Last edition 28.11.2017

Diagram elements

Introduction

Welcome to the Panthea VS diagram elements guide. This documentation section contains a description of all the elements used in the diagrams and included in the basic set supplied with the framework. Documentation of the elements delivered by individual packages is contained in them.

Note: the description of elements with the same logic of behavior, but with different input data will be merged into one subsection, in order to reduce the amount of information.

Branching

Section with descriptions of elements responsible for conditional and unconditional branching.

If

This section describes elements with branching by the condition **if**, by comparing two values.

IfBool

The element of checking the input boolean value is true or false.

Input points:

ValueToBeChecked **[bool]** – Input point for transmitting value of **bool** typeт, which should be checked.

Output points:

True [] – output point without data transmission, called if input value is **true.**

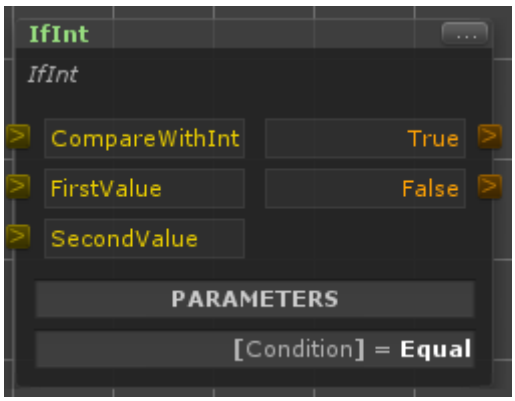
False [] - output point without data transmission, called if input value is **false.**

IfInt

IfFloat

Elements for comparing two values for types int and float. These elements work in two modes:

1. Comparison of the value with the internal value of the element set in the parameters.
2. Comparison of two external values among themselves. In this mode, the element will perform comparisons when it receives both values. The order of setting (transfer of data to the input point) does not matter.



Input points:

CompareWithInternal **[int, float]** – Input point for transmitting value, which must be compared with the internal value of the element, set through the parameters.

FirstValue **[Int, float]** – input point for transmitting the first value for the subsequent condition check.

SecondValue **[Int, float]** – input point for transmitting the second value for subsequent condition checking.

Output points:

True [] – output point without data transmission, called if compare result is true.

False [] - output point without data transmission, called if compare result is false.

Parameters:

HideFlag – configuration flag for input points.

Condition – the type of comparison between two values.

ValueForCompare – an internal value for comparison, used only if the value is passed through the input point CompareWithInternal.

IfStringEqual

IfVector2Equal

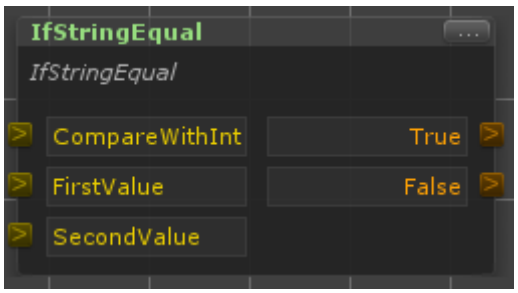
IfVector3Equal

IfVector4Equal

IfVSObjectEqual

Elements check for equality for types of string, Vector2, Vector3, Vector4. These elements work in two modes:

1. Comparison of the value with the internal value of the element set in the parameters.
2. Comparison of two external values among themselves. In this mode, the element will perform comparisons when it receives both values. The order of setting (transfer of data to the input point) does not matter.



Input points:

CompareWithInternal **[string, Vector2/3/4, VSObject]** – Input point for transmitting value, which must be compared with the internal value of the element, set through the parameters.

FirstValue **[string, Vector2/3/, VSObject]** – input point for transmitting the first value for the subsequent equality check.

SecondValue **[string, Vector2/3/4, VSObject]** – input point for transmitting the second value for subsequent equality check.

Output points:

True [] – output point without data transmission, called if input values are equal.

False [] - output point without data transmission, вызывается, called if input values are not equal.

Parameters:

HideFlag – configuration flag for input points.

ValueForCompare – an internal value for comparison, used only if the value is passed through the input point CompareWithInternal.

Switch

This section describes the elements for branching by value.

SwitchInt

SwitchFloat

SwitchColor

SwitchString

SwitchVector2

SwitchVector3

SwitchVector4

SwitchVSObject

Elements of branching by value for types int, float, Color, string, Vector2, Vector3, Vector4 and VSObject. These elements take on the input the value of the corresponding type and compare it with the set of values set in the parameters of the element, after which the output point corresponding to the value whose comparison returned true is called.

Note: The output points are automatically configured according to the set value and their names are the string representations of these values.

Input points:

ValueToBeChecked **[int, float, Color, string, Vector2/3/4, VSObject]** – Input point for transmitting value of bool type, which should be checked.

Output points:

Automatically configured by the set of values. All output points do not transmit data.

Parameters:

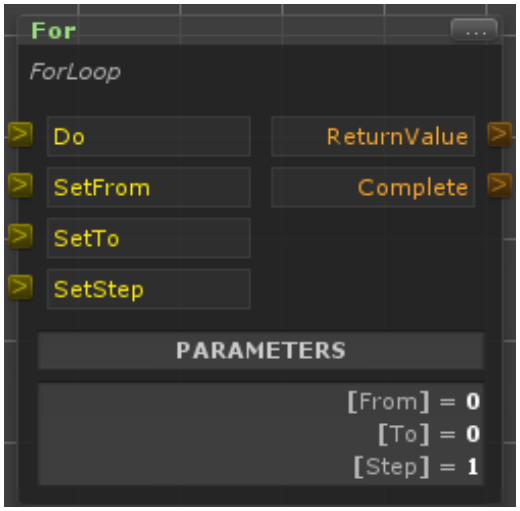
SwitchValues – set of values for branching.

Loops

This section describes the elements that implement the logic of loops.

For

An element that implements the logic of the classical **for** loop. It works with **int** data type. This element increments the value with the specified step between the two values.



Input points:

Do [] – input point without data transmission, executes a loop with the set parameters. The parameters can be set externally, or through the parameters of the element. If there was no input from outside, the parameters are always taken from the parameters.

SetFrom **[int]** – setting the initial value for the loop.

SetTo **[int]** – setting the end value for the loop.

SetStep **[int]** – setting the step value for the loop.

Output points:

ReturnValue **[int]** - output point transmitting the current value in the loop.

Complete**[] –** output point that is called after the loop is completed.

Parameters:

OnlyInternalParam – flag hiding the input points of the parameters settings.

From – initial value for the loop, updated in the debug mode.

To – end value for the loop, updated in the debug mode.

Step – incrementation step for the loop, updated in the debug mode.

ForEachColor

ForEachFloat

ForEachInt

ForEachString

ForEachVector2

ForEachVector3

ForEachVector4

ForEachVSObject

Elements of a loop that iterates through the array items with data types int, float, Color, string, Vector2, Vector3, Vector4, and VSObject.

Input points:

Do [] – input point without data transmission, executes the loop.

Output points:

ReturnValue **[int, float, Color, string, Vector2/3/4, VSObject]** - output point transmitting the current value in the loop.

Complete**[] –** output point that is called after the loop is completed.

References to diagram variables:

Variable – reference to an array variable with the data type of the corresponding element

ForEachList

Elements of a loop that performs an iteration through the list items.

Input points:

Do [] – input point without data transmission, executes the loop.

Output points:

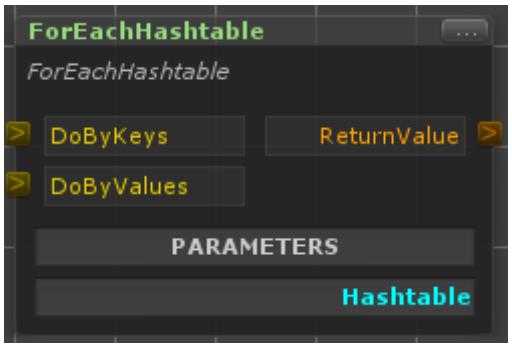
ReturnValue **[object]** - output point transmitting the current value in the loop.

References to diagram variables:

Variable – reference to the variable of the list.

ForEachHashtable

Elements of a loop that iterates through the keys or values in a hash table.



Input points:

DoByKeys [] – input point without data transmission, executes the loop of iterating through keys of hash table.

DoByValues [] – input point without data transmission, executes the loop of iterating through values of hash table.

Output points:

ReturnValue **[object]** - output point transmitting the current value in the loop.

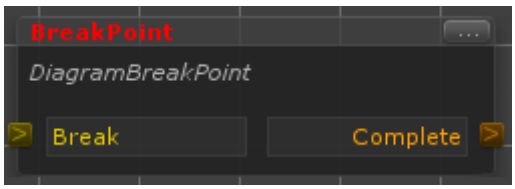
References to diagram variables:

Variable – reference to the variable of the hash table.

Debug

BreakPoint

An element that stops the diagram. Connect it to any output in the logic chain to pause the execution. If you want it to be paused prior to the execution of the next element logic, use number 1 as the breakpoint link order.

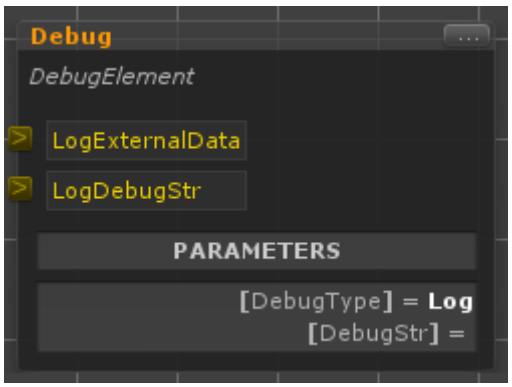


Input points:

Break [] – input point which is stopping execution of the diagram.

Debug

Element outputs debugging data to the Unity console. Also, outputs data to the log file in the application build.



Input points:

LogExternalData **[object]** – input point for outputting external data to the console, if the debugging type uses formatting, then the data will be substituted into the DebugStr string if it has the form of a formatted string.

LogDebugStr **[]** – input point of outputting the line set in the DebugStr parameter to the console.

Parameters:

DebugType – flag specifying the way to output debugging data. It uses the same type as Unity.

DebugStr – the string for output to the console, can have a formatted view.

Diagram

A section describing the elements for linking diagrams and transferring data between them and between the diagram and the outside code.

Messages

SendMessage

An element sending a message with or without data.

Input points:

SendDataMessage **[object]** – input point for sending a message with data.

SendEmptyMessage [] – input point for sending a message without data.

Output points:

Complete [] - output point, called when the message is sent.

Parameters:

Message – identifier of the sent message.

ReceiveMessage

An element that processes messages with a given identifier.

Output points:

DataMessage **[object]** - output point that is called when the message with data is processed.

EmptyMessage [] – output point that is called when the message without data is processed.

Parameters:

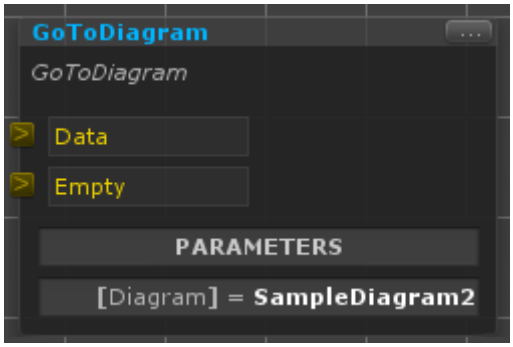
Message – identifier of the processed message.

Utils

Section with elements of interaction between diagrams.

GoToDiagram

An element of transition the logic flow to another diagram with or without data transfer.



Input points:

Data **[object]** - input point for the transition to the diagram and the transfer of data to it.

Empty [] – input point for the transition to the diagram without data transmission.

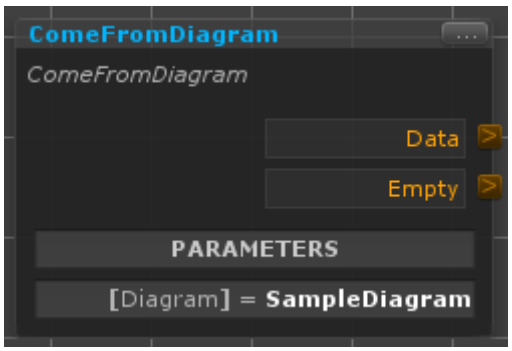
Parameters:

Diagram – diagram to which messages are sent.

ComeFromDiagram

An element for processing he data came from another diagram.

Note: each diagram can receive data from several other diagrams and process them in different ways.



Output points:

Data **[object]** - output point that transmits data came from another diagram.

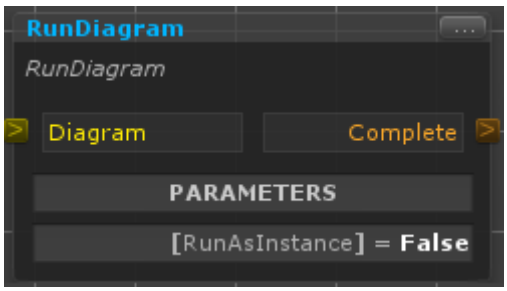
Empty [] – output point for processing transition from another diagram without data.

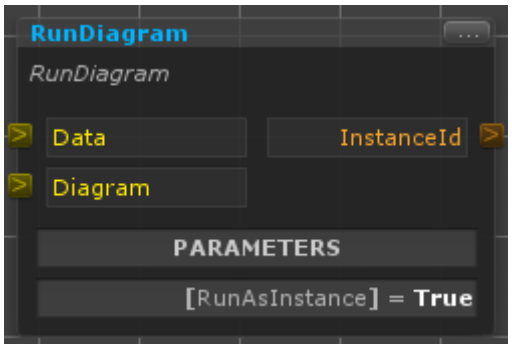
Parameters:

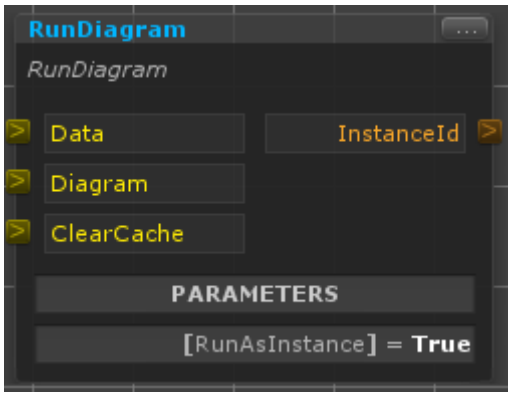
Diagram – the diagram from which the data comes.

RunDiagram

An element for loading and starting a diagram from external resources (TextAsset).







Input points:

Diagram**[TextAsset]** – input point for loading and starting a diagram from the TextAsset.

Data**[object]** – input point for transferring data to diagram instance. This point should be called before Diagram point**.**

ClearCache**[]** – input point for clearing the element cache.

Output points:

Complete [] - output point, called when the diagram is finished.

InstanceId **[string]** - output point that transmits identifier of diagram instance.

Parameters:

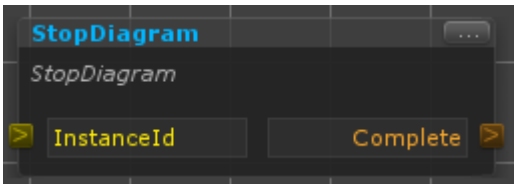
RunAsInstance – a flag for run a diagram as an instance.

CacheDiagramData – a flag for caching diagram data (to prevent re-loading the diagram from the asset if the diagram was started earlier).

Note: this element purpose is to execute diagrams, which are not added to the scene, and not initialized at the starting moment. They can live in a variable, be loaded from resources or even from the remote server. With **RunAsInstance** mode enabled, such diagram can be instantiated as multiple copies, for example, each controlling own object.

StopDiagram

An element for destroying a diagram instance by identifier.



Input points:

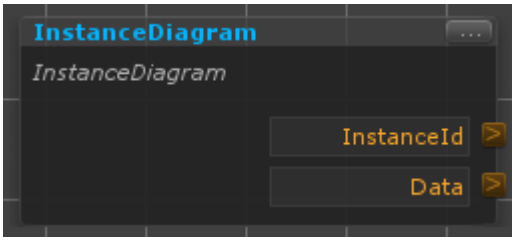
InstanceId **[string]** – input point for transferring diagram instance identifier.

Output points:

Complete [] - output point, called when the diagram is destroyed.

InstanceDiagram

An element for initializing diagram instance.



Output points:

InstanceId **[string]** - output point that transmits identifier of diagram instance.

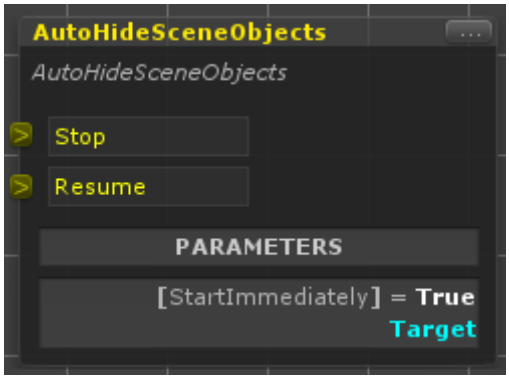
Data**[object]** – output point that transmits data for diagram instance.

GamePlay

Logics

AutoHideSceneObjects

An element for automatic hiding of objects (through the replacement of material), if they are between the target and the main camera.



Input points:

Stop **[]** – input point for stopping the operation of the element (events are stopped being processed).

Resume **[]** – input point for the resuming the element operations.

Parameters:

StartImmediately – flag for starting the element along with the launch of the scene.

ReplacementLayer – the mask of layers in which the objects are hiding through the replacement of the material.

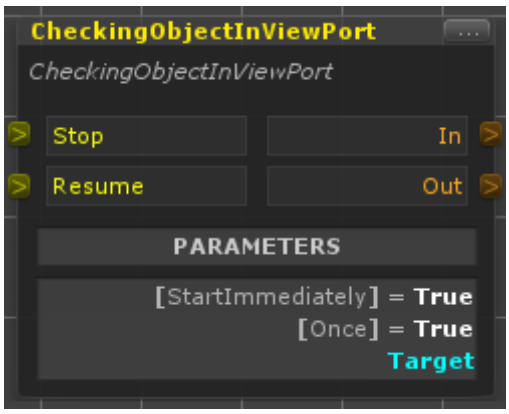
ReplacementMaterial – material that will be used for replacement (should be transparent).

References to diagram variables:

UnityObject – reference to the target object variable.

CheckingObjectInViewPort

An element for checking the position of the object in the scope of the camera.



Input points:

Stop **[]** – input point for stopping the operation of the element (events are stopped being processed).

Resume **[]** – input point for the resuming the element operations.

Output points:

In [] – output point called when the object falls within the scope of the camera (or is inside the scope of the camera at the time the element starts to work)

Out [] – output point called when the object leaves the camera's view (or is outside of it at the time the element starts to work)

Parameters:

StartImmediately – flag for starting the element along with the launch of the scene.

Once – flag indicating that the same event will be sent only once.

References to diagram variables:

UnityObject – reference to a variable with a scene object.

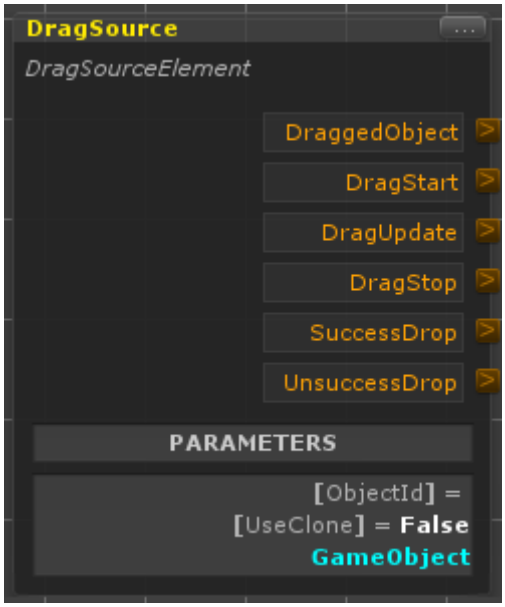
Mechanics

Drag and Drop

A section describing the elements that implement the mechanics of capturing, dragging and dropping an object.

DragSource

Element for implementing the logic of the object being dragged.



Output points:

DraggedObject **[VSObject(GameObject)]** – an instance of the current dragged object. Can be used to manipulate it.

DragStart **[]** – output point that is called at the time the object is taken.

DragUpdate **[]** – output point that is called when the object is dragged.

DragStop **[]** – output point that is called when the object is dropped.

SuccessDrop **[]** – output point that is called if the object is dropped on the needed target.

UnsuccessDrop **[]** – output point that is called if the object is dropped outside the needed target bounds.

Parameters:

ObjectId – string identifier of the dragged object. Used when an object can only be dropped to a specific target.

UseClone – flag, used for the cloning mode of the source object when dragging. In this case, the object remains in place, and copies of it are taken and dragged.

IsUnityGUI – flag indicating that the object to be dragged is a Unity GUI element.

DragAxis – field for setting the axes along which dragging is carried out, if the value on the axis is different from 0, then the object can move along it.

UseTouchPoint – flag indicating that the object will be dragged relative to the point of touch / click on the object. If false, the drag is performed relative to the position of the object (its pivot).

PlaceSortingOrder – sort position, which will be set after the object is dropped, is used only for sprites.

DragAboveAll – flag indicating that the object should be dragged above all other objects. Used for sprites and uGUI elements. After dropping the object, the object's sorting position will be restored to the original one.

UseRestorePosition – flag indicating that in the event of an object being dropped outside the target, it must return to its original position.

UseSmoothRestore – flag indicating that the restoration of the original position should be done smooth.

RestoreTime – position recovery time.

RestoreCurve – the animation curve for the change in the position value from the current to the initial, depending on the normalized time value.

References to diagram variables:

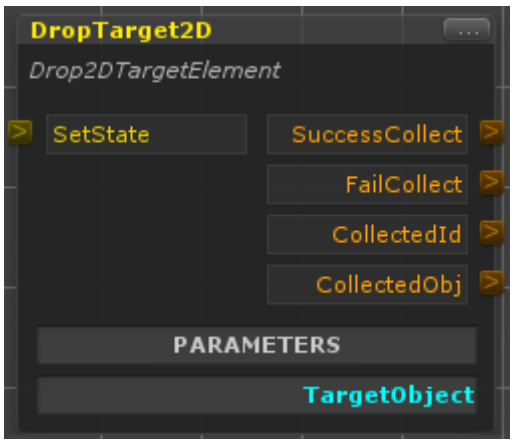
UnityObject – reference to the variable of the object being dragged.

DropTarget2D

DropTarget3D

DropTargetUnityGUI

Elements that implement the logic of the target object for the dragged object for 2D objects (Sprite), 3D objects and uGUI objects.



Input points:

SetState**[bool]** – input point for setting the state of the object's logic.

Output points:

SuccessCollect **[]** - output point, called in case the object was successfully dropped into the target.

FailCollect **[]** - output point, called in case the object was unsuccessfully dropped into the target.

CollectedId **[]** - output point that passes the string identifier of an object, successfully dropped into the target.

CollectedObj **[VSObject(GameObject)]** - output point that transmits an instance of an object, successfully dropped into the target.

Parameters:

CollectObjectId – a list of string identifiers of dragged objects that can be dropped into the target object.

References to diagram variables:

UnityObject – a reference to the variable of the scene object into which the dragged object is to be dropped.

FPS

FPSWalker

An element for the controller of the player's first-person shooter movement.

Input points:

SetWalkSpeed**[float]** – input point for the dynamic change of the character walking speed.

SetRunSpeed**[float]** – input point for the dynamic change of the character running speed.

MoveHorizontal **[float]** – input point for controlling the movement along the X axis (-1 to the left, 1 to the right).

MoveVertical **[float]** – input point for controlling the movement along the Z axis (-1 backward, 1 forward).

ToogleRun **[bool]** – input point for switching the running mode

Jump **[]** – input point for the jump.

Output points:

PlayerTakeFallDamage**[float]** - output point for sending the event of falling from a height that leads to damage (the height of falling is transmitted.

Parameters:

PlayerCanRun – flag indicating that the player can run.

PlayerCanJump – flag indicating that the player can jump.

PlayerCanFall – flag indicating that the player can fall and get damaged

PlayerCanSlide – flag indicating that the player can slide on inclined surfaces

SpeedCanChangedExternal – flag indicating that speed can be changed externally.

PlayerCanAirControl – flag indicating that the player can control the movement during the jump

DefaultWalkSpeed – default walking speed.

DefaultRunSpeed – default running speed.

DefaultSlideSpeed – default speed of sliding on inclined surfaces.

DiagonalSpeedModifier – speed modifier when moving diagonally.

SlideIfSlopeLimit – flag indicating that the player will slide down slopes that are greater than the Slope Limit as defined by the Character Controller. Attempting to jump up such slopes will also fail. The player has no control until the Slope Limit

AntiShakeModifier – a modifier for adjusting the stepwise slip along the slope to avoid jitter.

JumpSpeed – the speed of the y-axis movement during the jump.

Gravity – the value of gravity (determines the character falling speed)

FallingHeightThreshold – the value of the height above which the player is damaged when falling.

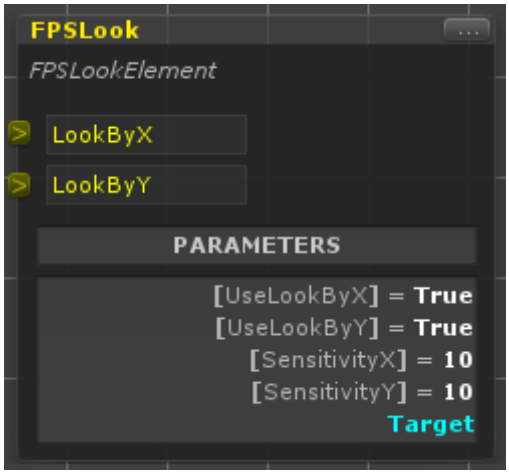
MaximumSimultaneousJumps – the value that determines the number of jumps that can be made simultaneously, at a value of 2 the player can perform a double jump.

