TeachWithGlass:
Improving the Teaching Experience through Google Glass

May 1, 2015

David Correa Orozco ‘15
Advisor: Tim Kraska
Reader: Anastasios Matzavinos

Abstract

This project explores the use of Google Glass in an educational setting. The exploration was done through the development of the Glassware application TeachWithGlass. The application syncs with PowerPoint and presents to the user his or her presenter notes as well as LaTeX equations that are rendered as images. TeachWithGlass is still under development. Professors in the engineering and computer science department at Brown University have evaluated the effectiveness of the application and the results have been promising. Nevertheless, further testing is still needed to arrive to a concrete conclusion on whether additional features or fine tuning of current features is needed.

Table of Contents

1 Introduction

Google Glass was introduced as a prototype in April 4, 2012 [1]. On January 15, 2015 Google announced that the prototype production would stop and Google Glass would graduate to a different stage [2]. Throughout its status as a prototype, Glass was most effective when it was given a particular use in a specific field. One of the most successful examples is Augmedix, a company that uses Glass to streamline the data processing of medical institutions. The company has raised $23 million in funding to continue growing [3]. Another example where Google Glass has been effective is the story of GuidiGo, which specializes in offering guided museum course through Google Glass [4]. From these examples one can infer that the future of Glass is in the workspace. This project further explores this idea in a university educational setting at Brown University.

2 Motivation

I have a great passion for teaching. For half of my undergraduate career I have been a teaching assistant and a head teaching assistant for several courses in the computer science department. Throughout all of my experiences I have sought ways to improve the student experience and their retention of knowledge. Similarly, I share a passion for wearable technology. During my sophomore year I traveled to Budapest over the summer to take a course on mobile development. That course has greatly influenced my education of computer science and I am certain that it continue to influence my career path. With these two components in place, this project emerged as a way to improve

the student learning experience by empowering the lecturer through Google Glass. With the current design of most slide presentation software, if the presenter relies on the presenter notes, he or she is bound to the podium. Furthermore, if these notes have complex equations, the lecturer also needs handwritten notes that can be taken to the chalkboard. We envisioned building an application that would allow presenters to have all of their notes delivered when they need them. We thought it was possible to combine both presenter notes and equations to reduce the overhead of switching between the two formats. Ultimately, we wanted to free the speaker from the podium and from handwritten notes so that they could focus on teaching the students. If we were able to make the integration with technology as seamless as possible, we believed that the main beneficiary would be the student as a result of the better engagement that the professor will have with the class.

3 Goals

The original goals for this application included the following technical requirements:

- Display the presentation notes for each slide in Microsoft PowerPoint.
- Parse LaTeX equations and present them as images to the user as part of the notes.
- Prevent the device from going to sleep while there is an ongoing presentation.
- Allow the user to control the flow of the presentation from the Glass device.
- Maintain a full synchronization between the Glass device and the PowerPoint presentation. Any changes made in the Glass device must be

reflected on the presentation and any change in the presentation itself must trigger an update in the Glass device.

With these technical requirements our two main goals were:

- Evaluate the effectiveness of the application in an academic setting. This evaluation will be done by having professors use the device during a lecture and then having them assess the experience.
- Set the foundation for a program that can continue growing and improving based on the feedback received from various assessments.

4 Related Work

In the educational space, Swivl is one of the most effective applications for lecturers. Swivl integrates video recording and slideshow presentations in an iPad app making it incredibly easy to share a delivered lecture with the attendants [5]. Another great software extensively used at the Computer Science Department at Brown University is Camtasia for lecture recording. Camtasia is used by several computer science courses and has enabled the students to access the lecture material outside of the classroom [6]. However, for most of the mainstream devices i.e. iPad, iPhone, and Android there are very few applications that focus on the delivery of the presentation and the interactions that the presenter has with his or her audience. This is also influenced by the intrusiveness and the attention requirements of mainstream mobile devices.

The purpose of Google Glass of being unobtrusive and allowing the user to jump in and out of the experience opens the door to a new range of educational applications. In this channel, there have been several efforts to create educational apps for Glass. There have been several educational hackathons, the

most prominent being the Pre-I/O Google Glass Hackathon in 2014. In the hackathon, there were several ideas regarding slide presentations, Lecture Assist focused on manipulating a presentation through Glass [7]. Outside of the hackathon space, the most promising application is called YourShow. It attempted not only to deliver the presenter notes to the device, but also to add control over the presentation through voice recognition. Additional features included broadcasting sound as well as live translation of the slide notes. Despite these promising features the development of the application seems to have ended in mid-2014. Furthermore, there was no support for Macintosh machines or for equations. PowerPoint and Keynote were also not supported. It only worked with Google Slides, requiring lecturers to export their presentations, which added to the learning curve of using a Google Glass device [8].

