Project Documentation

Project Title: Augmented Reality
Team Members: Ishaan Dube , Atri Bhattacharya , Narendra Roy
Team Mentor: Anurag Dwivedi

Basic aim:

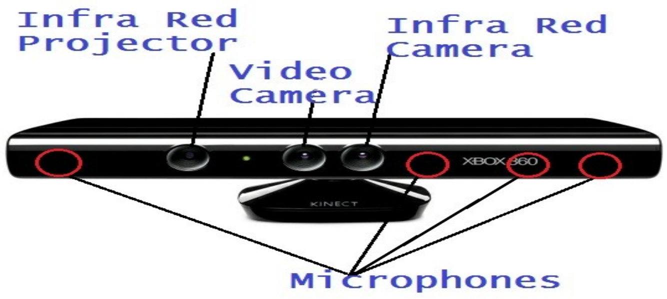
To create a magic show by augmenting objects to our environment in real time using Microsoft Kinect.

Motivation:

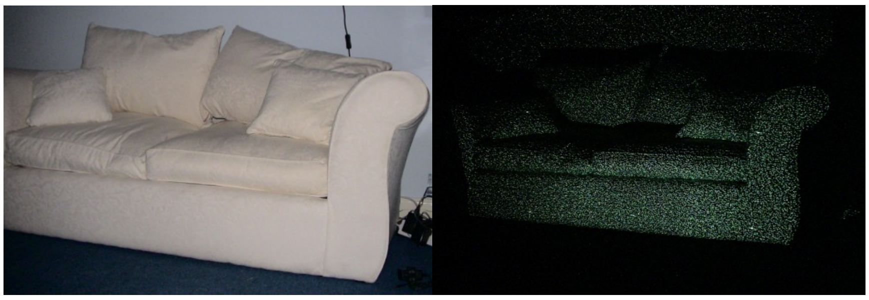
We wanted to make something special and exciting during the summers. So we started with the idea of plasma Speakers. But the Coordinators did not approve the project due to safety issues (had to deal with very high voltages).
We got this idea after watching videos of people playing games using Kinect, particularly because of a TED talk by illusionist Marco Tempest. We were impressed by the Google Glasses concept too.
So the decision to take up this project was finalized with the coordinators.

Theory:

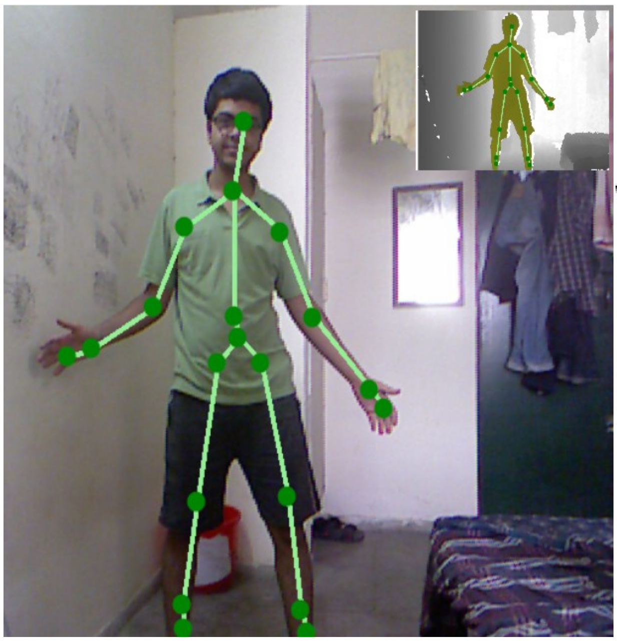
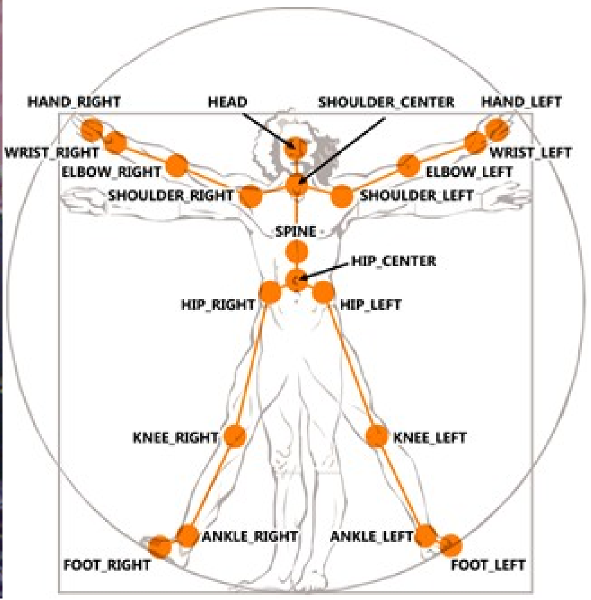
Augmented reality is a process by which data from the real world (for example video and object positions) are mixed with computer generated objects to produce something which is a combination of both worlds. The Kinect is a very useful tool for working with augmented reality as it provides high quality information about the world around it.
The sensors of kinect are illustrated in the following diagram.