References to diagram variables:

UnityObject – a reference to a variable containing the scene object that is associated with the player.

FPSLook

Элемент для контроллера управления взглядом шутера от первого лица An element for a look controller of a first-person shooter or walker.



Input points:

LookByX **[float]** – input point for controlling the view in the plane of the X axis (-1 to the left, 1 to the right).

LookByY **[float]** – input point for controlling the view in the plane of the Y axis (-1 down, 1 up).

Parameters:

UseLookByX – flag indicating that the player can control the view in the plane of the X axis.

UseLookByY – flag indicating that the player can control the view in the plane of the Y axis.

SensitivityX – sensitivity n the X-axis plane.

SensitivityY – sensitivity n the Y-axis plane.

MinXAngle – minimum rotation angle in the X axis plane.

MaxXAngle – maximum rotation angle in the X axis plane.

MinYAngle – minimum rotation angle in the Y axis plane.

MaxYAngle – maximum rotation angle in the Y axis plane.

References to diagram variables:

UnityObject – ссылка на переменную содержащую объект сцены, который ассоциируется с игроком.

SaveLoad

An element for saving and loading custom user data using UserPrefs. This element automatically configures the input and output points, depending on the set data set for saving.

Input points:

Save…**[int, float, string]** – input point for transmitting data, which should be saved..

Load**[]** – input point for loading previously saved data

Clear**[]** – input point for delete all saved data

Output points:

Return… **[int, float, string]** - output point transmitting the stored data corresponding to the identifier.

Parameters:

DataDefinition – a list with the definition of the data to save. Each list item is a string identifier (also used for naming input and output points) and the type of data that corresponds to it.

GUI

A section describing the elements that implement the logic of the uGUI system elements.

Button

An element for working with uGUI Button.

Input points:

Interactable **[bool]** – input point for setting the interactable parameter.

Update **[]** – input point for element re-initialization.

Output points:

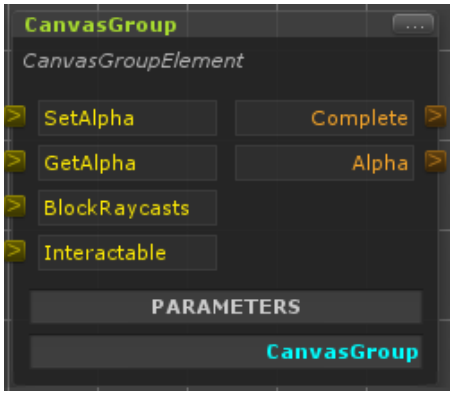
OnClick **[]** - output point that is called when the button is pressed.

References to diagram variables:

UnityObject – reference to a variable that contains the button object (the Button component).

CanvasGroup

An element for working with uGUI CanvasGroup.



Input points:

SetAlpha **[float]** - input point for setting transparency.

GetAlpha **[]** - input point to get the current transparency value.

BlockRaycasts **[bool]** - input point for setting the status of the event blocking.

Interactable **[bool]** – input point for setting the Interactable parameter

Output points:

Complete **[]** - output point that is called when the setting of transparency and the event blocking state are completed.

Alpha **[float]** – output point that transmits the current transparency value.

References to diagram variables:

UnityObject – a reference to a variable that contains the CanvasGroup object (the CanvasGroup component).

DropDown

An element to work with the drop-down list (uGUI DropDown).

Input points:

SetStringOptions **[]** - input point for setting a drop-down list from an external rowset.

SetSpriteOptions **[]** - input point for setting a drop-down list from an external set of sprites.

SetState **[int]** - input point for setting the index of the currently selected item from the drop-down list.

Interactable **[bool]** – input point for setting the interactable parameter.

Update **[]** – input point for element re-initialization.

Output points:

OnValueChanged **[int]** - output point that transfers the index of the selected item from the drop-down list.

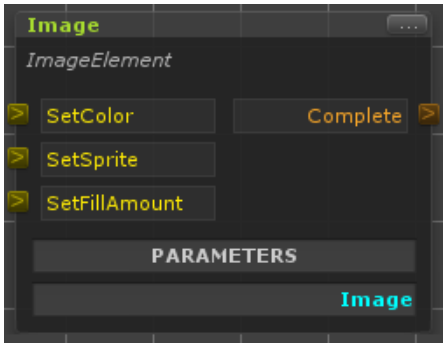
References to diagram variables:

ExternalOptions – a reference to a list variable (List) containing either strings or sprites.

UnityObject – a reference to a variable containing the drop-down list object (the DropDown component).

Image

Elements for working with uGUI Image.



Input points:

SetColor **[Color]** - input point for setting the color of the image.

SetSprite **[VSObject(Sprite)]** - input point for setting the sprite of the image.

SetFillAmout **[float]** - input point for setting fill amount of the image (for filled image type).

Output points:

Complete **[]** - output point that is called when the color or sprite is set.

Parameters:

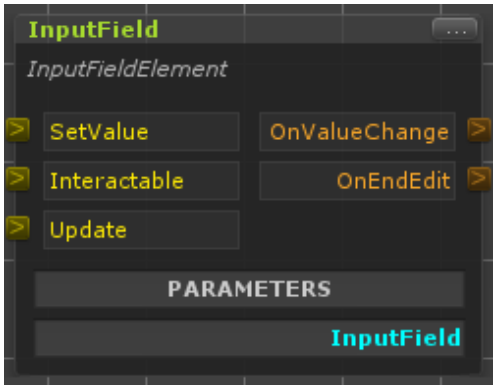
IsLocalized – a flag indicating that the image is a localizable resource. If the flag is true, then the options for setting the image localization are opened.

References to diagram variables:

UnityObject – a reference to a variable containing an Image object (an Image component).

InputField

Elements for working with the string input field (uGUI InputField).



Input points:

SetValue **[string]** - input point for setting a string in the input field.

Interactable **[bool]** – input point for setting the interactable parameter.

Update **[]** – input point for element re-initialization.

Output points:

OnValueChanged **[string]** - output point that transmits the current value in the input field when it changes

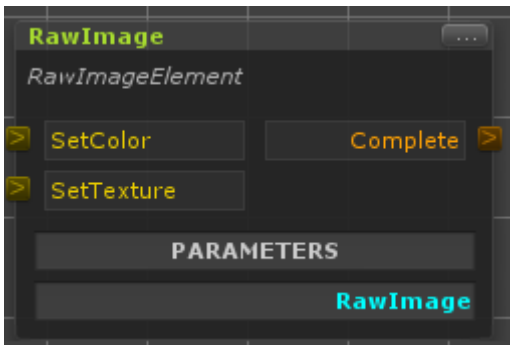
OnEndEdit **[string]** – output point that passes the value in the input field after the editing is completed.

References to diagram variables:

UnityObject – a reference to a variable that contains an input field object (an InputField component).

RawImage

Elements for working with uGUI RawImage.



Input points:

SetColor **[Color]** - input point for setting the color of the image.

SetTexture **[VSObject(Texture)]** - input point for setting the texture of the image.

Output points:

Complete **[]** - output point that is called when the color or texture is set.

Parameters:

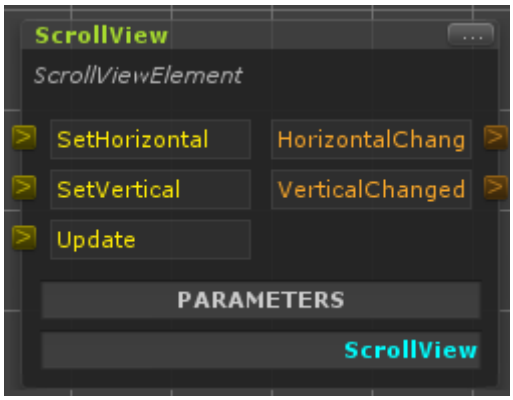
IsLocalized – a flag indicating that the image is a localizable resource. If the flag is true, then the options for setting the image localization are opened.

References to diagram variables:

UnityObject – a reference to the variable containing the RawImage object (the RawImage component).

ScrollView

An element for working with the scrolling area (uGUI ScrollView).



Input points:

SetHorizontal **[float]** - input point for setting the horizontal scroll position.

SetVertical **[float]** - input point for setting the vertical scroll position.

Update **[]** – input point for element re-initialization.

Output points:

HorizontalChanged **[float]** - output point that transmits the current value of the horizontal scroll position when it is changed.

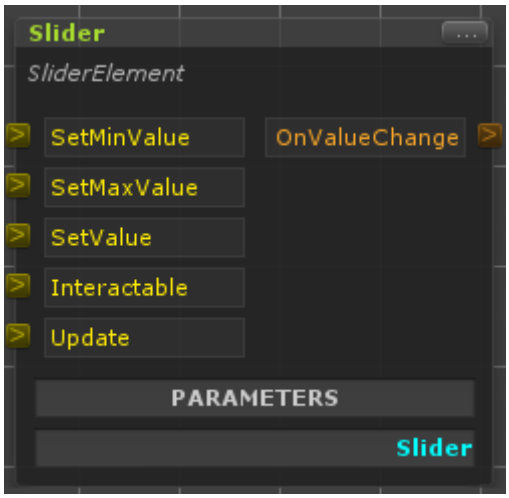
VerticalChanged **[float]** - output point that transmits the current value of the vertical scroll position when it is changed.

References to diagram variables:

UnityObject – a reference to a variable that contains the scroll view object (the ScrollView component).

Slider

An element for working with uGUI Slider.



Input points:

SetMinValue **[float]** - input point for setting the minimum value of the slider.

SetMaxValue **[float]** - input point for setting the maximum value of the slider.

SetValue **[float]** - input point for setting the current value of the slider.

Interactable **[bool]** – input point for setting the interactable parameter.

Update **[]** – input point for element re-initialization.

Output points:

OnValueChanged **[float]** - output point transmitting the current position of the slider when it changes.

References to diagram variables:

UnityObject – a reference to a variable that contains the slider object (Slider component).

Text

An element for working with text (uGUI Text).

Input points:

SetTextExternal **[string]** - input point for setting the text from external data.

SetFontExternal **[VSObject(Font)]** - input point for setting the font from external data.

SetTextInternal **[]** - input point for setting the text from element parameters.

SetFontInternal **[]** - input point for setting the font from element parameters.

Output points:

Complete **[]** - output point that is called when the text or font is set.

Parameters:

StringValue – a string to set to text

FontValue – a reference to the font for installation in the text.

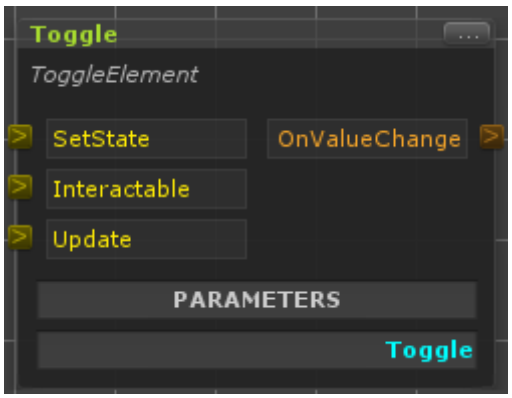
IsLocalized – a flag indicating that the text is a localizable resource. If the flag is true, then the options for setting the text localization are opened.

References to diagram variables:

UnityObject – a reference to a variable that contains the text object (Text component).

Toggle

An element for working with uGUI Toggle



Input points:

SetState **[bool]** - input point for setting the value of the switch.

Interactable **[bool]** – input point for setting the interactable parameter.

Update **[]** – input point for element re-initialization.

Output points:

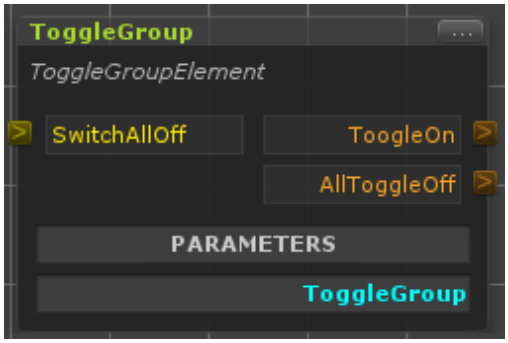
OnValueChanged **[bool]** - output point that transmits the current value of the switch when it is changed.

References to diagram variables:

UnityObject – a reference to a variable that contains Toggle object (Toggle component).

ToggleGroup

An element for working with uGUI ToggleGroup.



Input points:

SwitchAllOff **[]** - input point for switching off all toggles in a group.

Output points:

ToggleOn **[VSObject(Toggle)]** - output point that transmits the selected toggle.

AllToggleOff **[]** - output point that is called when all toggles in a group are off.

References to diagram variables:

UnityObject – a reference to a variable that contains Toggle object (ToggleGroup component).

Input

A section describing the elements that handle the user input.

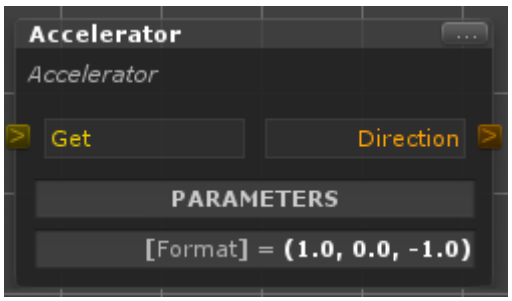
Classic

Section describing the elements that implement the logic of the classical input system of Unity.

Accelerometer

Gyroscope

Elements for obtaining accelerometer and gyro data with value conversion (only for mobile devices that support this function).



Input points:

Get **[]** - input point for obtaining accelerometer / gyro values.

Output points:

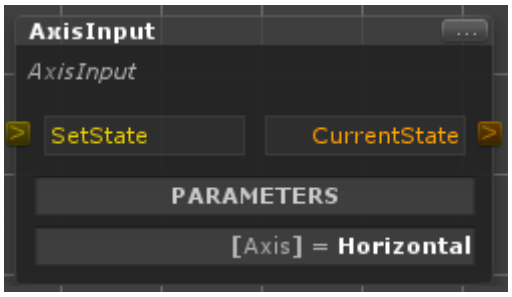
Direction**[Vector3]** - output point that transfers the value of the acceleration vector multiplied by the value transformation vector.

Parameters:

Format – vector of data transformation from the accelerometer / gyro.

AxisInput

An element for processing input commands based on Input.GetAxis.



Input points:

SetState **[bool]** - input point for setting the state of the element (to process or not the input commands).

Output points:

CurrentState**[float]** - output point that transmits the current input value along the axis.

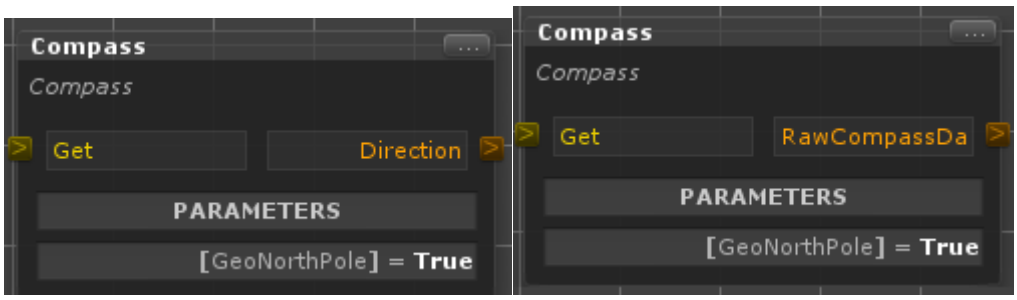
Parameters:

HandleAxis – input axis for processing (this is an automatically generated list defined through the Input Manager of the input axes)

Raw – a flag indicating the receipt of input data in raw format.

Compass

An element for obtaining data from the compass (only for mobile devices that support this function).



Input points:

Get **[]** - input point to get the current compass position value.

Output points:

Direction**[float]** - output point transmitting the current position of the compass in the form of an angle with respect to the direction to the north pole. The point is enabled when the RawData flag is off.

RawCompassData**[Vector3]** – output point transmitting the initial position data of the compass in the form of a direction vector. The point turns on when the RawData flag is on.

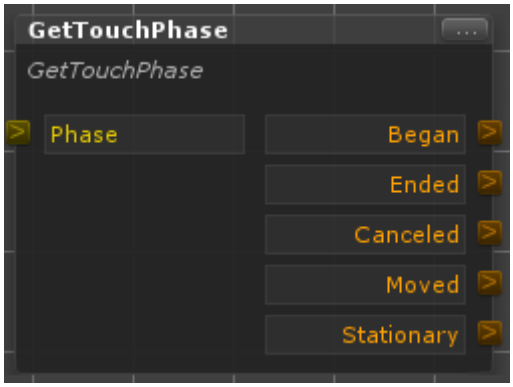
Parameters:

GeoNorthPole – a flag indicating which pole should be used to receive data on the position of the compass. When the flag is turned on, the data is obtained with respect to the geographical north pole, with the switched off - relative to the magnetic pole. This flag is only counted when the RawData flag is off.

RawData - flag indicating the receipt of data in raw format.

GetTouchPhase

An element used to interpret the events of interaction with the screen of a mobile device.



Input points:

Phase **[TouchPhase]** - input point for interpreting the events of interaction with the screen of the mobile device.

Output points:

Began **[]** - output point for the begin touch event (a finger was placed on a screen)

Ended **[]** - output point for the end touch event (a finger was raised from a screen)

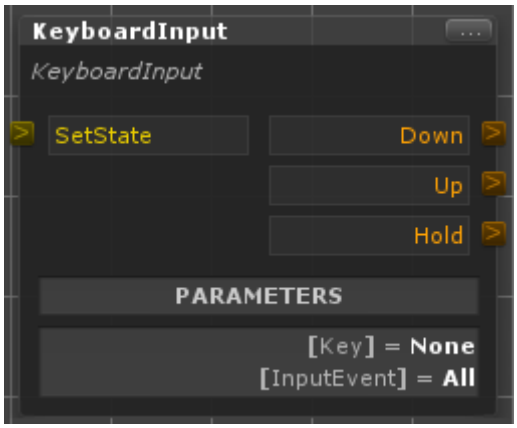
Canceled **[]** - output point for the event of cancel touching the screen.

Moved **[]** - output point for the event of finger movement on the screen.

Stationary **[]** - output point for the event of holding a finger at the touch point.

KeyboardInput

An element for processing input commands from the keyboard.



Input points:

SetState **[bool]** - input point for setting the state of the element (to process or not the input commands).

Output points:

Down **[]** - output point for the key press event.

Up **[]** - output point for the key release event.

Hold **[]** - output point for the key hold event.

Note: the set of output points is automatically generated depending on the InputEvent parameter.

Parameters:

Key – key identifier for processing.

InputEvent – the type of event to handle.

Location

An element for obtaining geo-location data (it is used only for mobile devices).

Input points:

StartLocation **[]** - input point for starting the geo-location service.

StopLocation **[]** - input point for stopping the geo-location service.

Output points:

ServiceDisable**[]** - output point that is called if the geo-location service is disabled on the mobile device.

Coordinate**[Vector3]** - output point that transmits the value of the current geo position in the form (latitude, height, longitude). The point is enabled when the RawData flag is off.

RawLocationData **[LocationInfo]** – output point that transmits the value of the current geo position in raw format. The point turns on when the RawData flag is on.

Parameters:

AccuracyInMeter – a parameter specifying the desired position accuracy in meters.

DistanceInMeter – a parameter specifying the step of updating the position in meters.

RawData – a flag indicating the receipt of data in raw format.

MouseInput

An element for processing input from the mouse.

Input points:

SetState **[bool]** - input point for setting the state of the element (to process or not the input commands).

Output points:

Down **[]** - output point for the mouse button press event.

Up **[]** - point for the mouse button release event.

Hold **[]** - output point for the mouse button hold event.

Note: the set of output points is automatically generated depending on the **InputEvent** parameter.

Parameters:

Button – mouse button identifier.

InputEvent – type of event for processing.

MousePosition

An element to get the current position of the mouse cursor.

Input points:

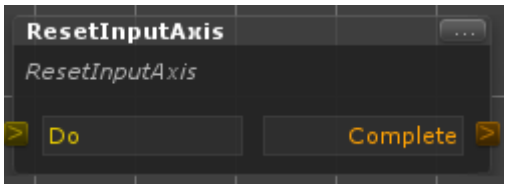
Get **[]** - input point to get the value of the mouse cursor position.

Output points:

Position **[Vector3]** - output point that transmits the value of the mouse position in the screen coordinates.

ResetInputAxis

An element for resetting the states of the input axes.



Input points:

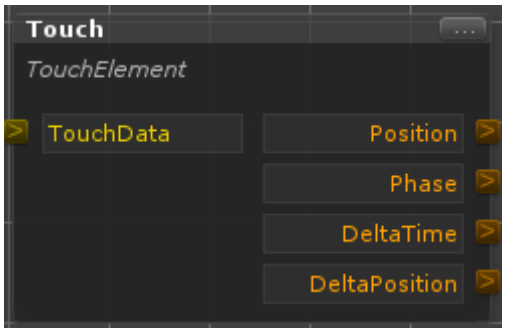
Do **[]** - input point for the command to reset the states of the input axes.

Output points:

Complete **[]** - output point that is called after the command to reset the states of the input axes.

Touch

An element for interpreting data from interaction with the screen of the mobile device.



Input points:

TouchData **[]** - input point for the data of touching the screen.

Output points:

Position **[Vector2]** - output point transmitting the position of interaction with the screen.

Phase **[TouchPhase]** - output point transmitting the data about type of the event of interaction with the screen.

DeltaTime **[float]** - output point transmitting the time elapsed after the last interaction with the screen.

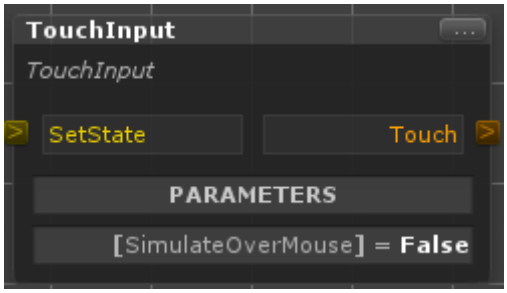
DeltaPosition **[Vector2]** - output point transmitting the difference relative to the last position of interaction with the screen.

Parameters:

Finger – a parameter specifying the index of the touch point (if the device supports multiple touches), at a value of -1, the element interprets all data, otherwise only the data with the corresponding index.

TouchInput

An element for processing the user interaction with the screen of the mobile device.



Input points:

SetState **[bool]** - input point for setting the state of the element (to process or not the input command).

Output points:

Touch**[Touch]** - output point that transmits the data about the current interaction with the screen.

Parameters:

SimulateOverMouse – a flag for simulating touches in the screen through the mouse. Used only in the Unity editor.

TouchLimit – a parameter of limiting the number of simultaneously processed touches. At 0, there are no restrictions.

MultiTouch – a flag, enabling support of multi-touch processing. With the flag turned off, only one touch will be processed (the very first one), in this case the parameter TouchLimit will be ignored.

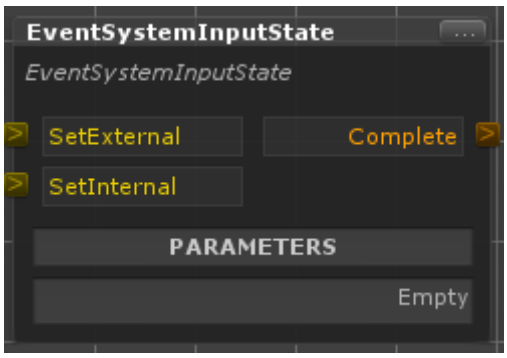
Note: in case of multiple touches, the data for each touch will be transmitted to the output point in the cycle.

EventSystem

A section describing the elements that implement the logic of working with the new Unity event system.

EventSystemInputState

An element for setting the event processing state from casters.



Input points:

SetExternal **[bool]** - input point for setting the state of the element from the outside.

SetInternal **[]** - input point for setting the state of the element according to the internal parameter.

Output points:

Complete **[]** - output point that is called when the state is set.

Parameters:

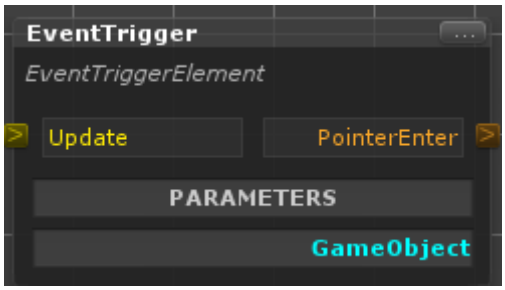
State – event handling state flag.

Auto – a flag of automatic search for all event caster.

RayCaster – a reference to the component of the caster (BaseRaycaster components). It is used in case it is necessary to disable a certain type of events (GUI, 2D or 3D)).

EventTrigger

An element handling the events of user interaction with the object.



Input points:

Update **[]** – input point for element re-initialization.

Output points:

Automatically configured depending on the events set for processing.

Parameters:

Events – a list of event types for processing.

References to diagram variables:

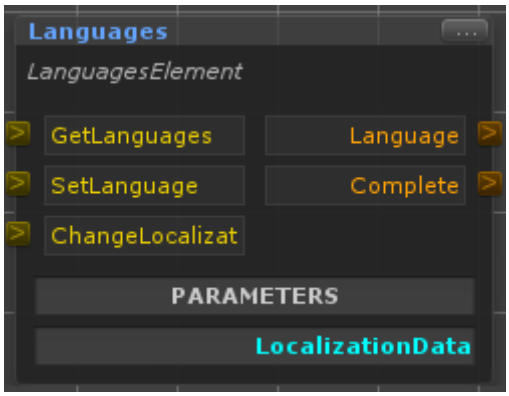
UnityObject – a reference to a variable with the object (GameObject) of the scene, the interaction with which you want to process

Localization

A section with a description of the elements that implement the mechanism for localizing the application.

