Plank: Multi-User Touch Table

Peter Oppold, Hector Rodriguez, Chris Sosa, Enrique Roche

School of Electrical Engineering and Computer Science, University of Central Florida, Orlando, Florida, 32816

***Abstract* —This paper describes the design of a multi-user touch table and software application capable of supporting multiple simultaneous users. The table uses diffused surface illumination (DSI) technology and is capable of both touch recognition and object recognition. Two pieces of software were implemented. The first interprets touch and object information transmitted from a vision library. The second is an implementation of a tower defense game.**

***Index Terms* — diffused surface illumination, multi-touch, infrared, fiducial**

I. Introduction

As large companies such as Microsoft, Samsung, Motorola, RIM, and Apple gravitate towards creating intuitive touch technologies for personal devices the feasibility of large multi-user touch surfaces replacing traditional input is becoming more commonplace. Planck will be capable of using both human touch and supplementary objects to replace traditional forms of input used for home computers. Planck's aim is to offer a more intuitive, high throughput, input scheme that allows multiple users to learn while doing and achieve a common goal cooperatively. To showcase these ideas Planck will feature a defense scenario application, named weDefend, which requires users to defend assets via placement of military units. It will offer a 40" display where multiple users can interact with different graphical as well as tangible objects to achieve the common goal of defending an area or asset.

A. Image Recognition System

The table utilizes a multi-touch technology known as Diffused Surface Illumination (DSI). This technology is similar to two earlier technologies known as Frustrated Total Internal Reflection (FTIR) and Diffused Illumination (DI)[3], but holds several distinct advantages. The main advantage of using this type of technology over FTIR is that it is capable of object recognition. DSI also does not require a compliant material in order to obtain accurate information from drags and pinches. One of the disadvantages of DSI is the increased cost implementation.

The technology works thanks to special edge illuminated acrylic. This acrylic was originally designed for signage, which could be illuminated from one or more sides, rather than from the back of the acrylic. Infrared light is illuminated into the edge of the acrylic and that light is evenly dispersed over the entirety of the acrylic layer. A camera is placed below the acrylic layer. Objects above the acrylic reflect light back down, which can then be picked up by the camera below as touches and objects. All of this information is processed by a computer using a vision library and interpreted in a program, which then outputs through a projector onto a separate piece of rear projection acrylic. This process is shown in Fig 1.

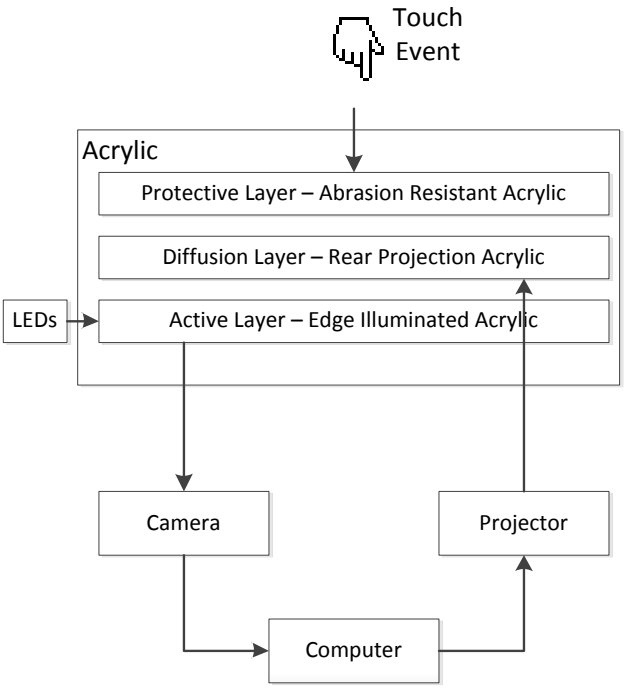

Fig 1. Overview of hardware involved in Diffused Surface Illumination touch screen.

