Skateboarding Simulator - Development Documentation

System Overview

The **Skateboarding Simulator** is built using **Unreal Engine 5.3** in **C++**, ensuring performance and extensibility. The project began with **Epic Games' Third Person Template** as a base, providing a solid foundation for character movement and camera control.

Key features of the system include:

- **Player-controlled Skateboarding Mechanics**
  - Acceleration (using **E** on the keyboard), deceleration (using **Q**), and jumping (using **Spacebar**)
  - Skateboard rotation adjusts dynamically to ground slope (this is a basic implementation)
  - Uses Mixamo animation for jumping and movement
- **Game Mode & Score System**
  - Implements `IScoreInterface` to track points and handle game-over scenarios
  - `AObstacle` detects successful jumps and updates the score
- **User Interface (HUD)**
  - Displays score updates via `USkaterHUDWidget`
  - Randomized **positive reinforcement** and **reset messages** for player engagement
- **Collision & Gameplay Interaction**
  - `AObstacle` handles both score updates and game-over detection using trigger boxes
  - The system uses **two collision zones**: space and obstacle. If the player successfully bypasses the obstacle within the specified range, they receive **10 points** and a **notification bump**.
  - `ISkaterInterface` ensures only valid skaters affect scoring

Thought Process

The development process started by structuring the essential **C++ classes** to define interactions and gameplay mechanics.

1. **Character Movement:** Expanded upon Unreal's third-person character, adding skateboard-specific functionality.
2. **Game Mode & Scoring:** Implemented an interface-driven scoring system to ensure modularity.
3. **HUD:** Integrated UI elements to display real-time feedback and encourage the player.
4. **Physics & Collisions:** Used trigger volumes for scoring and failure conditions.
5. **Testing & Polishing:** Conducted iterative playtesting to refine mechanics.

Interface Usage

- `ASkaterHUD : public AHUD, public ISkaterHUDInterface` - **Used interfaces to decouple the system, ensuring flexibility**
- `class ASkaterCharacter : public ACharacter, public ISkaterInterface, public ISkaterCharacterInterface` - **`ISkaterInterface`** defines whether this actor can win and interact with game rules, while **`ISkaterCharacterInterface`** handles animation communication without direct references**

Self-Assessment

The implementation met all **task requirements** while keeping the codebase clean and modular. Using **interfaces** for interactions improved flexibility, and the **HUD messaging system** enhanced the player experience. Given more time, I would refine skateboard physics and add trick animations. Additionally, I would retarget the skeleton fetched from Mixamo and use control leg adjustments to ensure the back leg does not go below the floor. I would also expand the system by making movement physics-based instead of relying purely on mathematical calculations.

Time Breakdown

The total time spent on the project was **14 hours**. However, I do not recall the exact breakdown per task.

Total Time Invested: 14 hours