Languages

An element for working with languages in localization data.



Input points:

GetLanguages**[]** - input point for obtaining a list of languages from localization data.

SetLanguage **[int]** - input point for setting the language by index.

ChangeLocalization **[VSObject(LocalizationData)]** - input point for changing localization data.

Output points:

Language **[]** –output point that passes the string representation of the name of the language from the set. Called in a loop for each language from the set.

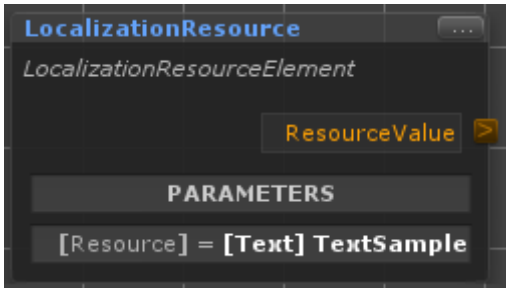
Complete **[]** - output point that is called when the language is set, data is changed, or complete language list is obtained.

References to diagram variables:

UnityObject – a reference to a variable with localization data (LocalizationData Asset).

LocalizationResource

An element for obtaining the localization resource.



Output points:

ResourceValue **[string, VSObject(Sprite), VSObject(Texture), VSObject(AudioClip)]** –output point that transmits data on the localization resource. The type of data transmitted is determined by the selected resource type.

Parameters:

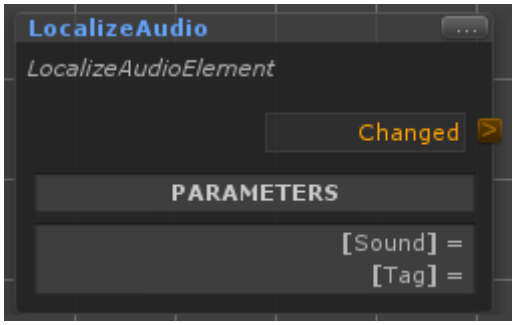
LocalizationData – selection of the localization database.

ResourceType – selection of resource type (for filtering tags).

LocalizationTag – selection of the localization tag.

LocalizeAudio

An element for sound localization.



Output points:

Changed **[]** – output point called by the localization event (when changing the language).

Parameters:

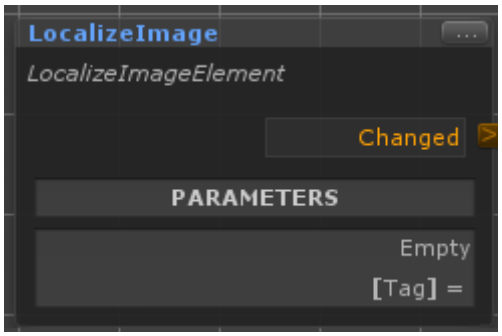
Channel – a channel in which the sound is located.

Sound – the sound that must be localized.

IsLocalized – a flag for setting localization parameters.

LocalizeImage

An element for localization of UGUI Image.



Output points:

Changed **[]** – output point called by the localization event (when changing the language).

Parameters:

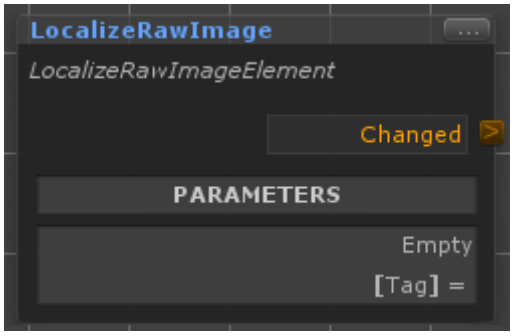
IsLocalized – a flag for setting localization parameters.

References to diagram variables:

UnityObject – reference to a variable that contains the image object (the Image component).

LocalizeRawImage

An element for localization of uGUI RawImage.



Output points:

Changed **[]** – output point called by the localization event (when changing the language).

Parameters:

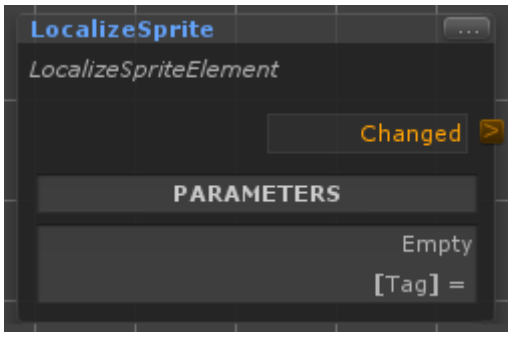
IsLocalized – a flag for setting localization parameters.

References to diagram variables:

UnityObject – reference to a variable that contains the raw image object (the RawImage component).

LocalizeSprite

An element for localization of Sprite.



Output points:

Changed **[]** – output point called by the localization event (when changing the language).

Parameters:

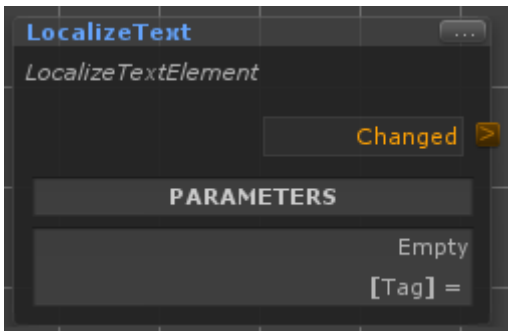
IsLocalized – a flag for setting localization parameters.

References to diagram variables:

UnityObject – reference to a variable that contains the sprite object (the SpriteRenderer component).

LocalizeText

An element for localization of uGUI Text.



Output points:

Changed **[]** – output point called by the localization event (when changing the language).

Parameters:

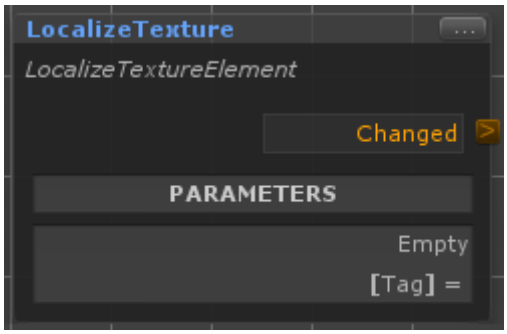
IsLocalized – a flag for setting localization parameters.

References to diagram variables:

UnityObject – reference to a variable that contains the text object (the Text component).

LocalizeTexture

An element for localization of a texture.



Output points:

Changed **[]** – output point called by the localization event (when changing the language).

Parameters:

TextureShaderName – texture identifier in the object material.

MaterialIndex – the index of the material on the object (0 - if there is only one material)

IsLocalized – a flag for setting localization parameters.

References to diagram variables:

UnityObject – reference to a variable that contains the scene object (GameObject).

Math

A section with elements that implement the logic of various mathematical operations and operations with data structures.

Color

ColorGetRGBA

An element for obtaining color channel values from the Color structure.

Input points:

Get **[Color]** - input point for transmitting an instance of the Color structure and obtaining color channel values from it.

Output points:

R **[float]** – the value of the red color channel.

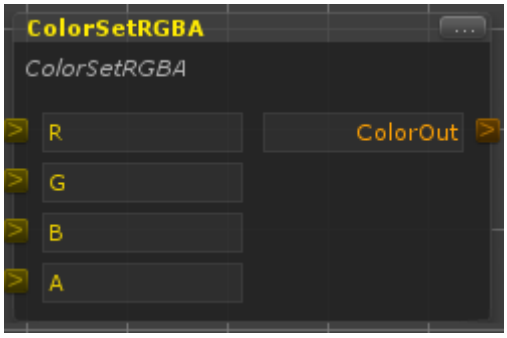
G **[float]** – the value of the green color channel.

B **[float]** – the value of the blue color channel.

A **[float]** – the value of the alpha channel.

ColorSetRGBA

An element for setting the color channel value in the Color structure.



Input points:

R **[float]** – the value of the red color channel.

G **[float]** – – the value of the green color channel.

B **[float]** – the value of the blue color channel.

A **[float]** – the value of the alpha channel.

Set**[]** - input point for setting the color

Output points:

ColorOut **[Color]** - output point for transmitting of the Color structure instance.

Parameters:

DataForSet – the configuration of input points for setting color channel values.

DefaultValue –the default value for the color. When you set a single channel, the remaining channel values are taken from the default value.

References to diagram variables:

DefaultFromVariable – reference to a variable that contains the default value for the color.

ColorLerp

An element for linear interpolation of a color within the specified limits according to the normalized time value. The rate of change of the time value can be changed using the animation curve.

Input points:

SetFrom **[Color]** – input point for setting the initial value.

SeTo **[Color]** – input point for setting the end value.

Tt **[float]** – Input point for transmitting of the normalized time value.

Output points:

Result **[Color]** - output point for transmitting the result of the operation.

Parameters:

DefaultFrom – default start value. If the value was set through the input point, then it will be used.

DefaultTo – default end value. If the value was set through the input point, then it will be used.

References to diagram variables:

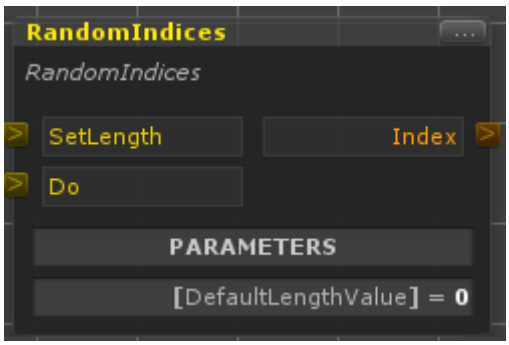
Curve – a reference to the variable containing AnimationCurve. The value may not be set if you do not want to change the rate of time changing.

Random

A section describing the elements for working with random numbers.

RandomIndices

An element for obtaining an array of randomly rearranged array indexes of a given size.



Input points:

SetLength **[int]** – input point for setting the length of the array. If not set, the length of the array will be taken from the parameters of the element.

Do **[]** – input point that performs rearranging of the indices.

Output points:

Index **[int]** - output point for transmitting of the array index. The values are passed in a loop.

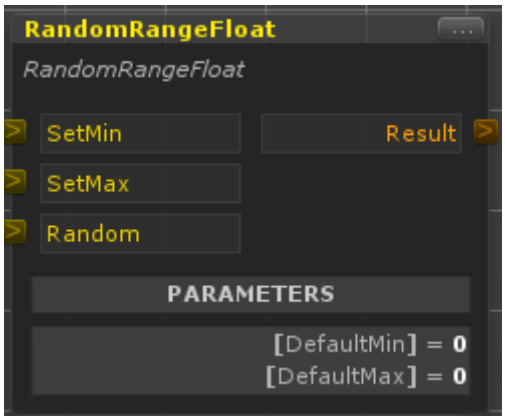
Parameters:

DefaultLengthValue – the default value for the array length. If the value was set through the input point, then it will be used further.

RandomRangeFloat

RandomRangeInt

Elements for obtaining a random value lying in the specified limits for integers (int) and fractional numbers (float) based on the Unity method Random.Range.



Input points:

SetMin **[float, int]** – input point for setting the minimum value.

SetMax **[float, int]** – input point for setting the maximum value.

Random **[]** – input point for obtaining a random value lying between the minimum and maximum.

Output points:

Result **[float, int]** - output point for transmission of the received random value.

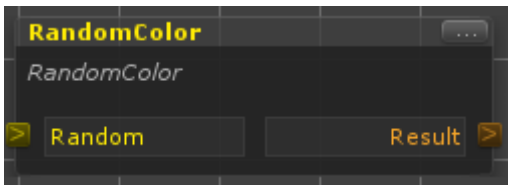
Parameters:

DefaultMin – the default minimum value. If the value was set through the input point, then it will always be used.

DefaultMax – the default maximum value. If the value was set through the input point, then it will always be used.

RandomColor

Element for obtaining a random color value.



Input points:

Random **[]** – input point for obtaining a random color value.

Output points:

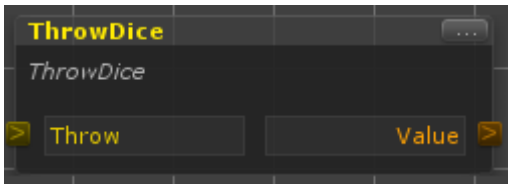
Result **[Color]** - output point for transmitting the result of the operation.

Parameters:

IncludeAlpha – flag for randomization of the alpha channel.

ThrowDice

An element that implements the logic of the roll of the dice with the given values of the faces and the probabilities of their rolling out. Also, the number of faces are set. For the correct functioning of the element, the sum of the probabilities of all faces must be equal to 1 at the normalized value, or 100 at absolute value.



Input points:

Throw **[]** – input point for a throw of dice with the specified parameters.

Output points:

Value **[float]** - output point for transferring the obtained value of the dice face.

Parameters:

Borders – a set of faces of the dice with the set value and the probability of its rolling out.

Simple

A section with elements that perform various simple mathematical operations.

MathAbs

An element for taking a module from a number.

Input points:

Opreand **[int, float]** – Input point for transmitting value.

Output points:

Result **[int, float]** - output point transmitting the result of the operation.

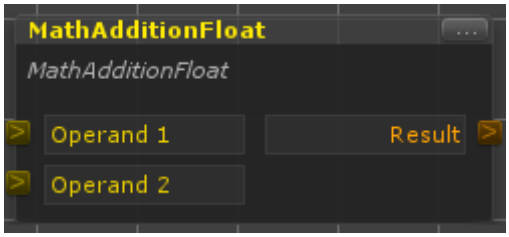
Parameters:

OperandType – the type of value over which the operation is performed (int or float)

MathAdditionFloat

MathAdditionInt

Elements for addition of numbers of type int and float.



Input points:

Automatically configured depending on the set parameter of the element that determines the number of operands.

Output points:

Result **[int, float]** - output point transmitting the result of the addition operation.

Parameters:

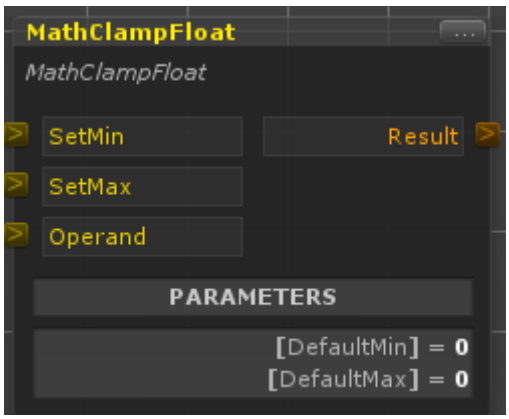
NumberOperands – number of operands.

InternalOperand – basic internal values of the first operand. The default value is zero.

MathClampFloat

MathClampInt

Elements for clamping values of type int and float within the specified limits.



Input points:

SetMin **[float, int]** – input point for setting the minimum value.

SetMax **[float, int]** – input point for setting the maximum value.

Operand **[float, int]** – Input point for transmitting value, which requires clamping.

Output points:

Result **[float, int]** - output point for transmitting the result of the operation.

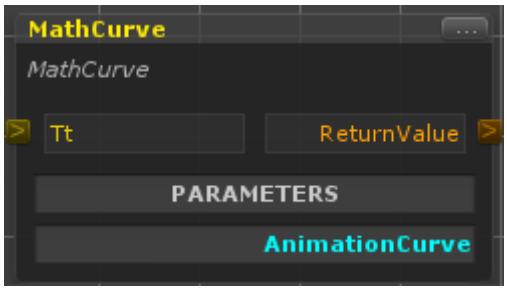
Parameters:

DefaultMin – the default minimum value. If the value was set through the input point, then it will always be used.

DefaultMax – the default maximum value. If the value was set through the input point, then it will always be used.

MathCurve

An element to get the value from the animation curve according to the normalized time value.



Input points:

Tt **[float]** – input point for transmission of the normalized time value.

Output points:

ReturnValue **[float]** - output point that transmits the values of the animation curve.

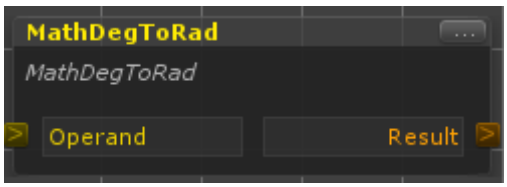
References to diagram variables:

Curve – a reference to the variable containing AnimationCurve.

MathDegToRad

MathRadToDeg

Elements for converting the angle between degrees and radians.



Input points:

Operand **[float]** – Input point for transmitting value in degrees or radians.

Output points:

Result **[float]** - output point transmitting converted to radians or degrees value.

MathFloor

Element for rounding the number to the minimum nearest integer value.

Input points:

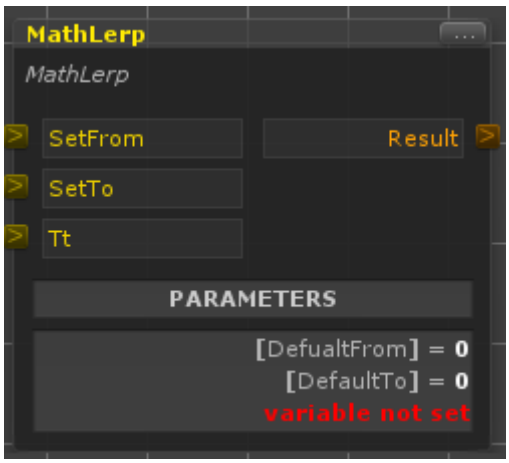
Operand **[float]** – input point for transmitting the source value.

Output points:

Result **[float]** - output point transmitting the result of the operation.

MathLerp

An element for linear interpolation of a number within the specified limits according to the normalized time value. The rate of change of the time value can be changed using the animation curve.



Input points:

SetFrom **[float]** – input point for setting the initial value.

SeTo **[float]** – input point for setting the end value.

Tt **[float]** – Input point for transmitting of the normalized time value.

Output points:

Result **[float]** - output point for transmitting the result of the operation.

Parameters:

DefaultFrom – default start value. If the value was set through the input point, then it will be used.

DefaultTo – default end value. If the value was set through the input point, then it will be used.

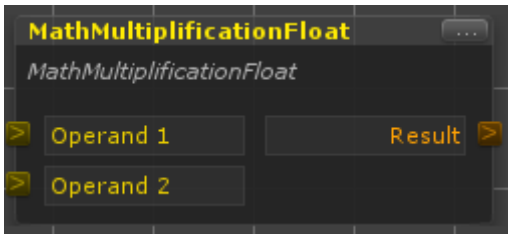
References to diagram variables:

Curve – a reference to the variable containing AnimationCurve. The value may not be set if you do not want to change the rate of time changing.

MathMultiplicationFloat

MathMultiplicationInt

An element for multiplying numbers of type int and float.



Input points:

Automatically configured depending on the set parameter of the element that determines the number of multipliers.

Output points:

Result **[int, float]** - output point transmitting the result of the multiplying operation.

Parameters:

NumberOperands – number of operands.

InternalOperand – base internal values of the first operand. The default value is zero.

MathNormalize

An element for obtaining the normalization of the incoming value relative to the set maximum.

Input points:

SetMax **[float]** – input point for setting the maximum value, relative to which normalization takes place.

Operand **[float]** – Input point for transmitting value, which requires normalization.

Output points:

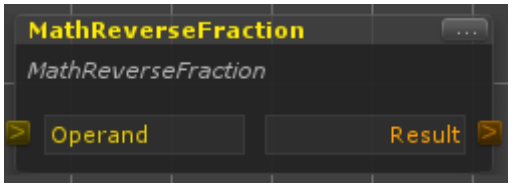
Result **[float]** - output point for transmitting the result of the operation.

Parameters:

DefaultMax – the default maximum value. If the value was set through the input point, then it will always be used.

MathReverseFraction

An element for the reverse of the fraction from the input value.



Input points:

Operand **[float]** – input point for transmitting value.

Output points:

Result **[float]** - output point transmitting the result of the operation.

MathReverseBool

Elements for inversion of the Boolean value.

Input points:

Operand **[bool]** – input point for transmitting the Boolean value.

Output points:

Result **[bool]** - output point transmitting the result of the operation.

MathReverseSignFloat

MathReverseSignInt

Elements for inversion of the sign of a number of type int and float.

Input points:

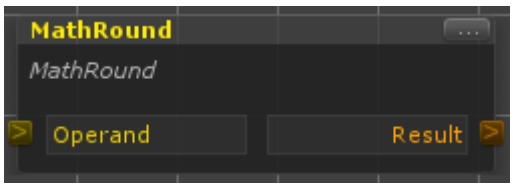
Operand **[bool, float, int]** – input point for transmitting the number

Output points:

Result **[bool, float, int]** - output point transmitting the result of the operation.

MathRound

An element for rounding the number to the nearest integer.



Input points:

Operand **[float]** – input point for transmitting the source value.

Output points:

Result **[float]** - output point transmitting the result of the operation.

Vector

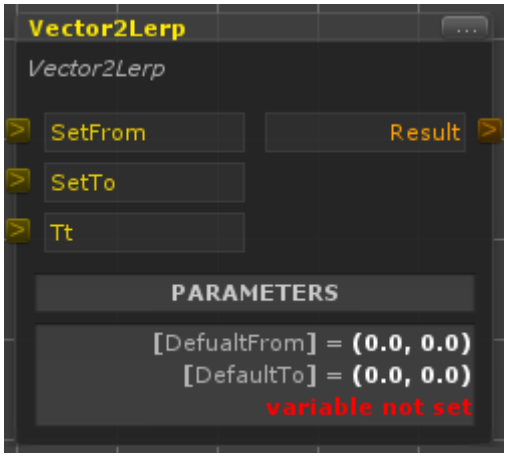
A section with elements that implement operations with working with vectors.

Vector2Lerp

Vector3Lerp

Vector4Lerp

Elements for linear interpolation of a vector within the specified limits, according to the normalized time value.



Input points:

SetFrom **[Vector2/3/4]** – input point for setting the initial value.

SeTo **[Vector2/3/4]** – input point for setting the end value.

Tt **[float]** – Input point for transmission of the normalized time value.

Output points:

Result **[Vector2/3/4]** - output point for transmitting the result of the operation.

Parameters:

DefaultFrom – default start value. If the value was set through the input point, then it will be used.

DefaultTo – default end value. If the value was set through the input point, then it will be used.

References to diagram variables:

Curve – a reference to the variable containing AnimationCurve. The value may not be set if you do not want to change the rate of time changing.

VectorAddition

An element for addition of two vectors.

Input points:

Vector[2/3/4]First **[Vector2/3/4]** – input point for setting the value of the first vector.

Vector[2/3/4]Second **[Vector2/3/4]** – input point for setting the value of the second vector.

Output points:

Result **[Vector2/3/4]** - output point for transmitting the result of the addition operation.

Parameters:

Vector – type of vector. This value configures the input and output points for the specified type.

VectorAngle

An element for obtaining the angle between two vectors. Supported for Vector2 and Vector3, Vector4 will always return 0 value.

Input points:

Vector[2/3]First **[Vector2/3]** – input point for setting the value of the first vector.

Vector[2/3]Second **[Vector2/3]** – input point for setting the value of the second vector.

Output points:

Result **[float]** - output point for transmitting the angle between vectors in degrees.

Parameters:

Vector – type of vector. This value configures the input and output points for the specified type.

VectorCross

An element for obtaining a vector product of two vectors. Supported for Vector2 and Vector3, Vector4 will always return 0 value.

Input points:

Vector[2/3]First **[Vector2/3]** – input point for setting the value of the first vector.

Vector[2/3]Second **[Vector2/3]** – input point for setting the value of the second vector.

Output points:

Result **[Vector2/3]** - output point for transmitting the result of the addition operation.

Parameters:

Vector – type of vector. This value configures the input and output points for the specified type.

VectorDirection

An element for obtaining a vector defining the base direction in the world coordinates. Supported for Vector2 and Vector3.

Input points:

Get – input point for obtaining the direction vector.

Output points:

Automatically configured depending on the set parameters.

Parameters:

Vector – type of vector. This value configures output points for a given type

DirectionType – type of the base direction (Vector2 does not support the forward and back types).

VectorDistance

An element for obtaining the distance between two points.

Input points:

Vector[2/3/4]First **[Vector2/3/4]** – input point for setting the value of the first vector.

Vector[2/3/4]Second **[Vector2/3/4]** – input point for setting the value of the second vector.

Output points:

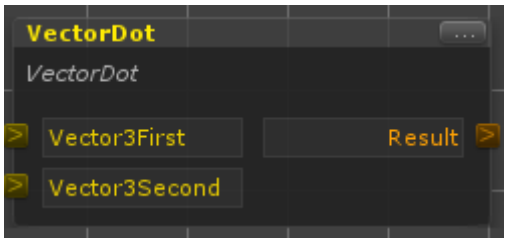
Result **[float]** - output point for transmitting the result of the operation.

Parameters:

Vector – type of vector. This value configures the input points for the specified type.

VectorDot

An element for obtaining the dot product of two vectors.



Input points:

Vector[2/3/4]First **[Vector2/3/4]** – input point for setting the value of the first vector.

Vector[2/3/4]Second **[Vector2/3/4]** – input point for setting the value of the second vector.

Output points:

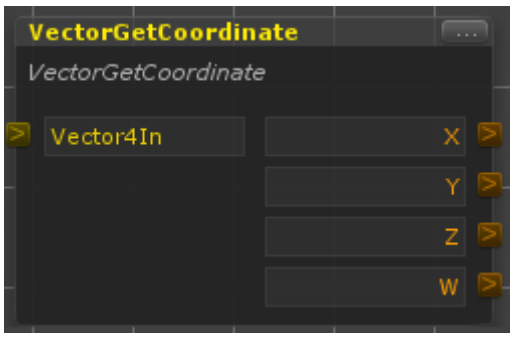
Result **[float]** - output point for transmitting the result of the operation.

Parameters:

Vector – type of vector. This value configures the input points for the specified type.

VectorGetCoordinate

An element for obtaining the coordinates of the incoming vector.



