**Crouch**

1/22/2024

Implementing crouch is not hard, as it is inherently a part of the character movement component. I’m attempting to adjust the “height” of the crouch, allowing the player to hide underneath surfaces that they could reasonably fit under using a trigger volume around the surface and an overlap check to see if the player is able to crouch lower in the area. If both the player is crouch and able to crouch lower, then the player successfully crouches lower. If either of those Booleans are false, nothing happens.

The latter part of that interaction is a success, however if the intended crouch functionality is successfully fully executed, the character movement suddenly seems to lose all friction in the environment while crouched. Un-crouching fixes this effect until the implementation is activated again. In the meantime, whenever the player attempts to go underneath an object, they continue sliding in the direction of their movement, and lose full mobility control. I don’t even know where to begin debugging this, but I did record a [short video](https://youtu.be/liGoGdVIVcc) of the issue and posted it into my programming text channel in my server.

1/23/2024

Brian recommended adjusting the radius of the capsule, since a capsule can only shrink to be as small as a sphere. Outside of that, he also thinks that the collision is just pushing the player out because the player shouldn’t fit. Maybe take away the geometry in the scene just so I can test the trigger volume?

1/24/2024

Brian’s suggestions worked. The crouch height now automatically adjusts depending on whether the player is trying to hide underneath something or not. While hiding underneath an object, the player retains full control over their movement, and the collision does not attempt to force the player out from hiding. This was the cause of the “ice mechanics.”

Now, there are still a couple of issues left. When the player un-crouches while they are hiding underneath something, upon leaving the hiding area, their height is set to the regular crouch height, even though they are no longer crouching. This issue resolves once the player crouches + un-crouches again (no need to activate the lowered crouch). The other issue is that when the player is approaching the trigger to lower the crouch (while crouched) and stops at the edge of the trigger volume, they’re camera will jump between the lower crouch height and regular crouch height.

I have a feeling that the latter issue will be solveable by adjusting the timing on the retriggerable delay nodes on the On Component Begin and On Component End Overlap events. This adjustment could either be a longer time, or replacing the retriggerable delay node with a standard delay node. *The delay needs to be there regardless or the program will crash due to a call stack infinite loop.* I would also like to add interpolation between the values to make the transition smoother, which could also fix the issue.

Interestingly enough, this “glitch” is present in Dishonored 1, though not in Dishonored 2.

The former issue could be resolved through adjusting my Boolean branch statements, possibly. To ensure that the functionality is called at the appropriate time, I have to bools, one bIsCrouched? Bool casted from the FP\_Character blueprint and another bIsOverlapped? Bool inherent in the BP\_LowerCrouch blueprint. If both are true, the player crouches lower. I need to add an exception to allow for the bIsCrouched? Bool to be false while still allowing the player to crouch lower while they are already hiding underneath an object without the capsule collision causing the environment to turn into an ice rink again.

1/29/2024

The issue where the player’s full height was reduced when un-crouching while hiding until they crouched/un-crouched again has been fixed. I was correct in my guess that I needed to adjust my *if* statements. Now, there is an additional check after whether crouchHit? Is true. If it is, and crouched? Is false, the player class inherent function “Un-Crouch” is called. If crouchHit? Is false and crouched? Is true, the player’s capsule height and radius are adjusted to their regular non-hiding crouch proportions.

The “bouncing” crouch issue is still present. I’m having Tyler help out with the issue as I type this. I believe it’s a problem related to the resizing of the player’s collision capsule. I need to resize the capsule height and radius to let the player fit underneath objects, but if it’s too large, the player slides around (see the 1/22/2024 entry). If the player’s capsule shrinks, the collision overlap is affected. The bounce is caused by the capsule shrinking out of the overlap volume, reverting back to its original proportions, and re-triggering the overlap volume (and shrinking again). To the player (and myself), it looks like the player is bouncing.

2/3/2024

Tyler fixed the crouch by making the trigger check for a specific component rather than the player’s capsule. The issue was caused because the shrinking/expanding capsule would do just that…shrink out of the volume which activated the regular crouched height, then expand to regular crouched height, which activated the shrinked height again. That’s why I was “bouncing.”

**Leaning**

2/5/2024

Lean mechanics have been successfully implemented after I briefly laid the groundworks for them on 2/3/2024. Overall, I’m really satisfied with how easy it was to implement compared to…*bend time*… The only issues that arose from the implementation was that it gave the player the ability to see through meshes in the environment, which could cause balancing issues later on down the line. This issue was fixed by adjusting the position of the First Person Camera component in relation to the player’s collision capsule.

The lean itself was implemented via two separate variables. The first is a reference to the FP Camera’s position, while the second is a blank placeholder reference to the same thing. When the player activates a lean the camera’s position in the world relative to the player is updated to shift (x) along its Y-axis. This accounts for crouch height as well. The proper values are transferred by adding (x) to the placeholder variable, and setting the placeholder variable to the actual camera location while lean is active.

For example, in my program, I add 40 to the camera’s Y-axis value, offsetting the positioning and allowing the player to lean to the right. To lean left, I subtract 40 (by adding -40).

**Clamber**

3/2/2024

Using a similar implementation to that of the Lean mechanic, I created a trigger volume that automatically sets the player’s maximum step height on their character movement component to quadruple the standard size of 45cm. If the player blinks towards a ledge/surface with that volume, they will (as long as they are simultaneously walking forward) step upwards onto the top of the ledge.

I want to provide some form of visual indication when the player is blinking via another Niagara system emitter. I’ll probably do something like cast to the blink target, and set the Niagara system from the normal blink distance indicator effect to a new effect.

Additionally, I’ve included some basic functionality that allows the player to step up on top of the ledge with the surface. This is a band-aid, however. It teleports the player to the top of the ledge by using the origin of the trigger volume and adding a float value of 20 to the z value of the origin, and setting that new vector to the player (actor’s) location.

3/13/2024

I think this is something I’ll need to talk to Kent about. I tried generating overlap events with the BlinkTarget, but I don’t know if it was working or not. On the bright side, I now have yet another new Niagara Emitter that isn’t being used.