
Interactive
Show Enhancement
For Carnival Dark Ride
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

Interactive Show Enhancement for Carnival Dark Ride

Mirko Miljevic
Rob Kipping
Imtiaz Miah

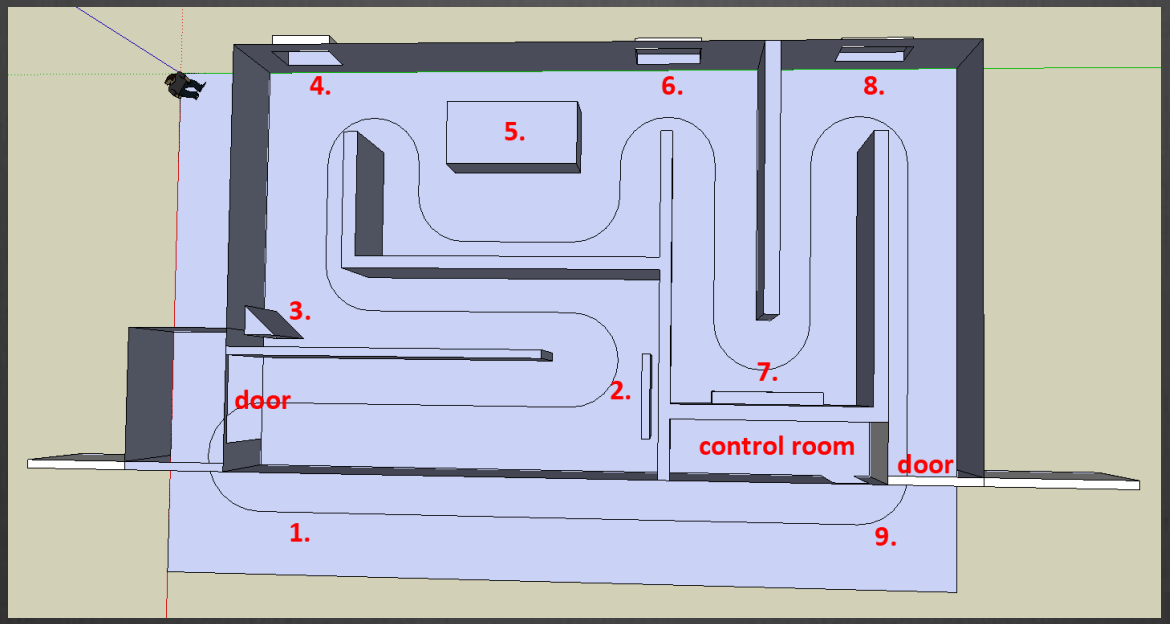
FLC: Dr. Kamran Raahemifar
Advisor: Dr. Kathryn Woodcock

Introduction

- "Haunted Mansion" is a type of travelling carnival dark ride that tours around North America with various carnival companies.
- The current Haunted Mansion ride is aging badly. Riders, in general, do not enjoy the experience as the ride is aging and deteriorating, and does not compare to the technologically advanced nature of entertainment today.
- The quality of ride and safety of riders is compromised as some riders exit cars during ride to damage ride elements, and risk being run over and severely injured by other ride vehicles.
- This EDP topic was suggested by Dr. Kathryn Woodcock of the THRILL Lab (human factors, ergonomics, and safety in amusement), and has potential marketability.

The “Haunted Mansion” carnival dark ride that serves as the focus of our study.

Project Breakdown



- Existing ride involves a fixed rail circuit, along which small motorized four-person vehicles.
- Each number represents a conventional “trick”, which is a device that is designed to scare riders. Tricks are often constructed of styrofoam, and motion is activated by proximity sensors.

Survey

- A survey was conducted by our group of 11 groups of riders as they exited the "Haunted Mansion"
- Results of survey were overwhelmingly negative.
- Results did bring valuable insight into how to improve the ride.

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.

Problems

1. Tricks are being damaged by “jumpers” (patrons who leave their ride vehicle before the ride is over).
2. Attendance of ride is lowered by poor quality of entertainment.
3. Riders are not entertained/enthralled by ride, and leave their seats, risking severe bodily injury or death.

Thesis

To reduce the risk of ride quality being compromised, and rider safety being put at ricks, replacing aging, damaged "tricks" with new technology to entertain riders will make the "haunted mansion" safer, more enjoyable, and more profitable for the travelling amusement industry.

By introducing "gamification" to ride to reward patrons with good behavior and to reprimand riders / alert ride operator of bad behavior.

We look to a closely related industry for inspiration.

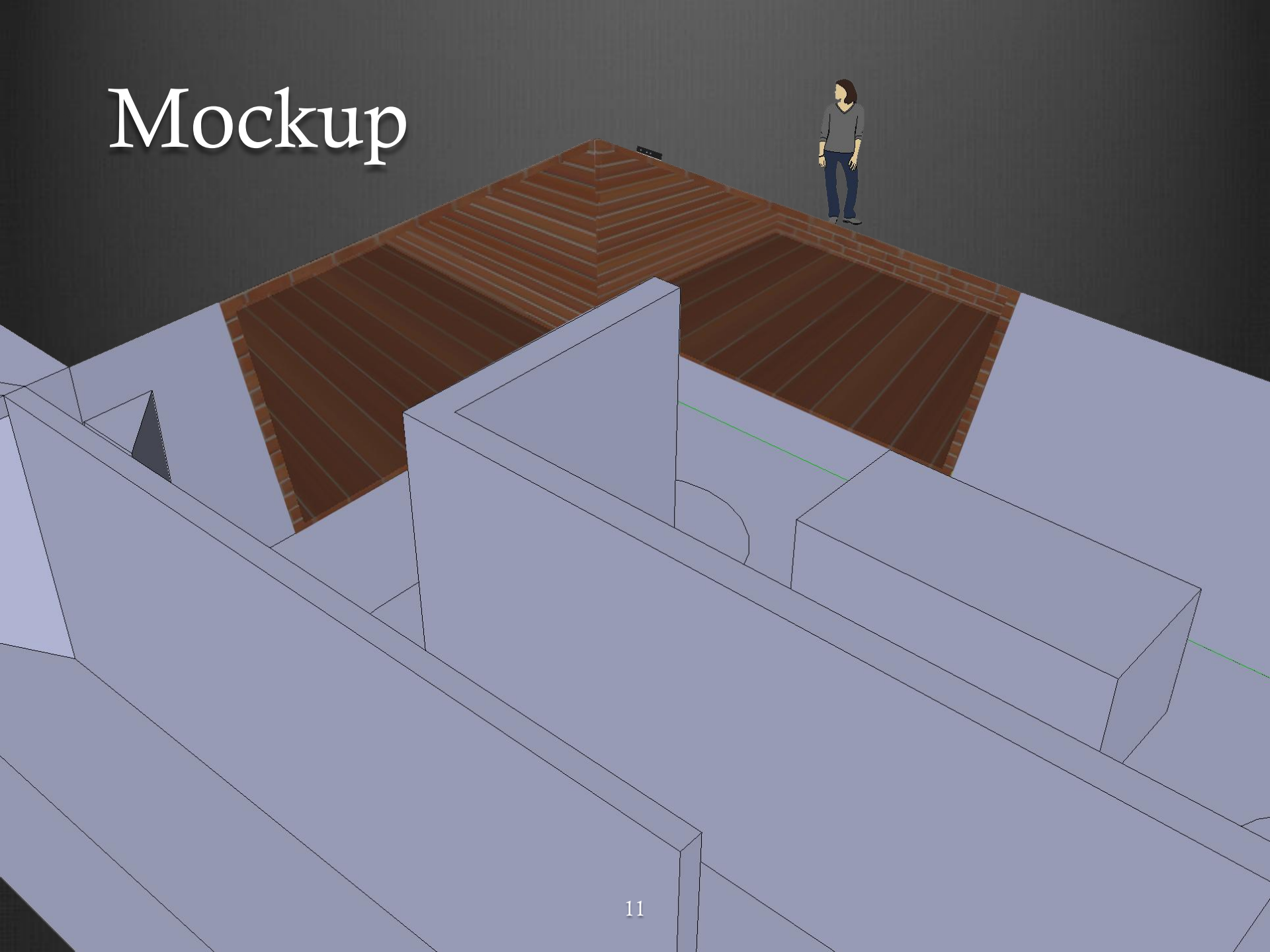
Industry Research

Inspiration can be drawn from an existing related rides: the permanently install theme park ride industry.

Result of Research

- Replace styrofoam “tricks” with projection screens.
  - Aging tricks will no longer be damaged, entertainment elements of ride can now be upgraded or replaced via software.
- Track exact location of ride vehicles within the ride.
  - Allows for entire system to be aware of where ride vehicles are at all times for both safety and show synchronization.
- Introduce interactive element for riders to become involved with the ride.
  - Instead of using guns or other physical props, use sensors that allow riders to interact with ride without damaging equipment. New gestures can be added via software.

Mockup

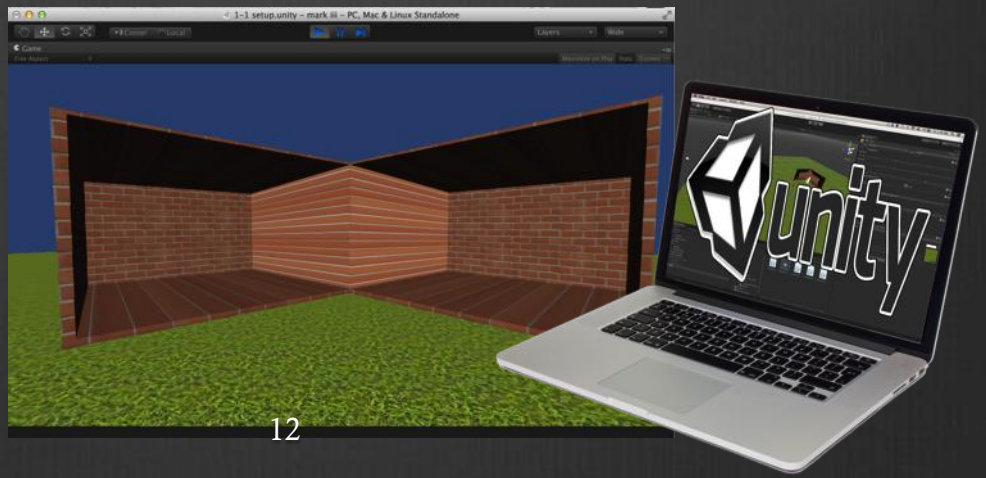


Breakdown of Components

X, Y
Coords

Gesture
Boolean

Output Video
for Projection Screens



Gesture Recognition

- Research has shown patrons are more likely to stay in their seats on a dark ride if engaged
- Some examples:

  -Toy Story Midway Mania (Disney)

  -Men in Black (Universal)

