Computer Graphics 1

Tutorial

Assignment 2

Summer Semester 2024

Ludwig-Maximilians-Universität München

Contact

If you have any questions:

cg1ss24@medien.ifi.lmu.de

Organization

- **Theoretical part:**
  - Prepares you for the exam
  - Solve the tasks at home
  - We present the solutions during the tutorial sessions
- **Practical part:**
  - We will indicate which parts you need to do at home
  - Else this will be done & explained during the tutorial session
  - Ask questions!

Task 1A: Vector Addition

1. **Given Vectors**

$$\vec{a} = (2, 3, 1)^T$$
$$\vec{b} = (5, 1, -2)^T$$

2. **Compute Addition**

$$\vec{a} + \vec{b} = \begin{pmatrix} 2 \\ 3 \\ 1 \end{pmatrix} + \begin{pmatrix} 5 \\ 1 \\ -2 \end{pmatrix} = \begin{pmatrix} 2+5 \\ 3+1 \\ 1-2 \end{pmatrix} = \begin{pmatrix} 7 \\ 4 \\ -1 \end{pmatrix}$$

Task 1B: Vector Subtraction

1. Scaling

$$2\vec{a} = 2 \times \begin{pmatrix} 2 \\ 3 \\ 1 \end{pmatrix} = \begin{pmatrix} 4 \\ 6 \\ 2 \end{pmatrix}$$

$$3\vec{b} = 3 \times \begin{pmatrix} 5 \\ 1 \\ -2 \end{pmatrix} = \begin{pmatrix} 15 \\ 3 \\ -6 \end{pmatrix}$$

2. Subtraction

$$2\vec{a} - 3\vec{b} = \begin{pmatrix} 4 \\ 6 \\ 2 \end{pmatrix} - \begin{pmatrix} 15 \\ 3 \\ -6 \end{pmatrix} = \begin{pmatrix} 4-15 \\ 6-3 \\ 2+6 \end{pmatrix} = \begin{pmatrix} -11 \\ 3 \\ 8 \end{pmatrix}$$

Task 1C: Dot Product

- **Given Vectors**

$$\vec{a} = (2, 3, 1)^T$$
$$\vec{b} = (5, 1, -2)^T$$

- **Formula**

$$\vec{a} \cdot \vec{b} = a_x b_x + a_y b_y + a_z b_z$$

$$\boxed{\vec{a} \cdot \vec{b} = (2 \cdot 5) + (3 \cdot 1) + (1 \cdot -2) = 10 + 3 - 2 = 11}$$

Task 1D: Cross Product

- **Given Vectors**

$$\vec{a} = (2, 3, 1)$$
$$\vec{b} = (5, 1, -2)$$

- **Compute for each component of the resulting vector**

$$a_y b_z - a_z b_y = 3 \times (-2) - 1 \times 1 = -6 - 1 = -7$$
$$a_z b_x - a_x b_z = 1 \times 5 - 2 \times (-2) = 5 + 4 = 9$$
$$a_x b_y - a_y b_x = 2 \times 1 - 3 \times 5 = 2 - 15 = -13$$

$$\boxed{(-7, 9, -13)}$$

Task 1E: Magnitude

- **Formula** $$\|\vec{a}\| = \sqrt{a_x^2 + a_y^2 + a_z^2}$$

- **Calculate the magnitude**

$$\boxed{\|\vec{a}\| = \sqrt{2^2 + 3^2 + 1^2} = \sqrt{4 + 9 + 1} = \sqrt{14}}$$

Task 1F: $\vec{a}^T \cdot \vec{b}$

- **As Dot Product:** same as Task 1C

- **As Matrix Multiplication**

$$\vec{a} \cdot \vec{b}^T = \begin{bmatrix} 2 \\ 3 \\ 1 \end{bmatrix} \cdot \begin{bmatrix} 5 & 1 & -2 \end{bmatrix} = \begin{bmatrix} 2 \times 5 & 2 \times 1 & 2 \times -2 \\ 3 \times 5 & 3 \times 1 & 3 \times -2 \\ 1 \times 5 & 1 \times 1 & 1 \times -2 \end{bmatrix}$$

$$= \begin{bmatrix} 10 & 2 & -4 \\ 15 & 3 & -6 \\ 5 & 1 & -2 \end{bmatrix}$$

Task 1G: $\vec{a} \cdot \vec{b}^T$

- **As Matrix Multiplication:**
  - Results in a 1x1 matrix, essentially a scalar, => the same as Task 1C

