The Agent Tool Lua API

1.01

Stephen Pearse

April 2015

Contents

Chapter 1

Class Index

1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 2

File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 3

Class Documentation

3.1 RGBA Class Reference

RGBA represents a pixel of data containing Red, Green, Blue and Alpha components.

`#include <RGBA.hpp>`

Public Member Functions

- RGBA new () RGBA.new(int r

  A default constructor will provide you with an RGBA with its attributes set to 0.
- RGBA new (int r, int g, int b)

  Allows the construction without providing an alpha component.
- int r ()
- int g ()
- int b ()
- int a ()
- int mean ()
- RGBA operator+ (int r)

  Allows the scalar addition of an RGBA affecting all of its components.
- RGBA operator+ (RGBA r)

  Allows addition of two RGBA's.
- RGBA operator- (int r)

  Allows the scalar subtraction from an RGBA.
- RGBA operator- (RGBA r)

  Allows subtraction of one RGBA from another.
- RGBA operator∗ (float r)

  Allows the scalar multiplication of an RGBA.
- RGBA operator∗ (RGBA r)

  Allows multiplication of one RGBA and another.

- RGBA operator∗ (float r)

  Allows the division of an RGBA by a scalar.

- RGBA operator/ (RGBA r)

  Allows division of one RGBA by another.

Public Attributes

- RGBA int **g**
- RGBA int int **b**
- RGBA int int int **a**

3.1.1 Detailed Description

RGBA represents a pixel of data containing Red, Green, Blue and Alpha components.

Colour attributes in The Agent Tool are represented as integer values between 0 and 255. To create a RGBA in Lua you must call RGBA.new() to create an instance of this object. To execute on of the classes functions you must utilize a colon followed by the name ot the function. For example colour:r() You can consequently set attributes via this mechanism. colour:r() = 255

3.1.2 Member Function Documentation

3.1.2.1 int RGBA::a ( )

Returns

The alpha component

3.1.2.2 int RGBA::b ( )

Returns

The blue component

3.1.2.3 int RGBA::g ( )

Returns

The green component

3.1.2.4 int RGBA::mean ( )

Returns

The average/mean of the Red, Green and Blue components

3.1.2.5 RGBA RGBA::new ( ) [new]

A default constructor will provide you with an RGBA with its attributes set to 0.

Allows the construction with provided attributes

3.1.2.6 RGBA RGBA::operator∗ (float *r*)

Allows the scalar multiplication of an RGBA.

Returns

A new RGBA reflecting the scalar multiplication

3.1.2.7 RGBA RGBA::operator∗ (RGBA *r*)

Allows multiplication of one RGBA and another.

Returns

A new RGBA reflecting the multiplication

3.1.2.8 RGBA RGBA::operator∗ (float *r*)

Allows the division of an RGBA by a scalar.

Returns

A new RGBA reflecting the division

3.1.2.9 RGBA RGBA::operator+ (int *r*)

Allows the scalar addition of an RGBA affecting all of its components.

Returns

A new RGBA reflecting the scalar summation

3.1.2.10 RGBA RGBA::operator+ (RGBA *r*)

Allows addition of two RGBA's.

Returns

A new RGBA reflecting the summation of two RGBA's

3.1.2.11 RGBA RGBA::operator- (int *r*)

Allows the scalar subtraction from an RGBA.

Returns

A new RGBA reflecting the scalar subtraction

3.1.2.12 RGBA RGBA::operator- (RGBA *r*)

Allows subtraction of one RGBA from another.

Returns

A new RGBA reflecting the subtraction

3.1.2.13 RGBA RGBA::operator/ (RGBA *r*)

Allows division of one RGBA by another.

Returns

A new RGBA reflecting the division

3.1.2.14 int RGBA::r ( )

Returns

The red component

The documentation for this class was generated from the following file:

- RGBA.hpp

3.2 Vec2 Class Reference

A Vec2 represents a pair of numbers that can be represent a either a two dimensional point or a vector.

`#include <Vec2.hpp>`

Public Member Functions

- Vec2 new ()

  A default constructor will provide you Vec2 with pair 0 0.
- Vec2 operator+ (Vec2 r)

  Allows the addition of two Vec2's.
- Vec2 operator+ (float r)

  Allows the scalar addition of a Vec2 affecting both components.
- Vec2 operator+ (int r)

  Allows the scalar addition of a Vec2 affecting both components.
- Vec2 operator- (Vec2 r)

  Allows the subtraction of one Vec2 from another.
- Vec2 operator- (float r)

  Allows the subtraction of a scalar value from a Vec2 affecting both components.
- Vec2 operator- (int r)

  Allows the subtraction of a scalar value from a Vec2 affecting both components.
- Vec2 operator∗ (float r)

  Allows the scalar multiplication of a Vec2.
- Vec2 operator∗ (int r)

  Allows the scalar multiplication of a Vec2.
- Vec2 operator/ (float r)

  Allows the division of a Vec2.
- float mag ()
- Vec2 norm ()
- Vec2 perp ()
- float dot (const Vec2 &v) const
- bool **operator==** (const Vec2 &r) const

3.2.1 Detailed Description

A Vec2 represents a pair of numbers that can be represent a either a two dimensional point or a vector.

To create a Vec2 in Lua you must call Vec2.new() to create an instance of this object. To execute on of the classes functions you must utilize a colon followed by the name ot the function. For example point:x() You can consequently set the x and y components via this mechanism. point:x() = 10

3.2.2 Member Function Documentation

3.2.2.1 float Vec2::dot (const Vec2 & *v*) const [inline]

Returns

The dot product of the Vec2 (x∗x + y∗y)

3.2.2.2 float Vec2::mag ( )

Returns

The magnitude of the Vec2 sqrt(x∗x + y∗y)

3.2.2.3 Vec2 Vec2::new ( ) [inline]

A default constructor will provide you Vec2 with pair 0 0.

Allows the construction of with a pair of floating point numbers.

Returns

The y value of the pair.

3.2.2.4 Vec2 Vec2::norm ( )

Returns

A normalized version of the Vec2

3.2.2.5 Vec2 Vec2::operator∗ (float *r*)

Allows the scalar multiplication of a Vec2.

Returns

A new Vec2 reflecting the scalar multiplication of the Vec2

3.2.2.6 Vec2 Vec2::operator∗ (int *r*)

Allows the scalar multiplication of a Vec2.

Returns

A new Vec2 reflecting the scalar multiplication of the Vec2

3.2.2.7 Vec2 Vec2::operator+ (Vec2 *r*)

Allows the addition of two Vec2's.

Returns

A new Vec2 reflecting the sum of two Vec2's

3.2.2.8 Vec2 Vec2::operator+ (float *r*)

Allows the scalar addition of a Vec2 affecting both components.

Returns

A new Vec2 reflecting the scalar summation

3.2.2.9 Vec2 Vec2::operator+ (int *r*)

Allows the scalar addition of a Vec2 affecting both components.

Returns

A new Vec2 reflecting the scalar summation

3.2.2.10 Vec2 Vec2::operator- (Vec2 *r*)

Allows the subtraction of one Vec2 from another.

Returns

A new Vec2 reflecting the subtraction of one Vec2 from another

3.2.2.11 Vec2 Vec2::operator- (float *r*)

Allows the subtraction of a scalar value from a Vec2 affecting both components.

Returns

A new Vec2 reflecting the scalar subtraction from the Vec2

3.2.2.12 Vec2 Vec2::operator- (int *r*)

Allows the subtraction of a scalar value from a Vec2 affecting both components.

Returns

A new Vec2 reflecting the scalar subtraction from the Vec2

3.2.2.13 Vec2 Vec2::operator/ (float *r*)

Allows the division of a Vec2.

Returns

A new Vec2 reflecting the scalar division of the Vec2

3.2.2.14 Vec2 Vec2::perp ( )

Returns

The perpendicual of the Vec2

The documentation for this class was generated from the following file:

- Vec2.hpp

Chapter 4

File Documentation

4.1 LuaApi.hpp File Reference

Functions

- void initai ()

  This MUST be called at the start of each Agent script. This initializes the parent script so that it is interpreted as an Agent by the system.
- float getDT ()

  A utility and timing function that provides the delta between the current frame and the previous (i.e time elapsed). Delta between now and previous frame.
- int getID ()

  A crucial function allowing an Agent to obtain its own ID. Provides the unique identifier for the currently active Agent.
- bool hasChildren ()

  Returns whether the current Agent has any children. Returns whether the current Agent has any children.
- bool hasChildrenID (int ID)

  Indicates whether an Agent, specified by an ID has any children. Indicates whether an Agent, specified by an ID has any children.
- int numberOfChildren ()

  Returns the number of children of the currently active Agent Returns the number of children of the currently active Agent.
- int numberOfChildrenID (int ID)

  Provides access to the number of children of a specified Agent. Provides access to the number of children of a specified Agent.
- int getChild (int childIndex)

  Allows you to obtain the unique ID of a child Agent. Allows you to obtain the unique ID of a child Agent through a relative index. For example, you can obtain the second child through getChild(1)
- int getChildID (int childID, int parentID)

Allows you to obtain the child of another Agent. Allows you to obtain a child from any Agent if its ID is known.

- int getParent ()

  Allows an Agent to obtain the ID of its parent. Allows an Agent to obtain the ID of its parent.

- int getParentID (int childID)

  Allows an Agent to obtain the ID of a parent of another if its ID is known. Allows an Agent to obtain the ID of a parent of another if its ID is known.

- void die ()

  Kills the currently active Agent. Kills the currently active Agent.

