Activity 4: Pong Game (attempt this on your own)

In this activity we are going to implement the classic Pong game. There are two players in this game. The aim of the game is to hit the ball with the racket and return the ball to the other player. Each hit earns a point. This game is developed using Unity 2018.

You can try out different versions of this classic game at www.ponggame.org.

Procedure:

1. Start Unity Hub
2. New a project and name it PongGame.
   Under Template, select 2D.
3. Under Project window, go to the assets->scene.
4. Rename the Sample Scene to PongGame
5. Change the game screen solution
   a. Go to Scene View-> Game panel
   b. Change the game screen solution to "Standalone (1024x768)".
6. Change the background colour
   a. Go to Hierarchy window, right click on Main Camera
   b. On the inspector window, go to background and change it to a brighter colour. (This is to create a good contrast, as the walls, rackets and balls are all black.)
7. Create an Image, Scripts and Materials folder
   a. Go to Project window; create three new folders "Images, "Scripts" and "Materials" under "Assets" folder.
8. Change the images import setting.
   a. Go to the "Images" folder, select all the images.
   b. Then go to the Inspector window, change the Pixels Per Unit to 1. (Pixels Per Unit value of 1 means that 1 x 1 pixels will fit into exactly one unit in the game world)

9. Adding the four walls of the Pong game.
    a. Drag two VWalls and two HWalls into the game scene to form the four walls of the Pong game.
    b. Make sure the camera is somewhere in the center.
    c. Rename the four walls with the appropriate name; VWallRight, VWallLeft, HWallTop, HWallBottom.
    d. You should get something like this.

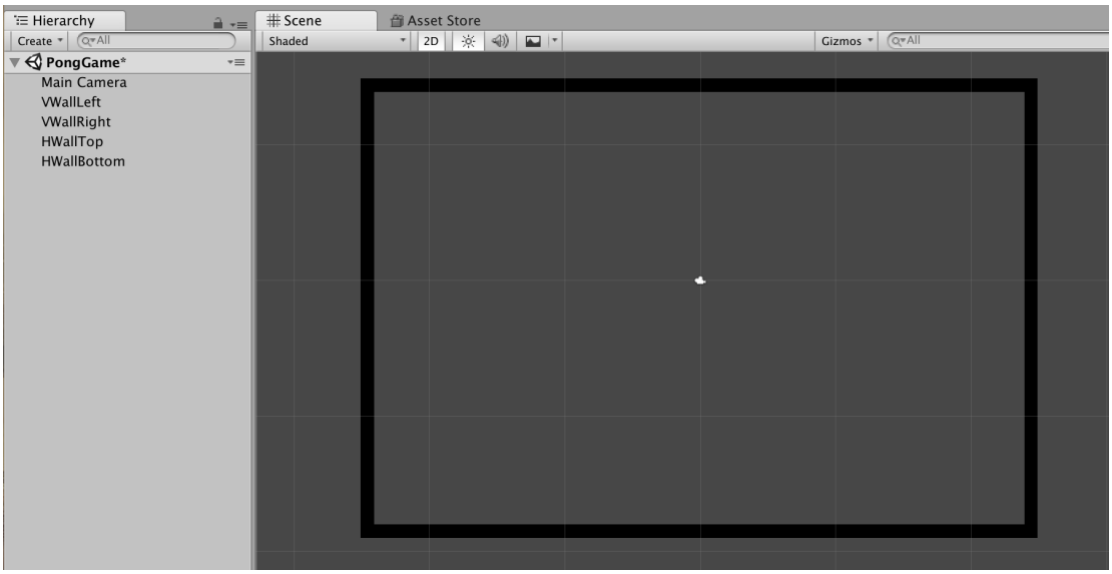

Figure 3.3 Activity 3-1

10. Changing the wall Physics.

    Currently the 4 walls are just images. In the game, we need to make the ball bounces off the 4 walls. To do this we need to add in Box Collider components for each wall.

    Go to the Hierarchy window; select all the walls.

    Go to Inspector window; Click on Add Component->2D Physics->Box Collider 2D.

    Take a look at the Scene and you will find that all the walls have a green outline. The green lines are the collider and are only visible in the Scene.