Task 1H: Matrix determinant

- **Formula(3x3 Matrix):** $\det(B) = b_{11}(b_{22}b_{33} - b_{23}b_{32}) - b_{12}(b_{21}b_{33} - b_{23}b_{31}) + b_{13}(b_{21}b_{32} - b_{22}b_{31})$
- **Given Matrix:**

$$B = \begin{bmatrix} 2 & 1 & 1 \\ 2 & 2 & 3 \\ 3 & -1 & 2 \end{bmatrix}$$

- **Compute determinant:**

$$\begin{aligned} \det(B) &= 2(2 \cdot 2 - 3 \cdot (-1)) - 1(2 \cdot 2 - 3 \cdot 3) + 1(2 \cdot (-1) - 2 \cdot 3) \\ &= 2(4 + 3) - 1(4 - 9) + 1(-2 - 6) \\ &= 2(7) - 1(-5) - 8 \\ &= 14 + 5 - 8 \\ &= 11 \end{aligned}$$

Task 1H: Matrix Product

- **Matrix Dimensions:**
  - Matrix A has dimensions 3×2 (3 rows, 2 columns)
  - Matrix B has dimensions 3×3 (3 rows, 3 columns)

- **Multiplying AB:** To multiply two matrices, the number of columns in the first matrix must match the number of rows in the second matrix. In the case of *AB*:
  - Matrix *A* (3x2) can not be multiplied by Matrix *B* (3x3) because their dimensions are not compatible (2 columns in *A* do not match 3 rows in *B*)

- **Multiplying BA:**
  - Matrix B (3x3) can multiply Matrix A (3x2) because the number of columns in B (3) matches the number of rows in A (3). The result will be a 3×2 matrix.

Task 1H: Matrix Product

- **The product BA results in a 3×2 matrix where each element is the dot product of the corresponding row of B and column of A:**

$$\text{Element } (i,j) \text{ of } BA = \sum_{k=1}^{3} B_{ik} A_{kj}$$

- **Calculation:**

Entry $(1,1)$ **of** BA**:**

$c_{11} = 2 \times 4 + 1 \times 2 + 1 \times 1 = 8 + 2 + 1 = 11$

Entry $(1,2)$ **of** BA**:**

$c_{12} = 2 \times 1 + 1 \times -2 + 1 \times -1 = 2 - 2 - 1 = -1$

Entry $(2,1)$ **of** BA**:**

$c_{21} = 2 \times 4 + 2 \times 2 + 3 \times 1 = 8 + 4 + 3 = 15$

Entry $(2,2)$ **of** BA**:**

$c_{22} = 2 \times 1 + 2 \times -2 + 3 \times -1 = 2 - 4 - 3 = -5$

Entry $(3,1)$ **of** BA**:**

$c_{31} = 3 \times 4 - 1 \times 2 + 2 \times 1 = 12 - 2 + 2 = 12$

Entry $(3,2)$ **of** BA**:**

$c_{32} = 3 \times 1 - 1 \times -2 + 2 \times -1 = 3 + 2 - 2 = 3$

$$BA = \begin{bmatrix} 11 & -1 \\ 15 & -5 \\ 12 & 3 \end{bmatrix}$$

Task 2A: Affine Transformations

- **Introduction to Homogeneous Coordinates**
  - Homogeneous coordinates are a system of coordinates used in computer graphics, which allows for the convenient representation of points at infinity and simplifies mathematical formulas when applying transformations such as translations, rotations, and scaling.

- **Representation of 3D Points**
  - In standard Cartesian coordinates, a point in 3D space is represented as (X, Y, Z).
  - In homogeneous coordinates, an extra dimension (W) is added, and the point is represented as (X, Y, Z, W).
  - For typical 3D operations, W is set to 1. Thus, a 3D point (X, Y, Z) becomes (X, Y, Z, 1) in homogeneous coordinates.