- void killID (int agentID)

  Allows an Agent to kill another Agent if its ID is known. Allows an Agent to kill another Agent if its ID is known.

- int spawn (int parentID, int nodeType, string agentFunctionName, int xPos, int yPos)

  Allows the spawning of an Agent (or another Node type) in world coordinates. he basic spawn function that allows the creation of new Agents. This is done so by specifying the parent ID, the name of the script function for the Agent to use and via providing world coordinates (as cartesian components or as a Vec2) for the Agent to spawn at.

- int spawn (int parentID, int nodeType, string agentFunctionName, Vec2 position)

  Allows the spawning of an Agent (or another Node type) in world coordinates. he basic spawn function that allows the creation of new Agents. This is done so by specifying the parent ID, the name of the script function for the Agent to use and via providing world coordinates (as cartesian components or as a Vec2) for the Agent to spawn at.

- int spawnRelative (int parentID, int nodeType, string agentFunctionName, int x-Pos, int yPos)

  Allows the spawning of an Agent (or another Node type) using coordinates relative to the currently active Agent. A spawn function that allows the creation of new Agents. This is done so by specifying the parent ID, the name of the script function for the Agent to use and via coordinates that are relative to the currently active Agent's position (as cartesian components or as a Vec2).

- int spawnRelative (int parentID, int nodeType, string agentFunctionName, Vec2 position)

  Allows the spawning of an Agent (or another Node type) using coordinates relative to the currently active Agent. A spawn function that allows the creation of new Agents. This is done so by specifying the parent ID, the name of the script function for the Agent to use and via coordinates that are relative to the currently active Agent's position (as cartesian components or as a Vec2).

- int spawnP (table propertyTable, int parentID, int nodeType, string agentFunction-Name, int xPos, int yPos)

  Allows the spawning of an Agent (or another Node type) with a table of properties for it to use. A spawn function that allows the creation of new Agents with a suite of properties stored within a table. This is done so by providing a table containing key value pairs {key = value}, the parent ID, the name of the script function for the Agent to use and via providing world coordinates (as cartesian components or as a Vec2) for the Agent to spawn at.

- int spawnP (table propertyTable, int parentID, int nodeType, string agentFunction-Name, Vec2 position)

Allows the spawning of an Agent (or another Node type) with a table of properties for it to use. A spawn function that allows the creation of new Agents with a suite of properties stored within a table. This is done so by providing a table containing key value pairs {key = value}, the parent ID, the name of the script function for the Agent to use and via providing world coordinates (as cartesian components or as a Vec2) for the Agent to spawn at.

- int spawnRelativeP (table propertyTable, int parentID, int nodeType, string agent-FunctionName, int xPos, int yPos)

  Allows the spawning of an Agent (or another Node type) with a table of properties for it to use with coordinates relative to the currently active Agent A spawn function that allows the creation of new Agents with a suite of properties stored within a table. This is done so by providing a table containing key value pairs {key = value}, the parent ID, the name of the script function for the Agent to use and via providing relative coordinates (as cartesian components or as a Vec2) for the Agent to spawn at.

- int spawnRelativeP (table propertyTable, int parentID, int nodeType, string agent-FunctionName, Vec2 position)

  Allows the spawning of an Agent (or another Node type) with a table of properties for it to use with coordinates relative to the currently active Agent A spawn function that allows the creation of new Agents with a suite of properties stored within a table. This is done so by providing a table containing key value pairs {key = value}, the parent ID, the name of the script function for the Agent to use and via providing relative coordinates (as cartesian components or as a Vec2) for the Agent to spawn at.

- float getLifetime ()

  Allows an Agent to obtain how long it has been alive in seconds. Allows an Agent to obtain how long it has been alive in seconds.

- float getLifetimeID (int ID)

  Allows an Agent to obtain how long another Agent has been alive in seconds. Allows an Agent to obtain how long another Agent has been alive in seconds.

- Vec2 getPosition ()

  Allows an Agent to quickly obtain its position. Allows an Agent to quickly obtain its position.

- Vec2 getPositionID (int ID)

  Allows Agents to quickly obtain the position of any given Agent. Allows Agents to quickly obtain the position of any given Agent.

- void setPosition (int xPos, int yPos)

  Allows an Agent to set its position. (Deprecated) Allows an Agent to set its position. This is deprecated, uses should use setP, specifying the position property. i.e set-P("pos" Vec2.new(100,100))

- void setPositionID (int xPos, int yPos, int ID)

  Allows an Agent to set the position of another Agent. Allows an Agent to set the position of another Agent. This is deprecated, uses should use setP, specifying the position property. i.e setPID("pos",Vec2.new(200,200),ID).

- void move (int x, int y)

  Allows an Agent to move by a specified amount. Allows an Agent to move by a specified amount.

- void move (Vec2 delta)

  Allows an Agent to move by a specified amount. Allows an Agent to move by a specified amount.

- void moveID (int x, int y, int ID)

Allows an Agent to move by a specified amount. Allows an Agent to move by a specified amount.

- void moveID (Vec2 delta, int ID)

  Allows an Agent to move by a specified amount. Allows an Agent to move by a specified amount.
- void initP (table propertyTable)

  Initializes the Agent with a suite of provided properties. Allows a function to be intialized as an Agent with a set of properties that are provided in a table of key value pairs {key=value}.
- void initPID (table propertyTable, int ID)

  Allows an Agent to initialize another Agent with a suite of properties. Allows an Agent to initialize another Agent with a suite of properties that are stored in a table of key value pairs {key=value}.
- void addP (string propertyID, bool value)

  Allows an Agent to add a new property. Allows an Agent to add a new property. The property can be of the following types: bool, float, int, string, Vec2.
- void addP (string propertyID, int value)

  Allows an Agent to add a new property. Allows an Agent to add a new property. The property can be of the following types: bool, float, int, string, Vec2.
- void addP (string propertyID, float value)

  Allows an Agent to add a new property. Allows an Agent to add a new property. The property can be of the following types: bool, float, int, string, Vec2.
- void addP (string propertyID, string value)

  Allows an Agent to add a new property. Allows an Agent to add a new property. The property can be of the following types: bool, float, int, string, Vec2.
- void addP (string propertyID, Vec2 value)

  Allows an Agent to add a new property. Allows an Agent to add a new property. The property can be of the following types: bool, float, int, string, Vec2.
- void addP (table propertyTable)

  Allows an Agent to add table of new properties. Allows an Agent to add table of new properties in key value pairs {key=value}. The property can be of the following types: bool, float, int, string, Vec2.
- void addPID (string propertyID, bool value, int ID)

  Allows an Agent to add a new property to another Agent. Allows an Agent to add a new property to another Agent. The property can be of the following types: bool, float, int, string, Vec2.
- void addPID (string propertyID, int value, int ID)

  Allows an Agent to add a new property to another Agent. Allows an Agent to add a new property to another Agent. The property can be of the following types: bool, float, int, string, Vec2.
- void addPID (string propertyID, float value, int ID)

  Allows an Agent to add a new property to another Agent. Allows an Agent to add a new property to another Agent. The property can be of the following types: bool, float, int, string, Vec2.
- void addPID (string propertyID, string value, int ID)

  Allows an Agent to add a new property to another Agent. Allows an Agent to add a new property to another Agent. The property can be of the following types: bool, float, int, string, Vec2.

- void addPID (string propertyID, Vec2 value, int ID)

  Allows an Agent to add a new property to another Agent. Allows an Agent to add a new property to another Agent. The property can be of the following types: bool, float, int, string, Vec2.

- void addPID (table propertyTable, int ID)

  Allows an Agent to add table of new properties to another Agent. Allows an Agent to add table of new properties in key value pairs {key=value} to another Agent. The property can be of the following types: bool, float, int, string, Vec2.

- void setP (string propertyID, bool value)

  Allows an Agent to set a property. If the property does not already exist, it will be created. Allows an Agent to set a property. If the property does not already exist, it will be created. If the current Agent is controlling a SuperCollider node, the system will attempt to set an argument of the same name to the proposed valued.

- void setP (string propertyID, int value)

  Allows an Agent to set a property. If the property does not already exist, it will be created. Allows an Agent to set a property. If the property does not already exist, it will be created. If the current Agent is controlling a SuperCollider node, the system will attempt to set an argument of the same name to the proposed valued.

- void setP (string propertyID, float value)

  Allows an Agent to set a property. If the property does not already exist, it will be created. Allows an Agent to set a property. If the property does not already exist, it will be created. If the current Agent is controlling a SuperCollider node, the system will attempt to set an argument of the same name to the proposed valued.

- void setP (string propertyID, string value)

  Allows an Agent to set a property. If the property does not already exist, it will be created. Allows an Agent to set a property. If the property does not already exist, it will be created. If the current Agent is controlling a SuperCollider node, the system will attempt to set an argument of the same name to the proposed valued.

- void setP (string propertyID, Vec2 value)

  Allows an Agent to set a property. If the property does not already exist, it will be created. Allows an Agent to set a property. If the property does not already exist, it will be created. If the current Agent is controlling a SuperCollider node, the system will attempt to set an argument of the same name to the proposed valued.

- void setPID (string propertyID, bool value, int ID)

  Allows an Agent to set a property of a specified Agent. If the property does not already exist, it will be created. Allows an Agent to set a property of a specified Agent. If the property does not already exist, it will be created. If the current Agent is controlling a SuperCollider node, the system will attempt to set an argument of the same name to the proposed valued.

- void setPID (string propertyID, int value, int ID)

  Allows an Agent to set a property of a specified Agent. If the property does not already exist, it will be created. Allows an Agent to set a property of a specified Agent. If the property does not already exist, it will be created. If the current Agent is controlling a SuperCollider node, the system will attempt to set an argument of the same name to the proposed valued.

