Project Proposal for Pathfinder Combat Simulator

Kenneth Devon Gaston

1. Proposal

This project will be based around creating a quick and dirty simulation of characters/creatures in the Pathfinder Tabletop RPG initiating combat. This project will target people who play this RPG, particularly the people who run sessions of these games (GMs, or Game Masters) and want to simulate how a creature/character they create would fare against characters/creatures that their players in the game would have (a quick way to test if a certain scenario is too difficult or too easy for their party, for example).
The way this program would conceptually work is that the user will provide some basic information to the program so that it may find and load the proper statistics for the character/creature. This includes the character/creatures name, level, weapon, and armor (along with hitdice for monsters; characters will have a predetermined die to roll for health depending on the class chosen)
The basic interface will be mainly buttons and menus. For the purposes of this program, only combat related factors will be observed. The user may pick creatures/characters from a preset list, but may also be able to enter the statistics of the creature the same way they would for a player character. If the character/creature has a class, the user will specify a class and level up to 20 and the program will find and load the proper statistics (Base Attack Bonus, hitdice, etc.) multiclassing will not be an option in the initial build. The user will then select equipment for their characters/creature by pressing the button that represents the equipment, or they may enter their own statistics for equipment and equip them to the character/creature.
The battle system will be based on dice rolls, a d20 for attack/save actions, and other dice as the situation warrants (1d4 if damage with a dagger is dealt, etc.); dice are simulated through a random number generator, with the minimum value possible to roll being a 1 and the maximum value varying depending on the die (8 for a d8, etc.). When the battle commences, initiative will be determined by a d20 roll; this will determine the order the characters may act, with priority being from highest to lowest (special case: a natural 20 is an automatic first place); any ties will reroll to the results to determine the order between the two tied values. When attacking, a d20 is rolled and all bonuses are added to the roll amount (roll amount + Base Attack Bonus) to determine if the attack hits. If the result is greater than or equal to the opponents Armor Class amount, the attack hits and damage is rolled (weapon die damage). Otherwise, the attack misses and the turn ends. (Special case: If the roll is a natural 20, a critical hit is threatened and another roll adding all current bonuses is made. If that roll equals the opponents Armor Class, the critical is confirmed and the damage roll is multiplied by the Critical Modifier. Otherwise, the attack still hits and deals damage as normal).
For simplicities sake, the program is only concerned about the cases where characters are in range to hit one another with weapons and will not take movement or Combat Maneuvers into account. There will also be no items usable outside of weapons and armor and no spell casting abilities implemented (at least initially). Skills will also not be used, and any abilities involving skills will not be implemented.

2. Basic Contract

The program will be done when a user can create a character/creature to fight up to 3 other creatures, giving a total of four character/creatures in combat, and use the results of the battle to determine who would win a fight between the characters/creatures.

3. Stretch Goals

3.1. Ability Scores

In most tabletop RPG systems including Pathfinder, all creatures/characters have point totals in statistics that determine certain aspects of their character. These will be created and implemented at this stage.

3.2. Implement Races

In the Pathfinder system, you choose a race for your character and each race has its own bonuses and penalties. At this stage, these will be implemented.

3.3. Implement Spells

Some classes have the ability to cast spells that can harm or heal others. Those spells will be implemented at this stage. For the purposes of this program, only spells that harm or heal in some way will be implemented.

3.4. Implement Feats

Feats are abilities that are gained every odd level that improve the character in some way. For the purposes of this program, only non-skill related combat feats will be implemented at this stage.

3.5. Implement Multiclassing

In Pathfinder, it is possible for characters to take levels in multiple classes, and at this stage, that capability will be implemented.

3.6. Implement Movement

Pathfinder often works under the assumption that a board of tiles is being used to show how characters move, and at this stage, all mechanics involving movement, including Combat Maneuvers, encumbrance, and flanking, will be implemented, along with a visual grid to represent all this.

3.7. Implement Additional Classes

There are many other classes that exist outside of the Core Rulebook for in Pathfinder, and at this stage more will be implemented. Classes from the Advanced Players Guide will be the primary focus at this stage, though others may also be added.

3.8. Proposed Project Timeline

1) Week 1: Create classes holding basic information (statistics, dice, etc.).
2) Week 2: Create test Character Class, Creature, and Equipment, begin implementing battle system.
3) Week 3: Battle System.
4) Week 4: Battle System, Initiative Queue.
5) Week 5: Battle System, Initiative Queue.
6) Week 6: Implement Equipment.
7) Week 7: Implement Equipment.
8) Week 8: Implement other classes.
9) Week 9: Implement other classes.
10) Week 10: Implement other creatures.
11) Week 11: Implement other creatures.

4. Use Cases

4.1. User Selects Premade Fighter Character

1) User clicks Create Character button.
2) User clicks Quick Create button.
3) User clicks Create Fighter button.
4) Character name, weapon, armor, and level generated (name is Fighter, weapon is Longsword, Armor is HeavyArmor, and level is 1).
5) Character stored in a list for later use.

4.2. Use Selects Premade Living Dead Monster

1) User clicks Create Characters button.
2) User clicks Quick Create button.
3) User clicks Create Living Dead button.
4) Monster name, BaseAttackBonus, hitdie, weapon, and level generated(name is Living Dead, BaseAttackBonus is 3, hit die is a D(4), weapon is Claws, and level is 3).
5) Monster stored in a list for later use.

4.3. User Initiates Battle; ¡ 2 Characters/Monsters in List

1) User clicks Initiate Battle button.
2) Error message prints Sorry, you must have at least 2 characters/monsters selected to initiate a battle.