Input points:

Vector[2/3/4]In **[Vector2/3/4]** – input point for setting the value of the vector whose coordinates you want to get.

Output points:

X **[float]** - output point for the transmission of the coordinate x.

Y **[float]** - output point for the transmission of the coordinate y.

Z **[float]** - output point for the transmission of the coordinate z.

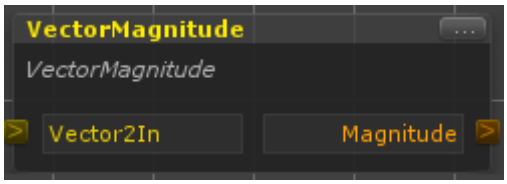
W **[float]** - output point for the transmission of the coordinate w.

Parameters:

Vector – type of vector. This value configures the input and output points for the specified type.

VectorMagnitude

An element for obtaining the magnitude (length) of the vector.



Input points:

Vector[2/3/4]In **[Vector2/3/4]** – input point for setting the value of the vector whose coordinates you want to get.

Output points:

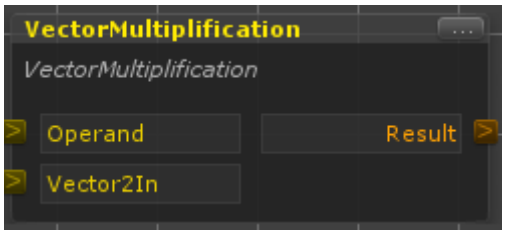
Magnitude **[float]** - output point for the transmission of the magnitude of the incoming vector.

Parameters:

Vector – type of vector. This value configures the input points for the specified type.

VectorMultiplification

An element for multiplying a vector by a number.



Input points:

Vector[2/3/4]In **[Vector2/3/4]** – input point for setting the value of the vector that you want to multiply.

Operand **[float]** – input point for transmitting the number by which to multiply the vector.

Output points:

Result **[Vector2/3/4]** - output point for transmitting the result of the multiplying operation.

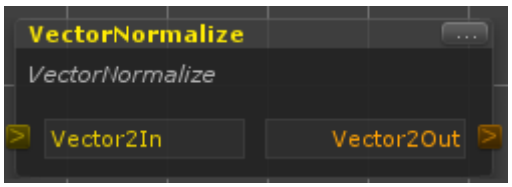
Parameters:

Vector – type of vector. This value configures the input and output points for the specified type.

InternalOperand – value of the number by which to multiply the vector by default. If the number has been set through the input point, then it will be used further.

VectorNormalize

An element for the normalization of the vector.



Input points:

Vector[2/3/4]In **[Vector2/3/4]** – input point for setting the value of the vector that you want to normalize.

Output points:

Vector[2/3/4]Out **[Vector2/3/4]** - output point for transmitting the result of the normalization operation.

Parameters:

Vector – type of vector. This value configures the input and output points for the specified type.

VectorSetCoordinate

An element for setting the values of coordinates in a vector.

Input points:

X **[float]** - input point for setting the x coordinate.

Y **[float]** - input point for setting the y coordinate.

Z **[float]** - input point for setting the z coordinate.

W **[float]** - input point for setting the w coordinate.

Output points:

Vector[2/3/4] **[Vector2/3/4]** - output point for transmitting of the resulting vector.

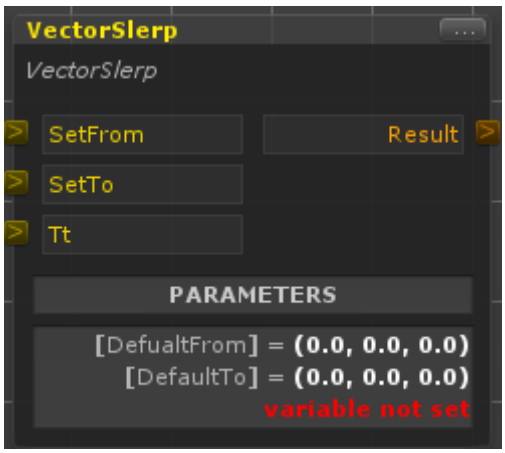
Parameters:

Vector – type of vector. This value configures the input and output points for the specified type.

DefaultValue – the default value for the vector. If you set any coordinate, the rest of the values are taken from the default vector.

VectorSlerp

Elements for spherical interpolation of a vector within the specified limits, according to the normalized time value.



Input points:

SetFrom **[Vector3]** – input point for setting the initial value.

SeTo **[Vector3]** – input point for setting the end value.

Tt **[float]** – Input point for transmission of the normalized time value.

Output points:

Result **[Vector3]** - output point for transmitting the result of the operation.

Parameters:

DefaultFrom – default start value. If the value was set through the input point, then it will be used.

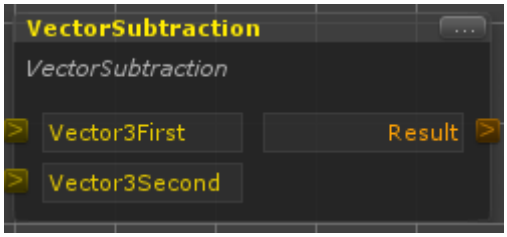
DefaultTo – default end value. If the value was set through the input point, then it will be used.

References to diagram variables:

Curve – a reference to the variable containing AnimationCurve. The value may not be set if you do not want to change the rate of time changing.

VectorSubtraction

An element for subtraction of two vectors (the second vector is subtracted from the first vector).



Input points:

Vector[2/3/4]First **[Vector2/3/4]** – input point for setting the value of the first vector.

Vector[2/3/4]Second **[Vector2/3/4]** – input point for setting the value of the second vector.

Output points:

Result **[Vector2/3/4]** - output point for transmitting the result of the subtraction operation.

Parameters:

Vector – type of vector. This value configures the input and output points for the specified type.

Mobile

UnityAds

An element for advertising with Unity Ads.

Input points:

Show **[]** – input point to show Ads.

Output points:

Finished **[]** – output point for an event of viewed Ad.

Skipped **[]** – output point for an event of skipped Ad.

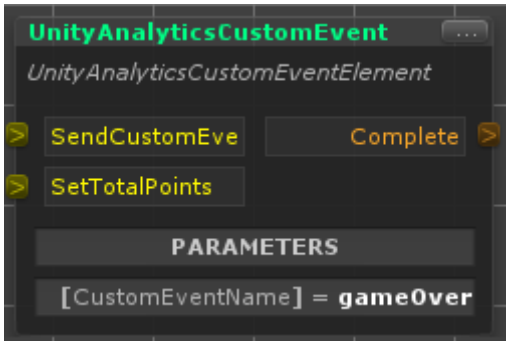
Failed **[]** – output point for an event of Ad error.

Parameters:

PlacementId – ID of the placement (read more in the Unity Ads documentation).

UnityAnalyticsCustomEvent

An element for sending a user event to Unity Analytics.



Input points:

SendCustomEvent **[]** – input point for sending an event.

Output points:

Complete **[]** – output point that is called when the event is successfully sent

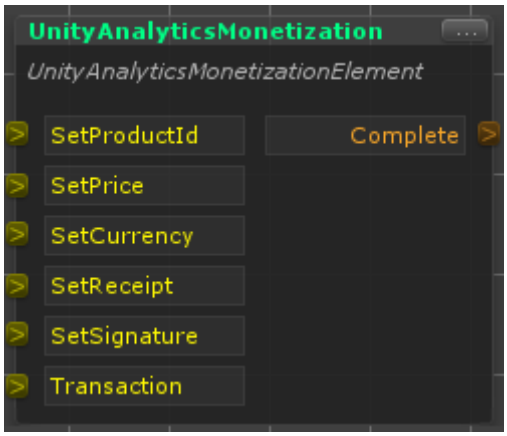
Parameters:

CustomEventName – event name.

CustomEventData – a list of data contained in the event. Each item contains a name and a data type. This list automatically generates input points for setting data values.

UnityAnalyticsMonetization

An element for sending monetization data to Unity Analytics (data on purchases).



Input points:

SetProductId **[string]** – input point for setting the ID of the purchased product

SetPrice **[float]** – input point for setting the product price.

SetCurrency **[string]** – input point for setting the product currency.

SetReceipt **[string]** – input point for setting the purchase information used for verification.

SetSignature **[string]** – input point for setting the Android signature (see the documentation for details).

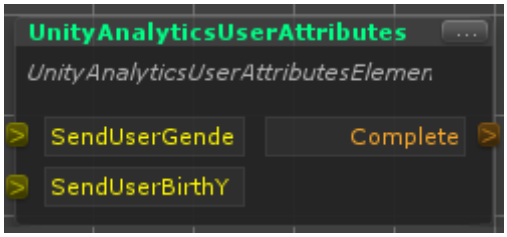
Transaction **[]** – input point for sending the transaction event.

Output points:

Complete **[]** – output point that is called when the event is successfully sent.

UnityAnalyticsUserAttribute

An element for sending the user data to Unity Analytics.



Input points:

SendUserGender **[int]** – input point for sending information about the user's gender (0 - Male, 1 - Female).

SendUserBirthYear **[int]** – input point for sending information about the user's year of birth (only 4 digits of the year).

Output points:

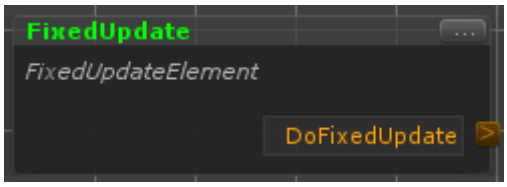
Complete **[]** – output point that is called when the event is successfully sent.

Mono

A section describing the elements that implement the processing of calls to MonoBehavior methods.

FixedUpdate

The processing element of the FixedUpdate method, called by fixed time intervals, defined by the fixedDeltaTime parameters in the Unity editor's physics settings.

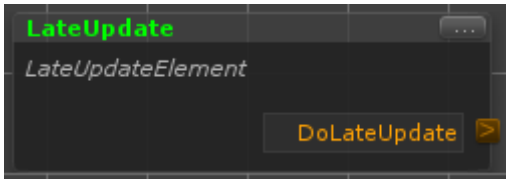


Output points:

DoFixedUpdate **[]** – output point to call on each FixedUpdate.

LateUpdate

Element for processing the method LateUpdate, called after the update of the frame.

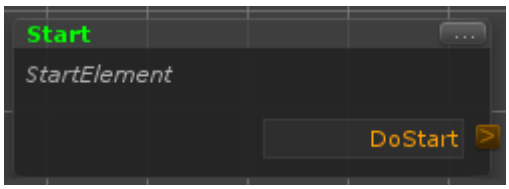


Output points:

DoLateUpdate **[]** – output point that is called on each LateUpdate.

Start

An element for processing the Start method, called before the first frame is rendered. This function called once for each logic element.

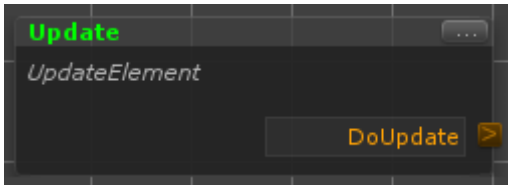


Output points:

DoStart **[]** – output point that is called when the Start method is called.

Update

The processing element of the Update method that is called each frame to update it.



Output points:

DoUpdate **[]** – output point that is called each time Update is called.

Physics

A section with a description of the elements that implement the logic of working with the physical Unity subsystem.

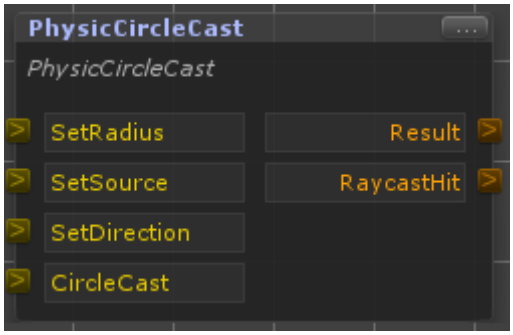
2D

This section describes the elements that work with 2D physics.

Casts

PhysicsCircleCast

An element for casting a circle and determining intersections with it.



Input points:

SetRadius **[float]** – input point for setting the radius of the circle.

SetSource **[Vector2]** – input point for setting the start point of the casting of the circle.

SetDirection **[Vector2]** – input point for setting the direction of the casting of the circle.

CircleCast **[]** – input point for launching a casting of a circle and determining the intersections with it.

Output points:

Result **[bool]** - output point that transmits the result of the casting (true or false).

RaycastHit **[RaycastHit2D]** - output point transmitting intersection parameters. In case the CastAll flag is enabled, the output point is called in the loop for all intersections, otherwise only for the first one.

Parameters:

Layers – setting the filter of the scene layers in which the intersection check takes place.

MaxDistance – the maximum distance from the starting point to the object with which the collision will be checked.

CastAll – a flag that enables checking of the intersection with all objects on the circle path, if it is off, then the check will be performed until the first intersection.

DefaultRaduis – radius of the sphere by default. If the value was set through the input point, then it will be used further.

PhysicsRaycast2D

An element for ray tracing and determining the intersection with it.

Input points:

SetSource **[Vector2]** – input point for setting the start point of the trace.

SetDirection **[Vector2]** – input point for setting the end point of the trace.

RayCast **[]** – input point for starting a ray casting and determining the intersections with it.

Output points:

Result **[bool]** - output point that transmits the result of the casting (true or false).

RaycastHit **[RaycastHit2D]** - output point transmitting intersection parameters. In case the CastAll flag is enabled, the output point is called in the loop for all intersections, otherwise only for the first one.

Parameters:

Layers – setting the filter of the scene layers in which the intersection check takes place.

MaxDistance – the maximum distance from the starting point to the object with which the collision will be checked.

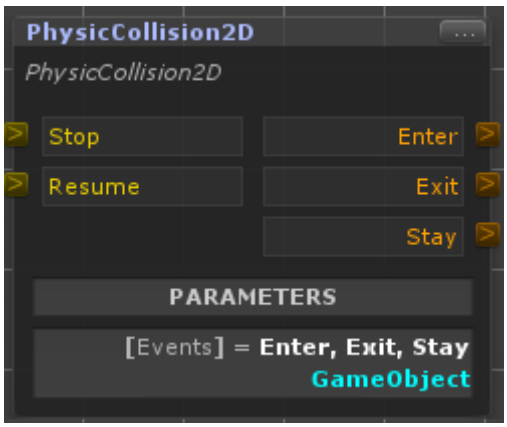
CastAll – a flag that enables checking of the intersection with all objects on the circle path, if it is off, then the check will be performed until the first intersection.

Events

PhysicsCollision2D

PhysicsTrigger2D

Elements for handling collision events and the intersection of a 2D object with other 2D objects.



Input points:

Stop **[]** – input point to stop the operation of the element (events are stopped being processed).

Resume **[]** – input point for the resume of the element.

Output points:

Enter **[Collision2D, Collider2D]** - output point called by a collision / intersection with the object.

Exit **[Collision2D, Collider2D]** - output point called by finishing collision / intersection with the object.

Stay **[Collision2D, Collider2D]** - output point called each frame while in a collision / intersection with the object.

Parameters:

Events – setting the processed events. This parameter automatically configures output points.

SubscribeOnStart – a flag indicating that the element starts working immediately. Otherwise, you need to call through the Resume input point.

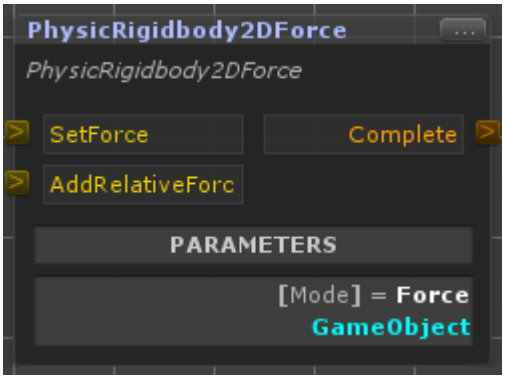
References to diagram variables:

UnityObject – a reference to a variable with an object (GameObject) for which physics events processing is required.

Force

PhysicsRigidbody2DForce

An element for applying a force to a 2D object.



Input points:

SetForce **[Vector2]** – input point for setting the force vector.

SetPosition **[Vector2]** – input point for setting a point on the object to which the force will be applied.

AddForce **[]** – input point for applying the force vector to the object.

AddRelativeForce **[]** – input point for applying a force vector to an object relative to its coordinate system.

AddForcAtPos **[]** – input point for applying the force vector to a given point of the object.

Note: the configuration of the input points depends on the parameters set in the element.

Output points:

Complete **[]** - output point that is called when the element work is finished.

Parameters:

PhysicForce – setting the method of applying force to the object.

DefaultForce – default value of the applied force.

DefaultPosition – default value of the force application point.

Mode – mode of applying force to the object (constant or impulse).

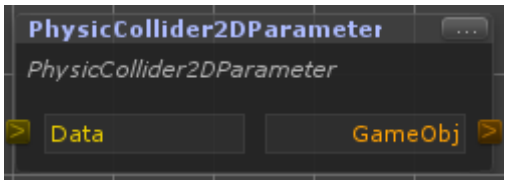
References to diagram variables:

UnityObject – a reference to a variable with an object (GameObject) to which you must apply a force.

Parameter

PhysicsCollider2DParameter

An element for obtaining data of intersection of 2D objects.



Input points:

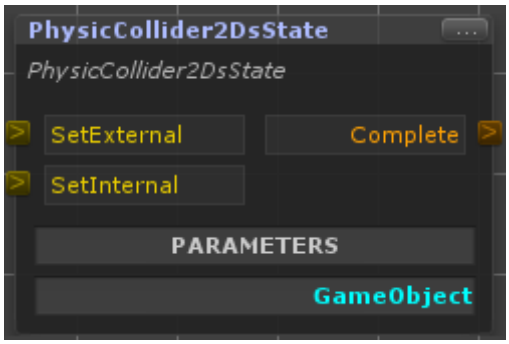
Data **[Collider2D]** – input point for transmitting data about the object with which the intersection occurred.

Output points:

GameObj **[VSObject(GameObject)]** - output point that transmits a reference to the scene object with which the intersection occurred.

PhysicsCollider2DsState

An element to enable or disable all 2D colliders on the scene object.



Input points:

SetExternal **[bool]** – input point for switching on or off 2D colliders according to external value.

SetInternal **[]** – input point for switching on or off 2D colliders according to the element parameter.

Output points:

Complete **[]** - output point that is called when the element work is finished.

Parameters:

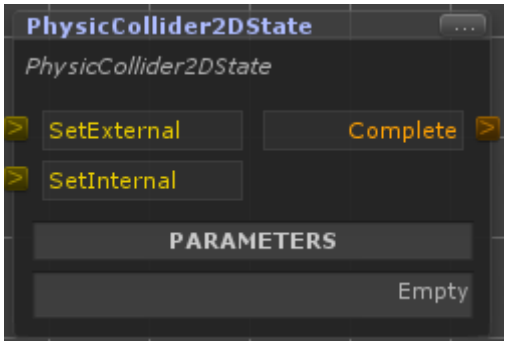
InternalValue – the state of the colliders that should be set.

References to diagram variables:

UnityObject – a reference to the variable with the object (GameObject) of the scene.

PhysicsCollider2DState

An element for switching the given 2D collider on or off.



Input points:

SetExternal **[bool]** – input point for switching the 2D collider on or off according to the external value.

SetInternal **[]** – input point for switching the 2D collider on or off according to the element parameter.

Output points:

Complete **[]** - output point that is called when the element work is finished.

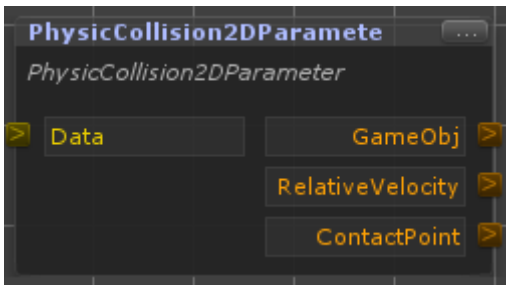
Parameters:

InternalValue – the state of the collider to be set.

2DCollider – a reference to the Collider2D component of the scene object.

PhysicsCollision2DParameter

An element for obtaining data on the collision of 2D objects.



Input points:

Data **[Collision2D]** – input point for transmitting data about the object with which the intersection occurred.

Output points:

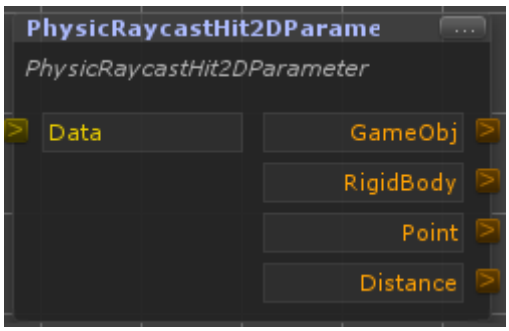
GameObj **[VSObject(GameObject)]** - output point that transmits a reference to the scene object with which the collision occurred.

RelativeVelosity **[Vector2]** - output point that transmits the collision velocity vector in the coordinates of the object.

ContactPoint **[Vector2]** - output point transmits the coordinates of the point of contact in a collision.

PhysicsRaycastHit2DParameter

An element for obtaining 2D trace data for ray / sphere.



Input points:

Data **[RaycastHit2D]** – input point for transmitting data about intersection between a ray and an object.

Output points:

GameObj **[VSObject(GameObject)]** - output point transmitting a reference to the scene object with which the intersection occurred.

Rigidbody **[VSObject(Rigidbody2D)]** - output point transmitting a reference to the Rigidbody2D component (if it exists) of the scene object with which the intersection occurred.

Point **[Vector2]** - output point transmitting the coordinates of the point of contact in a collision.

Distance **[float]** - output point that transmits the distance from the point the beginning of the trace to the object with which the intersection occurred.

PhysicsRigidbody2D

An element for setting the parameters of the Rigidbody2D component of the scene object.

Input points:

SetGravityScale **[float]** – input point for setting the gravity force scaling ratio for the object.

SetMass **[float]** – input point for setting the mass of the object.

SetDrag **[float]** – input point for setting the drag ratio.

SetAngularDrag **[float]** – input point for setting the angular drag ratio.

Output points:

Complete **[]** - output point that is called when the element work is finished.

References to diagram variables:

UnityObject – a reference to the variable with the object (GameObject) of the scene.

PhysicsRigidbody2DState

An element for controlling the state of the physical representation of a 2D object.

Input points:

Sleep **[]** – input point for switching an object to a sleep mode (the object ceases to be simulated by the physical subsystem).

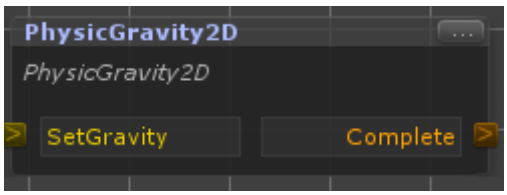
WakeUp **[]** – input point for waking the object.

References to diagram variables:

UnityObject – a reference to the variable with the object (GameObject) of the scene.

PhysicsGravity2D

An element for setting the vector and the force of gravity for 2D physics.



Input points:

SetGravity **[Vector2]** – an input point for setting the vector and the force of gravity.

Output points:

Complete **[]** - output point that is called when the element work is finished.

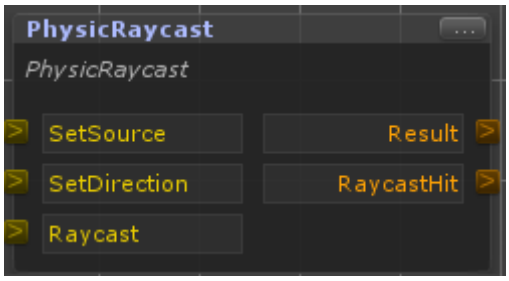
3D

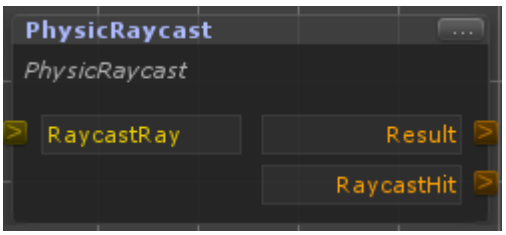
A section describing the elements that work with 2D physics

Casts

PhysicsRaycast

Element of ray tracing and defining intersections with the ray.





Input points:

SetSource **[Vector3]** – input point for setting the start point of the trace.

SetDirection **[Vector3]** – input point for setting the direction vector of the trace.

Raycast **[]** – input point for starting a ray casting and determining the intersections with the ray.

RaycastRay **[Ray]** – an input point for starting a ray casting from external data (Ray).

Output points:

Result **[bool]** - output point that transmits the result of the casting (true or false).

RaycastHit **[RaycastHit]** - output point transmitting intersection parameters. In case the CastAll flag is enabled, the output point is called in the loop for all intersections, otherwise only for the first one.

Parameters:

Layers – setting the filter of the scene layers in which the intersection check takes place.

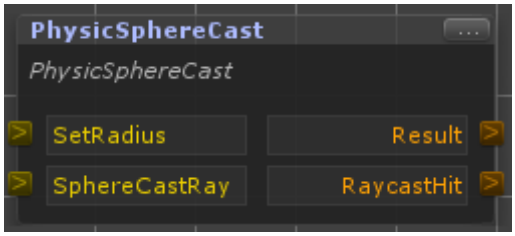
MaxDistance – the maximum distance from the starting point to the object with which the collision will be checked.

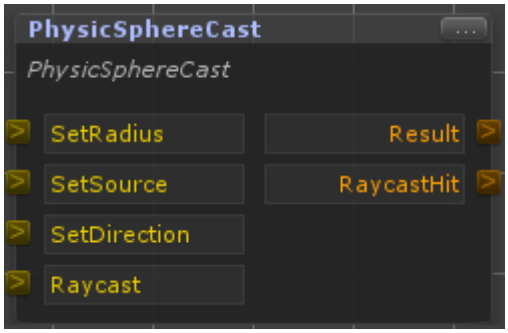
CastAll – a flag that enables checking of the intersection with all objects on the circle path, if it is off, then the check will be performed until the first intersection.

