April 2008

E-Merging Realities

Keith Chester
Worcester Polytechnic Institute

Robert Martin
Worcester Polytechnic Institute

William David Price
Worcester Polytechnic Institute

E-MERGING REALITIES

An Interactive Qualifying Project Report

submitted to the Faculty

Of the

WORCESTER POLYTECHNIC INSTITUTE

In partial fulfillment of the requirements for the

Degree of Bachelor of Science

By

_____

Keith Chester

and

_____

Robert Martin

and

_____

William D. Price

Date: April 23, 2008

_____

Professor Joseph Farbrook

Table of Contents

Abstract

E-Merging Realities is a project undertaken to utilize technology to display art and raise social awareness. An artistic design was built within the game Second Life that shows images designed to increase the users awareness of issues such as third world hunger. A special wheelchair controller was designed and a prototype built that is used to navigate through the piece. Ultimately we ended with a complete piece that uses interactive technology to address real human issues.

Executive Summary

The life style of the average modern American has become fast-paced and filled with the use of technology. In many peoples lives art can take a backseat to technology or even be forgotten about. The combination of art and technology is by no means a new concept, yet it is still in its infancy. Many experimental pieces appear on the Internet and some schools, such as Worcester Polytechnic Institute have even added Interactive Media major fields to help cultivate this new form of expression.

E-Merging Realities is an attempt to create a piece of art that utilizes technology to raise awareness on social issues. The initial concept of this project was to create some sort of link between a virtual world, in this case the game Second Life, and the real world. The goal was to have a stimulus in the real world create a reaction in the virtual world. This basic concept of cause and effect is basic enough, and used on a daily basis through the use of keyboards, mice, and other input devices. The goal of this project was not only to create this interaction, but also to do so in a way that holds meaning to the user, and draws an emotional response.

Many ideas were tossed around in the initial planning stages of the project that involved users in Second Life affecting something in the real world. When we were unable to nail down a concept using Second Life user input that we were comfortable with, we set to using the link to create it's own meaning.

Our advisor, Professor Farbrook, had experience with virtual art and suggested the creation of a piece within Second Life where would have some piece of real life hardware interact with the piece, or even be a part of it.

What we came up with was a wheelchair in the real world that would realistically control a virtual wheelchair within Second Life. The user, to navigate through the art piece designed within Second Life, would use this wheelchair. The wheelchair would represent a disadvantage that certain people in this world are born into, and limits their movement. Limiting the user in this way we felt could be interpreted by the user in many different ways, and made it more than a simple wheelchair simulator.

Setting forth from there we designed a method to read the speed and direction of each wheel, and send this data to a computer. On the computer we wrote code that would read the data from a serial port and send the pertinent information into Second Life. Code was then written within the game using the in-game scripting language to create realistic wheelchair movements.

A wheelchair was obtained, and a prototype of a mount was then built to hold this wheelchair. The mount kept the wheels from touching the ground, so they could move freely. Smaller contact wheels were installed onto the mount and attached to encoders which h read the speed and direction. This device was connected to a computer and our code was implemented to create a realistic moving wheelchair within the game.

The meaning behind the piece within Second Life is deep, and the wheelchair serves to add to the illusion and immersion of the piece. After demoing the unit ourselves, and with others we feel that we have developed an interesting new way to create art, and to convey a message. We also have come to the conclusion that our device could be further perfected in the future to increase functionality and to be used for other applications in addition to our original idea.

1. Introduction

As the years go on, technology is being used for more and more things. This technology has been put to use frequently to create alternate worlds for users to become immersed in. This has allowed adopters of technologies to go on adventures never imagined before, from being a professional athlete to battling monsters in a fantasy landscape. These virtual worlds have evolved greatly over the years and have begun to move beyond mere games to create forums used for the free exchange of ideas from users all over the world.

One such virtual world that allows users to interact with each other freely is the world of Second Life, an open source game developed and maintained by Linden Labs. This world allows its users, or residents, to create objects, and scripts that allow their creations to interact with other residents. Things such as clothes, cars, and houses were designed, created, and sold in this persistent world.

Potential was seen, and artists began to use this world's objects as medium to create their works. Some pieces brought awareness to social issues, others recreated masterpieces in three dimensions. Others still were original structures that the residents of the world came to behold.

It is through mediums like Second Life that virtual art has evolved and taken shape as a respected and growing field of artwork. This art can be used in many ways, and the team of E-Merging Realities has chosen Second Life as their medium.

This team has decided to develop new interactive technology to create a link between the real world and Second Life in a way that will be meaningful to the user. The technology will help put the user into a situation were they are able to navigate an area filled with images and objects designed to raise awareness about select global issues.

2. Methodology

2.1 Scope

The scope of this project is limited to the creation of a prototype link between Second Life and the real world. The final product produced by this project is meant to lay the groundwork for future advancements in our technology. The project team will be devoted to brainstorming the best way possible to develop this link and to implement a working demonstration of this link.

2.2 Goals

The goal of E-Merging Realities is to develop a prototype device that will interface with an artistic design created within the online game Second Life. Detailed objectives that will lead the accomplishment of the original goal follow:

- Create a prototype wheelchair mount that houses encoders
- Create software solutions to communicate effectively with Second Life
- Create a wheelchair object within Second Life that moves realistically
- Create objects and scripts in Second Life to be used in the art piece

2.3 Project Plan

The project team will consist of three student members and one faculty advisor member. The student members will be Keith Chester, Robert Martin, and William Price. The faculty advisor member will be Professor Joseph Farbrook.

Team meetings with the advisor will be held weekly during A, B and C term of the 2008-2009 school year. If necessary, an extension into D term 2008 will include advisor meetings at the advisors discretion.

Student member meetings will be called as needed, and will be the proper place for student collaboration on the project work. Secondary forms of communication will include: email, telephone calls, and informal sessions. Regular reports will be given on the status of each team member's area of development.

The work plan consists of dividing the development areas by the level of expertise from each member of the project team.

- Keith Chester
    - Encoder Selection/Installation
    - Microcontroller Software Development
- Robert Martin
    - Second Life Object Scripting
- William Price
    - Client Machine Software Development

    - Client Machine Hardware
    - Wheelchair Mount Hardware
- Professor Joseph Farbrook
    - Second Life Art Design

The student project team members will work the final paper on collaboratively. The entire team will do the final assembly of the hardware and software and each individual member as well as the team as a whole will perform all testing.

