Boxeymon: A Real-Time Turn-Based Combat Game Demo

Daniel Creighton

Youngstown State Univeristy

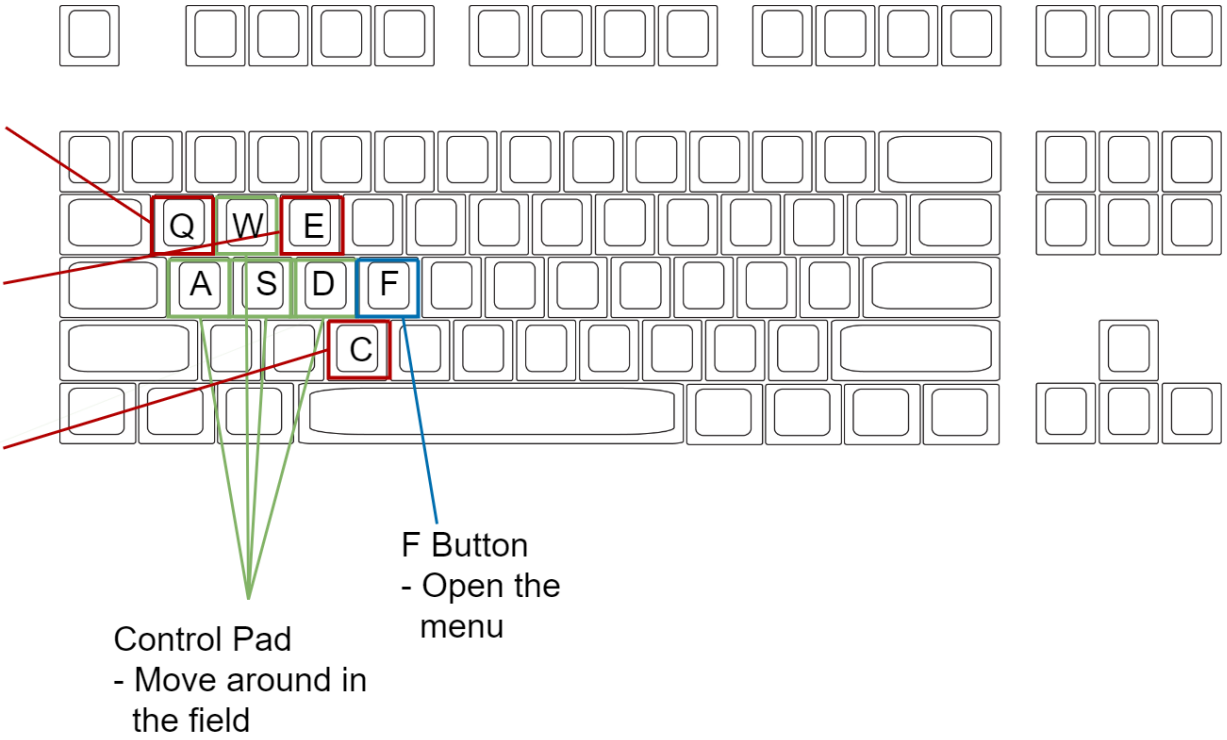
CSCI 4890: Computer Projects

Dr. John Sullins

December 14, 2022

User Manual

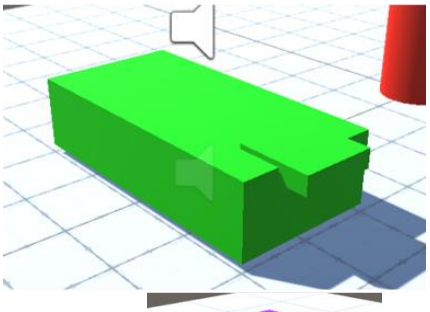
BASIC CONTROLS



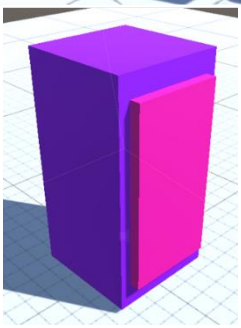
Camera controls

- Once the player has entered the game, the mouse is used to control the camera

OBJECT OF THE GAME



A battle is initiated at Spear Pillar on the peak of Mount Coronet! You and your trusty partner Pokemon, Torterra , are set to face the legendary ruler of time, the mighty Pokemon Palkia. Start the battle and avoid attacks that come your way, all while sending attacks of your own. Let your mana meter build over time and spend your mana on commands to attack. If you're cornered, spend your mana on healing resources instead.



