Monmouth College

# Developing for Virtual Reality Using Google Poly and Unity Engine

MitchPerez & Shayne Sendera
COMP 401
Dr. Mayfield & Dr. Utterback
November 2020

## Abstract

This paper addresses the development of a virtual reality video game, called InvestigationVR, utilizing Unity Engine, Google Poly Toolkit and an application called SketchBox in order to aid with its development. The game will be a single room experience containing many features and tasks revolving around a forensic analyst. The overarching goal of this game is to guide the player through an experience of conducting an investigation using evidence and the appropriate applications and tasks in order to determine who is at fault for an alleged crime. In order for the player to pick the person who is at fault, they must complete a list of tasks. These steps will be tracked on a progression basis that moves forward as the various forensic tasks are completed. After all progression is complete, the player is given access to three buttons corresponding to three different people within a criminal lineup. In order to achieve the aspirations of InvestigationVR, it is important to first understand SketchBox and the Google Poly toolkit and how each interacts with Unity Engine.

Sketchbox is a 3-dimensional sandbox application for the Oculus virtual reality headset. It allows for easy prototyping and designing of game objects. It also makes creating the rendering of game objects more simplistic thus making a more user friendly experience. This is very similar to another application from Google called Poly as it is built upon the same virtual framework. One of the advantages of Sketchbox is that it allows for collaborative design sessions with multiple virtual reality headsets. Sketchbox also has built in integration with the Google Poly Toolkit.

The Google Poly Toolkit is a plugin which allows the user to import Poly assets into the current project. These assets become objects, once imported into Unity, that can have certain properties applied to them at edit time and runtime of the project. With the aid of Sketchbox, objects can be uploaded to Google's repository of objects and accessed by Unity using the Google Poly Toolkit web API. This creates a much smoother workflow and allows for the primary focus to be adding functionality and scripting rather than graphic design.

Unity Engine is a cross-platform game engine that allows for users to make real time 3D projects for video games, animations, films, virtual reality experiences, etc. Within Unity, creating the game's scene as well as its functionality can be completed. This includes writing scripts, predominantly in C#, tailored to various properties in the occurring scene. Unity's ability to provide for good mutli-tasking allows for the user to access various windows and is essential to the workflow.

## Background & Introduction

Virtual Reality (VR) is the use of computer technology to create a simulated experience that can have various applications (Bardi). Ranging from gaming and entertainment to educational purposes, VR can produce both non-realistic and realistic environments. VR technology can span a few different types, such as non-immersive, semi-immersive, and fully-immersive (Poetker). Non-immersive VR is the most common form and has no virtual immersiveness to the environment. Semi-immersive VR contains a semi virtual environment, such as a flight simulator, while fully-immersive VR encases the player in virtual surroundings. VR headsets allow for players to be immersed into artificial environments in which they can interact with objects around them. How a simulation is developed also can be determined by the VR platform in use. An example of this are the different headsets produced by various companies, such as the Oculus from Facebook, the Vive from HTC, and many others. Each platform accounts for tracking and other various sensory data differently, which is why it is, at times, difficult to create applications for all VR technologies. The specific development of this project revolves around the Oculus platform and the Google Poly Toolkit integrated into Unity Engine.

Unity Engine (Unity) is a cross-platform game engine that allows for the real-time development of two-dimensional and three-dimensional projects. With few game development softwares available, Unity allows for great compatibility with VR and Google Poly Toolkit. It is more efficient to use over other game engines such as Unreal Engine because of its simplistic nature and easier learning curve for small development teams. Unity is also flexible as it can adapt using “C#  scripting, comprehensive APIs, and extensive documentation” (“AR and VR Games.”, Unity). The flexibility that is built in, allows for smaller teams to be able to have numerous tasks being worked on and each task can be completed in a timely manner. The rich interactivity that Unity allows for, gives the developer access to physics, rendering, and communications for seamless integration with other devices. It also supports the various VR platforms discussed prior which allows for easy integration of an Oculus Rift headset.