When the assembly is complete, the student team will demo the final product to the faculty advisor and submit their hardware for approval. The final date of acceptance is set on the WPI CDR deadline of April 24, 2008.

3. Software Development

3.1 Microcontroller Development

The need for sensors on the wheelchair stand quickly made it obvious that an embedded system on the wheelchair stand would be a necessity. The embedded system would react to the encoders and talk to the computer system running Second Life in some way.

Researching a number of projects by artists and other electronics hobbyists it became apparent that the communication would be best dealt with by the serial communication port so that two-way communications would be possible if needed.

When choosing the micro controller to use a popular choice was championed by a number of artists and enthusiasts across the Internet – Arduino. Arduino, named after an ancient Italian King, it is an open source micro controller development board designed for individuals without the expertise and training of an electrical engineer or computer programmer. It is popular with artists as it allows one to easily have a computer interact with the physical world via electronics

3.1.2 Arduino Hardware and Software Properties

The Arduino development board is based around an Atmel Amega AVR 16 bit micro controller and has a number of features. It has sixteen digital inputs and outputs, six analog interrupts, multiple power sources (3.3 and 5 Volts), and two interrupt capable pins and two pulse width modulation capable pins. The interrupt capable pins are key as they are the basis of the optical shaft encoders used within the project. The "serial port" that Arduino both is programmed and communicates over is also a very common USB to USB B wire, allowing us to easily port this from system to system.

Arduino was also an attractive option not just because of its capabilities but several other factors as well. The entire development package is under fifty dollars a unit, whereas the programming unit for an Atmel AVR micro controller averages at about sixty dollars alone, let alone the cost of the chips and power supplies. It is also open source hardware, allowing complete and extensive documentation to the hardware and the ability to modify and develop the hardware into whatever end

one may seek. The software IDE Arduino is also open source and compatible on every major operating system available.

Finally, Arduino is not programmed in Assembly or embedded C much like other micro controllers. It is instead written in Processing, a simplified programming language developed by the MIT Media Lab. Processing was originally designed for visual designers and those involved with the electronic arts that may not necessarily have a background in programming. The language allows one to easily access and deal with embedded hardware, especially sensor inputs and electronic outputs.

3.1.2 Working with Arduino

Once the board was chosen, familiarizing ourselves with the board was relatively easy. The amount of documentation for Arduino is significant due to its popularity in electronic hobbyists circle throughout the Internet, and the Arduino's website also contains useful tutorials on several aspects of the board.

After familiarizing ourselves through these tutorials working with the Arduino board was easy. Several sketches were made with a number of electronic components to ensure that we would have the prerequisite knowledge needed to program the encoders we would be using for the project.

3.1.3 The Arduino Program

The program written for the Arduino, contained within section 8.1.1, was designed as an interrupt based program. Interrupt programming is described more in depth in section 5.3.2. First, a number of variables and constants are declared.

The constants, the same as variables that are never changed, are mostly values so we can easily refer to the pin values that we are wiring to. The only other constant is the “ignore value” to deal with a difference in sensitivity, to be discussed later.

The setup( ) function is run immediately after these variables are initialized. This function is only ran once the board is powered for the first time and tells Arduino to prepare certain pins to perform as either digital inputs or outputs, attaches interrupts to certain pins, and opens the serial communication port open for communication at the designated speed, in this case a baud rate of 9600 bps.

Nothing loops in this function. If the encoders do not trip an interrupt, nothing at all happens. Instead of merely checking the encoders continuously the interrupt enabled hardware and software allows the encoders to instead notify the micro controller that they are updating. This runs a function that we attached to the interrupt pins within the setup( ) function. Since interrupts can be backed up within a system, we had to design this function to be quick, efficient, and as short as possible.

Each function that runs when triggered on an interrupt immediately increments a pin independent counter. It compares this counter to our “ignore constant” for that encoder. If it does not equal, it will simply end the function and wait for the next interrupt. If it does equal, we enter a new block of code. First we reset the counter for that interrupt and then read the “B” pin of the encoder (explained in section 5.3). Depending on the output of the “B” pin, the function will either send one of two letters, representative of the key that we are listening for in

Second Life. After this is performed, the function exits and again waits for the next interrupt. This function is short and quick because it was designed to be so to avoid issues with interrupts overloading the micro controller.

3.2 Client Machine Development

On the machine that would be running Second Life, some piece of software would be required that would take the data sent over the serial port and translate it somehow into something that could be read by Second Life. The original plan was to use a server to communicate with Second Life. This method was tested, however it had a delays based upon the connection to the server. These delays we felt were unacceptable for the uses we required this connection for.

It was decided that an application would need to read the serial port and somehow find a way to send data to Second Life without going through the Internet. Research was performed into the Second Life client, but no solution was discovered. It was decided that the best way to send input from our serial port into Second Life would be to emulate key-presses that would be read by the game client as if a user was pressing buttons on the keyboard.

The first programming language we attempted to use was Visual Basic. After some experimentation and research, we found the syntax was a little confusing and we chose to look into other options. The next language that was attempted was C#, another language that utilizes the Microsoft .NET framework. A class was found that

emulates the pressing of a keyboard key within the Windows libraries, and was incorporated into a serial reader program.

The final result was a Windows application that allowed the user to select which serial port they wish to read data from. The option to alter baud rate was also added into the program, though in the final product we did not allow the user to alter the baud rate. This software fires a data received event each time data appears on the selected COM port. The program then reads the string, which is one of four characters. Based upon what the string is, the program will then emulate the key-press of the corresponding key, which is then registered by Second Life as a normal input. Code snippets can be found in the appendix.

3.3 Second Life Development

The Second Life platform is interesting and provides a unique challenge from a developmental standpoint because it is a platform based on its user-generated content. However, in order to keep the system unified and maintainable it defines a set of methodologies that all such content is required to abide by. Since our project exists (partially) within the confines of this system, it must also adhere to its rules of development, notably, the Linden Scripting Language, or LSL.

3.3.1 Preliminary Development

It was important to grasp everything that LSL allowed us to do within the Second Life world in order to get an idea of what we could do and what we wanted to do with it. To attain a better understanding of this concept it is worth mentioning how scripts work. All development within Second Life is done by creating various

objects. These objects hold a number of user-made scripts, which begin executing as soon as the object is generated (also referred to as “rezzed”) in the virtual world. These scripts run continuously and can accept input such as player chat, physical interactions such as clicking and dragging, proximity to a specific item, etc., and then act upon these inputs by effecting items or players in the game world. LSL is a language comprised of a large set of pre-made functions to detect and perform specific actions such as those defined above within the context of the game and it is from this set of functions that all user-generated scripts are written.

