AR Codes for Immersive Learning
EE 496 Final Report

Semester: Spring 2019

Authors: Cristina McLaughlin, Quinn Methered
Faculty Advisor: Dr. Darren Carlson

Abstract: The objective of this project was to explore creating a browser based augmented reality immersive learning experience. Augmented Reality (AR) refers to technology that overlays information or virtual objects on real-world scenes in real-time. AR can introduce new methods of teaching, change the location and timing of studying, and make the learning process more engaging. This project tests the extent of interactivity that can be achieved through A-Frame, AR.js and several other AR modelling libraries working together. AR-Codes are a combination of AR markers with QR codes that form a fast path to cross-browser experiences—without having to download an app or type in a URL. Our team has determined that mobile browser-based AR experiences are a viable, cheap, and efficient way to deliver AR to the classroom— without requiring excessive equipment or proprietary software.

1 Introduction

Augmented Reality (AR) refers to technology that overlays information or virtual objects on real-world scenes in real-time [1]. It is rapidly evolving and has high potential for a variety of applications—as it's uses are not just limited to gaming. Currently, there are implementations of AR technology in fields like travel, interior design, and cosmetics. Education is another major field that is currently being affected; AR can introduce new methods of teaching, change the location and timing of studying, and make the learning process more engaging.

Smartphones are the ideal platform for AR in education due to portability, size, and availability. According to a nationally-representative survey, released in 2015 by Pearson, 53% of elementary school students, 66% of middle school students, 82% of high school students, and 86% of undergraduates use smartphones regularly or own one [2]. In addition, most students want to use mobile devices more often in the classroom and feel like they know more about these devices than their teachers [2].

This project is focused on developing AR experiences for the mobile browser platform. There are three criteria that motivate development. First, experiences will be browser-based meaning users will not be required to download apps. Second, this project will be run on AR capable smartphones, meaning users are not required to have expensive hardware or access to proprietary software. Lastly, our team is going to focus on optimizing mobile interactions, resulting in user intuitive designs.

The final project report is organized as follows. Section 2 contains the objectives and evaluation criteria. Section 3 discusses related work, along with similarities and differences. Section

4 contains design tools, design methodology, and final design. Section 5 contains group member roles and what each member contributed. Section 6 discusses previous and concurrent coursework that relates to the project. Section 7 goes over engineering standards and constraints of this project and Section 9 contains our concluding remarks along with future work and subsequent development.

2 Objectives and Evaluation Criteria

The overarching goal of this project was to create an immersive learning experience for mobile browsers based on AR-Codes. We utilized *marker-based AR,* where the digital world is anchored to the real-world through a pattern the AR program can recognize. AR-Codes are a combination of AR markers and QR codes to quickly transport users to content without typing in a URL. With these parameters as our focus, we set several objectives as follows.

Figure 1. AR-Code example

The first goal was to program simple models and understand how the libraries we were using worked together. The second goal was to further explore interactivity within the AR-Code platform. This objective was evaluated on how smooth gesturing was such as tapping, pinching, swiping, and other touch screen interactions. The last phase moved into customizing multiple markers, having them run in a single program, and creating interactions between them. In the original proposal our long-term goal was to create a PowerPoint or video that shows off what the Ambient Lab does and enhance the message using AR codes, however due to several problems we encountered reaching this goal was not possible.

3 Related Work

There are several sources that relate to this project. AR.js is a JavaScript library written by Jerome Etienne that focuses on AR for the web [4]. It is fast (up to 60 frames per second on relatively old phones), web-based, opensource, and works on any phone with webgl and webrtc. There are examples of projects using AR.js and A-Frame along with some tutorials online. Our project is like these examples because it be using the same libraries available. However, it differs in that there will be more of a focus on implementing interactivity as opposed to just rendering models.

Multiple educational AR apps have also come up during our research, but these differ from our project because they require downloads. However, they will be helpful to look at when we are planning designs. We have looked into the Expeditions app by Google which is an immersive leaning tool that allows users to explore AR objects from atoms to outer space [5]. After seeing the

minimalist examples provided by AR.js and A-Frame tutorials, and the complex models in downloadable applications, our project has aimed to find a happy medium between these.

