The New Hotness: BLE Locative AR
A Proximity-based Augmented Reality Drawing Tool

CS488 - Computer Graphics

Anthony Perritano (aperritano@gmail.com)
Brian Herman (brianherman@gmail.com)

12/12/2014

Table of Contents

Introduction

When the final project was announced The Team (Brian and Tony) decided to take the opportunity to do something ambitious, their proposition was as follows: Using the immense computer graphics knowledge they gained in class, go about creating a project that uses state of the art in-door location tracking technology as markers in an augmented reality drawing application for an iOS device. Users would set down iBeacons – small micro-locative proxemic devices that can be detected with a Bluetooth connection – in a space, these beacons serve as points in which when coupled with the application the user can draw objects and apply different shaders to them. During the proposal phase of this project there were naysayers. Ones who doubted the feasibility of not only an augmented reality drawing tool, but also the feasibility of using iBeacons as a marker within that tool. This paper will discuss the relevant research around this area, the proposal and outcomes of this experiment, the challenges we faced, and the methods we used to overcome those challenges.

Technological Foundation

What is Augmented Reality? The concept behind Augmented Reality (AR) is that it couples (augments) computer generated two-dimensional or three-dimensional graphics with a real scene (reality). This coupling extends the kinds of interactions visually and physically a user can experience within a space. AR differs from Virtual Reality (VR) in that the user is not completely immersed in a virtual world but instead is immersed in a real physical space with an added extra visual layer. In the first AR system — the Knowledge-Based Augmented Reality Maintenance Assistant (KARMA) [2] — users donned a clunky head mounted display to interact with the system (Figure 1).

Figure 1. KARMA [2] the first augmented reality system was used with a head mounted display

Today there is no expensive equipment; AR is available to everyone with a common camera-based smart phone (Figure 2).

Figure 2. Augmented Reality map applications are common on camera based smart phones today [5].

Our project is a mobile Augmented-Reality (AR) drawing tool application that leverages current advancements in indoor location tracking technologies — Bluetooth Low Energy (BLE) iBeacons.

What is an iBeacon? IBeacons are small unobtrusive devices (Figure 3) that add contextual intelligence to a given space. Some types of beacons have extra sensors such as accelerometers or temperature sensors [8] and can be 'stuck' on surfaces, people, things, artifacts, etc. This is the concept behind the Internet of Things (IoT), which is when everyday objects become ‘smart’ by being instrumented with sensors or some physical-digital augmentation. A place could be outfitted with iBeacons, which could broadcast information at specific points within that space. Users could receive that micro-locative information seamlessly as they move through the space via BLE on their device.

Figure 3. Tear down of an iBeacon (Estimote version) [9].

Related Work

In Gauglitz et al. [3], they use augmented reality in a remote collaborative drawing tool. The paper describes the use of depth inference for 2D annotations in a 3D space and gestures that help facilitate that interaction. When a user annotates the space with a drawing within a real scene, the system is able to adjust the depth and position of the rendered graphical overlay as the position of the mobile is changed. The authors go on to describe multiple mathematical estimation techniques for these adjustments (e.g., "Spray Paint", "Dominant surface plane", etc.). In addition to these techniques, the authors describe a number of gestures that are unique with this type application and the different interaction affordances they allow. What is really important here and how it relates to our work is the use of drawing in real-time with an AR system and the types of techniques needed to preserve perspective within the rendered real-virtual scene as the user moves within the space.

The Gravity [4] sketch is a 3D augmented reality drawing under development by a design group called Gravity from the UK. It's system is composed of a head mounted display and a pen tablet interface. With the headset on, users draw objects in 3D space with the pen as the tablet creates a 3D grid background for object positioning (Figure 4).

