

Multi Touch Poker Table_{MTPT}

Group IV
Nathaniel Boucher
Christopher Herod
Raeginald Timones

Table of Contents

1.0 Executive Summary	1
2.0 Requirements.....	4
2.1 Hardware Requirements	4
2.2 Software Requirement Specification	5
2.2.1 Poker Game.....	6
2.2.2 Menu.....	8
3.0 Hardware.....	9
3.1 Multi-Touch Screen	9
3.1.1.1 Technique	9
3.1.1.6 Design.....	13
3.1.1.7 Cost and Procurement	14
3.2 LED's	15
3.3.1 Cost and Procurement	16
3.3 Intake/Exhaust Fans	16
3.3.1 Research	16
3.3.2 Design.....	17
3.3.3 Cost and Procurement	19
3.4 Webcam.....	20
3.4.1 Research and Design Requirements	20
3.4.2 Cost and Procurement	21
3.5 Wireless Adapter.....	22
3.6.1 Research	22
3.6.2 Cost and Procurement	22
3.6 Projector.....	23
3.6.1 Research	23
3.6.2 Cost and Procurement	24
3.7 Framework	25
3.7.1 Research	25
3.7.2 Design.....	25
3.7.3 Material and Construction	26
3.7.4 Cost and Procurement	27
3.8 Power.....	28
3.8.1 Devices	28
3.8.2 Design.....	32
3.9 Computer	33
3.9.1 Specifications.....	34
3.9.2 Cost and Procurement	Error! Bookmark not defined.
4.0 Software.....	36
4.1 Overview	36
4.1.1 Block Diagram.....	36
4.1.2 Development Environment.....	37
4.2 Research.....	<u>4039</u>
4.2.1 Libraries	<u>4039</u>

4.2.1	PokerSource Poker-Eval Evaluator.....	43
4.3	Detailed Design.....	<u>4645</u>
4.3.1	Poker Game.....	<u>4645</u>
4.3.1.7	Table Interface.....	<u>6462</u>
4.3.1.8	Dealer Interface.....	<u>6765</u>
4.3.1.9	iPhone Interface.....	<u>6866</u>
4.3.1.10	The Gesture Handler.....	<u>6967</u>
4.3.2	Full Interface Guidelines.....	<u>7068</u>
	Restaurant Menu.....	<u>7270</u>
4.3.2.1	MenuWidget.....	Error! Bookmark not defined.
4.3.2.2	GestureHandler.....	<u>7371</u>
4.3.2.3	Menu Interface.....	<u>7775</u>
5.0	Design Summary.....	<u>7977</u>
6.0	Prototyping.....	<u>8381</u>
6.1	Hardware.....	<u>8381</u>
6.2	Software.....	<u>8381</u>
7.0	Test Plan.....	<u>8684</u>
7.1	Hardware.....	<u>8684</u>
7.1.1	Screen.....	<u>8684</u>
7.1.2	IR Camera.....	<u>8684</u>
7.1.3	LED's and Power Supply.....	<u>8684</u>
7.1.4	Fans and Temperature Sensor.....	<u>8684</u>
7.1.5	Wireless Router.....	<u>8684</u>
7.1.6	Framework.....	<u>8684</u>
7.1.7	Projector.....	<u>8684</u>
7.2	Software.....	<u>8785</u>
7.2.1	Poker Game Test Cases.....	<u>8785</u>
7.2.2	Restaurant Menu Test Cases.....	<u>8987</u>
8.0	Budget.....	<u>9088</u>
9.0	Milestones.....	<u>9189</u>
10.0	Appendices.....	i
11.1	Copyright permissions.....	i
11.2	Software Licenses.....	iii
11.3	Works Cited.....	X

List of Tables

Table 2-1 Poker Game Blind Structure	7
Table 3-1 Technique Decision Matrix	14
Table 3-2 Multi-Touch Screen Cost	15
Table 3-3 LED Cost Comparison Table	16
Table 3-4 Fan Cost Comparison.....	20
Table 3-5 IR Camera Options.....	21
Table 3-6 Dell 1100MP Specifications.....	24
Table 3-7 Framework Cost	27
Table 4-1 MTPT Support Libraries	40
Table 4-2 Planned Qt Implementation	42
Table 4-3 TouchData	43 42
Table 4-4 PokerSource Data Types	45 44
Table 4-5 PokerSource Functions	45 44
Table 4-6 Poker Hand Rankings.....	48 47
Table 4-7 QMap Implementation	51 50
Table 4-8 MTPT Gestures	77 75
Table 5-1 Poker Game Classes.....	81 79
Table 5-2 Restaurant Menu Classes	82 80

List of Figures

Figure 1-1 Multi Touch Poker Table SketchUp	2
Figure 1-2 MTPT Block Diagram	3
Figure 2-1 MTPT Hardware Components.....	5
Figure 2-2 MTPT Software Components	6
Figure 3-1 Frustrated Internal Reflection Technique	10
Figure 3-2 Rear Diffused Illumination technique	11
Figure 3-3 Laser Light Plane Technique.....	12
Figure 3-4 Diffused Screen Illumination.....	13
Figure 3-5 Fan Placement Drawing	18
Figure 3-6 Temperature Switch Circuit and PCB.....	19
Figure 3-7 Table Framework Drawing	27
Figure 3-8 LED wizard schematic and Power consumption	29
Figure 3-9 Power supply components	29
Figure 3-10 Typical ATX power supply connector	31
Figure 3-11 LED Power Supply Circuit.....	32
Figure 3-12 Internal Wire and Component layout of Table	33
Figure 4-1 MTPT Software Block Diagram	37
Figure 4-2 XCode Interface	39 38
Figure 4-3 PokerSource Card Mask Format.....	44 43
Figure 4-4 Texas Hold'em Event Diagram.....	47 46
Figure 4-5 Poker Game Exec Event Diagram.....	49 48

Figure 4-6 Deck Activity Diagram	5049
Figure 4-7 PokerExec Class Diagram	5254
Figure 4-8 Poker Game SIGNAL/SLOT Connections.....	5352
Figure 4-9 Betting Activity Diagram	5453
Figure 4-10 Player Hand Calculation Activity Diagram	5554
Figure 4-11 PokerServer Class Diagram	5756
Figure 4-12 Poker Game Sequence Diagram	5857
Figure 4-13 PokerPlayer Class Diagram	5958
Figure 4-14 TouchListener Class Diagram	6364
Figure 4-16 Player Interface	6563
Figure 4-17 Player Hole Cards	6967
Figure 4-18 Color Chooser	6967
Figure 4-19 Full Interface	7169
Figure 4-20 Active Player Interface	Error! Bookmark not defined.
Figure 4-21 MenuWidget Class Diagram.....	7270
Figure 4-22 Bet Gesture	7374
Figure 4-23 Check Gesture	7472
Figure 4-24 Pinch Gesture	7472
Figure 4-25 All In Gesture	7573
Figure 4-26 Fold Gesture	7674
Figure 4-27 Incorrect Fold Gesture.....	7674
Figure 5-1 MTPT Block Diagram	7977
Figure 5-2 MTPT Software Components	8179
Figure 6-1 Touchlib Configuration Application	8583
Figure 6-2 Smoke Demo	8583

1.0 Executive Summary

The Multi Touch Poker Table (MTPT) explores the growing popularity of multi-touch technology and showcases it with an entertaining game of poker. Using a less expensive approach, the MTPT allows four people to leisurely enjoy a game of Texas Hold'em Poker with their iPhone and iPod Touch devices. Commercial technology utilizes more expensive methods such as capacitive sensing. The MTPT utilizes visual techniques. This project is motivated by the growing ownership of such devices and multi touch technology in general. Touch screens have been around for a long period of time, but only lately people have had the pleasure of having multi touch capabilities for their own personal use, through phones and entertainment systems. These devices are carried by numerous people, including a growing number of enterprise users. The ability to use these devices to create community of pleasure locally is something that has not been explored. Ideal places for the end use of this table would be at a café, waiting room, or other public places. Along with providing a full game of poker, the table will provide a convenient way to view a menu of the location the table resides. There are multi touch tables out in the market today, specifically the Microsoft Surface. The goal of the MTPT is to bring a scaled version of tables, such as the Surface, to an end user at only a small fraction of the cost.

Texas Hold'em is a popular derivation of poker. The game was chosen for its ease of learning and enjoyment for all ages. Poker deemed an appropriate application to showcase the possibilities and capabilities of multi-touch technology for public use. The game involves user interactions that can be better simulated through touch commands than simple buttons. Furthermore, the software needed to create such an application challenges the group in design, execution, and testing.

This Senior Design I document outlines the requirements, research, design, prototype and test plan for the Multi Touch Poker Table. The documentation is divided between hardware and software. The full block diagram of the MTPT project is illustrated in each section will have the respective sections. These sections specially outline every aspect that will encompass the eventual prototype to be built by Group IV. The requirements specified herein are derived from the group members. These hardware and software requirements are based on the time and difficulty of the project. Based on extensive research from various sources, the group has documented a detailed design of a MTPT prototype. This prototype will then be tested with the test plan against the specified requirements.

The requirements section outlines the expected performance and specifications of the hardware and software of the MTPT. These requirements are explicitly stated. All research, work, and testing will derive from these requirements. The

requirements were derived from the personal expectations of the group. The major contributing factors that were considered when formulating the requirements include budget, time, and difficulty.

Research for the MTPT was conducted for the hardware and software tasks. Research involved in this project was directed towards hardware approaches for the multi-touch screen. Various optical methods were analyzed and considered. These approaches were compared against each other, the project's requirements, and budget. Software research was conducted for the best possible resources in libraries. These libraries are explained as well as their intended integration into the MTPT. Overall, the research portion provides the group with a foundation to design on top of. With a better understanding with what the group wishes to achieve, designs were formulated based on experience and the conducted research.

Design encompasses the majority of this document. The software detailed design is based upon research and current experience. These designs are not final and may change as a result of testing or requirement modifications. The MTPT software is divided into components. These components are further explained to the class level.

The prototype plan keeps in mind the division between the MTPT hardware and software that will eventually execute on the hardware. Once a prototype is made, it will be tested according to the MTPT test plan. The hardware will be tested accordingly before it is integrated with the software. Test cases were written to test the MTPT for proper functionality. The results from the project test plan will be compared to pass/fail criteria, expected results, and the specified requirements of the requirement specification section. Using set criteria, the group will ensure the prototype works according to expectation. A theoretical illustration of the prototype is shown in [Figure 1-1](#) (Coffee Mug, iPhone, and Couch Models obtained from Google 3D Warehouse).



Figure 1-1 Multi Touch Poker Table SketchUp

The MTPT hardware encompasses the multi-touch display, computer, router, iPhone devices, and power supply. All of the hardware will be enclosed in a designed constructed framework. The MTPT software includes the Poker Game software, Restaurant Menu software, and the iPhone interface. These software components with the exception of the iPhone interface will run on the computer of the MTPT hardware. The MTPT software will utilize a number of open source libraries, but the majority will be developed by the group. The complete block diagram of the MTPT project is illustrated in [Figure 1-2](#).

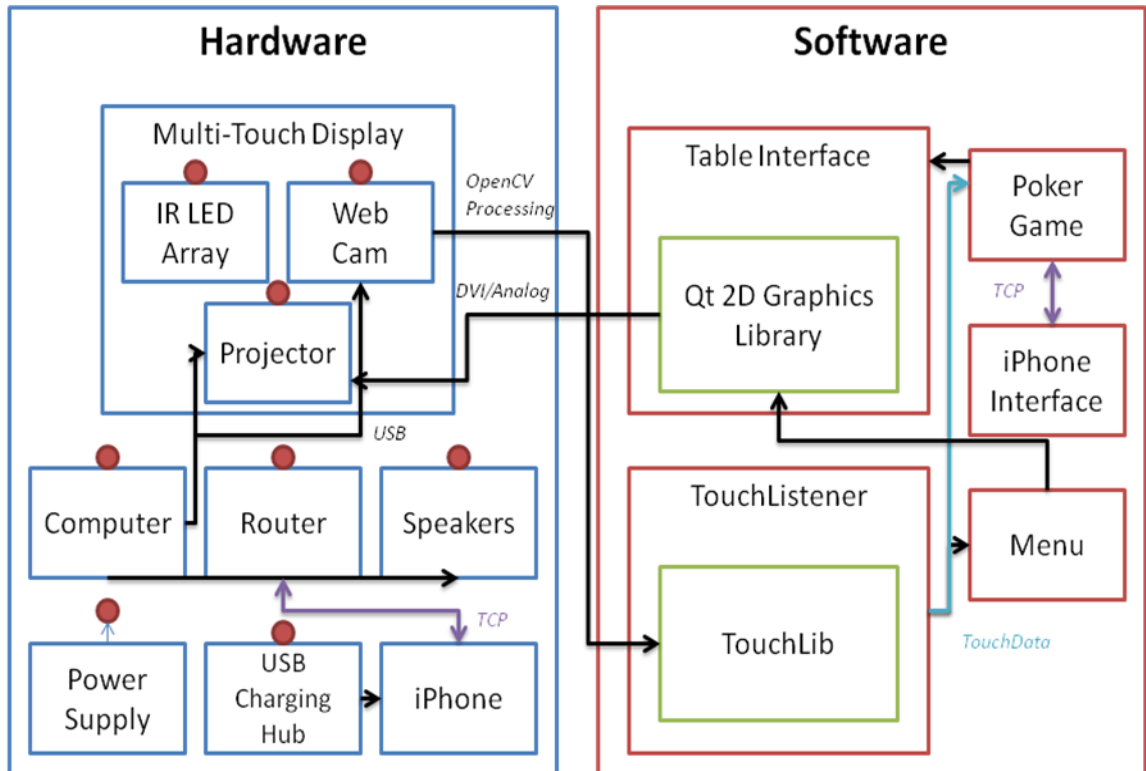


Figure 1-2 MTPT Block Diagram

Motivation for the MTPT derived from creating something from a popular device, the iPhone. Utilizing the iPhone will bring new views on its potential. Besides a working end product, the ultimate goal in undertaking the MTPT project is to create something for people to enjoy while gaining worthwhile experiences that will prove to be invaluable. The MTPT is, at its simplest level, a learning experience. By setting the bar above the comfort zone, the group will be challenged in making this project a success. The group's requirements for this project was not given nor advised; they set their own standard. This document outlines the plan, designs, and steps that Group IV will need to accomplish to achieve this goal.

2.0 Requirements

2.1 Hardware Requirements

In order to properly design a multi-touch device, requirements must be formulated. The hardware requirements encompassed all the physical aspects of the Multi-Touch Poker Table (MTPT). These basic requirements focused research in particular areas and helped mold the design specifications. The following components were derived from each component.

Multi-Touch Display

1. The technique used will allow the user to have multiple touch and dragging capabilities on the display screen.
2. There will be an option to control the sensitivity of the touch screen depending on light conditions.

Projector

1. The projector will have a minimum screen resolution of 800x600
2. The projector output will be able to be seen in high and low ambient light conditions
3. The projector will output a screen ratio of 4:3

IR Camera

1. The camera will have a minimum resolution of 640x480
2. The camera will have a frame rate of at least 30 FPS
3. The camera will transfer data at a minimum of 480 Mbits/s

Computer

1. The computer will have a 1.5 GHz dual core processor or higher
2. The computer will display all graphic without any noticeable freezes or glitches
3. The computer will have at least 2 GB of RAM

Table

1. The constructed table will not be higher than 3 1/2 feet
2. The table top will have enough room to accommodate up to 4 coffee mugs
3. The table will include a four docks to charge devices while playing using USB 2.0
4. The table enclosure will have a internal cooling system to prevent overheating

Wireless Device

1. Communication between the iPhone and computer will be wireless
2. Device will comply with IEEE 802.11g standards

Power

1. The MTPT will have only one 110 VAC external power cord

The hardware for this project contained numerous components based on these requirements. Figure 2.1 shows a block diagram of all the components and how

they interact with each other. The specifications of these components will be introduced in each components section.

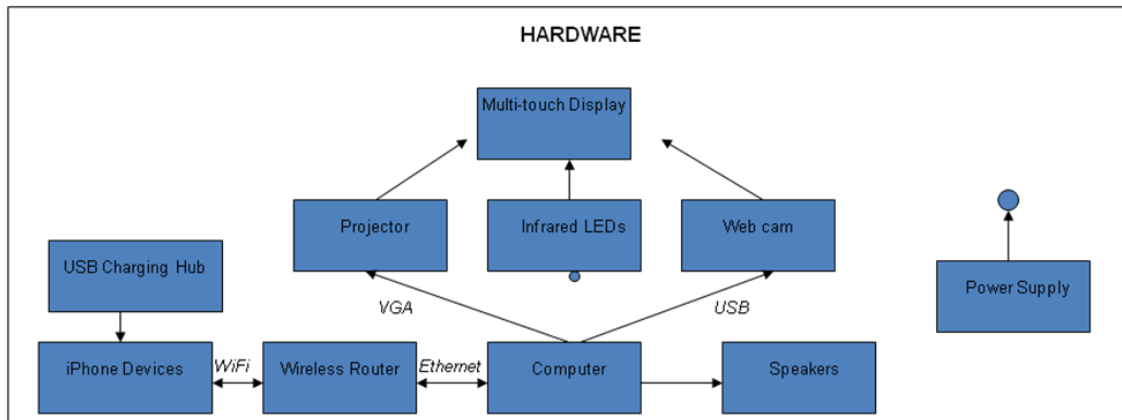


Figure 2-1 MTPT Hardware Components

2.2 Software Requirement Specification

The software that was developed for the MTPT was separated by function. The Poker Game component encompasses the game code, iPhone interface, and multi touch table interface. With exception to the iPhone interface, all components were developed in C++. The Menu component contains ordering functions and the user interface. The complete MTPT Software is illustrated in [Figure 2-2](#). The requirements for each software component will be outlined in this section. Each requirement is specific to the encompassing component. Additional requirements will be addressed at the end of the expected requirement. These requirements will be added once the previous specifications are met. Each additional requirement is prioritized by expected difficulty, from easy to difficult.

The following requirements are derived from the group's personal expectations of how far they would wish to endeavor on this project. Furthermore, the software requirements are based on the amount of time for the scope of the MTPT project. MTPT software design, development, prototyping, and testing are all derived from the software requirement specification. This section will be referenced in the ensuing sections. Each requirement is labeled with the appropriate component and number for traceability.

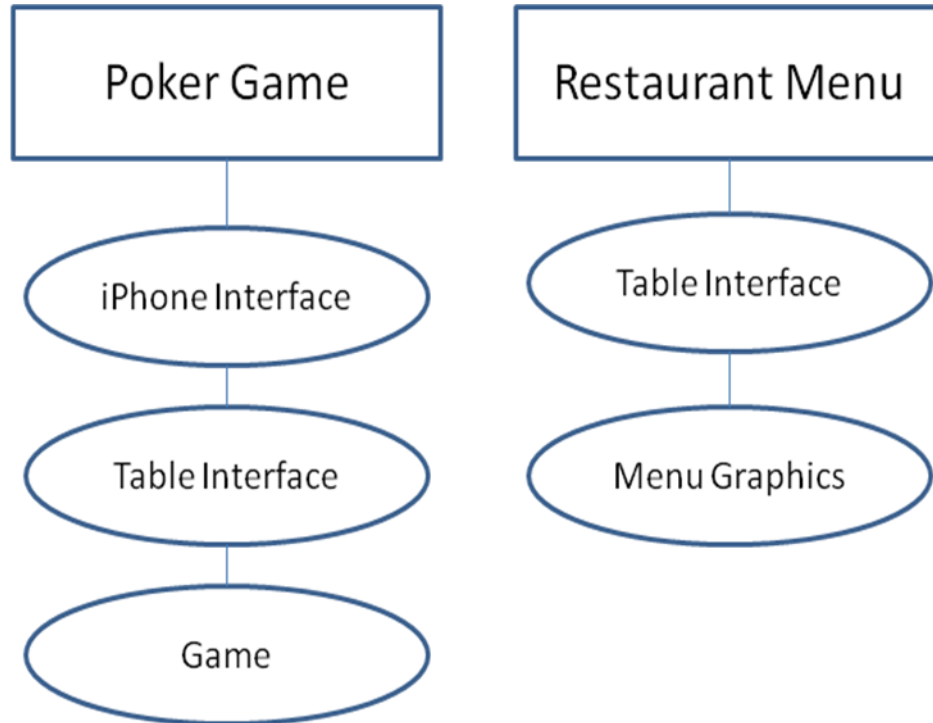


Figure 2-2 MTPT Software Components

2.2.1 Poker Game

2.2.1.1 iPhone Interface

The iPhone interface shall provide the user the two hole cards to be played during that hand of poker. For the remainder of this document, any reference to the iPhone device will include the supported iPod Touch versions as well. The devices chosen were determined by the devices that were released during the creation of this document.

[iPhone_001] The MTPT will interface with up to 4 iPhone/iPod Touch devices.

[iPhone_002] The following iPhone and iPod Touch devices will be supported:

- iPhone EDGE, 3G, 3GS
- iPod Touch 1st Generation, 2nd Generation

[iPhone_003] The MTPT will support the latest Operation System at the time of this document (3.0).

[iPhone_004] Hole cards will be displayed on the iPhone/iPod Touch devices.

[iPhone_005] The player will have the option to select a color for representation during the game

2.2.1.2 Game

The software needed to run the game will be at minimum a full working simulation of a four player game of No Limit Texas Hold'em. This includes the

use of one complete deck of 52 unique cards, the ability to complete an entire game with one distinct winner, and follows standard convention and rules. These rules that will be followed are outlined in “Robert’s Rules of Poker”[1]. On top of the regular rules, the software will follow personal requirements of chip values and time constraints of blind levels. The TBD requirements that follow the base requirements offer more options to the game, but are not needed to enjoy a full game of Hold’em.

[Game_001] The MTPT will support Tournament style No Limit Texas Hold’em Poker.

[Game_002] The game will support four players.

[Game_003] Blinds will raise every 10 hands as shown in [Table 2-1](#)~~Table 2-1~~.

Table 2-1 Poker Game Blind Structure

Level	Blinds small/big
1	100/200
2	200/400
3	300/600
4	400/500
5 (remove white chips)	500/1000

[Game_004] Each player will start with 10,000 units in chips.

[Game_005] The chip values that will be used will be as follows:

- Blue: 100
- Red: 500
- Black: 1000

2.2.1.3 Table User Interface

The requirements for the table interface dictate how the software shall directly interact with the user. These requirements are influenced on the desire to make the poker game experience seamless as much as possible. The group wishes for a fluid and natural game, as it is experienced with real cards. Furthermore, the interface shall be intuitive and showcase the capabilities of the technology used.

[Table_001] Players will join by touching one of four buttons on the table.

[Table_002] Player and chip locations will be determined which button they touch.

[Table_003] Game will prompt user to start when at least 2 people join.

[Table_004] A user option menu will be available with the following options:

- Start new game
- Pause
- End game

- [Table_005] Chips will be displayed on the table.
- [Table_006] Betting will be executed by touching and dragging appropriate chips.
- [Table_007] Community cards will be displayed on the center of table.
- [Table_008] The interface will use the device name on the iPhone/iPod Touch to identify each player.
- [Table_009] Player names will be displayed in the vicinity of each player's area.
- [Table_010] The interface will display the following game information during the course of the game:
- Current player to act
 - Pot amount
 - Community cards
- [Table_011] The interface will execute the following player actions with touch commands:
- Bet
 - Check
 - Fold
- [Table_012] Players will be notified of blind level raises the hand before the blinds are raised.

2.2.2 Menu

The menu shall offer an interface to view and order items from an actual restaurant. The menu shall feature fictional items with a drink and food menu. Also, game instructions will be made available. The user will have the ability to view and order from the menu. This menu will run in parallel with the main application of the poker game.

2.2.2.1 Table Interface

- [Table_013] The menu can be viewed anytime during the game.
- [Table_014] The menu will contain fictional items.
- [Table_015] Only one menu will be visible.
- [Table_016] The Menu will be closed using the "hide" button.

3.0 Hardware

3.1 Multi-Touch Screen

One of the main components to this project is the multi-touch screen. It allows the user to interact with the poker game and the menu software using multiple fingers. There have only been a few companies that have bought on to this emerging technology one of which is Microsoft. We found that the majority of the devices created today utilize an infrared light source, a camera sensor, and an optical response from a projector or LCD. These elements were used as the building blocks of our multi-touch screen.

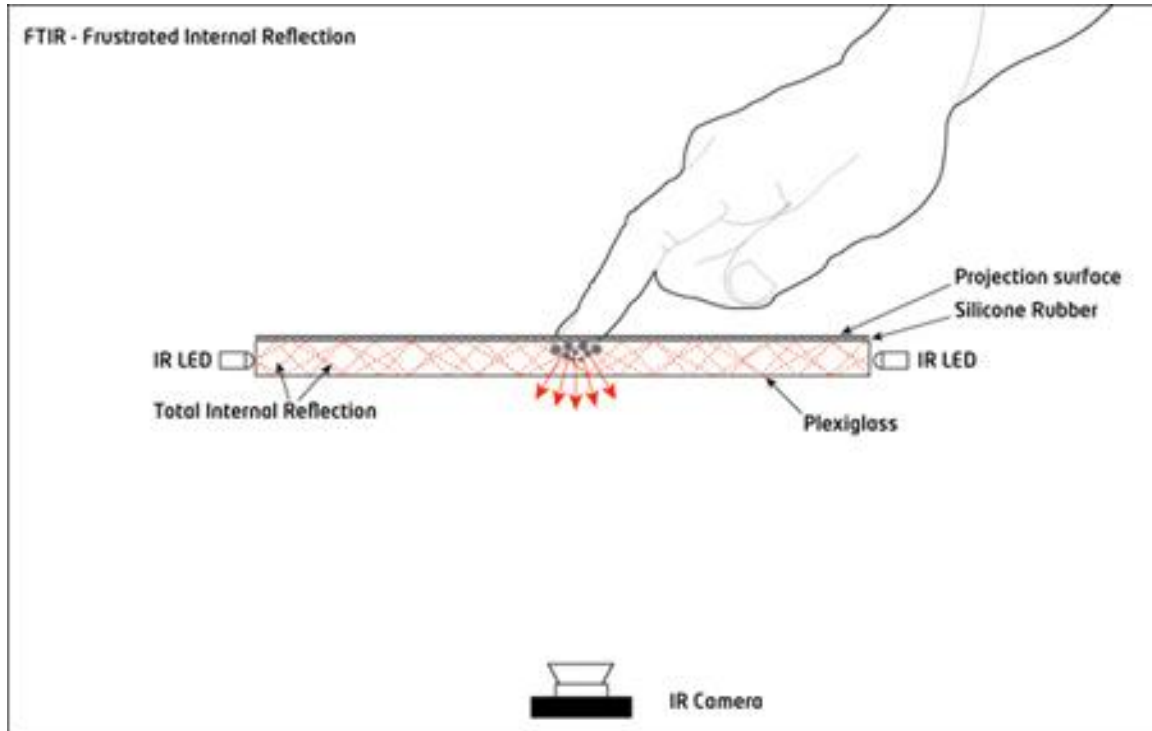
3.1.1.1 Technique

After researching touch screen technology, it was found that there are four main techniques that could possibly be used for our project. They are: Frustrated Total Internal Reflection (FTIR), Diffused Illumination (DI), Laser Light Plane (LLP), and Diffused Surface Illumination (DSI). These techniques all carry their own pros and cons which will be analyzed next.

3.1.1.2 *Frustrated Total Internal Reflection (FTIR)*

Frustrated Total Internal Reflection (FTIR) is a technique that can be implemented with a piece of acrylic panel and some infrared light emitting diodes (LED'S). The sides of the acrylic panel are buffed to a smooth finish and the LED's are incremented at a certain angle along those sides as to flood the panel with infrared light. The light is then reflected through the acrylic and once the user touches their finger to the panel, the light is "frustrated" as seen in [Figure 3-1](#). An infrared camera is used to pick up the light and its position is mapped using software.

