Computer Science

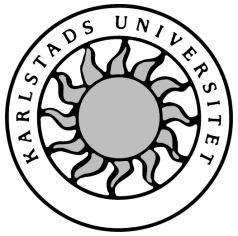
Johan Häger

An Evaluation of Google Glass

Design, Implementation and Evaluation of an Product Assembly Application for Google Glass and Smartphones

Degree Project for Master of Science in Engineering, Computer Engineering

June, 2015



Datavetenskap

Johan Häger

En Utvärdering av Google Glass

Design, Implementation and Utvärdering av en Produktsammansättningsapplikation för Google Glass och Smarttelefoner

---

Examensarbete för Civilingenjörsexamen i Datateknik

Juni, 2015

An Evaluation of Google Glass

Design, Implementation and Evaluation of an Product Assembly Application for Google Glass and Smartphones

Johan Häger

This dissertation is submitted in partial fulfilment of the requirements for the Degree Project for Master of Science in Engineering, Computer Engineering. All material in this dissertation which is not my own work has been identified and no material is included for which a degree has previously been conferred.

Johan Häger

Approved, 2015-06-05

Advisor: Donald F. Ross

Examiner: Thijs. J. Holleboom

Abstract

Assembling components in a production line could potentially be a tedious task, if performed stepwise by the book. However, an employee who is assembling many different products may not know all the steps by heart. As such they will be reliant on an instruction manual. However, an instruction manual must be carried around and, while assembling components, placed in the assembler's line of sight. Instead new technology could make the process more efficient. Google Glass places a display slightly above the user's line of sight and can be controlled via voice commands, and as such solves many of the problems associated with carrying around instruction manuals. This dissertation is an evaluation of Google Glass and describes the design, implementation and evaluation of an product assembly application for both Google Glass and smartphones. The smartphone version was implemented in order to provide a reference point as well as means of comparison with the Google Glass application. The test application used in the study was to read a QR code and download a set of assembly instructions. Testing was carried out on the different steps of the application, from when the QR code has been scanned until the information was displayed to the user. The results show that Google Glass is almost always slower, in all steps, compared to the smartphone equivalents. The conclusion is that Google must upgrade and improve on Google Glass and in particular the hardware. Google Glass overheats easily and the camera is of inferior quality. Google's implementation restrictions also limits what developers might be able to do with the device. However, Google Glass is easy to use and has potential to become a more useful device in the future.

Keywords: Google Glass, Android, Front-end, Performance evaluation

Abstract (In Swedish)

Montering av komponenter i en produktionslinje kan potentiellt vara en tradig och mekanisk uppgift, om det utförs stegvis enligt instruktioner. En anställd som bygger många olika produkter kan dock eventuellt inte samtliga steg utantill, utan blir beroende av en bruksanvisning. En bruksanvisning måste dock bäras runt och, vid montering av komponenter, lämnad i monterarens siktlinje. Ny teknik skulle kunna gőra processen mer effektiv. Google Glass heter den enhet som placerar en display något őver användarens siktlinje och kan styras via rőstkommandon, och således lőser många av de problem som är förknippade med att bära runt en bruksanvisning. Denna uppsats är en utvärdering av Google Glass och beskriver utformning, implementering och utvärdering av en produktsammansttningsapplikation för både Google Glass och smarttelefoner. Smarttelefon-versionen implementerades i syfte att ge en referenspunkt samt medel för jämförelse med Google Glass applikationen. Test-applikationen som används i studien kan skanna en QR-kod och ladda ner en uppsättning monteringsanvisningar. Testning utfördes på de olika stegen i applikationen, från när QR-koden har skannats tills informationen visas för användaren. Resultaten visar att Google Glass nästan alltid är långsammare, i alla steg, jämfört med smarttelefonekvivalenter. Slutsatsen är att Google måste uppgradera och förbättra Google Glass, och särskilt hårdvaran. Google Glass őverhettas lätt och kameran är av sämre kvalitet. Googles implementationsbegränsningar begränsar också vad utvecklarna skulle kunna gőra med enheten. Google Glass är dock lätt att använda och har potential att bli en mer användbar enhet i framtiden.

Nyckelord: Google Glass, Android, Front-end, Utvärdering av prestanda

Acknowledgements

Firstly, I would like to thank Richard Hoorn, who has been my partner in crime on this project. Without your amazing work on the back-end part of the project, my work on the front-end would simply be half the solution to the original problem.

I would also like to send a big thank you to Donald F. Ross at Karlstad University for being my supervisor throught the writing of this dissertation, for always pushing me forward and inspiring me to do even better.

Also at Karlstad University, I would like to thank Daniel Johansson, laboratory engineer at Karlstad University, for allowing me to borrow the equipment used in the experimental setup.

I would like to thank Mats Persson at Sogeti, Karlstad, for being my supervisor during the development process and making sure the project stayed on track, with feedback and check-ups during regular sprint demonstrations.

Thank you also to Kalle Henriksson at Sogeti Karlstad, for taking your time to help review the code and for coming with useful tips and tricks on how the code could be improved.

Thank you to Sogeti Karlstad, overall, for allowing me to take on such an interesting project and for allowing me to work with such exciting, new technology.

Also a big thank you overall to all the people over at Sogeti Karlstad for always making me feel welcome and for all the good chats and talks over breaks and lunches.

Finally, thank you to all those who have wished me good luck, or in any other way supported me throughout working on this project. Your kind words means a lot!

Contents

List of Figures

List of Tables

Listings

1 Introduction

Although assembling components into a fully functional product can be a tedious task, an employee assembling many different products may not be able to learn the process by heart, and will need to rely on instruction manuals.

However, carrying instruction manuals around would mean having to find specific instructions and also, while assembling components, the instructions will have to be placed in the assembler's line of sight. As such, the assembler's focus will have to shift between the components and the instructions, perhaps turn his or her head around, and the assembler may potentially even need to use his or her hands in order to browse the instructions.

One solution to the problem would be to use a TV or computer screen. For example, instructions could be browsed hands-free using a foot pedal. However, such technology requires the use of both hands and one foot. In other words, it would require some coordination. Although mobile, such technology would also had to be carried around, similar to printed instructions.

Using a smartphone removes the requirement for written instructions on paper and a smartphone is also small enough to be put in a pocket. However, a smartphone still requires touch operations and as such will occupy at least one of the assembler's hands at some point. A smartphone, similar to ring binders, will also have to be placed in the assembler's line of sight.

New technology may offer a better solution. Google Glass is a computational device, which looks like, and is worn as, regular glass frames [52]. Google Glass also has a display positioned slightly above the user's line of sight, meaning that the Google Glass display is always in the assembler's line of sight. Google Glass can also be entirely controlled via voice commands, meaning that the user may operate Google Glass completely hands-free.

This dissertation describes the design, implementation and evaluation of an application for Google Glass. In this project, the application will enable users to scan a quick response (QR) code related to a specific product and get information on which components are

needed to assemble the specific product, as well as instructions on how to assemble the components.

The application will also be built for smartphones, giving a reference point for the evaluation as well as a method of comparison with the Google Glass application. The reason for comparing the Google Glass application with a smartphone application is because smartphone are small and mobile, similar to Google Glass. Using the Google Glass application, users will be able to assemble products by following instructions, without having to shift their focus from what they are doing, and neither will the users have to use their hands in order to control the application.

The application, described in Figure 1.1, will scan a QR code, download information from a database based on the information received from the QR code, and then display the downloaded information in the form of a slide view. The application will exist in two versions, one for Google Glass and another for smartphones. However, they will both function in the same way and as such the device in Figure 1.1 could be either Google Glass or a smartphone.

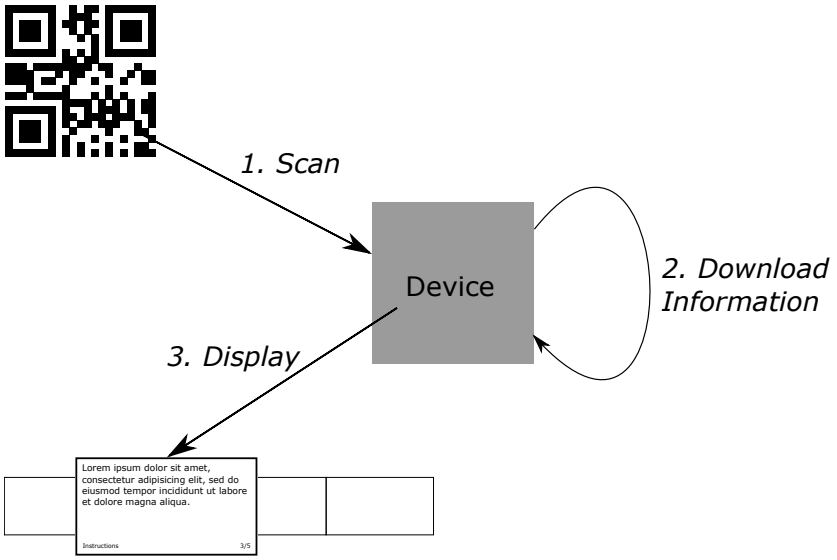


Figure 1.1: Application functionality.

The project was specified by, and implemented at, Sogeti Karlstad. Sogeti is an IT consulting company with focus on competence and modern technology. Since Sogeti works with applications for mobile plattforms, and one of the latest mobile technologies are Google Glass, evaluating the new technology and the potential customer use is of interest to Sogeti. The full project specification, although written in Swedish, can be found in Appendix D.

1.1 Discontinuance of the Explorer Program

It should be noted that, since this dissertation began, Google has chosen to cancel the so called "explorer program" for Google Glass [65]. The explorer program was the "open beta phase" of Google Glass where Google, although Google Glass was sold as a consumer product, was still developing Google Glass at Google's research center, called Google X. Google used users' input and feedback to improve on Google Glass as a product, both in terms of hardware and software.

Now, at the end of the open beta phase, Google has chosen to keep developing Google Glass, but without the help of consumers, meaning that any further updates, both in terms of hardware as well as software, will (as of now) not be publicly available. When discussing the future of Google Glass, Google stated that the company is "thrilled to be moving even more from concept to reality". As of January 20th, 2015, the Google Glass Explorer Edition was no longer available for sale. However, Google ensures that Google Glass will return "when they're ready", but does not go in to further details on when Google Glass will be available for sale again or what the new version will entail.

1.2 Back-End

The original project, of which this dissertation is a part, was divided in to two different parts. This dissertation will cover the front-end part of the project, which includes building the application, both the Google Glass version and the smartphone version. This disser-

tation will also discuss the different ways of presenting information and how information is presented in the application.

However, the second part of the project is the back-end part. The back-end part has been summed up in Figure 1.1 as "Download Information". The back-end part was implemented by Richard Hoorn and consists of a web application programming interface (API), connected to a database in which all product information is stored [92].

1.3 Expected Results

The expected result is that the Google Glass application will prove useful and valuable, more so than the smartphone application. In terms of test results, Google Glass is expected to be on par with the smartphone equivalents. Since the features of Google Glass are their selling points, performing better than, or equal to, smartphones would give a strong argument why Google Glass would be better than smartphones for assembly instructions.

However, should Google Glass perform worse than smartphones in some or all of the tests, Google Glass could still be deemed the better option due to its features. The hands-free aspect of Google Glass is a strong enough argument in itself why Google Glass should be preferred over smartphones.

In terms of the text length Google Glass may fit on screen Google Glass can not be expected to fit the same amount of information on screen as smartphones. The Google Glass screen is smaller than the screen on smartphones available on the market around the time that Google Glas was released (2013). For Google Glass to be considered the better option in terms of the amount of information that may fit on screen, the difference in the amount of information Google Glass is able to display, in comparison to smartphones, should be such that it would still be possible to deliver the necessary information to the user.

1.4 Project Results

The results show that Google Glass is almost always slower, both in terms of decoding the QR code and presenting the information to the user. On average Google Glass is about half a second slower than smartphone equivalents. In terms of the amount of information that may be displayed on the Google Glass display compared to the display on smartphones it was shown that the same amount of text that filled the screen on a smartphone needed three to four separate screens on Google Glass.

1.5 Dissertation Layout

Chapter 2 discusses the relevant background information regarding Google Glass. The chapter will include an introduction to what Google Glass is, how the device came about and what features it has. The background chapter will also discuss similar products to Google Glass, as well as compare Google Glass to smartphones. Finally, chapter 2 will include some discussion on topics relevant to the project, including QR code and ways of presenting information.

Chapter 3 is about the design of the project. The discussion revolves around how the application is intended to work, and what limitations may apply to the implementation of the application, both on Google Glass and smartphone. Chapter 3 also discusses the design of the tests carried out on the application.

Chapter 4 describes the implementation part of the project. The flow of the application is described in detail. Specific aspects of the application are also described in more detail, such as the layout of the slides as well as the voice commands. The experimental setup and how the tests were performed is also described here.

In chapter 5 the results of the tests are presented. The results are presented along with comments regarding their interpretation as well as comments on any potential error factors during testing.

The final chapter, chapter 6, contains conclusions on the project, including conclusions with regard to the test results, as well as conclusions on the project as a whole. Chapter six also includes comments based on personal user experience from using Google Glass for about three months. Finally future work is discussed and the dissertation is concluded with some concluding remarks.

Attached to the dissertation, after the reference list, are appendices. In appendix A abbreviations used throughout the dissertation are listed. Appendix B shows all the individual test results. Appendix C, although namned Code, does not contain the project code, but rather a reference to a separate file containing the code, as there is simply too much code to fit in the dissertation. Appendix D is the last appendix and contains the original project specification, written in Swedish.

2 Background

On April 4th, 2012, Google announced "Project Glass" [7]. Google Glass, as the device is now known, was under development for several years at Google's research and development department, Google X. As part of the announcement Google stated: "We think technology should work for you—to be there when you need it and get out of your way when you don't." [9].

Sergey Brin, one of the founders of Google, gave a Ted Talk in February, 2013, [13] where he talked about why Google decided to develop Google Glass. His argument was that users stayed on their smartphones for too long. Brin also argued that when users were using their smartphones they were looking down at a screen and were not aware of their surroundings. Instead Google wanted to create a device that would give the user notifications that could quickly be dealt and done with. Google also wanted to make the device hands-free and put the display where the user did not have to look down. Brin stated that the development team at Google X added a camera later on in the development process but in fact the camera had been a great addition to Google Glass and enabled the device to capture the user's surroundings, for instance by taking photographs.

Thad Starner, technical lead/manager (responsible for both the technical direction as well as people management [24]) on Google Glass, claimed that Google Glass was intended to be an extension of the self [95]. He compared Google Glass to a watch. Not in terms of where the user keeps his or her focus (with a watch you must look down, similar to a smartphone), but rather in terms of how a watch is easy to access and how the access is instant. Starner said that with Google Glass, Google wanted to minimise the time between intention and action.

2.1 What is Google Glass?

Google Glass, or simply "Glass" as the device is known within Google, is a head-mounted display (HMD) that can be seen as an augmented reality device (see Section 2.1.1 and Section 2.1.4 respectively) designed to bring notifications to the user more easily than a smartphone does. Google Glass is shown in Figure 2.1. According to Google, "Glass is designed to be there when the user needs it and to stay out of the way when the user does not" [54]. Google Glass is meant to give the user relevant information at relevant times.

(a) The user can control Google Glass with the touchpad.

(b) The display sits slightly above the user's line of sight, on the right hand side.

Figure 2.1: Google Glass is equipped with a touchpad and a camera [20].

Google Glass is partially controlled with a touchpad, but can also be controlled through voice commands. The touchpad sits on the right hand side of the user's glass frame and runs from the temple to the ear, as seen in Figure 2.1 (a). When the user touches anywhere on the touchpad Google Glass "wakes up" from stand by and displays the start screen (which consists of a clock). The display is mounted above the user's line of sight, on the right hand side (see Figure 2.1 (b)), and can be slightly adjusted so that the user can see all that is currently being displayed.

The display is a projection that goes through an optic lense in the glass piece, seen in Figure 2.1 (b), which creates a virtual image. A virtual image is an image that, projected through optic lenses, appears to be located at a point where the actual projection is not [90]. In the case of Google Glass the display appears to be located further away from the user

than the display actually is. The Google Glass display is said to be equivalent of a 25 inch high definition screen seen from a distance of approximately 2.5 meters [59].

2.1.1 Head-Mounted Display (HMD)

A head-mounted display (HMD) [81] is a device that is worn on the head and that places a small display in front of one or both of the user's eyes. The device can either be a stand alone device or a part of a helmet. A branch of HMDs are optical head-mounted displays (OHMDs) [85]. An OHMD is an HMD with a see-through display, for instance Google Glass.

2.1.2 Heads-Up Display (HUD)

A heads-up display (HUD) [82] is defined as any transparent display that, when presenting information, does not require users to look away from their usual viewpoints. In other words, an HUD may be an HMD and an HMD may be an HUD. While an HMD is always worn on the head, an HUD can be a stand-alone display. In contrast, an HUD must be a transparent display, which is a requirement an HMD does not have. An OHMD, however, is always an HUD since an OHMD has a transparent display.

2.1.3 Virtual Reality

Virtual reality [77] is defined as a computer generated simulation that enables users to interact with a three-dimensional (3D) environment. Virtual realities are common in interactive mediums such as video games. Virtual realities can also be combined with an HMD in order to completely enclose the user in the virtual reality. One such example is the Oculus Rift, seen in Figure 2.2, that completely covers the user's eyes, allowing the user to experience the virtual reality.

Google Glass is able to display a virtual reality but does not work as a virtual reality device. Google Glass only covers a small part of the user's field of vision and as such does

Figure 2.2: The HMD "Oculus Rift" is a virtual reality device [70].

not have the capability of simulating a 3D, interactive, environment in contrast to the Oculus Rift. Oculus Rift, unlike Google Glass, is able to replace the user's reality with a completely virtual reality since Oculus Rift completely covers the user's eyes.

2.1.4 Augmented Reality

Augmented reality [36] is defined as the combination of reality (or what is within current context being perceived as reality [1]) with useful, computer generated, data. Augmented reality, unlike virtual reality, is not meant to replace reality, but rather to enhance interaction with the current reality.

An HUD may create an augmented reality. The reason an HUD does not always create an augmented reality is due to the fact that the information being presented might not be useful within the current context. An augmented reality is, as stated above, meant to

[1] Augmented reality is for instance common in video games to give the player environmental and health information.

enhance reality, while an HUD does not have that requirement.

Google Glass is an HUD that has the potential (and intent) to create an augmented reality. Google Glass is intended to present useful information to users while not distracting them from reality. One example of useful information that could enhance users interaction with reality would be a shopping list while users are out shopping, as seen in Figure 2.3.

Figure 2.3: A shopping list while the user is out shopping is useful information [54].

2.2 User Interface

The Google Glass graphical user interface (GUI) is called a timeline [20] (see Figure 2.4). The timeline consists of a row of cards. Cards are basic applications such as a clock (see Figure 2.5 (a)) or information about the weather. Cards can also represent more in-depth applications, on Google Glass called “Immersions” (see Figure 2.5 (b) and (c)). Immersions handle activities such as receiving directions to a specific location or playing a game.

The first screen the user sees when starting up Google Glass is the home screen. The home screen displays a clock and also shows the text ”ok glass”, as seen in Figure 2.5 (a). The home screen is a part of the timeline and acts as the center point. Cards to the left of the home screen are upcoming activities such as an event in the user’s calendar or an upcoming flight. Cards to the right of the home screen are from the past. Cards from the

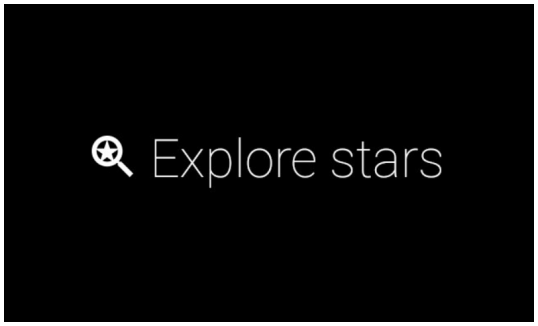
Figure 2.4: A visualisation of the timeline as the timeline is perceived by the user [20].

past will for instance show text messages or photos.

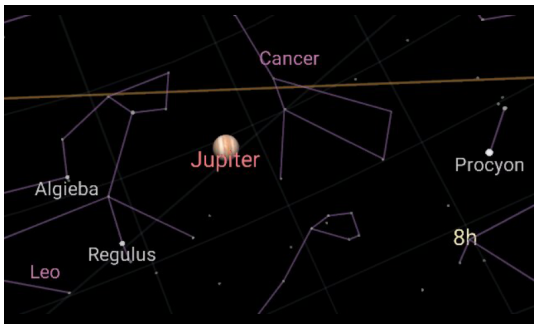
In order to move left on the timeline (forward in time) the user must swipe a finger backwards on the touchpad. In order to move right on the timeline (backward in time) the user must swipe a finger forward on the touchpad. The fact that the user must swipe backwards when stepping forward in time might not seem especially intuitive. In western culture a timeline is normally represented as going from left to right. One example is books, where the reader not only reads each line from left to right, but also turn pages from the right (the future) to the left (the past). However, on Google Glass, the swiping action could be thought of as swiping cards behind the back. Swiping forward when stepping backwards in time would then in turn mean bringing cards placed behind the back into focus. Cards in the past are behind the user while cards in the future are in front of the user.

When the user wants to turn off Google Glass the user swipes down on the touchpad. Swiping down on the touchpad will put Google Glass in stand-by mode. If the user wants to turn off Google Glass entirely, in other words power down the device, there is a power button on the opposite side of the touchpad. Holding down the power button for a few seconds will turn off Google Glass. See Figure 2.4 for a better visual understanding of how Google Glass works, as well as the video referenced in the caption.

(a) The Google Glass home screen is a card that displays a clock.



(b) The card "Explore stars" represents an immersion.



(c) The immersion "Explore stars" allows the user to look around at stars using the built-in head motion tracker.

Figure 2.5: Cards can either display basic applications or represent immersions.

Google Glass uses a bone conduction transducer (BCT) to transfer sound to the user [59]. The BCT transfers sound to the inner ear by conducting sound through the bones of the skull [78]. The advantage of this technique is that the sound maintains clarity, even in noisy environments. Also, since the user does not plug any earphone into their ears, external sound is not blocked out.

Google Glass also features a 5 megapixels camera [59]. The camera sits between the touchpad and the display, as seen in Figure 2.1 (b), and is capable of capturing video at a 720p resolution. The camera can be used for video conferencing, as Google showed in 2012 [6], but the camera can for instance also be used when the user wants to scan a QR Code.

The user can also interact with Google Glass using voice commands. As seen in Figure 2.5, the home screen consists not only of a clock but also of the words "ok glass", in quotes. "ok glass" indicates to the user that voice commands are available. The voice command menu is accessed as soon as the user says the words "ok glass". Doing so brings up a list of voice commands available, as seen in Figure 2.6.

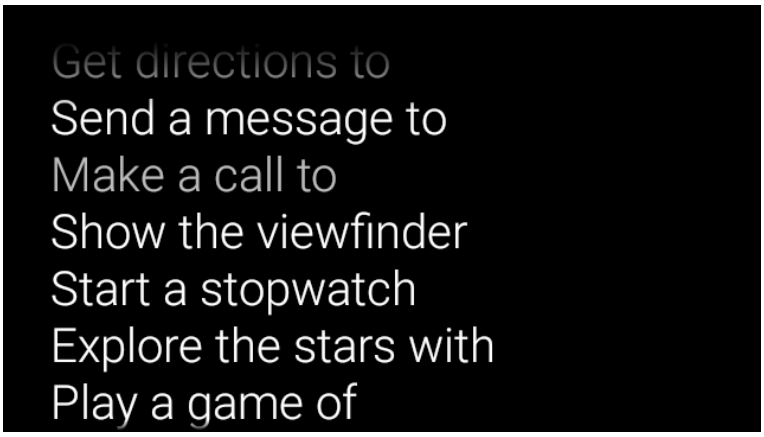


Figure 2.6: Saying "ok glass" will bring up the voice command menu [62].

In order to progress further the user must say one of the options displayed out loud. Doing so will either make Google Glass perform the task spoken or give the user the option to add an input option to the task chosen, depending on whether the chosen task may take input or not. For instance, if the user where to say "ok glass, Start a stopwatch", Google Glass would start a stopwatch. If the user where to say "ok glass, Send a message to", the user would also have to add to who the message should be sent.

Google Glass also supports head motions as a form of input from the user. Head motions are not enabled in the timeline as a way of input but tilting the head may wake up Google Glass from stand by mode, if the user has enabled the head wake up feature [53]. The head motion interface may also be used in certain immersions, such as "Explore stars" (seen in Figure 2.5 (c)) where the user is able to look around the star map by moving his or her head.

2.3 Similar Products

Today there are several products, either already on the market or under development, that are more or less similar to Google Glass. The following is a short list (a more extensive list of devices can be found on Wikipedia [85]) describing some of the competition Google Glass faces, with each product shown in Figure 2.7.

(a) Microsoft Hololens [68]

(b) Recon Jet [75]

(c) GlassUp [41]

(d) C Wear Interactive Glasses [74]

Figure 2.7: There are many OHMD devices similar to Google Glass [85].

2.3.1 Microsoft Hololens

Microsoft's offer in the augmented reality device space is an HUD that displays information in front of both of the user's eyes. Microsoft's device is called Microsoft Hololens and can be seen in Figure 2.7 (a) [68]. While Google Glass is meant to be worn at all times, Microsoft Hololens is rather a device users only wear when they intend to use Microsoft Hololens. Google Glass is, as Thad Starner stated [95], meant to be an extension of the self and is meant to be worn at all times, even though the user might not be actively using

Google Glass at the time, in order to bring helpful notifications and information to the user. Microsoft Hololens is rather a tool to be used actively for a certain purpose, such as modelling [67], and then put away. Google Glass may be used the same way if the user wants to, but that is not the intent.

The mot striking difference between Microsoft Hololens and Google Glass lies in the interaction with the real world. Google Glass is a two-dimensional (2D) display that sits slightly above the users line of sight. Microsoft Hololens, on the other hand, is meant to interact with the world even further.

Microsoft intends to give the user tools to work in a 3D space. Microsoft's concept video [69] of Microsoft Hololens shows examples of 3D modelling with the use of kinetic hand-movement detection. Microsoft Hololens will enable users to see what they are working on from different angles simply by walking around the object, just as if the object in question were real and had a physical mass.

2.3.2 Recon Jet

Recon Jet, seen in Figure 2.7 (b), is an HMD developed by Recon Instruments [75]. Recon Jet is suited for athletes. Because of the target audience, Recon Jet has been fitted with a display that has high contrast in order to give good readability in high ambient lighting. The display's virtual image appears as a 30 inch wide screen at approximately 2 meters distance [76], to be compared with Google Glass' virtual image which appears as a 25 inch high definition screen seen from a distance of 2.5 meters [59].