- p1=(4,6,1)
- p2=(4,1,1)

Task 2B: Affine Transformations

- **Translation Matrix in Homogeneous Coordinates:** translating a point by (dx,dy):

$$M = \begin{bmatrix} 1 & 0 & dx \\ 0 & 1 & dy \\ 0 & 0 & 1 \end{bmatrix} \qquad M = \begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{bmatrix}$$

- **Translating point p1:**

$$p1' = M \times p1 = \begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 4 \\ 6 \\ 1 \end{bmatrix} = \begin{bmatrix} 4-1 \\ 6+2 \\ 1 \end{bmatrix} = \begin{bmatrix} 3 \\ 8 \\ 1 \end{bmatrix}$$

- **Translating point p2:**

$$p2' = M \times p2 = \begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 4 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 4-1 \\ 1+2 \\ 1 \end{bmatrix} = \begin{bmatrix} 3 \\ 3 \\ 1 \end{bmatrix}$$

Task 2B: Affine Transformations

- **Translation Matrix in Homogeneous Coordinates:** translating a point by (dx,dy):

$$M = \begin{bmatrix} 1 & 0 & dx \\ 0 & 1 & dy \\ 0 & 0 & 1 \end{bmatrix} \qquad M = \begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{bmatrix}$$

- **Translating point p (x, y, 1):**

$$p' = Mp = \begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x-1 \\ y+2 \\ 1 \end{bmatrix}$$

Task 2B: Affine Transformations

- **Scaling Matrix in Homogeneous Coordinates:**

$$M = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- **Specific Scaling Matrix:**

$$M = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- **Applying the Matrix:**

$$p' = Mp = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} 2x \\ 3y \\ 1 \end{bmatrix}$$

Task 2B: Affine Transformations

- **Rotation Matrix in Homogeneous Coordinate**

$$M = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- **Specific Rotation Matrix for 90 Degrees Counterclockwise**

$$M = \begin{bmatrix} \cos(90^\circ) & -\sin(90^\circ) & 0 \\ \sin(90^\circ) & \cos(90^\circ) & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- **Applying the Matrix:**

$$p' = Mp = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} -y \\ x \\ 1 \end{bmatrix}$$

Task 2B: Affine Transformations p1

- **Translation:**

$$p1' = M_{\text{translate}} \cdot p1 = \begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 4 \\ 6 \\ 1 \end{bmatrix} = \begin{bmatrix} 3 \\ 8 \\ 1 \end{bmatrix}$$

- **Scaling:**

$$p1'' = M_{\text{scale}} \cdot p1' = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 3 \\ 8 \\ 1 \end{bmatrix} = \begin{bmatrix} 6 \\ 24 \\ 1 \end{bmatrix}$$

- **Rotation:**

$$p1''' = M_{\text{rotate}} \cdot p1'' = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 6 \\ 24 \\ 1 \end{bmatrix} = \begin{bmatrix} -24 \\ 6 \\ 1 \end{bmatrix}$$

Task 2B: Affine Transformations p2

- **Translation:**

$$p2' = M_{\text{translate}} \cdot p2 = \begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 4 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 3 \\ 3 \\ 1 \end{bmatrix}$$

- **Scaling:**

$$p2'' = M_{\text{scale}} \cdot p2' = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 3 \\ 3 \\ 1 \end{bmatrix} = \begin{bmatrix} 6 \\ 9 \\ 1 \end{bmatrix}$$

- **Rotation:**

$$p2''' = M_{\text{rotate}} \cdot p2'' = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 6 \\ 9 \\ 1 \end{bmatrix} = \begin{bmatrix} -9 \\ 6 \\ 1 \end{bmatrix}$$

Task 2C: Combined Matrix Transformation

- **M = M$_{rotate}$ · M$_{scale}$ · M$_{translate}$**

- **Overview:**

$$M = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 2 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{bmatrix}$$

- **Resulting Matrix:**

$$M = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 2 & 0 & -2 \\ 0 & 3 & 6 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & -3 & -6 \\ 2 & 0 & -2 \\ 0 & 0 & 1 \end{bmatrix}$$

Task 2C: Combined Matrix Transformation

Does the Order Matter?