B. Why Tower Defense?

The design of weDefend had to emulate an eagle eye view application capable of allowing multiple commanders to assess situations, give orders to military assets, and work in a team to accomplish a mutual objective. Drawing inspiration from the interactive entertainment industry a genre of game called real time strategy seemed an appropriate template due to many similarities between the

genre and the requirements of weDefend. The idea was to make a multiple user version of a real time strategy game. Once work on that design began it quickly became apparent that the massive complexity of having a multitude of menus and commands necessary to create a real time strategy application was out of scope with what would be accomplishable in the time given; It also did not blend nicely with how fiducials would become a part of the application.

The alternative design came from a sub-genre of real time strategy called tower defense. These applications consist of a simple yet effective goal which could easily be adapted to a multiple user paradigm. The modified multiplayer tower defense was not a new idea. However, adding multiple users who can control the application simultaneously was unique. The modified tower defense application would maintain the same simple objective of defending an asset on the map; however it would now be a mutual interest to all the users while still maintaining the idea of having multiple officers commanding military assets simultaneously.

C. Fiducial Markers

Fiducials are symbolic markers that have a specific meaning to our software. They are used to interact with the Showcase Application Software to trigger specific events, stamp out objects, and to locate certain objects on-screen. Position, orientation, velocity and acceleration metadata is available. Fig 2 shows several different types of fiducials. CCV 1.5 has implemented Reactivision's[4] 'amoeba' fiducials (d in the Fig 2) in order to speed up recognition time of the pattern on the surface.

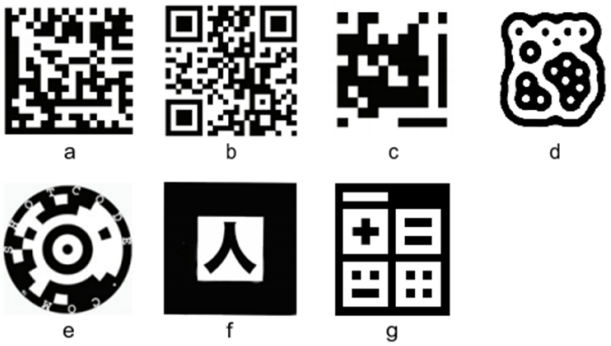

Fig 2. Examples of fiducial markers[1]

II. HARDWARE OVERVIEW

There were two primary purposes for the hardware. The first was to implement the system for touches. This required that the use multiple layers of acrylic, a source of infrared light which could illuminate the acrylic, and a camera to read the touch information.

The second purpose was to facilitate the interaction of multiple users for the showcase software. The table needed to be big enough to facilitate multiple users. It also had to stay cool enough to be used for extended periods of time. For familiarity for the users, the design of the touch screen was also made to resemble that of a touch screen smart phone. Lastly, the constraints of the project required that it also could be transported so that it could be presented. This required that it was small enough to fit through a door.

A. Acrylic Materials

The only type of acrylic that is required for this technology to work is the edge illuminated acrylic. But multiple layers of acrylic were used to achieve the final goal. The edge illuminated acrylic selected for this project was Endlighten XXL, with a 10 mm thickness. This acrylic is rated for a 39 inch penetration depth for visible light from each side of the acrylic. Research from other projects has shown that this depth is accurate for the infrared light being used, so this grade was chosen over the cheaper EndLighten XL, which has a shorter penetration depth.

A second layer of acrylic was used as a rear projection material, so the image from the projector can be seen. Because this layer diffuses light, this does have an impact on the quality of the infrared image used to obtain touch and object information. The material chosen was 7D513 Rear Projection acrylic. A third layer was used as a protective layer for the users to interact with. This is preferable because the Rear Projection acrylic and the EndLighten acrylic are softer and susceptible to scratching, which would begin to interfere with the quality of images being detected. Evonik's 0A000 MR2 mar-resistant acrylic was used for this layer.

B. Infrared Source

Because the infrared light is being dispersed over the surface of the acrylic, it was important to have a strong source of infrared light. Due to the construction of the table, it was also important that this light source be low profile. It was also important that as much of the light entered the acrylic as possible. This required that the source has a low half angle. Lastly, in order to have optimal visibility for the camera, the wavelength needed to be between 850 and 900nm.

