Vanda Engine Scripting Reference Manual

User guide
2023 v7.0

Table of contents

1. Introduction

Vanda engine allows the user to perform actions during runtime using scripts that are attached to objects. Vanda Engine scripts are written in Lua language --for more information about Lua language, please visit https://www.lua.org/. Each script consists of one or more events and the corresponding code along with APIs are usually written inside the events. Each event is called at a certain time by Vanda engine. For example, the Init() event is called once during the initialization of the corresponding object to which the script is attached, and the code inside it is executed.
In this guide, we will review the script editor, events and scripting APIs with examples.

2. Script Editor

To write and edit scripts, you can use the built-in script editor of Vanda Engine or other IDEs. In this section, we will describe the script editor of Vanda engine.
To access the script editor, use the Tools > Script Editor menu. In this section, we explain the script editor menu.

File Menu

New
creates a new lua script.

Open
Opens a standard File dialog that lets you select an existing Lua file.

Save
Saves the current script. For the first time, it opens a dialog that lets you select the file path.

Save As
Opens a File dialog that lets you save the script under a new name.

Exit
Exits the script editor.

Edit Menu

Undo
Lets you undo recent changes.

Redo
Lets you redo recent changes.

Copy
Use this command to copy the current selection to the clipboard as text.

Paste
This command allows you to insert at the cursor position text contained on the clipboard.

Debug Menu

Debug Script
Using this menu, you can check the syntax errors of your code. If no errors are found, the message *No Errors Found* is displayed in the *Errors* section of the editor.

Tools Menu

Script Utility
Opens a new dialog that allows you to view the projects, project resources, GUIs, game levels and their objects and copy their names if necessary. These names are used in scripting APIs and can be used as their input parameters.

Add Event
Opens a new dialog that allows you to add scripting events to your script.

Add Function
Opens a new dialog that allows you to add scripting APIs to your script.

Help Menu
Opens Scripting Reference Manual.

3. Events

Scripting events in Vanda Engine are functions written in Lua language with specified and reserved names that are executed by Vanda Engine at certain times. Not all objects support all of the introduced events. For example, the camera object supports `Init()` and `Update()` events, while the trigger object supports the `OnTriggerEnter(otherActorName)`, `OnTriggerStay(otherActorName)`, and `OnTriggerExit(otherActorName)` events.
Events are written in the following general form:

```lua
function function_name(optional_parameter)

end
```

For example, the `Init()` event of Vanda Engine in Lua language would be written as follows:

```lua
function Init()

end
```

While the `OnTriggerEnter(otherActorName)` event of Vanda Engine in Lua language would be written as follows:

```lua
function OnTriggerEnter(otherActorName)

end
```

In the examples above, the `Init()` event accepts no arguments, while the `OnTriggerEnter(otherActorName)` event accepts an argument that is the name of the physics actor entered into the trigger --This name is automatically sent to the event by Vanda Engine.
You have to write your desired code inside the event function. For example, to display a text in the console when the `Init()` event is called, you can use the following code:

```lua
function Init()
    PrintConsole("\nSample message")
end
```

In this section, we explain the scripting events supported by Vanda Engine.

3.1. Init

Definition

```
function Init()

end
```

Description

Suppose a script that has an `Init()` event is attached to an object. In this case, the `Init()` event is called exactly once before the `Update()` event when the corresponding object is initialized.

Example

```
function Init()
    PrintConsole("\nInit() Event was called")
end
```

3.2. OnExit

Definition

```
function OnExit()

end
```

Description

This event is specific to the Video object. Suppose a script that has an `OnExit()` event is attached to a video object. In this case, the `OnExit()` event is called when the video ends or is stopped by the user by pressing a key.

Example

```
function OnExit()
    PrintConsole("\nOnExit() Event was called")
end
```

3.3. OnSelect

Definition

```
function OnSelect()

end
```

Description

This event is specific to the prefab instance object. Suppose a script that has an `OnSelect()` event is attached to a prefab object. In this case, the `OnSelect()` event is called when an instance of that prefab is selected at runtime by the *`SelectPrefabInstances`* function.

Example

```
function OnSelect()
    PrintConsole("\nOnSelect() Event was called")
end
```

3.4. OnSelectMouseEnter

Definition

```
function OnSelectMouseEnter()

end
```

Description

This event is specific to the button object. Suppose a script that has an `OnSelectMouseEnter()` event is attached to a button object. In this case, the `OnSelectMouseEnter()` event is called once when the mouse cursor enters that button.

Example

```
function OnSelectMouseEnter()
    PrintConsole("\nOnSelectMouseEnter() Event was called")
end
```

3.5. OnSelectMouseLButtonDown

Definition

```
function OnSelectMouseLButtonDown()

end
```

Description

This event is specific to the button object. Suppose a script that has an `OnSelectMouseLButtonDown()` event is attached to a button object. In this case, the `OnSelectMouseLButtonDown()` event is called once when the mouse cursor is on the button and the user left clicks.

Example

```
function OnSelectMouseEnter()
    PrintConsole("\nOnSelectMouseEnter() Event was called")
end
```

3.6. OnSelectMouseRButtonDown

Definition

```
function OnSelectMouseRButtonDown()

end
```

Description

This event is specific to the button object. Suppose a script that has an `OnSelectMouseRButtonDown()` event is attached to a button object. In this case, the `OnSelectMouseRButtonDown()` event is called once when the mouse cursor is on the button and the user right clicks.

Example

```
function OnSelectMouseRButtonDown()
    PrintConsole("\nOnSelectMouseRButtonDown() Event was called")
end
```

3.7. OnTriggerEnter

Definition

```
function OnTriggerEnter(otherActorName)

end
```

Description

This event is specific to the trigger object. Suppose a script that has an `OnTriggerEnter(otherActorName)` event is attached to a trigger object. In this case, the `OnTriggerEnter(otherActorName)` event is called once when the main character or a prefab instance that has dynamic physics enters the trigger.

Parameter

otherActorName

This parameter is automatically sent to `OnTriggerEnter` event by Vanda engine. If a prefab instance that has dynamic physics is entered into the trigger, the name of its physics actor is sent to the `OnTriggerEnter` event. If the main character of the game enters the trigger, the value `nil` is sent to the `OnTriggerEnter` event.

Example 1

```
function OnTriggerEnter(otherActorName)
    PrintConsole("\nOnTriggerEnter() Event was called")
end
```

Assume that this script is attached to a trigger called "trigger1". In this case, if the main character or a prefab instance that has dynamic physics is entered into "trigger1", the message `"OnTriggerEnter() Event was called"` will be displayed.

Example 2

```
function OnTriggerEnter(otherActorName)
    if otherActorName == nil then
        PrintConsole("\nMain character entered the trigger and OnTriggerEnter() Event
was called")
    else
        prefab_instance_name = GetPrefabInstanceNameFromActor(otherActorName)

        message = string.format("\nOnTriggerEnter() Event was called. Prefab instance
name is : %s" ,prefab_instance_name)
        PrintConsole(message)
    end
end
```

Assume that this script is attached to a trigger named "trigger1". In this case, if the main character is entered into "trigger1", the message `"Main character entered the trigger and OnTriggerEnter() Event was called"` will be displayed. Otherwise, if a prefab instance that has dynamic physics is entered into this trigger, the name of its physics actor is sent to the `OnTriggerEnter` event. Using the `GetPrefabInstanceNameFromActor` function, we find the prefab instance name that **`otherActorName`** name belongs to and display it in the console.

3.8. OnTriggerExit

Definition

```
function OnTriggerExit(otherActorName)

end
```

Description

This event is specific to the trigger object. Suppose a script that has an `OnTriggerExit(otherActorName)` event is attached to a trigger object. In this case, the `OnTriggerExit(otherActorName)` event is called once when the main character or a prefab instance that has dynamic physics exits the trigger.

Parameter

otherActorName

This parameter is automatically sent to `OnTriggerExit` event by Vanda engine. If a prefab instance that has dynamic physics exits the trigger, the name of its physics actor is sent to the `OnTriggerExit` event. If the main character of the game exits the trigger, the value `nil` is sent to the `OnTriggerExit` event.

Example 1

```
function OnTriggerExit(otherActorName)
    PrintConsole("\nOnTriggerExit() Event was called")
end
```

Assume that this script is attached to a trigger called "trigger1". In this case, if the main character or a prefab instance that has dynamic physics exits "trigger1", the message `"OnTriggerExit() Event was called"` will be displayed.

Example 2

```
function OnTriggerExit(otherActorName)
    if otherActorName == nil then
          PrintConsole("\nMain character is out of the trigger and OnTriggerExit() Event 
was called")
    else
          prefab_instance_name = GetPrefabInstanceNameFromActor(otherActorName)

          message = string.format("\nOnTriggerExit() Event was called. Prefab instance 
name is : %s" ,prefab_instance_name)
          PrintConsole(message)
    end
end
```

Assume that this script is attached to a trigger named "trigger1". In this case, if the main character exits "trigger1", the message `"Main character is out of the trigger and OnTriggerExit() Event was called"` will be displayed. Otherwise, if a prefab instance that has dynamic physics exits this trigger, the name of its physics actor is sent to the `OnTriggerExit` event. Using the `GetPrefabInstanceNameFromActor` function, we find the prefab instance name that **`otherActorName`** name belongs to and display it in the console.

3.9. OnTriggerStay

Definition

```
function OnTriggerStay(otherActorName)

end
```

Description

This event is specific to the trigger object. Suppose a script that has an `OnTriggerStay(otherActorName)` event is attached to a trigger object. In this case, the `OnTriggerStay(otherActorName)` event is called as long as the main character or a prefab instance that has dynamic physics is being placed inside the trigger. For example, if the main character is being placed in the trigger for 1 second and the frame rate is 30, this event will be called 30 times per second.

Parameter

otherActorName

This parameter is automatically sent to `OnTriggerStay` event by Vanda engine. If a prefab instance that has dynamic physics is being placed inside the trigger, the name of its physics actor is sent to the `OnTriggerStay` event. If the main character of the game is being placed inside the trigger, the value `nil` is sent to the `OnTriggerStay` event.

Example 1

```
function OnTriggerStay(otherActorName)
    PrintConsole("\nOnTriggerStay() Event was called")
end
```

Assume that this script is attached to a trigger called "trigger1". In this case, if the main character or a prefab instance that has dynamic physics is being placed inside "trigger1", the message `"nOnTriggerStay() Event was called"` will be displayed.

Example 2

```
function OnTriggerStay(otherActorName)
    if otherActorName == nil then
          PrintConsole("\nMain character is being placed inside the trigger and
OnTriggerStay() Event was called")
    else
          prefab_instance_name = GetPrefabInstanceNameFromActor(otherActorName)

          message = string.format("\nOnTriggerStay() Event was called. Prefab instance
name is : %s" ,prefab_instance_name)
          PrintConsole(message)
    end
end
```

Assume that this script is attached to a trigger named "trigger1". In this case, if the main character is being placed inside the "trigger1", the message `Main character is being placed inside the trigger and OnTriggerStay() Event was called"` will be displayed. Otherwise, if a prefab instance that has dynamic physics is being placed inside this trigger, the name of its physics actor is sent to the `OnTriggerStay` event. Using the

`GetPrefabInstanceNameFromActor` function, we find the prefab instance name that **`otherActorName`** name belongs to and display it in the console.

3.10. Update

Definition

```
function Update()

end
```

Description

Suppose a script that has an `Update()` event is attached to an object. In this case, the `Update()` event is called every frame. For example if the frame rate is 30, this event will be called 30 times per second.

Example

```
function Update()
    PrintConsole("\nUpdate() Event was called")
end
```

4. APIs

APIs in Vanda engine are functions that allow the user to perform certain tasks at runtime. You should use APIs inside scripting events. APIs are defined in the following general form:

```
return1, return2,..., returnN API_name(argunment1,
argument2,...,argumentN)
```

An API may take the parameters `argunment1`, `argunment2`,...,`argunmentN`, performs an action, and returns the values `return1`, `return2`,..., `returnN` if necessary. An API may take no input arguments and return no value. But in any case, it does something at runtime. Here are some examples:

Example 1

```
ActivateThirdPersonCamera()
```

This function takes no input arguments and returns no value, and only activates the third-person physics camera attached to the game's main character.

Example 2

```
CreateFolder(string folderPath)
```

This function takes a `string` argument and creates a folder in the path "Assets/Data/folderPath". This function does not return a value.

Example 3

```
bool IsWaterVisible(string waterName)
```

This function receives the name of the water as a `string` value and determines whether this water is visible or not. The result is returned as a Boolean value of true or false.

Example 4

```
double,double,double GetCharacterControllerPosition()
```

This function does not receive any input arguments and returns the X, Y and Z position of the character controller as three double values.

In this section, we explain the scripting functions available in Vanda engine.

4.1. ActivateEngineCamera

Definition

`ActivateEngineCamera(string engineCameraName, float endTime[optional])`

Description

Engine cameras are created in Vanda engine using the Insert > Camera menu. Engine cameras are not enabled by default. To activate these cameras, you must use the ActiateEngineCamera function.

Parameters

engineCameraName

Specifies the name of the engine camera. You can also use the name "this" for this parameter. In this case, "this" refers to the camera object that this script is attached to.

endTime

By default, `ActivateEngineCamera` function enables the camera engine indefinitely. This parameter allows you to activate the engine camera for endTime. After endTime, the third-person or first-person physics camera is activated. This parameter is optional and must be equal to or greater than 0.0.

Example 1

```lua
--Script name is thisEngineCamera.lua
function Init()
    ActivateEngineCamera("this", 5.0)
end

function Update()

end
```

In this case, `"this"` string in the `ActivateEngineCamera` points to the camera that `thisEngineCamera.lua` script is attached to. For example, if `thisEngineCamera.lua` script is connected to a engine camera named "camera1", `"this"` will be equivalent to the name "camera1". `ActivateEngineCamera` function activates the engine camera for `5.0` seconds, after which the first-person or third-person physics camera is activated.

Example 2

```lua
--Script name is camera1EngineCamera.lua
function Init()
    ActivateEngineCamera("camera1")
end

function Update()

end
```

In this case, the `ActivateEngineCamera` function activates engine `"camera1"` - if it exists - indefinitely.

4.2. ActivateFirstPersonCamera

Definition

```
ActivateFirstPersonCamera()
```

Description

This function activates the first-person physics camera attached to the main game character.

Example

```
function Init()
    ActivateFirstPersonCamera()
end

function Update()

end
```

4.3. ActivateImportedCamera

Definition

`ActivateImportedCamera(string importedCameraFullName, float endTime[optional])`

Description

Imported cameras are cameras that are imported to vanda engine through a 3D software in COLLADA format. You can view and copy the names of the imported cameras of the current VScene through the tools > Imported Camera menu. You can also access the imported camera names from the Script Utility section of the script editor (Tools > Script Editor > Tools > Script Utility).
These cameras are not enabled by default. This function allows you to activate the imported camera.

Parameters

`importedCameraFullName`
Specifies the full name of the imported camera as seen in the tools > Imported Camera menu.

`endTime`
By default, `ActivateImportedCamera` function enables the imported camera indefinitely. This parameter allows you to activate the imported camera for endTime. After endTime, the third-person or first-person physics camera is activated. This parameter is optional and must be equal to or greater than 0.0.

Example 1

```
function Init()
    ActivateImportedCamera("1_VandaEngine-Pack1_balcony-camera", 5.0)
end

function Update()

end
```

In this case, the `ActivateImportedCamera` function activates the imported camera `"1_VandaEngine-Pack1_balcony-camera"` - if it exists - for 5.0 seconds. After 5.0 seconds, the first person or third person physics camera will be activated.

Example 2

```
function Init()
    ActivateImportedCamera("1_VandaEngine-Pack1_balcony-camera")
end

function Update()

end
```

In this case, the `ActivateImportedCamera` function activates the imported camera `"1_VandaEngine-Pack1_balcony-camera"` - if it exists - indefinitely.

4.4. ActivateImportedCameraOfPrefabInstance

Definition

`ActivateImportedCameraOfPrefabInstance(string prefabInstanceName, string prefabCameraName, float endTime[optional])`

Description

Imported cameras are cameras that are imported to vanda engine through a 3D software in COLLADA format. To view the imported cameras of prefab instances, open the VScene and click on the desired Prefab Instance in the "Prefabs and GUIs" section and press the Edit button. You can also access the imported camera names of prefab instances from the Script Utility section of the script editor (Tools > Script Editor > Tools > Script Utility). In the dialog that appears, you can view and copy the name of the prefab instance and its imported camera - if any.
These cameras are not enabled by default. This function allows you to activate the imported camera of prefab instance.

Parameters

prefabInstanceName
Specifies the name of the prefab instance. You can also use the name "this" for this parameter. In this case, "this" refers to the prefab instance that this script is attached to.

prefabCameraName
Specifies the name of the prefab camera.

endTime
By default, `ActivateImportedCameraOfPrefabInstance` function activates the imported camera indefinitely. This parameter allows you to activate the imported camera for endTime. After endTime, the third-person or first-person physics camera is activated. This parameter is optional and must be equal to or greater than 0.0.

Example 1

```
function Init()
	ActivateImportedCameraOfPrefabInstance("1_VandaEngine17-SamplePack1_v3_house7",
	"Camera-camera", 5.0)
end

function Update()

end
```

In this case, the `ActivateImportedCameraOfPrefabInstance` function activates the imported camera `"Camera-camera"` of the prefab instance `"1_VandaEngine17-SamplePack1_v3_house7"` - if it exists - for `5.0` seconds. After 5.0 seconds, the first person or third person physics camera will be activated.

Example 2

```
--Script name is prefabInstanceCamera.lua

function Init()
```

```
	ActivateImportedCameraOfPrefabInstance("this", "Camera-camera")
end

function Update()

end
```

If, in the Prefab Editor, you attach prefabInstanceCamera.lua script to a Prefab that has an imported "Camera-camera", the "this" parameter in the ActivateImportedCameraOfPrefabInstance function will point to instances of that Prefab in current VScene. For example, if you have an Instance named *instance1_a* from a Prefab named *a* to which this script is attached, "this" in ActivateImportedCameraOfPrefabInstance function refers to the name *instance1_a*.
In this case, ActivateImportedCameraOfPrefabInstance function activates the "Camera-camera" of Prefab Instance named *instance1_a* - if it exists - indefinitely.

4.5. ActivateThirdPersonCamera

Definition

```
ActivateThirdPersonCamera()
```

Description

This function activates the third-person physics camera attached to the main game character.

Example

```
function Init()
    ActivateThirdPersonCamera()
end

function Update()

end
```

4.6. AddForceToCharacterController

Definition

`AddForceToCharacterController(float forceX, float forceY, float forceZ, float forceSpeed, float forceDecreaseValue)`

Description

This function applies physics force to the main character of the game.

Parameters

`forceX, forceY, forceZ`

These three values determine the direction of the force that is assigned to the main character of the game. Vanda Engine normalizes the vector (*`forceX, forceY, forceZ`*).

`forceSpeed`

Determines the strength of the force.

`forceDecreaseValue`

Determines how fast the force decreases. The Venda engine multiplies this value by elapsedTime. For example, if we consider forceDecreaseValue as 1, the force will decrease by 1 unit per second.

Example

```
function OnTriggerEnter(otherActorName)
    AddForceToCharacterController(1.0, 10.0, 1.0, 20.0, 5.0)
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)

end
```

Let's assume that this script is attached to a trigger called trigger1. When the main character or any object with dynamic physics enters this trigger, a force of 20.0 units is applied to the character in the normalized direction (1.0, 10.0, 1.0) and its power decreases by 5 units per second.

4.7. AddForceToPrefabInstance

Definition

```
AddForceToPrefabInstance(string prefabInstanceName, float forceX, float
forceY, float forceZ, float forcePower)
```

Description

This function applies force to the prefab instance that has dynamic physics. To view the name of prefab instances, open the VScene and click on the desired Prefab Instance in the "Prefabs and GUIs" section and press the Edit button. You can also access the names of prefab instances from the Script Utility section of the script editor (Tools > Script Editor > Tools > Script Utility). In the dialog that appears, you can view and copy the name of the prefab instance.

Parameters

prefabInstanceName
Specifies the name of the prefab instance that has dynamic physics. You can also use the name "this" for this parameter. In this case, "this" refers to the prefab instance that this script is attached to.

forceX, forceY, forceZ
These three values determine the direction of the force that is applied to prefab instance. Vanda Engine normalizes the vector (*forceX, forceY, forceZ*).

forcePower
Determines the strength of the force.

Example 1

```
function Init()
AddForceToPrefabInstance("1_VandaEngine17-SamplePack1_f1_barrel", 1.0, 1.0, 1.0, 5.0)
end

function Update()

end
```

This function applies a force of `5.0` units in the normalized direction (`1.0, 1.0, 1.0`) to the `"1_VandaEngine17-SamplePack1_f1_barrel"` prefab instance.

Example 2

```
--name of the script is addforcetoprefabinstance2.lua
function Init()
AddForceToPrefabInstance("this", 1.0, 0.0, 0.0, 4.0)
end

function Update()

end
```

If, in the Prefab Editor, you attach `addforcetoprefabinstance2.lua` script to a Prefab, then `"this"` parameter in the `AddForceToPrefabInstance` function

`will point to instances of that Prefab in current VScene. For example, if you have an Instance named` *`instance1_a`* `from a Prefab named` *`a`* `to which this script is attached, "this" in AddForceToPrefabInstance function refers to the name` *`instance1_a`*`.`
This function applies a force of `4.0` units in the normalized direction (`1.0, 0.0, 0.0`) to the current prefab instance.

4.8. AddTorqueToPrefabInstance

Definition

```
AddTorqueToPrefabInstance(string prefabInstanceName, float torqueX, float
torqueY, float torqueZ, float torquePower)
```

Description

This function applies torque to the prefab instance that has dynamic physics. To view the name of prefab instances, open the VScene and click on the desired Prefab Instance in the "Prefabs and GUIs" section and press the Edit button. You can also access the names of prefab instances from the Script Utility section of the script editor (Tools > Script Editor > Tools > Script Utility). In the dialog that appears, you can view and copy the name of the prefab instance.

Parameters

prefabInstanceName

Specifies the name of the prefab instance that has dynamic physics. You can also use the name "this" for this parameter. In this case, "this" refers to the prefab instance that this script is attached to.

torqueX, torqueY, torqueZ

These three values determine the direction of the torque that is applied to prefab instance. Vanda Engine normalizes the vector (***torqueX, torqueY, torqueZ***).

torquePower

Determines the strength of the torque.

Example 1

```
function Init()
    AddTorqueToPrefabInstance("1_VandaEngine17-SamplePack1_f1_barrel", 1.0, 1.0, 1.0,
15.0)
end

function Update()

end
```

This function applies a torque of `15.0` units in the normalized direction (`1.0, 1.0, 1.0`) to the `"1_VandaEngine17-SamplePack1_f1_barrel"` prefab instance.

Example 2

```
--name of the script is addtorquetoprefabinstance2.lua
function Init()
AddTorqueToPrefabInstance("this", 1.0, 0.0, 0.0, 10.0)
end

function Update()

end
```

If, in the Prefab Editor, you attach addtorquetoprefabinstance2.lua script to a Prefab, then "this" parameter in the AddTorqueToPrefabInstance function will point to instances of that Prefab in current VScene. For example, if you have an Instance named *instance1_a* from a Prefab named *a* to which this script is attached, "this" in AddTorqueToPrefabInstance function refers to the name *instance1_a*.
This function applies a torque of 10.0 units in the normalized direction (1.0, 0.0, 0.0) to the current prefab instance.

4.9. AttachPrefabInstanceToWater

Definition

`AttachPrefabInstanceToWater(string prefabInstanceName, string waterObjectName)`

Description

This function attaches the prefab instance *prefabInstanceName* to the water *waterObjectName*. In this case, you can see the reflection of the prefab instance in the water.

Parameters

prefabInstanceName
Specifies the name of the prefab instance. You can also use the name "this" for this parameter. In this case, "this" refers to the prefab instance that this script is attached to.

waterObjectName
Specifies the name of the water. You can also use the name "this" for this parameter. In this case, "this" refers to the water object that this script is attached to.

Note: You can't use "this" string for both prefabInstanceName and waterObjectName at the same time.

Example 1

```lua
function Init()
    AttachPrefabInstanceToWater("1_VandaEngine17-SamplePack1_house2", "water1")
end

function Update()

end
```

Attaches prefab instance `"1_VandaEngine17-SamplePack1_house2"` to water object `"water1"`.

Example 2

```lua
--name of script is AttachPrefabInstanceToWater2.lua
function Init()
    AttachPrefabInstanceToWater("this", "water1")
end

function Update()

end
```

If, in the Prefab Editor, you attach `AttachPrefabInstanceToWater2.lua` script to a Prefab, then `"this"` parameter in the `AttachPrefabInstanceToWater` function will point to instances of that Prefab in current VScene. For example, if you have an Instance named *instance1_a* from a Prefab named *a* to which this script is attached, `"this"` in `AttachPrefabInstanceToWater` function refers to the name *instance1_a*.
This script attaches current prefab instance to the water object `"water1"`.

Example 3

```
--name of script is AttachPrefabInstanceToWater3.lua
function Init()
    AttachPrefabInstanceToWater("1_VandaEngine17-SamplePack1_house2", "this")
end

function Update()

end
```

Attaches prefab instance `"1_VandaEngine17-SamplePack1_house2"` to current water object. For example, if you attach the `AttachPrefabInstanceToWater3.lua` script to a water named "water1", then the name `"this"` will be equivalent to "water1".

4.10. CloseFile

Definition

`CloseFile(string filePath)`

Description

Closes the file located in "Assets/Data/filePath". If the file is not found, it returns an error message.

Parameters

`filePath`

File path in "Assets/Data/" folder.

Example

```
function Init()
    OpenFileForWriting("level1/data.bin")
    --write data to file here
    closefile("level1/data.bin")
end

function Update()

end
```

In this example, the function `closefile` Closes the `"data.bin"` file located in "Assets/Data/level1/" path.

4.11. CreateFolder

Definition

`CreateFolder(string folderPath)`

Description

Creates `folderPath` folder in the "Assets/Data/" path.

Parameters

`folderPath`

Folder path in "Assets/Data/" folder.

Example

```
function Init()
    CreateFolder("level1")
    CreateFolder("level1/subLevel1")
end

function Update()

end
```

The first call to the `CreateFolder` function creates a folder named `"level1"` in the "Assets/Data/" path. The second call to the `CreateFolder` function creates a folder named `"subLevel1"` in the path "Assets/Data/level1/". If we used only one function call as `CreateFolder("level1/subLevel1")`, no folder would be created and the function would return an error message. Always create folders from the root path one by one.

4.12. DeleteAllResources

Definition

`DeleteAllResources()`

Description

Removes all resource files from memory.

Example

```
function Init()
LoadResource("sounds", "mouseHover.ogg")
LoadResource("images", "pointer.dds")

DeleteAllResources()
end

function Update()

end
```

In this case, `DeleteAllResources` function unloads the two previous OGG and DDS resource files that were loaded in the memory using `LoadResource` function.

4.13. DeletePrefabInstance

Definition

`DeletePrefabInstance(string prefabInstanceName)`

Description

Removes the prefab instance *`prefabInstanceName`* from memory.

Parameters

`prefabInstanceName`

Specifies the name of the prefab instance.

Example

```
function Init()
    DeletePrefabInstance("1_VandaEngine17-SamplePack1_stab")
end

function Update()

end
```

This function removes the prefab instance `"1_VandaEngine17-SamplePack1_stab"` from memory.

4.14. DeleteResource

Definition

`DeleteResource(string resourceDirectoryName, string resourceFileName)`

Description

Removes the resource file *`resourceFileName`* in folder *`resourceDirectoryName`* from memory. You can access reource directory and file names through Tools > Script Editor > Tools > Script Utility.

Parameters

`resourceDirectoryName`
Specifies the resource directory name.

`resourceFileName`
Specifies the resource file name.

Example

```
function Init()
    LoadResource("sounds", "mouseHover.ogg")
    LoadResource("images", "pointer.dds")

    DeleteResource("sounds", "mouseHover.ogg")
end

function Update()

end
```

In this example, the `DeleteResource` function deletes the resource file `"mouseHover.ogg"` located in folder `"sounds"` from memory.

4.15. DetachPrefabInstanceFromWater

Definition

`DetachPrefabInstanceFromWater(string prefabInstanceName, string waterObjectName)`

Description

This function detaches the prefab instance ***prefabInstanceName*** from the water ***waterObjectName***. In this case, you can't see the reflection of the prefab instance in the water.

Parameters

prefabInstanceName
Specifies the name of the prefab instance. You can also use the name "this" for this parameter. In this case, "this" refers to the prefab instance that this script is attached to.

waterObjectName
Specifies the name of the water. You can also use the name "this" for this parameter. In this case, "this" refers to the water object that this script is attached to.

Note: You can't use "this" string for both prefabInstanceName and waterObjectName at the same time.

Example 1

```
function Init()
    DetachPrefabInstanceFromWater("1_VandaEngine17-SamplePack1_house2", "water1")
end

function Update()

end
```

Detaches prefab instance `"1_VandaEngine17-SamplePack1_house2"` from water object `"water1"`.

Example 2

```
--name of script is DetachPrefabInstanceFromWater2.lua
function Init()
    DetachPrefabInstanceFromWater("this", "water1")
end

function Update()

end
```

If, in the Prefab Editor, you attach `DetachPrefabInstanceFromWater2.lua` script to a Prefab, then `"this"` parameter in the `DetachPrefabInstanceFromWater` function will point to instances of that Prefab in current VScene. For example, if you have an Instance named *instance1_a* from a Prefab named *a* to which this script is attached, `"this"` in `DetachPrefabInstanceFromWater` function refers to the name *instance1_a*.
This script detaches current prefab instance from the water object `"water1"`.

Example 3

```lua
--name of script is DetachPrefabInstanceFromWater3.lua
function Init()
    DetachPrefabInstanceFromWater("1_VandaEngine17-SamplePack1_house2", "this")
end

function Update()

end
```

Detaches prefab instance `"1_VandaEngine17-SamplePack1_house2"` from current water object. For example, if you attach the `DetachPrefabInstanceFromWater3.lua` script to a water named "water1", then the name `"this"` will be equivalent to "water1".

4.16. DisableBloom

Definition

```
DisableBloom()
```

Description

As its name implies, this function disables the bloom effect.

Example

```
function Init()
    DisableBloom()
end

function Update()

end
```

4.17. DisableCharacterControllerJump

Definition

```
DisableCharacterControllerJump()
```

Description

As its name implies, this function disables the jump of main character.

Example

```
function Init()
    DisableCharacterControllerJump()
end

function Update()

end
```

4.18. DisableDepthOfField

Definition

`DisableDepthOfField()`

Description

As its name implies, this function disables the depth of field effect.

Example

```
function Init()
    DisableDepthOfField()
end

function Update()

end
```

4.19. DisableDirectionalShadow

Definition

```
DisableDirectionalShadow()
```

Description

This function disables the shadow of directional light.

Example

```
function Init()
    DisableDirectionalShadow()
end

function Update()

end
```

4.20. DisableFog

Definition

```
DisableFog()
```

Description

As its name implies, this function disables fog.

Example

```
function Init()
    DisableFog()
end

function Update()

end
```

4.21. DisableGeneralWaterReflection

Definition

```
DisableGeneralWaterReflection()
```

Description

This function disables reflection of all water objects.

Example

```
function Init()
    DisableGeneralWaterReflection()
end

function Update()

end
```

4.22. DisablePhysicsDebugMode

Definition

```
DisablePhysicsDebugMode()
```

Description

As its name implies, this function disables physics debug mode.

Example

```
function Init()
    DisablePhysicsDebugMode()
end

function Update()

end
```

4.23. DisablePhysicsGravity

Definition

```
DisablePhysicsGravity()
```

Description

As its name implies, this function disables physics gravity.

Example

```
function Init()
    DisablePhysicsGravity()
end

function Update()

end
```

4.24. DisablePhysicsGroundPlane

Definition

```
DisablePhysicsGroundPlane()
```

Description

As its name implies, this function disables default physics ground plane.

Example

```
function Init()
    DisablePhysicsGroundPlane()
end

function Update()

end
```

4.25. DisablePrefabInstanceMaterial

Definition

`DisablePrefabInstanceMaterial(string prefabInstanceName)`

Description

This function disables the material of prefab instance **`prefabInstanceName`**. In this case, its prefab material is used instead of prefab instance material. By default, prefab instance material is disabled.

Parameter

prefabInstanceName

Specifies the name of the prefab instance. You can also use the name "this" for this parameter. In this case, "this" refers to the prefab instance that this script is attached to.

Example 1

```
function Init()
    DisablePrefabInstanceMaterial("1_VandaEngine17-SamplePack1_f1_barrel")
end

function Update()

end
```

This script disables the material of prefab instance `"1_VandaEngine17-SamplePack1_f1_barrel"`.

Example 2

```
--Script name is DisablePrefabInstanceMaterial2.lua

function Init()
    DisablePrefabInstanceMaterial("this")
end

function Update()

end
```

If, in the Prefab Editor, you attach `DisablePrefabInstanceMaterial2.lua` script to a Prefab, then `"this"` parameter in the `DisablePrefabInstanceMaterial` function will point to instances of that Prefab in current VScene. For example, if you have an Instance named *instance1_a* from a Prefab named *a* to which this script is attached, `"this"` in `DisablePrefabInstanceMaterial` function refers to the name *instance1_a*.
This script disables the material of current prefab instance (for example, *instance1_a*).

4.26. DisableSkyFog

Definition

```
DisableSkyFog()
```

Description

This function disables sky fog. Note that sky fog is disabled by default.

Example

```
function Init()
    DisableSkyFog()
end

function Update()

end
```

4.27. DisableVSync

Definition

```
DisableVSync()
```

Description

This function disables VSync. Note that VSync is disabled by default.

Example

```
function Init()
    DisableVSync()
end

function Update()

end
```

4.28. DisableWaterShadow

Definition

`DisableWaterShadow(string waterName)`

Description

This function disables the shadow of reflections of objects in water.

Parameters

`waterName`

Specifies the water name. You can also use the name "this" for this parameter. In this case, "this" refers to the water object that this script is attached to.

Example 1

```
function Init()
    DisableWaterShadow("water1")
end

function Update()

end
```

Disables the shadow of reflections of objects in water `"water1"`.

Example 2

```
--name of script is DisableWaterShadow2.lua
function Init()
    DisableWaterShadow("this")
end

function Update()

end
```

Disables the shadow of reflections of objects in current water. For example, if you attach the `DisableWaterShadow2.lua` script to a water named "water1", then the name `"this"` will be equivalent to "water1".

4.29. DisableWaterSunReflection

Definition

`DisableWaterSunReflection(string waterName)`

Description

This function disables the reflection of the sun in the water.

Parameters

`waterName`

Specifies the water name. You can also use the name "this" for this parameter. In this case, "this" refers to the water object that this script is attached to.

Example 1

```
function Init()
    DisableWaterSunReflection("water1")
end

function Update()

end
```

Disables the reflection of the sun in water `"water1"`

Example 2

```
--name of script is DisableWaterSunReflection2.lua
function Init()
    DisableWaterSunReflection("this")
end

function Update()

end
```

Disables the reflection of the sun in current water. For example, if you attach the `DisableWaterSunReflection2.lua` script to a water named "water1", then the name `"this"` will be equivalent to "water1".

4.30. EnableBloom

Definition

```
EnableBloom()
```

Description

As its name implies, this function enables the bloom effect.

Example

```
function Init()
    EnableBloom()
end

function Update()

end
```

4.31. EnableCharacterControllerJump

Definition

`EnableCharacterControllerJump()`

Description

As its name implies, this function enables the jump of main character.

Example

```
function Init()
    EnableCharacterControllerJump()
end

function Update()

end
```

4.32. EnableDepthOfField

Definition

```
EnableDepthOfField()
```

Description

As its name implies, this function enables the depth of field effect.

Example

```
function Init()
    EnableDepthOfField()
end

function Update()

end
```

4.33. EnableDirectionalShadow

Definition

```
EnableDirectionalShadow()
```

Description

This function enables the shadow of directional light.

Example

```
function Init()
    EnableDirectionalShadow()
end

function Update()

end
```

4.34. EnableFog

Definition

```
EnableFog()
```

Description

As its name implies, this function enables fog.

Example

```
function Init()
    EnableFog()
end

function Update()

end
```

4.35. EnableGeneralWaterReflection

Definition

```
EnableGeneralWaterReflection()
```

Description

This function enables reflection of all water objects.

Example

```
function Init()
    EnableGeneralWaterReflection()
end

function Update()

end
```

4.36. EnablePhysicsDebugMode

Definition

```
EnablePhysicsDebugMode()
```

Description

As its name implies, this function enables physics debug mode.

Example

```
function Init()
    EnablePhysicsDebugMode()
end

function Update()

end
```

4.37. EnablePhysicsGravity

Definition

`EnablePhysicsGravity()`

Description

As its name implies, this function enables physics gravity.

Example

```
function Init()
    EnablePhysicsGravity()
end

function Update()

end
```

4.38. EnablePhysicsGroundPlane

Definition

```
EnablePhysicsGroundPlane()
```

Description

As its name implies, this function enables default physics ground plane.

Example

```
function Init()
    EnablePhysicsGroundPlane()
end

function Update()

end
```

4.39. EnablePrefabInstanceMaterial

Definition

`EnablePrefabInstanceMaterial(string prefabInstanceName)`

Description

This function enables the material of prefab instance `prefabInstanceName`. In this case, prefab instance material is used instead of its prefab material. By default, prefab instance material is disabled.

Parameter

prefabInstanceName

Specifies the name of the prefab instance. You can also use the name "this" for this parameter. In this case, "this" refers to the prefab instance that this script is attached to.

Example 1

```lua
function Init()
    EnablePrefabInstanceMaterial("1_VandaEngine17-SamplePack1_f1_barrel")
end

function Update()

end
```

This script enables the material of prefab instance `"1_VandaEngine17-SamplePack1_f1_barrel"`.

Example 2

```lua
--Script name is EnablePrefabInstanceMaterial2.lua

function Init()
    EnablePrefabInstanceMaterial("this")
end

function Update()

end
```

If, in the Prefab Editor, you attach `EnablePrefabInstanceMaterial2.lua` script to a Prefab, then `"this"` parameter in the `EnablePrefabInstanceMaterial` function will point to instances of that Prefab in current VScene. For example, if you have an Instance named *instance1_a* from a Prefab named *a* to which this script is attached, `"this"` in `EnablePrefabInstanceMaterial` function refers to the name *instance1_a*.

This script enables the material of current prefab instance (for example, *instance1_a*).

4.40. EnableSkyFog

Definition

```
EnableSkyFog()
```

Description

This function enables sky fog by setting the sky fog attribute to true. To activate the sky fog, you must also activate the general fog through the Modify > Fog menu or the EnableFog() function.

Example

```
function Init()
    EnableSkyFog()
end

function Update()

end
```

4.41. EnableVSync

Definition

```
EnableVSync()
```

Description

This function enables VSync. Note that VSync is disabled by default.

Example

```
function Init()
    EnableVSync()
end

function Update()

end
```

4.42. EnableWaterShadow

Definition

```
EnableWaterShadow(string waterName)
```

Description

This function enables the shadow of reflections of objects in water.

Parameters

`waterName`

Specifies the water name. You can also use the name "this" for this parameter. In this case, "this" refers to the water object that this script is attached to.

Example 1

```
function Init()
    EnableWaterShadow("water1")
end

function Update()

end
```

Enables the shadow of reflections of objects in water `"water1"`.

Example 2

```
--name of script is EnableWaterShadow2.lua
function Init()
    EnableWaterShadow("this")
end

function Update()

end
```

Enables the shadow of reflections of objects in current water. For example, if you attach the `EnableWaterShadow2.lua` script to a water named "water1", then the name `"this"` will be equivalent to "water1".

4.43. EnableWaterSunReflection

Definition

`EnableWaterSunReflection(string waterName)`

Description

This function enables the reflection of the sun in the water.

Parameters

`waterName`

Specifies the water name. You can also use the name "this" for this parameter. In this case, "this" refers to the water object that this script is attached to.

Example 1

```lua
function Init()
    EnableWaterSunReflection("water1")
end

function Update()

end
```

Enables the reflection of the sun in water `"water1"`

Example 2

```lua
--name of script is EnableWaterSunReflection2.lua
function Init()
    EnableWaterSunReflection("this")
end

function Update()

end
```

Enables the reflection of the sun in current water. For example, if you attach the `EnableWaterSunReflection2.lua` script to a water named "water1", then the name `"this"` will be equivalent to "water1".

4.44. ExecuteCyclicAnimation

Definition

`ExecuteCyclicAnimation(string prefabInstanceName, string animationClipName, float weightTarget, float delayIn)`

Description

A cyclic animation is an animation that is repeating itself. `ExecuteCyclicAnimation` adjusts the weight of a cyclic animation of prefab instance in a given amount of time. This can be used to fade in a new cycle or to modify the weight of an active cycle.

Parameters

prefabInstanceName
Specifies the name of the prefab instance. You can also use the name "this" for this parameter. In this case, "this" refers to the prefab instance that this script is attached to.

animationClipName
Specifies the name of the prefab instance animation. To view the name of the prefab instance animations, you can go to the Modify > Properties menu in the prefab editor, or select the name of the prefab instance from the Prefabs and GUIs section in the current VScene and press the Edit button.

weightTarget
Specifies the final weight of the animation clip ***animationClipName***. A value of 1 means full animation and a value of 0 means no animation. This value must be in the range (0.0,1.0].

delayIn
Specifies when the ***animationClipName*** reaches the ***weightTarget*** weight. This value must be 0.0 or higher.

Example 1

```
function Init()
    ExecuteCyclicAnimation("1_animation_test_boy", "defaultClip", 1.0, 0.5)
end

function Update()

end
```

In the first `0.5` seconds, the `"defaultClip"` animation value of prefab instance `"1_animation_test_boy"` goes from weight 0 to weight `1.0` (full animation).

Example 2

```
--name of script is executecyclicanimation2.lua

animation = true

function Init()

end
```

```
function Update()
    if animation == true then
          ExecuteCyclicAnimation("this", "run", 0.3, 1.0)
          animation = false
    end
end
```

If, in the Prefab Editor, you attach executecyclicanimation2.lua script to a Prefab that has an animation clip "run", then "this" parameter in the ExecuteCyclicAnimation function will point to instances of that Prefab in current VScene. For example, if you have an Instance named *instance1_a* from a Prefab named *a* to which this script is attached, "this" in ExecuteCyclicAnimation function refers to the name *instance1_a*. In this case, In the first 1.0seconds, the "run" animation value of prefab instance *instance1_a* goes from weight 0 to weight 0.3 (30% of animation "run").

4.45. ExecuteNonCyclicAnimation

Definition

```
ExecuteNonCyclicAnimation(string prefabInstanceName, string
animationClipName, float delayIn, float delayOut, float weightTarget, bool
lock)
```

Description

This function execute an animation of prefab instance once, instead of repeating it.

Parameters

prefabInstanceName
Specifies the name of the prefab instance. You can also use the name "this" for this parameter. In this case, "this" refers to the prefab instance that this script is attached to.

animationClipName
Specifies the name of the prefab instance animation. To view the name of the prefab instance animations, you can go to the Modify > Properties menu in the prefab editor, or select the name of the prefab instance from the Prefabs and GUIs section in the current VScene and press the Edit button.

delayIn
Specifies when the ***animationClipName*** reaches the ***weightTarget*** weight. This value must be 0.0 or higher.

delayOut
Specifies the fade out time at the end of the animation, when the weight of the animation reaches 0. This value must be 0.0 or higher.

weightTarget
Specifies the final weight of the animation clip ***animationClipName***. A value of 1 means full animation and a value of 0 means no animation. This value must be in the range (0.0,1.0]

lock
If this attribute is true, the animation will be locked at the last frame. For example, suppose you have a door animation and you want the door to remain open after the animation plays. In this case, you need to lock it in the last frame. Otherwise, after the animation ends, the door will return to the first state.

Note: delayIn + delayOut time must not be greater than the duration of animation animationClipName

Example 1

```
function Init()
    ExecuteNonCyclicAnimation("1_animation_test_boy", "defaultClip", 0.5, 0.7, 1.0,
false)
end

function Update()
```

```
end
```

In the first `0.5` seconds, the `"defaultClip"` animation value of prefab instance `"1_animation_test_boy"` goes from weight 0 to weight `1.0` (full animation). Then, `0.7` seconds before the end of the animation, the weight of the `"defaultClip"` animation starts to decrease, and at the end of the animation, its weight reaches zero. This animation is not locked in the last frame.

Example 2

```
--name of script is executenoncyclicanimation2.lua

animation = true

function Init()

end

function Update()
    if animation == true then
          ExecuteNonCyclicAnimation("this", "run", 0.5, 0.6, 0.4, true)
          animation = false
    end
end
```

If, in the Prefab Editor, you attach `executenoncyclicanimation2.lua` script to a Prefab that has an animation clip `"run"`, then `"this"` parameter in the `ExecuteNonCyclicAnimation` function will point to instances of that Prefab in current VScene. For example, if you have an Instance named *instance1_a* from a Prefab named *a* to which this script is attached, `"this"` in `ExecuteNonCyclicAnimation` function refers to the name *instance1_a*. In this case, In the first `0.5` seconds, the `"run"` animation value of prefab instance *instance1_a* goes from weight 0 to weight `0.4` (40% of animation `"run"`). Then, `0.6` seconds before the end of the animation, the weight of the `"defaultClip"` animation starts to decrease, and at the end of the animation, its weight reaches zero. This animation is locked in the last frame.

4.46. ExitGame

Definition

```
ExitGame()
```

Description

This function causes exit from the game.

Example

```
function OnSelectMouseLButtonDown()
    ExitGame()
end

function OnSelectMouseRButtonDown()

end

function OnSelectMouseEnter()

end
```

Assume that this script is attached to a button. In this case, whenever the user left clicks on that button, this script will exit the game.

4.47. GeneratePrefabInstance

Definition

`GeneratePrefabInstance(string prefabName, float XPos, float YPos, float ZPos, float XRot, float YRot, float ZRot, float XScale, float YScale, float ZScale)`

Description

This function creates an instance of prefab `prefabName` and returns its name.

Parameters

`prefabName`

Specifies the name of the prefab from which you want to create an instance. You can see the names of prefabs through the Script Utility dialog in the script editor (Tools > Script Editor > Tools > Script Utility).

`XPos, YPos, ZPos`

These three values specify the position of the generated prefab instance.

`XRot, YRot, ZRot`

These three values specify the rotation of the generated prefab instance.

`XScale, YScale, ZScale`

These three values specify the scale of the generated prefab instance.

Return Value

Returns the name of the generated prefab instance.

Example

```
prefab_instance = ""

function OnTriggerEnter(otherActorName)
    prefab_instance = GeneratePrefabInstance("VandaEngine17-SamplePack1_house2", 1.0, 
2.0, 3.0, 10.0, 20.0, 30.0, 0.3, 0.5, 0.7)
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)
    DeletePrefabInstance(prefab_instance)
end
```

Let's assume that this script is attached to a trigger called Trigger1. When the main game character or a dynamic object is entered into Trigger1, the `GeneratePrefabInstance` function is called and an instance of the prefab `"VandaEngine17-SamplePack1_house2"` is created at position (`1.0, 2.0, 3.0`) with rotation (`10.0, 20.0, 30.0`) and dimensions (`0.3, 0.5, 0.7`). Then the generated prefab instance name is stored in the `prefab_instance` variable.

Whenever the character or any other dynamic object exits Trigger1, the `DeletePrefabInstance` function deletes the generated prefab instance **`prefab_instance`** from memory.

4.48. Get3DSoundScriptBoolVariable

Definition

```
bool Get3DSoundScriptBoolVariable(string 3DSoundName, string variable)
```

Description

This function gets the value of the Boolean `variable` defined in the script attached to the `3DSoundName` 3D sound object.

Parameters

3DSoundName
Specifies the name of the 3D sound object.

variable
Specifies the name of the Boolean variable defined in the script attached to the `3DSoundName` 3D sound.

Return Value

Returns the value of the Boolean `variable` defined in the script attached to the `3DSoundName` 3D sound object.

Example

```
--script name is Get3DSoundScriptBoolVariable.lua attached a to game object such as water
value = false
function Init()
    value = Get3DSoundScriptBoolVariable("sound1", "a")
end

function Update()

end
```

Assuming that the variable `"a"` is defined with the value *true* in the script attached to the 3D sound object `"sound1"`, `Get3DSoundScriptBoolVariable` function returns the value *true*.

4.49. Get3DSoundScriptDoubleVariable

Definition

```
double Get3DSoundScriptDoubleVariable(string 3DSoundName, string variable)
```

Description

This function gets the value of the Double **`variable`** defined in the script attached to the **`3DSoundName`** 3D sound object.

Parameters

`3DSoundName`
Specifies the name of the 3D sound object.

`variable`
Specifies the name of the Double variable defined in the script attached to the **`3DSoundName`** 3D sound.

Return Value

Returns the value of the Double **`variable`** defined in the script attached to the **`3DSoundName`** 3D sound object.

Example

```
--script name is Get3DSoundScriptDoubleVariable.lua attached a to game object such as
water
return_value = 0.0

function Init()
    return_value = Get3DSoundScriptDoubleVariable("sound1", "a")
end

function Update()

end
```

Assuming that the variable `"a"` is defined with the value *1.0* in the script attached to the 3D sound object `"sound1"`, `Get3DSoundScriptDoubleVariable` function returns the value *1.0*.

4.50. Get3DSoundScriptIntVariable

Definition

`int Get3DSoundScriptIntVariable(string 3DSoundName, string variable)`

Description

This function gets the value of the Integer **`variable`** defined in the script attached to the **`3DSoundName`** 3D sound object.

Parameters

`3DSoundName`
Specifies the name of the 3D sound object.

`variable`
Specifies the name of the Integer variable defined in the script attached to the **`3DSoundName`** 3D sound.

Return Value

Returns the value of the Integer **`variable`** defined in the script attached to the **`3DSoundName`** 3D sound object.

Example

```
--script name is Get3DSoundScriptIntVariable.lua attached a to game object such as water
return_value = 0

function Init()
    return_value = Get3DSoundScriptIntVariable("sound1", "a")
end

function Update()

end
```

Assuming that the variable `"a"` is defined with the value *1* in the script attached to the 3D sound object `"sound1"`, `Get3DSoundScriptIntVariable` function returns the value *1*.

4.51. Get3DSoundScriptStringVariable

Definition

`string Get3DSoundScriptStringVariable(string 3DSoundName, string variable)`

Description

This function gets the value of the String **`variable`** defined in the script attached to the **`3DSoundName`** 3D sound object.

Parameters

`3DSoundName`
Specifies the name of the 3D sound object.

`variable`
Specifies the name of the String variable defined in the script attached to the **`3DSoundName`** 3D sound.

Return Value

Returns the value of the String **`variable`** defined in the script attached to the **`3DSoundName`** 3D sound object.

Example

```
--script name is Get3DSoundScriptStringVariable.lua attached a to game object such as
water
return_value = ""

function Init()
    return_value = Get3DSoundScriptStringVariable("sound1", "a")
end

function Update()

end
```

Assuming that the variable **`"a"`** is defined with the value "*hello*" in the script attached to the 3D sound object **`"sound1"`**, **`Get3DSoundScriptStringVariable`** function returns the value "*hello*".

4.52. GetAmbientSoundScriptBoolVariable

Definition

`bool GetAmbientSoundScriptBoolVariable(string ambientSoundName, string variable)`

Description

This function gets the value of the Boolean **`variable`** defined in the script attached to the **`ambientSoundName`** ambient sound object.

Parameters

`ambientSoundName`

Specifies the name of the ambient sound object.

`variable`

Specifies the name of the Boolean variable defined in the script attached to the **`ambientSoundName`** ambient sound.

Return Value

Returns the value of the Boolean **`variable`** defined in the script attached to the **`ambientSoundName`** ambient sound object.

Example

```
--script name is GetAmbientSoundScriptBoolVariable.lua attached a to game object such as
water
value = false
function Init()
    value = GetAmbientSoundScriptBoolVariable("sound1", "a")
end

function Update()

end
```

Assuming that the variable `"a"` is defined with the value *true* in the script attached to the ambient sound object `"sound1"`, `GetAmbientSoundScriptBoolVariable` function returns the value *true*.

4.53. GetAmbientSoundScriptDoubleVariable

Definition

`double GetAmbientSoundScriptDoubleVariable(string ambientSoundName, string variable)`

Description

This function gets the value of the Double **`variable`** defined in the script attached to the **`ambientSoundName`** ambient sound object.

Parameters

`ambientSoundName`

Specifies the name of the ambient sound object.

`variable`

Specifies the name of the Double variable defined in the script attached to the **`ambientSoundName`** ambient sound.

Return Value

Returns the value of the Double **`variable`** defined in the script attached to the **`ambientSoundName`** ambient sound object.

Example

```
--script name is GetAmbientSoundScriptDoubleVariable.lua attached a to game object such as water
return_value = 0.0

function Init()
    return_value = GetAmbientSoundScriptDoubleVariable("sound1", "a")
end

function Update()

end
```

Assuming that the variable `"a"` is defined with the value *1.0* in the script attached to the ambient sound object `"sound1"`, `GetAmbientSoundScriptDoubleVariable` function returns the value *1.0*.

4.54. GetAmbientSoundScriptIntVariable

Definition

`int GetAmbientSoundScriptIntVariable(string ambientSoundName, string variable)`

Description

This function gets the value of the Integer **`variable`** defined in the script attached to the **`ambientSoundName`** ambient sound object.

Parameters

`ambientSoundName`
Specifies the name of the ambient sound object.

`variable`
Specifies the name of the Integer variable defined in the script attached to the **`ambientSoundName`** ambient sound.

Return Value

Returns the value of the Integer **`variable`** defined in the script attached to the **`ambientSoundName`** ambient sound object.

Example

```lua
--script name is GetAmbientSoundScriptIntVariable.lua attached a to game object such as water
return_value = 0

function Init()
    return_value = GetAmbientSoundScriptIntVariable("sound1", "a")
end

function Update()

end
```

Assuming that the variable **`"a"`** is defined with the value *1* in the script attached to the ambient sound object **`"sound1"`**, **`GetAmbientSoundScriptIntVariable`** function returns the value *1*.

4.55. GetAmbientSoundScriptStringVariable

Definition

`string GetAmbientSoundScriptStringVariable(string ambientSoundName, string variable)`

Description

This function gets the value of the String **`variable`** defined in the script attached to the **`ambientSoundName`** ambient sound object.

Parameters

`ambientSoundName`
Specifies the name of the ambient sound object.

`variable`
Specifies the name of the String variable defined in the script attached to the **`ambientSoundName`** ambient sound.

Return Value

Returns the value of the String **`variable`** defined in the script attached to the **`ambientSoundName`** ambient sound object.

Example

```
--script name is GetAmbientSoundScriptStringVariable.lua attached a to game object such as water
return_value = ""

function Init()
    return_value = GetAmbientSoundScriptStringVariable("sound1", "a")
end

function Update()

end
```

Assuming that the variable **`"a"`** is defined with the value "*hello*" in the script attached to the ambient sound object **`"sound1"`**, **`GetAmbientSoundScriptStringVariable`** function returns the value "*hello*".

4.56. GetAnimationClipDurationOfPrefabInstance

Definition

`double GetAnimationClipDurationOfPrefabInstance(string prefabInstanceName, string animationClipName)`

Description

This function returns the time of **`animationClipName`** animation of the prefab instance **`prefabInstanceName`**.

Parameters

`prefabInstanceName`

Specifies the name of the prefab instance. You can also use the name "this" for this parameter. In this case, "this" refers to the prefab instance that this script is attached to.

`animationClipName`

Specifies the name of the prefab instance animation. To view the name of the prefab instance animations, you can go to the Modify > Properties menu in the prefab editor, or select the name of the prefab instance from the Prefabs and GUIs section in the current VScene and press the Edit button.

Return Value

Returns the time of **`animationClipName`** animation of the prefab instance **`prefabInstanceName`**.

Example 1

```
animationTime = 0.0

function Init()
    animationTime = GetAnimationClipDurationOfPrefabInstance("1_animation_test_boy",
"defaultClip")

    message = string.format("\nanimation duration is > %.2f" ,animationTime )
    PrintConsole(message)
end

function Update()

end
```

In this case, `GetAnimationClipDurationOfPrefabInstance` returns the the time of `"defaultClip"` animation of the prefab instance `"1_animation_test_boy"`. Then we print the return value of this function. The result would be something like this message:

```
animation duration is > 12.50
```

Example 2

```
--name of this script is GetAnimationClipDurationOfPrefabInstance2.lua

animationTime = 0.0
```

```
function Init()
    animationTime = GetAnimationClipDurationOfPrefabInstance("this", "defaultClip")

    message = string.format("\nanimation duration is > %.2f" ,animationTime )
    PrintConsole(message)
end

function Update()

end
```

If, in the Prefab Editor, you attach GetAnimationClipDurationOfPrefabInstance2.lua script to a Prefab that has "defaultClip" animation, then "this" parameter in the GetAnimationClipDurationOfPrefabInstance function will point to instances of that Prefab in current VScene. For example, if you have an Instance named *instance1_a* from a Prefab named *a* to which this script is attached, "this" in GetAnimationClipDurationOfPrefabInstance function refers to the name *instance1_a*.
In this case, GetAnimationClipDurationOfPrefabInstance function returns the time of "defaultClip" animation of Prefab Instance *instance1_a*.

4.57. GetAnisotropicFilteringValue

Definition

```
int GetAnisotropicFilteringValue()
```

Description

This function returns the anisotropic texture filtering value.

Return Value

Anisotropic texture filtering value.

Example

```
value = 0

function Init()
    value = GetAnisotropicFilteringValue()

    message = string.format("\nAnisotropic filtering value is > %d" ,value )
    PrintConsole(message)
end

function Update()

end
```

4.58. GetBloomColor

Definition

```
double, double, double GetBloomColor()
```

Description

This function returns the bloom color as three values of red, green and blue.

Return Value

Bloom color as three values of red, green and blue. Each value ranges from 0.0 to 1.0.

Example

```
red = 0.0
green = 0.0
blue = 0.0

function Init()
    red, green, blue = GetBloomColor()

    message = string.format("\nBloom color is : (%.2f, %.2f, %.2f)" , red, green, blue)
    PrintConsole(message)
end

function Update()

end
```

In this example, the `GetBloomColor` function returns the value of the red, green, and blue components of the bloom color. Then these three values are displayed on the console by the `PrintConsole` function.

4.59. GetBloomIntensity

Definition

```
double GetBloomIntensity()
```

Description

This function returns the bloom intensity.

Return Value

Bloom intensity.

Example

```
intensity = 0.0

function Init()
    intensity  = GetBloomIntensity()

    message = string.format("\nBloom intensity is : %.2f" ,intensity)
    PrintConsole(message)
end

function Update()

end
```

In this example, the `GetBloomIntensity` function returns the bloom intensity. Then intensity value is displayed on the console by the `PrintConsole` function.

4.60. GetCameraScriptBoolVariable

Definition

```
bool GetCameraScriptBoolVariable(string cameraName, string variable)
```

Description

This function gets the value of the Boolean **`variable`** defined in the script attached to the **`cameraName`** engine camera object.

Parameters

`cameraName`

Specifies the name of the engine camera object.

`variable`

Specifies the name of the Boolean variable defined in the script attached to the **`cameraName`** engine camera.

Return Value

Returns the value of the Boolean **`variable`** defined in the script attached to the **`cameraName`** engine camera object.

Example

```
--script name is GetCameraScriptBoolVariable.lua attached a to game object such as water
value = false
function Init()
    value = GetCameraScriptBoolVariable("camera1", "a")
end

function Update()

end
```

Assuming that the variable `"a"` is defined with the value *true* in the script attached to the engine camera object `"camera1"`, `GetCameraScriptBoolVariable` function returns the value *true*.

4.61. GetCameraScriptDoubleVariable

Definition

```
double GetCameraScriptDoubleVariable(string cameraName, string variable)
```

Description

This function gets the value of the Double **`variable`** defined in the script attached to the **`cameraName`** engine camera object.

Parameters

`cameraName`
Specifies the name of the engine camera object.

`variable`
Specifies the name of the Double variable defined in the script attached to the **`cameraName`** engine camera.

Return Value

Returns the value of the Double **`variable`** defined in the script attached to the **`cameraName`** engine camera object.

Example

```
--script name is GetCameraScriptDoubleVariable.lua attached a to game object such as
water
return_value = 0.0

function Init()
    return_value = GetCameraScriptDoubleVariable("camera1", "a")
end

function Update()

end
```

Assuming that the variable `"a"` is defined with the value *1.0* in the script attached to the engine camera object `"camera1"`, `GetCameraScriptDoubleVariable` function returns the value *1.0*.

4.62. GetCameraScriptIntVariable

Definition

```
int GetCameraScriptIntVariable(string cameraName, string variable)
```

Description

This function gets the value of the Integer **`variable`** defined in the script attached to the **`cameraName`** engine camera object.

Parameters

`cameraName`
Specifies the name of the engine camera object.

`variable`
Specifies the name of the Integer variable defined in the script attached to the **`cameraName`** engine camera.

Return Value

Returns the value of the Integer **`variable`** defined in the script attached to the **`cameraName`** engine camera object.

Example

```
--script name is GetCameraScriptIntVariable.lua attached a to game object such as water
return_value = 0

function Init()
    return_value = GetCameraScriptIntVariable("camera1", "a")
end

function Update()

end
```

Assuming that the variable `"a"` is defined with the value *1* in the script attached to the engine camera object `"camera1"`, `GetCameraScriptIntVariable` function returns the value *1*.

4.63. GetCameraScriptStringVariable

Definition

```
string GetCameraScriptStringVariable(string cameraName, string variable)
```

Description

This function gets the value of the String **`variable`** defined in the script attached to the **`cameraName`** engine camera object.

Parameters

`cameraName`
Specifies the name of the engine camera object.

`variable`
Specifies the name of the String variable defined in the script attached to the **`cameraName`** engine camera.

Return Value

Returns the value of the String **`variable`** defined in the script attached to the **`cameraName`** engine camera object.

Example

```
--script name is GetCameraScriptStringVariable.lua attached a to game object such as
water
return_value = ""

function Init()
    return_value = GetCameraScriptStringVariable("camera1", "a")
end

function Update()

end
```

Assuming that the variable `"a"` is defined with the value "*hello*" in the script attached to the engine camera object `"camera1"`, `GetCameraScriptStringVariable` function returns the value "*hello*".

4.64. GetCharacterControllerCapsuleHeight

Definition

```
double GetCharacterControllerCapsuleHeight()
```

Description

This function returns the height value of the physics character controller capsule.

Return Value

The height value of the physics character controller capsule.

Example

```
value = 0.0

function Init()
    value  = GetCharacterControllerCapsuleHeight()

    message = string.format("\nCharacter Controller Capsule Height is : %.2f" ,value)
    PrintConsole(message)
end

function Update()

end
```

First we get the height value of the physics character controller capsule. Then we display the result in the console using the `PrintConsole` function.

4.65. GetCharacterControllerCapsuleRadius

Definition

```
double GetCharacterControllerCapsuleRadius()
```

Description

This function returns the radius value of the physics character controller capsule.

Return Value

The radius value of the physics character controller capsule.