The Kinect is sold as the device that makes "you" the controller. It does this by introducing a new kind of camera that could view a scene and measure the "depth" of objects in front of the Kinect sensor bar. This allows the software in the controlling device to build an image of the scene in front of the sensor. This can detect the position and orientation of players in front of the sensor and build a virtual "skeleton" that describes where they are and how they are standing.
There are two elements to the Kinect depth camera. The first is the Infra-Red projector that projects a field of dots onto the scene in front of the sensor bar. The second element is an Infra-Red camera that views the dots in the scene. The position of the dots in the scene the camera views depends on the position in the scene from which they were reflected.


Above you can see two images of the same scene. The left hand image shows what the Kinect video camera sees. On the right you can see the pattern of dots that the Infra-Red camera sees. Some of the dots are bright “alignment” dots; others are used for more precise positioning.

Kinect Skeleton Tracking

A program can use the depth information from the sensor to detect and track the shape of the human body. The Kinect SDK will do this for our programs and provide skeletal position information that can be used in games and other program.
The skeletal tracking in the Kinect SDK can track six skeletons at the same time. For four of the bodies only simple location is provided but two will be tracked in detail. For those two bodies the SDK will provide the position in 3D space of 20 joint positions. The software will infer the position of joints if it is not able to view their position exactly, for example of the user is wearing clothing such as a skirt that obscures their limbs.

Kinect SDK

Software can be developed on Windows PC that uses the Kinect sensor by installing the Kinect for Windows Software Development Kit (SDK). It contains drivers for both managed (C# and Visual Basic .NET) and unmanaged (C++) applications that want to use the sensor. We used Visual studio, specifically Visual C# 2010 Express. We used Microsoft XNA Game Studio 4.0 which makes the task of making Video games, and in our case 2D and 3D objects quite easy.
It can be coded in Visual C# Express and integrated with our Kinect functions. This is very convenient for us and we decided to pursue our project along these lines.

We add the necessary references regarding Kinect SDK and XNA to the project to use pre-defined classes.
Event handlers can be defined for events such as ColorFrameReady, DepthFrameReady, SkeletonFrameReady, and AllFramesReady ,for functions which are executed when the program gets new input from the color camera and depth camera.
One can also get Data on the depth of every pixel,and whether a human player is present on that pixel or not.
One can specify various parameters for skeletal tracking as follows:

```
//Code Starts
kinectSensor.SkeletonStream.Enable (new TransformSmoothParameters ()//EDIT SMOOTHING PARAMETERS OF SKELETON STREAM
        {
          Smoothing = 0.5f,
          Correction = 0.5f,
          Prediction = 0.5f,
          JitterRadius = 0.05f,
          MaxDeviationRadius = 0.03f
        });
//These are slightly different from the default values.
```

