

# PokerBot - Beta SRS

Team Big Slick:     Nick Cross     Ryan Hermle  
                         Randy Merkel     Spencer Roberts

November 24, 2004

# Contents

0.1	Project Description . . . . .	2
0.1.1	Quick Description . . . . .	2
0.1.2	Purpose . . . . .	2
0.1.3	Tasks . . . . .	2
0.1.4	Domain . . . . .	2
0.1.5	Intended User . . . . .	2
0.1.6	Goals . . . . .	2
0.2	Existing Systems . . . . .	3
0.2.1	Poki's Poker Academy . . . . .	3
0.2.2	Robopoker . . . . .	5
0.3	Requirements . . . . .	6
0.3.1	Master Module . . . . .	6
0.3.2	Rules Module . . . . .	6
0.3.3	Bluff Module . . . . .	6
0.3.4	Main Memory Module . . . . .	7
0.4	System Design . . . . .	8
0.4.1	Class list with functions . . . . .	8
0.4.2	Important Data Formats . . . . .	9
0.5	Prototype and Implementation . . . . .	10
0.5.1	Implementation Language . . . . .	10
0.5.2	Development Tools . . . . .	10
0.5.3	Main Functionality of the Alpha Prototype . . . . .	10
0.5.4	Main Functionality of the Beta Prototype . . . . .	10
0.5.5	Installation and Startup Instructions . . . . .	11
0.5.6	Basic Rules . . . . .	11
0.6	Evaluation Criteria . . . . .	12
0.6.1	All modules are evaluated the same . . . . .	12
0.6.2	Alpha Evaluation . . . . .	12
0.6.3	Beta Evaluation . . . . .	12
0.6.4	Evaluating Goals . . . . .	12
0.7	User Feedback . . . . .	13
0.7.1	Alpha . . . . .	13
0.7.2	Beta . . . . .	13
0.8	Schedule . . . . .	14
0.9	Glossary . . . . .	15

## **0.1 Project Description**

### **0.1.1 Quick Description**

**Poker playing bot with four modules: master, memory, bluff, and rules. Each module is a part of an intelligent player's way of playing.**

### **0.1.2 Purpose**

**Create a competitive poker playing bot that can compete against human and other AI bots.**

### **0.1.3 Tasks**

1. Remember playstyle of players.
2. Evaluate own hand for playability.
3. Evaluate whether other players will fold.
4. Make an intelligent decision to raise, call, check, or fold based on different criteria from each module.

### **0.1.4 Domain**

**The domain is a windows machine. There is a server that is run from the command line that an external client user interface connects to. Big Slick uses windows sockets to communicate data from the server to the clients.**

### **0.1.5 Intended User**

**This bot is used automatically for whatever purpose the observer needs.**

### **0.1.6 Goals**

1. Carry out tasks in visible way to observer.
2. Can verifiably accomplish its goal.

## 0.2 Existing Systems

### 0.2.1 Poki's Poker Academy

Available from <http://www.poki-poker.com/download.html>

#### Good Qualities

- No erratic or odd behavior, they played like average humans might play.
- Various personalities and approaches for each AI player added interest and made them seem less predictable and harder to beat as a group.
- Collected information by analyzing patterns, strategies, strengths and weaknesses.

#### Poor Qualities

- Collects good data, but AI behavior was not noticeably modified by it.
- Game is very limited, players can only bet specified amounts in certain predictable increments.
- Used gain/loss chips system instead of having finite available chips which made individual AI decisions less important.
- Bots play the same regardless of their chip count.
- Bots tend to play their pocket cards and don't give nearly as much consideration to the community cards.

#### User Interface

- This bot has a very nice graphical user interface.
- Displays odds to win of the current hand.
- Can play against multiple opponents.

#### Learning

- This bot did not appear to have the capability to learn.
- In order to test this bot's intelligence, I decided to bet the maximum amount I could on every hand. It did not even appear to recognize this pattern. Bots with low hands would fold quickly, and only bots with high hands would challenge me, even after they saw me lose by bluffing.
- This bot falls victim to bluffing easily.

## **Intelligence**

- The non-learning capability of this bot seemed almost flawless.
- The odds of winning were calculated from every possible hand, starting with the two cards in the pocket and then recalculating every time new cards were shown.
- With two tens out on the flop, the last remaining bot that did not fold from my bluff folded quickly, knowing that I had a good chance of having a three of a kind.
- It analyzed players responses in different situations to try and predict future actions.
- Bots sometimes went against probability, which is the basis for a sort of instinct, to try and fool or beat human players.

## **Similarities to our Design**

- A major goal of ours is to be able to calculate the odds of our current hand in the same manner that this bot does.
- Our master module should also have the capability to predict a situation such as the opponent having a three of a kind when there is a pair out on the flop and they are betting high.
- Compares hand value to potential loss in continueing a hand.
- Takes position into account when making decisions.

