THE WEARABLE INSTRUMENT APPROACH FOR PILOTS

A Thesis

by

ERIC MICHAEL BURKE

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Chair of Committee,	Ann McNamara
Committee Members,	Louis Tassinary
	Thomas Ferris
	Sherman Finch
Head of Department,	Tim McLaughlin

August 2015

Major Subject: Visualization

ABSTRACT

Pilot errors caused by heads-down time or misinterpretation of published instrument approach procedures have been attributed to multiple incidents of fatal controlled flight into terrain while approaching airports in instrument meteorological conditions. This study was motivated by the idea that wearable heads-up devices such as Google Glass can supplement standard paper or tablet-based instrument approach plates by decreasing heads-down time and pilot error. In order to evaluate the utility of Google Glass in the field of aviation, this thesis was comprised of two phases: the development of a custom instrument approach software application for Google Glass, and the execution of a simulator study to compare the effects between the usage of Google Glass and current tablet-based instrument approach plates in regards to pilot error, preference, and heads-down time. Results showed that the introduction of Google Glass into the cockpit can help pilots fly a safer approach when compared to simply using a tablet-based approach plate alone. More specifically, when Google Glass was used together with a tablet-based instrument approach plate, pilots had a quicker reaction time when they did indeed commit a navigational error, and their total heads-down time was reduced, allowing them to focus more on cockpit instrumentation and flying the aircraft. While Google Glass is currently a moot point in the gadget world, the knowledge gained from this research should translate well to the development of more advanced software for forthcoming wearable heads-up devices.

DEDICATION

To my family

ACKNOWLEDGEMENTS

First and foremost, thank you to my wife Josephine for all of your patience, belief, nerves, and support during this entire journey. Thank you for all of those times when you waited on me to come home after spending all day and/or night at the lab. When I lost my way, you would help me find it again; you could find encouragement in my most discouraging of moments. A huge part of my success is because of you.

Special thanks to Kevin Gabriel, owner of Solo Flight Training in Houston, Texas and host of the simulator study for this research. Your kindness and willingness to help a stranger is most appreciated. Thank you to Professor and Aerospace Medical Association Safety Committee member Dr. Douglas Boyd for all of the extremely helpful aviation articles and information you have provided to me.

Thanks to my brother and USAF C-17 pilot Nick Burke, and to USAF C-130 pilot Aaron Sanchez, as well as former USAF F-16 and commercial "big iron" pilot Mr. Ed Cole. I could not have developed the study's software without all of your feedback and insights.

Thank you to my parents for helping us find our footing before starting the master's program, and for allowing me to set up a makeshift flight simulator in your house.

Advisory committee: thank you to committee chair Ann McNamara for meeting with me every week; to Louis Tassinary for convincing me to start this project as early as possible; to Thomas Ferris for all of the equipment from your Human Factors Lab; to Sherman Finch for teaching me the true meanings of mobile technology and user experience.

NOMENCLATURE

ADS-B	Automatic Dependent Surveillance Broadcast System
API	Application Programming Interface
ASRS	Aviation Safety Reporting System
ATIS	Automatic Terminal Information Service
EAP	Electronic Approach Plate
EFB	Electronic Flight Bag
FAA	Federal Aviation Administration
GA	General Aviation
HUD	Heads-Up Display
HMD	Head-Mounted Display
IAP	Instrument Approach Plate
IDE	Integrated Development Environment
IFR	Instrument Flight Rules
IHADSS	Integrated Helmet and Display Sight System
IMC	Instrument Meteorological Conditions
KIAH	Airport Code for the George Bush Intercontinental Airport
MDA	Minimum Descent Altitude
NDB	Non-Directional Beacon
PIC	Pilot in Command
UDP	User Datagram Protocol
VFR	Visual Flight Rules

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

TABLE Page

1. INTRODUCTION

The approach and subsequent landing of an aircraft in instrument meteorological conditions (IMC) is arguably one of the most complex and critical phases of flight. As pilots transition from the en route "cruise" phase to the approach, their workload increases as they prepare the aircraft for landing. This increase of workload can drastically increase a pilot's vulnerability to errors, especially when navigating through adverse weather conditions. In order to safely fly an approach, pilots use Federal Aviation Administration-certified instrument approach plates (IAPs), as seen in Figure 1.1. IAPs provide navigation, communication, and procedural information "to transition from en route flight to approach and landing, or if necessary, to execute a missed approach" [19, p. 7]. Traditionally, the IAP and other information vital to the safe and normal operation of the aircraft has been presented in paper format [19]. In addition to the IAP, this information includes flight manuals, checklists, and aeronautical charts including approach plates [26, p. 8D1-1]. Paper IAPs are black text on a white page, and provide pilots with the aeronautical information required to execute instrument approaches to airports [16]. As such, the sheer quantity of paper material required by pilots to carry onboard an aircraft can weigh up to 40 pounds. Moreover, many of these paper charts have a lifespan of only 56 days and must be continually replaced.

The layout of an instrument approach plate is depicted by areas A through D in Figure 1.1. Area A contains communication and airport identification information; Area B contains a plan view or overhead depiction of the terminal area; Area C displays a profile depiction of vertical navigation information; Area D displays minima data for landing including ceiling visibility minimums according to the type of

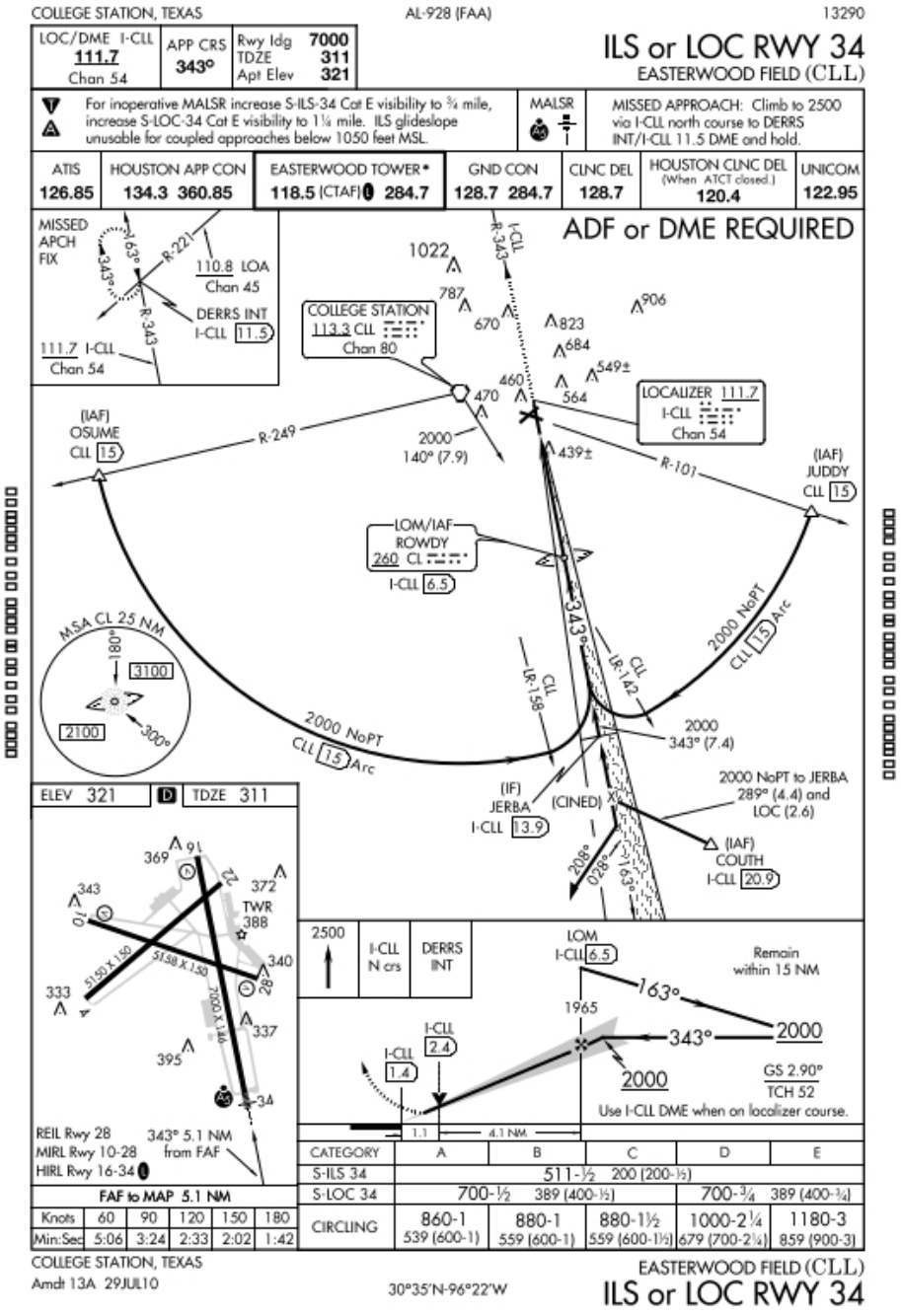

Figure 1.1: An IAP of Easterwood airport, College Station, Texas.

aircraft flying the approach [19].

Before beginning an approach, "the obvious starting point in approach plate review is to find out as early as possible what approach you will fly" [18, p. 85]. This is usually done during the enroute phase of flight using the automated terminal information service (ATIS) frequency found in AREA A of figure 1.1. The next step is to familiarize oneself with the airport's plan (overhead) view, as seen in AREA C. Afterwards, the pilot tunes in the frequency of the "localizer, VHF omnidirectional range, or non-directional beacon used for the final approach" into the navigation radio [18, p. 86]. The localizer, which was used primarily in this study, is an antenna located on the departure end of the runway and provides approaching aircraft with a "full 'fly-left' and a full 'fly-right' indication" [5, p. 127]. Next, the final approach course in AREA A is set using a dial on the horizontal situation indicator (HSI). The HSI is an instrument that provides the pilot with vertical and/or lateral guidance to the approach course. The pilot then turns his/her attention to the step-down fixes and minimum descent altitude (AREA D). Upon reaching the final approach fix shown in AREA B and C, the pilot starts a countdown timer to establish the missed approach point. The time can be compared with the list under the airport diagram in AREA D. If the timer runs out before the pilot breaks out of the clouds and establishes visual contact with the runway, the pilot is required to execute a missed approach. Instructions for the missed approach can be found in AREA A and C. While not a comprehensive explanation of how instrument approaches are flown, the above paragraph serves as an example of the linear flow of information on the approach chart. Note that the pilot does not need all of the information from the chart at once; only small amounts of this information are relevant at any given time during the approach.

Today, the paper-based flight bag and paper IAPs are slowly disappearing from

the cockpit due to the proliferation of the electronic flight bag (EFB). The EFB contains digital copies of the paper IAPs, and has been designed to "replace the heavy and cumbersome traditional [paper] pilot flight bag," with a small and lightweight device such as the iPad, android tablet, smartphone, or other hand-held mobile hardware [1, p. 4]. The EFB can contain a digital database of every IAP in existence and update them automatically. Indeed, the electronic version of the instrument approach plate offers a more flexible way of presenting approach information to the aircrew, and is currently used by many general aviation (GA) and airline pilots [19]. This feature, however, makes the task of designing EFBs and navigation-related software crucial because they will be used mainly during the phases of flight associated with a majority of aircraft accidents [11]. Due to the proliferation of the EFB, the electronic instrument approach plate will be considered as the "traditional" method of displaying approach information throughout the remainder of this paper.

There are quite a few commercial EFBs available to general aviation pilots that offer high resolution and current databases of nearly every IAP in existence. Popular examples include Garmin Pilot, ForeFlight, WingX Pro, and Avare. These applications run on Android and iOS devices; Garmin Pilot and WingX are supported on Android and iOS, while ForeFlight is available only for iOS. Due to a phone's screen size, it is more common to see these applications used on a tablet [6]. The EFB can use the device's built-in GPS receiver in order to provide situational awareness, ground speed, altitude, rate of turn and vertical airspeed to the pilot via user interface [10]. For increased GPS accuracy, an iPad or Android tablet can be wirelessly tethered via Bluetooth to an external GPS receiver such as the commercially-available Dual Electronics iPad/Android GPS XGPS150A. In regards to electronic IAPs, most of these applications can offer real-time position information of the airplane, superimposing a moving blue dot in the chart's plan view (AREA B

of figure 1.1).

As robust as these applications may be, they all share an intrinsic flaw: they do not provide constant heads-up capability. Instead, the pilot constantly needs to look away from cockpit instrumentation in order to read relevant approach information. In addition, electronic IAPs are just as complex and cluttered as paper IAPs, due to the fact that they are simply digital copies of their paper counterparts. Pilots must direct their attention towards the tablet displaying the approach plate, which is usually located out of direct line-of-sight in the cockpit. This can lead to safety issues where the pilot becomes distracted from actually flying the airplane, especially in rough weather conditions. As one can see in Figure 1.2, a pilot has used a stylus to manually re-write information in colorful, large numbers in order to disambiguate the important information from the irrelevant, thus enabling a shorter glance at the screen. As workload increases in the cockpit, these notes will most likely prove themselves to be very helpful, highlighting the most pertinent information. Regarding the iPad as an EFB, certified flight instructor and professional pilot Michelle Bassenesi (2011) states that "pilots may spend a considerable amount of time heads-down attempting to select and exercise system functions," especially those that have little experience with the device [1, p. 12]. She states that she had witnessed this first-hand as her pilot colleagues became distracted by their iPads during a routine general aviation flight [1, p. 12]. According to ForeFlight Mobile's (2014) FAQ webpage, "personal electronic devices such as the iPad and iPhone can take attention away from the most important and demanding task: flying the airplane and arriving safely" [7]. Indeed, most of incidents that occurred while using the iPad as an EFB, as reported by the Aviation Safety Reporting System, were caused by "distractions due to heads-down time and unfamiliarity with how the iPad worked" [1, p. 13].

One may argue that, despite featuring a real-time moving airplane icon, the elec-

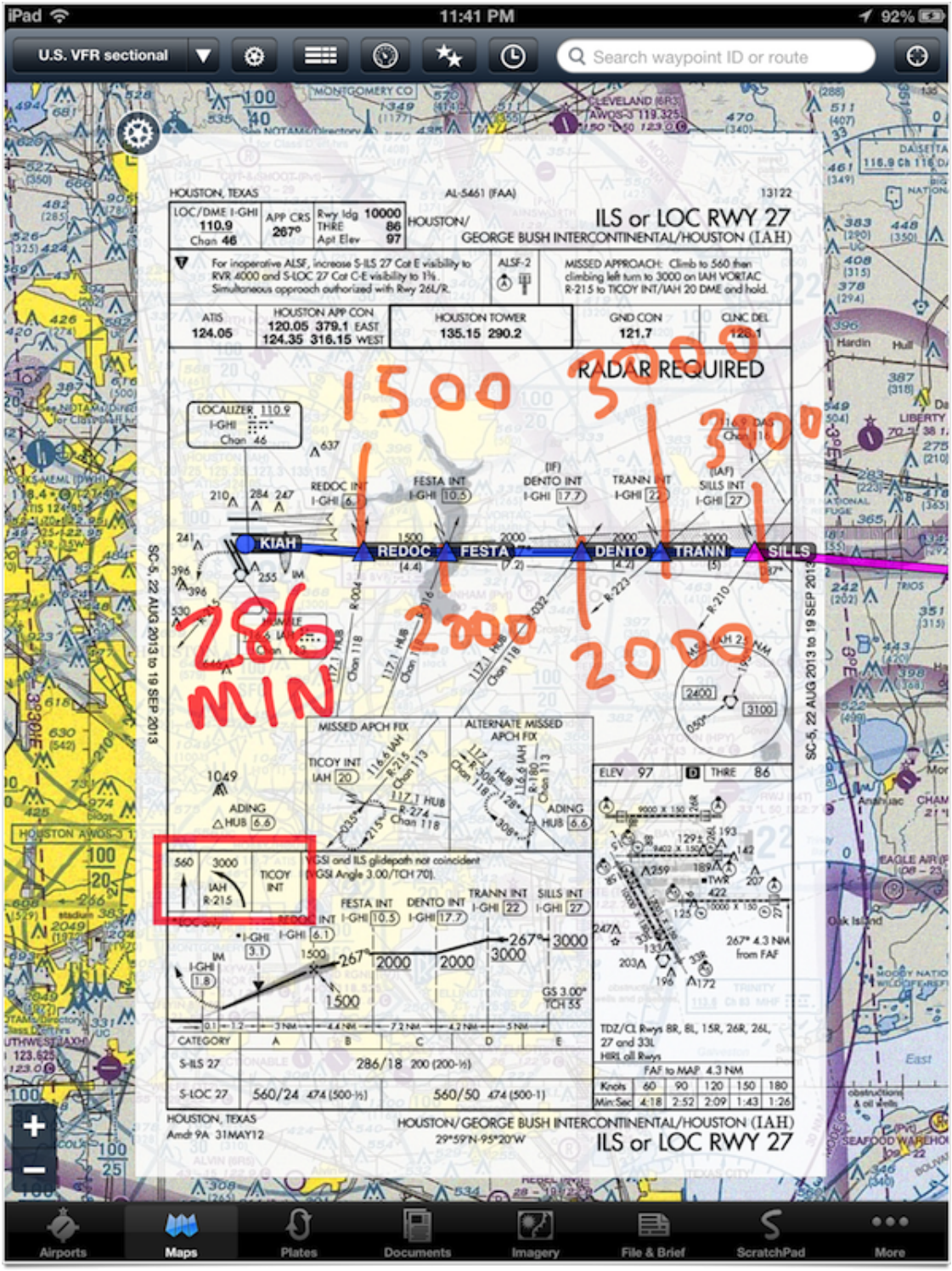

Figure 1.2: Foreflight Electronic IAP on the iPad. A stylus has been used to highlight important information [8].

tronic IAP is equally as cluttered as the traditional paper IAP. As mentioned earlier, notice in Figure 1.2 how the pilot has written the step-down altitudes in large, colorful numbers with a stylus, despite the intrinsic scrolling and zooming capabilities of the iPad. According to Mykityshyn et. al (1994), IAPs are intentionally information dense due to the fact that the ultimate liability of the charting agency and fear of litigation often precludes cartographers from simplifying the charts by removing less important information [19, p. 143].

