Creative NFT-Copyrighted AR Face Mask Authoring Using Unity3D Editor (Supplementary Material)

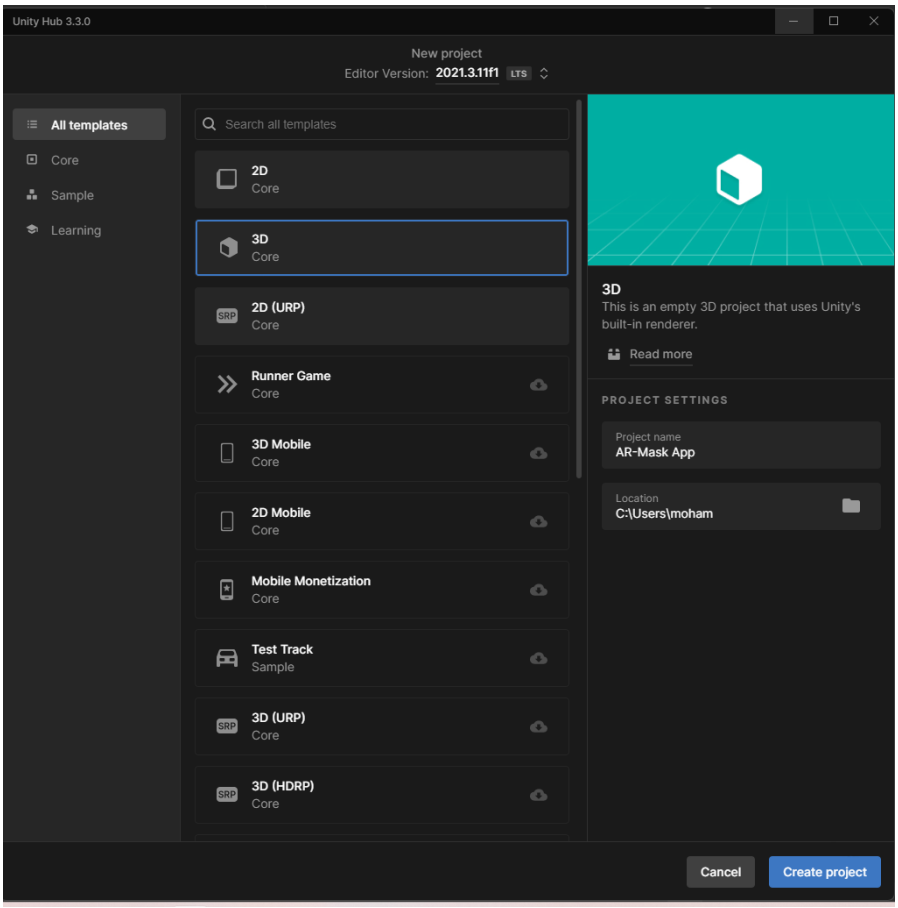
Mohamed Al Hamzy*†, Shijin Zhang$†, Hong Huang*, Wanwan Li*
University of South Florida* Columbia University$
† marked as joint first authors

What was needed: Must run on Unity Editor 2021.X (Newer versions are recommended to avoid gradle errors have not tested earlier versions), install guide https://www.youtube.com/watch?v=sbFgscbE3Lw

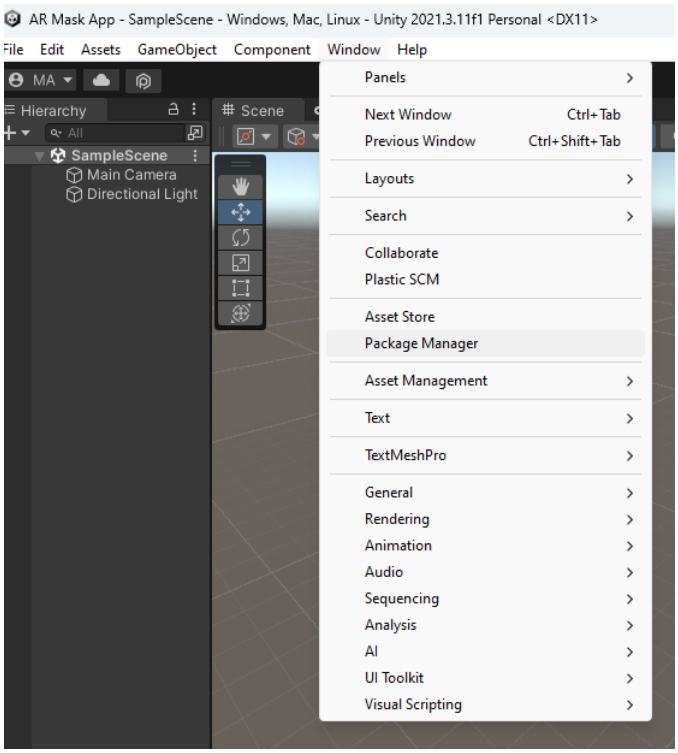
Android Phone with AR functionality, check https://developers.google.com/ar/devices.

-Building on IOS instead needs appropriate compatibility-

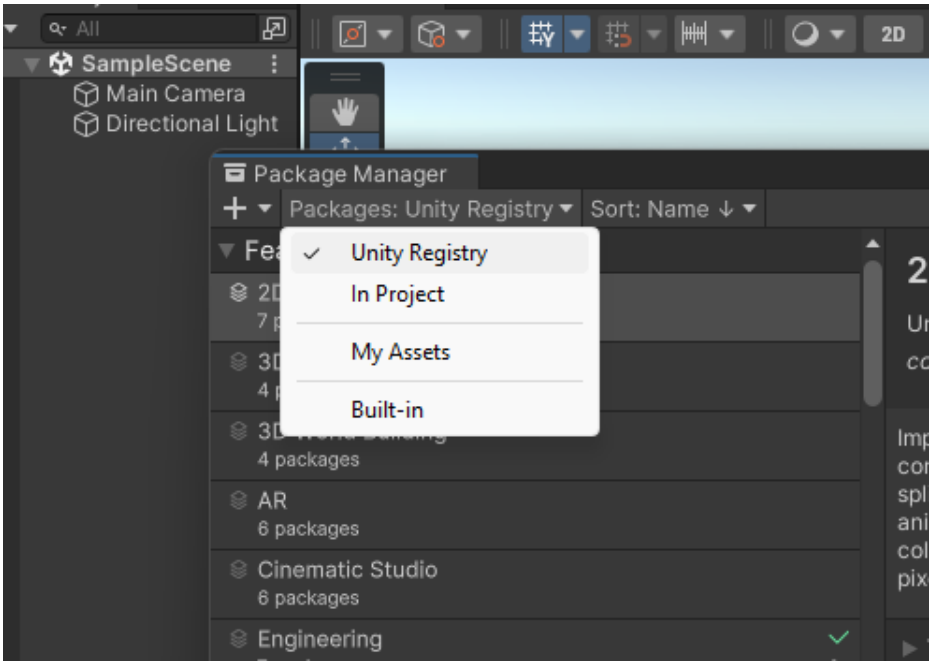
Instructions to replicate: *Follow requirements for what was needed before starting*



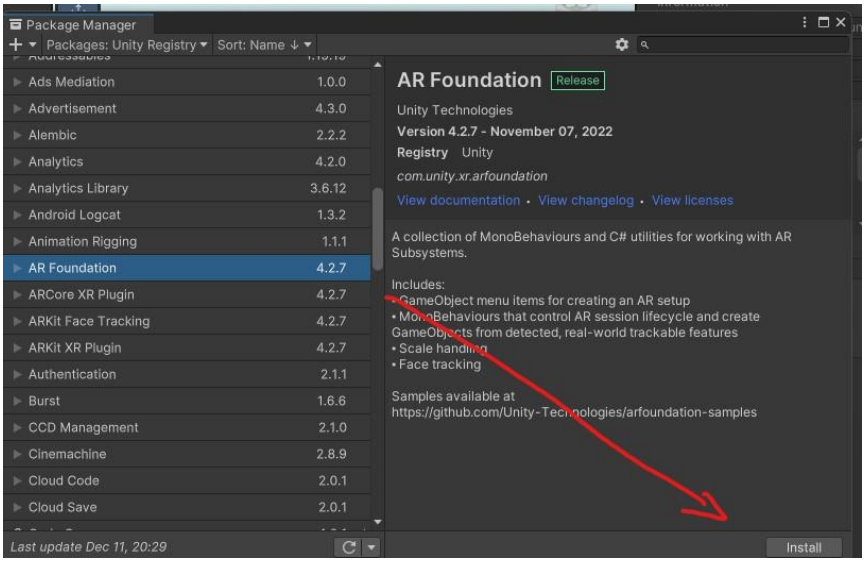
Create a project in AR foundation



Go to Window->Package Manager

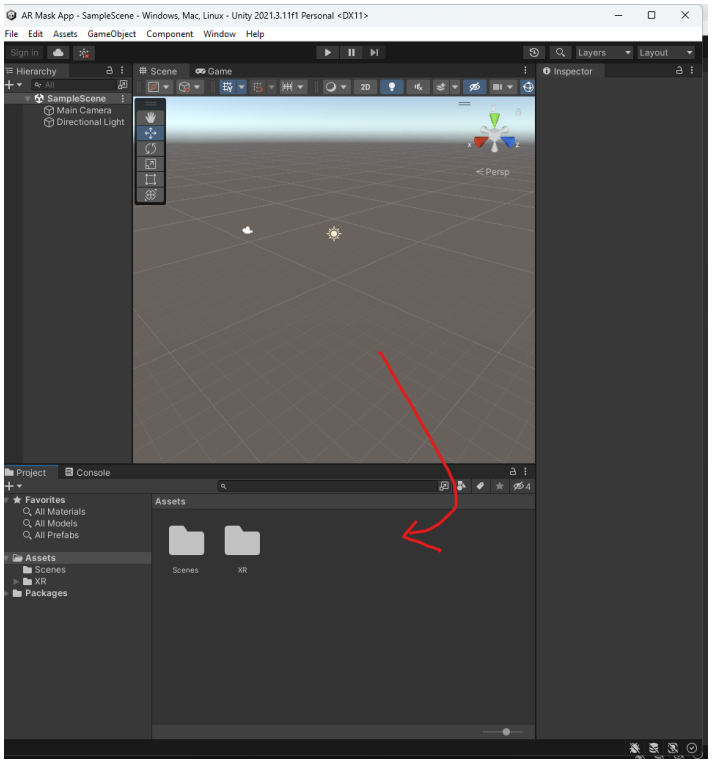


Click the dropdown menu next to the “+” button, to show all packages Unity Registry

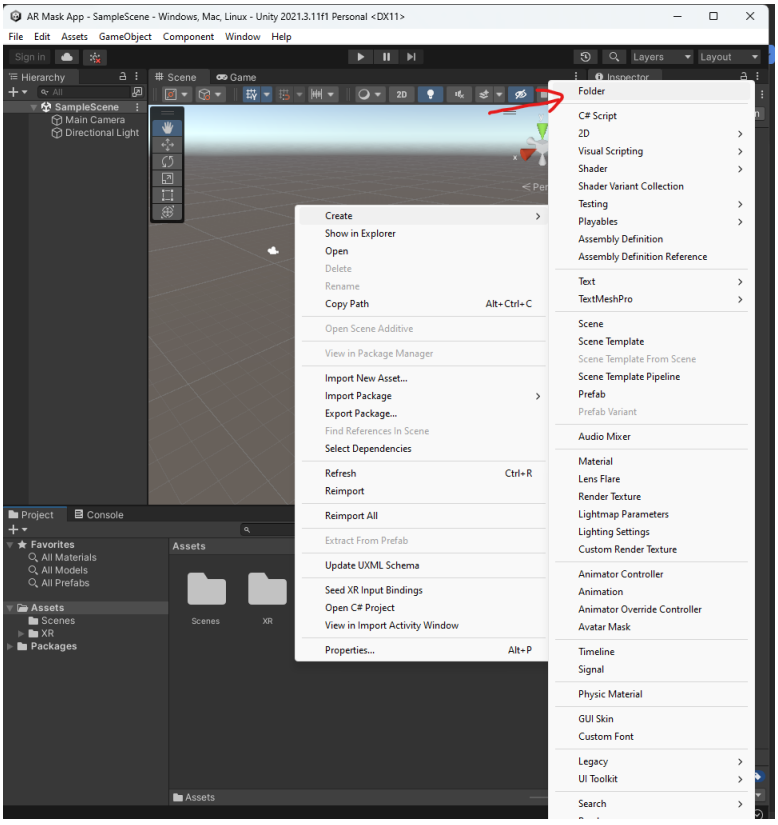


Find AR foundation and install

Repeat the last step for: ARCore XR Plugin

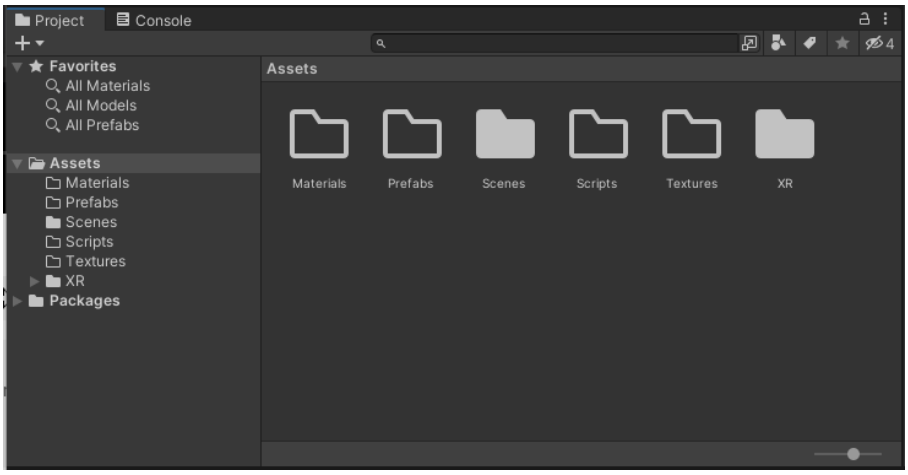


Find the assets folder

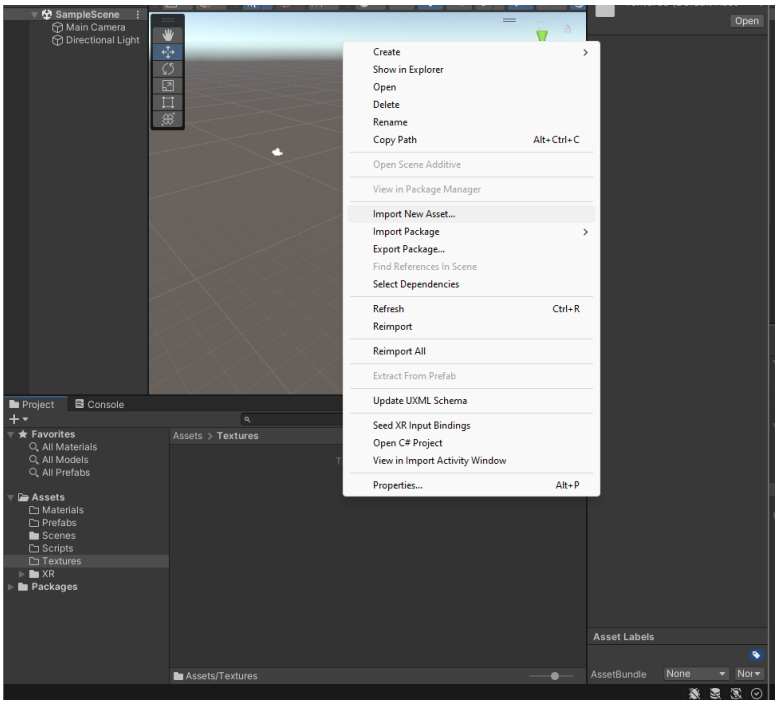


Right click on the folder and create a folder inside it called “Prefabs”

