# Games Programming Workshop 02

Contents

[Games Programming Workshop 01 1](#_Toc117034187)

[Exercise 1 2](#_Toc117034188)

# Exercise 1

In this exercise, you will create a 2D ‘Gravity Switching’ Platformer. We will add objects to the game that repeat what we did in Workshop 1, and we will continue use basic geometric objects as placeholders for more complex sprites.

1. In the Unity Hub app,

A blue screen with white text

Description automatically generated with medium confidence

1. Create a new 2D (Core) Project. Name the project whatever you like, though I have called mine 2D Gravity Platformer (If you are on a Uni PC, save this to the K: Drive)

Graphical user interface, application

Description automatically generated

1. Right click in the Assets folder in the Content Manager and choose Create > Folder

Graphical user interface, application

Description automatically generated

1. Add scripts

Graphical user interface, application, Teams

Description automatically generated

1. Create a new folder called Graphics and then add a 2D Square



1. Create a **Physics** folder
2. Add a **Physics 2D Material** from the **2D** right-click menu
3. Create a Prefab folder and add a Prefab for Enemy, Player and Spike
4. Add a **Sprite Renderer** component to each Prefab and set the sprite to **Square**
5. Set the colour of each Prefab to be different

A picture containing graphical user interface

Description automatically generated

1. Add **Enemy Controller script** to Enemy Prefab
2. Add **Box Collider 2D** to the Enemy Prefab
3. Add this material to the **Box Collider 2D** in Enemy

Graphical user interface, text, application

Description automatically generated

1. Add a **Rigidbody 2D** to the Enemy

Graphical user interface, text, application

Description automatically generated

1. Repeat all these steps for the **Player**
2. Change **Transform Scale** of **Player** to x0.5, y1, z1 to make it skinny

Graphical user interface, application

Description automatically generated

1. Change **Transform** of **Spike** to x3, y1, z1 to make it wide

Graphical user interface, application

Description automatically generated

1. Add the **Spike Controller script** and **Box Collider 2D**, but this time do not add a Rigidbody as the spikes are stationary and will not need physics.
2. Add a new prefab to the Prefabs folder and call it **Bounds.**
3. Add a new Tag called Bound and tag the Bounds prefab with this.

A screenshot of a computer

Description automatically generated with low confidence

1. Add a Sprite Renderer component and assign the **Square** graphic, setting the colour to dark grey.
2. Add a **Box Collider 2D** and assign the **2d Physics material**
3. Add a **Rigidbody 2D** and change the **Body Type** to **Kinematic**

Graphical user interface, application

Description automatically generated

1. Select the Main Camera object in the hierarchy and set the Background property to a light sky blue colour.

Graphical user interface, application

Description automatically generated

1. Drag instances of the Bounds prefab into the scene and resize them to build the level. As this will be a gravity switching game your test level should be quite long. Also add the Player, an Enemy and some Spikes to the level layout.

Timeline

Description automatically generated with medium confidence

1. Create a Coin Prefab and Add the **Sprite Renderer** and **Square**. Transform scaled 0.5. Add a **Box Collider 2D,** then add the **CoinController** and **Rotator** scripts**.**

A picture containing graphical user interface

Description automatically generated

1. Se the Coin **Rigidbody 2D** to have a **Mass** of 0.0001 and a **Gravity Scale** of 0. This will prevent it from falling to the ground and also stop the player from bouncing off it.
2. Add Coins to your level

Qr code

Description automatically generated with low confidence

1. Add **Legacy Text UI** component to the Hierarchy



1. Position the text for **Score** (In game Mode), rename to **Score Text**

Graphical user interface

Description automatically generated

1. Duplicate as **Game Over Text** and position in the middle of the screen.

Graphical user interface

Description automatically generated

1. Open the **PlayerController** script in **VS Code**

using System;

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class PlayerController : MonoBehaviour

{

    public delegate void OnHitSpikeAction();

    public delegate void OnHitCoinAction();

    public OnHitSpikeAction OnHitSpike;

    public OnHitCoinAction OnHitCoin;

    float speed = 1000;

    Vector3 leftBound;

    Vector3 rightBound;

    bool canJump;

    // Start is called before the first frame update

    void Start()

    {

        if (Physics2D.gravity.y > 0)

        {

            Physics2D.gravity \*= -1;

        }

    }

    // Update is called once per frame

    void Update()

    {

        ProcessInput();

    }

    private void ProcessInput()

    {

        if(Input.GetKey("left") || Input.GetKey("a"))

            this.GetComponent<Rigidbody2D>().AddForce(Vector2.left \* speed \* Time.deltaTime);

        if(Input.GetKey("right") || Input.GetKey("d"))

            this.GetComponent<Rigidbody2D>().AddForce(Vector2.right \* speed \* Time.deltaTime);

        if(Input.GetKeyDown("up") || Input.GetKey("w"))

            InvertGravity();

    }

    private void InvertGravity()