PhysicsRaycast – set the method of obtaining data for the trace.

PhysicsSphereCast

An element for tracing a sphere and determining intersections with it.





Input points:

SetRadius **[float]** – input point for setting the radius of the sphere.

SetSource **[Vector3]** – input point for setting the start point of the trace.

SetDirection **[Vector3]** – input point for setting the direction vector of the trace.

Raycast **[]** – input point for starting the casting of the sphere and determining the intersections with it.

SphereCastRay **[Ray]** – input point for starting casting of the sphere along a ray from external data (Ray).

Output points:

Result **[bool]** - output point that transmits the result of the casting (true or false).

RaycastHit **[RaycastHit]** - output point transmitting intersection parameters. In case the CastAll flag is enabled, the output point is called in the loop for all intersections, otherwise only for the first one.

Parameters:

Layers – setting the filter of the scene layers in which the intersection check takes place.

MaxDistance – the maximum distance from the starting point to the object with which the collision will be checked.

CastAll – a flag that enables checking of the intersection with all objects on the circle path, if it is off, then the check will be performed until the first intersection.

PhysicsRaycast – set the method of obtaining data for the trace.

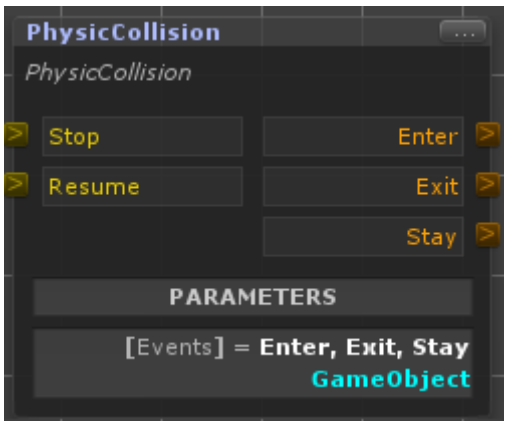
DefaultRadius – setting the radius of the sphere by default. If the value was set through the input point, then it will be used further.

Events

PhysicsCollision

PhysicsTrigger

Elements for handling collision events and the intersection of a 3D object with other 3D objects.



Input points:

Stop **[]** – input point for stopping the operation of the element (events are stopped being processed).

Resume **[]** – input point for the resuming the element operations.

Output points:

Enter **[Collision, Collider]** - output point called by a collision / intersection with the object.

Exit **[Collision, Collider]** - output point called when the collision / intersection with the object ends.

Stay **[Collision, Collider]** - output point called each frame while collision / intersection with the object occurs.

Parameters:

Events – setting the processed events. This parameter automatically configures output points.

SubscribeOnStart – a flag indicating that the element starts working immediately. Otherwise, you need to call through the Resume input point.

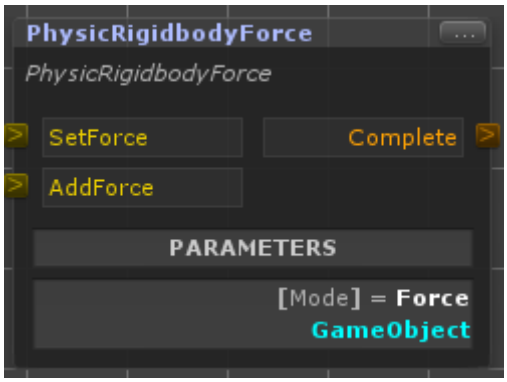
References to diagram variables:

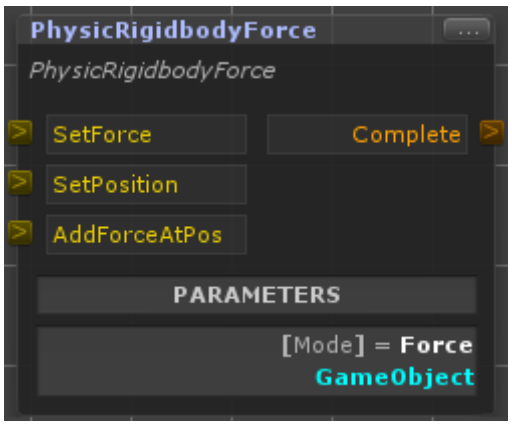
UnityObject – a reference to a variable with an object (GameObject) for which physics events processing is required.

Force

PhysicsRigidbodyForce

An element for applying a force to a 3D object.





Input points:

SetForce **[Vector3]** – input point for setting the force vector.

SetPosition **[Vector3]** –input point for setting a point on the object to which the force will be applied.

AddForce **[]** – input point for applying the force vector to the object.

AddRelativeForce **[]** – input point for applying a force vector to an object relative to its coordinate system.

AddForcAtPos **[]** – input point for applying the force vector to a given point of the object.

Note: the configuration of the input points depends on the parameters set in the element.

Output points:

Complete **[]** - output point that is called when the element work is finished.

Parameters:

PhysicForce – setting the method of applying force to the object.

DefaultForce – default value of the applied force. If the value was set through the input point, then it will be used further.

DefaultPosition – default value of the force application point. If the value was set through the input point, then it will be used further.

Mode – the mode of applying a force to an object (constant or impulsed).

References to diagram variables:

UnityObject – a reference to a variable with an object (GameObject) to which the force should be applied.

PhysicsRigidbodyExplosion

An element of simulation of the explosion effect for a 3D object.

Input points:

SetPosition **[Vector3]** – input point for setting the position of the source of the explosion.

SetForce **[float]** – input point for setting the force of the explosion.

SetRadius **[float]** – input point for setting the explosion radius.

DoExplosion **[]** – input point for applying the explosion effect to the object.

Output points:

Complete **[]** - output point that is called when the element work is finished.

Parameters:

DefaultPosition – default value of the explosion source position. If the value was set through the input point, then it will be used further.

DefaultForce – default value of explosion force. If the value was set through the input point, then it will be used further.

DefaultRadius – default value of explosion radius. If the value was set through the input point, then it will be used further.

UpwardsModifier – value of the ratio when applying force to the object along the vertical.

Mode – the mode of applying a force to an object (constant or impulsed).

Parameter

PhysicsColliderParameter

An element for obtaining objects intersection data.

Input points:

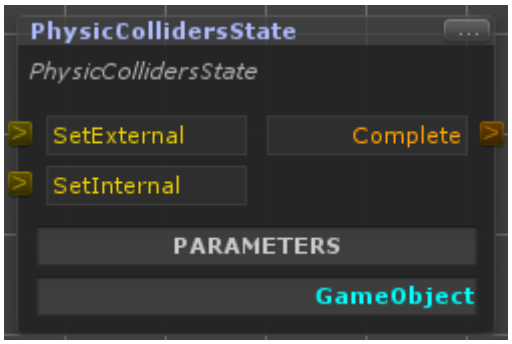
Data **[Collider]** – input point for transmitting data about the object with which the intersection occurred.

Output points:

GameObj **[VSObject(GameObject)]** - output point that transmits a reference to the scene object with which the intersection occurred.

PhysicsCollidersState

An element for enabling or disabling all colliders on the scene object.



Input points:

SetExternal **[bool]** – input point for switching on or off the colliders according to an external value.

SetInternal **[]** – input point for switching on or off the colliders according to the element parameter.

Output points:

Complete **[]** - output point that is called when the element work is finished.

Parameters:

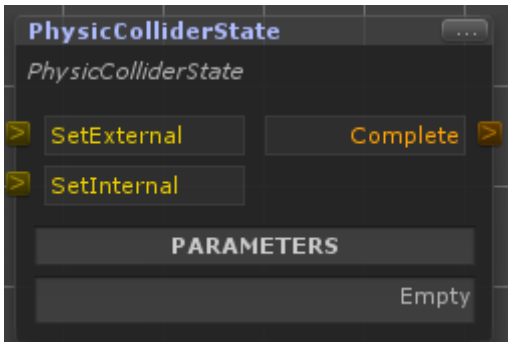
InternalValue – the state of the colliders that should be set.

References to diagram variables:

UnityObject – a reference to the variable with the object (GameObject) of the scene

PhysicsColliderState

An element for switching the given collider on or off.



Input points:

SetExternal **[bool]** – input point for switching on or off the colliders according to an external value.

SetInternal **[]** – input point for switching on or off the colliders according to the element parameter.

Output points:

Complete **[]** - output point that is called when the element work is finished.

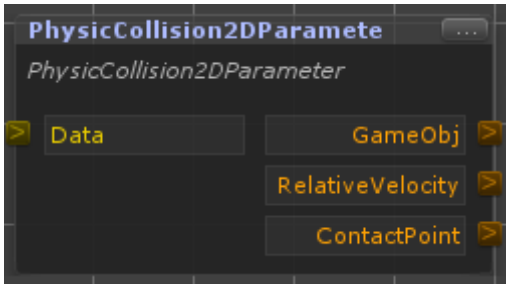
Parameters:

InternalValue – the state of the colliders that should be set.

3DCollider – a reference to the Collider component of the scene object.

PhysicsCollisionParameter

An element for obtaining data on collision of objects.



Input points:

Data **[Collision]** – input point for transmitting data about the object with which the intersection occurred.

Output points:

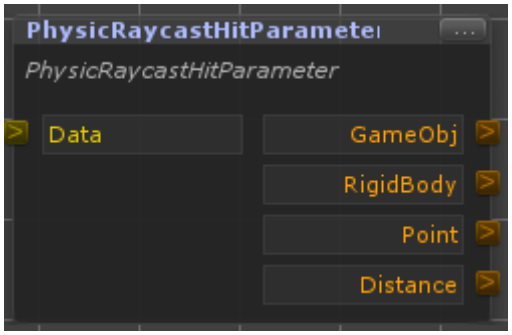
GameObj **[VSObject(GameObject)]** - output point that transmits a reference to the scene object with which the collision occurred.

RelativeVelosity **[Vector3]** - output point that transmits the collision velocity vector in the coordinates of the object.

ContactPoint **[Vector3]** - output point transmitting the coordinates of the point of contact in a collision.

PhysicsRaycastHitParameter

An element for obtaining trace data for ray/sphere.



Input points:

Data **[RaycastHit]** – input point for transmitting data about the intersection of a ray and an object.

Output points:

GameObj **[VSObject(GameObject)]** - output point transmitting a reference to the scene object with which the intersection occurred.

Rigidbody **[VSObject(Rigidbody)]** - output point that transmits a reference to the Rigidbody component (if it exists) of the scene object with which the intersection occurred.

Point **[Vector3]** - output point transmitting the coordinates of the point of contact in a collision.

Distance **[float]** - output point that transmits the distance from the point the beginning of the trace to the object with which the intersection occurred.

PhysicsRigidbody

An element for setting the parameters of the Rigidbody component of the scene object.

Input points:

SetUseGravity **[bool]** – input point for switching gravity for the object on or off.

SetMass **[float]** – input point for setting the mass of the object.

SetDrag **[float]** – input point for setting the drag ratio.

SetAngularDrag **[float]** – input point for setting the angular drag ratio.

Output points:

Complete **[]** - output point that is called when the element work is finished.

References to diagram variables:

UnityObject – a reference to the variable with the object (GameObject) of the scene.

PhysicsRigidbodyState

An element for controlling the state of the physical representation of a 3D object.

Input points:

Sleep **[]** – input point for switching an object to a sleep mode (the object ceases to be simulated by the physical subsystem).

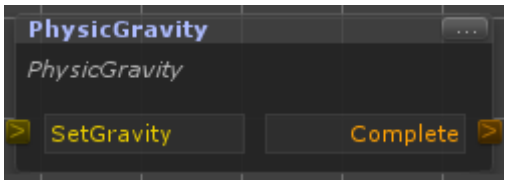
WakeUp **[]** – input point for waking the object.

References to diagram variables:

UnityObject – a reference to the variable with the object (GameObject) of the scene.

PhysicsGravity

An element for setting the vector and the force of gravity for 3D physics.



Input points:

SetGravity **[Vector3]** – input point for setting the vector and the force of gravity.

Output points:

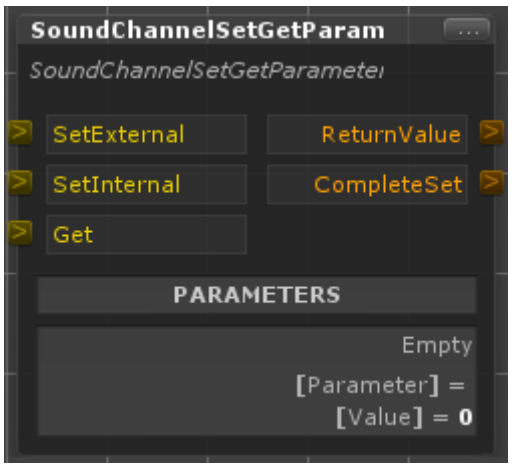
Complete **[]** - output point that is called when the element work is finished.

Sound

A section with a description of the elements that implement the logic of working with the Unity3D sound subsystem.

SoundChannelSetGetParameter

An element for setting channel parameters in AudioMixer.



Input points:

SetExternal **[float]** – input point for setting the channel parameter from external data.

SetInternal **[float]** – input point for setting the channel parameter from internal element value.

Get **[]** – input point for obtaining the parameter value.

Output points:

ReturnValue **[float]** – output point transmitting the value of the parameter.

CompleteSet **[]** – output point that is called when the parameter is set.

Parameters:

Parameter – a string identifier of the parameter set for AudioMixer.

Value – parameter value

References to diagram variables:

UnityObject – a reference to a variable containing AudioMixer asset.

SoundGroup

An element for working with a group of sounds.

Input points:

Play **[]** – input point for starting a sound from a group.

Stop **[]** – input point for stopping a sound from a group.

Pause **[]** – input point for pausing a sound from a group.

Output points:

Complete **[]** – output point called after starting, stopping, or pausing the sound.

Parameters:

StartVolume – start volume for starting the sound.

Delay – delay in seconds before the sound starts.

PlayMode – mode for selecting sounds from a group.

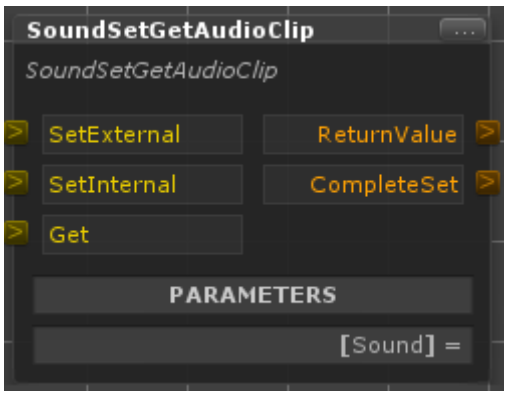
Group – a group of sounds. It is set via AudioMixerGroup.

SoundSetGetAudioClip

SoundSetGetAudioPitch

SoundSetGetAudioVolume

An element for setting and receiving an audio clip, volume and pitch of a sound.



Input points:

SetExternal **[VSObject(AudioClip), float]** – input point for setting an audio clip, volume or pitch from external data.

SetInternal **[]** –input point for setting an audio clip from the internal value of the element.

Get **[]** – input point for obtaining a reference to a sound audio clip.

Output points:

ReturnValue **[VSObject(AudioClip), float]** – output point that transmits a reference to a sound clip, a volume value, or a pitch.

CompleteSet **[]** – output point that is called when the audio clip is set.

Parameters:

Channel – the channel of the sound.

Sound – the identifier of the sound.

SoundSingle

An element for working with a single sound.

Input points:

Play **[]** – input point for staring the sound.

Stop **[]** – input point for stopping the sound.

Pause **[]** – input point for pausing the sound.

Output points:

Complete **[]** – output point called after starting, stopping, or pausing the sound.

Parameters:

StartVolume – start volume for starting the sound.

Delay – delay in seconds before the sound starts.

ChannelMode – a mode to play sound in the channel.

Channel – the channel of the sound.

Sound – the identifier of the sound.

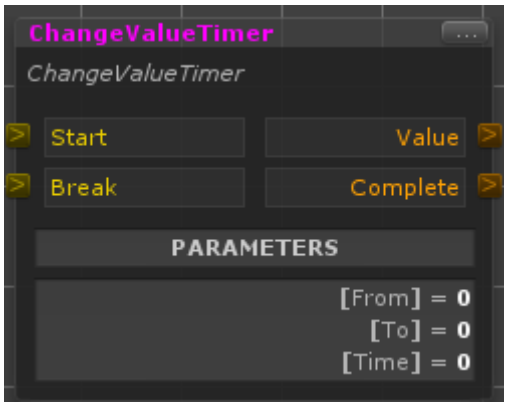
Islocalized – a flag for enabling sound localization and setting parameters for this.

Timers

A section with elements of working with timers.

ChangeValueTimer

An element that implements the timer for changing the value within the specified limits for a specified time.



Input points:

Start **[]** – input point for starting the timer.

Break **[]** – input point for forced stopping the timer.

Output points:

Value **[float]** – output point that transmits the current value.

Complete **[]** – output point that is called when the element work is finished.

Parameters:

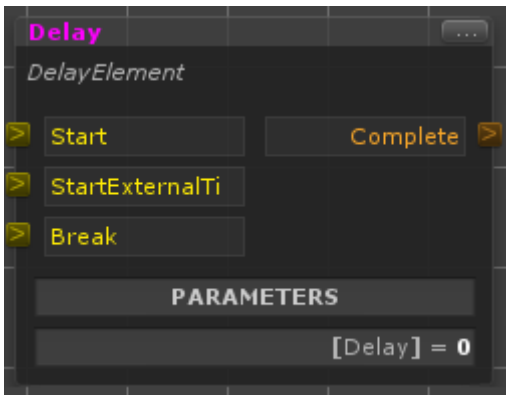
From – starting value.

To – end value.

Time – the time of changing a value from the starting to the end.

Delay

An element implementing a delay in the execution of logic by a given value.



Input points:

Start **[]** – input point for starting the delay.

StartExternalTime **[float]** – input point for starting the delay with the time value transferred to it.

Break **[]** – input point for forced stopping the delay.

Output points:

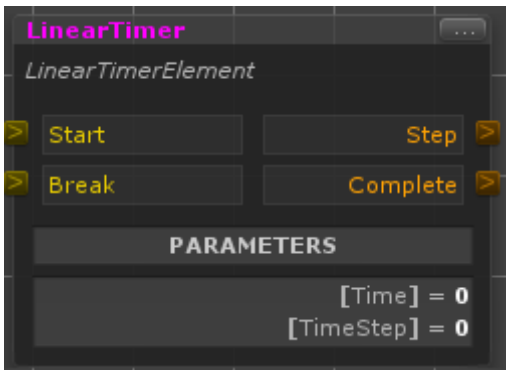
Complete **[]** – output point that is called when the element work is finished.

Parameters:

Delay – delay in seconds.

LinearTimer

An element that implements a timer with a step-by-step call at regular intervals



Input points:

Start **[]** – input point for starting the timer

Break **[]** – input point for forced stopping the timer.

Output points:

Step **[]** – output point that is called every time interval before the timer expires.

Complete **[]** – output point that is called when the element work is finished.

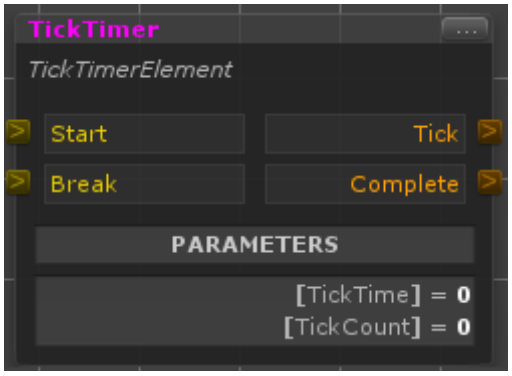
Parameters:

Time – the time of the timer in seconds.

TimeStep – the value of the call step in seconds.

TickTimer

An element that implements a timer with the set number of call steps.



Input points:

Start **[]** – input point for starting the timer.

Break **[]** – input point for forced stopping the timer.

Output points:

Tick **[]** – output point called when the step time is reached.

Complete **[]** – output point that is called when the element work is finished.

Parameters:

TickTime – time for one step.

TickCount – the number of steps.

Time

An element for working with Unity3d system time.

Input points:

GetTime **[]** – input point for obtaining the current time value from the moment the application was launched.

GetRealTime **[]** – input point to obtain the real-time value.

GetDeltalTime **[]** – input point to obtain the time spent on rendering the last frame.

GetDeltalTime **[]** – input point for obtaining a fixed time step.

SetTimeScale **[float]** – input point for setting the time scaling factor.

Output points:

Time **[float]** – output point that transmits the current time value.

RealTime **[float]** – output point that transmits the current real-time value.

DeltaTime **[float]** – output point transmitting the current time spent on rendering the last frame.

FixedDeltaTime **[float]** – output point that transmits the current value of a fixed time step.

Tweens

A section describing the elements that change various data over time, as well as elements that implement various effects associated with changing values.

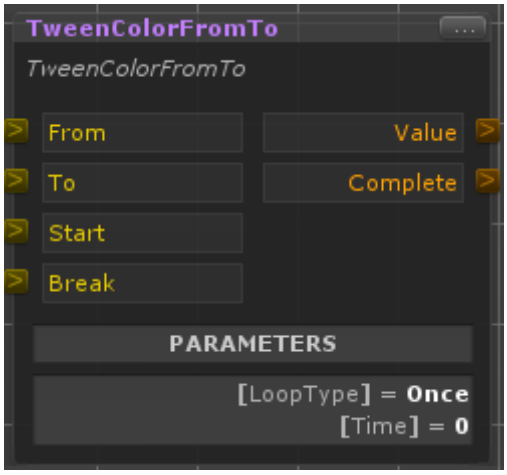
TweenColorFromTo

TweenQuaternionFromTo

TweenVector2FromTo

TweenVector3FromTo

An element for changing the color, quaternion, 2D and 3D vector values within the specified limits.



Input points:

From **[Color, Quaternion, Vector2, Vector3]** – input point for setting the start value.

To **[Color, Quaternion, Vector2, Vector3]** – input point for setting the end value.

Start **[]** – input point for starting the process of changing the value.

Break **[]** – input point to force the element to stop working.

Output points:

Value **[Color, Quaternion, Vector2, Vector3]** – output point transmitting the current value.

Complete **[]** – output point that is called when the element work is finished (If there is no loop mode for changing the value).

Parameters:

LoopType – color change mode.

Time – time for which the color value should change from the initial to the final.

ResetToStartIfBreak – a flag to reset the color to the initial value in the event of a forced stopping of the element.

DefaultFromValue – default starting value. If the value was set through the input point, then it will be used further.

DefaultToValue – default end value. If the value was set through the input point, then it will be used.

References to diagram variables:

Curve – a reference to the variable containing AnimationCurve. Animation curve is for changing the speed of time.

TweenColorUpDown

TweenQuaternionUpDown

TweenVector2UpDown

TweenVector3UpDown

An element for changing the color, quaternion, 2D and 3D vector values within the specified limits with a peak value.

Input points:

From **[Color, Quaternion, Vector2, Vector3]** – input point for setting the start value.

Up **[Color, Quaternion, Vector2, Vector3]** – input point for setting the peak value.

Down **[Color, Quaternion, Vector2, Vector3]** – input point for setting the end value.

Start **[]** – input point for starting the process of changing the value.

Break **[]** – input point to force the element to stop working.

Output points:

Value **[Color, Quaternion, Vector2, Vector3]** – output point transmitting the current value.

Complete **[]** – output point that is called when the element work is finished (If there is no loop mode for changing the value).

Parameters:

LoopType – color change mode.

UpTime – time for which the color value should change from the initial to the peak.

DownTime – time for which the color value should change from the peak to the end.

ResetToStartIfBreak – a flag to reset the color to the initial value in the event of a forced stopping of the element.

DefaultFromValue – default starting value. If the value was set through the input point, then it will be used further.

DefaultUpValue – default peak value. If the value was set through the input point, then it will be used further.

DefaultDownValue – default end value. If the value was set through the input point, then it will be used.

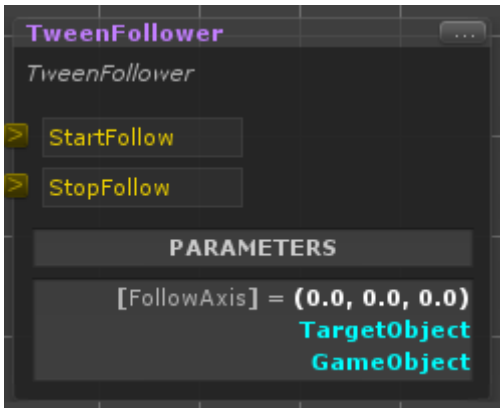
References to diagram variables:

CurveUp – a reference to the variable containing AnimationCurve. The animation curve to change the speed of the speed of time to reach the peak value.

CurveUp – a reference to the variable containing AnimationCurve. The animation curve to change the speed of the speed of time to reach the end value.

TweenFollower

An element that implements the mechanism of following an object after another object.



Input points:

StartFollow **[]** – input point to start following.

StopFollow **[]** – input point to stop following.

Parameters:

FollowAxis – the definition of the axes of the world coordinates along which the object will move when following (at 0 value, the axis motion is ignored).

SmoothAxis – the definition of the rates of smoothing motion along the axes (for a smooth start and end of the movement).

IncludeLookAt – a flag to enable the mode of «looking at» the object.

SmoothLookAt – smoothing rate for the «looking at» mode.

Board – setting the boundaries of the movement of the object. When the boundaries are reached, the object stops following. At 0, the borders are not counted.

> **Note**: for correct operation of the elements, if you need borders on any axis, you must set both values.

