TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

A Project Report

On

Unity-Based Ball Rolling Game

Submitted By:

Samir Pandey (HCE081BEI036)

Submitted To:

Department of Electronics and Computer Engineering

Himalaya College of Engineering

Chyasal, Lalitpur

Feb 22, 2026

ACKNOWLEDGEMENT

I am thankful to Himalayan College of Engineering and my friends for their help and support during the completion of this project.

I would like to thank my teachers and the Department of Electronics and Computer Engineering for their guidance. I also thank my friends and classmates for their suggestions and motivation during the 3D modeling and documentation work.

Thank you.

Samir Pandey [HCE081BEI036]

ABSTRACT

This project presents the development of a Unity-based ball rolling game. The process includes scene and level design in the Unity editor, physics setup (Rigidbody and colliders) for the ball and environment, C# scripting for player input and game logic, and camera follow. The goal is to build a playable 3D game that demonstrates real-time physics and game development workflow in Unity. The report describes the methodology, tools, and results of the ball rolling game implementation.

Table of Contents

List of Figures

List of Abbreviations

HCOE	Himalaya College of Engineering

List of Tables

1. INTRODUCTION

Computer Graphics (CG) and game development enable the creation of interactive 3D experiences. Game engines such as Unity provide tools for building physics-based games, including ball rolling and platformer mechanics. This project presents a mini project titled **Unity-Based Ball Rolling Game**, which demonstrates the design and implementation of a 3D ball rolling game using the Unity engine, including physics, controls, and level design.

1.1 Background Introduction

A typical game development pipeline involves scene design, physics setup, scripting, and user input handling. In Unity, game objects are composed of transforms, colliders, rigidbodies, and scripts. Physics engines simulate gravity, collision, and rolling behavior, enabling realistic ball movement. Modern game engines like Unity support C# scripting and a visual editor for rapid prototyping [1, 2].

1.2 Motivation

Many beginners wish to build playable games but lack exposure to game engines and physics-based mechanics. This project is motivated by the need to understand the workflow of creating a simple 3D game in Unity: setting up a ball, applying physics, designing levels, and implementing player controls. A ball rolling game serves as an accessible entry point to game development and computer graphics in real time.

1.3 Objectives

The main objectives of this project are:

- Design and implement a 3D ball rolling game using the Unity engine.
- Configure physics (Rigidbody, colliders) for realistic ball movement and rolling.
- Implement player controls (keyboard or input system) to move the ball.
- Create a simple level with obstacles and goals using the Unity editor.

1.4 Scope

This project focuses on:

- Unity scene setup and hierarchy
- Physics components (Rigidbody, Sphere Collider) and material setup

- C# scripting for input and game logic
- Level design and Unity interface usage

It does not cover advanced topics such as multiplayer, complex AI, or custom shaders, which are suggested as future enhancements.

2. LITERATURE REVIEW

2.1 Game Engines and Unity

Game engines provide an integrated environment for building interactive applications. Unity is widely used for 2D and 3D games, offering a scene graph, component-based architecture, and built-in physics (PhysX). Ball rolling and physics-based games rely on rigid body dynamics and collision detection [3].

2.2 Physics in Games

Physics simulation in games uses rigid bodies, colliders, and constraints. Parameters such as mass, drag, and angular drag affect rolling and movement. Friction and bounciness (physics materials) determine how the ball interacts with surfaces [1].

2.3 Input and Controls

Player input is typically handled via the Input Manager or the new Input System in Unity. Keyboard, mouse, or gamepad input can be mapped to forces or torque applied to the ball, enabling responsive and intuitive controls [2].

2.4 Related Tools and Frameworks

Unity, Unreal Engine, and Godot are widely used for game development. Unity's C# scripting, visual editor, and asset store make it suitable for academic projects and rapid prototyping of ball rolling and platformer games [2].

3. METHODOLOGY

3.1 System Overview

The project workflow is divided into the following modules:

- **Scene Setup Module:** Creation of the game level, platforms, and obstacles in the Unity editor.
- **Physics Module:** Rigidbody and collider configuration for the ball and environment.
- **Control Module:** C# scripts for player input and ball movement.
- **Integration Module:** Camera follow, goals, and game state (e.g., win condition).

3.2 Scene and Level Design

The game level was built in the Unity editor using primitive and custom 3D objects. Platforms, ramps, and obstacles were placed to create a ball rolling course. Transforms, colliders, and (where needed) rigid bodies were assigned to ensure correct physics behavior.

3.3 Ball and Physics Setup

A sphere GameObject was used as the ball. A Rigidbody component was added for gravity and dynamics, and a Sphere Collider for collision detection. Physics material was configured for appropriate friction and bounciness. Mass and drag were tuned for playable feel.

3.4 Input and Scripting

Player input was read using Unity's Input API (or Input System). Forces or torque were applied to the ball's Rigidbody based on input to produce rolling motion. Scripts were attached to the ball and other objects as needed for game logic.

3.5 Camera and Polish

A follow camera was set up to track the ball. Lighting and simple materials were applied to the scene. Win condition (e.g., reaching a goal zone) was implemented via triggers and scripts.

3.6 Screenshots / Figures

Figure 3.1 shows the Unity editor interface with the ball rolling game project. Figure 3.2 shows an additional view of the Unity setup or gameplay.

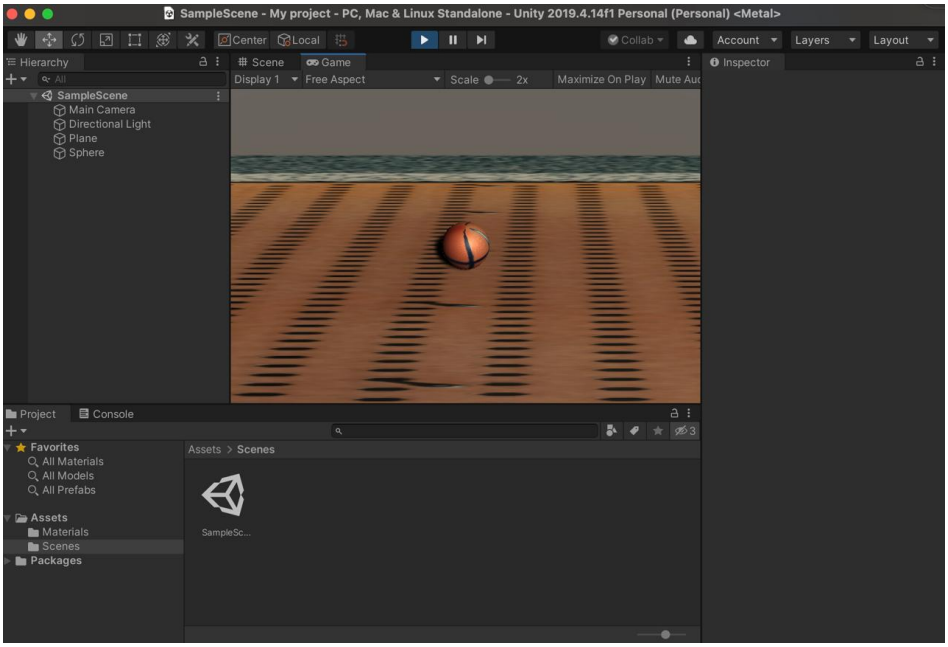

Figure 3.1: Unity editor interface for the ball rolling game project.

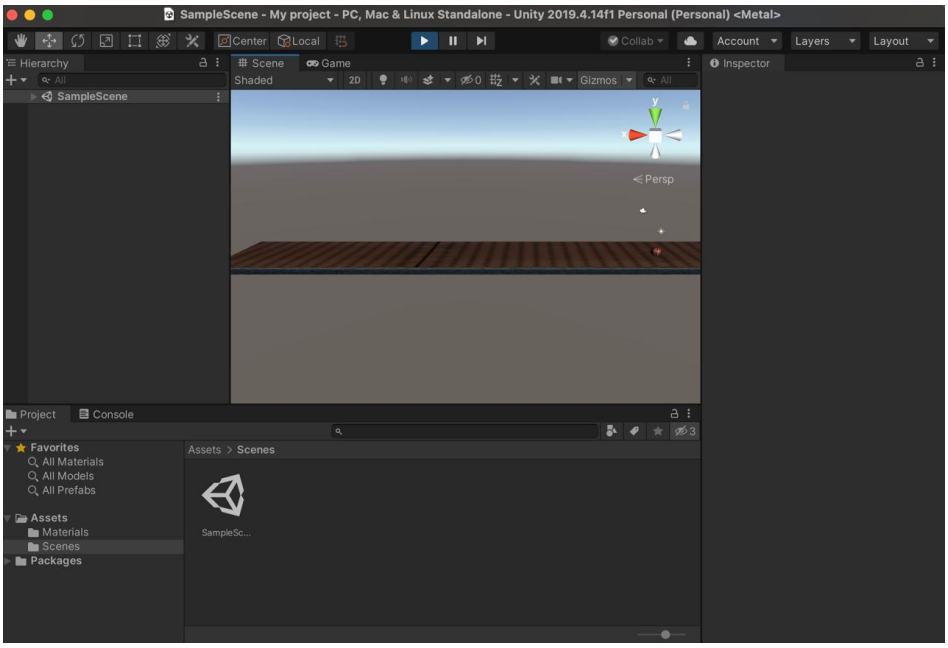

Figure 3.2: Unity interface view showing scene or gameplay setup.

4. RESULT AND ANALYSIS

4.1 Correctness of Game Behavior

The ball rolling game behaves as intended: the ball responds to player input, rolls under physics (gravity and collision), and interacts correctly with the level geometry. Collision detection and rigid body dynamics produce realistic rolling and bouncing. The camera follows the ball appropriately during play.

4.2 Physics and Control Behavior

The game exhibits expected physics and control behavior:

- **Movement:** The ball moves and rolls in response to input without slipping or erratic behavior when tuned correctly.
- **Collision:** The ball collides with platforms and obstacles and responds with appropriate reaction forces.
- **Stability:** The simulation remains stable under normal play; no unintended tunneling or jitter was observed with default settings.

Rigidbody and collider settings are consistent with Unity's physics engine and are suitable for a simple ball rolling game.

4.3 Performance Discussion

Game performance depends on scene complexity (number of colliders and rigid bodies), script logic, and rendering load. The current project uses a small number of objects and simple scripts, so it runs smoothly at real-time frame rates on typical hardware. Further optimization (e.g., object pooling, LOD) can be applied if the level is expanded.

4.4 Limitations

- Level design is limited to a single or few levels; no full game progression system.
- Advanced features such as power-ups, multiple balls, or multiplayer are not implemented.
- Mobile or touch input was not specifically tested; the project assumes keyboard/gamepad.
- Only one build target (e.g., PC) was verified; other platforms may need testing.

5. CONCLUSION AND FUTURE ENHANCEMENT

5.1 Conclusion

This project demonstrates the design and implementation of a Unity-based ball rolling game. The project covers scene setup, physics configuration, player controls, and level design within the Unity editor. The result is a playable 3D game that illustrates core concepts of game development and real-time physics, validating the use of Unity for educational and prototype projects.

5.2 Future Enhancements

- **Additional Levels:** Design multiple levels with increasing difficulty and varied obstacles.
- **Power-ups and Collectibles:** Add collectible objects, timers, or score systems.
- **Mobile Support:** Implement touch or accelerometer controls for mobile builds.
- **Visual and Audio Polish:** Improve materials, effects, and sound for a more polished experience.

A. APPENDICES

A.1 Key Software Tools Used

- **Unity:** Game engine for scene design, physics, scripting, and building the ball rolling game.
- **Visual Studio / VS Code:** For C# scripting and Unity integration.
- **Unity Input System:** For handling player input (keyboard/gamepad).

Bibliography

[1] J. D. Foley, A. van Dam, S. K. Feiner, and J. F. Hughes, *Computer Graphics: Principles and Practice*. Addison-Wesley, 3rd ed., 2013.

[2] Blender Foundation, "Blender documentation." `https://docs.blender.org`, 2024. Accessed: 2025.

[3] E. Angel and D. Shreiner, *Interactive Computer Graphics: A Top-Down Approach with WebGL*. Pearson, 7th ed., 2012.