**Duell Python Manual**

**Bug Report:**

No bugs to be reported.

**Feature Report:**

**Features Not Implemented:**

All features of the game are implemented.

**Additional features include:**

None.

**Reason for Choosing Python:**

Python is being regarded as one of the most frequently used programming languages in the programming world due to its user friendly syntax and powerful capabilities. The reason why I wanted to do the extra credit project in Python is to gain a familiarity with the language. It's important to be knowledgeable about a variety of programming languages, and before completing this project I have never worked in Python. Choosing Python allowed me to familiarize myself with the language and made me want to do more projects with Python.

**Experience Carrying Out the Project:**

Completing the Duell game in Python was overall a good experience, and I feel like it helped to teach me a lot about the language. Going from languages such as Java and C++ made Python feel a bit weird with its typing, but after working with it for a few hours it was easy to get used to. Knowing C++ before attempting to do a Python project definitely helped with the learning curve of working with a new language. Python's scoping and indentation rules were hard to get used to as well as how only some data types are changeable when passed to a function. Overall, I enjoyed working with Python and would like to work with it again.

**Description of Data Structures/Classes:**

**Classes:**

Board:

The board class has the dictionaries, diceHuman, and diceComputer, and duellBoard. These maps are used in order to keep track of where board pieces are located and to whom they belong to. The board class handles moving dice and keeping track of the die and open spaces.

BoardDisplay:

The boardDisplay class handles displaying the game board. It goes through the duellBoard dictionary in order to see whether to print a die or a space. If there is a die to be printed, it checks whether it is a computer or human and prints out the top and right of the die accordingly. The only purpose of the boardDisplay class is for printing.

Computer:

The computer class inherits from the player class and implements the computer player strategy and validates that a move is legal. The computer strategy is to attempt to win the game first by capturing the 1x1 die or the special tile, then the computer will attempt to defend its own 1x1 die if it can be captured, the computer will then attempt to defend any other die that may be captured, afterwards the computer will attempt to capture any enemy die, and if none of the previous moves are possible the computer will just attempt to move to a possible open space. The move is decided in the function decide\_move() which checks through all possible movements for the best possible move. The computer class validates that a move is legal before making the move on the board.

Game:

The game class handles each round in a game and which player, the human or computer, is currently playing. The game class also handles deciding which player will play first in a round. It also handles the serialization of a current game by saving a game and reading in a from a text file.

Human:  
The human class inherits from the player class and implements the human player help strategy when the human player asks for help making a move and validates that a move is legal. The human strategy is to attempt to win the game first by capturing the 1x1 die or the special tile, then the human should attempt to defend its own 1x1 die if it can be captured, it will then attempt to defend any other die that may be captured, afterwards the human will attempt to capture any enemy die, and if none of the previous moves are possible the human will just attempt to move to a possible open space. Which move to be suggested is decided in the function decide\_move(). The human class validates that a move is legal before suggesting the human should make a move.

Player:

The player class handles the player names. It is used as a base class for the human and computer class.

Tournament:

The tournament class keeps track of the rounds won by each player and also determines who won the match. It is used to update the number of wins a computer or human player has, either by winning a round or reading in the wins from a serialized file.

Dice:

The Dice class is used in order to keep track of the top and right of the dice, and is the sole purpose of the dice class.

**Dictionary:**

diceComputer:

The diceComputer dictionary is used to store the computer dice, which are mapped to the row and column coordinates as a tile. Each tile is a row and column which makes it a unique key.

diceHuman:

The diceHuman dictionary is used to store the human dice, and are mapped to row and column coordinates. Each tile is a row and column which makes it a unique key.

duellBoard:

The duellBoard Hashtable is used to store what type of piece is at each tile for the Duell game. The piece type can either be an "H" for human, "C" for computer, or "0" for an empty space.

