The Arena Battle: Report

By Maya Rhoades

**Description**

The game is a text-based Arena Battle game. It relies on object orientated programming for the characters, with 6-character classes in all. Each character has 2 or 3 attack options. The player can select who to play as and who to fight, as well as the ability to play again at the end of the game. There is a hero attack option that generates a random chance to regain heath. The goal of the game is to let a player choose who to be, who to fight, and let them fight as many times as they want in an arena that battles to the death.

**Analysis**

The Arena Battle has a lot of requirements. It requires:

Classes

There are 6 different classes in the program. 3 hero classes and 3 foe classes. The classes share similar structure with an initial method that contains their name and health, though the heroes are assigned a name by the player. Then the characters need attack methods. There are different attacks, so the fighting does not get boring. Every class has a light and heavy attack. Each class does a certain amount of damage for the attack and says a different line when attacking. The bad guy classes randomly choose which attack to do. They also choose randomly between two lines when attacking.

The hero also has a sidestep/potential to heal option. This option is a way to prolong the game and give the player a better chance to win. The program randomly picks a number 1 to 2 or 1 to 3 (depending on the class) and if it matches the right number in the class, the hero heals for a custom amount of health.

Aside from attacks, the classes also need an attack action method. This attack action on a hero displays and loops a menu that shows what options the player has to attack. It then lets the player chose an attack. If they mistype or do not choose an attack it will restart the menu loop. If they choose an attack it invokes the method of the hero’s class that is the correct attack. Each attack method subtracts a custom amount of health from the bad guy. The attack method for the bad guy classes randomly chooses an attack and then invokes the method for that attack, to subtract health.

The bad guy classes also contain two methods not seen in the hero classes. These are the taunt method and health indicator method. The taunt method picks a number between 1 and 12. The classes then have 5 different taunt lines assigned to 5 custom numbers. If the random number is picked it will print that taunt line. The taunts are there to motivate the player and add some uniqueness to the program. The health indicator method is the only way the player gets an idea of the bad guy’s health. The player can see their own health but not the bad guy’s. The indicator prints three different messages saying how the bad guy looks (ex: they look really injured). The messages are displayed when the bad guy’s health is below three quarters of their health, under half their health, and under ten. Since it is a text-based game this gives the player an idea if they are winning or not.

Functions

Functions are required to clean up the code and make it easier to read. There are 13 functions in my code. 4 of those simply print a menu. The menus are in functions because I liked the look of the code when they were not there. Another 4 of the functions have to do with file handling which I will mention in the next section.

Of the remaining 5 functions, 2 of them simply announce characters. The hero announce function welcomes the now named player character to the arena. The announce function, announces who the next arena battle will be between and starts the fight. The last 3 functions are very important.

The attack first function gives the players the ability to attack first or wait for the opponent to attack, which is something that would happen in any game. There is an easter egg in the attack first function, while all my code will loop back if you mistype, the attack first menu only lets you mistype 3 times, then it will tell you that you hesitated and the bad guy attacks first.

The combat loop is probably the single most important function. It contains all the combat. In it the bad guy attacks, the bad guy taunts the player, it checks that the hero is alive, displays the hero’s health, the hero attacks, the bad guy’s health is checked, the bad guy’s health is indicated, and then it loops all over again. The function is one while loop that loops until someone has won the arena.

The last function did not need to be a function, but I love functions, so the winner function checks both the hero’s and bad guy’s health and declares the winner.

While functions were not needed in this program, they are an excellent way to clean up the code, so by using them in combination with class methods I was able to easily see the code and what it was doing.

File Handling

File handling was not necessary for this program, but I loved the potential it had. Instead of just having a simple menu I got to have fully formed character descriptions. I have 4 functions that handle file handling. 2 of them create a simple menu for the hero and guy selection. They do this by only printing out certain lines of their selective text files. The other 2 functions will print out all but one line of the text files. They leave out this one line because it is used as a 4th print descriptions option for the simple menu and I did not want it in the descriptions themselves. File handling was a great way to get a lot of text in the program but have the program very neat.

Other requirements

Functions, Classes, and file handling were needed for this program but so were some other features. While loops were used to loop every menu in the game as well as loop the entire game so you can play as many times as you want. If/elif statements were used to determine what option the player has chosen in the menus, as well as to determine what action the bad guy picked. The try except statement was used to get rid of any errors that could be caused by putting letters into the input for the menus. Plain input was used for a press enter to continue option. This option was put into the game so the players had more control over the game and would make it easier to read.

