ABSTRACT OF CAPSTONE

Ian Levstein

The Graduate School

Morehead State University

April 9, 2018

CSI4FS® – A MARKERLESS AUGMENTED REALITY APPLICATION FOR FORENSIC SCIENCE CRIME SCENE INVESTIGATION TRAINING

Abstract of Capstone

A capstone submitted in partial fulfillment of the
Requirements for the degree of Doctor of Education in the
College of Education
At Morehead State University

By

Ian Levstein

Ashland, Kentucky

Committee Chair: Lenora J. Justice, Assistant Professor

Morehead, Kentucky

April 9, 2018

ABSTRACT OF CAPSTONE

CSI4FS® – A MARKERLESS AUGMENTED REALITY APPLICATION FOR FORENSIC SCIENCE CRIME SCENE INVESTIGATION TRAINING

Crime scene investigation (CSI) is an important part of the forensic science curriculum both at the graduate and undergraduate level. An augmented reality application written for a mobile broadband device, such as a smartphone, can be used by CSI instructors as part of the students’ training. By projecting interactive objects over a real-world display and receiving information from those interactive objects, forensic science students can enhance their skills by learning to better examine a crime scene from the clues received from those augmented reality objects. The infrastructure that allows the use of augmented reality applications for both education and entertainment is already built in to smartphones. This capstone reports on the design and development of an application for Android smartphones – Crime Scene Investigation for Forensic Science (CSI4FS®) – that specifically targets CSI training for forensic science students through a markerless augmented reality application.

CSI4FS® takes advantage of the ubiquitous nature of smartphones and the general ease with which current college students are able to manipulate those devices. Two sworn law enforcement officers, who are experienced crime scene investigators and adjunct professors in a graduate forensic science program, served as the subject matter experts for this application. The goal of this mobile application is to enhance CSI training by adapting existing CSI curriculum information and presenting that information in an innovative way to the student.

KEYWORDS: Augmented Reality, Forensic Science, Markerless, Training, Crime Scene Investigation, CSI

Candidate Signature

Date

CSI4FS® – A MARKERLESS AUGMENTED REALITY APPLICATION FOR FORENSIC SCIENCE CRIME SCENE INVESTIGATION TRAINING

By

Ian Levstein

Approved by

Dr. Elizabeth B. Perkins	
Committee Member	Date
Dr. Catherine G. Rushton	
Committee Member	Date
Dr. Lenora J. Justice	
Committee Chair	Date
Dr. Timothy L. Simpson	
Interim Department Chair	Date

RULES FOR THE USE OF CAPSTONES

Name	Date

CAPSTONE

Ian Levstein

The Graduate School

Morehead State University

April 9, 2018

CSI4FS® – A MARKERLESS AUGMENTED REALITY APPLICATION FOR FORENSIC SCIENCE CRIME SCENE INVESTIGATION TRAINING

Capstone

A capstone submitted in partial fulfillment of the
Requirements for the degree of Doctor of Education in the
College of Education
At Morehead State University

By

Ian Levstein

Ashland, Kentucky

Committee Chair: Lenora J. Justice, Assistant Professor

Morehead, Kentucky

April 9, 2018

DEDICATION

This capstone project is dedicated to two women named Sheila.

To my late mother, Sheila Levstein, who cried when I told her I had been admitted to a doctoral program. She did not live to see me graduate – but she continues to travel with me on the journey.

To my dear wife, Dr. Sheila Stephens, who encouraged me to pursue doctoral studies knowing full well the time commitment involved. Over the years her help, guidance, and encouragement never once waivered.

ACKNOWLEDGEMENTS

This capstone project would not have been possible without the involvement and support of many people:

To my doctoral committee chair – Dr. L. Jeannie Justice: your expert guidance throughout this process has been invaluable. This entire doctoral journey was much more fun because you were part of it – indeed, I would not now be here, but for you!

To the members of my doctoral committee – Dr. Catherine Rushton and Dr. Elizabeth Perkins: I deeply appreciate your ongoing participation and support.

To the members of Cohort 6, Top Gun – Chris Howes, Jami Hornbuckle, Jasmin Berrios, Jennifer Emberton, La Raissa Davis-Morris, and Sean Jackson: you have been my strength, my support, and my extended family throughout this amazing journey! I will never forget the encouragement you all gave when times were tough. I can’t wait to see how we will change the world!

To my subject matter experts – Lt. Dave Castle and Sgt. Steve Compton: I very much appreciate your help and your willingness to let me pick your brains!

To my research assistant – Cassie Passarella: I hope this was as much fun as you thought it would be. Your help was most welcome and I will never forget it.

To those who shared their skills – Alex Bethke and Jonathan Levstein: Thank you for sharing your talents when I was stuck. I will not forget your help.

To all those who beta tested the application and who offered suggestions for improvements and tweaks: Thank you! Your help was invaluable. I was able to incorporate many of your suggestions into the application and I know that it's better as a result. I now also have a list of additions for version 2.0!

To my wife and family – Sheila Stephens, Amy York, and Todd Stephens: many meals together were missed, family gatherings came and went without me, and weekend visits were almost unheard of… but I thank you all for never once saying that I neglected you or that I didn't care.

TABLE OF CONTENTS

Page

LIST OF FIGURES

DEFINITION OF TERMS

3D: Three dimensional; an object that can be displayed along the x, y, and z axes (Dorninger & Pfeifer, 2008).

Android: A mobile operating system based on Linux and developed by Google for use on compatible touch-screen smartphones (diCerbo, Girardello, Michahelles, & Voronkova, 2010).

Application (App): A software product created exclusively for mobile devices to add increased functionality to the device (Yang et al, 2015).

Augmented Reality (AR): A digital experience in which the real world is enhanced by computer-generated content tied to specific locations and/or activities (Yuen, Yaoyuneyong, & Johnson, 2011).

CSI: Crime scene investigation; a methodical and documented examination of a crime scene suitable for evidentiary use in a court of law (Poelman, Akman, Lukosch, & Jonker, 2012).

Digital Native: Someone born after 1988 who grew up in a digital, media-saturated world (Palfrey & Gasser, 2013).

GIF: Graphics Interchange Format. An image format developed by CompuServe in 1987 (Miano, 1999). The format is still in widespread use today.

GPS: Global positioning system; sends satellite-based geolocation information to a receiving device such as a smartphone (Zak, 2014).

Marker-Based: The simplest form of augmented reality for a mobile broadband device. A marker is an image that provides the platform upon which an augmented reality object can be superimposed (Lee, 2012).

Markerless: A more complex form of augmented reality for a mobile broadband device. The technology uses a GPS tracking system and image recognition to display the augmented reality information (Lee, 2012).

Mobile broadband device: A wireless, portable device such as a smartphone, capable of receiving an Internet signal delivered via cell towers (Wasko, 2013).

Smartphone: A hand-held device containing an operating system that integrates the functions of a personal computer, telephone, camera, and GPS (Zak, 2014).

Virtual Reality (VR): A three-dimensional simulation that couples a computer with a head-mounted display unit to create a full-immersion digital environment (Ausburn & Ausburn, 2010).

EXECUTIVE SUMMARY

There are some agreed on conventions for testing our knowledge, for critiquing our scholarship, for evaluating our art. But ultimately every idea, every principle, every theory originated in the mind of some scientist, artist, scholar, or writer. In a sense, I guess, they just made it up. – Dr. M. David Merrill

What is the core of the capstone?

The core of this capstone project is a technology simulation for use in forensic science undergraduate education training. Specifically, this project involves the development and creation of a markerless augmented reality (AR) application for forensic science crime scene investigation (CSI) training. Designed initially for students in a forensic science undergraduate degree program, the application will take the form of a mystery simulation – a crime that needs solving. Location information will be supplied by a mobile broadband device's GPS signal, and both clues and evidence will be superimposed using augmented reality. The player will tag and gather evidence, maintain the chain of custody, and determine what happened at the crime scene. The goal is to solve the mystery and, in the process of solving the mystery, learn about CSI protocols in the identification and collection of evidence from a crime scene. The simulation, however, goes beyond mere entertainment because the process and protocols used will match those required by law enforcement and the judicial system. By working through the simulation, forensic science students will become familiar with standard CSI protocols, documentation of evidence, seizure of evidence, and their own role in the judicial process. The AR application described here will be created for the Android platform. The various steps are outlined and consideration is noted for the various decisions made throughout the project.

Who is the capstone meant to impact?

The markerless augmented reality application that was developed and created as part of this capstone project will initially impact students in a forensic science undergraduate degree program at a mid-Atlantic academic institution. CSI training is an essential part of these students' curriculum and their training requires specialized staging of crime scenes for hands-on investigation. The staged sets are located at the facility's crime scene house – a three-story Victorian building located near campus. The curriculum includes a hands-on full-immersion simulation for teaching CSI skills. During class time, students study the search and seizure of evidence, evidence collection, and chain of custody protocols, and at the crime scene house the students put that knowledge to use developing skills in examining and collecting evidence and reconstructing the crime. This markerless augmented reality application will afford the students a complementary method of CSI training, free them from the confines of the crime scene house, and be available on demand.

While I anticipate almost complete buy-in from the students, there may be some hesitation from the CSI instructors either because of their unfamiliarity with apps or augmented reality or because they fear the loss of control in their classroom. I am not attempting to take anything away from the traditional methods of CSI training but instead offering a way for the students to move beyond the confines of a fixed location and staged sets. I want to help the students learn using a new tool – and perhaps one with which they are somewhat familiar and comfortable.

Ultimately, I do anticipate buy-in from the CSI instructors because the application has a few relative advantages over the current CSI training method. For example, when compared to the existing full-immersion simulation currently offered, it is certainly more convenient. Not only does it free the student from the fixed location, it will be available on demand. It may even be more fun because it is a simulation-based learning environment. There is also an opportunity for decreased discomfort since the student would be able to move freely about the crime scene environment without actually disturbing it and with the knowledge that the scene can be reset immediately if needed. That aspect will appeal to the CSI instructors – the ability to reset the crime scene within a few seconds rather than having to recreate the staging. The application is free for forensic science training. Having a low- or no-cost alternative for CSI training will appeal not only to CSI instructors, but also to CSI administrators.

CSI administrators are always looking to keep costs down – and this application will afford them the opportunity to spend no money at all to educate the students where CSI training is concerned. The application will cost them nothing, and there is no infrastructure to purchase, because the students' own smartphones already have the infrastructure built in. The Mobile Access 2015 report (Lenhart, 2015) states that approximately 75-80% of users across a broad spectrum of both age and ethnicity demographics, already own the type of mobile broadband device that application developers expect in order for instructors to use augmented reality environments with their students. Two sworn law enforcement officers, who are experienced crime scene investigators and adjunct professors in a graduate forensic science program, served as the subject matter experts for this application, so there are no conflicting issues with current CSI training practice, value, and needs.

In the future, the project could impact forensic science students across the U.S. and, perhaps, internationally. Students and instructors will be able to download, use, and modify the program to meet their specific needs. In time, the capstone project may impact law enforcement generally and crime scene investigators specifically.

How was the capstone project implemented?

Initially, there were only a few stakeholders in the project including students and faculty members. By the project's completion, law enforcement officers were able to use the application for training as easily as the students. As the project moved into beta testing, ongoing input from the students and faculty was sought in order to refine the application to make it as user friendly and as easily navigable as possible.

Capstone Project Beginnings

The idea for this project began as simple research into virtual reality (VR) and how VR could be used in an educational forensic science setting. Initially, the project concept was to create a VR application using Google Cardboard, but after watching a demonstration of augmented reality using Google Cardboard at the 2016 Association for Educational Communications and Technology (AECT) conference, my interest in

AR technology piqued. Using AR would solve a number of logistic issues – the two most difficult being programming and cost. With a VR application, for example, the cost alone of a head-mounted display unit for professional training purposes could easily climb upwards of $1,000. Using AR rather than VR renders the project more cost effective and manageable.

Unity®, Vuforia®, and C#

In order to create an augmented reality application, two specialized software products, Unity® and Vuforia®, are required. Both software programs are available at no cost, although registration is required for Vuforia® to obtain a non-commercial augmented reality license. The C# programming language is used to manipulate and move objects within the AR environment.

Marker-based AR vs Markerless AR

The simplest form of AR for a mobile device uses a marker-based technology. A marker is simply an image that provides the platform upon which an AR object can be superimposed. As the marker moves, the AR image moves; as the marker turns, the AR image turns. The AR image, however, can never be removed from the marker, which is to say that the AR image is always dependent on the existence of the marker. Without the marker, the AR image cannot exist. Therefore, a marker-based AR application is not feasible because it would be useful only if the user had the specific marker created for the project. To be fully effective and usable in any situation, a markerless application is required – and that is what was created for this capstone project. Markerless technology uses a device's GPS tracking system with image recognition to display the augmented reality information. It allows the application to be used anywhere without the restrictions placed on it by marker-based technology.