Example

```
value = 0.0

function Init()
    value  = GetCharacterControllerCapsuleRadius()

    message = string.format("\nCharacter Controller Capsule Radius is : %.2f" ,value)
    PrintConsole(message)
end

function Update()

end
```

First we get the radius value of the physics character controller capsule. Then we display the result in the console using the `PrintConsole` function.

4.66. GetCharacterControllerForcePower

Definition

```
double GetCharacterControllerForcePower()
```

Description

This function returns the force power value of the physics character controller.

Return Value

The force power value of the physics character controller.

Example

```
value = 0.0

function Init()
    value  = GetCharacterControllerForcePower()

    message = string.format("\nCharacter Controller Force Power is : %.2f" ,value)
    PrintConsole(message)
end

function Update()

end
```

First we get the force power value of the physics character controller. Then we display the result in the console using the `PrintConsole` function.

4.67. GetCharacterControllerJumpPower

Definition

```
double GetCharacterControllerJumpPower()
```

Description

This function returns the jump power value of the physics character controller.

Return Value

The jump power value of the physics character controller.

Example

```
value = 0.0

function Init()
    value  = GetCharacterControllerJumpPower()

    message = string.format("\nCharacter Controller Jump Power is : %.2f" ,value)
    PrintConsole(message)
end

function Update()

end
```

First we get the jump power value of the physics character controller. Then we display the result in the console using the `PrintConsole` function.

4.68. GetCharacterControllerPosition

Definition

```
double,double,double GetCharacterControllerPosition()
```

Description

Returns the 3D position of the physics character controller attached to the main character.

Return Value

Returns three values representing the 3D position of the physics character controller.

Example

```
x = 0.0
y = 0.0
z = 0.0

function Init()
    x, y, z = GetCharacterControllerPosition()
end

function Update()

end
```

Assume that character controller is placed at the location (1.0, 2.0, 3.0). In this case, the `GetCharacterControllerPosition` function returns the values 1.0, 2.0 and 3.0 respectively. Therefore, `x` , `y` and `z` will be equal to 1.0, 2.0 and 3.0 respectively.

4.69. GetCharacterControllerRunSpeed

Definition

```
double GetCharacterControllerRunSpeed()
```

Description

This function returns the running speed value of the physics character controller.

Return Value

The running speed value of the physics character controller.

Example

```
value = 0.0

function Init()
    value  = GetCharacterControllerRunSpeed()

    message = string.format("\nCharacter Controller Run Speed is : %.2f" ,value)
    PrintConsole(message)
end

function Update()

end
```

First we get the running speed value of the physics character controller. Then we display the result in the console using the `PrintConsole` function.

4.70. GetCharacterControllerStepOffset

Definition

```
double GetCharacterControllerStepOffset()
```

Description

This function returns the step offset value of the physics character controller.

Return Value

The step offset value of the physics character controller.

Example

```
value = 0.0

function Init()
    value  = GetCharacterControllerStepOffset()

    message = string.format("\nCharacter Controller Step Offset is : %.2f" ,value)
    PrintConsole(message)
end

function Update()

end
```

First we get the step offset value of the physics character controller. Then we display the result in the console using the `PrintConsole` function.

4.71. GetCharacterControllerWalkSpeed

Definition

```
double GetCharacterControllerWalkSpeed()
```

Description

This function returns the walking speed value of the physics character controller.

Return Value

The walking speed value of the physics character controller.

Example

```
value = 0.0

function Init()
    value  = GetCharacterControllerWalkSpeed()

    message = string.format("\nCharacter Controller Walk Speed is : %.2f" ,value)
    PrintConsole(message)
end

function Update()

end
```

First we get the walking speed value of the physics character controller. Then we display the result in the console using the `PrintConsole` function.

4.72. GetCursorX

Definition

```
double GetCursorX()
```

Description

Returns the value of the X component of the mouse cursor position.

Return Value

The value of the X component of the mouse cursor position.

Example

```
x = 0.0

function Init()
    x = GetCursorX()
end

function Update()

end
```

4.73. GetCursorY

Definition

```
double GetCursorY()
```

Description

Returns the value of the Y component of the mouse cursor position.

Return Value

The value of the Y component of the mouse cursor position.

Example

```
y = 0.0

function Init()
    y = GetCursorY()
end

function Update()

end
```

4.74. GetDepthOfFieldFocalDistance

Definition

```
double GetDepthOfFieldFocalDistance()
```

Description

This function returns the focal distance of depth of field effect.

Return Value

Focal distance of depth of field effect.

Example

```
distance = 0.0

function Init()
    distance  = GetDepthOfFieldFocalDistance()

    message = string.format("\nDepth of field focal distance is : %.2f" ,distance)
    PrintConsole(message)
end

function Update()

end
```

In this example, the `GetDepthOfFieldFocalDistance` function returns the focal distance of depth of field effect. Then distance value is displayed on the console by the `PrintConsole` function.

4.75. GetDepthOfFieldFocalRange

Definition

```
double GetDepthOfFieldFocalRange()
```

Description

This function returns the focal range of depth of field effect.

Return Value

Focal range of depth of field effect.

Example

```
range = 0.0

function Init()
    range  = GetDepthOfFieldFocalRange()

    message = string.format("\nDepth of field focal range is : %.2f" ,range)
    PrintConsole(message)
end

function Update()

end
```

In this example, the `GetDepthOfFieldFocalRange` function returns the focal range of depth of field effect. Then focal range value is displayed on the console by the `PrintConsole` function.

4.76. GetDirectionalShadowAlgorithm

Definition

```
string GetDirectionalShadowAlgorithm()
```

Description

This function returns the algorithm name of directional light shadow.

Return Value

Algorithm name of directional light shadow. The return values are:

- `"SHADOW_SINGLE_HL"`
- `"SHADOW_SINGLE"`
- `"SHADOW_MULTI_LEAK"`
- `"SHADOW_MULTI_NOLEAK"`
- `"SHADOW_PCF"`
- `"SHADOW_PCF_TRILIN"`
- `"SHADOW_PCF_4TAP"`
- `"SHADOW_PCF_8TAP"`
- `"SHADOW_PCF_GAUSSIAN"`

Example

```
value = ""

function Init()
    value = GetDirectionalShadowAlgorithm()

    message = string.format("\nDirectional shadow algorithm is : %s" ,value)
    PrintConsole(message)
end

function Update()

end
```

First we get the algorithm name of direcional light shadow. Then we display the result in the console using the `PrintConsole` function.

4.77. GetDirectionalShadowFarClipPlane

Definition

```
double GetDirectionalShadowFarClipPlane()
```

Description

This function returns the far clip plane value of directional light shadow.

Return Value

The far clip plane value of directional light shadow. This value is greater than zero.

Example

```
value = 0.0

function Init()
    value = GetDirectionalShadowFarClipPlane()

    message = string.format("\nDirectional shadow far clip plane is : %.2f" ,value)
    PrintConsole(message)
end

function Update()

end
```

First we get the far clip plane value of direcional light shadow. Then we display the result in the console using the `PrintConsole` function.

4.78. GetDirectionalShadowIntensity

Definition

```
double GetDirectionalShadowIntensity()
```

Description

This function returns the intensity value of directional light shadow.

Return Value

The intensity value of directional light shadow. This value is in the range [0.0,1.0].

Example

```
value = 0.0

function Init()
    value = GetDirectionalShadowIntensity()

    message = string.format("\nDirectional shadow intensity is : %.2f" ,value)
    PrintConsole(message)
end

function Update()

end
```

First we get the intensity value of direcional light shadow. Then we display the result in the console using the `PrintConsole` function.

4.79. GetDirectionalShadowLight

Definition

```
string GetDirectionalShadowLight()
```

Description

This function returns the directional light name that casts the shadows. It should be noted that only one directional light in current VScene can cast the shadows.

Return Value

The directional light name that casts the shadows.

Example

```
value = ""

function Init()
    value = GetDirectionalShadowLight()

    message = string.format("\nDirectional shadow light name is : %s" ,value)
    PrintConsole(message)
end

function Update()

end
```

First we get the directional light name that can cast the shadow. Then we display the result in the console using the `PrintConsole` function.

4.80. GetDirectionalShadowNearClipPlane

Definition

```
double GetDirectionalShadowNearClipPlane()
```

Description

This function returns the the near clip plane value of directional light shadow.

Return Value

The near clip plane of directional light shadow. This value is greater than zero.

Example

```
value = 0.0

function Init()
    value = GetDirectionalShadowNearClipPlane()

    message = string.format("\nDirectional shadow near clip plane is : %.2f" ,value)
    PrintConsole(message)
end

function Update()

end
```

First we get the near clip plane value of direcional light shadow. Then we display the result in the console using the `PrintConsole` function.

4.81. GetDirectionalShadowNumberOfSplits

Definition

```
int GetDirectionalShadowNumberOfSplits()
```

Description

This function returns the number of splits of directional light shadow.

Return Value

The number of splits of directional light shadow. Return values are 1, 2, 3 or 4.

Example

```
value = 0

function Init()
    value = GetDirectionalShadowNumberOfSplits()

    message = string.format("\nDirectional shadow number of splits is : %d" ,value)
    PrintConsole(message)
end

function Update()

end
```

First we get the number of splits of direcional light shadow. Then we display the result in the console using the `PrintConsole` function.

4.82. GetDirectionalShadowResolution

Definition

```
int GetDirectionalShadowResolution()
```

Description

This function returns the resolution of directional light shadow.

Return Value

The resolution of directional light shadow. Return values are 1024, 2048 or 4096.

Example

```
value = 0

function Init()
    value = GetDirectionalShadowResolution()

    message = string.format("\nDirectional shadow resolution is : %d" ,value)
    PrintConsole(message)
end

function Update()

end
```

First we get the resolution value of direcional light shadow. Then we display the result in the console using the `PrintConsole` function.

4.83. GetDirectionalShadowWeightOfSplits

Definition

```
double GetDirectionalShadowWeightOfSplits()
```

Description

This function returns the weight of splits value of directional light shadow.

Return Value

The weight of splits of directional light shadow. This value is in the range [0.0,1.0].

Example

```
value = 0.0

function Init()
    value = GetDirectionalShadowWeightOfSplits()

    message = string.format("\nDirectional shadow weight of splits is : %.2f" ,value)
    PrintConsole(message)
end

function Update()

end
```

First we get the weight of splits of direcional light shadow. Then we display the result in the console using the `PrintConsole` function.

4.84. GetDistanceBetweenPhysicsCameraAndCharacterController

Definition

```
double GetDistanceBetweenPhysicsCameraAndCharacterController()
```

Description

This function returns the distance between physics character controller and third person physics camera attached to the physics character controller.

Return Value

Distance between physics character controller and third person physics camera attached to the physics character controller.

Example

```
distance = 0.0

function Init()
    distance  = GetDistanceBetweenPhysicsCameraAndCharacterController()

    message = string.format("\nDistance between physics camera and main character is :
%.2f" ,distance)
    PrintConsole(message)
end

function Update()

end
```

In this example, the `GetDistanceBetweenPhysicsCameraAndCharacterController` function returns the distance between physics character controller and third person physics camera attached to the physics character controller. Then distance value is displayed on the console by the `PrintConsole` function.

4.85. GetDistanceOfPrefabInstanceFromPhysicsCamera

Definition

```
double GetDistanceOfPrefabInstanceFromPhysicsCamera(string
prefabInstanceName)
```

Description

Returns the distance of the prefab instance `prefabInstanceName` from the physics camera attached to the main game character.

Parameters

prefabInstanceName

Specifies the name of the prefab instance. You can also use the name "this" for this parameter. In this case, "this" refers to the prefab instance that this script is attached to.

Return Value

Distance of the prefab instance `prefabInstanceName` from the physics camera attached to the main game character.

Example 1

```
distance = 0.0

function Init()
    distance = GetDistanceOfPrefabInstanceFromPhysicsCamera("1_VandaEngine17-
SamplePack1_well")

    message = string.format("\nDistance is > %.2f" , distance )
    PrintConsole(message)
end

function Update()

end
```

Returns the distance of prefab instance `"1_VandaEngine17-SamplePack1_well"` from the physics camera attached to the main game character. Then we print the return value of this function using `PrintConsole` function.

Example 2

```
--name of script is GetDistanceOfPrefabInstanceFromPhysicsCamera2.lua

distance = 0.0

function Init()
    distance = GetDistanceOfPrefabInstanceFromPhysicsCamera("this")

    message = string.format("\nDistance is > %.2f" , distance )
    PrintConsole(message)
end
```

```
function Update()

end
```

If, in the Prefab Editor, you attach GetDistanceOfPrefabInstanceFromPhysicsCamera2.lua script to a Prefab, then "this" parameter in the GetDistanceOfPrefabInstanceFromPhysicsCamera function will point to instances of that Prefab in current VScene. For example, if you have an Instance named *instance1_a* from a Prefab named *a* to which this script is attached, "this" in GetDistanceOfPrefabInstanceFromPhysicsCamera function refers to the name *instance1_a*.
In this case, GetDistanceOfPrefabInstanceFromPhysicsCamera function returns the distance of prefab instance *instance1_a* from the physics camera attached to the main game character. Then we print the return value of this function using PrintConsole function.

4.86. GetElapsedTime

Definition

```
double GetElapsedTime()
```

Description

This function returns the elapsed time from the previous frame to the current frame in seconds.

Return Value

Elapsed time from the previous frame to the current frame in seconds

Example 1

```
elapsedTime = 0.0

function Init()

end

function Update()
    elapsedTime = GetElapsedTime()

    message = string.format("\nElapsed time is > %.2f" , elapsedTime)
    PrintConsole(message)
end
```

The result on my system is:

```
...
1598> Elapsed time is > 0.03
1599> Elapsed time is > 0.02
1600> Elapsed time is > 0.02
1601> Elapsed time is > 0.02
1602> Elapsed time is > 0.02
1603> Elapsed time is > 0.02
1604> Elapsed time is > 0.02
1605> Elapsed time is > 0.01
1606> Elapsed time is > 0.02
1607> Elapsed time is > 0.02
1608> Elapsed time is > 0.02
1609> Elapsed time is > 0.01
1610> Elapsed time is > 0.02
1611> Elapsed time is > 0.02
1612> Elapsed time is > 0.01
1613> Elapsed time is > 0.02
1614> Elapsed time is > 0.01
1615> Elapsed time is > 0.02
1616> Elapsed time is > 0.02
1617> Elapsed time is > 0.02
1618> Elapsed time is > 0.02
1619> Elapsed time is > 0.02
1620> Elapsed time is > 0.02
```

```
1621> Elapsed time is > 0.02
1622> Elapsed time is > 0.02
1623> Elapsed time is > 0.02
1624> Elapsed time is > 0.01
1625> Elapsed time is > 0.01
1626> Elapsed time is > 0.01
…
```

Example 2

```
elapsedSeconds = 0.0

function Init()

end

function Update()
    elapsedSeconds = elapsedSeconds + GetElapsedTime()

    message = string.format("\nElapsed seconds is > %.2f" , elapsedSeconds)
    PrintConsole(message)
end
```

The result on my system is:

```
...
1275> Elapsed seconds is > 0.03
1276> Elapsed seconds is > 0.04
1277> Elapsed seconds is > 0.05
1278> Elapsed seconds is > 0.07
1279> Elapsed seconds is > 0.08
1280> Elapsed seconds is > 0.10
1281> Elapsed seconds is > 0.12
1282> Elapsed seconds is > 0.13
1283> Elapsed seconds is > 0.15
1284> Elapsed seconds is > 0.16
1285> Elapsed seconds is > 0.18
1286> Elapsed seconds is > 0.19
1287> Elapsed seconds is > 0.21
1288> Elapsed seconds is > 0.22
1289> Elapsed seconds is > 0.24
1290> Elapsed seconds is > 0.25
1291> Elapsed seconds is > 0.27
1292> Elapsed seconds is > 0.29
1293> Elapsed seconds is > 0.30
...
```

4.87. GetEngineCameraAngle

Definition

`double GetEngineCameraAngle(string engineCameraName)`

Description

This function returns the angle of the **`engineCameraName`** engine camera.

Parameters

`engineCameraName`

Specifies the name of the engine camera. You can also use the name "this" for this parameter. In this case, "this" refers to the camera object that this script is attached to.

Return Value

Angle of the **`engineCameraName`** engine camera.

Example 1

```lua
angle = 0.0

function Init()
    angle = GetEngineCameraAngle("camera1")

    message = string.format("\nAngle is > %.2f" , angle )
    PrintConsole(message)
end

function Update()

end
```

Returns the angle of the `"camera1"` engine camera.

Example 2

```lua
--Name of script is GetEngineCameraAngle2.lua
angle = 0.0

function Init()
    angle = GetEngineCameraAngle("this")

    message = string.format("\nAngle is > %.2f" , angle )
    PrintConsole(message)
end

function Update()

end
```

In this case, `"this"` string in the `GetEngineCameraAngle` points to the camera that `GetEngineCameraAngle2.lua` script is attached to. For example, if `GetEngineCameraAngle2.lua` script is attached to a engine camera named "camera1", `"this"` will be equivalent to the name "camera1". In this example, `GetEngineCameraAngle` function returns the angle of current engine camera.

4.88. GetEngineCameraFarClipPlane

Definition

```
double GetEngineCameraFarClipPlane(string engineCameraName)
```

Description

This function returns the far clip plane of the **engineCameraName** engine camera.

Parameters

engineCameraName

Specifies the name of the engine camera. You can also use the name "this" for this parameter. In this case, "this" refers to the camera object that this script is attached to.

Return Value

Far clip plane of the **engineCameraName** engine camera.

Example 1

```
fcp = 0.0

function Init()
    fcp = GetEngineCameraFarClipPlane("camera1")

    message = string.format("\nFar Clip Plane is > %.2f" , fcp)
    PrintConsole(message)
end

function Update()

end
```

Returns the far clip plane of the `"camera1"` engine camera.

Example 2

```
--Name of script is GetEngineCameraFarClipPlane2.lua
fcp = 0.0

function Init()
    fcp = GetEngineCameraFarClipPlane("this")

    message = string.format("\nFar Clip Plane is > %.2f" , fcp)
    PrintConsole(message)
end

function Update()

end
```

In this case, `"this"` string in the `GetEngineCameraFarClipPlane` points to the camera that `GetEngineCameraFarClipPlane2.lua` script is attached to. For example, if `GetEngineCameraFarClipPlane2.lua` script is attached to a engine camera named "camera1", `"this"` will be equivalent to the name "camera1". In this example, `GetEngineCameraFarClipPlane` function returns the far clip plane of current engine camera.

4.89. GetEngineCameraNearClipPlane

Definition

```
double GetEngineCameraNearClipPlane(string engineCameraName)
```

Description

This function returns the near clip plane of the `engineCameraName` engine camera.

Parameters

engineCameraName

Specifies the name of the engine camera. You can also use the name "this" for this parameter. In this case, "this" refers to the camera object that this script is attached to.

Return Value

Near clip plane of the `engineCameraName` engine camera.

Example 1

```
ncp = 0.0

function Init()
    ncp = GetEngineCameraNearClipPlane("camera1")

    message = string.format("\nNear Clip Plane is > %.2f" , ncp)
    PrintConsole(message)
end

function Update()

end
```

Returns the near clip plane of the `"camera1"` engine camera.

Example 2

```
--Name of script is GetEngineCameraNearClipPlane2.lua
ncp = 0.0

function Init()
    ncp = GetEngineCameraNearClipPlane("this")

    message = string.format("\nNear Clip Plane is > %.2f" , ncp)
    PrintConsole(message)
end

function Update()

end
```

In this case, `"this"` string in the `GetEngineCameraNearClipPlane` points to the camera that `GetEngineCameraNearClipPlane2.lua` script is attached to. For example, if `GetEngineCameraNearClipPlane2.lua` script is attached to a engine camera named "camera1", `"this"` will be equivalent to the name "camera1". In this example, `GetEngineCameraNearClipPlane` function returns the near clip plane of current engine camera.

4.90. GetEngineCameraPan

Definition

`double GetEngineCameraPan(string engineCameraName)`

Description

This function returns the pan of the **`engineCameraName`** engine camera.

Parameters

`engineCameraName`

Specifies the name of the engine camera. You can also use the name "this" for this parameter. In this case, "this" refers to the camera object that this script is attached to.

Return Value

Pan of the **`engineCameraName`** engine camera.

Example 1

```
pan = 0.0

function Init()
    pan = GetEngineCameraPan("camera1")

    message = string.format("\nPan is > %.2f" , pan )
    PrintConsole(message)
end

function Update()

end
```

Returns the pan of the `"camera1"` engine camera.

Example 2

```
--Name of script is GetEngineCameraPan2.lua

pan = 0.0

function Init()
    pan = GetEngineCameraPan("this")

    message = string.format("\nPan is > %.2f" , pan )
    PrintConsole(message)
end

function Update()

end
```

In this case, `"this"` string in the `GetEngineCameraPan` points to the camera that `GetEngineCameraPan2.lua` script is attached to. For example, if `GetEngineCameraPan2.lua` script is attached to a engine camera named "camera1", `"this"` will be equivalent to the name "camera1". In this example, `GetEngineCameraPan` function returns the pan of current engine camera.

4.91. GetEngineCameraPosition

Definition

```
double,double,double GetEngineCameraPosition(string engineCameraName)
```

Description

This function returns the 3D position of engine camera `engineCameraName`.

Parameters

engineCameraName

Specifies the name of the engine camera. You can also use the name "this" for this parameter. In this case, "this" refers to the camera object that this script is attached to.

Return Value

This function returns the 3D position of engine camera `engineCameraName` as three values x, y, z.

Example 1

```
pos_x = 0.0
pos_y = 0.0
pos_z = 0.0

function Init()
    pos_x, pos_y, pos_z = GetEngineCameraPosition("camera1")

    message = string.format("\nCamera position is > (%.2f, %.2f, %.2f)" , pos_x, pos_y,
pos_z)
    PrintConsole(message)
end

function Update()

end
```

Returns the 3D position of the `"camera1"` engine camera.

Example 2

```
--name of script is GetEngineCameraPosition2.lua

pos_x = 0.0
pos_y = 0.0
pos_z = 0.0

function Init()
    pos_x, pos_y, pos_z = GetEngineCameraPosition("this")

    message = string.format("\nCamera position is > (%.2f, %.2f, %.2f)" , pos_x, pos_y,
pos_z)
    PrintConsole(message)
end
```

```
function Update()

end
```

In this case, `"this"` string in the `GetEngineCameraPosition` points to the engine camera that `GetEngineCameraPosition2.lua` script is attached to. For example, if `GetEngineCameraPosition2.lua` script is attached to a engine camera named "camera1", `"this"` will be equivalent to the name "camera1". In this example, `GetEngineCameraPosition` function returns the 3D position of current engine camera.

4.92. GetEngineCameraTilt

Definition

```
double GetEngineCameraTilt(string engineCameraName)
```

Description

This function returns the tilt of the **engineCameraName** engine camera.

Parameters

engineCameraName

Specifies the name of the engine camera. You can also use the name "this" for this parameter. In this case, "this" refers to the camera object that this script is attached to.

Return Value

Tilt of the **engineCameraName** engine camera.

Example 1

```
tilt = 0.0

function Init()
    tilt = GetEngineCameraTilt("camera1")

    message = string.format("\nTilt is > %.2f" , tilt )
    PrintConsole(message)
end

function Update()

end
```

Returns the tilt of the "camera1" engine camera.

Example 2

```
--Name of script is GetEngineCameraTilt2.lua

tilt = 0.0

function Init()
    tilt = GetEngineCameraTilt("this")

    message = string.format("\nTilt is > %.2f" , tilt )
    PrintConsole(message)
end

function Update()

end
```

In this case, "this" string in the GetEngineCameraTilt points to the camera that GetEngineCameraTilt2.lua script is attached to. For example, if

`GetEngineCameraTilt2.lua` script is attached to a engine camera named "camera1", `"this"` will be equivalent to the name "camera1". In this example, `GetEngineCameraTilt` function returns the tilt of current engine camera.

4.93. GetFogColor

Definition

```
double, double, double GetFogColor()
```

Description

This function returns the fog color as three values of red, green and blue.

Return Value

Fog color as three values of red, green and blue. Each value ranges from 0.0 to 1.0.

Example

```
red = 0.0
green = 0.0
blue = 0.0

function Init()
    red, green, blue = GetFogColor()

    message = string.format("\nFog color is : (%.2f, %.2f, %.2f)" , red, green, blue)
    PrintConsole(message)
end

function Update()

end
```

In this example, the `GetFogColor` function returns the value of the red, green, and blue components of the fog color. Then these three values are displayed on the console by the `PrintConsole` function.

4.94. GetFogDensity

Definition

```
double GetFogDensity()
```

Description

This function returns the fog density.

Return Value

Fog density.

Example

```
density = 0.0

function Init()
    density  = GetFogDensity()

    message = string.format("\nFog density is : %.2f" ,density)
    PrintConsole(message)
end

function Update()

end
```

In this example, the `GetFogDensity` function returns the fog density. Then density value is displayed on the console by the `PrintConsole` function.

4.95. GetGlobalSoundVolume

Definition

```
double GetGlobalSoundVolume()
```

Description

This function returns the global sound volume.

Return Value

Global sound volume.

Example

```
volume = 0.0

function Init()
    volume = GetGlobalSoundVolume()

    message = string.format("\nGlobal sound volume is > %.2f" , volume)
    PrintConsole(message)
end

function Update()

end
```

4.96. GetGUIButtonPosition

Definition

`int,int GetGUIButtonPosition(string GUIName, string buttonName)`

Description

This function returns the two-dimensional position of the button **`buttonName`** of GUI **`GUIName`** relative to the lower left part of the screen as two x, y values. You can view and copy the name of the GUIs and their buttons in the *Script Utility* section (Tools > Script Editor > Tools > Script Utility) or the *Prefabs and GUIs* section of the current VScene.

Parameters

`GUIName`
The name of the GUI to which the **`buttonName`** button belongs.

`buttonName`
The name of the button that belongs to **`GUIName`**.

Return Value

Two-dimensional position of the button **`buttonName`** of GUI **`GUIName`** relative to the lower left part of the screen as two x, y values.

Example

```
x = 0
y = 0
function OnSelectMouseLButtonDown()
    x,y = GetGUIButtonPosition("gui_test_test", "PlayGame")

    message = string.format("\nGUI button position is > %d, %d" , x,y)
    PrintConsole(message)
end

function OnSelectMouseRButtonDown()

end

function OnSelectMouseEnter()

end
```

Assume that this script is attached to a button named *ShowPosition* that belongs to a GUI named *gui_position*. In this case, whenever you left click on the *ShowPosition* button, the `GetGUIButtonPosition` function returns the 2D position of the `"PlayGame"` button from the GUI named `"gui_test_test"`. This script then displays the **`x`** and **`y`** positions on the console.

4.97. GetGUIButtonScriptBoolVariable

Definition

`bool GetGUIButtonScriptBoolVariable(string GUIName, string buttonName, string variable)`

Description

This function gets the value of the Boolean **`variable`** defined in the script attached to the **`buttonName`** button that belongs to **`GUIName`** GUI.

Parameters

`GUIName`
Specifies the the name of the GUI to which the **`buttonName`** button belongs.

`buttonName`
Specifies the the name of the button that belongs to **`GUIName`** GUI.

`variable`
Specifies the name of the Boolean variable defined in the script attached to the **`buttonName`** button.

Return Value

Returns the value of the Boolean **`variable`** defined in the script attached to the **`buttonName`** button that belongs to **`GUIName`** GUI.

Example

```lua
--script name is GetGUIButtonScriptBoolVariable.lua attached a to game object such as water
value = false
function Init()
    value = GetGUIButtonScriptBoolVariable("gui_pack1_button", "PlayGame", "a")
end

function Update()

end
```

Assuming that the variable `"a"` is defined with the value *true* in the script attached to the button object `"PlayGame"` that belongs to `"gui_pack1_button"` GUI , `GetGUIButtonScriptBoolVariable` function returns the value *true*.

4.98. GetGUIButtonScriptDoubleVariable

Definition

`double GetGUIButtonScriptDoubleVariable(string GUIName, string buttonName, string variable)`

Description

This function gets the value of the Double **`variable`** defined in the script attached to the **`buttonName`** button that belongs to **`GUIName`** GUI.

Parameters

`GUIName`
Specifies the the name of the GUI to which the **`buttonName`** button belongs.

`buttonName`
Specifies the the name of the button that belongs to **`GUIName`** GUI.

`variable`
Specifies the name of the Double variable defined in the script attached to the **`buttonName`** button.

Return Value

Returns the value of the Double **`variable`** defined in the script attached to the **`buttonName`** button that belongs to **`GUIName`** GUI.

Example

```lua
--script name is GetGUIButtonScriptDoubleVariable.lua attached a to game object such as
water
value = 0.0
function Init()
    value = GetGUIButtonScriptDoubleVariable("gui_pack1_button", "PlayGame", "a")
end

function Update()

end
```

Assuming that the variable `"a"` is defined with the value *1.0* in the script attached to the button object `"PlayGame"` that belongs to `"gui_pack1_button"` GUI , `GetGUIButtonScriptDoubleVariable` function returns the value *1.0*.

4.99. GetGUIButtonScriptIntVariable

Definition

```
int GetGUIButtonScriptIntVariable(string GUIName, string buttonName,
string variable)
```

Description

This function gets the value of the Integer **`variable`** defined in the script attached to the **`buttonName`** button that belongs to **`GUIName`** GUI.

Parameters

`GUIName`

Specifies the the name of the GUI to which the **`buttonName`** button belongs.

`buttonName`

Specifies the the name of the button that belongs to **`GUIName`** GUI.

`variable`

Specifies the name of the Integer variable defined in the script attached to the **`buttonName`** button.

Return Value

Returns the value of the Integer **`variable`** defined in the script attached to the **`buttonName`** button that belongs to **`GUIName`** GUI.

Example

```
--script name is GetGUIButtonScriptIntVariable.lua attached a to game object such as
water
value = 0
function Init()
    value = GetGUIButtonScriptIntVariable("gui_pack1_button", "PlayGame", "a")
end

function Update()

end
```

Assuming that the variable `"a"` is defined with the value *1* in the script attached to the button object `"PlayGame"` that belongs to `"gui_pack1_button"` GUI , `GetGUIButtonScriptIntVariable` function returns the value *1*.

4.100. GetGUIButtonScriptStringVariable

Definition

`string GetGUIButtonScriptStringVariable(string GUIName, string buttonName, string variable)`

Description

This function gets the value of the String **`variable`** defined in the script attached to the **`buttonName`** button that belongs to **`GUIName`** GUI.

Parameters

`GUIName`
Specifies the the name of the GUI to which the **`buttonName`** button belongs.

`buttonName`
Specifies the the name of the button that belongs to **`GUIName`** GUI.

`variable`
Specifies the name of the String variable defined in the script attached to the **`buttonName`** button.

Return Value

Returns the value of the String **`variable`** defined in the script attached to the **`buttonName`** button that belongs to **`GUIName`** GUI.

Example

```
--script name is GetGUIButtonScriptStringVariable.lua attached a to game object such as water
value = ""
function Init()
    value = GetGUIButtonScriptStringVariable("gui_pack1_button", "PlayGame", "a")
end

function Update()

end
```

Assuming that the variable **`"a"`** is defined with the value "*hello*" in the script attached to the button object **`"PlayGame"`** that belongs to **`"gui_pack1_button"`** GUI , `GetGUIButtonScriptStringVariable` function returns the value "*hello*".

4.101. GetGUIImagePosition

Definition

```
int,int GetGUIImagePosition(string GUIName, string imageName)
```

Description

This function returns the two-dimensional position of the image **`imageName`** of GUI **`GUIName`** relative to the lower left part of the screen as two x, y values. You can view and copy the name of the GUIs and their images in the *Script Utility* section (Tools > Script Editor > Tools > Script Utility) or the *Prefabs and GUIs* section of the current VScene.

Parameters

`GUIName`

Specifies the name of the GUI to which the **`imageName`** image belongs.

`imageName`

Specifies the the name of the image that belongs to **`GUIName`**.

Return Value

Two-dimensional position of the image **`imageName`** of GUI **`GUIName`** relative to the lower left part of the screen as two x, y values.

Example

```
x = 0
y = 0
function OnSelectMouseLButtonDown()
    x,y = GetGUIImagePosition("gui_test_test", "image1")

    message = string.format("\nGUI image position is > %d, %d" , x,y)
    PrintConsole(message)
end

function OnSelectMouseRButtonDown()

end

function OnSelectMouseEnter()

end
```

Assume that this script is attached to a button named *ShowPosition* that belongs to a GUI named *gui_position*. In this case, whenever you left click on the *ShowPosition* button, the `GetGUIImagePosition` function returns the 2D position of the `"image1"` image from the GUI named `"gui_test_test"`. This script then displays the `x` and `y` positions on the console.

4.102. GetGUIPosition

Definition

```
int,int GetGUIPosition(string GUIName)
```

Description

This function returns the X and Y of the GUI **GUIName** as a percentage of the screen width and height. You can view and copy the name of the GUIs in the *Script Utility* section (Tools > Script Editor > Tools > Script Utility) or the *Prefabs and GUIs* section of the current VScene.

Parameters

GUIName

Specifies the name of the GUI.

Return Value

Returns the X and Y of the GUIName as a percentage of the screen width and height. Each of these two values are in the range [-100, 100]. It should be noted that the width of the GUI ranges from -(screen width) to (screen width), or -100 to 100 percents, and the height of the GUI ranges from -(screen height) to (screen height), or -100 to 100 percents. So -100 means (-screen width) or (-screen height) and 100 means (screen width) or (screen height).

Example

```
x = 0
y = 0

function Init()
    x,y = GetGUIPosition("gui_SampleGUI17_MainMenu")

    message = string.format("\nGUI position is > %d, %d" , x,y)
    PrintConsole(message)
end

function Update()

end
```

In this example, `GetGUIPosition` returns the X and Y values as percentages of the screen width and height. For example assume that it returns -5 and 10 percents of the screen width and height, respectively. Also Assume that the width and height of the screen are equal to 1024 and 768 respectively. In this case, these numbers will be equal to (-5 * 1024 / 100 = -51.2) and (10 * 768 / 100 = 76.8) respectively, in screen coordinates.

4.103. GetGUITextPosition

Definition

```
int,int GetGUITextPosition(string GUIName, string textName)
```

Description

This function returns the two-dimensional position of the text **`textName`** of GUI **`GUIName`** relative to the lower left part of the screen as two x, y values. You can view and copy the name of the GUIs and their texts in the *Script Utility* section (Tools > Script Editor > Tools > Script Utility) or the *Prefabs and GUIs* section of the current VScene.

Parameters

`GUIName`
Specifies the name of the GUI to which the **`textName`** text belongs.

`textName`
Specifies the the name of the text that belongs to **`GUIName`**.

Return Value

Two-dimensional position of the text **`textName`** of GUI **`GUIName`** relative to the lower left part of the screen as two x, y values.

Example

```
x = 0
y = 0
function OnSelectMouseLButtonDown()
    x,y = GetGUITextPosition("gui_test_test", "text1")

    message = string.format("\nGUI text position is > %d, %d" , x,y)
    PrintConsole(message)
end

function OnSelectMouseRButtonDown()

end

function OnSelectMouseEnter()

end
```

Assume that this script is attached to a button named *ShowPosition* that belongs to a GUI named *gui_position*. In this case, whenever you left click on the *ShowPosition* button, the `GetGUITextPosition` function returns the 2D position of the `"text1"` text from the GUI named `"gui_test_test"`. This script then displays the `x` and `y` positions on the console.

4.104. GetLightAmbient

Definition

```
double,double,double GetLightAmbient(string lightObjectName)
```

Description

This function returns the ambient color of `lightObjectName` light as three values of red, green and blue. Each value ranges from 0 to 1.

Parameters

lightObjectName

Specifies the name of the light object. You can also use the name "this" for this parameter. In this case, "this" refers to the light object name to which this script is attached.

Return Value

Returns the ambient color of `lightObjectName` light as three values of red, green and blue. Each value ranges from 0 to 1.

Example 1

```
red = 0.0
green = 0.0
blue = 0.0

function Init()
    red, green, blue = GetLightAmbient("light1")

    message = string.format("\nLight ambient color is > (%.2f, %.2f, %.2f)" , red, green,
blue)
    PrintConsole(message)
end

function Update()

end
```

In this example, the `GetLightAmbient` function returns the value of the red, green, and blue components of the ambient color of light `"light1"`. Then these three values are displayed on the console by the `PrintConsole` function.

Example 2

```
--Script name is GetLightAmbient2.lua

red = 0.0
green = 0.0
blue = 0.0

function Init()
    red, green, blue = GetLightAmbient("this")
    message = string.format("\nLight ambient color is > (%.2f, %.2f, %.2f)" , red, green,
blue)
    PrintConsole(message)
end

function Update()

end
```

Assume that the above script named `GetLightAmbient2.lua` is attached to the light object named "light1". In this case, string `"this"` in the `GetLightAmbient` function will be equal to "light1". In our example, the function `GetLightAmbient` returns three values of red, green and blue ambient color of the light "light1".

4.105. GetLightDiffuse

Definition

`double,double,double GetLightDiffuse(string lightObjectName)`

Description

This function returns the diffuse color of `lightObjectName` light as three values of red, green and blue. Each value ranges from 0 to 1.

Parameters

`lightObjectName`

Specifies the name of the light object. You can also use the name "this" for this parameter. In this case, "this" refers to the light object name to which this script is attached.

Return Value

Returns the diffuse color of `lightObjectName` light as three values of red, green and blue. Each value ranges from 0 to 1.

Example 1

```
red = 0.0
green = 0.0
blue = 0.0

function Init()
    red, green, blue = GetLightDiffuse("light1")

    message = string.format("\nLight diffuse color is > (%.2f, %.2f, %.2f)" , red, green,
blue)
    PrintConsole(message)
end

function Update()

end
```

In this example, the `GetLightDiffuse` function returns the value of the red, green, and blue components of the diffuse color of light `"light1"`. Then these three values are displayed on the console by the `PrintConsole` function.

Example 2

```
--Script name is GetLightDiffuse2.lua

red = 0.0
green = 0.0
blue = 0.0

function Init()
    red, green, blue = GetLightDiffuse("this")
    message = string.format("\nLight diffuse color is > (%.2f, %.2f, %.2f)" , red, green,
blue)
    PrintConsole(message)
end

function Update()

end
```

Assume that the above script named `GetLightDiffuse2.lua` is attached to the light object named "light1". In this case, string `"this"` in the `GetLightDiffuse` function will be equal to "light1". In our example, the function `GetLightDiffuse` returns three values of red, green and blue diffuse color of the light "light1".

4.106. GetLightScriptBoolVariable

Definition

`bool GetLightScriptBoolVariable(string lightName, string variable)`

Description

This function gets the value of the Boolean **`variable`** defined in the script attached to the **`lightName`** light object.

Parameters

`lightName`
Specifies the name of the light object.

`variable`
Specifies the name of the Boolean variable defined in the script attached to the **`lightName`** light.

Return Value

Returns the value of the Boolean **`variable`** defined in the script attached to the **`lightName`** light object.

Example

```lua
--script name is GetLightScriptBoolVariable.lua attached a to game object such as water
value = false
function Init()
    value = GetLightScriptBoolVariable("light1", "a")
end

function Update()

end
```

Assuming that the variable `"a"` is defined with the value *true* in the script attached to the light object `"light1"`, `GetLightScriptBoolVariable` function returns the value *true*.

4.107. GetLightScriptDoubleVariable

Definition

`double GetLightScriptDoubleVariable(string lightName, string variable)`

Description

This function gets the value of the Double **`variable`** defined in the script attached to the **`lightName`** light object.

Parameters

`lightName`
Specifies the name of the light object.

`variable`
Specifies the name of the Double variable defined in the script attached to the **`lightName`** light.

Return Value

Returns the value of the Double **`variable`** defined in the script attached to the **`lightName`** light object.

Example

```lua
--script name is GetLightScriptDoubleVariable.lua attached a to game object such as water
return_value = 0.0

function Init()
    return_value = GetLightScriptDoubleVariable("light1", "a")
end

function Update()

end
```

Assuming that the variable **`"a"`** is defined with the value *1.0* in the script attached to the light object **`"light1"`**, **`GetLightScriptDoubleVariable`** function returns the value *1.0*.

4.108. GetLightScriptIntVariable

Definition

`int GetLightScriptIntVariable(string lightName, string variable)`

Description

This function gets the value of the Integer **`variable`** defined in the script attached to the **`lightName`** light object.

Parameters

`lightName`
Specifies the name of the light object.

`variable`
Specifies the name of the Integer variable defined in the script attached to the **`lightName`** light.

Return Value

Returns the value of the Integer **`variable`** defined in the script attached to the **`lightName`** light object.

Example

```lua
--script name is GetLightScriptIntVariable.lua attached a to game object such as water
return_value = 0

function Init()
    return_value = GetLightScriptIntVariable("light1", "a")
end

function Update()

end
```

Assuming that the variable `"a"` is defined with the value *1* in the script attached to the light object `"light1"`, `GetLightScriptIntVariable` function returns the value *1*.

4.109. GetLightScriptStringVariable

Definition

`string GetLightScriptStringVariable(string lightName, string variable)`

Description

This function gets the value of the String **`variable`** defined in the script attached to the **`lightName`** light object.

Parameters

`lightName`
Specifies the name of the light object.

`variable`
Specifies the name of the String variable defined in the script attached to the **`lightName`** light.

Return Value

Returns the value of the String **`variable`** defined in the script attached to the **`lightName`** light object.

Example

```lua
--script name is GetLightScriptStringVariable.lua attached a to game object such as water
return_value = ""

function Init()
    return_value = GetLightScriptStringVariable("light1", "a")
end

function Update()

end
```

Assuming that the variable **`"a"`** is defined with the value "*hello*" in the script attached to the light object **`"light1"`**, **`GetLightScriptStringVariable`** function returns the value "*hello*".

4.110. GetLightShininess

Definition

```
double GetLightShininess(string lightObjectName)
```

Description

This function returns the shininess of `lightObjectName` light object.

Parameters

`lightObjectName`

Specifies the name of the light object. You can also use the name "this" for this parameter. In this case, "this" refers to the light object name to which this script is attached.

Return Value

Returns the shininess of `lightObjectName` light.

Example 1

```
shininess = 0.0

function Init()
    shininess  = GetLightShininess("light1")

    message = string.format("\nLight shininess is > %.2f" ,shininess)
    PrintConsole(message)
end

function Update()

end
```

In this example, the `GetLightShininess` function returns the shininess value of of light `"light1"`. Then shininess value is displayed on the console by the `PrintConsole` function.

Example 2

```
--Script name is GetLightShininess2.lua

shininess = 0.0

function Init()
    shininess  = GetLightShininess("this")

    message = string.format("\nLight shininess is > %.2f" ,shininess)
    PrintConsole(message)
end

function Update()

end
```

Assume that the above script named `GetLightShininess2.lua` is attached to the light object named "light1". In this case, string `"this"` in the `GetLightShininess` function will be equal to "light1". In our example, the function `GetLightShininess` returns the shininess value of the light "light1".

4.111. GetLightSpecular

Definition

```
double,double,double GetLightSpecular(string lightObjectName)
```

Description

This function returns the specular color of `lightObjectName` light as three values of red, green and blue. Each value ranges from 0 to 1.

Parameters

`lightObjectName`

Specifies the name of the light object. You can also use the name "this" for this parameter. In this case, "this" refers to the light object name to which this script is attached.

Return Value

Returns the specular color of `lightObjectName` light as three values of red, green and blue. Each value ranges from 0 to 1.

Example 1

```
red = 0.0
green = 0.0
blue = 0.0

function Init()
    red, green, blue = GetLightSpecular("light1")

    message = string.format("\nLight specular color is > (%.2f, %.2f, %.2f)" , red,
green, blue)
    PrintConsole(message)
end

function Update()

end
```

In this example, the `GetLightSpecular` function returns the value of the red, green, and blue components of the specular color of light `"light1"`. Then these three values are displayed on the console by the `PrintConsole` function.

Example 2

```
--Script name is GetLightSpecular2.lua

red = 0.0
green = 0.0
blue = 0.0

function Init()
    red, green, blue = GetLightSpecular("this")
    message = string.format("\nLight specular color is > (%.2f, %.2f, %.2f)" , red,
green, blue)
    PrintConsole(message)
end

function Update()

end
```

Assume that the above script named `GetLightSpecular2.lua` is attached to the light object named "light1". In this case, string `"this"` in the `GetLightSpecular` function will be equal to "light1". In our example, the function `GetLightSpecular` returns three values of red, green and blue specular color of the light "light1".

4.112. GetMainCharacterScriptBoolVariable

Definition

`bool GetMainCharacterScriptBoolVariable(string variable)`

Description

This function gets the value of the Boolean **`variable`** defined in the script attached to the main character object.

Parameters

`variable`

Specifies the name of the Boolean variable defined in the script attached to the main character.

Return Value

Returns the value of the Boolean **`variable`** defined in the script attached to the main character object.

Example

```
--script name is GetMainCharacterScriptBoolVariable.lua attached a to game object such as
water
value = false
function Init()
    value = GetMainCharacterScriptBoolVariable("a")
end

function Update()

end
```

Assuming that the variable `"a"` is defined with the value *true* in the script attached to the main character object, `GetMainCharacterScriptBoolVariable` function returns the value *true*.

4.113. GetMainCharacterScriptDoubleVariable

Definition

```
double GetMainCharacterScriptDoubleVariable(string variable)
```

Description

This function gets the value of the Double **`variable`** defined in the script attached to the main character object.

Parameters

`variable`

Specifies the name of the Double variable defined in the script attached to the main character.

Return Value

Returns the value of the Double **`variable`** defined in the script attached to the main character object.

Example

```
--script name is GetMainCharacterScriptDoubleVariable.lua attached a to game object such as water
return_value = 0.0

function Init()
    return_value = GetMainCharacterScriptDoubleVariable("a")
end

function Update()

end
```

Assuming that the variable `"a"` is defined with the value *1.0* in the script attached to the main character object, `GetMainCharacterScriptDoubleVariable` function returns the value *1.0*.

4.114. GetMainCharacterScriptIntVariable

Definition

```
int GetMainCharacterScriptIntVariable(string variable)
```

Description

This function gets the value of the Integer **`variable`** defined in the script attached to the main character object.

Parameters

`variable`

Specifies the name of the Integer variable defined in the script attached to the main character.

Return Value

Returns the value of the Integer **`variable`** defined in the script attached to the main character object.

Example

```
--script name is GetMainCharacterScriptIntVariable.lua attached a to game object such as
water
return_value = 0

function Init()
    return_value = GetMainCharacterScriptIntVariable("a")
end

function Update()

end
```

Assuming that the variable `"a"` is defined with the value *1* in the script attached to the main character, `GetMainCharacterScriptIntVariable` function returns the value *1*.

4.115. GetMainCharacterScriptStringVariable

Definition

`string GetMainCharacterScriptStringVariable(string variable)`

Description

This function gets the value of the String **`variable`** defined in the script attached to the main character object.

Parameters

`variable`

Specifies the name of the String variable defined in the script attached to the main character.

Return Value

Returns the value of the String **`variable`** defined in the script attached to the main character object.

Example

```
--script name is GetMainCharacterScriptStringVariable.lua attached a to game object such
as water
return_value = ""

function Init()
    return_value = GetMainCharacterScriptStringVariable("a")
end

function Update()

end
```

Assuming that the variable **`"a"`** is defined with the value "*hello*" in the script attached to the main character object, `GetMainCharacterScriptStringVariable` function returns the value "*hello*".

4.116. GetMenuCursorSize

Definition

```
int GetMenuCursorSize()
```

Description

This function returns the menu cursor size as an integer number. You can set menu cursor size through Modify > Current VScene Properties menu or through SetMenuCursorSize function.

Return Value

Size of menu cursor.

Example

```
value = 0

function Init()
    value = GetMenuCursorSize()

    message = string.format("\nMenu cursor size is > %d" ,value )
    PrintConsole(message)
end

function Update()

end
```

4.117. GetMultisamplingValue

Definition

```
int GetMultisamplingValue()
```

Description

This function returns the value of multisampling.

Return Value

The value of multisampling.

Example

```
value = 0

function Init()
    value = GetMultisamplingValue()

    message = string.format("\nMultisampling value is > %d" ,value )
    PrintConsole(message)
end

function Update()

end
```

4.118. GetPhysicsActorGroup

Definition

```
string GetPhysicsActorGroup(string physicsActorName)
```

Description

This function receives the name of physics actor and returns its type as string.

Parameters

physicsActorName

Specifies the name of the physics actor belonging to the prefab instance.

Return Value

This function returns the type of physics actor as one of the following string values:

"KINEMATIC"

Kinematic is a dynamic actor that can ignore some rules of physics, and its rotation and translation is controlled by prefab instance.

"DYNAMIC"

A dynamic actor has its position and rotation updated by the physics simulation and controls the translation and rotation of its prefab instance.

"TRIGGER"

Triggers allow colliders to perform overlap tests.

"STATIC"

Static actor is immovable by the physics simulation.

"GROUND"

Default physics ground plane.

Example

```
function OnTriggerEnter(otherActorName)
    if GetPhysicsActorGroup(otherActorName) == "KINEMATIC" then
PrintConsole("\nKinematic Actor")
    elseif GetPhysicsActorGroup(otherActorName) == "DYNAMIC" then
PrintConsole("\nDynamic Actor")
    end
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)

end
```

Assume that this script is attached to a trigger. In this case, whenever a prefab instance that has a kinematic actor is entered into this trigger, a message titled `"Kinematic Actor"` will be displayed on the console. Otherwise, if the prefab instance that has a dynamic actor is entered into this trigger, a message titled `"Dynamic Actor"` will be displayed in the console.

4.119. GetPhysicsCameraAngle

Definition

```
double GetPhysicsCameraAngle()
```

Description

This function returns the angle of physics camera attached to the main character. You can set the physics camera angle through the Main Character Properties dialog (Insert > Main Character) or the *SetPhysicsCameraAngle* function.

Return Value

This function returns the angle of physics camera attached to the main character as a Double value in degrees.

Example

```
angle = 0.0

function Init()
    angle = GetPhysicsCameraAngle()

    message = string.format("\nPhysics camera angle is > %.2f" ,angle)
    PrintConsole(message)
end

function Update()

end
```

4.120. GetPhysicsCameraFarClipPlane

Definition

```
double GetPhysicsCameraFarClipPlane()
```

Description

This function returns the far clip plane value of physics camera attached to the main character.

Return Value

The far clip plane of of physics camera attached to the main character as a Double value. This value is greater than 0.0.

Example

```
fcp = 0.0

function Init()
    fcp = GetPhysicsCameraFarClipPlane()

    message = string.format("\nFCP is : %.2f" ,fcp)
    PrintConsole(message)
end

function Update()

end
```

First we get the far clip plane value of the physics camera attached to the main character. Then we display the result in the console using `PrintConsole` function.

4.121. GetPhysicsCameraMaxTilt

Definition

```
double GetPhysicsCameraMaxTilt()
```

Description

This function returns the maximum tilt of physics camera attached to the main character. You can set the maximum physics camera tilt through the Main Character Properties dialog (Insert > Main Character) or the *SetPhysicsCameraMaxTilt* function.

Return Value

This function returns the maximum tilt of physics camera attached to the main character as a Double value in degrees.

Example

```
maxTilt = 0.0

function Init()
    maxTilt = GetPhysicsCameraMaxTilt()

    message = string.format("\nPhysics camera max tilt is > %.2f" ,maxTilt)
    PrintConsole(message)
end

function Update()

end
```

4.122. GetPhysicsCameraMinTilt

Definition

```
double GetPhysicsCameraMinTilt()
```

Description

This function returns the minimum tilt of physics camera attached to the main character. You can set the minimum physics camera tilt through the Main Character Properties dialog (Insert > Main Character) or the *SetPhysicsCameraMinTilt* function.

Return Value

This function returns the minimum tilt of physics camera attached to the main character as a Double value in degrees.

Example

```
minTilt = 0.0

function Init()
    minTilt = GetPhysicsCameraMinTilt()

    message = string.format("\nPhysics camera min tilt is > %.2f" ,minTilt)
    PrintConsole(message)
end

function Update()

end
```

4.123. GetPhysicsCameraNearClipPlane

Definition

```
double GetPhysicsCameraNearClipPlane()
```

Description

This function returns the near clip plane value of physics camera attached to the main character.

Return Value

The near clip plane of of physics camera attached to the main character as a Double value. This value is greater than 0.0.

Example

```
ncp = 0.0

function Init()
    ncp = GetPhysicsCameraNearClipPlane()

    message = string.format("\nNCP is : %.2f" ,ncp)
    PrintConsole(message)
end

function Update()

end
```

First we get the near clip plane value of the physics camera attached to the main character. Then we display the result in the console using `PrintConsole` function.

4.124. GetPhysicsCameraTilt

Definition

```
double GetPhysicsCameraTilt()
```

Description

This function returns the current tilt value of the physics camera attached to the main character.

Return Value

This function returns the current tilt value of the physics camera attached to the main character as a Double value in degrees.

Example

```
tilt = 0.0

function Init()

end

function Update()
    tilt = GetPhysicsCameraTilt()

    message = string.format("\nPhysics camera tilt is > %.2f" ,tilt)
    PrintConsole(message)
end
```

4.125. GetPhysicsCameraYaw

Definition

```
double GetPhysicsCameraYaw()
```

Description

This function returns the current yaw value of the physics camera attached to the main character.

Return Value

This function returns the current yaw value of the physics camera attached to the main character as a Double value in degrees.

Example

```
yaw = 0.0

function Init()

end

function Update()
    yaw = GetPhysicsCameraYaw()

    message = string.format("\nPhysics camera yaw is > %.2f" ,yaw)
    PrintConsole(message)
end
```

4.126. GetPhysicsCollisionFlags

Definition

```
bool GetPhysicsCollisionFlags(string group1, string group2)
```

Description

Each physics actor in Vanda engine belongs to a specific group. For example, a dynamic physics actor belongs to the "DYNAMIC" group, while a static physics actor belongs to the "STATIC" group. This function returns true if collision detection between the given pair of groups is enabled at runtime, otherwise it returns false.
You can use the Tools > Current VScene Properties menu or `SetPhysicsCollisionFlags` function to enable/disable collision detection between physics actors belonging to a given pair of groups. Initially all pair of physics groups except (Trigger vs. Ground Plane) pair are enabled, meaning that collision detection happens between all physics actors except (Trigger vs. Ground Plane).

Parameters

group1
Specifies the first group. The following group types are supported:

"KINEMATIC"
Kinematic is a dynamic actor that can ignore some rules of physics, and its rotation and translation is controlled by prefab instance.

"DYNAMIC"
A dynamic actor has its position and rotation updated by the physics simulation and controls the translation and rotation of its prefab instance.

"TRIGGER"
Triggers allow colliders to perform overlap tests.

"STATIC"
Static actor is immovable by the physics simulation.

"GROUND"
Default physics ground plane.

group2
Specifies the second group. The supported groups are similar to the ***group1*** description.

Return Value

Return values are `true` or `false`. The `true` value means that collision detection between two physics actors a and b belonging to ***group1*** and ***group2*** occurs.

Example

```
flag = false
message = ""

function Init()
```

```
    flag  = GetPhysicsCollisionFlags("DYNAMIC", "KINEMATIC")

    if flag == true then
          message = string.format("\nCollision detection between dynamic and kinematic
actors is 'true'")
    else
          message = string.format("\nCollision detection between dynamic and kinematic
actors is 'false'")
    end

    PrintConsole(message)
end

function Update()

end
```

In this example, if the collision detection between dynamic and kinematic physics actors is enabled, `GetPhysicsCollisionFlags` returns `true`, otherwise it returns `false`. Then we print the result in the console using the `PrintConsole` function.

4.127. GetPhysicsDefaultDynamicFriction

Definition

```
double GetPhysicsDefaultDynamicFriction()
```

Description

This function returns the value of physics default dynamic friction.

Return Value

The value of physics default dynamic friction.

Example

```
value = 0.0

function Init()
    value  = GetPhysicsDefaultDynamicFriction()

    message = string.format("\nDefault physics dynamic friction is : %.2f" ,value)
    PrintConsole(message)
end

function Update()

end
```

First we get the value of physics default dynamic friction. Then we display the result in the console using the `PrintConsole` function.

4.128. GetPhysicsDefaultRestitution

Definition

```
double GetPhysicsDefaultRestitution()
```

Description

This function returns the value of physics default restitution.

Return Value

The value of physics default restitution.

Example

```
value = 0.0

function Init()
    value  = GetPhysicsDefaultRestitution()

    message = string.format("\nDefault physics restitution is : %.2f" ,value)
    PrintConsole(message)
end

function Update()

end
```

First we get the value of physics default restitution. Then we display the result in the console using the `PrintConsole` function.

4.129. GetPhysicsDefaultSkinWidth

Definition

```
double GetPhysicsDefaultSkinWidth()
```

Description

This function returns the value of physics default skin width.

Return Value

The value of physics default skin width.

Example

```
value = 0.0

function Init()
    value  = GetPhysicsDefaultSkinWidth()

    message = string.format("\nDefault physics skin width is : %.2f" ,value)
    PrintConsole(message)
end

function Update()

end
```

First we get the value of physics default skin width. Then we display the result in the console using the `PrintConsole` function.

4.130. GetPhysicsDefaultStaticFriction

Definition

```
double GetPhysicsDefaultStaticFriction()
```

Description

This function returns the value of physics default static friction.

Return Value

The value of physics default static friction.

Example

```
value = 0.0

function Init()
    value  = GetPhysicsDefaultStaticFriction()

    message = string.format("\nDefault physics static friction is : %.2f" ,value)
    PrintConsole(message)
end

function Update()

end
```

First we get the value of physics default static friction. Then we display the result in the console using the `PrintConsole` function.

4.131. GetPhysicsGravity

Definition

```
double, double, double GetPhysicsGravity()
```

Description

This function returns the X, Y and Z components of physics gravity.

Return Values

X, Y and Z components of physics gravity.

Example

```
x = 0.0
y = 0.0
z = 0.0

function Init()
    x, y, z = GetPhysicsGravity()

    message = string.format("\nPhysics gravity is : (%.2f, %.2f, %.2f)" , x, y, z)
    PrintConsole(message)
end

function Update()

end
```

First, we get the X, Y and Z components of physics gravity. Then we display the results in the console using the `PrintConsole` function.

4.132. GetPhysicsGroundHeight

Definition

```
double GetPhysicsGroundHeight()
```

Description

This function returns the value of physics ground height.

Return Value

The value of physics ground height.

Example

```
value = 0.0

function Init()
    value = GetPhysicsGroundHeight()

    message = string.format("\nPhysics ground height is : %.2f" ,value)
    PrintConsole(message)
end

function Update()

end
```

First we get the value of physics ground height. Then we display the result in the console using the `PrintConsole` function.

4.133. GetPrefabInstanceAmbient

Definition

```
double, double, double GetPrefabInstanceAmbient(string prefabInstanceName)
```

Description

This function returns the ambient color of prefab instance `prefabInstanceName`.

Parameter

prefabInstanceName

Specifies the name of the prefab instance. You can also use the name "this" for this parameter. In this case, "this" refers to the prefab instance that this script is attached to.

Return Values

This functions returns the red, green, and blue components of prefab instance ambient color. Each value is in the range [0.0,1.0].

Example 1

```
r = 0.0
g = 0.0
b = 0.0

function Init()
    r, g, b = GetPrefabInstanceAmbient("1_VandaEngine17-SamplePack1_wood_pile")

    message = string.format("\nAmbient color is : (%.2f, %.2f, %.2f)", r, g, b)
    PrintConsole(message)
end

function Update()

end
```

First we get the ambient color of prefab instance `"1_VandaEngine17-SamplePack1_wood_pile"`. Then we display the result in the console using `PrintConsole` function.

Example 2

```
--Name of script is GetPrefabInstanceAmbient2.lua

r = 0.0
g = 0.0
b = 0.0

function Init()
    r, g, b = GetPrefabInstanceAmbient("this")

    message = string.format("\nAmbient color is : (%.2f, %.2f, %.2f)", r, g, b)
    PrintConsole(message)
end
```

```
function Update()

end
```

If, in the Prefab Editor, you attach `GetPrefabInstanceAmbient2.lua` script to a Prefab, then `"this"` parameter in the `GetPrefabInstanceAmbient` function will point to instances of that Prefab in current VScene. For example, if you have an Instance named *instance1_a* from a Prefab named *a* to which this script is attached, `"this"` in `GetPrefabInstanceAmbient` function refers to the name *instance1_a*.
In this example, we get the ambient color of current prefab instance (for example, *instance1_a*). Then we display the result in the console using `PrintConsole` function.

4.134. GetPrefabInstanceDiffuse

Definition

```
double, double, double GetPrefabInstanceDiffuse(string prefabInstanceName)
```

Description

This function returns the diffuse color of prefab instance `prefabInstanceName`.

Parameter

`prefabInstanceName`

Specifies the name of the prefab instance. You can also use the name "this" for this parameter. In this case, "this" refers to the prefab instance that this script is attached to.

Return Values

This functions returns the red, green, and blue components of prefab instance diffuse color. Each value is in the range [0.0,1.0].

Example 1

```
r = 0.0
g = 0.0
b = 0.0

function Init()
    r, g, b = GetPrefabInstanceDiffuse("1_VandaEngine17-SamplePack1_wood_pile")

    message = string.format("\nDiffuse color is : (%.2f, %.2f, %.2f)", r, g, b)
    PrintConsole(message)
end

function Update()

end
```

First we get the diffuse color of prefab instance `"1_VandaEngine17-SamplePack1_wood_pile"`. Then we display the result in the console using `PrintConsole` function.

Example 2

```
-- Name of script is GetPrefabInstanceDiffuse2.lua

r = 0.0
g = 0.0
b = 0.0

function Init()
    r, g, b = GetPrefabInstanceDiffuse("this")

    message = string.format("\nDiffuse color is : (%.2f, %.2f, %.2f)", r, g, b)
    PrintConsole(message)
end
```

```
function Update()

end
```

If, in the Prefab Editor, you attach `GetPrefabInstanceDiffuse2.lua` script to a Prefab, then `"this"` parameter in the `GetPrefabInstanceDiffuse` function will point to instances of that Prefab in current VScene. For example, if you have an Instance named *instance1_a* from a Prefab named *a* to which this script is attached, `"this"` in `GetPrefabInstanceDiffuse` function refers to the name *instance1_a*.

In this example, we get the diffuse color of current prefab instance (for example, *instance1_a*). Then we display the result in the console using `PrintConsole` function.

4.135. GetPrefabInstanceEmission

Definition

```
double, double, double GetPrefabInstanceEmission(string
prefabInstanceName)
```

Description

This function returns the emission color of prefab instance `prefabInstanceName`.

Parameter

prefabInstanceName

Specifies the name of the prefab instance. You can also use the name "this" for this parameter. In this case, "this" refers to the prefab instance that this script is attached to.

Return Values

This function returns the red, green, and blue components of prefab instance emission color. Each value is in the range [0.0,1.0].

Example 1

```
r = 0.0
g = 0.0
b = 0.0

function Init()
    r, g, b = GetPrefabInstanceEmission("1_VandaEngine17-SamplePack1_wood_pile")

    message = string.format("\nEmission color is : (%.2f, %.2f, %.2f)", r, g, b)
    PrintConsole(message)
end

function Update()

end
```

First we get the emission color of prefab instance `"1_VandaEngine17-SamplePack1_wood_pile"`. Then we display the result in the console using `PrintConsole` function.