It was decided that surface mount LEDs would be used. SFH4258 LEDs were chosen. These have a half angle of 15 degrees, an 850nm wavelength, and supply 50mW of radiant flux at constant current. Each LED requires

100mA at 1.5V. Because the input from the power supply unit was 12V, it was decided to use chains of 7 LEDs and a current controlling resistor. These chains would then be placed in parallel and run along the length of the board. The surface mount LEDs would be mounted on PCBs. Each PCB would contain between 2 and 4 chains, which could be daisy chained to cover the entire border of the acrylic.

C. Camera

A camera, that's compatible with the computer, must be selected to capture images of the touches and objects interacting on the IR illuminated acrylic. This camera would need to be able to read near infrared light. In order for the system interface to have sufficient responsiveness, a frame rate of 60 fps was the ideal goal. A lower frame rate would be noticeable when dragging objects on the screen. Objects on the screen would lag behind the finger as the finger was dragged. In order to properly read objects on the screen, it was also important for the camera to have as high of a resolution as possible. It was decided that in order to meet viewing distance requirements, two cameras would also be used.

Several types of cameras were considered, including Firewire cameras, the PS3 Eye, and common webcams. Firewire cameras had the best picture quality, but they were also significantly more expensive and outside of the project's budget for the frame rate and resolution requirements we needed. The PS3 Eye is the most common camera used in previous projects, but the camera functions ideally at low resolutions. This is partially due to the hardware constraints of the camera, and partially due to the pixel format it uses, which is 16 bits per pixel. Webcams contain filters on them which block infrared light, but this filter could be removed on most cameras. Their picture quality is notably lower than Firewire camera, but they were less than one tenth the price, for the same specifications. Logitech HD Pro C910 webcams were used. These were capable of 640x480 at 60 frames per second as well as 1184x768 at 30 frames per second. These used a YUV12 pixel format, which is only 12 bits per pixel.

The infrared blocking filter could easily be removed from the camera, but, without a new filter to block the visible light, the touchscreen would not be able to be used unless it was in a very dark room. Cold mirrors were purchased which could be placed over the cameras. These reflect visible light, while allowing infrared light to pass through. This allows for only the relevant information to be viewed by the camera. Unfortunately, infrared light from other sources would still cause interference on the screen. This means that under some conditions, such as direct fluorescent lighting, touches may not be able to be read.

D. Control System

The table features a microcontroller circuit (see Figure 3) to allow for control over the sensitivity of the board as well as the speed of the fans. This system receives power from the 12 volt rail of the power supply unit from the computer. Two linear regulators are used to convert this 12 volt signal into a usable signal by the rest of the components. A TPS71633 was use for the 3.3V components and a TPS71650 was used for the 5.0V components.

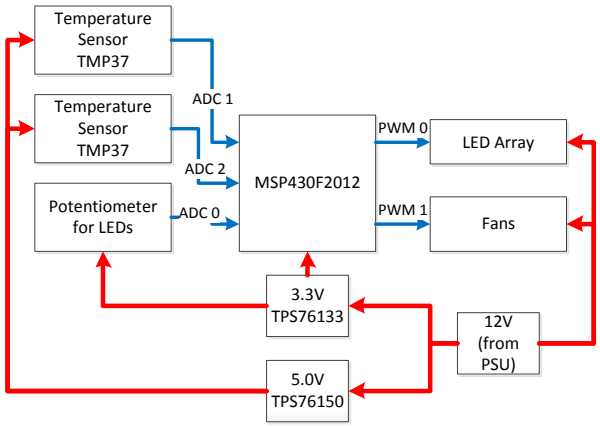

Fig 3. Overview of control Circuit.

The central brain of the control system is an MSP430G2231 microcontroller. This microcontroller takes input from two remote temperature sensors and a potentiometer. It then converts these analog signals into a pulse width modulated signal, which is used to control the LEDs and the fans on the system. These signals are then amplified by a series of transistors before going to the remote components.

The temperature sensors used were TMP37 linear temperature sensors. These take a 5V signal, which is then fed through a PNP transistor to amplify the output signal of the temperature sensor enough to reach the microcontroller. The two temperature sensors are placed away from the microcontroller at the two locations that create the most heat within the device. One will be placed near the motherboard of the computer, and the second will be placed near the projector.

