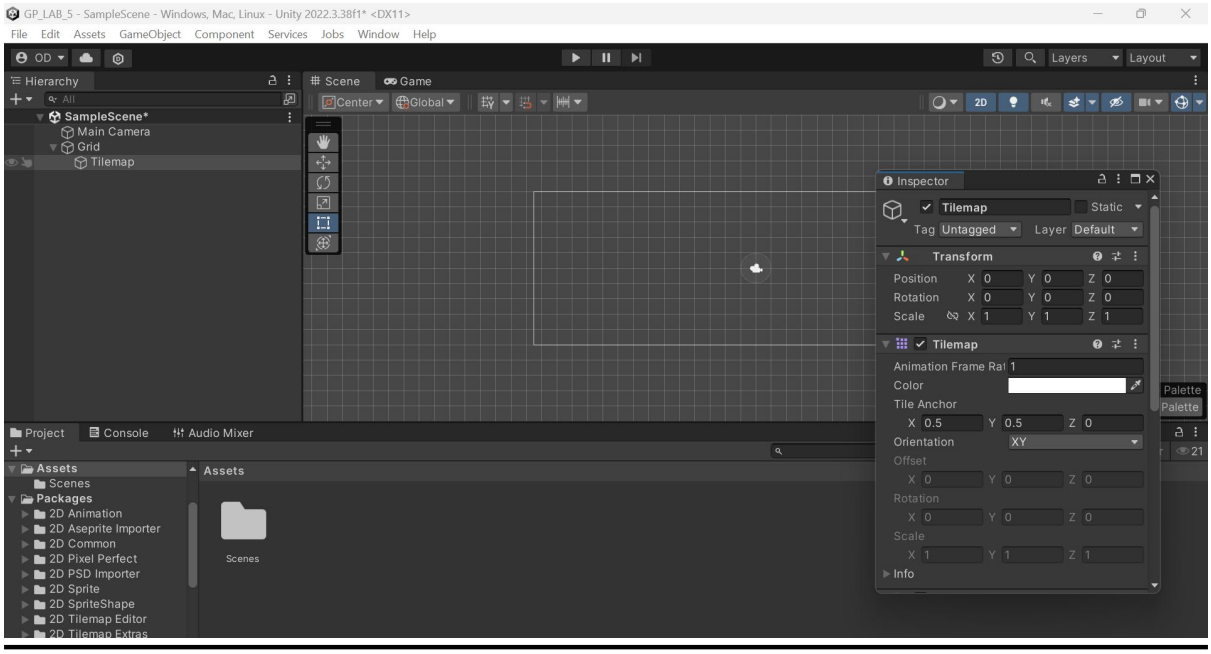
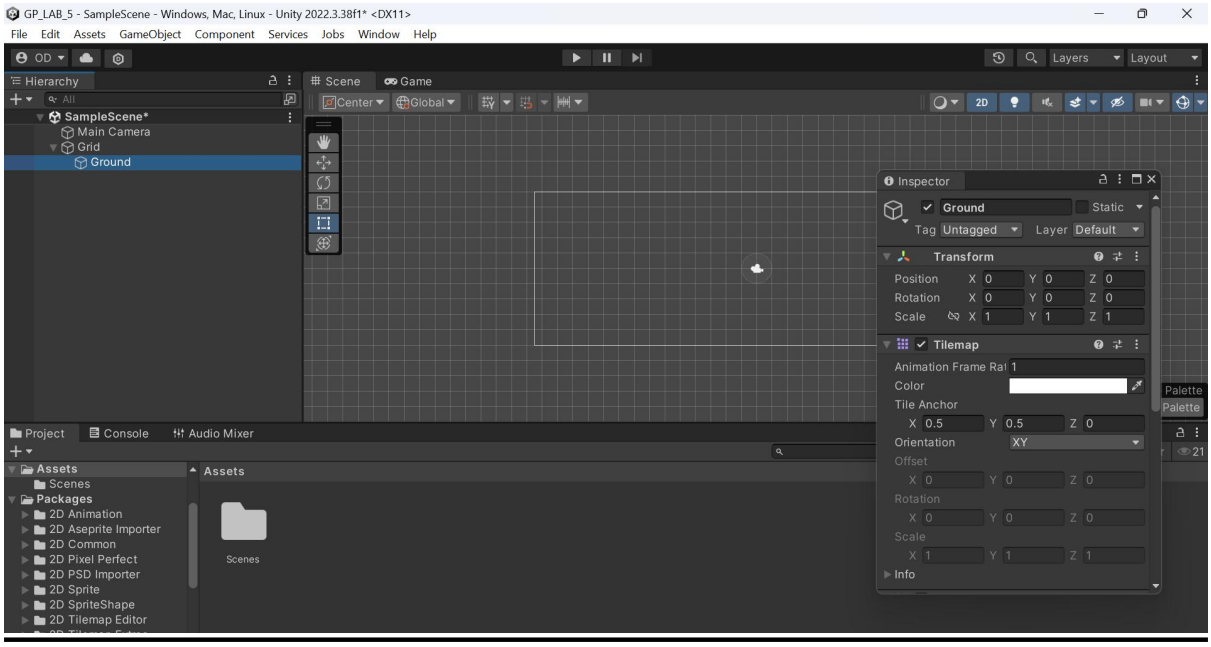
GAME PROGRAMMING LAB 2