Example 2

```
--Name of script is GetPrefabInstanceEmission2.lua

r = 0.0
g = 0.0
b = 0.0

function Init()
    r, g, b = GetPrefabInstanceEmission("this")

    message = string.format("\nEmission color is : (%.2f, %.2f, %.2f)", r, g, b)
    PrintConsole(message)
```

```
end

function Update()

end
```

If, in the Prefab Editor, you attach `GetPrefabInstanceEmission2.lua` script to a Prefab, then `"this"` parameter in the `GetPrefabInstanceEmission` function will point to instances of that Prefab in current VScene. For example, if you have an Instance named *instance1_a* from a Prefab named *a* to which this script is attached, `"this"` in `GetPrefabInstanceEmission` function refers to the name *instance1_a*.
In this example, we get the emission color of current prefab instance (for example, *instance1_a*). Then we display the result in the console using `PrintConsole` function.

4.136. GetPrefabInstanceNameFromActor

Definition

```
string GetPrefabInstanceNameFromActor(string physicsActorName)
```

Description

This function receives the physics actor **`physicsActorName`** and returns the name of the prefab instance to which **`physicsActorName`** belongs.

Parameters

`physicsActorName`

Specifies the name of the physics actor.

Return Value

This function returns the name of the prefab instance to which **`physicsActorName`** belongs.

Example

```
prefab_instance_name = ""

function OnTriggerEnter(otherActorName)
    prefab_instance_name = GetPrefabInstanceNameFromActor(otherActorName)

    message = string.format("\nPrefab instance name is > %s" ,prefab_instance_name)
    PrintConsole(message)
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)

end
```

Assume that this script is attached to a trigger. Whenever a prefab instance that has dynamic physics is entered into this trigger, the name of its physics actor is sent to the `OnTriggerEnter` event. Using the `GetPrefabInstanceNameFromActor` function, we find the prefab instance name that **`otherActorName`** name belongs to and display it in the console.

4.137. GetPrefabInstanceRadius

Definition

```
double GetPrefabInstanceRadius(string prefabInstanceName)
```

Description

This function receives the name of the prefab instance `prefabInstanceName` and returns its approximate radius.

Parameters

`prefabInstanceName`

Specifies the name of the prefab instance. You can also use the name "this" for this parameter. In this case, "this" refers to the prefab instance that this script is attached to.

Return Value

Returns approximate radius of prefab instance `prefabInstanceName`.

Example 1

```
radius = 0.0

function Init()
    radius = GetPrefabInstanceRadius("1_VandaEngine17-SamplePack1_well")

    message = string.format("\nPrefab instance radius is > %.2f" ,radius)
    PrintConsole(message)
end

function Update()

end
```

First, the `GetPrefabInstanceRadius` function returns the approximate radius of `"1_VandaEngine17-SamplePack1_well"`. Then we display the radius value in the console using the `PrintConsole` function.

Example 2

```
--Name of script is GetPrefabInstanceRadius2.lua

radius = 0.0

function Init()
    radius = GetPrefabInstanceRadius("this")

    message = string.format("\nPrefab instance radius is > %.2f" ,radius)
    PrintConsole(message)
end

function Update()

end
```

If, in the Prefab Editor, you attach `GetPrefabInstanceRadius2.lua` script to a Prefab, then `"this"` parameter in the `GetPrefabInstanceRadius` function will point to instances of that Prefab in current VScene. For example, if you have an Instance named *instance1_a* from a Prefab named *a* to which this script is attached, `"this"` in `GetPrefabInstanceRadius` function refers to the name *instance1_a*.

In this example, `GetPrefabInstanceRadius` function returns the approximate radius of current prefab instance (for example, *instance1_a*). Then we display the radius value in the console using the `PrintConsole` function.

4.138. GetPrefabInstanceRotate

Definition

`double,double,double GetPrefabInstanceRotate(string prefabInstanceName)`

Description

This function receives the name of the prefab instance `prefabInstanceName` and returns its rotation as three values x, y and z.

Parameters

prefabInstanceName

Specifies the name of the prefab instance. You can also use the name "this" for this parameter. In this case, "this" refers to the prefab instance that this script is attached to.

Return Value

This function returns the prefab instance rotation as three values x, y and z.

Example 1

```
rotateX = 0.0
rotateY = 0.0
rotateZ = 0.0

function Init()
    rotateX, rotateY, rotateZ = GetPrefabInstanceRotate("1_VandaEngine17-
SamplePack1_well")

    message = string.format("\nPrefab instance rotation is > (%.2f, %.2f, %.2f)" ,
rotateX, rotateY, rotateZ)
    PrintConsole(message)
end

function Update()

end
```

First, `GetPrefabInstanceRotate` function returns the rotation of `"1_VandaEngine17-SamplePack1_well"`. Then we display the rotation values in the console using the `PrintConsole` function.

Example 2

```
--Name of script is GetPrefabInstanceRotate2.lua

rotateX = 0.0
rotateY = 0.0
rotateZ = 0.0

function Init()
    rotateX, rotateY, rotateZ = GetPrefabInstanceRotate("this")
    message = string.format("\nPrefab instance rotation is > (%.2f, %.2f, %.2f)" ,
rotateX, rotateY, rotateZ)
    PrintConsole(message)
end

function Update()

end
```

If, in the Prefab Editor, you attach GetPrefabInstanceRotate2.lua script to a Prefab, then "this" parameter in the GetPrefabInstanceRotate function will point to instances of that Prefab in current VScene. For example, if you have an Instance named *instance1_a* from a Prefab named *a* to which this script is attached, "this" in GetPrefabInstanceRotate function refers to the name *instance1_a*.
In this example, GetPrefabInstanceRotate function returns the rotation of current prefab instance (for example, *instance1_a*). Then we display the rotation values in the console using the PrintConsole function.

4.139. GetPrefabInstanceScale

Definition

```
double,double,double GetPrefabInstanceScale(string prefabInstanceName)
```

Description

This function receives the name of the prefab instance `prefabInstanceName` and returns its scale as three values x, y and z.

Parameters

prefabInstanceName

Specifies the name of the prefab instance. You can also use the name "this" for this parameter. In this case, "this" refers to the prefab instance that this script is attached to.

Return Value

This function returns the prefab instance scale as three values x, y and z.

Example 1

```
scaleX = 0.0
scaleY = 0.0
scaleZ = 0.0

function Init()
    scaleX, scaleY, scaleZ = GetPrefabInstanceScale("1_VandaEngine17-SamplePack1_well")

    message = string.format("\nPrefab instance scale is > (%.2f, %.2f, %.2f)" , scaleX,
scaleY, scaleZ)
    PrintConsole(message)
end

function Update()

end
```

First, `GetPrefabInstanceScale` function returns the scale of `"1_VandaEngine17-SamplePack1_well"`. Then we display the scale values in the console using the `PrintConsole` function.

Example 2

```
--Name of script is GetPrefabInstanceScale2.lua

scaleX = 0.0
scaleY = 0.0
scaleZ = 0.0

function Init()
    scaleX, scaleY, scaleZ = GetPrefabInstanceScale("this")

    message = string.format("\nPrefab instance scale is > (%.2f, %.2f, %.2f)" , scaleX,
scaleY, scaleZ)
```

```
    PrintConsole(message)
end

function Update()

end
```

If, in the Prefab Editor, you attach GetPrefabInstanceScale2.lua script to a Prefab, then "this" parameter in the GetPrefabInstanceScale function will point to instances of that Prefab in current VScene. For example, if you have an Instance named *instance1_a* from a Prefab named *a* to which this script is attached, "this" in GetPrefabInstanceScale function refers to the name *instance1_a*.

In this example, GetPrefabInstanceScale function returns the scale of current prefab instance (for example, *instance1_a*). Then we display the scale values in the console using the PrintConsole function.

4.140. GetPrefabInstanceScriptBoolVariable

Definition

`bool GetPrefabInstanceScriptBoolVariable(string PrefabInstanceName, string variable)`

Description

This function gets the value of the Boolean **`variable`** defined in the script attached to the **`PrefabInstanceName`** prefab instance.

Parameters

`PrefabInstanceName`
Specifies the name of the prefab instance.

`variable`
Specifies the name of the Boolean variable defined in the script attached to the *`PrefabInstanceName`* prefab instance.

Return Value

Returns the value of the Boolean **`variable`** defined in the script attached to the *`PrefabInstanceName`* prefab instance.

Example

```
--script name is GetPrefabInstanceScriptBoolVariable.lua attached a to game object such
as water
value = false
function Init()
    value = GetPrefabInstanceScriptBoolVariable("1_VandaEngine17-SamplePack1_birdcage",
"a")
end

function Update()

end
```

Assuming that the variable `"a"` is defined with the value *true* in the script attached to the prefab instance `"1_VandaEngine17-SamplePack1_birdcage"`, `GetPrefabInstanceScriptBoolVariable` function returns the value *true*.

4.141. GetPrefabInstanceScriptDoubleVariable

Definition

`double GetPrefabInstanceScriptDoubleVariable(string PrefabInstanceName, string variable)`

Description

This function gets the value of the Double **`variable`** defined in the script attached to the **`PrefabInstanceName`** prefab instance.

Parameters

`PrefabInstanceName`
Specifies the name of the prefab instance.

`variable`
Specifies the name of the Double variable defined in the script attached to the *`PrefabInstanceName`* prefab instance.

Return Value

Returns the value of the Double **`variable`** defined in the script attached to the *`PrefabInstanceName`* prefab instance.

Example

```
--script name is GetPrefabInstanceScriptDoubleVariable.lua attached a to game object such as water
return_value = 0.0

function Init()
    return_value = GetPrefabInstanceScriptDoubleVariable("1_VandaEngine17-SamplePack1_birdcage", "a")
end

function Update()

end
```

Assuming that the variable `"a"` is defined with the value *1.0* in the script attached to the prefab instance `"1_VandaEngine17-SamplePack1_birdcage"`, `GetPrefabInstanceScriptDoubleVariable` function returns the value *1.0*.

4.142. GetPrefabInstanceScriptIntVariable

Definition

```
int GetPrefabInstanceScriptIntVariable(string PrefabInstanceName, string
variable)
```

Description

This function gets the value of the Integer `variable` defined in the script attached to the `PrefabInstanceName` prefab instance.

Parameters

`PrefabInstanceName`
Specifies the name of the prefab instance.

`variable`
Specifies the name of the Integer variable defined in the script attached to the `PrefabInstanceName` prefab instance.

Return Value

Returns the value of the Integer `variable` defined in the script attached to the `PrefabInstanceName` prefab instance.

Example

```
--script name is GetPrefabInstanceScriptIntVariable.lua attached a to game object such as
water
return_value = 0

function Init()
    return_value = GetPrefabInstanceScriptIntVariable("1_VandaEngine17-
SamplePack1_birdcage", "a")
end

function Update()

end
```

Assuming that the variable `"a"` is defined with the value *1* in the script attached to the prefab instance `"1_VandaEngine17-SamplePack1_birdcage"`, `GetPrefabInstanceScriptIntVariable` function returns the value *1*.

4.143. GetPrefabInstanceScriptStringVariable

Definition

`string GetPrefabInstanceScriptStringVariable(string PrefabInstanceName, string variable)`

Description

This function gets the value of the String **`variable`** defined in the script attached to the `PrefabInstanceName` prefab instance.

Parameters

`PrefabInstanceName`
Specifies the name of the prefab instance.

`variable`
Specifies the name of the String variable defined in the script attached to the `PrefabInstanceName` prefab instance.

Return Value

Returns the value of the String **`variable`** defined in the script attached to the `PrefabInstanceName` prefab instance.

Example

```
--script name is GetPrefabInstanceScriptStringVariable.lua attached a to game object such as water
return_value = ""

function Init()
    return_value = GetPrefabInstanceScriptStringVariable("1_VandaEngine17-SamplePack1_birdcage", "a")
end

function Update()

end
```

Assuming that the variable **`"a"`** is defined with the value "*hello*" in the script attached to the prefab instance `"1_VandaEngine17-SamplePack1_birdcage"`, `GetPrefabInstanceScriptStringVariable` function returns the value "*hello*".

4.144. GetPrefabInstanceShininess

Definition

```
double GetPrefabInstanceShininess(string prefabInstanceName)
```

Description

This function returns the shininess value of prefab instance `prefabInstanceName`.

Parameter

`prefabInstanceName`

Specifies the name of the prefab instance. You can also use the name "this" for this parameter. In this case, "this" refers to the prefab instance that this script is attached to.

Return Value

This function returns the shininess of prefab instance. This value is greater than or equal to 0.0.

Example 1

```
value = 0.0

function Init()
    value = GetPrefabInstanceShininess("1_VandaEngine17-SamplePack1_wood_pile")

    message = string.format("\nShininess is : (%.2f)", value)
    PrintConsole(message)
end

function Update()

end
```

First we get the shininess of prefab instance `"1_VandaEngine17-SamplePack1_wood_pile"`. Then we display the result in the console using `PrintConsole` function.

Example 2

```
--Name of script is GetPrefabInstanceShininess2.lua

value = 0.0

function Init()
    value = GetPrefabInstanceShininess("this")

    message = string.format("\nShininess is : (%.2f)", value)
    PrintConsole(message)
end

function Update()

end
```

`If, in the Prefab Editor, you attach GetPrefabInstanceShininess2.lua script to a Prefab, then "this" parameter in the GetPrefabInstanceShininess function will point to instances of that Prefab in current VScene. For example, if you have an Instance named` *`instance1_a`* `from a Prefab named` *`a`* `to which this script is attached, "this" in GetPrefabInstanceShininess function refers to the name` *`instance1_a`*`.`

In this example, we get the shininess of current prefab instance (for example, *`instance1_a`*). Then we display the result in the console using `PrintConsole` function.

4.145. GetPrefabInstanceSpecular

Definition

`double, double, double GetPrefabInstanceSpecular(string prefabInstanceName)`

Description

This function returns the specular color of prefab instance `prefabInstanceName`.

Parameter

prefabInstanceName

Specifies the name of the prefab instance. You can also use the name "this" for this parameter. In this case, "this" refers to the prefab instance that this script is attached to.

Return Values

This function returns the red, green, and blue components of prefab instance specular color. Each value is in the range [0.0,1.0].

Example 1

```
r = 0.0
g = 0.0
b = 0.0

function Init()
    r, g, b = GetPrefabInstanceSpecular("1_VandaEngine17-SamplePack1_wood_pile")

    message = string.format("\nSpecular color is : (%.2f, %.2f, %.2f)", r, g, b)
    PrintConsole(message)
end

function Update()

end
```

First we get the specular color of prefab instance `"1_VandaEngine17-SamplePack1_wood_pile"`. Then we display the result in the console using `PrintConsole` function.

Example 2

```
--Name of script is GetPrefabInstanceSpecular2.lua

r = 0.0
g = 0.0
b = 0.0

function Init()
    r, g, b = GetPrefabInstanceSpecular("this")

    message = string.format("\nSpecular color is : (%.2f, %.2f, %.2f)", r, g, b)
    PrintConsole(message)
```

```
end

function Update()

end
```

If, in the Prefab Editor, you attach GetPrefabInstanceSpecular2.lua script to a Prefab, then "this" parameter in the GetPrefabInstanceSpecular function will point to instances of that Prefab in current VScene. For example, if you have an Instance named *instance1_a* from a Prefab named *a* to which this script is attached, "this" in GetPrefabInstanceSpecular function refers to the name *instance1_a*.

In this example, we get the specular color of current prefab instance (for example, *instance1_a*). Then we display the result in the console using PrintConsole function.

4.146. GetPrefabInstanceTranslate

Definition

```
double,double,double GetPrefabInstanceTranslate(string prefabInstanceName)
```

Description

This function receives the name of the prefab instance `prefabInstanceName` and returns its position as three values x, y and z.

Parameters

prefabInstanceName

Specifies the name of the prefab instance. You can also use the name "this" for this parameter. In this case, "this" refers to the prefab instance that this script is attached to.

Return Value

This function returns the prefab instance position as three values x, y and z.

Example 1

```
posX = 0.0
posY = 0.0
posZ = 0.0

function Init()
    posX, posY, posZ = GetPrefabInstanceTranslate("1_VandaEngine17-SamplePack1_well")

    message = string.format("\nPrefab instance position is > (%.2f, %.2f, %.2f)" , posX,
posY, posZ)
    PrintConsole(message)
end

function Update()

end
```

First, `GetPrefabInstanceTranslate` function returns the position of `"1_VandaEngine17-SamplePack1_well"`. Then we display the position values in the console using the `PrintConsole` function.

Example 2

```
--Name of script is GetPrefabInstanceTranslate2.lua

posX = 0.0
posY = 0.0
posZ = 0.0

function Init()
    posX, posY, posZ = GetPrefabInstanceTranslate("this")

    message = string.format("\nPrefab instance position is > (%.2f, %.2f, %.2f)" , posX,
posY, posZ)
```

```
    PrintConsole(message)
end

function Update()

end
```

If, in the Prefab Editor, you attach GetPrefabInstanceTranslate2.lua script to a Prefab, then "this" parameter in the GetPrefabInstanceTranslate function will point to instances of that Prefab in current VScene. For example, if you have an Instance named *instance1_a* from a Prefab named *a* to which this script is attached, "this" in GetPrefabInstanceTranslate function refers to the name *instance1_a*.

In this example, GetPrefabInstanceTranslate function returns the position of current prefab instance (for example, *instance1_a*). Then we display the position values in the console using the PrintConsole function.

4.147. GetPrefabInstanceTransparency

Definition

```
double GetPrefabInstanceTransparency(string prefabInstanceName)
```

Description

This function returns the transparency of prefab instance `prefabInstanceName`.

Parameter

prefabInstanceName

Specifies the name of the prefab instance. You can also use the name "this" for this parameter. In this case, "this" refers to the prefab instance that this script is attached to.

Return Value

This function returns the transparency of prefab instance. This value is in the range [0.0,1.0].

Example 1

```
value = 0.0

function Init()
    value = GetPrefabInstanceTransparency("1_VandaEngine17-SamplePack1_wood_pile")

    message = string.format("\nTransparency is : (%.2f)", value)
    PrintConsole(message)
end

function Update()

end
```

First we get the transparency of prefab instance `"1_VandaEngine17-SamplePack1_wood_pile"`. Then we display the result in the console using the `PrintConsole` function.

Example 2

```
--Name of script is GetPrefabInstanceTransparency2.lua

value = 0.0

function Init()
    value = GetPrefabInstanceTransparency("this")

    message = string.format("\nTransparency is : (%.2f)", value)
    PrintConsole(message)
end

function Update()

end
```

If, in the Prefab Editor, you attach GetPrefabInstanceTransparency2.lua script to a Prefab, then "this" parameter in the GetPrefabInstanceTransparency function will point to instances of that Prefab in current VScene. For example, if you have an Instance named *instance1_a* from a Prefab named *a* to which this script is attached, "this" in GetPrefabInstanceTransparency function refers to the name *instance1_a*.
In this example, we get the transparency of current prefab instance (for example, *instance1_a*). Then we display the result in the console using the PrintConsole function.

4.148. GetScreenHeight

Definition

```
int GetScreenHeight()
```

Description

This function returns the height of the screen in pixels.

Return Value

Height of the screen in pixels.

Example

```
height = 0

function Init()
    height = GetScreenHeight()

    message = string.format("\nScreen height is > %d" , height)
    PrintConsole(message)
end

function Update()

end
```

4.149. GetScreenResolution

Definition

```
int GetScreenResolution()
```

Description

When running the game, you can select the resolution from the dialog that appears at the beginning of the game. This function returns the width of the selected resolution in pixels.

Return Value

If the current resolution of the monitor is selected, it returns 0, otherwise it returns the width of the selected resolution.

Example

```
resolution = 0

function Init()
    resolution = GetScreenResolution()

    message = string.format("\nScreen resolution is > %d" , resolution)
    PrintConsole(message)
end

function Update()

end
```

4.150. GetScreenWidth

Definition

```
int GetScreenWidth()
```

Description

This function returns the width of the screen in pixels.

Return Value

Width of the screen in pixels.

Example

```
width = 0

function Init()
    width = GetScreenWidth()

    message = string.format("\nScreen width is > %d" , width)
    PrintConsole(message)
end

function Update()

end
```

4.151. GetSelectionDistance

Definition

`double GetSelectionDistance()`

Description

This function returns the maximum distance from the camera that you can select a prefab instance using the *SelectPrefabInstances* function. You can set the maximum distance for selection through the *SetSelectionDistance* function.

Return Value

Returns the maximum distance from the camera that you can select a prefab instance using the *SelectPrefabInstances* function.

Example

```
selection_distance = 0.0

function Init()
    SetSelectionDistance(5.5)

    selection_distance = GetSelectionDistance()

    message = string.format("\nSelection distance is > %.2f" , selection_distance)
    PrintConsole(message)
end

function Update()

end
```

First, we set the maximum distance for selection to `5.5` using the `SetSelectionDistance` function. Then, using the `GetSelectionDistance` function, we return the maximum selection value, which in our example is equal to `5.5`. Finally, using the `PrintConsole` function, we display the selection value in the console. Below is the message displayed:

```
Selection distance is > 5.50
```

4.152. GetSkyPosition

Definition

```
double,double,double GetSkyPosition()
```

Description

This function returns sky position as three values x, y and z.

Return Value

Sky position as three values x, y and z.

Example 1

```
posX = 0.0
posY = 0.0
posZ = 0.0

function Init()
    posX, posY, posZ = GetSkyPosition()

    message = string.format("\nSky position is > (%.2f, %.2f, %.2f)" , posX, posY, posZ)
    PrintConsole(message)
end

function Update()

end
```

First, `GetSkyPosition` function returns sky position. Then we display the position values in the console using the `PrintConsole` function.

4.153. GetSkyScriptBoolVariable

Definition

`bool GetSkyScriptBoolVariable(string variable)`

Description

This function gets the value of the Boolean **`variable`** defined in the script attached to the sky object.

Parameters

`variable`

Specifies the name of the Boolean variable defined in the script attached to the sky object.

Return Value

Returns the value of the Boolean **`variable`** defined in the script attached to the sky object.

Example

```lua
--script name is GetSkyScriptBoolVariable.lua attached a to game object such as water
value = false
function Init()
    value = GetSkyScriptBoolVariable("a")
end

function Update()

end
```

Assuming that the variable `"a"` is defined with the value *true* in the script attached to the sky object, `GetSkyScriptBoolVariable` function returns the value *true*.

4.154. GetSkyScriptDoubleVariable

Definition

```
double GetSkyScriptDoubleVariable(string variable)
```

Description

This function gets the value of the Double **variable** defined in the script attached to the sky object.

Parameters

variable

Specifies the name of the Double variable defined in the script attached to the sky object.

Return Value

Returns the value of the Double **variable** defined in the script attached to the sky object.

Example

```
--script name is GetSkyScriptDoubleVariable.lua attached a to game object such as water
return_value = 0.0

function Init()
    return_value = GetSkyScriptDoubleVariable("a")
end

function Update()

end
```

Assuming that the variable "a" is defined with the value *1.0* in the script attached to the sky object, GetSkyScriptDoubleVariable function returns the value *1.0*.

4.155. GetSkyScriptIntVariable

Definition

```
int GetSkyScriptIntVariable(string variable)
```

Description

This function gets the value of the Integer **`variable`** defined in the script attached to the sky object.

Parameters

`variable`

Specifies the name of the Integer variable defined in the script attached to the sky object.

Return Value

Returns the value of the Integer **`variable`** defined in the script attached to the sky object.

Example

```
--script name is GetSkyScriptIntVariable.lua attached a to game object such as water
return_value = 0

function Init()
    return_value = GetSkyScriptIntVariable("a")
end

function Update()

end
```

Assuming that the variable `"a"` is defined with the value *1* in the script attached to the sky, `GetSkyScriptIntVariable` function returns the value *1*.

4.156. GetSkyScriptStringVariable

Definition

```
string GetSkyScriptStringVariable(string variable)
```

Description

This function gets the value of the String **`variable`** defined in the script attached to the sky object.

Parameters

`variable`

Specifies the name of the String variable defined in the script attached to the sky object.

Return Value

Returns the value of the String **`variable`** defined in the script attached to the sky object.

Example

```
--script name is GetSkyScriptStringVariable.lua attached a to game object such as water
return_value = ""

function Init()
    return_value = GetSkyScriptStringVariable("a")
end

function Update()

end
```

Assuming that the variable `"a"` is defined with the value "*hello*" in the script attached to the sky object, `GetSkyScriptStringVariable` function returns the value "*hello*".

4.157. GetSoundLoop

Definition

```
bool GetSoundLoop(string soundObjectName)
```

Description

This function returns the state of the sound loop as a Boolean value of true or false.

Parameters

soundObjectName

Specifies the ambient or 3D sound name. You can also use the name "this" for this parameter. In this case, "this" string refers to the name of the sound to which this script is attached.

Return Value

If the state of the loop is true, it returns *true,* otherwise it returns *false*.

Example 1

```
sound_loop = false
message = ""

function Init()
    sound_loop = GetSoundLoop("sound1")

    if sound_loop then
          message = string.format("\nSound Loop is ON")
    else
          message = string.format("\nSound Loop is OFF")
    end

    PrintConsole(message)
end

function Update()

end
```

First, we specify the loop state of `"sound1"`. Then we display its status in the console using the `PrintConsole` function.

Example 2

```
--Name of script is GetSoundLoop2.lua

sound_loop = false
message = ""

function Init()
    sound_loop = GetSoundLoop("this")

    if sound_loop then
          message = string.format("\nSound Loop is ON")
    else
          message = string.format("\nSound Loop is OFF")
    end

    PrintConsole(message)
end

function Update()

end
```

Assume that the above script named `GetSoundLoop2.lua` is attached to a sound object named "sound1". In this case, string `"this"` in the `GetSoundLoop` function will be equal to "sound1". In our example, the function `GetSoundLoop` returns the loop state of the sound "sound1".

4.158. GetSoundMaxDistance

Definition

```
double GetSoundMaxDistance(string 3DSoundObjectName)
```

Description

This function returns the maximum distance of 3D sound `3DSoundObjectName`.

Parameters

3DSoundObjectName

Specifies the 3D sound name. You can also use the name "this" for this parameter. In this case, "this" string refers to the name of the 3D sound to which this script is attached.

Return Value

Maximum distance of 3D sound.

Example 1

```
max_distance = 0.0

function Init()
    max_distance = GetSoundMaxDistance("sound1")

    message = string.format("\nSound max distance is > %.2f", max_distance)
    PrintConsole(message)
end

function Update()

end
```

First, we get the maximum distance of 3D sound `"sound1"`. Then we display it in the console using the `PrintConsole` function.

Example 2

```
--Name of script is GetSoundMaxDistance2.lua

max_distance = 0.0

function Init()
    max_distance = GetSoundMaxDistance("this")

    message = string.format("\nSound max distance is > %.2f", max_distance)
    PrintConsole(message)
end

function Update()

end
```

Assume that the above script named `GetSoundMaxDistance2.lua` is attached to a 3D sound object named "sound1". In this case, string `"this"` in the `GetSoundMaxDistance` function will be equal to "sound1". In our example, the function `GetSoundMaxDistance` returns the maximum distance of current 3D sound, which is "sound1".

4.159. GetSoundPitch

Definition

```
double GetSoundPitch(string soundObjectName)
```

Description

This function returns the pitch of ambient or 3D sound `soundObjectName`.

Parameters

soundObjectName

Specifies the ambient or 3D sound name. You can also use the name "this" for this parameter. In this case, "this" string refers to the name of the sound to which this script is attached.

Return Value

pitch of 3D or ambient sound.

Example 1

```
pitch = 0.0

function Init()
    pitch = GetSoundPitch("sound1")

    message = string.format("\nSound pitch is > %.2f", pitch)
    PrintConsole(message)
end

function Update()

end
```

First, we get the pitch of sound `"sound1"`. Then we display it in the console using the `PrintConsole` function.

Example 2

```
--Name of script is GetSoundPitch2.lua

pitch = 0.0

function Init()
    pitch = GetSoundPitch("this")

    message = string.format("\nSound pitch is > %.2f", pitch)
    PrintConsole(message)
end

function Update()

end
```

Assume that the above script named `GetSoundPitch2.lua` is attached to a sound object named "sound1". In this case, string `"this"` in the `GetSoundPitch` function will be equal to "sound1". In our example, the function `GetSoundPitch` returns the pitch of current sound, which is "sound1".

4.160. GetSoundPlay

Definition

```
bool GetSoundPlay(string soundObjectName)
```

Description

This function returns the sound playback status as a Boolean value of true or false.

Parameters

soundObjectName

Specifies the ambient or 3D sound name. You can also use the name "this" for this parameter. In this case, "this" string refers to the name of the sound to which this script is attached.

Return Value

If the sound is playing, it returns *true,* otherwise it returns *false*.

Example 1

```
sound_play = false
message = ""

function Init()
    sound_play = GetSoundPlay("sound1")

    if sound_play then
          message = string.format("\nSound is playing")
    else
          message = string.format("\nSound isn't playing")
    end

    PrintConsole(message)
end

function Update()

end
```

First, we specify the playback state of `"sound1"`. Then we display its status in the console using the `PrintConsole` function.

Example 2

```
--Name of script is GetSoundPlay2.lua

sound_play = false
message = ""

function Init()
    sound_play = GetSoundPlay("this")

    if sound_play then
          message = string.format("\nSound is playing")
```

```
	else
		message = string.format("\nSound isn't playing")
	end

	PrintConsole(message)
end

function Update()

end
```

Assume that the above script named `GetSoundPlay2.lua` is attached to a sound object named "sound1". In this case, string `"this"` in the `GetSoundPlay` function will be equal to "sound1". In our example, the function `GetSoundPlay` returns the playback state of the sound "sound1".

4.161. GetSoundPosition

Definition

```
double,double,double GetSoundPosition(string 3DSoundObjectName)
```

Description

This function receives the name of the 3D sound `3DSoundObjectName` and returns its position as three values x, y and z.

Parameters

`3DSoundObjectName`

Specifies the name of the 3D sound object. You can also use the name "this" for this parameter. In this case, "this" refers to the 3D sound name that this script is attached to.

Return Value

This function returns the 3D sound position as three values x, y and z.

Example 1

```
posX = 0.0
posY = 0.0
posZ = 0.0

function Init()
    posX, posY, posZ = GetSoundPosition("sound1")

    message = string.format("\nSound position is > (%.2f, %.2f, %.2f)" , posX, posY, 
posZ)
    PrintConsole(message)
end

function Update()

end
```

First, `GetSoundPosition` function returns the position of `"sound1"` 3D sound. Then we display the position values in the console using the `PrintConsole` function.

Example 2

```
--Name of script is GetSoundPosition2.lua

posX = 0.0
posY = 0.0
posZ = 0.0

function Init()
    posX, posY, posZ = GetSoundPosition("this")

    message = string.format("\nSound position is > (%.2f, %.2f, %.2f)" , posX, posY, 
posZ)
    PrintConsole(message)
```

```
end

function Update()

end
```

Assume that the above script named `GetSoundPosition2.lua` is attached to a 3D sound object named "sound1". In this case, string `"this"` in the `GetSoundPosition` function will be equal to "sound1". In our example, the function `GetSoundPosition` returns the position of current 3D sound, which is "sound1". Then we display the position values in the console using the `PrintConsole` function.

4.162. GetSoundReferenceDistance

Definition

```
double GetSoundReferenceDistance(string 3DSoundObjectName)
```

Description

This function returns the reference distance of 3D sound `3DSoundObjectName`.

Parameters

`3DSoundObjectName`

Specifies the 3D sound name. You can also use the name "this" for this parameter. In this case, "this" string refers to the name of the 3D sound to which this script is attached.

Return Value

Reference distance of 3D sound.

Example 1

```
ref_distance = 0.0

function Init()
    ref_distance = GetSoundReferenceDistance("sound1")

    message = string.format("\nSound reference distance is > %.2f", ref_distance)
    PrintConsole(message)
end

function Update()

end
```

First, we get the reference distance of 3D sound `"sound1"`. Then we display it in the console using the `PrintConsole` function.

Example 2

```
--Name of script is GetSoundReferenceDistance2.lua

ref_distance = 0.0

function Init()
    ref_distance = GetSoundReferenceDistance("this")

    message = string.format("\nSound reference distance is > %.2f", ref_distance)
    PrintConsole(message)
end

function Update()

end
```

Assume that the above script named `GetSoundReferenceDistance2.lua` is attached to a 3D sound object named "sound1". In this case, string `"this"` in the `GetSoundReferenceDistance` function will be equal to "sound1". In our example, the function `GetSoundReferenceDistance` returns the reference distance of current 3D sound, which is "sound1".

4.163. GetSoundRollOff

Definition

```
double GetSoundRollOff(string 3DSoundObjectName)
```

Description

This function returns the rolloff of 3D sound `3DSoundObjectName`.

Parameters

3DSoundObjectName

Specifies the 3D sound name. You can also use the name "this" for this parameter. In this case, "this" string refers to the name of the 3D sound to which this script is attached.

Return Value

Rolloff of 3D sound.

Example 1

```
rolloff = 0.0

function Init()
    rolloff = GetSoundRollOff("sound1")

    message = string.format("\nSound rolloff is > %.2f", rolloff)
    PrintConsole(message)
end

function Update()

end
```

First, we get the rolloff of 3D sound `"sound1"`. Then we display it in the console using the `PrintConsole` function.

Example 2

```
--Name of script is GetSoundRollOff2.lua

rolloff = 0.0

function Init()
    rolloff = GetSoundRollOff("this")

    message = string.format("\nSound rolloff is > %.2f", rolloff)
    PrintConsole(message)
end

function Update()

end
```

Assume that the above script named `GetSoundRollOff2.lua` is attached to a 3D sound object named "sound1". In this case, string `"this"` in the `GetSoundRollOff` function will be equal to "sound1". In our example, the function `GetSoundRollOff` returns the rolloff of current 3D sound, which is "sound1".

4.164. GetSoundVolume

Definition

```
double GetSoundVolume(string soundObjectName)
```

Description

This function returns the volume of ambient or 3D sound `soundObjectName`.

Parameters

soundObjectName

Specifies the ambient or 3D sound name. You can also use the name "this" for this parameter. In this case, "this" string refers to the name of the sound to which this script is attached.

Return Value

Volume of 3D or ambient sound.

Example 1

```
volume = 0.0

function Init()
    volume = GetSoundVolume("sound1")

    message = string.format("\nSound volume is > %.2f", volume)
    PrintConsole(message)
end

function Update()

end
```

First, we get the volume of sound `"sound1"`. Then we display it in the console using the `PrintConsole` function.

Example 2

```
--Name of script is GetSoundVolume2.lua

volume = 0.0

function Init()
    volume = GetSoundVolume("this")

    message = string.format("\nSound volume is > %.2f", volume)
    PrintConsole(message)
end

function Update()

end
```

Assume that the above script named `GetSoundVolume2.lua` is attached to a sound object named "sound1". In this case, string `"this"` in the `GetSoundVolume` function will be equal to "sound1". In our example, the function `GetSoundVolume` returns the volume of current sound, which is "sound1".

4.165. GetTerrainAmbient

Definition

```
double,double,double GetTerrainAmbient()
```

Description

This function returns the ambient color of terrain object as three values of red, green and blue. Each value ranges from 0 to 1.

Return Value

Ambient color of terrain object as three values of red, green and blue. Each value ranges from 0 to 1.

Example

```
red = 0.0
green = 0.0
blue = 0.0

function Init()
    red, green, blue = GetTerrainAmbient()

    message = string.format("\nTerrain ambient color is > (%.2f, %.2f, %.2f)" , red,
green, blue)
    PrintConsole(message)
end

function Update()

end
```

In this example, the `GetTerrainAmbient` function returns the value of the red, green, and blue components of the ambient color of terrain object. Then these three values are displayed on the console by the `PrintConsole` function.

4.166. GetTerrainDiffuse

Definition

```
double,double,double GetTerrainDiffuse()
```

Description

This function returns the diffuse color of terrain object as three values of red, green and blue. Each value ranges from 0 to 1.

Return Value

Diffuse color of terrain object as three values of red, green and blue. Each value ranges from 0 to 1.

Example

```
red = 0.0
green = 0.0
blue = 0.0

function Init()
    red, green, blue = GetTerrainDiffuse()

    message = string.format("\nTerrain diffuse color is > (%.2f, %.2f, %.2f)" , red,
green, blue)
    PrintConsole(message)
end

function Update()

end
```

In this example, the `GetTerrainDiffuse` function returns the value of the red, green, and blue components of the diffuse color of terrain object. Then these three values are displayed on the console by the `PrintConsole` function.

4.167. GetTerrainScriptBoolVariable

Definition

`bool GetTerrainScriptBoolVariable(string variable)`

Description

This function gets the value of the Boolean **`variable`** defined in the script attached to the terrain object.

Parameters

`variable`

Specifies the name of the Boolean variable defined in the script attached to the terrain object.

Return Value

Returns the value of the Boolean **`variable`** defined in the script attached to the terrain object.

Example

```
--script name is GetTerrainScriptBoolVariable.lua attached a to game object such as water
value = false
function Init()
    value = GetTerrainScriptBoolVariable("a")
end

function Update()

end
```

Assuming that the variable `"a"` is defined with the value *true* in the script attached to the terrain object, `GetTerrainScriptBoolVariable` function returns the value *true*.

4.168. GetTerrainScriptDoubleVariable

Definition

```
double GetTerrainScriptDoubleVariable(string variable)
```

Description

This function gets the value of the Double `variable` defined in the script attached to the terrain object.

Parameters

`variable`

Specifies the name of the Double variable defined in the script attached to the terrain object.

Return Value

Returns the value of the Double `variable` defined in the script attached to the terrain object.

Example

```
--script name is GetTerrainScriptDoubleVariable.lua attached a to game object such as water
return_value = 0.0

function Init()
    return_value = GetTerrainScriptDoubleVariable("a")
end

function Update()

end
```

Assuming that the variable `"a"` is defined with the value *1.0* in the script attached to the terrain object, `GetTerrainScriptDoubleVariable` function returns the value *1.0*.

4.169. GetTerrainScriptIntVariable

Definition

`int GetTerrainScriptIntVariable(string variable)`

Description

This function gets the value of the Integer **`variable`** defined in the script attached to the terrain object.

Parameters

`variable`

Specifies the name of the Integer variable defined in the script attached to the terrain object.

Return Value

Returns the value of the Integer **`variable`** defined in the script attached to the terrain object.

Example

```
--script name is GetTerrainScriptIntVariable.lua attached a to game object such as water
return_value = 0

function Init()
    return_value = GetTerrainScriptIntVariable("a")
end

function Update()

end
```

Assuming that the variable `"a"` is defined with the value *1* in the script attached to the terrain object, `GetTerrainScriptIntVariable` function returns the value *1*.

4.170. GetTerrainScriptStringVariable

Definition

`string GetTerrainScriptStringVariable(string variable)`

Description

This function gets the value of the String **`variable`** defined in the script attached to the terrain object.

Parameters

`variable`

Specifies the name of the String variable defined in the script attached to the terrain object.

Return Value

Returns the value of the String **`variable`** defined in the script attached to the terrain object.

Example

```
--script name is GetTerrainScriptStringVariable.lua attached a to game object such as
water
return_value = ""

function Init()
    return_value = GetTerrainScriptStringVariable("a")
end

function Update()

end
```

Assuming that the variable **`"a"`** is defined with the value "*hello*" in the script attached to the terrain object, `GetTerrainScriptStringVariable` function returns the value "*hello*".

4.171. GetTerrainShininess

Definition

```
double GetTerrainShininess()
```

Description

This function returns the shininess of terrain object.

Return Value

Shininess of terrain.

Example

```
shininess = 0.0

function Init()
    shininess  = GetTerrainShininess()

    message = string.format("\nTerrain shininess is > %.2f" ,shininess)
    PrintConsole(message)
end

function Update()

end
```

In this example, the `GetTerrainShininess` function returns the shininess value of terrain object. Then shininess value is displayed on the console by the `PrintConsole` function.

4.172. GetTerrainSpecular

Definition

```
double,double,double GetTerrainSpecular()
```

Description

This function returns the specular color of terrain object as three values of red, green and blue. Each value ranges from 0 to 1.

Return Value

Specular color of terrain object as three values of red, green and blue. Each value ranges from 0 to 1.

Example

```
red = 0.0
green = 0.0
blue = 0.0

function Init()
    red, green, blue = GetTerrainSpecular()

    message = string.format("\nTerrain specular color is > (%.2f, %.2f, %.2f)" , red,
green, blue)
    PrintConsole(message)
end

function Update()

end
```

In this example, the `GetTerrainSpecular` function returns the value of the red, green, and blue components of the specular color of terrain object. Then these three values are displayed on the console by the `PrintConsole` function.

4.173. GetTriggerScriptBoolVariable

Definition

`bool GetTriggerScriptBoolVariable(string triggerName, string variable)`

Description

This function gets the value of the Boolean **`variable`** defined in the script attached to the **`triggerName`** trigger object.

Parameters

`triggerName`
Specifies the name of the trigger object.

`variable`
Specifies the name of the Boolean variable defined in the script attached to the **`triggerName`** trigger object.

Return Value

Returns the value of the Boolean **`variable`** defined in the script attached to the **`triggerName`** trigger object.

Example

```
--script name is GetTriggerScriptBoolVariable.lua attached a to game object such as water
value = false
function Init()
    value = GetTriggerScriptBoolVariable("trigger1", "a")
end

function Update()

end
```

Assuming that the variable `"a"` is defined with the value *true* in the script attached to the trigger object `"trigger1"`, `GetTriggerScriptBoolVariable` function returns the value *true*.

4.174. GetTriggerScriptDoubleVariable

Definition

`double GetTriggerScriptDoubleVariable(string triggerName, string variable)`

Description

This function gets the value of the Double **`variable`** defined in the script attached to the **`triggerName`** trigger object.

Parameters

`triggerName`

Specifies the name of the trigger object.

`variable`

Specifies the name of the Double variable defined in the script attached to the **`triggerName`** trigger object.

Return Value

Returns the value of the Double **`variable`** defined in the script attached to the **`triggerName`** trigger object.

Example

```
--script name is GetTriggerScriptDoubleVariable.lua attached a to game object such as
water
return_value = 0.0

function Init()
    return_value = GetTriggerScriptDoubleVariable("trigger1", "a")
end

function Update()

end
```

Assuming that the variable `"a"` is defined with the value *1.0* in the script attached to the trigger object `"trigger1"`, `GetTriggerScriptDoubleVariable` function returns the value *1.0*.

4.175. GetTriggerScriptIntVariable

Definition

```
int GetTriggerScriptIntVariable(string triggerName, string variable)
```

Description

This function gets the value of the Integer **`variable`** defined in the script attached to the **`triggerName`** trigger object.

Parameters

`triggerName`

Specifies the name of the trigger object.

`variable`

Specifies the name of the Integer variable defined in the script attached to the `triggerName` trigger object.

Return Value

Returns the value of the Integer `variable` defined in the script attached to the `triggerName` trigger object.

Example

```
--script name is GetTriggerScriptIntVariable.lua attached a to game object such as water
return_value = 0

function Init()
    return_value = GetTriggerScriptIntVariable("trigger1", "a")
end

function Update()

end
```

Assuming that the variable `"a"` is defined with the value *1* in the script attached to the trigger object `"trigger1"`, `GetTriggerScriptIntVariable` function returns the value *1*.

4.176. GetTriggerScriptStringVariable

Definition

`string GetTriggerScriptStringVariable(string triggerName, string variable)`

Description

This function gets the value of the String **`variable`** defined in the script attached to the **`triggerName`** trigger object.

Parameters

`triggerName`
Specifies the name of the trigger object.

`variable`
Specifies the name of the String variable defined in the script attached to the **`triggerName`** trigger object.

Return Value

Returns the value of the String **`variable`** defined in the script attached to the **`triggerName`** trigger object.

Example

```
--script name is GetTriggerScriptStringVariable.lua attached a to game object such as 
water
return_value = ""

function Init()
    return_value = GetTriggerScriptStringVariable("trigger1", "a")
end

function Update()

end
```

Assuming that the variable **`"a"`** is defined with the value "*hello*" in the script attached to the trigger object **`"trigger1"`**, **`GetTriggerScriptStringVariable`** function returns the value "*hello*".

4.177. GetVideoDuration

Definition

```
double GetVideoDuration(string videoName)
```

Description

This function returns the duration of `videoName` video object.

Parameters

videoName

Specifies the video name. You can also use the name "this" for this parameter. In this case, "this" string refers to the name of the video to which this script is attached.

Return Value

Duration of video object.

Example 1

```
duration = 0.0

function Init()
    duration = GetVideoDuration("video1")

    message = string.format("\nVideo duration is (%.2f) seconds", duration)
    PrintConsole(message)
end

function Update()

end
```

First, we get the duration of video `"video1"`. Then we display the duration in the console using the `PrintConsole` function.

Example 2

```
--Name of script is GetVideoDuration2.lua

duration = 0.0

function Init()
    duration = GetVideoDuration("this")

    message = string.format("\nVideo duration is (%.2f) seconds", duration)
    PrintConsole(message)
end

function Update()

end
```

Assume that the above script named `GetVideoDuration2.lua` is attached to a video object named "video1". In this case, string `"this"` in the `GetVideoDuration` function will be equal to "video1". In our example, the function `GetVideoDuration` returns the duration of current video, which is "video1".

4.178. GetVideoLoop

Definition

```
bool GetVideoLoop(string videoName)
```

Description

This function returns the state of the video loop as a Boolean value of true or false.

Parameters

videoName

Specifies the video name. You can also use the name "this" for this parameter. In this case, "this" string refers to the name of the video name to which this script is attached.

Return Value

If the state of the loop is true, it returns *true,* otherwise it returns *false*.

Example 1

```
video_loop = false
message = ""

function Init()
    video_loop = GetVideoLoop("video1")

    if video_loop then
          message = string.format("\nVideo Loop is ON")
    else
          message = string.format("\nVideo Loop is OFF")
    end

    PrintConsole(message)
end

function Update()

end
```

First, we specify the loop state of `"video1"`. Then we display loop status in the console using the `PrintConsole` function.

Example 2

```
--Name of script is GetVideoLoop2.lua

video_loop = false
message = ""

function Init()
    video_loop = GetVideoLoop("this")

    if video_loop then
          message = string.format("\nVideo Loop is ON")
    else
          message = string.format("\nVideo Loop is OFF")
    end

    PrintConsole(message)
end

function Update()

end
```

Assume that the above script named GetVideoLoop2.lua is attached to a video object named "video1". In this case, string "this" in the GetVideoLoop function will be equal to "video1". In our example, the function GetVideoLoop returns the loop state of the video "video1".

4.179. GetVideoPlay

Definition

```
bool GetVideoPlay(string videoName)
```

Description

This function returns the video playback status as a Boolean value of true or false.

Parameters

videoName

Specifies the video name. You can also use the name "this" for this parameter. In this case, "this" string refers to the name of the video to which this script is attached.

Return Value

If the video is playing, it returns *true,* otherwise it returns *false*.

Example 1

```
video_play = false
message = ""

function Init()
    video_play = GetVideoPlay("video1")

    if video_play then
          message = string.format("\nVideo is playing")
    else
          message = string.format("\nVideo isn't playing")
    end

    PrintConsole(message)
end

function Update()

end
```

First, we specify the playback state of `"video1"`. Then we display its status in the console using the `PrintConsole` function.

Example 2

```
--Name of script is GetVideoPlay2.lua

video_play = false
message = ""

function Init()
    video_play = GetVideoPlay("this")

    if video_play then
          message = string.format("\nVideo is playing")
    else
          message = string.format("\nVideo isn't playing")
    end

    PrintConsole(message)
end

function Update()

end
```

Assume that the above script named `GetVideoPlay2.lua` is attached to a video object named "video1". In this case, string `"this"` in the `GetVideoPlay` function will be equal to "video1". In our example, the function `GetVideoPlay` returns the playback state of the video "video1".

4.180. GetVideoScriptBoolVariable

Definition

`bool GetVideoScriptBoolVariable(string videoName, string variable)`

Description

This function gets the value of the Boolean **`variable`** defined in the script attached to the **`videoName`** video object.

Parameters

`videoName`
Specifies the name of the video object.

`variable`
Specifies the name of the Boolean variable defined in the script attached to the **`videoName`** video object.

Return Value

Returns the value of the Boolean **`variable`** defined in the script attached to the **`videoName`** video object.

Example

```
--script name is GetVideoScriptBoolVariable.lua attached a to game object such as light
value = false
function Init()
    value = GetVideoScriptBoolVariable("video1", "a")
end

function Update()

end
```

Assuming that the variable **`"a"`** is defined with the value *true* in the script attached to the video object **`"video1"`**, **`GetVideoScriptBoolVariable`** function returns the value *true*.

4.181. GetVideoScriptDoubleVariable

Definition

`double GetVideoScriptDoubleVariable(string videoName, string variable)`

Description

This function gets the value of the Double **`variable`** defined in the script attached to the **`videoName`** video object.

Parameters

`videoName`

Specifies the name of the video object.

`variable`

Specifies the name of the Double variable defined in the script attached to the **`videoName`** video object.

Return Value

Returns the value of the Double **`variable`** defined in the script attached to the **`videoName`** video object.

Example

```
--script name is GetVideoScriptDoubleVariable.lua attached a to game object such as light
return_value = 0.0

function Init()
    return_value = GetVideoScriptDoubleVariable("video1", "a")
end

function Update()

end
```

Assuming that the variable `"a"` is defined with the value *1.0* in the script attached to the video object `"video1"`, `GetVideoScriptDoubleVariable` function returns the value *1.0*.

4.182. GetVideoScriptIntVariable

Definition

`int GetVideoScriptIntVariable(string videoName, string variable)`

Description

This function gets the value of the Integer **`variable`** defined in the script attached to the **`videoName`** video object.

Parameters

`videoName`
Specifies the name of the video object.

`variable`
Specifies the name of the Integer variable defined in the script attached to the **`videoName`** video object.

Return Value

Returns the value of the Integer **`variable`** defined in the script attached to the **`videoName`** video object.

Example

```
--script name is GetVideoScriptIntVariable.lua attached a to game object such as light
return_value = 0

function Init()
    return_value = GetVideoScriptIntVariable("video1", "a")
end

function Update()

end
```

Assuming that the variable `"a"` is defined with the value *1* in the script attached to the video object `"video1"`, `GetVideoScriptIntVariable` function returns the value *1*.

4.183. GetVideoScriptStringVariable

Definition

```
string GetVideoScriptStringVariable(string videoName, string variable)
```

Description

This function gets the value of the String **`variable`** defined in the script attached to the **`videoName`** video object.

Parameters

`videoName`

Specifies the name of the video object.

`variable`

Specifies the name of the String variable defined in the script attached to the **`videoName`** video object.

Return Value

Returns the value of the String **`variable`** defined in the script attached to the **`videoName`** video object.

Example

```
--script name is GetVideoScriptStringVariable.lua attached a to game object such as light
return_value = ""

function Init()
    return_value = GetVideoScriptStringVariable("video1", "a")
end

function Update()

end
```

Assuming that the variable **`"a"`** is defined with the value "*hello*" in the script attached to the video object **`"video1"`**, **`GetVideoScriptStringVariable`** function returns the value "*hello*".

4.184. GetVideoVolume

Definition

```
double GetVideoVolume(string videoName)
```

Description

This function returns the audio volume of video `videoName`.

Parameters

`videoName`

Specifies the video name. You can also use the name "this" for this parameter. In this case, "this" string refers to the name of the video to which this script is attached.

Return Value

Audio volume of video `videoName` .

Example 1

```
volume = 0.0

function Init()
    volume = GetVideoVolume("video1")

    message = string.format("\nVideo volume is > %.2f", volume)
    PrintConsole(message)
end

function Update()

end
```

First, we get the volume of video `"video1"`. Then we display it in the console using the `PrintConsole` function.

Example 2

```
--Name of script is GetVideoVolume2.lua

volume = 0.0

function Init()
    volume = GetVideoVolume("this")

    message = string.format("\nVideo volume is > %.2f", volume)
    PrintConsole(message)
end

function Update()

end
```

Assume that the above script named `GetVideoVolume2.lua` is attached to a video object named "video1". In this case, string `"this"` in the `GetVideoVolume` function will be equal to "video1". In our example, the function `GetVideoVolume` returns the volume of current video, which is "video1".

4.185. GetVSceneScriptBoolVariable

Definition

```
bool GetVSceneScriptBoolVariable(string variable)
```

Description

This function gets the value of the Boolean **`variable`** defined in the script attached to the VScene Script object.

Parameters

`variable`

Specifies the name of the Boolean variable defined in the script attached to the VScene Script object.

Return Value

Returns the value of the Boolean **`variable`** defined in the script attached to the VScene Script object.

Example

```lua
--script name is GetVSceneScriptBoolVariable.lua attached a to game object such as water
value = false
function Init()
    value = GetVSceneScriptBoolVariable("a")
end

function Update()

end
```

Assuming that the variable `"a"` is defined with the value *true* in the script attached to the VScene Script object, `GetVSceneScriptBoolVariable` function returns the value *true*.

4.186. GetVSceneScriptDoubleVariable

Definition

`double GetVSceneScriptDoubleVariable(string variable)`

Description

This function gets the value of the Double **`variable`** defined in the script attached to the VScene Script object.

Parameters

`variable`

Specifies the name of the Double variable defined in the script attached to the VScene Script object.

Return Value

Returns the value of the Double **`variable`** defined in the script attached to the VScene Script object.

Example

```
--script name is GetVSceneScriptDoubleVariable.lua attached a to game object such as
water
return_value = 0.0

function Init()
    return_value = GetVSceneScriptDoubleVariable("a")
end

function Update()

end
```

Assuming that the variable **`"a"`** is defined with the value *1.0* in the script attached to the VScene Script object, **`GetVSceneScriptDoubleVariable`** function returns the value *1.0*.

4.187. GetVSceneScriptIntVariable

Definition

`int GetVSceneScriptIntVariable(string variable)`

Description

This function gets the value of the Integer **`variable`** defined in the script attached to the VScene Script object.

Parameters

`variable`

Specifies the name of the Integer variable defined in the script attached to the VScene Script object.

Return Value

Returns the value of the Integer **`variable`** defined in the script attached to the VScene Script object.

Example

```
--script name is GetVSceneScriptIntVariable.lua attached a to game object such as water
return_value = 0

function Init()
    return_value = GetVSceneScriptIntVariable("a")
end

function Update()

end
```

Assuming that the variable `"a"` is defined with the value *1* in the script attached to the VScene Script object, `GetVSceneScriptIntVariable` function returns the value *1*.

4.188. GetVSceneScriptStringVariable

Definition

```
string GetVSceneScriptStringVariable(string variable)
```

Description

This function gets the value of the String `variable` defined in the script attached to the VScene Script object.

Parameters

`variable`

Specifies the name of the String variable defined in the script attached to the VScene Script object.

Return Value

Returns the value of the String `variable` defined in the script attached to the VScene Script object.

Example

```
--script name is GetVSceneScriptStringVariable.lua attached a to game object such as
water
return_value = ""

function Init()
    return_value = GetVSceneScriptStringVariable("a")
end

function Update()

end
```

Assuming that the variable `"a"` is defined with the value "*hello*" in the script attached to the VScene Script object, `GetVSceneScriptStringVariable` function returns the value "*hello*".

4.189. GetWaterFlowSpeed

Definition

```
double GetWaterFlowSpeed(string waterName)
```

Description

This function returns the flow speed of water object `waterName`.

Parameters

`waterName`

Specifies the water name. You can also use the name "this" for this parameter. In this case, "this" string refers to the name of the water to which this script is attached.

Return Value

Flow speed of water object.

Example 1

```
speed = 0.0

function Init()
    speed = GetWaterFlowSpeed("water1")

    message = string.format("\nWater flow speed is > %.2f", speed)
    PrintConsole(message)
end

function Update()

end
```

First, we get the flow speed of water `"water1"`. Then we display the water flow speed in the console using the `PrintConsole` function.

Example 2

```
--Name of script is GetWaterFlowSpeed2.lua

speed = 0.0

function Init()
    speed = GetWaterFlowSpeed("this")

    message = string.format("\nWater flow speed is > %.2f", speed)
    PrintConsole(message)
end

function Update()

end
```

Assume that the above script named `GetWaterFlowSpeed2.lua` is attached to a water object named "water1". In this case, string `"this"` in the `GetWaterFlowSpeed` function will be equal to "water1". In our example, the function `GetWaterFlowSpeed` returns the flow speed of current water, which is "water1".

4.190. GetWaterLightPosition

Definition

```
double,double,double GetWaterLightPosition(string waterName)
```

Description

This function receives the name of the water `waterName` and returns its light (sun) position as three values x, y and z.

Parameters

`waterName`

Specifies the name of the water. You can also use the name "this" for this parameter. In this case, "this" refers to the water that this script is attached to.

Return Value

This function returns the position of water light as three values x, y and z.

Example 1

```
posX = 0.0
posY = 0.0
posZ = 0.0

function Init()
    posX, posY, posZ = GetWaterLightPosition("water1")

    message = string.format("\nWater light position is > (%.2f, %.2f, %.2f)" , posX,
posY, posZ)
    PrintConsole(message)
end

function Update()

end
```

First, `GetWaterLightPosition` function returns the light position of water `"water1"`. Then we display the water's light position values in the console using the `PrintConsole` function.

Example 2

```
--Name of script is GetWaterLightPosition2.lua

posX = 0.0
posY = 0.0
posZ = 0.0

function Init()
    posX, posY, posZ = GetWaterLightPosition("this")

    message = string.format("\nWater light position is > (%.2f, %.2f, %.2f)" , posX,
posY, posZ)
    PrintConsole(message)
```

```lua
end

function Update()

end
```

Assume that the above script named `GetWaterLightPosition2.lua` is attached to a water object named "water1". In this case, string `"this"` in the `GetWaterLightPosition` function will be equal to "water1". In our example, the function `GetWaterLightPosition` returns the light position of current water, which is "water1".

4.191. GetWaterPosition

Definition

`double,double,double GetWaterPosition(string waterName)`

Description

This function receives the name of the water `waterName` and returns its position as three values x, y and z.

Parameters

`waterName`

Specifies the name of the water object. You can also use the name "this" for this parameter. In this case, "this" refers to the water name that this script is attached to.

Return Value

This function returns the water position as three values x, y and z.

Example 1

```
posX = 0.0
posY = 0.0
posZ = 0.0

function Init()
    posX, posY, posZ = GetWaterPosition("water1")

    message = string.format("\nWater position is > (%.2f, %.2f, %.2f)" , posX, posY,
posZ)
    PrintConsole(message)
end

function Update()

end
```

First, `GetWaterPosition` function returns the position of water `"water1"`. Then we display the position values in the console using the `PrintConsole` function.

Example 2

```
--Name of script is GetWaterPosition2.lua

posX = 0.0
posY = 0.0
posZ = 0.0

function Init()
    posX, posY, posZ = GetWaterPosition("this")

    message = string.format("\nWater position is > (%.2f, %.2f, %.2f)" , posX, posY,
posZ)
    PrintConsole(message)
```

```
end

function Update()

end
```

Assume that the above script named `GetWaterPosition2.lua` is attached to a water object named "water1". In this case, string `"this"` in the `GetWaterPosition` function will be equal to "water1". In our example, the function `GetWaterPosition` returns the position of current water, which is "water1". Then we display the position values in the console using the `PrintConsole` function.

4.192. GetWaterRotation

Definition

```
double GetWaterRotation(string waterName)
```

Description

This function returns the rotation of water `waterName` around Y axis in degrees.

Parameters

`waterName`

Specifies the water name. You can also use the name "this" for this parameter. In this case, "this" string refers to the name of the water to which this script is attached.

Return Value

Rotation of water `waterName` around Y axis in degrees.

Example 1

```
rotation = 0.0

function Init()
    rotation = GetWaterRotation("water1")

    message = string.format("\nWater rotation is > %.2f", rotation)
    PrintConsole(message)
end

function Update()

end
```

First, we get the rotation of water `"water1"` around Y axis. Then we display it in the console using the `PrintConsole` function.

Example 2

```
--Name of script is GetWaterRotation2.lua

rotation = 0.0

function Init()
    rotation = GetWaterRotation("this")

    message = string.format("\nWater rotation is > %.2f", rotation)
    PrintConsole(message)
end

function Update()

end
```

Assume that the above script named `GetWaterRotation2.lua` is attached to a water object named "water1". In this case, string `"this"` in the `GetWaterRotation` function will be equal to "water1". In our example, the function `GetWaterRotation` returns the Y rotation of current water, which is "water1".

4.193. GetWaterScale

Definition

```
double,double GetWaterScale(string waterName)
```

Description

This function receives the name of the water `waterName` and returns its scale as two values in the x and z direction.

Parameters

`waterName`

Specifies the name of the water object. You can also use the name "this" for this parameter. In this case, "this" refers to the water name that this script is attached to.

Return Value

This function returns the water scale as two values in the x and z direction.

Example 1

```
scaleX = 0.0
scaleZ = 0.0

function Init()
    scaleX, scaleZ = GetWaterScale("water1")

    message = string.format("\nWater scale is > (%.2f, %.2f)" , scaleX, scaleZ)
    PrintConsole(message)
end

function Update()

end
```

First, `GetWaterScale` function returns the scale of water `"water1"`. Then we display the scale values in x and z direction in the console using the `PrintConsole` function.

Example 2

```
--Name of script is GetWaterScale2.lua

scaleX = 0.0
scaleZ = 0.0

function Init()
    scaleX, scaleZ = GetWaterScale("this")

    message = string.format("\nWater scale is > (%.2f, %.2f)" , scaleX, scaleZ)
    PrintConsole(message)
end

function Update()
```

```
end
```

Assume that the above script named GetWaterScale2.lua is attached to a water object named "water1". In this case, string "this" in the GetWaterScale function will be equal to "water1". In our example, the function GetWaterScale returns the scale of current water, which is "water1". Then we display the scale values in the console using the PrintConsole function.

4.194. GetWaterScriptBoolVariable

Definition

```
bool GetWaterScriptBoolVariable(string waterName, string variable)
```

Description

This function gets the value of the Boolean **`variable`** defined in the script attached to the water object `waterName`.

Parameters

`waterName`

Specifies the name of the water object.

`variable`

Specifies the name of the Boolean variable defined in the script attached to the water object `waterName`.

Return Value

Returns the value of the Boolean **`variable`** defined in the script attached to the water object `waterName`.

Example

```
--script name is GetWaterScriptBoolVariable.lua attached a to game object such as light
value = false
function Init()
    value = GetWaterScriptBoolVariable("water1", "a")
end

function Update()

end
```

Assuming that the variable `"a"` is defined with the value *true* in the script attached to the water object `"water1"`, `GetWaterScriptBoolVariable` function returns the value *true*.

4.195. GetWaterScriptDoubleVariable

Definition

`double GetWaterScriptDoubleVariable(string waterName, string variable)`

Description

This function gets the value of the Double **`variable`** defined in the script attached to the water object **`waterName`**.

Parameters

`waterName`

Specifies the name of the water object.

`variable`

Specifies the name of the Double variable defined in the script attached to the water object **`waterName`**.