- void setPID (string propertyID, float value, int ID)

  Allows an Agent to set a property of a specified Agent. If the property does not already exist, it will be created. Allows an Agent to set a property of a specified Agent. If the property does not already exist, it will be created. If the current Agent is controlling a

SuperCollider node, the system will attempt to set an argument of the same name to the proposed valued.

- void setPID (string propertyID, string value, int ID)

  Allows an Agent to set a property of a specified Agent. If the property does not already exist, it will be created. Allows an Agent to set a property of a specified Agent. If the property does not already exist, it will be created. If the current Agent is controlling a SuperCollider node, the system will attempt to set an argument of the same name to the proposed valued.
- void setPID (string propertyID, Vec2 value, int ID)

  Allows an Agent to set a property of a specified Agent. If the property does not already exist, it will be created. Allows an Agent to set a property of a specified Agent. If the property does not already exist, it will be created. If the current Agent is controlling a SuperCollider node, the system will attempt to set an argument of the same name to the proposed valued.
- void setP (table propertyTable)

  Allows an Agent to set a table of properties. Allows an Agent to set a table of new properties. This table must contain key value pairs of data. If property in this table does not exist, a new property will be created for the data. If the current Agent is controlling a SuperCollider node, the system will attempt to set the arguments of the same names to the proposed values.
- void setPID (table propertyTable, int ID)

  Allows an Agent to set a table of properties in a specified Agent. Allows an Agent to set a table of new properties in a specified Agent. This table must contain key value pairs of data. If property in this table does not exist, a new property will be created for the data. If the current Agent is controlling a SuperCollider node, the system will attempt to set arguments of the same name to the proposed values.
- bool getP (string propertyName)

  Allows an Agent to obtain the value of a property. Allows an Agent to obtain the value of a property, which can be a bool, int, float, string or Vec2.
- bool getPID (string propertyName, int ID)

  Allows an Agent to obtain the value of a property which belongs to another Agent. Allows an Agent to obtain the value of a property, belonging to another Agent. This value can be a bool, int, float, string or Vec2.
- void changeMotionModel (string modelType)

  Allows an Agent to switch between types of motion. Allows an Agent to alternate between "Linear", "Physics","Static" and "Path" motion models.
- void changeMotionModelID (string modelType, int ID)

  Allows a specified Agent to switch between types of motion Allows a specified Agent to alternate between "Linear", "Physics","Static" and "Path" motion models.
- void scNew (string synthName, int nodeID)

  Allows an Agent to spawn a Supercollider synth of the provided typename with the provided index. Allows an Agent to spawn a Supercollider synth of the provided typename with the provided index.
- void scSet (string argName, float value, int nodeID)

  Allows an Agent to set the argument of a SuperCollider node. Allows an Agent to set the argument of a SuperCollider node.
- void scFree (int nodeID)

  Kills the SuperCollider node with the ID provided. Kills the SuperCollider node with the ID provided.

- int getNearest (Vec2 point)

  Allows an Agent to obtain the ID of another Agent, closest to the provided position. Allows an Agent to obtain the ID of another Agent, closest to the provided position.
- int getNearest ()

  Allows an Agent to obtain the ID of another Agent, closest to the provided position. Allows an Agent to obtain the ID of the closest other Agent.
- int getNearestID (int ID)

  Allows you to obtain the closest Agent to a specified Agent. Allows you to obtain the closest Agent to a specified Agent.
- void drawRelativeID (int screenIndex, Vec2 relativePos, int r, int g, int b, int ID)

  Allows the drawing of colour data on a screen canvas (whose ID must be provided). Allows the drawing of colour data on a screen canvas (whose ID must be provided) in coordinates relative to the Agent whose ID is provided.
- void drawRelativeID (int screenIndex, Vec2 relativePos, RGBA colourData, int I-D)

  Allows the drawing of colour data on a screen canvas (whose ID must be provided). Allows the drawing of colour data on a screen canvas (whose ID must be provided) in coordinates relative to the Agent whose ID is provided.
- void drawRelative (int screenIndex, Vec2 relativePos, int r, int g, int b, int ID)

  Allows the drawing of colour data on a screen canvas (whose ID must be provided). Allows the drawing of colour data on a screen canvas (whose ID must be provided) in coordinates relative to the currently active Agent.
- void drawRelative (int screenIndex, Vec2 relativePos, RGBA colourData, int ID)

  Allows the drawing of colour data on a screen canvas (whose ID must be provided). Allows the drawing of colour data on a screen canvas (whose ID must be provided) in coordinates relative to the currently active Agent.
- int getRedFromPoint (int screenIndex, Vec2 point)

  Allows an Agent to extract the red component of a point on a given screen canvas. Allows an Agent to extract the red component of a point on a given screen canvas.
- int getGreenFromPoint (int screenIndex, Vec2 point)

  Allows an Agent to extract the green component of a point on a given screen canvas. Allows an Agent to extract the green component of a point on a given screen canvas.
- int getBlueFromPoint (int screenIndex, Vec2 point)

  Allows an Agent to extract the blue component of a point on a given screen canvas. Allows an Agent to extract the blue component of a point on a given screen canvas.
- int getTotalFromPoint (int screenIndex, Vec2 point)

  Allows the extraction of the total/sum of all three colour components at a given point. Allows the extraction of the total/sum of all three colour components at a given point.
- float getMeanFromPoint (int screenIndex, Vec2 point)

  Allows the extraction of the mean/average of all three colour components at a given point. Allows the extraction of the mean/average of all three colour components at a given point.
- int getR (int screenIndex)

  Returns the red component at an Agents current position. Returns the red component at an Agents current position.
- int getRID (int screenIndex, int ID)

Returns the red component at a specified Agents position. Returns the red component at a specified Agents position.

- int getG (int screenIndex)

  Returns the green component at an Agents current position. Returns the green component at an Agents current position.
- int getGID (int canvasIndex, int agentID)

  Returns the green component at a specified Agents position. Returns the green component at a specified Agents position.
- int getB (int canvasIndex)

  Returns the blue component at an Agents current position. Returns the blue component at an Agents current position.
- int getBID (int canvasIndex, int agentID)

  Returns the blue component at a specified Agents position. Returns the blue component at a specified Agents position.
- int getTotalCol (int screenIndex)

  Allows the extraction of the total/sum of all three colour components at the Agents position. Allows the extraction of the total/sum of all three colour components at the Agents position.
- int getTotalID (int screenIndex, int agentID)

  Allows the extraction of the total/sum of all three colour components at a specified - Agents position. Allows the extraction of the total/sum of all three colour components at a specified Agents position.
- int getMeanColID (int agentID)

  Allows the extraction of the mean/average of all three colour components at the position of a specified Agent. Allows the extraction of the mean/average of all three colour components at the position of a specified Agent.
- int getMeanCol ()

  Allows the extraction of the mean/average of all three colour components at the position of the current Agent. Allows the extraction of the mean/average of all three colour components at the position of the current Agent.
- RGBA getRGBA (int screenIndex)

  Allows an Agent to obtain an instance of an RGBA class containing the data from a specified image screen at it's position. Allows an Agent to obtain an instance of an RGBA class containing the data from a specified image screen at it's position.
- RGBA getRGBA (Vec2 position, int screenIndex)

  Allows an Agent to obtain an instance of an RGBA class containing the data from a specified image screen at a position. Allows an Agent to obtain an instance of an RGBA class containing the data from a specified image screen at a position.
- RGBA getRGBAID (int screenIndex, int agentID)

  Allows an Agent to obtain an instance of an RGBA class containing the data from a specified image screen at the position of another Agent. Allows an Agent to obtain an instance of an RGBA class containing the data from a specified image screen at the position of another Agent.
- table getNearby (int radius)

  Allows an Agent to obtain the ID's of all of the Agents within a surrounding radius. Allows an Agent to obtain the ID's of all of the Agents within a surrounding radius.
- table getNearby (int radius, Vec2 position)

Allows an Agent to obtain the ID's of all of the Agents within the surrounding radius of a specified point. Allows an Agent to obtain the ID's of all of the Agents within the surrounding radius of a specified point.

