"SketchIt! Or Some Other Clever Name"
A Google Glass Application for Classical Studies

Kristin Muterspaw

December 16, 2014

Abstract

Augmented reality, the use of technology to enhance a person's senses in the real world, is becoming increasingly prevalent in society. I propose an augmented reality application, "SketchIt! Or Some Other Clever Name," designed for the optical head-mounted display, Google Glass. This application will be used to sketch models of ancient Greek structures and allow the user to view different angles and interact with the model. It can be used during travel, education, or simply to explore an interest in ancient buildings. The application will be written using Googles Glass Development Kit in conjunction the Wikitude software development kit, a set of tools designed specifically for augmented reality on Google Glass, which calls external JavaScript files for rendering the models. As Google Glass becomes more widely used, I hope to expand the application to include many other ancient sites. Once complete, it will be published in the Glassware Gallery, the collection of available applications for Glass, with the hopes of being utilized by students, educators, adventurers and more.

1 Introduction

Think for a second. When you are walking down the street, through the aisle on an airplane, around a restaurant, etc., what percentage of the individuals are looking at their edge devices? I would guess somewhere above seventy percent. Society is so concerned with being connected all of the time, which makes it difficult to hold conversations or interact with people. Another reason individuals are constantly on their hand-held devices is to lookup information on Google, browse social media, and get directions to name just a few. Depending on the device being used, augmented reality (AR) can help solve this problem.

There is no singular meaning behind the words augmented reality. Many people define augmented reality in their own terms, and a subset of people do not know what augmented reality is. Others define augmented reality as a "technology that enriches the sensorial perception of a person" [DVPR14]; Krevelen and Poelman describe it as a way to "supplement the real world with virtual (computer generated) objects that appear to coexist in the same space as the real world"[VKP10] . Augmented reality boosts the real world. It is the use of technology to amplify or enhance the sights and sounds (possibly smells, touch, taste) of a person's existence in the real world, allowing those that use it to experience more than those that do not [DVPR14]. Augmented reality is different than virtual reality. Augmented reality takes the existing environment and enhances it whereas virtual reality replaces the real environment with a completely different one, making it seem as if the person is existing in a completely new world [KCS14].

Augmented reality has already been used in a variety of different fields and continues to be implemented into new areas constantly. These areas include education (which is where my application resides) for enhancing learning and creating new and exciting study tools; medicine, applications for remote mentoring and shadowing, and applications that will show more information about the body[Gla13]; museums, enhancing the different exhibits[CCH14]; infrastructure, applications that will display different air ducts, for example, when held in front of a building [SMK$^+$09]; and gaming, displaying the characters and gaming environment onto the physical world. The ability to have more information in an accessible and aesthetically pleasing manner can enhance almost any discipline.

1.1 Background

Augmented reality can be implemented in two different ways: video monitor augmented reality, which is the easier and less expensive technique[VKP10], and projector-based augmented reality. The information is displayed through some kind of screen for video monitor AR, such as displaying statistics over a video of a sporting event on the television. For projector-based AR, the desired information is placed onto a physical object, allowing the user to see the information directly with the object, without a screen.

These two different techniques can be designed and used with of a plethora of devices. In [DVPR14] they discuss the basic, but necessary, capabilities a device must have in order to successfully produce augmented reality: a video camera, an Internet connection, Global Positioning System (GPS), and an Inertial-Measurement Unit (IMU). Most modern hand-held devices, such as smart phones and tablets, possess these qualities. However, when one of the goals is to decrease the use of hand-held devices, using them would not suffice. Individuals using these devices could use a Google search engine to retrieve any kind of information they desire, without the need for augmented reality.

The other main type of implementation for augmented reality is using head mounted displays (HMD). There are two types of HMDs: optical head mounted displays and monitor based HMDs. Both devices are worn on the head and show the augmented reality (whatever that may be) in front of the user's eyes. Monitor based HDMs completely block off the user's vision of the real world while displaying a digitized version of it through the HMD, and typically supplying additional information about the world [VKP10]. With optical head-mounted displays, people are brought back into the real world. Instead of having their faces constantly stuffed into their hand-held devices, blocking their vision, optical HMDs lay information over top of the environment or onto a small screen which the user can see by just wearing the device and looking forward. Having information right in front of one's eyes allows for more interaction with other people – they can hold conversations while still doing everything else. Wearing these devices, the users also have information readily available to them right in front of their eyes. Figure 1 shows a graphical diagram of an optical head-mounted display. The information, the augmented reality, from the monitor is displayed through optical combiners, allowing the user to still see the real environment. Compare this illustration with Figure 2 which is a diagram of a monitor based HMD. In that Figure you can see that the user is only seeing what the monitor is showing; it is blocking off the real environment and producing a digitized version of it.

