**Project-Based Learning Report**

on

**“Create a simple 2D game where players can control paddles to bounce a ball back and forth Using Unity.”**

Submitted in the partial fulfillment of the requirements.

For the Project-based learning **Augmented Reality & Virtual Reality**

In

Electronics & Communication Engineering

By

**Aashish Kumar 2014111065**

**Ankush Kumar Singh 2014111769**

**Ishit Verma 2014111135**

Under the guidance of Course In-charge

**Prof.** **Dr. M.V. Patil**

Bharati Vidyapeeth

(Deemed to be University)

College of Engineering,

Pune – 411043

**Academic Year: 2023-24**

**Bharati Vidyapeeth**

**(Deemed to be University)**

**College of Engineering,**

**Pune – 411043**

**DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING**

**CERTIFICATE**

This is to be Certified that the Project Based Learning report entitled,

**“Create a simple 2D game where players can control paddles to bounce a ball back and forth Using Unity.”** work is done

by

**Aashish Kumar 2014111065**

**Ankush Kumar Singh 2014111769**

**Ishit Verma 2014111135**

In partial fulfillment of the requirements for the award of credits for Project Based Learning (PBL) in **Augmented Reality & Virtual Reality**

Bachelor of Technology Semester-VII, Electronics and Communication Engineering.

**Date: 14th October 2023**

**Prof. Dr. M.V. Patil Dr. Arundhati A. Shinde**

**Course In charge Professor & Head**

**Electronics and Communication**

**Engineering**

**Index**

|  |  |  |
| --- | --- | --- |
| **Sr. No.** | **Name of the topic** | **Page No.** |
| 1. | Problem statement with Solution | 1-1 |
| 2. | Introduction to Unity | 2-5 |
| 3. | Software Used | 6-7 |
| 4. | Result and Analysis | 8-15 |
| 5. | Outcome & Conclusion | 16-16 |

1. **Problem Statement**

**Problem Statement:**

Design and develop a 2D Unity game resembling Pong where players control paddles to bounce a ball, emphasizing precise controls, realistic physics, and a scoring system. The goal is to create an engaging, classic arcade experience with a polished user interface.

**Solution:**

Creating a 2D game in Unity where players control paddles to bounce a ball is a classic rendition of the game Pong. The essential steps to achieve this are as follows:

1. Unity Setup: Begin by setting up Unity and creating a new 2D project.

2. Game Elements: Design the game assets - two paddles and a ball, which can be basic 2D sprites or shapes.

3. Scene Setup: Position the paddles at the top and bottom of the screen, placing the ball in the center.

4. Player Controls: Implement scripts to enable player control of the paddles, typically using keyboard input.

5. Physics: Attach Rigidbody2D components to the paddles and the ball for realistic physics interactions.

6. Ball Movement: Develop a script to move the ball initially in a certain direction and handle collision detection for bouncing off the paddles.

7. Scoring System: Create a scoring mechanism to track player scores, displaying them on the screen.

8. Game Over Conditions: Set criteria for ending the game, such as reaching a specific score, and create a game over screen.

9. UI Elements: Include UI components to show scores, game over messages, and restart options.

10. Testing and Debugging: Thoroughly test the game for any issues and refine it as needed.

11. Build and Export: Compile the game for your target platform (e.g., Windows, Android, iOS).

12. Publish or Share: Share your game with others or consider publishing it on the platform of your choice.

**2. Introduction to Unity**

**Introduction to Making a Game in Unity:**

Unity is a powerful and versatile game development engine that empowers developers to create games for various platforms, including PC, mobile, console, and more. Whether you're a beginner or an experienced developer, Unity provides a user-friendly interface and a wealth of tools to help you bring your game ideas to life. In this introduction, we'll take a brief look at the steps to make a game in Unity and describe some essential header functions commonly used in Unity development.

**Getting Started in Unity:**

1. **Install Unity:** The first step is to download and install Unity Hub, which is a management tool for Unity versions and projects. You can then choose the Unity version suitable for your project and install it.
2. **Create a New Project:** Open Unity Hub, create a new 2D or 3D project, and specify the project settings like platform, resolution, and aspect ratio.
3. **Unity Interface:** Unity's interface consists of several key components:
   * **Scene View:** This is where you design your game world by placing and arranging objects.
   * **Game View:** Shows what the game will look like when played.
   * **Hierarchy:** Lists all the objects in your scene.
   * **Project:** Contains your assets (e.g., textures, models, scripts).
   * **Inspector:** Provides details and properties of the selected object.
4. **Assets:** Import assets (e.g., textures, models, audio files) into your project using the "Assets" folder in the "Project" window.
5. **Creating GameObjects:** In Unity, everything is an object known as a GameObject. You can create GameObjects by right-clicking in the "Hierarchy" or "Scene" view and selecting "Create Empty" or other available options.

**Header Functions in Unity:**

1. **Start():** The Start() function is called once when a script-enabled object (Mon Behaviour) is initialized or enabled in the scene. It's often used for initialization tasks, such as setting initial variables or configuring components.
2. **Update():** The Update() function is called once per frame and is where most of your game's dynamic behaviour, input handling, and game logic reside. Anything that needs to update continuously should be placed in this function.
3. **Awake():** Awake() is called when a script-enabled object is initialized, even before Start(). It's typically used for setting up references to other objects or components, especially when you need to ensure they are available before Start() is called.
4. **OnEnable() and OnDisable():** These functions are called when an object is enabled or disabled in the scene, respectively. They are useful for managing resource allocation and cleanup.
5. **FixedUpdate():** FixedUpdate() is called at fixed time intervals and is often used for physics-related calculations, rigid body interactions, or anything that requires a fixed time step.
6. **LateUpdate():** LateUpdate() is called after all Update() functions have run. It's often used for camera-related operations and ensures that the camera's movement and positioning update after all other objects have moved.

These header functions are essential building blocks of Unity game development, allowing you to control the behaviour of Game Objects and manage game logic effectively. As you become more familiar with Unity, you'll find that these functions play a crucial role in creating interactive and engaging gameplay experiences.

**Introduction to Creating a Ping Pong Game in Unity:**

Ping Pong is a classic and timeless arcade game that has captured the hearts of gamers for generations. Creating a Ping Pong game in Unity is a fantastic way to learn game development fundamentals while delivering a fun and engaging gaming experience. In this introduction, we'll explore the exciting world of building a Ping Pong game in Unity, from designing the game mechanics to coding the paddle and ball interactions.

