Grocery Shopping Assistant for the Blind (GroZi)

UCSD TIES—Spring 2009

Faculty Advisor:
Serge Belongie

Student Advisors:
Hourieh Fakourhar
Peter Faymonville
Tess Winlock

Community Client:
National Federation of the Blind (NFB)

Client Representative:
John Miller—NFB Representative

Team Members:
Jaina Chueh, Grace Sze-en Foo, Bonnie Han, Steven Matsusaka
Thomas Mueller, Jasmine Nourblin, Amalia Prada
Amir Shirkhani, Michael Tran, Jeffrey Wurzbach

Table of Contents

List of Tables and Figures

Introduction

There are currently 1.3 million legally blind people living in the United States who face daily obstacles with routine tasks, especially in regards to their experiences within supermarkets and stores. Developing assistive technologies and handheld devices allows for the possibility of increasing independence for blind and visually impaired. Currently, many grocery stores treat those that are blind as "high cost" customers, and dramatically undersell to this market, neglecting to take their needs into consideration. The use of computational vision can be advantageous in helping these blind customers, as restrictions such as the limited ability of guide dogs of white canes, frequently changing store layouts, and existing resources do not allow for a completely independent shopping experience. Using technologies such as object recognition, sign reading, and text-to-speech notification could allow for a greater autonomous solution to the relevant problem.

In conjunction with Calit2, UCSD's Computer Vision Lab, and TIES, the GroZi project is working to develop a portable handheld device that can help the blind to collect information and navigate more efficiently within difficult environments as well as better locate objects and locations of interest. GroZi's primary research is focused on the development of a navigational feedback device that combines a mobile visual object recognition system with haptic feedback. Although still in its early stages of development, when complete, the GroZi system will allow a shopper to navigate the supermarket, find a specific aisle, read aisle labels, and use the handheld grocery assistant device to then scan the aisle for objects that look like products on the shopper's list (compiled online and downloaded onto the handheld device prior to going into the store).

This quarter, under the supervision of our advisor, Serge Belongie, we pursue the computer vision aspects of the project that allows for autonomous detection and localization in the near future. In the past quarter, our team successfully customized the User Interface (UI) for new labeling tasks as well as improved the computer program that allows for inserting and storing data into the database as effortlessly as possible. However, there is still room for improvement. While this improvement awaits refinement, the focus this quarter has been shifted to the mechanical aspects of the project including the design of the GroZi parrot device, the mechanics and theory behind its functionality, and an experimental design phase that improves understanding of what changes and modifications are necessary for better efficiency of the device and its usability. Additionally, a special request by John Miller has the team involved in challenging themselves to build a recording program that will facilitate communication for the blind in industrial work involving visual graphics. The following document will serve as a description of what has been accomplished thus far, what has been learned and overcome, and the processes involved in designing and implementing a usable grocery assistant device for the blind to assist future members of TIES GroZi team.

Approach and Methodology

This quarter, two projects were attended to. The mechanical aspect of GroZi, which is the major project this quarter, has been designated the task of creating a potential device that fits the requirement of the grocery assistant device and its functionality in terms of working parts and usability. As a special side project this quarter, which will be further advanced in the future, the Mouse-Click Recorder Program group has been designated the task of building and compiling program that assists in improving the independence of the visually-impaired when dealing with digital images in the industrial workplace. The team breakdowns are as follows:

Mechanical/Haptics

Wiimote-Bluetooth: Grace Sze-en Foo, Steven Matsusaka
Mechanical Design: Thomas Mueller, Michael Tran, Jeffrey Wurzbach
Experimental Design: Jasmine Nourblin, Amalia Prada

Mouse-Click

Jaina Chueh, Bonnie Han, Amalia Prada, Amir Shirkhani

Wiimote-Bluetooth

Why the Wiimote?

The goal of the GroZi Parrot is to direct a blind user toward the desired grocery product through the strategic use of haptics. If the device is to communicate with the user through haptics, then it must first be able to visually detect and locate a grocery product as well as the user's hand relative to the grocery product. The machine must then process and use the information to send haptic feedback to the user.

The device may seem extremely complex and technologically advanced however there is technology readily available today which can perform the tasks needed for this project, such as Nintendo's Wiimote. The Wiimote is actually a very powerful device and is perfect for the GroZi Parrot application. When connected to the Nintendo Wii, the Wiimote uses its infrared sensor and accelerometers to translate position and motion to the television screen. In the parrot device, an infrared LED placed on the user's hand is tracked using the IR sensor and the accelerometers to recognize position and motion. It is also equipped with a vibrating motor and speakers. These are some of the most important functions of the Wiimote as the vibrations are used as the actual haptic feedback provided to the user, and the speakers are used to indicate successful location of the desired product. Furthermore, the Wiimote connects to the Nintendo Wii via Bluetooth communication. The GroZi team uses the Wiimote's Bluetooth capabilities to connect to a computer. With an established connection between Wiimote and computer, software can then program the Wiimote's functions to fit the parrot projects specific needs.

General Procedure

The objective of the Wiimote/Bluetooth subteam was to connect a Wiimote to a laptop computer and develop and implement software which uses the various functions of the Wiimote to specifically fit the parrot prototype requirements. The general procedures taken by the Wiimote team can be broken into two basic parts: establishing a Bluetooth connection and implementing software.

Establishing a Bluetooth Connection

The first step was to establish the Bluetooth connection. This task proved to be rather frustrating and involved tedious hours of troubleshooting. In fact, this proved to be the more difficult task of this quarter. The problems encountered stem from compatibility issues between the computer and the Wiimote. Although both devices are Bluetooth capable, the Wiimote was not made for use with a normal computer but with the Wii. Furthermore, using the Windows Vista OS may have complicated the process even more.

The computer needs a Bluetooth device installed in order to be Bluetooth capable. Some computers come with an internal device, while others need an external adapter. One of the problems faced this quarter was finding a Bluetooth device which worked with the Wiimote. After doing research the team was able to find a list of compatible and incompatible devices. The entire list is included in the appendix. The list also includes working Bluetooth driver stacks to be used with the specific device. Specific driver stacks also needed to be used in order to successfully establish connection. This quarter the particular combination of device and driver stack used was the Rocketfish 2.0 Bluetooth Dongle and the Widcomm Bluetooth Software v. 5.1.0.1100.

Even with a listed compatible device and driver stack, the team still encountered connection problems. The Microsoft Bluetooth stacks installed on the Microsoft OS caused problems and interfered with the additional driver stacks which were needed. The solution to this problem was to rip, or remove and disable the Microsoft stacks from the system. After this was done the other Bluetooth driver stack was able to act independently without interference. Ripping the Microsoft Bluetooth stacks was a long process in itself. Instructions to perform this task are provided in the appendix. This guide is almost exclusively copied from http://www.dev-toast.com/2007/01/05/uncrippling-bluetooth-in-vista-rtm/ and edited according our own experience.

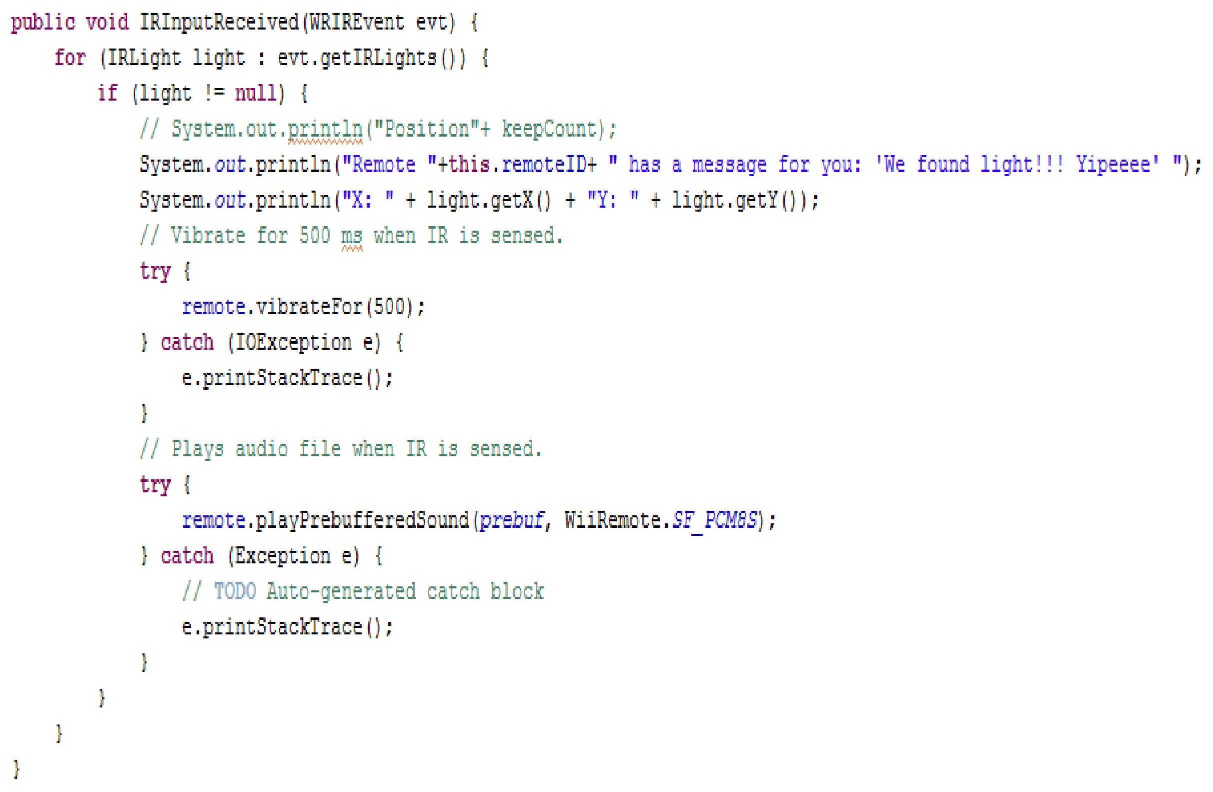
Software Development and Implementation

Once all of the compatibility issues were solved, software could be developed and implemented in order to suit the project requirements. The software was developed in the Java Runtime Environment using the Java Development Program Eclipse. The team used the Java library BlueCove in order to interface with the Bluetooth device. In addition, the Java library WiiRemoteJ was used. WiiRemoteJ is a free Java API and library for interacting with a Nintendo(R) Wii Remote (tm) and Balance Board (tm) through Bluetooth(R). WiiRemoteJ aims to do two things: first, it aims to create an easily accessible interface for Java developers wanting to work with the Wii Remote or Balance Board. Second, it aims to provide tools for developers using these devices to speed development. WiiRemoteJ comes with a complete javadoc, detailing every public class, field, and method. In addition, WiiRemoteJ aims to duplicate many of the same ideologies and methods used in standard Java libraries. This allows developers to jump into the development process relatively quickly.

Java Code

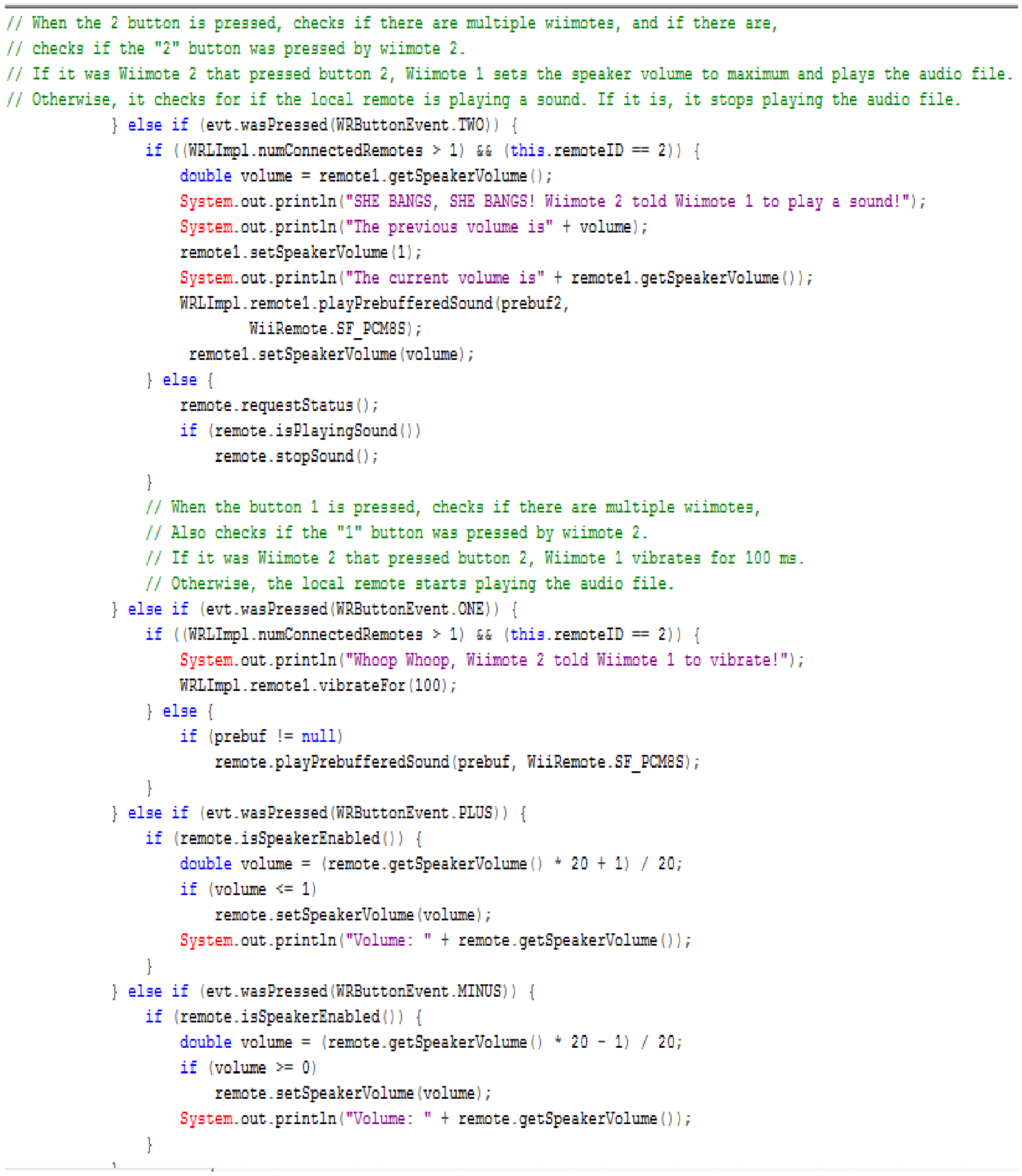
The original code that was used was the sample code by Michael Diamond which was included in the installation package downloaded from WiiRemoteJ. The code was then modified to fulfill the purposes of GroZi. Several functions programmed in the code are shown below.

- When the wiimote senses infrared (IR), it will output the coordinates of the IR source, vibrate and play the specified audio file.



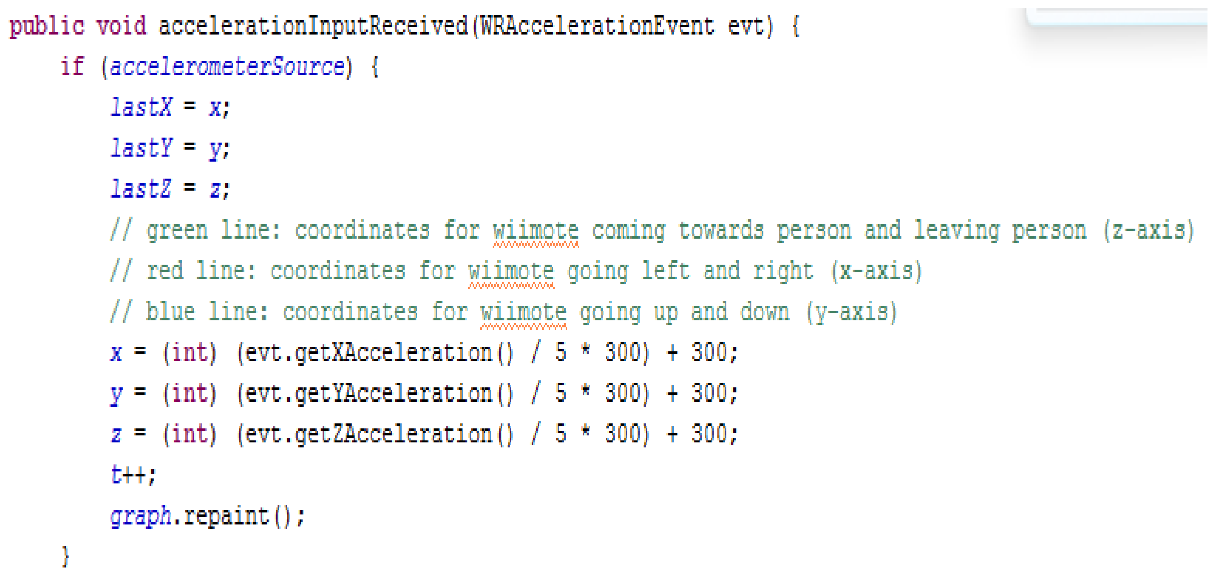
```java
public void IRInputReceived(WRIREvent evt) {
    for (IRLight light : evt.getIRLights()) {
        if (light != null) {
            // System.out.println("Position"+ keepCount);
            System.out.println("Remote "+this.remoteID+ " has a message for you: 'We found light!!! Yipeeee' ");
            System.out.println("X: " + light.getX() + "Y: " + light.getY());
            // Vibrate for 500 ms when IR is sensed.
            try {
                remote.vibrateFor(500);
            } catch (IOException e) {
                e.printStackTrace();
            }
            // Plays audio file when IR is sensed.
            try {
                remote.playPrebufferedSound(prebuf, WiiRemote.SF_PCM8S);
            } catch (Exception e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }
    }
}
```

- When there are at least 2 wiimotes connected, wiimote 2 is used to control wiimote 1. Several functions include:
  - Press button “1” on wiimote 2 to make wiimote 1 rumble for the specified time.
  - Press button “2” on wiimote 2 to make wiimote 1 play an audio file (music).
  - Press button “+” on wiimote 2 to make wiimote 1 play an audio file (hotter—product of interest is getting closer)
  - Press button “-“ on wiimote 2 to make wiimote 1 play an audio file (colder—product of interest is getting farther)