Creating a Marker-Based AR Object

As background for the project, I created a 3D augmented reality object. Creating the 3D object was a process involving several steps: creating a marker, creating the AR object, creating an animated sprite, and finally bringing all the components together so that they worked seamlessly. After the marker was created, the AR image could then be superimposed on the marker. The camera that displays the AR object must be able to see the marker from any angle and at a reasonable distance in order to render the AR object properly. An effective marker is highly textured with

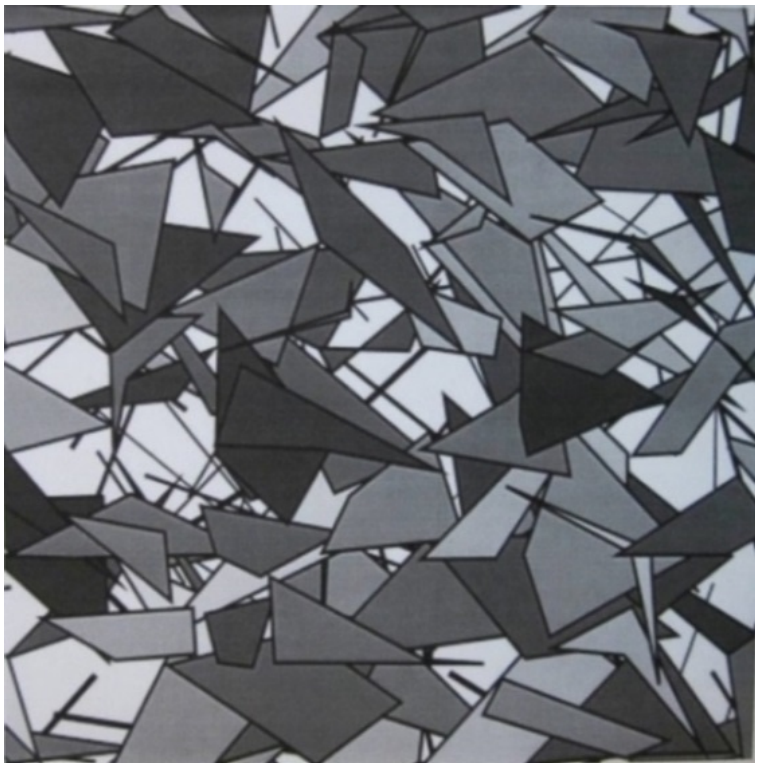


Figure 1. Highly textured image used as a marker.

high contrast, no repeating elements, no single focus area, and a good distribution of content. Figure 1 shows the marker used for testing the Unity® and Vuforia® software. The image was printed on plain white paper and even though the grayscale was muted somewhat by the printer (see Figure 2), the image met all the criteria and was suitable as an effective marker.

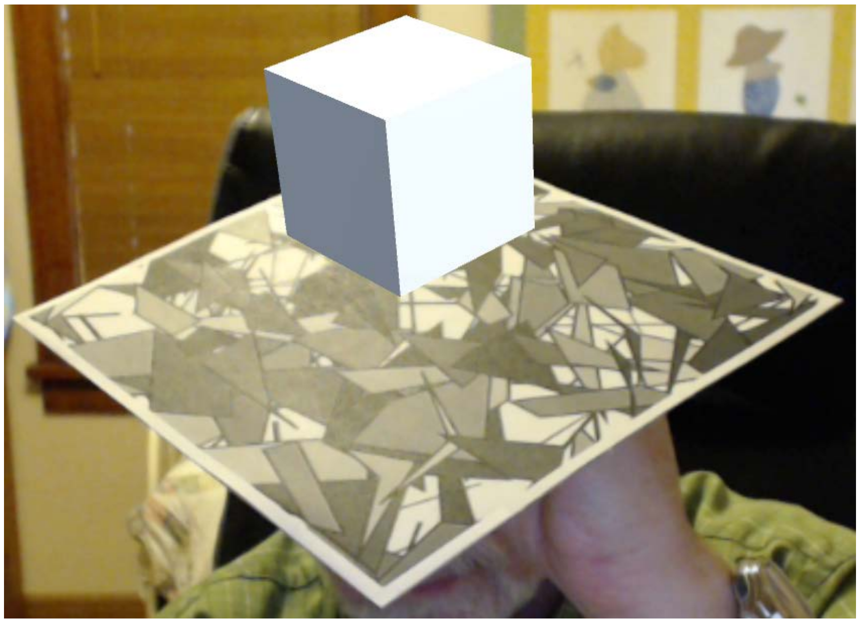


Figure 2. 3D cube superimposed above the marker.

Once the marker was created, it was imported into Unity® as a plane with dimensions visible only on the x and y axes. It should be noted that even though the paper on which the marker was printed had a definite thickness to it (z axis), it was negligible with respect to the camera. For the AR object, a cube was selected from Unity®'s 3D Object menu. Activating a webcam from within Unity® forces focus on the marker and superimposes the 3D object onto or above the plane of the marker.

Figure 2 shows the superimposed cube above the the marker. This was the first attempt at creating an AR object and the image shown depicts the view from the webcam. I am holding the printed marker, and the 3D cube appears to float above the plane of the marker. As the marker is moved, the cube moves with it; as the marker is turned, the cube also turns. At a more oblique angle, Figure 3 shows the 3D cube clearly floating above the plane of the marker.

A static 3D cube superimposed on a marker is interesting, but the novelty wears off quickly so the challenge was then to superimpose a dynamic object – an animated object – onto the marker. Unity® does not natively support animated AR objects so a workaround was required. An animated GIF was located online and imported into Adobe Photoshop®. The software deconstructed the GIF into its component frames, or layers, which then allowed the creation of a single sprite

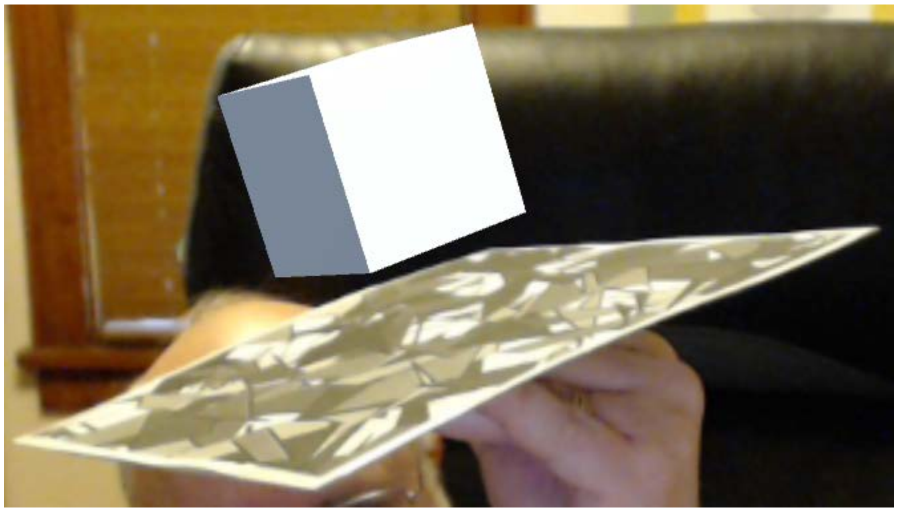


Figure 3. 3D cube floats above the plane of the marker.

file that contained an image of the GIF's ten-frame animation sequence from beginning to end (see Figure 4). The sprite file was imported into Unity®, and Unity® spliced the image into its component frames and recompiled the animated sequence. At this point, the webcam was able to project the animated object above the plane of the marker. Figure 5 demonstrates the animated AR object hovering well above the plane of the marker. Although Figure 5 displays a static image, the animated image is clearly evident in a live demonstration.

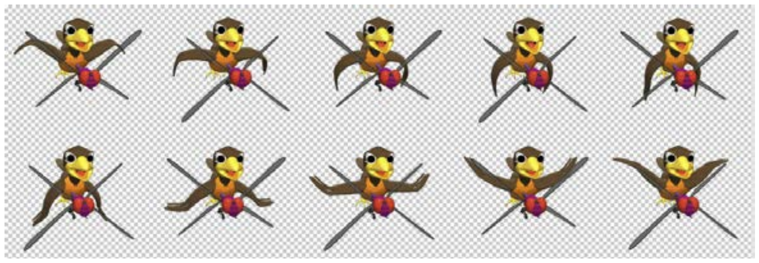


Figure 4. Sprite file of an animated GIF sequence.

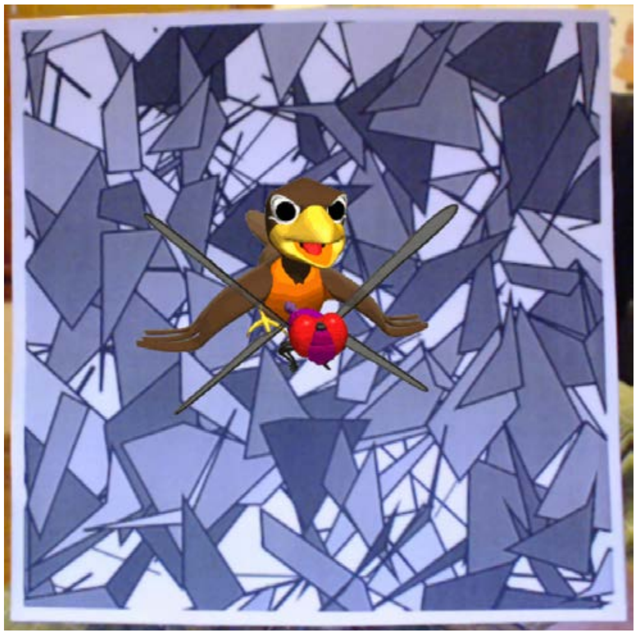


Figure 5. Animated image above the plane of the marker.

A Markerless Application

Marker-based augmented reality is restrictive because the AR object depends on the existence of the marker. It follows, therefore, that the only viable method for a truly mobile product for forensic science CSI training is the creation of a markerless application. This capstone project resulted in a markerless augmented reality product based on the needs of current students in a forensic science undergraduate degree program at a mid-Atlantic academic institution. Information about the types of activities that take place during a crime scene investigation were gathered from two of the adjunct faculty members who teach crime scene investigation to forensic science graduate students and who served as subject matter experts for the project. In addition to being faculty members, both are sworn law enforcement officers at a local police department and are experienced crime scene investigators. This helped to ensure that the project incorporated not only the information required during the search and seizure of evidence, but that protocols of evidence collection, preservation, and chain of custody were also followed. Ongoing collaboration with the officers, most notably during beta testing, was a key component to the success of this project.

Successful completion of this project required my proficiency with several software programs that were used to create this markerless augmented reality application. The main program, Unity®, had a fairly steep learning curve before even a simple game could be created. The add-on extension program, Vuforia®, was not as difficult to learn and easily integrated with Unity® however its ultimate value was minimal (see CSI4FS® Preliminary Work, page 41). I also needed to learn the C#

programming language in order to manipulate the various AR objects used in the application.

Why were this capstone and related strategies selected?

In its report, Strengthening Forensic Science in the United States: A Path Forward (2009), the National Research Council states that formal university training is preferred to on-the-job training. Further, limited resources for training and mentoring are common challenges for law enforcement (Drake, Giardino, Giardino & Nolte, 2016). In an economic climate where budgets are increasingly dwindling, law enforcement has shifted from training crime scene investigators on-the-job to a preferred method of hiring students from forensic science graduate programs who already have some crime scene investigation (CSI) skills as part of their graduate curriculum (Ludwig, Fraser, & Williams, 2012). The knowledge and skills of crime scene investigators are not only vital in criminal investigations and the prosecution of crimes, but are also crucial in understanding man-made disasters and global crimes (Lee & Pagliaro, 2013). For proficiency at a crime scene, all investigations must be methodical, logical, and systematic – and the expertise of the crime scene investigators is a reflection of their training and education (Lee & Pagliaro, 2013).

An experienced crime scene investigator can collect evidence and make interpretations about that evidence without destroying that evidence, while a novice investigator is at high risk to violate evidence protocols (Forsythe, 2004; Giova, 2011; Lee & Pagliaro, 2013). Improper investigation at a crime scene has the potential to negatively impact not only the investigator on both a personal and professional level, it also impacts local law enforcement, the court system, the attorneys, the accused, and, perhaps most importantly, the victim of a crime. One professional error has the potential to profoundly affect an individual, a community, a state, and a country (Baker, 2009; Forsythe, 2004).

Augmented Reality vs Virtual Reality

Augmented reality superimposes and integrates digital information with a user's environment in real time (Zak, 2014). It differs from virtual reality in that virtual reality gives the user a sense of presence and of being there. Ausburn and Ausburn (2010) suggest that virtual reality presence refers to the illusion of non-mediation, or the sense that the user has actually *been* to an event rather than merely observing an event. The person is not merely an observer in the virtual reality activity, but an active participant (Rothbaum, Hodges, Smith, Lee, & Price, 2000). Augmented reality lies midway between the real environment and the virtual environment; it bridges the gap between them in a seamless way (Lee, 2012). Where virtual reality creates a full-immersion simulation to mimic reality, augmented reality uses overlays to enhance reality. Augmented reality introduces digital information such as audio, video, graphics, and satellite-based global positioning system (GPS) data into a real environment.

The most common use of augmented reality is via the introduction of GPS data and graphics to a smartphone application. While virtual reality has focused on sophisticated head-mounted displays linked to a computer, augmented reality has focused almost exclusively on mobile broadband devices (Wasko, 2013). Good quality head-mounted display units cost from $400 to $3,000 (Lamkin, 2017); whereas, a mobile broadband device such as a smartphone generally costs less and since many people already have access to one there is no increased cost to the user to enjoy augmented reality. Further, many applications are distributed at no cost. One such application, Pokémon Go – a free, augmented reality, location-based game, proved very popular. It was released in mid-July 2016 by software development company Niantic. Within days, the game became a worldwide phenomenon and within two weeks, the application had been downloaded more than 100 million times (Doran & Davis, 2016; Moon, 2016; O'Rourke, 2016). The popularity of this application and others demonstrates that augmented reality is poised to become a breakthrough technology, and the markerless augmented reality application developed in this capstone project will take advantage of that popularity.