Return Value

Returns the value of the Double **`variable`** defined in the script attached to the water object **`waterName`**.

Example

```
--script name is GetWaterScriptDoubleVariable.lua attached a to game object such as light
return_value = 0.0

function Init()
    return_value = GetWaterScriptDoubleVariable("water1", "a")
end

function Update()

end
```

Assuming that the variable `"a"` is defined with the value *1.0* in the script attached to the water object `"water1"`, `GetWaterScriptDoubleVariable` function returns the value *1.0*.

4.196. GetWaterScriptIntVariable

Definition

```
int GetWaterScriptIntVariable(string waterName, string variable)
```

Description

This function gets the value of the Integer **`variable`** defined in the script attached to the water object **`waterName`**.

Parameters

`waterName`
Specifies the name of the water object.

`variable`
Specifies the name of the Integer variable defined in the script attached to the water object **`waterName`**.

Return Value

Returns the value of the Integer **`variable`** defined in the script attached to the water object **`waterName`**.

Example

```
--script name is GetWaterScriptIntVariable.lua attached a to game object such as light
return_value = 0

function Init()
    return_value = GetWaterScriptIntVariable("water1", "a")
end

function Update()

end
```

Assuming that the variable `"a"` is defined with the value *1* in the script attached to the water object `"water1"`, `GetWaterScriptIntVariable` function returns the value *1*.

4.197. GetWaterScriptStringVariable

Definition

```
string GetWaterScriptStringVariable(string waterName, string variable)
```

Description

This function gets the value of the String `variable` defined in the script attached to the water object `waterName`.

Parameters

`waterName`

Specifies the name of the water object.

`variable`

Specifies the name of the String variable defined in the script attached to the water object `waterName`.

Return Value

Returns the value of the String `variable` defined in the script attached to the water object `waterName`.

Example

```
--script name is GetWaterScriptStringVariable.lua attached a to game object such as light
return_value = ""

function Init()
    return_value = GetWaterScriptStringVariable("water1", "a")
end

function Update()

end
```

Assuming that the variable `"a"` is defined with the value "*hello*" in the script attached to the water object `"water1"`, `GetWaterScriptStringVariable` function returns the value "*hello*".

4.198. GetWaterTransparency

Definition

`double GetWaterTransparency(string waterName)`

Description

This function returns the transparency of water object `waterName`.

Parameters

`waterName`

Specifies the water name. You can also use the name "this" for this parameter. In this case, "this" string refers to the name of the water to which this script is attached.

Return Value

Transparency of water object.

Example 1

```lua
transparency = 0.0

function Init()
    transparency = GetWaterTransparency("water1")

    message = string.format("\nWater transparency is > %.2f", transparency)
    PrintConsole(message)
end

function Update()

end
```

First, we get the transparency of water `"water1"`. Then we display the water transparency in the console using the `PrintConsole` function.

Example 2

```lua
--Name of script is GetWaterTransparency2.lua

transparency = 0.0

function Init()
    transparency = GetWaterTransparency("this")

    message = string.format("\nWater transparency is > %.2f", transparency)
    PrintConsole(message)
end

function Update()

end
```

Assume that the above script named `GetWaterTransparency2.lua` is attached to a water object named "water1". In this case, string `"this"` in the `GetWaterTransparency` function will be equal to "water1". In our example, the function `GetWaterTransparency` returns the transparency of current water, which is "water1".

4.199. GetWaterUnderwaterColor

Definition

`double,double,double GetWaterUnderwaterColor(string waterName)`

Description

This function returns the underwater color of water **`waterName`** as three values of red, green and blue. Each value ranges from 0 to 1.

Parameters

`waterName`

Specifies the name of the water object. You can also use the name "this" for this parameter. In this case, "this" refers to the water object name to which this script is attached.

Return Value

Returns the underwater color of water **`waterName`** as three values of red, green and blue. Each value ranges from 0 to 1.

Example 1

```
red = 0.0
green = 0.0
blue = 0.0

function Init()
    red, green, blue = GetWaterUnderwaterColor("water1")

    message = string.format("\nUnderwater color of water is > (%.2f, %.2f, %.2f)" , red,
green, blue)
    PrintConsole(message)
end

function Update()

end
```

In this example, the `GetWaterUnderwaterColor` function returns the value of the red, green, and blue components of the underwater color of water `"water1"`. Then these three values are displayed on the console by the `PrintConsole` function.

Example 2

```
--Name of script is GetWaterUnderwaterColor2.lua

red = 0.0
green = 0.0
blue = 0.0

function Init()
    red, green, blue = GetWaterUnderwaterColor("this")
    message = string.format("\nUnderwater color of water is > (%.2f, %.2f, %.2f)" , red,
green, blue)
    PrintConsole(message)
end

function Update()

end
```

Assume that the above script named `GetWaterUnderwaterColor2.lua` is attached to a water object named "water1". In this case, string `"this"` in the `GetWaterUnderwaterColor` function will be equal to "water1". In our example, the function `GetWaterUnderwaterColor` returns three values of red, green and blue underwater color of the water "water1".

4.200. GetWaterUnderwaterFogDensity

Definition

```
double GetWaterUnderwaterFogDensity(string waterName)
```

Description

This function returns the underwater fog density of water object `waterName`.

Parameters

`waterName`

Specifies the water name. You can also use the name "this" for this parameter. In this case, "this" string refers to the name of the water to which this script is attached.

Return Value

Underwater fog density of water object.

Example 1

```
fog_density = 0.0

function Init()
    fog_density = GetWaterUnderwaterFogDensity("water1")

    message = string.format("\nUnderwater fog density of water is > %.2f", fog_density)
    PrintConsole(message)
end

function Update()

end
```

First, we get the underwater fog density of water `"water1"`. Then we display it in the console using the `PrintConsole` function.

Example 2

```
--Name of script is GetWaterUnderwaterFogDensity2.lua

fog_density = 0.0

function Init()
    fog_density = GetWaterUnderwaterFogDensity("this")

    message = string.format("\nUnderwater fog density of water is > %.2f", fog_density)
    PrintConsole(message)
end

function Update()

end
```

Assume that the above script named `GetWaterUnderwaterFogDensity2.lua` is attached to a water object named "water1". In this case, string `"this"` in the `GetWaterUnderwaterFogDensity` function will be equal to "water1". In our example, the function `GetWaterUnderwaterFogDensity` returns the underwater fog density of current water, which is "water1".

4.201. GetWaterUV

Definition

```
double GetWaterUV(string waterName)
```

Description

This function returns the texture UV of water object `waterName`.

Parameters

`waterName`

Specifies the water name. You can also use the name "this" for this parameter. In this case, "this" string refers to the name of the water to which this script is attached.

Return Value

Texture UV of water object.

Example 1

```
UV = 0.0

function Init()
    UV = GetWaterUV("water1")

    message = string.format("\nWater UV is > %.2f", UV)
    PrintConsole(message)
end

function Update()

end
```

First, we get the texture UV of water `"water1"`. Then we display it in the console using the `PrintConsole` function.

Example 2

```
--Name of script is GetWaterUV2.lua

UV = 0.0

function Init()
    UV = GetWaterUV("this")

    message = string.format("\nWater UV is > %.2f", UV)
    PrintConsole(message)
end

function Update()

end
```

Assume that the above script named `GetWaterUV2.lua` is attached to a water object named "water1". In this case, string `"this"` in the `GetWaterUV` function will be equal to "water1". In our example, the function `GetWaterUV` returns the texture UV of current water, which is "water1".

4.202. HideCursorIcon

Definition

`HideCursorIcon(string resourceDirectoryName_resourceFileName.dds)`

Description

This function hides the resource image `resourceDirectoryName_resourceFileName.dds`. To find the resource name in this function, first go to Script Editor (Tools > Script Editor). Then, use the Tools > Script Utility menu to open the Script Utility dialog and press the Project Resource button. You can now see all the resources in Script Utility dialog. In this dialog, you can find the desired resource image and click on the Copy Folder_File Name button to copy its full name. Then paste this name into the `HideCursorIcon` function. In order for the `HideCursorIcon` function to recognize this name, you must first have loaded the resource image through the `LoadResource` function (see the example).

Parameters

`resourceDirectoryName_resourceFileName.dds`
Specifies the full name of the resource image.

Example

```
timer = 0.0
hidden = false

function Init()
    LoadResource("Images", "Cursor.dds")
    ShowCursorIcon("Images_Cursor.dds", 5.0)
end

function Update()
    if timer < 5.0 then timer = timer + GetElapsedTime() end

    if timer >= 5.0 and not hidden then
        HideCursorIcon("Images_Cursor.dds")
        hidden = true
    end
end
```

First, using the `LoadResource` function, we load the `"Cursor.dds"` image located in the `"Images"` folder. Then we display this image using the `ShowCursorIcon` function. After `5.0` seconds have passed in the `Update()` event, we hide this resource image using the `HideCursorIcon` function.

4.203. HideGUI

Definition

`HideGUI(string guiName)`

Description

This function hides the GUI **`guiName`**.

Parameters

`guiName`

Specifies the GUI name.

Example

```
function OnTriggerEnter(otherActorName)
    HideGUI("gui_SampleGUI17_MainMenu")
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)

end
```

Assume that the above script is attached to a trigger named "trigger1". Whenever the main character or a prefab instance that has dynamic physics is entered into this trigger, the `"gui_SampleGUI17_MainMenu"` GUI will be hidden.

4.204. HideGUIButton

Definition

`HideGUIButton(string GUIName, string buttonName)`

Description

This function hides the button **`buttonName`** that belongs to the GUI **`GUIName`**.

Parameters

`GUIName`

Specifies the GUI name.

`buttonName`

Specifies the button name that belongs to the GUI **`GUIName`**.

Example

```
function OnTriggerEnter(otherActorName)
    HideGUIButton("gui_SampleGUI17_MainMenu", "PlayGame")
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)

end
```

Assume that the above script is attached to a trigger named "trigger1". Whenever the main character or a prefab instance that has dynamic physics is entered into this trigger, the button `"PlayGame"` that belongs to GUI `"gui_SampleGUI17_MainMenu"` will be hidden.

4.205. HideGUIImage

Definition

`HideGUIImage(string GUIName, string imageName)`

Description

This function hides the image `imageName` that belongs to the GUI `GUIName`.

Parameters

`GUIName`
Specifies the GUI name.

`imageName`
Specifies the image name that belongs to the GUI `GUIName`.

Example

```
function OnTriggerEnter(otherActorName)
    HideGUIImage("gui_SampleGUI17_MainMenuAbout", "backgroundImg")
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)

end
```

Assume that the above script is attached to a trigger named "trigger1". Whenever the main character or a prefab instance that has dynamic physics is entered into this trigger, the image `"backgroundImg"` that belongs to GUI `"gui_SampleGUI17_MainMenuAbout"` will be hidden.

4.206. HideGUIText

Definition

`HideGUIText(string GUIName, string textName)`

Description

This function hides the text **`textName`** that belongs to the GUI **`GUIName`**.

Parameters

`GUIName`
Specifies the GUI name.

`textName`
Specifies the text name that belongs to the GUI **`GUIName`**.

Example

```
function OnTriggerEnter(otherActorName)
    HideGUIText("gui_SampleGUI17_MainMenuAbout", "text1")
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)

end
```

Assume that the above script is attached to a trigger named "trigger1". Whenever the main character or a prefab instance that has dynamic physics is entered into this trigger, the text `"text1"` that belongs to GUI `"gui_SampleGUI17_MainMenuAbout"` will be hidden.

4.207. HideMenuCursor

Definition

```
HideMenuCursor()
```

Description

This function hides the menu cursor image. You can change the menu cursor image and its properties through the Current VScene Properties dialog (Tools > Current VScene Properties).

Example

```
function OnTriggerEnter(otherActorName)
    HideMenuCursor()
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)

end
```

Assume that the above script is attached to a trigger named "trigger1". Whenever the main character or a prefab instance that has dynamic physics is entered into this trigger, the menu cursor image will be hidden.

4.208. HidePrefabInstance

Definition

`HidePrefabInstance(string prefabInstanceName)`

Description

This function hides the prefab instance `prefabInstanceName`. To view the name of prefab instances, open the VScene and click on the desired Prefab Instance in the "Prefabs and GUIs" section and press the Edit button. You can also access the names of prefab instances from the Script Utility section of the script editor (Tools > Script Editor > Tools > Script Utility). In the dialog that appears, you can view and copy the name of the prefab instance.

Parameters

prefabInstanceName

Specifies the name of the prefab instance. You can also use the name "this" for this parameter. In this case, "this" refers to the prefab instance that this script is attached to.

Example 1

```
timer = 0.0
hidden = false

function Init()
end

function Update()
    timer = timer + GetElapsedTime()
    if timer >= 5.0 and not hidden then
        HidePrefabInstance("1_VandaEngine17-SamplePack1_eggbox")
        hidden = true
    end
end
```

After `5.0` seconds, `HidePrefabInstance` function will hide `"1_VandaEngine17-SamplePack1_eggbox"` prefab instance.

Example 2

```
--name of the script is HidePrefabInstance2.lua

timer = 0.0
hidden = false

function Init()
end

function Update()
    timer = timer + GetElapsedTime()
    if timer >= 5.0 and not hidden then
        HidePrefabInstance("this")
        hidden = true
    end
end
```

If, in the Prefab Editor, you attach `HidePrefabInstance2.lua` script to a Prefab, then `"this"` parameter in the `HidePrefabInstance` function will point to instances of that Prefab in current VScene. For example, if you have an Instance named *instance1_a* from a Prefab named *a* to which this script is attached, `"this"` in `HidePrefabInstance` function refers to the name *instance1_a*.

This script hides current prefab instance after `5.0` seconds.

4.209. IsCharacterControllerLocked

Definition

```
bool IsCharacterControllerLocked()
```

Description

This function determines whether the character controller is locked or not.

Return Value

If character controller is locked, it returns *true,* otherwise it returns *false*.

Example

```
locked = false
message = ""

function Init()
    locked = IsCharacterControllerLocked()

    if locked then
          message = string.format("\nCharacter controller is locked")
    else
          message = string.format("\nCharacter controller isn't locked")
    end

    PrintConsole(message)
end

function Update()

end
```

First, we determines whether the character controller is locked or not. Then we display its lock status in the console using the `PrintConsole` function.

4.210. IsGeneralWaterReflectionEnabled

Definition

```
bool IsGeneralWaterReflectionEnabled()
```

Description

This function determines whether the general water reflection is enabled or not.

Return Value

If general water reflection is enabled, it returns *true,* otherwise it returns *false*.

Example

```
reflection = false
message = ""

function Init()
    reflection = IsGeneralWaterReflectionEnabled()

    if reflection then
          message = string.format("\nGeneral water reflection is ON")
    else
          message = string.format("\nGeneral water reflection is OFF")
    end

    PrintConsole(message)
end

function Update()

end
```

First, we determines whether the general water reflection is enabled or not. Then we display general water reflection status in the console using the `PrintConsole` function.

4.211. IsKeyDown

Definition

```
bool IsKeyDown(string DirectInputKeyCode)
```

Description

This function determines whether the key `DirectInputKeyCode` is down or not.

Return Value

If key `DirectInputKeyCode` is down, it returns *true,* otherwise it returns *false*. Accepted string are:

string	Meaning
"DIK_ESCAPE"	Esc
"DIK_1"	1
"DIK_2"	2
"DIK_3"	3
"DIK_4"	4
"DIK_5"	5
"DIK_6"	6
"DIK_7"	7
"DIK_8"	8
"DIK_9"	9
"DIK_0"	0
"DIK_MINUS"	-
"DIK_EQUALS"	=
"DIK_BACK"	Back Space
"DIK_TAB"	Tab
"DIK_Q"	Q
"DIK_W"	W
"DIK_E"	E
"DIK_R"	R
"DIK_T"	T
"DIK_Y"	Y
"DIK_U"	U
"DIK_I"	I
"DIK_O"	O
"DIK_P"	P
"DIK_LBRACKET"	[
"DIK_RBRACKET"	]
"DIK_RETURN"	Enter
"DIK_LCONTROL"	Ctrl (Left)
"DIK_A"	A
"DIK_S"	S
"DIK_D"	D
"DIK_F"	F
"DIK_G"	G
"DIK_H"	H
"DIK_J"	J

"DIK_K"	K	
"DIK_L"	L	
"DIK_SEMICOLON"	;	
"DIK_APOSTROPHE"	'	
"DIK_GRAVE"	`	
"DIK_LSHIFT"	Shift (Left)	
"DIK_BACKSLASH"	\	
"DIK_Z"	Z	
"DIK_X"	X	
"DIK_C"	C	
"DIK_V"	V	
"DIK_B"	B	
"DIK_N"	N	
"DIK_M"	M	
"DIK_COMMA"	,	
"DIK_PERIOD"	.	
"DIK_SLASH"	/	
"DIK_RSHIFT"	Shift (Right)	
"DIK_MULTIPLY"	* (Numpad)	
"DIK_LMENU"	Alt (Left)	
"DIK_SPACE"	Space	
"DIK_CAPITAL"	Caps Lock	
"DIK_F1"	F1	
"DIK_F2"	F2	
"DIK_F3"	F3	
"DIK_F4"	F4	
"DIK_F5"	F5	
"DIK_F6"	F6	
"DIK_F7"	F7	
"DIK_F8"	F8	
"DIK_F9"	F9	
"DIK_F10"	F10	
"DIK_NUMLOCK"	Num Lock	
"DIK_SCROLL"	Scroll Lock	
"DIK_NUMPAD7"	7 (Numpad)	
"DIK_NUMPAD8"	8 (Numpad)	
"DIK_NUMPAD9"	9 (Numpad)	
"DIK_SUBTRACT"	- (Numpad)	
"DIK_NUMPAD4"	4 (Numpad)	
"DIK_NUMPAD5"	5 (Numpad)	
"DIK_NUMPAD6"	6 (Numpad)	
"DIK_ADD"	+ (Numpad)	
"DIK_NUMPAD1"	1 (Numpad)	
"DIK_NUMPAD2"	2 (Numpad)	
"DIK_NUMPAD3"	3 (Numpad)	
"DIK_NUMPAD0"	0 (Numpad)	
"DIK_DECIMAL"	. (Numpad)	
"DIK_F11"	F11	
"DIK_F12"	F12	
"DIK_F13"	F13	NEC PC-98

"DIK_F14"	F14	NEC PC-98
"DIK_F15"	F15	NEC PC-98
"DIK_KANA"	Kana	Japenese Keyboard
"DIK_CONVERT"	Convert	Japenese Keyboard
"DIK_NOCONVERT"	No Convert	Japenese Keyboard
"DIK_YEN"	¥	Japenese Keyboard
"DIK_NUMPADEQUALS"	=	NEC PC-98
"DIK_CIRCUMFLEX"	^	Japenese Keyboard
"DIK_AT"	@	NEC PC-98
"DIK_COLON"	:	NEC PC-98
"DIK_UNDERLINE"	_	NEC PC-98
"DIK_KANJI"	Kanji	Japenese Keyboard
"DIK_STOP"	Stop	NEC PC-98
"DIK_AX"	(Japan AX)	
"DIK_UNLABELED"	(J3100)	
"DIK_NUMPADENTER"	Enter (Numpad)	
"DIK_RCONTROL"	Ctrl (Right)	
"DIK_NUMPADCOMMA"	, (Numpad)	NEC PC-98
"DIK_DIVIDE"	/ (Numpad)	
"DIK_SYSRQ"	Sys Rq	
"DIK_RMENU"	Alt (Right)	
"DIK_PAUSE"	Pause	
"DIK_HOME"	Home	
"DIK_UP"	↑	
"DIK_PRIOR"	Page Up	
"DIK_LEFT"	←	
"DIK_RIGHT"	→	
"DIK_END"	End	
"DIK_DOWN"	↓	
"DIK_NEXT"	Page Down	
"DIK_INSERT"	Insert	
"DIK_DELETE"	Delete	
"DIK_LWIN"	Windows	
"DIK_RWIN"	Windows	
"DIK_APPS"	Menu	
"DIK_POWER"	Power	
"DIK_SLEEP"	Windows	

Example

```lua
AkeyDown = false

function Init()

end

function Update()
    AkeyDown = IsKeyDown("DIK_A")

    if AkeyDown then
          PrintConsole("\nA key is down")
    end

end
```

4.212. IsPrefabInstanceMaterialEnabled

Definition

```
bool IsPrefabInstanceMaterialEnabled(string prefabInstanceName)
```

Description

This function determines whether the material of prefab instance **prefabInstanceName** is enabled or not.

Parameters

prefabInstanceName

Specifies the prefab instance name. You can also use the name "this" for this parameter. In this case, "this" string refers to the name of the prefab instance to which this script is attached.

Return Value

If material of prefab instance **prefabInstanceName** is enabled, it returns *true*, otherwise it returns *false*.

Example 1

```
enabled = false
message = ""

function Init()
    enabled = IsPrefabInstanceMaterialEnabled("1_VandaEngine17-SamplePack1_wood_pile")

    if enabled then
          message = string.format("\nPrefab instance material is enabled")
    else
          message = string.format("\nPrefab instance material is disabled")
    end

    PrintConsole(message)
end

function Update()

end
```

First, we determines whether the material of prefab instance "1_VandaEngine17-SamplePack1_wood_pile" is enabled or not. Then we display its result in the console using the PrintConsole function.

Example 2

```
--Name of script is IsPrefabInstanceMaterialEnabled2.lua

enabled = false
message = ""

function Init()
    enabled = IsPrefabInstanceMaterialEnabled("this")
    if enabled then
          message = string.format("\nPrefab instance material is enabled")
    else
          message = string.format("\nPrefab instance material is disabled")
    end

    PrintConsole(message)
end

function Update()

end
```

If, in the Prefab Editor, you attach IsPrefabInstanceMaterialEnabled2.lua script to a Prefab, then "this" parameter in the IsPrefabInstanceMaterialEnabled function will point to instances of that Prefab in current VScene. For example, if you have an Instance named *instance1_a* from a Prefab named *a* to which this script is attached, "this" in IsPrefabInstanceMaterialEnabled function refers to the name *instance1_a*.
In our example, the function IsPrefabInstanceMaterialEnabled determines whether the material of current prefab instance, which is prefab instance "*instance1_a*", is enabled or not. then we display its result in the console using the PrintConsole function.

4.213. IsSkyFogEnabled

Definition

```
bool IsSkyFogEnabled()
```

Description

This function determines whether the sky fog is enabled or not.

Return Value

If sky fog is enabled, it returns *true,* otherwise it returns *false*.

Example

```
skyFog = false
message = ""

function Init()
    skyFog = IsSkyFogEnabled()

    if skyFog then
          message = string.format("\nSky fog is ON")
    else
          message = string.format("\nSky fog is OFF")
    end

    PrintConsole(message)
end

function Update()

end
```

First, we determines whether the sky fog is enabled or not. Then we display sky fog status in the console using the `PrintConsole` function.

4.214. IsVSyncEnabled

Definition

```
bool IsVSyncEnabled()
```

Description

This function determines whether the VSync is enabled or not.

Return Value

If VSync is enabled, it returns *true,* otherwise it returns *false*.

Example

```
VSync = false
message = ""

function Init()
    VSync = IsVSyncEnabled()

    if VSync then
          message = string.format("\nVSync is ON")
    else
          message = string.format("\nVSync is OFF")
    end

    PrintConsole(message)
end

function Update()

end
```

First, we determines whether the VSync is enabled or not. Then we display VSync status in the console using the `PrintConsole` function.

4.215. IsWaterShadowEnabled

Definition

```
bool IsWaterShadowEnabled(string waterName)
```

Description

This function determines whether the shadow of reflections of water **waterName** is enabled or not.

Parameters

waterName

Specifies the water name. You can also use the name "this" for this parameter. In this case, "this" string refers to the name of the water to which this script is attached.

Return Value

If shadow of reflections of water ***waterName*** is enabled, it returns *true,* otherwise it returns *false*.

Example 1

```
waterShadow = false
message = ""

function Init()
    waterShadow = IsWaterShadowEnabled("water1")

    if waterShadow then
          message = string.format("\nWater shadow is enabled")
    else
          message = string.format("\nWater shadow is't enabled")
    end

    PrintConsole(message)
end

function Update()

end
```

First, we determines whether the shadow of reflections of water `"water1"` is enabled or not. Then we display its result in the console using the `PrintConsole` function.

Example 2

```
--Name of script is IsWaterShadowEnabled2.lua

waterShadow = false
message = ""

function Init()
    waterShadow = IsWaterShadowEnabled("this")

    if waterShadow then
          message = string.format("\nWater shadow is enabled")
    else
          message = string.format("\nWater shadow is't enabled")
    end

    PrintConsole(message)
end

function Update()

end
```

Assume that the above script named `IsWaterShadowEnabled2.lua` is attached to a water object named "water1". In this case, string `"this"` in the `IsWaterShadowEnabled` function will be equal to "water1". In our example, the function `IsWaterShadowEnabled` determines whether the shadow of reflections of current water, which is water "water1", is enabled or not. then we display its result in the console using the `PrintConsole` function.

4.216. IsWaterSunReflectionEnabled

Definition

```
bool IsWaterSunReflectionEnabled(string waterName)
```

Description

This function determines whether the sun reflection of water **waterName** is enabled or not.

Parameters

waterName

Specifies the water name. You can also use the name "this" for this parameter. In this case, "this" string refers to the name of the water to which this script is attached.

Return Value

If sun reflection of water ***waterName*** is enabled, it returns *true,* otherwise it returns *false*.

Example 1

```
waterSunReflection = false
message = ""

function Init()
    waterSunReflection = IsWaterSunReflectionEnabled("water1")

    if waterSunReflection then
          message = string.format("\nWater sun reflection is enabled")
    else
          message = string.format("\nWater sun reflection is't enabled")
    end

    PrintConsole(message)
end

function Update()

end
```

First, we determines whether the sun reflection of water **"water1"** is enabled or not. Then we display its result in the console using the **PrintConsole** function.

Example 2

```
--Name of script is IsWaterSunReflectionEnabled2.lua

waterSunReflection = false
message = ""

function Init()
    waterSunReflection = IsWaterSunReflectionEnabled("this")

    if waterSunReflection then
          message = string.format("\nWater sun reflection is enabled")
    else
          message = string.format("\nWater sun reflection is't enabled")
    end

    PrintConsole(message)
end

function Update()

end
```

Assume that the above script named IsWaterSunReflectionEnabled2.lua is attached to a water object named "water1". In this case, string "this" in the IsWaterSunReflectionEnabled function will be equal to "water1". In our example, the function IsWaterSunReflectionEnabled determines whether the sun reflection of current water, which is water "water1", is enabled or not. then we display its result in the console using the PrintConsole function.

4.217. IsWaterVisible

Definition

`bool IsWaterVisible(string waterName)`

Description

This function determines whether the water **`waterName`** is visible or not.

Parameters

`waterName`

Specifies the water name. You can also use the name "this" for this parameter. In this case, "this" string refers to the name of the water to which this script is attached.

Return Value

If water ***`waterName`*** is visible, it returns *true,* otherwise it returns *false*.

Example 1

```
water_visible = false
message = ""

function Init()
    water_visible = IsWaterVisible("water1")

    if water_visible then
          message = string.format("\nWater is visible")
    else
          message = string.format("\nWater is invisible")
    end

    PrintConsole(message)
end

function Update()

end
```

First, we determines whether the water `"water1"` is visible or not. Then we display its result in the console using the `PrintConsole` function.

Example 2

```
--Name of script is IsWaterVisible2.lua

water_visible = false
message = ""

function Init()
    water_visible = IsWaterVisible("this")

    if water_visible then
          message = string.format("\nWater is visible")
    else
          message = string.format("\nWater is invisible")
    end

    PrintConsole(message)
end

function Update()

end
```

Assume that the above script named `IsWaterVisible2.lua` is attached to a water object named "water1". In this case, string `"this"` in the `IsWaterVisible` function will be equal to "water1". In our example, the function `IsWaterVisible` determines whether the current water, which is water "water1", is visible or not. then we display its result in the console using the `PrintConsole` function.

4.218. LoadResource

Definition

`LoadResource(string resourceDirectoryName, string resourceFileName)`

Description

This function loads the resource `resourceFileName` located in the `resourceDirectoryName` folder. In order for this function to load the desired resource, you must first add it through the *Add Resource to Current Project* dialog (File > Project > Add/Remove Resource to/from Current Project). You can go to the *Project Resources* section through the Script Utility dialog (Tools > Script Editor > Tools > Script Utility) and copy the names of the resources.

Parameters

`resourceDirectoryName`
Specifies the name of the folder where resourceFileName is located.

`resourceFileName`
Specifies the name of the resource file.

Example

```
function Init()
    LoadResource("Images", "Cursor.dds")
    ShowCursorIcon("Images_Cursor.dds", 5.0)
end

function Update()

end
```

First, using the `LoadResource` function, we load the `"Cursor.dds"` file located in the `"Images"` folder. Then we display it using the `ShowCursorIcon` function.

4.219. LoadVScene

Definition

`LoadVScene(string VSceneName)`

Description

This function loads the VScene `VSceneName`. You can view and copy the desired VScene name through the Script Utility dialog (Tools > Script Editor > Tools > Script Utility).

Parameters

`VSceneName`

Specifies the VScene name.

Example

```
function OnTriggerEnter(otherActorName)
    LoadVScene("Sample17Level1")
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)

end
```

Assume that the above script is attached to a trigger named "trigger1". Whenever the main character or a prefab instance that has dynamic physics is entered into this trigger, the VScene `"Sample17Level1"` will be loaded.

4.220. LockCharacterController

Definition

```
LockCharacterController()
```

Description

This function locks physics character controller. In this case, you cannot move the main game character or the camera attached to it using the keyboard or mouse.

Example

```
function OnTriggerEnter(otherActorName)
    LockCharacterController()
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)

end
```

Assume that the above script is attached to a trigger named "trigger1". Whenever the main character or a prefab instance that has dynamic physics is entered into this trigger, the main character will be locked.

4.221. OpenFileForReading

Definition

`OpenFileForReading(string filePath)`

Description

This function opens a binary file for reading. After reading the information of this file, you should use the `CloseFile` function to close the file.

Parameters

`filePath`

Specifies the file path. This path is located in the Assets/Data/ folder.

Example

```
bVar = false
fVar = 0.0
iVar = 0
sVar = "init"

function Init()
   --Create a folder in Assets/Data/ path
   CreateFolder("Lev1")

   --Create and open file to write data
   OpenFileForWriting("Lev1/level1.bin")
   WriteBoolVariableToFile(true)
   WriteFloatVariableToFile(2.0)
   WriteIntVariableToFile(3)
   WriteStringVariableToFile("level1")
   CloseFile("Lev1/level1.bin")

   --Open File to load data
   OpenFileForReading("Lev1/level1.bin")
   bVar = ReadBoolVariableFromFile()
   fVar = ReadFloatVariableFromFile()
   iVar = ReadIntVariableFromFile()
   sVar = ReadStringVariableFromFile()
   CloseFile("Lev1/level1.bin")
end
```

First, using the `CreateFolder` function, we create a folder called `"Lev1"` in the Assets/Data/ path. Then, using the `OpenFileForWriting` function, we open the `level1.bin` file located in the Assets/Data/`Lev1`/ path for writing. After writing information to the file, we close it using the `CloseFile` function.
Then, using the `OpenFileForReading` function, we open the `level1.bin` file located in the path Assets/Data/`Lev1`/ for reading and read its information in the same order as we wrote. Finally, we close the file using the `CloseFile` function.

4.222. OpenFileForWriting

Definition

```
OpenFileForWriting(string filePath)
```

Description

This function opens a binary file for writing. After writing the information to this file, you should use the `CloseFile` function to close the file.

Parameters

filePath

Specifies the file path. This path is located in the Assets/Data/ folder.

Example

```
bVar = false
fVar = 0.0
iVar = 0
sVar = "init"

function Init()
   --Create a folder in Assets/Data/ path
   CreateFolder("Lev1")

   --Create and open file to write data
   OpenFileForWriting("Lev1/level1.bin")
   WriteBoolVariableToFile(true)
   WriteFloatVariableToFile(2.0)
   WriteIntVariableToFile(3)
   WriteStringVariableToFile("level1")
   CloseFile("Lev1/level1.bin")

   --Open File to load data
   OpenFileForReading("Lev1/level1.bin")
   bVar = ReadBoolVariableFromFile()
   fVar = ReadFloatVariableFromFile()
   iVar = ReadIntVariableFromFile()
   sVar = ReadStringVariableFromFile()
   CloseFile("Lev1/level1.bin")
end
```

First, using the `CreateFolder` function, we create a folder called `"Lev1"` in the Assets/Data/ path. Then, using the `OpenFileForWriting` function, we open the `level1.bin` file located in the Assets/Data/`Lev1`/ path for writing. After writing information to the file, we close it using the `CloseFile` function.
Then, using the `OpenFileForReading` function, we open the `level1.bin` file located in the path Assets/Data/`Lev1`/ for reading and read its information in the same order as we wrote. Finally, we close the file using the `CloseFile` function.

4.223. PauseAll3DSounds

Definition

`PauseAll3DSounds([optional] string exception_1, [optional] string exception_2,..., [optional] string exception_n)`

Description

This function pauses all 3D sounds that are being played except for the 3D sounds sent to the function.

Parameters

[optional] string exception_1, [optional] string exception_2,..., [optional] string exception_n

Specifies the name of the 3D sounds that should not be paused by this function. If no name is passed to `PauseAll3DSounds` function, all 3D sounds that are being played will be paused.

Example

```
function OnTriggerEnter(otherActorName)
    if otherActorName == nil then
          PauseAll3DSounds("sound3D_2", "sound3D_3")
    end
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)

end
```

Assume that the above script is attached to a trigger named trigger1. Whenever the main game character enters "trigger1", all the 3D sounds that are playing except the 3D sounds `"sound3D_2"` and `"sound3D_3"` will be paused.

4.224. PauseAllAmbientSounds

Definition

`PauseAllAmbientSounds([optional] string exception_1, [optional] string exception_2,..., [optional] string exception_n)`

Description

This function pauses all ambient sounds that are being played except for the ambient sounds sent to the function.

Parameters

`[optional] string exception_1, [optional] string exception_2,..., [optional] string exception_n`

Specifies the name of the ambient sounds that should not be paused by this function. If no name is passed to `PauseAllAmbientSounds` function, all ambient sounds that are being played will be paused.

Example

```
function OnTriggerEnter(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
        PauseAllAmbientSounds("ambient2", "ambient3")
    end
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)

end
```

Assume that the above script is attached to a trigger named trigger1. Whenever the main game character enters "trigger1", all the ambient sounds that are playing except the ambient sounds `"ambient2"` and `"ambient3"` will be paused.

4.225. PauseAllAnimationsOfPrefabInstances

Definition

`PauseAllAnimationsOfPrefabInstances([optional] string exception_1, [optional] string exception_2,..., [optional] string exception_n)`

Description

This function pauses animations of all prefab instances except for the animations of prefab instances sent to the function.

Parameters

`[optional] string exception_1, [optional] string exception_2,..., [optional] string exception_n`

Specifies the name of prefab instances whose animation should not be paused. If no name is passed to `PauseAllAnimationsOfPrefabInstances` function, animations of all prefab instances will be paused.

Example

```
function OnTriggerEnter(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          PauseAllAnimationsOfPrefabInstances("1_animation_test_plane",
"2_animation_test_boy")
    end
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)

end
```

Assume that the above script is attached to a trigger named trigger1. Whenever the main game character enters "trigger1", animations of all prefab instances except the animations of prefab instances `"1_animation_test_plane"` and `"2_animation_test_boy"` will be paused.

4.226. PauseAllResourceSounds

Definition

`PauseAllResourceSounds([optional] string exception_1, [optional] string exception_2,..., [optional] string exception_n)`

Description

This function pauses all resource sounds that are being played except for the resource sounds sent to the function.

Parameters

[optional] string exception_1, [optional] string exception_2,..., [optional] string exception_n

Specifies the name of the resource sounds that should not be paused by this function. If no name is passed to `PauseAllResourceSounds` function, all resource sounds that are being played will be paused.

Example

```
function OnTriggerEnter(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          LoadResource("Sounds", "fire.ogg")
          LoadResource("Sounds", "river.ogg")
          LoadResource("Sounds", "ambient.ogg")

          PlayResourceSoundLoop("Sounds_fire.ogg")
          PlayResourceSoundLoop("Sounds_river.ogg")
          PlayResourceSoundLoop("Sounds_ambient.ogg")
    end
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          PauseAllResourceSounds("Sounds_ambient.ogg")
    end
end
```

Assume that the above script is attached to a trigger named trigger1. Whenever the main character enters "trigger1", we load and play `"fire.ogg"`, `"river.ogg"` and `"ambient.ogg"` resource sounds --In order for `LoadResource` function to load the resources, you must first add all resources through the *Add Resource to Current Project* dialog (File > Project > Add/Remove Resource to/from Current Project).
Whenever the main character exits "trigger1", all resource sounds that are playing except the resource sound `"ambient.ogg"` will be paused.

4.227. PauseAllSounds

Definition

`PauseAllSounds([optional] string exception_1, [optional] string exception_2,..., [optional] string exception_n)`

Description

This function pauses all ambient, 3D and resource sounds that are being played except for the ambient, 3D and resource sounds sent to the function.

Parameters

[optional] string exception_1, [optional] string exception_2,..., [optional] string exception_n

Specifies the name of the ambient, 3D and resource sounds that should not be paused by this function. If no name is passed to `PauseAllSounds` function, all ambient, 3D and resource sounds that are being played will be paused.

Example

```
function OnTriggerEnter(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          LoadResource("Sounds", "fire.ogg")
          LoadResource("Sounds", "river.ogg")
          LoadResource("Sounds", "ambient.ogg")

          PlayResourceSoundLoop("Sounds_fire.ogg")
          PlayResourceSoundLoop("Sounds_river.ogg")
          PlayResourceSoundLoop("Sounds_ambient.ogg")
    end
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          PauseAllSounds("ambient2", "river2", "Sounds_ambient.ogg")
    end
end
```

Assume that the above script is attached to a trigger named trigger1. Also, `"ambient2"` and `"river2"` in the example above are ambient and 3D sounds, respectively.
Whenever the main character enters "trigger1", we load and play `"fire.ogg"`, `"river.ogg"` and `"ambient.ogg"` resource sounds --In order for `LoadResource` function to load the resources, you must first add all resources through the *Add Resource to Current Project* dialog (File > Project > Add/Remove Resource to/from Current Project).

Whenever the main character exits "trigger1", all ambient, 3D and resource sounds that are playing except the the ambient sound `"ambient2"`, 3D sound `"river2"` and resource sound `"ambient.ogg"` will be paused.

4.228. PauseAllUpdateEvents

Definition

`PauseAllUpdateEvents([optional] string exception_1, [optional] string exception_2,..., [optional] string exception_n)`

Description

This function pauses the script's `Update()` event of all game objects except the script's `Update()` event of objects passed to the function.

Parameters

`[optional] string exception_1, [optional] string exception_2,..., [optional] string exception_n`

Specifies the name of the objects whose script's `Update()` event should not be paused by this function. If no name is passed to the function, `Update()` events of all game object scripts will be paused.

Example

```
function OnTriggerEnter(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          PauseAllUpdateEvents("water1", "sound1")
    end
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)

end
```

Assume that the above script is attached to a trigger named trigger1. Also assume that `"water1"` and `"sound1"` in the example above are the name of water and sound objects in the VScene, respectively.

Whenever the main character enters "trigger1", script's `Update()` event of all game objects except script's `Update()` event of `"water1"` and `"sound1"` objects will be paused.

4.229. PauseAnimationOfAllWaters

Definition

`PauseAnimationOfAllWaters([optional] string exception_1, [optional] string exception_2,..., [optional] string exception_n)`

Description

This function pauses animation of all water objects except for the animation of water objects sent to the function.

Parameters

`[optional] string exception_1, [optional] string exception_2,...,`
`[optional] string exception_n`

Specifies the name of waters whose animation should not be paused. If no name is passed to `PauseAnimationOfAllWaters` function, animation of all waters will be paused.

Example

```
function OnTriggerEnter(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          PauseAnimationOfAllWaters("water2", "water3")
    end
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)

end
```

Assume that the above script is attached to a trigger named trigger1. Also assume that `"water2"` and `"water3"` in the example above are the name of water objects in the VScene.
Whenever the main character enters "trigger1", animation of all waters except the animation of waters `"water2"` and `"water3"` will be paused.

4.230. PauseGame

Definition

```
PauseGame()
```

Description

This function pauses the game.

Example

```
function OnTriggerEnter(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          PauseGame()
    end
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)

end
```

Assume that the above script is attached to a trigger named trigger1. Whenever the main character enters "trigger1", game pauses.

4.231. PauseMainCharacterAnimations

Definition

```
PauseMainCharacterAnimations()
```

Description

This function pauses all animations of the main character.

Example

```
function OnTriggerEnter(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          PauseMainCharacterAnimations()
    end
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)

end
```

Assume that the above script is attached to a trigger named trigger1. Whenever the main character enters "trigger1", all animations of the main character are paused.

4.232. PausePhysics

Definition

```
PausePhysics()
```

Description

This function pauses the physics.

Example

```
function OnTriggerEnter(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          PausePhysics()
    end
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)

end
```

Assume that the above script is attached to a trigger named trigger1. Whenever the main character enters "trigger1", physics pauses.

4.233. PausePrefabInstanceAnimations

Definition

```
PausePrefabInstanceAnimations(string prefabInstanceName)
```

Description

This function pauses all animations of the prefab instance `prefabInstanceName`. To view the name of prefab instances, open the VScene and click on the desired Prefab Instance in the "Prefabs and GUIs" section and press the Edit button. You can also access the names of prefab instances from the Script Utility section of the script editor (Tools > Script Editor > Tools > Script Utility). In the dialog that appears, you can view and copy the name of the prefab instance.

Parameters

prefabInstanceName

Specifies the name of the prefab instance. You can also use the name "this" for this parameter. In this case, "this" refers to the prefab instance that this script is attached to.

Example 1

```
function OnTriggerEnter(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
        PausePrefabInstanceAnimations("1_animation_test_plane")
    end
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)

end
```

Assume that the above script is attached to a trigger named trigger1. Whenever the main character enters "trigger1", all animations of prefab instance `"1_animation_test_plane"` will be paused.

Example 2

```
--Name of script is PausePrefabInstanceAnimations2.lua

function Init()
    PausePrefabInstanceAnimations("this")
end

function Update()

end
```

If, in the Prefab Editor, you attach `PausePrefabInstanceAnimations2.lua` script to a Prefab, then `"this"` parameter in the `PausePrefabInstanceAnimations` function will point to instances of that Prefab in current VScene. For example, if you have an Instance named *instance1_a* from a Prefab named *a* to which this script is attached, `"this"` in `PausePrefabInstanceAnimations` function refers to the name *instance1_a*.
This function pause all animations of current prefab instance.

4.234. PauseResourceSound

Definition

`PauseResourceSound(string resourceDirectoryName_resourceFileName.ogg)`

Description

This function pauses resource sound **`resourceDirectoryName_resourceFileName.ogg`** that is being played. You can go to the *Project Resources* section through the Script Utility dialog (Tools > Script Editor > Tools > Script Utility), select the desired resource sound and hit "Copy Folder_File Name" button to copy the full name of the resource.

Parameters

`resourceDirectoryName_resourceFileName.ogg`
Specifies the full name of the resource sound.

Example

```
function OnTriggerEnter(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          LoadResource("Sounds", "fire.ogg")
          PlayResourceSoundLoop("Sounds_fire.ogg")
    end
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          PauseResourceSound("Sounds_fire.ogg")
    end
end
```

Assume that the above script is attached to a trigger named trigger1. Whenever the main character enters "trigger1", we load and play `"fire.ogg"` resource sound --In order for `LoadResource` function to load the resource sound, you must first add `"fire.ogg"` sound resource through the *Add Resource to Current Project* dialog (File > Project > Add/Remove Resource to/from Current Project).
Whenever the main character exits "trigger1", the resource sound `"fire.ogg"` will be paused.

4.235. PauseSound

Definition

`PauseSound(string soundObjectName1, string soundObjectName2, ..., string soundObjectNameN)`

Description

This function pauses all ambient and 3D sounds `soundObjectName1, soundObjectName2, ..., soundObjectNameN` that are playing.

Parameters

soundObjectName1, soundObjectName2, ..., soundObjectNameN

Specify the name of the ambient and 3D sounds that should be paused by this function. You can also use the name "this" for ***soundObjectName[N]***. In this case, "this" refers to the ambient or 3D sound that this script is attached to.

Example

```
function Init()
    PauseSound("this", "ambient2", "fire1")
end

function Update()

end
```

Assume that the above script is attached to an ambient sound named "ambient1". Also, `"ambient2"` and `"fire1"` in the example above are ambient and 3D sound names, respectively. In our example, `PauseSound` function pauses the current sound (which has a name equivalent to "ambient1"), the ambient sound `"ambient2"`, and the 3D sound `"fire1"`.

4.236. PauseUpdateEventOf3DSound

Definition

`PauseUpdateEventOf3DSound(string 3DSoundName)`

Description

This function pauses the script's `Update()` event of 3D sound **`3DSoundName`**.

Parameters

3DSoundName

Specifies the name of the 3D sound. You can also use the name "this" for this parameter. In this case, "this" refers to the 3D sound that this script is attached to.

Example 1

```
function OnTriggerEnter(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          PauseUpdateEventOf3DSound("river1")
    end
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)

end
```

Assume that the above script is attached to a trigger named trigger1. Whenever the main character enters "trigger1", script's `Update()` event of 3D sound `"river1"` will be paused.

Example 2

```
--Name of script is pauseupdateeventof3dsound2.lua

function Init()
    PauseUpdateEventOf3DSound("this")
end

function Update()

end
```

Assume that the above script named `pauseupdateeventof3dsound2.lua` is attached to a 3D sound object named "sound1". In this case, string `"this"` in the `PauseUpdateEventOf3DSound` function will be equal to "sound1". In our example, the function `PauseUpdateEventOf3DSound` pauses the script's `Update()` event of current 3D sound, which is "sound1".

4.237. PauseUpdateEventOfAll3DSounds

Definition

`PauseUpdateEventOfAll3DSounds([optional] string exception_1, [optional] string exception_2,..., [optional] string exception_n)`

Description

This function pauses the script's `Update()` event of all 3D sounds except the script's `Update()` event of 3D sounds passed to the function.

Parameters

`[optional] string exception_1, [optional] string exception_2,..., [optional] string exception_n`

Specifies the name of the 3D sounds whose script's `Update()` event should not be paused by this function. If no name is passed to the function, `Update()` events of all 3D sound scripts will be paused.

Example

```
function OnTriggerEnter(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          PauseUpdateEventOfAll3DSounds("river2", "river3")
    end
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)

end
```

Assume that the above script is attached to a trigger named trigger1. Also assume that `"river2"` and `"river3"` in the example above are the name of 3D sound objects.
Whenever the main character enters "trigger1", script's `Update()` event of all 3D sounds except script's `Update()` event of `"river2"` and `"river3"` 3D sounds will be paused.

4.238. PauseUpdateEventOfAllAmbientSounds

Definition

```
PauseUpdateEventOfAllAmbientSounds([optional] string exception_1,
[optional] string exception_2,..., [optional] string exception_n)
```

Description

This function pauses the script's `Update()` event of all ambient sounds except the script's `Update()` event of ambient sounds passed to the function.

Parameters

[optional] string exception_1, [optional] string exception_2,..., [optional] string exception_n

Specifies the name of the ambient sounds whose script's `Update()` event should not be paused by this function. If no name is passed to the function, `Update()` events of all ambient sound scripts will be paused.

Example

```
function OnTriggerEnter(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
        PauseUpdateEventOfAllAmbientSounds("ambient2", "ambient3")
    end
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)

end
```

Assume that the above script is attached to a trigger named trigger1. Also assume that `"ambient2"` and `"ambient3"` in the example above are the name of ambient sound objects. Whenever the main character enters "trigger1", script's `Update()` event of all ambient sounds except script's `Update()` event of `"ambient2"` and `"ambient3"` ambient sounds will be paused.

4.239. PauseUpdateEventOfAllEngineCameras

Definition

`PauseUpdateEventOfAllEngineCameras([optional] string exception_1, [optional] string exception_2,..., [optional] string exception_n)`

Description

This function pauses the script's `Update()` event of all engine cameras except the script's `Update()` event of engine cameras passed to the function.

Parameters

[optional] string exception_1, [optional] string exception_2,..., [optional] string exception_n

Specifies the name of the engine cameras whose script's `Update()` event should not be paused by this function. If no name is passed to the function, `Update()` events of all engine camera scripts will be paused.

Example

```
function OnTriggerEnter(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          PauseUpdateEventOfAllEngineCameras("camera2", "camera3")
    end
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)

end
```

Assume that the above script is attached to a trigger named trigger1. Also assume that `"camera2"` and `"camera3"` in the example above are the name of engine camera objects. Whenever the main character enters "trigger1", script's `Update()` event of all engine cameras except script's `Update()` event of `"camera2"` and `"camera3"` engine cameras will be paused.

4.240. PauseUpdateEventOfAllLights

Definition

`PauseUpdateEventOfAllLights([optional] string exception_1, [optional] string exception_2,..., [optional] string exception_n)`

Description

This function pauses the script's `Update()` event of all lights except the script's `Update()` event of lights passed to the function.

Parameters

`[optional] string exception_1, [optional] string exception_2,..., [optional] string exception_n`

Specifies the name of the lights whose script's `Update()` event should not be paused by this function. If no name is passed to the function, `Update()` events of all light scripts will be paused.

Example

```
function OnTriggerEnter(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          PauseUpdateEventOfAllLights("light2", "light3")
    end
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)

end
```

Assume that the above script is attached to a trigger named trigger1. Also assume that `"light2"` and `"light3"` in the example above are the name of light objects.
Whenever the main character enters "trigger1", script's `Update()` event of all lights except script's `Update()` event of `"light2"` and `"light3"` lights will be paused.

4.241. PauseUpdateEventOfAllPrefabInstances

Definition

`PauseUpdateEventOfAllPrefabInstances([optional] string exception_1, [optional] string exception_2,..., [optional] string exception_n)`

Description

This function pauses the script's `Update()` event of all prefab instances except the script's `Update()` event of prefab instances passed to the function.

Parameters

`[optional] string exception_1, [optional] string exception_2,..., [optional] string exception_n`

Specifies the name of the prefab instances whose script's `Update()` event should not be paused by this function. If no name is passed to the function, `Update()` events of all prefab instance scripts will be paused.

Example

```
function OnTriggerEnter(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          PauseUpdateEventOfAllPrefabInstances("1_animation_test_boy",
"1_animation_test_plane")
    end
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)

end
```

Assume that the above script is attached to a trigger named trigger1. Also assume that `"1_animation_test_boy"` and `"1_animation_test_plane"` in the example above are the name of prefab instances.
Whenever the main character enters "trigger1", script's `Update()` event of all prefab instances except script's `Update()` event of `"1_animation_test_boy"` and `"1_animation_test_plane"` prefab instances will be paused.

4.242. PauseUpdateEventOfAllWaters

Definition

`PauseUpdateEventOfAllWaters([optional] string exception_1, [optional] string exception_2,..., [optional] string exception_n)`

Description

This function pauses the script's `Update()` event of all waters except the script's `Update()` event of waters passed to the function.

Parameters

[optional] string exception_1, [optional] string exception_2,..., [optional] string exception_n

Specifies the name of the waters whose script's `Update()` event should not be paused by this function. If no name is passed to the function, `Update()` events of all water scripts will be paused.

Example

```
function OnTriggerEnter(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
        PauseUpdateEventOfAllWaters("water2", "water3")
    end
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)

end
```

Assume that the above script is attached to a trigger named trigger1. Also assume that `"water2"` and `"water3"` in the example above are the name of water objects.
Whenever the main character enters "trigger1", script's `Update()` event of all waters except script's `Update()` event of `"water2"` and `"water3"` waters will be paused.

4.243. PauseUpdateEventOfAmbientSound

Definition

`PauseUpdateEventOfAmbientSound(string ambientSoundName)`

Description

This function pauses the script's `Update()` event of ambient sound `ambientSoundName`.

Parameters

`ambientSoundName`

Specifies the name of the ambient sound. You can also use the name "this" for this parameter. In this case, "this" refers to the ambient sound name that this script is attached to.

Example 1

```
function OnTriggerEnter(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          PauseUpdateEventOfAmbientSound("ambient1")
    end
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)

end
```

Assume that the above script is attached to a trigger named trigger1. Whenever the main character enters "trigger1", script's `Update()` event of ambient sound `"ambient1"` will be paused.

Example 2

```
--Name of script is PauseUpdateEventOfAmbientSound2.lua

function Init()
    PauseUpdateEventOfAmbientSound("this")
end

function Update()

end
```

Assume that the above script named `PauseUpdateEventOfAmbientSound2.lua` is attached to an ambient sound object named "sound1". In this case, string `"this"` in the `PauseUpdateEventOfAmbientSound` function will be equal to "sound1". In our example, the function `PauseUpdateEventOfAmbientSound` pauses the script's `Update()` event of current ambient sound, which is "sound1".

4.244. PauseUpdateEventOfEngineCamera

Definition

`PauseUpdateEventOfEngineCamera(string engineCameraName)`

Description

This function pauses the script's `Update()` event of engine camera `engineCameraName`.

Parameters

engineCameraName

Specifies the name of the engine camera. You can also use the name "this" for this parameter. In this case, "this" refers to the engine camera name that this script is attached to.

Example 1

```
function OnTriggerEnter(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          PauseUpdateEventOfEngineCamera("camera1")
    end
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)

end
```

Assume that the above script is attached to a trigger named trigger1. Whenever the main character enters "trigger1", script's `Update()` event of engine camera `"camera1"` will be paused.

Example 2

```
--Name of script is PauseUpdateEventOfEngineCamera2.lua

function Init()
    PauseUpdateEventOfEngineCamera("this")
end

function Update()

end
```

Assume that the above script named `PauseUpdateEventOfEngineCamera2.lua` is attached to an engine camera object named "camera1". In this case, string `"this"` in the `PauseUpdateEventOfEngineCamera` function will be equal to "camera1". In our example, the function `PauseUpdateEventOfEngineCamera` pauses the script's `Update()` event of current engine camera, which is "camera1".

4.245. PauseUpdateEventOfLight

Definition

`PauseUpdateEventOfLight(string lightName)`

Description

This function pauses the script's `Update()` event of light `lightName`.

Parameters

`lightName`

Specifies the name of the light. You can also use the name "this" for this parameter. In this case, "this" refers to the light name that this script is attached to.

Example 1

```
function OnTriggerEnter(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          PauseUpdateEventOfLight("light1")
    end
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)

end
```

Assume that the above script is attached to a trigger named trigger1. Whenever the main character enters "trigger1", script's `Update()` event of light `"light1"` will be paused.

Example 2

```
--Name of script is PauseUpdateEventOfLight2.lua

function Init()
    PauseUpdateEventOfLight("this")
end

function Update()

end
```

Assume that the above script named `PauseUpdateEventOfLight2.lua` is attached to a light object named "light1". In this case, string `"this"` in the `PauseUpdateEventOfLight` function will be equal to "light1". In our example, the function `PauseUpdateEventOfLight` pauses the script's `Update()` event of current light, which is "light1".

4.246. PauseUpdateEventOfMainCharacter

Definition

`PauseUpdateEventOfMainCharacter()`

Description

This function pauses the script's `Update()` event of main character.

Example

```
function OnTriggerEnter(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          PauseUpdateEventOfMainCharacter()
    end
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)

end
```

Assume that the above script is attached to a trigger named trigger1. Whenever the main character enters "trigger1", script's `Update()` event of main character will be paused.

4.247. PauseUpdateEventOfPrefabInstance

Definition

`PauseUpdateEventOfPrefabInstance(string prefabInstanceName)`

Description

This function pauses the script's `Update()` event of prefab instance `prefabInstanceName`.

Parameters

prefabInstanceName

Specifies the name of the prefab instance. You can also use the name "this" for this parameter. In this case, "this" refers to the prefab instance name that this script is attached to.

Example 1

```
function OnTriggerEnter(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          PauseUpdateEventOfPrefabInstance("1_animation_test_plane")
    end
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)

end
```

Assume that the above script is attached to a trigger named trigger1. Whenever the main character enters "trigger1", script's `Update()` event of prefab instance `"1_animation_test_plane"` will be paused.

Example 2

```
--Name of script is PauseUpdateEventOfPrefabInstance2.lua

function Init()
    PauseUpdateEventOfPrefabInstance("this")
end

function Update()

end
```

If, in the Prefab Editor, you attach `PauseUpdateEventOfPrefabInstance2.lua` script to a Prefab, then `"this"` parameter in the `PauseUpdateEventOfPrefabInstance` function will point to instances of that Prefab in current VScene. For example, if you have an Instance named *instance1_a* from a Prefab named *a* to which this script is attached, `"this"` in `PauseUpdateEventOfPrefabInstance` function refers to the name *instance1_a*.

In this example, `PauseUpdateEventOfPrefabInstance` will pause the script's `Update()` event of current prefab instance (for example, *instance1_a*).

4.248. PauseUpdateEventOfSky

Definition

`PauseUpdateEventOfSky()`

Description

This function pauses the script's `Update()` event of sky object.

Example

```
function OnTriggerEnter(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          PauseUpdateEventOfSky()
    end
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)

end
```

Assume that the above script is attached to a trigger named trigger1. Whenever the main character enters "trigger1", script's `Update()` event of sky object will be paused.

4.249. PauseUpdateEventOfTerrain

Definition

`PauseUpdateEventOfTerrain()`

Description

This function pauses the script's `Update()` event of terrain object.

Example

```
function OnTriggerEnter(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          PauseUpdateEventOfTerrain()
    end
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)

end
```

Assume that the above script is attached to a trigger named trigger1. Whenever the main character enters "trigger1", script's `Update()` event of terrain object will be paused.

4.250. PauseUpdateEventOfVSceneScript

Definition

`PauseUpdateEventOfVSceneScript()`

Description

This function pauses the script's `Update()` event of VScene Script object.

Example

```
function OnTriggerEnter(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          PauseUpdateEventOfVSceneScript()
    end
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)

end
```

Assume that the above script is attached to a trigger named trigger1. Whenever the main character enters "trigger1", script's `Update()` event of VScene Script object will be paused.

4.251. PauseUpdateEventOfWater

Definition

`PauseUpdateEventOfWater(string waterName)`

Description

This function pauses the script's `Update()` event of water `waterName`.

Parameters

`waterName`

Specifies the name of the water. You can also use the name "this" for this parameter. In this case, "this" refers to the water name that this script is attached to.

Example 1

```
function OnTriggerEnter(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          PauseUpdateEventOfWater("water1")
    end
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)

end
```

Assume that the above script is attached to a trigger named trigger1. Whenever the main character enters "trigger1", script's `Update()` event of water `"water1"` will be paused.

Example 2

```
--Name of script is PauseUpdateEventOfWater2.lua

function Init()
    PauseUpdateEventOfWater("this")
end

function Update()

end
```

Assume that the above script named `PauseUpdateEventOfWater2.lua` is attached to a water object named "water1". In this case, string `"this"` in the `PauseUpdateEventOfWater` function will be equal to "water1". In our example, the function `PauseUpdateEventOfWater` pauses the script's `Update()` event of current water, which is "water1".

4.252. PauseWaterAnimation

Definition

`PauseWaterAnimation(string waterObjectName)`

Description

This function pauses animation of water **`waterObjectName`**.

Parameters

`waterObjectName`

Specifies the name of the water. You can also use the name "this" for this parameter. In this case, "this" refers to the water name that this script is attached to.

Example 1

```
function OnTriggerEnter(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          PauseWaterAnimation("water1")
    end
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)

end
```

Assume that the above script is attached to a trigger named trigger1. Whenever the main character enters "trigger1", animation of water `"water1"` will be paused.

Example 2

```
--Name of script is PauseWaterAnimation2.lua

function Init()
    PauseWaterAnimation("this")
end

function Update()

end
```

Assume that the above script named `PauseWaterAnimation2.lua` is attached to a water object named "water1". In this case, string `"this"` in the `PauseWaterAnimation` function will be equal to "water1". In our example, the function `PauseWaterAnimation` pauses animation of current water, which is "water1".

4.253. PlayAll3DSounds

Definition

`PlayAll3DSounds([optional] string exception_1, [optional] string exception_2,..., [optional] string exception_n)`

Description

This function plays all 3D sounds except for the 3D sounds sent to the function. If the loop state of each 3D sound is true, the sound will be played continuously, otherwise it will be played only once.

Parameters

`[optional] string exception_1, [optional] string exception_2,..., [optional] string exception_n`

Specifies the name of the 3D sounds that should not be played by this function. If no name is passed to `PlayAll3DSounds` function, all 3D sounds will be played.

Example

```
function OnTriggerEnter(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          PlayAll3DSounds("sound3D_2", "sound3D_3")
    end
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)

end
```

Assume that the above script is attached to a trigger named trigger1. Whenever the main game character enters "trigger1", all the 3D sounds except the 3D sounds `"sound3D_2"` and `"sound3D_3"` will be played.

4.254. PlayAll3DSoundsLoop

Definition

`PlayAll3DSoundsLoop([optional] string exception_1, [optional] string exception_2,..., [optional] string exception_n)`

Description

This function plays all 3D sounds continuously except for the 3D sounds sent to the function.

Parameters

`[optional] string exception_1, [optional] string exception_2,..., [optional] string exception_n`

Specifies the name of the 3D sounds that should not be played by this function. If no name is passed to `PlayAll3DSoundsLoop` function, all 3D sounds will be played continuously.

Example

```
function OnTriggerEnter(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          PlayAll3DSoundsLoop("sound3D_2", "sound3D_3")
    end
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)

end
```

Assume that the above script is attached to a trigger named trigger1. Whenever the main game character enters "trigger1", all the 3D sounds except the 3D sounds `"sound3D_2"` and `"sound3D_3"` will be played continuously.

4.255. PlayAll3DSoundsOnce

Definition

`PlayAll3DSoundsOnce([optional] string exception_1, [optional] string exception_2,..., [optional] string exception_n)`

Description

This function plays all 3D sounds once except for the 3D sounds sent to the function.

Parameters

`[optional] string exception_1, [optional] string exception_2,..., [optional] string exception_n`

Specifies the name of the 3D sounds that should not be played by this function. If no name is passed to `PlayAll3DSoundsOnce` function, all 3D sounds will be played once.

Example

```
function OnTriggerEnter(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          PlayAll3DSoundsOnce("sound3D_2", "sound3D_3")
    end
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)

end
```

Assume that the above script is attached to a trigger named trigger1. Whenever the main game character enters "trigger1", all the 3D sounds except the 3D sounds `"sound3D_2"` and `"sound3D_3"` will be played once.

4.256. PlayAllAmbientSounds

Definition

`PlayAllAmbientSounds([optional] string exception_1, [optional] string exception_2,..., [optional] string exception_n)`

Description

This function plays all ambient sounds except for the ambient sounds sent to the function. If the loop state of each ambient sound is true, the sound will be played continuously, otherwise it will be played only once.

Parameters

`[optional] string exception_1, [optional] string exception_2,..., [optional] string exception_n`

Specifies the name of the ambient sounds that should not be played by this function. If no name is passed to `PlayAllAmbientSounds` function, all ambient sounds will be played.