1.1 Wearable Computers in Aviation

The interest in wearable avionics has been piqued by Google Glass and other wearables that "very well could transform the whole experience of how we fly in the future" [22]. Google Glass is a miniature computer that is worn like a pair of eyeglasses, and features a small screen for displaying information over the wearer's right eye. It can be tethered via Bluetooth to a smartphone, enabling the wearer to read emails, surf the internet, receive instructions from Google Maps, talk on the phone, or send voice-to-text messages [22]. This is not an exhaustive list of Google Glass' capabilities, which has arguably not been exploited by developers since the beginning of the"Explorer Program" in 2013. However, according to *Flying Magazine*, connecting Google Glass to a database of Jeppesen navigation data could allow it to "potentially take over where the iPad or Android tablet leaves off" in respect to the display of navigation information and alerts [22].

1.2 Thesis Overview

Building upon the idea of wearable devices taking over where the tablet leaves off in the cockpit, this thesis aims to assess the utility of Google Glass during the critical approach and landing phase of flight in instrument conditions. This utility will be derived from the device's influence on pilot heads-down time, amount

of errors committed, reaction time, and overall opinion after being used to fly an instrument approach. These results will be compared to the usage of the traditional (electronic) instrument approach plate. It is important to note that results collected from this study are not Google Glass-specific, instead, they should translate well to forthcoming wearable technology.

2. LITERATURE REVIEW

This chapter provides specific evidence regarding the dangers of heads-down time and the complexity of approach procedures in a high-workload cockpit environment. In addition, previous studies and current technology related to the usage of wearable heads-up displays in aviation are introduced.

2.1 Distraction and Loss of Situational Awareness

While conducting an instrument approach on December 24, 1996, a Learjet 35A with two crew members and no passengers struck the ground at 250 knots, over 10 miles away from the Lebanon Municipal Airport in Lebanon, New Hampshire. After executing a missed approach from runway (RWY) 18 at Lebanon Municipal, the pilots attempted another landing at the same airport on RWY 25. At one point during the approach, the captain instructed the first officer to descend to 2,300 feet, which was well below the required altitude of 4,300 feet published on the IAP for their current position. Soon afterwards, the plane impacted trees, then terrain, and both crew members were fatally injured. According to the National Transportation Safety Board (NTSB), the captain's misinterpretation of the published step-down fix passage contributed greatly to the aircraft's early descent into terrain [20]. This misinterpretation may have stemmed from the pilots' rushed briefing of the new instrument procedure which increased their workload, leading to a loss of situational awareness along the approach course. Not only were both pilots unaware of their exact location in relation to the airport, but their preoccupation with other tasks, such as briefing the new approach procedure and flying the airplane distracted them from noticing their navigation errors.

According to Dismukes, Young, and Sumwalt, "crew preoccupation with one

task to the detriment of other tasks is one of the more common forms of error in the cockpit" [4, p. 4]. Dismukes et. al. reviewed NTSB and Aviation Safety Reporting System (ASRS) accident reports attributed to pilot error. Approximately 50% of those accidents resulted from distraction, interruption, or preoccupation. Their analysis showed that the task of monitoring the current status or position of the airplane was most often neglected while the pilot multitasked during critical moments of flight. The competing activities that distracted the pilots from performing other tasks, such as flying the airplane, were grouped into four categories: heads-down work, searching for other air traffic, responding to abnormal situations, and communication. Periods of heads-down work, such as completing paperwork or briefing approach plates, could create a situation that is "especially vulnerable because the monitoring pilot's eyes are diverted from other tasks" such as flying the airplane [4, p. 6]. The authors suggested that reducing the amount of heads-down tasks during critical phases of flight often associated with high workload levels can be achieved by rescheduling such tasks during lower workload phases of flight. In addition, they argued that it may be beneficial to integrate head-down tasks with the routine scan of instrumentation, switching back and forth between heads-up and heads-down tasks as often as possible. The pilot errors caused by distraction reviewed in Dismukes, et. al. affected pilots of all experience levels. According to design and usability expert Donald Norman, an experienced instrument pilot has the potential to manually fly an aircraft in a largely skill-based or subconscious manner. Certain parallel tasks, however, require the pilot to devote cognitive resources, such as levelling off or arresting his/her descent at a required altitude. The task of constantly checking the altimeter to match the minimum required altitude on the approach plate, for example, can interfere with other tasks such as keeping the aircraft on course, or preparing for upcoming approach maneuvers [21].

2.2 Related Work

2.2.1 Experimental Study of Electronically Based Instrument Approach Plates

As the EFB has become more popular in the cockpit due to its space- and weight-saving characteristics, it has been implemented within many different display mediums. In the early 1990s, the lack of high-resolution displays precipitated studies that sought to declutter the information on the plate in order to best utilise the screen's capabilities. One such study was conducted by Mykityshyn, Kuchar, and Hansman (1994), which compared six IAP formats in a case study with thirteen experienced airline pilots. Each IAP format differed from the others by either color, decluttering ability, moving aircraft symbol, or direction (north-up vs. track-up). The IAPs with decluttering ability allowed the pilots to layer information, by either removing or adding information deemed extraneous or important, respectively. Results of the study showed that pilots preferred this decluttering ability over the traditional approach plate format, yet performance results regarding reaction time and error rates were negligible between all formats. This study displayed each IAP format in a consistent location, next to the important navigation instruments such as artificial horizon, altitude, groundspeed, and heading indicators.

The affects of the IAP's location (heads-down vs. heads-up) on experimental results, however, was not taken into consideration. In addition, the technology used to display the IAPs is no longer representative of the commercial devices such as Apple and Android tablets/phones currently used in the cockpit. However, this study relates to the thesis due to the small display area and limited resolution (640 x 360 pixels) offered by Google Glass. In the attempt to create a useful tool for this device, information from the traditional approach plate will need to be presented in a simple, decluttered format.

2.2.2 Instrument Approach Plate Software

Currently there are many commercial over-the-shelf electronic flight bags available. They range "from small handheld, PDA devices targeted for general aviation aircraft to complex, multi-display, server-driven devices for high-end installation" [6, p. 9]. Due to the immense success celebrated by the iPad in personal and professional settings, it has also expanded its utility into the realm of aviation [14]. Examples include Garmin Pilot, Foreflight mobile, Apps4av's Avare, and Hilton Software's WingX Pro. As previously stated, the common issue shared with the devices running these software applications (iPad/Phone, Android tablet/phone) originates from the fact that the incorporation of such EFBs in the cockpit "has increased pilot heads-down time away from their primary task of aviating first, and then navigating and communicating" [14, p. 56]. Although a pilot can use a stylus to highlight important navigation information on the approach charts (Figure 1.2), these applications do not offer real-time altitude or position notifications that provide situational awareness to a possibly distracted and busy pilot.

2.2.3 Brilliant Eyes

In addition to the aforementioned applications for tablets and smartphones, Aerocross in McInney, Texas has developed a working prototype of a wearable HUD similar to Google Glass called "Brilliant Eyes." The first prototype, delivered in February 2013, uses information "generated by a commercially available iPad application that takes input from a portable attitude and heading reference system with GPS and Automatic Dependent Surveillance - Broadcast (ADS-B)" for position, weather, attitude and traffic [2]. Brilliant Eyes is an augmented reality head-mounted device (HMD) that renders an image similar to what would be found in fighter aircraft like the Air Force's F-35. However, this technology has not yet been

released nor published in experimental papers, thus the utility of the device remains unknown due to a lack of information from field testing. Should Aerocross incorporate approach plate procedures into their wearable HUD system, its utility could be compared to results of this thesis, due to the fact that they have taken the idea one step further into the realm of augmented reality.

2.2.4 Aero Glass

Aero Glass has been developing augmented reality solutions for pilots using either Google Glass, Epson Moverio, or other head mounted devices. According to the company, pilots wearing these devices running Aero Glass' software will enjoy heads-up airport, navigation, flight path, air traffic, and weather information, among others. Enjoying enthusiastic response from early adopters, Aero Glass is currently conducting a beta testing phase for their technology. In addition, Aero Glass has been present at Augmented World Expo and EAA Airventure conventions. While displaying visually stunning graphics and screenshots from their devices, little data exists as to the actual utility of the technology.

2.2.5 Adventia European College of Aeronautics

Two pilots from the Adventia European College of Aeronautics flew with Google Glass for the first time in aviation history on March 5, 2014. Emphasizing safety, productivity, efficiency, training, and environmental sustainability, Adventia's Google Glass software was implemented by an official Google development team called "Droiders," adapted from pre-existing checklist "glassware" used by surgeons in the Faculty of Medicine at Stanford University. Droiders reprogrammed this application in order to create checklist glassware used before, during, and after flight. In addition to checklists, the application features a real-time electronic IAP with a superimposed airplane symbol, similar to the electronic IAP found within EFBs like Foreflight.

This feature, however, appears to be merely a proof of concept rather than robust heads-up electronic IAP solution, due to the fact that the software simply displays a part of the IAP, which contains too much information for Glass' small display to be efficiently read by the pilot.

2.2.6 Integrated Helmet and Display Sight System

Pilots of the AH-64 Apache attack helicopter wear a monocular helmet-mounted display called the Integrated Helmet and Display Sight System (IHADSS). This device provides "pilotage, navigation, and weapon-aiming symbology and imagery" to the pilot's right eye [15, p. 110901-2]. Due to size and weight constraints, the IHADSS device is limited only to the pilot's right eye, forcing the pilot to adapt regardless of eye dominance or visual perception issues. Similarly, Google Glass' display location is limited to the wearer's right eye, thus human factors and ergonomic issues related to binocular rivalry, depth perception, image quality, and eye relief faced by pilots using the IHADSS system could be taken into consideration when designing a navigation application for Google Glass [12].

2.3 Summary

In summary, the risk of distraction or misinterpretation of published approach information from an IAP poses a real danger to pilots flying in instrument meteorological conditions. One danger in particular, controlled flight into terrain, is one of the most common types of fatal accidents known to instrument flight [3, p. 46]. Controlled flight into terrain occurs when a perfectly functioning aircraft makes premature contact with the ground resulting in loss of life or property. Such an occurrence is usually attributable to pilot error, much like the December 24, 1996 Learjet crash mentioned earlier in the chapter. To help mitigate this danger, as well as reduce distraction and loss of situational awareness in the cockpit, organizations

like Aerocross and the Adventia College of Aeronautics have been working towards solving this problem with head mounted devices. Currently, however, little information exists from experimental trials regarding the utility of head mounted displays in the civilian or commercial cockpit.

3. THE IMPORTANCE OF RESEARCH

What is the utility of novel head-mounted display (HMD) devices like Google Glass in the cockpit? Due to their heads-up nature, do HMDs decrease distraction and heads-down time by helping the pilot focus his/her attention on flying the airplane during critical phases of flight? Are they better or worse, or do they compliment the current electronic IAP on a tablet device? This study hypothesizes that, while not a replacement for the tablet IAP, a HMD can decrease pilot heads-down time and distraction when used in conjunction with a tablet IAP. Currently, however, there is no substantial published research that answers these questions.

Given the threat of divided attention and loss of situational awareness when multitasking, this thesis is motivated by the idea that HMD technology decreases distraction and pilot heads-down time during critical phases of flight when compared to the usage of current electronic EFB/IAP technologies that are standard in the general aviation cockpit. While not a complete replacement for the EFB, the HMD may offer helpful redundancy when coupled with a tablet IAP. This logic is based on the idea that the pilot will not need to spend large amounts of heads-down time while accessing important approach information, because it is already displayed on Google Glass' screen. Should the pilot misread the tablet IAP or lose situational awareness along the approach course and descend prematurely, Google Glass shall act as a safety barrier, alarming the pilot of this error.

The idea of using heads-up display technology as an aid to pilots has been accepted for decades, and the implementation of HUDs in the military and commercial sectors is nothing new. Currently, such technology is not available to the general aviation pilot, and costs many hundreds of thousands of dollars. The relatively low cost

and arguably small amount of content availabile for novel wearable display devices such as Google Glass, however, makes them attractive for human factors and user experience engineering research. As such, this thesis was developed to accomplish two phases:

Phase 1.) Design and develop a software application for Google Glass (glassware) that displays approach plate information that is relevant to the aircraft's current position. Despite the wealth of data presented on a standard approach plate, any information that is not relevant to the pilot at a specific time point will be ignored. Inspired by the idea that "it is impossible to memorize even a fraction of the information printed on an approach plate," this application will allow the pilot to remain heads-up while reading key information, and then to put it into use immediately afterwards [18, p. 85]. Looking back at the 1996 Learjet 35A accident in Lebanon, New Hampshire, a heads-up visual aid such as Google Glass that presents the current minimum altitude requirements to the pilots may have helped guide their attention to their altitude deviation as they descended below minimums. In addition, situational awareness may have been provided by a simple cross-check between Google Glass' instructions and the instructions published on the traditional IAP, comparing current and next step-down fixes. Finally, this study's prototype was built to be more intuitive and efficient than the aforementioned Google Glass software presented in March 2014 by the Adventia European College of Aeronautics.

Phase 2.) Conduct a case study with six instrument flight rules-certified (IFR) pilots in order to test the utility of Google Glass in the cockpit during instrument meteorological conditions (IMC). In order to obtain more accurate results, the ecological validity of the study was kept as high as possible in a full-motion flight simulator. Data related to heads-down time, pilot error and preference was recorded. Due to

the fact that the pilots in Mykityshyn, Kuchar, and Hansman's results preferred a decluttered display over the traditional format, qualitative data related to the decluttered instrument approach procedures displayed on Google Glass was also recorded. In addition, this thesis will take the affects of heads-up or heads-down location on qualitative and quantitative data into consideration.

The following quote from an unknown author (2002) underlines the core problem associated with this study:

> Aviation in itself is not inherently dangerous. But to an even greater degree than the sea, it is terribly unforgiving of any carelessness, incapacity or neglect [27].

4. METHODOLOGY

4.1 Definitions

The following list of terms define terminology within this study that may be unfamiliar to the reader. Most of these terms are present on the instrument approach plate that was used by the study's participants.

Approach Course - The direction of the approach course, based on a 0 - 359 degree heading.

ATIS - Automatic Terminal Information System. This provides a continuous broadcast of important airport information including wind speed, weather, active runways, and approaches. A pilot always listens to this information before starting or deciding on an approach.

Localizer - The localizer is an antenna located at the far end (departure end) of the runway and provides "course guidance throughout the descent path to the runway threshold from a distance of [at least]18 nautical miles" [5, p. 127]. The pilot inputs the localizer's frequency into the navigation panel, and frequently references the localizer indicator instrument that displays the aircraft's lateral deviation from the approach course.

Minimum Descent Altitude - No aircraft executing a non-precision approach (explained below) may operate below the minimum descent altitude, or MDA, unless the airport or airport environment can be clearly seen from this altitude [5, p. 168].

Missed Approach - A missed approach will be conducted if the pilot has descended to the MDA and does not establish visual contact with the airport or airport environment. The execution of a missed approach "occurs when your cockpit workload

is at a maximum," thus this procedure must be studied in the cruise phase of flight; before starting the instrument approach [5, p. 216].

Non-Directional Beacon - (NDB) An electronic navigation aid that provides navigation fixes or homing points using a low-frequency transmitter [5, p. 124]. The NDB used in this study was necessary for navigating the missed approach procedure.

Non-Precision Approach - A non-precision approach is an instrument approach where vertical guidance to the runway (with the exception of the localizer performance with vertical guidance procedure) is either lacking or not used. As such, the pilot keeps the aircraft at or above MDA until the airport environment is clearly visible. This approach contrasts to a *precision approach* in which the pilot may descend below the MDA with the help of vertical guidance from a glide-slope indicator found on the horizontal situation indicator (HSI). Pilots fly non-precision approaches for several reasons: if their aircraft is not equipped with a glide-slope indicator; if they are unfamiliar with the destination airport; if obstructions proximal to the airport (trees, towers, etc) make an approach closer to the ground impossible. If the pilot does not see the airport environment at the MDA before reaching the missed approach point (as indicated on the respective approach plate), he/she is required to execute a missed approach.

4.2 Phase 1 - Software Development

4.2.1 Overview

The first phase of this project required the development of a Google Glass software application called "glassware" that was installed natively on the device. The general idea for displaying approach information such as mandatory altitude changes, navigation radio frequencies, headings, etc. to the pilot via Google Glass entailed

constant querying of the plane's current GPS location and comparing this with instructions taken from a traditional IAP. More specifically, the aircraft's latitude, longitude, and altitude were constantly sent from the simulator to Google Glass and parsed by the glassware application. During the software development phase, the instructions from the traditional IAP were georeferenced and converted into code, allowing the application to use algorithms that displayed information relevant to the aircraft's location. The display of this information was carried out in the form of periodic push notifications. Attempting to satisfy Google's strict anti-distraction protocol, Google Glass' display remained OFF by default until new information became available, or if the pilot had committed an error. To do so, the screen would turn ON again with a chime sound, deliberately attempting to capture the pilot's attention. After remaining ON for an allotted reading time of 15-20 seconds, the screen turned OFF until, once again, newer information became available.

Due to time constraints, only one instrument procedure was selected for this study. The ILS or LOC 08R approach to George Bush Intercontinental (KIAH) was chosen due to the relatively large amount of required step-down fixes published on the procedure. During the IAP selection process, many of the IAPs reviewed had between one and three step-down fixes while the KIAH approach required six. Thus, KIAH was a more challenging approach, due to the fact that more step-down fixes increase the possibility of error, and keep the pilot's workload high throughout the entire procedure. Each step-down fix was located at a real-world geographic location. Thus, as the aircraft flew towards one of these locations, instructions related to that position on the IAP would be displayed on screen. When the aircraft intercepted this waypoint, the glassware application would change the information displayed to reflect the instructions at the next upcoming waypoint.