**Design**

For a closer look at the diagrams please check the files for the project on dropbox

**Diagram

Description automatically generated**

**Diagram

Description automatically generated**

**Arena Battle Flow Chart**

**Graphical user interface, diagram

Description automatically generated**

**Testing**

This chart tests every combination of heroes, all their abilities, and if the program runs correctly if they lose and win.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Combos | Check light attack | Check heavy attack | Check heal ability | Check winning | Check losing |
| Knight vs Assassin | **Checkmark with solid fill** | **Checkmark with solid fill** | **Checkmark with solid fill** | **Checkmark with solid fill** | **Checkmark with solid fill** |
| Knight vs Fighter | **Checkmark with solid fill** | **Checkmark with solid fill** | **Checkmark with solid fill** | **Checkmark with solid fill** | **Checkmark with solid fill** |
| Knight vs Ninja | **Checkmark with solid fill** | **Checkmark with solid fill** | **Checkmark with solid fill** | **Checkmark with solid fill** | **Checkmark with solid fill** |
| Viking vs Assassin | **Checkmark with solid fill** | **Checkmark with solid fill** | **Checkmark with solid fill** | **Checkmark with solid fill** | **Checkmark with solid fill** |
| Viking vs Fighter | **Checkmark with solid fill** | **Checkmark with solid fill** | **Checkmark with solid fill** | **Checkmark with solid fill** | **Checkmark with solid fill** |
| Viking vs Ninja | **Checkmark with solid fill** | **Checkmark with solid fill** | **Checkmark with solid fill** | **Checkmark with solid fill** | **Checkmark with solid fill** |
| Samurai vs Assassin | **Checkmark with solid fill** | **Checkmark with solid fill** | **Checkmark with solid fill** | **Checkmark with solid fill** | **Checkmark with solid fill** |
| Samurai vs Fighter | **Checkmark with solid fill** | **Checkmark with solid fill** | **Checkmark with solid fill** | **Checkmark with solid fill** | **Checkmark with solid fill** |
| Samurai vs Ninja | **Checkmark with solid fill** | **Checkmark with solid fill** | **Checkmark with solid fill** | **Checkmark with solid fill** | **Checkmark with solid fill** |

This table checks every place for input in my code and checks that the programs reacts correctly to wrong inputs. I will put in a string and the wrong numbers to check it.

|  |  |  |
| --- | --- | --- |
| What input is it | Check str input | Check wrong number input |
| Welcome menu | **Checkmark with solid fill** | **Checkmark with solid fill** |
| Hero Selection menu | **Checkmark with solid fill** | **Checkmark with solid fill** |
| Bad Guy Selection Menu | **Checkmark with solid fill** | **Checkmark with solid fill** |
| Attack First | **Checkmark with solid fill** | **Checkmark with solid fill** |
| Hero Attack Menu | **Checkmark with solid fill** | **Checkmark with solid fill** |
| Play again menu | **Checkmark with solid fill** | **Checkmark with solid fill** |

This table checks anything else in my code.

|  |  |
| --- | --- |
| Thing to check | Does it work |
| Quit option in main menu | **Checkmark with solid fill** |
| Quit option in play again menu | **Checkmark with solid fill** |
| Play again option | **Checkmark with solid fill** |
| Hero descriptions option | **Checkmark with solid fill** |
| Bad guy descriptions option | **Checkmark with solid fill** |
| Attack first easter egg | **Checkmark with solid fill** |
| Go first option | **Checkmark with solid fill** |
| Wait for opponent option | **Checkmark with solid fill** |

The Tables shows that all the options in my code are performing correctly.

**Critique**

The Arena Battle program worked incredibly well. It works perfectly for everything I wanted it to do. It loops the entire game, has different characters, different attacks, and is fun to play. I originally had all the attacks and class methods outside of the classes. The only thing in the classes was the name and health. I changed it and put them inside because it cleaned up the code and made it a lot more customizable with custom attack damages, custom healing, and tons of custom attack lines and taunts. It worked well to give my game some uniqueness. I do believe my game could have some improvements. I would like more custom attack moves, to have a strategy behind the bad guy attacks instead of randomness, and maybe some sort of stamina system so you cannot just use heavies for the whole fight. Despite the lack of these abilities, I think Arena Battle is a fun game and an excellent use of my python abilities.