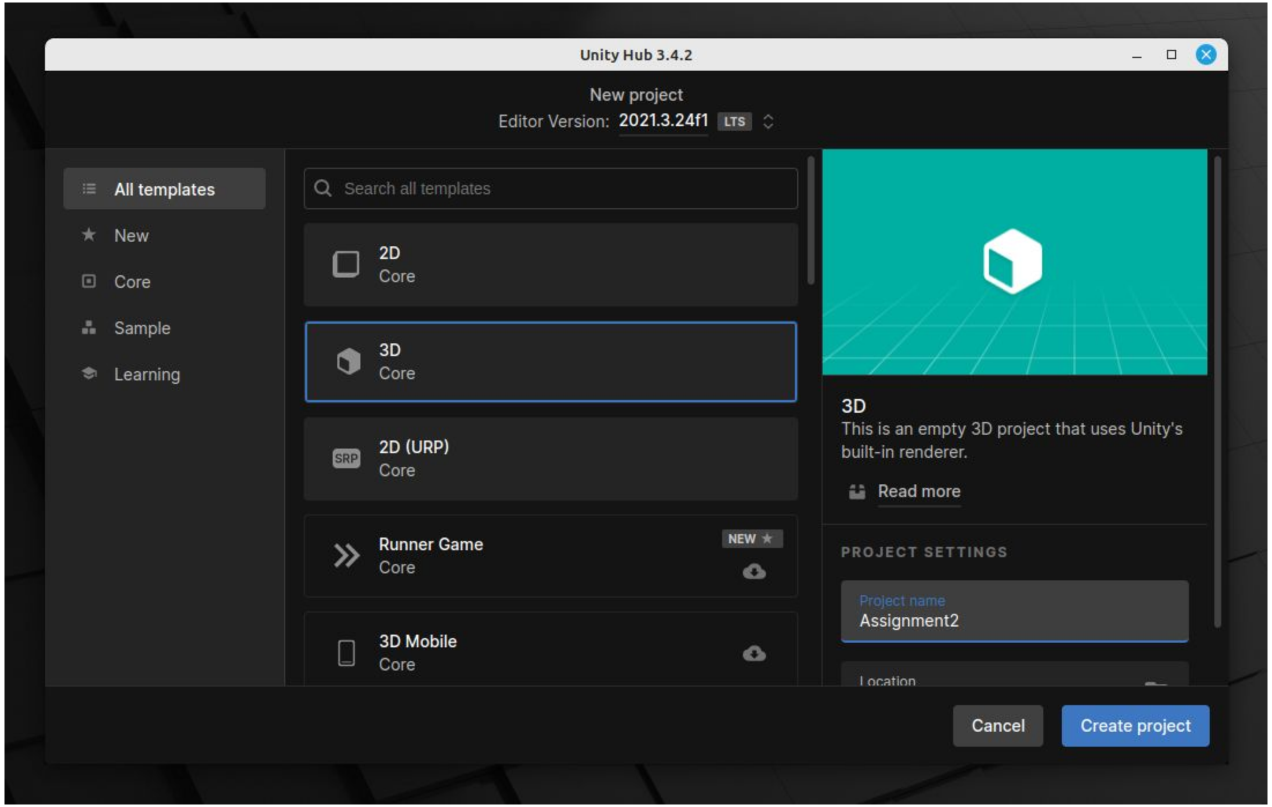
Task 3: Parse an .obj file

3D models can be stored and read in many ways, in this exercise we will focus in the Wavefront obj format. Try to develop a Unity3D powered obj parser. The primary goal is not to write the most performant parser, but to grasp the concepts of vertices, edges and faces.
In order to do that please consider the following example to render vertices and faces as an starting point of your project: https://github.com/nico778/files/MeshGeneration.cs
Also, although the obj format is extremely simple, we suggest you to familiarize yourself with the format: https://en.wikipedia.org/wiki/Wavefront_.obj_file

We suggest you to test your obj parser using the following file. However, feel free to try different 3D models in order to validate the stability of your program:
http://web.mit.edu/djwendel/www/weblogo/shapes/basic-shapes/sphere/sphere.obj