4 Design Tools, Design Methodology, and Final Design

Design tools required for this project have been split into two categories: hardware and software. The hardware requirements were an AR capable smartphone—we used a range including an iPhone 5 SE, Google Pixel 2, and iPhone 8—and a laptop for programming. The software used included IntelliJ IDEA, which has a built-in web server for previewing and debugging online applications. This feature was essential to project development because it made testing fast and easy. Other software included GitHub and GitHub pages for hosting documentation and the final webpage for each AR experience. The main libraries used were A-Frame, AR.js, three.js, and hammer.js. A-Frame is a web framework used in building HTML based virtual reality. AR.js is a JavaScript library that builds off A-Frame and brings AR to browsers. Three.js is a side library used in both A-Frame and AR.js for rendering 3-D models and hammer.js is a library that recognizes touch gestures for web-apps. Our team also used TinkerCAD and Blender, which are 3-D creation suites, to customize project prefabs. Lastly, we also relied on Discord for it's voice and screen-sharing capabilities. One member of the team would code while screen-sharing, while the other would provide input over voice chat.

Next, we will discuss the design methodology and problems that occurred during each stage of the process. The project was completed very systematically. We started with small building blocks, and slowly expanded on those; this ensured that we would never stray too far from content that we already understood.

The first step was to explore AR.js and A-Frame further; there are several tutorials we found to expand our understanding. We committed about 10 hours to reading through documentation and gathering examples, blog posts, and helpful websites. Our team has varying ranges of experience with HTML and JavaScript, so Quinn did extra research and followed tutorials using the Tutorialspoint website. Cristina also spent extra time learning about 3D modelling and researched how to create AR prefabs. During this stage we also ran into our first major problem. Most code from the tutorials were not working properly, such as this codepen example. We printed the AR markers to test the examples, but the AR objects would not show up.

We spent many hours trying to debug the problem. Quinn thought it was because the printer was not printing the AR markers dark enough to be recognized. We used a laser printer to reprint markers which did not work, and then used a picture of the marker on our phones to get the black as dark as possible, which failed as well. After debugging we narrowed the problem down to the script:

```
<script src="https://cdn.rawgit.com/jeromeetienne/AR.js/1.6.0/aframe/build/aframear.js">
</script>
```

RawGit is used to share example code or test pages in GitHub without having to set up a full static site. However, RawGit is shutting down and the above script no longer works, which has broken many useful AR.js examples. After discovering this we had to take extra time to figure out how to set up the scripts properly and include the correct libraries. Cristina spent several hours creating the GitHub environment so that the HTML files would read the framework scripts. She worked backwards by downloading the original AR.js GitHub repository and narrowing down

necessary files by deleting folders and finding out what broke the code. This section of the project required a lot of trial and error, but we were eventually able to get everything running properly.

The next stage in our design focused on programming introductory AR examples. The three main programs were a basic cube, basic sphere, and basic scene. The code for each of these was straightforward and understandable. After rendering them from localhost, we needed to test accessing them through QR codes. We created a repository ARdemos and built a GitHub Pages site from the master branch, that way the webpage has access to the HTML code for each demonstration. Figures 2 and 3 show the documentation page and an example of a demonstration page.

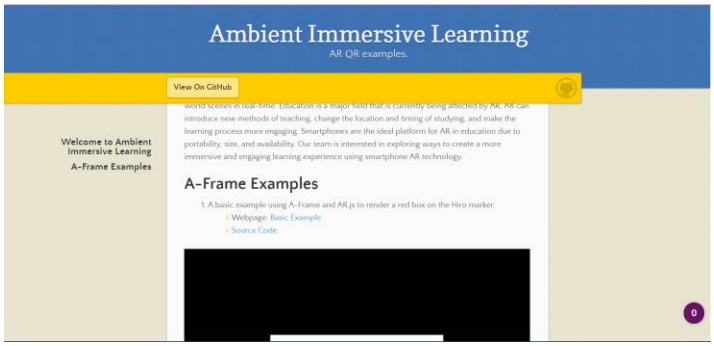

Figure 2. Webpage for documentation.