Oculus Integration is an asset package for Unity that contains core VR features such as “plugins, scripts, materials, textures, shaders, and other tracking data” (“Understand Oculus Integration Package Components.”, Oculus Developers). It is not among the standard libraries that are provided with Unity, but it is a free and easy access package that resides in the Unity asset store. The integration toolkit along with a few changes to the project settings is enough to deploy an Oculus VR headset functionally. It accounts for different tracking methods from Oculus such as outside in tracking and inside out tracking. Outside-in tracking consists of stationary sensors, placed strategically around a room. Whereas inside out has the sensors inside the headset and tracks your location and controllers from the headset. Depending on how recent the model was released will determine what versions of Oculus Integration that need to be utilized. The Oculus integration is essential for developing for the platform in Unity as many of the expectations of VR are integrated “under the hood” and allows for the developmental focus to confine to scene and script work. Poly is a VR framework provided by Google which functions as a sandbox application. It allows for the creation of three-dimensional game objects. In a sense, Poly can be used in a few different ways: creating the rendering of the scene, as well as individual assets. Anything that is created within the scene using Poly will be stationary and will not have VR interactivity because it will be represented as a single object when it is imported into Unity. On the other hand, any object that is imported separately will be interactable and potentially functional with the aid of C# scripting. All assets created and published publicly by users around the world are also stored in Google Poly's database. This allows for objects that were previously created to be used in projects along with Google's default creators license, which allows for the use of Google's assets with integrity. Sketchbox is another VR application developed upon the Google Poly framework. It is very similar to Poly in almost every aspect, but is more polished and has capabilities of creating game objects online. It's online capability allows for meetings to be held within the application itself enabling creators on the same development team to work together in VR in real time. Sketchbox is one of the only VR applications that allow for a team to work alongside each other in real-time.

The Google Poly Toolkit is a simple application programming interface (API) that allows for Poly integration into Unity Engine. The API allows for Unity to import assets directly into a project from Google's repository before and during runtime ("Poly API Guide", Google Developers). Being able to access assets before and during runtime allows for an improved efficiency due to its ability to test certain functionalities before and during game play. The Poly Toolkit API is not essential for importing game assets but it is a useful tool. Without it, importing many objects would be quite tedious as each Filmbox File (FBX), game object file type, needs to be individually added to a project.

The main focus of this project is to be able to create a fully-immersive VR video game, meaning that the entire player environment will be virtual. Although this has already been done and explored by many developers in recent years, this project attempts to highlight a newer streamlined version of developing for VR using many of the new technologies aforementioned. Previous workflow required the use of virtual programming languages in order to create the rendering of a custom scene with some degree of detail. The new process allows for easy virtual design of any object rendering as well as simplifies the process of implementing the object's renderings into the Unity scene. With the visual work streamlined, it allows for the primary concentration of work to be directed at game functionality rather than its graphical appearance. Thus allowing VR developers to better their in-game functionalities and making a game that feels like reality and the virtual world are one.

## Project Description and Analysis

InvestigationVR is the name of a concept VR game that specifically revolves around forensic analysis. The scene exists within an area that is split between a lab and a police station. The main idea of InvestigationVR is to demonstrate three different procedures that attribute to hair, blood, and fingerprint analysis by using the provided evidence at the start of the game. With completion of each task, useful information will be presented on a wall mounted tackboard within the police station. After all of the procedures are completed, functionality will be given to three buttons, also within the police station, each corresponding to an individual in a criminal lineup. Once one of three buttons has been pressed, the game ends and declares if you have chosen correctly.

The forensic science that takes place within InvestigationVR begins by using a few different pieces of evidence provided at the beginning. The evidence may vary between each loading of the game as ideally having three scenarios will make the experience more interesting. For each scenario there will be a hair sample on a slide, three blood samples, and some type of weapon used in an act of violence.

The evidence containing a piece of hair, prepared on a slide, is used for the game in order to push progression forward. We intend to have three different hair colors to differentiate between the lineup suspects. The player will take the slide to a microscope and place it under the objective lens microscope. Once there and completed, an image will appear on the tackboard corresponding to the hair placed under the lens. The image on the tackboard that appears will be a closeup image of the hair. Based on the pigmentation, medulas, and cortexes that appear to make up the hair a similar image can be matched to the profile of a suspect, which will also be present on the tackboard. Each loading of the game will have a different hair sample that corresponds to a different person within the lineup.