Formatted

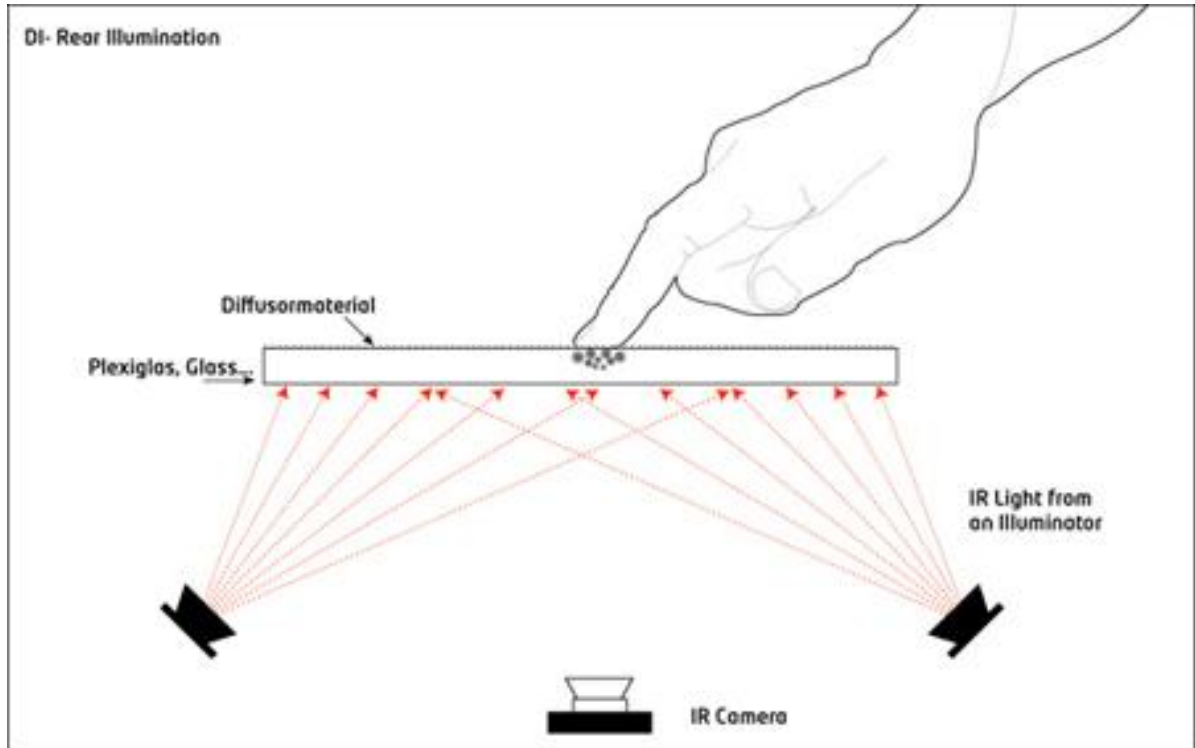


**Figure 3-1 Frustrated Internal Reflection Technique
(reprinted with permission NIU Group [4])**

One problem with the FTIR technique is it has a decreased sensitivity to dragging. The easiest way to improve the sensitivity is to coat the top of the acrylic with some sort of silicon or compliant layer. This presents a more consistent reflection pattern to the camera, thus increasing the sensitivity. It also helps protect the acrylic screen from scratches and light pollution. Products that have worked in previous projects are Sorta Clear 40, Lexel, and various brand of RTV silicone. These products are fairly cheap and easy to work with but also introduce added complexities to the fabrication of the project. Another down side to this technique is that we would have to purchase, wire, and mount about 100 IR LED's around the acrylic sheet. The LED's are only 40 cents apiece but mounting them all would be fairly time consuming. There are LED strips that can be purchased but they are more expensive (\$100).

3.1.1.3 Diffused Illumination (DI)

Diffused Illumination is a technique which can be implemented two different ways, front DI or rear DI. Rear DI uses illuminators to shine IR light onto an overhead acrylic or Plexiglas panel. A diffuser material is bonded to the top or bottom side of the panel to unevenly reflect light. When the user touches the screen, a higher concentration of light is reflected back to the IR camera. [Figure 3-2](#) illustrates the rear DI technique.



**Figure 3-2 Rear Diffused Illumination technique
(reprinted with permission NIU Group [4])**

Front DI uses a similar approach but the IR illuminators are positioned above the acrylic or Plexiglas panel and pointed downward towards the diffuser material and panel. The IR light is then reflected back vertically and when the user touches the screen, a shadow is created and picked up by the IR camera underneath. The rear DI technique is more practical for our project because mounting IR lights above the display would present accessibility problems. Some disadvantages to DI are increased false touch occurrences, lower contrast (needs increased software sensitivity), and illuminators have inconsistent spread of IR light. Advantages are that LEDs are not needed, and an added ability to sense objects. The cost of the illuminators is around \$40 each, making this technique very close in price to the FTIR technique.

3.1.1.4 Laser Light Plane (LLP)

Laser Light Plane (LLP) technique uses lasers to create an IR light plane above the surface. When the IR plane is disturbed, light waves are scattered and the IR camera picks them up (see [Figure 3-3](#)). Most of the designs seen during research have used 4 lasers, one at each corner, with line lenses attached to create a 120 degree light plane. Unfortunately, the lasers have to be positioned about 1mm above the touch surface so the computer tracks a touch

before the user actually touches the screen. This shouldn't present much of a problem unless the lasers are mounted to high and false touches occur. This technique seems to be the easiest to fabricate because an acrylic surface is not needed and IR LED's do not have to be mounted around the screen. A problem with this technique is that lasers have to be positioned very precisely so any bump or movement of the lasers after their set can greatly affect the sensitivity of the touch screen.

Hi Pressure

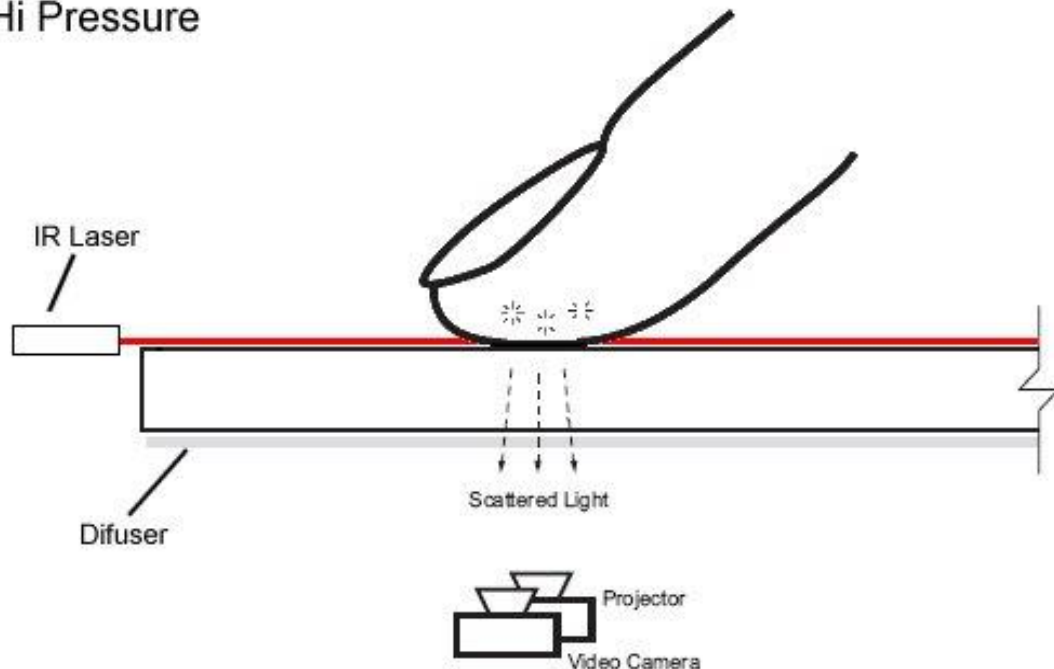


Figure 3-3 Laser Light Plane Technique
(reprinted with permission NIU Group [4])

3.1.1.5 Diffused Surface Illumination (DSI)

Diffused Surface Illumination (DSI) technique is very similar to the FTIR technique discussed earlier. It uses IR LED's, distributed around the touch screen, to inject light into a special acrylic surface. This surface material has small particles inside that defuse the light and act like tiny mirrors. This allows light to be distributed more evenly throughout the surface (see [Figure 3-4](#)[Figure 3-4](#)). When the user touches the material, IR light is then redirected to the camera below. DSI technique also gives an added capability of tracking objects and is pressure sensitive. Some of the disadvantages are that the camera used must be much more sensitive, and the acrylic surface is very expensive. The only company that was found that makes this material is called Evonics and they list it as \$1200 for a 40" x 80" piece. We obviously did not need a piece that size but it still was out of our price range.

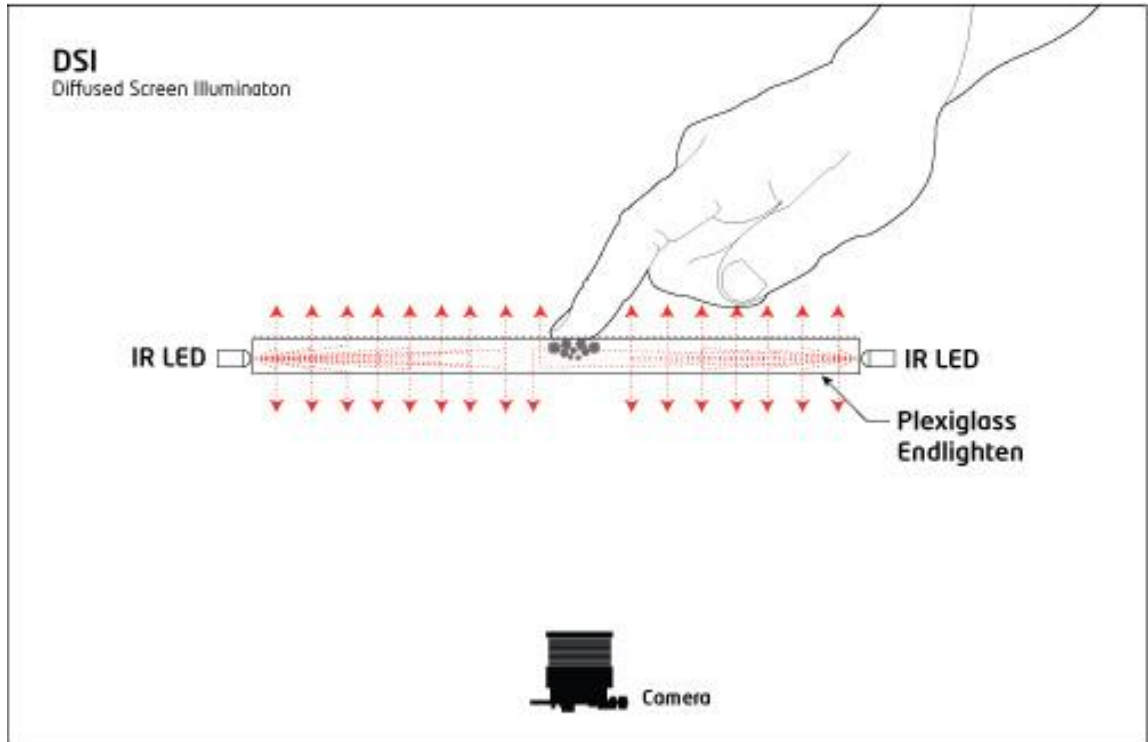


Figure 3-4 Diffused Screen Illumination
(reprinted with permission NIU Group [4])

3.1.1.6 Design

Before the design of the multi-touch screen could begin, our group had to make a choice on what technique to use. We decided to create a decision matrix, seen in [Table 3-1](#), to help in our final decision. The matrix gave a rating for each variable from 1-4 (1 being the best and 4 being the worst). The technique with the lowest score decided the technique that best fit our criteria. Unfortunately, the FTIR and LLP techniques had a tie in score so we decided to pick the FTIR technique. We made this decision due to the fact that the LLP technique uses infrared lasers to detect the users' finger. These lasers can be very hard to properly mount and present a safety risk to eyes. Since both techniques had an equal score, we thought the safer approach would be better.

Table 3-1 Technique Decision Matrix

	FTIR	Rear DI	LLP	DSI
Object Detection	4	1	3	2
Pressure Sensitivity	1	3	4	2
Contrast	1	3	2	4
Setup Time	3	4	1	2
Price	2	1	1	4
Total Score	11	12	11	14

The FTIR technique presents some interesting complexities to the design. The first of these is the screen. The screen materials found while researching were acrylic, Plexiglas and lexar. They all have good internal reflection characteristics for IR light and can be acquired fairly easily at House of Plastics in Orlando or at numerous businesses over the internet. Once acquired the edges needed to be sanded and buffed in order to allow the light from the LED's to pass through the screen. This could also be done at the manufacturer for a price, which is discussed in the cost and procurement section. In order to meet our requirements the screen must be at least 25". This was met with a 4:3 screen ratio.

The last element needed for the design of the screen was a compliant surface on the acrylic sheet. The compliant surface is used as a medium to transfer the users touch input to the acrylic screen. It creates a more consistent reflection of light to the camera. There are two basic ways to add a compliant surface. The first is to get a piece of trace paper that acts as a projection surface. Then, using a paint roller, spread the silicone material evenly on the paper and allow it to cure. Once cured, we placed the paper silicone onto the acrylic surface. The other method is similar but the silicone is spread directly to the acrylic sheet and trace paper is taped on afterwards. We used the first method because it gave us the desired result without potentially ruining the expensive acrylic sheet.

3.1.1.7 Cost and Procurement

The cost of numerous items is represented in [Table 3-2](#). The items purchased are annotated under the procurement status tab in green.

Table 3-2 Multi-Touch Screen Cost

Product Description	Manufacturer/Website	Price \$	Procurement Status
COMPIANT SURFACE			
Sorta Clear, 40 pint Kit		\$36.38	
DAP Silicione, 10.1fl oz.	Lowes	\$4.50	Acquired
GE Silicone, 10.1 fl oz.	Lowes	\$6.00	
Rosco Grey, 20"x24"	Rosebrand	\$6.50	
SCREEN MATERIAL			
Acrylic Sheet, 24"x32"	House of Plastics	\$53.00	Acquired
Buff and Polish edges	House of Plastics	\$10.00	Acquired
Plexiglass, 24"x48"	RPlastics	\$124.00	

3.2 LED's

Infrared LED's must be placed around the sides of the acrylic screen as explained in the previous chapter. These LED's have three main criteria that need be addressed before purchase which are wavelength, angle, and radiant intensity. The wavelength of light in the infrared spectrum is between 780-940 nm. LED's that are on the lower end of that spectrum will increase the sensitivity of the camera because cameras are designed to pick up wavelengths in the visible light spectrum. With increased sensitivity comes the added ability to analyze touch pressure. The radiant intensity of LED's is a direct reflection of their power. The diodes must be powerful enough for their light to reach the opposite side of the screen where positioned. A minimum of 80 mW per diode is needed in order to create enough reflection in the acrylic but a higher intensity will generate an improved contrast for the camera. The last criterion is the LED angle. The diode angle cannot be too high or too much light will escape the acrylic screen and reflect off objects, triggering false touches. If the angle is too small the light will not be adequately reflected internally. After researching similar projects, it was found that the ideal angle is 48 degrees +/- 12. Unfortunately most IR LED's have a smaller viewing angle than what is needed. To bypass this problem the MTPT will use a higher concentration of LED's mounted around the screen. This will adequately flood the acrylic and create high contrast blobs for the camera. Another problem that may arise is the decrease in intensity due to ambient light in the room. This problem will be solved by creating a voltage adjustment knob on the LED power supply which will allow the user to increase or decrease the LED intensity depending on the situation. If the MTPT is not sensing the users touch, then the adjustment knob can be turned up to allow more IR light into the acrylic surface. If the MTPT is so

sensitive that it starts detecting touches above the screen, then the intensity can be turned down to allow less IR light into the screen. The design of this adjustment knob will be addressed in the power section under devices.

3.3.1 Cost and Procurement

The cost of a few LED's is represented in ~~Table 3-3~~ [Table 3-3](#). The 850 nm LED's from Futurelec are the cheapest but are only 3mm in diameter. These would require more LED's to be placed around the screen. This in turn would require more holes to be drilled and more wire to solder. The LED flex strip is probably the best option because the LED's are already wired and only need a 12 volt DC power source. They are also 4 times more expensive than the other LED's. Since the group was trying to create a budget friendly multi-touch table, the flex strip was not used. The 880nm LED's from Digikey seem to be the best fit for the MTPT. They offered a high intensity, larger diameter, and ideal angle.

Table 3-3 LED Cost Comparison Table

LIGHT EMITTING DIODES			
IR LED, 880nm, 100mW, 5mm, 40 degree	Digikey	100 for \$31.68	
IR LED, 850nm, 150mW, 3mm, 40 degree	Futurelec	\$0.28	
IR LED Flex Strip, 78 per/m, 850nm, 1.97w, 120 degree angle	Environmental Lights	\$131.67	
1 ohm ¼ W Resistors	UCF	FREE	

3.3 Intake/Exhaust Fans

3.3.1 Research

The fans were used to help cool the internal components of the table. The computer and projector both have their own fans but if the internal temperature of the table is hot, then the fans will only blow hot air over the heat sinks.. Therefore, one exhaust fan and one intake fan were added to the table. These fans needed to be strong enough to rapidly intake cool outside air and exhaust warm air as the temperature inside the MTPT increased.

Choosing the proper type of fan was essential in keeping the table enclosure cool, while producing minimal noise distraction. The performance of fans is measured by air flow per time period or cubic feet per minute (CFM). This

measurement accounts for size and speed so the bigger the fan the more CFM produced. A larger fan also creates less noise due to a larger size to speed ratio. The bearing type is also a factor in the fan performance. There are two types of bearings in fans which are ball bearing and sleeve bearing. Ball bearing fans are typically more expensive and noisier but last much longer. Sleeve bearing fans are very quiet but can be made very cheaply and last for a short period. The key attribute for this project is quietness. If the fan is too loud it will distract the users and diminish the gaming experience. Therefore a sleeve bearing fan was preferred.

The placement of the fans in the table was crucial to their cooling abilities. If the fans were too close to each other, then they could create an air flow short circuit where the cool air coming in quickly gets directed back out by the exhaust fan. This can also happen if the fans are directly across from each other. They should be staggered so that air can freely circulate from top to bottom. With that said, the exhaust fans were positioned in an area that doesn't blow hot air on the user. This would have created an uncomfortable environment for whoever is sitting by the exhaust. The fans also must be covered by a grill to keep out any objects or material. The grill cover was as thin as possible to allow max airflow. If it is too thick it can decrease the efficiency of the fan and create unwanted noise.

One problem with installing the fans is that ambient light may enter through the fan blades. This can create sensitivity problems with the camera and screen interface because the excess IR light will enter the enclosure and drown out the screen blobs. A solution to this problem was to create baffle openings for the fans which direct any ambient light downward and away from the camera. Unfortunately this will also decrease the airflow in the enclosure. Another solution was to use air ducts that connect to the fans.

The fans must have the ability to sense when the table is getting hot. To solve this problem, a temperature controller was designed and integrated in to the fans power. The temperature switch triggered the fans to turn on when the temperature inside the table reached a set temperature and turn off when the temperature was back within tolerance.

3.3.2 Design

The fans were placed on opposite sides of the table enclosure. The exhaust fan is located on the lower side panel so warm air is not directed toward the user. The intake fan is located on the upper side panel. Both fans have a baffle on the outside of the table to minimize any ambient light intrusion into the MTPT enclosure. ~~Figure 3-5~~ ~~Figure 3-5~~ shows a drawing of the fan and enclosure. The intended airflow is depicted with blue arrows. If the two fans were not able to keep the enclosure below the acceptable temperature then an option to add two additional fans was available.

Formatted

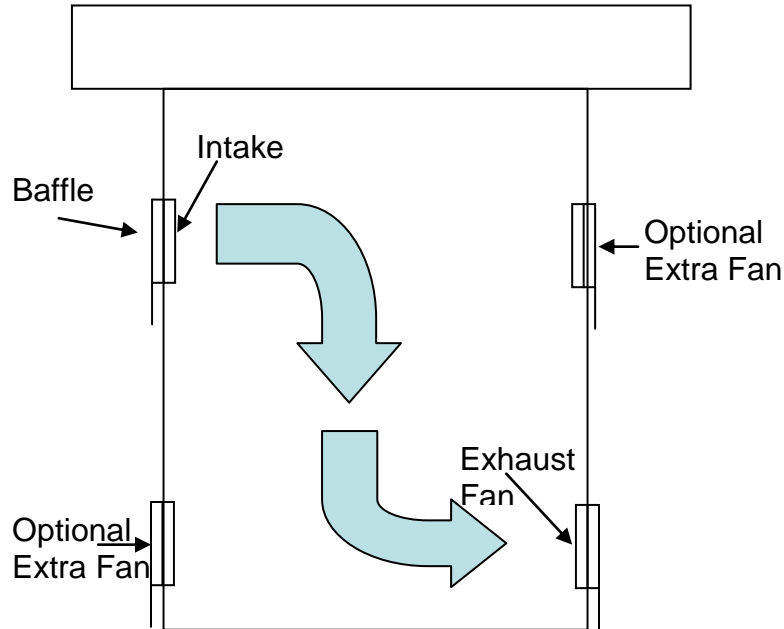


Figure 3-5 Fan Placement Drawing

This temperature switch was built with three components, a N-channel MOSFET, a potentiometer, and a NTC thermister. The MOSFET device must be able to handle the fans voltage and current load. The fans used for this project needed 12 volts and 2.5 watts of power apiece. Most MOSFET devices in the market can operate with voltage up to 40 volts and current up to 10 amps so finding one that meets these specifications was not a problem. The NTC thermister is a device changes resistively with respect to the temperature. The potentiometer allows adjustment of the voltage going to the thermister which in turn adjusts the temperature at which the thermister passes current. A manufacturing requirement for using the NTC thermister is that its contacts must be insulated. This was done with heat wrap tubing. [Figure 3-6](#) shows the circuit diagram of the temperature switch. Some other parts that were needed for this device were electrical tape, solder, and PCB.

To set the turn on temperature for this device, a few steps were taken. First, the enclosure was at an acceptable temperature. If the fans were on then the potentiometer would be turned down until the fans turn off. When the temperature in the enclosure reached the overheating level, the potentiometer was slowly adjusted until the fans turned on. This set the thermister to the overheating temperature and allowed current to flow the fans.

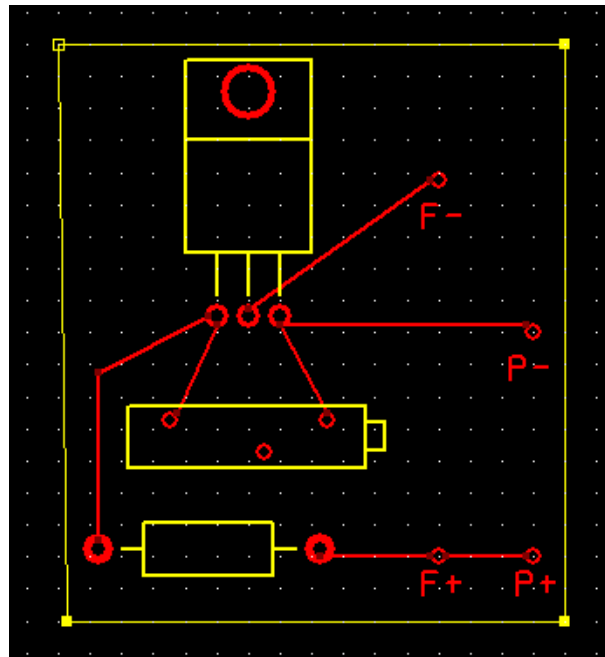
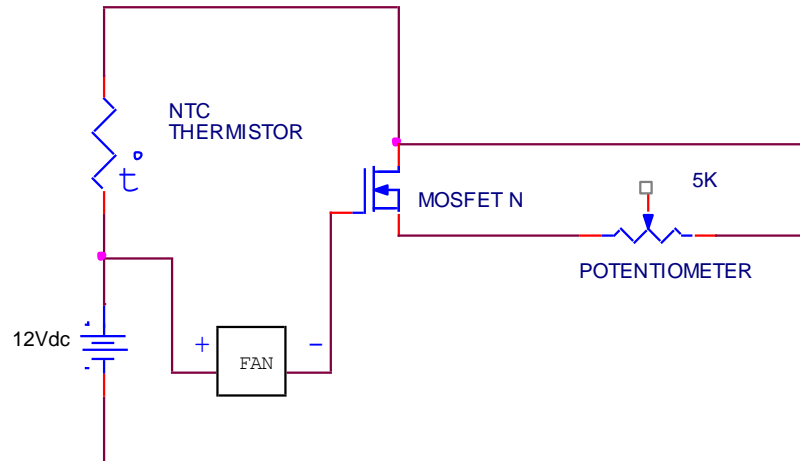


Figure 3-6 Temperature Switch Circuit and PCB

3.3.3 Cost and Procurement

~~Table 3-4~~ ~~Table 3-4~~ shows various fans available in the market. The GlacialTech fan was the cheapest found but has a fairly low speed, high power consumption and low CFM. Its strength is its quietness and price. The Yate Loon fan has a very high speed, high SFM and low power consumption. Its weakness is its noise level. The other two fans were a little overpriced for their specs. The Yate Loon fan was the best deal so it was used in the MTPT.

Table 3-4 Fan Cost Comparison

BRAND	SPECIFICATIONS	STORE	PRICE
GlacialTech Silent Blade II	120mm, 950 rpm, 18.5dBA, 2.4W, Sleeve bearing, 36 CFM	Microban	\$2.99
Zalman ZM-F3	120mm, 900-1800 rpm, Sleeve Bearing, 20-34 dBA	Provantage	\$8.79
Vantec SF12025L	120mm, .8A, .96W, 28 dBA, 53 CFM, 1500rpm, Double Ball Bearing	Radio Shack	\$9.99
Yate Loon	120mm, 1650rpm, 33 dBA, 70 CFM, .3A, Sleeve Bearing	BD Hardware House	\$4.99

3.4 Webcam

The infrared camera is used to display screen touches to the computer. The computer then can analyze the coordinates of the touch and act accordingly.

3.4.1 Research and Design Requirements

There are many types of cameras that can be used in a multi-touch screen. Perhaps the most popular is the web cam. These cameras are most commonly used for internet video but can be altered to function as an IR camera for our project. Some web cams use an external filter to block out IR light and some have an internal filter in the lens. The types that have the filter internal to the lens tend to be more expensive but also offer the highest capabilities. These cameras must be altered by changing the entire lens. This can be difficult because the lens must be designed with the proper focal point which is based on the distance between the lens and the surface, and the dimensions of the surface. Therefore, it would be much easier to acquire a camera that has an external filter. To better choose a camera of this type, there are three main elements that were evaluated. They are resolution, frame rate, and interface.

The resolution of the camera directly relates to the sensitivity of the touch-screen. If the camera has a high pixel per area count the software can be more precise in mapping the correct position of the user's touch. Most web cameras have a resolution of 320x240, which is not quite adequate for the size of our screen. Our project specifications required a resolution of 640x480 or higher.

The frame rate of a camera is measured in frames per second. Obviously, the higher the frame rate the more precise the information will be being recorded. If the frame rate is too low the user will notice a delay in touch response. Therefore, a frame rate of at least 30 FPS was needed in order to allow smooth operation.

The last element that was addressed with IR cameras was the interface. There are typically two types of web camera interfaces between the camera and the computer. The most common is USB. USB devices are compatible with all computers today and also tend to be very cheap. There are a few different versions of USB on the market today. USB 1.0 is the original version and has data transfer speeds of 1.5 Mbit/s. USB 2.0 is the most common and supports speeds up to 480 Mbits/s. USB 3.0 is the newest version which can support speeds up to 4.7 Gbits/s but has yet to be implemented into web cam technology. The less common interface is IEEE1394 or FireWire. FireWire is usually much faster in transferring data but is also more expensive. Also FireWire ports are less common on computers. Our camera required either FireWire or USB 2.0 interface. Either had high enough transfer speeds as not to create delays. Therefore, price determined the interface used in the project.