Example

```
function OnTriggerEnter(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          PlayAllAmbientSounds("ambient2", "ambient3")
    end
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)

end
```

Assume that the above script is attached to a trigger named trigger1. Whenever the main game character enters "trigger1", all the ambient sounds except the ambient sounds `"ambient2"` and `"ambient3"` will be played.

4.257. PlayAllAmbientSoundsLoop

Definition

`PlayAllAmbientSoundsLoop([optional] string exception_1, [optional] string exception_2,..., [optional] string exception_n)`

Description

This function plays all ambient sounds continuously except for the ambient sounds sent to the function.

Parameters

`[optional] string exception_1, [optional] string exception_2,..., [optional] string exception_n`

Specifies the name of the ambient sounds that should not be played by this function. If no name is passed to `PlayAllAmbientSoundsLoop` function, all ambient sounds will be played continuously.

Example

```
function OnTriggerEnter(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          PlayAllAmbientSoundsLoop("ambient2", "ambient3")
    end
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)

end
```

Assume that the above script is attached to a trigger named trigger1. Whenever the main game character enters "trigger1", all the ambient sounds except the ambient sounds `"ambient2"` and `"ambient3"` will be played continuously.

4.258. PlayAllAmbientSoundsOnce

Definition

`PlayAllAmbientSoundsOnce([optional] string exception_1, [optional] string exception_2,..., [optional] string exception_n)`

Description

This function plays all ambient sounds once except for the ambient sounds sent to the function.

Parameters

`[optional] string exception_1, [optional] string exception_2,..., [optional] string exception_n`

Specifies the name of the ambient sounds that should not be played by this function. If no name is passed to `PlayAllAmbientSoundsOnce` function, all ambient sounds will be played once.

Example

```
function OnTriggerEnter(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          PlayAllAmbientSoundsOnce("ambient2", "ambient3")
    end
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)

end
```

Assume that the above script is attached to a trigger named trigger1. Whenever the main game character enters "trigger1", all the ambient sounds except the ambient sounds `"ambient2"` and `"ambient3"` will be played once.

4.259. PlayAllPaused3DSounds

Definition

`PlayAllPaused3DSounds([optional] string exception_1, [optional] string exception_2,..., [optional] string exception_n)`

Description

This function plays all *paused* 3D sounds except for the paused 3D sounds sent to the function. If the loop state of each 3D sound is true, the paused 3D sound will be played continuously, otherwise it will be played only once.

Parameters

[optional] string exception_1, [optional] string exception_2,..., [optional] string exception_n

Specifies the name of the paused 3D sounds that should not be played by this function. If no name is passed to `PlayAllPaused3DSounds` function, all paused 3D sounds will be played.

Example

```
function OnTriggerEnter(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          PauseAll3DSounds()
    end
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)
    if otherActorName == nil then
          PlayAllPaused3DSounds("river2", "river3")
    end
end
```

Assume that the above script is attached to a trigger named trigger1. Also assume that `"river2"` and `"river3"` are 3D sound objects.
Whenever the main character enters "trigger1", all the 3D sounds that are playing will be paused.
Whenever the main character exits "trigger1", all the paused 3D sounds except the 3D sounds `"river2"` and `"river3"` will be played.

4.260. PlayAllPaused3DSoundsLoop

Definition

`PlayAllPaused3DSoundsLoop([optional] string exception_1, [optional] string exception_2,..., [optional] string exception_n)`

Description

This function plays all *paused* 3D sounds continuously except for the paused 3D sounds sent to the function.

Parameters

`[optional] string exception_1, [optional] string exception_2,...,`
`[optional] string exception_n`

Specifies the name of the paused 3D sounds that should not be played by this function. If no name is passed to `PlayAllPaused3DSoundsLoop` function, all paused 3D sounds will be played continuously.

Example

```
function OnTriggerEnter(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          PauseAll3DSounds()
    end
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)
    if otherActorName == nil then
          PlayAllPaused3DSoundsLoop("river2", "river3")
    end
end
```

Assume that the above script is attached to a trigger named trigger1. Also assume that `"river2"` and `"river3"` are 3D sound objects.
Whenever the main character enters "trigger1", all the 3D sounds that are playing will be paused.
Whenever the main character exits "trigger1", all the paused 3D sounds except the 3D sounds `"river2"` and `"river3"` will be played continuously.

4.261. PlayAllPaused3DSoundsOnce

Definition

`PlayAllPaused3DSoundsOnce([optional] string exception_1, [optional] string exception_2,..., [optional] string exception_n)`

Description

This function plays all *paused* 3D sounds once except for the paused 3D sounds sent to the function.

Parameters

`[optional] string exception_1, [optional] string exception_2,..., [optional] string exception_n`

Specifies the name of the paused 3D sounds that should not be played by this function. If no name is passed to `PlayAllPaused3DSoundsOnce` function, all paused 3D sounds will be played once.

Example

```
function OnTriggerEnter(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          PauseAll3DSounds()
    end
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)
    if otherActorName == nil then
          PlayAllPaused3DSoundsOnce("river2", "river3")
    end
end
```

Assume that the above script is attached to a trigger named trigger1. Also assume that `"river2"` and `"river3"` are 3D sound objects.
Whenever the main character enters "trigger1", all the 3D sounds that are playing will be paused.
Whenever the main character exits "trigger1", all the paused 3D sounds except the 3D sounds `"river2"` and `"river3"` will be played once.

4.262. PlayAllPausedAmbientSounds

Definition

```
PlayAllPausedAmbientSounds([optional] string exception_1, [optional]
string exception_2,..., [optional] string exception_n)
```

Description

This function plays all *paused* ambient sounds except for the paused ambient sounds sent to the function. If the loop state of each ambient sound is true, the paused ambient sound will be played continuously, otherwise it will be played only once.

Parameters

`[optional] string exception_1, [optional] string exception_2,..., [optional] string exception_n`

Specifies the name of the paused ambient sounds that should not be played by this function. If no name is passed to `PlayAllPausedAmbientSounds` function, all paused ambient sounds will be played.

Example

```
function OnTriggerEnter(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          PauseAllAmbientSounds()
    end
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)
    if otherActorName == nil then
          PlayAllPausedAmbientSounds("ambient2", "ambient3")
    end
end
```

Assume that the above script is attached to a trigger named trigger1. Also assume that `"ambient2"` and `"ambient3"` are ambient sound objects.
Whenever the main character enters "trigger1", all the ambient sounds that are playing will be paused. Whenever the main character exits "trigger1", all the paused ambient sounds except the ambient sounds `"ambient2"` and `"ambient3"` will be played.

4.263. PlayAllPausedAmbientSoundsLoop

Definition

```
PlayAllPausedAmbientSoundsLoop([optional] string exception_1, [optional]
string exception_2,..., [optional] string exception_n)
```

Description

This function plays all *paused* ambient sounds continuously except for the paused ambient sounds sent to the function.

Parameters

[optional] string exception_1, [optional] string exception_2,..., [optional] string exception_n

Specifies the name of the paused ambient sounds that should not be played by this function. If no name is passed to `PlayAllPausedAmbientSoundsLoop` function, all paused ambient sounds will be played continuously.

Example

```
function OnTriggerEnter(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          PauseAllAmbientSounds()
    end
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)
    if otherActorName == nil then
          PlayAllPausedAmbientSoundsLoop("ambient2", "ambient3")
    end
end
```

Assume that the above script is attached to a trigger named trigger1. Also assume that `"ambient2"` and `"ambient3"` are ambient sound objects.
Whenever the main character enters "trigger1", all the ambient sounds that are playing will be paused. Whenever the main character exits "trigger1", all the paused ambient sounds except the ambient sounds `"ambient2"` and `"ambient3"` will be played continuously.

4.264. PlayAllPausedAmbientSoundsOnce

Definition

`PlayAllPausedAmbientSoundsOnce([optional] string exception_1, [optional] string exception_2,..., [optional] string exception_n)`

Description

This function plays all *paused* ambient sounds once except for the paused ambient sounds sent to the function.

Parameters

`[optional] string exception_1, [optional] string exception_2,..., [optional] string exception_n`

Specifies the name of the paused ambient sounds that should not be played by this function. If no name is passed to `PlayAllPausedAmbientSoundsOnce` function, all paused ambient sounds will be played once.

Example

```
function OnTriggerEnter(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          PauseAllAmbientSounds()
    end
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)
    if otherActorName == nil then
          PlayAllPausedAmbientSoundsOnce("ambient2", "ambient3")
    end
end
```

Assume that the above script is attached to a trigger named trigger1. Also assume that `"ambient2"` and `"ambient3"` are ambient sound objects.
Whenever the main character enters "trigger1", all the ambient sounds that are playing will be paused. Whenever the main character exits "trigger1", all the paused ambient sounds except the ambient sounds `"ambient2"` and `"ambient3"` will be played once.

4.265. PlayAllPausedResourceSounds

Definition

`PlayAllPausedResourceSounds([optional] string exception_1, [optional] string exception_2,..., [optional] string exception_n)`

Description

This function plays all *paused* resource sounds except for the paused resource sounds sent to the function. If the loop state of each resource sound is true (For example, if it is played by the `PlayResourceSoundsLoop` function and then paused by the `PauseResourceSound` function), the paused resource sound will be played continuously, otherwise it will be played only once.

Parameters

`[optional] string exception_1, [optional] string exception_2,..., [optional] string exception_n`

Specifies the name of the paused resource sounds that should not be played by this function. If no name is passed to `PlayAllPausedResourceSounds` function, all paused resource sounds will be played.

Example

```
function OnTriggerEnter(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          LoadResource("Sounds", "fire.ogg")
          LoadResource("Sounds", "river.ogg")
          LoadResource("Sounds", "ambient.ogg")
          PlayAllResourceSoundsLoop()
    end
end

function OnTriggerStay(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          PauseAllResourceSounds()
    end
end

function OnTriggerExit(otherActorName)
    if otherActorName == nil then
          PlayAllPausedResourceSounds("Sounds_fire.ogg", "Sounds_river.ogg")
    end
end
```

Assume that the above script is attached to a trigger named trigger1.
Whenever the main character enters "trigger1", we load and play 3 resource sounds -- In order for `LoadResource` function to load the desired resource, you must first add it through the *Add Resource to Current Project* dialog (File > Project > Add/Remove Resource to/from Current Project). When the main character stays in the trigger, all the resource sounds that are playing will be paused. When the main character exits "trigger1", all the paused resource sounds except the resource sounds `"fire.ogg"` and `"river.ogg"` will be played.

4.266. PlayAllPausedResourceSoundsLoop

Definition

`PlayAllPausedResourceSoundsLoop([optional] string exception_1, [optional] string exception_2,..., [optional] string exception_n)`

Description

This function plays all *paused* resource sounds continuously except for the paused resource sounds sent to the function.

Parameters

[optional] string exception_1, [optional] string exception_2,..., [optional] string exception_n

Specifies the name of the paused resource sounds that should not be played by this function. If no name is passed to `PlayAllPausedResourceSoundsLoop` function, all paused resource sounds will be played continuously.

Example

```
function OnTriggerEnter(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          LoadResource("Sounds", "fire.ogg")
          LoadResource("Sounds", "river.ogg")
          LoadResource("Sounds", "ambient.ogg")
          PlayAllResourceSoundsLoop()
    end
end

function OnTriggerStay(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          PauseAllResourceSounds()
    end
end

function OnTriggerExit(otherActorName)
    if otherActorName == nil then
          PlayAllPausedResourceSoundsLoop("Sounds_fire.ogg", "Sounds_river.ogg")
    end
end
```

Assume that the above script is attached to a trigger named trigger1.
Whenever the main character enters "trigger1", we load and play 3 resource sounds -- In order for `LoadResource` function to load the desired resource, you must first add it through the *Add Resource to Current Project* dialog (File > Project > Add/Remove Resource to/from Current Project). When the main character stays in the trigger, all the resource sounds that are playing will be paused. When the main character exits "trigger1", all the paused resource sounds except the resource sounds `"fire.ogg"` and `"river.ogg"` will be played continuously.

4.267. PlayAllPausedResourceSoundsOnce

Definition

`PlayAllPausedResourceSoundsOnce([optional] string exception_1, [optional] string exception_2,..., [optional] string exception_n)`

Description

This function plays all *paused* resource sounds once except for the paused resource sounds sent to the function.

Parameters

[optional] string exception_1, [optional] string exception_2,..., [optional] string exception_n

Specifies the name of the paused resource sounds that should not be played by this function. If no name is passed to `PlayAllPausedResourceSoundsOnce` function, all paused resource sounds will be played once.

Example

```
function OnTriggerEnter(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          LoadResource("Sounds", "fire.ogg")
          LoadResource("Sounds", "river.ogg")
          LoadResource("Sounds", "ambient.ogg")
          PlayAllResourceSoundsLoop()
    end
end

function OnTriggerStay(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          PauseAllResourceSounds()
    end
end

function OnTriggerExit(otherActorName)
    if otherActorName == nil then
          PlayAllPausedResourceSoundsOnce("Sounds_fire.ogg", "Sounds_river.ogg")
    end
end
```

Assume that the above script is attached to a trigger named trigger1.
Whenever the main character enters "trigger1", we load and play 3 resource sounds -- In order for `LoadResource` function to load the desired resource, you must first add it through the *Add Resource to Current Project* dialog (File > Project > Add/Remove Resource to/from Current Project). When the main character stays in the trigger, all the resource sounds that are playing will be paused. When the main character exits "trigger1", all the paused resource sounds except the resource sounds `"fire.ogg"` and `"river.ogg"` will be played once.

4.268. PlayAllPausedSounds

Definition

```
PlayAllPausedSounds([optional] string exception_1, [optional] string
exception_2,..., [optional] string exception_n)
```

Description

This function plays all *paused* ambient, 3D and resource sounds except for the paused ambient, 3D and resource sounds sent to the function. If the loop state of each sound is true, the paused sound will be played continuously, otherwise it will be played only once.

Parameters

`[optional] string exception_1, [optional] string exception_2,...,`
`[optional] string exception_n`

Specifies the name of the paused ambient, 3D and resource sounds that should not be played by this function. If no name is passed to `PlayAllPausedSounds` function, all paused sounds will be played.

Example

```
function OnTriggerEnter(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          LoadResource("Sounds", "fire.ogg")
          LoadResource("Sounds", "river.ogg")
          LoadResource("Sounds", "ambient.ogg")
          PlayAllResourceSoundsLoop()
    end
end

function OnTriggerStay(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          PauseAllSounds()
    end
end

function OnTriggerExit(otherActorName)
    if otherActorName == nil then
          PlayAllPausedSounds("ambient2", "river_3D2", "Sounds_fire.ogg",
"Sounds_river.ogg")
    end
end
```

Assume that the above script is attached to a trigger named trigger1. Also assume that `"ambient2"` and `"river_3D2"` are ambient and 3D sound names, respectively.
Whenever the main character enters "trigger1", we load and play 3 resource sounds -- In order for `LoadResource` function to load the desired resource, you must first add it through the *Add Resource to Current Project* dialog (File > Project > Add/Remove Resource to/from Current Project).
When the main character stays in the trigger, all the sounds that are playing will be paused. When

the main character exits "trigger1", all the paused sounds except the ambient sound `"ambient2"`, 3D sound `"river_3D2"` and resource sounds `"fire.ogg"` and `"river.ogg"` will be played.

4.269. PlayAllPausedSoundsLoop

Definition

```
PlayAllPausedSoundsLoop([optional] string exception_1, [optional] string
exception_2,..., [optional] string exception_n)
```

Description

This function plays all *paused* ambient, 3D and resource sounds continuously except for the paused ambient, 3D and resource sounds sent to the function.

Parameters

[optional] string exception_1, [optional] string exception_2,...,
[optional] string exception_n

Specifies the name of the paused ambient, 3D and resource sounds that should not be played by this function. If no name is passed to `PlayAllPausedSoundsLoop` function, all paused sounds will be played continuously.

Example

```
function OnTriggerEnter(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          LoadResource("Sounds", "fire.ogg")
          LoadResource("Sounds", "river.ogg")
          LoadResource("Sounds", "ambient.ogg")
          PlayAllResourceSoundsLoop()
    end
end

function OnTriggerStay(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          PauseAllSounds()
    end
end

function OnTriggerExit(otherActorName)
    if otherActorName == nil then
          PlayAllPausedSoundsLoop("ambient2", "river_3D2", "Sounds_fire.ogg",
"Sounds_river.ogg")
    end
end
```

Assume that the above script is attached to a trigger named trigger1. Also assume that `"ambient2"` and `"river_3D2"` are ambient and 3D sound names, respectively.
Whenever the main character enters "trigger1", we load and play 3 resource sounds -- In order for `LoadResource` function to load the desired resource, you must first add it through the *Add Resource to Current Project* dialog (File > Project > Add/Remove Resource to/from Current Project). When the main character stays in the trigger, all the sounds that are playing will be paused. When the main character exits "trigger1", all the paused sounds except the ambient sound `"ambient2"`,

3D sound `"river_3D2"` and resource sounds `"fire.ogg"` and `"river.ogg"` will be played continuously.

4.270. PlayAllPausedSoundsOnce

Definition

`PlayAllPausedSoundsOnce([optional] string exception_1, [optional] string exception_2,..., [optional] string exception_n)`

Description

This function plays all *paused* ambient, 3D and resource sounds once except for the paused ambient, 3D and resource sounds sent to the function.

Parameters

[optional] string exception_1, [optional] string exception_2,..., [optional] string exception_n

Specifies the name of the paused ambient, 3D and resource sounds that should not be played by this function. If no name is passed to `PlayAllPausedSoundsOnce` function, all paused sounds will be played once.

Example

```
function OnTriggerEnter(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          LoadResource("Sounds", "fire.ogg")
          LoadResource("Sounds", "river.ogg")
          LoadResource("Sounds", "ambient.ogg")
          PlayAllResourceSoundsLoop()
    end
end

function OnTriggerStay(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          PauseAllSounds()
    end
end

function OnTriggerExit(otherActorName)
    if otherActorName == nil then
          PlayAllPausedSoundsOnce("ambient2", "river_3D2", "Sounds_fire.ogg",
"Sounds_river.ogg")
    end
end
```

Assume that the above script is attached to a trigger named trigger1. Also assume that `"ambient2"` and `"river_3D2"` are ambient and 3D sound names, respectively.
Whenever the main character enters "trigger1", we load and play 3 resource sounds -- In order for `LoadResource` function to load the desired resource, you must first add it through the *Add Resource to Current Project* dialog (File > Project > Add/Remove Resource to/from Current Project).
When the main character stays in the trigger, all the sounds that are playing will be paused. When the main character exits "trigger1", all the paused sounds except the ambient sound `"ambient2"`,

3D sound `"river_3D2"` and resource sounds `"fire.ogg"` and `"river.ogg"` will be played once.

4.271. PlayAllResourceSounds

Definition

`PlayAllResourceSounds([optional] string exception_1, [optional] string exception_2,..., [optional] string exception_n)`

Description

This function plays all resource sounds except for the resource sounds sent to the function. If the loop state of each resource sound is true (For example, if it is played by the `PlayResourceSoundsLoop` function and then paused by the `PauseResourceSound` function), the resource sound will be played continuously, otherwise it will be played only once.

Parameters

`[optional] string exception_1, [optional] string exception_2,..., [optional] string exception_n`

Specifies the name of the resource sounds that should not be played by this function. If no name is passed to `PlayAllResourceSounds` function, all resource sounds will be played.

Example

```
function OnTriggerEnter(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
        LoadResource("Sounds", "fire.ogg")
        LoadResource("Sounds", "river.ogg")
        LoadResource("Sounds", "ambient.ogg")
    end
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)
    if otherActorName == nil then
        PlayAllResourceSounds("Sounds_fire.ogg", "Sounds_river.ogg")
    end
end
```

Assume that the above script is attached to a trigger named trigger1.
Whenever the main character enters "trigger1", we load 3 resource sounds -- In order for `LoadResource` function to load the desired resource, you must first add it through the *Add Resource to Current Project* dialog (File > Project > Add/Remove Resource to/from Current Project).
When the main character exits "trigger1", all the resource sounds except the resource sounds `"fire.ogg"` and `"river.ogg"` will be played.

4.272. PlayAllResourceSoundsLoop

Definition

`PlayAllResourceSoundsLoop([optional] string exception_1, [optional] string exception_2,..., [optional] string exception_n)`

Description

This function plays all resource sounds continuously except for the resource sounds sent to the function.

Parameters

`[optional] string exception_1, [optional] string exception_2,..., [optional] string exception_n`

Specifies the name of the resource sounds that should not be played by this function. If no name is passed to `PlayAllResourceSoundsLoop` function, all resource sounds will be played continuously.

Example

```
function OnTriggerEnter(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          LoadResource("Sounds", "fire.ogg")
          LoadResource("Sounds", "river.ogg")
          LoadResource("Sounds", "ambient.ogg")
    end
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)
    if otherActorName == nil then
          PlayAllResourceSoundsLoop("Sounds_fire.ogg", "Sounds_river.ogg")
    end
end
```

Assume that the above script is attached to a trigger named trigger1.
Whenever the main character enters "trigger1", we load 3 resource sounds -- In order for `LoadResource` function to load the desired resource, you must first add it through the *Add Resource to Current Project* dialog (File > Project > Add/Remove Resource to/from Current Project).
When the main character exits "trigger1", all the resource sounds except the resource sounds `"fire.ogg"` and `"river.ogg"` will be played continuously.

4.273. PlayAllResourceSoundsOnce

Definition

`PlayAllResourceSoundsOnce([optional] string exception_1, [optional] string exception_2,..., [optional] string exception_n)`

Description

This function plays all resource sounds once except for the resource sounds sent to the function.

Parameters

[optional] string exception_1, [optional] string exception_2,..., [optional] string exception_n

Specifies the name of the resource sounds that should not be played by this function. If no name is passed to `PlayAllResourceSoundsOnce` function, all resource sounds will be played once.

Example

```
function OnTriggerEnter(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
        LoadResource("Sounds", "fire.ogg")
        LoadResource("Sounds", "river.ogg")
        LoadResource("Sounds", "ambient.ogg")
    end
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)
    if otherActorName == nil then
        PlayAllResourceSoundsOnce("Sounds_fire.ogg", "Sounds_river.ogg")
    end
end
```

Assume that the above script is attached to a trigger named trigger1.
Whenever the main character enters "trigger1", we load 3 resource sounds -- In order for `LoadResource` function to load the desired resource, you must first add it through the *Add Resource to Current Project* dialog (File > Project > Add/Remove Resource to/from Current Project).
When the main character exits "trigger1", all the resource sounds except the resource sounds `"fire.ogg"` and `"river.ogg"` will be played once.

4.274. PlayAllSounds

Definition

```
PlayAllSounds([optional] string exception_1, [optional] string
exception_2,..., [optional] string exception_n)
```

Description

This function plays all ambient, 3D and resource sounds except for the ambient, 3D and resource sounds sent to the function. If the loop state of each sound is true, the sound will be played continuously, otherwise it will be played only once.

Parameters

`[optional] string exception_1, [optional] string exception_2,..., [optional] string exception_n`

Specifies the name of the ambient, 3D and resource sounds that should not be played by this function. If no name is passed to `PlayAllSounds` function, all sounds will be played.

Example

```
function OnTriggerEnter(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          LoadResource("Sounds", "fire.ogg")
          LoadResource("Sounds", "river.ogg")
          LoadResource("Sounds", "ambient.ogg")
    end
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)
    if otherActorName == nil then
          PlayAllSounds("ambient2", "river3D_2", "Sounds_fire.ogg", "Sounds_river.ogg")
    end
end
```

Assume that the above script is attached to a trigger named trigger1. Also assume that `"ambient2"` and `"river_3D2"` are ambient and 3D sound names, respectively.
Whenever the main character enters "trigger1", we load 3 resource sounds -- In order for `LoadResource` function to load the desired resource, you must first add it through the *Add Resource to Current Project* dialog (File > Project > Add/Remove Resource to/from Current Project).
When the main character exits "trigger1", all the sounds except the ambient sound `"ambient2"`, 3D sound `"river_3D2"` and resource sounds `"fire.ogg"` and `"river.ogg"` will be played.

4.275. PlayAllSoundsLoop

Definition

`PlayAllSoundsLoop([optional] string exception_1, [optional] string exception_2,..., [optional] string exception_n)`

Description

This function plays all ambient, 3D and resource sounds continuously except for the ambient, 3D and resource sounds sent to the function.

Parameters

`[optional] string exception_1, [optional] string exception_2,..., [optional] string exception_n`

Specifies the name of the ambient, 3D and resource sounds that should not be played by this function. If no name is passed to `PlayAllSoundsLoop` function, all sounds will be played continuously.

Example

```
function OnTriggerEnter(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          LoadResource("Sounds", "fire.ogg")
          LoadResource("Sounds", "river.ogg")
          LoadResource("Sounds", "ambient.ogg")
    end
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)
    if otherActorName == nil then
          PlayAllSoundsLoop("ambient2", "river3D_2", "Sounds_fire.ogg",
"Sounds_river.ogg")
    end
end
```

Assume that the above script is attached to a trigger named trigger1. Also assume that `"ambient2"` and `"river_3D2"` are ambient and 3D sound names, respectively. Whenever the main character enters "trigger1", we load 3 resource sounds -- In order for `LoadResource` function to load the desired resource, you must first add it through the *Add Resource to Current Project* dialog (File > Project > Add/Remove Resource to/from Current Project). When the main character exits "trigger1", all the sounds except the ambient sound `"ambient2"`, 3D sound `"river_3D2"` and resource sounds `"fire.ogg"` and `"river.ogg"` will be played continuously.

4.276. PlayAllSoundsOnce

Definition

`PlayAllSoundsOnce([optional] string exception_1, [optional] string exception_2,..., [optional] string exception_n)`

Description

This function plays all ambient, 3D and resource sounds once except for the ambient, 3D and resource sounds sent to the function.

Parameters

`[optional] string exception_1, [optional] string exception_2,..., [optional] string exception_n`

Specifies the name of the ambient, 3D and resource sounds that should not be played by this function. If no name is passed to `PlayAllSoundsOnce` function, all sounds will be played once.

Example

```
function OnTriggerEnter(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          LoadResource("Sounds", "fire.ogg")
          LoadResource("Sounds", "river.ogg")
          LoadResource("Sounds", "ambient.ogg")
    end
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)
    if otherActorName == nil then
          PlayAllSoundsOnce("ambient2", "river3D_2", "Sounds_fire.ogg",
"Sounds_river.ogg")
    end
end
```

Assume that the above script is attached to a trigger named trigger1. Also assume that `"ambient2"` and `"river_3D2"` are ambient and 3D sound names, respectively.
Whenever the main character enters "trigger1", we load 3 resource sounds -- In order for `LoadResource` function to load the desired resource, you must first add it through the *Add Resource to Current Project* dialog (File > Project > Add/Remove Resource to/from Current Project).
When the main character exits "trigger1", all the sounds except the ambient sound `"ambient2"`, 3D sound `"river_3D2"` and resource sounds `"fire.ogg"` and `"river.ogg"` will be played once.

4.277. PlayAllStopped3DSounds

Definition

`PlayAllStopped3DSounds([optional] string exception_1, [optional] string exception_2,..., [optional] string exception_n)`

Description

This function plays all *stopped* 3D sounds except for the stopped 3D sounds sent to the function. If the loop state of each 3D sound is true, the stopped 3D sound will be played continuously, otherwise it will be played only once.

Parameters

`[optional] string exception_1, [optional] string exception_2,...,`
`[optional] string exception_n`

Specifies the name of the stopped 3D sounds that should not be played by this function. If no name is passed to `PlayAllStopped3DSounds` function, all stopped 3D sounds will be played.

Example

```lua
function OnTriggerEnter(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
        StopAll3DSounds()
    end
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)
    if otherActorName == nil then
        PlayAllStopped3DSounds("river2", "river3")
    end
end
```

Assume that the above script is attached to a trigger named trigger1. Also assume that `"river2"` and `"river3"` are 3D sound objects.
Whenever the main character enters "trigger1", all the 3D sounds that are playing will be stopped. When the main character exits "trigger1", all the stopped 3D sounds except the 3D sounds `"river2"` and `"river3"` will be played.

4.278. PlayAllStopped3DSoundsLoop

Definition

`PlayAllStopped3DSoundsLoop([optional] string exception_1, [optional] string exception_2,..., [optional] string exception_n)`

Description

This function plays all *stopped* 3D sounds continuously except for the stopped 3D sounds sent to the function.

Parameters

[optional] string exception_1, [optional] string exception_2,..., [optional] string exception_n

Specifies the name of the stopped 3D sounds that should not be played by this function. If no name is passed to `PlayAllStopped3DSoundsLoop` function, all stopped 3D sounds will be played continuously.

Example

```
function OnTriggerEnter(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          StopAll3DSounds()
    end
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)
    if otherActorName == nil then
          PlayAllStopped3DSoundsLoop("river2", "river3")
    end
end
```

Assume that the above script is attached to a trigger named trigger1. Also assume that `"river2"` and `"river3"` are 3D sound objects.
Whenever the main character enters "trigger1", all the 3D sounds that are playing will be stopped. When the main character exits "trigger1", all the stopped 3D sounds except the 3D sounds `"river2"` and `"river3"` will be played continuously.

4.279. PlayAllStopped3DSoundsOnce

Definition

`PlayAllStopped3DSoundsOnce([optional] string exception_1, [optional] string exception_2,..., [optional] string exception_n)`

Description

This function plays all *stopped* 3D sounds once except for the stopped 3D sounds sent to the function.

Parameters

`[optional] string exception_1, [optional] string exception_2,..., [optional] string exception_n`

Specifies the name of the stopped 3D sounds that should not be played by this function. If no name is passed to `PlayAllStopped3DSoundsOnce` function, all stopped 3D sounds will be played once.

Example

```
function OnTriggerEnter(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          StopAll3DSounds()
    end
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)
    if otherActorName == nil then
          PlayAllStopped3DSoundsOnce("river2", "river3")
    end
end
```

Assume that the above script is attached to a trigger named trigger1. Also assume that `"river2"` and `"river3"` are 3D sound objects.
Whenever the main character enters "trigger1", all the 3D sounds that are playing will be stopped. When the main character exits "trigger1", all the stopped 3D sounds except the 3D sounds `"river2"` and `"river3"` will be played once.

4.280. PlayAllStoppedAmbientSounds

Definition

`PlayAllStoppedAmbientSounds([optional] string exception_1, [optional] string exception_2,..., [optional] string exception_n)`

Description

This function plays all *stopped* ambient sounds except for the stopped ambient sounds sent to the function. If the loop state of each ambient sound is true, the stopped ambient sound will be played continuously, otherwise it will be played only once.

Parameters

`[optional] string exception_1, [optional] string exception_2,...,`
`[optional] string exception_n`
Specifies the name of the stopped ambient sounds that should not be played by this function. If no name is passed to `PlayAllStoppedAmbientSounds` function, all stopped ambient sounds will be played.

Example

```
function OnTriggerEnter(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          StopAllAmbientSounds()
    end
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)
    if otherActorName == nil then
          PlayAllStoppedAmbientSounds("ambient2", "ambient3")
    end
end
```

Assume that the above script is attached to a trigger named trigger1. Also assume that `"ambient2"` and `"ambient3"` are ambient sound objects.
Whenever the main character enters "trigger1", all the ambient sounds that are playing will be stopped. When the main character exits "trigger1", all the stopped ambient sounds except the ambient sounds `"ambient2"` and `"ambient3"` will be played.

4.281. PlayAllStoppedAmbientSoundsLoop

Definition

`PlayAllStoppedAmbientSoundsLoop([optional] string exception_1, [optional] string exception_2,..., [optional] string exception_n)`

Description

This function plays all *stopped* ambient sounds continuously except for the stopped ambient sounds sent to the function.

Parameters

`[optional] string exception_1, [optional] string exception_2,..., [optional] string exception_n`

Specifies the name of the stopped ambient sounds that should not be played by this function. If no name is passed to `PlayAllStoppedAmbientSoundsLoop` function, all stopped ambient sounds will be played continuously.

Example

```
function OnTriggerEnter(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          StopAllAmbientSounds()
    end
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)
    if otherActorName == nil then
          PlayAllStoppedAmbientSoundsLoop("ambient2", "ambient3")
    end
end
```

Assume that the above script is attached to a trigger named trigger1. Also assume that `"ambient2"` and `"ambient3"` are ambient sound objects.
Whenever the main character enters "trigger1", all the ambient sounds that are playing will be stopped. When the main character exits "trigger1", all the stopped ambient sounds except the ambient sounds `"ambient2"` and `"ambient3"` will be played continuously.

4.282. PlayAllStoppedAmbientSoundsOnce

Definition

```
PlayAllStoppedAmbientSoundsOnce([optional] string exception_1, [optional]
string exception_2,..., [optional] string exception_n)
```

Description

This function plays all *stopped* ambient sounds once except for the stopped ambient sounds sent to the function.

Parameters

[optional] string exception_1, [optional] string exception_2,..., [optional] string exception_n

Specifies the name of the stopped ambient sounds that should not be played by this function. If no name is passed to `PlayAllStoppedAmbientSoundsOnce` function, all stopped ambient sounds will be played once.

Example

```
function OnTriggerEnter(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          StopAllAmbientSounds()
    end
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)
    if otherActorName == nil then
          PlayAllStoppedAmbientSoundsOnce("ambient2", "ambient3")
    end
end
```

Assume that the above script is attached to a trigger named trigger1. Also assume that `"ambient2"` and `"ambient3"` are ambient sound objects.
Whenever the main character enters "trigger1", all the ambient sounds that are playing will be stopped. When the main character exits "trigger1", all the stopped ambient sounds except the ambient sounds `"ambient2"` and `"ambient3"` will be played once.

4.283. PlayAllStoppedResourceSounds

Definition

`PlayAllStoppedResourceSounds([optional] string exception_1, [optional] string exception_2,..., [optional] string exception_n)`

Description

This function plays all *stopped* resource sounds except for the stopped resource sounds sent to the function. If the loop state of each resource sound is true (For example, if it is played by the `PlayResourceSoundsLoop` function and then stopped by the `StopResourceSound` function), the stopped resource sound will be played continuously, otherwise it will be played only once.

Parameters

`[optional] string exception_1, [optional] string exception_2,..., [optional] string exception_n`

Specifies the name of the stopped resource sounds that should not be played by this function. If no name is passed to `PlayAllStoppedResourceSounds` function, all stopped resource sounds will be played.

Example

```
function OnTriggerEnter(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
        LoadResource("Sounds", "fire.ogg")
        LoadResource("Sounds", "river.ogg")
        LoadResource("Sounds", "ambient.ogg")
        PlayAllResourceSoundsLoop()
    end
end

function OnTriggerStay(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
        StopAllResourceSounds()
    end
end

function OnTriggerExit(otherActorName)
    if otherActorName == nil then
        PlayAllStoppedResourceSounds("Sounds_fire.ogg", "Sounds_river.ogg")
    end
end
```

Assume that the above script is attached to a trigger named trigger1.
Whenever the main character enters "trigger1", we load and play 3 resource sounds -- In order for `LoadResource` function to load the desired resource, you must first add it through the *Add Resource to Current Project* dialog (File > Project > Add/Remove Resource to/from Current Project). When the main character stays in the trigger, all the resource sounds that are playing will be stopped. When the main character exits "trigger1", all the stopped resource sounds except the resource sounds `"fire.ogg"` and `"river.ogg"` will be played.

4.284. PlayAllStoppedResourceSoundsLoop

Definition

`PlayAllStoppedResourceSoundsLoop([optional] string exception_1, [optional] string exception_2,..., [optional] string exception_n)`

Description

This function plays all *stopped* resource sounds continuously except for the stopped resource sounds sent to the function.

Parameters

[optional] string exception_1, [optional] string exception_2,..., [optional] string exception_n

Specifies the name of the stopped resource sounds that should not be played by this function. If no name is passed to `PlayAllStoppedResourceSoundsLoop` function, all stopped resource sounds will be played continuously.

Example

```
function OnTriggerEnter(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          LoadResource("Sounds", "fire.ogg")
          LoadResource("Sounds", "river.ogg")
          LoadResource("Sounds", "ambient.ogg")
          PlayAllResourceSoundsLoop()
    end
end

function OnTriggerStay(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          StopAllResourceSounds()
    end
end

function OnTriggerExit(otherActorName)
    if otherActorName == nil then
          PlayAllStoppedResourceSoundsLoop("Sounds_fire.ogg", "Sounds_river.ogg")
    end
end
```

Assume that the above script is attached to a trigger named trigger1.
Whenever the main character enters "trigger1", we load and play 3 resource sounds -- In order for `LoadResource` function to load the desired resource, you must first add it through the *Add Resource to Current Project* dialog (File > Project > Add/Remove Resource to/from Current Project).
When the main character stays in the trigger, all the resource sounds that are playing will be stopped. When the main character exits "trigger1", all the stopped resource sounds except the resource sounds `"fire.ogg"` and `"river.ogg"` will be played continuously.

4.285. PlayAllStoppedResourceSoundsOnce

Definition

```
PlayAllStoppedResourceSoundsOnce([optional] string exception_1, [optional] string exception_2,..., [optional] string exception_n)
```

Description

This function plays all *stopped* resource sounds once except for the stopped resource sounds sent to the function.

Parameters

[optional] string exception_1, [optional] string exception_2,..., [optional] string exception_n

Specifies the name of the stopped resource sounds that should not be played by this function. If no name is passed to `PlayAllStoppedResourceSoundsOnce` function, all stopped resource sounds will be played once.

Example

```
function OnTriggerEnter(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          LoadResource("Sounds", "fire.ogg")
          LoadResource("Sounds", "river.ogg")
          LoadResource("Sounds", "ambient.ogg")
          PlayAllResourceSoundsLoop()
    end
end

function OnTriggerStay(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          StopAllResourceSounds()
    end
end

function OnTriggerExit(otherActorName)
    if otherActorName == nil then
          PlayAllStoppedResourceSoundsOnce("Sounds_fire.ogg", "Sounds_river.ogg")
    end
end
```

Assume that the above script is attached to a trigger named trigger1.
Whenever the main character enters "trigger1", we load and play 3 resource sounds -- In order for `LoadResource` function to load the desired resource, you must first add it through the *Add Resource to Current Project* dialog (File > Project > Add/Remove Resource to/from Current Project).
When the main character stays in the trigger, all the resource sounds that are playing will be stopped. When the main character exits "trigger1", all the stopped resource sounds except the resource sounds `"fire.ogg"` and `"river.ogg"` will be played once.

4.286. PlayAllStoppedSounds

Definition

`PlayAllStoppedSounds([optional] string exception_1, [optional] string exception_2,..., [optional] string exception_n)`

Description

This function plays all *stopped* ambient, 3D and resource sounds except for the stopped ambient, 3D and resource sounds sent to the function. If the loop state of each sound is true, the stopped sound will be played continuously, otherwise it will be played only once.

Parameters

`[optional] string exception_1, [optional] string exception_2,...,`
`[optional] string exception_n`

Specifies the name of the stopped ambient, 3D and resource sounds that should not be played by this function. If no name is passed to `PlayAllStoppedSounds` function, all stopped sounds will be played.

Example

```
function OnTriggerEnter(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          LoadResource("Sounds", "fire.ogg")
          LoadResource("Sounds", "river.ogg")
          LoadResource("Sounds", "ambient.ogg")
          PlayAllResourceSoundsLoop()
    end
end

function OnTriggerStay(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          StopAllSounds()
    end
end

function OnTriggerExit(otherActorName)
    if otherActorName == nil then
          PlayAllStoppedSounds("ambient2", "river_3D2", "Sounds_fire.ogg",
"Sounds_river.ogg")
    end
end
```

Assume that the above script is attached to a trigger named trigger1. Also assume that `"ambient2"` and `"river_3D2"` are ambient and 3D sound names, respectively.
Whenever the main character enters "trigger1", we load and play 3 resource sounds -- In order for `LoadResource` function to load the desired resource, you must first add it through the *Add Resource to Current Project* dialog (File > Project > Add/Remove Resource to/from Current Project).
When the main character stays in the trigger, all the sounds that are playing will be stopped.
When the main character exits "trigger1", all the stopped sounds except the ambient sound

`"ambient2"`, 3D sound `"river_3D2"` and resource sounds `"fire.ogg"` and `"river.ogg"` will be played.

4.287. PlayAllStoppedSoundsLoop

Definition

```
PlayAllStoppedSoundsLoop([optional] string exception_1, [optional] string
exception_2,..., [optional] string exception_n)
```

Description

This function plays all *stopped* ambient, 3D and resource sounds continuously except for the stopped ambient, 3D and resource sounds sent to the function.

Parameters

[optional] string exception_1, [optional] string exception_2,...,
[optional] string exception_n

Specifies the name of the stopped ambient, 3D and resource sounds that should not be played by this function. If no name is passed to `PlayAllStoppedSoundsLoop` function, all stopped sounds will be played continuously.

Example

```
function OnTriggerEnter(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          LoadResource("Sounds", "fire.ogg")
          LoadResource("Sounds", "river.ogg")
          LoadResource("Sounds", "ambient.ogg")
          PlayAllResourceSoundsLoop()
    end
end

function OnTriggerStay(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          StopAllSounds()
    end
end

function OnTriggerExit(otherActorName)
    if otherActorName == nil then
          PlayAllStoppedSoundsLoop("ambient2", "river_3D2", "Sounds_fire.ogg",
"Sounds_river.ogg")
    end
end
```

Assume that the above script is attached to a trigger named trigger1. Also assume that `"ambient2"` and `"river_3D2"` are ambient and 3D sound names, respectively.
Whenever the main character enters "trigger1", we load and play 3 resource sounds -- In order for `LoadResource` function to load the desired resource, you must first add it through the *Add Resource to Current Project* dialog (File > Project > Add/Remove Resource to/from Current Project).
When the main character stays in the trigger, all the sounds that are playing will be stopped.
When the main character exits "trigger1", all the stopped sounds except the ambient sound

`"ambient2"`, 3D sound `"river_3D2"` and resource sounds `"fire.ogg"` and `"river.ogg"` will be played continuously.

4.288. PlayAllStoppedSoundsOnce

Definition

`PlayAllStoppedSoundsOnce([optional] string exception_1, [optional] string exception_2,..., [optional] string exception_n)`

Description

This function plays all *stopped* ambient, 3D and resource sounds once except for the stopped ambient, 3D and resource sounds sent to the function.

Parameters

`[optional] string exception_1, [optional] string exception_2,..., [optional] string exception_n`

Specifies the name of the stopped ambient, 3D and resource sounds that should not be played by this function. If no name is passed to `PlayAllStoppedSoundsOnce` function, all stopped sounds will be played once.

Example

```
function OnTriggerEnter(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          LoadResource("Sounds", "fire.ogg")
          LoadResource("Sounds", "river.ogg")
          LoadResource("Sounds", "ambient.ogg")
          PlayAllResourceSoundsLoop()
    end
end

function OnTriggerStay(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          StopAllSounds()
    end
end

function OnTriggerExit(otherActorName)
    if otherActorName == nil then
          PlayAllStoppedSoundsOnce("ambient2", "river_3D2", "Sounds_fire.ogg",
"Sounds_river.ogg")
    end
end
```

Assume that the above script is attached to a trigger named trigger1. Also assume that `"ambient2"` and `"river_3D2"` are ambient and 3D sound names, respectively.
Whenever the main character enters "trigger1", we load and play 3 resource sounds -- In order for `LoadResource` function to load the desired resource, you must first add it through the *Add Resource to Current Project* dialog (File > Project > Add/Remove Resource to/from Current Project).
When the main character stays in the trigger, all the sounds that are playing will be stopped.
When the main character exits "trigger1", all the stopped sounds except the ambient sound

`"ambient2"`, 3D sound `"river_3D2"` and resource sounds `"fire.ogg"` and `"river.ogg"` will be played once.

4.289. PlayResourceSound

Definition

`PlayResourceSound(string resourceDirectoryName_resourceFileName.ogg)`

Description

This function plays the resource sound `resourceDirectoryName_resourceFileName.ogg`. If the loop state of resource sound is true, the sound will be played continuously, otherwise it will be played only once.
You can go to the *Project Resources* section through the Script Utility dialog (Tools > Script Editor > Tools > Script Utility), select the desired resource sound and hit "Copy Folder_File Name" button to copy the full name of the resource.

Parameters

`resourceDirectoryName_resourceFileName.ogg`
Specifies the full name of the resource sound.

Example 1

```
function OnTriggerEnter(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          LoadResource("Sounds", "fire.ogg")
          PlayResourceSoundLoop("Sounds_fire.ogg")
          StopResourceSound("Sounds_fire.ogg")
    end
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          PlayResourceSound("Sounds_fire.ogg")
    end
end
```

Assume that the above script is attached to a trigger named "trigger1". Whenever the main character enters "trigger1", we load `"fire.ogg"` resource sound located in `"Sounds"` directory--In order for `LoadResource` function to load the resource sound, you must first add `"fire.ogg"` sound resource through the *Add Resource to Current Project* dialog (File > Project > Add/Remove Resource to/from Current Project). Then we play this sound continuously using the `PlayResourceSoundLoop` function. This function sets the loop state of the resource `"fire.ogg"` to *true*. Then we stop the `"fire.ogg"` resource sound using the `StopResourceSound` function.
Since the loop state of the `"fire.ogg"` sound is set to *true* by the `PlayResourceSoundLoop` function when the main character enters the trigger "trigger1", the `PlayResourceSound` function plays the `"fire.ogg"` sound *continuously* when the main character leaves the trigger "trigger1".

Example 2

```
function OnTriggerEnter(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          LoadResource("Sounds", "fire.ogg")
          PlayAllResourceSoundsOnce()
          StopAllResourceSounds()
    end
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          PlayResourceSound("Sounds_fire.ogg")
    end
end
```

Assume that the above script is attached to a trigger named "trigger1". Whenever the main character enters "trigger1", we load `"fire.ogg"` resource sound located in `"Sounds"` directory. Then we play all the resource sounds once using the `PlayAllResourceSoundsOnce` function. This function sets the loop state of all resource sounds, including `"fire.ogg"`, which is located in the `"Sounds"` folder, to *false*. Then, using the `StopAllResourceSounds` function, we stop all resource sounds.
Since the loop state of the `"fire.ogg"` sound is set to *false* by the `PlayAllResourceSoundsOnce` function when the main character enters the trigger "trigger1", the `PlayResourceSound` function plays the `"fire.ogg"` sound *only once* when the main character leaves the trigger "trigger1".

4.290. PlayResourceSoundLoop

Definition

`PlayResourceSoundLoop(string resourceDirectoryName_resourceFileName.ogg)`

Description

This function plays resource sound `resourceDirectoryName_resourceFileName.ogg` continuously. You can go to the *Project Resources* section through the Script Utility dialog (Tools > Script Editor > Tools > Script Utility), select the desired resource sound and hit "Copy Folder_File Name" button to copy the full name of the resource.

Parameters

`resourceDirectoryName_resourceFileName.ogg`
Specifies the full name of the resource sound.

Example

```
function OnTriggerEnter(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
        LoadResource("Sounds", "fire.ogg")
    end
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
        PlayResourceSoundLoop("Sounds_fire.ogg")
    end
end
```

Assume that the above script is attached to a trigger named trigger1. Whenever the main character enters "trigger1", we load `"fire.ogg"` resource sound located in `"Sounds"` directory--In order for `LoadResource` function to load the resource sound, you must first add `"fire.ogg"` sound resource through the *Add Resource to Current Project* dialog (File > Project > Add/Remove Resource to/from Current Project).
When the main character exits "trigger1", the resource sound `"fire.ogg"` will be played continuously.

4.291. PlayResourceSoundOnce

Definition

`PlayResourceSoundOnce(string resourceDirectoryName_resourceFileName.ogg)`

Description

This function plays resource sound `resourceDirectoryName_resourceFileName.ogg` once. You can go to the *Project Resources* section through the Script Utility dialog (Tools > Script Editor > Tools > Script Utility), select the desired resource sound and hit "Copy Folder_File Name" button to copy the full name of the resource.

Parameters

resourceDirectoryName_resourceFileName.ogg

Specifies the full name of the resource sound.

Example

```
function OnTriggerEnter(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          LoadResource("Sounds", "fire.ogg")
    end
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          PlayResourceSoundOnce("Sounds_fire.ogg")
    end
end
```

Assume that the above script is attached to a trigger named trigger1. Whenever the main character enters "trigger1", we load `"fire.ogg"` resource sound located in `"Sounds"` directory--In order for `LoadResource` function to load the resource sound, you must first add `"fire.ogg"` sound resource through the *Add Resource to Current Project* dialog (File > Project > Add/Remove Resource to/from Current Project).
When the main character exits "trigger1", the resource sound `"fire.ogg"` will be played once.

4.292. PlaySound

Definition

`PlaySound(string soundObjectName1, string soundObjectName2, ..., string soundObjectNameN)`

Description

This function plays ambient and 3D sounds `soundObjectName1, soundObjectName2, ..., soundObjectNameN`. If the loop state of each ambient or 3D sound is true, the sound will be played continuously, otherwise it will be played only once.

Parameters

`soundObjectName1, soundObjectName2, ..., soundObjectNameN`

Specify the name of the ambient and 3D sounds that should be played by this function. You can also use the name "this" for *`soundObjectName[N]`*. In this case, "this" refers to the ambient or 3D sound that this script is attached to.

Example

```
function Init()
    PlaySound("this", "river")
end

function Update()

end
```

Assume that the above script is attached to an ambient sound named "ambient1". Also, `"river"` in the example above is the name of a 3D sound. In our example, `PlaySound` function plays the current sound (which has a name equivalent to "ambient1"), and the 3D sound `"river"`.

4.293. PlaySoundLoop

Definition

`PlaySoundLoop(string soundObjectName1, string soundObjectName2, ..., string soundObjectNameN)`

Description

This function plays ambient and 3D sounds `soundObjectName1, soundObjectName2, ..., soundObjectNameN` continuously.

Parameters

`soundObjectName1, soundObjectName2, ..., soundObjectNameN`

Specify the name of the ambient and 3D sounds that should be played continuously by this function. You can also use the name "this" for ***`soundObjectName[N]`***. In this case, "this" refers to the ambient or 3D sound that this script is attached to.

Example

```
function Init()
    PlaySoundLoop("this", "river")
end

function Update()

end
```

Assume that the above script is attached to an ambient sound named "ambient1". Also, `"river"` in the example above is the name of a 3D sound. In our example, `PlaySoundLoop` function plays the current sound (which has a name equivalent to "ambient1"), and the 3D sound `"river"` continuously.

4.294. PlaySoundOnce

Definition

`PlaySoundOnce(string soundObjectName1, string soundObjectName2, ..., string soundObjectNameN)`

Description

This function plays ambient and 3D sounds `soundObjectName1, soundObjectName2, ..., soundObjectNameN` once.

Parameters

soundObjectName1, soundObjectName2, ..., soundObjectNameN

Specify the name of the ambient and 3D sounds that should be played once by this function. You can also use the name "this" for ***soundObjectName[N]***. In this case, "this" refers to the ambient or 3D sound that this script is attached to.

Example

```
function Init()
    PlaySoundOnce("this", "river")
end

function Update()

end
```

Assume that the above script is attached to an ambient sound named "ambient1". Also, `"river"` in the example above is the name of a 3D sound. In our example, `PlaySoundOnce` function plays the current sound (which has a name equivalent to "ambient1"), and the 3D sound `"river"` once.

4.295. PlayVideo

Definition

`PlayVideo(string videoName)`

Description

This function plays video **`videoName`**. If the loop state of video is true, it will be played continuously, otherwise it will be played only once.

Parameters

`videoName`

Specifies the name of the video object. You can also use the name "this" for this parameter. In this case, "this" refers to the video object that this script is attached to.

Example 1

```lua
--Name of script is PlayVideo1.lua

function Init()
    PlayVideo("this")
end

function Update()

end
```

In this case, `"this"` string in the `PlayVideo` function points to the video that `PlayVideo1.lua` script is attached to. For example, if `PlayVideo1.lua` script is attached to a video object named "video1", `"this"` will be equivalent to the name "video1". In our example, `PlayVideo` function plays the current video object, which is "video1".

Example 2

```lua
function OnTriggerEnter(otherActorName)
    PlayVideo("video1")
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)

end
```

Assume that the above script is attached to a trigger named trigger1. Whenever the main character or a prefab instance that has dynamic physics enters "trigger1", video `"video1"` will be played.

4.296. PlayVideoLoop

Definition

`PlayVideoLoop(string videoName)`

Description

This function plays video **`videoName`** continuously.

Parameters

`videoName`

Specifies the name of the video object. You can also use the name "this" for this parameter. In this case, "this" refers to the video object that this script is attached to.

Example 1

```
--Name of script is PlayVideoLoop1.lua

function Init()
    PlayVideoLoop("this")
end

function Update()

end
```

In this case, `"this"` string in the `PlayVideoLoop` points to the video that `PlayvideoLoop1.lua` script is attached to. For example, if `PlayVideoLoop1.lua` script is attached to a video object named "video1", `"this"` will be equivalent to the name "video1". In our example, `PlayVideoLoop` function plays the current video object, which is "video1", continuously.

Example 2

```
function OnTriggerEnter(otherActorName)
    PlayVideoLoop("video1")
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)

end
```

Assume that the above script is attached to a trigger named trigger1. Whenever the main character or a prefab instance that has dynamic physics enters "trigger1", video `"video1"` will be played continuously.

4.297. PlayVideoOnce

Definition

`PlayVideoOnce(string videoName)`

Description

This function plays video **`videoName`** once.

Parameters

`videoName`

Specifies the name of the video object. You can also use the name "this" for this parameter. In this case, "this" refers to the video object that this script is attached to.

Example 1

```lua
--Name of script is PlayVideoOnce1.lua

function Init()
    PlayVideoOnce("this")
end

function Update()

end
```

In this case, `"this"` string in the `PlayVideoOnce` points to the video that `PlayvideoOnce1.lua` script is attached to. For example, if `PlayVideoOnce1.lua` script is attached to a video object named "video1", `"this"` will be equivalent to the name "video1". In our example, `PlayVideoOnce` function plays the current video object, which is "video1", once.

Example 2

```lua
function OnTriggerEnter(otherActorName)
    PlayVideoOnce("video1")
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)

end
```

Assume that the above script is attached to a trigger named trigger1. Whenever the main character or a prefab instance that has dynamic physics enters "trigger1", video `"video1"` will be played once.

4.298. PrintConsole

Definition

`PrintConsole(string message)`

Description

This function displays the `message` text in the console of Vanda Engine editor.

Parameters

`message`

Specifies the text to be displayed on the console.

Example

```
function Init()
    message = string.format("\nHello World!")
    PrintConsole(message)
end

function Update()

end
```

The `PrintConsole` function In this example displays the message `Hello World!` in the console of editor.

4.299. ReadBoolVariableFromFile

Definition

```
bool ReadBoolVariableFromFile()
```

Description

This function reads a boolean variable from the currently open file. Before reading information from the file, make sure that you have opened the desired file for reading with the `OpenFileForReading` function.

Return Value

This function returns a boolean value.

Example

```
bVar = false

function Init()
   --Create a folder in Assets/Data/ path
   CreateFolder("Lev1")

   --Create and open file to write data
   OpenFileForWriting("Lev1/level1.bin")
   WriteBoolVariableToFile(true)
   CloseFile("Lev1/level1.bin")

   --Open File to load data
   OpenFileForReading("Lev1/level1.bin")
   bVar = ReadBoolVariableFromFile()
   CloseFile("Lev1/level1.bin")
end
```

First, using the `CreateFolder` function, we create a folder called `"Lev1"` in the Assets/Data/ path. Then, using the `OpenFileForWriting` function, we open the `level1.bin` file located in the Assets/Data/`Lev1`/ path for writing. After writing the Boolean value by the `WriteBoolVariableToFile` function, we close the file by the `CloseFile` function.
Then, using the `OpenFileForReading` function, we open the `level1.bin` file located in the path Assets/Data/`Lev1`/ for reading and read a boolean variable from the `level1.bin` file with the `ReadBoolVariableFromFile()` function. In our example, value of `bVar` is `true` after reading it. Finally, we close the file by the `CloseFile` function.

4.300. ReadFloatVariableFromFile

Definition

```
float ReadFloatVariableFromFile()
```

Description

This function reads a floating point variable from the currently open file. Before reading information from the file, make sure that you have opened the desired file for reading with the `OpenFileForReading` function.

Return Value

This function returns a floating point value.

Example

```
fVar = 0.0

function Init()
   --Create a folder in Assets/Data/ path
   CreateFolder("Lev1")

   --Create and open file to write data
   OpenFileForWriting("Lev1/level1.bin")
   WriteFloatVariableToFile(2.0)
   CloseFile("Lev1/level1.bin")

   --Open File to load data
   OpenFileForReading("Lev1/level1.bin")
   fVar = ReadFloatVariableFromFile()
   CloseFile("Lev1/level1.bin")
end
```

First, using the `CreateFolder` function, we create a folder called `"Lev1"` in the Assets/ Data/ path. Then, using the `OpenFileForWriting` function, we open the `level1.bin` file located in the Assets/Data/`Lev1`/ path for writing. After writing the floating point value by the `WriteFloatVariableToFile` function, we close the file by the `CloseFile` function. Then, using the `OpenFileForReading` function, we open the `level1.bin` file located in the path Assets/Data/`Lev1`/ for reading and read a floating point variable from the `level1.bin` file with the `ReadFloatVariableFromFile()` function. In our example, value of `fVar` is `2.0` after reading it. Finally, we close the file by the `CloseFile` function.

4.301. ReadIntVariableFromFile

Definition

```
int ReadIntVariableFromFile()
```

Description

This function reads an integer variable from the currently open file. Before reading information from the file, make sure that you have opened the desired file for reading with the `OpenFileForReading` function.

Return Value

This function returns an integer value.

Example

```
iVar = 0

function Init()
   --Create a folder in Assets/Data/ path
   CreateFolder("Lev1")

   --Create and open file to write data
   OpenFileForWriting("Lev1/level1.bin")
   WriteIntVariableToFile(3)
   CloseFile("Lev1/level1.bin")

   --Open File to load data
   OpenFileForReading("Lev1/level1.bin")
   iVar = ReadIntVariableFromFile()
   CloseFile("Lev1/level1.bin")
end
```

First, using the `CreateFolder` function, we create a folder called `"Lev1"` in the Assets/Data/ path. Then, using the `OpenFileForWriting` function, we open the `level1.bin` file located in the Assets/Data/Lev1/ path for writing. After writing an integer value by the `WriteIntVariableToFile` function, we close the file by the `CloseFile` function.
Then, using the `OpenFileForReading` function, we open the `level1.bin` file located in the path Assets/Data/Lev1/ for reading and read an integer variable from the `level1.bin` file with the `ReadIntVariableFromFile()` function. In our example, value of `iVar` is 3 after reading it. Finally, we close the file by the `CloseFile` function.

4.302. ReadStringVariableFromFile

Definition

`string ReadStringVariableFromFile()`

Description

This function reads a string variable from the currently open file. Before reading information from the file, make sure that you have opened the desired file for reading with the `OpenFileForReading` function.

Return Value

This function returns a string.

Example

```
sVar = "init"

function Init()
   --Create a folder in Assets/Data/ path
   CreateFolder("Lev1")

   --Create and open file to write data
   OpenFileForWriting("Lev1/level1.bin")
   WriteStringVariableToFile("level1")
   CloseFile("Lev1/level1.bin")

   --Open File to load data
   OpenFileForReading("Lev1/level1.bin")
   sVar = ReadStringVariableFromFile()
   CloseFile("Lev1/level1.bin")
end
```

First, using the `CreateFolder` function, we create a folder called `"Lev1"` in the Assets/ Data/ path. Then, using the `OpenFileForWriting` function, we open the `level1.bin` file located in the Assets/Data/`Lev1`/ path for writing. After writing a string value by the `WriteStringVariableToFile` function, we close the file by the `CloseFile` function. Then, using the `OpenFileForReading` function, we open the `level1.bin` file located in the path Assets/Data/`Lev1`/ for reading and read a string variable from the `level1.bin` file with the `ReadStringVariableFromFile()` function. In our example, value of `sVar` is `"level1"` after reading it. Finally, we close the file by the `CloseFile` function.

4.303. RemoveCyclicAnimation

Definition

`RemoveCyclicAnimation(string prefabInstanceName, string animationClipName, float delayOut)`

Description

This function fades out cyclic animation **`animationClipName`** of prefab instance **`prefabInstanceName`** in a given amount of time. A cyclic animation is an animation that is repeating itself.

Parameters

`prefabInstanceName`
Specifies the name of the prefab instance. You can also use the name "this" for this parameter. In this case, "this" refers to the prefab instance that this script is attached to.

`animationClipName`
Specifies the name of the prefab instance animation. To view the name of the prefab instance animations, you can go to the Modify > Properties menu in the prefab editor, or select the name of the prefab instance from the Prefabs and GUIs section in the current VScene and press the Edit button.

`delayOut`
Specifies the time when the animation ***`animationClipName`*** is completely removed. This value must be 0.0 or higher.

Example 1

```
function Init()
    RemoveCyclicAnimation("1_animation_test_boy", "defaultClip", 1.0)
end

function Update()

end
```

In this example, the `RemoveCyclicAnimation` function fades out the `"defaultClip"` animation belonging to the prefab instance `"1_animation_test_boy"` in `1.0` seconds.

Example 2

```
--name of script is RemoveCyclicAnimation2.lua

animation = true

function Init()

end

function Update()
    if animation == true then
          RemoveCyclicAnimation("this", "run", 1.0)
```

```
            animation = false
    end
end
```

If, in the Prefab Editor, you attach `RemoveCyclicAnimation2.lua` script to a Prefab that has an animation clip `"run"`, then `"this"` parameter in the `RemoveCyclicAnimation` function will point to instances of that Prefab in current VScene. For example, if you have an Instance named *instance1_a* from a Prefab named *a* to which this script is attached, `"this"` in `RemoveCyclicAnimation` function refers to the name *instance1_a*.
In our example, the `RemoveCyclicAnimation` function fades out the `"run"` animation belonging to the current prefab instance (for example, *instance1_a*) in `1.0` seconds.

4.304. RemoveFile

Definition

`RemoveFile(string filePath)`

Description

This function removes `filePath` file located in the "Assets/Data/" path.

Parameters

`filePath`

File path in "Assets/Data/" folder.

Example

```
function OnTriggerEnter(otherActorName)
   --Create a folder in Assets/Data/ path
   CreateFolder("Lev1")

   --Create and open file to write data
   OpenFileForWriting("Lev1/level1.bin")
   WriteBoolVariableToFile(true)
   CloseFile("Lev1/level1.bin")
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)
    RemoveFile("Lev1/level1.bin")
end
```

Assume that the above script is attached to a trigger named "trigger1".
Whenever the main character or a prefab instance that has dynamic physics enters "trigger1" trigger, we call `CreateFolder` function to create a folder named `"lev1"` in the "Assets/Data/" path. Then, using the `OpenFileForWriting` function, we open the `level1.bin` file located in the Assets/Data/`Lev1`/ path for writing (If this file doesn't exist, `OpenFileForWriting` function will create the file as well). After writing the Boolean value by the `WriteBoolVariableToFile` function, we close the file by the `CloseFile` function.
When the main character or a prefab instance that has dynamic physics exits "trigger1" trigger, we remove the `"level1.bin"` file located in the Assets/Data/`Lev1`/ path.

