```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.UI;
5  public class RocketMovementScript : MonoBehaviour
6  {
7      //GameObject Variables
8      //------------------------------------------------------------------------
       -------
9      public MeshFilter PlayerMesh;
10     public GameObject Player;
11     public GameObject camera1;
12
13     //Display Text Variables
14     //------------------------------------------------------------------------
       -------
15     public Text TextAccelerationX, TextAccelerationY, TextAccelerationZ,
         TextPosX, TextPosY, TextPosZ;
16     public float BoosterFuel, ThrusterFuel = 100.0f;
17
18     //Mercury Variables
19     //------------------------------------------------------------------------
       -------
20     public MyVector3 MercuryAcceleration = new MyVector3(0, 0, 0);
21     public float MercuryMass;
22     public static MyVector3 MercuryPos,SunPos = new MyVector3(0, 0, 0);
23
24     //Camera Variables
25     //------------------------------------------------------------------------
       -------
26     public MyVector3 Cameraoffset = new MyVector3(0, 0, -25);    // The
         offset is set as a vector 3 to ensure that the camera can be adjusted
         on all planes.
27     public int CameraPos;
28
29
30     //Normalized Vector Variables
31     //---------------------------
32     public static float NormalizedVectorPlanetX, NormalizedVectorPlanetY,
         NormalizedVectorPlanetZ;
33     public static MyVector3 NormalizedVectorPlanet = new MyVector3(0, 0, 0);
34     public float AngleFromRocketToPlanet = 0.0f;
35     public static MyVector3 NormalizedRocketVector;
36     //Motion Variables
37     //---------------------------
38     public MyVector3 Force       = new MyVector3(0, 0, 0);
39     public MyVector3 Momentum    = new MyVector3(0, 0, 0);
40     public MyVector3 Velocity    = new MyVector3(0, 0, 0);
41     public MyVector3 RocketForce = new MyVector3(0, 0, 0);
42     public float Mass = 1;
43     public float FlightOrbitRadius = 100.0f;
44
45     //Rocket TRS Variables
```

```
46      //------------------------------------------------------------------------
          -------
47
48      //Acceleration
49      //----------------------------
50      public static float AccelerationX, AccelerationY, AccelerationZ;
51      public MyVector3 Acceleration = new MyVector3(0, 0, 0);
52      //Rotation
53      //----------------------------
54      public float RpmX, RpmY, RpmZ;
55      public float RotX, RotY, RotZ = 0;
56      //Scale
57      //----------------------------
58      public float ScaleX, ScaleY, ScaleZ = 1;
59      //Translate
60      //----------------------------
61      public static float TranslateX, TranslateY, TranslateZ = 0;
62      public static MyVector3 Translate = new MyVector3(0, 0, 0);
63      //Vertices Array
64      //----------------------------
65      public Vector3[] ModelSpaceVertices;
66
67      //Boolean Operators
68      //------------------------------------------------------------------------
          -------
69      public bool Wselected;
70      public bool FlightComputerEnabled = false;
71      public bool GravityEnabled, CapAccelerationEnabled, SphereBoundEnabled,
          PlayerInputEnabled = true;
72      public static bool IsGrounded = false;
73      public float test = 0.0f;
74
75      //Other
76      //------------------------------------------------------------------------
          -------
77      public float time = 0;
78
79      // Start is called before the first frame update
80      //------------------------------------------------------------------------
          -------
81      void Start()
82      {
83          //Starting Position of Player
84          Translate.y = 55.0f;
85          PlayerMesh = Player.GetComponent<MeshFilter>();
86          //Gets a copy of all vertices from mesh and stores in array
87          ModelSpaceVertices = PlayerMesh.mesh.vertices;
88
89      }
90      // Update is called once per frame
91      //------------------------------------------------------------------------
          -------
92      void Update()
```

```
93      {
94
95
96          time = Time.deltaTime;
97          Velocity = PhysicsLibrary.VelocityCalculation(Velocity,
              Acceleration, time);
98          //Rotation Input
99          //------------------------------------------------------------------
              -----------
100         RotX += RpmX * Time.deltaTime;   //Ensures that Rotation will
              continue in space
101         RotY += RpmY * Time.deltaTime;
102         RotZ += RpmZ * Time.deltaTime;
103
104         //Update Planet Position
105         //------------------------------------------------------------------