The first stage of the Second Life software development was simply the ascertainment of all necessary information through testing a variety of LSL’s features and limitations. Some example scripts written during this phase include: Objects that can communicate with one another, a simple point and click ball game, and a controlled recursive object generation script. The final and most elaborate of these was a physics-based “hoverbot” object that follows any player who clicks on it, hovers (literally bobs up and down) over his/her right shoulder, and always faces the same direction as that person.

3.3.2 The Virtual Wheelchair: Design

At this point in time the project had been clearly defined and it was time to begin development of the actual wheelchair object in Second Life. This object had a specific purpose to act as a virtual counterpart to a real wheelchair and correspond to its input to make it move throughout the art installation that would later be devised.

In order for this object to be successful and create a meaningful link to the real wheelchair, several design considerations had to be met:

- Control: The wheelchair must be capable of handling inputs from the real chair and act upon those inputs correctly.
- Perspective: Everything must take place in a first-person perspective to give the user the feeling that they are in the virtual world.
- Level of Realism: In order to solidify the link between the real and the virtual, a certain level of realism in the motion and action of the chair had to be maintained.
- Robustness: Since the only interface between users and the virtual world is the physical wheelchair (limited control and mobility), the virtual chair has to be capable of handling potentially hazardous situations that users may not able to recover from on their own.

3.3.3 The Virtual Wheelchair: Implementation

To begin, a basic wheelchair-shaped object served as the base for the player and as the source point for the wheelchair script. Each user in Second Life takes the form of a digital avatar (formally called a "resident"), which serves as their in-game identity. This avatar can sit on the wheelchair, which starts up the script.

The first element to be hard-coded was the control configuration. LSL can take in user inputs from the keyboard, but as a limitation can only take in certain keys. We decided on using four key inputs for controls: W, A, S, and D which control clockwise and counter-clockwise wheel movements for both the left and right

wheels. The decision for this control schema was based upon the hardware implementation; specifically, the optical encoders. Each encoder sends out pulse signals for a specific segment of a rotation, which is converted by the microcontroller into a character representing the key press to be simulated (See Sections 3.1, 3.2, 4.3.2). This led to the next design decision: using the in-game physics engine as the basis for wheelchair movement.

LSL has several functions that allow for object movement; however, a drawback to these methods is that movement is delayed and choppy. In order to achieve a smooth movement the virtual wheelchair uses the in-game physics engine. This also works in tandem with the control implementation as each key press generated from the hardware system is translated into a physical impulse that acts upon the chair.

Another point worth mentioning is the fact that Second Life objects, while having the possibility to contain multiple components, are treated as a single entity and as such any physical force or impulse applied to the object is applied to the object's center of mass. In order to remain true to the realism aspect of our design objectives the wheelchair had to translate single wheel movements into both linear and rotational impulses that give the effect of two-wheeled movement.

Upon initial tests of this control system a new issue was raised. Since the physical wheelchair sits on a stand its wheels do not turn and handle in a similar fashion to a wheelchair on the ground, raw input data coming into Second Life had to be modified in order to attain the level of realism we desired. As it was, moving in a straight-line in the game was difficult due to the fact that each wheel on the

physical char moves a lot more freely when not in contact with the ground. To fix this, the input data was modified by applying extra impulses depending on how the chair was being manipulated in order to allow for a more user-friendly and believable control output.

Finally, the notion of robustness had to be dealt with. There are two situations in which the user could abuse the control system and create an unrecoverable situation. First, if the user tried to spin the wheel too fast (a feat that was far too easy to do since the physical wheelchair was not situated on the ground) it could fly forward at high speeds and send the virtual chair flying upon a collision. The second was that if the user turned too fast rotational impulses would spin the chair like a top uncontrollably. Both situations could result in the chair not only being in an unrealistic situation but also lead to it tipping over and rendering the user immobile. In response to this, two solutions were implemented.

The first problem was fixed almost by itself. The physical wheelchair supplies inputs as a variety of key presses, but the PC operating system that runs the Second Life platform automatically limits the number of key inputs that come in to a certain rate, resulting in a maximum speed based upon this value. However, relying solely on an OS configuration setting is not variable in design, and so additional parameters to control this speed were added into the script as well.

Solving the second problem was a larger concern. A tipped wheelchair would be impossible to recover from, and since the project was meant to be a stand-alone art installation if something went wrong there would be no one to fix it. The solution came in the form of an interesting self-righting mechanism that checks the local Z

vector of the chair and determines if it is tilted past a specific margin (Approx. 30 degrees). In such an event the chair will immediately use vector cross multiplication and use the result of the calculation to apply a rotational impulse that will tilt the chair back into an upright position. This works in all cases, even if the chair is upside-down. In action this mechanism produces a bobble effect that is somewhat undesired, although due to the velocity limit solutions and the design of the art installation this situation should be rare at best and is acceptable

4. Construction

4.1 Wheelchair

The project mandated the procurement of a wheelchair. We managed to find a comfortable and high quality wheel chair on EBay. It is an adult fitted chair to allow for most people to be able to sit in it comfortably and still be able to turn the wheels. Its weight is low allowing one to easily mount it into our designed stand, and its ability to fold up allows us to easily transport it.

The purpose of the wheel chair, artistically, was to put the observer in an uncomfortably helpless position. While the user is used to merely walking to where they want, the wheelchair lowers them into a disabled position that puts them into the frame of mind of limitation. The turning of the wheelchair's wheels would control the wheels of a Second Life wheel chair in which the user's avatar would be sitting in, giving them a sense of living vicariously through the avatar.

4.2 Wheelchair Mount

For our wheelchair to interact properly with the world of Second Life, we would need to develop a way to prevent the chair from moving while the wheels were in motion. The plan for this was to build a mount that would hold the chair above the ground, and also provide an option for mounting the microcontroller hardware.

4.2.1 Wheelchair Mount Objectives

The original plan was to develop a stand for the wheelchair that would serve multiple purposes. The stand objectives were to:

- Lift the chair high enough to prevent wheels from making contact with the ground.
- Provide a foundation to mount the sensor hardware
- Be stable, sturdy, and able to safely support anyone who uses the device
- Look professional and acceptable for display

4.2.2 Wheelchair Mount Prototype

In order to test the encoders and our initial design concepts, the development team developed a prototype stand. The stand built of wood and provided great insight into the design of the final wheelchair mount that was to be built. The initial design was functional, however was not deemed suitable for display in a museum. The advisor requested that a new version would be designed using the lessons learned in the prototype development.