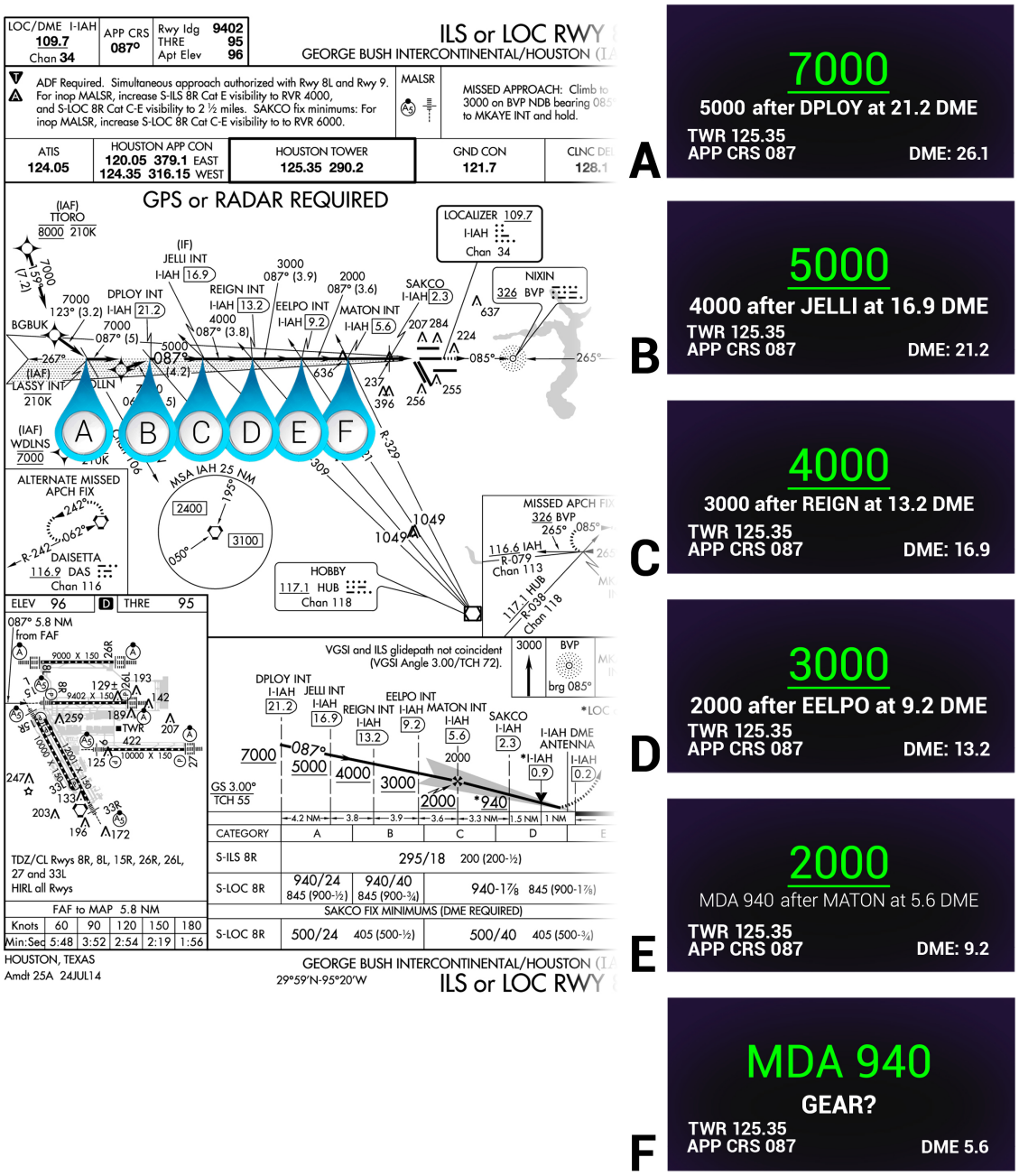

Figure 4.1: Each figure on the right represents what was presented on Glass' display based on the aircraft's position in relation to the instrument approach plate (left).

4.2.2 Breakdown of the Glassware Application

Taking a closer look at the glassware application, figure 4.1 presents "screenshots" of Google Glass' display, representing what the pilots saw at different locations along the approach procedure. In the upper right of figure 4.2, a box serves as a representation of what a pilot would see while flying with Google Glass. The green 7000 represents the step-down altitude of the upcoming waypoint, when compared to the approach chart, that the aircraft needs to be at or above. The color green represents that the aircraft's actual altitude is indeed at or above this number. The line of text beneath the 7000 represents the respective step-down altitude at the waypoint *after* the upcoming waypoint. While flying in instrument meteorological conditions (IMC), it is always important to remain one step ahead of your aircraft, and this line accomplishes that by providing the information of the second closest waypoint in advance. The numbers on the bottom left of the display represent course direction and frequency information, while the lower right provides a real time indication of the aircraft's distance from the airport.

If the aircraft were to descend below the minimums of the upcoming step-down waypoint, the display would turn on and an alarm chime would sound. Depending on the severity of the violation, the colors on the display would be either yellow or red, as depicted in figure 4.3. Yellow was shown when the aircraft's altitude exceeded 100' - 200' below minimums, symbolizing a mild warning for the pilot to stop descending. The screen would turn red when the aircraft descended 1000' or more below minimums and symbolized that the pilot should ascend immediately. This margin of error decreased, of course, as the minimum required altitudes became lower to the ground elevation.

Figure 4.2: A representation (upper right) of what a pilot would see while flying with Google Glass.

Figure 4.3: Display colors. Green = good; Yellow = warning to stop descending and fix altitude; Red = ascend immediately to avoid danger.

4.2.3 Development Lifecycle

Attempting to create a robust, accurate, and relevant prototype, the glassware was developed using an accelerated spiral software development lifecycle. The spiral model involves the end-user in each iteration of the software, which reduces the chances of misunderstandings and increases the accuracy of the software's content [9]. Beginning with a storyboard, working prototypes were iteratively built using Google Glass and a MacBook Pro running XPlane Flight Simulator. The details of the connection between Google Glass and XPlane are discussed later in the chapter. Each iteration of the software was tested and evaluated by two active duty Air Force Strategic Airlift pilots, as well as a retired commercial airline pilot.

4.2.4 Technology

The glassware was written in the Java programming language and deployed onto Glass using the Android Studio integrated development environment. The software adheres in part to the "object oriented" programming paradigm, in which objects, like "airports," are defined and generalized into classes, such as an "Airport" class, in order to perform reusable actions with multiple instances of the object. This study, however, utilized only one airport instance: KIAH. As such, the "Airport" class contained fields for KIAH's tower, ground, NDB, localizer, and ATIS frequencies, as well as an approach course field that was populated with information referenced from the ILS or LOC 08R IAP. Normally, such information should be stored and accessed from a database, allowing every approach chart in existence to be used. However, for this study, the "Airport" class took the place of a database which would have been more time consuming to implement. Due to the fact that the simulator study in phase two involved only one airport and one approach procedure, the amount of data that needed to be stored was limited, thus the "Airport" class was an acceptable

solution. Memory was allocated for the class and the fields were populated using the line of code below:

```
Airport KIAH = new Airport (29.993479, -95.360698, 109.7, 124.05, 125.35,
    326, 87, 121.7);
```

The first two numbers are the latitude and longitude of KIAH's localizer, respectively. The third number (109.7) represents the localizer frequency. The fourth and fifth numbers (124.05, 125.35) are the ATIS and tower frequencies. The sixth number (326) is the frequency of the NDB needed for the missed approach procedure. The seventh and eighth numbers (87, 121.7) are the approach course and ground control frequency numbers.

4.2.5 Connectivity

The glassware could receive GPS data in three different ways. For iterative development and testing with the MacBook Pro, the MacBook and Google Glass were connected to the same WiFi network, as shown in figure 4.4. Simulated altitude, latitude, and longitude data from XPlane on the MacBook were sent from port 49000 to the network via user datagram protocol. UDP allows computers on a shared IP network to send messages, also known as "datagrams" to other computers on the network, such as Google Glass. A "XPlane" class within the glassware application contained methods to receive and parse this data.

For usage in a real airplane, the glassware could receive GPS information via Google's MyGlass application installed on an Android smartphone. The MyGlass application tethers Glass to the smartphone over a wireless Bluetooth connection. Once Google Glass is tethered to the smartphone, Glass can receive calls, messages,

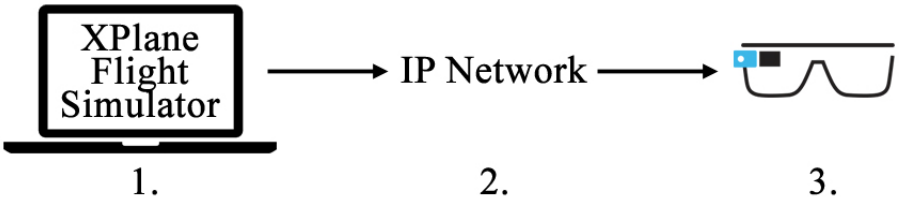

Figure 4.4: Packets containing GPS data were sent from the computer to Glass via UDP connection.

and even surf the internet using the phone's cellular connection. Due to the fact that Google Glass does not have an integrated GPS chip, the MyGlass application also allows Glass to receive location data from the phone's onboard GPS device. When creating software for Google Glass, one can access the smartphone's GPS data via the Android application programming interface (API) with the **LocationManager** and **LocationProvider** classes. This method was used several times in a real aircraft to field test the software with subject matter experts. Figure 4.5 demonstrates this connection.

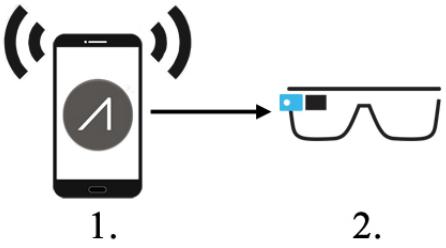

Figure 4.5: An Android smartphone with location service enabled sends GPS data over Bluetooth to Glass using the MyGlass application.

For the actual simulator study, Google Glass was connected to a full-size, full-motion flight simulator made by Redbird Flight Simulations of Austin, Texas. The

simulator used Microsoft Flight Simulator X, which could output the simulated latitude, longitude, and altitude data to a Bad Elf handheld GPS receiver using a proprietary USB device from Redbird called Cygnus. In order to send this data to the glassware, the Bad Elf was paired via Bluetooth with an Android smartphone that had mock locations enabled through the developer settings. Enabling mock locations overrode the phone's onboard GPS chip, forcing it to only consider the location information sent from the Bad Elf. Google Glass was then tethered via Bluetooth to the smartphone using the aforementioned MyGlass application, and a connection from the simulator to Glass was established. Figure 4.6 demonstrates the flow of data from the FMX simulator to the study's custom software on Google Glass.

Figure 4.6: Data flow from the Redbird FMX flight simulator (1.) to the Bad Elf GPS device (2.) to an Android smartphone (3.) and finally to Google Glass (4.).

4.3 Phase 2 - Flight Simulator Experimental Study

Google Glass with custom glassware application and a tablet displaying an IAP were tested in Redbird FMX flight simulators at Solo Flight Training in Houston, Texas and Redbird Flight Simulations in Austin, Texas. Figure 4.7 displays what the simulator looked like inside of the cabin. Quantitative and qualitative results

gathered from the usage of Glass and the tablet were compared to assess the utility of the device in the cockpit. Information from air traffic control was not provided; air traffic control can act as a "safety net" for pilots, reminding them of possible altitude violations or course deviations. In order to shift the responsibility of avoiding controlled flight into terrain onto the display devices, air traffic control was eliminated. Therefore, the participants were instructed to fly a complete non-precision approach as instructed on the tablet IAP or glassware application. A non-precision approach was chosen due to the fact that these approaches have a five-fold greater risk of controlled flight into terrain than a precision approach [3, p. 46].

Figure 4.7: In the cockpit of a Redbird FMX flight simulator, the pilot is using Google Glass and the custom-made application.

4.3.1 Research Participants

Six IFR-certified pilots participated in the study and were voluntarily recruited from the Texas A&M Flying Club of College Station, Texas, Solo Flight Training in Houston, and Redbird Flight Simulations in Austin. All participants were proficient IFR pilots and were comfortable flying in IMC. Two of the pilots were certified instrument flight instructors and one was a retired Army combat helicopter pilot. The others were experienced GA pilots. As expected, all participants were skilled tablet IAP users, yet all were unfamiliar with Google Glass. All participants were male, between the ages of 25 and 55, with a mean age of 45.

4.3.2 Flight Simulator Apparatus

The Redbird FMX flight simulator offered a full-sized airplane cockpit with 220-degree wrap-around monitors, realistic controls and avionics, aircraft seats, and a full-motion platform capable of pitch, yaw, and roll. The simulator's instrument panel could be customized with interchangeable avionics to reproduce an accurate cockpit of many different aircraft. As such, the simulator was set up as a Piper Arrow, which is a single engine aircraft commonly used as a flight trainer. Steam gauges (figure 4.8) were chosen as the avionics of choice instead of the more modern glass panel display system. Steam gauges are round instruments that display heading, altitude, airspeed, attitude, and rate of turn and climb, and are usually grouped together on the instrument panel in an arrangement known as the "six pack" or "basic-t." Historically, "steam gauges have been the norm, and instructional techniques have been optimised for steam gauges," and all of the participants were familiar with them [17, p. 62].

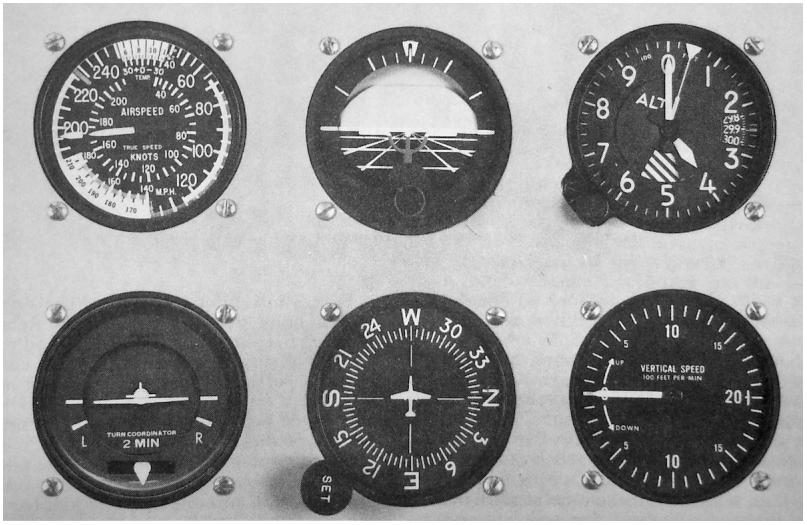

Figure 4.8: The steam gauge "six pack," similar to the instrumentation used in the experiment.

4.3.3 Experimental Design

The utility of Google Glass in the cockpit was tested and compared to that of an electronic tablet-based IAP as a *between participants* experiment. Three conditions were designed that involved the usage of these devices during two non-precision instrument approaches. In the first condition, pilots #1 and #4 flew both approaches using *only* the tablet-based IAP. This condition established the experimental control, due to the fact that tablet-based IAPs are currently quite popular among GA pilots. The second condition had pilots #2 and #5 fly the approaches *only* with Google Glass. Finally, the third condition required pilots #3 and #6 to fly with *both* Google Glass and the tablet-based IAP. Every pilot, regardless of device used, flew a simple practice approach with their respective device(s). A visual representation of this can be observed in table 4.1.

A *between-participants* experiment was chosen in favor of a *within-participants* experiment for several reasons. Firstly, each scenario occupied roughly 30 minutes of

Pilot #	Practice	Flight 1	Flight 2	Condition
Pilot 1	P	Scenario A	Scenario B*	Tablet
Pilot 2	P	Scenario B	Scenario A*	Glass
Pilot 3	P	Scenario A	Scenario B*	Both
Pilot 4	P	Scenario B	Scenario A*	Tablet
Pilot 5	P	Scenario A	Scenario B*	Glass
Pilot 6	P	-	Scenario A*	Both

Table 4.1: Between-participants flight simulator study. P = Practice Scenario; * = Scenario + Missed Approach. Note the missing scenario for pilot #6. This pilot was mistakenly instructed to complete the practice scenario using the wrong device, thus results from the first flight were disregarded.

the participants' time. If each pilot flew all three conditions, as a *within-participants* experiment requires, the study would have taken one and a half hours, plus an additional ten-15 minutes for the practice scenario for each pilot. This duration was neither in the time budget of the study nor of the participants. In addition, a longer experiment runs the risk of fatiguing the participants, where "performance in the later trials is worse than those on the earlier ones because the participant is either tired or bored" [23, p. 56]. As such, the *between-participants* method allowed the study to be limited to one hour for each participant. Secondly, a *between-participants* study may have avoided noticeable "carry over" effects that a *within-participants* study may have provoked as the participant switched from one device to another. For example, a study where the participant would have used Google Glass first and the tablet IAP second might have produced different results than using the tablet first and Glass second. Indeed, everyone has different levels of confidence and prior knowledge with new technologies, thus the difference in performance due to the order of devices used may not have been attributable to the devices themselves [23, p. 52].

As stated before, every pilot flew a practice scenario "P" before beginning with the experimental scenarios. The practice scenario's role was to familiarize the pilot

with the simulator and their assigned IAP device. The practice scenario did not involve the same approach flown in the experimental scenarios.

The overall backstory behind each experimental scenario was as follows: the pilot was cleared for a different approach than what they were expecting to fly during the study. However, as they neared the airport, the wind and weather conditions at the airport had suddenly changed, thus they were told to use another approach (RWY 08R) instead. According to the subject matter experts consulted during the glassware's development, such a situation is uncommon in real life. However, these situations can indeed occur, and thus require a pilot to quickly learn and adapt to a new approach procedure. Due to the aircraft's proximity to the airfield, the pilot's workload increases as he/she concentrates on briefing themselves of the new procedure. Both experimental scenarios used the same ILS or LOC RWY 08R procedure for the non-precision approach into KIAH. In the attempt to prevent the pilot from memorizing the procedure after the first flight, thus avoiding the "learning effect," each scenario started at a different geographic location and altitude. In addition, the RWY 08R procedure was specifically chosen for this study due to the large amount of step-down fixes published on the chart.

Scenario A began with the aircraft in dense clouds, 30 nautical miles northwest of the airport at 8000 feet mean sea level (msl). The thick cloud layer extended down to 1100' msl. The weather below 1100' msl was rainy with a visibility of two statute miles. Once the aircraft intercepted the localizer path at the LASSY initial approach fix, gusting crosswinds from the north at 15-20 knots made it more challenging for the pilot to remain on course, thus increasing the amount of effort expended to simply fly the aircraft. Figure 4.9 provides a top-down view of scenario A's flightpath.

Scenario B began similarly with the aircraft in dense clouds, 30 nautical miles southwest of the airport at 6000' msl. The cloud layer and breakout altitude of

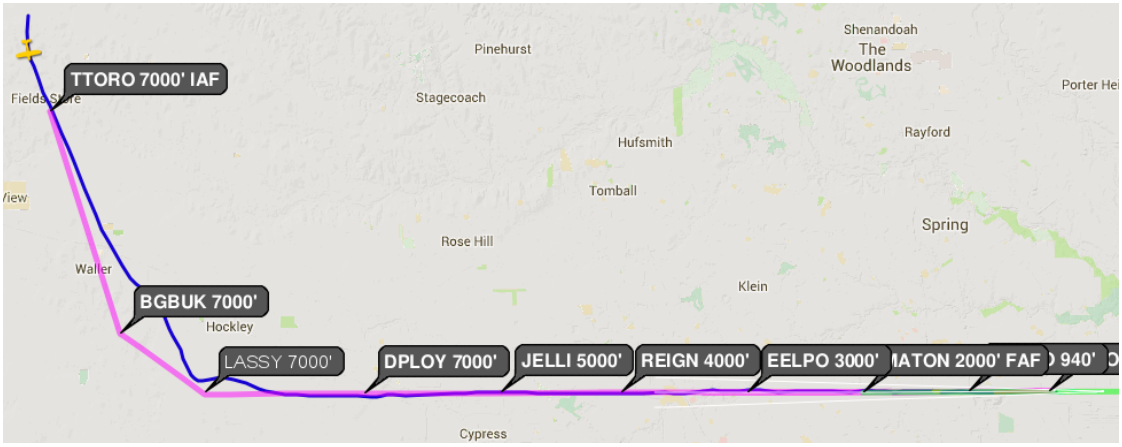

Figure 4.9: Scenario A. The yellow aircraft symbol represents the starting position of the aircraft.