4.4. User Initiates Battle; 2 Characters/Monsters in List

1) User clicks Initiate Battle button.
2) Initiative will be rolled and the list will be sorted from highest to lowest initiative amount rolled (Highest roll, goes first, lowest goes last; reverts to first character/creatures turn after last character/creatures turn).
3) TextBlock informing user which character/creature the character/creature whose turn it is will attack (note: can never select self).
4) As long as Hit Points of both characters/monsters != 0, the user will select who the character/monster attacks by clicking the Select button until the target the character/creature they wish the character/creature whose turn it is to attack, and then click the Attack button. If the rolled attack value ¿= to the targets armor class, the attack hits, damage is calculated, and targeted creature/characters Hit Points are reduced by that much; otherwise, it misses.
5) When a character/monster has 0 Hit Points or less, it is declared dead and is removed from the list.
6) When only 1 character/monster remains, it is declared the winner and the battle ends.

5. Functional Requirements

1) Add characters/creatures for combat.
2) Equip characters/creatures with weapons/armor.
3) Allow user to generate stats for character/creature.
4) Generate pre-made creatures for user on request.
5) Randomly roll stats of character/creature for user on request.
6) Delete characters/creatures from combat.
7) Allow user to add a character class to the character/creature.
8) Allow user to replace equipment for a character/creature as needed.
9) Display instructions in a text block for user (Please select Equipment by pressing button, etc.).
10) Make characters/creatures fight when requested.
11) Determines if a creature/character is successful in attacking another character/creature and then reduce the attacked characters/creatures Hit Points by a determined amount of damage if the attack is successful.
12) Removes character/creature from combat when defeated (Hit Points ¡= 0).
13) Order when characters/creatures act, or take their turn (i.e., access functions to attack one another), by the value of their Initiative rolls (ordered from highest value to lowest).
14) Reorder when characters/creatures take their turn when a creature/character dies.
15) End combat when only 1 character/creature remains and declare it the winner.

6. Non-Functional Requirements

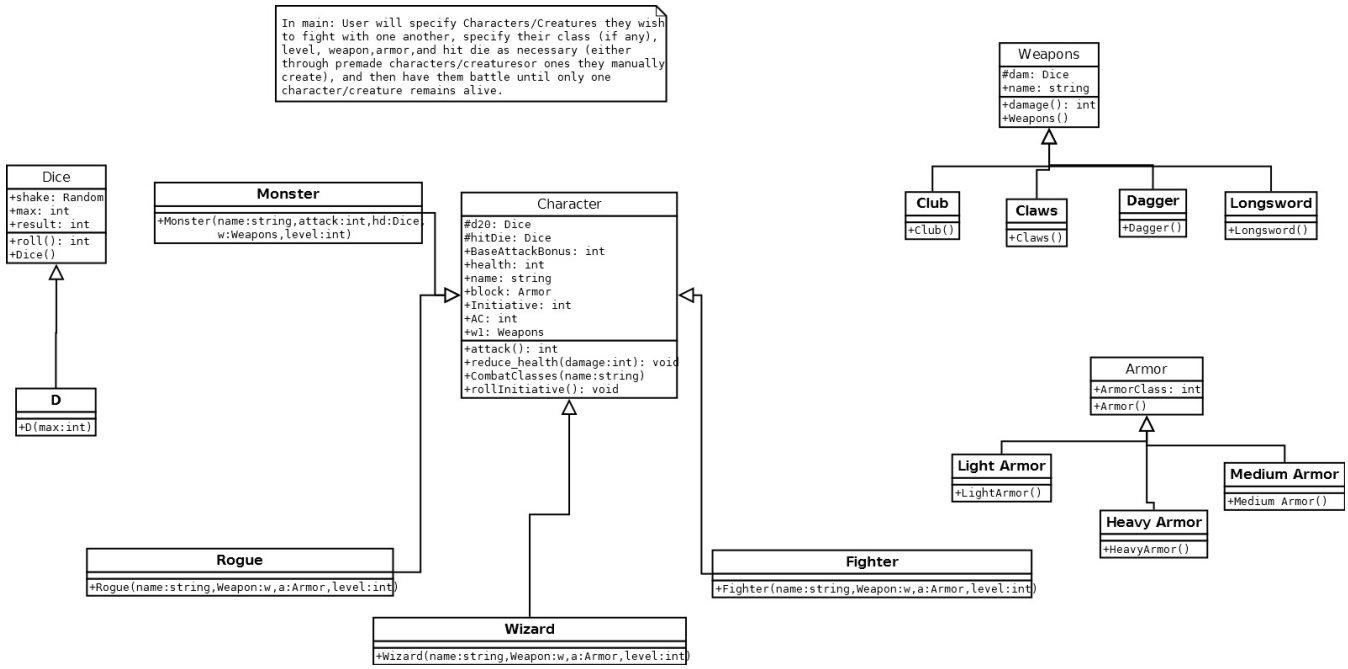
1) Must be able to contain at least 2 of characters/creatures in list during battle.
2) Data of characters/creatures before and during battle must be readable by user.
3) Results of accessing functions that modify the state of the characters/creatures (i.e., pressing buttons to give them equipment and typing in their stats) must be made clear to user via information printed out in a text block.
4) Program as a whole should always be accessible to user without fail.

7. UML

The UML for the project is depicted here1

8. Design Patterns

1) Abstract Factory: Creation of Dice objects
2) Template Method: All characters/creatures, weapons, and armor share same basic variables and functions that all do the same thing in each respective object, but they have differing information inside each respective constructor function



Combat.jpg

Figure 1. The UML for the proposed Project