RYERSON UNIVERSITY

Faculty of Engineering, Architecture and Science

Department of Electrical and Computer Engineering

Course Number	ELE700 / COE700
Course Title	Engineering Design
Semester/Year	F2012

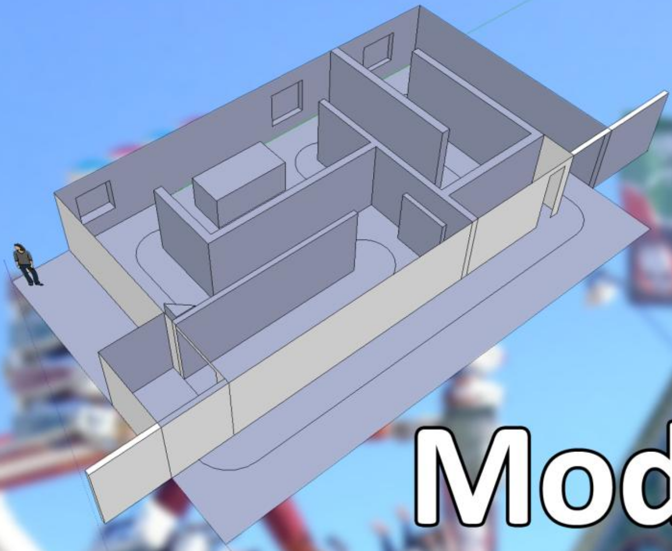
Faculty Lab Coordinator	Dr. Kaamran Raahemifar

Assignment Title	Mid Project Final Report

Submission Date	December 17, 2012
Due Date	December 17, 2012

Student Name	Rob Kipping	Imtiaz Miah	Mirko Miljevic
Student ID	500147810	500128765	500291018
Signature*			

Table of Contents



Carnival Dark Ride Modernization

An engineering study on improving a long-time amusement park mainstay using digital projection, gesture recognition, and augmented reality.

Mirko Miljevic
500291018

Rob Kipping
500147810

Imtiaz Miah
500128765

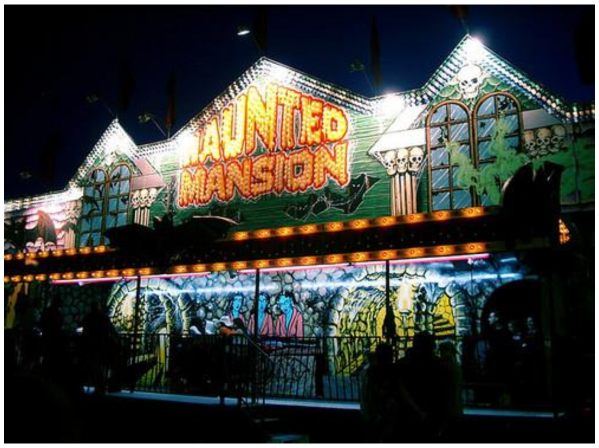


Figure 1 - The "Haunted Mansion" ride as viewed at the Canadian National Exhibition in Summer 2012. This ride serves as the subject of this project (also pictured on the title page).

Abstract

The carnival dark ride is a long-standing tradition of the travelling amusement park industry. Once considered the flagship ride of the carnival experience, the rides which are manufactured by a number of different entertainment suppliers and go by several different names (i.e. The "Haunted Mansion" in our case), have slowly lost their appeal due to their dated designs. Patrons of these rides (who will be referred to as "riders" in this document) can expect to be seated in a small two- or four-seated electric-powered ride vehicle on a double-rail track that travels through an enclosed, minimally lit room (hence the term "dark ride") that is composed of an expanded semi-truck trailer. As these rides almost always have a horror-related theme, several animated props known as "tricks" are located at different points throughout the ride, and automatically lit up and move when riders approach in an attempt to solicit a fright. However, this reaction is not that common with today's riders.

Despite the tricks present in these dark rides often being made up to resemble skeletons, rotting corpses, vampires, zombies, and other ghoulish characters seen in classic horror movies, riders will often quickly become bored and lose interest. In extreme cases, riders will leave the car mid-ride to vandalize the tricks, causing need for expensive and time-consuming repairs, and risking a high possibility of being run over by another car or being electrocuted. The focus of this study is therefore to improve the quality of the ride experience and encourage riders to remain seated to protect themselves and the ride.

But what is the best course of action to achieve such goals? We look to existing successful entertainment experiences: theme park rides and home video games, both of which offer many advanced technologies that successfully enthrall their users. The following sections describe our methodology, approach, application, and results of these techniques to the carnival dark ride experience.

Key Words

Carnival, entertainment, gesture recognition, video game engine, theme park, motion tracking, augmented reality, Arduino, Microsoft Kinect, Unity3D, digital projection.

Introduction

"Carnival Dark Ride Modernization" refers to the new technology that will be applied to the existing dark ride. While the current ride only gives riders a passive experience, the completed modernization will not only enhance the visual tricks, but introduce interactivity with them as well.

Description of the "Haunted Mansion"

To better convey why the ride requires modernization, the following diagram illustrates the interior of the current dark ride, and a breakdown of what the rider experiences.

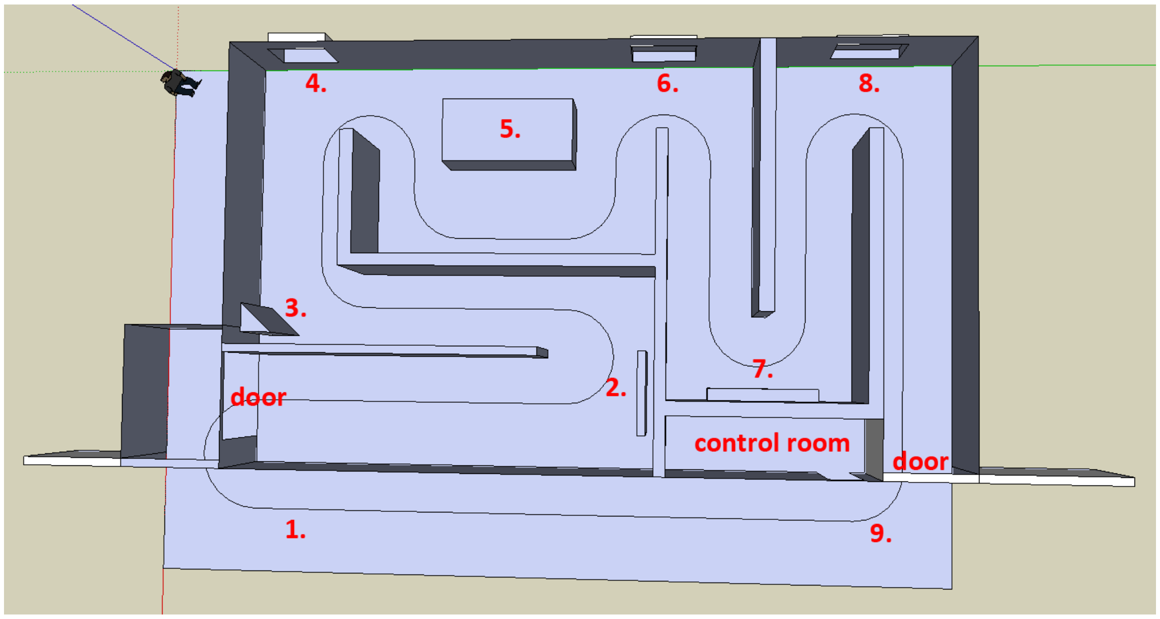


Figure 2 – Layout of the current dark ride, the "Haunted Mansion". The winding line through the ride represents the path the cars take from point 1 to 9.