4.305. RemoveFolder

Definition

`RemoveFolder(string folderPath)`

Description

This function removes the Assets/Data/`folderPath` along with all the folders and files inside it.

Parameters

`folderPath`

Folder path in "Assets/Data/" folder.

Example

```
function OnTriggerEnter(otherActorName)
   --Create a folder in Assets/Data/ path
   CreateFolder("Lev1")

   --Create a folder in Assets/Data/Lev1 path
  CreateFolder("Lev1/subLev1")

   --Create and open file to write data
   OpenFileForWriting("Lev1/subLev1/level1.bin")
   WriteBoolVariableToFile(true)
   CloseFile("Lev1/subLev1/level1.bin")
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)
    RemoveFolder("Lev1")
end
```

Assume that the above script is attached to a trigger named "trigger1".
Whenever the main character or a prefab instance that has dynamic physics enters "trigger1" trigger, we call `CreateFolder` function to create a folder named `"lev1"` in the "Assets/Data/" path. Then, we call `CreateFolder` function again to create a folder named `"subLev1"` in the "Assets/Data/`Lev1`" path. Then, using the `OpenFileForWriting` function, we open the `level1.bin` file located in the Assets/Data/`Lev1/subLev1`/ path for writing (If this file doesn't exist, `OpenFileForWriting` function will create the file as well). After writing the Boolean value by the `WriteBoolVariableToFile` function, we close the file by the `CloseFile` function.
When the main character or a prefab instance that has dynamic physics exits "trigger1" trigger, we remove the `"Lev1"` folder located in the Assets/Data/ path. This will remove `level1.bin` file and `subLev1` folder located in `"Lev1"` folder as well.

4.306. RemoveNonCyclicAnimation

Definition

`RemoveNonCyclicAnimation(string prefabInstanceName, string animationClipName)`

Description

This function removes non-cyclic animation **`animationClipName`** of prefab instance **`prefabInstanceName`**. Non-cycle animation is an animation that is executed only once instead of repeating.

Parameters

`prefabInstanceName`
Specifies the name of the prefab instance. You can also use the name "this" for this parameter. In this case, "this" refers to the prefab instance that this script is attached to.

`animationClipName`
Specifies the name of the prefab instance animation. To view the name of the prefab instance animations, you can go to the Modify > Properties menu in the prefab editor, or select the name of the prefab instance from the Prefabs and GUIs section in the current VScene and press the Edit button.

Example 1

```
function Init()
    RemoveNonCyclicAnimation("1_animation_test_boy", "defaultClip")
end

function Update()

end
```

In this example, the `RemoveNonCyclicAnimation` function removes the `"defaultClip"` animation belonging to the prefab instance `"1_animation_test_boy"`.

Example 2

```
--name of script is RemoveNonCyclicAnimation2.lua

animation = true

function Init()

end

function Update()
    if animation == true then
        RemoveNonCyclicAnimation("this", "run")
        animation = false
    end
end
```

If, in the Prefab Editor, you attach `RemoveNonCyclicAnimation2.lua` script to a Prefab that has an animation clip `"run"`, then `"this"` parameter in the `RemoveNonCyclicAnimation` function will point to instances of that Prefab in current VScene. For example, if you have an Instance named *instance1_a* from a Prefab named *a* to which this script is attached, `"this"` in `RemoveNonCyclicAnimation` function refers to the name *instance1_a*.
In our example, the `RemoveNonCyclicAnimation` function removes the `"run"` animation belonging to the current prefab instance (for example, *instance1_a*).

4.307. ResumeAllAnimationsOfPrefabInstances

Definition

`ResumeAllAnimationsOfPrefabInstances([optional] string exception_1, [optional] string exception_2,..., [optional] string exception_n)`

Description

This function resumes animations of all prefab instances except for the animations of prefab instances sent to the function.

Parameters

`[optional] string exception_1, [optional] string exception_2,..., [optional] string exception_n`

Specifies the name of prefab instances whose animation should not be resumed. If no name is passed to `ResumeAllAnimationsOfPrefabInstances` function, animations of all prefab instances will be resumed.

Example

```
function OnTriggerEnter(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          PauseAllAnimationsOfPrefabInstances()
    end
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)
    if otherActorName == nil then
          ResumeAllAnimationsOfPrefabInstances("2_animation_test_plane",
"2_animation_test_boy")
    end
end
```

Assume that the above script is attached to a trigger named trigger1. Whenever the main game character enters "trigger1", animations of all prefab instances will be paused. Whe the main character exits "trigger1", animations of all prefab instances except the animations of prefab instances `"2_animation_test_plane"` and `"2_animation_test_boy"` will be resumed.

4.308. ResumeAllUpdateEvents

Definition

```
ResumeAllUpdateEvents([optional] string exception_1, [optional] string
exception_2,..., [optional] string exception_n)
```

Description

This function resumes the script's `Update()` event of all game objects except the script's `Update()` event of objects passed to the function.

Parameters

`[optional] string exception_1, [optional] string exception_2,...,`
`[optional] string exception_n`

Specifies the name of the objects whose script's `Update()` event should not be resumed by this function. If no name is passed to the function, `Update()` events of all game object scripts will be resumed.

Example

```
function OnTriggerEnter(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          PauseAllUpdateEvents()
    end
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)
    if otherActorName == nil then
          ResumeAllUpdateEvents("water1", "sound1")
    end
end
```

Assume that the above script is attached to a trigger named trigger1. Also assume that `"water1"` and `"sound1"` in the example above are the name of water and sound objects in the VScene, respectively.
Whenever the main character enters "trigger1", script's `Update()` event of all game objects will be paused.
When the main character exits "trigger1", script's `Update()` event of all game objects except script's `Update()` event of `"water1"` and `"sound1"` objects will be resumed.

4.309. ResumeAnimationOfAllWaters

Definition

`ResumeAnimationOfAllWaters([optional] string exception_1, [optional] string exception_2,..., [optional] string exception_n)`

Description

This function resumes animation of all water objects except for the animation of water objects sent to the function.

Parameters

[optional] string exception_1, [optional] string exception_2,..., [optional] string exception_n

Specifies the name of waters whose animation should not be resumed. If no name is passed to `ResumeAnimationOfAllWaters` function, animation of all waters will be resumed.

Example

```
function OnTriggerEnter(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          PauseAnimationOfAllWaters()
    end
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)
    if otherActorName == nil then
          ResumeAnimationOfAllWaters("water2", "water3")
    end
end
```

Assume that the above script is attached to a trigger named trigger1. Also assume that `"water2"` and `"water3"` in the example above are the name of water objects in the VScene.
Whenever the main character enters "trigger1", animation of all waters will be paused.
When the main character exits "trigger1", animation of all waters except the animation of waters `"water2"` and `"water3"` will be resumed.

4.310. ResumeGame

Definition

```
ResumeGame()
```

Description

This function resumes the game.

Example

```
function OnSelectMouseLButtonDown()
    PauseGame()
end

function OnSelectMouseRButtonDown()
    ResumeGame()
end

function OnSelectMouseEnter()

end
```

Assume that the above script is attached to a button object named "button1". Whenever the user left clicks the button "button1", the game is paused. When the user right clicks the button "button1", the game resumes.

4.311. ResumeMainCharacterAnimations

Definition

```
ResumeMainCharacterAnimations()
```

Description

This function resumes all animations of the main character.

Example

```
function OnTriggerEnter(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          PauseMainCharacterAnimations()
    end
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          ResumeMainCharacterAnimations()
    end
end
```

Assume that the above script is attached to a trigger named "trigger1". When the main character enters "trigger1", all animations of the main character are paused. When the main character exits "trigger1", all animations of the main character are resumed.

4.312. ResumePhysics

Definition

```
ResumePhysics()
```

Description

This function resumes the physics.

Example

```
function OnSelectMouseLButtonDown()
    PausePhysics()
end

function OnSelectMouseRButtonDown()
    ResumePhysics()
end

function OnSelectMouseEnter()

end
```

Assume that the above script is attached to a button object named "button1". Whenever the user left clicks the button "button1", physics is paused. When the user right clicks the button "button1", physics resumes.

4.313. ResumePrefabInstanceAnimations

Definition

```
ResumePrefabInstanceAnimations(string prefabInstanceName)
```

Description

This function resumes all animations of the prefab instance `prefabInstanceName`. To view the name of prefab instances, open the VScene and click on the desired Prefab Instance in the "Prefabs and GUIs" section and press the Edit button. You can also access the names of prefab instances from the Script Utility section of the script editor (Tools > Script Editor > Tools > Script Utility). In the dialog that appears, you can view and copy the name of the prefab instance.

Parameters

prefabInstanceName

Specifies the name of the prefab instance. You can also use the name "this" for this parameter. In this case, "this" refers to the prefab instance that this script is attached to.

Example 1

```
function OnTriggerEnter(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          PausePrefabInstanceAnimations("1_animation_test_plane")
    end
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          ResumePrefabInstanceAnimations("1_animation_test_plane")
    end
end
```

Assume that the above script is attached to a trigger named trigger1. Whenever the main character enters "trigger1", all animations of prefab instance `"1_animation_test_plane"` will be paused. Whenever the main character exits "trigger1", all animations of prefab instance `"1_animation_test_plane"` will be resumed.

Example 2

```
--Name of script is ResumePrefabInstanceAnimations2.lua

pause_animation = true
time = 0.0

function Init()
    PausePrefabInstanceAnimations("this")
```

```
end

function Update()
    time = time + GetElapsedTime()
    if pause_animation and time >= 5.0 then
          ResumePrefabInstanceAnimations("this")
          pause_animation = false
    end
end
```

If, in the Prefab Editor, you attach ResumePrefabInstanceAnimations2.lua script to a Prefab, then "this" parameter in the ResumePrefabInstanceAnimations function will point to instances of that Prefab in current VScene. For example, if you have an Instance named *instance1_a* from a Prefab named *a* to which this script is attached, "this" in ResumePrefabInstanceAnimations function refers to the name *instance1_a*.
In this example, assume that the above script is attached to a prefab named *a* and we have an instance of it named *instance1_a*. first in the Init() event, we pause all animations of the current prefab instance named *instance_a*. Then, in the Update() event, after 5.0 seconds we resume all animations of the current prefab instance named *instance_a*.

4.314. ResumeUpdateEventOf3DSound

Definition

`ResumeUpdateEventOf3DSound(string 3DSoundName)`

Description

This function resumes the script's `Update()` event of 3D sound `3DSoundName`.

Parameters

3DSoundName

Specifies the name of the 3D sound. You can also use the name "this" for this parameter. In this case, "this" refers to the 3D sound that this script is attached to.

Example 1

```
function OnTriggerEnter(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
        PauseUpdateEventOf3DSound("river1")
    end
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
        ResumeUpdateEventOf3DSound("river1")
    end
end
```

Assume that the above script is attached to a trigger named trigger1. When the main character enters "trigger1", script's `Update()` event of 3D sound `"river1"` will be paused. When the main character exits "trigger1", script's `Update()` event of 3D sound `"river1"` will be resumed.

Example 2

```
--Name of script is ResumeUpdateEventOf3DSound2.lua

function Init()
    PauseUpdateEventOf3DSound("this")

    ResumeUpdateEventOf3DSound("this")
end

function Update()
    PrintConsole("\nUpdate")
end
```

Assume that the above script named `ResumeUpdateEventOf3DSound2.lua` is attached to a 3D sound object named "sound1". In this case, string `"this"` in the `ResumeUpdateEventOf3DSound` function will be equal to "sound1". In our example, we use `PauseUpdateEventOf3DSound` to pause the script's `Update()` event of current 3D sound, which is "sound1". Then we use `ResumeUpdateEventOf3DSound` to resume the script's `Update()` event of current 3D sound, which is "sound1".

4.315. ResumeUpdateEventOfAll3DSounds

Definition

```
ResumeUpdateEventOfAll3DSounds([optional] string exception_1, [optional]
string exception_2,..., [optional] string exception_n)
```

Description

This function resumes the script's `Update()` event of all 3D sounds except the script's `Update()` event of 3D sounds passed to the function.

Parameters

[optional] string exception_1, [optional] string exception_2,...,
[optional] string exception_n

Specifies the name of the 3D sounds whose script's `Update()` event should not be resumed by this function. If no name is passed to the function, `Update()` events of all 3D sound scripts will be resumed.

Example

```
function OnTriggerEnter(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          PauseUpdateEventOfAll3DSounds()
    end
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          ResumeUpdateEventOfAll3DSounds("river2", "river3")
    end
end
```

Assume that the above script is attached to a trigger named trigger1. Also assume that `"river2"` and `"river3"` in the example above are the name of 3D sound objects.
When the main character enters "trigger1", script's `Update()` event of all 3D sounds will be paused.
When the main character exits "trigger1", script's `Update()` event of all 3D sounds except script's `Update()` event of `"river2"` and `"river3"` 3D sounds will be resumed.

4.316. ResumeUpdateEventOfAllAmbientSounds

Definition

```
ResumeUpdateEventOfAllAmbientSounds([optional] string exception_1,
[optional] string exception_2,..., [optional] string exception_n)
```

Description

This function resumes the script's `Update()` event of all ambient sounds except the script's `Update()` event of ambient sounds passed to the function.

Parameters

[optional] string exception_1, [optional] string exception_2,..., [optional] string exception_n

Specifies the name of the ambient sounds whose script's `Update()` event should not be resumed by this function. If no name is passed to the function, `Update()` events of all ambient sound scripts will be resumed.

Example

```
function OnTriggerEnter(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          PauseUpdateEventOfAllAmbientSounds()
    end
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          ResumeUpdateEventOfAllAmbientSounds("ambient2", "ambient3")
    end
end
```

Assume that the above script is attached to a trigger named trigger1. Also assume that `"ambient2"` and `"ambient3"` in the example above are the name of ambient sound objects.
When the main character enters "trigger1", script's `Update()` event of all ambient sounds will be paused.
When the main character exits "trigger1", script's `Update()` event of all ambient sounds except script's `Update()` event of `"ambient2"` and `"ambient3"` ambient sounds will be resumed.

4.317. ResumeUpdateEventOfAllEngineCameras

Definition

```
ResumeUpdateEventOfAllEngineCameras([optional] string exception_1,
[optional] string exception_2,..., [optional] string exception_n)
```

Description

This function resumes the script's `Update()` event of all engine cameras except the script's `Update()` event of engine cameras passed to the function.

Parameters

`[optional] string exception_1, [optional] string exception_2,...,`
`[optional] string exception_n`

Specifies the name of the engine cameras whose script's `Update()` event should not be resumed by this function. If no name is passed to the function, `Update()` events of all engine camera scripts will be resumed.

Example

```lua
function OnTriggerEnter(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
         PauseUpdateEventOfAllEngineCameras()
    end
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
         ResumeUpdateEventOfAllEngineCameras("camera2", "camera3")
    end
end
```

Assume that the above script is attached to a trigger named "trigger1". Also assume that `"camera2"` and `"camera3"` in the example above are the name of engine camera objects.
Whenever the main character enters "trigger1", script's `Update()` event of all engine cameras will be paused.
Whenever the main character exits "trigger1", script's `Update()` event of all engine cameras except script's `Update()` event of `"camera2"` and `"camera3"` engine cameras will be resumed.

4.318. ResumeUpdateEventOfAllLights

Definition

```
ResumeUpdateEventOfAllLights([optional] string exception_1, [optional]
string exception_2,..., [optional] string exception_n)
```

Description

This function resumes the script's `Update()` event of all lights except the script's `Update()` event of lights passed to the function.

Parameters

`[optional] string exception_1, [optional] string exception_2,..., [optional] string exception_n`

Specifies the name of the lights whose script's `Update()` event should not be resumed by this function. If no name is passed to the function, `Update()` events of all light scripts will be resumed.

Example

```
function OnTriggerEnter(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          PauseUpdateEventOfAllLights()
    end
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          ResumeUpdateEventOfAllLights("light2", "light3")
    end
end
```

Assume that the above script is attached to a trigger named trigger1. Also assume that `"light2"` and `"light3"` in the example above are the name of light objects.
Whenever the main character enters "trigger1", script's `Update()` event of all lights will be paused.
Whenever the main character exits "trigger1", script's `Update()` event of all lights except script's `Update()` event of `"light2"` and `"light3"` lights will be resumed.

4.319. ResumeUpdateEventOfAllPrefabInstances

Definition

```
ResumeUpdateEventOfAllPrefabInstances([optional] string exception_1,
[optional] string exception_2,..., [optional] string exception_n)
```

Description

This function resumes the script's `Update()` event of all prefab instances except the script's `Update()` event of prefab instances passed to the function.

Parameters

[optional] string exception_1, [optional] string exception_2,..., [optional] string exception_n

Specifies the name of the prefab instances whose script's `Update()` event should not be resumed by this function. If no name is passed to the function, `Update()` events of all prefab instance scripts will be resumed.

Example

```
function OnTriggerEnter(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          PauseUpdateEventOfAllPrefabInstances()
    end
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          ResumeUpdateEventOfAllPrefabInstances("1_animation_test_boy",
"1_animation_test_plane")
    end
end
```

Assume that the above script is attached to a trigger named "trigger1". Also assume that `"1_animation_test_boy"` and `"1_animation_test_plane"` in the example above are the name of prefab instances.

Whenever the main character enters "trigger1", script's `Update()` event of all prefab instances will be paused.

Whenever the main character exits "trigger1", script's `Update()` event of all prefab instances except script's `Update()` event of `"1_animation_test_boy"` and `"1_animation_test_plane"` prefab instances will be resumed.

4.320. ResumeUpdateEventOfAllWaters

Definition

`ResumeUpdateEventOfAllWaters([optional] string exception_1, [optional] string exception_2,..., [optional] string exception_n)`

Description

This function resumes the script's `Update()` event of all waters except the script's `Update()` event of waters passed to the function.

Parameters

`[optional] string exception_1, [optional] string exception_2,..., [optional] string exception_n`

Specifies the name of the waters whose script's `Update()` event should not be resumed by this function. If no name is passed to the function, `Update()` events of all water scripts will be resumed.

Example

```
function OnTriggerEnter(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          PauseUpdateEventOfAllWaters()
    end
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          ResumeUpdateEventOfAllWaters("water2", "water3")
    end
end
```

Assume that the above script is attached to a trigger named "trigger1". Also assume that `"water2"` and `"water3"` in the example above are the name of water objects.
Whenever the main character enters "trigger1", script's `Update()` event of all waters will be paused.
Whenever the main character exits "trigger1", script's `Update()` event of all waters except script's `Update()` event of `"water2"` and `"water3"` waters will be resumed.

4.321. ResumeUpdateEventOfAmbientSound

Definition

```
ResumeUpdateEventOfAmbientSound(string ambientSoundName)
```

Description

This function resumes the script's `Update()` event of ambient sound `ambientSoundName`.

Parameters

ambientSoundName

Specifies the name of the ambient sound. You can also use the name "this" for this parameter. In this case, "this" refers to the ambient sound that this script is attached to.

Example 1

```
function OnTriggerEnter(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
        PauseUpdateEventOfAmbientSound("ambient1")
    end
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
        ResumeUpdateEventOfAmbientSound("ambient1")
    end
end
```

Assume that the above script is attached to a trigger named "trigger1". When the main character enters "trigger1", script's `Update()` event of ambient sound `"ambient1"` will be paused. When the main character exits "trigger1", script's `Update()` event of ambient sound `"ambient1"` will be resumed.

Example 2

```
--Name of script is ResumeUpdateEventOfAmbientSound2.lua

function Init()
    PauseUpdateEventOfAmbientSound("this")

    ResumeUpdateEventOfAmbientSound("this")
end

function Update()
    PrintConsole("\nUpdate")
end
```

Assume that the above script named `ResumeUpdateEventOfAmbientSound2.lua` is attached to an ambient sound object named "sound1". In this case, string `"this"` in the `ResumeUpdateEventOfAmbientSound` function will be equal to "sound1". In our example, we use `PauseUpdateEventOfAmbientSound` to pause the script's `Update()` event of current ambient sound, which is "sound1". Then we use `ResumeUpdateEventOfAmbientSound` to resume the script's `Update()` event of current ambient sound, which is "sound1".

4.322. ResumeUpdateEventOfEngineCamera

Definition

`ResumeUpdateEventOfEngineCamera(string engineCameraName)`

Description

This function resumes the script's `Update()` event of engine camera `engineCameraName`.

Parameters

engineCameraName

Specifies the name of the engine camera. You can also use the name "this" for this parameter. In this case, "this" refers to the engine camera that this script is attached to.

Example 1

```
function OnTriggerEnter(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
        PauseUpdateEventOfEngineCamera("camera1")
    end
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
        ResumeUpdateEventOfEngineCamera("camera1")
    end
end
```

Assume that the above script is attached to a trigger named "trigger1". When the main character enters "trigger1", script's `Update()` event of engine camera `"camera1"` will be paused. When the main character exits "trigger1", script's `Update()` event of engine camera `"camera1"` will be resumed.

Example 2

```
--Name of script is ResumeUpdateEventOfEngineCamera2.lua

function Init()
    PauseUpdateEventOfEngineCamera("this")

    ResumeUpdateEventOfEngineCamera("this")
end

function Update()
    PrintConsole("\nUpdate")
end
```

Assume that the above script named `ResumeUpdateEventOfEngineCamera2.lua` is attached to an engine camera object named "camera1". In this case, string `"this"` in the `ResumeUpdateEventOfEngineCamera` function will be equal to "camera1". In our example, we use `PauseUpdateEventOfEngineCamera` to pause the script's `Update()` event of current engine camera, which is "camera1". Then we use `ResumeUpdateEventOfEngineCamera` to resume the script's `Update()` event of current engine camera, which is "camera1".

4.323. ResumeUpdateEventOfLight

Definition

`ResumeUpdateEventOfLight(string lightName)`

Description

This function resumes the script's `Update()` event of light `lightName`.

Parameters

lightName

Specifies the name of the light. You can also use the name "this" for this parameter. In this case, "this" refers to the light that this script is attached to.

Example 1

```
function OnTriggerEnter(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          PauseUpdateEventOfLight("light1")
    end
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          ResumeUpdateEventOfLight("light1")
    end
end
```

Assume that the above script is attached to a trigger named "trigger1". When the main character enters "trigger1", script's `Update()` event of light `"light1"` will be paused. When the main character exits "trigger1", script's `Update()` event of light `"light1"` will be resumed.

Example 2

```
--Name of script is ResumeUpdateEventOfLight2.lua

function Init()
    PauseUpdateEventOfLight("this")

    ResumeUpdateEventOfLight("this")
end

function Update()
    PrintConsole("\nUpdate")
end
```

Assume that the above script named `ResumeUpdateEventOfLight2.lua` is attached to a light object named "light1". In this case, string `"this"` in the `ResumeUpdateEventOfLight` function will be equal to "light1". In our example, we use `PauseUpdateEventOfLight` to pause the script's `Update()` event of current light, which is "light1". Then we use `ResumeUpdateEventOfLight` to resume the script's `Update()` event of current light, which is "light1".

4.324. ResumeUpdateEventOfMainCharacter

Definition

`ResumeUpdateEventOfMainCharacter()`

Description

This function resumes the script's `Update()` event of main character.

Example

```
function OnTriggerEnter(otherActorName)
    --nil means main character controller
     if otherActorName == nil then
          PauseUpdateEventOfMainCharacter()
    end
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          ResumeUpdateEventOfMainCharacter()
    end
end
```

Assume that the above script is attached to a trigger named "trigger1". Whenever the main character enters "trigger1", script's `Update()` event of main character will be paused. Whenever the main character exits "trigger1", script's `Update()` event of main character will be resumed.

4.325. ResumeUpdateEventOfPrefabInstance

Definition

`ResumeUpdateEventOfPrefabInstance(string prefabInstanceName)`

Description

This function resumes the script's `Update()` event of prefab instance `prefabInstanceName`.

Parameters

`prefabInstanceName`

Specifies the name of the prefab instance. You can also use the name "this" for this parameter. In this case, "this" refers to the prefab instance name that this script is attached to.

Example 1

```
function OnTriggerEnter(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
        PauseUpdateEventOfPrefabInstance("1_animation_test_plane")
    end
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
        ResumeUpdateEventOfPrefabInstance("1_animation_test_plane")
    end
end
```

Assume that the above script is attached to a trigger named "trigger1". Whenever the main character enters "trigger1", script's `Update()` event of prefab instance `"1_animation_test_plane"` will be paused. Whenever the main character exits "trigger1", script's `Update()` event of prefab instance `"1_animation_test_plane"` will be resumed.

Example 2

```
--Name of script is ResumeUpdateEventOfPrefabInstance2.lua

function Init()
    PauseUpdateEventOfPrefabInstance("this")
    ResumeUpdateEventOfPrefabInstance("this")
end

function Update()
    PrintConsole("\nUpdate")
end
```

If, in the Prefab Editor, you attach ResumeUpdateEventOfPrefabInstance2.lua script to a Prefab, then "this" parameter in the ResumeUpdateEventOfPrefabInstance function will point to instances of that Prefab in current VScene. For example, if you have an Instance named *instance1_a* from a Prefab named *a* to which this script is attached, "this" in ResumeUpdateEventOfPrefabInstance function refers to the name *instance1_a*. In this example, we use PauseUpdateEventOfPrefabInstance to pause the script's Update() event of current prefab instance (for example, *instance1_a*). Then we use ResumeUpdateEventOfPrefabInstance to resume the script's Update() event of current prefab instance (for example, *instance1_a*).

4.326. ResumeUpdateEventOfSky

Definition

`ResumeUpdateEventOfSky()`

Description

This function resumes the script's `Update()` event of sky object.

Example

```
function OnTriggerEnter(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          PauseUpdateEventOfSky()
    end
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          ResumeUpdateEventOfSky()
    end
end
```

Assume that the above script is attached to a trigger named "trigger1". Whenever the main character enters "trigger1", script's `Update()` event of sky object will be paused. Whenever the main character exits "trigger1", script's `Update()` event of sky object will be resumed.

4.327. ResumeUpdateEventOfTerrain

Definition

`ResumeUpdateEventOfTerrain()`

Description

This function resumes the script's `Update()` event of terrain object.

Example

```
function OnTriggerEnter(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          PauseUpdateEventOfTerrain()
    end
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          ResumeUpdateEventOfTerrain()
    end
end
```

Assume that the above script is attached to a trigger named "trigger1". Whenever the main character enters "trigger1", script's `Update()` event of terrain object will be paused. Whenever the main character exits "trigger1", script's `Update()` event of terrain object will be resumed.

4.328. ResumeUpdateEventOfVSceneScript

Definition

```
ResumeUpdateEventOfVSceneScript()
```

Description

This function resumes the script's `Update()` event of VScene Script object.

Example

```
function OnTriggerEnter(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
        PauseUpdateEventOfVSceneScript()
    end
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
        ResumeUpdateEventOfVSceneScript()
    end
end
```

Assume that the above script is attached to a trigger named "trigger1". Whenever the main character enters "trigger1", script's `Update()` event of VScene Script object will be paused. Whenever the main character exits "trigger1", script's `Update()` event of VScene Script object will be resumed.

4.329. ResumeUpdateEventOfWater

Definition

`ResumeUpdateEventOfWater(string waterName)`

Description

This function resumes the script's `Update()` event of water `waterName`.

Parameters

`waterName`

Specifies the name of the water. You can also use the name "this" for this parameter. In this case, "this" refers to the water that this script is attached to.

Example 1

```
function OnTriggerEnter(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
        PauseUpdateEventOfWater("water1")
    end
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
        ResumeUpdateEventOfWater("water1")
    end
end
```

Assume that the above script is attached to a trigger named "trigger1". When the main character enters "trigger1", script's `Update()` event of water `"water1"` will be paused. When the main character exits "trigger1", script's `Update()` event of water `"water1"` will be resumed.

Example 2

```
--Name of script is ResumeUpdateEventOfWater2.lua

function Init()
    PauseUpdateEventOfWater("this")

    ResumeUpdateEventOfWater("this")
end

function Update()
    PrintConsole("\nUpdate")
end
```

Assume that the above script named `ResumeUpdateEventOfWater2.lua` is attached to a water object named "water1". In this case, string `"this"` in the `ResumeUpdateEventOfWater` function will be equal to "water1". In our example, we use `PauseUpdateEventOfWater` to pause the script's `Update()` event of current water, which is "water1". Then we use `ResumeUpdateEventOfWater` to resume the script's `Update()` event of current water, which is "water1".

4.330. ResumeWaterAnimation

Definition

```
ResumeWaterAnimation(string waterObjectName)
```

Description

This function resumes animation of water `waterObjectName`.

Parameters

`waterObjectName`

Specifies the name of the water. You can also use the name "this" for this parameter. In this case, "this" refers to the water that this script is attached to.

Example 1

```
function OnTriggerEnter(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          PauseWaterAnimation("water1")
    end
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          ResumeWaterAnimation("water1")
    end
end
```

Assume that the above script is attached to a trigger named "trigger1". Whenever the main character enters "trigger1", animation of water `"water1"` will be paused. Whenever the main character exits "trigger1", animation of water `"water1"` will be resumed.

Example 2

```
--Name of script is ResumeWaterAnimation2.lua

function Init()
    PauseWaterAnimation("this")

    ResumeWaterAnimation("this")
end

function Update()

end
```

Assume that the above script named `ResumeWaterAnimation2.lua` is attached to a water object named "water1". In this case, string `"this"` in the `ResumeWaterAnimation` function will be equal to "water1". In our example, we use `PauseWaterAnimation` function to pause animation of current water, which is "water1". Then we use `ResumeWaterAnimation` function to resume animation of current water, which is "water1".

4.331. ReverseExecuteNonCyclicAnimation

Definition

```
ReverseExecuteNonCyclicAnimation(string prefabInstanceName, string
animationClipName)
```

Description

This function plays the non cyclic animation **animationClipName** belonging to prefab instance **prefabInstanceName** in reverse. Non-cycle animation is an animation that is executed only once instead of repeating.

Parameters

prefabInstanceName

Specifies the name of the prefab instance. You can also use the name "this" for this parameter. In this case, "this" refers to the prefab instance that this script is attached to.

animationClipName

Specifies the name of the prefab instance animation. To view the name of the prefab instance animations, you can go to the Modify > Properties menu in the prefab editor, or select the name of the prefab instance from the Prefabs and GUIs section in the current VScene and press the Edit button.

Example 1

```
function OnTriggerEnter(otherActorName)
    ExecuteNonCyclicAnimation("1_animation_test_boy", "defaultClip", 0.5, 0.7, 1.0,
true)
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)
    ReverseExecuteNonCyclicAnimation("1_animation_test_boy", "defaultClip")
end
```

Assume that the above script is attached to a trigger object named "trigger1". When the main character or a prefab instance that has dynamic physics enters "trigger1", we execute the `defaultClip` animation of prefab instance `1_animation_test_boy` once. When the main character or a prefab instance that has dynamic physics exits "trigger1", we play the `defaultClip` animation of prefab instance `1_animation_test_boy` in reverse.

Example 2

```
--name of script is ReverseExecuteNonCyclicAnimation2.lua

animation = true
animation_time = 0.0
time = 0.0

function Init()
```

```
    ExecuteNonCyclicAnimation("this", "defaultClip", 0.5, 0.7, 1.0, false)

    animation_time = GetAnimationClipDurationOfPrefabInstance("this", "defaultClip")
end

function Update()
    time = time + GetElapsedTime()

    if animation == true and time > animation_time / 2.0 then
        ReverseExecuteNonCyclicAnimation("this", "defaultClip")
        animation = false
    end
end
```

If, in the Prefab Editor, you attach ReverseExecuteNonCyclicAnimation2.lua script to a Prefab that has an animation clip "defaultClip", then "this" parameter in the ExecuteNonCyclicAnimation function will point to instances of that Prefab in current VScene. For example, if you have an Instance named *instance1_a* from a Prefab named *a* to which this script is attached, "this" in ReverseExecuteNonCyclicAnimation function refers to the name *instance1_a*.
First in the Init() event, we use ExecuteNonCyclicAnimation to execute the defaultClip animation belonging to current prefab instance (for example, *instance1_a*). Then, using the function GetAnimationClipDurationOfPrefabInstance, we determine the duration of the defaultClip animation of the current prefab instance.
In the Update() event, we first calculate the elapsed time. Then, if the **animation** variable is equal to true (its initial value is true) and the elapsed time exceeds half of the defaultClip animation of the current prefab instance, we play the defaultClip animation of the current prefab instance in the reverse using the ReverseExecuteNonCyclicAnimation function. Finally, we set the animation variable to false so that the ReverseExecuteNonCyclicAnimation function is not executed again.

4.332. RotatePrefabInstance

Definition

`RotatePrefabInstance(string prefabInstanceName, float XRotationAngle, float YRotationAngle, float ZRotationAngle)`

Description

This function rotates the *transformable* prefab instance **`prefabInstanceName`** around the X, Y, and Z axes. For this function to work, in prefab mode, through the Modify > Prefab Properties menu, make sure the *transformable* option is checked for the desired prefab.

Parameters

`prefabInstanceName`
Specifies the name of the prefab instance. You can also use the name "this" for this parameter. In this case, "this" refers to the prefab instance that this script is attached to.

`XRotationAngle, YRotationAngle, ZRotationAngle`
Specifies the rotation of the prefab instance ***`prefabInstanceName`*** around the X, Y, and Z axes.

Example 1

```
rotateX = 0.0
rotateY = 0.0
rotateZ = 0.0

function Init()

end

function Update()
    rotateX = rotateX + GetElapsedTime()
    rotateY = rotateY + (GetElapsedTime() * 2.0)
    rotateZ = rotateZ + (GetElapsedTime() * 3.0)

    if rotateX > 360.0 then rotateX = rotateX - 360.0 end
    if rotateY > 360.0 then rotateY = rotateY - 360.0 end
    if rotateZ > 360.0 then rotateZ = rotateZ - 360.0 end

    RotatePrefabInstance("1_VandaEngine17-SamplePack1_well", rotateX, rotateY, rotateZ)
end
```

First, we increase the value of **`rotateX`**, **`rotateY`** and **`rotateZ`** variables according to time and make sure that their value is not more than `360.0` degrees. Then, using these three values and the `RotatePrefabInstance` function, we rotate the prefab instance `1_VandaEngine17-SamplePack1_well` around the X, Y and Z axes. It should be noted that the Transformable feature of prefab instance `1_VandaEngine17-SamplePack1_well` must be enabled for the function `RotatePrefabInstance` to work.

Example 2

```
--Name of script is RotatePrefabInstance2.lua
rotateX = 0.0
```

```
rotateY = 0.0
rotateZ = 0.0

function Init()

end

function Update()
    rotateX = rotateX + GetElapsedTime()
    rotateY = rotateY + (GetElapsedTime() * 2.0)
    rotateZ = rotateZ + (GetElapsedTime() * 3.0)

    if rotateX > 360.0 then rotateX = rotateX - 360.0 end
    if rotateY > 360.0 then rotateY = rotateY - 360.0 end
    if rotateZ > 360.0 then rotateZ = rotateZ - 360.0 end

    RotatePrefabInstance("this", rotateX, rotateY, rotateZ)
end
```

If, in the Prefab Editor, you attach `RotatePrefabInstance2.lua` script to a Prefab, then `"this"` parameter in the `RotatePrefabInstance` function will point to instances of that Prefab in current VScene. For example, if you have an Instance named *instance1_a* from a Prefab named *a* to which this script is attached, `"this"` in `RotatePrefabInstance` function refers to the name *instance1_a*.

First, we increase the value of **rotateX**, **rotateY** and **rotateZ** variables according to time and make sure that their value is not more than `360.0` degrees. Then, using these three values and the `RotatePrefabInstance` function, we rotate the current prefab instance (for example, *instance1_a*) around the X, Y and Z axes. It should be noted that the Transformable feature of current prefab instance must be enabled for the function `RotatePrefabInstance` to work.

4.333. SaveGeneralProperties

Definition

`SaveGeneralProperties()`

Description

This function saves all the initial dialog information of the game at runtime in the "Assets/config/conf_win32.dat" file. Whenever you run the game, the dialog information at the beginning of the game is loaded based on the information in the "conf_win32.dat" file. It should be noted that whenever you press the Play button of the initial dialog of the game, Vanda Engine automatically saves the dialog information in the "conf_win32.dat" file. The `SaveGeneralProperties()` function is only useful if you want to save this information while the game is running.
The following properties are saved by `SaveGeneralProperties()` function:

```
CBool m_useCurrentResolution; (current screen resolution)
CInt m_width; (current screen width)
CInt m_height; (current screen height)
CInt m_numSamples; (current multisampling number)
CInt m_anisotropy; (current texture anisotropic filtering number)
CBool m_showStartupDialog; (show dialog at statup?)
CBool m_disableVSync; (Is VSync disabled?)
CBool m_enableWaterReflection; (Is general water reflection enabled?)
CBool m_fullScreen; (Is full screen enabled?)
```

Example

```
function Init()
    SaveGeneralProperties()
end

function Update()

end
```

4.334. ScaleGUIButton

Definition

`ScaleGUIButton(string GUIName, string buttonName, double scaleValue)`

Description

This function sets the scale of the button **`buttonName`** that belongs to the GUI **`GUIName`**. In this case, the length and width of the button **`buttonName`** are multiplied by the **`scaleValue`**. A value of 1.0 for **`scaleValue`** will be equivalent to the initial size of the button.

Parameters

`GUIName`
Specifies the GUI name.

`buttonName`
Specifies the button name that belongs to the GUI **`GUIName`**.

`scaleValue`
Specifies the scale of the button **`buttonName`** that belongs to the GUI **`GUIName`**. This value must be equal to or greater than 1.0.

Example

```
function OnTriggerEnter(otherActorName)
    ScaleGUIButton("gui_SampleGUI17_MainMenu", "PlayGame", 2.0)
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)

end
```

Assume that the above script is attached to a trigger named "trigger1". Whenever the main character or a prefab instance that has dynamic physics is entered into this trigger, `ScaleGUIButton` function sets the scale of the button `"PlayGame"` that belongs to GUI `"gui_SampleGUI17_MainMenu"` to `2.0`. In this case, the length and width of the button `"PlayGame"` are multiplied by `2.0`.

4.335. ScaleGUIImage

Definition

`ScaleGUIImage(string GUIName, string imageName, double scaleValue)`

Description

This function sets the scale of the image **`imageName`** that belongs to the GUI **`GUIName`**. In this case, the length and width of the image **`imageName`** are multiplied by the **`scaleValue`**. A value of 1.0 for **`scaleValue`** will be equivalent to the initial size of the image.

Parameters

`GUIName`
Specifies the GUI name.

`imageName`
Specifies the image name that belongs to the GUI **`GUIName`**.

`scaleValue`
Specifies the scale of the image **`imageName`** that belongs to the GUI **`GUIName`**. This value must be equal to or greater than 1.0.

Example

```
function OnTriggerEnter(otherActorName)
    ScaleGUIImage("gui_SampleGUI17_MainMenuAbout", "backgroundImg", 1.5)
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)

end
```

Assume that the above script is attached to a trigger named "trigger1". Whenever the main character or a prefab instance that has dynamic physics is entered into this trigger, `ScaleGUIImage` function sets the scale of the image `"backgroundImg"` that belongs to GUI `"gui_SampleGUI17_MainMenuAbout"` to `1.5`. In this case, the length and width of the image `"backgroundImg"` are multiplied by `1.5`.

4.336. ScalePrefabInstance

Definition

`ScalePrefabInstance(string prefabInstanceName, float XScale, float YScale, float ZScale)`

Description

This function scales the *transformable* prefab instance **prefabInstanceName** in the X, Y, and Z directions. For this function to work, in prefab mode, through the Modify > Prefab Properties menu, make sure the *transformable* option is checked for the desired prefab.

Parameters

prefabInstanceName

Specifies the name of the prefab instance. You can also use the name "this" for this parameter. In this case, "this" refers to the prefab instance that this script is attached to.

XScale, YScale, ZScale

Specifies the scale of the prefab instance *prefabInstanceName* in the X, Y, and Z directions.

Example 1

```
scaleX = 1.0
scaleY = 1.0
scaleZ = 1.0

function Init()

end

function Update()
    scaleX = scaleX + (GetElapsedTime() * 0.1)
    scaleY = scaleY + (GetElapsedTime() * 0.2)
    scaleZ = scaleZ + (GetElapsedTime() * 0.3)

    if scaleX > 5.0 then scaleX = 1.0 end
    if scaleY > 5.0 then scaleY = 1.0 end
    if scaleZ > 5.0 then scaleZ = 1.0 end

    ScalePrefabInstance("1_VandaEngine17-SamplePack1_well", scaleX, scaleY, scaleZ)
end
```

First, we increase the value of `scaleX`, `scaleY` and `scaleZ` variables according to time and make sure that their value is not more than `5.0` units. Then, using these three values and the `ScalePrefabInstance` function, we scale the prefab instance `1_VandaEngine17-SamplePack1_well` in the X, Y and Z directions. It should be noted that the Transformable property of prefab instance `1_VandaEngine17-SamplePack1_well` must be enabled for the function `ScalePrefabInstance` to work.

Example 2

```
--Name of script is ScalePrefabInstance2.lua
```

```
scaleX = 1.0
scaleY = 1.0
scaleZ = 1.0

function Init()

end

function Update()
    scaleX = scaleX + (GetElapsedTime() * 0.1)
    scaleY = scaleY + (GetElapsedTime() * 0.2)
    scaleZ = scaleZ + (GetElapsedTime() * 0.3)

    if scaleX > 5.0 then scaleX = 1.0 end
    if scaleY > 5.0 then scaleY = 1.0 end
    if scaleZ > 5.0 then scaleZ = 1.0 end

    ScalePrefabInstance("this", scaleX, scaleY, scaleZ)
end
```

If, in the Prefab Editor, you attach ScalePrefabInstance2.lua script to a Prefab, then "this" parameter in the ScalePrefabInstance function will point to instances of that Prefab in current VScene. For example, if you have an Instance named *instance1_a* from a Prefab named *a* to which this script is attached, "this" in ScalePrefabInstance function refers to the name *instance1_a*.

First, we increase the value of scaleX, scaleY and scaleZ variables according to time and make sure that their value is not more than 5.0 units. Then, using these three values and the ScalePrefabInstance function, we scale the current prefab instance (for example, *instance1_a*) in the X, Y and Z directions. It should be noted that the Transformable property of current prefab instance must be enabled for the function ScalePrefabInstance to work.

4.337. SelectPrefabInstances

Definition

`SelectPrefabInstances(double mousePositionX, double mousePositionY, double selectionWidthSize, double selectionHeightSize)`

Description

This function selects *selectable* prefab instances. For this function to work, in prefab mode, through the Modify > Prefab Properties menu, make sure the *Selectable* option is checked for the desired prefab.

Parameters

`mousePositionX, mousePositionY`
Specify the center of a selection region in window coordinates.

`selectionWidthSize, selectionHeightSize`
Specify the width and height, respectively, of the selection region in window coordinates.

Example

```
function Init()
    LoadResource("images", "cursor.dds")
    ShowCursorIcon("images_cursor.dds", 5.0)
end

function Update()
    if IsKeyDown("0") then
        SelectPrefabInstances(GetCursorX(), GetCursorY(), 20.0, 20.0)
    end
end
```

First, we load and display `cursor.dds` resource image (In order for LoadResource function to load the desired resource, you must first add it through the Add Resource to Current Project dialog (File > Project > Add/Remove Resource to/from Current Project). Then, in the `Update()` event, in the `SelectPrefabInstances` function, we set the center of the selection to the mouse position using `GetCursorX()` and `GetCursorY()` functions and set the length and width of the selection to `20.0`. Whenever the user left-clicks, the `SelectPrefabInstances` function is called. If the prefab instance is in the selection region, it is selected and its `Onselect()` event is called.

4.338. Set3DSoundScriptBoolVariable

Definition

`Set3DSoundScriptBoolVariable(string 3DSoundName, string variable, bool value)`

Description

This function sets the value of the Boolean **`variable`** defined in the script attached to the **`3DSoundName`** 3D sound object to **`value`**.

Parameters

`3DSoundName`
Specifies the name of the 3D sound object.

`variable`
Specifies the name of the Boolean variable defined in the script attached to the **`3DSoundName`** 3D sound.

`value`
Specifies the Boolean value for the variable **`variable`**.

Example

```
--script name is Set3DSoundScriptBoolVariable.lua attached a to game object such as water

function Init()
    Set3DSoundScriptBoolVariable("sound1", "a", true)
end

function Update()

end
```

Assuming that the variable `"a"` is defined in the script attached to the 3D sound object `"sound1"`, `Set3DSoundScriptBoolVariable` function sets the `"a"` variable to *true*.

4.339. Set3DSoundScriptDoubleVariable

Definition

`Set3DSoundScriptDoubleVariable(string 3DSoundName, string variable, double value)`

Description

This function sets the value of the Double `variable` defined in the script attached to the `3DSoundName` 3D sound object to `value`.

Parameters

3DSoundName
Specifies the name of the 3D sound object.

variable
Specifies the name of the Double variable defined in the script attached to the `3DSoundName` 3D sound.

value
Specifies the Double value for the variable `variable`.

Example

```
--script name is Set3DSoundScriptDoubleVariable.lua attached a to game object such as
water

function Init()
    Set3DSoundScriptDoubleVariable("sound1", "a", 1.0)
end

function Update()

end
```

Assuming that the variable `"a"` is defined in the script attached to the 3D sound object `"sound1"`, `Set3DSoundScriptDoubleVariable` function sets the `"a"` variable to `1.0`.

4.340. Set3DSoundScriptIntVariable

Definition

`Set3DSoundScriptIntVariable(string 3DSoundName, string variable, int value)`

Description

This function sets the value of the Integer **`variable`** defined in the script attached to the **`3DSoundName`** 3D sound object to **`value`**.

Parameters

`3DSoundName`
Specifies the name of the 3D sound object.

`variable`
Specifies the name of the Integer variable defined in the script attached to the **`3DSoundName`** 3D sound.

`value`
Specifies the Integer value for the variable **`variable`**.

Example

```
--script name is Set3DSoundScriptIntVariable.lua attached a to game object such as water

function Init()
    Set3DSoundScriptIntVariable("sound1", "a", 1)
end

function Update()

end
```

Assuming that the variable `"a"` is defined in the script attached to the 3D sound object `"sound1"`, `Set3DSoundScriptIntVariable` function sets the `"a"` variable to **1**.

4.341. Set3DSoundScriptStringVariable

Definition

`Set3DSoundScriptStringVariable(string 3DSoundName, string variable, string value)`

Description

This function sets the value of the String **`variable`** defined in the script attached to the **`3DSoundName`** 3D sound object to **`value`**.

Parameters

`3DSoundName`
Specifies the name of the 3D sound object.

`variable`
Specifies the name of the String variable defined in the script attached to the **`3DSoundName`** 3D sound.

`value`
Specifies the String value for the variable **`variable`**.

Example

```
--script name is Set3DSoundScriptStringVariable.lua attached a to game object such as
water

function Init()
    Set3DSoundScriptStringVariable("sound1", "a", "hello")
end

function Update()

end
```

Assuming that the variable **`"a"`** is defined in the script attached to the 3D sound object **`"sound1"`**, **`Set3DSoundScriptStringVariable`** function sets the **`"a"`** variable to **`"hello"`**.

4.342. SetAmbientSoundScriptBoolVariable

Definition

`SetAmbientSoundScriptBoolVariable(string ambientSoundName, string variable, bool value)`

Description

This function sets the value of the Boolean **`variable`** defined in the script attached to the **`ambientSoundName`** ambient sound object to **`value`**.

Parameters

`ambientSoundName`
Specifies the name of the ambient sound object.

`variable`
Specifies the name of the Boolean variable defined in the script attached to the **`ambientSoundName`** ambient sound.

`value`
Specifies the Boolean value for the variable **`variable`**.

Example

```
--script name is SetAmbientSoundScriptBoolVariable.lua attached a to game object such as
water

function Init()
    SetAmbientSoundScriptBoolVariable("sound1", "a", true)
end

function Update()

end
```

Assuming that the variable `"a"` is defined in the script attached to the ambient sound object `"sound1"`, `SetAmbientSoundScriptBoolVariable` function sets the `"a"` variable to *true*.

4.343. SetAmbientSoundScriptDoubleVariable

Definition

`SetAmbientSoundScriptDoubleVariable(string ambientSoundName, string variable, double value)`

Description

This function sets the value of the Double `variable` defined in the script attached to the `ambientSoundName` ambient sound object to `value`.

Parameters

ambientSoundName
Specifies the name of the ambient sound object.

variable
Specifies the name of the Double variable defined in the script attached to the `ambientSoundName` ambient sound.

value
Specifies the Double value for the variable `variable`.

Example

```
--script name is SetAmbientSoundScriptDoubleVariable.lua attached a to game object such
as water

function Init()
    SetAmbientSoundScriptDoubleVariable("sound1", "a", 1.0)
end

function Update()

end
```

Assuming that the variable `"a"` is defined in the script attached to the ambient sound object `"sound1"`, `SetAmbientSoundScriptDoubleVariable` function sets the `"a"` variable to `1.0`.

4.344. SetAmbientSoundScriptIntVariable

Definition

`SetAmbientSoundScriptIntVariable(string ambientSoundName, string variable, int value)`

Description

This function sets the value of the Integer **`variable`** defined in the script attached to the **`ambientSoundName`** ambient sound object to **`value`**.

Parameters

`ambientSoundName`
Specifies the name of the ambient sound object.

`variable`
Specifies the name of the Integer variable defined in the script attached to the **`ambientSoundName`** ambient sound.

`value`
Specifies the Integer value for the variable **`variable`**.

Example

```
--script name is SetAmbientSoundScriptIntVariable.lua attached a to game object such as water

function Init()
    SetAmbientSoundScriptIntVariable("sound1", "a", 1)
end

function Update()

end
```

Assuming that the variable `"a"` is defined in the script attached to the ambient sound object `"sound1"`, `SetAmbientSoundScriptIntVariable` function sets the `"a"` variable to **`1`**.

4.345. SetAmbientSoundScriptStringVariable

Definition

`SetAmbientSoundScriptStringVariable(string ambientSoundName, string variable, string value)`

Description

This function sets the value of the String **`variable`** defined in the script attached to the **`ambientSoundName`** ambient sound object to **`value`**.

Parameters

`ambientSoundName`
Specifies the name of the ambient sound object.

`variable`
Specifies the name of the String variable defined in the script attached to the **`ambientSoundName`** ambient sound.

`value`
Specifies the String value for the variable **`variable`**.

Example

```lua
--script name is SetAmbientSoundScriptStringVariable.lua attached a to game object such
as water

function Init()
    SetAmbientSoundScriptStringVariable("sound1", "a", "hello")
end

function Update()

end
```

Assuming that the variable **`"a"`** is defined in the script attached to the ambient sound object **`"sound1"`**, **`SetAmbientSoundScriptStringVariable`** function sets the **`"a"`** variable to **`"hello"`**.

4.346. SetAnisotropicFilteringValue

Definition

`SetAnisotropicFilteringValue(int value)`

Description

This function sets the anisotropic texture filtering value to `value`.

Parameters

`value`

The value of anisotropic texture filtering to be set. Accepted values are 0, 2, 4, 8 or 16.

Example

```
function Init()
    SetAnisotropicFilteringValue(2)
end

function Update()

end
```

In this example, we set the value of anisotropic texture filtering to **2**.

4.347. SetBloomColor

Definition

`SetBloomColor(float red, float green, float blue)`

Description

This function sets the bloom color.

Parameters

red, green, blue

Specify the red, green and blue components of bloom color. Each of these three values must be between 0.0 and 1.0.

Example

```
function Init()
    SetBloomColor(0.75, 0.5, 0.25)
end

function Update()

end
```

In this example, we set the red, green, and blue components of bloom color to 0.75, 0.5, and 0.25, respectively.

4.348. SetBloomIntensity

Definition

```
SetBloomIntensity(float intensity)
```

Description

This function sets the bloom intensity.

Parameters

`intensity`

Specifies the bloom intensity. This value must be between 0.0 and 1.0.

Example

```
function Init()
    SetBloomIntensity(0.5)
end

function Update()

end
```

In this example, we set the bloom intensity to 0.5.

4.349. SetCameraScriptBoolVariable

Definition

`SetCameraScriptBoolVariable(string cameraName, string variable, bool value)`

Description

This function sets the value of the Boolean **`variable`** defined in the script attached to the **`cameraName`** engine camera object to **`value`**.

Parameters

`cameraName`
Specifies the name of the engine camera object.

`variable`
Specifies the name of the Boolean variable defined in the script attached to the **`cameraName`** engine camera.

`value`
Specifies the Boolean value for the variable **`variable`**.

Example

```
--script name is SetCameraScriptBoolVariable.lua attached a to game object such as water

function Init()
    SetCameraScriptBoolVariable("camera1", "a", true)
end

function Update()

end
```

Assuming that the variable `"a"` is defined in the script attached to the engine camera object `"camera1"`, `SetCameraScriptBoolVariable` function sets the `"a"` variable to *true*.

4.350. SetCameraScriptDoubleVariable

Definition

`SetCameraScriptDoubleVariable(string cameraName, string variable, double value)`

Description

This function sets the value of the Double **`variable`** defined in the script attached to the **`cameraName`** engine camera object to **`value`**.

Parameters

`cameraName`
Specifies the name of the engine camera object.

`variable`
Specifies the name of the Double variable defined in the script attached to the **`cameraName`** engine camera.

`value`
Specifies the Double value for the variable **`variable`**.

Example

```
--script name is SetCameraScriptDoubleVariable.lua attached a to game object such as
water

function Init()
    SetCameraScriptDoubleVariable("camera1", "a", 1.0)
end

function Update()

end
```

Assuming that the variable `"a"` is defined in the script attached to the engine camera object `"camera1"`, `SetCameraScriptDoubleVariable` function sets the `"a"` variable to `1.0`.

4.351. SetCameraScriptIntVariable

Definition

`SetCameraScriptIntVariable(string cameraName, string variable, int value)`

Description

This function sets the value of the Integer **`variable`** defined in the script attached to the **`cameraName`** engine camera object to **`value`**.

Parameters

`cameraName`
Specifies the name of the engine camera object.

`variable`
Specifies the name of the Integer variable defined in the script attached to the **`cameraName`** engine camera.

`value`
Specifies the Integer value for the variable **`variable`**.

Example

```
--script name is SetCameraScriptIntVariable.lua attached a to game object such as water

function Init()
    SetCameraScriptIntVariable("camera1", "a", 1)
end

function Update()

end
```

Assuming that the variable `"a"` is defined in the script attached to the engine camera object `"camera1"`, `SetCameraScriptIntVariable` function sets the `"a"` variable to `1`.

4.352. SetCameraScriptStringVariable

Definition

`SetCameraScriptStringVariable(string cameraName, string variable, string value)`

Description

This function sets the value of the String **`variable`** defined in the script attached to the **`cameraName`** engine camera object to **`value`**.

Parameters

`cameraName`

Specifies the name of the engine camera object.

`variable`

Specifies the name of the String variable defined in the script attached to the **`cameraName`** engine camera.

`value`

Specifies the String value for the variable **`variable`**.

Example

```
--script name is SetCameraScriptStringVariable.lua attached a to game object such as
water

function Init()
    SetCameraScriptStringVariable("camera1", "a", "hello")
end

function Update()

end
```

Assuming that the variable `"a"` is defined in the script attached to the engine camera object `"camera1"`, `SetCameraScriptStringVariable` function sets the `"a"` variable to `"hello"`.

4.353. SetCharacterControllerCapsuleHeight

Definition

`SetCharacterControllerCapsuleHeight(float height)`

Description

This function sets the value of physics character controller capsule height to `height`.

Parameters

`height`

Specifies the capsule height of physics character controller. This value must be greater than 0.0.

Example

```
function Init()
    SetCharacterControllerCapsuleHeight(3.0)
end

function Update()

end
```

This script sets the physics character controller capsule height to `3.0`.

4.354. SetCharacterControllerCapsuleRadius

Definition

`SetCharacterControllerCapsuleRadius(float radius)`

Description

This function sets the value of physics character controller capsule radius to **`radius`**.

Parameters

`radius`

Specifies the capsule radius of physics character controller. This value must be greater than 0.0.

Example

```
function Init()
    SetCharacterControllerCapsuleRadius(2.0)
end

function Update()

end
```

This script sets the physics character controller capsule radius to `2.0`.

4.355. SetCharacterControllerForcePower

Definition

`SetCharacterControllerForcePower(float forcePower)`

Description

This function sets the value of physics character controller force power to `forcePower`.

Parameters

`forcePower`

Specifies the force power of physics character controller.

Example

```
function Init()
    SetCharacterControllerForcePower(10.0)
end

function Update()

end
```

This script sets the physics character controller force power to `10.0`.

4.356. SetCharacterControllerJumpPower

Definition

`SetCharacterControllerJumpPower(float jumpPower)`

Description

This function sets the value of physics character controller jump power to `jumpPower`.

Parameters

`jumpPower`

Specifies the jump power of physics character controller.

Example

```
function Init()
    SetCharacterControllerJumpPower(15.0)
end

function Update()

end
```

This script sets the physics character controller jump power to `15.0`.

4.357. SetCharacterControllerPosition

Definition

```
SetCharacterControllerPosition(float x, float y, float z)
```

Description

This function sets the three dimensional position of physics character controller.

Parameters

x, y, z

Specify the position of physics character controller.

Example

```
function Init()
    SetCharacterControllerPosition(2.5, 5.0, 7.0)
end

function Update()

end
```

This script sets the X, Y and Z position of physics character controller to `2.5`, `5.0` and `7.0`, respectively.

4.358. SetCharacterControllerRunSpeed

Definition

`SetCharacterControllerRunSpeed(float speed)`

Description

This function sets the value of physics character controller running speed to `speed`.

Parameters

speed

Specifies the running speed of physics character controller.

Example

```
function Init()
    SetCharacterControllerRunSpeed(10.0)
end

function Update()

end
```

This script sets the physics character controller running speed to `10.0`.

4.359. SetCharacterControllerStepOffset

Definition

`SetCharacterControllerStepOffset(float stepOffset)`

Description

This function sets the value of physics character controller step offset to `stepOffset`.

Parameters

`stepOffset`

Specifies the step offset of physics character controller. This value must be equal to or greater than 0.0.

Example

```
function Init()
    SetCharacterControllerStepOffset(0.2)
end

function Update()

end
```

This script sets the physics character controller step offset to `0.2`.

4.360. SetCharacterControllerWalkSpeed

Definition

`SetCharacterControllerWalkSpeed(float speed)`

Description

This function sets the value of physics character controller walking speed to `speed`.

Parameters

speed

Specifies the walking speed of physics character controller.

Example

```
function Init()
    SetCharacterControllerWalkSpeed(5.0)
end

function Update()

end
```

This script sets the physics character controller walking speed to `5.0`.

4.361. SetDepthOfFieldFocalDistance

Definition

`SetDepthOfFieldFocalDistance(float focalDistance)`

Description

This function sets the focal distance value of depth of field effect to `focalDistance`.

Parameters

`focalDistance`

Specifies the focal distance of depth of field effect. This value must be equal to or greater than 0.0.

Example

```
function Init()
    SetDepthOfFieldFocalDistance(10.0)
end

function Update()

end
```

This script sets the focal distance of the depth of field effect to `10.0`.

4.362. SetDepthOfFieldFocalRange

Definition

`SetDepthOfFieldFocalRange(float focalRange)`

Description

This function sets the focal range value of depth of field effect to `focalRange`.

Parameters

`focalRange`

Specifies the focal range of depth of field effect. This value must be equal to or greater than 0.0.

Example

```
function Init()
    SetDepthOfFieldFocalRange(20.0)
end

function Update()

end
```

This script sets the focal range of the depth of field effect to `20.0`.

4.363. SetDirectionalShadowAlgorithm

Definition

`SetDirectionalShadowAlgorithm(string shadowAlgorithmCode)`

Description

This function sets the algorithm of directional light shadow to `shadowAlgorithmCode`.

Parameters

`shadowAlgorithmCode`

Specifies the algorithm of directional light shadow. Accepted values are:

- `"SHADOW_SINGLE_HL"`
- `"SHADOW_SINGLE"`
- `"SHADOW_MULTI_LEAK"`
- `"SHADOW_MULTI_NOLEAK"`
- `"SHADOW_PCF"`
- `"SHADOW_PCF_TRILIN"`
- `"SHADOW_PCF_4TAP"`
- `"SHADOW_PCF_8TAP"`
- `"SHADOW_PCF_GAUSSIAN"`

Example

```
function Init()
    SetDirectionalShadowAlgorithm("SHADOW_PCF")
end

function Update()

end
```

This script sets the algorithm of direcional light shadow to `"SHADOW_PCF"`.

4.364. SetDirectionalShadowFarClipPlane

Definition

`SetDirectionalShadowFarClipPlane(float farClipPlane)`

Description

This function sets the far clip plane of directional light shadow to `farClipPlane`.

Parameters

`farClipPlane`

Specifies the far clip plane of directional light shadow. This value must be greater than zero.

Example

```
function Init()
    SetDirectionalShadowFarClipPlane(70.0)
end

function Update()

end
```

This script sets the far clip plane of directional light shadow to `70.0`.

4.365. SetDirectionalShadowIntensity

Definition

`SetDirectionalShadowIntensity(float shadowIntensity)`

Description

This function sets the intensity of directional light shadow to `shadowIntensity`.

Parameters

`shadowIntensity`

Specifies the intensity of directional light shadow. This value should be in the range [0,1].

Example

```
function Init()
    SetDirectionalShadowIntensity(0.9)
end

function Update()

end
```

This script sets the intensity of directional light shadow to `0.9`.

4.366. SetDirectionalShadowLight

Definition

`SetDirectionalShadowLight(string directionalLightName)`

Description

This function specifies the directional light that can cast the shadows. It should be noted that only one directional light in current VScene can cast the shadows.

Parameters

`directionalLightName`

Specifies the directional light name that can cast the shadows.

Example

```
function Init()
    SetDirectionalShadowLight("light2")
end

function Update()

end
```

Assume that `"light2"` is a directional light. The `SetDirectionalShadowLight` function in this example determines that `"light2"` will cast the shadows.

4.367. SetDirectionalShadowNearClipPlane

Definition

`SetDirectionalShadowNearClipPlane(float nearClipPlane)`

Description

This function sets the near clip plane of directional light shadow to `nearClipPlane`.

Parameters

`nearClipPlane`

Specifies the near clip plane of directional light shadow. This value must be greater than zero.

Example

```
function Init()
    SetDirectionalShadowNearClipPlane(0.2)
end

function Update()

end
```

This script sets the near clip plane of directional light shadow to `0.2`.

4.368. SetDirectionalShadowNumberOfSplits

Definition

`SetDirectionalShadowNumberOfSplits(int numberOfSplits)`

Description

This function sets the number of splits of directional light shadow to `numberOfSplits`.

Parameters

`numberOfSplits`

Specifies the number of splits of directional light shadow. Accepted values are 1, 2, 3 and 4.

Example

```
function Init()
    SetDirectionalShadowNumberOfSplits(2)
end

function Update()

end
```

This script sets the number of splits of directional light shadow to **2**.

4.369. SetDirectionalShadowResolution

Definition

`SetDirectionalShadowResolution(int shadowResolution)`

Description

This function sets the resolution of directional light shadow to **`shadowResolution`**.

