```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class TestQuat : MonoBehaviour
6  {
7      public GameObject PlanetBody;
8      public MyVector3 PlanetCentre, SunPos;
9      public float t = 0;
10     public float y = 0;
11
12     public float sunoffset;
13     public float OrbitAxisX = 0;
14     public float OrbitAxisY = 0;
15     public float OrbitAxisZ = 0;
16     public float OrbitAxis2X = 0;
17     public float OrbitAxis2Y = 2;
18     public float OrbitAxis2Z = 0;
19     public float Vertex1 = 2;
20     public float Vertex2 = 0;
21         public float Vertex3 = 0;
22     float VelocityX, VelocityY, VelocityZ;
23     float RotX, RotY, RotZ = 1;
24     float ScaleX, ScaleY, ScaleZ = 1;
25     float TranslateX, TranslateY, TranslateZ = 0;
26     Vector3[] ModelSpaceVertices;
27     public MeshFilter Planet;
28     // Start is called before the first frame update
29     void Start()
30     {
31
32         //Planet = PlanetBody.GetComponent<MeshFilter>();
33         //ModelSpaceVertices = Planet.mesh.vertices;
34     }
35
36     // Update is called once per frame
37     void Update()
38     {
39         //Vector3[] TransformedVertices = new Vector3
             [ModelSpaceVertices.Length];
40         //Matrix4By4 T = MyTransform.Translate(TranslateX, TranslateY,
             TranslateZ);
41         ////Rotation is in radians
42         //Matrix4By4 R = MyTransform.Rotation(RotX, RotY, RotZ);
43         //Matrix4By4 S = MyTransform.Scale(ScaleX, ScaleY, ScaleZ);
44         //Matrix4By4 M = MyTransform.TRS(T, R, S);
45
46         //for (int i = 0; i < TransformedVertices.Length; i++)
47         //{
48         //    TransformedVertices[i] = M * ModelSpaceVertices[i];
49         //}
50         //Planet.mesh.vertices = TransformedVertices;
51         //Planet.mesh.RecalculateNormals();
```

```
52          //Planet.mesh.RecalculateBounds();
53
54
55
56          t += Time.deltaTime;
57          //OrbitAxis2X += Time.deltaTime;
58
59          //OrbitAxis2Z -= Time.deltaTime;
60          //WORKS
61          MyQuaternion q = new MyQuaternion(y, new MyVector3(OrbitAxisX,
              OrbitAxisY, OrbitAxisZ));
62
63
64          //    q.PrintStats();
65          //WORKS
66          MyQuaternion r = new MyQuaternion(t, new MyVector3(OrbitAxis2X,
              OrbitAxis2Y, OrbitAxis2Z));
67          Debug.Log((r*q.Inverse()).ToUnityQuaternion());
68          //   r.PrintStats();
69
70
71          //DOESNT WORK Y AXIS IS BROKEN
72
73
74          MyQuaternion slerped = MyQuaternion.SLERP(q, r, t);
75
76         // slerped.PrintStats();
77
78          MyVector3 p = new MyVector3(Vertex1, Vertex2, Vertex3);
79
80         // p.PrintStats();
81
82          //WORKS
83          MyQuaternion K = new MyQuaternion(p);
84
85        //  K.PrintStats();
86
87          MyQuaternion newK = slerped * K * slerped.Inverse();
88
89     //     newK.PrintStats();
90
91          MyVector3 newP = MyVector3.GetAxis(newK);
92
93          // newP.PrintStats();
94
95          //Debug.Log(newP.ToUnityVector());
96      //    slerped.Inverse().PrintStats();
97          PlanetBody.transform.position = newP.ToUnityVector();
98      }
99  }
```