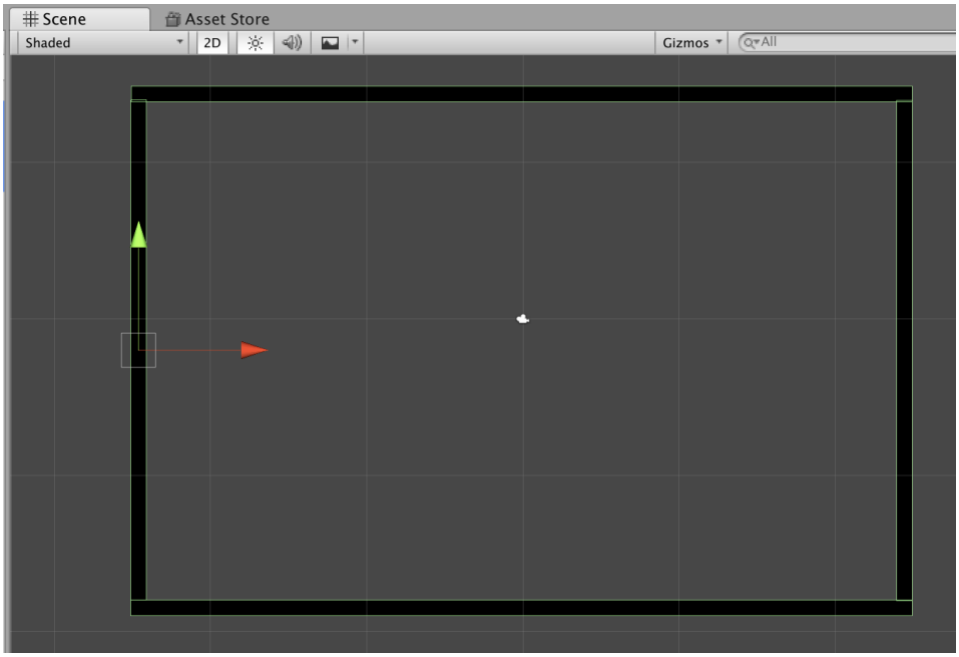

Figure 3.4 Activity 3-2

11. Adding the two Rackets
    a. Drag two Rackets to the Scene.
    b. Rename the two Rackets with the appropriate name; RacketRight and RacketLeft.
    c. Similarly later in the game we need to ball bounce off the two Rackets. We need to add in Box Collider 2D component to both Rackets.
    d. Go to the Hierarchy window; select the two Rackets.
    e. Go to Inspector window; Click on Add Component->2D Physics->Box Collider 2D.
    f. However since the two Rackets are moving objects, we need to add in Rigidbody 2D component for the collision detection to works.
    g. With the two Rackets in the Hierarchy window selected, go to Inspector window, click on Add Component->2D Physics-Rigidbody 2D.
    h. Note: If an object does not has Rigidbody2D then Unity assumes the object non-moving. Therefore Unity will not check for collisions between two non-moving objects, because they will never collide. This is to improve the efficient of the game.

i. In the Inspector window, Rigid 2D section of the two Rackets:
j. Set the Gravity Scale to 0 to disable Gravity as there is no gravity in the game.
k. Enable Freeze Rotation Z as the two Rackets should not rotate.
l. Set Collision Detection to Continous and enable Interpolation. (for a more precise Physics simulation)
m. You should get something like this in the Racket's Rigid 2D section.

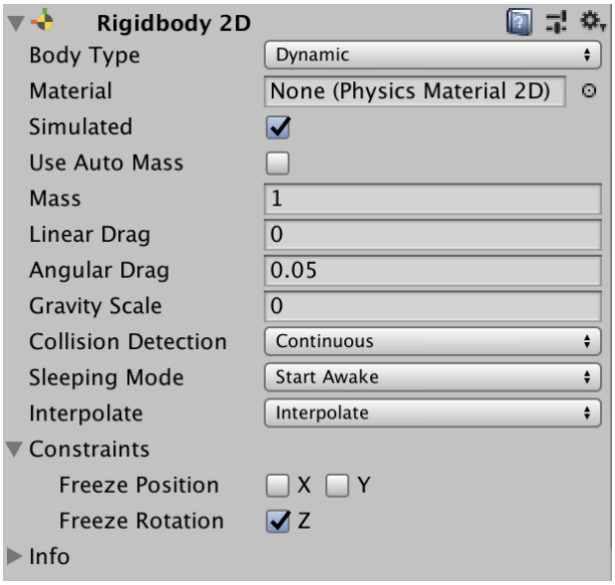

Figure 3.5 Activity 3-3

12. Script for Racket control

a. Go to Project window, under the Scripts folder, create a new C# script and name it Racket.
b. Open the C# script for editing.
c. The Start function is called automatically for one time when the game starts.
d. The Update function is called automatically around 60 times per second.
e. For codes that deal with Physics, it is better to put it under FixedUpdate function as it is called at a fixed time interval.

