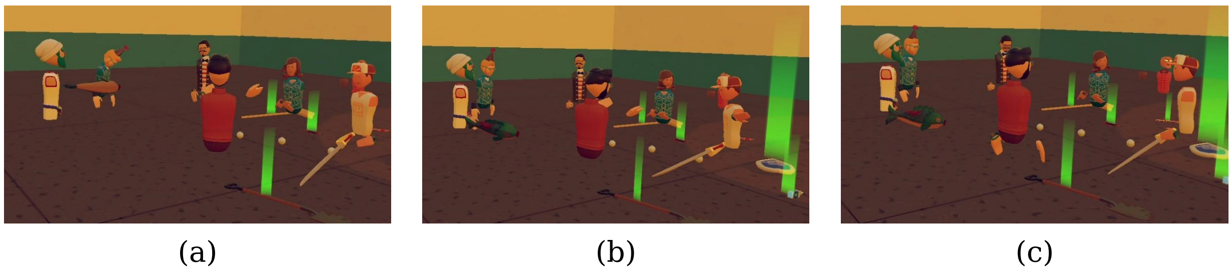


Figure 102: Successful turn-taking on SocialVR. (a): Researcher introduces the task to the group. (b): Researcher invites participant to his right to take the turn and uses pointing and gaze to smoothly pass the turn. (c): The participant pointed takes the turn, without a pause, and the rest of the group gaze at him while he speaks.

4.4 Discussion

In the following, we will discuss the results presented above. First, we will provide an overview of the findings that resulted from our analysis of participant collaboration and medium use, and discuss them in relation to their impact on embodied creativity. Second, we will discuss the implications of moving on from current out-of-medium videocalls or SocialVR in-medium platforms by highlighting potential pitfalls and opportunities for future remote mediums.

4.4.1 Nature of the Space

We saw in section 4.3.2.1 that there was a difference between the nature of ideas that dealt with objects. Between the two mediums, movements that had an effect on the creation of objects were more predominant in SocialVR whereas movements that had an effect on qualities of an object were more predominant in the videocalls (Figure 58). Furthermore, a similar difference was found in Figure 68 from section 4.3.2.6, where the SocialVR workshops had more game-like interactions generated (attacks, defense actions, equip items) that their videocall counterpart.

We believe the space where participants ideated influenced these differences. The physical qualities of a real-life space (i.e. colour, light, complexity, furniture) have been proven to have a positive or negative effect on creativity [179, 47], and this effect might be translated to a remote setting, even if virtual. The home office environments that participants where present in during the videocall workshops are "mundane" environments, in the sense that there is nothing special that can bias participants consciously or unconsciously to interact with "magical" elements (Figure 61.d). On the other hand, the SocialVR space was full of magical and fantastical elements, such as magic wands, body-sized swords, bows with infinite arrows or snowballs that never melt (Figure 61.a). Additionally, the dimensions of the in-medium SocialVR room were exaggerated in an unrealistic fashion, with very tall walls, big windows and an wide empty central space present in the virtual environment (Figure 103). The qualities of each space might have influenced participants to generate ideas that (1) control objects present in their home office surroundings during videocalls (i.e. move the curtains by picking a tea bag in Figure 74), and (2) generate objects that will populate a seemingly empty virtual space (i.e. stretch arms to multiply objects in Figure 59).

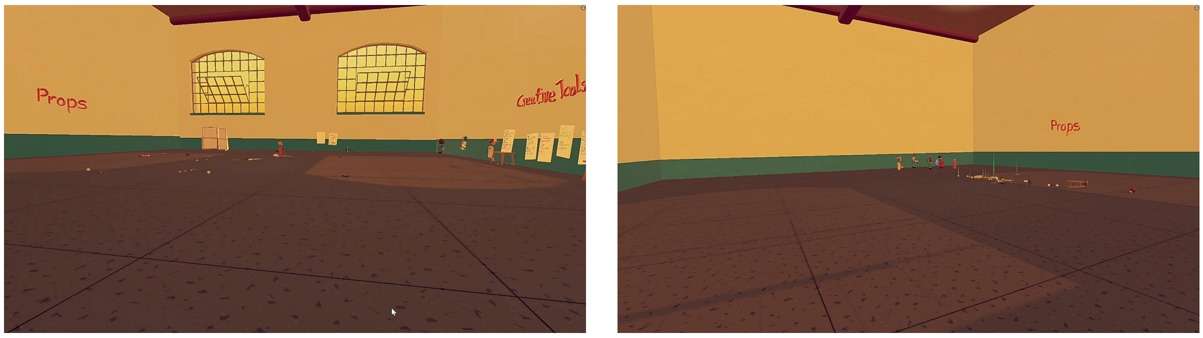


Figure 103: SocialVR space. The SocialVR space designed by the authors in Rec Room was an exaggeratedly big room, designed in a way that participants would have enough space to be separated and not hear each other during ideation. This created a big and empty central space where the body sensitising and body demonstration activities would take place.

Therefore, we speculate that the space that surrounds participants influenced their way of thinking. Specifically in-medium, the influence of the space in the embodied ideation process opens the possibility for future work where it would be possible to construct different virtual spaces to align with ideation goals. For instance, we could imagine a workshop designed to bodystorm rehabilitation movements for hospital patients happening in a virtual hospital rehabilitation room. Or a workshop designed to bodystorm technical operator movement interactor with manufacturing robots happening in a virtual automation space.

4.4.2 Joint Action Space

In section 4.3.2.10 we introduced the effect that SocialVR's spatial co-location had on embodied ideation, allowing participants to create interactions that require of multiple players to physically coordinate (Figure 72). As noted by Gaver[87], this is co-location is not available in video mediated communication. We also introduced instances of playful interaction and joint exploration of affordances in the virtual space (Figure 87). While in videocalls participants needed to pretend to interact with other imaginary players when ideating and demonstrating movement interactions (Figure 71), in SocialVR participants can benefit from the joint co-location benefits in their movement enactment and elaboration. Furthermore, the instances of play (Figure 61.b) and playful co-located interaction with virtual props (Figure 88) shows the positive effect of in-medium virtual co-location in embodied ideation. Co-location can also have positive effects on the learnability of the technology as participants can help each other explore the affordances of the platform and the avatar-controller mapping by mixing movements and conversation (Figure 87).

We believe that the positive effect observed from participant's in-medium virtual co-location comes from its impact on joint attention [192, 144] or shared attention [278], due to its effect on social cognition. Previous research on joint or shared attention used VR systems to study the effect of human-avatar gaze in attention, and showed how the medium could offer a controllable setting for experimental studies [144]. SocialVR offers the ability of participants to naturally use head gaze during communication, which is one of the key identifying factors of joint attention [192, 144]. SocialVR also offers participants to have a joint real-time understanding of the state and use of virtual props in the co-located space (Figure 88.a). Joint attention then allows participants to explore together ideas or affordances in the medium by maximising embodied communication and minimising moments of breakdown or lack of attention. This might have had a direct effect on the nature of the ideas and movements ideated in the space, therefore prompting movements cooperative and co-located in nature (Figure 88).

4.4.3 Playfulness, Novelty and Creativity

We saw how in SocialVR there were instances of playful interactions (section 4.3.2.10) and exploration of the affordances of the avatars (section 4.3.2.9). Previous literature have explored the relationship between creativity and adult play [2, 171]. Play can be a precursor of creativity [171], therefore the playful interactions that participants experienced might have influenced the ideation process in SocialVR. However, we want to go a step further and argue that novelty might also influence the playfulness of the interactions. Participant recruitment was controlled to ensure that the game designers participating in the workshops had previous professional experience with VR. The intention behind the VR novelty control was to reduce any potential affect on the exploration of the space and the interaction with other participants, since VR is such an immersive medium that the novelty itself can distort participant behaviour.

However, even though the participants had previous experience and familiarity with VR technology, not all of them might be familiar with Rec Room, the selected SocialVR platform for our workshops. The unfamiliarity might have expressed itself through the joint exploration of affordances introduced in section 4.3.2.10 (Figure 87). A participant not knowing how to close the fist of their avatar, or trigger a pre-scripted animation might indicate exploration because of unfamiliarity with the platform. Therefore, it could indicate a certain degree of novelty for some of the participants. This novelty could lead to curiosity during participant exploration of the space. Participants would look to a virtual prop and playfully explore how the prop works, and because of the co-located aspect of SocialVR and the game-like interactions that virtual props support in Rec Room, participants could find intuitive to aim the prop at another nearby avatar. The reaction of the receiving participant, even if they were not new or unexperienced with Rec Room, might be influenced by the playful curiosity from the participant that started the interaction, which in end can create an appropriate space for play between two or more co-located users. Hence, the VR space might enable novelty and play which then could influence participant creativity.

4.4.4 Bodystorming and the Adjacent Possible

In section 4.3.2.9 we reported how we coded instances of "Embodied Exploration" across in-medium and out-of-medium conditions. Some of the embodied exploration instances had to do with how a participant would explore the mapping of the VR controller to the virtual avatar in SocialVR (Figure 87), while others would be of how a participant in a videocall used their webcam to see how much of the movement is being captured (Figure 86.b-c). However, there were other sorts of embodied exploration instances in which participants explored variations of the same movement or interaction while bodily demonstrating or mimicking the demonstration of another participant.

We believe that these instances of bodily exploring variations of the same movement are connected to the adjacent possible concept [20]. The adjacent possible was introduced in evolutionary biology as a principle to explain how living organism could behave or mutate to explore and actualise what is available at hand, but has since been studied in other contexts, including design studies. [20] discussed how it is possible to design to better support adjacent possibles, but arguing that physical and virtual spaces can be built to support three different principles: diversifiability, traversability, and sensoriability. We believe that body storming follows the sensoriability affordance from [20], because when participants are bodystorming, what they are following is a full-body and mind thinking process in which they are sensing the possibility space of their bodies and movement.

This relationship of the exploration of the adjacent possibles interacting with physical movement could be potentially used in different ideation workshops. Since we saw this phenomena manifesting itself in both out-of-medium videocalls and in-medium SocialVR, we could foresee how tailoring the workshop structure to have an starting movement could make use of the adjacent possible principle. For instance a workshop that tries to ideate new ways of opening a door could start with one or two different existing ways of opening doors. Then participants could discuss how would they change the movement if they had different physical abilities or were required to change the direction in which the door opens.

4.4.5 Non-Verbal Cues, Repair Strategies and Turn-Taking influence Remote Ideation

There were differences in the support for non-verbal cues found in the videocall and SocialVR workshops as seen in section 4.3.2.12. Despite the fact that both mediums offer the chance to bodily demonstrate full-body interactions, the videocalls showed better support for facial expressions (Figure 75) and accesible gestural communication (Figure 94). On the other hand, SocialVR excelled in supporting head gazing and pointing (Figure 96), as well as the positive effect from spatial co-location in communication (Figure 87). The videocall support of facial gesturing was observably used in participant repair strategies, indicating intent or expressing emotions (Figure 98), whereas head gaze was constantly used in SocialVR during clarification requests and embodied clarifications (Figure 99). Finally, we observed how in videocalls there could be confusing turn takes during communication (Figure 100) and how turn-taking was smoother in SocialVR (Figure 102).

We believe that the support of head gaze and pointing in a co-located virtual environment positively influenced turn-taking in SocialVR, which an existing effect from face-to-face communication [135]. Similarly, the lack of head gazing and pointing negatively affected turn-taking in videocalls, but was partially compensated by speech pauses and explicit turn leading 101. However, pauses can be a problematic indicator of turn ends as communication can be further negatively affected in situations where unwanted latency occurs over the videocall network [255]. In such situations, unwanted overlaps might occur that could be detrimental to communication [245].

Furthermore, we believe that our analyses shows from a qualitative perspective how smooth communication has a positive effect on embodied in-medium remote ideation. Previous research argued that communication is an intrinsic component of collaborative creativity due to its co-creative nature [270, 96]. We agree with such statement, as we could observe positive effects from smooth turn-taking and joint attention in the SocialVR embodied explorations of ideas. We could observe how participants would collaboratively enact movements to each other during the force-pairing stage by "picking-up" movements that another member was performing and adapting it slightly. These instances of collaborative embodied exploration effectively constituted instances of movement co-creation where non-verbal and verbal channels were used by group members in their remote ideation process. Nonetheless, we observed instances of embodied exploration in the videocall workshops as well, but their existence and reach was limited by the inherent limitations of the medium, as participants could get confused as to when to contribute verbally to the conversation and even physical expressions of turn-taking could get missed during the remote ideation process (Figure 100).

4.4.6 Medium Constraints

In section 4.3.2.11 and 4.3.2.12 we showed how in SocialVR, unlike in the videocall counterpart, there was a lack of fine-grained expressivity of body limbs and facial expressions due to current limitations in consumer-grade VR systems and the avatar-controller representation (Figure 95). On the other hand, the videocall medium had a clear limitation from the camera field of view (FOV) that didn't allow participants to show to other participants their full bodily intent during ideation (Figure 89). Hence, each medium can constrain the ideation process because of the inherent limitations of the technology.

However, even though apparently the medium limitations are a constraint for collaborative ideation, we argue that in the case of VR they can also be a desirable constraint. The existing technological constrains limiting ideation have a 1:1 mapping with existing consumer-grade technology [210]. The ideas generated in the SocialVR workshops are well fitted to be prototyped with current technologies in VR, whereas several of the ideas generated in videocalls require of sensors that are not currently mass-marketed (i.e. eye-tracking sensors, lip sensors) or sensors that are unlikely to even exist (i.e. sensor to detect the granularity of the jaw movements from the "chew to lead dog" interaction in Figure 63). Therefore, paradoxically, technological limitations from the medium can be useful during ideation because they reduce the likelihood of generating ideas that cannot be currently implemented. We consider this a core contribution, suggesting a positive effect of in-medium embodied ideation for VR motion control design.

4.4.7 Making the Most of Each Medium

As we have seen throughout this study, neither of the two mediums is perfect and each medium supports remote embodied collaborative ideation differently. Furthermore, we want to take this opportunity to look into the future and suggest strategies to make the most of each medium in a remote setting. One of the biggest challenges found in the video calls is the limited field view. This is something that could be leverage by translating known solutions from video-based systems into video-mediated remote embodied ideation. For instance, we could imagine a setting where specialised rooms are equipped with cameras in different positions and angles to cover for this lack of field view. In addition, once the participant is joining a group discussion, the system can intelligently show the camera feed that best captures the entire movement of the participant. In the case of the participant leaving the field of view of their current camera, the system would automatically switch to the next one that captures their full body.

In social VR, the challenges in communication were caused by the lack of full body tracking and facial tracking. We believe that these barriers are mostly technological. Hence, we envision a future where 1:1 avatar representation of nonverbal cues is highly accurate and responsive. That way, remote group embodied creativity workshops can support embodied exploration of movement interactions that make extensive use of facial features or lower body interactions. However, there are still some usability challenges that will need extensive HCI research. For instance, one of the usability challenges that we observed in social VR was unwanted object grabbing. We believe that for a fully successful remote embodied creativity experience, there needs to be better support for basic object interactions with creativity tools such as pens, whiteboards, or natural and intuitive drawing interactions. Enhancing the capabilities of such creativity tools in the immersive space (i.e. by allowing magical super capabilities to a pen or whiteboard), would also allow for a better exploration off embodied design interactions. Furthermore, unlike in video calls, participants in social VR spaces don't have access to web browsers to search an import preferences into the group conversation. Social VR spaces could support browsing and researching audio visual references from the Internet to enhance idea sharing by just being able to "show what you mean" to the rest of the group.

4.4.8 Looking into the Future

Finally, we would like to bring our ideal scenario based on the findings from this study. Even in a future where VR headsets are fully technologically developed, there are some important drawbacks to the requirement of using VR for a prolonged period of time. We know that a portion of the population would suffer from motion sickness when using virtual reality. Additionally, extensive use of VR can cause fatigue either on the eyes or on the neck. Still, there are very important advantages in remote embodied ideations when using VR. Hence, we would want to leverage those advantages while minimising the disadvantages. That is why we suggest a hybrid remote medium, where moving from video call to a social VR space, and vice versa, with the same group of people is effortless and very comfortable. With a hybrid remote medium we are able to (a) reuse the learned affordances from regular desktop and videocall usage, reducing motion sickness and physical fatigue; and (b) incorporate the best aspects of remote bodystorming in a co-located virtual space with SocialVR. In addition, we would support McVeigh-Schultz and Isbister's[182] call to go beyond replicating real world affordances and use the technological capabilities of both video and social VR to enhance what is possible in bodystorming.

4.4.9 Limitations

We are aware that this study has a series of limitations that could potentially affect the results and our conclusions. Firstly, the particular case study and population chosen to participate in the workshops have a strong effect on the types of interactions ideated and, potentially, in the playfulness of the in-medium SocialVR communication. We acknowledge that for the results of our study to be fully generalisable more research is needed with different case studies and population backgrounds (age, gender, culture, professional occupation, education).

Additionally, the data from this study was collected during the first wave of the worldwide COVID-19 pandemic, which forced the out-of-medium condition to be remote. The pandemic is likely to have an effect on the attitude that participants bring into remote workshops, since most of us were experiencing limited physical and social interaction when the workshops were run. Therefore, there might have been a heightening effect on participant positive or negative social experience, resulting in a potentially abnormal behaviour. Based on our observations and analysis, we believe that such effect is likely to be small.

4.5 Conclusion

We have observed that both in-medium and out-of-medium conditions can support embodied collaborative ideation, with special emphasis on the in-medium opportunities.

On one hand, the expressivity of out-of-medium video calls is apparently more limited because of the lack of co-presence and limited field of view. However, we argue that video calls are a successful medium when it comes to generating movement interactions where facial features are extensively used because of the high-resolution capabilities of modern webcams. In addition, we found that participant explored a more diverse range of inputs modalities in remote group videocalls. Still, the lack of support for head gaze and pointing can negatively affect embodied creativity in a group video call setting since it can deter from smooth communication.

On the other hand, in-medium social VR offers a very promising space for remote collaborative embodied ideation. There seems to be strong benefits when communicating between participants in social VR because of the support for standard nonverbal cues, such as head gaze and pointing. In return, conversation events such as turn taking are more fluid and group communication is strengthened, allowing participants to better explore the embodied design space. In addition, the support for virtual co-location seems to benefit the ideation of social interactions that require inter-player cooperation. Nevertheless, expressivity in VR is challenged by the lack of granularity and control from users' facial expressions. Plus, the relationship between the in-medium avatar representation and communication seems to be equally important. We argue that as long as the social VR platform offers a consistent in-medium affordance of the interaction to generate (i.e. generating VR movement interactions through Social VR), is not a barrier for creativity and the benefits of the in-medium work outweighs the limitations that it presents.

Finally, there seems to be a prevalence on ideation of object interactions, yet from a different nature in or out of medium. Out-of-medium ideas seemed to interact with existing everyday objects more while in-medium ideas seemed to focus on generation of objects. We argue that participants creative thinking might have been influenced by the space they inhabited, with out-of-medium workshops taking place at home offices and in-medium workshops taking place in an exaggeratedly big virtual space lacking furniture or a rich diversity of props. Exploring workshop designs that take place in specifically designed in-medium virtual environments can steer participants to align ideas generated with workshop goals

Therefore, we have shown the positive outcomes of in-medium embodied ideation for VR game motion control design because of the richness in playfulness, creativity, communication and medium affordances. Next step in the embodied design process is idea implementation and evaluation, something that Chapter 5 will investigate in detail.

5 IML for Prototyping VR Motion Controlled Gameplay

5.1 Introduction

The previous chapter investigated in-medium embodied ideation from a VR game creator perspective. This chapter focuses on the second and third stages of the embodied motion control design process: implementation and evaluation. Prior research argued that embodied interaction design enjoyed a focused loop of design and implementation for embodied creative reflection [82] and suggested IML as a technological intervention to support such loop [93]. Thus,this Chapter will look at the opportunities and challenges that game creators encounter in prototyping and evaluating game motion controls with InteractML, our novel IML tool.

To do so, we conducted 4 game jams where we asked participants (n=17) to create a VR game motion control using InteractML over 2-3 days. This framing served for participants to consider game software engineering and player experience while creating their IML prototypes. To analyse how participant game developers implement and evaluate movement interactions in InteractML, we particularly looked at feature selection strategies, perceived output quality, and model evaluation strategies. Perceived output quality refers to the degree in which participants subjectively judged their implemented movement interactions from a human-centred IML perspective [276]. Finally, we were interested in how participants used the tool as part of their game creation process inside the game engine. In sum, we studied how game creators using IML tackle (1) feature engineering, (2) IML iterative training, (3) IML evaluation with an experiential focus, and (4) IML integration into a game engine. To do so, we collected field notes, observations, think-alouds, interviews and focus group data. To make sense of the data collected, we followed a reflexive thematic analysis.

We found that creators followed an embodied thinking process during their prototyping, which affected their engagement with the IML model steering loop and evaluation. We conclude that while IML can serve well the purpose of implementing more nuanced movement interactions for motion-controlled VR games, it presently suffers from a number of limitations, some of which are inherent to the supervised learning paradigm, some specific to the current InteractML implementation. These require further design work to make movement interaction design with IML truly accessible to non-ML experts.

5.2 Method

Grounded in our research aim of understanding opportunities and challenges that game creators encounter in prototyping and evaluating game motion controls with

InteractML, we chose a qualitative study design. Our qualitative approach is necessary because prior research showed how traditional ML quantitative metrics aren't able to inform about human-centred dimensions of IML [73, 26, 276]. To achieve an understanding of prototyping and evaluation processes, we opted for a workshop structure where participants were initially introduced into ML, IML and InteractML specifically, and then proceed to free individual prototyping. We targeted a sample size of 12 to 20 to ensure good code and meaning saturation [112]. We chose to collect field notes, observations, think-alouds, interviews and focus group data to better qualitatively capture the human-machine interaction phenomena and mental model maturing of participants throughout their working process. For analysis, we adopted reflexive thematic analysis to recursively explore and interpret patterned meaning about prototyping and evaluation of game motion controls accross our dataset. Each workshop had a different cohort of participants that created a VR motion-controlled game using InteractML over a period of 2 to 3 days.

5.2.1 Participants

In total, 17 participants (10 male-identifying, 6 female-identifying, 1 gender varian/non-confirming) took part in the study across 4 in-person workshops, with an average of 4 participants per workshop (see Table 11). All participants were adult UK residents and recruited to have prior experience creating VR games with game engines but not experts at machine learning. 16 out of 17 participants didn't have any previous experience with machine learning. Participants were recruited via email and course leaders of UK university degrees focused on VR, interactive media, or games. While we tried to recruit working professionals from larger game studios, we were not successful in doing so, and so our participant pool was chiefly composed of students or student-level independent developers. Participant data was anonymised and each participant was assigned an anonymised identifier consisting of the beginning 'R_' and three random alphanumeric characters (e.g. R_Pv9).

5.2.2 Procedure

Workshops were conducted in person at VR laboratories of Goldsmiths, University of London and the University of York. The VR laboratories offered each participant a high-end computer with a Unity3D installation with a template InteractML project, together with a consumer grade VR headset (i.e. Oculus Rift, Oculus Quest 2, HTC Vive, HTC Vive Pro). The first two workshops lasted for two days, but after participants reported that they would benefit from at least an extra day of interaction with the tool, we added an extra day of implementation time (vi, see table 12) for the last two workshops. The core part of the study involved individually implementing movement interactions of their choice using InteractML in VR. This was to observe each participant's individual engagement with all stages behind InteractML's workflow, instead of risking that only part of a group would

Table 11: Participant details per workshop

Participant ID	Workshop	Age	Gender	VR Exp	ML Exp
R_3P5	1	34	male	High	None
R_1Ih	1	31	female	Medium	None
R_W1X	1	42	male	Medium	None
R_3Mc	2	35	female	Low	None
R_3Gw	2	20	female	Medium	None
R_2bW	2	22	male	High	None
R_Pv9	3	25	Non-conforming	Low	None
R_11i	3	22	male	Low	None
R_0oi	3	22	male	Medium	None
R_3qq	3	27	female	Medium	None
R_2b1	3	22	female	Low	None
R_ZEL	3	26	male	Low	None
R_1BQ	4	24	male	Medium	None
R_1H7	4	19	male	Low	None
R_2uv	4	28	male	Low	None
R_2bU	4	18	female	Low	None
R_1Ib	4	29	male	Low	Basic

Table 12: Workshop Phase Lengths

(i) IML introductory lecture	60 mins
(ii) Bodystorming ideas	10 mins
(iii) Bodystorming IML classifier	10 mins
(iv) InteractML tutorial feature by feature	2h - 4h
(v) Breaking down previous ideas on whiteboard into features	10 mins
(vi) Free implementation of ideas with InteractML	Rest of workshop (1-2 days)

work with InteractML as part of group work. While InteractML supports a range of classifier algorithms, unbeknownst to the participants, we restricted the workshop implementation of the tool to the k-nearest neighbour (kNN) classifier. This was to simplify participant teaching and later analysis. Regression or dynamic-time warping (DTW) algorithms differ in how they calculate decision boundaries, and this variance could have affected how participants understand and interpret the system output.

The full workshop structure can be found in table 12. At the beginning of the workshops, participants were briefed about the purpose of the study and introduced themselves to each other. participants then received an introduction into IML, with a bodystorming session followed by a hands-on InteractML tutorial.

The introduction was a 60 minute lecture covering the theoretical basis of supervised learning and focused on classification tasks, labelled datasets, and the human-in-the-loop component of IML.

The bodystorming session served both to allow participants to ideate potential interactions to implement (as seen in the previous chapter) and to conduct a didactic and performative IML exercise. It was always done around a whiteboard, either as one big group or with two groups of at least two people. The instructor would pretend to act out as a supervised learning classifier and react to the training movements that participants would pretend to teach the instructor. The intention was to simplify learning of the machine learning methodology before participants were presented with InteractML's interface, thus allowing participants to learn the logic of ML before learning the interface or InteractML.

The hands-on InteractML tutorial sequentially introduced all features and each participant implemented a simple classifier utilising each feature, thus implementing 5 small classifiers over the course of the tutorial.

Finally, the instructor requested participants to go back to the whiteboard where their ideas were and discuss how to break their interactions down into features. The tutorial lasted for half-a-day to a day depending on participants' learning needs, as some cohorts learned quicker than others.

The remainder of the workshop time was used for individual prototyping of at least one full movement interaction for a small VR game. With a full movement interaction, we mean a combination of a working classifier and a custom visual representation in a virtual scene.

5.2.3 Data Collection

Consent to voice and video record the workshops was gathered from participants. Video recordings consisted of relevant spontaneous behaviours that participants showed, such as unexpected usage or modifications of the tool/system, or illustrative working processes. During the workshop, we gathered field-notes with observational insights and ran regular video-recorded think-alouds by seating close to a participant that was observably doing something of interest, then requested them to verbalise their thoughts and rationale while using InteractML. At the end of the workshop we asked participants to save screenshots of their projects and their IML graphs, and organised a one hour focus group about the experience in the workshop and their design process with InteractML compared to other engine toolkits. Finally, on the day after the workshop, we arranged individual semi-structured interviews over a video-call about fine-grained details covering their experience doing (a) feature selection, (b) model steering, (c) model evaluation and (d) player-experience evaluation dimensions. Before each interview, the main author presented screenshots of the participant project and relevant field notes to aid with participant's process recall.

5.2.4 Analysis

Interviews and focus groups were transcribed verbatim. To code the field notes, transcripts and IML graph screenshots, we used the qualitative analysis software MAXQDA [308]. In total, the data from 17 participants across four in-person workshops spanned 17 hours of interviews, 4 hours of focus group and approximately 60 think-aloud instances (3-4 per participant).

We performed a thematic analysis in which we followed an initial deductive top-down coding process on the data collected, followed by a second bottom-up inductive coding step in order to ensure coding reliability and capture themes missed in the first deductive phase [28].

The first top-down analysis focused on interactive supervised learning (a) feature selection, (b) model training, (c) model evaluation, (d) integration into a game engine, and (e) player-experience evaluation dimension and yielded 208 low-level codes, 24 high-level codes, 4 themes and 4 sub-themes.

We looked at participants per workshop, coding first the field notes per participant to ground our later interpretation of interview and focus group transcripts. During coding we asked ourselves questions such as; how did participants understand each feature? How did their usage of the features affect their training process? Why did they record data the way they did? What is the relationship between their data recording and their graph structure? What did they find relevant during evaluation? How did it affect their interpretation of their classifiers? How did it affect their model steering and further node creation or deletion? How did they pipe their classifiers outputs to their custom scripts? How did that affect their interpretation of their classifiers and overall model steering? What experiential qualities did they consider important about their end-result? How did they express such qualities in relation to their end-result?

Additionally, we coded participant's IML graphs screenshots looking at (a) which features were selected and (b) which classifier nodes were created, then cross-referenced their graph structure with relevant transcript instances mentioning feature selection or classifier training.

To ensure coding reliability, we performed a second deductive bottom-up analysis on a subset of the entire dataset, which included 6 participants, accounting for 6 hours of interviews, 2 hours of focus groups and approximately 20 think-aloud instances (3-4 per participant). The results from the second bottom-up analysis enriched the insights yielding 101 new low-level codes, 10 new high-level codes and one new sub-theme. The new codes were grouped into the original 4 themes, therefore they didn't change the number of themes from the first top-down analysis. Since we didn't synthesise any new themes, we assumed saturation was reached with a total of 409 low-level codes, 34 high-level codes, 4 themes and 5 sub-themes from the top-down and bottom-up analyses.

Table 13: Theme and sub-theme structure

THEMES	**SUBTHEMES**
1. Creators Displayed an Embodied Thinking Process	
2. Tension between Embodied Thinking and Supervised Learning requirements	2.1. Tension in how each feature affects model inference 2.2. Tension in understanding the particularities of each feature with regards to time 2.3. Tension to tackle the mismatch between natural embodied point of reference to machine point of reference
3. Human engagement with IML models is a constant in-medium iterative process	3.1. Affected by model output visualisation (positive or negative effect) 3.2. People do quick iterations and trial and error while engaging with the system 3.3. Immersive Model Steering can be negatively affected by media breaks
4. Structure benefits Embodied Thinking in Immersive Interactive Supervised Learning	4.1. People create structure when there is none present
5. IML Visual Scripting as part of a game engine facilitates IML processes and ML mental modelling	5.1. Modularity wasn't obvious to users 5.2. Challenges in separating system responsibilities and diagnosing problems
6. The user evaluation of Expressive IML Outputs show a tension between accuracy, visual flicker and embodied experiential qualities	6.1. Embodied design qualities considered when evaluating IML

Finally, results were discussed with co-authors during over a month of iterative writing and regular meetings every two weeks. The discussion identified a common theme among several sub-themes relating to *embodied thinking* [229] and the separation of the tension between embodied thinking and supervised learning requirements from the embodied thinking theme. Furthermore, the discussion also resulted in a further splitting of sub-themes to better communicate the findings. Consequently, we added two additional themes and reworked the sub-themes structure, resulting in a total of 6 themes and 10 sub-themes. The themes and sub-themes can be found in table 13.

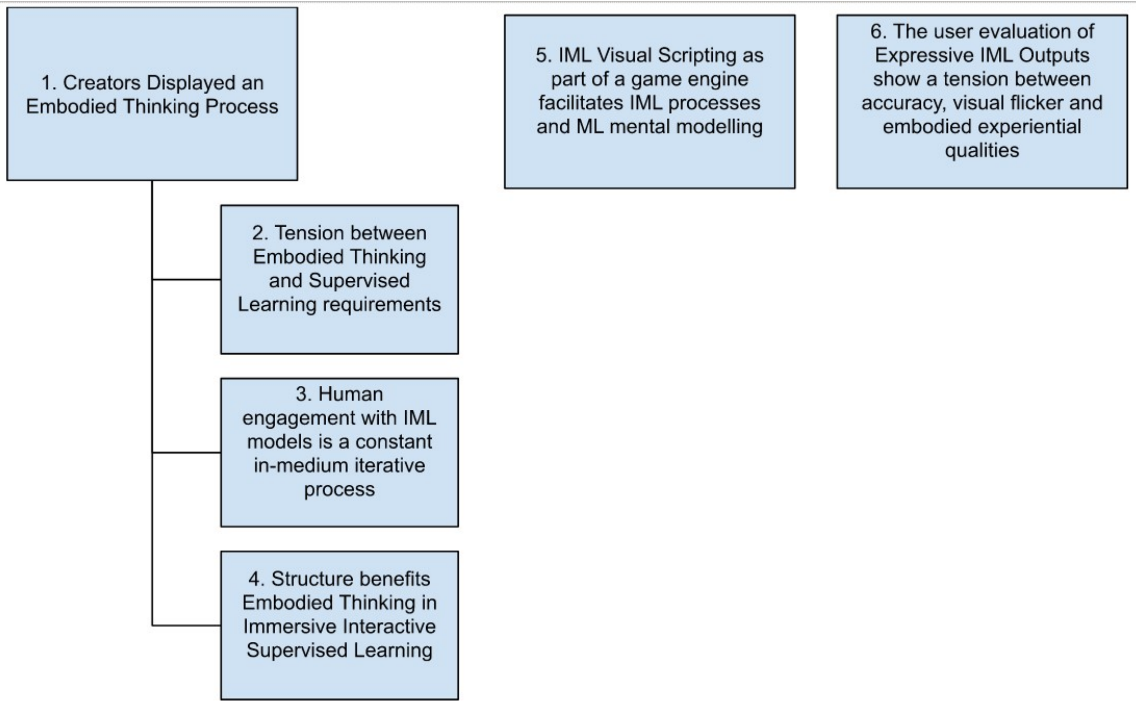

1. Creators Displayed an Embodied Thinking Process
2. Tension between Embodied Thinking and Supervised Learning requirements
3. Human engagement with IML models is a constant in-medium iterative process
4. Structure benefits Embodied Thinking in Immersive Interactive Supervised Learning
5. IML Visual Scripting as part of a game engine facilitates IML processes and ML mental modelling
6. The user evaluation of Expressive IML Outputs show a tension between accuracy, visual flicker and embodied experiential qualities

Figure 104: Relation Diagram between Themes

5.3 Results

We proceed to present and define the five main themes and their corresponding sub-themes, with detailed descriptors and data instances where relevant.

