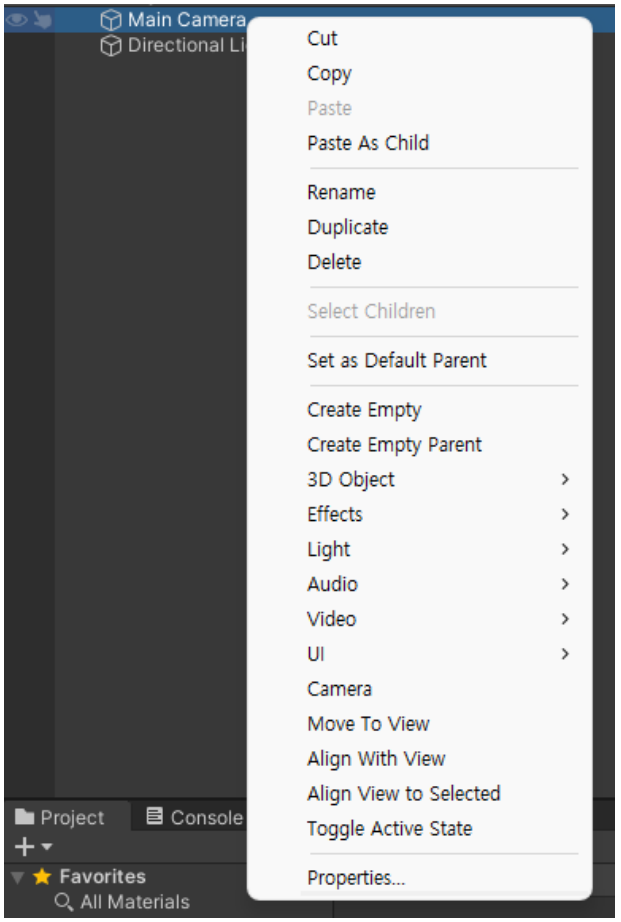
[Unity]Basic_Transforms

Sung Soo Hwang

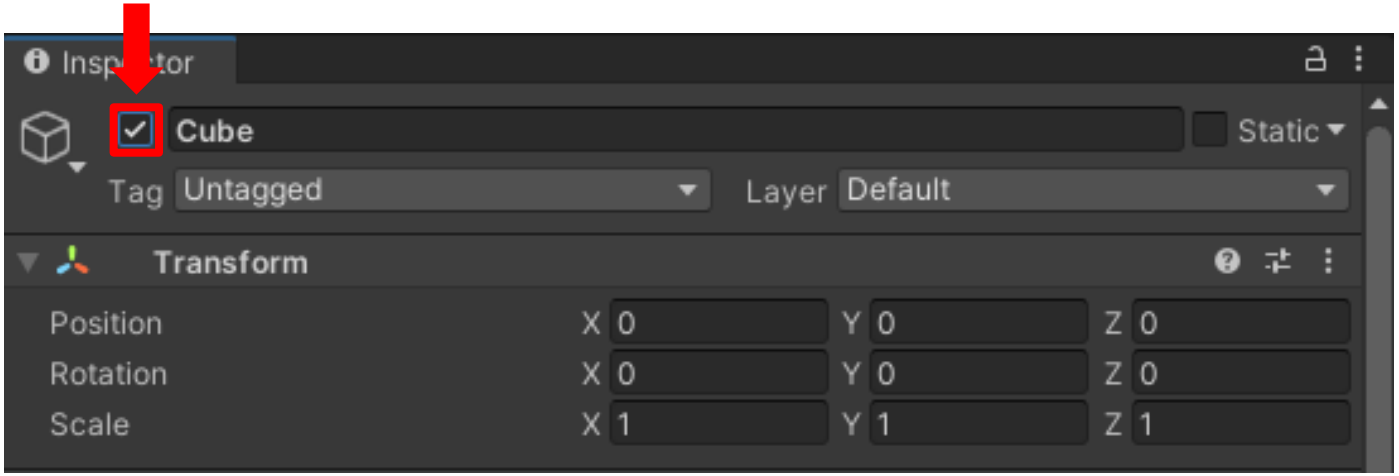
Tip 1

- Move To View
  - Move the component to the center of the view
- Align With View
  - Move the component to the location where you are looking at the component
  - If you want the object to be rendered at the point you are looking at, right click on the main camera component and click Align With View
- Align View to Selected
  - Move to the view of the component