Simulation Use in High-Risk Profession Training

Edwin Link introduced a flight simulator, the Link Trainer, in 1929 (Page, 2000). Since then, simulation technology has enabled both students and professionals to gain experience without endangering life and limb in as close to real-world situations as possible (Page, 2000). In 1934, the U.S. Army Air Corps purchased six Link Trainers as a result of the deaths of twelve pilots over a 78-day period because the pilots were not familiar with Instrumental Flight Rules. By 1955, the Federal Aviation Administration required recertification on a simulator to maintain commercial pilots' licenses (Rosen, 2008). During the early 1960s, the National

Aeronautics and Space Administration’s Gemini program used a combination of both analog and digital simulators, but by the launch of Apollo 7 in 1968, digital simulators were used exclusively for astronaut training. Sophisticated simulation technologies became available driven by the military’s need for high-resolution graphics – a direct consequence of the Link Trainer (Macedonia, 2002). Based on the success with flight training, power plant simulators began to emerge in the 1970s (Rosen, 2008).

While simulation entered medical student training in the 1930s with the introduction of the simulated patient (Gaba, 2004), decades later the graphics development for gaming platforms provided the realism needed to catapult the potential for high-fidelity patients (Macedonia, 2002; Rosen, 2008). High-fidelity patient models were introduced to medical student training in the 1990s and offered realistic skin tension, joints, muscles and, in some of the more sophisticated models, heart and lung systems (Gordon, Wilkerson, Shaffer, & Armstrong, 2001; Rodgers, Securro, & Pauley, 2009).

Surgical trainees who underwent simulator training outperformed their colleagues without the simulator experience (Seymour et al., 2002). Those who had received simulation training for a procedure were faster and made fewer errors. Simulation training contributed to the development of skills relevant for real procedures and shortened the learning curve for new procedures (Seymour et al., 2002). Simulation training with more realistic, high fidelity training showed even better results. While there was no significant difference between the static and high-fidelity simulation groups on the written examination, the high-fidelity simulation group outperformed the static simulation group on actual performance of the skill (King & Reising, 2011).

Noting such results and because of the potential to cause great harm to oneself or others associated with training in high-risk professions, computer-based simulations have become the de facto method for student and professional training in medicine and surgery (Ahmed & Dann, 2015; Khan, Aydin, Khan, Dasgupta, & Ahmed, 2015), and in other difficult or dangerous situations. In addition to medical and military combat training, such simulations have now entered the fields of firefighting (Dugdale, Pavard, Pallamin, el-Jed, & Maugan, 2004), air traffic control (Glover & Lygeros, 2004), and hazardous waste management (West, Slatin, Sanborn, & Volicer, 2009), and continues to be used in airplane, helicopter, naval, and submarine training (Macedonia, 2002; Rosen, 2008). Simulation training technology provides a valuable hands-on learning experience without risk of damage to equipment, and without risk of harm to students, instructors, clients, and observers.

Simulation in Law Enforcement

Law enforcement is notably missing from the list of high risk, high danger professions taking advantage of simulation training. While the FBI uses simulation for rapid-response and hostage situation training, non-Federal law enforcement has been slow to embrace computer-based augmented reality simulation training for cadets and active duty officers with the possible exception of firearms training and high-speed chase techniques (Forsythe, 2004).

Computer-animated crime scene reconstruction, however, is replacing more traditional photographs and verbal descriptions in courtrooms (Agarwal, Gupta, Gupta, & Gupta, 2011; Gee, Escamilla-Ambrosio, Webb, Mayol-Cuevas, & Calway, 2010; Lee & Pagliaro, 2013). Simulation technology plays an important part by allowing the court members not just to view but to better experience what happens at a crime scene (Schofield, 2009; Xu, 2012). Computer vision, 3D graphics, motion tracking, natural language visualization, and forensic computing are changing the landscape of the courtroom. While the level of interactivity and the level of detail are dependent on the crime, concerns such as relevance, authenticity, and fairness must still be verified before being used as evidence (Ma, Zheng, & Lallie, 2010).

Although not specific to CSI, a descriptive study of twenty-six forensic science students found that students enhanced their critical thinking through simulation environments (Baker, 2009). Further, 100% of the students reported improved knowledge of forensic evidence collection, and 88% improved their understanding of the investigative process. Baker (2009) noted that role-playing, active learning, and critical thinking were mutually reinforcing in a simulation environment.

Wethal (2010) notes that CSI is largely a hands-on applied discipline relying on experience and competence. In their study on VR simulation, Grantcharov et al. (2004) determined that confidence, knowledge, and critical decision skills improved significantly for students who trained using simulation technology.

Augmented Reality in Education

Augmented reality has the potential to provide for the discovery of information in the real world as well as providing a powerful contextual on-site learning experience (Zak, 2014). One such use of augmented reality in education came about in the late 1990s and early 2000s when schools in the U.S. saw a new device in the classroom: a SMART board. This device gave users the ability, for example, to write over a Microsoft Powerpoint slide, and then save the written information along with the slide information (Lee, 2012).

In their engineering research, Liarokapis et al. (2004) suggest that complicated mechanisms and difficult theories in higher education can be easily accepted and understood by students with the use of augmented reality technology. Indeed, augmented reality technology has been used successfully in such areas as astronomy, chemistry, biology, mathematics, geometry, and physics (Lee, 2012). These are all science-based subjects so it is reasonable to suggest that students in a forensic science course would also be able to make use of augmented reality technology. Augmented reality as used in education impacts four areas: interactive education, simplicity, efficiency and effectiveness, and contextual information.

Interactive education combines social networking with other technologies. AR can not only engage learners in a variety of interactive ways that have never before been possible but can provide each individual with a unique discovery path with rich content from computer-generated environments and models (Lee, 2012; Singhal, Bagga, Goyal, & Saxena, 2012). As more digital natives enter higher education, simplicity becomes paramount. Any technology that is more complicated than it needs to be, will be neglected in favor of technology that simplifies learning and takes advantage of the way students learn in a digital environment (Yuen, Yaoyuneyong, & Johnson, 2011). Efficiency and effectiveness of augmented reality in an education setting can be promoted by offering rich content with computer-generated imagery. Augmented reality can provide an entertaining, motivating, and engaging environment that is stimulating, attractive, and exciting for students (Lee, 2012). Finally, contextual information, such as GPS location can enhance the quality of the information the student receives as part of a lesson using augmented reality.

The Mobile Access 2015 report (Lenhart, 2015) states that 94% of all teenagers are online daily while 91% of all teenagers use a mobile device for their online access. The report further suggests that approximately 75-80% of users across a broad spectrum of both age and ethnicity demographics, already own the type of mobile broadband device that application developers expect in order for instructors to use augmented reality environments with their students. The increased use of mobile broadband devices such as smartphones coupled with the availability of free applications and the naturally curious nature of students pursuing forensic science as a career suggests that the infrastructure is ready for an AR application that serves the training needs of those forensic science students.

Augmented Reality in CSI Training

While classroom instruction aids CSI training, Baker (2009) suggests that a superior field experience can be achieved using an augmented reality simulation. AR technology can provide the novice crime scene investigator with the opportunity to think critically and to apply theory. Further, the best simulations involve learner participation using all of the senses: auditory, visual, and psychomotor (Baker, 2009). An augmented reality simulation can provide hands-on learning that is engaging for the student while teaching skills of collaboration, creativity and critical thinking (Johnson, Adams & Cummins, 2012). I believe an augmented reality CSI application would allow the student to walk through a crime scene, collect evidence items into custody, tag important information, interpret evidence, maintain the chain of custody, and interview witnesses – all without disturbing the original crime scene.

Simulations have been used successfully in pilot training, air traffic control, medicine, firefighting, law enforcement, and other professions (Dugdale, Pavard, Pallamin, el-Jed, & Maugan, 2004; Gaba, 2004; Glover & Lygeros, 2004; Gordon, Wilkerson, Shaffer, & Armstrong, 2001; Page, 2000; Rosen, 2008; Schofield, 2009; Xu, 2012). Wayne et al. (2008) found simulation training to be significantly superior to traditional education because it offered heightened sensory response to a situation. I believe that an augmented reality application could provide an enhanced real-world CSI experience for forensic science undergraduate students and thereby complement the traditional method of CSI training.

I believe that a markerless augmented reality application specifically for CSI training is feasible, and for this project I created a markerless augmented reality application specifically for forensic science CSI training. That application was made available at no cost to the education community.

Issues with Implementation in Education

While the concept of augmented reality can trace its roots back to 1901, the term was not coined until 1992. Its practical use dates to 1968 and the development of a system that used computer-based graphics to demonstrate simple wireframe drawings. Even now, 50 years after its initial development, adopting augmented reality in education is still quite challenging. Integration issues with traditional learning methods, costs for the development and maintenance of the augmented reality devices and systems, and a general resistance to new technologies all contribute to the problem. Further, there has been little financial support from the government and a general lack of awareness that augmented reality can be used in an academic setting, and many questions still linger about its usefulness (Lee, 2012).

When was the capstone implemented?

This project began in early September, 2017. Input from the subject matter experts was gathered during November and during beta testing of the application. The software coding and programming to build the application took place from September through mid-February, 2018. Beta testing of the application began in mid-February and ran through early March; the application was complete by mid-March.

Impact of the capstone

Impact is largely minimal at this point in time. Once the application is fully released and more widely available, that may change. One of the first issues during beta testing was that the application would not download properly from the Google Play Store (see Step One, page 62). This, however, was a simple fix and involved repackaging the application properly using Unity®.

Comments from the various beta testers were largely favorable. Many enjoyed the simple user interface and the changing AR objects. They all agreed that the text was clear and that presenting the simulation in the form of a lesson enhanced both the application's appeal and its usefulness. One tester suggested that the original royal blue text color was difficult to read when superimposed over the AR tablet, so the text color was subsequently lightened to make it more readable. Another tester suggested including information about the landscape design of the application and to make sure users rotated their devices appropriately. The tester also suggested clarification on the process of touching and holding an object else the on-screen information appears and disappears too quickly to follow. These suggestions were incorporated into the existing beta release.

There were also comments regarding features for future versions of the application. Several testers suggested the use of a menu screen at the beginning of the simulation so that when several lessons have been developed, users can choose which lesson to review. One suggestion was the use of a toolbar at the top of the screen where, once an AR object was viewed, its icon would then reside in the toolbar. This would help to serve as a checkpoint in case the user wanted to go back and review an item after having already seen the simulation – even within the same lesson. One tester suggested spawning the AR objects in various locations so that the user would then have to hunt around for the objects as a means of making the simulation more interactive. One tester suggested that the application incorporate a portrait design so that the phone did not need to rotate. There also needs to be an increased emphasis on the mystery aspect of the simulation to make the simulation more enjoyable for the end user. Also, introducing drop-down lists that allowed users to select and pursue different paths of inquiry, even if those paths were ultimately incorrect, would better help the students to focus on correct assessments and analysis.

Limitations of the study

Several limitations to this capstone project were realized. One such limitation was generalizability due to the small target population for the project. While the resulting application might be marketed in the future to a wider audience, the initial target for this capstone was 30-40 people taken from a convenience sample of students registered in a forensic science undergraduate degree program at a mid-Atlantic academic institution. Another limitation, also dealing with the target population, was their level of comfort using an augmented reality application. The forensic science students had different abilities and their success using the application was a combination of their expertise with such applications and my ability to create an intuitive AR environment. A final limitation was my own lack of programing experience. While this was (mostly) overcome with practice during the course of the project, early failures definitely impacted progress.

Reflections

My initial focus on using Vuforia® as a means of creating a markerless AR application was a waste of both time and energy. My initial research suggested that Vuforia® would be a useful software program in the creation of a markerless AR application but, months later, I found its value to be minimal at best. I spent months trying to get Vuforia® to create a markerless environment – to no avail. This was an unexpected setback to the capstone project. I was still able to move forward but, on reflection, I wish I had done better research initially. I believe, however, that this was a natural stepping stone in the process since only by doing or creating, do we come to understand what works and what does not. It is sometimes difficult to predict the next step in a process until results have been realized for prior steps – and such is the creation process. It was a disappointing moment, but it was an important lesson.

I have become aware that the doctoral journey is less about being an independent researcher and more about pushing forward with a project when you are working alone and unsupervised. On more than one occasion I seriously considered ending my doctoral work and not finishing the project because I was completely overwhelmed by the amount of work I had set for myself. It took an incredible amount of self-disciple to stay on course and see this project through to completion.

Perhaps what is really most important is that I had a completely unrealistic expectation of what I was about to tackle. I knew what I wanted to do, but when the proposal defense was over and I was committed to a course of action, I really was not sure I would be able to finish what I had started. I had no idea how to develop an application using Unity®, I had no programming experience with C# or any other computer language, and I had no idea how to even get an application off my computer onto a smartphone.

I have enjoyed this capstone project tremendously. It has been a lot of fun to watch the small bits and pieces of code come together over time and then work together in unison. Sometimes things worked; sometimes they did not – and much of the time I was stumbling around – but, in the end, it all came together. I have created something which did not, before, exist – and I am quite proud of that.

CAPSTONE PROJECT – CSI4FS®

The full project documentation is available at http://www.csi4fs.com. The acronym CSI4FS® stands for: Crime Scene Investigation for Forensic Science.

