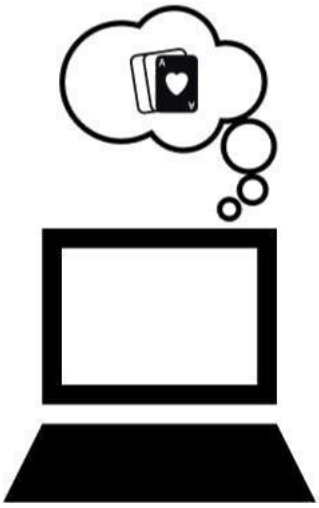
Texas Hold'Em AI Program

Report 5 on work of Week 6

Joseph E. Carrick

October 9, 2015



Application Development Section

1 The Project Concept Proposal

1.1 Purpose

The purpose of this project is to tackle some of the challenges involved in the field of A.I. programming of incomplete information games by creating a Texas Hold'em playing A.I. program.

A.I. programming in games has been around since the 50's. Since then it has proved to match and even rival human players in various games, especially those where brute force search tree algorithms can be utilized (chess, checkers, othello). Texas Hold'em Poker however, is a game of imperfect information, meaning there are elements of the game that are not knowable to the players. This minimizes the computing advantage of A.I. programs. The lack of knowledge in a Hold'em game makes the moves of the players seemingly intuitive, as they have to guess what cards the other players have. This presents an interesting coding challenge.

Typical implementations of A.I. in Texas Hold'em focus on areas of the game where the A.I.'s computational strengths can be utilized. Statistical analysis of one's own hand and calculating pot odds are particularly suited for A.I. processing. Implementations such as these give an advantage to the A.I. in so far as analysis is involved, but outside of this realm the A.I. is lacking.

The most successful Texas Hold'em-playing A.I. today is Claudico from Carnegie Mellon. Claudico is programmed to devise a winning strategy based on rules given to it, which in this case are the rules of Texas Hold'em. The algorithms Claudico uses are not built exclusively for playing Texas Hold'em. This type of A.I. computing requires a lot of processing power and can produce unexpected results.

On April 24th, 2015 Carnegie Mellon pitted Claudico against four of the top professional heads-up Texas Hold'em players in a game of No Limit Texas Hold'em Poker at the Rivers Casino. The match consisted of 80,000 one-on-one games against the computer, with the winner determined by final bankroll. The result: humans still prevail in the world of Texas Hold'em- but only by a small margin.

From this match one can see the potential room for improvement in this particular field of artificial intelligence. Although the time and resources required to program an A.I. like Claudico are unattainable in the scope of this project, there is perhaps a small contribution this project can make through a different approach to this problem.

Professional Hold'em strategy was intentionally excluded from Claudico's al-

gorithms making Claudico a very 'pure' type of A.I.. Since the aim of this project is just to make an A.I. that is good at Hold'em, the algorithms will include Hold'em strategy. The challenge in this approach is that there are an endless number of puzzling scenarios the A.I. can find itself in, and the best possible version of this program would hypothetically need to solve all of them.

The first goal of this project is to design an A.I. that can play heads-up tounament style Texas Hold'em Poker and be able to take into account the statistical analysis of it's own hand. This is fairly straight forward as formulas for determining hand strength and corresponding bet sizes are given in numerous Hold'em strategy books and online. Time permitting, the next step of this project will be to design an A.I. that can also take into account certain game mechanics such as table position and number of players, make guesses as to the strength of opponents' hands, and have the ability pull off bluffing. Implementing any number of these abilities in a Hold'em A.I. should theoretically make it better at playing the game.

The second goal of this project is to create an interface on which to play the A.I.. A preliminary interface will be needed in order to test the A.I. throughout it's development. The two current candidates for this interface are one, a simple text output display, and two, 'theaigames.com'(a website were Texas Hold'em A.I. programs face off against each other). The end interface goal of this project is implementation on a web server.

If a web server interface is achievable, the audience for this project would hypothetically be any Texas Hold'em enthusiast, specifically those interested in A.I. programming.

1.2 Challenges

There are certain axioms in Texas Hold'em strategy that are generally excepted in the professional poker community. Beyond these, however, high-level Texas Hold'em strategy is not singularly formulated. Many professional Hold'em players have their own methodology that works for them. This presents a challenge in programming Hold'em strategy to the A.I.. That challenge is deciding which strategy is optimal for a machine player to use.