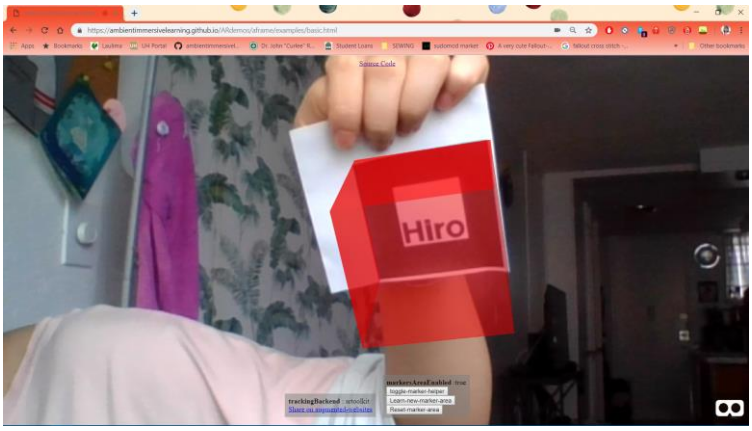

Figure 3. Basic-cube.html being hosted through GitHub pages.

The nect step was to create an AR-Code for the experience. During our research we found that it is better to print the QR code within the AR marker, rather than the other way around. This is because AR markers need to be large so that AR.js can estimate position with higher accuracy. In addition, QR codes do not need to be very large so it makes more sense to position the codes like in the left image of Figure 4, rather than the right. Jerome Etienne hosts an AR-Code generator where a marker with a QR code is generated after inputting an HTML webpage.

Figure 3. AR-QR codes

After getting comfortable with the frameworks and being able to launch a simple AR model, we have moved to exploring the interactivity within AR.js. Our main objectives were tapping, scaling, dragging, and implementing animations tied to each gesture. Our team realized that any object interactivity would require JavaScript scripts to handle mouse events. We did some research and found that A-Frame has its own cursor feature that became available in v0.6.1 by setting <a-scene cursor="rayOrigin: mouse">. Unfortunately, according to the API there are no hovering/hovered or mouseenter/mouseexit states for mobile. This limited how users can interact with the object being rendered.

After programming more examples, it became clear to our team that relying solely on A-Frame and AR.js API was not sufficient for a smooth mobile experience. We conducted more research to find ways of better handling touch gestures and found hammer.js. Hammer is an open-source library that can recognize gestures made by touch, mouse, and pointerEvents. The library made is easy to subscribe to different touch events such as pinching, swiping, and rotating.

The final stages of our design focused on testing multiple marker interactions, which including creating custom markers and running them within the same program. Creating custom markers was more difficult than expected because there are limitations on size and shape. After conducting extra research, we learned: markers must be square, markers cannot have white/transparent areas, and markers must be simple but recognizable. Marker icons are transformed into .patt files that are encoded into greyscale and then read by AR.js. This is also known as marker training; AR.js has a simple marker training website.

We created a first set of markers based on words, but they failed. The icons were too detailed and just through observation one can see the encoding is not recognizable. Figure 4 shows our first failed marker and it's corresponding .patt file. We created a second round of markers to train, relying more on simple icons. This batch was successful and the encoding, seen in Figure 5, is recognizable even to the human eye. We created ten custom markers based on different simple icons.

Figure 4. Failed marker and encoding

Figure 5. Successful marker and encoding

The next step in the design plan was implementing multiple markers in one demonstration. We decided to have each marker tied to a different colored box. Programming was relatively straightforward, each marker is declared in the scene and a link to the .patt file is included as shown:

```
<a-marker preset="custom" type="pattern"
url="https://raw.githubusercontent.com/ambientimmersivelearning/ARdemos/master/markers/bow.patt">
      <!-- Add your augmented reality here -->
      <a-box position='0 0 0' material='opacity: 1; side:double; color:red;'>
      </a-box>
   </a-marker>
```

When implementing this we ran into a problem where the program would not properly load the .patt file if we included the file from the directory. Switching to loading it from the raw GitHub source fixed this issue. When running multiple markers, we expected the shapes to distort easily and for the AR.js tracking to not handle them sufficiently. However, we were surprised that performance did not decrease at all with ten markers in the same scene.