2 "SketchIt! Or Some Other Clever Name"

With the advancement of technology, walking through a museum is quickly becoming worn out. One of the greatest and most beneficial uses for augmented reality is in enhancing museum experiences. Universities and museums are beginning to experiment with integrating augmented reality into their exhibits. This would allow the visitor to, depending on the museum, interact with, learn more about, or be immersed into the exhibit or a particular artifact. Integrating augmented reality in museum experiences allows for a more informative, enjoyable experience. However, what if you are not in a museum but you still want to see an

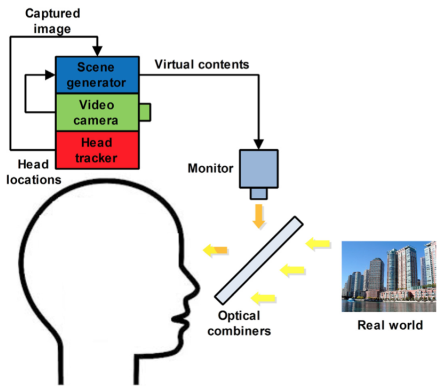

Figure 1: Optical See-Through HMD [DVPR14].

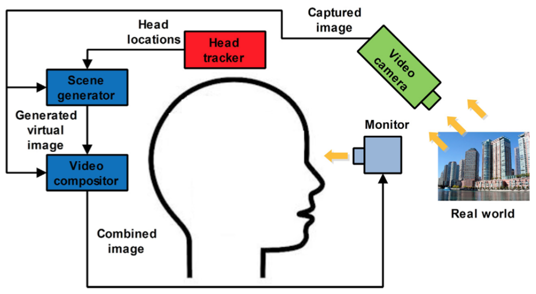

Figure 2: Monitor-based HMD [DVPR14].

object or structure and learn more about it?

To resolve this problem, I present "SketchIt! Or Some Other Clever Name." "SketchIt! Or Some Other Clever Name" is being created for use with the optical head-mounted display, Google Glass, and will be able to render 3D models of ancient structures in real-time. To begin, the glassware (technical name used for applications designed for Google Glass) will render the structures both what they looked like originally and what they look like today. The user will be able to choose whether they want to see modern day or original appearance. Google Glass responds to voice commands, and my application will take advantage of that feature. Wearing Glass, the user can open the application by saying "ok glass, sketch the... Parthenon," for example, and "SketchIt! Or Some Other Clever Name" will render that structure in 3D.

Throughout Ancient Greece there are hundreds of structures and sites to see. Some, like the Temple of Zeus at Olympia, are considered highly special – some would call them ancient wonders of the world. These structures are not only important to Classical Studies academia but also to the world as a whole. The ancient Greeks were involved in discovering many things society holds dear today – information about astronomy, physics, and math, as well as medicine and philosophy. Although, going to to visit these places in person is ideal, not everyone has the opportunity or means to travel. The application I have proposed will be able to bring Ancient Greece to almost everyone. Classical Studies professors would be able to use this in their classrooms to enhance their teaching of ancient structures and buildings, and students would be able to use it as a study tool.

3 Implementation

There are two principle components to my application: the hardware and software. "SketchIt! Or Some Other Clever Name" is designed specifically for Google Glass using an Android programming environment with Google Glass specific software development kits and libraries.

3.1 Hardware

Augmented reality requires specific hardware. There are fundamental features that the chosen device for implementing augmented reality (for successful implementations of it, at least) must possess: GPS, for tracking the user's precise location within the world; a camera, in order to augment whatever the user is looking at; an Inertial Measure Unit (IMU), for tracking when the user moves the device; and the ability to make an Internet connection. An Internet connection is useful when rendering the images. For most devices, in particular HMDs, battery life and processing power is limited. Rendering 3D models requires a lot of power for the graphics, thus, draining the battery quickly. Having an Internet connection would allow the application to send all rendering requests to an external server, wait for a response in the form of a rendered object, and display that to the screen. Doing this, the device would maintain a longer battery life, which would allow for extended use of the application and device itself.