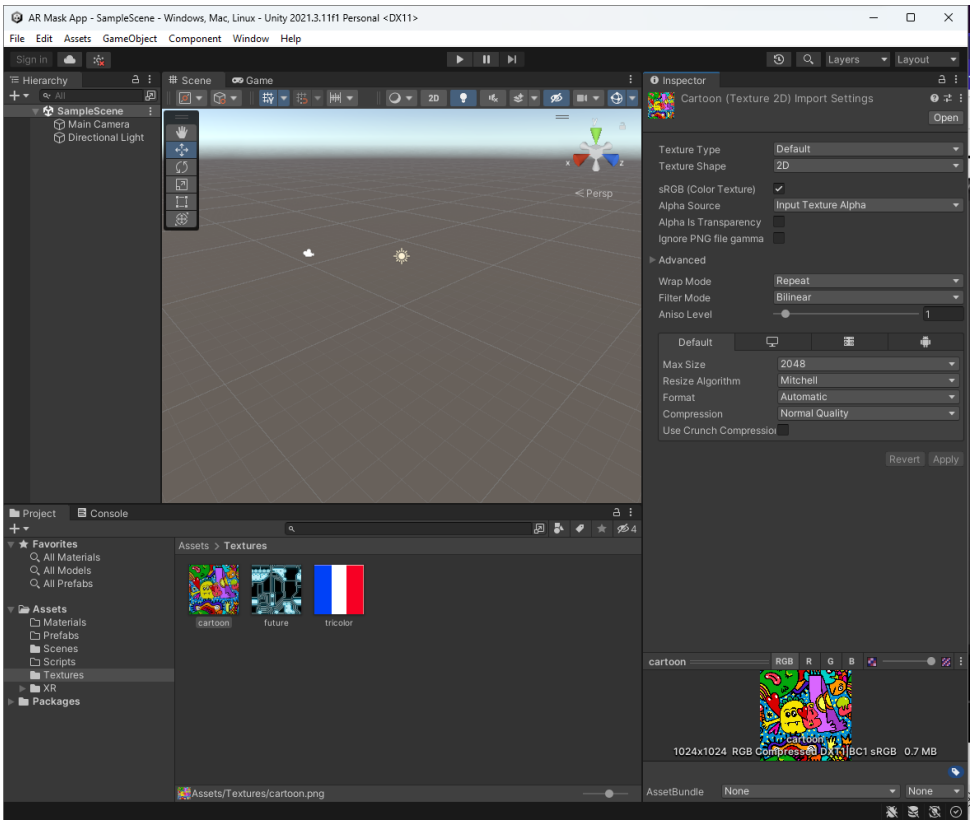
Repeat the last step and create three other folders called: “Materials”, “Textures” and “Scripts”



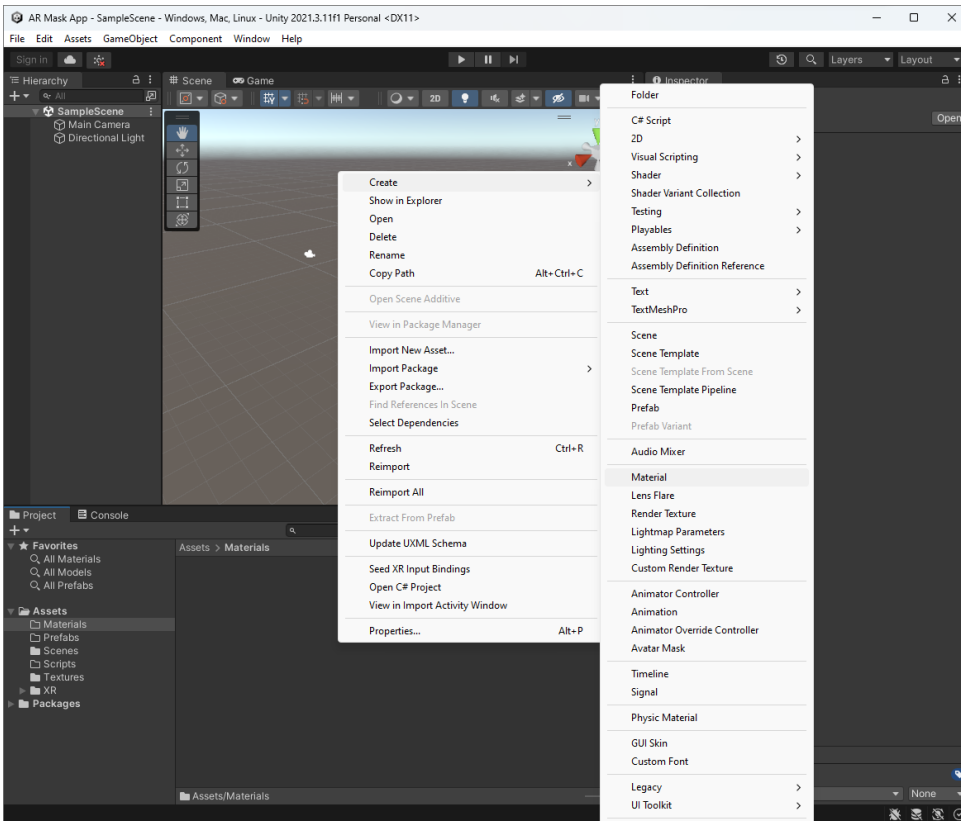
Your Assets folder should look like the snippet above now



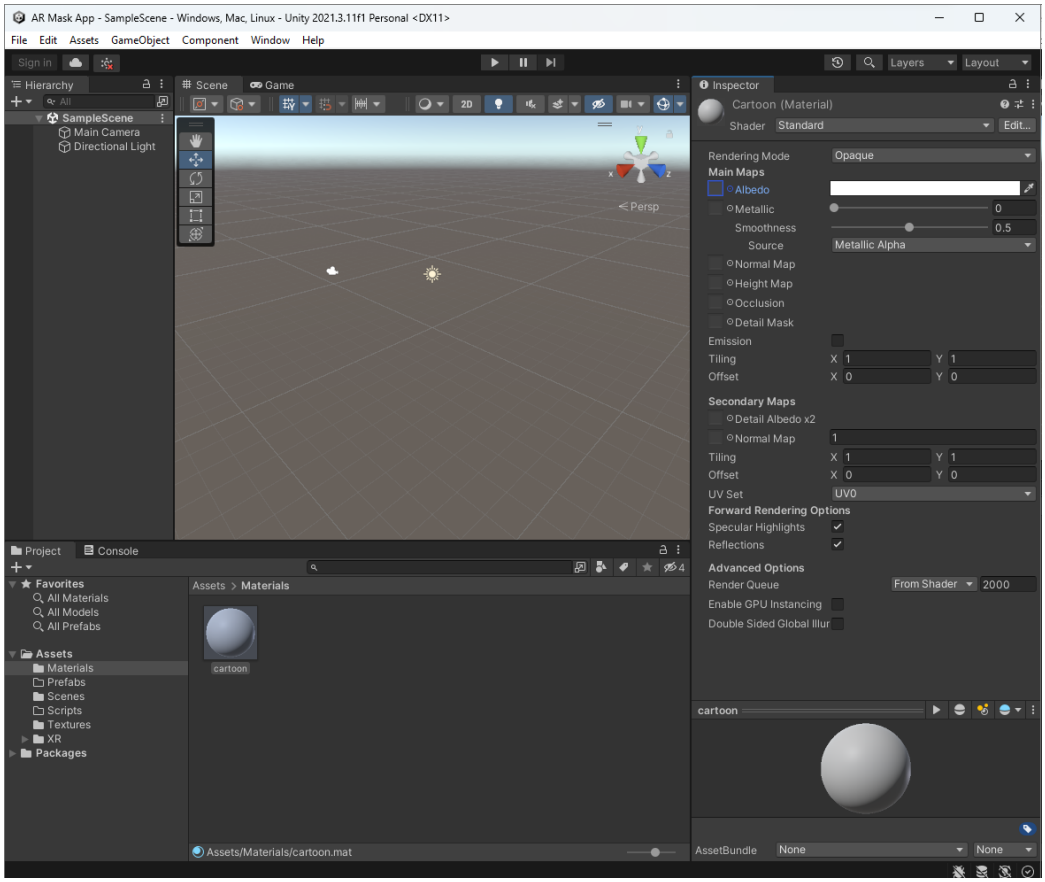
Head into the textures folder to import new textures. You can do this by dragging and dropping PNG textures to the folder. Or right click and choose to import new assert. Find the textures you want to upload, as long as they are in the compatible PNG format.



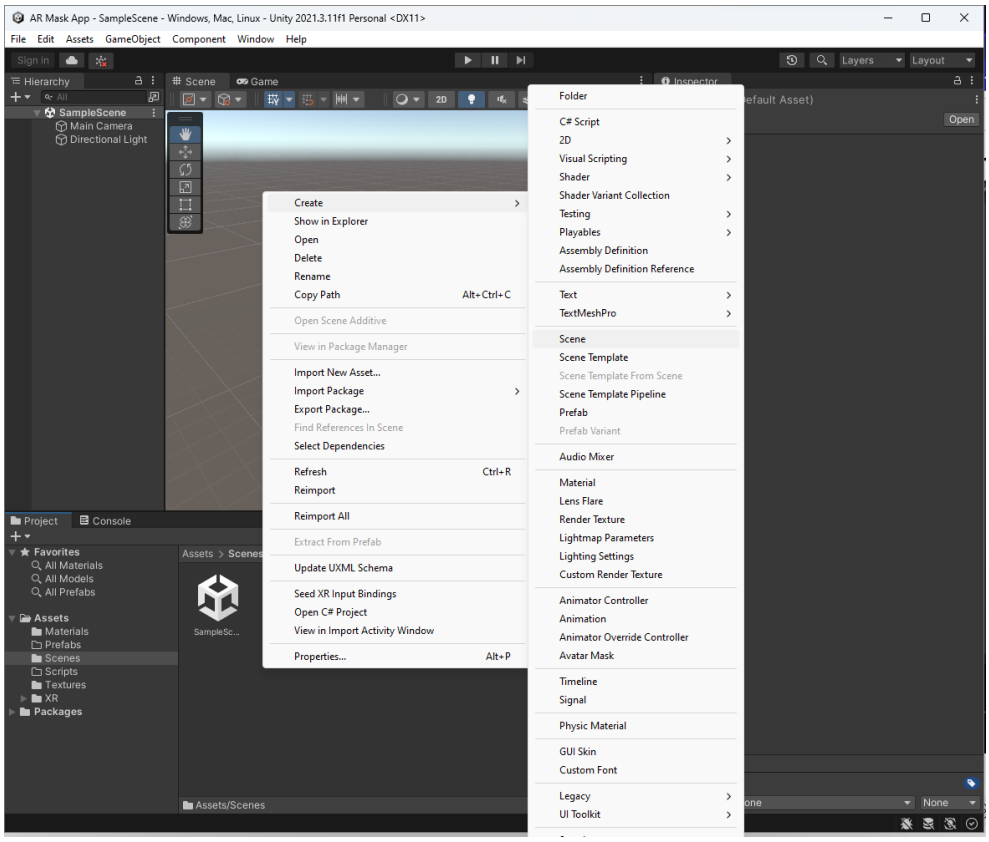
For any textures that have hollow or transparent parts (where the mask should be invisible). Left click on the texture and make sure you check the box that says Alpha is Transparency in the inspector tab. Click Apply on the bottom.