The ride begins at point 1, where four riders have entered a car, and the operator of the ride as activated the section of track the car is sitting on (known as the "zone"), sending electricity from the track to the car's motor, causing it to move forward along the track. At this point, the car will travel along the track at a uniform speed and not stop until the ride is done. As the car moves around the corner, a large door swings open, and the riders enter the dark portion of the ride.

As the car straightens out, riders see the first trick at point 2; a life-size skeleton that rocks back and forth on an electric chair while appearing to being executed. A strobe light flashes on and off to simulate the electrocution. After the full 180° turn, riders should experience the second trick at point 3, which is a mummy in what appears to be an iron maiden. However, when our group rode this ride in person, the trick activated too late, well after the car passed it. Most likely the late activation was due to a misplaced motion sensor, which are responsible for activating tricks based on the condition of an oncoming car. The next trick at point 4 was a window box that featured a badly modeled dismembered torso lit up by a white strobe light. Next, the trick at point 5 was a large cage with steel bars that features a skeleton attempting to break free by shaking the bars (as shown below in Figure 3).

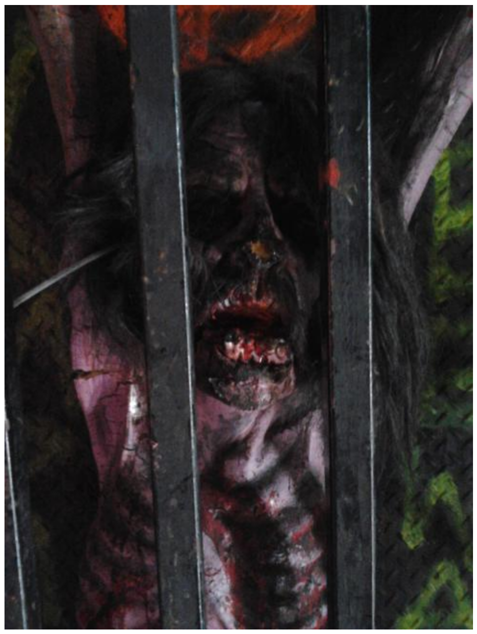


Figure 3 - A goulish trick on the "Haunted Mansion" dark ride.
Photo credit: Imtiaz Miah

Point 6 featured another window box trick of a skeleton. The trick at point 7 was a large steel coffin meant to open and expose a vampire, but the coffin failed to open and lay in repose. The final trick at point 8 was another window box containing the upper half of a dummy holding a flamethrower. The flamethrower lit up with a red light bulb and emitted a very loud burst of air. After the exit door opened, the car exited the dark ride, and riders disembarked the car, preparing it for the next group of riders.

Course of Action

The proposed course of action is to retrofit this ride with new existing technology while engaging riders to stay in their seats. The group in particular is interested in using the popular Microsoft Kinect device along with projectors which will project a customized environment for the riders to interact.

The Kinect has two purposes. One is to engage and entertain the riders which will serve the purpose of making the ride more entertaining. Another is to ensure that riders are not doing anything they're not supposed to such as getting off the ride. Rider's will be rewarded for good behaviour and engaging in the ride, much like a video game, and be punished for inappropriate actions.

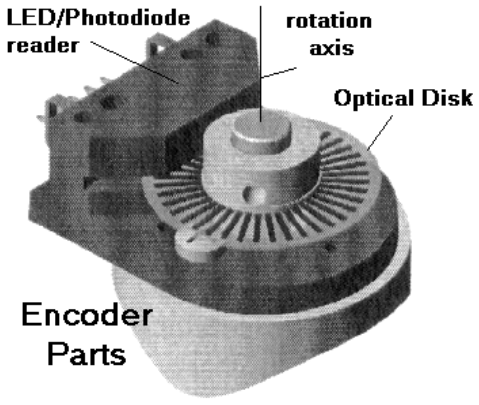
The custom environment will contain elements that will directly feedback with the position of the ride and use forced perspective techniques. The environment will be created in unity3D engine. This enables the students to interact with data from the kinect and the position of the car. More details are mentioned in the theory.

Purpose of Position Tracking

Position Tracking will be needed to understand where, along the track, the patron is. When the patron arrives at the location of the simulation, the simulated environment will be able to adjust to the view of the patron, based on their position. This gives it a more realistic feel to the ride (instead of a stationary image) and thus provides more immersion. The method used to achieve position tracking that will be looked upon in this lab report is using an encoder.

Encoder

There are many different types of encoders but a few that were looked at for this project. The main one being is a rotary (optical) encoder. This encoder counts the number of rotations of the wheel and calculates the position based on the circumference of that wheel. In more technical terms a rotary encoder "is an electro-mechanical device that converts the angular position or motion of a shaft or axle to an analog or digital code." [1] There were many different techniques researched in how to implement a rotary encoder. One way, and the most obvious way, is to use an optical disk (such as the one in figure 4). An optical disk is a transparent disk that has black slits a LED and a Photodiode (a photo detector that turns light into current or voltage). This one of the most commonly used technology and the way it works will be explained in the theory portion of this lab.



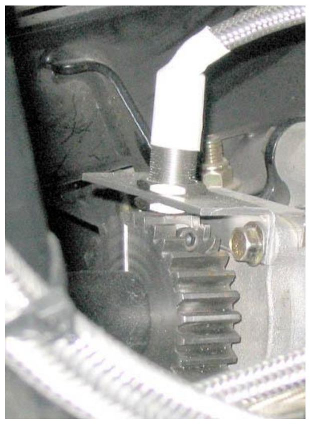


Figure4 – Optical Disk Encoder **Figure 5** – Magnet Encoder Sensing Gear Teeth

Another type of rotary encoder is show in figure 5. The magnetic encoder uses strips of magnetized material placed on the rotating disc and gets sensed by a Hall-effect sensor or magneto-resistive sensor. They can be used to sense gear teeth without a separate encoder disc, such as the one in figure 5. The third type of rotary encoder uses conductive tracks. It is a rare type of encoder and can be used in digital multimeters. It consists of a series of copper pads etched into a printed circuit board where contact brushes sense the conductive area. The final method, and the one we will be using on this project, is similar to the optical disk method. Instead of an optical disk, a makeshift disk with holes in it will be used and an infrared sensor will be placed on top of it. Since all of these methods use the same concept the theory portion will expand on how to use an encoder.

Purpose of Kinect

The Kinect's purpose is to take in rider data in video form. The camera will be fixated on top of the projector screen which will have a limited time and range to take in rider data. As mentioned earlier, the kinect serves as both a security measure and as a tool to make the ride more intuitive; both of which require gesture recognition algorithms. For the ride itself, the group has proposed 3 types of gestures to be recognized within this ride:

- To recognize when one of the patrons leave the ride
- Simple ducking gesture to avoid an oncoming obstacle
- Simple hand pointing or waving gesture
- Facial reconstruction on the Unity3D environment or audio inputs from the user (bonus)

This may not seem like a lot but this is merely to be a proof of concept for recommendations on how to retrofit these rides. Gesture recognition is not an easy task either as most techniques use advanced machine learning algorithms which can use math too advanced for the students.

Inappropriate Behaviour

To test inappropriate behaviour, our group is going with a marker-based approach on the seats. It would work in such a way that the seat has a customized marker on its seat. If the rider leaves the seat at any point in the ride, the kinect will detect the marker and the ride will have to stop immediately. In a sense, the markers can be thought of as a flag and will be detected using simple image recognition techniques. Principle Component Analysis (SVMs) along with Support Vector Machines (SVMs) would probably be more than sufficient enough to for detecting this flag.

Another inappropriate action to consider is if the rider decides to stand up during the ride. In this case, the rider is performing an unsafe act, yet they are still blocking the marker which becomes an issue. This will be discussed further in the gesture recognition part in the proceeding section.

