3D ENDLESS RUNNER KIT

USER MANUAL

Table of Contents

Introduction

Getting Started

1. Import the unity package into an empty project.
2. Go to Project > Scenes > Scene_1.
3. Click "Don't Save" if prompted to save the current scene. (The project only uses a single scene file)
4. Press the play button to get an idea of the menu flow and gameplay.

Frequent References

The documentation will frequently refer to some elements that are the core of the project:

Player

The *Player* is the prefab that contains the primary character of the game. This prefab also contains various scripts. The word "Player" will not be used to refer to the user in this document. The *Player* behavior is controlled by *ControllerScript.js*.

Patch

Patches are the environments/ surroundings of the game. They are generated by *PatchesRandomizer.js* during run time. Each patch has the default displacement of 3000 meters.

At any particular time, two patches are always active, the one the *Player*, is on and the immediate next one. Every time the *Player* passes a patch completely it is destroyed and the next one is generated.

Spline

The spline is a line passing over each patch, which defines the path of the *Player* and the location where obstacles will be generated.

CPs

The spline is manipulated using CPs, which is short for Check Points. If the patch is completely straight, only two CPs are needed to draw a straight spline. However, if the path has bends and curves, multiple CPs are needed to be arranged to draw a spline that has exactly the same shape and centered on the path of the patch. Take a look at the sample patches to acquaint yourself with how CPs are related to the spline.

Custom Menu

The default GUI that consists of a set of menus and the HUD is referred in the documentation and script comments by the term "custom menu". This is because the default GUI implementation does not use the GUI class or most of the components provided by Unity.

Audience

The user guide is composed assuming that the reader/ developer has an understanding of game development in Unity and Javascript programming for Unity.

Project Architecture

The project doesn't use many game objects, physics, GUI components etc. provided by Unity to maximize performance on mobile platforms.

Physics

For instance the character/ player physics has been defined as needed in the *ControllerScript.js.*

Menus

The menus provided in the Endless Kit do not use the GUI library of Unity. Their implementation can be found in *MenuScript.js*.

Collisions

There are two kinds of collisions detected during gameplay. The first and the most common one is the front collisions with obstacles detected and handled by the *PlayerFrontCollider.*

The second type of collisions is called "stumble" which are detected by *PlayerSidesCollider* if the *Player* bumps into an obstacle sideways. This triggers the *Enemy* (police car by default) to chase the *Player* for a unit time (5 seconds by default) and then move out of the camera. If the player stumbles (collides sides ways) into another obstacle while the *Enemy* is chasing it, game over routine is called and the death scene is played.

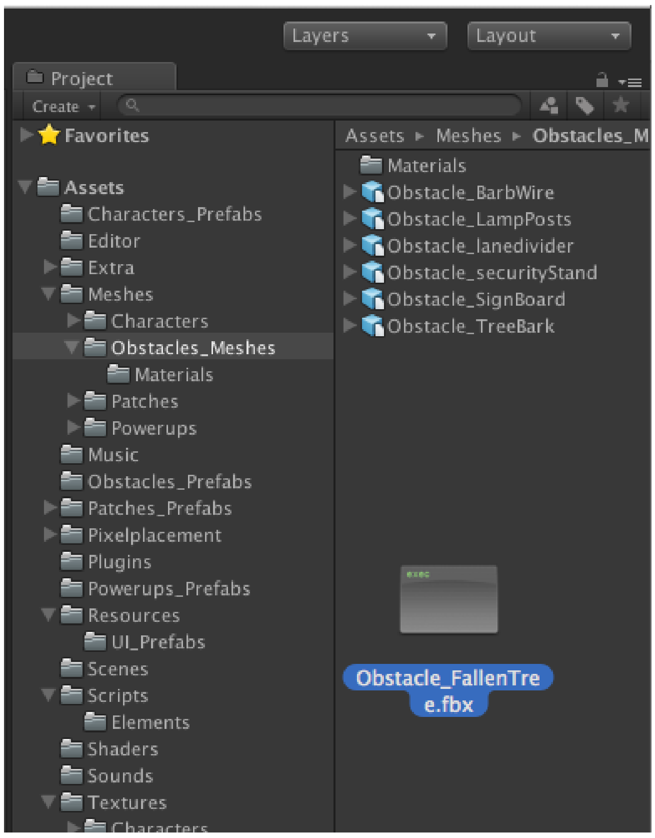
Sounds

All the sounds used in the scene are 2D sounds. All the audio sources have been added to the *SoundManager* prefab located in the scene. A script also named as *SoundManager* has been added to the prefab and is responsible for playing or stopping a sound as needed. Exposed variables are used to hold the references of the audio sources.

Personalizing the Template

Adding an Obstacle

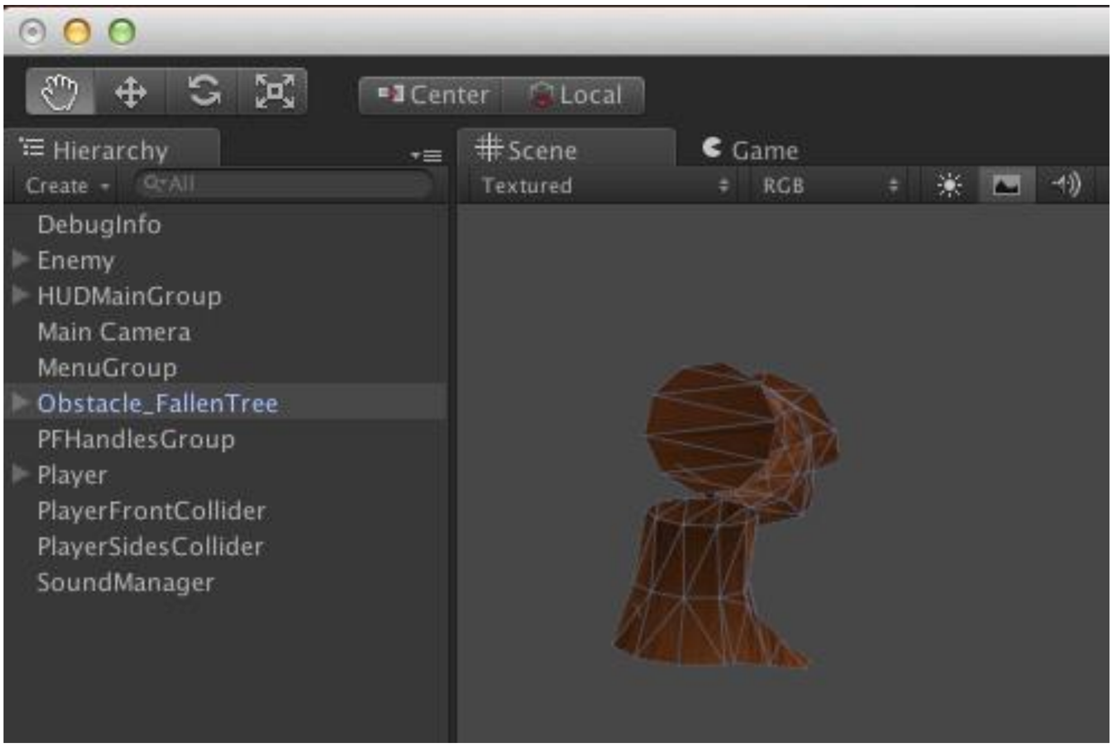
1. Import the .fbx of the asset that you will use as an obstacle.



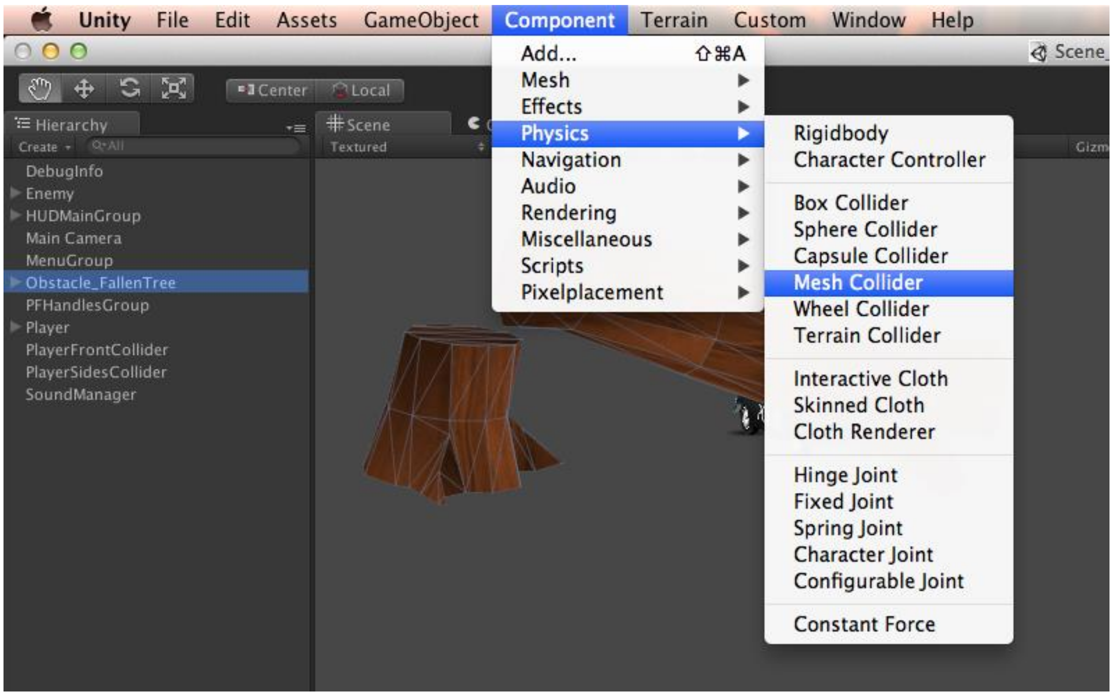
Optimization Notes:

- Use Model > Scale Factor to adjust the scale.
- Use Model > Mesh Compression to compress the imported asset as much as you can without compromising its appearance.
- Select "None" in Rig > Animation Type if your asset isn't rigged.
- Uncheck the Import Animation option in Animations > Import Animation if your asset isn't animated.

2. Drag the imported asset in "Hierarchy" tab to create a prefab.



3. Select Component > Physics > Mesh Collider to add a mesh collider to your obstacle asset.

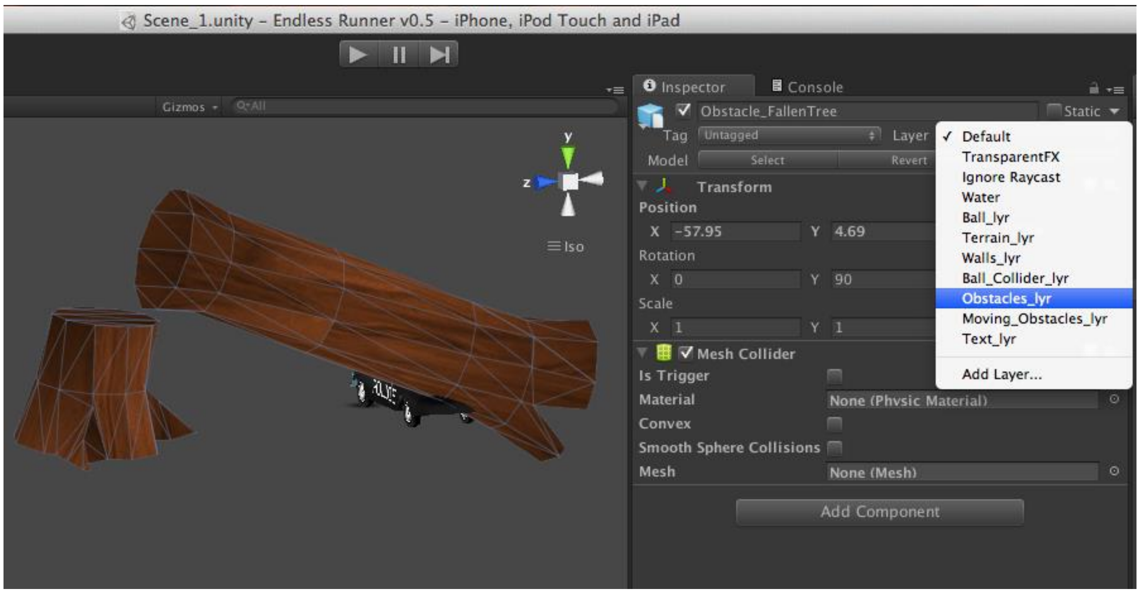


Optimization Notes:

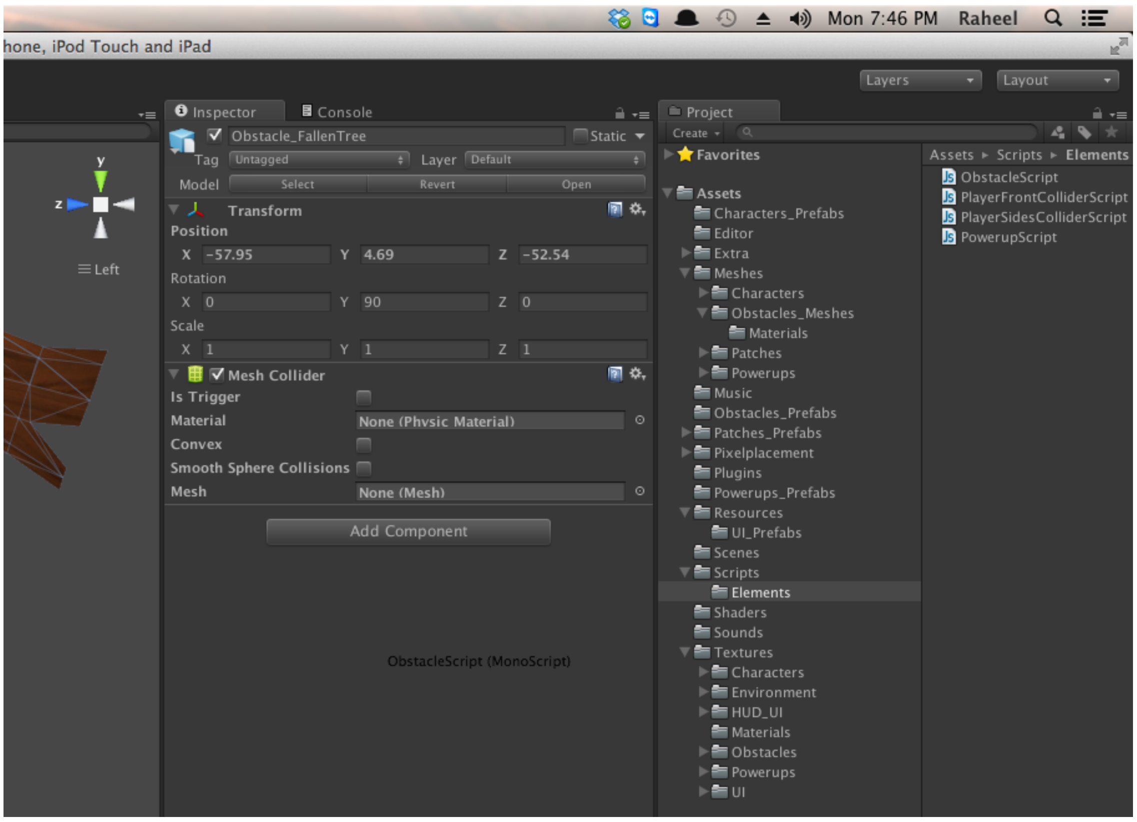
- Use primitive colliders such as Sphere, Box etc. for better performance during gameplay.
- Do not scale the collider component on your prefab for optimizing object pooling.

4. Select Layer > Obstacles_lyr in the *Inspector* tab.

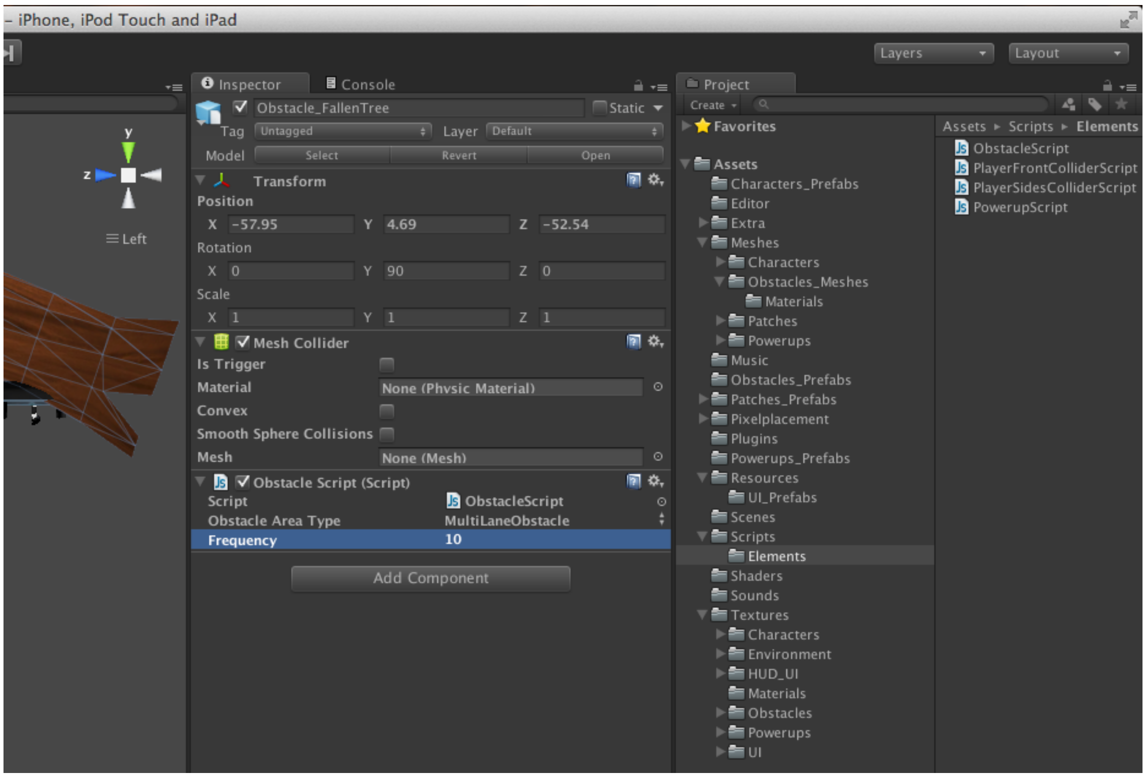
Note: Make sure the object with the collider has the layer "Obstacle_lyr". You can also change the layer of all the objects of an obstacle prefab to "Obstacle_lyr" to keep things simple.