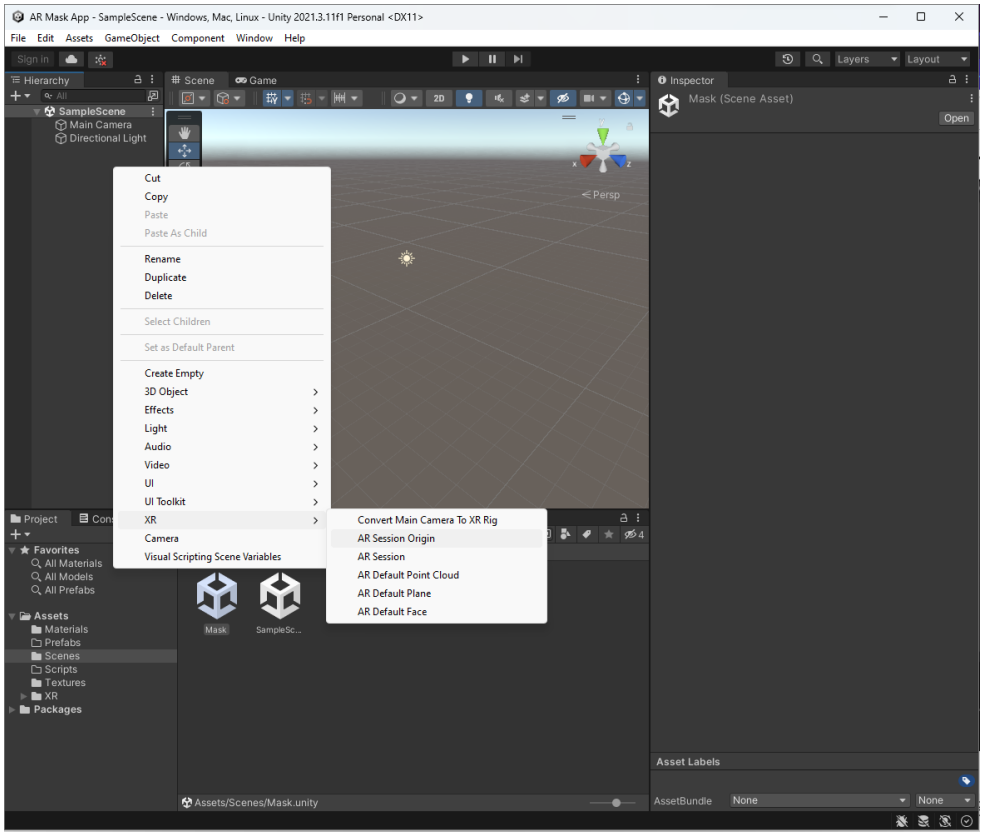
Head into the Materials folder and create a corresponding material for each texture. This step is necessary to be able to fit the textures into the masks.



We need to associate each material to its texture. Left click the material and click the circle next to Albedo in the inspector tab. This will let you find the texture in the assets, make sure you choose the correct one.

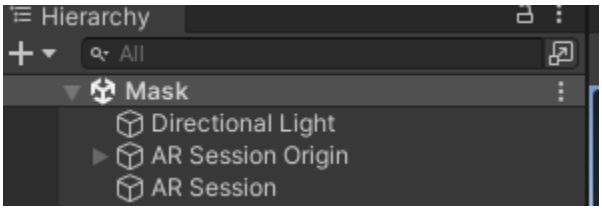


Now head to the scenes folder and create a new scene.



Right click the hierarchy and choose XR->AR session origin.

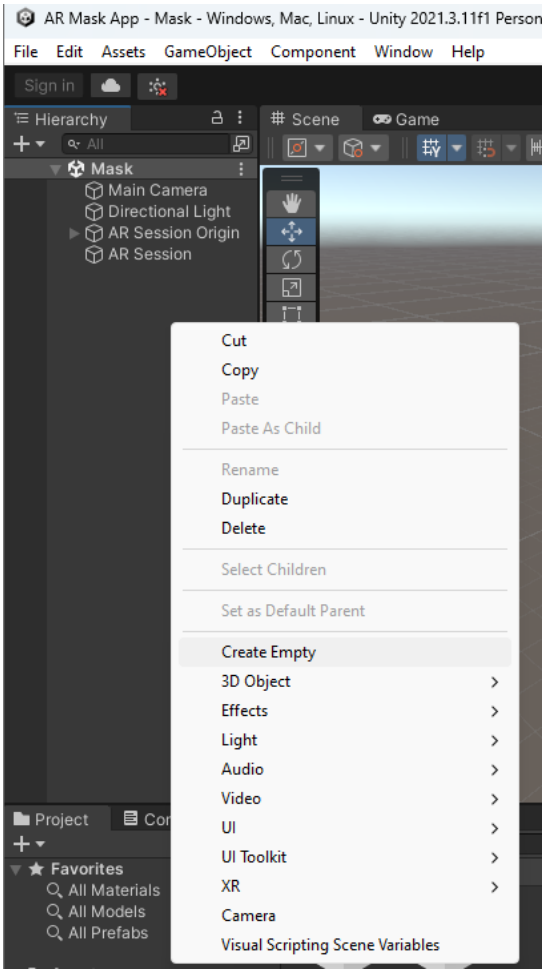
Repeat the same step and choose XR->AR session.



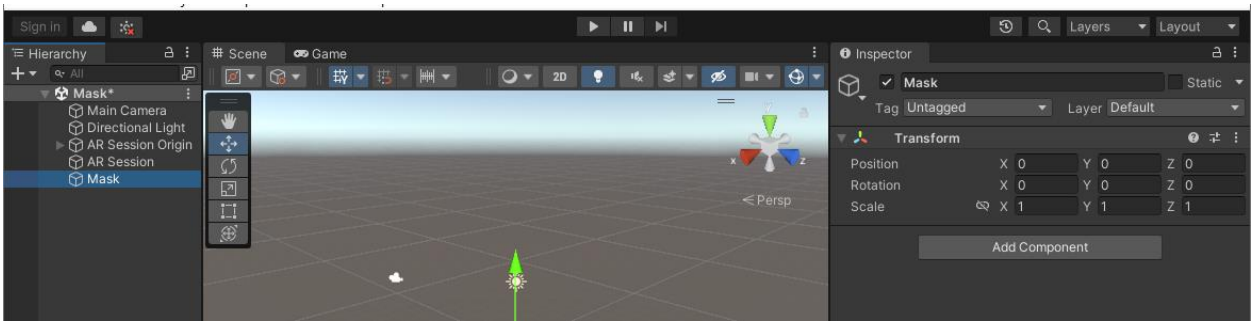
They should show up on your hierarchy like so.

You can delete the Main Camera in the hierarchy. As we are not going to need it.

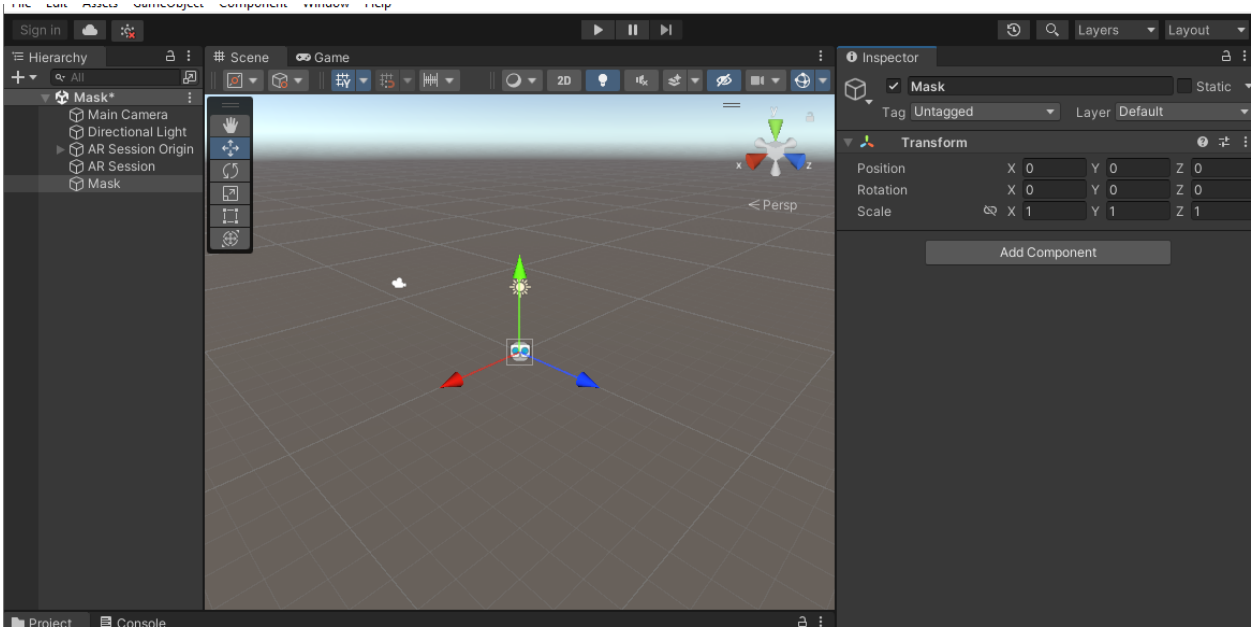
Make sure you save your work often, now is a good idea to do that.



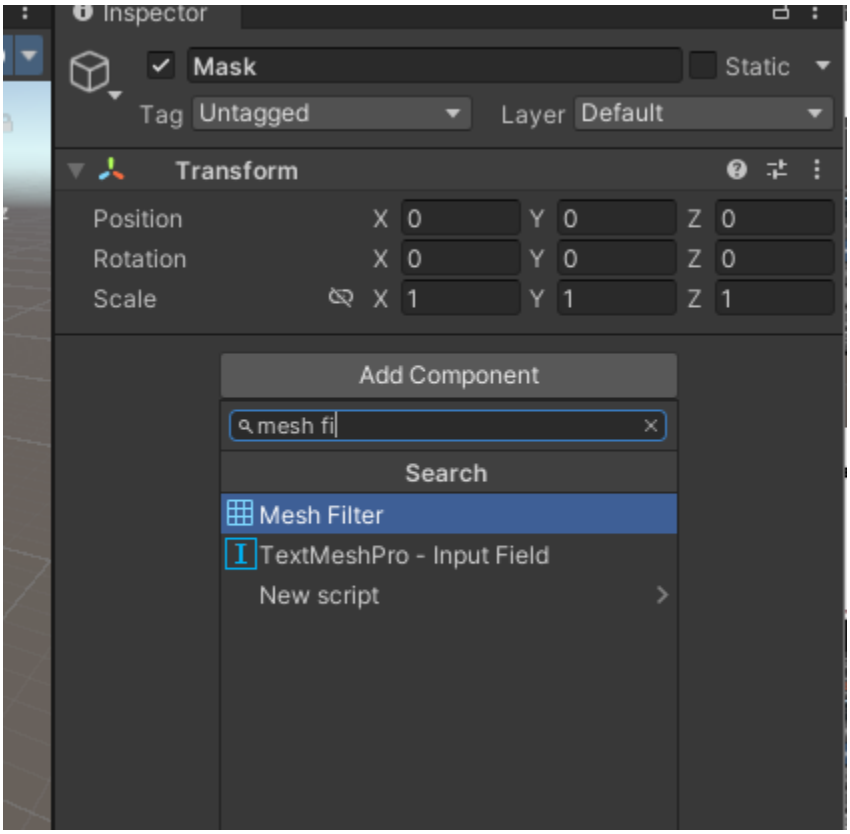
Right click the hierarchy and “create empty” face object. I’ll keep it simple and name it “Mask”.



Make sure the game object is centered to the origin by making sure its position is (0,0,0) in the inspector tab.

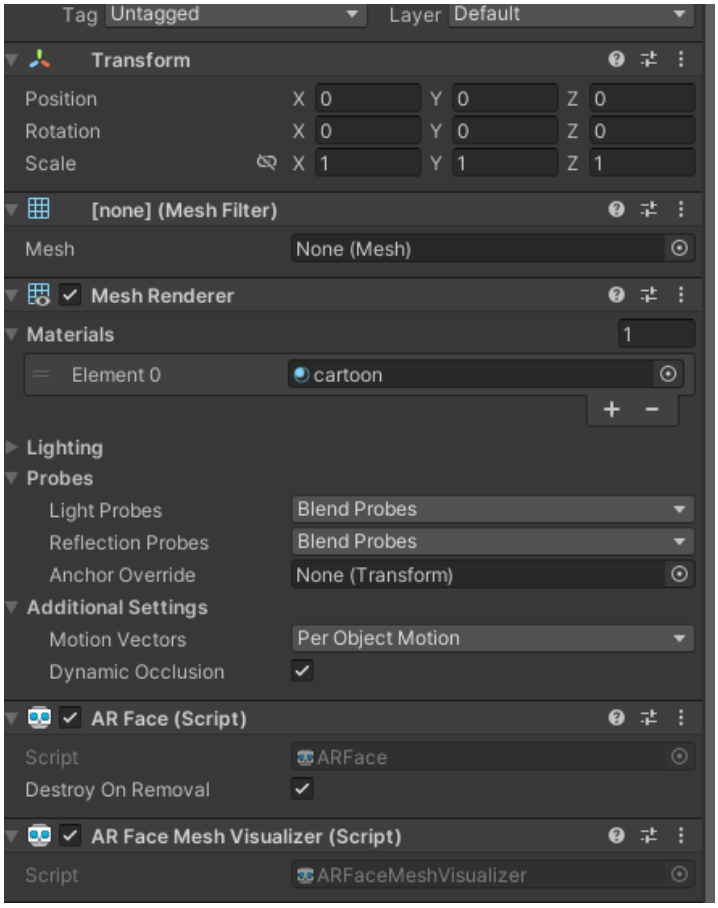


While the game object (Mask) is selected, click on Add component in the inspector tab.

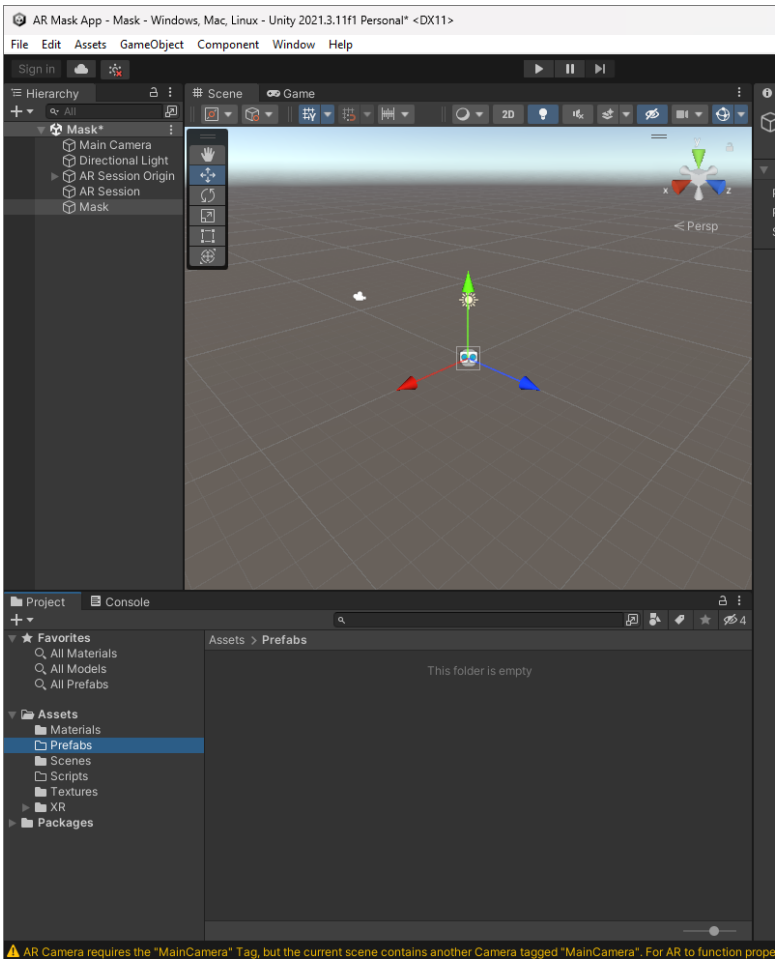


Find and left click “Mesh Filter”.

