Skateboarding Simulator Prototype - Assessment Document

Introduction

This document outlines the development process for the Skateboarding Simulator Prototype, created as part of an interview assessment. The project showcases core gameplay mechanics implemented in Unreal Engine 5.3 using C++ to demonstrate programming skills, game design understanding, and system implementation.

System Overview

The prototype features:

- **Movement Mechanics**: Player-controlled skateboarding, with functionalities for speeding up, slowing down, and jumping.
- **Obstacle System**: Designed to reward successful jumps while penalizing failures. The logic leverages collision detection and point adjustments.
- **Points System**: Tracks and updates player score dynamically via the collision manager.
- **UI (HUD)**: Displays the player's current score. While the HUD logic was implemented in C++ using a collision manager, its display relies on a Widget Blueprint due to time constraints.
- **Level Design**: A compact skate park with obstacles and jump challenges was created using assets from the **City Park Environment Collection**. Characters and animations were sourced from Mixamo.

Thought Process

I prioritized essential gameplay mechanics (movement and jumping) and ensured modular, reusable code for future scalability. Collision-based logic governs the point system, rewarding successful interactions. The HUD, though partially implemented in Blueprint, highlights the game's dynamic feedback loop.

Self-Assessment

Strengths:

- Clean, modular C++ implementation for mechanics and scoring.
- Effective asset integration and level design within the given time.
- Incremental commits for transparency.

Weaknesses:

- Time constraints led to HUD reliance on Blueprint.
- Limited polish on non-essential features, such as obstacle variety.

Time Breakdown

- **Movement, Jumping, and Speed Controls**: 8 hours
- **Obstacle System and Points Logic**: 5 hours
- **UI and HUD Integration**: 4 hours
- **Level Design and Asset Integration**: 3 hours
- **Playtesting and Debugging**: 2 hours
  Total Time Invested: 22 hours