1100' remained the same. The wind speed and gusts also remained the same, but blew from the opposite direction. The 6000' starting altitude was 1000' below the minimums of the first waypoint, LASSY, thus the amount of time needed by the pilot to notice and correct this issue was recorded. Figure 4.10 provides a top-down view of scenario B's flightpath.

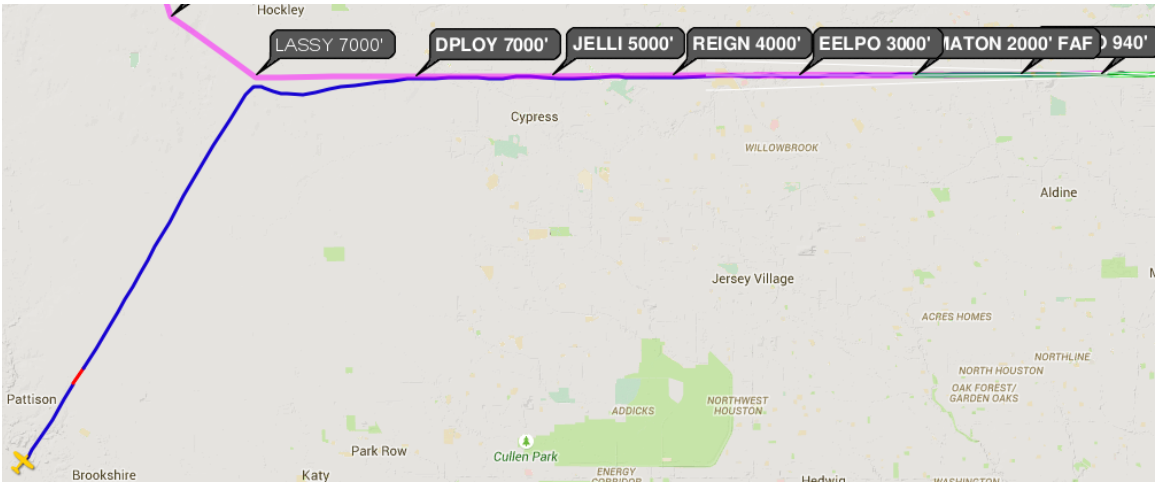

Figure 4.10: Scenario B.

Table 4.1 displays a "*" behind the second scenario of every participant. This denotes a required missed approach at the end of the respective scenario. Unbeknownst to the pilot, weather conditions at the airport were set to be inconducive to a safe non-precision approach, thus the pilot needed to execute a missed approach. Instructions for the missed approach procedure were present on both the tablet IAP and Google Glass application. Assigning the missed approach to the second flight of every participant effectively created an experimental control for this procedure. If in fact a "learning effect" was present from the first to second flight, every pilot would have at least initiated the challenging missed approach with the same "experience" level.

4.3.4 Procedure

The six pilots were tested over a period of three non-consecutive days. Originally, the study was planned for two consecutive Fridays, but a last minute cancellation of one of the participants precipitated the need for a third day. Upon arrival at the testing center, the participants were introduced to the experiment and their assigned IAP device(s). After signing a consent form, they completed a short practice scenario in the simulator, followed by either scenario A or B. For the pilots using Google Glass, the practice scenario ensured that they could read Glass' screen without any difficulty. The pilots using the tablet positioned the device either on their lap or on a tablet holder, located to the lower left of the instrument panel. The experimenter was present in the cockpit at all times during the study, but would only answer technical questions that otherwise may have hindered the pilot from completing the scenario. After the first flight, the participants were given a short questionnaire regarding their perceived level of workload and proficiency while flying the approach with their assigned IAP device(s). During this time, they were also given the chance to relax

for at least ten minutes. Afterwards, the pilots completed the second flight scenario. At the conclusion of the study, the participants were similarly questioned about the second flight, and were given a final questionnaire regarding their opinion of the IAP device and, if Google Glass was used, their suggestions for overall improvement of the technology for implementation in general aviation.

4.3.5 Measures

Video recordings of the pilots documented the amount of heads-down times (glances) the pilot made in order to read information from the tablet IAP. In addition, the duration of the glance was also recorded. These recordings were obviously more important for the analysis of pilots #1, #3, #4, and #6, due to the fact that pilots #2 and #5 did not use a tablet IAP and thus, did not need to look down. Because the pilots were asked to use the think-aloud protocol as they navigated, audio recordings documented their decisions made as they progressed along the approach procedure. Flight performance data such as minimum altitude observance and navigational accuracy was recorded by a web application called Cloud Ahoy, which was installed on the Android smartphone receiving GPS data from the simulator. Using this data, Cloud Ahoy recorded every step of the flight onto a debriefing website. An example of this can be seen in the following figure:

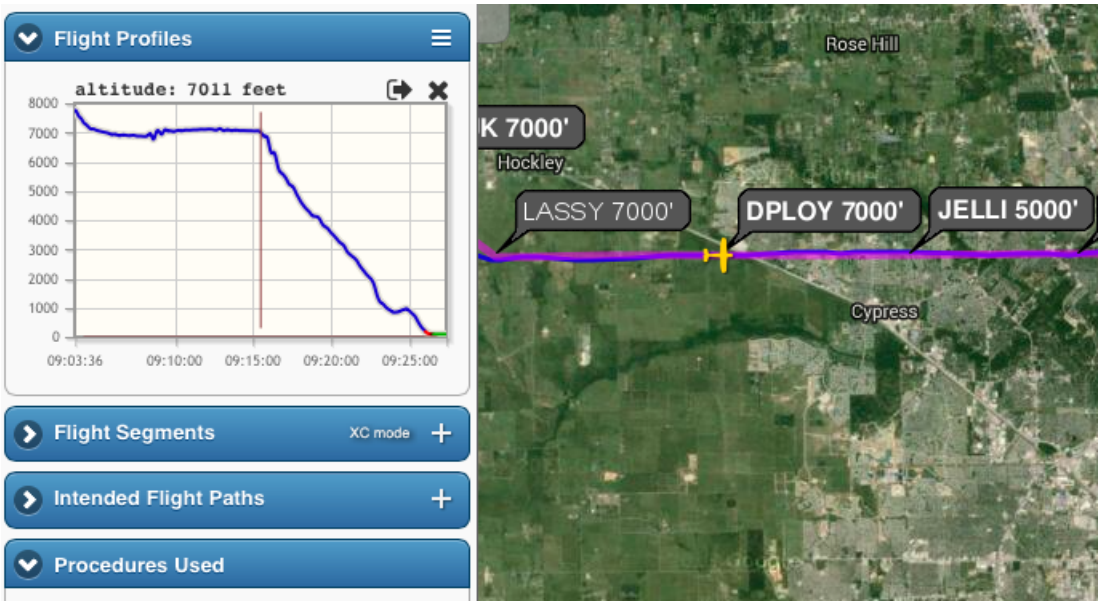

Figure 4.11: Flight debriefing in Cloud Ahoy. The yellow aircraft symbol represents the current position of the airplane along the flight path. The bubbles labeled "DPLOY," "JELLI," etc. represent the step-down fixes on an IAP and their respective geographic locations.

5. RESULTS

5.1 Summary of Hypothesis

It was hypothesized that the heads-up nature of Google Glass could supplement current paper or tablet instrument approach-display methods. More specifically, it was expected that Google Glass could reduce pilot heads-down time and distraction when used in conjunction with the tablet IAP. In addition, based on previous NTSB reports of controlled flight into terrain by pilots misinterpreting their approach charts, it was predicted that the pilots using Google Glass would make less altitude-related mistakes when compared to the usage of a tablet IAP by itself.

Of the six participants, four were completely confident that they had accurately and safely flown both experimental scenarios with their respective IAP device(s). In addition, only one mentioned that he had been reading approach information for too long during both scenarios. On average, it took the participants around 25 minutes to complete each scenario. Despite the relative difficulty of the scenarios, there were no accidents due to weather, unrecoverable pilot error, or controlled flight into terrain.

5.2 Heads-Down Time

5.2.1 Number of Glances

The number of glances, which represent the number of times the pilots looked away from cockpit instrumentation in order to read instrument approach instructions (with lower numbers being better), was affected by the device(s) used. Results from the Google Glass-only condition have been omitted from this section due to the fact that the users never needed to look down in order to read navigational instructions. For the other conditions, each glance was tallied regardless of the duration. The

pilots in the tablet-only condition looked down an average of 59 times, while those in the tablet + Google Glass condition looked an average of 22 times. However, in chapter 4 it was stated that one of the four scenarios in the tablet + Glass condition was omitted due to a mistake made during the practice scenario preceding it. As a result, one of the pilots, pilot #6, effectively had only one valid scenario count towards the final results. Figure 5.1 provides a graphical representation of these results.

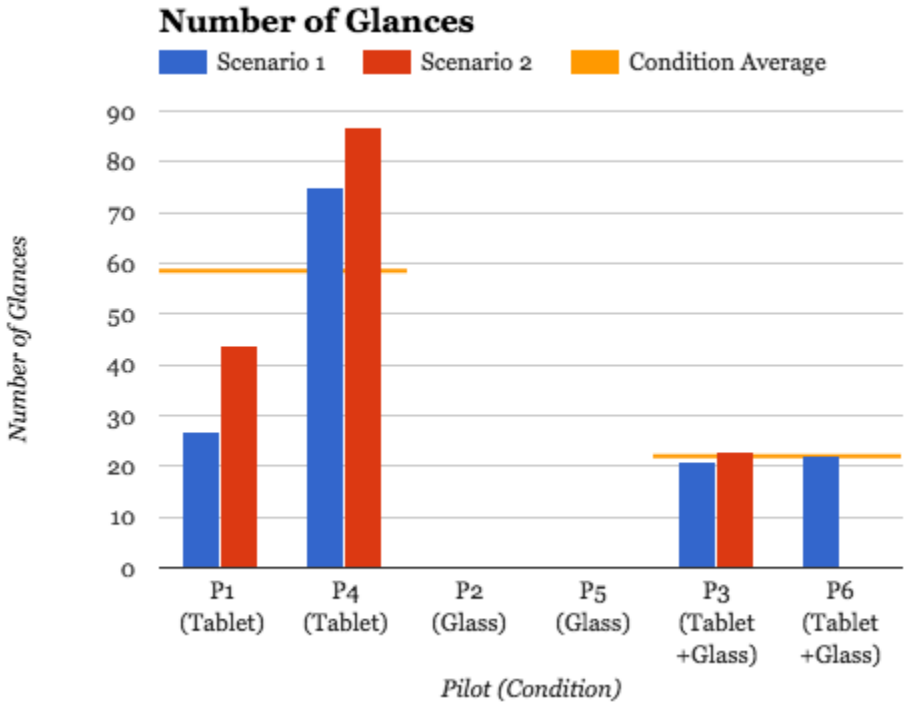

Figure 5.1: Number of glances

5.2.2 Average Duration Per Glance

The average duration per glance was slightly shorter within the tablet-only condition than the tablet + Google Glass condition. This duration was calculated from

when the pilot began looking at the tablet IAP until he resumed his scan of the instrument panel, and is graphically depicted in figure 5.2. Indeed, the pilots in the tablet-only condition spent an average of 2.11 seconds per heads-down occurrence, while the tablet + Google Glass group spent an average of 2.54 seconds per occurrence.

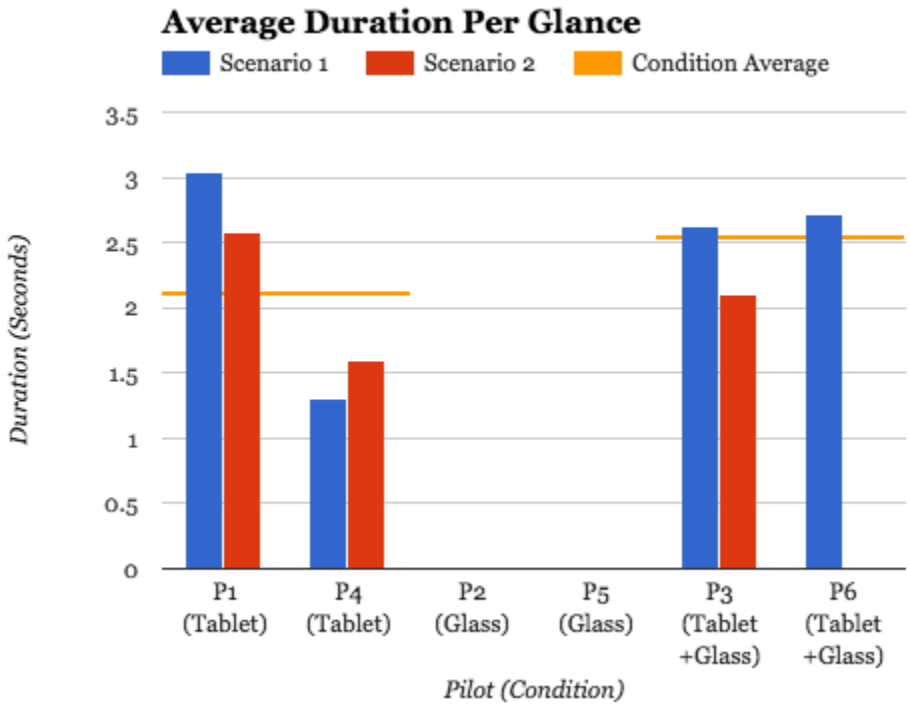

Figure 5.2: Average duration per glance

5.2.3 Average Heads-Down Time

The average heads-down time represents the average amount of time that the pilots in the respective condition spent heads-down, reading instructions from the tablet IAP. While the tablet-only condition produced a shorter average heads-down time per glance, as seen in figure 5.3, the the overall average of the heads-down time

for this condition was still 100% longer than the tablet + Google Glass condition. This was due to the higher average number of glances, as shown in figure 5.1.

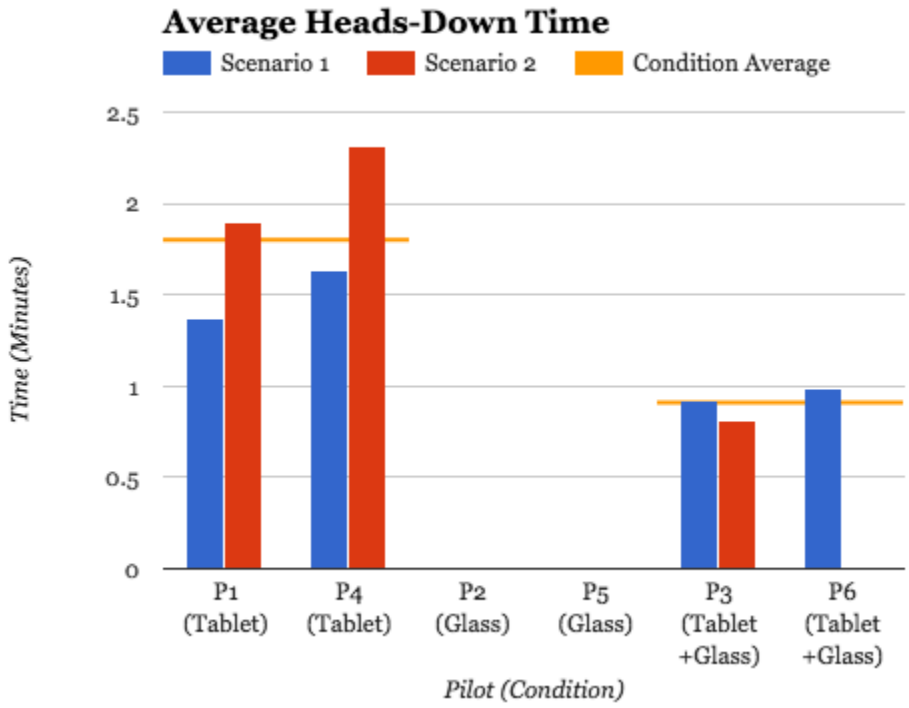

Figure 5.3: Average heads-down time

5.2.4 Number of Glances Per Minute

The number of glances per minute was calculated by breaking the approach course into seven segments. Each segment represented the duration of flight between each step-down fix. A 2 x 2 x 7 mixed-model analysis of variance was performed using two display conditions (tablet and tablet + glass), two scenarios (A and B), and the aforementioned seven time segments. Table 5.1 provides a visual representation of this analysis. The effects of the displays on glances per minute were treated as fixed factors and between-subjects; each display represented independent "fixed" variables,

and each pilot used only one type of display. The effects of the scenarios were treated similarly as fixed factors, yet labeled as within-subjects, for the scenarios were flown by every participant. Finally, the effects of time were also considered as fixed factors and within-subjects. The mixed-model ANOVA showed that there was a large mean difference between the two display conditions, as well as a large variability across the seven time segments. However, due to the small sample size of participants and large amount of the variability of means, the p-values never reached conventional levels of statistical importance ($\alpha = 0.05$) for either condition ($p > \alpha$). Despite the results, the pattern of means for glances per minute and display does suggest that the usage of Google Glass might lead to fewer downward glances throughout the approach. This is represented in figure 5.4, in which the tablet-only condition returned an expected mean of 2.37, while the tablet + Google Glass condition returned .8119.

Source	Deg. of Freedom	Sums of Squares	Mean Square	F-ratio	P-value
Intercept	1	142.734	142.734	47.668	0.0203
Display	1	25.4904	25.4904	8.5129	0.1001
Pilot	2	5.98865	2.99433	2.0392	0.1441
Scenario	1	0.08868	0.08868	0.0604	0.8072
Time	6	18.4128	3.06881	2.0899	0.0772
Error	38	55.7992	1.46840	-	-
Total	48	109.118	-	-	-

Table 5.1: Mixed-model ANOVA. 2 display x 2 scenario x 7 time.