3.4.2 Cost and Procurement

While researching the cost of cameras, a list of possible candidates and their prices were documented as shown in [Table 3-5](#). One of the problems of researching these cameras is the lack of information on camera lenses. The manufacturers do not state if there is an internal or external IR filter on the lens. If the filter is external then we could take the camera apart and remove it. If it's internal then we would have to order a separate lens which could be difficult to attain. The cameras listed in the table represent ones which have a known external filter. The Playstation 3 Eye had the best capabilities for its price. The only problem we faced is that it isn't designed for a PC but there are drivers that make it compatible.

Table 3-5 IR Camera Options

Product Description	Manufacturer/Website	Price \$	Procurement Status
IR Camera			
Playstation 3 Eye, USB 2.0, 640 x 480 @ 60 fps ; 320 x 240 @ 120 fps	Dell	\$34.99	
Unibrain Fire-I digital camera, IEEE 1394 (FireWire), 640 x 480 @ 30 fps	Office Depot	\$104.99	
Xbox Live Vision Camera, USB 2.0, 640 X 480 @ 30 fps	Newegg.com	\$24.99	

Logitech Quickcam Communicate STX Web camera, USB 2.0, 640 x 480 @ 30 fps	Newegg.com	\$34.99	
---	------------	---------	--

3.5 Wireless Adapter

The wireless adapter allows the poker table software to communicate with the iPhone devices. For this application a wireless router was needed. Wireless routers are most commonly used in networks to provide internet capabilities to multiple computers wirelessly. Internet capabilities for the iPhones via Wi-Fi will be an option for the MTPT depending on the home stations capabilities.

3.6.1 Research

Wireless routers are typically governed by a set of standards called IEEE 802.11. These standards govern wireless local area network (WLAN) computer communication in the 2.4, 3.6 and 5 GHz frequency bands. There are 4 main standards that most wireless devices conform too which include IEEE 802.11a, 802.11b, 802.11g and 802.11n.

IEEE 802.11a transmits in the 5 GHz frequency range and can transmit data at 54 Mbits per second. The iPhone is only compatible with b and g standards so this type of router will not work.

IEEE 802.11b is the most popular standard but has become outdated over the years. It transmits in the 2.4 GHz frequency and has the capability of transmitting 11 Mbits per second. This standard would work fine for the poker game data but if the internet option is used, the connection would be very slow.

IEEE 802.11g transmits in the 2.4 GHz frequency as well but can transmit data at a rate of 54 Mbits per second. This standard uses orthogonal frequency-division multiplexing to split the frequency into small sub-frequencies to pass information.

IEEE 802.11n is the newest standard that can transmit data up to 140 Mbits per second. Unfortunately the iPhone does not support this standard.

The MTPT uses the IEEE 802.11g standard in its wireless router. This standard leaves room for the option of a faster internet connection while easily meeting requirements for poker game data transfer.

3.6.2 Cost and Procurement

The wireless router used for the MTPT was acquired through donation. The specifications are listed below.

Linksys Wireless G Router

Linksys Wireless G Router

- Connectivity Technology:** Wired, Wireless
- Integrated Switch:** 4-port switch
- Data Transfer Rate:** 54 Mbps
- Frequency Band:** 2.4 GHz

- Data Link Protocol:** Ethernet, IEEE 802.11b, IEEE 802.11g, Fast Ethernet
- Switching Protocol:** Ethernet
- Network / Transport Protocol:** TCP/IP, IPX/SPX
- Routing Protocol:** RIP-1 , RIP-2
- Remote Management Protocol:** HTTP
- Communication Mode:** Full-duplex
- Status Indicators:** Power , Link activity

3.6 Projector

The projector is a vital component in the Multi-Touch Poker Table (MTPT). It provides the user with visual feedback during usage.

3.6.1 Research

There are two basic types of projector in the market that can be used with the MTPT: Liquid Crystal Display (LCD) and Digital light Projection (DLP). LCD projectors use an array of tiny lights that change color according to the pattern they are representing. Since each light is controlled, LCD's tend to have a very sharp contrast and vibrant colors. One problem with these projectors is that the lights tend to dim over time. The projector that will be installed in the MTPT will most likely be used so this could potentially create a problem. LCD's also produce what is called "the screen door effect" which tends to make the picture look granular.

The second type of projector is DLP. These projectors use tiny mirrors, mounted on a chip, that move to create the picture. Each mirror is considered a pixel in a very large array. Color is injected into the picture using a color wheel that emits red, blue and green. DLP projectors tend to lack a little in color but have a higher contrast and deeper blacks than LCD.

Both types have their advantages and disadvantages but either would work fine for this project. However, there are some properties that need to be addressed when the projector is picked out. The projector brightness, measured in ANSI lumens, is typically desired to be as high as possible. For the MTPT this is not the case because if the brightness is too high it can cause the user eye problem after prolonged use. The same effect happens when sitting too close to a television.

Another property is the projector's throw distance. All projectors have a minimum and maximum throw distance. The MTPT projector should have a very short throw distance to enable good screen resolution. If the throw distance is longer than the actual path, a mirror can be placed inside the table enclosure to increase the throw length.

The aspect ratio is also an important property. The MTPT is designed to use a 4:3 ratio. This ratio gives all the users ample room around the table. Some projectors have the ability to switch between 16:9 and 4:3 but its important that the MTPT projector has 4:3 at the very least.

The last property that is important for the MTPT projector is resolution. The software that will be used to run the poker game and menu use intense graphics. In order for the user to fully appreciate these graphics, a minimum resolution of 800x600 is needed. A higher resolution would be ideal but the requirement is high enough that screen picture will not be pixilated.

3.6.2 Cost and Procurement

~~Table 3-6~~ ~~Table 3-6~~ shows the specifications of the projector that has been purchased. The Dell 11100MP meets all the requirements set and has a high screen resolution. The brightness is at a moderate level which alleviated potential eye problems. The only problem is the throw distance. The minimum throw distance is larger than the table enclosure so mirrors were purchased and placed in order to lengthen the distance. Fortunately, the projector was acquired used for \$300..

Table 3-6 Dell 11100MP Specifications

Dell 11100MP Projector Specifications	
MSRP (USD) :	\$699
Brightness (Lumens) :	1400 ANSI
Contrast (Full On/Off) :	2100:1
Variable Iris:	No
Audible Noise:	34.0 dB
Eco-Mode:	32.0 dB
Weight:	4.9 lbs
Size (inches) (HxWxD) :	4.0 x 9.9 x 8.4
Std. Lens: Focus:	Manual
Zoom:	Manual, 1.20:1
Throw Dist (feet) :	3.9 - 32.8
Image Size (inches) :	27.5 - 275.0
Optional Lenses:	No
Digital Zoom:	**
Digital Keystone:	Vertical
Lens Shift:	No
Warranty:	2 Years
Performance:	
H-Sync Range:	15.0 - 70.0kHz
V-Sync Range:	5 - 85Hz
Compatibility:	
HDTV:	720p, 1080i, 576i
EDTV/480p:	Yes
SDTV/480i:	Yes
Component Video:	Yes
Video:	Yes
Digital Input:	**
Computers:	Yes
Display: Type:	DLP (1)
Color Wheel Segs:	**
Color Wheel Speed:	**
Native:	800x600 Pixels
Maximum:	1400x1050 Pixels
Aspect Ratio:	4:3 (SVGA)
Lamp: Type:	200W UHP
Life:	2000 hours
Eco-Mode Life:	2500 hours
Quantity:	1
Speakers:	2.0W Mono
Max Power:	250W
Voltage:	100V - 240V

	FCC Class:	B
	Special:	**
	Status:	Out of Production
	First Ship:	Apr 2005
	Last Ship:	May 2006

3.7 Framework

3.7.1 Research

Framework for this project must house each component of the MTPT. It must be big enough to accommodate all the components while still meeting the design requirements of a height less than 40". The table will be used in a restaurant or coffee shop environment so it must be aesthetically pleasing as well. It must contain cup holders for each player in order to decrease the risk of drinks spilling and possibly damaging the hardware. The typical diameter of a cup holder is 3" with a depth of 3". A problem may arise when an awkwardly sized cup is introduced so a replaceable cap should be optional.

The table will be used with standard counter stools so the table height will be slightly taller than a normal dinner table. Counter stools are on average 24"-26" in height and counter tables are normally around 36". In order for the players to be seated and properly see the screen, the chair should be proportionate to these measurements.

The table contains a door that gives access to all the components. The door is easily accessible and sturdy. It also contains a lock to secure all components from theft.

3.7.2 Design

After researching the specific requirements of the table, a general design was constructed. Figure 3-7 show a basic drawing of the table and its dimensions. The table is 32 inches tall and has a maximum width of 36 inches. There is 8 inches of space between the user and the screen, and a holder for the phone to charge. The table base was recessed 7 inches to give the user proper leg room when sitting. One addition to the design was 2 sets of baffled holes on sides for ventilation. Fans were inserted into the holes to circulate air into the box and prevent the components from overheating. The door was located on side 2 with one piano hinge. A locking feature for the door may be positioned opposite the hinge.

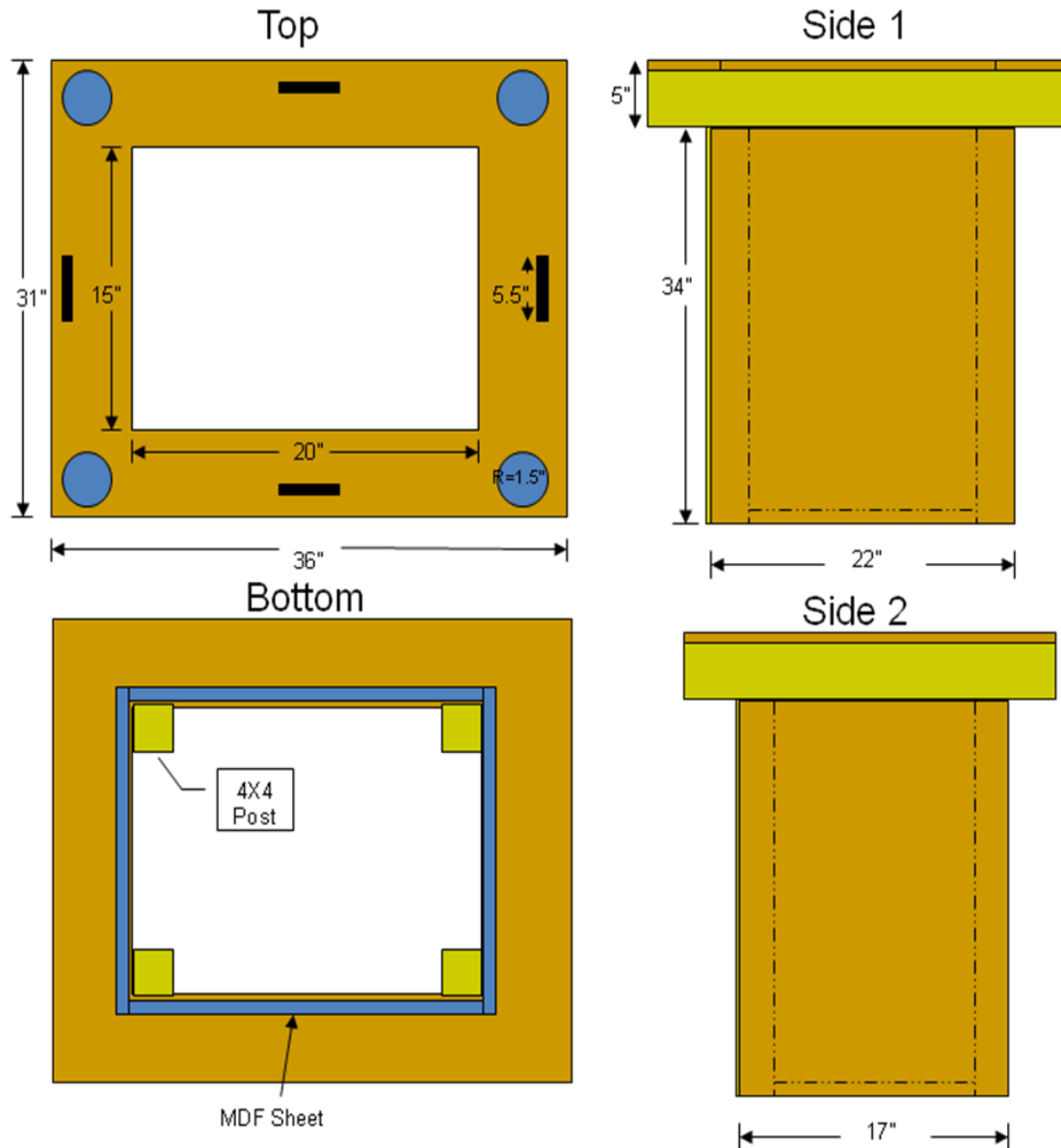


Figure 3-7 Table Framework Drawing

3.7.3 Material and Construction

It was found that there were three basic ways to construct the table: buy a prefabricated storage container and add to it; build the table from scratch; pay someone to build the table. The cost of a prefabricated container with doors was around \$100 and made out of plywood. This technique would still require further fabrication of the actual table portion that houses the screen. Paying a carpenter to build the table would have been expensive but would have saved in assembly time. Building the table from scratch would take a lot of time but allowed for

more customization. These were all feasible options but building from scratch was the most cost efficient technique which was a major factor in the project. There are many different types of materials that could be used in the construction of the table which range from Medium-Density Fiberboard (MDF), plywood, plastic, and wood. These materials used must be attainable, workable, and cost efficient. The table was assembled primarily from wood due to its availability and variety at hardware stores. The outside panels were made of MDF due to its reasonable price. The table was built in a home woodshop with a full set up of woodworking tools.

3.7.4 Cost and Procurement

A list of the materials and tools needed is shown in [Table 3-7](#). These materials were subject to change if any unforeseen complications arose, and as different building materials are experimented with over the course of the table build time. The prices of all the materials were competitive so the store that was most convenient during the build will be chosen.

Table 3-7 Framework Cost

MATERIAL	STORE	PRICE	TOOLS	Store	Price
4x4 wood	Lumber Liquidators	8' \$5.50	Power drill		N/A
	Lowe's	8' \$5.71	Power saw		N/A
	Home Depot	8' \$5.82	Level		N/A
2x4	Lumber Liquidators	8' \$2.36	Sandpaper		N/A
	Lowe's	92" \$2.41	Screwdriver		N/A
	Home Depot	8' \$2.52	Hammer		N/A
MDF sheet	Lowe's	1/2" 47"x99" \$19.25	Table saw		N/A
			paint brush	Lowe's	\$2.00
Screws	Lowe's	\$2.00			
Nails	Lowe's	\$2.00			
Wood glue	Lowe's	\$4.00			
Stain	Lowe's	Free			
Door hinges	Lowe's	\$1.00			
Roll-around wheels	Lowe's	\$5.00			

3.8 Power

The MTPT houses many components that use electrical power to operate. These components required different voltages and current. Some required 110 V AC from the wall socket and the others required 12V DC. All of these components could be powered using the computers power supply and an extension cord. In an effort to gain an increased knowledge in electronics, the power supply for the LED's and table fans were designed and fabricated.

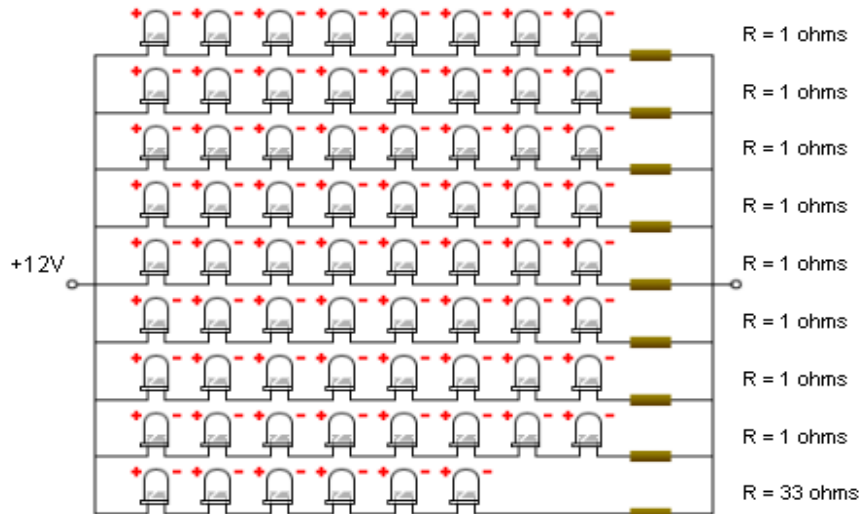
3.8.1 Devices

3.8.1.1 LED's

The LED's that were needed for the MTPT operate with a max forward voltage of 1.5V DC and an intensity of 100mW. There was 70 total IR LED's around the perimeter of the screen which gave 1 LED per inch. There was also a voltage adjustment knob to control the LED's intensity. To control the LED's, a power supply and circuit was fabricated. The power supply took 120V, 60 Hz AC from the wall receptacle and converted it to DC or use DC power from another source.

There are three main types of power supplies that can be used for the LED's. They are battery power supply, linear power supply and switch mode power supply. A Battery power supply uses DC power and would be easily implemented into the MTPT but it would require the owner to constantly change out batteries over time. Since the MTPT will always need power from a grid for the other components, there is no need to use battery power. A switch-mode power supply uses a transistor to switch between saturation and cutoff mode at high frequencies. Since the switching speed of the transistor is much higher than the frequency of the A/C, small transformers can be used to step down the voltage. The biggest advantage to this type of power supply is that it can be made very small and efficient. It also doesn't generate much heat. The disadvantages are that it can create a significant shock hazard, potentially damage the load, and creates noise through the output voltage. Linear power supplies typically refer to a power supply with a transformer, bridge, filter capacitor, and a regulator. The advantages of this type of power supply are it isolates the A/C from the user, making the design somewhat safer. These power supplies can also deliver a very wide range of power from the A/C input. The major disadvantage is that they require more space and produce a lot of heat. Since space is not an issue and there will be intake/exhaust fans inside the table framework for cooling, the disadvantages are not a problem. The MTPG uses a linear power supply to power the LED's because the parts were easily attainable and the device are very safe.

Before the power supply could be designed the power needs for the LED's were found. ~~Figure 3-Figure 3-~~ shows the circuit design for the LED's and all the requirements needed for the power supply from an online LED array calculator.



The wizard says: In solution 0:

- each 1 ohm resistor dissipates 10 mW
- the wizard says the color code for 1 is brown black gold
- the wizard thinks 1/4W resistors are fine for your application
- the 33 ohm resistor dissipates 330 mW
- the wizard says the color code for 33 is orange orange black
- the wizard thinks 1W resistors are needed for your application
- together, all resistors dissipate 410 mW
- together, the diodes dissipate 10500 mW
- total power dissipated by the array is 10910 mW
- the array draws current of 900 mA from the source.

Figure 3-8 LED wizard schematic and Power consumption
Reprinted with permission from <http://led.linear1.org/led.wiz>

There are five components that work together in series to generate usable power to the LED's. ~~Figure 3-Figure 3-~~ shows these components in order as they convert AC to DC.

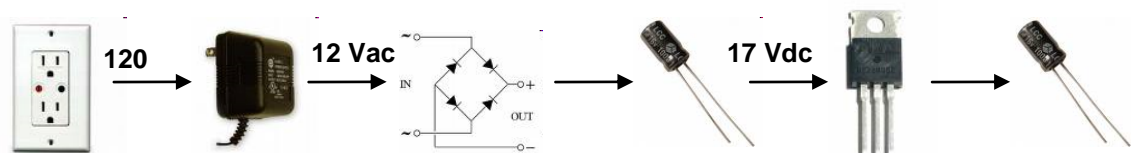


Figure 3-9 Power supply components

With the power consumption known, the design of the power supply started at the transformer. The wall receptacle produces 120 volts A/C that must first be stepped down to a reasonable voltage of 12 volts. This is done using a simple wall transformer like the ones used for most cell phone chargers. Since the total current in the LED circuit is 1000 mA, the transformer must output at least that. Once the voltage is stepped down, it passes through a bridge rectifier that only

allows the positive half of the A/C wave to pass. These can be found at Radio Shack for \$1.50. The resultant alternating positive voltage must be smoothed out using a 1000 uF capacitor. Then a voltage regulator is added to keep the output voltage constant with respect to an internal reference voltage which helps reduce any voltage errors. The voltage regulator will also protect the LED circuit from over-current. They can be found in most electronic stores for \$1. To have the ability to adjust the output voltage, a 5k potentiometer will be added. This may be needed to fine tune the LED performance. When the power supply is turned on and encounters the LED load there will be a drop in output voltage which may cause the LED's to not work properly. By adding a potentiometer, the user can adjust the voltage to a proper range. Lastly, another 1uF capacitor will be added to act as a high pass filter to block and high frequencies coming from the load or the A/C input. An will be an experimental option of adding another 5K pot between the voltage regulator and load. This potentiometer will be used to adjust the resistance with the voltage constant. This in turn should function as an LED intensity adjustor by increasing or decreasing current. Research was done on LED intensity adjustment and it was found that a pulse width modulation device is ideal. Unfortunately the device needed is does not easily combine with a linear power supply so the experimental option was tested.

3.8.1.2 Intake/Exhaust Fans

The MTPT fans needed 12 volts of DC power to operate efficiently. Each fan consumes about 2.5W. Since the LED power supply will only supply a max of 1.2 watts so another option was explored. One option was to increase the current potential in the LED power supply. A problem with this approach was heat. The linear power supply designed for the LED's gives off lots of heat. If the current is sufficiently stepped up the power supply will create more heat through the voltage regulator. Since there is no fan directly mounted on the device, failure could occur. The second option was to use the computers power supply. The computer power supply was designed to handle higher current loads. It is also a switch-mode power supply so it does not create lots of excess heat. ~~Figure 3~~ ~~Figure-3~~ shows a picture of a typical computer power supply connector. The fans were wired to the 12v side of the connector for power.

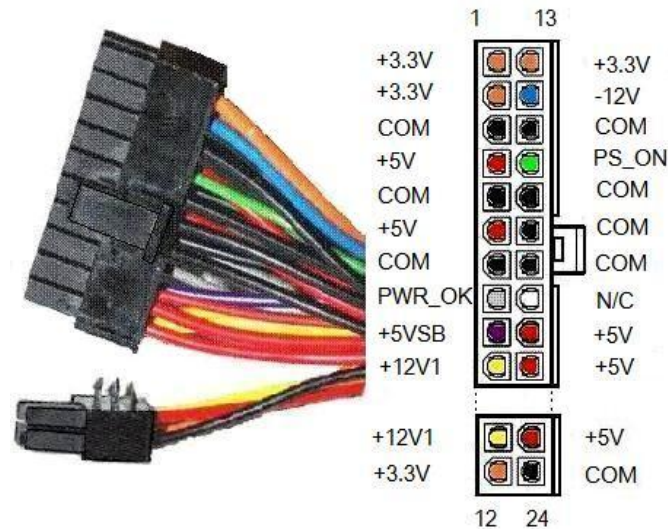


Figure 3-10 Typical ATX power supply connector
 Reprinted with permission from <http://skylab.org/~chugga/mpegbox/>

3.8.1.3 Camera

The IR camera uses a constant supply of DC to power itself. The camera power was outputted from the computer through a USB or Fire-wire cable. USB cables always output 5 volts and up to 2.5W of power. Fire-wire on the other hand had the added capability of producing up to 30 volts and 60 watts of power. For the scope of this project, Fire-wire's power capabilities are not needed because cameras don't typically consume a lot of power.

3.8.1.4 Projector

The projector needed 110V AC power to operate properly. It uses a maximum of 250 watts which is easily attainable from a wall receptacle. Power from the wall receptacle was outputted inside the table enclosure through an extension cord. The extension cord was protected against voltage and current surges with a surge protector. This prevented the MTPT from potential damage due to a lightning strike or power surge.

3.8.1.5 WiFi

The wireless router will also use 110 V AC power to operate. The max current load for this device was around .3 watts. Since the current need for the router is very low, it was plugged into the extension receptacle with the projector. This device was also protected against voltage and current surges through a surge protector.

3.8.1.6 Charging Station

The charging station allows up to 4 users the ability to charge their phone. The connection will be at the base of each phone holder on the top of the table. The iPhone charges with 5 volts from the USB adapter. Four iPhone USB adapters can be used to charge the devices and will connect to the computers USB

connection. To link the computers USB to the iPhone's USB, an extension bank was used which was purchased at Radio Shack for \$6. If connecting all the charging ports into one computer USB port causes any data problems with the computer, an option to wire the charging ports directly to the computers 5V power supply was plausible.

3.8.2 Design

Figure 3-11 shows the LED power supply circuit design. The components were mounted on PCB board and soldered onto place. Figure 3-12 shows the layout of the components and wires inside the table. The power wires inside the table were secured to the floor and walls to reduce accidental connection breaks while moving.

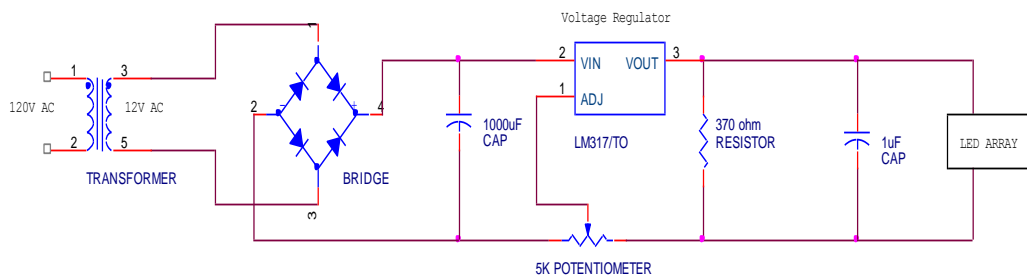


Figure 3-11 LED Power Supply Circuit

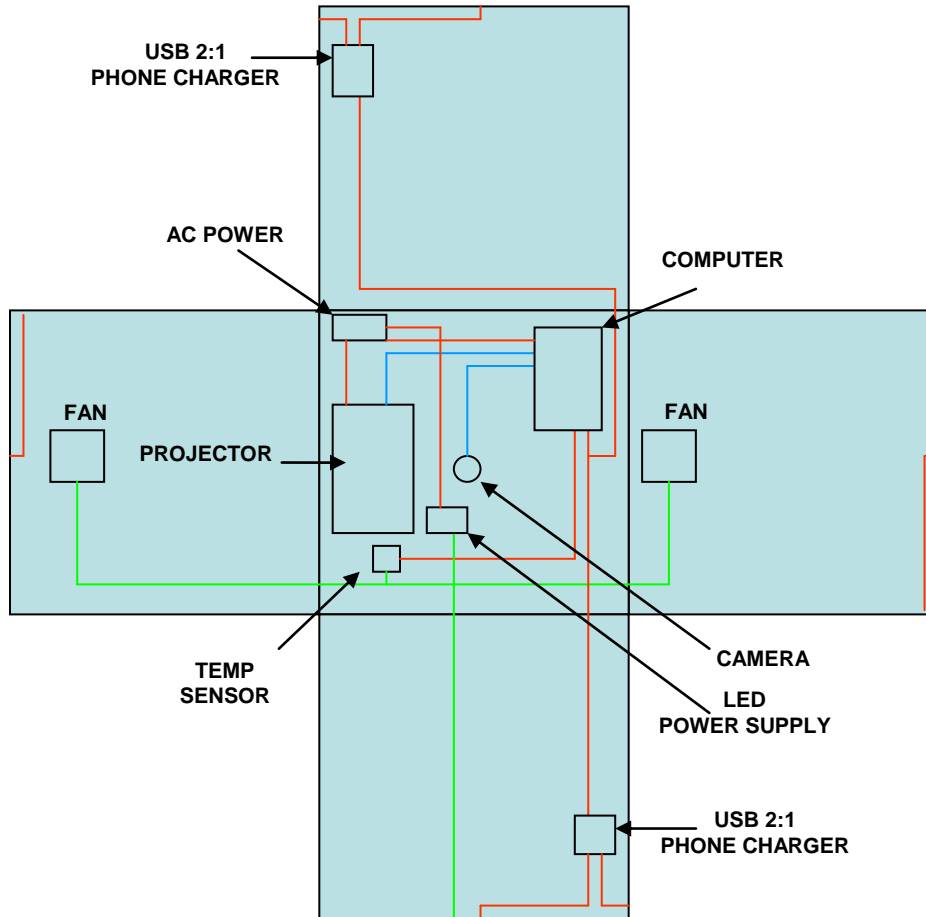


Figure 3-7 Internal Wire and Component layout of Table

3.9 Computer

The computer is the brain of the MTPT. It integrates all the components and gives the programmer complete control over all functions. The requirements state that the computer should be capable of rendering all software graphics and processes without breaks or signs of overload. The hardware requirements of the software programs required at least 2 GB of RAM and a dual core processor. Fortunately, these requirements were met fairly easily due to the rapid pace of computer technology. One of the great features of this table is that the software and computer can easily be upgraded without affecting the functionality of the MTPT. This gave the table a prolonged lifespan in the fast passed world of electronics. The software for the design project was written and

designed to take advantage of the hardware's features in order to maximize quality and performance.

