Diamond 1.0

Visual Scripting for Unity
Documentation

www.mezanix.com

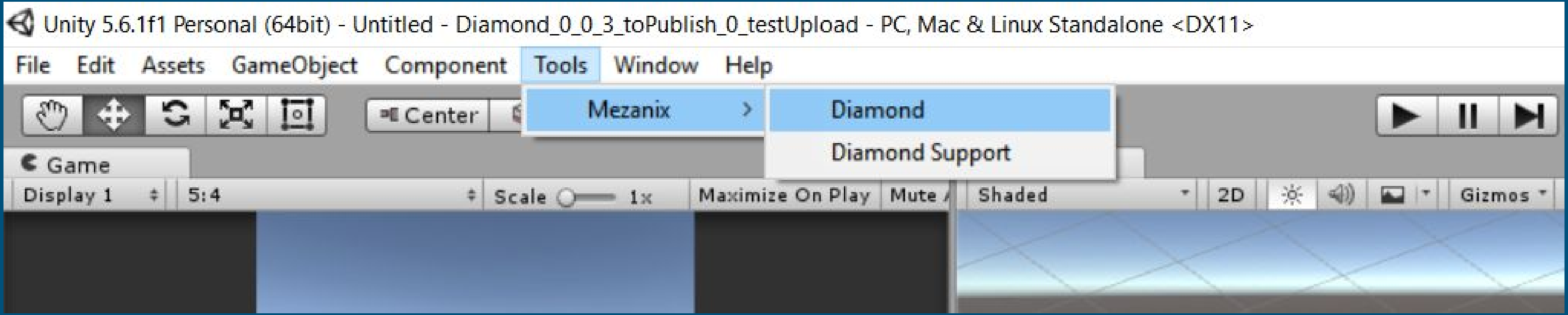
Before Diving in Diamond

Diamond use the State Machine concept. You can have an idea about it here

https://en.wikipedia.org/wiki/Finite-state_machine

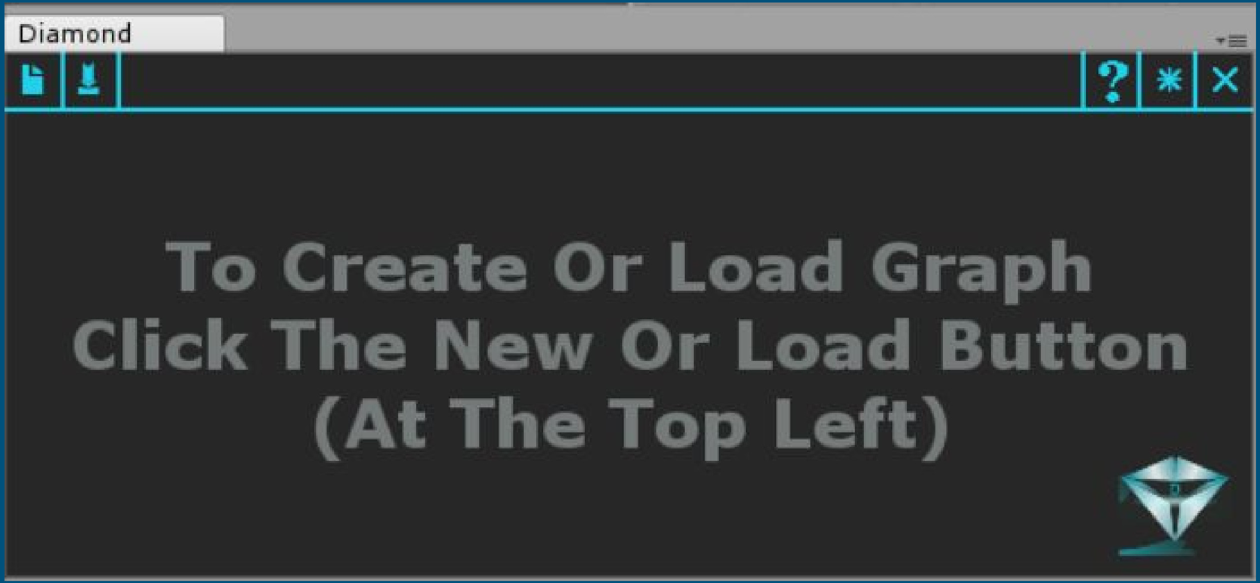
Open Diamond

Go to the Unity top menu: Tools / Mezanix / Diamond.



Diamond Window

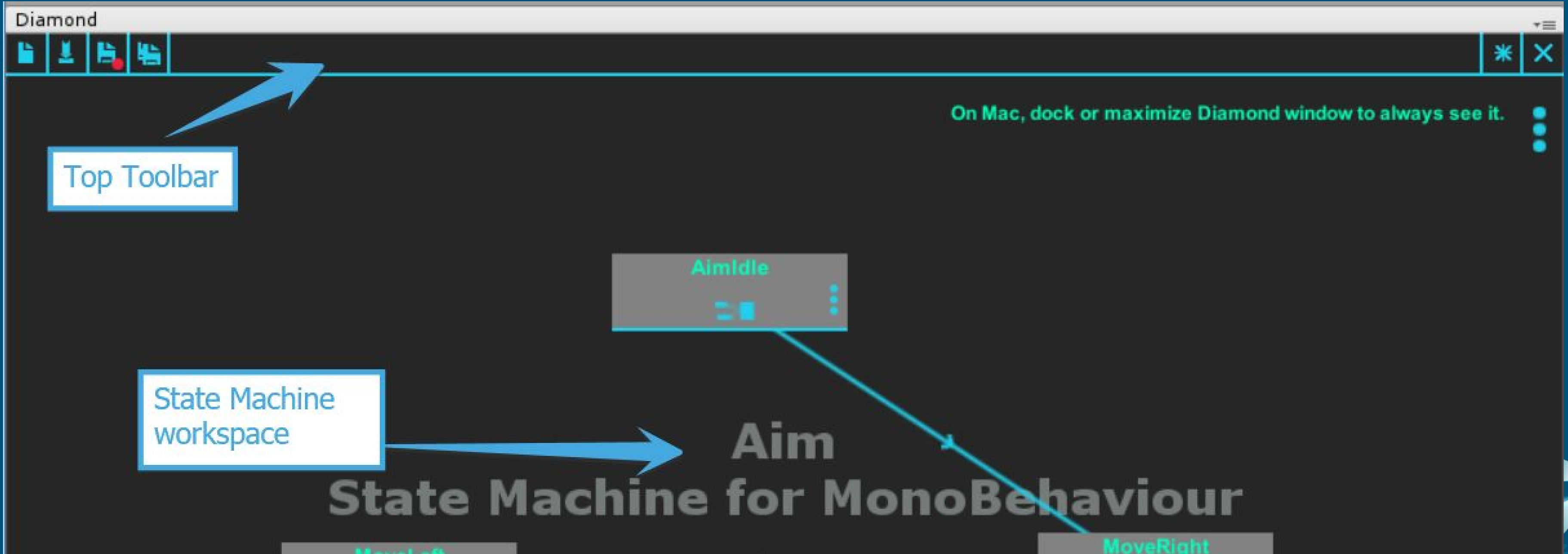
You will see this window:



To Create Or Load Graph
Click The New Or Load Button
(At The Top Left)

Diamond Window with Graph

If you already have a graph loaded, you will see this window:

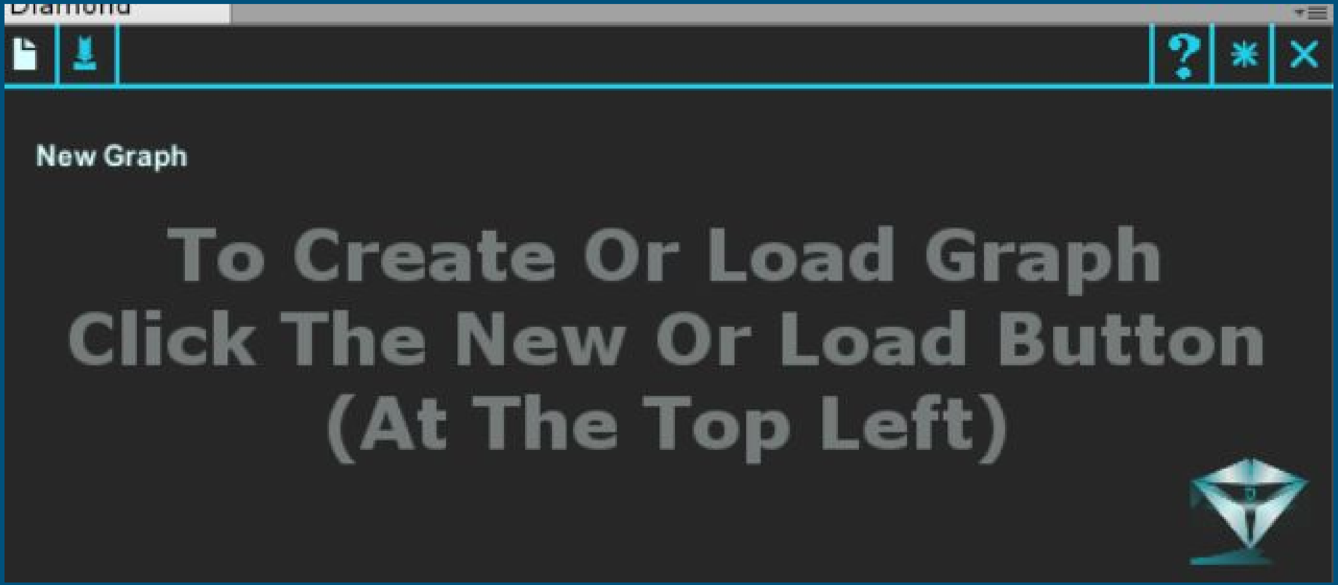


On Mac, dock or maximize Diamond window to always see it.

Aim
State Machine for MonoBehaviour

Create a new Graph

Click the new button

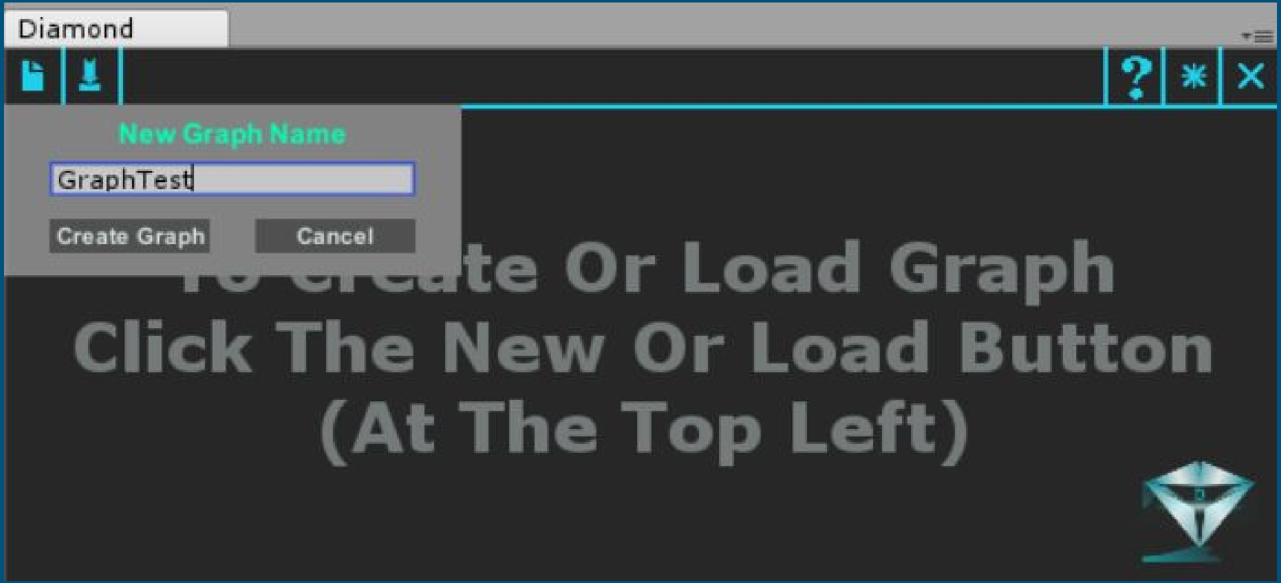


New Graph

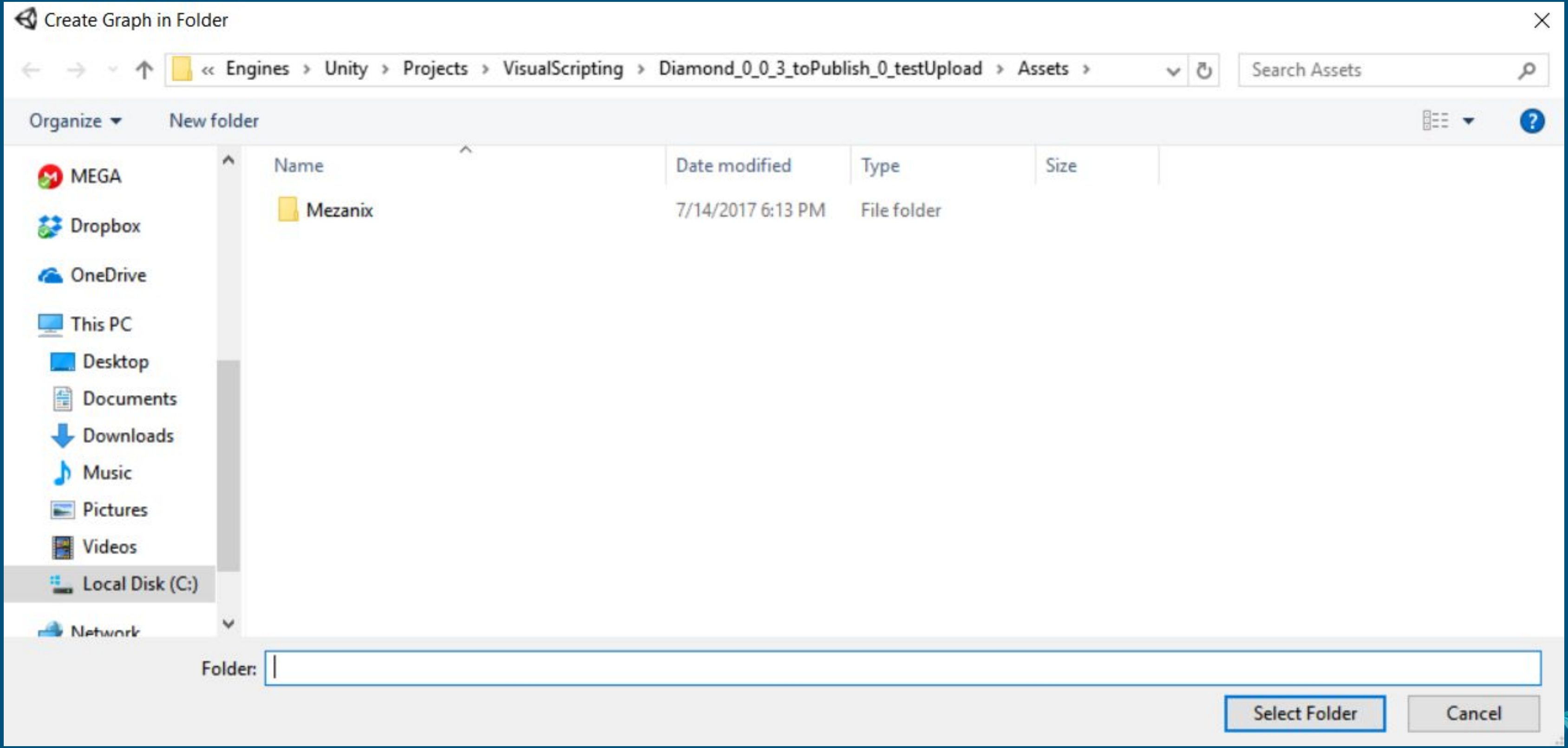
To Create Or Load Graph
Click The New Or Load Button
(At The Top Left)

Create a new Graph

The New Graph menu will appear, call it GraphTest and click Create Graph

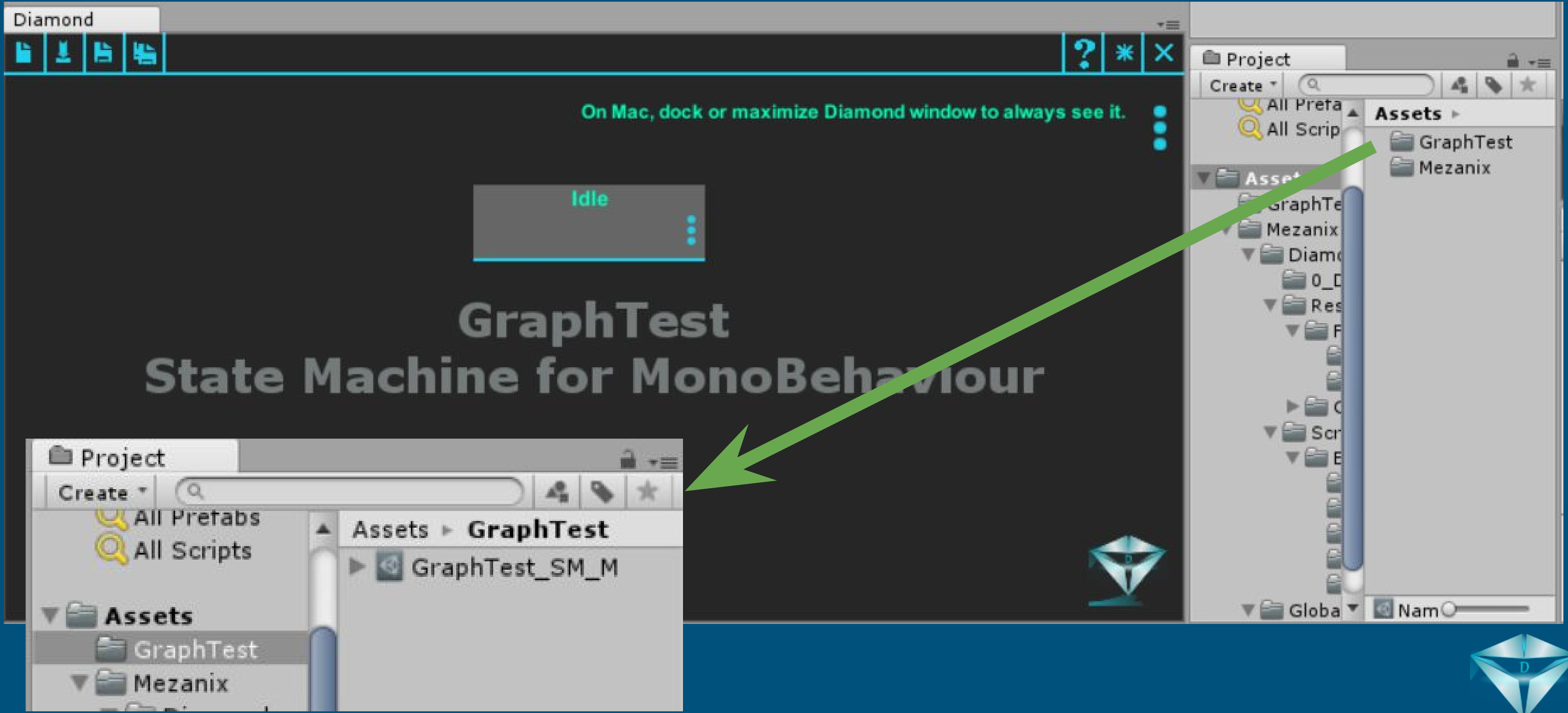


After choosing a name, Diamond ask you where to save your graph and prompt you somewhere in your Unity project folder (same path for your last loaded graph, otherwise at the project root).