5.3.1 Creators Displayed an Embodied Thinking Process

When interacting with the tool, both in and outside of VR, creators used their own bodies to construct their mental models about the interactive loop with the tool and the elements involved. With mental model, we here refer to the users' internal representation of how our system works, both from a software and machine learning perspective, based on Staggers and Norcio (1993) [277]. In short, creators interacted with InteractML in a form of *embodied thinking*[229].

Creators consistently used non-symbolic physical gestures and cues – when engaging in individual silent problem solving, speaking with each other, and while individually interacting with the computer. This indicates bodily anchored reasoning about the IML system state and its internal interpretation of movement data. We saw examples of embodied thinking in the embodied 'dialogue' that participants had with their systems as they were bodily recording data and training the models. To explore the decision boundaries of their trained model – at what point it categorised a hand movement as 'fast' or 'slow', say – creators engaged in an iterative, two-way process: they taught the machine what movements they wanted it to learn, and the machine learning system 'taught' creators what gestures it did actually learn. Once there was a mismatch between expectations and results, participants observably repeated movements with slight variations to bodily explore how the model understood their movements. This physical exploration was notably not just there for eliciting more observational data of the system's decision boundary. For even when the model wasn't running, participants would still silently gesture and 'play through' different movements as they were attempting to diagnose what was happening with their own movements and bodily understand how the classifier interpreted them.

Participant R_1lh described this process as a literal embodied dialogue: "I need to know what kind of movement will confuse the machine. So the machine will not tell me directly, so I have to test two different kinds of categories... And then the machine tells me 'oh you're right on that. Oh, you are not right'. So I noticed how the machine teaches me". To further illustrate her quote, the embodiment of the dialogue is constant, as the participant need to move in order to interpret the answer from the model. R_2bW similarly stated "you end up learning what that version of the machine... what likes and dislikes in the movement, what kind of boundaries it has overlapping or whatnot. So you can kind of accommodate that". Both participants considered that they were 'saying' things to the classifier through their interactions (selecting features, recording data, directly evaluating it) and 'listening' to what the classifier 'said' via bodily and visually interpreting its output. We interpreted the embodiment of the dialogue as an indicator of embodied thinking, because participants needed ground their cognitive processes on their body usage to understand what their ML models were 'telling' them.

For example, participant R_2bW created a reactive virtual dance floor in which the lights of the scenario would react to the way in which he moved. To that end, he trained a classifier that would discriminate between a 'hands up' pose, a 'one hand up, one hand right' pose, and a 'one hand up, one hand left' pose. Once he recorded data for each pose, he proceeded to directly evaluate the classifier and performed all the movements, but realised that for the area in between poses (i.e. moving one of the hands up after placing it on the side), the model didn't recognise his movements as expected. He then kept directly evaluating to assess where the movements didn't overlap as he wanted to. Before recording additional data, he performed the movements without running the classifier, observably reflecting on where he wanted the movements to overlap. He then recorded additional data and directly evaluated the classifier again to 'listen' to what the classifier 'liked' or 'disliked'. He engaged in additional iterations where he reflected on what the classifier 'answered' to what he bodily 'said' to help define the boundaries in between movements to his preference. We interpreted such reflective moments with clear body movements as instances of embodied thinking, where the participant grounded their reflective thinking process in their body and movement interpretation of the classifier mental model.

5.3.2 Tension between Embodied Thinking and Supervised Learning requirements

We observed a tension between participants' embodied thinking and the supervised learning requirements that InteractML posed. Participants ideated movement interactions mixing gestures and words, describing what they were doing when they performed a movement. A movement could be anything from a pose to a sequence of gestures, and many times participants would understand such a spectrum as a single movement.

However, in order for the classifier in InteractML to recognise the full movement, participants needed to break down their idea into a dataset made of pairs of human-recorded movement data with a label. For a start, this data structuring didn't match the creators' existing embodied mental models about how to describe a movement. This is because participants would express their movements as whole actions, but many times they are formed of distinct *sequences of actions*, even if small sequences. For instance, when discussing ideas with their colleagues, participants would describe a walking forward motion simply as "walking", whereas there is a structured movement pattern to identify when someone is walking versus running (i.e. first one foot forward, next opposite foot forward, but not too fast nor with the feet too much apart from each other). Therefore, to structure a dataset for a walking motion, the participant would need to break down the motion into distinct identifiable sub-components that a classifier could discriminate, including movement examples of walking and running with certain features selected. And that is unlike their spontaneous intuition of such an ingrained motion such as walking.

Furthermore, participants reported difficulties remembering what each label meant after using the system for a few hours, even though they created the labels and recorded their own movement for each label. Participants resorted to the use of on-screen note nodes where they would keep track of what each label was meant to mean (Fig. 105), but even then, they reported difficulties remembering the nuances of their own movement, which illustrates the tension between how they understood the interaction in a first-person embodied way and how the supervised learning classifier would treat it from a third-person symbolic perspective. We interpreted this finding beyond strategic cognitive offloading [237, 193], because we observed how participants did not use note nodes from the beginning, and resorted to them as they iteratively defined the structure of their movements during model steering. Participants would begin constructing a dataset that represented their spontaneous embodied understanding of a movement instead of the more symbolic structured sequence that a classifier would need to discriminate better. For instance, they would begin defining walking with examples of only walking, or defining archery as just the arrow shooting movement, but as they iterated on their models they realised how their dataset required way more structure than what they anticipated. Additionally, once they constructed a better dataset "for the machine", we could observe how they forgot the nuance that they wanted to express on a particular step of the sequence. To illustrate this point, participant R_1Ih tried to bodily record a

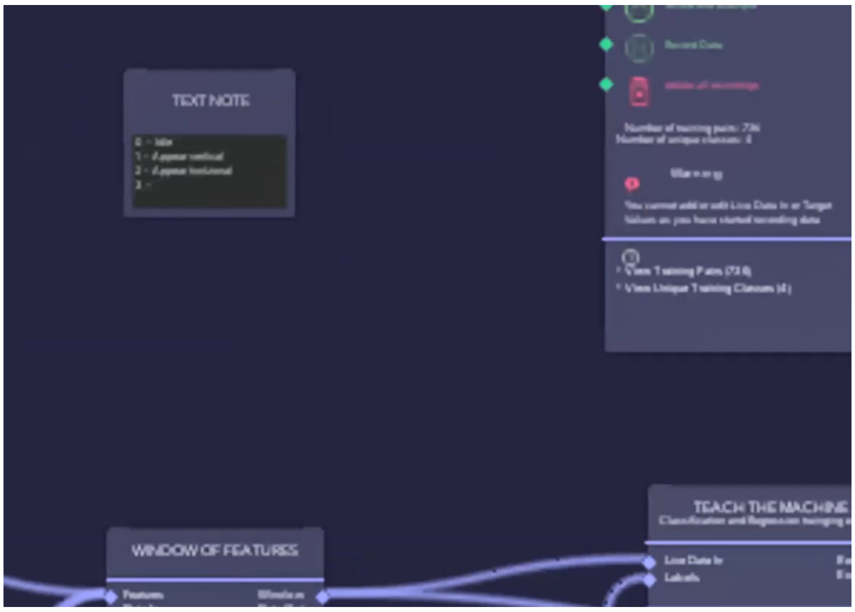


Figure 105: - The participant placed a text note early on close to her training dataset node explaining what each label meant. However, she didn't update the note as she recorded more cases with more labels, and she changed the movement behind each label to improve the recognition of her classifier.

dataset for magical archery, in which loading the bow with an arrow is performed in a magical way by separating both hands at a certain height and the participant wrote a note stating "1 - load arrow" –meaining that label "1" is paired with the "load arrow" motion. However, as model steering iterations went on, she wanted to updated her dataset, but she found it challenging to recall what the nuance was in each motion just from the "load arrow" written description. She explained how challenging it was during the interview:

> "R_1Ih: Sorry, in this part I think one thing is very, uhm, useful. Like, if I, uhm... If I easily put like uhm... if my label is one, and then like... what, what [does] it mean? Like, the label meaning. I think it is very useful for me, so I actually put a note at first to say that label one is that, and label two is that, and label three is that. But the node is not very, uhm... I mean, I haven't really updated it, because the one they're training and I put five inside, and if I want to change it [the note] to another one, I have to type a lot of things. So I think if, if I will do it a second time... I don't know, I have to improve this part. For myself thinking as well. Like how I do that, how I train it, right. How I teach the machine and what I changed in it. And like... the different labels. Also, also, I have to let myself to remember which label is what. In case I forget in my script here to just to call my function. [Laughter]
>
> Researcher: Right.
>
> R_1Ih: Yeah, yeah. So... and I also need to know that if my label two is not, uhm... my label two is not my movement, but what it is? It is not

> only my movement, but maybe I have three not my movement to one, this one is here [Gesture backwards] and the other one is here [Gesture forwards] and I have to distinguish between them. So, I think I need a better note. [Laughter]"

In the previous quote, participant R_1Ih explained how it wasn't enough to describe the motion with only text, as she later on struggled to exactly recall the nuance of the each motion during model steering, and found challenging to remember how exactly she broke down the motion into different sub-stages. We interpreted this finding beyond cognitive offloading, because participants were engaged in active embodied thinking during iterations, but the structural requirements of a supervised learning dataset difficulted their work as they needed to always find a "correct" way to describe –or re-describe –a movement as they were improving their model with data. Therefore, we interpreted such challenges as indicators of a tension between participant's embodied thinking process and the supervised learning requirements of InteractML.

Moreover, the workflow friction that participants experimented included additional dimensions. Firstly, creators found unintuitive providing varied human examples in a way that would benefit the supervised learning classifier. This was because it was obvious to them how to recognise a movement as a human from one example, whereas the classifier could require repeated examples with time segmentations. Participant R_3Gw explained how she might have "started recording a little earlier or a little too late". See the full quote below:

> "R_3Gw: I could have perfected the movement of the hand. Because, as I mentioned earlier, you know it would, at the end of the day, kind of hail the queued animation and the queued text when I was even halfway. Whereas, when I was fully above my head, for some reason, it would stop even though that's the position I put in. Uhm, so either, either I recorded it, started recording it a little earlier or a little too late."

Hence, in the previous quote the participant described how challenging it was to correctly segment her movements to provide variations to the classifier she was training. We interpreted this instance as an example of the unintuitiveness of the segmentation requirement of a supervised learning classifier.

Additionally, creators found it challenging to record distinct enough examples to get good recognition results, because they found it difficult to understand why the supervised learning model struggled to distinguish between two movements that are obviously different to the human eye but yield false positives during inference time. The following quote from participant R_W1X illustrates this phenomena:

> "Researcher: Was your rationale based on: are they detected well? Or, was there something else?

> R_W1X: It was, are they detect well? And also, are they detected differently? Are they distinct enough that it doesn't light up both lights? Because it thinks that gestures, also that one, so it had to be distinct and also detect accurately. So I think that was, uhm, the main four... When I was trying to get four at the end that would be detected differently. But, uhm, yeah could it all easily be done and detected well."

In the previous quote, the participant built a classifier to detect four different gestures that needed to be performed at a random order to open a locked box. He found it challenging to ensure that his gestures are "detected differently" and are "distinct enough" for them not to trigger false positives. This is a challenge for their spontaneous embodied thinking, as there exists a tension in how the participant bodily discriminates movement, and how the movement is discriminated by the classifier. Such tension in distinctiveness poses problems for the perceived classifier accuracy, as he pointed out when he said that his movement "had to be distinct and also detect accurately". Hence, it was difficult to bodily distinguish when the motion of two or more cases overlapped resulting in performance degradation with no intuitive embodied difference to users.

Furthermore, participants found it challenging to structure a labelled dataset because of not being sure when to stop adding additional cases to their dataset to help discriminate every other case (i.e. the participant wants to detect when the letter O is done in midair, but not when the number cero is gestured in midair, or half of an eight is gestured in midair. Participant R_W1X explains it in the following quote:

> "R_W1X: Just because there was no way of like, uhm, in machine learning it's very hard to say, these are the gestures I want, and now I'm going to perform and train it with every other possible gesture. Because that's obviously like an infinite number. And yeah, it's very hard to train that default case of like, not matching one of the gestures I wanted. You sort of, I think, that's probably where machine learning isn't great. You need an alternative to do that."

In here, the participant ended up understanding that the supervised learning algorithm required a clear and large set of training examples to discriminate "every other possible gesture" from the one they were training, that is the "default case of like, not matching one of the gestures I wanted". However, that wouldn't need to be the case if the feature selection is done in a way that can help reduce the amount of cases to feed the "default" class. For instance, features such as velocity (which includes direction) or the distance between the hand and the head can help discriminate whether the user is performing one motion or a different, without adding "every other possible gesture" to the training dataset. We interpreted this as a tension between the supervised learning requirements of the training dataset or feature selection of a classifier and the embodied discrimination of a user.

Besides, participants found it difficult to recreate consistent behaviour across trained models, because of the impossibility to re-record the same movement twice as there is a natural occurring variation in repetitive human body movement to which the classifier is sensitive and which isn't bodily obvious to humans. For instance, participant R_3Gw explains it in the quote below:

> "R_3Gw: I guess, kind of matching the, uhm, the positions for both the machines. Because essentially, if you hail, hail a taxi and then the text also has to appear, it has to recognize the same kind of positions. Uhm. I think that was probably the most challenging thing, just kind of match, match the position of the hand from beginning to end. But since again, it was pretty straightforward positions that are quite different from each other."

In the previous quote, the participant attempted to train two different models to recognise the same movement, because she wanted a taxi and a floating 3D text to appear when she "hailed" to the taxi. This by itself is an instance of a misunderstanding of the capibilities of the system, since the same model could have been used to process the same motion. Still, despite of her misunderstanding, her intention was to train two different models to detect the same movement, and she bodily felt that two recorded training sets with the same motion should be produce two models that would detect the same movement. However, she realised how each classifier understood the motion differently to her, hence her embodied cognitive discrimination of how to describe a motion was more generalisable than the decision boundary of each classifier. We interpreted this dissonance as an instance of the tension between participant's embodied thinking and the supervised learning requirements of InteractML, because it isn't bodily obvious to a human that a motion that is clearly discriminated by the creator has so much nuance once processed by a machine learning classifier.

Finally, participants didn't find it obvious how to structure interactions that required a specific sequence of movements, because of the dissonance between their embodied understanding of a movement and the symbolic logic of the supervised learning classifier. Participant R_1Ih explained such sequencing problem with her "invoke a bow" interaction in the following quote:

> "R_1Ih: Uhm, kind of. Because I want to call it to appear [the bow] right, but I also want to make it disappear. So, those two functions actually are a little bit to make those things harder. Because if I only want to make it appear, that's quite easy, so I already implement it. But what yeah, the problem is one I want to do another movement and to make it to disappear. And then, the problem is to go to a second stage. And say, when that happened, and I only can only implement it in the right time. Like, it now only can disappear, but not really appear. So that is like, uhm, lots of times I think the problem can be solved. So this

> should be yeah, but uhm... yeah, I think uhm... If it could be invoked and disappeared. And actually I still have a step 3! [Laughter]"

In the previous quote, the participant was explaining how she thought that making her magical bow appear was "quite easy", but making the bow "appear" and "disappear" was challenging. From our observations, this was because her body expressivity of the gesture "bow appear" and "bow disappear" indicated that she understood both movements as distinct and that the "bow disappear" movement wasn't a priority during her embodied thinking process, but rather the sequence of gestures required to make the full interaction, which involved movements to make the magical bow (1) "appear", (2) "load an arrow", (3) "shot the arrow", and (4) "make the bow disappear". However, she found out that the classifier wouldn't discriminate priorities on gesture recognition, hence as soon as the bow appeared, if she made the motion for the bow to disappear, the classifier would infer the correct motion and indeed made the bow disappear. This was counter-intuitive to her, as she struggled to prioritise the "bow appear" recognition over the "bow disappear", up to the extent where she couldn't even proceed to the next step in the interaction sequence when she explains that "And actually I still have a step 3!". We interpreted this instance as an example of a tension of how creators bodily think about their own interactions, what they consider a priority of recognition at a particular step, and how the classifier doesn't have any form of prior knowledge about when to prioritise one motion over the other for recognition.

Therefore, we considered this theme as the richest of the synthesised themes from our analysis, where participants exhibited a tension between their embodied thinking processes and the supervised learning requirements of the system. From this theme, we consider important to highlight three subthemes delving deeper into prominent dimensions of such tension: (1) how each feature affects model inference, (2) how time affects each feature, and (3) the participants' first-person point of reference and the machine's third-person point of reference during movement.

5.3.2.1 Tension in how each feature affects model inference We observed a tension between how a feature was understood and how it affected the perceived accuracy of the model. The movement features that InteractML offers are known movement qualities for which creators could already have embodied references: position, rotation, velocity, distance between two points. However, it was not obvious for participants which features would make a trained classifier discriminate a given gesture better or worse. For instance, the participant R_3Mc implemented an interaction in which a light would turn on or off based on whether the controller position was 'below waist' ('light off') or 'above head' (for 'light on'). Then she wanted to introduce an additional movement to change the intensity of the light that was more nuanced than just controller position. For this, she trained a third class 'move controller up' for 'light dimmed' that should have a visual performative feel to it, so that the player would feel inside an interactive and responsive light display. To add that performative feel, she thought that the most obvious feature was rotation, since she naturally identified rotation as the feature most ‘obviously’ distinct from position based on her own bodily movement understanding. Yet she struggled to get the classifier to work as she intended, because probably the variation in her positions were bigger than the variations in her rotations, meaning that eventhough she bodily felt her hands rotations as quite pronounced they weren’t as numerically significant as the position differences. She then attempted to add velocity as an additional feature, which made her classifier “good enough”. However, she then wasn’t sure which feature had the most impact on her classifier, and whether she should now remove rotation or whether she could train a better classifier with only rotation and velocity. She attempted to use a window of features attempting different combinations of position, rotation and velocity. She couldn’t verbalise why her classifier performed similarly with or without rotation, but she preferred to have it included as she explained, “to probably improve it [the classifier] I then added the rotation”.

This tension in feature understanding and model inference, is directly related to the overall tension in their embodied thinking and the supervised learning requirements. Participants experimented with features in the virtual world by visually observing the numerical values yielded by each feature both in the VR panel and the desktop node interface. Afterwards, they would solidify their understanding by recording a simple dataset with a feature and two or three labels to observe what effect they have on a model. When their expected mental model of how the feature should affect the classifier didn’t match the classified result, participants would engage in an iterative loop of providing more training movement examples, or deleting and recording new movement examples to understand how each feature affected the classifier inference. Hence, for every feature introduced and worked with, participants found that they needed to overcome their natural embodied understanding of what a ‘position’, ‘rotation’ or movement ‘velocity’ is, and relearn a mental model of how the classifier treats these seemingly intuitive movement qualities. In the following quote, participant R_3Gw explains how she didn’t understand how each feature affected model inference:

> “R_3Gw: You know, at one point, you need to think about it [a feature effect], uhm. If I would do, if I were to do something differently, probably also explore the rotations and velocity a little bit further. Because those are the kind of things, I mean rotation I kind of understood. But the velocity went a little bit over my head I, I... [Gasp] I wish I could say that I understand it.”

In the previous quote, the participant expressed that the most straight-forward features to understand from an embodied perspective were ‘position’ and ‘rotation’, yet she “wished” to understand ‘velocity’ because “it went a little bit over my head”. This was a surprising finding, since we would assume that velocity is a more immediate quality to understand than rotation, especially given the evidence behind spatial rotation processing in psychology [167]. However, this instance and our observations indicated that participants found both features quite immediate to bodily understand, yet not quite as much to ascertain how much they affect their classifier inference as we saw with participant's R_3Mc 'lights on or off' example, where her rotation didn't seem important to discriminate movements and she could construct an argument as to why. We interpreted both of these instances as examples of the tension between how participants bodily felt that a feature should affect motion recognition, and how it symbolically affects the machine learning algorithm.

5.3.2.2 Tension in understanding the particularities of each feature with regards to time Participants understood features in an embodied way, similarly to how they were thinking about their own movement. They reported details into how the features affected their own understanding of the movement in an embodied way and how certain features were easier to understand than others. For instance, the easiest feature to understand was raw position, and the hardest feature to understand was either velocity or the window of features. Participants reported that how difficult they found it to understand different features was due to how they understood their own movements in an embodied manner. The way in which the classifier understands raw position was closer to the way in which participants understood their own body positions while recording data, but still it wasn't the same and it took some time to fully mature. This is because participants understood positions without reflecting on time, but rather on a 'volume space' relative to their body core or field of view, while the classifier processed positions just as points in space. Hence, even-though participants' mental model of the position feature in regards to time better matched the actual functioning of the system, but it wasn't fully matured to understand that there wasn't any time taken into account by the system at first.

On the other hand, velocity or the window of features were more difficult to understand because of them requiring more than one game engine frame of data, thus involving a machine understanding of time. Game engines usually parse time as 'frames per second', something that originates from the visual renderer of the engine generating the virtual scene onto a pixel-format image in a very fast succession, which creates the effect of visual animation on-screen. Most game engines use the renderer update loop to also compute other processes in order to keep rendered visuals and game logic in sync. Our IML system similarly updates each feature information with every rendered frame. A second of human-perceived time thus usually involves 30, 60, or even more frames rendered by the game engine. This machine understanding of time was observed to be very different from how participants naturally understood time in their own movements. We usually observed this tension manifesting throughout all workshops with the four stage process described in the previous sub-theme when participants engaged with a feature and a gesture involving time, either with simpler features (i.e. position or rotation) or with features that intrinsically use more than frame of data (i.e. velocity or window of features). Such friction with a feature involving time can be illustrated with the following quote of participant R_3Gw:

"R_3Gw: Even though, I, you know, mine isn't really static but that's why

> I put it put in the window so that kind of allowed me to use another type of gesture, so it wouldn't just be a pose... ... I wouldn't love to believe the window [of features] affected it [model inference]. To be quite honest, I think I have like a slight understanding of [the window of features] as, as you explained that it, it just gives more time for the movement and it tracks more time and, like per second. Uhm..."

In the previous quote, the participant attempted to use the window of features to enrich her movement "so it wouldn't just be a pose". However, she explained how, while she felt that the window of features "affected it [model inference]", she only had a "slight understanding" and that "it just gives more time for the movement". We considered this an excellent illustration as an example of such tension in understanding how time is computed in each feature. She said that the window of feature gives "more time" compared to the raw position that she was using, yet the raw position doesn't compute any time. The fact that she understood "more time", means that she had an understanding that the raw position computed 'some' time. This is incorrect and shows how her mental model was enough for her to select features and get the classifier "to work", but it isn't matured enough to properly understand how the only way for the position feature to compute time is via a window of features.

5.3.2.3 Tension to tackle the mismatch between natural embodied point of reference to machine point of reference Like the game engine, for any object in a (virtual) scene, InteractML determines all coordinates relative to the centre of the virtual world. Thus, the mathematic representation of all features in InteractML are calculated from a 'third-person perspective': the tracked VR system controllers and headset are themselves represented and tracked as an offset in reference to the fixed three-dimensional coordinate system of the game engine with a central, fixed origin point. Participants in contrast understood, modelled, and referenced their movements from an embodied first-person perspective, in which movement position or velocity were understood in reference to their own body centre, head or hands as the origin point.

Movements in reference to their body core involved moving their whole body around space (i.e. walking, crouching or dancing, see fig. 106.a), movements in reference to their head involved their point of view (i.e. manipulating objects in front or behind themselves, drawing in-air shapes, see fig. 106.b), and movements in reference to their hands involved the palm of their hand or the tip of their fingers as a starting or end point of (i.e. throwing magical rays from the tip of their fingers, see fig. 106.c). These varying, natural embodied first-person reference points clashed with the third-person modeling offered in InteractML.

The feature that offered the least mental friction was raw movement position, as it allowed less friction when implementing movement interactions with a body core point of reference. However, the same mental model that allowed them to successfully implement an initial position-based interaction with a body core point of ref-

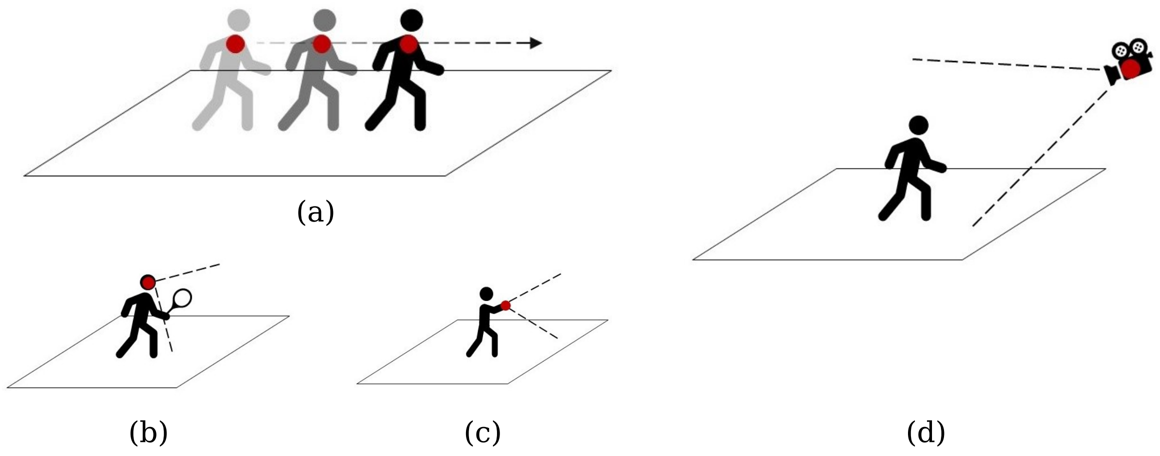


Figure 106: (Left, a-c) Participants showed an spontaneous first-person and diverse embodied point of reference depending on the movement interaction. (Right, d) The IML system was built following the spatial assumptions from the game engine and hence implements a third-person point of reference when processing movement data by default.

erence failed when they attempted to implement an interaction with a first-person head point of reference using the same position feature. To facilitate their mental model maturity, we onboarded participants in the introduction into a feature that we called 'Distance to First Point', in which they could calculate the distance between more than one point in the third-person system perspective, thus translating the system's third-person point of reference into a first-person point of reference when the first point in the feature was their virtual head or hand. However, participants didn't grasp such symbolic representation and complained how the system did not match their body intuitions. Let's take participant's R_2bW answers during the interview to illustrate this point:

> "R_2bW: . . . the triangle you, you'd have to do it in a specific area. And even that, uhm, it wouldn't just recognize the gesture itself, it would recognize all the points and positions in it [the triangle]. So that if you just went into any of those positions, without doing the gesture, it would still switch [the output], uhm, to the corresponding case. Which kind of made it so that I had to make sure that none of the gestures would overlap in their movement or, uhm. . . work. Yeah, more or less that."

In the previous quote, the participant explained how he constructed a mental model of how the classifier understood his motion to draw a triangle. His mental model was correct, as in it correctly modelled the fact that the position by itself, would be understood from a third-person perspective by the computer. However, his problem was that he was trying to perform a movement in relation to his own body from a first-person perspective, and couldn't grasp how the 'Distance to First Point' feature would help solving that issue, and instead he attempted to "make sure that none of the gesture would overlap in their movement". This was a solution that limited the creator movement expressivity and we interpreted it as an instance of the tension existing between participants' first-person point of reference and the computer's third-person point of reference.

5.3.3 Human engagement with IML models is a constant in-medium iterative process, affected by model output visualisation

Participant engagement with the IML system was a constant in-medium iterative process spanning (a) learning about the system, (b) correctly interpreting the system's state, and (c) steering the system into a desired state.

We define learning about the IML system as the process that allows sufficient understanding of the system functionalities for independent work. InteractML had several elements that participants needed to initially learn to perform basic IML model steering iterations. At the beginning of a workshop, participants followed a tutorial led by us that onboarded them to InteractML and taught them: (1) how to drag and drop GameObjects into the graph, (2) how to create nodes to extract features from a GameObject, (3) how to pipe and record movement data from the selected features, (4) how to train and run a classifier, and (5) how to pipe the classifier output into a custom script.

Within the framework of Grossman and colleagues (2009) [105] for software learnability, we focus on the user understanding of the system functionality, and consider system learning as a necessary initial step while engaging with the IML system during the initial learnability category that Grossman and colleagues (2009) define as "Initial performance with the system". However, we consider that learning during the extended learnability category that Grossman and colleagues (2009) define as "change in performance over time" can happen during iterative human engagement with the system, but it is not a necessary first step. Users might have learned everything they about the system, but model steering increases model performance with the system because of the process itself, rather than any human learning. This is because users already learned the 'basics' and they could engage in iterative IML model steering without learning many of the other aspects of the interaction with their classifier. For instance, participants would eventually need to learn how to debug their classifiers, or how to transform their classifier outputs into their desired animations or in-game actions. But none of those would impede their ability to engage with their classifiers.

Interpreting the system's state was the next step in engaging with the IML system once participants understood the system functionalities, and the first step if participants already understood the system functionalities. Participants needed to interpret the system state in order to plan the current steering iteration, and they followed different strategies to do so. For instance, participants could explore the decision boundaries of the classifier via physical direct evaluation to interpret where a movement is being misrecognised. We use the definition provided by Sperrle and colleagues (2021) [276] that defines interpretability as an inductive process in which "a system is interpretable when users can understand why it behaves in a

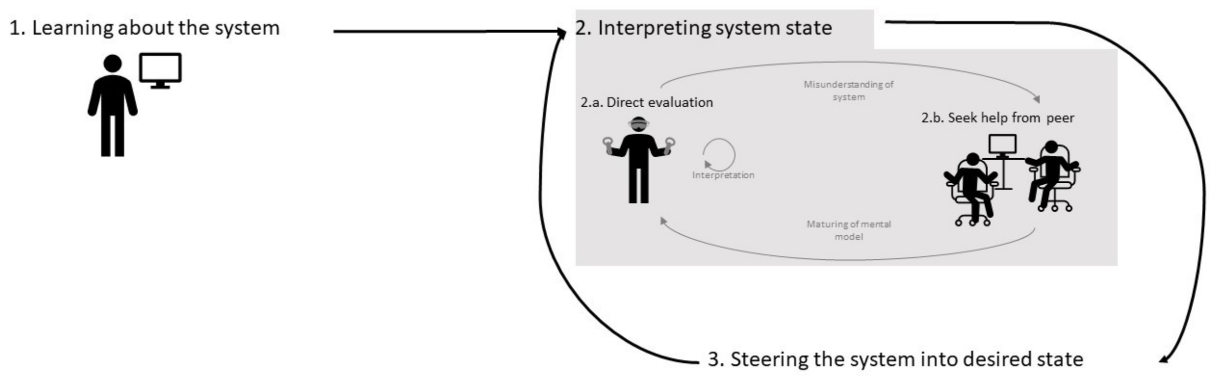

1. Learning about the system
2. Interpreting system state
2.a. Direct evaluation
Misunderstanding of system
2.b. Seek help from peer
Interpretation
Maturing of mental model
3. Steering the system into desired state

Figure 107: Three stage human engagement with IML models diagram

given way under given circumstances". We understand that interpretability refers to the behaviour of the machine learning model, rather than the overall system. This is because in the case of InteractML, the overall system exists in the form of a plugin of a game engine, which introduces several layers of complexity beyond the machine learning behaviour. The goal of participants was to successfully interpret whether their classifiers recognised the movements in their desired way.

Steering the system was the final step in engaging the IML system once participants had interpreted it. Participants could take many different actions to steer their classifiers, such as recording additional training movement data, changing features or improving their custom script to better pipe the classifier output into their interactions. Steering an IML system refers to guiding or adjusting the IML model as it learns from user feedback [63]. Because the user is actively involved in the training process, providing feedback to the IML model and helping to steer it towards better performance requires a previous step of interpreting the system's state. Even if the interpretation is incomplete, users will always require some degree of assessment prior to a steering activity. For instance, participants could be observably struggling to correctly interpret why their classifiers miscategorised their movements when drawing a mid-air shape, but still proceed to record more training data to steer the model in the hope to better interpret the system next iteration. Hence, we saw a close relationship between interpretation and steering, in which participants wanted to correctly steer their classifiers to better interpret them, and wanted to correctly interpret their classifiers to better steer them.

Therefore, we could illustrate engagement loop as a three-stage interaction process (Fig. 107). Let us use feature selection as an example of such process:

1. **Learning about the system:** In the first stage, participants selected a feature for a movement interaction. In this stage, we would visually observe signs of an initial, embodied mental model as the participant exemplified movement data that seemed intuitive for humans to learn from but was likely to yield bad recognition results. For instance, they would record velocity samples from their movement while performing poses or pauses between their movement,

hence training the classifier to admit pauses in its decision boundaries while recognising movement. This would confuse the classifier when discriminating between movement and non-movement (i.e. posing idly in between gesturing), and showed how the embodied mental model of participants didn't discriminate such fine-grained details from their own movement.

2. **Correctly interpreting system state:** In the second stage, participants would directly evaluate their classifier and realised that it misrecognised their movements, showing a flickery behaviour (i.e. not having stability in its output) with very low accuracy or with too many false positives. We observed verbal or nonverbal expressions of confusion, illustrating their mismatch in expectations. A participant could stay in this stage attempting to re-record movement data more than once and still observing how the model misbehaves regardless of what approach they take. Once participants realised that the problem didn't come from a misconfiguration of their system, but rather from a misunderstanding of the system, they would seek help with the intention of maturing their current natural embodied understanding of a feature to better represent the supervised learning classifier. For that goal, they would reach a colleague from which they can observe good classifier results with a similar feature or gesturing goal, or reach directly to the researchers for clarifications. The explanations that we would give would either be a repetition of the explanation given during the tutorial, or a deeper explanation of how the symbolic data of a feature is understood by the classifier and sketches of what is the translation between a gesture and the symbolic nature of a supervised learning feature.