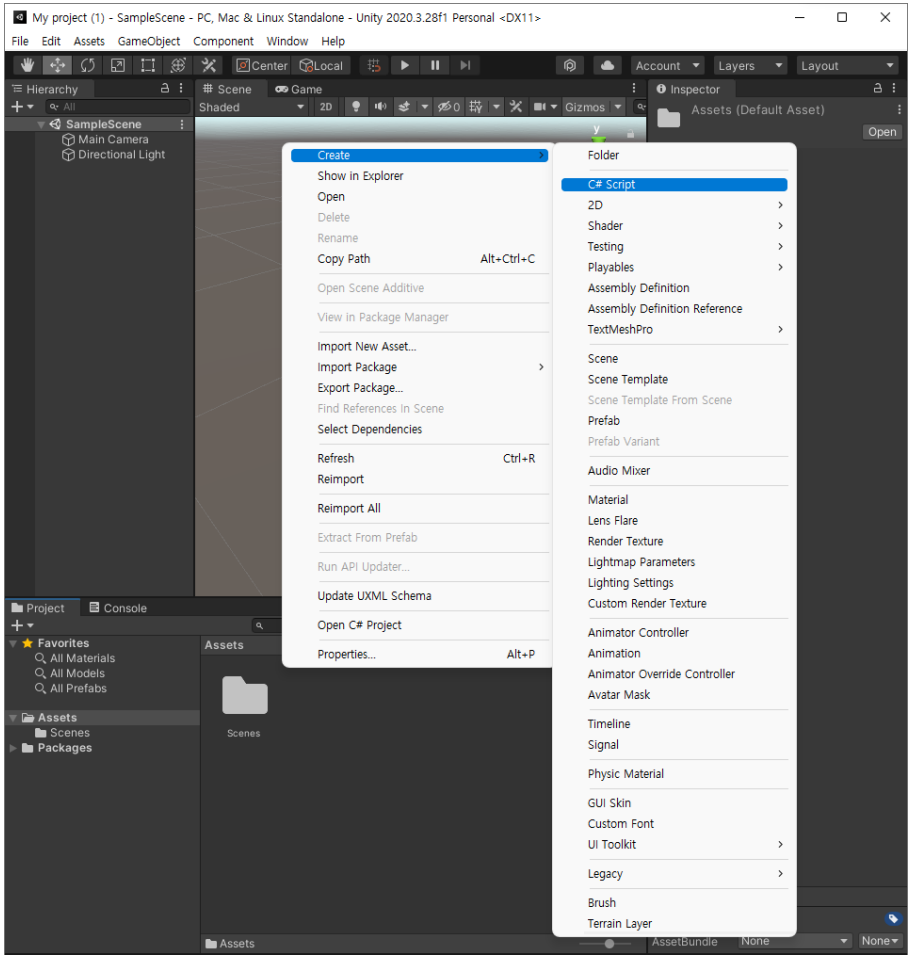
Tip 2

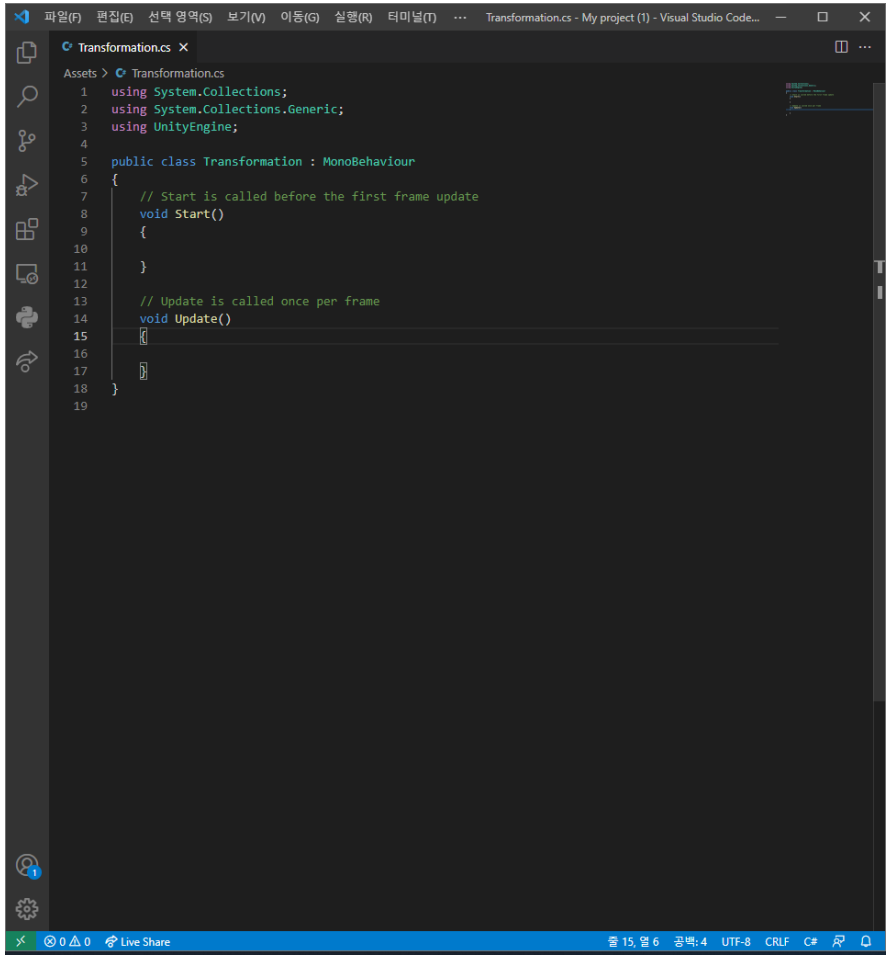
- In this practice, it will be easier to check the changed results by creating default objects and using this check box



How to transform a cube/plane

- Create a C# script
- Open the C# script





* C# script

- C# Script Basic Structure Description

```
1   using System.Collections;
2   using System.Collections.Generic;
3   using UnityEngine;
4
5   public class Rotation : MonoBehaviour
6   {
7       // Start is called before the first frame update
8       void Start()
9       {
10
11      }
12
13      // Update is called once per frame
14      void Update()
15      {
16
17      }
18  }
19
```

Important necessary modules for the program

Start is called on the frame when a script is enabled just before any of the Update methods are called the first time.

Update is called every frame.

Code

- Whole Code

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

0 references
public class Transformation : MonoBehaviour
{
    3 references
    public Mesh mesh;
    2 references
    public MeshFilter meshFilter;

    // Start is called before the first frame update
    0 references
    void Start() {
        meshFilter = GetComponent<MeshFilter>();
        mesh = meshFilter.mesh;

        Transform();
    }

    void Transform() {
