Lab 12: Project

CSE/IT 107

NMT Computer Science

> "The limits of my language mean the limits of my world."
>
> — Ludwig Wittgenstein

> "Any sufficiently advanced technology is indistinguishable from magic."
>
> — Arthur Clarke

> "Imagination is more important than knowledge."
>
> — Albert Einstein

---

1 Introduction

This will be your final lab for the semster.

2 tkinter

tkinter is a basic graphics library for Python. You will be using it as your user interface for this lab.

3 Project

Boilerplate

Remember that this lab *must* use the boilerplate syntax introduced in Lab 5.

For this assignment, you will be creating a multiplayer game similar to the combat system of the Pokémon or Final Fantasy games. You will be writing two programs, though odds are they will be sharing a decent amount of code. Your two main files should be named battle.py and battle_server.py, though you are encouraged to have more than just those two files.

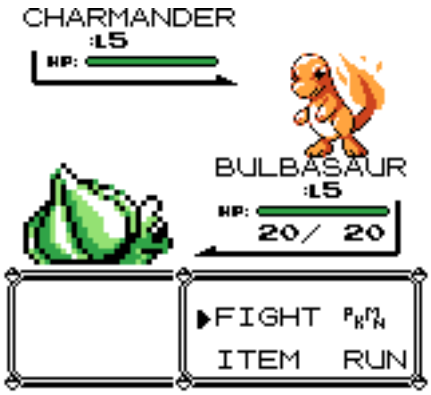

Figure 1: Pokémon battle screen. (from Wikipedia)

3.1 Game

The game you are creating will be a two-player battle game. Each player will have a "party" of monsters. At the start of the match, each player will choose one of their three monsters to become active. Then a series of turns will occur until one player's entire party of monsters has been defeated.

3.1.1 Turns

A turn consists of each player choosing one action to perform. The possible actions fall into two categories: switching the player's active monster or using a move from the currently active monster.

If a player chooses to switch their active monster, they must then choose from their current inactive and undefeated monsters. The selected monster will become active while their current active monster will become inactive.

If a player chooses to use a move from their currently active monster, they will then choose one of up to four moves that their monster knows.

Once both players have chosen a move for the turn, the moves will be executed. If both players chose to switch monsters, then the order that the switches occur does not matter. If one player chose to switch their monster while the other chose to use a move, then the switching player's action will go first. If both players choose to use a move, then the player whose monster has a

higher Speed stat (see Section 3.1.2) will perform its move first. The second monster will only perform its move if it is still undefeated after the first monster performs its move.

If at the end of a turn either or both monsters have had their Health reduced to 0 or less, then that monster has been defeated. Its user must immediately switch out their monster for one that has not been defeated. This switch occurs before the start of the next turn. If no undefeated monsters remain, that player has been defeated. If both players are defeated at once, the match ends in a draw.

3.1.2 Monsters

Each monster will have a number of stats:

Attack Indicates attack power of the monster. The base damage of any attack Move that this monster uses is modified by

$$base\_damage \cdot \left(1 + \frac{attack}{100}\right)$$

Defense Indicates defensive strength of the monster. Any incoming damage to the monster is reduced by

$$damage \cdot \left(1 - \frac{defense}{100}\right)$$

A monster is not allowed to have a Defense greater than 75.

Health The maximum health that a monster has. Any time it is attacked the monster's current health will be reduced, thought the maximum will stay the same.

Speed Indicates the speed of the monster. Is used to determine which of the active monsters will use its Move first.

Name Each monster must have a name so that you can properly empathize with its plight.

In addition to these stats, each monster can have up to 4 moves, as defined in Section 3.1.3

3.1.3 Moves

A Move is an action that a monster can perform when it is active. There are four types of Move:

Offensive An offensive Move deals damage to the opponent's active monster. Each offensive Move has a base damage stat that helps determine how much damage it does, based on the two formulas in Section 3.1.2.

Heal A healing Move restores health to the monster making the Move. A healing Move has a stat indicating how much health it will restore. A monster cannot be healed above its maximum health.

Buff A Move that temporarily increases one of the active monster's stats. A buff move has three attributes: the stat it increases (Attack, Defense, or Speed), the amount it increases that stat by, and how many turns it lasts. A buff can allow a stat to be increased beyond the point it normally could be.

Debuff A Move that temporarily decreases one of the stats of the opponent's active monster. Has the same attributes as buff, but decreases instead of increases.

In addition to the above mentioned attributes, each move must have a name.

3.2 Client (battle.py)

You will need to write a client using tkinter that provides a nice user interface to those playing the game. You should model your interface off of Figure 1, though you are free to modify it as you see fit. The interface needs to do the following:

- Display the name, current health, max health, attack, defense, and speed of the currently active monsters from both players.
- Display information about what is happening in the game. That is, what moves each monster is using, what the results of those moves are, what monsters are switched for
- Allow the user to choose between their various options each turn, as shown in Section 3.1.1.

In addition, your client will need to connect to your server in order to send the actions of its player and receive the results of each turn. It will also need to receive information about the opponent's monsters at the start of a game.

When the client is first started, it must load the information about its party of monsters from a text file. This information must then be sent to the server upon connection.

3.3 Server (battle_server.py)

You will need to write a server that allows two clients to connect to it in order to play the game that I really wish I had come up with a name for. This server will carry out the majority of the game's logic and handle sending and receiving information from the players. It will need to fulfill the following tasks:

- Receive information about each player's party, store that information, and send it to the players as necessary.
- Wait for each player to select an action, then calculate the outcome of both actions.
- Detect when one player has won and inform each player.
- Anything else you find would be handy for the server to take care of.

You are free to use whatever protocol you wish for communicating data between the server and the clients.

3.4 Extra Credit

3.5 Terms

Active The current "fighting" monster. Each player can only have one active monster at a time.

Damage The amount subtracted from a monster's current health after it is attacked. The full formula for damage is

$$base\_damage \cdot \left(1 + \frac{attack}{100}\right) \cdot \left(1 - \frac{defense}{100}\right)$$

Defeated A monster that has had its health points reduced to 0. A defeated monster cannot be active.

Monster A mysterious creature that we are using to fight for questionably moral reasons. (Section

4 Submitting

Files to submit:

- battle.py (Section 3)
- battle_server.py (Section 3)
- Any other files created. (Section 3)

You may submit your code as either a tarball (instructions below) or as a .zip file. Either one should contain all files used in the exercises for this lab. The submitted file should be named either `cse107_firstname_lastname_lab11.zip` or `cse107_firstname_lastname_lab11.tar.gz` depending on which method you used.

For Windows, use a tool you like to create a `.zip` file. The TCC computers should have `7z` installed. For Linux, look at lab 1 for instructions on how to create a tarball or use the "Archive Manager" graphical tool.

Upload your tarball or .zip file to Canvas.