There are three basic options for attaining the MTPT computer. Option 1 is to purchase the computer components and build the computer. This option would allow the group to hand pick the components needed for the computer and not waste money on unneeded parts. It would also add to the project build time. Option 2 is to purchase a preassembled desktop computer that meets the requirements. This option will save time during building and allow the group to focus on other issues. Unfortunately this is the most expensive option and doesn't fit nicely in the group budget.

Option 3 is to find a donated computer that fits the requirements. This option is obviously the cheapest and also allows the group to focus on software and the MTPT build. One problem is that this is not a realistic option for MTPT production. A solution to this could be to allow the buyer to supply their own computer and sell the software package and table. Although there are no plans to produce MTPT's it is still an important design question.

After careful consideration it was decided to go with option 3. This provided room in the budget and allowed for more time in software development.

3.9.1 Specifications

The computer donated has the following specifications:

PROCESSOR

- Intel Core 2 Duo T5500 / 1.66 GHz
- Dual-Core
- Bus speed 667 MHz
- Chipset Mobile Intel 945GM Express

Cache Memory

- L2 cache
- 2.0 MB
- RAM
- 2 GB DDR2 SDRAM – 667 MHz

Environmental Parameters

- Min Operating Temp 32 degrees F
- Max Operating Temp 95 degrees F

Storage

- 80 GB-Serial ATA-150
- CD-RW/DVD 24X

Video

- Intel GMA 900 Dynamic Video Memory Technology 3.0
- RAM size = 224 MB

Networking

- Dell Wireless 1390

- Standards = IEEE 802.11b, IEEE 802.11g
- 1 GB Ethernet link

4.0 Software

4.1 Overview

The Multi Touch Poker Table includes two software components: the Poker Game, interfaced with the iPhone and table, and Restaurant Menu. The Poker Game software is the main application of this project, simulating a full robust game of Texas Hold'em. The game utilizes the iPhone as the player interface. These components are illustrated in [Figure 2-2](#)~~Figure 2-2~~.

4.1.1 Block Diagram

The multi touch table interface communicates the software with a created touch listener. This communication between the selected web cam and the MTPT software components is illustrated in the block diagram in [Figure 2-2](#)~~Figure 2-2~~. The touch listener utilizes the open source library TouchLib to convert the captured image. This image of 'blobs' is converted to useful TouchData that are sent to the MTPT components depending on where and how the user input was executed. The touch listener directly handles the user inputs and passes that event on to the suitable component. The Poker Game software component uses the iPhone as an additional interface along with the multi touch hardware. The iPhone interface communicates with the poker software by exchanging packets over Wi-Fi. The TCP protocol was utilized. Finally, all the software components output their respective graphics back to the hardware using Qt's framework for 2D graphics. All of the poker software with the exception of the iPhone software runs from the selected PC component.

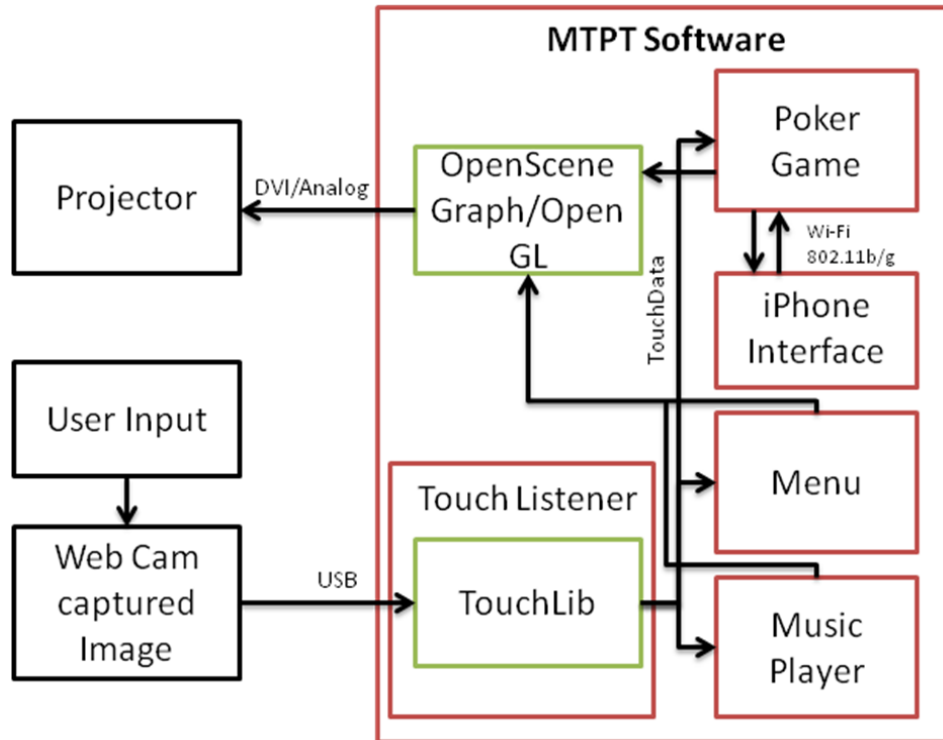


Figure 4-1 MTPT Software Block Diagram

4.1.2 Development Environment

All the software components, excluding the iPhone software, was developed in C++. Microsoft Visual Studio 2005 was the IDE of choice. Additional to the developed software, the libraries are available in C/C++. Visual Studio was chosen for its ease of debugging and vast array of support and features such as Intellesense. Furthermore, most of the libraries that were researched in the following section include Visual Studio build solutions and projects. Software was developed on the group's Intel based personal computers using Microsoft Visual Studio 2005 Express and the Harris Lab at the university.

The iPhone uses a specific set of current technologies to provide the rich user experience that it consistently delivers. Development for the iPhone must be completed on an Apple OSX Machine using Apple's own Integrated Development Environment named Xcode.

The iPhone leverages Apple's extensive Cocoa framework to add identical functionality to the system. The Cocoa Framework is a native object-oriented programming language built for OSX and the iPhone. Cocoa is derived from Apple's roots at NeXT and is a compilation of AppKit and FoundationKit. The iPhone uses a Darwin kernel based upon the Mach3 micro-kernel similar to that used in OSX.

iPhones have ARM-based processors and, while the Darwin Kernel is open-

source, the ARM port is not. This kernel is a Unix derivative and allows many useful, native features to be used. For example, the Memory Management features are very rich and allow the iPhone to run effectively on small amounts of memory or scale extensively for multiple gigabyte systems.

MVC (Model-View-Controller) style design is favored by XCode to accomplish components that do not affecting each other for interface design. This type of interface design isolates the different pieces of the interface into separate parts. The Model manipulates the application's data, the view delivers the results to the end user, and the Controller collects user data input.

Figure 4-2 illustrates the interface design portion of XCode. These View elements may be linked to the Model and Controller aspects using this view.

Formatted



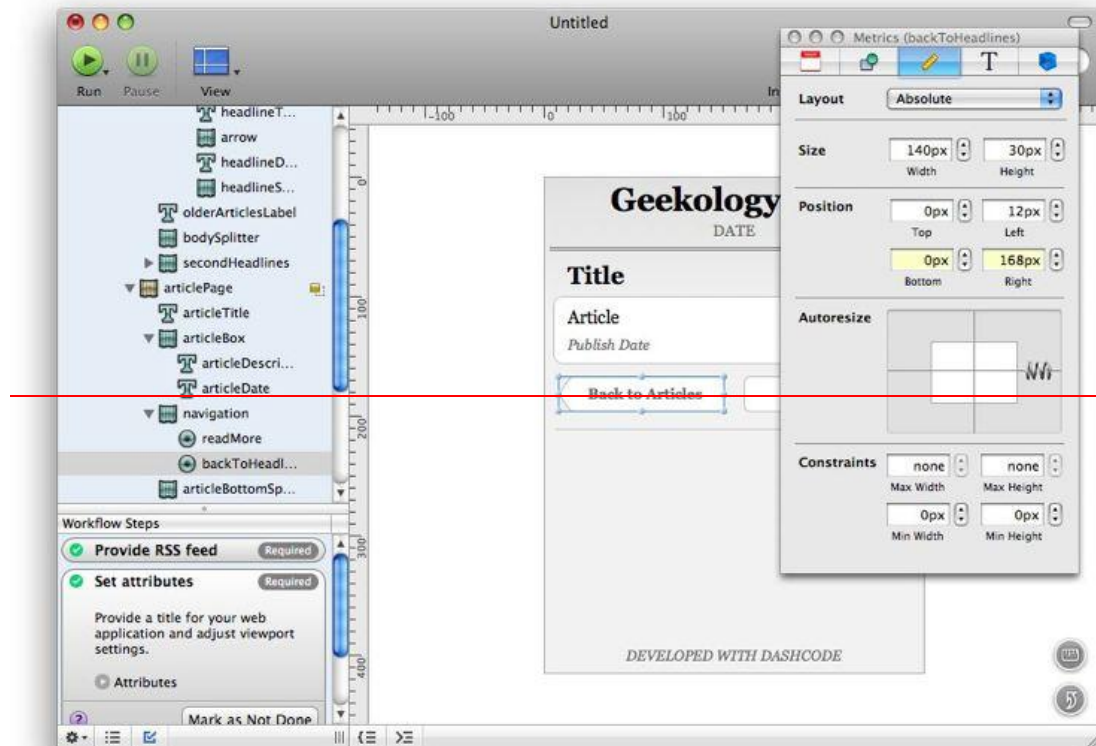


Figure 4-2 XCode Interface

Objective C is the language used by Apple to develop applications for the iPhone. This language is an extension of the C Programming Language with Smalltalk-style messaging. The language is primarily used on OSX and the iPhone; however, the gcc compiler can be used to create generic Objective-C executables.

One benefit of using Objective-C in general is that generic C code may be used anywhere in the application. The Objective-C language is essentially a strict superset layer on top of C with a powerful class library. This allows any C programmer with knowledge of an Object-Oriented language to pick up this language with ease as well.

Objective-C uses messages to get data to and from objects. This means that an object can receive data of any type. There is no type checking for messages that are sent to objects. There is no guarantee that an object can handle the data that is being sent. Furthermore, there is no guarantee that a response was given. When an object cannot handle a piece of data or chooses not to, the returned value is a null pointer.

Some of the basic conventions of Objective-C are as follows. The Interface is defined in the header files (".h"). The Implementation is defined in the method files (".m"). This is similar to C but note the change in extension from ".c" to ".m".

Once an object is designed and implemented, the object may be instantiated. The memory for the object must be allocated first. Then the object must be initialized. Once both of these steps have been accomplished, the object can be used.

Protocols exist in Objective-C. These protocols are used to create interfaces that can be implemented. This is similar to other languages like Java (Abstract classes are similar to informal protocols). Protocols allow multiple, tailored instantiation. Informal protocols may be implemented or may not be implemented. Objective-C 2.0 requires that all formal protocols be implemented fully before a compile can complete.

Objective-C is used with a library or framework in order to provide additional inherent feedback. The library used for the iPhone (as well as OSX) is the Cocoa Framework. This supports the TCP Protocol that was used for connection to the Main Computer. In addition, it provides SVG Graphics Rendering support which was used for the Players Cards as well as OpenGL Graphics support for any possible animations.

Delegates are another useful structure inside of Objective-C. With the use of delegates, methods can be moved and componentized. These methods can be coded ahead of time and easily integrated into a new program. The code is linked together by setting the delegate of this file. This is essentially setting a parent of a child piece of code. For example, the networking stack on the iPhone follows this concept in order to enable error handling as well as run-loops for non-blocking procedure calls.

4.2 Research

4.2.1 Libraries

The MTPT software utilized three open source libraries in assisting with bringing the table to life. The libraries that were used include TouchLib, Qt, and PokerSource. TouchLib was used to handle touch screen events. Qt, brings the convenience and object types of Java to C++. A list of the libraries along with version and license type can be found in [TABLE]. Each library was researched for their APIs and possible future implementation toward the MTPT Software. The following sections outline how TouchLib, Qt, and PokerSource can potentially contribute to the project's software development. Table 4-1 lists the libraries that were used to develop the MTPT software.

Table 4-1 MTPT Support Libraries

Library	Version	License
TouchLib	Beta v2.0	New BSD
Qt	4.5.2	LGPL
PokerSource	137.0	GNU General Public License

		V3
--	--	----

4.2.2 *TouchLib*

TouchLib is an open source library for detecting blobs created on multi touch screens. The library tracks the infrared light captured by the webcam and creates “touch events”, such as finger down and finger released. This library handled detecting the touch events and the created software components handle these events. TouchLib was the chosen multi-touch library for a number of reasons. First, the library is written in the language the MTPT software was written in, C++. Secondly, the license was a new BSD License, allowing us to freely use the library. Finally, source code was provided for the library and sample demos, assisting in the learning process.

TouchLib includes a configuration application to calibrate the multi-touch setup with the chosen webcam and multi-touch screen. In addition to the configuration application, the library includes three sample demo applications that showcase the touch screen capabilities of the library. Also, source code and Visual Studio projects are included, assisting in the learning process.

The library incorporates OpenCV, a framework used to manipulate images and video. OpenCV or Open Source Computer Vision is simply a library for programming real time computer vision[3]. Its implementation include face recognition, motion tracking, and gesture recognition. TouchLib uses OpenCV in processing the blobs detected by fingers or objects and converting those events through filter techniques into useful events a software engineer may need.

TouchLib provides two useful interface classes: ITouchListener and ITouchScreen. The ITouchListener interface provides functions to handle the touch events. These touch events include: finger down, finger update, and finger up. The parameter in all three virtual functions is the TouchData of that touch instance. TouchData members are listed in [Table 4-3](#)Table 4-3[3].

4.2.3 *Qt*

Qt is an open source, cross platform, development C++ library. The approach of Qt is “write once, compile anywhere”, allowing the group to compile the final application regardless of which platform they are running[5]. Qt includes a vast, extensive C++ class library for multiple uses such as multithreading, GUI, and 2D graphic development. This library is being used for its excellent documentation, support, and ease of use. The framework is similar to having a full Java library with documentation in C++. Qt’s fully object-oriented approach makes integration with the MTPT software as simple as possible. Once integrated with

Visual Studio, the developer can take advantage of the Visual Studio debugger and Intellisense.

Qt was used in handling the multithreaded tasks that the MTPT needed to support. Classes such as QThread, allows the group to develop the application with ease. With the provided documentation, examples with source, and full online class documentation, software development, along with learning, was a simpler process. This learning was enhanced by Qt's examples. Furthermore, Qt is currently used and supported in various companies such as Adobe, Boeing, Google, and Skype[5].

Along with threading, Qt provided class functionality in networking and graphics. These classes proved useful in the development of the MTPT software components. QTcpServer, for example, provides functions to set a port to listen to and will notify the application when a connection is available. Qt implementation for each class in the MTPT Software Components is described in detail in the Detailed Design Section. Qt provides support for 2D and 3D graphics. In this project, 2D graphics were only be used per time constraint. Qt's graphics classes support for scaleable vector graphics. The library provides a powerful 2D graphics canvas called Qt Graphics View. It is a surface for creating and interacting a number of graphic items together. Graphics View uses a Binary Space Partitioning tree to discover items at a fast rate. This results in creating visuals on screens in real time [5]. A summary table of the planned implementation of specific Qt classes, data structures, and data types are listed and accompanied by Qt's descriptions in [Table 4-2](#) and 4-3[5].

A very useful functionality of Qt in this project was the utilization of slots and signals. Slots and signals is Qt's alternate approach to previous implementations of callbacks. Qt authors state that callbacks, used in other frameworks, have two fundamental flaws: not type-safe and the callback is coupled to the processing function since it must know which callback to call[3]. Qt's slots and signals are alternatively used to connect to two object events or functions. When a certain event occurs, such as a user click, a signal is emitted. This signal can be connected to a slot, which is a function of an object. This slot is executed every time that particular signal is emitted. In order to connect signalA emitted from objectA to slotB in object, the developer calls the connect() function as follows: `connect(objectA, SIGNAL(signalA()),object,SLOT(slotB()))`. Furthermore, parameters may be passed between the signal and slot. Qt's classes already have built in signals and slots, but developers may add their own for application use as long as the class inherits from QObject. Also, multi signals can be connected to one slot.

Table 4-2 Planned Qt Implementation

Name	Description
QThread	Represents a separate thread within an application. Thread is spawned by calling QThread::start()
QTcpServer	Provides a TCP-based server. Class used to listen

	and accept connections
QTcpSocket	Provides a TCP socket. Allows the sending and receiving of TCP packets
QMap<Key,T>	Template class that provides a skip-list based dictionary. Provides functions to add, remove, and lookup contents.
QVector<T>	Provides a dynamic array
QString	Provides a simpler approach to handle strings. Possible to convert to standard char*, append, and further manipulate.
QDataStream	Provides serialization of binary data that is independent of operating system, CPU, or endianness.
QTimer	Provides a timeout() signal that can be selected to slots. Fires at a given rate.
QGraphicsScene	Provides a surface for managing a large number of 2D graphical items. Container for QGraphicsItems.
QGraphicsItem	Provides foundation to write custom graphic items.
QGraphicsView	Provides a widget for displaying the contents of a QGraphicsScene.

Table 4-3 TouchData

Type	Name	Properties
Int	ID	Unique identifier blob
Float	tagID	Unique identifier fiducial
Float	X	Horizontal position
Float	Y	Vertical position
Float	height	Height of fiducial
Float	width	Width of fiducial
Float	angle	Rotation of fiducial
Float	area	Size of blob
Float	dX	Delta movement in horizontal axis
Float	dY	Delta movement in vertical axis

4.2.1 PokerSource Poker-Eval Evaluator

The PokerSource Poker-Eval library includes support for the game in mind, Hold'em, which uses seven cards to make the best hand. As described in [reference holdem section], the player's two hole cards have to be combined with the five community cards to create the best hand. A library to support seven card evaluations is very important, as most libraries found only supported standard five card hands. PokerSource is completely written in C and includes Visual Studio projects to compile, supporting our development environment and language.

The library evaluates hands by assigning an arbitrary number to the hand. A higher value means a stronger hand. Cards and hands are represented in 52 bits. Each bit represents one of the 52 cards in a deck. These 52 bits can be represented as a 64-bit integer. The card or hand mask is broken down in [Figure 4-3](#)[Figure 4-3](#)[2]. Unused bits are represented with x's.

AKQJ T987 6543 2XXX AKQJ T987 6543 2XXX AKQJ T987 6543 2XXX AKQJ T987 6543 2XXX



Figure 4-3 PokerSource Card Mask Format

For example, a full house with kings full of deuces can be represented with the following. Kings are highlighted in blue, while the two deuces are highlighted in red:

0100 0000 0000 0000 0100 0000 0000 0000 0100 0000 0000 1000 0000 0000 0000
0100

For a single card representation, such as the ace of spades, one bit is used:

1000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0000

Using PokerSource's representation of cards, all 52 cards can be uniquely represented as one number. In order to combine cards to make hands, an OR operation was applied. Furthermore, each of the 52 cards were assigned to the SVG that is to represent that card on the interface. This library assisted in cutting time and effort in the mathematical calculations, but leaves enough room for the group to write the game code that controls and distributes the cards. With the open-source distributable, an example project using the library was found at Coding The Wheel[1]. This example helped in the learning process on how to properly implement the library. The following section outlines the plan on how the group implemented PokerSource.

4.2.1.1 Implementation

PokerSource did not supply the Hold'em game, but provide the data types and functions needed for the MTPT software. These types and methods were implemented in the classes for the Poker Game Software. The important data types that were used for the development software are illustrated in [Table 4-4](#)[Table 4-4](#). Functions that were implemented are described in [Table 4-5](#)[Table 4-5](#). A deck of cards can be represented as an array of StdDeck_CardMask's

with size 52. Shuffling of the deck can be simulated by retrieving one of the indexes(0 to 51) randomly and setting that card mask to 0.

Table 4-4 PokerSource Data Types

Name	Purpose	Notes
Index	A number between 0 and 51. Uniquely identifies a card.	0 = Two of Hearts 51 = Ace of Spades Type: int
StdDeck_Rank_%n%	Ranks cards 2 to Ace with single value	n = 0...9,TEN,JACK,QUEEN,KING,ACE
StdDeck_Suit_%n%	Gives a value to suits	HEARTS = 0 DIAMONDS = 1 CLUBS = 2 SPADES = 3
HandVal	Strength of a given hand represented as a value	32 bits Methods to derive are hidden behind the evaluator, unneeded by program software
StdDeck_CardMask	Standard Card Mask	64 bits Little Endian format used as shown in Figure 4-3

Table 4-5 PokerSource Functions

StdDeck_Mask(int index)	
Description	Computes card mask for given card
Arguments	Index: 0 = 2 of hearts, 51 = Ace of Spades
Return Type	StdDeck_CardMask
StdDeck_Rank(int index)	
Description	Returns rank of card(2 thru 13)
Arguments	Index: 0 = 2 of hearts, 51 = Ace of Spades
Return Type	Int
StdDeck_CardMask_OR(StdDeck_CardMask result, StdDeck_Card op1, StdDeck_CardMas op2)	
Description	Combines two masks into one. Method to create hands from cards.
Arguments	result – mask to hold result of bit OR of op1 and op2 masks
Return Type	StdDeck_CardMask
StdDeck_StdRules_EVAL_N(StdDeck_CardMask m, int N)	
Description	Returns the hand value of the given mask of N cards
Arguments	M – hand mask of N cards
Return Type	HandVal

4.3 Detailed Design

4.3.1 Poker Game

4.3.1.1 Overview

The Poker Game Software component is the core component of this project. This software includes the game software, the source needed to communicate to the four iPhones, as well as the graphics library used to display the game on the MTPT. In this section, the software design is explained. Furthermore, class diagrams and explanations will be illustrated along with various diagrams needed to understand the software design. The software utilizes multiple threads to run. First, there was a communication thread. This thread was used to connect the four iPhones. This thread maintained the client/server functions of the software. To go deeper into, this thread handles incoming and outgoing packets to the iPhone using the TCP protocol. Next, a touch listener was used to detect touches on the MTPT. This listener communicated with the main Poker game executive and update with the appropriate user inputs.

The Poker Game software utilized Qt's open source framework as well as the PokerSource library. Qt types and classes will be explained in each section, as well as the planned intent on their use. Poker Game is planned to be developed in C++. In this section, the game to be developed, No Limit Texas Hold'em will be discussed. The rules and basic game understanding will be explained. Next, the software design will be discussed through class diagrams and explanations. These designs evolved as coding and unit testing progresses. Furthermore, this design reflects the requirements established in section [2.2.12.2.4](#).

4.3.1.2 No Limit Texas Hold'em

There are numerous variations of the card game poker, but the Multi Touch Table specifically showcases the most widely played game, No Limit Texas Hold'em. To understand the software, a basic understating of the implemented game will need to be explained. In Texas Hold'em, players combine two personal cards called 'hole cards' with five community cards to make the best poker hand. 'No Limit' refers to the fact a person can bet up to any amount and can put all their chips in anytime. A simple event diagram of Texas Hold'em is illustrated in [Figure 4-4](#)~~Figure 4-4~~. 'No Limit' in this type of game refers to rule that a player, at any time, place their remaining chips in the pot and go 'all in'. In a game of Texas Hold'em, each player is dealt two cards faced down. A round of

betting occurs. Next, three community cards are displayed on the table. This is called the 'flop'. A second round of betting occurs. Next, another community card is dealt. This is called the 'turn'. A third round of betting occurs. Finally, the last community card, the 'river', is dealt. A final round of betting occurs and each player shows their hand[1]. A table showing poker hand ranks can be found in [Table 4-6](#) [7]. These rankings are illustrated from highest to lowest. In the event two or more players have the same hand, the player with the highest card wins. For example, if one player has a 3 to 7 straight, a player with a 4 to 8 straight wins. Also, for full houses the player with the highest three of a kind in the hand wins. Finally, the lowest straight a person may have is ace to five, while the highest is ten to ace; the ace can only be the lowest or highest card in the straight.

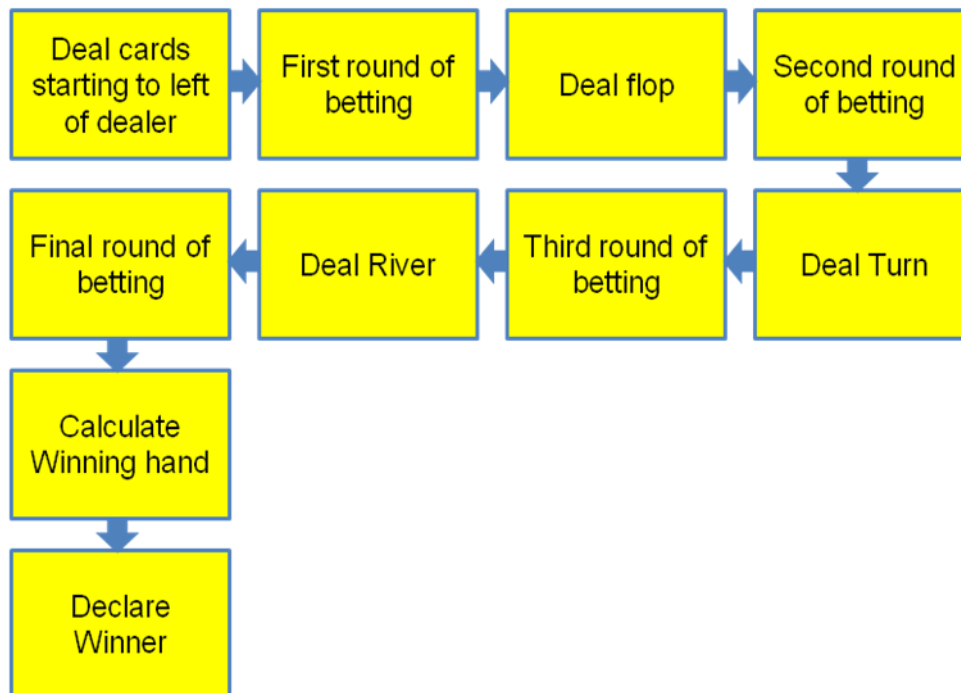

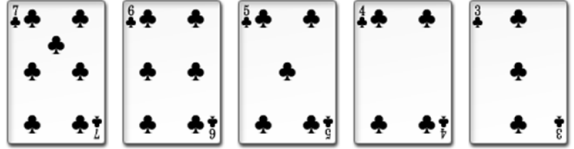
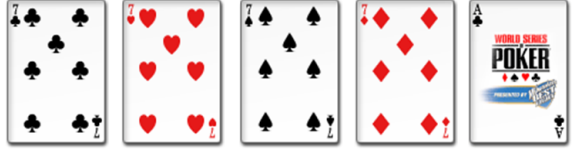


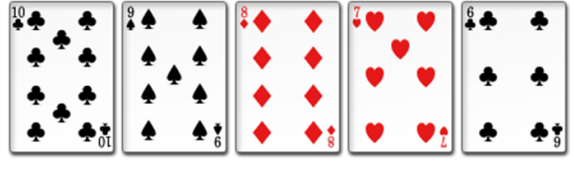



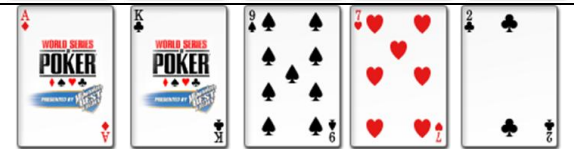


Figure 4-4 Texas Hold'em Event Diagram

Table 4-6 Poker Hand Rankings

Name	Description	Illustration
Royal Flush	Ace high straight flush	
Straight Flush	Straight with same suit	
Four of a Kind	Four cards of same value	
Full House	Three cards of one value and a pair of another	
Flush	Any five cards of the same suit	
Straight	Five cards of sequential rank (lowest: Ace to Five, highest: 10 to Ace)	
Three of a Kind	Three cards of same value	
Two Pair	Two cards of same value and another two of same value	
One Pair	Two cards of same value	
High Card	Lowest: two, highest: Ace	

4.3.1.3 PokerExec Class

The PokerExec class handles the main poker game functions needed to simulate a four player game of Texas Hold'em. PokerExec is responsible for initializing each player, the 52 card deck, and executing the functions in the Texas Hold'em sequence illustrated in [Figure 4-5](#)~~Figure 4-5~~. This figure outlines the timeline used for the executive. First, during initialization, all threads are created. The threads are then opened to listen for touches and connections from the user to join the game. Once a player joins the game, a PokerPlayer instance was created using that user's device name. The executive continues with the starting the game once two people join and wish to start the Texas Hold'em game. The PokerExec assigns positions to each player as explained in Section 4.3.1.2. Next, while utilizing the implemented touch listener and PokerServer, the PokerExec class runs the poker game set in [Figure 4-4](#)~~Figure 4-4~~. As a player loses all their respective chips and gets eliminated, the PokerExec class destroys that instance. The PokerExec terminates once the winner is declared. The PokerExec executes the methods to calculate the data that be translated to a graphical interface.