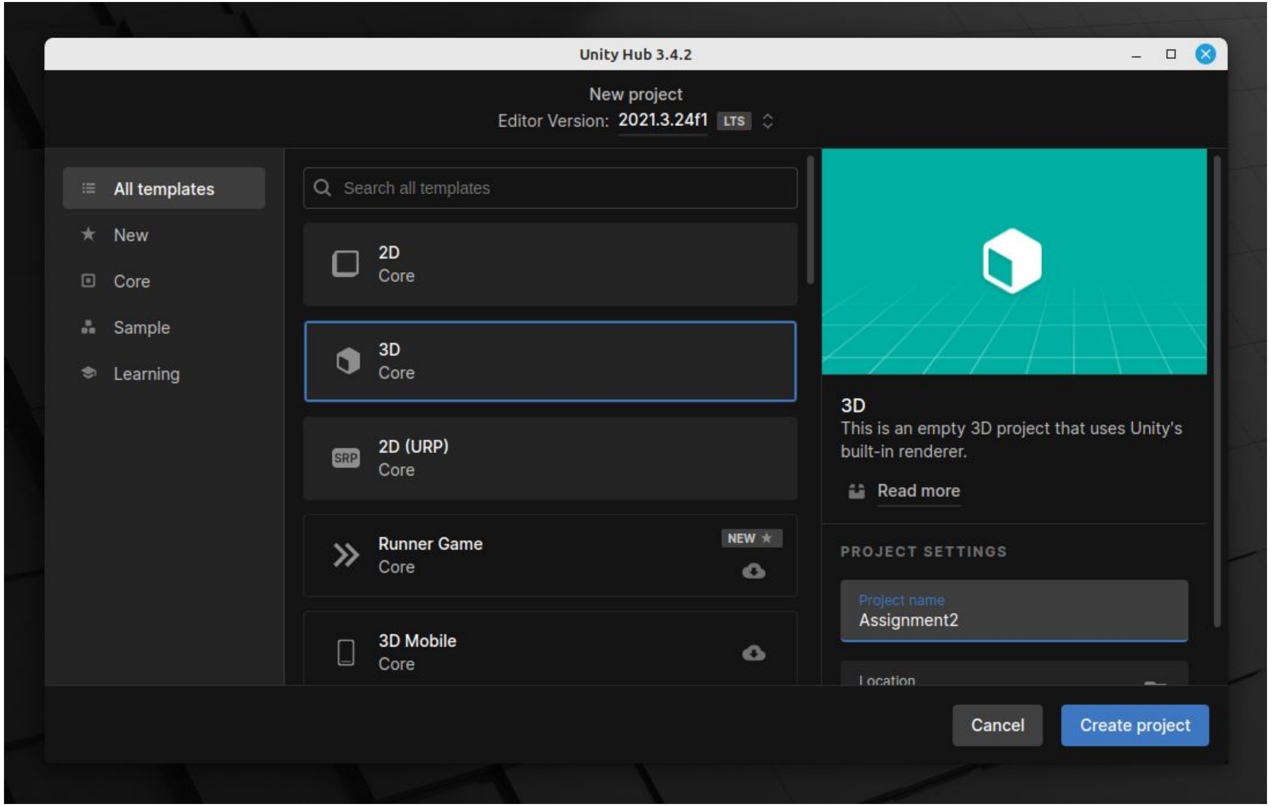
- the resulting transformation matrix and the geometric transformations' effects on the objects will vary significantly with the order of operations
- Scaling affects distances from the origin, changing the effect of subsequent rotations or translations
- Rotations change the direction of subsequent translations or the axes along which scaling occurs
- Translations move points outright, affecting the base reference from which rotations or scaling would then occur

Task 3: Parse an .obj file

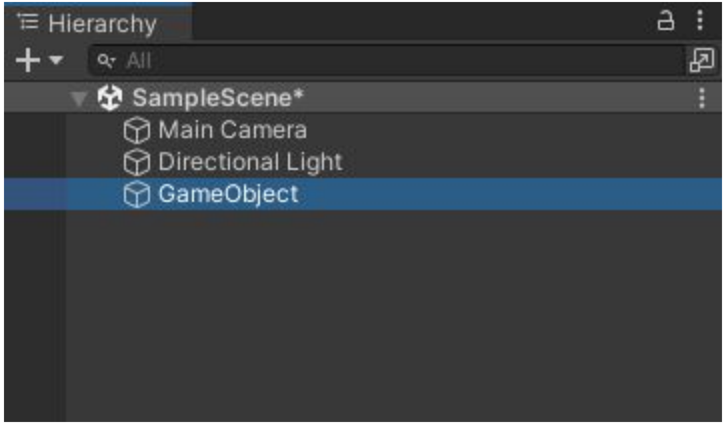
3D models can be stored and read in many ways, in this exercise we will focus in the Wavefront obj format. Try to develop a Unity3D powered obj parser. The primary goal is not to write the most performant parser, but to grasp the concepts of vertices, edges and faces.
In order to do that please consider 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