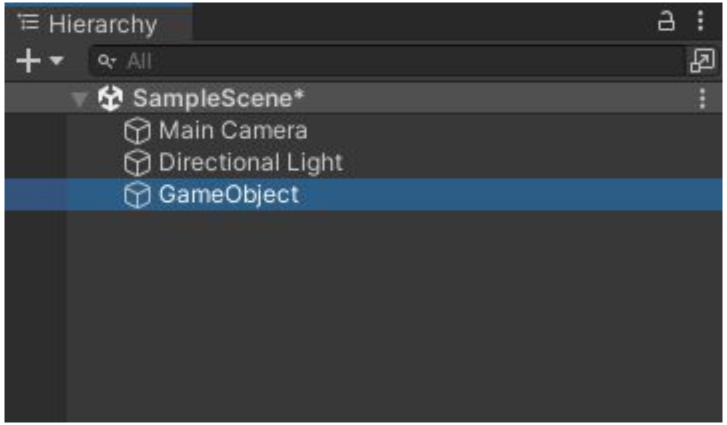
Task 3: Create new project



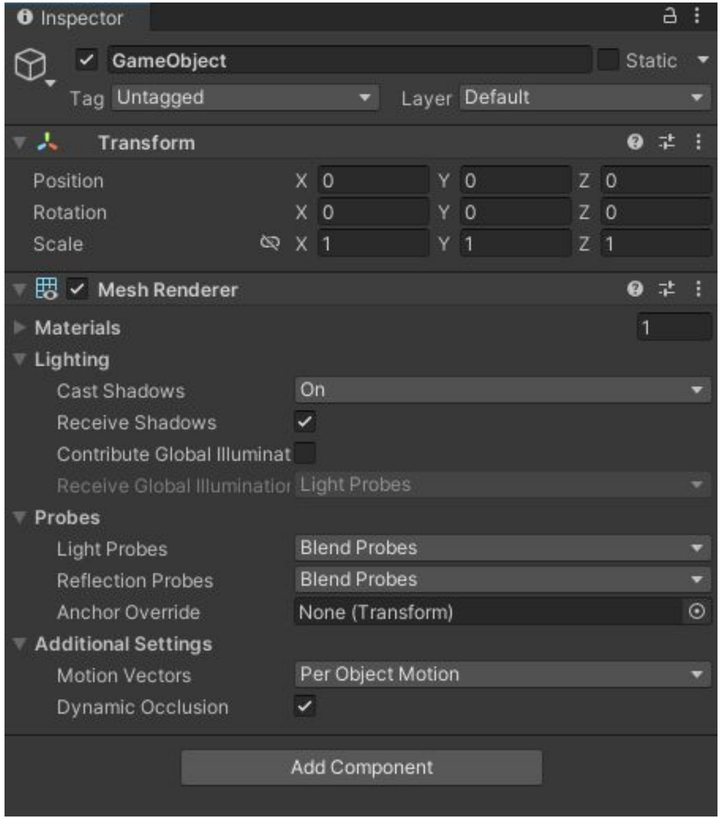
3D
This is an empty 3D project that uses Unity's built-in renderer.

Project name
Assignment2

Task 3: GameObject -> Create Empty



Task 3: Add Component -> MeshRenderer



Inspector

GameObject — Static

Tag: Untagged — Layer: Default

Transform

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

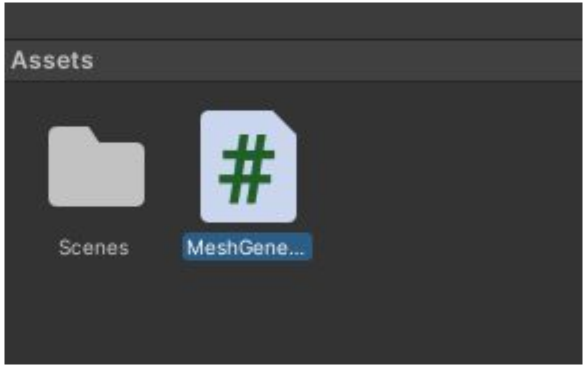
Mesh Renderer

Materials	1
Lighting	
Cast Shadows	On
Receive Shadows	✓
Contribute Global Illuminat	
Receive Global Illuminatio	Light Probes
Probes	
Light Probes	Blend Probes
Reflection Probes	Blend Probes
Anchor Override	None (Transform)
Additional Settings	
Motion Vectors	Per Object Motion
Dynamic Occlusion	✓