Unlike Google Glass, Recon Jet's display is located below the user's line of sight, as seen in Figure 2.7. Recon Jet's target audience, athletes, are used to having their information below line of sight. For instance a bike may have dashboard mounted to the handlebar, or an athlete might be using a watch to check the time. Google Glass is meant to be worn at all times while the location and the brightness of the Recon Jet display indicates that Recon Jet is meant to only be used while the athlete is working out and not more regularly.

2.3.3 GlassUp

GlassUp is an Italian company that received most of its founding for the HMD device, GlassUp [39] (seen in Figure 2.7 (c)), through the crowd-funding site Indiegogo [40]. GlassUp has been accused of being too similar to Google Glass, partially because of the name of the device [19]. GlassUp does however make distinctions between the two products. On GlassUp's Indiegogo page the company made the comparison that looking at Google Glass' display was similar to looking in the back view mirror while GlassUp was similar to looking out the windscreen. The comparison referenced the fact that Google Glass' display is located above the user's line of sight, similar to a rear view mirror.

GlassUp instead displays information close to the center of the user's line of sight. GlassUp claimed, on the company's Indiegogo page, that the display was placed closer to the center of the users line of sight so that there would be less strain on the user's eyes. However, the biggest difference from Google Glass is that GlassUp is meant only to act as a second screen. GlassUp is a "receive only" device which displays information from the device currently connected through bluetooth, for instance a smartphone. GlassUp does not do any calculations on its own and must stay connected to a bluetooth device in order to display information [40].

2.3.4 C Wear Interactive Glasses

C Wear Interactive Glasses [73], seen in Figure 2.7 (d), is an industry focused device developed by Penny in Västerås, Sweden [10]. C Wear Interactive Glasses projects an image onto the actual glass in front of the user's right eye and as such covers a larger area than similar devices such as Google Glass, Recon Jet and GlassUp [72]. The display is said to be perceived as a 75 inch display at a distance of 2.1 meters [74]. The projection is transparent which enables users to still see what is happening in front of them.

Being industry focused, C Wear Interactive Glasses is also equipped with a hands-free user interface that does not require voice command. C Wear Interactive Glasses uses a

jaw sensor which lies against the user's jawbone muscle. The sensor detects tension in the muscle, which registers as a click, to be compared with a touch on the Google Glass touchpad [74].

C Wear Interactive Glasses, similar to GlassUp, is designed to be connected to an external device [74]. However, where GlassUp is connected through bluetooth C Wear Interactive Glasses is connected to the external device through an adapter which can send data and visual information via USB and HDMI. The external device can be a smartphone, a tablet, a PC or even a TV.

2.4 A Comparison with Smartphones

Compared to smartphones, one of the biggest advantages of Google Glass is the fact that Google Glass is an HMD. With a smartphone the user needs to either hold the smartphone in either one or both hands, or alternatively put the smartphone on a table. In other words can Google Glass offer a hands-free experience that smartphones cannot.

Another advantage of Google Glass, compared to smartphones, also comes from the fact that Google Glass is an HMD. The user does not need to look away in order to see what is currently being displayed. Google Glass does not distract from what the user is currently doing as much as a smartphone where the user needs to either look away or hold up the smartphone in front of his or her eyes.

However, smartphones give the user a bit more control. The control comes from the fact that smartphones supports multi-touch, which Google Glass does not. On a smartphone users may also touch directly on the screen, in contrast to Google Glass where the touchpad sits on the right hand side of the user. Smartphones also have a larger touch area than Google Glass.

The smartphone screen size has been increasing ever since the iPhone first launched in 2007 [84], as seen in Figure 2.8. Looking at currently available smartphones, in Figure 2.9, the increase in screen size is set to continue as the average screen size is approaching five

inches. In terms of comparison with Google Glass the increase in screen size means that more information could be displayed on a smartphone than on Google Glass.

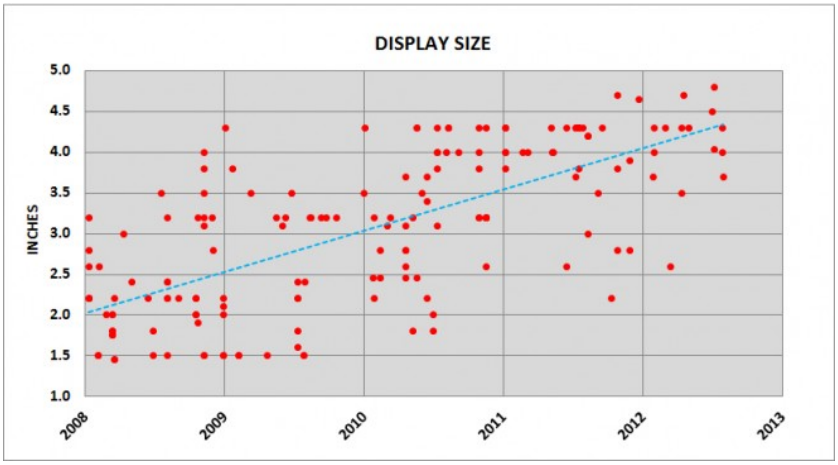


Figure 2.8: Smartphone screens have been increasing in size for several years [12].

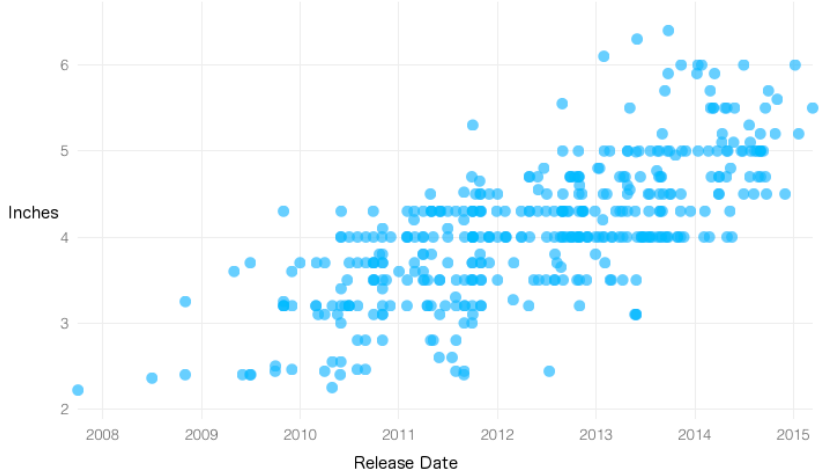


Figure 2.9: Screen sizes of the most popular, currently available smartphones [34].

However, one of the biggest differences between smartphones and Google Glass is the plural, smartphones. There are several smartphone brands competing on the market, each offering several models. Google Glass is simply Google Glass, one product. As seen in Section 2.3, Google Glass does face competitors that have approached HMDs differently,

and as HMDs increase in popularity there is potential for an even wider offering of models and screen sizes.

2.5 Limitations of Google Glass

In terms of limitations of Google Glass, one early concern with Google Glass came from people who wore regular glasses every day, as Google Glass seemed to require separate frames. Isabelle Olsson at Google responded on the issue on April 12th, 2012, with the following: "We ideally want Project Glass to work for everyone, and we're experimenting with designs that are meant to be extendable to different types of frames." [8].

Today a number of eyecare providers have been trained for Google Glass and Glass frames. These trained eyecare providers are, however, mostly located in the United States [28], but Google points out that many eyecare providers should be able to help place lenses on Google Glass' frames [29].

As described in an article posted in Forbes in 2013 [11], one more alarming concern has been the health of the user's eyes. Concerns were raised regarding eye strain and misalignment of the user's eyes, as Google Glass placed a screen above one eye and not both. Google also saw these potential issues and approached Eli Peli, professor of ophthalmology who had been studying HMDs for two decades, as the development of Google Glass started.

In the Forbes article, Peli claimed that Google Glass has been designed with more safety and comfort in mind than previous, similar products. Peli pointed out that Google Glass is see-through and only covers a small part of the user's field of vision. As such, Google Glass does not require a potentially poorly adjusted camera to capture the environment and display the environment to the user, which could cause eye strain.

Peli also pointed out that Google Glass, although meant to be worn at all times, is meant only to be used for short periods of time. Google Glass is meant to give the user notifications that can be quickly dealt with. The user should not be looking at the display for long periods of time, which would have the potential to lead to eye strain. While Peli

stated that the risks are not zero, he still claimed that the likelihood of Google Glass causing any damage is minimal.

Even though, according to Google's expert, there might not be any health risks involved, there is still a question of how much help Google Glass may be to users. A study performed in 2002 [96], regarding the effects of OHMDs, showed that OHMDs may only be of help to users under controlled forms. Whenever the surrounding evnironment becomes too distracting, for instance within a moving crowd, performance decreases. The study however noted that pilots had been able to successfully turn HMD into a tool they could use to their advantage. Since the study was not carried out over a long period of time the participants were potentially not given enough time to get used to wearing and using their HMDs, explaining the poor performance when using a distracting background.

2.6 QR Code

As mentioned in Section 2.2, Google Glass is equipped with a camera that could be used to take photos from the user's perspective. One potential use of the camera would be to scan QR codes. The QR Code was announced in 1994. Having been under development for several years at Denso Wave [16], the goal was to create a new form of barcode that could carry more information than a linear barcode and that could also be easily read.

A conventional barcode is capable of storing approximately 20 digits, while a QR code can store several thousand digits [17]. In a QR code, information is encoded using standardised encoding modes and displayed as a 2D barcode. A QR code has several standardised fields, as seen in Figure 2.10. Using position fields, a QR code can be read from any direction, compared to a conventional barcode which can only be read horizontally [18].

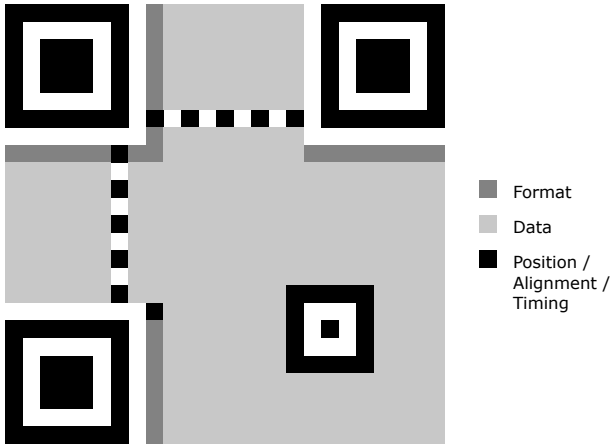


Figure 2.10: The standardised fields of a QR code [86].

A QR code can be used to encode information, originally written with alphabetic characters, Japanese symbols (Kanji, Katakana and Hiragana) or numeric characters [15]. With the help of a QR code, information which would otherwise have taken up a large space can now be easily fitted in smaller areas.

2.6.1 Decoding

Decoding a QR code is a fairly straight forward process which, although time consuming, can be done by hand, as described in several guides on the Internet [14, 27, 93]. A QR code may be divided in to three standardised fields, which help QR code scanners to identify and decode QR codes. The three fields can be seen in Figue 2.10. The position, alignment and timing fields are used to identify and position the QR code correctly as a QR code may be scanned from any direction. In order to decode a QR code the three main position fields residing in three of the QR code's corners must be positioned as seen in Figure 2.11.

The next step is to identify the mask used on the data field in the QR code. The mask is used to remove any large, empty or filled, areas within the data field which otherwise could make the decoding process more difficult for QR code scanners. The mask field is a part of the format field, seen in Figure 2.10. The format field holds information about

error correction, as well as which mask has been used on the data field. The mask is found among the first five bits in the format field. The first two bits of the first five bits of the format field contain information regarding the amount of error correction data the QR code contains, and the other three contain information on which mask has been used.

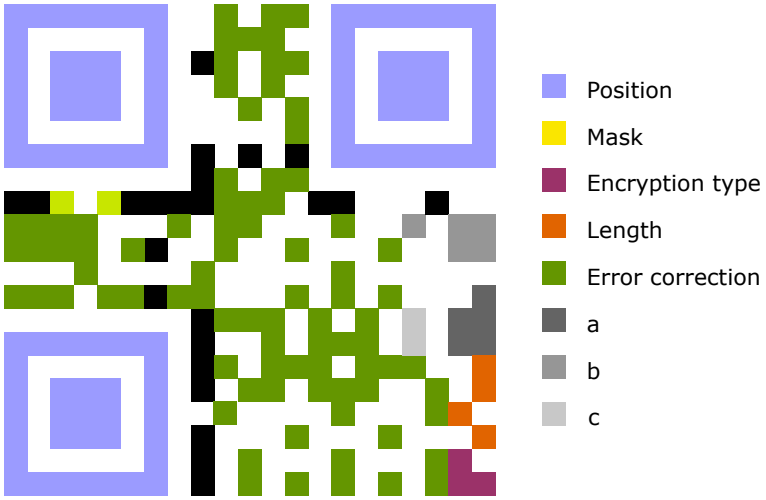


Figure 2.11: A QR code example, encoded with the string ”abc”.

In Figure 2.11 the mask field has the value of 101, seen more clearly in Figure 2.12. Filled blocks should be interpreted as a 1 and empty blocks should be interpreted as a 0. However, the mask field is always XOR:d with 101 prior to being printed in the QR code and must as such be XOR:d back to the original mask value. In the case of Figure 2.11 the original mask value is calculated as follows:

$$101 \text{ XOR } 101 = 000$$

Figure 2.12: Mask pattern encoded as 101.

There are eight different mask patterns in total, each represented by a unique bit string. The mask pattern represented by 000 can be seen in Figure 2.13, while the rest of the mask

patterns may be found here [5]. The mask used in Figure 2.11 means that a bit should be flipped when the following formula is true:

$$(i + j)\ mod\ 2 = 0,$$

where i and j represent the indexes of the specific block, horizontally and vertically respectively [5].

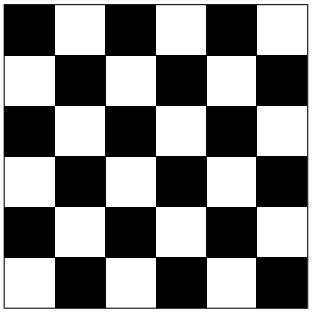


Figure 2.13: Mask pattern represented by the bit string 000 [5].

Having decoded the mask pattern the next step is to decode the data area. The data area always contains a header. The header contains information on the encryption type as well as data length. The blocks containing the encryption type are always the size of four blocks, and always located in the lower right corner, as seen in Figure 2.11. The number of blocks used for the data length may vary between eight and ten blocks depending on the encryption type.

As such the first step in decoding the data area is to decode the encryption type. The information in the data area is to be read from the lower right and upwards, in a zig-zag pattern, with a width of two blocks. When reaching the top the zig-zag pattern continues, although downwards. See Figure 2.14 for a better understanding of the zig-zag pattern. The start point is always in the lower right corner since the encryption type must be decoded first.

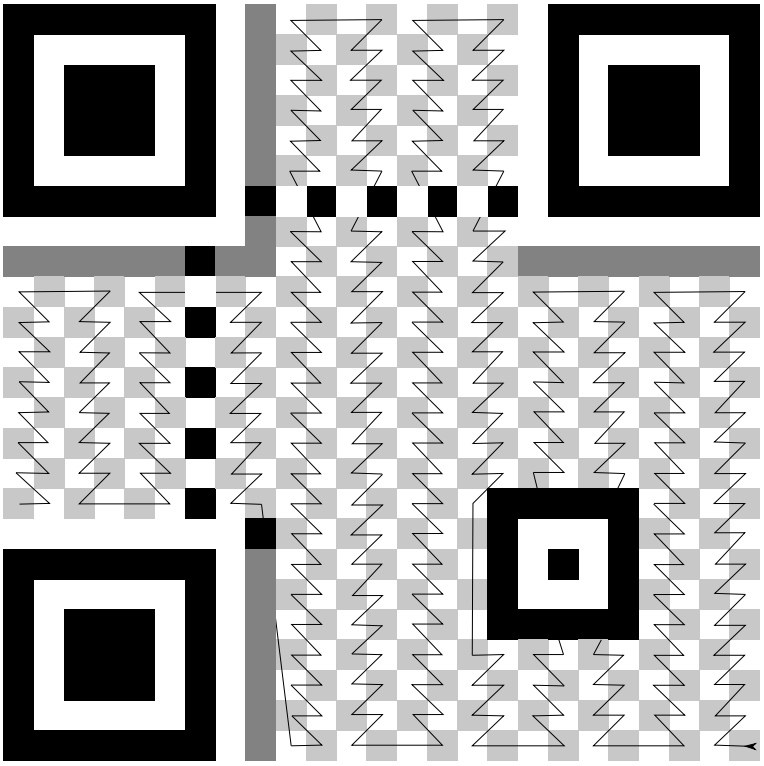


Figure 2.14: The zig-zag pattern used when decoding a QR code.

Using the zig-zag pattern, the encryption type bit string in Figure 2.11 can be found to be 1101, seen more clearly in Figure 2.15. However, the encryption type is a part of the data section and as such must be unmasked. Using the mask formula gives the following results:

$$(20 + 20)\ mod\ 2 = 40\ mod\ 2 = 0$$
$$(20 + 19)\ mod\ 2 = 39\ mod\ 2 = 1$$
$$(19 + 20)\ mod\ 2 = 39\ mod\ 2 = 1$$
$$(19 + 19)\ mod\ 2 = 38\ mod\ 2 = 0$$

Figure 2.15: Encryption type encoded as 1101.

As such the blocks at position (i, j) = (19, 19), and (i, j) = (20, 20) must be flipped. The "flipping" process may be done by XOR:ing the massked encryption type bit string with a bit string representing the bits that must be "flipped", putting ones at the positions where the corresponding bit in the masked encryption type bit string must be flipped, and zeroes at the position where the corresponding bit does not need to be flipped. The masked encryption type bit string, 1101, must as such be XOR:d with 1001, as follows:

$$1101\ XOR\ 1001 = 0100$$

There are several encryption types used in QR codes, however the most common ones are the following (represented by the following bit strings):

- Numeric 0001
- Alphanumeric 0010
- 8-Bit Byte 0100

The encryption type used in Figure 2.11 is as such of encryption type 8-bit Byte.

After having decoded the encryption type the next step is to decode the length. The encoded length is the length of the message encoded in the QR code. Since the message encoded in the QR code may not cover the entire data section of the QR code (the rest is made up of error correction information) it is necessary to know the length of the message in order to tell when the message ends. Since the encryption type used in Figure 2.11 is an 8-bit Byte the size of each field that follows is eight bits in size.

The length field in Figure 2.11, seen also in Figure 2.16, is the following bit string: 10011010. However, the length field is a part of the data section and must as such be unmasked in order for the original length value to be obtained.

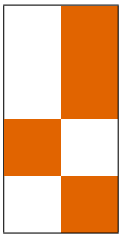


Figure 2.16: The message length encoded as 10011010.

$$(18 + 20)\ mod\ 2 = 38\ mod\ 2 = 0$$
$$(18 + 19)\ mod\ 2 = 37\ mod\ 2 = 1$$
$$(17 + 20)\ mod\ 2 = 37\ mod\ 2 = 1$$
$$(17 + 19)\ mod\ 2 = 36\ mod\ 2 = 0$$
$$(16 + 20)\ mod\ 2 = 36\ mod\ 2 = 0$$
$$(16 + 19)\ mod\ 2 = 35\ mod\ 2 = 1$$
$$(15 + 20)\ mod\ 2 = 35\ mod\ 2 = 1$$
$$(15 + 19)\ mod\ 2 = 34\ mod\ 2 = 0$$

The encryption type bit string must as such be XOR:d with 10011001, as follows:

$$10011010\ XOR\ 10011001 = 00000011 = 3$$

The message in Figure 2.11 is as such of length 3.

Finally, the data is decoded. The first 8-bit Byte, seen in Figure 2.17, gives the following bit string: 11111000. However, being a part of the data section, the bit string must be unmasked, as follows:

$$(14 + 20)\ mod\ 2 = 34\ mod\ 2 = 0$$
$$(14 + 19)\ mod\ 2 = 33\ mod\ 2 = 1$$
$$(13 + 20)\ mod\ 2 = 33\ mod\ 2 = 1$$
$$(13 + 19)\ mod\ 2 = 32\ mod\ 2 = 0$$
$$(12 + 20)\ mod\ 2 = 32\ mod\ 2 = 0$$

$$(12 + 19)\ mod\ 2 = 31\ mod\ 2 = 1$$
$$(11 + 20)\ mod\ 2 = 31\ mod\ 2 = 1$$
$$(11 + 19)\ mod\ 2 = 30\ mod\ 2 = 0$$

The first 8-bit Byte bit string must as such be XOR:d with 10011001.

$$11111000\ XOR\ 10011001 = 01100001 = 64 + 32 + 1 = 97$$

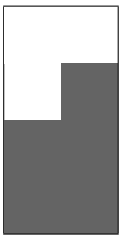


Figure 2.17: The first 8-bit Byte encoded as 11111000.

The second 8-bit Byte, seen in Figure 2.18, reaches the top of the data section and as such the last four bits must be read horizontally to the left, as described in Figure 2.14. Doing so gives the following bit string: 11110100. Again, the bit string must be unmasked.

$$(10 + 20)\ mod\ 2 = 30\ mod\ 2 = 0$$
$$(10 + 19)\ mod\ 2 = 29\ mod\ 2 = 1$$
$$(9 + 20)\ mod\ 2 = 29\ mod\ 2 = 1$$
$$(9 + 19)\ mod\ 2 = 28\ mod\ 2 = 0$$
$$(9 + 18)\ mod\ 2 = 27\ mod\ 2 = 1$$
$$(9 + 17)\ mod\ 2 = 26\ mod\ 2 = 0$$
$$(10 + 18)\ mod\ 2 = 28\ mod\ 2 = 0$$
$$(10 + 17)\ mod\ 2 = 27\ mod\ 2 = 1$$

The second 8-bit Byte bit string must as such be XOR:d with 10010110.

$$11110100\ XOR\ 10010110 = 01100010 = 64 + 32 + 2 = 98$$

Figure 2.18: The second 8-bit Byte encoded as 11110100.

The third, and final 8-bit Byte (seen in Figure 2.19), to be decoded is read downwards, as described in Figure 2.14, giving the following bit string: 00000101. The bit string must be unmasked:

$$(11 + 18)\ mod\ 2 = 29\ mod\ 2 = 1$$
$$(11 + 17)\ mod\ 2 = 28\ mod\ 2 = 0$$
$$(12 + 18)\ mod\ 2 = 30\ mod\ 2 = 0$$
$$(12 + 17)\ mod\ 2 = 29\ mod\ 2 = 1$$
$$(13 + 18)\ mod\ 2 = 31\ mod\ 2 = 1$$
$$(13 + 17)\ mod\ 2 = 30\ mod\ 2 = 0$$
$$(14 + 18)\ mod\ 2 = 32\ mod\ 2 = 0$$
$$(14 + 17)\ mod\ 2 = 31\ mod\ 2 = 1$$

The third 8-bit Byte bit string must as such be XOR:d with 01100110.

$$00000101\ XOR\ 01100110 = 01100011 = 64 + 32 + 2 + 1 = 99$$

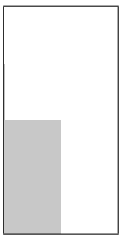


Figure 2.19: The third 8-bit Byte encoded as 00000101.

Since the message length was found to be of size 3, all parts of the message in Figure 2.11 have been found. The rest of the data section contains information regarding error correction, used in case the QR code was somehow damaged.

The message in Figure 2.11 has been decoded as 97, 98 and 99. Converting there numbers using an ASCII table gives the following result: a, b and c [2]. The encoded message in Figure 2.11 has as such been decoded to "abc", which is correct and may be checked by scanning Figure 2.11 using a QR code scanner.

Although the decoding of a QR code may seem like an extensive process, the process may be divided in to the following five parts:

1. Aligning the position
2. Finding the mask used on the data section
3. Unmasking the encryption type
4. Unmasking the length of the encoded message
5. Unmasking the message

Although time consuming, decoding information from a QR code by hand is possible and follows the same steps as a QR code scanner.

2.7 Information (and Ways of Presenting Information)

In 1985, Sture Allén, professor of computational linguistics, and Einar Selander, honorary doctor at Umeå University, in their book—Information on Information—defined information after having gone through "a large number of examples from texts of different kinds" [94]. Allén and Selander defined information as "a certain amount of facts or ideas". While defining the concept of information can be done, there are several ways of presenting information.

2.7.1 Text

Text is one of the oldest forms of presenting information, with written text dating back to 4000 years B.C. [79]. Text is also a simple form of presentation that does not require much high end hardware. Other forms of presenting information require more memory, more computational power and more graphical power. Text also has the advantage that users can read through text at their own pace. Text may be viewed independently of time, in contrast to sound and video.

Text does however have the disadvantage of requiring attention. The person reading the text must focus their attention on the text throughout and cannot look away in order to receive all the information being presented. Text is also restricted to the language the text has been written in. Several texts must be written in order to make the text available to people who might not have the native language in the specified country as their first language. For instance, in 2010, 44.7% of the Spanish speaking population in the United States spoke English less than "very well" [3].

2.7.2 Images

The advantage of using images as the form of presenting information is that one can show the viewer the information rather than telling the viewer the information. Showing the viewer could potentially mean that more information could be presented within a smaller space than text could achieve. Images also give the same advantage as text in terms of at what pace the viewer could perceive the information. Images, similar to text, may be viewed independently of time, in contrast to sound and video.

In a similar way to text, images require the viewers attention in order to understand the information. The viewer cannot look away from an image and still receive the information. Another disadvantage with images is the fact that images can be interpreted in different ways. The saying "a picture is worth a thousand words" goes both ways. On one hand images may present much information with one single image. On the other hand the

information may not be crystal clear and not as clear cut as a text might be.

Images may also present information in two different ways. One way is with photographs. Photographs may present abstract and/or concrete information and visualise what might be difficult to describe only using words. Another way of using images is by presenting information using graphs. Graphs are usually clearer with regard to the information they present. However, graphs may in some cases be a less useful way of presenting information. Graphs are used to present statistical information and are usually easier than photographs to translate into words. Statistical information, and as such also graphs, can summarise a period in time where as a photograph captures a moment.

2.7.3 Audio

Images and text both share the disadvantage of requiring the user's vision in order for the information to be perceived. Audio solves this problem. While audio does require the user's attention audio uses a different sense than text and images. With audio as the form of presenting information, the listener can look away and yet still receive the information that is being presented. In other words audio is well suited for multitasking as long as the other task the listener is performing does not involve listening to audio as well.

Audio does, however, have the disadvantage of having to be understood in real-time. The listener does not possess the same amount of control as he or she does with either text or images. Audio may be paused and rewound, but the fact that audio is still tied to a timeline is a disadvantage. Another disadvantage with audio is that, similar to text, audio is dependent on the language. If information were to be used on a world-wide basis, several audio files would be required (given that the audio contained spoken words) translated into different languages.