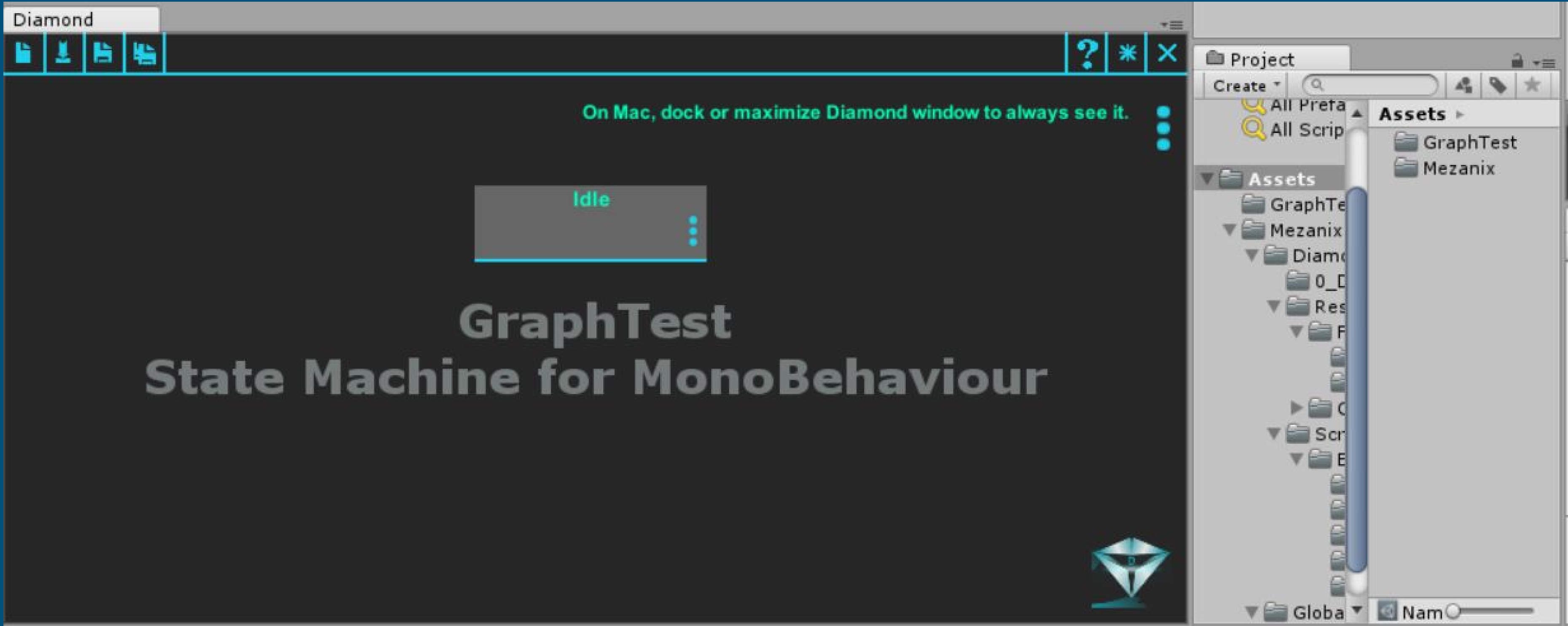
Name	Date modified	Type	Size
Mezanix	7/14/2017 6:13 PM	File folder	

Diamond creates a folder for your graph with the same name of the graph. Inside it you have your graph object with the "_SM_M" suffix standing for "State Machine For MonoBehaviour"



So it's a State Machine graph. A state machine holds many states and has only one active state at each time. At least one state has to be in the state machine. Every new graph has his default state created with it, the Idle state.

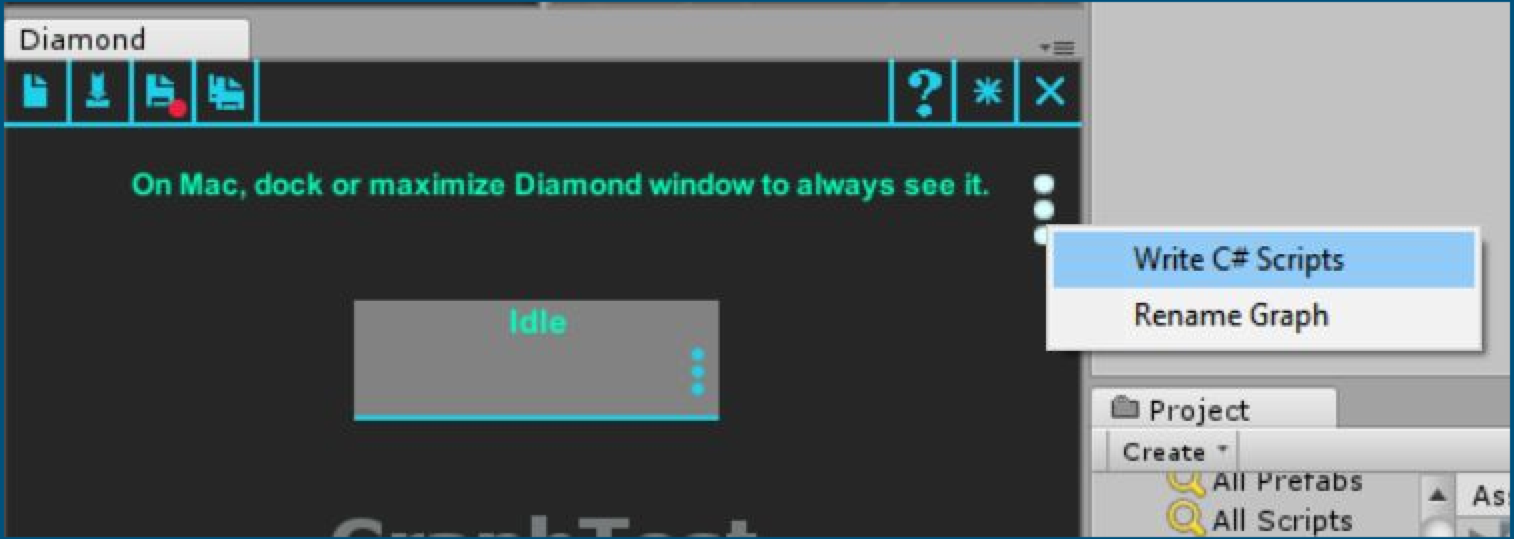
You can't delete or rename the Idle state.


Diamond
On Mac, dock or maximize Diamond window to always see it.
Idle
GraphTest
State Machine for MonoBehaviour
Project
Create
Assets
GraphTest
Mezanix

Generate your C# scripts

Your graph is now created, you can generate your scripts by clicking the options button of your graph (top right) .

Return to slide

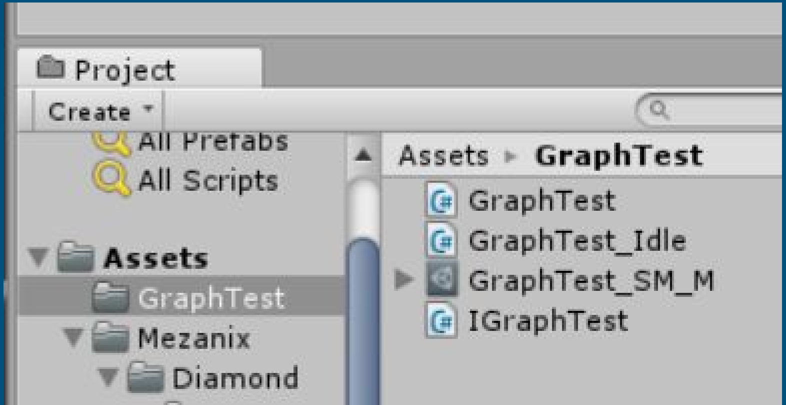


Even if your graph has no logic yet, scripts are generated. In the generated scripts, there's only one that Unity accepts to attach to a gameobject, it’s the only one deriving from MonoBehaviour and has the same name of our graph, in this case the GraphTest.cs one..

Other scripts are helpers, representing your graph elements. In our empty graph we have one state, the Idle state. So we have a script called GraphTest_Idle.cs.

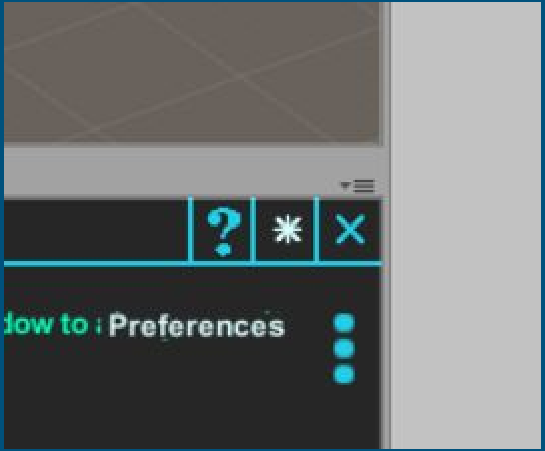
Notice that we have a script called IGraphTest.cs (the graph name + “I” as prefix). It’s an Interface used to manage the switching between the states.

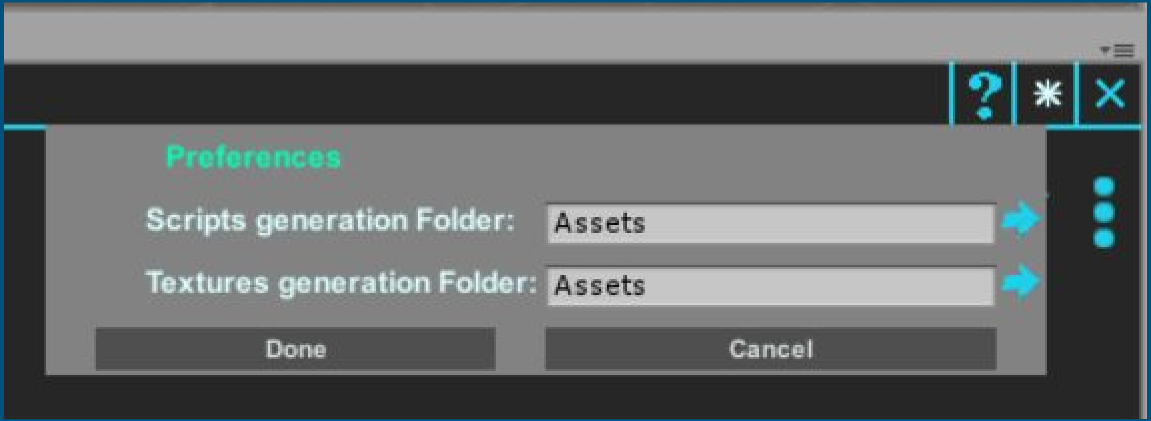
Return to slide



Generate your C# scripts, where ?

Click the preferences asterix (top right), click the blue arrow at the right and a explorer / finder window will open asking to select the scripts generation path



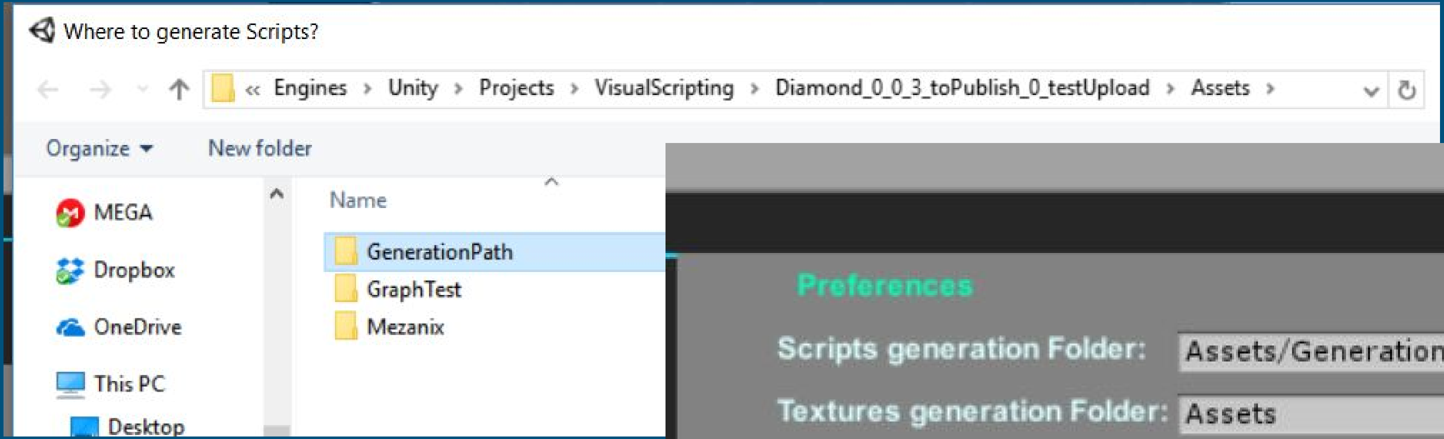


Scripts generation Folder: Assets

Textures generation Folder: Assets

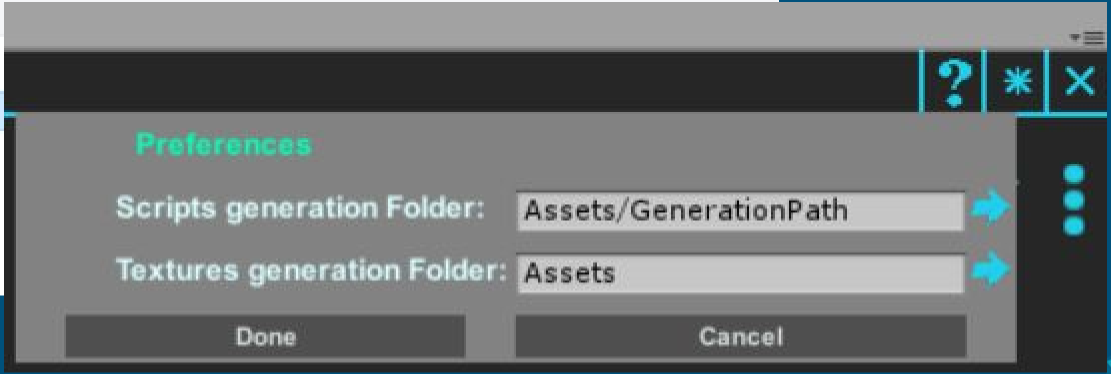
Generate your C# scripts, where ?

Inside the project I created a folder called “GenerationPath” and selected it.
Last and not least, click Done to close menu and save the preferences.


Where to generate Scripts?

« Engines › Unity › Projects › VisualScripting › Diamond_0_0_3_toPublish_0_testUpload › Assets ›

GenerationPath
GraphTest
Mezanix

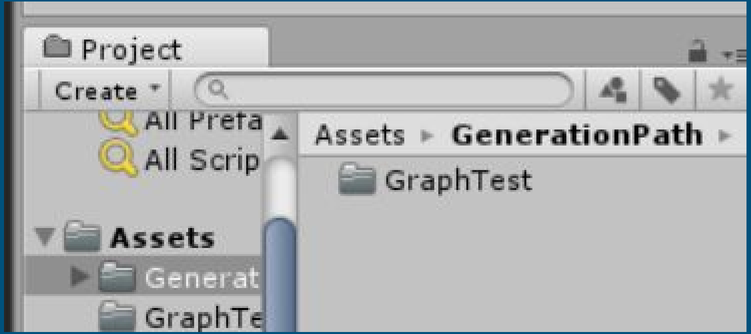

Preferences

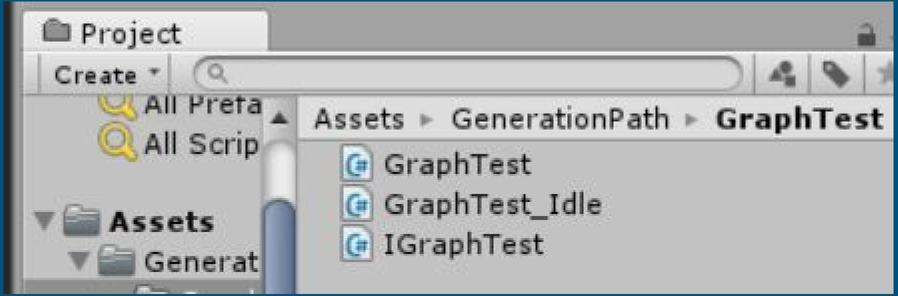
Scripts generation Folder: Assets/GenerationPath

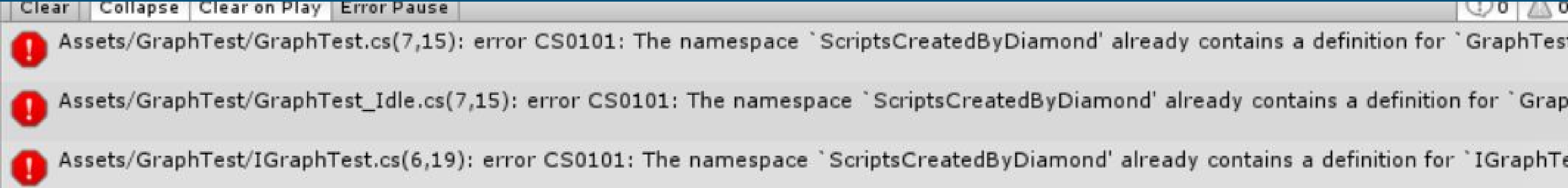
Textures generation Folder: Assets

Generate your C# scripts, where ?

Now generate your scripts again. Scripts are generated in the desired path but we have an error saying that we have already a definition of our classes.







```
Assets/GraphTest/GraphTest.cs(7,15): error CS0101: The namespace `ScriptsCreatedByDiamond' already contains a definition for `GraphTes
Assets/GraphTest/GraphTest_Idle.cs(7,15): error CS0101: The namespace `ScriptsCreatedByDiamond' already contains a definition for `Grap
Assets/GraphTest/IGraphTest.cs(6,19): error CS0101: The namespace `ScriptsCreatedByDiamond' already contains a definition for `IGraphTe
```

Generate your C# scripts, where ?

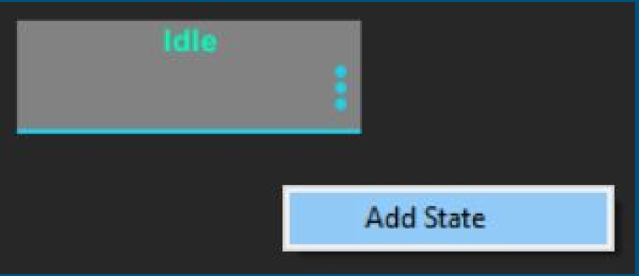
Remember that we have already generated the scripts for the same graph in another folder, that was the project root folder (slide).

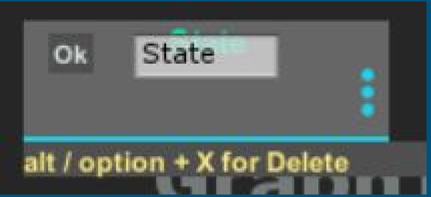
We haven't the right to create 2 classes with the same name in the same namespace in a C# solution.