Another piece of evidence is made up of three small vials. The vials contain a prepared solution of a DNA template, blood, and DNA polymerase, a necessary enzyme, that needs to be placed within the Polymerase Chain Reaction (PCR) machine (National Center for Biotechnology Information, U.S. National Library of Medicine). The player will need to take the prepared solutions to the PCR machine and place each sample inside. There will be a short timer to simulate the PCR synthesis and its result will be added to the tackboard. This task will need to be repeated three times, once for each vial, and compare all three results after its completion. Once all of the images appear on the tackboard, the player must match the DNA helices that are present to one of the suspect profiles. Two of the helices will be replicas as they are the two vials of blood that were the victim's blood. The third helix that appears will not be similar to the other two, this is the suspects DNA sample. The matching samples of blood will remain consistent throughout all loadings of the game, however the unmatching sample will vary from each load of InvestigationVR based on the given scenario.

The last piece of evidence that spawns at the start of InvestigationVR is a weapon that was used in the committed offense: a gun, a knife, and a bowling pin. Each scenario will have a unique item that will be used for fingerprinting via the superglue method. This procedure requires the player to place the weapon in a hood vent alongside the proper solution, cyanoacrylate, and wait a small portion of time for its corresponding result to appear on the tackboard. The result will be an image of one of three fingerprints containing apparent minutiae, which refers to specific "plot points on a fingerprint including characteristics such as ridge bifurcation or ridge endings" (Beal). Following suit to handling the result, the player will see the corresponding image on one of the corresponding suspect profiles. Each piece of evidence is a key item for navigating game progression.

In order to track game progression InvestigationVR will utilize a wall mounted tackboard that is essential for presenting the player information about a scenario as they complete the various forensic tasks. This feature will appear as a tackboard seen in an ongoing criminal investigation, meaning there is already some information present and connected. The player will need to fill in the missing information in order to make an educated assumption of who the suspect may be.

The specific information that will appear visible from the beginning will consist of criminal profiles, some photographic evidence from the crime scene, and task hints. The criminal profiles will consist of: an image of the individual, height, weight, eye color, age, sex, hair color, DNA helix, hair sample, fingerprint, and any prior records. Some of the information presented on the three criminal profiles may prove to provide useful information for judgement when it becomes time to solidify a verdict. For example, the fingerprint, DNA helix, and hair sample will be used for matching the resulting images to the suspects' profiles. The photographic evidence of the crime scene will consist of images of a fictional crime scene in which the evidence resided in. Images of collected evidence will also be present throughout the board in order to give the player a physical description of the crime.  The last of the given information are various hints about how to do each task. This is a feature to ensure the tasks and progression of InvestigationVR are not hindered from a lack of knowledge in the field of forensic science.These hints would explicitly give written directions to the player step by step about how to complete the analysis portion of InvestigationVR.

Once the player completes the aforementioned tasks: hair analysis, blood analysis, and fingerprinting, images of the results will appear on the wall mounted tackboard. These images will appear on a piece of paper and they will correspond to its specific task. The player can have up to five different results appear on the tackboard; one each for both the finger print and hair analysis, and three for the blood analysis. The information portrayed in the resulting images are unique to each item. The fingerprint analysis yields an image of a fingerprint. It will depict a whorl, arch, swoop, or double whorl fingerprint. More specifically, the image will have arrows pointing to key characteristics of the print that displays its uniqueness, also known as minutiae. The blood analysis results in a DNA helix which will contain distinguishable base pairs. The base pairs will be appropriately matched inside of the DNA helix according to their bonds; adenine with thymine, and guanine with cytosine. The resulting sequence can later be compared to the sequence of the suspects.

After the player completes all of the tasks and the tackboard is filled out, three buttons, positioned on the back wall in the police station, will become functional. These buttons correlate to a potential suspect among the lineup. When the player believes they know who the criminal is, they can push one of three buttons in order to complete InvestigationVR. The tackboard will once again contain information regarding if the player was correct or not. Now that the player is complete with the process that InvestigationVR simulates, they can also push a reset button located near the tackboard. This button will reload the game scene by loading one of three scenarios aforementioned.

Other game objects that will be used in InvestigationVR will primarily consist of various chemicals and tools that relate to a lab setting. Many of these objects, mostly chemicals, will have a purpose towards the game progression. The other objects that mostly consist of tools and a few irrelevant chemicals are implemented to add some difficulty and help with the immersion of a lab. On the counter top, there will be various instruments, similar to forceps, mortar and pestle, and pipettes. There will also be beakers that contain chemicals, most of which are used in forensic labs, but only one that works with the fingerprinting analysis. The various chemicals consist of Ninhydrin, Sulfuric Acid, Sodium Hydroxide, Propylene, and Cyanoacrylate which is used for placing inside the fume hood for the fingerprinting task. All of these objects will be interactable, although only few of them will be used throughout the game to complete the tasks.

