```
 1 using System;
 2 using JetBrains.Annotations;
 3 using UnityEngine;
 4
 5 /**
 6  * I am implementing a Lunar Lander Game.
 7  *
 8  * What I have done so far:
 9  *
10  * - Basic movement of rocket:
11  *   - Rotation & Thrust
12  * - Collision Detection, and landing hardness
   threshold.
13  */
14
15 public class Player : MonoBehaviour
16 {
17     [SerializeField] private int maxRot;
18
19     private Rigidbody2D rb2;
20     private Vector2 velocity;
21     private Vector2 prevPos;
22     private Vector2 curPos;
23     private float curAngle = 0.0f;
24     private int fuelSupply = 10000;
25     private const float rotIncrement = 0.16f;
26     private const float thrustVelocityIncrement = 0.
   009f;
27
28
29     // Start is called before the first frame update
30     void Start()
31     {
32     }
33
34     private void Awake()
35     {
36         rb2 = GetComponent<Rigidbody2D>();
37         rb2.gravityScale = 0.3f;
38         curPos = rb2.position;
39         prevPos = curPos;
```

```
40          velocity = new Vector2();
41          velocity.x += 1.2f;
42          rb2.velocity = velocity;
43      }
44
45      // Update is called once per frame
46      void Update()
47      {
48          curPos = rb2.position;
49          if (Input.GetKey(KeyCode.W)) AddThrust();
50          if (Input.GetKey(KeyCode.A)) RotateCW();
51          if (Input.GetKey(KeyCode.D)) RotateACW();
52
53          prevPos = curPos;
54          //prevVelocityY = r
55      }
56
57      private void AddThrust()
58      {
59          if (fuelSupply == 0)
60          {
61              return;
62          }
63
64          // todo get rid of framerate dependence.
65          bool posAngle = (curAngle > 0);
66          velocity = rb2.velocity;
67
68          if (posAngle)
69          {
70              velocity.y += (float) Math.Cos(
    ConvertToRadians(curAngle)) * thrustVelocityIncrement
    ;
71              velocity.x += (float) -(Math.Sin(
    ConvertToRadians(curAngle)) * thrustVelocityIncrement
    );
72          }
73          else
74          {
75              velocity.y += (float) (Math.Cos(
    ConvertToRadians(-curAngle)) *
```

```
 75 thrustVelocityIncrement);
 76             velocity.x += (float) (Math.Sin(
    ConvertToRadians(-curAngle)) *
    thrustVelocityIncrement);
 77         }
 78
 79         //velocity.y += 0.01f;
 80         rb2.velocity = velocity;
 81         fuelSupply -= 1;
 82         Debug.Log(fuelSupply);
 83         //rb2.velocity.x = velocityX;
 84     }
 85
 86     public double ConvertToRadians(double angle)
 87     {
 88         return (Math.PI / 180) * angle;
 89     }
 90
 91     private void RotateCW()
 92     {
 93         if (curAngle < -90)
 94         {
 95             return;
 96         }
 97
 98         rb2.transform.Rotate(0, 0, -rotIncrement,
    Space.Self);
 99         curAngle -= rotIncrement;
100         Debug.Log(curAngle);
101     }
102
103     private void RotateACW()
104     {
105         if (curAngle > 90)
106         {
107             return;
108         }
109
110         rb2.transform.Rotate(0, 0, rotIncrement,
    Space.Self);
111         curAngle += rotIncrement;
```

```
112             Debug.Log(curAngle);
113         }
114
115         [UsedImplicitly]
116         public void Landed()
117         {
118             if (rb2.velocity.y < -0.5f)
119             {
120                 Debug.Log(rb2.velocity.y + "Bang!!!");
121                 //SceneManager.LoadScene("SampleScene
    "); todo uncomment when done
122             }
123             else if (rb2.velocity.y < -0.25f)
124             {
125
126                 Debug.Log(rb2.velocity.y + "Hard Landing
    !");
127                 //SceneManager.LoadScene("SampleScene
    "); todo uncomment when done
128
129             }
130             else
131             {
132                 Debug.Log(rb2.velocity.y + "BUTTER :)");
133                 //SceneManager.LoadScene("SampleScene
    "); todo uncomment when done
134             }
135         }
136 }
```

```
fileFormatVersion: 2
guid: 3449cafba28941949b3beb8dceb64c02
MonoImporter:
  externalObjects: {}
  serializedVersion: 2
  defaultReferences: []
  executionOrder: 0
  icon: {instanceID: 0}
  userData:
  assetBundleName:
  assetBundleVariant:

```

File - C:\Users\micha\Lunar Lander\Assets\Scripts\CollisionDetector.cs

```
 1 using System.Collections;
 2 using System.Collections.Generic;
 3 using UnityEngine;
 4 using UnityEngine.SceneManagement;
 5
 6 public class CollisionDetector : MonoBehaviour
 7 {
 8     private List<GameObject> Colliders = new List<
   GameObject>();
 9
10     private void OnTriggerEnter2D(Collider2D col)
11     {
12         SendMessage("Landed");
13         Debug.Log("Hit msg sent");
14     }
15
16     // private void OnTriggerExit2D(Collider2D other)
17     // {
18     //     if (Colliders.Contains(other.gameObject))
   Colliders.Remove(other.gameObject);
19     // }
20 }
```

```
fileFormatVersion: 2
guid: ee25d458ce8cdbc4a96f0e081ef75bfa
MonoImporter:
  externalObjects: {}
  serializedVersion: 2
  defaultReferences: []
  executionOrder: 0
  icon: {instanceID: 0}
  userData:
  assetBundleName:
  assetBundleVariant:

```