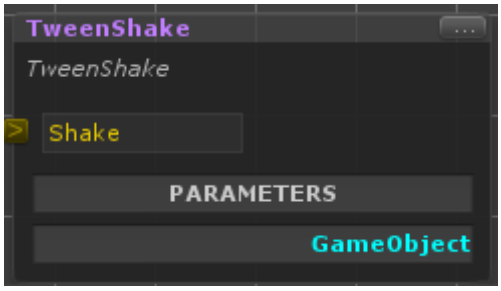
References to diagram variables:

Target – a reference to a variable with an object (GameObject), which should be followed.

UnityObject – a reference to a variable with an object (GameObject), which will followed.

TweenShake

An element that implements the shake effect of the object.



Input points:

Shake **[]** – input point for starting the shake effect.

Parameters:

IntencityPosition – setting the intensity of the changing the position along the axes.

IntencityRotation – setting the intensity of changing the rotation along the axes.

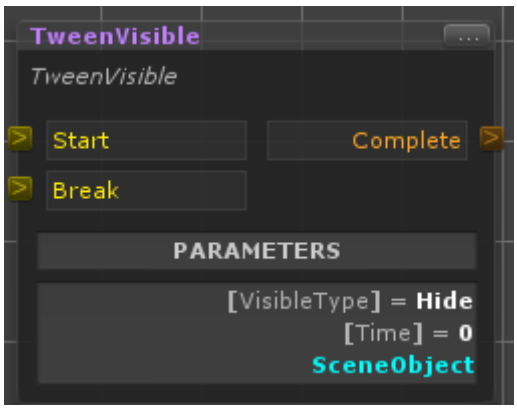
AttenuationTime – the effect strength decay time.

References to diagram variables:

UnityObject – a reference to a variable with an object (GameObject), to which the shake effect applies.

TweenVisible

An element for smooth control of object visibility (2D or 3D), by tuning of transparency.



Input points:

Start **[]** – input point for starting the process of changing the value.

Break **[]** – input point to force the element to stop working.

Output points:

Complete **[]** – output point that is called when the element work is finished.

Parameters:

VisibleType – the element work mode (show or hide the object).

Time – the time for changing object transparency.

TargetVisible – the final normalized value of transparency (from 0 to 1)

IsSprite – a flag indicating that the scene object is a 2D object.

References to diagram variables:

SceneObject – a reference to a variable with a scene object (GameObject) whose transparency should be controlled.

TweenTranslate

An element that moves an object in a given direction.

Input points:

SetDirection **[Vector3]** – input point for setting the direction of movement.

SetSpeed**[float]** – input point for setting the speed of movement.

StartTranslate **[]** – input point for starting the movement.

StopTranslate **[]** – input point for stopping the movement.

Parameters:

SpaceType – the coordinate system in which the movement (local or world) takes place.

DefaultSpeed – default speed value. If the value was set through the input point, then it will be used further.

DefaultDirection – default direction value. If the value was set through the input point, then it will be used further.

References to diagram variables:

UnityObject – a reference to a variable with an object (GameObject).

TweenRotate

An element for object rotation.

Input points:

SetAxis **[Vector3]** – input point for setting the axis of rotation.

SetSpeed**[float]** – input point for setting the speed of rotation.

StartRotate **[]** – input point for starting the rotation.

StopRotate **[]** – input point for stopping the rotation.

Parameters:

SpaceType – the coordinate system in which the rotation (local or world) is performed.

DefaultSpeed – default rotation speed value. If the value was set through the input point, then it will be used further.

DefaultAxis – default rotation axis. If the value was set through the input point, then it will be used further.

References to diagram variables:

UnityObject – a reference to a variable with an object (GameObject).

TweenRotateAround

An element for rotating an object around a point.

Input points:

SetCenterPoint **[Vector3]** – input point for setting the center of rotation.

SetAxis **[Vector3]** – input point for setting the axis of rotation.

SetSpeed**[float]** – input point for setting the speed of rotation.

StartRotate **[]** – input point for starting the rotation.

StopRotate **[]** – input point for stopping the rotation.

Parameters:

SpaceType – the coordinate system in which the rotation (local or world) is performed.

DefaultCenterPoint – default point of the rotation center. If the value was set through the input point, then it will be used further.

DefaultSpeed – default rotation speed value. If the value was set through the input point, then it will be used further.

DefaultAxis – default rotation axis. If the value was set through the input point, then it will be used further.

References to diagram variables:

UnityObject – a reference to a variable with an object (GameObject).

Unity Subsystem

A section with elements that implement the logic of working with various Unity subsystems.

AssetBundle

An element for loading an asset from a bundle by the name.

Input points:

Bundle **[AssetBundle]** – input point for transmitting the reference to AssetBundle.

GetAssetByName **[string]** – input point for transmitting the name of the asset to be downloaded from AssetBundle.

Output points:

UnityObjectAsset **[VSObject]** – output point for passing the reference to the asset (Unity Object).

TextAsset **[TextAsset]** – output point for passing the reference to the text asset.

Parameters:

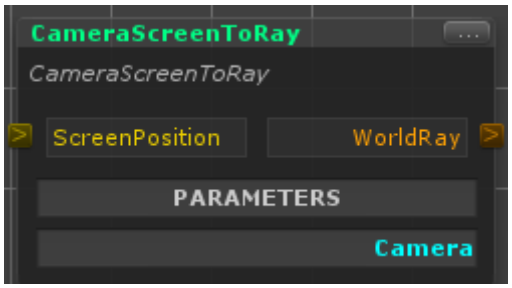
IsTextAsset – flag indicating that the loaded asset is a text asset.

Camera

A section with elements that implement working with the camera.

CameraScreenToRay

An element for obtaining ray parameters from the position of the mouse cursor in the screen coordinates relative to the specified camera.



Input points:

ScreenPosition **[Vector3]** – input point for transmitting the cursor position in screen coordinates.

Output points:

WorldRay **[Ray]** –output point transmitting the ray parameters for subsequent tracing.

Parameters:

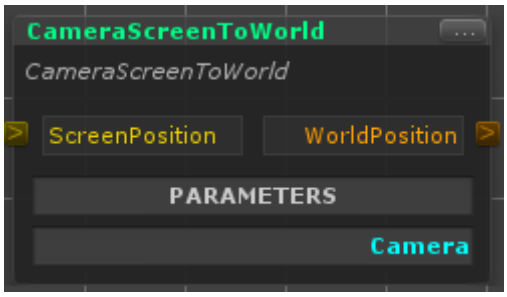
Plane – the distance from the camera along the z-axis at which the screen coordinates are converted.

References to diagram variables:

UnityObject – a reference to a variable with the Camera component.

CameraScreenToWorld

An element for converting the screen coordinates to the world coordinates relative to the specified camera.



Input points:

ScreenPosition **[Vector3]** – input point for transmitting the cursor position in screen coordinates.

Output points:

WorldPosition **[Vector3]** – output point transmitting the position in the world coordinates.

Parameters:

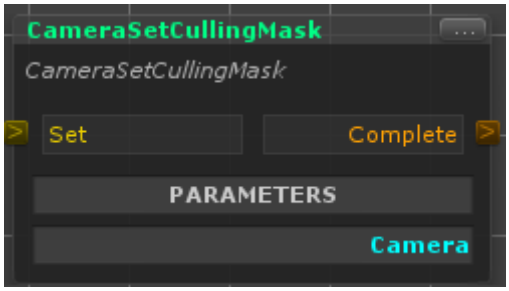
Plane – the distance from the camera along the z-axis at which the screen coordinates are converted.

References to diagram variables:

UnityObject – a reference to a variable with the Camera component.

CameraSetCullingMask

An element for setting a layer mask in the specified camera.



Input points:

Set **[]** – input point for setting the layer mask from the element parameters.

Output points:

Complete **[]** – output point that is called when the parameter is set.

Parameters:

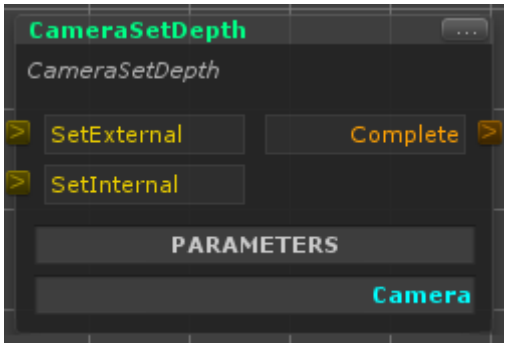
CullingMask – layers rendering mask.

References to diagram variables:

UnityObject – a reference to a variable with the Camera component.

CameraSetDepth

An element for setting the depth of the camera rendering.



Input points:

SetExternal **[float]** – Input point for setting a parameter from an external value.

SetInternal **[]** – Input point for setting a parameter from the internal value of the element.

Output points:

Complete **[]** – output point that is called when the parameter is set.

Parameters:

Depth – camera rendering depth.

References to diagram variables:

UnityObject – a reference to a variable with the Camera component.

CameraSetFoV

An element for setting the camera viewing angle.

Input points:

SetExternal **[float]** – Input point for setting a parameter from an external value.

SetInternal **[]** – Input point for setting a parameter from the internal value of the element.

Output points:

Complete **[]** – output point that is called when the parameter is set.

Parameters:

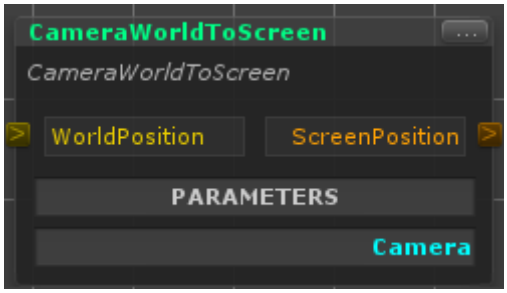
FieldOfView – camera field of view.

References to diagram variables:

UnityObject – a reference to a variable with the Camera component.

CameraWorldToScreen

An element for converting the position in the world coordinates to the screen coordinates relative to the specified camera.



Input points:

WorldPosition **[Vector3]** – input point for transmitting the position in world coordinates.

Output points:

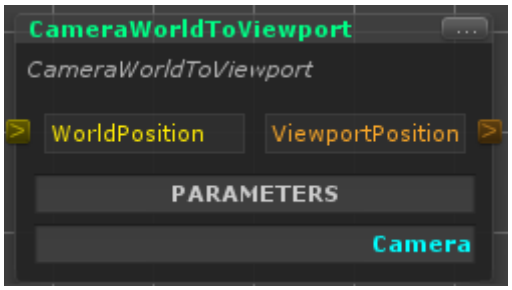
ScreenPosition **[Vector3]** – output point transmitting the position in screen coordinates.

References to diagram variables:

UnityObject – a reference to a variable with the Camera component.

CameraWorldToViewport

An element for converting the position in the world coordinates to the viewort coordinates relative to the specified camera.



Input points:

WorldPosition **[Vector3]** – input point for transmitting the position in world coordinates.

Output points:

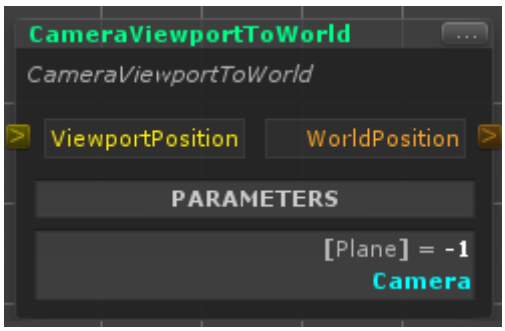
ViewportPosition **[Vector3]** – output point transmitting the position in viewport coordinates.

References to diagram variables:

UnityObject – a reference to a variable with the Camera component.

CameraViewportToWorld

An element for converting the viewport coordinates to the world coordinates relative to the specified camera.



Input points:

ViewportPosition **[Vector3]** – input point for transmitting the cursor position in viewport coordinates.

Output points:

WorldPosition **[Vector3]** – output point transmitting the position in the world coordinates.

Parameters:

Plane – the distance from the camera along the z-axis at which the screen coordinates are converted.

References to diagram variables:

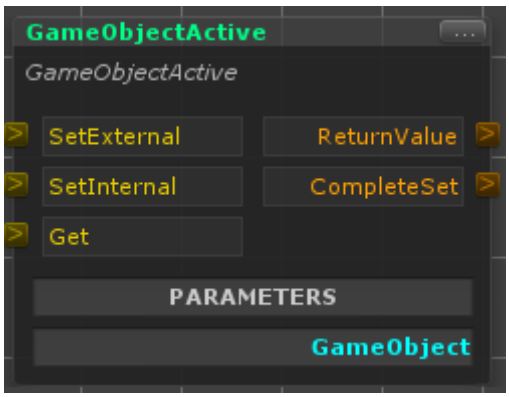
UnityObject – a reference to a variable with the Camera component.

GameObject

A section describing the elements that implement the logic of working with the scene objects.

GameObjectActive

An element for activating / deactivating the object and obtaining the current state of activity of the scene object.



Input points:

SetExternal **[bool]** – Input point for setting a parameter from an external value.

SetInternal **[]** – Input point for setting a parameter from the internal value of the element.

Get **[]** – input point for obtaining the current state of the activity of the object.

Output points:

ReturnValue **[bool]** – output point transmitting the current state of the object activity.

CompleteSet **[]** – output point that is called when the parameter is set.

Parameters:

HideFlag – a flag for configuring input and output points.

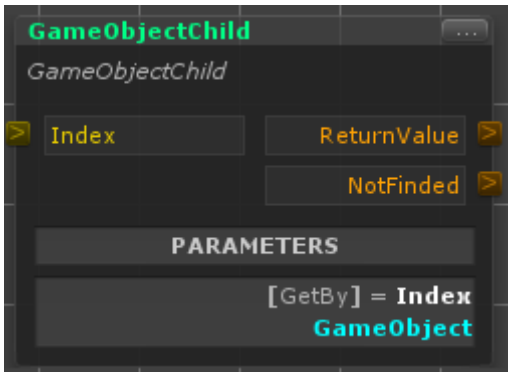
InternalValue – object activity state flag,

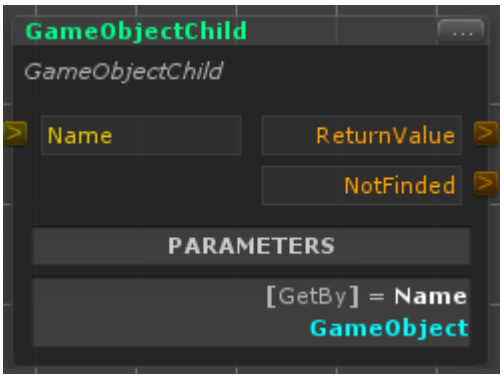
References to diagram variables:

UnityObject – a reference to a variable with a scene object (GameObject).

GameObjectChild

An element to obtain the child object.





Input points:

Index **[int]** – input point for the index of the child object.

Name **[string]** – output point for transmitting the name of the child object.

Output points:

ReturnValue **[VSObject(GameObject)]** – output point transmitting the object reference.

NotFinded **[]** – output point that is called if the child object was not found.

Parameters:

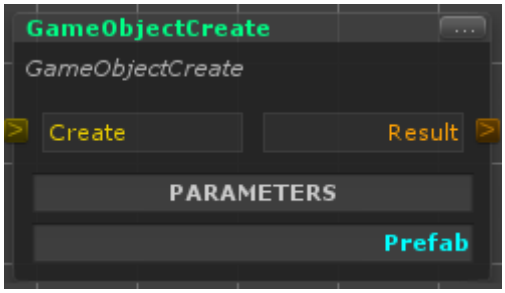
GetBy – setting the method of obtaining the child object (by index or name). This value automatically configures the input points.

References to diagram variables:

UnityObject – a reference to a variable with a scene object (GameObject).

GameObjectCreate

An element for creating scene objects from prefabs or other scene objects.



Input points:

Create **[]** – input point for creating an object in the scene.

Output points:

Result **[VSObject(GameObject)]** – output point that transmits the reference to the created object.

Parameters:

UsePoolManager – a flag enabling the usage of the object pool when creating an object, the prefab name is used as the identifier in the pool.

References to diagram variables:

UnityObject – reference to a variable with a prefab or scene object (GameObject).

GameObjectDestroy

An element for destroying the scene object.

Input points:

Do **[VSObject]** – input point for transmitting a reference to the object to be deleted.

Output points:

Complete **[]** – output point that is called when the object is deleted.

Parameters:

UsePoolManager – a flag to use the object pool when deleting an object

Group – group ID in the object pool.

Delay – the delay before deleting an object.

GameObjectGetComponent

GameObjectAddComponent

An element to get and add the component of the scene object by name.

Input points:

GameObj **[VSObject]** – input point for transmitting a reference to the scene object.

Output points:

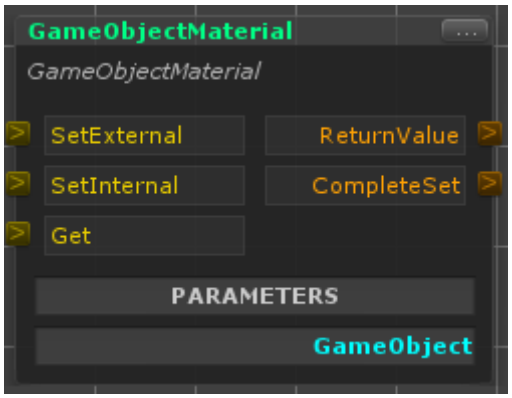
ObjComponent **[VSObject]** – output point that transmits the reference to the component of the scene object.

Parameters:

ComponentName – the name of the component.

GameObjectMaterial

An element for setting and retrieving the material of the scene object.



Input points:

SetExternal **[VSObject(Material)]** – Input point for setting a parameter from an external value.

SetInternal **[]** – Input point for setting a parameter from the internal value of the element.

Get **[]** – input point for retrieving the current object material.

Output points:

ReturnValue **[VSObject(Material)]** – output point transmitting the object material.

CompleteSet **[]** – output point that is called when the parameter is set.

Parameters:

HideFlag – a flag for configuring input and output points.

InternalValue – a reference to the material.

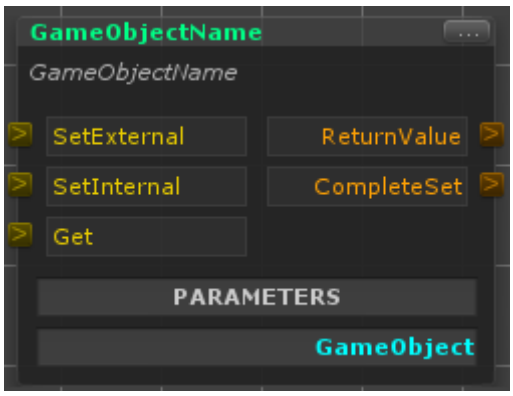
MaterialIndex – the index of material that needs to be set in the object, if the object supports multimaterials, otherwise leave it at 0 value.

References to diagram variables:

UnityObject – a reference to a variable with a scene object (GameObject).

GameObjectName

An element for setting and retrieving the name of the object.



Input points:

SetExternal **[string]** – Input point for setting a parameter from an external value.

SetInternal **[]** – Input point for setting a parameter from the internal value of the element.

Get **[]** – input point for retrieving the current object name.

Output points:

ReturnValue **[string]** – output point transmitting the object name.

CompleteSet **[]** – output point that is called when the parameter is set.

Parameters:

HideFlag – a flag for configuring input and output points.

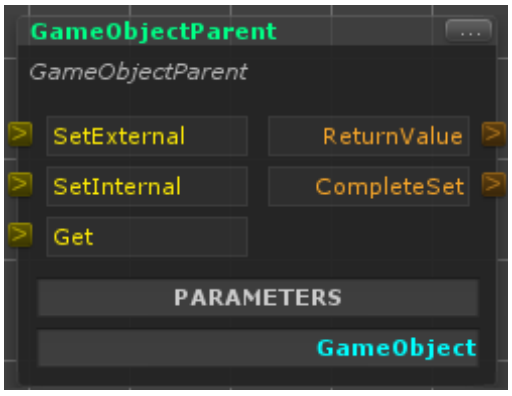
InternalValue – the new name of the object.

References to diagram variables:

UnityObject – a reference to a variable with a scene object (GameObject).

GameObjectParent

An element for setting and retrieving the parent of the object.



Input points:

SetExternal **[VSObject]** – Input point for setting a parameter from an external value.

SetInternal **[]** – Input point for setting a parameter from the internal value of the element.

Get **[]** – input point for retrieving the current object parent.

Output points:

ReturnValue **[VSObject]** – output point transmitting the object parent.

CompleteSet **[]** – output point that is called when the parameter is set.

Parameters:

HideFlag – a flag for configuring input and output points.

InternalValue – the new name of the object.

References to diagram variables:

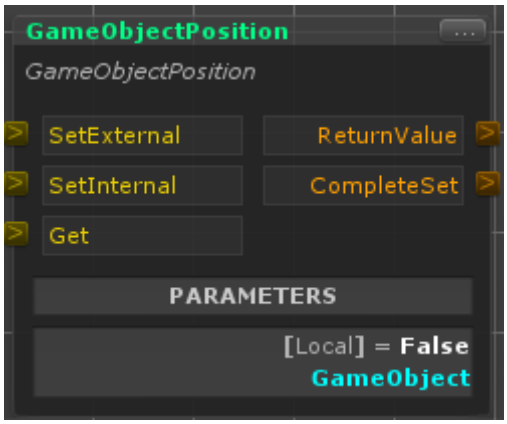
UnityObject – a reference to a variable with a scene object (GameObject).

GameObjectPosition

GameObjectRotation

GameObjectScale

An element for setting and obtaining the position, rotation and scale of the object.



Input points:

SetExternal **[Vector3]** – Input point for setting a parameter from an external value.

SetInternal **[]** – Input point for setting a parameter from the internal value of the element.

Get **[]** – input point for retrieving the current object position/rotation/scale.

Output points:

ReturnValue **[Vector3]** – output point transmitting the object position/rotation/scale.

CompleteSet **[]** – output point that is called when the parameter is set.

Parameters:

HideFlag – a flag for configuring input and output points.

InternalValue – new position, rotation (in Euler) or scale of the object.

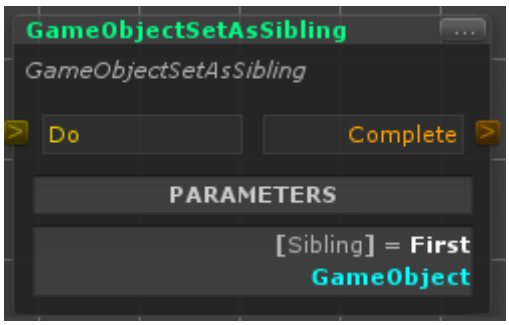
Local – flag to set or get the local value of the parameter.

References to diagram variables:

UnityObject – a reference to a variable with a scene object (GameObject).

GameObjectSetAsSibling

An element for setting the position of an object in the hierarchy.



Input points:

Do **[]** – input point for setting the position of an object in the hierarchy.

Output points:

Complete **[]** – output point that is called when the parameter is set.

Parameters:

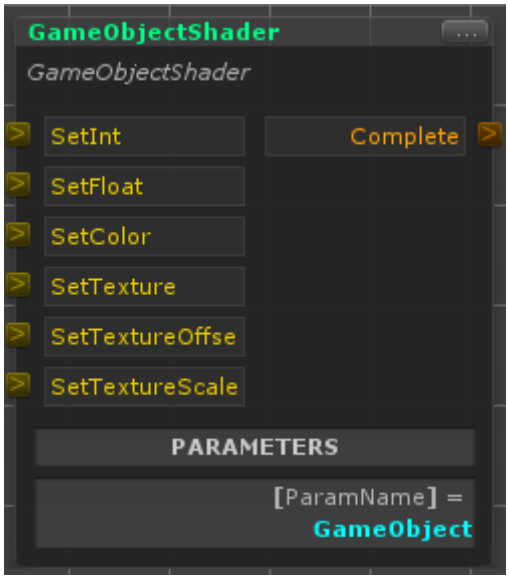
Sibling – the position in the hierarchy (the first, the last).

References to diagram variables:

UnityObject – a reference to a variable with a scene object (GameObject).

GameObjectShader

An element for setting the shader parameters of the scene object material.



Input points:

SetInt **[int]** – input point for setting an int value.

SetFloat **[float]** – input point for setting a float value.

SetColor **[Color]** – input point for setting a Color value.

SetTexture **[VSObject(Texture)]** – input point for setting a texture.

SetTextureOffset **[Vector2]** – input point for setting the offset of texture coordinates.

SetTextureScale **[Vector2]** – input point for setting the scale of texture coordinates.

Get **[]** – input point for obtaining the current value of the shader parameter

Output points:

Complete **[]** – output point that is called when the parameter is set.

ReturnIntValue **[int]** – output point that transmits the value of the shader parameter.

ReturnFloatValue **[float]** – output point that transmits the value of the shader parameter.

ReturnColor **[Color]** – output point that transmits the value of the shader parameter.

ReturnTexture **[VSObject(Texture)]** – output point that transmits the value of the shader parameter.

ReturnTextureOffset **[Vector2]** – output point that transmits the value of the shader parameter.

ReturnTextureScale **[Vector2]** – output point that transmits the value of the shader parameter.

Parameters:

HideFlag – a flag for configuring input and output points.

ParameterType – a parameter for configuring the input points.

ParamName – the name of parameter in the shader.

MaterialIndex – the index of the material whose shader parameters will be changed. In case of single material, leave value an 0.

References to diagram variables:

UnityObject – a reference to a variable with a scene object (GameObject).

GameObjectShadows

An element for controlling the state of the shadows of the scene object.

Input points:

SetReceive **[bool]** – input point for setting the state of receiving shadows by the object.

SetCast **[float]** – input point for setting the state of casting shadows by the object.