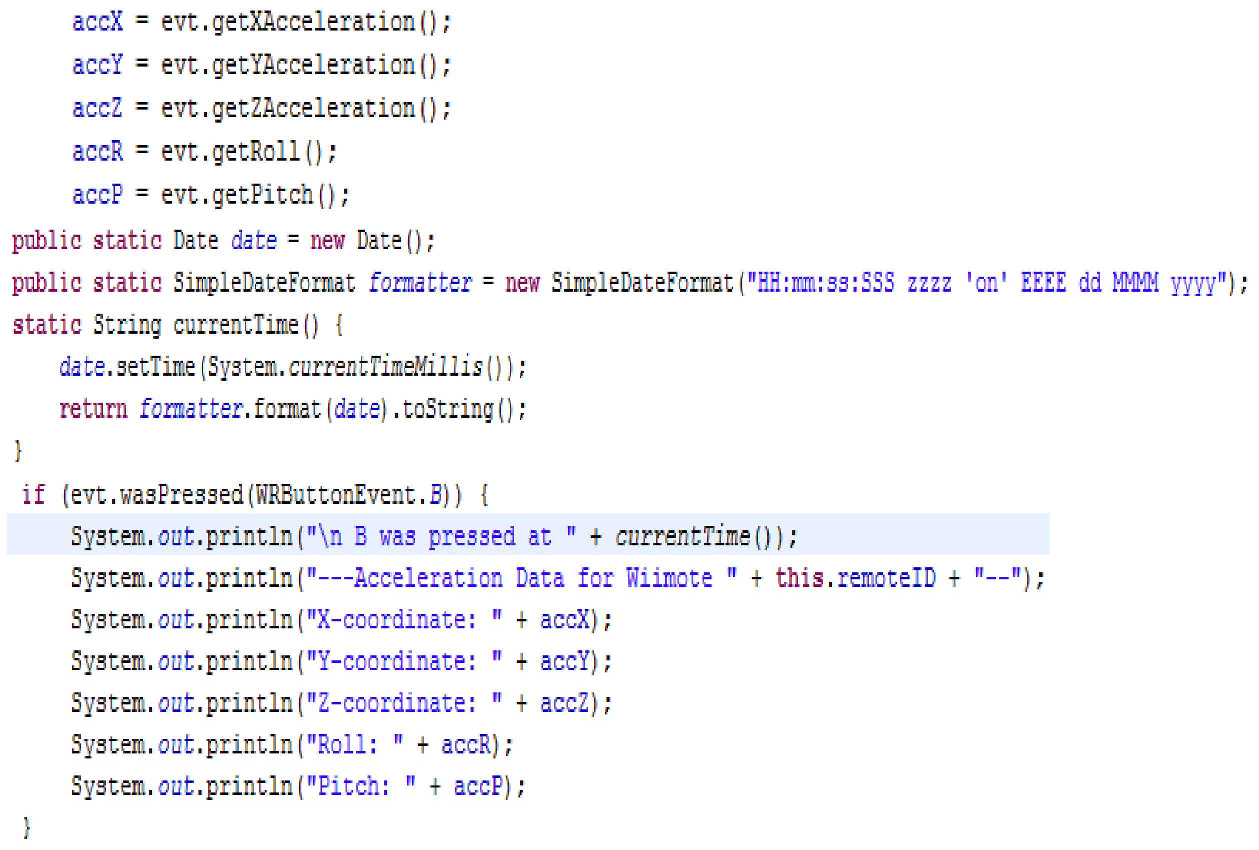
```
// When the 2 button is pressed, checks if there are multiple wiimotes, and if there are,
// checks if the "2" button was pressed by wiimote 2.
// If it was Wiimote 2 that pressed button 2, Wiimote 1 sets the speaker volume to maximum and plays the audio file.
// Otherwise, it checks for if the local remote is playing a sound. If it is, it stops playing the audio file.
            } else if (evt.wasPressed(WRButtonEvent.TWO)) {
                if ((WRLImpl.numConnectedRemotes > 1) && (this.remoteID == 2)) {
                    double volume = remote1.getSpeakerVolume();
                    System.out.println("SHE BANGS, SHE BANGS! Wiimote 2 told Wiimote 1 to play a sound!");
                    System.out.println("The previous volume is" + volume);
                    remote1.setSpeakerVolume(1);
                    System.out.println("The current volume is" + remote1.getSpeakerVolume());
                    WRLImpl.remote1.playPrebufferedSound(prebuf2,
                            WiiRemote.SF_PCM8S);
                     remote1.setSpeakerVolume(volume);
                } else {
                    remote.requestStatus();
                    if (remote.isPlayingSound())
                        remote.stopSound();
                }
                // When the button 1 is pressed, checks if there are multiple wiimotes,
                // Also checks if the "1" button was pressed by wiimote 2.
                // If it was Wiimote 2 that pressed button 2, Wiimote 1 vibrates for 100 ms.
                // Otherwise, the local remote starts playing the audio file.
            } else if (evt.wasPressed(WRButtonEvent.ONE)) {
                if ((WRLImpl.numConnectedRemotes > 1) && (this.remoteID == 2)) {
                    System.out.println("Whoop Whoop, Wiimote 2 told Wiimote 1 to vibrate!");
                    WRLImpl.remote1.vibrateFor(100);
                } else {
                    if (prebuf != null)
                        remote.playPrebufferedSound(prebuf, WiiRemote.SF_PCM8S);
                }
            } else if (evt.wasPressed(WRButtonEvent.PLUS)) {
                if (remote.isSpeakerEnabled()) {
                    double volume = (remote.getSpeakerVolume() * 20 + 1) / 20;
                    if (volume <= 1)
                        remote.setSpeakerVolume(volume);
                    System.out.println("Volume: " + remote.getSpeakerVolume());
                }
            } else if (evt.wasPressed(WRButtonEvent.MINUS)) {
                if (remote.isSpeakerEnabled()) {
                    double volume = (remote.getSpeakerVolume() * 20 - 1) / 20;
                    if (volume >= 0)
                        remote.setSpeakerVolume(volume);
                    System.out.println("Volume: " + remote.getSpeakerVolume());
                }
```

- Display the accelerograph of the wiimotes



```
public void accelerationInputReceived(WRAccelerationEvent evt) {
    if (accelerometerSource) {
        lastX = x;
        lastY = y;
        lastZ = z;
        // green line: coordinates for wiimote coming towards person and leaving person (z-axis)
        // red line: coordinates for wiimote going left and right (x-axis)
        // blue line: coordinates for wiimote going up and down (y-axis)
        x = (int) (evt.getXAcceleration() / 5 * 300) + 300;
        y = (int) (evt.getYAcceleration() / 5 * 300) + 300;
        z = (int) (evt.getZAcceleration() / 5 * 300) + 300;
        t++;
        graph.repaint();
    }
```

- Output the x, y, z-positions, roll and yaw of the wiimote when the button B is pressed. Also prints out when the button B was pressed.



```
    accX = evt.getXAcceleration();
    accY = evt.getYAcceleration();
    accZ = evt.getZAcceleration();
    accR = evt.getRoll();
    accP = evt.getPitch();
public static Date date = new Date();
public static SimpleDateFormat formatter = new SimpleDateFormat("HH:mm:ss:SSS zzzz 'on' EEEE dd MMMM yyyy");
static String currentTime() {
    date.setTime(System.currentTimeMillis());
    return formatter.format(date).toString();
}
 if (evt.wasPressed(WRButtonEvent.B)) {
     System.out.println("\n B was pressed at " + currentTime());
     System.out.println("---Acceleration Data for Wiimote " + this.remoteID + "--");
     System.out.println("X-coordinate: " + accX);
     System.out.println("Y-coordinate: " + accY);
     System.out.println("Z-coordinate: " + accZ);
     System.out.println("Roll: " + accR);
     System.out.println("Pitch: " + accP);
 }
```

- Disconnects all wiimotes when one wiimote is disconnected. This solves the problem of having to restart the computer every time the program is run.

```
// If one wiimote disconnects, all other wiimotes will disconnect too.
public void disconnected() {
    System.out.println("Remote disconnected... Please Wii again.");
    try {
        if (remote5 != null) {
            remote5.disconnect();
            System.out.println(" \n Remote 5 disconnected!!!");
            remote5 = null;
        }
    } catch (Exception e) {
    }
    try {
        if (remote4 != null) {
            remote4.disconnect();
            System.out.println(" \n Remote 4 disconnected!!!");
            remote4 = null;
        }
    } catch (Exception e) {
    }
    try {
        if (remote3 != null) {
            remote3.disconnect();
            System.out.println(" \n Remote 3 disconnected!!!");
            remote3 = null;
        }
    } catch (Exception e) {
    }
    try {
        if (remote2 != null) {
            remote2.disconnect();
            System.out.println(" \n Remote 2 disconnected!!!");
            remote2 = null;
        }
    } catch (Exception e) {
    }
    try {
        if (remote1 != null) {
            remote1.disconnect();
            System.out.println(" \n Remote 1 disconnected!!!");
            remote1 = null;
        }
    } catch (Exception e) {
    }
}
```

Geiger counter vs. 5 Wiimote Model

The wiimote subteam was able to successfully establish connection and implement simple software needed for initial testing of the entire parrot prototype. The team was able to establish connections and implement software to multiple Wiimotes with one laptop at a single time. The software was written so that the wiimote vibrates and plays a sound when an IR source is sensed. This would correspond to vibrating when sensing the IR LED located on the user's hand. In later prototypes the vibrations would be programmed in more strategically to fit the haptic need of the project.

The initial idea of the haptic feedback was to place four Wiimotes around the user's upper body: two on the chest and two on the back (on the left and right sides of the body). Thus by vibrating a specific Wiimote, the vibration could tell the user to turn left, right, front, back, up, or down. A fifth wiimote mounts on top of the shoulder and would be used as the IR sensor and audio speaker.

Due to initial concerns with establishing the Wiimote connection, the GroZi team decided to look into a different approach and to use only one Wiimote. The idea was to vibrate one Wiimote at appropriate frequencies corresponding to the distance from the product. For example, the vibrations would occur at low frequency when far away and would increase as the user got closer. This approach is the method of the Geiger counter found in metal detectors. Rough estimates of frequencies were tabulated for initial testing as shown below.

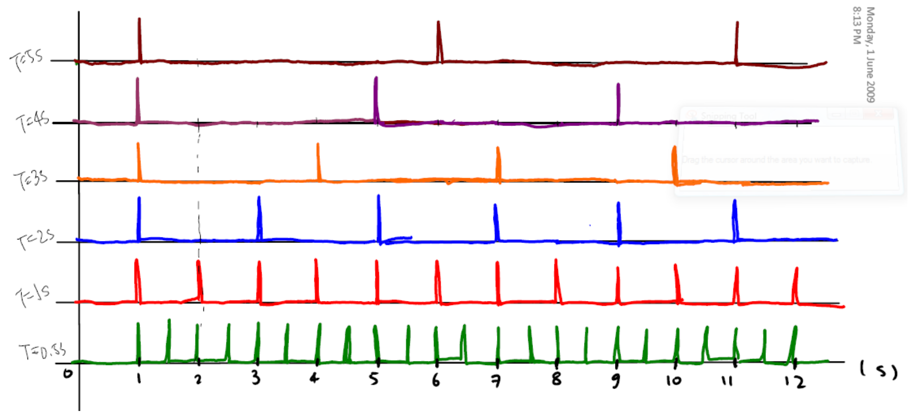


Figure 1: The figure above illustrates the different frequencies graphically. Below is the rough frequency distribution ranges decided upon for initial testing.

- When the device is within 25 feet (7.62meters) of the object, the period, T = 5 seconds and the frequency, F=0.2Hz.
- When the device is within 20 feet (6.10meters) of the object, T=4 seconds and F=0.25Hz.
- When the device is within 10 feet (3.05meters) of the object, T=3 seconds and F=0.33Hz.
- When the device is within 5 feet (1.52meters) of the object, T=2 seconds and F=0.5Hz.
- When the device is within 2.5 feet (0.762meters) of the object, T=1 second and F=1Hz.
- When the device is within 1 feet (0.3meters) of the object desired, T= 0.5s and F=2Hz.
- Once the hand is above the product, a sound file will be played.

Radius, m	Frequency, Hz
7.62	0.2
6.10	0.25
3.05	0.33
1.52	0.5
0.76	1
0.30	2

Table 1: The table above describes the relationship between the radius and frequency of vibration.

Initial Testing and Results

Initial tests were performed on the Geiger model of the parrot prototype. Experiments using the Soylent poster shelves built by the Experimental Design Subsection were performed with John equipped with a Wiimote programmed to vibrate and play an audio sound by commanding it manually from another Wiimote. As John approached the products, the frequency of vibration increased and when the product was found an audio sound would indicate success.

An important result found during the experiments was the need for a larger distribution range of frequencies within a certain radius of the product. The rough estimates listed above proved to be inefficient because once John's hand would enter a certain radius of the product, the frequencies would stop changing and John had to feel around for the product without any significant assistance. Thus within a certain radius the frequencies should still vary quite greatly to ensure haptic feedback all the way until successful location of the product. Complete procedures and results are described in detail in the Experimental Design Subsection.

What's Next?

In the future, the program should be modified and improved in order to further serve the project needs. The Geiger model was chosen for this quarter however the subteam has already demonstrated the ability to connect to multiple Wiimotes (the maximum actually being seven). Thus both the Geiger and multiple Wiimote models should be improved and implemented in the code. Furthermore, the haptics of the project should be rethought for both models. For the Geiger model new frequency ranges should be decided upon. For the multiple Wiimote model, the overall design of the haptics should be outlined (which Wiimotes should vibrate and when). Furthermore, the future team should attempt to integrate the different components of the parrot prototype – the Wiimotes, web camera, LED glove, and mounting backpack. Once these tasks are performed, further testing can be performed using a more realistic parrot prototype.

Mechanical Design

Introduction

The goal of the mechanical and hardware design subsection of the Haptics section is to explain what advances in the hardware design were made during the Spring 2009 quarter.

Mechanical Design

This quarter, we designed a Wiimote carrier, camera carrier and a mount for the carriers. The carrier parts screw together and then the Wiimote carrier has screw holes that the mount connects via. The mount is designed to hold the assembly on the shoulder strap and form a stable platform fort the machine vision system. This assembly has not been built yet. We also made a glove with IR LEDs and a battery attached to it for testing tracking of the user's hand. A revision of the IR LED board has already been designed, but not built.

The required functions and features for the Shoulder Mount assembly are:

1. Stable platform for the camera
2. Suitable holder for the Wiimote
3. Keep the Wiimote and camera in the same visual plane (assuming the camera is not in a pan/tilt mode)
4. Comfortable for the user
5. Minimal weight

The mount is designed to attach to the user's right shoulder. This position is assumed to give an optimum field of view if the user puts the glove on their right hand. By keeping the Wiimote and the camera in the same plane, a simple linear map can be done in software. This will allow a much simpler set of algorithms to be used for guiding the user's hand; however this is assuming that the camera is not panning and tilting.

The mount assembly was designed in Solidworks. A model of the shoulder mount was made and tested for fit. It seems to fit adult male users by reversing it, while it fits adult female users in its current configuration. Figure 2 shows a rendering of the Shoulder Mount Assembly. We have not completed any fit testing on children.

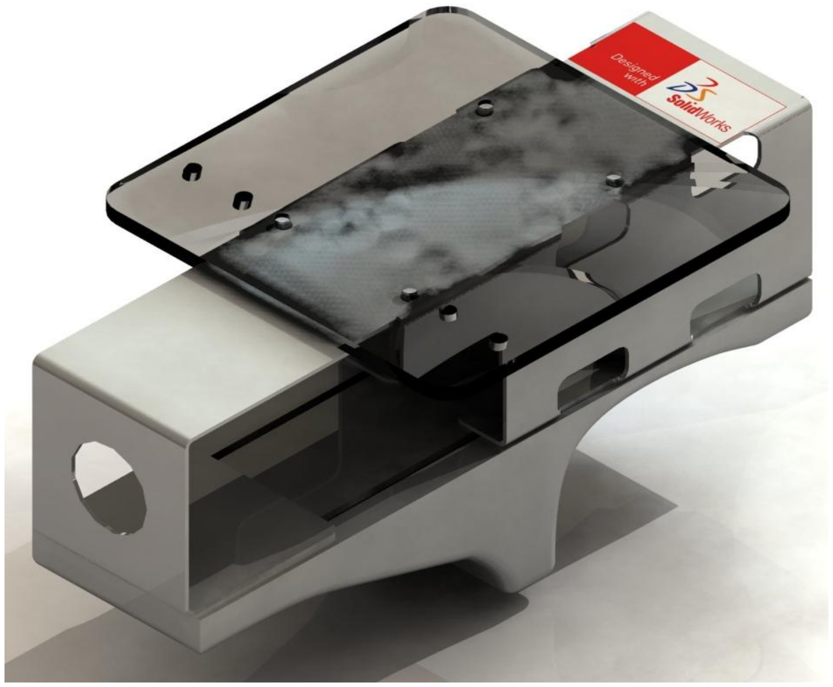


Figure 2: A computer generated rendering of the Shoulder Mount Assembly

The clear plate on the top of the assembly will be made from acrylic. Acrylic can be given smooth edges that are required for a human-wearable device to prevent any lacerations to the user's face, if they were to turn their head suddenly into the mount assembly. The partly enclosed box in the middle of the assembly is the Wiimote carrier. It will be made of sheet aluminum to minimize weight, which is estimated at about one ounce. The bottom part of the assembly is the shoulder mount. Its material has yet to be determined, although for prototyping purposes, it will be made from a product called Machinable wax. This will allow us recycle old prototypes and the chips from the machining process into making another prototype, by simply melting them down and recasting a billet. Additionally the Machinable wax is easy on tooling and self-lubricating. Machinable wax will not be suitable material for production of the actual product, but it will be a convenient material for making prototypes. For a production model, something like injection modeled plastic would be used.