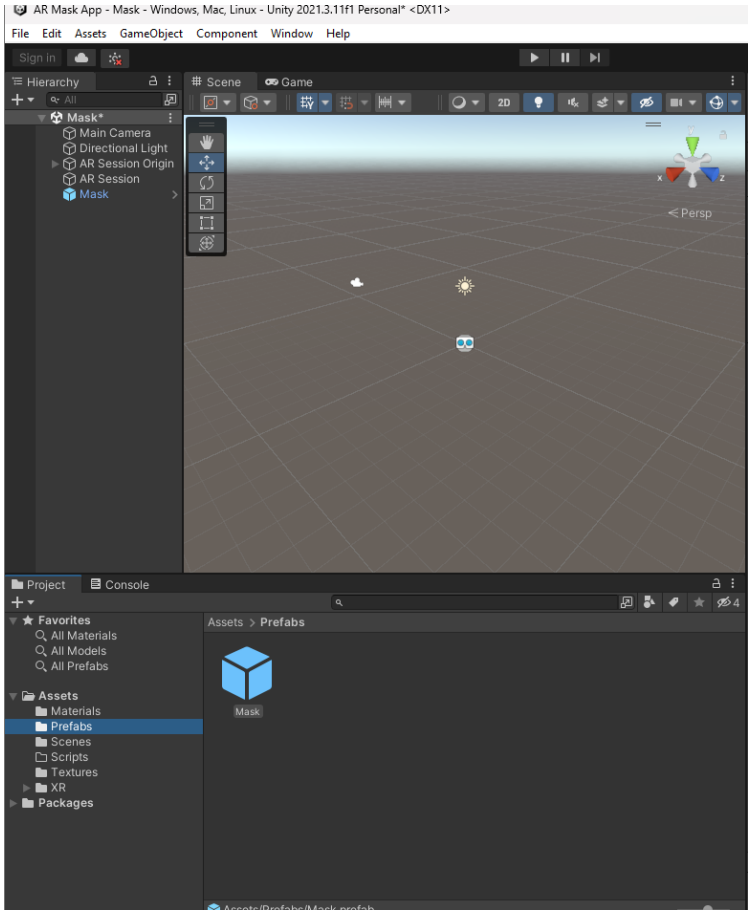
Repeat the last step and add “Mesh Renderer”, “AR Face” and “AR Face Mesh Visualizer”.



Your inspector tab for the game object “Mask” should look like this now.

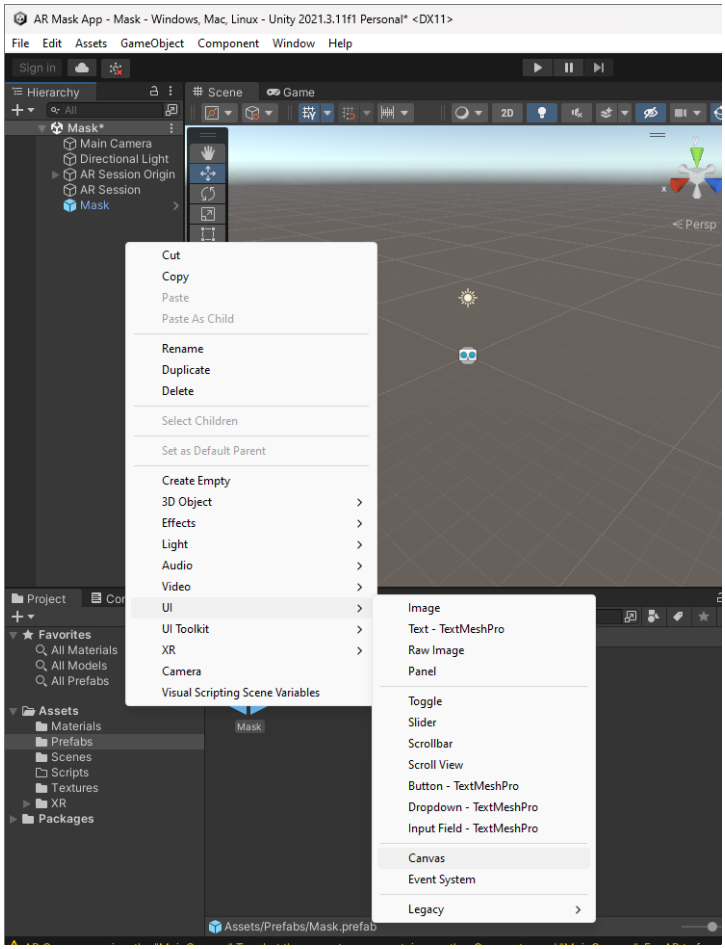


Open the Prefabs folder and drag the “Mask” game object from the hierarchy to the folder.

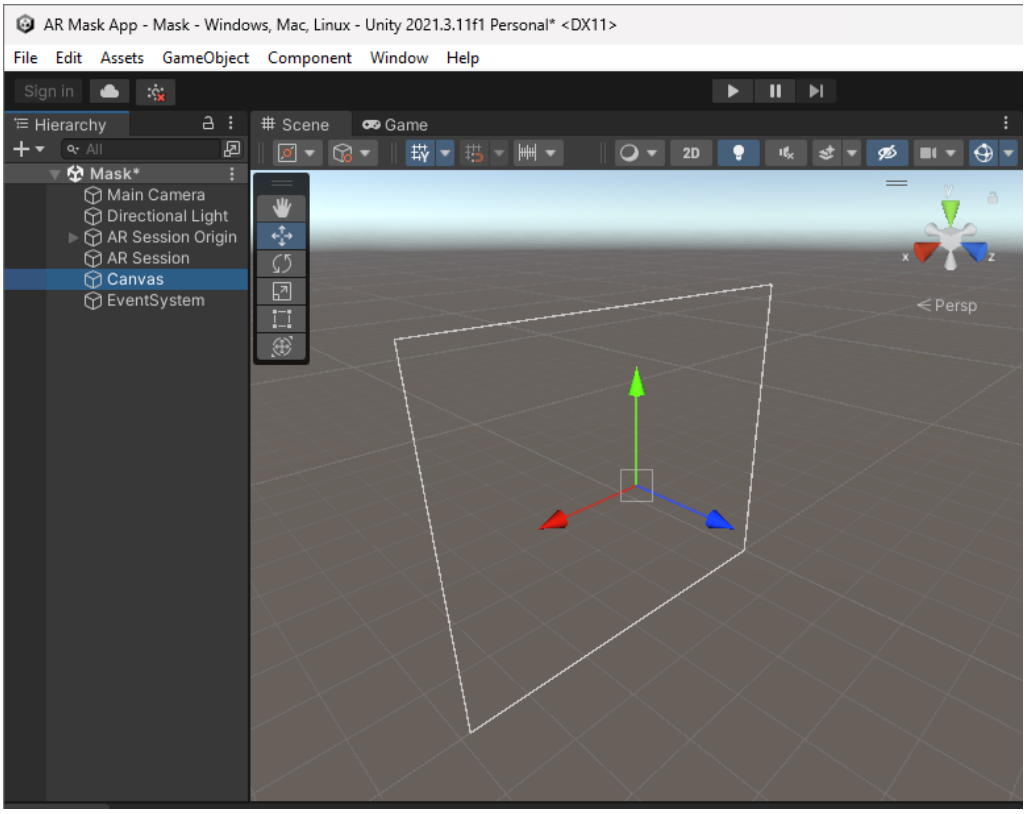


You will notice the game object icon turning blue, that means it is now a prefab.

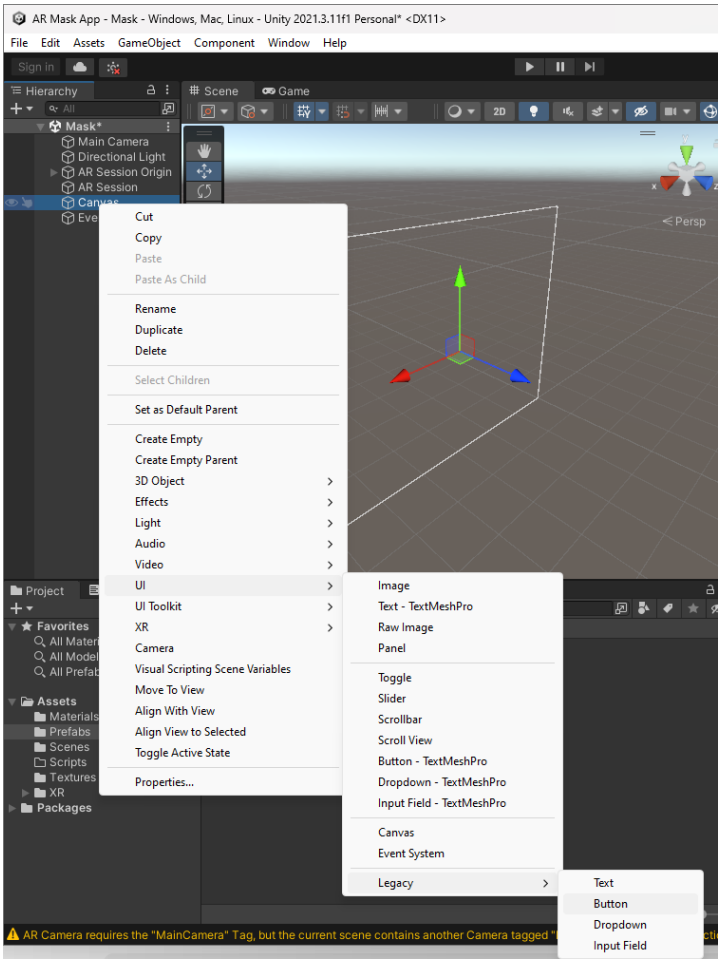
You can now delete the game object you created from the hierarchy. Nothing will happen to the version of the game object which is now a prefab (located in the prefabs folder)



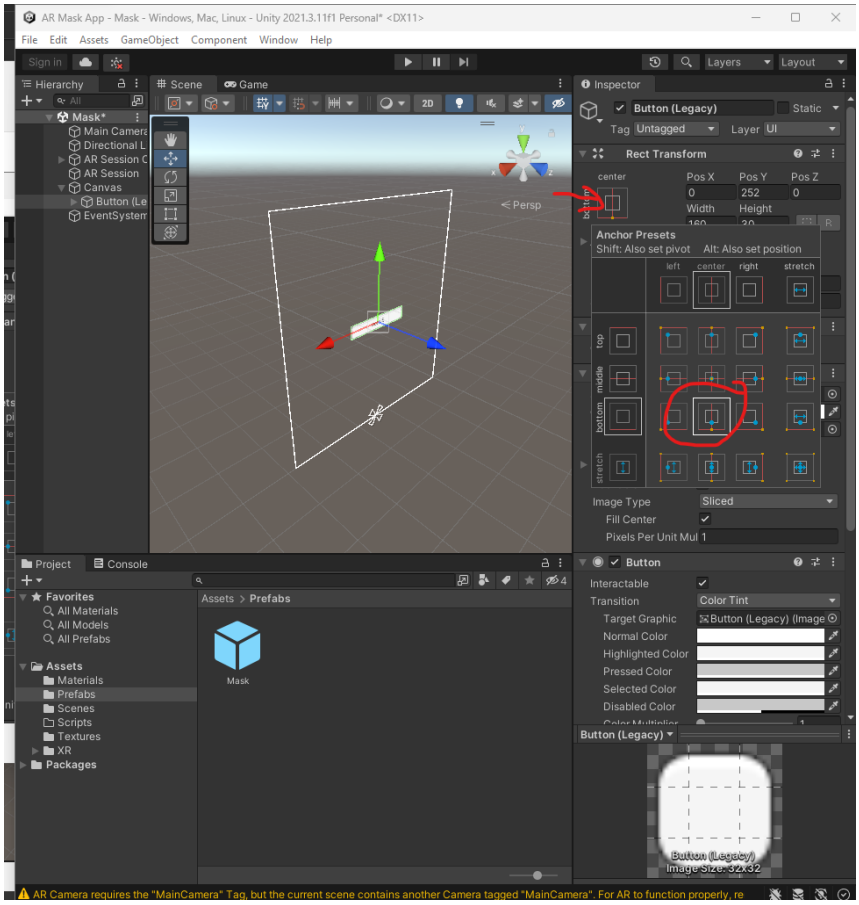
We want to add some interactivity to the android app and make the user be able to switch between face masks with different materials on them. So we will create an interface through Unity's Canvas UI. Right click on the hierarchy and go to UI->Canvas.



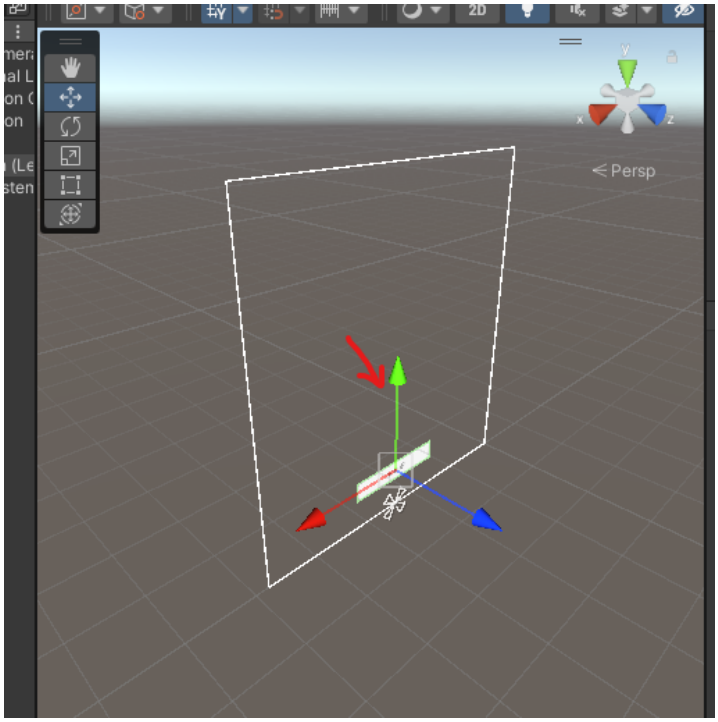
Double left-clicking the canvas on the hierarchy will show how it is laid out by default.