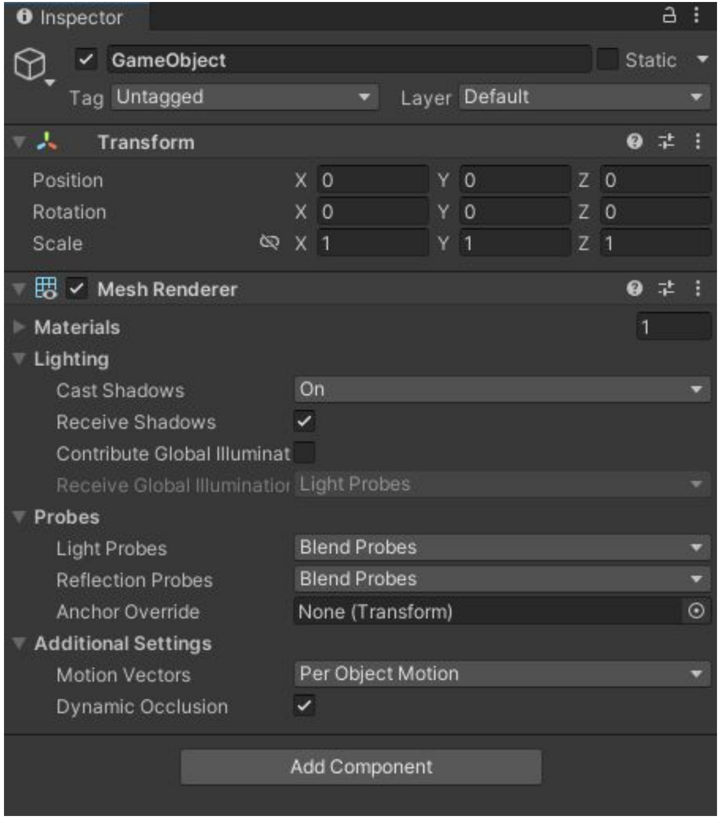
Task 3: Create new project



Task 3: GameObject -> Create Empty



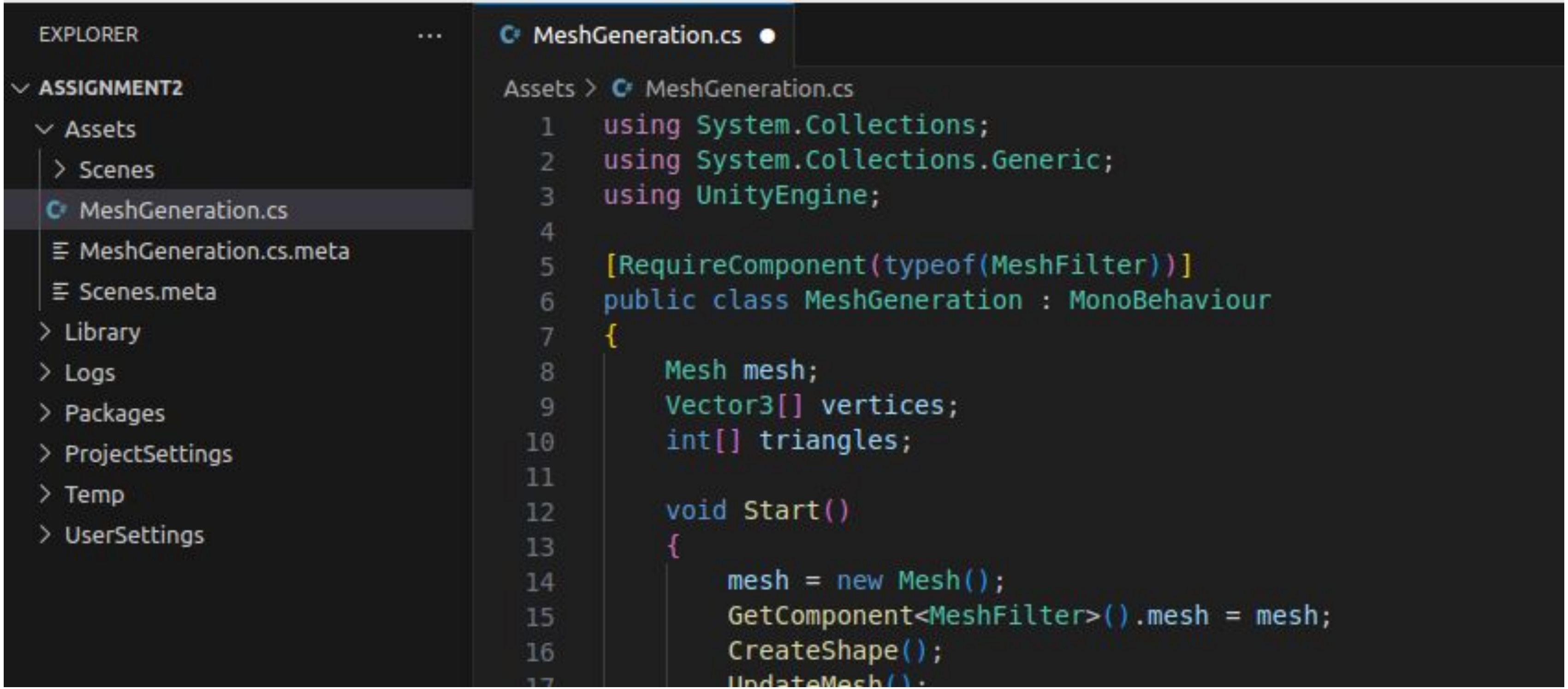
Task 3: Add Component -> MeshRenderer



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

File name and Class name have to be identical!

Task 3: Open project folder in VSCode



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