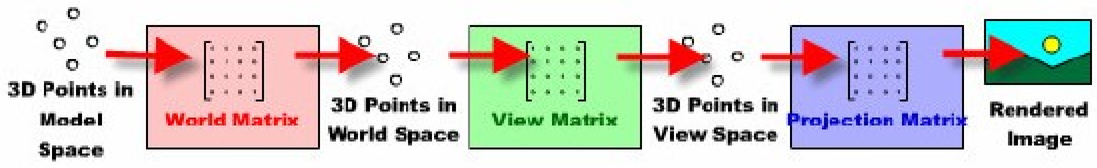
The image frame from the camera is rendered as a 2D texture and displayed onto the screen. We are using a resolution of 640 ×480.

Overview

We tried out a basic gesture at the beginning-clapping. A 2D hat appears on the lower hat when we clap out hands,and disappears when we clap again. The hat can be made smaller or larger when we come closer to the kinect or move further away.
We can transfer the hat from one hand to the other when the hands come close, or from the hands to the head.
We need the hat to be 3-D to look realistic,and incorporate a perspective view.We downloaded a few 3D models,and learned a bit of Autodesk 3DS Max to make our own models.

World, View, and Projection Matrices

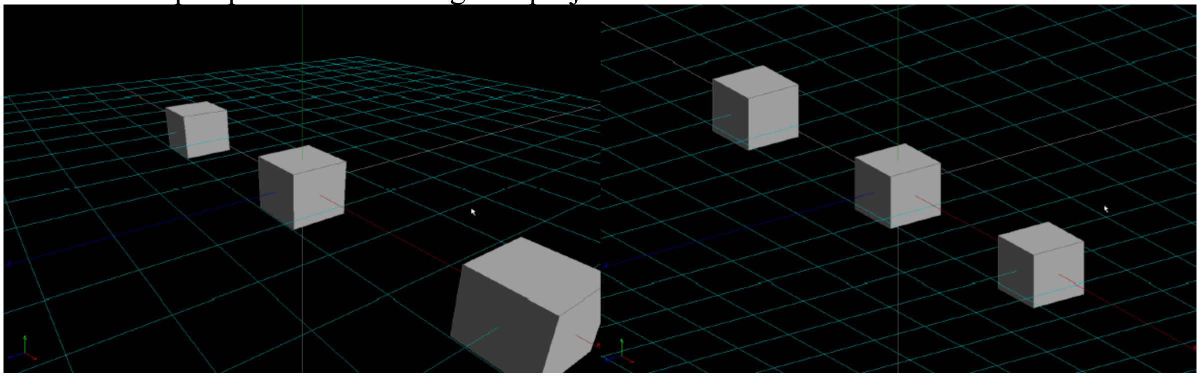
3D models require a lot of processing with matrices. When we go to draw a model on the screen, there are typically three different transformations that need to be done. There are three different matrices that correspond to these three transformations. The standard transformation path looks like the image below:



In model space,the coordinates of our vertices represent where they are in relationship with the rest of the model. At this point, we apply the world matrix. This transforms our model space coordinates into world space. World space coordinates represent where the vertices (and the entire model) are in relationship to the whole world. The world matrix basically tells us where the model is located in the world.

This next transformation is the view matrix. This matrix will put our coordinates into view space, which is where the vertices are in relationship to the viewer. That is, where the player's camera or eye is located at. In a sense, this transformation tells us where the player is located in the world.

The third and final step of the process is to apply the projection matrix. This matrix essentially tells us what type of camera we are using. It gives the computer the information it needs to determine where on the screen each of our vertices should appear. (It "projects" the points onto the screen.) This transformation essentially gets the vertices into screen coordinates, although the computer still has more work to do as it draws the model on the screen (like rasterization).
We can have perspective and orthogonal projection matrices.