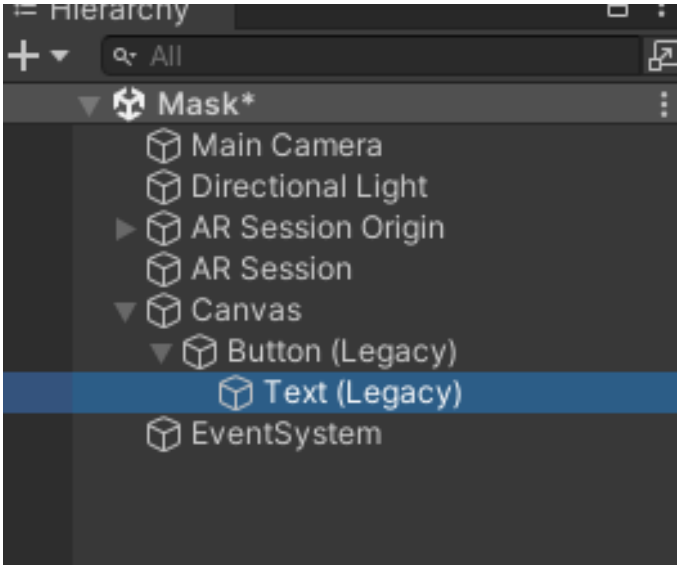
Now, right-click on the Canvas in the hierarchy and find UI->Legacy->Button. Finding the button selection might differ from one version to another.



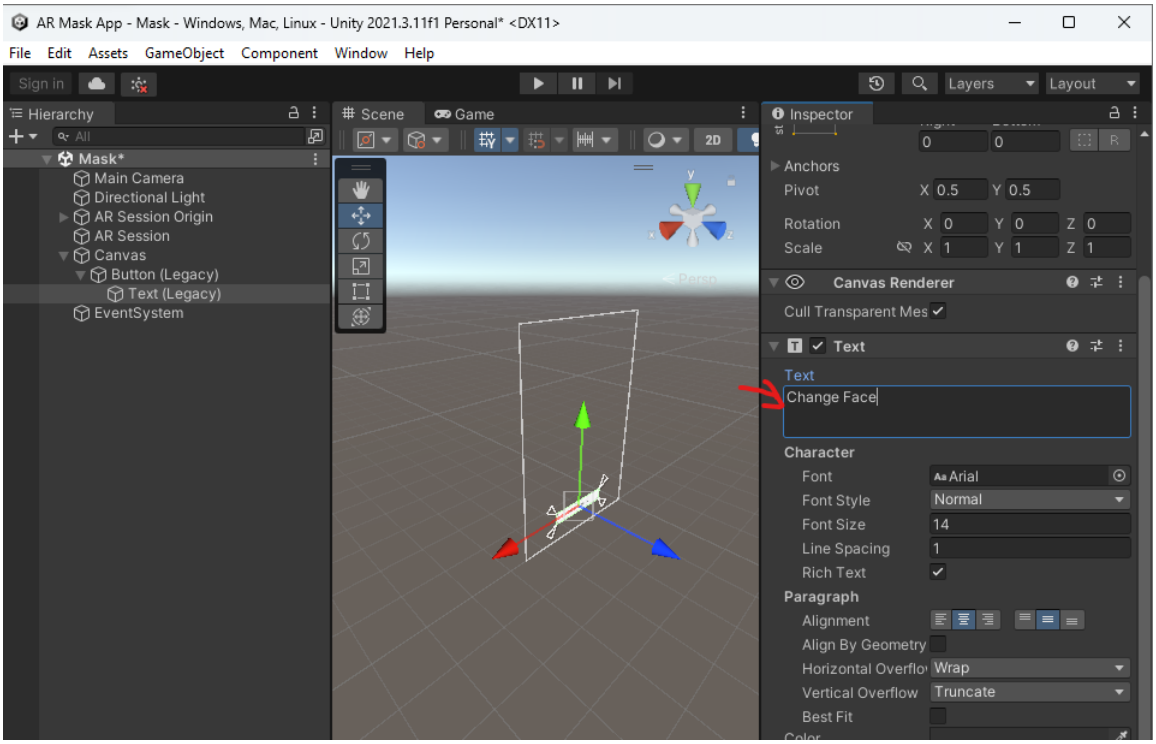
Select the Button in the hierarchy and click the box under Rect Transform. Then choose the prefeed position you want to anchor the button to in the canvas layout. In this case, we choose the bottom.



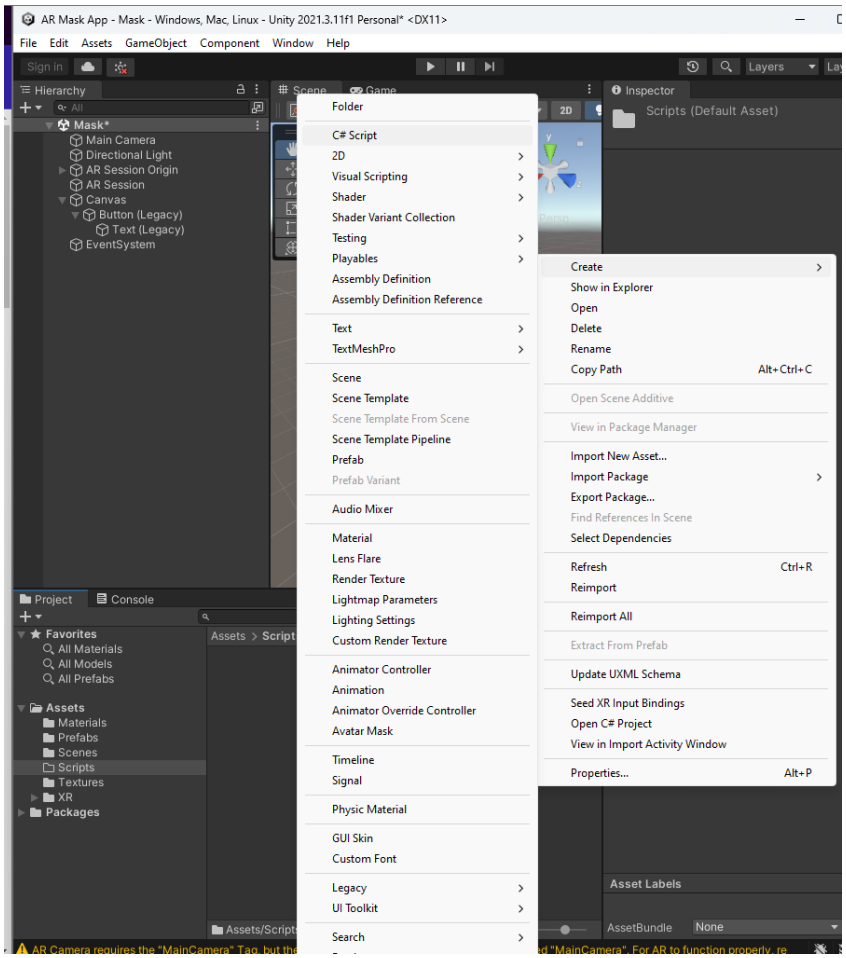
Use the green arrow in the scene to move the position of the button as desired in the interface.



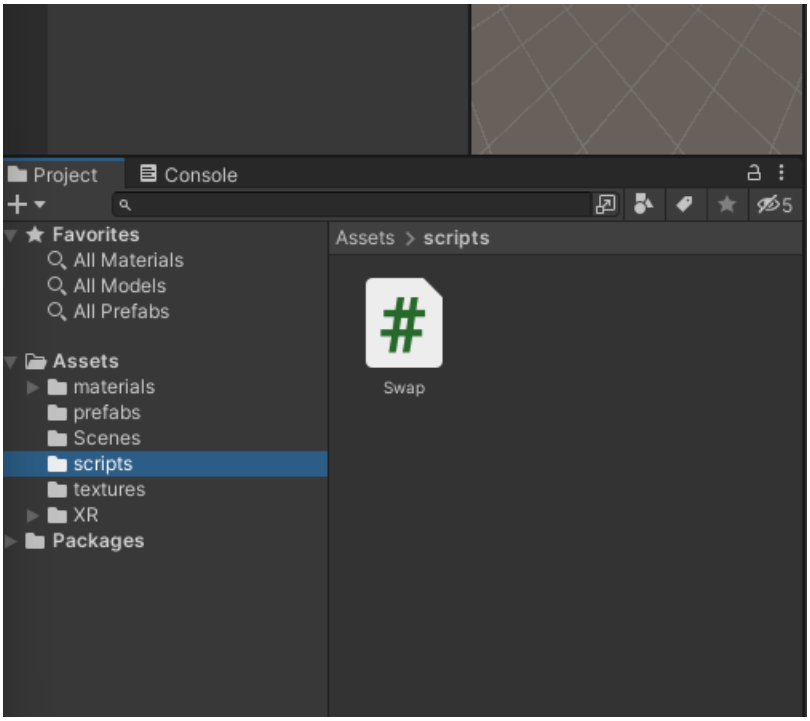
Find the Text object in the hierarchy and select it.



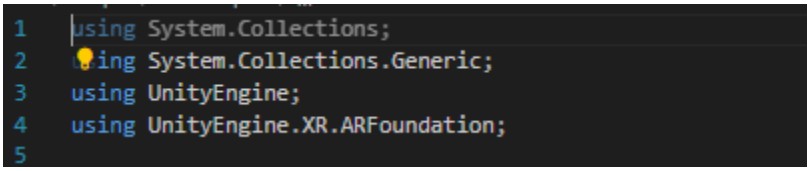
Edit the text on the button to say, “Change Face”.



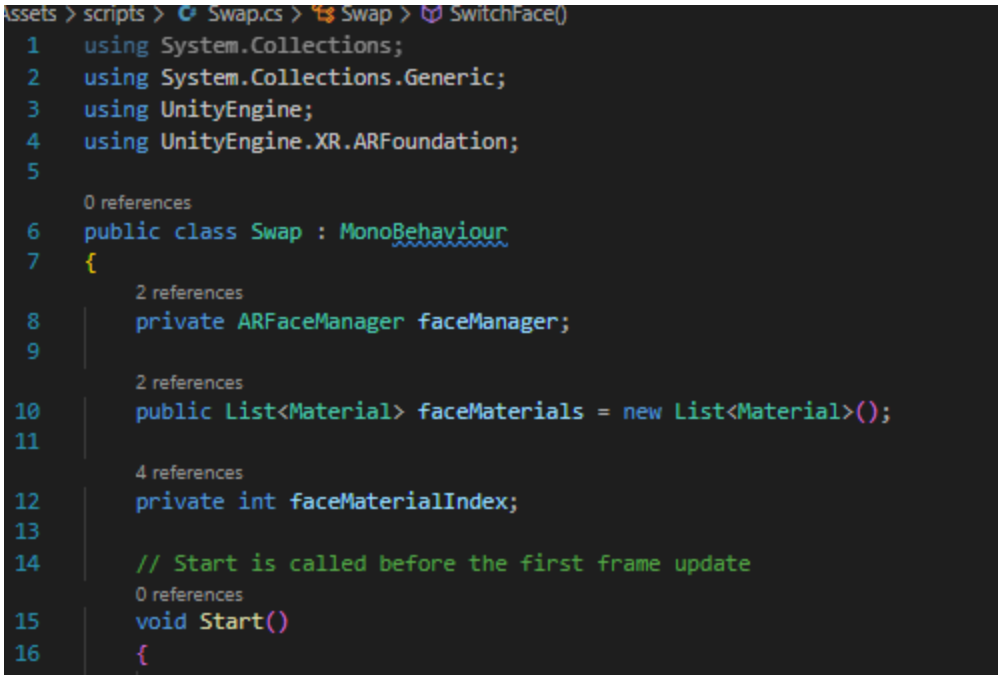
Now go to Assets->Scripts and create a new C# script.



I will name it Swap since it will contain the script to swap face materials/textures.



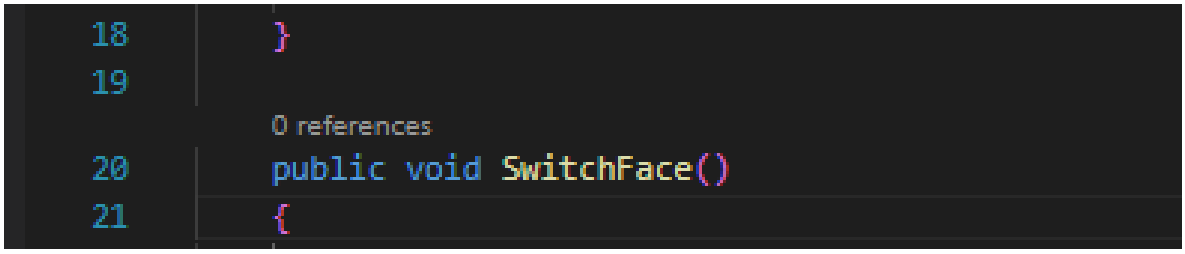
Double click the Swap file and open in your preferred editor. Add line 4 in the screenshot above to gain access to AR foundation XR namespace.



Inside the Swap class, add line 8 which will give us access to the AR Face Manager component in the session origin we created earlier.

Add line 10 to declare a list that will hold all materials we want to switch between.

Add line 12 to declare an index identifier for looping through the materials list.



Rename the update function to SwitchFace, this is where all the switching functionality will take place.