4.2.3 Wheelchair Mount Final Design

After the prototype stand was tested, construction began on the final version. Materials were acquired from the Higgins Machine Shop and with the help of Machine Shop staff member Tom Coletta the construction began. An special type of aluminum was selected for the construction thanks to special hardware designed for it. The hardware allowed the final product to be adjusted to fit our wheelchair perfectly, and allow the possibility of adapting the stand for use with future wheelchairs in the case of damage or to expand.

The design consists of an aluminum base with four leg stands that the wheelchair frame rests on. Sensor hardware is installed in the front two legs of the stand, and the axle runs through spacers that are secured inside the aluminum.

The final version of the wheelchair was tested with people from weights varying from 145lb to 250lb and was found to be stable and safe in each case. The final design was deemed acceptable for display.

4.3 Sensor Selection and Methodology

The encoder selection was mandated by the requirements for the project itself. We desired the ability to read the rotation of a wheel chair accurately and translate its direction into an input for our computer. The only viable option is a quadrature optical shaft encoder.

4.3.1 Optical Shaft Encoder Background

Optical shaft encoders consist of discs with alternating black/white lines extending out from the center of the disc. This is placed on the shaft we wish to measure. The electronics of the sensor has a light sensor that can detect the changes between these lines. As the shaft we are measuring turns, so does the encoder disc. The rotation is recorded by the changing of the lines in front of the light sensor. The detected change causes the electronics to send an impulse to the micro controller. Once the micro controller receives this impulse, it causes a software interrupt.

4.3.2 Interrupt Programming

There are two types of programming in embedded systems. The first is polling. This has the micro controller check one sensor after another for a change in their state, one at a time. The obvious problem with this method is that, while the micro controller is fast, it can miss important updates of sensor information. The second, which optical shaft encoders are dependent upon, is event driven programming. Event driven program waits for events, in this case interrupts, to activate the necessary steps and update information based on sensor readings.

An interrupt will stop whatever the micro controller is doing and activate a function of our choice. When this function terminates, the regular program will resume. Hence, “interrupt”.

4.3.3 Quadrature Encoders

Our optical shaft encoder had to be quadrature. This means that there are either two tracks of alternating lines translated 90DEGREESSYMBOL or two light

sensors translated on a single track. This offset allows the encoder to read two different values. One light sensor is treated as an interrupt trigger. When this light sensor sees a darkened line, it checks the other light sensor to see if it sees a dark or a light line. Because it is in a 90 degree translated position, it will see one or the other based upon what direction the shaft is turning.

4.4 Client Machine Hardware

It was determined that if the E-Merging Realities hardware was to be displayed in a museum or similar institution that a computer would need to go along with the rest of the hardware. This would allow the entirety of the project to move from location to location without concerns about compatibility on different machines.

Parts were picked and then assembled by the project team to give a working custom PC to suit this purpose. The specs of the machine built follow:

- 3.00 GHz Intel Pentium 4 Processor
- 1GB Crucial Ballistix DDR2 RAM
- Gigabyte GA-P31-S3G Motherboard
- 80GB Western Digital SATA Hard Drive
- GeForce 8500GT 512MB Graphics Card
- 16x DVD-ROM Drive
- Cooler Master ATX Case
- 430W Power Supply

The following software was installed on the machine:

- Microsoft Windows XP Professional Edition
- Mozilla Firefox
- Microcontroller Driver Software
- USB-Serial Port Driver
- Second Life
- SecondLifeLink (Custom E-Merging Realities Software)

5. Developer Reactions

5.1 Keith Chester

I have always been first and foremost a technical person. While other children in kindergarten busied themselves with the fear or joy of coloring within the lines of a picture I instead busied myself with my LEGOs, making vivid creations that became more complex and far more technically sound as I got older. Art has never been my forte, but something about this project has spoken to me. Perhaps it's that I am beginning to grow older and become more aware of the world around me in this age of CNN, or the persistence of tragedy in our news.

This project has allowed me to express myself in ways that a technical project never would be able to. The chill I had down my spine rolling through our Second Life art piece was just as rewarding as the joy of seeing my creation work – but also simultaneously several times more harrowing and bone chilling. It says a lot

when your own work inspires within you the same feelings you sought for it to bring out in other peoples.

By the end of this project I have quite a bit more appreciation for modern art that is now involving the recent events of the day and trying to bring out either a little more awareness of what we are ever so fortunate to be *capable* of ignoring or protest the unspeakable horrors that we unfortunately still have in this world.

5.1 Robert Martin

This project provides a unique insight to the compilation of man and machine that is slowly becoming more prevalent in today's society. When I first heard the project proposal I was intrigued. As a (previously) frequent user of online virtual worlds and games it was interesting to take a step in a different direction and add a sense of reality to the virtual setting that I was so used to being almost purely fantastical. The thought of pushing the boundary always kept my mind a bit more open as I worked on the project.

Overall, I had a lot of fun messing around with Second Life's scripting language and creating really intricate scripts like the virtual wheelchair. Although fun the tasks were also challenging and ultimately very rewarding once all the project components were put together.

5.3 William Price

E-Merging Realities has taught me a lot over the course of the project work. I learned about a whole new world within the game community, the art community.

Through my collaboration with Professor Farbrook I gained a great deal of experience working with someone who is more artistically inclined than necessarily technology inclined. His experience was rewarding for me because the final product is something we can both be proud of. The technology is impressive and functional. It is used perfectly within the piece that Professor Farbrook designed.

Seeing our wheelchair implementation in action was a very rewarding experience. Many hours were spent by the entire team writing and testing code, building the mount in the machine shop, and running around in Second Life trying to figure out how to bring it all together. Moving the wheelchair through Professor Farbrook's piece was especially gratifying because I felt that intended effect. I felt restricted in my movements as I looked at many chilling and moving graphics.

Seeing the hard work of the student team, and the artistic vision of Professor Farbook come together was a great experience, and I would call E-Merging Realities a resounding success. I hope that we have laid the groundwork for future projects to step into this new world, and I hope our technology is used to its fullest extent and is able to touch people from all over the world.