Gesture Recognition

The first step is to take snapshots of the riders when they get on the ride. Key features to be extracted once the riders are settled in are height and facial features. This part will have sufficient lighting as it is the beginning of the ride so the feature extraction will be of high quality in comparison to what we would have obtained from within the dark ride itself. The reason height is taken into account is because it will be easier to detect when there is a sudden change in the rider's height when a rider attempts to stand up or duck. As the Kinect is set up on top the projector screen, it will have to be calibrated accordingly to account for the vehicle coming closer and closer to projected screen.

For recognizing a ducking gesture, the program simply has to know when the rider's head is down. To avoid confusion from the rider standing up, the program has to keep track of the head movements and height. For this part, we wouldn't really need to use anything too advanced of a technique. A simple boundary check would work for this part along with the mean shift method to keep track of the head movements.

For hand gestures and other advanced techniques, we can use the more advanced Hidden Markov Model (HMM) machine learning algorithm as this is the most commonly used one. A library has to be trained into these motions, which will be compiled by the students during production of this project. If the gestures match the library, points will be awarded to the rider. In the theory section, the algorithms will be described in more detail.

Survey

Ride Duration

After timing five cars entering and exiting the ride, the duration between point 1 and point 9, representing the entire ride experience, was found to be an average of 50 seconds. Interestingly though, according to the “Haunted Mansion” manufacturer’s website[1], the specified ride duration is supposed to be 2 minutes. Whether this time includes loading and unloading the cars, or if the operators of the specific ride we observed modified the ride to be shorter, is up to speculation (however, we timed an average unloading to loading time as 20 seconds, which still puts the ride duration under the manufacturer’s specified duration suggesting the ride vehicles are moving twice as fast as they should be moving).

Rider Reactions

After the ride, 11 groups of riders were interviewed. The following table represents the feedback given:

Description of Subject(s)	"What did you think of the ride?"	"Improvements?"
family of four	Not scary, tricks can be seen before being activated, rid was the same twenty years ago.	Have tricks come down on you from the ceiling.
adult couple approx. 40 years old	Safe ride. Fun.	Needs an actual ending. 3D stuff?
four kids approx. 10 years old	Not scary. (Note: one girl was on her cellphone the entire ride)	Needs air conditioning, more people (tricks).
young boy approx. 7 years old	"Lame". Tricks are seen ahead of time (not dark enough).	3D. Less phony stuff.
young boy approx. 5 years old	Not sure, had eyes closed entire ride.	Not sure.
young boy approx. 10 years old	"Scary!!"	Not sure.
adult couple approx. 25 years old	Not scary.	Not sure.
one boy, one girl (siblings) approx. ten years old	Both kids loved the ride and rushed back into line after the interview.	Shooting at things, earning points, winning prizes.
woman approximately 30 years old	Good.	Temperature was alright.
adult couple approx. 25 years old	More entertaining than scared.	Needs more big tricks, such as the skeleton in the cage.
boy and girl approx. 15 years old	More entertaining than scary.	Needs real people chasing you around.

Table 1 - Rider Reactions and Suggestions

Literature Review

Research for interactive digital image projection for entertainment purposes began with exploring higher quality dark rides found at theme parks. During a recent trip to both Universal Studios and Walt Disney World in Orlando, Florida, notes were taken on best practices used in theme park dark rides to engage riders and maximize enjoyment during the course of a ride.

Digital Image Projection and Forced Perspective

One ride that proved to be the most relevant to our research was “The Amazing Adventures of Spiderman” at the Islands of Adventure theme park at Universal Studios, Florida (as pictured in Figure 6).

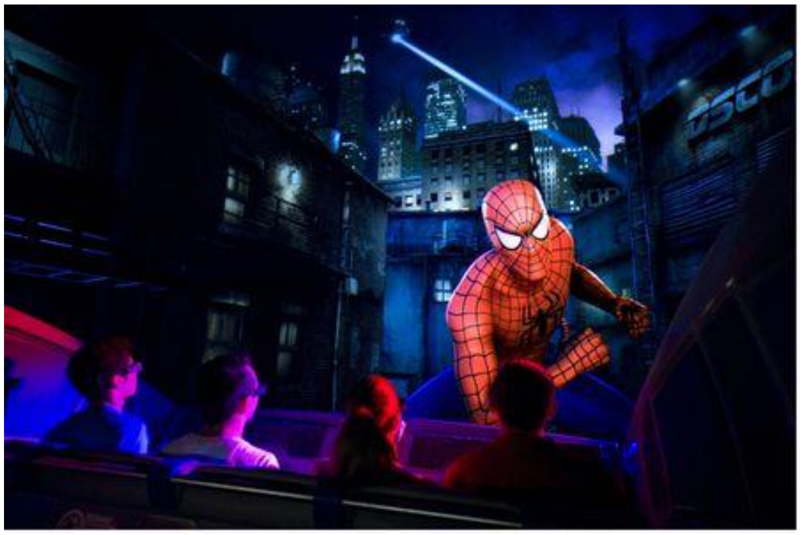


Figure 6 - Riders experiencing the theme park dark ride “The Amazing Adventures of Spiderman” [2]

Similar to our goal for the dark ride, “The Amazing Adventures of Spiderman” uses multi-passenger ride vehicles that move along a fixed track and features fixed digital projections screens on walls. Most interestingly though, is that the video projected on to walls during the ride provide an augmented reality effect similar to the forced perspective effect that we wish to achieve. This effect is explained in a behind the scenes video found on YouTube[3] and serves to be highly useful in our designs.

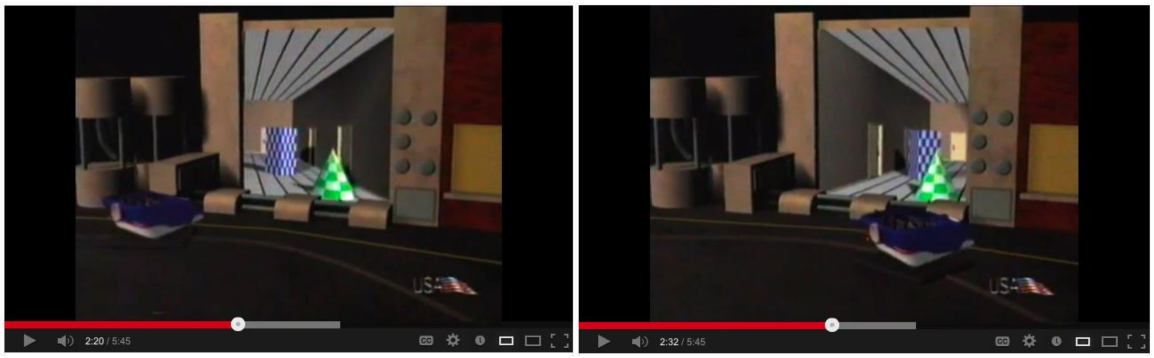


Figure 7 - Screenshots from a behind the scenes video explaining an augmented reality effect known as “squinching” utilized on the ride “The Amazing Adventures of Spiderman” at Universal Studios, Florida.

In a behind the scenes video demonstrating the design of the Spiderman ride, the augmented reality effect is referred to as “squinching” (as shown in Figure 6). The narrator of the video describes squinching as “determining what that distortion is going to be, and counteracting it with additional distortion on the screen.”

3D Software

After consulting Dr. Matthew Kyan of Ryerson University regarding the scope of our project, and our requirements for digital video generation and interactivity, the software chosen that would best fit those requirements was Unity3D.