E. Image Display

Planck uses the Hitachi CP-AW251N short-throw projector to provide the display image for viewing the showcase application. Much research was done on the image display but the feasibility of a short-throw projector

could not be over-looked. No mirrors need to be used to maximize throw distance; the enclosure height is not of concern as short-throw projectors can achieve a big image in a small space; and there is no risk of damage as no hardware components needs to be removed and re-layered as in LCD televisions. The short-throw projector will display a 720p image.

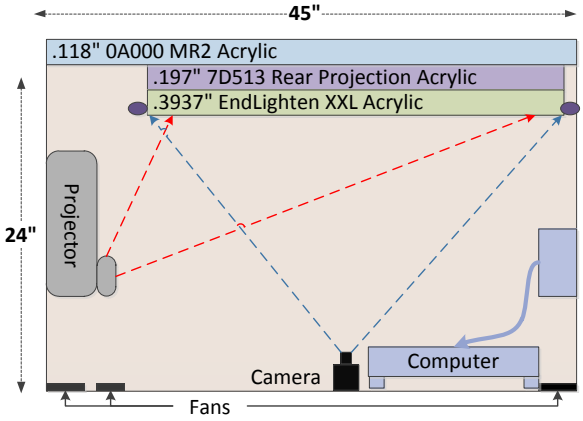

Fig 4. Internal Overview of Table

The projector is mounted vertically against the wall of the enclosure (see Figure 4). Collecting information from the manufacturer's manual, the vertical offset and throw distance required to achieve a 45" diagonal display were found to be a minimal 0" throw distance, but an 11.5" vertical offset. Upon further testing, it was discovered that the intake was blocked when mounted directly against the wall. To counter-act this, the projector was mounted at an angle and further away from the wall. The keystone effect was produced, but was corrected using built-in Hitachi software. The correct 45" display size was achieved through the zoom function.

F. Computer

To ensure that the computer would never be a bottleneck in the system, high-end off-the-shelf components were chosen. The processor chosen is a quad-core Intel i7. The video card supports DirectX 11. To effectively take advantage of Intel's aggressive prefetching scheme, the computer has 8GB's of main memory installed on the system. An SSD was chosen to boot up with minimal delay. Finally, a 700w power supply was chosen to effectively power the computer as well as the LED's that encompass the acrylic. The computer is mounted on the floor of the enclosure to an internal frame of a standard computer case (see Figure 4).

G. Enclosure

The enclosure was constructed out of ¾" Oak veneer. Oak was chosen because of it being a strong hardwood and it being visually appealing at a low price point. Veneer was chosen because of its cost effectiveness and because it comes in large 4'x8' sheets.

The enclosure was constructed essentially as an open-faced box with a full acrylic screen, much like a smart-phone. The IR LED's that surround the acrylic were mounted inside a channel that encompasses the acrylic on all sides. Eliminating the top-frame of the box shortens the border and allows the surface acrylic to span edge to edge on the enclosure.

Legs were designed and built to provide the needed 36" height the system requires. Rather than making the enclosure bigger, the enclosure will be built to be as minimal as possible and the use of legs will be implemented. Legs became a necessity as the enclosure would not fit through the door without a smaller enclosure and removal legs. An extra benefit to this design is the possibility of shortening the system on a short notice. The legs were designed to be removed with minimal work. The legs were designed and constructed to withstand 400lbs of weight.

The three layers of acrylic are sandwiched at the very top of the box. The most inner layer rests on wooden mounts inside the box. This is level with the channel that the LED's are mounted in. The next layer is the exact size of the first and is mounted freely directly on-top. The last mar-resistant layer rest directly on-top of the box and is secured with acrylic rods that have been welded onto the acrylic layer. Holes were cut in the box, and the acrylic rods lock into each hole. This prevents movement of all three acrylic layers. The layering process and the LED channel can be viewed in Fig 5.