2.7.4 Video

Since video consists of many images bundled together, video gives the same advantages as images in terms of showing the viewer the information instead of telling him or her the information. Video presents the viewer with images at such speed that the images give the impression of movement. Video may also include audio. The inclusion of audio gives video all the same advantages as audio, as long as the audio is not dependent on the images (in such a case, the viewer would then not be able to look away from the video). In other words video could potentially give the advantages of two other forms of information presentation.

However, similar to audio, video is presented in real-time. The viewer is bound to the playback speed of the video. Even though a video may be paused or even rewound, the viewer does not possess the same amount of control as with images or text. With text and images the reader/viewer can deceide the pace at which the information should be perceived for themselves. If the video does not include audio, video (similar to images or text) requires full attention in order for the information to be perceived.

2.8 Summary

Google Glass was announced in 2012 along with the statement "We think technology should work for you—to be there when you need it and get out of your way when you don't." [9]. Google wanted to create a device where the user did not have to look down [13], as well as a device where the time between intention and action was minimised [95].

Google Glass is a small HMD that is partially controlled with a touchpad mounted on the right hand side of the frames. The display sits slightly above the user's line of sight, on the right hand side. Google Glass' display is a projection that goes through an optic lense, creating a virtual image which makes the perception of the display to be equivalent of a 25 inch high definition screen seen from a distance of approximately 2.5 meters [59].

Today there are many products similar to Google Glass, either already on the market or in development. Microsoft Hololens is an HMD, focused on letting the user work in a 3D space (with for instance 3D modelling [67]) by covering both of the user's eyes. Recon Jet, GlassUp and C Wear Interactive Glasses are three products more similar to Google Glass in the sense that they only display information in front of one eye. However, Recon Jet is aimed at athletes, to be used while athletes are working out. GlassUp and C Wear Interactive Glasses are meant to be connected to an external device, such as a smartphone or a PC. Google Glass is a stand-alone device, meant to be worn at all times.

The GUI of Google Glass is called a timeline and consists of a row of cards [20]. Cards are basic activities, such as a clock, but may also represent more in-depth applications (such as a game) on Google Glass called "Immersions". The center point of the timeline is the home screen and the first screen the user sees when turning on Google Glass. Cards to the left of the home screen are upcoming events, such as a flight, and cards to the right of the home screen are from the past, such as text messages. The user moves left on the timeline by swiping a finger backwards on the touchpad and in order to move right the user must swipe a finger forwards on the touchpad.

In order to play sounds, Google Glass uses a BCT which transfers sound through the

bones of the skull [59]. The advantage of this technique is that external sound is not blocked out. In order to capture the environment Google Glass is also equipped with a 5 megapixles camera [59] and a microphone. Using the camera and the microphone the user may give input to Google Glass, for instance when using voice commands in order to control Google Glass hands-free.

The hands-free experience is also what sets Google Glass appart from regular smartphones. Smartphones must be held by the user or put on a table. The user must also look down at the screen of a smartphone, in contrast to Google Glass which puts the display slightly above the user's line of sight. Smartphones do however give the user a bit more control with multi-touch and touchscreen. Another advantage of smartphones is the larger screen. Smartphone screens have been increasing in size ever since the iPhone launched in 2007 [12]. However, as HMDs increase in popularity, there is potential for a wider offering of models and screen sizes.

Smartphones and Google Glass may in many cases be used for similar applications, for instance QR code scanning, since both smartphones and Google Glass are equipped with a camera. The QR code was announced in 1994 and was developed by Denso Wave [16]. The goal was to create a new form of barcode that could carry more information than a linear barcode and be easily read. While a conventional barcode is capable of storing approximately 10 digits, a QR code can store several thousand digits [17]. A QR code also uses position fields, which make the QR code readable from any direction, compared to a conventional barcode which can only be read horizontally [18]. A QR code can be used to encode information, originally written with alphabetic characters, Japanese symbols or numeric characters [15].

Information has been defined as "a certain amount of facts or ideas" [94] and may be presented in a number of different ways. The four main ways of presenting information are text, images, sound and video. Text and images both have the advantage of being independent of time. Readers/viewers may process the information at their own pace.

Images may also be divided into photographs and graphs. The difference between the two lies in the fact that graphs are used to present statistical information. Text and images do however both share the disadvantage of requiring readers/viewers vision.

Audio solves the problem of requiring the user's vision since audio uses a different sense. The listener may as a consequence multitask in terms of listening to the information being presented through audio while performing other tasks. For instance, the listener may be listening to the radio while driving a car. In contrast to text and images, audio is not independent of time. Video has the same disadvantage as audio of not being independent of time. However, video is unique as video is the only form of presenting information which may combine images and as such show movement. Adding audio to video gives viewers the advantage of choice as they may choose to either watch or only listen.

3 Design

The application designed and implemented in parallel with this dissertation is a proof-of-concept of an application used for assembling components. The application is intended to function as a substitute for instruction manuals, where Google Glass will allow users to scroll through the instructions hands-free. A proof-of-concept of a similar application for smartphones has been designed and implemented at the same time, in order to provide a point of reference as well as help evaluate the pros and cons of using Google Glass. The application for Google Glass and the application for smartphones will look and function the same, unless otherwise mentioned, in order to help with the evaluation and comparison.

The application works as seen in Figure 3.1. First the user must use the application to scan a QR code. The QR code is then decoded. The decoded information from the QR code is an ID used to download information regarding the product connected to the QR code the user just scanned. The downloaded information contains the product name, as well as a list of the necessary components and the instructions needed to build the specific product. The product information is sorted into classes representing the product information, where attributes of the product class contains information on components and instructions. The product information is then sent to the display.

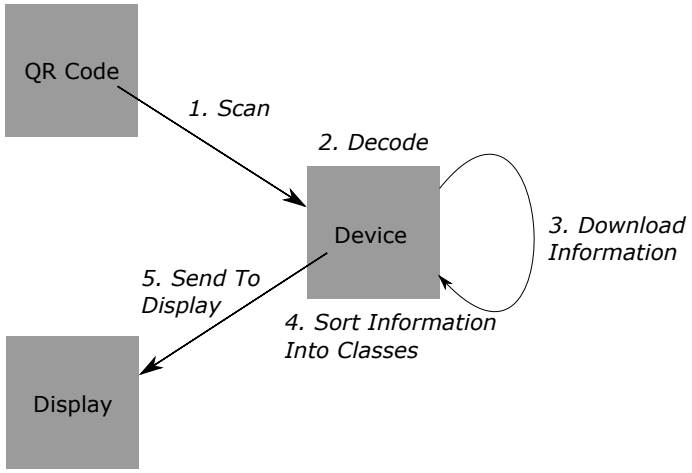


Figure 3.1: Application functionality.

As seen in Figure 3.2, the application is designed as a slide view, with only one slide being displayed at a time. Instructions are to be divided into several steps, each presented on a separate slide which users may scroll through at their own pace. On Google Glass, users may scroll through the application using voice commands in order to make the application truly hands-free. By applying the slide view design users may focus on one instruction at a time.

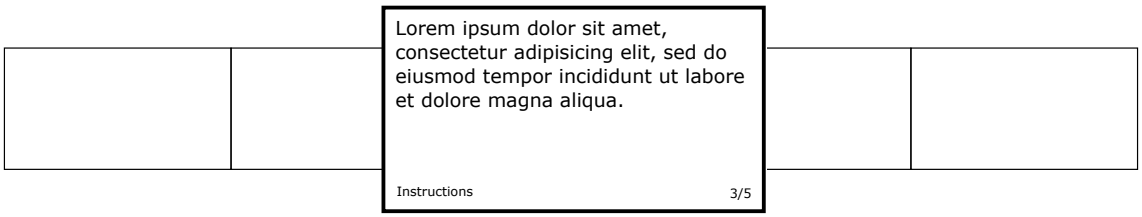


Figure 3.2: A simple sketch of the application's GUI design.

The instructions are the major focus of each slide, with each instruction using most of the slide. At the bottom of each slide additional information may be found, such as a label which specifies that the information being presented on the current slide is, for instance, an instruction (other information may also be presented, such as components required to complete the instructions). The design has been heavily inspired by Google's own guidelines [54] as to how applications for Google Glass should be designed.

3.1 Glassware Flow Designer

Google provides developers with a design tool to help them visualise applications prior to implementation. The design tool, called "Glassware Flow Designer" [51], allows developers to discover recommended design patterns and to draw out the flow of their application prior to implementation. "Glassware" comes from the fact that Google Glass applications are sometimes refered to as "Glassware" [58].

In Figure 3.3 the Glass Flow Design for the Google Glass application described in this dissertation is shown. When the application launches the user should scan a QR code.

After successfully doing so, the product name will be displayed, in order to help the user confirm that the correct instructions have been downloaded. This is followed by the slides which show the required components. After the list of components follows the instructions. The slides can be controlled either by swiping on the Google Glass touchpad or via voice commands. The next card in line is positioned to the right of the current card.

Although each card could be seen as happening in the future, and should as such be positioned to the left of the current card, it should be noted that the main menu timeline of Google Glass and the application differ in terms of how dependent the cards are of eachother. In the main menu timeline on Google Glass cards are simply related in terms of when they occur. The cards are otherwise not related to each other. In the application, however, each card is dependent on the previous card. The first instruction comes before the second, the second instruction comes before the third, and so forth. As such the cards should be scrolled through more as a story, or a book, where the the user swipes to read the next page.

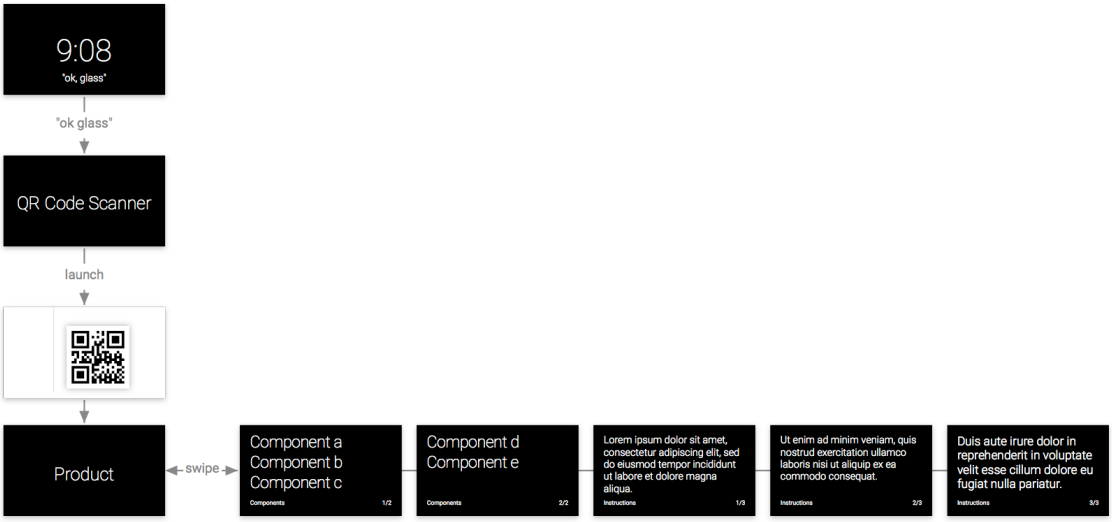


Figure 3.3: Glass Flow Design of the Google Glass application.

3.2 Presenting Information on Google Glass

As part of the design guidelines for Google Glass, Google provides developers with a card layout template, seen in Figure 3.4. The different coloured regions are intended for different types of information. The red area is the main area intended for presenting information in text form with the green squares representing the preferred margins. The thick blue line, almost at the bottom, marks the footer. The footer should hold supplementary information, such as a user name or a timestamp. The blue, slightly transparent, area to the left is mainly intended for images, with associated text presented to the right. The grey area, seemingly appearing behind all the other coloured areas, represents the entire card, with a size of 640 pixels wide and 360 pixels high [49].

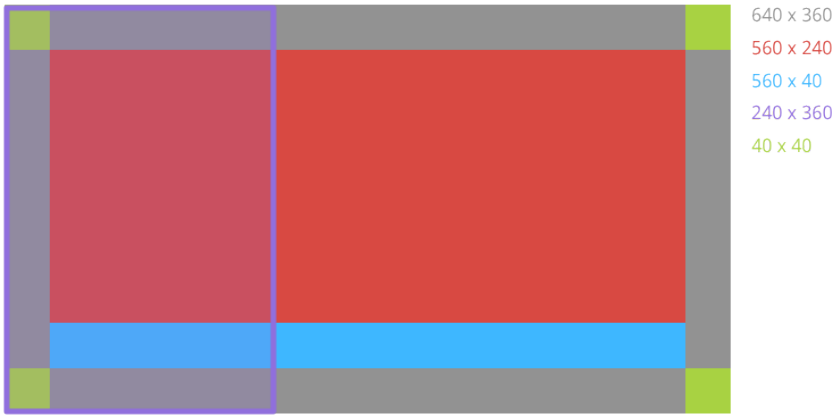
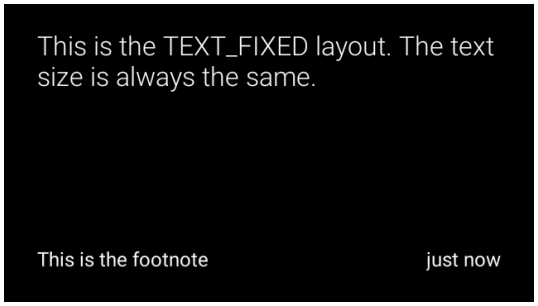


Figure 3.4: Google's design guidelines include a card layout template [49].

Google goes even further in providing developers with guidelines for the design of cards. Google provides developers with a set of fixed card layouts. Specific card layouts set up the necessary margins and leave the developers to input the information to be displayed. Four examples of fixed card layouts can be seen in Figure 3.5.

In terms of the information displayed, Google's default typeface family is called "Roboto". Google states that Roboto's geometrical forms and open curves makes for a natural reading rhythm [56]. Roboto is the typeface family used on all of Google's standard card layouts, some of which are seen in Figure 3.5. Google uses different typfaces from the Roboto type-

(a) Text layout.



(b) Fixed text layout.

(c) Columns layout.

(d) Title layout.

Figure 3.5: Four different standard card layouts [48].

face family for different texts [49]. Roboto Light is most common, with Roboto Regular being used for footnote text and Roboto Thin being used for larger texts, such as titles on the title card layout seen in Figure 3.5 (d).

One of the advantages of using Google's default layout is the fact that the text is dynamically resized to fit the card. Dynamically resized text means that the text is only as small as the text needs to be. However, there is a minimum size text may be downsized to. At 32 pixels the text is as small as the text may be and any text that does not fit on the card at that point is truncated.

Due to the limitation on the amount of text that may be presented on screen at the same time Google has provided developers with guidelines on how to present written information on Google Glass [49]. The guidelines for writing are five in total and read as follows:

- **Keep it brief.** Be concise, simple and precise. Look for alternatives to long text such as reading the content aloud, showing images or video, or removing features.

- **Keep it simple.** Pretend you're speaking to someone who's smart and competent, but doesn't know technical jargon and may not speak English very well. Use short words, active verbs, and common nouns.

- **Be friendly.** Use contractions. Talk directly to the reader using second person ("you"). If your text doesn't read the way you'd say it in casual conversation, it's probably not the way you should write it.

- **Put the most important thing first.** The first two words (around 11 characters, including spaces) should include at least a taste of the most important information in the string. If they don't, start over. Describe only what's necessary, and no more. Don't try to explain subtle differences. They will be lost on most users.

- **Avoid repetition.** If a significant term gets repeated within a screen or block of text, find a way to use it just once.

Another part of Google Glass applications, where Google has provided guidelines, is voice commands [62]. However, voice commands might be the most restrictive area of all in terms of what Google recommend developers to do. Any voice command not officially approved by Google is not allowed in an application if the application is to be released on MyGlass. MyGlass is the official store from which users may buy their Google Glass applications. However, MyGlass is not installed on Google Glass but rather on, for instance, a smartphone [30].

Only by using the approved main voice commands [64] will applications be approved and allowed on MyGlass. These approved main voice commands do not, however, include for instance "next slide". Unapproved voice commands may be used during development and for separate releases. Developers may also use the built-in speech recognition. However,

using speech recognition would mean not being able to launch the voice recognition simply saying "ok glass", which is the case with approved and unapproved voice commands.

In terms of getting a voice command approved, Google has set up a checklist for developers to cross off as they design their voice commands [61]. The checklist includes not designing a voice command which is similar to an already existing voice command. The voice command should also be long enough to ensure high recognition quality, yet short enough to fit on a single line on the Google Glass display.

3.3 Presenting Information on Smartphones

Despite being two different devices, the smartphone and Google Glass, Google's design recommendations for smartphones share similarities. For instance, similar to Google's design guidelines for Google Glass, Google recommends developers to keep information brief and to use short phrases with simple words [45].

However, in contrast to Google's design guidelines for Google Glass, Google does give developers freedom to make their own decisions. "Deviate with purpose", as Google states. Google recommends developers to develop applications which are easy and fun to use.

One design guideline for applications on smartphones provided by Google is to highlight what is most important in the application. For instance, in a camera application the shutter button is the most important button. As such the shutter button should be the most prominent component in the application and easy to find, similar to Figure 3.6, making the application easy to use in terms of the core feature.

In terms of clear cut differences between designing applications for smartphones compared to designing applications for Google Glass the most important one is perhaps where the user touches on the device. The Google Glass touchpad is mounted on the right hand side of the user, while the display sits in front of the user. On a smartphone the display and the touch-area is the same area. As such, buttons and other similar, intuitive, touch-objects are well suited for applications on smartphones. On Google Glass the user does not

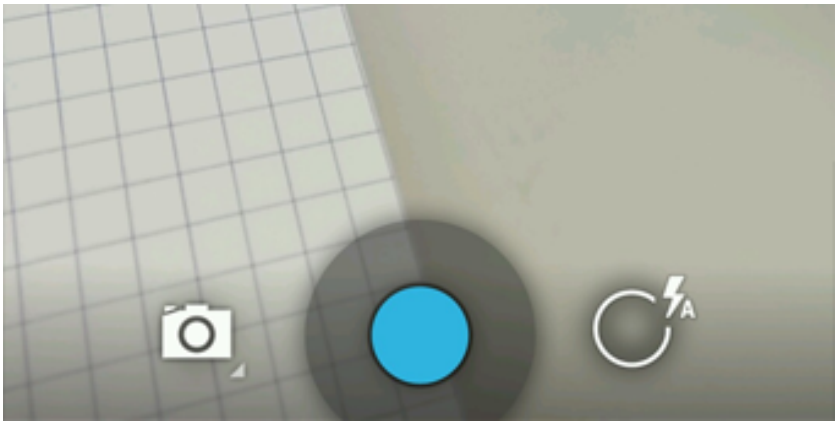


Figure 3.6: The most important component should be the most prominent [45].

have any pointer on the screen, meaning that, for instance, menus can not be dependent on the user selecting an option by touching the option on the screen.

The use of voice command on Google Glass does help menus on Google Glass seem more similar to those on a smartphone since the users may select an option with their voice. However, the use of buttons and icons is less practical. On the other hand, since Google Glass may be controlled by voice command, developers might also hide functionality and decide not to show buttons and icons on the screen since the user does not need a target to touch in order to interact with the application.

One example of where Google Glass might hide functionality is in a camera application. In Figure 3.6, part of a camera application for smartphones is shown, with three buttons on screen. An application for Google Glass would not need to display any buttons since the users need only to say what they want to do. If, however, the Google Glass user did not use the voice command functionality, a simple touch of the Google Glass touchpad might for instance bring up a menu, displaying possible options, similar to how saying "ok glass" would bring up the voice menu.

All the above guidelines and restrictions have been taken in to consideration when designing the application for smartphone as well as Google Glass. While the application does have the same functionality, the most noticeable difference between the two applications

is that the smartphone application have buttons, enabling the user to jump forward, past slides to reach, for instance, the instructions, where as in the Google Glass application such features have been limited to voice commands.

3.4 Test Cases

The following aspects of the application are to be tested in order to help determine whether Google Glass is a viable option to use as replacement for an instruction manual when assembling components. Each test will be performed 30 times, for statistical significance [71].

3.4.1 Text Length

Since the Google Glass display is small and limited in space, the amount of text that may by displayed on screen is as a result also limited. As such, one interesting test case is to see where the text limit lies. The test consists of trying to find the text length limit, on both the Google Glass display, as well as a smartphone display. The text used should consist of characters distributed in similar fashion to English text [1].

When the smartphone display has been filled with information, how many slides does the same amount of information require on Google Glass? If the number of slides that the Google Glass application must use in order to display all the information is significantly greater than that of the smartphone application, the use of a Google Glass application might not be preferred as the user must potentially scroll through more slides than when using a smartphone equivalent application.

3.4.2 Distance to the QR Code

Generally, the recommendation regarding at what distance the user should be positioned, in relation to the QR code, is the size of the QR code times ten [4]. However, different devices will have different delay time before registering the QR code in frame. As such, one test regards the time from the start of the camera until Google Glass has registered the QR code. The time is to be compared with that of smartphones.

Three different distances will be tested. The size of the QR code will be optimised according to the formula stated above, with the scanning distance set at two decimeters. The reason for testing for other scanning distances, yet with the same QR code size is because most users might not be aware of the optimal scanning distance, as well as to determine whether Google Glass has any advantages when scanning either closer to or further away from the QR code.

The size of the QR code is calculated as

$$\frac{2}{10} = 0.2 \; decimeters$$

.

3.4.3 Complexity of the QR Code

Depending on the number of characters encoded by the QR code, the density of the QR code changes. The density of the QR code increases as the number of characters encoded by the QR code increases. In other words, the number of black and white squares increase. As such, one interesting test case is where the variable is the density of the QR code, or more specific; the number of characters encoded in the QR code.

The number of characters will vary between 1, 50 and 100. The values were chosen in order to give the results big enough room so that potential difference can be determined while still keeping the number of characters to a realistic minimum.

3.4.4 Display Time

The speed at which Google Glass registers the QR code is important to whether the device will gain preference over regular smartphones. However, another interesting aspect is how fast downloaded information may be displayed on screen, from the point that the information has been downloaded. This test will evaluate three different information sizes, meant to represent three different ways of presenting information. 100 kilobyte represent text, 1 megabyte represent an image and 10 megabyte represent video.

3.5 Test Units

- Google Glass
- Samsung Galaxy SII
- Samsung Galaxy SIII

The reason for using these three units was the fact that the testing required physical devices. Since the testing included scanning a QR code from a various distances, physical devices were necessary. The use of the two specific smartphone models (the Samsung Galaxy SII and the Samsung Galaxy SIII) was partially due to availability during testing. However, another reason was the fact that both models were among the most widely distributed smartphone models during the time of Google Glass' release. The Samsung Galaxy SII was released in 2011 and the Samsung Galaxy SIII in 2012, with Google Glass being released in 2013. The Samsung Galaxy SII and the Samsung Galaxy SIII have in total sold 100 million units to date [21, 25].

Another reason for not using more modern smartphones was due to the Google Glass version used. The Google Glass unit used was the so called "Explorer Edition 1". Google Glass Explorer Edition 1 was released in February, 2013 [38]. An updated version, "Explorer Edition 2", was released in the summer of 2014 [31].

A few updates were made to Google Glass with the new version, most noticeably a doubling of available RAM. Google Glass Explorer Edition 1 has only 1 GB RAM [22] where Google Glass Explorer Edition 2 has 2 GB RAM [31]. The Samsung Galaxy SII and the Samsung Galaxy SIII both have 1 GB RAM as well [87, 88]. As such using more modern smartphones in testing could be seen as unfair to Google Glass since Google Glass also exists in a more modern version.

The Samsung Galaxy SII and the Samsung Galaxy SIII both run Android as their operating system [87, 88]. Since Google Glass' operating system is also Android [80], the development of the application was deemed to be easier and faster, not having to convert specific functionality to iOS or Windows Phone. Comparing the devices would also be more easily done if the operating system were the same. All of the test devices having Android as their operating system also meant that they would all be able to follow Google's design guidelines, as opposed to, for instance, Apple's design guidelines and principles [35].

Additional technical specifications regarding the test units can be found in Table 3.1.

Table 3.1: Technical specifications of the three test units.

Component	**Google Glass [80]**	**Samsung Galaxy SII [87]**	**Samsung Galaxy SIII [88]**
RAM	1 GB [22]	1 GB	1 GB
CPU	OMAP 4430 dual core [23]	1.2 GHz dual core ARM Cortex A9	1.4 GHz quad core Cortex A9
Screen Size	Equivalent of a 25 inch screen from 2.5 meters [59]	4.3 inches	4.8 inches
Screen Resolution	640 * 360 pixles	480*800 pixles	720*1280 pixles
Operating System	Android	Android	Android

3.6 Summary

The application designed and implemented in parallel with this dissertation is a proof-of-concept of an application used for assembling components. The functionality of the application may be divided in to several steps. First the application scans a QR code. The QR code is then decoded and the decoded information from the QR code is used in order to download instructions on the specific product belonging to the QR code which was scanned. The downloaded instructions are sorted into classes and displayed on screen.

The application was designed to be run on Google Glass, however a proof-of-concept of a similar application for smartphones has been designed and implemented as well. The smartphone application provides a point of reference as well as means to evaluate the pros and cons of using Google Glass. The application is designed as a slide view, with each instruction appearing on a separate slide. The slide view design was used so that users may focus on one instruction at a time.

The design of the application also follows the design guidelines provided by Google [54]. For instance, Google recommends developers to keep their application and the content within the application simple. Google recommend developers of Google Glass applications especially to mind the small screen and as such to keep the information brief, and the most important information first.

Google also provides developers, designing applications for Google Glass, with even more specific recommendations, as Google recommend using the predefined card layouts [48]. The predefined card layout sets up the recommended margins and leaves developers only to input the information which should be displayed on the specific card.

The design guidelines for smartphone applications are not as specific, although Google does recommend keeping information simple [45]. Google also recommend developers to highlight the most important feature of an application. One example of how the guidelines have been taken into consideration when designing the application is how both the Google Glass application and the smartphone application starts in camera view, where the user

may scan a QR code. There is no start menu, instead the most important feature, the camera screen, is the first screen the user sees when launching the application.