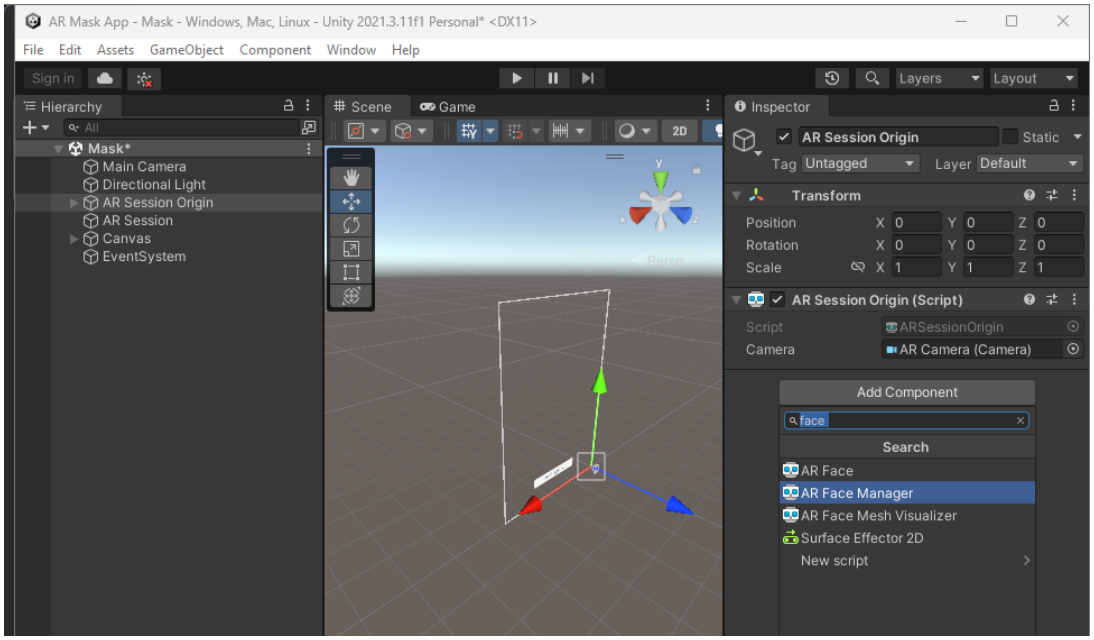
```
19
        0 references
20      public void SwitchFace()
21      {
22          foreach (ARFace face in faceManager.trackables)
23          {
24              face.GetComponent<Renderer>().material = faceMaterials[faceMaterialIndex];
25          }
26          faceMaterialIndex++;
27
28          if(faceMaterialIndex == faceMaterials.Count)
29          {
30              faceMaterialIndex = 0;
31          }
32      }
33  }
34
```

A loop inside the SwitchFace function will loop through the face manager's materials. This is initiated as seen in line 22. Then inside the loop we will get each material from the array faceMaterials. A known functionality of the AR Face Manager component.

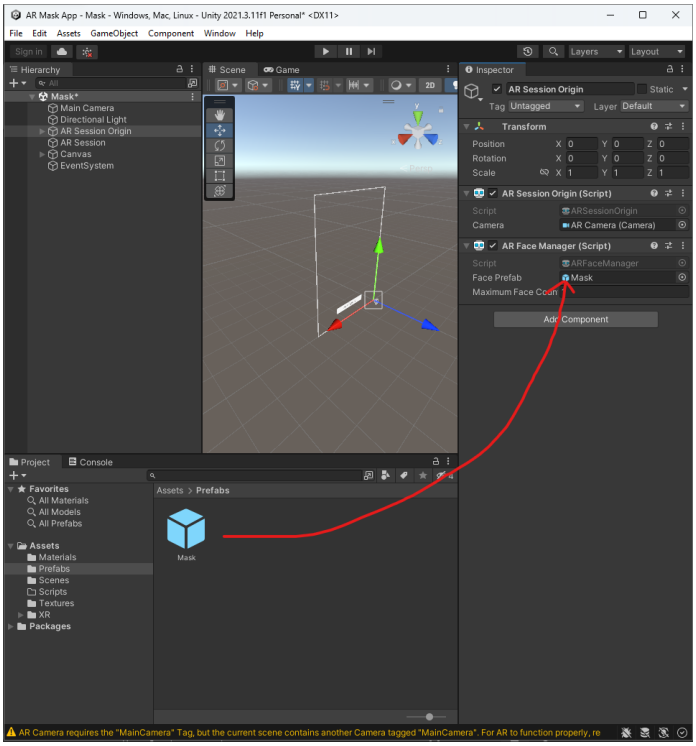
Outside the loop at line 26, we add an incrementation of the index to move to the next material.

Finally, we use an if statement to check if the index reaches the (beyond) maximum material count, so that it resets. This allows the toggling to happen in a circular fashion.

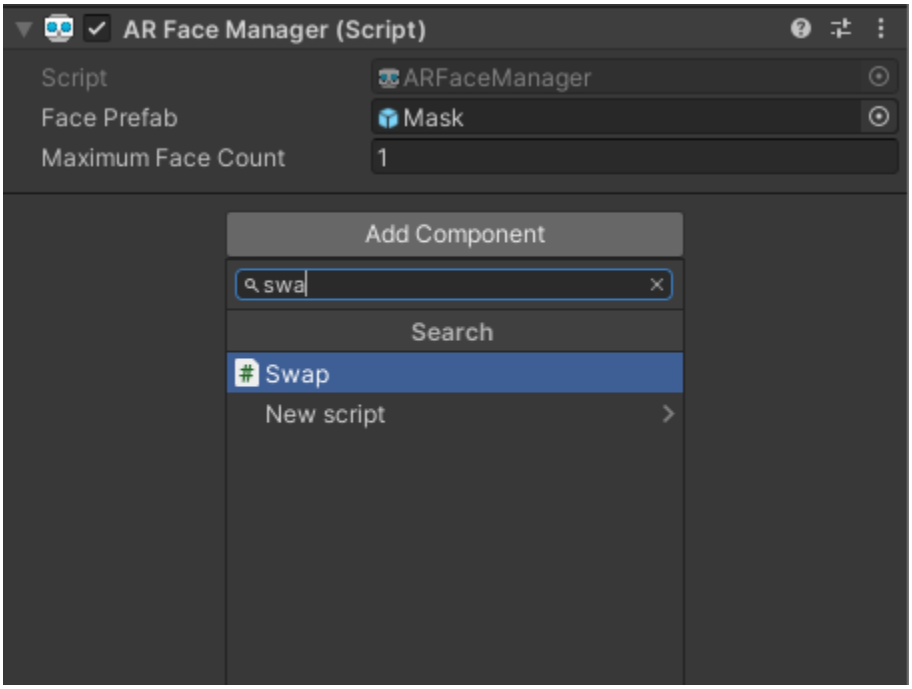
Save your changes and exit the editor.



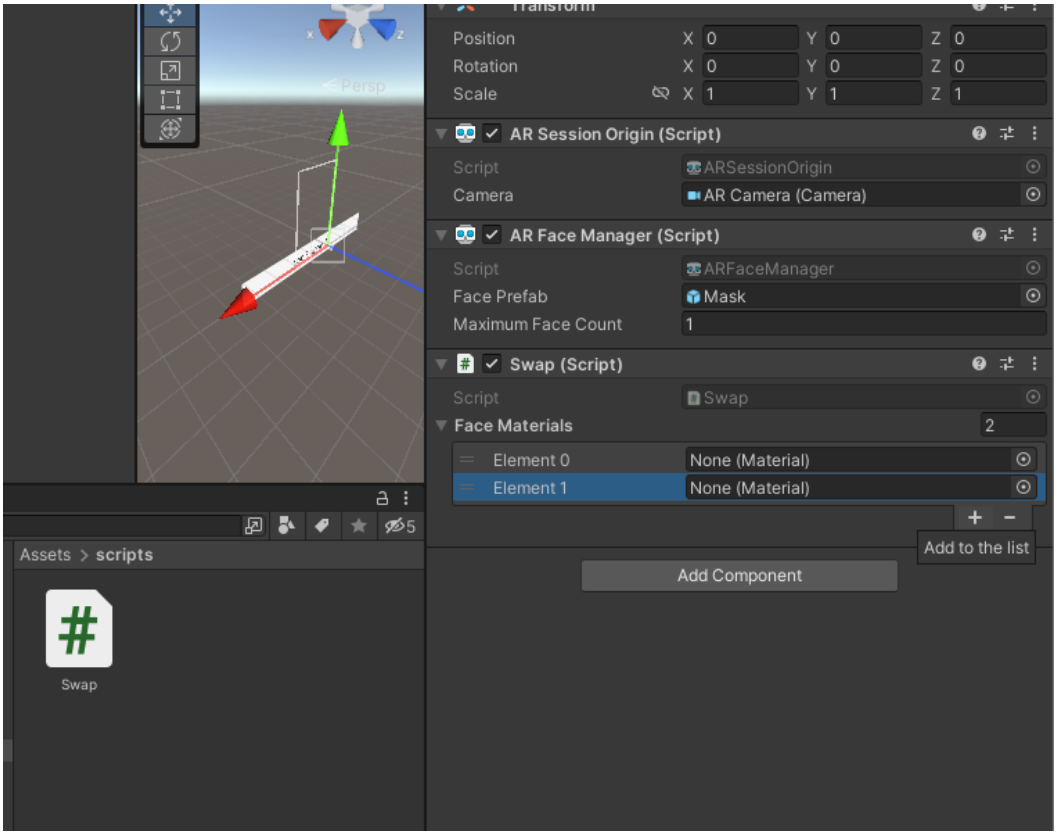
Select the AR Session Origin in the hierarchy and add a new component called AR Face Manager. (Similar to how we added the last few components)



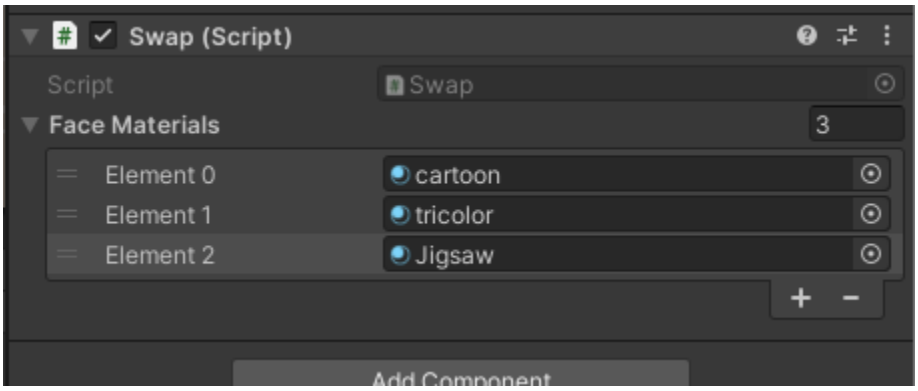
Go to the prefabs folder, select the AR Session Origin from the hierarchy. Now, drag and drop the prefab we created earlier to the AR Face Manager component where it says Face Prefab.



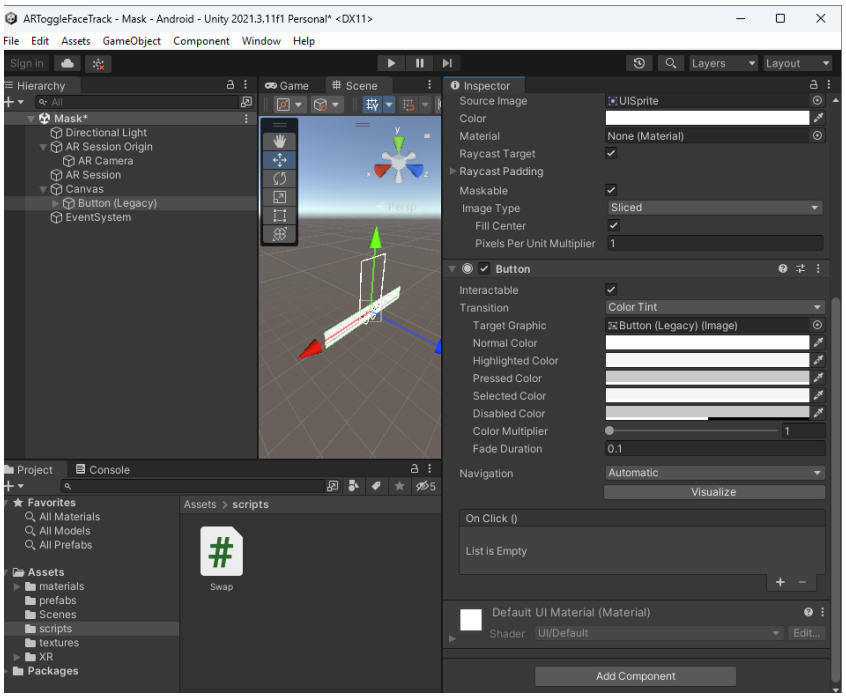
Before leaving, add our customized script as a component under AR Face Manager.



