Preston Schnell

Mahmood Jasim

CSC 4444

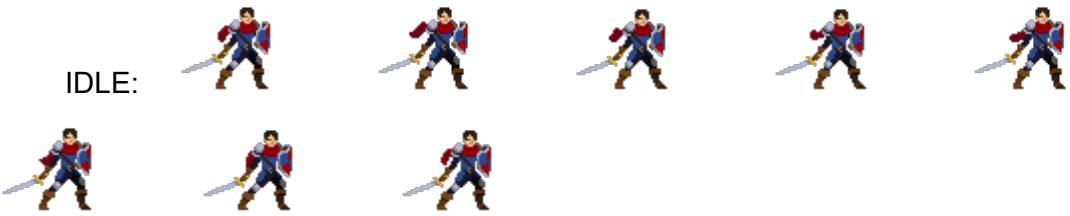
28 April 2024

Final Project Report

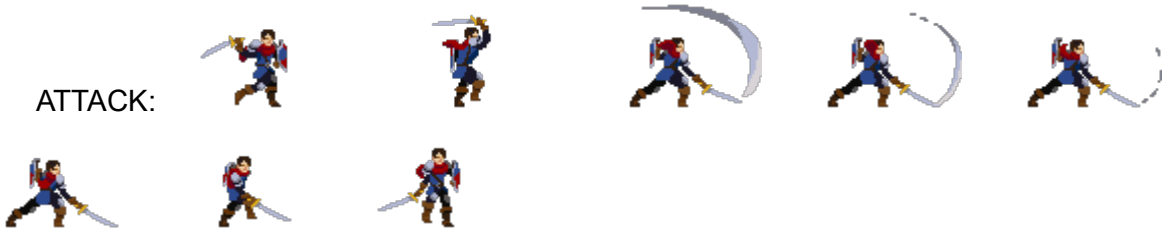
For my final project, I was tasked with creating an RPG game with an adversarial agent that would attack my character, heal themselves from damage, and respond to consecutive attacks to overpower the player's options. The character who is left standing with the most health is the winner.

In the game, the adversarial agents' tasks are to minimize the player's health to maximize their returns. I have designed the enemy NPCs to respond to each one of my actions such as swinging a sword, blocking attacks, and healing from damage. The AI has unique animations for attacking, healing, taking damage, and dying. I designed their animation system using free sprite animations found online and gathered inside of my main project files. I took their names and called them into my code in order from 0 to 7. 0 would represent their starting state, and 7 would represent the end of the animation. These would be played repeatedly based on the action being recalled using a loop. For instance, the idle animation is constantly running since the character must stand there, but when the attack function is called, then the idle animation is interrupted and the files 0-7 for the attack animation are played.

IDLE:



ATTACK:

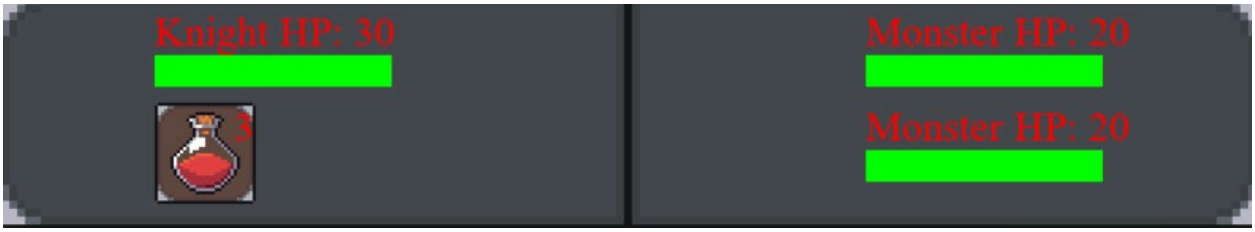


In addition to the main functions that the AI NPCs perform, they also act upon a delay system that waits for each turn to be over before the next one can begin. In other words, the AI NPCs can only attack once the player finishes their turn. Conversely, the player can only attack when BOTH enemy AI NPCs have finished their turns. In this sense, I wanted the player to feel challenged, and be encouraged to take different approaches to their gameplay and style. Would

the player want to attack an enemy one by one? How about switching back and forth between turns and healing? It all depends on how they approach the environment based on the rules of the game.

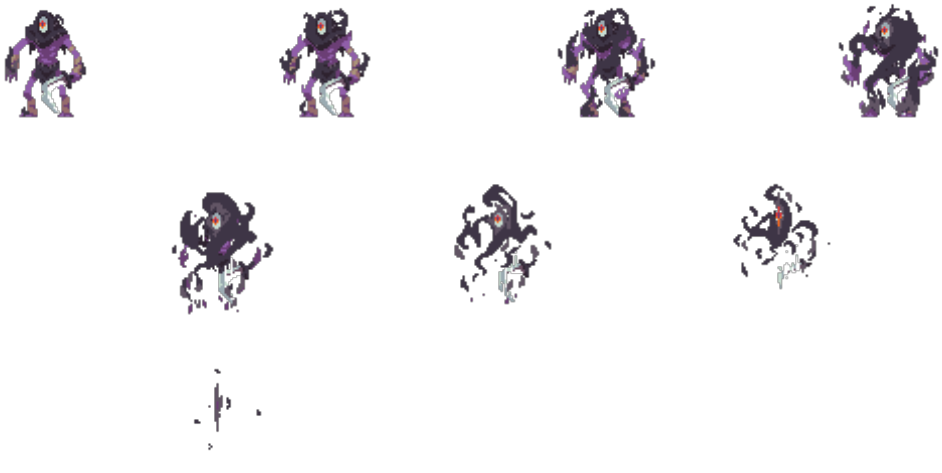
My motivation behind this project was to build a turn based artificial intelligence to further my capacity for game design. The enemy NPCs have intelligent responsive actions that are meant to minimize the player's advantage and maximize their own. The enemies act based on a system that is implemented using pre-defined values of player and enemy health, amount of hp healed using potions, and a random function that determines the amount of damage dealt. I knew when I started implementing these functions that I wanted to make something entirely different from others using an animation system so that I could see what I was doing more than having to implement static features.

In the creation process of the potion system, I made sure to declare a set value of player health and amount healed and designed the function under player health to not exceed the max_hp. More accurately, the player's amount healed would not exceed their max health, nor would they be able to heal without losing a turn. I indicated this system with the classic red and green bars, indicating to the player they were losing or gaining health without the green bar exceeding the previously declared space. I had some problems with the green bar exceeding the player health interface in the past, but figured it out nonetheless.



The health interface is also indicated by sprite animations based on the current state. When healed, the amount will be placed above the sprite with green text, while damage done will be indicated by the proper amount in red text. Whenever an NPC does not have health the system stops taking turns from the NPC in the dead state, and moves to the next available enemy.

HURT: 

DEATH: 

Since I specialize in game design, I was happy with the way this project turned out both aesthetically and mechanically. Throughout my coding process, I utilized a variety of different resources to figure out the best ways to implement my functions without wasting time. In my project, I used PyGame, which is an amazing resource and widely used for industry standard game mechanics and systems. I also implemented a button module in a separate file since I was having trouble getting my potion system to work. In the creation of this game, I utilized Python, PyGame, Sublime Text, and the Unity Asset Store (Free Pixel Art).