Perspective views are realistic.Orthogonal should be used if we need to reduce processing,or if it doesn’t make a noticeable difference.

Our Approach

•Get skeleton data and manipulate positions of Joints for tricks

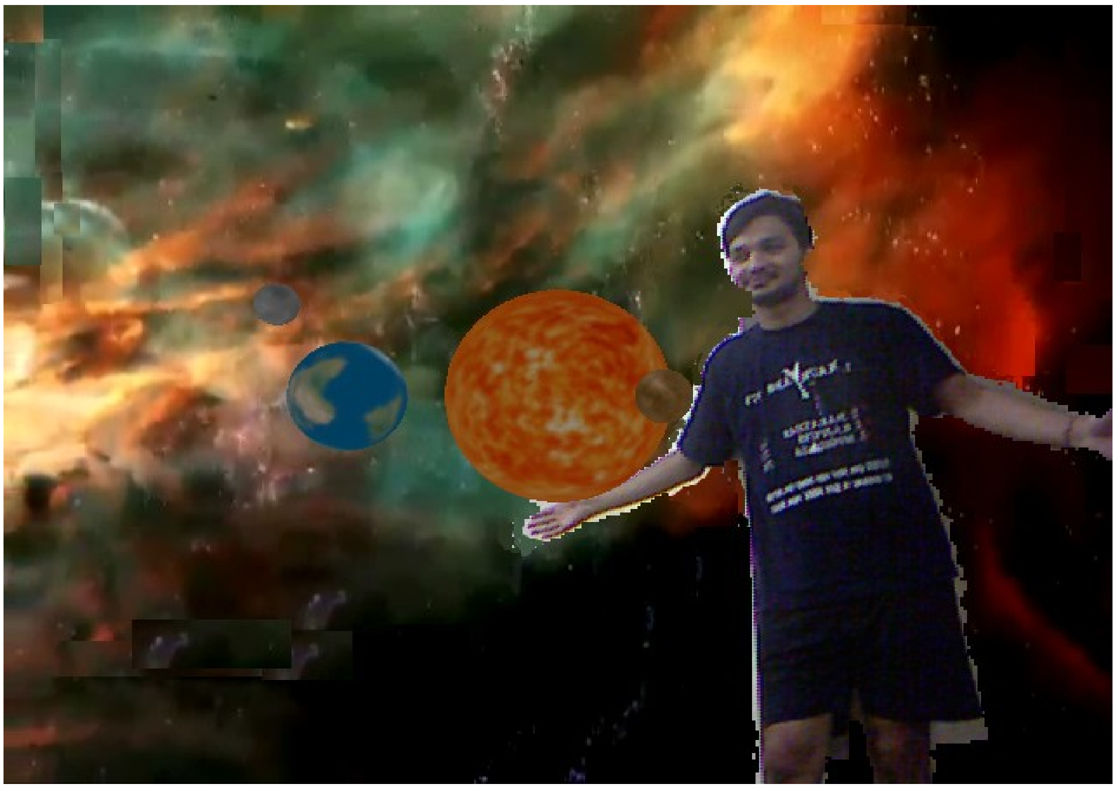
•Using gestures like clapping,waving of hand,etc.

• Use coordinates of tracked points for displaying objects at appropriate place and rescaling them

•Use depth data to uniquely detect the player if required

So we have a 3D hat, and we added different tricks like taking a ball out of the hat, and putting it back in.The hat can be made to move according to the depth of the joints.
Or we can change the size.This can be done by multiplying the world matrix by a scaling or translation matrix.

We can also have interfaces where only the player is shown in front of a different background. And right now we have to include the “background change” by simple gesture like swiping our hands in front, vertically, but still we are facing a tough time in doing so.
We added a solar system kind of stuff, comprising of the sun and 3 planets along with the earth’s moon. And the gesture we defined is with a magic wand, which is based on the distances of the hand with the kinect in multiple frames.The wand rotates according to the how you are 'holding' it.
Each time the gesture is recognized,a new planet appears.