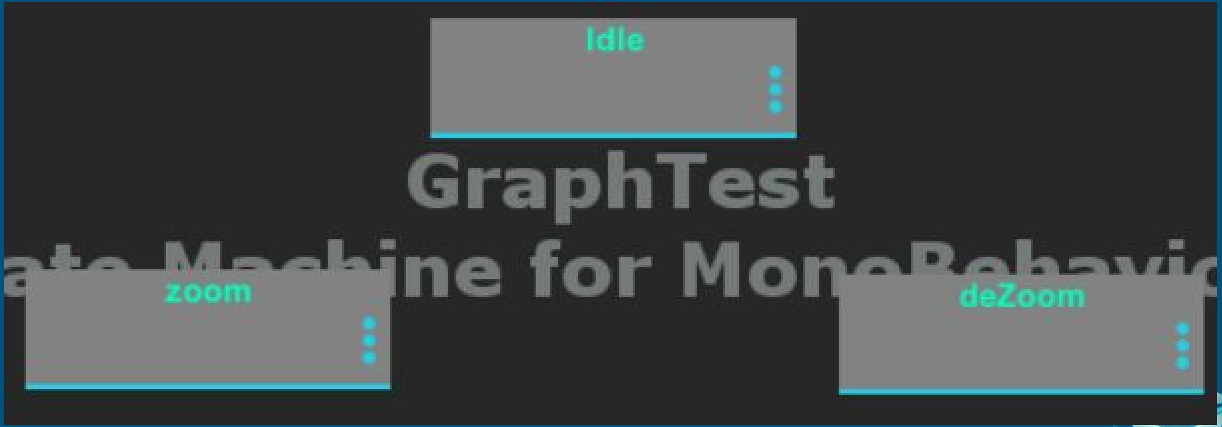
So if you want a new path to your scripts, simply delete the old ones and ask Diamond to generate your new scripts at the new path. If your script is already attached to a gameobject, after modifying its path, you need sometimes to reattach it.

Create a new state

In your state machine workspace (somewhere near the Idle state) right click and choose “Add State”. A new state is added asking you to rename it, you can simply press return (or click OK) if you want the default name. I named it “zoom”. Create another state and call it “deZoom”.





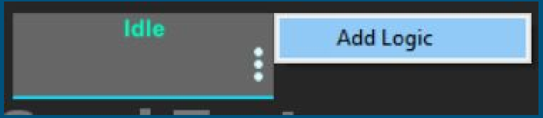


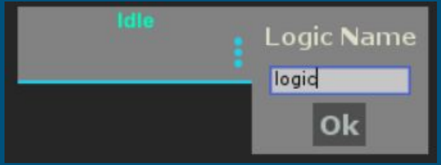
Create a new logic

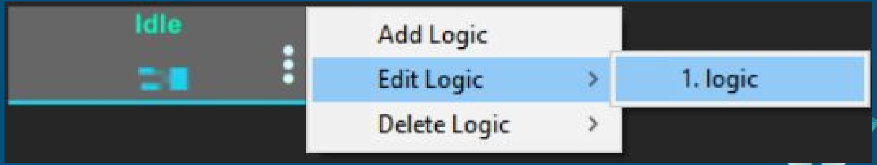
The default state is the Idle. letting the graph without transitions, means running only the Idle state even if another states existe. To go from the Idle to the deZoom state we have to create a logic in the Idle state. Click the Idle's option button and choose "Add Logic". Call your logic "logic". To edit your logic, click again the Idle's option button and choose "Edit Logic -> 1. logic".

Now the logic editor is opened (see the next slide)

Return to slide

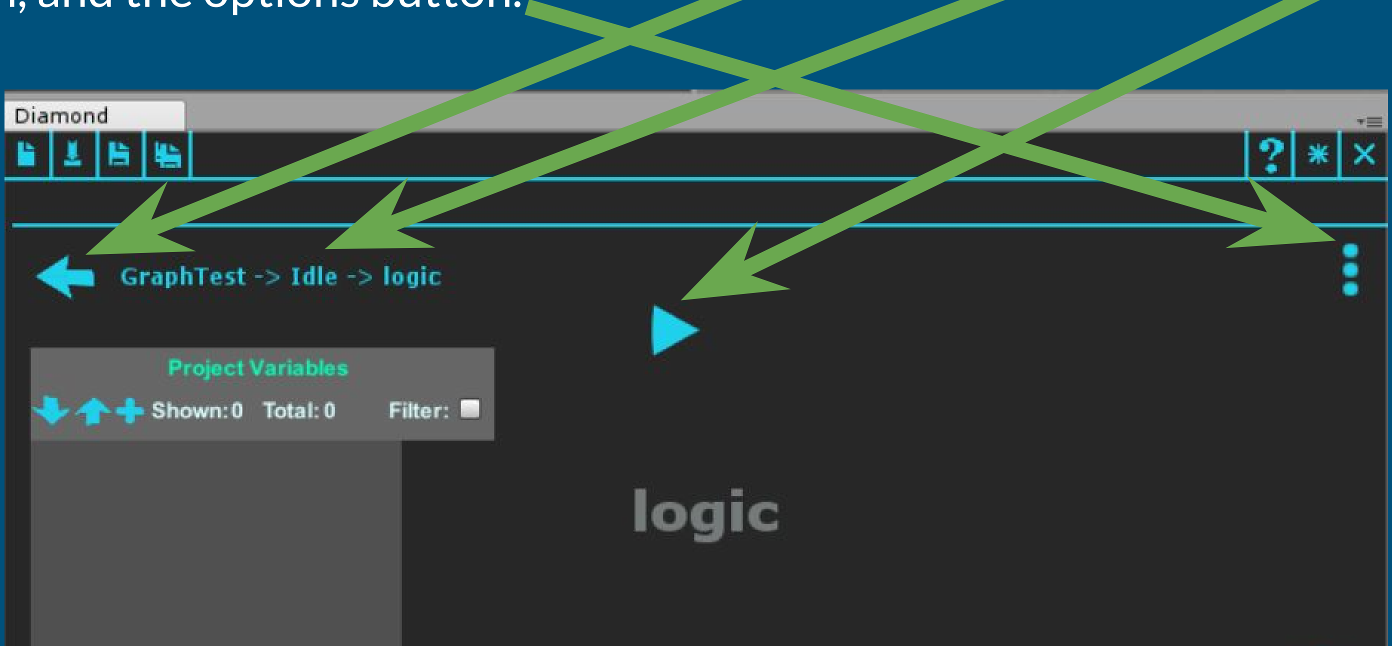






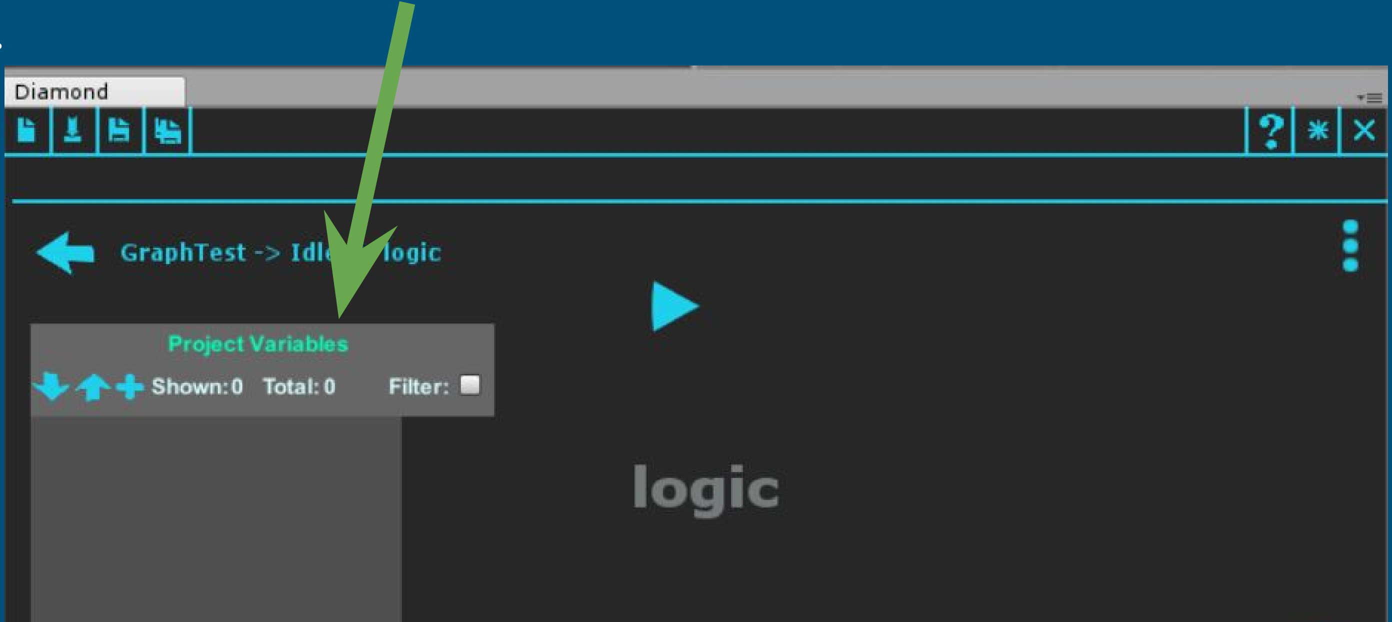
The logic editor

4 things to know about the logic editor: the back button, the address, the play button, and the options button.



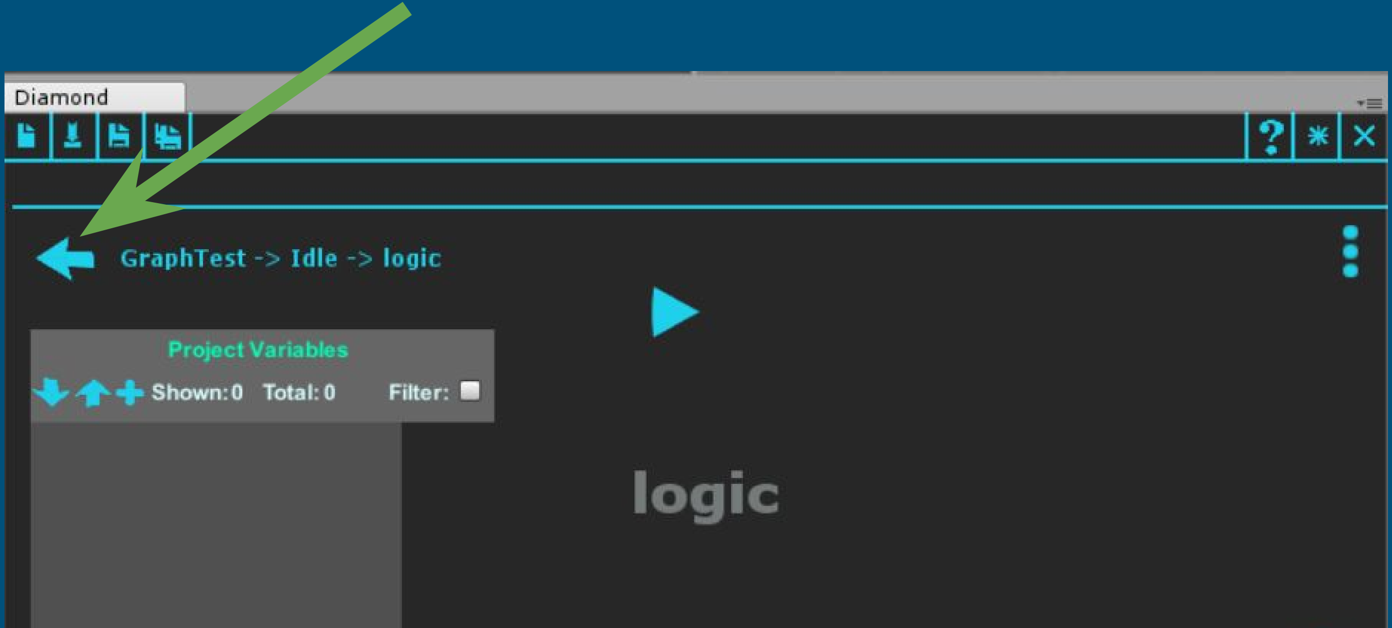
The logic editor

One more thing: the project variables. Variables that are static, so they concern the whole project and could be used to communicate data between logics.



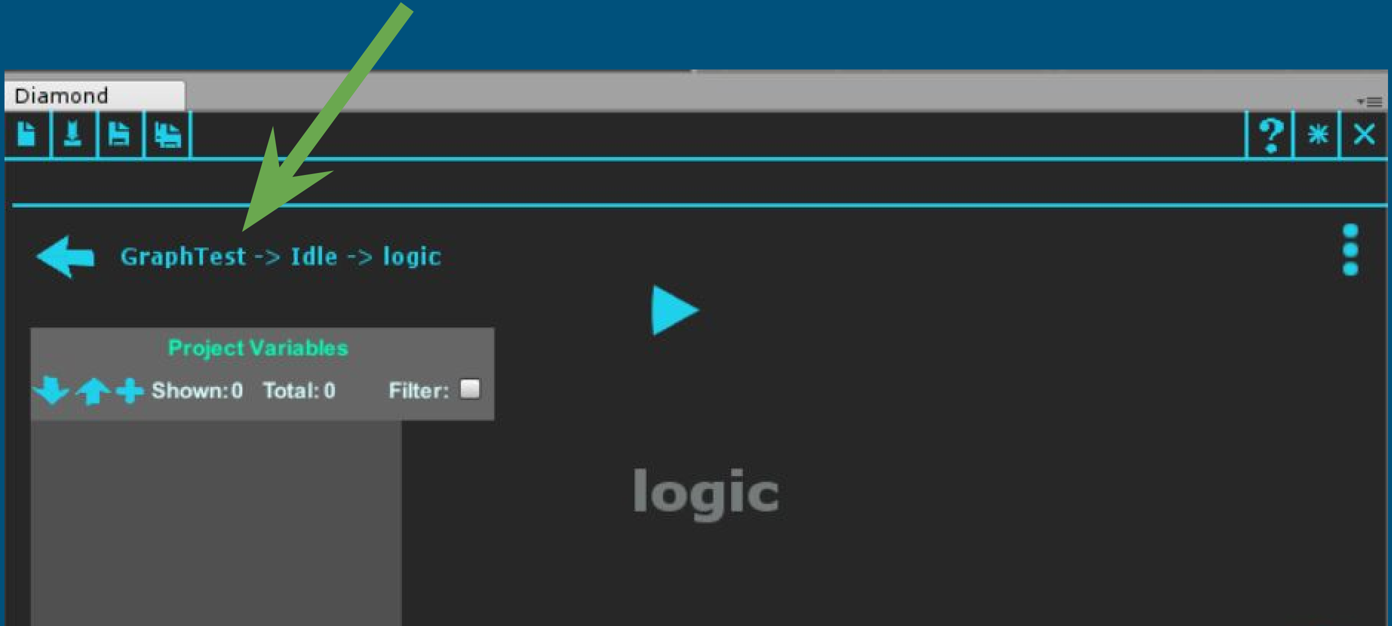
The logic editor - the back button

The back button is simply to return to the graph root (the states editor) where you had defined your states. Hotkey: alt / option + b



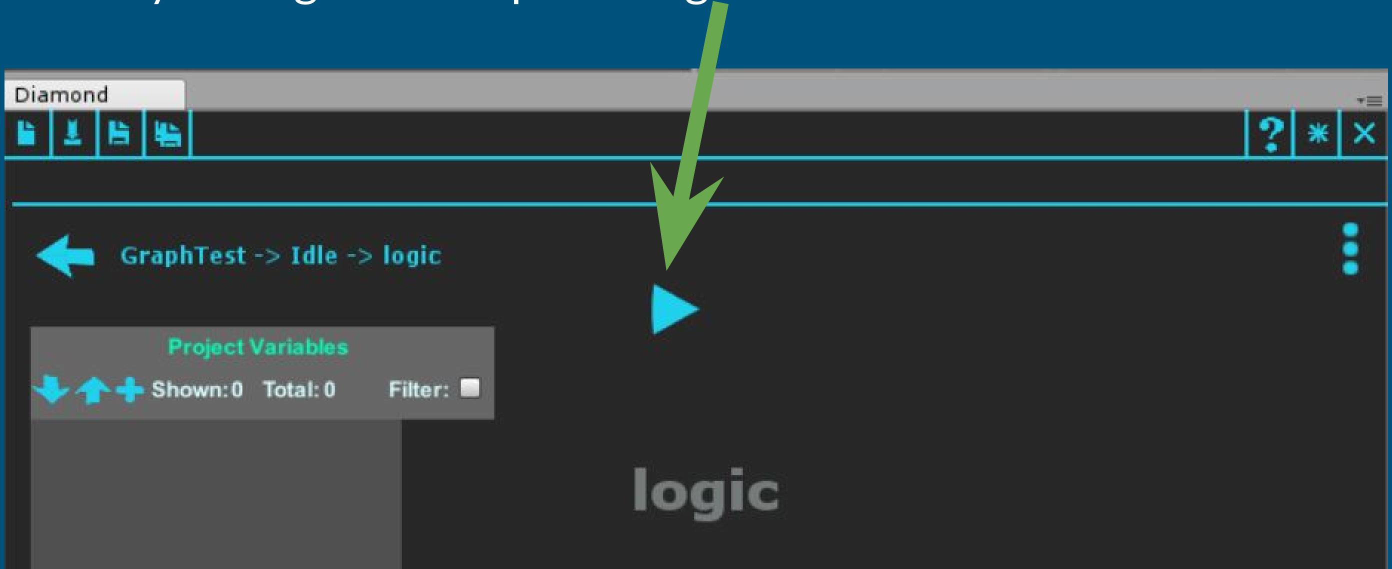
The logic editor - the address

The address indicates where you are, in this case, we are in the graph "GraphTest", in the state "Idle", and finally in the logic "logic".



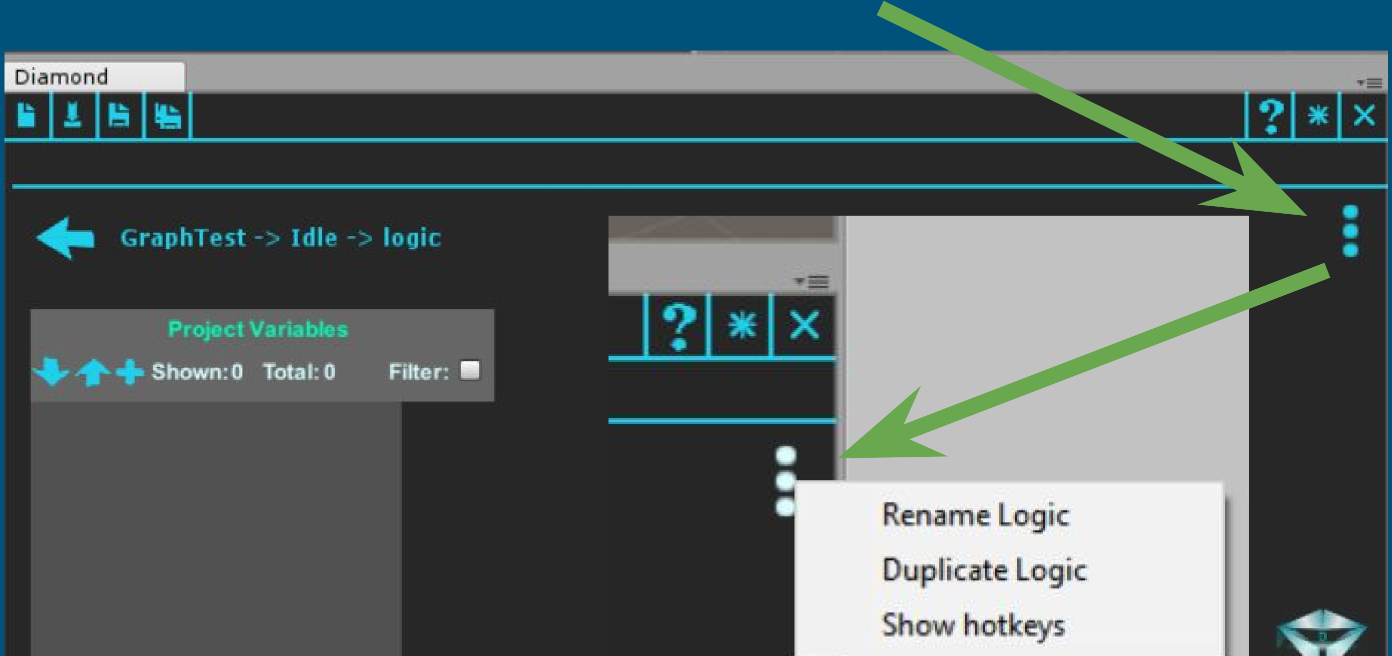
The logic editor - the play button

The play button help you testing your logic. So you can see the logic executed in front of your eyes before generating the scripts. If you return back to the states editor your logic will stop running



The logic editor - the options button

The options button let you rename, duplicate your logic, and show the hotkeys list.



