Skating Game by Dino Zulfic

Introduction:

After recieveing the developer task from the studio, the first thing that started was researching. Since Skateboarding games have a particular feeling to them, looking up some gameplay videos to get a general feeling for it.

The Game:

Creating the Game system was a very pleasing task. Initally, I was thinking of creating a WASD controller with a basic camera attached to it.
However, after viewing some gameplay (particulary the Tony Hawk games) it seemed that there was a lot of comunication between the camera angle and the forward vector of the player character. To avoid fine tuning the math for long, I decided to go with a more "Funny" approach. Inspired by playing games with an unfitting controller as a child, particularly playing TUROK: DINOSAUR HUNTER with a flight stick (that was my birthday present lol), I have decided to structure the system diferently.

Movement: Acceleration and Breaking

Movement is being handled inside of the custom Player Character Class in C++.
What this entails is that, while the W and S keys both are inputs, the A and D keys are not. W as an input gives the player acceleration, while S reduces it if the Velocity exceeds 0 (in Size) To achieve the desired efect, I have created a custom CharacterMovementComponent that overrides the PhysWalking function together with the GetMaxSpeed function.
By playing around with the movement component settings inside of the editor, and adjusting friction I have created a "sliding" type of movement.
However, when on a skateboard the speed doesn't ramp up like in a car, which is why the TryIncrementVelocity function adds on velocity while holding the W input.
This function also fires of a Multicast Delegate which is in turn used to play an AnimMontage for the player pushing on the skateboard, effectively creating the feeling of the speed increasing with each push.
While this system is functional, there is still a lot that there can be done.For instance, firing off the AnimMontage while on max speed does not happen.
Moving the camera however will decrease the velocity (speed up it's decrease) the sharper you turn.

The Character:

The character handles movement like stated, and also handles the input. I prefer putting the input inside of the character class in case there should be diferent controllable characters.

This however is purely preference, as it could have been handled in the Controller as well.
The Character also handles Ragdolling. The Ragdoll is very rudimental basically the mesh falls, the capsule does not. Movement is not prohibited during this, because it was quite funny while testing, so I decided to leave it in. Can be corrected ofcourse.
The character also handles jumping, with a slight cooldown, as to not be able to jump consecutively. Mixamo gave me some problems with animation importing, so a jump animation is missing, however that would easily be handled in the anim blueprint, since it consits of just the default state and an animation.

Miscelanious details:

-Using a delegate to trigger the animation inside of a blueprint by using an BlueprintImplementableEvent is only one way to do this. If structured diferently the AnimMontage could have been the one triggering the increase.

-StepIncreasElapsedTime uses the BIG_NUMBER macro in its declaration(initialized in header), basically used for guaranteeing that nothing will be bigger than that value.

-TWeakObjectPtr<USkateMovementComponent> SkateMovementRef is a fun way to handle a component reference on the owner.

-FObjectInitializer is used in the constructor of the Character to enable overriding a component that is on the Character class by default, in this case the movement component.

Blueprints:

My usual workflow entails that C++ is used for calculation and big systems most of the time, and blueprints for content creation and prototyping. This is not always the case, since the widgets used in the project are completely made in BP.

The PlayerState blueprint is what keeps track of the player score, which is being shown on the screen. I used quite a lot of casting here since I wanted to get it done quickly as a prototype, which is not advised.

The Obstacle blueprint is basically just an actor that has a mesh and a box component.
The Box Component has an Begin Overlap event which is used to check if the player has cleared the obstacle. Default points is 100.

LevelBlueprint is basically used only for widgets like main menu.

Level:

The level itself is a level from an asset pack. While I do enjoy creating levels in my spare time from time to time, I consider my artistical skills to not be on an satisfactory level.
While I do excell at designing gameplay systems, the art part of game development is not one of my strengths.

What I would Add:

Music, SoundFX, More animations, and the usual, however what I would have really liked to do is to use Splines for both Grinding and tracing around a corner pipe in the air. Since a spline would already be in place, acnhoring the player to it seems like it would create that old school Tony Hawk feeling, however I could not find maps for that, and frankly, did not have the time to implement everything.

Assessment of personal performance:

It was a fun project, I believe I created a fun prototype in the time I had to work on the project. While I do prefer working on things more detailed, a complete project in detail takes time. Considering the approches used and the general goofy feeling of the game, I would say that I am satisfied with the project, would like to keep working on it however, to improve its quality.