A Wiimote needs at least two 940nm IR point sources to track location. Thus to track the location of the user's hand, a pair of IR point sources needs to follow the user's hand. We decided the simplest implementation is to take two IR LEDs and solder them to a perf board. Then attach the pref board to a glove. Since gloves are readily available, only the board design remained. The following feature list was composed after a first pass at building such a glove (Block Zero) was completed.

The required features for the IR LED board are:

1. Stable mounting for two IR LEDs
2. Switch with off position marked in a way that can be felt
3. LEDs emit 940nm IR light

The Block Zero build of the IR LED board is very simple. It was made from a few parts found in the obligatory parts drawer. It lacks a formal switch and battery holder. Presently, there is a screw and some nuts that act as the switch and batteries are soldered together and then zip tied to the pref board. The standing design is useful for nothing but a first draft and crude testing, due to its lack of replaceable batteries and formal switch. Also, it appears based on cell phone camera tests the IR LEDs have a very narrow field of view, maybe 30° maximum. This means that the Wiimote will have to be in nearly direct line of sight with the tops of the IR LEDs. As built, this not possible, without bending the user's hand at a strange angle. Based on John's feedback about Block Zero, we know that the design needs to have a switch with some level of tactile feedback, in addition to a touch based label for the on or off position. To address this in Block Zero B, the PCB will have a cut out by the switch's off position. The switch is specified as a slide switch. A picture of the Block Zero B design is shown in Figure 3.

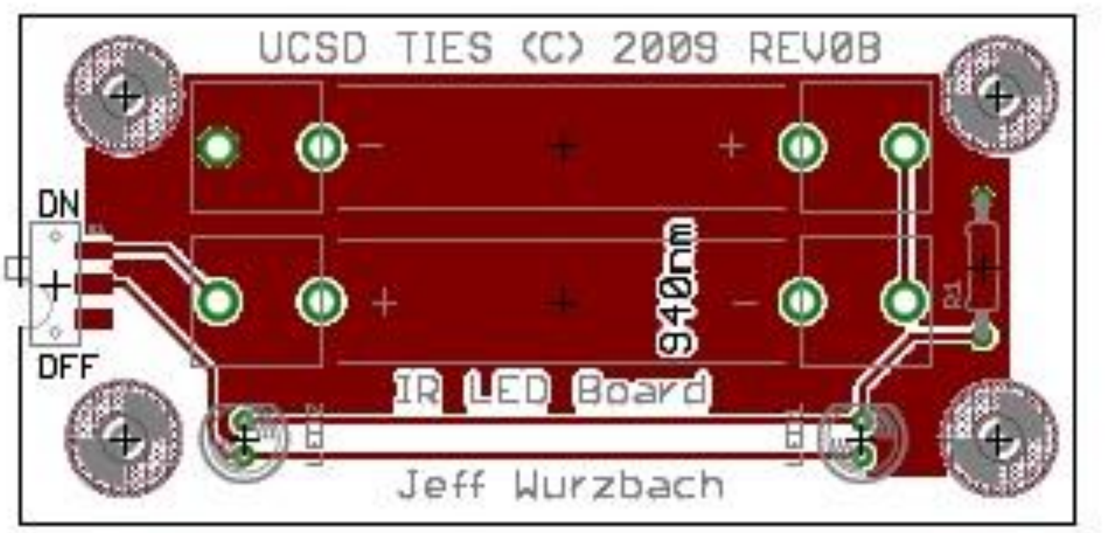


Figure 3: Purposed Block Zero B PCB layout of the IR LED board

As Figure 3 shows, the new design has 2 AA battery holders, a switch and two 5mm LEDs. The cut out by the switch is cleary visible on the middle left of the PCB. The design is mostly through-hole parts, with the only SMD being the switch. The PCB measures 3.10"x1.50", for a total of 4.65 square inches. The design passes DRC tests for Advanced Circuits and Sparkfun Electronics. The theory of operation is very simple and proven in Block Zero. Current flows from the batteries to the switch, which if in the on position, allows current to flow into the LED array. After the LED array, the current goes through a current limiting resistor and back the to the batteries. If the switch is in the off position, then no current flows due to an open circuit condition. The batteries are wired in series to give 3Vdc. The two LEDs are wired in parallel. The Block Zero configuration uses the same basic schematic. Most of the power is used in the LEDs. No power consumption measurements have been taken, due to the early prototype nature of the design. Future versions of the LED board may be reduced to a wristband or LED ring.

What's next?

The next steps are to fabricate the designed parts and then test them in the context of the complete Haptics and vision system. In terms of the IR LED board, several variants of angular LED position, and substitution of LEDs to units with different fields of view should be tested to find optimal tracking accuracy. For example, the Block Zero IR LED board has the plane of the LEDs parallel to the PCB. Other angles like 30°, 45° and 90° with respect to the plane of the PCB need to be built and tested. The battery life of the LED board also needs to be measured.

For the Shoulder Mount assembly, the actual parts need to be machined. This entails getting access and time on machines by either getting people who are authorized to use the MAE design labs, the CNC machine in CAL-(IT)^2, the water jet at Scripps Institute of Oceanography, or going to an off campus shop and paying them to make the parts.

Experimental Design

Introduction

The goal of the Experimental Design subsection, as part of the Haptics section, is to create tests and methods that will help us better understand our computer vision system and its limits. Until the prototype is ready for testing, the methods outlined below should be used to gauge the efficacy of the system design itself. This quarter, we have begun designing these experiments and some "rough" testing has been done but a completed prototype has yet to be formally tested.

Methods

What are we trying to test? Once the prototype is fully assembled and working, we want to know what its capabilities are- can it decipher between the correct product and a similar but incorrect one? Specifically, we want to see if our computer vision system can detect the difference between:

- Tide with Bleach and regular Tide
- Tide in color and Tide in monochrome
- Different sizes of Tide (small, medium, large)
- Multiple brand names on one product (ex. Tide with Febreeze)
- Different Tide "scents"-ex. Original vs. Ocean Breeze

Additionally:

- How far away must one be from the product for it to be recognizable?
- What are the maximum angles in 3-D in which products are within the system's view?
- What is the time to acquire a product?

Tide is just one of many products we want our system to be able to detect, but we have selected it as a good testing candidate because it comes in over 19 different types (there were 8 different small boxes, 8 different medium boxes, and 3 different big boxes of Tide at an Albertson's in Southern California), lending much versatility to our analysis. The first five questions test our system's capabilities, while the last three test the "usability" of our system. These questions are just as important, if not more, than the first five questions.

More pertinent data collected from Albertson's:

- Each aisle is about 6' from floor to top of shelf.
- The Tide aisle has 6 shelves and ~13.7 inches between shelves.
- The shampoo aisle has ~11 inches between shelves.
- Toothpaste and hair dye aisles have ~8 inches between shelves.
- Soup aisle has ~11.5 inches between shelves. Only Campbell's stores soup on its side in dispensers. All other brands stack vertically.
- To grab an object naturally, the body is about 17 inches away from the shelf.

- Some Tide competitors include: Gain, Sunburst, Sun Classic, Cheer, Dreft, Arm & Hammer, All, and generic store brands.

To answer the aforementioned questions, three reusable poster boards were created this quarter. All three are white foam boards that have actual 3-D products from Ralph's and the Sunshine Store velcroed on the front. Velcro was used because it allows the items on each board to be easily switched out, moved around, or replaced, and it allows the blind user to actually acquire the item by removing it from the "shelf." The first board is entitled "Soylent Shelf" and it consists of four products: a box of Cheezits, a Snickers bar, a Campbell's Chicken Noodle soup can, and a Kit Kat bar.

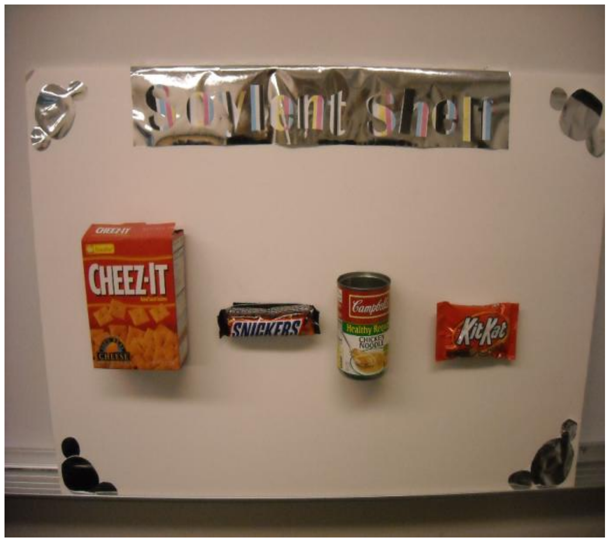


Figure 4: Soylent Shelf

The second, "Soylent Laundry," improves upon the first board because it utilizes anatomical shelf spacing as was measured at Albertson's (13.7 inches between shelves). It is comprised of some 3D products and some product images that were printed out: Tide "Ocean Breeze," regular Tide, regular Tide with Bleach in a box (3D), Windex(3D), Bounce(3D), Tide with Febreeze, and regular Tide with bleach in a bottle (3D).

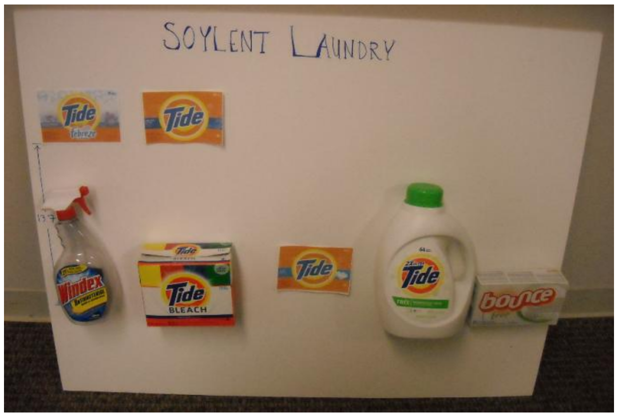


Figure 5: Soylent Laundry

The third and final board made this quarter, "Soylent Soup," improves again upon the second board. It also utilizes anatomical shelf spacing (10" between shelves), and actual Cup Noodle soup placement was observed and recorded in both the Sunshine Store and Ralph's.

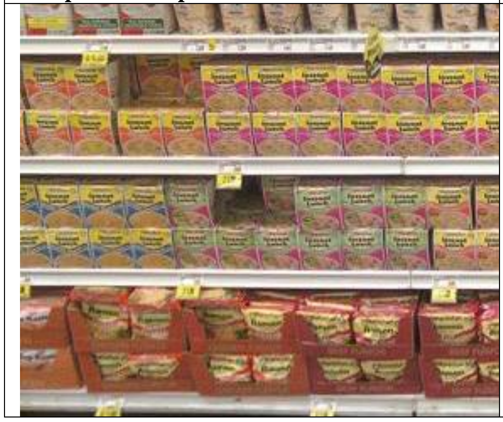
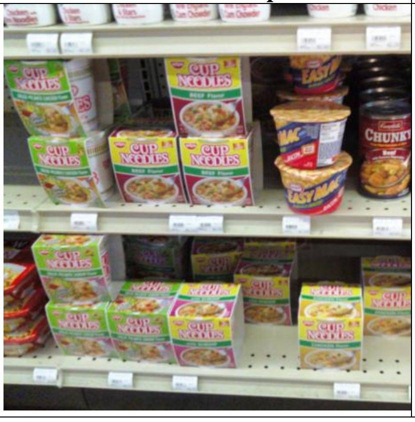
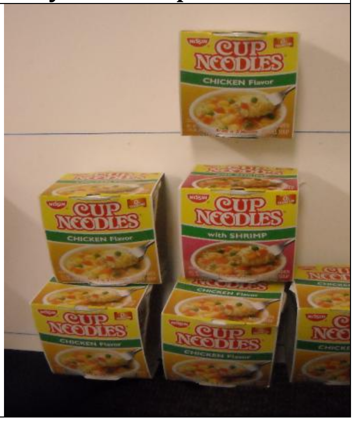
Ralph's Soup Aisle	Sunshine Store Soup Aisle	"Soylent Soup" Board
		

Figure 6: Simulation comparison between real and model soup aisles

It was noted that soups are often stacked directly on top of each other, and flavors are often misplaced and mixed with other flavors on the shelves. We chose to model our board after the Sunshine Store. We created three clusters of soup, on two shelves, that range in detection difficulty. The easiest detection puzzle is on the far right. A pink box of Shrimp flavor is mixed with four yellow Chicken flavored soups. Neither the color scheme nor the flavor names are similar. The middle cluster is slightly more difficult, as there are red Beef soups mixed in with red Chicken Pecante soups. The leftmost cluster is the most difficult detection puzzle: pink Shrimp flavored soups are mixed in with pink Shrimp Pecante soups. Here both the color scheme and the flavor names are similar. We hope this improvement creates a life-like testing environment. Because the "Soylent Soup" board is most like a real grocery store, it should be used in testing.

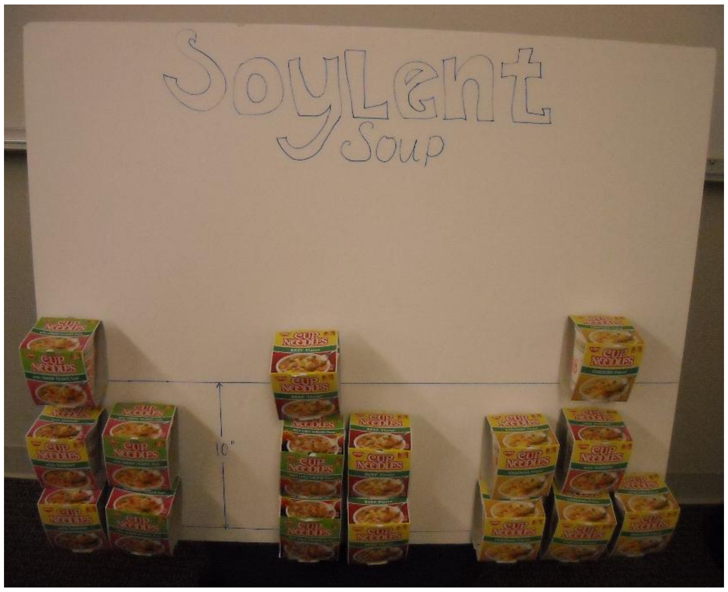


Figure 7: Soylent Soup

During Week 10 of Spring quarter, a rough experiment was performed using the Soylent Soup board (pictured above). The experiment is outlined below:

Soylent Soup Geiger Experiment

- John is already on the soup "aisle"
- Probability of selection is 1/16.
- Distance from board to label on stick: 55 inches to grip. Grip is 12.5 inches. (John begins ~55" from the board).
- Big board dimensions: 40" by 16" is search area, 40" x 32" is total board dimensions.
- Board is 36" off the ground
- Soup cup dimensions: 3.75" x 4.5"

Tapping frequency:
Slow: 11 taps/10 seconds ~ 1.1Hz
Fast: 20 taps/10 seconds ~ 2Hz
Amalia is proctor and time keeper for first 2 trials. For trials 3-7 Tess is the proctor and Jeff is the time keeper.

Trial 1: (with left and right instruction from Grace & Wiimote)

Shrimp soup
Time: 23 seconds

Trial 2: Geiger
Beef soup
Time: 18 seconds

Trial 3: Geiger
Chicken Soup
Time: 13.57 seconds

Trial 4: Geiger
Spicy Chicken soup
Time: 18.84 seconds

Trial 5: Geiger
Shrimp soup
Time: 23 seconds

Trial 6: Geiger + LRUD Instructions every second
Beef soup
Time: 9.63 seconds

Trial 7: Geiger + LRUD Instructions every second
Shrimp soup
Time: 9.78

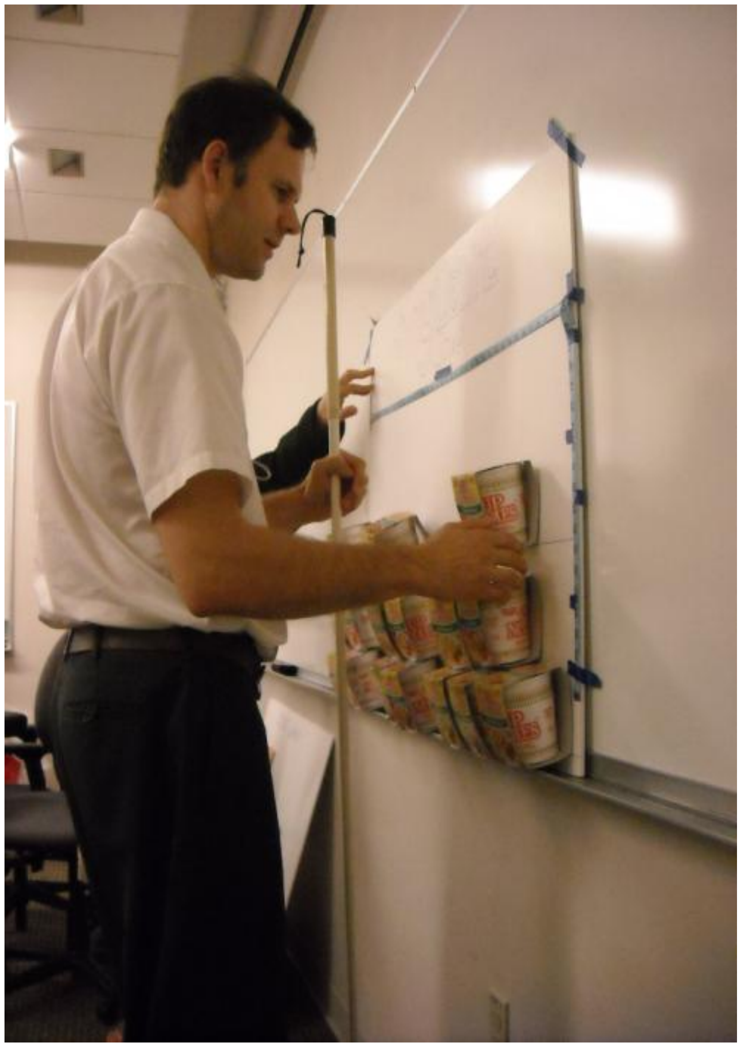


Figure 8: Soylent Soup testing with John

Trials 6&7 average: 9.705 seconds
Trials 2-5 average: 18.35 seconds

These averages *suggest* that the Left/Right/Up/Down instructions are beneficial to the user. However, this experiment was *very* rough and approximate, and more formal trials should be run. Only once was John was timed walking the 55 inch distance to the board. This time was 2.73 seconds. Theoretically, this time should be measured during each trial, and separated from the search time. John's distance from the board should be measured exactly. Many more trials should be run (about 30) to increase statistical power and decrease variability in measurements.

