COS 125 Design Document

Design Document
FIVE DICE, THREE ROLLS
Spencer Ward
October 7, 2016

SECTION 1. INTRODUCTION

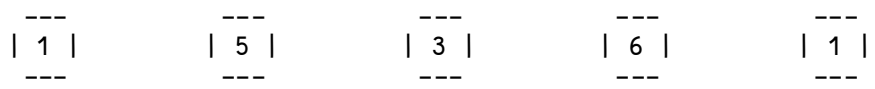
Five Dice, Three Rolls is a command line version of the classic dice game, Yahtzee. The game will always display the dice and the scoresheet, and the users will interact with the game through text prompts. The game will support any number of players, but is best played in a small group. The target audience will be elementary school students and up, as long as they know how to type in the command line and can understand the rules of Yahtzee. The game will be meant for entertainment, mostly in small groups.

SECTION 2. OVERVIEW OF THE GAME

The game is played in a series of thirteen rounds, with the players choosing one of the thirteen scoring types for each round, without repetition. Every round, each player rolls five dice, then selects as many dice as they want to reroll, and then rerolls again. Their score in the scoring type they chose is recorded in a score table. At the end of the thirteen rounds, the sub scores are summed up, and the player with the highest total wins the game. Scores and dice are displayed in text. The game does not support saving, high scores, or sound.

SECTION 3. THE USER INTERFACE

When first starting the game, it will ask the user how many players they would like, and whether or not they want to use any modifiers, such as a different number of dice, rerolls, or scoring types. Once started, game will output a score grid, with the columns labelled by the scoring types, and the rows labelled by the player numbers. The grid will be drawn to show as many players as are needed, although the game will need to be scrolled through if the player count is too high. Below this score grid, there will be a set of simple ASCII dice, spread across the screen to make them easier to read.



The game will be driven by a user input field as the bottom of the screen, with prompts for player switching and actions during every round. When rerolling dice, players will be able to input a list in a simple syntax, e.g. "D1, D3, D4" that the game will use as the reroll selection. Every time the game screen needs to be updated, the game will clear the console and redraw the grid and dice.

SECTION 4. ARCHITECTURE OF THE GAME

As the game is intended for local play only, there will be no need for saving high scores or using online resources. The game will need to keep track of a list of players and, for each of those players, a list of scores and their current totals.

SECTION 5. SCOPE OF EFFORT

UI design for the project will require a control function to call functions for drawing the score grid, the dice, and the user input prompt. The user input function will handle most of the function calls for the program, as the display and background calculations only need to be updated on user input. The game will also need functions for each of the thirteen different score types. The program will also need a function to clear the screen and redraw the output every time it needs to be updated.

SECTION 6. IMPLEMENTATION PLAN AND TIMELINE

The game will be designed with each version adding new features to the game.

The first version will be focused on implementing the UI control functions and the functions for drawing the score grid and the dice. This version should take about an hour to write.

The second version will be focused on adding the user input functions, to handle redrawing the UI and prompting the user for input. Writing the prompts and basic text parsing should take under an hour.

The third version will add the game rules, including dice selection and rolling, along with rounds and scores. Adding dice controls should take around half an hour, and rounds and scoring should take another half hour.

The final version will include score handling, including all of the functions to calculate scores, adding them to the score grid, and letting the game choose a winner. Depending on how long the score types take to implement, this version will take around one to two hours.

Working for an hour a day, versions 4 should take two days, and versions 1, 2, and 3 should take one day each, meaning that the project should be finishable in six days, leaving an extra day for overflow and testing.

SECTION 7. RESOURCES USED

The game will use the following:

Basic, original ASCII art

Lib/random.py: Built in library for generating pseudo-random numbers, to be used for handling dice rolls.

Lib/os.py: Built in library for using proprietary operating system services, to be used for executing the command to clear the console.

SECTION 8. CONCLUDING REMARKS

The game should be simple to play and easy to enjoy, as long as the players know how to use the different scoring types. The final version of the game might include a reference to the Hasbro rules sheet, for players who have never played the game before.

SECTION 9. BIBLIOGRAPHY

https://en.wikipedia.org/wiki/Yahtzee: Easy reference for Yahtzee rules.

https://docs.python.org/2/library/random.html: Documentation for built in random module.

http://stackoverflow.com/questions/517970/how-to-clear-python-interpreter-console: Python code for clearing the console.

SECTION 10. WORK ASSISTANCE STATEMENT

https://docs.python.org/2/library/random.html: Documentation for built in random module.