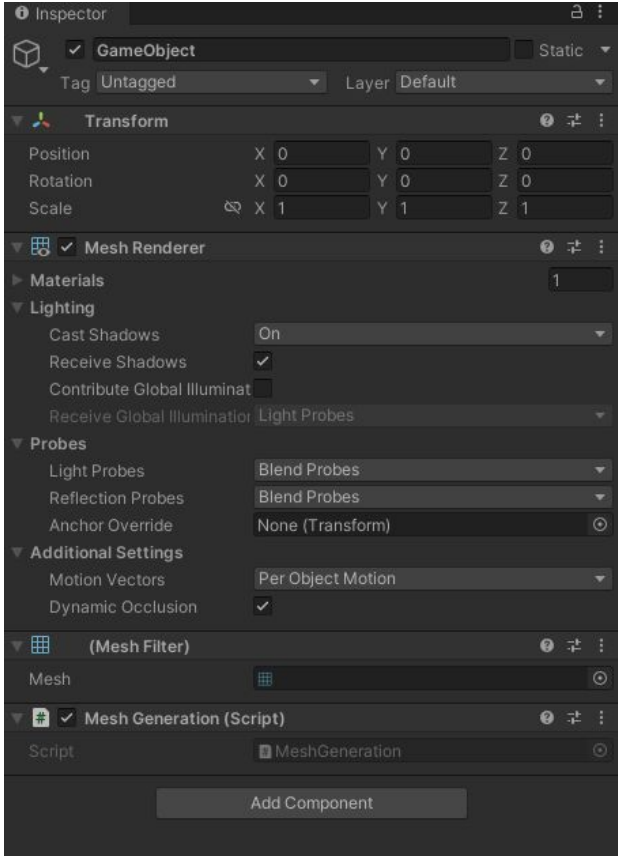
```
void CreateShape()
   {
       vertices = new Vector3[]
       {
           new Vector3(0, 0, 0),
           new Vector3(0, 0, 1),
           new Vector3(1, 0, 0),
           new Vector3(1, 0, 1)
       };

       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

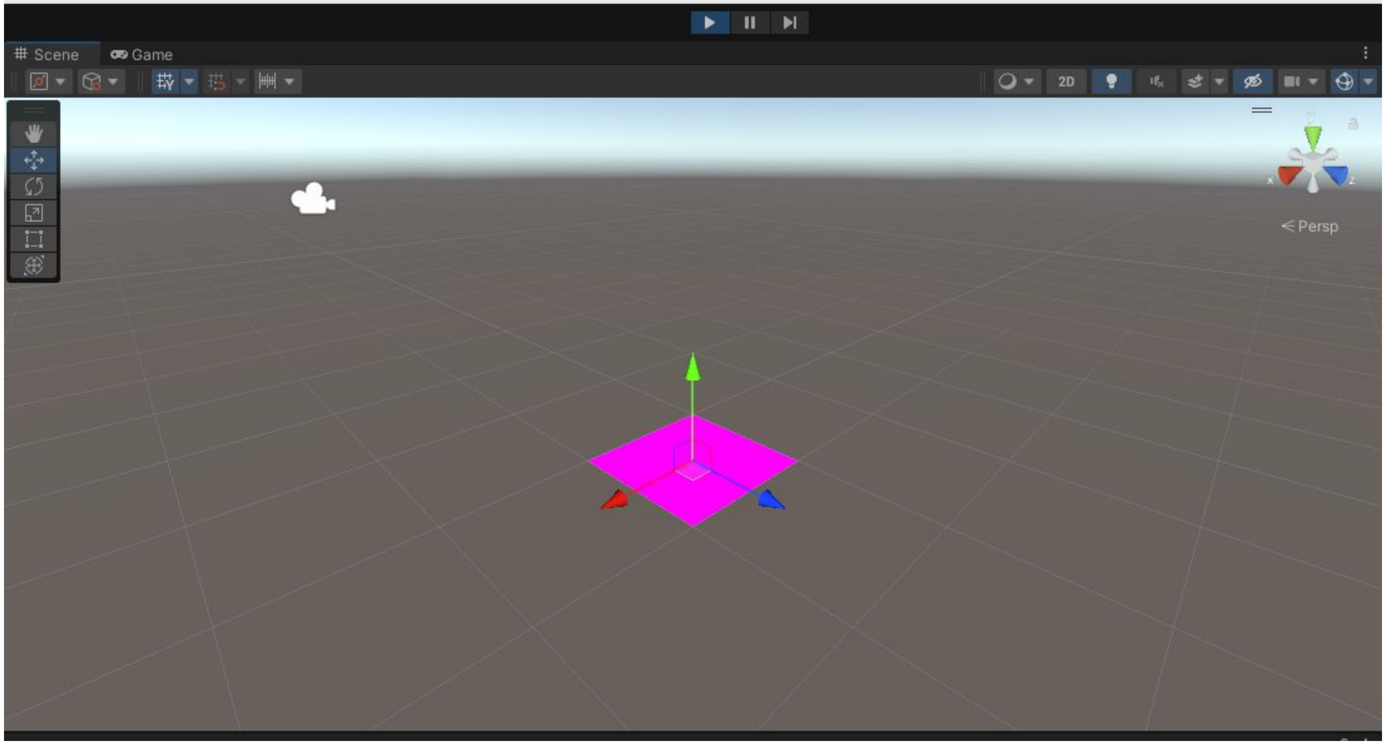
```
void UpdateMesh()
   {
       mesh.Clear();

