SciKids

Final Report

Version 1.0

5/9/19

SciKids

Sponsor: Elizabeth Glass

Mentor: Austin Sanders

Samantha Earl, Claudia Coronel, Gwen Morris

Table of Contents

1. Introduction

1.1 Problem Statement

There are a multitude of STEM career opportunities in North America, and many of these positions are yet to be filled. According to the U.S. Department of Commerce, STEM occupations are growing at 17% annually while other occupations are growing at 9.8% annually. The United States Bureau of Labor Statistics has projected a large shortage of about 1 million STEM professionals in the country. The workforce is considered an important factor in supporting United States innovative enterprise, global competitiveness, and national security, and the shortage of STEM-educated workers can damage these areas. Our project would facilitate the introduction of STEM-related concepts to young students and those who are considering a career change.

An effective way of creating an initial interest in STEM is to present broad, general STEM concepts to young children in an entertaining way. Studies have shown that middle school is a pivotal time for children in deciding where their interests lie, so this age group needs to have material that can help cement student interests. By early exposure of STEM areas, we hope to begin and cultivate STEM interests which would then lead to a greater likelihood of students choosing a career in the field. Older students, such as high school and college students, need more career readiness material. This includes information on building resumes, interview skills, and achieving important standards for career readiness. Our client, Elizabeth Glass, wants to provide a system that can solidify STEM interests in younger students and provide material that focuses heavily on career development skills for older students.

1.2 Solution Vision

The solution that the team has implemented is a gesture-based learning system for STEM recruitment and support in the kindergarten through college and community age individuals. A gesture-based system itself is defined as a system that allows individuals to engage in a virtual environment using motions and movements that mirror real-life actions. There are several benefits that the team and sponsor have identified that lead to including gestures for the project. Gesture-based learning systems can increase an individual's intrinsic motivation. Gesture-based systems have also been shown to increase memory retention and aid in learning concepts through physical means. By choosing to implement motions, students are able to have a virtual, hands-on learning approach which will allow the users to actively and physically engage in their learning. This solution extends the core functionality of the problem statement by:

- *be openly available to anyone in the community*
- *provide information on STEM in a visually rewarding way*
- *enhance individuals professional skills by providing up to date information in the STEM field*

Overall, our solution consists of an interactive system that will be able to educate users on the possibilities of a STEM-based education as well as provide resources to users looking to start a future career.

2. Process Overview

Our project was a year-long project with extensive planning and scheduling in order to satisfy our sponsor. In the beginning, we created a Team Standards Document, which contained all of the roles in our team and reached an agreement of expectations of each other. We then moved into the process of planning. We had weekly meetings with our capstone mentor and monthly meetings with our sponsor. During the first couple of meetings with our sponsor, we were focusing on gaining a better idea of the vision of the project. Our sponsor made it very clear that this project was completely new to Flagstaff and that we were going to be creating something brand new.

To begin building this new system, we started with developing requirements. At this time we identified what the special features of our project were going to be as well as the functional and non-functional requirements. After mapping out the requirements we built a timeline to follow as well as dedicated positions where we felt like each team member would do best. When we had a clear idea of what we were able to create a technological feasibility analysis of the potentially new items were going to be using. From this, we decided on three different technologies: Unity, the Intel RealSense D435 camera, and Nuitrack.

Shortly after choosing our technologies and creating a small demo of what our system might look like we were able to move into creating our software design document where we discussed implementation and architectural overviews of our system. This was extremely helpful during the development phase of our project. During the development of our system, we ran into a few challenges, of which we provided solutions to in our later design reviews. After development, we went into the testing portion. Later we displayed the final demo to our sponsor as well as our client, meeting all of the requirements we laid out in the first semester.