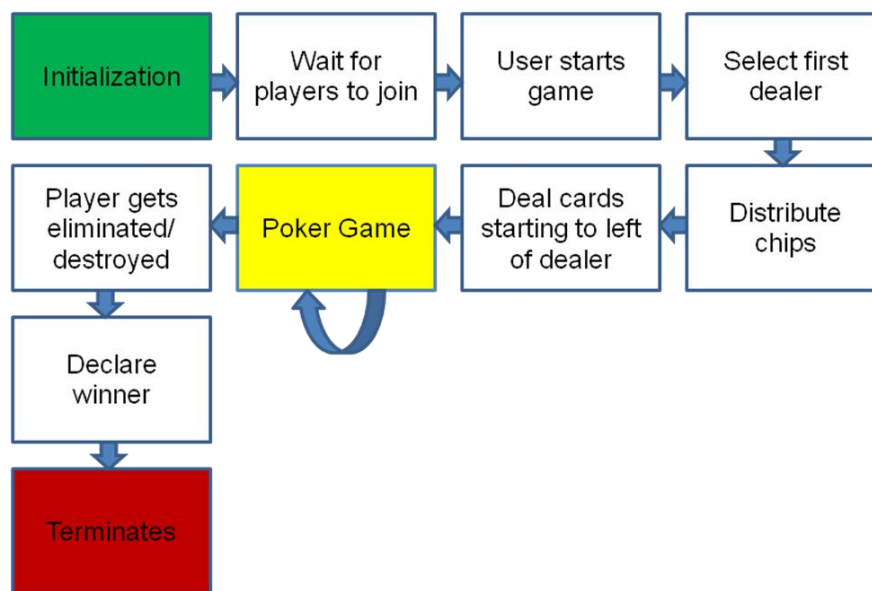


Figure 4-5 Poker Game Exec Event Diagram

PokerExec will have functions to execute a typical hand in Texas Hold'em as illustrated in [Figure 4-4](#)~~Figure 4-4~~. The class has a 'deck' of cards represented as a QQueue of StdDeck_CardMask's. Using a QVector data structure, enables the class to use the QQueue::insert() and QQueue::remove() functions to add and remove cards to the deck. In a four player game of Hold'em, only a maximum of 13 cards be needed to play a hand. This excludes the cards that are 'burned' during a hand. In a real game, the dealer discards one card before the flop, turn, and river are shown. Two cards per player are used for hole cards, three for the flop, one for the turn, and one for the river. Rather than iterating 52 random cards, only 13 were inserted in the QQueue before each hand. The

QQueue deck is cleared with the QQueue::clear() function. The activity diagram illustrating the approach to shuffle or initialize the deck is shown in

[Figure 4-6](#)~~Figure 4-6~~.

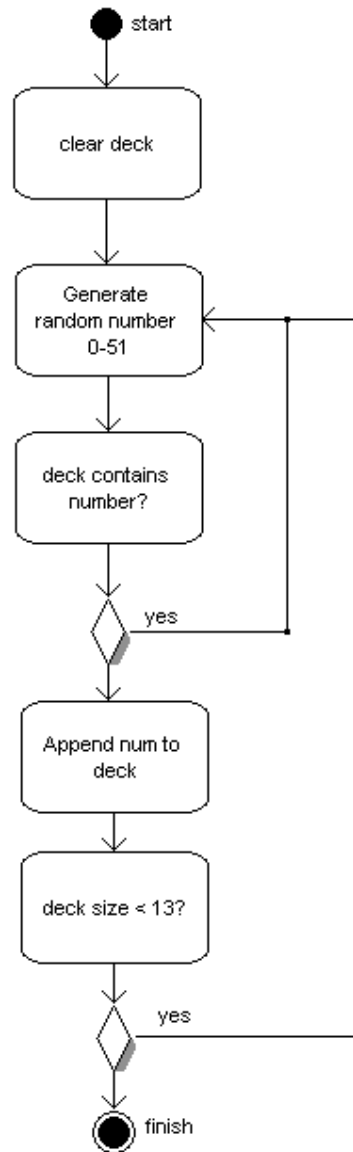


Figure 4-6 Deck Activity Diagram

In order to deal the cards to the players, PokerExec dequeues cards and distribute to the appropriate PokerPlayer. The PokerExec class notifies the PokerPlayer class data is ready to be sent through the TCP port by setting an appropriate boolean inside the PokerPlayer thread to true. This latches the event to send data over the TCP port inside the PokerPlayer thread only when PokerExec commands it to. Using this approach, all data that is sent over the TCP port is sent within each PokerPlayer thread. This ensures that the port is

safely read and written from. The sequence diagram to show TCP control between the Poker classes is shown in **Error! Reference source not found.**

PokerPlayers are stored using a QMap<key,type>. The key is an int and the type is a PokerPlayer class. The key is used to store the player's position and is used to iterate through the poker players during each hand. PokerPlayers are created in the AddPokerPlayer() function. This SLOT is connected to the newConnection() signal of the pokerServer. After the PokerPlayer instance is created, PokerExec starts the PokerPlayer thread by calling start(). Planned implemented QMap functions are listed in

Table 4-7 QMap Implementation

Function	Description
void clear()	Used to clear PokerPlayers after each hand to reset order.
insert(int position, PokerPlayer player)	Used to rearrange players after each round and the beginning of the game
remove(int position)	Used to remove a player if they fold

The PokerExec class utilizes the Pokersource API for its game functions. The CalculateWinner () function returns the key of the PokerPlayer with the highest hand value. Pokersource's function StdDeck_StdRules_EVAL_N() function be used to calculate hand values of each hand. An approach was implemented to find the best possible 5 card hand value of the combination of the player's two hole cards and five community cards. Using the noted static equation for combinations, there are 21 combinations of possible hands between one player's two hole cards and the five community cards. To calculate a player's hand, PokerExec iterates through all possible 21 handmasks using the player's 2 hole cards with the 5 community cards. The highest value is tracked during this process.

$${}_nC_r = n!/(r!(n-r)!),$$

where $r = 5$ card hand, $n = 7 = 2$ hole cards + 3 community cards

Qt's signals and slots functions are used within PokerExec to connect the PokerPlayer instances and the PokerServer class to PokerExec. Signals may be added to any class that implements the QObject class. Signals are connected to slots of two objects in a fashion that when a signal is emitted the associated slot function is called. This strategy is implemented between the three Poker Game classes. To start, when a new connection is available, PokerServer emits the newConnection() signal. This signal is inherited by QTcpServer. Once the signal is emitted, PokerExec is notified of the new connection, the new iPhone. A PokerPlayer is created using the socket number, utilizing a useful feature of Qt's slots and signals. When a signal is emitted with a parameter that parameter must match the connected slot. The parameter is then passed between signal

and slot. PokerExec is notified of player actions as well as which player is responsible for those actions. The PokerExec class is notified of betting, folding, and other actions. After receiving these signals, PokerExec can update the game critical values such as the current pot size and active players. The PokerExec class is illustrated in [Figure 4-7](#). These connections are illustrated in [Figure 4-8](#).

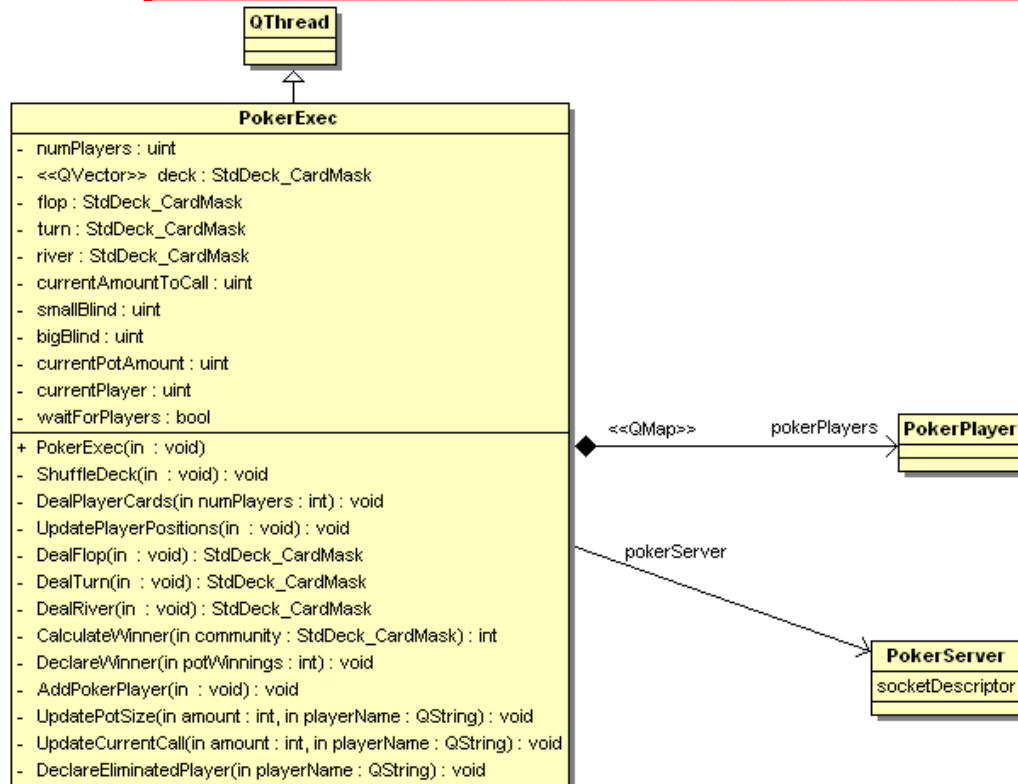


Figure 4-7 PokerExec Class Diagram

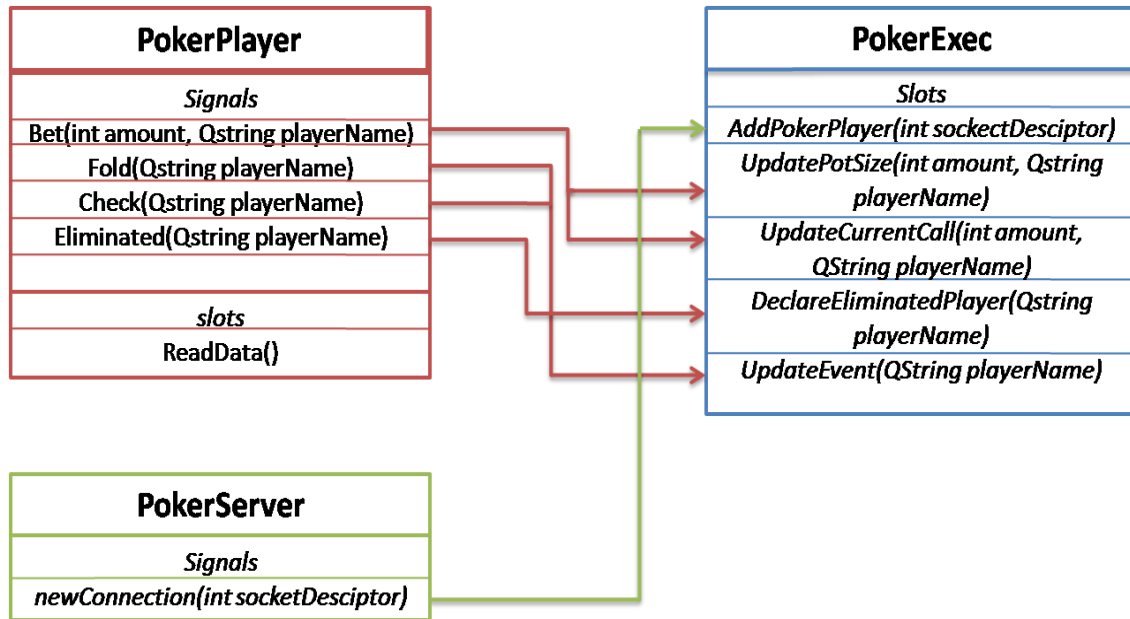


Figure 4-8 Poker Game SIGNAL/SLOT Connections

Handling all functions for a typical game, **PokerExec** is responsible for ensuring a proper betting round. During a round of betting between two or more players, each player may have to choice to bet (raise), fold, or check. In order for a round to continue all players involved must have put in the same amount in that round into the pot. During the entire game, a player is designated as the dealer for position purposes only. The player to the left of the dealer receives the first card. Two automatic blinds, or forced bets, are enforced in Hold'em instead of using antes. The person to the left of the dealer is ruled to post the small blind amount and the person to the left of that player is required to make the big blind bet. Blinds ensure that there is an active pot in each hand. Blinds reflect the requirements set in section [2.2.12-2.4](#). In the first round of betting mentioned in section [4.3.1.24-3.1.2](#), players must match the big blind to see the flop. The first person to act is the person to the left of the big blind. If that player or any player there afterward would like to stay in the hand, they must 'call' or match the big blind. If everyone calls the big blind and action ends at the person who already has the big blind, they have the option to 'check', stating that they opt out to re-raise, because they already have matched the current pot bet. A player can check only if they already have money that is equivalent to the current bet. If there is no current bet, a player may check because the current pot is equal to the money that player has in, which is nothing. An activity diagram on bet handing is illustrated in [Figure 4-9](#).

three pots are made. Side pots are created in the event a player's bet is equal to the player chip count.

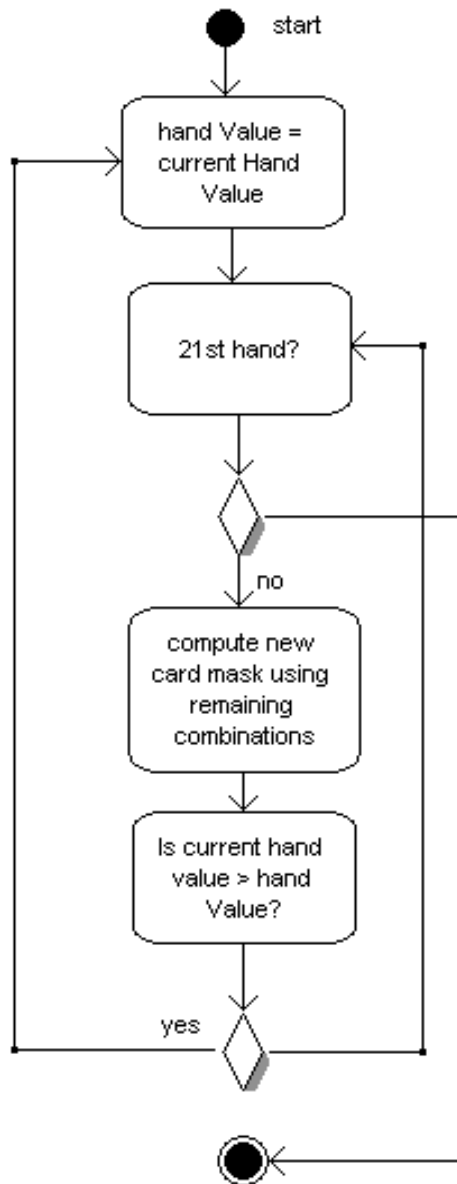


Figure 4-10 Player Hand Calculation Activity Diagram

4.3.1.4 *PokerServer Class*

The PokerServer class has the responsibility of sending and receiving packets between the PokerExec class and the iPhone Interface. The PokerServer utilizes Transmission Control Protocol (TCP). TCP was chosen for its reliability and its ease of integration using Qt. Furthermore, depending on which commands the PokerServer class receives from the iPhone interface, the PokerServer handles and pass the packets accordingly. PokerServer utilizes Qt's TCP classes in a separate thread to handle communication. The PokerServer class diagram is illustrated in [Figure 4-11](#).

Qt provides two classes to implement TCP, QTcpServer and QTcpSocket. QTcpServer is used to handle incoming TCP connections. QTcpSocket provides the interface for the protocol. The QTcpServer class handles incoming connections from up to four iPhones. The function QTcpServer::listen() is used to set up the server wait for new connections. The PokeServer utilizes the listen() function during initialization of the application. When a player joins the properly joins the network on the iPhone, QTChpServer::incomingConnection() is called. This function is a virtual protected, leaving the implementation to the group. When incomingConnection() is called, a new PokerPlayer be created and initialized. The game only starts when two iPhones have connected. Once two iPhones have connected, the PokerServer lets the PokerExec class know that the game is ready to begin. The PokerServer emits the signal QTcpServer::newConnection() when each client connects. This signal is connected to the SLOT AddPokerPlayer() of the PokerExec class to keep count of the number of connected iPhones. After the connections have been established by QTcpServer, QTcpSocket, interfaced with the PokerPlayer class, be used to read and send data.

In order for the game to run with the iPhone interface, certain game critical data must be passed between the server and the iPhone client. When dealing cards (i.e. sending the iPhone messages containing card values), the PokerExec must know which cards are sent out and to which iPhone client. Next, when a user bets or wins a pot, the PokerServerClass exchanges the number of chips involved and update that data in the PokerPlayer class. The following list summarizes the data that must be passed between the PokerServer class and the iPhone interface:

- Hole Cards (type: unsigned int, range: min = 0, max = 51)
- Chips(type: unsigned int)
- Player Name (type: QString)
- Interface Color(type: short)

TCP messages will be handled in a separate thread within the PokerServer class. PokerServerThread will be responsible for properly handling message

exchanges between the iPhones and the PokerServer class in a separate QThread. This allows the game to continue to run while the player decides what to do on their turn.

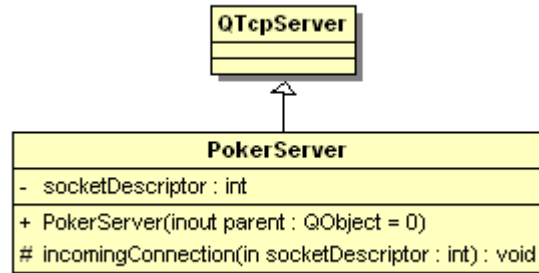


Figure 4-11 PokerServer Class Diagram

4.3.1.5 *PokerPlayer Class*

The PokerPlayer class has the main responsibility of holding updated information of each player and controlling transmissions between the iPhone interface. PokerPlayer is the class that directly communicates with the iPhone interface using the QTcpSocket class. Inheriting from QThread, PokerPlayer runs a separate thread and based on game conditions, safely write and read from the TCP socket. This class holds the name of the player, the number of chips the player currently has, and the 2 hole cards the player currently holds. The poker game application has up to four instantiations of this class for each player that joins the game. The class diagram is illustrated in [Figure 4-13](#).

When a player joins the table through the PokerServer class, described in section [4.3.1.4](#), PokerServer creates and start a new thread to control further communication between the iPhone and PokerPlayer interface. After the PokerServer initializes the thread, the PokerExec class notifies the PokerPlayer thread when a message is expected. This control of port communication ensure that the TCP port is only read from when it is necessary such as when it is the current player's turn to act during a hand. This optimizes the system by not utilizing the thread when a message is not sent.

Cards are stored as StdDeck_CardMask's, using Pokersource's types. Two cards can be represented on one mask, eliminating the need to have two separate masks per player class. Public functions to set the PokerPlayer's class hole cards were implemented. This function is called for each player each the time the deck is shuffled and dealt. These hole cards along with other data are sent to the iPhone using QTcPSocket's write() function. Although the cards are written to the port, they remain unchanged as the player does not have the ability to change their cards once they receive them. This would be called cheating in a real poker game.

Poker chips are stored as an unsigned int. When a PokerPlayer object is created, the player chip count is initialized according to the requirements established in section [2.2.1.22-2.4.2](#). Public functions, AddChips() and RemoveChips(), be used to update the player chip count. Each player has an assigned name that is displayed on the table interface. This name is stored as a String. Another class can get the player's name by calling the public function GetPlayerName(). Along with a name, each player has a position. This value is enumerated and determines the order when a player bets in each hand.

PokerPlayer inherits from the QThread class. Running a separate thread for each player controls where the TCP port is written to. Current game conditions in PokerExec controls when data is expected to be read and written to. This is illustrated in [Figure 4-12](#). The QThread is started from the PokerExec class where each PokerPlayer object is created.

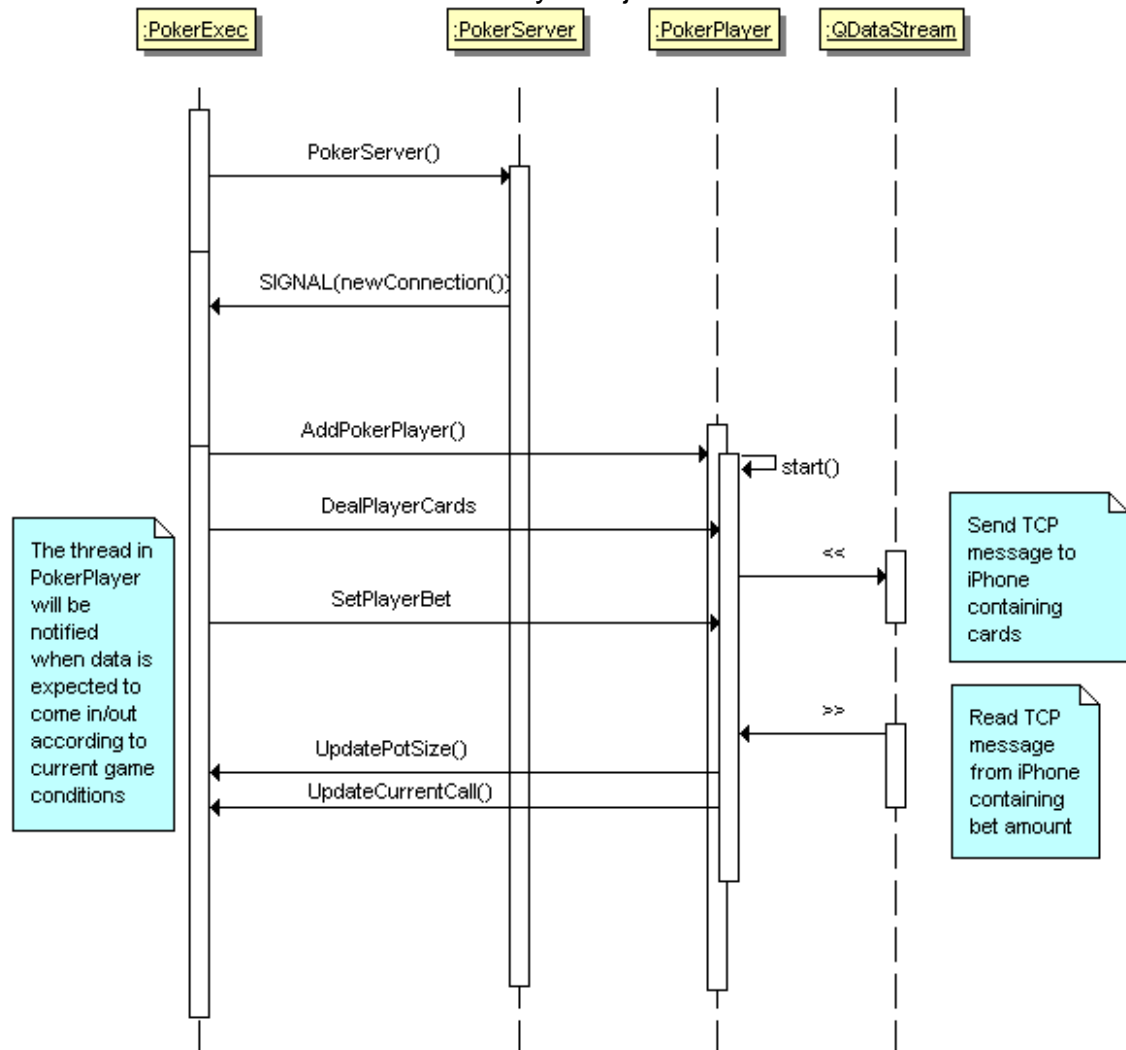


Figure 4-12 Poker Game Sequence Diagram

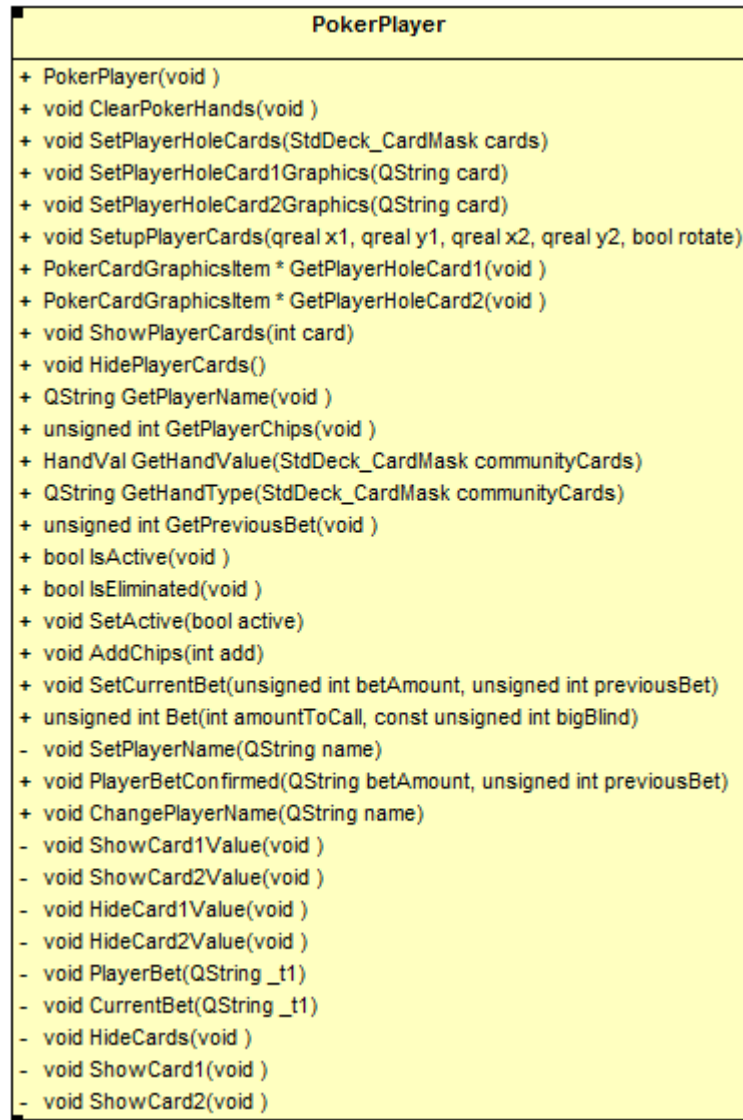


Figure 4-13 PokerPlayer Class Diagram

4.3.1.6 *PokerCardGraphicsItem Class*

The `PokerCardGraphicsItem` class is used to generate .svg graphics to the screen. This class has various slots to properly hide and show the `QGraphicsItem`. Furthermore, this class is used to animate dealt cards using the animation member. Finally this class derives `TouchObject` in order to show and hide player hole cards.