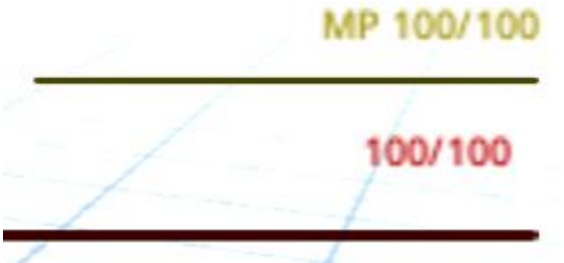
THE MENU SCREEN

Health Bars

- Display current health for the player, companion, and boss

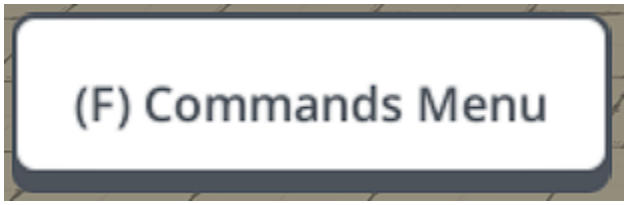
Mana Meter

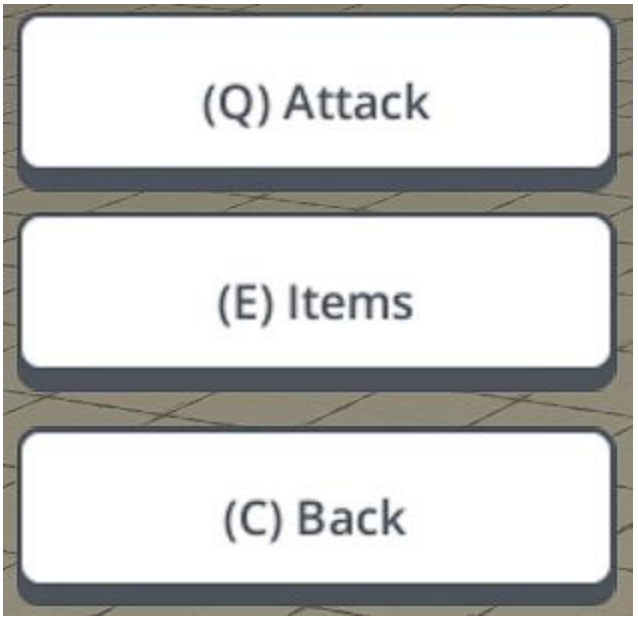
- Generates mana resources over time
- Mana is spent on options inside the combat menu



Combat Menu

- The main on-screen menu, navigate between the sub-menus and spend your mana on powerful attacks or vital healing