        var M = Matrix4x4.identity;

        M[0,0] = 1f;
        M[0,0] = 0f;
        M[0,0] = 0f;
        M[0,0] = 0f;

        M[1,0] = 0f;
        M[1,1] = 1f;
        M[1,2] = 0f;
        M[1,3] = 0f;

        M[2,0] = 0f;
        M[2,1] = 0f;
        M[2,2] = 1f;
        M[2,3] = 0f;

        M[3,0] = 0f;
        M[3,1] = 0f;
        M[3,2] = 0f;
        M[3,3] = 1f;

        TransformMesh(M);
    }

    1 reference
    void TransformMesh(Matrix4x4 T) {
        var srcVerts = mesh.vertices;
        var outVerts = new Vector3[srcVerts.Length];

        for (int i = 0; i < outVerts.Length; ++i) {
            outVerts[i] = T.MultiplyPoint3x4(srcVerts[i]);
        }

        mesh.vertices = outVerts;
    }
}
```

Code

- Whole Code – 1

```
1   using System.Collections;
2   using System.Collections.Generic;
3   using UnityEngine;
4
    0 references
5   public class Transformation : MonoBehaviour
6   {
        3 references
7       public Mesh mesh;
        2 references
8       public MeshFilter meshFilter;
9
10      // Start is called before the first frame update
        0 references
11      void Start() {
12          meshFilter = GetComponent<MeshFilter>();
13          mesh = meshFilter.mesh;
14
15          Transform();
16      }
17
```

Member variables for getting Mesh of planets

Code

- Whole Code – 2