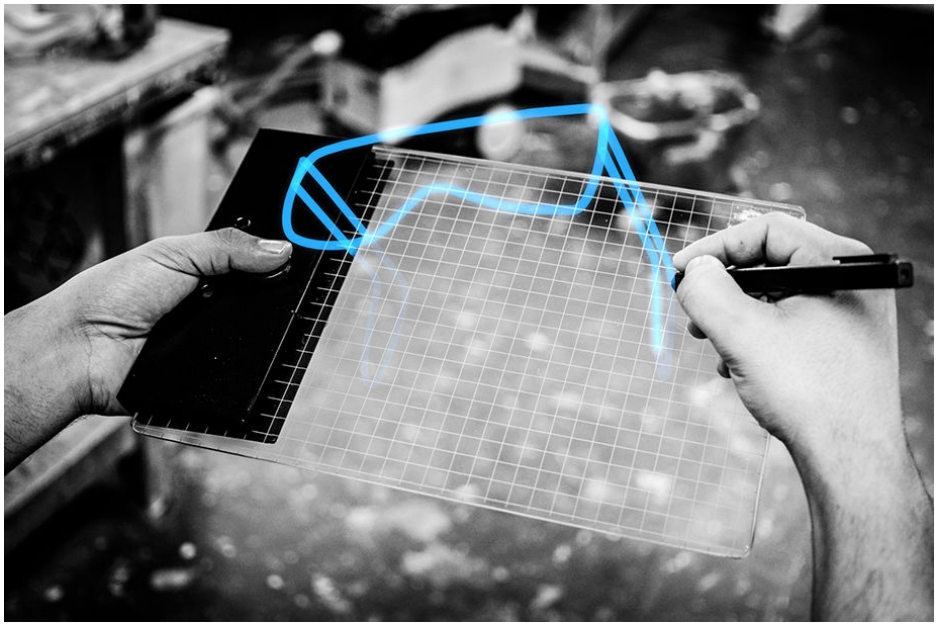

Figure 4. Gravity 3D pen based drawing system [4].

Gravity is important because it provides us with an exemplar of an augmented reality drawing application that is using 'natural' drawing interaction to construct 3D virtual objects in real space. Even though our application is different (drawing in a personal space vs. drawing in a public space) it gives us some inspiration what an advanced system using beacons good be – one could imagine using beacons within a space as object anchors, using a pen to draw within the scene shapes or used as pivot points in object rotation (akin to Blender).

Another interesting reference is Chen et al [1], they present work on how to apply non-photorealistic rendering (NPR) to an augmented reality scene. Their goal is to render an AR scene in a 'painterly' way, otherwise known as brush strokes. The paper describes how tensor fields – a complex mathematical concept used to describe vector components – are used on each frame of the scene to render the brush strokes in the appropriate direction and magnitude. This paper aligns with one of the main goals of our work, which is the application of shader techniques to an AR scene. However, our difference is that we would use NPR on the actual virtual objects in the scene versus the entire scene as describe in the paper.

How can iBeacons be used with Augmented Reality? Normally AR applications use QR codes as markers or computer vision for object detection to display virtual representations in a scene. In our case iBeacons will take the place of those techniques. We can envision a use case where a user places a number of iBeacons in a space at different lengths apart. Depending on the spatial arrangement of the beacons the AR system will draw a virtual line or a shape representation on the scene as the user holds up their mobile device. The shapes will be rendered in three-dimensions with an applied shader (e.g., non-photorealistic rendering (e.g. NPR, phong, etc.) which the user will have the affordance of choosing.

Project Proposal (Original)

A representation (Table 1) of the work needed to complete the project:

Step	Description
1. Choose an iOS-based AR toolkit to use for implementation.	There are a number of iOS-based AR toolkits to choose from open-source to paid proprietary. Our criterion is that it is open source, has an active development community, and a low learning curve.
2. Create an iOS base implementation from a toolkit example.	As part of the learning curve, we will create our own iOS project, create an interface and take one of the examples out of the toolkit and try to integrate it in.
3. Render objects and lines within a scene from QR codes.	Rendering the virtual objects within the scene is at the core of this project. Before adding the complexity of iBeacons, we can solidify the drawing algorithms using QR code markers as a stand in mechanism.
4. Add shader mechanisms to render objects.	In this stage we will give the user the ability to switch shaders on the rendered objects in the scene.
5. Add beacon micro-location support.	The final stage of development, we will use algorithms to figure out beacon distances and spatial arrangements to use as inputs to render objects. If we have done our jobs correctly in step 3, we can appropriate the algorithms used with the QR code markers to take beacon inputs.