NAME : OM SUBRATO DEY

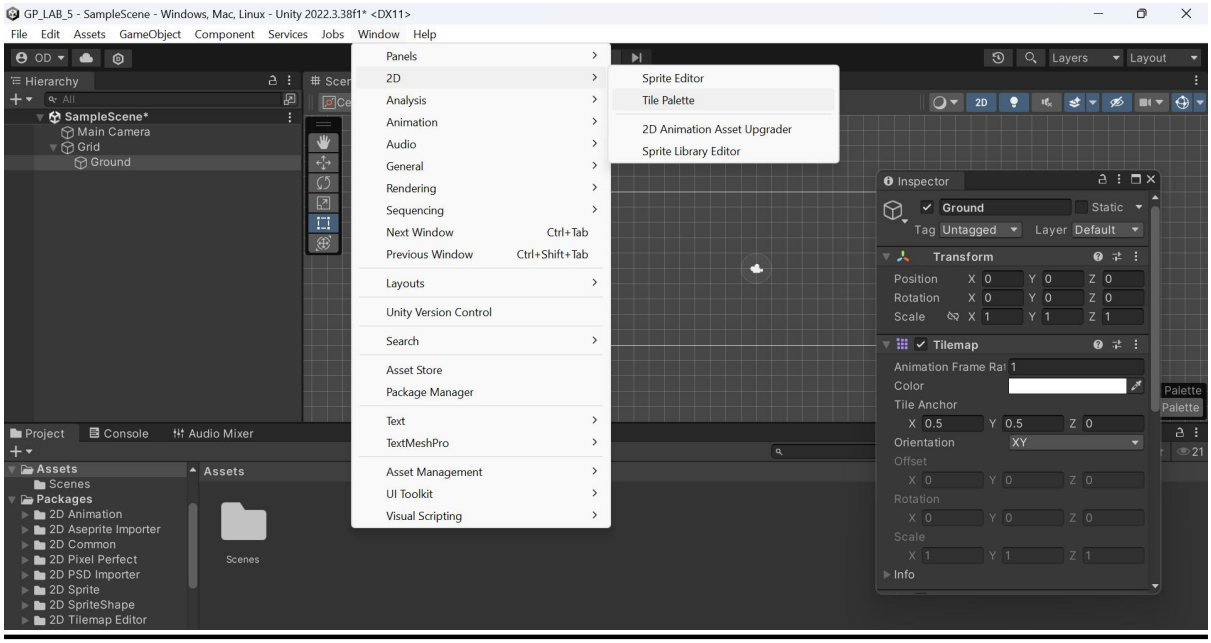
REGISTER NUMBER : 21BAI1876

1. Tile Based Grid – System Construction for the game

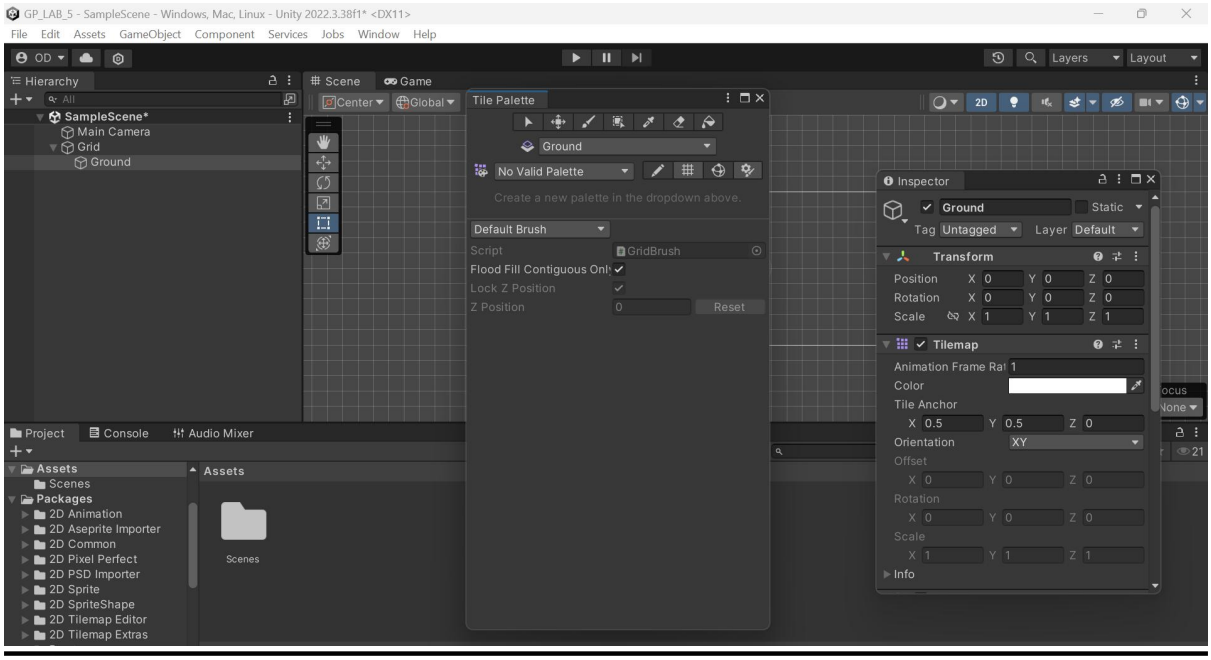




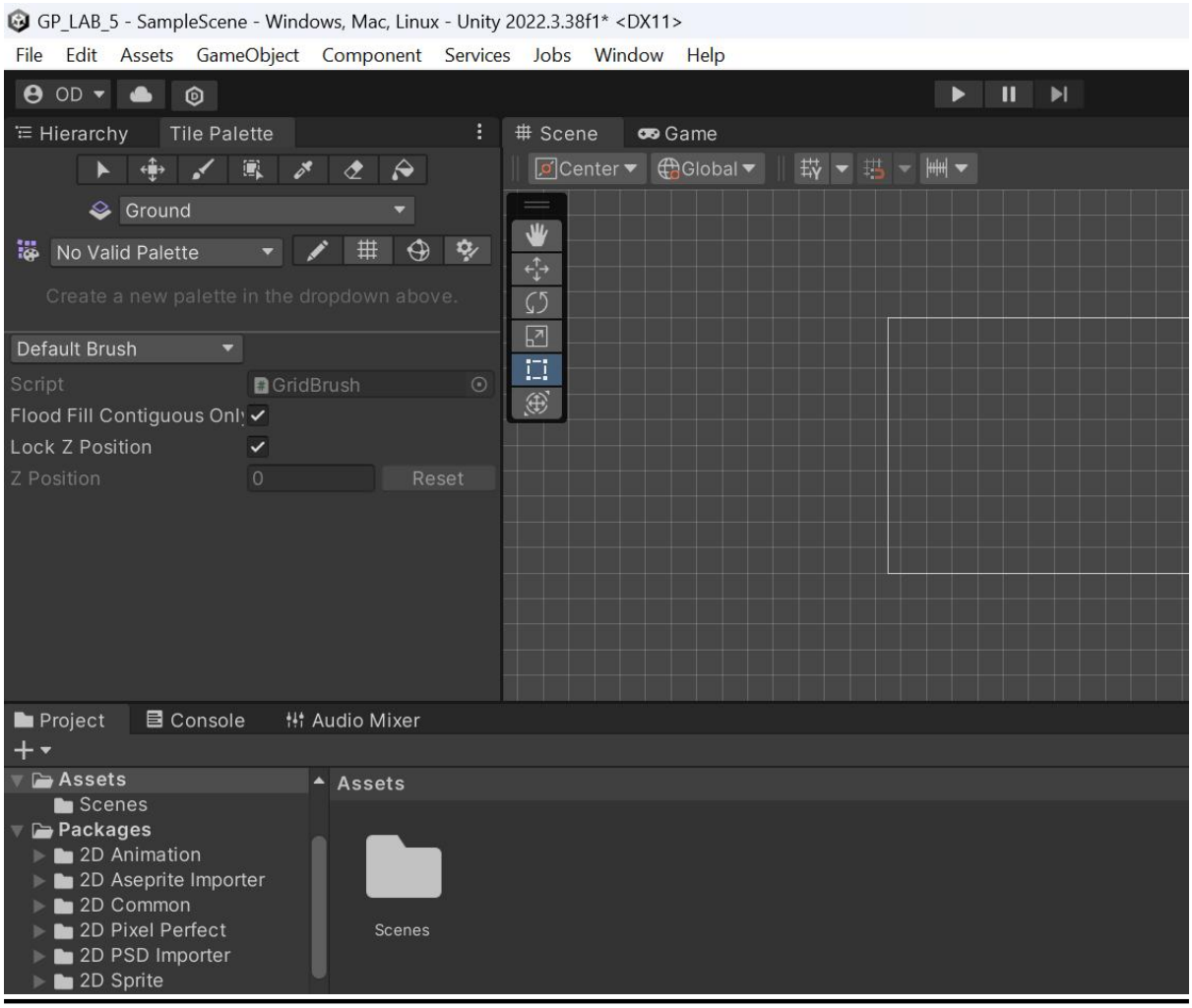
Go to Window -> 2D -> Tile Palette



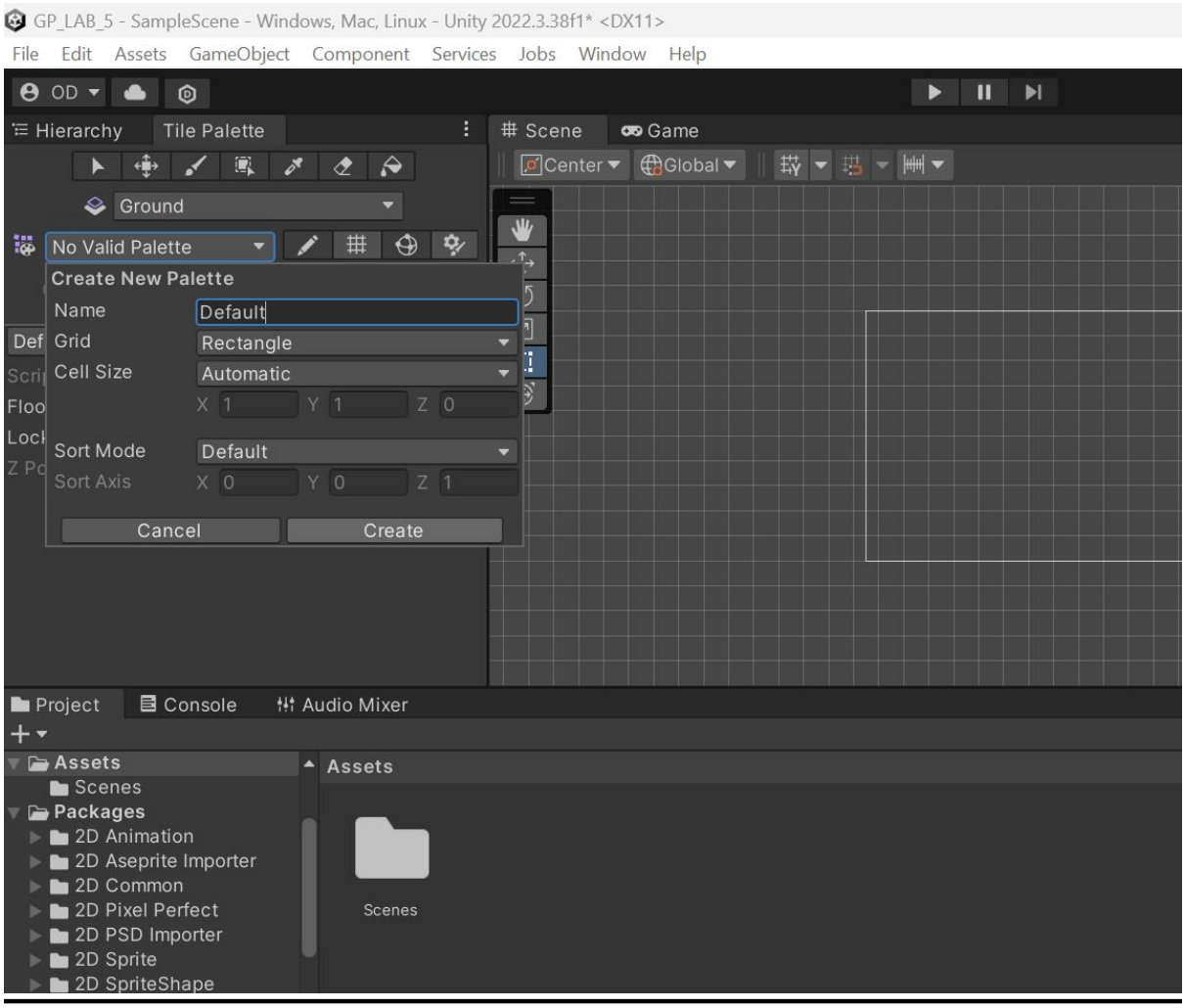
Open Tile Palette:



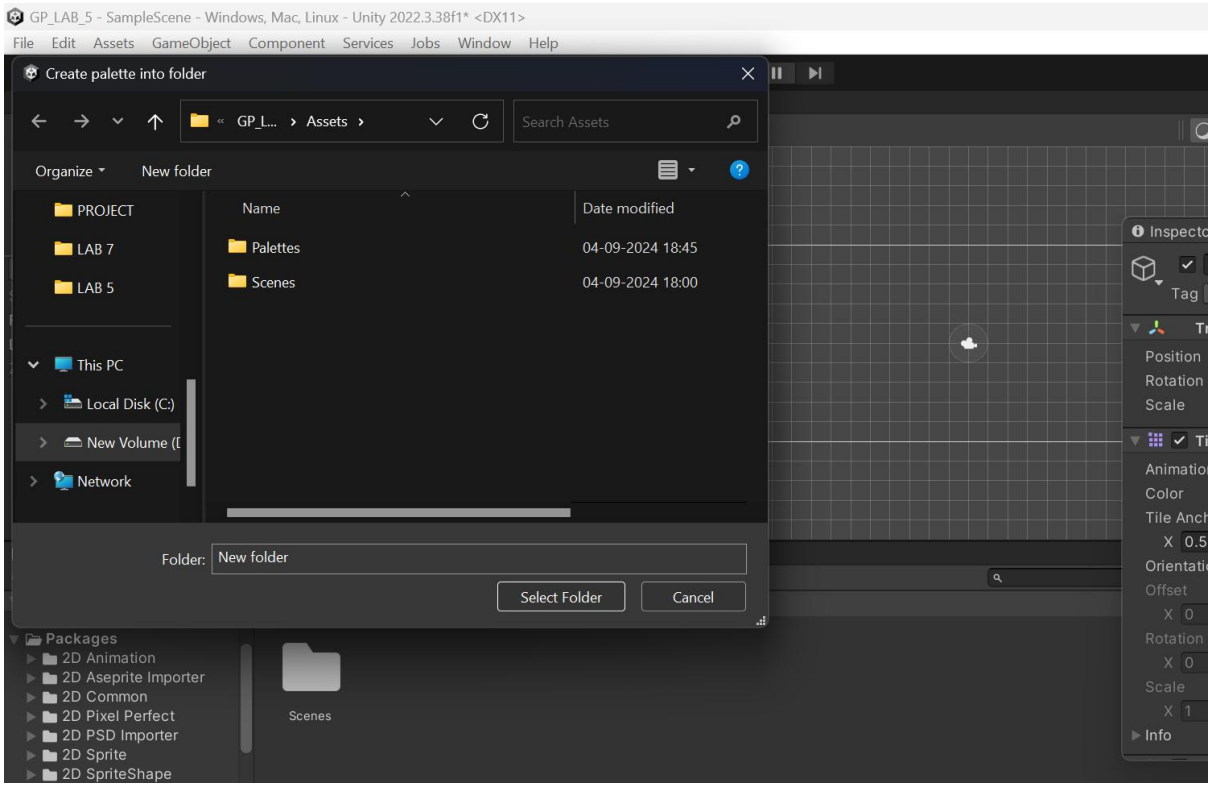
Dragged to hierarchy window:

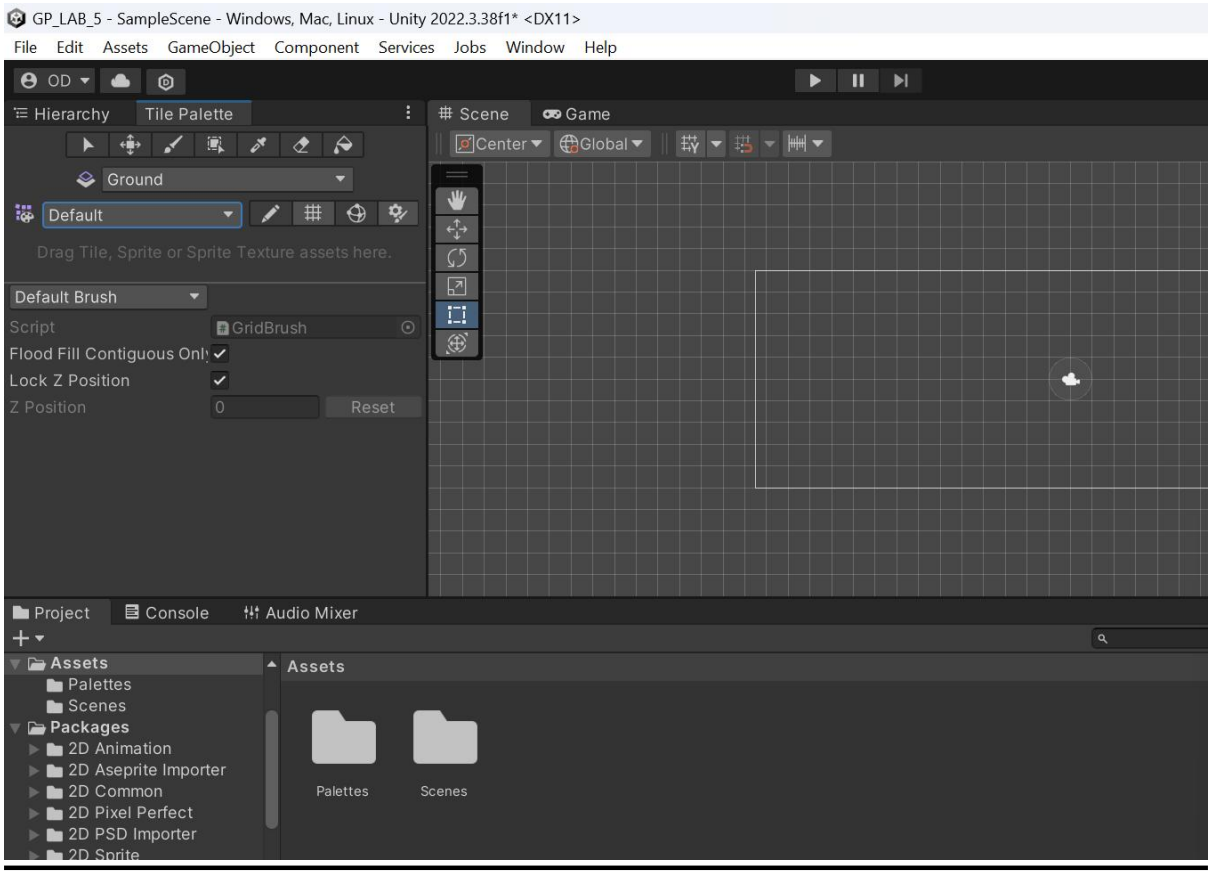


Create new palette:

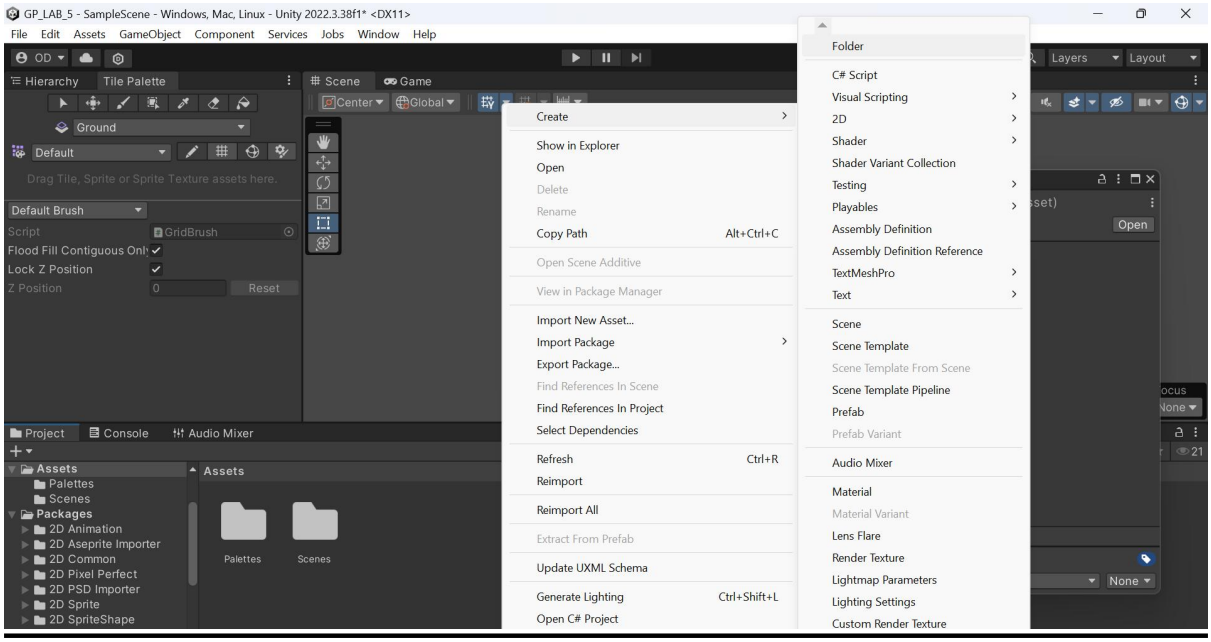


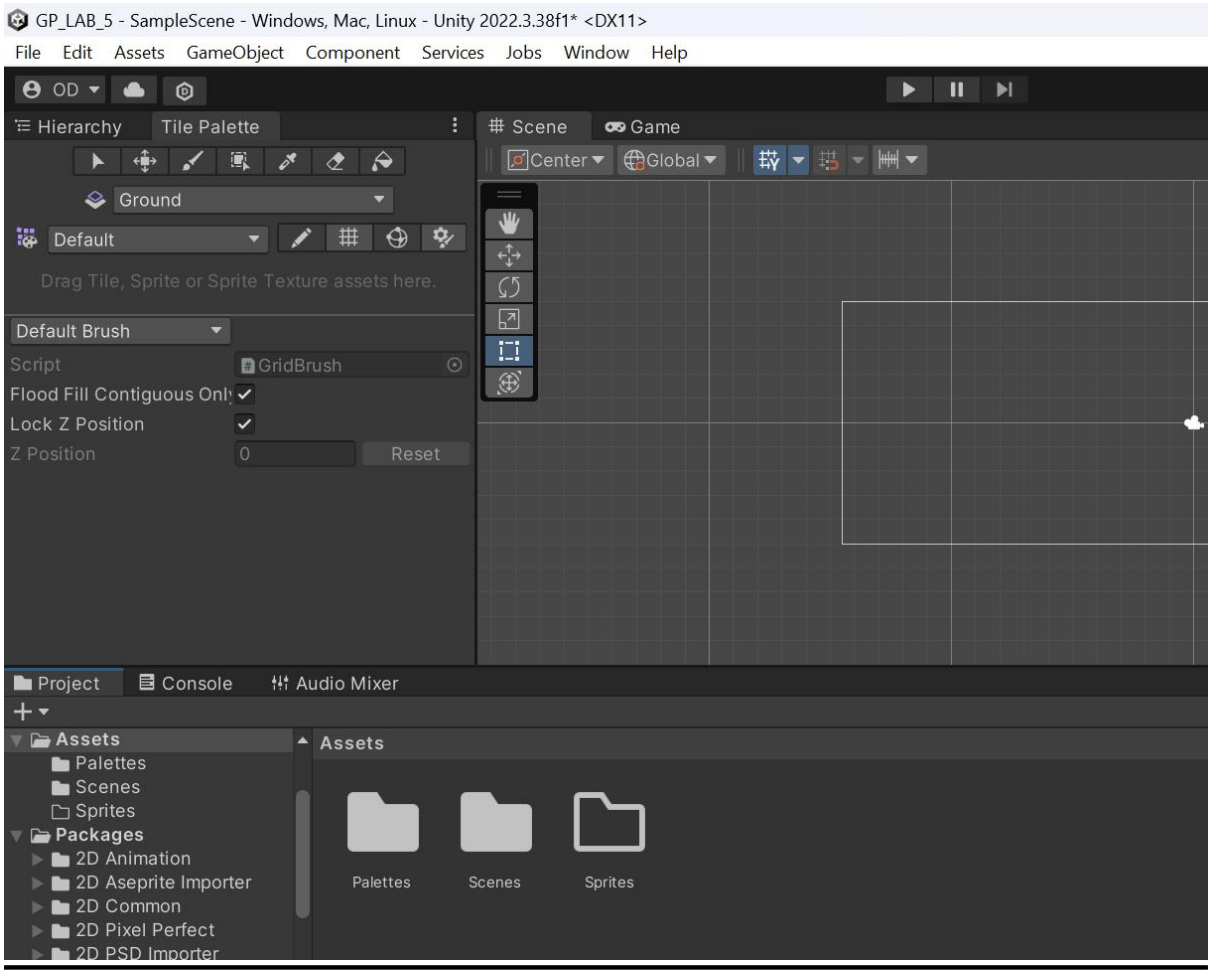
Create new folder palletes and add it there