Low Cost Solution

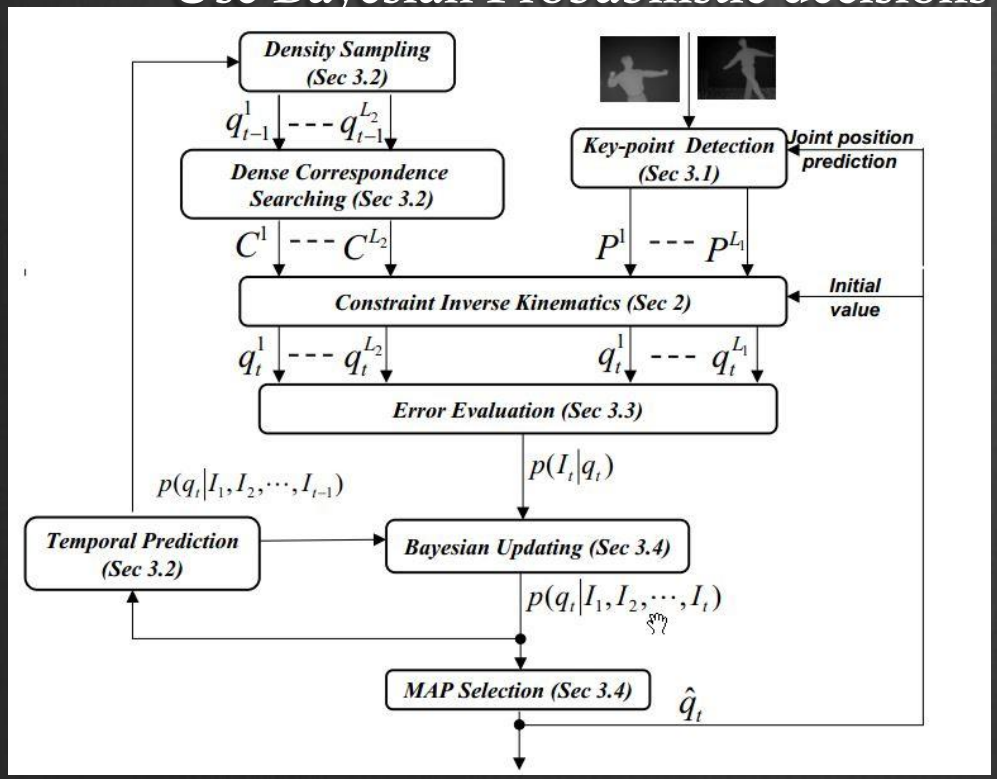
- Unfeasible in carnivals
  - This technology requires too high cost, calibration, and is susceptible to wear (most carnivals travel). It also has to be robust and away from reach
- Solution: Use the Kinect
  - Low Cost device, requires no controller except patrons' gestures
  - Can also be used as a security device. Track if user is leaving the seat

Gestures Used

- Hand Heights
  - Can use this gesture to hit objects, move obstacles out of the way.
  - Can also use to force user to perform arm gestures
- Ducking
  - Can be used to avoid obstacles such as logs hitting you. Can award negative points if hit from this
- Walking off set
  - Negative points can be awarded if leaving the screen

Skeletal Tracking

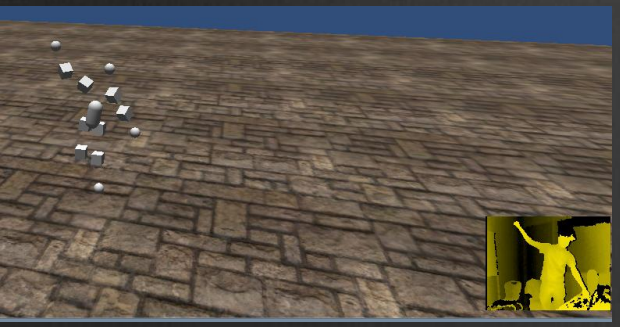
- Kinect has great skeleton tracking built in!
- Need to extract parts (hands, legs, torso, head)
- Use Bayesian Probabilistic decisions for tracking



Gesture Technique Process

Skeleton Tracking

Wrappers in Unity



Coding in Gestures

Transferring Data to Unity

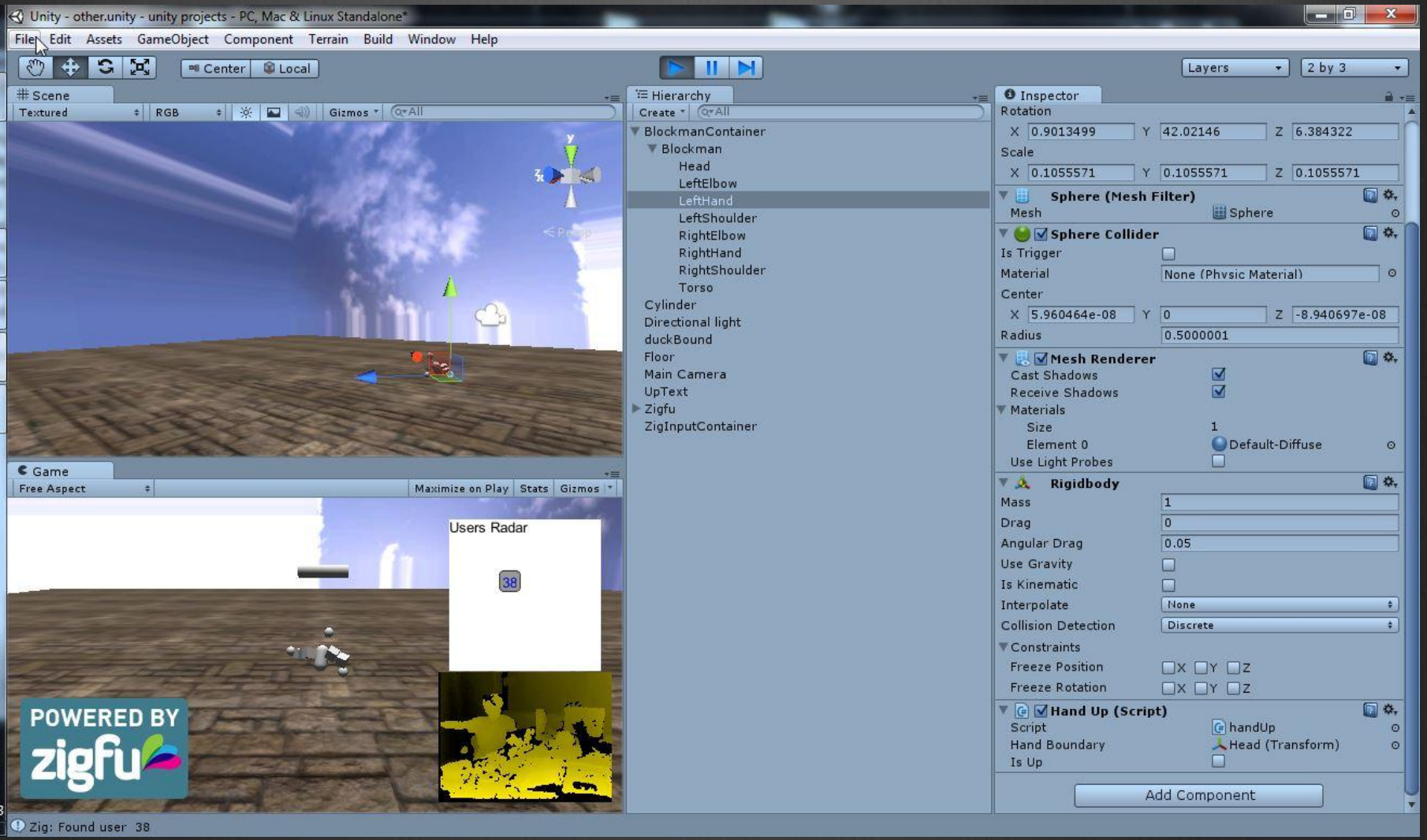
- Problem: How do we translate this knowledge to a game engine?
- Zigfu is a wrapper which does this but much is required to script in gestures and limbs
- Gestures needed for good and bad behavior
- Must be robust and not prone to errors

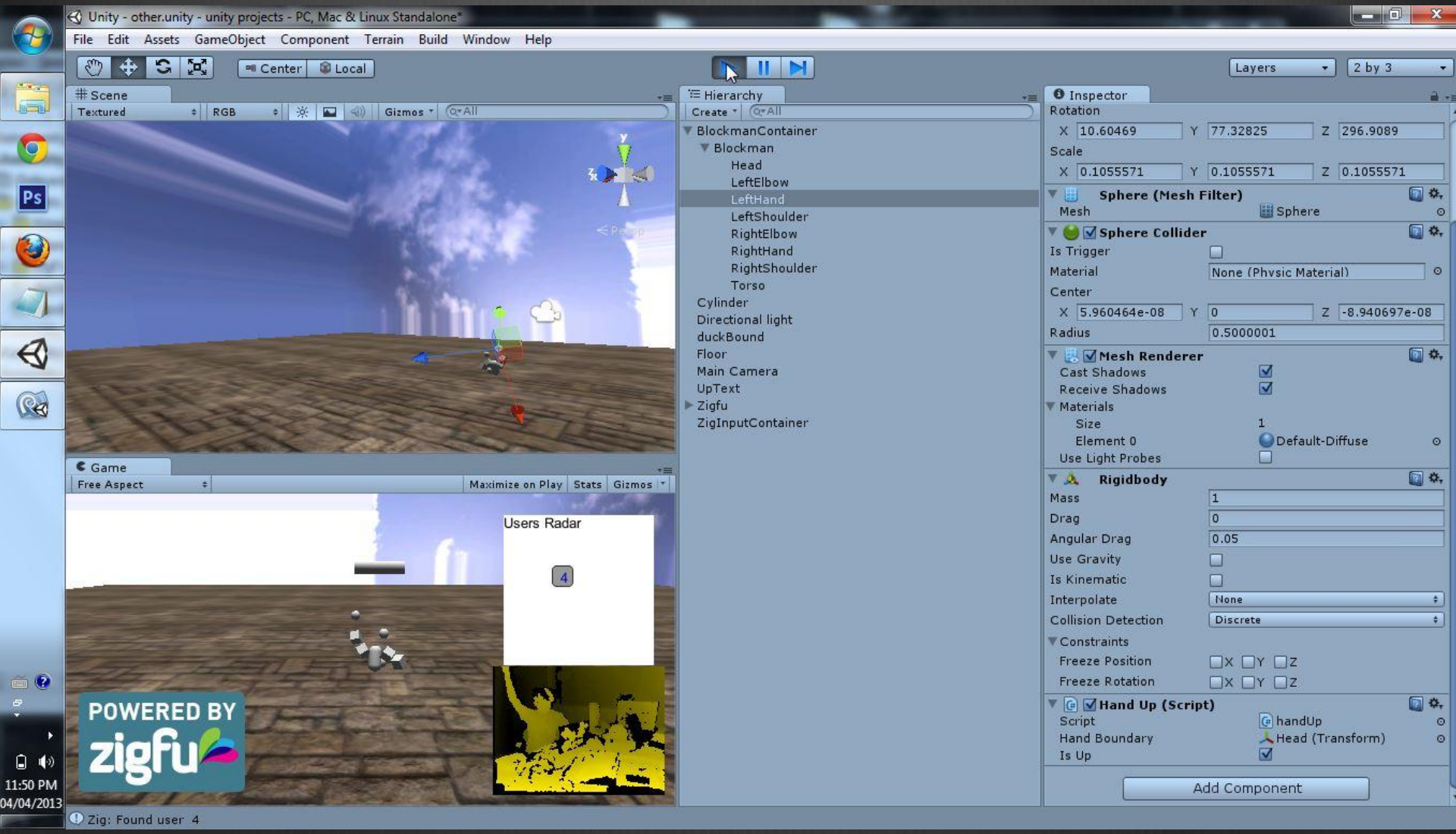
Pseudocode for Ducking

- Take in Skeleton data
- Extract and separate head
- Find a mean euclidean distance between head object and torso object (problem with taking from plane, position will change over time)
- Script in rigidbody material (physics) to ensure collisions get detected

