Warmachine

Close combat

simulator

**Introduction**

For the project, I decided to take one of my favorite table top games and recreate one of the most exciting parts or it, the close combat phase! In Warmachine, two or more players set up models on a 4x4 field and then take turns moving attacking with their different models. The system uses a D6 model, meaning that it uses 6 sided die to determine results. There is a lot of strategy and technique to the game, but in the end the final factor in the game that determines someone will win or lose is their luck. Movement and abilities are crucial, but my skills (however little they may be) as a programmer can’t recreate the entirety of the field, I can however, recreate the hitting and wounding phase. After moving and anything else that may happen while playing, models can come face to face and attempt to take each other out. In order to try and kill each other, their individual stats are matched up and dice are rolled to see what the outcome will be.



**Gameplay**

The hitting and wounding phase occurs using 2D6, meaning two 6 sided die. A model 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:

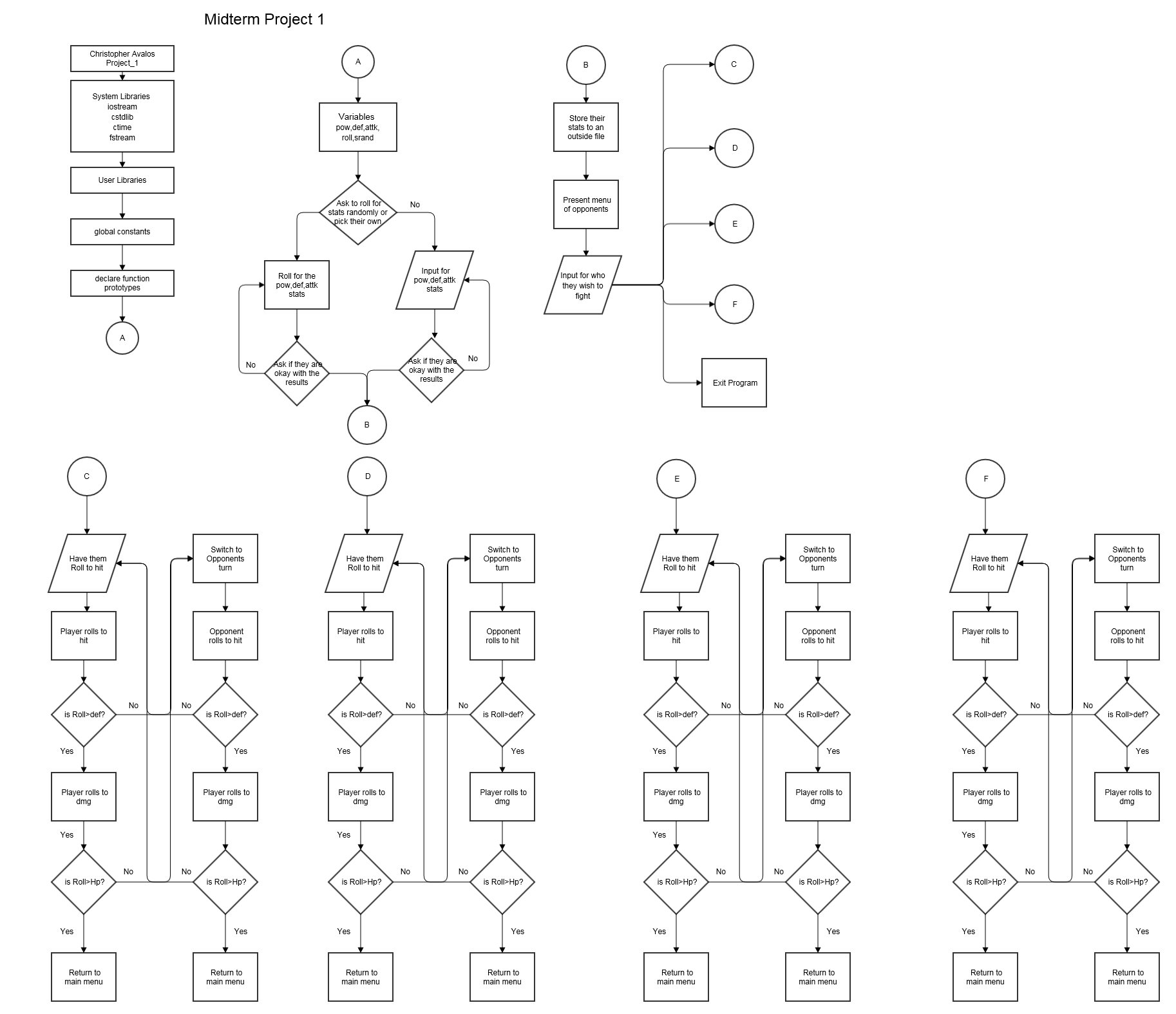
1. Roll to hit using 2D6+MAT of attacker
2. Check to see if the hit is over the DEF of the opposing model, if it is, the hit is successful, if not, the model missed and the attack step ends.
3. If the hit succeeds, roll for damage using 2D6+POW of the attacking model, minus the Health of the opposing model. If the healther 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**

For the project, recreating the different aspects of getting into close combat would be very time consuming. So for this project, I broke down to the simplist term and used a general stat generator to create the character the player would be using to fight. Finally, all states are rolled in the background and all the player has to do is press R to roll dice.

**FlowChart**



**Summary**

This was a great learning experience. 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 implent 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 implent function in the future in order to be able to better clean up my code.

|  |  |
| --- | --- |
| **Object** | **Line(s) where it’s found** |
| Lines of Code | 488 |
| Pass by Value | 20,29,118 |
| Pass By Reference | 19,28,34 |
| Returning Primitive Data types | 488 |
| Write Files | 41,110 |
| Arrays | 120,216,469 |