Output points:

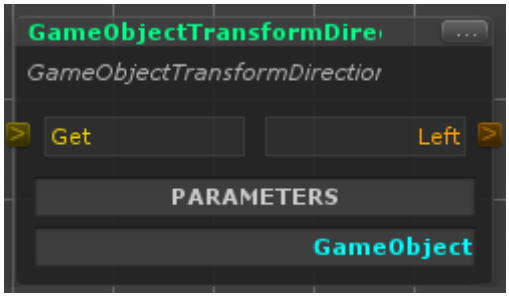
Complete **[]** – output point that is called when the parameter is set.

References to diagram variables:

UnityObject – a reference to a variable with a scene object (GameObject).

GameObjectTransformDirection

An element for obtaining vectors of local directions of the coordinate axes of the object.



Input points:

Get [] - input point for obtaining the direction vector of the object local coordinates.

Output points:

Automatically configured depending on the parameters of the element (the normalized Vector3 value of the direction is transmitted)

Parameters:

DirectionType – direction of coordinate axes.

References to diagram variables:

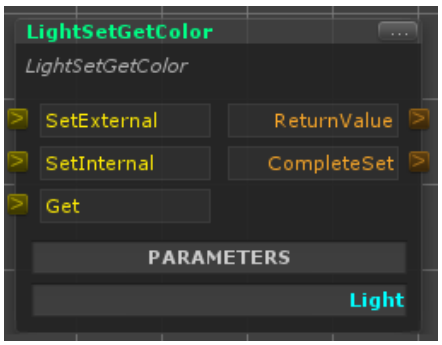
UnityObject – a reference to a variable with a scene object (GameObject).

Light

A section with elements that implement the logic of working with light sources.

LightSetGetColor

An element for setting the color of the light source.



Input points:

SetExternal **[Color]** – Input point for setting a parameter from an external value.

SetInternal **[]** – Input point for setting a parameter from the internal value of the element.

Get **[]** – input point to get the current value of the parameter.

Output points:

ReturnValue **[Color]** – output point transmitting the current parameter value.

CompleteSet **[]** – output point that is called when the parameter is set.

Parameters:

HideFlag – a flag for configuring input and output points.

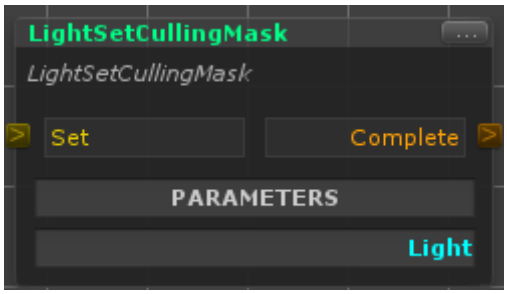
InternalValue – a value of parameter to set.

References to diagram variables:

UnityObject – a reference to a variable with the Light component.

LightSetCullingMask

An element for setting the layers mask for the light source.



Input points:

Set **[]** – input point for setting the layers illumination mask from the parameters of the element.

Output points:

Complete **[]** – output point that is called when the parameter is set.

Parameters:

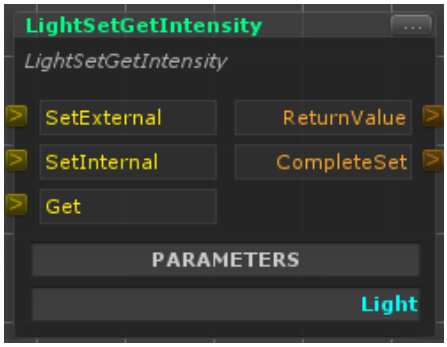
CullingMask – layers illumination mask.

References to diagram variables:

UnityObject – a reference to a variable with the Light component.

LightSetGetIntensity

An element for setting the intensity of the light source.



Input points:

SetExternal **[float]** – Input point for setting a parameter from an external value.

SetInternal **[]** – Input point for setting a parameter from the internal value of the element.

Get **[]** – input point to get the current value of the parameter.

Output points:

ReturnValue **[float]** – output point transmitting the current parameter value.

CompleteSet **[]** – output point that is called when the parameter is set.

Parameters:

HideFlag – a flag for configuring input and output points.

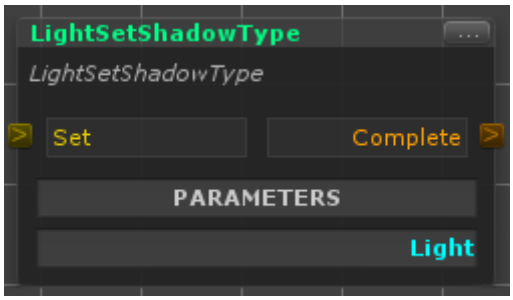
InternalValue – a value of parameter to set.

References to diagram variables:

UnityObject – a reference to a variable with the Light component.

LightSetShadowType

An element for changing the type of shadows that objects are casting when illuminated by a light source.



Input points:

Set **[]** – input point for setting the type of shadows from the parameters of the element.

Output points:

Complete **[]** – output point that is called when the parameter is set.

Parameters:

Shadows – shadows type.

References to diagram variables:

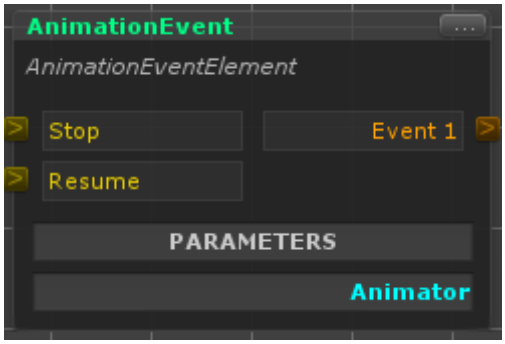
UnityObject – a reference to a variable with the Light component.

Mecanim

A section describing the elements that implement the logic of working with the Unity animation subsystem.

AnimationEvent

An element for subscribing to the event the occurrence of the specified frame in the animation.



Input points:

Stop **[]** – input point for stopping event processing.

Resume **[]** – input point for resuming event processing.

Output points:

Automatically configured according to the set parameters of the element, describing the necessary events.

Parameters:

SubscribeAtStart – a flag for enabling event processing at application startup.

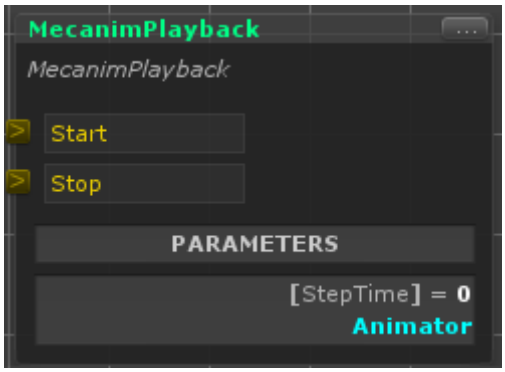
Events – a set of events for processing with a description. This field automatically takes all the animations attached to the object. Each event is described by the name and frame number when it needs to be called. For this set, input points are configured whose names correspond to the names of events.

References to diagram variables:

UnityObject – a reference to a variable with the Animator component.

MecanimPlayback

An element for playing previously recorded sequence of animation frames.



Input points:

Start **[]** – input point to start playback of the recorded sequence.

Stop **[]** – input point to stop the playback.

Parameters:

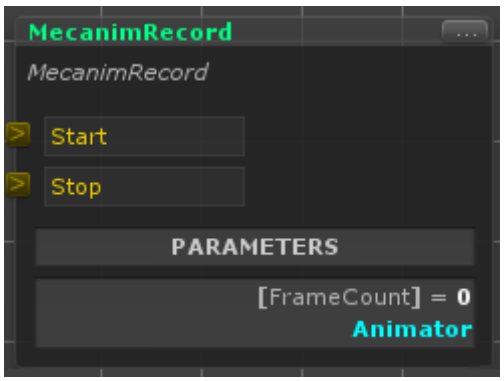
StepTime – step in seconds with which the animation will be played.

References to diagram variables:

UnityObject – a reference to a variable with the Animator component.

MecanimRecord

An element for recording the sequence of animation frames, for the purpose of later playback.



Input points:

Start **[]** – input point for starting recording.

Stop **[]** – input point for stopping recording.

Parameters:

FrameCount – the recording buffer size in frames (should be greater than 0).

References to diagram variables:

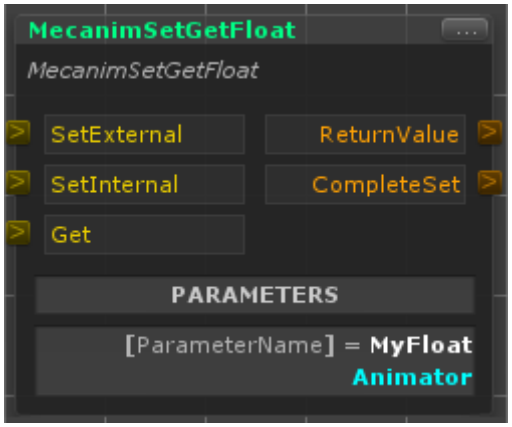
UnityObject – a reference to a variable with the Animator component.

MecanimSetGetBool

MecanimSetGetFloat

MecanimSetGetInt

Elements for setting and getting values of the parameters of the state machine (AnimatorController) of types bool, float and int.



Input points:

SetExternal **[bool, float, int]** – Input point for setting a parameter from an external value.

SetInternal **[]** – Input point for setting a parameter from the internal value of the element.

Get **[]** – input point to get the current value of the parameter.

Output points:

ReturnValue **[bool, float, int]** – output point transmitting the current parameter value.

CompleteSet **[]** – output point that is called when the parameter is set.

Parameters:

HideFlag – a flag for configuring input and output points.

InternalValue – a value of parameter to set.

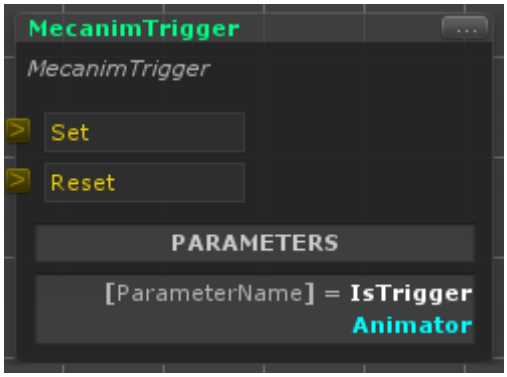
MecanimParameter – selection of the parameter to set (automatically generated list for the parameter type)

References to diagram variables:

UnityObject – a reference to a variable with the Animator component.

MecanimTrigger

An element for controlling the state machine trigger (AnimatorController)



Input points:

Set **[]** – input point for setting the trigger.

Reset **[]** – input point for resetting the trigger.

Parameters:

MecanimParameter – a list of the trigger type parameters presenting in the object animator.

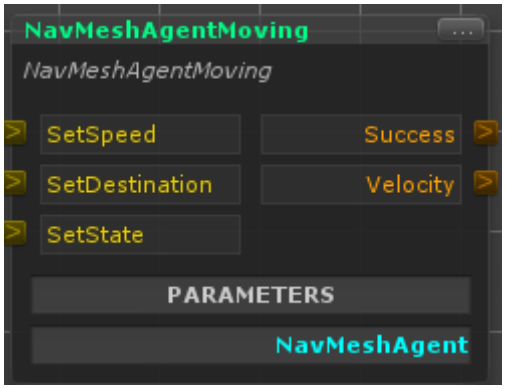
References to diagram variables:

UnityObject – a reference to a variable with the Animator component.

Navigation

NavMeshAgentMoving

An element for moving an object to a specified point with path finding in a previously calculated area based on the NavMesh subsystem.



Input points:

SetSpeed **[float]** – input point for setting the speed of the object.

SetDestination **[Vector3]** – input point for setting the target point and start moving to it.

SetState **[]** – input point for setting the state of the object motion (forced stop and resumption of movement to a specified point).

Output points:

Success **[bool]** – output point that transmits the state of the path finding to a given point (whether the path is found).

Velocity **[Vector3]** – output point transmitting the current velocity vector of the object.

Parameters:

DefaultSpeed – default speed value. This value is used if, at the time the movement starts, the object speed is set to 0 value.

References to diagram variables:

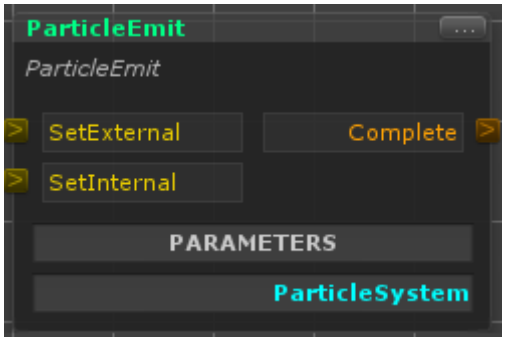
UnityObject – a reference to a variable with the NavMeshAgent component.

Particle System

A section with elements that implement the logic of working with particle systems.

ParticleEmit

An element for the forced emission of particles from a particle system.



Input points:

SetExternal **[int]** – input point for particle emission from external data.

SetInternal **[]** – input point for the emission of particles from the internal value of the element.

Output points:

Complete **[]** – output point that is called when the element work is finished.

Parameters:

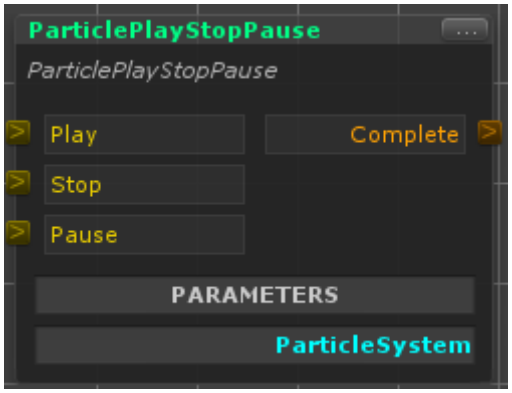
EmitCount – number of particles for emission.

References to diagram variables:

UnityObject – a reference to a variable with the ParticleSystem component.

ParticlePlayStopPause

An element for controlling the particle system.



Input points:

Play **[]** – input point for starting a particle system.

Stop **[]** – input point for stopping the particle system.

Pause **[]** – input point for pausing the particle system.

Output points:

Complete **[]** – output point that is called when the element work is finished.

Parameters:

WithChildren – a flag indicating that the management of the particle system involves the management of all its descendants in the hierarchy.

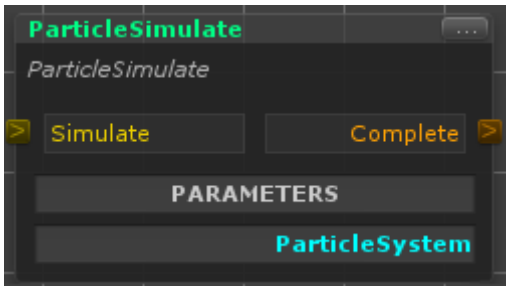
StopBehaviour – a way to stop the particle system (only if the Stop input point is used).

References to diagram variables:

UnityObject – a reference to a variable with the ParticleSystem component.

ParticleSimulate

An element for transferring the state of the particle system to a given time position.



Input points:

Simulate **[float]** – input point for transmitting the time position and transferring the state of the particle system to it.

Output points:

Complete **[]** – output point that is called when the element work is finished.

Parameters:

WithChildren – a flag indicating that the transfer of the state of the particle system includes all its descendants in the hierarchy.

Restart – a flag indicating that the time count starts from the beginning state of the particle system, otherwise from the current one.

FixedTimeStep – a flag indicating that when calculating the particle system state, a fixed time step should be used based on the fixedDeltaTime value.

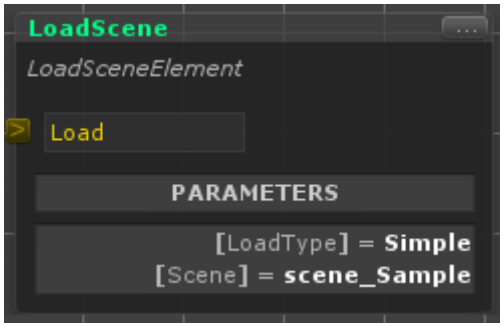
References to diagram variables:

UnityObject – a reference to a variable with the ParticleSystem component.

Scene

LoadScene

An element for loading a scene.



Input points:

Load **[]** – input point for loading the scene according to the set parameters of the element.

Parameters:

LoadType – the method of the scene loading.

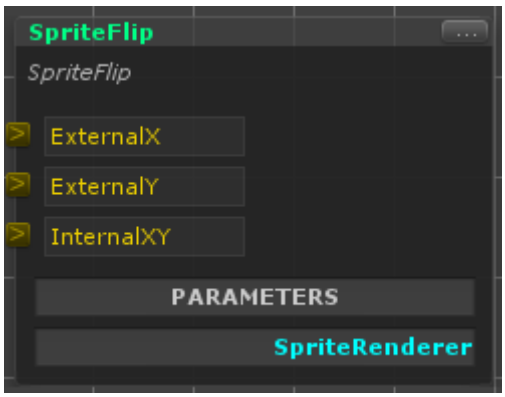
Scene – select the scene to load. This list is automatically generated based on the settings of the application build.

Sprite

A section with a description of the elements that implement the logic of working with sprites.

SpriteFlip

An element for inversion of the image of the sprite.



Input points:

ExternalX **[bool]** – input point for setting the inversion along the X axis from an external value.

ExternalY **[bool]** – input point for setting the inversion along the Y axis from an external value.

InternalXY **[bool]** – input point for setting the inversion along the axes from the parameters of the element.

Output points:

Complete **[]** – output point that is called when the element work is finished.

Parameters:

FlipX – X-axis inversion flag.

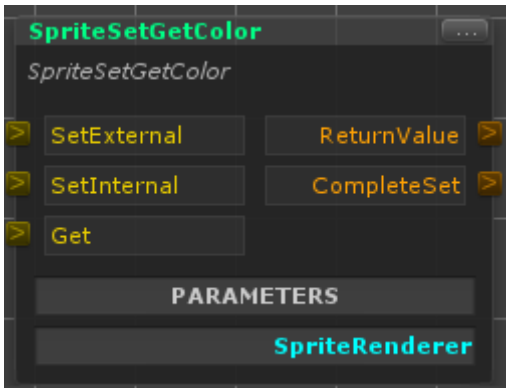
FlipY – Y-axis inversion flag.

References to diagram variables:

UnityObject – a reference to a variable with the SpriteRenderer component.

SpriteSetGetColor

An element for setting and retrieving the color of the sprite.



Input points:

SetExternal **[Color]** – Input point for setting a parameter from an external value.

SetInternal **[]** – Input point for setting a parameter from the internal value of the element.

Get **[]** – input point to get the current value of the parameter.

Output points:

ReturnValue **[Color]** – output point transmitting the current value of the sprite color.

CompleteSet **[]** – output point that is called when the parameter is set.

Parameters:

HideFlag – a flag for configuring input and output points.

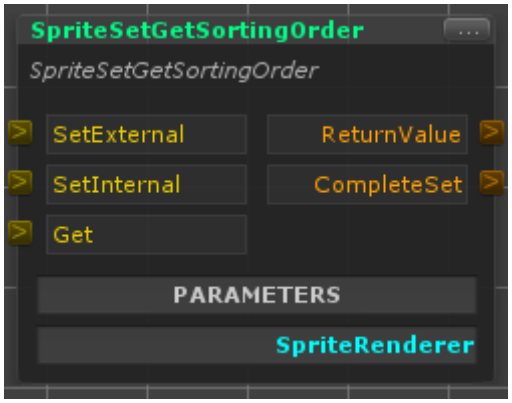
InternalValue – a value of parameter to set.

References to diagram variables:

UnityObject – a reference to a variable with the SpriteRenderer component.

SpriteSetGetSortingOrder

An element for setting the order of the sprite rendering.



Input points:

SetExternal **[int]** – Input point for setting a parameter from an external value.

SetInternal **[]** – Input point for setting a parameter from the internal value of the element.

Get **[]** – input point to get the current value of the parameter.

Output points:

ReturnValue **[int]** – output point transmitting the current value of the sprite rendering order number.

CompleteSet **[]** – output point that is called when the parameter is set.

Parameters:

HideFlag – a flag for configuring input and output points.

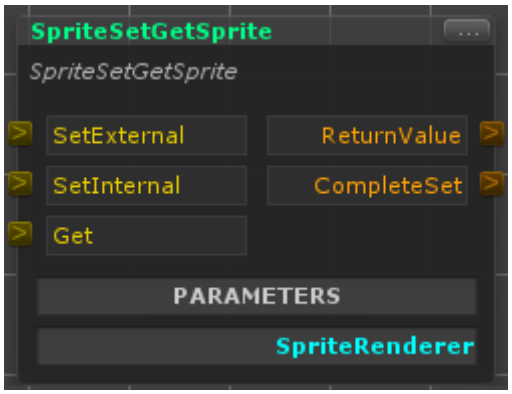
InternalValue – a value of parameter to set.

References to diagram variables:

UnityObject – a reference to a variable with the SpriteRenderer component.

SpriteSetGetSprite

Element for setting and getting a reference to the image of the sprite.



Input points:

SetExternal **[VSObject(Sprite)]** – Input point for setting a parameter from an external value.

SetInternal **[]** – Input point for setting a parameter from the internal value of the element.

Get **[]** – input point to get the current value of the parameter.

Output points:

ReturnValue **[VSObject(Sprite)]** – output point transmitting the current value of sprite rendering order number.

CompleteSet **[]** – output point that is called when the parameter is set.

Parameters:

HideFlag – a flag for configuring input and output points.

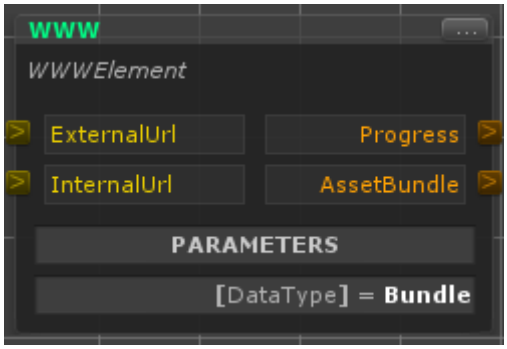
InternalValue – a reference to the sprite image (Sprite component).

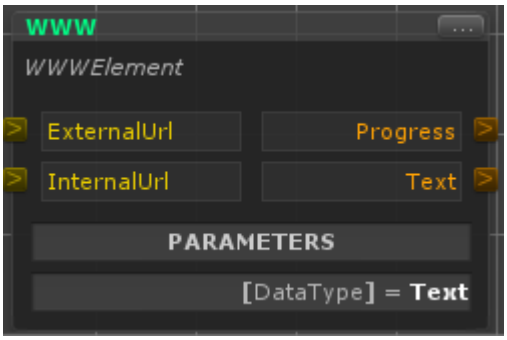
References to diagram variables:

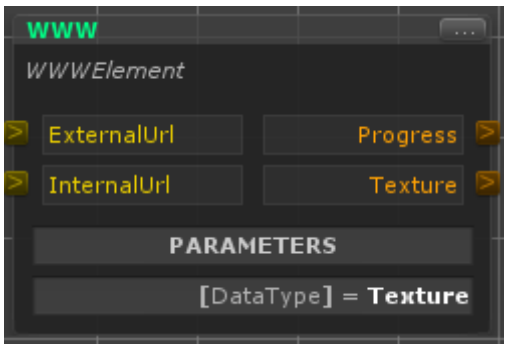
UnityObject – a reference to a variable with the SpriteRenderer component.

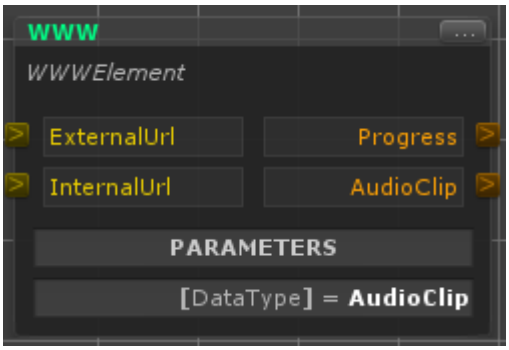
WWW

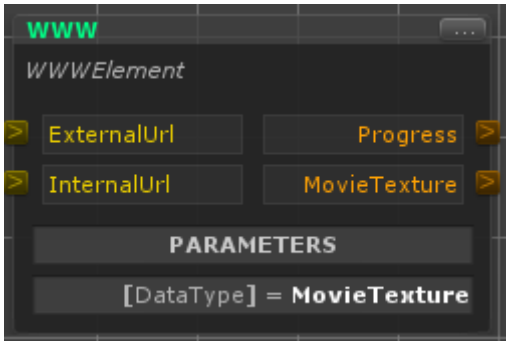
An element for downloading data by reference using the Unity3d WWW method.











Input points:

ExternalUrl **[string]** – input point for loading data by reference from an external value.

InternalUrl **[]** – input point for loading data by reference from element parameters.

Output points:

Progress **[float]** – output point transmitting the current value of the progress in normalized form.

AssetBundle **[AssetBundle]** – output point transmitting the downloaded data as a reference to AssetBundle.

Text **[string]** – output point that transmits downloaded data of string type.

Texture **[VSObject(Texture)]** – output point that transmits downloaded data of texture type.

AudioClip **[VSObject(AudioClip)]** – output point that transmits downloaded data of audioclip type.

MovieTexture **[VSObject(Texture)]** – output point that transmits downloaded data of MovieTexture type as a reference to a Texture.

Parameters:

DataType – setting the type of data to be downloaded. Output points are automatically configured by this parameter.

Url – download link.

FromCashe – a flag for loading data from the cache (if the cache is empty, the link is downloaded).

Version – version for download, used to compare the data in the cache and by reference.

Crc – checksum to check when downloading from the cache.

Utils

Array

A section with a description of the elements that implement the logic of working with members of arrays.