Pseudocode for hand up

- What constitutes a hands being up? Hands being above head
  - Take in Skeleton data
  - Extract and separate head and hand data
  - Script in rigidbody material (physics) to ensure collisions get detected
  - Simply set the boundary with if statements comparing the planes of hands and the head object (boolean)


Unity - other.unity - unity projects - PC, Mac & Linux Standalone*
File Edit Assets GameObject Component Terrain Build Window Help
Center Local
Layers
2 by 3
Scene
Textured RGB Gizmos All
Persp
Game
Free Aspect Maximize on Play Stats Gizmos
Users Radar
38
POWERED BY zigfu
Hierarchy
Create All
BlockmanContainer
Blockman
Head
LeftElbow
LeftHand
LeftShoulder
RightElbow
RightHand
RightShoulder
Torso
Cylinder
Directional light
duckBound
Floor
Main Camera
UpText
Zigfu
ZigInputContainer
Inspector
Rotation
X 0.9013499 Y 42.02146 Z 6.384322
Scale
X 0.1055571 Y 0.1055571 Z 0.1055571
Sphere (Mesh Filter)
Mesh Sphere
Sphere Collider
Is Trigger
Material None (Physic Material)
Center
X 5.960464e-08 Y 0 Z -8.940697e-08
Radius 0.5000001
Mesh Renderer
Cast Shadows
Receive Shadows
Materials
Size 1
Element 0 Default-Diffuse
Use Light Probes
Rigidbody
Mass 1
Drag 0
Angular Drag 0.05
Use Gravity
Is Kinematic
Interpolate None
Collision Detection Discrete
Constraints
Freeze Position X Y Z
Freeze Rotation X Y Z
Hand Up (Script)
Script handUp
Hand Boundary Head (Transform)
Is Up
Add Component
Zig: Found user 38
11:51 PM
04/04/2013


Unity - other.unity - unity projects - PC, Mac & Linux Standalone*
File Edit Assets GameObject Component Terrain Build Window Help
Center Local
Layers
2 by 3
Scene
Textured RGB Gizmos
Persp
Hierarchy
Create
BlockmanContainer
Blockman
Head
LeftElbow
LeftHand
LeftShoulder
RightElbow
RightHand
RightShoulder
Torso
Cylinder
Directional light
duckBound
Floor
Main Camera
UpText
Zigfu
ZigInputContainer
Inspector
Rotation
X 10.60469 Y 77.32825 Z 296.9089
Scale
X 0.1055571 Y 0.1055571 Z 0.1055571
Sphere (Mesh Filter)
Mesh Sphere
Sphere Collider
Is Trigger
Material None (Physic Material)
Center
X 5.960464e-08 Y 0 Z -8.940697e-08
Radius 0.5000001
Mesh Renderer
Cast Shadows
Receive Shadows
Materials
Size 1
Element 0 Default-Diffuse
Use Light Probes
Rigidbody
Mass 1
Drag 0
Angular Drag 0.05
Use Gravity
Is Kinematic
Interpolate None
Collision Detection Discrete
Constraints
Freeze Position X Y Z
Freeze Rotation X Y Z
Hand Up (Script)
Script handUp
Hand Boundary Head (Transform)
Is Up
Add Component
Game
Free Aspect
Maximize on Play Stats Gizmos
Users Radar
4
POWERED BY zigfu
Zig: Found user 4
11:50 PM
04/04/2013

Bad Behavior – Standing!

- Legs are automatically taken off the skeleton tracking
- Activates once user stands up and tries to leave
- Although kinect has depth perception, bad at position tracking, where Mirko's part comes in
- We can use this conflicting experiences to detect bad behaviour

Things to Add

- Facial Recognition – to Reconstruct people's faces onto objects and characters in the game engine
- Swiping and waving – by taking in velocity and how long it takes for hand to move

Position Tracking

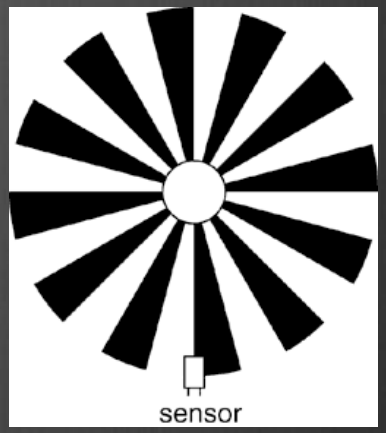
For a Dark Ride Vehicle

By: Mirko Miljevic

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.

Encoder

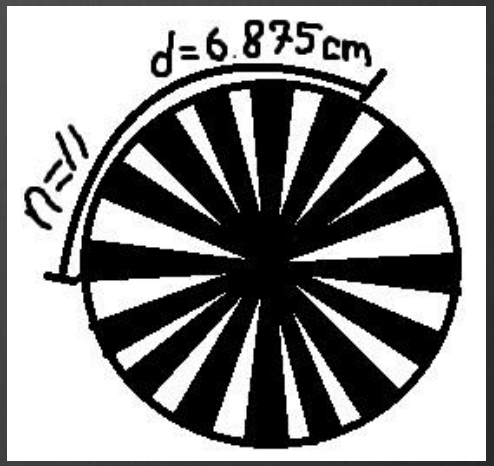


How does an Encoder work?

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 it reads a black division. If there are 40 divisions then every 20th count of ‘1’ would mean the distance travelled is that of the circumference of the wheel.

EXAMPLE #1

An optical disk with 32 divisions (n=32), and a circumference of 20cm (c = 20cm) would yield a travel distance of;

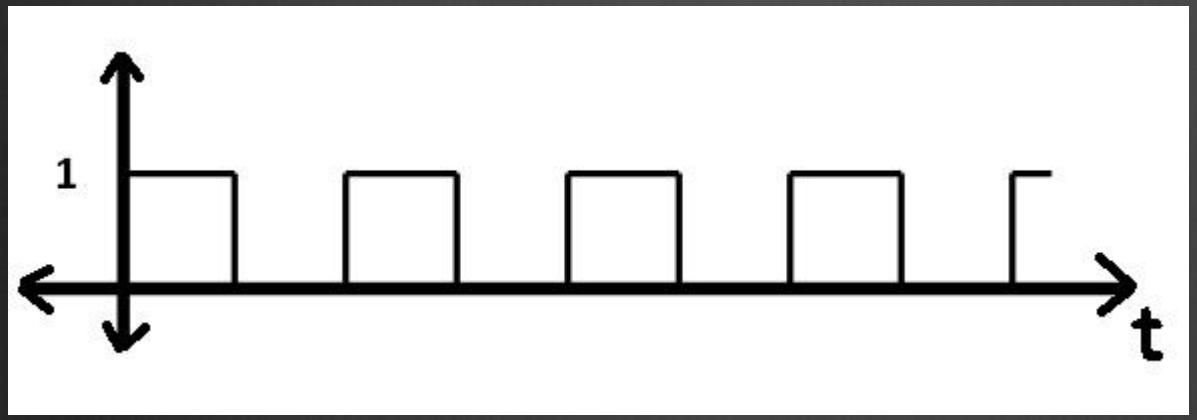


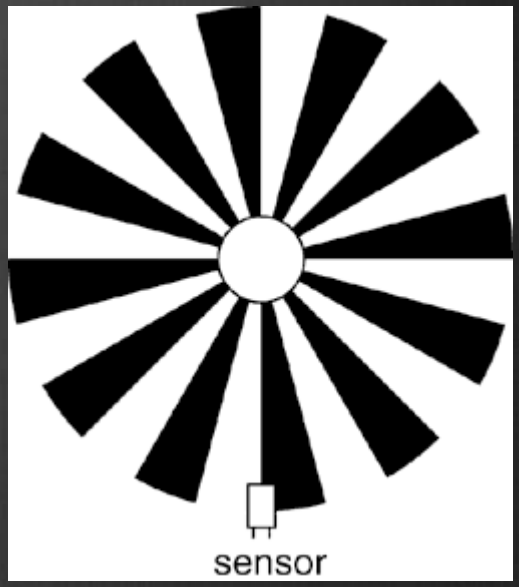
$$d = \frac{c}{n} = \frac{20cm}{32} = 0.625cm/n$$

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.

UNDERSTANDING MOVEMENT

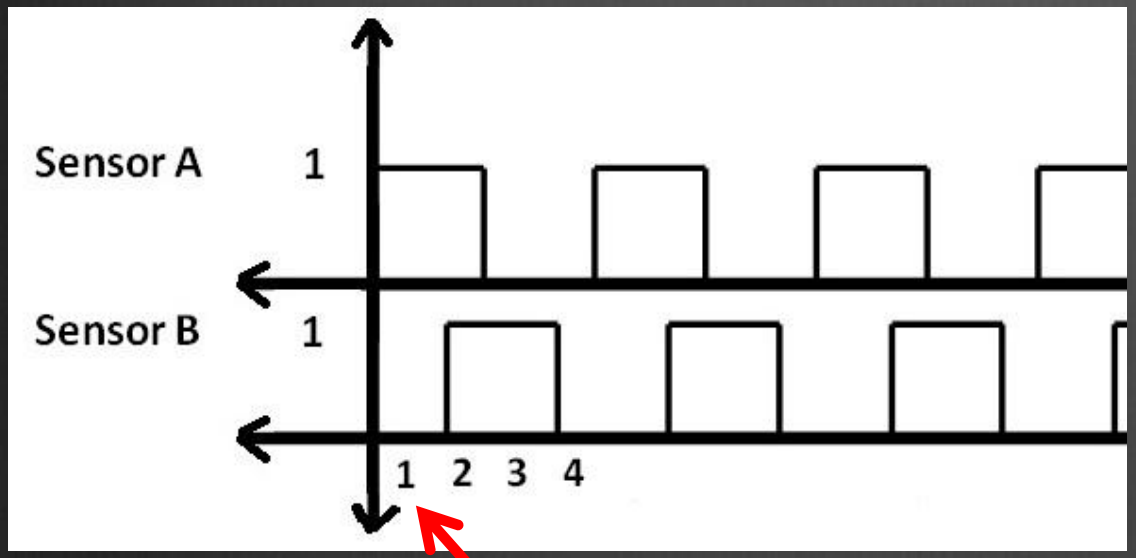
One Directional Movement

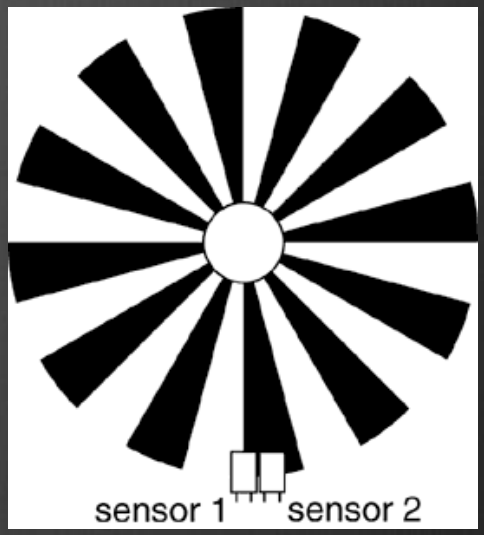




For one direction, all you need is one encoder that will read a HIGH or a LOW signal.