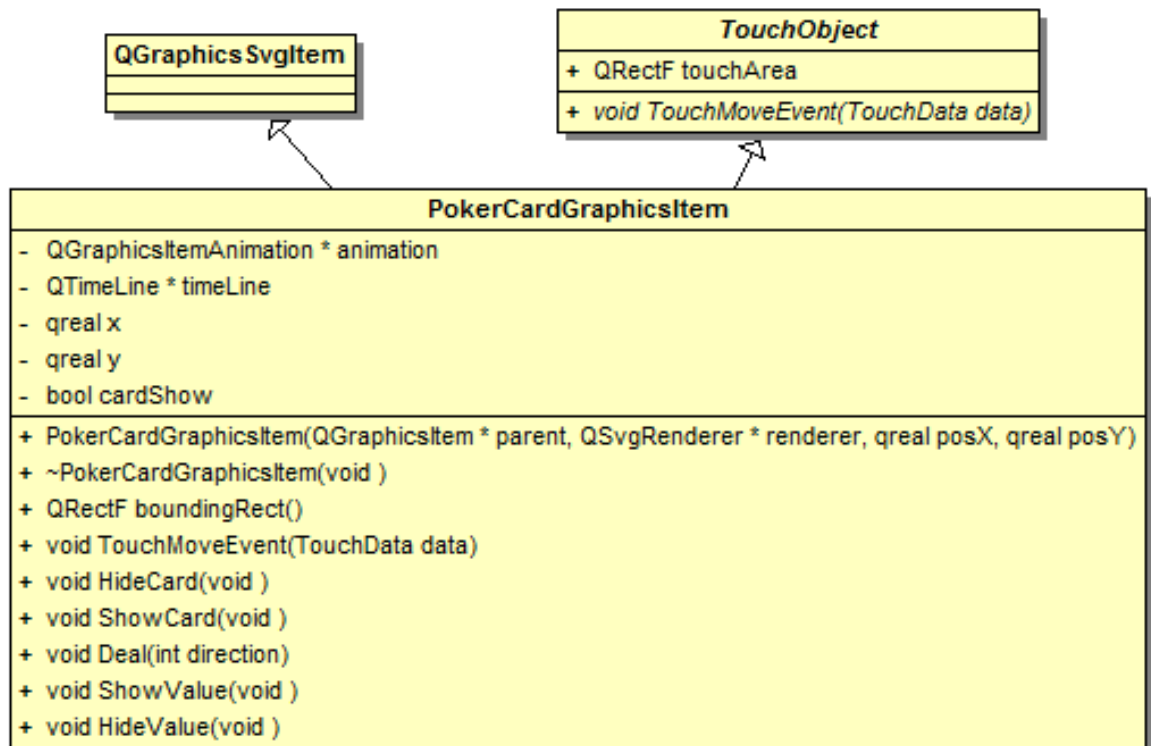


Figure 4-14 `PokerCardGraphicsItem` Class Diagram

4.3.1.7 DealerButtonGraphicsItem Class

The DealerButtonGraphicsItem class is the QGraphicsItem that represents the dealer button during the game. The brush is made using a QPixmap that uses a .jpg file. The dealer button has slots to hide and show the object as well as to reposition the graphics item in the scene. The class diagram is illustrated in Figure 4-15.

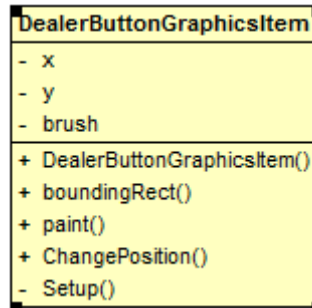


Figure 4-15 DealerButtonGraphicsItem Class Diagram

4.3.1.8 TextGraphicsItem Class

The TextGraphicsItem class is used by all text in the Poker Game Component. This class implements slots to properly update the text along with hiding and showing the object. It is created with a position and font. The class diagram is illustrated in Figure 4-16.

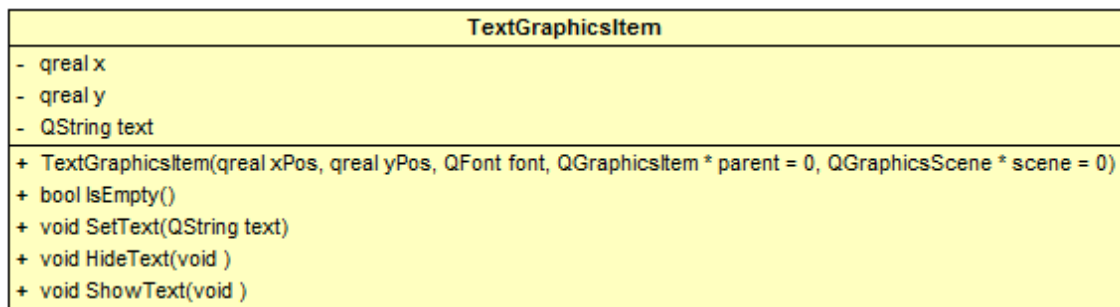


Figure 4-16 TextGraphicsItem Class Diagram

4.3.1.9 TouchListener

The TouchListener class handles touch inputs on the interface and handles actions accordingly. TouchListener implements the TouchLib API. TouchLib processes the readings from the USB camera and notifies clients of touch events. The poker game software translates user interactions on the multi-touch screen to game data using the TouchListener. This class uses the TouchListener interface provided by Touchlib. The three implemented functions from ITouchListener include fingerDown(TouchData data), fingerUpdate(TouchData data), and fingerUp(TouchData data). FingerDown is used to notify the class that a finger has just been made active. FingerUpdate notifies the class that a finger has moved. Finally, fingerUp notifies that a finger is no longer active. The TouchListener class diagram is shown in [Figure 4-17](#) ~~Figure 4-15~~.

TouchListener is implemented by using the location to determine appropriate action. Outlined in section [4.3.1.10](#) ~~4.3.1.8~~, location can be extracted from the TouchData that is passed in. Emitting a signal each time one of the ITouchListener functions are called, can notify certain classes of the touch events. Furthermore, the class only emits the signal TouchDetected() if the coordinates are appropriate. Utilizing Qt's SLOT and SIGNAL functionality eliminates the need to further implement a callback to classes that worry about touch events.

TouchListener utilizes callbacks to TouchObjects. Classes that wish to receive touch events must implement the TouchObject interface. TouchListener registers these TouchObject's into a QList of TouchObject*'s as shown below:

From PokerExec.cpp:

```
touchListener->RegisterTouchObject(chip1Bottom);
touchListener->RegisterTouchObject(chip5Bottom);
touchListener->RegisterTouchObject(chip10Bottom);
```

From TouchListener.cpp:

```
void TouchListener::RegisterTouchObject(TouchObject* touchObject)
{
    touchObjects.append(touchObject);
}
```

These TouchObjects are inserted into the QList using the insert() function. TouchListener later goes through the entire list to check if the touch detected is within the box defined by the QRectF TouchArea. If the coordinate is within the denoted rectangle, TouchListener calls the callback function TouchMoveDetected(). This process is shown below:

```
void TouchListener::fingerUpdate(TouchData data)
```

```

{
    // Iterate through registered touch objects
    for(int i = 0; i < touchObjects.size(); i++)
    {
        if(InBounds(data.X*100,data.Y*100,touchObjects.at(i)-
>touchArea))
        {
            // Call callback if touch is found in touchObject's
area
            touchObjects.at(i)->TouchMoveEvent(data);
        }
    }
}

```

The function `InBounds` takes in two parameters, an `x` and `y`, and checks if that coordinate is in bounds of a `QRect`, the third parameter. A `QRect` is defined by a corner, usually an `x` and `y` coordinate, and the length and width of the rectangle. `TouchListener` samples `Touchlib` events at a rate of 2 Hz to minimize duplicate touches being processed. This sampling takes place in the `run()` functions of the implemented `QThread`.

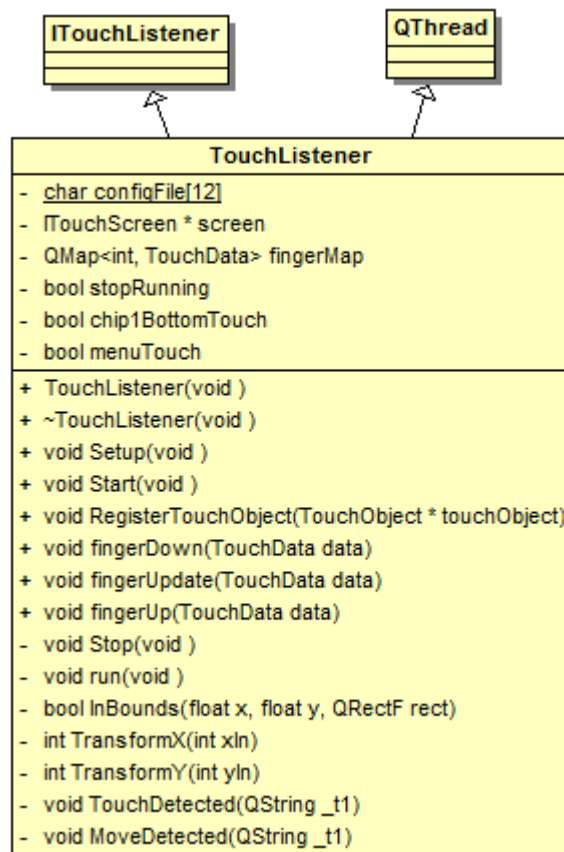
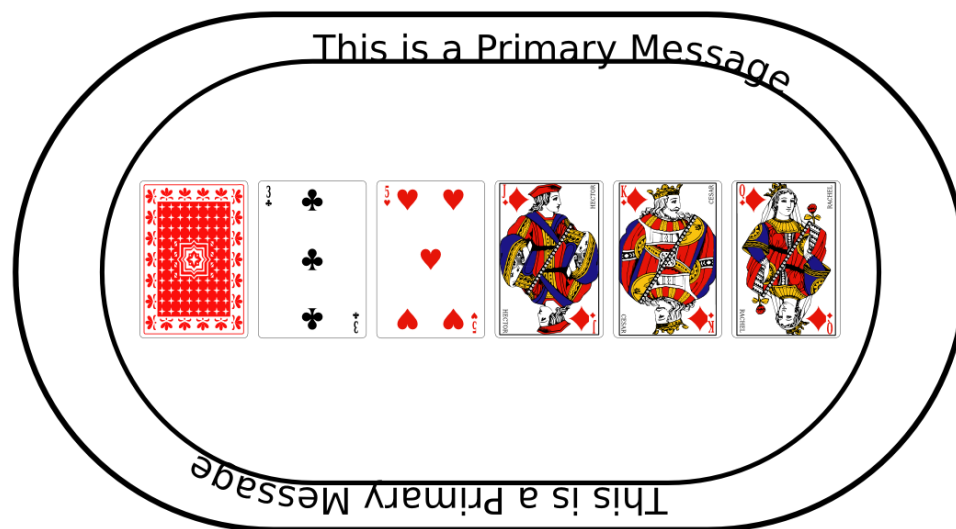


Figure 4-17 TouchListener Class Diagram

4.3.1.10 Table Interface

4.3.1.7.1 The Bet Zone

This zone handles the chips that are bet on a given player's turn. This zone is only visible and active during a player's turn. The interface has a Clear/Cancel button on the edge of the Bet Zone. This design choice allows a simplification of the coding of this interface and increases stability of the betting. The Bet Zone displays the current bet amount as well.



The Gesture Handler ensures that if a player's turn is not active, this zone behaves as the Auxiliary Zones. The Gesture Handler also handles the Bet Gesture for this zone. The Bet Gesture is outlined in the Gesture section.

4.3.1.7.2 The Chip Zone

The Chip Zone is the area in which the Poker Chips are stored. The Poker Chips are grouped by denomination. Chips may be moved into the Bet Zone, when this zone is active. The Chip Zone have a floating marquee that displays the Player's total Chip Value as well as the Player's Name and each individual chip count.

4.3.1.7.3 The Color Marquee

This marquee allows the player have graphical feedback about the current state of the game. Once the player has chosen a color to be referenced by and the game has begun, the marquee spring into action. The marquee designates the Player Area's outer boundary. This ensures that the gestures of the player are within the bounds of the program's Gesture Handler. The marquee pulses when the player's turn is active. Visual feedback allows the player to simply “know” when their turn is active. The Color Marquee also indicates the boundary edge for the Scrolling Marquee zone. [Figure 4-18](#) ~~Figure 4-16~~ illustrates the Color Marquee.

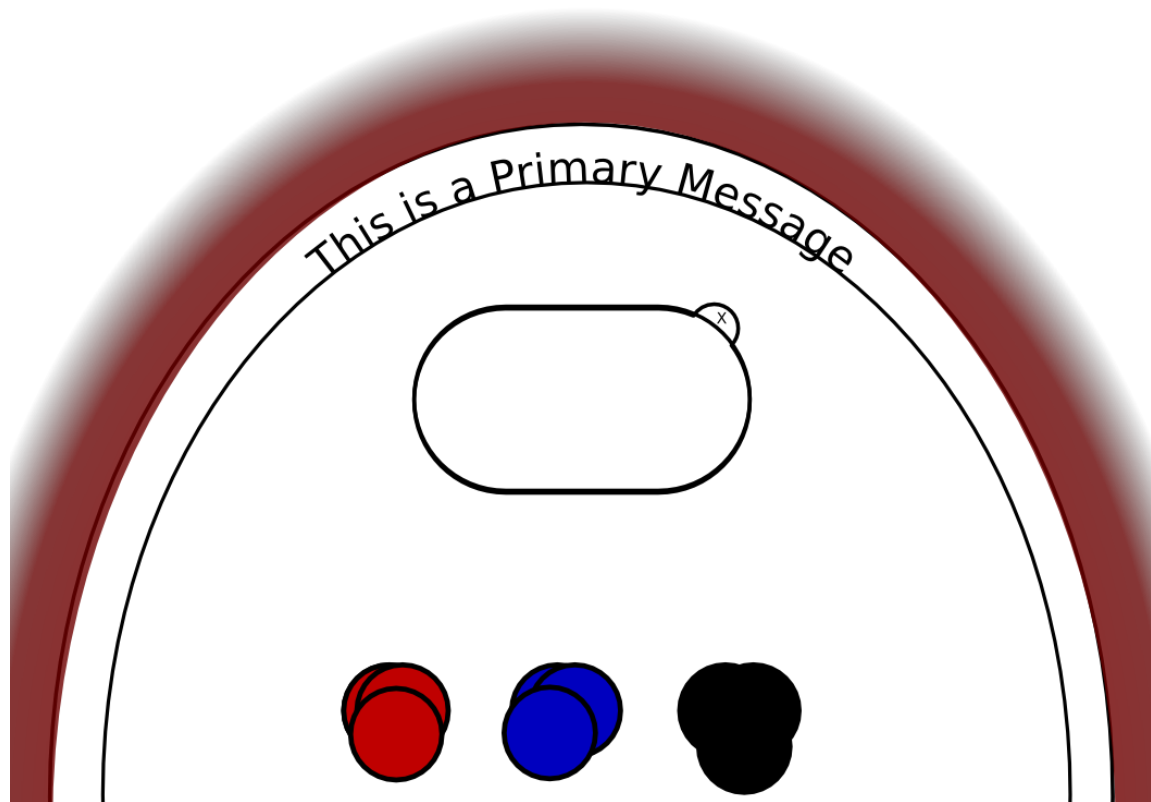


Figure 4-18 Player Interface

4.3.1.7.4 The Auxiliary Zones

There are 3 Auxiliary zones on the Player Interface. The Gesture Handler contains a switch in order to facilitate the correct Gestures for the correct time. These zones are on either side of the Chip Zone. An additional Auxiliary Zone is in the same area as the Bet Zone when the Bet Zone is not active. These Auxiliary Zones support the Check Gesture as well as the Fold Gesture.

4.3.1.7.5 *The Scrolling Marquee*

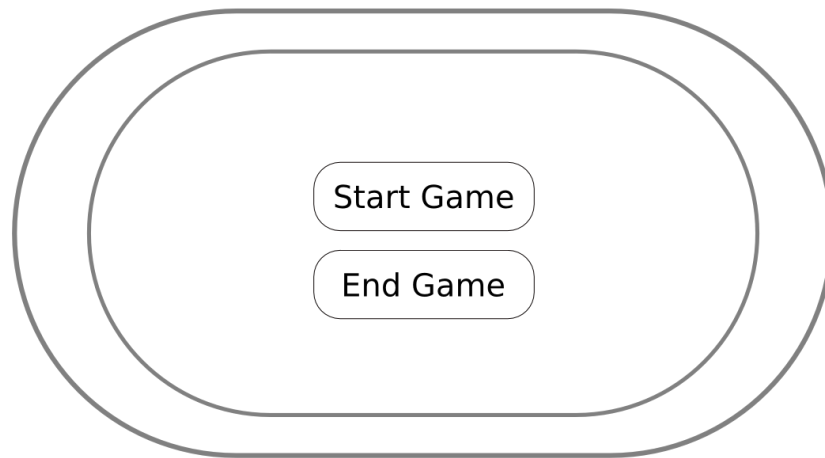
This marquee is a message zone that is physically near to the player so messages can be given to the player in an unobtrusive way. This zone allows the Game Messages to be displayed as well as Interface Messages (such as Current Song Playing and Order Notifications). This zone allows messages to be sent to a specific player only as well as mimic the Main Message Marquee part of the Dealer Interface.

4.3.1.7.4.1 *The Game Menu*

The Game Menu is the interface for the start of the game. This interface is the visual way that Player's may start the game once the iPods have been connected to the system. This is a player action interface and appears in the center of the table. This interface supports various commands related to the flow of the game.

Functions

- Start Game
- Exit Game



This Menu hides the dealer cards if they are visible as well as any chips in the interfaces and lowers the opacity of the Player Zones to focus the user's attention to the center of the table.

4.3.1.7.4.2 *Marquee Handler / Message Handler*

The Message Handler is the starting point for any message to the user in the MTPT. These messages can be displayed in a variety of ways and include other visual feedback methods. The Message Handler ensures that messages are dispersed in a manner that is conducive to the main purpose of the table. The

messages must be handled and displayed to the user in order to make the game progress normally and only display relevant information.

There are two types of messages:

- Primary Messages
- Secondary Messages

The Message Handler has a queue for Primary Messages and another for Secondary Messages. Each of the queues holds the Message ID, the Message Type (Primary or Secondary), the Message and a Player ID if the message is specific to a player.

Primary Messages

These messages take priority over any queued Secondary Messages. These messages are used to ensure that the game progresses. Primary Messages may be directed towards a specific player as well as the Scrolling Marquee.

Possible Primary Messages include:

- Current Turn Notifications
- Current Pot Notification

Secondary Messages

These messages are of an auxiliary type. They are surpassed by any Primary Messages that are in the Primary Message Queue.

Possible Secondary Messages include:

- Advertisements for the current location (“Support Austin, Drink Coffee”)

4.3.1.11 Dealer Interface

The Dealer Interface is defined as the interface present in the center of the MTPT. The Dealer interface is a single interface not to be used more than once. This interface is designed to handle all of the required Dealer information such as the Dealer's Cards and information regarding the game's current state. Auxiliary Messages may be displayed using the Circular Marquee. The Dealer Interface does not support any Gestures within its boundaries.

4.3.1.8.1 The Circling Marquee

The Circling Marquee exists around the Dealer Interface. This is the boundary of the Dealer Interface. This marquee displays information for the current game such as scores, current player and various other messages that are required to make sure they come continues as expected. Auxiliary Messages can be displayed using this interface if they are for every player and not a specific player.

4.3.1.8.2 The Dealer Cards

The Dealers Cards are displayed in the direct center of the table. The three initial cards are dealt at the beginning of the game with an additional card added up to 5 total per the rules of the game. This area supports no Gestures.

4.3.1.12 iPhone Interface

The iPhone Interface was the most simple and straightforward of the interfaces. The iPhone interface is designed to be as foolproof and stable as possible. The Player needs access to their current hand throughout the entire game as well as their current chip count, player color and other useful information.

The iPod Interface features two modes:

- Options Mode
- Play Mode

Options Mode

The Color Chooser

The Player on connection to the Main Computer may select a color to be identified by. This color is used in order to give visual feedback for the player. The Player Interface edge is identified with the Player's color. The Color Chooser is illustrated in [Figure 4-20](#)~~Figure 4-18~~.

The Player Name

The Player Name is used in the Scrolling Marquee around the Dealer Area as well as the Scrolling Marquee around the individual Player Interfaces. The Player Name is captured using the standard iPhone keyboard entry in Landscape Mode. The Options Button is not functional if a player name is not entered.

Play Mode

The Player Cards

The Player Cards are the primary interface element of the iPhone Interface. These two cards are displayed prominently in the interface. The cards are not able to be moved across the screen in any manner. [Figure 4-19](#)~~Figure 4-17~~ illustrates how the cards are displayed on each of the player's iPhones.

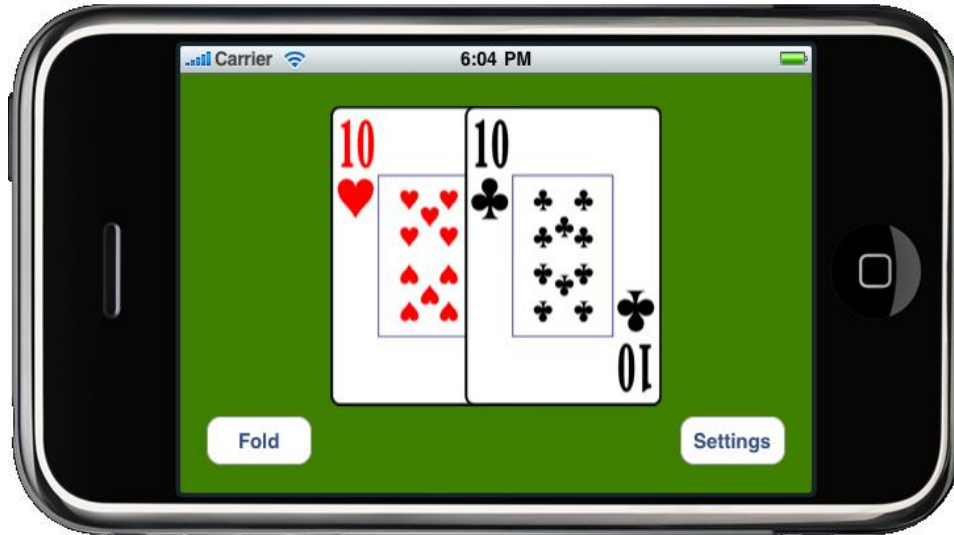


Figure 4-19 Player Hole Cards

The Options Button

The Options Button switches the iPod from Play Mode to Options Mode or vice versa. This button exists in both interfaces. It appears as an “earmarked” page from a book. The entire interface visibly flips at each press of the button. This function can be accessed at any time during the game and the Main Interface reflects the changes immediately.



Figure 4-20 Color Chooser

4.3.1.13 The Gesture Handler

The Gesture Handler ensures that all gestures are handled correctly. This is a service provided by the main computer which interfaces to Touchlib. The Gesture

handler is aware to ensure that only certain gestures can happen in certain places of the table.

The Gesture Handler supports the following gestures:

- Bet Gesture
- Check Gesture
- Pinch Gesture (PDF/Menu Viewer)
- Object Selection (Close Button on Bet Zone, Close Button on PDF/Menu Viewer)
- All In Gesture
- Fold Gesture

Bet Handler

The Bet Handler keeps track of the betting portion of the game. This service may be polled for information about the current totals for each player as well as the total pot for each round. The Bet Handler accepts the player's ID and returns the player's current holdings in money as well the values of their current chips. The Bet Handler is separate from all code regarding the actual game and is designed to simply manage the funds. All of the game logic code is provided by an additional service.

The Bet Handler also provides support for the current pot of the round. When the Bet Handler is notified of a winner of a particular round, the funds are automatically transferred to a specific player and their chip count is updated. A Primary Message could be used to display the total winnings for that round. This message may not be dispatched by the Bet Handler.

4.3.2 Full Interface Guidelines

The Main Interface features four Player Zones and a Dealer Zone. The Restaurant Menu may be called into focus by the user as well as the Game Menu. [Figure 4-21](#)~~Figure 4-19~~ shows all of the interface. This is an example including all Player Zones active and a current game In Session. The Restaurant Menu is also active. **Error! Reference source not found.** illustrates the Full Interface with the Red Player active in a bet. A round is also in session.

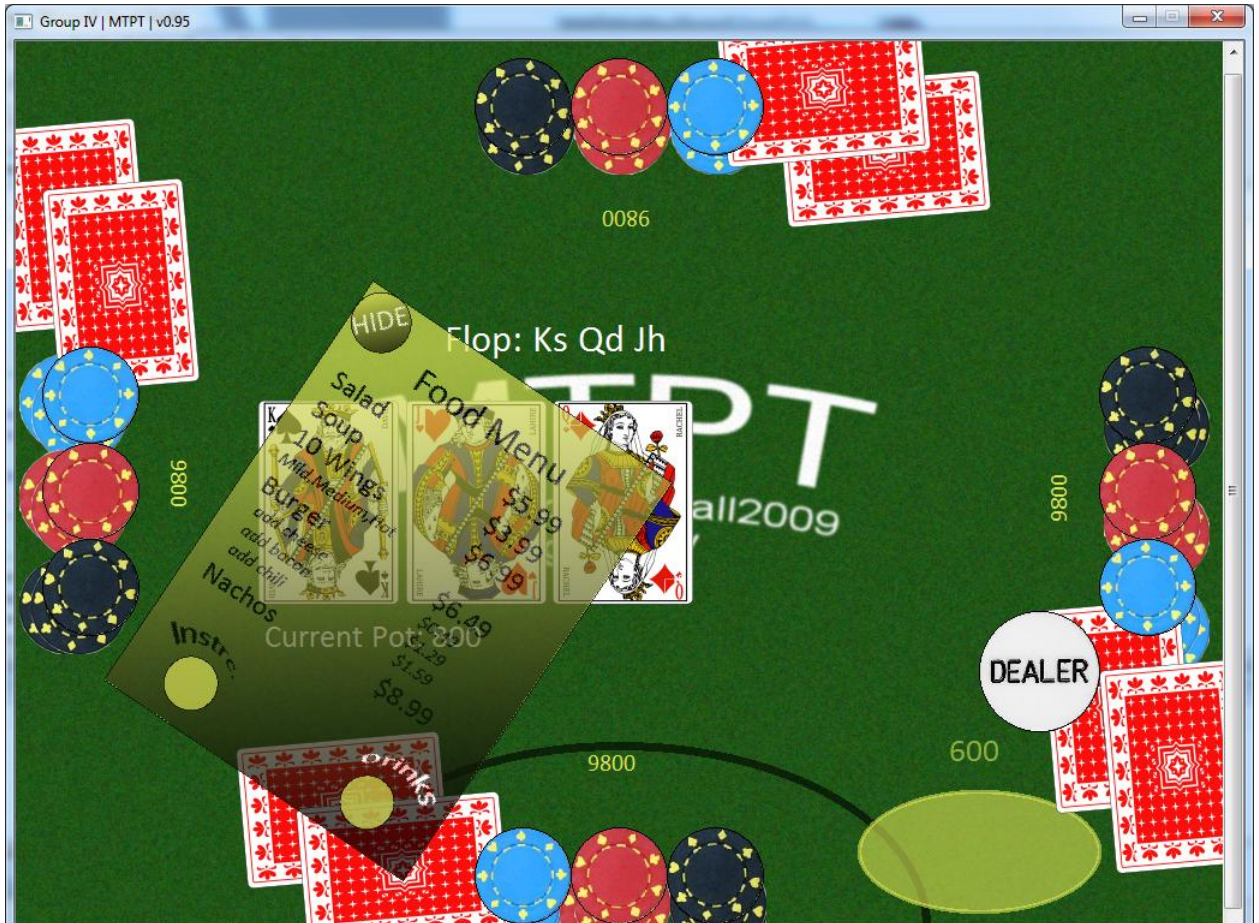


Figure 4-21 Full Interface

Restaurant Menu

The Restaurant Menu Software Component is responsible for the display and manipulation of a table top menu display while a poker game is running. This menu may be called during any time of the poker game is running. The Restaurant Menu allows users to display the menu and manipulate it while the game is playing. This showcases the fundamentals and possibilities of multi touch technology. Furthermore, the Restaurant Menu showcases the possibilities of having the table at a public table such as a restaurant.