Experimental Design Costs

Product	Place of Purchase	Quantity	Unit Price
Large Poster Board	UCSD Bookstore	2	$5.45
Small Poster Board	UCSD Bookstore	1	$3.45
Velcro Pack of 4 sets	UCSD Bookstore	4	$4.49
Cup Noodle Soup	Vons	16	$0.60
Tide with Bleach	Earl's Place	1	$7.00
		Total	$48.91

Table 2: Experimental Design Cost

The remaining food items not included in this list were "donated" products from team members, who had already purchased them for personal use. This total cost is relatively low.

Until the prototype is ready, experiments can be run using a human experimenter who will control the "Geiger" wiimote, placed in the breast pocket of the blind subject. Using the "Soylent Soup" board, an experiment to collect selection times for a sighted individual (using sight) and a blind individual (using the human-aided Geiger system until the actual prototype is working) should be performed as such:

Procedures

1) The blind person should begin some distance "X" away from the poster board, perhaps not aligned at the horizontal center of the board. The board should be covered, hiding the placement of products.* The blind user has a wiimote in his/her pocket that will vibrate at specified frequencies, like a Geiger counter.

2) A third party identifies one particular soup for selection and silently points to it, so the user can't hear where it is on the board. The soups will work well for this experiment, because though there are different flavors on the board, they all feel the same from a tactile standpoint. The time measured is between when the third party says "start" and when the correct item has been pulled off of the board.

3) A fourth party trails the blind user and simulates the computer vision system, which works like a Geiger Counter. As the user gets closer to the object, the fourth party makes the pocketed wiimote vibrate more frequently. Once the user places their hand over the correct object, the fourth party says “ding.” When this object has been pulled off of the board, the fourth party says “correct” or “incorrect.”

4) The total number of trials, total number of successes, time per trial, and subject's name for both the sighted subject trials and the blind subject trials should be recorded.
Repeat the experiment (steps 1,2, and 4) for a sighted user. The experiment should be repeated 30 times for each user.
* this step is only necessary for trials involving a sighted user.

What’s Next?

The test outlined above should be performed formally. Once the prototype is ready, the boards should be used to carry out the tests described in the Methods section, using the prototype. The procedures described above for the Soylent Soup experiment should be repeated with the prototype instead of the fourth-party computer vision simulator when the prototype is working. Additionally, some of the products currently on the boards have not yet been entered in the GroZi 120 database, and should be before tests are carried out. These products are: Spicy Chicken Soup, Beef Soup, Shrimp Soup, Shrimp Pecante Soup, Tide with Bleach, Tide with Febreeze, Tide Ocean Breeze, and Bounce.

After the boards have been used for the above tests, the next step would be to carry out these tests on the prototype in the Sunshine store or a grocery store. Perhaps most importantly, acquisition time data should be recorded for an experiment in which a blind user tries to select the same object (experimental) as a sighted person (control). This experiment should be repeated many times, so a lot of data can be collected. Their times to acquire should be recorded and compared using a statistical test, such as a t-test or a one-way ANOVA. This statistical evidence will help us understand how effective our system is in comparison to the biological human vision system. Additionally, a metronome can be used to synchronize the frequency of vibration for different distances from the object which, if implemented, would ideally reduce the time required to reach the product of interest.

Mouse-Click Recorder Program

What is the Mouse Click Recorder Program?

The Mouse Click Recorder is a new software program that increases the independence of the visually impaired by allowing them to review a digital image. The MCR program can provide image annotation in a format that is easy for a blind person to use and is designed for a sighted assistant to run while its output is accessible for the blind. When using the mouse click recorder program, a sighted person can highlight a point of interest that a blind person can reference.

Imagine that a sighted person points to several locations on an image displayed on the screen of a computer. The person makes different comments while pointing at each location. The mouse click recorder program allows a sighted person to record coordinates clicked by a mouse on an image and save these coordinates into a generated output file that includes the description of the image, the location (x and y coordinates) of the mouse click, and a note describing the location.

General Overview of Usage

The Mouse Click Recorder program is stored in a .jar file that runs similarly to an .exe (executable) file. It can be downloaded from the UCSD Grozi website. The user must have the most recent Java Runtime Environment installed on their computer. (JRE is a free application that contains the minimum requirements to run a .jar file.) Once this is done, the user can simply double click on the .jar file for the program to start running.

The MCR is a simple program that writes to file the location of any mouse click on an image. It provides the user an opportunity to leave a note next to any mouse click location. Suppose the input file is "input_file.jpg." After clicking on the image with MCR, the generated output file would appear in the same folder named "input_file.txt." This output file would contain information of the mouse click pixel coordinates and label associated with that coordinate. The MCR program is able to open any .jpg, .bmp, .gif, or .png image as long as it is stored within a folder on his or her computer.

Example of Applicability of Program

The Mouse Click Recorder program has applicability to blind professionals in a number of jobs. Given an image of any real-world object, a form of measurement, and several clicked coordinates of the image from a sighted assistant, a blind person can calculate and gain a better perception of the dimensions of real world objects. For example, a blind biology student can use photography as a recording tool to track a plant's growth over time.

The student is provided a jpg image of a plant like the one shown in Figure 9, and asked to determine the height of the plant. The jpg image also contains a picture of a ruler next to the plant. The student can calculate the height of the plant, after determining the pixel locations of the bottom and top of the ruler as well as the bottom and top of the plant with the help of his sighted assistant (to click the designated locations). The blind biology student may now review an ASCII file to get the recorded pixel coordinates from the image along with related notes

Here is an illustrative example:

Given an image **plant.jpg,** let the plant be P pixels vertically and let the tape measure be T pixels vertically.

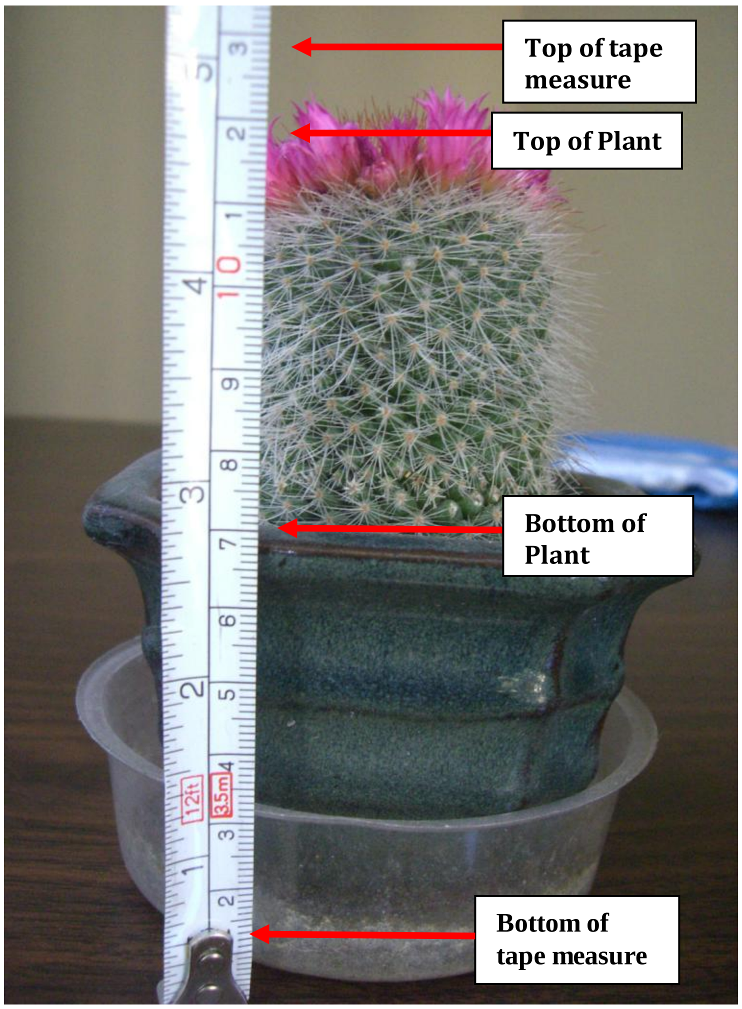


Figure 9: Plant Measurement

After clicking on the image and marking the top and bottom of the plant, and the top and bottom of the ruler, the output file **plant.txt**, should contain the annotated coordinates:

Row | Column:	**Comments:**
127|451	Top of Plant
137|59	Bottom of Plant
132|255	Top of ruler
136|5	Bottom of ruler

Therefore, to compute the height:

P = top of plant – bottom of plant
= 451 - 59 = 392

T = top of ruler – bottom of ruler
= 255 – 5 = 250

Suppose the tape measure is in meters, H = 1.0 meter. Then to compute the total height of the plant:

Total = H * P / T
= 1.0 * 392 / 250 = 1.57 m

Mouse Click Recorder Installation Procedures

Downloading Java Runtime Environment

1. Go to http://java.com/en/download/index.jsp
2. Click on **free Java download** to download the most recent JRE on your computer.

Downloading MCR

1. Go to http://grozi.calit2.net/
2. Under the '**downloads**' section, click on the '**Mouse Click Recorder Software**' link.
3. A pop up window will appear, click '**save**' to save the file. The Mouse Click Recorder program should now be on your computer in zip format.
4. Extract the **MouseClicker.zip** file.
   To extract a zip file to a specific location in Windows, right click on the file, click **Extract file**, and select the path of the location you want to extract the file to.
5. Double click on the **MouseClickRecorder.jar** file to run the program.

What is included with MCR installation package

The user will find the following files in the mouse click recorder folder:

a. The **MouseClickRecorder_v#.jar** file that runs the MCR program.
b. **Lib** folder: Contains code that allows mouse clicker to run on your computer.
c. **SampleImage** folder: contains the following images:

a.	Brain images	**brain1.jpg**, **brain2.jpg**, **brain3.jpg**, **brain4.jpg**
b.	Plant image	**plant.jpg,**
c.	Circuit board image	**circuitboard.jpg**
d.	Text file	**brain1.txt,** **plant.txt**

The text file is a sample text file that contains labels with coordinates that represent locations on the image **brain1.jpg** and **plant.jpg** that have been clicked with a mouse beforehand.

d. Documentation of how to run the program.

Running the Mouse Click Recorder

User Interface

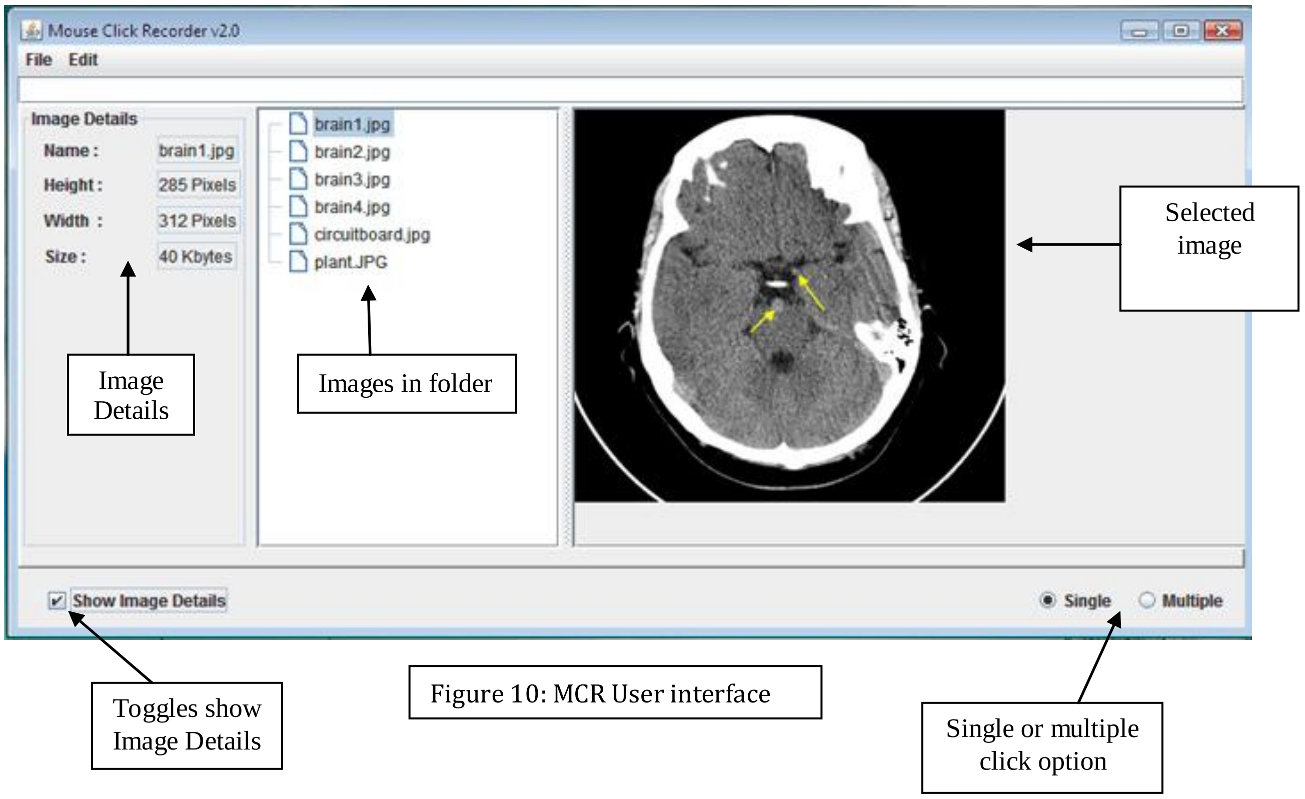


Figure 10: MCR User interface

- **Image details:** Display the description of the image – filename, pixel width and pixel height, and the size of the file. Image details can be toggled on and off to hide/show the panel.
- **Images in folder:** All images in the opened directory will be displayed in the file listings. MCR can only read .jpg, .gif, .bmp, .png images.
- **Single button:** Allows user to enter a label in the textbox, press and click on a point in the picture. MCR will store the coordinate under that label.
- **Multiple click button:** Allows user to enter a label in the textbox and click on multiple points in the picture. MCR will store all of these coordinates under the same label.
- **Selected Image:** The selected image will appear on the right hand panel once clicked. To activate MCR's

Generated Output File Format

The MCR program will generate an output file with the same name as the image that was clicked. i.e. Given a file named **filename.jpg**, the MCR program will create an output file in the same directory named **filename.txt**.

The output file will contain the following information:

- Header with image details – pixel height and width
- Pixel row
- Pixel column
- Label associated with clicked coordinates

i.e. Given an image 8.5 inches wide by 11 inches tall, 4 possible corners could be:

- upper left corner (0,0)
- upper right corner (0, 849)
- lower left corner (1049,0)
- lower right corner (1049, 849)

Each line of the generated txt file would contain coordinates of the mouse clicks and a label/note.

```
i.e.  output file:  "200 | 300   picture tag"
                    " ...  |  ...   center of brain area"
                    " ...  |  ...   dark region in brain"
                       ...     ...
```

Mouse Click Recorder Tutorial

Step-by-step instructions on how to get started with the Mouse Click Recorder Program.

1. **Start the program.**
   Double click on the .jar file to start up the MCR program.

2. **Select the folder that contains the images.**
   The MCR program will immediately bring you to a file open window like the one featured below. MCR can only open folders, hence if there are any image files that you wish to open, move them into a folder. Then select the folder that contains the image you wish to click on.

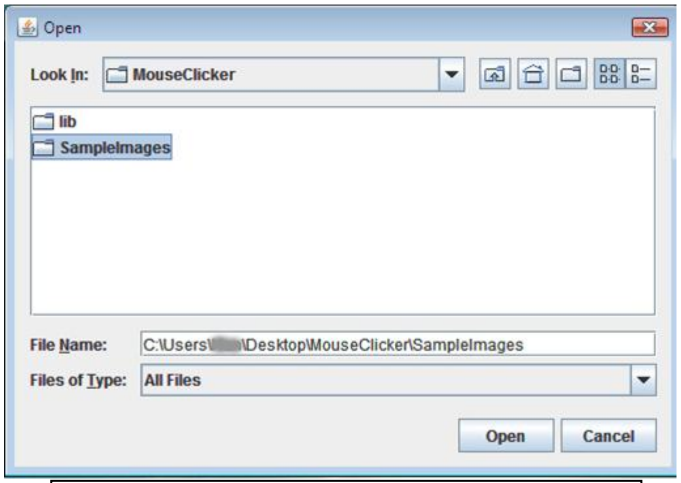


Figure 11: MCR Tutorial 1—File open window

3. **Mouse Clicker Interface.**
   Once you have selected the folder with the image(s) inside, the following window should appear with a list of all the image files in that folder directory. Notice the check box and the 'Single' and 'Multiple' option at the bottom of the window.
   Show Image Details: display details about the selected image. The default is checked.
   Single: Stores one coordinate to one label. This is the default option.
   Multiple: Stores multiple coordinates to one label.