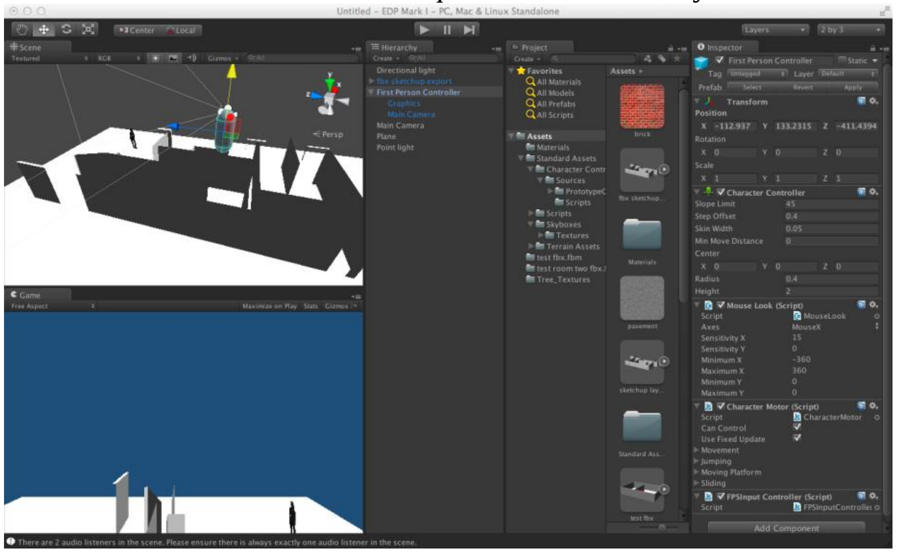


Figure 8 – Screenshot of Unity3D running a simulation of the “Haunted Mansion”

Unity3D is a video game engine designed specifically to be easy to use, and highly adaptable for various different video game development requirements. Unity3D features integration into Microsoft’s .NET development platform as well as scripting with C# or Javascript, which will allow for inputting external information from the Kinect gesture recognition software and the position tracking hardware, and process that input into scripted activities like skewing virtual rooms for augmented reality and moving characters or creating text for feedback from recognized gestures

Theory

Position Tracking

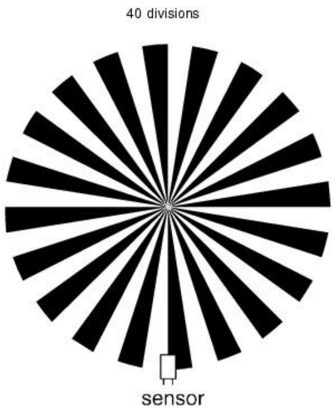


Figure 9 – Optical

Step 1: Reading Values - The optical encoder will be used to describe the theory even though the same concept can be applied to other types of encoders. A representation of an encoder with labels is show in figure 4 and we can see that when it is attached to the shaft of the wheel, as the shaft on an encoder spins so does the optical disk. A sensor reads a '1' or a '0', based on the declaration in the code, for every time the LED light reaches the photodiode. If there are 40 divisions (figure 9) then every 20th count of '1' would mean the distance travelled is that of the circumference of the wheel.

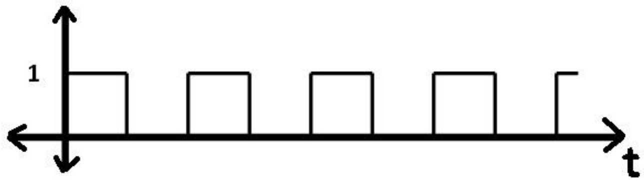


Figure 10 – One Sensor Output for Constant

A more accurate way to look at this would be to calibrate the wheel and to assign a number, n, for every time the sensor reads a switch between high and low. For a wheel with 40 divisions, when n=40, the distance travelled, d, will equal the circumference, c.

Example #1 - For example say we have an optical disk with 32 divisions (n=32), and a circumference of 20cm (c = 20cm) would yield a travel distance of;

$$d = \frac{c}{n} = \frac{20\ cm}{32} = 0.625 cm/n$$

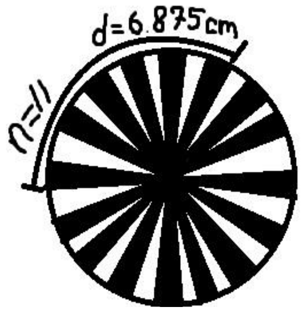


Figure 11 – Example #1: Optical Disk

For every count of n, there will be an added distance travelled of 0.625cm. Thus, if 11 divisions where read, that would mean a travel distance of 11 x 0.625cm = 6.875cm.

Step 2: Direction Distinguishment – So far we have covered how to read an optical disk and calculate a distance. However, one sensor only works for a forward direction. What if we wanted to move both forwards and backwards? For that we will need two sensors, be it side by side using the same disk or using two sensors with two separate disks it is still the same. The main idea behind this technique is to have two sensors with a slight offset such as the ones in the simulation **figure 12** and in **figure 13**.

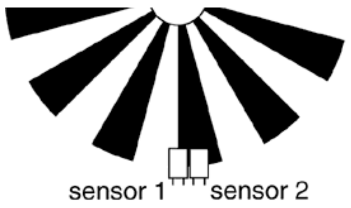
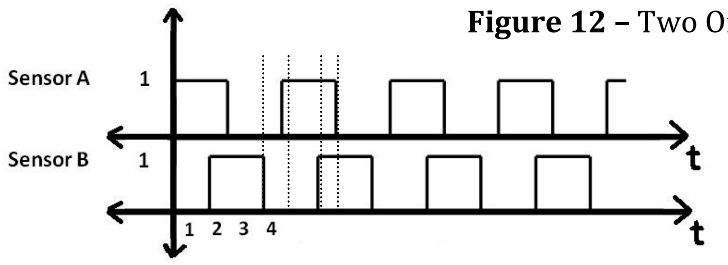


Figure 12 – Two Offset Sensors on



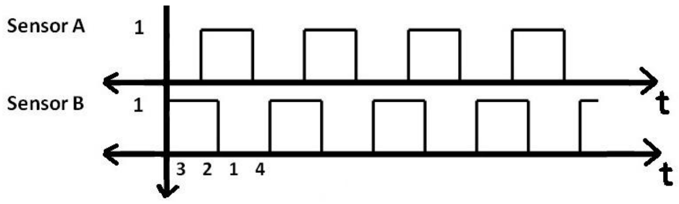


Figure 13a – Two Sensor Clockwise Pattern

Figure 13b– Two Sensor Counter-

Assigned Value	Binary Output
1	10
2	11
3	01
4	00

Table 3 – Binary

With the first sensor (sensor A or 1) slightly ahead of the second sensor then there will be a binary output of '10' as sensor 1 is ON and sensor 2 is OFF. We assign that output a value of **1**. When the two sensors are both ON (or HIGH) then the microprocessor receives a binary output of '11' and we assign it a value of **2**. Once sensor 1 turns OFF again and sensor 2 is still ON there is a binary output of '01', which we assign a value of **3**. Lastly the final stage is when they are both OFF and with a binary output of '00' we assign the value **4** to it. Therefore when the wheel is going forwards we will be counting in ascending order (2,3,4,1,2,3,4,1…etc.) . When the wheel starts going backwards we will begin to count in descending order (1,4,3,2,1,4,3….etc.).
To keep the measurements simple, we will only assign values of n (number of divisions) to one disk. When the wheel is going forwards we ADD the distance d for every n (**example #1**). On the other hand, when the wheel is spinning backwards we SUBTRACT the distance d for every count n.

Digital Image Projection

The digital image generation portion of this project is the generation of two-dimensional video that is projected onto the large video screens in the final presentable portion of the project, replacing the tricks in the old dark ride, and providing visual feedback to the riders' interaction. This portion also serves as the bridge between the rider position tracking and gesture recognition portions, and can be considered the integrating factor of this design groups' efforts.

The requirements for this step were to generate video for 2D video projection that is "squinched" to the riders position to provide an augmented reality effect, and a method for inputting location information of the riders and sensing triggered gesture recognitions and relaying feedback to the riders through the projections.

Squinching (Augmented Reality on a Projection Screen via Distortion)

In the effect in the video, we see a ride vehicle moving slowly on a track lateral to a large video screen on a wall. In order to achieve the video effect, two movements in the video are required.