5 Architecture

To address our requirements and meet our objectives we developed three different versions of TeachWithGlass over the course of the year. The first one, the Heroku version, named after the platform as a service (PaaS) used to host the application, is explained in section 5.4. The purpose of this application was to rapidly create a prototype that met the technical specifications. This version allowed us to validate our assumptions regarding the technology, making sure that our goals were possible. The second version, described in section 5.5, used Google's App Engine instead of Heroku. It built upon what was learned from the Heroku version to create a more robust server infrastructure that would scale appropriately with multiple users. This version emphasized on the usability of the application, focusing on reducing the amount of work performed by the user

to start using the application. The final version, detailed in section 5.6, continued to use Google's infrastructure. The emphasis of this version was on the usability of the user interface. During the time of the development, the application was submitted to Google for review. From this review, there were several suggestions on the user interface design. This final version used these suggestions as building blocks to push the existing design to that of a polished product. This final version made the final step to take TeachWithGlass from a prototype to a polished program that was ready to be evaluated by the academic community at Brown University. What follows is a more detailed discussion of each of the components and the different versions of the application.

The architecture of TeachWithGlass consists of three main applications: the server, the Glassware and the desktop application. The server is responsible for the communication between the Glassware and the desktop application. It is also responsible for maintaining the state of the presentation. The Glassware is responsible for delivering the notes and the user experience in the Glass device. The desktop application, besides establishing a permanent communication with the server, is responsible for manipulating and communicating with the presentation software. All three applications were made with the Java programming language.

5.1 Desktop application

The desktop application is a simple program that maintains the connection with the server while at the same time managing the presentation. The method of connecting with the server — as will be discussed in the sections below — has changed throughout the different versions. The synchronization with the presentation is done through the use of AppleScripts. Changing the

slides, retrieving the presenter notes, and obtaining the current presentation index is all done through the execution of scripts. For example, to retrieve the notes, the Java application creates a text file, executes an AppleScript that retrieves the notes and puts them in the file. The Java app then reads the file and processes the notes. If the notes contain LaTeX equations, then using JLaTeXMath — a third-party library for LaTeX rendering — generates an image for that equation. After generating all the images in the notes it proceeds to upload the images to the server along with the notes. After performing the upload, the app continuously executes AppleScripts to read the state of the presentation and send it the server. In the first iterations of this project, the AppleScripts would also manipulate the presentation.

5.2 Server

The server is responsible for maintaining a session through which the desktop app and the Glassware communicate. Through this session, the notes, the images and the state of the presentation are transferred to the user. There have been two main server architectures: The Heroku and the Google App Engine (GAE) architectures. The Heroku architecture keeps a session by having the user deploy his or her own server. The GAE architecture gets rid of this deployment requirement and is able to host multiple sessions for multiple users using the service simultaneously.

5.3 Glassware

The Glassware is the Google Glass application that runs in the device and delivers the notes and equations to the user. The application has to be constantly

pinging the server to obtain the current state. Needless to say, it also needs to provide a useful user experience.

