Project Proposal - Dungeons and Dragons Character Creator

Rachael Devlin (rdevlin), David Ricardo (kigaltan), and Rachel Thornton (rthorn)  
January 16, 2015

1 Overview

We plan to code a web app to generate D&D characters given certain user-specified parameters. At the end of the project, our program will be able to:

• Allow user to choose a class and race  
• Provide a UI to guide the user through character creation  
• Calculate character's game statistics  
• Display full statistics needed for character sheet

2 Game Components

We will implement the Dungeons and Dragons 5th Edition rules for character creation as published in the Player's Handbook (http://dnd.wizards.com/products/tabletop-games/rpg-products/rpg\_playershandbook).  
Our program will store character's data in the main Character class, with other classes used to organize more complex properties such as Character Class, Race, Background, and Weapon, all of which have properties of their own to store. (NOTE: Character Class (called Char\_Class in the program) is seperate from the class Character. Classes in D&D are Wizard, Barbarian, Rogue, Fighter, etc., named Char\_Class in order to avoid confusion with classes in Python.)

2.1 Character class  
Each instance of the Character class will store all of the information for that character. For variables, it will have:

* level, an int (defaults to 1; possibly add higher level functionality)
* proficiencies, a dictionary storing lists of proficiencies listed by type (weapon, armor, etc.)
* saves and skills, dictionaries storing modifiers of ability saves and skills
* languages, a list storing languages the character knows (Elven, Dwarvish, etc.)
* ability\_scores, a dictionary storing ability scores under their description (Strength, Agility, etc.)
* initiative, an int

The Character class will also have instances of Char\_Class(), Race(), and Background(). These methods will contain similar variables related to their effect on the character, which will then be used to calculate the above values. Char\_Class() and Race() will also have subclasses Archetype() and Subrace(), respectively, to handle different variations of the classes/races.   
Significant methods in Character include:

* proficiency\_bonus(level): returns proficiency bonus based on character level
* armor\_class(): returns armor class based on class and proficiencies
* calculate\_skill(skills[key]): returns calculated skill modifier for a given skill
* calculate\_speed(): returns calculated speed based on class and race

2.2 Databases  
Databases of relevant information from the handbook will be created and linked to the python files.

2.3 GUI

We will implement a GUI using Django. Players will be able to give their character's name, race, class, and ability scores. Further customization of the character may be added depending on time constraints.

The mainLoop() method will collect this information from the user, create a new Character object with it, and then display the complete statistics in a simple text format. A more complex visual format for the final result may be considered if time allows.

2.4 Calculations

The program will do all mathematical calculations for the user. Ability, saving throw, and skill modifiers will be calculated based on the given parameters, as will armor class, hit points, etc. Choices not made by the user will be made by the program.

3 UI

Players will be able to type a name for their character in a text box with a maximum character length. Race and class will be chosen via drop down menus, as there are limited options, and brief descriptions may be displayed to benefit new players. Choosing ability scores (6 values ranging from 8 to 20) will be done using an interface with sliders that determines whether or not given combinations of scores are valid. Every time the user makes a change on the page, a function will run, checking if all parameters have been set. If so, the "Done" button at the bottom of the screen will become enabled. Otherwise, it will be disabled and appear grayed out. User will then submit their final choices by clicking the enabled button. Calculations will then ensue, and the final results will be displayed on screen, possibly in a downloadable file.

4 Work distribution

David will be in charge of the UI, as he has the most experience in web UIs. Rachael will be in charge of organizing and incorporating into Django the databases of information needed from the handbook. Rachel will focus on structuring and coding the Character class and related classes and methods. Though we all have specific areas we are in charge of, we will be involved in all areas and not limited to certain tasks.

5 Checkpoint

By our meeting on Tuesday, January 20th or Wednesday, January 21st, the back-end organization of the program will be functional and complete. Databases will be structured and connected through Django to the code, though the databases may not be completed filled with information yet as that is a tedious and time-consuming process. The Character class and all related classes and functions will be complete, though alterations may be made later to improve or add to it. We will have done work on the UI but do not expect it to be complete. We will then focus on typing up data, improving UI, and adding features for the remainder of the week.

Minimum functionality for the program is that the user inputs race, class, and ability scores, and the program determines all statistics based on those and random functions and displays these statistics in plain text.