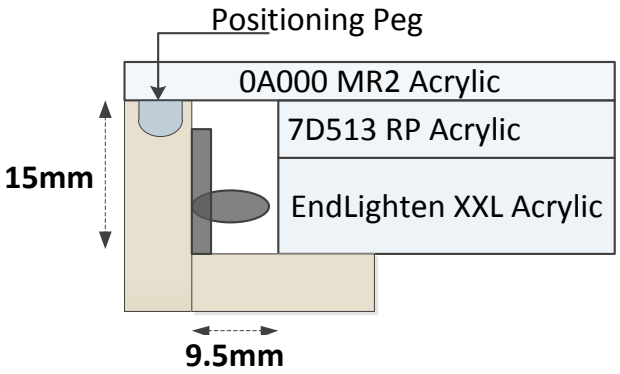

Fig 5. Positioning of LED Channel.

H. Cooling

Three fans were mounted inside the enclosure to dissipate built up heat from the various hardware devices. The fans will connect to the MSP40 in the Control System for power and will receive fan control based upon the MSP40's temperature sensors. The fans chosen are two Artic cooling AF12PWM 120mm case fans and one Panasonic Panaflo 80mm. The AF12PWM delivers 57 CFM at 1350rpms with 0.5 Sone of noise being produced. They have a good longevity and low noise output due to using fluid bearings. The AF12PWM also has the fourth wire for Power Width Modulation (PWM) that the microcontroller will use to control fan output. All fans will be mounted from the bottom of the enclosure for a stealth appearance. The two AF12PWM fans were mounted directly next to the power supply and the projector. This is the direct ventilation for these two devices, as they are the hottest in the system. The Panaflo 80mm fan is mounted with a PVC pipe that will draw air from the top of the enclosure. This grabs any hot air collecting at the top of the enclosure. Numerous holes were cut in open areas of the floor to allow for fresh air intake. The fan placement can be viewed in Fig 4.

III. SOFTWARE OVERVIEW

The software for the system consists of two main systems. The first is the Touch and Fiducial Recognition System. This system takes the infrared images from the camera and converts that into a usable data structure that the program may use. Within this system are two main components, the vision library and a class which packages the data.

The second system is the Showcase Application Software. This is the actual interactive program which showcases the different functions of the table and outputs visual data back to the user. An overview of the flow of these two systems can be seen in Fig 6.

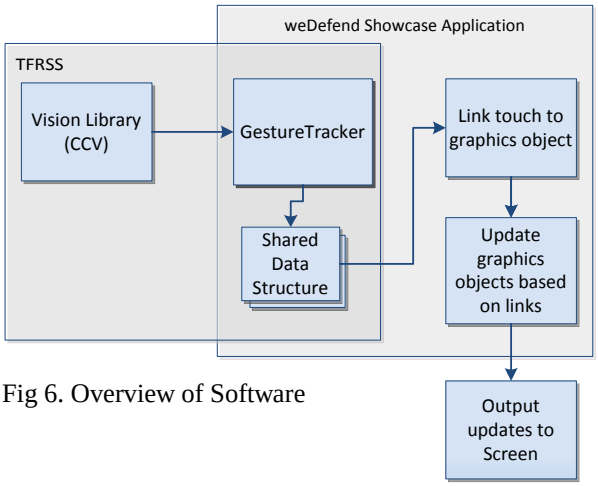

Fig 6. Overview of Software

A. Touch and Fiducial Recognition Software System

The Touch and Fiducial Recognition Software System (TFRSS) is comprised of two main parts. The first part, CCV 1.5, is a piece of software that interprets the information received from the camera. If there is more than one camera being used as input, CCV takes care of stitching those two input signals together. CCV processes the input video which contains images of the touches and Fiducials found on the surface of Planck to create as output normalized touch objects and fiducial objects. Each touch object contains relevant description metadata such as its touchID number, normalized X-Y location point data, etc. Fiducials are also decoded and associated metadata about the fiducial's position, orientation, speed, as well as it's unique ID, etc. are acquired and added to the object[2]. Touch and fiducial objects are output from CCV using the TUIO protocol via network socket on port 3333 to the second part of the TFRSS, GestureTracker.

GestureTracker is a class written in C# that listens to the output of CCV. Its object is declared as a thread of the Showcase Application Software in order to simplify inter-process communication between it and the Showcase Application Software. The messages received from CCV, via the network, are decoded using a TUIO socket listener/decoder to create touch and fiducial objects. The attainment of these objects in GestureTracker triggers events that process them, allowing them to be used in the Showcase Application Software.