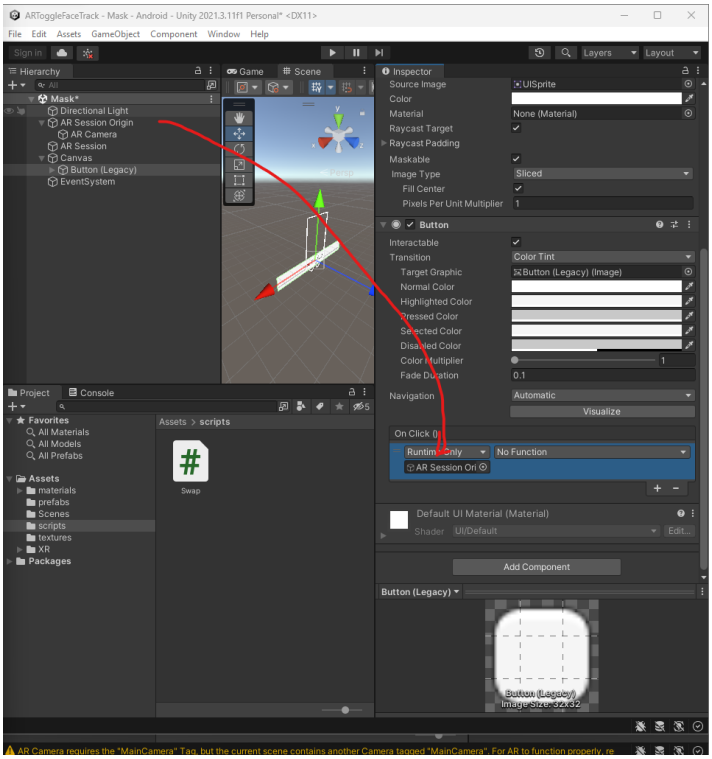
Add as many faces as you want to the interface by clicking the “+” under Face Materials. Click on the circles next to the Element row to associate materials to them (NOT textures)



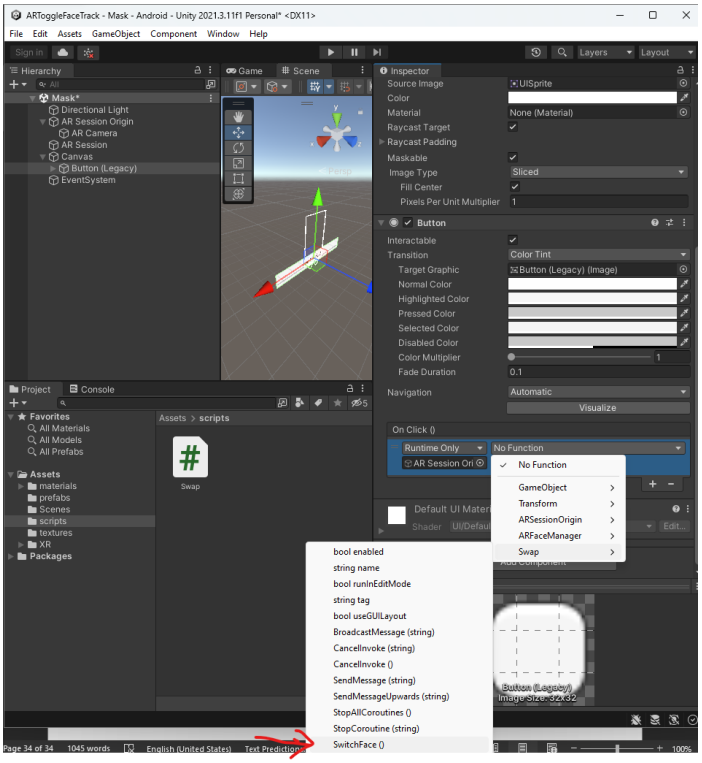
Should look something like this now.



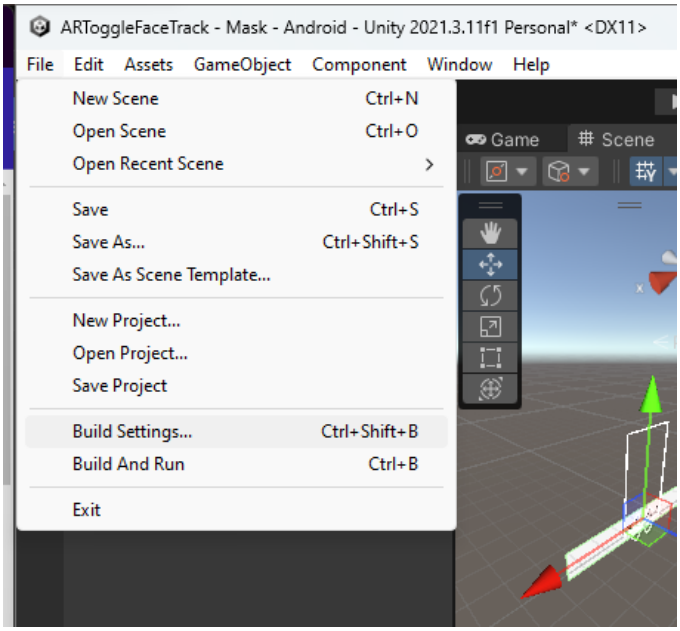
To associate the function we made to switch faces to the interface button: Find the button on the hierarchy, after selecting find the On Click () section in the inspector. Then click the “+” sign.



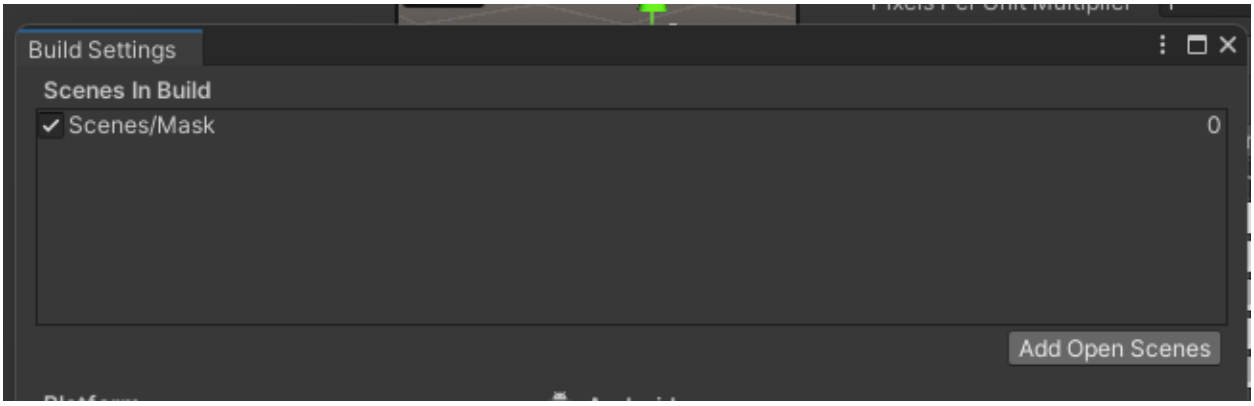
Drag the AR Session Origin to the rectangle under “Runtime Only”.



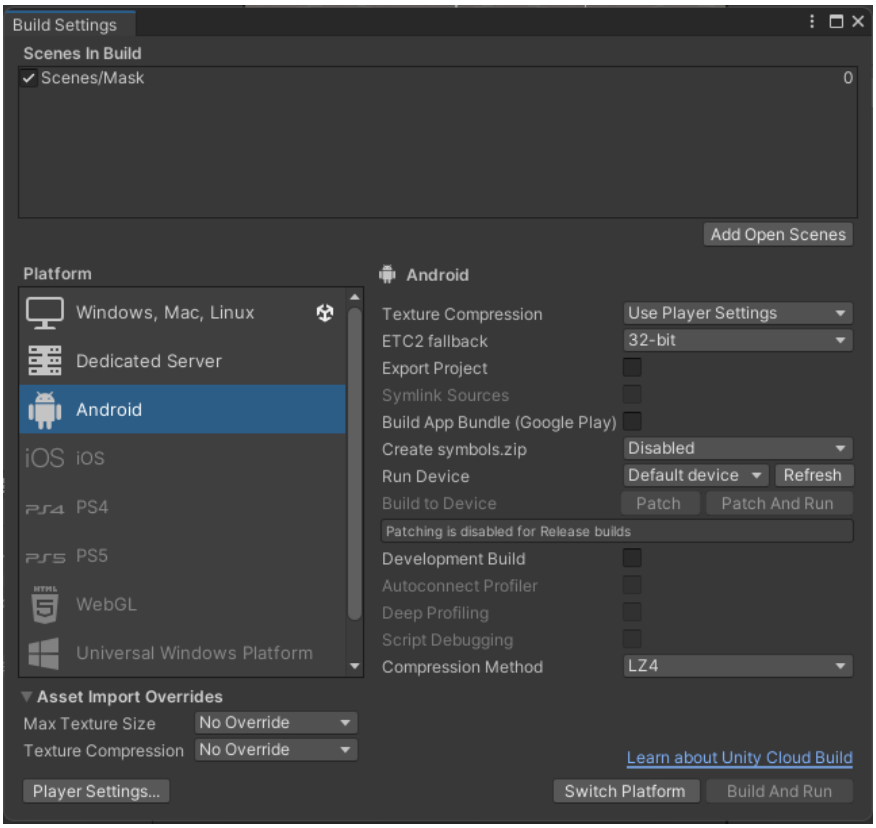
Then, click the drop down menu next that says “No Function” to invoke a function of this button. Click SwitchFace().



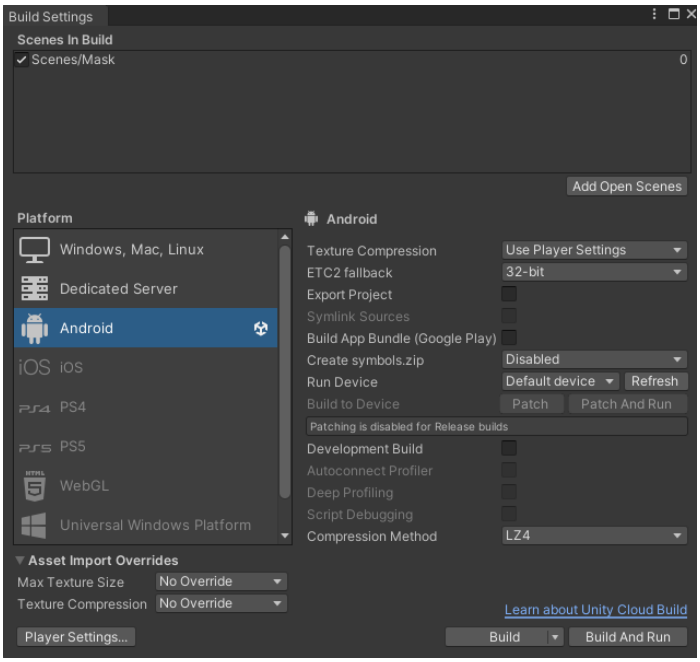
Go to File->Build Settings



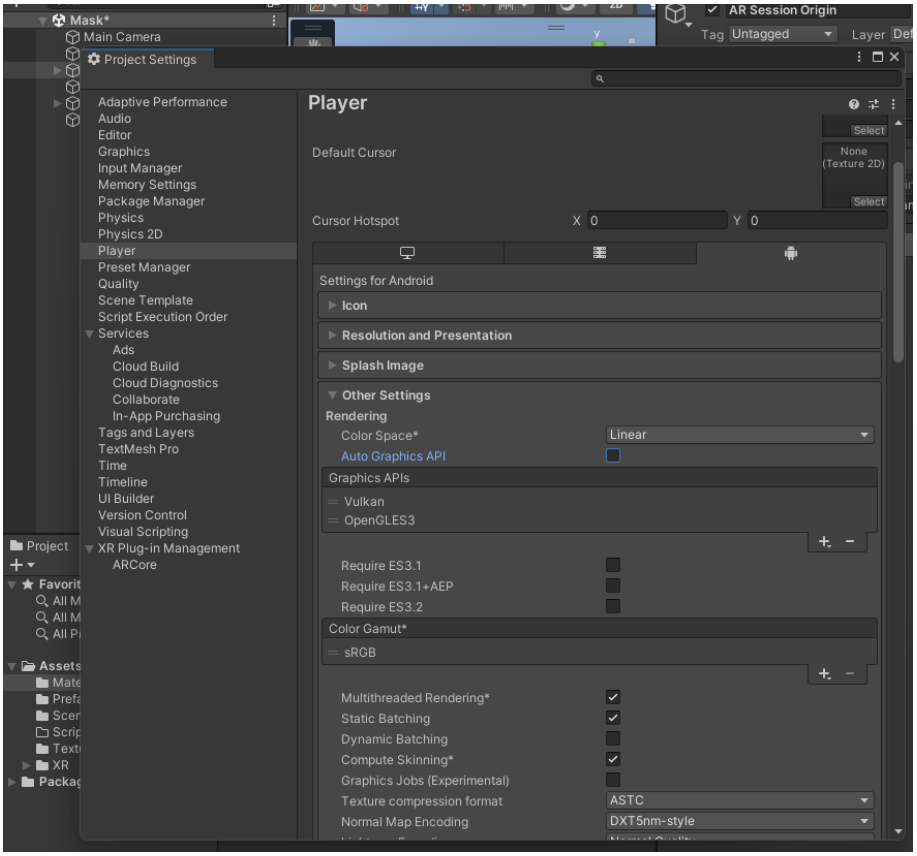
Click Add Open Scenes



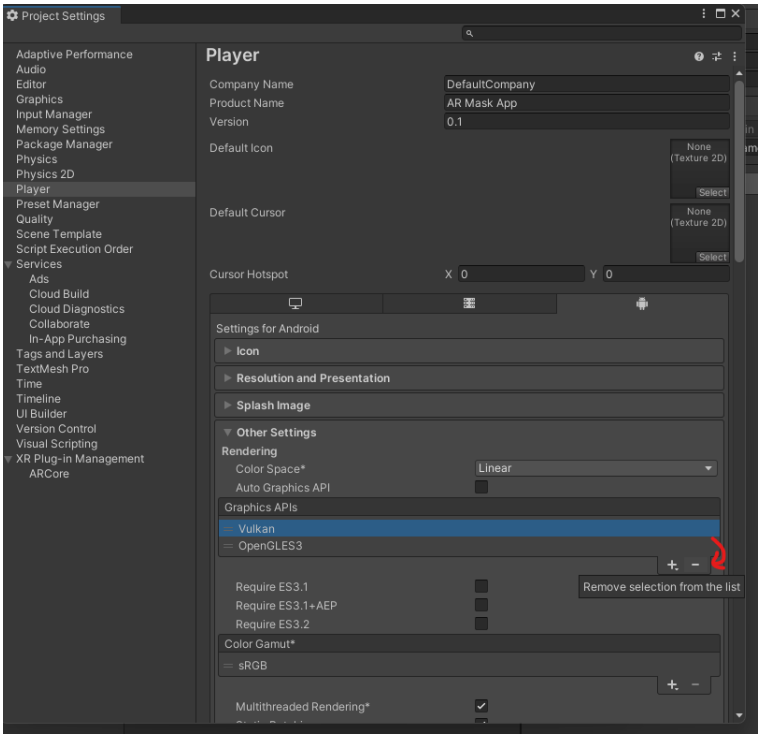
Choose Android under “Platform”. And click Switch Platform on the bottom right of the pop up.