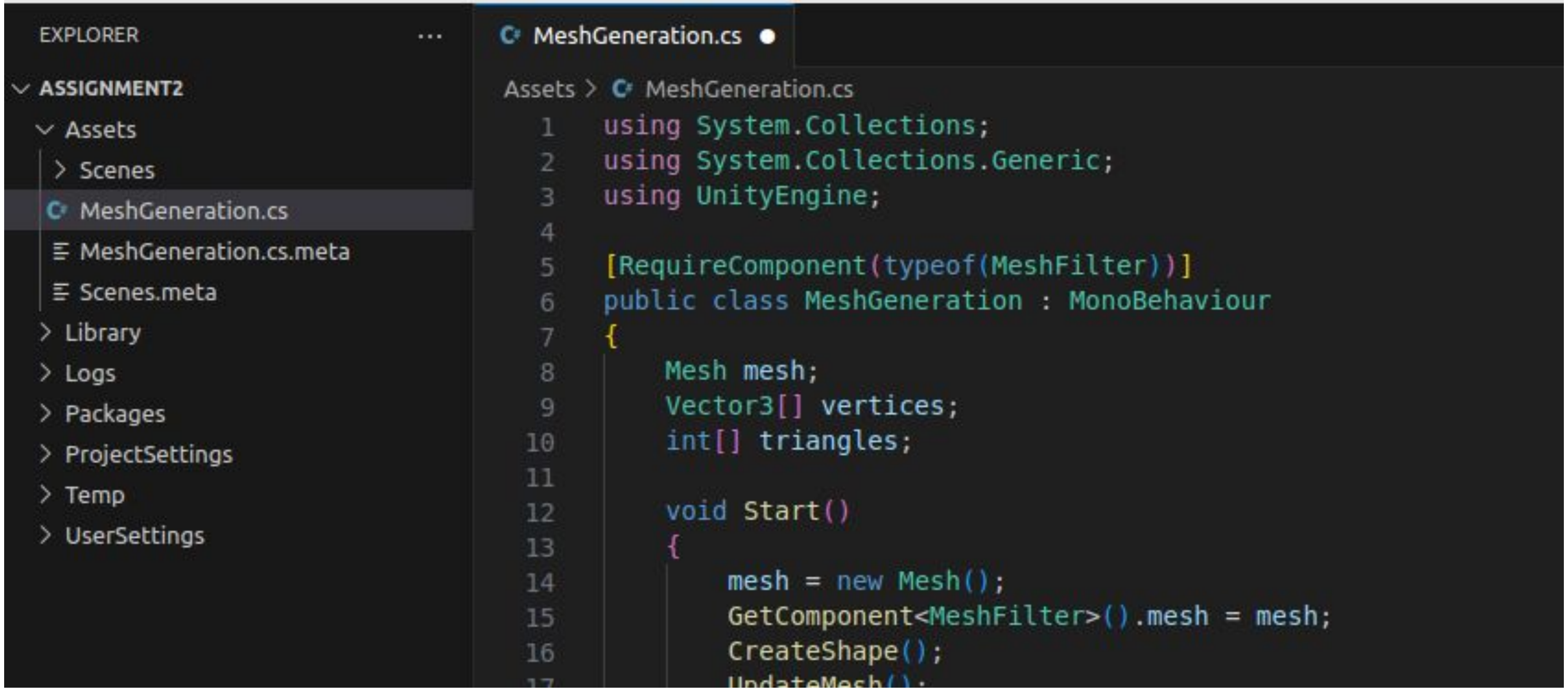
Add Component

Task 3: Assets -> Create -> C# Script

File name and Class name have to be identical!