    {

        Physics2D.gravity \*= -1;

    }

    public void OnCollisionEnter2D(Collision2D collision)

    {

        if(collision.gameObject.GetComponent<SpikeController>() != null)

        {

            if (OnHitSpike != null)

            {

                OnHitSpike();

            }

        }

        else if(collision.gameObject.GetComponent<EnemyController>() != null)

        {

            if(OnHitSpike != null)

            {

                OnHitSpike();

            }

        }

        else if(collision.gameObject.GetComponent<CoinController>() != null)

        {

            GameObject.Destroy(collision.gameObject);

            if(OnHitCoin != null)

            {

                OnHitCoin();

            }

        }

    }

}

1. Open the **EnemyController** in **VSCode**
2. using System.Collections;
3. using System.Collections.Generic;
4. using UnityEngine;
5. public class EnemyController : MonoBehaviour
6. {
7. public float PatrolSpeed = 300f;
8. public float PatrolDuration = 3f;
9. float patrolTimer;
10. Vector2 direction;
11. // Start is called before the first frame update
12. void Start()
13. {
14. this.direction = Vector2.left;
15. }
16. // Update is called once per frame
17. void Update()
18. {
19. patrolTimer += Time.deltaTime;
20. if(patrolTimer >= PatrolDuration)
21. {
22. this.GetComponent<Rigidbody2D>().velocity = Vector2.zero;
23. patrolTimer = 0;
24. direction \*= -1;
25. }
26. this.GetComponent<Rigidbody2D>().AddForce(this.direction \* PatrolSpeed \* Time.deltaTime);
27. }
28. }
29. Open the **Rotator** script in **VSCode**
30. using System.Collections;
31. using System.Collections.Generic;
32. using UnityEngine;
33. public class Rotator : MonoBehaviour
34. {
35. public float X\_Speed = 0f;
36. public float Y\_Speed = 0f;
37. public float Z\_Speed = 400f;
38. // Start is called before the first frame update
39. void Start()
40. {
42. }
43. // Update is called once per frame
44. void Update()
45. {
46. this.transform.localEulerAngles += new Vector3(
47. this.transform.localEulerAngles.x + X\_Speed,
48. this.transform.localEulerAngles.y + Y\_Speed,
49. this.transform.localEulerAngles.z + Z\_Speed
50. ) \* Time.deltaTime;
51. }
52. }
53. Back in Unity, set the Rotator Script on the Coin Prefab to have X\_Speed=0, Y\_Speed=0, Z\_Speed = 200
54. Also, try adding the Rotator script to a Bounds object and rotate it on the Z axis to create a spinning obstacle.
55. Open the **GamesceneManager** in **VSCode**
56. using System.Collections;
57. using System.Collections.Generic;
58. using UnityEngine;
59. using UnityEngine.UI;
60. using UnityEngine.SceneManagement;
61. using System;
62. public class GameSceneManager : MonoBehaviour
63. {
64. public Camera MainCamera;
65. public Text ScoreText;
66. public Text GameOverText;
67. public PlayerController player;
68. int score;
69. float gameTimer;
70. bool gameOver;
71. // Start is called before the first frame update
72. void Start()
73. {
74. Time.timeScale = 1; //start time
75. player.OnHitSpike += OnGameOver;
76. player.OnHitCoin += OnHitCoin;
77. ScoreText.enabled = true;
78. GameOverText.enabled = false;
79. }
80. // Update is called once per frame
81. void Update()
82. {
83. MainCamera.transform.position = new Vector3(
84. Mathf.Lerp(MainCamera.transform.position.x, player.transform.position.x, Time.deltaTime \* 10),
85. Mathf.Lerp(MainCamera.transform.position.y, player.transform.position.y, Time.deltaTime \* 10),
86. MainCamera.transform.position.z);
87. if(gameOver)
88. {
89. if(Input.GetKeyDown("r"))
90. {
91. SceneManager.LoadScene (SceneManager.GetActiveScene().name);
92. }
93. return; // Skip the following lines if GameOver
94. }
96. ScoreText.text = "Score: " + score;
97. if(player.transform.position.y < -10)
98. OnGameOver();
99. }
101. private void OnHitCoin()
102. {
103. this.score += 100;
104. }
105. private void OnGameOver()
106. {
107. gameOver = true;
108. ScoreText.enabled = false;
109. GameOverText.enabled = true;
110. GameOverText.text = "Game Over!\nPress R to Resart";
111. Time.timeScale = 0; //stop time
112. }
114. }
115. Back in **Unity**, Add a new Empty GameObject to the Scene Hierarchy and drag the Main Camera into it.

Graphical user interface, text, application

Description automatically generated

1. Add the Game Scene Manager script component to the Scene Manager object



1. Set the Public Properties to the objects in your scene

Graphical user interface, application

Description automatically generated

**Extensions**

* Extend the level layout
* Add a prefab for a Goal, which completes the level when you collide with it.