Once it has been selected, implementing the strategy is the next challenge. Any strategy beyond statistical analysis of one's own hand requires observation of how their opponents play. This might require a detailed game log with analysis and, or, personality profiles (aggressive, conservative, analytical, etc).

Another challenge with implementing a strategy is weighting the decisions of the A.I.. When a human player faces a decision on the poker table most high-level strategy books recommend the player go through a list of questions about the scenario at hand using the answers to those questions as pros and cons in his

decision. Humans have the ability to get very good at assigning importance to pros and cons without a numerical system of measurement; and on account of Texas Hold'em being a game played primarily by humans, the system of assigning importance to pros and cons as given in most Hold'em strategy books is not based on numerical measurements. In fact, a lot of high-level poker strategy is fuzzy.

Here is an example of this system applied by Phil Gordon in his book, *Phil Gordon's Little Green Book*. To deal with the scenario 'Playing Great Hands When They Raise' (p 46) Gordon suggests asking these questions:

- (How good is my) position?
- How good is my opponent?
- How strong is their hand?
- How do they like to play?
- How strong is my hand?
- How many chips do I have?

Gordan's follow up advice is great for a human player trying to get a feel for how to play in this particular situation, but his advice is such that would make programming it difficult. For example, most of his advice for these questions are worded like, "If ... then I am more likely to ...". Figuring out how to quantify a statement like this and weighing the importance of each question versus the others will take some thought.

One of the core elements in poker strategy is calculating ones own hand strength, or winning percentage, with the cards currently in hand. These calculations can get dicey as one can see by the following math. There are 52 cards in a Texas Hold'em deck, with 2 in each player's hand and 5 on the table. The total number of possible outcomes therefor is the product of each combination of set of cards:

C(52,2) * C(50,2) * C(48,5) = 1326 * 1225 * 1712304 = 2.781381e+12

With almost 3 trillion possibilities consider, it is apparent that a simple brute force algorithm is not the way to go; at least not with the resources at hand. An efficient hand ranking algorithm is necessary for this project, and is a forseable challange.

1.3 Measures

The completeness of this project is measured by two outcomes: a successful Hold'em-playing A.I., and a playable interface for the game. Success in Texas Hold'em is difficult to define. Texas Hold'em players that are generally considered successful could be those with a high win-to-loss ratio or those with the

highest accumulated chip stack. Since the game type for this project is Heads-up tournament style Texas Hold'em poker, a good approximation for success is a win-to-loss ratio higher than 1 to 1. The skill level of the A.I.'s opponents may throw a kink in this approximation however, so another method of approximating success may be to simply run a play-by-play analysis of the A.I.'s decisions to determine if they are consistent with high-level poker strategy.

1.4 Future Extensions

If basic statistical analysis of one's own hand is as advanced as this program gets, an obvious future extension is the incorporation of 'intuitive' plays as described in the goals section. However, since there are endless scenarios in Texas Hold'em and many various strategies to deal with these scenarios, there will always be room for improvement.

Update (September 23, 2015) - A practical future extension of this project could be to create an interface that allows the user to adjust the AI's playing style or difficulty. There are different ways this could be done. One is to allow the user to input a number value representing a point on a scale between 100 percent conservative and 100 percent aggressive. Another way is to pre-program poker styles that emulate various professional poker players, and have the user pick one he wants to play against. One more, less practical, way is to create an interface that allows the user to program the AI directly.

1.5 Resources

- Platform: Update (October 2, 2015) - DigitalOcean for server.
- Languages: Python for A.I. programming; HTML, Javascript, and PHP for interface.
- Libraries: Update (October 2, 2015) - 'Tkinter' for testing-GUI. 'random' and 'math' for A.I. algorithms.
- Tools: Update (October 2, 2015) - PyScripter, GitHub, Atom, and vim for code editing. Django for web framework. Namecheap for domain name registration and SSL certificates. ShareLaTex for documentation and reports.

2 Inspiration

2.1 Motivation

Texas Hold'em Poker has been a hobby of mine for about two years now. I love the game and I play it frequently online and with friends. The idea to make an A.I. Hold'em program came to me last semester (Spring 2015) when I started reading Gus Hansen's book *Every Hand Revealed*. In the book, Hansen goes through almost every hand he played at the Aussie Millions describing the process of his deliberation for each play he made. While reading this book I couldn't help but think of how interesting and challenging the task of creating a Texas Hold'em A.I. would be. I decided to consider it for my senior project. Over the summer the idea became more and more appealing to me, and I decided to go with it.