Working as a team proved to be one of the biggest factors of the success of our project. As a team, we communicated daily through our group chat. We made time for weekly meetings and allocated tasks accordingly. As a team, when struggles arose we brought them to light as quickly as possible. When one of us needed help in a section we made it very clear. Without the communication and specification documents we wrote, we couldn’t have tackled this project. This shows a general overview of the software engineering cycle we went through as SciKids.

3. Requirements

In order to develop a product that matches our solution vision, we have gathered and placed our requirements in functional and nonfunctional sections. In the acquisition phase of the project, the team met with Elizabeth several times a month discussing what our sponsor would like and how the team would approach meeting the ideas our sponsor brought to our attention. The team provided several outlines of the system to ensure that the team was continuing to meet the sponsor’s expectations. During development, these requirement specifications acted as a guide in ensuring that the specific problems and solutions presented are indeed what makes our product of value to the users. Here are the domain level requirements for our product:

1. *Users will be able to gestures as a means of navigation through the system*
2. *Users will be able to access different modules through our system.*
3. *Users will be able to by making a personal profile and see personalized available modules.*
4. *Users will be able to save personal progression and set high scores.*
5. *System must be extensible for future developers.*

Each of these different domains contains functional and nonfunctional requirements. These domains work together to complete the overall purpose of our system. The next section in our document will be addressing these high and low-level requirements with respect to their types.

3.1 Functional

The functional requirements of the system include tasks that the system should be able to do. The first requirement is that the program has a main menu area. From this main menu page, users are able to access any of the content and resources that will be provided by the system. The second requirement is gesture recognition. Gestures are to be used as a method of navigation through the main menu of the program, along with a mouse and keyboard for the desktop version of this system. The third requirement is the implementation of a login system. Users are able to create an account and log in to a previously created account with a username and password. Validation ensures that the password and login are correct, error feedback is provided if not correct, and the login information the user provides can fit certain requirements. The fourth requirement was the user’s ability to save high scores and personal progression. Through the database associated with the login process, users are able to save high scores from games that are played. Finally, the last requirement is system extensibility for future developers. The team has created a simple way for developers to import their files into the system, and we have provided navigational tools that developers can use to allow users to view the new content.

3.2 Nonfunctional

The performance requirements of our system are important because they judge the operation of the system, rather than specific behavior. Here we will include some of the following performance requirements we set as goals for our software.

One important aspect of the project includes having a physically portable system for our sponsor. All of the hardware is small and easily carried by one person. The computer and camera will fit inside a bag so that a user can carry this system around to areas in the community and increase the availability of the system in the Flagstaff community. It was important that the GUI and module content is engaging and informative for the users. The information on STEM must be accurate and represent characteristics of STEM fields and skills. This information is expected to be engaging as well so the users will want to use the system and engage in the content the software provides.

4. Architecture and Implementation

The solution that the team has implemented is a gesture-based learning system that allows individuals to engage in a virtual environment using motions and movements for STEM recruitment. Recruitment primarily focuses on kindergarten through college and community aged individuals. In order to do this, we bought a camera that can track user depth and positioning as well as software that can translate user motions into certain gestures that we can later assign actions to.

As stated in section 2, our gesture-based system involves 3 main technologies - the Intel D435 Camera, Nuitrack, and Unity. A more detailed description of each technology can be seen below.

- *Intel D435 Camera***:** This camera includes a pair of depth sensors, an RGB sensor, and an infrared projector.
- *Nuitrack*: Nuitrack is a middleware that can track a 3D skeletal body and recognizes certain gestures a user makes. By utilizing some of its available modules we will be able to bring value to hand gestures within our system.
- *Unity*: Unity is a game development engine that is compatible with a wide range of technology. We will use this to create the GUI and backend processes, which is made

simpler through Unity. Nuitrack also offers Unity support that allows developers to directly map gestures to GUIs made through Unity.