3. **Steering system into desired state:** In the third and final stage, participants managed to better understand how the classifier processed feature data from their training movement dataset and could proceed to independently engage with regular IML iterations. However, if the participant failed to mature their understanding, they would repeatedly engage with stages two until either they managed to proceed to stage three or give up and simplify their feature set or movement interaction to a previously successful implementation.

Additionnally, we found that human engagement with an IML system is (1) affected by model output visualisation, (2) relies on quick iterations and trial and error during engagement, and (3) sensitive to media breaks. We will proceed to unpack each sub-theme below and how they affect all three relevant stages of IML system engagement.

5.3.3.1 System output display IML engagement was affected by the classifier output visualisation. The default visualisation of the classifier output was displayed as the inferred result on either the IML model node in the desktop node interface, or on the 3D panel in the VR interface. The classified result depends on the label that the user chose when recording training data. For instance, if the user recorded a movement dataset in which the movement ‘clapping’ was assigned the label ‘1’ and the movement ‘waving’ was assigned the label ‘2’, the classifier would display ‘1’ or ‘2’ on the desktop node and the VR 3D panel depending on whether the user claps or waves during inference time. In the this scenario, since there isn’t an ‘idle’ state in the training set, the classifier wouldn’t be able to discriminate when the user is not clapping or waving, resulting in a ‘flickery’ display that constantly switches between ‘1’ or ‘2’. Additionally, if the user was translating the inferred label to an in-game visualisation (i.e. casting a spell, summoning a weapon, etc.), their engagement would be more affected by the visualisation since their visual implementation could amplify or hide the classifier state details.

User learning of the system is affected by the IML classifier output visualisation as it helps the user understand the functionalities available in regards to model performance. This is as well important for the user to learn how their existing known metaphors from their prior interaction with the game engine can affect model inference, if at all. This can be illustrated by the following quote from participant R_3Gw:

> “R_3Gw: At the end of the day, I could get so much like only so much from the graph itself by looking at it. By checking the integer, whether it’s changing or not. So, is it even recognizing the changing position? So that allowed me to understand whether it was recorded correctly and whether it knows what I’m doing and knows what it’s [the classifier] supposed to do to make it [the output].”

In the previous quote, the participant explains how “by checking the integer” output she was able to understand (a) if the system was processing data at all (“it is even recognizing the changing position?”, (b) if the system was trained with her latest training data (“whether it was recorded correctly”), (c) if the system could recognise her movements correctly (“whether it knows what I’m doing”), and (d) if her script was translating the integer output correctly into her in-game visualisation (“knows what it’s [the classifier] supposed to do to make it [the output]”).

Furthermore, user interpretation of the system can be highly affected by the IML classifier output since it is one of the key pieces of information that participants would use to inform their interpretation of the model state. For instance, in the example with the recorded training dataset where ‘clapping’ is ‘1’ and ‘waving’ is ‘2’, the flickery display of the output would be a clear indication that the training dataset is lacking an ‘idle’ class or similar. However, this same visualisation can as well be deceiving, since the user might have recorded an ‘idle’ class with its corresponding label, but the problem might lay somewhere else in the configuration space (for instance, the movements might physically overlap, the training data might be noisy, the classifier can’t learn with the selected features, the classifier wasn’t retrained after recording an additional class, etc.). Therefore, interpretation can be highly affected by the IML model output visualisation. An illustration could come from the following quote from participant R_2bW:

> “R_2bW: When I made just the first area, they felt [the features], the features were making it work. And I could see that was working because then everything was acting as it was supposed to. The machine [output] works just fine. Come to the second part of it, yeah, I think the features were working as they’re meant to. I think it was more of a script issue, towards the end of that part I can’t say for certain, right.”

In the previous quote, the participant explained how his system for detecting poses in a dance floor was “working because everything was acting as it was supposed to. The machine [output] works just fine.”. Thanks to being able to visualise the model output in real-time , he managed (a) to conclude that “the features were working as they’re meant to” and, (b) he managed to diagnose that whatever problems he wasn’t satisfied with were due to “a script issue”. Yet, because he didn’t follow a full evaluation run at the end, he claimed that “towards the end of that part I can’t say for certain”, although the real-time output visualisation helped him reach a preliminary diagnosis of the problem.

Because of how strong of an effect the classifier output has on the interpretation of the IML system, it in turn affects how the participant decides to steer the classifier behaviour. Interpretation and steering can be understood as two separate processes (or interpretation can be understood as a sub-process from the overall model steering process), but they are intertwined in that the way the user interprets the model state will inform the way the user intervenes (or not) to steer the IML model into a desired state. For instance, participant R_3Gw explained her experience in problem interpretation and steering intervention via observing the “integer changing” that led her to interpret that the model was working as expected, but instead the configuration of the particle system “need to spawn quicker or they need more time to spawn”. See the full quote below:

> “R_3Gw: I’d say so because sometimes, uhm, even working on the wind project, you could see the integer changing as you’re testing. And so, you can see that it is recognizing, but for example the particle system isn’t uhm spawning. It’s not initiating, so you still understand. You see that the machine has learned, it has recognized your movements, your gestures. So you see that the problem is not in how you recorded it, rather that the particles need to spawn quicker or they need more time to spawn or whatsoever. It allows you, I guess, to like have an extent like an angle on the problem-solving situation and understand that it’s not just that it could be anything else in the scene”

5.3.3.2 People do quick iterations and trial and error while engaging with the system We observed that participant engagement with the system was iterative, which is something in line with previous IML literature [276]. The fact that our IML system was presented as a node-based game engine plugin with a VR interface and that participant interaction was predominantly embodied didn’t change this finding. We understand iterations as a full loop of engaging with the IML system (i.e. interpreting and steering the classifier until a satisfactory state) with a window of time of no longer than a few minutes. Participants performed quick iterations, from seconds to less than five minutes, while they were engaged in the core model steering process. That is, participants already learned about the system, successfully interpreted its state and were in the process of adjusting the system configuration.

Quick iterations were not only desired, but they allowed for more trial and error which in turn helped with learning. When the mental model of the participant wasn't correct, we could see that performing seamless, comfortable and quick iterations where they could try out different features or data configurations benefitted their explorative learning about the functionalities and boundaries of the IML system.

Similarly, quick iterations and trial and error benefitted system interpretation, as participants could visualise what the classifier inferred under different movements or configurations, and allowed them to form a more accurate interpretation of the IML model inner state and its effect on their scene.

And, as above mentioned, quick iterations facilitated the core model steering process because of their positive effect on learning and interpreting. We additionally observed that their steering iterations were predominantly embodied, since participants directly evaluated their classifier while moving in VR. Participant R_1Ih verbalises how in her opinion that model steering "need some time ... to see whether it is good or not [the classifier]" with "several motions", where she explicitly mentions that "we have that kind of design thinking". See the full quote below:

> "Researcher: So could you tell me what was challenging about this process of testing and running [the classifier]?
>
> R_1Ih: Ah, testing and running. I think, the challenges like it need, it need some time. Like, because we will have to, we'll have to, train states. And to see whether it is good or not [the classifier]. And even we have several motions, we have that kind of design thinking. It still needs time to see whether it works well, or not to. . . our accuracies are good or not, as well..."

5.3.3.3 Immersive Model Steering is preferred to be done constantly in-medium, which makes it sensitive to media breaks Since participants steered predominantly their IML systems in an embodied manner, it made the steering process vulnerable to media breaks. We understand media breaks as interruptions in users' flow in their current medium where they are forced to swap to a different one, hence 'breaking' their medium usage [139].

If participants were bodily steering their classifier while in VR, but something in their node configuration required them to switch back to the desktop node interface, they would report how frustrating or tedious switching back and forth between the two mediums was. Participants verbalised that, depending on their style of working, they preferred doing most of their IML steering work in either VR or regular desktop. The reasoning behind was diverse, with most participants preferring to maintain the VR embodied engagement with the system as much as possible since “that is what the players will interact with” or they feel that “they have better control over what the model learns”. To comfortably steer their classifiers, participants wanted to visualise the model output in the VR 3D panel while wearing the headset and gazing directly at the 3D panel. They would want to ensure a clear line of sight with the 3D panel and they could even construct their virtual scene around it. Additionally, they would place their in-game controlled visualisation also in front of the panel, so they could observe both at the same time. For instance, if a tree is meant to grow when the classifier outputs label ‘1’ after recognising a ‘waving’ gesture, users would try and place their tree in front of the 3D panel to clearly read what the classifier output is and, at the same time, observe if the tree indeed grows.

If participants were unable to fully engage with the IML system in VR (i.e. including the ability to recording data, change labels, train the classifier and enter or leave the inference state), they would complain of how tedious was to swap between the two media and how that made their creations difficult and even obstructed the correct interpretation of the classifier state and system problem diagnosis. These limitations were prominent in the first two workshops as the VR interface was just limited to visualising the model output in VR and the functionality to control when to record data in VR and change the label was either not functional or prone to errors. We bug-fixed the problems to ensure that in the later two workshops the in-VR functionality worked without issues (i.e. recording data, changing labels, visualising model output and structured class-by-class testing of the model). Still, re-selecting features, separating the training datasets into different nodes, or piping the model output to a custom script still required participants to use the desktop node-interface.

However, some participants preferred to limit their work in VR to just the model steering and engage with the classifier as much as possible while on the desktop interface. To avoid bodily evaluating their classifiers in VR, they would ask another workshop colleague to perform the testing movements for them while they look at the node graph and virtual scene from the game engine editor window. Or they would perform the testing movements with the VR system, but not wear the headset and rather hold the headset on one of their hands, shoulder or even table to still be able to look at the screen while using the controllers. When asked about their reason to avoid engaging with the system in VR, they explained that it was tedious to get into and out of VR, or that they couldn’t look at the rest of the game engine editor while steering the model. This might indicate that they relied on already learned interaction metaphors from the game engine that didn’t translate into the embodied and immersive IML model steering process.

5.3.4 Structure benefits Embodied Thinking in Immersive Interactive Supervised Learning

Once the participants' mental models were better matured to represent the supervised learning requirements, we observed that they benefited from structure in their iterative use of InteractML. What we mean with structure is the organisation and arrangement of the various components that make up part of a system. This can include the logical flow of data and control, the hierarchy of objects and assets, and the relationships and dependencies between different parts of the users implementations. In the visual scripting interface, structure refers to the organisation of nodes that perform specific tasks and the connections between them, hence defining the flow of data and control through the IML system. In the game engine editor, structure refers to the organisation of game objects, assets, and code. The structure in the game engine project determines how the different components of the game are organised and how they interact with each other to implement the desired movement interaction. Similarly, there is also structure present in the visual scripting interface of InteractML, especially in the explicit testing procedure that participants are asked to follow when directly evaluating their classifiers.

We found that all these forms of structure helped with the embodied thinking process of participants, but each differently. For instance, participants reported that they found the explicit class-by-class testing procedure during the direct evaluation stage useful as it allowed them to have a better feel of their trained classifiers and their training data in an embodied way. We believe this is because of the structured nature of the task in which participants are required to go class by class bodily exemplifying movements that are meant to be classified in that class. Participant R_3Gw explains how with the "testing and running" he had enough time "to go up and down, up and down and track the position that it works best at". See the full quote below:

> "R_3Gw: Yeah, it was after, you know, after recording the movements. Then we transferred it to test and run. To teach and then test and run and with the testing and running that's kind of when I started first noticing OK, OK so it's kind of not, not fully the position. And then, after we've already tested. And then ran both, both models ,that's when I had like enough time to go up and down, up and down and track the position that it works best at."

Additionally, the structured data flow in the visual scripting interface of InteractML was reported to help participants in their embodied thinking. Nodes are placed and connected left-to-right following the same structuring process of a supervised learning task, in which first the data is loaded, then features are selected from the dataset and then the model is trained on the selected features from the dataset. InteractML's visual scripting flow emulates this principles in the visual flow of nodes, where features are selected on the left and the model is trained on the right. Participants reported that the mixture of real-time visualisations on the nodes and their positioning helped during their thinking process, and we observed how they were gesturing while silently thinking during their working process.

5.3.4.1 People create structure when there is none present Furthermore, as participants engaged with the system iteratively, they created structure to aid their embodied thinking process. Firstly, as there wasn't an existing way of linking numerical labels to an human-readable format, participants placed notes close to each classifier node to clarify what each label did.

Secondly, participants visually structured their datasets into groups of nodes according to relevant labels or movements. For instance, participant 'R_3P5 whose movement interaction relied on non–verbal communication with a virtual character, structured his dataset into different nodes with each corresponding to a different recognisable gesture towards the virtual character. He also structured his idle class (i.e. all the movements that shouldn't trigger any response from the virtual character) into separate nodes, but each with its own movement dataset (Fig. 108). He explained during the interview that this additional structure allowed him to simplify his working process as he could select which movements to focus on, and could erase or augment the dataset of certain movements without affecting the rest. He explained how he realised the dataset was "controllable", where he works with "a label called zero and I train everything on zero. And then I do another [node] database that I only do training on the one, and I do everything on there. And then I feed them all into one learning machine and machine learning output. And yeah, so by learning how to use the multiple database [nodes] actually is very helpful.". See the full quote below:

> "R_3P5: Also, I know the system can actually add on top. To add any more details, the two [machine learning] systems have any one [training] data. Data, so you can have your [training] database. So I realised this is controllable. Because if I'm doing other design and other than this [Gestures], I probably can do only a [training] data. For example, a label called zero and I train everything on zero. And then I do another [node] database that I only do training on the one, and I do everything on there. And then I feed them all into one learning machine and machine learning output. And yeah, so by learning how to use the multiple database [nodes] actually is very helpful."

Thirdly, participants created additional structure in the game engine editor by implementing feature visualisations that would match their mental models of their own body movements. When asked about this during the interviews, participants explained that it helped them to better understand how the classifier understood their movement, and to better understand how their movement was being processed by their selected features. The visualisation consisted of a coloured trail created as they moved their virtual hands in the scene (Fig. 109). The coloured trail would fade after the same amount of time as they understood their selected

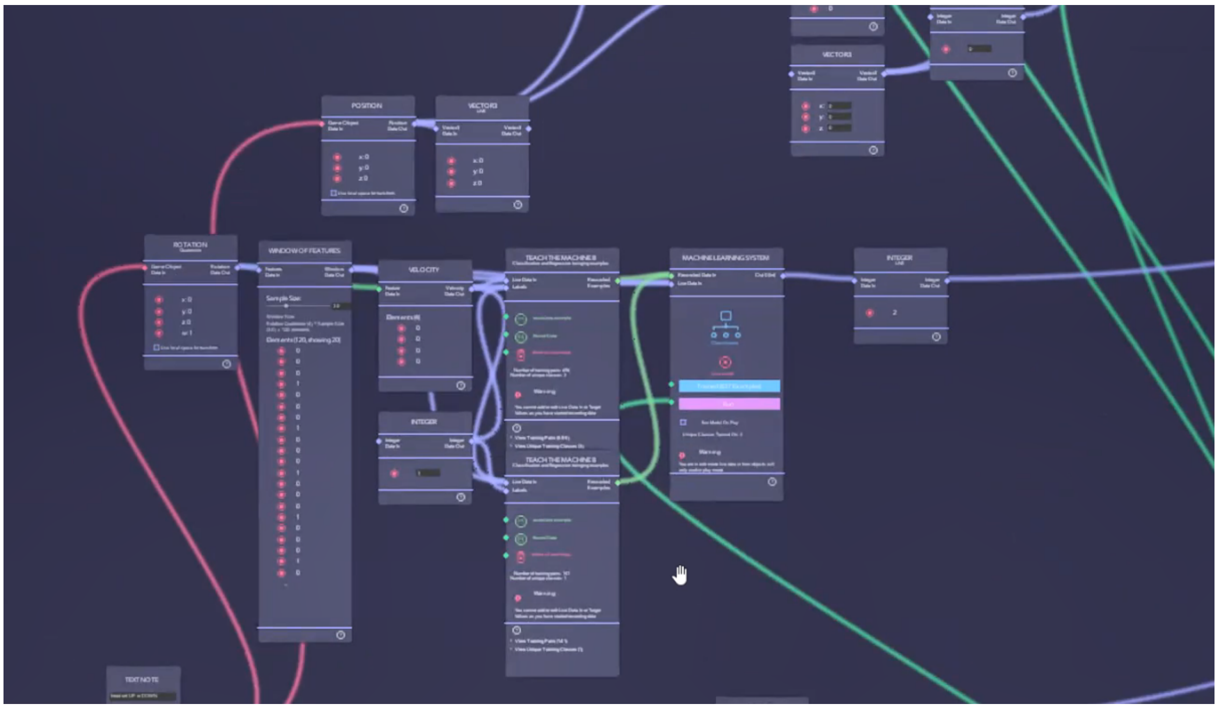


Figure 108: In this screenshot from participant R_3P5's graph, he used two different training dataset nodes, each with its own label, to better structure his steering process.

windows of features would "look" at their movement. Additionally, participants would also create virtual elements that would help them structure their embodied interaction in the virtual scene, such as a virtual lectern with inscriptions of the gestures to perform at real-scale (Fig. 110). This particular virtual element was intended to help the creator record movement data with a consistent scale, something he believed was relevant for the accuracy of his classifier.

5.3.5 IML Visual Scripting as part of a game engine facilitates IML processes and ML mental modelling

Apart from the structure participants could bring into their node graphs, participants found advantages when using InteractML's visual scripting as part of the game engine editor set of tools. Participants reported that the real-time visualisation of each of the nodes helped them in their learning and interpretation of the data flow into the classifier. They explained that visualising how their movements were computed in real-time by their selected features helped them understand how each feature worked, and how the classifier could understand their movement. The real-time visualisation on each of the nodes helped see whether all data is flowing correctly to the classifier, or whether they forgot to connect some nodes together in their graph. Participant R_2bW illustrates the advantages of visually seeing "a changing value" to understand if "there is something wrong with the node". See the full quote below:

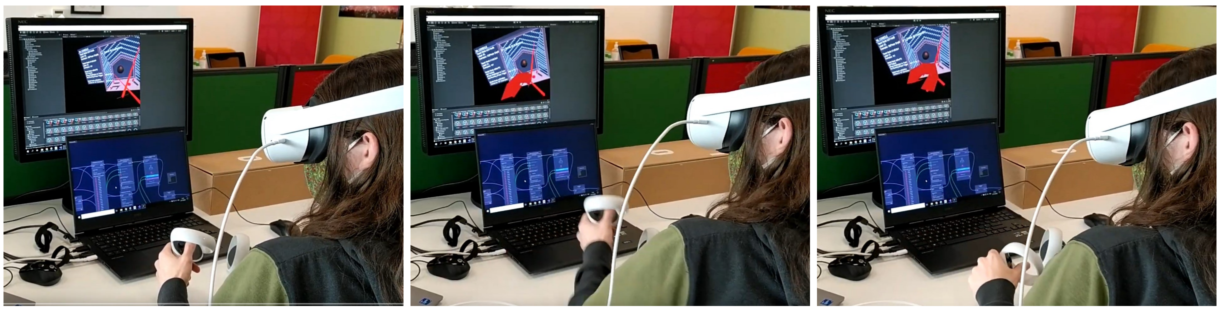


Figure 109: A participant records additional movement data for his movement interactions. He implemented a red trail to visualise in real-time what the window of features 'sees' from his movement. In the pictures the trail can be seen 'growing' as he makes the gesture. In the end the trail will disappear matching the sample size of the window of features.

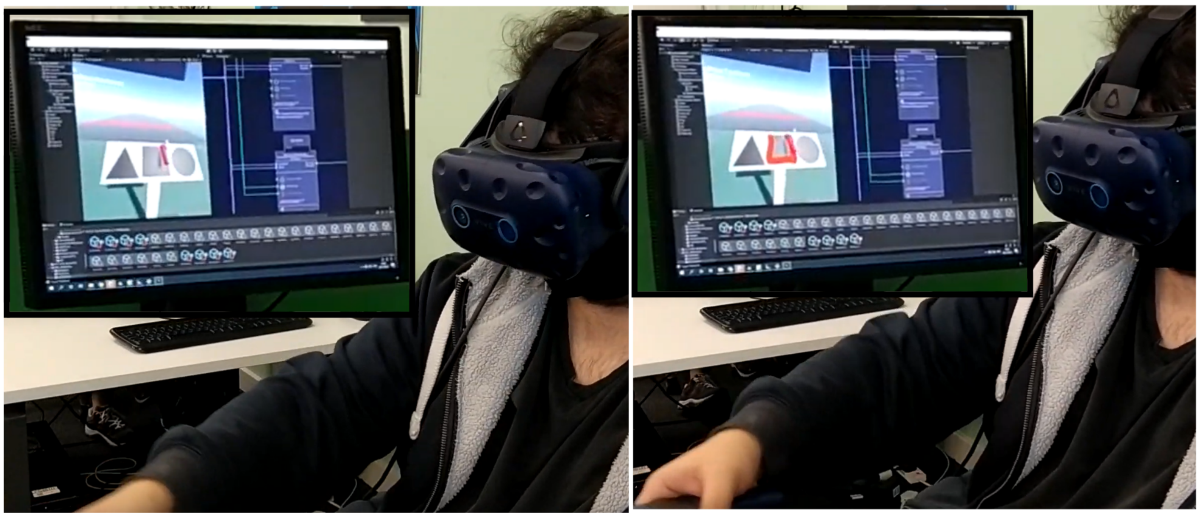


Figure 110: A participant created a virtual lectern with the shapes he was recording in an attempt to maintain consistency in the way he moved. He described how he thought the reason why his model wasn't performing well enough was because of the variations in his own mid-air movement. He structured his recording process by creating and bodily following the contours of the lectern shapes.

> “R_2bW: it’s pretty easy to see for position, rotation or any node with a changing value, velocity any of those. If the value is low, if it changes or, it means either they’re not connected right or there is something wrong with the node. See, you can kind of figure that one out quite easily”

Furthermore, participants reported that creating, connecting or disconnecting certain nodes helped them with diagnosing problems with their graph because they could better visually understand which part of the graph wasn’t working. This modularity and flexibility of visual scripting helped participants with their supervised learning workflow as well, since they could make one classifier run in real-time, while another classifier is off, which helped them see if one of the models misbehaves under certain conditions. Moreover, the visual modular flexibility helped participants better interpret their IML systems, since via successful diagnosis they can gain an intuition of the underlying ’black box’ of the classifier. Thus, the visual modular flexibility improved participant system controllability, due to the possibility of triggering conditions visually without writing programming code. Participant R_3P5 illustrates that “there’s a lot of good ways of using the current system” to use“visual coding language systems where you can control the flow within the graph”. See the full quote below:

> “R_3P5: Um, adding one point on, on the using experiences. I find... to understand the system is very important. Especially today we have lots of debugging and creating small functions. And I think, because I find there’s a lot of good ways of using the current system. For example you can add more layers of library on top of the library, and then you can create subsystems. And you can enhance your previous learn, machine learning data with only one single set, so you can test out, for example, one single node. For example, um [label] zero. I add some more data in it [label zero] and I can try other data. And that can be another function that sometimes controls the other machine Learning System while they’re on and off. So that’s really handy. So I think that’s a really good way [of working]. I’ve never seen this in the other visual, visual coding language systems where you can control the flow within the graph so that’s really, really cool.”

We observed as well that participants visually represented their mental models of how the IML system processed data through iterative node placement. We could observe this by comparing early stages of their graphs during think-alouds to the later version of the graphs. In their early graphs, nodes would be scattered around the graph as they explored the different functionalities of each node. But we observed that participants reflected their understanding of the data flow into the graph as they learned about the system and created more structured graphs. We believe this visual representation of their mental models helped them solidify their interpretation of the system, which in turn allowed them to better steer their classifiers.

Furthermore, participants reported that the IML workflow offered benefits for game development in virtual reality. Participants found that the direct embodied evaluation of their classifiers had benefits compared to other graph plugins because it gave them a sense of "where the error is", whereas other graph plugins don't have this embodied channel of feedback. Additionally, participants reported that they perceived the IML workflow as an efficient way of saving VR development time compared to traditional methods. Because, even when the creator is able to write functions to process movement data, the amount of effort and time required to reach nuanced results is considerable. They explained that this was particularly the case with traditional gesturing solutions. These traditional gesturing solutions rely on simple positional sequences after virtually interacting with invisible 'colliders' that send a signal to a script. These collider-based systems are usually restrictive on what kinds of movements are detected, since it is more straightforward for participants to physically exemplify the movements they want, rather than design and engineer a number of colliders that will need to be hit in a certain sequence. Whereas with IML, participants found that they could save time on writing down cases combining code, feature extraction, and collider placement.

In spite of the fact that visual scripting and the IML workflow offered opportunities, it suffered from a number of limitations that either we observed or participants reported throughout their work with the tool. We will describe in detail each synthesised sub-theme below.

5.3.5.1 Modularity wasn't obvious to users Even though there were benefits from participant usage of InteractML's visual scripting interface, the modular placement of nodes in the graph wasn't obvious to all participants. During the first two workshops we observed that even though some participants intuitively understood that a modular placement of nodes was possible, most participants didn't and they realised after talking to their peers during the workshops. They reported feeling at a disadvantage since they could easily perceive the benefits against the perceived workflow struggles of not being able to bring structure into their graphs. For instance, when asked about data collection, participant R_W1X explained how he "only realised later" that he could have "separate training node", and that in turn "would have made that [data recording] more sensible". See the full quote below:

> "Researcher: Was there anything easy about this process of steering the model to behave as you wanted through data collection?
>
> R_W1X: No, because I think I only realised later that I could have added in the extra, uhm, you know, more... just the fact that I added more data for a particular, uhm, feature by having separate, separate training nodes. And I think, I think I only realised that yesterday, so... I probably would have done that earlier. Which would have made that [data recording] more sensible. But I didn't. Yeah, I didn't realise that earlier on that you could have multiple [recording nodes]."

5.3.5.2 Challenges in separating system responsibilities and diagnosing problems In regards to system debugging, participants found it challenging to balance the responsibilities of each workflow layer. A complete prototype of a movement interaction with InteractML has three layers: (1) the IML processing of movement data, (2) the game script translating the classifier output to an in-game state, and (3) the visual representation of the game state. For instance, one of the participants created an interaction in which quickly moving her arms forward would make a magical bow appear in her hands. This particular movement interaction was divided into (1) a classifier processing the data from a window of features with 30 samples per data entry, with the distance between the position of the left and right hand to the head, the velocity of the position of the left hand, and the rotation of the left hand; (2) a game script receiving the output of the classifier and translating the outputs ‘0’ to ‘idle’ logic and ‘1’ to ‘make bow appear’ logic; and (3) the 3D model of the bow together with a particle system to beautify its appearance.

Participants struggled with balancing what was the responsibility of the (1) classifier processing the data and (2) the game script translating outputs into game states. For the previous example, participant R_3Gw experienced flickering issues with her classifier outputting ‘0’ or ‘1’, since her idle class wasn’t stable enough to correctly discriminate when the bow had already appeared. Once this problem has been diagnosed, there could be several interventions to reduce the classifier flicker. One option would be to improve the stability of the idle class, either by editing the training data or the selection of features, to better discriminate when the user has already summoned the bow in her hands. A different intervention could be done on the game script side, where a written programming function could ignore the classifier flicker and only read the output once before the bow appeared. The participant in question chose the latter approach, and smoothed out the flicker in her script.

However, this became a challenge as the classifier incrementally had more outputs than ‘idle’ or ‘bow appear’, and also accounted for the outputs ‘bow disappear’ and ‘bow fire’. Her script became increasingly difficult to manage since it was smoothing out classifier output flicker and, at the same time, it managed the progression of states in her interaction, activating or deactivating 3D models and particle systems accordingly. In the following quote, she explains how she could see the her classifier was reacting “either in an odd way or in a correct way”, which indicates output flicker. She was unsure “how much can I actually implement” and if she should record more data to fix it or “go look into my code and see if something is wrong there”. We interpreted this as an instance of how challenging it was to discern if the flicker problem should be fixed on the data side, or the game script side. See the full quote below:

> “R_3Gw: Because at some point, even when I would put down my hand it would show one. So, or like 101010 and change. So since I saw that it was reacting either in an odd way or in a correct way, that affected it. Okay, so how much can I actually implement and how much I can actually

> do with it, or should I record it or, should I, you know, or should I go look into my code and see if something is wrong there. Because everything else is responding [except my code]."

Participant R_3P5 experiencing a similar issue with his interaction and decided to tackle the classifier output flicker by rigorously modularising his training dataset making each class more stable to reduce the classifier output flicker. However, as his interaction grew in labels, it became tedious to change features and record again training movement data. He resorted to a feature selection "trick", in which he made the classifier to recognise all movements above his head height and below his waist height to be recognised as 'idle', hence simplifying his classifier decision boundaries. As he explained it, he "just take out the x and z [axis] to act on that" and that "once I know how the system works, I can basically use tricks to trick the system" or "a clever way to train the system, to make the system work much better". See the full quote below:

> "R_3P5: yeah definitely, for example, in the beginning that, the head up and down. I just take out the x and z [axis] to act on that. And there are lots of things. I think, once I know how the system works, I can basically use tricks to trick the system. Well, well, the better [term]... let's put another word, like using a better way to train the system. Or even a clever way to train the system, to make the system work much better."

These two examples illustrate the common challenge among all participants of balancing the responsibilities between the classifier and the game script. Each of the above-mentioned implementations has its advantages and disadvantages, but show a fundamental challenge in deciding how to prioritise responsibilities and where to spend development effort in the IML system workflow. Spending more effort on the game script side can be challenging for game creators without much programming experience, but could simplify working with the training dataset. On the other hand, spending effort on the classifier side to fine-tune the training dataset or features can require a rigorous structure and workflow when adding data or labels to the classifier, but simplifies code complexity in the game script.

Following on the participants' effort in data recording, participants reported that they found it tedious to record data again when diagnosing a problem with their classifiers. If there was a problem with the data or with the features, there could be a lot of tedious trial and error to change a feature, number of samples in the window of features, or movements recorded in a specific way to see what works. This problem grew in acuteness the more cases the classifier needed to recognise. Participant R_2bW complains about it in the following quote:

> "R_2bW: Uhm, in the grand scheme of things, that wouldn't be so much of a problem. It would be uhm... not an annoyance, but tedious to have to record things a lot."

Similarly to how there existed a challenge balancing layers (1) and (2) of InteractML's workflow, it also existed a challenge establishing the relationship between layers (2) the game script translating the classifier output to an in-game state, and (3) the visual representation of the game state. Participants reported that it was challenging working with how the game script processed game states every frame, and how animations would play while moving in VR, especially when animations are timed. Because of this, participants reported how they found it challenging to correctly establish this relationship to convey the look and feel of the interactions they initially ideated. Participant R_1Ih explained how she realised that her script calls "update all the time", and how her animation "go through the whole animation and it will start again and again". She expressed that she believed the issue with her animation to be a "coding problem, but not really a machine learning problem. Because, even when I don't use machine learning, ... you have to, uhm, put data into the update ... when the participant wants to invoke the bow". See the full quote below where she explains how challenging she found translating the classifier output into the "invoke the bow" interaction:

> R_1Ih: I think the feel is a bit difficult. Because, uhm, I think people. . . how to say... like the VR environment gives people an immersive experience right, but people still like what can easily be recognised a bit differently. Like if it's delayed, right? And so I feel like all the positions are not really good. When I do the movement and... but, it does not really show it and I have to do another, a second time. So I think many things are very influenced by the result. But actually, I think it is okay, to be honest. Yeah because , in the end of it, the main problem is not reaching that kind of experience. It's because [Laughter] Because my script it's just the update all the time, using the machine learning [system]. The condition is if, uhm, the machine learning label equals one, and they will invoke the bow to do some interaction with the animation. But for the animation, my way, my way is like to use the delta time. So it's actually, I think it is limited, like you can only call it once to do that, from start to the end of the animation. But when you always update it and call it, it will just stop in the middle of it. It [the bow] won't really go through the whole animation and it will start again and again. I think that is kind of a coding problem, but not really a machine learning problem. Because, even when I don't use machine learning, sometimes our function, you have to, uhm, put data into the update to make like, no matter when the participant wants to invoke the bow. And you need to let the computer know now you should be able to do that, right. So, I think that kind of problem needs to be solved. But yeah, what I say is not the fault of the machine learning system."

Because of the challenges balancing responsibilities in each of the IML workflow layers, participants reported how debugging and diagnosing problems with their classifiers was a multidimensional task. In each of the layers of the IML workflow, there are several sublayers where a problem could lay, and we observed different strategies that participants followed to decide in which layer the problem was. For instance, participants reported that the real-time classifier output label helped diagnosing whether there was a problem with the graph configuration or the game script code, since they could observe if the inferred output updated to the correct value but their in-game effect didn't update the state accordingly. Therefore, participants reported that "scene things" were usually a problem with their game scripts, but problems with the inferred value itself would lead to a graph configuration problem. Participant R_2bW explain how his configuration allowed for "a very simple way of checking if it works or not" because if his 3D model "didn't turn red, then it meant there was something wrong with the script. If it didn't turn green it meant that there was something wrong with the connection between the script or the machine. Or it was the machine. So that kind of helped to narrow it down". See the full quote below:

> "R_2bW: So I would, uhm.... I would set up in the script when it's changing its case. It's just a very simple way of checking if it works or not. So, in the first area, it was very simple to see if the capsules would turn red or if they would turn green. If it didn't turn red, then it meant there was something wrong with the script. If it didn't turn green it meant that there was something wrong with the connection between the script or the machine. Or it was the machine. So that kind of helped to narrow it down."