6. Future Opportunities

6.1 Future Opportunities for Wheelchair Hardware

The wheelchair hardware we developed can be used for far more than a mere art piece. The hardware has several applications within the real world that can prove quite useful.

Because of the hardware's ease to interface with any computer system or operating system and its emulation of an everyday keyboard, it can very easily be adapted into nearly any program. This ease to integrate the hardware into other programs can prove quite useful and keep its spectrum of use widespread.

One particular application that often came up was the use of the wheelchair simulator as just that – a wheelchair simulator. It is often difficult to determine how a virtual model of a building or the design of an office or room will work for handicapped people. By allowing designers, whom they themselves are unaware of the struggles and difficulty of maneuvering for wheelchair bound people, to move through their virtual designs in the early stages of planning as if they were in a wheelchair we are making more handicapped friendly designs possible.

Another application is to outright raise awareness for the handicapped. By adding more hardware to our wheelchair simulator one can easily simulate how hard it is to do certain tasks when you have to also operate a manual wheelchair. It is difficult to imagine such hardships until you go through it yourself.

A third application is for outright game design. The wheelchair is a well-designed game controller that can allow a more immersive game experience if that game is based around wheelchairs.

6.2 Future of Second Life and Social Issues

Second Life's open nature, allowing the user to create, design, program, and distribute their works in any way they see fit, causes the game to be nothing less than a cauldron of creativity. The idea of a single creative piece instantly being distributed around the world for all to see with few restrictions will continue to attract new artists for years to come. There will be many art pieces in the future. The true power of Second Life, however, lies in its political capabilities.

Throughout the news Second Life is repeatedly coming up. United States Presidential candidates are setting up Second Life Campaign headquarters. Virtual protests over the world's many injustices happen regularly, allowing people to amass and express their opinion en masse with relative location no longer being an issue. Everyday that these events go on in Second Life it brings to light that the problems of Africa, or Germany, or China, or Iraq, or any location throughout the world is no longer a local problem but a global one where players from throughout the world can join together to help.

Second Life will not be going away anytime soon – rather, it is far more likely that you will begin to see more headlines generated by this world community giant.

7. Appendices

7.1 Code Snippets

Attached are snippets of code used in the different forms of software in our implementation. The programming languages follow:

- Microcontroller Code: Embedded C
- Client Machine Code: C# .NET
- Second Life Scripts: LSL (Linden Scripting Language)

7.1.1 Microcontroller Code

```
/* IQP Arduino Code
* Written by Keith Chester, Rob Martin, and William Price
*/

int leftEncoderInterrupt = 2;
int rightEncoderInterrupt = 3;
int leftEncoderB = 6;
int rightEncoderB = 7;
volatile int temp = 0;
int countl = 0;
int countr = 0;
int maxcount = 10;
void leftEncoder(){
 countl++;
 temp = digitalRead(leftEncoderB);
 if (countl > maxcount) {
  countl = 0;
  if(temp == HIGH) Serial.print("a");
  else Serial.print("w");
 }
}
void rightEncoder(){
 countr++;
 temp = digitalRead(rightEncoderB);
 if (countr > 4) {
  countr = 0;
  if(temp == HIGH) Serial.print("s");
  else Serial.print("d");
 }
}
void setup(){
 Serial.begin(9600);
 attachInterrupt(0, leftEncoder, RISING);
```

```
  attachInterrupt(1, rightEncoder, RISING);
  pinMode(leftEncoderB, INPUT);
  pinMode(rightEncoderB, INPUT);
  pinMode(leftEncoderInterrupt, INPUT);
  pinMode(rightEncoderInterrupt, INPUT);

}
void loop(){
}
```

7.1.2 SecondLifeLink Client Machine Code

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.IO.Ports;
using System.Diagnostics;
using System.Threading;



namespace SecondLifeSerialLink
{
   public partial class Form1 : Form
   {
      //This is the declaration for the Serial Port. Standards are no Parity
          and One Stop Bit
      private SerialPort port = new SerialPort("COM1", 9600, Parity.None,
          8, StopBits.One);

      //This is a process that will be used to open notepad, only exists for
          testing purposes.
      private Process p = new Process();

      //Also for testing purposes, this function is used in the SetText
          Function
      delegate void SetTextCallback(string text);

      public Form1()
      {
         //Calls on designer code to create Form
```

```
        InitializeComponent();
    }


    // When Set port is clicked, updates the serial port with the desired
information.
    private void btnSetPort_Click(object sender, EventArgs e)
    {
        //Set Port Name and Baud Rate
        port.PortName = PortNameBox.Text;
        port.BaudRate = 9600;

        //Update Information box on Form
        lblPortName.Text = port.PortName;
        lblBaudRate.Text = port.BaudRate.ToString();

    }

    //For threading purposes, single threaded application
    [STAThread]


    //When Start button is clicked, open the port and wait for data. Disable
        Start and Set port buttons.
    private void btnStart_Click(object sender, EventArgs e)
    {
        //Open the Serial port
        port.Open();

        //If the port is opened, disable Set and Start buttons, enable Stop.
            Update status.
        if (port.IsOpen)
        {
            btnStart.Enabled = false;
            btnSetPort.Enabled = false;
            btnStop.Enabled = true;
            lblStatus.Text = "Link is Enabled";
        }

        //Call Event Handler function when data is receieved on port.
        port.DataReceived += new
System.IO.Ports.SerialDataReceivedEventHandler(port_DataReceived);

    }
```

```
        //When Stop button is clicked, close port and re-enable buttons
            previously closed.
        private void btnStop_Click(object sender, EventArgs e)
        {
            //Close Port!
            port.Close();

            //If the port is closed, disable Stop button, enable Set and Start. Update Status
            if (port.IsOpen == false)
            {
                btnStart.Enabled = true;
                btnStop.Enabled = false;
                btnSetPort.Enabled = true;
                lblStatus.Text = "Link is Disabled";
            }
        }

        private void port_DataReceived(object sender,
System.IO.Ports.SerialDataReceivedEventArgs e)
        {
            //copy the data waiting on the port to a string
            string read = port.ReadExisting().ToString();

            //For testing purposes, sets (invisible) Data label to the value of the
             string
            SetText(read);