Table 1. Project proposal plan (original).

Project Proposal (Modified)

This table (Table 2) represents a re-orientation of goals during development:

Step	Description	Comments
1. Choose an iOS-based AR toolkit to use for implementation.	There are a number of iOS-based AR toolkits to choose from open-source to paid proprietary. Our criteria are that it is open source, has an active development community, and a low learning curve.	After evaluating a number of toolkits such as Qualcomm's Vufuria we decided to go with a web-based version that is accessible from the desktop. This version was more accessible and allowed us to get up and running fast. The toolkit is called JSARToolkit (https://github.com/kig/JSARToolKit). JSARToolkit is a javascript version of the ARToolkit. The big change here is that it runs in a browser on a desktop or laptop fixed with a camera. Mobile platforms are not supported.
2. Create an iOS base implementation from a toolkit example.	As part of the learning curve, we will create our own iOS project, create an interface and take one of the examples out of the toolkit and try to integrate it in.	We were able to take some of the basic examples in JSARToolkit and turn them in to a project.
3. Render objects and lines within a scene from QR codes.	Rendering the virtual objects within the scene is at the core of this project. Before adding the complexity of iBeacons, we can solidify the drawing algorithms using QR code markers as a stand in mechanism.	We were able to successfully add this functionality to the project.
4. Add shader mechanisms to render objects.	In this stage we will give the user the ability to switch shaders on the rendered objects in the scene.	We have a number of shaders that can be applied to the object being rendered.
5. Add beacon micro-location support.	The final stage of development, we will use algorithms to figure out beacon distances and spatial arrangements to use as inputs to render objects. If we have done our jobs correctly in step 3, we can appropriate the algorithms used with the QR code markers to take beacon inputs.	Adding iBeacon support has been a challenge. These challenges are highlighted below.

Table 2. Project proposal modified.

iBeacon Challenges

Going into this project the idea of using an iBeacon as a marker seems like a really doable thing. However in practice it does not work well, we didn't anticipate that getting accurate distance would be so troublesome. The application needs a stable distance in order to render the scene. The difference between using a QR based marker and iBeacon is that application is not using computer vision to identify an object in the scene, but instead the beacon is broadcasting its signal and the application is reading that and determining the distance from itself to the beacon using the signal strength. The main problem with this approach is the signal strength fluctuates based on a number of variables that we do not have control over. For example, signals could be reflected or obstructed by materials such as metal, glass, concrete, WIFI signals, mobile phone signals, microwave ovens, and even some external monitors [6].

One way to get around this fluctuation is to detect an average distance and categorize them ordinally into a 'zone'. The beacon community has some established definitions such as unknown, far, near, and immediate zones (Figure 5).

Figure 5. Beacon zones [7].

Once the distance has been averaged and the zone determined the software can throw out any outliers and use that average in the rendering of objects. When the zone changes a new average will be computed and the representation with be re-rendered to reflect the new distance.

Using the Tool

The following is a walk through in how to use the tool:

1. The user opens a web browser on their computer and types in address “localhost:8000”. The page will load with the camera.
2. The user replicates (physically or digitally) a number of markers (Figure 6) that serve as vertexes.

