***Introduction to Graphics and Mobile Gaming***

**Unity LAB 3: Part 2**

**Shadows**

**Issue 1.0**

Contents

[1 Introduction 1](#_Toc36120143)

[2 Static shadows 1](#_Toc36120144)

[2.1 Rendering the cube map 1](#_Toc36120145)

[2.2 Applying custom shaders 5](#_Toc36120146)

[3 Dynamic shadows 5](#_Toc36120147)

[4 GUI elements 7](#_Toc36120148)

[4.1 Changing the shadows dynamically 7](#_Toc36120149)

[5 Scripts 8](#_Toc36120150)

[6 Installing the app 8](#_Toc36120151)

# Introduction

In this lab, we will be continuing our work with shaders, putting the previous labs’ theory into practice.

Load the previous lab up in Unity, and let’s get started.

The first thing we want to do is add a folder and re-name it “shaders”.

While in the new shaders folder, import the following shaders: (right-click and then select import new asset and select the provided ones from this lab folder)

* ***ChessPiece.shader***
* ***RoomShadows.shader***
* ***Common.cginc***
* ***ShadowMap.shader***

The first two shaders will be used for the chess pieces, and for other meshes that are placed in the room, the third shader is a file used by the first 2 to apply local correction when rendering.

After having imported them, double-click on the first two and open them in ***Visual Studio*** (which Unity uses by default), go through the code and see how it works.

# Static shadows

## Rendering the cube map

Now, there is a list of things that we will have to do to implement the custom shaders correctly:

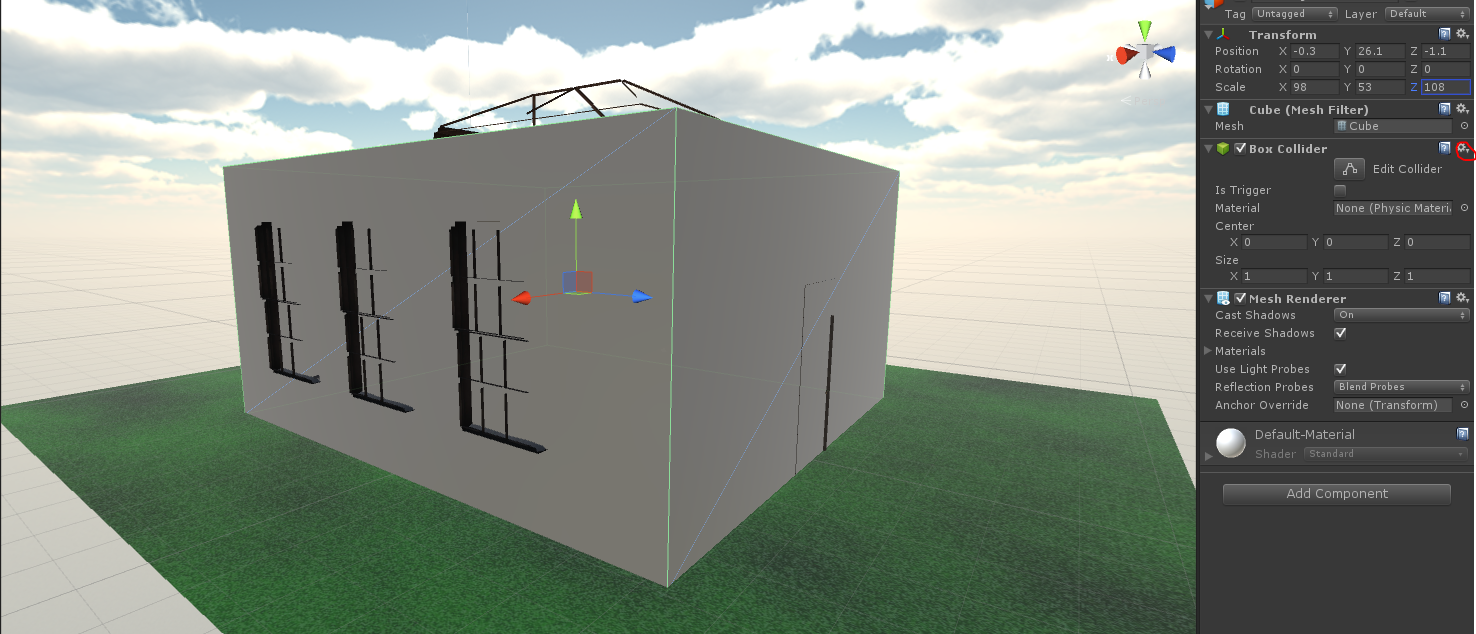
1. In the assets folder->new folder-> name it ***Editor*** and import *GenerateStaticCubemap.cs* from the assets/editor folder.

