Erik Chan, ejc233
Gary Gao, gg392
CS 4701
3-24-17

AI Practicum Project Proposal: Poker

Description

We will be building a game playing program for the card game Poker. Among the core functionalities of the system will be the distribution of playing cards, ability to bet a certain amount of currency, opportunity to either call or fold, and the determination of the highest poker card ranking. We will build classes for the game itself, each poker hand, each card, each ranking, each deck, a human player, and a set of A.I. players with different levels of intelligence and varied strategies.

Our AI will be smart about its plays. It will look at the hand it has and determine a value based on its starting hand. It looks at how many of the other players have folded and how much money it has relative to the other players. The AI should also look at the size of the bets the other players have betted and the possible cards that are available based on the starting sequence of cards on the table.

If our AI knows it has the highest tier of a winning hand and if the other players display a level of uncertainty in their play (low bet/raise size, low number of players who have not folded) and it has a confident level of knowledge that it will gain money/points then our AI should try to get the most money it can in that situation.

We will be exploring mainly the area of gameplay. We will implement minmax trees, search spaces for our operators, search algorithms (informed, uninformed, adversarial). For our trees, it will be large and more complicated than the normal minmax trees with 2 players. The trees would be based off the number of remaining players in the game/round and the number of revealed cards. There are a lot of possible states because of the nature of poker as a random number generator game. The AI would know its own hand and the revealed cards so it can do some type of inference on the other players. Our search space would include a lot of possible states and the operators to raise the bet, fold, etc. We can use search algorithms for our AI to make better moves (i.e. we can have heuristics that can help our AI decide a better move).

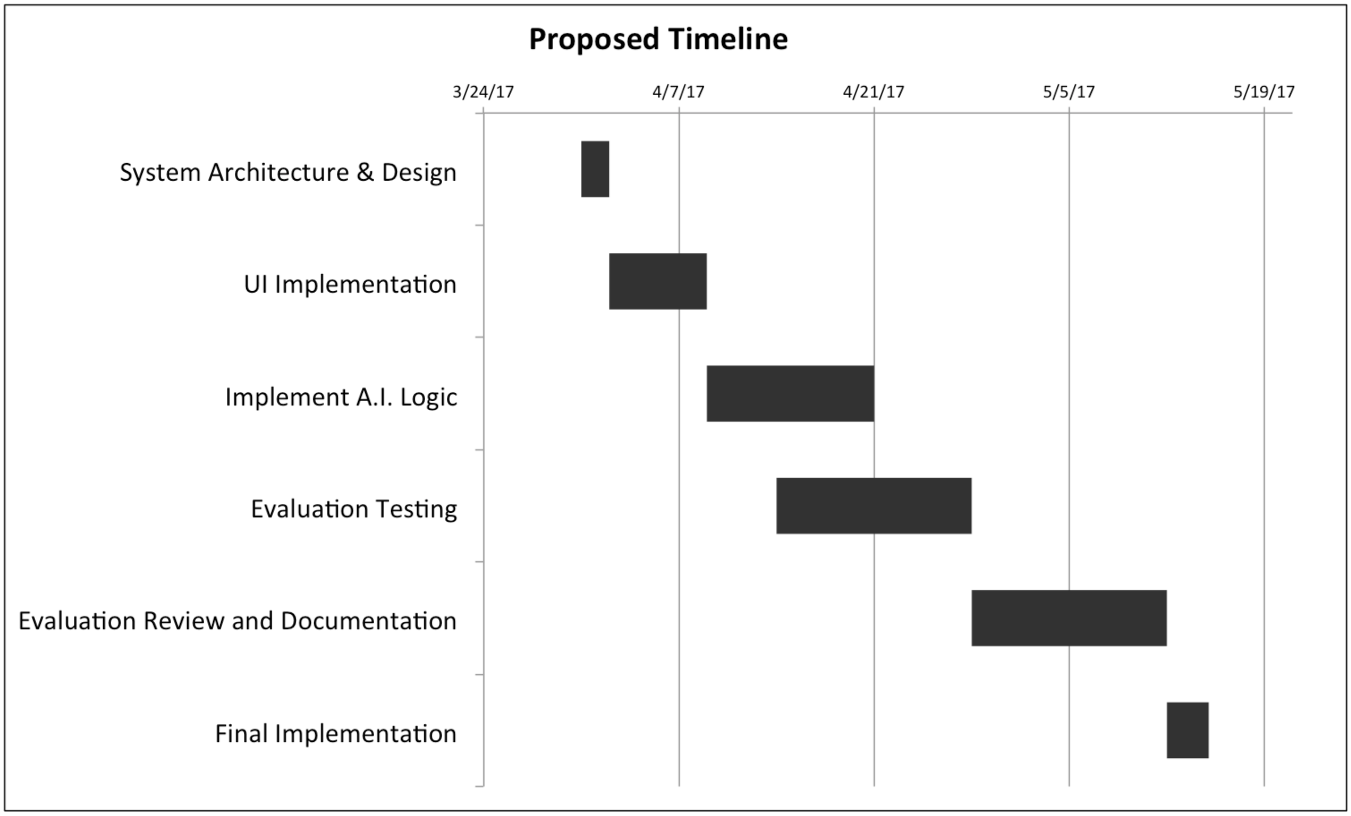
Evaluation

The project will involve finding the best possible states in our system (game). Because poker involves a lot of random number generation and probability theory, the players and AI in the game will have to evaluate their performance by the number of points/coins they win after each round of the game. We can test this by implementing different AIs which could consider varying level of states. A beginner AI will have played a few times, and a medium-level AI would have played multiple times, and a master level AI would have played many times. It would be expected that the higher level AIs would end up with a higher score/money after each round. So Poker starts with a dealing handing out cards, and all of the other players paying the ‘small blind’. An excellent AI will be able to decide to drop out if it has a subpar hand, or stay in if it thinks it has a feasible chance of winning. Then the game plays out and the bets start getting bigger and bigger and it becomes more expensive to stay it. An excellent AI should recognize when it should drop out according to the chance it has to win. After each round of poker (a round being one game of poker), then the AIs can be evaluated according to the amount of money won after the round and if it is still able to play poker (bankrupt players drop out).

Timeline

We will be using a modified waterfall approach in that we work on significant parts of the project in sprints. We will first spend time actually designing the system. We plan to code in Java, but we have not determined the libraries we will be using yet. Next, we will implement the game itself as well as the graphics. We will spend the bulk of our time implementing the Artificial Intelligence logic in the form of a computer-player. After that is built, we will do some evaluation testing to determine the strength of the computer-player. Finally, we will review those results and make changes as needed.

A Gantt chart modeling our proposed schedule and a timeline of critical tasks are included below.



Critical Tasks and Phases		
Start Date	**End Date**	**Task**
3/31/2017	4/2/2017	System Architecture & Design

4/2/2017	4/9/2017	UI Implementation
4/9/2017	4/21/2017	Implement A.I. Logic
4/14/2017	4/28/2017	Evaluation Testing
4/28/2017	5/12/2017	Evaluation Review and Documentation
5/12/2017	5/15/2017	Final Implementation

Important Project Dates	
Date	**Milestone**
3/24/2017	Project Proposal
4/21/2017	Status Reports
5/15/2017	Final Report