Members:
Anubhav Ashok
Suvamsh Shivaprasad
Sai Avala
Tucker Long

Interacting with augmented reality objects using kinect and unity

Description:

The purpose of this project is to be able to use the data from Kinect in order to interact with augmented reality objects that are projected by the Unity3D framework. An example scenario would be is if you're sitting at a table and the camera is pointed from the top. Using the data from the camera, we would project augmented reality objects using Unity3D. An example object that we might interact with is a coffee cup.

Objectives:

- Objective: Use Unity to create the virtualized version of Earth's globe
  - Result: A unity game environment that has a single empty sphere
  - Result: A sphere that contains portions of Earth's map (North America, EMEA, Asia Pacific), so that the glob looks like Earth
- Objective: Integrate Kinect with Unity
  - Result: A Unity game environment that shows Kinect video streams (depth stream)
  - Result: A Unity game environment that displays Kinect skeleton stream
- Objective: Use Kinect to interact with game objects in Unity
  - Result: The Earth globe created in unity is interactable by game markers. We can successfully move the globe around with our hands 90% of the time
  - Result: Using the Kinect, we can zoom in on the globe

Ambition:

- Let's say we get the above objectives working. Our final demo aims to be able to use Unity3D to insert a globe onto a surface. We can then interact with the globe by spinning it around, and hopefully switching back and forth between streetview and satellite view projections within the video. We would insert Google streetview images into the video once we've pinpointed a location to zoom in on and we would use Google Earth as the globe that we would interact with.

Updated Schedule:

- Schedule is based on each objective

Schedule: Use Unity to create the virtualized version of Earth's globe

Date	Activities	Deliverables
Nov 3	Install Unity and KinectSDK on our computers	Development environments should be set up

Schedule: Integrate Kinect with Unity

Date	Activities	Deliverables
Nov 10	Create unity objects and plan implementation	Create sphere in Unity
Nov 13	Make sphere object look like Earth - What we want to do is use the MapBox API to map the static tiles onto a 3d sphere	The sphere resembles Earth (80% similarity)
Nov 14	Now need to integrate Kinect with Unity - Kinect Depth and Skeleton videos should be shown - Display the Depth and Skeleton streams separately - We can detect the Kinect on our computers and can run the samples provided by Microsoft's Kinect SDK	Read in Kinect Streams

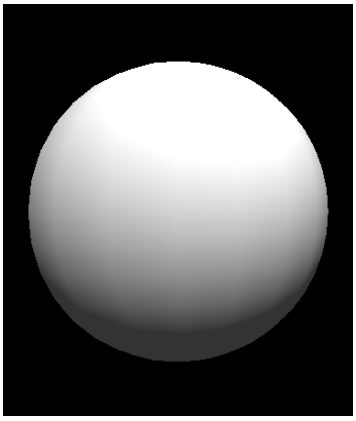
Schedule: Use Kinect to interact with game objects in Unity

Date	Activities	Deliverables
Nov 16	- Create game markers in Unity - These game markers are recognized by our hand movements in front of the Kinect and will be used to move objects around (object can 100% freely move around the game space) - Identify our hand at a minimum 80% of the time - Detect the depth stream from kinect to match collision of hand with the game	We can use Kinect to interact game markers in Unity Kinect detects object markers in Unity

	marker	
Nov 18	- Set the Kinect marker to be the Earth Globe - Use Kinect depth stream to detect Earth Globe upon collision between game object - Recognize both hands. Given a frame both hands must be detected 100% of the time.	Spin the globe by using the Kinect
Nov 20	- Kinect frame detection of both hands attempting to zoom in (like a swimming effect) - As before, both hands must be detected within a frame 100% of the time. - The globe should scale in size as we zoom into it	Zoom into the globe and out of the globe Finish Progress Report 2
Nov 21	- If we zoom into a location, the globe should grab streetview image of the location - Generate another moving effect to move around the map	Zoom into and out of the location
Nov 22	- Finish final report and finalize code implementation and upload to Github Repo	
Nov 23	- Practice demo. We will demonstrate our ability to spin the globe, zoom into and out of a location, and move around the location after zooming in completely	

Progress Report 1

Unity set up



Our first mini-goal was to set up the Unity environment and create a sphere that would represent our globe. We installed all the Unity dependencies and got started with a few basic Unity tutorials to get oriented to the environment.