Task 3: Open project folder in VSCode



```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

[RequireComponent(typeof(MeshFilter))]
public class MeshGeneration : MonoBehaviour
{
    Mesh mesh;
    Vector3[] vertices;
    int[] triangles;

    void Start()
    {
        mesh = new Mesh();
        GetComponent<MeshFilter>().mesh = mesh;
        CreateShape();
```

Task 3: MeshGeneration.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

[RequireComponent(typeof(MeshFilter))]
public class MeshGeneration : MonoBehaviour
{
   Mesh mesh;
   Vector3[] vertices;
   int[] triangles;

   void Start()
   {
       mesh = new Mesh();
       GetComponent<MeshFilter>().mesh = mesh;
       CreateShape();
       UpdateMesh();
   }
```

Task 3: MeshGeneration.cs

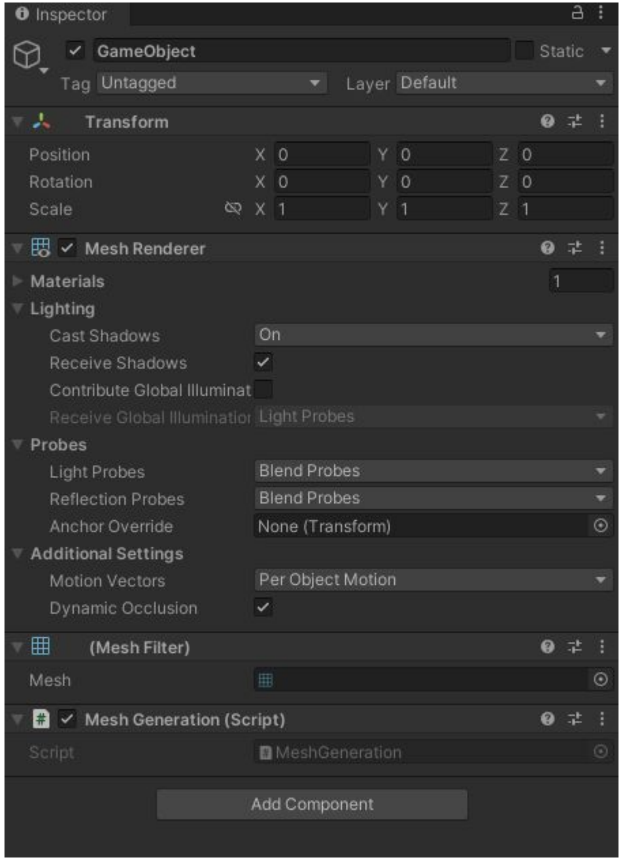
```
void CreateShape()
   {
       vertices = new Vector3[]
       {
           new Vector3(0, 0, 0),
           new Vector3(0, 0, 1),
           new Vector3(1, 0, 0),
           new Vector3(1, 0, 1)
       };

       triangles = new int[]
       {
           0, 1, 2, // First triangle: vertices 0, 1, 2
           2, 1, 3  // Second triangle: vertices 2, 1, 3
       };
   }
```

Task 3: MeshGeneration.cs

```
void UpdateMesh()
   {
       mesh.Clear();

       mesh.vertices = vertices;
       mesh.triangles = triangles;