5.4 Version I: Heroku

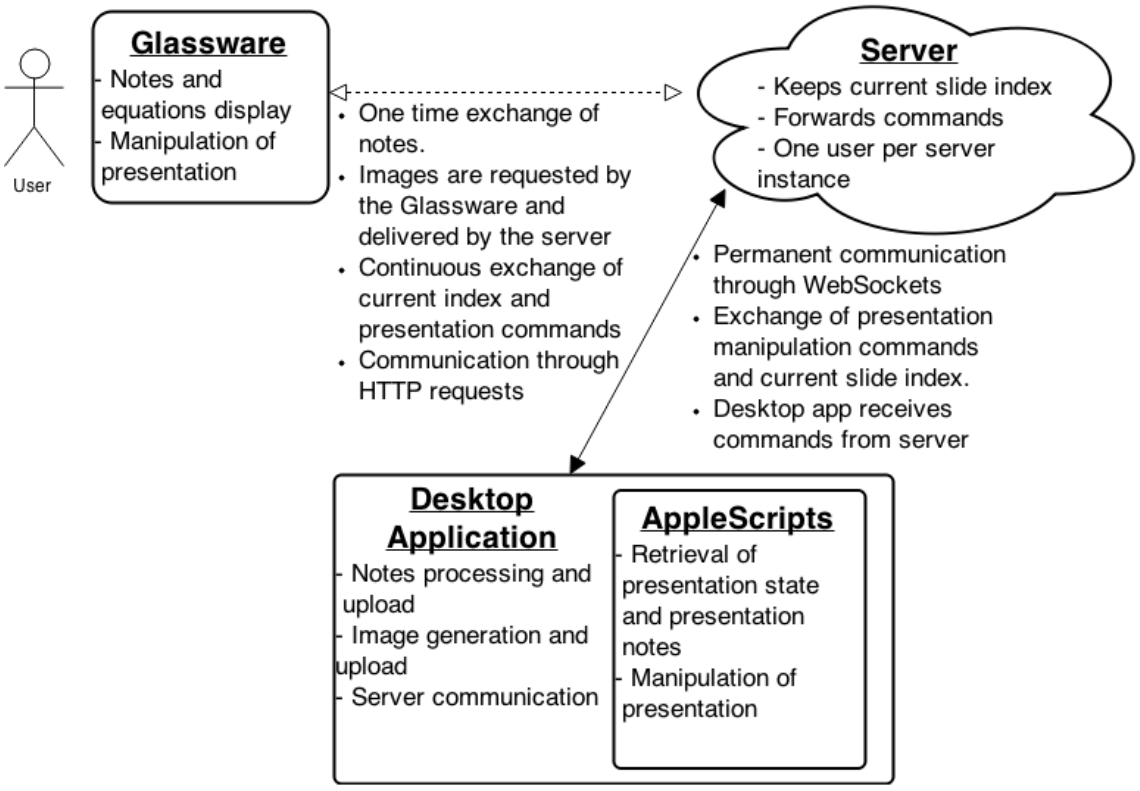

Figure 0: Heroku Server Architecture

This was the very first version of TeachWithGlass and it had all the important features implemented. Figure 0 summarizes the architecture and the role of the main components of the application. Figure 1 and 2 below show the appearance of this first version of the application.

Server: At this stage of the development the server only worked with a single client. If multiple people wanted to use the application they would have had to run their own instance of the server. The communication with the Glassware was maintained through HTTP requests and the communication with the desktop application was through WebSockets. The main advantage of using

WebSockets was the permanent connection that it established between the server and the desktop client. This synchronization was useful to immediately transfer the actions requested by the Glassware and to trigger HTTP requests from the server to the Glassware when the presentation changed. The main disadvantage of this architecture was how cumbersome it was for a user to install the application and start using it. The user would have to clone the repository from Github, create a free account in Heroku and deploy a new application. Regarding the images of equations, these were cached on the server and delivered to the Glassware on request.

Desktop application: As mentioned in the previous section the communication with the server was done through WebSockets. This was particularly useful because it required nothing but opening the connection with the server. Any time that there was an incoming message the Java application would run the appropriate AppleScript script and the presentation would be manipulated. At this stage of the development this component had no graphical user interface. Any modifications had to be done to the code itself and then ran from the command line.

Glassware: The application communicates with the server only through HTTP requests. Selecting "start presentation" from the options menu begins **presentation mode**, when this happens, the notes stored in the server are sent to the Glassware. Additionally, this causes the desktop application to establish its permanent connection with the server.

Once the notes are received, the Glassware application displays the notes for the first slide (slide 0) on the display. Receiving the notes also starts the constant check the Glass device has to do on the server to fetch the most recent index. If a new index is detected, the device displays the notes for that particular slide.

Aside from the constant check it has on the server, the application has the following important features:

- It has a **gesture listeners** that when triggered sends specific actions to the server to manipulate the PowerPoint presentation. **Swiping forward** sends a command to the serve that is forwarded to the desktop application to **go to the next slide**. **Swiping back** does the same thing but **takes the presentation to the previous slide**.
- The Glass application is also responsible for detecting the presence of LaTeX equations and requesting the server for the appropriate image to display.
- When the application is in **presentation mode**, it keeps the screen on so that at all times the user is able to read the notes.
- To terminate the constant check the desktop application is maintaining, the user must select **end presentation** from the options menu.

5.4.1 Notes Format

The notes are fetched from the presenter notes sections located below a slide in editor mode in Microsoft PowerPoint® '11. To optimize the user experience, the user should write the notes for a particular slide separated by a new line. This allows the Glass application to separate the notes by small paragraph and to have better visibility in the device. **For equations**, the user must have the equations separated by empty lines and surrounded with '<<' '>>'. When the Glass application detects this markup, it requests the given equation from the server and displays it in the device. In this version of the application there was no optimization on the presenter notes. If the note was too large it would simply overflow the screen and it would not have been invisible to the

user. Furthermore, the font in the screen would be the same size regardless of the number of words in the note.

The following is an example of the format used for the notes of a particular slide:

The Pythagorean Theorem was well known before Pythagoras, but he is believed to be the first on to prove it.