**The Ping Pong Game Concept:**

Ping Pong, also known as table tennis, is a two-player game where each player controls a paddle to hit a ball back and forth across a table, attempting to score points by making the ball pass the opponent's paddle. The game is simple to understand but challenging to master, making it an ideal choice for a beginner's Unity project.

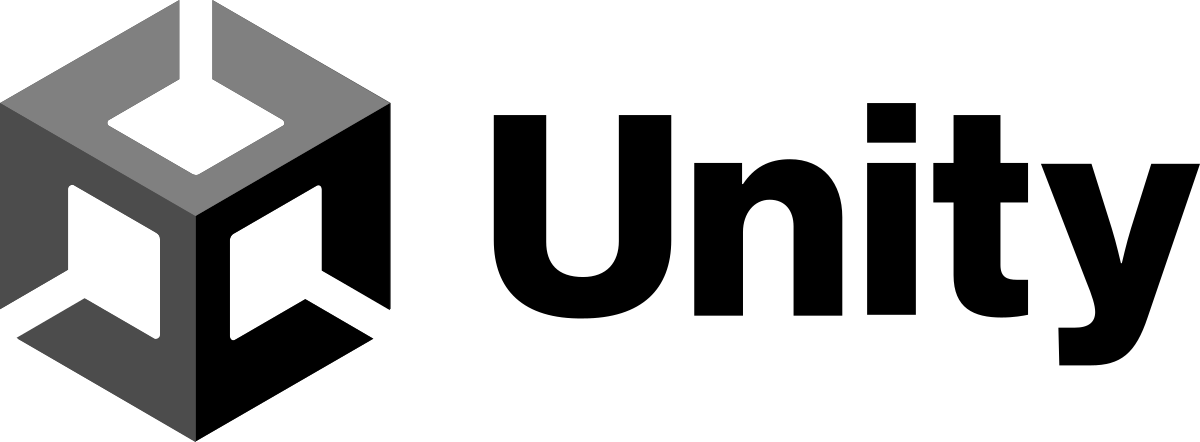
**Key Components of a Unity Ping Pong Game:**

1. **Paddles:** In a Ping Pong game, each player's paddle is controlled by the keyboard or a touch-based input method. Creating responsive and smooth paddle movement is a key aspect of the game's enjoyment.
2. **Ball:** The ball is the central element of the game, bouncing back and forth between the paddles. Proper physics and collision handling for the ball are crucial for realistic gameplay.
3. **Game Rules:** Implementing the rules of Ping Pong, such as scoring, win conditions, and ball speed adjustments, is essential to create an authentic gaming experience.
4. **Game UI:** Designing a user interface to display the score, player names, and game status is necessary for player engagement and feedback.
5. **Sound Effects:** Adding sound effects, like the sound of the ball hitting the paddle or the table, enhances the overall gaming experience.
6. **Game Physics:** Unity's physics engine can be used to create realistic ball movement, including bouncing, reflection, and spin.****

Pong typically involves two paddles and a ball. You'll control one of the paddles, and your opponent (either AI or another player) controls the other paddle. Learn the controls, which are usually simple:

* Move the paddle up: Use the up-arrow key or the "W" key.
* Move the paddle down: Use the down arrow key or the "S" key.
* objective in Pong is to prevent the ball from passing your paddle while trying to score points by getting the ball past your opponent's paddle.
* Use paddle to hit the ball back and forth across the screen.
* The game continues until one player reaches a predetermined score (often 11 or 21 points) or until a time limit is reached.

**3. Software UsedTop of Form**



Unity is a popular and versatile game development engine and platform that is widely used to create video games, simulations, interactive experiences, and even some non-gaming applications. It is known for its ease of use, cross-platform compatibility, and extensive capabilities. Here's an overview of how Unity works and its various applications:

**Working of Unity:** Unity operates on a component-based system and uses a visual development environment, making it accessible to both beginners and experienced developers. Here's a simplified breakdown of how Unity works:

1. **Scene Creation:** Unity projects are organized into scenes, which represent different levels, environments, or segments of a game or application. Within each scene, you can place various game objects, including 3D models, 2D sprites, lights, cameras, and more.
2. **Components:** Game objects in Unity are composed of components, which are reusable and configurable scripts or properties that define an object's behaviour and appearance. Unity provides a wide range of built-in components, and you can also create custom ones using C# scripts.
3. **Scripting:** Unity primarily uses C# for scripting. Developers can write scripts to control game logic, physics, animations, and interactions. These scripts can be attached to game objects as components, allowing for easy integration into the scene.
4. **Physics:** Unity has a robust physics engine that enables realistic interactions between objects. Rigidbody components can be added to game objects to simulate physics, gravity, collisions, and more.
5. **Graphics and Rendering:** Unity supports 2D and 3D graphics rendering, including shaders and effects for creating visually stunning scenes. It also provides tools for optimizing performance and creating visually appealing environments.
6. **Animation:** Unity offers a powerful animation system that allows developers to create complex animations for characters, objects, and UI elements. Animation clips can be created and controlled through the Animation window or code.
7. **Asset Management:** Unity includes a comprehensive asset management system, making it easy to import, organize, and use 3D models, textures, audio, and other resources in your projects.
8. **Cross-Platform Development:** Unity supports deployment to multiple platforms, including Windows, macOS, Linux, iOS, Android, consoles (e.g., PlayStation, Xbox), VR/AR devices, and the web. It simplifies the process of targeting various devices and operating systems.

**Applications of Unity:**

1. **Video Games:** Unity is most famous for its use in game development. It's used to create a wide range of games, from 2D mobile games to complex 3D AAA titles.
2. **Virtual Reality (VR) and Augmented Reality (AR):** Unity is a go-to choose for creating VR and AR experiences, ranging from immersive simulations to training applications and interactive marketing campaigns.
3. **Simulation and Training:** Unity is used for creating simulations in various fields, such as aviation, healthcare, military, and industrial training. It provides a realistic environment for users to practice and learn.
4. **Architectural Visualization:** Architects and designers use Unity to visualize buildings and interiors in real-time, allowing clients to explore and experience designs before construction.
5. **Educational Apps:** Unity is utilized in educational software to create interactive and engaging learning experiences, from interactive science simulations to language learning apps.
6. **Film and Animation:** Unity's real-time rendering capabilities are used in the film and animation industry to create previsualization, animated shorts, and even feature films.
7. **Medical and Healthcare:** Unity is used for medical simulations, patient education, and virtual therapy applications, helping healthcare professionals and patients alike.
8. **Product Visualization:** Companies use Unity to showcase products in a virtual environment, enabling customers to interact with and customize products before making a purchase.