```
18      void Transform() {
19          var M = Matrix4x4.identity;
20
21          M[0,0] = 1f;
22          M[0,0] = 0f;
23          M[0,0] = 0f;
24          M[0,0] = 0f;
25
26          M[1,0] = 0f;
27          M[1,1] = 1f;
28          M[1,2] = 0f;
29          M[1,3] = 0f;
30
31          M[2,0] = 0f;
32          M[2,1] = 0f;
33          M[2,2] = 1f;
34          M[2,3] = 0f;
35
36          M[3,0] = 0f;
37          M[3,1] = 0f;
38          M[3,2] = 0f;
39          M[3,3] = 1f;
40
41          TransformMesh(M);
42      }
43
        1 reference
44      void TransformMesh(Matrix4x4 T) {
45          var srcVerts = mesh.vertices;
46          var outVerts = new Vector3[srcVerts.Length];
47
48          for (int i = 0; i < outVerts.Length; ++i) {
49              outVerts[i] = T.MultiplyPoint3x4(srcVerts[i]);
50          }
51
52          mesh.vertices = outVerts;
53      }
54  }
```

4x4 Homogeneous Matrix (3D)

Transforms a position by matrix multiplication

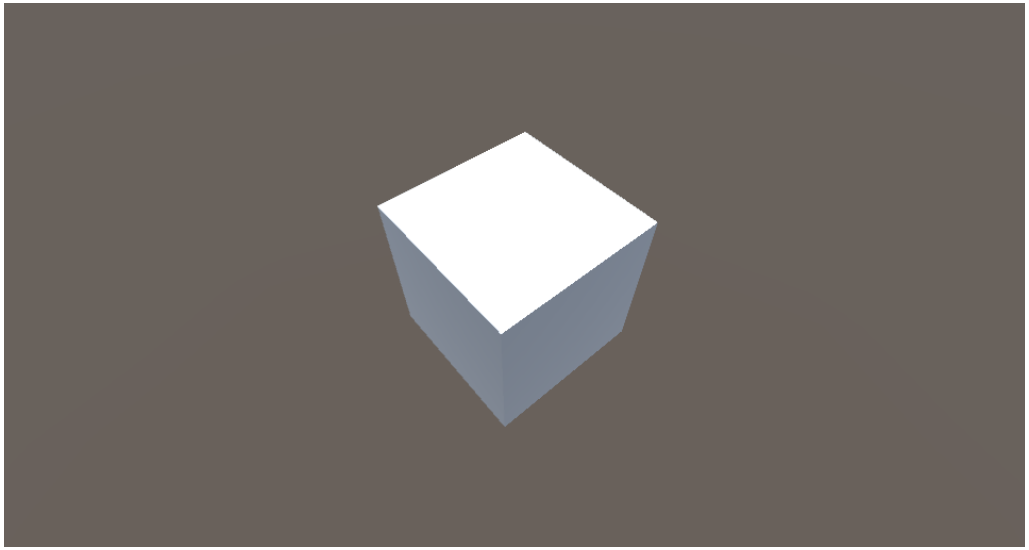
Update mesh of the planets

Scaling

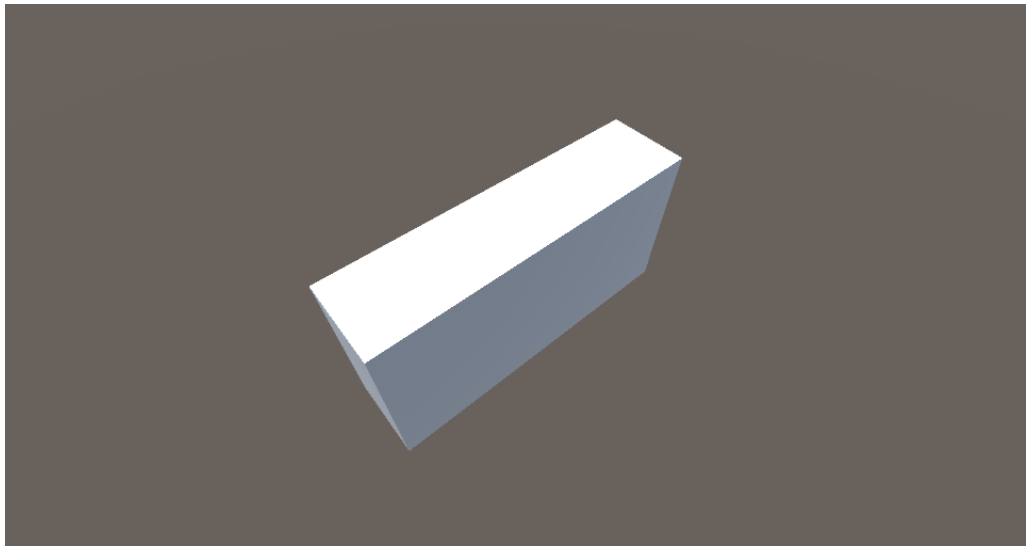
```
void Transform() {
        var M = Matrix4x4.identity;