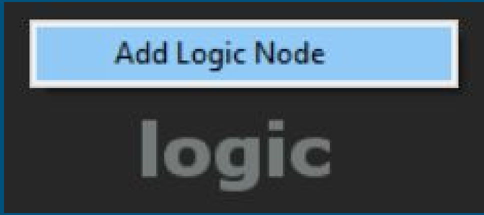
Create a new logic node

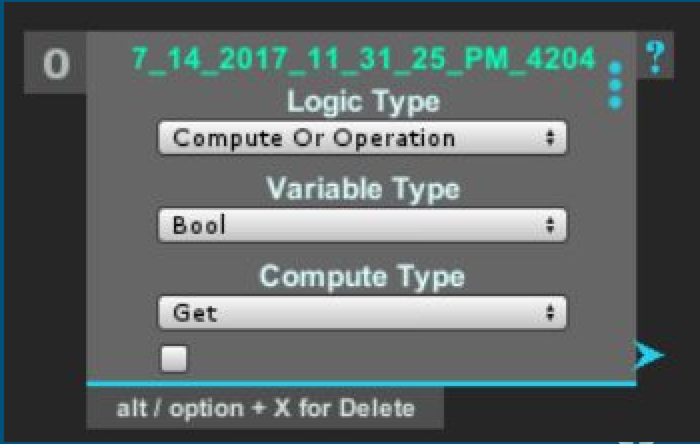
In your logic editor, right click and choose “Add Logic Node”.

After his name, the logic node has 3 stages:

1. Logic Type - what I want to do?
2. Variable Type - on which variable?
3. Compute Type - what to do with this variable?

Return to slide





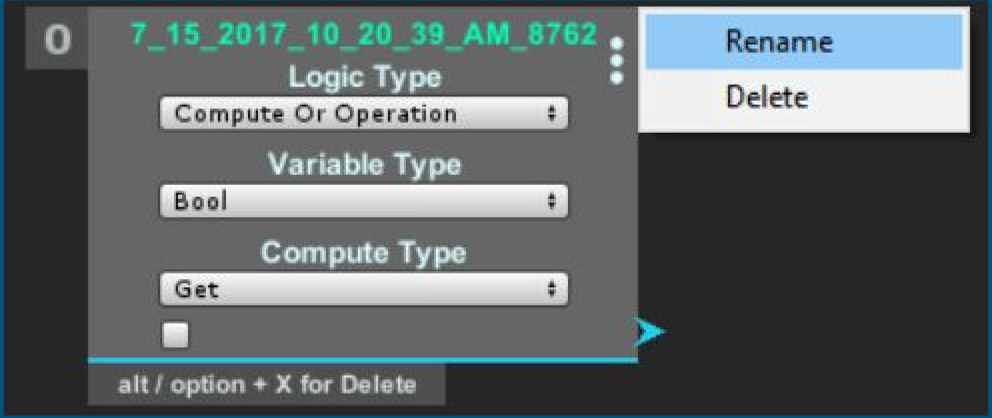
Logic Type
Compute Or Operation
Variable Type
Bool
Compute Type
Get

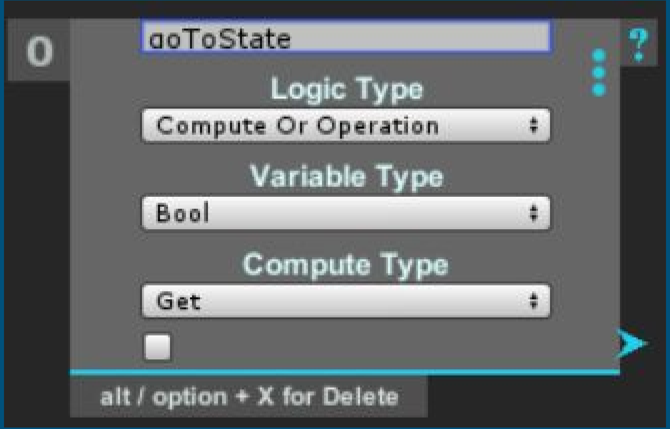
alt / option + X for Delete

Rename logic node

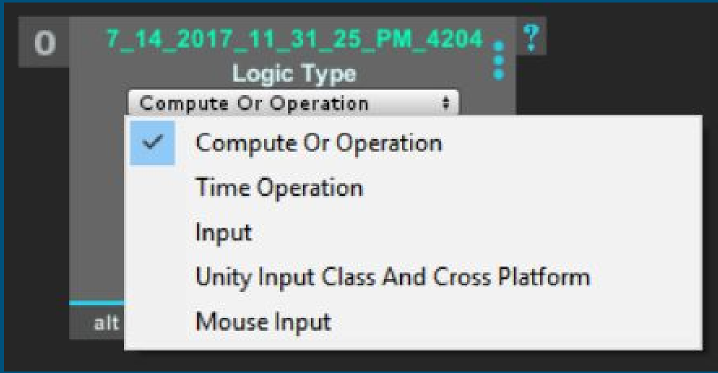
Click the logic node options button (top right) select rename and write goToState. Press the return key to confirm.

Return to slide





The logic node - the Logic Type


0
7_14_2017_11_31_25_PM_4204
Logic Type
Compute Or Operation
✓ Compute Or Operation
Time Operation
Input
Unity Input Class And Cross Platform
Mouse Input
alt

The Logic Type has 5 options:

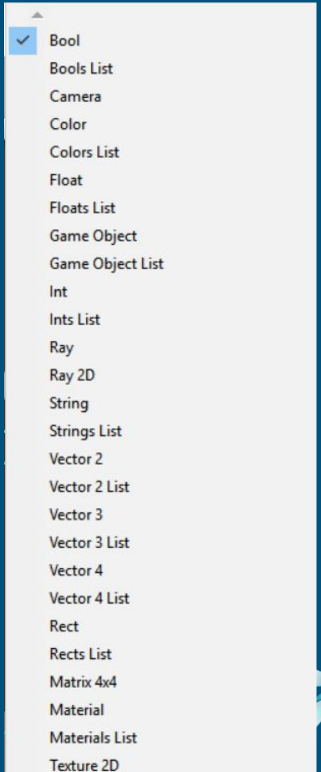
1. Compute Or Operation - the most common one, any kind of operation like addition for an int or changing the Field Of View of a camera
2. Time Operation - about time and frames,. To have access to the frame delta time, the time since level load, and perform periodic pulses, etc ..
3. Inputs - and cross platform ones, the simple Input is for desktop only, second one is for all inputs and the third one (Mouse Input) is to have the mouse position after freeing the cursor. Some First Person cameras lock the cursor to the screen and we have to free the cursor before getting its position

The logic node - the Variable Type

The Variable Type can be simple as a bool, float, int, Vector3 etc..

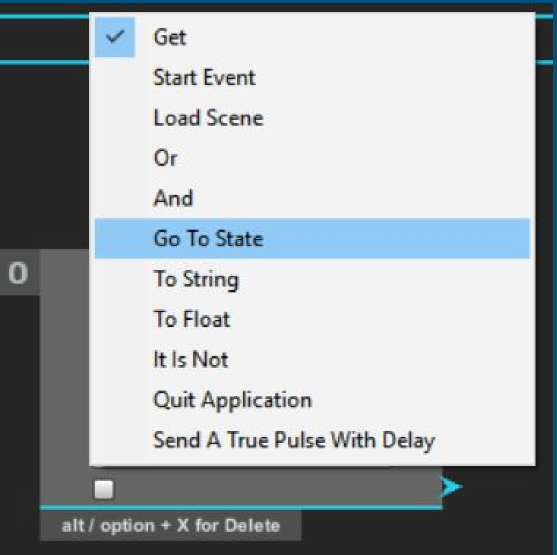
Or an object like a game object, a component transform, a camera, a Texture2D, etc..

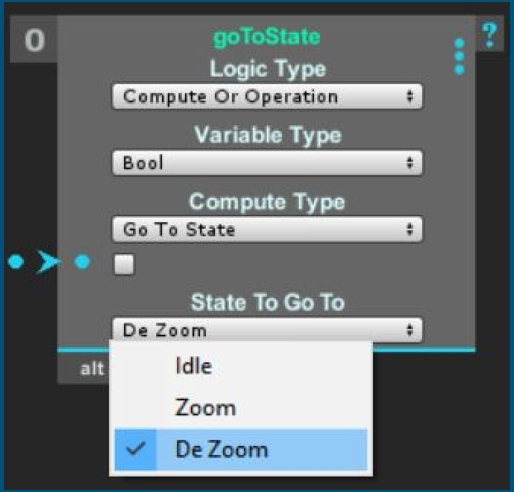
The combination of the Logic Type and the Variable Type choices tells to the Logic Node what Compute Type list to show to you



The logic node - the Compute Type

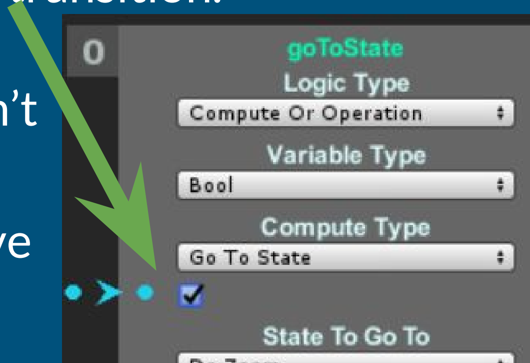
If we have selected Compute Or Operation for the Logic Type and Bool for the Variable Type, the Compute Type will show us this list in which I selected the “Go To State”. Is to make a transition to another state.




goToState
Logic Type: Compute Or Operation
Variable Type: Bool
Compute Type: Go To State
State To Go To: De Zoom

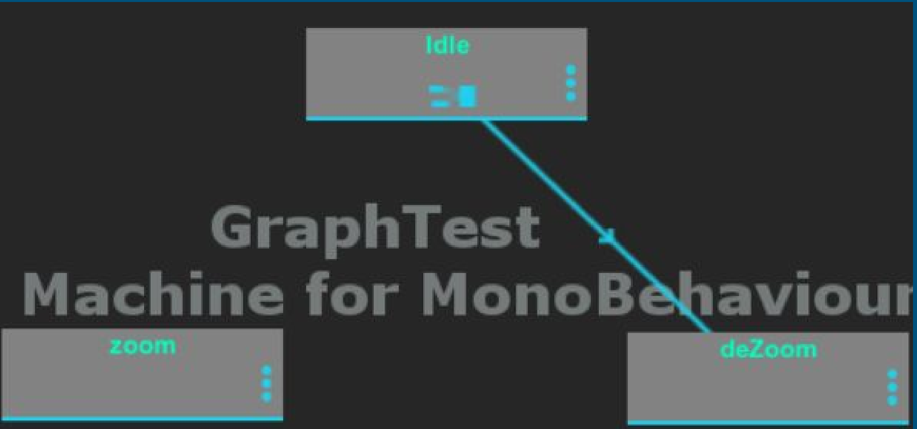
I selected to go to the deZoom state. I have to check the box to activate the transition.

P.S. If you don’t see other states list, save your graph


goToState
Logic Type: Compute Or Operation
Variable Type: Bool
Compute Type: Go To State
State To Go To

State’s Transition

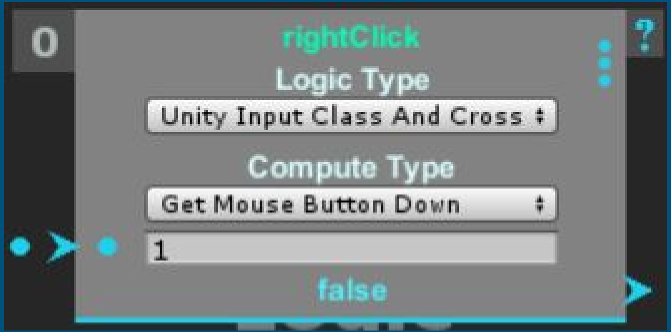
After creating the go to state logic node in the previous slide, return to the states editor (back button) and you will see an arrow indicating the transition. Idle is the first activated state, since we have checked the box of the goToState logic node (previous slide), at the first frame of the game, the transition will be executed and the deZoom state will become the activated one.



Even if the box in the goToState logic node isn’t checked, the transition in the left image will still showing itself indicating that potentially we have a transition between these states depending on the bool value (the checkbox) of the goToState logic node.

Zoom Camera - Right Click

After creating the Idle->deZoom transition, let's dive in the deZoom state. Create and edit a logic in the deZoom state (like in slide). Add a logic node (like in slide), rename it rightClick (like in slide), for its Logic Type select Unity Input Class And Cross Platform, for its Compute Type select Get Mouse Button Down and in the field enter 1 (0->left, 1->right, and 2->middle, you can go to 6 for more complex mouses).

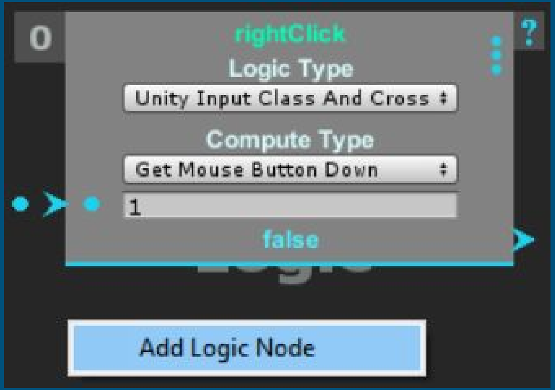


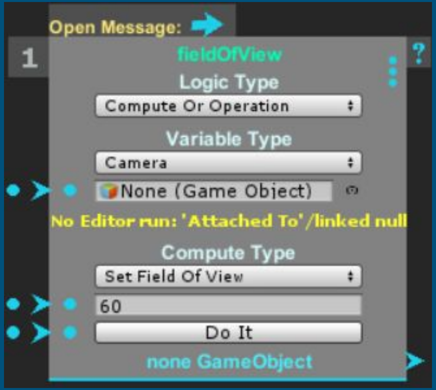
0
rightClick
Logic Type: Unity Input Class And Cross
Compute Type: Get Mouse Button Down
1
false

This logic node means: when the right mouse is clicked down, the output will switch to True once.

Zoom Camera - Field OF View

Near to your rightClick logic node add a new one and rename it fieldOfView. In Logic Type select Compute Or Operation, in Variable Type select Camera, in Compute Type select Set Field Of View and fill the field by 60.


0
rightClick
Logic Type
Unity Input Class And Cross
Compute Type
Get Mouse Button Down
1
false

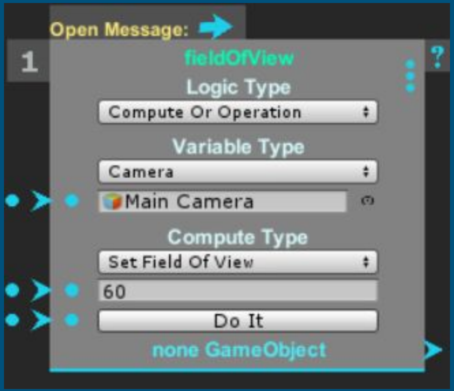

Open Message:
1
fieldOfView
Logic Type
Compute Or Operation
Variable Type
Camera
None (Game Object)
No Editor run: 'Attached To'/linked null
Compute Type
Set Field Of View
60
Do It
none GameObject

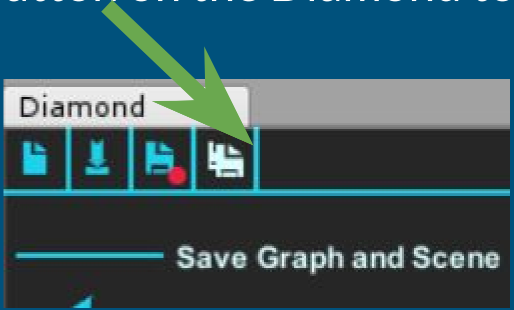
This logic node means: search a camera component in the gameobject and put its field of view to 60 degrees. We are in the deZoom state so we put the field of view to its default value (60).

Zoom Camera - Fill the GameObject Field

Return to slide

From your hierarchy, drag the main camera and drop it in our logic node gameobject field. Save both your Diamond graph and your scene by clicking in the double save button on the Diamond top toolbar.

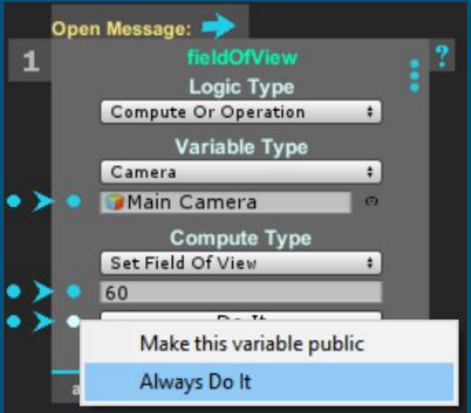


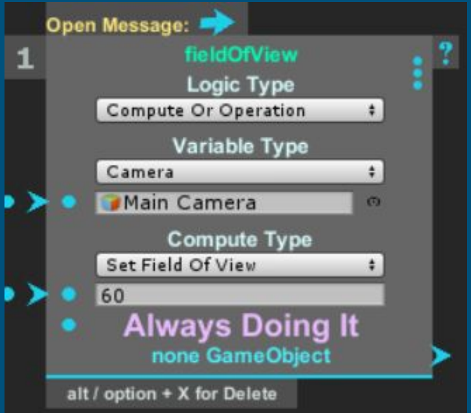


Even if the double save button isn’t red, you need to click it to save your graph and to force saving the scene. This tells the scene to assign the Main Camera to the graph field, so after closing and reopening Unity, the Main Camera will still in the graph field.

Zoom Camera - Always Do IT

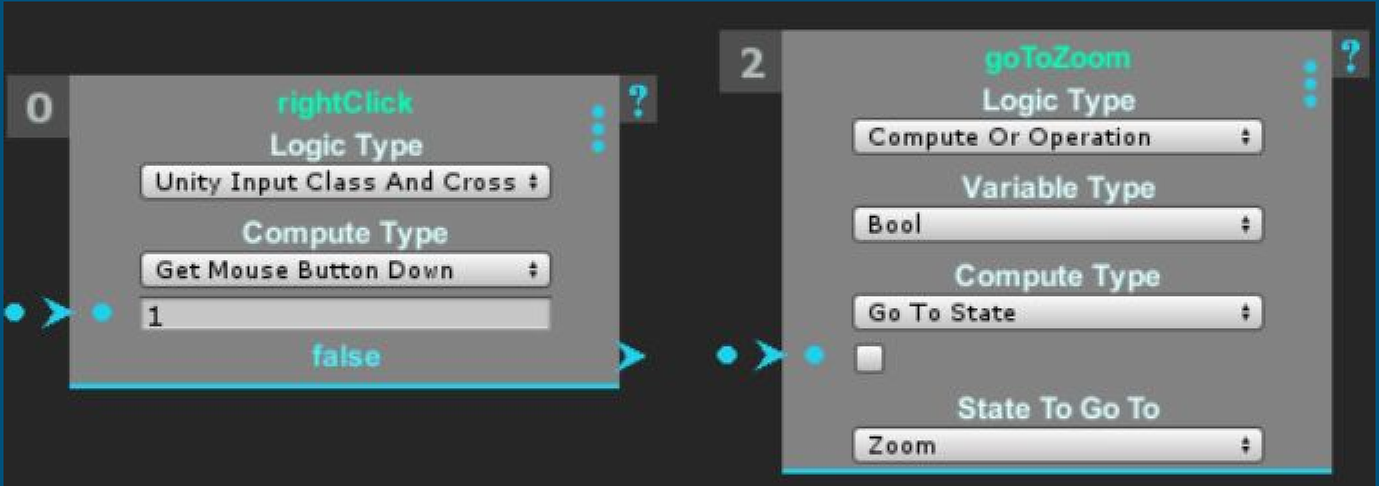
Click the options dot at the right of your Do It button, select Always Do It. Do It button control whether or not doing the logic node actions. Our case means: Whenever you are in the deZoom state, you have to put 60 in the Main Camera's field of view.