5. Go to Project > Assets > Scripts > Elements.
6. Add the "ObstacleScript.js" on your obstacle's prefab.



7. Set the *Obstacle Area Type* on the *Obstacle Script* Component.
8. Set the Frequency. (Default Value is 10)



Obstacle_FallenTree
Static
Tag Untagged Layer Default
Model Select Revert Open

Transform
Position X -57.95 Y 4.69 Z -52.54
Rotation X 0 Y 90 Z 0
Scale X 1 Y 1 Z 1

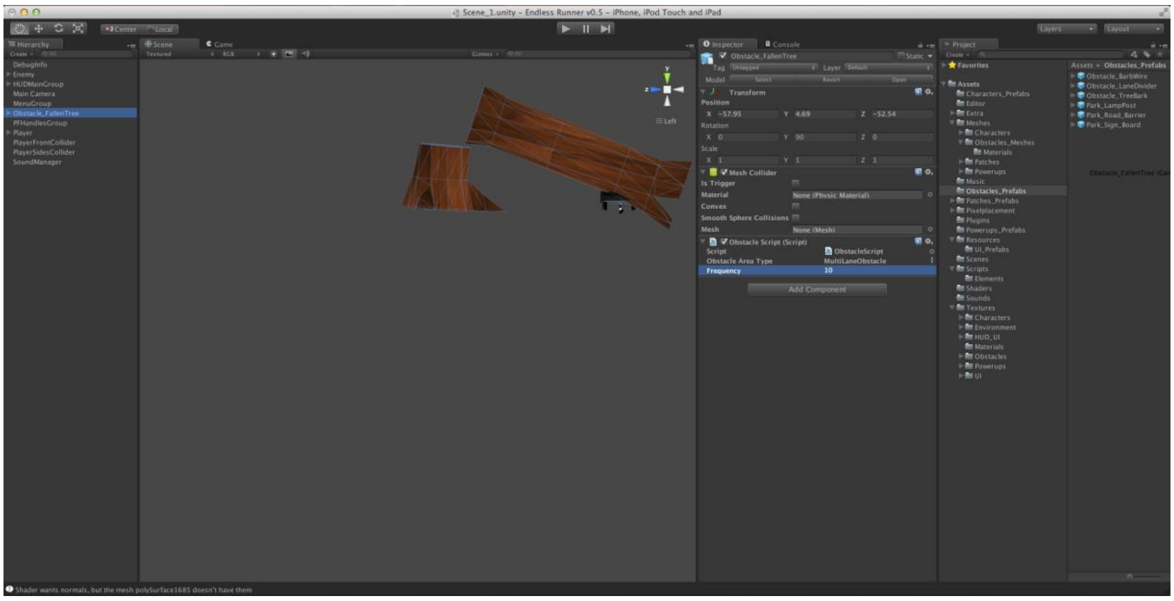
Mesh Collider
Is Trigger
Material None (Physic Material)
Convex
Smooth Sphere Collisions
Mesh None (Mesh)

Obstacle Script (Script)
Script ObstacleScript
Obstacle Area Type MultiLaneObstacle
Frequency 10

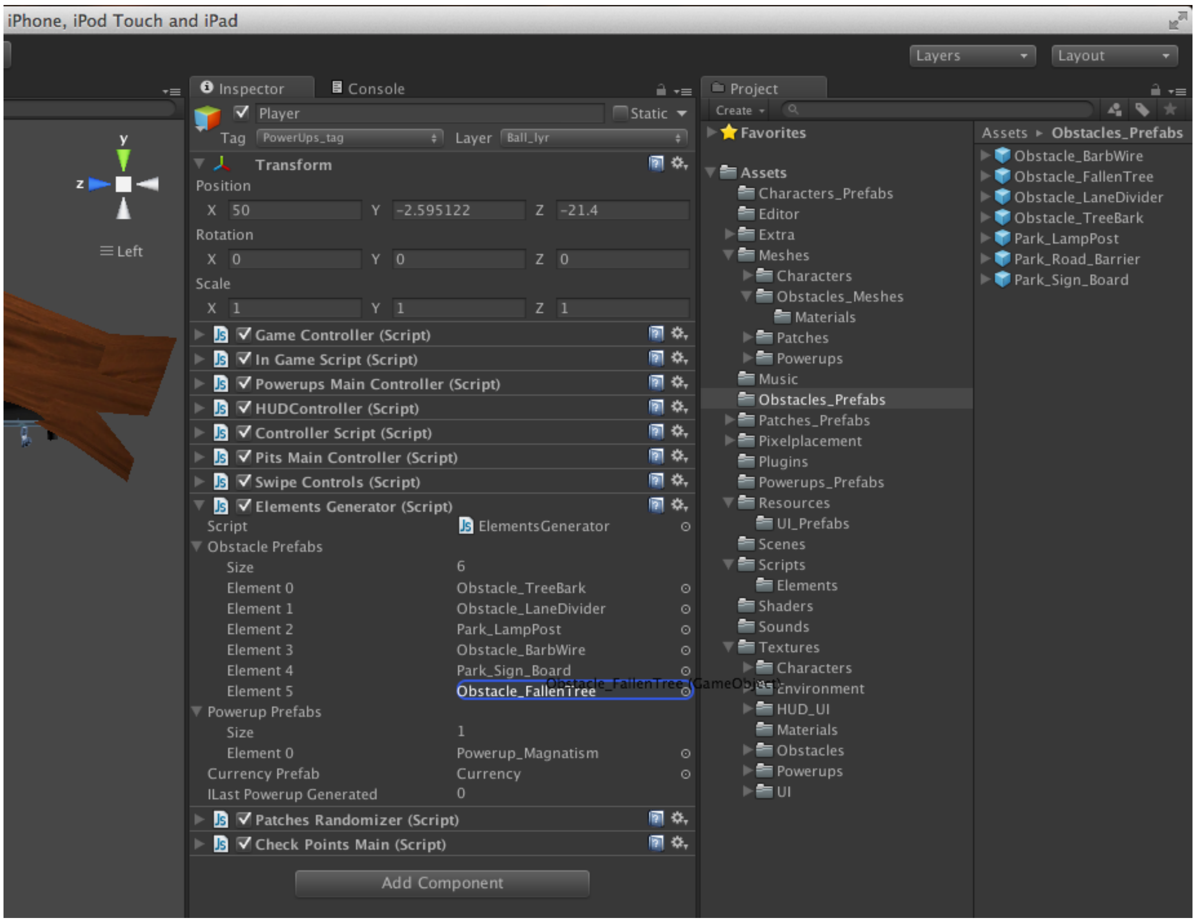
Assets
Characters_Prefabs
Editor
Extra
Meshes
Characters
Obstacles_Meshes
Materials
Patches
Powerups
Music
Obstacles_Prefabs
Patches_Prefabs
Pixelplacement
Plugins
Powerups_Prefabs
Resources
UI_Prefabs
Scenes
Scripts
Elements
Shaders
Sounds
Textures
Characters
Environment
HUD_UI
Materials
Obstacles
Powerups
UI

ObstacleScript
PlayerFrontColliderScript
PlayerSidesColliderScript
PowerupScript

9. Drag the newly created prefab to *Obtacles_Prefabs* or any other folder in the *Project* tab to create a prefab.


Scene_1.unity - Endless Runner v0.5 - iPhone, iPod Touch and iPad
Hierarchy
Scene
Game
Inspector
Console
Project
Add Component

10. To make the created obstacle appear on the path during gameplay, you need to add it to Hierarchy > Player > Elements Generator > Obstacle Prefabs.
11. Remove the newly created obstacle from the Hierarchy tab before launching the game.



Player — Static
Tag PowerUps_tag — Layer Ball_lyr

Transform
Position X 50 Y -2.595122 Z -21.4
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1

Game Controller (Script)
In Game Script (Script)
Powerups Main Controller (Script)
HUDController (Script)
Controller Script (Script)
Pits Main Controller (Script)
Swipe Controls (Script)
Elements Generator (Script)

Script	ElementsGenerator
Obstacle Prefabs	
Size	6
Element 0	Obstacle_TreeBark
Element 1	Obstacle_LaneDivider
Element 2	Park_LampPost
Element 3	Obstacle_BarbWire
Element 4	Park_Sign_Board
Element 5	Obstacle_FallenTree
Powerup Prefabs	
Size	1
Element 0	Powerup_Magnatism
Currency Prefab	Currency
ILast Powerup Generated	0

Patches Randomizer (Script)
Check Points Main (Script)

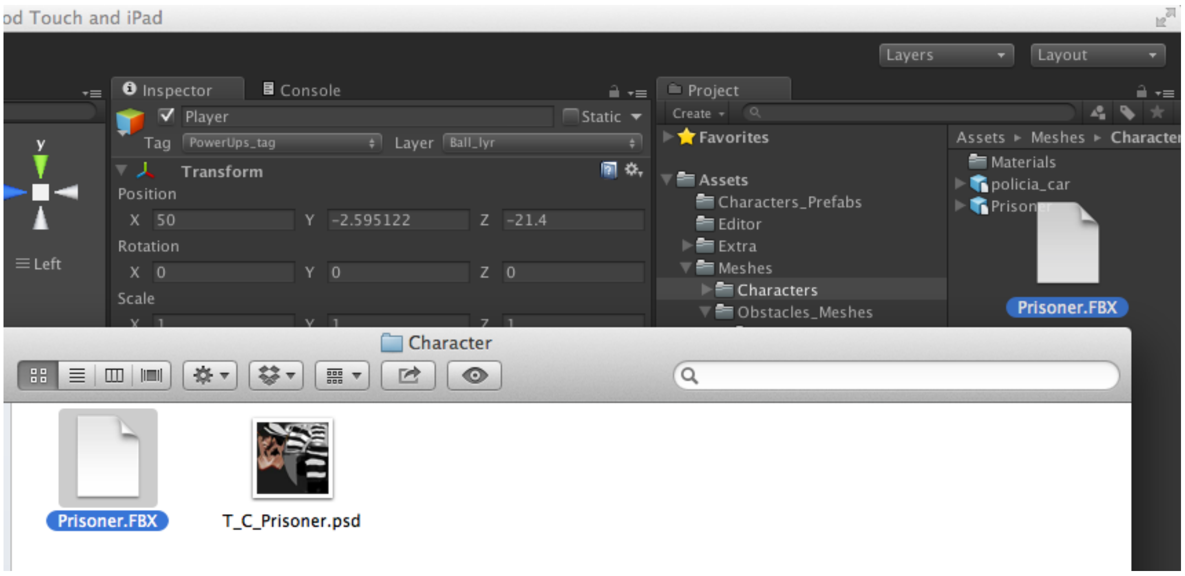
Obstacle_FallenTree (GameObject)

Assets
- Characters_Prefabs
- Editor
- Extra
- Meshes
  - Characters
  - Obstacles_Meshes
    - Materials
  - Patches
  - Powerups
- Music
- Obstacles_Prefabs
- Patches_Prefabs
- Pixelplacement
- Plugins
- Powerups_Prefabs
- Resources
  - UI_Prefabs
- Scenes
- Scripts
  - Elements
- Shaders
- Sounds
- Textures
  - Characters
  - Environment
  - HUD_UI
  - Materials
  - Obstacles
  - Powerups
  - UI

Assets ▸ Obstacles_Prefabs
- Obstacle_BarbWire
- Obstacle_FallenTree
- Obstacle_LaneDivider
- Obstacle_TreeBark
- Park_LampPost
- Park_Road_Barrier
- Park_Sign_Board

Replacing Player Character

1. Import the player .fbx that will be used as the main character.



Player — Static
Tag PowerUps_tag — Layer Ball_lyr

Transform
Position X 50 Y -2.595122 Z -21.4
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1

Assets
- Characters_Prefabs
- Editor
- Extra
- Meshes
  - Characters
  - Obstacles_Meshes

Assets ▸ Meshes ▸ Characte
- Materials
- policia_car
- Prisoner

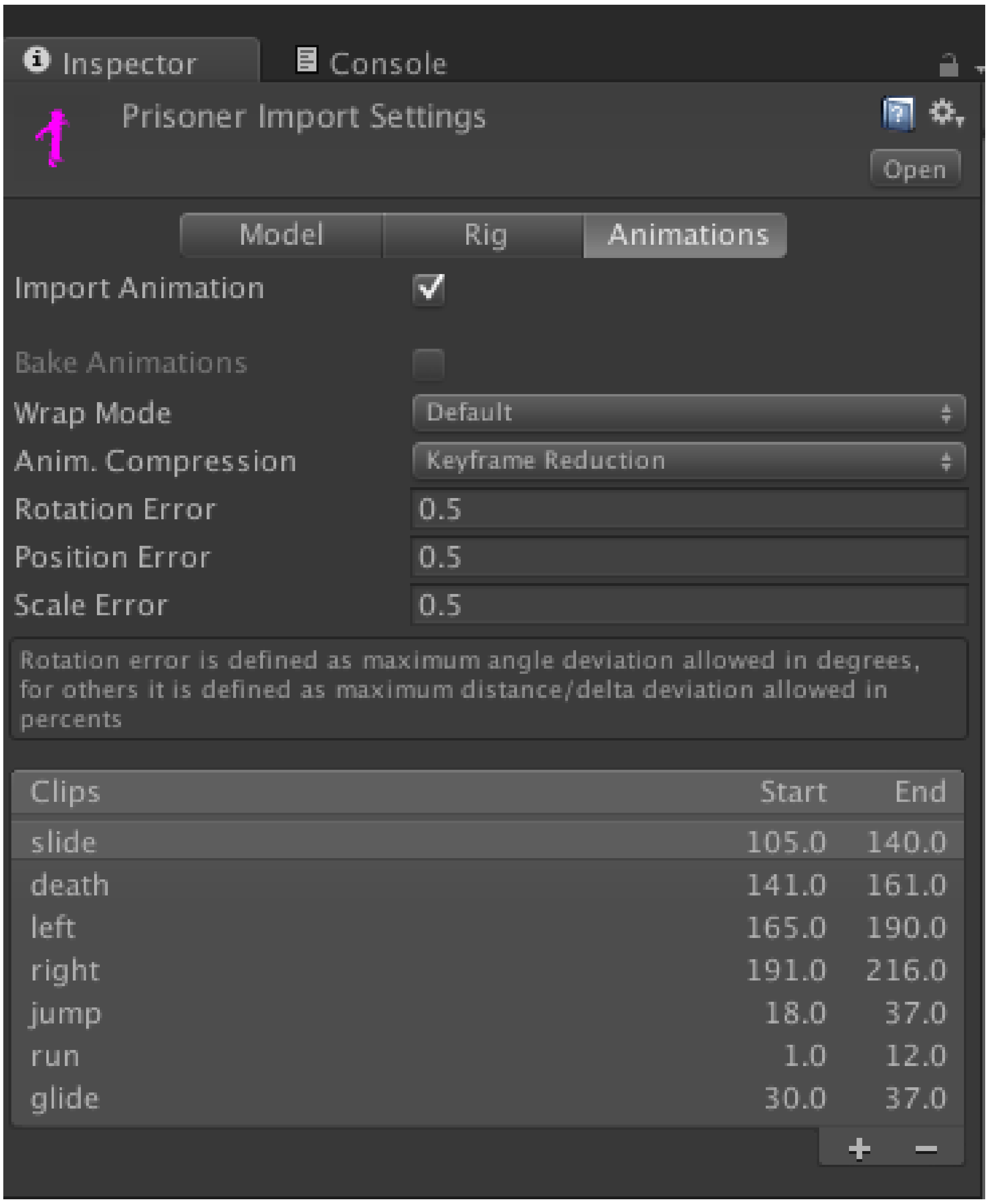
Prisoner.FBX

Prisoner.FBX   T_C_Prisoner.psd

Optimization Notes:

- Use <u>Model > Scale Factor</u> to adjust the scale.
- Use <u>Model > Mesh Compression</u> to compress the imported asset as much as you can without compromising its appearance.

2. Define the animation frames.



Prisoner Import Settings

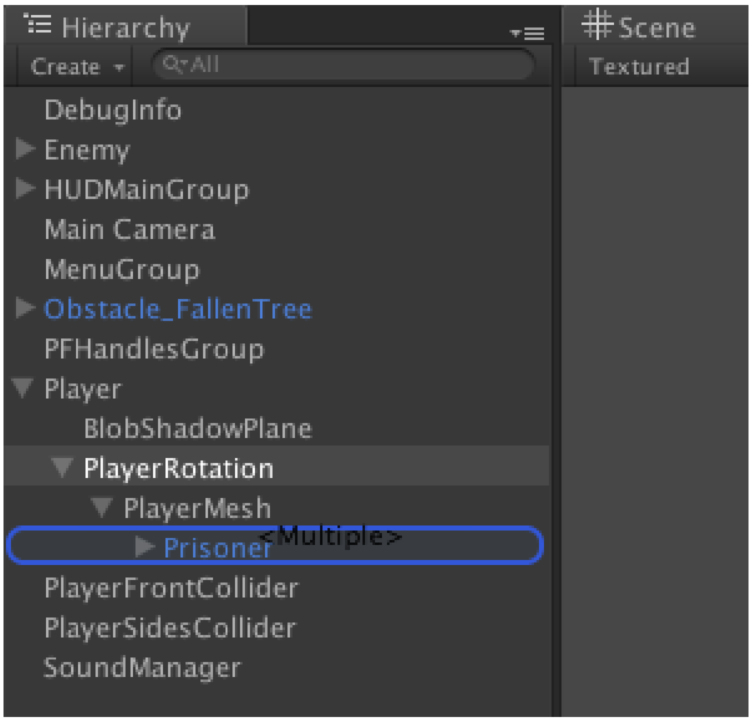
Import Animation	✓
Bake Animations	
Wrap Mode	Default
Anim. Compression	Keyframe Reduction
Rotation Error	0.5
Position Error	0.5
Scale Error	0.5

Rotation error is defined as maximum angle deviation allowed in degrees, for others it is defined as maximum distance/delta deviation allowed in percents

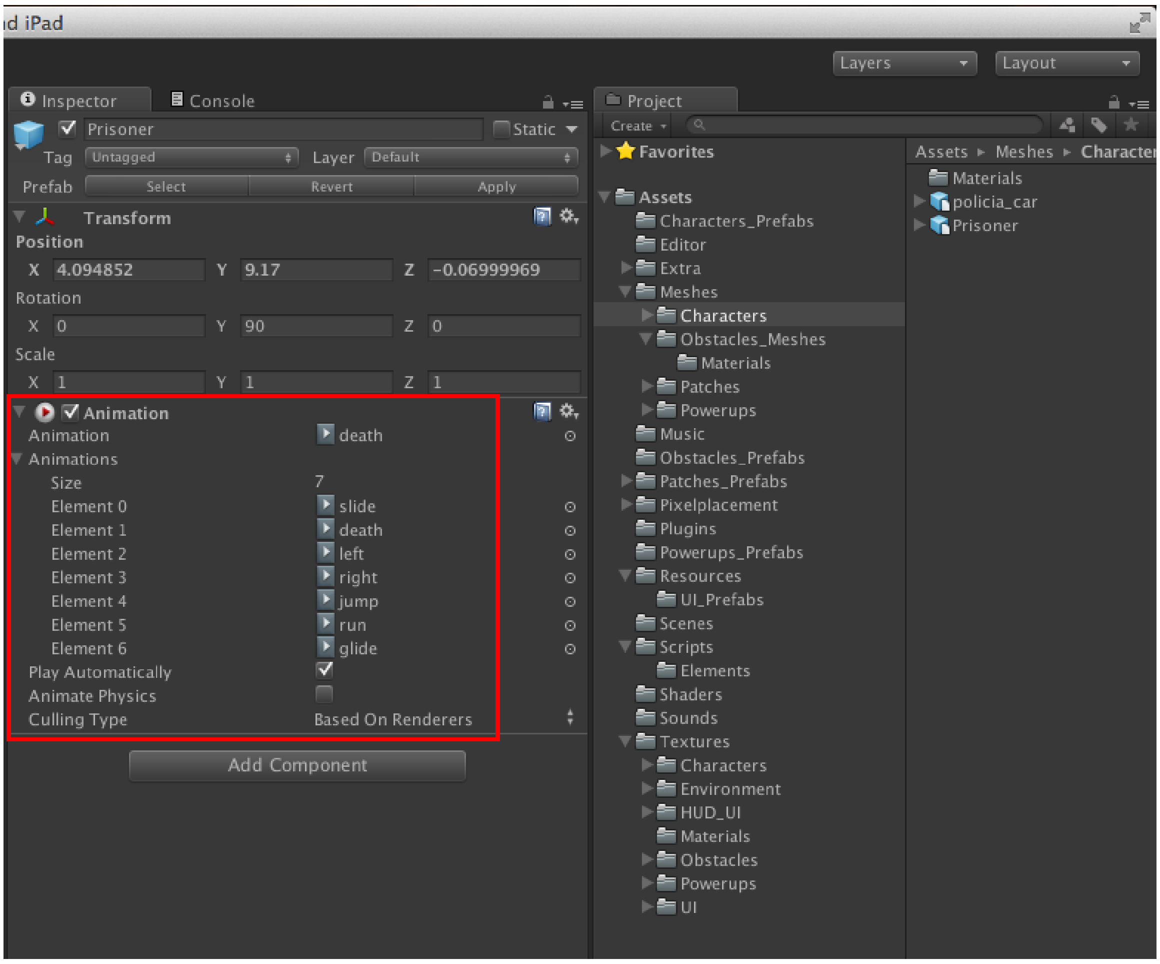
Clips	Start	End
slide	105.0	140.0
death	141.0	161.0
left	165.0	190.0
right	191.0	216.0
jump	18.0	37.0
run	1.0	12.0
glide	30.0	37.0

Note: The *ControllerScript.js* uses all the animations listed in the above screenshot. If a certain animation is not defined, an exception will be thrown when the script attempts to play that animation.