The first movement is the point of view into the virtual room. The riders' view, which can be referred to as the "camera", moves in the same direction as the ride vehicle, but always faces the far wall in the virtual room. This motion is rotational (as shown below).

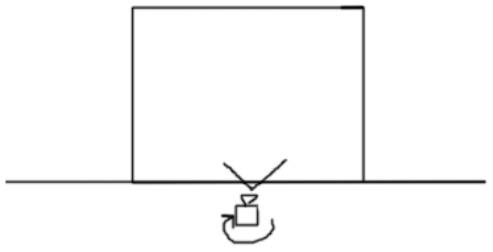


Figure 14 - A virtual camera rotates slightly inside a virtual room as viewed from above the ride.

The second movement required for the effect is fully distorting the virtual room to make up for distortion caused by the riders seeing the projection screen at a non-perpendicular angle (as shown below).

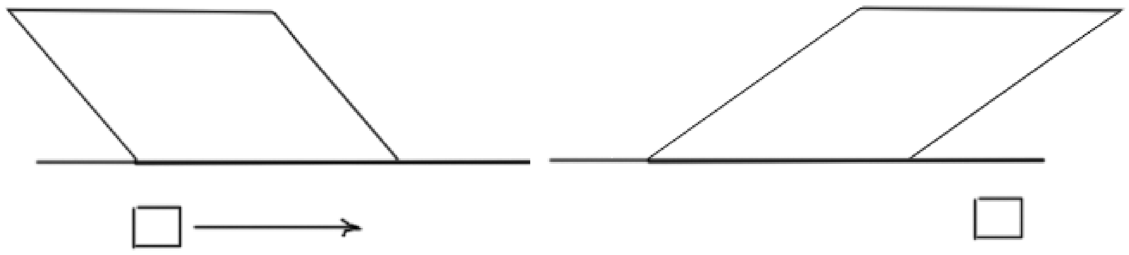


Figure 15 - (as viewed from above) As the ride vehicle moves from left to right, the room is distorted (skewed) to compensate for riders' view distortion as viewed from above the ride.

As can be determined from the behind the scenes video of the Spiderman ride, the boundaries of the virtual room which exists within a projection must remain true to the rider as they would if riders were looking into an actual room, i.e. the two corners of the virtual room must remain locked to the front corners of the projection screen. The only way to achieve this, while still moving the back wall of the ride, is to physically distort the entire virtual room by skewing the back wall in the same direction as the riders, while leaving the front wall locked to the projection screen boundaries.

As this effect relies completely on the position of the riders relative to the screen, the position-tracking portion of this project becomes

Interactivity Model for Pilot Project

The interactive component of the ride refers to the ability for the ride to provide instant visual feedback to the riders via the projection screens on whether their gestures have been recognized by the Kinect camera. The easiest way for these systems to relay information to Unity3D is via simulated keyboard presses for when gestures are recognized, and simulated mouse movement to represent rider position tracking.

As Unity3D features prefabricated first person shooter-style controllers, minimal modifications would have to be made to adapt the systems to work together.

Gesture Recognition Algorithms

Principle Component Analysis

Principle component analysis works through taking samples called "Eigenvalues". These samples can be thought of as the training set to be averaged which will be subtracted from each frame one by one in the sample. Let us call this set A. Once this set is obtained, we take the eigenvectors by using the Covariance matrix of A. As this would imply a very large number, we use a linear algebra property to take the largest eigenvectors in order to use a smaller matrix. This can all be summarized in the following equations.

1) $\Phi = \text{Img} - \mu$
2) $A = [\Phi_1, \Phi_2, \Phi_3, \ldots \Phi_n]$ - *These are the eigenvalues*
3) $\text{Cov} = AA^T$ - *In order to calculate the eigenvectors, we take the covariance which gives a large matrix*
4) $u_i = Av_i$ - *In order to combat the large matrix, we simply take the largest eigenvalues from this smaller matrix which is all we need.*

For the image matching part itself, the steps above are repeated for the frame in question where the mean difference is taken from the frame to be tested. It is then projected onto the eigenvector and each of the normalized values we got from the samples are compared with the normalized frame with the minimum euclidean distance.

Mean Shift Method

The mean shift method will be used for head tracking. This is because of it is actually a relatively simple algorithm that continuously goes towards the direction of maximum increase in density. A common technique in image tracking is to use colour histograms and follow that continuously. For our use, this would be ideal, as we don't expect the head to go off camera unless the rider leaves the ride. We also expect a gradual movement of the head which makes it ideal because as mentioned above, the classifier continuously moves towards the direction of maximum increase in density. The diagram below demonstrates how this works:

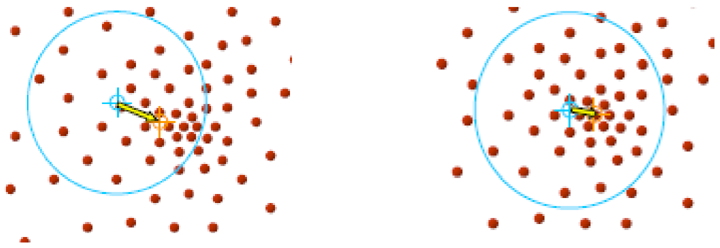


Figure 16 - Mean Shift Method - The circular area represents the set cluster size which moves towards higher densities

The equation for the mean shift method is:

$$m(x) = \frac{\sum_{x_i \in N(x)} K(x_i - x) x_i}{\sum_{x_i \in N(x)} K(x_i - x)}$$

Where *x* represents the initial estimate (iterative attempts) and the *K(x_i -x)* function represents the weight of adjacent points to iterate again until m(x) converges.

Hidden Markov Model

The hidden Markov model is a widely used machine-learning algorithm that is based on training samples. It is based on the idea of "hidden states" which can be considered parts of the models we are after. In other words, the hidden states can be considered parts of the library that we build for the gestures we want to be recognized. Each state is hidden when going into as an input but can be "seen" when going as in output. When going to the next state, it goes through a probability distribution on which state is it likely to go over most. There are many other factors in here and without going into too much detail of how it works, it can be summarized on the trellis diagram below:

$$\begin{tikzcd}
{} \arrow[r] & x(t-1) \arrow[r] \arrow[d] & x(t) \arrow[r] \arrow[d] & x(t+1) \arrow[r] \arrow[d] & {} \\
 & y(t-1) & y(t) & y(t+1) &
\end{tikzcd}$$

Figure 17 - Hidden Markov Model Trellis Diagram

We can think of the x components as hidden or latent variables where they represent all the gestures in our library of our interest. The y components can be thought of as the "observed" state, which is actually random variables, but they represent the "input" data or gesture that we are trying to decode.

Pilot Project

Since all simulations performed in this pilot project are software based, results are listed with the simulation processes. For the position tracking, the IR sensor is tested and data is taken in. For the Unity3D engine, a user input flag is tested. And finally, for the kinect, a simple captured video is tested to see when the rider leaves the ride. Also, some gestures and tracking are tested with source code that is already available which is being studied by the students.

Position Tracking

The sketchup simulation is a 3D mockup of the setup for the position tracking scheme. In the image below (**figure 10**) it shows the placement of the microcontroller and the IR sensors with the two offset disks. In the link provided (http://youtu.be/LpzlcBDOBSo) there is a youtube video narration of the sketchup model.