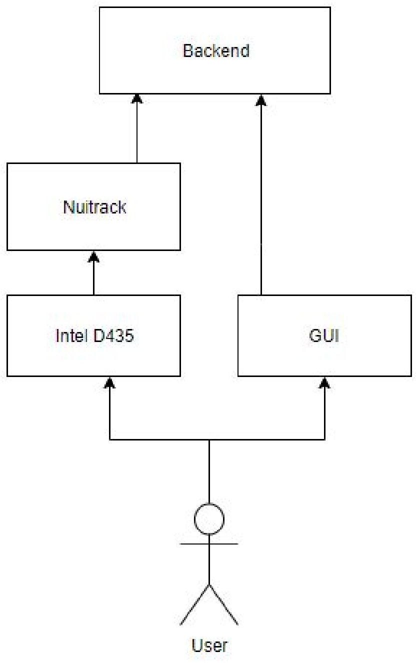


Figure 1: System Diagram

Figure 1 above shows how each component interacts with each other. The user will connect the camera to the computer and the software will verify that the camera is connected. The user will then interact with the system by the way of our Intel Realsense camera and GUI. The gestures that the camera picks up will then be interpreted through the Nuitrack SDK. The gestures recognized by the SDK will be mapped to specific actions in the backend that will allow the user to interact with the system.

The Intel RealSense camera and the Nuitrack SDK will allow us to track and map user gestures while Unity will allow us to create both the GUI and backend processes. Together, we were able to create gesture-controlled software that many people can use, as is part of our client’s requirements.

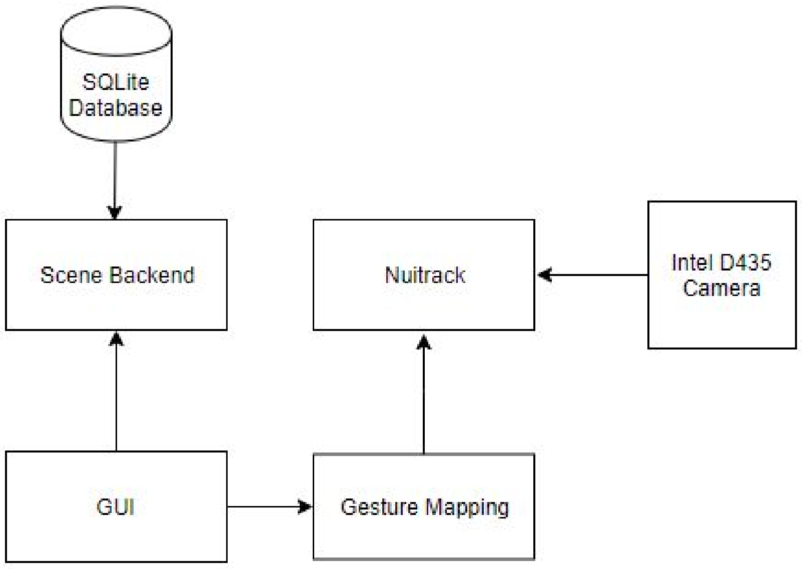


Figure 2: Architecture Diagram

A general overview of the system's flow is depicted in figure 2. The GUI and backend are created together using Unity and C# in Visual Studio. At first, the user will be able to select options to either log in, register, access modules, or learn more about the team. The login and register portion of the software will be completed by using SQLite, an embedded SQL database engine. Since Unity uses scenes to host properties of a specific game, our modules are configured as different scenes. Another great feature that Unity provides is the ability to import external scenes into a current project. This allows future programmers to load new content into the modules.

The local SQLite database is used to store usernames, passwords, and any scores or progress made through modules. This will be done using three connected tables that will interact with each other to keep track of the high scores, what game it corresponds to, and the individual who achieved the high score.

The Intel D435 camera and Nuitrack is used to read user positioning and gestures. These gestures are then mapped to specific actions. For example, a closed fist indicates a mouse click and can be used to click buttons, which allows the user to navigate through the menu. These actions are specified through backend scripts.