There are six primary events processed by GestureTracker. Three of these involve finger touch objects and the other three involve fiducial objects received from CCV. The three events that can occur with touch objects are:

1) A new touch was added to the surface
2) A touch was removed from the surface
3) An existing touch was moved on the surface and therefore needs to be updated.
4) A new fiducial was added to the surface
5) A fiducial was removed from the surface
6) An existing fiducial was moved on the surface and therefore needs to be updated.

Every fiducial has a unique ID number associated with its pattern. There can be more than one instance of the same fiducial located on the surface. Because of this a fiducial ID and a unique session ID are included to assist in differentiating one instance of a fiducial from another identical fiducial. Two more support methods exist within GestureTracker. These are public methods tasked with sharing the two shared data lists of touch and fiducial objects to the showcase application.

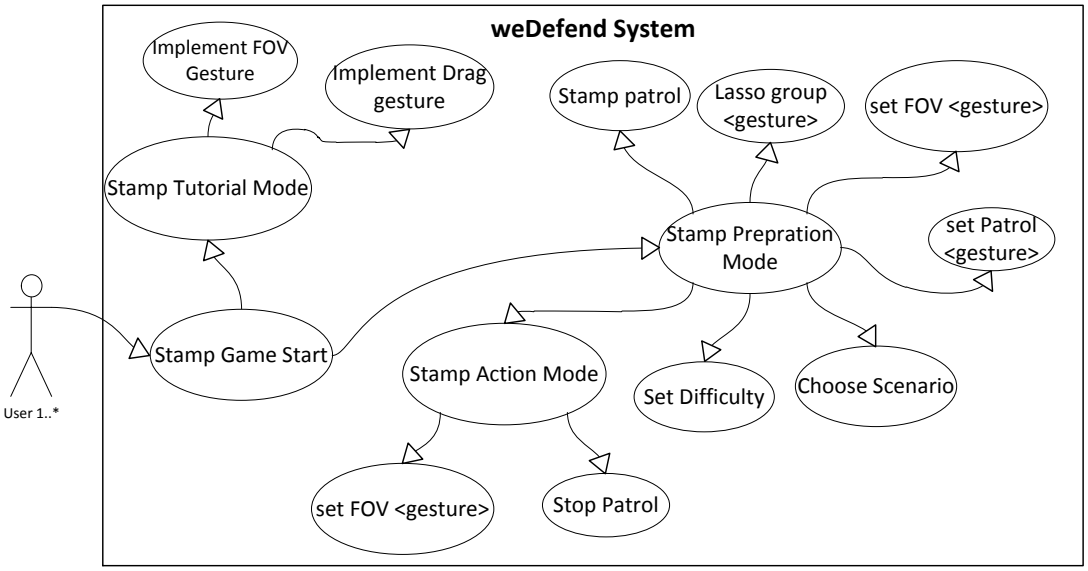

Fig 7. Use Case for WeDefend

As information is received from CCV and the touch objects and fiducials are added, updated, and removed from the lists, these two lists are shared with the Showcase Program entitled weDefend through the respective methods. The two lists comprised of touch objects and fiducial objects are the sole medium used to communicate input information from the surface to the weDefend application. WeDefend uses the touch and fiducial objects inside these two data structures as input in order to manipulate the graphical objects on the surface. To prevent weDefend from interfering with the shared data structure functions of GestureTracker, a simple lock mechanism was implemented. The lock mechanism works as follows: whenever a new object is being added, updated, or removed from the lists, the list being manipulated is locked to any other methods needing access. Thorough testing of our software has illustrated that the latency introduced by locking the lists is negligible. Two public methods within GestureTracker give access to the shared data structures. Copies of the structures are returned to the program requesting them in order to avoid further thread atomicity issues.