            //IF statements to check the data received, emulates keypress for
             appropriate key.
            if (read.Contains("w")) SendKeys.SendWait("w");
            if (read.Contains("a")) SendKeys.SendWait("a");
            if (read.Contains("s")) SendKeys.SendWait("s");
            if (read.Contains("d")) SendKeys.SendWait("d");
        }

        private void SetText(string text)
        {
            // InvokeRequired required compares the thread ID of the
            // calling thread to the thread ID of the creating thread.
            // If these threads are different, it returns true.

            if (this.lblData.InvokeRequired)
            {
                SetTextCallback d = new SetTextCallback(SetText);
                this.lblData.Invoke(d, new object[] { text });
```

```
        }
        else
        {
           this.lblData.Text = text;
        }
     }

  }
}
```

7.1.3 Second Life Code

```
//Script by Rob Martin, Keith Chester, and William Price
key user;
integer Ltic;
integer Rtic;
float WHEEL = .08;

default //empty chair - initial state
{
   on_rez(integer n) //always reset on rez
   {
      llResetScript();
   }

   state_entry()
   {
      //Chair is immovable in the empty state
      llSetSitText("Sit Here");
      llSetText("* Wheelchair *\nMOBILE - physics\nW, A, S, D\nmouselook\nno
timer\nTilt prevention\nsmooth controls", <1,1,1>, 1);
      llSetStatus(STATUS_PHYSICS, FALSE);
      llSay(0, "Empty");
   }
   changed(integer c) //checks if someone is sitting in the chair
   {
      if ((c==CHANGED_LINK)&&(llGetLinkNumber()==1))
      {
         //get key of the person sitting
         user = llGetLinkKey(llGetNumberOfPrims());
         state permissions;
      }
   }
}
state permissions //get permissions for controls
{
```

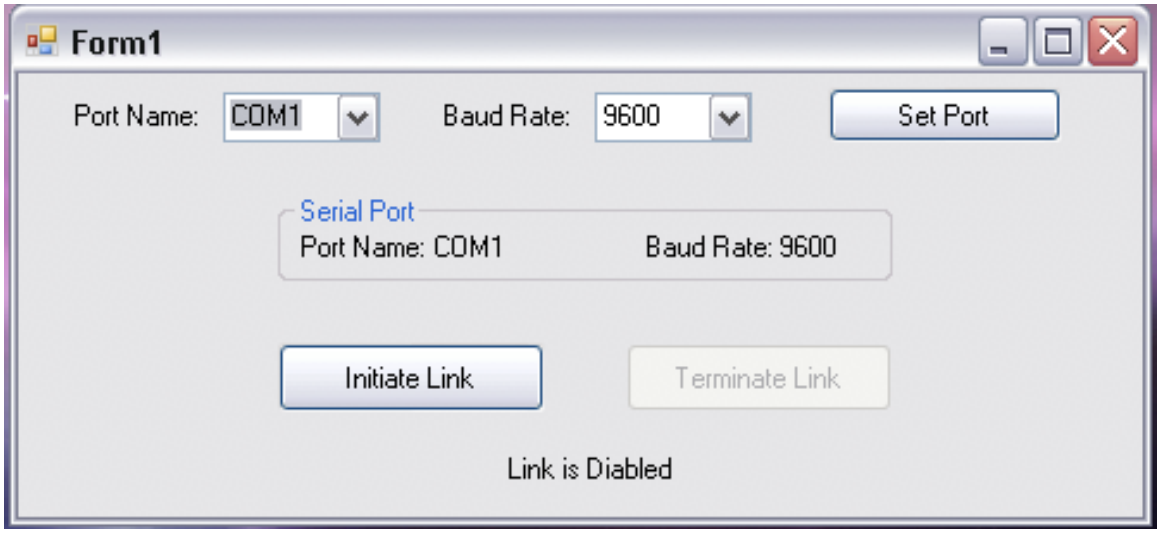
```
    on_rez(integer n) //always reset on rez, regardless of state
    {
        llResetScript();
    }

    state_entry()
    {
        integer perms = PERMISSION_TAKE_CONTROLS;
        if ((llGetPermissions()&perms)==perms) state on;

        else llRequestPermissions(user, perms); //make request
        if ((llGetPermissions()&perms)==perms) state on; //success?
        else //fail
        {
            llSay(0, "Permission request failed.");
            state default; //return to empty state
        }
    }

    //in case script hangs here, click to return to original empty state
    touch_start(integer total_number)
    {
        state default;
    }
}
state on
{
    on_rez(integer n) //always reset on rez, regardless of state
    {
        llResetScript();
    }

    state_entry()
    {
        llSetText("", <0,0,0>, 1);
        llSetStatus(STATUS_PHYSICS, TRUE); //enable physics
        llSitTarget(ZERO_VECTOR,ZERO_ROTATION); //clear sit target
        llSitTarget(<-.3,0,0>, <0,0,PI,0>); //set sitting position on chair
        //Displays the name of the current sitting avatar.
        llSay(0, llGetLinkName(llGetNumberOfPrims()) + " is now sitting.");
        //Force mouselook
        llForceMouselook(TRUE);
        //Configure user controls to move wheelchair
        integer controls =  CONTROL_FWD |
                            CONTROL_BACK |
                            CONTROL_LEFT |
```

```
                CONTROL_RIGHT;
    llTakeControls(controls, TRUE, FALSE);
    llSetTimerEvent(.001); //start timer