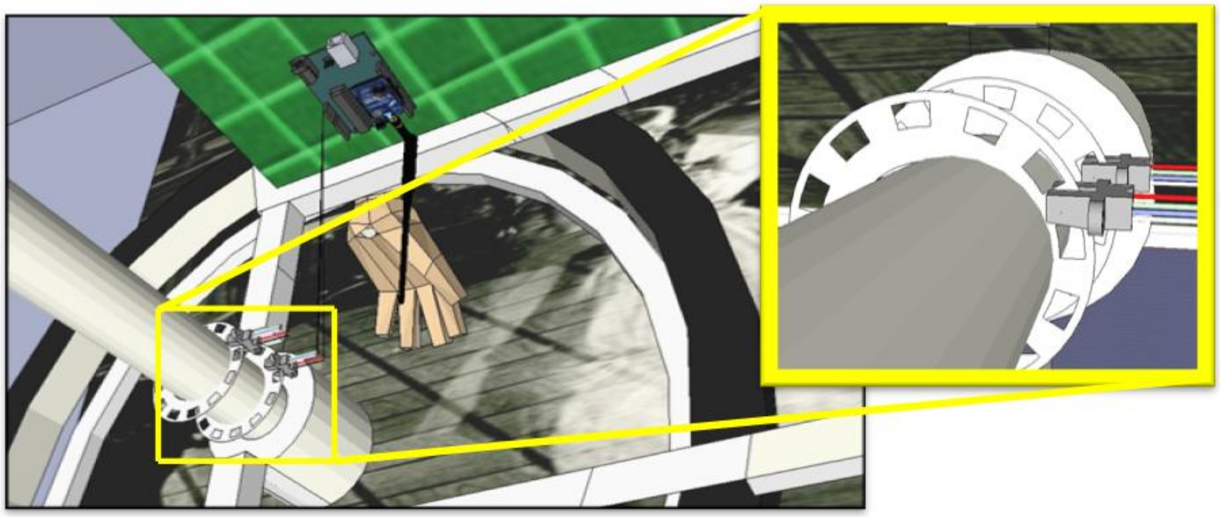
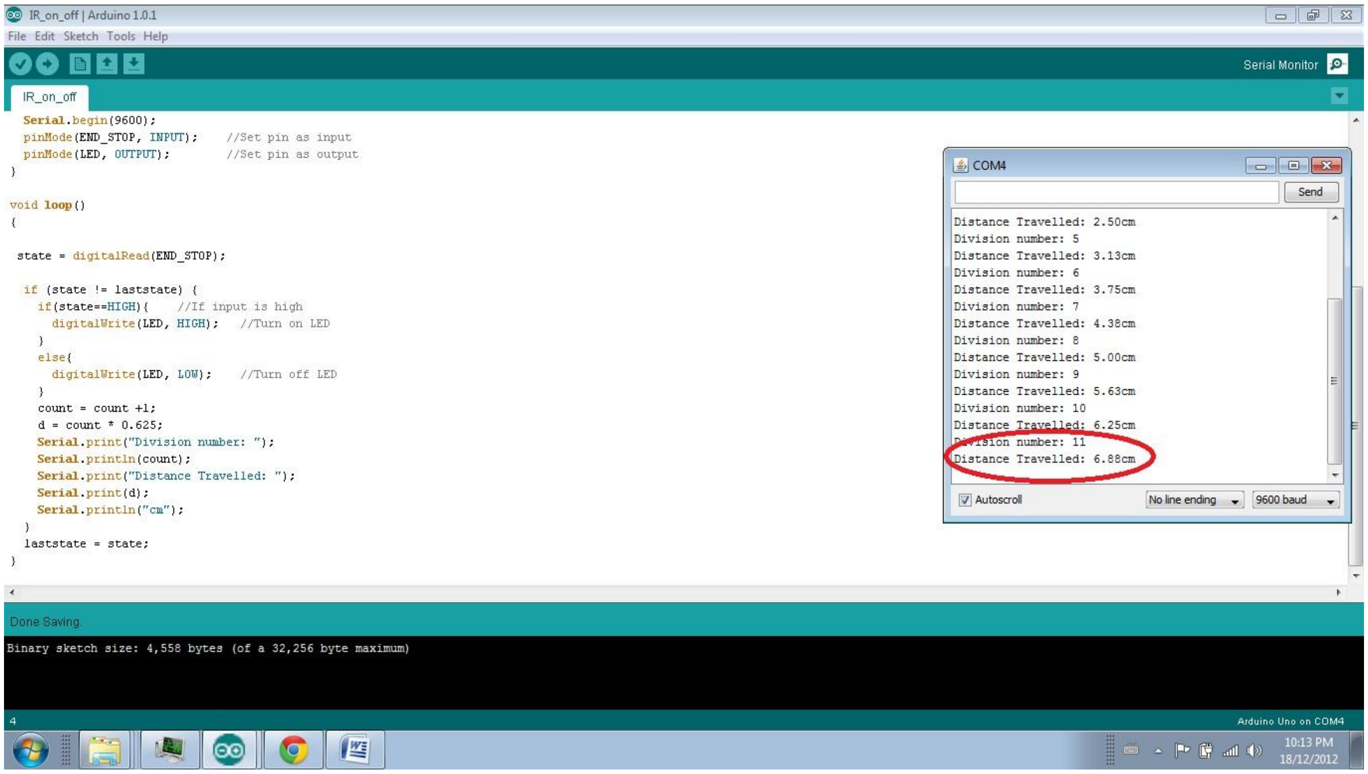


Figure 18 - Sketchup Model of Wheelchair Tracking

The code show in the simulation section calculates the distance travelled for a wheel moving forwards in a straight line. Using **Example #1** as a reference, if the wheel has a circumference, c, of 20 cm and had n = 32 divisions. Then each division would be a distance d = 0.625cm/n. Thus in **Figure XXX** below we see for 11 divisions the distance travelled was 6.88cm as was in the example, and in **Figure YYY** the distance travelled for 32 divisions was 20cm which is the circumference of the wheel. To see the full code, please refer to the appendix

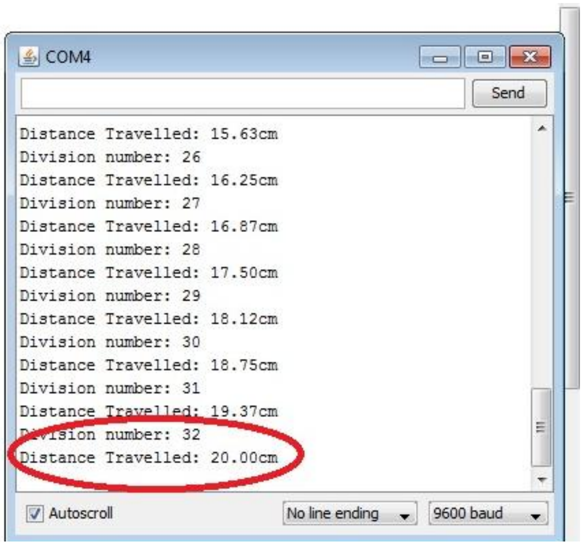


```
  Serial.begin(9600);
  pinMode(END_STOP, INPUT);    //Set pin as input
  pinMode(LED, OUTPUT);        //Set pin as output
}

void loop()
{

 state = digitalRead(END_STOP);

  if (state != laststate) {
    if(state==HIGH){    //If input is high
      digitalWrite(LED, HIGH);   //Turn on LED
    }
    else{
      digitalWrite(LED, LOW);    //Turn off LED
    }
    count = count +1;
    d = count * 0.625;
    Serial.print("Division number: ");
    Serial.println(count);
    Serial.print("Distance Travelled: ");
    Serial.print(d);
    Serial.println("cm");
  }
  laststate = state;
}
```

```
Done Saving.
Binary sketch size: 4,558 bytes (of a 32,256 byte maximum)
```

```
Distance Travelled: 2.50cm
Division number: 5
Distance Travelled: 3.13cm
Division number: 6
Distance Travelled: 3.75cm
Division number: 7
Distance Travelled: 4.38cm
Division number: 8
Distance Travelled: 5.00cm
Division number: 9
Distance Travelled: 5.63cm
Division number: 10
Distance Travelled: 6.25cm
Division number: 11
Distance Travelled: 6.88cm
```

Figure 19 – Arduino IDE: Calculation of Distance for 11 Divisions



```
Distance Travelled: 15.63cm
Division number: 26
Distance Travelled: 16.25cm
Division number: 27
Distance Travelled: 16.87cm
Division number: 28
Distance Travelled: 17.50cm
Division number: 29
Distance Travelled: 18.12cm
Division number: 30
Distance Travelled: 18.75cm
Division number: 31
Distance Travelled: 19.37cm
Division number: 32
Distance Travelled: 20.00cm
```

Figure 20 – Arduino IDE: Calculation of Distance for 32 Divisions (Circumference)

Unity3D Interactivity

Interactivity in with a 3D environment in Unity3D is demonstrated below. As the user approaches the blue sphere, pressing the “E” key on the keyboard changes the colour of the sphere from blue to red. As mentioned above, the "E" key can be thought of as a trigger or flag to represents a number of things. In the final version of the project, interactivity will be more complex, such as a character on the projection screens running away, or a ghoulish figure popping up to scare the riders.