        // X-axis: x2
        // Z-axis: x0.5
        M[0,0] = 2f; // (2 x 0.5)
        M[0,1] = 0f;
        M[0,2] = 0f;
        M[0,3] = 0f;

        M[1,0] = 0f;
        M[1,1] = 1f;
        M[1,2] = 0f;
        M[1,3] = 0f;

        M[2,0] = 0f;
        M[2,1] = 0f;
        M[2,2] = 0.5f; // (1 x 0.5)
        M[2,3] = 0f;

        M[3,0] = 0f;
        M[3,1] = 0f;
        M[3,2] = 0f;
        M[3,3] = 1f;

        TransformMesh(M);
}
```

$$\begin{bmatrix} X_{new} \\ Y_{new} \\ Z_{new} \\ 1 \end{bmatrix} = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} X_{old} \\ Y_{old} \\ Z_{old} \\ 1 \end{bmatrix}$$

3D Scaling Matrix



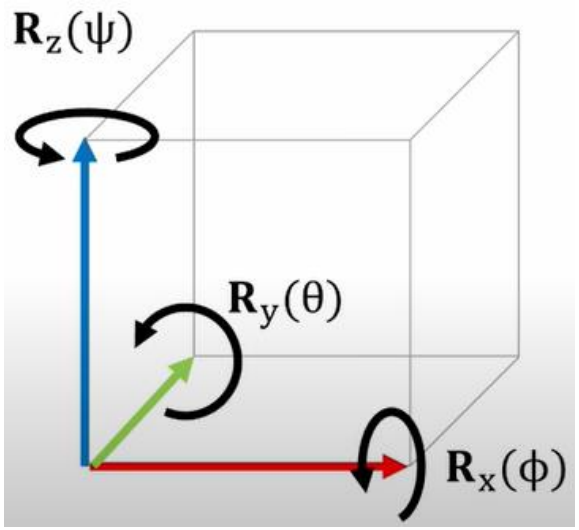
Original 3D object



Scaled 3D object

Rotation

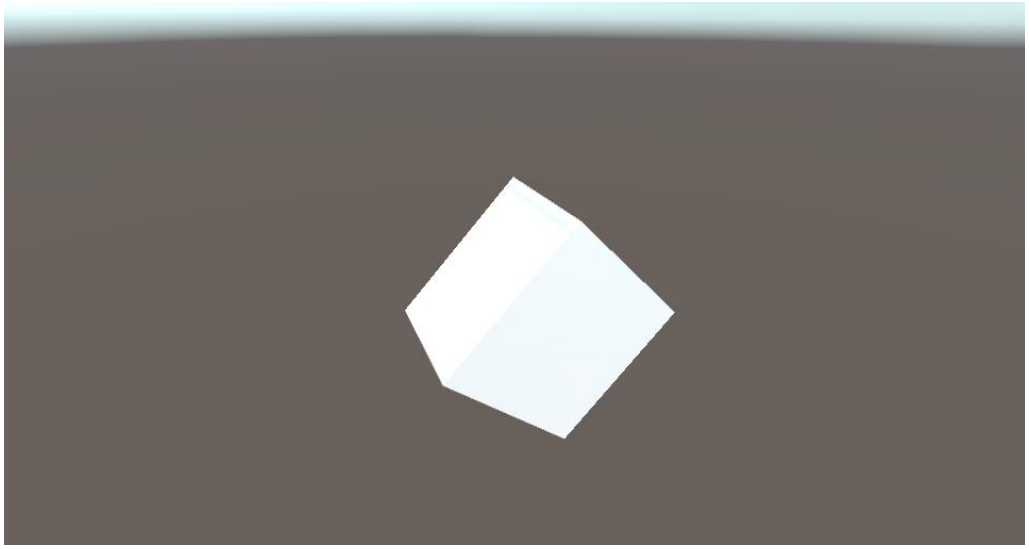
```
void Transform() {
        var M = Matrix4x4.identity;