Interestingly, it was also observed with the bi-modal distribution in figure 5.5 that time periods #1 and #4 (Start-DPLOY and REIGN-EELPO) produced the largest amount of glances per minute. More glances per minute during the beginning of the flight made sense, due to the fact that the pilots were briefing themselves of the approach information. However, the peak of expected means during the REIGN-

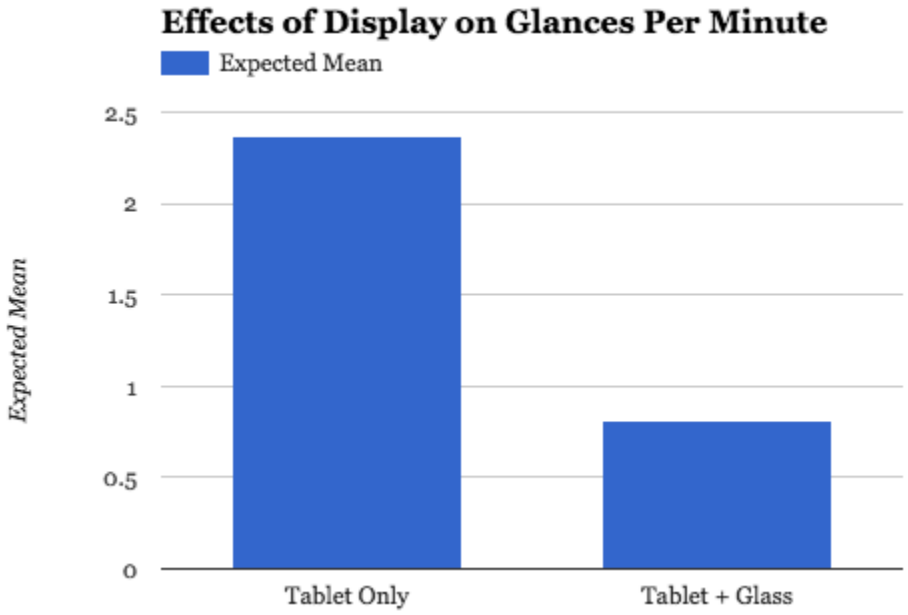

Figure 5.4: Effects of display on glances per minute

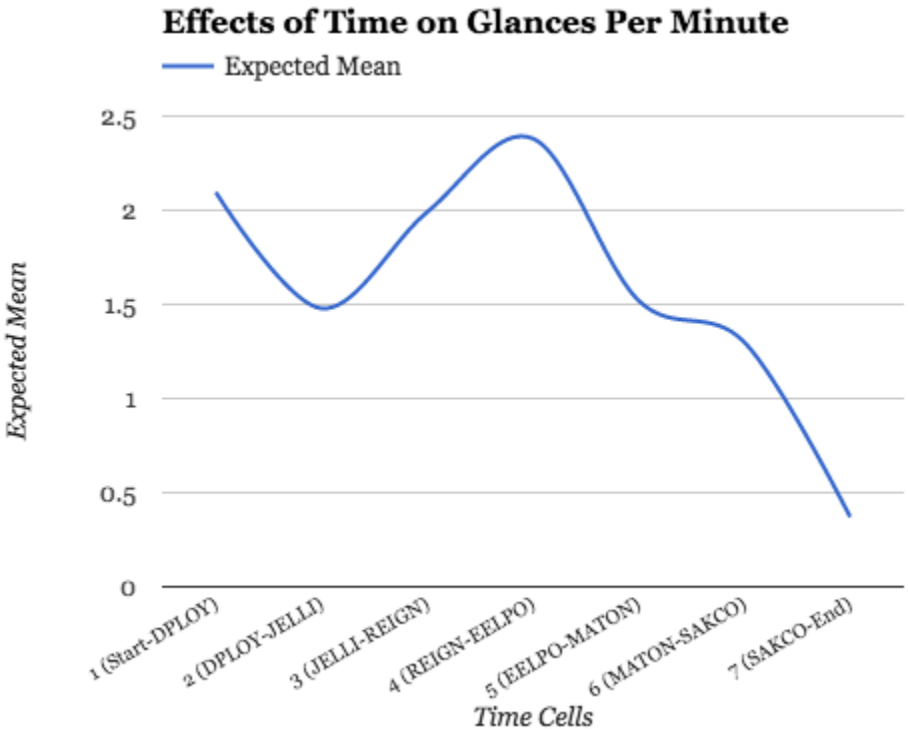

Figure 5.5: Effects of time on glances per minute. The pattern of means suggests that there were two times during the scenarios that required the most glances per minute.

EELPO time segment (#4) was unexpected. This could have resulted from the pilots preparing themselves for reaching the final approach fix located in the following time segment #5 (MATON). The final approach fix in this study was 5.6 nautical miles from the airport and was represented on the IAP by a small cross on the side-view of the approach course. According to the subject-matter experts consulted during the software development phase, it is important to have gear, flaps, and airspeed set for landing when reaching this point. In order to determine the actual reason behind the peak of expected means in period #4, a study with more participants would be needed.

5.3 Flight Performance

5.3.1 Altitude Violations Per Scenario

The number of instances when the aircraft descended below minimum altitude requirements was recorded from all conditions. At higher altitudes (7000' - 4000'), a deviation less than 200' below minimums was not recorded, while at lower altitudes (3000' - 2000'), a deviation of less than 100' was similarly ignored. These "buffers" represented margins of error suggested by the subject matter experts during the glassware's iterative development phase. At the minimum descent altitude, however, no deviations were tolerated. Despite the fact that over half of the participants reported being confident that they had correctly and accurately flown each scenario, a closer look at the altitude violations revealed that everyone, regardless of condition, had descended below required minimums at least once throughout the entire experiment. A total of four violations occurred in the tablet-only condition, nine in the Google Glass-only condition, and four in the tablet + Google Glass condition. As seen in figure 5.6, the tablet-only condition committed an average of one violation per scenario; the Glass-only condition committed two, and the tablet + Glass con-

dition committed 1.3. Due to the fact that one of the four scenarios in the tablet + Glass condition was omitted from the final results, more violations in this condition were possible, thus the reported number of four serves as the *least* amount of possible violations in this condition.

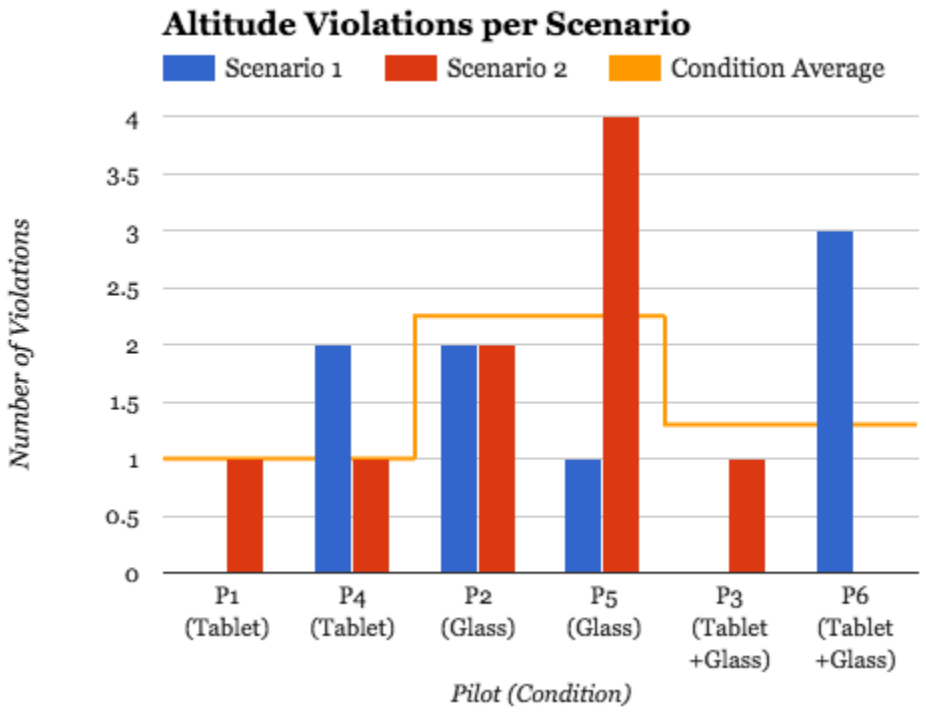

Figure 5.6: Altitude violations per scenario

5.3.2 Average Reaction Time Per Altitude Violation

The average reaction time, as shown in figure 5.7, represents the average amount of time per violation that it took the pilots to realize their aircraft's incorrect altitude and subsequently ascend within or above the aforementioned "buffers." The slowest reaction time came from the tablet-only condition at 1 minute and 33 seconds per violation. A faster reaction time was recorded from the Google Glass-only condition

at 23 seconds. The quickest reaction time came from the tablet + Google Glass condition at 19 seconds. Once again, due to the omitted scenario in the tablet + Google Glass condition, this time serves as an accurate prediction of the actual average reaction time in this condition. Despite the slight differences between the reaction times of the Google Glass-only and tablet + Google Glass conditions, it is believed that the devices themselves had little affect on these results. Instead, the differences can more likely be attributed to the performance of the pilots themselves.

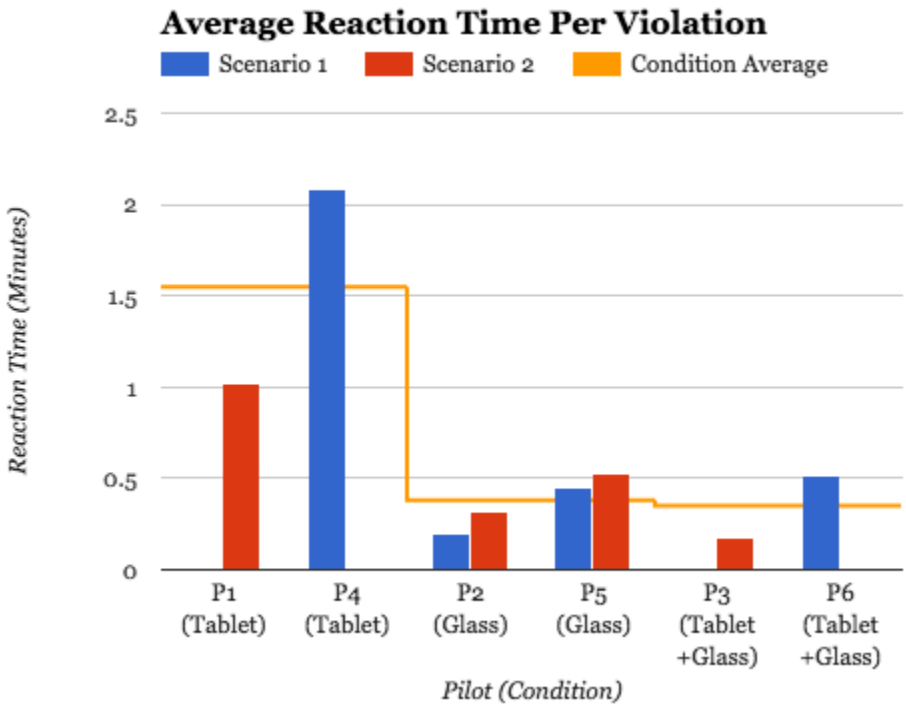

Figure 5.7: Average reaction time per violation

5.3.3 Ascent Time to 7000' in Scenario B

Ascent time to 7000' in scenario B represents the amount of time it took the participants to ascend from the starting altitude of 6000' to the published required

altitude of 7000', before arriving at the first waypoint, LASSY. Both pilots in the tablet-only condition never ascended to 7000' feet. In the Google Glass-only condition, pilot #2 took 46 seconds to ascend to 7000', while pilot #5 took 2 minutes and 26 seconds. In the tablet + Google Glass condition, only one measurement was recorded, for the omitted scenario of pilot #6 contained the results in question. As such, pilot #3 took 2 minutes and 18 seconds to ascend. It is important to note that the accuracy of this test, in aviation terms, could easily be discounted. Indeed, pilots typically descend from an enroute structure to a waypoint or step-down fix on an approach chart; they usually do not expect to ascend to begin an approach. However, this test arguably served as a good measure for the real-time utility of Google Glass or similar head-mounted devices that provide instructions or reminders in the form of push notifications. In addition, it can be argued that many air accidents are caused because the pilot was not "expecting" to have to do something, sometimes this has been as simple as *flying at a higher altitude*. Every pilot was instructed to fly the approach exactly as instructed on the IAP, thus any neglect thereof or lack of double checking with the experimenter could be argued as a mistake. In addition, consider a scenario where a pilot has briefed him/herself of a specific approach, only to be told shortly before arrival that the expected runway has been closed. This would precipitate the need for quickly selecting another runway, and thus, an alternate approach procedure. Should this occur in real life, air traffic control will more often than not vector the pilot to a new position and altitude, however this may not always be the case. Thus, in order to correctly fly the new procedure, it is possible that the pilot would need to ascend again to a new starting altitude.

5.4 Summary of Results

Despite the omitted scenario in the tablet + Glass condition, one could conclude that a head-mounted device like Google Glass used in conjunction with a tablet IAP reduces the amount of times a pilot must look down in order to read an IAP while flying in IMC. Although the average heads-down time per occurrence from the tablet-only condition was shorter, and no differences in the amount of violations between the tablet-only and tablet + Glass conditions were recorded, the tablet-only condition produced the longest reaction times when the pilot did indeed violate minimum altitude requirements. For example, one of the pilots in the tablet-only condition misread the IAP and began descending prematurely. With no altitude warning provided by the tablet IAP, the pilot finally realized his mistake after 1 minute and 17 seconds into the descent; nearly 1000' feet below the current required altitude. According to Dismukes, et. al, "controlled flight into terrain is one of the most common types of fatal [...] accidents," and a frequent scenario thereof "is crashing short of the runway while executing a non-precision approach requiring stepdown fixes" [3, p. 46]. As such, this misinterpretation of the approach procedure represents a real danger in aviation, especially when mountains, power lines, trees, or tall buildings can greatly reduce the survivable margin of error. In addition, both pilots in the tablet-only condition did not realize the need to ascend from 6000' to 7000' in the beginning of scenario B. Every other participant who flew with Google Glass ascended quickly after starting this scenario.

The participants in the Google Glass-only condition committed more altitude violations than the other two conditions combined. Although the reaction time with Google Glass was shorter and thus less "serious" violations were committed, one of the participants in this condition was doubtful about his flight performance and

adherence to the approach procedure. Upon further questioning, this concern was partially due to the glassware's linear presentation of information. Both pilots who flew in this condition stated that they wanted to see all of the IAP information before starting the approach, just like they could when flying with a traditional tablet or paper IAP. This makes sense, due to the fact that pilots typically attempt to memorize as much of the information on the IAP as possible before starting the procedure. After completing the study, both participants mentioned that Google Glass was not a robust replacement for the tablet IAP, but would offer an excellent form of redundancy for pilots when used in conjunction with an electronic or paper IAP.

Due to these results, one may argue that Google Glass represents another "slice of cheese" in James Reason's swiss cheese model. The swiss cheese model suggests that "barriers in the system (the slices themselves) are intended to prevent errors that result in [...] adverse events" [24]. In this study, these barriers, or defenses, are the pilot's experience, cockpit instrumentation, and preparedness for the approach. According to Reason, multiple failures of the barriers "must be aligned for any adverse events to occur" [24]. Thus, the study's weather conditions, and the approach's difficulty and time sensitive nature were designed to exploit the flaws in each of these layers of "cheese." As such, Google Glass represents one more of these barriers, essentially forming a safety net that could prevent a common mode failure when the pilot, although experienced, has forgotten to cross-check cockpit instrumentation with the approach instructions due to high levels of workload from the last-minute nature or difficulty level of the approach. Air traffic control, although omitted from this study, forms another "slice," yet as as seen with the December 1996 Learjet crash, they can also fail to notify the pilot in enough time (or at all).

6. CONCLUSION AND FUTURE PLANS

Due to air accidents in the past involving prolonged heads-down time and the misinterpretation of traditional IAPs, this study aimed to develop a robust and unobtrusive software application for Google Glass that attempted to reduce pilot workload, distraction, and the danger of controlled flight into terrain for pilots approaching an airport in IMC. The current lack of scholarly publications related to this type of research offered the chance for the thesis to pioneer something that not many had tried before. Therefore, a case study with six IFR-certified pilots was conducted in a full-sized, full-motion flight simulator in order to compare the heads-down time, and number and duration of step-down altitude violations made by pilots using either an electronic IAP on a tablet, Google Glass with custom approach software, or a combination of the two while conducting a non-precision localizer approach. The experimental study, however, was not carried out perfectly. Due to a mistake made during a practice flight in the tablet + Google Glass condition, the results of the following experimental scenario were omitted from the paper. Unfortunately, this lead to an unbalanced experiment with incomplete results from the tablet + Google Glass condition. Despite the imbalance, the results have shown that the usage of Google Glass and a tablet-based IAP together help pilots fly a safer approach when compared to the usage of a tablet-based or paper IAP alone. More specifically, pilots using Google Glass during an approach had a quicker reaction time when they did indeed commit an altitude violation, and the total heads-down time was drastically reduced, allowing them to focus more on their scan of cockpit instrumentation.

6.1 Glassware Revisions

As expected, the qualitative data recorded from the two participants in the Glass-only condition provided valuable feedback that would make the development of a more precise iteration of the glassware application possible. For example, both pilots mentioned that they were missing the morse code identifiers that accompany localizer, non-directional beacon, or VHF Omnidirectional Range navigation frequencies found on the standard IAP. The morse code identifiers, as shown in figure 6.1, are simply morse code symbols that allow the pilot to confirm that they have input the correct frequency into their navigation radio. Indeed, after dialing in a frequency, the radio plays back the audible morse code beeps, allowing the pilot to compare them with the symbols printed on the IAP. In addition, the participants mentioned that it is generally a good idea to have at least the first two steps of the missed approach procedure memorized before crossing over the final approach fix. The final approach fix is located four to seven miles from the airport and represents when the aircraft may either intercept the glide slope (for a precision approach) or descend to the MDA (for a non-precision approach).

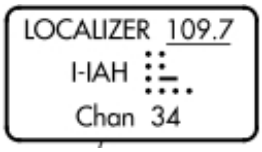

Figure 6.1: I-IAH localizer frequency with morse code identifier.

In addition, both pilots mentioned that they wanted to receive the approach course, minimum descent altitude, and localizer frequency as soon as they started the flight scenario. Indeed, the experimental version of the glassware displayed the

approach course and localizer frequency only when the aircraft flew within five miles of the first waypoint, LASSY; after roughly six minutes after the start of the flight. In addition, the minimum descent altitude was displayed towards the end of the approach when it became most relevant to the pilot. Despite multiple iterations with the subject matter experts during the glassware's development, the participants felt that these items were still displayed too late, thus contributing to their lack of confidence while flying the approach and trust in the glassware. The pilots flying in the tablet + Glass condition did not notice this shortcoming. If another study could be conducted with these revisions, perhaps the results obtained in the Glass-only condition would be more favorable to the usage of the technology as a stand-alone device.