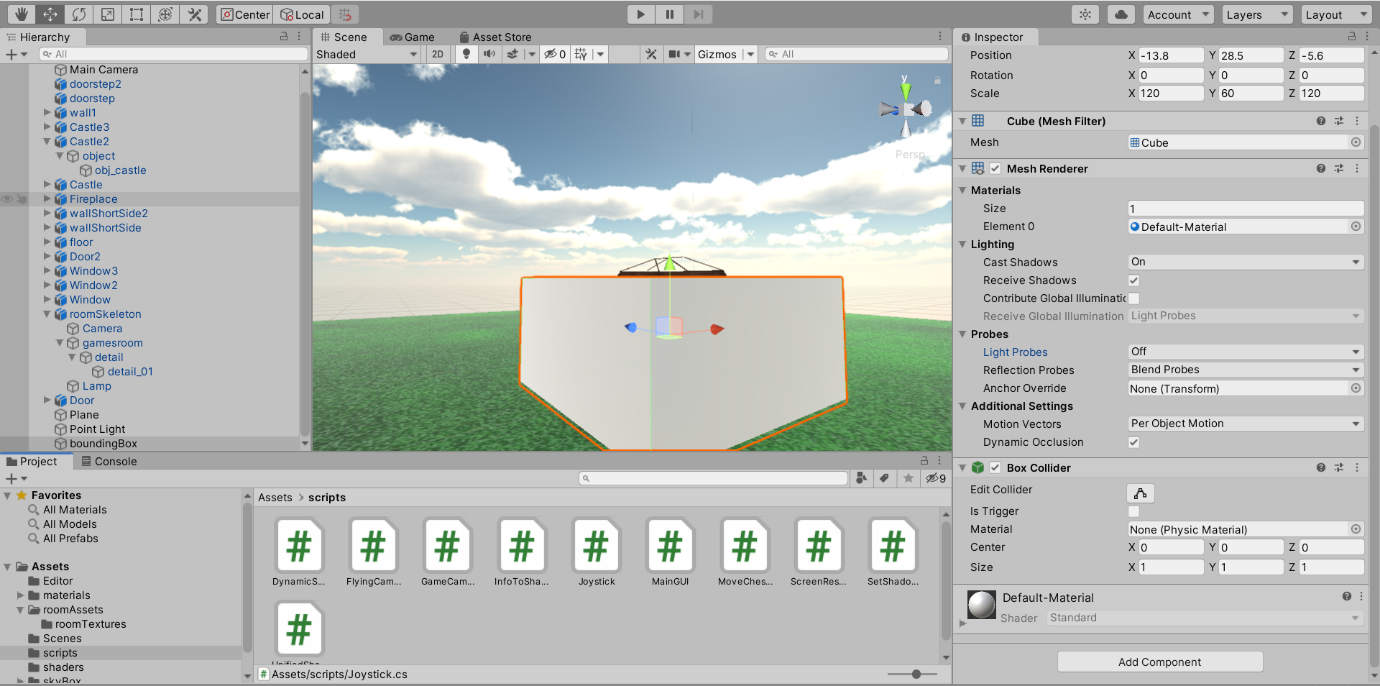
This will add the **Tools** option tab on top of the scene window (next to GameObject and Component)

1. Then, go to Assets, then new folder and name it “***scripts****”* ->import all scripts by opening the folder in the explorer, then copying the scripts from the provided lab folder.
2. Then, we add a **point light** (GameObject->Lights) and place it in front of the windows (be sure to delete all present types of lights first). We will add all lighting to the scene at a later time, so we’ll ignore how dark the room looks for now.
3. Next, we need to **create dynamic and static layers**:
   * To do so, double-click on the door (or any other wall), then inspector, then layer, then click *default,* then add layer, and then first.
   * Then, select a free slot
   * Add a *staticObj*
   * Add a *dynObj*
   * For all objects that do not move, we will apply the staticObj layer, for others, we use dynObj
   * Now, select all static objects and apply a static layer to them from the inspector panel changing
   * Do the same for dynamic objects as well (chess pieces are dynamic because they could move position)

You can select more than one object using shift and control in the Hierarchy panel.

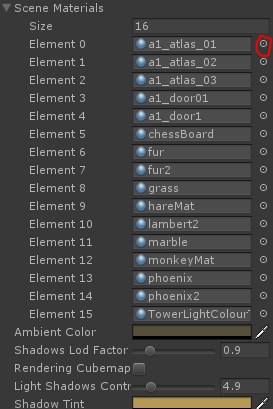
1. Create a **cube**: go to GameObject > then 3D Object > then cube
   * Re-name it **“*boundingBox”***
   * Set the scale so that it surrounds the room and move it to encase the room structure as we have exampled in the image below.
   * The plane scale has changed from 100 -> 30.





* Change the “light probes” from Blend proves to off in the inspector
* Box collider, click on the small gear symbol and then click remove component
* Untick mesh renderer, this way, you won’t see the box anymore, but it’s still there.