Participants also followed regular debugging strategies making extensive use of the game engine editor console to diagnose problems. However, diagnosing problems with the IML graph configuration was reported to be challenging, since it was difficult for participants to correctly interpret the dropdown of recorded training data. Participants reported that, because of the dropdown only having numbers, they couldn't understand what they meant and most of the time ignored any step of data exploration in their diagnosis process. Instead, participants relied on gaining an embodied intuition of where the problem might be, and performed debugging iterations trying different feature or data combinations with a small subset of their classes. We observed participants following a simple-to-complex feature selection strategy when diagnosing problems with their selected features. This strategy relied on a simplification of their feature selection and an incremental addition of features to diagnose if a feature, or combination of features, is causing the problem. Additionally, we observed participants usually following a sequential movement data recording strategy, from their idle class to each of the cases they are diagnosing, instead of re-recording only one of the cases. We believe this debugging strategy allowed participants to gain a better embodied ownership of their training dataset during the diagnosis process. In regards such embodied ownership of data, the participant R_3Gw explained how "you are the one doing the gestures" which are "like your emotions", and points out how "You are physically working". See the full quote below:

> "R_3Gw: Uhm you, you are the one doing the gestures. And even if it's, again, like this something that you know that it at the end of the day will work best for you. But, still could have been incorporated for other people as well depending on what kind of gestures you're doing. Because you're the one developing you're the one setting it. It's like your emotions. So you physically worked on the project as well, which is kind of like what sometimes lacks when you're doing digital work. You are physically working but you're like, you know, there's only a limit of physical activity that you do when you're working on something in Unity."

Finally, the last challenge that participants reported and we observed was how different the thinking requirements of a supervised learning workflow are to those of the rest of the game engine. The supervised learning thinking requirements refers to (a) how to strategize feature selection, (b) how to structure the data and the labels, and (c) how to diagnose problems when the classifier misrecognised gestures. The supervised learning way of thinking is required to correctly use InteractML as part of the engine, but many other game engine system metaphors don't translate well into InteractML. Thus, effectively training classifiers requires a different skillset from regular game development that can be time consuming to obtain without prior explanations. Even if iterative use of the tool teaches about machine learning, we observed how some participants struggled with the supervised learning way of thinking and requested instructor clarifications during their working process often. Participant R_1lh stresses "how you design with machine learning thinking" is "the interesting and the difficult part" because "if no one teaches me, it is really difficult to design anything". See the full quote below:

> "R_1lh: I think the interesting and the difficult part are the same. It's like how you design with machine learning thinking. And well for this time, I think because we have these workshops, and the basic learning of the machine learning uh I mean, the logic of the machine learning. How you train it to do the categories is very useful for me really, to make the next thing for my interaction. But if no one teaches me, it is really difficult to design anything."

5.3.6 The user evaluation of Expressive IML Outputs show a tension between accuracy, visual flicker and embodied experiential qualities

We found that the most important qualities during the evaluation stage were classifier accuracy, classifier output flicker, and the different embodied experiential qualities pursued by creators. Classifier accuracy refers to the correctness of the recognition of a movement with the label it was trained for [276], whereas classifier output flicker can be defined as the temporal instability of the recognition of a movement for the duration of such movement. The different movement experiential qualities mentioned by participants were enjoyment, naturalness, presence, smoothness, responsiveness, fluidity, expressiveness and self-consciousness. They referred mostly to positive movement experiences, except for self-consciousness which was understood as the excessive self-awareness of one's movements while performing "silly movements". Participant R_W1X explains how some of the gestures he tried "weren't fine or felt like they could be self-conscious" in the following quote:

> "R_W1X: Yes, yeah yeah I was trying different gestures. Because some clearly, think, weren't fine or felt like they could be self-conscious, you know. If people are. . . [Pause] So there are certain ones I tried earlier on, and either eliminated because they didn't work well with the machine learning, or they felt like they could be too difficult. Or, or they could cause self consciousness, or something like that. So I did, I did go through a range of different gestures and then swapped them around a bit."

Participants reported intended player enjoyment was one of the most important qualities mentioned by participants, and they used the word "fun" to describe enjoyable interactions. They noted that making their interactions "fun" was, many times, one of their primary goals. Participants also mentioned that enjoyment was related to other embodied experiential qualities such as naturalness and expressiveness. Naturalness refers to how closely the IML system mimics intuitive human movement. Expressiveness refers to how well the IML system captures and conveys the user's intended movement and it was important because it made them feel like they were able to fully convey a feeling through their movements.

Presence refers to how users really felt that they were "there" performing their interaction in the virtual environment. Participants noted that presence was important because it was one of the core qualities in virtual reality, and a desired experiential achievement. They also noted that presence could be enhanced by incorporating visual or auditory cues that help users feel like they are "really there, doing the movement". Smoothness and fluidity seemed to be used interchangeably, and refer to how well different movements flow together in response to IML system inputs, with no sudden or jerky changes. Participants noted that smoothness and fluidity were important because it made them feel like they were performing a natural sequence of movements rather than a series of disjointed movements. Responsiveness refers to how quickly and accurately their classifiers react to user inputs. Participants noted that responsiveness was important because it made them feel like they had greater control over their movements and reduced frustration when trying to perform complex movements.

One of the insights we gained from the study was how there existed a tension in the IML model steering strategy of participants, where they tried to maximise their classifiers accuracy while minimising their classifiers output flicker. Participants would firstly steer the classifier to reach a minimum level of accuracy, which is what many reported as “to get it working”. They relied on observing the numerical output of the classifier and a simple sphere changing colour in this stage. They performed embodied direct evaluations on their classifiers to interpret their perceived level of accuracy. Even if the numerical output or the colour of the sphere would flicker, they seemed to accept a certain level of flicker at this stage. However, once they piped the classifier output into their own 3D object or effect, we observed how the same classifier output flicker would cause the participant to perceive a lower level of accuracy. This is because the output flicker was magnified by not allowing full animation sequence to play correctly, and was a cause for tension. Hence, we believe that this tension between classifier accuracy and classifier output flicker shifted how accurate their classifier actually was, and participants followed to perform further iterations to steer their IML implementation into a state that would yield more temporal stability. Participant R_3Gw explained how she identified visual flicker during model steering:

> “R_3Gw: If it was changing quickly, you know. Or changing in the wrong position. So, if I was in, you know, my hand is up but it’s showing 0, for example. Or if I’m doing something else and it’s doing 010101 and actually calling in the animation.”

To reduce visual flicker, participants resorted to different interventions: (1) adding more movement training data, (2) including a window of features in their feature selection and experimenting with the size of the window, (3) breaking down a big classifier into simpler ones recognising less cases each, or (4) write programming functions in their game script. Regardless of the intervention, participants reported how, by making their classifier more temporally stable, they could affect how it positively felt to bodily interact with their creation.

For instance, even though participants reported feeling more satisfied with their interaction after reducing the visual flicker, they would pursue to make their interactions “more expressive”. This is because a movement could be perceived as accurate by participants (i.e. it is recognised when the user does the gesture), but at the same time inexpressive (i.e. it doesn’t convey the intended feel). Hence, creators want to make movements that are well recognised, but also expressive so that the movement conveys a certain feeling when performed (i.e. a taxi being stopped feels satisfactory, but the movement the classifier accurately recognised was very uncomfortable). Participant R_3Gw explained how she didn’t felt satisfied with the interaction she got working to stop a taxi in the following quote:

> “R_3Gw: I don’t really feel satisfied with how it [the movement interaction] feels just because it, like, it doesn’t give you the satisfaction of taxi stopping. So it’s not really satisfying to keep raising [the hand] also because for the taxi I have to really raise my hand. And how I wanted it [the movement interaction] to be, which is about how it works really nice. But

> for the taxi it's kind of like an in-between awkward, so I wouldn't say that it was very satisfactory. Nor would I say it is like a game-like feeling."

Another instance to exemplify the tension between perceived accuracy and positive experiential movement movement qualities came from participant R_2bW explaining how making their classifier "too accurate" with certain features restricted "movement fluidity and realism". This is because certain features can make a model infer false positives. For example, the distance from the hand to the shoulder is computed the same if a player moved their hand forward, or overstretched their hand backwards, even for a frame. See below his explanation with the distance feature and movement fluidity:

> "R_2bW: This kind of [workflow], uhm... although it expands your ability to control certain movements, and what with the fluidity of movements and do something with that. It also hinders what exact movements you can do. Because if you're reaching for something with interactML and you've got those, uhm, the machine hasn't like saved my shoulder too outstretched far. That one movement will trigger something which kind of limits what you can do. In the distance from my palm to my shoulder you can't really do anything there. But without it you can't just put the point as the hand at the end and you can do individual points between and they can actually do different things. But it's all situational based, I would say. So you may want something between the points but that would also stop you from having that fluidity and realism. But... so it's good and bad for either one."

5.3.6.1 Embodied design qualities considered when evaluating IML Because of the tension getting the movement interactions to deliver certain experiential qualities, we considered it important to understand what qualities participants pursued as they engaged with the IML system.

Firstly, we found that participants thought that movement recognition needed to make sense thematically. Participants described how there needed to be a meaningful use of a movement to interact with a virtual object, otherwise they couldn't see a benefit of using a gestural recognition approach. This is something that was described as well as an ideation challenge, since participants needed to find a use case where ML-driven movement recognition would make sense compared to a traditional recognition approach. Participant R_W1X framed it in term of gestures with objects and gestures with features, and how "pushing a button" could "have been hard coded a lot easier". See the full quote below:

> "R_W1X: And even at the very beginning, when I was saying what objects am I gonna interact with. It is very much tied in with, well, what gestures would you interact with this object and what gestures with other

> features. ... So yeah like an interaction like just pushing a button on the alarm, which I think was too simple. That could have been hard coded a lot easier."

Secondly, participants reported that they pursued a certain "magicality" with their movement interactions. Realism wasn't necessarily pursued since participants wanted to control visual effects that weren't possible in real life, such as fire appearing as a bow is summoned or an exaggerated stop of a taxi. In this regard, we observed a relationship between the interaction 3D graphics and its embodied game feel. Participants reported that without the animated visual layer over their interaction they found it difficult to convey the desired level of "magicality". This finding is inline with Swink's (2009) theory of *game feel* [288], where visual feedback in traditional gamepad videogames is similarly essential for good game feel. Participant R_W1X explained how his interaction visuals "really tied into the fun of the thing" in the following quote:

> "R_W1X: I think the screen script came quite early on. And it possibly did come get modified to sort of give more visual, uhm, changes in the world. And also things like that setback, the time delay before say a light went off and the next one went on that got buried. Because that felt like it really tied into the fun of the thing. And also, it was too easy if, you know, if the lights stayed on for like three seconds that's made it ridiculously easy. So I gradually reduced it to sort of one second. And that felt still possibly a bit easier, it could have even been putting them all together in half a second but, uhm... Yeah one second was about right."

Furthermore, participants showed an interest in their classifier working for a diversity of bodies, and that their movement design choice allowed for the player to bodily learn how to play with an increasing complexity curve. Our finding is reflective of Csikszentmihalyi's theory of flow, in which challenge and experience can be correlated [51, 133]. Participant R_2bW described the embodied challenge increase in his prototype taking into account "taller and shorter people", where the amount of movement was increased with difficulty, and his training dataset included "different versions of the same movement to allow for that [stature] margin of error". See the full quote below:

> "R_2bW: So for the first area there were three basic poses that didn't require much movement. I would, uhm, train the machine with three different classes. And I would move for each class I would do a little bit of movement just in the general area. So you're not having to get the exact positioning of the controllers correct. Because that would be a massive pain for anyone to try to get the exact same position as someone else. Plus, you never know with taller and shorter people if they can

> reach the same area. So that's how I did that bit. And then the second area was a lot more, the movements, I guess, I had to make several, I'd have to do the same movement a few times back and forth in varying degrees of distance from each other. I guess that's how you'd want it. So that is because I was going to do the same movement slightly differently, as well. Even though it's because everyone has a different stature. So I needed to make sure that I recorded a bunch of different movements or a bunch of different versions of the same movement to allow for that [stature] margin of error."

Finally, participants reported feelings of "satisfaction" when developing with IML because of how well they could control their creations and an embodied ownership of the created artefact. We found it highly interesting how participants described that, in a way, the player would be physically interacting with the creator since the classifier is trained and tuned thanks to the movement data of the creator. This suggests that making an interaction feeling stiff or not fluid can raise the creator's feelings of self-awareness as of how players would judge the way the creator moves. This can be illustrated with participant R_2bW explaining how "If people enjoy the action you've done, you can take that as 'oh i did that myself', rather than the computer" because "there's some aspect of you [in the action]". See the full quote below:

> "R_2bW: I like the way it is, but you have that added factor of you can show people that this is probably something you've done yourself that makes you kind of get a feel for the person as well. And it kind of, it helps instil a sense of self-confidence in yourself as well. If people enjoy the action you've done, you can take that as 'oh i did that myself', rather than the computer. That's all, not necessarily how you've set the graph. There's, there's some aspect of you [in the action], which can be a real accomplishment for certain people."

5.4 Discussion

5.4.1 Implications and Future Work

Participants were able to create classifiers without being experts in machine learning. This shows that IML, as implemented in InteractML, can be used to support the embodied design and thinking processes that participants exhibited. This is in part due to the embodied nature of the recording and evaluation process, in which the participants physically move to record data and to evaluate their classifiers, and proved advantageous compared to traditional methods using colliders or rule-based approximations. Still, the same embodied process that enables participants to better prototype motion controls also brings new barriers to their implementation, because of the tension between the very embodied thinking displayed by participants and the inner workings of interactive supervised learning.

Prior motion control design guidelines didn't cover such challenges, and instead focused on broad recommendations for design such as "Use head orientation to control user's gaze and the body's orientation to control steering"[205], "Create interaction paradigms that adapt to individual differences in player range of motion" [89], or "Design multiple gestures for one event, when needed" [263]. It is true, however, that certain motion design guidelines from the literature can be applicable when working with gestures, because even while being broad recommendations, tackled some of our results. For instance, Norton and colleagues (2010) [205] guideline "Full body gestures may have cross interference therefore care should be taken in assigning functions" matches with the problem that participants encountered of designing distinct enough movements to be recognised. From the game industry perspective, Jack (2011) [129] mentioned as one of his guidelines that "Computers don't interpret people, they interpret points", which matches with our finding regarding the dissonance between the human and the computer point of reference. Still, none of the existing design guidelines tackle continous movement and IML-drive development fall short for the challenges arising from designing with an interactive supervised learning. Therefore, we could synthesise a set of IML-driven motion control design guidelines from our results that tackle broad recommendations for motion control game creation, inline with previous literature:

- **Make use as much as possible of creator movement, even during system configuration.** This guideline makes use of one of our core findings, that creators grounded their thinking process in their body. Therefore, any part of the design loop that doesn't support such embodied thinking capabilities risks hindering the tacit creative practice of embodied thinking.

- **Facilitate feature experimentation via reducing iteration cost.** Since features can be difficult to understand, we found that the more participants iterated with the system, the more they managed to mature their mental models. It is difficult to create the same environment for "peer discussion" that we had during our workshops, and something that participants made use of extensively during their working process. What we can recommend is to facilitate iterations as much as possible.

- **Configure features to compute from a first-person perspective relative to the creator's field of view.** Since we found that creators bodily thinking was grounded in their body, so was their thinking perspective. In computer science, usually features are computed from a third-person perspective, but this introduced unnecessary friction that could have been solved by computing all features from a first-person perspective.

- **Data and features can be manipulated fully in-medium and as flexibly as possible.** Because model steering was performed constantly in-medium and was sensitive to media-breaks, we suggest that the entire design process should be undertaken fully in-medium, which includes data and feature manipulation. Additionally, designers should be able to modify their dataset fully in-medium, adding examples or removing certain examples if they wish to.

- **Modularise the training dataset as much as possible.** One of the strategies that participants found useful was the modularisation of the training dataset into configurable sets. Such a modular approach reduces the iteration cost, because the designer only needs to add or remove movement examples to a particular class, or to a particular movement.

- **Start with a distinct set of movements, tackle nuance as a challenge in itself** Even though our embodied IML process was meant to support more nuanced movements that traditional gestural recognisers, all the challenges that participants faced made it clear that it would have been a better approach to 'start simple' with less nuance, and iteratively add nuance once a base functional recogniser is in place. This guideline is meant to reduce friction throughout the iterative process.

- **Prioritise visual flicker from the beginning.** Since visual flicker was one of the most common challenges participants faced, prioritising visual stability should be a focus from the beginning of the design process. Iteratively adding new cases after each case is stable can be a good strategy.

- **Place the focus on movement, not objects.** Most existing VR interaction toolkits are completely object-focused, and it is difficult to make a better job than them with an IML system like InteractML. Instead, the strength of our results is on the diversity of movement that was recognised, which is something that VR interaction toolkits [301, 83] or gestural recognisers [306, 232] can't tackle appropriately. Therefore, the focus with our approach should be on movement exclusively, and use other specialised tools for other tasks.

Participants also expressed a desire to use the tool more after the workshops and they benefited from the visual scripting interface, indicating that they found it useful and valuable for their work. Also, from our observations, the full integration of InteractML into the game engine facilitated game creation processes as expected [217], because participants were able to stay in-medium when working on other engine parts and reuse known interaction metaphors from the game engine editor. This suggests that IML could potentially be used in regular game development workflows to facilitate the creation of motion-controlled video games. Yet, to properly address industry workflows, further research with workflow integration in game studios should be undertaken.

Looking towards the future, there are many exciting use cases for InteractML and its methodology. For example, InteractML could be used to expand any of the ideas participants implemented, fine-tuning their features and data on longer development periods (certainly longer than one or two days) to improve game feel. This includes improving the reactivity of the environment to player movement, games where players use their bodies to interact with characters, vehicles or objects in the game world, or games where players must solve puzzles using physical poses, gestures or movements. Additionally, InteractML could be integrated with VR SDKs (Software Development Kits) or popular toolkits with already implemented examples. That allows the discoverability of the paradigm when developers download SDKs and read the official documentation, and can enrich the interactivity of VR experiences by making movement recognition a more common-place feature.

Of course, InteractML can have use cases outside of games development. InteractML can be used by researchers in areas where researchers need to record movement data and use ML in a game engine. Some examples can be non-verbal human-agent interaction, novel forms of movement interaction for regular desktop work or virtual instrument creation. Other fields can benefit from simulations built with InteractML as well, like physical rehabilitation, sports or medical training. A merge of any these fields with motion-controlled serious gaming could be possible. For instance, by using IML to create motion-controlled videogames that are tailored to specific rehabilitation needs, patients could engage in fun and interactive exercises that help them recover from injuries or illnesses. This could include games where patients must perform specific movements or gestures to complete tasks or achieve goals.

Furthermore, there is a very exciting opportunity to create interactive experiences where the player contributes movement data to the recorded dataset, thus tailoring the movement experience to particular player needs. This idea about end-user authoring of movement interactions can be extended to any of the previously mentioned fields. For instance, coming back to the example of physical rehabilitation, patients could tweak existing interactions that are found to be physically challenging to perform.

Overall, this study demonstrates the potential of InteractML for supporting embodied design and thinking processes in a variety of contexts. These findings suggest that InteractML has a bright future ahead as a tool for creating engaging and intuitive motion-controlled videogames and interactive experiences.

5.4.2 Implications for IML

One of the biggest challenges arising from our results is how it wasn't straightforward for participants to select the most optimal features for their classifiers. Fiebrink's Wekinator [74] supports feature selection in a similar fashion to InteractML, albeit it offers more challenges as creators are required to write their own feature extractors in the programming language of their choice and pipe the data to Wekinator via a network protocol, while InteractML offers already written movement features in the visual scripting interface. McCallum and Fiebrink (2019, 2020)[177, 178] further investigated an intervention to support feature engineering with Wekinator, where they modified the system interface to display selection options of 200 relevant features synthesised from literature [236]. Because of the amount of features, McCallum and Fiebrink (2019) offered an automated feature selector, together with the option to manually select features in the interface because not all automated feature selection would be well suited for a general IML loop where the dataset is small or the task performed can't be known in advance. Their results were fascinating. On the one hand, participants decided to choose features that were familiar to them, which "most often, these were means and first-order differences" [177]. On the other hand, quantitative accuracy evaluation showed that user selected features performed worse that any of the automated solutions. Yet most participants reported that they completed the task satisfactory, with a third of them reported that not all their gestures had good performance.

In their follow up study in 2020, McCallum and Fiebrink [178] also included semi-structured questionnaires and a more diverse set of participants and tasks to better understand the disparity in the accuracy and self-reports from participants. Their results were inline with those of their prior study, and concluded that participants wanted to select features to select features to realise their designs, because raw data would lead to worse accuracy, but automated features would lead to worse self-reported subjective ratings. Similarly to our results, McCallum and Fiebrink (2020) reported how difficult was for participants to interactively select and evaluate appropriate features sets, sometimes changing their task instead of their features. They discussed that an interface intervention could potentially better structure empirical experimentation with candidate features.

McCallum and Fiebrink's (2019, 2020) results offer a very interesting set of insights to reflect on from our own results. Firstly, we found that our participants found challenging to understand features. Because of the amount of features McCallum and Fiebrink offered, they didn't enter into detail as to *what* was challenging about a certain feature, but rather found that the "means and first-order differences" were the most chosen features. They argue that it might be that participants were "bad" at feature selection, but we speculate that the challenging aspect of more sophisticated features is to think about them in a first-person body experience. Features are, in the end, symbolic representations that help the machine make better sense of the data, not the human. Secondly, McCallum and Fiebrink found a consistent mismatch between classifier accuracy and participant subjective rating in both their studies. Again, they argue that a solution is to help participants to make better decisions from an accuracy perspective. However, what if that is not what participants are after? In our results we also had a focus on *perceived accuracy*, but explored during the interviews the particular experiential goal of participants, where most of them were more worried about visual flicker and enjoyment or immersion than perfect accuracy. Which might mean that maybe accuracy is a poor quantifiable dimension of movement to maximise. What about movement features that might degrade accuracy but increase temporal stability, thus reducing visual flicker? What about feature sets that degrade accuracy but increase enjoyment? The literature on movement experiential qualities introduced in the background chapter in subsection 2.4 tackles a diversity of constructs (i.e. engagement, immersion, game feel, enjoyment, presence, embodiment) that is not possible to easily quantify, except for the use of questionnaires or physiological sensors that are still in early research stages or are not fit for the quick iterative nature of model steering. Even though our review tried to cover the state of the art, our own results show experiential qualities that we couldn't have predicted. The experiential qualities that surprised us were smoothness, fluidity, responsiveness and self-consciousness; and the embodied design qualities that we didn't expect were thematic meaning of movement, magicality of movement, and diversity of bodies support. We can speculate that, if we had more participants, there was a likelihood that more of such unexpected qualities would arise. Hence, maybe we should ask ourselves: is it even possible to generalise onto one quantifiable metric all of those desired dimensions of movement? Maybe that was the issue that prior work found with accuracy in human IML model evaluation [73, 177, 178], that accuracy falls short to describe such richness and diversity of body experiential expressiveness ingrained in the movement data.

Additionally, McCallum and Fiebrink (2020) [178] suggests to bring more structure to feature selection. Such suggestion is inline with our findings that structure is preferred in an IML process, with participants highlighting the structured class-by-class testing interface to make sense of their classifiers. Also, we found that participants wanted to create structure were none was present, which suggests that indeed feature selection might benefit from a structured process to explore feature combinations. We suggest that future work might tackle labelling feature sets and models with a subjective feeling score, to iterative construct a dataset of not only the pairs of human movement and interaction outputs, but tuples including scores for additional relevant experiential evaluation such as enjoyment, smoothness or magicality. This could be in the form of additional guided visual steps when performing model steering iterations that introduce structure to select feature and help with rich movement evaluations.

Another of our findings relates to the process that participants followed when engaging with the IML loop. We described such process of human engagement with IML models with three stages (1) learning about the system, (2) correctly interpreting system state, (3) steering the system into desired state. Our findings are reminiscent of Katherine Compton's "grokloop" [49], in which she modelled casual creators creative process in four stages: (1) build a hypothesis, (2) modify the model, (3) evaluate the result, and (4) update the model. Compton considers the grokloop as being a way for the user to interact with a generative tool and examine its possibility space, where "the speed of learning depends on how short the loop is"[49]. Lai and colleagues (2020) [152] reflect on the grokloop from a mixed-initiative procedural content generation (MI-PCG) for games perspective. In their work, Lai and colleagues discuss that there exists three game design pillars for MI-PCG which are (1) respect user control, (2) respect the creative process, and (3) respect existing work processes. In their case, the grokloop is reminiscent of the second pillar *respect the creative process*, where Lai and colleagues argue that the creative process for MI-PCG is "a feedback loop of trying something out, seeing the result, making changes, seeing the new result, making further changes, and so forth until the designer is satisfied". Additionally, Lai and colleagues argue that

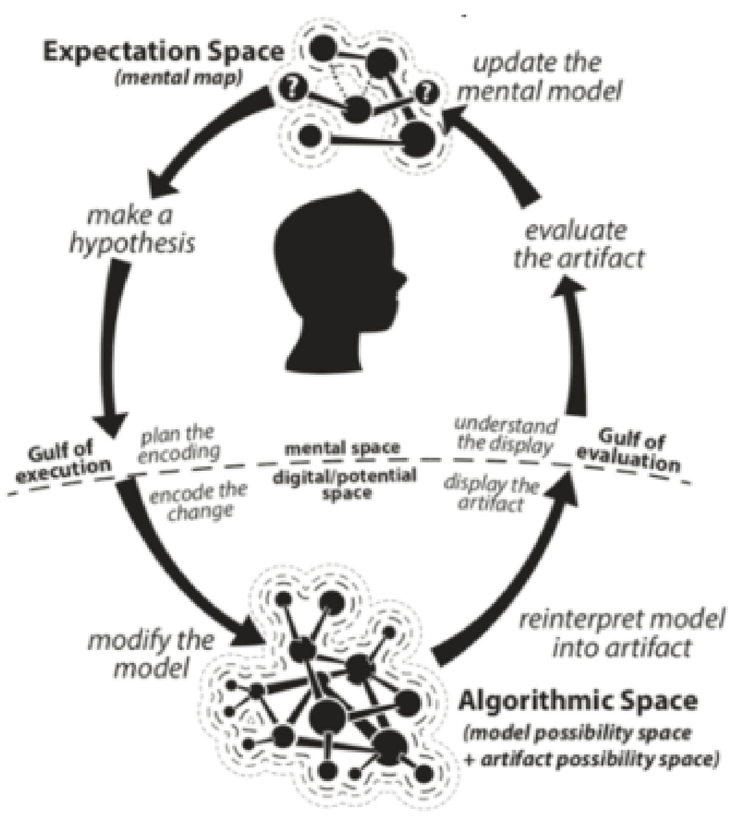


Figure 111: Visualising Kate Compton's grokloop [49]

"To stay focused on the task, it is important that this feedback loop is as short as possible", where the authors reflect on the similarities of Kate Compton's grokloop [49] and how it is transferable to MI-PCG. Furthermore, Lai and colleagues explicitly mention how in IML, the model steering loop is an ML-specific variant of the grokloop that is "in opposition to classic ML algorithms where a high level of technical skill is often required, and training data often needs to be fed to the algorithm for a long time, without an interactive interface to test when the user is happy with the result of the learning" [152]. Our results show that, not only Lai and colleagues' (2020) and Kate Compton's (2019) creative processes are applicable to IML for motion control design, but also that there can be an additional social dimension for creators when they are unable to properly mature their mental model of the IML model and request a discussion with a working peer. We speculate that this might indicate that, apart from the embodied thinking that participants displayed when individually working on their own systems, there might be a social aspect to their embodied thinking in which creators are able to verbally and non-verbally translate their tacit understanding of their mental model to a peer that isn't familiar with their problem. Through verbal and non-verbal explanations, and through bodily interacting with the classifier of the participant in need of help, the peer might be able to bodily and cognitively construct a *better* mental model of the system to help the participant asking for help correctly interpret the system state.

These observations expand current knowledge of embodied forms of design [116]: they show that creators not only design in an embodied manner, but also think in an embodied manner.Our observations also suggest that participants generated embodied knowledge [93] about the system.

Nevertheless, we can't ignore the technological challenges and cognitive frictions described in the results. Firstly, we saw a tension between participants' embodied mental models and the operational logic of supervised learning underpinning InteractML. Participants ideated movement interactions mixing gestures and words, but in order for the classifier in InteractML to recognize the full movement, participants needed to break down their idea into a dataset including pairs of human-recorded movement data with a label. This data structuring requirement didn't always align with participants' embodied thinking. That in turn leads to tensions in how each feature affects model inference, understanding the particularities of each feature with regards to time, and tackling the mismatch between natural embodied point of reference to machine point of reference. In short, these tensions arise from prior non-expert embodied expectations not being met (i.e. the machine will discriminate like me, the features are calculated as I understand them, I can teach the machine as I teach a human).

To address these conflicts, there could be interface interventions. We found that structure did help with some of these tensions, yet graph modularity wasn't obvious. One straight-forward intervention is to better visually guide participants to include more structure by default. This can be done via 'smart' suggestions, similarly to what modern integrated development environments (IDEs) use to help programmers write code [79]. InteractML's node interface could suggest nodes as the user interacts with the graph. For example, InteractML can suggest to automatically create training data nodes as the user records additional labels, and to distribute each label to a training data node by default, thus making graphs more modular and easier to modify, extend and debug. Additionally, InteractML could provide richer data visualisations for both live recording and playback of recorded movement data, similarly to how participants created their own coloured trails to match their window of features. InteractML could display semi-transparent 3D visualisations in VR. Points that are created for positions, rotated snapshots of tracked objects, straight-line distance lines drawn between objects or velocity vector lines. These feature data visualisations could help maturing users' mental models of how features are computed already in the training data recording stage, since participants don't need to wait until evaluating their ML models to begin a "dialogue" with the system. On top of that, InteractML could provide a virtual 'recording&testing' room, with virtual furniture and elements that could help users perform movements more consistently. These virtual elements could be tailored to fit the user's particular movement needs, as we've seen with the virtual lectern in the results section. The advantage is that we are prompting the user to use the virtual space around them in their immersive embodied model steering process.

Additionally, model inference visualisations could help the above mentioned tensions by making the system more transparent. Model inference visualisations can extend training data representations to include the most informative cases for participants. For instance, one interesting example would be with only positional data

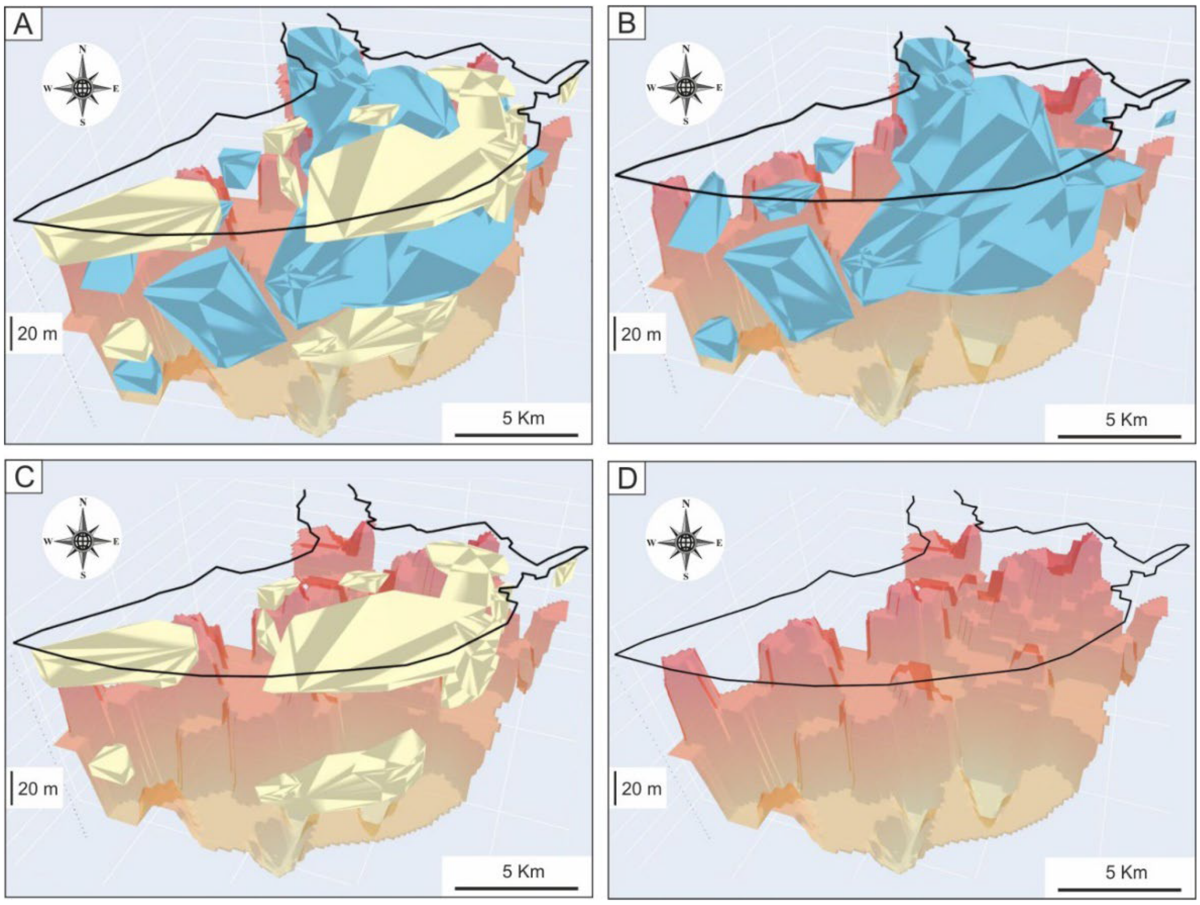


Figure 112: Diagram from Bullejos and colleagues (2022) [38] work on visualising a KNN model trained on four classes representing each a different type of soil. In the 3D visualisation, each diagram removes a different class showing what are the decision boundaries per class in 3D space.

inference, in which the 3D space of the virtual scene gets coloured to construct three-dimensional decision boundaries. The classifier would colour every possible point in the scene, and the user could navigate the visualisation both in editor mode or in VR. This idea is a three-dimensional extension of known kNN 2D visualisations [292, 38]. Another example is with velocity data or distance data inference, in which a generated representation of the VR headset and controllers could perform variations of existing recorded movements and a coloured vector line would be drawn from the tracked objects. Each colour would represent a known label, as in the previous positional data inference visualisation. And especially since with the current InteractML process implementation users are already recording testing cases, therefore we could simply reuse the movements they record to evaluate their models. However, this approach gets more complicated as more features are selected, and would likely require very specific visualisations when several features are selected for model inference.