FORWARD & BACKWARD MOVEMENT



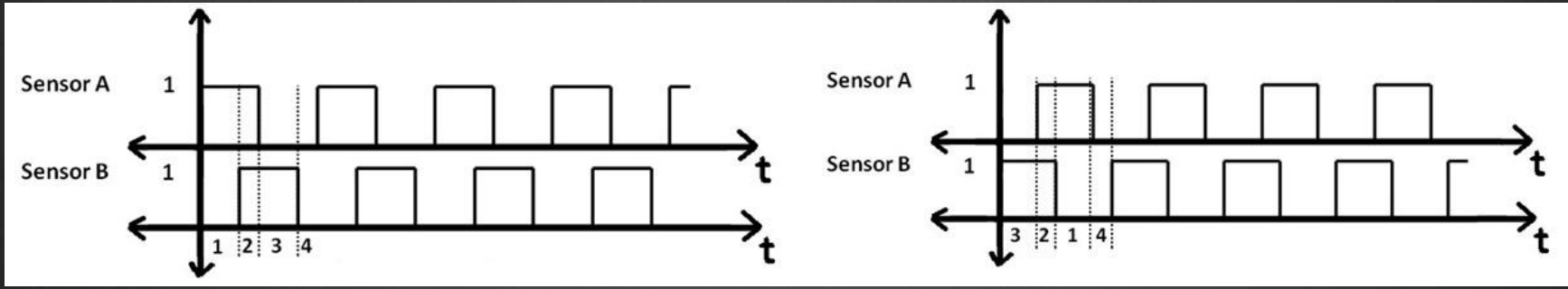


(HIGH, LOW) or (1,0)

Two sensors, that are off set a bit, create various patterns such as the one above. (i.e. State 1 = HIGH, LOW)

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

FORWARD & BACKWARD
(Continued…)



FORWARD MOVEMENT

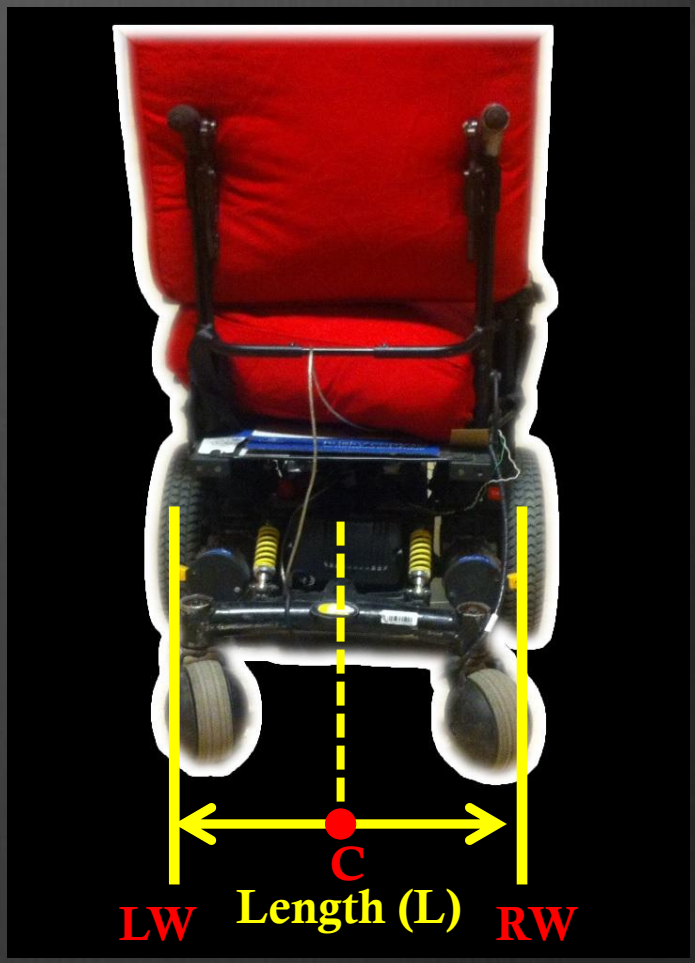
For forward movement, the states change in a pattern of increasing order (1,2,3,4,1,2,3….)

BACKWARD MOVEMENT

For backward movement the states change in decreasing order (3,2,1,4,3,2,1,4,3……)

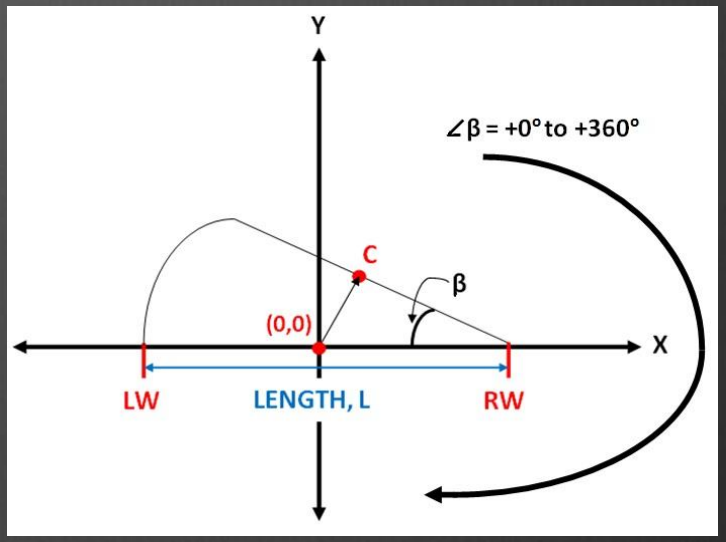
CALCULATING TURNING

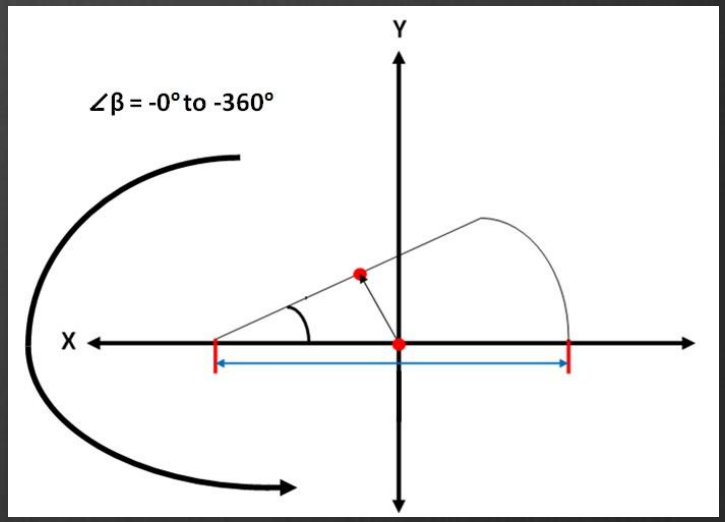
To calculate the turning of a two wheeled system we must initialize a few things. First we separate the left wheel (LW) and the right wheel (RW). The length from the centre of the LW and the centre of the RW is taken. Where the middle point of that length, L, is the orientation point of the chair which is used to tell us where on the xy-plane that point (the chair) is.



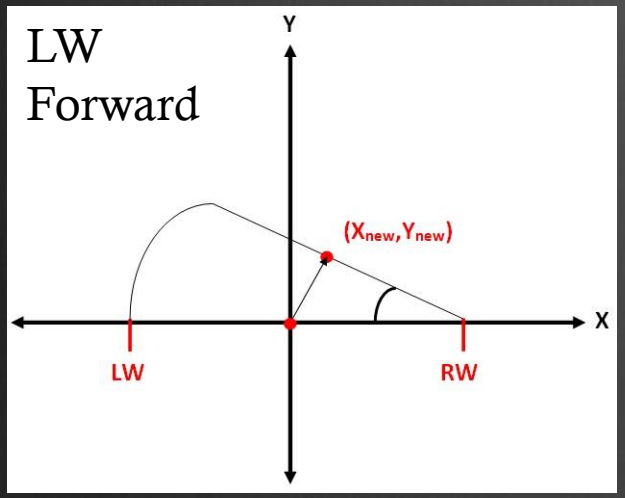
CALCULATING TURNING

The microcontroller not only reads the encoder signals and relays information wirelessly, it also does the math. Every few microseconds it calculates the change in position of each wheel (forwards/backwards). This is why the angle Beta is very important. The image on the top shows that Beta is considered to be positive if it ventures to the clockwise direction of the Y-axis and the image on the bottom shows that Beta is considered to be negative if it ventures to the counter-clockwise direction of the Y-axis.

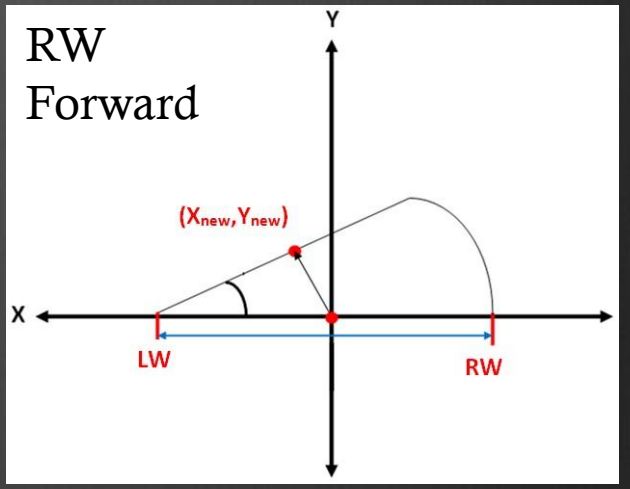


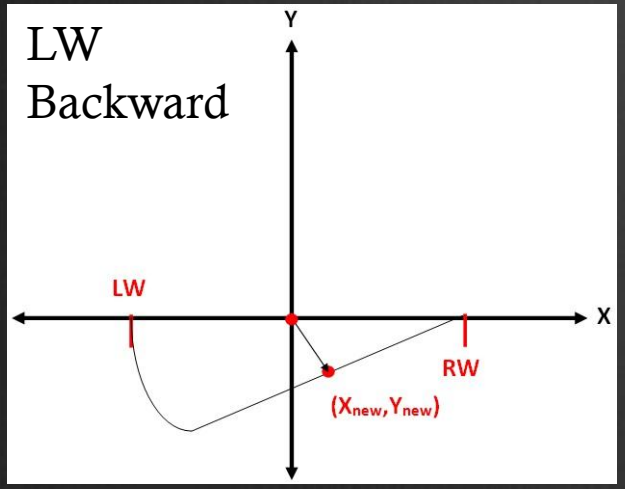


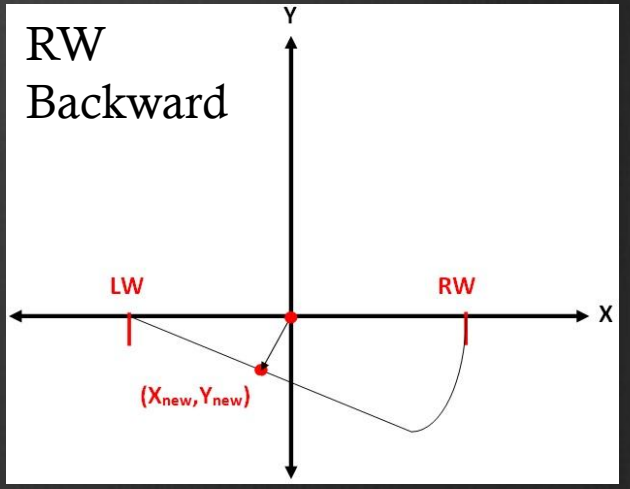
X and Y Coordinates



When the LW goes forward the new X and Y coordinates are in the upper right quadrant. When the RW goes forward the new X and Y coordinates are in the upper left quadrant. When the LW goes backwards the new X and Y coordinates are in the lower right quadrant . When the RW goes backwards the new X and Y coordinates are in the lower left quadrant.