Items

ArrayIntItems

ArrayFloatItems

ArrayStringItems

ArrayColorItems

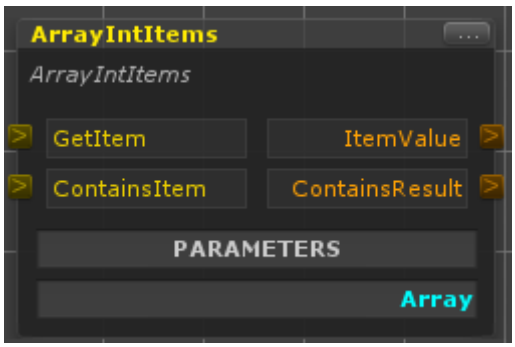
ArrayVector2Items

ArrayVector3Items

ArrayVector4Items

ArrayVSObjectItems

Elements for retrieving a member of an array and checking the value existence among its members.



Input points:

GetItme **[int]** – input point to get the value by index.

ContainsItem **[int, float, string, Color, Vector2/3/4, VSObject]** – input point for checking the value existence in the array.

Output points:

ItemValue **[int, float, string, Color, Vector2/3/4, VSObject]** – output point that transmits the value of the array member according to the index obtained through the GetItem.

ContainsResult **[int]** – output point that transmits the result (result is index of item, if -1, then item not found) of checking the value existence in the array.

References to diagram variables:

UnityObject – a reference to a variable with an array.

Sampling

ArrayIntSampling

ArrayFloatSampling

ArrayStringSampling

ArrayColorSampling

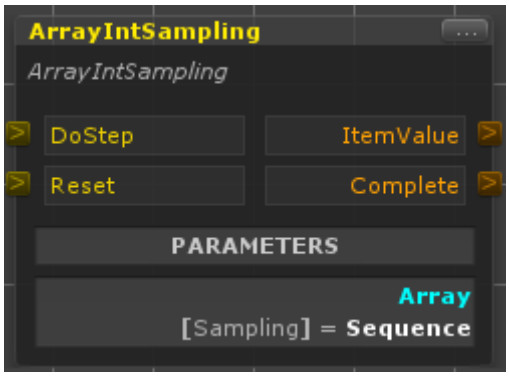
ArrayVector2Sampling

ArrayVector3Sampling

ArrayVector4Sampling

ArrayVSObjectSampling

Elements for retrieving values from an array.



Input points:

DoStep **[]** – input point for executing the sampling step.

Reset **[]** – input point for resetting the current sampling state to the initial state (this point is switched on if the sampling mode is not random and the loop flag is not set).

Output points:

ItemValue **[int, float, string, Color, Vector2/3/4, VSObject]** – output point that transmits the value of the selected array member.

Complete **[]** – output point, called after a full iterating through all array members (this point is switched on if the loop flag is not set).

Parameters:

SamplingType – setting the method for retrieving array members.

Loop – a flag of loop selection, indicates that after iterating through all the members of the array, the process starts anew at the next step.

References to diagram variables:

UnityObject – a reference to a variable with an array.

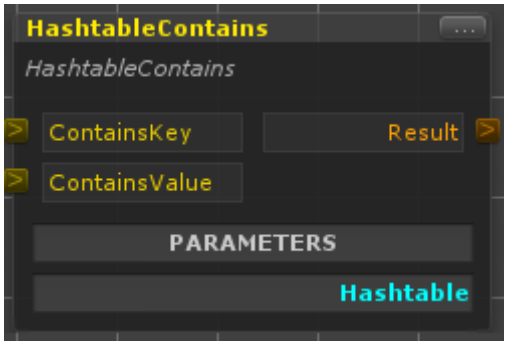
Dynamic

Hashtable

A section describing the elements that implement the hash table.

HashtableContains

An element for checking the existence of data in keys or values of hash table.



Input points:

ContainsKey **[object]** – input point for checking the existence of a key.

ContainsValue **[]** – input point for checking the existence of a value.

Output points:

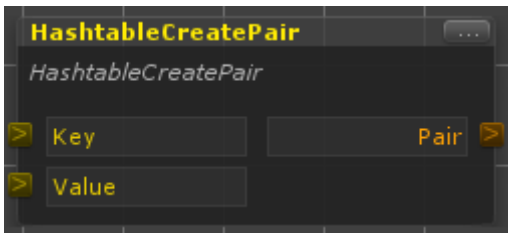
Result **[bool]** – output point that transmits the result of the test.

References to diagram variables:

UnityObject – a reference to a variable with a hash table.

HashtableCreatePair

An element for creating a key-value pair, for subsequent setting in a hash table.



Input points:

Key **[object]** – input point for setting a key.

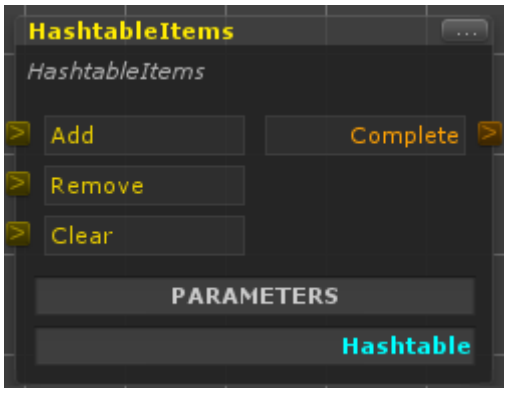
Value **[object]** – input point for setting a value.

Output points:

Pair **[HashtablePair]** – output point transmitting the reference to the created pair. The pair is created when both the key and the value are set, the order of setting is not important.

HashtableItems

An element for managing hash table data.



Input points:

Add **[HashtablePair]** – input point for adding data to the table (key-value pair is transmitted).

Remove **[object]** – input point for deleting data from the table (the key is transmitted).

Clear **[]** – input point for deleting all data from the table.

Output points:

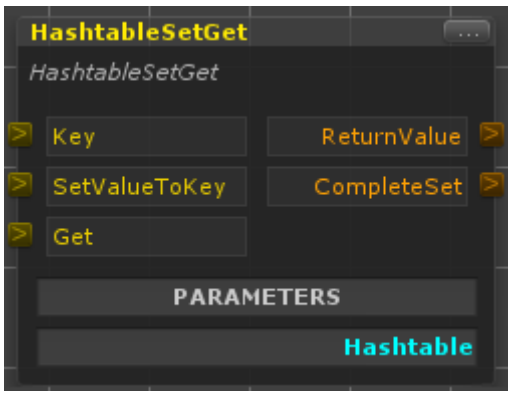
Complete **[]** – output point that is called when the element work is finished.

References to diagram variables:

UnityObject – a reference to a variable with a hash table.

HashtableSetGet

An element for setting and retrieving data from a hash table.



Input points:

Key **[object]** – input point for setting a key.

SetValueToKey **[object]** – input point for setting a value for the key.

Get **[object]** – input point for getting the value for the key.

Output points:

ReturnValue **[object]** – output point that transmits the value by the key.

CompleteSet **[]** – output point that is called when the value is set.

Parameters:

HideFlag – a flag for configuring input and output points.

References to diagram variables:

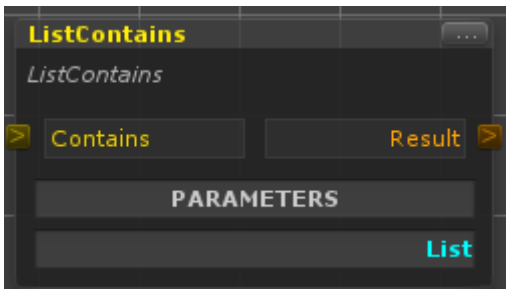
UnityObject – a reference to a variable with a hash table.

List

A section with a description of the elements that implement the working with a list.

ListContains

An element for checking the existence of the specified data in a list.



Input points:

Contains **[object]** – input point for checking the existence of the data in the list.

Output points:

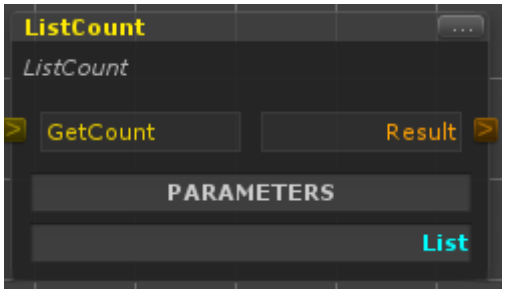
Result **[bool]** – output point that transmits the result of checking.

References to diagram variables:

UnityObject – a reference to a variable with a list.

ListCount

An element to get the number of members of the list.



Input points:

GetCount **[]** – input point for obtaining the value of the number of members of the list.

Output points:

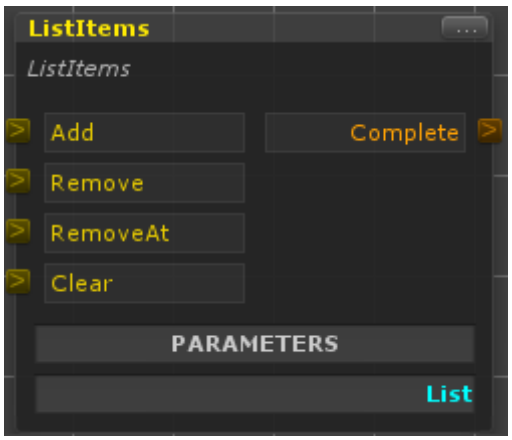
Result **[int]** – output point that transmits the number of members of the list.

References to diagram variables:

UnityObject – a reference to a variable with a list.

ListItems

An element for managing the list data.



Input points:

Add **[object]** – input point for adding data to the list.

Remove **[object]** – input point for deleting data from the list by value.

RemoveAt **[int]** – input point for deleting data from the list by index.

Clear **[]** – input point for deleting all data from the list.

Output points:

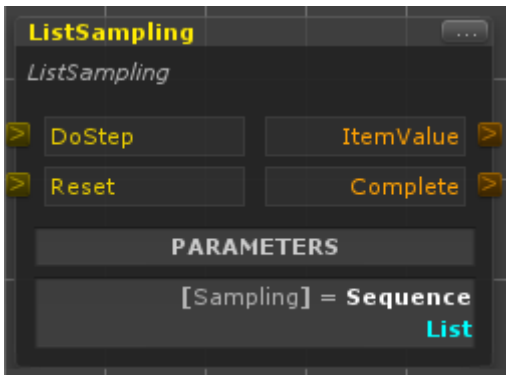
Complete **[]** – output point that is called when the element work is finished.

References to diagram variables:

UnityObject – a reference to a variable with a list.

ListSampling

An element for sampling data from the list.



Input points:

DoStep **[]** – input point for executing the sampling step.

Reset **[]** – input point for resetting the current sample state to the initial state (this point is switched on if the sampling mode is not random and the loop flag is not set).

Output points:

ItemValue **[object]** – output point that transmits the value of the selected list member.

Complete **[]** – output point, called after a full iterating through all array members (this point is switched on if the loop flag is not set).

Parameters:

SamplingType – setting the method for retrieving array members.

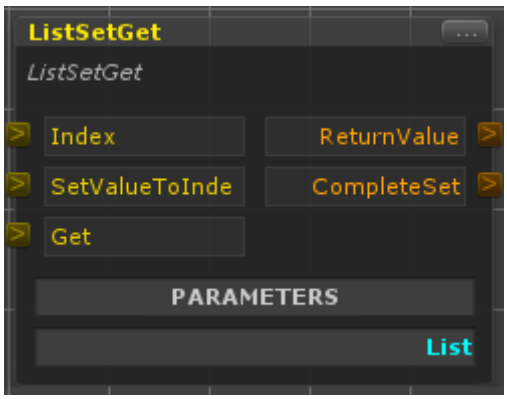
Loop – a flag of loop selection, indicates that after iterating through all the members of the array, the process starts anew at the next step.

References to diagram variables:

UnityObject – a reference to a variable with a list.

ListSetGet

An element for setting and retrieving data from the list.



Input points:

Index **[int]** – input point for index transmission.

SetValueToIndex **[object]** – input point for setting the value by index.

Get **[int]** – input point for getting the data by index.

Output points:

ReturnValue **[object]** – output point that transmits the value by key.

CompleteSet **[]** – output point that is called when the value is set.

Parameters:

HideFlag – a flag for configuring input and output points.

References to diagram variables:

UnityObject – a reference to a variable with a list.

Link

A section with a description of the elements that implement the logic of additional connection of elements in the diagram.

PassThrough

An element for checking the correspondence of input states to a condition and branching based on the results of this test.

Input points:

Value1..N **[bool]** – input point for transmitting of state.

Output points:

ConditionDone **[]** – output point, called if the input value satisfies the specified condition.

ConditionFail **[]** – output point that is called if the input values do not satisfy the specified condition.

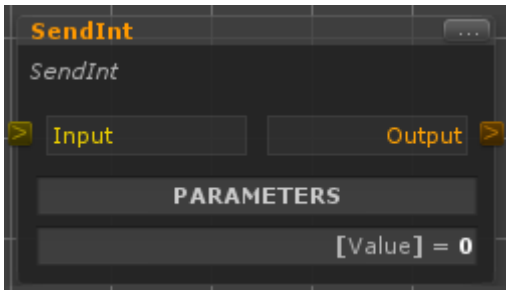
Parameters:

NumberOfInputPoints – setting the number of input values. This parameter automatically configures the input points.

Condition – type of condition for verification.

SendInt
SendFloat
SendString
SendBool
SendColor
SendVector2
SendVector3
SendVector4
SendVSObject

Elements for transmitting preset data of various types in the diagram call chain.



Input points:

Input **[]** – input point that causes data to be transmitted to the output point.

Output points:

Output **[bool, int, float, string, Color, Vector2/3/4, VSObject]** – output point that transmits the data set in the parameters.

Parameters:

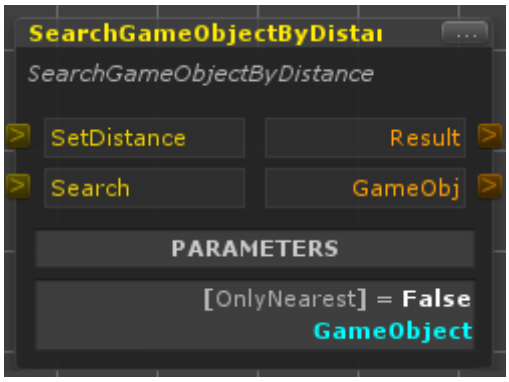
Value – value for transmission to the output point.

Search

A section with a description of the elements that implement the logic of searching for scene objects.

SearchGameObjectByDistance

An element for searching objects of the scene by the distance from the target object.



Input points:

SetDistance **[float]** – setting the minimum distance, from which the search will be performed (i.e. objects located at a distance no less than the established one will be found).

Search **[]** – input point for starting the search.

Output points:

Result **[bool]** – output point that transmits the search result.

GameObj **[VSObject]** – output point that transmits references to found scene objects. If several objects are found, the output point will be called in a loop for each object.

Parameters:

SearchInLayer – layer mask for searching for objects.

SearchByTag – a tag for searching for objects.

OnlyNearest – a flag for searching for the nearest object that satisfies the condition.

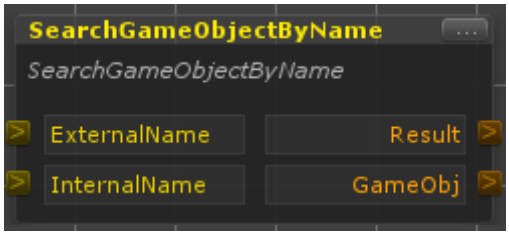
DefaultDistance – default search distance value. If the value was set through the input point, then it will be used further.

References to diagram variables:

UnityObject – a reference to a variable with the scene object related to which the search is conducted.

SearchGameObjectByName

An element for searching objects by name.



Input points:

ExternalName **[string]** – input point for searching by name set from the outside.

InternalName **[]** – input point for searching by name set from the parameters of the element.

Output points:

Result **[bool]** – output point that transmits the search result.

GameObj **[VSObject]** – output point transmitting a reference to the found scene object.

Parameters:

Name – the name of the object for search.

Примечание: this element uses the method GameObject.Find, therefore, if there are several objects with the same name, it always returns only one of them (the first one satisfying the name matching condition).

SearchGameObjectByTag

An element for searching objects by a tag.

Input points:

ExternalTag **[string]** – input point for searching by tag set from the outside.

InternalTag **[]** – input point for searching by tag set from the parameters of the element.

Output points:

Result **[bool]** – output point that transmits the search result.

GameObj **[VSObject]** – output point that transmits references to found scene objects. If several objects are found, the output point will be called in a loop for each object.

Parameters:

Tag – the tag of the object for search.

AllMatches - a flag for searching for all objects that meet the condition.

Typecast

A section describing the elements that implement the logic of converting data from one format to another, as well as converting data types.

IntToFloat

FloatToInt

Elements for converting between integer and fractional number.

Input points:

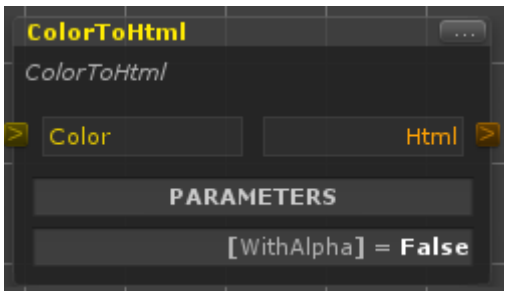
Value **[int, float]** – input point for transmitting data for the conversion.

Output points:

ReturnValue **[float, int]** – output point that transmits the result of conversion.

ColorToHtml

An element for conversion a color to the string html format.



Input points:

Color **[Color]** – input point for the color to be converted.

Output points:

Html **[string]** – output point that transmits the result of color conversion to the html format.

Parameters:

WithAlpha – a flag for accounting the alpha channel when converting value of color.

EulerToQuaternion

An element for converting the rotation from the Euler angles to a quaternion.

Input points:

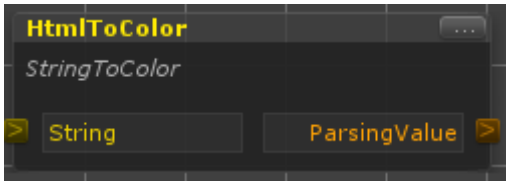
Euler **[Vector3]** – input point for the rotation in the Euler angles.

Output points:

Quater **[Quaternion]** – output point that transmits the result of conversion to a quaternion.

HtmlToColor

An element of color conversion from html format to Color.



Input points:

Html **[string]** – input point for the color to be converted.

Output points:

ParsingValue **[Color]** – output point that transmits the result of the color conversion.

ObjectTo

An element for casting an object to a given type.

Input points:

Value **[object]** – input point for transmitting data for the conversion.

Output points:

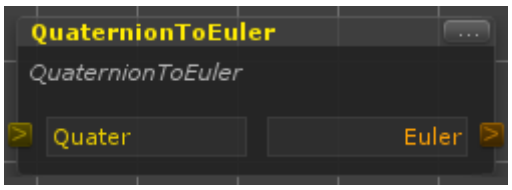
Automatically configured for the set type for casting.

Parameters:

TypeCast – type to which input data should be casted.

QuaternionToEuler

An element for converting a rotation from a quaternion to Euler angles.



Input points:

Quater **[Quaternion]** – input point for the transmission of rotation in the form of a quaternion.

Output points:

Euler **[Vector3]** – output point transmitting the conversion result to the Euler angles.

StringToInt

StringToFloat

Elements for converting a string to a number (integer and fractional).

Input points:

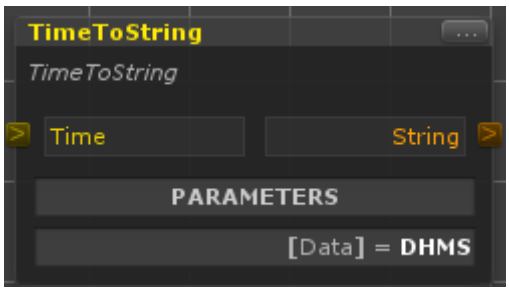
String **[string]** – input point for transferring a string for conversion.

Output points:

ParsingValue **[int, float]** – output point transmitting the result of the conversion.

TimeToString

An element for conversion the time in seconds to a formatted string.



Input points:

Time **[float]** – input point for the time in seconds.

Output points:

TimeStr **[string]** – output point transmitting the time as a formatted string.

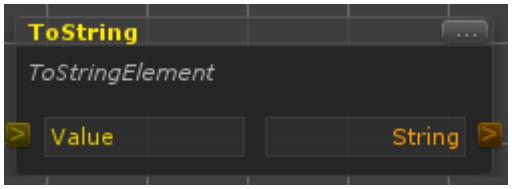
Parameters:

Data – time format (days, hours, minutes, seconds).

FormattedString – the format of the time string (formatting based on the C# standard). Example: {0}; {1}; {2}, here 0, 1, 2 are indices for substituting values (in this case, hours, minutes of a second).

ToString

An element for converting data to a string.



Input points:

Value **[object]** – input point for transmitting data.

Output points:

String **[string]** – output point that transmits the result of the conversion to a string.

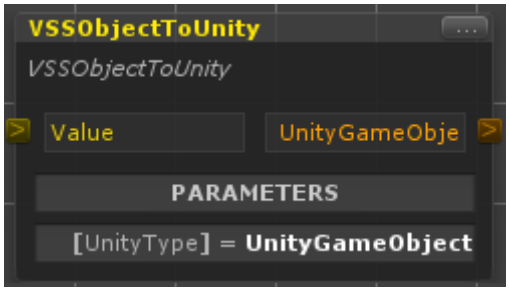
Parameters:

Format – the string for output in formatted view

Note: all data that support the ToString () method is converted.

VSObjectToUnity

An element for getting a reference to Unity components from the inner VSObject wrapper.



Input points:

Value **[VSObject]** – input point for transmitting data.

Output points:

Automatically configured for a given data type.

Parameters:

UnityType – the type of the Unity component to retrieve from the VSObject.

Variable

A section with a description of the elements that implement the logic of working with the variables of the diagram.

SetGet

SetGetVariableInt

SetGetVariableFloat

SetGetVariableBool

SetGetVariableString

SetGetVariableColor

SetGetVariableVector2

SetGetVariableVector3

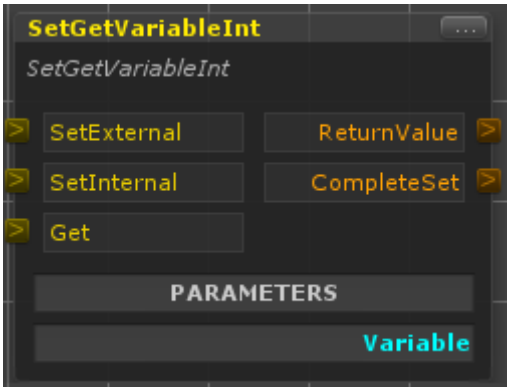
SetGetVariableVector4

SetGetVariableVSObject

SetGetVariableHashtable

SetGetVariableList

Elements for setting and retrieving the value of a diagram variable.



Input points:

SetExternal **[int, float, bool, string, Color, Vector2/3/4, VSObject, Hashtable, List<object>]** – input point for setting the value of a variable from an external value.

SetInternal **[]** – input point for setting the value of a variable from the element parameters.

Get **[]** – input point to get the current value of the parameter.

Output points:

ReturnValue **[int, float, bool, string, Color, Vector2/3/4, VSObject, Hashtable, List<object>]** – output point that transmits the value of the diagram variable.

CompleteSet **[]** – output point that is called when the parameter is set.

Parameters:

HideFlag – a flag for configuring input and output points.

InternalValue – a value to set to a variable.

References to diagram variables:

Variable – a reference to the variable of the diagram.

Subscriber

SubscriberVariableInt

SubscriberVariableFloat

SubscriberVariableBool

SubscriberVariableString

SubscriberVariableColor

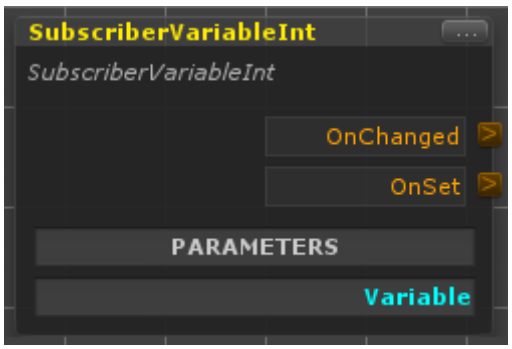
SubscriberVariableVector2

SubscriberVariableVector3

SubscriberVariableVector4

SubscriberVariableVSObject

Elements for handling the events of changing and setting the value of a variable.



Output points:

OnChanged **[]** – output point that is called if the new value of the variable differs from the old one.

OnSet **[int, float, bool, string, Color, Vector2/3/4, VSObject]** – output point that is called when the value is set to a variable (with the transmit of this value).

References to diagram variables:

Variable – a reference to the variable of the diagram.

Variable

Array

Variables that define data arrays of different types. The variable arrays within the Panthea VS system are fixed datasets, so there no elements responsible for deleting and adding members of the array. For this purpose, dynamic repositories of the list type and hash table are used.

Color[]

Variable for an array of colors.

Float[]

Variable for an array of floats.

Int[]

Variable for an array of integer numbers.

String[]

Variable for an array of strings.

Vector2[]

Variable for an array of 2D vectors.

Vector3[]

Variable for an array of 3D vectors.

Vector4[]

Variable for an array of 4D vectors.

VSObject[]

Variable for an array of references to objects, components, prefabs.

Base

Variables of base types.

AnimationCurve

Variable for the animation curve.

Bool

Variable for a boolean value.

Color

Variable for a color.

Float

Variable for a floating-point number.

Int

Variable for a integer number.

String

Variable for a string.

Vector2

Variable for 2D vector.

Vector3

Variable for 3D vector.

Vector4

Variable for 4D vector.

VSObject

Variable for the reference to object, component, prefab.

Dynamic

Variable for the creation of dynamic data stores, which are filled and used in the execution of the application. Storage data cannot be preset, but can be initialized at startup.

Hashtable

Variable for a hash table. The store of key-value pairs. Both the key and the value can be of any arbitrary type.

List

Variable for a list of packed (boxed) values.