B. Showcase Application Software

WeDefend leverages Microsoft XNA 4.0 as its rendering platform. The GestureTracker class provides a shared data structure in which XNA can then see touch and fiducial objects. XNA provides a class called game1.cs, which contains the two workhorse functions that every rendering API provides: update() and draw(). It also provides many helper classes for 2D and 3D graphics such as Vector, Matrix, Texture, Sprite, and Font, all of which are utilized by weDefend.

The application uses a simple finite state machine whose transitions are triggered by fiducial objects to change modes. Once in the preparation phase a user can manipulate soldiers using touches and stamp out soldiers using fiducials. Each class of interactive object has its own respective update and draw methods. The update method runs the linking algorithm discussed above and updates the objects based on a linked touch if such a touch exists. The draw method then draws the objects with their updated fields. For each soldier that exist in weDefend, its respective update and draw methods are called in the game1.cs update() and draw() methods. The field of view and patrol route puck objects exist as instances inside the soldier class and exhibit similar behavior. Their respective update and draw methods are called within the update and draw methods of the soldier who owns them. In this way each object independently checks for its own input. However, objects that are owned by others are limited in that their update and draw methods are never called if their owner is not being manipulated. The linking algorithm is essential for any object that is going to be manipulated via touches. Therefore every interactive touch object's update method is passed in a copy of the touch list provided by GestureTracker.

The game play itself consists of activating the action state while in the preparation state by using a fiducial and then using the field of views of the soldiers placed to fend off insurgents from entering a restricted area on the map, denoted by a green translucent rectangle. If enough insurgents enter this area the game ends. Insurgents can be

killed when they enter the field of view of a soldier at which time they will be fired upon and die after being hit twice. Insurgents are spawned probabilistically based on a random variable. The insurgents always move towards the restricted area at a set speed based on the game clock.

C. Multi-User Input Collisions

The idea of having multiple users on one input device is similar to the age old problem of having too many independent actions trying to control one asset. The simplest way to avoid this problem is to allow users to have ownership over their input. Unfortunately using hardware or some other specific solution to this problem was beyond the scope of this project. Therefore some means of mitigating input collision had to be handled within the software itself.

The discussion to this solution will be limited to how weDefend handles mitigating multiple touches which are essentially experienced by the system as disembodied inputs that don't belong to any specific function but act upon the application based solely on where the touch is.

weDefend contains three basic interactive graphical objects that can be manipulated. The three objects include soldiers, a soldier's field-of-view (FOV), and a patrol route puck. In a traditional input scheme a method called picking would be used to allow the system to discern what the user is hovering over or clicking on. Once an object is picked it activates a new input state which includes a sub-set of commands that belong to that specific object. If this kind of input were possible in weDefend it would mean that users would have to wait for a user to finish with that object sub-set before the next user could do what he/she pleases. This would essentially break up the application into discrete sets of time where only one user can affect the application. This essentially renders the multi-user paradigm useless.

Instead of completely abandoning picking, weDefend modifies it so that it can be used with multiple users. In a very simple and elegant solution each object has its own independent input state that is triggered by touches and regulated by what the master state of the application is. For instance, if weDefend is in the preparation phase then it is possible to drag a soldier, adjust a soldier's FOV, and/or create a patrol route for a soldier and still be able to do all these actions simultaneously to any other soldier on the screen. Collisions are handled by imposing that a soldier must have a finger linked to it in order to adjust its FOV or create a patrol route and by linking touches to soldiers. Touches are linked to a soldier by seeing if the coordinates of the touch in application coordinates fall within the radius of a soldier. At that point the linking algorithm will either detect that a touch is still linked and update the soldier based on the touch input, or detect that the touch that was linked is no longer on the board and unlink the soldier from that touch. This algorithm runs for each interactive object every time the screen is redrawn. Once a touch is linked it creates a sense that the soldier object has been picked and is now in a state where it's FOV and patrol route puck can now be interacted with. The collisions for multiple FOVs and pucks are handled in the exact same way as soldiers; the only difference being that FOVs and pucks are independent objects that are owned by soldiers. However, both the FOV and puck only run the linking algorithm if their associated soldier is currently linked as a touch. They are treated as independent objects in terms of input, as long as the soldier is in a linked state.