6.2 Future Directions

Since the beginning of the study, Google Glass has become a "moot" point in the gadget world. It was expensive ($1500), imposing, and poorly marketed. In addition, most of the apps developed for Google Glass by the "Explorers" program were not innovative enough; they accomplished nothing more than what one could already do with a smartphone. However, the overarching and future-oriented goal of this study was not simply to focus on building content for Glass. Instead, the goal was to explore what could be done with novel HMD technology to improve the current and fallible instrument approach system. Should the HMD prove itself useful in a potentially dangerous and high-stress situation, then the methodology of displaying time-sensitive and important information on the device could be further applied to future projects in several ways; to forthcoming heads-up technology like Microsoft's HoloLens, or to the enrichment of current tablet electronic flight bags.

6.2.1 Tablet-Based Electronic Flight Bag

This study could support the enrichment of current commercial over-the-shelf EFBs that are becoming increasingly popular in general and commercial aviation. All of these applications offer a wide range of safety-oriented features, such as pop-up notifications when other air traffic is in the vicinity, when weather conditions have changed, or when runways at destination airports have closed. Thanks to the study, another similar feature could be added that resembles the glassware application's method of displaying simplified information and warnings based on real-time GPS location. A similar wearable solution has been released by Hilton Software, developers of the WingX Pro tablet EFB. They have developed a Pebble smartwatch version of their WingX software that provides vibration feedback based on certain pre-defined conditions. When tethered via Bluetooth to a tablet running the WingX tablet software, the watch vibrates when an instrument approach timer has expired, when passing over a step-down altitude waypoint, when violating a pre-selected minimum altitude, or when descending below 1000' of the destination's field elevation [13]. However, the altitude warning with the smartwatch is not automatic, leaving room for error should the wearer program in the wrong altitude. Figure 6.2 displays a wireframe of what this thesis-inspired information display feature could look like. Based on GPS location, small pop-up messages could highlight relevant information, reducing heads-down time and the amount of data that the pilot needs to remember.

6.2.2 Microsoft HoloLens

At the time of this writing, the upcoming HoloLens from Microsoft promises seamless integration of the digital and real worlds, allowing interactions with spatial representations of people, objects, and information. The device is a completely self-contained computer and, unlike Google Glass, does not rely on a connection to a

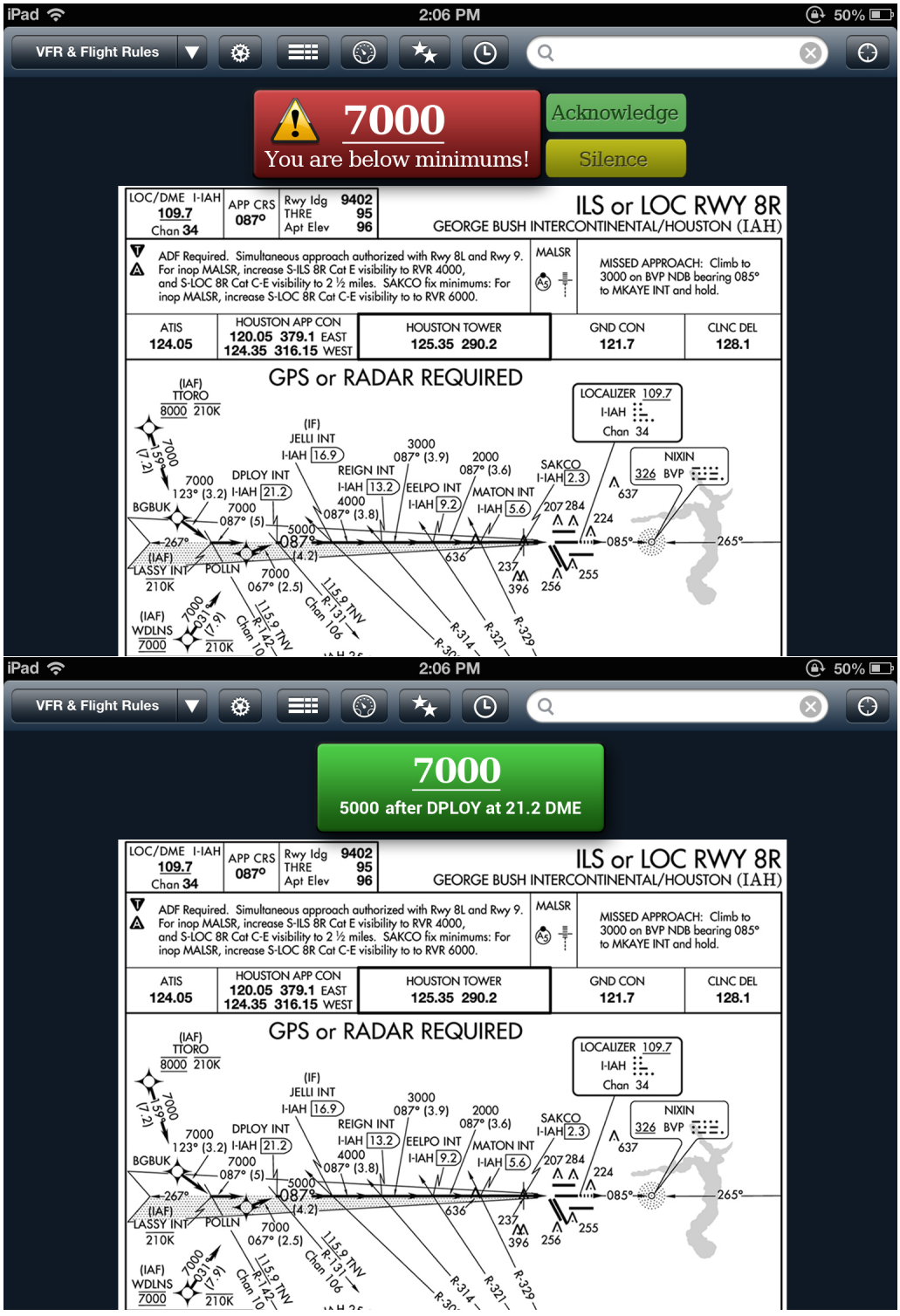

Figure 6.2: A hypothetical alert and notification represented on a tablet IAP.

smartphone or any other computer. Worn on the user's head, the HoloLens "uses a novel holographic display [...] that can trick the eye into perceiving 3-D objects," and "sensors in the headset allow the device to figure out how to present virtual objects so they fit in with the real world" [25]. Essentially, a HoloLens wearer can walk around a hologram, viewing and manipulating it from any angle. In addition, HoloLens uses cameras to track eye and hand movements, allowing the user to interact with information via hand gestures. Using the results from this thesis, an IAP application for the HoloLens could be developed that places approach information in the 3-D space of a cockpit. The pilot could pick and choose what information he/she needs from an IAP and set them "virtually" in front of the instrument panel with a simple hand movement. Should a piece of information become irrelevant, another swipe of the hand could knock it away from view. In addition, an alarm system, much like the one this thesis' glassware featured, should be built into the software. Indeed, the thesis showed that such an alarm can quickly remind pilots of errors, allowing for a quick response time in correcting the error. Imagine a scenario with the HoloLens, for example, where the pilot accidentally brushes important 3-D floating textual or graphical information out of his/her field of view. Should the pilot fail to realize their mistake, the application would put the information back in the original location with an audible or visual warning. Similarly, if the pilot forgets to place relevant information in front of the instrument panel, the software could do so automatically.

REFERENCES

[1] Michelle Bassanesi and Gary Tindall. Heads up, heads down: The ipad and its use in the general aviation cockpit. http://www.aviationplatform.com/attachments/article/62/Bassanesi_Ipad.pdf, November 2011.

[2] John W Croft. Finding focus; texas company readies 'augmented reality' for general aviation cockpits. *Aviation Week & Space Technology*, 175(35):47–48, October 2013.

[3] R Key Dismukes, Benjamin A Berman, and Loukia D Loukopoulos. *The limits of expertise: Rethinking pilot error and the causes of airline accidents*. Ashgate Publishing, Ltd., 2007.

[4] R Key Dismukes, Grant E Young, Robert L Sumwalt III, and Cynthia H Null. Cockpit interruptions and distractions: Effective management requires a careful balancing act. http://asrs.arc.nasa.gov/publications/directline/dl10_distract.htm, December 1998.

[5] Federal Aviation Administration. *Instrument Flying Handbook*. Number AC 61-27C. Department of Transportation, 1980.

[6] Fredric S Fitzsimmons. The electronic flight bag: A multi-function tool for the modern cockpit. Technical report, DTIC Document, 2002.

[7] Foreflight. Frequently asked questions. http://foreflight.com/support/georef#56.

[8] Foreflight Mobile. Ds annotations plate on map. http://www.foreflight.com/static/img/5.4/ds-Annotations-plate-on-map.png.

[9] Armando Fox and David Patterson. *Engineering Software as a Service*. Number 1.1.0. Strawberry Canyon LLC, July 2014.

[10] Garmin. Garmin pilot. `https://buy.garmin.com/en-US/US/on-the-go/apps/garmin-pilot-/prod115856.html`.

[11] Christopher J Hamblin. Usability of mobile devices in the cockpit, 2004.

[12] Keith L Hiatt, Clarence E Rash, Eric S Harris, and William H Gilberry. Apache aviator visual experiences with the ihadss helmet-mounted display in operation iraqi freedom. Technical report, DTIC Document, 2004.

[13] Hilton Software. Wearable wingx: Wingx pro7 for pebble. http://hiltonsoftware.com/pr/WingXPro7ForPebble-PressRelease.pdf, March 2014.

[14] Robert E Joslin. Human factors hazards of ipads in general aviation cockpits. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, volume 57, pages 56–60. SAGE Publications, 2013.

[15] Hua Li, Xin Zhang, Guangwei Shi, Hemeng Qu, Yanxiong Wu, and Jianping Zhang. Review and analysis of avionic helmet-mounted displays. *Optical Engineering*, 52(11):110901–110901, 2013.

[16] Kristen K Liggett, John M Reising, Thomas J Solz, and David C Hartsock. A comparison of military electronic approach plate formats. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, volume 40, pages 15–19. SAGE Publications, 1996.

[17] Roneil S Lindo, John E Deaton, John H Cain, and Celine Lang. Methods of instrument training and effects on pilots' performance with different types of

flight instrument displays. *Aviation Psychology and Applied Human Factors*, 2(2):62, 2012.

[18] J. Mac McClellan. How to read approach charts. *Flying Magazine*, 119(5):84–91, May 1992.

[19] Mark G Mykityshyn, James K Kuchar, and R John Hansman. Experimental study of electronically based instrument approach plates. *The International Journal of Aviation Psychology*, 4(2):141–166, 1994.

[20] National Transportation Safety Board. Ntsb identification: Nyc97fa194. `http://www.ntsb.gov/_layouts/ntsb.aviation/brief2.aspx?ev_id=20001208X07226&ntsbno=NYC97FA194&akey=1`.

[21] Donald A Norman and Tim Shallice. *Attention to action*. Springer, 1986.

[22] Stephen Pope. The future of aviation. *Flying Magazine*, 141(7):78–82, July 2014.

[23] Helen C. Purchase. *Experimental Human-Computer Interaction*. Cambridge University Press, 32 Avenue of the Americas, New York, NY 10013-2473, USA, 2012.

[24] J Reason, E Hollnagel, and J Paries. Revisiting the swiss cheese model of accidents. *Journal of Clinical Engineering*, 27:110–115, 2006.

[25] Tom Simonite. Microsoft's hololens will put realistic 3-d people in your living room. Computing news, MIT Technology Review, May 2015.

[26] Peter Skaves. Electronic flight bag (efb) policy and guidance. In *Digital Avionics Systems Conference (DASC), 2011 IEEE/AIAA 30th*, pages 8D1–1. IEEE, 2011.

[27] Unknown. Great aviation quotes. `http://www.skygod.com/cgi/search.pl`.

APPENDIX A

GOOGLE GLASS SOFTWARE CODE

This program was written in the Java programming language by Eric Burke. The software was written as a LiveCard application. Code snippets for the XplaneConnection.java file have been taken from Zubair Khan's open-source Avare EFB software and modified for usage in this study.

A.1 Airport Class

```
// Created by Eric on 01/11/14.
public class Airport
{
        double locLatitude;
        double locLongitude;
        double locFreq;
        double atisFreq;
        double towerFreq;
        double ndbFreq;
        double gndFreq;
        int appCrs;

    public Airport(double locLatitude, double locLongitude, double
        locFreq, double atisFreq, double towerFreq, double ndbFreq, int
        appCrs, double gndFreq) {
        this.locLatitude = locLatitude;
        this.locLongitude = locLongitude;
```

```
        this.locFreq = locFreq;
        this.atisFreq = atisFreq;
        this.towerFreq = towerFreq;
        this.ndbFreq = ndbFreq;
        this.appCrs = appCrs;
        this.gndFreq = gndFreq;
    }
    public double getLocLatitude() {return locLatitude;}
    public double getLocLongitude() {return locLongitude;}
    public double getLocFreq() {return locFreq;}
    public double getAtisFreq() {return atisFreq;}
    public double getTowerFreq() {return towerFreq;}
    public double getNdbFreq() {return ndbFreq;}
    public int getAppCrs() {return appCrs;}
    public double getGndFreq() {return gndFreq;}
}
```