Open Message:
1
fieldOfView
Logic Type: Compute Or Operation
Variable Type: Camera
Main Camera
Compute Type: Set Field Of View
60
Make this variable public
Always Do It


Open Message:
1
fieldOfView
Logic Type: Compute Or Operation
Variable Type: Camera
Main Camera
Compute Type: Set Field Of View
60
Always Doing It
none GameObject
alt / option + X for Delete

Zoom Camera - Go To Zoom

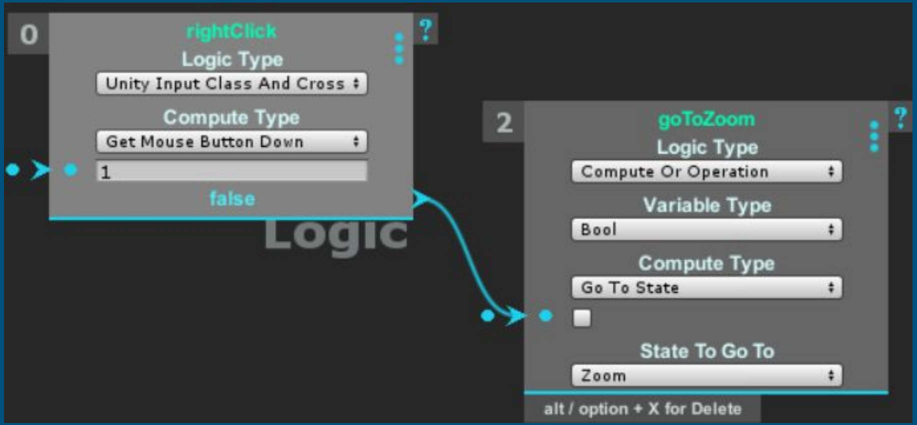
Near to the rightClick logic node, add a new logic node rename it goToZoom and fill its fields like showing.


0
rightClick
Logic Type
Unity Input Class And Cross
Compute Type
Get Mouse Button Down
1
false

2
goToZoom
Logic Type
Compute Or Operation
Variable Type
Bool
Compute Type
Go To State
State To Go To
Zoom

Zoom Camera - Go To Zoom - Link

Click once at the output of the rightClick logic node, move to the input (the triangle) of the goToZoom logic node (no need to drag). This logic means: when I right click one, I will active the transition deZoom->zoom to zoom in the camera.



0 rightClick

Logic Type: Unity Input Class And Cross

Compute Type: Get Mouse Button Down

1

false

2 goToZoom

Logic Type: Compute Or Operation

Variable Type: Bool

Compute Type: Go To State

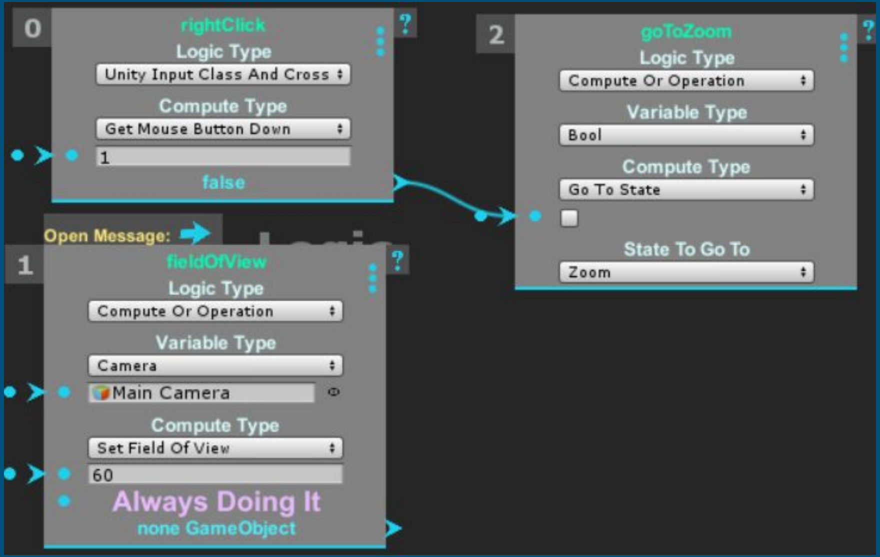
State To Go To: Zoom

alt / option + X for Delete

Both input and output are boolean. Don't worry about types compatibilities, Diamond won't accept to connect incompatible types.

Zoom Camera - deZoom logic

Our deZoom logic is complete now and should look like this.



0 rightClick
Logic Type: Unity Input Class And Cross
Compute Type: Get Mouse Button Down
1
false

Open Message:

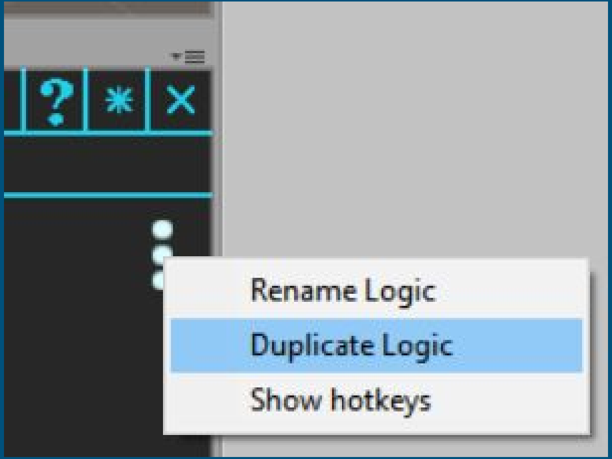
1 fieldOfView
Logic Type: Compute Or Operation
Variable Type: Camera
Main Camera
Compute Type: Set Field Of View
60
Always Doing It
none GameObject

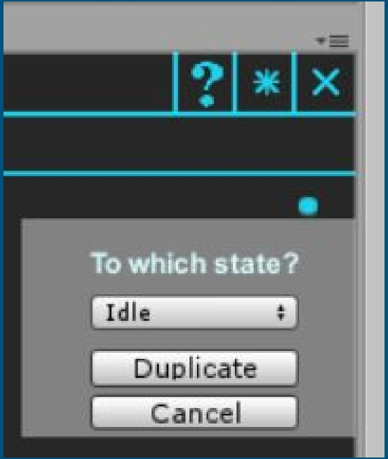
2 goToZoom
Logic Type: Compute Or Operation
Variable Type: Bool
Compute Type: Go To State
State To Go To: Zoom

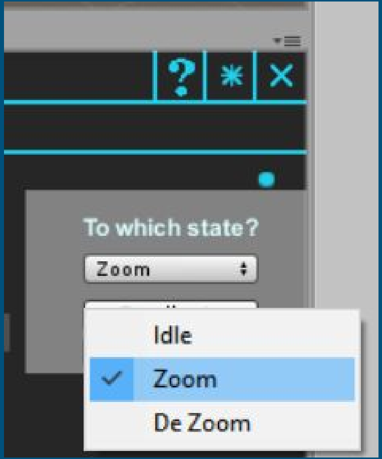
A logic node for the right click input, another to set the camera field of view, and the last one to go to the zoom state.

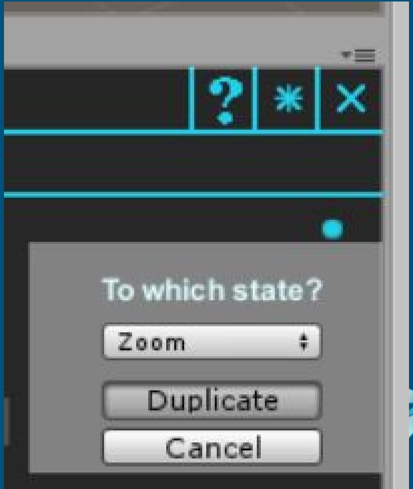
Zoom Camera - duplicate logic

Since our zoom logic is similar to the deZoom one, we will duplicate the deZoom logic. Choose to duplicate it to the zoom state.



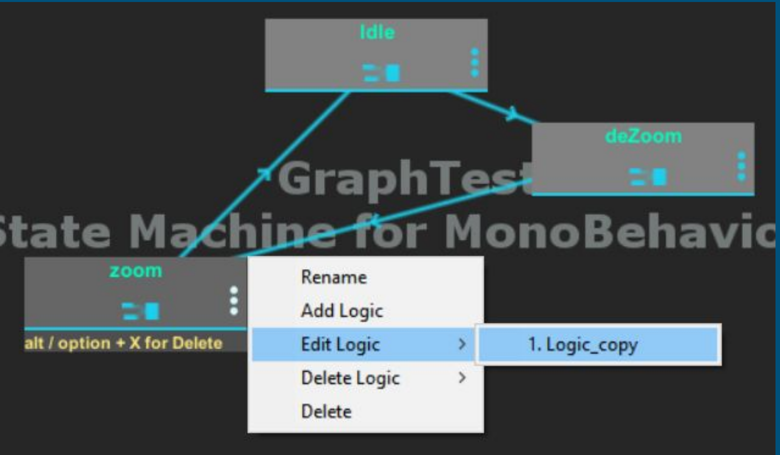


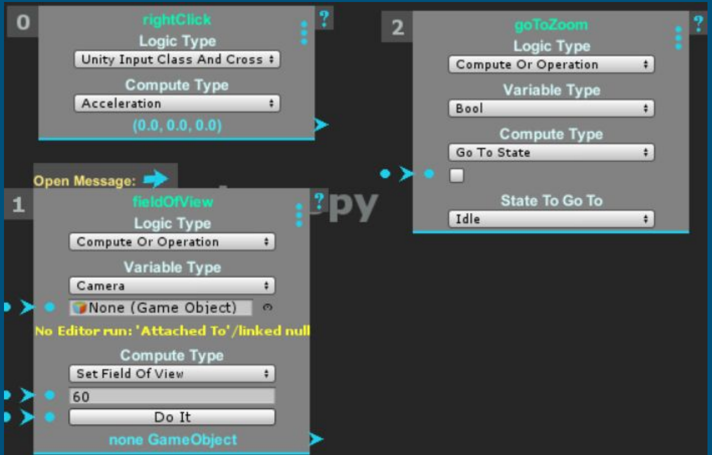




Zoom Camera - duplicate logic

Now return to the states editor (back button or alt / option + b) and click the options button of the zoom state. You will see our logic copy in the zoom state, edit it. You see the logic copied but need some modification (next slide).

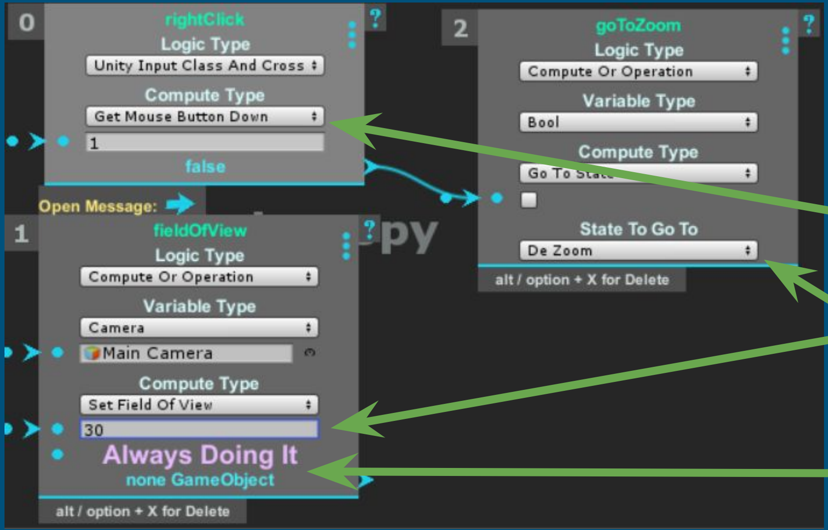




Zoom Camera - duplicate logic

Now, modify the duplicated logic like showing.

This state zoom in, to zoom in you need a lower camera's field of view



0 rightClick
Logic Type: Unity Input Class And Cross
Compute Type: Get Mouse Button Down
1
false
Open Message:

1 fieldOfView
Logic Type: Compute Or Operation
Variable Type: Camera
Main Camera
Compute Type: Set Field Of View
30
Always Doing It
none GameObject
alt / option + X for Delete

2 goToZoom
Logic Type: Compute Or Operation
Variable Type: Bool
Compute Type: Go To State
State To Go To: De Zoom
alt / option + X for Delete

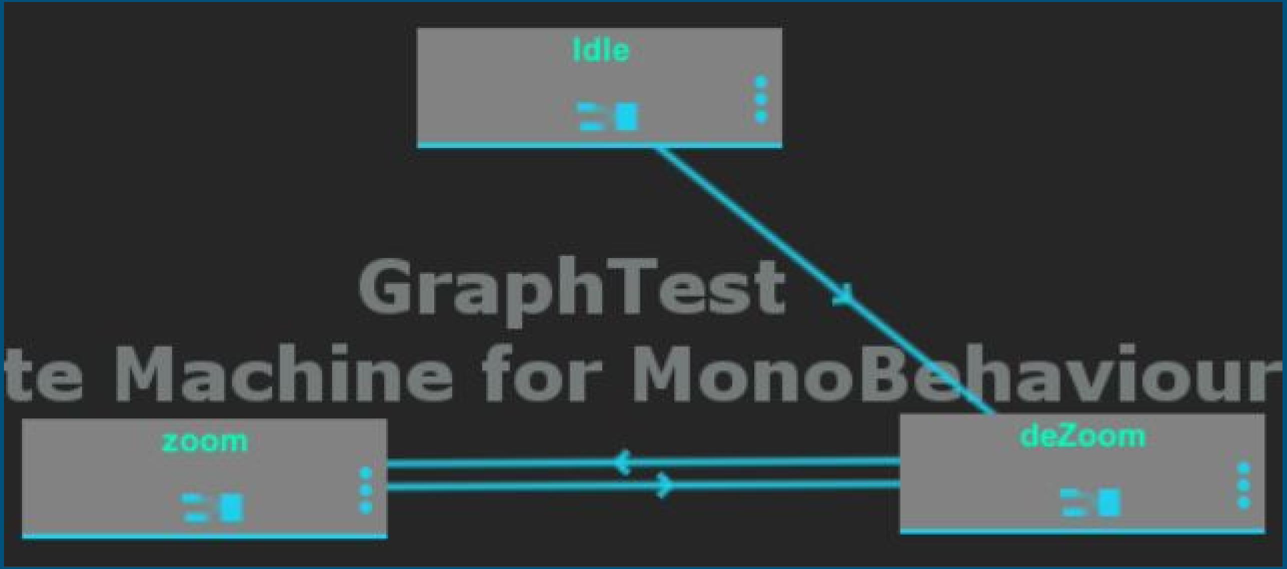
3 modifications to do:

1. Get Mouse Button Down
2. Set the Field Of View to 30
3. Set the destination state to De Zoom

Don't forget to always do it

Zoom Camera - Global View

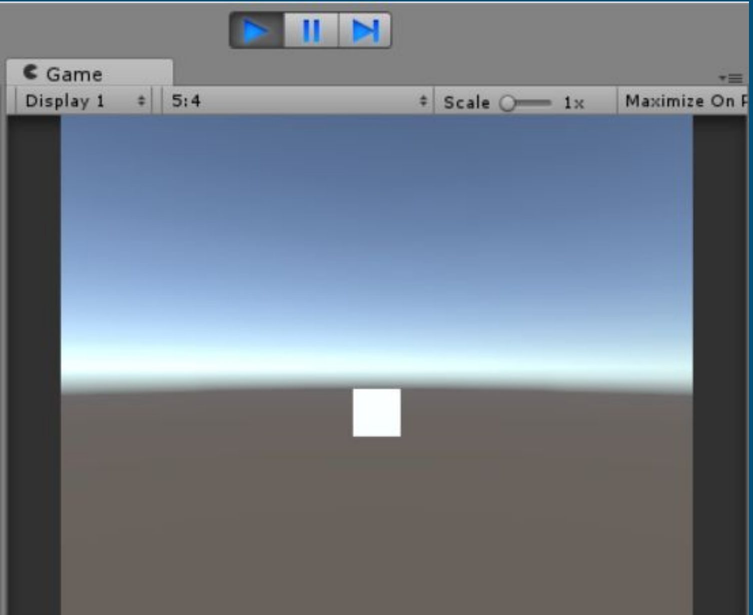
By returning to our states editor notice the 3 transitions:

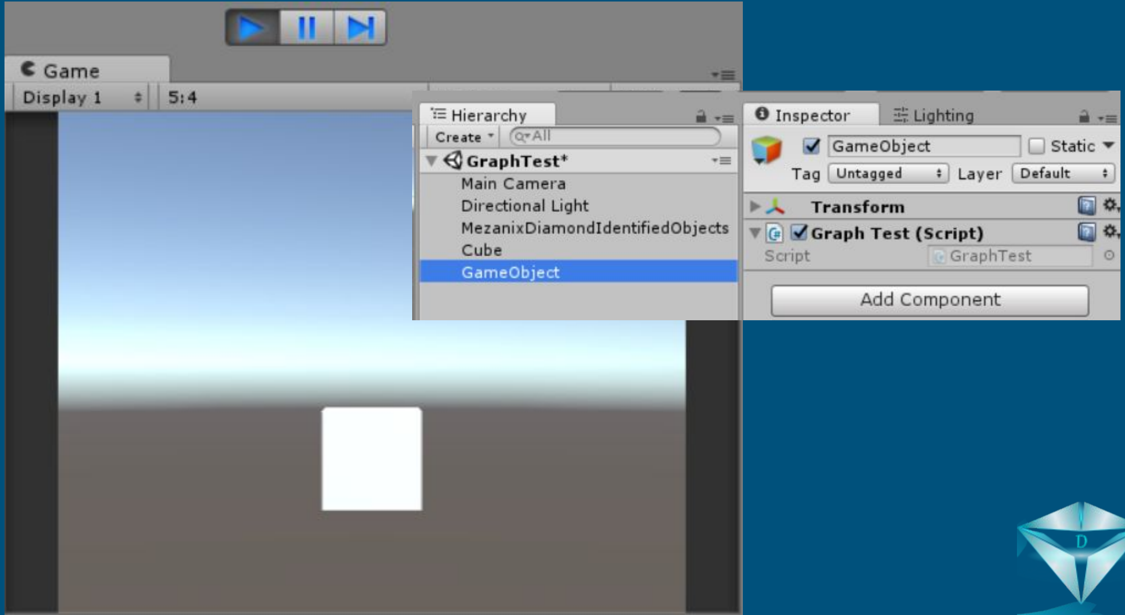


1. Idle->deZoom - on the game start
2. deZoom->zoom and inversely on right mouse click.

Zoom Camera - Apply

Now that our graph is done, write C# scripts (like slide). In your hierarchy, create a cube, put it in front of your Main Camera. Create also an empty gameobject and attach to it the GraphTest.cs script. Play the scene, left click in the Game window to activate it. Now clicking on the right click switches between zoom in / out of the Main Camera.