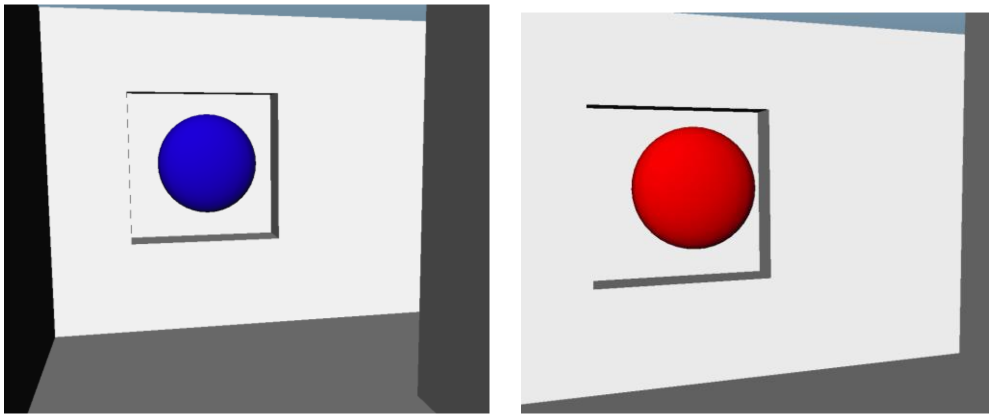


Figure 21 - A simple demonstration of a Unity3D script that detects input from the gesture recognition system.

The main function of interest can be summarized in a javascript code found in the appendix.

Inappropriate Behaviour Detection

For this part, a video feed was captured and saved with the kinect. It is a 9 second video (278 frames) of the student initially sitting in front of the camera and eventually leaving the seat in the 6 second mark (around frame 190). The objective of this part of the pilot project is to show that the program can detect when the rider leaves the seat.

The student took the following steps for collecting. First a reference frame was needed to compare to when the seat is empty. This can be thought of as the marker frame which is constantly checked on whether the rider left the seat. Throughout the video feed, the program constantly compares the current frame with the reference frame. If a certain threshold is met, it is detected as a match and the flag sets off that the rider has left the seat. The figures below show the program working.

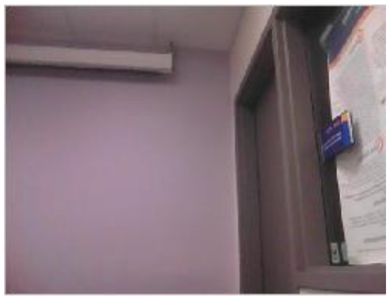


Figure 22 – Reference Frame

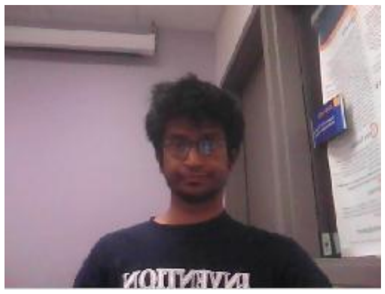


Figure 23 - Live Capture Frame (RIDER IN SEAT)

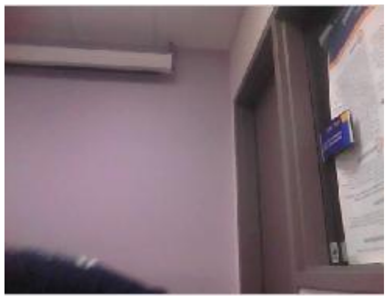


Figure 24 - Live Capture Frame (RIDER LEFT SEAT)

As we can see, when rider left the seat, you can still see part of the student's shirt (frame 185). The threshold value can be modified a bit to make it more accurate.

Though this is only testing using simple histogram comparisons, PCA needs to be considered for a number of reasons. One being that the ride is constantly moving so the reference frame also needs to have sort of a library. A threshold has to be determined when the seat is empty and not empty in this case. Also, the student used a grayscale histogram, more accurate results can be obtained by using colour channels.

Testing More Advanced Recognition

Skeletal reconstruction, depth stream, and facial detection were tested successfully with available sample code off the internet. They worked surprisingly well in dark settings and will be suitable for our use in the dark ride. The codes tested used many algorithms, including HMM, but some were also out of the students understanding at this moment. However, as part of the goal for the pilot project, it is proven to work and can be implemented in the dark ride.

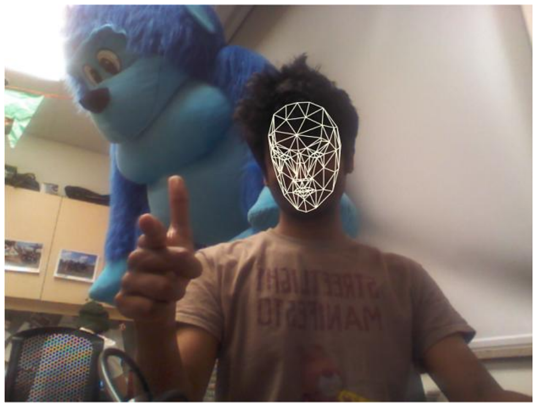


Figure 26 - Face Tracking using the "Face Tracking Basics - WPF" Source

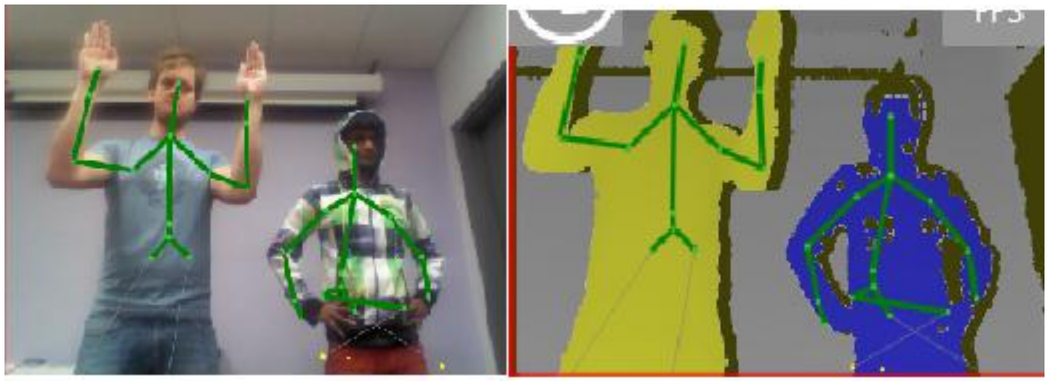


Figure 27 - Skeletal and Depth Image Stream using the "Kinect Explorer"

Conclusion

Through the course of our simulations, we have successfully completed the initial steps towards finishing our design goal for the dark ride modernization:

- Successfully recognize a gesture using the Kinect camera
- Successfully interpret wheel rotation on model chair with an Arduino
- Successfully model an interactive scene in Unity3D

Although, it is clear that there is much more work to be done. For the next phase, we will combine all elements into the Unity3D engine and will require an actual wheel ride vehicle (which will most likely be composed of a wheelchair) complete with position-tracking sensors. Actual projection screens with a mounted Kinect camera will be required for presenting a virtual room through “squinching”. One of the milestones to look forward to is combining everything in the Unity3D engine although this pilot project showed the individual goals that are gradually leading to that step. The completed demonstrable project will be similar to the mockup in **Figure 23**.