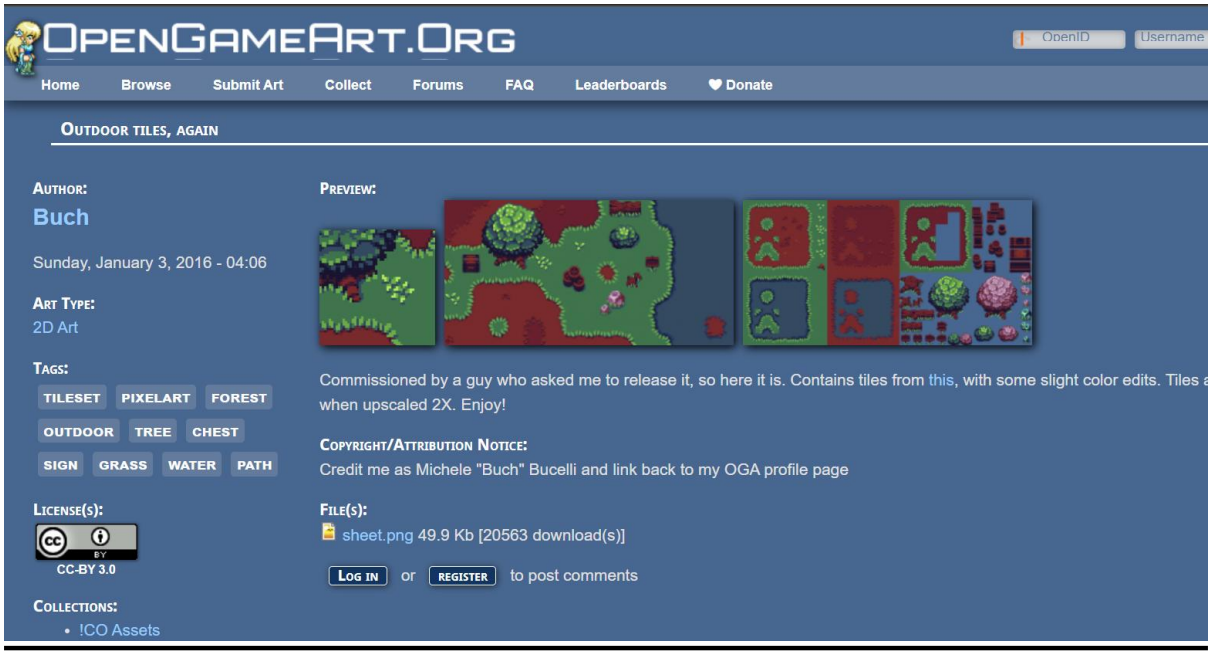


Create another folder named sprites under Assets

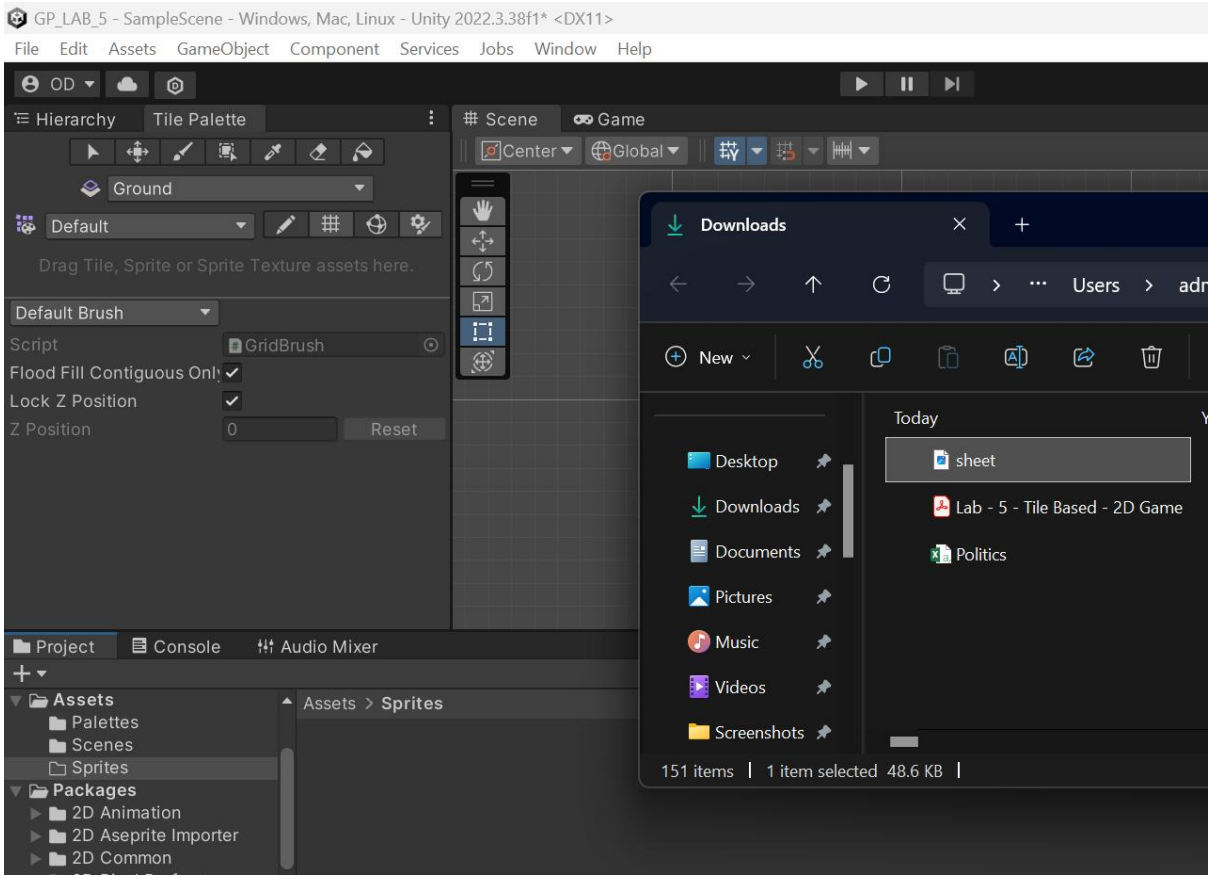




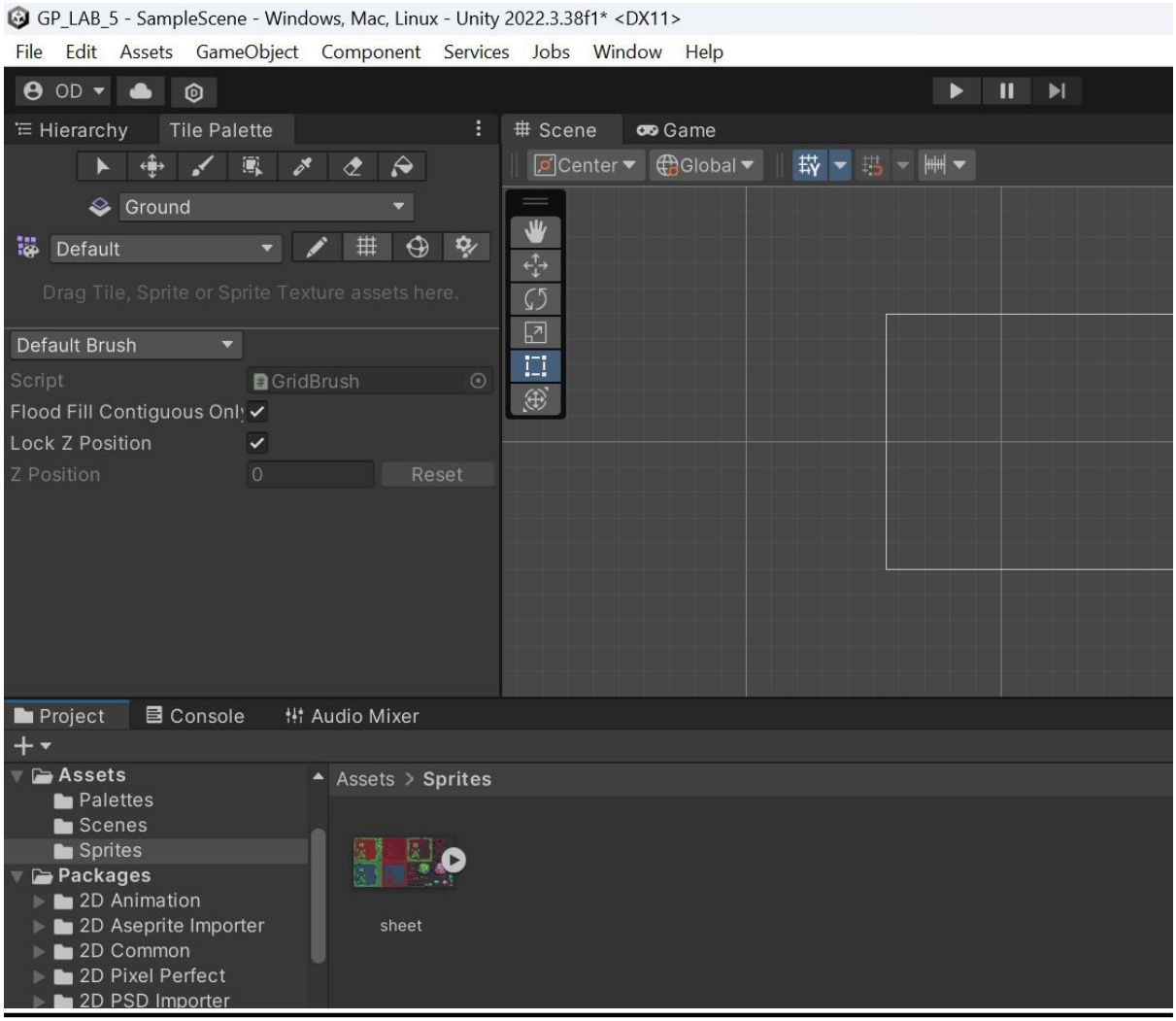
Downloading tilesheet from this respective website



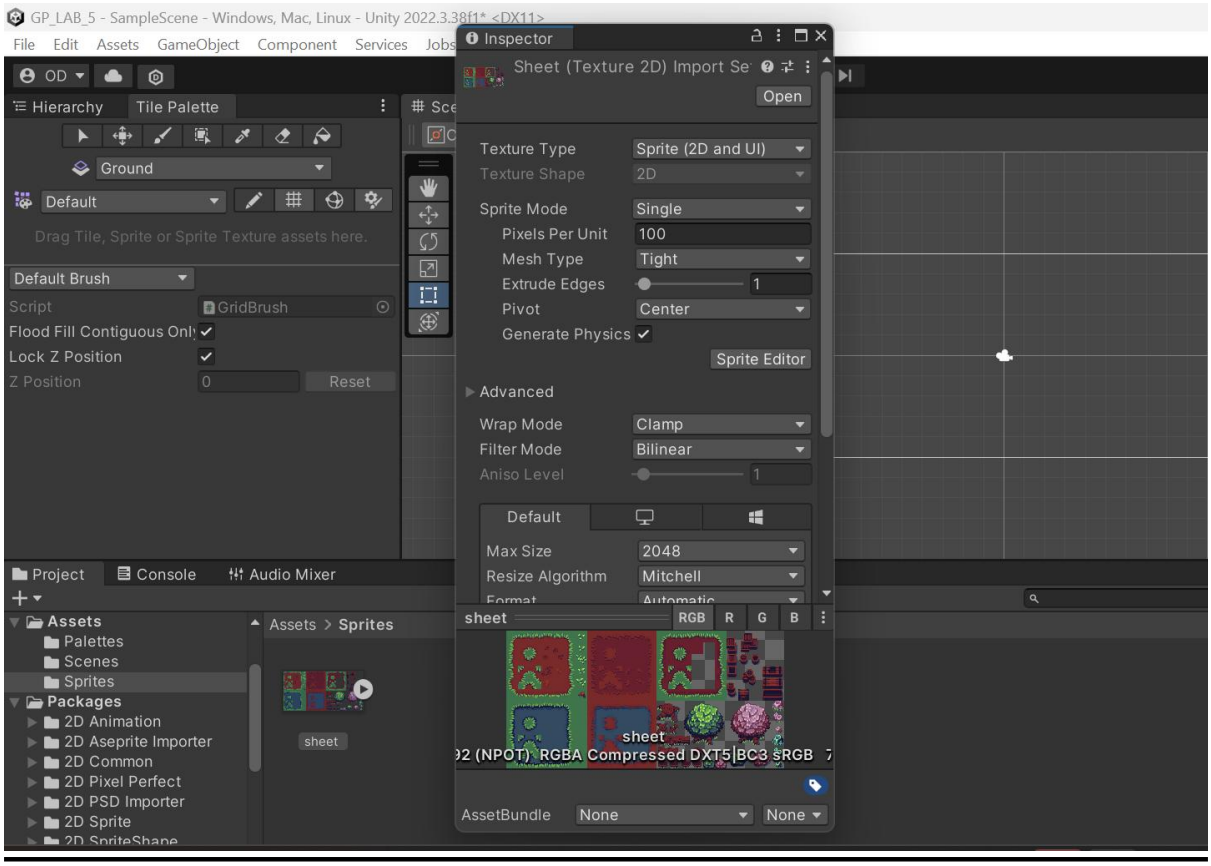
Dragging the sheet.png file into sprites folder



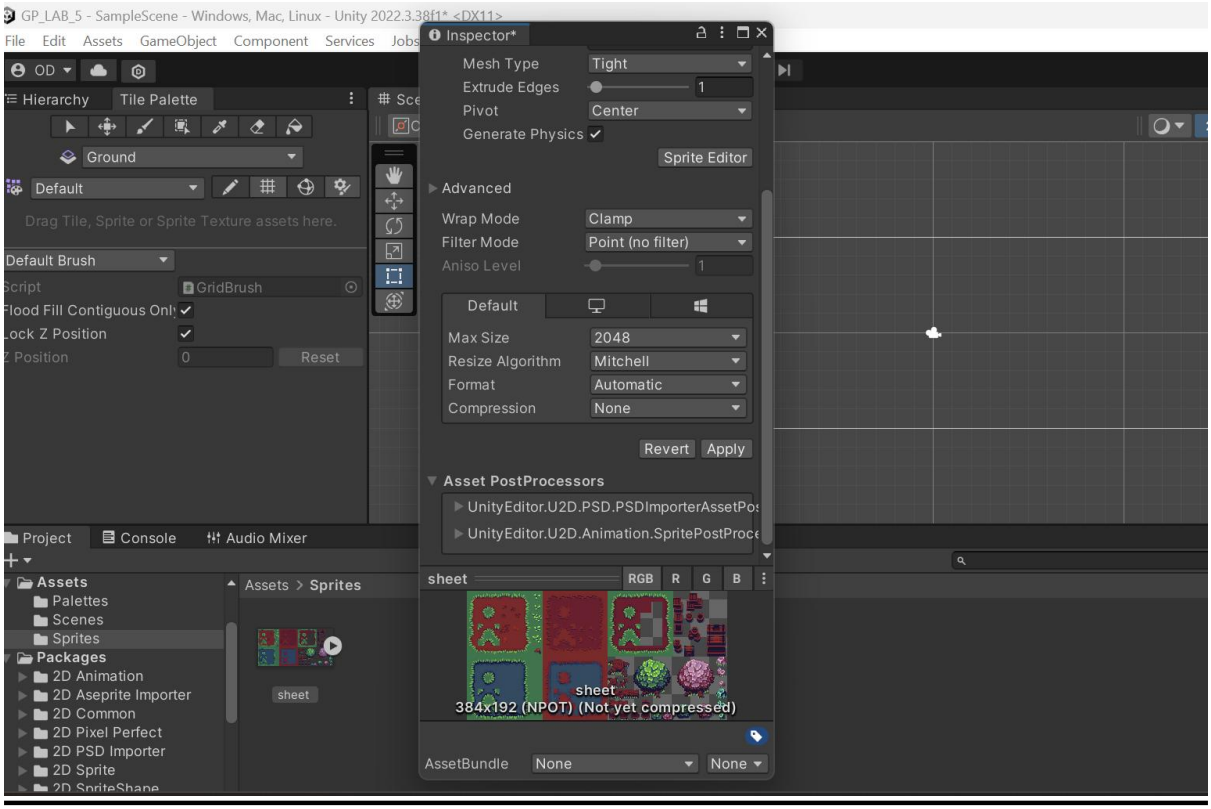
After dragging:



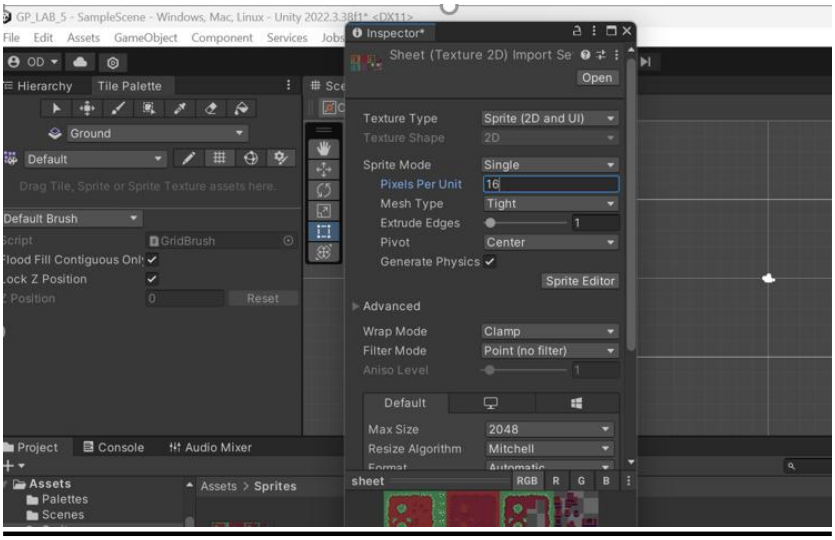
Open inspector window for sprite:



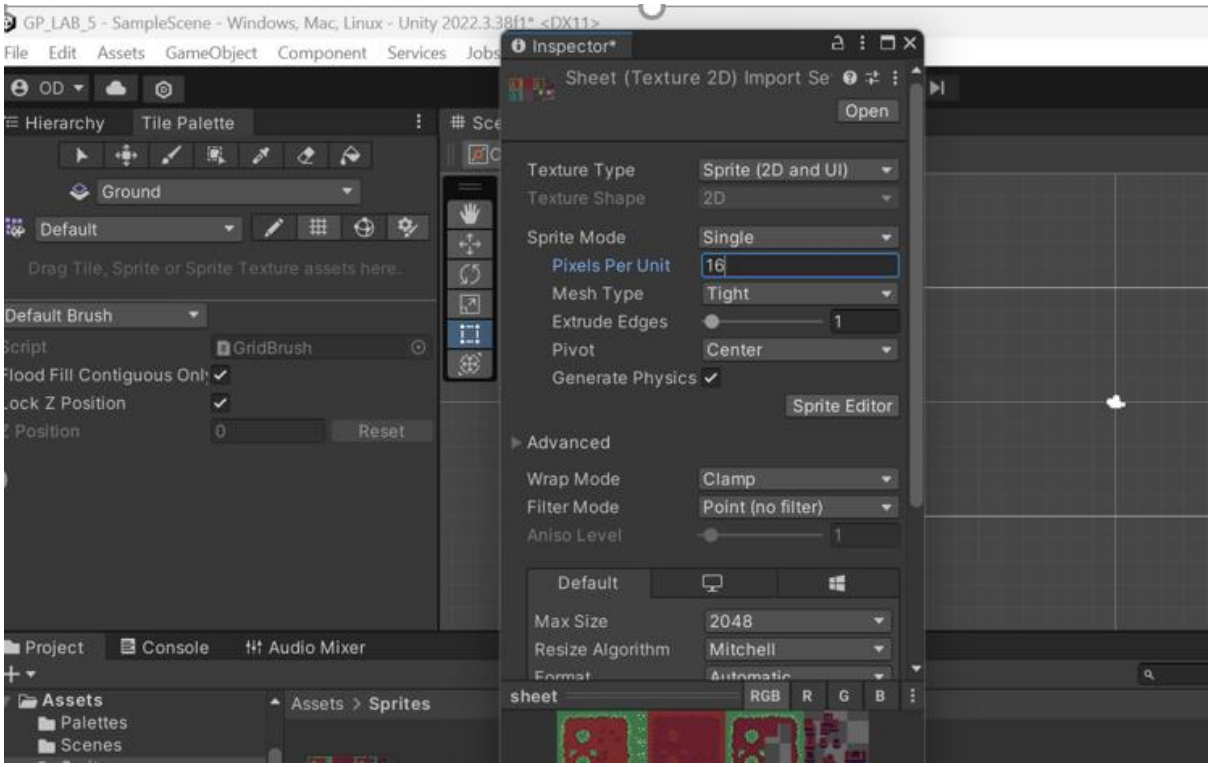
Selecting the Filter mode from Bilinear -> Point (no filter) and the compression from normal quality to none



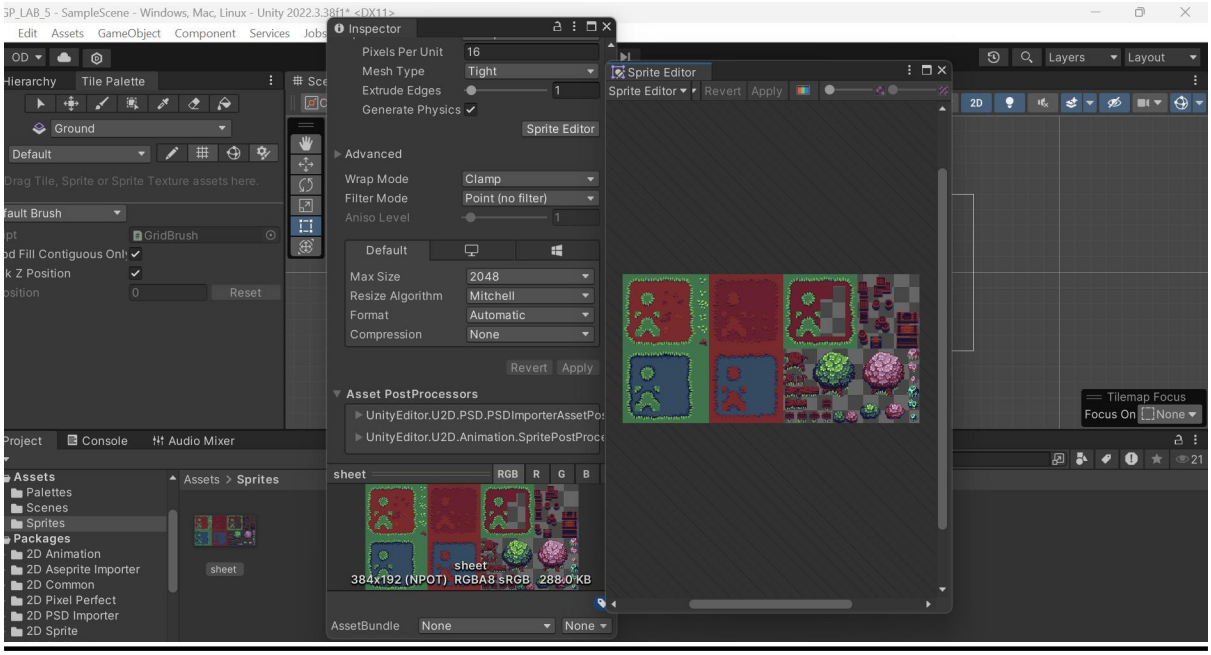
Also change pixels per unit from 100 to 16



Change the sprite mode from single to multiple:

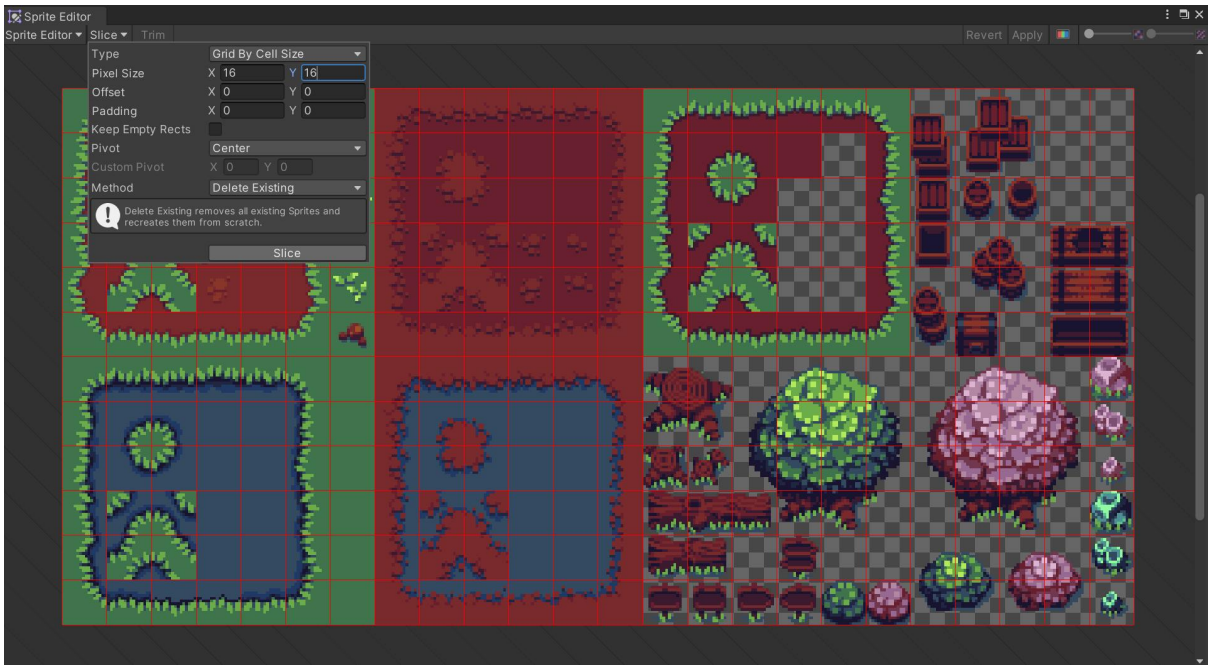


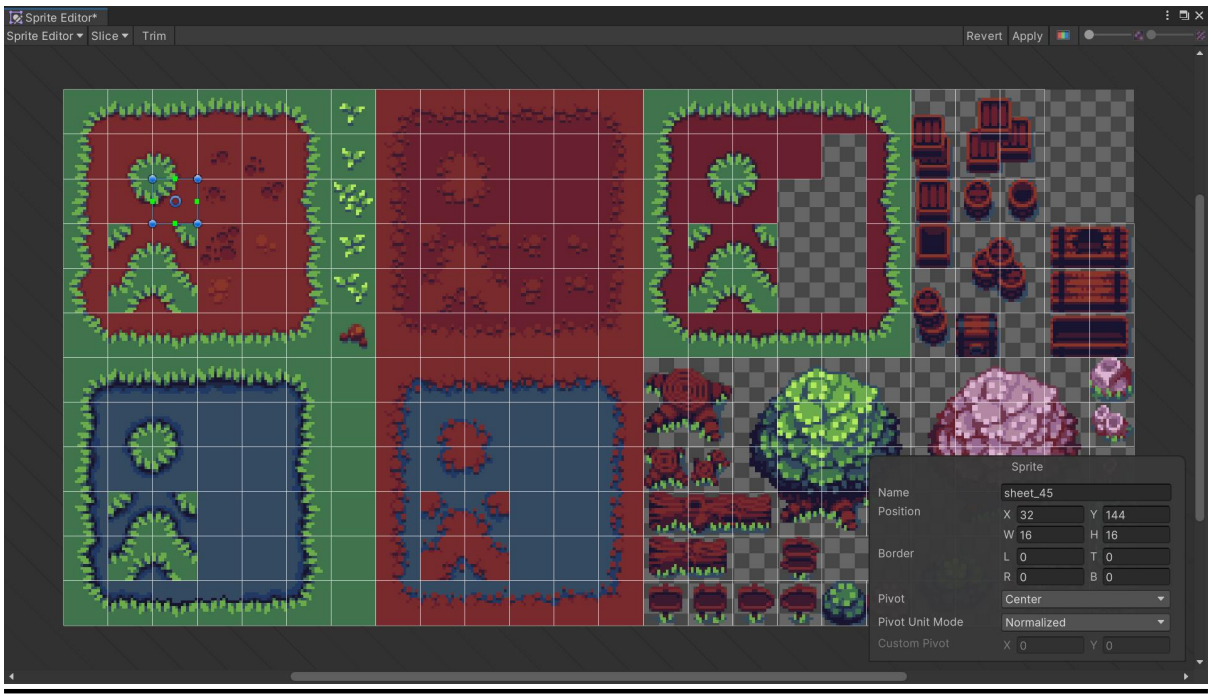
Click Apply and then press Sprite Editor



Maximize the Sprite Editor Window and then select Type as Grid By Cell Size

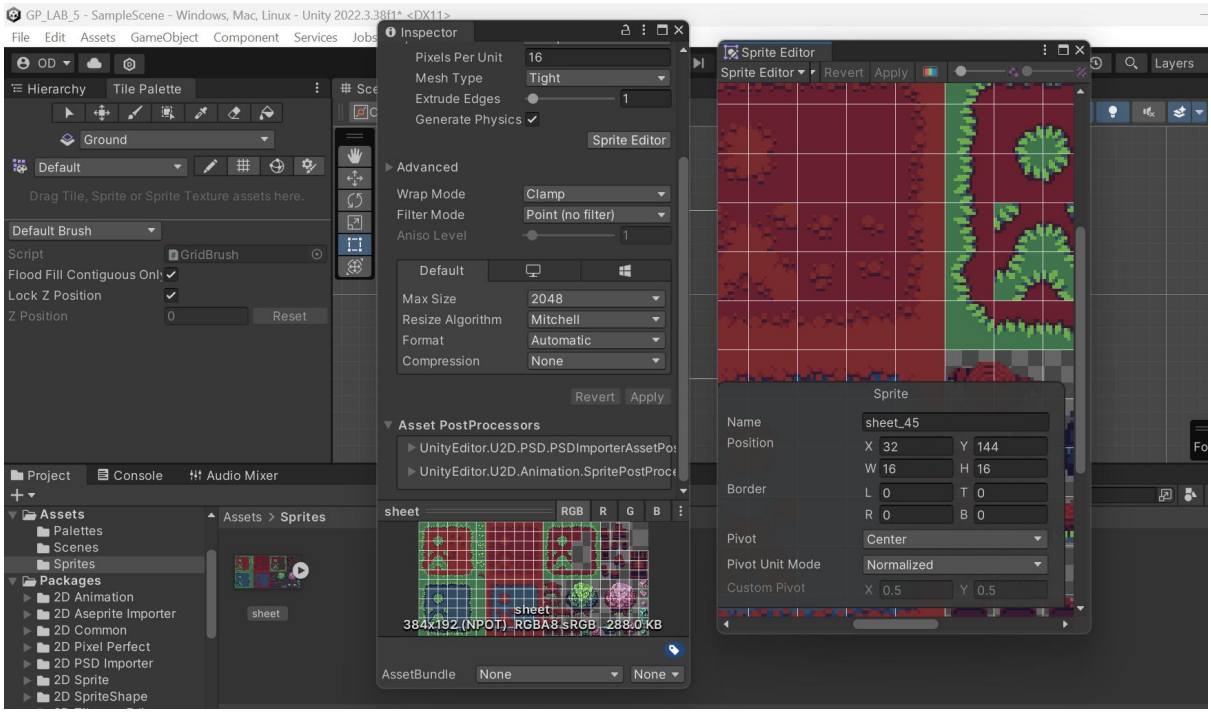
➔ **Also select the pixel size as 16*16 and then click slice**



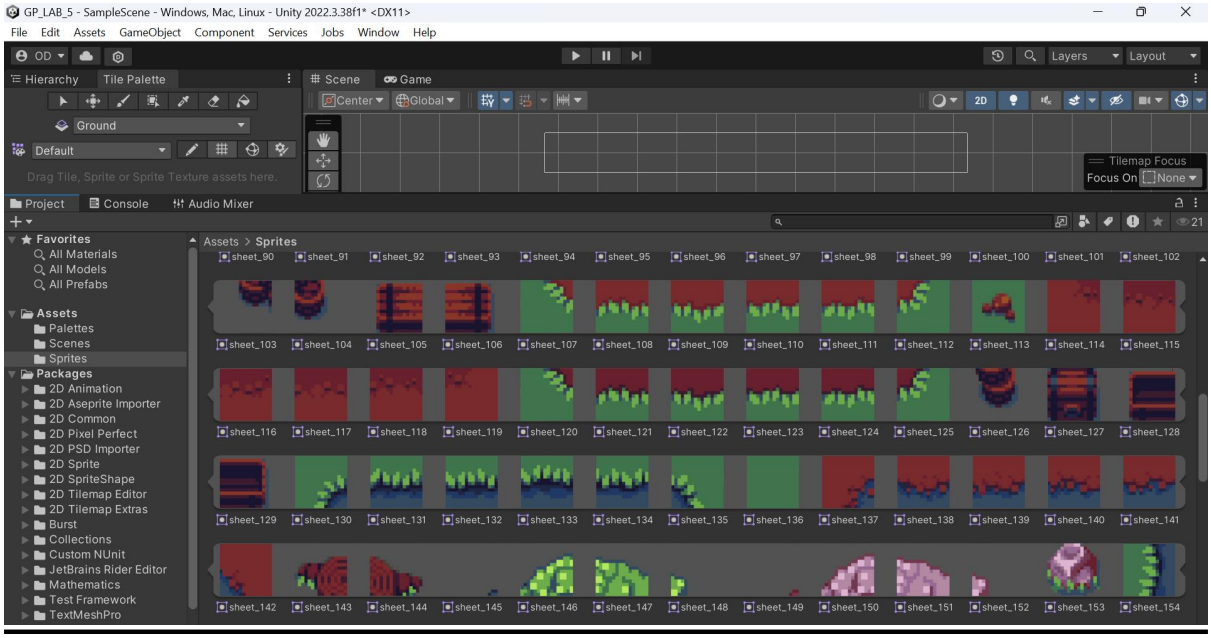


Therefore, the slice gets created respectively.

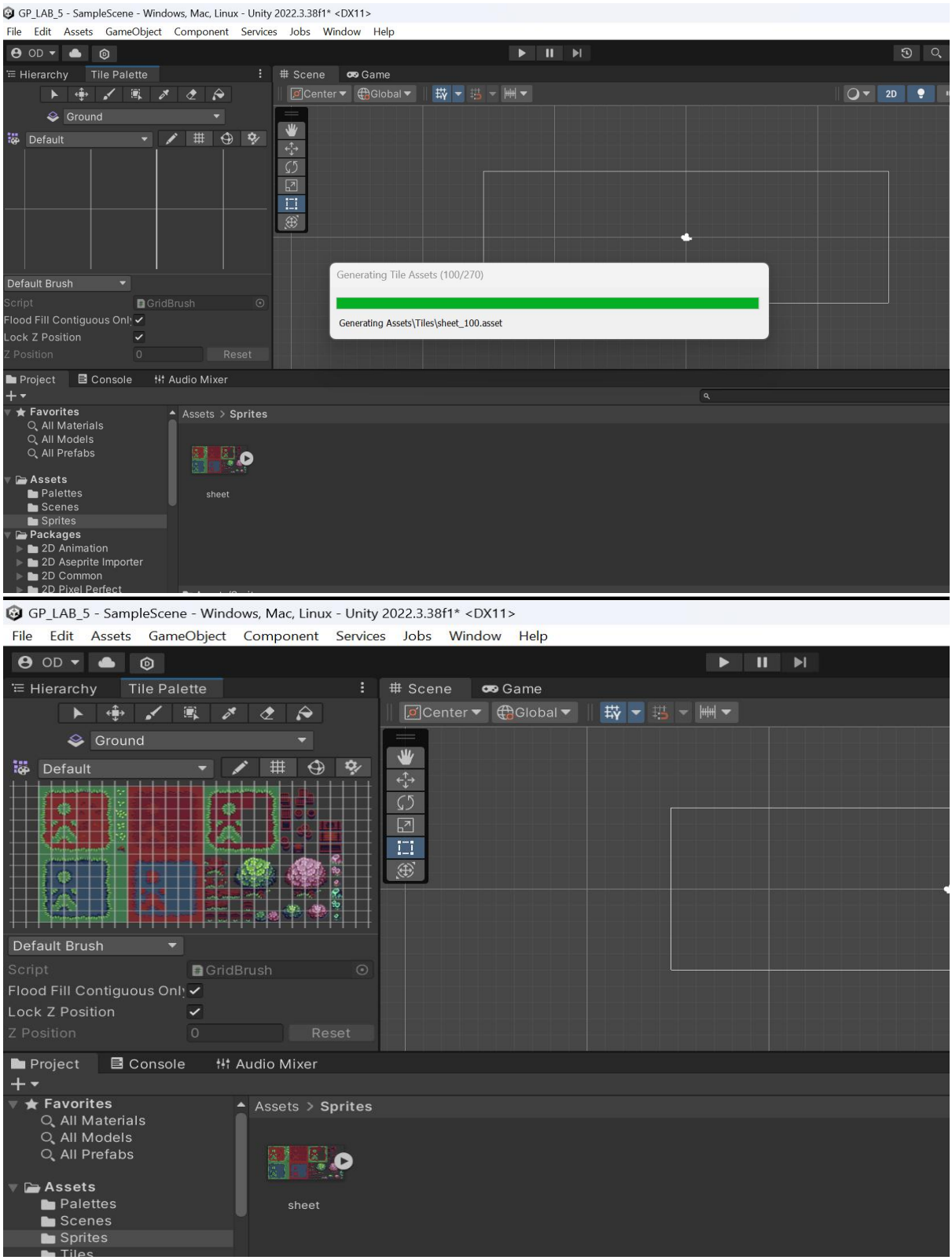
Click Apply and then changes get saved.