}

control(key id, integer held, integer change)
{

    //Wheelchair controls to be set up as follows:
    //Left Wheel Forwards: W key or UP arrow
    //Left Wheel Backwards: A key or LEFT arrow
    //Right Wheel Forwards: S key or DOWN arrow
    //Right Wheel Backwards: D key or RIGHT arrow
    if (((~held)&change&CONTROL_FWD) || //forward key release (W)
        (held&(~change)&CONTROL_FWD) ||
        (held&change&CONTROL_FWD))
    {
        Ltic = 1; //increment Ltic
        llApplyImpulse(7*WHEEL*llGetMass()*<-1,0,0>, TRUE);
        llApplyRotationalImpulse(WHEEL*llGetMass()*<0,0,1>*-1, TRUE);
        //llWhisper(0, "L-F: W");
    }
    if (((~held)&change&CONTROL_LEFT) || //left key release (A)
            (held&(~change)&CONTROL_LEFT) ||
            (held&change&CONTROL_LEFT))
    {
        Ltic = -1; //decrement Ltic
        llApplyImpulse(7*WHEEL*llGetMass()*<-1,0,0>*-1, TRUE);
        llApplyRotationalImpulse(WHEEL*llGetMass()*<0,0,1>, TRUE);
        //llWhisper(0, "L-B: A");
    }
    if (((~held)&change&CONTROL_BACK) || //backward key release (S)
            (held&(~change)&CONTROL_BACK) ||
            (held&change&CONTROL_BACK))
    {
        Rtic = 1; //increment Rtic
        llApplyImpulse(7*WHEEL*llGetMass()*<-1,0,0>, TRUE);
        llApplyRotationalImpulse(WHEEL*llGetMass()*<0,0,1>, TRUE);
        //llWhisper(0, "R-F: S");
    }
    if (((~held)&change&CONTROL_RIGHT) || //right key release (D)
            (held&(~change)&CONTROL_RIGHT) ||
            (held&change&CONTROL_RIGHT))
    {
        Rtic = -1; //decrement Rtic
```

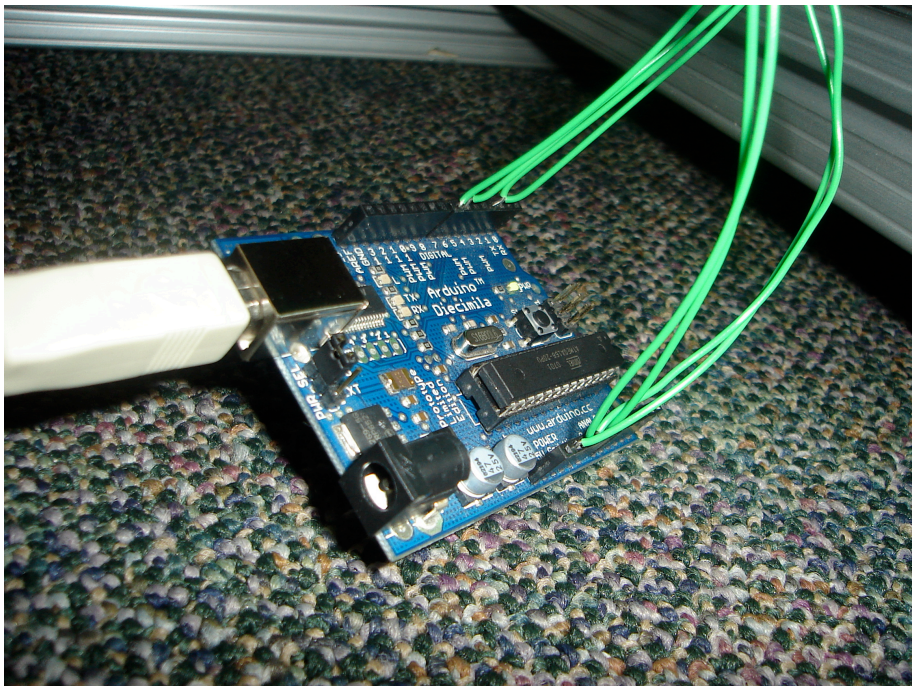
```
            llApplyImpulse(7*WHEEL*llGetMass()*<-1,0,0>*-1, TRUE);
            llApplyRotationalImpulse(WHEEL*llGetMass()*<0,0,1>*-1, TRUE);
            //llWhisper(0, "R-B: D");
        }

    }

    timer()
    {
        //The wheelchair is prone to unruly inputs which could potentially
        //tip over the chair, rendering it immobile.
        //Must check for significant tilt and re-position if necessary.
        vector up = <0,0,1>*llGetRot();
        //up is a vector with magnitude 1 and direction pointing above the chair
        if (up.z < .7) { //~30 degree tolerance
            //re-position
            llApplyRotationalImpulse(.5*llGetMass()*(up%<0,0,1>), FALSE);
        }
        else
        {
            //These lines are to assist in the physics of a wheelchair
            //Simply acting on the raw input from the physical chair returns
            //slightly different results since the chair is not actually moving
            //on the ground and is therefore not affected by surface friction.
            //This helps even the scale and make the in-game device more
            //straightforward and user-friendly.
            llApplyImpulse(-.07*llGetMass()*llGetVel(), FALSE);
            llApplyRotationalImpulse(-.01*llGetMass()*llGetOmega(), FALSE);
            if (Ltic == Rtic) {
                llApplyRotationalImpulse(-.15*llGetMass()*llGetOmega(), FALSE);
            } else {
                llApplyImpulse(-.15*llGetMass()*llGetVel(), FALSE);
            }
        }
    }