Figure 6. Multiple markers running together

The last goal in the design methodology was having markers interact with one another. We found an example through research where the distance between two markers could be measured. We tweaked the example by customizing it with our own markers using Bow and Check. The example used the three.js library, which we were not heavily exposed to in previous demos, so much of our time was spent reading through the code and trying to understand it.

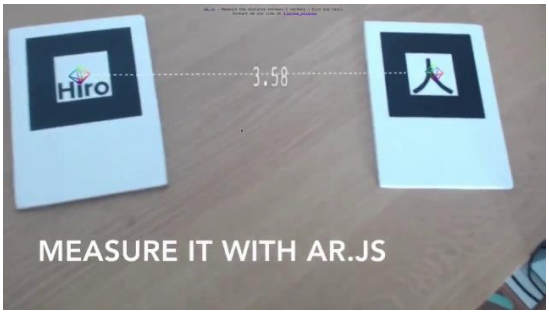

Figure 6. Measuring the distance between markers

Based on this interaction we also tried to implement a chemistry demonstration. The idea was to have different atoms tied to induvial markers, and when they were in close range of one another they could combine into new molecules. We had a function checking for a distance < 2 between two markers and then the marker would update. However, the rendering is not working properly. Our rendering function is below:

```
onRenderFcts.push(function(){
        var length = markerRoot1.position.distanceTo(markerRoot2.position)

        if(length < 2){
            var newcolor = new THREE.Color("rgb(0,102,255)");
            markerRoot1.material.color = newcolor;
            markerRoot1.material.color.needsUpdate = true;
        }
    })
```

The program freezes when the markers are within 2 units of one another, so we know the length is registering. However, when we were testing a simple color change the material does not update. There are issues with updating the object material, and we were unable to get this working by the end of the semester.

The original final design in the project proposal was meant to be an immersive learning experience, but the problems we encountered during the project kept us from completing it. However, the foundation that we set up during this project was successful. We were able setup the AR environment working, optimize gesturing through hammer.js, implement multiple markers, and begin exploring multi-marker interactivity.

This project is unique in that are not currently many alternate solutions that we could use to create the same type of experience. While there are other open source AR libraries, none are as

well-maintained and up to date as AR.js. We could have tried alternate AR routes like using the Hololens or Vive for VR, but it would defeat the purpose of accessible AR.

5 Team Member Roles

Our teamed worked together most of the project besides during research where we went through tutorials separately. Cristina was the designated team leader. She designated simple projects for Quinn to work on because this project was his EE 296. She created documents and tutorials for him to follow. She also moved forward and began programming for the more complicated interactions including using hammer.js gesturing and setting up the multi-marker examples.

Quinn spent the first part of the project learning HTML and researching JavaScript. During the semester he conducted research to pull examples, ideas, and solutions from the web. He worked on the first basic AR.js examples and helped with documentation.

When coding the AR demonstrations, we both tried to work on them together. We both feel that code is more understandable if we write it together as opposed to committing things separately to GitHub, and in addition Quinn did not have access to a reliable computer that can run the IntelliJ IDE. When coding, Cristina would screenshare with Quinn so he could understand and provide input when necessary.

6 Previous and Concurrent Course Work

As stated earlier, Cristina has more exposure to software due to being in Computer Engineering, while Quinn only has coding experience from EE 160: Programming for Engineers and EE 491F: Internet of Things. Cristina has taken ICS 314 which is a software engineering class

that relies on JavaScript; this class provided a lot of experience with GitHub pages that has proven useful when we needed to host the AR HTML code. ICS 314 also leaves class documents available to prior students, so she provided these to Quinn as a stepping stone into learning JavaScript. Cristina has taken EE 491E: Multimedia Programming which has given her background in event handlers and designing prefabs. This prior knowledge made it easy to understand how to implement clicking and zooming. EE 342: Introduction to Statistics was also helpful when calculating the amount of combinations that were required to designate interactions between multiple markers.