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

       mesh.RecalculateNormals();
   }
}
```

Task 3: Add script to GameObject

Drag and drop script from Assets to Inspector



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(' ');
```

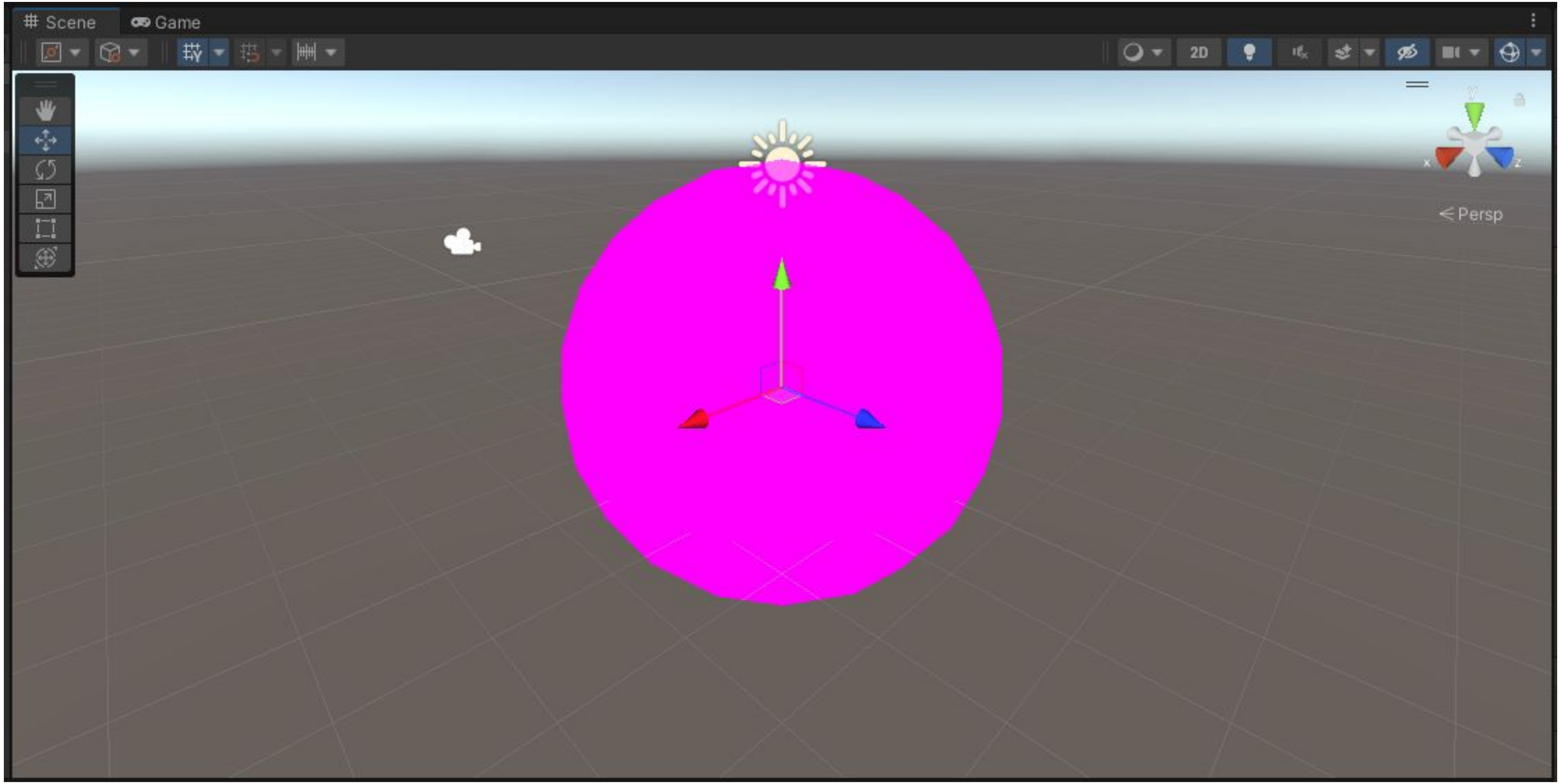
Task 3: ReadFile method

```
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);
            }
```