3.2 Google Glass

My application will be designed solely for use on Google Glass. Google Glass is an example of an optical head-mounted display. Optical combiners are used in conjunction with a monitor that displays the images and graphics to the user on a small screen. The device sits on a user's face just as a pair of glasses would. There is a small display that is placed over the right eye where the applications are displayed. Figure 3 is a photograph of Google Glass disassembled with all of the main parts labeled. The device has all capabilities mentioned above, which make it ideal for augmented reality – GPS, IMU, Wi-Fi interface, camera and monitor [Gar14]. Google Glass has technical features above the five fundamental ones just mentioned that

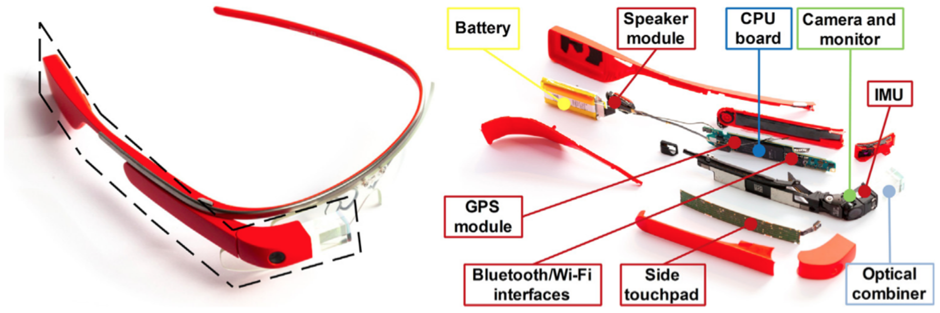

Figure 3: Disassembled Google Glass[DVPR14]

sets it apart from other devices used for augmented reality. Glass has a speaker module that allows the user to talk to the device in the form of commands, and vice versa — the device can also talk to the user. The voice command interface is only one of the ways that the user can interact with the device. If using the voice command interface is not preferable to the user, the touchpad on the side of the device can be used to navigate through the device. The touchpad can recognize patterns that the user has specified for some application – for example, there can be a lock on the device (similar to a lock on a cell phone) that can be unlocked using a pattern on the touchpad [Goo14c]. Glass has two other technical parts that sets it apart from other devices: Bluetooth interface and a slide touchpad for navigation through different parts of the Glass. Using Bluetooth, the device is capable of pairing with other Bluetooth capable devices (on set up the device is required to pair with a mobile device; this is used for downloading applications to the Glass).

As I mentioned earlier, most any "smart" device is capable of implementing augmented reality. There are a couple of reasons why Google Glass, instead of other edge devices, seemed to be the most appropriate device for this application. Google Glass is an optical head-mounted display, meaning it is worn on the face exactly like a pair of glasses. I did not want my project to contribute to society's inability to look away from their hand-held devices. So with this application running on Google Glass, a conversation can still be kept with other people, but most importantly the user will still be looking in front of them instead of into their hands. Other details that were considered when choosing the hardware were price, technical capabilities and availability. Google Glass has capabilities that set it apart from other devices, as mentioned above. It also has 16GB of flash storage and a fairly powerful processor. Google Glass is one of the only devices that is minimalist, open source and fairly inexpensive. It is currently only in the beta testing days. Individuals can sign up to be a part of the Explorer's Program, which is the group of testers for the device. Within a month, Google will contact the individual with information on how to purchase the device.The device costs $1500, which some may find expensive. However, if the use of the device is meant for the classroom, just having one pair would suffice. The students and professors would be able to pass around the device and experience it, instead of each student having their own device.

3.3 Software

Having chosen Google Glass as the hardware platform for “SketchIt! Or Some Other Clever Name,” there are specific rules that must be followed while designing my application. Glass applications are a subset of Android applications, which are all written using the Android Software Development Kit (SDK). The core language for Android applications is Java, and the Android SDK comes equipped with Android specific libraries for creating applications for the Android operating system (mostly libraries for dealing with user interfaces (UI)). The Eclipse integrated development environment (IDE) is the default for all Android development – it comes as an install with a plugin for Android called Android Development Tools [Goo14a] with the Android SDK download. Eclipse is just an environment that developers use to write their code. With the ADT plugin installed, a developer can create a new Android application and see all of the graphical interface widgets that their application contains, on a device with the size, make, and model of their choice.

Libraries and SDKs

In addition to the standard Android SDK, there are other libraries and SDKs that are used in the application. Google Glass is an optical head-mounted display which implies augmented reality, but because Google Glass is such a new device, there are not many documented uses or libraries of Google Glass being utilized for in depth/detailed augmented reality. There is one SDK solely for creating augmented reality applications to run on Google Glass, called the Wikitude Augmented Reality SDK for Google Glass [Wik14].