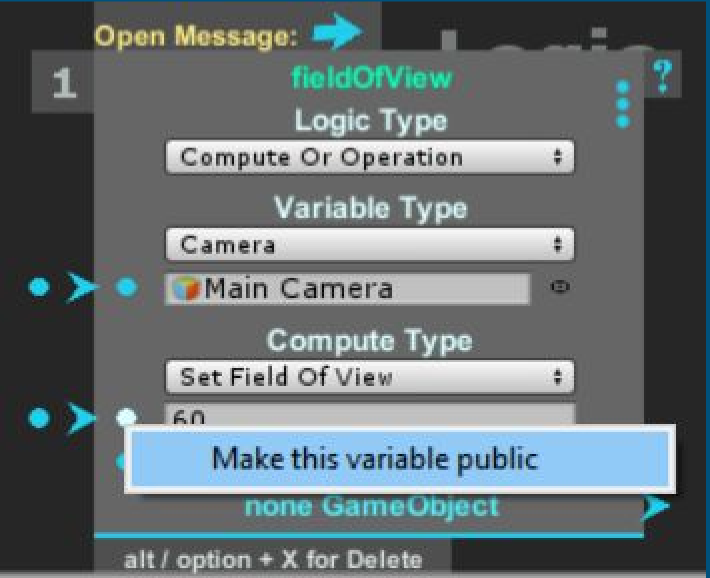


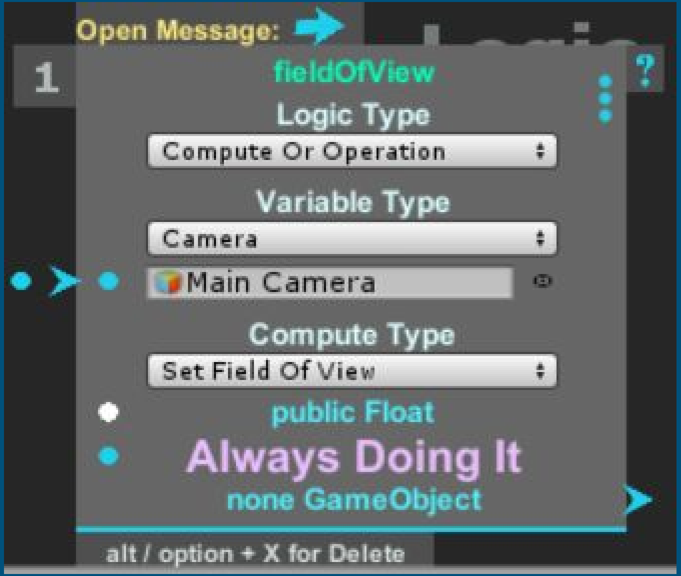
GraphTest*
Main Camera
Directional Light
MezanixDiamondIdentifiedObjects
Cube
GameObject

GameObject
Tag: Untagged Layer: Default
Transform
Graph Test (Script)
Script: GraphTest

Zoom Camera - Public variables

We have finished the zoom camera feature. But Diamond graphs can be made in multiple ways. In the zoom camera, we had 2 values for the field of view, 30 for zoom-in (zoom) and 60 for zoom-out (deZoom). But these 2 values still in the graph, sometimes we need to modify variables directly in the generated script, so we need to make these variables public.


Open Message:
1
fieldOfView
Logic Type
Compute Or Operation
Variable Type
Camera
Main Camera
Compute Type
Set Field Of View
60
Make this variable public
none GameObject
alt / option + X for Delete

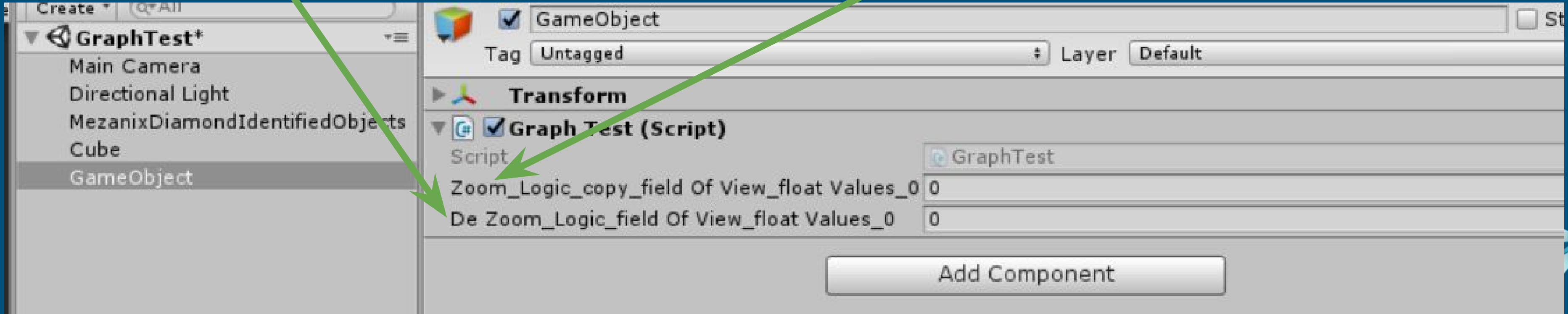

Open Message:
1
fieldOfView
Logic Type
Compute Or Operation
Variable Type
Camera
Main Camera
Compute Type
Set Field Of View
public Float
Always Doing It
none GameObject
alt / option + X for Delete

Go to the deZoom state, edit the logic and in the fieldOfView logic node make the variable public by clicking on the options dot at the right of the variable.

Zoom Camera - Public variables

The same thing you had done in the previous slide, now do it to the zoom state, to make the variable of the fieldOfView logic node public. Generate the scripts, and you will see 2 public variables in the script inspector of the gameobject. The public variable name is composed like so: “stateName” _ “logicName” _ “logicNodeName” _ “variableName”.

By the state names we can identify that the first one is for the zoom state and the second for the deZoom state.

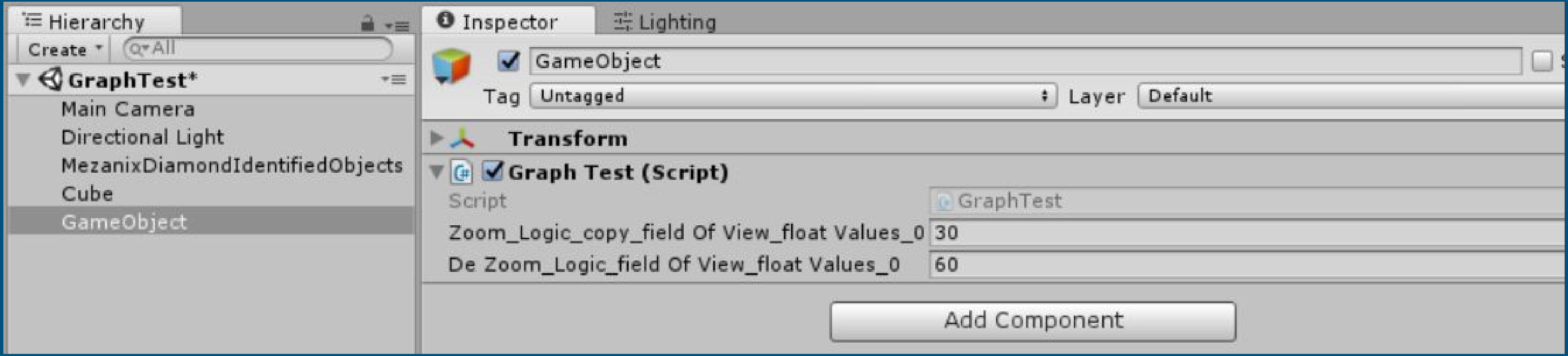


Zoom_Logic_copy_field Of View_float Values_0 0
De Zoom_Logic_field Of View_float Values_0 0

Zoom Camera - Public variables

So now, independently of your Diamond graph, you can iterate and modify your variables without the need to reopen your graph or regenerate your script.

Diamond is a script generator, once you are happy with your graph, you generate your script and you continue dealing with your script.



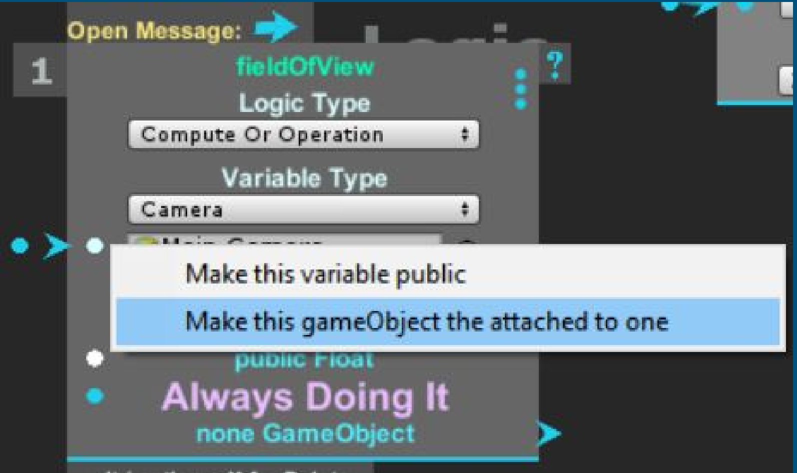
GameObject

Tag Untagged Layer Default

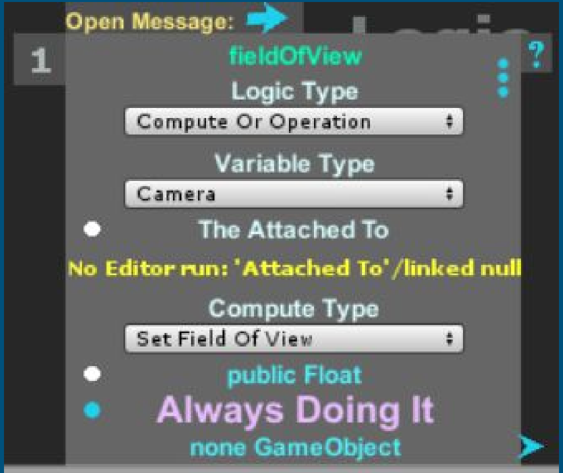
Script	GraphTest
Zoom_Logic_copy_field Of View_float Values_0	30
De Zoom_Logic_field Of View_float Values_0	60

Zoom Camera - Script on Camera

In our example of zoom camera we had filled the gameobject field by the main camera (slide). We had generated the script, attached it to an empty gameobject, and the script was acting in the main camera because it was filled by this camera. We can achieve the same logic without need to fill any gameobject field. Since we know that the script will act on the camera, we can attach it to the camera and tell to it to act on the gameobject holding it. To do so, go in both states, zoom and deZoom, edit the logic and in the fieldOfView logic node make the game object field "The Attached To One".



Open Message:
1
fieldOfView
Logic Type
Compute Or Operation
Variable Type
Camera
Make this variable public
Make this gameObject the attached to one
public Float
Always Doing It
none GameObject



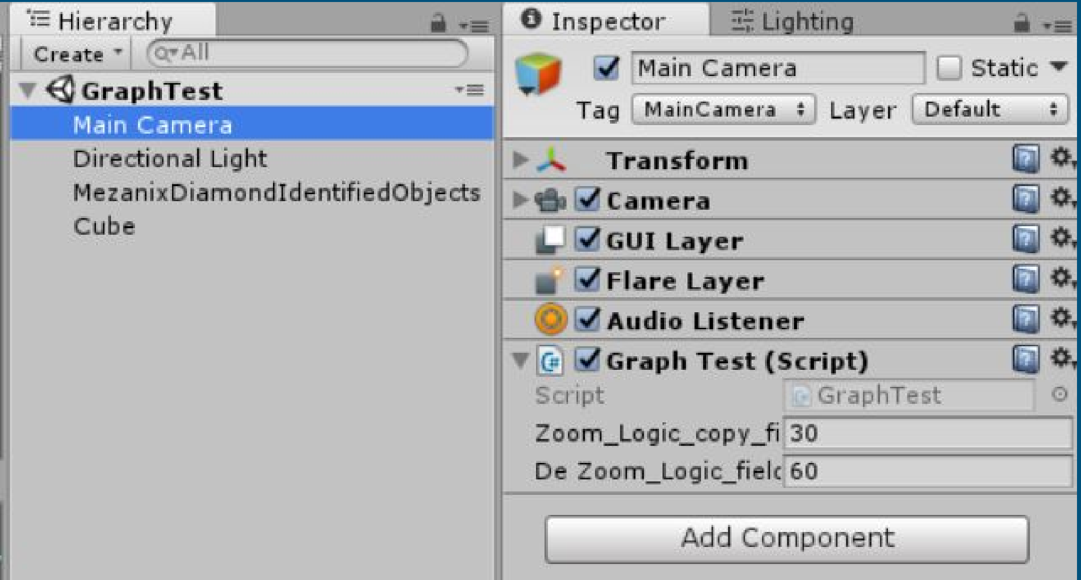
Open Message:
1
fieldOfView
Logic Type
Compute Or Operation
Variable Type
Camera
The Attached To
No Editor run: 'Attached To'/linked null
Compute Type
Set Field Of View
public Float
Always Doing It
none GameObject

To make an "Attached To field" simply click the dot options at the right of the gameobject field

Zoom Camera - Script on Camera

Now generate your script, delete your empty gameobject from the scene (the one that was holding the GraphTest.cs script), and attach the script on the main camera.

Now you have a good script, with editable variables, and acting on its gameobject's camera. So this script, you can put it in another scene or another project and attach it to a camera and it will work without need to use Diamond graph again.



GraphTest
Main Camera
Directional Light
MezanixDiamondIdentifiedObjects
Cube

Main Camera ✓ — Static
Tag MainCamera — Layer Default
Transform
✓ Camera
✓ GUI Layer
✓ Flare Layer
✓ Audio Listener
✓ Graph Test (Script)
Script GraphTest
Zoom_Logic_copy_fi 30
De Zoom_Logic_fielc 60

Conclusion

You know now how Diamond works. You can have further information here

http://mezanix.com/portfolio/diamond-documentation/

Or see other files at:

Assets / Mezanix / Diamond / 0_Documentation

Diamond 1.1

Visual Scripting for Unity
Documentation
New Features of the 1.1 Version

www.mezanix.com

Before Diving in Diamond

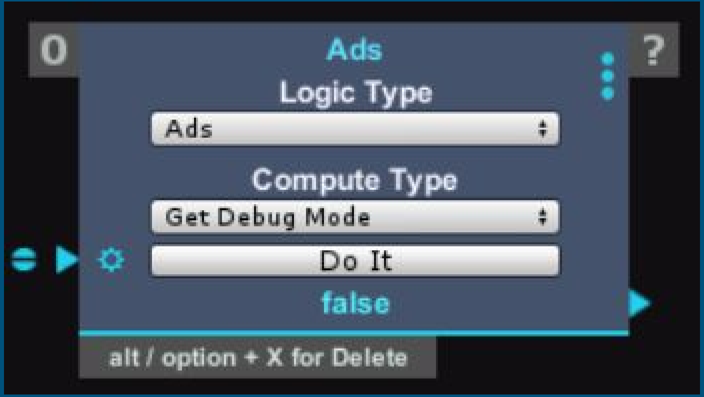
Diamond use the State Machine concept. You can have an idea about it here

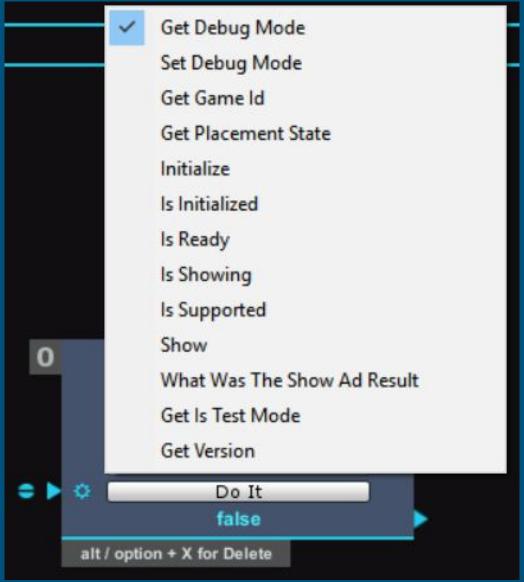
https://en.wikipedia.org/wiki/Finite-state_machine

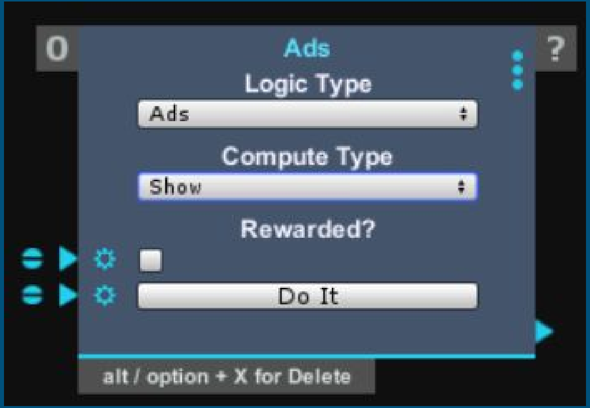
Unity Ads

Diamond 1.1

Configure your logic node like so: Logic Type = Ads.
In the Compute Type you have the list of Ads functions. The most important one is 'Show'. Is to show video Ads, you can show rewarded or not rewarded video ads.

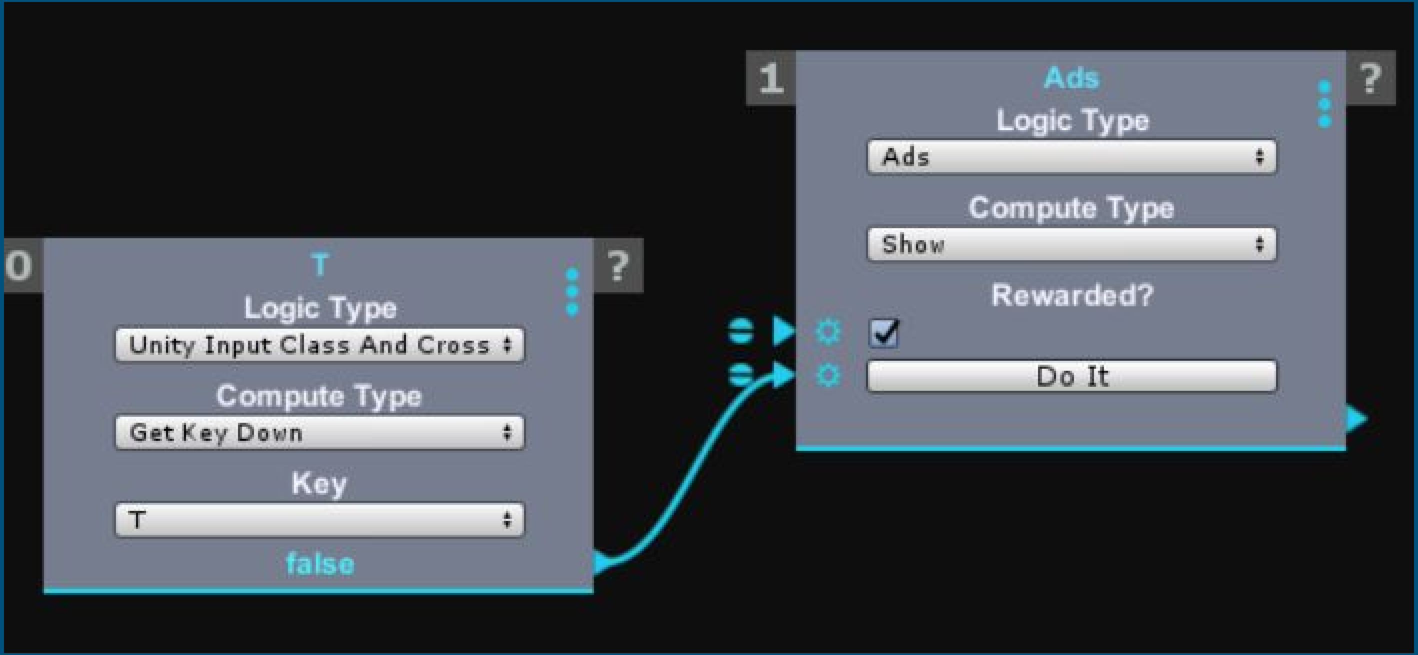






Unity Ads Example 1/2

Configure 2 logic nodes like below. In your build settings switch to a mobile platform. Generate your script and attach it to any gameobject.