              -----------
106         SunPos = MatrixOrbit.SunPos;
107         //Debug.Log(SunPos.ToUnityVector());
108         MercuryPos = (MatrixOrbit.MercuryPos * 100.0f);
109
110         //Update Text Onscreen
111         //------------------------------------------------------------------
              -----------
112         UpdateText();
113
114         //Toggle Elements
115         //------------------------------------------------------------------
              -----------
116
117         //if (Input.GetKeyDown(KeyCode.T)){FlightComputerEnabled = true;}
118         //if (FlightComputerEnabled == true)
119         //{
120         //    SphereBoundEnabled = false;
121         //    CapAccelerationEnabled = false;
122         //    GravityEnabled = false;
123         //    PlayerInputEnabled = false;
124
125         //    Translate.x = FlightOrbitRadius * Mathf.Cos(time);
126         //    Translate.y = FlightOrbitRadius * Mathf.Sin(time);
127         //}
128         if (SphereBoundEnabled == true)
129         {
130             //PhysicsLibrary.SphereBounds3(50.0f, MercuryPos, Translate,
                  NormalizedRocketVector);
131             PhysicsLibrary.SphereBounds(50.0f, SunPos, Translate,
                  NormalizedRocketVector);
132
133         }
134         //Apply Gravity
135         //------------------------------------------------------------------
              ------------
136         if (GravityEnabled == true){PlanetGravity(SunPos);}
```

```
137         if (CapAccelerationEnabled == true){ CapAcceleration(); CapVelocity
              ();}
138
139         if (PlayerInputEnabled == true||Input.GetKeyDown(KeyCode.K))
140         {
141
142             PlayerInput();
143             PlayerInputEnabled = true;
144             SphereBoundEnabled = true;
145             CapAccelerationEnabled = true;
146             GravityEnabled = true;
147             FlightComputerEnabled = false;
148         }
149
150         //PlanetGravity(MarsPos);
151         //Information
152         //------------------------------------------------------------------
              ------------
153         MercuryAcceleration = new MyVector3(AccelerationX, AccelerationY,
              AccelerationZ);
154         RocketForce = PhysicsLibrary.ForceCalculation(Mass, Acceleration);
155         //Calculates Angle of Planet relative to the Player
156         AngleFromRocketToPlanet = MyVector2.VectorsToRadians(new MyVector2
              (NormalizedVectorPlanetX, NormalizedVectorPlanetY));
157         //Camera properties
158         //------------------------------------------------------------------
              ------------
159         Cameraoffset = ChangeCamPos();
160         camera1.transform.position = (Translate +
              Cameraoffset).ToUnityVector();
161         //Mesh TRS
162         //------------------------------------------------------------------
              ------------
163         TranslateX = Translate.x;
164         TranslateY = Translate.y;
165         TranslateZ = Translate.z;
166
167         Vector3[] TransformedVertices = new Vector3
              [ModelSpaceVertices.Length];
168         Matrix4By4 T = MyTransform.Translate(Translate.x, Translate.y,
              Translate.z);
169         Matrix4By4 R = MyTransform.Rotation(RotX, RotY, RotZ); //Rotation is
               in radians
170         Matrix4By4 S = MyTransform.Scale(ScaleX, ScaleY, ScaleZ);
171         Matrix4By4 M = MyTransform.TRS(T,R,S);
172         for (int i = 0; i < TransformedVertices.Length; i++)
              {TransformedVertices[i] = M * ModelSpaceVertices[i];}
173         PlayerMesh.mesh.vertices = TransformedVertices;
174         PlayerMesh.mesh.RecalculateNormals();
175         PlayerMesh.mesh.RecalculateBounds();
176     }
177     public MyVector3 ChangeCamPos()
178     {
```

```
179         if (Input.GetKeyDown(KeyCode.C))
180         {
181             CameraPos++;
182             if (CameraPos == 1){ Cameraoffset = new MyVector3(0, 3, 0);}
183             if (CameraPos == 2){ Cameraoffset = new MyVector3(0, -3,
                  0);camera1.transform.Rotate(-180.0f, 0, 0);}
184             if (CameraPos == 3){ Cameraoffset = new MyVector3(0, 0,
                  25);CameraPos = 0;}
185         }
186         return Cameraoffset;
187     }
188     public void PlanetGravity(MyVector3 planetpos)
189     {
190         //Checks if Translate is within sphere boundaries
191         MyVector3 PlanetRocketDistance = new MyVector3(planetpos.x +
              Translate.x, planetpos.y + Translate.y, planetpos.z +
              Translate.z);
192         //Applies Gravity Acceleration
193         NormalizedVectorPlanet = PlanetRocketDistance.NormalizeVector();