Registered Trademark: CSI4FS®

Throughout this capstone project, the name of the AR application has been specifically written as: CSI4FS®. When the project began, I believed that this application might have some future commercial value. Wanting to protect my interests, I registered a business with the Secretary of State for the Commonwealth of Kentucky. The business, Levstein Holdings, LLC, registered an application with the U.S. Patent and Trademark Office (USPTO) to trademark the smartphone application. The registration was subsequently approved and CSI4FS® holds USPTO trademark

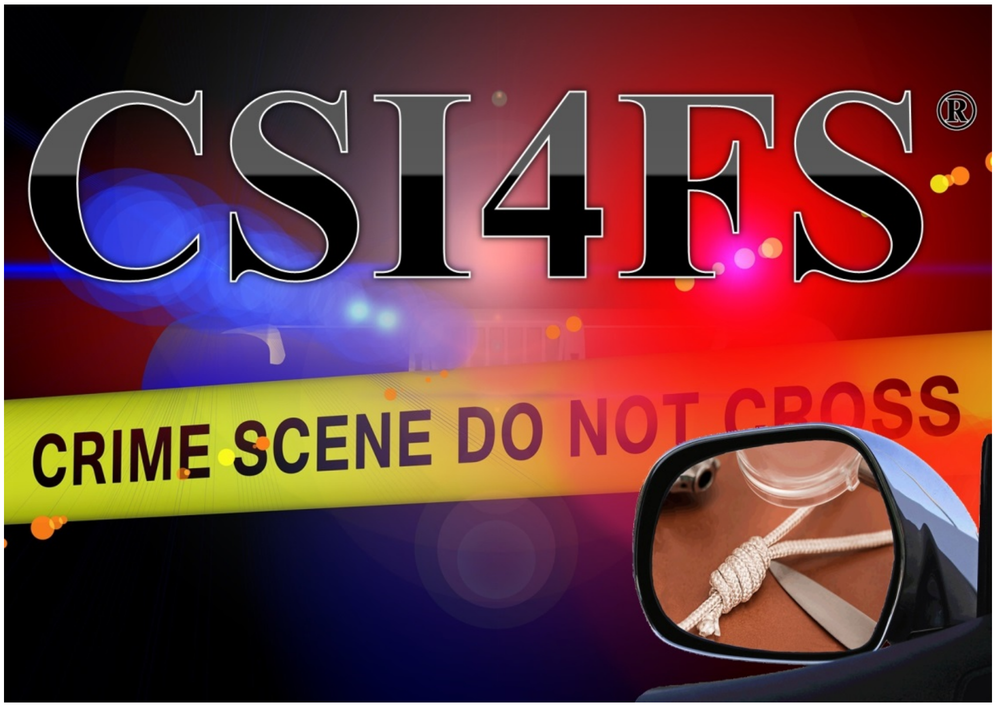


Figure 6. Final CSI4FS® logo.

#87560544. Figure 6 shows the final logo for CSI4FS®. The various elements used in the logo were composed from several free images available at Pixabay (www.pixabay.com). The Pixabay elements used in the CSI4FS® logo are both copyright- and royalty-free for personal and commercial use. Appendix A through Appendix F details the major C# script components used in this project (see page 74). Some proprietary C# script has been redacted.

CSI4FS® Preliminary Work

Following approval of the capstone project proposal on June 9, 2017, I spent the next few weeks selecting a name for the application and for the Web site that will both highlight this capstone project and allow online downloads of the application. I decided on CSI4FS® for the application name and www.csi4fs.com for the domain name and Web site. On July 1, 2017, the domain name was registered through www.007names.com but, when site hosting services were purchased for the Web site on October 5, 2017, the registration was transferred to www.bluehost.com. The domain is registered for a period of ten years, and the hosting services are contracted for a term of three years after which it will automatically renew for a further term of three years.

Several unexpected dead ends appeared over the course of the next few months. In May, 2017, I purchased two books I believed would be useful for developing the C# code to build the application. One book, Unity Games, contains comprehensive tutorials on using the Unity® game engine. The other book, Developing AR Games for iOS and Android, offered tutorials for creating AR applications using Unity® and Vuforia®. By September 13, it became clear that the latter of the two books was all but useless. It outlined information that was already known and was used in the creation of the marker-based application used as part of the capstone proposal. Indeed, by October 9, I realized that Vuforia® itself was designed solely for marker-based AR and would not aid the development of this capstone project. As a result, I looked for a different program to help build the AR application. One such program, Wikitude v7.1 (www.wikitude.com) that is used in high-end markerless AR applications had a trial version available, but the trial version was embedded with watermarks that made it unsuitable for this project, and the non-trial version was $2,935 and, so, cost-prohibitive. I ultimately abandoned the idea of using a program other than Unity® to create the AR application.

Coding for the application began on September 24. Several online videos were accessed which contained tutorials for using Unity® v5.5.4, working with C#, and creating simple games. The tutorials began with the creation of a simple game called Roll a Ball. Game creation was successful and the program ran well on my home computer. However, porting the game from the home computer to my smartphone, a Samsung Galaxy S4® (SGH-I337 with Android OS v5.0.1), was fairly complicated despite instructions that suggested the contrary. In order to successfully port the program, several non-intuitive steps had to be taken including downloading the Android Software Developer's Kit® (SDK®) from the Android Web site, renaming the Tools folder, and installing the new SDK® package to that folder. Further, there was a known problem with the Java Developer's Kit® (JDK®) v9.0 which came with the

Android SDK®, and I had to downgrade to JDK® v8.0.144 which worked. It was very frustrating trying to piece together seemingly disparate tidbits of information into a cohesive structure. After much stumbling about, I successfully ported the Roll a Ball game from the computer to the smartphone on October 2. Once the process was in place, several small games were created and successfully ported in quick succession from the computer to the Galaxy S4®.

Porting Programs from Computer to Smartphone

Porting, or transferring, a program from the computer to the smartphone is a 3-step process. In the first step, you ensure that your smartphone is recognized as a portable media device. In the second step, you must install two developer kits that work with Unity® and handle the communication between Unity® and the phone. You then tell Unity® where to find specific folders. In the third step, you port the program file to the phone. Steps one and two are done only once after which Unity® knows where the files are and how to locate them. The third step must be done each time you want to port a program from the computer to the phone.

Step One:

1) For the phone to be correctly recognized by the computer and, by extension, Unity®, you must install the correct USB driver. For the Samsung Galaxy S4® phone, I used the USB driver at http://developer.samsung.com/galaxy/others/android-usb-driver-for-windows.
2) Once the driver is installed, connect the phone to the computer using a USB cable. The cable must allow the computer to see the phone as a portable media

device so that files can be moved back and forth between the two. Not all USB phone cables do this (see Figure 7). In order to work properly, your phone must be in developer mode and you must enable USB debugging. On my Galaxy S4®, I went to Settings → More → Developer Options and enabled developer mode. Under Debugging, I checked the USB debugging box. When

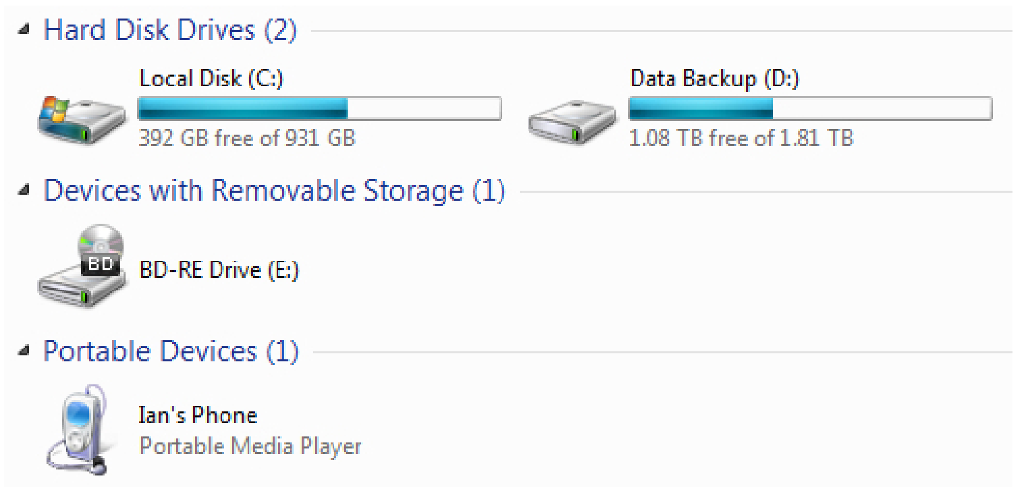
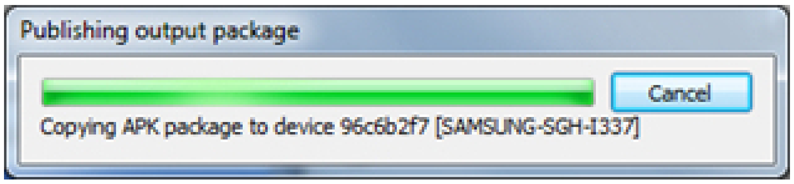


Figure 7. Phone is recognized as a Portable Media device.

the phone and driver are working together properly, the phone will be identified by type (in this case, a Samsung SGH-I337 (see Figure 8)), and you will get a message on the phone asking whether or not to allow USB debugging from the computer.


Publishing output package

Copying APK package to device 96c6b2f7 [SAMSUNG-SGH-I337]

Figure 8. Phone is identified as Samsung SGH-I337.

Step Two:

1) Now, you must install the Android Software Developer's Kit (SDK®, available at https://developer.android.com/studio/index.html). It is not necessary to download the entire Android Studio® package (see Android Studio® and Java Code, page 51) if you do not want to; you can chose to download only the SDK® package and it is entirely your choice whether to download one or the other. Either way, the default install settings are fine. Note: it may be necessary to disable your antivirus program during the install. Once installed:
    a. Open the SDK® manager.
    b. Under Tools, check all SDK® build tools for v21.1.2 and higher.
    c. Under Extras, check Google Play services.
    d. Click the install button and accept the licenses to begin the install.

    The SDK® software did not work properly at first, so I had to reinstall the standalone SDK® package. As a result, I had to rename the Tools folder in the existing SDK® folder and install the standalone SDK® package files to that existing SDK® folder.
    e. Once the install is complete, you can exit the SDK® software.
2) Next, you must install the Java Developer's Kit® (JDK®, available at http://www.oracle.com/technetwork/java/javase/downloads/index.html).
    a. This install is fairly straight forward.
    b. Accept the default settings.
    c. Once the install is complete, you can exit the JDK® software.

Now you must tell Unity® where to find those specific folders.

3) In Unity®, click Edit → Preferences → External Tools.
    a. Under Android → SDK®, browse to the location of the SDK® folder. This is usually found at C:\Users\<Username>\AppData\Local\ Android\Sdk – but could also be found at C:\Android\Sdk.
4) Under Android → JDK®, browse to the location of the JDK® install. This is usually found at C:\Program Files\Java\jdk<version>. Since different versions of the JDK® can be installed on the same computer, it is important to tell Unity® the specific version being used. I initially installed the most recent version (JDK® v9.0) but the software would not work properly with my Unity® install and smartphone, and so I downgraded to v8.0.144 which did work.

Step Three:

Now you can port the program from the computer to the smartphone.

1) In Unity®, click File → Build Settings.
    a. Under Scenes in Build, select the scene that you want to port to the phone. The current scene is used if no specific scene is selected.
2) Under Platform, select the platform upon which you want the scene to be mounted. In this case: Android. The Unity® logo appears next to the platform selected for the build, so if the Android platform is not currently selected, you must do so and click the Switch Platform button. The Unity® logo will appear next to the Android platform once it is properly selected. Note: If the Android

Plug-in is not yet on your computer, get it from: http://download.unity3d.com.

a. Click the Player Settings button. Under Other Settings → Identification, give the program a unique Bundle Identifier like: com.CSI4FS.Gyro where the last section is the name that displays in the application folder.
b. Click the Build and Run button – and the scene files will be compiled and the program file will be ported to the phone. You will need to provide a name for the program. Note that if the Build button is clicked instead of the Build and Run button, the program will be compiled for the Android platform, but the appropriate file will not be subsequently transferred to the phone which then would require that extra step to be done manually.
c. Once the program is successfully ported to the phone, it will appear in the Apps folder listed alphabetically alongside all the other apps on the phone.

Porting of the program is now complete.