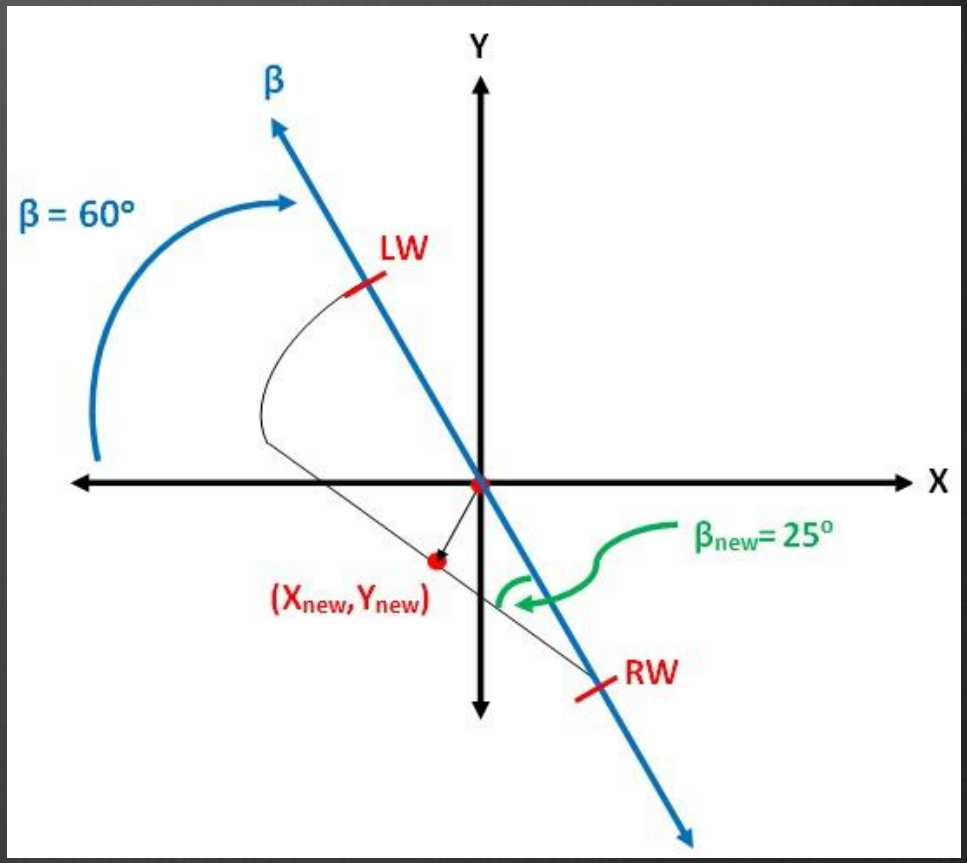






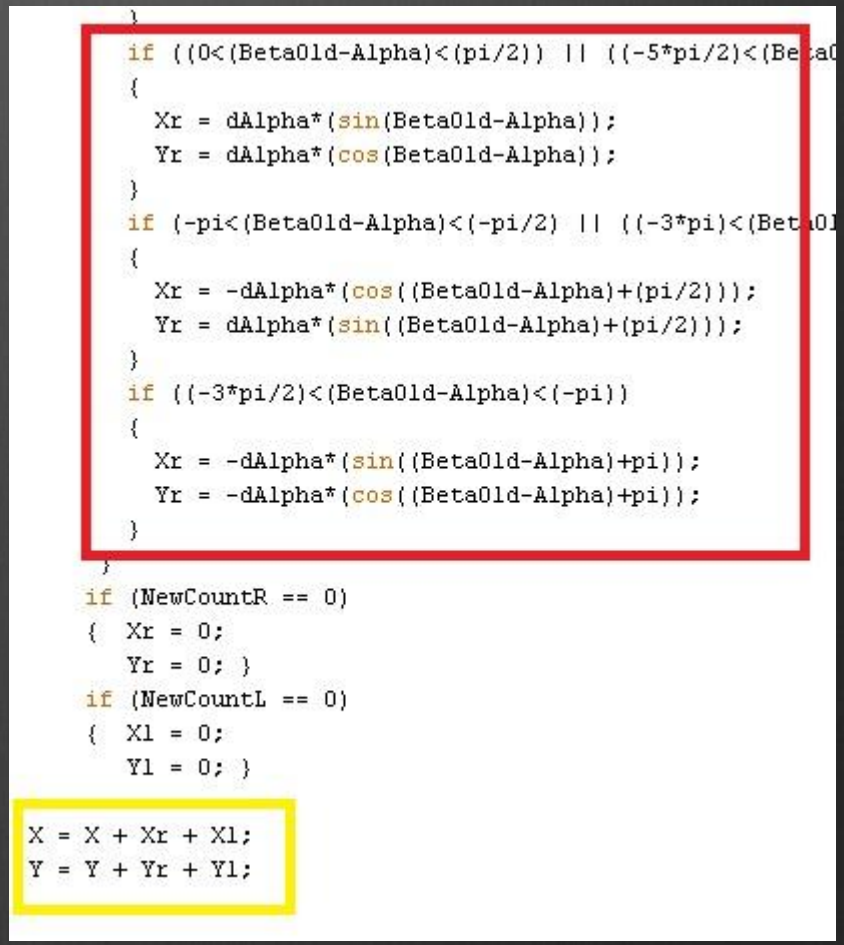
EXAMPLE #2

Take for example; after a few calculations Beta (the chair) was at one instant 60° in the positive direction (to the right). If at that instant the LW moved backwards the new X and Y coordinates would be in the bottom right quadrant. The next Beta would be calculated to be at Beta – $Beta_{new}$ (i.e. 60 – 25 = 35°).



X and Y Coordinates (continued…)

Essentially that leaves you with a bunch of scenarios (which are shown as 'if statements' in Arduino). Since the X and Y coordinates for the LW and the RW are separate the final value for X is the old X value plus the LW-X value and RW-X value. The same goes for the Y value.



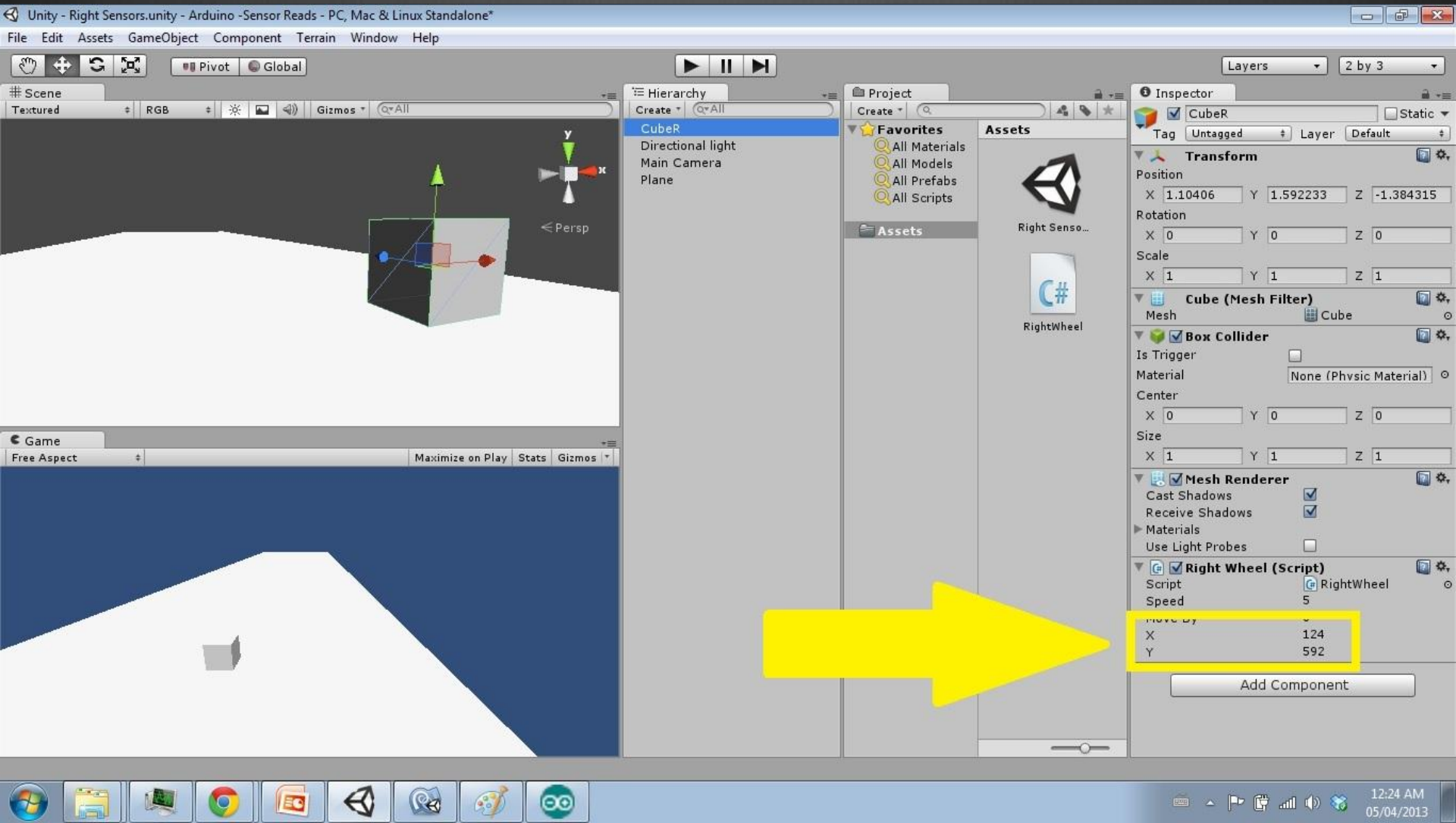
```
    }
    if ((0<(BetaOld-Alpha)<(pi/2)) || ((-5*pi/2)<(Be
    {
      Xr = dAlpha*(sin(BetaOld-Alpha));
      Yr = dAlpha*(cos(BetaOld-Alpha));
    }
    if (-pi<(BetaOld-Alpha)<(-pi/2) || ((-3*pi)<(Bet
    {
      Xr = -dAlpha*(cos((BetaOld-Alpha)+(pi/2)));
      Yr = dAlpha*(sin((BetaOld-Alpha)+(pi/2)));
    }
    if ((-3*pi/2)<(BetaOld-Alpha)<(-pi))
    {
      Xr = -dAlpha*(sin((BetaOld-Alpha)+pi));
      Yr = -dAlpha*(cos((BetaOld-Alpha)+pi));
    }
  }
  if (NewCountR == 0)
  {  Xr = 0;
     Yr = 0; }
  if (NewCountL == 0)
  {  Xl = 0;
     Yl = 0; }

X = X + Xr + Xl;
Y = Y + Yr + Yl;
```