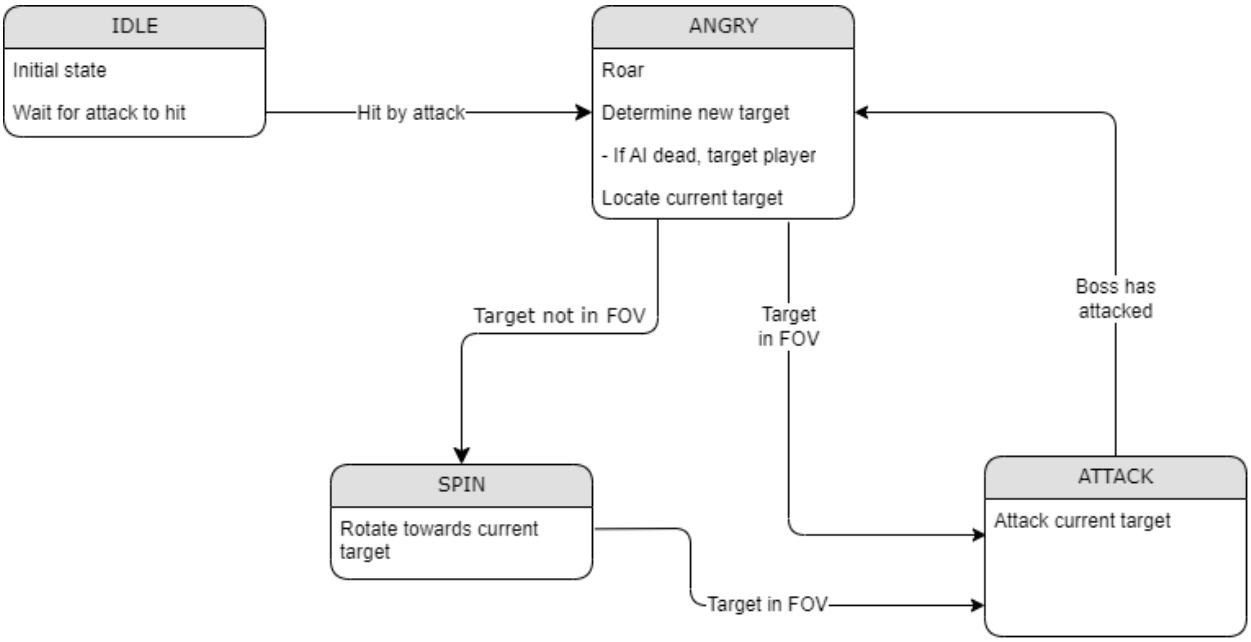




Design

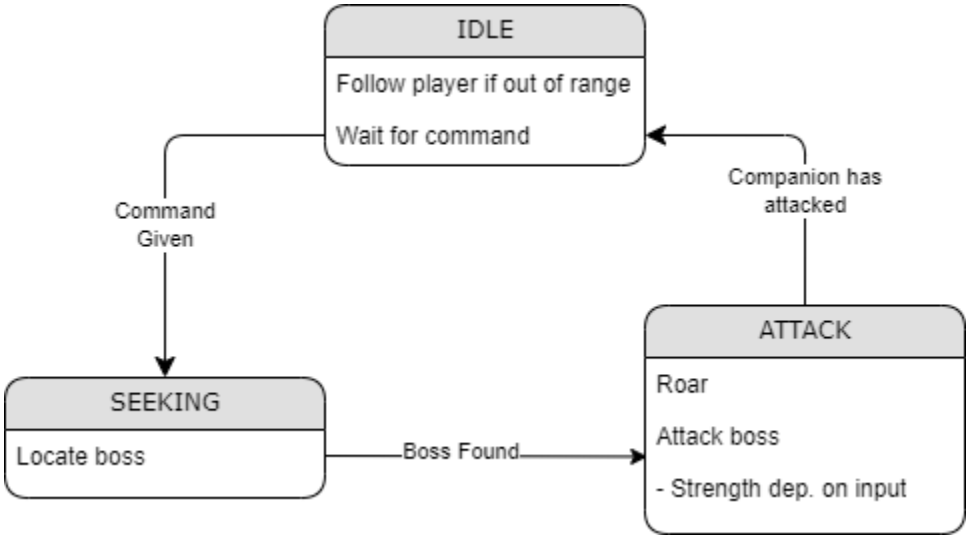
The most important object to design was the player. Without it there is no game at all. Luckily designing the player was fairly straightforward. The difference between 3D and 2D was a little daunting at first but I quickly got used to it. I knew I wanted to have the mouse be used for the camera, so the plugin cinemachine was handy in that regard. Movement along the x and y axes (x and z in Unity) was tied to the direction the camera faced, that way forward was always relative. Originally I wanted the player to jump too, but I couldn't get it to work and it really wasn't necessary in the game, so it was better to just remove that feature.

When it comes to the boss and the companion, I designed a couple finite state machines for both. Neither was too complex in order to streamline things.



Boss FMS Design

The boss starts in an idle state because I didn't want the player to start being attacked as soon as the game loaded. Inside the angry state, the boss selects a target using a weighted number generator. A higher weight is given to the companion to ensure the player doesn't die too soon. The location of the chosen target determines the next state. If the boss can see the target in its field of view, it goes straight into the attack state where it launches its attack. If not, then it goes into the spin state to aim towards the target and then goes into the attack state from there. After the boss attacks it returns to the angry state, chooses a new target (that can be the same target as before), and the loop continues.