Development of CSI4FS®

By October 5, I had sufficiently coded a scenario using Unity® so that an animated object was superimposed on whatever the smartphone camera was showing. Figure 9 shows this first real breakthrough for the project. The Unity® coding was sufficient to display an augmented reality character over a real-time scenario. The photo was taken in my home office using a digital camera and shows both a bed with a Disney® blanket and my hand-held smartphone displaying the AR character

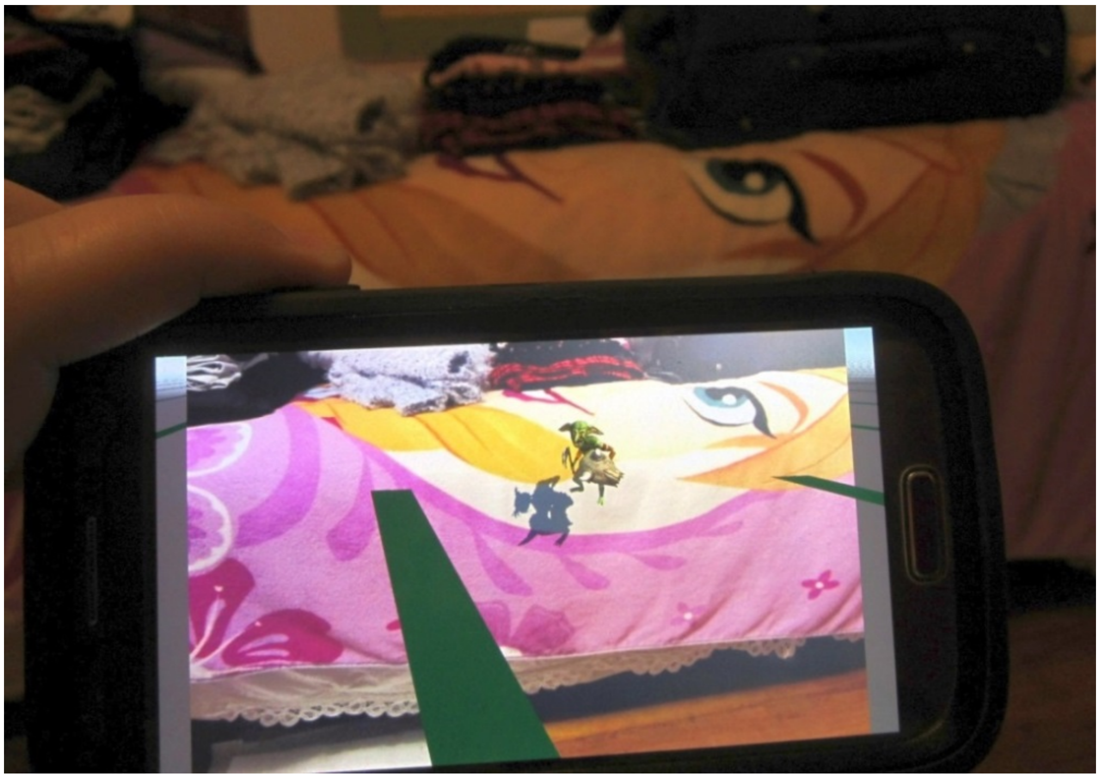


Figure 9. Character and shadow displayed on a smartphone.

superimposed on the scene. The green lines are part of a background grid system upon which the character is superimposed. At this point, two issues needed addressing: the green lines of the background grid system, and the fact that the character only appeared when the camera was facing north. That is to say, when the camera was panned around the room, the character disappeared from view and reappeared only when the camera returned to face north. This seemed to be an odd behavior and the reason it was happening was largely unknown.

In Figure 10, the green background grid still exists but has been sufficiently altered so as to appear invisible to the user. This required that the C# code be

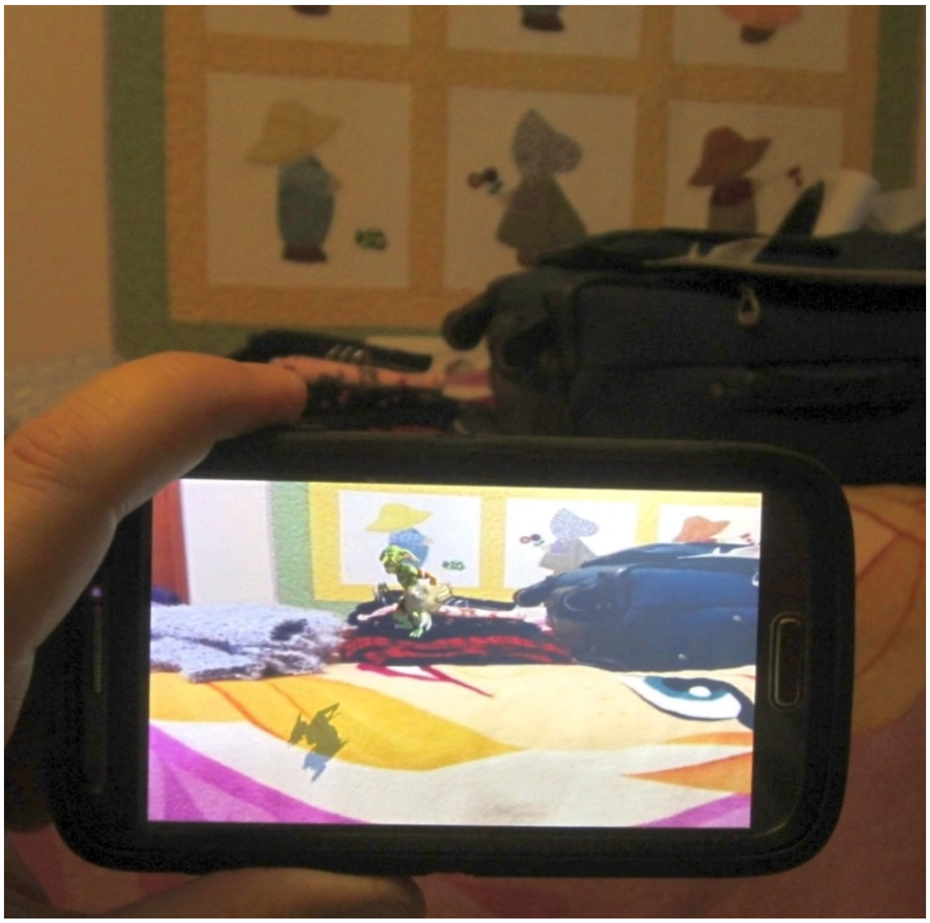


Figure 10. Aspect ratio issue.

altered. Specifically, two changes were made to the grid_base (see Appendix C) script: 1) a final "}" was placed at the end of the script, and 2) the code: input.worldPos.y/_GridSpacing was changed to: input.worldPos.z/_GridSpacing.

A new issue, however, arose. As you can see in Figure 10, when compared to the wall quilt hanging in the room where the photo was taken, the smartphone display appears to show the background images as being stretched horizontally. After some experimentation with coding changes to the x-axis, the truth was realized: it was the vertical aspect (y-axis) that was compressed making the quilt images appear squashed in the display. Corrections to the y-axis were made and Unity® was altered from a "free aspect" display to a 16:10 landscape display. By October 8, the correct aspect ratio was being correctly displayed (see Figure 11).

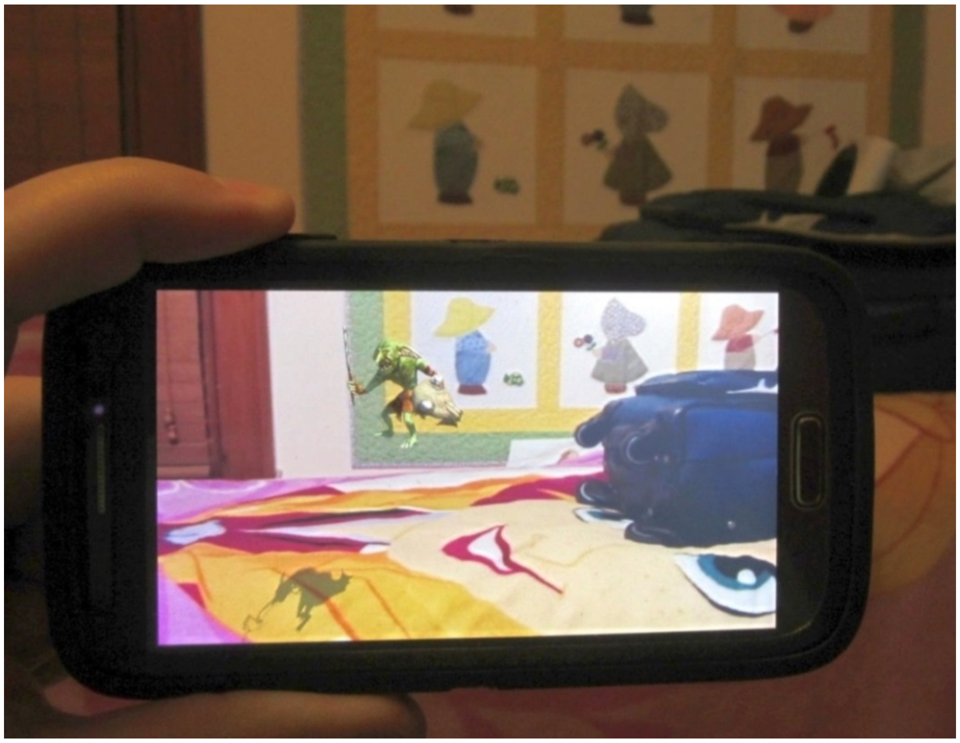


Figure 11. Aspect ratio corrected.

The next few weeks saw the C# code refined. Since it was imperative that I create objects for the AR project, and not use objects created by others, I downloaded and installed Blender (https://www.blender.org). Blender is a free and open source 3D creation software package that would allow me to create 3D objects and characters although no characters were actually created. Two weeks later, the C# code was altered sufficiently so that the AR character seen in the smartphone display in Figures 9, 10, and 11, no longer appeared only when the camera faced north. The character now appeared on the display regardless of the orientation of the camera. As the camera pans around, the character moves with the camera and stays in the display at all times. However, having done this, I questioned whether or not this was a good idea considering the overall vision of the AR application. If the goal of the application is to display AR objects in a real environment, then it is unlikely that those objects will move around with the camera as the user moves through the environment. So, as much as this might have seemed an interesting addition to the program, the feature was removed prior to beta-testing.

Android Studio® and Java Code

To be useful to the project, a touchable object was needed so the user can touch an object as part of the CSI simulation and get a response from the device. That response could be more information, another clue, or perhaps a pointer giving directions. I had difficulty getting a touchable object to display properly on the smartphone using only C# scripting – so, in January 2018, a new approach was taken to the creation of this function. I installed the full Android Studio® v2.3.3 package (see URL link, page 45). Unlike the Android SDK® package, the full Studio® package includes an Integrated Development Environment (IDE) that allows the user to create Android applications by directly manipulating the Java programming code. This is convenient since the Android operating system is a Linux variant and uses extensive Java code. I reasoned that if I could get a touchable object on the smartphone using a

Java script, it might be easier to simply port that Java code to C# rather than attempt to code it in C# from scratch. This proved to be an appropriate strategy.

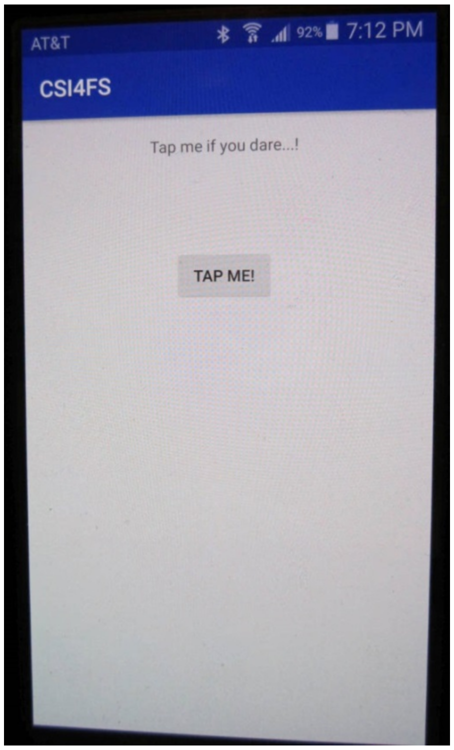


Figure 12. Phone prompts user to take action.

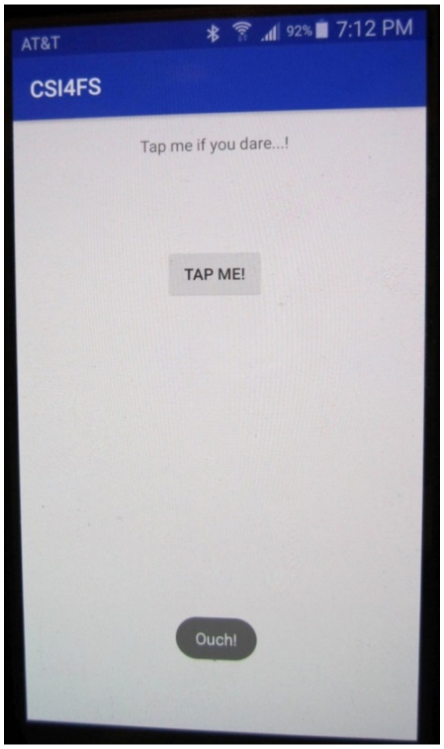


Figure 13. Phone responds to the action.

Including some experimentation, it took about two weeks to create the Java script – the results of which are shown in Figures 12 and 13. The Java code prompts the user to take action and then responds to that action. In this specific script, the user is prompted to tap the button displayed on the smartphone. Indeed, the user is taunted into tapping the button. Once the button is tapped, the phone responds appropriately and humorously. The response remains on the screen for about five seconds and then disappears. When the button is again tapped, the response reappears. While this prompt and response script seems simple enough, it was a necessary step in moving the project forward. Armed with this Java script, I recoded the script for C#.

Several days later and with some modifications, I created a C# script for a series of five cubes that change color when touched. The results are displayed in Figures 14 and 15. As with the earlier Java code, the phone responds appropriately when an object is touched. With this C# script, the five cubes are all one color when the program begins, but when a cube is touched it changes color – specifically: blue, red, gray, green, and black (from left to right). In Figure 15, the first, fourth, and fifth cubes are being touched. The cubes change color only for as long as they are touched. Once you remove your finger from the screen, the cube immediately returns to its default color. This new C# script was the basis on which I created appropriate touchable objects for CSI4FS®.