3. Drag the imported asset in Hierarchy > Player > PlayerRotation > PlayerMesh.



4. Make sure the *Animation* component of the character prefab has all the animations that will be used by the *ControllerScript.js.*

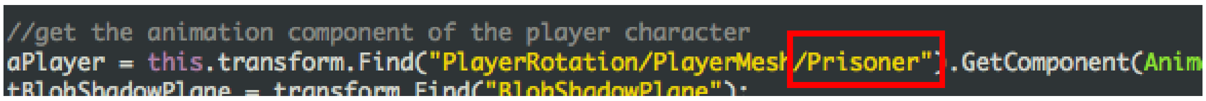


Prisoner Static
Tag Untagged Layer Default
Prefab Select Revert Apply

Transform
Position
X 4.094852 Y 9.17 Z -0.06999969
Rotation
X 0 Y 90 Z 0
Scale
X 1 Y 1 Z 1

Animation
Animation	death
Animations	
Size	7
Element 0	slide
Element 1	death
Element 2	left
Element 3	right
Element 4	jump
Element 5	run
Element 6	glide
Play Automatically	✓
Animate Physics	
Culling Type	Based On Renderers

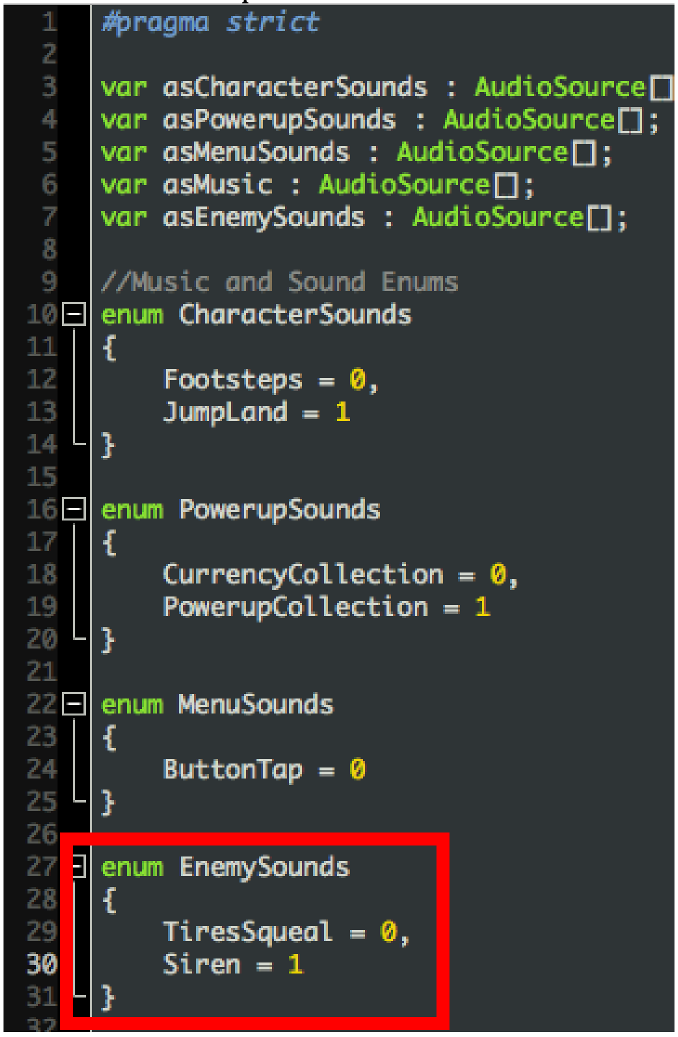
5. Change the name of the character to your player character in the *Start()* function of the *ControllerScript.js* to get the animations.



```
//get the animation component of the player character
aPlayer = this.transform.Find("PlayerRotation/PlayerMesh/Prisoner").GetComponent(Anim
tBlobShadowPlane = transform.Find("BlobShadowPlane");
```

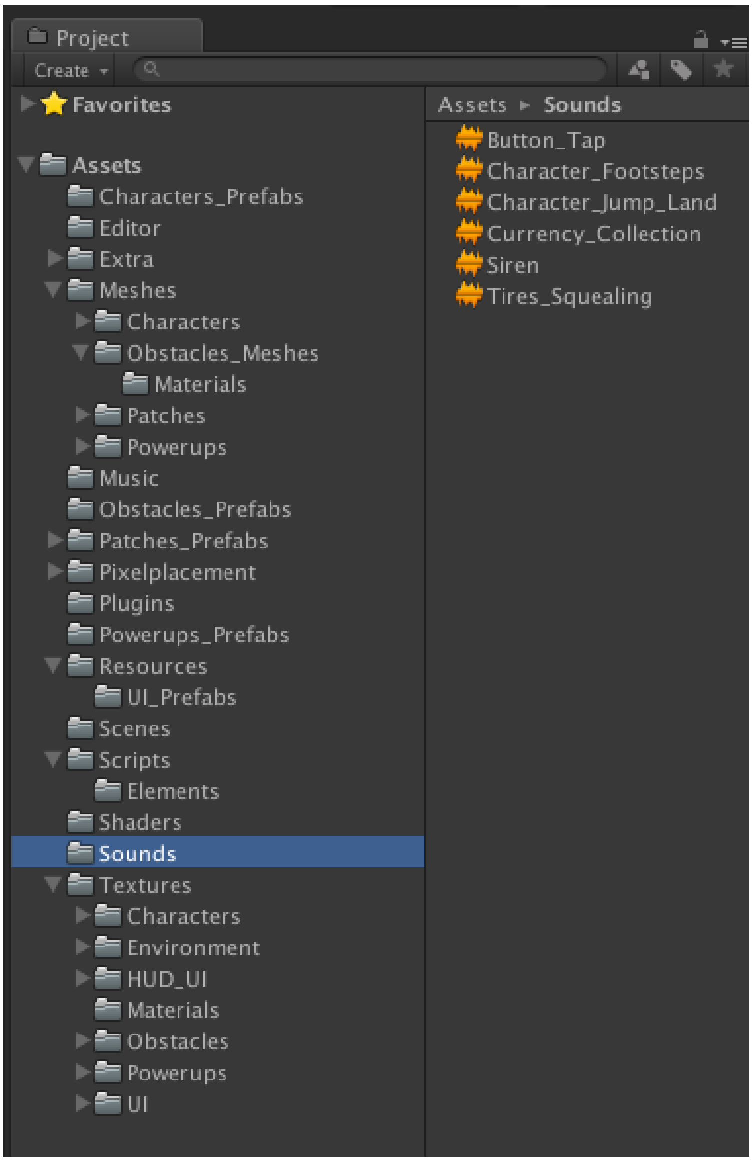
Adding a Sound

1. Go to Project > Scripts > SoundManager.
2. Add the sound's name in the relevant *enum.* Lets take the Siren sound played by the police car as an example.



```
#pragma strict

var asCharacterSounds : AudioSource[]
var asPowerupSounds : AudioSource[];
var asMenuSounds : AudioSource[];
var asMusic : AudioSource[];
var asEnemySounds : AudioSource[];

//Music and Sound Enums
enum CharacterSounds
{
	Footsteps = 0,
	JumpLand = 1
}

enum PowerupSounds
{
	CurrencyCollection = 0,
	PowerupCollection = 1
}

enum MenuSounds
{
	ButtonTap = 0
}

enum EnemySounds
{
	TiresSqueal = 0,
	Siren = 1
}
```

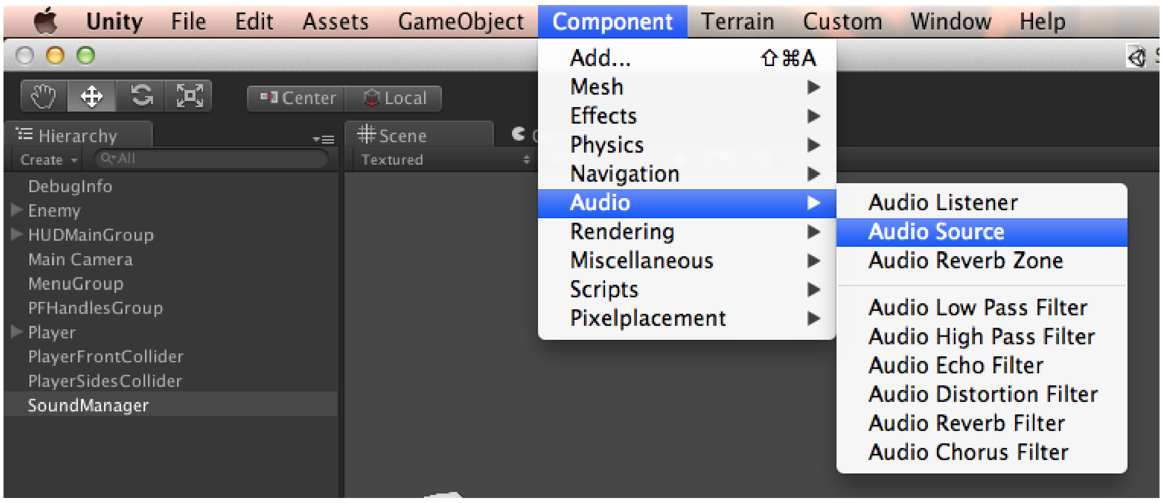
3. Import the sound. (Drag and drop the sound from the Finder/ Window to Project > Assets > Sounds)



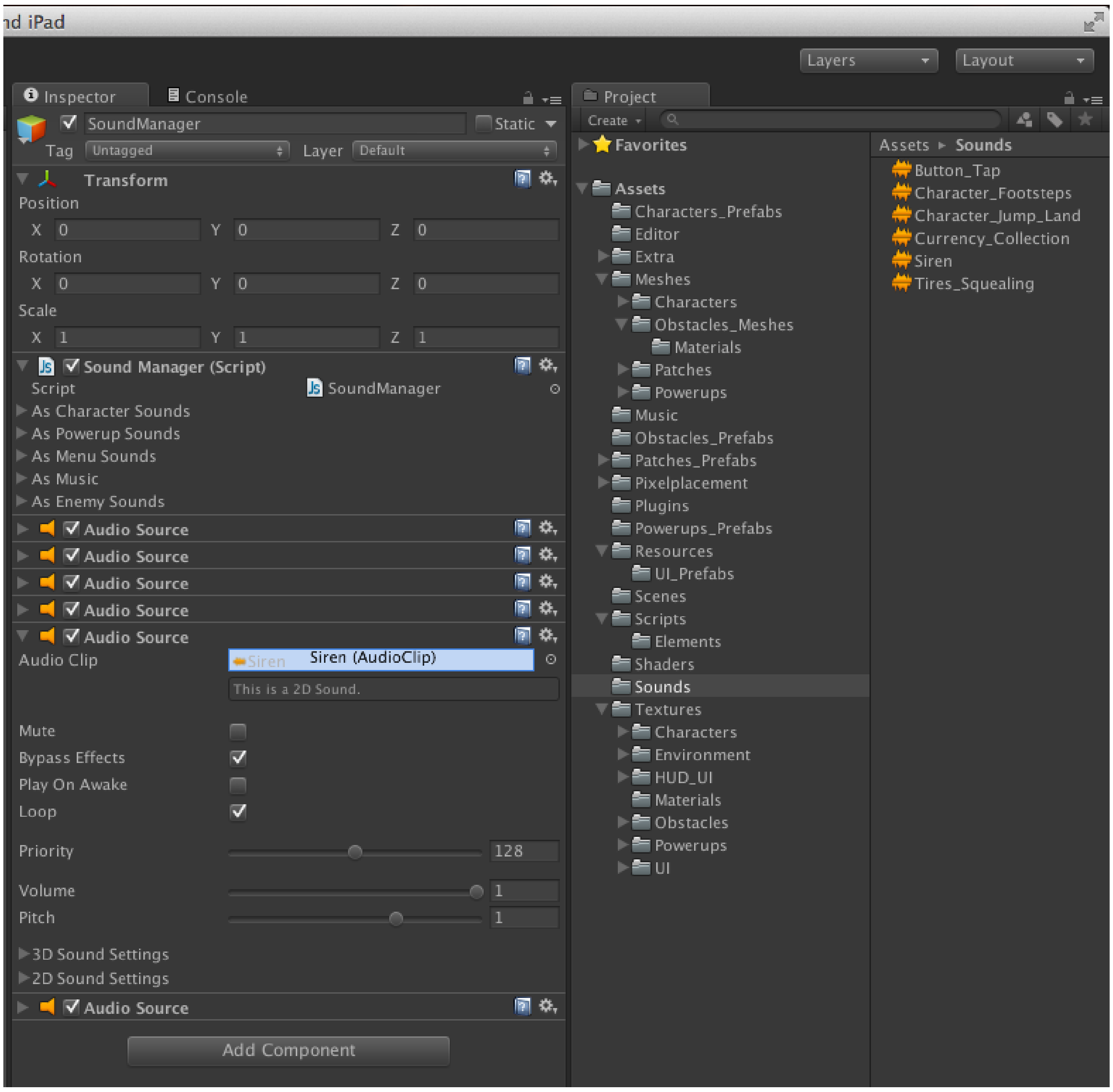
Optimization Notes:

- Change the *Audio Format* and *Load type* as needed.
- Uncheck *3D Sound* as the project by default doesn't uses 3D sounds.

4. Select Hierarchy > SoundManager.
5. Select Component > Audio > Audio Source. (This will add another Audio Source to your *SoundManager* prefab to be used later)



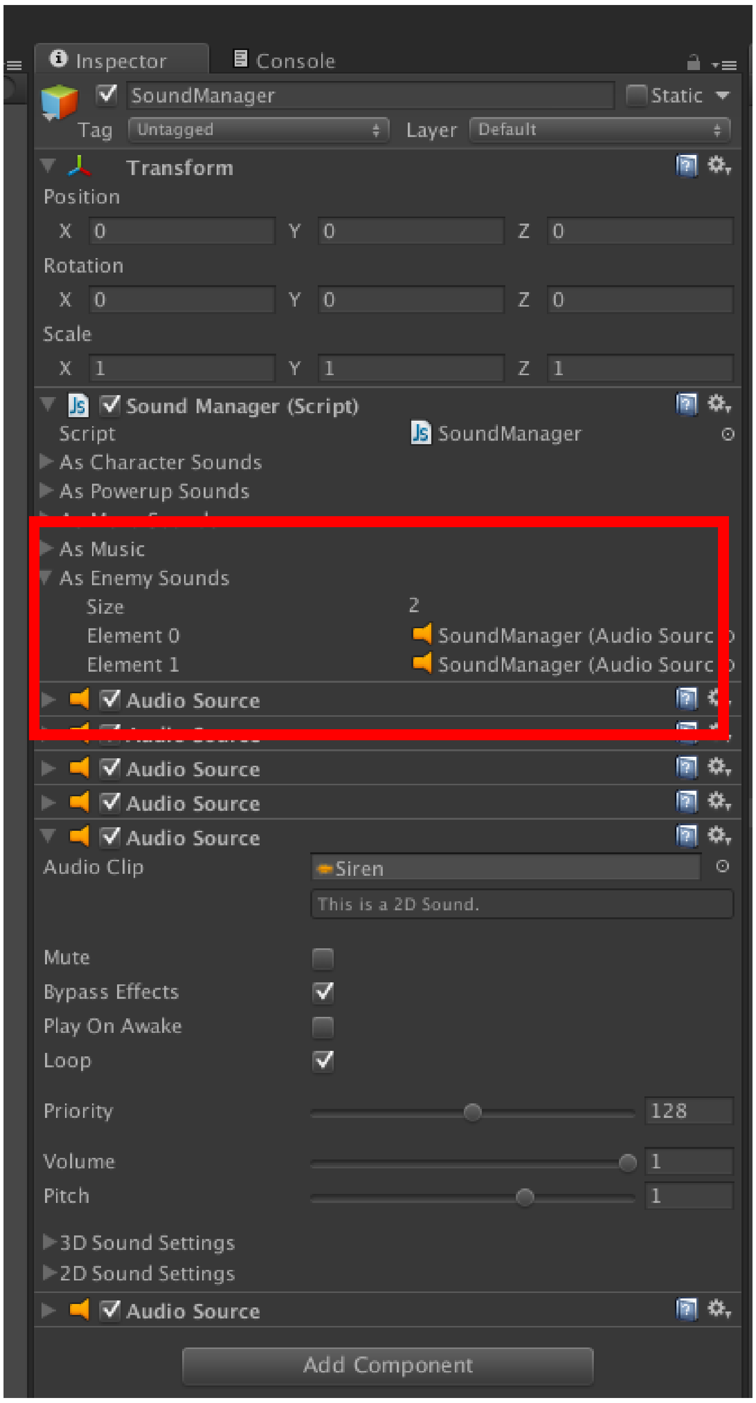
6. Drag the sound from Project > Sounds to Audio Source > Audio Clip of the newly created Audio Source Component.



Position X 0 Y 0 Z 0
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1

Audio Clip: Siren (AudioClip)
This is a 2D Sound.
Mute: ☐
Bypass Effects: ✓
Play On Awake: ☐
Loop: ✓
Priority: 128
Volume: 1
Pitch: 1

7. Increase the array size of the relevant array.


Inspector | Console

SoundManager | Static

Tag: Untagged | Layer: Default

Transform

Position: X 0, Y 0, Z 0

Rotation: X 0, Y 0, Z 0

Scale: X 1, Y 1, Z 1

Sound Manager (Script)

Script: SoundManager

As Character Sounds

As Powerup Sounds

As Music

As Enemy Sounds

Size: 2

Element 0: SoundManager (Audio Sourc

Element 1: SoundManager (Audio Sourc

Audio Source

Audio Source

Audio Source

Audio Source

Audio Source

Audio Clip: Siren

This is a 2D Sound.