2.2 Profession

A.I. programming in gaming is a big industry with lots of opportunity. Since the goal of this project is to emulate professional poker strategy in a program, the type of A.I. programming used in games falls right in line with the type of A.I. programming I hope to achieve in this project. That is to say, the A.I. mimics intelligence rather than trying to reinvent the wheel. Statistical analysis of systems with incomplete information is applicable to more fields than just games. This kind of analysis is used fields such as medicine, auctions, negotiation and cybersecurity. This project could point me in the direction of this kind of research, which also interests me.

3 Vision and Scope

Many of the world's greatest thinkers and leaders in science and technology (Steven Hawking, Bill Gates, Elon Musk, Sam Harris and more) agree on the possibility of true artificial intelligence being achieved in the future. This possibility makes the development of AI all the more interesting and engaging. However, the development of true artificial intelligence (with consciousness and originality) is generally agreed to be long way off. In the mean time, programming machines to mimic human reason in specific situations has proven to be very useful, and has seen a large margin of success. The ideal of this project is to achieve this success in Texas Hold'em strategy. This means creating an AI with actions indistinguishable to those of a proficient player.

Programming a computer to calculate the best odds of winning a hand is computationally straight forward, and is an important consideration in almost any Hold'em strategy. However, deeming this type of program to be AI wouldn't do justice to the ideal of this project. It is therefor the very minimum in this project's scope. Programming a computer to mimic a more complex human process, such as a specific Hold'em strategy that incorporates psychology, is much more true to the vision of artificial intelligence, and is where the scope of this project extends to. Possible variations of the AI program at edge the scope include creating several AIs that individually mimic the style of different Hold'em professionals, and creating an AI that switches between styles of play based the style of it's opponent.

4 Software Requirements Specifications

1.
   - Statement: Software needs to run a Texas Hold'em game loop.
   - Evaluation Method: The user can play through a complete game of Texas Hold'em- ending when either player runs out of chips.
   - Dependencies: Card, Table, DetermineHand and GamePhase classes
   - Priority: Essential
   - Requirement Revision History: September 11, 2015, Initial software plan for game play mechanics.

2.
   - Statement: Software needs to accept user input for game actions.
   - Evaluation Method: The user can start a new game, end the current game, check, raise, call, and fold; and these actions correctly update the GUI..
   - Dependencies: Game Loop, GUI; Card and Table classes.
   - Priority: Essential
   - Requirement Revision History: September 11, 2015, Initial software plan for GUI.

3.
   - Statement: Software needs to accept AI input for game actions.
   - Evaluation Method: The AI can check, raise, call, and fold; and these action correctly update the GUI.
   - Dependencies: Game Loop, GUI, AI program; Card and Table classes.
   - Priority: Essential
   - Requirement Revision History: September 11, 2015, Initial software plan for AI program and GUI.

4.
   - Statement: Program needs to be hosted on a web server.
   - Evaluation Method: The user can access the game via the Internet.
   - Dependencies: Game Loop, GUI; Card, Table, DetermineHand and GamePhase classes,
   - Priority: High
   - Requirement Revision History: September 11, 2015, Initial software plan for user interface.

5.
   - Statement: Software needs calculate hand strength and pot odds.
   - Evaluation Method: The AI program consistently computes the same hand strength as the hand strength calculator on pokernews.com, and can compute the ratio between the call amount versus the pot stack.
   - Dependencies: Card, Table, DetermineHand, and GamePhase classes.
   - Priority: Essential

- Requirement Revision History: September 11, 2015, Initial software plan for AI program.

6.
- Statement: Software needs to keep track of how conservative or aggressive the human player bets on the pre-flop, flop, turn, and river phases.
- Evaluation Method: A database of the human player's bets (above or below the recommended amount) on every phase that the hands are revealed is generated and updated throughout the game.
- Dependencies: Hand strength and pot odds calculator; Card, DetermineHand, Table, and GamePhase classes.
- Priority: Middle
- Requirement Revision History: October 9, 2015, Initial software plan for opponent betting database.

7.
- Statement: AI software needs to guess what hands the human player might have.
- Evaluation Method: The AI consistently guesses the correct range of hands that the human player's hand is in.
- Dependencies: Hand strength and pot odds calculator, opponent betting database, opponent hand guessing algorithm; Card, DetermineHand, Table, and GamePhase classes.
- Priority: High
- Requirement Revision History: October 9, 2015, Initial software plan for AI program.