The Drawing fuctions places objects in the order the functions are called.
So a object which is farther away will be shown on top of a closer object if we draw the further object after drawing the closer object.
We had to make an array and perform some sorting function to decide the order in which to draw the planets.
The animation is implemented by changing the world matrices incrementally in the Update function, which means that we continuously changed world matrix in subsequent frames so that it appears like the animation.
No matter how much we tried to incorporate Voice commands into our program, we just weren't able to make the program work. We did manage to use voice commands in simpler programs. We are thinking of incorporating this feature later.

The Final Show

- ➔ At the beginning,the head,left and wright wrists and shoulders have bright balls on them.These five joints are being tracked.Touch the other three joints with your hands,and they disappear.
- ➔ Clap and the show begins.A stage appears.Only the player is shown,rest of the scene isn't.
- ➔ Whats a magician without a hat? The demonstrator says this while bringing your right hand to your head.
- ➔ Clap your hands.The hat appears on your lower hand.
- ➔ Clap again.The clap disappears!
- ➔ Clap,the hat appears.It can be transferred from one hand to the other.
- ➔ Wear the hat on your head.
- ➔ Take it off.Pull out a toy rabbit.You can put it back in the hat.

- ➔ Take out a ball from the hat now.Put it back in.
- ➔ Wear the hat.
- ➔ Now lets go above the clouds.Clap,and you are among clouds.
- ➔ To start playing the game take your right hand to your left shoulder.
- ➔ Six clouds at a time appear on the screena nd drift rightwards.you can burst them with your right hand.
- ➔ If you've had enough,put your left hand to your right shoulder.
- ➔ The game has ended,but you are still among the clouds.Take your right hand to your left shoulder,and the screen turns to normal video,with you wearing the hat.
- ➔ Take off the hat.Take your magic wand out from the hat.
- ➔ Wave the hat with a very fast forward-backward motion.The sun appears,and the hat disappears.
- ➔ The Earth,moon,mercury and venus appear on subsequent waves.
- ➔ You can sen your solar system up away from view by raising your hands above your head.
- ➔ Now clap to have your stage back.
- ➔ It was a great presentation.Take a bow to have Thank You displayed on the screen.

Note:Clapping doesn't necessarily mean that you clap.It refers to bringing your hands close.

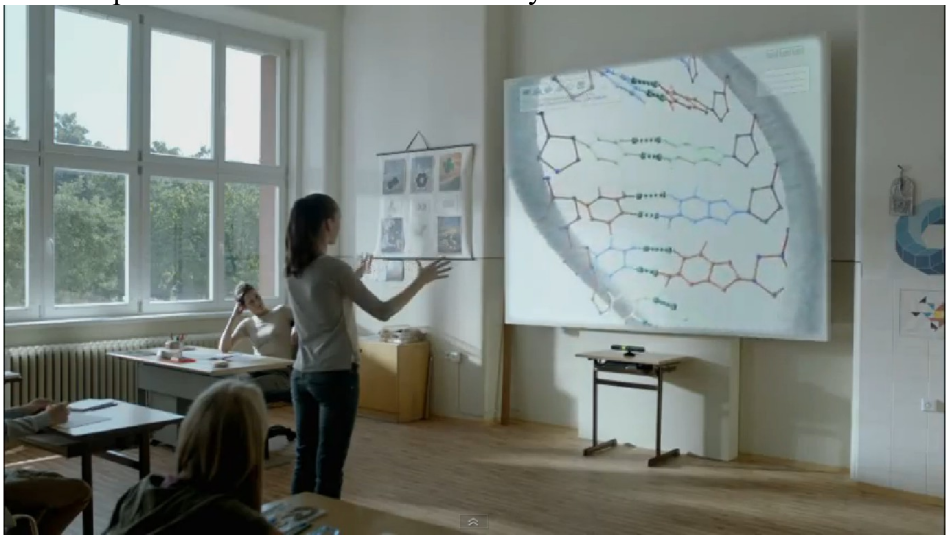
Further developments and Future Scope:

1) Enable voice Commands.
2) Enable better gestures , and possibly store gestures as XML files and have advanced processing for better gestures.
3) Add sparkles and other tricks to lengthen the show.
4) Create a two-player game, and a 3D virtual world.

