|  |
| --- |
| **Poker Tech Design Document** |
| **Final Project** |
|  |
|  |

By:

Levi Squires

Justin Morritt

Table of Contents

[Introduction 3](#_Toc400605648)

[Main Game Loop 4](#_Toc400605649)

[Game Application Modules 5](#_Toc400605650)

[User Interfeace and Inputs 5](#_Toc400605651)

[Coding Standards 6](#_Toc400605652)

[Development Tools 7](#_Toc400605653)

# 

**Introduction**

The Poker hand evaluator is a game designed to simulate one hand of poker from a possible four poker game variants. The main goal of this project is to be able to choose a poker type, input the number of players and have a winning result displayed.

Bonus goals for this game are as follows:

* Implement different hand evaluation algorithms and allow the user to choose a different hand evaluation algorithm (completed).
* The game will stop at certain points during the game and calculate the odds of winning for each player.
* Implement a betting system.
* Allow the user to play interactively and choose to fold, bet, raise, etc.
* Implement AI for the game that will allow them to play their own hands and make decisions like fold, bet or raise.

The main objective and any other bonus objectives implemented need to be fully working and compiling by the project due date.

The deadline for this project is Friday, October 10, 2014. To keep on schedule a list of milestones needs to be followed.

Milestone 0.2

Implement a table with a menu that includes different constructors and destructors based on chosen game.

Milestone 0.3

Players and dealers created through appropriate tables.

Milestone 0.4

Dealer’s deal() function gives cards to player. Players display cards to table.

Milestone 0.5

Players builds a set of CUEs and are evaluated to choose the best one. Each hand is given a text description and displayed to table.

The project code including any pseudo code will be written entirely in C plus plus.

# Main Game Loop

The main game loop for our game is just a simple to call to the game function. Our goal from the beginning was to have the main loop be as small as possible so we built the game function to encapsulate the entire project. The game function includes: the main menu, the set game types and players, change hand evaluator along with the option to quit the game.

When a game type is chosen and a number of players have been inputted the create table function is called and a table is created that handles player creation and dealer creation along with calling the evaluator and displaying a winner. Then each player and the dealer is destructed and the table is then deleted as well.

# Game Application Modules

**Game Items**

The poker game uses only one item and that is a standard deck of 52 cards. These cards are shuffled then dealt to each player through the use of a dealer. All four poker variants use all 52 cards.

**File Loading**

The entire project excluding the hash evaluator is loaded in as soon as the game starts. The hash evaluator will be loaded in when it is first selected to be used and will remain loaded until the end of the program along with everything else.

**User Interface and Inputs**

The main interface the user will be using is a text based menu that has numbered controls to move throughout various options of the game.

1. **Texas Hold em**
2. **Omaha High**
3. **7 Card Stud**
4. **5 Card Draw**
5. **Evaluator Type: Sorted/Hash**

**(0) Quit**

This is the first menu that is displayed.

(**0) Quit**

1. **Main Menu**
2. **Replay!**
3. **Change Evaluator, Current: Sorted/Hash**

This is the menu displayed after showing a winning hand.

All player input will be through the number keys 0 – 5.

# Coding Standards

**C++ Code**

All code should be kept out of the main.cpp file as this is to only include the game function necessary for running the game.

All functions, classes and variables will be declared using a camelcase format with the exception of constants which will be declared using all caps. All private variables will start with an underscore to easily identify them and all member variables will begin with m\_.

All header definitions should be in the #define Header\_H\_\_ format.

All files need to be stored on the GitHub repository and any time changes are made a comment with the changes must be included.

Any time a class constructor is called test code must be implemented to ensure each class destructor is being called. This test code will be removed from the final version.

# Development Tools

The development tools that will be used in this project are Visual Studio 2013 for making the project files and writing code, GitHub as the repository, Microsoft Word for creating the design documents and White Star UML for the UML file.