f. Update the Racket script with the following codes:

Racket.cs

No selection

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class Racket : MonoBehaviour {
6     public float speed = 30;
7     public string  axis =  "Vertical";
8
9     void FixedUpdate(){
10        float v = Input.GetAxisRaw(axis);
11        GetComponent<Rigidbody2D>().velocity = new Vector2(0, v)*speed;
12    }
13 }
```

Figure 3.6 Activity 3-4

g. Input.GetAxisRaw() function is to get the user inputs from various sources (e.g. mouse, keyboard and joysticks) and is configurable via the Project Setting. The default "Vertical" axis setting returns the value 1 for W and up arrow and returns the value -1 for S and down arrow. If none of the keys are pressed, it returns a value of 0. The movement of the Racket is determined by the velocity what can be modelled using a Vector.

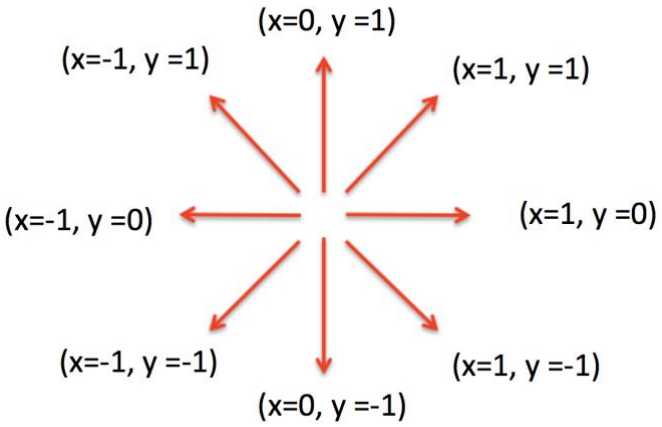

Figure 3.7 Activity 3-5

h. Drag the Racket Script from the Project Window to the two Rackets in the Inspector window as one of the component.

i. You should get something like this:

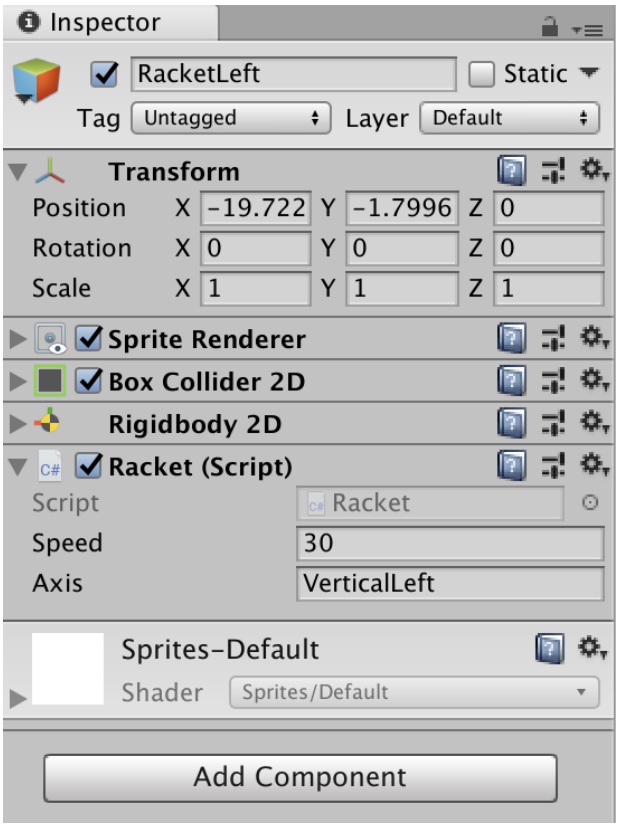

Figure 3.8 Activity 3-6

13. Test the game.
    a. Click on the play button.
    b. You can now control the two Rackets using the w,s or up, down keys.
    c. The Rackets should be blocked by the top and bottom walls.
    d. However we need to have separate controls for each Rackets.

14. Axis Control for left racket

a. Go to Unity toolbar menu, select Edit->Project Setting->Input
b. Go to Inspector window; select "Vertical". This is the setting for the function Input.GetAxisRaw() "Vertical" setting.
c. Rename it to "VerticalLeft".
d. Set Negative Button to s.
e. Set Positive Button to w.
f. Clear both Alt Negative Button and Alt Positive Button.
g. Set Joy Num to "Joystick 1"
h. You should have something like this:

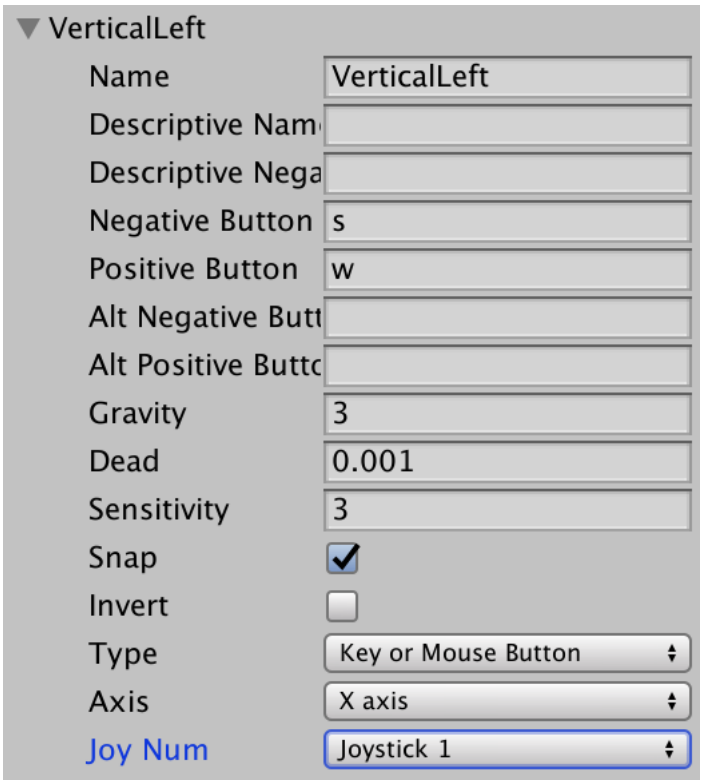

Figure 3.9 Activity 3-7

15. Axis Control for right racket

a. In the Inspector window of the Axis control, change the size from 18 to 19.
b. A new “Cancel” axis control will appear at the bottom.
c. Click on the new ”Cancel” axis control:
d. Rename to “VerticalRight”
e. Set Negative Button to down.
f. Set Positive Button to up.
g. Clear both Alt Negative Button and Alt Positive Button.
h. Set Joy Num to “Joystick 2”
i. You should have something like this:

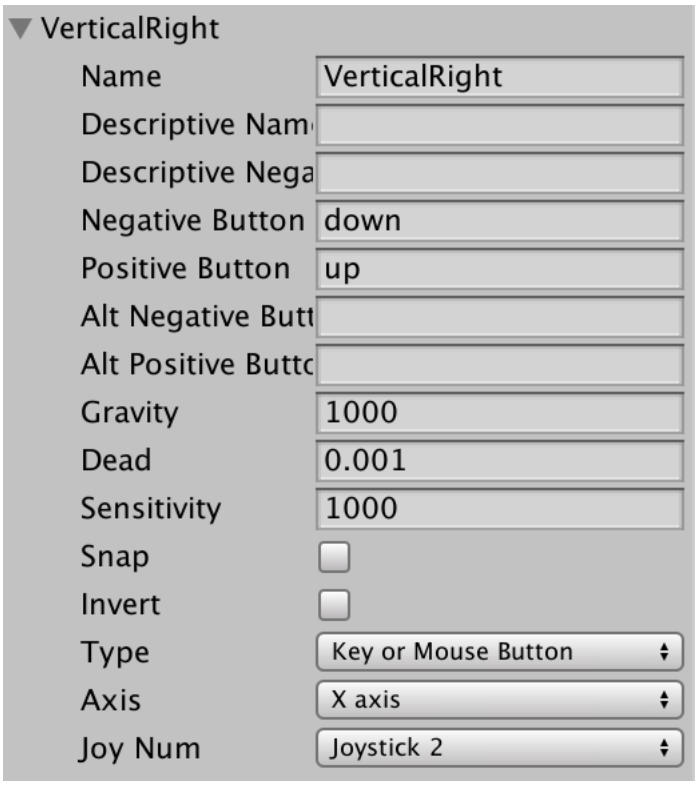

Figure 3.10 Activity 3-8

16. Attached Axis Control to the two Rackets

    a. Go to Hierarchy window, click on "RacketLeft".
    b. Go to Inspector window, under the Racket(Script) section, change the Axis value to "VerticalLeft".
    c. Go to Hierarchy window, click on "RacketRight".
    d. Go to Inspector window, under the Racket(Script) section, change the Axis value to "VerticalRight".
    e. Test the game and you should be able to control each rackets separately.

17. Adding the Ball to the game scene.

    a. Go to Project window, under the "Images" folder, drag the Ball to the game scene.
    b. Make sure that the Ball is in the middle of the two Rackets.
    c. Add the Box Collider 2D component to the Ball for collision detection.
    d. To enable the ball to bounce off the walls and rackets, we need to add Physics Material to the Collider
        i. Go to Project, Material folder
        ii. Right click->Create->Physics2D Material.
        iii. Name it as BallMaterial.
        iv. Go to the Inspector, set Bounciness value to 1.
        v. Go to the Project window and drag the BallMaterial into the Material slot of the Ball's Collider.

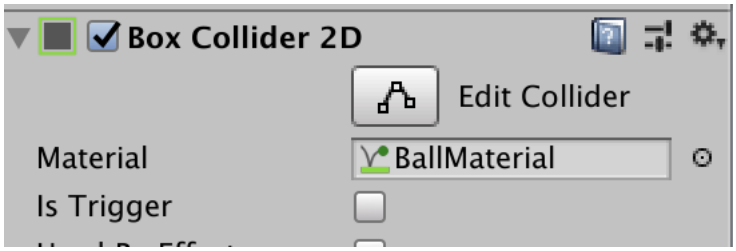

Figure 3.11 Activity 3-9

e. As the Ball is a moving object, we need to add Rigidbody2D to the Ball.

i. Set Gravity Scale to 0.
ii. Set Ball Mass to 0.001 (very small) so the Rackets will not get oushed away when collision happens.
iii. Enable Free Rotation z.
iv. Set Continous Collision to Continous.
v. Set Interpolate to Interpolate.
vi. You should get something like this:

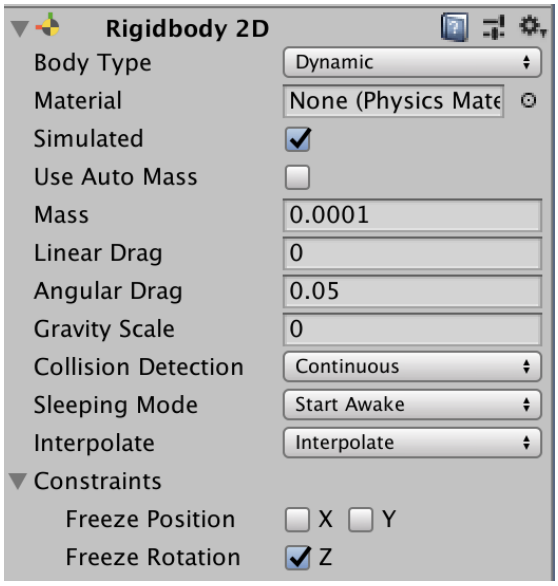

Figure 3.12 Activity 3-10

18. Script for Ball control
    a. Go to Project window, under the Scripts folder, create a new C# script and name it Ball.
    b. Open the C# script for editing.
    c. Update the Racket script with the following codes:

```
1  using UnityEngine;
2
3  public class Ball : MonoBehaviour {
4
5      public float speed = 30;
6
7      void Start(){
8          // Initial Ball Velocity
9          GetComponent<Rigidbody2D>().velocity = Vector2.right * speed;
10     }
11
12     // The function OnCollisionEnter2D is called automatically
13     // when there is a collision
14     // object c has the collision information.
15     // c.gameObject - the racket object
16     // c.transform.position - the racket's position
17     // c.collider - the racket's collider
18     void OnCollisionEnter2D(Collision2D c){
19
20         float hitPos = hitPosition(transform.position,
21                                    c.transform.position,
22                                    c.collider.bounds.size.y);
23
24         // Hit the left Racket
25         if (c.gameObject.name == "RacketLeft"){
26             // Calculate direction, make length=1 via .normalized
27             Vector2 direction = new Vector2(1, hitPos).normalized;
28             // Set Velocity with dir * speed
29             GetComponent<Rigidbody2D>().velocity = direction * speed;
30
31         }
32
33         // Hit the right Racket
34         if (c.gameObject.name == "RacketRight"){
35             // Calculate direction, make length=1 via .normalized
36             Vector2 direction = new Vector2(-1, hitPos).normalized;
37             // Set Velocity with dir * speed
38             GetComponent<Rigidbody2D>().velocity = direction * speed;
39
40         }
41     }
42
```