Mute: ☐

Bypass Effects: ✓

Play On Awake: ☐

Loop: ✓

Priority: 128

Volume: 1

Pitch: 1

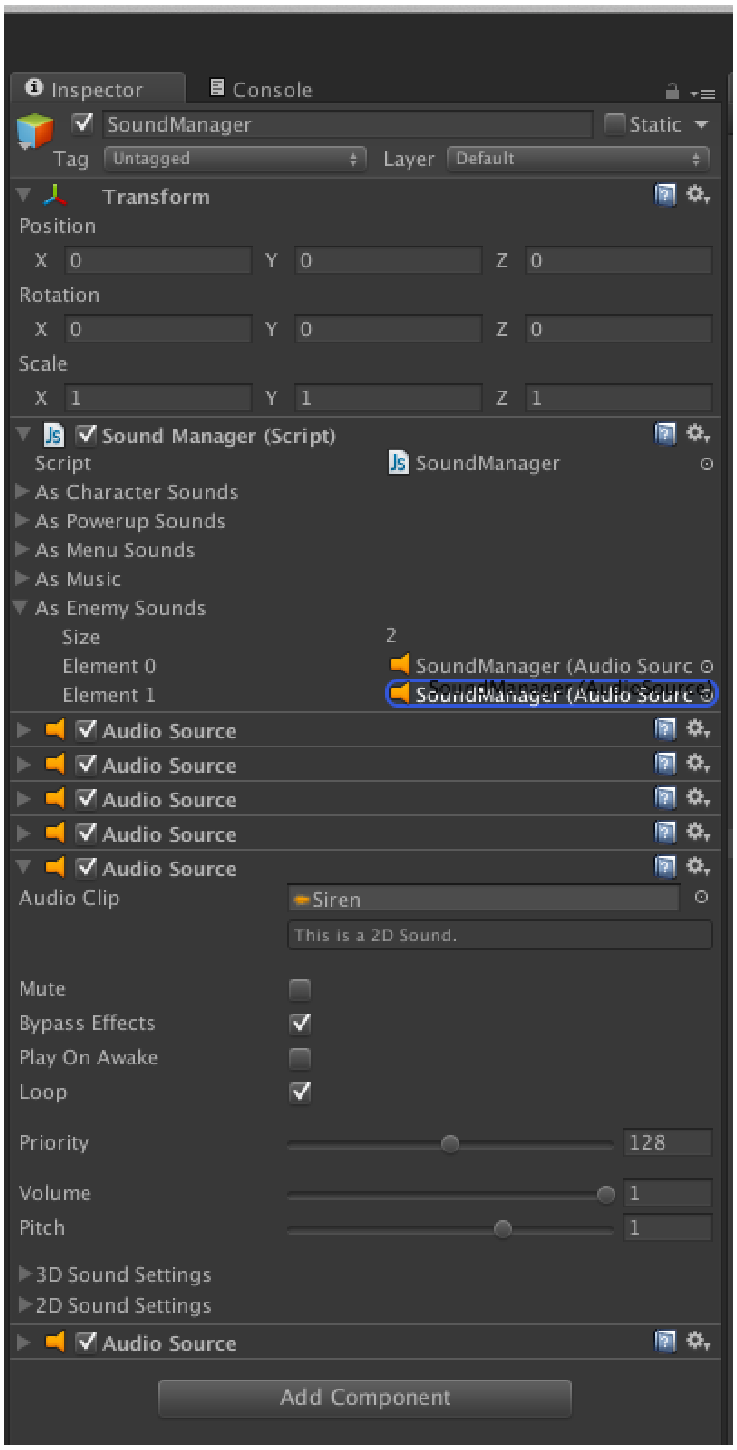
3D Sound Settings

2D Sound Settings

Audio Source

Add Component

8. Drag the Audio Source and drop it as the last index of the array.

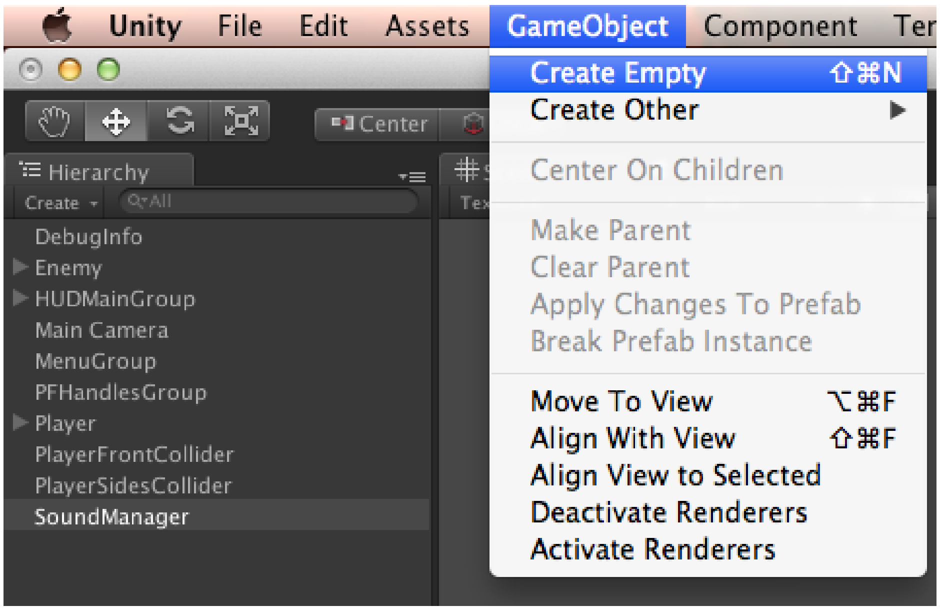


```
SoundManager
Tag  Untagged          Layer  Default
Transform
Position
  X 0        Y 0        Z 0
Rotation
  X 0        Y 0        Z 0
Scale
  X 1        Y 1        Z 1
Sound Manager (Script)
  Script                SoundManager
  As Character Sounds
  As Powerup Sounds
  As Menu Sounds
  As Music
  As Enemy Sounds
    Size                2
    Element 0           SoundManager (Audio Sourc
    Element 1           SoundManager (Audio Sourc
Audio Source
Audio Source
Audio Source
Audio Source
Audio Source
  Audio Clip            Siren
                        This is a 2D Sound.
  Mute                  ☐
  Bypass Effects        ✓
  Play On Awake         ☐
  Loop                  ✓
  Priority              128
  Volume                1
  Pitch                 1
  3D Sound Settings
  2D Sound Settings
Audio Source
```

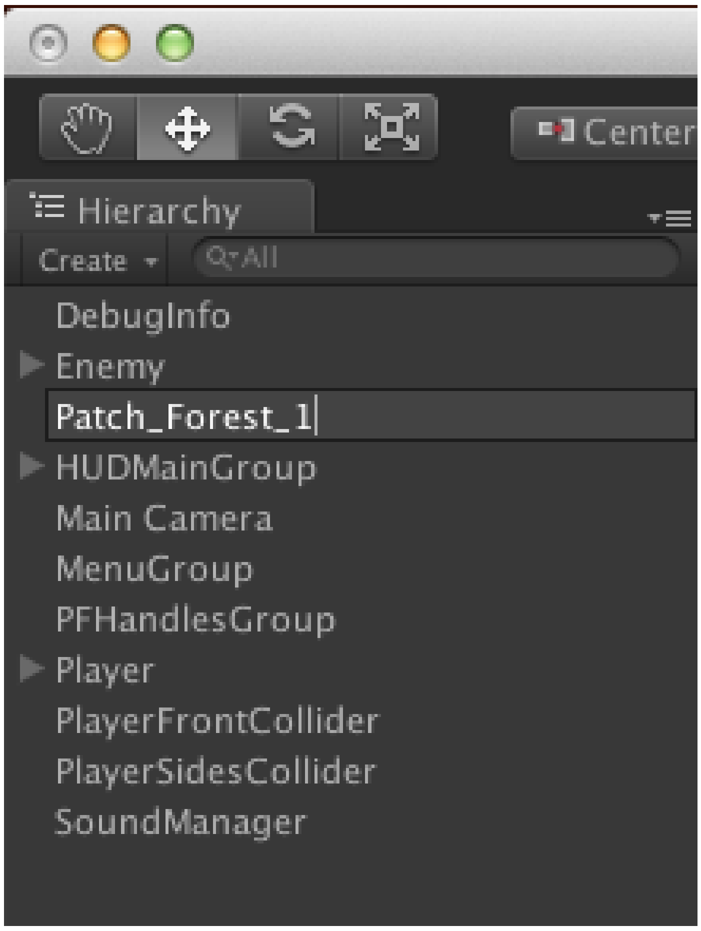
9. Use the "playSound(…)" function from SoundManager in the scripts where you need to use the newly added sound.

Adding a Patch

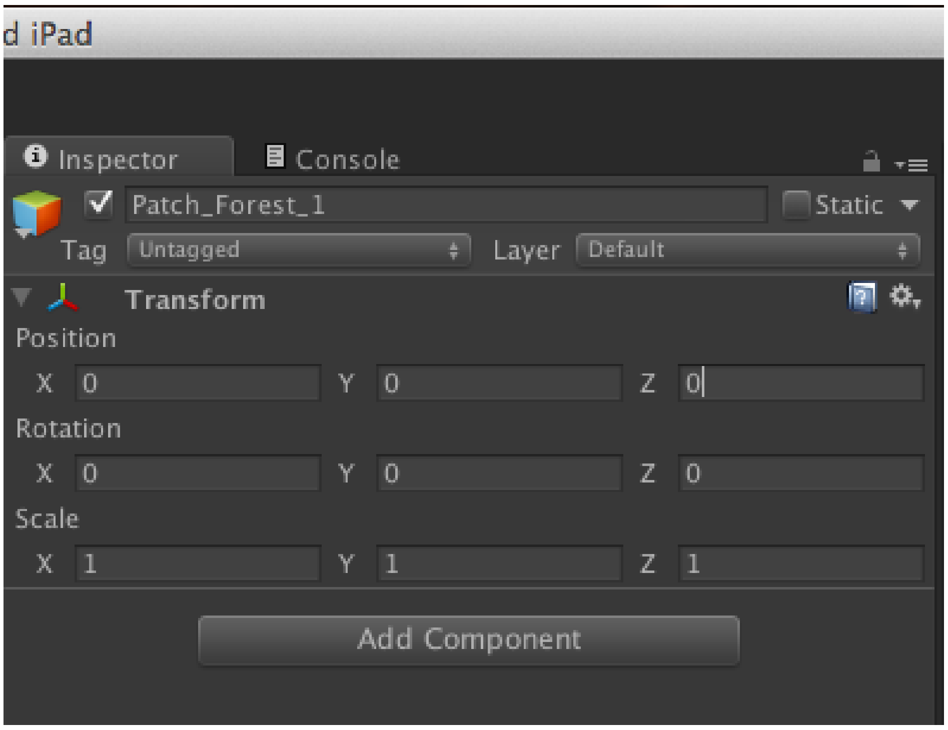
1. In the *Hierarchy* tab, add a new game object by selecting GameObject > CreateEmpty.



2. Name the empty game object.



3. Set the game object position and rotation transforms to zero.


d iPad

Inspector | Console

Patch_Forest_1 | Static

Tag: Untagged | Layer: Default

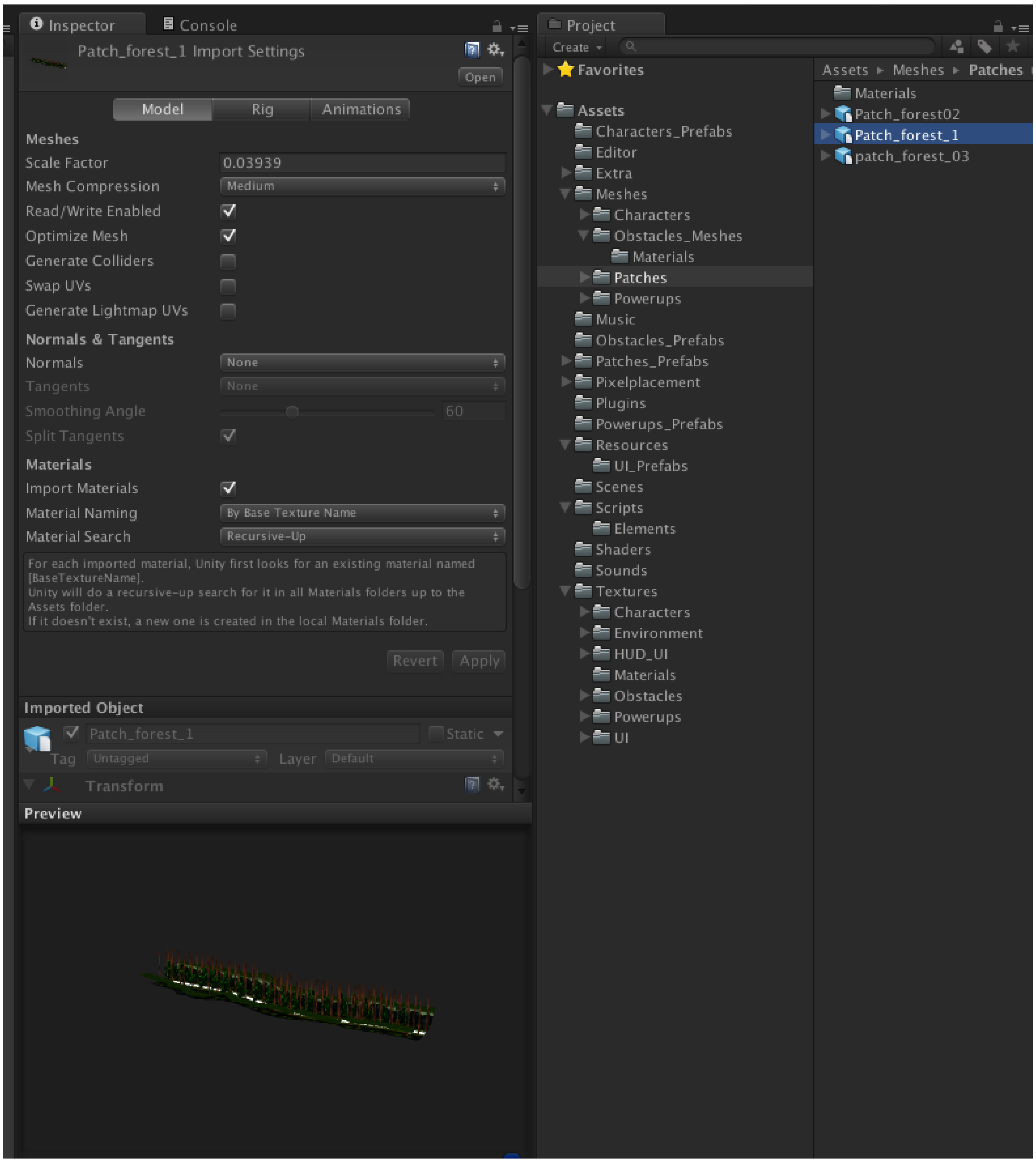
Transform

Position: X 0, Y 0, Z 0

Rotation: X 0, Y 0, Z 0

Scale: X 1, Y 1, Z 1

4. Import the patch .fbx.



Patch_forest_1 Import Settings

Meshes
Scale Factor 0.03939
Mesh Compression Medium
Read/Write Enabled ✓
Optimize Mesh ✓
Generate Colliders
Swap UVs
Generate Lightmap UVs

Normals & Tangents
Normals None
Tangents None
Smoothing Angle 60
Split Tangents ✓

Materials
Import Materials ✓
Material Naming By Base Texture Name
Material Search Recursive-Up

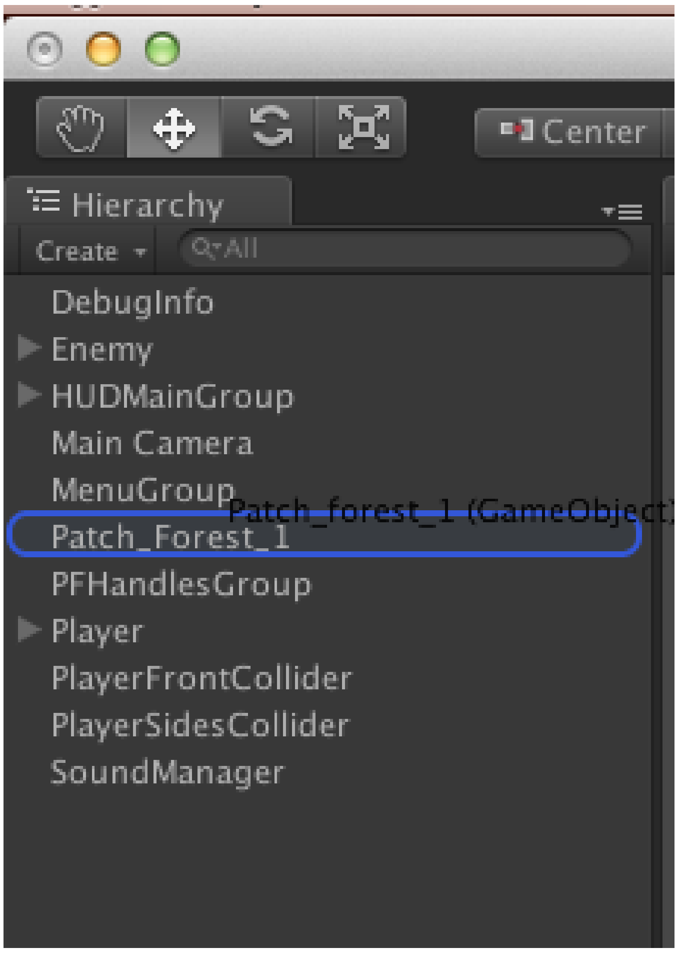
For each imported material, Unity first looks for an existing material named [BaseTextureName].
Unity will do a recursive-up search for it in all Materials folders up to the Assets folder.
If it doesn't exist, a new one is created in the local Materials folder.

Imported Object
Patch_forest_1 Static
Tag Untagged Layer Default
Transform

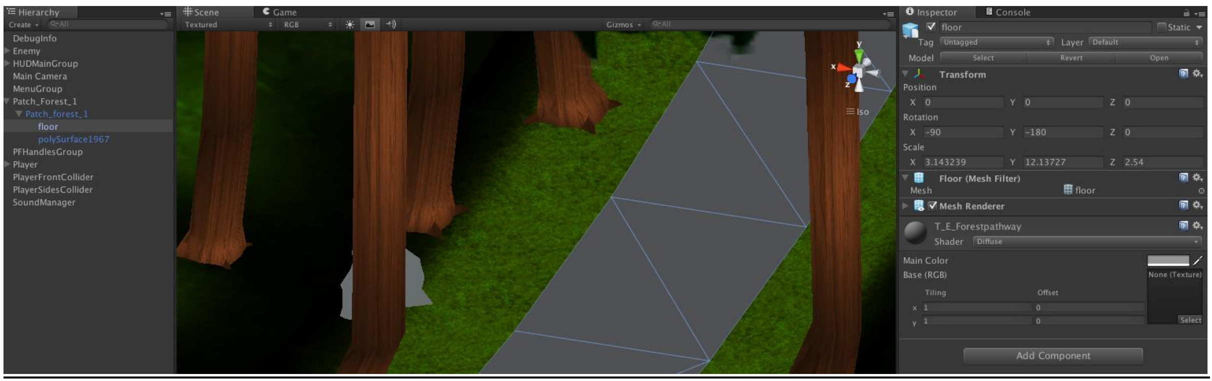
Optimization Notes:

- Use Model > Scale Factor to adjust the scale.
- Use Model > Mesh Compression to compress the imported asset as much as you can without compromising its appearance.
- Select "None" in Rig > Animation Type if your asset isn't rigged.
- Uncheck the Import Animation option in Animations > Import Animation if your asset isn't animated.

5. Drag the imported .fbx and drop it in the created empty game object.

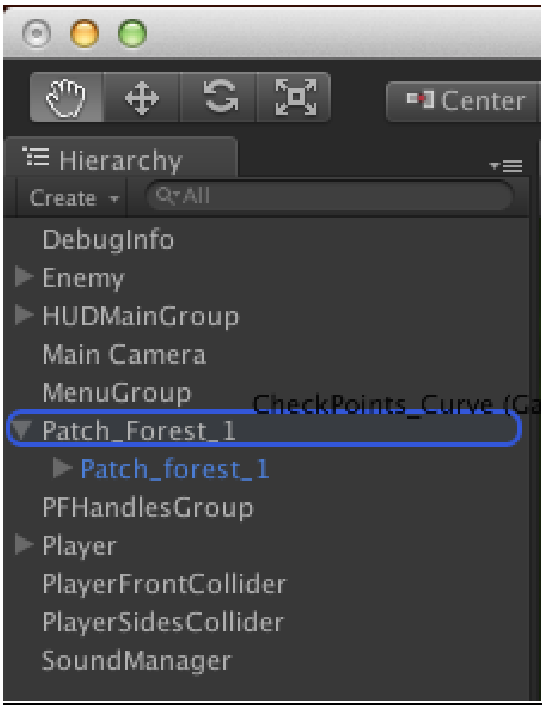


6. Select the floor of your patch where the character will run and perform two steps:
    a. Select <u>Component > Physics > Mesh Collider</u>.
    b. Select <u>Layer > Terrain_lyr.</u>

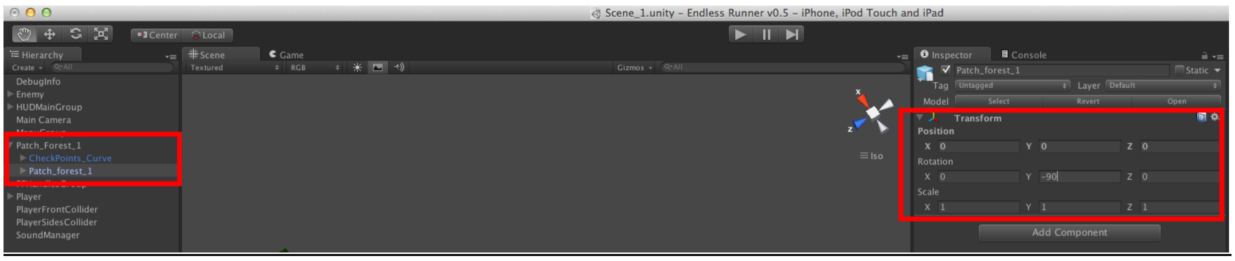


floor Static
Tag Untagged Layer Default
Model Select Revert Open
Transform
Position X 0 Y 0 Z 0
Rotation X -90 Y -180 Z 0
Scale X 3.143239 Y 12.13727 Z 2.54
Floor (Mesh Filter)
Mesh floor
Mesh Renderer
T_E_Forestpathway
Shader Diffuse
Main Color
Base (RGB) None (Texture)
Tiling x 1 y 1 Offset x 0 y 0

7. Select <u>Project > Patches_Prefabs > Sample_CPs.</u>
8. Drag the “CheckPoints_Straight” prefab into the patch prefab if your patch’s path is in a straight line. If the path is not in a straight line, drag the “CheckPoints_Curve” instead.