       mesh.RecalculateNormals();
   }
}
```

Task 3: Add script to GameObject

Drag and drop script from Assets to Inspector



Inspector

✓ GameObject — Static
Tag: Untagged — Layer: Default

Transform

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

✓ **Mesh Renderer**

Materials: 1

Lighting
- Cast Shadows: On
- Receive Shadows: ✓
- Contribute Global Illuminat
- Receive Global Illuminatior: Light Probes

Probes
- Light Probes: Blend Probes
- Reflection Probes: Blend Probes
- Anchor Override: None (Transform)

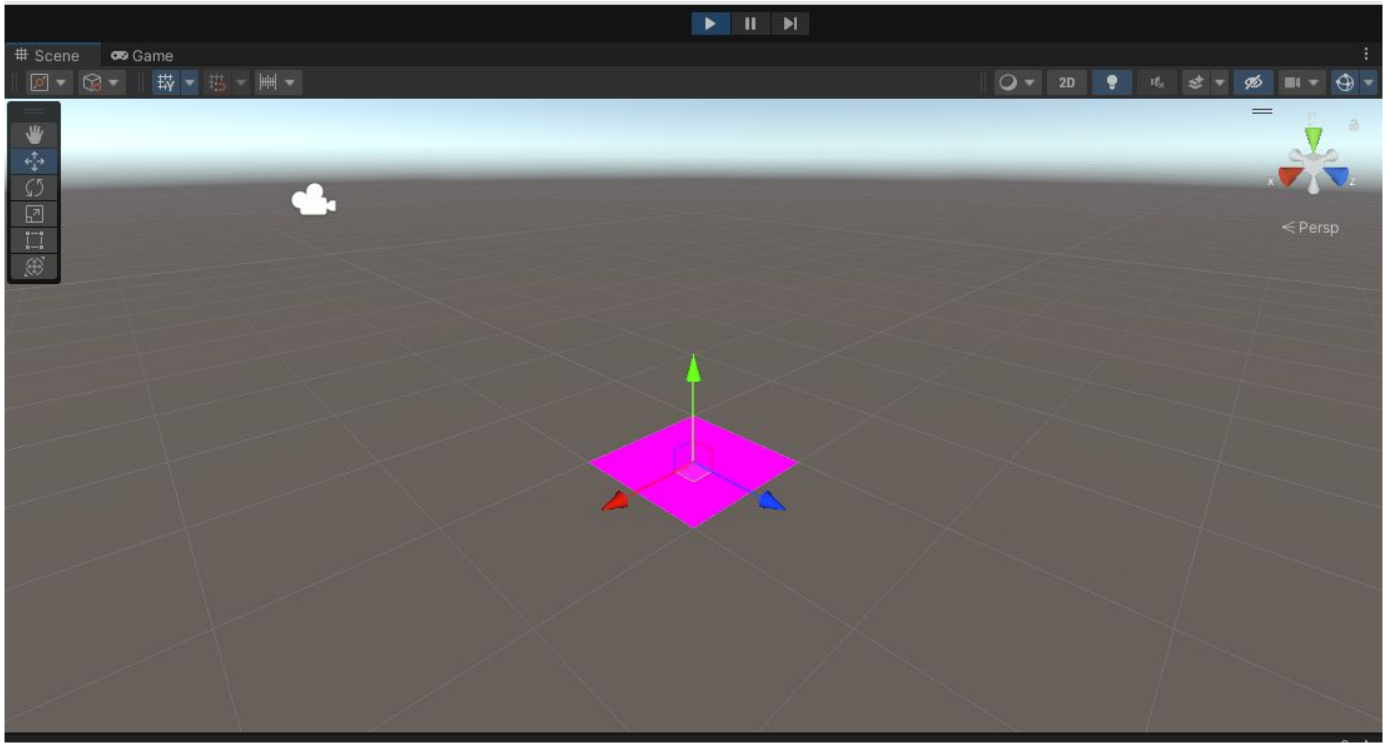
Additional Settings
- Motion Vectors: Per Object Motion
- Dynamic Occlusion: ✓

(Mesh Filter)
- Mesh

✓ **Mesh Generation (Script)**
- Script: MeshGeneration

Add Component

Task 3: Now we can see the two triangles



Now adapt the script to parse an .obj file

- **.obj file parser:**
  - No hardcoded mesh data!
  - Ignore potential texture and normal data in .obj files for this task
  - Get familiar with the wavefront format
  - Sample solution shown later

Now adapt script to parse .obj file

- **.obj file parser:**
    - the `ReadFile` function reads the specified `.obj` file line by line and extracts the vertex and face data.
    - It assumes that the `.obj` file has vertices defined with lines starting with "v " and faces defined with lines starting with "f "
    - To use the script, replace `"/path/to/wherever/your/file/is.obj"` with the actual file path of your `.obj` file.
    - Script assumes that lines containing mesh data are separated by empty lines

Task 3: .obj parser sample solution

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using System.IO;
using System.Globalization;

[RequireComponent(typeof(MeshFilter))]
public class ObjFileParser : MonoBehaviour
{
   Mesh mesh;

   void Start()
   {
       mesh = new Mesh();
       GetComponent<MeshFilter>().mesh = mesh;
       ReadFile("/path/to/wherever/your/file/is.obj");
   }
```