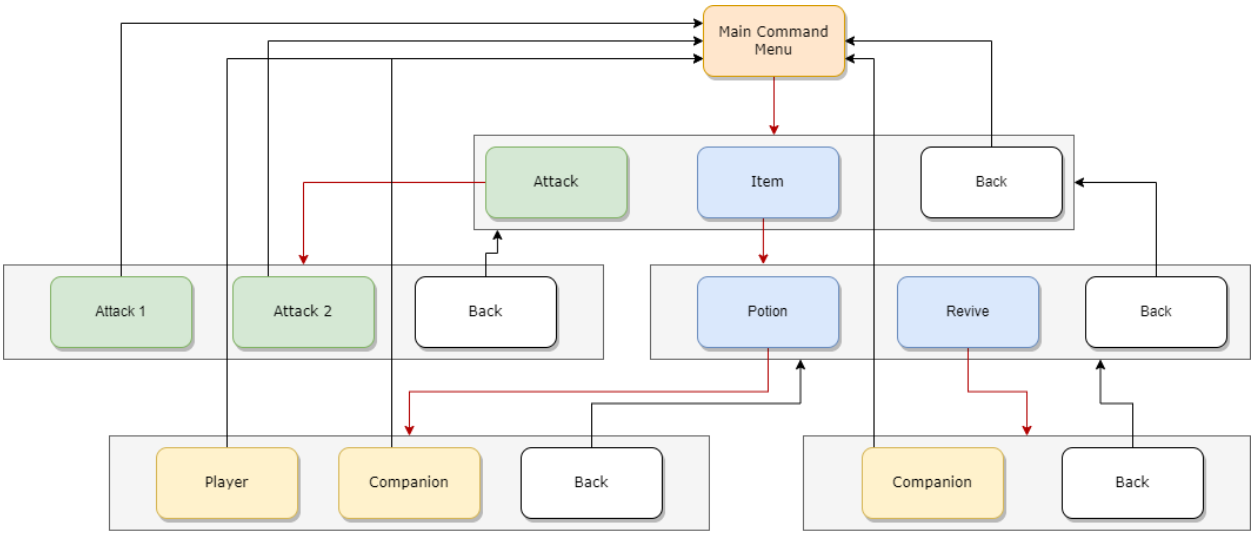


Companion FSM Design

The companion is even less complex than the boss. A straight loop through three states, the companion starts in its own idle state. In this state it can follow the player around as they move around. Besides this, all it does is wait to receive a command from the player, which is given when the player selects an attack in the command menu. Once the companion receives a command it moves to the seeking state where it uses the same field of view mechanic as the boss in order to aim at the boss. After it locates the boss it moves to the attack state where it sends the attack that the player requested.

As for other features, some of them are pretty simple. The health bars all display the health of their assigned targets. The mana bar took inspiration from the ATB meter in the Final Fantasy series (VII Remake specifically). All it does is fill a bar from 0 to 100 for a specified amount of time. The final version was supposed to be slower, but I thought the current speed that I used for testing also worked for gameplay. The weighted number generator mentioned before works with distribution between a table of values rather than a percentage. Right now the companion is set to 80 and the player is

set to 20. If you pulled from a hat, there would be 80 companions and 20 players in the hat. The field of view mechanic is used for the boss and the companion. It checks for any GameObjects that overlap with a set radius on a specified layer, verifies if a GameObject is the current target, and calculates the distance between the user and the target. The combat menu is just a series of button components connected to each other. The diagram shows how they communicate with each other. The lowest levels also call functions depending on the input given, such as attacks calling attack functions and so on.



Combat Menu Design

Personal Thoughts

It's already been sprinkled throughout this paper but my main inspirations for this project were the Pokemon series and the Final Fantasy VII Remake game. I thought a mix of the combat from the two would make for a fun little demo. Moving into 3D for the first time was scary at first but I did start to get a grasp on things. The biggest hurdle I never got around to were models and animations. I knew learning to model and rig

something could have been a showcase on its own so I didn't want to spend all of my time on that. I really enjoyed the Game Programming and AI classes I took and wanted to apply the knowledge I acquired in them, so I made my focus the design of the game. There were some features that I couldn't add due to time or other reasons, like a flow of time manipulation or auto-attacks from the companion, but I think the end product still turned out well without them.

My personal favorite part of my project is probably the on-screen combat menu. It took a little bit to get working but once I did it was super satisfying to see it work exactly as intended. It was also easy to expand and add more options when I needed to. An unexpected hiccup that occurred was the speed at which my game played. My AI would go through their states almost instantaneously and it would ruin the entire flow. I had to suddenly find a way to delay the transitions of states and it was super annoying. Despite the roadblocks and missing features, I'm proud of how my project turned out. The feeling I get when I make something is always worth the headaches to me. Being able to step back and say to myself "I made this" is always surreal but makes me excited to see where I can expand into and how I can grow with future projects.