Wikitude SDK for augmented reality

The Wikitude Google Glass SDK is not a native Android SDK, but it is designed for use with Android applications and devices, of which Google Glass is an extension. Included in the SDK are example applications, code, and documentation on the different methods and classes used for creating augmented reality displays for Google Glass. it has the ability to do image recognition, 3D rendering, geolocating images, tracking of objects, among other functions. It works by allowing the Android application or Glass application to call an external library called ARchitect World. When the application makes a call to ARchitect World, it creates an instance of a type called architectView. The architectView instance controls the Android necessities on the device, such as sensor events (Gyroscope, Infrared, motion sensors) as well as displaying the images to the device using an Android type called SurfaceView. SurfaceViews are used for the user interface that will be changing often (for example a camera preview), and because the display changes when the user moves, using a SurfaceView is the best way to accommodate for that. When an instance of architectView is created, there's an external call to the JavaScript/HTML/CSS files which contain the information about the models or pictures, and controls what should happen given user input or movement. The SurfaceView is used only to show on the Android device what these files create. The JavaScript/HTML/CSS files can live within the actual Android application in the assets folder or on another server. Placing the JavaScript/HTML/CSS files on an external server would require more code but would likely consume less power consumption than would be consumed if they stayed on the device, as it takes a lot of power to render 3D objects. For my application, the files will initially reside in the assets folder for testing and early stages of the code, but in the future will reside on an external server. While using this SDK, the OpenGL library will also be used by the Android application to render the graphics to the display.

Google Glass GDK for Glass development

When Google first started the Explorers Program for beta testing Google Glass, they released a Google Glass Development Kit (GDK) [Goo14b] for those wishing to create applications for the device. The GDK was released as a preview, meaning the whole GDK is not available yet. The GDK comes with documentation and example applications for the developers to use as guides, as well as libraries that are used specifically for Google Glass. Google Glass applications use the data type: Card, Live Card, Scroll Cards for the user interface, and the GDK provides libraries that are used for creating those. In Figure 3, a flowchart of a sample Glass application, a Compass, is provided.

The individual black rectangles are examples of Cards. The first card for all applications of Google Glass is the “ok glass” card. “ok glass” is spoken by the user and the live card is changed to a scrolling list of different applications that can be started by voice commands; the user then speaks the name of the application they want to open, in this case it's the compass. The application is now open and the person can begin using the compass. Once inside the application, the user interacts with it by moving around or by tapping on the touchpad. On a tap, the user is shown a new Card that when selected, as in the case of the Compass application, can read aloud the direction the user is facing. Scrolling on the touchpad (sliding forward or backward) the user is presented with another Card option where the user can stop the application. Another option to open applications on Glass is by taping on the “ok glass” card. On a tap, the user will be able to scroll through different available applications and tap the one they wish to open. When doing that the user

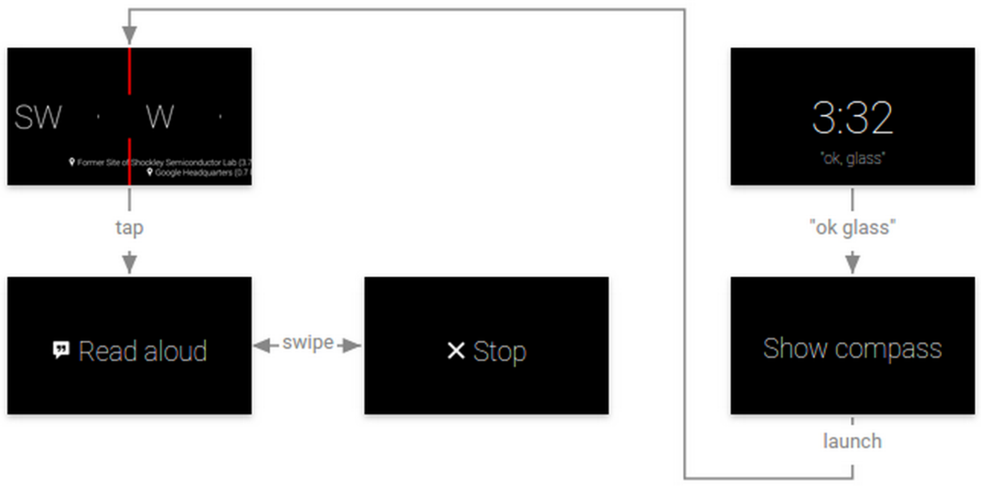

Figure 4: Flowchart of Compass Application[Goo14b]

will be taken to the same card that appeared when opening the application by voice command. The GDK that Google provides is full of libraries that allow working with these types of UIs possible.

3.4 "SketchIt! Or Some Other Clever Name" software

The algorithms are discussed in this section. Included at the end of this document is a detailed outline of them.