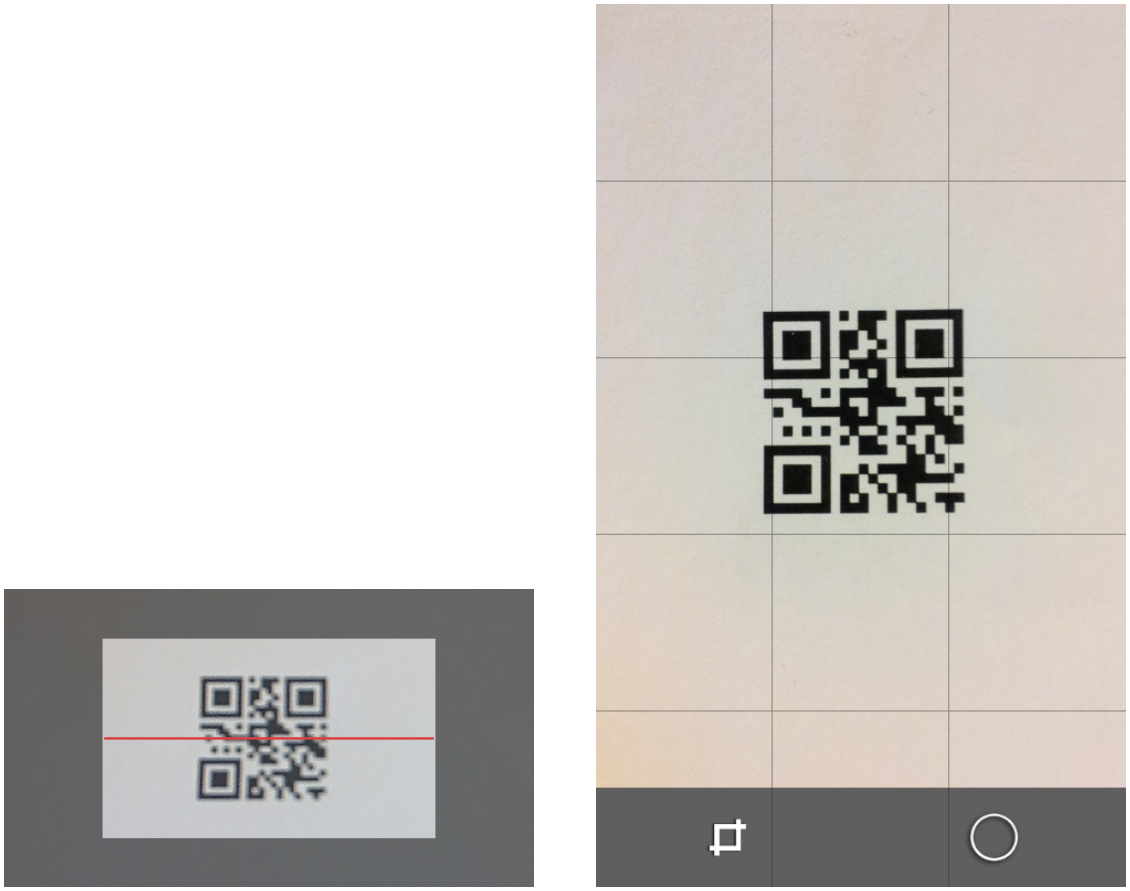
The application will also be tested, both on Google Glass as well as the smartphones. One test case is text length, as one limitation of Google Glass is the small display. The test will consist of maximising the number of characters that may fit on one slide in the smartphone application, and comparing the result with how many slides the same number of characters would require in the Google Glass application.

Another test compares the distances from the device to the QR code. Are there any differences between the smartphone application and the Google Glass application when the distance to the QR code is altered? The complexity of the QR code is another test that will be performed. Does the complexity of the QR code, which increases with the number of characters encoded, have any impact specific to either the smartphone application or the Google Glass application? The fourth, and final test, to be performed measures the display time. Does the display time differ between the smartphone application and the Google Glass application for different sizes of information?

All of these tests will be performed 30 times for statistical significance [71], and all of these tests will be performed on Google Glass as well as the smartphones, the Samsung Galaxy SII and the Samsung Galaxy SIII. The reason for using the two specific smartphone models was for one the need of physical devices used for testing, but also the fact that these two smartphones were very prominent on the market when Google Glass was released in 2013. The Samsung Galaxy SII and the Samsung Galaxy SIII have sold a total of 100 million units and are as such good representatives of smartphones comparable to Google Glass [21, 25].

4 Implementation

As the application launches, the first screen the user sees, in both the Google Glass version and the smartphone version, is the camera screen. The user must, in order to proceed further within the application, scan a QR code. Scanning a QR code is done by positioning the device's camera in such a way that the QR code can be seen on screen, as seen in Figure 4.1. The user does not need to press any shutter button as the application automatically recognises the QR code pattern if seen on screen.



(a) The Google Glass application.

(b) The smartphone application.

Figure 4.1: The application is scanning a QR code.

The reason for not providing a menu on the start screen, or even requiring the user to

press a shutter button, is because the application should be simple, easy to use and focus on what is important. Since the the first step when using the application is to scan a QR code in order to receive the necessary information on the specific product, the scanning is also the main focus on the first screen of the application.

When the QR code has been scanned the application decodes the QR code. The decoding process is handled by the Zebra Crossing (ZXing) library [91]. ZXing is an open source barcode image processing library which uses the default QR code reader installed on the device.

The smartphone application was based directly on the ZXing library, where as the Google Glass application was based on a port of the ZXing library to Google Glass, called BardcodeEye [26]. The main difference between ZXing and BarcodeEye is the fact that BarcodeEye is an example application ready to be run on Google Glass, in contrast to the ZXing library which is a library and as such needs to be attached to a runnable application.

The BarcodeEye application for Google Glass is, however, a bare bone application, used as an example and introduction on how ZXing may be implemented in an application for Google Glass. BarcodeEye displays the decoded information from the QR code and also gives the user the option to search the internet using the information previously decoded from the QR code.

As the QR code is meant to encode only a product ID, and the application will then use the ID to download the product information (rather than having all of the instructions encoded directly in the QR code), the BarcodeEye application had to be modified. The first modification was on the graphical layout of the BarcodeEye application. The change of layout was mostly done due to the fact that the BarcodeEye application only displayed plain text, not taking in to account for instance a mix of image and text.

In order to display information, BarcodeEye used the deprecated class `Card`, as seen in Listing 4.1. The application now instead uses the `CardBuilder` class, as seen in Listing 4.2, as recommended by Google [42]. The `CardBuilder` class allows users to input a desired

layout style as an argument to the constructor of the `CardBuilder` class. With the `Card` class, developers could (in a separate method call) set where on the card the image would appear. For instance, in order to create a columns card, as seen in Figure 4.7, the code would look as the code in Listing 4.1. Using the recommended `CardBuilder` class instead would look as the code in Listing 4.2. With the `CardBuilder` class, developers can now use default layouts, enabling more consistent application design across Google Glass applications. If a developer wishes to use a custom layout instead, the developer can simply use the custom layout as input to the `CardBuilder` class' constructor instead.

Listing 4.1: Instancing of the deprecated class Card

```
1 Card card = new Card(context);
2 card.setImageLayout(Card.ImageLayout.LEFT);
```

Listing 4.2: Instancing of the recommended class CardBuilder

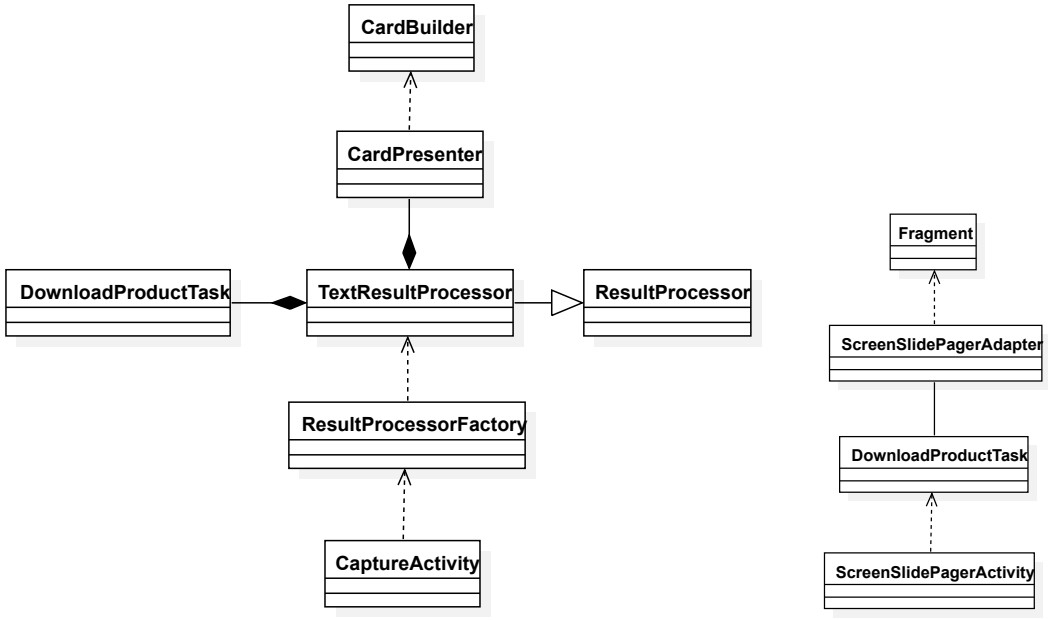
```
1 CardBuilder cardBuilder = new CardBuilder(context, CardBuilder.Layout.COLUMNS);
```

Since the smartphone application also used the ZXing library, but without any pre-existing application, no changes similar to those made to the Google Glass application had to be made for the smartphone application. Instead the smartphone application was built from the beginning to make use of the ZXing library's functionality, similar to how the ZXing library was integrated in the base Google Glass application.

The Google Glass application and the smartphone application are similar in how they are built. However, they look a bit different from each other, as seen in Figure 4.2, which shows a UML diagram over the slide view part of the respective applications. The Google Glass version has a few extra classes involving result processors.

The result processors come from the base version of the Google Glass application, where more specific information was expected to be encoded in the QR code. For instance, the base version contained an international standard book number (ISBN) [83]

processor for books, which was not needed in the application described in this dissertation. In the base version, the `ResultProcessorFactory` returned a specific result processor depending on which information was to be processed. However, in the application only the `TextResultProcessor` is used and `TextResultProcessor` is as such also the only remaining `ResultProcessor`. The Google Glass version should be rebuilt so that the `DownloadProductTask` replaces all three of the result processor classes seen in Figure 4.2. However, such refactoring has not been done at this point.



(a) Google Glass. (b) Smartphone.

Figure 4.2: UML diagrams.

In Figure 4.2 (b), the class `Fragment` can be seen. The `Fragment` class is actually only an interface which is implemented by several other classes, however the UML diagram in Figure 4.2 (b) has been simplified as all classes inheriting the `Fragment` class essentially do what the `CardBuilder` class in the Google Glass application does, which is setting up the specific layout of the slide. In the smartphone application there is a class for each different layout, in comparison to the Google Glass application where the `cardBuilder` class handles the layout through argument, as seen in Listing 4.2. As such another potential refactor

which could be made is to create a general class that inherits the `Fragment` class and takes the desired layout as an argument, similar to the `CardBuilder` class in the Google Glass application. Since the `CardBuilder` class is a Google Glass specific feature the `CardBuilder` class could not be used in the smartphone application.

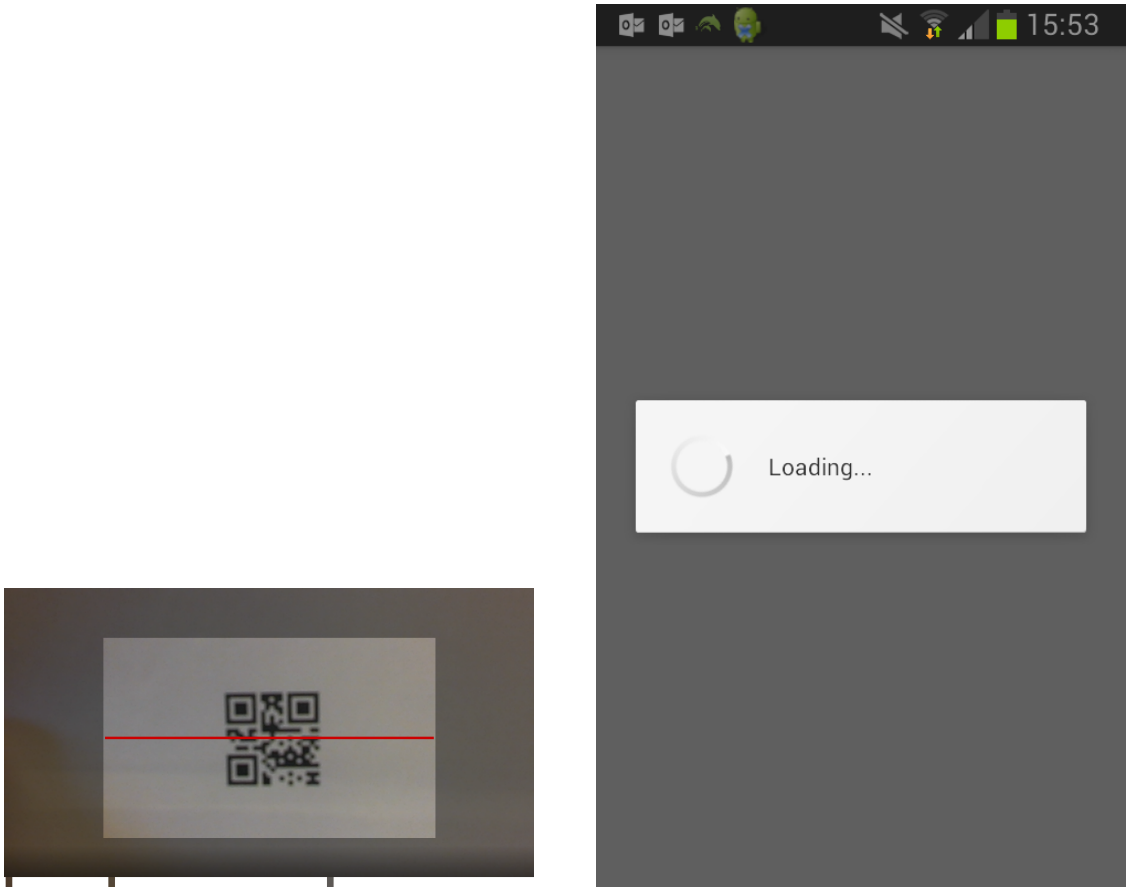
Having scanned and decoded a QR code the decoded information is then used to download information about the specific product. The downloaded information contains the product name, as well as the necessary components and instructions for assembling the product. The components and instructions may be represented by text, images or both.

The way the download process uses the decoded product ID is by concatenating the product ID with a URL adress, which connects to a database containing all necessary information regarding the specific product. The download process was done using a `AsyncTask`. An `AsyncTask` is a class which performs operations in the background on a different thread than the user interface [47]. As such, the user interface is not frozen when the download process is taking place.

In the Google Glass application, a loading bar pops up at the bottom of the screen when information is being downloaded, as seen in Figure 4.3 (a), indicating that the application is loading. Google calls the bar an "Indeterminate Slider", as the the class used in the implementation is the `Slider` class [57]. A similar animation was added to the smartphone application. However, in contrast to the Google Glass application's loading bar, the smartphone application displays a spinning wheel at the center of the screen, seen in Figure 4.3 (a), indicating that the application is loading. The spinning wheel was implemented using the `ProgressDialog` class [55].

In both cases of loading animation the animation is started in the `AsyncTask`, just before the download begins. The loading animation stops when the `AsyncTask` has finished downloading the product information.

The download process also includes the creation and initialisation of an instance of the `Products` class. The instance contains the name of the product and an image of the



(a) The Google Glass application (b) The smartphone application.

Figure 4.3: The loading screens.

product as the product will look when the user is finished assembling all the components (the existence of an image is dependent on whether there was an image of the product stored in the database or not).

The `Products` class instance will also contain a list of components, as well as a list of instructions. Both components and instructions are classes themselves. Similar to the `Products` class, instances of both the `Components` class and the `Instructions` class will contain a string and potentially an image, depending on whether there is an image stored in the database or not. In the case of components the string attribute will contain the name of the component, in contrast to instances of the `Instructions` class where the

string instead will contain the instruction itself.

When the downloaded information is displayed, the first screen the user sees contains the product name as well as an image of the product (if an image existed in the database). In the example case, Lego parts are to be assembled in order to make up the so called "Space Pirate", seen in Figure 4.4.

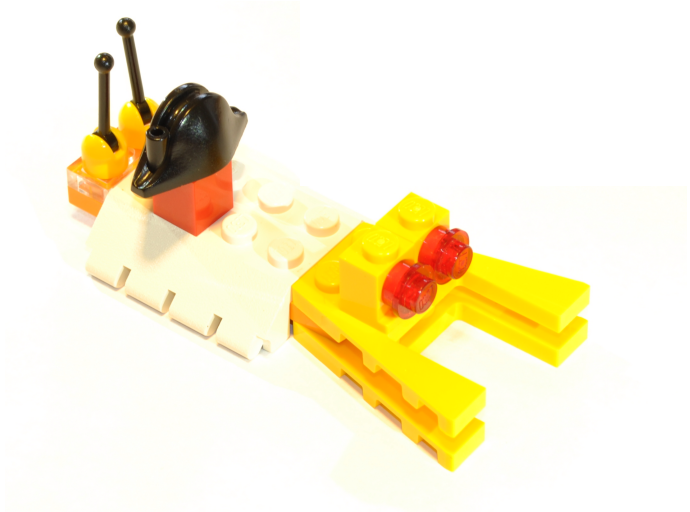
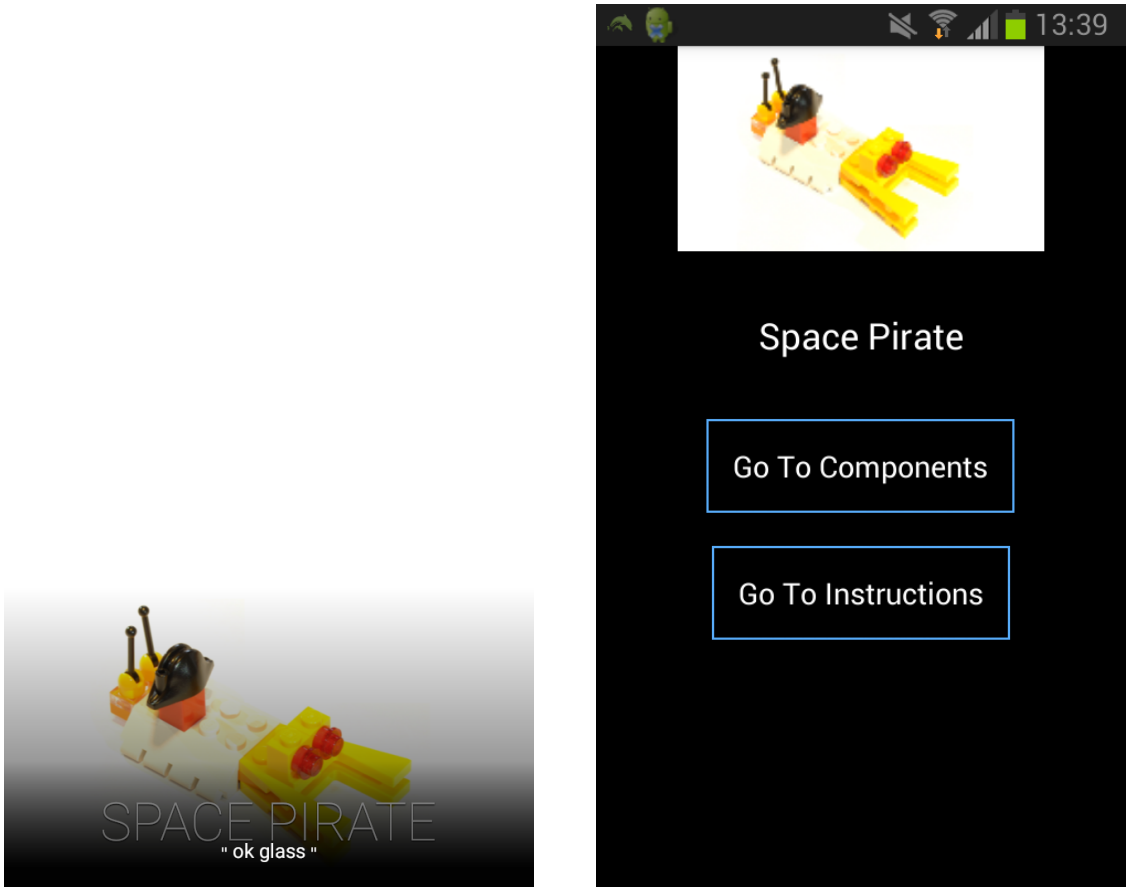


Figure 4.4: The product.

The first information the user sees displayed on screen, after the QR code has been scanned and the information has been downloaded, is the title slide for the Space Pirate product, seen in Figure 4.5.

The next slide in line after the title slide is the first slide containing information on the components necessary for assembling the specific product. Each component has its own slide as the component may contain an image in combination with the name of the component. Examples of a component described in only text can been seen in Figure 4.6, and examples of when the component has been described with both text and image can be seen in Figure 4.7.

As seen in Figure 4.6, the text in the Google Glass application is larger than the text in the smartphone application. The difference in size is due to the automatic scaling in

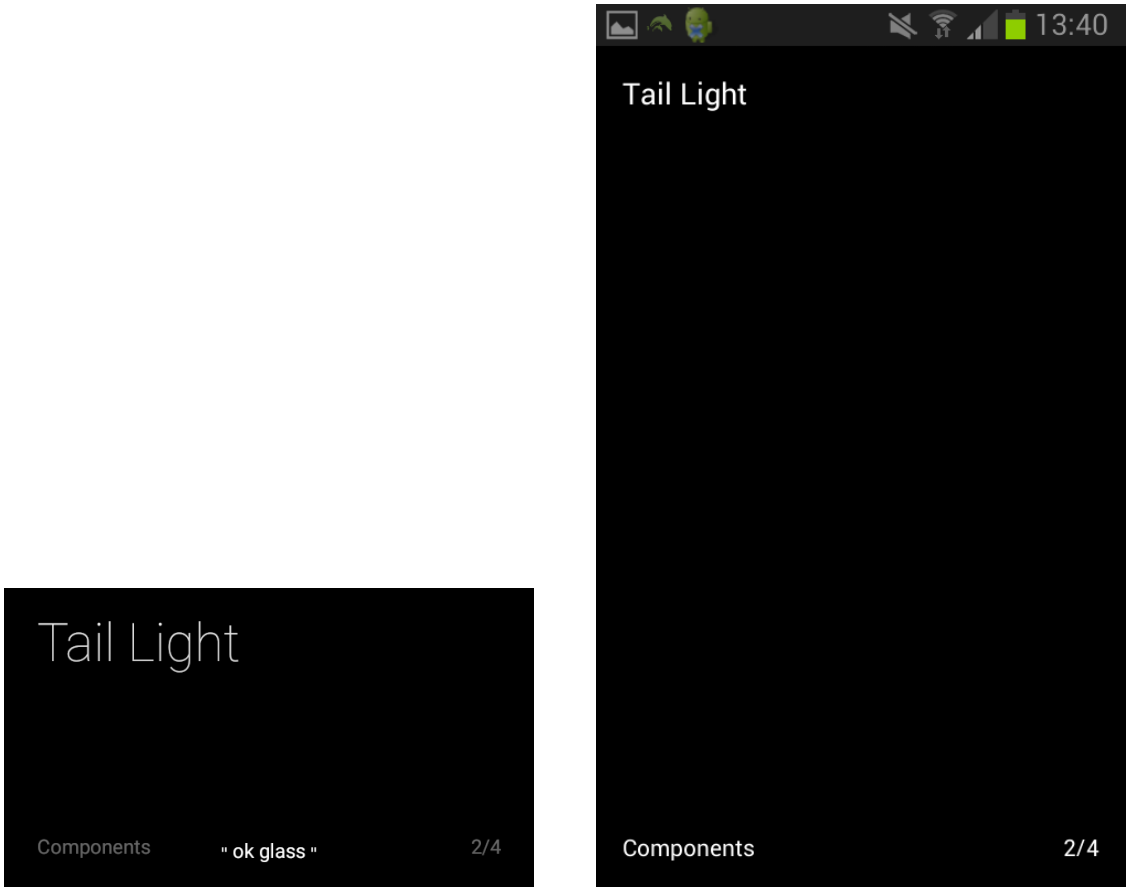


(a) The Google Glass application

(b) The smartphone application.

Figure 4.5: The title card of the application.

the Google Glass application. If the text in the Google Glass application were to cover the entire screen the text size would automatically be downscaled. In the smartphone application, however, the text is always the same size. The size of the text in the smartphone application was set to `MEDIUM` which is one of the four different default sizes of text used in android applications [60]. The footer text, which in Figure 4.6 says "Components", as well as "1/4", is of the default size `SMALL`. In Figure 4.5 the size of the product name text is of the default size `LARGE`. The reason behind the choice of the different text sizes is to keep the smartphone application similar to the Google Glass application, making the comparison of the two version easier.



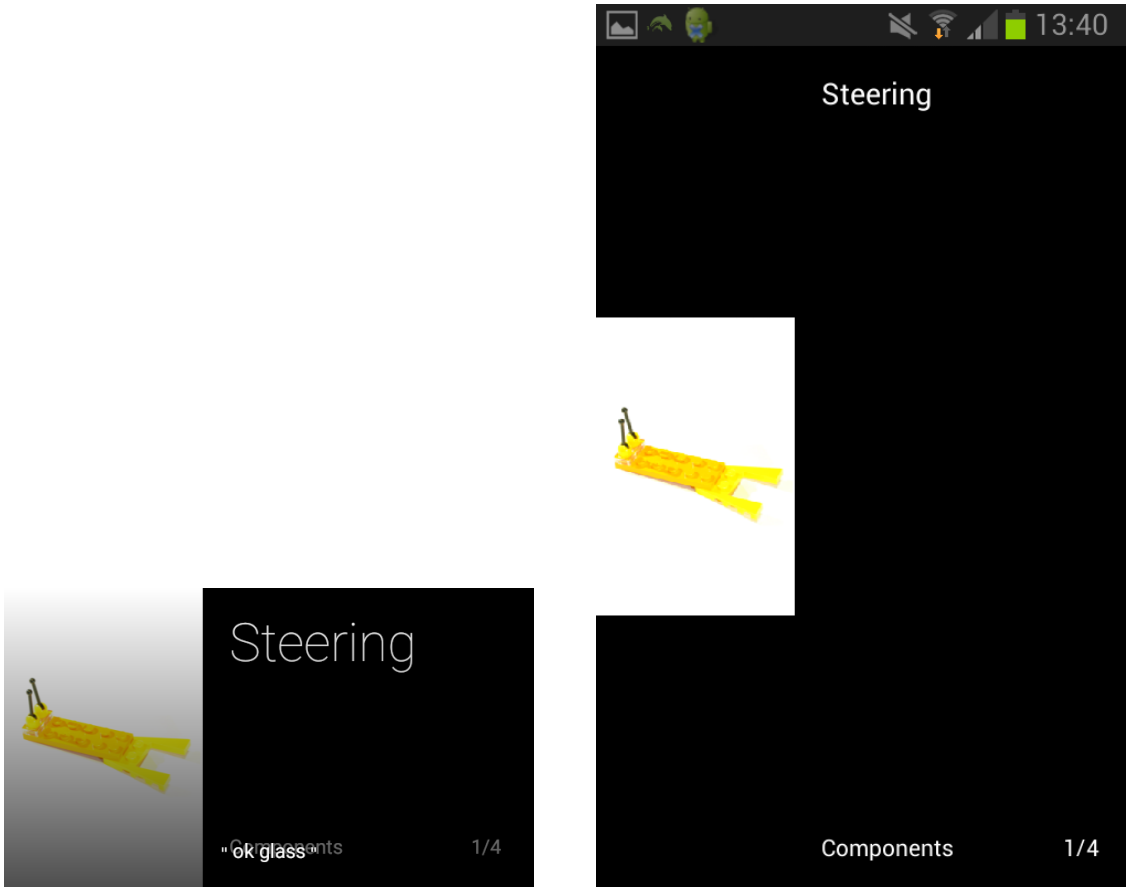
(a) The Google Glass application

(b) The smartphone application.