This theorem is known as a relation in Euclidean geometry among the three sides of a right triangle. Let's consider the equation:

<<a^{2}+b^{2}=c^{2}>>

Ask questions about what other forms can be derived from the original equation

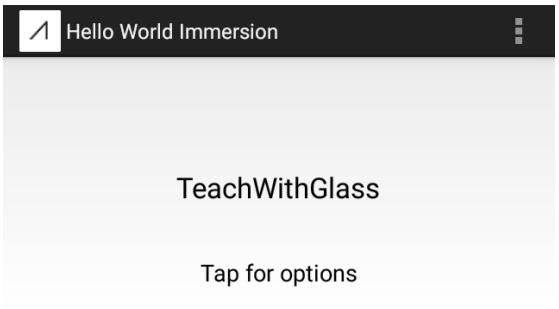

Figure 1: First intro page.

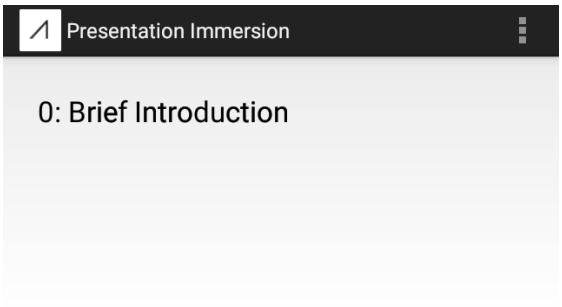

Figure 2: Equal font size

5.5 Version II: Google App Engine

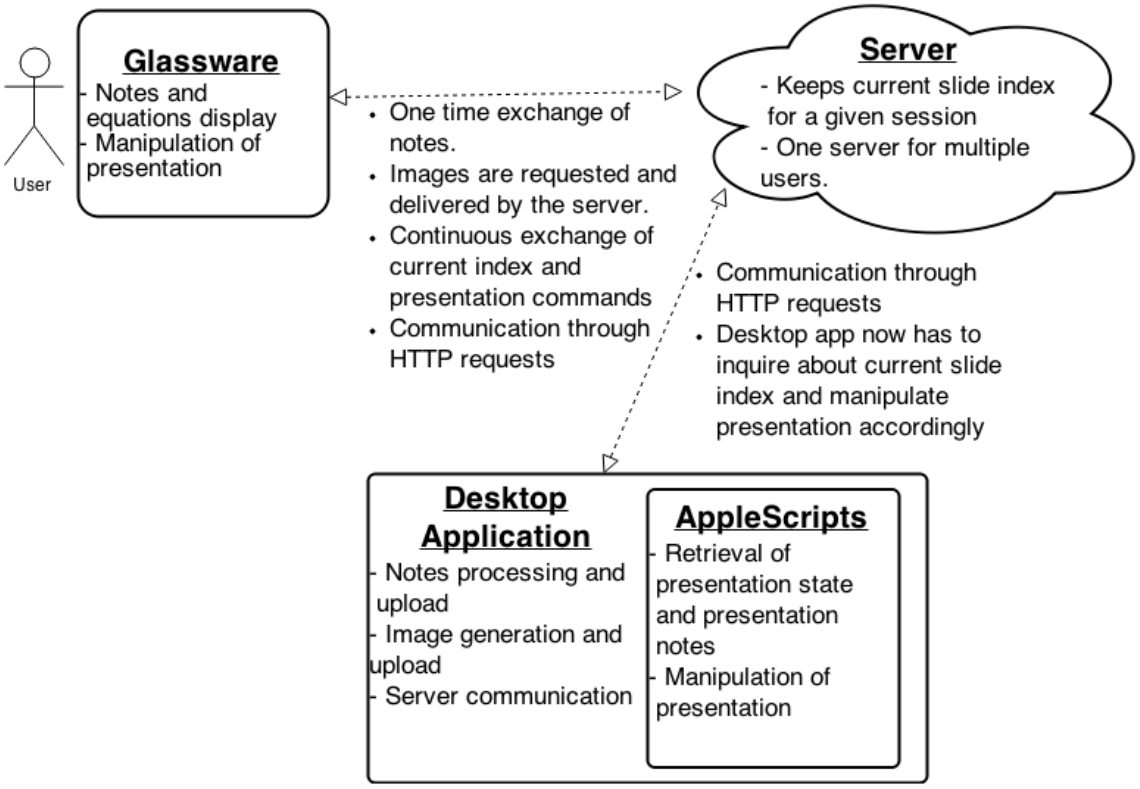

Figure 3: Google App Engine Server Architecture

The main objective of this version was to improve the installation process for the user and to support multiple users with a single server. Figure 3 summarizes the architecture and the role of the main components of the application. Figure 4 is a screenshot of the simple GUI built for the desktop app to interact with the user. Figure 5 displays how the user is prompted to enter the session code provided by the server. The use of sessions was crucial in allowing multiple users on a single server instance.