## Foundations

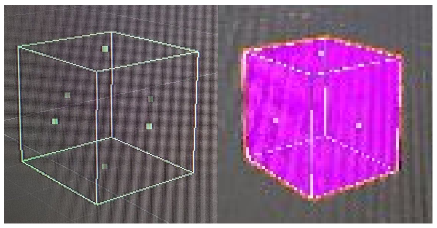
For the fulfilment of the aspirations presented in this project, it is first important to understand the technical details and challenges of its implementation in greater depth. The details are not very complicated concepts to necessarily understand, but if not utilized appropriately can cause difficult challenges and pose little to no benefit producing a functioning product.

Google Poly is a framework allowing the user to import assets into Unity Engine allowing for the creation of environments at a faster rate. It can function as such with the integration of Google Poly Toolkit and Sketchbox. These two applications: Google Poly Toolkit and Sketchbox, allow for the creation and storage of assets that Google Poly is able to access through Unity's interface.

While Google Poly and the Poly Toolkit are useful and creative tools to create game assets, the technical aspects of developing InvestigationVR begin to take their course primarily in the Unity Engine phase of the project. The reasoning for this is that Google Poly is primarily used for the creative side of the project consisting of designing specific objects relative to the topic while Unity creates virtual renderings from the Poly asset files that have been created. Unity is the development environment where all the different points of view of the project start to become integrated with one another. The various technical aspects and challenges of the project become more apparent through both visual appearance and functionality of a scene. The specific technical details of this project entail creating the scene with Oculus capabilities, considering the Rigidbody class, explaining colliders and collisions, as well as scripting in C#.

Creating the scene in Unity is the first technical aspect of creating InvestigationVR. As simple as it may seem, this is the case because it is the foundation of what the rest of the virtual reality experience will be built off of. Creating the baseline scene also requires the proper implementation of the Oculus Integration Toolkit in order to have the proper hardware, VR headset, functioning in any way. Creating the scene begins with importing the scene object created within Google Poly or Sketchbox. In the specific case of this project, the object will be created using Sketchbox, which is a nearly identical application to Google Poly. Then the scene will need a three-dimensional plane in order to simulate the floor. This is because the Oculus Integration Toolkit calculates the players height and location based on the distance to the floor in reality. A plane in Unity is also, by default, an object that acts as a collision object with the virtual camera within the scene. After the scene object is placed on the plane, the default Unity camera can be removed from the scene and will be replaced with an OVRCameraRig. OVRCameraRig is a class provided by the Oculus Integration Toolkit and is used to represent the player within the scene built off of Oculus' framework. The OVRCameraRig provides tracking data as well as contains very simple event handling functionality. This allows for accurate tracking of the Oculus hardware throughout the scene during runtime. This data is generated by the sensors that reside on the headset and is used by Oculus to place the player at the correct location in the scene. The main problem with the Oculus Integration Toolkit is that there is no published information of exactly how the sensory data is computed, which can sometimes cause confusion when visual bugs occur during later development. Once the Oculus Integration Toolkit and hardware are functioning in Unity, the Google Poly Toolkit API can be utilized to import other various assets into the scene. At this point the Unity program is in the same state as a “Hello World” program where it is just an example of a functioning program that does nothing.

With the scene all static objects as well as functional dynamic objects in the scene need to be given colliders. A collider is the underlying bounding box of a game object. It is an invisible boundary which gives a game object realism in the sense that it is a physical object awaiting realistic functionalities. This would allow for the numerous assets to have collisions, the physical contact of two or more colliders, in order to stay within the scene and interact with one another. Without colliders, game objects that specifically have gravity will fall out of sight of the play area. Thus losing the object completely, until reloading of the scene, which would cause the same problem. Any object that needs to uphold any physical properties must first have a collider. Colliders are also used to allow an interaction to occur between an asset and the player. This happens through the use of two different types of colliders: physical colliders and trigger colliders. Physical colliders, as previously mentioned, uphold the area of an asset. A trigger collider, on the other hand, does not uphold physical properties but instead is used to trigger a reaction to an event. Trigger colliders have several different use cases in order to fabricate some type of event, in our case creating buttons and tasks for the player to use. For example, if a player wants to pick up an object within the scene, the player grabs the object with the touch controller. On the surface, that is what happens. But in reality what happens is a trigger collider, on the virtual reality controller, interacts with the physical collider of the object, as the physical depiction of the object is just a rendering of its image. The pressing of a button on the controller creates an instance of an event on the physical collider, allowing it to be picked up or moved.