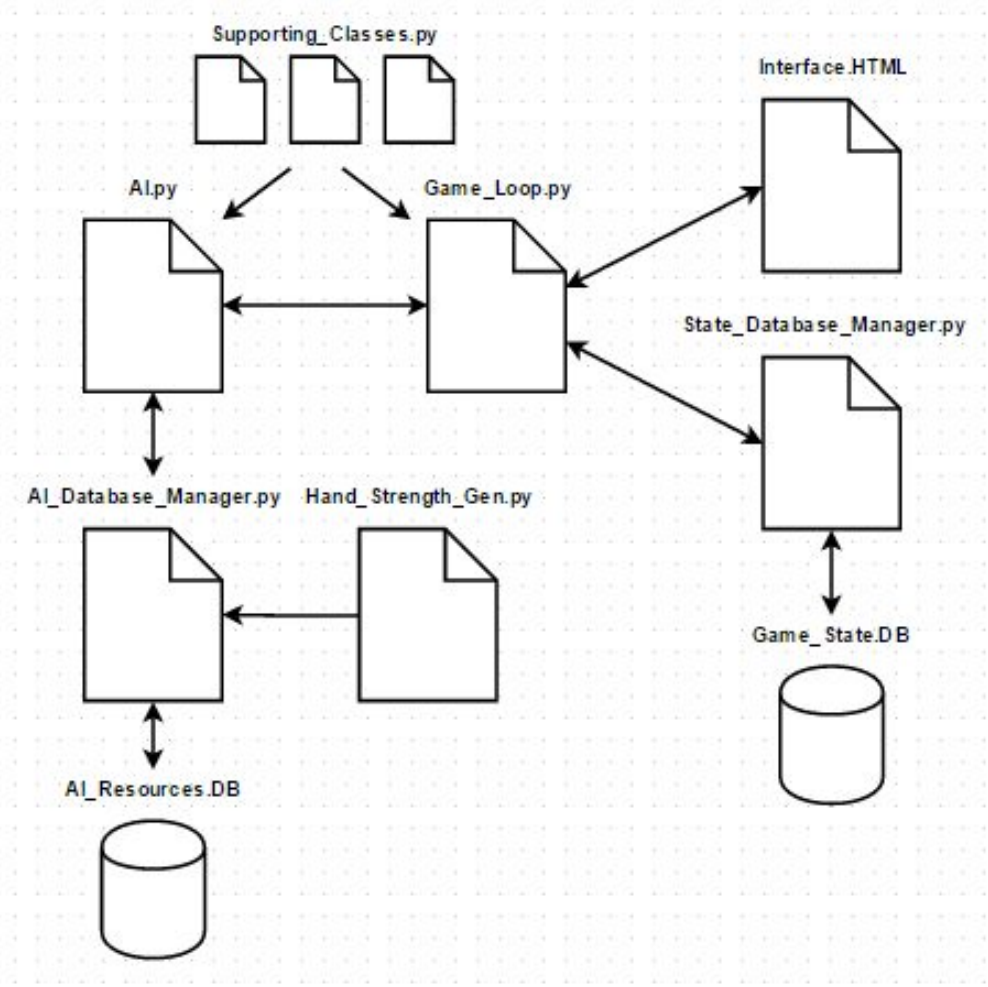
8.
- Statement: AI software needs to make efficient bets to maximize it's overall returns.
- Evaluation Method: The AI's win to loss ratio is greater than 1 to 1.
- Dependencies: Hand strength and pot odds calculator; Card, DetermineHand, Table, and GamePhase classes.
- Priority: Middle
- Requirement Revision History: October 9, 2015, Initial software plan for AI program.

9.
- Statement: User needs to be able to pull up an analysis summary of his game-play after each game.
- Evaluation Method: If followed, the advice in the analysis summary will make the player win more often against the AI.
- Dependencies Hand strength and pot odds calculator, opponent betting database, opponent hand guessing algorithm.
- Priority: If time permits
- Requirement Revision History: October 9, 2015, Initial software plan for Website.