Server: There was a complete redesign of the server architecture so that it would run in GAE. The challenge of this approach was to find a balance between allowing multiple users simultaneously and going through the process of the authenticating the users each time. At this stage, **sessions** were introduced to the project. Sessions address the issue of having multiple users simultaneously by

separating their content through a session code generated by the server. The images of equations however, were public and saved only once for the sake of space.

This ease of use came at the expense of WebSockets, which are not currently supported by GAE. The regular Java sockets are supported only for paid applications and thus inaccessible for this project [9]. This limitation led to a redesign in the implementation of the permanent communication between the server and the desktop client. This component of the application was implemented through HTTP requests. This change in protocol complicated the double manipulation of the PowerPoint presentation by the Glassware and by the desktop. Without the permanent communication, the desktop application now had to continuously ping the server to request the current state of the presentation.

Desktop application: Communication with the server had to be re-implemented using HTTP requests. The client would send a POST request whenever it detected a change in the presentation but it would continually execute GET requests to read any changes on the presentation made by the Glassware. This led to a few synchronization issues because the server now maintained the state of the presentation, as opposed to immediately reading it from the desktop client when there was a WebSocket connection. Additionally, a graphical user interface GUI was created for the desktop client so that the user could specify the session for the presentation.

Glassware: There were no changes to the Glassware in this version.