Parameters

`shadowResolution`

Specifies the resolution of directional light shadow. Accepted values are 1024, 2048 and 4096.

Example

```
function Init()
    SetDirectionalShadowResolution(1024)
end

function Update()

end
```

This script sets the resolution of directional light shadow to **`1024`**.

4.370. SetDirectionalShadowWeightOfSplits

Definition

`SetDirectionalShadowWeightOfSplits(float weightOfSplits)`

Description

This function sets the weight of splits of directional light shadow to ***`weightOfSplits`***.

Parameters

`weightOfSplits`

Specifies the weight of splits of directional light shadow. This value should be in the range [0,1].

Example

```
function Init()
    SetDirectionalShadowWeightOfSplits(0.6)
end

function Update()

end
```

This script sets the weight of splits of directional light shadow to `0.6`.

4.371. SetDistanceBetweenPhysicsCameraAndCharacterController

Definition

```
SetDistanceBetweenPhysicsCameraAndCharacterController(float distance)
```

Description

This function sets the the distance between physics camera and physics character controller.

Parameters

distance

Specifies the distance between physics camera and physics character controller. This value must be greater than 0.0.

Example

```
function Init()
    SetDistanceBetweenPhysicsCameraAndCharacterController(5.0)
end

function Update()

end
```

This script sets the distance between physics camera and physics character controller to `5.0`.

4.372. SetEngineCameraAngle

Definition

`SetEngineCameraAngle(string engineCameraName, float angle)`

Description

This function sets the angle value of the engine camera `engineCameraName` to `angle`.

Parameters

engineCameraName

Specifies the name of the engine camera. You can also use the name "this" for this parameter. In this case, "this" refers to the camera object that this script is attached to.

angle

Specifies the engine camera angle.

Example 1

```
function Init()
    ActivateEngineCamera("camera1")
    SetEngineCameraAngle("camera1", 30.0)
end

function Update()

end

function Update()

end
```

First we activate the engine camera `"camera1"`. Then we set the angle of engine camera `"camera1"` to `30.0` degrees.

Example 2

```
--Name of script is SetEngineCameraAngle2.lua

function Init()
    ActivateEngineCamera("this")
    SetEngineCameraAngle("this", 30.0)
end

function Update()

end
```

In this case, `"this"` string in the `SetEngineCameraAngle` points to the camera that `SetEngineCameraAngle2.lua` script is attached to. For example, if `SetEngineCameraAngle2.lua` script is attached to a engine camera named "camera1", `"this"` will be equivalent to the name "camera1". In this example, we activate the current engine camera (for example, "camera1"). Then we set the angle of current engine camera to `30.0` degrees.

4.373. SetEngineCameraFarClipPlane

Definition

`SetEngineCameraFarClipPlane(string engineCameraName, float farClipPlane)`

Description

This function sets the far clip plane value of the engine camera `engineCameraName` to `farClipPlane`.

Parameters

`engineCameraName`

Specifies the name of the engine camera. You can also use the name "this" for this parameter. In this case, "this" refers to the camera object that this script is attached to.

`farClipPlane`

Specifies the far clip plane of engine camera. This value must be greater than 0.0.

Example 1

```
function Init()
    ActivateEngineCamera("camera1")
    SetEngineCameraFarClipPlane("camera1", 20.0)
end

function Update()

end

function Update()

end
```

First we activate the engine camera `"camera1"`. Then we set the far clip plane of engine camera `"camera1"` to `20.0`.

Example 2

```
--Name of script is SetEngineCameraFarClipPlane2.lua

function Init()
    ActivateEngineCamera("this")
    SetEngineCameraFarClipPlane("this", 20.0)
end

function Update()

end
```

In this case, `"this"` string in the `SetEngineCameraFarClipPlane` points to the camera that `SetEngineCameraFarClipPlane2.lua` script is attached to. For example, if `SetEngineCameraFarClipPlane2.lua` script is attached to a engine camera named "camera1", `"this"` will be equivalent to the name "camera1". In this example, we activate the current engine camera (for example, "camera1"). Then we set the far clip plane of current engine camera to **`20.0`**.

4.374. SetEngineCameraNearClipPlane

Definition

`SetEngineCameraNearClipPlane(string engineCameraName, float nearClipPlane)`

Description

This function sets the near clip plane value of the engine camera `engineCameraName` to `nearClipPlane`.

Parameters

engineCameraName

Specifies the name of the engine camera. You can also use the name "this" for this parameter. In this case, "this" refers to the camera object that this script is attached to.

nearClipPlane

Specifies the near clip plane of engine camera. This value must be greater than 0.0.

Example 1

```lua
function Init()
    ActivateEngineCamera("camera1")
    SetEngineCameraNearClipPlane("camera1", 0.1)
end

function Update()

end
```

First we activate the engine camera `"camera1"`. Then we set the near clip plane of engine camera `"camera1"` to `0.1`.

Example 2

```lua
--Name of script is SetEngineCameraNearClipPlane2.lua

function Init()
    ActivateEngineCamera("this")
    SetEngineCameraNearClipPlane("this", 0.1)
end

function Update()

end
```

In this case, `"this"` string in the `SetEngineCameraNearClipPlane` function points to the camera that `SetEngineCameraNearClipPlane2.lua` script is attached to. For example, if `SetEngineCameraNearClipPlane2.lua` script is attached to a engine camera named "camera1", `"this"` will be equivalent to the name "camera1". In this example, we activate the current engine camera (for example, "camera1"). Then we set the near clip plane of current engine camera to `0.1`.

4.375. SetEngineCameraPan

Definition

```
SetEngineCameraPan(string engineCameraName, float pan)
```

Description

This function sets the pan value of the engine camera `engineCameraName` to `pan`.

Parameters

engineCameraName
Specifies the name of the engine camera. You can also use the name "this" for this parameter. In this case, "this" refers to the camera object that this script is attached to.

pan
Specifies the engine camera pan.

Example 1

```lua
function Init()
    ActivateEngineCamera("camera1")
    SetEngineCameraPan("camera1", 70.0)
end

function Update()

end
```

First we activate the engine camera `"camera1"`. Then we set the pan of engine camera `"camera1"` to `70.0`.

Example 2

```lua
--Name of script is SetEngineCameraPan2.lua

function Init()
    ActivateEngineCamera("this")
    SetEngineCameraPan("this", 70.0)
end

function Update()

end
```

In this case, `"this"` string in the `SetEngineCameraPan` function points to the camera that `SetEngineCameraPan2.lua` script is attached to. For example, if `SetEngineCameraPan2.lua` script is attached to a engine camera named "camera1", `"this"` will be equivalent to the name "camera1".
In this example, we activate the current engine camera (for example, "camera1"). Then we set the pan of current engine camera to `70.0`.

4.376. SetEngineCameraPosition

Definition

`SetEngineCameraPosition(string engineCameraName, float x, float y, float z)`

Description

This function sets the position of the engine camera `engineCameraName`.

Parameters

`engineCameraName`

Specifies the name of the engine camera. You can also use the name "this" for this parameter. In this case, "this" refers to the camera object that this script is attached to.

`x, y, z`

Specify the X, Y and Z components of engine camera position.

Example 1

```lua
function Init()
    ActivateEngineCamera("camera1")
    SetEngineCameraPosition("camera1", 2.5, 5.0, 7.0)
end

function Update()

end
```

First we activate the engine camera `"camera1"`. Then we set the position of engine camera `"camera1"` to `(2.5, 5.0, 7.0)`.

Example 2

```lua
--Name of script is SetEngineCameraPosition2.lua

function Init()
    ActivateEngineCamera("this")
    SetEngineCameraPosition("this", 2.5, 5.0, 7.0)
end

function Update()

end
```

In this case, `"this"` string in the `SetEngineCameraPosition` function points to the camera that `SetEngineCameraPosition2.lua` script is attached to. For example, if `SetEngineCameraPosition2.lua` script is attached to a engine camera named "camera1", `"this"` will be equivalent to the name "camera1".
In this example, we activate the current engine camera (for example, "camera1"). Then we set the position of current engine camera to `(2.5, 5.0, 7.0)`.

4.377. SetEngineCameraTilt

Definition

`SetEngineCameraTilt(string engineCameraName, float tilt)`

Description

This function sets the tilt value of the engine camera `engineCameraName` to `tilt`.

Parameters

engineCameraName

Specifies the name of the engine camera. You can also use the name "this" for this parameter. In this case, "this" refers to the camera object that this script is attached to.

tilt

Specifies the engine camera tilt.

Example 1

```
function Init()
    ActivateEngineCamera("camera1")
    SetEngineCameraTilt("camera1", -20.0)
end

function Update()

end
```

First we activate the engine camera `"camera1"`. Then we set the tilt of engine camera `"camera1"` to `-20.0`.

Example 2

```
--Name of script is SetEngineCameraTilt2.lua

function Init()
    ActivateEngineCamera("this")
    SetEngineCameraTilt("this", -20.0)
end

function Update()

end
```

In this case, `"this"` string in the `SetEngineCameraTilt` function points to the camera that `SetEngineCameraTilt2.lua` script is attached to. For example, if `SetEngineCameraTilt2.lua` script is attached to a engine camera named "camera1", `"this"` will be equivalent to the name "camera1".
In this example, we activate the current engine camera (for example, "camera1"). Then we set the tilt of current engine camera to `-20.0`.

4.378. SetFogColor

Definition

`SetFogColor(float red, float green, float blue)`

Description

This function sets the fog color.

Parameters

`red, green, blue`

Specify the red, green and blue components of fog color. Each of these three values must be in the range [0.0,1.0].

Example

```
function Init()
    SetFogColor(0.25, 0.5, 0.7)
end

function Update()

end
```

In this example, we set the red, green, and blue components of fog color to `0.25`, `0.5`, and `0.7`, respectively.

4.379. SetFogDensity

Definition

```
SetFogDensity(float density)
```

Description

This function sets the fog density.

Parameters

`density`

Specifies the fog density. This value must be greater than 0.0.

Example

```
function Init()
    SetFogDensity(0.5)
end

function Update()

end
```

In this example, we set the fog density to `0.5`.

4.380. SetGlobalSoundVolume

Definition

`SetGlobalSoundVolume(float volume)`

Description

This function sets the global sound volume.

Parameter

`volume`

Specifies the global sound volume. This value must be in the range [0.0, 1.0].

Example

```
function Init()
    SetGlobalSoundVolume(0.5)
end

function Update()

end
```

This script sets the global sound volume to `0.5`.

4.381. SetGUIButtonPosition

Definition

`SetGUIButtonPosition(string GUIName, string buttonName, int x, int y)`

Description

This function sets the two-dimensional position of the button **`buttonName`** of GUI **`GUIName`** relative to the lower left part of the screen as two x, y values. You can view and copy the name of the GUIs and their buttons in the *Script Utility* section (Tools > Script Editor > Tools > Script Utility) or the *Prefabs and GUIs* section of the current VScene.

Parameters

`GUIName`
The name of the GUI to which the **`buttonName`** button belongs.

`buttonName`
The name of the button that belongs to **`GUIName`**.

`x, y`
Specify the two-dimensional position of the button **`buttonName`** of GUI **`GUIName`** relative to the lower left part of the screen as two x, y values.

Example

```
function Init()
    SetGUIButtonPosition("gui_SampleGUI17_MainMenu", "PlayGame", GetScreenWidth() / 2,
GetScreenHeight() / 2)
end

function Update()

end
```

In this example, the `SetGUIButtonPosition` function sets the X and Y position of the `"PlayGame"` button from the GUI named `"gui_SampleGUI17_MainMenu"` to (screen width / 2) and (screen height / 2), respectively.

4.382. SetGUIButtonScriptBoolVariable

Definition

`SetGUIButtonScriptBoolVariable(string GUIName, string buttonName, string variable, bool value)`

Description

This function sets the value of the Boolean **`variable`** defined in the script attached to the **`buttonName`** button that belongs to **`GUIName`** GUI.

Parameters

`GUIName`
Specifies the the name of the GUI to which the **`buttonName`** button belongs.

`buttonName`
Specifies the the name of the button that belongs to **`GUIName`** GUI.

`variable`
Specifies the name of the Boolean variable defined in the script attached to the **`buttonName`** button.

`value`
Specifies the value of variable **`variable`** to be set.

Example

```
--script name is SetGUIButtonScriptBoolVariable.lua attached a to game object such as
water

function Init()
    SetGUIButtonScriptBoolVariable("gui_pack1_button", "PlayGame", "a", true)
end

function Update()

end
```

Assuming that the variable `"a"` is defined in the script attached to the button object `"PlayGame"` that belongs to `"gui_pack1_button"` GUI , `SetGUIButtonScriptBoolVariable` function sets the value of variable `"a"` to *true*.

4.383. SetGUIButtonScriptDoubleVariable

Definition

`SetGUIButtonScriptDoubleVariable(string GUIName, string buttonName, string variable, double value)`

Description

This function sets the value of the Double **`variable`** defined in the script attached to the **`buttonName`** button that belongs to **`GUIName`** GUI.

Parameters

`GUIName`

Specifies the the name of the GUI to which the **`buttonName`** button belongs.

`buttonName`

Specifies the the name of the button that belongs to **`GUIName`** GUI.

`variable`

Specifies the name of the Double variable defined in the script attached to the **`buttonName`** button.

`value`

Specifies the value of variable **`variable`** to be set.

Example

```lua
--script name is SetGUIButtonScriptDoubleVariable.lua attached a to game object such as water

function Init()
    SetGUIButtonScriptDoubleVariable("gui_pack1_button", "PlayGame", "a", 1.0)
end

function Update()

end
```

Assuming that the variable `"a"` is defined in the script attached to the button object `"PlayGame"` that belongs to `"gui_pack1_button"` GUI , `SetGUIButtonScriptDoubleVariable` function sets the value of variable `"a"` to *1.0*.

4.384. SetGUIButtonScriptIntVariable

Definition

`SetGUIButtonScriptIntVariable(string GUIName, string buttonName, vstring variable, int value)`

Description

This function sets the value of the Integer **`variable`** defined in the script attached to the **`buttonName`** button that belongs to **`GUIName`** GUI.

Parameters

`GUIName`
Specifies the the name of the GUI to which the **`buttonName`** button belongs.

`buttonName`
Specifies the the name of the button that belongs to **`GUIName`** GUI.

`variable`
Specifies the name of the Integer variable defined in the script attached to the **`buttonName`** button.

`value`
Specifies the value of variable **`variable`** to be set.

Example

```
--script name is SetGUIButtonScriptIntVariable.lua attached a to game object such as
water

function Init()
    SetGUIButtonScriptIntVariable("gui_pack1_button", "PlayGame", "a", 1)
end

function Update()

end
```

Assuming that the variable `"a"` is defined in the script attached to the button object `"PlayGame"` that belongs to `"gui_pack1_button"` GUI , `SetGUIButtonScriptIntVariable` function sets the value of variable `"a"` to *1*.

4.385. SetGUIButtonScriptStringVariable

Definition

`SetGUIButtonScriptStringVariable(string GUIName, string buttonName, string variable, string value)`

Description

This function sets the value of the String **`variable`** defined in the script attached to the **`buttonName`** button that belongs to **`GUIName`** GUI.

Parameters

`GUIName`
Specifies the the name of the GUI to which the **`buttonName`** button belongs.

`buttonName`
Specifies the the name of the button that belongs to **`GUIName`** GUI.

`variable`
Specifies the name of the String variable defined in the script attached to the **`buttonName`** button.

`value`
Specifies the value of variable **`variable`** to be set.

Example

```lua
--script name is SetGUIButtonScriptStringVariable.lua attached a to game object such as water

function Init()
    SetGUIButtonScriptStringVariable("gui_pack1_button", "PlayGame", "a", "hello")
end

function Update()

end
```

Assuming that the variable `"a"` is defined in the script attached to the button object `"PlayGame"` that belongs to `"gui_pack1_button"` GUI , `SetGUIButtonScriptStringVariable` function sets the value of variable `"a"` to "*hello*".

4.386. SetGUIImagePosition

Definition

`SetGUIImagePosition(string GUIName, string imageName, int x, int y)`

Description

This function sets the two-dimensional position of the image **`imageName`** of GUI **`GUIName`** relative to the lower left part of the screen as two x, y values. You can view and copy the name of the GUIs and their images in the *Script Utility* section (Tools > Script Editor > Tools > Script Utility) or the *Prefabs and GUIs* section of the current VScene.

Parameters

`GUIName`
Specifies the name of the GUI to which the **`imageName`** image belongs.

`imageName`
Specifies the the name of the image that belongs to **`GUIName`**.

`x, y`
Specify the two-dimensional position of the image **`imageName`** of GUI **`GUIName`** relative to the lower left part of the screen as two x, y values.

Example

```
function Init()
    SetGUIImagePosition("gui_SampleGUI17_MainMenuAbout", "backgroundImg",
GetScreenWidth() / 2, GetScreenHeight() / 2)
end

function Update()

end
```

In this example, `SetGUIImagePosition` function sets the X and Y components of 2D position of the `"backgroundImg"` image from the GUI named `"gui_SampleGUI17_MainMenuAbout"` to (screen width / 2) and (screen height / 2), respectively.

4.387. SetGUIPosition

Definition

`SetGUIPosition(string GUIName, int x, int y)`

Description

This function specifies the X and Y of the GUI **`GUIName`** as a percentage of the screen width and height. You can view and copy the name of the GUIs in the *Script Utility* section (Tools > Script Editor > Tools > Script Utility) or the *Prefabs and GUIs* section of the current VScene.

Parameters

`GUIName`
Specifies the name of the GUI.

`x, y`
specify the X and Y of the GUIName as a percentage of the screen width and height. Each of these two values must be in the range [-100, 100]. -100 means (-screen width) or (-screen height) and 100 means (screen width) or (screen height). It should be noted that the width of the GUI ranges from (-screen width) to (screen width) and the height of the GUI ranges from (-screen height) to (screen height).

Example

```
function Init()
    SetGUIPosition("gui_SampleGUI17_MainMenu", -5, 10)
end

function Update()

end
```

In this example, `SetGUIPosition` function sets the X and Y of the GUI named `"gui_SampleGUI17_MainMenu"` to -5 and 10 percents of the screen width and height, respectively. Assuming that the width and height of the screen are equal to 1024 and 768 respectively, these numbers will be equal to (-5 * 1024 / 100 = -51.2) and (10 * 768 / 100 = 76.8) respectively, , in screen coordinates.

4.388. SetGUITextPosition

Definition

`SetGUITextPosition(string GUIName, string textName, int x, int y)`

Description

This function sets the two-dimensional position of the text **`textName`** of GUI **`GUIName`** relative to the lower left part of the screen as two x, y values. You can view and copy the name of the GUIs and their texts in the *Script Utility* section (Tools > Script Editor > Tools > Script Utility) or the *Prefabs and GUIs* section of the current VScene.

Parameters

`GUIName`
Specifies the name of the GUI to which the **`textName`** text belongs.

`textName`
Specifies the the name of the text that belongs to **`GUIName`**.

`x, y`
Specify the two-dimensional position of the text **`textName`** of GUI **`GUIName`** relative to the lower left part of the screen as two x, y values.

Example

```
function Init()
    SetGUITextPosition("gui_SampleGUI17_MainMenu", "text1", GetScreenWidth() / 2,
GetScreenHeight() / 2)
end

function Update()

end
```

In this example, `SetGUITextPosition` function sets the 2D position of the text `"text1"` from the GUI named `"gui_SampleGUI17_MainMenu"` to (screen width / 2) and (screen height / 2), respectively.

4.389. SetLightAmbient

Definition

`SetLightAmbient(string lightObjectName, float red, float green, float blue)`

Description

This function sets the ambient color of `lightObjectName` light as three values of red, green and blue. Each value ranges from 0.0 to 1.0.

Parameters

`lightObjectName`

Specifies the name of the light object. You can also use the name "this" for this parameter. In this case, "this" refers to the light object name to which this script is attached.

`red, green, blue`

Specify the ambient color of `lightObjectName` light as three values of red, green and blue. Each value ranges from 0.0 to 1.0.

Example 1

```
function Init()
    SetLightAmbient("light1", 0.25, 0.5, 0.75)
end

function Update()

end
```

In this example, the `SetLightAmbient` function sets the value of the red, green, and blue components of the ambient color of light `"light1"` to `0.25`, `0.5` and `0.75`, respectively.

Example 2

```
--Script name is SetLightAmbient2.lua

function Init()
    SetLightAmbient("this", 0.25, 0.5, 0.75)
end

function Update()

end
```

Assume that the above script named `SetLightAmbient2.lua` is attached to the light object named "light1". In this case, string `"this"` in the `SetLightAmbient` function will be equal to "light1". In our example, the function `SetLightAmbient` sets the values of red, green and blue components of ambient color of current light, which is "light1", to `0.25`, `0.5` and `0.75`, respectively.

4.390. SetLightDiffuse

Definition

`SetLightDiffuse(string lightObjectName, float red, float green, float blue)`

Description

This function sets the diffuse color of `lightObjectName` light as three values of red, green and blue. Each value ranges from 0.0 to 1.0.

Parameters

`lightObjectName`

Specifies the name of the light object. You can also use the name "this" for this parameter. In this case, "this" refers to the light object name to which this script is attached.

`red, green, blue`

Specify the diffuse color of `lightObjectName` light as three values of red, green and blue. Each value ranges from 0.0 to 1.0.

Example 1

```
function Init()
    SetLightDiffuse("light1", 0.25, 0.5, 0.75)
end

function Update()

end
```

In this example, the `SetLightDiffuse` function sets the value of the red, green, and blue components of the diffuse color of light `"light1"` to `0.25`, `0.5` and `0.75`, respectively.

Example 2

```
--Script name is SetLightDiffuse2.lua

function Init()
    SetLightDiffuse("this", 0.25, 0.5, 0.75)
end

function Update()

end
```

Assume that the above script named `SetLightDiffuse2.lua` is attached to the light object named "light1". In this case, string `"this"` in the `SetLightDiffuse` function will be equal to "light1". In our example, the function `SetLightDiffuse` sets the values of red, green and blue components of diffuse color of current light, which is "light1", to `0.25`, `0.5` and `0.75`, respectively.

4.391. SetLightScriptBoolVariable

Definition

`SetLightScriptBoolVariable(string lightName, string variable, bool value)`

Description

This function sets the value of the Boolean **`variable`** defined in the script attached to the **`lightName`** light object.

Parameters

`lightName`
Specifies the name of the light object.

`variable`
Specifies the name of the Boolean variable defined in the script attached to the **`lightName`** light.

`value`
Specifies the value of variable **`variable`** to be set.

Example

```
--script name is SetLightScriptBoolVariable.lua attached a to game object such as water

function Init()
    SetLightScriptBoolVariable("light1", "a", true)
end

function Update()

end
```

Assuming that the variable `"a"` is defined in the script attached to the light object `"light1"`, `SetLightScriptBoolVariable` function sets the value of `"a"` to *true*.

4.392. SetLightScriptDoubleVariable

Definition

`SetLightScriptDoubleVariable(string lightName, string variable, double value)`

Description

This function sets the value of the Double **`variable`** defined in the script attached to the **`lightName`** light object.

Parameters

`lightName`
Specifies the name of the light object.

`variable`
Specifies the name of the Double variable defined in the script attached to the **`lightName`** light.

`value`
Specifies the value of variable **`variable`** to be set.

Example

```
--script name is SetLightScriptDoubleVariable.lua attached a to game object such as water

function Init()
    SetLightScriptDoubleVariable("light1", "a", 1.0)
end

function Update()

end
```

Assuming that the variable `"a"` is defined in the script attached to the light object `"light1"`, `SetLightScriptDoubleVariable` function sets the value of `"a"` to *1.0*.

4.393. SetLightScriptIntVariable

Definition

`SetLightScriptIntVariable(string lightName, string variable, int value)`

Description

This function sets the value of the Integer **`variable`** defined in the script attached to the **`lightName`** light object.

Parameters

`lightName`
Specifies the name of the light object.

`variable`
Specifies the name of the Integer variable defined in the script attached to the **`lightName`** light.

`value`
Specifies the value of variable **`variable`**.

Example

```
--script name is SetLightScriptIntVariable.lua attached a to game object such as water

function Init()
    SetLightScriptIntVariable("light1", "a", 1)
end

function Update()

end
```

Assuming that the variable `"a"` is defined in the script attached to the light object `"light1"`, `SetLightScriptIntVariable` function sets the value of `"a"` to *1*.

4.394. SetLightScriptStringVariable

Definition

`SetLightScriptStringVariable(string lightName, string variable, string value)`

Description

This function sets the value of the String `variable` defined in the script attached to the `lightName` light object.

Parameters

`lightName`

Specifies the name of the light object.

`variable`

Specifies the name of the String variable defined in the script attached to the `lightName` light.

`value`

Specifies the value of variable `variable`.

Example

```
--script name is SetLightScriptStringVariable.lua attached a to game object such as water

function Init()
    SetLightScriptStringVariable("light1", "a", "hello")
end

function Update()

end
```

Assuming that the variable `"a"` is defined in the script attached to the light object `"light1"`, `SetLightScriptStringVariable` function sets the value of `"a"` to "*hello*".

4.395. SetLightShininess

Definition

`SetLightShininess(string lightObjectName, float shininess)`

Description

This function sets the shininess of light object `lightObjectName`.

Parameters

`lightObjectName`

Specifies the name of the light object. You can also use the name "this" for this parameter. In this case, "this" refers to the light object name to which this script is attached.

`shininess`

Specifies the shininess of light `lightObjectName`. This value must be greater than or equal to 0.0.

Example 1

```
function Init()
    SetLightShininess("light1", 100.0)
end

function Update()

end
```

In this example, the `SetLightShininess` function sets the shininess value of of light `"light1"` to `100.0`.

Example 2

```
--Name of script is SetLightShininess2.lua

function Init()
    SetLightShininess("this", 100.0)
end

function Update()

end
```

Assume that the above script named `SetLightShininess2.lua` is attached to the light object named "light1". In this case, string `"this"` in the `SetLightShininess` function will be equal to "light1". In our example, the function `SetLightShininess` sets the shininess value of current light (for example light "light1") to `100.0`.

4.396. SetLightSpecular

Definition

`SetLightSpecular(string lightObjectName, float red, float green, float blue)`

Description

This function sets the specular color of light `lightObjectName` as three values of red, green and blue. Each value ranges from 0.0 to 1.0.

Parameters

`lightObjectName`

Specifies the name of the light object. You can also use the name "this" for this parameter. In this case, "this" refers to the light object name to which this script is attached.

`red, green, blue`

Specify the specular color of light `lightObjectName` as three values of red, green and blue. Each value ranges from 0.0 to 1.0.

Example 1

```
function Init()
    SetLightSpecular("light1", 0.25, 0.5, 0.75)
end

function Update()

end
```

In this example, the `SetLightSpecular` function sets the value of the red, green, and blue components of the specular color of light `"light1"` to `0.25`, `0.5` and `0.75`, respectively.

Example 2

```
--Script name is SetLightSpecular2.lua

function Init()
    SetLightSpecular("this", 0.25, 0.5, 0.75)
end

function Update()

end
```

Assume that the above script named `SetLightSpecular2.lua` is attached to the light object named "light1". In this case, string `"this"` in the `SetLightSpecular` function will be equal to "light1". In our example, the function `SetLightSpecular` sets the values of red, green and blue components of specular color of current light, which is "light1", to `0.25`, `0.5` and `0.75`, respectively.

4.397. SetMainCharacterScriptBoolVariable

Definition

`SetMainCharacterScriptBoolVariable(string variable, bool value)`

Description

This function sets the value of the Boolean **`variable`** defined in the script attached to the main character object.

Parameters

`variable`

Specifies the name of the Boolean variable defined in the script attached to the main character.

`value`

Specifies the value of the Boolean **`variable`** to be set.

Example

```
--script name is SetMainCharacterScriptBoolVariable.lua attached a to game object such as
water

function Init()
    SetMainCharacterScriptBoolVariable("a", true)
end

function Update()

end
```

Assuming that the variable **`"a"`** is defined in the script attached to the main character object, `SetMainCharacterScriptBoolVariable` function sets the value of **`"a"`** to *true*.

4.398. SetMainCharacterScriptDoubleVariable

Definition

`SetMainCharacterScriptDoubleVariable(string variable, double value)`

Description

This function sets the value of the Double **`variable`** defined in the script attached to the main character object.

Parameters

`variable`

Specifies the name of the Double variable defined in the script attached to the main character.

`value`

Specifies the value of the Double **`variable`** to be set.

Example

```
--script name is SetMainCharacterScriptDoubleVariable.lua attached a to game object such
as water

function Init()
    SetMainCharacterScriptDoubleVariable("a", 1.0)
end

function Update()

end
```

Assuming that the variable **`"a"`** is defined in the script attached to the main character object, **`SetMainCharacterScriptDoubleVariable`** function sets the value of **`"a"`** to *1.0*.

4.399. SetMainCharacterScriptIntVariable

Definition

`SetMainCharacterScriptIntVariable(string variable, int value)`

Description

This function sets the value of the Integer **`variable`** defined in the script attached to the main character object.

Parameters

`variable`

Specifies the name of the Integer variable defined in the script attached to the main character.

`value`

Specifies the value of the Integer **`variable`** to be set.

Example

```
--script name is SetMainCharacterScriptIntVariable.lua attached a to game object such as
water

function Init()
    SetMainCharacterScriptIntVariable("a", 1)
end

function Update()

end
```

Assuming that the variable **`"a"`** is defined in the script attached to the main character, `SetMainCharacterScriptIntVariable` function sets the value of **`"a"`** to *1*.

4.400. SetMainCharacterScriptStringVariable

Definition

`SetMainCharacterScriptStringVariable(string variable, string value)`

Description

This function sets the value of the String **`variable`** defined in the script attached to the main character object.

Parameters

`variable`

Specifies the name of the String variable defined in the script attached to the main character.

`value`

Specifies the value of the String **`variable`** to be set.

Example

```
--script name is SetMainCharacterScriptStringVariable.lua attached a to game object such
as water

function Init()
    SetMainCharacterScriptStringVariable("a", "hello")
end

function Update()

end
```

Assuming that the variable **`"a"`** is defined in the script attached to the main character object, `SetMainCharacterScriptStringVariable` function sets the value of **`"a"`** to "*hello*".

4.401. SetMenuCursorSize

Definition

```
SetMenuCursorSize(int cursorSize)
```

Description

This function sets the menu cursor size as an integer number.

Parameter

`cursorSize`

Specifies the menu cursor size to be set. This value must be greater than 0.

Example

```
function Init()
    SetMenuCursorSize(8)
end

function Update()

end
```

This script sets the menu cursor size to 8.

4.402. SetMultisamplingValue

Definition

`SetMultisamplingValue(int numSamples)`

Description

This function sets the value of multisampling.

Parameter

`numSamples`

Specifies the value of multisampling to be set. Accepted values are 0, 2, 4, 8 and 16. A value of 0 will disable multisampling.

Example

```
function Init()
    SetMultisamplingValue(2)
end

function Update()

end
```

This script sets the multisampling value to **2**.

4.403. SetPhysicsCameraAngle

Definition

`SetPhysicsCameraAngle(float angleDegree)`

Description

This function sets the angle of physics camera attached to the main character.

Parameter

`angleDegree`

Specifies the angle of physics camera attached to the main character in degrees.

Example

```
function Init()
    SetPhysicsCameraAngle(27.5)
end

function Update()

end
```

This scripts sets the angle of physics camera attached to the main character to `27.5` degrees.

4.404. SetPhysicsCameraFarClipPlane

Definition

`SetPhysicsCameraFarClipPlane(float fcp)`

Description

This function sets the far clip plane of physics camera attached to the main character to `fcp`.

Parameter

fcp

Specifies the far clip plane value of of physics camera attached to the main character. This value must be greater than 0.0.

Example

```
function Init()
    SetPhysicsCameraFarClipPlane(10.25)
end

function Update()

end
```

In this script, we set the far clip plane value of the physics camera attached to the main character to `10.25`.

4.405. SetPhysicsCameraMaxTilt

Definition

`SetPhysicsCameraMaxTilt(float maxTiltDegree)`

Description

This function sets the maximum tilt of physics camera attached to the main character. It should be noted that the tilt of the physics camera never exceeds this value.

Parameter

`maxTiltDegree`

Specifies the maximum tilt of physics camera attached to the main character in degrees.

Example

```
function Init()
    SetPhysicsCameraMaxTilt(57.5)
end

function Update()

end
```

This scripts sets the maximum tilt of physics camera attached to the main character to **`57.5`** degrees.

4.406. SetPhysicsCameraMinTilt

Definition

`SetPhysicsCameraMinTilt(float minTiltDegree)`

Description

This function sets the minimum tilt of physics camera attached to the main character. It should be noted that the tilt of the physics camera is never less than this value.

Parameter

`minTiltDegree`

Specifies the minimum tilt of physics camera attached to the main character in degrees.

Example

```
function Init()
    SetPhysicsCameraMinTilt(-57.5)
end

function Update()

end
```

This scripts sets the minimum tilt of physics camera attached to the main character to **`-57.5`** degrees.

4.407. SetPhysicsCameraNearClipPlane

Definition

`SetPhysicsCameraNearClipPlane(float ncp)`

Description

This function sets the near clip plane of physics camera attached to the main character to **`ncp`**.

Parameter

ncp

Specifies the near clip plane value of of physics camera attached to the main character. This value must be greater than 0.0.

Example

```
function Init()
    SetPhysicsCameraNearClipPlane(2.5)
end

function Update()

end
```

In this script, we set the near clip plane value of the physics camera attached to the main character to `2.5`.

4.408. SetPhysicsCameraTilt

Definition

`SetPhysicsCameraTilt(float tiltDegree)`

Description

This function sets the current tilt value of the physics camera attached to the main character.

Parameter

`tiltDegree`

Specifies the current tilt value of the physics camera attached to the main character in degrees.

Example

```
function Init()
    SetPhysicsCameraTilt(15.5)
end

function Update()

end
```

This scripts sets the current tilt of physics camera attached to the main character to `15.5` degrees.

4.409. SetPhysicsCameraYaw

Definition

`SetPhysicsCameraYaw(float yawDegree)`

Description

This function sets the current yaw value of the physics camera attached to the main character.

Parameter

`yawDegree`

Specifies the current yaw value of the physics camera attached to the main character in degrees.

Example

```
function Init()
    SetPhysicsCameraYaw(150.5)
end

function Update()

end
```

This scripts sets the current yaw of physics camera attached to the main character to `150.5` degrees.

4.410. SetPhysicsCollisionFlags

Definition

`SetPhysicsCollisionFlags(string group1, string group2, bool flag)`

Description

Each physics actor in Vanda engine belongs to a specific group. For example, a dynamic physics actor belongs to the "DYNAMIC" group, while a static physics actor belongs to the "STATIC" group. With this function one can set whether collisions should be detected between physics actors belonging to a given pair of groups at runtime. You can also use the Tools > Current VScene Properties menu to enable/disable collision detection between physics actors belonging to a given pair of groups. Initially all pair of physics groups except (Trigger vs. Ground Plane) pair are enabled, meaning that collision detection happens between all physics actors except (Trigger vs. Ground Plane).
Collision detection between two physics actors a and b occurs if:
`SetPhysicsCollisionFlags(a->group`, `b->group, true`).

Parameters

group1
Specifies the first group. The following group types are supported:

"KINEMATIC"
Kinematic is a dynamic actor that can ignore some rules of physics, and its rotation and translation is controlled by prefab instance.

"DYNAMIC"
A dynamic actor has its position and rotation updated by the physics simulation and controls the translation and rotation of its prefab instance.

"TRIGGER"
Triggers allow colliders to perform overlap tests.

"STATIC"
Static actor is immovable by the physics simulation.

"GROUND"
Default physics ground plane.

group2
Specifies the second group. The supported groups are similar to the ***group1*** description.

flag
This boolean value specifies whether collisions should be detected between physics actors belonging to a given pair of groups. Accepted values are `true` and `false`. The `true` value means that collision detection between two physics actors a and b belonging to ***group1*** and ***group2*** occurs.

Example 1

```
function Init()
    SetPhysicsCollisionFlags("DYNAMIC", "DYNAMIC", false)
```

```
end

function Update()

end
```

In this case, collision detection is disabled for all dynamic physics actors.

Example 2

```
function Init()
    SetPhysicsCollisionFlags("DYNAMIC", "STATIC", false)
end

function Update()

end
```

In this case, collision detection between dynamic and static physics actors is disabled.

Example 3

```
function Init()
    SetPhysicsCollisionFlags("DYNAMIC", "KINEMATIC", true)
end

function Update()

end
```

In this case, collision detection between dynamic and kinematic physics actors is enabled.

4.411. SetPhysicsDefaultDynamicFriction

Definition

`SetPhysicsDefaultDynamicFriction(float dynamicFriction)`

Description

This function sets the physics default dynamic friction to **`dynamicFriction`**.

Parameter

`dynamicFriction`

Specifies the value of physics default dynamic friction to be set. This value must be equal to or greater than 0.0.

Example

```
function Init()
    SetPhysicsDefaultDynamicFriction(0.1)
end

function Update()

end
```

This scripts sets the physics default dynamic friction to `0.1`.

4.412. SetPhysicsDefaultRestitution

Definition

`SetPhysicsDefaultRestitution(float restitution)`

Description

This function sets the physics default restitution to `restitution`.

Parameter

`restitution`

Specifies the value of physics default restitution to be set. This value must be in the range [0.0,1.0]

Example

```
function Init()
    SetPhysicsDefaultRestitution(0.8)
end

function Update()

end
```

This scripts sets the physics default restitution to `0.8`.

4.413. SetPhysicsDefaultSkinWidth

Definition

`SetPhysicsDefaultSkinWidth(float skinWidth)`

Description

This function sets the physics default skin width to `skinWidth`.

Parameter

`skinWidth`

Specifies the value of physics default skin width to be set. This value must be greater than 0.0.

Example

```
function Init()
    SetPhysicsDefaultSkinWidth(0.3)
end

function Update()

end
```

This scripts sets the physics default skin width to `0.3`.

4.414. SetPhysicsDefaultStaticFriction

Definition

`SetPhysicsDefaultStaticFriction(float staticFriction)`

Description

This function sets the physics default static friction to `staticFriction`.

Parameter

`staticFriction`

Specifies the value of physics default static friction to be set. This value must be equal to or greater than 0.0.

Example

```
function Init()
    SetPhysicsDefaultStaticFriction(0.1)
end

function Update()

end
```

This scripts sets the physics default static friction to `0.1`.

4.415. SetPhysicsGravity

Definition

`SetPhysicsGravity(float x, float y, float z)`

Description

This function sets the X, Y and Z components of physics gravity.

Parameter

`x, y, z`

Specify the X, Y and Z components of physics gravity.

Example

```
function Init()
    SetPhysicsGravity(-1.3, -6.8, -1.1)
end

function Update()

end
```

This scripts sets the X, Y and Z components of physics gravity to `-1.3`, `-6.8` and `-1.1`, respectively.

4.416. SetPhysicsGroundHeight

Definition

`SetPhysicsGroundHeight(float height)`

Description

This function sets the value of physics ground height.

Parameter

`height`

Specifies the value of physics ground height to be set.

Example

```
function Init()
    SetPhysicsGroundHeight(-2.1)
end

function Update()

end
```

This script sets the physics ground height to `-2.1`.

4.417. SetPrefabInstanceAmbient

Definition

`SetPrefabInstanceAmbient(string prefabInstanceName, float red, float green, float blue)`

Description

This function sets the ambient color of prefab instance **`prefabInstanceName`**. In order for this function to change the ambient color of prefab instance, you must enable the material of prefab instance **`prefabInstanceName`**. For this purpose, you can click on the prefab instance **`prefabInstanceName`** in the *Prefabs and GUIs* section of Vanda Engine editor and click the *Edit* button to activate the *Enable Prefab Instance Material* option in the dialog that appears. You can also use the *EnablePrefabInstanceMaterial* function to enable the prefab instance material at runtime. In this case, prefab instance material is used instead of its prefab material.

Parameters

`prefabInstanceName`
Specifies the name of the prefab instance. You can also use the name "this" for this parameter. In this case, "this" refers to the prefab instance that this script is attached to.

`red, green, blue`
Specify the red, green, and blue components of prefab instance ambient color. Each value is in the range [0.0,1.0].

Example 1

```
function Init()
    EnablePrefabInstanceMaterial("1_VandaEngine17-SamplePack1_f1_barrel")
    SetPrefabInstanceAmbient("1_VandaEngine17-SamplePack1_f1_barrel", 0.75, 0.5, 0.25)
end

function Update()

end
```

First we enable the material of prefab instance `"1_VandaEngine17-SamplePack1_f1_barrel"`. Then we set the ambient color of prefab instance `"1_VandaEngine17-SamplePack1_f1_barrel"` to `(0.75, 0.5, 0.25)`.

Example 2

```
--Script name is SetPrefabInstanceAmbient2.lua

function Init()
    EnablePrefabInstanceMaterial("this")
    SetPrefabInstanceAmbient("this", 0.75, 0.5, 0.25)
end

function Update()

end
```

`If, in the Prefab Editor, you attach SetPrefabInstanceAmbient2.lua script to a Prefab, then "this" parameter in the SetPrefabInstanceAmbient function will point to instances of that Prefab in current VScene. For example, if you have an Instance named` *`instance1_a`* `from a Prefab named` *`a`* `to which this script is attached, "this" in SetPrefabInstanceAmbient function refers to the name` *`instance1_a`*`.`
In this example, we enable the material of current prefab instance (for example, *`instance1_a`*).
Then we set the ambient color of current prefab instance (for example, *`instance1_a`*) to **`(0.75, 0.5, 0.25)`**.

4.418. SetPrefabInstanceDiffuse

Definition

`SetPrefabInstanceDiffuse(string prefabInstanceName, float red, float green, float blue)`

Description

This function sets the diffuse color of prefab instance **`prefabInstanceName`**. In order for this function to change the diffuse color of prefab instance, you must enable the material of prefab instance **`prefabInstanceName`**. For this purpose, you can click on the prefab instance **`prefabInstanceName`** in the *Prefabs and GUIs* section of Vanda Engine editor and click the *Edit* button to activate the *Enable Prefab Instance Material* option in the dialog that appears. You can also use the *EnablePrefabInstanceMaterial* function to enable the prefab instance material at runtime. In this case, prefab instance material is used instead of its prefab material.

Parameters

`prefabInstanceName`

Specifies the name of the prefab instance. You can also use the name "this" for this parameter. In this case, "this" refers to the prefab instance that this script is attached to.

`red, green, blue`

Specify the red, green, and blue components of prefab instance diffuse color. Each value is in the range [0.0,1.0].

Example 1

```lua
function Init()
    EnablePrefabInstanceMaterial("1_VandaEngine17-SamplePack1_f1_barrel")
    SetPrefabInstanceDiffuse("1_VandaEngine17-SamplePack1_f1_barrel", 0.75, 0.5, 0.25)
end

function Update()

end
```

First we enable the material of prefab instance `"1_VandaEngine17-SamplePack1_f1_barrel"`. Then we set the diffuse color of prefab instance `"1_VandaEngine17-SamplePack1_f1_barrel"` to `(0.75, 0.5, 0.25)`.

Example 2

```lua
--Script name is SetPrefabInstanceDiffuse2.lua

function Init()
    EnablePrefabInstanceMaterial("this")
    SetPrefabInstanceDiffuse("this", 0.75, 0.5, 0.25)
end

function Update()

end
```

`If, in the Prefab Editor, you attach SetPrefabInstanceDiffuse2.lua script to a Prefab, then "this" parameter in the SetPrefabInstanceDiffuse function will point to instances of that Prefab in current VScene. For example, if you have an Instance named` *`instance1_a`* `from a Prefab named` *`a`* `to which this script is attached, "this" in SetPrefabInstanceDiffuse function refers to the name` *`instance1_a`*`.`

In this example, we enable the material of current prefab instance (for example, *`instance1_a`*).
Then we set the diffuse color of current prefab instance (for example, *`instance1_a`*) to **`(0.75, 0.5, 0.25)`**.

4.419. SetPrefabInstanceEmission

Definition

`SetPrefabInstanceEmission(string prefabInstanceName, float red, float green, float blue)`

Description

This function sets the emission color of prefab instance **`prefabInstanceName`**. In order for this function to change the emission color of prefab instance, you must enable the material of prefab instance **`prefabInstanceName`**. For this purpose, you can click on the prefab instance **`prefabInstanceName`** in the *Prefabs and GUIs* section of Vanda Engine editor and click the *Edit* button to activate the *Enable Prefab Instance Material* option in the dialog that appears. You can also use the *EnablePrefabInstanceMaterial* function to enable the prefab instance material at runtime. In this case, prefab instance material is used instead of its prefab material.

Parameters

`prefabInstanceName`

Specifies the name of the prefab instance. You can also use the name "this" for this parameter. In this case, "this" refers to the prefab instance that this script is attached to.

`red, green, blue`

Specify the red, green, and blue components of prefab instance emission color. Each value is in the range [0.0,1.0].

Example 1

```
function Init()
    EnablePrefabInstanceMaterial("1_VandaEngine17-SamplePack1_f1_barrel")
    SetPrefabInstanceEmission("1_VandaEngine17-SamplePack1_f1_barrel", 0.75, 0.5, 0.25)
end

function Update()

end
```

First we enable the material of prefab instance `"1_VandaEngine17-SamplePack1_f1_barrel"`. Then we set the emission color of prefab instance `"1_VandaEngine17-SamplePack1_f1_barrel"` to `(0.75, 0.5, 0.25)`.

Example 2

```
--Script name is SetPrefabInstanceEmission2.lua

function Init()
    EnablePrefabInstanceMaterial("this")
    SetPrefabInstanceEmission("this", 0.75, 0.5, 0.25)
end

function Update()

end
```

If, in the Prefab Editor, you attach `SetPrefabInstanceEmission2.lua` script to a Prefab, then `"this"` parameter in the `SetPrefabInstanceEmission` function will point to instances of that Prefab in current VScene. For example, if you have an Instance named *instance1_a* from a Prefab named *a* to which this script is attached, `"this"` in `SetPrefabInstanceEmission` function refers to the name *instance1_a*.

In this example, we enable the material of current prefab instance (for example, *instance1_a*). Then we set the emission color of current prefab instance (for example, *instance1_a*) to **`(0.75, 0.5, 0.25)`**.

4.420. SetPrefabInstanceScriptBoolVariable

Definition

`SetPrefabInstanceScriptBoolVariable(string prefabInstanceName, string variable, bool value)`

Description

This function sets the value of the Boolean **`variable`** defined in the script attached to the prefab instance `PrefabInstanceName`.

Parameters

`PrefabInstanceName`
Specifies the name of the prefab instance.

`variable`
Specifies the name of the Boolean variable defined in the script attached to the prefab instance *`PrefabInstanceName`*.

`value`
Specifies the value of the Boolean **`variable`** to be set.

Example

```
--script name is SetPrefabInstanceScriptBoolVariable.lua attached a to game object such as water

function Init()
    SetPrefabInstanceScriptBoolVariable("1_VandaEngine17-SamplePack1_birdcage", "a", true)
end

function Update()

end
```

Assuming that the variable **`"a"`** is defined in the script attached to the prefab instance `"1_VandaEngine17-SamplePack1_birdcage"`, `SetPrefabInstanceScriptBoolVariable` function sets the value of **`"a"`** to *true*.

4.421. SetPrefabInstanceScriptDoubleVariable

Definition

`SetPrefabInstanceScriptDoubleVariable(string prefabInstanceName, string variable, double value)`

Description

This function sets the value of the Double **`variable`** defined in the script attached to the prefab instance `PrefabInstanceName`.

Parameters

`PrefabInstanceName`
Specifies the name of the prefab instance.

`variable`
Specifies the name of the Double variable defined in the script attached to the prefab instance *`PrefabInstanceName`*.

`value`
Specifies the value of the Double **`variable`** to be set.

Example

```
--script name is SetPrefabInstanceScriptDoubleVariable.lua attached a to game object such
as water

function Init()
    SetPrefabInstanceScriptDoubleVariable("1_VandaEngine17-SamplePack1_birdcage", "a",
1.0)
end

function Update()

end
```

Assuming that the variable `"a"` is defined in the script attached to the prefab instance `"1_VandaEngine17-SamplePack1_birdcage"`, `SetPrefabInstanceScriptDoubleVariable` function sets the value of `"a"` to *1.0*.

4.422. SetPrefabInstanceScriptIntVariable

Definition

`SetPrefabInstanceScriptIntVariable(string prefabInstanceName, string variable, int value)`

Description

This function sets the value of the Integer `variable` defined in the script attached to the prefab instance `PrefabInstanceName`.

Parameters

PrefabInstanceName
Specifies the name of the prefab instance.

variable
Specifies the name of the Integer variable defined in the script attached to the prefab instance `PrefabInstanceName`.

value
Specifies the value of the Integer `variable` to be set.

Example

```
--script name is SetPrefabInstanceScriptIntVariable.lua attached a to game object such as
water

function Init()
    SetPrefabInstanceScriptIntVariable("1_VandaEngine17-SamplePack1_birdcage", "a", 1)
end

function Update()

end
```

Assuming that the variable `"a"` is defined in the script attached to the prefab instance `"1_VandaEngine17-SamplePack1_birdcage"`, `SetPrefabInstanceScriptIntVariable` function sets the value of `"a"` to *1*.

4.423. SetPrefabInstanceScriptStringVariable

Definition

`SetPrefabInstanceScriptStringVariable(string prefabInstanceName, string variable, string value)`

Description

This function sets the value of the String **`variable`** defined in the script attached to the prefab instance `PrefabInstanceName`.

Parameters

`PrefabInstanceName`
Specifies the name of the prefab instance.

`variable`
Specifies the name of the String variable defined in the script attached to the prefab instance `PrefabInstanceName`.

`value`
Specifies the value of the String **`variable`** to be set.

Example

```
--script name is SetPrefabInstanceScriptStringVariable.lua attached a to game object such
as water

function Init()
    SetPrefabInstanceScriptStringVariable("1_VandaEngine17-SamplePack1_birdcage", "a",
"hello")
end

function Update()

end
```

Assuming that the variable `"a"` is defined in the script attached to the prefab instance `"1_VandaEngine17-SamplePack1_birdcage"`, `SetPrefabInstanceScriptStringVariable` function sets the value of `"a"` to "*hello*".

4.424. SetPrefabInstanceShininess

Definition

`SetPrefabInstanceShininess(string prefabInstanceName, float shininess)`

Description

This function sets the shininess of prefab instance **`prefabInstanceName`**. In order for this function to change the shininess of prefab instance, you must enable the material of prefab instance **`prefabInstanceName`**. For this purpose, you can click on the prefab instance **`prefabInstanceName`** in the *Prefabs and GUIs* section of Vanda Engine editor and click the *Edit* button to activate the *Enable Prefab Instance Material* option in the dialog that appears. You can also use the *EnablePrefabInstanceMaterial* function to enable the prefab instance material at runtime. In this case, prefab instance material is used instead of its prefab material.

Parameters

`prefabInstanceName`
Specifies the name of the prefab instance. You can also use the name "this" for this parameter. In this case, "this" refers to the prefab instance that this script is attached to.

`shininess`
Specifies the shininess of prefab instance. This value must be greater than or equal to 0.0.

Example 1

```
function Init()
    EnablePrefabInstanceMaterial("1_VandaEngine17-SamplePack1_f1_barrel")
    SetPrefabInstanceShininess("1_VandaEngine17-SamplePack1_f1_barrel", 20.0)
end

function Update()

end
```

First we enable the material of prefab instance `"1_VandaEngine17-SamplePack1_f1_barrel"`. Then we set the shininess of prefab instance `"1_VandaEngine17-SamplePack1_f1_barrel"` to `20.0`.

Example 2

```
--Script name is SetPrefabInstanceShininess2.lua

function Init()
    EnablePrefabInstanceMaterial("this")
    SetPrefabInstanceShininess("this", 20.0)
end

function Update()

end
```

`If, in the Prefab Editor, you attach SetPrefabInstanceShininess2.lua script to a Prefab, then "this" parameter in the SetPrefabInstanceShininess function will point to instances of that Prefab in current VScene. For example, if you have an Instance named` *`instance1_a`* `from a Prefab named` *`a`* `to which this script is attached, "this" in SetPrefabInstanceShininess function refers to the name` *`instance1_a`*`.`

In this example, we enable the material of current prefab instance (for example, *`instance1_a`*).
Then we set the shininess of current prefab instance (for example, *`instance1_a`*) to `20.0`.

4.425. SetPrefabInstanceSpecular

Definition

`SetPrefabInstanceSpecular(string prefabInstanceName, float red, float green, float blue)`

Description

This function sets the specular color of prefab instance **`prefabInstanceName`**. In order for this function to change the specular color of prefab instance, you must enable the material of prefab instance **`prefabInstanceName`**. For this purpose, you can click on the prefab instance **`prefabInstanceName`** in the *Prefabs and GUIs* section of Vanda Engine editor and click the *Edit* button to activate the *Enable Prefab Instance Material* option in the dialog that appears. You can also use the *EnablePrefabInstanceMaterial* function to enable the prefab instance material at runtime. In this case, prefab instance material is used instead of its prefab material.

Parameters

`prefabInstanceName`

Specifies the name of the prefab instance. You can also use the name "this" for this parameter. In this case, "this" refers to the prefab instance that this script is attached to.

`red, green, blue`

Specify the red, green, and blue components of prefab instance specular color. Each value is in the range [0.0,1.0].

Example 1

```
function Init()
    EnablePrefabInstanceMaterial("1_VandaEngine17-SamplePack1_f1_barrel")
    SetPrefabInstanceSpecular("1_VandaEngine17-SamplePack1_f1_barrel", 0.75, 0.5, 0.25)
end

function Update()

end
```

First we enable the material of prefab instance `"1_VandaEngine17-SamplePack1_f1_barrel"`. Then we set the specular color of prefab instance `"1_VandaEngine17-SamplePack1_f1_barrel"` to `(0.75, 0.5, 0.25)`.

Example 2

```
--Script name is SetPrefabInstanceSpecular2.lua

function Init()
    EnablePrefabInstanceMaterial("this")
    SetPrefabInstanceSpecular("this", 0.75, 0.5, 0.25)
end

function Update()

end
```

`If, in the Prefab Editor, you attach SetPrefabInstanceSpecular2.lua script to a Prefab, then "this" parameter in the SetPrefabInstanceSpecular function will point to instances of that Prefab in current VScene. For example, if you have an Instance named` *`instance1_a`* `from a Prefab named` *`a`* `to which this script is attached, "this" in SetPrefabInstanceSpecular function refers to the name` *`instance1_a`*`.`
In this example, we enable the material of current prefab instance (for example, *`instance1_a`*).
Then we set the specular color of current prefab instance (for example, *`instance1_a`*) to **`(0.75, 0.5, 0.25)`**.

4.426. SetPrefabInstanceTransparency

Definition

```
SetPrefabInstanceTransparency(string prefabInstanceName, float
transparency)
```

Description

This function sets the transparency of prefab instance **prefabInstanceName**. In order for this function to change the transparency of prefab instance, you must enable the material of prefab instance **prefabInstanceName**. For this purpose, you can click on the prefab instance **prefabInstanceName** in the *Prefabs and GUIs* section of Vanda Engine editor and click the *Edit* button to activate the *Enable Prefab Instance Material* option in the dialog that appears. You can also use the *EnablePrefabInstanceMaterial* function to enable the prefab instance material at runtime. In this case, prefab instance material is used instead of its prefab material.

Parameters

prefabInstanceName

Specifies the name of the prefab instance. You can also use the name "this" for this parameter. In this case, "this" refers to the prefab instance that this script is attached to.

transparency

Specifies the transparency of prefab instance. This value must be in the range [0.0,1.0].

Example 1

```
function Init()
    EnablePrefabInstanceMaterial("1_VandaEngine17-SamplePack1_f1_barrel")
    SetPrefabInstanceTransparency("1_VandaEngine17-SamplePack1_f1_barrel", 0.5)
end

function Update()

end
```

First we enable the material of prefab instance "1_VandaEngine17-SamplePack1_f1_barrel". Then we set the transparency of prefab instance "1_VandaEngine17-SamplePack1_f1_barrel" to 0.5.

Example 2

```
--Script name is SetPrefabInstanceTransparency2.lua

function Init()
    EnablePrefabInstanceMaterial("this")
    SetPrefabInstanceTransparency("this", 0.5)
end

function Update()

end
```

If, in the Prefab Editor, you attach `SetPrefabInstanceTransparency2.lua` script to a Prefab, then `"this"` parameter in the `SetPrefabInstanceTransparency` function will point to instances of that Prefab in current VScene. For example, if you have an Instance named *instance1_a* from a Prefab named *a* to which this script is attached, `"this"` in `SetPrefabInstanceTransparency` function refers to the name *instance1_a*.
In this example, we enable the material of current prefab instance (for example, *instance1_a*).
Then we set the transparency of current prefab instance (for example, *instance1_a*) to `0.5`.

4.427. SetScreenResolution

Definition

`SetScreenResolution(int screenWidth)`

Description

When running the game, you can select the resolution from the dialog that appears at the beginning of the game. You can also set the resolution of the screen at runtime using `SetScreenResolution` function.

Parameter

`screenWidth`

Specifies the width of the screen resolution in pixels. Acceptable values are:

- **0** : Current screen resolution is selected.
- **800** : 800 X 600
- **1024** : 1024 X 768
- **1280** : 1280 X 720
- **1920** : 1920 X 1080
- **2560** : 2560 X 1440
- **3840** : 3840 X 2160
- **7680**: 7680 X 4320

Example

```
function Init()
    SetScreenResolution(1920)
end

function Update()

end
```

This script sets the screen resolution to 1920 X 1080.

4.428. SetSelectionDistance

Definition

`SetSelectionDistance(float selectionDistance)`

Description

This function sets the maximum distance from the camera that you can select a prefab instance using the *SelectPrefabInstances* function.

Parameter

`selectionDistance`

Sets the maximum distance from the camera that you can select a prefab instance using the *SelectPrefabInstances* function. This value must be greater than 0.0.

Example

```
function Init()
    LoadResource("images", "cursor.dds")
    ShowCursorIcon("images_cursor.dds", 5.0)
    SetSelectionDistance(5.5)
end

function Update()
    if IsKeyDown("0") then
        SelectPrefabInstances(GetCursorX(), GetCursorY(), 20.0, 20.0)
    end
end
```

First, in the `Init()` event, we load and show resource cursor icon `"cursor.dds"` located in `"images"` folder (In order for `LoadResource` function to load the desired resource, you must first add it through the *Add Resource to Current Project* dialog (File > Project > Add/Remove Resource to/from Current Project). Then we set the maximum distance for selection to `5.5` using the `SetSelectionDistance` function. Then, in the `Update()` event, in the `SelectPrefabInstances` function, we set the center of the selection to the mouse position using `GetCursorX()` and `GetCursorY()` functions and set the length and width of the selection to `20.0`. Whenever the user left-clicks, the `SelectPrefabInstances` function is called. If the prefab instance is in the selection region and its distance from camera is less than 5.5 units, it is selected and its `Onselect()` event is called.

4.429. SetSkyPosition

Definition

`SetSkyPosition(float x, float y, float z)`

Description

This function sets the sky position.

Parameter

`x, y, z`

Specify the X, Y and Z components of sky position.

Example 1

```
function Init()
    SetSkyPosition(2.5, 5.0, 7.5)
end

function Update()

end
```

This script sets the X, Y and Z components of sky position to `2.5`, `5.0` and `7.5`, respectively.

4.430. SetSkyScriptBoolVariable

Definition

`SetSkyScriptBoolVariable(string variable, bool value)`

Description

This function sets the value of the Boolean **`variable`** defined in the script attached to the sky object.

Parameters

`variable`
Specifies the name of the Boolean variable defined in the script attached to the sky object.

`value`
Specifies the value of the Boolean **`variable`** to be set.

Example

```
--script name is SetSkyScriptBoolVariable.lua attached a to game object such as water

function Init()
    SetSkyScriptBoolVariable("a", true)
end

function Update()

end
```

Assuming that the variable `"a"` is defined in the script attached to the sky object, `SetSkyScriptBoolVariable` function sets the value of `"a"` to *true*.

4.431. SetSkyScriptDoubleVariable

Definition

`SetSkyScriptDoubleVariable(string variable, double value)`

Description

This function sets the value of the Double **`variable`** defined in the script attached to the sky object.

Parameters

`variable`

Specifies the name of the Double variable defined in the script attached to the sky object.

`value`

Specifies the value of the Double **`variable`** to be set.

Example

```
--script name is SetSkyScriptDoubleVariable.lua attached a to game object such as water

function Init()
    SetSkyScriptDoubleVariable("a", 1.0)
end

function Update()

end
```

Assuming that the variable **`"a"`** is defined in the script attached to the sky object, `SetSkyScriptDoubleVariable` function sets the value of **`"a"`** to *1.0*.

4.432. SetSkyScriptIntVariable

Definition

`SetSkyScriptIntVariable(string variable, int value)`

Description

This function sets the value of the Integer **`variable`** defined in the script attached to the sky object.

Parameters

`variable`
Specifies the name of the Integer variable defined in the script attached to the sky object.

`value`
Specifies the value of the Integer **`variable`** to be set.

Example

```lua
--script name is SetSkyScriptIntVariable.lua attached a to game object such as water

function Init()
    SetSkyScriptIntVariable("a", 1)
end

function Update()

end
```

Assuming that the variable `"a"` is defined in the script attached to the sky, `SetSkyScriptIntVariable` function sets the value of `"a"` to *1*.

4.433. SetSkyScriptStringVariable

Definition

`SetSkyScriptStringVariable(string variable, string value)`

Description

This function sets the value of the String **`variable`** defined in the script attached to the sky object.

Parameters

`variable`

Specifies the name of the String variable defined in the script attached to the sky object.

`value`

Specifies the value of the String **`variable`** to be set.

Example

```
--script name is SetSkyScriptStringVariable.lua attached a to game object such as water

function Init()
    SetSkyScriptStringVariable("a", "hello")
end

function Update()

end
```

Assuming that the variable **`"a"`** is defined in the script attached to the sky object, `SetSkyScriptStringVariable` function sets the value of **`"a"`** to "*hello*".

4.434. SetSoundLoop

Definition

`SetSoundLoop(string soundObjectName, bool loop)`

Description

This function sets the loop state of the sound **`soundObjectName`** to true or false.

Parameters

`soundObjectName`

Specifies the ambient or 3D sound name. You can also use the name "this" for this parameter. In this case, "this" string refers to the name of the sound to which this script is attached.

`loop`

Specifies the state of the sound loop. Accepted values are *true* or *false*.

Example 1

```lua
function Init()
    SetSoundLoop("sound1", false)
    PlaySound("sound1")
end

function Update()

end
```

First, we set the loop status of `"sound1"` to `false`. Then we play `"sound1"`. Since the loop status of sound `"sound1"` is `false`, this sound will only be played once.

Example 2

```lua
--Name of script is SetSoundLoop2.lua

function Init()
    SetSoundLoop("this", true)
    PlaySound("this")
end

function Update()

end
```

Assume that the above script named `SetSoundLoop2.lua` is attached to a sound object named "sound1". In this case, string `"this"` in the `SetSoundLoop` function will be equal to "sound1". In our example, we set the loop state of current sound, which is "sound1", to `true`. Then we play current sound, which is "sound1". Since the loop status of current sound is `true`, this sound will be played continuously.