A $1\text{m}^3$ cube object and its collider (with and without rendering)

Once the assets have colliders applied to them, the Rigidbody class from the Unity standard library can be taken into consideration. The Rigidbody class is responsible for providing various physical properties such as gravity. It is applied to specific objects individually and can have scripted behaviors based on an instance of an event, otherwise the default physics engine is very simplistic and doesn't handle collisions very well. In the specific use case of InvestigationVR, rigidbodies will be used primarily for dynamic objects that can be interacted with in order to have basic gravity to allow them to fall to the ground after being dropped.

The functionality aspect of many of the objects and scenarios doesn't come from Unity alone. Unity as we know is a game engine. Game engines in general are used to create games, just as they specify. A videogame is essentially a scene in which one has a motivation to complete a goal. The game aspect of a video game is within the storyline that is implemented into a given scene. Unity accounts for the fact that a scene that doesn't have a purpose isn't a videogame at all.

In order to add the functionality and specific events to InvestigationVR, Visual Studio from Microsoft will be used to write scripts that account for game progression and event handling. These scripts are used to both alter existing properties in the scene, such as velocity of an object, as well as standard creation and classes that C# allows for. Majority of the events that occur within InvestigationVR will be scripted in C#, the default language that Unity Engine requires. The scripts that will be written and are the most essential will be devised for each main feature. The significant scripts that will be written consist of "FingerPrint.cs", "BloodHelix.cs", "HairAnalysis.cs" and "Tackboard.cs"

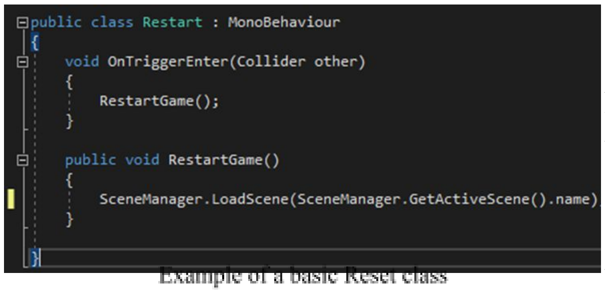
In regards to adding the functionality to move along progression in InvestigationVR, "FingerPrint.cs" will incorporate the processing for multiple different inputs. The script will be constantly checking for objects, within the scene, that are placed within a trigger collider that is set in place at the bottom of the fume hood. It will be checking for specific object names such as "Evidence" and "Cyanoacrylate" as there are many game objects that are not relevant to the specific task that will also reside within the scene. When writing the "FingerPrint.cs" script in Visual Studio, each object in the game will need to be accounted for as well as its state of relevance towards a specific task. When the two specific objects, listed previously, within the bounds of the trigger collider, the script will start a fifteen second timer. When the timer finishes, a function called makeAppear() will be called from the tackboard script in order to make the correlating result appear on the tackboard.

In order to complete the blood analysis, we will also be programming a script called "BloodHelix.cs". "BloodHelix.cs" , similar to "FingerPrint.cs", will be constantly checking for objects being placed within a slot on the PCR machine. Although instead of checking for two objects, it will be checking for three different objects called "VictimBlood", "VictimDuplicate", and "SuspectBlood". Each object is visually represented by a vile of solution containing the blood sample and can only be placed inside the PCR machine one at a time. Once one of the correct objects is placed inside the trigger collider's zone, then the script will begin a thirty second timer. This task will repeat itself for two more iterations as there are three separate vials. Once a timer completes, the image of the specific sample will appear on the tackboard by calling makeAppear() from the tackboard.

For the final task, "HairAnalysis.cs" will check for the "HairSample" to enter a trigger collider located on the microscope slide, similarly to the two previous tasks. Once the proper object, “HairSample”, enters the collider’s area, a timer will ensue for fifteen seconds. If any object that is not “HairSample” enters the collider’s zones then the timer will not start. Once the fifteen seconds have concluded, makeAppear() will be called from the tackboard script and an image relating to “HairSample” will appear on the tackboard and help the investigator solve the crime.