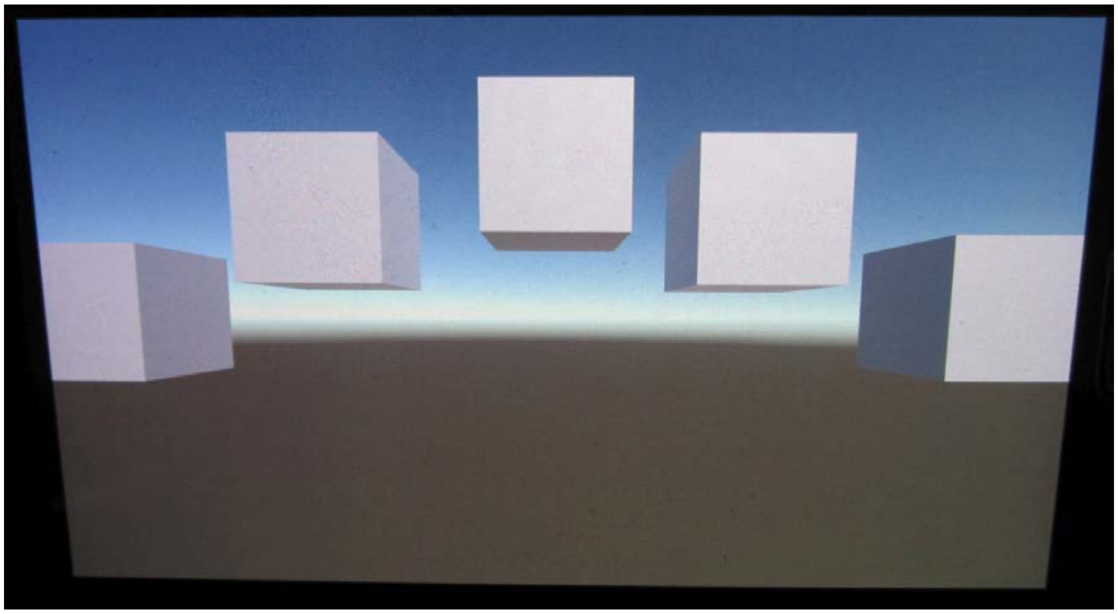


Figure 14. Five cubes are displayed on the smartphone.

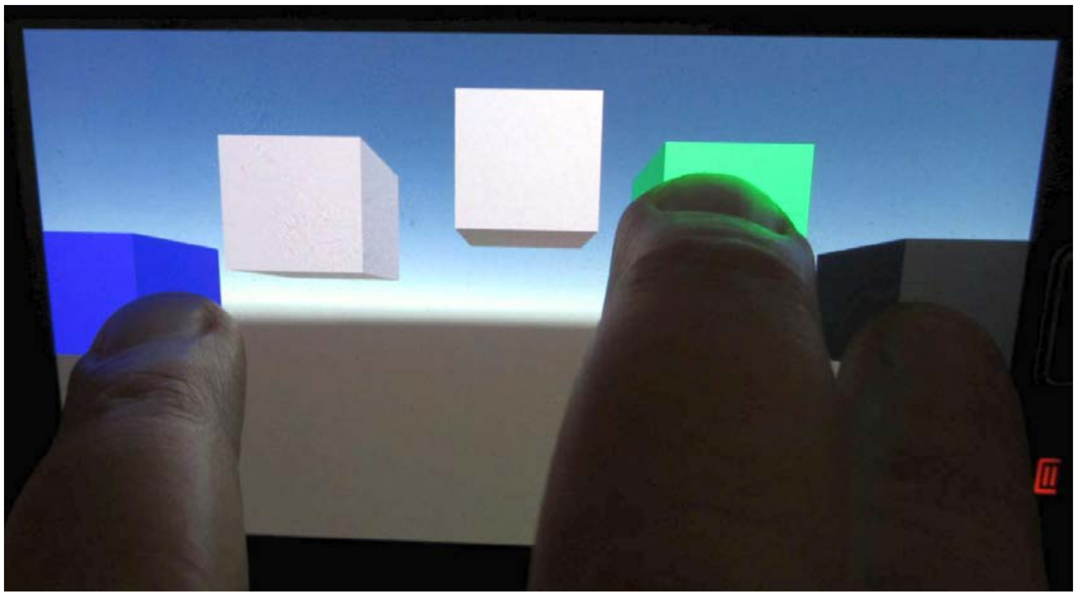


Figure 15. The cubes change color when touched.

By January 27, three touchable objects – a USB flash drive, a knife, and a camera – had been created and placed in the scene. These objects appear when the camera faces north but now, after five months of working with Unity®, it was no longer a mystery. The Unity® plane upon which the real-world view is superimposed stands perpendicular to the background grid system and the built-in Unity® camera sits behind that plane. In other words, Unity®'s own built-in camera faces north which is why the objects appear when the phone faces north. Figure 16 shows the three touchable objects on the phone's display screen in my lab at work.

Creating touchable objects superimposed on a real-world view was a more difficult process than with the earlier touchable cubes. Each object had to have a box collider component added. A box collider is Unity®'s way of determining what is or

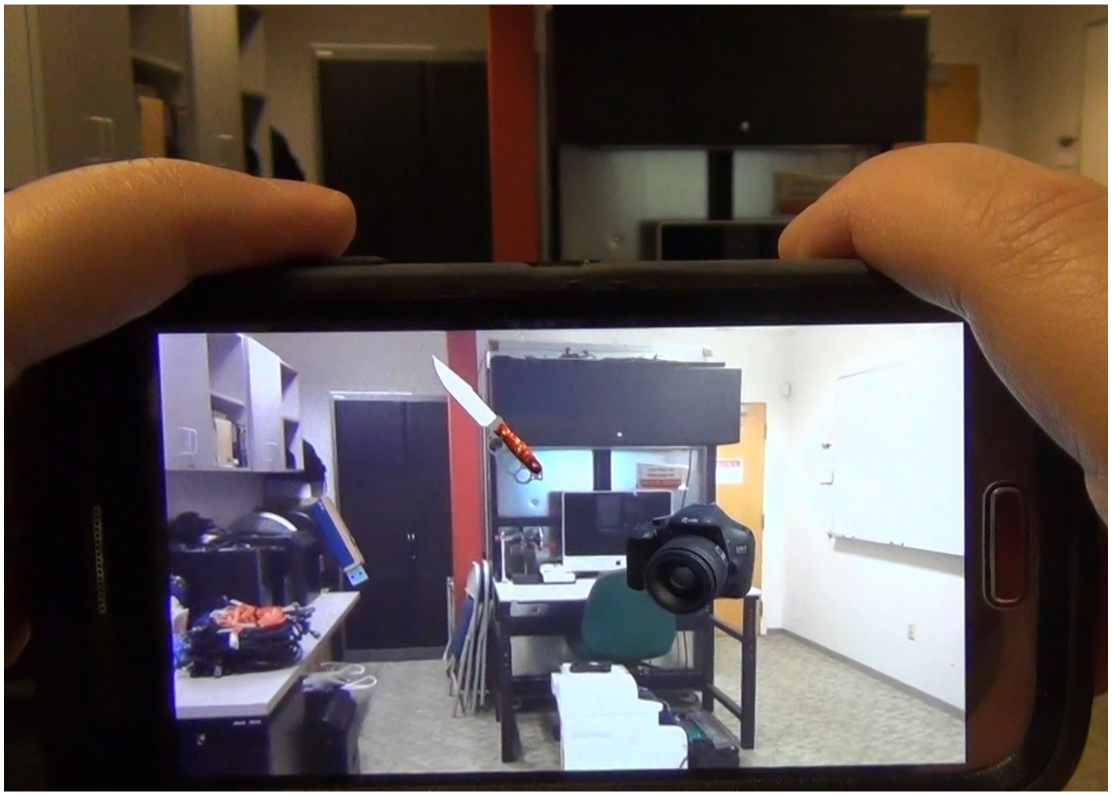


Figure 16. Three touchable objects displayed on the phone.

is not a solid object that can be manipulated. Unity® needed to see the objects as solid so that they could be touched and give a response. The cubes were created with box colliders already built in, but the objects were not – and it took a while to come to that realization. Prior to adding a box collider component, the objects simply disappeared from view when touched, but with a box collider component attached they now reacted to touch and gave a response. Like the cubes, they were initially programmed to change color when touched – but, now that the objects were responding to touch, they could be programmed to give a more appropriate project-based response.

February 3 saw yet another milestone reached. New C# code was written to replace the earlier code that caused the touchable objects to change color. The new code allowed text information to appear when the object was touched. In turn, this paved the way for the creation of an entire text file containing several lines of information which would appear in succession when the objects were touched. Within a week, a text file was created and successfully displayed on the smartphone. With each touch of the object, new information appeared so that by touching the same object multiple times, the user got an entire dialogue of useful information. So long as the object stayed touched, the text information remained on the display screen. For all practical purposes, CSI4FS® was complete and ready for beta testing.

Crime Scene Investigation Protocols

As early as March, 2017, I made contact with two members of the local Police Department (PD) both of whom are crime scene investigators and members of the PD's Forensic Investigation Unit. Both are also adjunct professors at a mid-Atlantic university and teach courses in crime scene investigation for graduate forensic science students. These two became my subject matter experts (SMEs). Each became aware of this project during casual conversation with me, but on November 28, both met with me to discuss crime scene investigation and what aspects of an investigation would be most appropriate in a smartphone application. In order for me to build a useful application, it was necessary to understand the six steps to processing a crime scene. The SMEs have adapted these six steps from Gardner and Bevel (2009). Each step must be taken sequentially in order to preserve evidence suitable for analysis and later probative purposes in court:

1. **Assessing**. Although this is the first step of an investigation, it is a continuous and ongoing step. First, a perimeter is established to bring the crime scene under control. Often, a second perimeter is established within the first one so that police and other emergency responders who have a purpose at the scene do not arbitrarily enter or damage the scene further. Factors such as the size of the crime scene, lighting, and environmental conditions are all taken into consideration. Assessment then involves determining the full scope of the crime scene, ensuring crime scene integrity, ensuring that any search warrants are correct and signed, developing an appropriate team approach, determining an appropriate search technique, and ensuring the safety of the crime scene investigation team.
2. **Observing**. At this step, the crime scene investigator looks for persons and objects in plain sight. The investigator carefully examines the scene and observes details and content that may be altered by subsequent crime scene processing (such as the order of bedding on top of a victim) or altered by time (such as ice cubes in a glass). The investigator will often locate a vantage point near the perimeter and, simply, stop and look – and then document via notes or audio recording, everything that is seen.
3. **Documenting**. This is the step that is, perhaps, most familiar to the general public from watching television. Documenting involves the creation of photographs, video recordings, and sketches that show the full and complete

context of the crime scene as found by the investigative team. It also includes notes prepared by the investigator during the initial scene observation. Order and purpose are most important, and nothing is done out of sequence. Photographs are often taken in groups of three: an overall photo that captures the condition of the scene before any alteration can occur, a photo that establishes evidence to show its context within the scene, and a close-up photo to show detail that may not be otherwise evident. The investigator will often use a marker containing a ruler to frame evidence-establishing and close-up photographs. Sketches, similar to floor plans, are often drawn to map the crime scene and show the relationship of objects in a room with objects in other rooms.

4. **Searching**. Searching the crime scene is an intrusive process and is done only after the scene context is documented. Searching ensures that all evidence and details are noted, and critical observation plays an important role throughout this step. Objects may be found beneath, behind, or inside other objects, or in positions where easy observation is not possible. Crime scene searches are conducted in multiple iterations to ensure that all evidence is located and nothing is missed.
5. **Collecting**. Collecting evidence is an intrusive process. Evidence must be collected without destroying or damaging that evidence since improperly collected evidence can prevent or cause problems with its subsequent analysis. Collection must be done with the forensic analyst in mind and evidence must

be secured in the most appropriate method possible. The only exception that allows collection to be done prior to documentation is if the evidence is fragile and is prone to significant alteration as crime scene processing unfolds – such as in the case of a shoe mark in dirt as a storm is imminent, or a small fiber on the floor at the crime scene. When fragile evidence is collected, documentation must take place immediately following the collection. The onus is on the investigator to know what kinds of evidence are possible at any given crime scene.

6. **Analyzing**. Finally, analysis of the crime scene completes the process. Such on-scene analysis includes examining bullet trajectories or defining the origin of a blood spatter pattern. It can also include processing surfaces with various chemicals for fingerprints. On-scene analysis is always intrusive and is done only after all other evidence items are documented and collected. During analysis, the investigator could encounter new evidence and must then be prepared to backtrack and reprocess the scene to ensure that the new evidence is fully documented.

When they teach crime scene investigation skills as part of their duties as adjunct faculty, the SMEs first introduce the class to a scene. Students are asked to look around the scene, pick out evidence, and attempt to exploit it for its probative value. Two questions are asked: 1) What hints and clues are there? and 2) How does this help us solve the crime? These questions form the basis for the class instruction, and the SMEs suggested that this process be built into the application. They further suggested introducing “red herrings” into the application which would lead to wrong choices – which would ultimately help students focus on correct assessment and analysis. The conversation also included a suggestion for using drop-down lists with variables so that users could choose different paths.

Publishing and Beta Testing the Application

Creating the single application CSI4FS® was a matter of writing and testing (and often rewriting) several separate scripts each of which tackled a specific task. In some cases, two or more smaller scripts were combined to create a separate small program which was later integrated into the final application. Each separate script and small program was then combined with the others into one program – all the time hoping that the various scripts would not conflict with each other. The various C# scripts used as major components in the design of CSI4FS® are listed in Appendix A through F (proprietary code is redacted). Scripts for the minor components are not listed. The major component scripts are described below along with the specific function that each performs:

- **Find_Camera**: this script locates and controls the smartphone’s camera. It automatically adjusts the displayed image so as to remain upright even if the phone is rotated (see Appendix A).
- **Gyroscope**: this script checks for the existence of the phone’s internal gyroscope and monitors rotation along the x, y, and z axes in real time as the

phone moves. It locks in place any displayed AR objects so that they remain upright and remain correctly oriented with respect to each other when the phone moves or is panned around (see Appendix B).