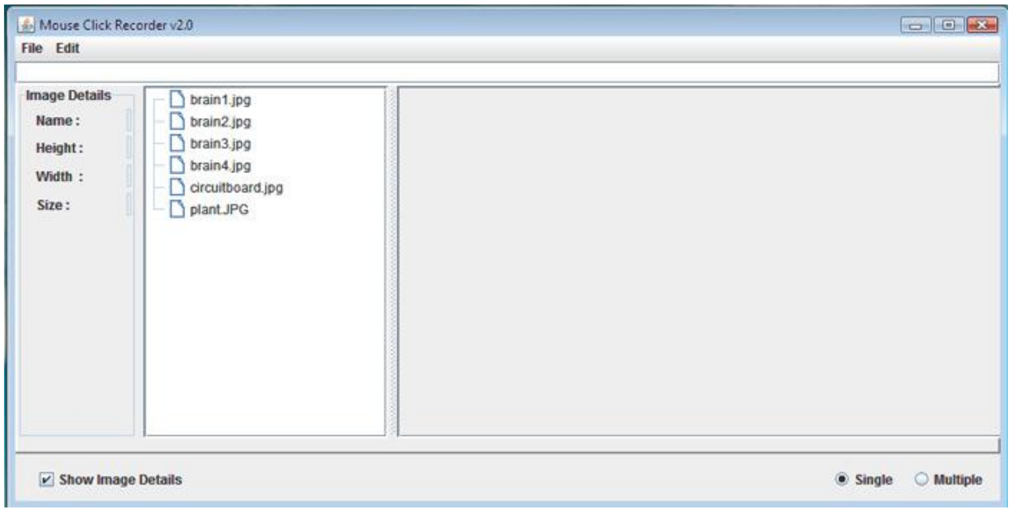


Figure 12: MCR Tutorial 2—Mouse Clicker interface

4. **Select an image.**
   Select a picture. Doing so would cause the image to show up in the right hand side of the MouseClickWindow. Let's select **brain1.jpg** as shown below:

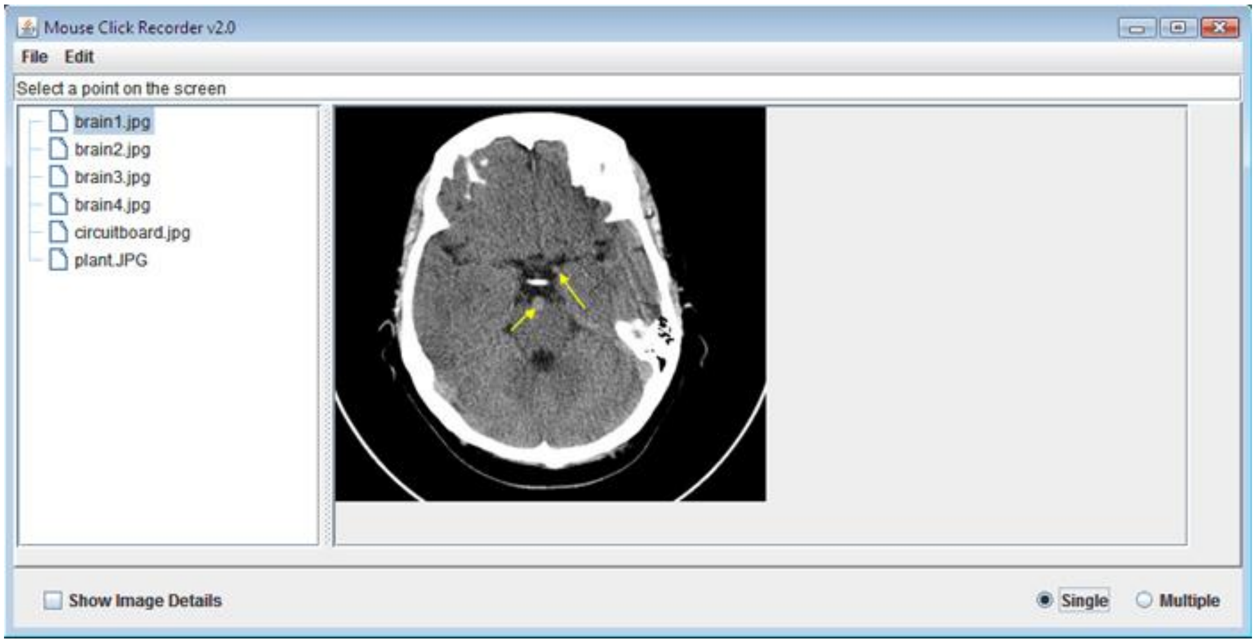


Figure 13: MCR Tutorial 3—Selecting an image

5. **Record a point.**
   In order for MCR to save the coordinates, **enter a label** in the text-box then **press enter** to save the label. Finally, **click** on any region within the picture. The MCR will save the clicked coordinate in a txt file.
   Note: In order for MCR to save the coordinates with the label, always remember to enter the label, press ENTER, then click.

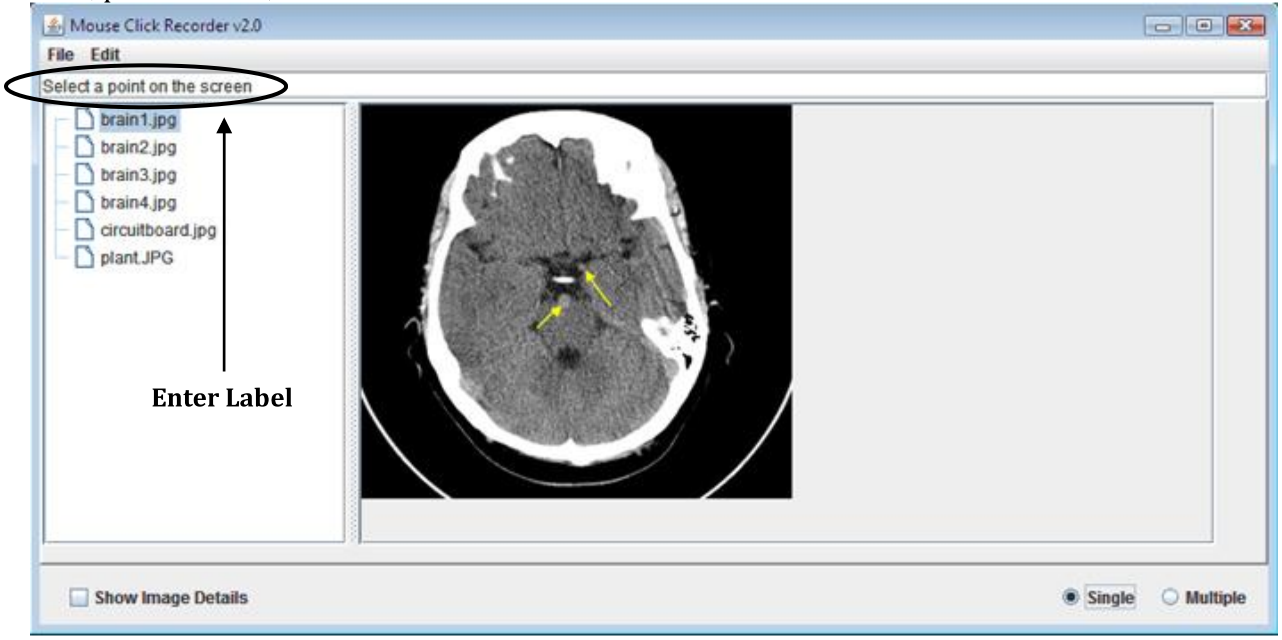


Figure 14: MCR Tutorial 4—Recording a point

6. **Single click vs. Multiple click.**
   a. **Record a point with a label.**
   Single click: Click on the **Single** option. Enter a label in the textbox, press enter to save the label, and click on a point in the picture.
   b. **Record multiple points with same label.**
   Multiple click: Click on the **Multiple** option. Enter a label in the textbox, press enter to save the label, click on multiple points in the picture. MCR will store all of these coordinates under the same label you have entered. The coordinate recording function will be disabled when another label is entered or the **Single** option is selected.

7. **Generated output file.**
   Open up the folder directory on your computer that contains the image you previously clicked on. A corresponding text file with the same name as your image should appear in the same directory.
   i.e. If the file is named 'brain1.jpg', an output file named 'brain1.txt' will be generated.

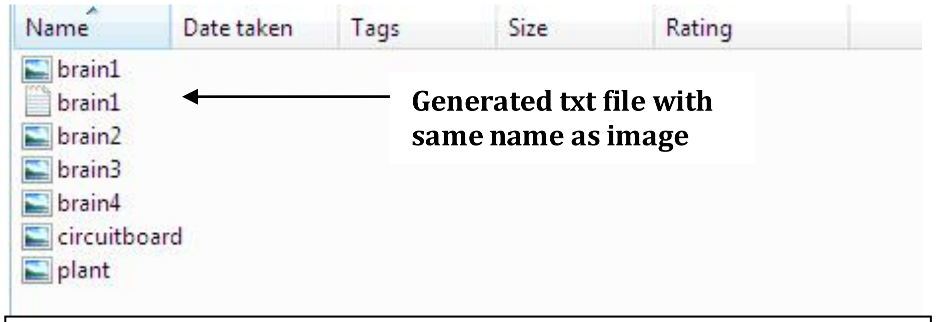
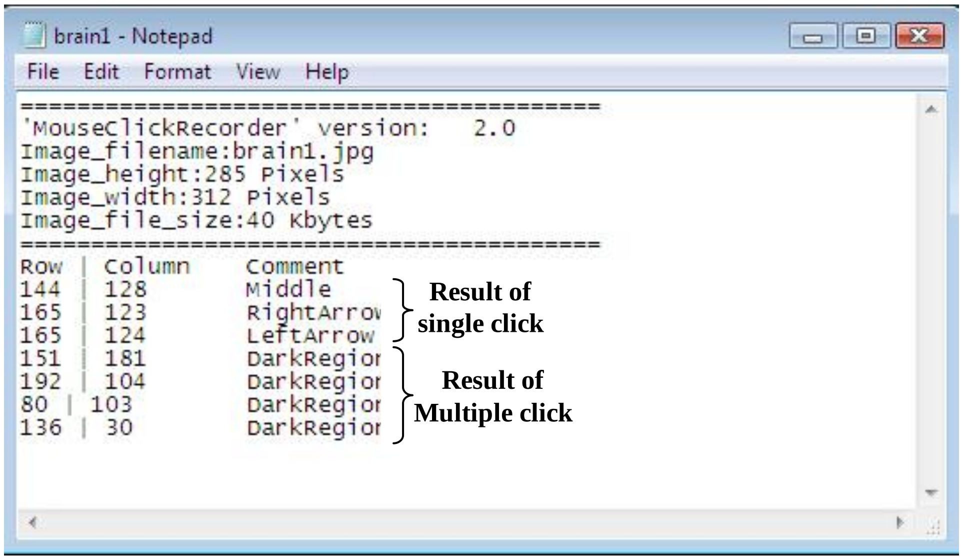


Figure 15: MCR Tutorial 5—The output icons (corresponding .jpg and .txt)

8. **Output file.**
   Open the output file to see the list of coordinates and the labels associated with the clicked coordinates.



```
========================================
'MouseClickRecorder' version:   2.0
Image_filename:brain1.jpg
Image_height:285 Pixels
Image_width:312 Pixels
Image_file_size:40 Kbytes
========================================
Row | Column     Comment
144 | 128        Middle
165 | 123        RightArrov
165 | 124        LeftArrow
151 | 181        DarkRegior
192 | 104        DarkRegior
80  | 103        DarkRegior
136 | 30         DarkRegior
```

Figure 16: MCR Tutorial 6—Details of the .txt output

Mouse Click Recorder Code

* The Mouse Click Recorder is written in JAVA and uses the Swing library.

<u>How an image is fetched and displayed in the Panel (FROM START TO FINISH):</u>

Call Hierarchy to get an image:

MouseClickWindow() constructor → initComponents()

initComponents()→ Calls getCenterPanel()

getCenterPanel() →Calls getJSplitPane0()

getJSplitPane0()→ Calls getJScrollPane0() and getJScrollPane1()

getJScrollPane0()→ Calls getJTree0()

getJScrollPane1()→Calls getImageLabel()

getJTree0()→ invokes a SelectionListener addTreeSelectionListener()

addTreeSelectionListener()→ Calls imageLabelMouseMouseReleased()

imageLabelMouseMouseReleased()

Obtains the path of the file selected on the Tree. Creates an **ImageIcon** "i" and points the path to "i".

Obtains the height and width of the image:

i.getImage().getHeight(null);

i.getImage().getWidth(null);

Stores the height and width for parallel use:

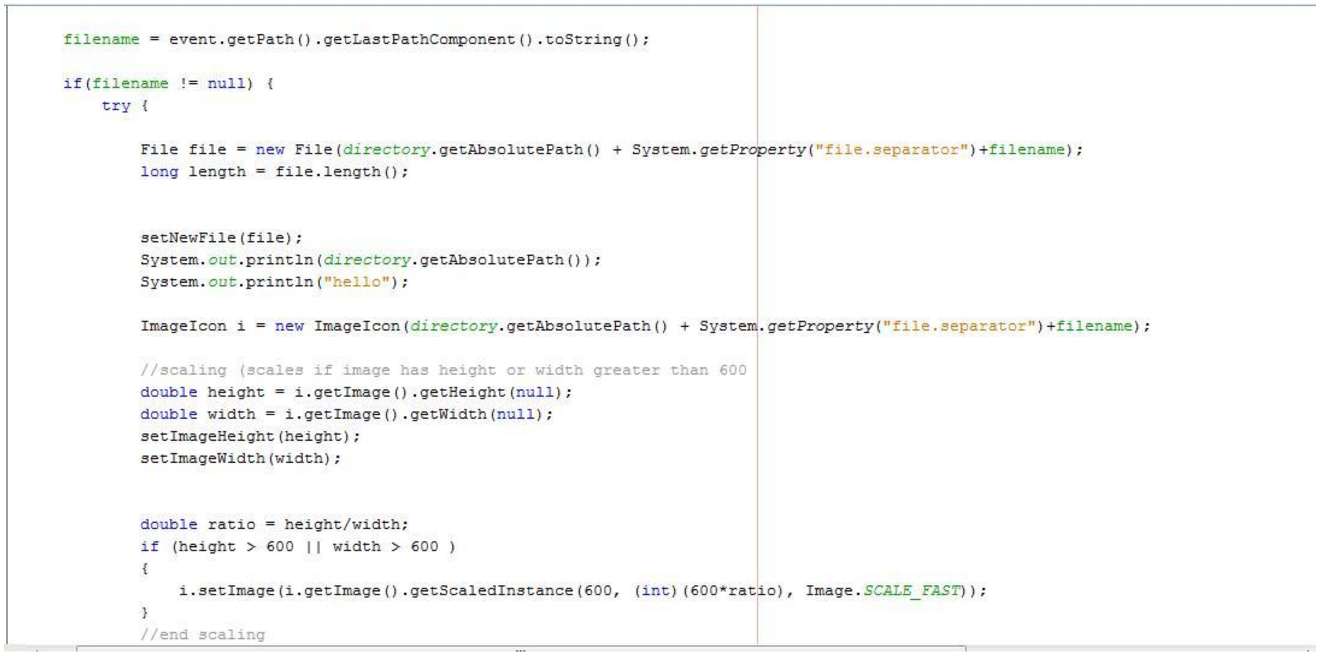
setImageHeight(height);

setImageWidth(width);

Sets the ImageIcon "i" to JLabel "ImageLabel"

imageLabel.setIcon(i);

imageLabel.repaint();



```
    filename = event.getPath().getLastPathComponent().toString();

    if(filename != null) {
        try {

            File file = new File(directory.getAbsolutePath() + System.getProperty("file.separator")+filename);
            long length = file.length();


            setNewFile(file);
            System.out.println(directory.getAbsolutePath());
            System.out.println("hello");

            ImageIcon i = new ImageIcon(directory.getAbsolutePath() + System.getProperty("file.separator")+filename);

            //scaling (scales if image has height or width greater than 600
            double height = i.getImage().getHeight(null);
            double width = i.getImage().getWidth(null);
            setImageHeight(height);
            setImageWidth(width);


            double ratio = height/width;
            if (height > 600 || width > 600 )
            {
                i.setImage(i.getImage().getScaledInstance(600, (int)(600*ratio), Image.SCALE_FAST));
            }
            //end scaling
```

How a Mouse Click is fetched and passed on to be saved :

Call Hierarchy to get a Mouse Click:

MouseClickWindow() constructor → initComponents()

initComponents()→ Calls getCenterPanel()

getCenterPanel() →Calls getJSplitPane0()

getJSplitPane0()→ Calls getJScrollPane0() and getJScrollPane1()

getJScrollPane0()→ Calls getJTree0()

getJScrollPane1()→Calls getImageLabel()

getImageLabel()→ invokes MouseListener mouseReleased()

mouseReleased() ()→ Calls imageLabelMouseMouseReleased ()

imageLabelMouseMouseReleased ():

Looks and the selection Mode:

If “multiple” is selected:

It stores the passed in point that was selected

It Calls saveAnnotation() on the point and the label in the Commentfield.

if(this.selectMode == 1)

mouseCoords = event.getPoint();

saveAnnotation(directory, filename, mouseCoords.x, mouseCoords.y,

getCommentField().getText());

if "single" is selected:

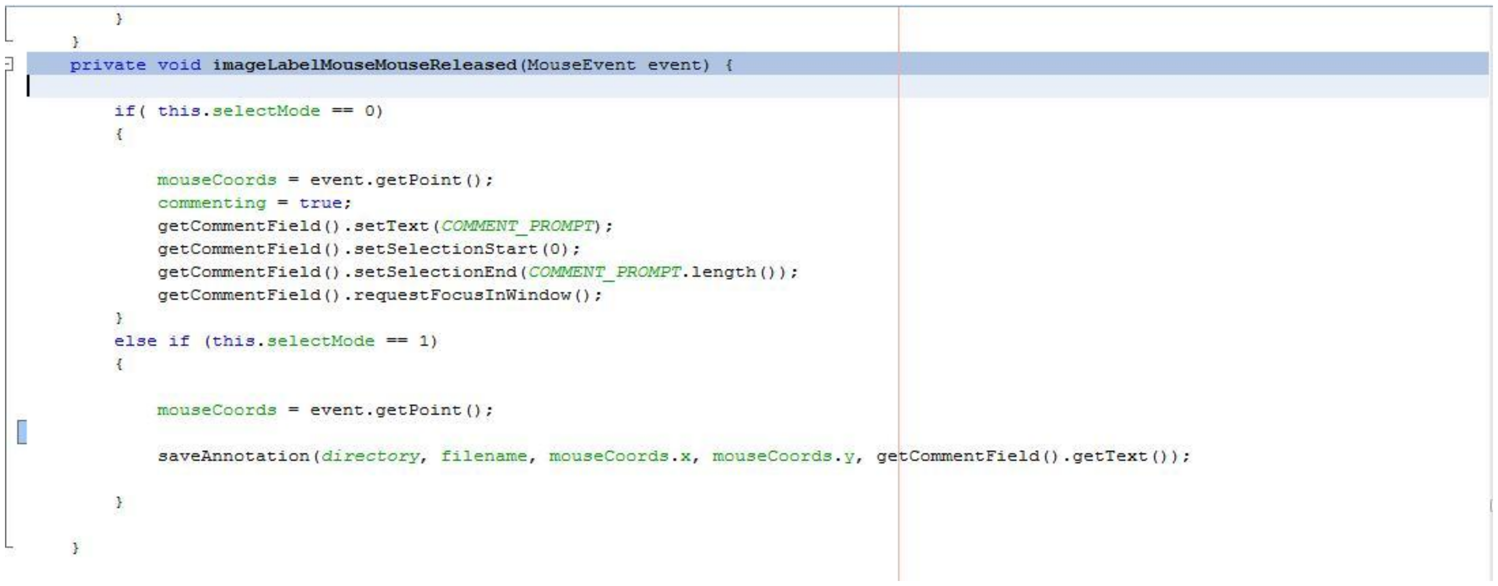
It stores the passed in point that was selected

Prompts the user to input something for the label.

if(this.selectMode == 0)

mouseCoords = event.getPoint();;

getCommentField().setText(COMMENT_PROMPT);



```
        }
    }
    private void imageLabelMouseMouseReleased(MouseEvent event) {

        if( this.selectMode == 0)
        {

            mouseCoords = event.getPoint();
            commenting = true;
            getCommentField().setText(COMMENT_PROMPT);
            getCommentField().setSelectionStart(0);
            getCommentField().setSelectionEnd(COMMENT_PROMPT.length());
            getCommentField().requestFocusInWindow();
        }
        else if (this.selectMode == 1)
        {

            mouseCoords = event.getPoint();

            saveAnnotation(directory, filename, mouseCoords.x, mouseCoords.y, getCommentField().getText());

        }

    }
```

<u>**How a label is fetched and passed on to be saved in text :**</u>

Call Hierarchy to get a Mouse Click:

MouseClickWindow() constructor → initComponents()

initComponents()→ Calls getCommentField()

getCommentField() → invokes actionPerformed ()

actionPerformed ():

Looks and the selection Mode:

If “multiple” is selected:

It highlights the text.