Task 3: ReadFile method

```
void ReadFile(string filePath)
   {
       List<Vector3> vertices = new List<Vector3>();
       List<int> triangles = new List<int>();

       StreamReader reader = new StreamReader(filePath);

       while (!reader.EndOfStream)
       {
           string line = reader.ReadLine().Trim();
           if (line == "") continue; // skip empty lines

           string[] values = line.Split(' ');
```

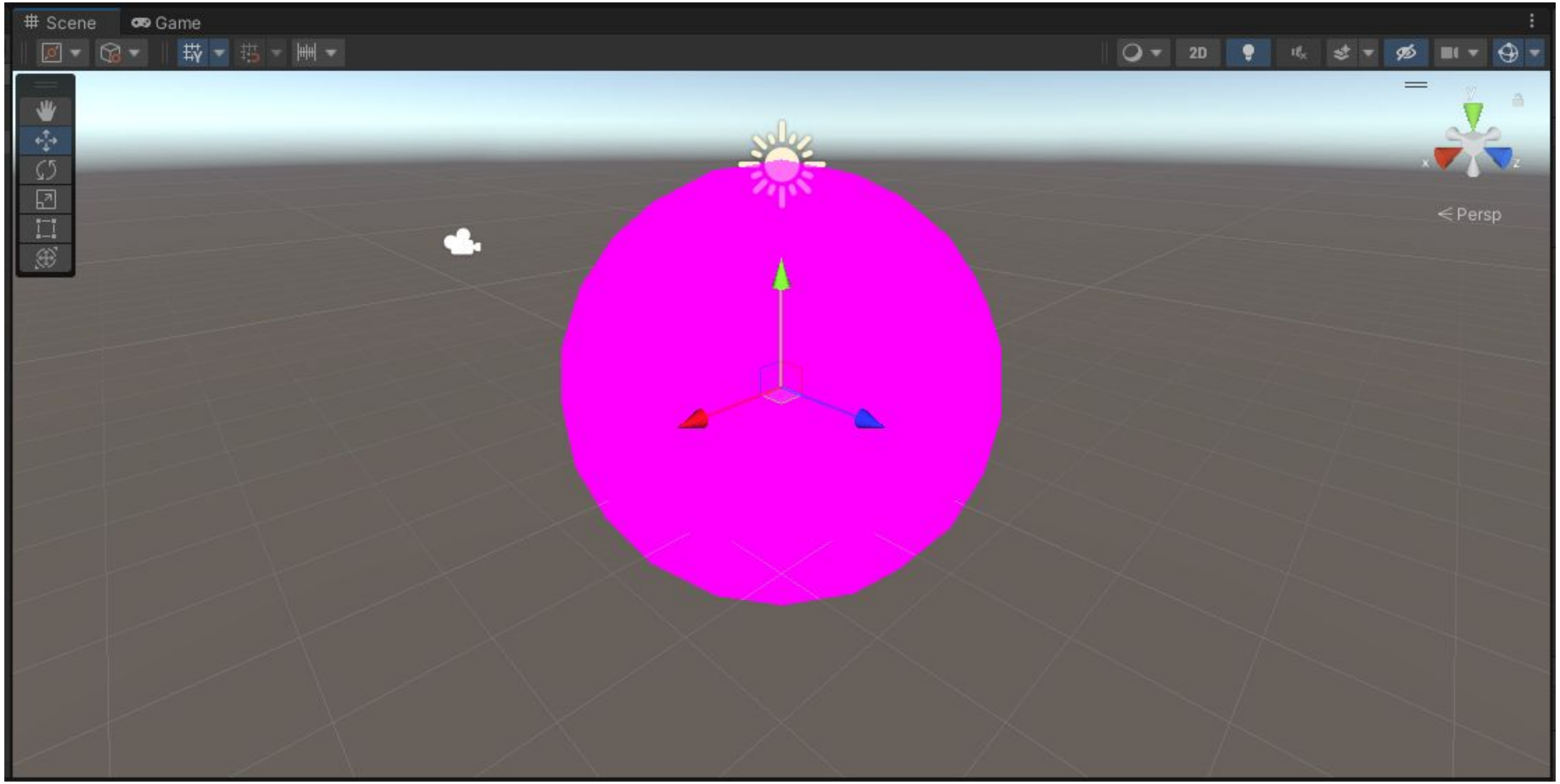
Task 3: ReadFile method

```
if (values[0] == "v")
           {
               float x = float.Parse(values[1], CultureInfo.InvariantCulture);
               float y = float.Parse(values[2], CultureInfo.InvariantCulture);
               float z = float.Parse(values[3], CultureInfo.InvariantCulture);
               vertices.Add(new Vector3(x, y, z));
           }
else if (values[0] == "f")
           {
               int v1 = int.Parse(values[1].Split('/')[0]) - 1;
               int v2 = int.Parse(values[2].Split('/')[0]) - 1;
               int v3 = int.Parse(values[3].Split('/')[0]) - 1;

               triangles.Add(v1);
               triangles.Add(v2);
               triangles.Add(v3);
           }
```