Figure 4.6: A component slide from the application.

The card design used for the component slide containing only text, seen in Figure 4.6, is the predefined `TEXT` layout [48], used in similar fashion to how the title card design was specified in Listing 4.2. A component slide may also contain an image as complement to text. The card design used for the component slide containing both text and image, seen in Figure 4.7 is the predefined `COLUMNS` layout [48], used in similar fashion to how the title card design was specified in Listing 4.2

After all the component slides follow the instruction slides. These slides may also contain either text only or both text and an image, and these layouts are specified the same way as the component slides. However, an instruction slide may also consist of only



(a) The Google Glass application

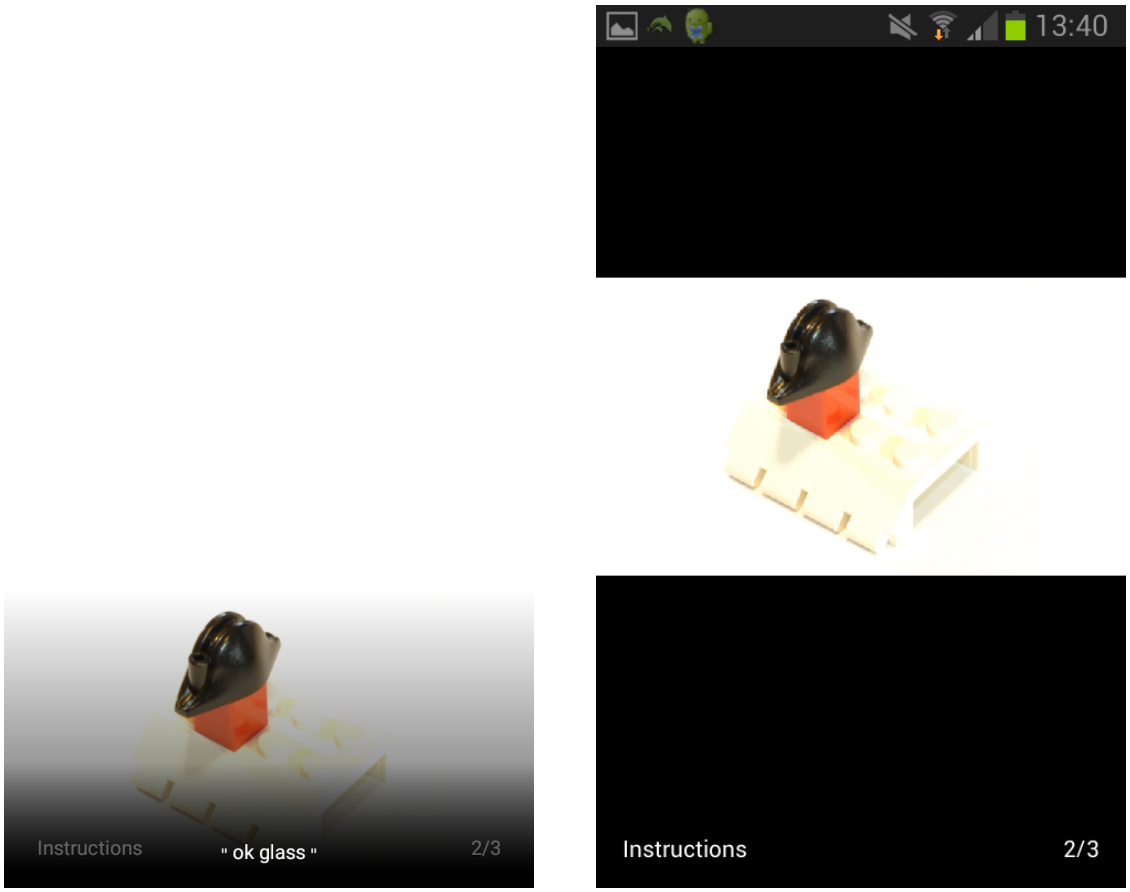
(b) The smartphone application.

Figure 4.7: A component slide from the application.

an image, and no text at all. The reason for having an image only layout as a possible card layout for an instruction slide is because an instruction may be best described with an image. Describing the same instruction in text may take up more slides than the image, as the image will only take up one slide. A slide containing only an image and no text will also mean that the image will be shown in a larger scale than when using both text and an image. As such more detail can be shown and users may get a better understanding for how the components should be assembled.

As seen in Figure 4.8, one instruction is described with only an image and no text. Since the placement of the components are important, and potentially hard to specify in text,

an image may describe the placement better. Note also that the smartphone application still has room for text. Potentially the image in the smartphone application could be expanded, for instance by giving the user the ability to zoom by pinch. However, such a feature does not exist. In the Google Glass application, the image only card uses the predefined `CAPTION` layout [48].



(a) The Google Glass application (b) The smartphone application.

Figure 4.8: An instruction slide from the application.

The Google Glass application may be controlled by swiping across the Google Glass touchpad, but the Google Glass application may also be controlled using voice commands. The voice commands can be used as simple replacement for the touchpad, where users may swipe between cards as expected in a slide view. However, using voice commands users

may also “jump” in the application. By for instance saying “ok glass, show components” the application jumps to the first component slide, regardless of which slide the user is currently viewing. If the user is currently already viewing the first component slide nothing will happen. The voice command menu of the Google Glass application can be seen in Figure 4.9. Although the smartphone application does not have any voice command features the title slide contains buttons, as seen in Figure 4.5 (b) giving the user the ability to skip straight to the instructions if so desired.

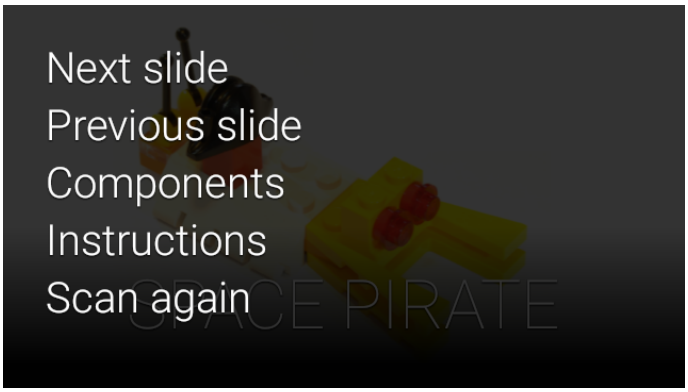


Figure 4.9: The voice command menu.

4.1 Android Studio

Both the smartphone application, as well as the Google Glass application, were developed in Android Studio [46]. Android Studio is a development environment developed by Google. Both applications were initially being developed in Eclipse [37], however development soon shifted to Android Studio as Android Studio is now the official integrated development environment (IDE) for Android [43]. The change was made without complications as Android Studio contains an import feature, enabling developers to import projects previously not developed in Android Studio [43].

4.2 API level

The API (application programming interface) level is an attribute of Android applications which identifies the required API level of a device attempting to run the specific application [66]. Each API level contains a certain set of features, which must exist on the device in order for the device to be able to run a specific application.

The Google Glass application has the API level 19, as Google started distributing the Glass Development Kit (GDK) with API level 19, enabling developers to use Google Glass specific features and to develop applications for Google Glass [50]. The smartphone application, on the other hand, uses the API level 12, as no Google Glass specific features were used in the smartphone application.

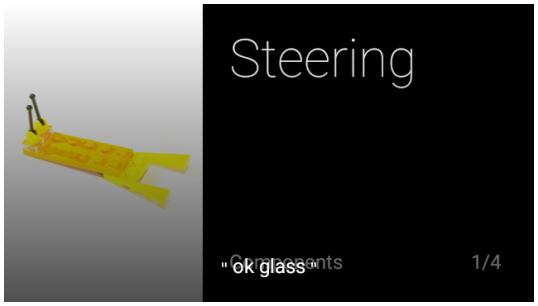
Android applications also have a target API level, describing the API level best suited for the application, as some changes might have been made in the API to the functionality used in the application. The Google Glass application has the target API level 19, which is the same level as the required API level. The smartphone application has the target API level 22, as that was the latest API level released as of writing this dissertation [44].

4.3 Card Layout

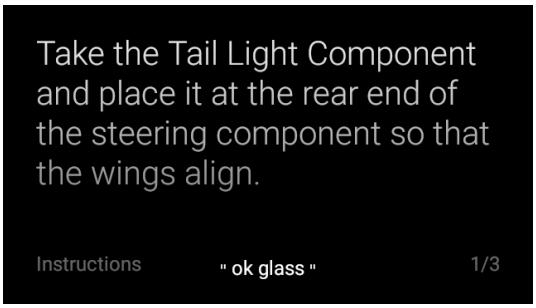
Google provides developers with a set of predefined layouts for different types of cards, which were used in the Google Glass application and used as basis for the design of the different layouts for the slides in the smartphone application. The following predefined layouts were used in the implementation: “Title”, “Columns”, “Text” and “Caption”. The Title layout was used for the first card of the slide view, seen in Figure 4.10 (a), which shows the product name as well as an image of the product as it is supposed to look when finished.

The Columns card layout, seen in Figure 4.10 (b) was used for when an instruction or component was to be presented with both text and an image. Since the Columns layout

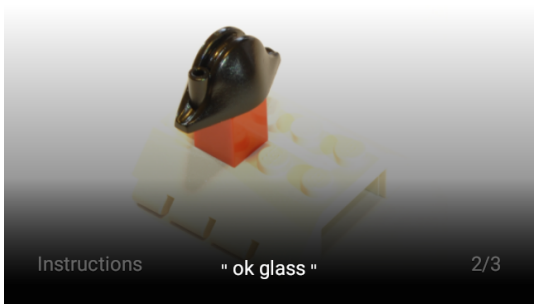
(a) The title card layout.



(b) The column card layout.



(c) The text card layout.



(d) The image card layout.

Figure 4.10: The different layouts used within the Google Glass application.

split the card, with an image to the left and text to the right, the Columns layout was the most reasonable choice when presenting both text and an image. An alternative would have been to display the text on top of the image. However, then the image could potentially have been hidden behind a larger amount of text. The title card used both text and image as the specifics of the image was not as important, and the text was only the product name, which should not be long enough to cover up the entire screen.

If the information, either a component or an instruction, were to be presented as text only, the Text layout, seen in Figure 4.10 (c) was used. The Text layout displayed dynamically sized text. In other words, if there was a lot of text being displayed the text would be resized to fit the screen.

If the card were to only contain an image, and no text, the Caption layout was used, seen in Figure 4.10 (d). The Caption layout is similar to the Title layout, but in contrast to the Title layout the Caption layout also has both a footer and a timestamp at the bottom of the card on the left and right side respectively. The Caption layout enables the use of an actual caption. However a caption is not required by the Caption layout and neither was a caption used in the application as the Caption layout was used when no text was meant to be displayed (not counting the footer and the timestamp which appears on all cards except for the title card).

Using the predefined layouts, the desired layout design was easily achieved as the process consisted mostly of plug-and-play. The `CardBuilder` class' constructor took the layout as an argument, as seen in Listing 4.3. When an instance of the `CardBuilder` class was created, what remained was to simply input the necessary information, such as the instruction text. Setting an image was done slightly differently than written information as images were loaded in using a separate thread. As soon as the `CardBuilder` method `getView` was called, the card was built with the information that had been input.

Listing 4.3: Initialisation of the CardBuilder class

```
1 CardBuilder cardBuilder = new CardBuilder(context, CardBuilder.Layout.COLUMNS)
2    .setText(getText())
3    .setFootNote(mFootNote)
4    .setTimestamp(mTimeStamp);
5
6 cardBuilder = (new LoadImage(isTitleCard(),
     getByteArray()).doInBackground(cardBuilder));
7
8 return cardBuilder.getView();
```

4.4 Voice Commands

The Google Glass application gives users the option to use voice commands in order to navigate the slides. The user opens the voice command menu by saying "ok glass", at any point in the application, when "ok glass" is shown at the bottom of the screen. The voice command feature is available at all times, except when the camera is active. In other words the voice commands are unavailable when the application is waiting to scan a QR code.

The voice command menu contains the following options.

- **Show next slide**

  The application scrolls to the next slide. If the current slide is the last slide, and in other words no other slides are following, the application does nothing.

- **Show previous slide**

  The application scrolls to the previous slide. If the current slide is the first slide, and in other words no other slides comes before the current slide, the application does nothing.

- **Show components**

  The application scrolls to the first slide showing information on a component. If the user is currently on the first slide showing information on a component the application does nothing.

- **Show instructions**

  The application scrolls to the first slide showing an instruction. If the user is currently on the first slide showing an instruction the application does nothing.

- **Scan again**

  The application launches the camera and expects the user to scan another QR code.

Implementing voice commands in the Google Glass application was done by following Google's step-by-step guide on how to implement voice commands in Google Glass applications [63]. Listing 4.4 shows the XML written according to Google's step-by-step guide, which gives the voice commands used in the application. Using contextual voice commands, according to Google's step-by-step guide, means that "ok glass" is displayed at the bottom of the screen at all times when voice commands are available. "ok glass" also comes with a dark overlay, seen, for instance, in Figure 4.10, which is transparent, yet darkens the slides, especially near the bottom of the slides where "ok glass" is shown. Although the dark overlay could potentially distort the slides a bit, especially when the slide contains only an image and no text, altering the "ok glass" overlay in any way was not possible at the point of writing this dissertation [32, 33]. The dark overlay does, however, ensure that "ok glass" is always visible, no matter what the background image looks like.

Although none of the voice commands has been sent in for official approval by Google, most of the voice commands follow the design guidelines provided by Google [61]. As seen in Table 4.1, 11 of the 15 voice command guidelines provided by Google have been followed. However, some of the guidelines have been applied to some or most of the voice command used within the application, but not all. For instance, "Show components" does not follow the first guideline of the voice command checklist, yet the "Show components" voice command remains a part of the application as "Show components" is a key feature of the application.

Listing 4.4: The voice command menu XML file

```
<menu xmlns:android="http://schemas.android.com/apk/res/android">
   <item
      android:id="@+id/next_menu_item"
      android:title="Show next slide" >
   </item>
   <item
      android:id="@+id/previous_menu_item"
      android:title="Show previous slide" >
   </item>
   <item
      android:id="@+id/components_menu_item"
      android:title="Show components" >
   </item>
   <item
      android:id="@+id/instructions_menu_item"
      android:title="Show instructions" >
   </item>
   <item
      android:id="@+id/scan_menu_item"
      android:title="Scan again" >
   </item>
</menu>
```

Table 4.1: Voice Command Checklist [61].

	Guideline	**Acheived**
1	Is general enough to apply to multiple Glassware, but still has a clear purpose	Yes
2	Is colloquial and can explain Glass features in a conversation	Yes
3	Is comfortable to say in public	Yes
4	Brings the user from intent to action as quickly as possible	Yes
5	Avoids brand words	Yes
6	Is long enough to ensure high recognition quality (at least three syllables)	Yes
7	Fits on a single line	Yes
8	Does not sound similar to existing commands	Yes
9	Does not require immediate interactivity in Mirror API Glassware.	Yes
10	Has an imperative verb with an object	Yes
11	Uses articles when possible	No
12	Uses definite articles only when the object is definite	No
13	Uses “this” when there is only one relevant instance of the object	No
14	Uses me and my when appropriate	No
15	Refers to Glass as the subject carrying out the action	Yes

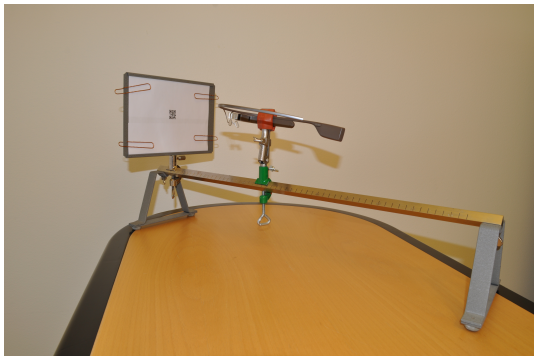
The reason for not following guidelines 11–14 was because they would make the voice commands longer. As the voice commands could potentially be said often while using the application, as the user may proceed through the slides quite quickly, shorter voice commands make for more comfortable use. As the voice commands still follow guideline 6, the voice commands were deemed to be long enough to still ensure high recognition quality.

4.5 Test Cases

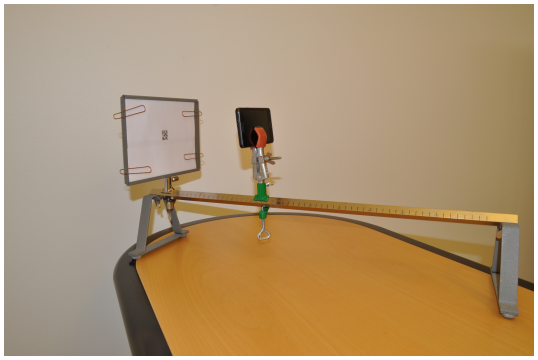
The tests were carried out using an optical bench to guarantee scientific accuracy, with a screen holder at the zero point, where the QR code was positioned. The device being tested, Google Glass or a smartphone, was then positioned at the specified mark on the optical bench, using a clamp, and pointed towards the QR code. See Figure 4.11 for a

better understanding of the experimental setup.

As seen in Figure 4.11 Google Glass was mounted in such a way that the camera sat a bit closer to the QR code than where the clamp marked on the optical bench. In order to compensate for the slight misalignment the clamp was positioned a few centimeters back from the specified mark and as such not used to determine the distance to the QR code. Instead the camera on Google Glass was used to pinpoint the exact distance to the QR code, and the clamp was positioned in such a way that the camera on Google Glass was at the distance specified in each experiment. In other words, even though the clamp was not at the same distance to the QR code for the smartphone tests as for the Google Glass tests, the camera of each device was.



(a) The experimental setup for Google Glass.



(b) The experimental setup for the Samsung Galaxy SII.

Figure 4.11: The experimental setup.

Although not shown i Figure 4.11, each device was connected to a computer via a USB cable. The result time of each test was obtained from the log within Android Studio after each run, since when running an android application via Android Studio, log information may be obtained through the log within Android Studio.

In order to measure the time needed for the results of each test a specific class was built, called `Timer` (seen in Listing 4.5). The `Timer` class was built using the singleton design pattern. A singleton class is a class that can only be instanced once during the entire

execution of an application. However, the instance of a singleton class lives throughout the entire execution and may be accessed from anywhere in the application [89].

Using the singleton pattern meant that the timer could be started in one class, and stopped in another, without having to pass the instance around, potentially affecting the performance of each device.

Listing 4.5: The Timer class

```
1
2  public class Timer {
3     private static Timer ourInstance = new Timer();
4     public static Timer getInstance() { return ourInstance; }
5     private Timer() { }
6
7     private boolean timerRunning = false;
8     private Long startTime;
9     private Long stopTime;
10
11    public void startTimer() {
12       if(timerRunning) { Log.d("TIMER", "Timer already running"); }
13       else  { startIme = System.nanoTime(); }
14    }
15
16    public void stopTimer() {
17       if(!timerRunning) { Log.d("TIMER", "No timer running"); }
18       else { stopTime = System.nanoTime()); }
19    }
20
21    private long getElapsedTime(int timerID) { return stopTime - startTime; }
22
23    public void logElapsedTime(String information) {
24       Log.d("TIMER", information + ": " + String.valueOf(getElapsedTime() + "
             nano seconds");
25    }
26 }
```

4.5.1 Text Length

When evaluating the text length, the text string used was not a predefined string, but rather a randomised string. The text was randomly generated using the distribution of characters in regular English text. One might argue that technical texts have a slightly different distribution of characters, but Google recommends developers to be personal when writing text meant to be displayed to the user [49]. The text was also short enough to fit a smartphone screen and as such the difference using slightly different distribution of characters would not have any major effect on the results.

Listing 4.6 shows how each character was randomly selected. `randchar` was called from within a for loop, where the character was added to a string. The number of loops determined how long the text was going to be. The `doubleList` list contained the distribution of each character according the their distribution in the english language [1], and the `alph` list contained all the individual characters, including whitespace. As a decimal number between zero and one was randomly selected, a corresponding character was picked out based on the distribution of that character.

A recursive method located the corresponding character and returned the character back up to the calling method `randchar`, which in turns returned the character to the for loop in which a string of random characters were collected.

Listing 4.6: The randomizer class

```
1  public RandomizeEnglishText() {
2     doubleList = new ArrayList<>();
3     doubleList.add(0.0651738);                          // A
4     doubleList.add(doubleList.get(0) + 0.0124248);  // B
5     doubleList.add(doubleList.get(1) + 0.0217339);  // C
6        [...]
7     doubleList.add(doubleList.get(23) + 0.0145984); // Y
8     doubleList.add(doubleList.get(24) + 0.0007836); // Z
9     doubleList.add(doubleList.get(25) + 0.1918182); // _
10    alph = Arrays.asList("a", "b", "c", "d", "e", "f", "g", "h", "i", "j", "k",
          "k", "m", "n", "o", "p", "q", "r", "s", "t", "u", "v", "w", "x", "y",
          "z", " ");
11 }
12 private double randfrom(double min, double max) {
13    Random rand = new Random();
14    double range = (max - min);
15    return min + range * rand.nextDouble();
16 }
17 private String getChar(int pos, double rand) {
18    return (rand <= doubleList.get(pos) || pos+1 <= alph.size()) ?
19       alph.get(pos) :
20       getChar(pos+1, rand);
21 }
22 public String randchar() {
23    double rand = randfrom(0, 1);
24    return getChar(0, rand);
25 }
```

4.5.2 Distance to the QR Code

Using the `Timer` class, seen in Listing 4.5, the elapsed time from the start of the application until the QR code was decoded was measured for each device. The test was performed 30 times for each device, with 3 different distances between the QR code and the device. Figure 3.1 described the application functionality and this test measures steps 1 and 2 in Figure 3.1.

Although the `Timer` class will give the elapsed time in nanoseconds the result will be presented in seconds in order to clearly show the significance of each time as small, nano seconds, differences between devices will not matter as much as seconds.

4.5.3 Complexity of the QR Code

Using the `Timer` class, seen in Listing 4.5, the elapsed time from the start of the application until the QR code was decoded was measured for each device. The test was performed 30 times for each device, with 3 different complexities of the QR code. Figure 3.1 described the application functionality and this test measures steps 1 and 2 in Figure 3.1.

Although the `Timer` class will give the elapsed time in nano seconds the result will be presented in seconds in order to clearly show the significance of each time as small, nanoseconds, differences between devices will not matter as much as seconds. Figure 4.12 shows the three different QR codes used for the complexity test.

(a) 1 encoded character.

(b) 50 encoded characters.

(c) 100 encoded characters.

Figure 4.12: The three different QR codes used in the complexity test.

4.5.4 Display Time

Using the `Timer` class, seen in Listing 4.5, the elapsed time from the point that the product information had ben downloaded until the information was sent to the display was measured for each device. The test was performed 30 times for each device, with 3 different sizes of product information. Figure 3.1 described the application functionality and this test measures steps 4 and 5 in Figure 3.1.

Although the `Timer` class will give the elapsed time in nanoseconds the result will be presented in seconds in order to clearly show the significance of each time as small, nano seconds, differences between devices will not matter as much as seconds.

4.6 Summary

The application works in such a way that the first screen the user sees when launching the application, on both Google Glass and smartphones, is the camera screen. The user is then to position the device in such a way that the QR code may be scanned by the device's camera. The QR code contains a product ID for a specific product.

The user does not need to press any shutter button in order to scan the QR code. Instead the application will automatically recognise any QR code pattern that appears in the camera view, as well as scan the QR code. The reason for not implementing a start menu or any similar start screen, to show the user before the camera screen is displayed, is to, according to Google's design guidelines [45, 54], maintain the focus on what the application is intended to do and to keep the application simple and easy to use.

Next the application will decode the QR code. The decoding process is done by the ZXing library, which is an open source barcode image processing library [91]. The QR code contains a product ID which is then used in the downloading process. The downloading process entails connecting to a database containing information about different products, and, by using the decoded product ID, retrieving the information on the specific product.

The downloaded information contains the product name, as well as a list of components and the instructions necessary to assemble the product. All the information is then sorted in to respective classes and the information may be displayed to the user. When the product information is being downloaded a loading animation is displayed on screen. On Google Glass the loading information is a loading bar at the bottom of the screen, and in the smartphone application the loading animation is a spinning wheel.

When the download process has finished, the information is displayed to the user in the form of a slide view. The first slide that is displayed to the user is the title slide. The title slide contains the name of the product as well as an image (if an image existed in the database). Following the title slide are the component slides. Each component has their own slide due to the fact that a component may be described in both text and an image.

After all the component slides follow the instruction slides. Similar to the components an instruction may be presented by text only, or by both text and an image. In contrast to the components, however, instructions may also be presented with an image and no text.

As Google provides developers of Google Glass applications with predefined layouts [48], these layouts were used in the Google Glass application. The layouts used were “Title”, “Columns”, “Text” and “Caption”. The predefined layouts were also used as basis for the slide layouts in the smartphone application.

The Title layout was used for the title card. The Columns layout was used for the slides with both text and an image. The Text layout was used for the slides with text only. The Caption layout was used for the slides containing only an image. All layouts, except for the Title layout, also contain text at the bottom of the screen called “footer” and “timestamp”. The footer contains information on whether the current slide is a component slide or an instruction slide. The timestamp contains information on which slide is currently being viewed.

While browsing through the slides in the Google Glass application, the user may also navigate using voice commands. The voice commands available in the Google Glass appli-

cation are “Show next slide”, “Show previous slide”, “Show components”, “Show instructions” and “Scan again”.

Most of the voice commands follow 11 out of the 15 voice command guidelines provided by Google [61]. For example, “Show components” does not follow the guideline which states: “Is general enough to apply to multiple Glassware, but still has a clear purpose”. “Show components” is a specific voice command and could potentially apply to multiple Google Glass applications, but not all.

While viewing the slides “ok glass” is shown at the bottom of the screen. “ok glass” indicates that voice commands are available and saying “ok glass” at that point brings up the voice command menu, showing all available voice commands. However, “ok glass” is also shown in combination with a dark, transparent, overlay, which ensures “ok glass” is always visible no matter what image is shown on screen. However, the dark overlay also means that any image shown is darkened by the overlay.