    changed(integer c)
    {
        if (c==CHANGED_LINK) //user stands up
        {
            llReleaseControls();
            state default; //return to empty state
        }
    }
}
```

8.2 Photographs

7.2.1 Software Screens

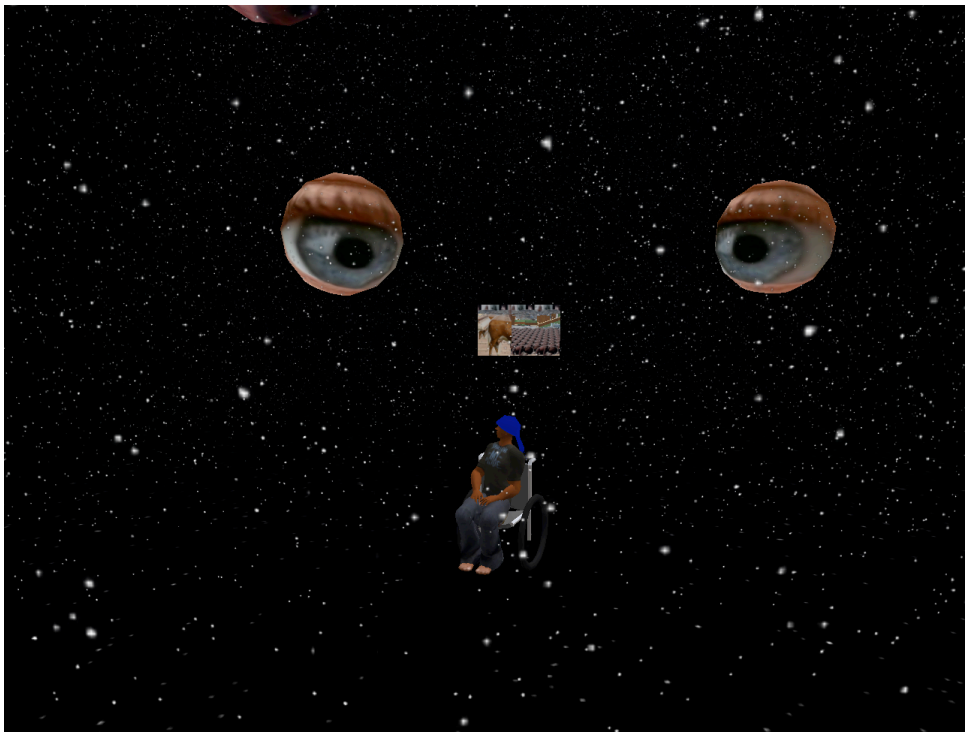


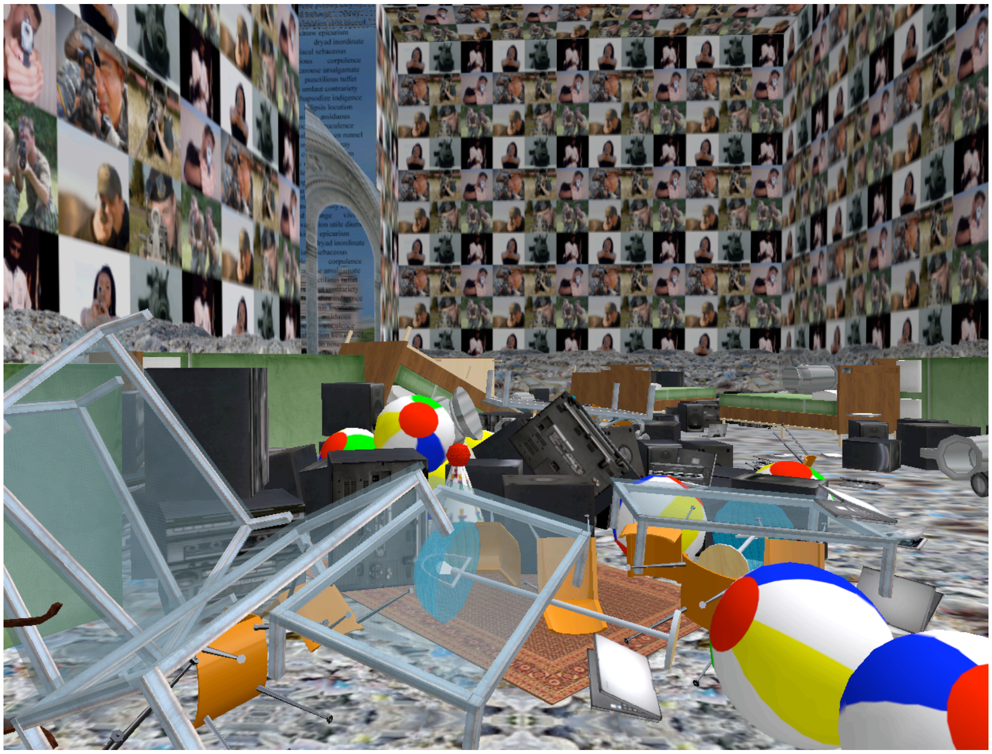
7.2.2 Wheelchair Hardware


Arduino
Diecimila
www.arduino.cc
POWER


racer SX5

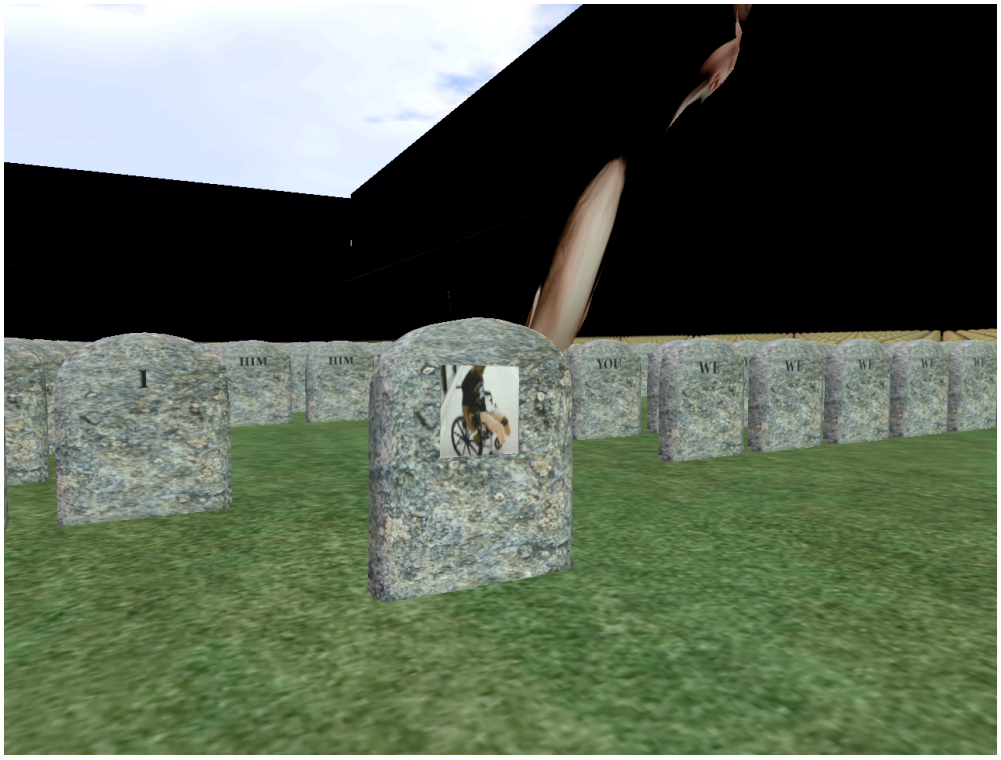
7.3.3 Second Life Screenshots






sedulity
doublet dyspeptic partible
anhydrous
rhinal dearth
philatelist harangue mesencephalon
malversation
amatory
camelopard
resplendent hermetic temporize antitussive
beamish disparate laconic
asepsis
attenuate
ebullition
maladroit
tarmacadam
epochal
lubricious
promontory paucity grist libation
interlocutor
oakum alfresco
myope autolysis
matchlock napery
ermine
coccoid coccoid muckrake ampoule
cloying
austerity raiment jape
mountebank
firmament
piquant escarpment
voltaic galvanic
pilosity
pinnate
blithesome quince
diadem
panada ermine
mountebank malefic
conflux
efflux
brindled


FRED
HAMMOND


I
HIM
HIM
YOU
WE
WE
WE
WE
WE