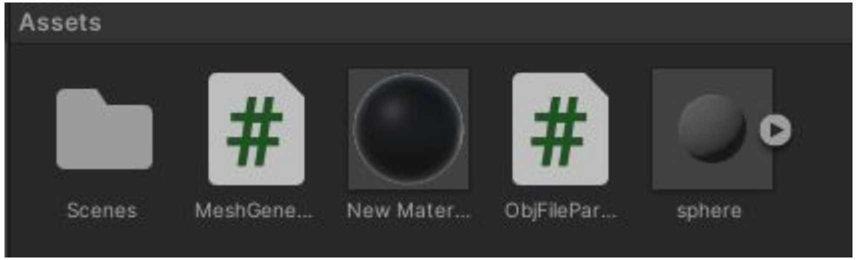
Task 3: ReadFile method

```
}

        reader.Close();

        mesh.Clear();
        mesh.vertices = vertices.ToArray();
        mesh.triangles = triangles.ToArray();
        mesh.RecalculateNormals();
   }
}
```

Task 3: Result



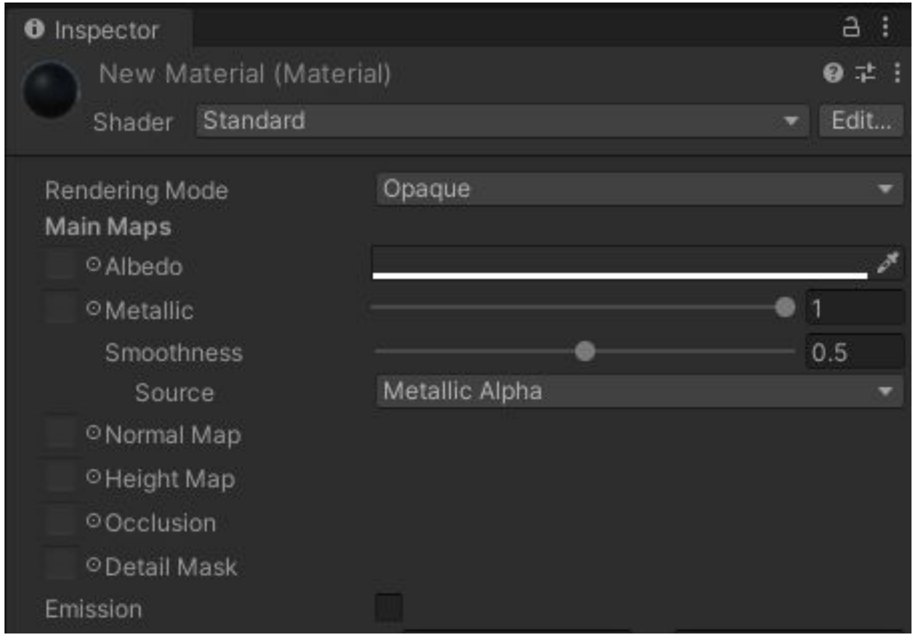
Task 3: Assets -> Create -> Material



Task 3: Material

Change Material Properties according to your liking:

- Metallic: Responsible for light reflectance behaviour



Inspector

New Material (Material)

Shader: Standard

Rendering Mode: Opaque

Main Maps

Albedo

Metallic: 1

Smoothness: 0.5

Source: Metallic Alpha

Normal Map

Height Map

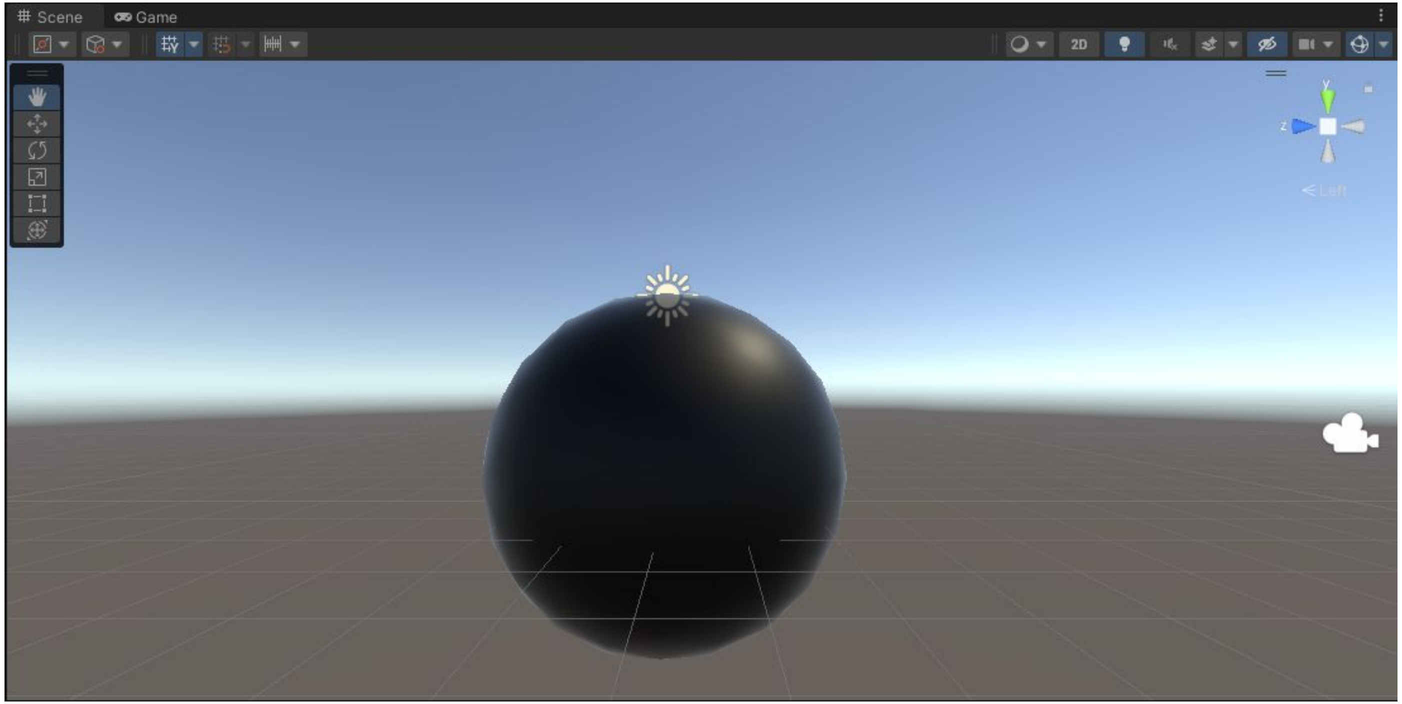
Occlusion

Detail Mask

Emission

Task 3: Material

Drag and drop material into Hierarchy not GameObject Inspector!


Scene
Game
2D
Left