**Log:**  **Total: 14 hours**

**December 3, 2016**

**-**Created player, human, computer, dice, tournament, game, board, boardDisplay, and main python files.

-Added get\_name() and set\_name() to the player class.

-Added decide\_first\_player(), get\_current\_player(), and set\_current\_player() to the game class.

(1 hour)

Total: 1 hour

**December 4, 2016**

-Added print\_board() to the boardDisplay class.

-Added get\_top(), get\_right(), set\_top(), set\_right() to the dice class.

-Added initialize\_human\_dice(), initialize\_defualt\_board(), initialize\_computer\_dice(), get\_human\_top(), get\_human\_right(), get\_computer\_top(), get\_human\_top(), and get\_tile\_type() to the board class.

(2 hour)

Total: 2 hour

**December 5, 2016**

- Added check\_collision(), movement\_direction(), and is\_valid\_movement() to the human class.

(1 hour)

Total: 1 hour

**December 6, 2016**

**-**Added calclulate\_frontal(), calculate\_lateral(), move\_frontal\_human(), move\_backwards\_human(), move\_left\_human(), and move\_right\_human() to the board class.

(1 hour 30 minutes)

-Added move\_dice\_human() to the board class.

(30 minutes)

Total: 2 hours

**December 7, 2016**

-Added check\_collison(), movement\_direction(), and is\_valid\_movement() to the computer class.

(1 hour)

Total: 1 hour

**December 8, 2016**

-Added move\_dice\_computer(), move\_frontal\_computer(), move\_backwards\_computer(), move\_left\_computer(), move\_right\_computer() to the board class.

(1 hour)

Total: 1 hour

**December 9, 2016**

**-**Added get\_computer\_special\_row(), get\_computer\_special\_column(), get\_human\_special\_column(), and get\_human\_special\_row() to the board class

-Added move\_to\_open\_space(), kill\_a\_human\_dice(), defend\_computer\_dice(), defend\_special\_die(), move\_to\_special\_enemey\_tile(), move\_to\_special\_enemy\_die() to the computer class.

(2 hours)

Total: 2 hours

**December 10, 2016**

**-**Added move\_to\_special\_enemy\_die(), move\_to\_special\_enemy\_tile(), defend\_special\_die(), defend\_human\_dice(), kill\_a\_computer\_dice(), move\_to\_open\_space(), get\_computer\_path\_frontal(), get\_computer\_path\_lateral() to the human class.

(2 hours)

Total: 2 hours

**December 11, 2016**

-Added game\_serialization\_write() and game\_serialization\_read() to the game class.

(2 hours)

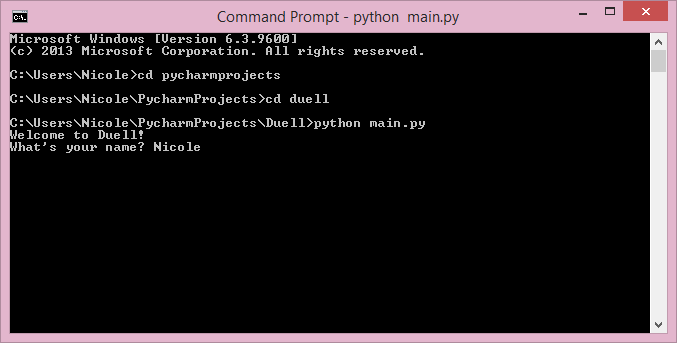
Total: 2 hours

**How to Run the Program:**

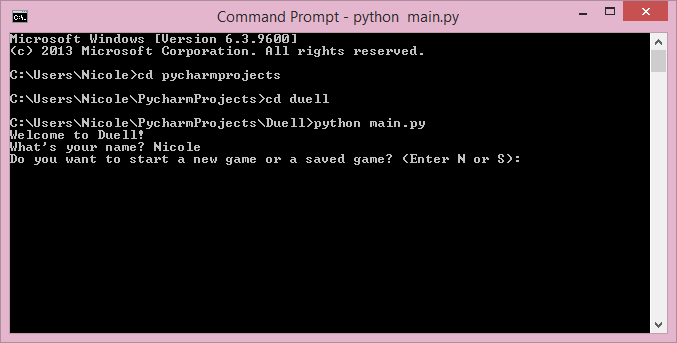
The program runs using version 3.5.2 of Python. The program is run from the command line and is run from the file main.py.