Furthermore, these model inference data visualisations can, in a way, disrupt the current direct embodied classifier evaluation since they introduce a 'record and playback' loop, instead of a 'record and move-to-test' loop. Studying whether these 'playbacks' of training data and model inference visualisations support or hinder participant's observed embodied thinking process is therefore an open topic for future work.

Still, just improving the transparency of the training data and model inference doesn't address the problems that participants reported with fundamental machine learning requirements of supervised learning algorithms. These algorithms learn from a curated training dataset, and we observed how making any changes to the selected features required participants to record data again. Even without changing the selected features, the current IML methodology can require users to record again when there is a problem with a certain label. A possible solution is to separate movement recording from feature selection, which would allow users to record movements without having to worry about selecting features before. This could take the form of a recording process in which the movement is recorded 1:1, with a playback of the VR headset and controllers played after recording the data. From there, the user could visualise all the features and select the ones that they consider important, which means users have less friction when iteratively experimenting with features. Furthermore, techniques to automate feature selection altogether could be explored. By using additional machine learning algorithms to automatically select features based on the recorded movements, users could save time and effort in the feature selection process [178]. This would also help to ensure that the selected features are relevant and useful for the task at hand. Automating the selection of the window of features can simplify meaningfully, as creators found it challenging to comprehend each feature in regards to time. There exist current automated techniques to automate window sample size selection [293]. It might as well be that granting 'invisibility' to the time segmentation process could further obstruct user's understanding of features in regards to time, since the time element in their movements is never tackled explicitly during interaction. Thus, future research should explore whether an intervention automating window sampling is detrimental to user understanding of features while, paradoxically, allowing them to iterate faster.

Even if future automated interventions simplified the feature selection process, the current supervised learning algorithms can require large amounts of data for exceptionally nuanced and complicated movement interactions. This would still make recording movement data tedious, hence algorithmic solutions for recording less training data could be explored. One interesting solution would be to use data augmentation techniques, which involve generating new training data from existing labelled data. For example, a data augmentator can generate variations of the original movement data by applying transformations to existing movement data by adding noise or distortion, or changing the speed or direction of movement [120, 72]

Another solution would be transfer learning: leveraging pre-trained models that have already been trained on a large dataset of movement data [275]. These pre-trained models could have already learned to extract relevant features and patterns from the movement data, which can be useful to train robust classifiers with less additional data.

Furthermore, one-shot or few-shot learning algorithms could drastically reduce the amount required for training. One-shot learning algorithms can learn to recognize new classes of movement from a single or very few examples, which can reduce the need for large amounts of labelled data [312].

Finally, reinforcement learning (RL) classifiers can be used instead of supervised learning classifiers via demonstration-based learning. Demonstrations are examples of movements that are provided to the RL algorithm by users. The RL algorithm learns to imitate the demonstrations and then further refines its performance by loops of automated sampling and training to maximise a reward function [253].

Less time spent recording movement would, in theory, mean that game creators would have more time to spend on feature selection, model evaluation or any of the many game development tasks. Nonetheless, the algorithmic interventions explained above do require more computational resources to train and likely more inference resources than the k-nearest neighbour (kNN) algorithm employed in this study. They might as well miss the benefits from the strong influence of training data on kNN's decision boundaries that was speculated in previous IML literature [73]. Therefore, future work might investigate how less data-hungry algorithms are qualitatively evaluated by game creators compared to kNN, and what they consider important when evaluating decision boundaries from classifiers.

Overall, these algorithmic solutions have the potential to significantly reduce the amount of data that needs to be recorded in order to create accurate and effective IML models. By continuing to explore these solutions and refine their implementation, we can unlock even more potential for creating engaging and intuitive motion-control videogames and other applications with IML.

5.4.3 Supporting Game Creators with IML

We found that ‘ideal’ engagement with InteractML would be a fast, unbroken IML steering loop, enabled by providing all relevant inputs and outputs directly and fully ‘in-medium’ – in VR or on desktop . Consequently, all the stages in the IML process should be fully supported both in VR and outside of VR, not only partly as currently implemented in InteractML. If users want to engage with the entire IML loop in VR, there should be appropriate spatial metaphors translating each of the desktop options without negatively impacting their functionality. Users should be able to select features and see recordings of their own movements. Similarly, users should be able to inspect and modify the game scene from VR, as well as pipe classifier outputs in their scripts. Obviously, the reason why this hasn’t been implemented is because it is a herculean design research task. What do all IML spatial metaphors need to look like in VR? How do they fit in the overall spatial game development process? And how can users comfortably program scripts in VR?

Narrowing down questions around better supporting users during their scripting efforts, users reported how it was challenging to separate system responsibilities and debug. As explained in the results, a complete prototype of a movement interaction with InteractML has three layers: (1) the IML processing of movement data, (2) the game script translating the classifier output to an in-game state, and (3) the visual representation of the game state. Participants found it challenging to discern which layer was responsible for an issue, and how to diagnose problems with each layer. A possible non-technical intervention is to include teaching resources about this topic. This could take the form of synchronous tutorials where students are introduced to each of the challenges identified in this thesis and they are tasked with training ‘toy’ classifiers where they need to balance on the pros and cons of solving a flicker problem between layers (1) and (2), or solving an animation problem between layers (2) and (3). These tutorials could also be delivered asynchronously without an instructor in the form of online community resources, in which users are presented with the ‘toy’ classifiers and they can work on the issues at their own pace.

Additionally, participants explained how challenging it was to diagnose problems during debugging on each layer or in-between layers because of the symbolic opacity of data. Numbers and numerical labels did not mean much for participants when inspecting their datasets, and instead relied on bodily evaluating trained models to bodily ‘explore’ their dataset. Participants followed regular software debugging strategies by making extensive use of the game editor console. We suggest extending the capabilities of the game editor debugger to better interpret symbolic problems with the supervised learning parts of the system. Furthermore, the visual scripting interface could offer break points, stopping the execution at a given point to interpret the system’s state. Feature data visualisations could greatly improve symbolic data interpretation during debugging. That said, even with better visualisations, our observations suggest a fundamental difficulty in translating the system’s numerical output into users’ embodied interpretation. Hence clarifying the symbolic nature of data on screen might only have a limited positive effect on problem diagnosis. A very interesting approach would be to offer embodied debugging capabilities, where not only the user can benefit from more complete debugging tools and visualisations, but can also bodily interpret breakpoints by physically evaluating the classifier 'snapshot' at a given time while in VR, hence grounding debugging on their embodied thinking process. We believe that, for the particular problem of training movement interactions, this embodied debugging methodology could yield superior results to regular data explainability interventions. How this embodied debugging intervention would be designed and evaluated remains an open research question for future work.

Finally, the last theme from the results focused on the tension that arises in the user evaluation of expressive IML outputs between accuracy, visual flicker, and embodied experiential qualities. This tension was due to the factors potentially conflicting with each other. For example, improving accuracy may come at the cost of increased visual flicker or reduced embodied experiential qualities. Similarly, reducing visual flicker may come at the cost of embodied experiential qualities. All the previously mentioned interface and algorithmic interventions should allow users to correctly teach the system what they want to express with their movements. This is a key concept, since the diversity of desired experiential qualities could potentially have grown if our dataset was bigger. Our interpretation was that it wouldn't be possible to generalise what every creator would want to express with a movement, hence giving the most control possible to game creators is of utmost importance.

However, the technical tension between accuracy and visual flicker is equally important, since players would only be able to correctly experience the intended experience through the audiovisual feedback of the game. A possible solution would be to always require some sort of visualisation alongside the classifier inferred label, even if a default one on a primitive, while teaching and evaluating the system. This could change the observed predominant training strategy, in which participants tried to get a classifier "working" first by looking at the symbolic output on the graph or VR panel before piping it into a custom script. Hopefully visualising some degree of flicker could prompt creators to address this tension from the very beginning. Additionally, quantitative metrics could be shown alongside this visualisation, both for accuracy and temporal flicker. However, we are cautious when recommending this approach, because a quantitative evaluation could be understood as a score to maximise by participants and could deter them from conveying their original experiential qualities. This phenomenon, known as intent drift in IML literature, could therefore be negatively heightened by an automated quantitative evaluation. Hence, future research should look at understanding how visually displaying quantitative metrics affect the embodied creative process studied in this thesis.

5.4.4 Limitations

While this study provides valuable insights into how IML could be used for motion-controlled VR game development, we are aware that this study has limitations that should be acknowledged.

Firstly, this study only explored one algorithm (k nearest-neighbour) for classification tasks. Other algorithms may have different strengths and weaknesses. For instance, the chosen k-nearest neighbour algorithms don't provide an intrinsic representation of time and require an explicit supervised learning feature to account for it (i.e., the window of features). Classification algorithms such as InteractML's dynamic time-warping (DTW) could prove easier or more intuitive, but they also suffer from known explicit time segmentation challenges (i.e. the user needs to specify when the gestures begins and ends during inference time). Fully automated time classification algorithms might be better suited for IML interactions, with newer research offering potential solutions [293]. It is also possible that participants would have reacted differently if the task to tackle was a regression task. Future research could explore a wider range of algorithms to identify how they support different types of IML movement interaction prototyping..

Secondly, this study did not explore any quantitative metrics for evaluating the performance of participant classifiers. While participants provided valuable feedback on their subjective experiences with the system, it would be useful to also collect quantitative data on factors such as accuracy and temporal stability. This would provide a more objective measure of the system's performance and could help understand how user perceived accuracy and diverse subjective experiential judgement correlates with objective algorithmic behaviour.

Thirdly, this study only involved one to two days of full prototyping with each participant. While this was sufficient to gain insights into how users engage with IML systems, it is possible that longer periods of prototyping may reveal additional insights or challenges. Future research could explore longer prototyping periods to gain a more comprehensive understanding of how users engage with IML systems over time.

Finally, this study did not explore teamwork dynamics when using InteractML. While participants worked individually during the prototyping sessions, it is likely that teamwork dynamics would play an important role in real-world applications of IML systems. Future research could explore how teamwork affects engagement with IML systems and how these dynamics can be optimised for maximum effectiveness.

Overall, while this study provides valuable insights into how IML can support VR game development, there are several limitations that should be addressed in future research to gain a more comprehensive understanding of these systems and their potential applications.

5.5 Conclusion

This chapter looked into the opportunities and challenges that game creators encounter in prototyping and evaluating game motion controls in VR with InteractML. We observed that participants displayed embodied thinking to train their classifiers, which resulted in clashes between 'naive' embodied mental models of interaction and communication and the operational logic of supervised learning. Participants expected position to be anchored in their first-person body, head, or hand as a reference point, and that a single demonstration would suffice – as it does in human interaction. This produced a particular embodied grokloop of iterative learning, interpretation and steering of the system. Participants benefited from working in-medium and adding structure to their working process, but found it challenging to modularise their ML implementations and separate system responsibilities along the development pipeline. Still, participants found that the visual scripting interface and the IML workflow better supported them in implementing movement interactions compared to their prior experience with traditional rule-based approaches. Finally, we found that the diversity of experiential qualities that participants were interested in delivering with their creations could conflict with their ability to balance perceived classifier accuracy or visual flicker.

Overall, while there is potential for IML to enhance game development processes, there are also significant challenges that need to be addressed before it can be widely adopted. Future research could look to improve the IML methodology either from a human-centred approach to reduce friction during user interaction, and from an algorithmic perspective to reduce the amount of training data required.

6 Discussion and Conclusions

6.1 Introduction

The introduction (Chapter 1) motivated the need for a novel design process and IML tool for VR embodied game motion control design and posited the main research questions that each chapter would provide answers to.

In chapter 2, we found that game creators require tools as part of the game engines they already use to create games, and prior motion control tools were focused on object interactions or inflexible in their movement-focused methodologies. Furthermore, we reviewed that game motion controls can elicit diverse and rich experiential qualities, and that embodied interaction design methodologies might help elicit such qualities better than traditional game design ones. Prior research argued that embodied interaction design processes for interactive technology should be accompanied by implementation of the ideas generated to foster embodied creative reflection, where IML is suggested as a well-suited solution to explore. We conducted a literature review of IML systems that operate on movement data, and demonstrated that, although these IML systems offered effective solutions for various movement-related problems, they did not address the particular needs of game creation practice and game motion control design and implementation. Therefore, Chapter 2 showed that there is a research gap in embodied design and development methodologies for motion controlled videogames in VR.

In Chapter 3 we then tackled such need for a tool with InteractML, our in-engine visual node IML workflow that does not require prior expertise with ML techniques. The tool includes a VR module to perform in-VR model steering loops. The chapter contributes This chapter advances the field of interactive machine learning by presenting a novel IML system that is integrated into a game-engine and supports visual programming to enable non-expert interaction. Moreover, it enriches motion control and embodied interaction design methodologies by incorporating IML implementation and design principles into motion controls design methodologies.

We therefore evaluated InteractML on the three main stages of motion controls creation, with Chapter 4 tackling ideation and Chapter 5 tackling implementation and evaluation. Chapter 4 investigated how game creators ideated VR embodied interactions in-medium, and contributed to advance the field of embodied ideation by showing evidence that the methodology produced rich and diverse IML ideas, and that participants behaved and socially ideated in-VR making use of the embodied affordances of the medium. Chapter 5 investigated how game creators implemented and evaluated their IML ideas with InteractML. The Chapter contributed to the fields of of game motion control design methods and development toolkits as well as the field of interactive machine learning with evidence that participants displayed embodied thinking in the entire IML process, which led to conflicts between their intuitive embodied mental models of interaction and communication and the underlying logic of supervised learning. Furthermore, the Chapter showed an embodied grokloop of participants, with the benefit of using InteractML in-medium and being able to socially include peers in the mental model maturing loop.

Therefore, in the current Chapter, we will review and reflect on the contributions of this thesis. Section 6.2 revisits all the research questions posited in the introduction chapter, summarises all of the contributions presented and synthesises the contribution to research from this thesis as a whole. Section 6.4.1 discusses the implications of our contributions to motion control design and development and suggests solutions to existing problems found. Sections 6.4.2 and 6.4.3 expands on the future work for the methodology explored in this thesis and section 6.5 closes the chapter with a summary of the thesis findings and closing remarks.

6.2 Contributions to Knowledge

The introduction in Chapter 1 presented how existing embodied design processes and IML solutions aren't well suited for game creators designing and implementing motion controls. Therefore, the chapter posited four research questions to tackle this gap, with the overall goal of this thesis being to investigate the role of a novel IML design process and tool for VR embodied motion control design. We now proceed to analyse the extent to which each research question has been answered whithin the body of this thesis.

6.2.1 RQ1: What are the design and implementation considerations for an IML tool for VR game motion control design?

Whithin Chapter 2 the following research question was examined:

> **Research Question 1:** What are the design and implementation considerations for an IML tool for VR game motion control design?

We considered that we brought answers to this theoretical research question based on the literature review carried out within the background Chapter. Section 2.2 provided an overview of the existing motion controllers for gaming applications, and demonstrated how VR has become the domain where motion controls have experienced a revival after the failure of previous commercial consumer entertainment systems in the early 2010s. The section also defined the terminology to technically characterise the tracking features of motion controllers, such as form factors, button layouts, and the predominance of 6 degrees of freedom in current VR systems.

Section 2.3 presented a critical analysis of the principles, tools, and processes of motion game design and development. We identified a lack of standardised motion game design guidelines for VR motion controls and a scarcity of research on the actual or ideal practices or methods of game motion design. Regarding game development, we noted the dominance of industrial challenges such as integration into complex production processes and tooling pipelines, especially the requirement for any tooling and process to be compatible with the chosen game engine. A review of popular industry tools for creating interaction in VR revealed that existing solutions approach interactions from an object-centric perspective with trigger-response pairs, or from a gestural perspective that requires explicit movement segmentation and only considers the main input source (i.e. the VR controller).

Section 2.4 illustrated the diversity of the conceptualisation of experiential qualities of movement, and raised questions of how to better frame and describe motion interactions in digital games. Dimensions such as flow, immersion, engagement or enjoyment affect the "feeling" of the motion controlled game and the different interactional aesthetics conveyed. And, because of the predominance of motion controls in VR, presence and theories of embodiment provided grounds for the body as a central part of the experience and, even, cognition.

Section 2.5 presented embodied interaction design theories and methodologies grounded on the body, providing evidence that designing embodied interactions can benefit from methods where body movements are central to the design process. The section examined that the first-person *lived* body can be employed for reasoning, creativity and reflection. Lastly, we reviewed literature arguing embodied creative reflection requires a tight loop of design and implementation, with IML as a suggested implementation method.

Section 2.6 provided an overview of IML and its relevant work. We observed that the IML process consists of rapid and focused iterations in the model steering loop. The section then examined relevant IML systems from the literature that work with movement data, showing that none of the current systems meet the specific requirements of game creation practice and game motion control design and implementation outlined in section 2.3.

We therefore synthesised the following design and implementation considerations for an IML tool for VR game motion control development:

1. From section 2.2, we synthesised that the tool should support current mass-marketed forms of motion controls, with special emphasis on two-handed motion controlled VR systems.

2. From section 2.3 we synthesised that the tool should fully work in-engine and provide flexible exploration of movement-focused interactions given the lack of standard design methods.

3. From section 2.4 we synthesised that the tool should support developers expressing a rich diversity of experiential qualities.

4. From section 2.5 we synthesised that the tool should support a design and implementation methodology grounded in body movement, with bodystorm-

ing and IML being theoretically promising design and implementation approaches.

5. From section 2.6 we synthesised that current IML solutions are not suited for the game creation specific considerations synthesised from Section 2.3, therefore motivating the need for a new IML tool. From this section we also synthesised the IML requirements that the tool should support

We believe that the literature review and our synthesis of design and implementation considerations positively answered research question 1 and helped theoretically ground how InteractML should work.

6.2.2 RQ2: What is a functional IML tool that addresses these considerations?

Within Chapter 3, and with the implementation evidence of Chapter 5, the following research question was examined:

> **Research Question 2**: What is a functional IML tool that addresses these considerations?

We considered that we positively answered this practical research question describing the design decision and implementation details of InteractML in Chapter 3, together with the evidence from the implementation workshops in Chapter 5. InteractML presents a visual and embodied approach to designing movement-based interactions within the game engine, thus reducing the need for prior knowledge in machine learning techniques. Additionally, the tool incorporates a VR module, enabling in-VR model steering iterations by leveraging the motion tracking capabilities of the VR system. Consequently, InteractML empowers game developers to create nuanced movement interactions using IML, thereby attempting to capture the embodied experiential expressiveness that game creators want to deliver.

Within Chapter 3, section 3.4 established the main contribution of the chapter, which is InteractML's in-engine visual and embodied IML workflow. Section 3.5 described the implementation details of the tool. Section 3.6 detailed how each of the IML stages worked from an interface and methodology perspective by describing each node and relating it to an IML methodology step, with the VR module details described in section 3.5.8. Lastly, section 3.11 reflected on the core contribution of the tool and how it advances the state of the art of game motion control toolkits and IML. Whithin Chapter 5, the results shown in section 5.3 showed how creators used the tool to successfully prototype motion controls in VR, and how richly their expressive intent was.

Thus, the consideration 1 was addressed by the VR support of InteractML described in Chapter 3 and the VR prototypes developed by game creators in Chapter 5.

The consideration 2 was addressed by the fully in-engine nature of InteractML described in Chapter 3 and the in-engine working process investigated in Chapter 5. The consideration 3 was addressed with the description of the flexible design frame of InteractML to support a rich set of experiential qualities in Chapter 3, and the rich evidence from creators experiential intent reported in Chapter 5. The consideration 4 and 5 were addressed by the possibilities of InteractML's IML stages, where game creators visually select features, models and configure the graph; and model steering can be performed in-medium making the most of the principles behind embodied interaction design and thus, attempting to support a rich expression of experiential qualities from creators embodied design. Such possibilities are supported by the evidence gathered in Chapter 5, where creators indeed managed to successfully follow in-medium embodied IML model steering iterations and produced working models that reacted to their movements. In section 5.4 we discussed what these meant for IML and how InteractML was a significant contribution in supporting (a) an embodied working process, and (b) game creation needs compared to prior IML working processes.

Thus, Chapter 3 contributes to the growing field of IML by advancing the state-of-the-art through the development of a game-specific IML system, seamlessly integrated into a game engine and visually programmed to facilitate non-expert engagement. Moreover, Chapter 5 evidence of creators successfully prototyping motion controls in VR using InteractML supports the design intent from Chapter 3. Hence, InteractML enriches the motion control and embodied interaction design fields by incorporating in-VR IML implementation and design principles into motion control design frameworks.

6.2.3 RQ3: What are the opportunities and challenges that game creators encounter when ideating VR game motion controls in-medium?

Chapter 4 examined the following research question:

> **Research Question 3**: What are the opportunities and challenges that game creators encounter when ideating VR game motion controls in-medium?

We provided insights into the research question with the qualitative analysis detailed in section 4.3 and subsequent reflection on section 4.4. Theme 1 synthesised the output space of the interactions ideated by creators, with subtheme 1.a detailing the effects on virtual elements of the ideas and subtheme 1.b the social aspect of ideas. Theme 2 synthesised the input space of the interactions ideated by creators, with subtheme 2.a reflected the richness of body modalities, and subtheme 2.b reflected the expressiveness of movement in participants' ideas. Theme 3 synthesised the expressiveness in the participants' embodied ideation process. Subtheme 3.a reflected the embodied nature of in-medium affordance explore, subtheme 3.b reflected the playfulness present in co-located ideation, and subtheme

3.c reflected socio-technical challenges of in-medium embodied ideation. Lastly, theme 4 synthesised the communicative nature of in-medium embodied ideation. Subtheme 4.a reflected that non-verbal cues were inherent in communication and embodied ideation, subtheme 4.b reflected the embodied repair strategies during ideation, and subtheme 4.c reflected the importance of smooth turn-taking during in-medium remote embodied ideation.

Section 4.4 reflected on the meaning of the findings, and we discussed that the space where the embodied ideation took place influenced the playfulness of the ideas and the enriching effect of in-medium co-located embodied ideation. We also discussed the interplay between play, novelty and creativity and how does it affect the exploration of affordances in-medium. Furthermore, we argued that the embodied cognition displayed by participants when ideating variations of an interaction was related to the theory of the adjacent possible, in which embodied thinking process is cognitively grounded in the possibility space of participants' bodies and movement. Moreover, we argued that smooth communication and joint attention had a positive effect on social co-creation, as participants 'picked-up' each other movements and continued them. Additionally, we discussed how actually designing in-VR positively used the constrained affordances of the medium to focus on more realisable IML ideas. On the other hand, in-medium embodied ideation also suffered from socio-technical and usability constrains that were potentially detrimental to idea generation.

Therefore, Chapter 4 contributed to the fields of embodied interaction design knowledge, showing the opportunities for in-medium social embodied ideation and the socio-technical challenges that such a process entails.

6.2.4 RQ4: What are the opportunities and challenges that game creators encounter in prototyping and evaluating game motion controls with an IML tool?

Chapter 5 examined the following research question:

> **Research Question 4**: What are the opportunities and challenges that game creators encounter in prototyping and evaluating game motion controls with an IML tool?

We qualitatively answered the research question with the analysis presented in section 5.3. Theme 1 synthesised that creators displayed an embodied thinking process throughout their implementation and evaluation of their interactions. Theme 2 synthesised the tension existing between their embodied thinking and the IML requirements. Subtheme 2.1 delved deeper into the tension between participant embodied understanding of features and the actual computational effect, subtheme 2.2 described the tension between participant embodied understanding of features and time, and subtheme 2.3 reflected on the tension between participant first-person body point of reference for thinking and the third-person computer understanding. Theme 3 synthesised the in-medium iterative process of participant engagement with IML models. Subtheme 3.1 described that the in-medium engagement process can is affected by model output display, subtheme 3.2 described that people rely on quick iterations of trial and error, and subtheme 3.3 reflected the negative impact that media-breaks can have of in-medium model steering. Theme 4 synthesised the positive effect of structure in the editor and in the process, and subtheme 4.1 described that game creators can create structure on their process when there is none present. Theme 5 synthesised that visual scripting facilitated working with IML and maturing mental models about IML. Subtheme 5.1 tackled the issue of modularity not being obvious to creators, and subtheme 5.2 described the challenges in separating system responsibilities during problem diagnosis. Lastly, theme 6 synthesised the tension existing between perceived accuracy, interaction visual flicker and intended embodied experiential qualities, with subtheme 6.1 detailing the richness of intended embodied experiential qualities.

Section 5.4 reflected on the meaning of findings on motion control design and development and IML. We discussed that, while game creators managed to implement IML motion controls without being ML experts, there were patterned opportunities and challenges across their interaction. We suggested a set of design guidelines to make use of present opportunities and minimise challenges, grounded in our findings that creators profited from in-medium embodied work on cognitive and productive dimensions. We henceforth suggested to incorporate as much movement as possible throughout the entire IML process, to reduce iteration cost, to compute features from a first-person body perspective, to suppport fully in-medium embodied work, to modularise the training dataset, to design a distinct set of movements with nuance tackled incrementally, to prioritise interaction visual flicker and to focus on movement instead of objects. We argued that visual scripting had a positive impact on the overall IML-driven motion control prototyping and evaluation because game creators could visually structure the data flow through the IML graph and get a 'visual intuition' of where the problem laid during debugging.

Furthermore, we reflected how our findings on interactive feature selection compared to those from the literature [177, 178], and while the literature argued that users were "bad" at feature selection, our findings go beyond that and argued that features don't match creators embodied first-person perspective. We also extend prior IML literature arguing that accuracy is a deceiving and incomplete quantifiable metric for game motion controls, since we found that visual flicker or movement experiential qualities are far richer potentially quantifiable metrics to account for. Furthermore, we discussed that our findings about the social aspect of human engagement with IML models expand prior work on mixed-initiative generative creative processes [49, 152]. Lastly, we discussed how one of the chapter's core contributions was how 'ideal' engagement with InteractML would be a full in-medium working loop, either on desktop or VR, without media-breaks. We speculated how such full in-VR work would look like and described future interventions.

Therefore, Chapter 5 contributed to the fields of game motion control design and development, human-centred IML and embodied interaction design, discussing the socio-technical opportunities for in-engine IML embodied motion control implementation and evaluation, and the challenges that such an embodied working process entails.

6.2.5 General insights about investigating the role of a novel IML design process and tool for VR embodied motion control design

So far, the thesis has detailed the contributions of each chapter. However, when we began motivating our work, we understood the need to create a novel IML design process ad tool tailored for VR game motion control creation specific needs throughout the main stages of creation: ideation, implementation and evaluation [305, 98]. Each empirical chapter contributed knowledge onto each stage, but, what are the overall contributions to knowledge after going through the entire process? We will proceed to

We believe that the thesis yield three core contributions: (1) designing *for* VR benefits from designing *in* VR, (2) designing embodied interactions is and benefits from an embodied process with a social embodied cognitive dimension to it, and (3) the logic of supervised machine learning of movement clashes with people's embodied logic of how movement and learning work. We will proceed below to detail each contribution:

Designing *for* VR benefits from designing *in* VR.We found that designing VR motion controls *within* VR improved the users creative process alongside the ideation, implementaton and evaluation stages. Chapter 4 found that embodied ideation in SocialVR provided a rich medium for creativity because of the support for non-verbal cues and the specific in-medium affordance exploration that participants would have missed were they not in VR. Furthermore, Chapter 5 found that IML-driven VR game motion control implementation and evaluation benefits from an in-VR working process, because of the positive use of creators' first-person embodied thinking and the negative effect of media-breaks that require creators to place or remove the headset.

Wearing the VR headset, holding the controllers and pressing the buttons offers a physical feeling that helps creators bodily *feel* the nuances of the medium during ideation, with an immediacy that would be impossible to feel withtout being in VR. If a creator moves in a energetic and open manner, they can feel the weight of the headset and controllers, and how the stereoscopic VR image looks like when they are moving. For instance, when they want to include a button press in their interaction, they can bodily *feel* how the button springs when they press it, how the movement works in conjunction with the motion, and how their avatar immediately reacts to the button and motion press. Similarly, while implementing and evaluating a movement interaction in VR, the game creator can make use of their embodied thinking process and focus on it while working fully in-medium, hence why media-breaks were so disruptive when switching from VR to desktop and vice-versa. The immediacy of the visual effect also helped creators grasp important qualities that could have been missed, such as how prominent the interaction visual flicker was once they saw and felt their own IML model. Recording and training IML models in-VR also allowed them to better perform the human-model "dialogue" to bodily feel how their movements affected the decision boundaries of the model, and how their training data was bodily structured.

Therefore, we consider that creating *for* VR benefits from creating *in* VR is the first general insight gained from the thesis work.

Designing embodied interactions *is* and *benefits from* an embodied process with a social embodied cognitive dimension to it.We found that designing embodied interactions *is* an embodied process and *benefits from* an embodied process with a social embodied cognitive dimension to it alongside the ideation, implementaton and evaluation stages. Chapter 2 argued that game creators followed a thinking process grounded in the body, in which they sensed the possibility space of their movements while ideating. The Chapter also found how social co-creation was embodied, with creators continuing each other's movements or interacting with co-located props and constructing social interactions from individual interactions. Additionally, it found how relevant non-verbal cues and smooth communication was important from a body perspective, therefore allowing creators to bodily express and bodily understand each other movement ideas. Chapter 5 found that game creators displayed an embodied thinking process and it argued that creators thought about features and model decision boundaries via a first-person body experience. Furthermore, the chapter found that the richness of experiential qualities that creators want to convey with their interactions is also grounded in the body, with dimensions such as fluidity, smoothness or self-consciusness of movement present in their embodied evaluation process. It similarly found that creators included a social aspect to their 'grokloop' of human engagement with IML models, in which to correctly interpret a model they could include a peer to discuss their movement interactions, making use of a similar set of non-verbal and communicative processes to those from the ideation Chapter, but grounding the discussion on the embodied feeling of interacting with an already implemented model to correctly interpret it.

Prior literature argued that designers should "design by moving" [121] and that movement interfaces are effective because of their non-presentational and sensorimotor grounding [94], that the body can be used for both reasoning and creativity [281], and that embodied interaction design can benefit from embodied interaction implementation [82, 93]. However, our contribution goes beyond current work by providing evidence of embodied interaction design with an observable social embodied cognitive dimension, where creators can bodily express and understand each other designs through social communication grounded in the body. We found qualitative evidence of creators bodily communicating ideas while performing embodied interaction co-design in Chapter 4, bouncing movement ideas between participants via body movement reflection and exploring adjacent possible ideas collaboratively. Similarly, Chapter 5 provided qualitative evidence for how participants incorporated an embodied social dimension to their IML "grokloop", where their interpretation of their recently trained classifier would benefit from sharing their embodied understanding with co-located peers that could bodily construct a *better* mental model of their system to support them.

Therefore, we consider that designing embodied interactions is and benefits from an embodied process with a social embodied cognitive dimension to it, is the second general insight gained from the thesis work.

The logic of supervised machine learning of movement clashes with people's embodied logic of how movement and learning work. We found that the IML logic of movement clashes with people's embodied logic of how movement and learning work alongside the ideation, implementaton and evaluation stages. Chapter 4 found that participants exploration of their in-medium body to avatar affordances was grounded in a first-person body experience, and similarly Chapter 5 found that participant understanding and mental modelling of the IML system was grounded on an first-person body experience. Such first-person perception clashes with supervised learning third-person logic. Therefore, this is a core contribution to human-centred IML, because current movement-driven IML literature investigate how to make users better at feature selection [177, 178] or how users want to evaluate their models [73, 276], but it doesn't focus on the *why* users interact with the IML work from a first-person lived body perspective.

Therefore, we consider that the IML logic of movement clashes with people's embodied logic of how movement and learning work, is the third general insight gained from the thesis work.

6.3 Limitations

We present the main limitations shared accross all studies presented throughout this thesis:

Population: We acknowledge that the samples in both empirical studies might have affected our results, and limited their generalisability because participants were either university students with graduate experience in game creation, or independent developers with limited knowledge of game studios industrial working processes. In order to generalise our findings onto game studios, future work should deploy and evaluate InteractML on game studios with business needs to better generalise our claims.

Limited workshop time: Both empirical studies presented in this thesis were performed on workshops, either remote or in-person, an involved limited amounts of time with each participant. This approach enabled us to synthesise insights about InteractML, embodied ideation and the user engagement with IML systems. However, extending the duration of the studies might uncover further insights or difficulties. Therefore, future studies are encouraged to investigate the long-term effects of including an IML-driven intervention for motion control design and implementation.

Limited large scale adoption generalisability: This thesis employed a qualitative methodology that relied on a limited sample size that might not be representative of the larger population of game creators, and instead focused on rich specific human-centred insights about our IML embodied design process and tool. We acknowledge the limitations of our method and suggest future large scale quantitative research grounded on our rich insights to better understand the generalisability of the thesis contributions.

COVID-19: The data from this thesis was collected during 2020 and 2022, when the worldwide COVID-19 pandemic required the UK population to enter or exit lockdowns, and social restrictions were gradually 'eased' in 2022. Therefore, the pandemic was likely to have an effect on participants, because of limited physical and social interaction in the remmote workshops, or because of the usage of infection prevention tools (e.g. masks, tests) in the in-person workshops. Based on our analysis, we believe any effect on the data to be small.

6.4 Implications and Future Work

6.4.1 Implications for Game Development

In chapter 5 we saw that one of the challenges that creators faced was how ML thinking differs from regular game dev thinking.