## **Differences from our Design**

- This bot never bluffs. Our bot will have the ability to bluff, as well as implement strategies such as slow playing.
- Our bot will establish patterns of the opponent. A simple pattern such as maximum betting on every single hand should be easily detected, and our bot will start matching these bets even if its hand does not have extremely high odds of winning.
- We will not be implementing a GUI interface.
- Our bot will not have personalities.

## 0.2.2 Robopoker

Available from <http://www.contrib.andrew.cmu.edu/~nja/AI/poker.html>

### Good Qualities

- It seems to have personality. I played against 5 bots and one of them constantly folded while another always called no matter what it had.
- They can effectively evaluate what they have and take an action.

### Poor Qualities

- Aggressive bots always bet, regardless of what hand they have.
- Conservative bots always fold if they are being raised, regardless of what hand they have.
- Even though they can take an action, they play poorly.

### Similar Qualities

- Robopoker takes position into account.
- Robopoker does not look at the cards and discover how good they are for it, but also how good they might be for an opponent and takes this into account.
- Robopoker incorporates the fact that people change their betting strategy when their stacks change.
- Robopoker has separate strategies for pre-flop and post-flop game-play.

### Qualities our Bot will not Implement

- Robopoker is designed around a heuristic where different factors change its betting style. This is similar to reflex bot behavior.
- The bot picks personalities at the start. This is not necessarily conducive to a GOOD AI system.
- Robopoker does not try to bluff the other players.

### Displays of Intelligence

- Plays poker in a very reflex oriented manner.
- Changes its strategies as other people's stacks grow and shrink.
- Remembers how people play and adjusts strategy to compensate.

## 0.3 Requirements

### 0.3.1 Master Module

1. Acknowledges cards, sends cards to rules, and receives probability to win.
2. Has  $x$  amount of strategies to choose from, and each strategy has its own individual criteria.
3. Ask bluffing module “Should I bluff?”. Which is akin to asking “Will my opponents fold?” Bluff module returns yes or no along with estimated cost to make people fold.
4. Asks memory module to guess what cards opponents have. Memory module answers in general terms: bad to great scale.
5. Chooses to fold, call, or raise based on each strategy.
6. Uses intelligent and applicable criteria.
7. Leaves a footprint (like to the console and/or log file) about what it decided, and why.
8. Converts the information gathered by each module and measures each one’s weight on a point scale.

### 0.3.2 Rules Module

1. Take cards from master module.
2. Decides probability of winning with pocket and flop considered.

### 0.3.3 Bluff Module

1. Receives short term memory of one hand (including flop).
2. Figures out if bluff is a good idea.
  - Checks flop to make sure nothing too strong is possible.
  - Looks at short term memory to see if someone is playing strongly that would indicate they really do have a big hand.
  - Looking for declining bets and other signs of weakness.
3. Determines how much money it takes to make each player in turn (highest common factor) fold.
4. Returns to master:
  - If the bluff is a good or bad idea.
  - How much money to make everyone fold.

### 0.3.4 Main Memory Module

1. Tree structure with tables.
2. Recieves short term memory element and logs revelent information into main memory.
3. Capable of answering questions from bluff module where it takes in the short term memory and an amount to return a yes or a no if they will fold.
4. Capable of answering questions from master module where it returns what cards an opponent likely has on a scale from bad to great.

## 0.4 System Design

### 0.4.1 Class list with functions

#### **BigSlick.cpp (Master)**

- BigSlick(int, int, int)
- Action\* decision()
- void setPocket(Card\*, Card\*, int, int, int\*)
- void setFlop(Card\*, Card\*, Card\*)
- void setRiver(Card\*)
- void setTurn(Card\*)
- void setDec(int, int, int, int\*, int\*, int)
- void setWin(int, int\*, int)
- void setSMB(int, int)
- void setLB(int, int)
- void flip(int, char, char, char, char)

#### **Memory.cpp**

- Memory(int, int)
- void setMemory(STM\*)
- int guessHand(STM\*, int)
- int willFold(STM\*, int)

#### **Bluff.cpp**

- Bluff(Memory\*, int)
- int shouldIBluff(STM\*)

#### **Rules.cpp**

- float getHandValue(Card\*[2], Card\*[5])

## 0.4.2 Important Data Formats

### Card.h

- char num
- char suit
- Card(char num, char suit)

### Action.h

- int act
- int raise
- int toMe
- Action\* newAct
- Action()

### STM.h

- int curRound
- int numPlayers
- int potSize
- int numWinners
- int\* position
- int\* winner
- int\* startMoney
- Action\*\* playAction[4]
- Card\* cards[5]
- Card\*\*\* pocket
- STM()
- STM(int, int\*)
- void addAction(int, Action\*)
- void finishHand(int, int\*, Card\*\*\*)

## **0.5 Prototype and Implementation**

### **0.5.1 Implementation Language**

The AI robot and interfaces will be implemented using C++ and OpenGL.