```
43      //determine which part of the racket was hitted
44      // 1 <- at the top of the racket
45      // 0 <- at the middle of the racket
46      // -1 <- at the bottom of the racket
47      float hitPosition(Vector2 ballPosition, Vector2 racketPosition,
48                  float racketHeight){
49
50          return (ballPosition.y - racketPosition.y) / racketHeight;
51      }
52  }
```

Figure 3.13 Activity 3-11

d. Drag the Ball Script from the Project Window to the Ball in the Inspector window as one of the component.

e. You should get something like this:

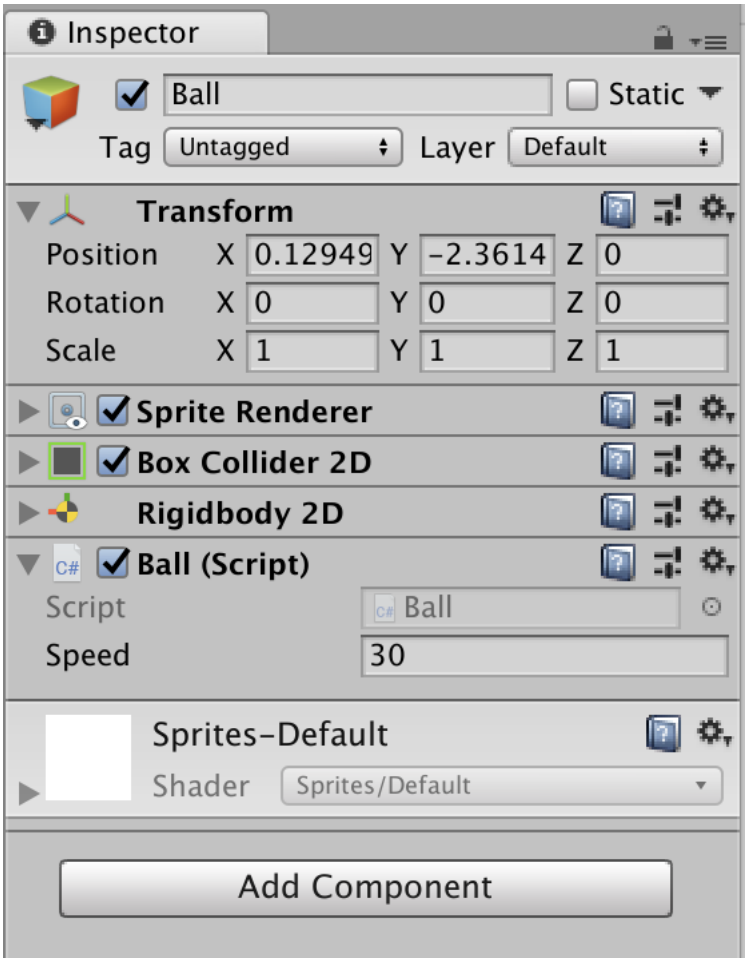

Figure 3.14 Activity 3-12

19. Run and test the game.

20. Attempt the following tasks on your own.
    a. Create a Starting scene with instruction
    b. Include a game score for each player.
    c. The game ends after 30 secs.
    d. Create an Ending scene
    e. When the gamer is done with the game, show the Ending scene.
    f. The gamer has the option to replay or exit the game.