A.2 LiveCard Service

```
package com.example.eric.approachlive;

import com.google.android.glass.media.Sounds;
import com.google.android.glass.timeline.LiveCard;
import com.google.android.glass.timeline.LiveCard.PublishMode;
import com.google.android.glass.timeline.GlRenderer;
import android.app.Activity;
import android.app.PendingIntent;
import android.app.Service;
```

```
import android.content.Context;
import android.content.Intent;
import android.graphics.Color;
import android.location.Criteria;
import android.location.Location;
import android.location.LocationListener;
import android.location.LocationManager;
import android.media.AudioManager;
import android.opengl.EGLConfig;
import android.os.Bundle;
import android.os.Handler;
import android.os.IBinder;
import android.os.PowerManager;
import android.util.Log;
import android.widget.RemoteViews;
import android.opengl.GLES20;
import android.widget.Toast;

import java.io.IOException;
import java.io.InputStream;
import java.util.List;
import java.util.Timer;
import java.util.TimerTask;

/**
 * A {@link Service} that publishes a {@link LiveCard} in the timeline.
 */
```

```
public class LiveCardService extends Service {
    // Log tag
    private static final String TAG = "ApproachLiveCardService";

    // Live card and remoteviews on the card
    private LiveCard mLiveCard;
    RemoteViews remoteViews;

    // Location Services
    private LocationManager mLocationManager;
    private List<String> mLocationProviders;
    private String mLocationProvider;
    private Handler mHandler;

    // Declare GPS variables
    double mDistNXT;
    double mDistLOC;
    double mDist;
    double mEstablished;
    private String mAltText;
    private String mDme;
    private String mAppCourse;
    private String mLowerRightInfo;
    private String mLowerLeftInfo;
    private String mLowerUpLeftInfo;
    private String mTowerFrequency;
```

```
private String nextInfo;
private String missedInfo;
private String underlineInfo;

// Classes
Airport kiah;

// XPlane Class
XplaneConnection XP;

// Waypoint active boolean values
public boolean ttoroActive = false;
public boolean bgbukActive = false;
public boolean lassyActive = false;
public boolean dployActive = false;
public boolean jelliActive = false;
public boolean reignActive = false;
public boolean eelpoActive = false;
public boolean matonActive = false;
public boolean sakcoActive = false;
public boolean kiahActive = false;
public boolean landActive = false;
public boolean missedActive = false;

// Waypoint reached boolean values
public boolean ttoroReached = false;
public boolean bgbukReached = false;
```

```
public boolean lassyReached = false;
public boolean dployReached = false;
public boolean jelliReached = false;
public boolean reignReached = false;
public boolean eelpoReached = false;
public boolean matonReached = false;
public boolean sakcoReached = false;
public boolean kiahReached = false;

// Push notification counter boolean value
private boolean counter = false;
private boolean altCounter = false;
private boolean screenTurnOff = false;

// XP data boolean value
public boolean xPlaneStarted = false;
// Use either GPS or Xplane data
public boolean xPlaneConnected = false;

@Override
public IBinder onBind(Intent intent) {
    return null;
}
@Override
public int onStartCommand(Intent intent, int flags, int startId) {

    if("TTORO".equals(intent.getAction()) && !ttoroActive) {
```

```
        counter = false; // This allows to switch from one to the next
            without pushNot. problems
        altCounter = false;
        screenTurnOff = false;
        resetWaypoints();
        ttoroActive = true;
        Toast.makeText(this, "TTORO selected.",
            Toast.LENGTH_SHORT).show();
        // Start Xplane listener and receiver. If() allows starting XP
            from any WP if !running.
        if (!xPlaneStarted && xPlaneConnected){
            xPlaneStarted = true;
            this.XP = new XplaneConnection();
            XP.start();
        }
        pushNotification(2);
    }

    if("BGBUK".equals(intent.getAction()) && !bgbukActive) {
        counter = false; // This allows to switch from one to the next
            without pushNot. problems
        altCounter = false;
        screenTurnOff = false;
        resetWaypoints();
        bgbukActive = true;
        Toast.makeText(this, "BGBUK selected.",
            Toast.LENGTH_SHORT).show();
```

```
    // Start Xplane listener and receiver. If() allows starting XP
        from any WP if !running.
    if (!xPlaneStarted && xPlaneConnected){
        xPlaneStarted = true;
        this.XP = new XplaneConnection();
        XP.start();
    }
    pushNotification(2);
}

if("LASSY".equals(intent.getAction()) && !lassyActive) {
    counter = false; // This allows to switch from one to the next
        without pushNot. problems
    altCounter = false;
    screenTurnOff = false;
    resetWaypoints();
    lassyActive = true;
    Toast.makeText(this, "LASSY selected.",
        Toast.LENGTH_SHORT).show();
    // Start Xplane listener and receiver. If() allows starting XP
        from any WP if !running.
    if (!xPlaneStarted && xPlaneConnected){
        xPlaneStarted = true;
        this.XP = new XplaneConnection();
        XP.start();
    }
    pushNotification(2);
```

```
    counter = false;
}

if("DPLOY".equals(intent.getAction()) && !dployActive) {
    counter = false; // This allows to switch from one to the next
        without pushNot. problems
    altCounter = false;
    screenTurnOff = false;
    resetWaypoints();
    dployActive = true;
    Toast.makeText(this, "DPLOY selected.",
        Toast.LENGTH_SHORT).show();
    // Start Xplane listener and receiver. If() allows starting XP
        from any WP if !running.
    if (!xPlaneStarted && xPlaneConnected){
        xPlaneStarted = true;
        this.XP = new XplaneConnection();
        XP.start();
    }
    pushNotification(2);
}

if("JELLI".equals(intent.getAction()) && !jelliActive) {
    counter = false; // This allows to switch from one to the next
        without pushNot. problems
    altCounter = false;
    screenTurnOff = false;
```

```
        resetWaypoints();
        jelliActive = true;
        Toast.makeText(this, "JELLI selected.",
            Toast.LENGTH_SHORT).show();
        // Start Xplane listener and receiver. If() allows starting XP
            from any WP if !running.
        if (!xPlaneStarted && xPlaneConnected){
            xPlaneStarted = true;
            this.XP = new XplaneConnection();
            XP.start();
        }
        pushNotification(2);
    }

    if("REIGN".equals(intent.getAction()) && !reignActive) {
        counter = false; // This allows to switch from one to the next
            without pushNot. problems
        altCounter = false;
        screenTurnOff = false;
        resetWaypoints();
        reignActive = true;
        Toast.makeText(this, "REIGN selected.",
            Toast.LENGTH_SHORT).show();
        // Start Xplane listener and receiver. If() allows starting XP
            from any WP if !running.
        if (!xPlaneStarted && xPlaneConnected){
            xPlaneStarted = true;
```

```
        this.XP = new XplaneConnection();
        XP.start();
    }
    pushNotification(2);
}

if("EELPO".equals(intent.getAction()) && !eelpoActive) {
    counter = false; // This allows to switch from one to the next
        without pushNot. problems
    altCounter = false;
    screenTurnOff = false;
    resetWaypoints();
    eelpoActive = true;
    Toast.makeText(this, "EELPO selected.",
        Toast.LENGTH_SHORT).show();
    // Start Xplane listener and receiver. If() allows starting XP
        from any WP if !running.
    if (!xPlaneStarted && xPlaneConnected){
        xPlaneStarted = true;
        this.XP = new XplaneConnection();
        XP.start();
    }
    pushNotification(2);
}

if("MATON".equals(intent.getAction()) && !matonActive) {
```

```
        counter = false; // This allows to switch from one to the next
            without pushNot. problems
        altCounter = false;
        screenTurnOff = false;
        resetWaypoints();
        matonActive = true;
        Toast.makeText(this, "MATON selected.",
            Toast.LENGTH_SHORT).show();
        // Start Xplane listener and receiver. If() allows starting XP
            from any WP if !running.
        if (!xPlaneStarted && xPlaneConnected){
            xPlaneStarted = true;
            this.XP = new XplaneConnection();
            XP.start();
        }
        pushNotification(2);
    }

    if("SAKCO".equals(intent.getAction()) && !sakcoActive) {
        counter = false; // This allows to switch from one to the next
            without pushNot. problems
        altCounter = false;
        screenTurnOff = false;
        resetWaypoints();
        sakcoActive = true;
        Toast.makeText(this, "SAKCO selected.",
            Toast.LENGTH_SHORT).show();
```

```
    // Start Xplane listener and receiver. If() allows starting XP
        from any WP if !running.
    if (!xPlaneStarted && xPlaneConnected){
        xPlaneStarted = true;
        this.XP = new XplaneConnection();
        XP.start();
    }
    pushNotification(2);
}

if("KIAH".equals(intent.getAction()) && !kiahActive) {
    counter = false; // This allows to switch from one to the next
        without pushNot. problems
    altCounter = false;
    screenTurnOff = false;
    resetWaypoints();
    kiahActive = true;
    Toast.makeText(this, "KIAH selected.",
        Toast.LENGTH_SHORT).show();
    // Start Xplane listener and receiver. If() allows starting XP
        from any WP if !running.
    if (!xPlaneStarted && xPlaneConnected){
        xPlaneStarted = true;
        this.XP = new XplaneConnection();
        XP.start();
    }
    pushNotification(2);
```

```
}

if("LANDING".equals(intent.getAction()) && !landActive) {
    counter = false; // This allows to switch from one to the next
        without pushNot. problems
    altCounter = false;
    screenTurnOff = false;
    resetWaypoints();
    landActive = true;
    Toast.makeText(this, "DO YOU SEE AIRPORT ENVIRONMENT?",
        Toast.LENGTH_LONG).show();
    // Start Xplane listener and receiver. If() allows starting XP
        from any WP if !running.
    if (!xPlaneStarted && xPlaneConnected){
        xPlaneStarted = true;
        this.XP = new XplaneConnection();
        XP.start();
    }
        pushNotification(2);
}

if("MISSED".equals(intent.getAction()) && !missedActive) {
    counter = false; // This allows to switch from one to the next
        without pushNot. problems
    altCounter = false;
    screenTurnOff = false;
    resetWaypoints();
```

```
        missedActive = true;
        Toast.makeText(this, "MISSED APPROACH SELECTED.",
            Toast.LENGTH_LONG).show();
        // Start Xplane listener and receiver. If() allows starting XP
            from any WP if !running.
        if (!xPlaneStarted && xPlaneConnected){
            xPlaneStarted = true;
            this.XP = new XplaneConnection();
            XP.start();
        }
            pushNotification(2);
    }

    if (mLiveCard == null) {
        // Create live card object
        mLiveCard = new LiveCard(this, TAG);
        // Set rendering view of the live card
        remoteViews = new RemoteViews(getPackageName(),
            R.layout.live_card02);
        // Can contain TextView, ImageView, etc
        mLiveCard.setViews(remoteViews);
        // Display the options MENU when the live card is tapped.
        Intent menuIntent = new Intent(this,
            LiveCardMenuActivity.class);
        mLiveCard.setAction(PendingIntent.getActivity(this, 100,
            menuIntent, 0));
        mLiveCard.publish(PublishMode.REVEAL);
```

```
        // Runs this live card every xxxx milliseconds
        // mHandler = new Handler();
        // mHandler.postDelayed(runnable, 2000);

        // setup location manager
        mLocationManager = (LocationManager)
            getSystemService(LOCATION_SERVICE);

        Criteria criteria = new Criteria();
        criteria.setAccuracy(Criteria.ACCURACY_FINE);
        mLocationProviders = mLocationManager.getProviders(criteria,
            true /* enabledOnly */ );
        for (String provider : mLocationProviders) {
            mLocationManager.requestLocationUpdates(provider,
                    1000, // update every 1 sec (1000 millisec)
                    0, // update every 1 meters
                    mLocationListener);
        }
    }
    else {mLiveCard.navigate();}
    return START_STICKY;
}
/*
private Runnable runnable = new Runnable() {
    @Override
    public void run() {
```

```
        makeUseOfNewLocation(XP);
        mHandler.postDelayed(this, 2000);
    }
};*/
// listener for the location
private LocationListener mLocationListener = new LocationListener() {
    @Override
    public void onLocationChanged(Location location) {
        makeUseOfNewLocation(location);
    }
    @Override
    public void onProviderDisabled(String provider) {
    }
    @Override
    public void onProviderEnabled(String provider) {
    }
    @Override
    public void onStatusChanged(String provider, int status, Bundle
        extras) {
    }
};

protected void makeUseOfNewLocation(Location location) {
    if (location != null) {

        double GPSlatitude;
        double GPSlongitude;
```

```
double GPSaltitude;


GPSlatitude = location.getLatitude();
GPSlongitude = location.getLongitude();
GPSaltitude = location.getAltitude() * 3.28;
/*
GPSlatitude = XP.xLat;
GPSlongitude = XP.xLon;
GPSaltitude = XP.xAlt;
*/
boolean established = false;
int currentMinimum;
int color = Color.BLACK;

// kiah airport information
kiah = new Airport(29.993479, -95.360698, 109.7, 124.05,
    125.35, 326, 87, 121.7);

// Populate distance function
mDistLOC = distance(GPSlatitude, GPSlongitude,
    kiah.getLocLatitude(), kiah.getLocLongitude());

// Convert doubles to strings
mAppCourse = getResources().getString(R.string.course_info,
    kiah.appCrs);
mTowerFrequency = getResources().getString(R.string.tower_info,
    kiah.getTowerFreq());
```

```
/*::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::*/
/*::       Hardcoded KIAH check for ILS LOC APPROACH          :*/
/*::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::*/
 if (lassyActive) {
    mDistNXT = distance (GPSlatitude, GPSlongitude, 29.9919,
        -95.8607); // Lat Long of LASSY
    mDme = getResources().getString(R.string.lassy_info,
        mDistNXT);
    if (mDistNXT > 5.0 || mDistNXT < 5.0 && mDistNXT > 0.5) {
        if (mDistNXT > 5.0){
            mLowerLeftInfo =
                getResources().getString(R.string.atis_info,
                kiah.atisFreq);
        }
        else if (mDistNXT < 5.0 && mDistNXT > 0.5){
            pushNotification(1);
            mLowerUpLeftInfo =
                getResources().getString(R.string.loc_info,
                kiah.locFreq);
            mLowerLeftInfo =
                getResources().getString(R.string.course_info,
                kiah.appCrs);
            remoteViews.setTextColor(R.id.atisfreq, Color.CYAN);
            remoteViews.setTextColor(R.id.locfreq, Color.CYAN);
        }
        mAltText = "7000";
```

```
underlineInfo = "_____________";
nextInfo = "7000 after LASSY";

if (GPSaltitude < 6800){
    pushNotification(3);

    if (GPSaltitude >= 6300){
        color = Color.BLACK;
        remoteViews.setTextColor(R.id.latLonCoordinate,
            Color.YELLOW);
        remoteViews.setTextColor(R.id.underline,
            Color.YELLOW);
    }
    else if (GPSaltitude < 6300) {
        color = Color.RED;
        remoteViews.setTextColor(R.id.latLonCoordinate,
            Color.WHITE);
        remoteViews.setTextColor(R.id.underline,
            Color.WHITE);
    }
}
else {
    if (altCounter){
        screenTurnOff = false;
        altCounter = false;
        pushNotification(4);
    }
```

```
            color = Color.BLACK;
            remoteViews.setTextColor(R.id.latLonCoordinate,
                Color.GREEN);
            remoteViews.setTextColor(R.id.underline,
                Color.GREEN);
        }
    }
    else if (mDistNXT < 0.5 && mDistNXT > 0) {
        counter = false;
        pushNotification(1);
        altCounter = false;
        screenTurnOff = false;
        lassyActive = false;
        dployActive = true;
    }
}

if (dployActive) {
    mDist = distance (GPSlatitude, GPSlongitude,
        kiah.locLatitude, kiah.locLongitude);
    mDistNXT = distance (GPSlatitude, GPSlongitude, 29.9924,
        -95.7647); // Lat Lon of DPLOY
    mDme = getResources().getString(R.string.dme_info, mDist);
    mLowerLeftInfo =
        getResources().getString(R.string.course_info,
        kiah.appCrs);
```

```
mLowerUpLeftInfo =
    getResources().getString(R.string.tower_info,
    kiah.towerFreq);
remoteViews.setTextColor(R.id.locfreq, Color.CYAN);
remoteViews.setTextColor(R.id.atisfreq, Color.WHITE);

if (mDistNXT > 0.3) {
    mAltText = "7000";
    underlineInfo = "_____________";
    nextInfo = "5000 after DPLOY at 21.2 DME";

    if (GPSaltitude < 6800){
        pushNotification(3);

        if (GPSaltitude >= 6300){
            color = Color.BLACK;
            remoteViews.setTextColor(R.id.latLonCoordinate,
                Color.YELLOW);
            remoteViews.setTextColor(R.id.underline,
                Color.YELLOW);
        }
        else if (GPSaltitude < 6300){
            color = Color.RED;
            remoteViews.setTextColor(R.id.latLonCoordinate,
                Color.WHITE);
            remoteViews.setTextColor(R.id.underline,
                Color.WHITE);
```

```
            }
        }
        else {
            if (altCounter){
                screenTurnOff = false;
                altCounter = false;
                pushNotification(4);
            }
            color = Color.BLACK;
            remoteViews.setTextColor(R.id.latLonCoordinate,
                Color.GREEN);
            remoteViews.setTextColor(R.id.underline,
                Color.GREEN);
        }
    }
    else if (mDistNXT < 0.3 && mDistNXT > 0) {
        counter = false;
        pushNotification(1);
        altCounter = false;
        screenTurnOff = false;
        dployActive = false;
        jelliActive = true;
    }
}

if (jelliActive) {
```

```
mDistNXT = distance (GPSlatitude, GPSlongitude, 29.9927,
    -95.68334); // Lat Lon of JELLI
mDist = distance (GPSlatitude, GPSlongitude,
    kiah.locLatitude, kiah.locLongitude);
mDme = getResources().getString(R.string.dme_info, mDist);
mLowerLeftInfo =
    getResources().getString(R.string.course_info,
    kiah.appCrs);
mLowerUpLeftInfo =
    getResources().getString(R.string.tower_info,
    kiah.towerFreq);
remoteViews.setTextColor(R.id.atisfreq, Color.WHITE);
remoteViews.setTextColor(R.id.locfreq, Color.WHITE);

if (mDistNXT > 0.3) {
    mAltText = "5000";
    underlineInfo = "______________";
    nextInfo = "4000 after JELLI 16.9 DME";

    if (GPSaltitude < 4800){
        pushNotification(3);

        if (GPSaltitude >= 4500){
            color = Color.BLACK;
            remoteViews.setTextColor(R.id.latLonCoordinate,
                Color.YELLOW);
```

```
                remoteViews.setTextColor(R.id.underline,
                    Color.YELLOW);
            }
            else if (GPSaltitude < 4500){
                color = Color.RED;
                remoteViews.setTextColor(R.id.latLonCoordinate,
                    Color.WHITE);
                remoteViews.setTextColor(R.id.underline,
                    Color.WHITE);
            }
        }
        else {
            if (altCounter){
                screenTurnOff = false;
                altCounter = false;
                pushNotification(4);
            }
            color = Color.BLACK;
            remoteViews.setTextColor(R.id.latLonCoordinate,
                Color.GREEN);
            remoteViews.setTextColor(R.id.underline,
                Color.GREEN);
        }
    }
    else if (mDistNXT < 0.3 && mDistNXT > 0) {
        counter = false;
        pushNotification(1);
```

```
            altCounter = false;
            screenTurnOff = false;
            jelliActive = false;
            reignActive = true;
        }
    }

    if (reignActive) {
        mDistNXT = distance (GPSlatitude, GPSlongitude, 29.9929,
            -95.6110); // Lat Lon REIGN
        mDist = distance (GPSlatitude, GPSlongitude,
            kiah.locLatitude, kiah.locLongitude);
        mDme = getResources().getString(R.string.dme_info, mDist);
        mLowerLeftInfo =
            getResources().getString(R.string.course_info,
            kiah.appCrs);
        mLowerUpLeftInfo =
            getResources().getString(R.string.tower_info,
            kiah.towerFreq);
        remoteViews.setTextColor(R.id.atisfreq, Color.WHITE);
        remoteViews.setTextColor(R.id.locfreq, Color.WHITE);

        if (mDistNXT > 0.3) {
            mAltText = "4000";
            underlineInfo = "_____________";
            nextInfo = "3000 after REIGN at 13.2 DME";
```

```
if (GPSaltitude < 3800){
    pushNotification(3);

    if (GPSaltitude >= 3500){
        color = Color.BLACK;
        remoteViews.setTextColor(R.id.latLonCoordinate,
            Color.YELLOW);
        remoteViews.setTextColor(R.id.underline,
            Color.YELLOW);
    }
    else if (GPSaltitude < 3500){
        color = Color.RED;
        remoteViews.setTextColor(R.id.latLonCoordinate,
            Color.WHITE);
        remoteViews.setTextColor(R.id.underline,
            Color.WHITE);
    }
}
else {
    if (altCounter){
        screenTurnOff = false;
        altCounter = false;
        pushNotification(4);
    }
    color = Color.BLACK;
    remoteViews.setTextColor(R.id.latLonCoordinate,
        Color.GREEN);
```

```
            remoteViews.setTextColor(R.id.underline,
                Color.GREEN);
        }
    }
    else if (mDistNXT < 0.3 && mDistNXT > 0) {
        counter = false;
        pushNotification(1);
        altCounter = false;
        screenTurnOff = false;
        reignActive = false;
        eelpoActive = true;
    }
}

if (eelpoActive) {
    mDistNXT = distance (GPSlatitude, GPSlongitude, 29.9931,
        -95.53576); // Lat Lon EELPO
    mDist = distance (GPSlatitude, GPSlongitude,
        kiah.locLatitude, kiah.locLongitude);
    mDme = getResources().getString(R.string.dme_info, mDist);
    mLowerLeftInfo =
        getResources().getString(R.string.course_info,
        kiah.appCrs);
    mLowerUpLeftInfo =
        getResources().getString(R.string.tower_info,
        kiah.towerFreq);
    remoteViews.setTextColor(R.id.atisfreq, Color.WHITE);
```

```
remoteViews.setTextColor(R.id.locfreq, Color.WHITE);


if (mDistNXT > 0.3) {
    mAltText = "3000";
    underlineInfo = "_____________";
    nextInfo = "2000 after EELPO at 9.2 DME";

    if (GPSaltitude < 2900){
        pushNotification(3);

        if (GPSaltitude >= 2700){
            color = Color.BLACK;
            remoteViews.setTextColor(R.id.latLonCoordinate,
                Color.YELLOW);
            remoteViews.setTextColor(R.id.underline,
                Color.YELLOW);
        }
        else if (GPSaltitude < 2700){
            color = Color.RED;
            remoteViews.setTextColor(R.id.latLonCoordinate,
                Color.WHITE);
            remoteViews.setTextColor(R.id.underline,
                Color.WHITE);
        }
    }
    else {
        if (altCounter){
```

```
                screenTurnOff = false;
                altCounter = false;
                pushNotification(4);
            }
            color = Color.BLACK;
            remoteViews.setTextColor(R.id.latLonCoordinate,
                Color.GREEN);
            remoteViews.setTextColor(R.id.underline,
                Color.GREEN);
        }
    }
    else if (mDistNXT < 0.3 && mDistNXT > 0) {
        counter = false;
        pushNotification(1);
        altCounter = false;
        screenTurnOff = false;
        eelpoActive = false;
        matonActive = true;
    }
}

if (matonActive) {
    mDistNXT = distance (GPSlatitude, GPSlongitude, 29.99326,
        -95.46647); // Lat Lon MATON
    mDist = distance (GPSlatitude, GPSlongitude,
        kiah.locLatitude, kiah.locLongitude);
    mDme = getResources().getString(R.string.dme_info, mDist);
```

```
mLowerLeftInfo =
    getResources().getString(R.string.course_info,
    kiah.appCrs);
mLowerUpLeftInfo =
    getResources().getString(R.string.tower_info,
    kiah.towerFreq);
remoteViews.setTextColor(R.id.atisfreq, Color.WHITE);
remoteViews.setTextColor(R.id.locfreq, Color.WHITE);

if (mDistNXT > 0.3) {
    mAltText = "2000";
    underlineInfo = "_____________";
    nextInfo = "MDA 940 aft MATON 5.6 DME";

    if (GPSaltitude < 1900){
        pushNotification(3);

        if (GPSaltitude >= 1700){
            color = Color.BLACK;
            remoteViews.setTextColor(R.id.latLonCoordinate,
                Color.YELLOW);
            remoteViews.setTextColor(R.id.underline,
                Color.YELLOW);
        }
        else if (GPSaltitude < 1700){
            color = Color.RED;
```

```
                remoteViews.setTextColor(R.id.latLonCoordinate,
                    Color.WHITE);
                remoteViews.setTextColor(R.id.underline,
                    Color.WHITE);
            }
        }
        else {
            if (altCounter){
                screenTurnOff = false;
                altCounter = false;
                pushNotification(4);
            }
            color = Color.BLACK;
            remoteViews.setTextColor(R.id.latLonCoordinate,
                Color.GREEN);
            remoteViews.setTextColor(R.id.underline,
                Color.GREEN);
        }
    }
    else if (mDistNXT < 0.3 && mDistNXT > 0) {
        counter = false;
        pushNotification(1);
        altCounter = false;
        screenTurnOff = false;
        matonActive = false;
        sakcoActive = true;
    }
```

```
}

if (sakcoActive) {
    mDistNXT = distance(GPSlatitude, GPSlongitude, 29.99336,
        -95.40295); // Lat Lon SAKCO
    mDist = distance (GPSlatitude, GPSlongitude,
        kiah.locLatitude, kiah.locLongitude);
    mDme = getResources().getString(R.string.dme_info, mDist);
    mLowerLeftInfo =
        getResources().getString(R.string.course_info,
        kiah.appCrs);
    mLowerUpLeftInfo =
        getResources().getString(R.string.ground_info,
        kiah.gndFreq);
    remoteViews.setTextColor(R.id.atisfreq, Color.WHITE);
    remoteViews.setTextColor(R.id.locfreq, Color.CYAN);

    if (mDistNXT > 0.3){
        mAltText = "MDA 940";
        nextInfo = "GEAR?";
        underlineInfo = "";

        if (GPSaltitude < 940){
            pushNotification(3);
            color = Color.RED;
            remoteViews.setTextColor(R.id.latLonCoordinate,
                Color.WHITE);
```

```
            }
            else {
                if (altCounter){
                    screenTurnOff = false;
                    altCounter = false;
                    pushNotification(4);
                }
                color = Color.BLACK;
                remoteViews.setTextColor(R.id.latLonCoordinate,
                    Color.GREEN);
            }
        }
        else if (mDistNXT < 0.3 && mDistNXT > 0) {
            counter = false;
            pushNotification(1);
            altCounter = false;
            screenTurnOff = false;
            kiahReached = false;
            sakcoActive = false;
            kiahActive = true;
        }
    }

    if (kiahActive) {
        mDist = distance (GPSlatitude, GPSlongitude,
            kiah.locLatitude, kiah.locLongitude);
        mDme = getResources().getString(R.string.dme_info, mDist);
```

```
mLowerLeftInfo =
    getResources().getString(R.string.course_info,
    kiah.appCrs);
mLowerUpLeftInfo =
    getResources().getString(R.string.ground_info,
    kiah.gndFreq);
remoteViews.setTextColor(R.id.atisfreq, Color.WHITE);
remoteViews.setTextColor(R.id.locfreq, Color.WHITE);

if (mDist > 1.0 || mDist < 1.0 && mDist > 0.3) {
    if (mDist > 1.0) {
        counter = false;
        mAltText = "MDA 940";
        nextInfo = "VDP at 0.9 DME";
        underlineInfo = "";
    }
    else {
        pushNotification(1);
        mAltText = "MDA 940";
        nextInfo = "You have reached the VDP";
        underlineInfo = "";
    }
}

else if (mDist < 0.3 && GPSaltitude >= 1200) {
    counter = false;
    pushNotification(1);
```

```
            altCounter = false;
            screenTurnOff = false;
            kiahActive = false;
            missedActive = true;
        }
        if (GPSaltitude < 940){
            pushNotification(3);
            color = Color.RED;
            remoteViews.setTextColor(R.id.latLonCoordinate,
                Color.WHITE);
        }

        else {
            if (altCounter){
                screenTurnOff = false;
                altCounter = false;
                pushNotification(4);
            }
            color = Color.BLACK;
            remoteViews.setTextColor(R.id.latLonCoordinate,
                Color.GREEN);
        }
    }

    if (landActive) {
        mDist = distance (GPSlatitude, GPSlongitude,
            kiah.locLatitude, kiah.locLongitude);
```

```
mDme = getResources().getString(R.string.dme_info, mDist);
mLowerLeftInfo =
    getResources().getString(R.string.course_info,
    kiah.appCrs);
mLowerUpLeftInfo =
    getResources().getString(R.string.ground_info,
    kiah.gndFreq);
mAltText = "Arrival";
nextInfo = "Airport elevation 96";
underlineInfo = "";

if (GPSaltitude > 1500){
    pushNotification(3);
    color = Color.BLACK;
    remoteViews.setTextColor(R.id.latLonCoordinate,
        Color.YELLOW);
}
else {
    if (altCounter){
        screenTurnOff = false;
        altCounter = false;
        pushNotification(4);
    }
    color = Color.BLACK;
    remoteViews.setTextColor(R.id.latLonCoordinate,
        Color.GREEN);
```

```
    }
}


if (missedActive) {
    mDme = "BVP 326";
    mLowerLeftInfo = "HUB 117.1";
    mLowerUpLeftInfo = "IAH 116.1";
    mAltText = "";
    nextInfo = "";
    missedInfo = "Climb to 3000 on BVP NDB bearing 085 to MKAYE
        INT and hold.";
    if (GPSaltitude < 2900){
        pushNotification(3);
        color = Color.BLACK;
        remoteViews.setTextColor(R.id.missedInfo, Color.YELLOW);
    }
    else {
        if (altCounter){
            screenTurnOff = false;
            altCounter = false;
            pushNotification(4);
        }
        color = Color.BLACK;
        remoteViews.setTextColor(R.id.missedInfo, Color.GREEN);
    }
}
```

```
        else if (!missedActive){missedInfo = "";}
        // Populate the layout display variables
        remoteViews.setTextViewText(R.id.underline, underlineInfo);
        remoteViews.setTextViewText(R.id.nextInfo, nextInfo);
        remoteViews.setTextViewText(R.id.latLonCoordinate, mAltText);
        remoteViews.setTextViewText(R.id.missedInfo, missedInfo);
        remoteViews.setTextViewText(R.id.distance, mDme);
        remoteViews.setTextViewText(R.id.atisfreq, mLowerLeftInfo);
        remoteViews.setTextViewText(R.id.locfreq, mLowerUpLeftInfo);
        remoteViews.setTextViewText(R.id.appcourse, mLowerRightInfo);
        remoteViews.setInt(R.id.layout, "setBackgroundColor", color);
        // display it
        mLiveCard.setViews(remoteViews);
    }
}
// Turn off all active waypoints
public void resetWaypoints() {
    ttoroActive = false;
    bgbukActive = false;
    lassyActive = false;
    dployActive = false;
    jelliActive = false;
    reignActive = false;
    eelpoActive = false;
    matonActive = false;
    sakcoActive = false;
    kiahActive = false;
```

```
    landActive = false;
    missedActive = false;
}
private double distance(double lat1, double lon1, double lat2, double
    lon2) {
    double theta = lon1 - lon2;
    double dist = Math.sin(deg2rad(lat1)) * Math.sin(deg2rad(lat2)) +
        Math.cos(deg2rad(lat1)) * Math.cos(deg2rad(lat2)) *
        Math.cos(deg2rad(theta));
    dist = Math.acos(dist);
    dist = rad2deg(dist);
    dist = dist * 60 * 1.1515;
    dist = dist * 0.8684;
    return (dist);
}
/*::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::*/
/*::  This function converts decimal degrees to radians       :*/
/*::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::*/
private double deg2rad(double deg) {
    return (deg * Math.PI / 180.0);
}
/*::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::*/
/*::  This function converts radians to decimal degrees       :*/
/*::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::*/
private double rad2deg(double rad) {
    return (rad * 180 / Math.PI);
}
```

```
public void pushNotification(int type){
    PowerManager pm = (PowerManager) getSystemService(POWER_SERVICE);
    PowerManager.WakeLock wl =
        pm.newWakeLock(PowerManager.FULL_WAKE_LOCK |
        PowerManager.ACQUIRE_CAUSES_WAKEUP |
        PowerManager.ON_AFTER_RELEASE, "My Tag");
    AudioManager audio = (AudioManager)
        getSystemService(Context.AUDIO_SERVICE);
    switch(type){
        case 1:
            if(!pm.isScreenOn() && !counter) {
                // Always call setViews() to update the live card's
                    RemoteViews.
                mLiveCard.setViews(remoteViews);
                // Move to the live card before turning on the screen
                mLiveCard.navigate();

                audio.playSoundEffect(Sounds.SUCCESS);

                wl.acquire(20000);
                counter = true;
            }
            else if (pm.isScreenOn() && !counter){
                wl.acquire(10000);
                audio.playSoundEffect(Sounds.SUCCESS);
                counter = true;
            }
```

```
    break;
case 2:
    if (pm.isScreenOn() && !counter){
        // Always call setViews() to update the live card's
            RemoteViews.
        mLiveCard.setViews(remoteViews);
        // Move to the live card before turning on the screen
        mLiveCard.navigate();

        audio.playSoundEffect(Sounds.SUCCESS);

        wl.acquire(20000);
        counter = true;
        //Log.d(TAG, "In push 2 and counter is " + counter);
    }
    break;
case 3:
    if (!pm.isScreenOn()){
        // Always call setViews() to update the live card's
            RemoteViews.
        mLiveCard.setViews(remoteViews);
        // Move to the live card before turning on the screen
        mLiveCard.navigate();

        audio.playSoundEffect(Sounds.ERROR);

        wl.acquire(20000);
```

```
                altCounter = true;
                Log.d(TAG, "Keeping screen on");
            }
            break;
        case 4:
            if (pm.isScreenOn() && !screenTurnOff){
                // Always call setViews() to update the live card's
                    RemoteViews.
                mLiveCard.setViews(remoteViews);
                // Move to the live card before turning on the screen
                mLiveCard.navigate();

                wl.acquire();
                wl.release();
                screenTurnOff = true;
                Log.d(TAG, "Turning screen off");
            }
            break;
    }
}
 @Override
 public void onDestroy() {
    // stop listening to location
    if(mLocationProviders != null) {
        mLocationManager.removeUpdates(mLocationListener);
    }
    if (xPlaneConnected){
```

```
            // mHandler.removeCallbacks(runnable);
            XP.disconnect();
        }
        if (mLiveCard != null && mLiveCard.isPublished()) {
            mLiveCard.unpublish();
            mLiveCard = null;
        }
        // System.exit kills everything on exit, freeing up the port
            connected to Xplane.
        System.exit(0);
        super.onDestroy();
    }
}
```