- table getNearbyID (int radius, int agentID)

  Allows an Agent to obtain the ID's of all of the Agents within the surrounding radius of a specified Agent. Allows an Agent to obtain the ID's of all of the Agents within the surrounding radius of a specified Agent.
- Vec2 getWorldSize ()

  Allows an Agent to obtain the size of the current world. Allows an Agent to obtain the size of the current world.
- void oscSF (string ipAddress, string portNumber, string path, string stringData, float number)

  Allows the sending of a string and a float via OSC to a specified IP address, port and path. (Deprecated please use 'send' instead) Allows the sending of a string and a float via OSC to a specified IP address, port and path. (Deprecated please use 'send' instead)
- void oscF (string ipAddress, string portNumber, string path, float number)

  Allows the sending of float via OSC to a specified IP address, port and path. (-Deprecated please use 'send' instead) Allows the sending of a float via OSC to a specified IP address, port and path. (Deprecated please use 'send' instead)
- void oscSI (string ipAddress, string portNumber, string path, string stringData, int number)

  Allows the sending of a string and an integer via OSC to a specified IP address, port and path. (Deprecated please use 'send' instead) Allows the sending of a string and an integer via OSC to a specified IP address, port and path. (Deprecated please use 'send' instead)
- void oscI (string ipAddress, string portNumber, string path, int number)

  Allows the sending of float via OSC to a specified IP address, port and path. (-Deprecated please use 'send' instead) Allows the sending of an integer via OSC to a specified IP address, port and path. (Deprecated please use 'send' instead)
- void oscSS (string ipAddress, string portNumber, string path, string stringData, string secondStringData)

  Allows the sending of two string via OSC to a specified IP address, port and path. (Deprecated please use 'send' instead) Allows the sending of a pair of strings via OSC to a specified IP address, port and path. (Deprecated please use 'send' instead)
- void oscS (string ipAddress, string portNumber, string stringData)

  Allows the sending of a string via OSC to a specified IP address, port and path. (-Deprecated please use 'send' instead) Allows the sending of a string via OSC to a specified IP address, port and path. (Deprecated please use 'send' instead)
- void killChildren ()

  Kills all of the current Agent's children. Kills all of the current Agent's children.
- void killChildrenID (int agentID)

  Kills all of the specified Agents children. Kills all of the specified Agents children.
- void addPathPoint (int pathIndex, int xPos, int yPos)

  Allows an Agent to add a point to a path. If a path with the specified index does not exist it will create a new one. Allows an Agent to add a point to a path. If a path with the specified index does not exist it will create a new one.
- void addPathPoint (int pathIndex, int Vec2)

Allows an Agent to add a point to a path. If a path with the specified index does not exist it will create a new one. Allows an Agent to add a point to a path. If a path with the specified index does not exist it will create a new one.

- void removePathPoint (int pathIndex, int pointIndex)

  Allows an Agent to remove a point in a path. Allows an Agent to remove a point in a path.
- void movePathPoint (int pathIndex, int pointIndex, int xPos, int yPos)

  Allows an Agent to move a point in a path. Allows an Agent to move a point in a path.
- void movePathPoint (int pathIndex, int pointIndex, Vec2 position)

  Allows an Agent to move a point in a path. Allows an Agent to move a point in a path.
- int addPath (int xPos, int yPos)

  Allows an Agent to create a new path. A single point must be provided. Allows an Agent to create a new path. A single point must be provided.
- int addPath (Vec2 position)

  Allows an Agent to create a new path. A single point must be provided. Allows an Agent to create a new path. A single point must be provided.
- void removePath (int pathIndex)

  Allows an Agent to remove/delete a path with a given index. Allows an Agent to remove/delete a path with a given index.
- void movePath (int pathIndex, int xDelta, int yDelta)

  Allows an Agent to move a path by a delta. Allows an Agent to move a path by a delta.
- void movePath (int pathIndex, Vec2 delta)

  Allows an Agent to move a path by a delta. Allows an Agent to move a path by a delta.
- bool hasP (string propertyName)

  Allows an Agent to check whether it has a property with a specified name. Allows an Agent to check whether it has a property with a specified name.
- bool hasPID (string propertyName)

  Allows an Agent to check whether another Agent contains a property with a specified name. Allows an Agent to check whether another Agent contains a property with a specified name.
- string getType ()

  Allows an Agent to obtain its own type. Allows an Agent to obtain its own type.
- string getTypeID ()

  Allows an Agent to obtain the type of another Agent. Allows an Agent to obtain the type of another Agent.
- void send (string ipAddress, string portNumber, string path,...data)

  Allows an Agent to send OSC data to a specified IPAddress, port and path. - Allows an Agent to send OSC data to a specified IPAddress, port and path. A user can append as many pieces of data to send as they wish. This data can be numbers, strings and tables containing key value pairs of data. e.g send("127.0.0.-1","8000","/example",100,"hello",3.14159,{more=info,data=200})
- int currentScreen ()

  Allows an Agent to obtain the index of the currently visible screen. Allows an Agent to obtain the index of the currently visible screen.
- void drawLine (int screenIndex, Vec2 startPosition, Vec2 endPosition, int red, int green, int blue)

Allows an Agent to draw a line between two points on a specified screen. Allows an Agent to draw a line between two points on a specified screen.

- void drawLine (int screenIndex, Vec2 startPosition, Vec2 endPosition, RGBA colourData)

  Allows an Agent to draw a line between two points on a specified screen. Allows an Agent to draw a line between two points on a specified screen.
- void drawRect (int screenIndex, Vec2 bottomLeft, Vec2 topRight, int red, int green, int blue, int thickness)

  Allows an Agent to draw a rectangle on a specified screen. Allows an Agent to draw a rectangle on a specified screen.
- void drawRect (int screenIndex, Vec2 bottomLeft, Vec2 topRight, RGBA colourData, int thickness)

  Allows an Agent to draw a rectangle on a specified screen. Allows an Agent to draw a rectangle on a specified screen.
- void drawCircle (int screenIndex, Vec2 centrePosition, int radius, int red, int green, int blue, int thickness)

  Allows an Agent to draw a circle on a specified screen. Allows an Agent to draw a circle on a specified screen.
- void drawCircle (int screenIndex, Vec2 centrePosition, int radius, RGBA colourData, int thickness)

  Allows an Agent to draw a circle on a specified screen. Allows an Agent to draw a circle on a specified screen.

4.1.1 Detailed Description

4.1.2 Function Documentation

4.1.2.1 void addP (string *propertyID,* bool *value*)

Allows an Agent to add a new property. Allows an Agent to add a new property. The property can be of the following types: bool, float, int, string, Vec2.

Returns

void

Parameters

propertyID	The name of the new property
value	The initial value of the property

4.1.2.2 void addP (string *propertyID,* int *value*)

Allows an Agent to add a new property. Allows an Agent to add a new property. The property can be of the following types: bool, float, int, string, Vec2.

Returns

void

Parameters

propertyID	The name of the new property
value	The initial value of the property

4.1.2.3 void addP (string *propertyID*, float *value*)

Allows an Agent to add a new property. Allows an Agent to add a new property. The property can be of the following types: bool, float, int, string, Vec2.

Returns

void

Parameters

propertyID	The name of the new property
value	The initial value of the property

4.1.2.4 void addP (string *propertyID*, string *value*)

Allows an Agent to add a new property. Allows an Agent to add a new property. The property can be of the following types: bool, float, int, string, Vec2.

Returns

void

Parameters

propertyID	The name of the new property
value	The initial value of the property

4.1.2.5 void addP (string *propertyID*, Vec2 *value*)

Allows an Agent to add a new property. Allows an Agent to add a new property. The property can be of the following types: bool, float, int, string, Vec2.

Returns

void

Parameters

propertyID	The name of the new property
value	The initial value of the property

4.1.2.6 void addP (table *propertyTable*)

Allows an Agent to add table of new properties. Allows an Agent to add table of new properties in key value pairs {key=value}. The property can be of the following types: bool, float, int, string, Vec2.

Returns

void

Parameters

property-Table	A table of property key value pairs

4.1.2.7 int addPath (int *xPos,* int *yPos*)

Allows an Agent to create a new path. A single point must be provided. Allows an Agent to create a new path. A single point must be provided.

Returns

The index of the newly created path

Parameters

xPos	Cartesian X coordinate
yPos	Cartesian Y coordinate

4.1.2.8 int addPath (Vec2 *position*)

Allows an Agent to create a new path. A single point must be provided. Allows an Agent to create a new path. A single point must be provided.

Returns

The index of the newly created path

Parameters

position	First point of the new path

4.1.2.9 void addPathPoint (int *pathIndex,* int *xPos,* int *yPos*)

Allows an Agent to add a point to a path. If a path with the specified index does not exist it will create a new one. Allows an Agent to add a point to a path. If a path with the specified index does not exist it will create a new one.

Returns

void

Parameters

pathIndex	Index of the path to use
xPos	Cartesian X coordinate
yPos	Cartesian Y coordinate

4.1.2.10 void addPathPoint (int *pathIndex,* int *Vec2*)

Allows an Agent to add a point to a path. If a path with the specified index does not exist it will create a new one. Allows an Agent to add a point to a path. If a path with the specified index does not exist it will create a new one.

Returns

void

Parameters

pathIndex	Index of the path to use
Vec2	Point to add

4.1.2.11 void addPID (string *propertyID,* bool *value,* int *ID*)

Allows an Agent to add a new property to another Agent. Allows an Agent to add a new property to another Agent. The property can be of the following types: bool, float, int, string, Vec2.

Returns

void

Parameters

propertyID	The name of the new property
value	The initial value of the property
ID	ID for the specified Agent

4.1.2.12 void addPID (string *propertyID,* int *value,* int *ID*)

Allows an Agent to add a new property to another Agent. Allows an Agent to add a new property to another Agent. The property can be of the following types: bool, float, int, string, Vec2.

Returns

void

Parameters

propertyID	The name of the new property
value	The initial value of the property
ID	ID for the specified Agent

4.1.2.13 void addPID (string *propertyID,* float *value,* int *ID*)

Allows an Agent to add a new property to another Agent. Allows an Agent to add a new property to another Agent. The property can be of the following types: bool, float, int, string, Vec2.

Returns

void

Parameters

propertyID	The name of the new property
value	The initial value of the property
ID	ID for the specified Agent

4.1.2.14 void addPID (string *propertyID,* string *value,* int *ID*)

Allows an Agent to add a new property to another Agent. Allows an Agent to add a new property to another Agent. The property can be of the following types: bool, float, int, string, Vec2.

Returns

void

Parameters

propertyID	The name of the new property
value	The initial value of the property
ID	ID for the specified Agent

4.1.2.15 void addPID (string *propertyID,* Vec2 *value,* int *ID*)

Allows an Agent to add a new property to another Agent. Allows an Agent to add a new property to another Agent. The property can be of the following types: bool, float, int, string, Vec2.