9. Rotate the patch if it does not have the same orientation as the spline.



Patch_forest_1
Tag: Untagged   Layer: Default
Model: Select | Revert | Open

Transform
Position: X 0   Y 0   Z 0
Rotation: X 0   Y -90   Z 0
Scale: X 1   Y 1   Z 1

Adjusting CPs of a Straight Patch

a. Adjust the <u>Model > Scale Factor</u> of your .fbx and make sure the patch is exactly the size of the spline.

Note: Make sure the spline is centered on the path.

b. Select <u>Hierarchy > *patch name*.</u>

c. Click *CheckPoints_Straight*.

d. From the title bar, select <u>Custom > Patch CP Generator</u>. (This saves all the changes you have made to the spline)

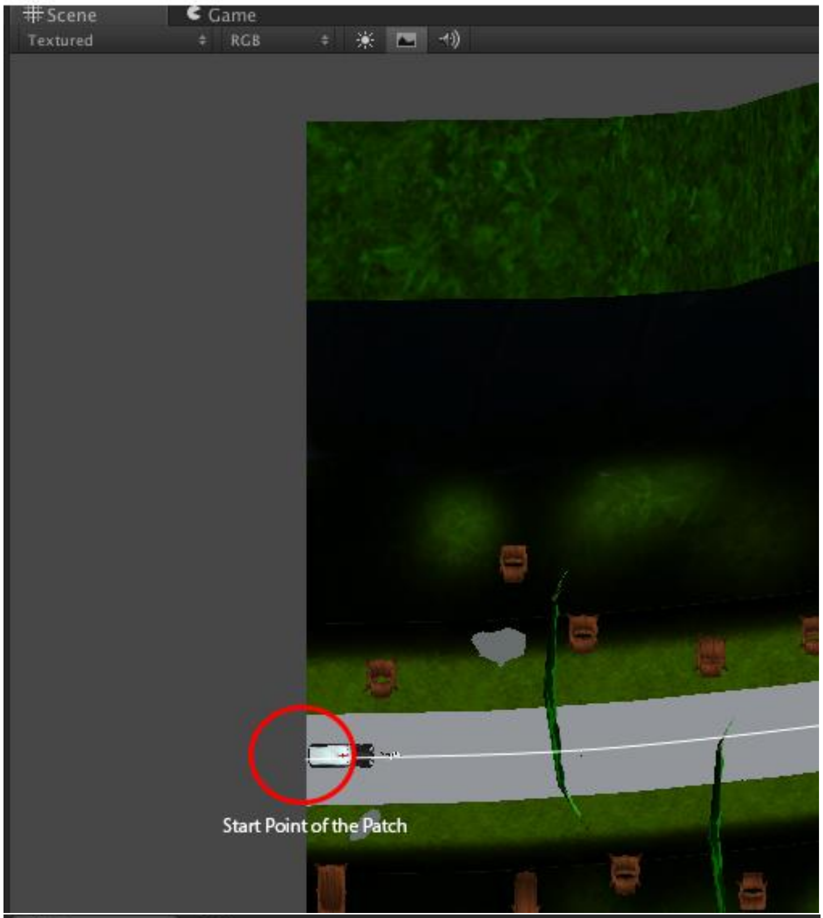
e. Select the **patch name*.*

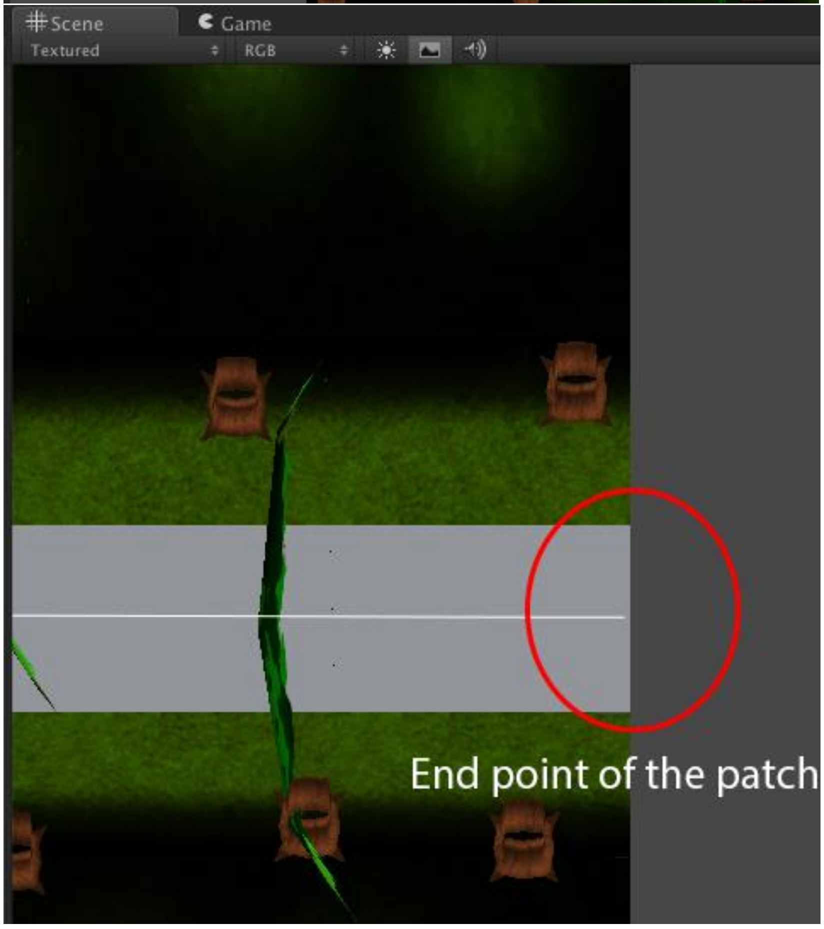
f. Drag the patch to <u>Project > Patches_Prefabs</u> to save the created prefab.

Adjusting CPs of a Curved Patch

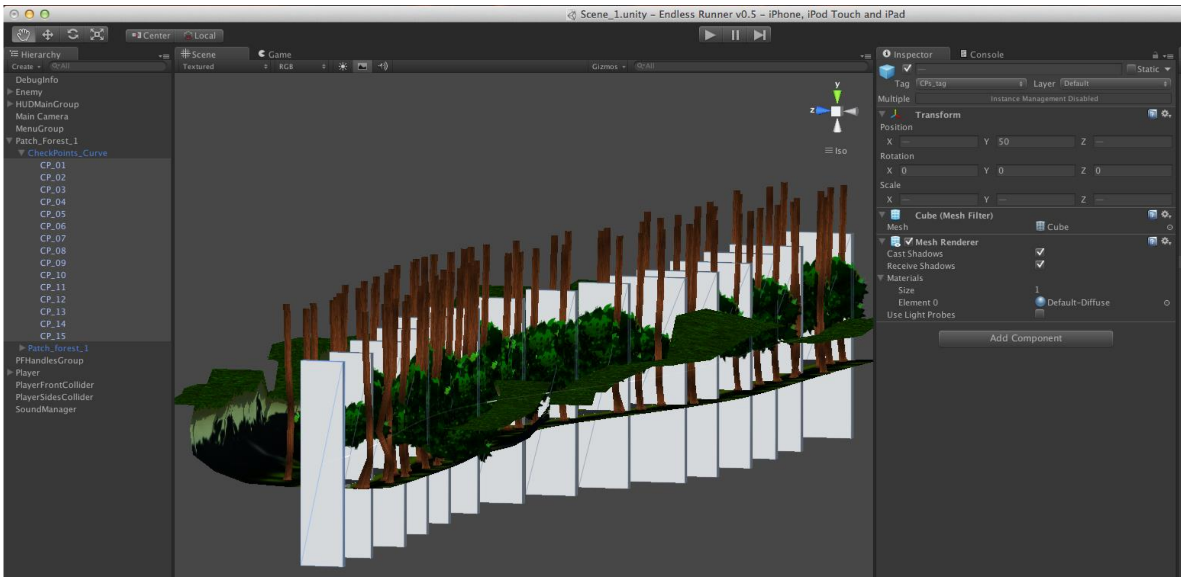
a. Adjust the <u>Model > Scale Factor</u> of your .fbx and make sure the patch is exactly the size of the spline.

Note: The start and end points of the spline should be centered on the start and end of the path.

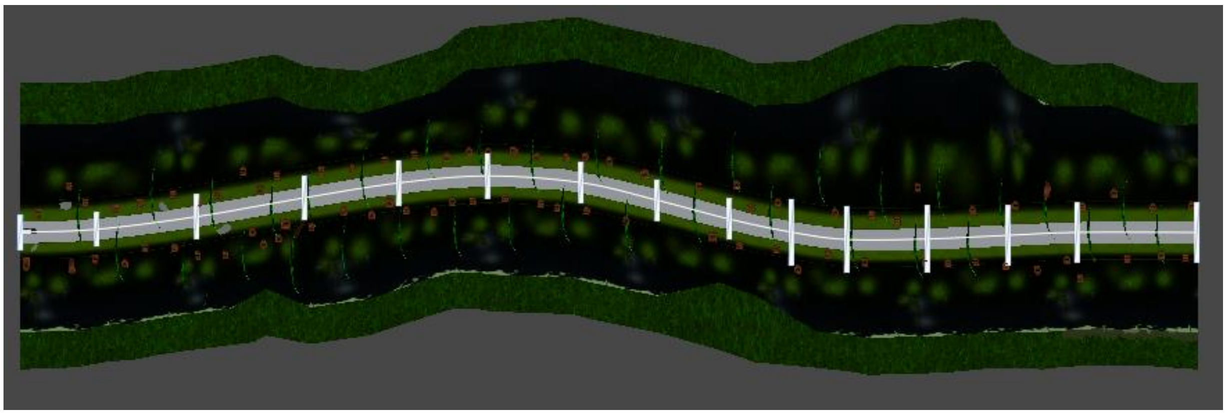

Scene
Game
Textured
RGB
Start Point of the Patch


Scene
Game
Textured
RGB
End point of the patch

b. Select <u>Hierarchy > *patch name* > CheckPoints_Curve</u>.
c. Select all the CPs.
d. Check the *Mesh Renderer* component from <u>Inspector > Mesh Renderer.</u> (This will display all the Check Points that shape the spline)



e. Adjust the CPs so that the spline is centered on the path at every point on that path. You can only change the position of the CPs; changing their rotation or scale will have no effect on the spline.

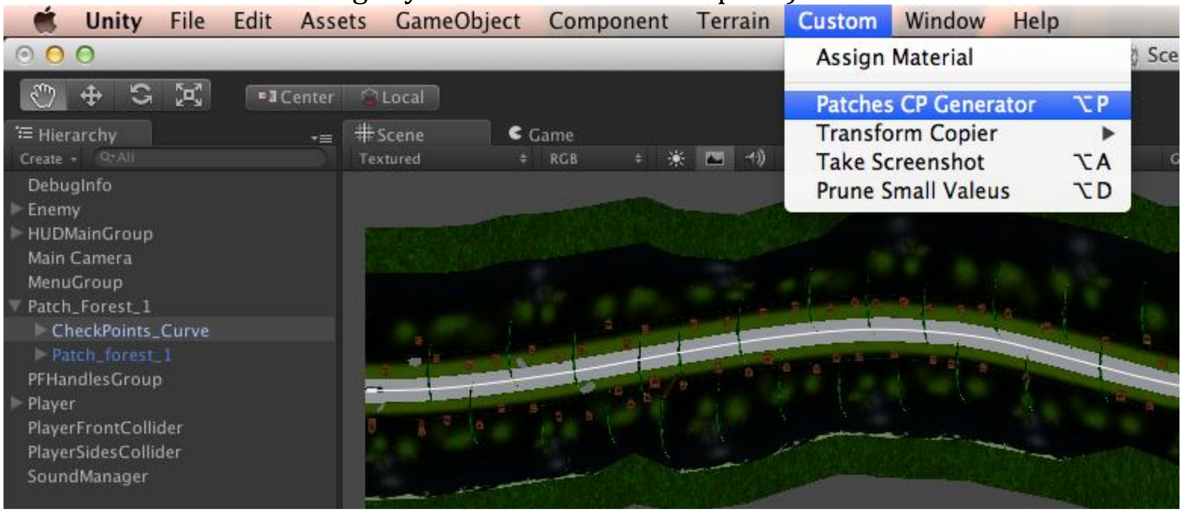


Notes:

- Make sure your Editor's camera is in Isometric mode. (You can change the camera mode by clicking the cube in the top right corner of the screen.)
- Make sure the Editor is in top view.
- You can add or remove CPs as required but make sure not to miss a number in the sequence.

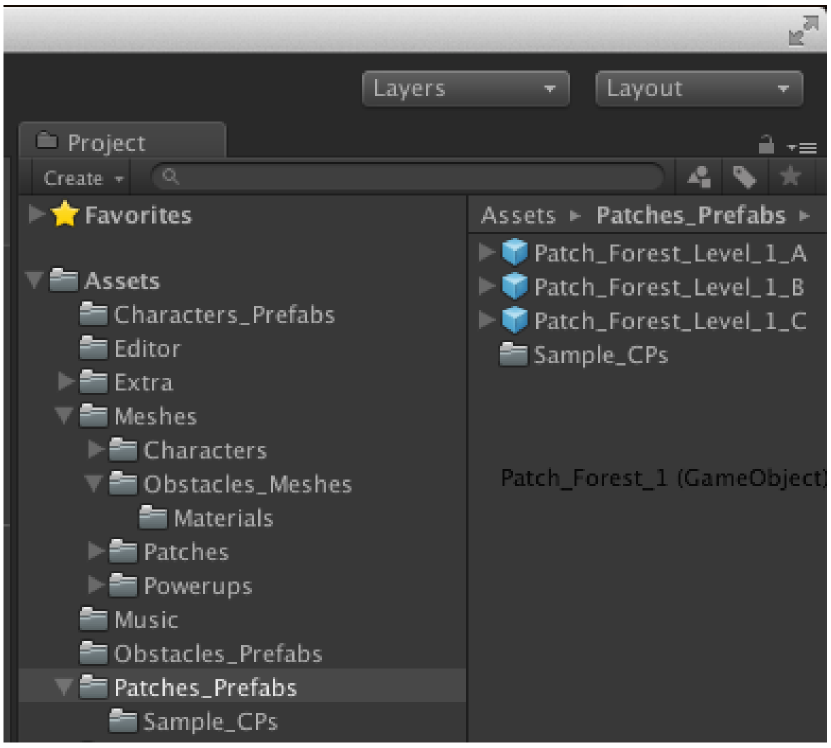
f. Check off the *Mesh Renderer* component from <u>Inspector > Mesh Renderer</u> when you're done adjusting the spline.
g. Click *CheckPoints_Straight/ CheckPoints_Curve* or whatever you have named the prefab containing the checkpoints.

h. From the title bar, select Custom > Patch CP Generator. (This saves all the changes you have made to the spline)

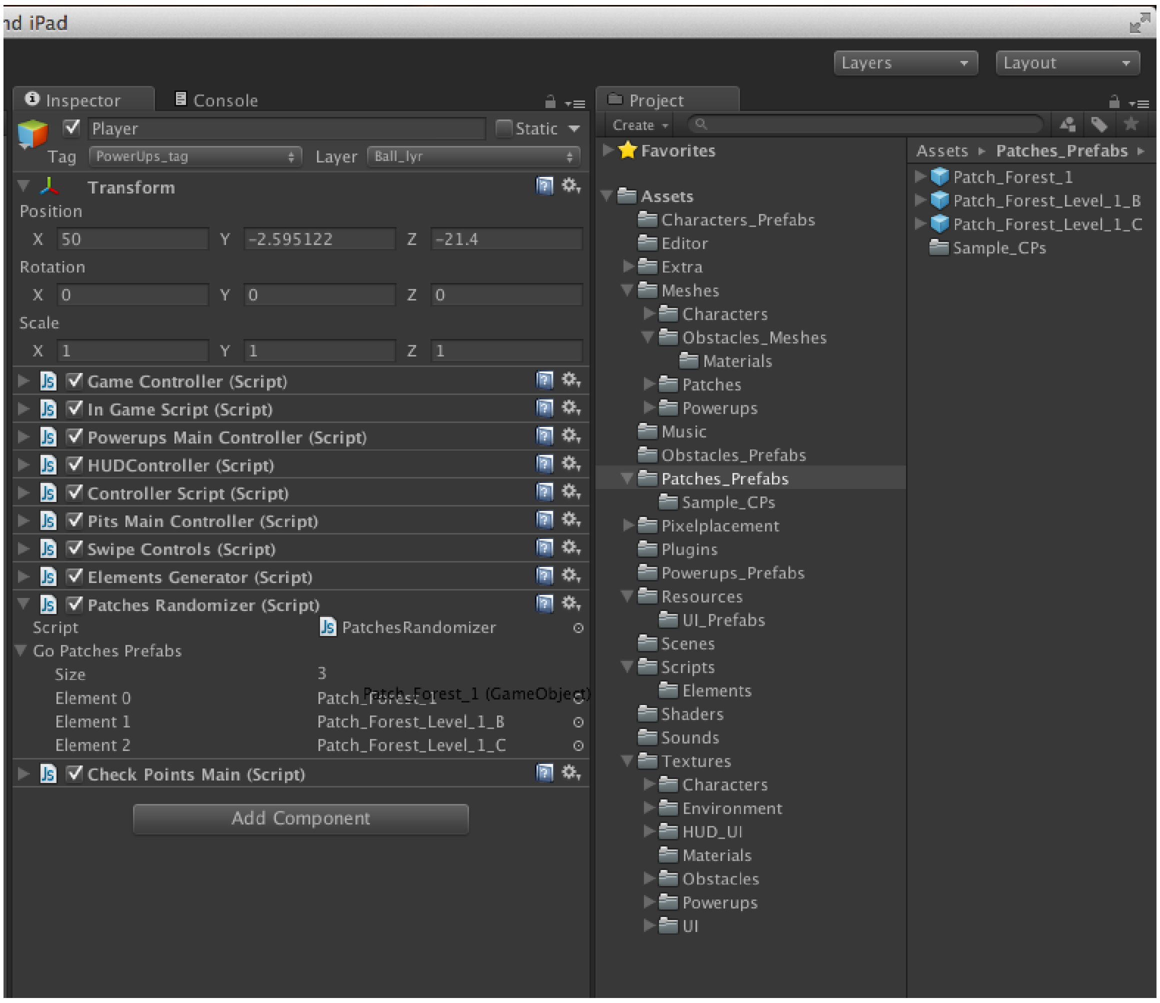


i. Select the *patch name*.

j. Drag the patch to Project > Patches_Prefabs to save the created prefab.



10. Once the patch has been saved, select Player > Patches Randomizer from the Hierarchy.
11. Increase the array size of "Go Patches Prefabs".
12. Drag and add the newly created patch from Project > Assets > Patches_Prefabs, to the *Go Patches Prefabs* array.



```
Player                                   Static
Tag  PowerUps_tag        Layer  Ball_lyr
Transform
Position   X 50    Y -2.595122   Z -21.4
Rotation   X 0     Y 0           Z 0
Scale      X 1     Y 1           Z 1
Game Controller (Script)
In Game Script (Script)
Powerups Main Controller (Script)
HUDController (Script)
Controller Script (Script)
Pits Main Controller (Script)
Swipe Controls (Script)
Elements Generator (Script)
Patches Randomizer (Script)
  Script                 PatchesRandomizer
  Go Patches Prefabs
    Size                 3
    Element 0            Patch_Forest_1   Patch_Forest_1 (GameObject)
    Element 1            Patch_Forest_Level_1_B
    Element 2            Patch_Forest_Level_1_C
Check Points Main (Script)
```

13. Remove the patch from the *Hierarchy* tab.

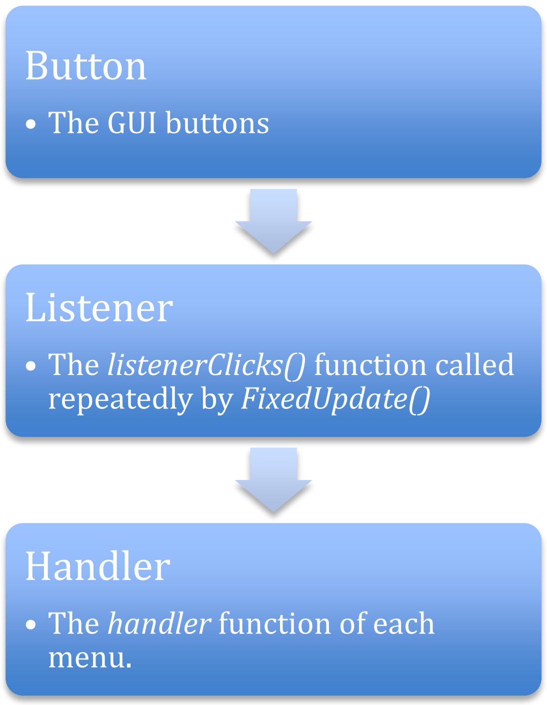
Custom Menus

Menu Script Architecture (Custom Menu)

The architecture of the menu controller, which is the *MenuScript.js* script, relies on a three-tier architecture. The first layer consists of the buttons that in-turn consist of a plane, a material and a box collider.