5 System Design and Architecture

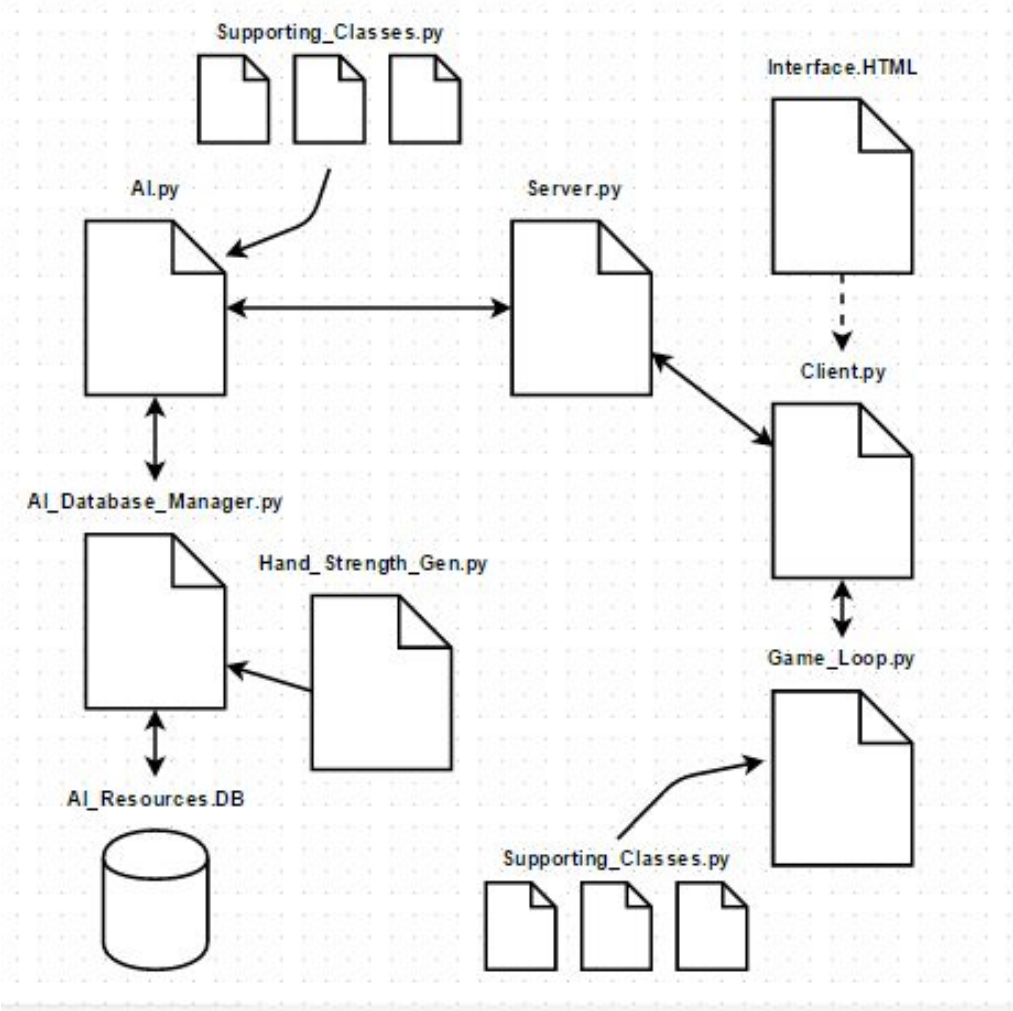
5.1 Browser-based Design

There are two current possibilities for the architecture of this project. The first and original idea is to run the interface in a web browser. In this set up, the interface is written in HTML, CSS and Javascript, and the user calls the Game_Loop.py file to start and input actions throughout the game. To prevent the game from starting over every time the user calls Game_Loop.py to input actions, the game loop uses states which are stored in the Game_State.DB. Every time the user calls the game loop, the most current state of the game is pulled from the database and is used to start the loop where it left off before the user was prompted to act.