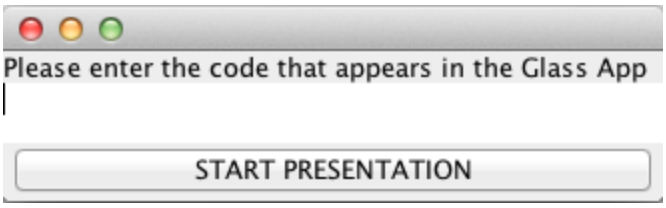

Figure 4: Java GUI

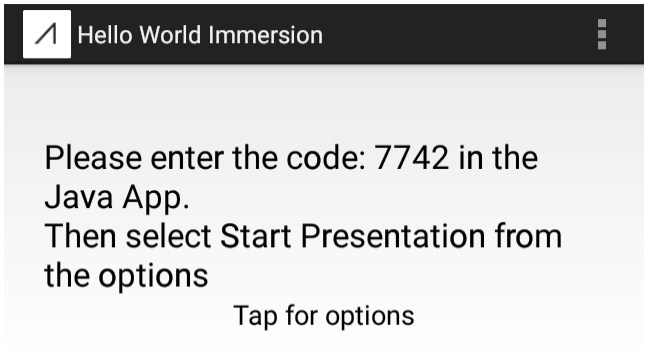

Figure 5: Sessions were introduced

5.6 Version III: Image Caching and UI redesign

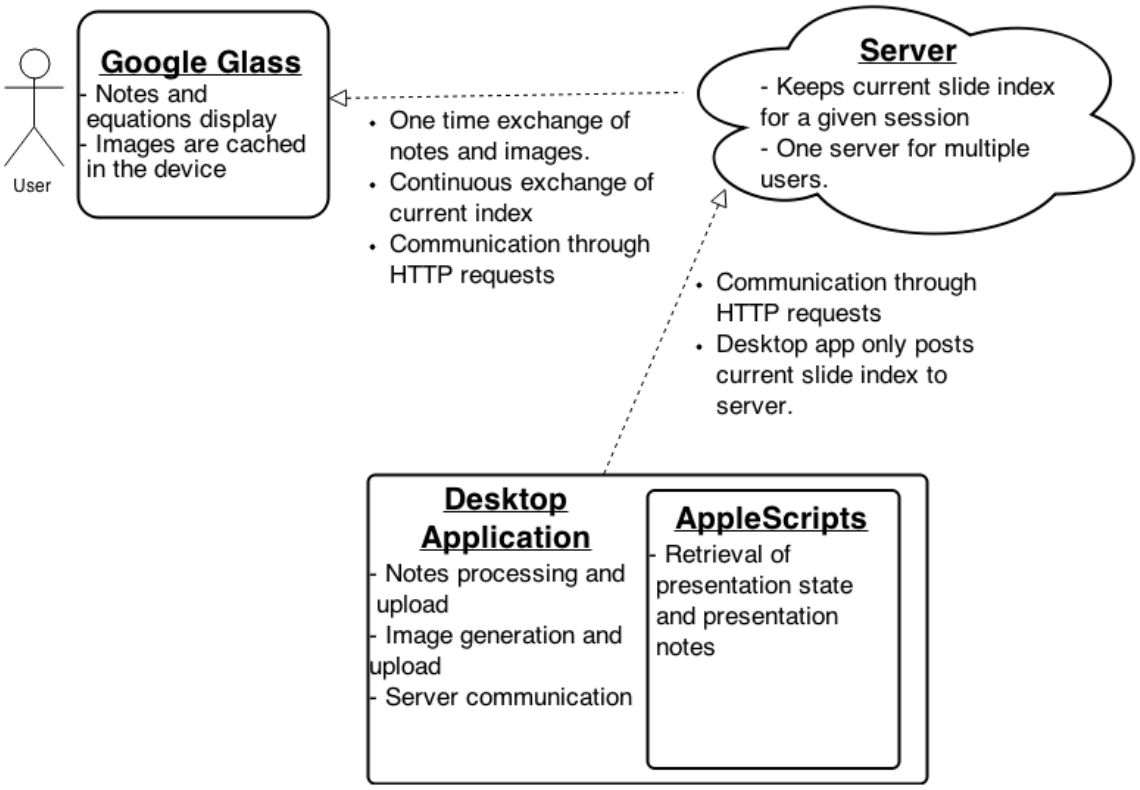

Figure 6: Image Caching Architecture

The main focus of this version was on enhancing the existing user experience. Figure 6 summarizes the architecture and the role of the main components of the application. Figure 7-12 provide a visual proof of the design differences between this architecture and the previous two.

Server: There were no changes to the server in this version.

Desktop application: As mentioned above, the permanent connection through WebSockets is now implemented using HTTP requests. Since AppleScripts have to be executed to retrieve the state of the presentation and to manipulate it, the synchronization was not easy. There were several issues of serialization where the order of the presentation manipulations was not preserved. For this reason, the manipulation capabilities from the Glassware were removed so that the desktop application would no longer have to worry

about any changes made by the Glass device. With the only modifications coming from the desktop client, the Java app could focus exclusively on determining the presentation state and communicating it to the server.

Google Glass Application: As mentioned above, the manipulation capabilities were removed. Furthermore, to speed up the delivery of the notes, the images were cached inside the device as opposed to have them delivered across the network. Additionally, at this stage, the Glassware was submitted to Google for review and from their suggestions, there were several design changes. The design changes included resizing the font according to the number of words in a note and splitting notes in case they would overflow. There were additional minor changes to the process of establishing a session.