Returns

void

Parameters

propertyID	The name of the new property
value	The initial value of the property
ID	ID for the specified Agent

4.1.2.16 void addPID (table *propertyTable,* int *ID*)

Allows an Agent to add table of new properties to another Agent. Allows an Agent to add table of new properties in key value pairs {key=value} to another Agent. The property can be of the following types: bool, float, int, string, Vec2.

Returns

void

Parameters

property-Table	A table of property key value pairs
ID	ID for the specified Agent

4.1.2.17 void changeMotionModel (string *modelType*)

Allows an Agent to switch between types of motion. Allows an Agent to alternate between "Linear", "Physics","Static" and "Path" motion models.

Returns

void

Parameters

modelType	String representing the type of motion model

4.1.2.18 void changeMotionModelID (string *modelType,* int *ID*)

Allows a specified Agent to switch between types of motion Allows a specified Agent to alternate between "Linear", "Physics","Static" and "Path" motion models.

Returns

void

Parameters

modelType	String representing the type of motion model
ID	ID for the specified Agent

4.1.2.19 int currentScreen ( )

Allows an Agent to obtain the index of the currently visible screen. Allows an Agent to obtain the index of the currently visible screen.

Returns

Index of the currently visible screen

4.1.2.20 void die ( )

Kills the currently active Agent. Kills the currently active Agent.

Returns

void

4.1.2.21 void drawCircle (int *screenIndex,* Vec2 *centrePosition,* int *radius,* int *red,* int *green,* int *blue,* int *thickness*)

Allows an Agent to draw a circle on a specified screen. Allows an Agent to draw a circle on a specified screen.

Returns

void

Parameters

screenIndex	Index of the screen to draw on
centre-Position	Centre of the circle
radius	Radius of the circle
red	Amount of red to draw
green	Amount of green to draw
blue	Amount of blue to draw
thickness	Thickness of the border

4.1.2.22 void drawCircle (int *screenIndex,* Vec2 *centrePosition,* int *radius,* RGBA *colourData,* int *thickness*)

Allows an Agent to draw a circle on a specified screen. Allows an Agent to draw a circle on a specified screen.

Returns

void

Parameters

screenIndex	Index of the screen to draw on
centre-Position	Centre of the circle
radius	Radius of the circle
colourData	Colour data to draw
thickness	Thickness of the border

4.1.2.23 void drawLine (int *screenIndex,* Vec2 *startPosition,* Vec2 *endPosition,* int *red,* int *green,* int *blue*)

Allows an Agent to draw a line between two points on a specified screen. Allows an Agent to draw a line between two points on a specified screen.

Returns

void

Parameters

screenIndex	Index of the screen to draw on
startPosition	Starting position
endPosition	End Position
red	Amount of red to draw
green	Amount of green to draw
blue	Amount of blue to draw

4.1.2.24 void drawLine (int *screenIndex,* Vec2 *startPosition,* Vec2 *endPosition,* RGBA *colourData*)

Allows an Agent to draw a line between two points on a specified screen. Allows an Agent to draw a line between two points on a specified screen.

Returns

void

Parameters

screenIndex	Index of the screen to draw on
startPosition	Starting position
endPosition	End Position
colourData	Colour data to draw

4.1.2.25 void drawRect (int *screenIndex,* Vec2 *bottomLeft,* Vec2 *topRight,* int *red,* int *green,* int *blue,* int *thickness*)

Allows an Agent to draw a rectangle on a specified screen. Allows an Agent to draw a rectangle on a specified screen.

Returns

void

Parameters

screenIndex	Index of the screen to draw on
bottomLeft	Bottom left of the rectangle
topRight	Top right of the rectangle
red	Amount of red to draw
green	Amount of green to draw
blue	Amount of blue to draw
thickness	Thickness of the border

4.1.2.26 void drawRect (int *screenIndex,* Vec2 *bottomLeft,* Vec2 *topRight,* RGBA *colourData,* int *thickness*)

Allows an Agent to draw a rectangle on a specified screen. Allows an Agent to draw a rectangle on a specified screen.

Returns

void

Parameters

screenIndex	Index of the screen to draw on
bottomLeft	Bottom left of the rectangle
topRight	Top right of the rectangle
colourData	Colour data to draw
thickness	Thickness of the border

4.1.2.27 void drawRelative (int *screenIndex,* Vec2 *relativePos,* int *r,* int *g,* int *b,* int *ID*)

Allows the drawing of colour data on a screen canvas (whose ID must be provided). Allows the drawing of colour data on a screen canvas (whose ID must be provided) in coordinates relative to the currently active Agent.

Returns

void

Parameters

screenIndex	Index of the screen to draw on
relativePos	The relative position to draw
r	Red component
g	Green component
b	Blue component
ID	ID for the specified Agent

4.1.2.28 void drawRelative (int *screenIndex,* Vec2 *relativePos,* RGBA *colourData,* int *ID*)

Allows the drawing of colour data on a screen canvas (whose ID must be provided). Allows the drawing of colour data on a screen canvas (whose ID must be provided) in coordinates relative to the currently active Agent.

Returns

void

Parameters

screenIndex	Index of the screen to draw on
relativePos	The relative position to draw
colourData	Colour data to draw
ID	ID for the specified Agent

4.1.2.29 void drawRelativeID (int *screenIndex,* Vec2 *relativePos,* int *r,* int *g,* int *b,* int *ID*)

Allows the drawing of colour data on a screen canvas (whose ID must be provided). Allows the drawing of colour data on a screen canvas (whose ID must be provided) in coordinates relative to the Agent whose ID is provided.

Returns

void

Parameters

screenIndex	Index of the screen to draw on
relativePos	The relative position to draw
r	Red component
g	Green component
b	Blue component
ID	ID for the specified Agent

4.1.2.30 void drawRelativeID (int *screenIndex,* Vec2 *relativePos,* RGBA *colourData,* int *ID*)

Allows the drawing of colour data on a screen canvas (whose ID must be provided). Allows the drawing of colour data on a screen canvas (whose ID must be provided) in coordinates relative to the Agent whose ID is provided.

Returns

void

Parameters

screenIndex	Index of the screen to draw on
relativePos	The relative position to draw
colourData	Colour data to draw
ID	ID for the specified Agent

4.1.2.31 int getB (int *canvasIndex*)

Returns the blue component at an Agents current position. Returns the blue component at an Agents current position.

Returns

Returns the blue component at an Agents current position.

Parameters

canvasIndex	Index of the screen

4.1.2.32 int getBID (int *canvasIndex*, int *agentID*)

Returns the blue component at a specified Agents position. Returns the blue component at a specified Agents position.

Returns

Returns the blue component at a specified Agents position.

Parameters

canvasIndex	Index of the screen
agentID	ID for the specified Agent

4.1.2.33 int getBlueFromPoint (int *screenIndex*, Vec2 *point*)

Allows an Agent to extract the blue component of a point on a given screen canvas. Allows an Agent to extract the blue component of a point on a given screen canvas.

Returns

The green component of a given point.

Parameters

screenIndex	Index of the screen
point	The point in question

4.1.2.34 int getChild (int *childIndex*)

Allows you to obtain the unique ID of a child Agent. Allows you to obtain the unique ID of a child Agent through a relative index. For example, you can obtain the second child through getChild(1)

Returns

ID of a child Agent.

Parameters

childIndex	The index of the child

4.1.2.35 int getChildID (int *childID,* int *parentID*)

Allows you to obtain the child of another Agent. Allows you to obtain a child from any Agent if its ID is known.

Returns

ID of a child Agent

Parameters

childID	The child ID relative to the parent.
parentID	The ID of the parent.

4.1.2.36 float getDT ( )

A utility and timing function that provides the delta between the current frame and the previous (i.e time elapsed). Delta between now and previous frame.

Returns

The delta between now and the previous frame.

4.1.2.37 int getG (int *screenIndex*)

Returns the green component at an Agents current position. Returns the green component at an Agents current position.

Returns

Returns the green component at an Agents current position.

Parameters

screenIndex	Index of the screen

4.1.2.38 int getGID (int *canvasIndex,* int *agentID*)

Returns the green component at a specified Agents position. Returns the green component at a specified Agents position.

Returns

Returns the green component at a specified Agents position.

Parameters

canvasIndex	Index of the screen
agentID	ID for the specified Agent

4.1.2.39 int getGreenFromPoint (int *screenIndex*, Vec2 *point*)

Allows an Agent to extract the green component of a point on a given screen canvas. Allows an Agent to extract the green component of a point on a given screen canvas.

Returns

The green component of a given point.

Parameters

screenIndex	Index of the screen
point	The point in question

4.1.2.40 int getID ( )

A crucial function allowing an Agent to obtain its own ID. Provides the unique identifier for the currently active Agent.

Returns

The unique identifier of the currently active Agent.

4.1.2.41 float getLifetime ( )

Allows an Agent to obtain how long it has been alive in seconds. Allows an Agent to obtain how long it has been alive in seconds.

Returns

The time which the Agent has been alive

4.1.2.42 float getLifetimeID (int *ID*)

Allows an Agent to obtain how long another Agent has been alive in seconds. Allows an Agent to obtain how long another Agent has been alive in seconds.

Returns

The time which the specified Agent has been alive

Parameters

ID	ID of the Agent

4.1.2.43 int getMeanCol ( )

Allows the extraction of the mean/average of all three colour components at the position of the current Agent. Allows the extraction of the mean/average of all three colour components at the position of the current Agent.

Returns

The mean/average across all three colour components.