Fig 1. Empty sphere in 2d

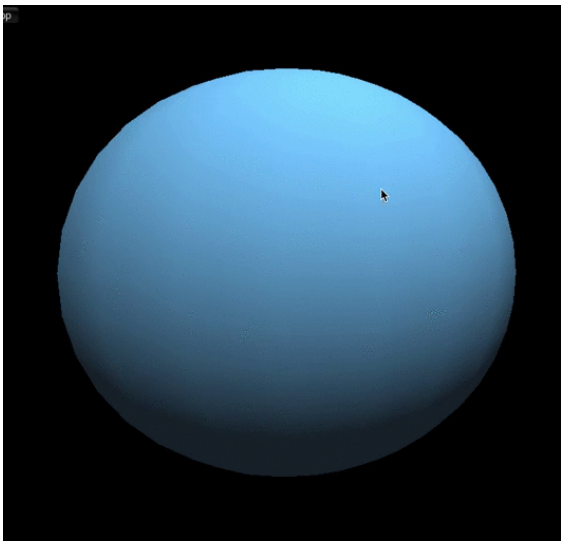
MapBox API

Next we played around with the MapBox API and wrote a Unity script to pull static tiles from the MapBox API and map them to the sphere.

We realized that if we zoomed in, we would have to pull multiple tiles to match the granularity of the map, but we decided to continue with getting the Unity interactions set up first.

Fig 2. Sample MapBox tile

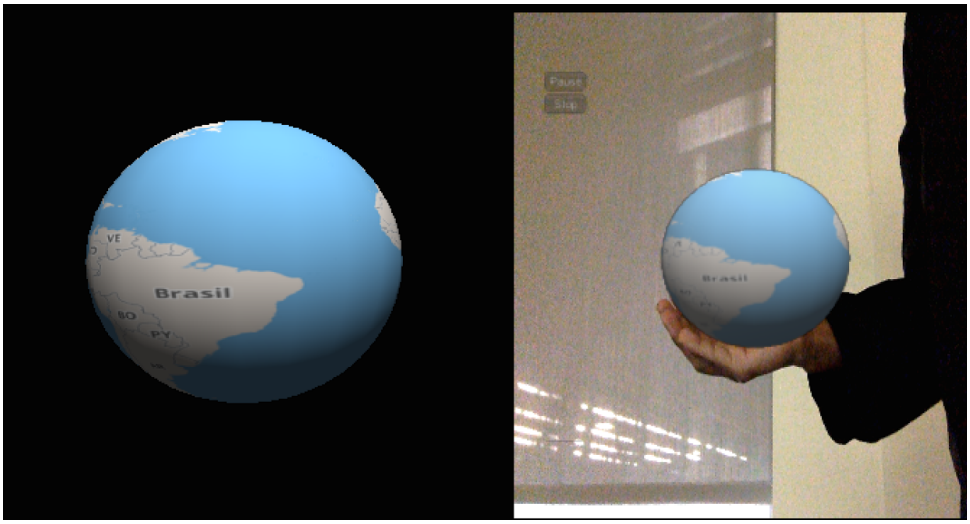
Unity Interactions



Next we added some simple key bindings to the globe to test out the full dimensions of Unity as we hadn't yet got the Kinect up and running. Furthermore, we could use these key bindings with Kinect directly, once it was ready. The gif to the left shows example movements.

Fig 3. Example movements

Webcam Test



Next we overlayed the globe on a webcam video capture stream by rendering the globe on the left half of the screen and pasting it (with transparency) on the video stream.

Fig 3. Globe overlay on video

Next Steps

Our next steps would be to get the kinect to interact with our 3d globe and render the result on the screen. We would also have to figure out a couple things such as how the texture would map when the globe was zoomed into a region and test the speed.

Challenges

We actually faced many challenges during the initial process of the project. For the first objective, we had to install and setup our development environment. This includes being able to recognize the Kinect on our computers as well as installing Unity so we could include game objects for us to interact with. In our case, it was the Earth Globe. The detailed list of challenges are below:

- Challenge: Setting up our development environments
    - The problem was that we all had Apple Macbook computers and we soon realized that the official Kinect SDK is only supported on Windows machines. We started by installing Windows VMs, but this process took much longer. We started resolving this problem by setting up third party drivers for our Macbooks by installing OpenNI, SensorKinect, and Nite to resolve this issue. And, we were able to write the C# scripts that were needed to write the code that is used to interface with the Kinect by using the Unity3D Mono Development Environment.
- Challenge: Creating the game objects in Unity
    - The problem that we had with this is that after creating the sphere and laying images of the Earth over the globe, we struggled with coming up a way in order to spin the globe around. As the Earth spinned we noticed that the globe wasn't the images weren't spinning smoothly. Essentially what happened was that we could see the frames/lag as Earth spinned around. To get this done required a simple modification in the Unity script.