**4. Result & Analysis**

**Step 1:** Set up the Unity Project

Open Unity and create a new 2D project.

**Step 2:** Create the Game Objects

Create a new 2D sprite for the paddles and the ball. You can import your own images or use simple shapes.

Add a Box Collider 2D component to the paddles and a Circle Collider 2D to the ball.

Create two empty GameObjects for the paddles and one for the ball to serve as their transform parents. These will allow you to adjust the paddles' positions relative to the parent's center.

**Step 3:** Design the Scene

Create a 2D background or play area. You can use a simple rectangle sprite or draw it using the Tilemap system in Unity.

Place the paddles and the ball in the scene. Adjust their positions as needed.

**Step 4:** Implement Physics

Select the ball GameObject and add a Rigidbody2D component. This will enable physics interactions.

Adjust the Rigidbody2D's Gravity Scale to 0 to prevent the ball from falling due to gravity.

In the Rigidbody2D component, set the Collision Detection to "Continuous" for more accurate collision detection.

**Step 5:** Write the Paddle Control Script

Create a new C# script called "PaddleController" and attach it to both paddle GameObjects.

In the script, you can use the Rigidbody2D component to move the paddles. Here's a basic example of how to move a paddle using the arrow keys for player 1:

C# Code

using UnityEngine;  
  
public class PaddleController : MonoBehaviour  
{  
 public float speed = 5.0f; // Adjust the paddle speed in the Inspector.  
  
 void Update()  
 {  
 float moveInput = Input.GetAxis("Vertical");  
 Vector2 movement = new Vector2(0, moveInput) \* speed \* Time.deltaTime;  
 transform.Translate(movement);  
 }  
}

Repeat this step for the second paddle, but use a different input axis for player 2 (e.g., "Vertical2").

**Step 6:** Write the Ball Movement Script

Create a new C# script called "BallController" and attach it to the ball GameObject.

In the script, give the ball an initial force to start the game:

C# code

using UnityEngine;  
  
public class BallController : MonoBehaviour  
{  
 public float initialSpeed = 5.0f; // Adjust the initial speed in the Inspector.  
  
 void Start()  
 {  
 // Give the ball an initial force.  
 Vector2 initialDirection = new Vector2(1, Random.Range(-1f, 1f)).normalized;  
 GetComponent<Rigidbody2D>().velocity = initialDirection \* initialSpeed;  
 }  
}

**Step 7:** Implement Collision Detection

Attach a script to an empty GameObject in the scene, called "GameManager," to handle collision detection and scoring.

In the GameManager script, use the OnCollisionEnter2D method to detect collisions between the ball and the paddles. Update the ball's direction accordingly and keep score.

C# code

using UnityEngine;  
using UnityEngine.UI;  
  
public class GameManager : MonoBehaviour  
{  
 public Text player1ScoreText;  
 public Text player2ScoreText;  
  
 private int player1Score = 0;  
 private int player2Score = 0;  
  
 void OnCollisionEnter2D(Collision2D collision)  
 {  
 if (collision.gameObject.CompareTag("Paddle"))  
 {  
 // Reflect the ball's velocity.  
 Vector2 reflection = Vector2.Reflect(GetComponent<Rigidbody2D>().velocity, collision.contacts[0].normal);  
 GetComponent<Rigidbody2D>().velocity = reflection;  
 }  
 else if (collision.gameObject.CompareTag("Goal"))  
 {  
 // Update scores and reset the ball's position.  
 if (collision.gameObject.name == "GoalLeft")  
 {  
 player2Score++;  
 ResetBall();  
 }  
 else if (collision.gameObject.name == "GoalRight")  
 {  
 player1Score++;  
 ResetBall();  
 }  
  
 // Update the UI.  
 player1ScoreText.text = "Player 1: " + player1Score;  
 player2ScoreText.text = "Player 2: " + player2Score;  
 }  
 }  
  
 void ResetBall()  
 {  
 // Reset the ball's position and give it an initial force.  
 GetComponent<Rigidbody2D>().position = Vector2.zero;  
 GetComponent<Rigidbody2D>().velocity = Vector2.zero;  
 GetComponent<BallController>().Start();  
 }  
}

**Step 8:** Set up the User Interface (UI)

Create UI Text elements to display the scores for player 1 and player 2.

Create an empty GameObject called "Canvas" and make the UI Text elements children of the Canvas.

**Step 9:** Create Goal Triggers

Create two empty GameObjects named "GoalLeft" and "GoalRight" to act as triggers for scoring goals. Place them at the left and right ends of the play area.

Attach Box Collider 2D components to these goal trigger GameObjects and set them as triggers.

**Step 10:** Build and Play the Game

Save your scenes and build your game.

**Code Simulation:**

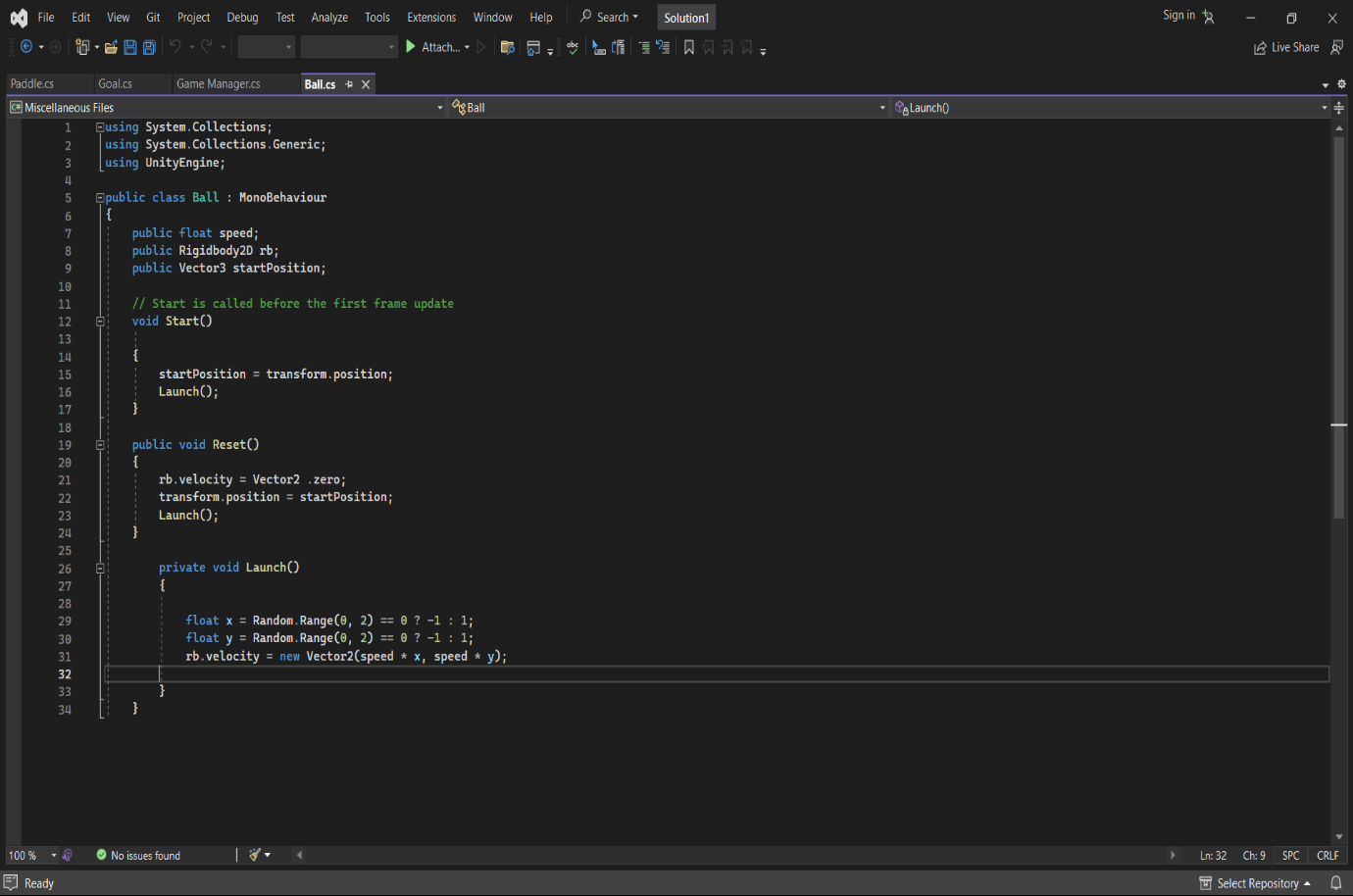


Figure 1 Code for Ball

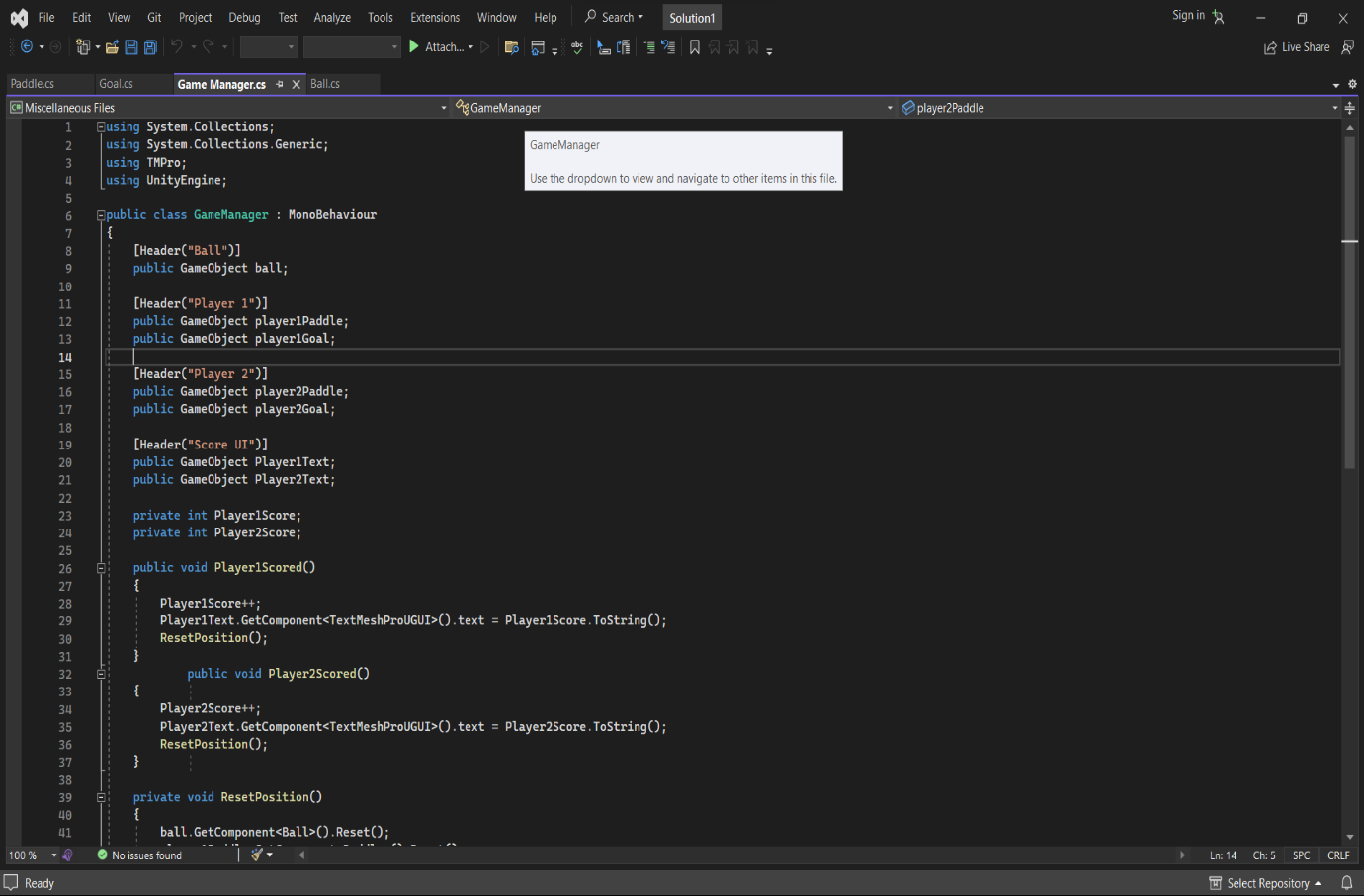


Figure 2 Code for Game Manager

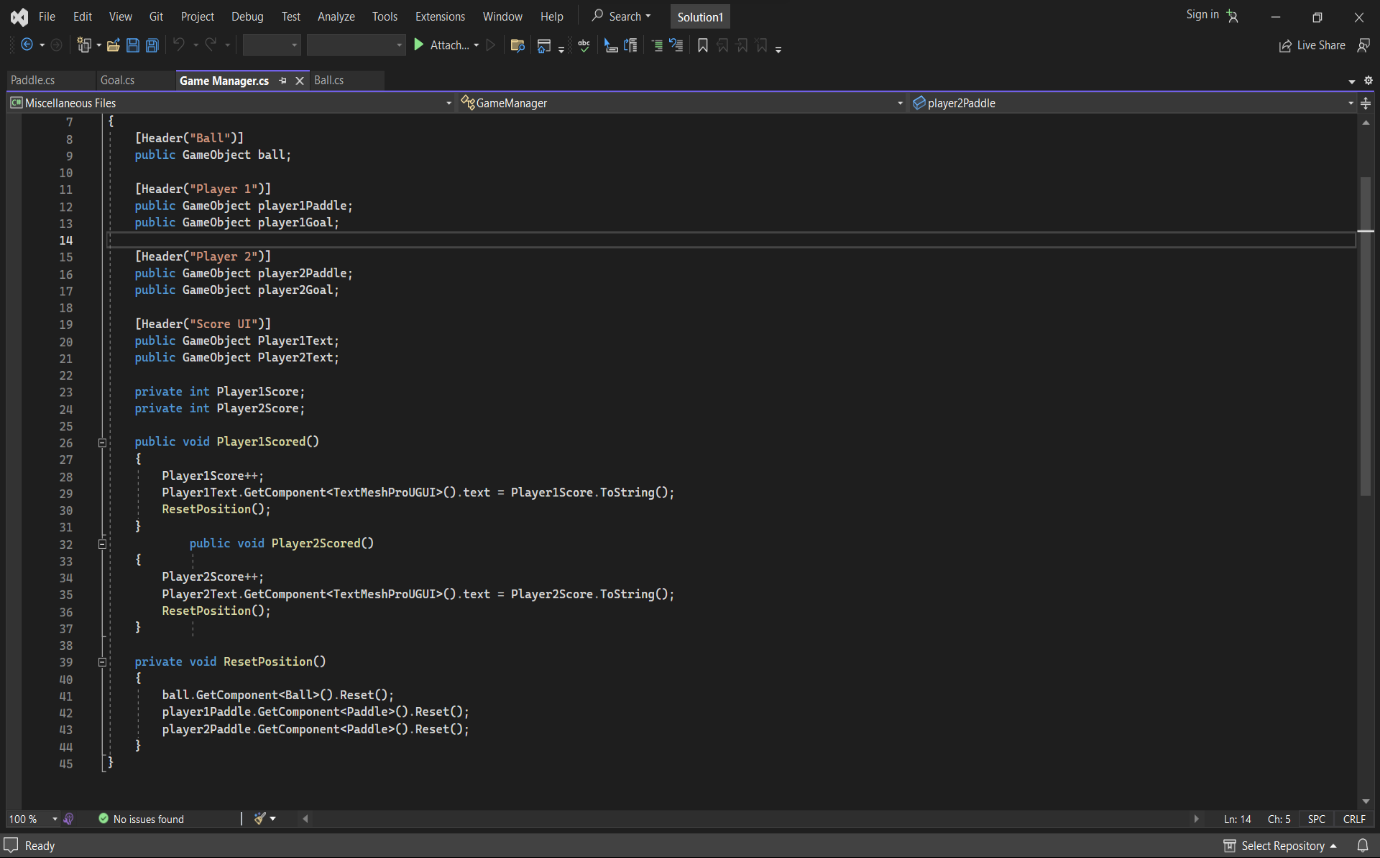


Figure 3 Code for Game Manager

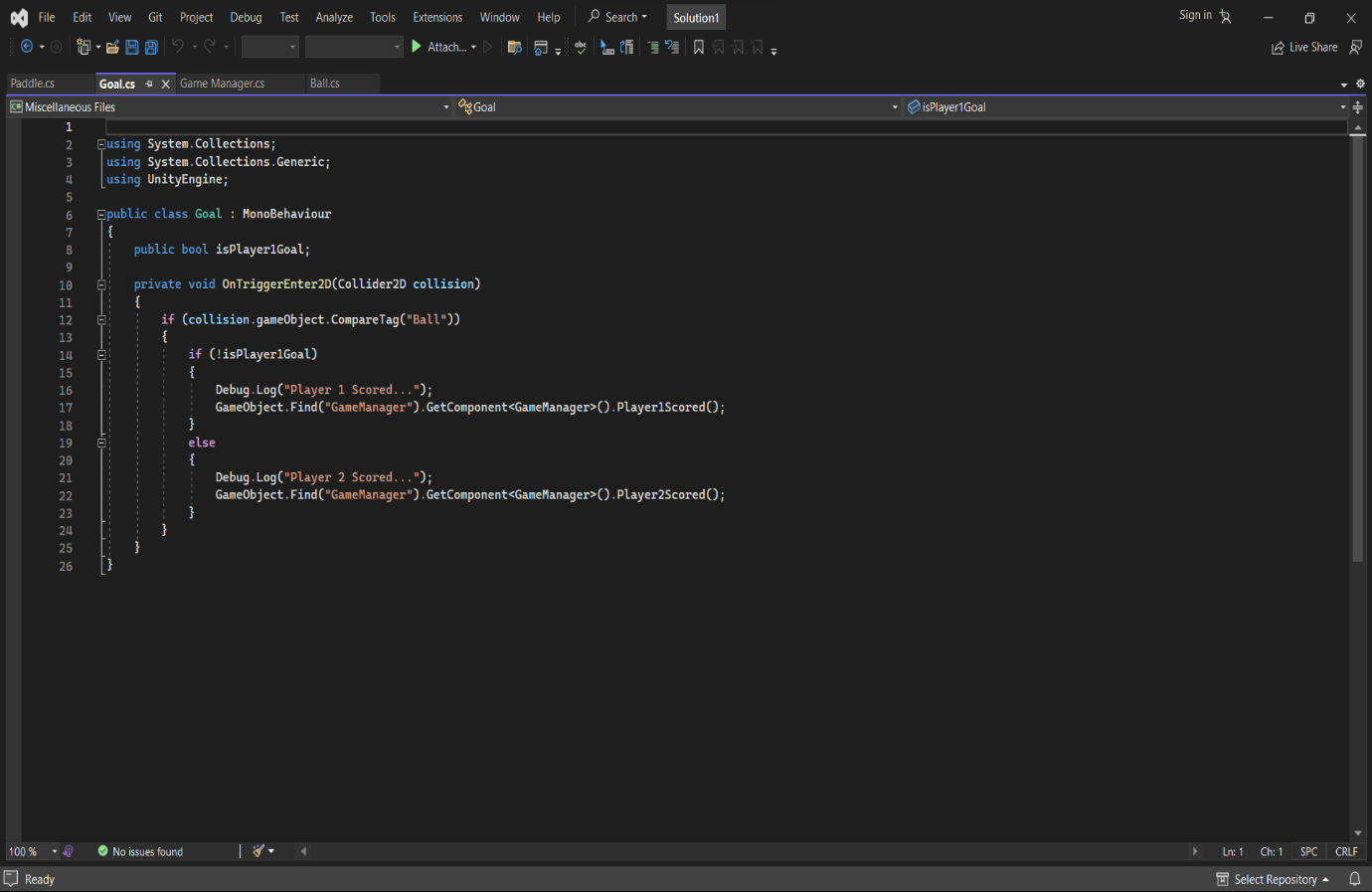


Figure 4 Code for Goal

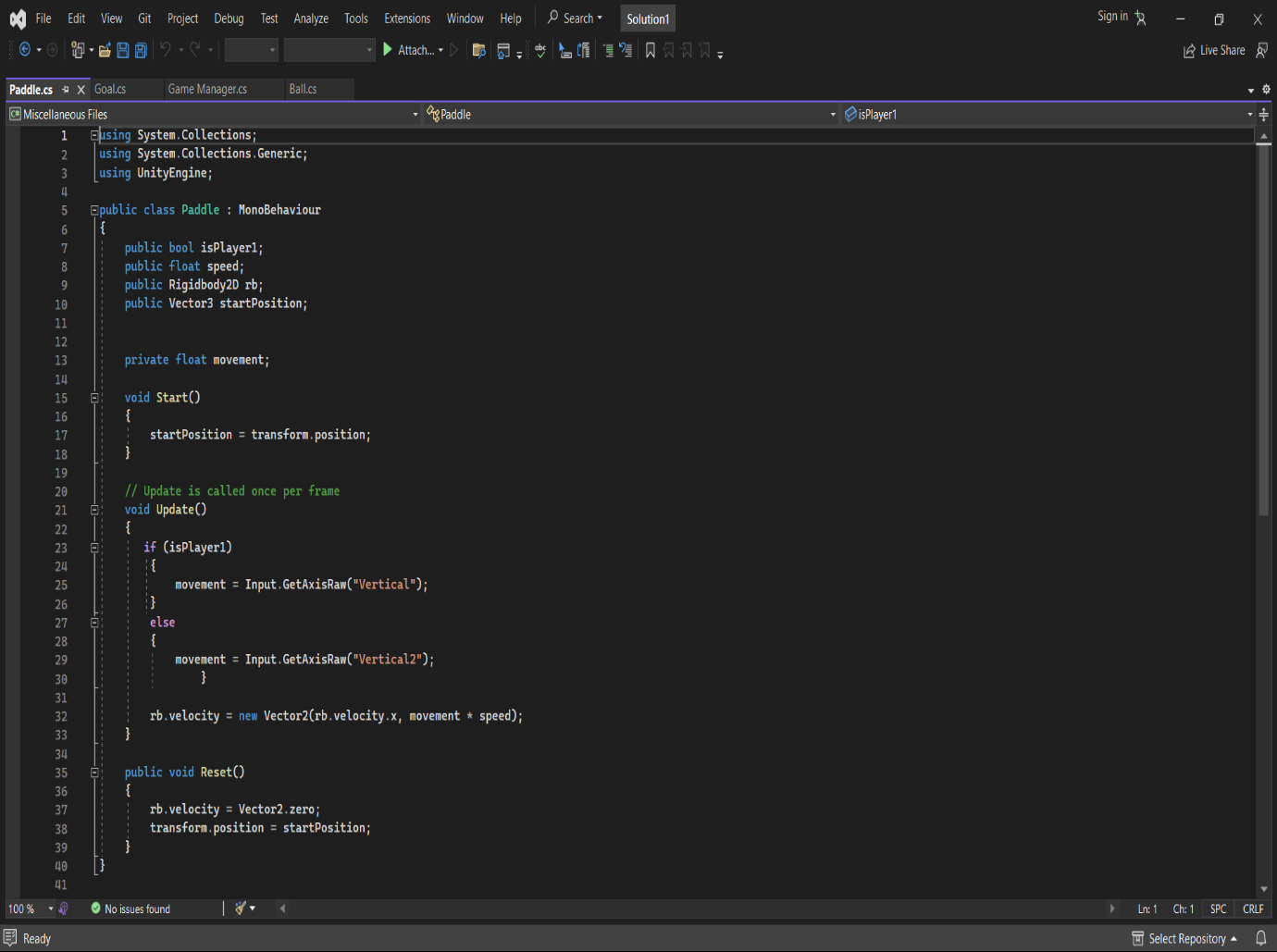
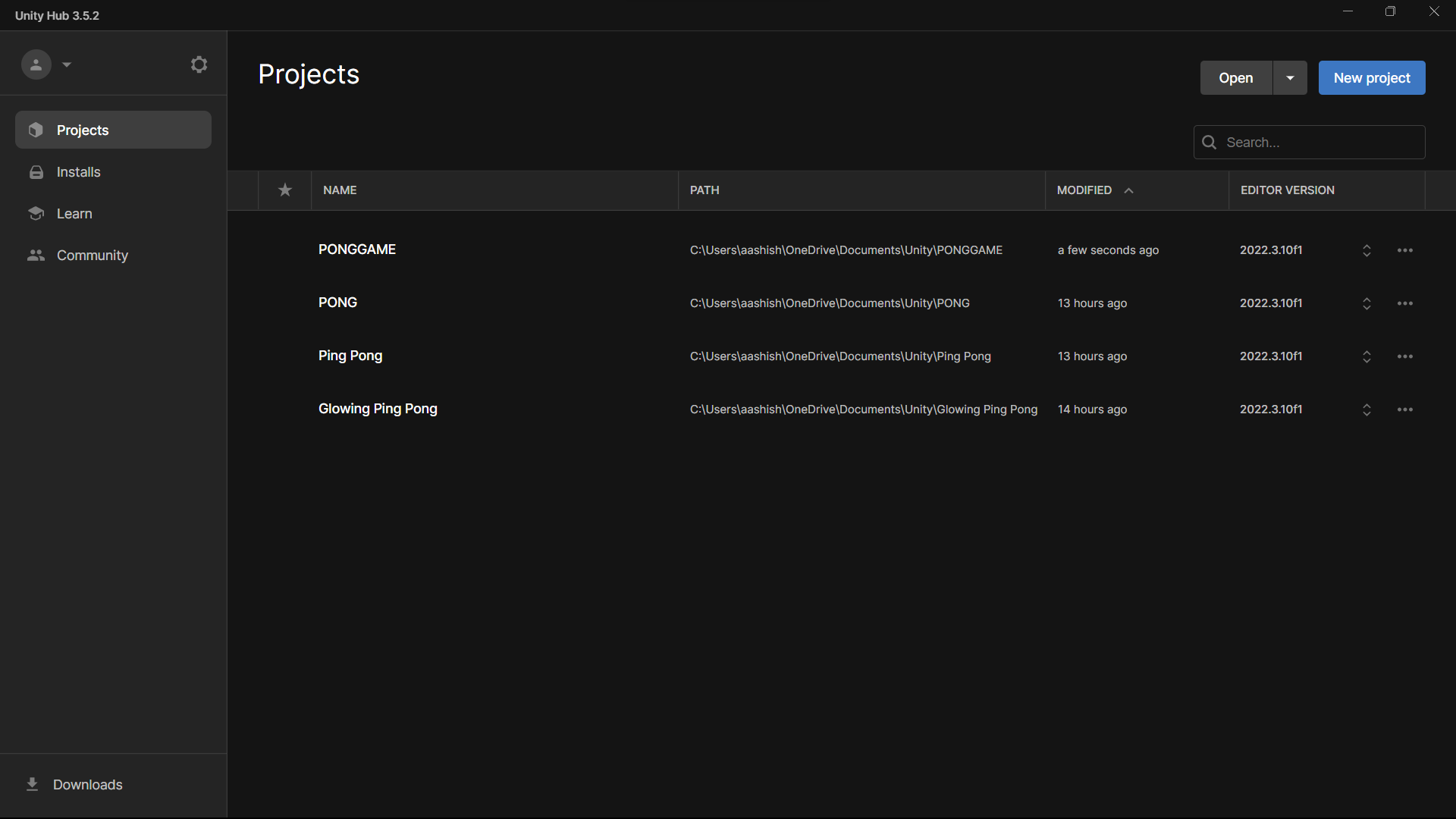


Figure 5 Code for Paddles



Bottom of Form

Figure 6 This is the place where project saves.

**Output:**

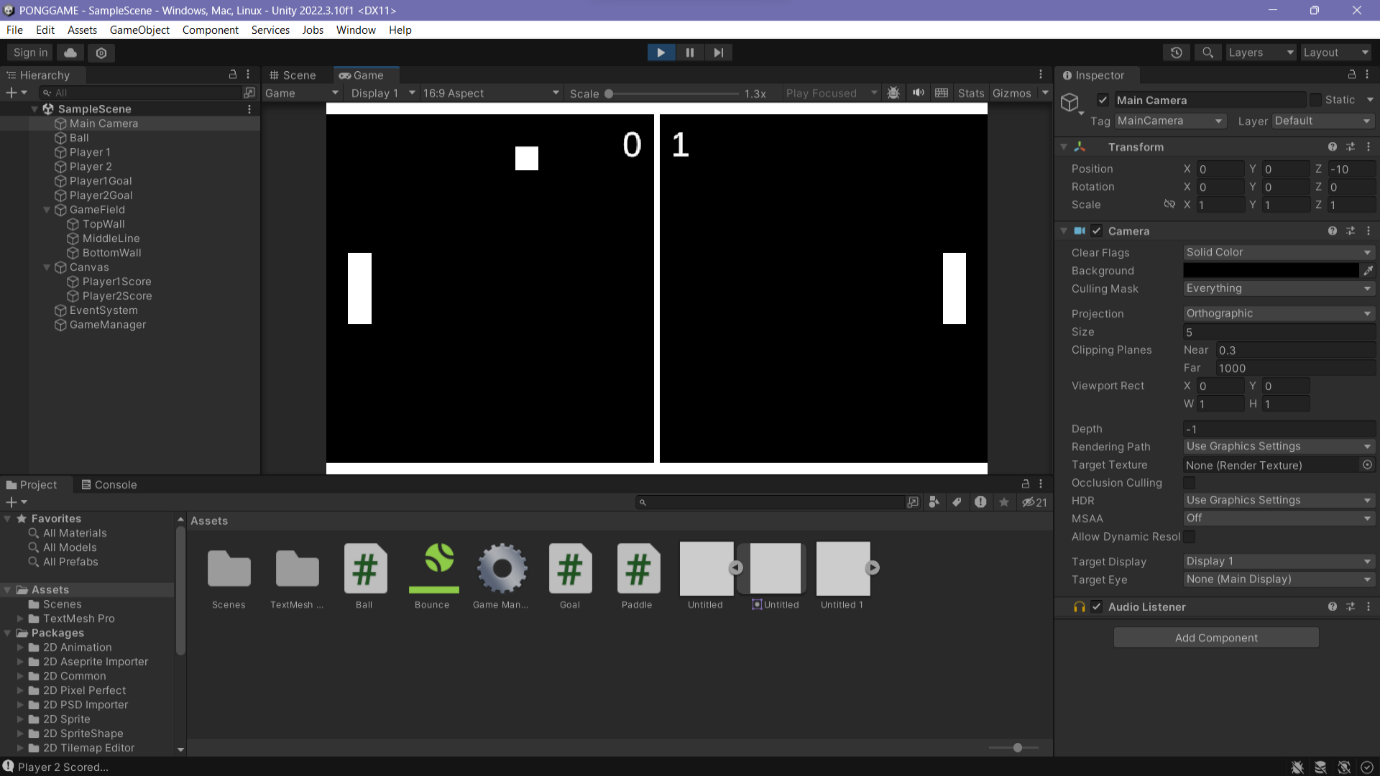


Figure 7 Output of the program

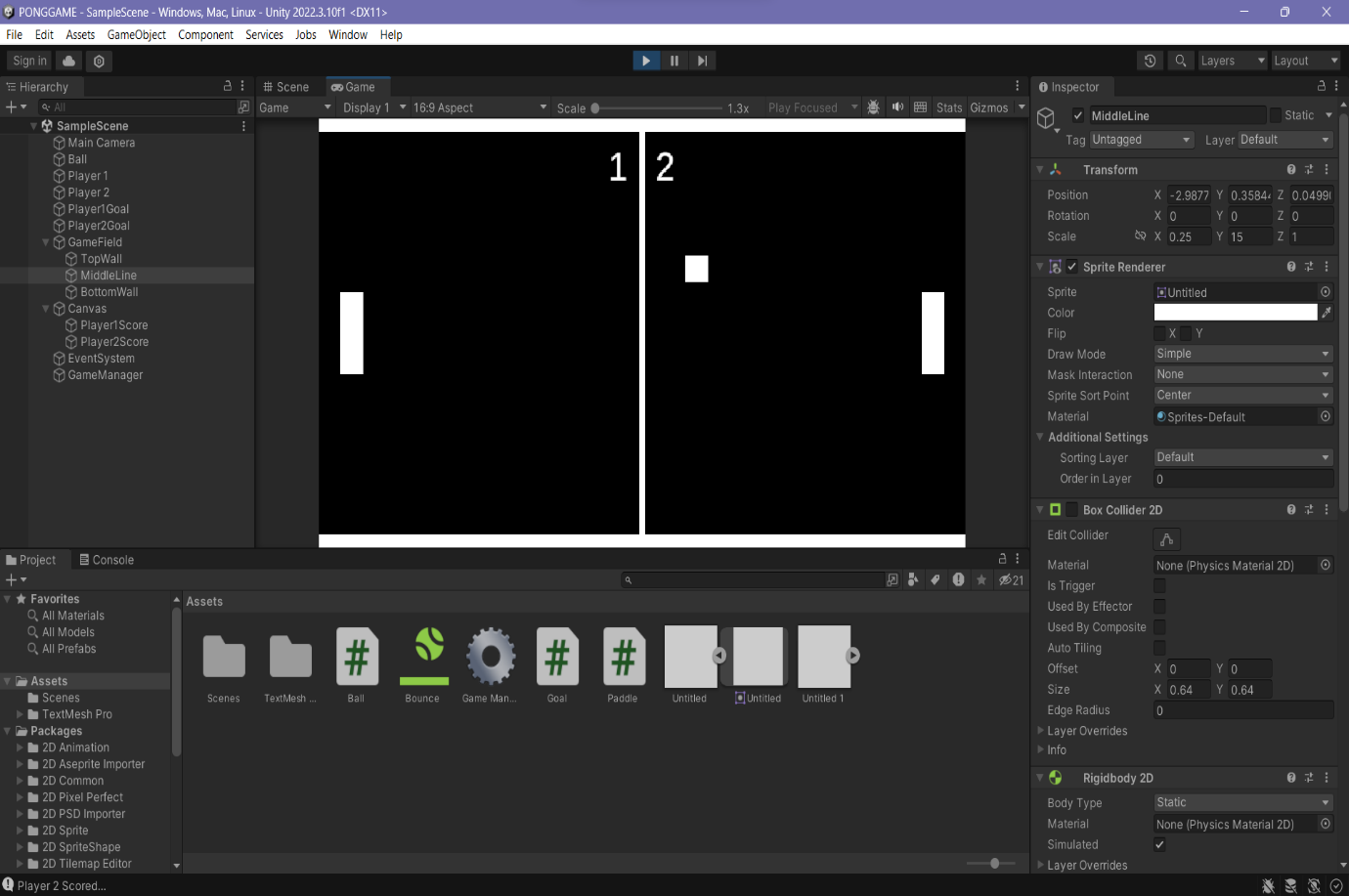


Figure 8 Output of the program

**5. Course Outcome & Conclusion**

**Outcome:**

CO2: Identify various geometric modelling techniques.

The experiment involved creating a simple 2D Pong game in Unity. Here are the key outcomes and conclusions:

* Game Development in Unity: This experiment demonstrated the process of developing a basic 2D game in Unity, a popular game development engine.
* Scene Setup: We learned how to set up the Unity project, create game objects, and design the game scene.
* Physics and Collisions: Physics interactions were implemented by adding Rigidbody2D components to the ball and using collision detection to handle interactions between the ball and paddles.
* Player Controls: Player control was implemented by writing scripts that allowed players to move the paddles using keyboard inputs (e.g., arrow keys and "W" and "S" keys.
* Scoring System: We implemented a scoring system that updated scores when the ball passed through the goal triggers.
* Room for Improvement: While the game is functional, there is room for improvement and expansion.
* Learning Opportunities: This experiment provided hands-on experience with game development concepts, including physics, collision detection, and user input handling.

**Conclusion:**

The experiment successfully demonstrated the creation of a functional Pong game in Unity. It served as a valuable introduction to Unity's game development workflow, including scene setup, physics, scripting, and user interface integration.

Overall, the experiment demonstrated the process of creating a simple 2D game in Unity, showcasing the fundamental aspects of game development within the Unity environment. Further development and refinement of the game could lead to a more complete and enjoyable gaming experience.