The tack board feature that is used throughout InvestigationVR is the visual representation of how the player receives their information as well as any help they might need to complete the mentioned tasks. It is also one of the core game mechanics as it tracks the games progression as the game is played. The tack board is a physical game object that is hung on the wall in between the two areas, discussed in the last section, and holds an array of images and documents that give the player a greater understanding about the suspects. It also keeps track of the game state by having three variables representing the truth value of the completion of each task. By default these variables are set to false and completing each task calls the makeAppear() function in the tackboard script which switches this truth value to be equal to true. In order to update this change, a function called changeGameState() will change the image's occupancy so that it is visible to the player on the tackboard. Once visible to the player, they can then look at the images and use the information from the tackboard to solve the mystery. At the same time that changeGameState() is called for the last time, activateButtons() is called on the three GuiltyButtons on the back wall of the police station. activateButtons() purpose is to allow the three GuiltyButtons to be functioning aspects of the game.

The other scripts that will need to be written are those that serve a purpose to the game but only have a singular purpose. The two other scripts that will be needed are ResetButton as well as three versions of a GuiltyButton. The reset button is a simple script that calls the load() function that can be used to load the same scene over again with a new scenario. The button works by having a trigger collider within the base of the button that is set to call the LoadScene() function. In InvestigationVR respectively, each scenario that is loaded will be a randomly generated number from zero to two and will be incorporated into the load() function call. The GuiltyButton script will need the slightly different variations. A pushing of a guilty button will write the chosen verdict to gamestate before it displays whether or not the player was correct with their choice. The reasoning for three variations specifically is due to the three different suspects to choose from. Typically, each choice can be determined and handled within one script, but due to the nature of each button representing separate objects within the Unity scene their functionality must be split apart.



```
public class Restart : MonoBehaviour
{
    void OnTriggerEnter(Collider other)
    {
        RestartGame();
    }

    public void RestartGame()
    {
        SceneManager.LoadScene(SceneManager.GetActiveScene().name)
    }
}
```

Example of a basic Reset class

The physical buttons in Unity will be created by having a spring joint component, a part of the Unity standard library, push and retract when a player interacts with it. In order to trigger the previously mentioned events, a small, flat trigger collider will be placed under the pressable part of the button. When the player interacts with the button by pushing it, the interactable object comes in contact with the collider creating an instance of that event.

## Implementation Plan and Time Line

The implementation and timeline of the InvestigationVR project is the schedule in which this project, InvestigationVR, will be completed with. It is broken up over an approximate time of twelve weeks, with six checkpoint meetings every two weeks apart. The semester itself is longer than twelve weeks but due to other assignments such as poster creation and presenting, InvestigationVR will only be given these twelve weeks to have working software.

In order to implement the project in a timely fashion, the research of specific software such as Google Poly, Sketchbox, and Unity has been conducted. This reinforces the core features and expectations that were already set for InvestigationVR and its features. To ensure that Unity Engine functions properly with the hardware that is in use, the Oculus Integration Toolkit and Google Poly Toolkit will need to be imported and installed in the correct locations.

The first two weeks of the spring 2021 semester will involve the creation of the virtual scene in Unity Engine using Sketchbox and Google Poly. As mentioned previously this allows for the creation of three dimensional game objects to be created using the perspective of VR technology. Using these softwares, both the scene object and the numerous game objects can be quickly modeled for easy access by Unity at a later time. The goal of the first checkpoint is to be able to have a baseline, functional scene that can be displayed through the Oculus Rift headset that will be in use. This would essentially be the 'hello world' program of the project that would ensure all the proper settings and packages  and assets have been added properly. If any extra time is available after project setup, it would also be possible to begin creating any colliders that functional game objects will need.

Weeks three and four, the second checkpoint, consist of beginning on the fingerprinting task feature of InvestigationVR. This checkpoint can be broken down into two smaller, weekly, iterations that make up the entire feature: the task and its correlating result on the tackboard. The task itself will require the evidence that is in the form of a weapon to be placed in a fume hood with the proper chemical, cyanoacrylate, in order for the fingerprint to appear on the tackboard. This will take every bit of eight to ten hours for that week as setting up trigger colliders can be quite tedious at times, especially when having to account for specific objects within a given area. This portion of the second checkpoint is the setup for what the player is going to do during runtime. The second portion of checkpoint two will focus on the underlying scripts that enable visibility of the resulting fingerprint on the tackboard. Ideally, around this timeframe we would like to have a mostly working task; outside of a few potential bug fixes.