        // Z-axis rotation
        // Theta = 45°
        M[0,0] = 0.70711f; // cos45° ≒ 0.70711
        M[0,1] = -0.70711f; // -sin45° ≒ -0.70711
        M[0,2] = 0f;
        M[0,3] = 0f;

        M[1,0] = 0.70711f; // sin45° ≒ 0.70711
        M[1,1] = 0.70711f; // cos45° ≒ 0.70711
        M[1,2] = 0f;
        M[1,3] = 0f;

        M[2,0] = 0f;
        M[2,1] = 0f;
        M[2,2] = 1f;
        M[2,3] = 0f;

        M[3,0] = 0f;
        M[3,1] = 0f;
        M[3,2] = 0f;
        M[3,3] = 1f;

        TransformMesh(M);
}
```



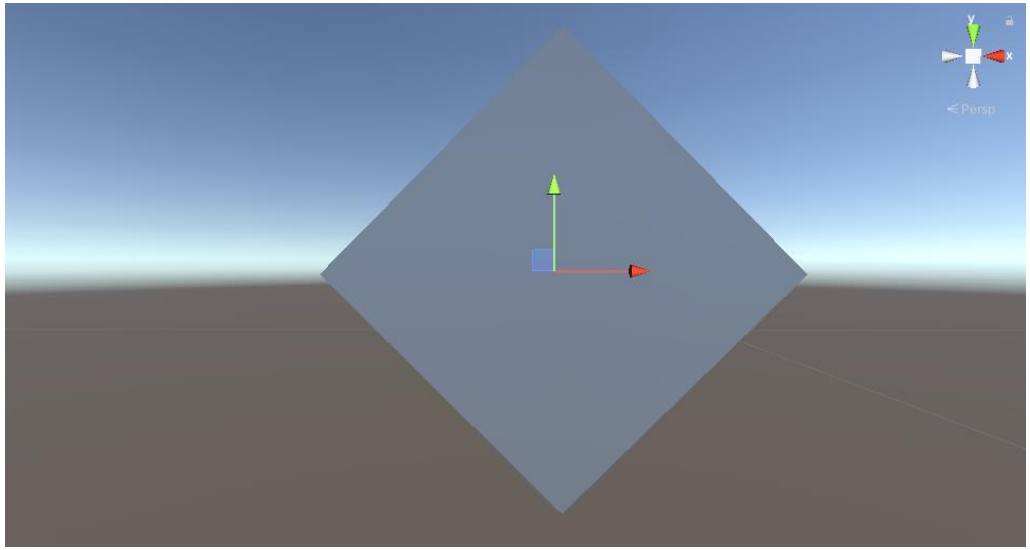
$$\begin{bmatrix} X_{new} \\ Y_{new} \\ Z_{new} \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} X_{old} \\ Y_{old} \\ Z_{old} \\ 1 \end{bmatrix}$$

3D Rotation Matrix

(For Z-Axis Rotation)



Rotated 3D object

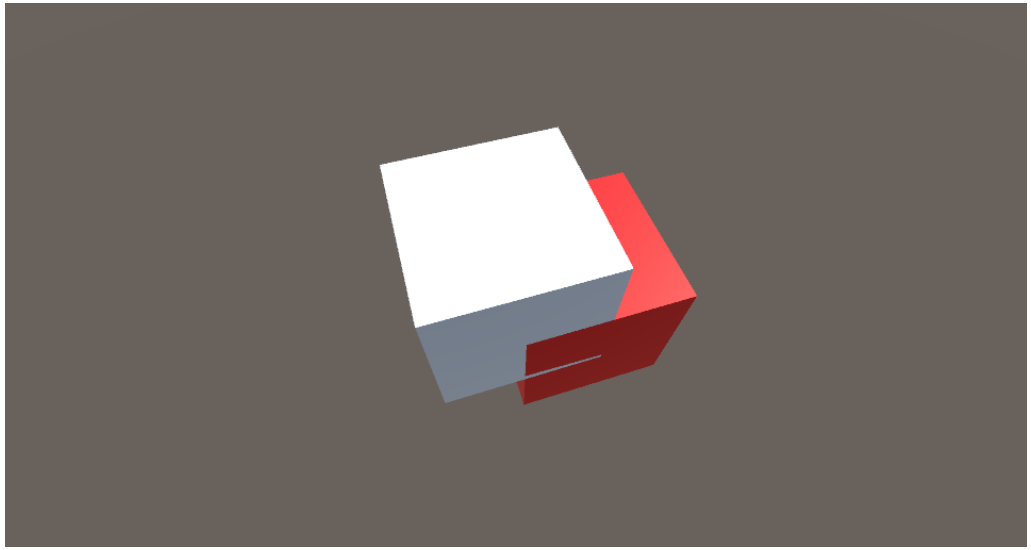


Rotated 3D object (side)

Translation

