Project 10 – Epic Battle Simulator

CS 5001, Cantrell
Ryan Looney
11/29/2021

Contents

Reflection

1) Easiest and hardest parts of the assignment?

The hardest parts of the assignment:

There were several hard parts about this assignment –

As the program grew larger, I found it hard to keep track of each method, what it did, and why I created it in the first place. When there were operations like switching a weapon's type to melee, I would forget if I made it happen in a Weapon class method or a Character class method. I can see how diagrams and good documentation become paramount to writing and maintaining good code when it comes to OOP.

Design choices, though the most fun, lead to scope creep and more difficult implementation challenges. I wanted to implement fun and exciting ideas into this project, but as I started piling them on, the more difficult I realized the implementation would be, and the more time it started to add. Every design choice, such as adding a paradigm of character movement, lead to increased complexity in the code that I had not foreseen.

The easiest parts of the assignment:

The easiest thing for me was *coming up with* new ideas to implement**.** Thinking in terms of classes and objects makes coding a more natural process, in my opinion, and it really opened the door for me as far as seeing how software works like building blocks with classes and objects. Implementing the ideas, as noted above, was a more difficult task.

2) What did you learn?

In my opinion this project was the most fun and most educational project of any of the ones we have done. I learned very much about object-oriented programming as I sought to implement my own ideas to make the battle simulator more robust. I found that object-oriented programming offers convenient paradigms to implement intuitive ideas to model real-world scenarios, but adds complexity in implementation, and forces one to consider more variables and edge cases, as well as control flow of object modification, as objects grow in complexity.

3) What grade would you give yourself?

I think I would give myself a 29/30, or a 30/30 for this project. I followed the project guidelines and implemented creative and robust extensions.

Project Deviations

- Weapon Attack: Weapon attack takes more than just self because I added battle mechanisms expanded upon below that require it to account for attributes of the weapon's user.

- Missed Attacks: Weapon attack also returns more than just 0 or the weapon's attack. If an attack misses hitting the targeted character, it returns -1 to signify such.

- Print Statements – I did not print character or weapon attributes after each attack because I thought it would clutter up output. Instead, I printed these attributes during character and weapon selection.

Extensions

4) Ranged Weapon Sub-Class with Inheritance

For this project I looked up a bit about inheritance and created a Ranged Weapon class that inherits attributes from the parent Weapon class. It adds other attributes such as ranged_strength, accuracy, and ammunition. This adds a new dynamic to the game.

There is also a ranged_attack method that is different from the melee_attack method in that the character's attack has nothing to do with their physical strength. It takes an accuracy modifier from the character user of the weapon and determines the damage of the attack only based on the weapon's strength.

Adding this class also necessitated that I distinguish ranged weapons from melee weapons with a *type* attribute.

5) Movement Paradigm

I added a movement paradigm to the game where there is a distance between characters at the start of the simulation, that reduces by character walking speed on each iteration. Characters that have melee weapons will move closer to the challenger, while characters that have ranged weapons will look to shoot at the other character.

When a ranged weapon breaks or runs out of ammo it will become a melee weapon, and the character will wield it as such, and begin moving towards the other player accordingly.

