Project V\_2

Christopher Avalos

Fight simulator

**Introduction**

The objective of this project was to recreate the fighting portion of an RPG. This would be very similar to what would be encountered during a game of dungeons or dragons, or a very limited look at what fighting in an RPG video game would look like. In older generation RPG video games, players would take turn selecting an option for their selected character to use. These could vary from attacking to defending to even using items to heal themselves. For my project, I simplified down to just attacking, and the one attacking first would always be the player. Now this was done in order to keep the project simple, while still being able to show all the required processes for the prompt. With just the simple mechanic of attacking, the code was able to reach over 500 lines of code, with messages included. I could have implemented a SPD variable into the code that could be compared and then added to a die roll to see which one would go first. I could have also added something to the defense modifier to make it to where it could help to “heal” the player or give an added bonus to their defense when used. In conclusion, this is just a small sample of something that can be expanded into further as my knowledge in programming grows. The use of classes allows the code to look a little cleaner and separating the different aspects of the code, like dice rolls and player statistics, to be able to altered with on an individual basis without risking the rest of the code.

**Gameplay**

The hitting and wounding parts of the game occurs using 2D6, meaning two 6 sided die. A player and their opponent has a number of individual stats but for the sake of simplicity, and this project, I will only be using a few, the MAT, POW, and DEF. The health pool of a model will also come into play but will largely be left as a set standard depending on the type of opponent. MAT, short for melee attack, determines what the models base attack modifier is, POW is the power of the model, meaning how hard the model hits, and DEF is the defense of a model, meaning what the opposing model must reach, through a combination of their MAT and 2D6 roll, in order to successfully hit the model. After the model hits, their pow has to significant enough damage to hit them hard enough and dwindle their HP, which is rolled against using 2D6+POW of the attacking model. So the order of operations would go through the following steps:

1. Roll to hit using 2D6+MAT of attacker

2. Check to see if the hit is at or over the DEF of the opposing model, if it is, the hit is successful, if not, the model missed and the attack step ends and then it switches to the opposite player.

3. If the hit succeeds, roll for damage using 2D6+POW of the attacking model, minus the Health of the opposing model. If the health reaches 0, the model dies.