Android/Java Files

"SketchIt! Or Some Other Clever Name" has two sets of distinct software — Android and Wikitude/Architect World. Thus, there are two distinct algorithms that were created. The user can start the application by saying "ok glass. Sketch the..." On saying this, the user is presented with a list of all of the possible structures that the application can render; by looking up and down the user can scroll through the list. The structure to render can be spoken – for example, "ok glass. Sketch the... Parthenon." Like the sample compass application shown above, the user can also tap the "ok glass" screen and be able to scroll through the different available structures to render, using the touchpad to navigate through the application as opposed to the voice command interface.

When a structure is specified, by voice or tap, the application will then create an instance of architectView, using the JavaScript files in the assets directory of the application. The architectView sends out a request specifying the path of the HTML file that contains the name of the JavaScript file for the structure the user has chosen to render. Each structure has its own folder and set of JavaScript/HTML/CSS files. This request returns an instance of the architectView camera activity, so the 3D objects, returned from the ARchitect World call, can be displayed. A location listener, sensor listener, and accuracy variable of each is created when the architectView is executed. These are used to track the user's precise location, in which direction the user is looking, as well as the user's head movement. The model will look like it is stationary as the user turns around it (like the real structure would stay still as the person moved around it), but the software is actually reading code to turn the model. On every location change the listener will get the users location and re-render the structure as necessary. The orientation sensor is also calibrated at every interval of a certain number of milliseconds.

After this, the structure is rendered. Figure 5 shows an example of what it could look like to render the Parthenon. This is shown once the user says "ok glass. Sketch the... Parthenon."

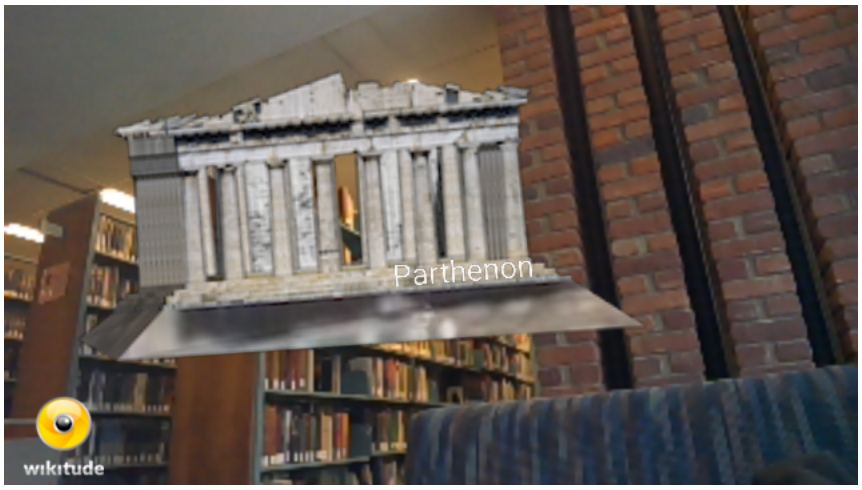

Figure 5: Rendering of the Parthenon

Notice the Parthenon was placed on top of the real environment – the user can still see everything in front of them. Once the user is presented with the structure, they can walk around and see the different sides of the structure, and if desired, ask to be taken inside. The user can go inside the structure by saying "ok glass. take me inside," or by tapping and selecting the "take me inside" option from the menu. The application would send a new architectView request for the inside of the structure. While inside the structure, the user would be able to turn left or right and the application would track their head movement to show whatever the user would be looking if they were physically inside the structure. This would involve sending multiple requests to the ARchitect View UrlLauncher, executing the same steps as the original render when needing to turn and show the outer sides of the structure.

Anytime during the use of the application, the user can tap the touchpad and be shown a group of scrollable cards each specifying a different option available. The different options include: take me inside, sketch something else, about section, or exit application. The user will have the option of enabling a feature that will display information about the structure while viewing it as well. This information will include date of construction, physical location of the object, date of destruction (if applicable), and possibly the architecture style and materials of the structure. To exit the application, the user can swipe down on the touchpad (standard for Google Glass applications).

ARchitect World/JavaScript Files

For the ARchitect World implementation, written in JavaScript, each structure is an array of information. The information about each structure is: the name, description (date built, location in the world, and destruction date if applicable), and an AR.Model object from the ARchitect World library. The AR.Model is a datatype provided by the Wikitude SDK that uses an existing 3D model file, created in some piece of CAD software, and then encoded into a Wikitude specific format, .wt3. The AR.Model type has different methods that will be used for scaling and tilting the object when the user is moving or looking around. There are also methods that will be used for specifying which direction the user is heading.