```
void Transform() {
        var M = Matrix4x4.identity;

        // X_new = X_old - 0.5
        // Y_new = Y_old + 0.5
        M[0,0] = 1f;
        M[0,1] = 0f;
        M[0,2] = 0f;
        M[0,3] = -0.5f;

        M[1,0] = 0f;
        M[1,1] = 1f;
        M[1,2] = 0f;
        M[1,3] = 0.5f;

        M[2,0] = 0f;
        M[2,1] = 0f;
        M[2,2] = 1f;
        M[2,3] = 0f;

        M[3,0] = 0f;
        M[3,1] = 0f;
        M[3,2] = 0f;
        M[3,3] = 1f;

        TransformMesh(M);
}
```

Translation

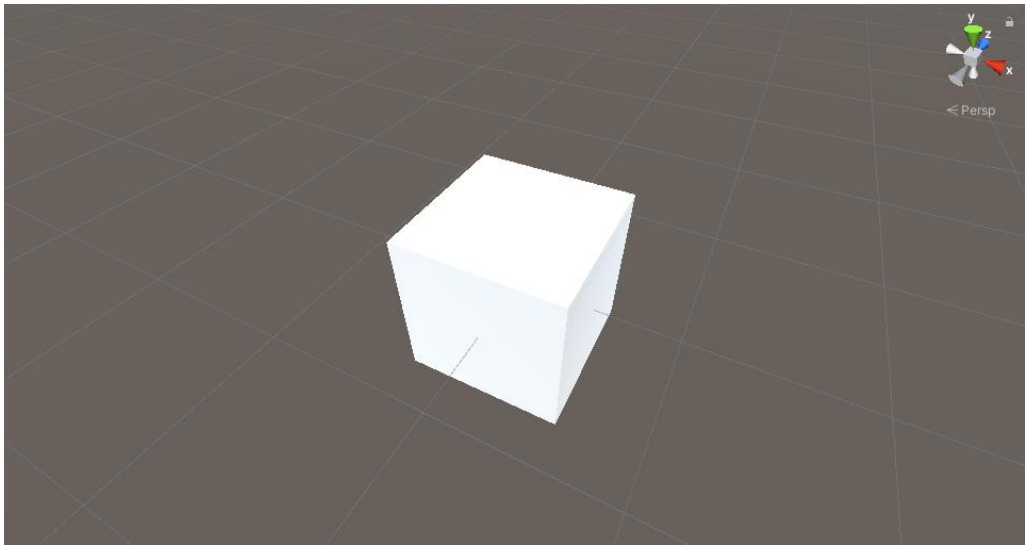


Original 3D object (red)
Translated 3D object (white)

Shear

```
void Transform() {
        var M = Matrix4x4.identity;

        // 3D shearing matrix in X-axis and Y-axis
        M[0,0] = 1f;
        M[0,1] = 0.5f;
        M[0,2] = 0f;
        M[0,3] = 0f;

        M[1,0] = 0.5f;
        M[1,1] = 1f;
        M[1,2] = 0f;
        M[1,3] = 0f;

        M[2,0] = 0.5f;
        M[2,1] = 0.5f;
        M[2,2] = 1f;
        M[2,3] = 0f;

        M[3,0] = 0f;
        M[3,1] = 0f;
        M[3,2] = 0f;
        M[3,3] = 1f;