- **Grid_Base**: this script creates the plane (either visible or invisible) on which an AR object exists (see Appendix C).
- **Touch_Me**: this script allows an AR object to accept a touch input on the device. The response from the touch input depends on the script linked to it (see Appendix D).
- **Text_on_Touch**: this script causes an AR object to display text when it is touched. The text can be either a single line or multiple lines. For multiple lines, successive touches on the object will display each new line of text (see Appendix E).
- **Multi_Touch**: this script causes a single AR object to display when the simulation starts and, after successive touches, disappears and calls the next AR object to appear. Similarly, the second object then calls the third. When the third object has finished giving information, it also disappears (see Appendix F).

Preparing CSI4FS® for publication was a reasonably simple 2-step process: 1) create a signed Android application package (APK); and 2) upload the signed APK to the Google Play Store. To publish and beta test CSI4FS®, a developer account was required (https://play.google.com/apps/publish/signup). There was a $25.00 lifetime fee to register and create the account. Once CSI4FS® was uploaded, a beta tester could download the application directly from the Google Play Store to their Android device and run it.

Step One:

The signed APK is created in Unity®. First, the Build Settings are altered to prepare for the creation of the signed APK. Open Unity® and select File → Build Settings:

1) Click the Player Settings button.
    a. Enter a Company Name and Product Name for the application.
    b. Select a default icon for the application. This is the application logo that will be displayed on the phone in the Apps folder.
    c. Under Other Settings.
        i. Verify the Bundle Identifier, Version number, Bundle Version Code, and Minimum API Level for the build. The lower the API level, the greater the number of phones that can access the application – although some features of the application may not work on older phones. I selected API level 9; my own Galaxy S4® uses API level 21.
    d. Under Publisher Settings, check Create New Keystore. A keystore is simply a database of keys. Like any database, it can hold and keep track of multiple keys.
        i. Click Browse Keystore, select a folder, and create a name for

the keystore.

ii. Enter and then enter and confirm a Keystore password.

iii. Under Key, select: Create an alias. The alias can be anything. The alias is used to keep track of keys for different projects.

iv. Enter and then confirm an alias password.

v. Complete the Certificate information.

e. Once the keystore and key are created, you must then apply the specific key by selecting your key from the alias pull-down menu.

2) Click the Build button. Enter a location for the build and Unity® will prepare the signed APK. This is the file that is uploaded to the Google Play Store. You can close Unity® once the APK is created.

Step Two:

The APK file is now ready for upload to the Google Play Store.

1) Log in to the Google Play Console at: https://developer.android.com/distribute/console/index.html. Select Store listing from the menu in the left column if you are not already on that page.

a. Under Product details, create a title for the application.

i. Complete the application's short and full descriptions.

b. Under Graphic Assets, select and upload several images of the application. You will need a minimum of five images of your application – and you will need them whether or not your application is for a smartphone only:

            1. A main image (I used the CSI4FS® logo, see page 40).
            2. An icon image at 512 x 512 pixels.
            3. A feature graphic image at 1024 x 500 pixels.
            4. An image for Tablet at 7.5 inches x 4.7 inches.
            5. An image for Tablet at 10 inches x 7 inches.

    c. Under Categorization, select whether your program is an application or a game.
        i. Select the Category.
        ii. You need a Content rating, but that step is done a bit later on.
    d. Complete the Contact details.
    e. If a Privacy Policy is required, you must link to a Web site where details of the policy are available. Because CSI4FS® uses the phone's built-in camera, a privacy policy was required. At this point, is was sufficient to enter a location for the policy even though the Web site for CSI4FS® was not yet available.
    f. Save a draft copy of this form.

2) Select App releases from the menu in the left column.
    a. Select Production, Alpha, or Beta releases for the application as appropriate.
    b. Opt-in or opt-out of Google Play App Signing (I opted out).
    c. Browse to find your signed APK file from Step One. Once the upload is complete:

i. Modify the release name if desired.

ii. Enter release notes if desired.

3) Select Content rating from the menu in the left column.

   a. Enter and confirm a valid email address.

   b. Complete the rating questionnaire → Save questionnaire.

   c. Calculate rating → Apply the ratings. Ratings will be applied to the Store listing section.

4) Select Pricing and distribution from the menu in the left column.

   a. The application is free by default. If you want to charge for the application, a merchant account is required. CSI4FS® is being distributed free of charge.

   b. Select the countries in which your application will be available. During beta testing, only the U.S. was selected for CSI4FS®.

   c. Complete the child-directed and ads section.

   d. Under Consent, you must acknowledge that you agree with the content guidelines and that the application conforms to U.S. export laws.

Once each of these four main sections (APK upload, Store listing, Content rating, and Pricing and distribution) is complete, the application will be available for publishing. A check mark will appear next to each of these items in the menu in the left column to confirm that each section has been successfully completed. You are then directed to review the package and, finally, publish it. The first beta version of CSI4FS® was published on February 14.

Several refinements were made to the beta version of the application during beta testing. Most notably, I rewrote part of the C# script so that only one object appeared on the display at a time. Once all the text information for the first object had been displayed, that first object disappeared and a second object became visible. Once all the text information for the second object had been displayed it, too, disappeared and a third object became visible.

During beta testing, several suggestions for immediate improvement were made and acted upon, and there were also suggestions for long-term future considerations for the application. In the Impact section (see page 36), these immediate and future considerations are noted.

REFERENCES

Ahmed, S., & Dann, S. (2015). The virtual future of education and training. *The Bulletin of the Royal College of Surgeons of England*, *97*(10), 431-432.

Agarwal, A., Gupta, M., Gupta, S., & Gupta, S. C. (2011). Systematic digital forensic investigation model. *International Journal of Computer Science and Security (IJCSS)*, *5*(1), 118-131.

Ausburn, L. J., & Ausburn, F. B. (2010, December). Spheres of reality: A conceptualization of desktop virtual environments in career and technical education and an implementation training model. In *Proceedings of the National Career and Technical Education Research and Professional Development Conference, Las Vegas, NV.* Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.454.7019&rep=rep1&type=pdf.

Baker, T. E. (2009). Police Criminalistics: Learning Modalities and Evaluation. *Forensic Examiner*, *18*(3), 52-56.

diCerbo, F., Girardello, A., Michahelles, F., & Voronkova, S. (2010, November). Detection of malicious applications on Android OS. In *International Workshop on Computational Forensics* (pp. 138-149). Springer Berlin Heidelberg.

Doran, L., & Davis, M. R. (2016). As Pokémon craze grows, educators weigh game's value. *Education Week, 35*(37).

Dorninger, P., & Pfeifer, N. (2008). A comprehensive automated 3D approach for building extraction, reconstruction, and regularization from airborne laser scanning point clouds. *Sensors*, *8*(11), 7323-7343.

Drake, S. A., Giardino, E., Giardino, A., & Nolte, K. (2016). Leadership decisions influencing medicolegal death investigation: “We wear a lot of hats.” *Forensic Science Policy & Management: An International Journal*, *7*(3-4), 51-57. doi: 10.1080/19409044.2015.1138006

Dugdale, J., Pavard, B., Pallamin, N., el-Jed, M., & Maugan, L. (2004, May 3-4). Emergency fire incident training in a virtual world. *Proceedings ISCRAM*, Brussels, 167-172.

Forsythe, C. (2004). The future of simulation technology for law enforcement: Diverse experience with realistic simulated humans. *FBI Law Enforcement Bulletin*, *73*(1), 19-23.

Gaba, D. M. (2004). The future vision of simulation in health care. *Quality & Safety in Health Care*, *13*(Suppl.1), i2-i10. doi:10.1136/qshc.2004.009878.

Gardner, R. M., & Bevel, T. (2009). Practical crime scene analysis and reconstruction. V. J. Geberth (Ed.). Boca Raton, FL: CRC Press.

Gee, A. P., Escamilla-Ambrosio, P. J., Webb, M., Mayol-Cuevas, W., & Calway, A. (2010). Augmented crime scenes: Virtual annotation of physical environments for forensic investigation. In *Proceedings of the 2nd ACM Workshop on Multimedia in Forensics, Security and Intelligence* (pp. 105-110). ACM.

Giova, G. (2011). Improving chain of custody in forensic investigation of electronic digital systems. *International Journal of Computer Science and Network*

Security, *11*(1), 1-9.

Glover, W., & Lygeros. J. (2004). A stochastic hybrid model for air traffic control simulation. *Hybrid Systems: Computation and Control, 2993*, 372-386.

Gordon, J. A., Wilkerson, W.M., Shaffer, D.W., & Armstrong, E.G. (2001). “Practicing” medicine without risk: Students’ and educators’ responses to high-fidelity patient simulation. *Academic Medicine, 76*(5), 469-472.

Grantcharov, T. P., Kristiansen, V. B., Bendix, J., Bardram, L., Rosenberg, J., & Funch-Jensen, P. (2004). Randomized clinical trial of virtual reality simulation for laparoscopic skills training. *British Journal of Surgery*, *91*(2), 146-150.

Johnson, L., Adams, S., & Cummins, M. (2012). *The NMC horizon report: 2012 higher education edition*. The New Media Consortium. Retrieved from http://bibliotecadigital. org/bitstream/001/563/1/2012-Horizon-Report-HE.pdf

Khan, R., Aydin, A., Khan, M. S., Dasgupta, P., & Ahmed, K. (2015). Simulation-based training for prostate surgery. *British Journal of Urology International*, *116*(4), 665-674.

King, J. M., & Reising, D. L. (2011). Teaching advanced cardiac life support protocols: The effectiveness of static versus high-fidelity simulation. *Nurse Educator*, *36*(2), 62-65.

Lamkin, P. (2017, March 14). Best VR headsets 2017: HTC Vive, Oculus, PlayStation VR compared. *Wearable*. Retrieved from https://www.wareable.com/vr/best-vr-headsets-2017.

Lee, H. C. & Pagliaro, E. M. (2013). Forensic evidence and CSI. *Journal of Forensic Investigation. 1*(2):5.

Lee, K. (2012). Augmented reality in education and training. *TechTrends, 56*(2), 13-21.

Lenhart, A. (2015, April 9). Mobile access shifts social media use and other online activities. *Pew Research Center*. Retrieved from http://www.pewinternet.org/2015/04/09/mobile-access-shifts-social-media-use-and-other-online-activities/.

Liarokapis, F., Mourkoussis, N., White, M., Darcy, J., Sifniotis, M., Petridis, P., … & Lister, P. F. (2004). Web3D and augmented reality to support engineering education. *World Transactions on Engineering and Technology Education, 3*(1), 11-14.

Ludwig, A., Fraser, J., & Williams, R. (2012). Crime scene examiners and volume crime investigations: an empirical study of perception and practice. *Forensic Science Policy & Management: An International Journal*, *3*(2), 53-61. doi: 10.1080/19409044.2012.728680

Ma, M., Zheng, H. & Lallie, H. (2010). Virtual reality and 3D animation in forensic visualization. *Journal of Forensic Sciences, 55*(5), 1227-1231.

Macedonia, M. (2002). Games, simulation, and the military education dilemma. In *Internet and the University: 2001 Forum* (pp. 157-167).

Merrill, M. D. (2000). Write your dissertation first and other essays on a graduate education. Retrieved from http://mdavidmerrill.com/Papers/Graduate Education.PDF.

Miano, J. (1999). *Compressed image file formats: Jpeg, png, gif, xbm, bmp*. Reading, MA: Addison-Wesley Longman, Inc.

Moon, M. (2016, August 1). Pokémon Go hits 100 million downloads. *Engadget*. Retrieved from https://www.engadget.com/2016/08/01/Pokemon-go-100-million-downloads/.

National Research Council. (2009). *Strengthening forensic science in the United States: A path forward*. National Academies Press. Retrieved from https://www.ncjrs.gov/pdffiles1/nij/grants/228091.pdf.

O'Rourke, M. (2016). Our new Pokémon overlords. *Risk Management*, *63*(6), 48.

Page, R. L. (2000). Brief history of flight simulation. *SimTecT 2000 Proceedings*, 11-17.

Palfrey, J. G., & Gasser, U. (2013). *Born digital: Understanding the first generation of digital natives*. Basic Books.

Poelman, R., Akman, O., Lukosch, S., & Jonker, P. (2012). As if being there: Mediated reality for crime scene investigation. *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work – CSCW '12.*

Rodgers, D. L., Securro Jr, S., & Pauley, R. D. (2009). The effect of high-fidelity simulation on educational outcomes in an advanced cardiovascular life support course. *Simulation in Healthcare*, *4*(4), 200-206.

Rosen, K. R. (2008). The history of medical simulation. *Journal of Critical Care*, *23*, 157-166.

Rothbaum, B. O., Hodges, L., Smith, S., Lee, J. H., & Price, L. (2000). A controlled study of virtual reality exposure therapy for the fear of flying. *Journal of Consulting and Clinical Psychology, 68*(6), 1020-1026.

Schofield, D. (2009). Animating evidence: Computer game technology in the courtroom. *Journal of Information Law and Technology, 1*, 1-21.

Seymour, N. E., Gallagher, A. G., Roman, S. A., O'Brien, M. K., Bansal, V. K., Andersen, D. K., & Satava, R. M. (2002). Virtual reality training improves operating room performance: Results of a randomized, double-blinded study. *Annals of Surgery, 236*(4), 458-464.

Singhal, S., Bagga, S., Goyal, P., & Saxena, V. (2012). Augmented chemistry: Interactive education system. *International Journal of Computer Applications*, *49*(15).