4.1.2.44 int getMeanColID (int *agentID*)

Allows the extraction of the mean/average of all three colour components at the position of a specified Agent. Allows the extraction of the mean/average of all three colour components at the position of a specified Agent.

Returns

The mean/average across all three colour components.

Parameters

agentID	ID of the specified Agent

4.1.2.45 float getMeanFromPoint (int *screenIndex,* Vec2 *point*)

Allows the extraction of the mean/average of all three colour components at a given point. Allows the extraction of the mean/average of all three colour components at a given point.

Returns

The mean/average across all three colour components.

Parameters

screenIndex	Index of the screen
point	The point in question

4.1.2.46 table getNearby (int *radius*)

Allows an Agent to obtain the ID's of all of the Agents within a surrounding radius. Allows an Agent to obtain the ID's of all of the Agents within a surrounding radius.

Returns

Table containing the ID's of Agents within a radius.

Parameters

radius	Radius of the area to search

4.1.2.47 table getNearby (int *radius,* Vec2 *position*)

Allows an Agent to obtain the ID's of all of the Agents within the surrounding radius of a specified point. Allows an Agent to obtain the ID's of all of the Agents within the surrounding radius of a specified point.

Returns

Table containing the ID's of Agents within a radius of a position.

Parameters

radius	Radius of the area to search
position	Centre position of the area to search

4.1.2.48 table getNearbyID (int *radius,* int *agentID*)

Allows an Agent to obtain the ID's of all of the Agents within the surrounding radius of a specified Agent. Allows an Agent to obtain the ID's of all of the Agents within the surrounding radius of a specified Agent.

Returns

Table containing the ID's of Agents within a radius around the specified Agent.

Parameters

radius	Radius of the area to search
agentID	ID for the specified Agent

4.1.2.49 int getNearest (Vec2 *point*)

Allows an Agent to obtain the ID of another Agent, closest to the provided position. Allows an Agent to obtain the ID of another Agent, closest to the provided position.

Returns

The ID of the Agent closest

Parameters

point	The point you wish to test against.

4.1.2.50 int getNearest ( )

Allows an Agent to obtain the ID of another Agent, closest to the provided position. Allows an Agent to obtain the ID of the closest other Agent.

Returns

The ID of the Agent closest

4.1.2.51 int getNearestID (int *ID*)

Allows you to obtain the closest Agent to a specified Agent. Allows you to obtain the closest Agent to a specified Agent.

Returns

The ID of the Agent closest

Parameters

ID	ID for the specified Agent

4.1.2.52 Vec2 getP (string *propertyName*)

Allows an Agent to obtain the value of a property. Allows an Agent to obtain the value of a property, which can be a bool, int, float, string or Vec2.

Returns

Returns the value of the property specified

Parameters

property-Name	The name of the properties whose value you want to obtain

4.1.2.53 int getParent ( )

Allows an Agent to obtain the ID of its parent. Allows an Agent to obtain the ID of its parent.

Returns

The ID of the Agents parent.

4.1.2.54 int getParentID (int *childID*)

Allows an Agent to obtain the ID of a parent of another if its ID is known. Allows an Agent to obtain the ID of a parent of another if its ID is known.

Returns

The ID of Parent for a specified Agent

Parameters

childID	The ID of the child whose parent you wish to obtain.

4.1.2.55 Vec2 getPID (string *propertyName,* int *ID*)

Allows an Agent to obtain the value of a property which belongs to another Agent. - Allows an Agent to obtain the value of a property, belonging to another Agent. This value can be a bool, int, float, string or Vec2.

Returns

Returns the value of the property specified

Parameters

property-Name	The name of the properties whose value you want to obtain
ID	ID for the specified Agent

4.1.2.56 Vec2 getPosition ( )

Allows an Agent to quickly obtain its position. Allows an Agent to quickly obtain its position.

Returns

The position of the currently active Agent.

4.1.2.57 Vec2 getPositionID (int *ID*)

Allows Agents to quickly obtain the position of any given Agent. Allows Agents to quickly obtain the position of any given Agent.

Returns

The position of the specified Agent

Parameters

ID	The ID of the Agent in question

4.1.2.58 int getR (int *screenIndex*)

Returns the red component at an Agents current position. Returns the red component at an Agents current position.

Returns

Returns the red component at an Agents current position.

Parameters

screenIndex	Index of the screen

4.1.2.59 int getRedFromPoint (int *screenIndex,* Vec2 *point*)

Allows an Agent to extract the red component of a point on a given screen canvas. Allows an Agent to extract the red component of a point on a given screen canvas.

Returns

The red component of a given point.

Parameters

screenIndex	Index of the screen
point	The point in question

4.1.2.60 RGBA getRGBA (int *screenIndex*)

Allows an Agent to obtain an instance of an RGBA class containing the data from a specified image screen at it's position. Allows an Agent to obtain an instance of an RGBA class containing the data from a specified image screen at it's position.

Returns

Colour data at the Agents current position

Parameters

screenIndex	Index of the screen

4.1.2.61 RGBA getRGBA (Vec2 *position,* int *screenIndex*)

Allows an Agent to obtain an instance of an RGBA class containing the data from a specified image screen at a position. Allows an Agent to obtain an instance of an RGBA class containing the data from a specified image screen at a position.

Returns

Colour data at a position

Parameters

position	position of the data to extract
screenIndex	Index of the screen

4.1.2.62 RGBA getRGBAID (int *screenIndex,* int *agentID*)

Allows an Agent to obtain an instance of an RGBA class containing the data from a specified image screen at the position of another Agent. Allows an Agent to obtain an instance of an RGBA class containing the data from a specified image screen at the position of another Agent.

Returns

Colour data at the position of the specified Agent

Parameters

screenIndex	Index of the screen
agentID	ID for the specified Agent

4.1.2.63 int getRID (int *screenIndex,* int *ID*)

Returns the red component at a specified Agents position. Returns the red component at a specified Agents position.

Returns

Returns the red component at a specified Agents position.

Parameters

screenIndex	Index of the screen
ID	ID for the specified Agent

4.1.2.64 int getTotalCol (int *screenIndex*)

Allows the extraction of the total/sum of all three colour components at the Agents position. Allows the extraction of the total/sum of all three colour components at the Agents position.

Returns

The total (sum) of the colour components

Parameters

screenIndex	Index of the screen

4.1.2.65 int getTotalFromPoint (int *screenIndex,* Vec2 *point*)

Allows the extraction of the total/sum of all three colour components at a given point. Allows the extraction of the total/sum of all three colour components at a given point.

Returns

The total (sum) of the colour components

Parameters

screenIndex	Index of the screen
point	The point in question

4.1.2.66 int getTotalID (int *screenIndex,* int *agentID*)

Allows the extraction of the total/sum of all three colour components at a specified - Agents position. Allows the extraction of the total/sum of all three colour components at a specified Agents position.

Returns

The total (sum) of the colour components

Parameters

screenIndex	Index of the screen
agentID	ID for the specified Agent

4.1.2.67 string getType ( )

Allows an Agent to obtain its own type. Allows an Agent to obtain its own type.

Returns

The type of Agent (the name)

4.1.2.68 string getTypeID ( )

Allows an Agent to obtain the type of another Agent. Allows an Agent to obtain the type of another Agent.

Returns

The type of Agent (the name)

4.1.2.69 Vec2 getWorldSize ( )

Allows an Agent to obtain the size of the current world. Allows an Agent to obtain the size of the current world.

Returns

The size of the current world

4.1.2.70 bool hasChildren ( )

Returns whether the current Agent has any children. Returns whether the current Agent has any children.

Returns

Whether the current Agent has any children

4.1.2.71 bool hasChildrenID (int *ID*)

Indicates whether an Agent, specified by an ID has any children. Indicates whether an Agent, specified by an ID has any children.

Returns

Returns whether an Agent has any children.

Parameters

ID	ID of the Agent

4.1.2.72 bool hasP (string *propertyName*)

Allows an Agent to check whether it has a property with a specified name. Allows an Agent to check whether it has a property with a specified name.

Returns

Returns whether a property exists in the current Agent

Parameters

property-Name	Name of the property to check

4.1.2.73 bool hasPID (string *propertyName*)

Allows an Agent to check whether another Agent contains a property with a specified name. Allows an Agent to check whether another Agent contains a property with a specified name.

Returns

Returns whether a property exists in the specified Agent

Parameters

property-Name	Name of the property to check

4.1.2.74 void initai ( )

This MUST be called at the start of each Agent script. This initializes the parent script so that it is interpreted as an Agent by the system.

Returns

void

4.1.2.75 void initP (table *propertyTable*)

Initializes the Agent with a suite of provided properties. Allows a function to be intialized as an Agent with a set of properties that are provided in a table of key value pairs {key=value}.

Returns

void

Parameters

property-Table	A table of property key value pairs

4.1.2.76 void initPID (table *propertyTable,* int *ID*)

Allows an Agent to initialize another Agent with a suite of properties. Allows an Agent to initialize another Agent with a suite of properties that are stored in a table of key value pairs {key=value}.

Returns

void

Parameters

property-Table	A table of property key value pairs
ID	ID for the specified Agent

4.1.2.77 void killChildren ( )

Kills all of the current Agent's children. Kills all of the current Agent's children.

Returns

void

4.1.2.78 void killChildrenID (int *agentID*)

Kills all of the specified Agents children. Kills all of the specified Agents children.

Returns

void

Parameters

agentID	ID for the specified Agent

4.1.2.79 void killID (int *agentID*)

Allows an Agent to kill another Agent if its ID is known. Allows an Agent to kill another Agent if its ID is known.