Connection with Unity 3D

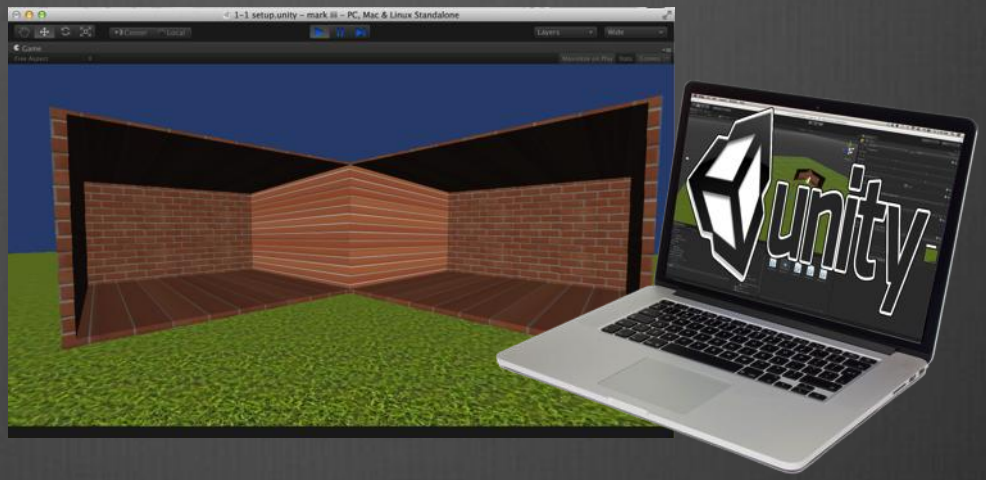
To import the X and Y values into Unity 3D, we've created a script in C# that reads the values from the COM4 port (COM8 for the wireless module). The values are read as a string array, split up, and then parsed into an integer number…..

```
using UnityEngine;
using System.Collections;
using System.IO.Ports;


public class RightWheel : MonoBehaviour {

    public float speed;
    public float MoveBy;
    public int X = 0;
    public int Y = 0;


    SerialPort sp = new SerialPort("COM4", 9600);

    // Use this for initialization
    void Start () {

        sp.Open();
        sp.ReadTimeout = 500;

    }

    // Update is called once per frame
    void Update () {
        string value = sp.ReadLine(); //Read the information
        string[] vec3 = value.Split(',');
        X = int.Parse(vec3[0]);
        Y = int.Parse(vec3[1]);

}}
```

….End Result = X and Y Coordinates in UNITY 3D



Output Video
for Projection Screens



Augmented Reality Effect and Projection Video Generation

Problem

- Adding create visual “tricks” that are interactive with riders can be accomplished by using projection screens.
- However, the space available inside the ride is very limited.
- Riders placed too close to a screen will have difficulty distinguishing objects.
- We can simulate a larger area within the ride by creating the illusion that the interior of the ride is actually large than the confines of the unfolded trailer by using an augmented reality effect.
- This effect requires precise, low latency information about the coordinates of the riders in order to work.

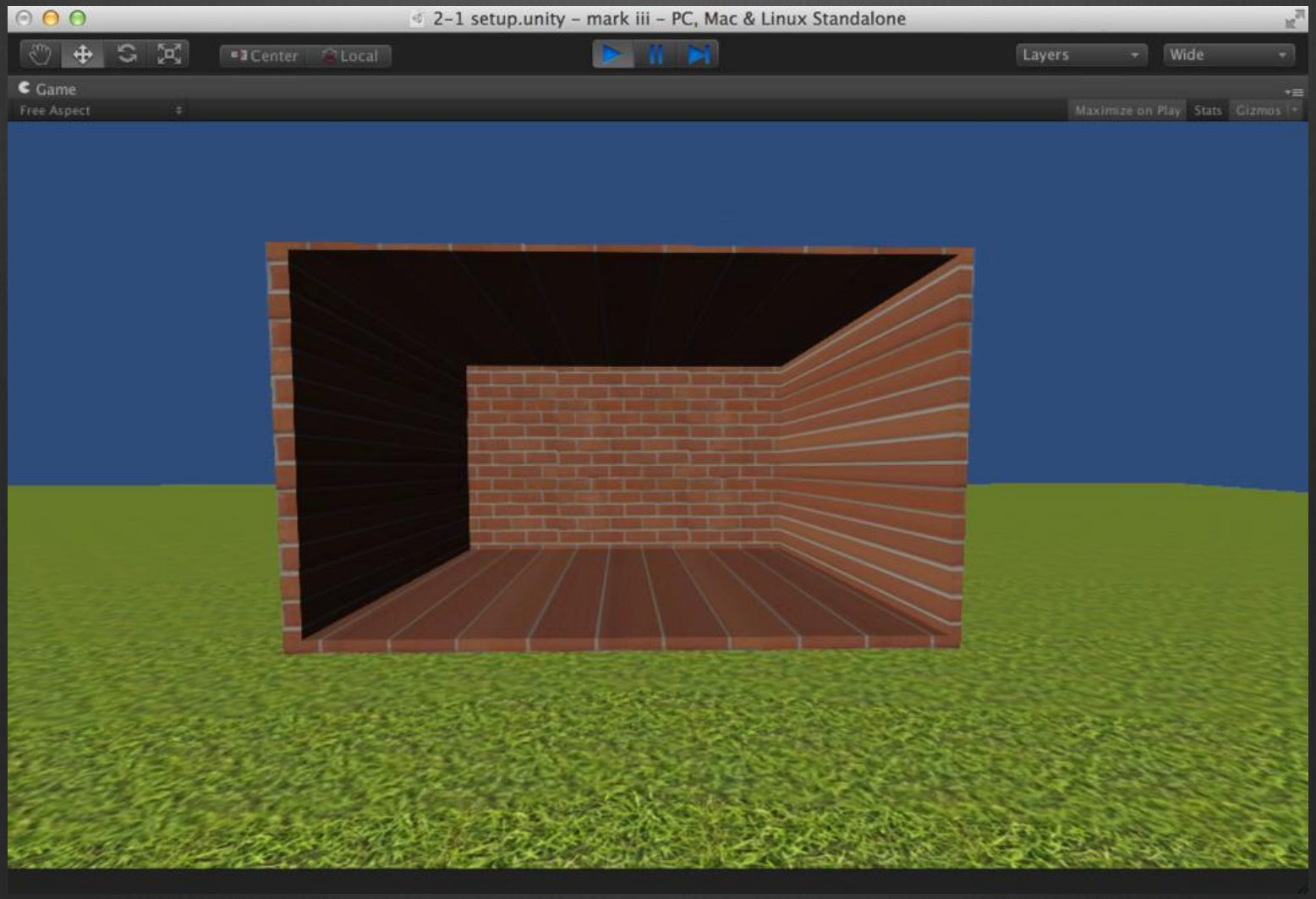
Industry Research

- This effect is used extensively in one of the most impressive theme park industry dark-ride known as “The Amazing Adventures of Spider Man” at Universal Studios, Florida.
- Our group experienced the ride first hand in February of this year.
- The following video is a behind the scenes expose of the engineering involved behind the process.

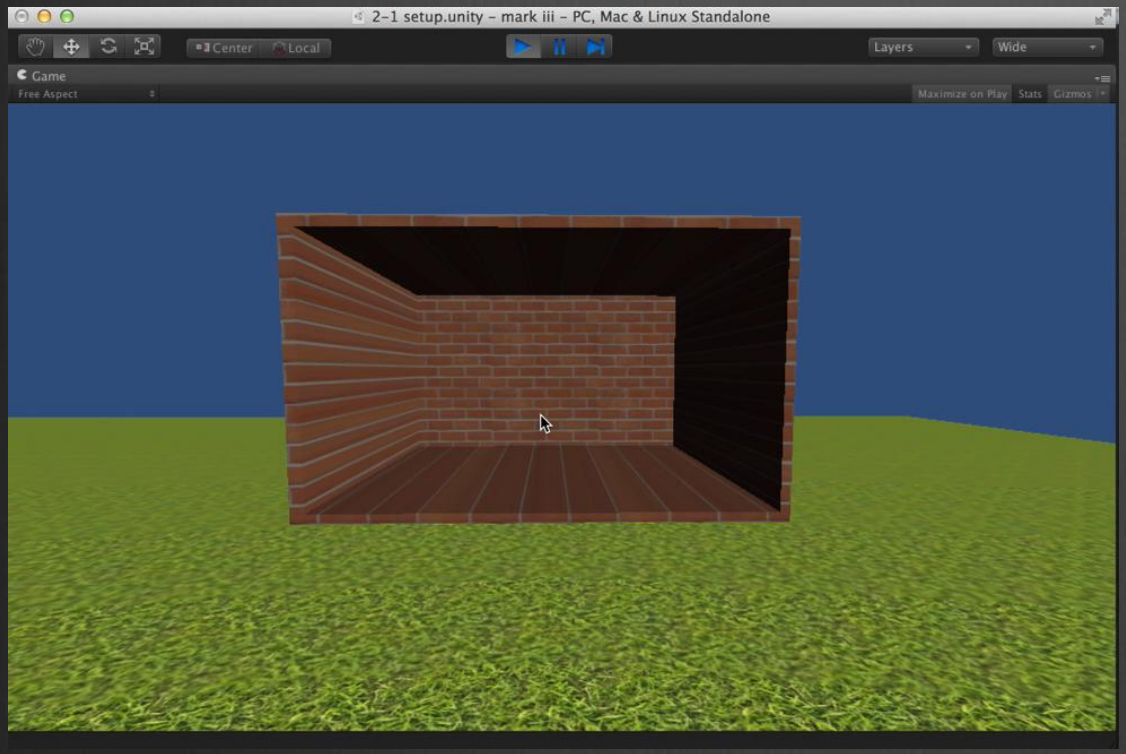
Adapting the Effect for our Project

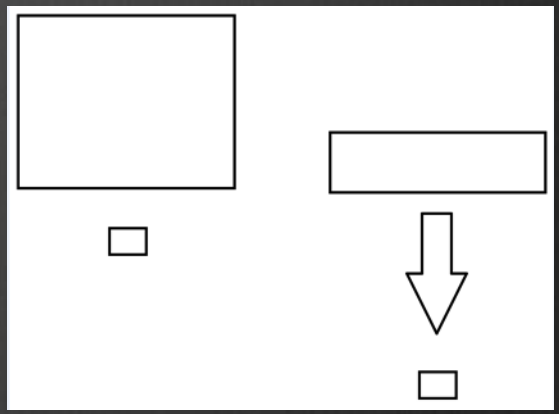
- Since video projected onto walls will need to be interactive with riders, **pre-rendered** video like the video used in "Spider-man" will not suffice.
- Unity3D, a commonly available, highly modifiable video game engine is used in order to generate the effect in **real-time**.
- The effect, which was referred to as "squinching" in the previous video, is not well documented, so the effect must be reverse engineered.
- Once the effect can be replicated, the possibility of extending it to multiple screens becomes a reality.