This is a non-trivial problem, since the rest of the game engine systems function in a very different way. Nonetheless, new paradigms of work have previously been successfully introduced and adopted in game development. One of the most significant ones was 3D rendering at the end of the twentieth century. 3D drastically changed the way games were made, not only by 'just' adding a third dimension when writing gameplay code, but by introducing completely novel ideas such as skeletal animation [254] and programmable shaders [159]. These breakthroughs in computer graphics demanded new skill sets and paradigms of work that required people learning new ways of thinking to specialise in 3D character animation, rigging, skinning and texturing [168], or shader programming [159]. We assume that, even though ML thinking is now a fairly different way of thinking for game creators, the adoption of ML powered tools in game engines will slowly reduce the friction in this sense and ML-thinking will be one of the many ways in which a game creator needs to think to build a videogame. There has been steady work during the last decades transferring ML research breakthroughs into game engines, with examples such as the Kinect computer vision algorithms [321] and reinforcement learning game playing agents with unity ml-agents [198]. And, with this trajectory, current relevant ML research, such as large language models (LLMs) to generate programming code [319] or ML-based asset generators [233], will inevitably be transferred into game engines. This will hopefully contribute to communicating the data-centric nature of machine learning that will eliminate the cognitive friction when a creator describes the behaviour of a tool by recording a training dataset.

Nonetheless, eliminating friction with the ML-way of thinking does not reflect on the most important finding of this thesis, which is that game creators grounded their design and thinking in their bodies during motion-control prototyping. That is, they follow an embodied development process. We could speculate that this finding could potentially be translated to other game engine systems, but which ones? Would an IML workflow benefit the system? If not, which other embodied workflows?

We can make the educated guess that systems that could benefit from IML-driven embodied development metaphors should be the first ones to be investigated. For instance, the game animation pipeline could employ embodied IML. Instead of the animator using a mouse to describe animations, the animator could physically demonstrate animations. This, in a way, already happens with motion captured data, where actors perform movements under the eye of high-fidelity motion capture systems. However, animators later clean motion captured data 'by hand' using a mouse-driven GUI to tweak recorded animations to make them work with virtual characters. Animators could as well follow an embodied development process when cleaning up animations, where instead of using mouse based GUIs they directly physically demonstrate how to correct the recorded data. Even following the spirit of interactive machine learning, performers themselves could interactively visualise how the final animation will look while recording data, and correct it through embodied demonstrations. Previous IML interventions have been explored in the field of virtual character animation [91, 93], but what we suggest here is an embodied development intervention into the entire animation pipeline. Even for low-budget animations, techniques of embodied IML-driven animations could be made with webcams and/or VR kits, where the indie game creator has a full embodied control of how the movement data is recorded and displayed by the virtual character.

Another game engine system that could benefit from an IML-driven embodied development metaphor are camera control and animation systems, where creators can teach the computer how the camera should be animated and controlled in response to player input. This might be a potential natural fit because the game engine camera system is already an abstraction that replicates how a real camera functions. Game creators could animate the camera by recording demonstrations of themselves holding the virtual camera in VR and recording its position, zoom or pan attributes. Previous work explored how VR cameras could be used to explore camera takes during pre-production for computer generated movies [13], but we are suggesting programming camera behaviours through embodied demonstrations. We could even generalise this principle to any virtual object, thus creating an object-focused embodied teaching metaphor where creators teach how a virtual object should behave through embodied demonstrations. For example, game creators might demonstrate how a gun should animate when it is shot by moving the gun, and link each of the animations with its corresponding in-game action.

Moreover, this methodology could include programming the behaviour of particle systems by bodily describing emission trajectories or demonstrating with the distance between two hands the size-over-lifetime attributes of emitted particles. Similarly, general object placement in the scene could, up to a certain degree, be programmed via embodied demonstrations. For instance, the game creator could select an object in a VR interface and perform strokes with their upper arms to place objects in the scene, as if they would be ‘throwing them out of a basket’, and specify their animation attributes over time or in response to player input or in-game events. This workflow would be limited in scope, but could potentially be used to quickly sketch a reactive scenario in VR by bodily throwing objects around, or to prototype certain fine-grained game events such as rocks falling off a cliff (from recorded variations of the creator throwing a rock) after triggering a certain in-game condition. Likewise, it could be possible to explore asset generation via pairing body movements and asset outputs. This might enable the possibility of generation of complex 3D models defining how they should move (i.e. moving like a monkey, generate a monkey). Game audio generation or reactive audio could benefit from IML, as it is already and active area of research that could be translated into game development [74, 291].

6.4.2 Multimodal Embodied Development

Furthermore, there is evidence that body gesturing is entangled with verbal communication [25, 227, 214]. Thus, this might indicate that the embodied cognition process mentioned in [145, 93], and the embodied thinking process observed in this thesis have a verbal component to it. This might make sense as in the ideation study participants expressed their movements to each other verbally and non-verbally as seein in Chapter 4. In the implementation study, participants also used verbal communication when describing their movements to their peers during the ideation phase, but we observed silent periods of embodied thinking during their time with the computer in Chapter 5. The fact that participants were silent in the embodied thinking instances we observed doesn’t mean that they were not ‘speaking’ to themselves while thinking and gesturing, and rather they might have not displayed any evidence of verbally grounded thinking because talking to the computer is, not only irrelevant, but socially awkward in a shared working environment. We don’t know if this was the case, but there is room for future research investigating the potential verbal and non-verbal relationship during embodied cognition and game creation.

Still, if we assumed that there was indeed some relationship, and this is just speculation, we could look into real-life instances where this relationship was present. For example, if we look at the way dance instructors teach their students, it is not enough for the instructor to physically demonstrate dance moves, but they need to verbally explain the steps and how each of them connect together in a sequence. Likewise, it is not enough for the students to only practise the physical movements, but also they benefit from verbally asking questions and engaging with other students in loops of verbal and non-verbal demonstrations to correctly understand the dance moves. Hence, assuming that embodied cognition is both grounded in the body and speech, we could further enhance embodied development processes to also include verbal signals. Game creators could implement movement interactions in the same way that they ideated them, doing physical demonstrations accompanied with verbal explanations. This could therefore reduce the friction originating from the interaction with a visual interface, and could potentially refer to the "dialogue" that participants understood from their interaction with their classifiers (Chapter 5).

This is a very exciting idea, and it can drastically enhance the previously suggested embodied interventions in animation, camera, particle and object interactions. Creators could verbally explain to the computer the qualities of their movements, and verbally and non-verbally correct the computer inferred outputs when they don't match their expectations. For example, an animation could be physically demonstrated together with labels and the order the movement in the sequence. Camera behaviour can be generalised mixing verbal and non-verbal demonstrations (e.g. 'the camera needs to zoom onto the character, and then move in this way for a dramatic effect [the user physically moves the virtual camera]'). Particle emission attributes such as colour, size, or rate of emission can be detailed through verbal explanations as the gestural strokes are demonstrated, and correspondingly with object placement in the virtual scene (e.g. the user describes that a street should be created in front of him, and physically gestures in which way crossings and traffic should go). Embodied 3D model generation could be enhanced via verbal descriptions. For instance, the user moves as a monkey and verbally describes what colour the monkey should be or points to some areas of their body and specifies additions like 'place an armour here' or 'remove the hair here and place a tattoo of a red rose'. Furthermore, embodied audio generation could be explicitly instructed verbally (i.e. the user moves and explains how they want piano notes matching upper body movement and a violin lower body), or imitate sounds or instruments (i.e. humming) as they move to generate more nuanced audio from their motion.

Additionally, this multimodal embodied development process could even help to program the behaviour of game playing agents via multimodal reinforcement learning demonstrations, in a reminiscent manner of how people would teach pets how to behave. Users could point to a location, say 'walk there this way' and then proceed to physically demonstrate how the game playing agent should walk. Likewise, if the agent should behave more believably, the user could verbally correct the agent saying ‘not like, you should look into the eyes of your speaker as you are listening but not too deeply, like this’ then proceed to demonstrate gaze behaviours. Theoretically, any sort of behaviour could be described following the suggested multimodal embodied development process, such as algebra [229] or the original example of dancing.

6.4.3 Collaborative Multimodal Embodied Development

Moreover, we can speculate that for these multimodal embodied development methods to really ‘shine’ they should be extended to support **collaborative work.** Since we saw how much potential there was for embodied ideation in SocialVR in Chapter 4, we can translate the benefits of co-location and smoother communication into IML. Thus, as creators collaboratively bodystorm and explain to each other how their movement interactions would feel like, they can collaboratively construct machine learning models on-the-fly to genuinely *feel* whether their ideas match their expectations. Creators could easily modify their creations just by giving verbal corrections and body demonstrations, equivalently to how a couple of dance instructors collaborate to teach a students about the choreography.

We believe that this is where the methodology should head in the future, hence blurring the lines between collaborative embodied ideation and multimodal embodied implementation into one flexible methodology grounded in the body and mind. Future research could investigate (a) which mediums are better suited for collaborative multimodal embodied development (e.g.. How in-person interventions compare to the technological ‘magicality’ of SocialVR?), (b) which algorithms can better support two or more creators steering IML models simultaneously, (c) how participants’ mature their mental models of their IML models via iterative collaborative embodied development, and (d) what collaborative debugging strategies do participants follow.

6.5 Conclusion

This thesis has explored how a novel in-engine IML design process and tool –InteractML–support game creators designing and implementing motion controls in VR. Our findings suggests that the tool was successful in facilitating the creation of movement-focused interactions. Furthermore, we found that designing for VR is better done in VR, supporting an in-medium embodied creative reflection loop where the affordances of VR, the movement ideas and the IML system can be bodily explored. We additionally found that there is a social embodied cognitive dimension to embodied interaction design, where creators bodily co-create with each other or bodily interpret each other IML systems. Lastly, we found that the IML logic of movement computation clashes with creators’ embodied understanding of their own movement, with a dissonance with the creator first-person lived body experience and the machine third-person computational processing.

We hope that the work presented in this thesis helps to focus future research on deployment and evaluations of our tool and methodology in game studios and can help bring a new paradigm of game motion control design.

References

[1] N. Abel. *NewtonVR: Physics-based interaction on the Vive (Part 1)*. [Accessed 22 Sep. 2015]. 2015. URL: http://www.vrinflux.com/newton-vr-physics-based-interaction-on-the-vive/.

[2] David J. Abramis. “Play in Work: Childish Hedonism or Adult Enthusiasm?” In: *American Behavioral Scientist* 33.3 (1990), pp. 353–373. ISSN: 15523381. DOI: 10.1177/0002764290033003010.

[3] Ritu Agarwal and Elena Karahanna. “Time Flies When You’re Having Fun: Cognitive Absorption and Beliefs about Information Technology Usage”. In: *MIS Quarterly* 24.4 (2000), 665–694. ISSN: 02767783. DOI: 10.2307/3250951. URL: http://www.jstor.org/stable/3250951.

[4] N. Albertini, A. Brogni, R. Olivito, E. Taccola, B. Caramiaux, and M. Gillies. “Designing natural gesture interaction for archaelogical data in immersive environments”. In: *Virtual Archaeology Review* 8.16 (2017), pp. 12–21.

[5] Saiqa Aleem, Luiz Fernando Capretz, and Faheem Ahmed. “Critical success factors to improve the game development process from a developer’s perspective”. In: *Journal of Computer Science and Technology* 31 (2016), pp. 925–950.

[6] S. Amershi, M. Cakmak, W.B. Knox, and T. Kulesza. “Power to the People: The Role of Humans in Interactive Machine Learning”. In: *AI Magazine* 35.4 (2014), p. 105.

[7] Víctor Hugo Andaluz, Fernando A. Chicaiza, Cristian Gallardo, Washington X. Quevedo, José Varela, Jorge S. Sánchez, and Oscar Arteaga. “Unity3D-MatLab Simulator in Real Time for Robotics Applications”. In: *Augmented Reality, Virtual Reality, and Computer Graphics*. Ed. by Lucio Tommaso De Paolis and Antonio Mongelli. Cham: Springer International Publishing, 2016, pp. 246–263. ISBN: 978-3-319-40621-3.

[8] Craig Alan Anderson, Douglas A Gentile, and Katherine E Buckley. *Violent video game effects on children and adolescents: Theory, research, and public policy*. NY, USA: Oxford University Press, 2007.

[9] Eike Falk Anderson, Steffen Engel, Peter Comninos, and Leigh McLoughlin. “The Case for Research in Game Engine Architecture”. In: *Proceedings of the 2008 Conference on Future Play: Research, Play, Share*. Future Play ’08. Toronto, Ontario, Canada: Association for Computing Machinery, 2008, pp. 228–231. ISBN: 9781605582184. DOI: 10.1145/1496984.1497031.

[10] Vladislav Angelov, Emiliyan Petkov, Georgi Shipkovenski, and Teodor Kalushkov. “Modern virtual reality headsets”. In: *2020 International congress on human-computer interaction, optimization and robotic applications (HORA)*. IEEE. 2020, pp. 1–5.

[11] James J Appleton, Sandra L. Christenson, and Michael J. Furlong. “Student engagement with school: critical conceptual and methodological issues of the construct”. In: *Psychology in the Schools* 45.5 (2008), 369–386. DOI: 10.1002/pits.

[12] Laura Aymerich-Franch. “Presence and Emotions in Playing a Group Game in a Virtual Environment: The Influence of Body Participation”. In: *Cyberpsychology, Behavior, and Social Networking* 13.6 (Dec. 2010), pp. 649–654. ISSN: 21522715. DOI: 10.1089/CYBER.2009.0412. URL: https://www.liebertpub.com/doi/10.1089/cyber.2009.0412.

[13] Philippe Bédard. “Virtual Production and the Transformation of Cameras Mechanical, Virtual, and Actual”. In: *Animation* 17.2 (2022), pp. 226–243. DOI: 10.1177/17468477221102498.

[14] Steve Benford, Holger Schnädelbach, Boriana Koleva, Rob Anastasi, Chris Greenhalgh, Tom Rodden, Jonathan Green, Ahmed Ghali, Tony Pridmore, Bill Gaver, Andy Boucher, Brendan Walker, Sarah Pennington, Albrecht Schmidt, Hans Gellersen, and Anthony Steed. “Expected, Sensed, and Desired: A Framework for Designing Sensing-Based Interaction”. In: *ACM Transactions on Computer-Human Interaction* 12.1 (Mar. 2005), pp. 3–30. ISSN: 1073-0516. DOI: 10.1145/1057237.1057239.

[15] Dave Bennett and Patrick Jalbert. *Until You Fall: Building Satisfying VR Combat on a Budget*. 2020. URL: https://youtu.be/G5a5VIdjiJA (visited on 06/27/2023).

[16] Björn Berg Marklund, Henrik Engström, Marcus Hellkvist, and Per Backlund. “What empirically based research tells us about game development”. In: *The Computer Games Journal* 8 (2019), pp. 179–198.

[17] Donald J. Berndt and James Clifford. “Using Dynamic Time Warping to Find Patterns in Time Series”. In: *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining*. AAAIWS’94. Seattle, WA: AAAI Press, 1994, pp. 359–370.

[18] Ferran Altarriba Bertran, Alexandra Pometko, Muskan Gupta, Lauren Wilcox, Reeta Banerjee, and Katherine Isbister. “Designerly Tele-Experiences: A New Approach to Remote Yet Still Situated Co-Design”. In: *ACM Transactions on Computer-Human Interaction* 29.5 (Oct. 2022). ISSN: 1073-0516. DOI: 10.1145/3506698.

[19] Nadia Bianchi-Berthouze. “Understanding the Role of Body Movement in Player Engagement”. In: *Human–Computer Interaction* 28.1 (2013), 40–75. ISSN: 1532-7051. DOI: 10.1080/07370024.2012.688468. URL: http://www.tandfonline.com/doi/pdf/10.1080/07370024.2012.688468.

[20] Lennart Björneborn. “Adjacent Possible”. In: *The Palgrave Encyclopedia of the Possible* (2020), pp. 1–12. DOI: 10.1007/978-3-319-98390-5_100-1.

[21] Renaud Blanch, Yves Guiard, and Michel Beaudouin-Lafon. “Semantic Pointing: Improving Target Acquisition with Control-Display Ratio Adaptation”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI ’04. Vienna, Austria: Association for Computing Machinery, 2004, pp. 519–526. ISBN: 1581137028. DOI: 10.1145/985692.985758.

[22] Mark Blythe and Marc Hassenzahl. “The Semantics of Fun: Differentiating Enjoyable Experiences”. In: *Funology: From Usability to Enjoyment*. Dordrecht: Springer Netherlands, 2004, pp. 91–100. ISBN: 978-1-4020-2967-7. DOI: 10.1007/1-4020-2967-5_9. URL: https://doi.org/10.1007/1-4020-2967-5_9.

[23] Costas Boletsis, Amela Karahasanovic, and Annita Fjuk. “Virtual Bodystorming: Utilizing Virtual Reality for Prototyping in Service Design”. In: *Augmented Reality, Virtual Reality, and Computer Graphics*. Ed. by Lucio Tommaso De Paolis, Patrick Bourdot, and Antonio Mongelli. Cham: Springer International Publishing, 2017, pp. 279–288. ISBN: 978-3-319-60922-5.

[24] Bert Bongers and Gerrit C. Veer. “Towards a Multimodal Interaction Space: Categorisation and Applications”. In: *Personal and Ubiquitous Computing* 11.8 (Dec. 2007), pp. 609–619. ISSN: 1617-4909. DOI: 10.1007/s00779-006-0138-8.

[25] Hans R Bosker, Marieke Hoetjes, Wim Pouw, and Lieke van Maastricht. *Gesture-speech coupling in L2 lexical stress production: A pre-registration of a speech acoustic and gesture kinematic study*. July 2021. DOI: 10.31219/osf.io/w2ezs.

[26] Nadia Boukhelifa, Anastasia Bezerianos, and Evelyne Lutton. “Evaluation of Interactive Machine Learning Systems”. In: *Human and Machine Learning: Visible, Explainable, Trustworthy and Transparent*. Ed. by Jianlong Zhou and Fang Chen. Cham: Springer International Publishing, 2018, pp. 341–360. ISBN: 978-3-319-90403-0. DOI: 10.1007/978-3-319-90403-0_17. URL: https://doi.org/10.1007/978-3-319-90403-0_17.

[27] Elizabeth A. Boyle, Thomas M. Connolly, Thomas Hainey, and James M. Boyle. “Engagement in digital entertainment games: A systematic review”. In: *Computers in Human Behavior* 28.3 (May 2012), 771–780. ISSN: 07475632. DOI: 10.1016/j.chb.2011.11.020. URL: http://linkinghub.elsevier.com/retrieve/pii/S0747563211002640.

[28] Virginia Braun and Victoria Clarke. “Using thematic analysis in psychology”. In: *Qualitative Research in Psychology* 3.2 (2006), pp. 77–101. ISSN: 14780887. DOI: 10.1191/1478088706QP063OA.

[29] Jeffrey Breugelmans, Yingzi Lin, Ronald R. Mourant, and Maura Daly Iversen. “Biosensor-Based Video Game Control for Physically Disabled Gamers”. In: *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* 54.28 (Sept. 2010), pp. 2383–2387. ISSN: 1541-9312. DOI: 10.1177/154193121005402805.

[30] E Oran Brigham. *The fast Fourier transform and its applications*. Prentice-Hall, Inc., 1988.

[31] Thor Brigsted. *xNode: Unity Node Editor*. https://github.com/Siccity/xNode. 2018.

[32] Jeanne H. Brockmyer, Christine M. Fox, Kathleen a. Curtiss, Evan McBroom, Kimberly M. Burkhart, and Jacquelyn N. Pidruzny. "The development of the Game Engagement Questionnaire: A measure of engagement in video game-playing". In: *Journal of Experimental Social Psychology* 45.4 (July 2009), 624–634. ISSN: 00221031. DOI: 10.1016/j.jesp.2009.02.016. URL: http://linkinghub.elsevier.com/retrieve/pii/S0022103109000444.

[33] R. J. Brodie, L. D. Hollebeek, B. Juric, and a. Ilic. "Customer Engagement: Conceptual Domain, Fundamental Propositions, and Implications for Research". In: *Journal of Service Research* 14.3 (July 2011), 252–271. ISSN: 1094-6705. DOI: 10.1177/1094670511411703. URL: http://jsr.sagepub.com/cgi/doi/10.1177/1094670511411703.

[34] M. Brooks, S. Amershi, B. Lee, S.M. Drucker, A. Kapoor, and P. Simard. "FeatureInsight: Visual support for error-driven feature ideation in text classification". In: *2015 IEEE Conference on Visual Analytics Science and Technology (VAST)*. 2015, pp. 105–112.

[35] Emily Brown and Paul Cairns. "A grounded investigation of game immersion". In: *Extended abstracts of the 2004 conference on Human factors and computing systems - CHI '04*. New York, New York, USA: ACM Press, 2004, p. 1297. ISBN: 1581137036. DOI: 10.1145/985921.986048. URL: http://portal.acm.org/citation.cfm?doid=985921.986048.

[36] Kevin Browne and Christopher Anand. "An empirical evaluation of user interfaces for a mobile video game". In: *Entertainment Computing* 3.1 (2012), 1–10. ISSN: 18759521. DOI: 10.1016/j.entcom.2011.06.001.

[37] Erik Brynjolfsson, John J Horton, Adam Ozimek, Daniel Rock, Garima Sharma, and Hong-Yi TuYe. "COVID-19 and remote work: an early look at US data". In: *National Bureau of Economic Research* (2020).

[38] Manuel Bullejos, David Cabezas, Manuel Martín-Martín, and Francisco Javier Alcalá. "A K-Nearest Neighbors Algorithm in Python for Visualizing the 3D Stratigraphic Architecture of the Llobregat River Delta in NE Spain". In: *Journal of Marine Science and Engineering* 10.7 (July 2022), p. 986. ISSN: 2077-1312. DOI: 10.3390/jmse10070986.

[39] G. Burdea and P. Coiffet. *Virtual reality technology*. 2nd. Wiley-IEEE Press, 2003. ISBN: 978-0-471-36089-6.

[40] Paul Cairns. "Engagement in Digital Games". In: *Why Engagement Matters: Cross-Disciplinary Perspectives of User Engagement in Digital Media*. Springer, 2016, 81–104.

[41] M. Cakmak and A.L. Thomaz. "Eliciting good teaching from humans for machine learners". In: *Artificial Intelligence* 217 (2014), pp. 198–215.

[42] Gordon Calleja. *In-Game: From Immersion to Incorporation*. Cambridge, MA, London: MIT Press, 2011.

[43] Gordon Calleja. “Incorporation: A Renewed Understanding of Presence and Immersion in Digital Games”. In: *DiGRA 2011*. DiGRA, 2011.

[44] Capcom. *Street Fighter II*. 1991.

[45] Baptiste Caramiaux, Frederic Bevilacqua, and Atau Tanaka. “Beyond recognition: Using gesture variation for continuous interaction”. In: *CHI’13 Extended Abstracts on Human Factors in Computing Systems*. 2013, pp. 2109–2118.

[46] Güven ÇATAK, Server MASALCI, and Seray ŞENYER. “A Guideline Study for Designing Virtual Reality Games”. In: *AJIT-e: Academic Journal of Information Technology* 11.43 (2020), pp. 12–36.

[47] Canan Ceylan, Jan Dul, and Serpil Aytac. “Can the office environment stimulate a manager’s creativity?” In: *Human Factors and Ergonomics in Manufacturing & Service Industries* 18.6 (Nov. 2008), pp. 589–602. ISSN: 1520-6564. DOI: 10.1002/HFM.20128.

[48] Darren Chappell, Virginia Eatough, Mark N. O. Davies, and Mark Griffiths. “EverQuest—It’s Just a Computer Game Right? An Interpretative Phenomenological Analysis of Online Gaming Addiction”. In: *International Journal of Mental Health and Addiction* 4.3 (July 2006), 205–216. ISSN: 1557-1874. DOI: 10.1007/s11469-006-9028-6. URL: http://link.springer.com/10.1007/s11469-006-9028-6.

[49] Katherine Compton. “Casual Creators: Defining a Genre of Autotelic Creativity Support Systems”. Copyright - Database copyright ProQuest LLC; ProQuest does not claim copyright in the individual underlying works; Last updated - 2023-06-21. PhD thesis. University of California Santa Cruz, 2019, p. 685. ISBN: 9781085668880. URL: https://www.proquest.com/dissertations-theses/casual-creators-defining-genre-autotelic/docview/2300563742/se-2.

[50] RAPIDMIX Consortium. *Rapid-Mix*. 2015. URL: http://rapidmix.goldsmithsdigital.com/.

[51] Mihaly Csikszentmihalyi. *Flow*. Harper & Row, 1990. ISBN: 0060162538.

[52] Mihaly Csikszentmihalyi and Jeanne Nakamura. “The concept of flow”. In: *The handbook of positive psychology*. Vol. 13. New York, NY, US: Oxford University Press, 2002, 89–105. ISBN: 0195135334. DOI: 10.1007/978-94-017-9088-8_16.

[53] Padraig Cunningham and Sarah Jane Delany. “k-Nearest neighbour classifiers”. In: *Multiple Classifier Systems* 34.8 (2007), pp. 1–17.

[54] Da Viking Code Studio. *PDollar Point-Cloud Gesture Recognizer Unity Asset*. 2014. URL: https://assetstore.unity.com/packages/tools/input-management/pdollar-point-cloud-gesture-recognizer-21660.

[55] Gustav Dahl and Martin Kraus. “Measuring how game feel is influenced by the player avatar’s acceleration and deceleration”. In: *Proceedings of the 19th International Academic Mindtrek Conference on - AcademicMindTrek ’15* (2015), 41–46. DOI: 10.1145/2818187.2818275. URL: http://dl.acm.org/citation.cfm?doid=2818187.2818275.

[56] N. Dahlbäck, A. Jönsson, and L. Ahrenberg. “Wizard of Oz Studies — Why and How”. In: *Knowledge-Based Systems* 6.4 (Dec. 1993), pp. 258–266. ISSN: 0950-7051. DOI: 10.1016/0950-7051(93)90017-N.

[57] Carlos Gonzalez Diaz, Phoenix Perry, and Rebecca Fiebrink. “Interactive Machine Learning for More Expressive Game Interactions”. In: *2019 IEEE Conference on Games (CoG)*. 2019, pp. 1–2. DOI: 10.1109/CIG.2019.8848007.

[58] James Diebel. “Representing attitude: Euler angles, unit quaternions, and rotation vectors”. In: *Matrix* 58.15-16 (2006), pp. 1–35.

[59] Tom Djajadiningrat, Ben Matthews, and Marcelle Stienstra. “Easy doesn’t do it: Skill and expression in tangible aesthetics”. In: *Personal and Ubiquitous Computing* 11.8 (2007), 657–676. ISSN: 16174909. DOI: 10.1007/s00779-006-0137-9.

[60] Sarah A. Douglas and Anant Kartik. Mithal. *The Ergonomics of Computer Pointing Devices*. New York, NY, USA: Springer-Verlag New York, Inc., 1997, p. 233. ISBN: 3-540-19986-1.

[61] Edward Downs and S. Shyam Sundar. “" We won" vs. "They lost": Exploring ego-enhancement and self-preservation tendencies in the context of video game play”. In: *Entertainment Computing* 2.1 (2011), pp. 23–28. ISSN: 18759521. DOI: 10.1016/j.entcom.2011.03.012.

[62] Romina Druta, Cristian Druta, Paul Negirla, and Ioan Silea. “A Review on Methods and Systems for Remote Collaboration”. In: *Applied Sciences* 11.21 (2021). ISSN: 2076-3417. DOI: 10.3390/app112110035.

[63] J.J. Dudley and P.O. Kristensson. “A Review of User Interface Design for Interactive Machine Learning”. In: *ACM Transactions on Interactive Intelligent Systems* 8.2 (2018), pp. 1–37.

[64] Theston E. Fox. *Virtual Reality Toolkit (VRTK)*. 2016. URL: https://github.com/ExtendRealityLtd/VRTK.

[65] *ELAN*. 2020. URL: https://archive.mpi.nl/tla/elan.

[66] Epic Games. *Blueprint Overview*. 2014. URL: https://docs.unrealengine.com/4.27/en-US/ProgrammingAndScripting/Blueprints/Overview/ (visited on 03/10/2024).

[67] Eva Eriksson, Thomas Riisgaard Hansen, and Andreas Lykke-Olesen. “Movement-Based Interaction in Camera Spaces: A Conceptual Framework”. In: *Personal and Ubiquitous Computing* 11.8 (Dec. 2007), pp. 621–632. ISSN: 1617-4909. DOI: 10.1007/s00779-006-0134-z.

[68] J.A. Fails and D.R. Olsen. “Interactive machine learning”. In: *Proceedings of the 8th international conference on Intelligent user interfaces - IUI ’03*. 2003, p. 39.

[69] Xiaowen Fang, Jingli Zhang, and Susy S. Chan. “Development of an Instrument for Studying Flow in Computer Game Play”. In: *International Journal of Human-Computer Interaction* 29.7 (July 2013), 456–470. DOI: 10.1080/10447318.2012.715991. URL: http://www.tandfonline.com/doi/abs/10.1080/10447318.2012.715991.

[70] Martin Fasterholdt, Martin Pichlmair, and Christoffer Holmgard. “You Say Jump, I Say How High? Operationalising the Game Feel of Jumping”. In: *Proceedings 1st International Joint Conference DiGRA and FDG* (2016), 1–16.

[71] William M. Felton and Russell E. Jackson. “Presence: A Review”. In: *International Journal of Human-Computer Interaction* 38.1 (2021), pp. 1–18. ISSN: 15327590. DOI: 10.1080/10447318.2021.1921368. URL: https://www.tandfonline.com/doi/abs/10.1080/10447318.2021.1921368.

[72] João P. Ferreira, Thiago M. Coutinho, Thiago L. Gomes, José F. Neto, Rafael Azevedo, Renato Martins, and Erickson R. Nascimento. “Learning to dance: A graph convolutional adversarial network to generate realistic dance motions from audio”. In: *Computers & Graphics* 94 (2021), pp. 11–21. ISSN: 0097-8493. DOI: https://doi.org/10.1016/j.cag.2020.09.009. URL: https://www.sciencedirect.com/science/article/pii/S0097849320301436.

[73] R. Fiebrink, P.R. Cook, and D. Trueman. “Human model evaluation in interactive supervised learning”. In: *Proceedings of the 2011 annual conference on Human factors in computing systems - CHI ’11*. 2011, p. 147.

[74] R.A. Fiebrink. “Real-time Human Interaction with Supervised Learning Algorithms for Music Composition and Performance”. PhD thesis. Princeton University, 2011.

[75] Rebecca Fiebrink. “Machine learning as meta-instrument: Human-machine partnerships shaping expressive instrumental creation”. In: *Musical Instruments in the 21st Century*. Springer, 2017, pp. 137–151.

[76] Rebecca Fiebrink, Dan Trueman, and Perry R Cook. “A Meta-Instrument for Interactive, On-the-Fly Machine Learning.” In: *Proc. of the International Conference on New Interfaces for Musical Expression*. 2009, pp. 280–285.

[77] Michael Filsecker and Daniel Thomas Hickey. “A multilevel analysis of the effects of external rewards on elementary students’ motivation, engagement and learning in an educational game”. In: *Computers & Education* 75 (June 2014), 136–148. ISSN: 03601315. DOI: 10.1016/j.compedu.2014.02.008. URL: http://linkinghub.elsevier.com/retrieve/pii/S0360131514000426.

[78] Maiken Hillerup Fogtmann, Jonas Fritsch, and Karen Johanne Kortbek. “Kinesthetic Interaction: Revealing the Bodily Potential in Interaction Design”. In: *Proceedings of the 20th Australasian Conference on Computer-Human Interaction: Designing for Habitus and Habitat*. OZCHI ’08. Cairns, Australia: Association for Computing Machinery, 2008, pp. 89–96. ISBN: 0980306345. DOI: 10.1145/1517744.1517770.

[79] Stephen R. Foster, William G. Griswold, and Sorin Lerner. “WitchDoctor: IDE support for real-time auto-completion of refactorings”. In: *2012 34th International Conference on Software Engineering (ICSE)*. 2012, pp. 222–232. DOI: 10.1109/ICSE.2012.6227191.

[80] J. Françoise, F. Bevilacqua, and T. Schiphorst. “GaussBox: Prototyping Movement Interaction with Interactive Visualizations of Machine Learning”. In: *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems - CHI EA ’16*. 2016, pp. 3667–3670.

[81] Jules Françoise and Frédéric Bevilacqua. “Motion-Sound Mapping through Interaction: An Approach to User-Centered Design of Auditory Feedback Using Machine Learning”. In: *ACM Transactions on Interactive Intelligent Systems* 8.2 (June 2018). ISSN: 2160-6455. DOI: 10.1145/3211826.

[82] Jules Françoise, Yves Candau, Sarah Fdili Alaoui, and Thecla Schiphorst. “Designing for Kinesthetic Awareness: Revealing User Experiences through Second-Person Inquiry”. In: *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. CHI ’17. Denver, Colorado, USA: Association for Computing Machinery, 2017, pp. 5171–5183. ISBN: 9781450346559. DOI: 10.1145/3025453.3025714.

[83] Vittoria Frau, Lucio Davide Spano, Valentino Artizzu, and Michael Nebeling. “XRSpotlight: Example-Based Programming of XR Interactions Using a Rule-Based Approach”. In: *Proceedings of ACM Human-Computer Interaction* 7.EICS (June 2023). DOI: 10.1145/3593237.

[84] Brian J Gajadhar, Henk Herman Nap, Yvonne a W De Kort, Wijnand a Ijsselsteijn, B J Gajadhar, H H Nap, Y a W Kort, and W a Ijsselsteijn. “Out of Sight , out of Mind : Co-Player Effects on Seniors ’ Player Experience”. In: *Proceedings of the Fun and Games Conference* (2010), 74–83. ISSN: 17582229. DOI: 10.1145/1823818.1823826.

[85] Gamasutra. *Gamasutra - Microsoft launches Xbox Adaptive Controller to make games more accessible*. 2018. URL: https://www.gamasutra.com/view/news/318288/Microsoft_launches_Xbox_Adaptive_Controller_to_make_games_more_accessible.php.

[86] M.W Gardner and S.R Dorling. “Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences”. In: *Atmospheric Environment* 32.14 (1998), pp. 2627–2636. ISSN: 1352-2310. DOI: https://doi.org/10.1016/S1352-2310(97)00447-0. URL: https://www.sciencedirect.com/science/article/pii/S1352231097004470.