Returns

void

Parameters

agentID	The ID of the Agent you wish to kill.

4.1.2.80 void move (int *x*, int *y*)

Allows an Agent to move by a specified amount. Allows an Agent to move by a specified amount.

Returns

void

Parameters

x	Amount to move in the x plane
y	Amount to move in the y plane

4.1.2.81 void move (Vec2 *delta*)

Allows an Agent to move by a specified amount. Allows an Agent to move by a specified amount.

Returns

void

Parameters

delta	Delta to move by

4.1.2.82 void moveID (int *x*, int *y*, int *ID*)

Allows an Agent to move by a specified amount. Allows an Agent to move by a specified amount.

Returns

void

Parameters

x	Amount to move in the x plane
y	Amount to move in the y plane
ID	The ID of the Agent

4.1.2.83 void moveID (Vec2 *delta*, int *ID*)

Allows an Agent to move by a specified amount. Allows an Agent to move by a specified amount.

Returns

void

Parameters

delta	Delta to move the specified Agent by
ID	The ID of the Agent

4.1.2.84 void movePath (int *pathIndex*, int *xDelta*, int *yDelta*)

Allows an Agent to move a path by a delta. Allows an Agent to move a path by a delta.

Returns

void

Parameters

pathIndex	Index of the path to move
xDelta	Cartesian X delta
yDelta	Cartesian Y delta

4.1.2.85 void movePath (int *pathIndex,* Vec2 *delta*)

Allows an Agent to move a path by a delta. Allows an Agent to move a path by a delta.

Returns

void

Parameters

pathIndex	Index of the path to move
delta	Delta to move the path by

4.1.2.86 void movePathPoint (int *pathIndex,* int *pointIndex,* int *xPos,* int *yPos*)

Allows an Agent to move a point in a path. Allows an Agent to move a point in a path.

Returns

void

Parameters

pathIndex	Index of the path to use
pointIndex	Index of the point to move int the path
xPos	Cartesian X coordinate
yPos	Cartesian Y coordinate

4.1.2.87 void movePathPoint (int *pathIndex,* int *pointIndex,* Vec2 *position*)

Allows an Agent to move a point in a path. Allows an Agent to move a point in a path.

Returns

void

Parameters

pathIndex	Index of the path to use
pointIndex	Index of the point to move int the path
position	Position to move the point to

4.1.2.88 int numberOfChildren ( )

Returns the number of children of the currently active Agent Returns the number of children of the currently active Agent.

Returns

The number of children.

4.1.2.89 int numberOfChildrenID (int *ID*)

Provides access to the number of children of a specified Agent. Provides access to the number of children of a specified Agent.

Returns

The number of children.

Parameters

ID	Parent ID

4.1.2.90 void oscF (string *ipAddress,* string *portNumber,* string *path,* float *number*)

Allows the sending of float via OSC to a specified IP address, port and path. (-Deprecated please use 'send' instead) Allows the sending of a float via OSC to a specified IP address, port and path. (Deprecated please use 'send' instead)

Returns

void

Parameters

ipAddress	IP address of the machine you wish to connect to
portNumber	The port you wish to connect to
path	The path you wish to send via
number	number you wish to send

4.1.2.91 void oscI (string *ipAddress,* string *portNumber,* string *path,* int *number*)

Allows the sending of float via OSC to a specified IP address, port and path. (-Deprecated please use 'send' instead) Allows the sending of an integer via OSC to a specified IP address, port and path. (Deprecated please use 'send' instead)

Returns

void

Parameters

ipAddress	IP address of the machine you wish to connect to
portNumber	The port you wish to connect to
path	The path you wish to send via
number	number you wish to send

4.1.2.92 void oscS (string *ipAddress,* string *portNumber,* string *stringData*)

Allows the sending of a string via OSC to a specified IP address, port and path. (-Deprecated please use 'send' instead) Allows the sending of a string via OSC to a specified IP address, port and path. (Deprecated please use 'send' instead)

Returns

void

Parameters

ipAddress	IP address of the machine you wish to connect to
portNumber	The port you wish to connect to
stringData	string of data

4.1.2.93 void oscSF (string *ipAddress,* string *portNumber,* string *path,* string *stringData,* float *number*)

Allows the sending of a string and a float via OSC to a specified IP address, port and path. (Deprecated please use 'send' instead) Allows the sending of a string and a float via OSC to a specified IP address, port and path. (Deprecated please use 'send' instead)

Returns

void

Parameters

ipAddress	IP address of the machine you wish to connect to
portNumber	The port you wish to connect to
path	The path you wish to send via
stringData	string containing the potential parameter name
number	number you wish to send

4.1.2.94 void oscSI (string *ipAddress,* string *portNumber,* string *path,* string *stringData,* int *number*)

Allows the sending of a string and an integer via OSC to a specified IP address, port and path. (Deprecated please use 'send' instead) Allows the sending of a string and an integer via OSC to a specified IP address, port and path. (Deprecated please use 'send' instead)

Returns

void

Parameters

ipAddress	IP address of the machine you wish to connect to
portNumber	The port you wish to connect to
path	The path you wish to send via
stringData	string containing the potential parameter name
number	number you wish to send

4.1.2.95 void oscSS (string *ipAddress,* string *portNumber,* string *path,* string *stringData,* string *secondStringData*)

Allows the sending of two string via OSC to a specified IP address, port and path. (-Deprecated please use 'send' instead) Allows the sending of a pair of strings via OSC to a specified IP address, port and path. (Deprecated please use 'send' instead)

Returns

void

Parameters

ipAddress	IP address of the machine you wish to connect to
portNumber	The port you wish to connect to
path	The path you wish to send via
stringData	string containing the potential parameter name
second-StringData	string of data

4.1.2.96 void removePath (int *pathIndex*)

Allows an Agent to remove/delete a path with a given index. Allows an Agent to remove/delete a path with a given index.

Returns

void

Parameters

pathIndex	Index of the path to remove

4.1.2.97 void removePathPoint (int *pathIndex,* int *pointIndex*)

Allows an Agent to remove a point in a path. Allows an Agent to remove a point in a path.

Returns

void

Parameters

pathIndex	Index of the path to use
pointIndex	Index of the point to remove from the path

4.1.2.98 void scFree (int *nodeID*)

Kills the SuperCollider node with the ID provided. Kills the SuperCollider node with the ID provided.

Returns

void

Parameters

nodeID	The SuperCollider NodeId you wish to kill.

4.1.2.99 void scNew (string *synthName,* int *nodeID*)

Allows an Agent to spawn a Supercollider synth of the provided typename with the provided index. Allows an Agent to spawn a Supercollider synth of the provided typename with the provided index.

Returns

void

Parameters

synthName	The name of the synth
nodeID	The SuperCollider NodeId you wish to use.

4.1.2.100 void scSet (string *argName,* float *value,* int *nodeID*)

Allows an Agent to set the argument of a SuperCollider node. Allows an Agent to set the argument of a SuperCollider node.

Returns

void

Parameters

argName	The name of the argument
value	The argument value
nodeID	The SuperCollider NodeId you wish to use.

4.1.2.101 void send (string *ipAddress,* string *portNumber,* string *path,* *data*)

Allows an Agent to send OSC data to a specified IPAddress, port and path. - Allows an Agent to send OSC data to a specified IPAddress, port and path. A user can append as many pieces of data to send as they wish. This data can be numbers, strings and tables containing key value pairs of data. e.g send("127.0.0.-1","8000","/example",100,"hello",3.14159,{more=info,data=200})

Returns

void

Parameters

ipAddress	IP address of the machine you wish to connect to
portNumber	The port you wish to connect to
path	The path you wish to send via
data	Variadic data of type float, integer, string or a table containing key value pairs of data

4.1.2.102 void setP (string *propertyID,* bool *value*)

Allows an Agent to set a property. If the property does not already exist, it will be created. Allows an Agent to set a property. If the property does not already exist, it will be created. If the current Agent is controlling a SuperCollider node, the system will attempt to set an argument of the same name to the proposed valued.

Returns

void

Parameters

propertyID	The name of the new property
value	The new value for the property

4.1.2.103 void setP (string *propertyID,* int *value*)

Allows an Agent to set a property. If the property does not already exist, it will be created. Allows an Agent to set a property. If the property does not already exist, it will be created. If the current Agent is controlling a SuperCollider node, the system will attempt to set an argument of the same name to the proposed valued.

Returns

void

Parameters

propertyID	The name of the new property
value	The new value for the property

4.1.2.104 void setP (string *propertyID,* float *value*)

Allows an Agent to set a property. If the property does not already exist, it will be created. Allows an Agent to set a property. If the property does not already exist, it will be created. If the current Agent is controlling a SuperCollider node, the system will attempt to set an argument of the same name to the proposed valued.

Returns

void

Parameters

propertyID	The name of the new property
value	The new value for the property

4.1.2.105 void setP (string *propertyID,* string *value*)

Allows an Agent to set a property. If the property does not already exist, it will be created. Allows an Agent to set a property. If the property does not already exist, it will be created. If the current Agent is controlling a SuperCollider node, the system will attempt to set an argument of the same name to the proposed valued.

Returns

void

Parameters

propertyID	The name of the new property
value	The new value for the property

4.1.2.106 void setP (string *propertyID,* Vec2 *value*)

Allows an Agent to set a property. If the property does not already exist, it will be created. Allows an Agent to set a property. If the property does not already exist, it will be created. If the current Agent is controlling a SuperCollider node, the system will attempt to set an argument of the same name to the proposed valued.