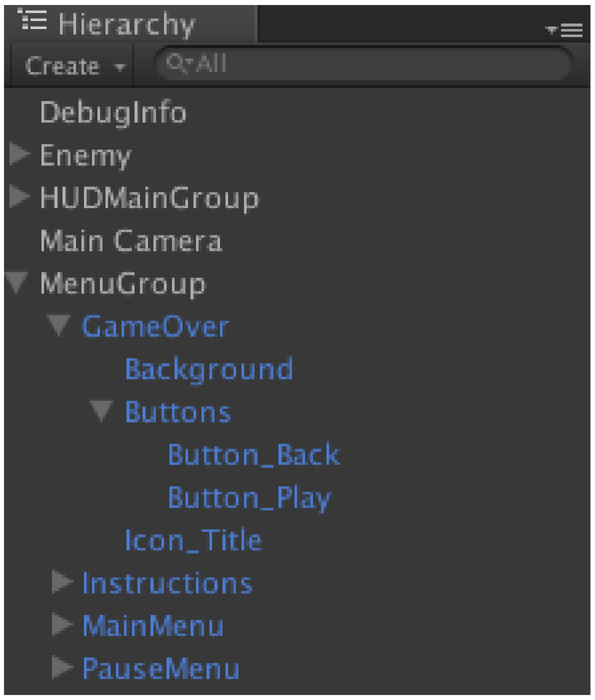
The second layer is the listener that consists of a function named *listenerClicks()* in the *MenuScript.js*. This function is called by the *FixedUpdate()* function to listen for any interaction with the screen. If the user taps on the screen, the camera raycasts the menu in search of the button tapped. If the raycast finds a collider in its way, a relevant handler is called. By relevant, we mean the handler of the active menu.

Finally, the third layer consists of handler functions. Each menu has a single handler function that defines the implementation of all the buttons the menu has. When the listener function calls the handler function, it also passes the transform of the button tapped as a parameter. The handler function uses this transform and compares it with the array of buttons' transforms, to check which button's implementation to execute.


Button
• The GUI buttons
Listener
• The *listenerClicks()* function called repeatedly by *FixedUpdate()*
Handler
• The *handler* function of each menu.

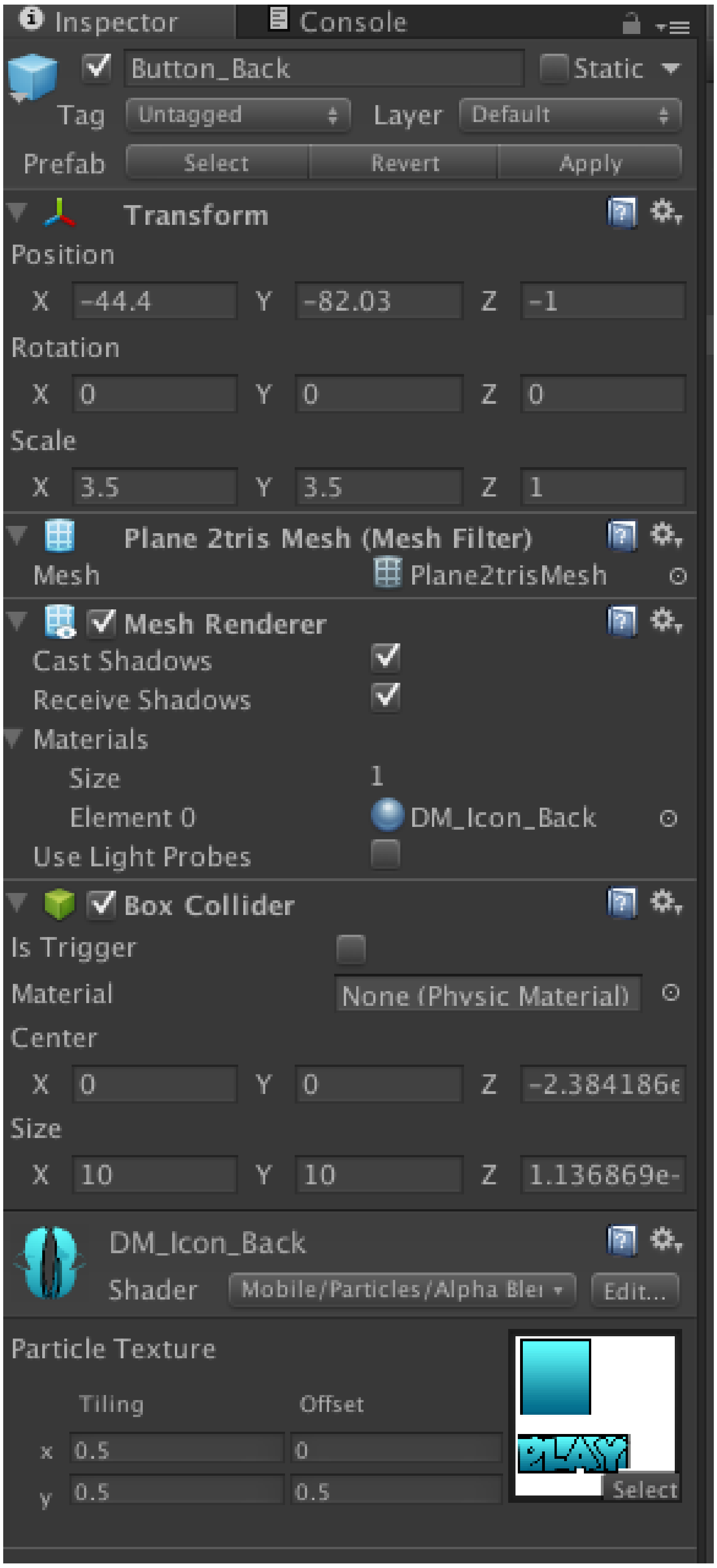
Adding a Custom Menu

The *MenuGroup* prefab in the Hierarchy tab is the parent of all the menu prefabs used. The menus are present in the scene at all times. They are displayed by bringing them in front of the HUD Camera by the *MenuScript.js*. The project uses the low-poly plane named *Plane2tirsMesh* located at <u>Project > Assets > Editor</u>. As always, the lower the poly-count, the better the performance.



By convention in the project, an empty GameObject is created to contain the backgrounds, icons and buttons (*GameOver* is used as an example in the above screen shot). All of the buttons are enclosed in yet another empty GameObject named *Buttons*. Each button is a plane that also has a box collider as a component to detect raycasts (raycasts are used to detect taps on buttons).

In the following example the button named *Button_Back* is shown with all its components. Notice the low-poly plane used instead of the default plane used by Unity. Secondly, observe that a box collider has been attached to the button to detect raycasts. A box collider can be added by selecting the plane and then selecting <u>Component > Physics > Box Collider</u> from the title bar.


Inspector | Console

Button_Back — Static
Tag: Untagged — Layer: Default
Prefab: Select | Revert | Apply

Transform

Position: X -44.4, Y -82.03, Z -1
Rotation: X 0, Y 0, Z 0
Scale: X 3.5, Y 3.5, Z 1

Plane 2tris Mesh (Mesh Filter)

Mesh: Plane2trisMesh

Mesh Renderer ✓

Cast Shadows: ✓
Receive Shadows: ✓
Materials
- Size: 1
- Element 0: DM_Icon_Back

Use Light Probes: ☐

Box Collider ✓

Is Trigger: ☐
Material: None (Physic Material)
Center: X 0, Y 0, Z -2.384186e
Size: X 10, Y 10, Z 1.136869e-

DM_Icon_Back

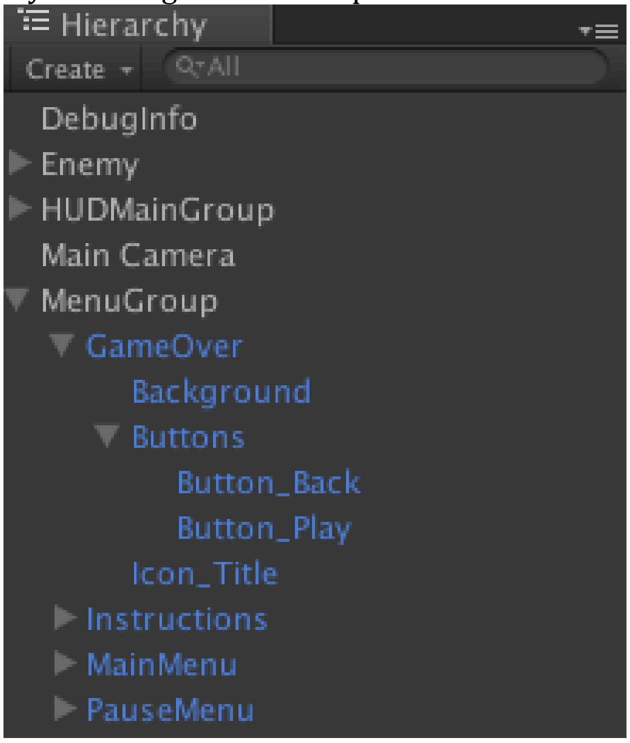
Shader: Mobile/Particles/Alpha Blei — Edit...

Particle Texture

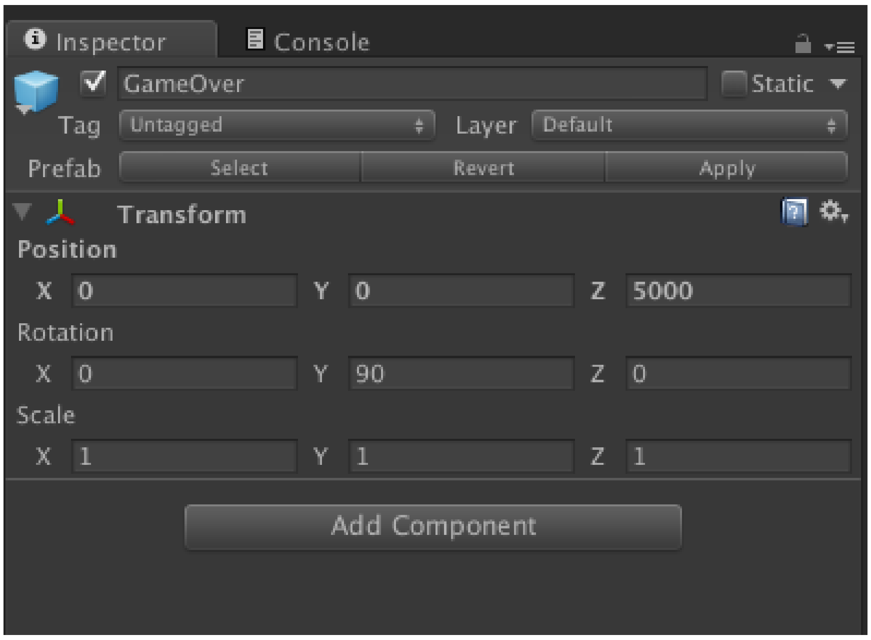
	Tiling	Offset
x	0.5	0
y	0.5	0.5

1. As mentioned before, make sure all components of the menu are enclosed in a single game object. (For the sake of example, lets call the menu "GameOver")

2. Place the *GameOver* prefab in the *MenuGroup* prefab located in the Hierarchy tab along with its components.



3. Set the position of the *GameOver* prefab at 0,0,5000. (This brings the prefab into the HUD Camera as the HUD Camera is located at (0,0,5000))



Inspector | Console

GameOver | Static

Tag: Untagged | Layer: Default

Prefab: Select | Revert | Apply

Transform

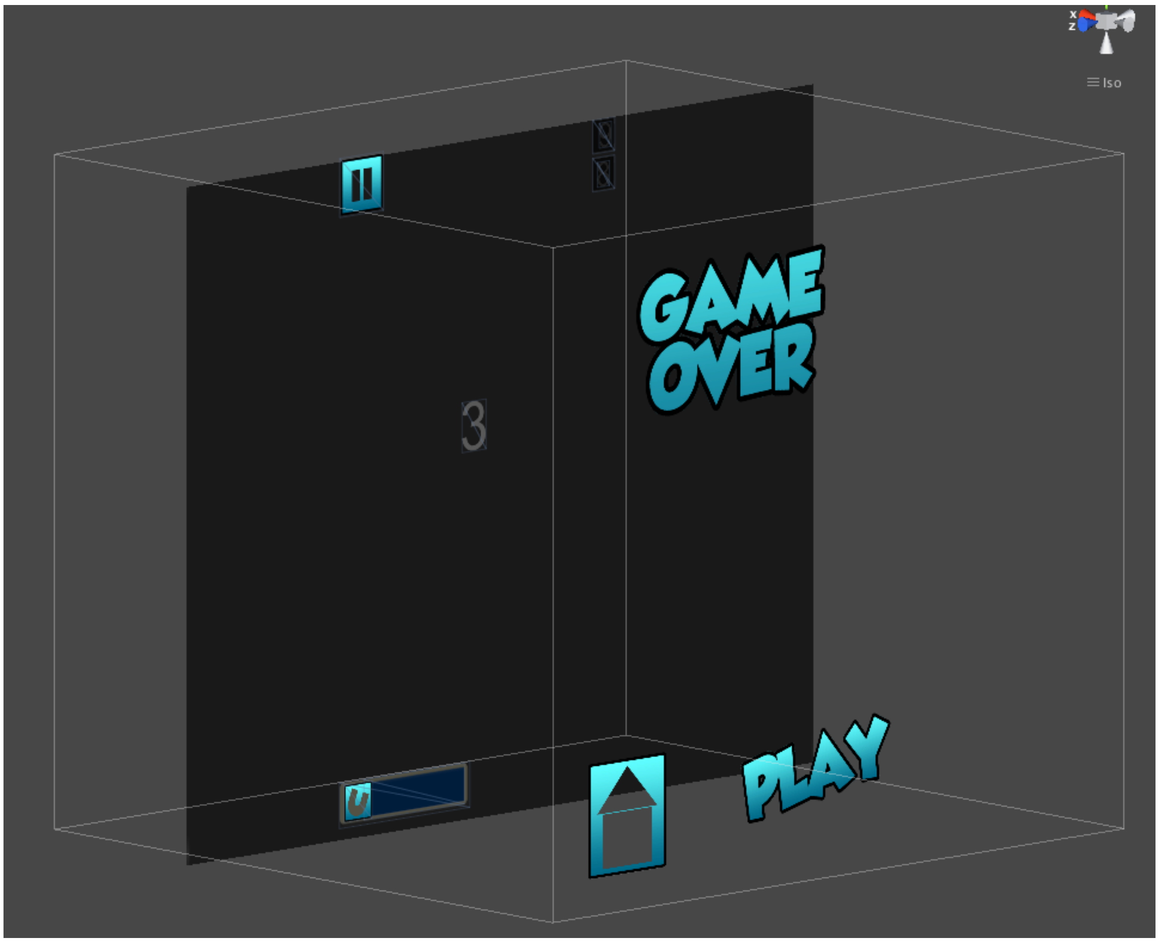
Position: X 0 | Y 0 | Z 5000

Rotation: X 0 | Y 90 | Z 0

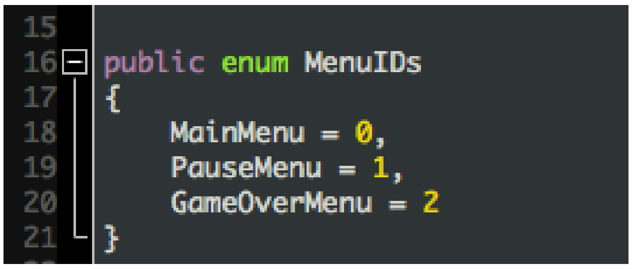
Scale: X 1 | Y 1 | Z 1

Add Component

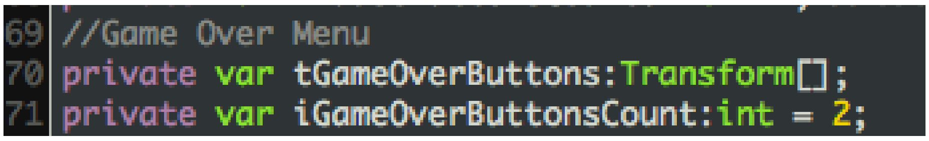
4. Double click on Hierarchy > HUDMainGroup to display it in the Editor and make sure the menu items are facing towards the negative x-axis. The screen shot shows the menu title, the buttons, the semi-transparent background and also the HUD elements in the background (ignore these while designing a menu).



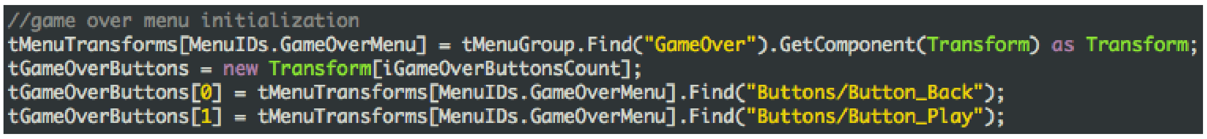
5. Open the *MenuScript.js* from Project > Assets > UI > Scripts or C# Scripts.
6. Add the name of the newly created menu in the *MenuIDs* enum.


```
public enum MenuIDs
{
    MainMenu = 0,
    PauseMenu = 1,
    GameOverMenu = 2
}
```

7. Create a Transform type array to store the transforms of the buttons in the GameOver menu.
8. Create an integer type variable to store the number of button in the GameOver menu.

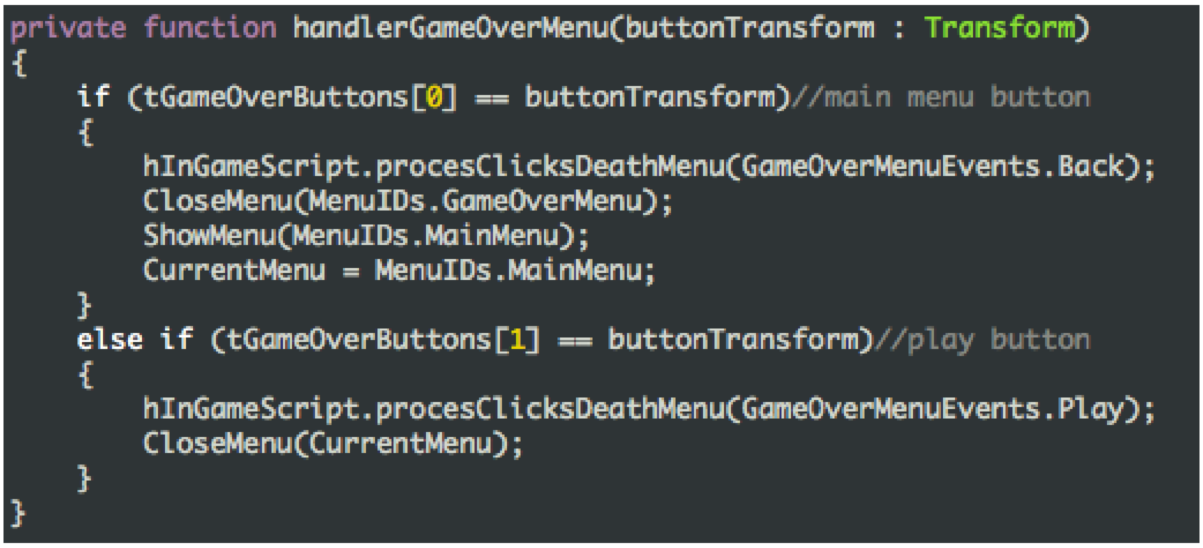

```
//Game Over Menu
private var tGameOverButtons:Transform[];
private var iGameOverButtonsCount:int = 2;
```

9. Store the *GameOver* menu's transform the *tMenuTransform* array.
10. Initialize the transform type array that holds the references of the buttons in the *GameOver* menu.