        TransformMesh(M);
}
```

Shear



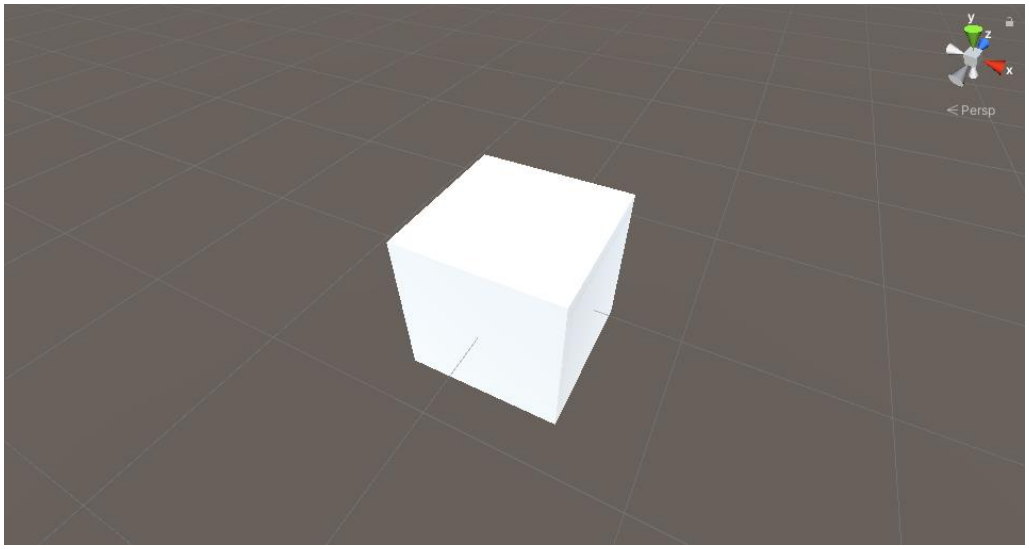
Original 3D object

Sheared 3D object

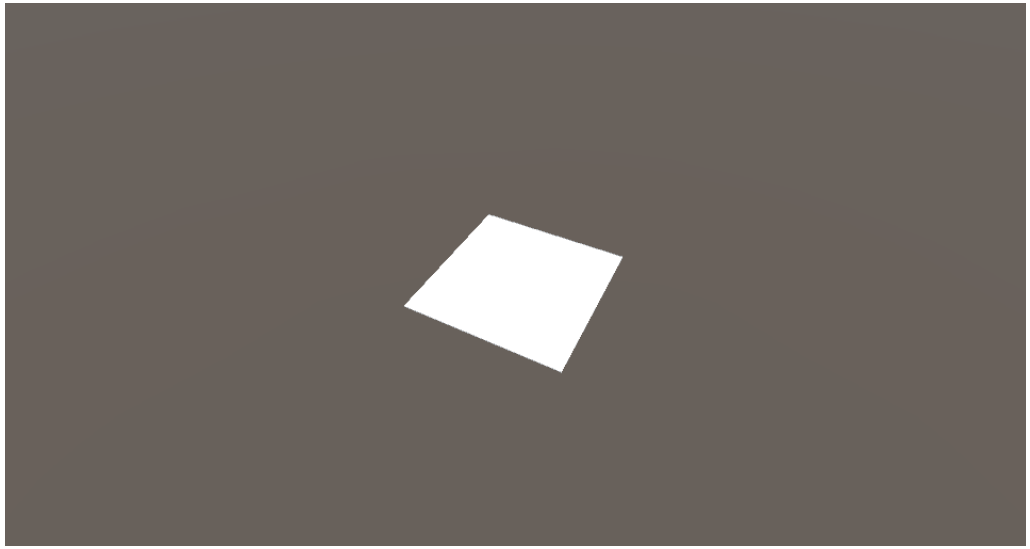
Projection

```
void Transform() {
        var M = Matrix4x4.identity;

        // projection on the XZ plane
        M[0,0] = 1f;
        M[0,1] = 0f;
        M[0,2] = 0f;
        M[0,3] = 0f;

        M[1,0] = 0f;
        M[1,1] = 0f;
        M[1,2] = 0f;
        M[1,3] = 0f;

        M[2,0] = 0f;
        M[2,1] = 0f;
        M[2,2] = 1f;
        M[2,3] = 0f;

        M[3,0] = 0f;
        M[3,1] = 0f;
        M[3,2] = 0f;
        M[3,3] = 1f;

        TransformMesh(M);
}
```

Projection



Original 3D object



Projected 3D object