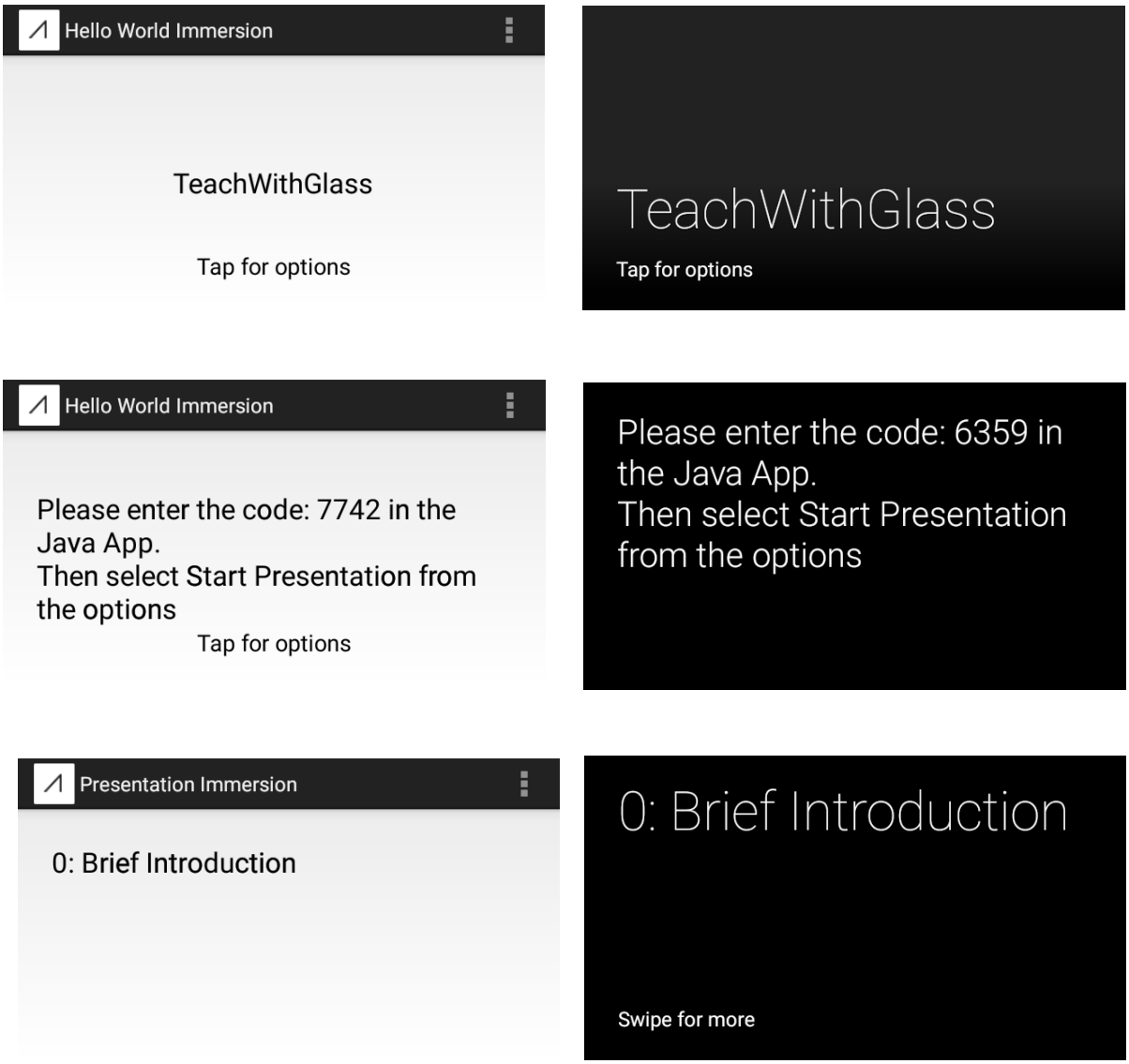

Figure 7-12: Visual comparison between version I & II (left) and version III (right)

6 Publication on Glassware Store

To submit a Glassware project one has to answer a survey with several questions that inquire about the procedure through which the Glassware was developed. It also requires having certain design aspects such as a logo, a promotional image and a website. After one submits the project to Google, the Glassware goes through an evaluation process in which the user experience, the functionality and the design of the application are considered. This is a much more thorough review compared to the regular Android application review which has minimum requirements. The underlying objective is to control quality, design, and consistency across the applications [10].

6.1 Suggestions made by Google & Lessons Learned

There were around ten suggestions made by Google. Some of them were regarding the lack of design assets. Another batch of them had to do with design aspects of the Glassware and the last section of the improvements had to do with the user experience. This set of suggestions was what guided the third version of the architecture. The major changes are the following:

- A white background is very ineffective while presenting notes. The presentation of the notes had to be changes to a black background with white letters instead.
- Using the same font size regardless of the number of words is a waste of space. The font should be bigger with fewer words and readjust accordingly.

- It was imperative that the user experience was as easy and streamlined as possible. There was a small lag when equations had to be requested and displayed to the user. Without any feedback it is very easy for the user to think that the application is broken and stop using the application. This led to caching images to allow for quicker retrieval and delivery.

Going through Google's review process made TeachWithGlass emphasize on the user experience. Reaching this stage of usability was extremely important to continue developing. Without something that people can easily use, the feedback from testing would always be hindered by the usability limitations.

7 Installation Procedure

Glassware: To install TeachWithGlass the user logs into his or her Google account, proceeds to the Glassware store and downloads it to the Glass device linked with that account.

Server: The current version of the server requires no installation. The user simply has to run the other two applications.

Desktop application: The user must have at least Java 7 installed. Once this requirement is fulfilled, the user would have to download a jar file and a folder with scripts. The jar file would run the desktop application.

8 Initial User Studies

8.1 Anecdotal

From personal experience, the application is fast and easy to use. I have used it several times with mock presentations that have multiple equations or

that alternate between equations and written notes. In all of the cases, the use was satisfactory. The desktop application even detects once the presentation is not in presentation mode and resumes the state detection once the presentation is back in full screen mode.

After using it in several occasions, I was able to jump in and out of the experience glancing at the notes or equations when needed.

8.2 Tech Showcase at Brown University

On Friday April 10, 2015 I used TeachWithGlass in a real setting for the first time. I was going to present it at a Tech Demo event where the Education Technology Center of Brown University would present modern educational tools to faculty and staff. The presentation lasted less than twenty minutes and there were no equations involved but I was able to use TeachWithGlass to better engage with my audience by separating myself from the podium. At this showcase I also received very positive feedback and one more volunteer willing to test TeachWithGlass in a real educational setting. Professor Pendse, the Chief Information Officer of Brown University was willing to be a tester for the experience.

8.3 Lessons from CS1951A

On Thursday April 16, 2015 Professor Tim Kraska used TeachWithGlass in one of his lectures for CS1951A: Introduction to Data Science. The lecture lasted approximately 90 minutes; it took place in a medium size lecture hall in front of 80 students. From the feedback obtained after the evaluation there were several positive comments. In particular:

- It is very easy to use and set up.

- It was not disruptive to the flow of the lecture.
- It helped him immensely to remember all the points that he wanted to make for a particular slide.
- The students did not seem to be bothered by its use.
- The equations were particularly useful to guide the students during several iClicker questions.

Additionally, there were a few considerations that need to be taken into account for future use.