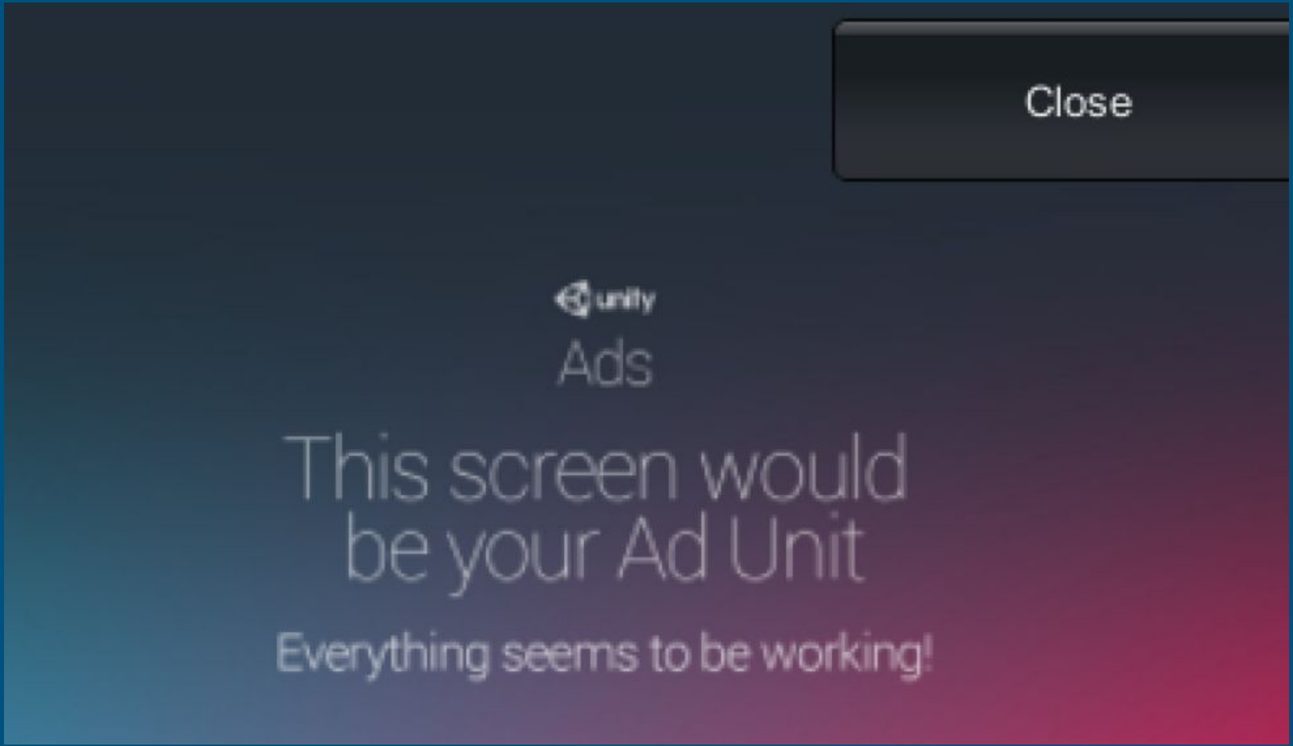


0
T
Logic Type: Unity Input Class And Cross
Compute Type: Get Key Down
Key: T
false

1
Ads
Logic Type: Ads
Compute Type: Show
Rewarded? ☑
Do It

Unity Ads Example 2/2

Play your scene click inside the game window and press T. You will see this screen showing that your test is ok.



This screen would be your Ad Unit

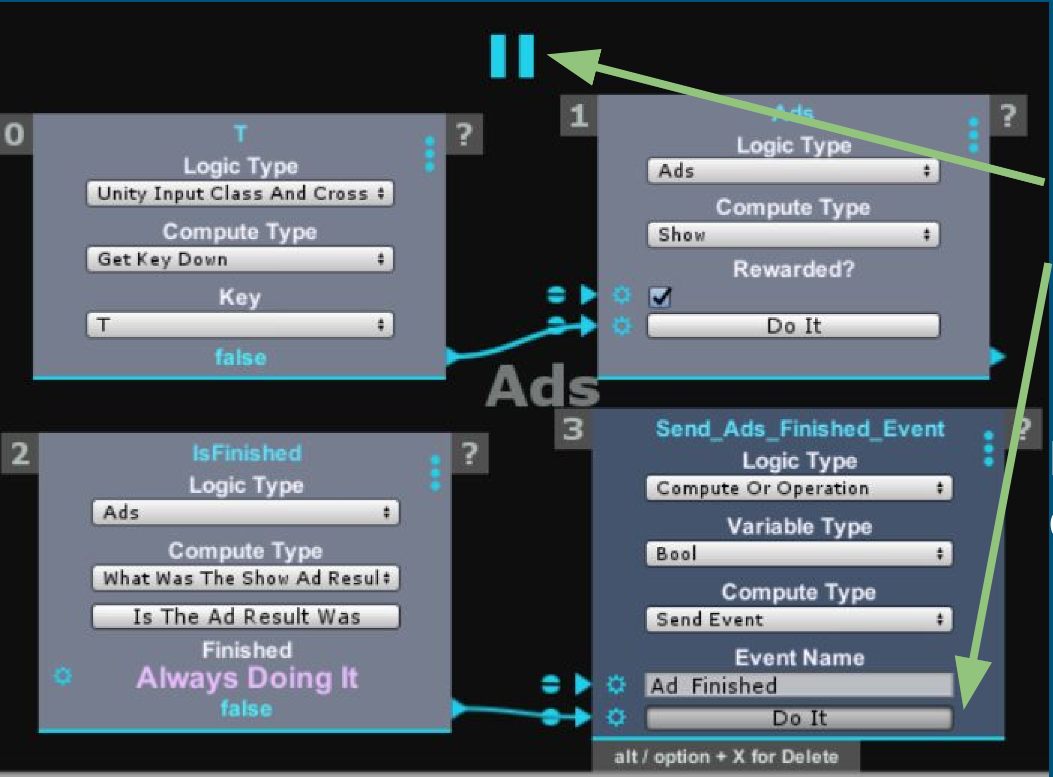
Everything seems to be working!

Return to Slide

Unity Ads. Reward the player 1/3

Diamond 1.1

Use Event System (not only for Ads, event system is a broadcasting to let graphs talk to each other)



Continue configuring your logic like in the image, play the logic in Diamond and click on the Do It button of the 'Sent_Ads_Finished_Events' node, so this event is registered.

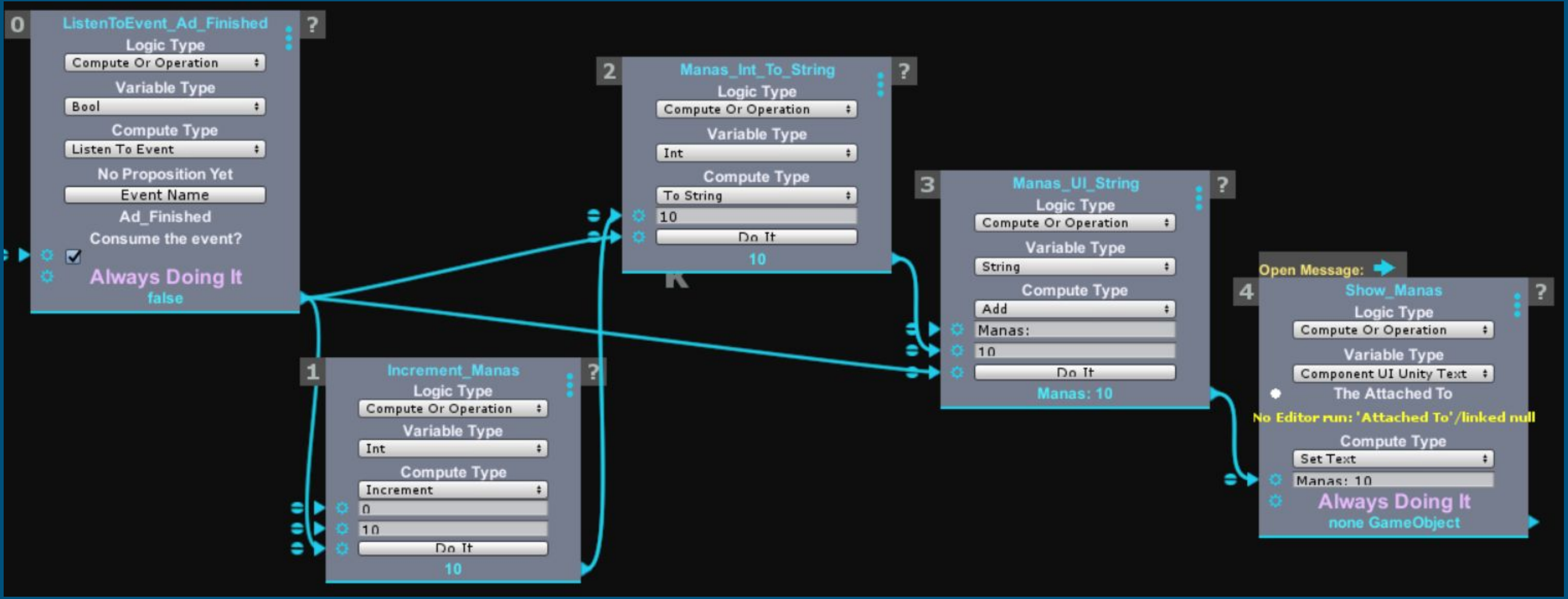
Now return to the states view and write C# scripts.

Unity Ads. Reward the player 2/3

Diamond 1.1

Use Event System

Create a new Diamond graph in which create a logic. Configure the logic like below. Write script and attach it to a Unity UI text in your scene. Play in Unity Editor, press T key and you will see the image of this slide, click on the Close button and see the result (next slide)



0 ListenToEvent_Ad_Finished
Logic Type: Compute Or Operation
Variable Type: Bool
Compute Type: Listen To Event
No Proposition Yet
Event Name: Ad_Finished
Consume the event? ✓
Always Doing It
false

1 Increment_Manas
Logic Type: Compute Or Operation
Variable Type: Int
Compute Type: Increment
0
10
Do It
10

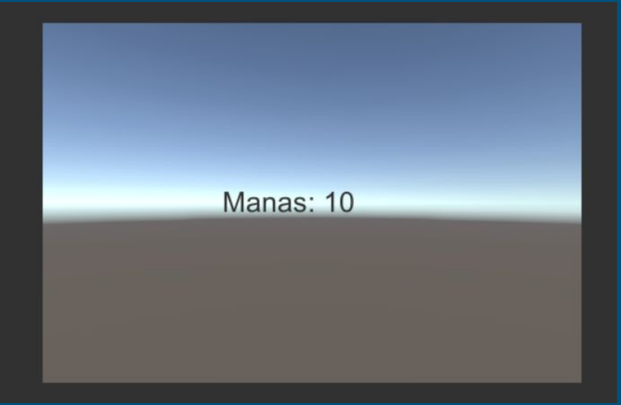
2 Manas_Int_To_String
Logic Type: Compute Or Operation
Variable Type: Int
Compute Type: To String
10
Do It
10

3 Manas_UI_String
Logic Type: Compute Or Operation
Variable Type: String
Compute Type: Add
Manas:
10
Do It
Manas: 10

Open Message:
4 Show_Manas
Logic Type: Compute Or Operation
Variable Type: Component UI Unity Text
The Attached To
No Editor run: 'Attached To'/linked null
Compute Type: Set Text
Manas: 10
Always Doing It
none GameObject

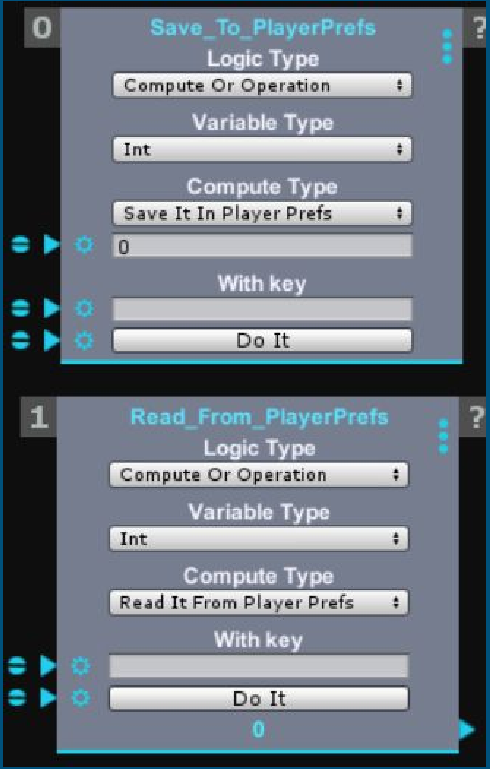
Unity Ads. Reward the player 3/3

If you repeat pressing T and clicking close button the Manas will be increased by 10 each time.



Congrats, you have configured rewarded video ads and test it in your Unity editor.

P.S. If you stop playing and replay your Unity Editor you will notice that your manas are re-init to zero. To save your manas in disque use the PlayerPrefs functions available in the int logic node.

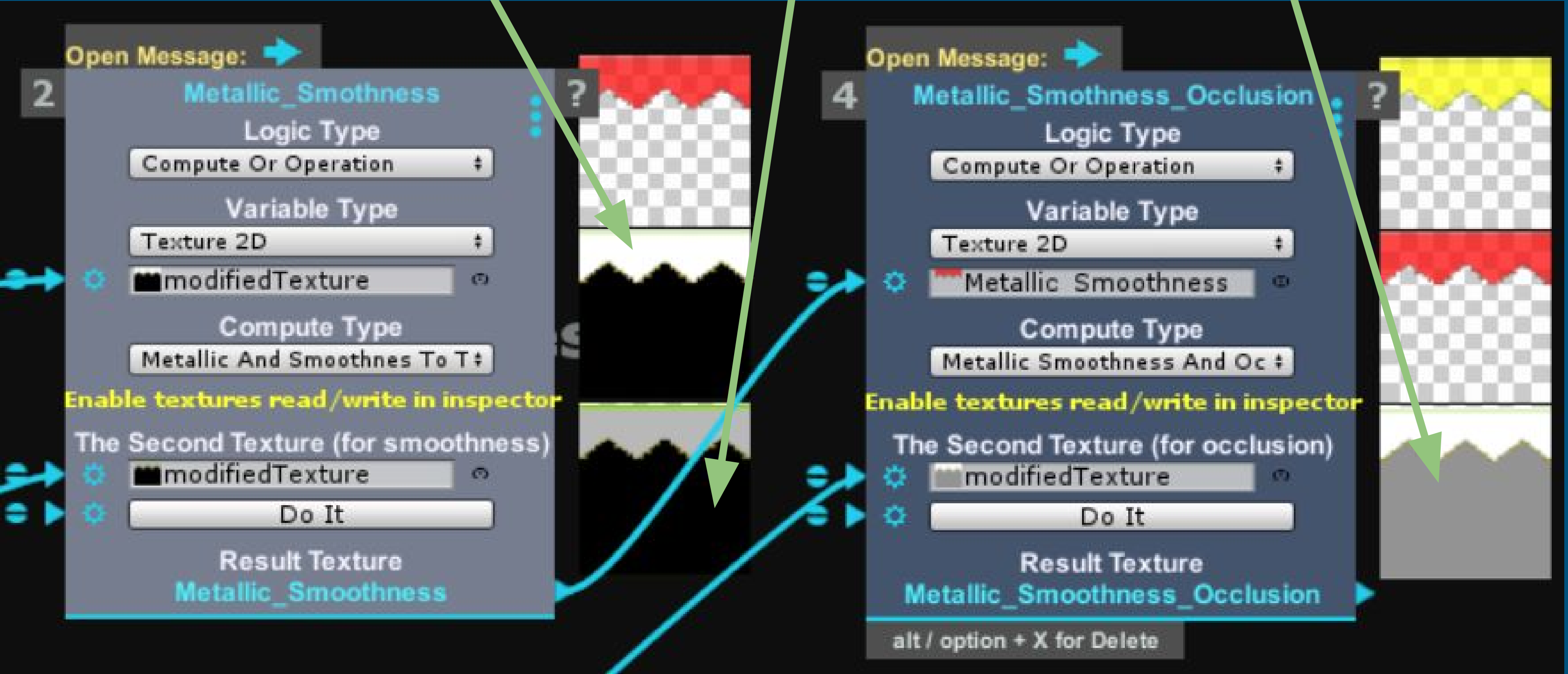


0 Save_To_PlayerPrefs
Logic Type: Compute Or Operation
Variable Type: Int
Compute Type: Save It In Player Prefs
0
With key
Do It

1 Read_From_PlayerPrefs
Logic Type: Compute Or Operation
Variable Type: Int
Compute Type: Read It From Player Prefs
With key
Do It
0

Prepare PBR Textures

Combine Metallic Smoothness and Occlusion maps in one map



Open Message:

2 Metallic_Smothness

Logic Type: Compute Or Operation

Variable Type: Texture 2D

modifiedTexture

Compute Type: Metallic And Smoothnes To T

Enable textures read/write in inspector

The Second Texture (for smoothness): modifiedTexture

Do It

Result Texture: Metallic_Smoothness

Open Message:

4 Metallic_Smothness_Occlusion

Logic Type: Compute Or Operation

Variable Type: Texture 2D

Metallic_Smoothness

Compute Type: Metallic Smoothness And Oc

Enable textures read/write in inspector

The Second Texture (for occlusion): modifiedTexture

Do It

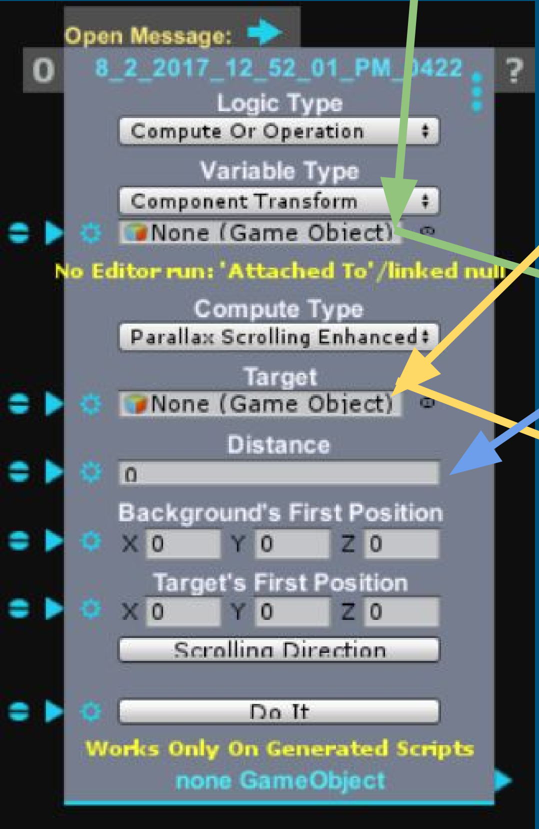
Result Texture: Metallic_Smoothness_Occlusion

alt / option + X for Delete

Enhanced Parallax Scrolling

Diamond 1.1

Move background according to player moves. Distance = 0 -> Fixed background



Open Message:

0 8_2_2017_12_52_01_PM_0422 ?