A Working One Wall Example



Components of the Effect - Zooming





As the viewer moves away from the box, the back wall appears to grow larger. As the viewer moves closer, the back wall appears to shrink.

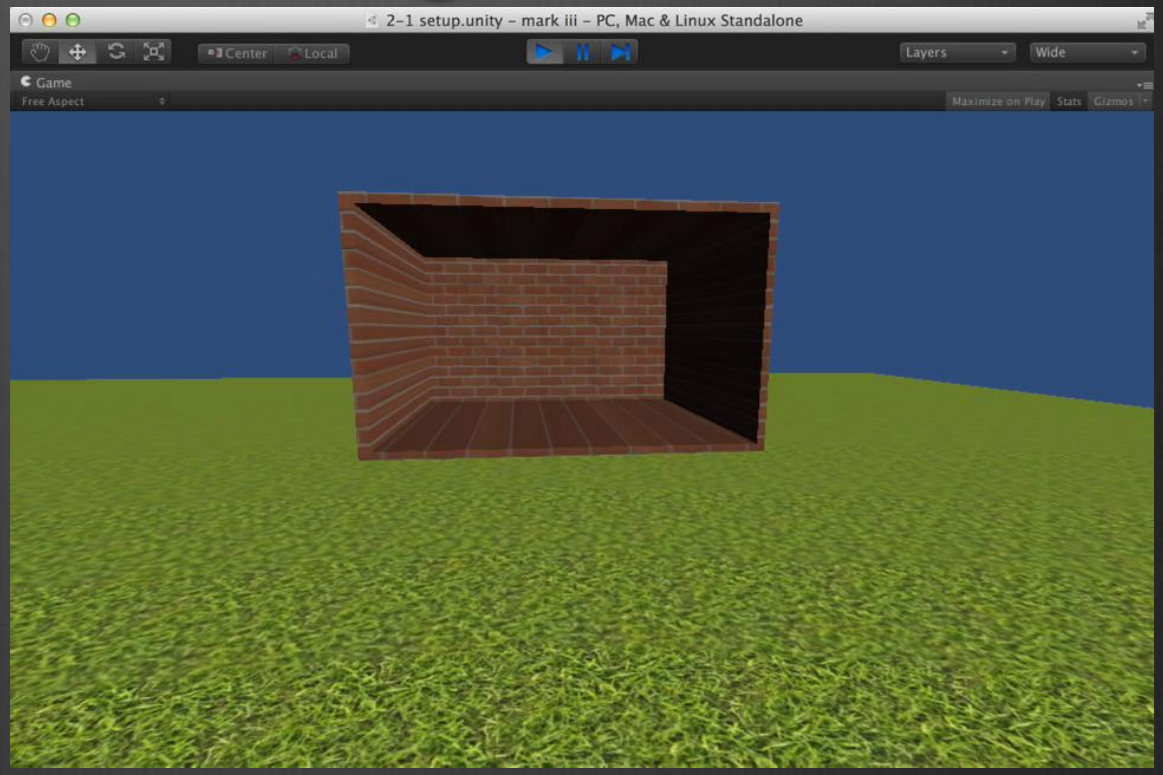
This is due to the "perspective view" that is seen by the common eye, and emulated in the above 3D game engine.

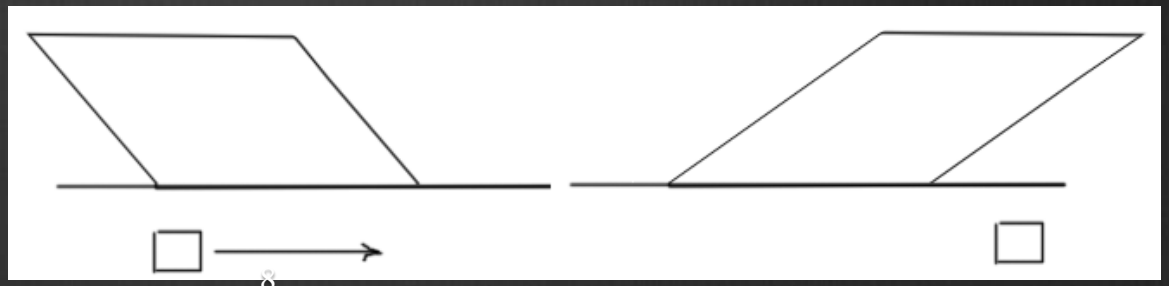
Components of the Effect - Skewing

As the viewer moves laterally across the screen, the back wall appears to move with the viewer.

If the viewer moves far enough, the back wall disappears completely.

This effect is replicated using "skewing".

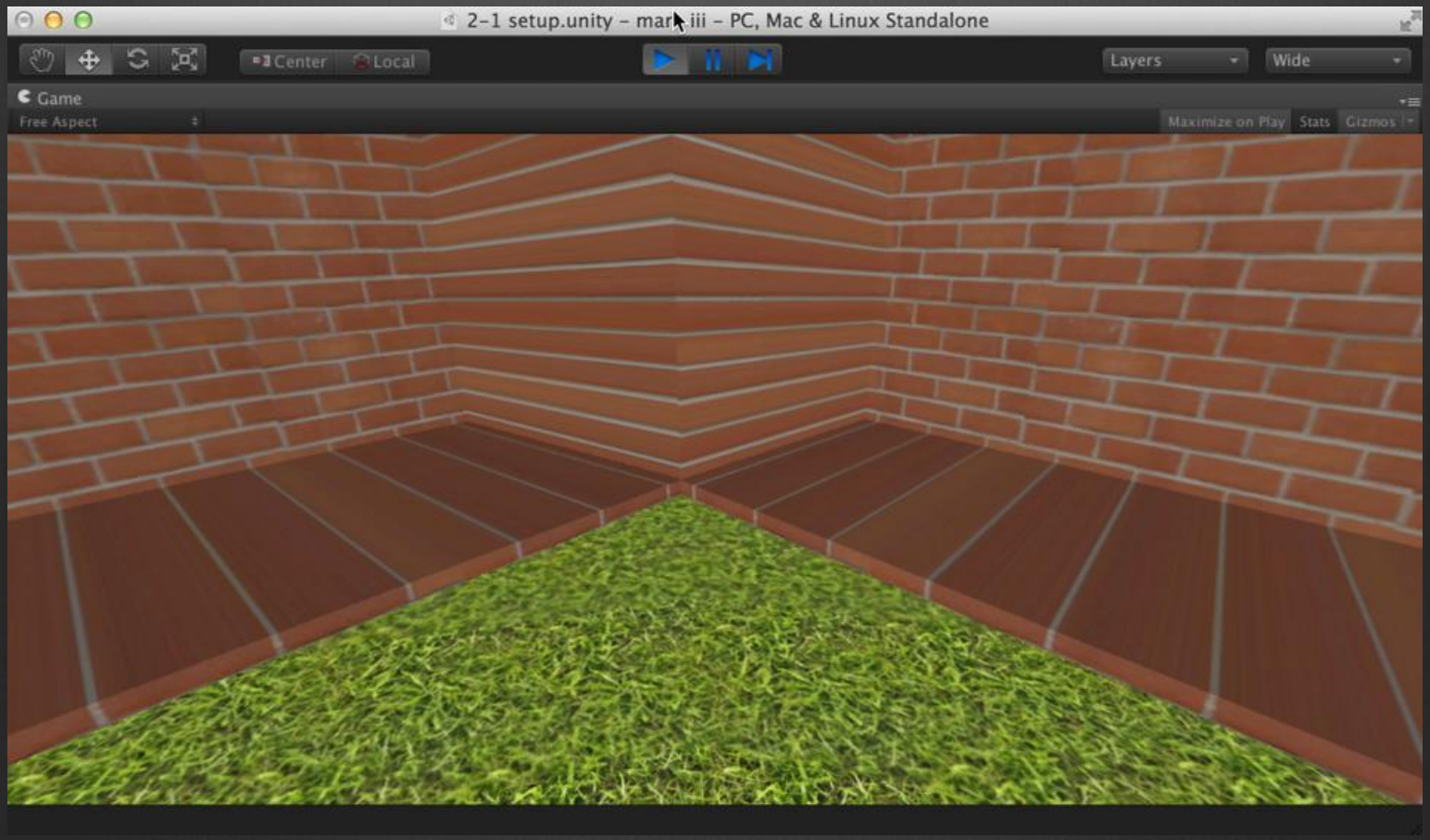




Morphing and Skewing

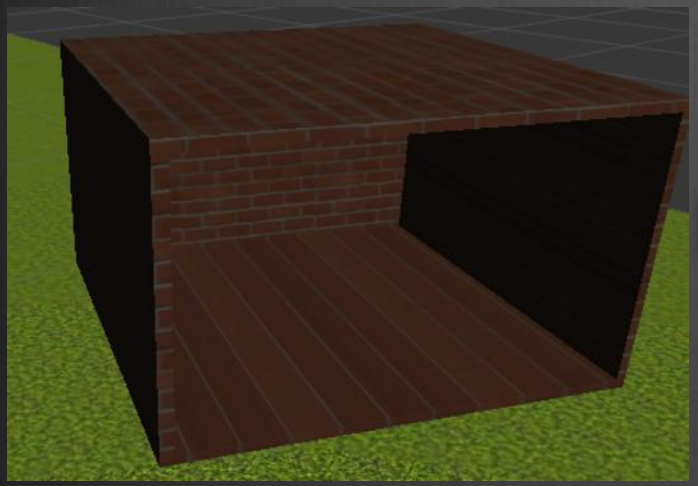
- Morphing and skewing is not supported directly by Unity3D.
- A third-party package called “Mega-Fiers” was purchased and utilized.