- At the beginning it feels very geeky to wear a Google Glass and one has to get used to the feeling of having the small screen above one's face.
- To ease the students into this different experience, Professor Kraska offered the students to use the device after class and several students did.

Even with these considerations, Professor Kraska would definitely use it again. With this positive initial feedback, we feel confident that other professors would be willing to try out the experience. It is our hope that after several more positive evaluations TeachWithGlass will be available to all professors at Brown University and that eventually it will be adopted at other educational institutions.

9 Code

The code for the project is located in several repositories.

- The Heroku Server is located in: https://github.com/crro/GoogleGlassServerHeroku
- The Google App Engine Server is located in: https://github.com/crro/GoogleGlassServerAppEngine

- The Desktop app and the AppleScripts are located in: https://github.com/crro/GoogleGlassDesktopApp
- The Google Glass App is located in: https://github.com/crro/GoogleGlassLecturingApplication

10 Conclusion

Technology is becoming more and more embedded in our everyday lives. While individuals are quickly adapting to the new technologies, other areas such as education, are lagging behind. This lag comes more from the lack of development to adopt existing technologies rather than the development of new ones. TeachWithGlass is a step in the positive direction of developing for adoption.

TeachWithGlass began with the goals of synchronizing a Glassware application with slide presentation software to empower a user giving lecture. It did so by freeing the user from the software's presenter notes and from the handwritten notes for mathematical derivations by rendering images of LaTeX equations into the device. In its early stage, TeachWithGlass was a program with three different components that needed to run at the same time in order for the technical specifications to be met. After two semester of work it is now an application that is easy to install and use. It does no longer require users to install their own server, it is integrated with the Glassware store and it only requires the user to enter a one-time code to set up a session.

The project is still under development and now entering a more exciting phase of exploration. We are currently looking for evaluators to obtain feedback on how to improve it. The initial results have been promising and the

possibilities for future improvement and adding additional features are exciting. It is our hope that TeachWithGlass will continue growing and that it will also inspire other people in the field to further explore the adaptation of existing technologies for educational purposes.

11 Acknowledgements

There are many people to thank for this project. First and foremost I want to thanks my parents David Correa Guerra and Maria de Lourdes Orozco Campos and my amazing sister Daniela Correa Orozco. You have been my motivation, the catalyst of my personal and academic growth and my source of strength. Together we have gone through very rough patches and yet we have made it this far. Thank you to my thesis readers Tim Kraska and Anastasios Matzavinos for sponsoring me and believing in this project. Thank you Andy van Dam (avd) for your support and advice throughout my career at Brown; even though it took me a summer of research and an appendectomy for you to remember my name and its correct pronunciation. Thank you Tom Doeppner (twd) for your academic advice, support and teachings during my darkest times (a.k.a. weenix).

Thank you to my selected family of 144 Lloyd Ave. Javier Flores Kim, Rafael Eduardo Contreras, Tauseef Khan, Tuka Rentsen, and Viktor Gavrielov, your moral support and friendship have made this trip incredible and memorable. Thank you Carlota Pereda Serras for keeping me sane, for always supporting me and giving me the strength and companion needed to keep going. Finally, thank you to the faculty and staff of the Applied Mathematics and Computer Science departments at Brown University, to the entire Brown community and to all my friends and extended family.

12 References

1. Google. (2012, April 4). Project Glass: One Day. Retrieved April 21, 2015, from https://plus.google.com/ GoogleGlass/posts/aKymsANgWBD
2. Google. (2015, January 15). We're graduating from Google[x] labs. Retrieved April 21, 2015, from https://plus.google.com/ GoogleGlass/posts/9uiwXY42tvc
3. Augmedix. (2015, January 12). Augmedix Announces Additional $16 Million in Series A Funding. Retrieved April 21, 2015, from http://www.augmedix.com/press-release-1230349
4. GuidiGo. (n.d.). Discover or create guided tours for Google Glass. Retrieved April 21, 2015, from https://www.guidigo.com/glass
5. Swivl. (n.d.). About Swivl. Retrieved April 21, 2015, from http://www.swivl.com/about/
6. TechSmith. (n.d.). Capture, edit, and share your ideas with TechSmith Camtasia. Retrieved April 21, 2015, from https://www.techsmith.com/camtasia.html
7. Lecture Assist. (n.d.). Retrieved April 21, 2015, from http://challengepost.com/software/lecture-assist
8. YourShow - Empowering Public Speakers with Google Glass. (n.d.). Retrieved April 17, 2015, from https://yourshow.superhumanlabs.com/about
9. Sockets Java API Overview. (n.d.). Retrieved April 17, 2015, from https://cloud.google.com/appengine/docs/java/sockets/
10. Glassware Review Request. (n.d.). Retrieved April 17, 2015, from https://developers.google.com/glass/distribute/form