(label assumed to be inserted when Image was clicked)

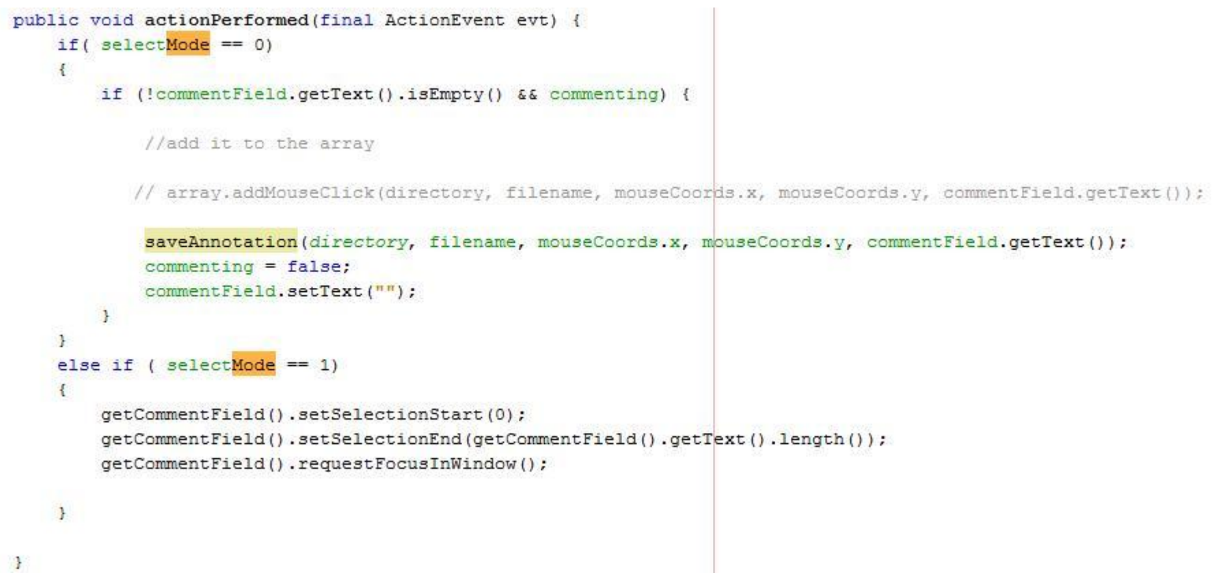
```
if ( selectMode == 1)

        setSelectionStart(0);

        setSelectionEnd(getCommentField().getText().length());

        requestFocusInWindow();
```

if “single” is selected:

It calls saveAnnotation() on the mouse click and the comment.

It sets the comment to an empty string.

```
if ( selectMode == 0)

        saveAnnotation(directory, filename, mouseCoords.x, mouseCoords.y, commentField.getText());
        commentField.setText("");
```



```
public void actionPerformed(final ActionEvent evt) {
    if( selectMode == 0)
    {
        if (!commentField.getText().isEmpty() && commenting) {

            //add it to the array

           // array.addMouseClick(directory, filename, mouseCoords.x, mouseCoords.y, commentField.getText());

            saveAnnotation(directory, filename, mouseCoords.x, mouseCoords.y, commentField.getText());
            commenting = false;
            commentField.setText("");
        }
    }
    else if ( selectMode == 1)
    {
        getCommentField().setSelectionStart(0);
        getCommentField().setSelectionEnd(getCommentField().getText().length());
        getCommentField().requestFocusInWindow();

    }

}
```

<u>How a label is passed in to a text :</u>

saveAnnotation() takes in (File directory, String fileName, int x, int y, String label, string) as its parameters.

It creates a file name with .txt as the extension.

int extPos = fileName.lastIndexOf(".");

String annotationFileName = fileName.substring(0,extPos+1) + "txt";

File annotationsFile = new

File(directory.getAbsolutePath()+System.getProperty("file.separator")+annotationFileName);

It checks if the file is new:

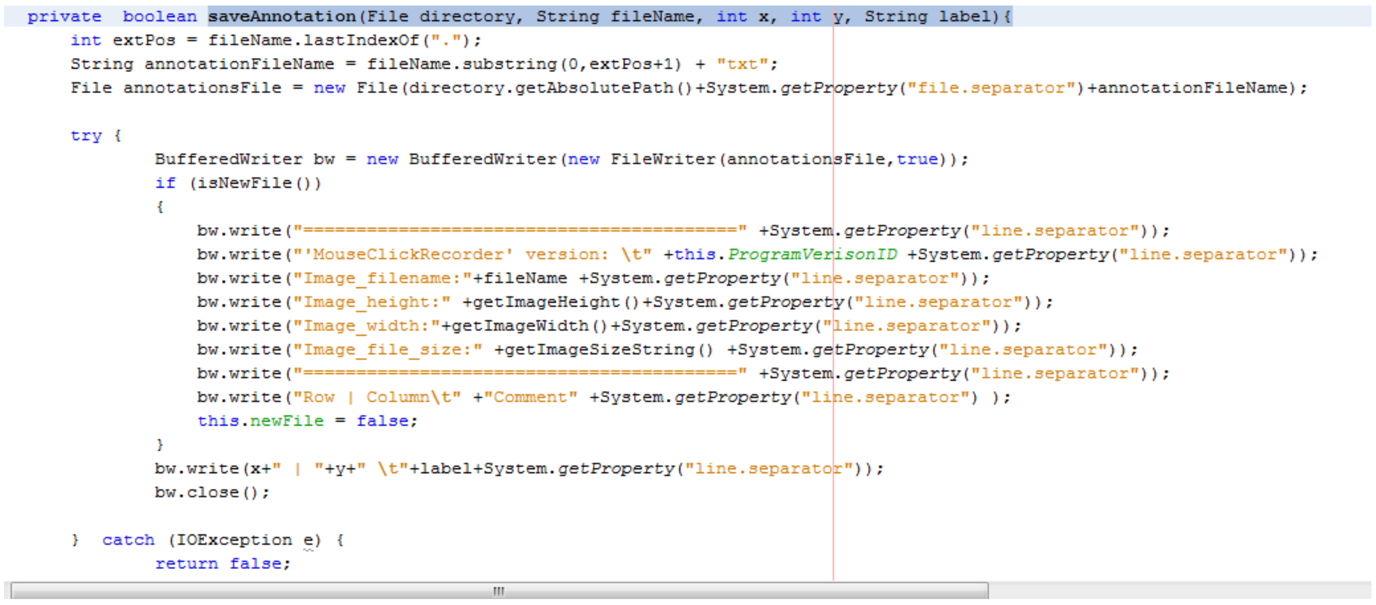
> If it's a new file writes the Title buffer to the txt file.
>
> If it's an old file it continues with the process and writes the values of "x", "y" and "label" to the txt file.

isNewFile()?

if (isNewFile())

{

```
        bw.write("=========================================="

        +System.getProperty("line.separator"));

        bw.write("'MouseClickRecorder' version: \t" +this.ProgramVerisonID

        +System.getProperty("line.separator"));

       bw.write("Image_filename:"+fileName +System.getProperty("line.separator"));

       bw.write("Image_height:" +getImageHeight()+System.getProperty("line.separator"));

       bw.write("Image_width:"+getImageWidth()+System.getProperty("line.separator"));

       bw.write("Image_file_size:" +getImageSizeString() +System.getProperty("line.separator"));

         bw.write("=========================================="
        +System.getProperty("line.separator"));

        bw.write("Row | Column\t" +"Comment" +System.getProperty("line.separator") );

       this.newFile = false;

        }

bw.write(x+" | "+y+" \t"+label+System.getProperty("line.separator"));
 bw.close();
```



```
private  boolean saveAnnotation(File directory, String fileName, int x, int y, String label){
    int extPos = fileName.lastIndexOf(".");
    String annotationFileName = fileName.substring(0,extPos+1) + "txt";
    File annotationsFile = new File(directory.getAbsolutePath()+System.getProperty("file.separator")+annotationFileName);

    try {
            BufferedWriter bw = new BufferedWriter(new FileWriter(annotationsFile,true));
            if (isNewFile())
            {
                bw.write("==========================================" +System.getProperty("line.separator"));
                bw.write("'MouseClickRecorder' version: \t" +this.ProgramVerisonID +System.getProperty("line.separator"));
                bw.write("Image_filename:"+fileName +System.getProperty("line.separator"));
                bw.write("Image_height:" +getImageHeight()+System.getProperty("line.separator"));
                bw.write("Image_width:"+getImageWidth()+System.getProperty("line.separator"));
                bw.write("Image_file_size:" +getImageSizeString() +System.getProperty("line.separator"));
                bw.write("==========================================" +System.getProperty("line.separator"));
                bw.write("Row | Column\t" +"Comment" +System.getProperty("line.separator") );
                this.newFile = false;
            }
            bw.write(x+" | "+y+" \t"+label+System.getProperty("line.separator"));
            bw.close();

    } catch (IOException e) {
            return false;
```

isNewFile(): checks if in the current folder there exists a file with the "filename" and extension "txt". It returns "true" if file is not there.

```
String temp = file.getPath().substring(0,file.getPath().length()-4) + ".txt";

      File tempFile = new File(temp);

     if(tempFile.exists())

            this.newFile = false;
```

What's Next?

Future plans for the Mouse Click Recorder.

- Allow the user to save coordinates without entering a label.
- Implemented a remove/undo button for deleting coordinates.
- Start up menu. Current MCR immediately prompts user for a folder. Next version could potentially have a menu page to indicate MCR is running before prompting user.
- Fix header block for the output file. It currently only outputs the header just the first time the file is created, and not additionally following the opening of other files.

Similar Applications

The following links provide information about applications similar to MCR. One of such is the facebook's image tagging application:

- http://209.85.173.132/search?q=cache:fTbJfkaVe7wJ:www.codeproject.com/KB/aspnet/ImageTagArticle.aspx+facebook+tagging+code&cd=6&hl=en&ct=clnk&gl=us
- http://www.designers-chair.com/2009/03/facebook-picture-tagging-system/

Ripping the Microsoft Bluetooth Stacks

1) Download installation packages
    - You may download the WIDCOMM drivers from http://www.rapidsharedownload.net/software/widcomm-bluetooth-software-5.1.0.1100/ or http://www.broadcom.com/support/bluetooth/update.php
    - According the dev-hack, after version 5.1.0.1100 the driver stops supporting 3rd party chipsets, specifically CSR chipsets. However, it does not seem to be a major problem if you download a later version.

2) Determine which Bluetooth device you have
    - Once you have downloaded and extracted the drivers, it's time to minimize your folder and head into your device manager to determine exactly what hardware you have installed. **Make sure you have your Bluetooth adapter inserted in your computer (working or not).**
    - Right click "Computer" on your desktop, and then select **Properties**
    - On the top left click **Device Manager.**
    - Expand your Bluetooth tree to see your Bluetooth radio (if installed) or look at **Other Devices** to see if it appears there (if not installed).
    - Double click your Bluetooth radio device (The name may vary depending upon manufacturer and model). Then click the **Details** tab.
    - Click the drop down box and select **Device ID**
    - Now with that window open go back into the extracted drivers folder. We need to check and see if the installer already supports your device.
    - In the root folder go into the appropriate folder for your system (it should be in C:\Program Files\WIDCOMM\Bluetooth Software\bin). If you're in 32 bit Vista, enter **Win32**. For 64 bit, enter **Win64.**
    - Open up **btwusb.inf** in notepad (should already be the default viewer so you can just double click it.)
    - Now look at your hardware ID in the device manager it should be formatted like this:
        - USB\VID_XXXX&**PID**_XXXX where "XXXX" is a 4 digit hexadecimal number (There may be revision information after the basic Hardware ID's such as: USB\VID_413C&PID_8126&REV_0100).
    - Quickly copy or memorize the 4 digit number after **VID.**
    - In notepad hit Ctrl+F to open the find dialog. Enter your VID as **VID_XXXX** where the XXXX is your four digit number specific to your device. Hit Enter.

- If your device is found, then look again and confirm that your **VID and PID** numbers match exactly then and only then may you skip to **Step 4 -=- Removing Old Drivers Completely**
- If you did not find your VID and PID then please continue on to **Step 3 -=- Preparing the New Installation.**

3) Preparing the new installation
   - Since your device wasn't specifically included in the installer script that doesn't necessarily mean that the drivers aren't compatible. There's a very good chance that they are and should work fine (since most Bluetooth devices are created alike and their functionality is determined by the profiles they support). In this step we need to modify the installer script to include your specific device. You should already have btwusb.inf opened up for your specific platform. Add the following lines:
     1. Under [ControlFlags] section add:
        ExcludeFromSelect=USB\VID_XXXX&PID_XXXX (change the XXXX with the numbers from your device)
     2. This varies for different versions of windows.
        - 32bit windows:
          Under ***[WIDCOMM.NTx86.5.1]*** add ***%****.DeviceDesc%=BTWUSB, USB\VID_XXXX&PID_XXXX ; My BT USB Dongle*** Where "****" is a string or single word that's all caps and would be the name of your hardware vendor. EXAMPLE: %DellFangorn.DeviceDesc%=BRSMARTUSB, USB\VID_413C&PID_8126
        - 64bit Windows:
          Under ***[WIDCOMM.NTamd64]*** add ***%****.DeviceDesc%=BTWUSB, USB\VID_XXXX&PID_XXXX ; My BT USB Dongle*** Where "****" is a string or single word that's all caps and would be the name of your hardware vendor. EXAMPLE: %DellFangorn.DeviceDesc%=BRSMARTUSB, USB\VID_413C&PID_8126
     3. Save and Close the file.

4) Removing old drivers completely
   - What we're going to do here here is effectively remove the installer script files from Windows' view. So now when the device is found it won't be able to find a driver. And you should select "Don't ask Again" when prompted to search for new drivers for your device after you uninstall it.
   - Now, if you still have your device manager window open and you already have the Windows or manufacturer driver installed, please uninstall ANYTHING Bluetooth related. (HINT: If you remove the Bluetooth Radio device first, it will

take everything else with it.) You can uninstall devices simply by Right clicking them and selecting Uninstall.

5) Disabling Microsoft Bluetooth stack completely

Now that the installation is prepared we have to make sure that Microsoft's automatic and crippled Bluetooth driver doesn't just jump in the way and automatically install itself when you're trying to install the new driver. We have to disable Microsoft's installer script for Bluetooth devices.

1. Navigate to **C:\Windows\inf\**
2. Rename **bth.inf** to **bth.inf.old;**
3. Rename **bth.pnf** to **bth.pnf.old**

Once everything is uninstalled, close all files and dialogs and restart your machine. **NOTE: This is important! If you are running Windows Vista x64, you absolutely HAVE to press F8 on boot. Right after your BIOS post, pound F8 until you're presented with a menu. You MUST select "Disable Driver Signing Enforcment" or you will not be able to install this driver. As of yet, there is no other way to disable this other than attaching a debugger to the kernel which isn't practical.**

6) Installing WIDCOMM drivers

And now is the time you've all been waiting for. Time to run the installer for the WIDCOMM Drivers.(Please see the bright red note above if you're running Windows Vista RTM x64). Make sure your Bluetooth Device is **unplugged** from your machine.

1. Browse to where you have the drivers unpacked.
2. Browse directly into your platform folder(i.e. Win32 or Win64)
3. Do not run **Setup.exe**
   instead run **Inst.exe.**
4. Go through the installer, when it comes up and says that no Bluetooth device detected, click **CANCEL** to continue the installation without installing a device first. It will always say this even if your device is internal or still connected.
5. Windows will bitch at you for not installing signed drivers about 11 times so go ahead and make sure you're clicking away at "**Install Anyway**"
6. The WIDCOMM drivers will install the following virtual devices and profiles:
   1. Bluetooth Communications ports(COM 4 and 5)
   2. Bluetooth LAN Access Server Driver
   3. Bluetooth Virtual HID Mouse
   4. Bluetooth Virtual HID Keyboard
   5. Hands-Free Audio
   6. Stereo Audio(also known in xp as High Quality Bluetooth Audio or A2DP)

7. Once Vista is done installing the various devices then click Finish in the installer to close it.

You'll now notice You'll have the Bluetooth icon down in the task bar which should be red and a My Bluetooth Places icon on your desktop. Do not click either of them yet. We're almost done.