6) Additional Battle Mechanics and Necessary Attributes

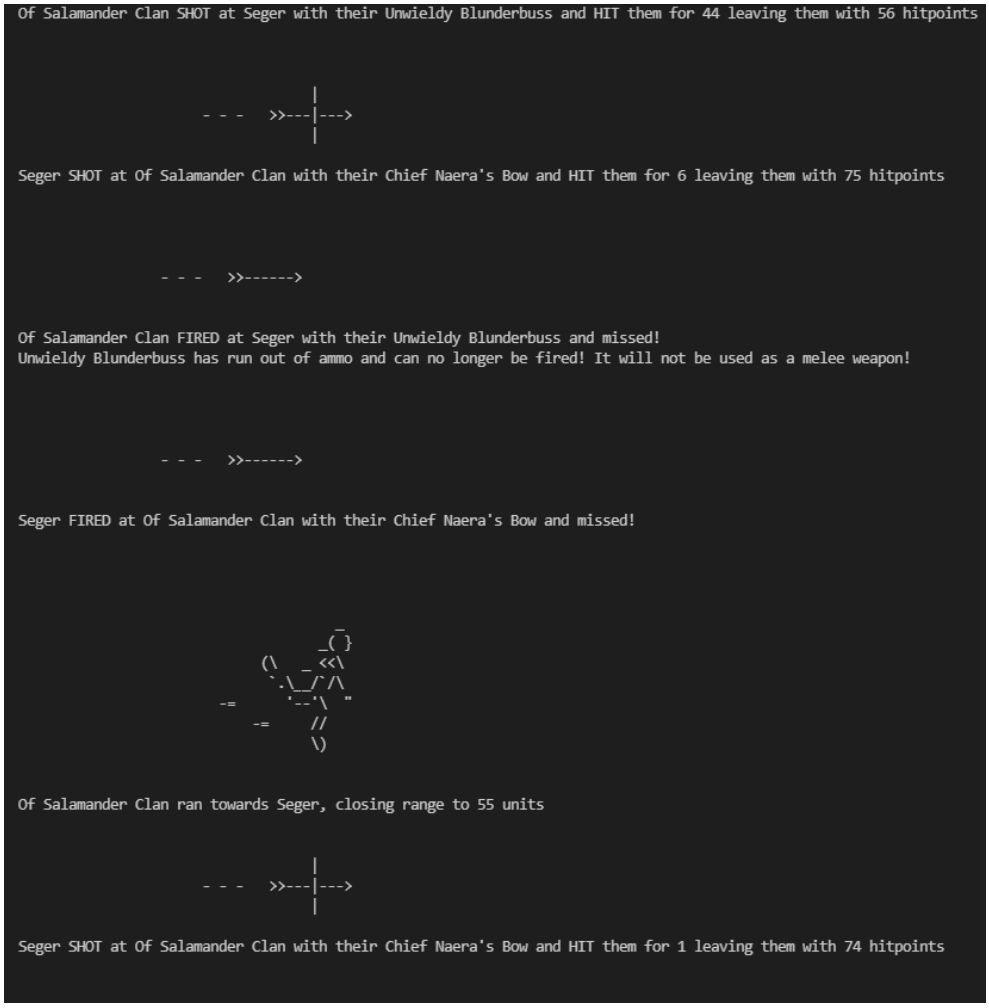
I thought it would be fun if characters had more personality such as good shooting or good melee ability. To reflect this, I made it so that characters have shooting accuracy and melee prowess attributes that determine the chance that *their attack will land* on any given attack roll.

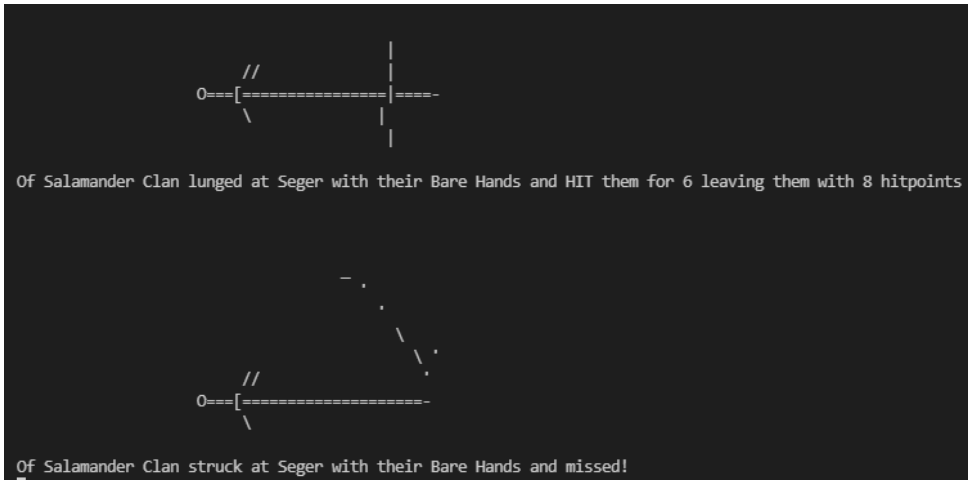
Weapons also have similar variables: Melee weapons have encumbrance, reflecting how difficult they are to handle, and ranged weapons have accuracy, reflecting their inherent difficulty to shoot (muskets are far less likely to hit than bows!).

When ranged weapons break or run out of ammo, they will be used thereafter as melee weapons.

7) ASCII Animations

Leveraging ascii art I found on the internet, the battles are slightly animated!





Acknowledgements

Source	Project Impact
Rimworld - https://store.steampowered.com/app/294100/RimWorld/	Rimworld is a game. I got inspiration from it as far as the battle mechanisms, and it is where I took the character names from.
https://www.w3schools.com/python/python_classes.asp https://www.w3schools.com/python/python_inheritance.asp	These links helped me understand classes and inheritance better so I could create my Ranged Weapon sub-classes
Asciiart.com	Found all ascii art here, none of it my work. I modified it some for my uses here.
https://www.w3schools.com/python/ref_dictionary_update.asp	Refreshed my knowledge of dictionaries as I worked.
https://stackoverflow.com/questions/2084508/clear-terminal-in-python/2084521	Showed me how to use a print command to clear the terminal.