Each structure will also have an AR.Label attribute. The label will contain the different pieces of information mentioned before about the structure. This AR.Label will be used if the user enables that option. The JavaScript file will have three separate functions: one for head movement, one for information while viewing, and one for being inside the structure. For head movement the structure needs to be re-rendered even with the slightest movement. The function uses the Wikitude SDK object, AR.PropertyAnimation, which takes the current location of the structure and moves it East, West, North, South or a combination – depending on what direction the user is moving their head. The SDK has constant variables that will be used for the amount to move the object. This will use the methods of the AR.Model type for modifying the actual model object. For the information while viewing function, it will set the different attributes to the object.

The function for being inside the structure is slightly more complex. The function will use the AR.ModelAnimation object for animation. The model must already contain the animation when creating an instance of the AR.ModelAnimation. Therefore, I will need to either have images of everywhere inside of the structure, or import through the AR.Model object and animated model. Figure 6 is a flow chart of what it looks like when the user opens "SketchIt! Or Some Other Clever Name" and interacts using taps.

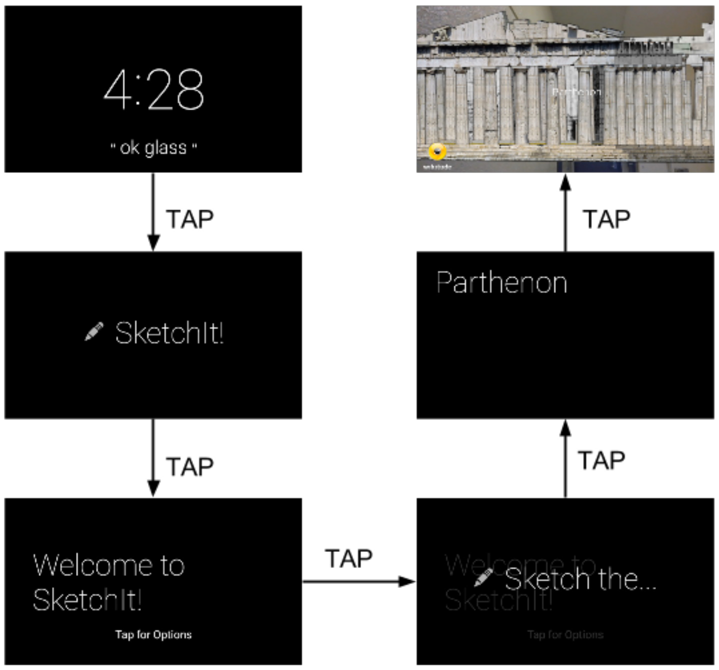

Figure 6: Flow of SketchIt!

3.5 3D models

For the AR.Model object, an existing model file is required. The models used in "SketchIt! Or Some Over Clever Name" mostly come from pre-existing models. These pre-existing models come from SketchUp 3D Warehouse, which is a collection of different 3D models created by the community using SketchUp CAD software[Nav14]. Using existing models limits the variety and amount of structures I can allow my application to render; the variety of structures is minimal as well as the detail of each. Only the well-known structures, like those on the Acropolis in Athens: Parthenon, Erectheion, Temple of Athena Nike – and other well-known structures like the Temple of Apollo at Delphi will have multiple models available. Modifications were required for some of the models, such as scale and removal of background or surface. To make those modifications, SketchUp Make was used.

4 Results

"SketchIt! Or Some Other Clever App Name" is currently capable of rendering images of 3D models. As the user turns, the images stays in the same position as when first rendered, which is what I was hoping to implement for turning around the structure. The user can scroll through the list of structures and choose one to render. However, there are a small number of available structures to render. The amount of pre-existing available 3D models is miniscule. I found that the Parthenon has the most pre-existing models available, and the Acropolis of Athens as a whole close behind. There are other structures that have a couple of models available but most of those are not to the detail that I want them to be. The Wikitude SDK, used for rendering, requires that the models be in a specific format. An encoder is provided with the SDK to convert 3D models in .dae or .fbx format to Wikitude .wt3 format. However, the encoder strips the models of their texture during the encoding. The detail and design and materials of the structure are a few of the most important aspects when studying it, so having no texture decreases how useful the application actually is.

Throughout my research I discovered that because Google Glass is such a relatively new technology, and therefore not available to everyone, there are limited resources available for creating applications for it. The SDK that I chose to use was the only one designed specifically for Google Glass. The kit that Google produces for development on the device is only a sample kit, with limited documentation and sample code. Google Glass's processing power is impressive for such a small device, but it is limiting when running augmented reality applications such "SketchIt! Or Some Other Clever Name." Complex renderings and visualizations do not function properly on Glass because so much work is needed for processing quality renderings. Several times when testing the application, Glass would overheat and require me to stop using it for a little while in order for it to cool off.

Augmented reality has proven to be useful in many different fields, and although I have not been able to test my application in a classroom setting, the success of augmented reality applications so far insists that the application will be beneficial.

