Skateboarding BGS Task

Author: Bastián Castro Salazar

Initialization:

The idle animations of different stances were downloaded for possible uses in the mobility of the character. The skeleton that came with the animation was used to import the other acceleration and jump animations. The inputs created are Accelerate with the W key, Brake with the S key, Handling with the A and D keys, Jump with the space key and Crouch with LeftControl.
The construction of the character started with the skeletal mesh to be used. And a folder architecture for the organization.

Animation Blueprint:

Having the character base, the Animation Blueprint is created with a state machine, idle, falling and crouch states. The programming handles mainly the falling state with the Character Movement.

Character Movement:

To simulate the skateboard movement, the friction with the ground is lowered and then a function is created to accelerate, which generates an impulse and coordinates with the animation to propel itself. To turn left or right the character is rotated, to solve the lateral sliding, a function is created to adjust the forward speed of the skateboard, connected to an Event Tick, along with the rotation of the camera and the adjustment of the skateboard to the inclination of the terrain.

Crouching reduces the size of the character's collision capsule and adjusts its position so that the mesh is not buried, a Blueprint Interface was created to do the crouching animation.

The brake generates a backward impulse of the skate, slowly slowing down the character. Finally the jump was used the Character Movement component to jump at a certain speed and the jump animation was used and coordinated to make it look better.

Score System:

The scoring system was created in a component called ScoreSystem and added to the character. A function was made to modify the Score

variable, also a multiplier that increases when jumping through the control points. A Blueprint Interface was created to send the signal from these control points, increasing the Score and the Multiplier.
The object that is used to obtain points has a collision that detects the player, with an Overlap Event checks if it has the ScoreSystem component to increase an amount of score and if it increases the multiplier.

User interface:

The user interface has the score and multiplier with a bar that determines how long the multiplier lasts.
A Blueprint Interface was used to change the values from the scoring component and be able to display a bar that decreases over time based on how much of the multiplier is left.

Map setup:

A landscape was created and props such as stairs, benches and jumping rails were added. Also a fountain with a 1000 points object was added on the top of the fountain.