ZERO TEAM - Assignment 2: Physics and Player Package

Team Members: Felipe Paz, Seth Parsons, Thomas Moore

Video Demo: https://www.youtube.com/watch?v=9TuRDIMPuCM

Overall Game Design:

For this assignment, we decided to implement a platforming game. This was the natural decision to make as we felt it would be easiest to incorporate the assignment requirements in a cohesive and fun manner. We could showcase different movement and character physics by interacting with various terrain, and it would be easy to develop a token collection system that would give our game an objective.

The goal of our game is to make it through the level - utilizing various types of terrain traversal while avoiding obstacles - and to collect as many tokens as possible. The game will be in 3D played from the 3rd person perspective behind a simple rectangular character model. This model will be able to complete basic movements: directional movement, jumping, falling, and walking on an incline. The character model will also be able to interact with ledges in three different ways: stopping at a ledge, falling off a ledge, and hanging from a ledge. The character model will also be able to glide through the air. For our custom movement, we decided to implement a double jump for the character. In terms of object interaction, there will be collectable tokens that give the player points, and harmful spikes which take away HP when walked into. Object interaction will be reflected through the GUI: there will be a HP bar and a token counter. This will be a simple bar located in the bottom left corner of the screen, and use numerical values starting at 100 for the HP and 0 for the tokens.

This game will likely be linear, and we will try to guide the player through different movement styles throughout the level. This will result in a more interactive experience as the character model performs more unique actions. In order to improve the interactive experience, we plan to add sound effects - an idle sound effect while the character is simply moving, and sound effects when the character interacts with tokens and spikes.

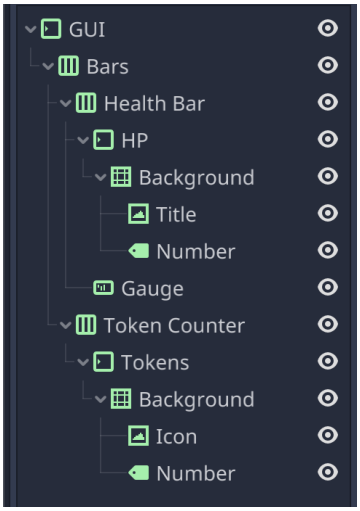
Software architecture and design plan:

We will likely break up our game into two components: our player package, and our level package. The following will be a description on how we plan to break each package up into smaller components, and how we plan to implement these components and integrate them together. The following describes the current state of our milestone.

Player Package

Scene hierarchy:

We started by designing our player node in Godot. The root node of our player is a KinematicBody node. This is because we are trying to implement the physics environment ourselves and not rely on the Godot provided one. The root node has one child, a CollisionShape node for collision detection. Our CollisionShape has a child node for our rectangular MeshInstance, and a smaller triangular MeshInstance node was placed on top of the rectangle to denote the forward face of the mesh. We will eventually decide on textures to use for the mesh, sourced from AmbientCG.



Within the Level scene, our Player node has two children: a Spatial node called Pivot that controls camera functionality, and a MarginContainer for our GUI. The scene hierarchy for our GUI is pictured on the left.

Functionality:

We started by trying to implement basic directional movement of the character model based on directional input. This was implemented successfully by writing an _input function that handled activated input singletons mapping to the user's input map. There is an option for left/right directional inputs to use either rotational or strafing movement. This option can be turned on via the Godot editor. We recommend that rotation be kept off, as the camera left/right directional mouse input handles rotation of the character.

After basic movement was implemented, the camera functionality was developed. This was done by attaching a camera directly behind the character, and attaching out camera script to this node. The script essentially makes the camera respond to various mouse inputs. If the escape key is pressed and the mouse is uncaptured, left click recaptures the mouse. Swiping in each direction will rotate the camera with respect to the face of the model. We capped vertical rotation directly above the model so that we don't loop over the top and force unnatural rotation of the model. We also implemented a raycast for the camera object to zoom into the model when the camera collides with a shape. This is to help with rotating under the model, as well as rotating the camera into any walls. An image displaying this functionality is shown below.

We followed this up by implementing our gravity and jumping. Gravity was implemented by keeping a separate fall vector for our character, and updating this vector's y by subtracting gravity * delta whenever the KinematicBody was not in contact with the ground. Jumping by pressing the spacebar sets this vector's y to 5. Falling past a certain point in our level resets the model back to its start position and sets the fall vector's y back to zero.

We have yet to attach GUI functionality, as we have no meaningful ways to interact with our level to affect the GUI, but there is a health bar and a token counter set up. The picture above shows what our GUI currently looks like, assets were sourced from the Godot GUI tutorial. Colliding with spikes in our level and falling off the map will decrease health, while picking up tokens will increase the token counter.

For the future, we plan to implement walking on an incline, gliding, a double jump, and ledge interactions. We haven't done much brainstorming for each movement, but we don't anticipate each giving us much trouble.

We believe that double jumping will be the easiest - we will likely implement some sort of boolean that detects whether the player has jumped once already and is midair. If the boolean is set to true, the player can press jump again to double jump - this sets the boolean to false to prevent infinite jumping. The boolean will be reset whenever the player hits the ground.

After figuring out the double jumping functionality, we will work on designing a level that allows for working on walking at an incline. Once we figure out that functionality, we will work on ledge interactions. We will likely have a player raycast that detects collision with the edge of a ledge, and prevents the player from simply walking off. Jumping over the ledge at this point will allow the player to fall off the ledge. We have yet to decide on an approach to ledge

hanging, though initial thoughts point towards a specific button input allowing attachment to an attachable nearby ledge.

We will work on gliding somewhere in parallel here. This will likely be done by turning gravity off and having a constant decrease in height as opposed to acceleration with respect to gravity. We have yet to test it out, but we assume that this is sufficient functionality for simulating gliding. We may play around with triggering this functionality with a separate input while midair, and perhaps orienting the character model horizontally while gliding. We will have tunable parameters to affect the angle and speed at which gliding occurs.

Level Package

Scene hierarchy:

We have a parent Level node that contains our player child and will have various StaticBody nodes as the environment. We have yet to implement anything but a simple floor for our environment, but anticipate that we will have various different types of static bodies to reflect interesting terrain.

Functionality:

As of now, falling off the floor body makes the character model fall until a certain height at which the character's position is reset to start.

Division of labor:

Seth P. worked on implementing camera functionality and jumping. Felipe P. worked on directional movement. Thomas M. worked on this report and implemented the GUI. We have yet to divide responsibilities for unimplemented functionality.