5 Why is this important?

You may be wondering why any of this matters; why making a Google Glass application for use in Ancient and Classical studies is even necessary. Classical studies is the study of ancient things — art, language, history, archaeology. This subject can be learned by reading books and seeing images and watching documentaries, etc. but it is best learned by going and actually seeing the places and information learned. There are limitations to this, however. If a student outside of the country of which they are studying wants to to travel to Greece, for example, it is costly and time consuming. The means to travel to these countries is not universal. Using "SketchIt! Or Some Other Clever Name," the student would be able to experience much more than they would by just looking at pictures and reading books. This would enhance the professors teaching as well as provide an exciting, almost "hands-on" approach to learning Ancient and Classical studies.

This would make the entire field of Ancient and Classical studies more inviting and appealing to prospective students.

In addition to being used in classrooms, museums are another place where this application could be proven useful. Like use in the classroom, the museum would be able to integrate this application into their exhibits — giving more information and a new look to the structures. The application could be adapted for whatever use the museum is desiring — perhaps showing the structures in ruins (or how they look today). There are many different uses for this application in the real world; it will also be available to all Google Glass owners; having an application that is educational as opposed to the majority of applications which are designed primarily for entertainment.

6 Limitations/Drawbacks

Using state-of-the-art technology in any aspect and particularly in a field where technology is typically not used, does not come without limitations and drawbacks. There are limitations when introducing any type of technology into fields in the Humanities division; experts in those fields are slow to adapt new technology to their work, many believing that it is not needed in their field. This can be an obstacle when it is time to present my project to professors in Ancient and Classical studies.

Using Google Glass as the hardware for my application presents more obstacles. Society is split on their views of Google Glass (and HMDs as a whole). One half of society is excited to experience new technology and adapt wearing a computer on their face into their everyday life. The other half of society is highly against it — they call the other half "cyborgs" and are paranoid that the device is watching them. This half refuses to research what Google Glass actually does; for instance, facial recognition is banned in all Google Glass applications. If an application contains any type of facial recognition Google will not publish it on their store. It is true that Glass can take video and pictures but it cannot not do anymore than what a "smart" device can, such as a tablet.

7 Future Work

Ultimately, "SketchIt! Or Some Other Clever Name" will be published to the Google Glassware Gallery. There are many improvements that it must undergo before that can be done. As of right now, "SketchIt! Or Some Other Clever Name" is only capable of rendering images of the models of different structures. The actual models of the structures have yet to be successfully implemented. Allowing the user to be taken inside the structure will be implemented as well. This will allow the user to experience more of the structure and learn what is on the inside. Even being at the actual site in Ancient Greece may not be able to give the person this experience, as many of the structures are in ruins and therefore do not allow visitors to go inside.

Due to the limited number of existing 3D models of structures, there is minimal balance between original appearance of the structure versus appearance today. Ideally, I would like to be able to either create or find 3D models of more structures of both modern day appearance and original appearance, and allow the user to specify which appearance they would like to see. Each tells a different story and gives different information.

"SketchIt! Or Some Other Clever Name" could be used while physically looking at the site as well. In the future, I would like to add the capability of rebuilding a structure. For example, if the user is in Greece looking at the Temple of Zeus at Olympia (which is 70% in ruins) the user could say "ok glass. Rebuild this temple," and the "SketchIt! Or Some Other Clever Name" would reconstruct the temple by using image and natural object recognition. The application would be able to track the user's precise geo-location, determine that they are in Olympia, Greece at the site of the temple, and reconstruct it.

8 Conclusions

My application "SketchIt! Or Some Other Clever Name" is for use in the classical studies field — it enhances the learning and teaching of ancient structures. By rendering different ancient structures and displaying information about it, the user is learning more by being able to interact with the structure and visualize it in 3D. Wherein before, they would only be looking at some kind of picture. As technology advances, the details and tools used for augmented reality are making it available and beneficial for all disciplines. Google Glass proved to be the most capable device for this application – it uses state of the art technology and it is an optical head mounted display which allows for interaction with other people. Augmented reality is already dominating some fields, being used in the medical industries in hospitals and schools; in architecture and utilities fields when designing new buildings; and in entertainment with video games. This list continues.

References

[CCH14] Chia-Yen Chen, Bao Rong Chang, and Po-Sen Huang. Multimedia Augmented Reality Information System for Museum Guidance. *Personal Ubiquitous Comput.*, 18(2):315–322, Feb 2014.

[DVPR14] Pasquale Daponte, Luca De Vito, Francesco Picariello, and Maria Riccio. State of the art and future developments of the Augmented Reality for measurement applications. *Measurement*, 57(0):53 – 70, 2014.