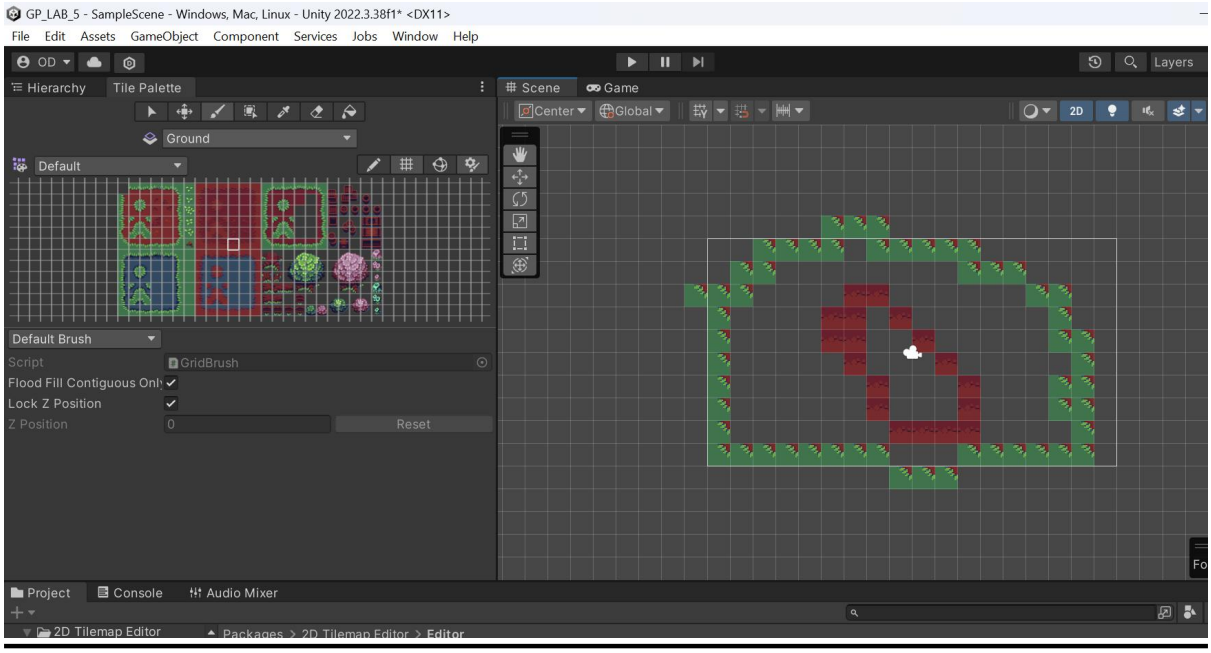
We get all single sprites saved now after this respective operation.



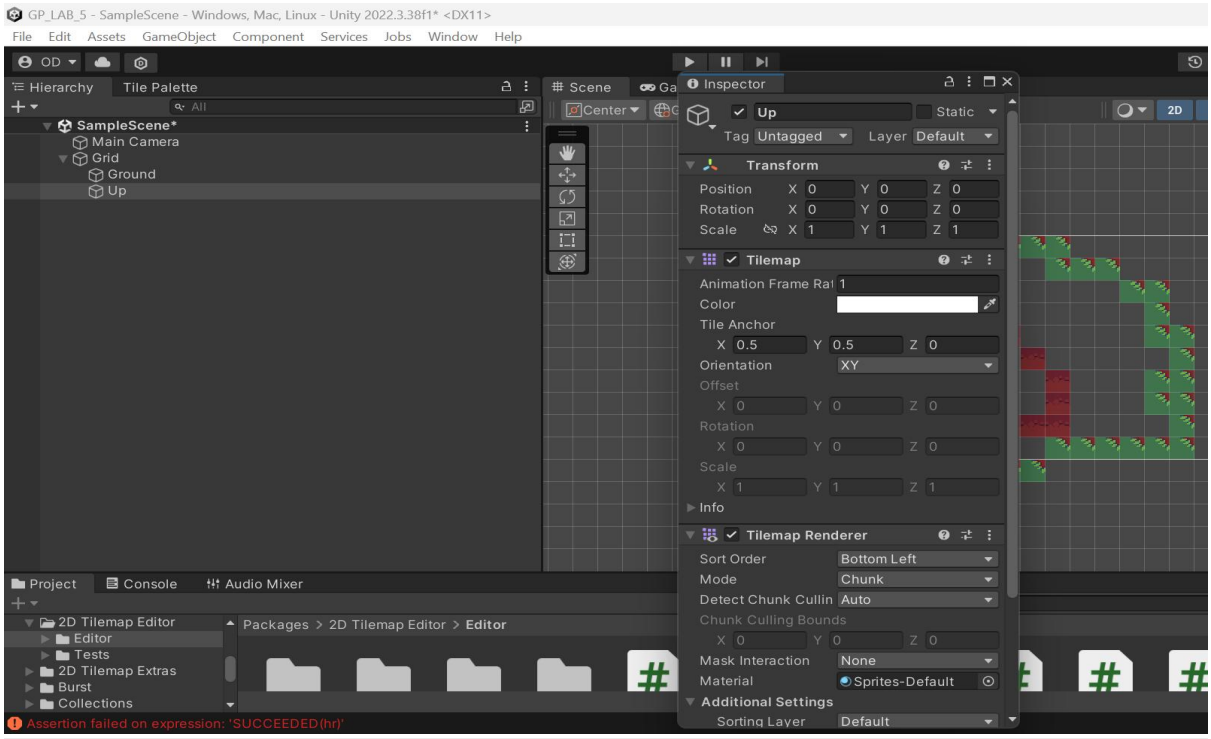
Dragging entire tilemap sheet to the tile palette grid.Then creating a new folder after adding the tile sheet in the grid and then we get a loaded data



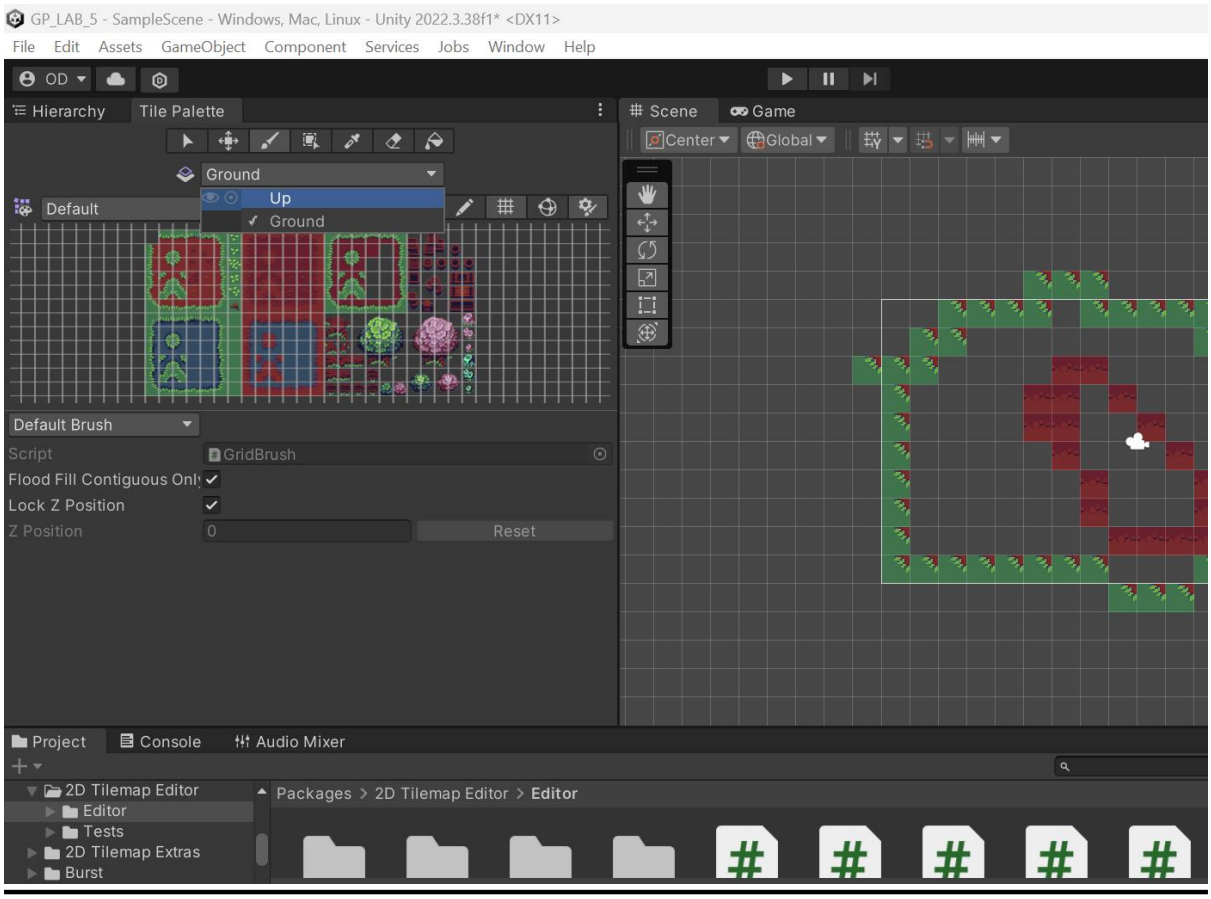
Validation checking to see if the procedure is working or not.



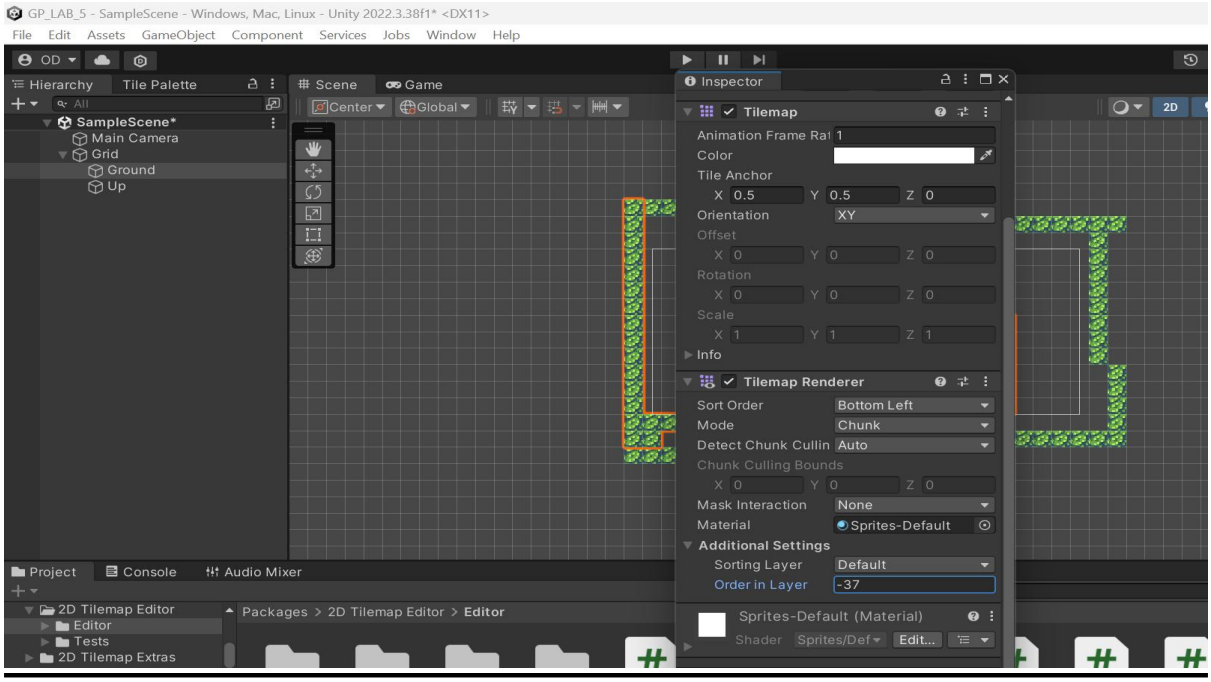
We need to create another tilemap for the purpose so, similarly as before



