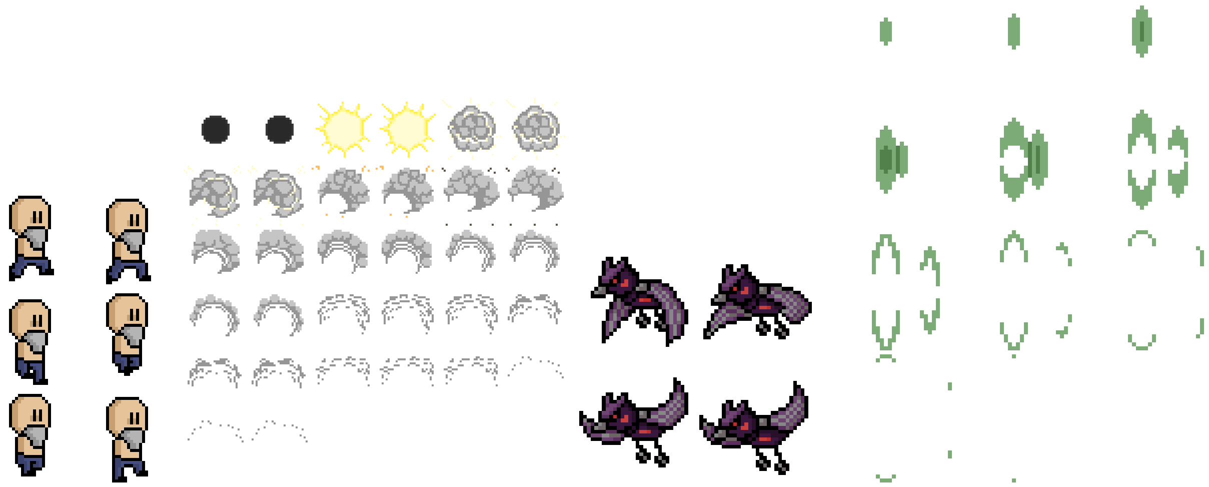
Team Ornithophobia Design Rationale

By Sameer Bugti

Pattern Justification

State Pattern

The first design pattern I would've implemented in my part of the group project is State, the behaivoral pattern. This is because my section has now been focused on the visual aspect, like animation handling, and that is best done through this pattern. While using state handling to swap whatever displayed animation is on screen is best on a large scale, our project is not going to be that big, so I've decided to use a basic logic handler through the player's variables. I access the player characteristics through the parent node in Godot, and through various checks on whether the player is touching the ground and whether they are moving, it plays the correct animation. Attached below is a prototype of the animated player character. It has been updated for the final deliverable, as well as added VFX and SFX for quality of life.



Various spritesheets all created by me!

Iterator Pattern (Scrapped)

Unfortunately, the initial leaderboard feature has been scrapped from our design. However, if it was still to be implemented and under my supervision, I would use the Iterator behaivoral pattern. The leaderboard would be stored within the local level editor, and would be read in as a linked list, with each ranking pointing towards the next. This would allow the Iterator to be easily implemented, as linked list traversal uses iteration by default. Another thing to note is that my team is made up of 3 people, so while I only have one pattern implemented, my team's total far exceeds the 2 required for partial implementation.

Testing Report

While Edward has handled all the testing, and his documentation contains all of that, I would like to use this section to clarify on some troubles we encountered. Firstly, due to the nature of our project's workspace (which was using GDExtension to make C++ compatible with Godot), we were not able to use Google unit testing, which was what was expected of us. Instead, we came to a compromise using Godot Unit Testing, a built-in functional library of tests that we could utilize almost identically to Google's version. This has been approved with the professor and you can refer to Edward's contributions for all specifics on testing.

Challenges and Solutions

Godot AnimationStateMachine

While going over some patterns, I noticed that the Godot engine actually innately supports many of the patterns as custom nodes. One of which being an AnimationStateMachine Node. As the name says, this node handles the animation of any node through a state machine, implementing the State behaivoral pattern. My first thought was to use this in tandem with the player controller to satisfy the pattern requirements, but I soon realized that would not be possible. Because Godot uses its own custom language called GDScript, we had to convert all relevant codespaces to C++. Because of this, basic Player nodes were transformed into Custom Nodes, where they would inherit all properties of the nodes they are based off of, plus some more to satisfy pattern design. However, the AnimationStateMachine node required that the parent node be a basic Player node, making it incompatible with how my other teammates were configuring their code. Since this ended up being the case, the compromise I came up with was a Custom AnimatedSprite2D node, where all the state switching (or logic) is handled through the codespace instead of the node. Other than that, there haven't been many challenges through my coding process.

Other than that however, this project was mostly smooth for my contributions. I had great teammates to fall back on whenever I encountered issues, and help online if none of us knew what to do. The freedom in allowing us to use Godot really helped us realize what we wanted from each of our problemspaces, and for that I'm grateful. Personally, I recognize that most of my contributions have ended up being art related, as that is what my problemspace tackles, but I do think I did a fair bit of coding (atleast enough to justify contributing as a developer of the game). Finally, I would love to continue this project outside of class and give it the love it deserves, as I’m clearly padding to make this report look more professional. Thank you for reading and have a great winter break!