The following checkpoint, weeks five and six, will revolve around the implementation of the reload/restart button as well as the three guilty buttons that will reside at the criminal lineup. The restart button will be a large red button located near the tackboard. When pressed, it intuitively does what it is named after, restarts the game. The only caveat here is that it will eventually choose from three different scenarios, to prevent InvestigationVR from becoming boring quickly. The guilty buttons located at the criminal lineup will each be located directly in front of the person that they correspond to respectively. The main reason for the implementation of all the button objects is to give InvestigationVR a "pre-alpha" completion meaning that the game can be played from a starting point to an ending point. At this point in time InvestigationVR should have all of its game objects, one functioning task, and working buttons to finish and restart the game and be wrapped up into a short version of the end product.

With a complete story line now in place, weeks seven through ten will focus on the other two tasks in InvestigationVR. Weeks seven and eight, checkpoint four, will begin the process of developing the blood analysis task and its tack board game progression. The process is divided similarly to the hair analysis task that, at this point, was previously implemented. The two iterations of the blood analysis will also be in the forms of what the player will do and see as well as how the resulting DNA helices from each blood vial will appear on the tackboard.

The following two weeks and iterations of the project will comprise creating the task and game progression of the hair analysis with the microscope. As stated prior this is broken down in exactly the way that the two previous game features have. It too is divided into the portion of the task that has to be done by the player, such as implementing the trigger collider on the microscope as well as checking for specific objects in that area. The second iteration of this feature is in regards to the scripts under that hood that allow for the hair image result to appear on the tack board. Weeks seven through ten are for adding more content to the game, such that the game is no longer a one and done kind of deal.

The final two weeks and last checkpoint, overall, is to discuss some of our methodologies of testing InvestigationVR. The game can be tested by essentially playing each scenario over and over, each time attempting to recreate a potential bug. Testing for InvestigationVR will be essential as many classes and tracking features may sometimes seem buggy, depending on the input the user provides to the game. For example, if the player swings an object around very quickly there is the possibility of the object calculating an incorrect location within the scene which will create a visual stuttering effect. In order to correct many of these visual and computational infractions rigorous and timely testing will need to take place. This also reassures that the previously implemented features all get along in such a fashion that they are still operational. With the testing process complete, it would come time to start preparing for posters and presentations for the InvestigationVR virtual reality video game.

In the case that some of the discussed implementations are too complex to integrate within this timeframe, scaling back of some features may be important to keep the end result of the project functional. Scaling back can begin with the simplification of the pool of objects that will be present in the scene. Outside of the main pieces of evidence and one or two chemical solutions, the rest of the game assets are non-functional and serve the purpose of adding immersion and confusion to the process of the game. Without these objects InvestigationVR would still remain a functional piece of software that runs the player through three tasks, it would just seem more straight forward and dry. Another form of scaling back some difficulty due to time constraints can be simplifying the information that will be presented on the tackboard. Some of the information such as the criminal profiles will need to be specially created text within the scene. To get the proper formatting of such an essential component it could be tedious and take more time than originally planned. Under the correct circumstances the tackboard can be simplified into only having the essential information for completion of InvestigationVR. In the ideal situation scaling back of the project will be unnecessary as we believe it is reasonable to complete the amount of work discussed in the given amount of time of approximately twelve weeks.

## Conclusion

The creation of a fully-immersive VR game was once a challenging process. The intention of creating Investigation with tools such as Unity Engine, Google Poly Toolkit, and Sketchbox has allowed for this process to become much more streamlined. This can benefit both the developers for this genre of entertainment as well as encourage outsiders to to worry less about the fine technical details. Another benefit of the process is that productivity and workflow have been greatly simplified as integration between different technologies hasn't always had a straightforward solution to this problem. Although there are still minor problems to occur throughout the project, the aforementioned technologies limit many bugs and glitches to occur within the scripts written by the creator(s).

The scripts written in C# is what comprises most of the game functionality. It is overall responsible for any and all event handling that occurs within InvestigationVR. It accounts for anything that is placed in the fume hood, and can differentiate between specific objects entering its collider. This is similar functionality to both the blood analysis and hair analysis as each task also accounts for a physical collider entering the area of a trigger collider. These events are important fundamentally as without them there is no tracking of the progression within the game.