4.435. SetSoundMaxDistance

Definition

`SetSoundMaxDistance(string 3DSoundObjectName, float maxDistance)`

Description

This function sets the maximum distance of 3D sound `3DSoundObjectName` to `maxDistance`.

Parameters

`3DSoundObjectName`

Specifies the 3D sound name. You can also use the name "this" for this parameter. In this case, "this" string refers to the name of the 3D sound to which this script is attached.

`maxDistance`

Specifies the maximum distance of 3D sound `3DSoundObjectName` to be set. This value must be greater than or equal to 0.0.

Example 1

```
function Init()
    SetSoundMaxDistance("sound1", 1.7)
end

function Update()

end
```

This script sets the maximum distance of 3D sound `"sound1"` to `1.7`.

Example 2

```
--Name of script is SetSoundMaxDistance2.lua

function Init()
    SetSoundMaxDistance("this", 2.5)
end

function Update()

end
```

Assume that the above script named `SetSoundMaxDistance2.lua` is attached to a 3D sound object named "sound1". In this case, string `"this"` in the `SetSoundMaxDistance` function will be equal to "sound1". In our example, the function `SetSoundMaxDistance` sets the maximum distance of current 3D sound, which is "sound1", to `2.5`.

4.436. SetSoundPitch

Definition

`SetSoundPitch(string soundObjectName, float pitch)`

Description

This function sets the pitch of ambient or 3D sound **`soundObjectName`** to **`pitch`**.

Parameters

`soundObjectName`

Specifies the ambient or 3D sound name. You can also use the name "this" for this parameter. In this case, "this" string refers to the name of the sound to which this script is attached.

`pitch`

pitch of 3D or ambient sound **`soundObjectName`**. This value must be greater than 0.0.

Example 1

```
function Init()
    SetSoundPitch("sound1", 1.5)
end

function Update()

end
```

This script sets the pitch of sound `"sound1"` to `1.5`.

Example 2

```
--Name of script is SetSoundPitch2.lua

function Init()
    SetSoundPitch("this", 0.5)
end

function Update()

end
```

Assume that the above script named `SetSoundPitch2.lua` is attached to a sound object named "sound1". In this case, string `"this"` in the `SetSoundPitch` function will be equal to "sound1". In our example, the function `SetSoundPitch` sets the pitch of current sound, which is "sound1", to `0.5`.

4.437. SetSoundPosition

Definition

`SetSoundPosition(string 3DSoundObjectName, float x, float y, float z)`

Description

This function sets the position of 3D sound **3DSoundObjectName**.

Parameters

3DSoundObjectName

Specifies the name of the 3D sound object. You can also use the name "this" for this parameter. In this case, "this" refers to the 3D sound name that this script is attached to.

x, y, z

Specify the 3D position of 3D sound **3DSoundObjectName** as three values x, y and z.

Example 1

```
function Init()
    SetSoundPosition("sound1", 2.5, 5.0, 7.0)
end

function Update()

end
```

In this script, `SetSoundPosition` function sets the position of 3D sound `"sound1"` to `(2.5, 5.0, 7.0)`.

Example 2

```
--Name of script is SetSoundPosition2.lua

function Init()
    SetSoundPosition("this", 2.5, 5.0, 7.0)
end

function Update()

end
```

Assume that the above script named `SetSoundPosition2.lua` is attached to a 3D sound object named "sound1". In this case, string `"this"` in the `SetSoundPosition` function will be equal to "sound1". In our example, the function `SetSoundPosition` sets the position of current 3D sound, which is "sound1", to `(2.5, 5.0, 7.0)`.

4.438. SetSoundReferenceDistance

Definition

`SetSoundReferenceDistance(string 3DSoundObjectName, float distance)`

Description

This function sets the reference distance of 3D sound `3DSoundObjectName` to `distance`.

Parameters

`3DSoundObjectName`

Specifies the 3D sound name. You can also use the name "this" for this parameter. In this case, "this" string refers to the name of the 3D sound to which this script is attached.

`distance`

Specifies the reference distance of 3D sound `3DSoundObjectName` to be set. This value must be greater than or equal to 0.0.

Example 1

```
function Init()
    SetSoundReferenceDistance("sound1", 4.5)
end

function Update()

end
```

This script sets the reference distance of 3D sound `"sound1"` to `4.5`.

Example 2

```
--Name of script is SetSoundReferenceDistance2.lua

function Init()
    SetSoundReferenceDistance("this", 5.0)
end

function Update()

end
```

Assume that the above script named `SetSoundReferenceDistance2.lua` is attached to a 3D sound object named "sound1". In this case, string `"this"` in the `SetSoundReferenceDistance` function will be equal to "sound1". In our example, the function `SetSoundReferenceDistance` sets the reference distance of current 3D sound, which is "sound1", to `5.0`.

4.439. SetSoundRollOff

Definition

`SetSoundRollOff(string 3DSoundObjectName, float rollOff)`

Description

This function sets the rolloff of 3D sound `3DSoundObjectName` to `rollOff`.

Parameters

`3DSoundObjectName`

Specifies the 3D sound name. You can also use the name "this" for this parameter. In this case, "this" string refers to the name of the 3D sound to which this script is attached.

`rollOff`

Specifies the rolloff of 3D sound `3DSoundObjectName` to be set. This value must be greater than or equal to 0.0.

Example 1

```
function Init()
    SetSoundRollOff("sound1", 1.5)
end

function Update()

end
```

This script sets the rolloff of 3D sound `"sound1"` to `1.5`.

Example 2

```
--Name of script is SetSoundRollOff2.lua

function Init()
    SetSoundRollOff("this", 0.5)
end

function Update()

end
```

Assume that the above script named `SetSoundRollOff2.lua` is attached to a 3D sound object named "sound1". In this case, string `"this"` in the `SetSoundRollOff` function will be equal to "sound1". In our example, the function `SetSoundRollOff` sets the rolloff of current 3D sound, which is "sound1", to `0.5`.

4.440. SetSoundVolume

Definition

`SetSoundVolume(string soundObjectName, float volume)`

Description

This function sets the volume of ambient or 3D sound `soundObjectName` to `volume`.

Parameters

`soundObjectName`

Specifies the ambient or 3D sound name. You can also use the name "this" for this parameter. In this case, "this" string refers to the name of the sound to which this script is attached.

`volume`

Specifies the volume of 3D or ambient sound `soundObjectName` to be set. This value must be in the range [0.0,1.0].

Example 1

```lua
function Init()
    SetSoundVolume("sound1", 0.1)
end

function Update()

end
```

We set the volume of sound `"sound1"` to `0.1`.

Example 2

```lua
--Name of script is SetSoundVolume2.lua

function Init()
    SetSoundVolume("this", 0.2)
end

function Update()

end
```

Assume that the above script named `SetSoundVolume2.lua` is attached to a sound object named "sound1". In this case, string `"this"` in the `SetSoundVolume` function will be equal to "sound1". In our example, the function `SetSoundVolume` sets the volume of current sound, which is "sound1", to `0.2`.

4.441. SetTerrainAmbient

Definition

`SetTerrainAmbient(float red, float green, float blue)`

Description

This function sets the ambient color of terrain object.

Parameters

red, green, blue

Specify the red, green and blue components of terrain ambient color. Each value is in the range [0.0,1.0].

Example

```
function Init()
    SetTerrainAmbient(0.25, 0.5, 0.75)
end

function Update()

end
```

In this example, the `SetTerrainAmbient` function sets the red, green, and blue components of the terrain ambient color to `(0.25, 0.5, 0.75)`, respectively.

4.442. SetTerrainDiffuse

Definition

`SetTerrainDiffuse(float red, float green, float blue)`

Description

This function sets the diffuse color of terrain object.

Parameters

`red, green, blue`

Specify the red, green and blue components of terrain diffuse color. Each value is in the range [0.0,1.0].

Example

```
function Init()
    SetTerrainDiffuse(0.25, 0.5, 0.75)
end

function Update()

end
```

In this example, the `SetTerrainDiffuse` function sets the red, green, and blue components of the terrain diffuse color to `(0.25, 0.5, 0.75)`, respectively.

4.443. SetTerrainScriptBoolVariable

Definition

`SetTerrainScriptBoolVariable(string variable, bool value)`

Description

This function sets the value of the Boolean **`variable`** defined in the script attached to the terrain object.

Parameters

`variable`

Specifies the name of the Boolean variable defined in the script attached to the terrain object.

`value`

Specifies the value of the Boolean **`variable`** to be set.

Example

```
--script name is SetTerrainScriptBoolVariable.lua attached a to game object such as water

function Init()
    SetTerrainScriptBoolVariable("a", true)
end

function Update()

end
```

Assuming that the variable `"a"` is defined in the script attached to the terrain object, `SetTerrainScriptBoolVariable` function sets the value of `"a"` to *true*.

4.444. SetTerrainScriptDoubleVariable

Definition

`SetTerrainScriptDoubleVariable(string variable, double value)`

Description

This function sets the value of the Double **`variable`** defined in the script attached to the terrain object.

Parameters

`variable`

Specifies the name of the Double variable defined in the script attached to the terrain object.

`value`

Specifies the value of the Double **`variable`** to be set.

Example

```
--script name is SetTerrainScriptDoubleVariable.lua attached a to game object such as
water

function Init()
    SetTerrainScriptDoubleVariable("a", 1.0)
end

function Update()

end
```

Assuming that the variable **`"a"`** is defined in the script attached to the terrain object, **`SetTerrainScriptDoubleVariable`** function sets the value of **`"a"`** to *1.0*.

4.445. SetTerrainScriptIntVariable

Definition

`SetTerrainScriptIntVariable(string variable, int value)`

Description

This function sets the value of the Integer **`variable`** defined in the script attached to the terrain object.

Parameters

`variable`

Specifies the name of the Integer variable defined in the script attached to the terrain object.

`value`

Specifies the value of the Integer **`variable`** to be set.

Example

```
--script name is SetTerrainScriptIntVariable.lua attached a to game object such as water

function Init()
    SetTerrainScriptIntVariable("a", 1)
end

function Update()

end
```

Assuming that the variable `"a"` is defined in the script attached to the terrain object, `SetTerrainScriptIntVariable` function sets the value of `"a"` to *1*.

4.446. SetTerrainScriptStringVariable

Definition

`SetTerrainScriptStringVariable(string variable, string value)`

Description

This function sets the value of the String **`variable`** defined in the script attached to the terrain object.

Parameters

`variable`

Specifies the name of the String variable defined in the script attached to the terrain object.

`value`

Specifies the value of the String **`variable`** to be set.

Example

```
--script name is SetTerrainScriptStringVariable.lua attached a to game object such as
water

function Init()
    SetTerrainScriptStringVariable("a", "hello")
end

function Update()

end
```

Assuming that the variable **`"a"`** is defined in the script attached to the terrain object, `SetTerrainScriptStringVariable` function sets the value of **`"a"`** to "*hello*".

4.447. SetTerrainShininess

Definition

`SetTerrainShininess(float shininess)`

Description

This function sets the shininess of terrain object.

Parameter

`shininess`

Specifies the shininess of terrain object to be set. This value must be greater than or equal to 0.0.

Example

```
function Init()
    SetTerrainShininess(50.0)
end

function Update()

end
```

In this example, the `SetTerrainShininess` function sets the shininess value of terrain object to `50.0`.

4.448. SetTerrainSpecular

Definition

`SetTerrainSpecular(float red, float green, float blue)`

Description

This function sets the specular color of terrain object.

Parameters

`red, green, blue`

Specify the red, green and blue components of terrain specular color. Each value is in the range [0.0,1.0].

Example

```
function Init()
    SetTerrainSpecular(0.25, 0.5, 0.75)
end

function Update()

end
```

In this example, the `SetTerrainSpecular` function sets the red, green, and blue components of the terrain specular color to `(0.25, 0.5, 0.75)`, respectively.

4.449. SetTriggerScriptBoolVariable

Definition

`SetTriggerScriptBoolVariable(string triggerName, string variable, bool value)`

Description

This function sets the value of the Boolean **`variable`** defined in the script attached to the trigger object **`triggerName`** .

Parameters

`triggerName`
Specifies the name of the trigger object.

`variable`
Specifies the name of the Boolean variable defined in the script attached to the trigger object **`triggerName`**.

`value`
Specifies the value of the Boolean **`variable`** to be set.

Example

```lua
--script name is SetTriggerScriptBoolVariable.lua attached a to game object such as water

function Init()
    SetTriggerScriptBoolVariable("trigger1", "a", true)
end

function Update()

end
```

Assuming that the variable `"a"` is defined in the script attached to the trigger object `"trigger1"`, `SetTriggerScriptBoolVariable` function sets the value of `"a"` to *true*.

4.450. SetTriggerScriptDoubleVariable

Definition

`SetTriggerScriptDoubleVariable(string triggerName, string variable, double value)`

Description

This function sets the value of the Double **`variable`** defined in the script attached to the trigger object **`triggerName`**.

Parameters

`triggerName`

Specifies the name of the trigger object.

`variable`

Specifies the name of the Double variable defined in the script attached to the trigger object **`triggerName`**.

`value`

Specifies the value of the Double **`variable`** to be set.

Example

```
--script name is SetTriggerScriptDoubleVariable.lua attached a to game object such as
water

function Init()
    SetTriggerScriptDoubleVariable("trigger1", "a", 1.0)
end

function Update()

end
```

Assuming that the variable `"a"` is defined in the script attached to the trigger object `"trigger1"`, `SetTriggerScriptDoubleVariable` function sets the value of `"a"` to *1.0*.

4.451. SetTriggerScriptIntVariable

Definition

`SetTriggerScriptIntVariable(string triggerName, string variable, int value)`

Description

This function sets the value of the Integer **`variable`** defined in the script attached to the trigger object **`triggerName`**.

Parameters

`triggerName`
Specifies the name of the trigger object.

`variable`
Specifies the name of the Integer variable defined in the script attached to the trigger object **`triggerName`**.

`value`
Specifies the value of the Integer **`variable`** to be set.

Example

```
--script name is SetTriggerScriptIntVariable.lua attached a to game object such as water

function Init()
    SetTriggerScriptIntVariable("trigger1", "a", 1)
end

function Update()

end
```

Assuming that the variable `"a"` is defined in the script attached to the trigger object `"trigger1"`, `SetTriggerScriptIntVariable` function sets the value of `"a"` to *1*.

4.452. SetTriggerScriptStringVariable

Definition

`SetTriggerScriptStringVariable(string triggerName, string variable, string value)`

Description

This function sets the value of the String **`variable`** defined in the script attached to the trigger object **`triggerName`**.

Parameters

`triggerName`

Specifies the name of the trigger object.

`variable`

Specifies the name of the String variable defined in the script attached to the trigger object **`triggerName`**.

`value`

Specifies the value of the String **`variable`** to be set.

Example

```
--script name is SetTriggerScriptStringVariable.lua attached a to game object such as
water

function Init()
    SetTriggerScriptStringVariable("trigger1", "a", "hello")
end

function Update()

end
```

Assuming that the variable `"a"` is defined in the script attached to the trigger object `"trigger1"`, `SetTriggerScriptStringVariable` function sets the value of `"a"` to "*hello*".

4.453. SetVideoLoop

Definition

`SetVideoLoop(string videoName, bool loop)`

Description

This function sets the loop state of the video **`videoName`** to true or false.

Parameters

`videoName`

Specifies the video name. You can also use the name "this" for this parameter. In this case, "this" string refers to the name of the video to which this script is attached.

`loop`

Specifies the state of the video loop. Accepted values are *true* or *false*.

Example 1

```
function Init()
    SetVideoLoop("video1", true)
    PlayVideo("video1")
end

function Update()

end
```

First, we set the loop status of `"video1"` to `true`. Then we play `"video1"`. Since the loop status of video `"video1"` is `true`, this video will be played continuously.

Example 2

```
--Name of script is SetVideoLoop2.lua

function Init()
    SetVideoLoop("this", false)
    PlayVideo("this")
end

function Update()

end
```

Assume that the above script named `SetVideoLoop2.lua` is attached to a video object named "video1". In this case, string `"this"` in the `SetVideoLoop` function will be equal to "video1".
In our example, we set the loop state of current video, which is "video1", to `false`. Then we play current video, which is "video1". Since the loop status of current video is `false`, this video will only be played once.

4.454. SetVideoScriptBoolVariable

Definition

`SetVideoScriptBoolVariable(string videoName, string variable, bool value)`

Description

This function sets the value of the Boolean **`variable`** defined in the script attached to the video object **`videoName`**.

Parameters

`videoName`

Specifies the name of the video object.

`variable`

Specifies the name of the Boolean variable defined in the script attached to the video object **`videoName`**.

`value`

Specifies the value of the Boolean **`variable`** to be set.

Example

```
--script name is SetVideoScriptBoolVariable.lua attached a to game object such as light

function Init()
    SetVideoScriptBoolVariable("video1", "a", true)
end

function Update()

end
```

Assuming that the variable `"a"` is defined in the script attached to the video object `"video1"`, `SetVideoScriptBoolVariable` function sets the value of `"a"` to *true*.

4.455. SetVideoScriptDoubleVariable

Definition

`SetVideoScriptDoubleVariable(string videoName, string variable, double value)`

Description

This function sets the value of the Double **`variable`** defined in the script attached to the video object **`videoName`**.

Parameters

`videoName`
Specifies the name of the video object.

`variable`
Specifies the name of the Double variable defined in the script attached to the video object `videoName`.

`value`
Specifies the value of the Double **`variable`** to be set.

Example

```
--script name is SetVideoScriptDoubleVariable.lua attached a to game object such as light

function Init()
    SetVideoScriptDoubleVariable("video1", "a", 1.0)
end

function Update()

end
```

Assuming that the variable `"a"` is defined in the script attached to the video object `"video1"`, `SetVideoScriptDoubleVariable` function sets the value of `"a"` to *1.0*.

4.456. SetVideoScriptIntVariable

Definition

`SetVideoScriptIntVariable(string videoName, string variable, int value)`

Description

This function sets the value of the Integer **`variable`** defined in the script attached to the video object **`videoName`**.

Parameters

`videoName`
Specifies the name of the video object.

`variable`
Specifies the name of the Integer variable defined in the script attached to the video object **`videoName`**.

`value`
Specifies the value of the Integer **`variable`** to be set.

Example

```lua
--script name is SetVideoScriptIntVariable.lua attached a to game object such as light

function Init()
    SetVideoScriptIntVariable("video1", "a", 1)
end

function Update()

end
```

Assuming that the variable `"a"` is defined in the script attached to the video object `"video1"`, `SetVideoScriptIntVariable` function sets the value of `"a"` to *1*.

4.457. SetVideoScriptStringVariable

Definition

`SetVideoScriptStringVariable(string videoName, string variable, string value)`

Description

This function sets the value of the String **`variable`** defined in the script attached to the video object **`videoName`**.

Parameters

`videoName`

Specifies the name of the video object.

`variable`

Specifies the name of the String variable defined in the script attached to the video object `videoName`.

`value`

Specifies the value of the String **`variable`** to be set.

Example

```
--script name is SetVideoScriptStringVariable.lua attached a to game object such as light

function Init()
    SetVideoScriptStringVariable("video1", "a", "hello")
end

function Update()

end
```

Assuming that the variable `"a"` is defined in the script attached to the video object `"video1"`, `SetVideoScriptStringVariable` function sets the value of `"a"` to "*hello*".

4.458. SetVideoVolume

Definition

`SetVideoVolume(string videoName, float volume)`

Description

This function sets the volume of video **`videoName`** to **`volume`**.

Parameters

`videoName`

Specifies the video name. You can also use the name "this" for this parameter. In this case, "this" string refers to the name of the video to which this script is attached.

`volume`

Specifies the volume of video **`videoName`** to be set. This value must be in the range [0.0,1.0].

Example 1

```
function Init()
    SetVideoVolume("video1", 0.1)
end

function Update()

end
```

In this script, we set the volume of video `"video1"` to `0.1`.

Example 2

```
--Name of script is SetVideoVolume2.lua

function Init()
    SetVideoVolume("this", 0.35)
end

function Update()

end
```

Assume that the above script named `SetVideoVolume2.lua` is attached to a video object named "video1". In this case, string `"this"` in the `SetVideoVolume` function will be equal to "video1". In our example, the function `SetVideoVolume` sets the volume of current video, which is "video1", to `0.35`.

4.459. SetVSceneScriptBoolVariable

Definition

`SetVSceneScriptBoolVariable(string variable, bool value)`

Description

This function sets the value of the Boolean **`variable`** defined in the script attached to the VScene Script object.

Parameters

`variable`

Specifies the name of the Boolean variable defined in the script attached to the VScene Script object.

`value`

Specifies the value of the Boolean **`variable`** to be set.

Example

```
--script name is SetVSceneScriptBoolVariable.lua attached a to game object such as water

function Init()
    SetVSceneScriptBoolVariable("a", true)
end

function Update()

end
```

Assuming that the variable **`"a"`** is defined in the script attached to the VScene Script object, `SetVSceneScriptBoolVariable` function sets the value of **`"a"`** to *true*.

4.460. SetVSceneScriptDoubleVariable

Definition

`SetVSceneScriptDoubleVariable(string variable, double value)`

Description

This function sets the value of the Double **`variable`** defined in the script attached to the VScene Script object.

Parameters

`variable`

Specifies the name of the Double variable defined in the script attached to the VScene Script object.

`value`

Specifies the value of the Double **`variable`** to be set.

Example

```
--script name is SetVSceneScriptDoubleVariable.lua attached a to game object such as
water

function Init()
    SetVSceneScriptDoubleVariable("a", 1.0)
end

function Update()

end
```

Assuming that the variable `"a"` is defined in the script attached to the VScene Script object, `SetVSceneScriptDoubleVariable` function sets the value of `"a"` to *1.0*.

4.461. SetVSceneScriptIntVariable

Definition

`SetVSceneScriptIntVariable(string variable, int value)`

Description

This function sets the value of the Integer **`variable`** defined in the script attached to the VScene Script object.

Parameters

`variable`

Specifies the name of the Integer variable defined in the script attached to the VScene Script object.

`value`

Specifies the value of the Integer **`variable`** to be set.

Example

```
--script name is SetVSceneScriptIntVariable.lua attached a to game object such as water

function Init()
    SetVSceneScriptIntVariable("a", 1)
end

function Update()

end
```

Assuming that the variable **`"a"`** is defined in the script attached to the VScene Script object, `SetVSceneScriptIntVariable` function sets the value of **`"a"`** to *1*.

4.462. SetVSceneScriptStringVariable

Definition

`SetVSceneScriptStringVariable(string variable, string value)`

Description

This function sets the value of the String **`variable`** defined in the script attached to the VScene Script object.

Parameters

`variable`

Specifies the name of the String variable defined in the script attached to the VScene Script object.

`value`

Specifies the value of the String **`variable`** to be set.

Example

```
--script name is SetVSceneScriptStringVariable.lua attached a to game object such as
water

function Init()
    SetVSceneScriptStringVariable("a", "hello")
end

function Update()

end
```

Assuming that the variable **`"a"`** is defined in the script attached to the VScene Script object, **`SetVSceneScriptStringVariable`** function sets the value of **`"a"`** to "*hello*".

4.463. SetWaterFlowSpeed

Definition

`SetWaterFlowSpeed(string waterName, float speed)`

Description

This function sets the flow speed of water object `waterName` to `speed`.

Parameters

waterName

Specifies the water name. You can also use the name "this" for this parameter. In this case, "this" string refers to the name of the water to which this script is attached.

speed

Specifies the flow speed of water object `waterName` to be set.

Example 1

```lua
function Init()
    SetWaterFlowSpeed("water1", 1.1)
end

function Update()

end
```

In this script, we set the flow speed of water `"water1"` to `1.1`.

Example 2

```lua
--Name of script is SetWaterFlowSpeed2.lua

function Init()
    SetWaterFlowSpeed("this", -0.05)
end

function Update()

end
```

Assume that the above script named `SetWaterFlowSpeed2.lua` is attached to a water object named "water1". In this case, string `"this"` in the `SetWaterFlowSpeed` function will be equal to "water1". In our example, the function `SetWaterFlowSpeed` sets the flow speed of current water, which is "water1", to `-0.05`.

4.464. SetWaterInvisible

Definition

`SetWaterInvisible(string waterName)`

Description

This function makes the water `waterName` invisible.

Parameters

`waterName`

Specifies the water name. You can also use the name "this" for this parameter. In this case, "this" string refers to the name of the water to which this script is attached.

Example 1

```lua
function Init()
    SetWaterInvisible("water1")
end

function Update()

end
```

In this script, we make the water `"water1"` invisible.

Example 2

```lua
--Name of script is SetWaterInvisible2.lua

function Init()
    SetWaterInvisible("this")
end

function Update()

end
```

Assume that the above script named `SetWaterInvisible2.lua` is attached to a water object named "water1". In this case, string `"this"` in the `SetWaterInvisible` function will be equal to "water1". In our example, the function `SetWaterInvisible` makes the current water, which is "water1", invisible.

4.465. SetWaterLightPosition

Definition

`SetWaterLightPosition(string waterName, float lx, float ly, float lz)`

Description

This function sets the light (sun) position of the water `waterName`.

Parameters

`waterName`

Specifies the name of the water. You can also use the name "this" for this parameter. In this case, "this" refers to the water that this script is attached to.

`lx, ly, lz`

Specify the X, Y and Z components of the light position of water `waterName`.

Example 1

```lua
function Init()
    SetWaterLightPosition("water1", -14.5, 2.7, 23.0)
end

function Update()

end
```

In this script, we set the light position of water `"water1"` to `(-14.5, 2.7, 23.0)`.

Example 2

```lua
--Name of script is SetWaterLightPosition2.lua

function Init()
    SetWaterLightPosition("this", 23.0, 3.5, 27.2)
end

function Update()

end
```

Assume that the above script named `SetWaterLightPosition2.lua` is attached to a water object named "water1". In this case, string `"this"` in the `SetWaterLightPosition` function will be equal to "water1". In our example, the function `SetWaterLightPosition` sets the light position of current water, which is "water1", to (`23.0, 3.5, 27.2`).

4.466. SetWaterPosition

Definition

`SetWaterPosition(string waterName, float x, float y, float z)`

Description

This function sets the position of the water `waterName`.

Parameters

`waterName`

Specifies the name of the water object. You can also use the name "this" for this parameter. In this case, "this" refers to the water name that this script is attached to.

`x, y, z`

Specify the X, Y and Z components of water position.

Example 1

```
function Init()
    SetWaterPosition("water1", 1.5, -2.0, 4.0)
end

function Update()

end
```

In this example, we set the position of water `"water1"` to `(1.5, -2.0, 4.0)`.

Example 2

```
--Name of script is SetWaterPosition2.lua

function Init()
    SetWaterPosition("this", 4.7, 1.0, -3.6)
end

function Update()

end
```

Assume that the above script named `SetWaterPosition2.lua` is attached to a water object named "water1". In this case, string `"this"` in the `SetWaterPosition` function will be equal to "water1". In our example, the function `SetWaterPosition` sets the position of current water, which is "water1", to `(4.7, 1.0, -3.6)`.

4.467. SetWaterRotation

Definition

`SetWaterRotation(string waterName, float rotationY)`

Description

This function sets the rotation of water **`waterName`** around Y axis in degrees.

Parameters

`waterName`

Specifies the water name. You can also use the name "this" for this parameter. In this case, "this" string refers to the name of the water to which this script is attached.

`rotationY`

Specifies the rotation of water **`waterName`** around Y axis in degrees.

Example 1

```
function Init()
    SetWaterRotation("water1", -37.5)
end

function Update()

end
```

In this script, we set the rotation of water `"water1"` around Y axis to `-37.5` degrees.

Example 2

```
--Name of script is SetWaterRotation2.lua

function Init()
    SetWaterRotation("this", 127.4)
end

function Update()

end
```

Assume that the above script named `SetWaterRotation2.lua` is attached to a water object named "water1". In this case, string `"this"` in the `SetWaterRotation` function will be equal to "water1". In our example, the function `SetWaterRotation` sets the Y rotation of current water, which is "water1", to `127.4` degrees.

4.468. SetWaterScale

Definition

`SetWaterScale(string waterName, float scaleX, float scaleZ)`

Description

This function sets the scale of water **`waterName`** in the X and Z direction.

Parameters

`waterName`

Specifies the name of the water object. You can also use the name "this" for this parameter. In this case, "this" refers to the water name that this script is attached to.

`scaleX, scaleZ`

Specify the scale of water **`waterName`** in the X and Z direction. Each of these values must be equal to or greater than 0.01.

Example 1

```
function Init()
    SetWaterScale("water1", 11.5, 23.5)
end

function Update()

end
```

In this example, `SetWaterScale` function sets the scale of water `"water1"` in the X and Z direction to `11.5` and `23.5`, respectively.

Example 2

```
--Name of script is SetWaterScale2.lua

function Init()
    SetWaterScale("this", 2.5, 14.2)
end

function Update()

end
```

Assume that the above script named `SetWaterScale2.lua` is attached to a water object named "water1". In this case, string `"this"` in the `SetWaterScale` function will be equal to "water1". In our example, the function `SetWaterScale` sets the X and Z scale of current water, which is "water1", to `2.5` and `14.2`, respectively.

4.469. SetWaterScriptBoolVariable

Definition

`SetWaterScriptBoolVariable(string waterName, string variable, bool value)`

Description

This function sets the value of the Boolean **`variable`** defined in the script attached to the water object `waterName`.

Parameters

`waterName`
Specifies the name of the water object.

`variable`
Specifies the name of the Boolean variable defined in the script attached to the water object `waterName`.

`value`
Specifies the value of the Boolean **`variable`** to be set.

Example

```
--script name is SetWaterScriptBoolVariable.lua attached a to game object such as light

function Init()
    SetWaterScriptBoolVariable("water1", "a", true)
end

function Update()

end
```

Assuming that the variable `"a"` is defined in the script attached to the water object `"water1"`, `SetWaterScriptBoolVariable` function sets the value of `"a"` to *true*.

4.470. SetWaterScriptDoubleVariable

Definition

`SetWaterScriptDoubleVariable(string waterName, string variable, double value)`

Description

This function sets the value of the Double **`variable`** defined in the script attached to the water object **`waterName`**.

Parameters

`waterName`
Specifies the name of the water object.

`variable`
Specifies the name of the Double variable defined in the script attached to the water object **`waterName`**.

`value`
Specifies the value of the Double **`variable`** to be set.

Example

```
--script name is SetWaterScriptDoubleVariable.lua attached a to game object such as light

function Init()
    SetWaterScriptDoubleVariable("water1", "a", 1.0)
end

function Update()

end
```

Assuming that the variable `"a"` is defined in the script attached to the water object `"water1"`, `SetWaterScriptDoubleVariable` function sets the value of `"a"` to *1.0*.

4.471. SetWaterScriptIntVariable

Definition

`SetWaterScriptIntVariable(string waterName, string variable, int value)`

Description

This function sets the value of the Integer **`variable`** defined in the script attached to the water object **`waterName`**.

Parameters

`waterName`
Specifies the name of the water object.

`variable`
Specifies the name of the Integer variable defined in the script attached to the water object **`waterName`**.

`value`
Specifies the value of the Integer **`variable`** to be set.

Example

```
--script name is SetWaterScriptIntVariable.lua attached a to game object such as light

function Init()
    SetWaterScriptIntVariable("water1", "a", 1)
end

function Update()

end
```

Assuming that the variable `"a"` is defined in the script attached to the water object `"water1"`, `SetWaterScriptIntVariable` function sets the value of `"a"` to *1*.

4.472. SetWaterScriptStringVariable

Definition

`SetWaterScriptStringVariable(string waterName, string variable, string value)`

Description

This function sets the value of the String **`variable`** defined in the script attached to the water object **`waterName`**.

Parameters

`waterName`
Specifies the name of the water object.

`variable`
Specifies the name of the String variable defined in the script attached to the water object **`waterName`**.

`value`
Specifies the value of the String **`variable`** to be set.

Example

```
--script name is SetWaterScriptStringVariable.lua attached a to game object such as light

function Init()
    SetWaterScriptStringVariable("water1", "a", "hello")
end

function Update()

end
```

Assuming that the variable `"a"` is defined in the script attached to the water object `"water1"`, `SetWaterScriptStringVariable` function sets the value of `"a"` to "*hello*".

4.473. SetWaterTransparency

Definition

`SetWaterTransparency(string waterName, float transparency)`

Description

This function sets the transparency of water object **`waterName`** to **`transparency`**.

Parameters

`waterName`

Specifies the water name. You can also use the name "this" for this parameter. In this case, "this" string refers to the name of the water to which this script is attached.

`transparency`

Specifies the transparency of water object **`waterName`**. This value must be in the range [0.0,1.0]

Example 1

```lua
function Init()
    SetWaterTransparency("water1", 0.4)
end

function Update()

end
```

In this script, we set the transparency of water `"water1"` to `0.4`.

Example 2

```lua
--Name of script is SetWaterTransparency2.lua

function Init()
    SetWaterTransparency("this", 0.34)
end

function Update()

end
```

Assume that the above script named `SetWaterTransparency2.lua` is attached to a water object named "water1". In this case, string `"this"` in the `SetWaterTransparency` function will be equal to "water1". In our example, the function `SetWaterTransparency` sets the transparency of current water, which is "water1", to `0.34`.

4.474. SetWaterUnderwaterColor

Definition

`SetWaterUnderwaterColor(string waterName, float red, float green, float blue)`

Description

This function sets the underwater color of water `waterName`.

Parameters

`waterName`

Specifies the name of the water object. You can also use the name "this" for this parameter. In this case, "this" refers to the water object name to which this script is attached.

`red, green, blue`

Specify the red, green and blue components of underwater color of water `waterName`. Each value is in the range [0.0,1.0].

Example 1

```
function Init()
    SetWaterUnderwaterColor("water1", 0.25, 0.5, 0.75)
end

function Update()

end
```

In this example, the `SetWaterUnderwaterColor` function sets the value of the red, green, and blue components of the underwater color of water `"water1"` to `(0.25, 0.5, 0.75)`, respectively.

Example 2

```
--Name of script is SetWaterUnderwaterColor2.lua

function Init()
    SetWaterUnderwaterColor("this", 0.25, 0.5, 0.75)
end

function Update()

end
```

Assume that the above script named `SetWaterUnderwaterColor2.lua` is attached to a water object named "water1". In this case, string `"this"` in the `SetWaterUnderwaterColor` function will be equal to "water1". In our example, the function `SetWaterUnderwaterColor` sets three values of red, green and blue underwater color of the water "water1", to `(0.25, 0.5, 0.75)`, respectively.

4.475. SetWaterUnderwaterFogDensity

Definition

`SetWaterUnderwaterFogDensity(string waterName, float density)`

Description

This function sets the underwater fog density of water object **`waterName`**.

Parameters

`waterName`

Specifies the water name. You can also use the name "this" for this parameter. In this case, "this" string refers to the name of the water to which this script is attached.

`density`

Specifies the underwater fog density of water object **`waterName`**. This value must be equal to or greater than 0.0.

Example 1

```
function Init()
    SetWaterUnderwaterFogDensity("water1", 0.15)
end

function Update()

end
```

In this script, we set the underwater fog density of water `"water1"` to `0.15`.

Example 2

```
--Name of script is SetWaterUnderwaterFogDensity2.lua

function Init()
    SetWaterUnderwaterFogDensity("this", 0.2)
end

function Update()

end
```

Assume that the above script named `SetWaterUnderwaterFogDensity2.lua` is attached to a water object named "water1". In this case, string `"this"` in the `SetWaterUnderwaterFogDensity` function will be equal to "water1". In our example, the function `SetWaterUnderwaterFogDensity` sets the underwater fog density of current water, which is "water1", to `0.2`.

4.476. SetWaterUV

Definition

`SetWaterUV(string waterName, float UV)`

Description

This function sets the texture UV of water object **`waterName`** to **`UV`**.

Parameters

`waterName`

Specifies the water name. You can also use the name "this" for this parameter. In this case, "this" string refers to the name of the water to which this script is attached.

`UV`

Specifies the texture UV of water object **`waterName`** in the U and V direction.

Example 1

```
function Init()
    SetWaterUV("water1", 0.5)
end

function Update()

end
```

In this script, we set the texture UV of water `"water1"` to `0.5`.

Example 2

```
--Name of script is SetWaterUV2.lua

function Init()
    SetWaterUV("this", 6.5)
end

function Update()

end
```

Assume that the above script named `SetWaterUV2.lua` is attached to a water object named "water1". In this case, string `"this"` in the `SetWaterUV` function will be equal to "water1". In our example, the function `SetWaterUV` sets the texture UV of current water, which is "water1", to `6.5`.

4.477. SetWaterVisible

Definition

`SetWaterVisible(string waterName)`

Description

This function makes the water **`waterName`** visible.

Parameters

`waterName`

Specifies the water name. You can also use the name "this" for this parameter. In this case, "this" string refers to the name of the water to which this script is attached.

Example 1

```
function Init()
    SetWaterVisible("water1")
end

function Update()

end
```

In this script, we make the water **`"water1"`** visible.

Example 2

```
--Name of script is SetWaterVisible2.lua

function Init()
    SetWaterVisible("this")
end

function Update()

end
```

Assume that the above script named `SetWaterVisible2.lua` is attached to a water object named "water1". In this case, string **`"this"`** in the `SetWaterVisible` function will be equal to "water1". In our example, the function `SetWaterVisible` makes the current water, which is "water1", visible.

4.478. ShowCursorIcon

Definition

`ShowCursorIcon(string resourceDirectoryName_resourceFileName.dds, float cursorSize)`

Description

This function shows the resource image `resourceDirectoryName_resourceFileName.dds`. To find the resource name in this function, first go to Script Editor (Tools > Script Editor). Then, use the Tools > Script Utility menu to open the Script Utility dialog and press the Project Resource button. You can now see all the resources in Script Utility dialog. In this dialog, you can find the desired resource image and click on the Copy Folder_File Name button to copy its full name. Then paste this name into the `ShowCursorIcon` function. In order for the `ShowCursorIcon` function to recognize this name, you must first have loaded the resource image through the `LoadResource` function (see the example).

Parameters

`resourceDirectoryName_resourceFileName.dds`
Specifies the full name of the resource image.

`cursorSize`
Specifies the size of the resource image. This value must be greater than 0.0.

Example

```
timer = 0.0
hidden = false

function Init()
    LoadResource("Images", "Cursor.dds")
    ShowCursorIcon("Images_Cursor.dds", 5.0)
end

function Update()
    if timer < 5.0 then timer = timer + GetElapsedTime() end

    if timer >= 5.0 and not hidden then
          HideCursorIcon("Images_Cursor.dds")
          hidden = true
    end
end
```

First, using the `LoadResource` function, we load the `"Cursor.dds"` image located in the `"Images"` folder. Then we display this image with size `5.0` using the `ShowCursorIcon` function. After `5.0` seconds have passed in the `Update()` event, we hide this resource image using the `HideCursorIcon` function.

4.479. ShowGUI

Definition

```
ShowGUI(string guiName)
```

Description

This function shows the GUI **guiName**.

Parameters

guiName

Specifies the GUI name.

Example

```
function OnTriggerEnter(otherActorName)
    HideGUI("gui_SampleGUI17_MainMenu")
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)
    ShowGUI("gui_SampleGUI17_MainMenu")
end
```

Assume that the above script is attached to a trigger named "trigger1". Whenever the main character or a prefab instance that has dynamic physics is entered into this trigger, the `"gui_SampleGUI17_MainMenu"` GUI will be hidden. Whenever the main character or a prefab instance that has dynamic physics exits this trigger, the `"gui_SampleGUI17_MainMenu"` GUI will be displayed.

4.480. ShowGUIButton

Definition

`ShowGUIButton(string GUIName, string buttonName)`

Description

This function shows the button **`buttonName`** that belongs to the GUI **`GUIName`**.

Parameters

`GUIName`

Specifies the GUI name.

`buttonName`

Specifies the button name that belongs to the GUI **`GUIName`**.

Example

```
function OnTriggerEnter(otherActorName)
    HideGUIButton("gui_SampleGUI17_MainMenu", "PlayGame")
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)
    ShowGUIButton("gui_SampleGUI17_MainMenu", "PlayGame")
end
```

Assume that the above script is attached to a trigger named "trigger1". Whenever the main character or a prefab instance that has dynamic physics is entered into this trigger, the button `"PlayGame"` that belongs to GUI `"gui_SampleGUI17_MainMenu"` will be hidden. Whenever the main character or a prefab instance that has dynamic physics exits this trigger, the button `"PlayGame"` that belongs to GUI `"gui_SampleGUI17_MainMenu"` will be displayed.

4.481. ShowGUIImage

Definition

`ShowGUIImage(string GUIName, string imageName)`

Description

This function shows the image **`imageName`** that belongs to the GUI **`GUIName`**.

Parameters

`GUIName`

Specifies the GUI name.

`imageName`

Specifies the image name that belongs to the GUI **`GUIName`**.

Example

```
function OnTriggerEnter(otherActorName)
    HideGUIImage("gui_SampleGUI17_MainMenuAbout", "backgroundImg")
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)
    ShowGUIImage("gui_SampleGUI17_MainMenuAbout", "backgroundImg")
end
```

Assume that the above script is attached to a trigger named "trigger1". Whenever the main character or a prefab instance that has dynamic physics is entered into this trigger, the image `"backgroundImg"` that belongs to GUI `"gui_SampleGUI17_MainMenuAbout"` will be hidden. Whenever the main character or a prefab instance that has dynamic physics exits this trigger, the image `"backgroundImg"` that belongs to GUI `"gui_SampleGUI17_MainMenuAbout"` will be displayed.

4.482. ShowGUIText

Definition

`ShowGUIText(string GUIName, string textName)`

Description

This function shows the text `textName` that belongs to the GUI `GUIName`.

Parameters

`GUIName`

Specifies the GUI name.

`textName`

Specifies the text name that belongs to the GUI `GUIName`.

Example

```
function OnTriggerEnter(otherActorName)
    HideGUIText("gui_SampleGUI17_MainMenuAbout", "text1")
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)
    ShowGUIText("gui_SampleGUI17_MainMenuAbout", "text1")
end
```

Assume that the above script is attached to a trigger named "trigger1". Whenever the main character or a prefab instance that has dynamic physics is entered into this trigger, the text `"text1"` that belongs to GUI `"gui_SampleGUI17_MainMenuAbout"` will be hidden. Whenever the main character or a prefab instance that has dynamic physics exits this trigger, the text `"text1"` that belongs to GUI `"gui_SampleGUI17_MainMenuAbout"` will be displayed.

4.483. ShowMenuCursor

Definition

```
ShowMenuCursor([optional] int cursorSize)
```

Description

This function shows the menu cursor image. You can change the menu cursor image and its properties through the Current VScene Properties dialog (Tools > Current VScene Properties).

Parameter

cursorSize

Specifies the menu cursor size. this parameter is optional. If this value is not specified, the menu cursor size specified in the Current VScene Properties dialog will be used. This value must be greater than 0.

Example 1

```
function OnTriggerEnter(otherActorName)
    ShowMenuCursor(4)
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)
    HideMenuCursor()
end
```

Assume that the above script is attached to a trigger named "trigger1". Whenever the main character or a prefab instance that has dynamic physics is entered into this trigger, the menu cursor image with size 4 will be displayed. Whenever the main character or a prefab instance that has dynamic physics exits this trigger, the menu cursor image will be hidden.

Example 2

```
function OnTriggerEnter(otherActorName)
    ShowMenuCursor()
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)
    HideMenuCursor()
end
```

Assume that the above script is attached to a trigger named "trigger1".
Whenever the main character or a prefab instance that has dynamic physics is entered into this trigger, the menu cursor image will be displayed. Since we have not specified the menu cursor size

in the `ShowMenuCursor` function, the menu cursor size specified in the Current VScene Properties dialog will be used.
Whenever the main character or a prefab instance that has dynamic physics exits this trigger, the menu cursor image will be hidden.

4.484. ShowPrefabInstance

Definition

`ShowPrefabInstance(string prefabInstanceName)`

Description

This function shows the prefab instance `prefabInstanceName`. To view the name of prefab instances, open the VScene and click on the desired Prefab Instance in the "Prefabs and GUIs" section and press the Edit button. You can also access the names of prefab instances from the Script Utility section of the script editor (Tools > Script Editor > Tools > Script Utility). In the dialog that appears, you can view and copy the name of the prefab instance.

Parameters

prefabInstanceName

Specifies the name of the prefab instance. You can also use the name "this" for this parameter. In this case, "this" refers to the prefab instance that this script is attached to.

Example 1

```
timer = 0.0
shown = false

function Init()
    HidePrefabInstance("1_VandaEngine17-SamplePack1_eggbox")
end

function Update()
    timer = timer + GetElapsedTime()
    if timer >= 5.0 and not shown then
        ShowPrefabInstance("1_VandaEngine17-SamplePack1_eggbox")
        shown = true
    end
end
```

Assume that this script is attached to a game object such as main character. First, we hide the prefab instance `"1_VandaEngine17-SamplePack1_eggbox"`. After `5.0` seconds, `ShowPrefabInstance` function will show `"1_VandaEngine17-SamplePack1_eggbox"` prefab instance.

Example 2

```
--name of the script is ShowPrefabInstance2.lua

function Init()
    ShowPrefabInstance("this")
end

function Update()

end
```

```
If, in the Prefab Editor, you attach ShowPrefabInstance2.lua script to
a Prefab, then "this" parameter in the ShowPrefabInstance function will
point to instances of that Prefab in current VScene. For example, if you
have an Instance named instance1_a from a Prefab named a to which this script
is attached, "this" in ShowPrefabInstance function refers to the name
instance1_a.
```

In this example, assume that this script is attached to a prefab named *prefab_a* and we have an instance of it named `instance1_prefab_a` and `instance1_prefab_a` is hidden at the beginning of the game. In this case, this script shows current prefab instance, which is `instance1_prefab_a`.

4.485. StopAll3DSounds

Definition

`StopAll3DSounds([optional] string exception_1, [optional] string exception_2,..., [optional] string exception_n)`

Description

This function stops all 3D sounds that are being played except for the 3D sounds sent to the function.

Parameters

`[optional] string exception_1, [optional] string exception_2,..., [optional] string exception_n`

Specifies the name of the 3D sounds that should not stop by this function. If no name is passed to `StopAll3DSounds` function, all 3D sounds that are being played will stop.

Example

```
function OnTriggerEnter(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          StopAll3DSounds("sound3D_2", "sound3D_3")
    end
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)

end
```

Assume that the above script is attached to a trigger named "trigger1". Whenever the main game character enters "trigger1", all the 3D sounds that are playing except the 3D sounds `"sound3D_2"` and `"sound3D_3"` will stop.

4.486. StopAllAmbientSounds

Definition

`StopAllAmbientSounds([optional] string exception_1, [optional] string exception_2,..., [optional] string exception_n)`

Description

This function stops all ambient sounds that are being played except for the ambient sounds sent to the function.

Parameters

`[optional] string exception_1, [optional] string exception_2,..., [optional] string exception_n`

Specifies the name of the ambient sounds that should not stop by this function. If no name is passed to `StopAllAmbientSounds` function, all ambient sounds that are being played will stop.

Example

```
function OnTriggerEnter(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          StopAllAmbientSounds("ambient2", "ambient3")
    end
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)

end
```

Assume that the above script is attached to a trigger named "trigger1". Whenever the main game character enters "trigger1", all the ambient sounds that are playing except the ambient sounds `"ambient2"` and `"ambient3"` will stop.

4.487. StopAllResourceSounds

Definition

```
StopAllResourceSounds([optional] string exception_1, [optional] string
exception_2,..., [optional] string exception_n)
```

Description

This function stops all resource sounds that are being played except for the resource sounds sent to the function.

Parameters

[optional] string exception_1, [optional] string exception_2,..., [optional] string exception_n

Specifies the name of the resource sounds that should not stop by this function. If no name is passed to `StopAllResourceSounds` function, all resource sounds that are being played will stop.

Example

```
function OnTriggerEnter(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          LoadResource("Sounds", "fire.ogg")
          LoadResource("Sounds", "river.ogg")
          LoadResource("Sounds", "ambient.ogg")

          PlayResourceSoundLoop("Sounds_fire.ogg")
          PlayResourceSoundLoop("Sounds_river.ogg")
          PlayResourceSoundLoop("Sounds_ambient.ogg")
    end
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          StopAllResourceSounds("Sounds_ambient.ogg")
    end
end
```

Assume that the above script is attached to a trigger named "trigger1". Whenever the main character enters "trigger1", we load and play `"fire.ogg"`, `"river.ogg"` and `"ambient.ogg"` resource sounds --In order for `LoadResource` function to load the resources, you must first add all resources through the *Add Resource to Current Project* dialog (File > Project > Add/Remove Resource to/from Current Project).
Whenever the main character exits "trigger1", all resource sounds that are playing except the resource sound `"ambient.ogg"` will stop.

4.488. StopAllSounds

Definition

`StopAllSounds([optional] string exception_1, [optional] string exception_2,..., [optional] string exception_n)`

Description

This function stops all ambient, 3D and resource sounds that are being played except for the ambient, 3D and resource sounds sent to the function.

Parameters

`[optional] string exception_1, [optional] string exception_2,..., [optional] string exception_n`

Specifies the name of the ambient, 3D and resource sounds that should not stop by this function. If no name is passed to `StopAllSounds` function, all ambient, 3D and resource sounds that are being played will stop.

Example

```
function OnTriggerEnter(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          LoadResource("Sounds", "fire.ogg")
          LoadResource("Sounds", "river.ogg")
          LoadResource("Sounds", "ambient.ogg")

          PlayResourceSoundLoop("Sounds_fire.ogg")
          PlayResourceSoundLoop("Sounds_river.ogg")
          PlayResourceSoundLoop("Sounds_ambient.ogg")
    end
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          StopAllSounds("ambient2", "river2", "Sounds_ambient.ogg")
    end
end
```

Assume that the above script is attached to a trigger named "trigger1". Also, `"ambient2"` and `"river2"` in the example above are ambient and 3D sounds, respectively.
Whenever the main character enters "trigger1", we load and play `"fire.ogg"`, `"river.ogg"` and `"ambient.ogg"` resource sounds --In order for `LoadResource` function to load the resources, you must first add all resources through the *Add Resource to Current Project* dialog (File > Project > Add/Remove Resource to/from Current Project).

Whenever the main character exits "trigger1", all ambient, 3D and resource sounds that are playing except the the ambient sound `"ambient2"`, 3D sound `"river2"` and resource sound `"ambient.ogg"` will stop.

4.489. StopResourceSound

Definition

`StopResourceSound(string resourceDirectoryName_resourceFileName.ogg)`

Description

This function stops resource sound `resourceDirectoryName_resourceFileName.ogg` that is being played. You can go to the *Project Resources* section through the Script Utility dialog (Tools > Script Editor > Tools > Script Utility), select the desired resource sound and hit "Copy Folder_File Name" button to copy the full name of the resource.

Parameters

`resourceDirectoryName_resourceFileName.ogg`
Specifies the full name of the resource sound.

Example

```
function OnTriggerEnter(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          LoadResource("Sounds", "fire.ogg")
          PlayResourceSoundLoop("Sounds_fire.ogg")
    end
end

function OnTriggerStay(otherActorName)

end

function OnTriggerExit(otherActorName)
    --nil means main character controller
    if otherActorName == nil then
          StopResourceSound("Sounds_fire.ogg")
    end
end
```

Assume that the above script is attached to a trigger named "trigger1". Whenever the main character enters "trigger1", we load and play `"fire.ogg"` resource sound --In order for `LoadResource` function to load the resource sound, you must first add `"fire.ogg"` sound resource through the *Add Resource to Current Project* dialog (File > Project > Add/Remove Resource to/from Current Project).
Whenever the main character exits "trigger1", the resource sound `"fire.ogg"` will stop.

4.490. StopSound

Definition

`StopSound(string soundObjectName1, string soundObjectName2, ..., string soundObjectNameN)`

Description

This function stops all ambient and 3D sounds `soundObjectName1, soundObjectName2, ..., soundObjectNameN` that are playing.

Parameters

`soundObjectName1, soundObjectName2, ..., soundObjectNameN`

Specify the name of the ambient and 3D sounds that should stop by this function. You can also use the name "this" for ***`soundObjectName[N]`***. In this case, "this" refers to the ambient or 3D sound that this script is attached to.

Example

```
function Init()
    StopSound("this", "ambient2", "fire1")
end

function Update()

end
```

Assume that the above script is attached to an ambient sound named "ambient1". Also, `"ambient2"` and `"fire1"` in the example above are ambient and 3D sound names, respectively. In our example, `StopSound` function stops the current sound (which has a name equivalent to "ambient1"), the ambient sound `"ambient2"`, and the 3D sound `"fire1"`.

4.491. StopVideo

Definition

```
StopVideo(string videoName)
```

Description

This function stops video `videoName`.

Parameters

`videoName`

Specifies the name of the video object. You can also use the name "this" for this parameter. In this case, "this" refers to the video object that this script is attached to.

Example 1

```
--Name of script is StopVideo1.lua
--You can attach this script to a video object

timer = 0.0
stop = false

function Init()
    PlayVideo("this")
end

function Update()
    timer = timer + GetElapsedTime()
    if timer >= (GetVideoDuration("this") / 5.0) and not stop then
        StopVideo("this")
        stop = true
    end
end
```

In this case, `"this"` string in the `StopVideo` points to the video that `StopVideo1.lua` script is attached to. For example, if `StopVideo1.lua` script is attached to a video object named "video1", `"this"` will be equivalent to the name "video1".
First we play the current video object. Then in the `Update()` event, we stop the current video after 20% of the current video's total duration.

Example 2

```
--You can attach this script to a video object named "video1"
timer = 0.0
stop = false

function Init()
    PlayVideo("video1")
end

function Update()
    timer = timer + GetElapsedTime()
    if timer >= (GetVideoDuration("video1") / 5.0) and not stop then
```

```
          StopVideo("video1")
          stop = true
    end
end
```

Assume that the above script is attached to a video object named "video1". First we play the video `"video1"`. Then in the `Update()` event, we stop the video `"video1"` after 20% of the total duration of the video `"video1"`.

4.492. TranslatePrefabInstance

Definition

`TranslatePrefabInstance(string prefabInstanceName, float XPosition, float YPosition, float ZPosition)`

Description

This function moves the *transformable* prefab instance **`prefabInstanceName`** to the (X, Y, Z) position. For this function to work, in prefab mode, through the Modify > Prefab Properties menu, make sure the *Transformable* option is checked for the desired prefab.

Parameters

`prefabInstanceName`
Specifies the name of the prefab instance. You can also use the name "this" for this parameter. In this case, "this" refers to the prefab instance that this script is attached to.

`XPosition, YPosition, ZPosition`
Specify the X, Y and Z components of 3D position of the prefab instance ***`prefabInstanceName`***.

Example 1

```
translateX = 0.0
translateY = 0.0
translateZ = 0.0

function Init()

end

function Update()
    translateX = translateX + (GetElapsedTime() * 0.1)
    translateY = translateY + (GetElapsedTime() * 0.2)
    translateZ = translateZ + (GetElapsedTime() * 0.3)

    if translateX > 5.0 then translateX = 0.0 end
    if translateY > 5.0 then translateY = 0.0 end
    if translateZ > 5.0 then translateZ = 0.0 end

    TranslatePrefabInstance("1_VandaEngine17-SamplePack1_well", translateX, translateY,
translateZ)
end
```

First, we increase the value of **`translateX`**, **`translateY`** and **`translateZ`** variables according to time and make sure that their value is not more than **`5.0`** units. Then, using these three values and the `TranslatePrefabInstance` function, we translate the prefab instance `1_VandaEngine17-SamplePack1_well` in the X, Y and Z directions. It should be noted that the Transformable property of prefab instance `1_VandaEngine17-SamplePack1_well` must be enabled for the function `TranslatePrefabInstance` to work.

Example 2

```
--Name of script is TranslatePrefabInstance2.lua
```

```
translateX = 0.0
translateY = 0.0
translateZ = 0.0

function Init()

end

function Update()
    translateX = translateX + (GetElapsedTime() * 0.1)
    translateY = translateY + (GetElapsedTime() * 0.2)
    translateZ = translateZ + (GetElapsedTime() * 0.3)

    if translateX > 5.0 then translateX = 0.0 end
    if translateY > 5.0 then translateY = 0.0 end
    if translateZ > 5.0 then translateZ = 0.0 end

    TranslatePrefabInstance("this", translateX, translateY, translateZ)
end
```

If, in the Prefab Editor, you attach `TranslatePrefabInstance2.lua` script to a Prefab, then `"this"` parameter in the `TranslatePrefabInstance` function will point to instances of that Prefab in current VScene. For example, if you have an Instance named *instance1_a* from a Prefab named *a* to which this script is attached, `"this"` in `TranslatePrefabInstance` function refers to the name *instance1_a*.

First, we increase the value of **`translateX`**, **`translateY`** and **`translateZ`** variables according to time and make sure that their value is not more than `5.0` units. Then, using these three values and the `TranslatePrefabInstance` function, we translate the current prefab instance (for example, *instance1_a*) in the X, Y and Z directions. It should be noted that the Transformable property of current prefab instance must be enabled for the function `TranslatePrefabInstance` to work.

4.493. UnlockCharacterController

Definition

```
UnlockCharacterController()
```

Description

This function unlocks physics character controller. In this case, you can move the main game character or the camera attached to it using the keyboard or mouse.

Example

```
function OnSelectMouseLButtonDown()
    LockCharacterController()
end

function OnSelectMouseRButtonDown()
    UnlockCharacterController()
end

function OnSelectMouseEnter()

end
```

Assume that the above script is attached to a button object. Whenever you left click on this button, the main character will be locked. Whenever you right click on this button, the main character will be unlocked.

4.494. WriteBoolVariableToFile

Definition

`WriteBoolVariableToFile(bool value)`

Description

This function writes a boolean value to the currently open file. Before writing information to the file, make sure that you have opened the desired file for writing with the `OpenFileForWriting` function.

Parameter

`value`

Specifies a boolean value to write to the currently opened file.

Example

```
bVar = false

function Init()
   --Create a folder in Assets/Data/ path
   CreateFolder("Lev1")

   --Create and open file to write data
   OpenFileForWriting("Lev1/level1.bin")
   WriteBoolVariableToFile(true)
   CloseFile("Lev1/level1.bin")

   --Open File to load data
   OpenFileForReading("Lev1/level1.bin")
   bVar = ReadBoolVariableFromFile()
   CloseFile("Lev1/level1.bin")
end
```

First, using the `CreateFolder` function, we create a folder called `"Lev1"` in the Assets/Data/ path. Then, using the `OpenFileForWriting` function, we open the `level1.bin` file located in the Assets/Data/`Lev1`/ path for writing. After writing the Boolean value `true` by the `WriteBoolVariableToFile` function, we close the file by the `CloseFile` function.
Then, using the `OpenFileForReading` function, we open the `level1.bin` file located in the path Assets/Data/`Lev1`/ for reading and read a boolean variable from the `level1.bin` file with the `ReadBoolVariableFromFile()` function. In our example, value of `bVar` is `true` after reading it. Finally, we close the file by the `CloseFile` function.

4.495. WriteFloatVariableToFile

Definition

`WriteFloatVariableToFile(float value)`

Description

This function writes a floating point value to the currently open file. Before writing information to the file, make sure that you have opened the desired file for writing with the `OpenFileForWriting` function.

Parameter

`value`

Specifies a floating point value to write to the currently opened file.

Example

```
fVar = 0.0

function Init()
   --Create a folder in Assets/Data/ path
   CreateFolder("Lev1")

   --Create and open file to write data
   OpenFileForWriting("Lev1/level1.bin")
   WriteFloatVariableToFile(2.0)
   CloseFile("Lev1/level1.bin")

   --Open File to load data
   OpenFileForReading("Lev1/level1.bin")
   fVar = ReadFloatVariableFromFile()
   CloseFile("Lev1/level1.bin")
end
```

First, using the `CreateFolder` function, we create a folder called `"Lev1"` in the Assets/Data/ path. Then, using the `OpenFileForWriting` function, we open the `level1.bin` file located in the Assets/Data/`Lev1`/ path for writing. After writing the floating point value `2.0` by the `WriteFloatVariableToFile` function, we close the file by the `CloseFile` function. Then, using the `OpenFileForReading` function, we open the `level1.bin` file located in the path Assets/Data/`Lev1`/ for reading and read a floating point variable from the `level1.bin` file with the `ReadFloatVariableFromFile()` function. In our example, value of `fVar` is `2.0` after reading it. Finally, we close the file by the `CloseFile` function.

4.496. WriteIntVariableToFile

Definition

`WriteIntVariableToFile(int value)`

Description

This function writes an integer value to the currently open file. Before writing information to the file, make sure that you have opened the desired file for writing with the `OpenFileForWriting` function.

Parameter

`value`

Specifies an integer value to write to the currently opened file.

Example

```
iVar = 0

function Init()
   --Create a folder in Assets/Data/ path
   CreateFolder("Lev1")

   --Create and open file to write data
   OpenFileForWriting("Lev1/level1.bin")
   WriteIntVariableToFile(3)
   CloseFile("Lev1/level1.bin")

   --Open File to load data
   OpenFileForReading("Lev1/level1.bin")
   iVar = ReadIntVariableFromFile()
   CloseFile("Lev1/level1.bin")
end
```

First, using the `CreateFolder` function, we create a folder called `"Lev1"` in the Assets/Data/ path. Then, using the `OpenFileForWriting` function, we open the `level1.bin` file located in the Assets/Data/Lev1/ path for writing. After writing an integer value 3 by the `WriteIntVariableToFile` function, we close the file by the `CloseFile` function. Then, using the `OpenFileForReading` function, we open the `level1.bin` file located in the path Assets/Data/Lev1/ for reading and read an integer variable from the `level1.bin` file with the `ReadIntVariableFromFile()` function. In our example, value of `iVar` is 3 after reading it. Finally, we close the file by the `CloseFile` function.

4.497. WriteStringVariableToFile

Definition

`WriteStringVariableToFile(string value)`

Description

This function writes a string value to the currently open file. Before writing information to the file, make sure that you have opened the desired file for writing with the `OpenFileForWriting` function.

Parameter

`value`

Specifies a string value to write to the currently opened file.

Example

```
sVar = "init"

function Init()
   --Create a folder in Assets/Data/ path
   CreateFolder("Lev1")

   --Create and open file to write data
   OpenFileForWriting("Lev1/level1.bin")
   WriteStringVariableToFile("level1")
   CloseFile("Lev1/level1.bin")

   --Open File to load data
   OpenFileForReading("Lev1/level1.bin")
   sVar = ReadStringVariableFromFile()
   CloseFile("Lev1/level1.bin")
end
```

First, using the `CreateFolder` function, we create a folder called `"Lev1"` in the Assets/Data/ path. Then, using the `OpenFileForWriting` function, we open the `level1.bin` file located in the Assets/Data/`Lev1`/ path for writing. After writing a string value `"level1"` by the `WriteStringVariableToFile` function, we close the file by the `CloseFile` function. Then, using the `OpenFileForReading` function, we open the `level1.bin` file located in the path Assets/Data/`Lev1`/ for reading and read a string variable from the `level1.bin` file with the `ReadStringVariableFromFile()` function. In our example, value of `sVar` is `"level1"` after reading it. Finally, we close the file by the `CloseFile` function.