In terms of the testing performed on the application, the experimental setup consisted of an optical bench on which the QR code was positioned at the zero mark. Each specific device was positioned so that the camera of each device was positioned according to the specifications of each test. Each test result was then obtained through a laptop with which each device was connected via a USB cable. The test results were printed out from a timer class, called `Timer`, which was implemented using the singleton design pattern, meaning that the class had only one global instance which could be accessed from anywhere within the application [89].

The test which did not require the experimental setup was the text length test. Instead the text length test was performed using a class which randomised English characters, which were then concatenated together to form a longer string.

Three other tests were performed using the experimental setup. In the first of these three tests the distance to the QR code was varied. In the second test the complexity of the QR code was varied. In the third and final test the size of the downloaded information

was varied. Each of these tests were performed 30 times, for each device and each different specification, to ensure statistical significance [71].

5 Results

Overall the results show that Google Glass was slower than the smartphones in all stages of the application. Google Glass performed about half a second slower than the smartphones, where the Samsung Galaxy SIII proved to be faster than the Samsung Galaxy SII. Using different complexities of the QR code Google Glass was only able to read the QR code encoding one character. The Samsung Galaxy SII managed to read the QR code encoding 50 characters, however not the QR code encoding 100 characters. The Samsung Galaxy SIII managed to read all three of the QR codes which represented three increasing levels of complexity, encoding 1, 50 and 100 characters.

Google Glass was also only able show somewhere between 200 and 220 characters on screen at the same time, when using a layout design where only text was presented. Using a layout design which displays both text and an image the number of characters Google Glass may show on screen was somewhere between 100 and 120.

Since the layout design for the Google Glass application was used as basis for the smartphone application many of the screens in the smartphone application are relatively empty, and could potentially contain much more information.

5.1 Demonstration Case

The result of the implementation is described in the following demonstration case, where the goal is to build a Lego figure, called "Space Pirate", seen in Figure 5.1. The following walkthrough of the demonstration will cover both the Google Glass version as well as the smartphone version. The Lego parts will already be partially assembled in order to reduce the number of slides, as the general idea of the application will still be presented. There are four different components used to assemble the Space Pirate. All of the components can be seen in Figure 5.2.

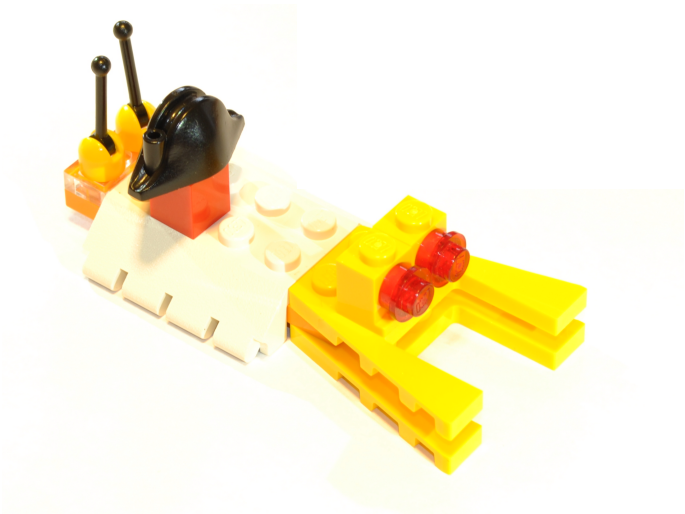
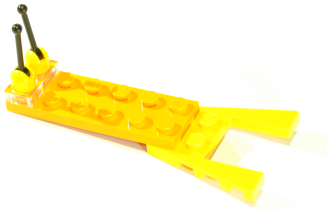
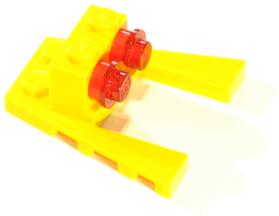


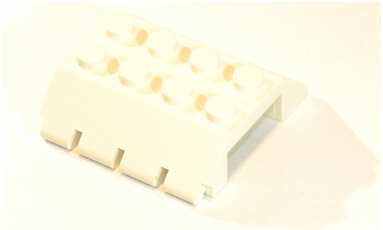
Figure 5.1: Space Pirate.



(a) Steering.



(b) Tail Light.



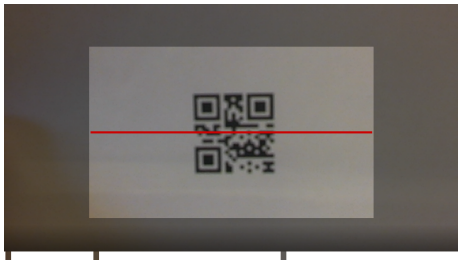
(c) Body.

(d) Pirate.

Figure 5.2: Components.

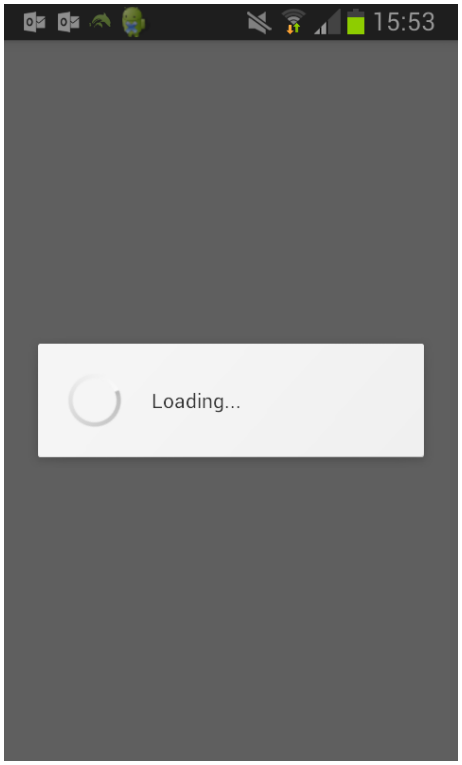
In order to receive any information on the product the user must scan the product specific QR code, as seen in Figure 5.3 (a) and in Figure 5.3 (c). When the QR code has been scanned the information will be downloaded, as seen in Figure 5.3 (b) and in Figure 5.3 (d).

(a) Scanning a QR code in the Google Glass application.



(b) Loading in the Google Glass application.

(c) Scanning a QR code in the smartphone application.

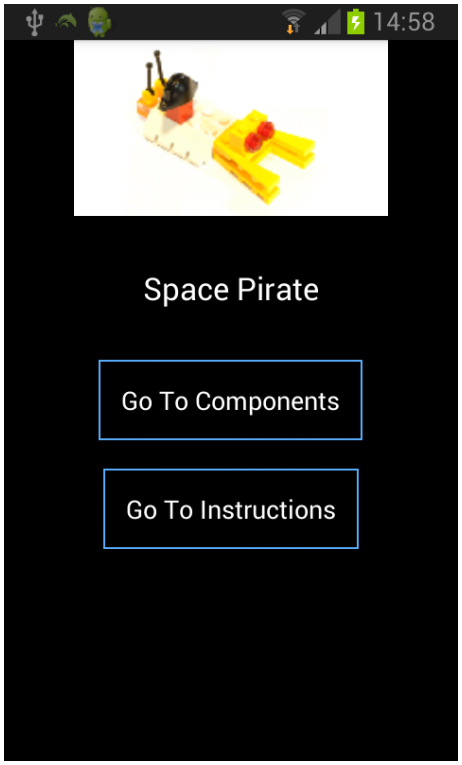


(d) Loading in the smartphone application.

Figure 5.3: Scanning and loading screens.

The first slide the user sees, when the product information has been downloaded and is displayed to the user, is the title slide. The title slide for the Space Pirate can be seen Figure 5.4 and shows the name of the product, as well as an image of what the product will look like when assembled. In Figure 5.4 (a), the Google Glass application can be seen, and in Figure 5.4 (b) the smartphone application can be seen.

(a) The title card in the Google Glass application.



(b) The title slide in the smartphone application.

Figure 5.4: The title slide.

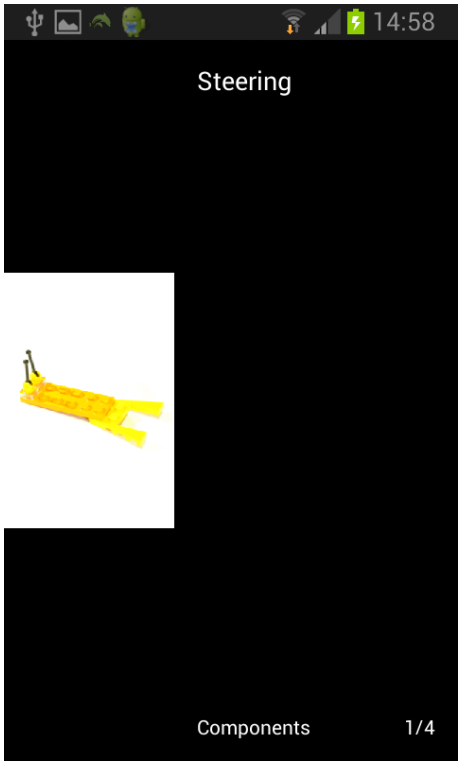
Using the Google Glass application users can, by saying “ok glass”, bring up the voice command menu, seen in Figure 5.5, and navigate through the application. In the smartphone application, no voice commands exist, but users may use the two buttons on the title slide in order to skip through the slides to either the components or the instructions. However, in both the Google Glass application and the smartphone application users can

simply swipe, using a finger, to the next slide in line.

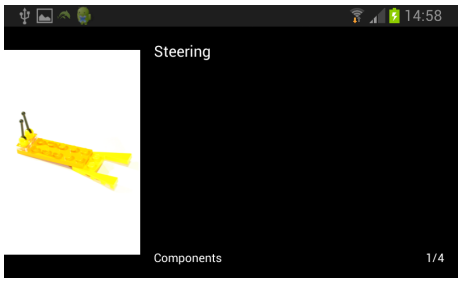
Figure 5.5: The voice command menu.

The second slide of the application is the first component slide, seen in Figure 5.6. Here the user sees the first component needed for assembling the Space Pirate, which is the Steering component, seen in Figure 5.6 (d). While the Google Glass application is fixed in its layout, the smartphone application can be altered in terms of the screen orientation. This is because while Google Glass is mounted on the user's head and may not be rotated, the smartphone can. As such the smartphone application can be viewed in either portrait orientation mode, seen in Figure 5.6 (b), or in landscape orientation mode, seen in Figure 5.6 (c). In both cases the image allocates the same percentage of screen space in width as the image does in the Google Glass application, which is $\frac{3}{8}$ of the screen. For the remainder of this demonstration walkthrough the screen shots of the smartphone application will be in landscape mode, but it should be noted that all slides in the smartphone application may be viewed in both portrait orientation mode and landscape orientation mode.

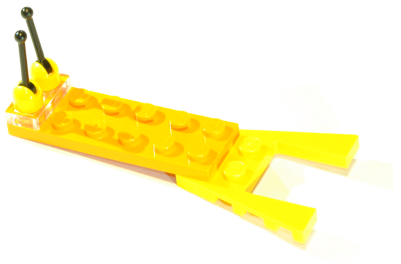
(a) The first component card.



(b) The first component slide in portrait screen orientation.



(c) The first component slide in landscape screen orientation.

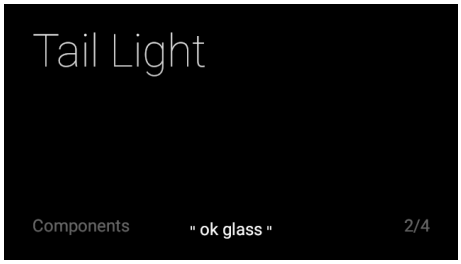


(d) The Steering component.

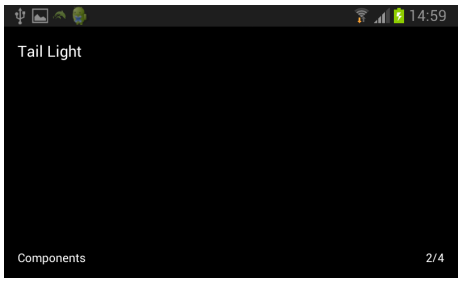
Figure 5.6: The first component.

The next slide is the second component slide, seen in Figure 5.7, which uses only text to describe the component. The component is called "Tail Light", and can be seen in Figure 5.7 (c). The reason for not using an image to describe the Tail Light component

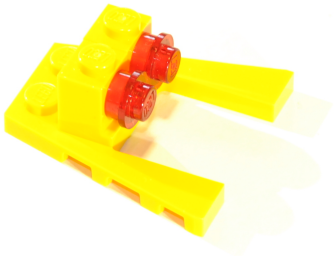
is because the text was deemed sufficient enough for the user to distinguish the Tail Light component from the others.



(a) The second component card.



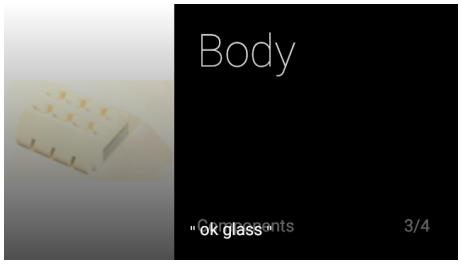
(b) The second component slide.



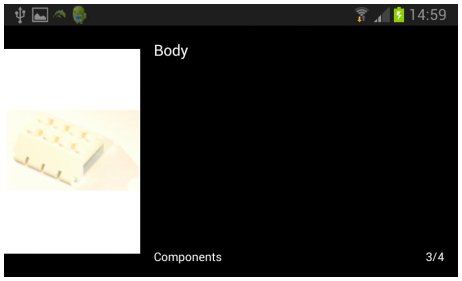
(c) The Tail Light component.

Figure 5.7: The second component.

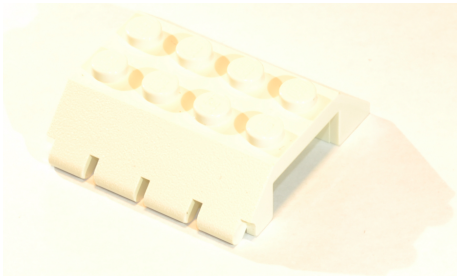
The fourth slide in the application is the third component slide, seen in Figure 5.8. Similar to the first component slide, the third component slide uses both and image as well as text in order to describe which component the user need. The component is called "Body" and can be seen in Figure 5.8 (c). The reason for using both text and an image to describe the Body component is because the component name is a general one and could potentially apply to the Steering component as well.



(a) The third component card.



(b) The third component slide.

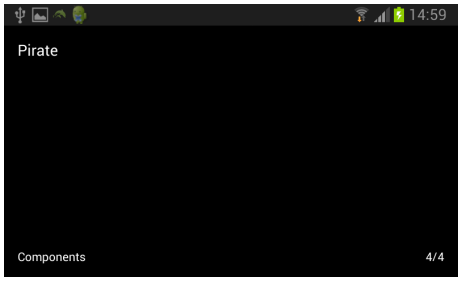


(c) The Body component.

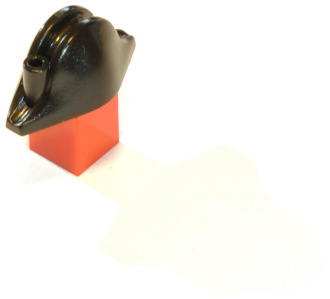
Figure 5.8: The third component.

The next slide is the fourth and last component slide, seen in Figure 5.9. Similar to Figure 5.7, the fourth component is also only described in text. The component is called "Pirate" and can be seen in Figure 5.9 (c).

(a) The fourth component card.



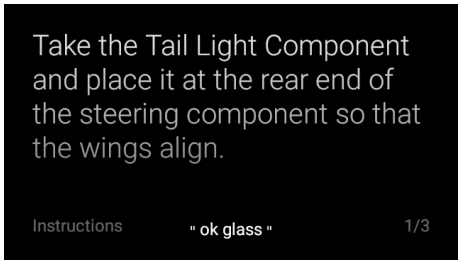
(b) The fourth component slide.



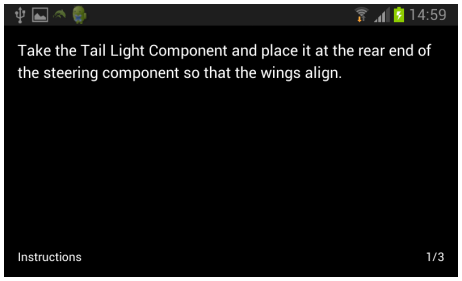
(c) The Pirate component.

Figure 5.9: The fourth component.

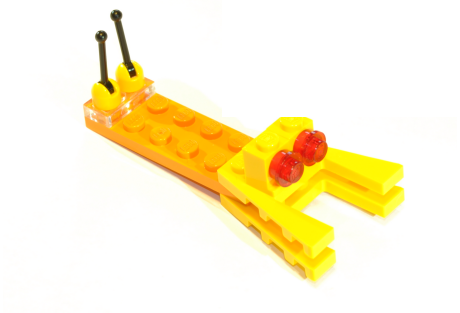
After the components come the instructions. At this point the user should have gathered all of the components and be ready to assemble the product. The first instruction slide can be seen in Figure 5.10. The goal is to combine the Tail Light component and the Steering component in to what can be seen in Figure 5.10 (c).



(a) The first instruction card.



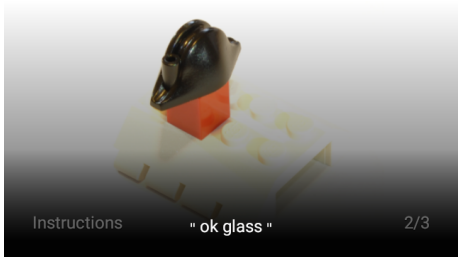
(b) The first instruction slide.



(c) The goal of the first instruction.

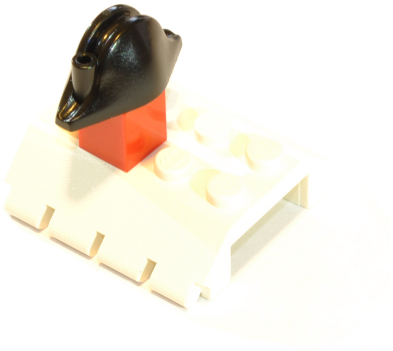
Figure 5.10: The first instruction.

The second instruction is described with only an image and no text, as seen in Figure 5.11. The reason for this is because the alignment of the components would be difficult to describe in text and a larger image makes it easier for the user to see how the components should fit together. Using both text and an image would mean that the image would be scaled down. The user is, with the instruction, supposed to combine the Body component and the Pirate component in to what can be seen in Figure 5.11 (c).



(a) The second instruction card.

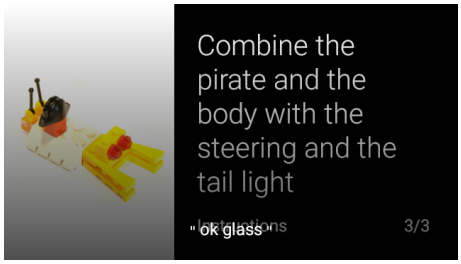
(b) The second instruction slide.



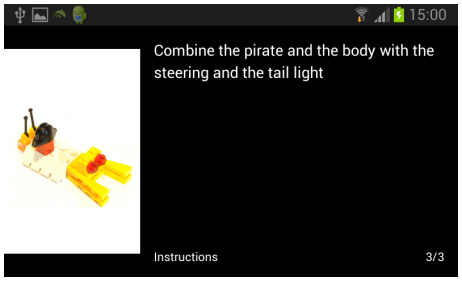
(c) The goal of the second instruction.

Figure 5.11: The second instruction.

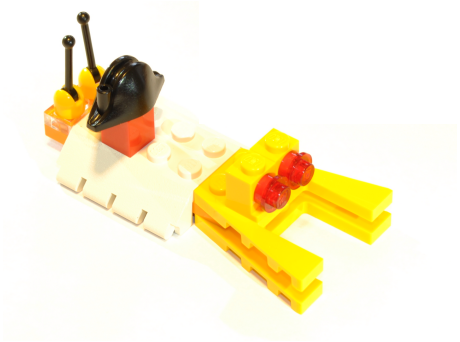
The final slide, seen in Figure 5.12, describes the final instruction in both text and an image. The reason for using both text and an image is because the text may contain enough information on how to combine the components and the image can be used as visual guideline. Potentially the image could be enough information on its own, and could as such be shown in full scale instead. However, since the image does not need to be full scale it will take up less data space and as such be faster to download compared to a full scale image.



(a) The third instruction card.



(b) The third instruction slide.



(c) The goal of the third instruction.

Figure 5.12: The third instruction.

When the user has completed what the third and last instructions say, the product is fully assembled. As such, the application may now be closed or another QR code may be scanned. In the Google Glass application the user can say “ok glass” and then “Scan again”, in order to bring up the camera and scan another QR code. In the smartphone application the user may press the default back button on the smartphone in order to bring up the camera screen and scan another QR code.

5.2 Text Length

The following are the results from evaluating the number of characters that may fit on a card in the Google Glass application, both when using only text and when using both text and an image. The test was performed only once for each device since, despite randomising

the characters used, the text length would not differ significally from one case to another. As seen in the following results, none of the tests resulted in an additional card with only one character. Had that been the case, different combinations of characters may have changed the number of cards necessary to display all of the information. As that was not the case only one test was deemed sufficient enough to determine the number of cards required to display the same information as displayed on a smartphone device.

However, some trial and error was part of the process of finding how many characters would fill the smartphone screens. 550 was deemed a suitable number for the Samsung Galaxy SII och 750 was deemed a suitable number for the Samsung Galaxy SIII, as that many characters filled up the screen and yet still left some room for longer characters or longer words, as seen in Figure 5.13.

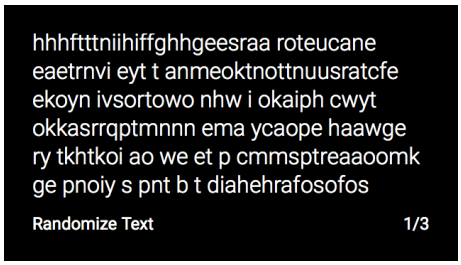
(a) The Samsung Galaxy SII screen.

(b) The Samsung Galaxy SIII screen.

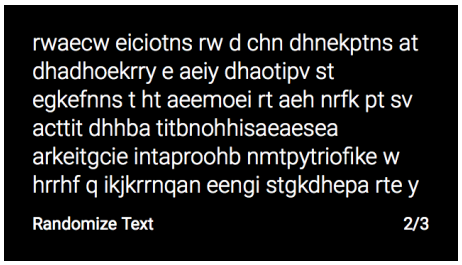
Figure 5.13: The maximum text length on the smartphone application.

For the Samsung Galaxy SII, 600 characters did not fit on the screen, and for the Samsung Galaxy SIII, 800 characters did not fit on the screen. As such 550 and 750 respectively was set as the maximum number of characters, although depending on the characters being displayed the text may use up more or less space. For example, an 'i' does not take up as much space as a 'w'. The filled smartphone screens can be seen in Figure 5.13.

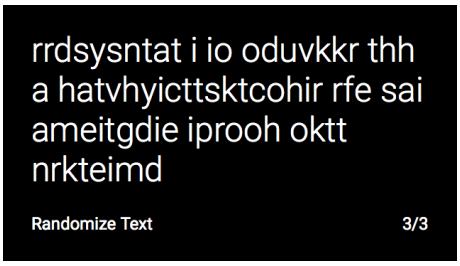
When translating the results from the smartphones to Google Glass the same text was used and simply copied over to the Google Glass application. The result of the text that filled the Samsung Galaxy SIII screen, seen in Figure 5.13 (a), can be seen in Figure 5.14.



(a) The first slide.



(b) The second slide.



(c) The third slide.

Figure 5.14: The Samsung Galaxy SII text.

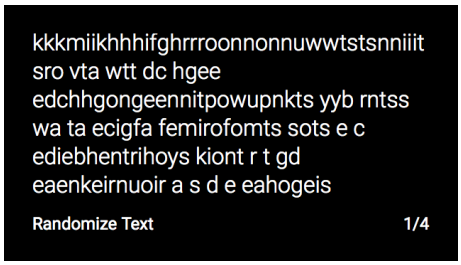
As seen, the text that filled the Samsung Galaxy SII screen took up roughly two and a half cards in the Google Glass application. Although the screen is nearly filled in Figure 5.14 (c), the text is dynamically sized depending on the amount of text being displayed. As such the card can be deemed about half full. Table 5.1 shows the exact number of char-

acters and words on each card, showing that the last card contains about half the number of characters as the other two cards.

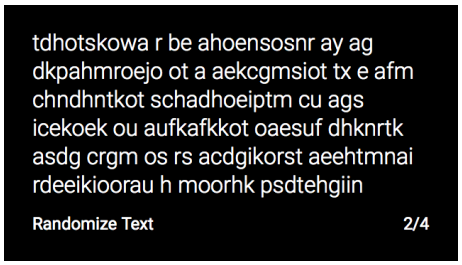
Table 5.1: Details on text length on Google Glass from Samsung Galaxy SII.

Figure	**Number of words**	**Number of characters**
Figure 5.14 (a)	30	229
Figure 5.14 (b)	35	229
Figure 5.14 (c)	13	92
Figure 5.14 (Total)	78	550

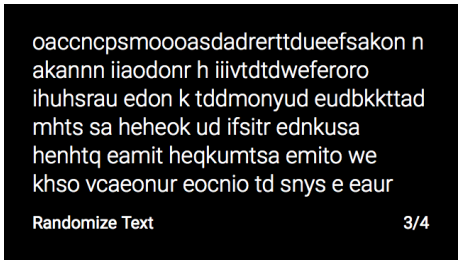
The result of using the text that filled the Samsung Galaxy SIII screen, seen in Figure 5.13 (b), in the Google Glass application can be seen in Figure 5.15. As seen the Samsung Galaxy SIII screen fit more characters than the Samsung Galaxy SII screen and as such three and a half cards in the Google Glass application was needed to display all the text that filled the Samsung Galaxy SIII screen. The exact number of characters and words on each card can be seen in Table 5.2.



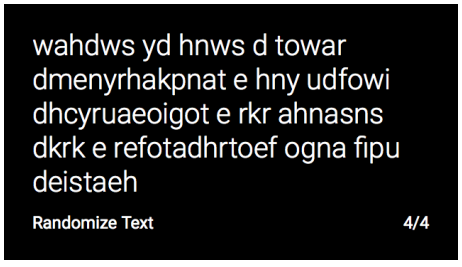
(a) The first slide.



(b) The second slide.



(c) The third slide.



(d) The fourth slide.

Figure 5.15: The Samsung Galaxy SIII text.

Table 5.2: Details on text length on Google Glass from Samsung Galaxy SIII.

