STATUS REPORT

Changes to original design:

1. Significant change were made to our original thoughts on how to structure the gui. We are now using tkinter, which has much more robust tutorials online, as well as being simpler to use and implement the functioning scrabble game.
2. We updated our thoughts on the algorithms:
    a. Hyper-parameter tuning over the many parameters we expect to pop up – minimax tree depth/width, classification strictness, etc. We hope to get a sense using either gradient descent or evolutionary algorithms of which work best
    b. Simulated games – this was mentioned in our project proposal, but since the feedback mentioned it as well we confirmed we wat to run simulated minimaxing over potential tilesets
3. Scope changes:
    a. For simplicity, the game will be restricted to one human player vs one ai player, with potentially a hidden backend with ai vs ai for testing

Updated Timetable:

As of April 25th, 2017:

- Significant work on gui
- Significant work on back-end implementation

April 30th:

-Complete implementation of scrabble, back-end and GUI

May 7th:

-Simple brute-force scrabble AI completed, game can be played fully human vs. AI

-Start project report, keep up to date as project completes

May 10th:

-First iteration of intelligent AI done, including minimax tree with alpha beta pruning.

May 13th:

-Hyper-parameter tuning, cleanup up structural algorithms, testing against self, human amateurs, and possible human “pros”.

May 15th:

-Fine tune report, hand in

ORIGINAL PROPOSAL

CS 4701 Project Proposal

Daniel Stoyell (dms524) and Anil Vadali (arv42)

3/22/17

Scrabble AI

For our project we would like to build an AI for Scrabble. For this AI, we'd like to explore using elements of both machine learning and artificial intelligence. The main algorithm that underlies our AI would be adversarial search, specifically a heavily limited and modified version of minimax. From that, we have several ideas on how to improve performance. Generally, we believe that a simple brute force approach to finding all valid moves is computationally short enough to be feasible. Once we have the set of all valid moves and their point value, we can get around the imperfect information of scrabble by "simulating" several dozen randomly generated opponents with different tile sets, and minimax that downward to whatever depth we'd like, and aggregate the effectiveness of a given move across all generated opponents to get an idea of how good that move is. In the endgame, however, Scrabble is a game of perfect information, and thus for the last several moves of the game we plan to use a direct minimax with alpha-beta pruning to find the optimal endgame play.

To evaluate this algorithm, we will be using 3 different approaches. First, as the bot improves we will store older versions of the bot, and benchmark progress by playing the bot against older versions of itself (and current versions) to determine if it is improving. We will also potentially be using evolutionary algorithms to fine tune parameters within the bot, by playing against itself. Second, we will be playing the bot against human opponents. We hope to have a bot that is able to consistently beat amateur

human players, but hope to increase the margin over time. We might also contact the Cornell Scrabble Team for advice on bot strategy, and also for better opponents. Third, there is a existing scrabble AI called Maven that is by far the most popular and widely used AI in electronic releases of the game. We hope to play our bot against this AI, and benchmark progress by the average margin of loss (and perhaps victory).

We will be developing a graphics interface that will display the board, tiles, and scores. We will be basing this interface primarily on the interface used for the "Breakout" game that was implemented in CS 1110. We will be modifying this "Breakout" interface appropriately to create the features of Scrabble. We are using the prebuilt graphics library just to minimize the work needed to create the Scrabble GUI so we can focus on AI. The primary language we will be using in this project is Python and we will further be using the Kivy library of Python to create this graphics interface. Other approaches that we could pursue if this approach fails is using additional Python libraries such a tKinter, PyGTK, and PyGame to build the features from scratch. For word formulation, the AI would most likely be using an Anagram Solver that takes in its current tiles and the "open" tiles on the board, and produces all the words possible.

Timeline:

March 24th: Complete project proposal

- Map out rough timeline of completion
- Begin planning AI implementation
- Lock down details of Scrabble GUI

April 8th (End of spring break):

- Complete Scrabble GUI

- Have move finder completed, with a simple “Best Move” AI to play against

April 21st: Status Report

- Complete revised project proposal
- Have significant work completed on AI (Minimaxing, move evaluation, etc)

May 7th:

- Complete AI, including revisions and algorithms
- Complete testing against self, humans, and Maven

May 15th:

- Complete group report
- Complete individual reports