**Screen Shots:**

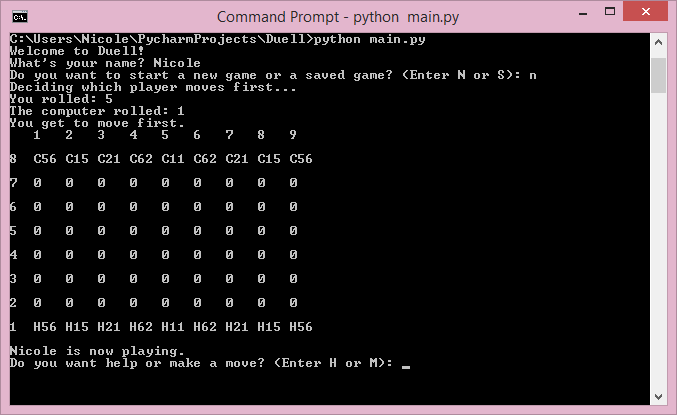
1. Entering a human name.



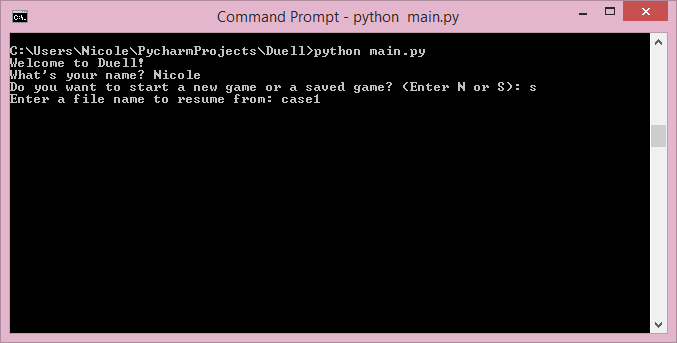
1. Asking to start from a new or saved game.



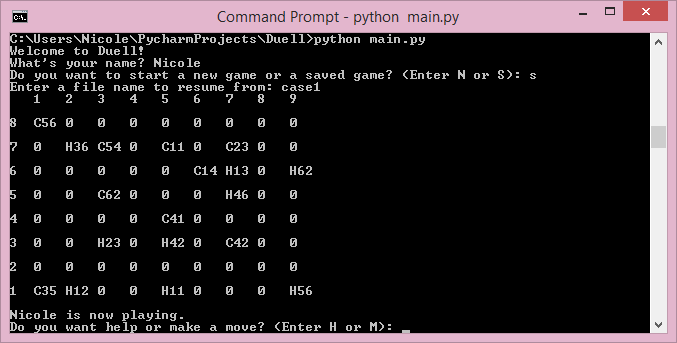
1. A new game board.



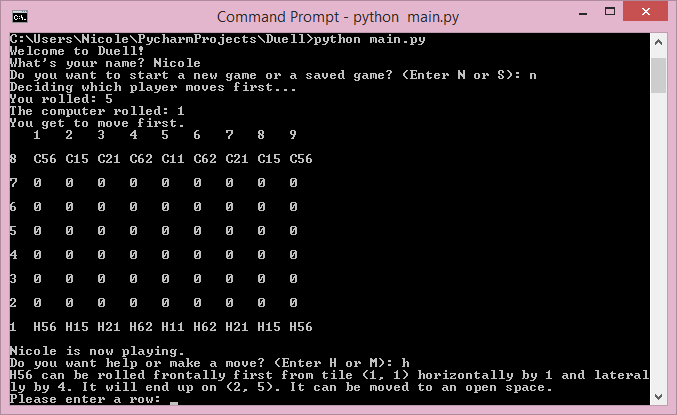
1. Asking for a file name to read from.