Quinn has relevant experience in troubleshooting both software and hardware from EE 160 and EE 491F. When we ran into the scripting problems, he compartmentalized each piece of the project to determine the problem. First, he started with obvious physical factors like a clouded camera lens or the AR marker, then moved to the code and checked each script link to see if it was working. EE 491F also provided him with basic experience in JavaScript.

7 Engineering Standards and Constraints

This project follows engineering standards and constraints, the following considerations are made: economic, environmental, sustainability, manufacturability, ethical, health and safety, social and political.

Economic

This project correlates to the education industry which is valued at about 1350 billion dollars in 2017 and is expected to rise as education becomes entwined with technology. The U.S. government has already granted a total of $430,000 to developers who want the EdSim Challenge

which explores ideas of AR in education. While this project has no economic impact currently, its base ideas are relevant to the education industry.

In addition, we did not make any purchases for the project itself. All software we used was open source and our code also remains public on GitHub.

Environmental and Sustainability

This project did not involve the environment or relate to sustainability issues. In the future an immersive learning project could be programmed to relate to either of these topics.

Manufacturability

There was no manufacturing involved in this project. The system was fully based on software and programming. In future iterations, it would be beneficial to print the markers on plastic rather than relying on sheets of paper.

Ethical

There are some possibilities for ethical problems relating to the project. Some could use the open source code to create AR simulations of inappropriate content, however we have avoided this by keeping all demonstrations created professional. We also created AR demonstrations to enhance the teaching of facts. This project does not require human subjects because it is still only exploring the development stages.

Health and Safety

This project has no impact on health or safety. It relies on the usage of a smart phone, but it is the operator's prerogative to understand any major health risks that come with mobile devices.

Social and Political

Education and social issues are integrated. It is well known that richer districts often have better public schools and technology than those of poorer districts. This likely means that any AR technology implemented in the education system will go to schools that have the budget for it. This project is beneficial in that it does not rely on having expensive hardware like the Vive or HoloLens. If AR in education is more web-based than hardware based it will be available to a broader range of students.

8 Conclusion

In conclusion, this has been a very enjoyable and interesting project to work on. here were basic components that made it perfect for an EE 296 project, and more complicated problems that made it EE 496 appropriate. AR-Codes are a viable, cheap, and efficient way to deliver AR to the classroom— without requiring excessive equipment, proprietary software, or extensive knowledge of technology on the part of the teacher. Many of the available Mixed Reality frameworks are inaccessible to programming laymen/educators interested in developing simple, customized classroom activities. A-Frame and AR.js combat this by using basic HTML with minimal setup. With additional time and development this platform could be at the forefront of bringing AR into the classroom, adding it to textbooks, or including models on assignments.

In the future, subsequent developers can build on what we have as foundation. One of the main goals should be to finish the chemistry demonstration by figuring out where the rendering is erroring out. An interesting experience could also be having different circuit elements linked to each marker and having them create a circuit when linked together. Future designs could also include a

UI. The physical interactions between the user and the marker sets this type of AR apart from any other.

References

[1] The Franklin Institute. "What's the Difference Between AR, VR, and MR?" The Franklin Institute, 29 Sept. 2017, www.fi.edu/difference-between-ar-vr-and-mr.

[2] Poll, Harris. Pearson Student Mobile Device Survey 2015 . Pearson, 2015, Pearson Student Mobile Device Survey 2015 , www.pearsoned.com/wp-content/uploads/2015-Pearson-Student-Mobile-Device-Survey-Grades-4-12.pdf.

[3] Etienne, Jerome, and Jerome Etienne. "AR-Code:a Fast Path to Augmented Reality – ARjs – Medium." Medium.com, Medium, 4 Apr. 2017, medium.com/arjs/ar-code-a-fast-path-to-augmented-reality-60e51be3cbdf.

[4] Jeromeetienne. "Jeromeetienne/AR.js." GitHub, github.com/jeromeetienne/AR.js/blob/master/README.md.

[5] "Bring Your Lessons to Life with Expeditions | Google for Education." Google Expeditions, Google, edu.google.com/products/vr-ar/expeditions/?modal_active=none.

[6] "A-Frame – Make WebVR." A-Frame, aframe.io/.