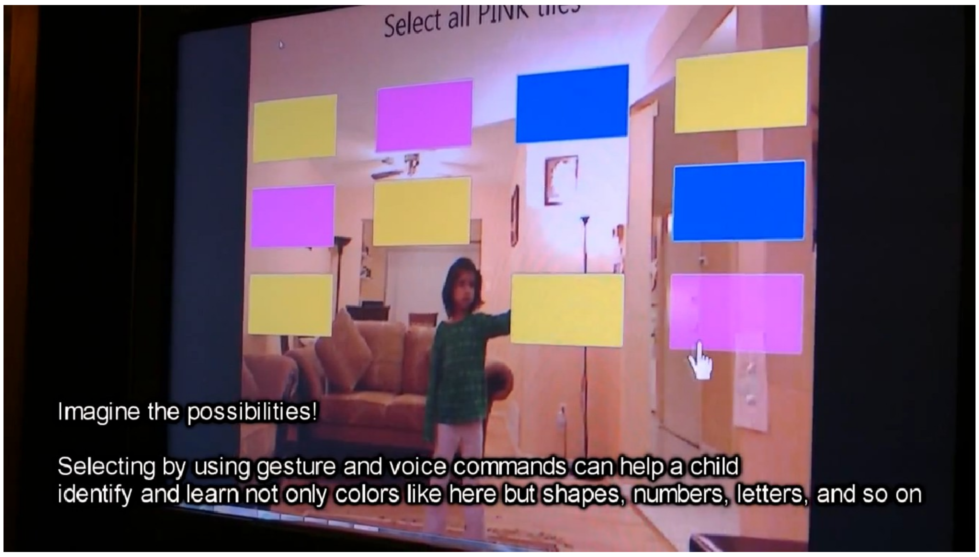
Utility

Kinect can be used to transform the way people innovate, discover, and interact with their world.

1) Kinect can be used as a teaching aid in schools and colleges by instructors.3D models and complex molecules alike can be analyzed with ease.



2) Kinect for Windows speech and gesture recognition capabilities were used to simplify complex learning processes and develop ways to make early childhood education more fun and interactive.



Imagine the possibilities!

Selecting by using gesture and voice commands can help a child identify and learn not only colors like here but shapes, numbers, letters, and so on

3) Companies are exploring new ways for doctors to navigate MRIs and CAT scans with the wave of their hands.

4) With Kinect for Windows, a Swivel virtual fitting room can be created ; a program which allows users to select and try on clothing all from their Kinect for Windows system.

Useful Links:

- EBook for basic 2D application with Kinect and XNA: http://channel9.msdn.com/coding4fun/kinect/The-Purple-Book-Using-Kinect-for-Windows-with-XNA
- Kinect For Windows QuickStart Series: http://channel9.msdn.com/Series/KinectQuickstart
- An XNA tutorial: http://rbwhitaker.wikidot.com/xna-tutorials
- A good book on C# for beginners: http://www.robmiles.com/c-yellow-book/

A word of thanks

We would like to thank our mentors Anurag Dwivedi and Nikhil Gupta along with Rudra Pratap Suman and Ganesh for their guidance, patience, suggestions and their believe in us . They inspired us to learn a lot and work on this exciting project and checked the progress of our project so sincerely that even if , sometimes, when we were like, frustrated and hopeless regarding our project they showed us the way.
Moreover, it was fun doing this project in first years summers as we got to learn so much, we actually learned a new programming language and a set of new tools in it for creating exciting games.