[87] William W. Gaver. “The Affordances of Media Spaces for Collaboration”. In: *Proceedings of the 1992 ACM Conference on Computer-Supported Cooperative Work*. CSCW ’92. Toronto, Ontario, Canada: Association for Computing Machinery, 1992, pp. 17–24. ISBN: 0897915429. DOI: 10.1145/143457.371596.

[88] K.M. Gerling, R.L. Mandryk, M. Miller, M.R. Kalyn, M. Birk, and J.D. Smeddinck. “Designing Wheelchair-Based Movement Games”. In: *ACM Transactions on Accessible Computing* 6.2 (2015), pp. 1–23.

[89] Kathrin Gerling, Ian Livingston, Lennart Nacke, and Regan Mandryk. “Full-Body Motion-Based Game Interaction for Older Adults”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI ’12. Austin, Texas, USA: Association for Computing Machinery, 2012, pp. 1873–1882. ISBN: 9781450310154. DOI: 10.1145/2207676.2208324.

[90] Nicholas Gillian and Joseph A Paradiso. “The Gesture Recognition Toolkit”. In: *The Journal of Machine Learning Research* 15.1 (2014), pp. 3483–3487.

[91] M. Gillies, H. Brenton, and A. Kleinsmith. “Embodied design of full bodied interaction with virtual humans”. In: *Proceedings of the 2nd International Workshop on Movement and Computing - MOCO ’15*. 2015, pp. 1–8.

[92] M. Gillies, B. Lee, N. D’Alessandro, J. Tilmanne, T. Kulesza, B. Caramiaux, R. Fiebrink, A. Tanaka, J. Garcia, F. Bevilacqua, A. Heloir, F. Nunnari, W. Mackay, and S. Amershi. “Human-Centred Machine Learning”. In: *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems - CHI EA ’16*. 2016, pp. 3558–3565.

[93] Marco Gillies. “Understanding the Role of Interactive Machine Learning in Movement Interaction Design”. In: *ACM Transactions on Computer-Human Interaction* 26.1 (Feb. 2019), pp. 1–34. DOI: 10.1145/3287307.

[94] Marco Gillies and Andrea Kleinsmith. “Non-representational interaction design”. In: *Contemporary Sensorimotor Theory*. Springer. 2014, pp. 201–208.

[95] Marco Gillies, Andrea Kleinsmith, and Harry Brenton. “Applying the CASSM Framework to Improving End User Debugging of Interactive Machine Learning”. In: *Proceedings of the 20th International Conference on Intelligent User Interfaces - IUI ’15*. New York, New York, USA: ACM Press, 2015, pp. 181–185. ISBN: 9781450333061. DOI: 10.1145/2678025.2701373.

[96] Vlad Petre Glăveanu. “Creativity and Sociality”. In: *Distributed Creativity, Thinking Outside the Box of the Creative Individual*. 1st ed. Springer, Cham, 2014, pp. 33–48. DOI: 10.1007/978-3-319-05434-6_3.

[97] Mario Gollwitzer and André Melzer. “Macbeth and the Joystick: Evidence for moral cleansing after playing a violent video game”. In: *Journal of Experimental Social Psychology* 48.6 (2012), 1356–1360. ISSN: 00221031. DOI: 10.1016/j.jesp.2012.07.001.

[98] Arlindo Gomes, Lucas Figueiredo, Walter Correia, Veronica Teichrieb, Jonysberg Quintino, Fabio Q. B. da Silva, André Santos, and Helder Pinho. “Extended by Design: A Toolkit for Creation of XR Experiences”. In: *2020 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct)*. 2020, pp. 57–62. DOI: 10.1109/ISMAR-Adjunct51615.2020.00029.

[99] Carlos Gonzalez Diaz. “Interactive Machine Learning for Games Controls Customisation and More Expressive Interactions”. Develop:Brighton Conference 2019. 2019. URL: https://www.iggi-phd.org/news/iggi-students-and-staff-2019-iggi-conference.

[100] Carlos Gonzalez Diaz. “Interactive Machine Learning for More Expressive Game Interactions”. IGGI Conference 2019. 2019. URL: https://youtu.be/kkKU3MyBspM.

[101] Carlos Gonzalez Diaz. “InteractML for VR Games Interactions”. Games Week Berlin 2020. 2020. URL: https://youtu.be/Vfcbm6OfBGQ.

[102] Carlos Gonzalez Diaz, Nicola Plant, Clarice Hilton, Michael Zbyszyński, Rebecca Fiebrink, Phoenix Perry, Ruth Gibson, Bruno Martelli, Sebastian Deterding, and Marco Gilles. “Bodystorming in SocialVR to Support Collaborative Embodied Ideation”. In: *CHI 2021 Workshop on Social VR*. New York, NY, USA: ACM, 2021, p. 3.

[103] Jason Gregory. “Collision and Rigid Body Dynamics”. In: *Game Engine Architecture*. Ed. by Jeff Lander and Matt Whiting. Taylor & Francis Ltd., 2009. Chap. 12, pp. 595–685. ISBN: 9781439865262.

[104] Jason Gregory. *Game engine architecture*. 1st ed. Taylor & Francis Ltd., 2009.

[105] Tovi Grossman, George Fitzmaurice, and Ramtin Attar. “A Survey of Software Learnability: Metrics, Methodologies and Guidelines”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI ’09. Boston, MA, USA: Association for Computing Machinery, 2009, pp. 649–658. ISBN: 9781605582467. DOI: 10.1145/1518701.1518803.

[106] Matthew Guzdial, Nicholas Liao, Jonathan Chen, Shao Yu Chen, Shukan Shah, Vishwa Shah, Joshua Reno, Gillian Smith, and Mark O. Riedl. “Friend, collaborator, student, manager: How design of an AI-driven game level editor affects creators”. In: *Conference on Human Factors in Computing Systems - Proceedings* (May 2019). DOI: 10.1145/3290605.3300854. arXiv: 1901.06417.

[107] Minna Hara and Saila Ovaska. “Heuristics for Motion-Based Control in Games”. In: *Proceedings of the 8th Nordic Conference on Human-Computer Interaction: Fun, Fast, Foundational*. NordiCHI ’14. Helsinki, Finland: Association for Computing Machinery, 2014, pp. 697–706. ISBN: 9781450325424. DOI: 10.1145/2639189.2639246.

[108] Tilo Hartmann and Peter Vorderer. “It’s okay to shoot a character: Moral disengagement in violent video games”. In: *Journal of Communication* 60.1 (2010), 94–119. ISSN: 00219916. DOI: 10.1111/j.1460-2466.2009.01459.x.

[109] Marc Hassenzahl. “The Thing and I: Understanding the Relationship Between User and Product”. In: *Funology 2: From Usability to Enjoyment*. Springer, Dordrecht, 2003, 31–42. DOI: 10.1007/1-4020-2967-5_4.

[110] ROBERT HECHT-NIELSEN. “III.3 - Theory of the Backpropagation Neural Network**Based on “nonindent” by Robert Hecht-Nielsen, which appeared in Proceedings of the International Joint Conference on Neural Networks 1, 593–611, June 1989. © 1989 IEEE.” In: *Neural Networks for Perception*. Ed. by Harry Wechsler. Academic Press, 1992, pp. 65–93. ISBN: 978-0-12-741252-8. DOI: https://doi.org/10.1016/B978-0-12-741252-8.50010-8. URL: https://www.sciencedirect.com/science/article/pii/B9780127412528500108.

[111] C Heeter, C Sarkar, B Palmer-Scott, and S Zhang. “Engineering sociability: Friendship drive, visibility, and social connection in anonymous co-located local Wi-Fi multiplayer online gaming”. In: *International Journal of Gaming and Computer-Mediated Simulations* 4.2 (2012), 1–18. ISSN: 19423888. DOI: 10.4018/jgcms.2012040101.

[112] Monique M. Hennink, Bonnie N. Kaiser, and Vincent C. Marconi. “Code Saturation Versus Meaning Saturation: How Many Interviews Are Enough?” In: *Qualitative Health Research* 27.4 (Mar. 2017), pp. 591–608. ISSN: 15527557. DOI: 10.1177/1049732316665344. URL: https://journals.sagepub.com/doi/10.1177/1049732316665344.

[113] Clarice Hilton, Nicola Plant, Carlos Gonzalez Diaz, Phoenix Perry, Ruth Gibson, Bruno Martelli, Michael Zbyszyński, Rebecca Fiebrink, and Marco Gillies. “InteractML: Making machine learning accessible for creative practitioners working with movement interaction in immersive media”. In: *Proceedings of the 27th ACM Symposium on Virtual Reality Software and Technology*. NY, USA: ACM, 2021. ISBN: 9781450390927. DOI: 10.1145/3489849.

[114] Ken Hinckley, Robert J.K. Jacob, Colin Ware, Jacob O. Wobbrock, and Daniel Wigdor. “Input/Output Devices and Interaction Techniques”. In: *Computing Handbook, Third Edition*. Chapman and Hall/CRC, May 2014. Chap. 21, pp. 1–54. DOI: 10.1201/b16812-25. URL: http://www.crcnetbase.com/doi/abs/10.1201/b16812-25.

[115] Wouter van den Hoogen, Karolien Poels, Wijnand IJsselsteijn, and Yvonne de Kort. “Between Challenge and Defeat: Repeated Player-Death and Game Enjoyment”. In: *Media Psychology* 15.4 (2012), pp. 443–459. ISSN: 15213269. DOI: 10.1080/15213269.2012.723117.

[116] Kristina Hook. *Designing with the body: Somaesthetic interaction design*. Mit Press, 2018.

[117] Kristina Höök, Anna Ståhl, Martin Jonsson, Johanna Mercurio, Anna Karlsson, and Eva Carin Banka Johnson. "Somaesthetic design". In: *Interactions* 22.4 (July 1, 2015), pp. 26–33. ISSN: 15583449. DOI: 10.1145/2770888.

[118] Eva Hornecker and Jacob Buur. "Getting a Grip on Tangible Interaction: A Framework on Physical Space and Social Interaction". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '06. Montréal, Québec, Canada: Association for Computing Machinery, 2006, pp. 437–446. ISBN: 1595933727. DOI: 10.1145/1124772.1124838.

[119] HTC and Valve Corporation. *HTC Vive Controller*. 2014. URL: https://www.vive.com/uk/accessory/controller/.

[120] Blair Hu, Ann M. Simon, and Levi Hargrove. "Deep Generative Models With Data Augmentation to Learn Robust Representations of Movement Intention for Powered Leg Prostheses". In: *IEEE Transactions on Medical Robotics and Bionics* 1.4 (2019), pp. 267–278. DOI: 10.1109/TMRB.2019.2952148.

[121] Caroline Hummels, Kees C. Overbeeke, and Sietske Klooster. "Move to Get Moved: A Search for Methods, Tools and Knowledge to Design for Expressive and Rich Movement-Based Interaction". In: *Personal and Ubiquitous Computing* 11.8 (Dec. 2007), pp. 677–690. ISSN: 1617-4909. DOI: 10.1007/s00779-006-0135-y.

[122] F. Iacopetti, L. Fanucci, R. Roncella, D. Giusti, and A. Scebba. "Game Console Controller Interface for People with Disability". In: *2008 International Conference on Complex, Intelligent and Software Intensive Systems*. 2008, pp. 757–762.

[123] Wijnand Ijsselsteijn, Wouter Van Den Hoogen, Christoph Klimmt, Yvonne De Kort, Craig Lindley, Klaus Mathiak, Karolien Poels, Niklas Ravaja, Marko Turpeinen, and Peter Vorderer. "Measuring the Experience of Digital Game Enjoyment". In: *Proceeding of Measuring Behavior, 6th International Conference on Methods and Techniques in Behavioral Research 2008*. Maastricht, The Netherlands: Wageningen: Noldus Information Technology, 2008, 88–89.

[124] Wijnand IJsselsteijn, tuenl Yvonne de Kort, tuenl Karolien Poels, Audrius Jurgelionis, and Francesco Bellotti. "Characterising and Measuring User Experiences in Digital Games". In: *ACE'07* (2007).

[125] Katherine Isbister. "Emotion and motion: games as inspiration for shaping the future of interface". In: *Interactions* 785 (2011), 24–27. ISSN: 1072-5520. DOI: http://doi.acm.org/10.1145/2008176.2008184. URL: http://katherineinterface.com/isbisterinteractions.pdf.

[126] Katherine Isbister and Florian Mueller. "Guidelines for the design of movement-based games and their relevance to HCI". In: *Human-Computer Interaction* 30.3-4 (May 2015), pp. 366–399. ISSN: 07370024. DOI: 10.1080/07370024.2014.996647.

[127] Katherine Isbister, Rahul Rao, Ulf Schwekendiek, Elizabeth Hayward, and Jessamyn Lidasan. “Is More Movement Better? A Controlled Comparison of Movement-based Games”. In: *Proceedings of the 6th International Conference on Foundations of Digital Games*. New York: ACM Press, 2011, 331–333. ISBN: 9781450308045. DOI: 10.1145/2159365.2159429.

[128] Katherine Isbister, Ulf Schwekendiek, and Jonathan Frye. “Wriggle”. In: *Proceedings of the 2011 annual conference extended abstracts on Human factors in computing systems - CHI EA ’11*. New York, New York, USA: ACM Press, 2011, p. 1885. ISBN: 9781450302685. DOI: 10.1145/1979742.1979919. URL: http://portal.acm.org/citation.cfm?doid=1979742.1979919.

[129] Alan Jack. *8 & 1/2 Simple Rules for Designing My Motion Control Game*. 2011. URL: https://www.gamedeveloper.com/design/8-1-2-simple-rules-for-designing-my-motion-control-game (visited on 06/27/2023).

[130] Kalle Jegers. “Pervasive game flow”. In: *Computers in Entertainment* 5.1 (Jan. 2007), p. 9. ISSN: 15443574. DOI: 10.1145/1236224.1236238.

[131] Charlene Jennett, Anna L. Cox, Paul Cairns, Samira Dhoparee, Andrew Epps, Tim Tijs, and Alison Walton. “Measuring and defining the experience of immersion in games”. In: *International Journal of Human-Computer Studies* 66.9 (Sept. 2008), 641–661. ISSN: 10715819. DOI: 10.1016/j.ijhcs.2008.04.004. URL: http://linkinghub.elsevier.com/retrieve/pii/S1071581908000499.

[132] Mads Møller Jensen, Sarah Kristin Thiel, Eve Hoggan, and Susanne Bødker. “Physical Versus Digital Sticky Notes in Collaborative Ideation”. In: *Computer Supported Cooperative Work: CSCW: An International Journal* 27.3-6 (Dec. 1, 2018), pp. 609–645. ISSN: 15737551. DOI: 10.1007/s10606-018-9325-1.

[133] Seung A Annie Jin. ““Toward Integrative Models of Flow”: Effects of Performance, Skill, Challenge, Playfulness, and Presence on Flow in Video Games”. In: *Journal of Broadcasting and Electronic Media* 56.2 (2012), 169–186. ISSN: 08838151. DOI: 10.1080/08838151.2012.678516.

[134] Seung-A Annie Jin. ““I Feel Present. Therefore, I Experience Flow:” A Structural Equation Modeling Approach to Flow and Presence in Video Games”. In: *Journal of Broadcasting & Electronic Media* 55.1 (2011), 114–136.

[135] Kristiina Jokinen, Hirohisa Furukawa, Masafumi Nishida, and Seiichi Yamamoto. “Gaze and turn-taking behavior in casual conversational interactions”. In: *ACM Transactions on Interactive Intelligent Systems* 3.2 (July 2013), pp. 1–30. ISSN: 21606463. DOI: 10.1145/2499474.2499481.

[136] Brigitte Jordan and Austin Henderson. “Interaction Analysis: Foundations and Practice”. In: *The Journal of the Learning Sciences* 4.1 (1995), pp. 39–103.

[137] Andreas Jungherr and Damien B. Schlarb. “The Extended Reach of Game Engine Companies: How Companies Like Epic Games and Unity Technologies Provide Platforms for Extended Reality Applications and the Metaverse”. In: *Social Media + Society* 8.2 (2022), p. 20563051221107641. DOI: 10.1177/20563051221107641.

[138] Christopher M. Kanode and Hisham M. Haddad. “Software Engineering Challenges in Game Development”. In: *2009 Sixth International Conference on Information Technology: New Generations*. 2009, pp. 260–265. DOI: 10.1109/ITNG.2009.74.

[139] Amy K. Karlson, Shamsi T. Iqbal, Brian Meyers, Gonzalo Ramos, Kathy Lee, and John C. Tang. “Mobile Taskflow in Context: A Screenshot Study of Smartphone Usage”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI ’10. Atlanta, Georgia, USA: Association for Computing Machinery, 2010, pp. 2009–2018. ISBN: 9781605589299. DOI: 10.1145/1753326.1753631.

[140] Fakhreddine Karray, Milad Alemzadeh, Jamil Abou Saleh, and Mo Nours Arab. “Human-Computer Interaction: Overview on State of the Art”. In: *International Journal on Smart Sensing and Intelligent Systems* 1.1 (2008), pp. 137–159. ISSN: 1178-5608.

[141] S. Katan, M. Grierson, and R. Fiebrink. “Using Interactive Machine Learning to Support Interface Development Through Workshops with Disabled People”. In: *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems - CHI ’15*. 2015, pp. 251–254.

[142] Konstantina Kilteni, Raphaela Groten, and Mel Slater. “The sense of embodiment in virtual reality”. In: *Presence: Teleoperators and Virtual Environments* 21.4 (2012), pp. 373–387.

[143] Joonhwan Kim, Sung Park, Marc Hassenzahl, and Kai Eckoldt. “The Essence of Enjoyable Experiences: The Human Needs”. In: *International Conference of Design, User Experience, and Usability*. Springer, Berlin, Heidelberg, 2011, 77–83. DOI: 10.1007/978-3-642-21675-6_9.

[144] Kwanguk Kim and Peter Mundy. “Joint attention, social-cognition, and recognition memory in adults”. In: *Frontiers in Human Neuroscience* 0.JUNE 2012 (June 2012), p. 172. ISSN: 16625161. DOI: 10.3389/FNHUM.2012.00172/BIBTEX.

[145] David Kirsh. “Embodied Cognition and the Magical Future of Interaction Design”. In: *ACM Transactions on Computer-Human Interaction* 20.1 (Apr. 2013). ISSN: 1073-0516. DOI: 10.1145/2442106.2442109.

[146] Martin Klasen, René Weber, Tilo T.J. Kircher, Krystyna A. Mathiak, and Klaus Mathiak. “Neural contributions to flow experience during video game playing”. In: *Social Cognitive and Affective Neuroscience* 7.4 (2012), 485–495. ISSN: 17495016. DOI: 10.1093/scan/nsr021.

[147] A. Kleinsmith and M. Gillies. “Customizing by doing for responsive video game characters”. In: *International Journal of Human-Computer Studies* 71.7-8 (2013), pp. 775–784.

[148] Scott R Klemmer, Björn Hartmann, and Leila Takayama. “How Bodies Matter: Five Themes for Interaction Design”. In: *Proceedings of the 6th conference on Designing Interactive systems*. New York, NY, USA: ACM, June 2006, pp. 140–149. ISBN: 1595933670. DOI: 10.1145/1142405.1142429. URL: https://dl.acm.org/doi/10.1145/1142405.1142429.

[149] S. Klooster and C.J. Overbeeke. “Designing products as an integral part of choreography of interaction : the product’s form as an integral part of movement”. English. In: *Design and Semantics of Form and Movement (DESFORM 2005)*. Ed. by L. Feijs, S. Kyffin, and B. Young. 1st European Workshop on Design and Semantics of Form and Movement (DeSForM 2005), November 11, 2005, Newcastle upon Tyne, UK, DeSForM 2005. Eindhoven University of Technology, 2005, pp. 23–35. URL: http://www.semantics.id.tue.nl/.

[150] T. Kulesza, S. Amershi, R. Caruana, D. Fisher, and D. Charles. “Structured labeling for facilitating concept evolution in machine learning”. In: *Proceedings of the 32nd annual ACM conference on Human factors in computing systems - CHI ’14*. 2014, pp. 3075–3084.

[151] Roberta Kwok. “Deep Learning Powers a motion-tracking revolution”. In: *Nature Toolbox* 574 (Sept. 2019), pp. 137–138. DOI: 10.1038/d41586-019-02942-5.

[152] Gorm Lai, William Latham, and Frederic Fol Leymarie. “Towards Friendly Mixed Initiative Procedural Content Generation: Three Pillars of Industry”. In: *Proceedings of the 15th International Conference on the Foundations of Digital Games*. FDG ’20. Bugibba, Malta: Association for Computing Machinery, 2020. ISBN: 9781450388078. DOI: 10.1145/3402942.3402946.

[153] Nicole Lazzaro. “Chapter 20 - The Four Fun Keys”. In: *Game Usability*. Ed. by Katherine Isbister and Noah Schaffer. Boston: Morgan Kaufmann, 2008, pp. 317–343. ISBN: 978-0-12-374447-0. DOI: https://doi.org/10.1016/B978-0-12-374447-0.00020-2. URL: https://www.sciencedirect.com/science/article/pii/B9780123744470000202.

[154] Eva Lenz, Sarah Diefenbach, and Marc Hassenzahl. “Aesthetics of interaction: a literature synthesis”. In: *NordiCHI’14*. New York: ACM Press, 2014, 628–637. ISBN: 9781450325424. URL: http://dl.acm.org/citation.cfm?id=2639198.

[155] David Lewis. “Analog and Digital”. In: *Noûs* 5.3 (Sept. 1971), p. 321. ISSN: 00294624. DOI: 10.2307/2214671. URL: http://www.jstor.org/stable/2214671?origin=crossref.

[156] Hongjian Liao and Zhe Qu. “Virtual experiment system for electrician training based on Kinect and Unity3D”. In: *Proceedings 2013 International Conference on Mechatronic Sciences, Electric Engineering and Computer (MEC)*. 2013, pp. 2659–2662. DOI: 10.1109/MEC.2013.6885480.

[157] Anthony M. Limperos, Michael G. Schmierbach, Andrew D. Kegerise, and Frank E. Dardis. "Gaming Across Different Consoles: Exploring the Influence of Control Scheme on Game-Player Enjoyment". In: *Cyberpsychology, Behavior, and Social Networking* 14.6 (June 2011), 345–350. ISSN: 2152-2715. DOI: 10.1089/cyber.2010.0146.

[158] Shu-Fang Lin. "Gender Differences and the Effect of Contextual Features on Game Enjoyment and Responses". In: *Cyberpsychology, Behavior, and Social Networking* 13.5 (2010), 533–537. ISSN: 2152-2715. DOI: 10.1089/cyber.2009.0293.

[159] Erik Lindholm, Mark J. Kilgard, and Henry Moreton. "A User-Programmable Vertex Engine". In: *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '01. New York, NY, USA: Association for Computing Machinery, 2001, pp. 149–158. ISBN: 158113374X. DOI: 10.1145/383259.383274.

[160] S Lindley, J Le Couteur, and N Bianchi-Berthouze. "Stirring up Experience through Movement in Game Play: Effects on Engagement and Social Behaviour". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 2008, pp. 511–514. ISBN: 9781605580111. DOI: 10.1145/1357054.1357136.

[161] Lian Loke and Toni Robertson. "Moving and Making Strange: An Embodied Approach to Movement-Based Interaction Design". In: *ACM Transactions on Computer-Human Interaction* 20.1 (Apr. 2013). ISSN: 1073-0516. DOI: 10.1145/2442106.2442113.

[162] Jonas Löwgren. "Toward an articulation of interaction aesthetics". In: *The New Review of Hypermedia and Multimedia* 15.2 (2009), pp. 129–146.

[163] H. Lü, J.A. Fogarty, and Y. Li. "Gesture script". In: *Proceedings of the 32nd annual ACM conference on Human factors in computing systems - CHI '14*. 2014, pp. 1685–1694.

[164] Elizabeth J Lyons, Deborah F Tate, Dianne S Ward, J Michael Bowling, Kurt M Ribisl, and Sriram Kalyararaman. "Energy Expenditure and Enjoyment during Video Game Play". In: *Medicine & Science in Sports & Exercise* 43.10 (Feb. 2011), p. 1. ISSN: 0195-9131. DOI: 10.1249/MSS.0b013e318216ebf3.

[165] William H. Macey and Benjamin Schneider. "The Meaning of Employee Engagement". In: *Industrial and Organizational Psychology* 1.1 (Mar. 2008), 3–30. ISSN: 1754-9426. DOI: 10.1111/j.1754-9434.2007.0002.x. URL: http://doi.wiley.com/10.1111/j.1754-9434.2007.0002.x.

[166] Matthieu Macret, Alissa N. Antle, and Philippe Pasquier. "Can a paper-based sketching interface improve the gamer experience in strategy computer games?" In: *4th International Conference on Intelligent Human Computer Interaction: Advancing Technology for Humanity, IHCI 2012*. 2012. ISBN: 9781467343695. DOI: 10.1109/IHCI.2012.6481817.

[167] Yukiko Maeda and So Yoon Yoon. “A meta-analysis on gender differences in mental rotation ability measured by the Purdue spatial visualization tests: Visualization of rotations (PSVT: R)”. In: *Educational Psychology Review* 25 (2013), pp. 69–94. DOI: 10.1007/s10648-012-9215-x.

[168] George Maestri. *Digital Character Animation 2: Essential Techniques*. Pearson Education, 1999. ISBN: 1562059300.

[169] Paul P Maglio and David Kirsh. “Epistemic action increases with skill”. In: *Proceedings of the eighteenth annual conference of the cognitive science society*. Routledge. 1996, pp. 391–396.

[170] S. Mailles-Viard Metz, P. Marin, and E. Vayre. “The shared online whiteboard: An assistance tool to synchronous collaborative design”. In: *European Review of Applied Psychology* 65.5 (2015), pp. 253–265. ISSN: 1162-9088. DOI: https://doi.org/10.1016/j.erap.2015.08.001.

[171] Charalampos Mainemelis and Sarah Ronson. “Ideas are Born in Fields of Play: Towards a Theory of Play and Creativity in Organizational Settings”. In: *Research in Organizational Behavior* 27 (2006), pp. 81–131. ISSN: 01913085. DOI: 10.1016/S0191-3085(06)27003-5.

[172] Mary Mainsbridge. *Body as Instrument: Performing with Gestural Systems in Live Electronic Music*. Bloomsbury Publishing USA, 2022.

[173] John Maloney, Mitchel Resnick, Natalie Rusk, Brian Silverman, and Evelyn Eastmond. “The Scratch Programming Language and Environment”. In: *ACM Transactions on Computer Education* 10.4 (Nov. 2010). DOI: 10.1145/1868358.1868363.

[174] Elena Márquez Segura, Laia Turmo Vidal, Asreen Rostami, and Annika Waern. “Embodied Sketching”. In: *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. CHI ’16. San Jose, California, USA: Association for Computing Machinery, 2016, pp. 6014–6027. ISBN: 9781450333627. DOI: 10.1145/2858036.2858486.

[175] Paul Marshall, Alissa Antle, Elise Van Den Hoven, and Yvonne Rogers. “Introduction to the Special Issue on the Theory and Practice of Embodied Interaction in HCI and Interaction Design”. In: *ACM Transactions on Computer-Human Interaction* 20.1 (Apr. 2013). ISSN: 1073-0516. DOI: 10.1145/2442106.2442107.

[176] MARUI-Plugin Inc. *MiVRy Unity Plugin*. 2019. URL: https://www.marui-plugin.com/mivry/.

[177] Louis McCallum and Rebecca Fiebrink. “Supporting Feature Engineering in End-User Machine Learning”. In: *CHI 2019 Workshop on Emerging Perspectives in Human-Centered Machine Learning*. ACM, Mar. 2019. URL: https://research.gold.ac.uk/id/eprint/26117/.

[178] Louis Mccallum and Rebecca Fiebrink. “The Challenge of Feature Engineering in Programming for Moving Bodies”. In: *End-User Programming for Moving Bodies*. 2020.

[179] Janetta Mitchell Mccoy and Gary W Evans. “The Potential Role of the Physical Environment in Fostering Creativity”. In: *Creativity Research Journal* 14 (2002), pp. 409–426. ISSN: 1532-6934. DOI: 10.1207/S15326934CRJ1434_11.

[180] Mitchell W. McEwan, Alethea L. Blackler, Daniel M. Johnson, and Peta A. Wyeth. “Natural mapping and intuitive interaction in videogames”. In: *Proceedings of the first ACM SIGCHI annual symposium on Computer-human interaction in play - CHI PLAY ’14*. New York, New York, USA: ACM Press, 2014, pp. 191–200. ISBN: 9781450330145. DOI: 10.1145/2658537.2658541.

[181] Rory McGloin, Kirstie Farrar, and Marina Krcmar. “Video Games, Immersion, and Cognitive Aggression: Does the Controller Matter?” In: *Media Psychology* 16.1 (2013), 65–87. ISSN: 1521-3269. DOI: 10.1080/15213269.2012.752428. URL: http://www.tandfonline.com/doi/abs/10.1080/15213269.2012.752428.

[182] Joshua McVeigh-Schultz and Katherine Isbister. “A “beyond being there” for VR meetings: envisioning the future of remote work”. In: *Human–Computer Interaction* 0.0 (2021), pp. 1–21. DOI: 10.1080/07370024.2021.1994860. eprint: https://doi.org/10.1080/07370024.2021.1994860.

[183] Elisa D. Mekler, Julia Ayumi Bopp, Alexandre N. Tuch, and Klaus Opwis. “A systematic review of quantitative studies on the enjoyment of digital entertainment games”. In: *Proceedings of the 32nd annual ACM conference on Human factors in computing systems - CHI ’14*. New York, New York, USA: ACM Press, 2014, 927–936. ISBN: 9781450324731. DOI: 10.1145/2556288.2557078.

[184] Maurice Merleau-Ponty. *The visible and the invisible: Followed by working notes*. Northwestern University Press, 1968.

[185] Maurice Merleau-Ponty and Colin Smith. *Phenomenology of perception*. Vol. 2012. Routledge London, 1962.

[186] Meta. *Meta Quest VR Headsets*. 2023. URL: https://www.meta.com/quest.

[187] Microsoft. *Kinect*. 2010. URL: https://en.wikipedia.org/wiki/Kinect (visited on 02/02/2016).

[188] Microsoft. *Mixed Reality Toolkit*. 2017. URL: https://github.com/microsoft/MixedRealityToolkit-Unity.

[189] Microsoft. *Reflection and Attributes*. Microsoft .NET Documentation. 2023. URL: https://learn.microsoft.com/en-us/dotnet/csharp/advanced-topics/reflection-and-attributes/.

[190] Thomas B. Moeslund and Erik Granum. “A Survey of Computer Vision-Based Human Motion Capture”. In: *Computer Vision and Image Understanding* 81.3 (2001), pp. 231–268. ISSN: 1077-3142. DOI: https://doi.org/10.1006/cviu.2000.0897. URL: https://www.sciencedirect.com/science/article/pii/S107731420090897X.

[191] Maria Montessori. *The montessori method*. Transaction publishers, 2013.

[192] Peter Mundy and Lisa Newell. “Attention, joint attention, and social cognition”. In: *Current Directions in Psychological Science* 16.5 (Oct. 2007), pp. 269–274. ISSN: 09637214. DOI: 10.1111/j.1467-8721.2007.00518.x.

[193] D. H. Murphy. “Strategic Offloading: How the Value Of To-be-remembered Information Influences Offloading Decision-making”. In: *Applied Cognitive Psychology* 37 (4 2023), pp. 749–767. DOI: 10.1002/acp.4051.

[194] Robin L. Nabi and Marina Krcmar. “Conceptualizing Media Enjoyment as Attitude: Implications for Mass Media Effects Research”. In: *Communication Theory* 14.4 (Nov. 2004), 288–310. ISSN: 1050-3293. DOI: 10.1111/j.1468-2885.2004.tb00316.x.

[195] Lennart Nacke and Anders Drachen. “Towards a Framework of Player Experience Research”. In: *Proceedings of the EPEX 11 Workshop*. 2011. ISBN: 9781450302678.

[196] Lennart Nacke and Craig A Lindley. “Affective Ludology, Flow and Immersion in a First- Person Shooter: Measurement of Player Experience”. In: *Loading...: The Journal of the Canadian Game Studies Association 3, 5.* 3 (2009). URL: http://arxiv.org/abs/1004.0248.

[197] Lennart Nacke and Craig A Lindley. “Flow and immersion in first-person shooters”. In: *Proceedings of the 2008 Conference on Future Play Research, Play, Share - Future Play ’08*. New York, New York, USA: ACM Press, 2008, p. 81. ISBN: 9781605582184. DOI: 10.1145/1496984.1496998.

[198] Abhishek Nandy and Manisha Biswas. “Unity ML-Agents”. In: *Neural Networks in Unity: C# Programming for Windows 10*. Berkeley, CA: Apress, 2018, pp. 27–67. ISBN: 978-1-4842-3673-4. DOI: 10.1007/978-1-4842-3673-4_2.

[199] Kieran Newell, Kunjal Patel, Magnus Linden, Michael Demetriou, Michal Huk, and Mohammed Mahmoud. “Evolution of Software Development in the Video Game Industry”. In: *2021 International Conference on Computational Science and Computational Intelligence (CSCI)*. 2021, pp. 1989–1996. DOI: 10.1109/CSCI54926.2021.00367.

[200] James Newton-King. *Json.NET: a high-performance JSON framework for .NET*. https://github.com/JamesNK/Newtonsoft.Json. 2007.

[201] Nintendo. *Wii Remote*. 2006. URL: https://en.wikipedia.org/wiki/Wii_Remote (visited on 03/20/2016).

[202] Donald A. Norman. “Natural User Interfaces Are Not Natural”. In: *Interactions* 17.3 (May 2010), pp. 6–10. ISSN: 1072-5520. DOI: 10.1145/1744161.1744163.

[203] Donald A. Norman. *Things That Make Us Smart: Defending Human Attributes in the Age of the Machine*. USA: Addison-Wesley Longman Publishing Co., Inc., 1993. ISBN: 0201626950.