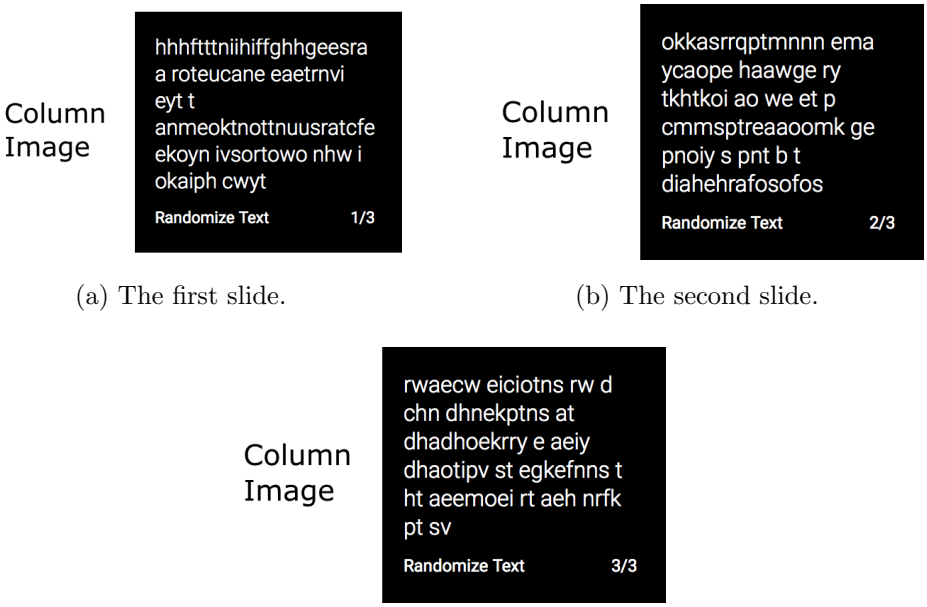
Figure	**Number of words**	**Number of characters**
Figure 5.15 (a)	26	199
Figure 5.15 (b)	32	213
Figure 5.15 (c)	29	217
Figure 5.15 (d)	19	121
Figure 5.15 (Total)	106	750

Instructions described in text may also be displayed in combination with an image. When text combined with an image is used as the design layout for a slide, the maximum number of characters that can be displayed changes. The image, according to Google's predefined card layout design [49], makes up $\frac{3}{8}$ of the card, and as such the text makes up the other $\frac{5}{8}$. As such the maximum number of characters can be calculated using the maximum number of characters used when presenting only text.

For the Samsung Galaxy SII, the maximum number of characters when using both text and an image was calculated as follows:

$$\frac{5}{8} * 550 \;=\; 343.75$$

. As three quarters of a character is of no use the maximum number of characters on the Samsung Galaxy SII screen is 343 characters. The result of using the same text as when displaying only text on the Samsung Galaxy SII, only shorten down to 343 characters, can be seen in Figure 5.16, with specific numbers in Table 5.3.



(a) The first slide.

(b) The second slide.

(c) The third slide.

Figure 5.16: The Samsung Galaxy SII text and image.

Table 5.3: Details on text length on Google Glass from Samsung Galaxy SII.

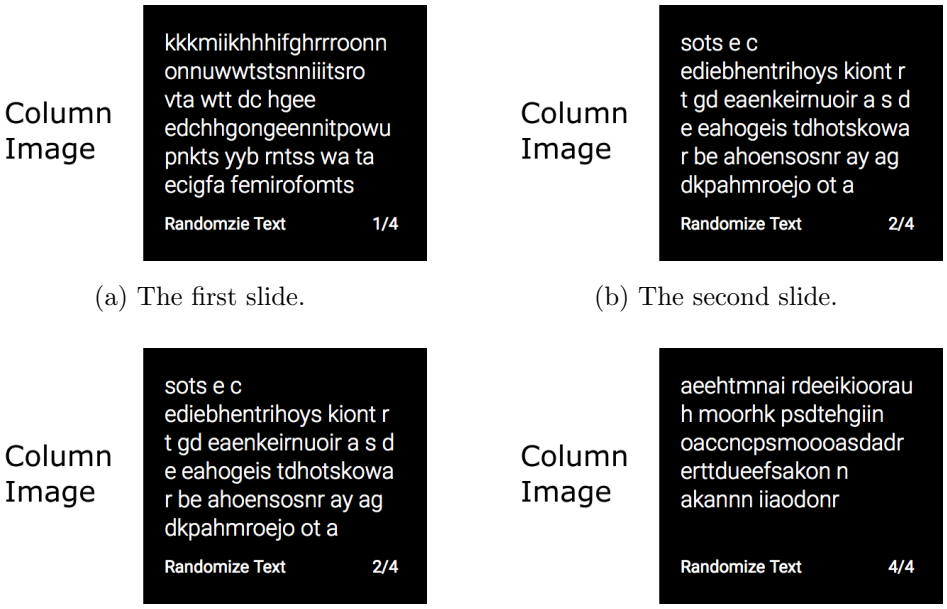
Figure	**Number of words**	**Number of characters**
Figure 5.16 (a)	12	118
Figure 5.16 (b)	18	111
Figure 5.16 (c)	21	114
Figure 5.16 (Total)	51	343

The maximum number of characters, when using both text and and image on the Samsung Galaxy SIII, was calculated the same way as for the Samsung Galaxy SII, as follows:

$$\frac{5}{8} * 750 \;=\; 468.75$$

. As three quarters of a character is of no use the maximum number of characters when displaying both text and an image is as such 468 characters. The result of using the same text as in Figure 5.13, only shorten down to 468 characters, in the Google Glass application,

can be seen in Figure 5.17. The application requires about three and a half card to be able to display all of the text. The exact number of characters and words on each card can be fount in Table 5.4.



(a) The first slide.

(b) The second slide.

(c) The third slide.

(d) The fourth slide.

Figure 5.17: The Samsung Galaxy SIII text and image.

Table 5.4: Details on text length on Google Glass from Samsung Galaxy SIII.

Figure	**Number of words**	**Number of characters**
Figure 5.17 (a)	12	127
Figure 5.17 (b)	23	124
Figure 5.17 (c)	18	115
Figure 5.17 (d)	9	102
Figure 5.17 (Total)	62	468

Overall, a slide in the Google Glass application, which contains only text, can fit around 200 characters, which corresponds to somewhere between 25 to 30 words. When using both text and and image the number of characters that may fit on a slide in the Google Glass application is about 115, which corresponds to about 15 words. In total the Google Glass

application requires 3 to 4 cards for every slide in the smartphone application, depending on the size of the smartphone screen, in order to display the same text as a filled slide in the smartphone application.

5.3 Distance to the QR Code

In Table 5.5, the average time for registering and decoding a QR code while varying the distance to the QR code can be seen. The size of the QR code was optimised for a distance of two decimeters between the QR code and the device, which also shows in the results as all devices registered the QR code fastest at a distance of two decimeters. The QR code used encoded only one character, an 'a'. The results from every individual run can be seen in Appendix B

Table 5.5: Average time of registering a QR code with varying distance.

Distance (decimeter(s))	**Google Glass (sec)**	**Samsung Galaxy SII (sec)**	**Samsung Galaxy SIII (sec)**
1.0	1.919976807	1.965959850	1.839656134
2.0	1.831889852	1.649790392	1.487449920
3.0	2.227158610	1.921767210	1.568837591

5.4 Complexity of the QR Code

Table 5.6 shows the average time for each device to both register and decode a QR code of varying complexity. The distance to the QR codes was two decimeters, as the size of the QR code was optimised for a distance of two decimeters between the QR code and the device. As seen in Table 5.6 some of the results are infinite, meaning that the device was never able to register the QR code due to the high complexity of the QR code. The Samsung Galaxy SIII was able to both register and decode all three of the QR codes, The

Samsung Galaxy SII was not able to decode the most complex QR code while Google Glass was only able to register the simplest QR code.

One explanation as to why Google Glass was not able to register any QR code other than the simplest one would be that the Google Glass camera did not auto focus as well as both the Samsung Galaxy SII and the Samsung Galaxy SIII. As such, the complexity of the QR code made it difficult for Google Glass to register the QR code. The difference can be seen in Figure 4.1. Every single test run for each device can be found in Appendix B.

Table 5.6: Average time of registering a QR code with varying density.

Encoded Characters	**Google Glass (sec)**	**Samsung Galaxy SII (sec)**	**Samsung Galaxy SIII (sec)**
1	1.831889852	1.649790392	1.487449920
50	*Infinite*	2.096434673	1.913361687
100	*Infinite*	*Infinite*	1.949743816

5.5 Display Time

The average display time of each device, with varying information sizes, can be seen in Table 5.7. As the size of the information increased, so did the average display time. The increase in time came as a result of all information being downloaded at the same time, which is then handled by the device. Although only the currently visible information is sent to the display, there is still more information handled by the device and as such the device must search through more information in order to determine which information is to be displayed.

The display time is also affected by other background processes running at the same time. The average display time also only shows the time from when the information was sorted into classes until the information was sent to the display, and as such does not take in to account the potential delay the display might add in order to actually display the

information to the user. Every single test run for each device can be found in Appendix B.

Table 5.7: Average display time for Google Glass with varying information size.

Information Size (Bytes)	**Google Glass (sec)**	**Samsung Galaxy SII (sec)**	**Samsung Galaxy SIII (sec)**
1 000	0.310802205	0.008374156	0.025498950
100 000	0.442440796	0.020444892	0.034109654
1 000 000	0.582170613	0.050938751	0.078170083

5.6 Summary

While the Google Glass application and the smartphone application essentially have the same functionality, there are some differences between the two. The Google Glass application may be controlled using voice commands, which is a feature the smartphone application does not have. Otherwise both of the applications follow Google's design guidelines [45, 54] in terms of being easy to use and both the Google Glass application and the smartphone application focus on what is important in the application, which is to scan a QR code and view product information.

Examining the amount of text that may fit the screen on the different devices, it was shown that a full screen of text on the Samsung Galaxy SII covered about two and a half screens on Google Glass. When filling up an entire screen on the Samsung Galaxy SIII, the same amount of text filled up about three and a half screens on Google Glass. In other words, the amount of information that may be displayed on Google Glass is relatively limited compared to smartphone devices.

In terms of the other test results it can be seen that the Google Glass application is almost always slower that the smartphone application, on both the Samsung Galaxy SII and the Samsung Galaxy SIII. Google Glass was only about 0.4 seconds behind the Samsung Galaxy SII when scanning the QR code from different distances.

However, when the complexity of the QR code was varied, Google Glass proved to be less effective. While the Samsung Galaxy SII was not able to scan and decode the QR code that encoded 100 characters, Google Glass was not even able to scan and decode the QR code that encoded 50 characters. The Samsung Galaxy SIII managed to scan and decode all the different QR codes, although the more complex QR codes took about half a second longer to scan and decode.

The display time test was the test where Google Glass proved to be least effective, in comparison to the other tests. Both the Samsung Galaxy SII and the Samsung Galaxy SIII manage to send information to the display in under 0.1 seconds, even though the information size was 1 megabyte. Google Glass, however, always took longer than 0.1 seconds, even when the information size was only 1 kilobyte.

6 Conclusions

As seen in the test results, the Google Glass application was almost always the slowest in all of the tests. However, as the test results only show the time from when the QR code has been scanned until when the information is being displayed on screen, taking into account the fact that the difference in time was about half a second, the difference in time might not be significant.

After the user has scanned a QR code and the information is displayed, that is when the actual usage of the application starts. A user might be able to look past the fact that the Google Glass application is slower at scanning and presenting information as the advantages of Google Glass lies not in the speed of the device but rather in the user interface.

The fact that users do not have use their hands is the major advantage of using Google Glass over a smartphone. Although voice commands are possible to implement in the smartphone application as well, the user must still hold the smartphone, or place the smartphone on a surface, in order to see the information displayed. Google Glass puts the display in front of the user and does not require the user to hold the display in any way.

However, the fact that Google Glass was not able to scan more complex QR codes is definitely a negative aspect. Although the product ID:s used in this test were not long enough as to where complexity would become an issue, the complexity of the QR code will be an issue when the database contains many, many more products and the product ID:s are such that the complexity of the QR code is too high for Google Glass to handle.

As such, one clear improvement Google must make to Google Glass is to upgrade the camera. Not only to be able to compete with smartphone cameras, which, as seen, were much better already at the time of Google Glass' release, but simply to be able to identify QR codes with higher complexity and at longer distances.

Another important aspect of Google Glass is the amount of information the device can display on one screen. As shown in the text length test, a card on Google Glass which

only displays text can hold somewhere between 200 and 220 characters, in comparison to smartphones with similar layout design, which can hold somewhere between 550 and 750 characters.

According to Google's guidelines for writing text for Google Glass, the text should be brief and simple, with the most important part of the text first and without repetition. Clearly, with the limitation on text length, writing short and precise text is important on Google Glass. Displaying information to the user without having to divide text up on several cards is important, for instance when writing instructions. Requiring the user to scroll back and forth between cards in order to read one instruction would not be good design, even though scrolling back and forth can be performed via voice commands. As such, instructions should be simple and brief, similar to the way they are in the application described in this dissertation.

6.1 Personal User Experience

Having used Google Glass about every weekday for nearly four months there are a few comments that can ben made, and a few conclusions that can be drawn, simply from personal user experience. Please note that these comments are not based on any scientific studies, but are rather the opinions of the author of this dissertation.

First of all it should be said that Google Glass is very easy to use. The core features of the device are easy to grasp, such as how to navigate using voice commands, and how to use some of the built-in applications, such as taking a picture or finding the shortest route to a specific location.

However, getting used to wearing Google Glass might take some time, somewhere between a few days and a week of constant usage. Having a display slightly above the line of sight did become a bit irritating at times, especially after a weekend or a few days of not wearing Google Glass all day. However, the Google Glass display does not stay on all the time. Instead the display times out after a short period when Google Glass is not being

used.

Since Google Glass projects an image on to glass, which is then perceived by the user as the display, Google Glass is transparent when not active. As such, Google Glass, although noticeable, can be ignored when not active after a few days of usage. As stated previously, simply wearing Google Glass will take a few days to get used to.

The fact that Google Glass uses a BCT (bone conduction transducer) instead of regular headphones is appreciated as Google Glass is meant to be worn at all times, and as such users might not want to plug their ears. Doing so would potentially mean users would not be able to hear surrounding sounds, and would have to take off Google Glass when, for instance, talking to another person.

However, Google Glass does have some serious issues. One major issue is the fact that Google Glass overheats very easily. Simply interacting with an application a little too quickly will cause Google Glass to overheat. When overheated, the Google Glass touchpad, behind which the CPU sits, gets really hot. The heat can be felt by the user as the nearside of the Google Glass touchpad lies against the temple. The heat does occasionally become very distracting, and could potentially even be uncomfortable depending on the user.

When overheating, Google Glass will also not run as smoothly and a message will even be displayed on the home screen, which states "Google Glass must cool down to run smoothly".

Another downside of Google Glass is the restrictions Google has placed on the development side. The fact that voice commands must be approved does make sense in terms of the voice commands which start an application. Users want to be able to distinguish one application from another. However, when running an application, developers should be able to use the voice commands best suited for the application.

The "ok glass" overlay is also not a very good solution since part of the screen becomes more or less unusable when the dark overlay covers such a big portion of the screen. Developers should be able to better customise the design of applications, and not be restricted to

the rules and guidelines Google has decided. The fact that Google wants to make sure that Google Glass applications follow a certain standard might seem like a good idea. However, these restrictions limit the developers' creativity.

All in all, Google Glass feels like a device meant for more casual use at this time. Google Glass is not stable enough to handle more industry focused work, where the workload on the device would be very high. Used as a device which could display simple information, Google Glass seems well suited, but put under pressure, and used for longer periods of time, Google Glass falls short.

6.2 Project Evaluation

Overall, the project went well. The project goal was to create a proof-of-concept of an instruction application using Google Glass, as well as to evaluate Google Glass in terms of the user experience. These goals have been reached without any major setbacks.

More difficult aspect of the project included understanding the Google Glass example application, BarcodeEye, as the application in some ways covered more than what the proof-of-concept application was meant to cover. For instance, BarcodeEye handled ISBNs, which was not needed in this project. If redone, the BarcodeEye code could have been better refactored along the way, where unnecessary code would have been removed as soon as such code was deemed unnecessary. Doing so might have compressed the code somewhat and also made changes easier to implement.

Being able to update an image on a slide also proved difficult, as the slides were to be loaded in faster than the images could be loaded in on the slides, and as such the user either had to wait for the application or only have a default image instead of the intended one. This was first solved by passing the `ImageView`, which contained the image, around, which is normally not good praxis as you rather want to pass the object going inside the view around as argument instead.

Through code review and evaluation the issue was solved by implementing a listener on

the specific slide containing an image, such that when the image was loaded the method implemented through the listener was called, and the slide updated with the loaded image.

The project was developed through sprints, lasting two weeks each, with a sprint demonstration at the end of each sprint. The sprint demonstration consisted of a demonstration the work done so far to the supervisor, making sure the project was still on the right track and that the development was not falling behind schedule. The feedback from the supervisor showed that the project indeed stayed on the intended path and did not fall behind schedule.

6.3 Back-End Conclusions

The dissertation involving the back-end part of the project, written by Richard Hoorn, concludes that while Google Glass is an impressive piece of technology, Google Glass was still outperformed by smartphones since Google Glass is not able to handle larger data sizes. Data sizes above one megabyte were very difficult or even impossible for Google Glass to handle.

A possible solution would be to not let Google Glass do all the work, but rather connect Google Glass to a smartphone, from which information is then streamed. As such, Google Glass would simply act as a display, displaying the information sent from the smartphone, and would not have to handle all of the downloaded information at the same time.

Hoorn also discussed the possibilities of headaches, which Google Glass could cause and the fact that looking up at the screen could cause eye strain. These concerns have been adressed by Google, who states that Google Glass is not meant for usage of longer periods of time. Using Google Glass longer periods of time will also cause them to overheat, causing both performance issue as well as a very hot exterior. As such, Google Glass is perhaps not suited for industry use, where Google Glass would potentially be used over longer periods of time, but is rather suited and designed for the more casual user.

6.4 Future Work

Since Google has discontinued the Google Glass Explorer programme, and as such Google Glass is not available for purchase at the time of writing this dissertation, it is somewhat hard to argue for any future work to be done on the application. However, that aside, some further testing should be performed.

For instance, should the voice commands be tested, both in a silent environment, but also in a noisy environment. The test should be carried out in order to better determine if Google Glass is a viable device to be used in an industry environment, where there is potentially a lot of noise.

The application should also be tried out by actual users for an extensive period of time, and the results of such a test period evaluated, in order to determine whether some parts of the application should be redesigned or not. Following Google's design guidelines is only one aspect of the development process. In the end it is the users who should be satisfied with the result.

6.4.1 Official Approval of Voice Commands

The voice commands used in the Google Glass application have not yet been officially approved by Google, nor have they been submitted to Google. Since the discontinuation of the Google Glass Explorer programme has Google removed the voice command approval form, which developers were meant to fill in and submit to Google.

Having custom-made, contextual voice commands officially approved by Google was a requirement for Google Glass applications with custom-made, contextual voice commands to be released on MyGlass. Whether this requirement is still present when the new version of Google Glass is eventually is released remains to be seen.

Potentially customised voice commands could be designed using the voice recognition feature built in to the Android operating system. However, doing so would require a design similar to that of Google's contextual voice command system, which uses "ok glass" as the

phrase that loads up the voice command menu and listens for other voice commands. An application which always listens for all voice commands could become difficult to use in a noisy environment since Google Glass would listen for (and potentially recognise) several different phrases, instead of one specific phrase.

6.4.2 TextResultProcessor

In the Google Glass application, the `TextResultProcessor` class is not required any more and should as such be removed. At this point the `TextResultProcessor` class is only used as middleware between an instance of the `Products` class, sent as an argument from the `DownloadProductTask` class, and a list of `CardPresenter`. Instead the `CardPresenter` class should only be instanced once for each product, and maintain the instance of the Products class itself.

The reason the `TextResultProcessor` class exists in the first place is due to how the Google Glass application was originally built, where all information presented was encoded directly in the QR code. At that point the `TextResultProcessor` was used when the encoded information was a text string. At this point the only information encoded in the QR codes are product ID:s.

The smartphone application already functions in this way, where the information stored in the instance of the `Products` class is used directly when a slide is created, instead of first being sorted through a middleware class.

6.4.3 A General Fragment

The smartphone application should only have one general fragment instead of different ones for different purposes. A general fragment should be implemented in order for the smartphone application to become even more similar to the Google Glass application, which uses the `CardBuilder` class. The `CardBuilder` class acts as a general case that takes the layout as input. The smartphone application could be designed in a similar way, where a

general fragment takes the layout as an argument. At this point there is one fragment for each individual layout.

6.5 Concluding Remarks

Google Glass is an interesting attempt at developing technology further. The potential of Google Glass is exciting. However, the fact that Google Glass is intended for use in short bursts, simply to take a picture or to send a message, makes it hard to argue in favour of using Google Glass in any situation apart from more casual ones. As long as any work performed on the device is not too complex or demanding in terms of hardware, Google Glass can be very helpful.

However, Google Glass should not be seen as a competitor to smartphones, but rather a complementary device which may be used together with a smartphone. In terms of displaying instructions to a user, who is assembling components, Google Glass could be useful as the user does not have to shift focus and look away. As long as the instructions are short, simple and do not take up too much data space, Google Glass would be the preferred device over smartphones.

However, the smartphone's bigger screen and stronger hardware make smartphones the preferred choice when the instructions are complex and contain, for instance, a video, as the video's data size potentially would be too large for Google Glass to handle without performance issues.

One solution would be to compress the video, both in length and data size, or to have the video stream from a smartphone on to Google Glass, using Google Glass and a smartphone as complements to each other. As Google continues to develop Google Glass behind closed doors, it will be interesting to see how Google Glass has evolved once it is revealed again. Until such time, one can only hope Google keeps improving on Google Glass, and relaxes of some of their restrictions, as Google Glass, despite its flaws, shows great potential.

References

[1] Electronic Frontier Foundation (2004). *Statistical Distributions of English Text*. http://www.data-compression.com/english.html [2015-03-23].

[2] Ascii Table (2010). *Ascii Table*. http://www.asciitable.com.

[3] United States Census Bureau (2010). *Language Spoken At Home*. http://factfinder.census.gov/faces/tableservices/jsf/pages/productview.xhtml?pid=ACS_10_1YR_S1601&prodType=table [2015-03-06].

[4] QR Struff (2011). *What Size Should A Printed QR Code Be*. http://www.qrstuff.com/blog/2011/01/18/what-size-should-a-qr-code-be [2015-03-24].

[5] Wikipedia (2011). *QR Code Mask Patterns*. http://commons.wikimedia.org/wiki/File:QR_Code_Mask_Patterns.svg [2015-03-30].

[6] Google (2012). *Project Glass: Live Demo At Google I/O* [Online video]. http://youtu.be/D7TB8b2t3QE [2015-02-24].

[7] Google (2012). *Project Glass: One day...* [Online video]. http://youtu.be/9c6W4CCU9M4 [2015-02-16].

[8] Olsson, Isabelle (2012). *Google Glass Frames*. https://plus.google.com/110625673290805573805/posts/Nmc8LuwFw5M [2015-02-16].

[9] Parviz, Babak. Lee, Steve. Thrun, Sebastian. (2012). *Google Glass*. https://plus.google.com/u/0/wm/4/+GoogleGlass/posts/aKymsANgWBD [2015-02-16].

[10] Penny (2012). *Penny - Company*. http://www.penny.se/eng/company/company.html [2015-03-04].

[11] Ackerman, Elise (2013). *Could Google Glass Hurt Your Eyes? A Harvard Vision Scientist And Project Glass Advisor Responds*. http://www.forbes.com/sites/eliseackerman/2013/03/04/could-google-glass-hurt-your-eyes-a-harvard-vision-scientist-and-project-glass-advisor-responds/ [2015-02-23].

[12] Bonnington, Christina (2013). Smartphone screen sizes keep on growing—but not for much longer. http://www.wired.com/2013/04/why-big-smartphone-screens/ [2015-03-02].

[13] Brin, Sergey (2013). *Why Google Glass?* Long Beach, United States of America. http://www.ted.com/talks/sergey_brin_why_google_glass [2015-02-02].

[14] Casselman, Bill (2013). *How to Read QR Symbols Without Your Mobile Telephone*. http://www.ams.org/samplings/feature-column/fc-2013-02 [2015-03-30].

[15] Demo Wave (2013). *Information capacity and version of the QR Code*. http://www.qrcode.com/en/about/version.html [2015-03-02].

[16] Denso Wave (2013). *History of QR Code*. http://www.qrcode.com/en/history/ [2015-02-24].

[17] Denso Wave (2013). *Types of QR Code*. http://www.qrcode.com/en/codes/ [2015-02-25].

[18] Denso Wave (2013). *What is a QR Code?* http://www.qrcode.com/en/about/ [2015-02-25].

[19] Farber, Dan (2013). *GlassUp takes on Google Glass and Google legal*. http://www.cnet.com/news/glassup-takes-on-google-glass-and-google-legal/ [2015-03-02].

[20] Google (2013). *Google Glass How-to: Getting Started* [Online video]. http://youtu.be/4EvNxWhskf8 [2015-02-07].

[21] Kwang, Kevin (2013). *Samsung Galaxy S smartphones slaes hit 100M*. http://www.zdnet.com/article/samsung-galaxy-s-smartphones-sales-hit-100m/ [2015-03-25].

[22] Lee, Jay (2013). *I realize the with innovative products like Glass, the experience is more...* https://plus.google.com/+JayLee/posts/GwvagwVN6Hz [2015-03-25].

[23] Torborg, Scott Simpson, Star (2013). *Google Glass Teardown*. http://www.catwig.com/google-glass-teardown/ [2015-04-07].

[24] Welsh, Matt (2013). *Running a software team at Google*. http://matt-welsh.blogspot.com/2013/04/running-software-team-at-google.html [2015-03-03].

[25] Agomouh, Fionna (2014). *Samsung Galaxy S5 Hits 10 Million Sales Mark: Smartphone Beats Galaxy S4 Initial Sales Record By Two Days*. http://www.ibtimes.com/samsung-galaxy-s5-hits-10-million-sales-mark-smartphone-beats-galaxy-s4-initial-sales-1582476 [2015-03-25].

[26] BarcodeEye (2014). *BarcodeEye*. https://github.com/BarcodeEye [2015-01-27].

[27] Fuller, Andrew (2014). *Decoding small QR codes by hand*. http://blog.qartis.com/decoding-small-qr-codes-by-hand/ [2015-03-30].

[28] Google (2014). *Find a Preferred Provider*. http://www.google.com/glass/help/frames/providers/ [2015-02-02].

[29] Google (2014). *Frames*. http://www.google.com/glass/help/frames/ [2015-02-02].

[30] Google (2014). *MyGlass*. https://www.google.com/glass/help/myglass/ [2015-03-17].