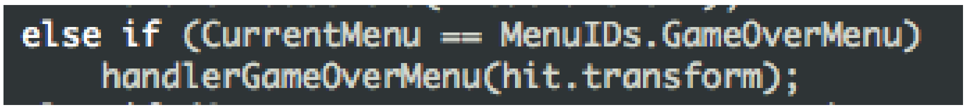
```
//game over menu initialization
tMenuTransforms[MenuIDs.GameOverMenu] = tMenuGroup.Find("GameOver").GetComponent(Transform) as Transform;
tGameOverButtons = new Transform[iGameOverButtonsCount];
tGameOverButtons[0] = tMenuTransforms[MenuIDs.GameOverMenu].Find("Buttons/Button_Back");
tGameOverButtons[1] = tMenuTransforms[MenuIDs.GameOverMenu].Find("Buttons/Button_Play");
```

11. Create a handler function for *GameOver* menu to define the implementation of its buttons. In our case, we have two buttons.



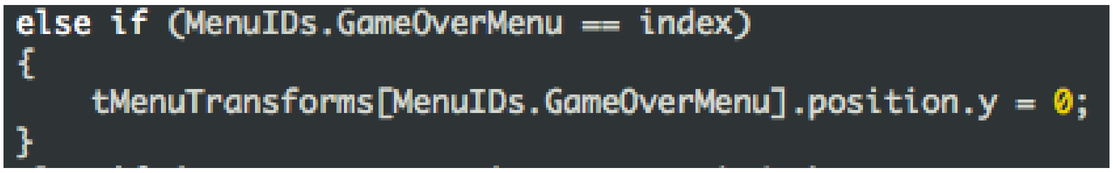
```
private function handlerGameOverMenu(buttonTransform : Transform)
{
    if (tGameOverButtons[0] == buttonTransform)//main menu button
    {
        hInGameScript.procesClicksDeathMenu(GameOverMenuEvents.Back);
        CloseMenu(MenuIDs.GameOverMenu);
        ShowMenu(MenuIDs.MainMenu);
        CurrentMenu = MenuIDs.MainMenu;
    }
    else if (tGameOverButtons[1] == buttonTransform)//play button
    {
        hInGameScript.procesClicksDeathMenu(GameOverMenuEvents.Play);
        CloseMenu(CurrentMenu);
    }
}
```

12. Add a new case in the *listenerClicks* function. This function will call the *GameOver* handler function if its button is pressed.



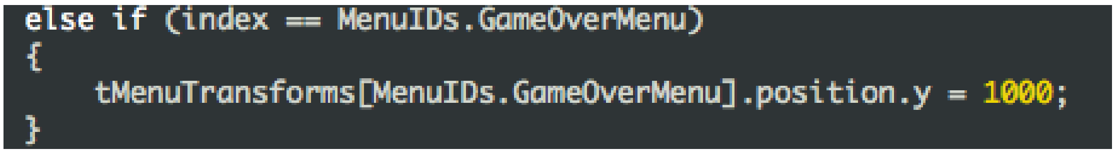
```
else if (CurrentMenu == MenuIDs.GameOverMenu)
    handlerGameOverMenu(hit.transform);
```

13. Add a new case in the ShowMenu(…) function to display the *GameOver* menu prefab when needed. Setting the prefab's y position to zero brings it in front of the HUD Camera.



```
else if (MenuIDs.GameOverMenu == index)
{
    tMenuTransforms[MenuIDs.GameOverMenu].position.y = 0;
}
```

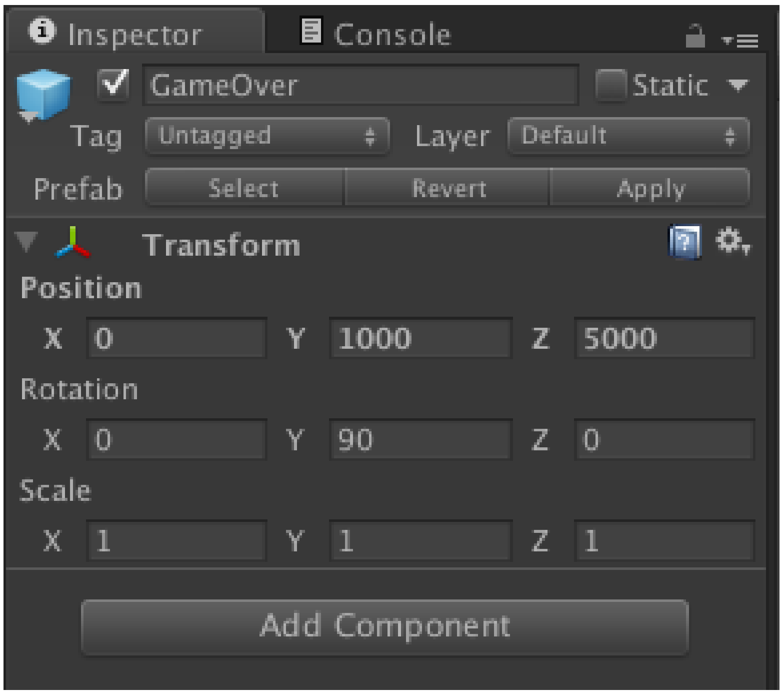
14. Add a case and its implementation in the CloseMenu(…) function as shown below. Setting the prefab's y position to 1000 moves it away from the HUD Camera.



```
else if (index == MenuIDs.GameOverMenu)
{
    tMenuTransforms[MenuIDs.GameOverMenu].position.y = 1000;
}
```

Note: Whenever you need to use the created menu, call the ShowMenu(…) function passing the appropriate parameter. Similarly, to close the menu, call the CloseMenu(…) function with the same parameter.

15. Set the position of the newly created GameOver menu prefab to (0,1000,5000). This will remove it from the HUD Camera in default conditions.


Inspector | Console

GameOver | Static

Tag: Untagged | Layer: Default

Prefab: Select | Revert | Apply

Transform

Position: X 0 | Y 1000 | Z 5000

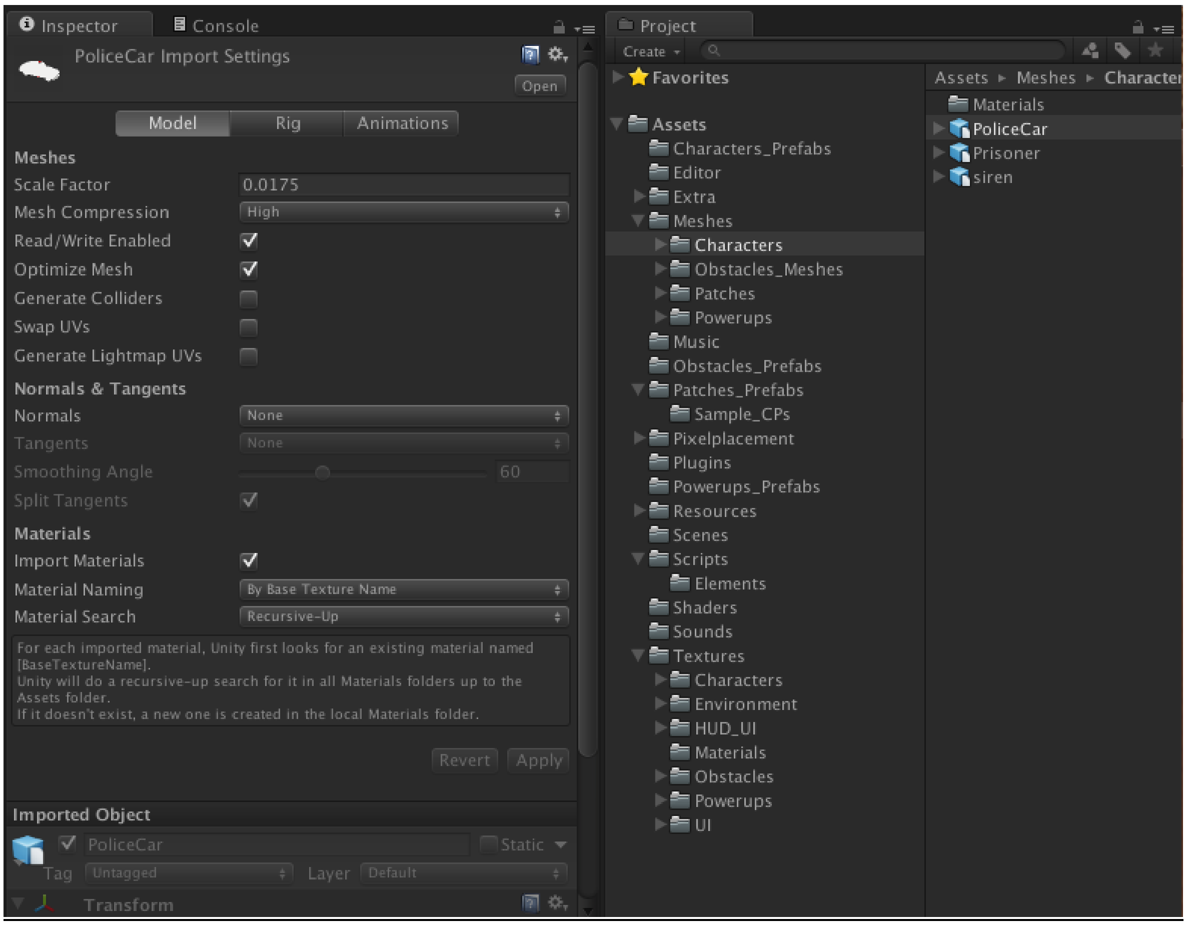
Rotation: X 0 | Y 90 | Z 0

Scale: X 1 | Y 1 | Z 1

Add Component

Replacing the Enemy Character

1. Import the player .fbx that will be used as the main enemy.
   Optimization Notes:
   - Use Model > Scale Factor to adjust the scale.
   - Use Model > Mesh Compression to compress the imported asset as much as you can without compromising its appearance.
   - Select "None" in Rig > Animation Type if your asset isn't rigged.
   - Uncheck the Import Animation option in Animations > Import Animation if your asset isn't animated.



PoliceCar Import Settings

Meshes
Scale Factor 0.0175
Mesh Compression High
Read/Write Enabled ✓
Optimize Mesh ✓
Generate Colliders ☐
Swap UVs ☐
Generate Lightmap UVs ☐

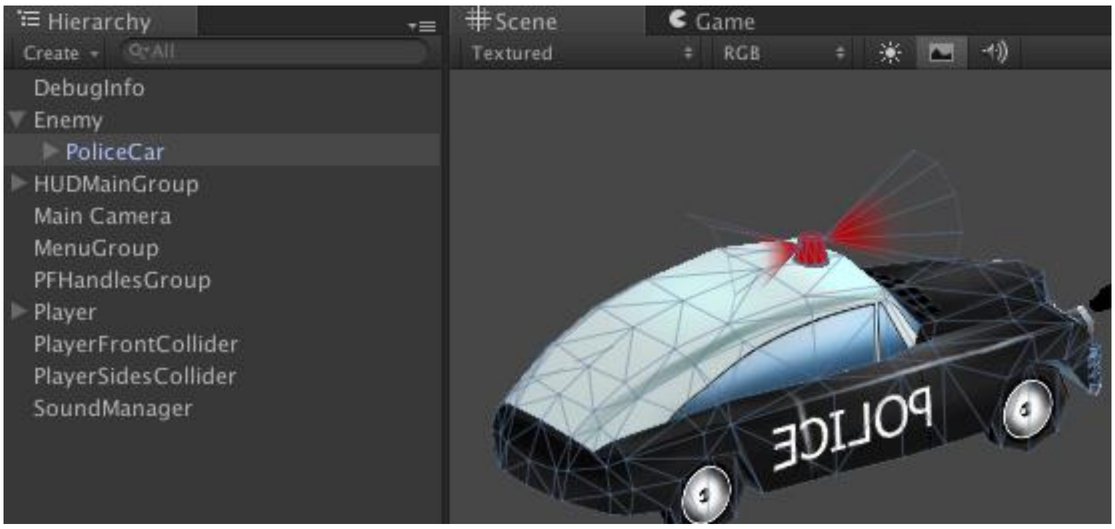
Normals & Tangents
Normals None
Tangents None
Smoothing Angle 60
Split Tangents ✓

Materials
Import Materials ✓
Material Naming By Base Texture Name
Material Search Recursive-Up

For each imported material, Unity first looks for an existing material named [BaseTextureName].
Unity will do a recursive-up search for it in all Materials folders up to the Assets folder.
If it doesn't exist, a new one is created in the local Materials folder.

Imported Object
PoliceCar Static
Tag Untagged Layer Default

2. Drag the imported asset in Hierarchy > Enemy.

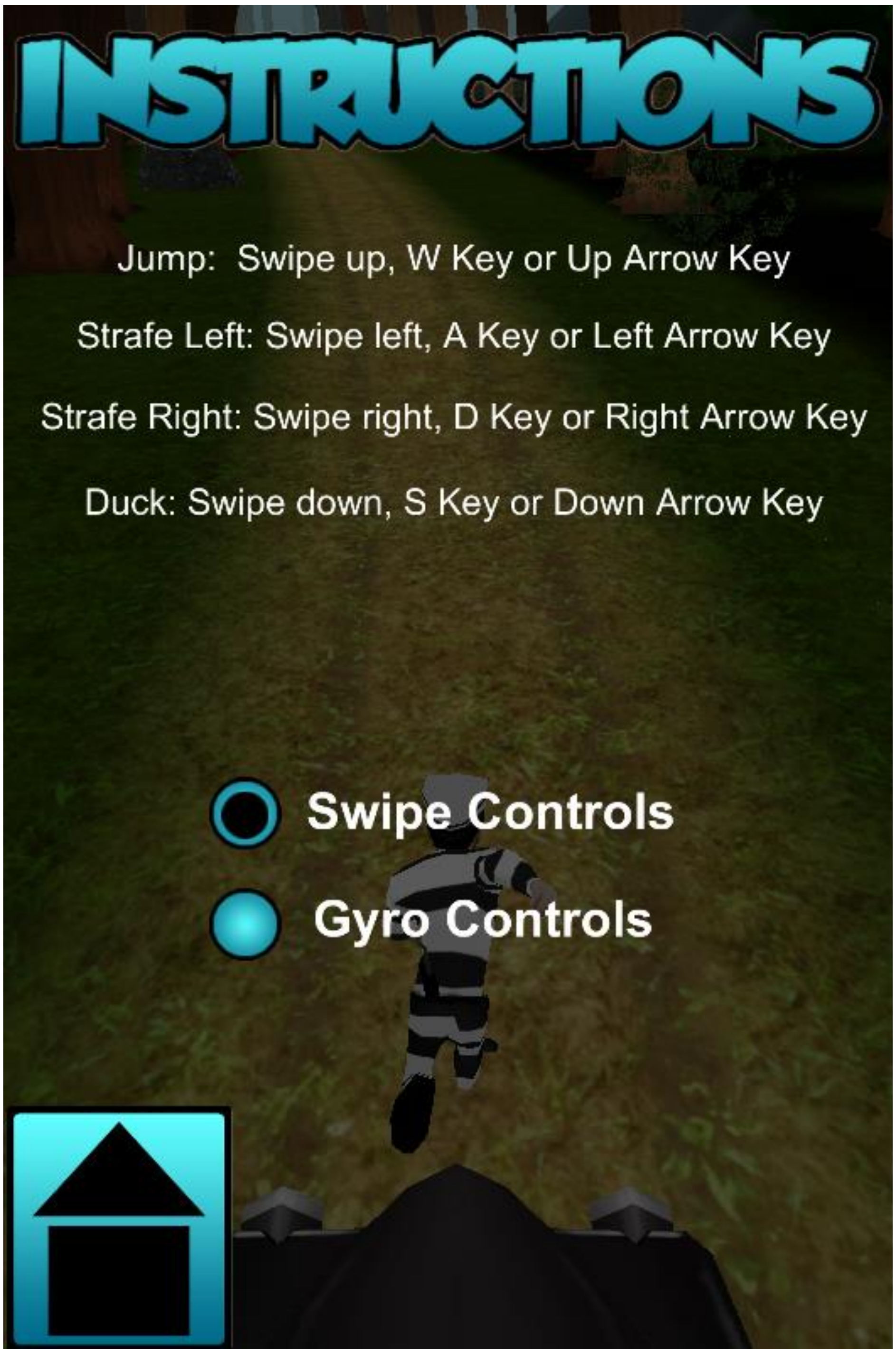


3. Change the implementation in the Project > Assets > Scripts > *EnemyController.js* as needed.

How to Switch between Gyro and Swipe Controls

The type of controls can be switched from the information menu. Clicking the following link located on the main menu accesses the instructions menu:

The radio buttons in the in the instructions menu can be used to toggle between swipe and gyro controls.



INSTRUCTIONS

Jump: Swipe up, W Key or Up Arrow Key

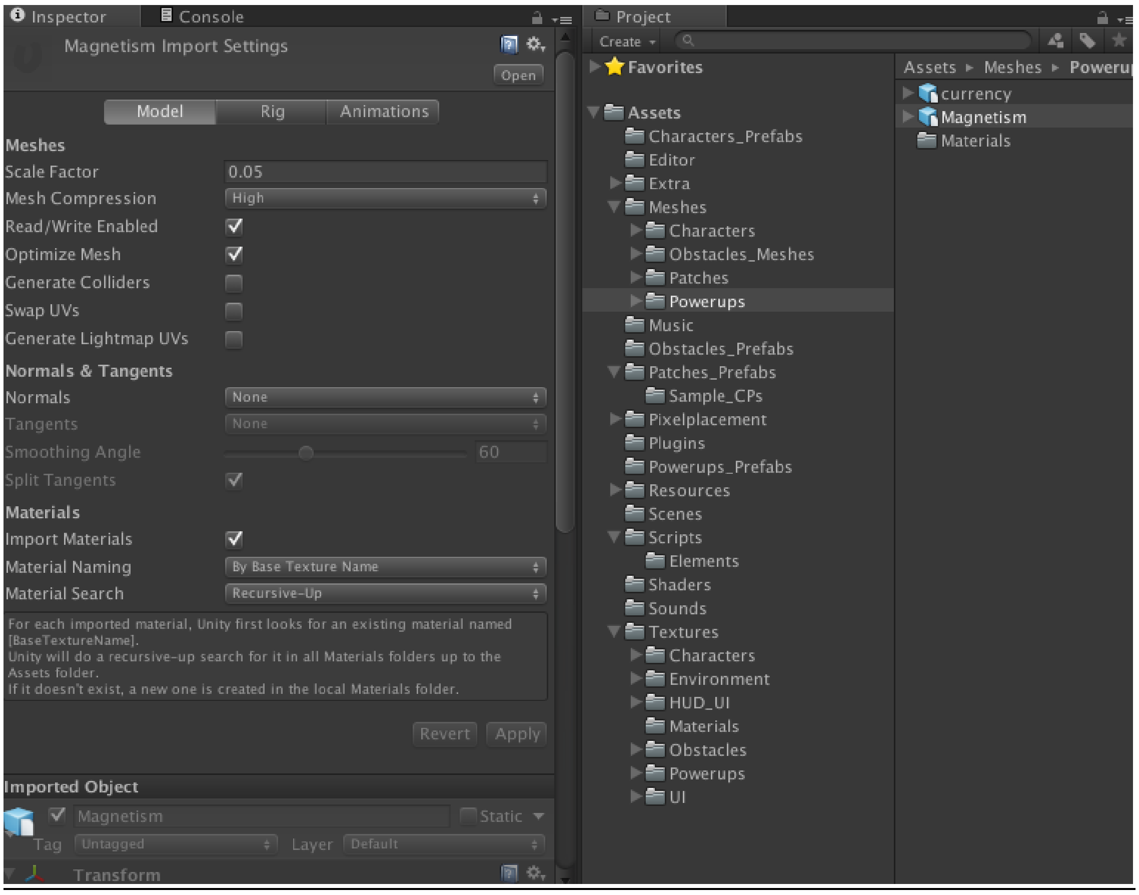
Strafe Left: Swipe left, A Key or Left Arrow Key

Strafe Right: Swipe right, D Key or Right Arrow Key

Duck: Swipe down, S Key or Down Arrow Key

Adding a Power-up

1. Import the .fbx of the asset that you will use as an obstacle.
   Optimization Notes:
   - Use Model > Scale Factor to adjust the scale.
   - Use Model > Mesh Compression to compress the imported asset as much as you can without compromising its appearance.
   - Select "None" in Rig > Animation Type if your asset isn't rigged.
   - Uncheck the Import Animation option in Animations > Import Animation if your asset isn't animated.