Wasko, C. (2013). What teachers need to know about augmented reality enhanced learning environments. *TechTrends*, *57*(4), 17-21.

Wayne, D. B., Didwania, A., Feinglass, J., Fudala, M. J., Barsuk, J. H., & McGaghie, W. C. (2008). Simulation-based education improves quality of care during cardiac arrest team responses at an academic teaching hospital: A case-control study. *Chest Journal*, *133*(1), 56-61.

West, C., Slatin, C., Sanborn, W., & Volicer, B. (2009). Computer-based simulation in blended learning curriculum for hazardous waste site worker health and safety training. *International Journal of Information and Communication*

Technology Education. 5(1), 62-73.

Wethal, T. (2010). Beyond books & lecture: Students and law enforcement get hands-on experience in the Marshall University crime scene house. *Law Enforcement Technology, March*, 46-50.

Xu, F. (2012). Crime scene reconstruction based on virtual reality. *Journal of Criminalistics and Law, 17*(3), 149-160.

Yang, W., Xiao, X., Andow, B., Li, S., Xie, T., & Enck, W. (2015, May). Appcontext: Differentiating malicious and benign mobile app behaviors using context. In *Software Engineering (ICSE), 2015 IEEE/ACM 37th IEEE International Conference on* (Vol. 1, pp. 303-313). IEEE.

Yuen, S., Yaoyuneyong, G., & Johnson, E. (2011). Augmented reality: An overview and five directions for AR in education. *Journal of Educational Technology Development and Exchange*, *4*(1), 119-140.

Zak, E. (2014). Do you believe in magic? Exploring the conceptualization of augmented reality and its implications for the user in the field of library and information science. *Information Technology and Libraries (Online)*, *33*(4), 23-50.

APPENDICES

Appendix A – C# Script for Find_Camera

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class Find_Camera : MonoBehaviour
{

	private bool camAvailable;
	private WebCamTexture backCam;
	private Texture defaultBackground;

	public RawImage background;
	public AspectRatioFitter fit;

	//use this for initialization
	private void Start()
	{
		//check for front and back camera
		defaultBackground = background.texture;
		WebCamDevice[] devices = WebCamTexture.devices;

		if (devices.Length == 0)
		{
			Debug.Log ("No camera detected");
			camAvailable = false;
			return;
		}

		for (int i = 0; i < devices.Length; i++)
		{
			if (!devices [i].isFrontFacing)
			{
		backCam = new WebCamTexture (devices [i].name, Screen.width, Screen.height);
			}
		}

		if (backCam == null)
		{
			Debug.Log ("Unable to find back camera");
			return;
		}

		backCam.Play ();
		background.texture = backCam;
```

```
                camAvailable = true;
        }

        //update is called once per frame
        private void Update()
        {
                if (!camAvailable)
                return;

                float ratio = (float)backCam.width / (float)backCam.height;
                fit.aspectRatio = ratio;

                float scaleY = backCam.videoVerticallyMirrored ? -1f : 1f;
                background.rectTransform.localScale = new Vector3 (1f, scaleY, 1f);

                int orient = -backCam.videoRotationAngle;
                background.rectTransform.localEulerAngles = new Vector3 (0, 0, orient);
        }
}
```

Appendix B – C# Script for Gyroscope

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class Gyroscope : MonoBehaviour
{
	//Gyroscope
	private Gyroscope gyro;
	private GameObject cameraContainer;
	private Quaternion rotation;

	//Camera
	private WebCamTexture cam;
	public RawImage background;
	public AspectRatioFitter fit;

	private bool arReady = false;

	//use this for initialization
	private void Start()
	{
		//Check to see if both services are supported
		//Gyroscope
		if (!SystemInfo.supportsGyroscope)
		{
			Debug.Log ("Device does not have a Gyroscope");
			return;
		}

		//Camera
		for (int i = 0; i < WebCamTexture.devices.Length; i++)
		{
			if (!WebCamTexture.devices[i].isFrontFacing)
			cam = new WebCamTexture (WebCamTexture.devices[i].name,
			Screen.width, Screen.height);
			break;
		}

		//If no camera is found, exit the program
		if (cam == null)
		{
			Debug.Log ("Device does not have a back camera");
			return;
		}

		//Both services are supported, enable them
		cameraContainer = new GameObject ("Camera Container");
		cameraContainer.transform.position = transform.position;
```

```
                transform.SetParent (cameraContainer.transform);

                gyro = Input.gyro;
                gyro.enabled = true;
                cameraContainer.transform.rotation = Quaternion.Euler (90f, 0, 0);
                rotation = new Quaternion (0, 0, 1, 0);

                cam.Play ();
                background.texture = cam;

                arReady = true;
        }

        //update is called once per frame
        private void Update()
        {
                if (arReady)
                {
                        //Update Camera
                        float ratio = (float)cam.width / (float)cam.height;
                        fit.aspectRatio = ratio;

                        float scaleY = cam.videoVerticallyMirrored ? -1f : 1f;
                        background.rectTransform.localScale = new Vector3 (1f, scaleY, 1f);

                        int orient = -cam.videoRotationAngle;
                        background.rectTransform.localEulerAngles = new Vector3 (0, 0, orient);

                        //Update Gyroscope
                        transform.localRotation = gyro.attitude * rotation;
                }
        }
}
```

Appendix C – C# Script for Grid_Base

```
Shader "Grid"
{
        Properties
        {
                _GridThickness("Grid Thickness", Float) = 0.02
                _GridSpacing("Grid Spacing", Float) = 10.0
                _GridColor("Grid Color", Color) = (0.5, 0.5, 1.0, 1.0)
                _OutsideColor("Color Outside Grid", Color) = (0.0, 0.0, 0.0, 0.0)
        }
                SubShader
                {
                Tags
                {
                "Queue" = "Transparent"
                // draw after all opaque geometry has been drawn
                }
                Pass
                {
                ZWrite Off
                // don't write to depth buffer in order not to occlude other objects
                Blend SrcAlpha OneMinusSrcAlpha
                // use alpha blending

        CGPROGRAM
                #pragma vertex vert
                #pragma fragment frag

                uniform float _GridThickness;
                uniform float _GridSpacing;
                uniform float4 _GridColor;
                uniform float4 _OutsideColor;

                struct vertexInput
                {
                        float4 vertex : POSITION;
                };
                struct vertexOutput
                {
                        float4 pos : SV_POSITION;
                        float4 worldPos : TEXCOORD0;
                };

                vertexOutput vert(vertexInput input)
                {
                        vertexOutput output;

                        output.pos = mul(UNITY_MATRIX_MVP, input.vertex);
                        output.worldPos = mul(unity_ObjectToWorld, input.vertex);
                        // transform input.vertex from object coordinates to world coordinates;
                        return output;
                }

                float4 frag(vertexOutput input) : COLOR
                {
                        if (frac(input.worldPos.x / _GridSpacing) < _GridThickness ||
                        frac(input.worldPos.z / _GridSpacing) < _GridThickness) {
                        return _GridColor;
                }
                else
                {
                        return _OutsideColor;
                }
                }

        ENDCG
        }
        }
}
```

Appendix D – C# Script for Touch_Me

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Touch_Me : MonoBehaviour
{
   public LayerMask touchInputMask;

   private List<GameObject> touchList = new List<GameObject>();
   private GameObject[] touchesOld;
   private RaycastHit hit;

   // update is called once per frame
   void Update()
   {

   // compile code for PC use only
#if UNITY_EDITOR
      if (Input.                                                            Up(0))
      {
         touche
         touchL
         touchL

            Ray

            if (P
            {
               G
               to

               if
               {                Proprietary Code, Script Redacted


               }
               if
               {


               }
               if
               {


               }
            }

         foreach (GameObject g in touchesOld)
```

```
            {
                if (!touchList.Contains(g))
                {
                    g.                                                                                  equireReceiver);
                }
            }
        }
#endif

    // compile c
        if (Input.t
        {
            touche
            touchL
            touchL

            foreach
            {
                Ray

                if (P
                {
                    G
                    to

                    if          Proprietary Code, Script Redacted
                    {


                    }
                    if
                    {


                    }
                    if
                    {


                    }
                    if
                    {


                    }
                    if
                    {


                    }
                }
```

```
      }

      fore
      {
        if
        {
                                                      Receiver);
        }
      }
    }
  }
}
```

Proprietary Code, Script Redacted

Appendix E – C# Script for Text_on_Touch

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class Text_on_Touch : MonoBehaviour
{
    private void Update()
    {
        if
        {

        }

        //
        if
        {

        }
        {

        }
        {

        }   Proprietary Code, Script Redacted
    }

    priva
    {
        V
        R                                        Touch);

        R
        if                                       )
        {









            }
          }
        }
    }

    private void ShowTextOnTouchDown()
    {
        Vector3 rawTouch = GetTouchPosition();
        R

        R
        if
        {




        }
        {

        }
    }

    publ
    {
        //                                                          device.
#if UN
```

Proprietary Code, Script Redacted

```
#else
        re
#endif
    }

    #reg
    publ
    {
        b
#if UN

#else
        v
#endif
        re
    }

    priv
    {
        if
        {

        }
        return false;
    }

    private bool IsTouchStartMobile()
    {
        if (Input.touchCount == 1 && Input.GetTouch(0).phase == TouchPhase.Began)




    }
    #

    #
    p
    {

#if U

#els

#en

    }

    p
    {                 Proprietary Code, Script Redacted





    }

    p
    {
                                                                      Moved ||
Inpu





    }
    #

    p

    [F
    public Text display;
    public string[] textLines;
}
```

Appendix F – C# Script for Multi_Touch

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Multi_Touch : MonoBehaviour

{
        void Start ()
   {

      {

      }


   priv
   {
      if
      {



                 Proprietary Code, Script Redacted

                                                          active









      }
   }

   publ
   {
      //                                                  bile device.
#if UN

          {
            return Vector3.zero;
      } else
      {
         return Input.GetTouch(0).position;
      }
#else

#end
   }

   #r

   pu
   {

#if U

#else

#end

   }

   pri
   {
```

Proprietary Code, Script Redacted

```
   }

   pri
   {
                                                      Began)

   }
   #e

   #r
   pu
   {

#if U

#else

#endif
      return value;
```

```
    }

    pr
    {

    }

    pr
    {
                                                              Moved ||
Inpu
                 Proprietary Code, Script Redacted

    }
    #e

    [H
    pu

    [H
    pu

// the
    pu
    public GameObject nextObject;
}
```

VITA

IAN LEVSTEIN

EDUCATION

May, 1978	Bachelor of Musical Arts University of Windsor Windsor, Ontario
June, 1983	Bachelor of Education University of Windsor Windsor, Ontario
May, 2001	Master of Science Marshall University Huntington, West Virginia
Pending	Doctor of Education Morehead State University Morehead, Kentucky

PROFESSIONAL EXPERIENCES

2012-Present	Computer Operations Manager Forensic Science Center Marshall University Huntington, West Virginia
2009-2012	Information Technology Specialist Forensic Science Center Marshall University Huntington, West Virginia
2001-2009	Information Technology Consultant School of Medicine Marshall University Huntington, West Virginia
1998-2001	Test Administrator Federal Bureau of Prisons Federal Correctional Institute – Ashland Ashland, Kentucky

1983-1996	Adult Education Teacher / Music Teacher Frontenac County Board of Education Kingston, Ontario

HONORS

2014	Delta Delta Epsilon National Honor Society for Forensic Science Huntington, West Virginia
2012	Upsilon Pi Epsilon International Honor Society for Computer Sciences Ft. Lauderdale, Florida
2002	Beta Gamma Sigma National Honor Society for Business Huntington, West Virginia
1992	Dr. A. Vibert Douglas Award RASC – Kingston Center Kingston, Ontario
1986	Teacher Service Award Frontenac County Board of Education Kingston, Ontario

PUBLICATIONS

Jenkins, C. D., Sammons, J., Levstein, I., & Fenger, T. (2018) A novel approach for validation of infotainment system Trackpoint maps using Java OpenStreetMap Editor®. *Marshall University College of Science,* Huntington, WV.

Wright, J., Levstein, I., Mosley, D., & Fenger, T. (2018). The walls are listening: A forensic study of home personal assistants. *American Academy of Forensic Science*, Seattle, WA.

Tredway, K., Bradley, K., Fenger, T., & Levstein, I. (2018). Overcoming the hurdles of imaging, storing, and archiving digital evidence. *American Academy of Forensic Science*, Seattle, WA.

Bobka, T., Levstein, I., Westurn, N., & Fenger, T. (2017). An analysis of a photocopier hard drive for forensically relevant artifacts. *American Academy of Forensic Science*, New Orleans, LA.

Baber, K., Brunty, J., Boggs, R., Levstein, I., & Fenger, T. (2015). Age estimation of adolescents and adults using dimensions of the eye and pupil in "Selfie" photographs. *American Academy of Forensic Science*, Orlando, FL.

Green, J., Levstein, I., Boggs, R., & Fenger, T. (2015). Steganography analysis: Efficacy and response-time of current steganalysis software. *American Academy of Forensic Science*, Orlando, FL.

Chom, E., Brunty, J., Levstein I., & Fenger, T. (2011). Forensic analysis of a Playstation® 3 slim model. *American Academy of Forensic Science*, Chicago, IL.

Corvo, M., Cathcart, K., Levstein I., & Compton, S. (2011). Comparison and evaluation of crime scene reconstruction software using Visual Statement FX3, 3D Eyewitness, and Crime Zone. *Marshall University Forensic Science Center*, Huntington, WV.