[31] Google (2014). *Thanks to you, Glass just got even better*. https://plus.google.com/+GoogleGlass/posts/1tSYsPCCsGf [2015-03-25].

[32] Stack Overflow (2014). *content near "ok glass" gets grayed out*. http://stackoverflow.com/questions/25533765/content-near-ok-glass-gets-grayed-out [2015-04-29].

[33] Stack Overflow (2014). *Hide "ok glass" menu when enable contextual voice commands*. http://stackoverflow.com/questions/24733681/hide-ok-glass-menu-when-enable-contextual-voice-commands [2015-04-29].

[34] Taylor, Ben (2014). *Why smartphone screens are getting bigger: Specs reveal a surprising story*. http://www.pcworld.com/article/2455169/why-smartphone-screens-are-getting-bigger-specs-reveal-a-surprising-story.html [2015-02-26].

[35] Apple (2015). *Designing for iOS*. https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/MobileHIG/index.html [2015-05-09].

[36] Augmented Reality (2015). *Encyclopædia Britannica Online*. academic.eb.com/EBchecked/topic/1196641/augmented-reality [2015-02-19].

[37] Eclipse (2015). Eclipse. https://eclipse.org [2015-04-17].

[38] Glass Almanac (2015). *The History Of Google Glass*. http://glassalmanac.com/history-google-glass/ [2015-03-25].

[39] GlassUp (2015). *GlassUp*. http://www.glassup.net [2015-02-23].

[40] GlassUp (2015). *GlassUp: Augmented Reality glasses that display messages from your smartphone*. https://www.indiegogo.com/projects/glassup-augmented-reality-glasses-that-display-messages-from-your-smartphone [2015-02-23].

[41] GlassUp (2015). *GlassUp Features*. http://www.glassup.net/features.html [2015-02-26].

[42] Google (2015). Card. https://developers.google.com/glass/develop/gdk/reference/com/google/android/glass/app/Card.html [2015-04-13].

[43] Google (2015). Migrating to android studio. http://developer.android.com/sdk/installing/migrate.html [2015-04-17].

[44] Google (2015). *Android API Differences Report*. http://developer.android.com/sdk/api_diff/22/changes.html [2015-05-11].

[45] Google (2015). *Android Design Principles*. https://developer.android.com/design/get-started/principles.html [2015-02-16].

[46] Google (2015). *Android Studio Overview*. http://developer.android.com/tools/studio/index.html [2015-03-24].

[47] Google (2015). *AsyncTask*. http://developer.android.com/reference/android/os/AsyncTask.html [2015-05-02].

[48] Google (2015). *Card Design - Layout*. https://developers.google.com/glass/develop/gdk/card-design#layouts [2015-03-13].

[49] Google (2015). *Card Regions*. https://developers.google.com/glass/design/style [2015-02-16].

[50] Google (2015). *GDK Quick Start*. https://developers.google.com/glass/develop/gdk/quick-start [2015-05-22].

[51] Google (2015). *Glassware Flow Designer*. https://developers.google.com/glass/tools-downloads/glassware-flow-designer [2015-03-12].

[52] Google (2015). *Google Glass*. https://www.google.com/glass/start/ [2015-05-06].

[53] Google (2015). *Head Wake Up*. https://support.google.com/glass/answer/3079855 [2015-03-02].

[54] Google (2015). *Principles*. https://developers.google.com/glass/design/principles [2015-02-02].

[55] Google (2015). *ProgressDialog*. http://developer.android.com/reference/android/app/ProgressDialog.html [2015-05-05].

[56] Google (2015). *Roboto*. http://www.google.com/fonts/specimen/Roboto [2015-03-16].

[57] Google (2015). *Slider*. https://developers.google.com/glass/develop/gdk/slider [2015-05-05].

[58] Google (2015). *Start building great Glassware*. https://developers.google.com/glass/ [2015-05-11].

[59] Google (2015). *Tech spechs*. https://support.google.com/glass/answer/3064128?hl=en [2015-02-24].

[60] Google (2015). *Typography*. http://developer.android.com/design/style/typography.html [2015-05-22].

[61] Google (2015). *Voice Command Checklist*. https://developers.google.com/glass/distribute/voice-checklist [2015-03-16].

[62] Google (2015). *Voice input*. https://developers.google.com/glass/develop/gdk/voice [2015-03-02].

[63] Google (2015). *Voice Input*. https://developers.google.com/glass/develop/gdk/voice [2015–04-29].

[64] Google (2015). *VoiceTriggers.Command*. https://developers.google.com/glass/develop/gdk/reference/com/google/android/glass/app/VoiceTriggers.Command [2015-03-17].

[65] Google (2015). *We're graduating from Google[x] labs*. https://plus.google.com/+GoogleGlass/posts/9uiwXY42tvc [2015-05-07].

[66] Google (2015). *What is API Level?* http://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels [2015-05-11].

[67] IGN (2015). *Microsoft's HoloLens Live Demonstration* [Online video]. http://youtu.be/b6sL_5Wgvrg [2015-03-04].

[68] Microsoft (2015). *Microsoft HoloLens*. http://www.microsoft.com/microsoft-hololens/en-us [2015-02-23].

[69] Microsoft (2015). *Microsoft Hololens Concept Video* [Online video]. http://compass.surface.com/assets/b2/15/b215c2a2-1c5b-4707-9a92-fffca6fb82fe.mp4?n=Microsoft_HoloLens_TransformYourWorld_816p23.mp4 [2015-02-23].

[70] Oculus (2015). *Low Latency 360° Head Tracking*. https://www.oculus.com/rift/ [2015-02-09].

[71] Pennsylvania State University (2015). *Confidence Intervals and the Central Limit Theorem*. https://onlinecourses.science.psu.edu/stat506/node/8 [2015-03-25].

[72] Penny (2015). *PDS Projecting Display System*. http://www.penny.se/eng/products/pds.html [2015-03-02].

[73] Penny (2015). *Penny*. http://www.penny.se/eng/index.html [2015-02-23].

[74] Penny (2015). *Penny Products*. http://www.penny.se/eng/products/c-moreig.html [2015-02-26].

[75] Recon Instruments (2015). *Recon Jet*. http://www.reconinstruments.com/products/jet/ [2015-02-26].

[76] Recon Instruments (2015). *Technical Specification*. http://www.reconinstruments.com/products/jet/tech-specs/ [2015-02-26].

[77] Virtual Reality (VR) (2015). *Encyclopædia Britannica Online*. http://academic.eb.com/EBchecked/topic/630181/virtual-reality-VR/ [2015-02-19].

[78] Wikipedia (2015). *Bone conduction*. http://en.wikipedia.org/wiki/Bone_conduction [2015-02-24].

[79] Wikipedia (2015). *Cuneiform*. http://en.wikipedia.org/wiki/Cuneiform [2015-02-23].

[80] Wikipedia (2015). *Google Glass*. http://en.wikipedia.org/wiki/Google_Glass [2015-04-07].

[81] Wikipedia (2015). *Head-Mounted Display*. http://en.wikipedia.org/wiki/Head-mounted_display [2015-02-23].

[82] Wikipedia (2015). *Head-Up Display*. http://en.wikipedia.org/wiki/Head-up_display [2015-02-23].

[83] Wikipedia (2015). *International Standard Book Number*. http://en.wikipedia.org/wiki/International_Standard_Book_Number [2015-05-22].

[84] Wikipedia (2015). *iPhone*. http://en.wikipedia.org/wiki/IPhone [2015-03-04].

[85] Wikipedia (2015). *Optical Head-Mounted Display*. http://en.wikipedia.org/wiki/Optical_head-mounted_display [2015-02-23].

[86] Wikipedia (2015). *QR Code*. http://en.wikipedia.org/wiki/QR_code [2015-02-24].

[87] Wikipedia (2015). *Samsung Galaxy SII*. http://en.wikipedia.org/wiki/Samsung_Galaxy_S_II [2015-03-25].

[88] Wikipedia (2015). *Samsung Galaxy SIII*. http://en.wikipedia.org/wiki/Samsung_Galaxy_S_III [2015-03-25].

[89] Wikipedia (2015). *Singleton Pattern*. http://en.wikipedia.org/wiki/Singleton_pattern [2015-05-22].

[90] Wikipedia (2015). *Virtual Image*. http://en.wikipedia.org/wiki/Virtual_image [2015-02-27].

[91] Zxing Project (2015). *Zxing project*. `https://github.com/zxing` [2015-01-27].

[92] Hoorn, Richard. *Google Glass: The Design, implementation, and Performance Evaluation of an assembly system for Google Glass*. Master's thesis, Karlstad University, 2015.

[93] robomatics (2013). *How to Decode a QR Code by Hand* [Online video]. `https://youtu.be/KA8hDldvfv0` [2015-03-30].

[94] Allén, Sture Selander, Einar. *[Information on Information]*. Studentlitteratur, Lund, 1985 (In Swedish).

[95] Starner, Thad. Project glass: An extension of the self. *Pervasive Computing, IEEE*, 12(2):14–16, 2013.

[96] Laramee, Robert S. Ware, Colin. Rivalry and interference with a head-mounted display. *ACM Trans. Comput.-Hum. Interact.*, 9(3):238–251, sep 2002.

A Abbreviations

2D Two-Dimensional

3D Three-Dimensional

API Application Programming Interface

BCT Bone Conduction Transducer

GDK Glass Development Kit

GUI Graphical User Interface

HMD Head-Mounted Display

HUD Heads-Up Display

IDE Integrated Development Environment

ISBN International Standard Book Number

OHMD Optical Head-Mounted Display

QR Code Quick Response Code

ZXing Zebra Crossing

B Results

Table B.1: Results of scanning QR code at different distances with Google Glass

Test Number	1 decimeter	2 decimeters	3 decimeters
1	1884216308	1798065186	2296325683
2	1682800293	1705902100	2415893555
3	2043151856	1937561035	2408782959
4	2487091065	1779327392	2346679688
5	1316070557	1948822022	2336975098
6	1631652832	1777801514	2341278076
7	1576843262	1941070556	2184875488
8	1772125244	2041961670	2347503662
9	2140167236	2060516358	2296203613
10	1911987304	1757171630	2170745849
11	1884277345	1767211914	2357238769
12	1929656983	2313415528	2337341308
13	1709838868	1731719971	2460479736
14	1819946288	1789154053	2208862304
15	1881225586	1959869384	2198150634
16	1790405272	1852996826	2317962646
17	1574829101	1649383545	1774505615
18	1825531006	1702728271	2231719971
19	1731201172	1674377441	2281311035
20	2475097657	1735839844	1698455811
21	2370330810	1776123047	2375762940
22	1872375489	1714141846	2186004639
23	2327819824	2099090575	2258178712
24	1753479003	1826843263	2193664551
25	1721496582	1849182128	2280761718
26	2481842039	1802490234	1814788818
27	1602935791	1912353515	1650085449
28	2181121826	1636383056	2349945068
29	1797210694	1716491699	2250000000
30	2422576904	1698699951	2444274902

Table B.2: Results of scanning QR code at different distances with Samsung Galaxy SII

Test Number	**1 decimeter**	**2 decimeters**	**3 decimeters**
1	2139868085	1486211750	1643240375
2	1467511210	1433458333	1645772458
3	1531975085	1544185626	2269734332
4	1999989252	1474127458	1652179501
5	2117759919	1437559418	1784042751
6	1590932000	1346899542	1959105167
7	2309159125	1542022917	2183700002
8	1722160542	1421151250	1430372917
9	1442118042	1537669377	2517021085
10	2225043835	1922686667	1584863210
11	2559335792	1634340084	1544045376
12	2087568125	1493966751	2455554542
13	2399355752	1654488126	2096796210
14	1835165960	1681548458	1444782750
15	2013470417	1577887875	2794100627
16	1947420919	1971720543	2164583833
17	1914936335	1497237917	1976264377
18	1964812292	1473154916	2001674627
19	1836791835	1332783794	2090852043
20	1291630877	1760558543	1331759376
21	1715746333	1808396667	1840536334
22	2281021168	1940200418	2039864458
23	2106378627	1551906084	2214860250
24	2048848293	2127196127	1541294919
25	1953401960	1389226667	1873071125
26	1901472292	1534745500	2040191418
27	2009275501	1439262916	1916855625
28	1843962793	2145346709	1627921959
29	2639261585	2315900375	1939550709
30	2082421543	2017870959	2048423918

Table B.3: Results of scanning QR code at different distances with Samsung Galaxy SIII

Test Number	**1 decimeter**	**2 decimeters**	**3 decimeters**
1	1749676834	1779497917	1513588627
2	1980956916	1389859375	1552651834
3	1784749169	1264649875	1513143459
4	1842951960	1513206585	1352977542
5	1754621419	1473651001	1307146084
6	1644033335	1392401627	1453716167
7	1730130585	1282980458	1582548250
8	1672027044	1688112459	1311798583
9	1737328751	1432261543	1462818377
10	2373175125	1317017667	1316425376
11	1848027542	1463129292	2047718001
12	1576518375	1676956292	1675308210
13	1642278708	1709122334	1916630376
14	1750943335	1423191250	2168636127
15	1792001168	1402313542	1774322918
16	1733112083	1544061083	1798746335
17	2075140127	1554050960	2053470334
18	1940637085	1406057500	1583177875
19	1879888500	1666031252	1327241125
20	1690253250	1350420626	1350043708
21	1694755291	1506406918	1429665709
22	1966512835	1445166958	1401702668
23	1741643376	1664807709	1242497667
24	2036178543	1692449501	1511735834
25	1694977000	1474073667	1383570126
26	1851916252	1655224458	1549562168
27	1653187293	1203773709	1560395085
28	1966861500	1297300542	1317753292
29	1885895208	1287375710	1785062417
30	2499305419	1667945793	1821073459

Table B.4: Results of scanning QR code of different densities with Google Glass

	1 Encoded Character	**50 Encoded Characters**	**100 Encoded Characters**
1	1798065186		
2	1705902100		
3	1937561035		
4	1779327392		
5	1948822022		
6	1777801514		
7	1941070556		
8	2041961670		
9	2060516358		
10	1757171630		
11	1767211914		
12	2313415528		
13	1731719971		
14	1789154053		
15	1959869384		
16	1852996826		
17	1649383545		
18	1702728271		
19	1674377441		
20	1735839844		
21	1776123047		
22	1714141846		
23	2099090575		
24	1826843263		
25	1849182128		
26	1802490234		
27	1912353515		
28	1636383056		
29	1716491699		
30	1698699951		

Table B.5: Results of scanning QR code of different densities with Samsung Galaxy SII

	1 Encoded Character	**50 Encoded Characters**	**100 Encoded Characters**
1	1486211750	2024130959	
2	1433458333	1969146751	
3	1544185626	2236472876	
4	1474127458	2076327083	
5	1437559418	2219363335	
6	1346899542	2231630084	
7	1542022917	2143448876	
8	1421151250	1874537335	
9	1537669377	1969802000	
10	1922686667	1959907251	
11	1634340084	1999703669	
12	1493966751	1916870585	
13	1654488126	2090464252	
14	1681548458	2093513208	
15	1577887875	2254044209	
16	1971720543	2120755668	
17	1497237917	2253129752	
18	1473154916	2184249042	
19	1332783794	2064304543	
20	1760558543	2360741002	
21	1808396667	2101858835	
22	1940200418	2167193710	
23	1551906084	2046369502	
24	2127196127	2310106833	
25	1389226667	2130918084	
26	1534745500	2046775085	
27	1439262916	1942044501	
28	2145346709	2240394250	
29	2315900375	1967590667	
30	2017870959	1897246252	

Table B.6: Results of scanning QR code of different densities with Samsung Galaxy SIII

	1 Encoded Character	**50 Encoded Characters**	**100 Encoded Characters**
1	1779497917	2032751544	1887171167
2	1389859375	1998325293	1819777001
3	1264649875	1955494667	1988476627
4	1513206585	1656547125	1916022668
5	1473651001	2189044751	1880171334
6	1392401627	1972979041	1727605668
7	1282980458	2132110417	2092458209
8	1688112459	1718851126	2213131667
9	1432261543	1866888252	1979007460
10	1317017667	1697766835	1803592752
11	1463129292	1723378125	2000521042
12	1676956292	2041532793	2135961500
13	1709122334	1988374500	2045468750
14	1423191250	1913362543	1959357169
15	1402313542	1738970334	1991219667
16	1544061083	1995461668	2075882709
17	1554050960	1822996709	1895777792
18	1406057500	2099073044	2025029959
19	1666031252	2030953417	2111278042
20	1350420626	2145817792	1914733292
21	1506406918	1953587583	1947725500
22	1445166958	1875210209	2060683710
23	1664807709	1959188459	1814745458
24	1692449501	2109525960	1782884626
25	1474073667	1846324500	1886035542
26	1655224458	1911180292	1872026709
27	1203773709	1646990960	2026801625
28	1297300542	1744290210	1683916542
29	1287375710	1667269792	1893676792
30	1667945793	1966602667	2061173500

Table B.7: Results of displaying information of different sizes on Google Glass

Test Number	**1 kB**	**100 kB**	**1 MB**
1	253570557	442565918	605865479
2	302947998	368041992	542236329
3	309509278	388549804	591369630
4	330413818	382720947	637756348
5	325225830	408355713	436096192
6	308532715	345550537	547424317
7	339141846	468597412	575408936
8	163146973	405181885	383300781
9	350677490	363525391	563415528
10	317047119	258819580	511657714
11	289825439	387298584	581054688
12	334625243	368591309	683319091
13	357788086	358184814	531799316
14	286682129	256286621	553283693
15	342498779	376190186	551635742
16	309417724	403717041	580993652
17	298553467	462371827	549407959
18	283508300	394836426	582153320
19	348236084	396270752	706787110
20	319580079	516387940	476898193
21	313873291	540405274	576416016
22	320098878	521606446	563262940
23	288299561	438934327	603668214
24	308593750	727264404	553710937
25	338897705	550872803	564880371
26	321868897	534332277	582336426
27	332733154	549041748	693267823
28	297607421	557159424	748901368
29	317657470	543640137	506378174
30	313507080	557922364	880432129

Table B.8: Results of displaying information of different sizes on Samsung Galaxy SII

Test Number	**1 kB**	**100 kB**	**1 MB**
1	6797250	14496709	64364083
2	6326709	43183958	100282458
3	6746292	24071374	18386459
4	7058875	31881374	80973374
5	11829542	28462958	84470625
6	10667375	13489125	109992750
7	8890167	12720376	18701084
8	21200959	28104084	11930708
9	5742833	14041624	37545749
10	7117500	34152708	37839583
11	6092042	13555542	32261416
12	7110333	14802667	26862500
13	11012625	14330666	32655250
14	9552500	29462791	38692001
15	8780000	40578625	73218541
16	10533500	11147917	11013791
17	5902250	13767874	10906625
18	10135791	12711084	137211250
19	8074709	13150458	29192460
20	8935250	13934084	102380958
21	6071583	14262083	29384416
22	6577125	15152625	31167750
23	6078875	23965041	32326793
24	5932333	29859749	38545333
25	6505000	28264250	35596375
26	8270792	19016750	79581583
27	7437583	13678375	88178667
28	10711625	13444250	69627500
29	5428708	19024708	26571250
30	9704542	14632958	38301208

Table B.9: Results of displaying information of different sizes on Samsung Galaxy SIII

Test Number	**1 kB**	**100 kB**	**1 MB**
1	45458667	39401666	87967500
2	19902667	31100959	84084292
3	41686916	26440126	74972542
4	20914917	29435958	90547042
5	23337708	47066459	73427167
6	22120000	52401791	37535458
7	22092541	32144417	78092125
8	26526917	35520667	23304625
9	21858542	47763958	69872875
10	21574041	29292875	70221041
11	26992500	36875791	118110042
12	22648708	37272000	113794584
13	20634042	26331666	69038042
14	23051125	31668000	102973167
15	24605791	27136583	72928792
16	34179333	29959792	66249875
17	24432041	29541250	81965833
18	22108541	26804375	67999167
19	22469167	31402334	79773458
20	20666375	28152458	112654583
21	21058209	32599208	69238500
22	31040333	25747000	66882792
23	22904250	33012250	67235334
24	25762126	37820708	94330208
25	29134500	33178042	81698416
26	23086958	39990250	74808583
27	26097626	36171749	73372666
28	29608500	44156125	86602001
29	25589501	38544333	72212833
30	23425959	26356833	83208958

C Code

See separate file.

D Project Specification (In Swedish)

Uppdragsbeskrivning

Google Glass

Version 1.0

Mats Persson

Distributionslista

Befattning	Bolag/enhet	Namn	Åtgärd	Info.
Student	KaU	Richard Hoorn		
Student	KaU	Johan Häger		
Konsult/handledare	Sogeti	Mats Persson		
Konsultchef	Sogeti	Åsa Maspers		
Säljare	Sogeti	Bengt Löwenhamn		

Innehållsförteckning

Ändringsförteckning

Version	Datum	Ändring
1.0	2014-10-15	Dokumentet skapats

1. Allmän beskrivning av uppdraget

1.1 Bakgrund

Sogeti Sverige AB (Sogeti) är ett IT-konsultbolag med bred verksamhet, stort fokus på kompetens och modern teknik.

Sogeti jobbar mycket med applikationer inom mobilitet och en av de nyaste teknikerna här är Google Glass.

Google Glass introducerar ett helt nytt sätt att tänka på angående hur man interagerar med applikationer och helt nya användningsområden för mobila applikationer.

2. Google Glass

Syftet med detta uppdrag är att tillverka en Proof of Concept på en applikation till Google Glass, applikationen ska kunna hämta/spara data från en web service. Applikationen ska hjälpa användare i en produktion genom att bidra med vilka delar som behövs för att kunna montera något och även en enklare instruktion över hur det ska se ut när det är klart.

Ett annat viktigt syfte med detta uppdrag är att ta reda på hur Google Glass är ur ett användarperspektiv, är det behagligt att använda en hel dag? Osv.
Även prestandamässigt är det intressant för att veta om Google Glass är nog stabila för att kunna rekommenderas till kunder på företag inom tillverkningsindustrin.
Och slutligen så är bra/dåliga sidor med denna nya teknik intressant. Finns det begräsningar? T.ex. hur länge räcker batteri m.m.

I uppdraget ingår en back-end och en front-end och nedan ser vi lite exempel på saker man kan fokusera på inom de olika delarna.

I back-end finns följande att fokusera på:

- Prestanda, hur kan man optimera en back-end som pratar med liknande appar för att skicka så lite data som möjligt så snabbt som möjligt?
- Eftersom vi jobbar med Microsofts plattform så är det intressant att veta om man ser tydliga plus/minus med att hosta tjänsten i Azure istället för att man t.ex. hostar den själv på sitt egna nätverk.

Vi ser helst att back-end utvecklas med WebAPI eller WCF då Sogeti har stort fokus på Microsofts plattform.

I front-end som ska köras på Google glass ser vi följande man kan fokusera på:

- Användarvänlighet. Som när touchenheter kom så är detta också ett nytt unikt sätt att interagera på, hur görs detta på bästa sätt? Vad skiljer sig från applikationer till telefoner?
- Multitasking, hur mycket kan göra samtidigt och hur hanteras detta?
- Prestanda

2.1 Case

Följande case ska uppdragstagarna utgå ifrån.

Ett företag har en produktionslinje med ett antal stationer där man monterar ihop saker vid varje station. Delarna man använder för monteringen har en kod på sig som man kan scanna av. Problemet företaget har är att det är många variationer av saker som monteras på varje station och arbetarna byter ofta platser så de har svårt att hålla reda på vad som behövs för att montera och hur det ska monteras.

2.2 . Optioner

Följande är förslag på vidareutveckling av detta som uppdragstagarna själv får plocka från om tid finns.

2.2.1 Option 1 – Jämföra Google Glass med andra liknande produkter

Jämföra Google Glass med en annan liknande produkt genom att välja en mindre del av applikationen som man implementerar. Ett förslag på annan produkt är företaget Pennys motsvarighet till Google Glass.

2.2.2 Option 2 – Bildigenkänning

För att slippa att ha koder på varje produkt så skulle det vara bra om applikationen kan känna igen produkterna genom att man bara tittar på produkterna, detta dels för att se vilka möjligheter det finns för bildigenkänning på Google Glass och för att ta reda på vad som redan finns färdigt inom detta område och vad som krävs av företaget för att få bildigenkänning att fungera.

2.2.3 Option 3 – Känna av position

För att underlätta för arbetarna så skulle det vara bra om applikationen kunde känna igen vart i produktionslinjen man sitter och jobbar och då filtrera listan av produkter man har att välja mellan så det blir lättare att hitta produkter.

2.2.4 Option 4 - Röststyrning

Kunna styra applikationen med rösten och ta reda på vad det finns för begränsningar för detta. T.ex. hur tyst måste det vara runt användaren för att det ska fungera?

3. Genomförande/arbetssätt

3.1 Rutiner

Sogeti tillhandahåller arbetsplatser, datorer, hårdvara samt erforderliga utvecklingsverktyg.
Uppdragstagarna kommer att ha access till Sogetis nätverk och förväntas nyttja vår TFS-server för versionshantering.

3.2 Genomförande

Uppdragstagarna planerar själv genomförandet och Sogeti tillhandahåller stöttning både projektstyrningsmässigt och rent implementationstekniskt. Sogeti tillhandahåller all programvara och hårdvara som behövs.

Förslagsvis används SCRUM med en sprintlängd på 2-3 veckor som sätts upp där uppdragstagarna specificerar vad de tror att de hinner med i början av varje sprint och har en demo för en eller flera på Sogeti i slutet på varje sprint.

4. Stöd/kvalitetssäkring

4.1 Granskningar

Vid behov genomförs granskning som kan initieras av både handledare och uppdragstagare.

Lämpligen definieras några granskningspunkter vid planeringen av projektet.

4.2 Testarbete

Funktions-, system- och integrationstest görs av uppdragstagarna.

5. Leveranser

5.1 Dokumentation

Systemdokumentation görs av uppdragstagarna. Dokumenten lagras i projektarkiv hos Sogeti.

Lämpligen levereras en enkel användarinstruktion.

6. Konfigurationsstyrning

All programkod och tillhörande specifikationer och andra utvecklingsdokument ska versionshanteras med hjälp av Microsoft TFS.

7. Miljö

Utvecklingsverktyg väljs av uppdragstagarna tillsammans med handledare. Hårdvara som behövs för projektet tillhandahålls av Sogeti.

8. Uppföljning och Rapportering

8.1 Rapportering internt/externt

8.1.1 Statusrapportering

Rapportering av status och framskridande i utvecklingen beslutas i samråd vid projektuppstart.

8.1.2 Möten

Möten hålls vid behov. Vid uppstart läggs lämpligt antal avstämningsmöten in i projektplanen.

8.1.3 Slutrapportering

Arbetet presenteras för Sogeti i samband med lämpligt månadsmöte alternativt lunchmöte.