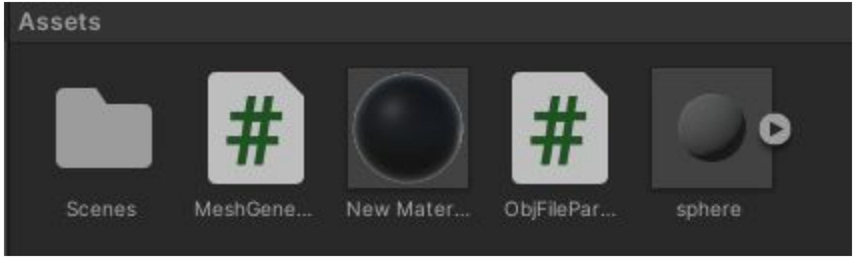
Task 3: ReadFile method

```
}

        reader.Close();

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

Task 3: Result



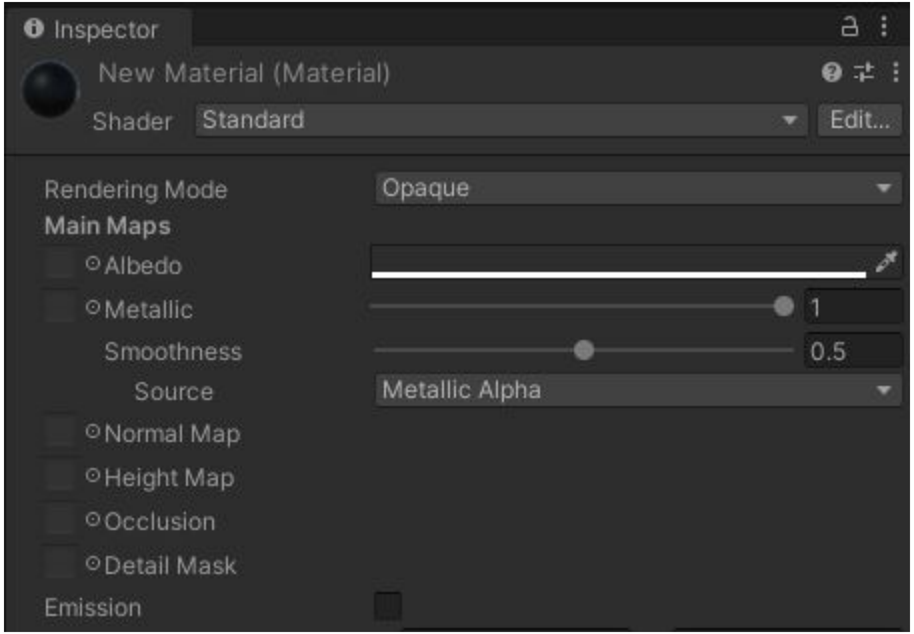
Task 3: Assets -> Create -> Material



Task 3: Material

Change Material Properties according to your liking:

- Metallic: Responsible for light reflectance behaviour



Task 3: Material

Drag and drop material into Hierarchy not GameObject Inspector!