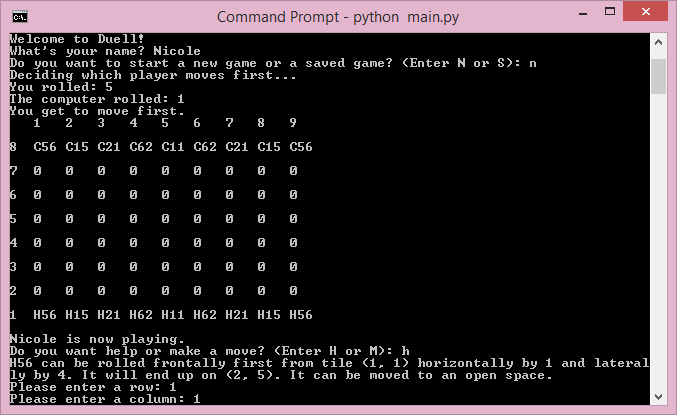
1. Board after being loaded from a saved file.



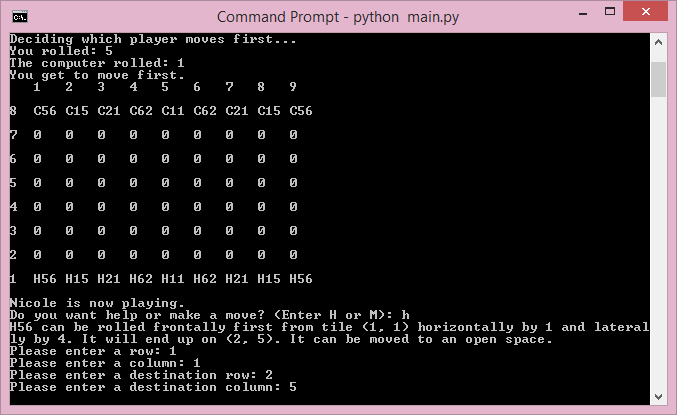
1. The human player asking for help.



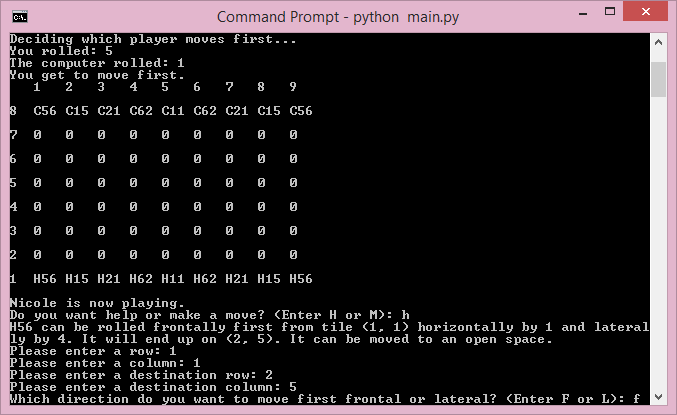
1. Entering the row and column of the dice to move.



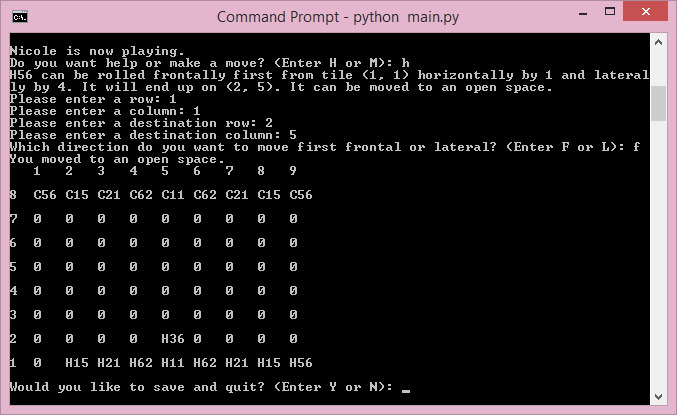
1. Entering row and column of the tile to move to.