194         Acceleration -= NormalizedVectorPlanet * Time.deltaTime;
195         Translate += Acceleration * Time.deltaTime;
196 
197     }
198 
199     public void CapAcceleration()
200     {
201         if (Acceleration.y >= -10.0f * Mathf.Cos(-RotZ) ||
202             Acceleration.y >= 10.0f  * Mathf.Cos(-RotZ) ||
203             Acceleration.x >= -10.0f * Mathf.Sin(-RotZ) ||
204             Acceleration.x >= 10.0f  * Mathf.Sin(-RotZ) ||
205             Acceleration.z >= -10.0f * Mathf.Sin(-RotZ) ||
206             Acceleration.z >= 10.0f  * Mathf.Sin(-RotZ)
207             )
208         {
209             if (Acceleration.x <= -10.0f)
210             {
211                 Acceleration.x = -10.0f;
212             }
213             if (Acceleration.x >= 10.0f)
214             {
215                 Acceleration.x = 10.0f;
216             }
217             if (Acceleration.y <= -10.0f)
218             {
219                 Acceleration.y = -10.0f;
220             }
221             if (Acceleration.y >= 10.0f)
222             {
223                 Acceleration.y = 10.0f;
224             }
225             if (Acceleration.z <= -10.0f)
226             {
227                 Acceleration.z = -10.0f;
```

```
228             }
229             if (Acceleration.z >= 10.0f)
230             {
231                 Acceleration.z = 10.0f;
232             }
233         }
234     }
235     public void CapVelocity()
236     {
237         if (Velocity.y >= -10.0f * Mathf.Cos(-RotZ) ||
238             Velocity.y >= 10.0f * Mathf.Cos(-RotZ) ||
239             Velocity.x >= -10.0f * Mathf.Sin(-RotZ) ||
240             Velocity.x >= 10.0f * Mathf.Sin(-RotZ) ||
241             Velocity.z >= -10.0f * Mathf.Sin(-RotZ) ||
242             Velocity.z >= 10.0f * Mathf.Sin(-RotZ)
243             )
244         {
245             if (Velocity.x <= -10.0f)
246             {
247                 Velocity.x = -10.0f;
248             }
249             if (Velocity.x >= 10.0f)
250             {
251                 Velocity.x = 10.0f;
252             }
253             if (Velocity.y <= -10.0f)
254             {
255                 Velocity.y = -10.0f;
256             }
257             if (Velocity.y >= 10.0f)
258             {
259                 Velocity.y = 10.0f;
260             }
261             if (Velocity.z <= -10.0f)
262             {
263                 Velocity.z = -10.0f;
264             }
265             if (Velocity.z >= 10.0f)
266             {
267                 Velocity.z = 10.0f;
268             }
269         }
270     }
271
272
273     public void PlayerInput()
274     {
275         if (Input.GetKey(KeyCode.W))
276         {
277             //Acceleration.x -= 2 * Mathf.Sin(RotZ);
278             //Acceleration.y += 2 * Mathf.Cos(RotZ);
279             //Acceleration.z += 2 * Mathf.Sin(RotY);
280             Acceleration += new MyVector3(Mathf.Sin(-RotZ),Mathf.Cos
```

```
                (RotZ),Mathf.Sin(RotY))/50.0f;
            Translate += Acceleration * Time.deltaTime;
            Wselected = true;
        }
        if (Input.GetKey(KeyCode.UpArrow))   {RpmY += Time.deltaTime;}
        if (Input.GetKey(KeyCode.DownArrow)) {RpmY -= Time.deltaTime;}
        if (Input.GetKey(KeyCode.LeftArrow)) {RpmZ += Time.deltaTime;}
        if (Input.GetKey(KeyCode.RightArrow)){RpmZ -= Time.deltaTime;}

    }
    //public void FlightComputer(MyVector3 StartPoint,MyVector3
      EndPoint,bool TakeCurrentPos)
    //{
    //    if (Input.GetKey(KeyCode.W))
    //    {
    //        Translate += Acceleration * Time.deltaTime;
    //        Acceleration.x -= 2 * Mathf.Sin(RotZ) * Time.deltaTime;
    //        Acceleration.y += 2 * Mathf.Cos(RotZ) * Time.deltaTime;
    //        Acceleration.z += 2 * Mathf.Sin(RotY) * Time.deltaTime;
    //        //Debug.Log(Mathf.Cos(RotZ));
    //        Wselected = true;
    //    }
    //    if (Input.GetKey(KeyCode.UpArrow)) { RotY += Time.deltaTime; }
    //    if (Input.GetKey(KeyCode.DownArrow)) { RotY -= Time.deltaTime; }
    //    if (Input.GetKey(KeyCode.LeftArrow)) { RotZ -= Time.deltaTime; }
    //    if (Input.GetKey(KeyCode.RightArrow)) { RotZ += Time.deltaTime; }

    //}

    public void UpdateText()
    {
        TextAccelerationX.text = "AccelerationX: " + Acceleration.x;
        TextAccelerationY.text = "AccelerationY: " + Acceleration.y;
        TextAccelerationZ.text = "AccelerationZ: " + Acceleration.z;
        //TextThrusterFuel.text =  "ThrusterFuel: " + ThrusterFuel;
        //TextBoosterFuel.text =   "BoosterFuel: " + BoosterFuel;
        TextPosX.text = "PosX: " + Translate.x;
        TextPosY.text = "PosY: " + Translate.y;
        TextPosZ.text = "PosZ: " + Translate.z;

    }
}

```