### **0.5.2 Development Tools**

Visual C++ is the primary programming tool.

### **0.5.3 Main Functionality of the Alpha Prototype**

**Server and client functionality**

1. GUI Interface
2. Allows AI and Human players.
3. Plays a few rounds before crashing.

**Requirements satisfied**

**0.3.1.1** Does not acknowledge cards, uses random for hand value.

**0.3.1.2** Has zero strategies calls random.

**0.3.1.4** Returns medium hand from opponents all the time.

**0.3.1.5** It chooses to fold, call, or raise.

**0.3.1.8** It converts the information gathered and decides what to do.

**0.3.2.2** It decides the probability of winning, but is random

**0.3.3.4** Bluff is not implemented at all.

**0.3.4.2** Recieves a short term memory to evaluate the past, but only returns a average hand.

### **0.5.4 Main Functionality of the Beta Prototype**

**Server and client functionality**

1. Game runs successfully until someone loses all their money. Segfaults.
2. Vastly improved server and client interactivity.

### **Requirements satisfied**

**0.3.1.1** Looks at the pocket cards and produces probability on that.

**0.3.1.2** Has one strategy, conservative. High call chance.

**0.3.1.7** Prints to the screen what decisions it makes and why.

**0.3.2.1** Takes pocket cards from the master module.

**0.3.2.2** Not random anymore for probability, uses pocket cards.

### **0.5.5 Installation and Startup Instructions**

1. Acquire executables.
2. Start Server.
3. Start a Client for the human observer.
4. Connect Client to Server.
5. The server adds AIs to the server.
6. The server starts the game.

### **0.5.6 Basic Rules**

**All Texas Holdem Tournament rules apply.**

## 0.6 Evaluation Criteria

### 0.6.1 All modules are evaluated the same

**The AI reports this information on each decision to allow the user/observer to evaluate the intelligence of the decision.**

1. Shows point tally.
2. The strategy used.
3. Output from bluffing module.
4. Final decision.
5. One sentence reason.

### 0.6.2 Alpha Evaluation

1. Start a game.
2. Play until crash while observing AI's decisions.
3. Confirm that they raise, call, and fold.

### 0.6.3 Beta Evaluation

1. Start a game.
2. Play a few rounds while observing AI's decisions, both visually and through the information displayed on the server console.
3. The bots should make intelligent decisions based on their pocket card strength, and amount of risk each bet to them is.
4. The bots should demonstrate a conservative playing style unless they have an exceptionally high hand, where they will go all in.

### 0.6.4 Evaluating Goals

**Testing the AI agent against a human player will show whether the agent is achieving its goal of becoming a competitive poker player.**

**Add another agent to an ongoing game and see if the older agents perform better. This demonstrates that the memory module works effectively.**

## 0.7 User Feedback

### 0.7.1 Alpha

The server and client can be started successfully. The bots randomly decide to fold, call or raise. They should consider their hands and not fold when they have good hands. The server crashes every three or four rounds.

### 0.7.2 Beta

The game is relatively playable, as long as noone loses all their money. The conservative bot is predictable because of the high emphasis on the amount of money bet to them. A suggestion would be to add multiple playing styles to make them more challenging.

## 0.8 Schedule

Week	Activity
1	Make groups. Brainstorm idea.
2	Write SRS. Contact the other group. SRS draft due.
3	Write interfaces and memory modules.
4	Create skeleton classes. Module's interface. Alpha release.
5	Implement features.
6	Implement features. Beta release, 75% functionality.
7	Test and evaluate.
8	Test and evaluate. Final release.

## 0.9 Glossary

**Community:** Cards dealt face up that all players at a table share.

**Flop:** The second round where there are 3 community cards.

**Game:** Consisting of multiple hands which extends from the time players begin play at a table to the time there is only one player left at a table.

**Hand:** Consisting of multiple rounds (four in Texas Hold'em) which extends from the time cards are dealt to the time one pot is won.

**Pocket:** The two cards dealt face-down to a player.

**Pot:** The sum of all bets from one hand.

**Pre-flop:** The round immediately after pocket cards are dealt and before the flop.

**River:** The fourth round where there are 4 community cards.

**Round:** As in "Round of betting" where every player has had the chance to bet once.

**Stack:** The amount of money a player has at any given time.

**Table:** All players involved in a game up to a max of 10 (6 norm).

**Turn:** The third round where there are 4 community cards.