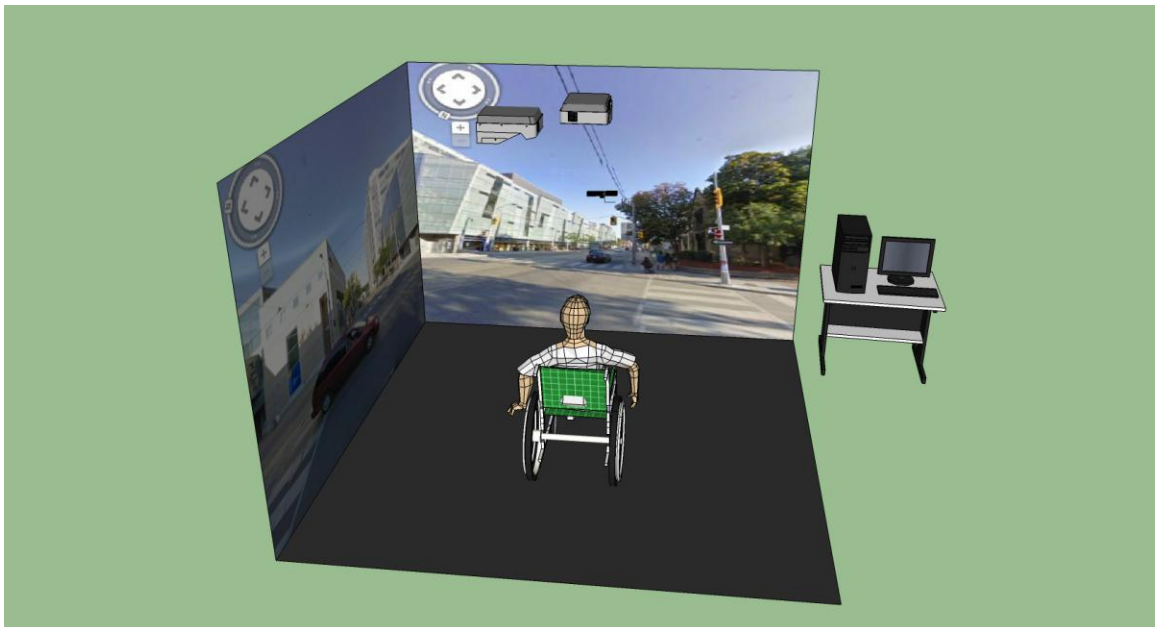


Figure 23 – Final Presentable Project

References

1 – Owens Trailers, accessed December 17th, 2012.
http://www.owentrailers.com/dark_ride.shtml

2 - Theme Park Insider, Accessed December 17th, 2012.
http://www.themeparkinsider.com/reviews/universal's_islands_of_adventure/the_amazing_adventures_of_spider-man/

3 – Youtube, Accessed December 17th, 2012.
http://www.youtube.com/watch?v=9rJqA4ivyQw

4 - Youtube, Accessed December 17th, 2012. http://youtu.be/LpzlcBDOBSo

Appendix

Arduino Code

The following is the beginning steps taken to learning the Arduino IDE along with the Infrared Sensors and how it all works. The code below is Test #2 and it turns the LED on when the IR signal is connected and back off when the signal is interrupted. It also counts how many times it switches from HIGH to LOW, meaning a change in division. Looking in the results section will show you an example.

```
int END_STOP = 2;          // IR Signal pin
int LED = 11;              //Pin LED is attached to
int state = 0;
int laststate = 0;
int count = 0;
float d = 0;

void setup()
{
  Serial.begin(9600);
  pinMode(END_STOP, INPUT);   //Set pin as input
  pinMode(LED, OUTPUT);       //Set pin as output
}

void loop()
{
 state = digitalRead(END_STOP);

  if (state != laststate) {
    if(state==HIGH){    //If input is high
      digitalWrite(LED, HIGH);   //Turn on LED
    }
    else{
      digitalWrite(LED, LOW);    //Turn off LED
    }
    count = count +1;
    d = count * 0.625;
    Serial.print("Division number: ");
    Serial.println(count);
    Serial.print("Distance Travelled: ");
    Serial.print(d);
    Serial.println("cm");
  }
  laststate = state;
}
```

Unity3D Code

```
var sphere : Rigidbody; // Expose the variable "sphere" to Unity, and assign the sphere
pre-fab in the scene to this


function Update() {

if( Input.GetButtonDown( "Use" ) ) { // The "Use" key in Unity is specified to the
keyboard key "E"
sphere.texture.color = Color.Red; // change the property of the model "sphere" to "Red"
} }
```

MATLAB Inappropriate Behaviour Detection

```
vid = mmreader('Capture.avi');
RefFrame = imread('Snapshot2.jpg'); %template of empty seat

i=1;
figure;imshow(read(vid,1)); % displays first frame
disp('RIDER IN SEAT')
while i<vid.NumberOfFrames % loops though total number of frames

    frame = read(vid, i);

    CurrentFrame = frame;

    % the normalized histogram takes in the histogram data of the image
    % and divides by number of elements in the image array
    hist1 = imhist(CurrentFrame(:))./numel(CurrentFrame);
    hist2 = imhist(RefFrame(:))./numel(RefFrame);

    % this compared both histograms
    Compare_h1h2 = sum(abs(hist1 - hist2));

    if Compare_h1h2<=0.2 %this sets the threshold value
        disp('RIDER LEFT SEAT')
        break
    end
    i=i+1;
end
figure;imshow(frame); %displays frame which program believes rider left
seat
figure;imshow(RefFrame); %template empty seat display
```

Receipt for Kinect Camera

Greetings from Amazon.ca!

We thought you'd like to know we shipped your items, and that this completes your order.

Thanks for shopping at Amazon.ca, and we hope to see you again soon!

You can track the status of this order, and all of your orders, online by visiting the Your Account page athttp://www.amazon.ca/your-account/.
There, you can:
 * track order and shipment status.
 * review estimated delivery dates.
 * cancel unshipped items.
 * return items, and do much more!

The following items were included in this shipment:

Qty	Item	Price	Shipped	Subtotal
1	X360 Kinect sensor for Window	CDN$ 229.99	1	CDN$ 229.99

Sold by Amazon.ca, Inc - taxed

Item Subtotal: CDN$ 229.99
Shipping and handling: CDN$ 0.00
Total Tax: CDN$ 29.90
GST: CDN$ 29.90
PST: CDN$ 0.00
Total: CDN$ 259.89
Paid by Visa: CDN$ 259.89

This shipment was sent to:

Rob Kipping
140 Carlton Street
Apartment PH4
Toronto, ON M5A 3W7
Canada

via UPS.

For your reference, the number you can use to track your package is
1Z6W941E2032584280. You can refer to our Web site's Help page
or: http://www.amazon.ca/wheres-my-stuff/ to retrieve current tracking information.
Please note that tracking information might not be available immediately.

If you've explored the links on the Your Account page, but still need to get in touch with us about your order, you can find an e-mail form in our Help department
at http://www.amazon.ca/help/.

Please note: This e-mail was sent from a notification-only address that cannot accept incoming e-mail. Please do not reply to this message.

Thank you for shopping with us!

Amazon.ca
http://www.amazon.ca/