7) Installing your device

- Plug your Bluetooth adapter in(if it is external)
- If your adapter is internal, then open up the Device Manager and click **Action>Scan for Hardware Changes** to initiate the installation process.
- When it asks you to search for drivers, select to Browse for the location yourself.
- Browse to the following: **C:\Program Files\WIDCOMM\bin**
- Then click ok and the installer will find the drivers assuming you correctly edited the installer script.
- Once the device is installed(if successful) You're ready to rock.

8) Enjoying unrestricted Bluetooth functionality

- Once everything is installed and patched you should be able to right click the Bluetooth icon down in the task bar, select **Begin Using Bluetooth.** With any luck, you'll start the Bluetooth Wizard and be able to choose which profiles your computer will provide to devices. If you get a license error, then something went wrong with the patching or you didn't choose the right device in the patcher. If you get a device not found, then I recommend completely uninstalling the device and starting from Step 6.
- Should you need to Uninstall and return to your default driver then you need to open up Device Manager then, through the control panel, remove the WIDCOMM Drivers(Add/Remove Programs). The drivers will prompt you to remove your device. You can do that or just right click your Bluetooth radio in the Device Manager and click Uninstall, the driver uninstallation will resume automatically.

List of Compatible and Incompatible Bluetooth Devices

Compatible Bluetooth devices
(http://wiibrew.org/wiki/List_of_Working_Bluetooth_Devices)

Working Bluetooth Devices on XP/Linux, but are not compatible with Vista

- ISSC Bluetooth Device with included BlueSoleil 1.6 drivers. Works much better with widcomm stack 5.0.1+
- Integrated System Solution Corp. KY-BT100 Bluetooth Adapter USB2.0 (Debian-Kernel 2.6.18/bluez, WMD)
- Swann USB Bluetooth adapter (1.1). Really an ISSC adapter. Comes with old version of BlueSoleil drivers, can download new version. Both old and new versions work fine, except both reboot computer if you try to read wiimote serial (Serial number is disabled in GlovePIE 0.24 for this reason). Works better with Widcomm stack. Will not work on Vista no matter what you do.

Working Bluetooth Devices in alphabetical order

- a-Quip A/BT-1 USB 2.0 Bluetooth Dongle works with included BlueSoleil 2.3 driver
- Abe UB22S (The USB Dongle sold by LEGO for the Mindstorms Robot. Recognized Wiimote and connected right away. Had some difficulties getting it communicate with GlovePie but it works with standard Windows XP Bluetooth stack.)
- Acer BT-700 Bluetooth USB 1.1 Dongle (Works with the newest BlueSoleil version)
- Acortech ES-388 Bluetooth USB Adapter. Comes with an old version of BlueSoleil that does not seem to work, but the current 2.3 version works great.
- Advent USB Bluetooth adapter, v2.0 + EDR, Class 1 (P/N ADE-C1EDR) (Toshiba Bluetooth Stack)
- Apple computers - all internal Bluetooth Cards in (MacBook, Mac Mini, iMac, etc.)
- Anycom USB-200. Install drivers that come with product then install drivers from http://www.broadcom.com/products/bluetooth_update.php.
- Asus USB-BT21 Mini Bluetooth v2.0+EDR. Works using the drivers from the Broadcom website. Skip the pairing process.
- AZiO USB Bluetooth v2.0 + EDR Class 1 adapter dongle (Uses Toshiba bluetooth stack)
- Belkin Bluetooth USB Adaptor F8T001 v2 (using BlueSoleil, WIDCOMM untested)
- Belkin Bluetooth USB Adapter F8T003 ver. 2 (using BlueSoleil, WIDCOMM untested)
- Belkin Bluetooth USB Adapter F8T009
- Belkin Bluetooth USB Adapter F8T012 under XP works with: http://www.broadcom.com/products/bluetooth_update.php
- Belkin Bluetooth USB Adapter F8T012uk (not under Vista)
- Belkin Bluetooth USB Adapter F8T013 under XP works with: http://www.broadcom.com/products/bluetooth_update.php

- "Billionton Class 1" (Fake - is really Silicon Wave). LED is centered, red and flashes. Hardware IDs: USB\Vid_0c10&Pid_0000&Rev_1500 The 'antenna' is fake. Works with Microsoft, BlueSoleil (came with, works with latest) and Widcomm stacks.
- BlueSynchrOne COM ONE 1 Bluetooth Dongle USB
- Cables Unlimited Bluetooth Adapter with BlueSoleil driver.
- Cambridge Silicon Radio, Ltd Bluetooth Dongle (HCI mode) (Bus 001 Device 048: ID 0a12:0001)
- Cellink BTA-6030 Bluetooth Dongle using BlueSoleil.
- Cirago BTA-6060 with BlueSoleil 2.6.0.8 r.070517 (stack 06.03.27.20061108) from cirago (tested in WinXP Pro SP2)
- Cirago BTA-3210 USB 2.0 Micro V2.0+EDR Bluetooth Dongle, comes with with Toshiba Stack (Tested in WinXp Pro SP3)
- Cirago Bluetooth BT v2.0 Micro USB Adapter, comes with with Toshiba Stack (Tested in WinXp Pro SP3)
- CompUSA brand USB dongle using BlueTooth version 1.2 and using the default Toshiba stack.
- Conceptronic CBT100U with BlueSoleil driver.
- Conceptronic CBTU2A with Toshiba Stack
- CUWAN BT-100M (no driver on Linux, driver CD included for Windows) - cheap Chinese dongle (~6-8$)
- D-Link DBT-120 (B3 or B4 or C1) using Toshiba stack or BlueSoleil stack. (Toshiba connects faster and easier)
- D-Link DBT-122 with default D-Link driver AND this update for it: http://www.broadcom.com/products/bluetooth_update.php
    - Note: for H/W Ver: C1, if you use the drivers and such from the installation CD, the Wii remote will connect but you will not be able to use programs like GlovePIE. I ended up uninstalling those and using the stack from the link above and it worked like a charm after that.
- Dell Bluetooth USB Reciever Mouse/Keyboard combo (same as Logitech Mouse/Keyboard above)
- Dell TrueMobile Bluetooth Modules (Standard issue included with many Dell Laptops) Latitude D820 has onboard module that works best with BlueSoleil
- Dell Wireless 355 Module with Bluetooth 2.0 + EDR Technology (WIDCOMM, tested on a Dell Inspiron E1705)
    - (Jan 19-2008) Doesn't work on Dell Inspiron 1520 on Vista with any Dell WIDCOMM driver or microsoft stack (even though it is a broadcom device the drivers from the broadcom web site refuses to be installed for this device). The Wiimote gets connected and GlovePIE sees it with the bluetooth hack and allows setting leds and rumble but can't read any buttons, camera

    or accelerometer data. Wiinremote recognizes the device but can't receive any data.
  - (Jan 19-2008) The bluesoleil hack for this device sometimes makes it work for a couple of minutes before requiring a reinstall to be able to see it again and also breaks compatibility with some other bluetooth devices (headsets mostly).
  - (Feb 12-2008) I attempted it with my original WIDCOMM stack and had no luck. After I went out and purchased a Kensington USB dongle and installed the included WIDCOMM stack. After I did, I tried out the internal Bluetooth and it worked. So, it seems if you get an updated WIDCOMM stack you can get it to work, because I got it working, no weird hacking required.
- Dell Wireless 360 Module with Bluetooth 2.1 + ERD Technology (Using Toshiba stack)
- Delock USB Bluetooth adapter (Uses widcomm bluetooth stack)
- Elegance generic Bluetooth USB adapter v1.2 (Works with latest BlueSoleil drivers).
- EMTEC USB dongle Bluetooth 100m (EKCOB100) using BlueSoleil.
- ENCORE ENUBT-C1E USB 2.0 Bluetooth Adapter (BlueSoleil - Comes with Dongle)
- GXT Bluetooth Class 1 USB Dongle
- Hewlett-Packard (HP) Compaq nc6120 integrated Bluetooth module. Working with Microsoft stack, does not work with original drivers (Widcomm stack).
- Hewlett-Packard (HP) BT450 with included Widcomm driver. Working with WIDCOMM tutorial above.
- IBM Thinkpad Integrated Bluetooth II (X31 series, maybe more) - WIDCOMM v5
- IOGEAR USB BLUETOOTH ADAPTER GBU201 WORKS PERFECTLY ON WINDOWS VISTA 32 BITS WITH THE TOSHIBA BLUETOOTH STACK 5.10.06(T) AND ALSO WITH THE TOSHIBA BLUETOOTH STACK 5.10.12(T). BUT DO NOT WORK WITH THEIR OWN STACK.
- IOGear USB dongle model GBU211 using default stack (BlueTooth version 1.2) (2.0 doesn't work with Blue Soleil, and does not work with the default stack)
- **IOGear Model GBU221** (NOT to be confused with GBU211) BT version 2.0 (GBU211 is version 1.2) using default MS stack, and will not connect to BlueSoleil. Works in GlovePIE 0.26 with the "TroubleShooter > Bluetooth Fix" menu. Works in MiiTransfer. Works under Linux. Uses same chipset as Wii's BT board (BCM2045)
  - BCM2405 adaptors in Windows don't like it when connected Wiimotes go into powersave mode and require a full resync to reset the Wiimote if this happens.
  - GBU221 worked on one computer using Vista's BT drivers with RMX with one Wiimote, but then doesn't work on two other computers with Vista and XP with a different Wiimote. I'm wondering if it's the Wiimote. I'll have to try it out with the Wiimote that worked. --Yqbd 00:48, 11 March 2007 (PST)

    - GBU321 worked perfectly on Vista Ultimate computer with serial IOGear driver.
    - Used GlovePIE Bluetooth Fix and GlovePIE is able to change the lights and make it rumble, but it's not getting the accelerometer and camera data.
    - Seems to be working now with: http://www.broadcom.com/products/bluetooth_update.php
- JustCom 1.1/1.2/2.0 BTD1-6300E, (wrong VID listed), Cambridge Silicon Radio, comes with Toshiba stack (works with Microsoft and Toshiba)
- Kensington Bluetooth dongle (model 33348) using WIDCOMM stack - (see Discussion page for troubleshooting)
- Kensington 33085CA with included Widcomm driver. Working with WIDCOMM tutorial above.
- Laser AO-USBBD using the latest BlueSoleil driver. Download from BlueSoleil website.
- Lenovo/IBM Thinkpad X60s Integrated BCM2045b using Microsoft Stack (tested with Vista), requires Latest driver as of 7/2007,
    - Works, but must pair for each use since the wiimote doesn't use a passkey, which is required for automatic pairing.
    - DOES NOT WORK. SEE NON WORKING SECTION FOR DETAILS
- Lenovo 3000 N100, integrated Broadcom 2045 - WIDCOMM v5.0.1.1500 ~ connect using 'Bluetooth Setup Wizard' ~ 'I want to find a specific Bluetooth Device', NOT using 'View Devices in Range'
- Logitech MX Bluetooth Mouse/Keyboard combo (dongle m/n:c-uv35) - WIDCOMM (Had trouble with with WIDCOMM 5.0.1.2800 drivers under XP). Works fine with BlueSoleil 3.2.2.8 stack version 05.04.11.20060413 WIDCOMM drivers would not pair. In the bttl.ini file you must modify one of the USB device VID and PID to VID=046D PID=C709 and rename the Manufacture to Logitech then start Bluesoleil. It will recognize it then. If Wiimote dosent show up, to to Tools...Find Device... and search by name "Nintendo". It will show up. Don't pair it, just click connect. Also, one important note, as you plug in the dongle, hold the red connect button to put it in dongle mode rather than embedded mode for keyboard and mouse.). Works fine with Toshiba Stack, Version 6.10.07.2. As with BlueSoleil, you have to plug it in with dongle mode instead of embedded. Widcomm worked but turned not only my system unstable(for certain them, nothing else), thus BlueSoleil or Toshiba recommended.
- Logitech LBT-UA200C1 - Works with Bluesoleil.
- Maplin unbranded BTU-22A2 Bluetooth EDR Dongle (stock code A35GU), runs BlueSoleil (Windows XP)
- Mikomi Bluetooth v1.2 Adapter BC03RUT1-01. Works with the included Bluesoleil 1.6 Stack.

- Motion Computing LS800 internal Bluetooth adapter - Cambridge Silicon Radio BC417 (CSR BlueCore4) VID_10AB&PID_1005&REV_1657 - Works with Toshiba drivers (Tested on Vista, should be fine on XP too). Microsoft Stack on Vista can read buttons, but cannot read gyrometer, VGA cam, battery, and cannot set LED's or rumble. Nunchuck not tested, but expect that it works with Toshiba stack as everything on main controller functioned fine. Also tested BlueSoleil drivers but could not get them to recognize BlueTooth radio, despite the fact that the VOIP version for Vista says CSR only. Toshiba stack need not be purchased as Toshiba stack is the official stack for the LS800. Motion Computing recommends Microsoft Stack for Vista currently because of a "known issue" with Vista and Toshiba stack, however no problems have been encountered using the Toshiba stack on Vista
- Motorola MPT 3.0 BT USB - Works with latest Bluesoleil. Not tested with included BT stack.
- MSI 3X Faster Star Key Bluetooth 2.0 Transceiver
- MSI BToes Bluetooth 1.2 using BlueSoleil
- MSI BToes 2.0 using BlueSoleil.
- MSI BToes 2.0 using BlueZ (under Ubuntu 7.04 Herd 5, Kernel 2.6.20, CWiid drivers)
- MSI pc2pc using BlueSoleil (VID : 0A5C / PID : 200A) or using Microsoft stack (tested on Vista)
- MSI Star Key 2.0 USB Bluetooth 2.0 Transceiver (BlueTooth 2.0, using widcomm bluetooth stack, available on msi's web page), tested on Debian Lenny 2.6.24-1-686
- Planet BT-200U - Works with Bluesoleil, but not with Bundled WIDCOMM
- Sepia SPA-510 Bluetooth 2.0 EDR USB Dongle
- Silicon Wave Internal Bluetooth (Toshiba stack)
- Silicon Wave Exter USB Dongle
- SiteCom USB Bluetooth dongles
    - Type cn500 with BlueSoleil stack (tested: 2.3 standard and 2.6.0.6)
    - Type cn502 with Widcomm stack
    - Type cn521 with BlueSoleil and Toshiba stack
- Sony Vaio FE21M internal Toshiba Bluetooth chip, using Toshiba stack
- Sony Vaio SZ2, TX2 internal Bluetooth (Toshiba)
- Syba SD-U1BTC2-IS - With included BlueSoleil driver.
- Targus USB Bluetooth adapter (BlueTooth 2.0, Widcomm bluetooth stack)
- TDK BRBLU04 (PC card): Worked with its own Widcomm drivers (probably works with the default ones). Untested on Vista.
- Technika USB bluetooth adaptor (TechBlue1) with supplied driver stack
- Tecom USB Dongle "Bluetooth USB EDR Dongle BT3034" (BT v. 2.0) (after days and days of work and reconfig. i managed to get it to work fully. will update with exact stack etc..but its the widcomm drivers..)

- TrendNet TBW-101UB (BlueSoleil)(also ships with WIDCOMM stack ver 5.1 which gives access denied in wiin remote - works when BT virtual keyboard HID is uninstalled before scanning for hardware changes with wiimote in discovery)
- TrendNet TBW-104UB (BlueSoleil)
- TrendNet TBW-105UB (BlueSoleil, BlueCove on WinXP, AvetanaBT on Linux Redhat) Works with http://www.broadcom.com/products/bluetooth_update.php
- Trust Bluetooth 2.0 USB Adapter BT-1300tp (using WIDCOMM 4.0.1.700 (Downloadable from the Trust website))
- Trust Bluetooth 2.0 EDR USB Adapter BT-2100p (using Toshiba stack 4.00.35 included or BlueSoleil)
- Trust BT-2400p (Ultra Small) (Broadcom 2045 2.0), BUT PLEASE SEE the LIST of Non-Working Bluetooth Devices BELOW.
- Trust Bluetooth 2.0 EDR USB Adaptater BT-2150p
- Trust Bluetooth 2.0 EDR USB Adapter BT-2200Tp (using both WIDCOMM and BlueSoleil)
- Trust Bluetooth 2.0 EDR USB Adapter BT-2305p (using Toshiba stack)
- Zolid Bluetooth 2.0 Class 1 apapter (Comes with Bluesoleil 3.0)
- Zoom 4310 using Toshiba (v 3.03.13) stack.
- Zoom 4311 (comes with BlueSoleil v 3.2.2.8 r070421)
- Zoom 4320AF using BlueSoleil stack.