Meshes

Scale Factor	0.05
Mesh Compression	High
Read/Write Enabled	✓
Optimize Mesh	✓
Generate Colliders	
Swap UVs	
Generate Lightmap UVs	

Normals & Tangents

Normals	None
Tangents	None
Smoothing Angle	60
Split Tangents	✓

Materials

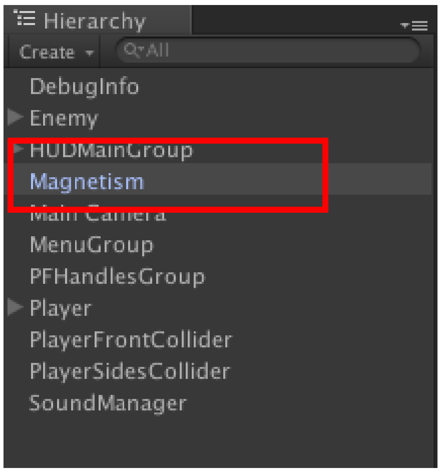
Import Materials	✓
Material Naming	By Base Texture Name
Material Search	Recursive-Up

For each imported material, Unity first looks for an existing material named [BaseTextureName].
Unity will do a recursive-up search for it in all Materials folders up to the Assets folder.
If it doesn't exist, a new one is created in the local Materials folder.

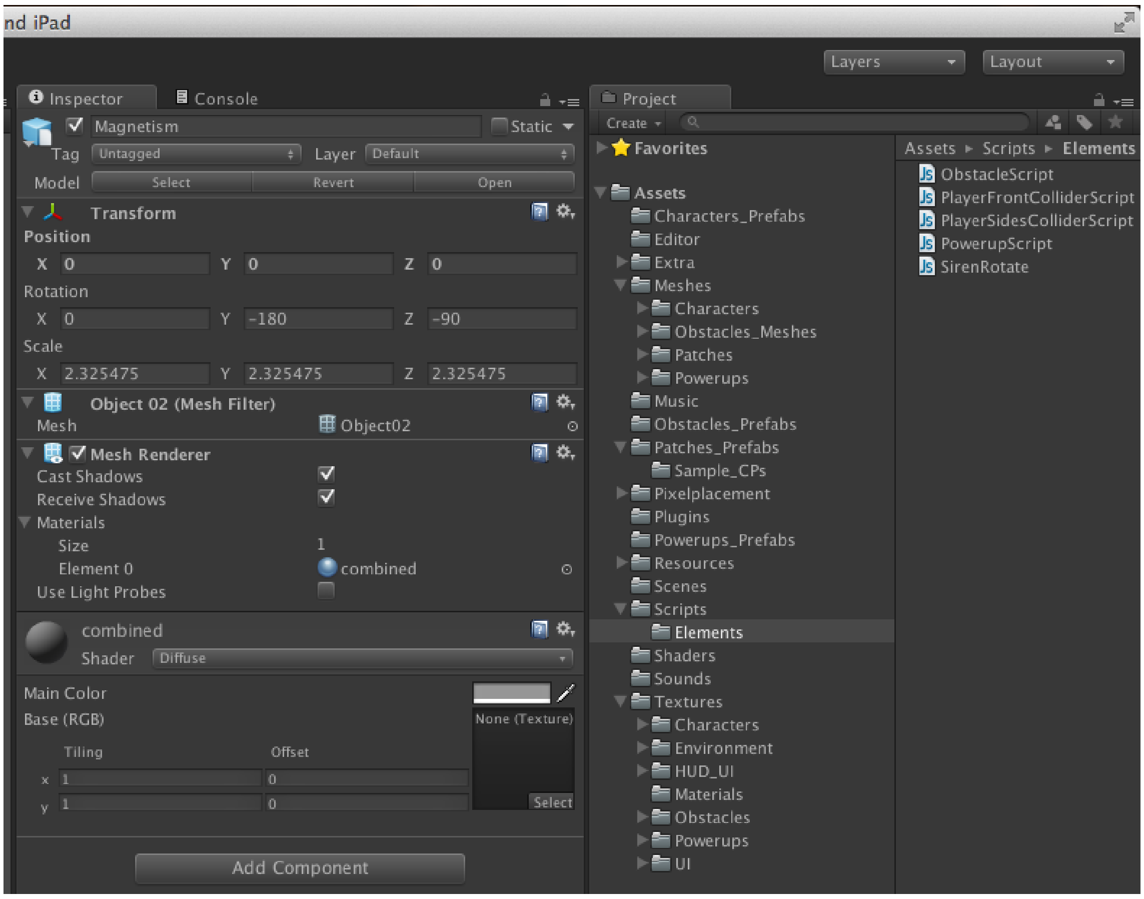
Imported Object

Magnetism	Static
Tag Untagged	Layer Default

2. Drag the imported asset in "Hierarchy" tab to create a prefab.



3. Go to Project > Assets > Scripts > Elements.



Magnetism
Tag Untagged Layer Default
Model Select Revert Open

Transform
Position X 0 Y 0 Z 0
Rotation X 0 Y -180 Z -90
Scale X 2.325475 Y 2.325475 Z 2.325475

Object 02 (Mesh Filter)
Mesh Object02

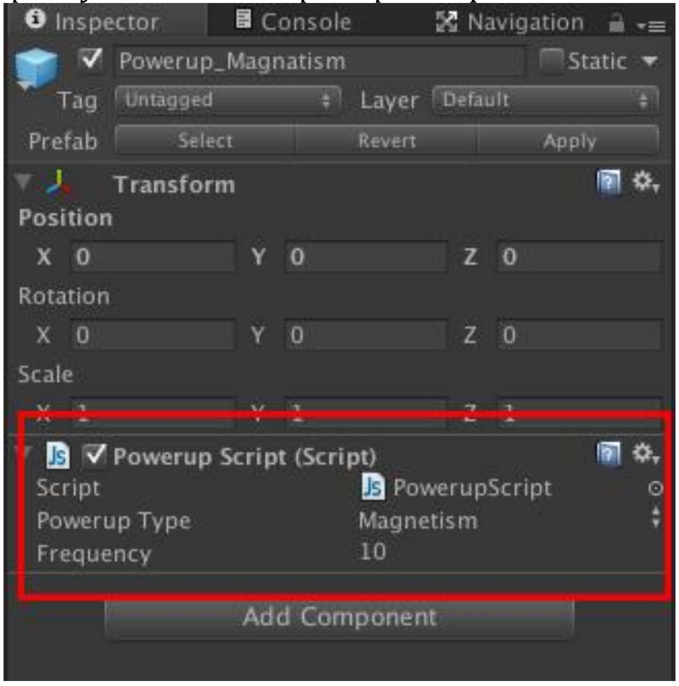
Mesh Renderer
Cast Shadows ✓
Receive Shadows ✓
Materials
Size 1
Element 0 combined
Use Light Probes

combined
Shader Diffuse
Main Color
Base (RGB) None (Texture)
Tiling Offset
x 1 0
y 1 0

Assets
- Characters_Prefabs
- Editor
- Extra
- Meshes
  - Characters
  - Obstacles_Meshes
  - Patches
  - Powerups
- Music
- Obstacles_Prefabs
- Patches_Prefabs
  - Sample_CPs
- Pixelplacement
- Plugins
- Powerups_Prefabs
- Resources
- Scenes
- Scripts
  - Elements
- Shaders
- Sounds
- Textures
  - Characters
  - Environment
  - HUD_UI
  - Materials
  - Obstacles
  - Powerups
  - UI

Assets ▸ Scripts ▸ Elements
- ObstacleScript
- PlayerFrontColliderScript
- PlayerSidesColliderScript
- PowerupScript
- SirenRotate

4. Add the "PowerupScript.js" on your power-up's prefab.
5. Set the *Powerup Type* on the *Powerup Script* component.
6. Set the *Frequency* on the *Powerup Script* component.

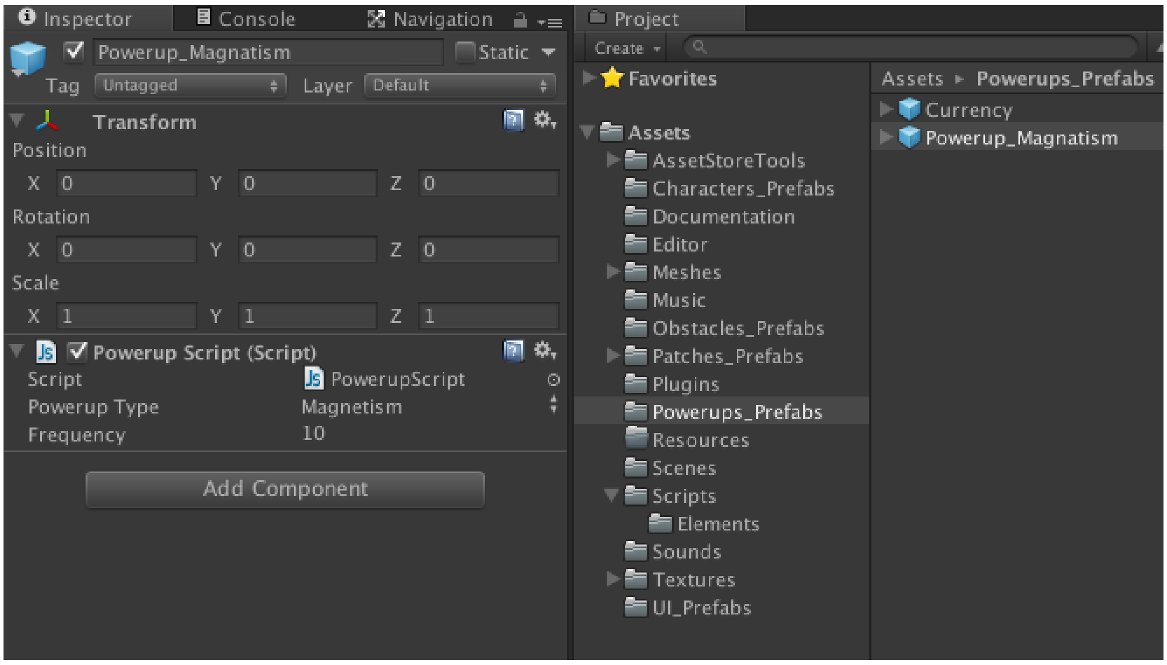


Powerup_Magnatism
Tag Untagged Layer Default
Prefab Select Revert Apply

Transform
Position X 0 Y 0 Z 0
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1

Powerup Script (Script)
Script PowerupScript
Powerup Type Magnetism
Frequency 10

7. Drag and drop your newly created power-up prefab to Project > Powerups_Prefabs.

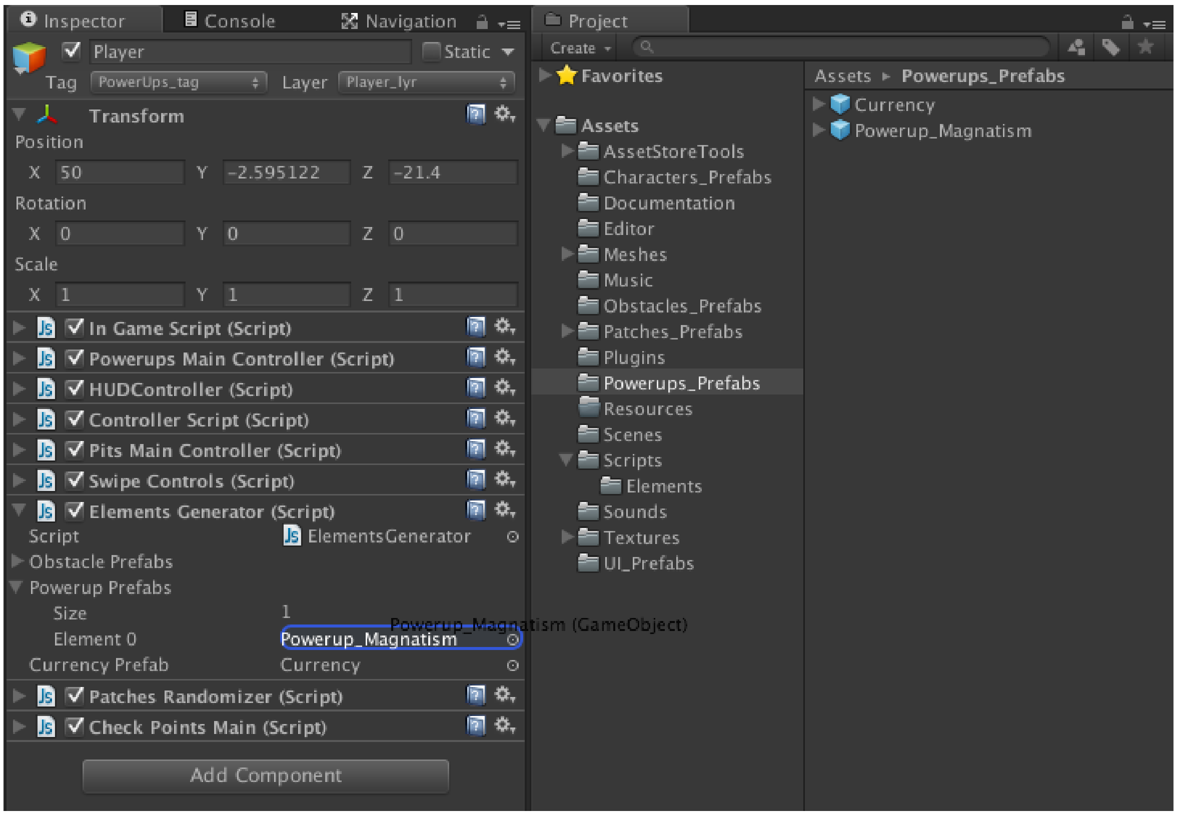

Powerup_Magnatism
Tag Untagged Layer Default

Transform
Position X 0 Y 0 Z 0
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1

Powerup Script (Script)
Script PowerupScript
Powerup Type Magnetism
Frequency 10

Assets ▸ Powerups_Prefabs
Currency
Powerup_Magnatism

8. New, we have to tell the *ElementGenerator* script about our newly created power-up so that it will generate the power-up on the path during gameplay. To do that, simply drag and drop the saved prefab to the *ElementGenerator.js* script that's a part of the *Player* prefab. The prefab has to drop in the *Powerup Prefabs* hierarchy to take effect.


Player
Tag PowerUps_tag Layer Player_lyr

Transform
Position X 50 Y -2.595122 Z -21.4
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1

In Game Script (Script)
Powerups Main Controller (Script)
HUDController (Script)
Controller Script (Script)
Pits Main Controller (Script)
Swipe Controls (Script)
Elements Generator (Script)
Script ElementsGenerator
Obstacle Prefabs
Powerup Prefabs
Size 1
Element 0 Powerup_Magnatism
Currency Prefab Currency
Patches Randomizer (Script)
Check Points Main (Script)

Assets ▸ Powerups_Prefabs
Currency
Powerup_Magnatism

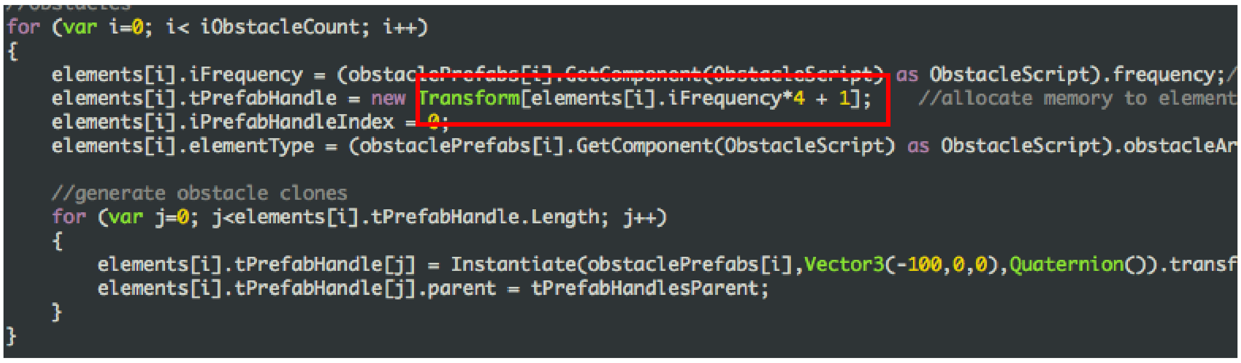
9. Finally delete the power-up prefab from the Hierarchy tab. It does not belong in the scene by default because it is generated programmatically during runtime.

Frequently Asked Questions

Obstacles and/ or currency units disappear whenever the *Player* gets to a new patch.

The *ElementsGenerator* script instantiates and pools all the elements (currency, obstacles and power-ups) when the game starts. When a new patch is generated, elements from these pools are placed at the needed locations. Sometimes when a very few obstacles are used, the elements' clones pooled are not enough for placement in multiple patches. In this case, whenever a new patch is generated, obstacles currently in front of the player are relocated to the next patch making them disappear suddenly from the screen.

To fix this, all you need to do is to increase the number of clones in the pool. The following screen highlights the code in the *setPrefabHandlers* function in the *ElementGenerator* script where the number of obstacles to pool is calculated. Lets say the frequency of an element was set to 10; the clones pooled will be 41. The number '4' can be replaced with a higher number to generate more elements.



```
for (var i=0; i< iObstacleCount; i++)
{
    elements[i].iFrequency = (obstaclePrefabs[i].GetComponent(ObstacleScript) as ObstacleScript).frequency;/
    elements[i].tPrefabHandle = new Transform[elements[i].iFrequency*4 + 1];    //allocate memory to element
    elements[i].iPrefabHandleIndex = 0;
    elements[i].elementType = (obstaclePrefabs[i].GetComponent(ObstacleScript) as ObstacleScript).obstacleAr

    //generate obstacle clones
    for (var j=0; j<elements[i].tPrefabHandle.Length; j++)
    {
        elements[i].tPrefabHandle[j] = Instantiate(obstaclePrefabs[i],Vector3(-100,0,0),Quaternion()).transf
        elements[i].tPrefabHandle[j].parent = tPrefabHandlesParent;
    }
}
```

How do I change the length of the patches to a value other than the default (3000 meters)?

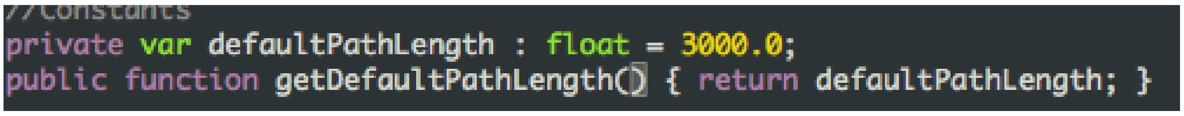
The patches used in the project are 3000 meters by default.

1. To change their length, the first step is to create new patches of the required length in a 3D modeling tool.

2. The second step is to set the checkpoints as explained in the "Adding a Patch" tutorial.

3. Once you have the patches, you also need to tell the *CheckPointsScript* the size of the patches to generate. If you skip this step, the patches will be generated after the default (3000 meter) distance.

Change the *defaultPathLength* variable in *CheckPointsScript* to match the size of your new patches.



```
//Constants
private var defaultPathLength : float = 3000.0;
public function getDefaultPathLength() { return defaultPathLength; }
```