A.3 Xplane Connection

```
/*
Copyright (c) 2012, Apps4Av Inc. (apps4av.com)
All rights reserved.
Redistribution and use in source and binary forms, with or without
    modification, are permitted provided that the following conditions are
    met:
    * Redistributions of source code must retain the above copyright
        notice, this list of conditions and the following disclaimer.
    *    * Redistributions in binary form must reproduce the above
        copyright notice, this list of conditions and the following
        disclaimer in the documentation and/or other materials provided
        with the distribution.
```

```
    *
    *    THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND
       CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES,
       INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF
       MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
       DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS
       BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
       EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED
       TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
       DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
       ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR
       TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF
       THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
       SUCH DAMAGE.
*/
package com.example.eric.approachlive;

import android.util.Log;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import org.json.JSONObject;

public class XplaneConnection {
    public XplaneConnection() {}
    private Thread mThread;
    double xLon, xLat, xAlt;
    DatagramSocket mSocket;
```

```
private static boolean mRunning;

// default port is 0
private int mPort = 49002;
private boolean mConnected = false;
final String TAG = "XplaneConnection";

public void start() {
    Log.d(TAG, "Starting XPlane Listener");
    mRunning = true;
    /*
     * Thread that reads Xplane
     */
    mThread = new Thread() {
        @Override
        public void run() {
            Log.d(TAG, "Xplane reading data");
            byte[] buffer = new byte[1024];
            /*
             * This state machine will keep trying to connect to
             * ADBS/GPS receiver
             */
            while(mRunning) {
                int red = 0;
                /*
                 * Read.
                 */
```

```
red = read(buffer);
if(red <= 0) {
    if(!mRunning) {
        break;
    }
    try {
        Thread.sleep(1000);
    } catch (Exception e) {
    }
    /*
     * Try to reconnect
     */
    //Log.d(TAG, "Listener error, re-starting listener");
    disconnect();
    connect(mPort);
    continue;
}

String input = new String(buffer);
if(input.startsWith("XGPS")) {
    String tokens[] = input.split(",");
    if(tokens.length >= 6) {
        /*
         * Make a GPS location message from ownship
             message.
         */
        JSONObject object = new JSONObject();
```

```
try {
    object.put("type", "ownship");
    object.put("longitude",
        Double.parseDouble(tokens[1]));
    xLon = Double.parseDouble(tokens[1]);
    //Log.d(TAG, "longitude " +
        Double.parseDouble(tokens[1]));
    object.put("latitude",
        Double.parseDouble(tokens[2]));
    xLat = Double.parseDouble(tokens[2]);
    object.put("speed",
        Double.parseDouble(tokens[5]));
    object.put("bearing",
        Double.parseDouble(tokens[4]));
    object.put("altitude",
        Double.parseDouble(tokens[3]));
    xAlt =
        (Double.parseDouble(tokens[3]))*3.28084;
    object.put("time",
        System.currentTimeMillis());
} catch (Exception e1) {
    continue;
}
/*
if(mHelper != null) {
    try {
        mHelper.sendDataText(object.toString());
```

```
                        Logger.Logit(object.toString());
                    } catch (Exception e) {
                    }
                }*/

            }
        }
    }
};
mThread.start();
}

public boolean connect(int port) {
    mPort = port;
    /*
     * Make socket
     */
    try {
        mSocket = new DatagramSocket(mPort);
    }
    catch(Exception e) {
        Log.d(TAG, "Failed! Connecting socket " + e.getMessage());
        return false;
    }
    Log.d(TAG, "Success!");
    mConnected = true;
```

```
    return true;
}

public void disconnect() {
    Log.d(TAG, "Disconnecting from device");
    //Logger.Logit("Disconnecting from device");
    /*
     * Exit
     */
    try {
        mSocket.close();
    }
    catch(Exception e2) {
        Log.d(TAG, "Error stream close");
    }
    mConnected = false;
    Log.d(TAG, "Listener stopped");
}

private int read(byte[] buffer) {
    DatagramPacket pkt = new DatagramPacket(buffer, buffer.length);
    try {
        mSocket.receive(pkt);
    }
    catch(Exception e) {
        return -1;
    }
```

```
        return pkt.getLength();
    }
}
```

A.4 LiveCard Menu

```
package com.example.eric.approachlive;


import android.app.Activity;
import android.content.Intent;
import android.util.Log;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.Toast;
import android.app.Service;
import android.app.IntentService;
/**
 * A transparent {@link Activity} displaying a "Stop" options menu to
    remove the {@link com.google.android.glass.timeline.LiveCard}.
 */
public class LiveCardMenuActivity extends Activity {
    private static final String TAG = "ApproachLiveCardMenu";
    @Override
    public void onAttachedToWindow() {
        super.onAttachedToWindow();
        // Open the options menu right away.
        openOptionsMenu();
    }
```

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.live_card, menu);
    return true;
}
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.action_stop:
            // Stop the service which will unpublish the live card.
            stopService(new Intent(this, LiveCardService.class));
            return true;
        case R.id.action_land:
            Intent land = new Intent(this, LiveCardService.class);
            land.setAction("LANDING");
            startService(land);
            return true;
        case R.id.action_missed:
            Intent missed = new Intent(this, LiveCardService.class);
            missed.setAction("MISSED");
            startService(missed);
            return true;
        /* Commented out as to not delete forever
        case R.id.action_ttoro:
            Intent ttoro = new Intent(this, LiveCardService.class);
            ttoro.setAction("TTORO");
            startService(ttoro);
```

```
            return true;
        case R.id.action_bgbuk:
            Intent bgbuk = new Intent(this, LiveCardService.class);
            bgbuk.setAction("BGBUK");
            startService(bgbuk);
            return true;
        */
        case R.id.action_lassy:
            Intent lassy = new Intent(this, LiveCardService.class);
            lassy.setAction("LASSY");
            startService(lassy);
            return true;
        case R.id.action_dploy:
            Intent dploy = new Intent(this, LiveCardService.class);
            dploy.setAction("DPLOY");
            startService(dploy);
            return true;
        case R.id.action_jelli:
            Intent jelli = new Intent(this, LiveCardService.class);
            jelli.setAction("JELLI");
            startService(jelli);
            return true;
        case R.id.action_reign:
            Intent reign = new Intent(this, LiveCardService.class);
            reign.setAction("REIGN");
            startService(reign);
            return true;
```

```
        case R.id.action_eelpo:
            Intent eelpo = new Intent(this, LiveCardService.class);
            eelpo.setAction("EELPO");
            startService(eelpo);
            return true;
        case R.id.action_maton:
            Intent maton = new Intent(this, LiveCardService.class);
            maton.setAction("MATON");
            startService(maton);
            return true;
        case R.id.action_sakco:
            Intent sakco = new Intent(this, LiveCardService.class);
            sakco.setAction("SAKCO");
            startService(sakco);
            return true;
        case R.id.action_kiah:
            Intent kiah = new Intent(this, LiveCardService.class);
            kiah.setAction("KIAH");
            startService(kiah);
            return true;
        default:
            return super.onOptionsItemSelected(item);
    }
}
@Override
public void onOptionsMenuClosed(Menu menu) {
    super.onOptionsMenuClosed(menu);
```

```
        // Nothing else to do, finish the Activity.
        finish();
    }
}
```