Logic Type: Compute Or Operation

Variable Type: Component Transform

None (Game Object)

No Editor run: 'Attached To'/linked null

Compute Type: Parallax Scrolling Enhanced

Target: None (Game Object)

Distance: 0

Background's First Position: X 0 Y 0 Z 0

Target's First Position: X 0 Y 0 Z 0

Scrolling Direction

Do It

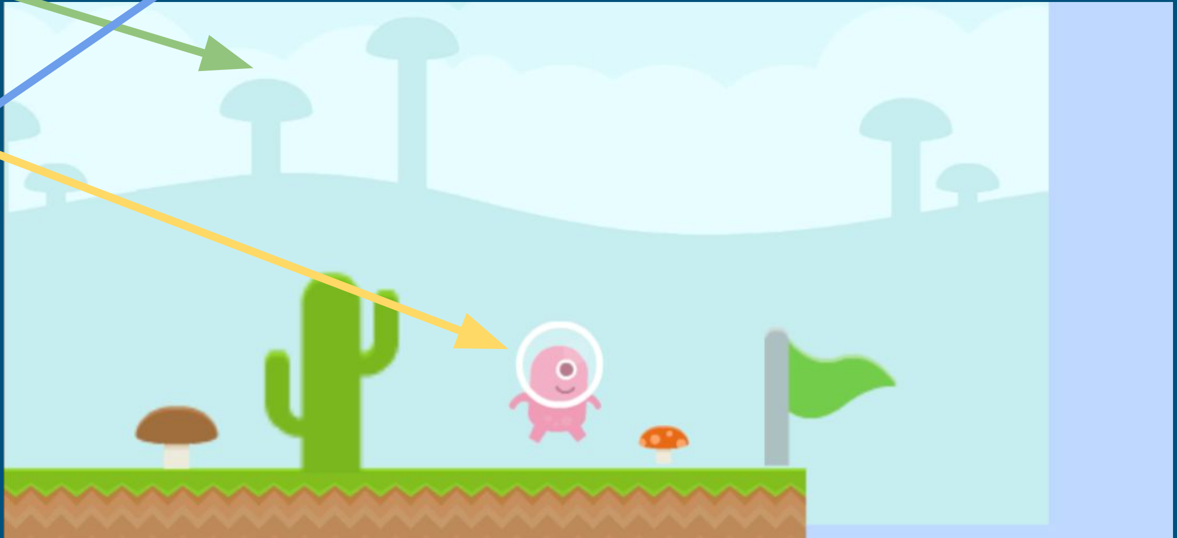
Works Only On Generated Scripts

none GameObject

Distance = 0 -> Fixed background

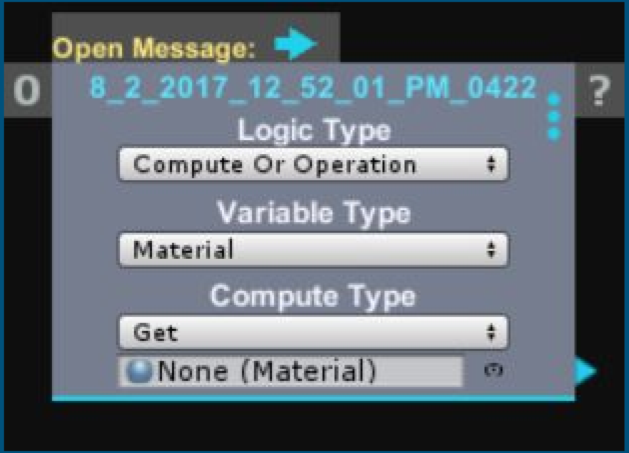
Distance = [0, 100] -> Smoothed movement

Distance = 100 -> Same movement as player



Procedural Materials

In the Compute Type, access functions, variables, and your procedural properties of your procedural materials (for example your substances).



Open Message:

0

8_2_2017_12_52_01_PM_0422

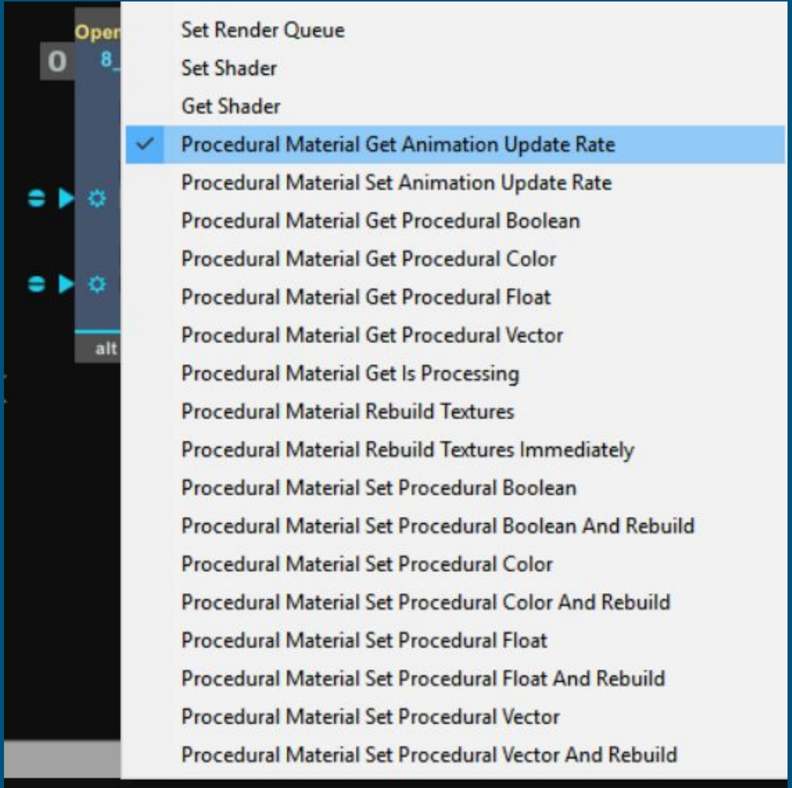
?

Logic Type: Compute Or Operation

Variable Type: Material

Compute Type: Get

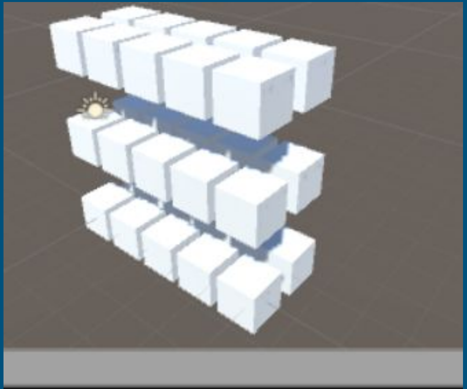
None (Material)

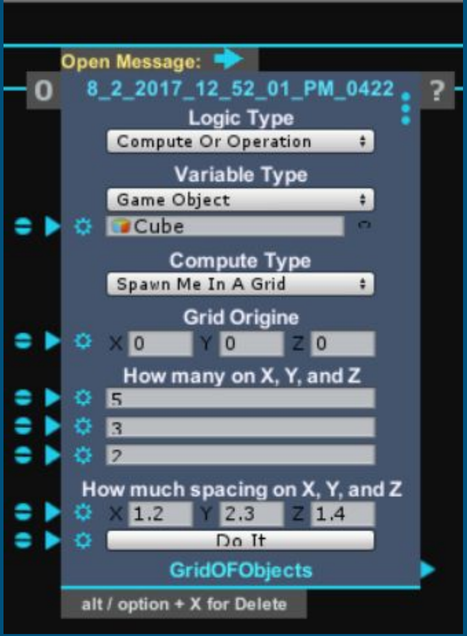


- Set Render Queue
- Set Shader
- Get Shader
- ✓ Procedural Material Get Animation Update Rate
- Procedural Material Set Animation Update Rate
- Procedural Material Get Procedural Boolean
- Procedural Material Get Procedural Color
- Procedural Material Get Procedural Float
- Procedural Material Get Procedural Vector
- Procedural Material Get Is Processing
- Procedural Material Rebuild Textures
- Procedural Material Rebuild Textures Immediately
- Procedural Material Set Procedural Boolean
- Procedural Material Set Procedural Boolean And Rebuild
- Procedural Material Set Procedural Color
- Procedural Material Set Procedural Color And Rebuild
- Procedural Material Set Procedural Float
- Procedural Material Set Procedural Float And Rebuild
- Procedural Material Set Procedural Vector
- Procedural Material Set Procedural Vector And Rebuild

3D Grid Generator

Spawn any gameobject (prefab or scene object) in a customizable 3D grid.
Works in Runtime and in Editor.





Open Message:

0 8_2_2017_12_52_01_PM_0422 ?

Logic Type: Compute Or Operation

Variable Type: Game Object

Cube

Compute Type: Spawn Me In A Grid

Grid Origine: X 0 Y 0 Z 0

How many on X, Y, and Z: 5, 3, 2

How much spacing on X, Y, and Z: X 1.2 Y 2.3 Z 1.4

Do It

GridOFObjects

alt / option + X for Delete

Conclusion

Thank you for downloading diamond 1.1

For any further questions and suggestions. Here is the mezanix website, mezanix email, and the diamond Unity Forum

www.mezanix.com

Diamond Unity Forum

mezanixissa@gmail.com

Diamond 1.1.2

Visual Scripting for Unity
Documentation
New Features of the 1.1.2 Version

www.mezanix.com

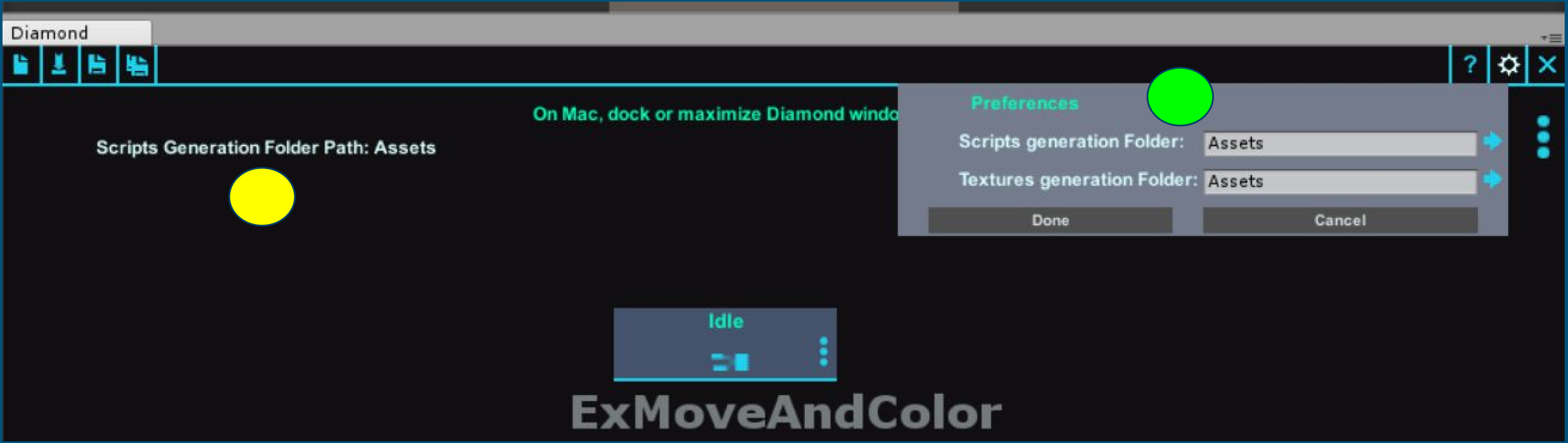
Before Diving in Diamond

Diamond use the State Machine concept. You can have an idea about it here

https://en.wikipedia.org/wiki/Finite-state_machine

Per Graph Scripts Generation Path

Each graph has its own scripts generation path indicated permanently in Diamonds editor and editable in the preferences.


Diamond

On Mac, dock or maximize Diamond windo

Scripts Generation Folder Path: Assets

Preferences

Scripts generation Folder: Assets

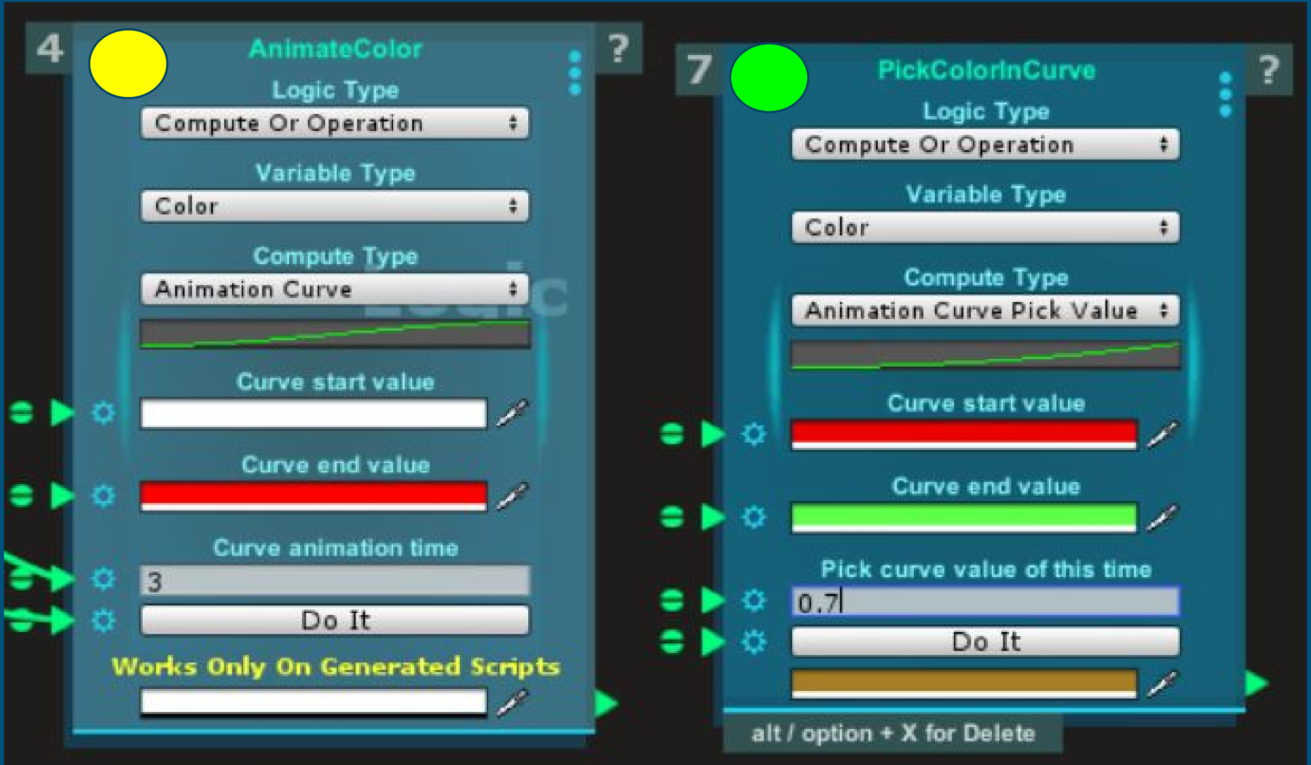
Textures generation Folder: Assets

Diamond 1.1.2

Animation Curves

Diamond 1.1.2

Animate floats, vectors and colors. In this image you can animate or pick a color on the curve according to 0.7 value on the curves normalized horizontal axis.



4 AnimateColor

Logic Type: Compute Or Operation

Variable Type: Color

Compute Type: Animation Curve

Curve start value

Curve end value

Curve animation time: 3

Do It

Works Only On Generated Scripts

7 PickColorInCurve

Logic Type: Compute Or Operation

Variable Type: Color

Compute Type: Animation Curve Pick Value

Curve start value

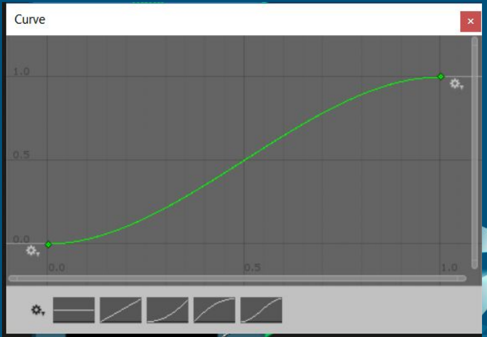
Curve end value

Pick curve value of this time: 0.7

Do It

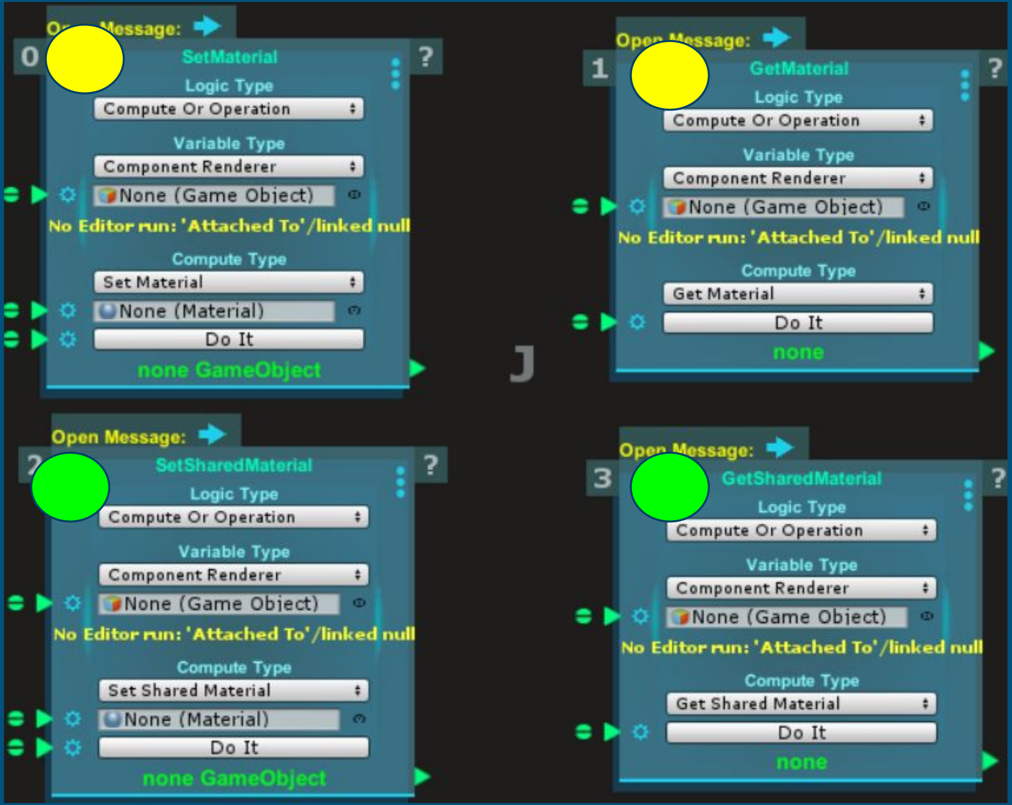
alt / option + X for Delete

Click on the curve to edit it.
Double click on the curve line to add an editable Bezier point on the curve.
Right click Bezier point for more options.



Set and get material of game object

Access material and shared material.



0 SetMaterial
Logic Type: Compute Or Operation
Variable Type: Component Renderer
None (Game Object)
No Editor run: 'Attached To'/linked null
Compute Type: Set Material
None (Material)
Do It
none GameObject

1 GetMaterial
Logic Type: Compute Or Operation
Variable Type: Component Renderer
None (Game Object)
No Editor run: 'Attached To'/linked null
Compute Type: Get Material
Do It
none

2 SetSharedMaterial
Logic Type: Compute Or Operation
Variable Type: Component Renderer
None (Game Object)
No Editor run: 'Attached To'/linked null
Compute Type: Set Shared Material
None (Material)
Do It
none GameObject

3 GetSharedMaterial
Logic Type: Compute Or Operation
Variable Type: Component Renderer
None (Game Object)
No Editor run: 'Attached To'/linked null
Compute Type: Get Shared Material
Do It
none

Diamond 1.1.2

Conclusion

Thank you for downloading diamond 1.1.2

For any further questions and suggestions. Here is the mezanix website, mezanix email, and the diamond Unity Forum

www.mezanix.com

Diamond Unity Forum

mezanixissa@gmail.com