**CSCI 120 Final Project: The Adventure of Pie**

**Team Members**

* Muhammad Suhayb Malik (Section 1)
* Lucian Terhorst (Section 1)
* Kadin Reed (Section 1)
* Matthew Zaluski (Section 2)

**Introduction**

Our project is a turn-based combat game, inspired by the combat system of Pokémon, that follows three battles of our player character: Pie. The name of the game and main character is a clever turn on Py, since it is coded in python and based on the pygame library. The three enemies gradually increase in difficulty such that your attacks need to be carefully planned to defeat the final enemy.

As a balancing tool, the game features a stamina cost to using abilities, because of which the strongest ability also has the highest stamina cost and thus cannot be used repeatedly. Health replenishes each round, but stamina does not, thus there is also a resource management aspect to the game.

**Structure and Components**

The main project folder is titled: ‘CSCI 120 Final Project’ and contains two python files called ‘Classes.py’ and ‘Source File.py.’ These contain all the code for our project. Classes contains two classes, which are named player\_brawler, containing methods for the player abilities as well as stores for their health and stamina, and the Enemy class. The Enemy class stores the enemy’s health and a random range between which the enemy does damage each turn. Our Source File contains the implementation of these classes and methods to create a playable game. It uses pygame to import assets and display them on the screen. The class methods defined earlier are used within our event loop to control the game states.

Our folder also contains all our assets, which except the logo and font have all been designed by members of the team. The font and game logo folders are quite self-explanatory. The graphics folder contains all our background assets, player character as well as our enemies. The Music folder contains the song which loops in the background.

**Key Functionalities**

As stated in the previous section, the Classes file starts off by defining the player\_brawler class. It is initialized with max health, stamina, damage done by abilities as well as Booleans that control whether it is the player’s or the enemy’s turn. Methods are defined to switch between turns (called set\_my\_turn), heal the player and compute how much damage he has taken that turn. Next, we have four methods that control each of the abilities that the player character can use and finally a function that checks if the player is still alive after each turn.

Secondly, the Enemy class is initialized with max health, and a range between which it deals damage. Methods to compute damage taken, deal the random damage, and check if the enemy is alive are then implemented.

These classes gave us a very good framework upon which to build our game. Now we had to implement these into our pygame source file. We first defined display\_player\_health, display\_player\_stamina, and display\_enemy\_health functions to dynamically display these on our screen by constantly being updated by our class methods.



We initialize pygame and set our screen as a 1200x600 window with our logo and title. Pygame is then used to import our images and fonts and the get\_rect method allows us to essentially make hitboxes for our images. This is also where the player and enemy objects are declared. Some Boolean variables to control and switch between game states and stages are declared here as well. For creating text, we use our font with the render method.

Finally, we create a while loop that displays our assets and their interactions on the screen. The first part of this is the event loop, which checks for player and enemy attacks each turn and uses class methods to compute their consequences.

Outside of this, when the game is active, the blit() method allows us to display assets on the screen. When the game is not active, we see either a lose or a win message with the option to restart by pressing the spacebar. The functionality of this is also handled by the event loop. Finally, at the end of the loop, the display has to be updated and an fps is set to ensure similar functionality on all computers.

**Known Problems**

One problem we faced while coding was using .sleep() to time things like hit animations and attack reactions, as this would cause the whole program to come to a hault. A limitation we ran into was being unable to impliment animations for characters and attacks due to the issue with the .sleep() function. Also, we were limited by time, not being able to implement a start screen and a more dyanmic HUD layout for things like health and stamina bar.

**Potential Future Extensions**

In the future we could add an item shop screen where the player could purchase potions and powerups in between fights to add another layer to the battles. There could eventually be an option for a second player to play as the enemies instead of them being controlled by the computer. Also, we could impliment a feature where the player could fight multiple enemies at once to increase the difficulty.

**Team Member Contribution**

As the team leader, Suhayb guided group members into which areas of code they needed to impliment to the starter code. He focused on player input, as well as the gameplay functionality between the player and enemy. Lucian directed the creative narrative and gameplay aspects as well as focused on coding the classes. Matthew focused on coding the interactive HUD as well as the hit and background animations. Both Matthew and Lucian created the artistic assets of the game, which included the player, the enemies, the background, and the music. Kadin was the general debugger and created the structure that the functions and classes would be based upon. He also scheduled out times in which out group would meet and set individual goals for what we needed to accomplish.

**References**

We referenced this video for pygame information: <https://www.youtube.com/watch?v=AY9MnQ4x3zk>

We used this website for the creation of sprites: <https://www.piskelapp.com/p/create/sprite>

We used this website to create the music: <https://www.soundtrap.com/studio/>