4.3.2.1 RestaurantMenu

Menu Widget is the actual object that the user sees. In this case, it is a rendered menu displaying fictional items and prices. RestaurantMenu inherits from QGraphicsItem. QGraphicsItem is the base class for of Qt's graphic items. This base class provides foundation for a developers custom's items. QGraphicsItem requires two implemented functions: boundingRect() and paint(). The pure virtual function boundingRect() defines the outside boundary of a graphic item; in the RestaurantMenu's case, the border. All graphic handling and painting is done in this bounded region. The other pure virtual function, paint(), draws the actual widget. This is the function where painting and drawing of actual shapes take place. The RestaurantMenu class diagram is show in figure [Figure 4-22](#) [Figure 4-20](#).



Figure 4-22 RestaurantMenu Class Diagram

4.3.2.2 GestureHandler

The Gesture Handler ensures that all gestures are handled correctly. This is a service provided by the main computer which interfaces to Touchlib. The Gesture handler is location aware to ensure that only certain gestures can happen in certain places of the table.

- Bet Gesture
- Check Gesture
- Pinch Gesture (PDF/Menu Viewer)
- Object Selection (Close Button on Bet Zone, Close Button on PDF/Menu Viewer)
- All In Gesture

Bet Gesture

The Bet Gesture is defined as a swipe (from Left to Right, oriented by the Player Zone) across the bet zone. This gesture initiates the Bet. The Chips inside of the Player's Bet Zone are passed to the Bet Handler. The Dealer Zone registers the chips and generates chips around the zone to simulate the effect of a real bet in Poker. The Bet Gesture is illustrated in [Figure 4-23](#)~~Figure 4-21~~.

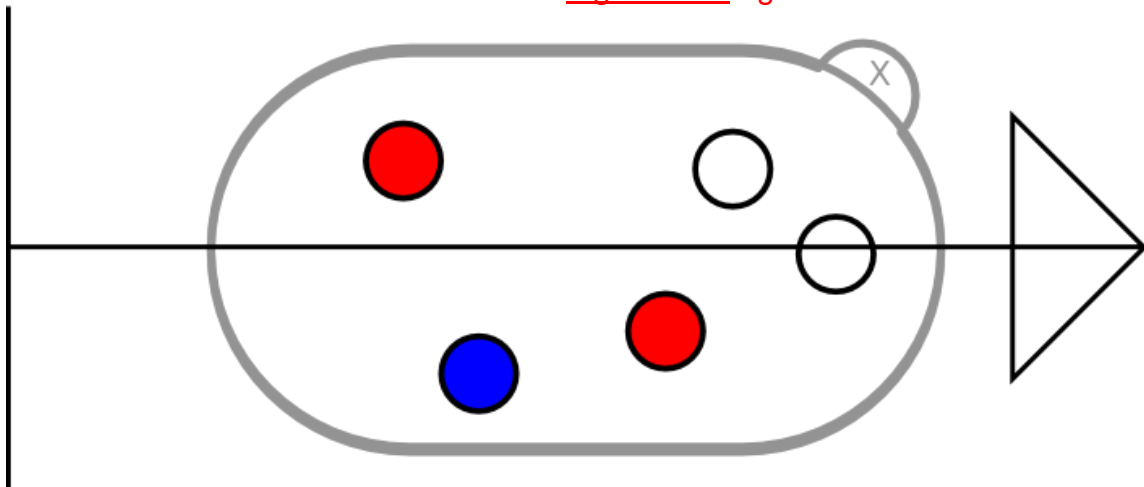


Figure 4-23 Bet Gesture

Check Gesture

The Check Gesture is defined in [Figure 4-24](#)~~Figure 4-22~~. The circles represent the player's Fingers.

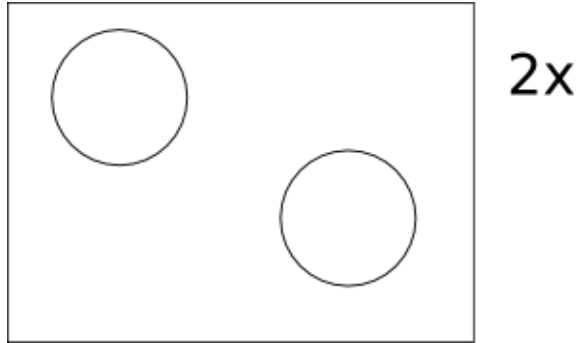


Figure 4-24 Check Gesture

This configuration must be made twice for this to be registered as a Check Gesture. The Check Gesture passes the “check” to the PokerExec class to ensure the game progresses. The Turn is increased and the Player’s Interface registers the change by indicating it is no longer the previous Player’s turn. This is indicated by the removal of the Glow around the Player Zone.

Pinch Gesture

The Pinch Gesture is used with the Restaurant Viewer. This is used to Zoom In and Zoom Out. The gesture works in the same fashion as the iPod. The Pinch Gesture is illustrated in [Figure 4-25](#)[Figure 4-23](#).

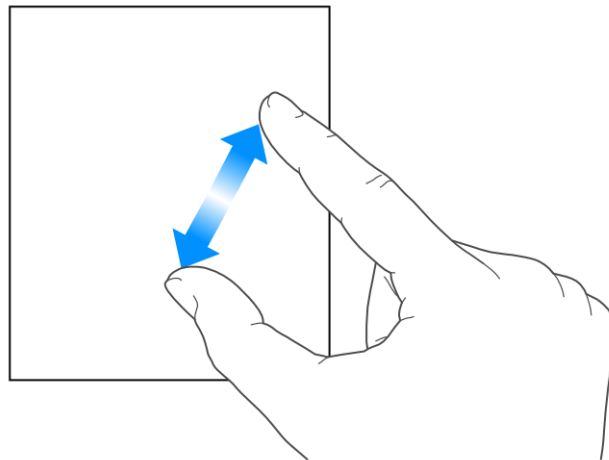


Figure 4-25 Pinch Gesture

Object Selection

Certain Objects on the screen are able to be picked up and moved around. The Chips may be individually moved across the screen as well as the Restaurant Menu. There are some objects that are part of the interface that may not be moved such as the visible boundaries of the zones; however, not every object can be efficiently coded in such a manner. Every attempt is made to ensure these objects can move freely across the interface.

Possible Objects:

- Chips
- Restaurant Menu
- Dealer Cards
- Game Menu

All In Gesture

The All-In Gesture is defined as a cupping of hands around the Player's chips. The cupping must be performed in one motion towards the bet zone and must encompass all of the chips. This gesture is illustrated in [Figure 4-26](#)[Figure 4-24](#).

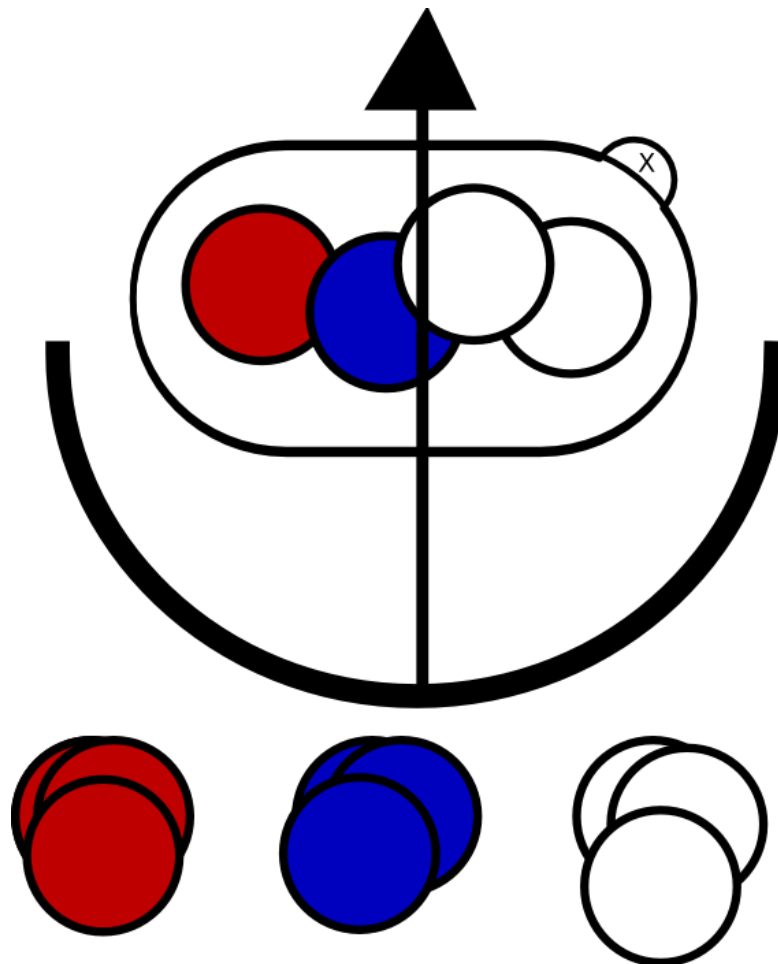


Figure 4-26 All In Gesture

Fold Gesture

The Fold Gesture is a multi-part gesture. This Gesture uses data sent by the Player's iPod as well as data parsed from TouchLib to ensure that the Fold Gesture is not misused or misinterpreted. The iPhone must be placed on the table outside of the player's area. The iPhone must be registered to the player who is folding and the iPhone must be nearest to that player's Zone. [Figure 4-27](#)[Figure 4-25](#) illustrates a correct fold gesture.

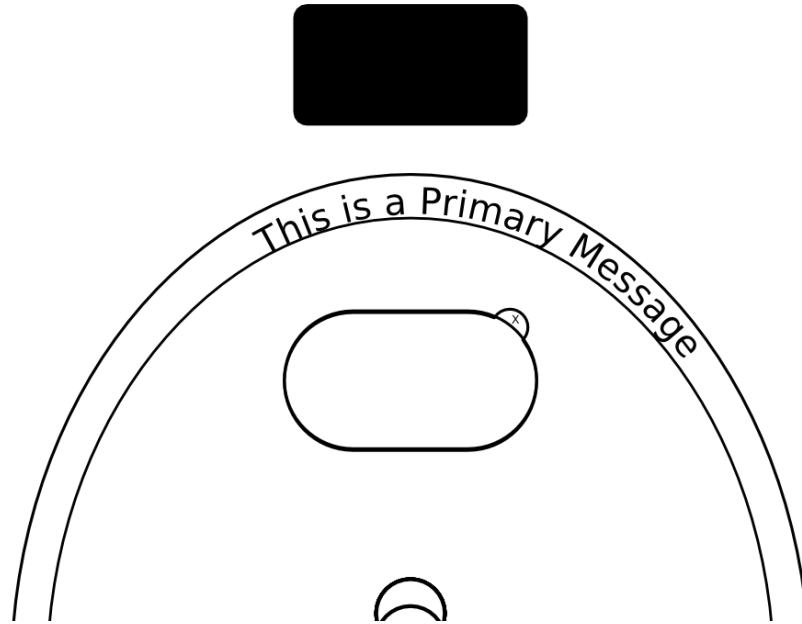


Figure 4-27 Fold Gesture

The iPod then queries for its current orientation. The iPhone includes an 3D Accelerometer which can be used to determine orientation in three space. This ensures that the iPod is upside down, face flat down on the MTPT. If the iPod is not face down, the Gesture is not executed. [Figure 4-28](#)[Figure 4-26](#) illustrates an incorrect fold gesture

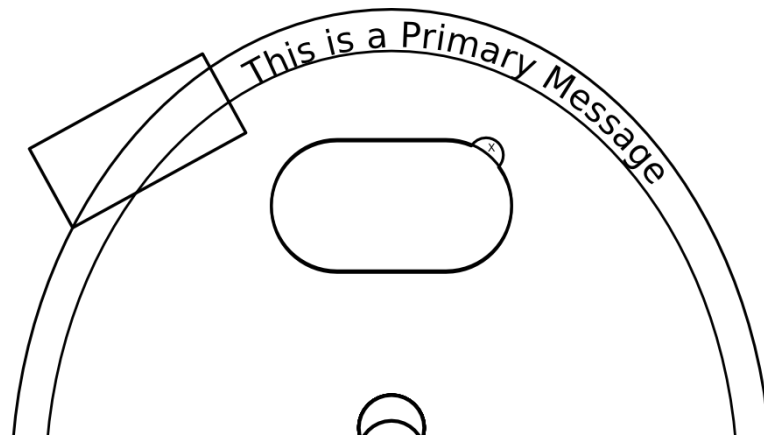


Figure 4-28 Incorrect Fold Gesture

Table 4-8 describes where each of the gestures can be executed. Each Gesture may only be registered in each of these zones and nowhere else.

Table 4-8 MTPT Gestures

Gesture	Zones
Bet	Bet Zone in Player Zones (1-4)
Check	Player Zones (1-4)
Pinch	Restaurant Menu (Popup)
Object Selection	All
All In	Player Zones (1-4)
Fold	Player Zones (1-4)

4.3.2.3 Menu Interface

The Restaurant Menu is able to be called to the screen using the Restaurant Gesture and can be placed anywhere on the table. By default, it appears near the given gesture. The menu is fluid and self-contained. The interface is able to be zoomed, paged, and moved around the entire table. The menu is essentially a PDF viewer with a custom scalable interface. The interface is touch friendly and allows easy gestures to control the flow of movement.

This menu can be called to the screen using the Restaurant Gesture and can be placed anywhere on the table. By default, it appears near the given gesture. The menu is fluid and self-contained. The interface be able to be zoomed, paged, and moved around the entire table. The menu is essentially a PDF viewer with a custom scalable interface. The interface be touch friendly and allow easy gestures to control the flow of movement.

Design Requirements

- The Restaurant Menu be called using the Restaurant Gesture.
- This gesture call the menu to focus.
- The Menu be closed using the “x” button.
- The menu may be scaled using a “pinch” gesture.
- Paging completed using a Flick gesture

Refreshing Chilled Beverages			Frozen Beverages		
Organic Tea	Med.	Lg.	Med.	Lg.	
Berried Treasures Iced Tea	1.85	2.00	<u>Yak</u>	3.60	4.10
Passion Fruit Green Iced Tea			Frozen coffee with a hint of chocolate & vanilla		
*Try it Sweet Add a shot of Vanilla	.45		<u>Chai Chai</u>	3.90	4.35
			Oregon Chai blended with milk or soy		
Coffee			<u>Brando</u>	4.35	4.80
Iced Coffee (sweet or un-sweet)		2.00	Chai Chai w/ raspberry		
Iced Latte	Med.3.40	Lg. 3.70	<u>Chunkey Monkey</u>	3.90	4.35
			Frozen coffee with chocolate & banana		
Soda			<u>Artic Mocha</u>	3.90	4.35
Coke, Diet Coke, Sprite		1.15	Frozen coffee with chocolate		
Captain Eli's Rootbeer		2.25	<u>Caramel Frost</u>	3.90	4.35
			Frozen coffee with caramel		
Izze Sparkling Juices:		2.25	Domestic Beer		
Blackberry, Blueberry, Clementine, Grapefruit,			Budweiser, Bud Light		3.00
Lemon, Pear, Pomegranate,			PBR		2.00
			Imports		
Bottled Water		1.99	Heineken, Kirin		3.50
			Guinness		4.00
			Organic Samuel Smith		4.50
			Paulaner Hefe-weizen		
			Microbrews		
			Please see our display case for selection		3.50
			Mississippi Mudd		6.00



The PDF Menu must be able to be scaled and zoomed due to resolution constraints of the system. A projector of ample resolution cannot be purchased in order to view a high enough resolution copy of the PDF Menu or any type of content, for that matter. To remedy this problem, the menu is re-rendered to fit out user interface.

5.0 Design Summary

The Design Summary section reviews the overall design of the MTPT Project. These hardware and software components were described in detail in the previous sections. The MTPT hardware encompasses the multi-touch display, computer, router, iPhone devices, and power supply. All of the hardware is enclosed in a designed constructed framework. The MTPT software includes the Poker Game software, Restaurant Menu software, and the iPhone interface. These software components with the exception of the iPhone interface runs on the computer of the MTPT hardware. The MTPT software utilizes a number of open source libraries, but the majority was developed by the group. The complete block diagram of the MTPT project is illustrated in [Figure 5-1](#) ~~Figure 5-4~~.

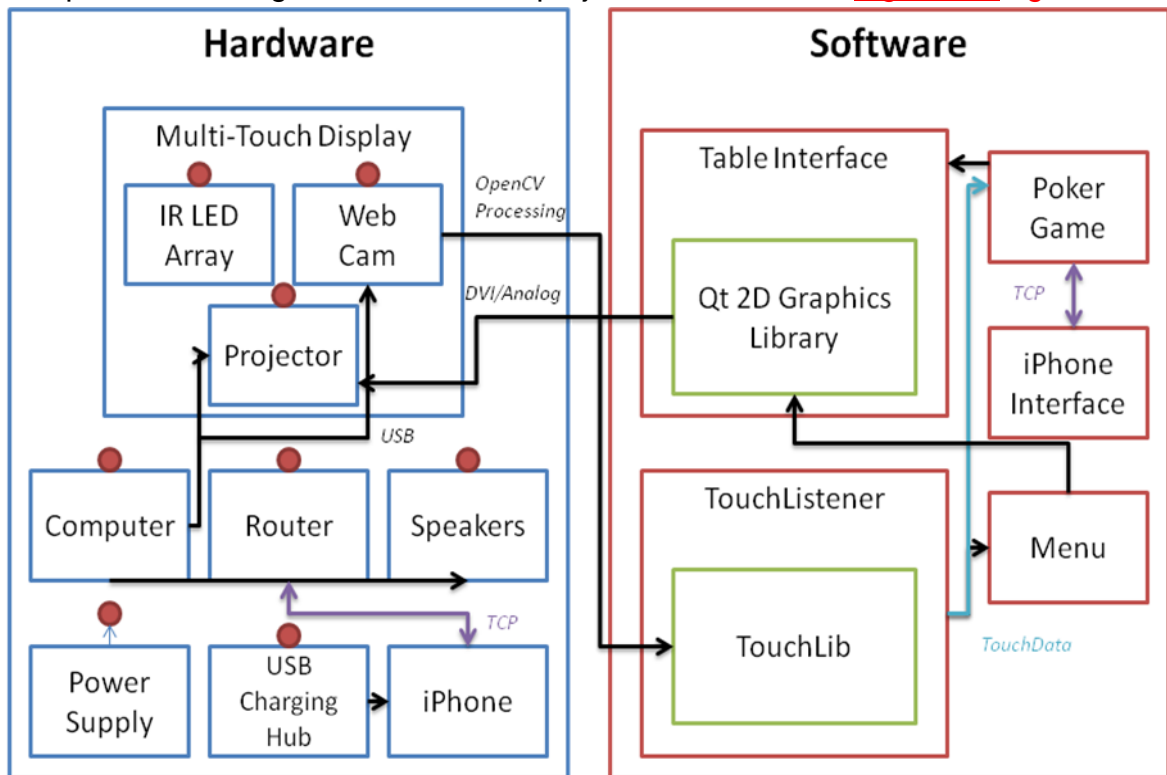


Figure 5-1 MTPT Block Diagram

The hardware for the MTPT consists of four main components which include the multi-touch display, computer, router and framework. The multi-touch display consists of three sub-components which include the screen, web cam, and projector. The screen utilizes the Frustrated Total Internal Reflection (FTIR) technique to give the user multi-touch capability. This technique uses 70 880 nm IR LEDs to flood an acrylic screen with light. When the user touches the screen the light is frustrated and directed towards the IR camera below. In order to increase the sensitivity of this interaction a compliant layer was added to the

screen made of silicone and trace paper. The trace paper is used to display the visible screen from the projector.

The infrared camera is used to display screen touches to the computer. The computer software then can analyze the coordinates of the touch and act accordingly.

The projector for the MTPT is the single most expensive component of the project. It provides the user with visual feedback on the screen. The projector acquired meets and exceeds the requirements laid out. One of the problems encountered with the projector is the throw distance of the display is longer than the table itself. To solve this, a mirror was installed in the table enclosure to lengthen the throw distance.

The computer is the brain of the MTPT. It integrates all the components and gives the programmer complete control over all functions. One of the key features of the MTPT is the ability to upgrade the computer as technology improves. Therefore the computer was designed to be modular by allowing the user to use any computer that meets requirements.

The wireless router allows the poker table software to communicate with the iPhone devices. The poker game software sends the cards wirelessly through router to the iPhone devices. This in turn gives the user the ability to hide their cards.

The power supplied to the MTPT comes from one 110 VAC power receptacle. It then powers the computer, projector and wireless adapter. The computer use its power supply to convert AC to DC and power the USB charging hub, IR camera, and temperature controlled fans. The LED's are powered by a linear power supply that was design and built by the group. This power supply also allow for LED intensity adjustment to control the sensitivity of the touch screen depending on light conditions.

The table framework of the MTPT was designed to be used in a restaurant or coffee shop. Therefore it has room for cup holders and an area in front of the user for a small plate. The height of the table was design so that standard counter stools could be used for seating, allowing the users an optimal level to see the screen. The table was stained and painted to give it an appealing modern look.

The Multi Touch Poker Table includes three software components: the Poker Game, interfaced with the iPhone and table, and Restaurant Menu. The Poker Game software is main application of this project, simulating a full robust game of Texas Hold'em. The game utilizes the iPhone as the player interface. These features are encompassed in the Menu. . All effort was used to have these software components completed before the end of December. These components are illustrated in [Figure 5-2](#)[Figure 5-2](#).

The MTPT Software Components utilizes three open source libraries: Qt, Touchlib, and Pokersource. Qt is an open source, cross platform framework in C++. Qt was used mainly with providing the MTPT Software with threading, TCP networking, and graphics. This library proved to be very useful and versatile with it's in depth support and options.

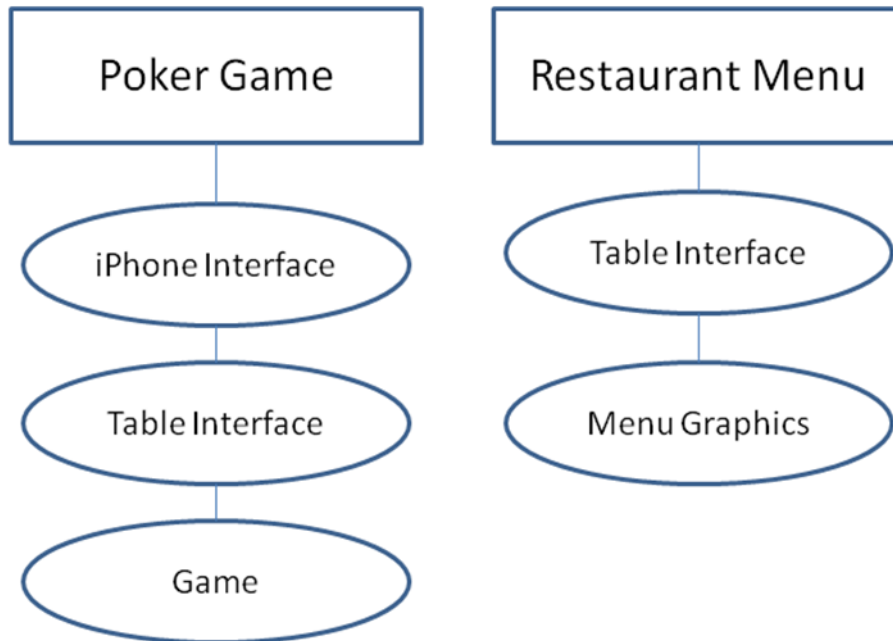


Figure 5-2 MTPT Software Components

The Poker Game Software component is the core component of this project. This software included the game software, the source needed to communicate to the four iPhones, as well as the graphics library that were used to display the game on the MTPT. The software utilizes multiple threads to run. First, there was a communication thread. This thread was used to connect the four iPhones. This thread maintains the client/server functions of the software. To go deeper into, this thread handles incoming and outgoing packets to the iPhone using the TCP protocol. Next, a touch listener was used to detect touches on the MTPT. This listener communicated with the main Poker game executive and update with the appropriate user inputs. The classes that were implemented along with a short description are listed in [Table 5-1](#) ~~Table 5-4~~.

Table 5-1 Poker Game Classes

Class Name	Description
PokerExec	Main executive for poker game, responsible for Texas Hold'em game code and game data
PokerServer	Responsible for establishing and maintaining connections

	with the iPhones
PokerPlayer	Responsible for holding updated information for each player, handles transmissions between iPhone interface
TouchListener	Handles touch inputs, passes touch events from Touchlib to applicable classes to handle

The Restaurant Menu Software Component is responsible for the display and manipulation of a table top menu display while a poker game is running. The menu features the MTPT gesture functionality. This menu may be called during any time of the poker game is running. The Restaurant Menu allows users to display the menu and manipulate it while the game is playing. This showcases the fundamentals and possibilities of multi touch technology. Furthermore, the Restaurant Menu showcases the possibilities of having the table at a public table such as a restaurant. The classes that were implemented are listed and described in [Table 5-2](#).

Table 5-2 Restaurant Menu Classes

Class Name	Description
MenuWidget	Restaurant Menu QGraphicsItem that be displayed and manipulated
GestureHandler	Handles touch events, used to manipulate MenuWidget

The MTPT Software was developed in C++ using Microsoft Visual Studio. One executable was used to run the entire application for the MTPT. This application ran off the housed computer in the unit. After prototyping the entire unit as described in section [6.06-0](#), it was thoroughly tested against the specified requirements listed in Section [2.02-0](#).

6.0 Prototyping

6.1 Hardware

The MTPT prototype was split into two groups and constructed in parallel. Group 1 consists of all the electrical components and group 2 encompasses the physical table.

Group 1 prototyping started with building the temperature sensor and LED power supply. Once built, individual component testing can be performed which is laid out in the hardware test plan. This ensured component functionality before the build process begins. Once the components have been deemed functional they can be installed in to the MTPT according to the table layout diagram shown in the power section.