[204] K. L. Norman. “GEQ (Game Engagement/Experience Questionnaire): A Review of Two Papers”. In: *Interacting with Computers* 25.4 (July 2013), 278–283. ISSN: 0953-5438. DOI: 10.1093/iwc/iwt009.

[205] Juliet Norton, Chadwick A. Wingrave, and Joseph J. LaViola. “Exploring Strategies and Guidelines for Developing Full Body Video Game Interfaces”. In: *Proceedings of the Fifth International Conference on the Foundations of Digital Games*. FDG ’10. Monterey, California: Association for Computing Machinery, 2010, pp. 155–162. ISBN: 9781605589374. DOI: 10.1145/1822348.1822369.

[206] Nicolas Nova and Laurent. Bolli. *Joypads! : the design of game controllers*. 1 edition. CreateSpace Independent Publishing Platform, 2014, p. 187. ISBN: 1497323495.

[207] Kenton O’hara, Richard Harper, Helena Mentis, Abigail Sellen, and Alex Taylor. “On the Naturalness of Touchless: Putting the “Interaction” Back into NUI”. In: *ACM Transactions on Computer-Human Interaction* 20.1 (Apr. 2013). ISSN: 1073-0516. DOI: 10.1145/2442106.2442111.

[208] Heather L. O’Brien. “Theoretical Perspectives on User Engagement”. In: *Why Engagement Matters: Cross-Disciplinary Perspectives of User Engagement in Digital Media*. Springer, 2016, 1–26.

[209] Heather L. O’Brien and Elaine G. Toms. “What is user engagement? A conceptual framework for defining user engagement with technology”. In: *Journal of the American Society for Information Science and Technology* 59.6 (Apr. 2008), 938–955. ISSN: 15322882. DOI: 10.1002/asi.20801. URL: http://doi.wiley.com/10.1002/asi.20801.

[210] Oculus. *Oculus Rift*. 2015. URL: https://www.oculus.com/en-us/rift/.

[211] Oculus. *Oculus Touch*. 2015. URL: https://www.oculus.com/en-us/rift/%7B%5C#%7Doculus-touch.

[212] Antti Oulasvirta, Esko Kurvinen, and Tomi Kankainen. “Understanding contexts by being there: case studies in bodystorming”. In: *Personal and ubiquitous computing* 7 (2003), pp. 125–134.

[213] S. Parke-Wolf, H. Scurto, and R. Fiebrink. “Sound Control: Supporting Custom Musical Interface Design for Children with Disabilities.” In: *Proc. of the International Conference on New Interfaces for Musical Expression*. 2019.

[214] Lara Pearson and Wim Pouw. “Gesture–vocal coupling in Karnatak music performance: A neuro–bodily distributed aesthetic entanglement”. In: *Annals of the New York Academy of Sciences* 1515.1 (2022), pp. 219–236. DOI: https://doi.org/10.1111/nyas.14806.

[215] Christopher Pedersen, Julian Togelius, and G.N. Yannakakis. “Modeling Player Experience for Content Creation”. In: *IEEE Transactions on Computational Intelligence and AI in Games* 2.1 (Mar. 2010), 54–67. ISSN: 1943-068X. DOI: 10.1109/TCIAIG.2010.2043950. URL: http://ieeexplore.ieee.org/document/5420018/.

[216] Phoenix Perry, Carlos Gonzalez Diaz, and Rebecca Fiebrink. “Interactive Machine Learning for Games”. Experimental AI for Games. Games Developers Conference (GDC) 2019. 2019. URL: https://www.gdcvault.com/play/1025906.

[217] Fábio Petrillo, Marcelo Pimenta, Francisco Trindade, and Carlos Dietrich. “What Went Wrong? A Survey of Problems in Game Development”. In: *Computer Entertainment* 7.1 (Feb. 2009). DOI: 10.1145/1486508.1486521.

[218] Jeffrey S. Pierce and Randy Pausch. “Generating 3D Interaction Techniques by Identifying and Breaking Assumptions”. In: *Virtual Reality* 11.1 (Mar. 2007), pp. 15–21. ISSN: 1359-4338.

[219] Nicola Plant, Ruth Gibson, Carlos Gonzalez Diaz, Bruno Martelli, Michael Zbyszyński, Rebecca Fiebrink, Marco Gillies, Clarice Hilton, and Phoenix Perry. “Movement interaction design for immersive media using interactive machine learning”. In: *ACM International Conference Proceeding Series*. New York, NY, USA: Association for Computing Machinery, July 2020, pp. 1–2. ISBN: 9781450375054. DOI: 10.1145/3401956.3404252.

[220] Nicola Plant, Clarice Hilton Goldsmiths, Marco Gillies, Michael Zbyszyński, Rebecca Fiebrink, Phoenix Perry, Carlos Gonzalez Diaz, Ruth Gibson, Studio Gibson, and / Martelli. “Programming by Moving: Interactive Machine Learning for Embodied Interaction Design”. In: *Workshop "The UX of Interactive Machine Learning" at NordiCHI ’20*. 2020. DOI: 10.1145/3359852.3359869.

[221] Nicola Plant, Clarice Hilton, Marco Gillies, Rebecca Fiebrink, Phoenix Perry, Carlos González DÍaz, Ruth Gibson, Bruno Martelli, and Michael Zbyszyński. “Interactive Machine Learning for Embodied Interaction Design: A tool and methodology”. In: *TEI 2021 - Proceedings of the 15th International Conference on Tangible, Embedded, and Embodied Interaction* (Feb. 2021). DOI: 10.1145/3430524.3442703.

[222] Nicola Plant, Clarice Hilton, and Carlos Gonzalez Diaz. “Using Machine Learning to Design Movement Interaction in VR”. Tate Modern for International Women’s day 2020. 2020. URL: https://sites.gold.ac.uk/4i/tate-modern-for-international-womens-day/.

[223] Karolien Poels, Wouter van den Hoogen, Wijnand Ijsselsteijn, and Yvonne de Kort. “Pleasure to Play, Arousal to Stay: The Effect of Player Emotions on Digital Game Preferences and Playing Time”. In: *Cyberpsychology, Behavior, and Social Networking* 15.1 (2012), pp. 1–6. ISSN: 2152-2715. DOI: 10.1089/cyber.2010.0040.

[224] Karolien Poels, Yvonne de Kort, and Wijnand Ijsselsteijn. “"It is Always a Lot of Fun!": Exploring Dimensions of Digital Game Experience Using Focus Group Methodology”. In: *Proceedings of the 2007 Conference on Future Play*. Future Play ’07. Toronto, Canada: Association for Computing Machinery, 2007, pp. 83–89. ISBN: 9781595939432. DOI: 10.1145/1328202.1328218.

[225] Age Poom, Kati Orru, and Rein Ahas. “The carbon footprint of business travel in the knowledge-intensive service sector”. In: *Transportation Research Part D: Transport and Environment* 50 (2017), pp. 292–304. ISSN: 1361-9209. DOI: https://doi.org/10.1016/j.trd.2016.11.014.

[226] R. Porter, J. Theiler, and D. Hush. “Interactive Machine Learning in Data Exploitation”. In: *Computing in Science & Engineering* 15.5 (2013), pp. 12–20.

[227] Wim Pouw and Susanne Fuchs. “Origins of vocal-entangled gesture”. In: *Neuroscience & Biobehavioral Reviews* 141 (2022), p. 104836. ISSN: 0149-7634. DOI: https://doi.org/10.1016/j.neubiorev.2022.104836.

[228] Andrew K. Przybylski, Richard M. Ryan, and C. Scott Rigby. “The motivating role of violence in video games”. In: *Personality and Social Psychology Bulletin* 35.2 (2009), 243–259. ISSN: 01461672. DOI: 10.1177/0146167208327216.

[229] Luis Radford. “The Progressive Development of Early Embodied Algebraic Thinking”. In: *Mathematics Education Research Journal* 26.2 (June 2014), pp. 257–277. ISSN: 1033-2170. DOI: 10.1007/s13394-013-0087-2. URL: http://link.springer.com/10.1007/s13394-013-0087-2.

[230] H. Raghavan, O. Madani, and R. Jones. “Active Learning with Feedback on Features and Instances”. In: *Journal of Machine Learning Research* 7.8 (2006), pp. 1655–1686.

[231] Raphael Marques. *Gesture Recognizer Unity Asset*. 2017. URL: https://assetstore.unity.com/packages/tools/input-management/gesture-recognizer-86410.

[232] Raving Bots Studio. *VR Magic Gestures AI Unity Asset*. 2017. URL: https://assetstore.unity.com/packages/tools/behavior-ai/vr-magic-gestures-ai-88011.

[233] Ygor Rebouças Serpa and Maria Andréia Formico Rodrigues. “Towards Machine-Learning Assisted Asset Generation for Games: A Study on Pixel Art Sprite Sheets”. In: *2019 18th Brazilian Symposium on Computer Games and Digital Entertainment (SBGames)*. 2019, pp. 182–191. DOI: 10.1109/SBGames.2019.00032.

[234] Johnmarshall Reeve. “Introduction”. In: *Understanding Motivation and Emotion*. 5th. Wiley, 2008. Chap. 1, 2–23. ISBN: 978-0-470-39223-2.

[235] Leonard Reinecke, Ron Tamborini, Matthew Grizzard, Robert Lewis, Allison Eden, and Nicholas David Bowman. “Characterizing Mood Management as Need Satisfaction: The Effects of Intrinsic Needs on Selective Exposure and Mood Repair”. In: *Journal of Communication* 62.3 (2012), pp. 437–453. ISSN: 00219916. DOI: 10.1111/j.1460-2466.2012.01649.x.

[236] Jorge Luis Reyes-Ortiz, Alessandro Ghio, Xavier Parra, Davide Anguita, Joan Cabestany, and Andreu Catala. “Human Activity and Motion Disorder Recognition: towards smarter Interactive Cognitive Environments.” In: *ESANN*. 2013. ISBN: 9782874190810.

[237] Evan F Risko and Sam J Gilbert. “Cognitive offloading”. In: *Trends in cognitive sciences* 20.9 (2016), pp. 676–688.

[238] S. Rogers. “The Three Cs, Part 3: Controls”. In: *Level up!: the guide to great video game design*. John Wiley & Sons, 2014, pp. 163–177.

[239] Yvonne Rogers and Henk Muller. “A Framework for Designing Sensor-Based Interactions to Promote Exploration and Reflection in Play”. In: *International Journal of Human-Computer Studies* 64.1 (Jan. 2006), pp. 1–14. ISSN: 1071-5819. DOI: 10.1016/j.ijhcs.2005.05.004.

[240] Philip R Ross and Stephan AG Wensveen. “Designing aesthetics of behavior in interaction: Using aesthetic experience as a mechanism for design”. In: *International Journal of Design* 4.2 (2010), pp. 3–13.

[241] James A. Russell. “A circumplex model of affect.” In: *Journal of Personality and Social Psychology* 39.6 (1980), 1161–1178. ISSN: 0022-3514. DOI: 10.1037/h0077714.

[242] Richard M. Ryan and Edward L. Deci. “Intrinsic and Extrinsic Motivations: Classic Definitions and New Directions”. In: *Contemporary Educational Psychology* 25.1 (Jan. 2000), 54–67. ISSN: 0361476X. DOI: 10.1006/ceps.1999.1020.

[243] Richard M. Ryan, C. Scott Rigby, and Andrew Przybylski. “The motivational pull of video games: A self-determination theory approach”. In: *Motivation and Emotion* 30.4 (2006), 347–363. ISSN: 01467239. DOI: 10.1007/s11031-006-9051-8.

[244] Margaret Schedel, Rebecca Fiebrink, and Phoenix Perry. “Wekinating #000000Swan: Using machine learning to create and control complex artistic systems”. In: *Proceedings of the 11th International Conference on New Interfaces for Musical Expression* (June 2011), pp. 453–456. ISSN: 2220-4806.

[245] Emanuel A. Schegloff. “Overlapping Talk and the Organization of Turn-Taking for Conversation”. In: *Language in Society* 29.1 (2000), pp. 1–63. URL: www.jstor.org/stable/4168983.

[246] Dennis Schleicher, Peter Jones, and Oksana Kachur. “Bodystorming as embodied designing”. In: *interactions* 17.6 (2010), pp. 47–51.

[247] Mike Schmierbach, Anthony M Limperos, and Julia K. Woolley. “Feeling the Need for (Personalized) Speed: How Natural Controls and Customization Contribute to Enjoyment of a Racing Game Through Enhanced Immersion”. In: *Cyberpsychology, Behavior, and Social Networking* 15.7 (July 2012), 364–369. ISSN: 2152-2715. DOI: 10.1089/cyber.2012.0025.

[248] Mike Schmierbach, Qian Xu, Anne Oeldorf-Hirsch, and Frank E. Dardis. “Electronic Friend or Virtual Foe: Exploring the Role of Competitive and Cooperative Multiplayer Video Game Modes in Fostering Enjoyment”. In: *Media Psychology* 15.3 (2012), 356–371. ISSN: 15213269. DOI: 10.1080/15213269.2012.702603.

[249] Henrik Schoenau-Fog. “The Player Engagement Process – An Exploration of Continuation Desire in Digital Games”. In: *DiGRA 2011*. Hilversum, 2011.

[250] Henrik Schoenau-Fog and Schoenau-Fog Henrik. “The Player Engagement Process – An Exploration of Continuation Desire in Digital Games”. In: *DiGRA 2011 Conference: Think Design Play* 6 (2014), pp. 1–18.

[251] Alexander Schwartz and Devin Reimer. *The Job Simulator Postmortem: VR Design, Tech and Business Lessons Learned*. 2017. URL: https://youtu.be/G5a5VIdjiJA (visited on 06/27/2023).

[252] May-li Seah and Paul Cairns. “From immersion to addiction in videogames”. In: *Proceedings of the 22nd British HCI Group Annual Conference on People and Computers: Culture, Creativity, Interaction*. Liverpool, United Kingdom: British Computer Society, 2008, 55–63. ISBN: 978-1-906124-04-5. URL: http://dl.acm.org/citation.cfm?id=1531514.1531522.

[253] W. Seok, Y. Kim, and C. Park. “Pattern recognition of human arm movement using deep reinforcement learning”. In: *2018 International Conference on Information Networking (ICOIN)*. 2018, pp. 917–919. DOI: 10.1109/ICOIN.2018.8343257.

[254] F.J. Seron, R. Rodriguez, E. Cerezo, and A. Pina. “Adding support for high-level skeletal animation”. In: *IEEE Transactions on Visualization and Computer Graphics* 8.4 (2002), pp. 360–372. DOI: 10.1109/TVCG.2002.1044521.

[255] Lucas M. Seuren, Joseph Wherton, Trisha Greenhalgh, and Sara E. Shaw. “Whose turn is it anyway? Latency and the organization of turn-taking in video-mediated interaction”. In: *Journal of Pragmatics* 172 (Jan. 2021), pp. 63–78. ISSN: 0378-2166. DOI: 10.1016/J.PRAGMA.2020.11.005.

[256] Daniel M. Shafer, Corey P. Carbonara, and Lucy Popova. “Controller Required? The Impact of Natural Mapping on Interactivity, Realism, Presence, and Enjoyment in Motion-Based Video Games”. In: *Presence: Teleoperators and Virtual Environments* 23.3 (Oct. 2014), pp. 267–286. DOI: 10.1162/PRES_a_00193.

[257] Daniel M. Shafer, Corey P. Carbonara, and Lucy Popova. “Spatial Presence and Perceived Reality as Predictors of Motion-Based Video Game Enjoyment”. In: *Presence: Teleoperators and Virtual Environments* 20.6 (Dec. 2011), pp. 591–619. DOI: 10.1162/PRES_a_00084.

[258] W. Shanklin. *Sony PlayStation VR review: Broken promise*. 2016. URL: http://newatlas.com/psvr-review/45938/.

[259] Kennon M Sheldon, Andrew J Elliot, Youngmee Kim, and Tim Kasser. “What Is Satisfying About Satisfying Events ? Testing 10 Candidate Psychological Needs”. In: *Journal of Personality and Social Psychology* 80.2 (2001), 325–339. DOI: 10.1037//0022-3514.80.2.325.

[260] Allen Sherrod. *Ultimate 3D Game Engine Design & Architecture*. USA: Charles River Media, Inc., 2006. ISBN: 1584504730.

[261] John L. Sherry. “Flow and Media Enjoyment”. In: *Communication Theory* 14.4 (Nov. 2004), 328–347. ISSN: 1050-3293. DOI: 10.1111/j.1468-2885.2004.tb00318.x.

[262] Kyong Jin Shim, Jaideep Srivastava, and Kuo Wei Hsu. “An exploratory study of player performance, motivation, and enjoyment in massively multiplayer online role-playing games”. In: *Proceedings - 2011 IEEE International Conference on Privacy, Security, Risk and Trust and IEEE International Conference on Social Computing, PASSAT/SocialCom 2011*. 2011, 135–140. ISBN: 9780769545783. DOI: 10.1109/PASSAT/SocialCom.2011.156.

[263] Chaklam Silpasuwanchai and Xiangshi Ren. “Designing concurrent full-body gestures for intense gameplay”. In: *International Journal of Human-Computer Studies* 80 (2015), pp. 1–13. ISSN: 1071-5819. DOI: https://doi.org/10.1016/j.ijhcs.2015.02.010. URL: https://www.sciencedirect.com/science/article/pii/S1071581915000439.

[264] Josep Silva. “A survey on algorithmic debugging strategies”. In: *Advances in Engineering Software* 42.11 (Nov. 2011), pp. 976–991. ISSN: 0965-9978. DOI: 10.1016/J.ADVENGSOFT.2011.05.024.

[265] P. Skalski, R. Tamborini, a. Shelton, M. Buncher, and P. Lindmark. “Mapping the road to fun: Natural video game controllers, presence, and game enjoyment”. In: *New Media & Society* 13.2 (2011), 224–242. ISSN: 1461-4448. DOI: 10.1177/1461444810370949.

[266] Richard Skarbez, Frederick P. Brooks, and Mary C. Whitton. “A Survey of Presence and Related Concepts”. In: *ACM Computing Surveys (CSUR)* 50.6 (Nov. 2017), p. 96. ISSN: 15577341. DOI: 10.1145/3134301. URL: https://dl.acm.org/doi/10.1145/3134301.

[267] Mel Slater. “Place illusion and plausibility can lead to realistic behaviour in immersive virtual environments”. In: *Philosophical Transactions of the Royal Society B: Biological Sciences* 364.1535 (Dec. 2009), pp. 3549–3557. ISSN: 14712970. DOI: 10.1098/RSTB.2009.0138. URL: https://royalsocietypublishing.org/doi/10.1098/rstb.2009.0138.

[268] Mel Slater, Anthony Steed, John McCarthy, and Francesco Maringelli. “The Influence of Body Movement on Subjective Presence in Virtual Environments”. In: *Human Factors* 40.3 (Sept. 1998), pp. 469–477. ISSN: 00187208. DOI: 10.1518/001872098779591368. URL: https://journals.sagepub.com/doi/abs/10.1518/001872098779591368?journalCode=hfsa.

[269] José L. Soler-Domínguez, Carla De Juan, Manuel Contero, and Mariano Alcañiz. “I walk, therefore I am: a multidimensional study on the influence of the locomotion method upon presence in virtual reality”. In: *Journal of Computational Design and Engineering* 7.5 (Oct. 2020), pp. 577–590. ISSN: 22885048. DOI: 10.1093/JCDE/QWAA040. URL: https://dx.doi.org/10.1093/jcde/qwaa040.

[270] Stephan Sonnenburg. “Creativity in Communication: A Theoretical Framework for Collaborative Product Creation”. In: *Creativity and Innovation Management* 13.4 (Dec. 2004), pp. 254–262. ISSN: 1467-8691. DOI: 10.1111/J.0963-1690.2004.00314.X.

[271] Sony. *PlayStation VR Aim Controller*. 2017. URL: https://www.playstation.com/en-gb/explore/accessories/playstation-vr-aim-controller/.

[272] Sony Computer Entertainment. *PlayStation Move*. 2010. URL: https://en.wikipedia.org/wiki/PlayStation_Move.

[273] Sony Interactive Entertainment. *PlayStation VR*. 2016. URL: https://www.playstation.com/en-gb/explore/playstation-vr/.

[274] Sony Interactive Entertainment. *PlayStation VR 2*. 2023. URL: https://www.playstation.com/en-gb/ps-vr2/.

[275] Rahil Soroushmojdehi, Sina Javadzadeh, Alessandra Pedrocchi, and Marta Gandolla. “Transfer learning in hand movement intention detection based on surface electromyography signals”. In: *Frontiers in Neuroscience* 16 (Nov. 2022), p. 977328. DOI: 10.3389/fnins.2022.977328.

[276] F. Sperrle, M. El-Assady, G. Guo, R. Borgo, D. Horng Chau, A. Endert, and D. Keim. “A Survey of Human-Centered Evaluations in Human-Centered Machine Learning”. In: *Computer Graphics Forum* 40.3 (June 2021), pp. 543–568. ISSN: 1467-8659. DOI: 10.1111/CGF.14329. URL: https://onlinelibrary.wiley.com/doi/full/10.1111/cgf.14329%20https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.14329%20https://onlinelibrary.wiley.com/doi/10.1111/cgf.14329.

[277] Nancy Staggers and Anthony F. Norcio. “Mental models: concepts for human-computer interaction research”. In: *International Journal of Man-machine studies* 38.4 (1993), pp. 587–605.

[278] Lisa J. Stephenson, S. Gareth Edwards, and Andrew P. Bayliss. “From Gaze Perception to Social Cognition: The Shared-Attention System:” in: *Perspectives on Psychological Science* 16.3 (Feb. 2021), pp. 553–576. ISSN: 17456924. DOI: 10.1177/1745691620953773.

[279] David Sudnow. *Ways of the hand: The organization of improvised conduct*. MIT Press, 1993.

[280] SUPERHOT Team. *SUPERHOT VR*. 2017.

[281] Dag Svanæs. “Interaction Design for and with the Lived Body: Some Implications of Merleau-Ponty’s Phenomenology”. In: *ACM Transactions on Computer-Human Interaction* 20.1 (Apr. 2013). ISSN: 1073-0516. DOI: 10.1145/2442106.2442114.

[282] Dag Svanæs. “Interaction is Orthogonal to Graphical Form”. In: *INTERACT ’93 and CHI ’93 Conference Companion on Human Factors in Computing Systems*. CHI ’93. Amsterdam, The Netherlands: Association for Computing Machinery, 1993, pp. 79–80. ISBN: 0897915747. DOI: 10.1145/259964.260100.

[283] Dag Svanæs. “Kinaesthetic thinking: The tacit dimension of interaction design”. In: *Computers in Human Behavior* 13.4 (1997). Norwegian Perspectives on Computing in Complex Domains, pp. 443–463. ISSN: 0747-5632. DOI: https://doi.org/10.1016/S0747-5632(97)00020-4. URL: https://www.sciencedirect.com/science/article/pii/S0747563297000204.

[284] Dag Svanæs. *Understanding interactivity: steps to a phenomenology of human-computer interaction.* Norges teknisk-naturvitenskapelige universitet, 2000.

[285] Dag Svanaes and Gry Seland. “Putting the Users Center Stage: Role Playing and Low-Fi Prototyping Enable End Users to Design Mobile Systems”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI ’04. Vienna, Austria: Association for Computing Machinery, 2004, pp. 479–486. ISBN: 1581137028. DOI: 10.1145/985692.985753.

[286] Penelope Sweetser, Daniel Johnson, Peta Wyeth, Aiman Anwar, Yan Meng, and Anne Ozdowska. “GameFlow in Different Game Genres and Platforms”. In: *Computers in Entertainment* 15.3 (Apr. 2017), 1–24. ISSN: 15443574. DOI: 10.1145/3034780. URL: http://dl.acm.org/citation.cfm?doid=3044431.3034780.

[287] Penelope Sweetser and Peta Wyeth. “GameFlow”. In: *Computers in Entertainment* 3.3 (July 2005), p. 3. ISSN: 15443574. DOI: 10.1145/1077246.1077253.

[288] Steve Swink. *Game Feel: A Game Designer’s Guide to Virtual Sensation*. Amsterdam et al: Morgan Kaufman, 2009.

[289] Ron Tamborini, Nicholas David Bowman, Allison Eden, Matthew Grizzard, and Ashley Organ. “Defining Media Enjoyment as the Satisfaction of Intrinsic Needs”. In: *Journal of Communication* 60.4 (Dec. 2010), 758–777. ISSN: 00219916. DOI: 10.1111/j.1460-2466.2010.01513.x. URL: http://doi.wiley.com/10.1111/j.1460-2466.2010.01513.x.

[290] Ron Tamborini, Matthew Grizzard, Nicholas David Bowman, Leonard Reinecke, Robert J. Lewis, and Allison Eden. “Media Enjoyment as Need Satisfaction: The Contribution of Hedonic and Nonhedonic Needs”. In: *Journal of Communication* 61.6 (Dec. 2011), pp. 1025–1042. ISSN: 00219916. DOI: 10.1111/j.1460-2466.2011.01593.x.

[291] Atau Tanaka. “Embodied Musical Interaction”. In: *New Directions in Music and Human-Computer Interaction*. Ed. by Simon Holland, Tom Mudd, Katie Wilkie-McKenna, Andrew McPherson, and Marcelo M. Wanderley. Cham: Springer International Publishing, 2019, pp. 135–154. ISBN: 978-3-319-92069-6. DOI: 10.1007/978-3-319-92069-6_9.

[292] Jian Tang, Jingzhou Liu, Ming Zhang, and Qiaozhu Mei. “Visualizing Large-Scale and High-Dimensional Data”. In: *Proceedings of the 25th International Conference on World Wide Web*. WWW ’16. Montréal, Québec, Canada: International World Wide Web Conferences Steering Committee, 2016, pp. 287–297. ISBN: 9781450341431. DOI: 10.1145/2872427.2883041.

[293] Eugene M. Taranta II, Corey R. Pittman, Mehran Maghoumi, Mykola Maslych, Yasmine M. Moolenaar, and Joseph J. Laviola Jr. “Machete: Easy, Efficient, and Precise Continuous Custom Gesture Segmentation”. In: *ACM Transaction on Computer-Human Interaction* 28.1 (Jan. 2021). ISSN: 1073-0516. DOI: 10.1145/3428068.

[294] Unity Technologies. *Publishing Builds*. Unity3D Documentation. 2022. URL: https://docs.unity3d.com/Manual/PublishingBuilds.html.

[295] Sabine Trepte and Leonard Reinecke. “The Pleasures of Success: Game-Related Efficacy Experiences as a Mediator Between Player Performance and Game Enjoyment”. In: *Cyberpsychology, Behavior, and Social Networking* 14.9 (2011), 555–557. ISSN: 2152-2715. DOI: 10.1089/cyber.2010.0358.

[296] T. Tsandilas, W. Mackay, and F. Bevilacqua. “Musink: composing music through augmented drawing”. In: *Proceedings of the 2009 ACM SIGGRAPH Symposium on Video Games - Sandbox ’09*. 2009, pp. 89–96.

[297] Nikolaos Tsigilis and Argiris Theodosiou. “Temporal Stability of the Intrinsic Motivation Inventory”. In: *Perceptual and Motor Skills* 97.1 (Aug. 2003), 271–280. ISSN: 0031-5125. DOI: 10.2466/pms.2003.97.1.271. URL: http://journals.sagepub.com/doi/10.2466/pms.2003.97.1.271.

[298] John Underkoffler and Hiroshi Ishii. “Illuminating light: An optical design tool with a luminous-tangible interface”. In: *Proceedings of the SIGCHI conference on Human factors in computing systems*. 1998, pp. 542–549.

[299] Unity Technologies. *Blueprint Overview*. 2019. URL: https://docs.unity3d.com/Packages/com.unity.shadergraph@5.0/manual/index.html (visited on 03/10/2024).

[300] Unity Technologies. *Unity 3D*. 2005. URL: http://unity3d.com/.

[301] Unity Technologies. *XR Interaction Toolkit*. 2018. URL: https://docs.unity3d.com/Packages/com.unity.xr.interaction.toolkit@0.0/manual/index.html.

[302] Universal Gesture. *Universal Gesture for VR Unity Asset*. 2018. URL: https://assetstore.unity.com/packages/tools/input-management/universal-gesture-for-vr-129142.

[303] Valve Corporation. *Knuckles Quick Start*. 2017. URL: http://steamcommunity.com/sharedfiles/filedetails/?id=943406651.

[304] Andries Van Dam. “Post-WIMP user interfaces”. In: *Communications of the ACM* 40.2 (1997), pp. 63–67.

[305] Jean Vanderdonckt and Radu-Daniel Vatavu. “Designing, Engineering, and Evaluating Gesture User Interfaces”. In: *Extended Abstracts of the 2018 CHI Conference on Human Factors in Computing Systems*. CHI EA ’18. Montreal QC, Canada: Association for Computing Machinery, 2018, pp. 1–4. ISBN: 9781450356213. DOI: 10.1145/3170427.3170648.

[306] Radu-Daniel Vatavu, Lisa Anthony, and Jacob O. Wobbrock. “Gestures as Point Clouds: A $P Recognizer for User Interface Prototypes”. In: *Proceedings of the 14th ACM International Conference on Multimodal Interaction*. ICMI ’12. Santa Monica, California, USA: Association for Computing Machinery, 2012, pp. 273–280. ISBN: 9781450314671. DOI: 10.1145/2388676.2388732.

[307] Y. Vazquez-Alvarez, M. Grierson, and R. Fiebrink. “Physical movement as an interaction modality for machine learning”. In: *Proceedings of the 27th International BCS Human Computer Interaction Conference (HCI)*. 2013, pp. 208–217.

[308] VERBI Software. *MAXQDA 2022*. Berlin, Germany, 2021. URL: maxqda.com.

[309] Peter Vorderer, Christoph Klimmt, and Ute Ritterfeld. “Enjoyment: At the Heart of Media Entertainment”. In: *Communication Theory* 14.4 (Nov. 2004), 388–408. ISSN: 1050-3293. DOI: 10.1111/j.1468-2885.2004.tb00321.x.

[310] Alf Inge Wang and Njål Nordmark. “Software Architectures and the Creative Processes in Game Development”. In: *Entertainment Computing - ICEC 2015*. Ed. by Konstantinos Chorianopoulos, Monica Divitini, Jannicke Baalsrud Hauge, Letizia Jaccheri, and Rainer Malaka. Cham: Springer International Publishing, 2015, pp. 272–285. ISBN: 978-3-319-24589-8.

[311] Dangxiao WANG, Yuan GUO, Shiyi LIU, Yuru ZHANG, Weiliang XU, and Jing XIAO. “Haptic display for virtual reality: progress and challenges”. In: *Virtual Reality & Intelligent Hardware* 1.2 (Apr. 2019), pp. 136–162. ISSN: 2096-5796. DOI: 10.3724/SP.J.2096-5796.2019.0008.

[312] Xun Wang, Ke Sun, Ting Zhao, Wei Wang, and Qing Gu. “Dynamic Speed Warping: Similarity-Based One-shot Learning for Device-free Gesture Signals”. In: *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*. 2020, pp. 556–565. DOI: 10.1109/INFOCOM41043.2020.9155491.

[313] Kazuki Wayama, Tomoyuki Yokogawa, Sousuke Amasaki, Hirohisa Aman, and Kazutami Arimoto. “Verifying Game Logic in Unreal Engine 5 Blueprint Visual Scripting System Using Model Checking”. In: *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*. ASE ’22. New York, NY, USA: Association for Computing Machinery, 2023. ISBN: 9781450394758. DOI: 10.1145/3551349.3560505. URL: https://doi.org/10.1145/3551349.3560505.

[314] David Weibel and Bartholomäus Wissmath. “Immersion in computer games: The role of spatial presence and flow”. In: *International Journal of Computer Games Technology* (2011). ISSN: 16877047. DOI: 10.1155/2011/282345.

[315] Joris Weijdom. “Performative Prototyping in Collaborative Mixed Reality Environments: An Embodied Design Method for Ideation and Development in Virtual Reality.” In: *Sixteenth International Conference on Tangible, Embedded, and Embodied Interaction*. TEI ’22. Daejeon, Republic of Korea: Association for Computing Machinery, 2022. ISBN: 9781450391474. DOI: 10.1145/3490149.3501316.

[316] S. A. G. Wensveen, J. P. Djajadiningrat, and C. J. Overbeeke. “Interaction Frogger: A Design Framework to Couple Action and Function through Feedback and Feedforward”. In: *Proceedings of the 5th Conference on Designing Interactive Systems: Processes, Practices, Methods, and Techniques*. DIS ’04. Cambridge, MA, USA: Association for Computing Machinery, 2004, pp. 177–184. ISBN: 1581137877. DOI: 10.1145/1013115.1013140.

[317] Ludwig Wittgenstein. “On Certainty.(tr. GEM Anscombe) Oxford”. In: *UK: Blackwell* (1953).

[318] Jingming Xie. “Research on key technologies base Unity3D game engine”. In: *2012 7th International Conference on Computer Science & Education (ICCSE)*. 2012, pp. 695–699. DOI: 10.1109/ICCSE.2012.6295169.

[319] Frank F. Xu, Uri Alon, Graham Neubig, and Vincent Josua Hellendoorn. “A Systematic Evaluation of Large Language Models of Code”. In: *Proceedings of the 6th ACM SIGPLAN International Symposium on Machine Programming*. MAPS 2022. San Diego, CA, USA: Association for Computing Machinery, 2022, pp. 1–10. ISBN: 9781450392730. DOI: 10.1145/3520312.3534862.

[320] Michael Zbyszyński, Mick Grierson, Matthew Yee-King, and Leon Fedden. “Write once run anywhere revisited: machine learning and audio tools in the browser with C++ and emscripten”. In: *Web Audio Conference 2017*. Aug. 2017. URL: https://research.gold.ac.uk/id/eprint/20968/.

[321] Zhengyou Zhang. “Microsoft Kinect Sensor and Its Effect”. In: *IEEE MultiMedia* 19.2 (2012), pp. 4–10. DOI: 10.1109/MMUL.2012.24.