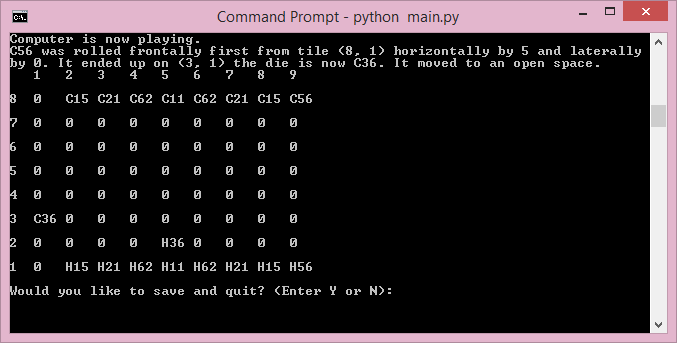
1. Entering whether to move frontally or laterally first.



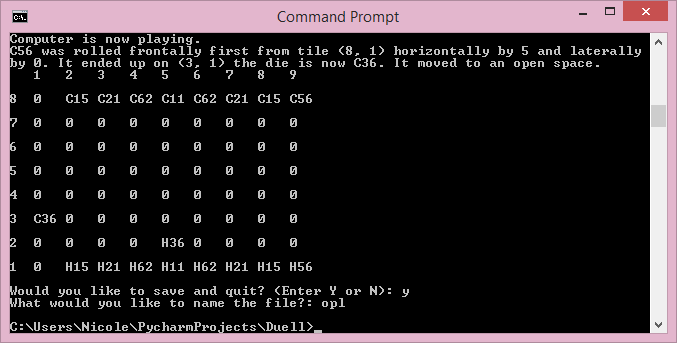
1. A complete human move.



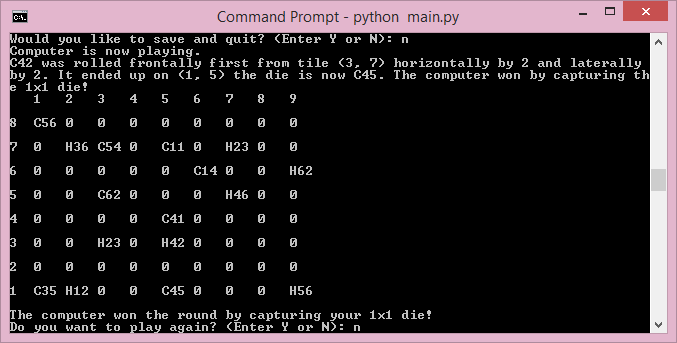
1. The computer player making a move.



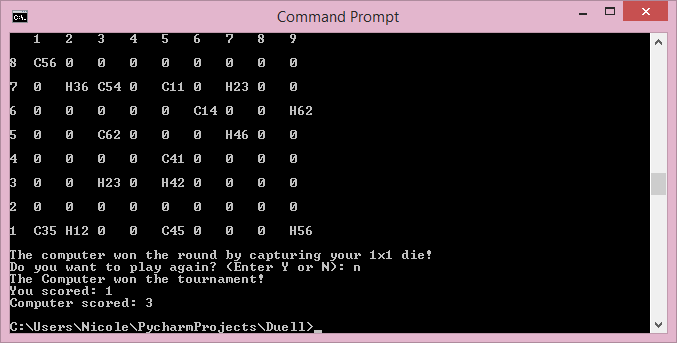
1. Human being asked to save and entering a name to save to.



1. Human being asked to play another round.



1. The end of the tournament.



1. Picture of full board with all elements.