Returns

void

Parameters

propertyID	The name of the new property
value	The new value for the property

4.1.2.107 void setP (table *propertyTable*)

Allows an Agent to set a table of properties. Allows an Agent to set a table of new properties. This table must contain key value pairs of data. If property in this table does not exist, a new property will be created for the data. If the current Agent is controlling a SuperCollider node, the system will attempt to set the arguments of the same names to the proposed values.

Returns

void

Parameters

property-Table	A table of key value properties

4.1.2.108 void setPID (string *propertyID,* bool *value,* int *ID*)

Allows an Agent to set a property of a specified Agent. If the property does not already exist, it will be created. Allows an Agent to set a property of a specified Agent. If the property does not already exist, it will be created. If the current Agent is controlling a SuperCollider node, the system will attempt to set an argument of the same name to the proposed valued.

Returns

void

Parameters

propertyID	The name of the new property
value	The new value for the property
ID	ID for the specified Agent

4.1.2.109 void setPID (string *propertyID,* int *value,* int *ID*)

Allows an Agent to set a property of a specified Agent. If the property does not already exist, it will be created. Allows an Agent to set a property of a specified Agent. If the property does not already exist, it will be created. If the current Agent is controlling a SuperCollider node, the system will attempt to set an argument of the same name to the proposed valued.

Returns

void

Parameters

propertyID	The name of the new property
value	The new value for the property
ID	ID for the specified Agent

4.1.2.110 void setPID (string *propertyID,* float *value,* int *ID*)

Allows an Agent to set a property of a specified Agent. If the property does not already exist, it will be created. Allows an Agent to set a property of a specified Agent. If the property does not already exist, it will be created. If the current Agent is controlling a SuperCollider node, the system will attempt to set an argument of the same name to the proposed valued.

Returns

void

Parameters

propertyID	The name of the new property
value	The new value for the property
ID	ID for the specified Agent

4.1.2.111 void setPID (string *propertyID,* string *value,* int *ID*)

Allows an Agent to set a property of a specified Agent. If the property does not already exist, it will be created. Allows an Agent to set a property of a specified Agent. If the property does not already exist, it will be created. If the current Agent is controlling a SuperCollider node, the system will attempt to set an argument of the same name to the proposed valued.

Returns

void

Parameters

propertyID	The name of the new property
value	The new value for the property
ID	ID for the specified Agent

4.1.2.112 void setPID (string *propertyID,* Vec2 *value,* int *ID*)

Allows an Agent to set a property of a specified Agent. If the property does not already exist, it will be created. Allows an Agent to set a property of a specified Agent. If the property does not already exist, it will be created. If the current Agent is controlling a SuperCollider node, the system will attempt to set an argument of the same name to the proposed valued.

Returns

void

Parameters

propertyID	The name of the new property
value	The new value for the property
ID	ID for the specified Agent

4.1.2.113 void setPID (table *propertyTable,* int *ID*)

Allows an Agent to set a table of properties in a specified Agent. Allows an Agent to set a table of new properties in a specified Agent. This table must contain key value pairs of data. If property in this table does not exist, a new property will be created for the data. If the current Agent is controlling a SuperCollider node, the system will attempt to set arguments of the same name to the proposed values.

Returns

void

Parameters

property-Table	A table of key value properties
ID	ID for the specified Agent

4.1.2.114 void setPosition (int *xPos*, int *yPos*)

Allows an Agent to set its position. (Deprecated) Allows an Agent to set its position. - This is deprecated, uses should use setP, specifying the position property. i.e setP("pos" Vec2.new(100,100))

Returns

void

Parameters

xPos	X cartesian coordinate
yPos	Y cartesian coordinate

4.1.2.115 void setPositionID (int *xPos*, int *yPos*, int *ID*)

Allows an Agent to set the position of another Agent. Allows an Agent to set the position of another Agent. This is deprecated, uses should use setP, specifying the position property. i.e setPID("pos",Vec2.new(200,200),ID).

Returns

void

Parameters

xPos	X cartesian coordinate
yPos	Y cartesian coordinate
ID	The ID of the Agent

4.1.2.116 int spawn (int *parentID*, int *nodeType*, string *agentFunctionName*, int *xPos*, int *yPos*)

Allows the spawning of an Agent (or another Node type) in world coordinates. he basic spawn function that allows the creation of new Agents. This is done so by specifying the parent ID, the name of the script function for the Agent to use and via providing world coordinates (as cartesian components or as a Vec2) for the Agent to spawn at.

Returns

The ID of the newly created Agent

Parameters

parentID	The ID of the new Agents parent
nodeType	Should be set to 1 for Agents
agent-Function-Name	The name of the ai_function to use
xPos	Cartesian X coordinate
yPos	Cartesian Y coordinate

4.1.2.117 int spawn (int *parentID*, int *nodeType*, string *agentFunctionName*, Vec2 *position*)

Allows the spawning of an Agent (or another Node type) in world coordinates. he basic spawn function that allows the creation of new Agents. This is done so by specifying the parent ID, the name of the script function for the Agent to use and via providing world coordinates (as cartesian components or as a Vec2) for the Agent to spawn at.

Returns

The ID of the newly created Agent

Parameters

parentID	The ID of the new Agents parent
nodeType	Should be set to 1 for Agents
agent-Function-Name	The name of the ai_function to use
position	A vector representing the position

4.1.2.118 int spawnP (table *propertyTable,* int *parentID,* int *nodeType,* string *agentFunctionName,* int *xPos,* int *yPos*)

Allows the spawning of an Agent (or another Node type) with a table of properties for it to use. A spawn function that allows the creation of new Agents with a suite of properties stored within a table. This is done so by providing a table containing key value pairs {key = value}, the parent ID, the name of the script function for the Agent to use and via providing world coordinates (as cartesian components or as a Vec2) for the Agent to spawn at.

Returns

The ID of the newly created Agent

Parameters

property-Table	A table of property key value pairs
parentID	The ID of the new Agents parent
nodeType	Should be set to 1 for Agents
agent-Function-Name	The name of the ai_function to use
xPos	Cartesian X coordinate
yPos	Cartesian Y coordinate

4.1.2.119 int spawnP (table *propertyTable,* int *parentID,* int *nodeType,* string *agentFunctionName,* Vec2 *position*)

Allows the spawning of an Agent (or another Node type) with a table of properties for it to use. A spawn function that allows the creation of new Agents with a suite of properties stored within a table. This is done so by providing a table containing key value pairs {key = value}, the parent ID, the name of the script function for the Agent to use and via providing world coordinates (as cartesian components or as a Vec2) for the Agent to spawn at.

Returns

The ID of the newly created Agent

Parameters

property-Table	A table of property key value pairs
parentID	The ID of the new Agents parent
nodeType	Should be set to 1 for Agents
agent-Function-Name	The name of the ai_function to use
position	Global position

4.1.2.120 int spawnRelative (int *parentID,* int *nodeType,* string *agentFunctionName,* int *xPos,* int *yPos*)

Allows the spawning of an Agent (or another Node type) using coordinates relative to the currently active Agent. A spawn function that allows the creation of new Agents. This is done so by specifying the parent ID, the name of the script function for the Agent to use and via coordinates that are relative to the currently active Agent's position (as cartesian components or as a Vec2).

Returns

The ID of the newly created Agent

Parameters

parentID	The ID of the new Agents parent
nodeType	Should be set to 1 for Agents
agent-Function-Name	The name of the ai_function to use
xPos	Relative X position
yPos	Relative Y position

4.1.2.121 int spawnRelative (int *parentID,* int *nodeType,* string *agentFunctionName,* Vec2 *position*)

Allows the spawning of an Agent (or another Node type) using coordinates relative to the currently active Agent. A spawn function that allows the creation of new Agents. This is done so by specifying the parent ID, the name of the script function for the Agent to use and via coordinates that are relative to the currently active Agent's position (as cartesian components or as a Vec2).

Returns

The ID of the newly created Agent

Parameters

parentID	The ID of the new Agents parent
nodeType	Should be set to 1 for Agents
agent-Function-Name	The name of the ai_function to use
position	Relative position

4.1.2.122 int spawnRelativeP (table *propertyTable,* int *parentID,* int *nodeType,* string *agentFunctionName,* int *xPos,* int *yPos*)

Allows the spawning of an Agent (or another Node type) with a table of properties for it to use with coordinates relative to the currently active Agent A spawn function that allows the creation of new Agents with a suite of properties stored within a table. This is done so by providing a table containing key value pairs {key = value}, the parent ID, the name of the script function for the Agent to use and via providing relative coordinates (as cartesian components or as a Vec2) for the Agent to spawn at.

Returns

The ID of the newly created Agent

Parameters

property-Table	A table of key value properties
parentID	The ID of the new Agents parent
nodeType	Should be set to 1 for Agents
agent-Function-Name	The name of the ai_function to use
xPos	Relative X coordinate
yPos	Relative Y coordinate

4.1.2.123 int spawnRelativeP (table *propertyTable,* int *parentID,* int *nodeType,* string *agentFunctionName,* Vec2 *position*)

Allows the spawning of an Agent (or another Node type) with a table of properties for it to use with coordinates relative to the currently active Agent A spawn function that allows the creation of new Agents with a suite of properties stored within a table. This is done so by providing a table containing key value pairs {key = value}, the parent ID, the name of the script function for the Agent to use and via providing relative coordinates (as cartesian components or as a Vec2) for the Agent to spawn at.

Returns

The ID of the newly created Agent

Parameters

property-Table	A table of key value properties
parentID	The ID of the new Agents parent
nodeType	Should be set to 1 for Agents
agent-Function-Name	The name of the ai_function to use
position	Relative position