Non-Working Bluetooth Devices

- Anycom 'Blue 250' (links with Wiimote and receives data packets but nothing else (under Windows, but Linux works fine). BlueSoleil drivers won't even see this dongle)
- Belkin Bluetooth USB Adapter F8T013xx1, F8T012 (links with Wiimote and receives data packets but nothing else, BlueSoleil won't recognize the dongle, Microsoft stack works but only partially (only led and rumble nothing else.)) But works perfectly on Linux.
    - Works on modified BlueSoleil only. Other stacks work, but require the program to be adjusted to non-BS-stacks.
    - UPDATE: F8T012 works with: http://www.broadcom.com/products/bluetooth_update.php
- D-link DBT-120 USB adapter Bluetooth 2.0 (recognised wiimote, sends and recieves packets, but won't read movement).
- Digicom Palladio 100M with Widcomm 5.1.0.1100 driver. Finds RLV and HID but is unable to finish pairing, bytes sent (1), bytes received (5).
- IBM Thinkpad A31p Integrated Bluetooth, Software version 1.4.2 (based on Widcomm or MS Stack ?), Recognizes Wiimote but won't pair, not tested with BlueSoleil

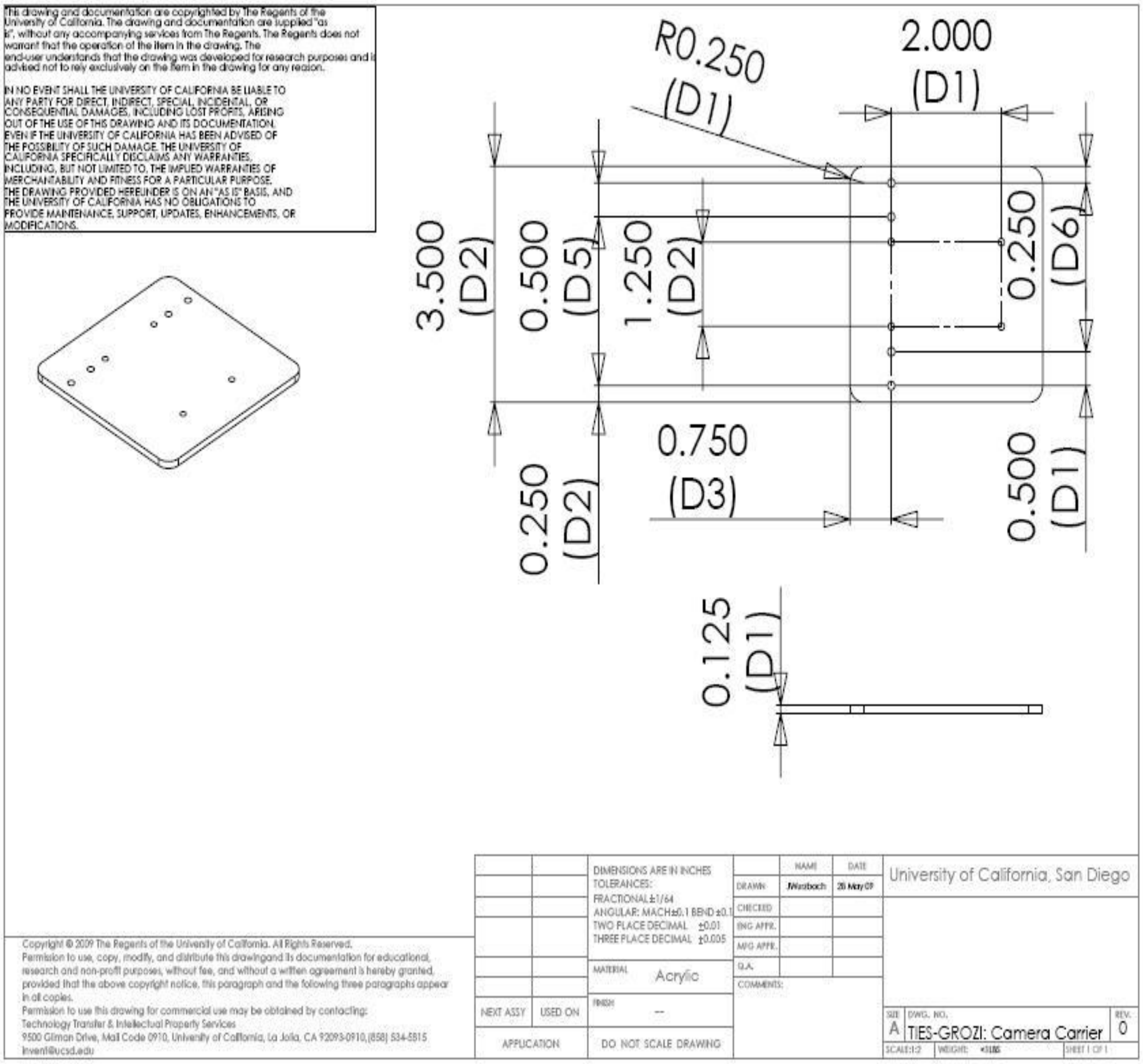
- Jabra A320s Bluetooth Stero USB Adapter. Wiimote pairs for brief second, then looses connection.
- Jensen WBT431 (bought it for $10; recognizes wiimote, but doesn't pair with it)
- Micradigital £9.99 argos model - will recognise Wiimote, but won't get past the pairing stage
- Microsoft Wireless Transceiver for Bluetooth 2.0 (BlueTooth version 2.0 using Microsoft Stack)
- Microsoft Wireless Transceiver for Bluetooth 3.0 (Using Microsoft Stack) - Sees and connects to Wiimote. Test programs can find and recognize it, but data is never received.
- Motorola PC820 (Tried WinXP SP1, SP2. Widcomm drivers (Detects, connects, drops,) Windows drivers, Bluesoleil (unsupported, so hacked, still nothing.))
- Motorola PC850 dont work on windows Vista (testing lot of drivers). will just recognise Wiimote. No sound device work.
- OvisLink BT 201-202 (tested with latest Widcomm and Bluesoleil stack) - will recognise Wiimote, but won't get past the pairing stage
- Quatech BTA-B01U (Broadcom chip w/antenna bundled with Widcomm stack) Recognized Wiimote but won't pair, and BlueSoleil won't recognize the dongle.
- Sandberg bluetooth link
- Targus Rechargeable Bluetooth media notebook mouse software version 1.00.01(just bought it and Vista cannot see it. Targus software out of date no plans for a patch)
- TrendNet TBW-105UB Broadcom 2045 chip, Widcomm stack. After update, will pair with Wiimote and let you send LED and rumble data to it, but will not read buttons or any input. Bluesoleil incompatible - if hacked to support the dongle, it will think it is disconnected a second after activating it. Windows stack doesn't work either. Search messageboards and you'll find no one can make this thing work with a Wiimote.
  - UPDATE (9/30/08) This dongle has been tested and works perfectly with BlueCove on Windows XP and AvetanaBT on Linux Redhat.
- Trust BT-2400p (Ultra Small) (Broadcom 2045 2.0). With its bundled software the Wiimote is seen but not paired in any way. Probably this will be done with some special settings.
  - UPDATE (4/05/09) After some further tries actually this device works with BT 1.1 old win stack (even if you have also installed BT 2.0 of Toshiba at the same time) BUT you must remember, even once you have had it paired at least one time, to always use the "Add device" command, even when the wii icon is visualized among the previosly recognized devices. If you repeat the pairing procedure this way EVERY TIME (with about 3 or 4 attempts each, probably due to some bugs on the BTHENUM Registry associated Key) you'll

be always able to finally connect to the wiimote via GlovePie. The regedit key "HKEY_LOCAL_MACHINE\SYSTEM\ControlSetXXX\Enum\BTHENUM\{00001124-0000-1000-8000-00805f9b34fb}_VID&0002057e_PID&0306" will continue to add +1 known devices, but my experience left room to at least 50 instances without futher problems, and if there will be a limit we'll always be able to restore a previous registry backup with a smaller number in this particular key (manually in realtime you can't because the key is locked by the system).

Unknown-Working Bluetooth Devices

- Abocom Model: UBT3KH P/N: UBT3KH03910 Geeks.com
- AnyCom 2.0 Bluetooth Dongle + Widcomm Bluetooth Software 5.1.0.3300 on a Dell laptop
  Note: Problematic - Works immediately after fresh stack install. After reboot & device re-insertion, it connects but can lose data stream. (Works with Linux on my Dell Laptop and normal PC :))
- Billionton, Cambridge Silicon Radio, Hardware Ids USB\Vid_0a12&Pid_0001&Rev_0525, USB\Vid_0a12&Pid_0001 worked with bluesoleil's trial drivers using GlovePIE and WiinRemote, but could not get it to work using MS Bluetooth and widcomm 1.4.2 build 10. widcomm 1.4.2 build 10 drivers discovered the wiimote and hid service and displayed packets being sent and received when using GlovePIE and WiiRemote, but GlovePIE and WiiRemote couldn't seem to work with the controller? Still trying.
- Lenovo/IBM ThinkPad T61: Using ThinkPad Bluetooth w/ EDR and Microsoft Bluetooth Enumerator. Wii Remote can be found (connects as Nintendo RVL-CNT-01); however, cannot communicate reliably. Rarely is able to send data to computer.
- Made-in-China brand. "BT2.0" VID=0A12 (Cambridge Silicon Radio, Ltd) PID=0001. Comes with Extended Systems XTNDConnect Blue Manager. Driver: Windigo Systems, PBTUSB02C2 Bluetooth, csrbc01.sys (see discussion page)

Computer Aided Design (CAD) Drafts



DIMENSIONS ARE IN INCHES
TOLERANCES:
FRACTIONAL ±1/64
ANGULAR: MACH±0.1 BEND ±0.1
TWO PLACE DECIMAL ±0.01
THREE PLACE DECIMAL ±0.005

MATERIAL Acrylic
FINISH --
NEXT ASSY | USED ON
APPLICATION
DO NOT SCALE DRAWING

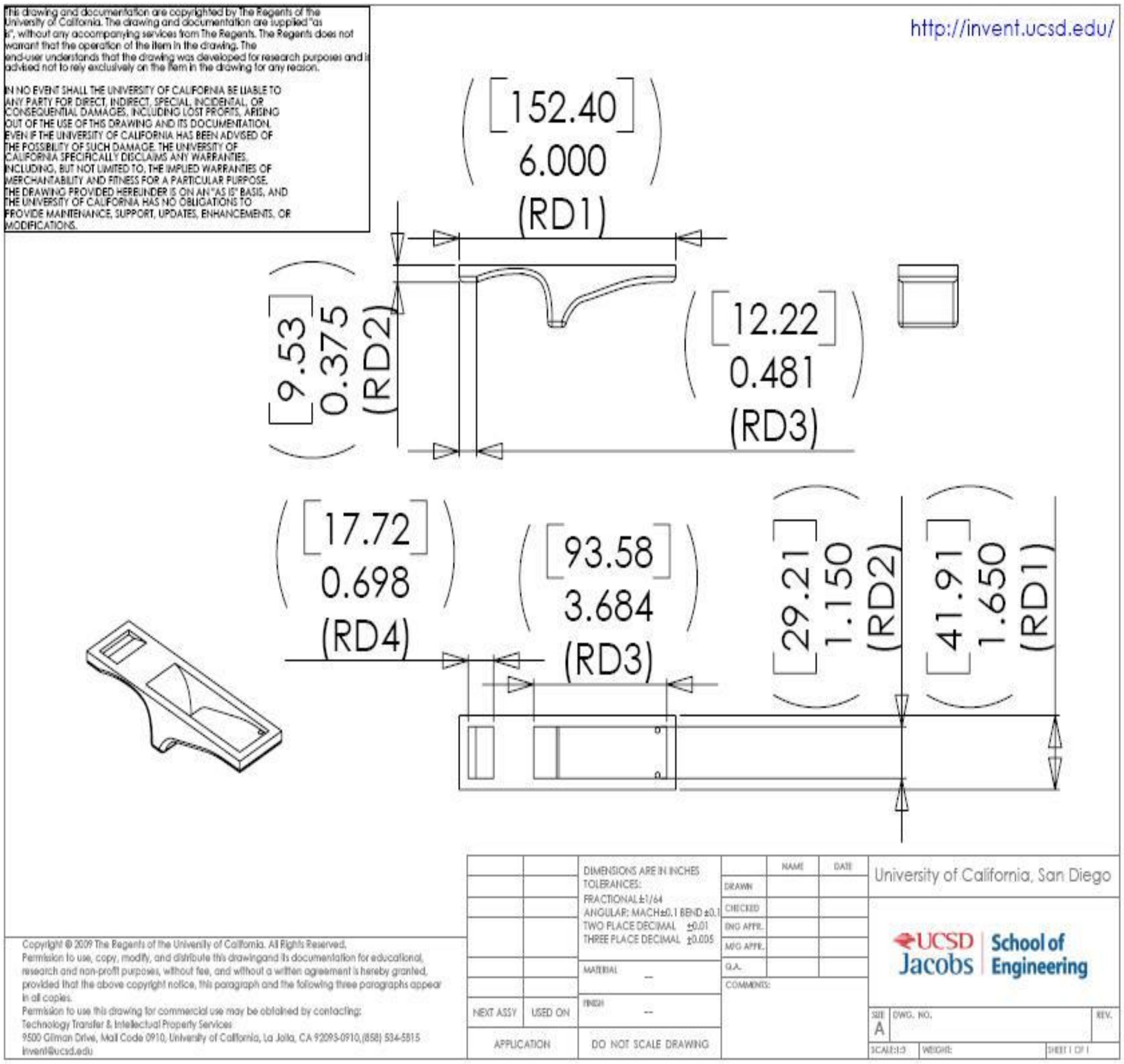
	NAME	DATE
DRAWN	JWasbach	28 May 09
CHECKED		
ENG APPR.		
MFG APPR.		
Q.A.		

COMMENTS:

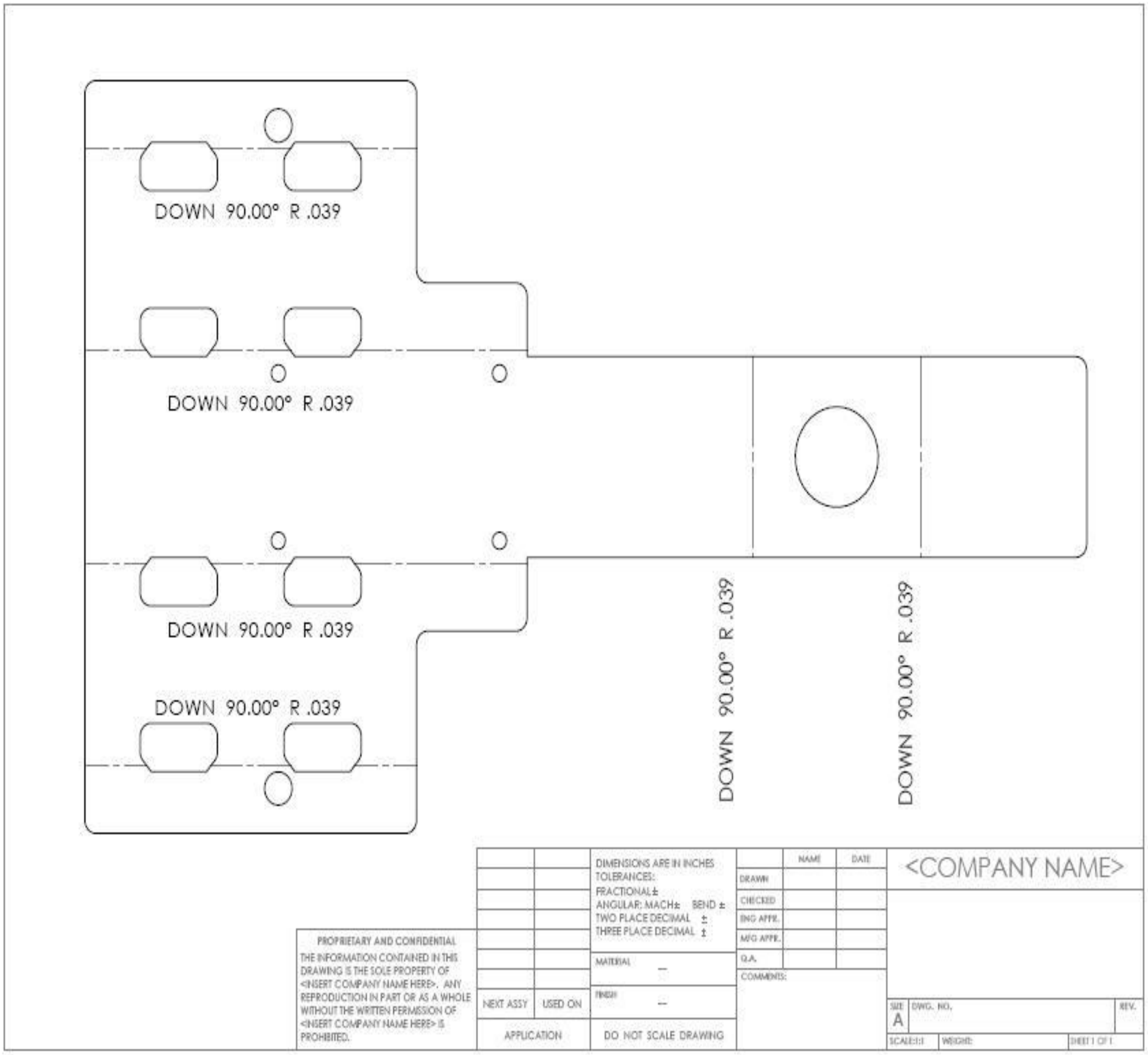
University of California, San Diego

SIZE A | DWG. NO. TIES-GROZI: Camera Carrier | REV. 0
SCALE:1:2 | WEIGHT: [illegible] | SHEET 1 OF 1

CAD Draft of Camera Carrier



CAD Draft of Shoulder Mount



CAD Draft of Wiimote Carrier

References

1. http://www.wiili.org/index.php/Compatible_Bluetooth_Devices

2. http://wiibrew.org/wiki/List_of_Working_Bluetooth_Devices

(Compatible Bluetooth Devices)

3. http://www.wiimoteproject.com/bluetooth-and-connectivity-knowledge-center/a-summary-of-windows-bluetooth-stacks-and-their-connection/

4. http://www.dev-toast.com/2007/01/05/uncrippling-bluetooth-in-vista-rtm/

5. http://www.rapidsharedownload.net/software/widcomm-bluetooth-software-5.1.0.1100/

6. http://www.wiili.org/forum/bluecove-210-on-bluez-tips-t6355.html

http://java.sun.com/docs/books/tutorial/java/nutsandbolts/variables.html (for anyone who wants to learn Java)

7. http://xii9190.wordpress.com/page/15/ (mainly to find out how to set vibrate)

http://www.wiili.org/forum/wiiremote-disconnection-problem-t5359.html (help on disconnection problem, but I did it slightly differently in my code)

8. http://www.wiili.org/forum/bluetooth-fails-to-initialize-without-wiiremotedisconnect()-t6805.html

http://wiki.multimedia.cx/index.php?title=PCM (Audio file needs to be a signed 8-bit PCM)

9. http://grozi.calit2.net/files/TIESGroZiWi09.pdf