Group 2 prototyping started with the screen. The screens compliant layer must be constructed and combined with the acrylic sheet. The table top framework can then be built, allowing the screen to be temporarily installed to the table. Once installed, the LED's can be placed around the screen and testing can begin on the screen, LED's and camera integration. The bottom section of the table that encloses all the components can be built in parallel. Once individual component and integrated testing is complete, the components can be installed into the bottom section and wired. The table top can then be combined with the bottom section and final testing commenced. Finally the table was stained and painted

6.2 Software

The software for the MTPT was divided by all three group members. Each software component was developed and tested independently of each other. The Poker Game source outside of the interface was developed and tested using a console application. Using inputs from the user and printing out results to the screen ensures that the game mechanics under the hood works accordingly before integrating with Qt's graphics framework. On the other side, the graphical interface was developed alongside of the game code. This was integrated with the hardware while coding and testing to ensure proper display. The interface source was not required the game code underneath it to run a sample display. By developing the software so they are independent of each other, relying on object-oriented principles, small prototyping took place until a final prototype is complete.

The first piece of hardware that was integrated was the webcam. The TouchListener was the only class that needed sufficient testing with the web cam to ensure proper calibration of TouchLib. Once the full touch screen is

constructed, it was calibrated using the provide TouchLib calibration application shown in [Figure 6-1](#)~~Figure 6-4~~, reprinted with permission from Seth Sandler[6]. Using the whole hardware unit, with projector, LED array, and webcam, the unit needed to be calibrated with the library successfully before running and developing any software applications on it. The Touchlib calibration application outputs a config.xml configuration file. The sliders shown in figure [Figure 6-1](#)~~Figure 6-4~~, we calibrate it until the rectify window shown in the bottom left corner only shows the finger tip blobs with no background noise. After all filters were set, the configuration file was outputted. This configuration file was used in the real calibration of the MTPT with Touchlib. The next configuration phase consists of touching points on the actual screen of the MTPT to ensure proper point to point translation from the screen to Touchlib. After the multi screen has been successfully calibrated, the two sample demos were run to test the calibration. Touchlib includes two simple demos for users to test, a smoke demo and a simple game of pong. The smoke demo is shown in [Figure 6-2](#)~~Figure 6-2~~. Running these demos assisted the group in verifying touch calibration and also helped verify the projector is in the correct position. The smoke demo included with the library is shown in [Figure 6-2](#)~~Figure 6-2~~(reprinted with permission from Seth Sandler[6]). This demonstration shows the graphic possibilities of the library. Touches on the screen create colors with a smoke effect.

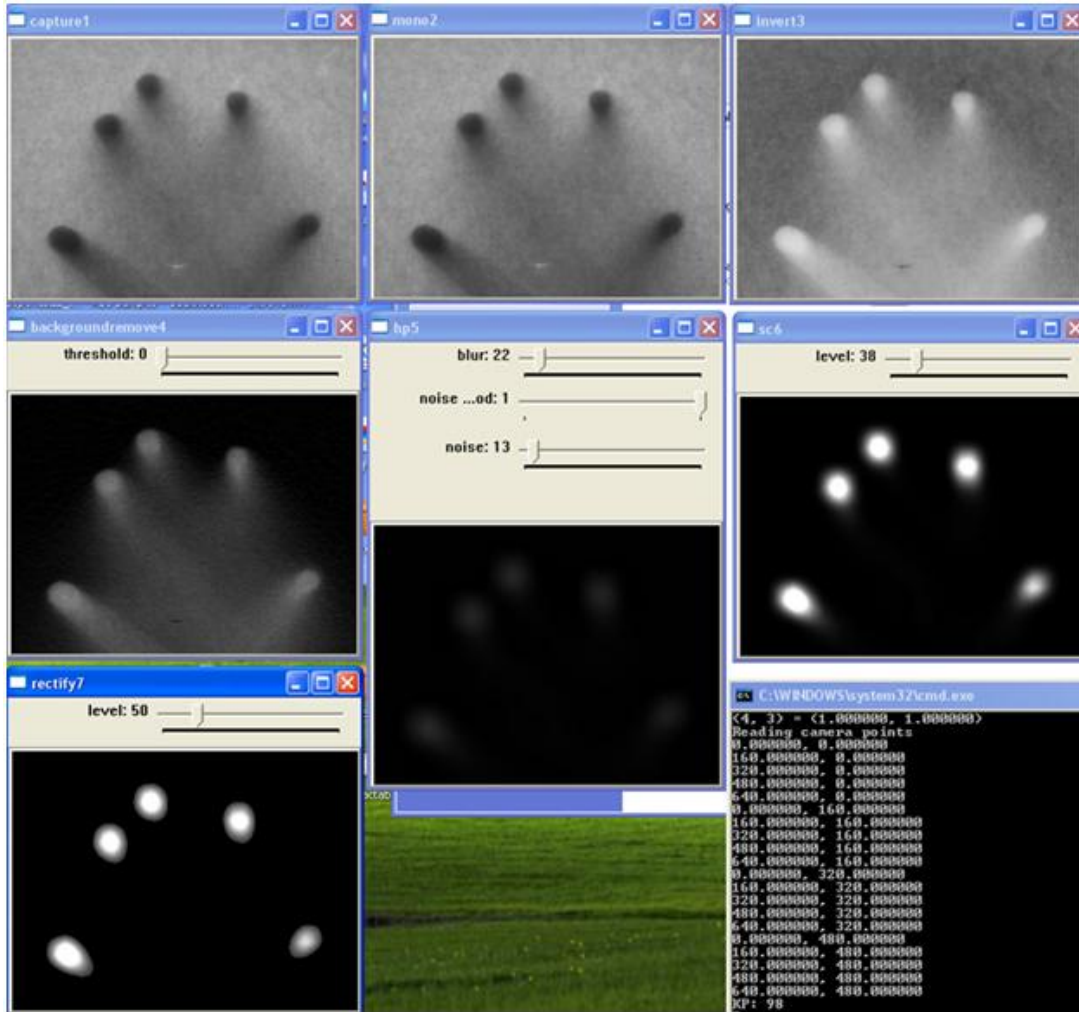


Figure 6-1 Touchlib Configuration Application

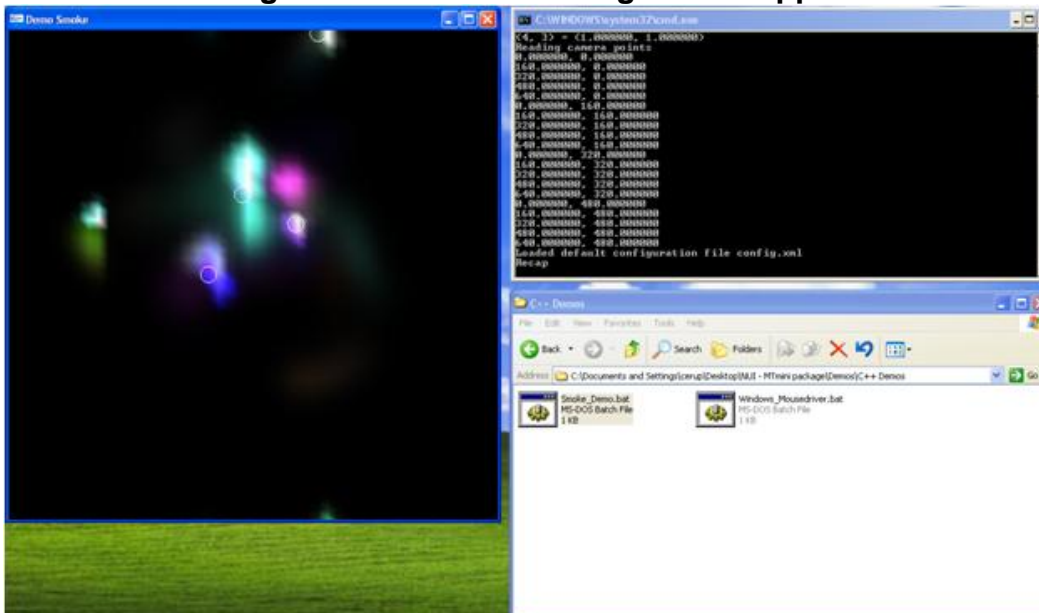


Figure 6-2 Smoke Demo

7.0 Test Plan

7.1 Hardware

The hardware test plan tested all components of the MTPT. The individual components were tested first to ensure basic functionality. Once complete, they were integrated and tested outside of the physical table. The final testing was done when all components are installed in the table. Listed below are the individual test plans for every component of the MTPT.

7.1.1 Screen

The screen was constructed and tested to ensure the proper compliant layer thickness and LED brightness. The IR camera was used to detect proper blob sensitivity on the screen. For the screen to pass its test, the blobs must be easily recognizable to the camera.

7.1.2 IR Camera

The IR camera was connected to the computers USB adapter. The camera was powered up and an IR LED was powered on. The camera should pick up the IR LED to pass.

7.1.3 LED's and Power Supply

The LED array and Power Supply circuit was constructed and wired. All LED's should be functional and the optional LED intensity adjuster was used to fluctuate current in the LED's. The adjustment knob should vary the brightness of the LED's.

7.1.4 Fans and Temperature Sensor

The temperature sensor and fans was wired together and powered using a 12V source. The temperature sensor was calibrated to pass current when the temperature reaches 95 degrees F. The fans should turn on once this temperature is reached and off when the temperature is below 95 degrees F.

7.1.5 Wireless Router

The wireless router was powered using a 110 VAC source. It was then connected to the computer and communication testing took place between the computer, wireless router and the iPhone.

7.1.6 Framework

The framework of the MTPT should be study. A basic push and pull test was conducted to ensure the table is strong enough to be put to use. The enclosure door should also be lockable and open with minimal effort.

7.1.7 Projector

The projector was powered using a 110 VAC source. It was then connected to the computers video card. The display was projected on to the touch screen. The projector screen was calibrated to ensure proper brightness and contrast.

Once all individual component testing is complete, the components were integrated together and basic software testing was completed to test their interactions. The final testing took place when the components are installed into the table framework. This included detailed software testing which is explained in the next section.

7.2 Software

This software test plan outlines the planned testing of the MTPT software suite. The lowest level of testing was done at the class level. Unit testing was done on all suitable classes, ensuring each function works as it should. By unit testing, integration with the software and hardware went more smoothly. Unit testing for the purposes of this project was user driven rather than automated. All actions and results were outputted to a suitable text file to be reviewed against set pass/fail criteria.

Working from the ground up from the smallest unit to the final prototype ensures the quality of the software maintains until full integration with hardware is conducted.

The Software Test Cases outlined in the following sections were run after all unit test cases planned are passed. Each test was traced back to the Software Requirements Specification established in section 3.2. The output of each software test case that does not test graphical requirements outputs a text file containing the marquee strings, noting all Poker Game actions. This output was suitable evidence when declaring a test a pass or fail. The test cases were conducted once final integration with the hardware has taken place. Any redesign or requirement changes required retesting and evaluation. Software test procedures were written once the MTPT software has been developed.

7.2.1 Poker Game Test Cases

The Poker Game test cases test the functionality of the Poker Game Software Components shown in [Figure 2-2](#)~~Figure 2-2~~. After each Poker unit test has passed, all Poker Game software components were integrated. These test cases were performed as a full integration with the MTPT Hardware Components.

7.2.1.1 Case I – Poker Rules

Case I tested the basic functionality of the full game code with the hardware. The main test for the poker rules were poker hand evaluations. The software was tested against the hand rankings established in [Table 4-6](#)~~Table 4-6~~. A test procedure outlines the steps to test all player and game functionality. The requirements that were tested included Game_001 to Game_006. Special cases within a typical No Limit Texas Hold'em game were tested in separate cases.

These special cases included an all in bet, proper side pot handling, and player elimination.

Case I will validate the following software requirements:

- [Game_001] The MTPT will support Tournament style No Limit Texas Hold'em Poker.
- [Game_002] The game will support from two to four players.
- [Game_003] Blinds will raise every 15 minutes as shown in [Figure 2-2](#).
- [Game_004] Each player will start with 10,000 units in chips.
- [Game_005] The chip values that will be used will be as follows:
 - White: 100
 - Gold: 500
 - Black: 1000
- [iPhone_004] Hole cards will be displayed on the iPhone/iPod Touch devices.

7.2.1.2 Case II – All In

Case II tested the software handling of a player going all in, as described in Section [4.3.1.24.3.1.2](#). This case further tests the requirement [Game_001], requiring the software run No Limit Texas Hold'em.

7.2.1.3 Case III – Side Pot

Case III tested the software handling of calculating side pots. The software ensured side pot values are correct and the suitable players win the proper amount of chips. This case further tests the requirement [Game_001].

7.2.1.4 Case IV – Player Elimination

Case IV tested player elimination handling. This test case validated proper player handling when a player loses all of their chips. When a player is eliminated from the tournament, that player position should show an empty betting zone. On the next hand after a player is eliminated, the blind button should act. This case further tests the requirement [Game_001].

7.2.1.5 Case V – iPhone Communication

Case V tested communication between the iPhone and MTPT. This test ensured that the MTPT software only connected to a maximum of four iPhones. Furthermore, this case tested how the MTPT reacted to a dropped connection. Case IV ensures that the game continued despite a dropped connection.

Case V validates the following software requirements:

- [iPhone_001] The MTPT will interface with up to 4 iPhone/iPod Touch devices

- [iPhone_002] The following iPhone and iPod Touch devices will be supported:
iPhone EDGE, 3G, 3GS
iPod Touch 1st Generation, 2nd Generation
- [iPhone_003] The MTPT will support the latest Operation System at the time of this document (3.0).

7.2.1.6 Case VI – Table Interface

Case VI tested the table interface of the MTPT. Every user interaction in the poker game was tested accordingly for all four positions. Also, this case tested the instance when two people attempt to interact with their respective zones at the same time. This handling ensured that only the active player was able to make the appropriate actions. Case VI validated the requirements in section [2.2.1.32-2.1.3](#).

7.2.2 Restaurant Menu Test Cases

The Restaurant Menu Test Cases validate the software requirements established in section [2.2.22-2.2](#). These test cases validate the functionality of the Restaurant Menu Software Components.

7.2.2.1 Case I – Gesture Handling

Case I tested all of the incorporated gestures of the MTPT. This test case was not involved the Poker Game software. The only requirement to run this test is for the menu to be called by the user.

Case I validates the following requirements:

- [Table_016] The Restaurant Menu will be called using the Restaurant Gesture
- [Table_017] This gesture will call the menu to focus
- [Table_018] The Menu will be closed using the “x” button
- [Table_019] The menu may be scaled using a “pinch” gesture
- [Table_020] Paging will completed using a Flick gesture

7.2.2.2 Case 2 – Menu Calling

Case 2 tested the execution of the menu code within a real poker game. This case ensured the menu does not interfere with the main application and should run accordingly.

Case 2 validates the following requirements:

- [Table_013] The menu can be viewed anytime during the game

- [Table_014] The menu will contain actual items from a chosen restaurant/café
- [Table_015] Only one menu will be visible

8.0 Budget

The MTPT budget was compiled after all research and design specifications were acquired. When the group first started the project, the estimated budget was \$375. There was major emphasis to make this device an affordable technology. Every component and part was price matched to cut costs. After all pricing was complete; the total cost was \$412. This was very close to our original estimate but is expected to increase due to unforeseen issues. Therefore, the total group projected budget was \$500. This splits up to approximately \$167 per person. There will be an option to sell the MTPT, when complete, to the highest bidder. The table below shows the components and parts that will be purchased and their prices.

	QUANTITY	UNIT PRICE	TOTAL PRICE	BUDGET	OVER/UNDER
MULTI-TOUCH SCREEN					
DAP Silicione, 10.1fl oz.	1	\$4.50	\$4.50		
Rosco 20"x24"	1	\$12.00	\$12.00		
Acrylic Sheet, 17"x22"	1	\$63.90	\$63.90		
SUBTOTAL			\$80.40	\$100.00	-\$19.60
LED					
IR LED, 880nm	80	\$0.47	\$37.60		
SUBTOTAL			\$37.60	\$50.00	-\$12.40
INTAKE/EXHAUST FAN					
Yate Loon	2	\$4.99	\$9.98		
SUBTOTAL			\$9.98	\$10.00	-\$0.02
IR CAMERA					
Playstation 3 Eye	1	\$34.99	\$34.99		
SUBTOTAL			\$34.99	\$40.00	-\$5.01
POWER SUPPLY					
LED Power Supply Parts	1	\$5.00	\$5.00		
Perf Board	1	\$5.00	\$5.00		
Soder	1	\$3.00	\$3.00		
Power Cord Extention	1	\$10.00	\$10.00		
SUBTOTAL			\$23.00	\$10.00	\$13.00
PROJECTOR					
Mirror	1	\$5.00	\$5.00		
Projector Mount	1	\$25.00	\$25.00		
Optima EP719	1	\$300.00	\$300.00		
SUBTOTAL			\$330.00	\$150.00	\$180.00
FRAMEWORK					
Birchwood 4'x8'	1	\$39.00	\$39.00		
Wood Glue	1	\$3.00	\$3.00		
Pine 1"x6"	2	\$19.25	\$38.50		
Nails	1	\$2.00	\$2.00		
Screws	1	\$2.00	\$2.00		
Wheels	4	\$2.00	\$8.00		
SUBTOTAL			\$92.50	\$50.00	\$42.50
TOTAL			\$688.87	\$410.00	\$198.47
			x100		
Cost of 100 Units			\$68,887.00		

9.0 Milestones

SEMESTER I		Week								Planned Start	Planned Finish					
MILESTONE	Name	3	4	5	6	7	8	9	10							
Research Project	ALL	X														
Initial Project and Group Identification Document	ALL	X														
Project Design	ALL		▲								6/7/2009	6/21/2009				
Research/Design/Cod e	Boucher/ Timones		▲								6/14/2009	6/28/2009				
Research Hardware Components	Herod/ Boucher		▲								6/12/2009	6/28/2009				
Component Integration Review	ALL				X		▲				6/21/2009	6/24/2009				
Find/Order Parts	Herod/ Boucher						▲				6/28/2009	7/14/2009				
Initial Document Review	ALL						X				7/5/2009	7/10/2009				
Final Document Review	ALL							X			7/08/2009	7/09/2009				
Hand In Document	ALL									▲	7/10/2009	7/10/2009				
Aquire Parts	ALL									▲	7/05/2009	7/21/2009				
Code/Unit Test	ALL						▲				7/05/2009	9/28/2009				
Assemble Hardware	ALL									▲	8/07/2009	9/28/2009				
SEMESTER II		SEMESTER II Week												Planned Start	Planned Finish	
MILESTONE	Name	1	2	3	4	5	6	7	8	9	10	11	12			
Aquire Parts	ALL	▲													7/05/2009	7/21/2009
Code/Unit Test	ALL													7/12/2009	9/28/2009	
Assemble Hardware	ALL	▲												8/07/2009	9/28/2009	
Software Integration	Boucher/ Timones			▲										8/14/2009	10/12/2009	
Testing	ALL						▲							10/5/2009	11/02/2009	
Re-Design/Review	ALL											X		11/02/2009	11/12/2009	
Presentation	ALL												X	11/12/2009	11/30/2009	

10.0 Appendices

11.1 Copyright permissions

-----Original Message-----

From: linear [<mailto:linear@linear1.org>]
Sent: Monday, August 03, 2009 1:13 PM
To: Herod, Christopher A (Chris)
Subject: Re: copyright permission

Herod, Christopher A (Chris) wrote:

> Hi I'm a Senior at the University of Central Florida majoring in
> Electrical Engineering. As a senior design project, my group is
> building a multi-touch poker table. I found your LED wizard online
> and would like to use a schematic formulated by your program in our
> documentation and design. Full credit will be given to your website.
> If you have any questions please contact me.
>
> I also have a question about building an IR LED array. I would like
> to adjust the intensity of the array with a dial of some sort. Do you
> know of any (inexpensive) way or device that can control the current
> through the LED array with a dial. A potentiometer would be ideal but
> I'm not sure how to keep the voltage constant. Could I add a voltage
> regulator and attach a pot between the regulator and the load? I'm
> powering the LEDs with a simple low voltage linear power supply. Any
> information would be appreciated. Thank you
>
> Chris Herod
>
> chris_herod@hotmail.com

Hi Chris,

Permission granted, you may reproduce the output for any purpose you like. Good luck with the project. My senior design class was 13 years ago, but I know what you're up against and I'm happy to help if I can.

I'll get a detailed answer on the second question in your hands shortly, but the short answer is: use PWM because they are quite nonlinear with current.

all the best,
Rob

-----Original Message-----

From: Jeff Mucha [<mailto:chugga@magic.skylab.org>]
Sent: Monday, August 03, 2009 12:04 PM
To: Herod, Christopher A (Chris)
Cc: mucha@skylab.org
Subject: Re: copyright permission

You have my permission to use my diagrams and material in your project.

Thank you for the exposure!

-Jeff Mucha

On Mon, 3 Aug 2009, Herod, Christopher A (Chris) wrote:

> Hi I'm a Senior at the University of Central Florida majoring in
> Electrical Engineering. As a senior design project, my group is
> building a multi-touch poker table. I found your car MP3 player
> online and would like to use the ATX figure diagram in our
> documentation to explain the power supply. Full credit will be given
> to your website. If you have any questions please contact me. Thank
> you
>
> Chris Herod
>
> chris_herod@hotmail.com
>
>

From: Gökrem Çetin [mailto:gorkem.cetin@gmail.com]
Sent: Monday, July 13, 2009 4:43 PM
To: Herod, Christopher A (Chris)
Cc: the-book@nuigroup.com
Subject: Re: Documentation permission

Dear Chris

Thanks for letting us know. You may also use the ISBN number 978-0-578-03156-9 to refer to the book for citation purposes.

Best
Gökrem

2009/7/13 Herod, Christopher A (Chris) <Christopher.A.Herod@usa-spaceops.com>

Hi I'm a Senior at the University of Central Florida majoring in Electrical Engineering. As a senior design project, my group is building a multi-touch poker table. I found your Multi-Touch Technologies book online and would like to use figures 2 through 5 in our documentation to explain the different technologies. Full credit will be given to your group and website. If you have any questions please contact me. Thank you

Chris Herod

Orbiter Mechanisms Engineering
321-861-3227
Christopher.A.Herod@usa-spaceops.com



Re: Copyright Permission Senior Design

Wednesday, August 5, 2009 11:09 PM

From:

"Seth Sandler" <cerupcat@gmail.com>

To:

"Reggie Timones" <r.timones@yahoo.com>

Hi Raeginald,

No problem. Feel free to use the images as needed. If you're working on is public, let me know, i'd love to check it out.

-Seth

On Wed, Aug 5, 2009 at 7:51 PM, Reggie Timones <r.timones@yahoo.com> wrote:

I am a Computer Engineering student currently working on my senior design documentation. May I reproduce your screen shots(Figures 2, 3) in your getting started v1.0 .pdf?

Thank you,

Raeginald Timones

--

seth sandler

Web: sethsandler.com

Twitter: sethsandler

Skype: sethsandler

11.2 Software Licenses

Touchlib New BSD:

<OWNER> = Regents of the University of California

<ORGANIZATION> = University of California, Berkeley

<YEAR> = 1998

In the original BSD license, both occurrences of the phrase "COPYRIGHT HOLDERS AND CONTRIBUTORS" in the disclaimer read "REGENTS AND CONTRIBUTORS".

Here is the license template:

Copyright (c) <YEAR>, <OWNER>

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

Neither the name of the <ORGANIZATION> nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Pokersource GNU General Public License V3 or later:

GNU GENERAL PUBLIC LICENSE

Version 3, 29 June 2007

Copyright © 2007 Free Software Foundation, Inc. <<http://fsf.org/>>

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program--to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS

0. Definitions.

"This License" refers to version 3 of the GNU General Public License.

"Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

"The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

To “modify” a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a “modified version” of the earlier work or a work “based on” the earlier work.

A “covered work” means either the unmodified Program or a work based on the Program.

To “propagate” a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To “convey” a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays “Appropriate Legal Notices” to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

The “source code” for a work means the preferred form of the work for making modifications to it. “Object code” means any non-source form of a work.

A “Standard Interface” means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The “System Libraries” of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A “Major Component”, in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The “Corresponding Source” for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work’s System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work’s users, your or third parties’ legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program’s source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

- a) The work must carry prominent notices stating that you modified it, and giving a relevant date.
- b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to “keep intact all notices”.
- c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.

d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.

b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.

c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.

d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.

e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

"Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or

b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or

- c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or
- d) Limiting the use for publicity purposes of names of licensors or authors of the material; or
- e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or
- f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered “further restrictions” within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An “entity transaction” is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party’s predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A “contributor” is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor’s “contributor version”.

A contributor’s “essential patent claims” are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, “control” includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor’s essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a “patent license” is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To “grant” such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. “Knowingly relying” means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient’s use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate,

modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is “discriminatory” if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License “or any later version” applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the “copyright” line and a pointer to where the full notice is found.

```
<one line to give the program's name and a brief idea of what it does.>
Copyright (C) <year> <name of author>
```

This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the

GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program. If not, see <http://www.gnu.org/licenses/>. Also add information on how to contact you by electronic and paper mail. If the program does terminal interaction, make it output a short notice like this when it starts in an interactive mode:

```
<program> Copyright (C) <year> <name of author>
This program comes with ABSOLUTELY NO WARRANTY; for details type `show w'.
This is free software, and you are welcome to redistribute it
under certain conditions; type `show c' for details.
```

The hypothetical commands `show w' and `show c' should show the appropriate parts of the General Public License. Of course, your program's commands might be different; for a GUI interface, you would use an "about box". You should also get your employer (if you work as a programmer) or school, if any, to sign a "copyright disclaimer" for the program, if necessary. For more information on this, and how to apply and follow the GNU GPL, see <http://www.gnu.org/licenses/>. The GNU General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License. But first, please read <http://www.gnu.org/philosophy/why-not-lgpl.html>.

Qt GNU LGPL v. 2.1 Version

This version is available for development of proprietary and commercial applications in accordance with the terms and conditions of the GNU Lesser General Public License version 2.1.

Support services are available separately for purchase.

Benefit to Qt

The benefit to Qt is that any changes made to Qt must be made available under the terms of this license.

Qt GNU GPL v. 3.0 Version

This version is freely available for the development of open source software governed by the GNU General Public License version 3.0 ("GPL").

This version is available for the development of open source Software for Windows, Linux, Unix, and Mac OS X under the GPL. Please note the following:

- We provide the Qt GNU GPL v. 3.0 version with no warranty. Support is available separately for purchase.
- The Qt GNU GPL v. 3.0 version may be freely copied and distributed, put on FTP sites and CD-ROMs, etc. under the terms of the GPL.
- Based on the "Quid Pro Quo" principle, if you wish to derive a commercial advantage by not releasing your application under an open source license, you must purchase an appropriate number of commercial licenses from us before you begin developing commercial software. By purchasing commercial licenses, you are no longer obliged to publish your source code.
- If you wish to use the Qt Open Source Edition, you must contribute all your source code to the open source community in accordance with the GPL when your application is distributed.

Benefit to Qt

The benefit to Qt is that any changes made to Qt must be made available under the terms of this license.

11.3 Works Cited

- [1] Ciaffone, Robert. "Robert's Rules Of Poker". Harrah's License Company, LLC. 2009. < http://www.worldseriesofpoker.com/learn/rules_holdem.asp>

- [2] Devlin, James. "The Great Poker Hand Evaluator Roundup". 05 September 2008. Coding the Wheel. < <http://www.codingthewheel.com/archives/poker-hand-evaluator-roundup>>

- [3] Muller, Laurence. "Unofficial Touchlib Reference. Multigesture.net. 20 June 2008.

- [4] NUI Group Authors. Multi-Touch Technologies. 1st edition[Community Release]. May 2009. 2009 NUI Group. < <http://nuicode.com/projects/wiki-book/files>>.

- [5] "Qt 4.4 Whitepaper". Nokia Corporation. 2008.

- [6] Sandler, Seth. "How to Setup Touchlib and Run Multitouch Demos using a MTmini. Version 1.0. <http://ssandler.wordpress.com/MTmini>

- [7] World Series of Poker. "Hand Strength". Harrah's License Company, LLC. 2009. <<http://www.worldseriesofpoker.com/learn/hands.asp>>