In order to implement many of the features that are talked about within this paper, understanding the different concepts within the chosen softwares and technologies is vital. The different components and integrations that are centered around Unity Engine and this project all span a wide variety of functionalities and serve their own purposes toward the development of a fully-immersive forensic analysis virtual reality video game. Some of the technology integrations such as the Google Poly Toolkit and Sketchbox revolve around the graphical design aspect of the entire project. The main purpose of using graphical design softwares is to allow for a larger repository of game objects. If an object doesn't already exist, it can simply be modeled in VR. The integration window within Unity also makes importing objects into the scene quick and easy. Unity Engine and the Oculus Integration Toolkit are the other softwares being utilized within this project. Unity is a game engine that enables the development of both two and three dimensional games. Since it is also cross platform and universal to most operating systems, it is a very powerful framework to build a game from scratch. Many of its properties allow for the integration of VR headset tracking as well as incorporating many physical properties.

This paper's purpose is to raise awareness of this VR development process as it is still relatively new and unpopular. It is a very intuitive and easy way to develop in virtual reality even without necessarily having any previous experience or knowledge. It provides the full development experience, from graphics to game progression and event handling, of creating a virtual scene and the assets that it needs to become functional. Without such a streamlined process, the amount of features and detail of InvestigationVR would be nearly unachievable within the allotted time of the Spring 2021 semester. Overall, creating a virtual reality game using Unity Engine, Google Poly, and other aforementioned technologies has bridged a gap that has been limiting motivation and inspiration of many young or inexperienced developers. The popularity of virtual reality, over the last decade, has been growing exponentially. With a simplified modeling and development process,  demands for both semi-immersive and fully-immersive virtual reality experiences can be fulfilled.

## Bibliography & Works Cited

Bardi, Joe. “What Is Virtual Reality? VR Definition and Examples.” *Marxent*, Https://Www.marxentlabs.com/Wp-Content/Uploads/2018/10/marxLogo397x58.Png, 21 Sept. 2020, www.marxentlabs.com/what-is-virtual-reality/.

Beal, Vangie. “Minutiae.” *What Is Minutiae? Webopedia Definition*, www.webopedia.com/TERM/M/minutiae.html.

“Collision.” Unity, https://docs.unity3d.com/ScriptReference/Collision.html.

“Configure Unity Settings.” *Oculus Developers*, developer.oculus.com/documentation/unity/unity-conf-settings/.

“Inside-out v Outside-in: How VR Tracking Works, and How It's Going to Change.” *Wareable*, 3 May 2017, www.wareable.com/vr/inside-out-vs-outside-in-vr-tracking-343.

“Oculus Documentation.” *Oculus Developers*, developer.oculus.com/documentation/.

“PCR Basics.” *Thermo Fisher Scientific - US*, www.thermofisher.com/us/en/home/life-science/cloning/cloning-learning-center/invitrogen-school-of-molecular-biology/pcr-education/pcr-reagents-enzymes/pcr-basics.html.

Poetker, Bridget. “What Is Virtual Reality? (+3 Types of VR Experiences).” *G2*, learn.g2.com/virtual-reality.

“Poly API Guide | Google Developers.” *Google*, Google, developers.google.com/poly/develop/api.

“Polymerase Chain Reaction (PCR).” *National Center for Biotechnology Information*, U.S. National Library of Medicine, www.ncbi.nlm.nih.gov/probe/docs/techpcr/.

Robertson, Sally. “Hair Analysis in Forensic Science.” *News*, 21 May 2019, www.news-medical.net/life-sciences/Hair-Analysis-in-Forensic-Science.aspx.

Technologies, Unity. “AR and VR Games.” *Unity*, unity.com/solutions/ar-and-vr-games.

Technologies, Unity. “Colliders.” *Unity*, docs.unity3d.com/Manual/CollidersOverview.html.

Technologies, Unity. “Getting Started with VR Development in Unity.” *Unity*, docs.unity3d.com/Manual/VROverview.html.

“Understand Oculus Integration Package Components.” *Oculus Developers*, developer.oculus.com/documentation/unity/unity-utilities-overview/.

“Unity Quickstart | Poly | Google Developers.” *Google*, Google, developers.google.com/poly/develop/unity.

“Unity 3D: Collisions Basics.” *BinPress*, www.binpress.com/unity-3d-collisions-basics/.