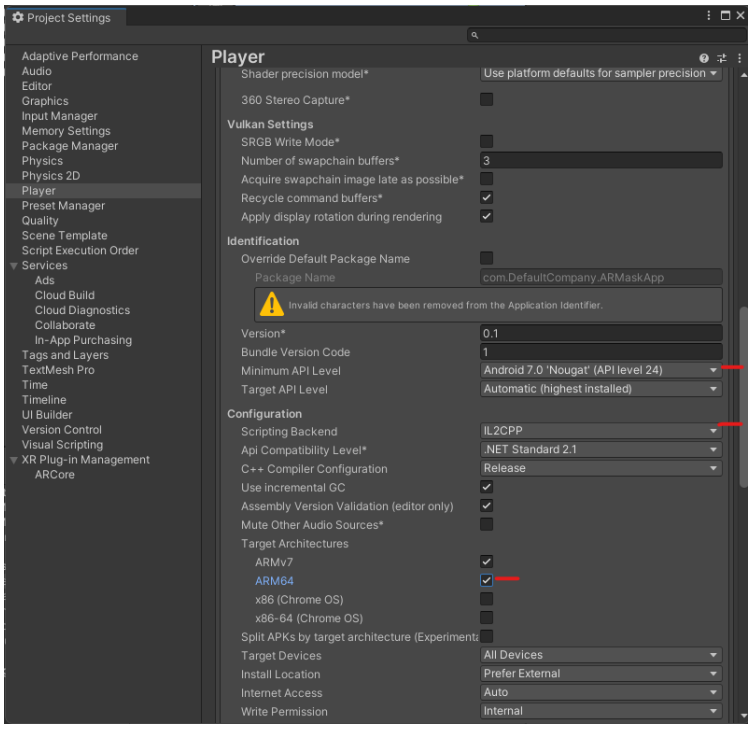
Click on Player Settings on bottom left.



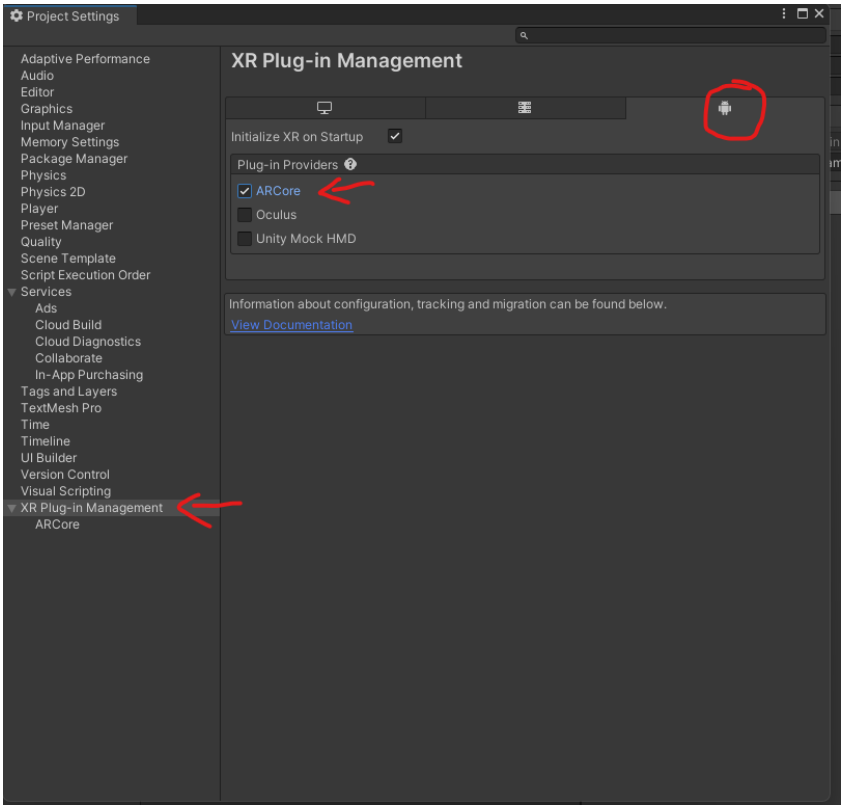
Uncheck Auto Graphics API.



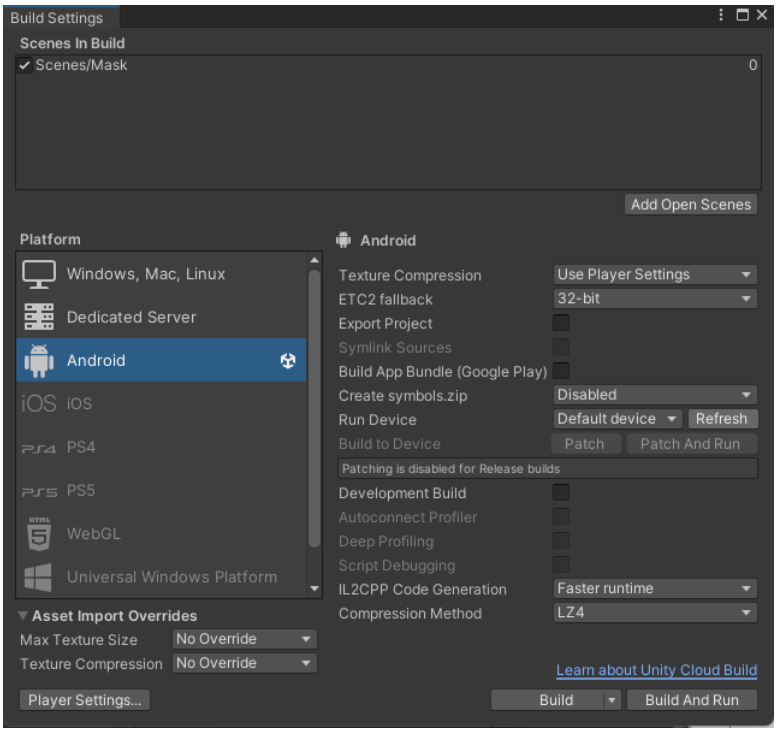
Click on Vulkan and remove it.



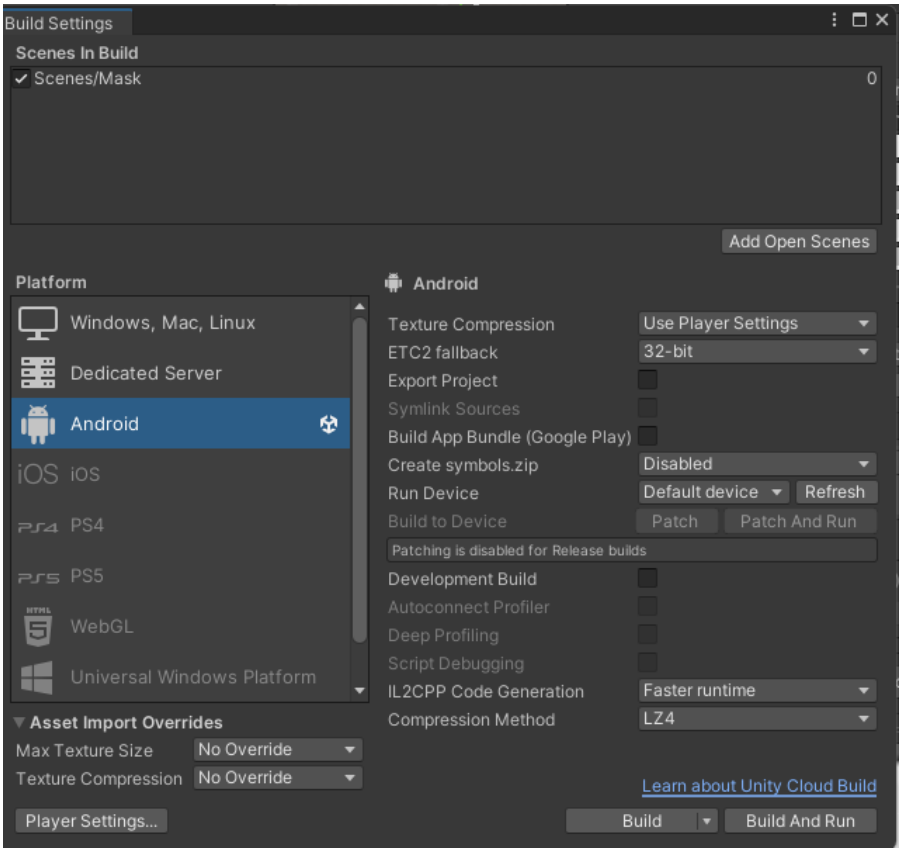
For Android, these settings would have to be changed (shown in red)



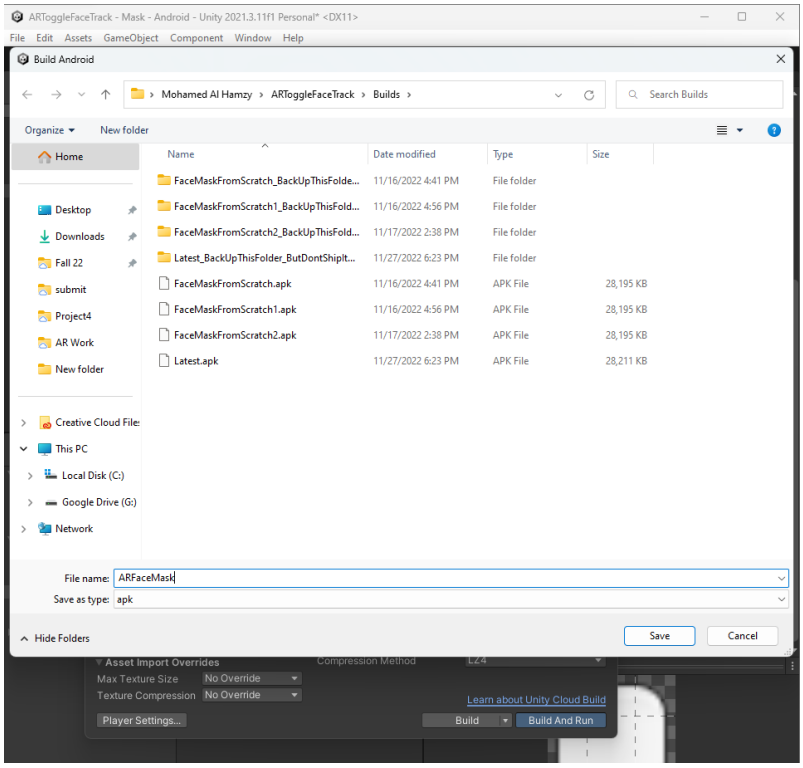
Find the AR Plug-In Management on the left, and on the android platform tab (circled). Check the ARCore box.



With your android phone connected. Go back to build setting and next to Run Device. Click on Refresh, then click the drop down menu next to it to find your phone. And select it.

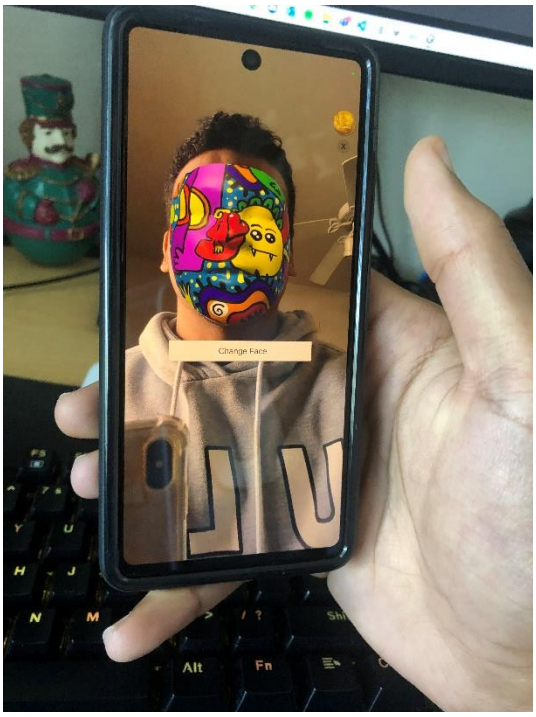


Click on Build and Run.

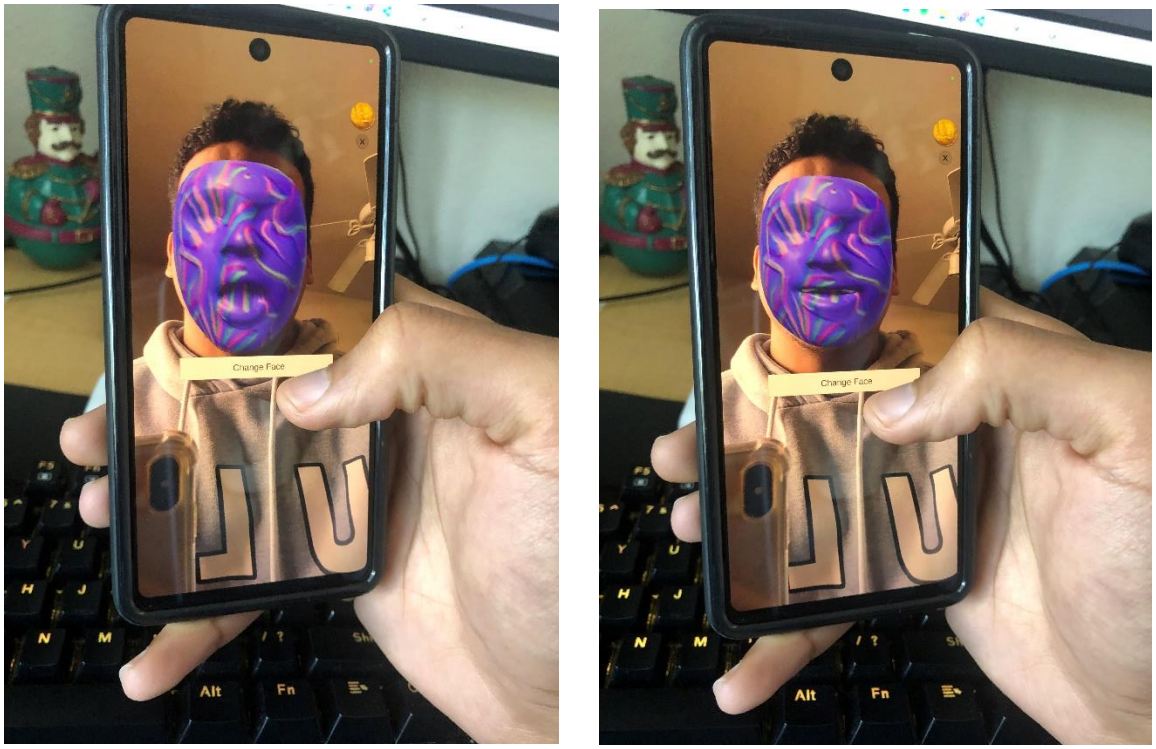


Rename and choose where you want to save your build, I like creating a folder for it inside the Assets folder.

The app should run automatically on the phone.



How the app should look initially



Toggling to another face.