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

public class MyQuaternion
{


    public float w, x, y, z;

    public MyQuaternion(float Angle,MyVector3 Axis)
    {
        float halfAngle = Angle / 2;
        w = Mathf.Cos(halfAngle);
        x = Axis.x * Mathf.Sin(halfAngle);
        y = Axis.y * Mathf.Sin(halfAngle);
        z = Axis.z * Mathf.Sin(halfAngle);
    }
    public MyQuaternion(MyVector3 Axis)
    {
        w = 0.0f;
        x = Axis.x;
        y = Axis.y;
        z = Axis.z;
    }
    public static MyQuaternion operator*(MyQuaternion S,MyQuaternion R)
    {
        MyVector3 SAxis = new MyVector3(S.x, S.y, S.z);
        MyVector3 RAxis = new MyVector3(R.x, R.y, R.z);
        MyQuaternion rs = new MyQuaternion(0,new MyVector3(0,0,0));


        rs.w = S.w * R.w - MyVector3.Dot(SAxis, RAxis);

        MyVector3 s  = (MyVector3.ScaleVector(RAxis, S.w))
                     + (MyVector3.ScaleVector(SAxis, R.w))
                     + (MyVector3.VectorCrossProduct(RAxis, SAxis));
        rs.x = s.x;
        rs.y = s.y;
        rs.z = s.z;
        return rs;
    }
    public static MyQuaternion operator -(MyQuaternion lhs)
    {
        MyQuaternion rs = new MyQuaternion(0, new MyVector3(0, 0, 0));


        rs.w = -lhs.w;
        rs.x = -lhs.x;
        rs.y = -lhs.y;
        rs.z = -lhs.z;

        return rs;
```

```
54      }
55      public static MyQuaternion Vector4ToQuaternion(MyVector4 vec4)
56      {
57          MyQuaternion rv = new MyQuaternion(0,new MyVector3(0,0,0));
58
59
60          rv.w = vec4.w;
61
62          rv.x = vec4.x;
63          rv.y = vec4.y;
64          rv.z = vec4.z;
65
66          return rv;
67      }
68      public MyQuaternion GetAxisAngle()
69      {
70          MyQuaternion rv = new MyQuaternion(0, new MyVector3(0, 0, 0));
71
72          float halfAngle = Mathf.Acos(w);
73          rv.w = halfAngle * 2;
74
75          rv.x = x / Mathf.Sin(halfAngle);
76          rv.y = y / Mathf.Sin(halfAngle);
77          rv.z = z / Mathf.Sin(halfAngle);
78
79          return rv;
80      }
81      public MyQuaternion GetAxis()
82      {
83          MyQuaternion rv = new MyQuaternion(0, new MyVector3(0, 0, 0));
84
85          rv.x = x;
86          rv.y = y;
87          rv.z = z;
88
89          return rv;
90
91      }
92      public static MyQuaternion SLERP(MyQuaternion q, MyQuaternion r,float t)
93      {
94          //Doesnt Break
95          t = Mathf.Clamp(t, 0.0f, 1.0f);
96          //Q.INVERSE IS FKED
97
98          MyQuaternion d = r * q.Inverse();
99          //DOESNT ENTIRELY WORK
100         MyVector4 AxisAngle = MyVector4.QuaternionToVector4(d).GetAxisAngle
              ();
101
102         MyQuaternion dT = new MyQuaternion(AxisAngle.w * t, new MyVector3
              (AxisAngle.x,AxisAngle.y, AxisAngle.z));
103
104         //Using this works but circle is still an oval
```

```
105         // MyQuaternion dT = d;
106          return dT * q;
107      }
108      public MyQuaternion Inverse()
109      {
110          MyQuaternion rv = new MyQuaternion(0, new MyVector3(0, 0, 0));
111
112          rv.w = w;
113
114          rv.SetAxis(-GetAxis());
115
116          return rv;
117      }
118      public MyQuaternion PrintStats()
119      {
120          MyQuaternion rv = new MyQuaternion(0, new MyVector3(0, 0, 0));
121
122          Debug.Log(rv.w + " " + rv.x + " " + rv.y + " " + rv.z);
123
124          return rv;
125      }
126      public Quaternion ToUnityQuaternion()
127      {
128          Quaternion Qr = new Quaternion(w, x, y, z);
129
130
131          return Qr;
132      }
133      public MyQuaternion SetAxis(MyQuaternion setQuat)
134      {
135          MyQuaternion rv = new MyQuaternion(0, new MyVector3(0, 0, 0));
136
137          rv.x = setQuat.x;
138          rv.y = setQuat.y;
139          rv.z = setQuat.z;
140
141          return rv;
142      }
143  }
144
```