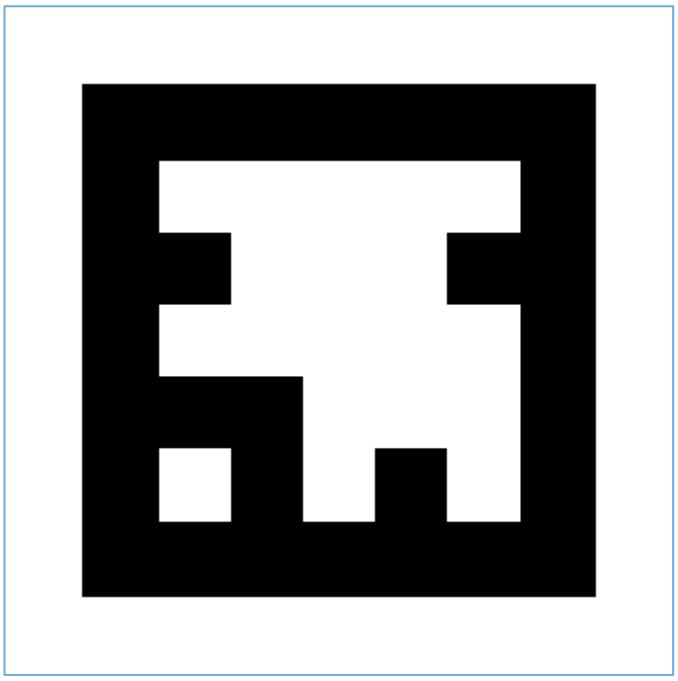

Figure 6. Marker for drawing

3. The user places those markers on a surface or in front of the camera to make a shape. For example, a user can make a triangle, a square, or many other shapes depending on how many vertexes they have.

4. Once shapes are drawn the user can toggle on or off a number of different opengl shaders (e.g., NPR, phong, etc.) on the rendered object.

5. Repeat as desired.

Breakdown

Demos (See Movie):

1. The first demo is of a sphere with a wireframe and a shader that just makes every object the color yellow.

2. The second demo is of a sphere that has been transformed with perlin noise.

3. The third demo is of a sphere that has been textured, transformed with perlin noise and has a time element that changes the sphere.

(Brian) The project does not successful draw lines between markers.
The following code does not work.

```
for(var i in markers){
loop through all the markers
        var m = markers[i];
        if (!m.model) {
          m.model = new THREE.Object3D();
        }
         geometry = new THREE.Geometry();
if a model matrix exists extract x and y
         if(m.model){
           var x = m.model.matrix;
           var y = m.model.matrix;
Push the vertices to the geometry of the shape.
           geometry.vertices.push(new THREE.Vector3(x,y,-50));
         }
      }
If there is a shape draw it!
      if(geometry){
        geometry.faces.push( new THREE.Face3( 0, 1, 2 ) );
        geometry.computeFaceNormals();
        m.model.matrixAutoUpdate = false;
        var triangle= new THREE.Mesh( geometry, new THREE.MeshNormalMaterial() );
        triangle.position.z = -50;
        scene.add(triangle);
      }
```

Conclusion

While the project did not work out as intended it was great to be able to explore this space. At twenty-five years old, augmented reality is still conceptually in its infancy. Meaning it's a powerful tool that has

the potential to shape the way we see the world, but the technology still needs to be developed so that it fits more tightly in our lives.

References

[1] Chen, J., Turk, G., and Macintyre, B. Painterly rendering with coherence for augmented reality. *VR Innovation (ISVRI), 2011 IEEE International Symposium on*, (2011), 103–110.

[2] Feiner, S., Macintyre, B., and Seligmann, D. Knowledge-based augmented reality. *Communications of the ACM 36*, 7 (1993), 53–62.

[3] Gauglitz, S., Nuernberger, B., Turk, M., and Höllerer, T. In touch with the remote world: remote collaboration with augmented reality drawings and virtual navigation. ACM Request Permissions (2014).

[4] Gravity. Gallery - Sketch in 3D space using Augmented Reality. *gravitysketch.com*. http://gravitysketch.com/gallery.html.

[5] MOTT, R. Imaging Notes Magazine. *imagingnotes.com*, 2012. http://www.imagingnotes.com/go/article.php?mp_id=322.

[6] Santos, A. Potential Sources of Wireless Interference. *estimote.com*, 2014. http://community.estimote.com/hc/en-us/articles/200794267-Potential-Sources-of-Wireless-Interference.

[7] Sullivan, J. Apps That Know Where You Are: Our Experimentation With Apple's iBeacon Technology - Inside the Nerdery. *blog.nerdery.com*, 2013. http://blog.nerdery.com/2013/11/nerdery-labs-ibeacon-experiments/.

[8] Team, E. Nearables are here: introducing Estimote Stickers. http://blog.estimote.com/post/95382199590/nearables-are-here-introducing-estimote-stickers.

[9] *estimote.com*.