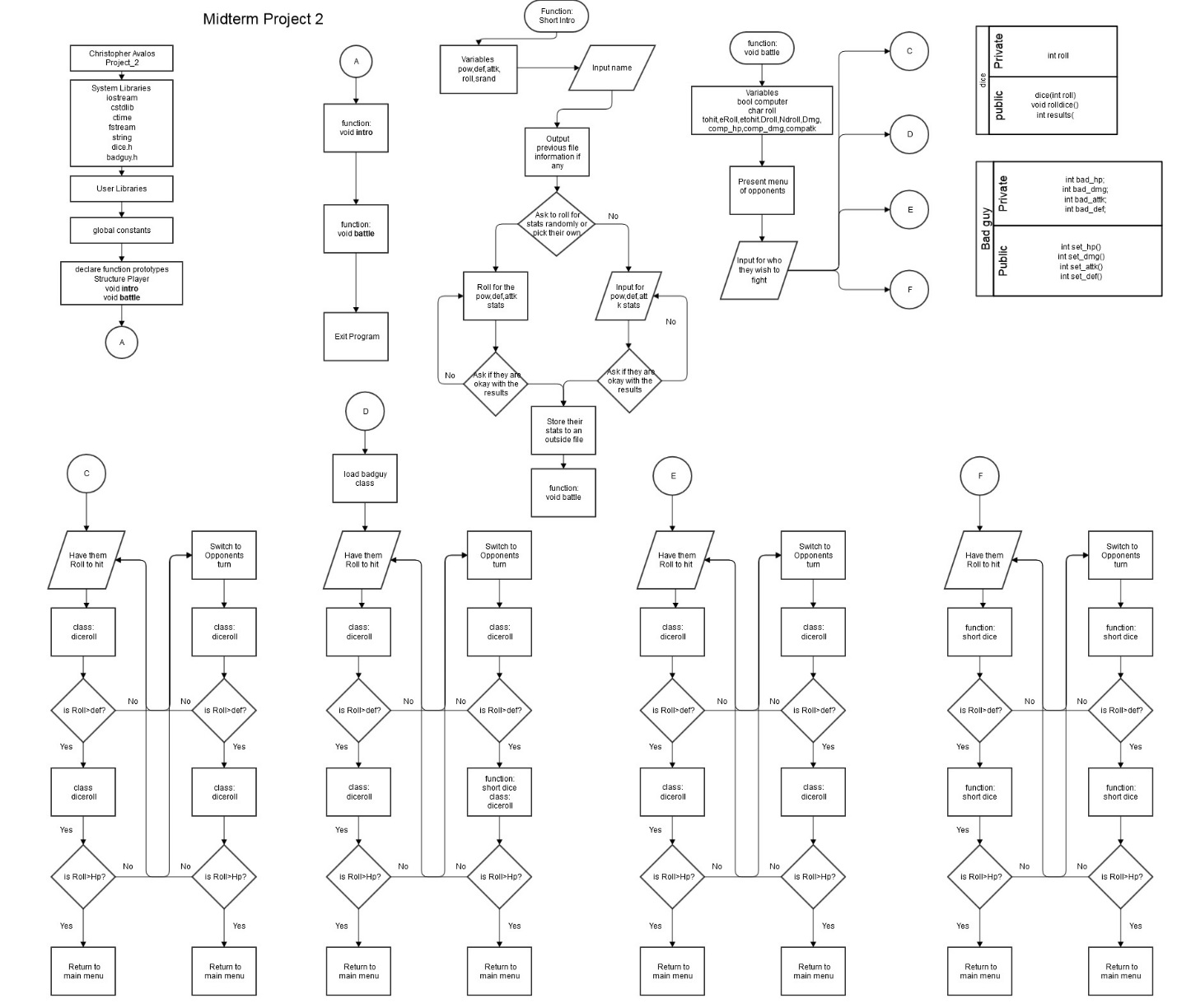
4. If the opposing model survives the attack, then it is then able to counter attack, if it does not, then the attack phase ends and the attacker wins combat.

That explains how the combat sequence normally goes, and will be used a basis for the pseudo-code.

**My version of it**

Video games and Dungeons and dragons both have many aspects to it beyond just the ability to fight. Fighting is just a small portion of the game and it can be expanded with many different variables helping to increase to decrease the standard stat lines of a player. For the project, I will keep it in the simplest terms possible with no means of affecting the core stats outside of the indented dice roll. After each fight the player will have their health reset and the players will be able to continue fighting weather or not they will or lose. There is also a stat tracker used to see how many times the player has won or loss. I will also use classes and structures to hold onto the different aspect of the game. The dice roll, player stats, and enemy stats will all be into either a structure or a class.

**FlowChart**



**Summary**

I was able to implement all the different aspect of coding that I was taught so far and come out with a fairly simple RPG type game. I could make the game much longer and implement all sorts of different types of menus, but it would be much more time consuming and the level of potential errors would increase exponentially. I hope to be able to implement function in the future in order to be able to better clean up my code. One of them being the ability for the player and AI opponent to be able to pick different options for it to be able to choose from depending on the situation. The use of classes does make the code look a little cleaner, with a lot less showing up on a single cpp file page, and being able to work on individual parts of a program without compromising the main cpp file. However, it also makes it more difficult of the finer parts of class creation are not grasped.

|  |  |
| --- | --- |
| Lines of Code | 622 |
| Structure | Line 21 |
| Char String | Line 25,61 |
| Functions using structures | Line 31.44,159 |
| Pointer | Line 40,61 |
| Writing from a file | Line 69 |
| Class instance | Dice.h |
| constructor | Dice.h, line 22 |
| Class cpp | Dice.cpp |
| Inline function | Dice.h, line23 |