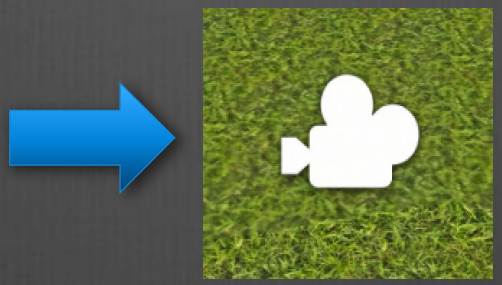
Two Wall Example

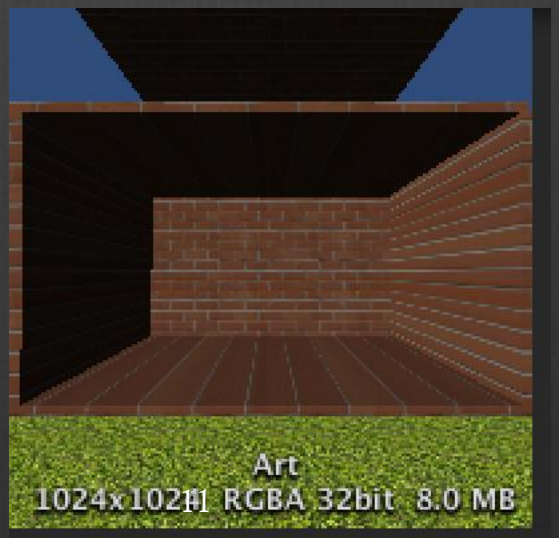


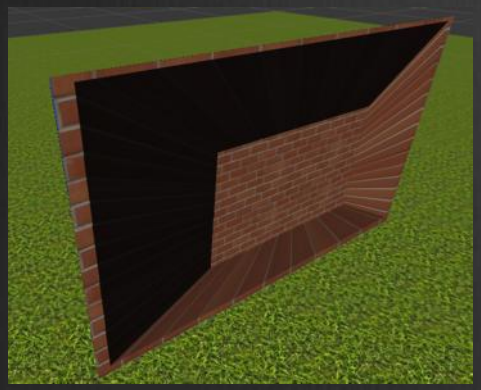
Note: This effect requires some tuning.

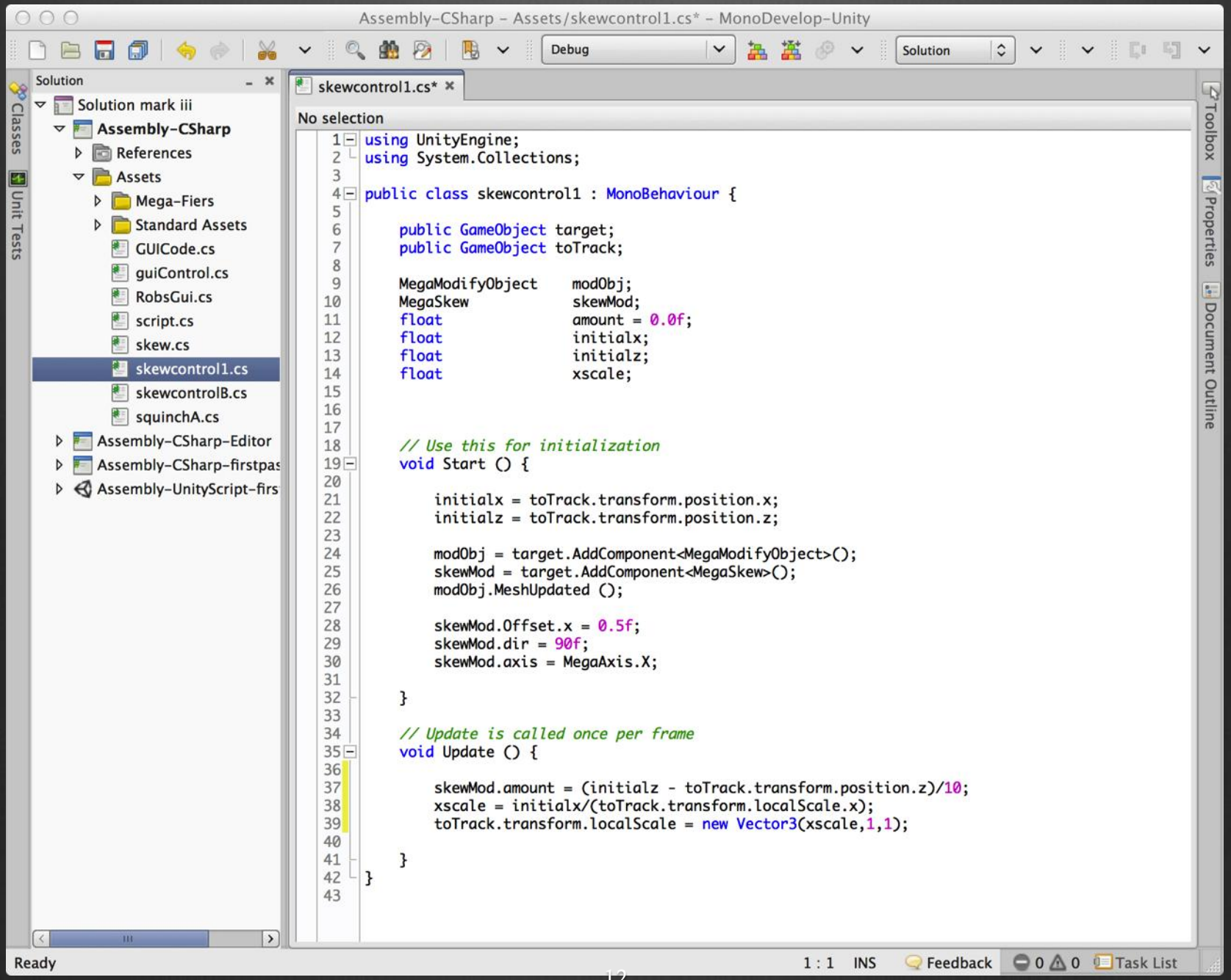
How Does This Work?

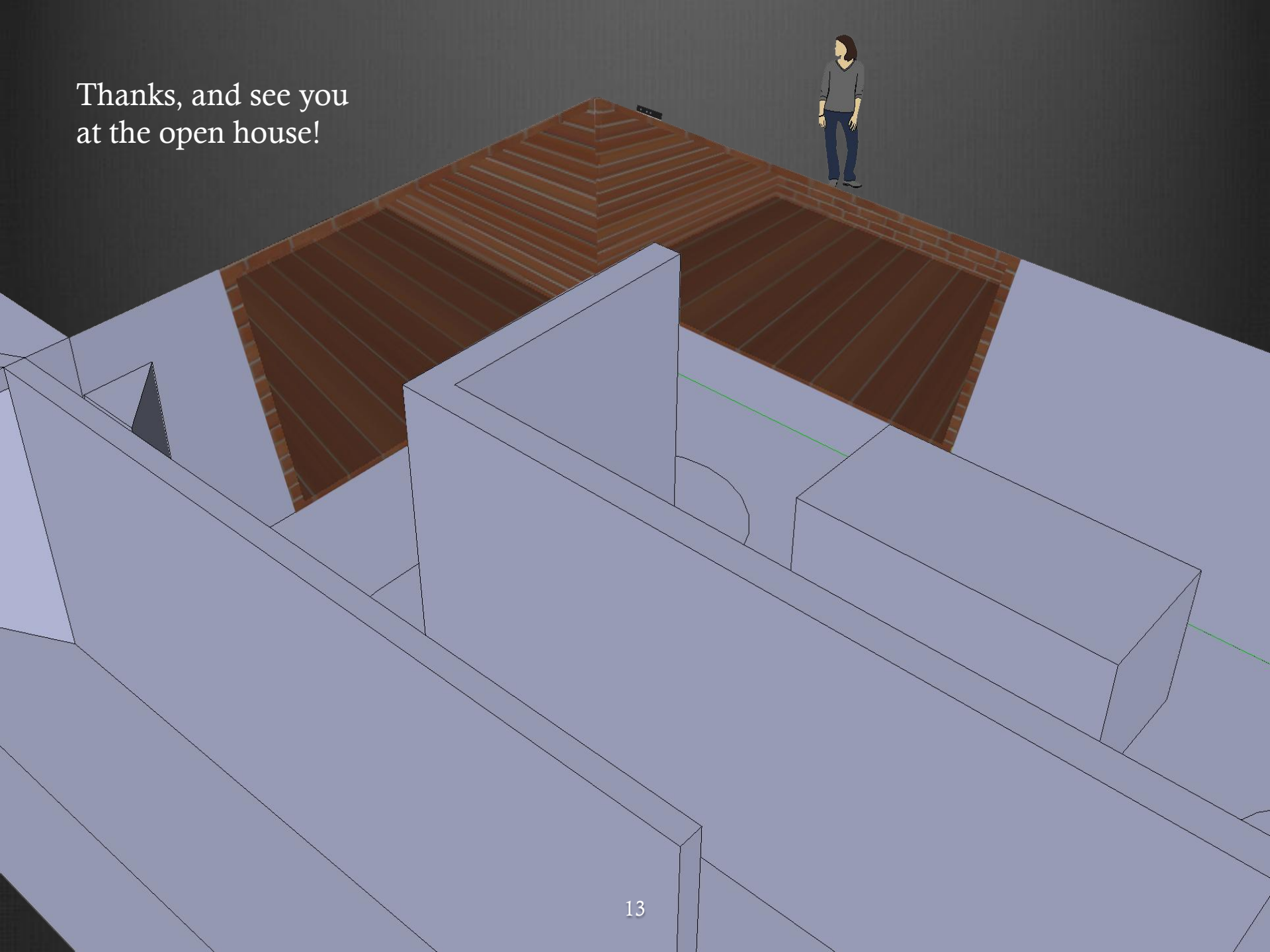









Assembly-CSharp - Assets/skewcontrol1.cs* - MonoDevelop-Unity
Debug
Solution
Solution
Solution mark iii
Assembly-CSharp
References
Assets
Mega-Fiers
Standard Assets
GUICode.cs
guiControl.cs
RobsGui.cs
script.cs
skew.cs
skewcontrol1.cs
skewcontrolB.cs
squinchA.cs
Assembly-CSharp-Editor
Assembly-CSharp-firstpas
Assembly-UnityScript-firs
skewcontrol1.cs*
No selection
using UnityEngine;
using System.Collections;
public class skewcontrol1 : MonoBehaviour {
public GameObject target;
public GameObject toTrack;
MegaModifyObject modObj;
MegaSkew skewMod;
float amount = 0.0f;
float initialx;
float initialz;
float xscale;
// Use this for initialization
void Start () {
initialx = toTrack.transform.position.x;
initialz = toTrack.transform.position.z;
modObj = target.AddComponent<MegaModifyObject>();
skewMod = target.AddComponent<MegaSkew>();
modObj.MeshUpdated ();
skewMod.Offset.x = 0.5f;
skewMod.dir = 90f;
skewMod.axis = MegaAxis.X;
}
// Update is called once per frame
void Update () {
skewMod.amount = (initialz - toTrack.transform.position.z)/10;
xscale = initialx/(toTrack.transform.localScale.x);
toTrack.transform.localScale = new Vector3(xscale,1,1);
}
}
Classes
Unit Tests
Toolbox
Properties
Document Outline
Ready
1 : 1
INS
Feedback
0 0
Task List


Thanks, and see you
at the open house!