1. Create an **empty object**: select *GameObject* and then *Create Empty*
   * Rename it **“*unifiedShaderController.”***
   * Open the scripts folder from the Project tab and then click on the object we just created and drag the *unifiedShaderController* script into the Inspector tab.
   * In inspector, click on the scene materials arrow.
   * Set the size to 16; this will allow us to add materials to each scene. These are all the materials that can appear in the scene.
   * On the right of each element, click on the circle and select the material, and make sure we add all materials for walls, doors, chessboard, ceiling, floor, doorsteps, and chess pieces.

You might notice you can’t find some of them because we have not used them yet. Create a new folder *‘advanced’* in your Assets folder and copy-paste all the contents from the Assets>advanced in the FileExplorer.

* + When adding assets in the future, they may have come out black; this is because the material hasn’t been set here.
  + Ambient colour set to a light yellow (e.g., a RGBA of 84-78-59-0).
  + Set the Shadows Lod factor to 2.5.
  + Shadows control = 4.9.
  + Shadow tint to white (e.g., a RGBA of 182-154-84-0).

1. Create an **empty object**: select GameObject and then create empty
   * Rename it “***infoToShaders”***
   * Open the scripts folder from the Project tab and then click on the object we just created and drag the *InfoToShaders* script into the Inspector tab.
   * In the Inspector tab, click on the Shadow Mats arrow and set the number to 16 again; then, proceed to load all elements as before.
   * To add the Shadow light, add the point light by dragging and dropping to the inspector from the Hierarchy panel.
   * For the Room B Box, drag the *boundingBox* objectin as done for the point light.
2. Next, we are going to create a **Cubemap**:

* Go to our Room folder and then to our roomTextures folder.
* Right-click on the textures folder and select create and then legacy and finally cube map.
* Rename the new object “*cube map”*
* In the inspector panel for the cube map, set the face size to 512, set the mipmaps to yes, untick linear, and finally, tick readable.

1. Next, we want to **render** our scene to our cubemap to allow our lighting to be based on a local cubemap:
   * Go to Tools and then select Renderinto Cubemap
   * In the window that pops up, select Render position and drag the bounding box object from Hierarchy panel
   * Under Cubemap, drag the cubemap object from the Project panel (room textures folder)
   * Under Camera layer mask, all unticked apart from staticObj (easy to click *nothing* and then staticObj)
   * Press Play and then select *unifiedShaderController* from hierarchy panel
   * Tick *rendering cubemap* in the Inspector tab for this element
   * Click “Render!” in “Render to cubemap” window that’s just been filled in
   * Our cubemap has now been rendered!
   * Press again on Play button to stop the rendering process
   * If there is an object in the scene that does not display properly, ensure that the object’s shader has been set to Mobile bump/diffuse. You might need to render the cubemap again if you forgot to set any of the shaders.
   * Open the cubemap, which should be in your folder, and expand the tab on the right; next to “cube map” at the top of the panel, click on RGB stripes. This is showing the alpha channel of the room as seen from above.

## Applying custom shaders

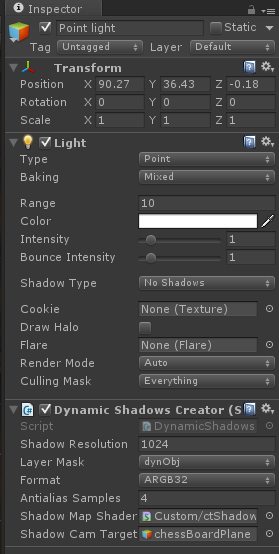
Now we need to go through our meshes one by one and change the material shader from mobile to custom/roomShadows, and for the chess pieces, use custom/chessPiece.

* A screenshot of a cell phone

  Description automatically generatedPass the cubemap to the Static Shadow Map section
* Set the Static Shadow Factor to 0.5
* Set the Static Shadow Angle Threshold to −0.01
* Try to move the point light to see the shadows better in the room

# Dynamic shadows

Now we need to enable our dynamic objects to have shadows as well; this will be for our in-game interactable objects like our chess pieces.



* Attachthe *dynamicShadowCreator* script to PointLight by dragging it from the Project panel to the Inspector of the point light.
* This will attach the camera to the light and create a run time shadow camera at the light position.
* Set the Layer mask to *dynObj*.
* In the Shadow map shader field for this script, drag the *ShadowMap* shader from the shaders folder.
* In the camera target field, drag the *chessBoardPlane* from the Hierarchy panel (make sure that all chess pieces have their layer set to *dynObj*).
* We need to set the Chess board and Chess room to receive dynamic shadows
  + Add the *SetShadows* scripts to the roomSkeleton, doors, fireplace, and chess board by dragging the script to the corresponding Inspector panels.
  + Dragthe point light to the Shadows Camera section of the added *setShadows script.*

The chess board inspector panel will look as follows.