IV. CONCLUSION

The primary goals of Plank: that the table implement a non-traditional input system through touch and object recognition, and the device be of sufficient size to accommodate multiple users collaborating simultaneously, were met within the two-semester time period.

It was found that the method used for implementing multi-touch recognition has several limitations that may prevent it from becoming main stream. The technology does not work ideally in well-lit areas. Many types of lighting, including fluorescent light, produce significant amounts of infrared light which may prevent touch events from properly being read. Also, projectors and rear projection materials do not give ideal pictures in well-lit areas. This means that this technology would be best suited for dimly lit lobbies and bars, rather than being fully usable in well-lit offices and homes.

It may be possible to implement touch recognition further into the infrared spectrum. Using LED lighting or similar instead of fluorescent in the room may cause less interference with the recognition. It may also be more beneficial to integrate the system with LCD technology or Plasma technology, rather than projection.

Thread atomicity played a larger factor in production of software than initially expected. None of the members had any direct experience with multithreaded programs. It may have been more efficient to implement a message passing method, rather than the type of locking implemented.

In general, it was found that fiducial recognition was not perfect. Often, fiducial recognition would drop out. This was observed most often when fiducials were being moved around on the screen. Different types of fiducials may be needed in order to get more consistent recognition. This can be compensated for with intelligent software, but may not be able to be completely solved.

ACKNOWLEDGEMENT

The authors wish to acknowledge Dave Kotick and Ron Wolff for mentoring us through our multi-touch journey. Our project has certainly benefited from their advice and we are very grateful for the time they have contributed.

We would also like to thank Dr. Paul Varcholik for sharing his vast knowledge of multi-touch technology and Dr. Richie for pushing us to do more. Chris Sosa would like to acknowledge the support by WORKFORCE CENTRAL FLORIDA. Enrique Roche would like to acknowledge the support by WORKFORCE CENTRAL FLORIDA.

We would also like to give a special thanks to all of the family members and close friends of the members of Planck. Without their unwavering commitment and support, we could not have realized this project.

Finally, we offer our thanks and gratitude to the professors who have so kindly agreed to review our project we have worked so hard on.

REFERENCES

[1] Enrico Costanza and Jeffrey Huang, "Designable Visual Markers," *CHI 2009,* April 2009

[2] Martin Kaltenbrunner, Till Bovermann, Ross Bencina, and Enrico Costanza, "TUIO: A Protocol for Table-Top Tangible User Interfacesm," *6th International Gesture Workshop,* 2005

[3] NUI Group Authors, "Multi-Touch Technologies," Community Release, May 2009

[4] ReacTIVision 1.4. (n.d.). Retrieved May 1, 2012, from ReacTIVision 1.4 Documentation: http://reactivision.sourceforge.net/#usage

BIOGRAPHY

Peter Oppold is a 27 year-old graduating with his second degree in Computer Engineering from the University of Central Florida (UCF). Prior to Computer Engineering, he studied Criminal Justice and attended the Seminole Law Enforcement Academy. He is currently working as a Software Engineer intern at Lockheed Martin on the F-35 program, where he will continue his employment.

Héctor Enrique Rodríguez-Simmonds is a senior receiving his BSCpE with a minor in Computer Science in the Department of Electrical Engineering and Computer Science at the University of Central Florida. He currently works for Lockheed Martin Missiles and Fire Control as a CWEP Co/Op student in Systems Engineering on MEADS. He will begin pursuing a PhD in Computer Engineering at Virginia Polytechnic Institute and State University in the Fall 2012.

Christopher Anthony Sosa is a soon to be BSCpE graduate with a minor in mathematics from the University of Central Florida. He is an avid gamer, whose interests lie in computer graphics, artificial intelligence, game logic, human computer interfaces, simulation, serious games, and software development in general. He designed, built, and led a small team of students to create a computer system for a satellite on the UCF knightsat UNP team sponsored by Air Force Research Labs. He is currently seeking employment in a related field or hoping to attend graduate school for game design.

Enrique Roche is a senior receiving his Bachelors in Electrical Engineering from the University of Central Florida. His primary interests are in computer architecture, hardware/software codesign, augmented reality, and human-computer interaction. He hopes to pursue a career in a related field.