Brian Powers

Pablo Rivas

Software Development 1

Project 2 write up

December 13 2016

This Project was made to make the life of the DM (Dungeon Master) as easy as possible. Dungeons and dragons is such a complex game that involves so much individuality and quick thinking that the rule book is so complex and is situational making it hard for the dungeon master to decide what to deem possible for the players. Maybe I should rewind and talk about how the game is played. Basically, a group of nerds sit around a table and there are two categories of people playing; the players that play as a group of adventures and the king (or queen) nerd plays as the dungeon master. The players all have their character that they play for and work together as a team of adventures to solve puzzles, fight monsters and complete quests. The dungeon master plays for the monsters, speaks for all the people they meet and plays god for the game. The game is an open world that is created by the DM but the decisions of the players elaborates the world and everything that happens is either caused by the players or the DM because of a decision the players made. The sporadic nature of the game makes it very difficult to make any definite rules such as “When this happens do this”, the best you can do is give the dm and players things to compare their current situation to. Everything the person can do is split up into Six main categories. Strength, Dexterity, Wisdom, Charisma, Intelligence and Constitution. Strength is used for attacking with melee weapons, making athletic checks such as swimming, jumping, throwing and wrestling an enemy. Dexterity Involves using a ranged weapon, Making balance, escape artist and open lock checks. Wisdom is for casting spells, understanding languages and texts. Only a game made by true nerds would have Charisma as a fantasy power but I digress; Charisma is used for charming or intimidating a person, and it can be used to read people. Intelligence is used for having a vast knowledge in a topic that can come in handy such as architecture or alchemy. And Constitution is used to calculate the players health. All actions made by the players are decided skill checks and by fate. Each skill is associated with an ability modifier (The higher your ability score the higher the modifier). Everything is split into Skills and you are required to make a skill check every time you want to attempt a skill, which consists of the player rolling a 20-sided dice. A one is a critical failure, resulting in what you are doing to go horribly wrong. A 20 is a critical success which results in the player having the ability to bend reality and make their character do truly amazing things. Attacks are also made with a 20 sided die that is competing with a players enemy’s AC (Armor class), which is a definitive number that their roll has to beat.

But enough about how the game is played lets talk about the programs. Starting with the Dice roll generator. In the players hand book or monster manual the amount of dice is written in a certain syntax. For instance if a monsters claw does three six sided dice of damage it would be written 3 D 6. At times rolls are described in that syntax but have a constant that is added to it and it is written 3 D 6 + constant. The user is first asked to give the syntax of how many of what type of dice and then is asked from another system.out, plus how much. This will be so helpful to a DM, think about when an eleven headed hydra does a full attack with all its heads and having to roll for each of them with a single D 20.

I also made a program for skill checks. As of right now it can only gives advice for the top five skills used when I DM which are: listen, search, spot, climb and move silently. It takes the roll passes it to a method that is named after the skill type where ranges in if else statements decide what string is returned which gives the dm something to compare their current situation to. I hope to expand this program to apply to characters with a lot of bonuses to skills at higher levels. Also, I want to eventually make a method for each type of skill check there is.

The program titled Races is the start of a future character generator. As of right now it is a bullet pointed version of the racial benefit pages of the player’s handbook. While playing, you can see if a character has a bonus to a skill or has an ability to help your current situation because of their race.

Now for the big boy, everyone’s favorite part of D&D, Battle. The battle program is very complex with more loops than six flags. It involves a plethora of array lists and has taught me how to work with hashmaps. The first part of any encounter is the players rolling initiative. Initiative decides who goes first. The program begins by getting the number of players, their names, initiative and health. Then gets the number of monsters, what kind of monster the players are facing their initiative modifier and their health points. Everyone has an initiative modifier that is a plus to their roll so the program just asks for that constant and rolls a D 20 and calculates the monster’s initiatives for you. The monster manual gives each monster a number of dice to roll for the monster’s health and next to the dice syntax in parenthesis is an average that I personally regularly use as a DM. ­The program then takes all the names and respective health and puts them into a health hashmap that doesn’t need to be sorted. The rest are put into array lists of strings. It takes the player or monsters name and initiative and links them then sorts that array list in a separate class. It has a while loop that is true so the program will not end when the array is finished because the game has to be played until all of the monsters or all of the players are dead. It presents the initiative of the person who’s turn it is followed by their name. It then preforms a search of a hashmap called health and if it finds that key it displays the key and value. It then asks how much damage the pl ayer did and if they miss I would simply input zero. It shows you the hashmaps key and value after the damage has been inflicted, and says if they are alive or dead. If dead it removes it from the array list and the hashmap.