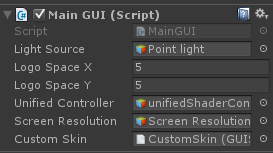
If we encounter any problems, make sure only the chess pieces have their layer set to *dynObj*.

Click Play. The first thing we should now notice is that all chess pieces project shadows. Moreover, if we look in the Hierarchy panel, a new *ShadowCamera* should have appeared. Select the chess board; in the inspector panel under the shader section, we should see that the *Dyn runtime shadows* section has been populated. Double-click on the icon with the eye/mountain; a new alpha channel map should appear showing us all the elements that produce dynamic shadows!

# GUI elements

Next, we will develop 3 GUI elements. The first is to enable the user to change the shadows settings, the second is to enable the user to move the chess pieces, and the last is to enable the user to move around the room.

## Changing the shadows dynamically

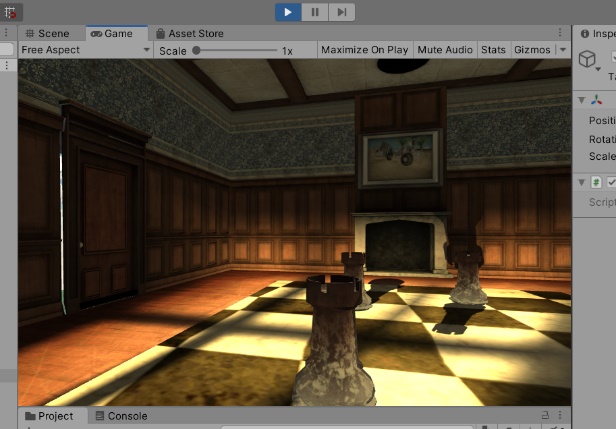
1. The first thing to do is create a new **empty object**, rename it *GUIElements, and check its position is 0, 0, 0*.
2. Repeat this again, renaming the new object *MainGUI*, and from the Hierarchy panel, drag it and drop it on the top of GUI Elements (this will make it a child of GUI Elements).
3. Create another empty object and rename it *ScreenResolution* and make it a child of GUIElements as well. Then, we click on the Screen resolution element from the Hierarchy panel, and in the inspector tab, drag and drop the *Screen resolution* *script* from the scripts folder.
4. Now we need to add the MainGUI script as well:

* Click on the MainGUI element on the Hierarchy panel, and in the Inspector tab, drag and drop the *MainGUI script* from the scripts folder.
* Fill in the LightSource field by dragging the point light element from the Hierarchy panel and then check that both LogoX and LogoY are set to 5.
* Fill the Unified controller field by dragging the *unifiedShaderController* element from the Hierarchy panel.
* Fill the Screen Resolutionfield by dragging the ScreenResolution element from the hierarchy panel.

1. Now create a **new folder** for the custom skin field and rename it ***GUI***:
   * Import a new asset *CustomSkin.guiskin* from the Assets folder
   * Fill the Custom Skin field by dragging the *CustomSkin asset*

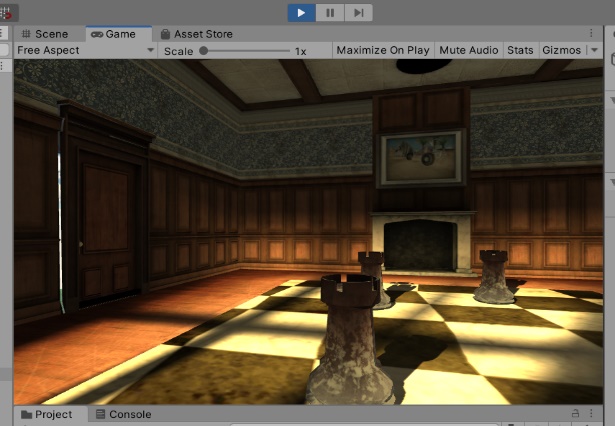
# Scripts

In this lab, we used a number of scripts; please take the time to open them and understand how each of them work. In addition, when running your application in editor windows, your shadows will be mirrored



To fix it, go to Shaders > RoomShaders.shader > comment the following line:

output.shadowsVertexInScreenCoords.y = output.shadowsVertexInScreenCoords.w - output.shadowsVertexInScreenCoords.y;



# Installing the app

Once done, connect your phone just as previously and click on **Build Settings -> Build and Run**. Make sure the platform you’re using is Android, that you have uncommented the line above and that the company name and product name are set in the Player Settings.

Try using the two sliders on the sides of your phone screen. You can adjust the direction and intensity of shadows. The shadows will move left or right, but not up and down. If we want to change the height of the shadows, do so by moving the Point Light in the scene.