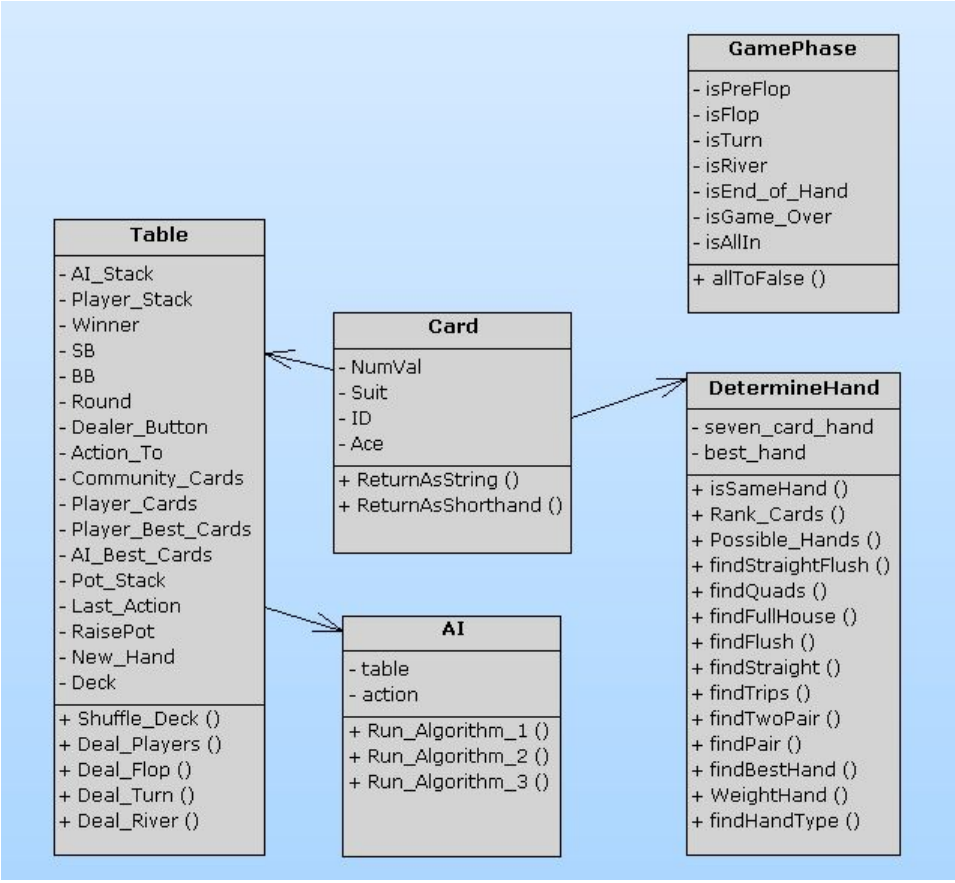
5.2 Client-download Design

The second possible design for the game is a downloadable client design where the game loop and interface are run on a client file on the user's computer. In this design the user would visit the Interface.HTML page to download the client, review the rules of the game, and read information about the project. The Client.py file communicates with the Server.py file to get the AI's moves, while the AI.py file communicates with the Server.py file to get the updated game information, which is running on the client's side. Both sides have their own copies of the supporting classes. This design eliminates the need for the Game_State database because the Game_Loop.py file does not need to be re-run every time the user inputs an action.



5.3 Classes

There are three supporting classes for the Game_Loop and AI files. These are Table, DetermineHand, and GamePhase. Table handles the dealer's actions, as well as stores information about the game's states. GamePhase also stores state information, and DetermineHand does the hand ranking logic. Card class is a support-supporting class; it supports Table and DetermineHand. The AI file is also a class, but it is not considered a supporting class because it does not support the game loop. The AI class simply decides what action to play when it is the AI's turn based on the game state information passed to it by the Table class.



6 Implementation

- README.txt
- THAIP_Game_Loop.py
- THAIP_Class_Table.py
- THAIP_Class_Card.py
- THAIP_Class_DetermineHand.py
- THAIP_Class_GamePhase.py
- THAIP_Class_AI.py

The THAIP_Game_Loop file runs a Texas Hold'em heads-up game in a loop. This is the file that the user runs to the play the game. The THAIP_Game_Loop stores information about the set up of the table (player chip stacks and cards, community cards, the pot, etc) in the THAIP_Class_Table file as the class Table, and information about the current phase of the game is stored in the THAIP_Class_GamePhase file as the class GamePhase.

The THAIP_Class_DetermineHand file contains a class that is called by the game loop to evaluate a player's hand strength and type. The AI class in the THAIP_Class_AI file is called in the game loop when the AI's action is required. The Card class in the THAIP_Class_Card file is called by the Table and DetermineHand classes to get Card objects. The game loop and all classes were completed by September 20, 2015.

Executive Section

To: Dr. Matt Jadud

From: Joseph Carrick

Subject: Texas Hold'Em AI Program

Date: October 9, 2015

Accomplishments

Researched 2d web game engines in python and how to make a downloadable client. Decided on browser based architecture. Worked on the hand strength generator program. Worked on installing Pyjamas (a 2d web game engine supported by Django).

Challenges

Designing efficient algorithms to calculate hand strength. Installing Pyjamas.

Time Spent

2 hours were spent on the hand strength generator program. 2 hours were spent researching 2d game engines and downloadable clients. 2 hours were spent trying to install Pyjamas.

Goals

Install Pyjamas. Set up simple browser based interface.