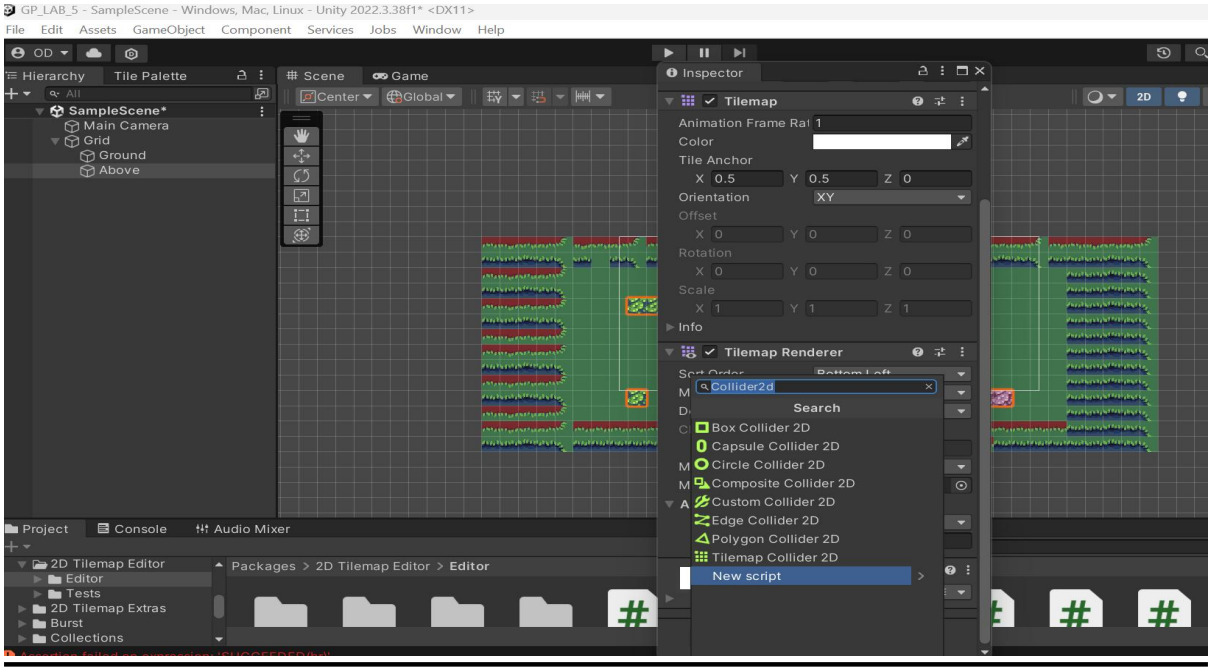
Change to ‘Up’:


GP_LAB_5 - SampleScene - Windows, Mac, Linux - Unity 2022.3.38f1* <DX11>
File Edit Assets GameObject Component Services Jobs Window Help
OD
Hierarchy
Tile Palette
Ground
Up
Ground
Default
Default Brush
Script
GridBrush
Flood Fill Contiguous Only
Lock Z Position
Z Position
0
Reset
Scene
Game
Center
Global
Project
Console
Audio Mixer
2D Tilemap Editor
Editor
Tests
2D Tilemap Extras
Burst
Packages > 2D Tilemap Editor > Editor

Selecting the Up tilemap layer as 0 and the Ground tilemap layer to some suitable negative number.



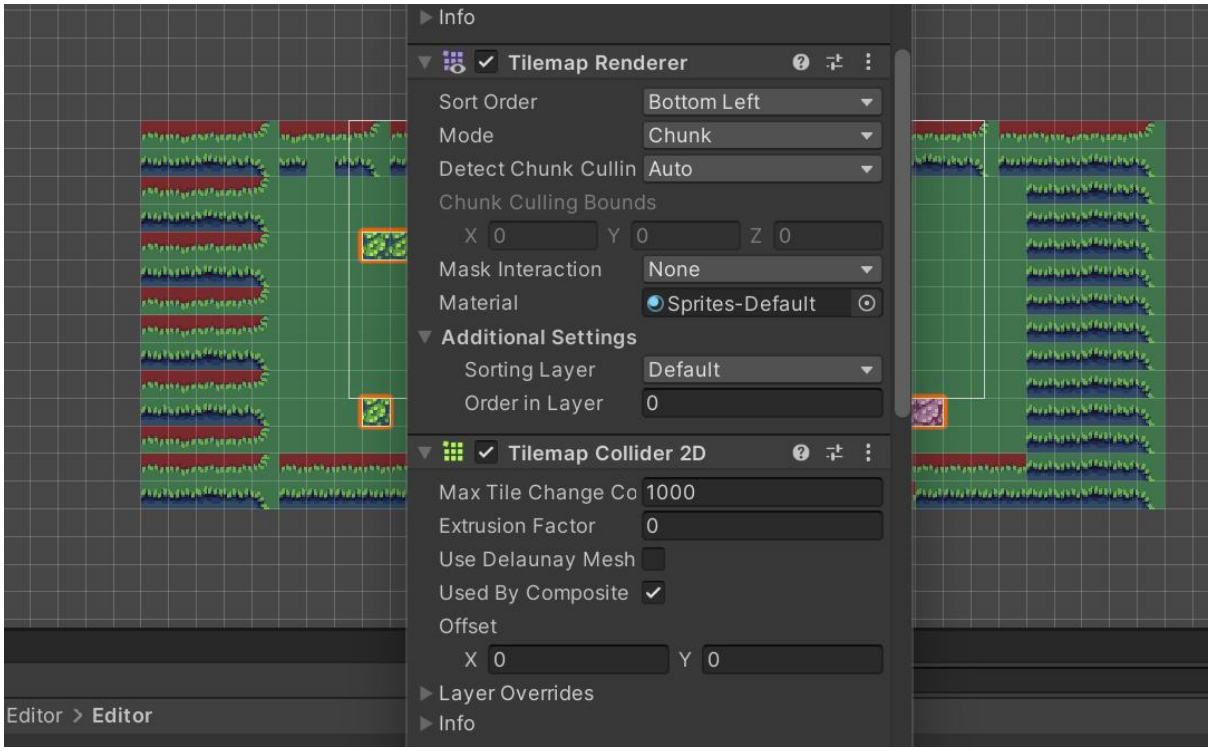
Adding Collider2D component for the Above Tilemap layer



Also add composite collider 2D for the scenario. The Rigidbody 2D automatically gets added as a separate component because the composite collider 2D is dependent on the Rigidbody 2D Component.

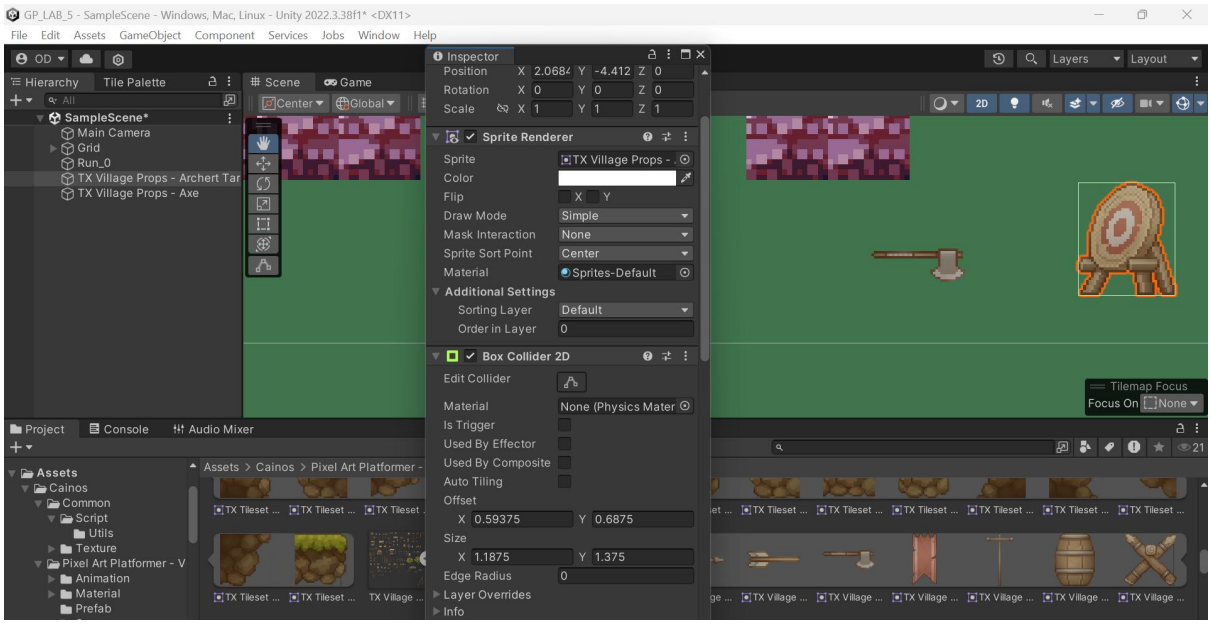
In RigidBody 2D Componet select the body type from dynamic to static.

Select the specs as following:



2.Merging Mechanism:

Downloading suitable assets with objects to show demonstration.



Adding the archery target board and the axe as objects which will be used for merging.

We have taken into consideration the 2 different objects from the asset store package available named ‘Cainos’

We will make the use of the below C# script to perform merging of the two objects.

CODE:

```
using UnityEngine;
public class AxeTargetMerge : MonoBehaviour
{
```

```
    public GameObject axe; // The axe GameObject
    public GameObject targetBoard; // The archery target board
GameObject
    public Transform mergePoint; // The point on the target
board where the axe will merge


    // Optional: Sound or visual effects
    public AudioClip mergeSound;
    public ParticleSystem mergeEffect;


    private bool hasMerged = false;


    void OnCollisionEnter(Collision collision)
    {
        // Check if the axe collided with the target board
        if (collision.gameObject == axe && !hasMerged)
        {
            MergeAxeWithTarget();
        }
    }


    void MergeAxeWithTarget()
    {
        // Disable physics for the axe to make it stick to the
target
        Rigidbody axeRb = axe.GetComponent<Rigidbody>();
        if (axeRb != null)
        {
            axeRb.isKinematic = true;
            axeRb.velocity = Vector3.zero;
            axeRb.angularVelocity = Vector3.zero;
```

```
        }

        // Move the axe to the merge point on the target board
        axe.transform.position = mergePoint.position;
        axe.transform.rotation = mergePoint.rotation;

        // Optional: Play sound and visual effects
        if (mergeSound != null)
        {
            AudioSource.PlayClipAtPoint(mergeSound,
mergePoint.position);
        }
        if (mergeEffect != null)
        {
            Instantiate(mergeEffect, mergePoint.position,
mergePoint.rotation);
        }

        // Set hasMerged to true to prevent multiple merges
        hasMerged = true;
    }
}
```

Add above code as a script component for the objects.

3. Space Management: