CS354R Assignment 2 Project Retrospective Report

Felipe Paz, Seth Parsons, Thomas Moore

Felipe Paz: player movement/rotation, gliding, double jumps, walkable angle

I used the singleton instance of Input and processed any WASD user input so it would be reflected in the game according to the state of the player. The player movement was fairly straightforward; it is stored in a vector that reflects direction relative to the character and it has to be adjusted before processing it. The adjustment of it for the player rotation was a bit of a challenge at the beginning. The move_and_slide function from KinematicBody is used for this movement. This function was really useful when dealing with the walkable angle, since it has a parameter that indicates the angle at which a surface is considered a floor or a wall. This angle is a property in the player node.

Double jump was very easy. Similar to the jump from the ground, it is done by adjusting the movement vector of the player and keeping track if a previous double jump was performed. Gliding was a little more complicated, since the gravity needs to be adjusted and user input should still be processed.

Thomas Moore: GUI, object interactions, audio

I worked on the GUI, object interactions, and audio. Implementing the GUI was fairly simple - I followed along with the recommended Godot GUI tutorial and adjusted the GUI based on our game's needs. It was fairly easy to figure out how to update the GUI's labels and progress bars whenever there was a successful collision. I will note here the terrible syntax that comes

with converting between Godot string's and C strings - this was probably the worst part of the GUI updates.

Token interaction was fairly easy to implement too. There was a helpful guide on StackOverflow that described connecting an Area to the KinematicBody to help with Area collisions - this was the main route I chose to go through with Token interaction. The player's KinematicBody would walk over the Token's Area, and since the player has an area connected to its KinematicBody, I could use the area_entered signal to handle token interactions. Hooking up the token interactions to the GUI and sound was easy.

The main difficulty on my end was trying to figure out how to interact with the spikes. At first, we initially intended for spikes to be permanent StaticBody nodes that were part of our level - interacting with them would not delete them from the level. However, we could not get the collision to properly work with our player. I first tried using move_and_collide() to detect collision with the spike - no dice. It was super inconsistent to even get move_and_collide() to provide a response, and every time it did none of the objects that were collided with registered as spikes. I then tried to use move_and_slide() - a similar situation occurred. Then, I attempted to use the body_entered() callback from the Player's area. This yielded no results either. I then tried to register an Area node for the spike as a child of the StaticBody, hoping to use the area_entered() callback. This approach also yielded no results. In the interest of time, we ultimately had to compromise and implement the spikes in a similar fashion to tokens - walking over them consumes the spikes and damages the player with no physics response. Overall, it was extremely frustrating trying to make collisions work between our Player and spikes, mostly because the move_and_collide()/move_and_slide() functions are slow, inefficient, and inconsistent.

I don't really have much to say about integrating audio. I found a bunch of free clips on a website and hooked up the imported streams to various AudioStreamPlayers. There's a keybind to mute audio, and there's an option within the editor to start the game with audio enabled/disabled. Muting audio just pauses the background music and prevents the object interaction sound bytes from playing.

Seth Parsons: camera movement, basic jumping/gravity, and ledge stop/hang/fall

Camera movement and basic jumps/gravity were pretty simple - easier to implement than I expected, mainly because of the wealth of resources on the internet I could reference while writing up the code (like YouTube tutorials with GDScript I could reference for getting mouse input, rotating a camera node, or Godot API documentation for how the move_and_collide() function works, what the vector structures are, etc). What was much harder was the ledge interactions - those apply less generally to any random person's games and I had a harder time figuring out exactly what math I needed to use to get the results I wanted. Raycasts I understood, but I had a hard time figuring out how to make the player snap to the direction of the ledge once the game decided they were on a ledge. This case arose because I'm a bit rusty on my vector math and online resources were scarce - in the future I think I'll try to look for more math-focused help rather than godot-specific help, but definitely continue to reference the godot forums and reddit posts as they were super helpful during this assignment.

Sources

GUI assets were sourced from:

https://docs.godotengine.org/en/3.0/getting_started/step_by_step/ui_game_user_interface.html

Sounds were sourced from: https://freesound.org/

Collision help:

https://stackoverflow.com/questions/69728827/how-do-i-detect-collision-between-a-kinematicbody-2dplayer-node-and-a-rigidbod

Snapping to ledge direction help:

https://godotengine.org/qa/132087/how-to-make-the-character-face-the-plane-you-are-climbing-on