[Gar14] Simson Garfinkel. Glass, Darkly. *MIT Technology Review*, page 70, Feb 2014.

[Gla13] Wendy Glauser. Doctors among early adopters of Google Glass. *Canadian Medical Association Journal*, 2013.

[Goo14a] Inc. Google. Android Developer Tools with Eclipse. http://developer.android.com/tools/help/adt.html, 2014.

[Goo14b] Inc. Google. Google Glass Development Kit. https://developers.google.com/glass/develop/gdk/index, 2014.

[Goo14c] Inc. Google. Google Glass Tech Specs. https://support.google.com/glass/answer/3064128?hl=en, 2014.

[KCS14] Sarah Kenderdine, Leith K. Y. Chan, and Jeffrey Shaw. Pure Land: Futures for Embodied Museography. *J. Comput. Cult. Herit.*, 7(2):8:1–8:15, June 2014.

[Nav14] Trimble Navigation. Sketchup. http://www.sketchup.com/, 2014.

[SMK$^+$09] Gerhard Schall, Erick Mendez, Ernst Kruijff, Eduardo Veas, Sebastian Junghanns, Bernhard Reitinger, and Dieter Schmalstieg. Handheld Augmented Reality for Underground Infrastructure Visualization. *Personal Ubiquitous Comput.*, 13(4):281–291, May 2009.

[VKP10] DWF Van Krevelen and R Poelman. A survey of Augmented Reality technologies, applications and limitations. *International Journal of Virtual Reality*, 9(2):1, 2010.

[Wik14] Wikitude. Wikitude Augmented Reality Software Development Kit for Google Glass. http://www.wikitude.com/products/eyewear/google-glass-augmented-reality-sdk/, 2014.

Algorithms and Data Structures, Software

For the main Glassware Application:

— User launches “Insert Clever App Name Here” by selecting the icon

- — User can also launch the app by saying, ”ok, glass. Let’s build something....”
- — Opening Screen is a list of all the different structures that can be rendered
  - — When a user looks up or down the list scrolls
- — User can either click that screen or saying one of the structures listes
- — On click of that screen:
  - — A screen will pop up that says something similar to “Scroll through to see all the structures that the app can render.”
  - — User can use touchpad to scroll forward to see an “About” section.
- — User speaks the name of a structure or selects one from the list
  - — Application sends a request as an architectView type to a server where all of my .js files are living. The request will send out a url of the specified structure to render. The urls are in a string array. There is a different url for each structure that can be rendered.

    Creating a ARchitect View calls UrlLauncher class which takes the url and returns an instance of ARchitect View camera activity.
- — Wait for a response from the architectView and display that to the screen.
- — Start location listener, and orientation listener. This will be used for tracking the user’s location and head movement. We want the model to turn as the user turns.
- — On location and sensor changed:
  - — Get user’s precise location and where their head is. Re-render the structure.
  - — At every interval of a certain number of milliseconds, calibrate the orientation sensor.
- — A possible aspect of the application that I’m contemplating implementing:

  If the user speaks the command, ”ok, glass. take me inside.”

  The application would send a new ARchitect View request for the inside of the structure.

  While inside the structure, the user would just be able to turn left or right and the application would track their head movement and show where ever the user is looking at.

  This would involve sending multiple requests to the ARchitect View UrlLauncher. And doing everything as the initial image, just doing it constantly.

— If the user taps the touch pad. Show menu of options which is different scrollable cards.

Different options:

Build something else, selecting this would send the user back the list of structures

Exit application, which would just close the application.

— Do everything above until the user exits the app by swiping down.

For the JavaScript ARchitect World:

— Each structure will be an array of information. The information will include:

— Name

— Description (date built, where it is in the world, destruction date is applicable.

— AR.Model object from the ARchitect World library, which is the model file of the structure.

— The models will be created using an external tool, most like Autodesk Maya or similar.

— Each structure will have a label, AR.Label. The label will contain a description of the structure if the user sets the option for displaying information about the structure while viewing it.

— Function function for head movement:

— This function will use AR.PropertyAnimation object which takes the current location of the structure and move it East, West, North, or South, or a combination. Depending on what direction the user is moving their head.

The Wikitude SDK has constant variables that I will use for movement of the user's head.

— Function for information while viewing:

— The function will just set the different attributes that are requested to the object

— Function for being inside the structure:

— This function will use the AR.ModelAnimation object.

— I will need to either have pictures of literally everywhere inside of the structure or import through the AR.Model object an animated model of the structures.

If the latter is possible, then that makes more sense to use because having 1000+ for each structure seems a little extreme.