The system the team ended up building is similar to the system envisioned by the sponsor and the team at the end of planning and requirements acquisition. The team ended up changing some of the software and methodology that was originally planned for the system in order to be more compatible with the hardware the team chose. We initially planned to use the Intel RealSense SDK however, we switched to Nuitrack by the second semester. This change gave us more methods for gesture interpretation. The database system also changed from SQL to SQLite as the project progressed. Unity was still used to provide the GUI and C# was used to create the script files. The changes in software improved the development process for the project allowing for better compatibility of all the structures that were incorporated within the project.

5. Testing

We have done three different testing strategies: unit, integration, and usability testing. These strategies ensured that our product was fully functional and met our client's expectations. We first broke the system into smaller, individual units in order to test each unit of source code works. Unity offers its own test framework, TestRunner. TestRunner requires classes with functions that test specific methods of our source code. Our team has created a total of 5 test classes to test the Registration, Login, and DBManager classes along with methods for gesture recognition and methods implemented in the provided modules. Functionalities provided by Unity and Nuitrack, such as clicking buttons, changing scenes, object collisions, etc., were not tested. This decision is based on the assumption that these methods have already been thoroughly tested by the providers before public distribution.

Unit testing looks at individual components of the software to confirm that each unit performs as expected. To ensure the functionality of our system, we have created several unit tests. We have created these tests using Unity's built-in Test Runner module. This allowed us to write tests in C# and run them in the Unity editor.

Test	**Equivalence Partitions**	**Boundary Values**	**Sample Input/Action**	**Expected Outputs**
UserFound()	N/A	N/A	User stands in front of the depth-sensing camera.	User's body is interpreted with Nuitrack's SDK. A text will appear on the screen saying "User found"
HandTracking()	N/A	N/A.	User stands in front of the depth-sensing camera and waves their right hand around the screen	User's hands are interpreted with Nuitrack's SDK. A text will appear on the screen showing coordinates of hands at a specific moment in time
ItemDestroyed()	N/A	N/A	User stands in front of the camera and touches objects on the screen.	User's contact with the objects will cause them to disappear, triggering a text on the screen displaying "Destroyed"

Here is an example of one of the unit testing that we performed on our system. The Gesture recognition portion of the system is responsible for ensuring that the user will be able to use gestures as a means of navigation through the system, as well as interact with the different modules available. Therefore, the items that we have tested focused on the backend's ability to recognize and interpret these gestures.

The purpose of integration testing is to make sure all of the units are capable of interacting and working together. As the user navigates through our system, we want to ensure that data is not lost across different aspects of our project. We want to make certain that when the different modules or our system are combined, they deliver optimal results. In order to prove the effectiveness of our system, we performed integration testing. During our testing, we incorporated the different modules combined as to exhibit functionality.

Integration Test	**Sample Input**	**Expected Output**
Math activity can be used with gestures	User will stand in front of a depth-sensing camera for body recognition. As equations are shown on the screen in a timely manner the user will need to destroy the game object with the right answer.	As game objects with the correct answer are destroyed. The system will validate the answer and if it is the correct one, it will increment the score.

Here we have an example of one of the integration testing that we performed on our system. This mainly focused on the Math Game portion, which works with both our database as well as gestures.

Finally, the team finished off the testing phase with usability testing. In order to decide how the usability for our system should be tested, we had to take into consideration who will be using it. The purpose of this phase was to ensure that users can easily learn how to use our product with little to no difficulty. Because our system is meant for users of almost any age, the learning curve had to be as simple as possible.

Purpose:	The user will find the system engaging and enjoyable
Task:	Run through the respective module in your age group, try at least one of the activities
Plan:	Taking at least two individuals for each of the modules we have developed and asking them their opinions on the content. Posing the right questions based on graphics and gestures.
Survey	● Did you find the material engaging and informative? ● On a scale of 1-10 how enthusiastic were you to continue? ● Would you use this system again?

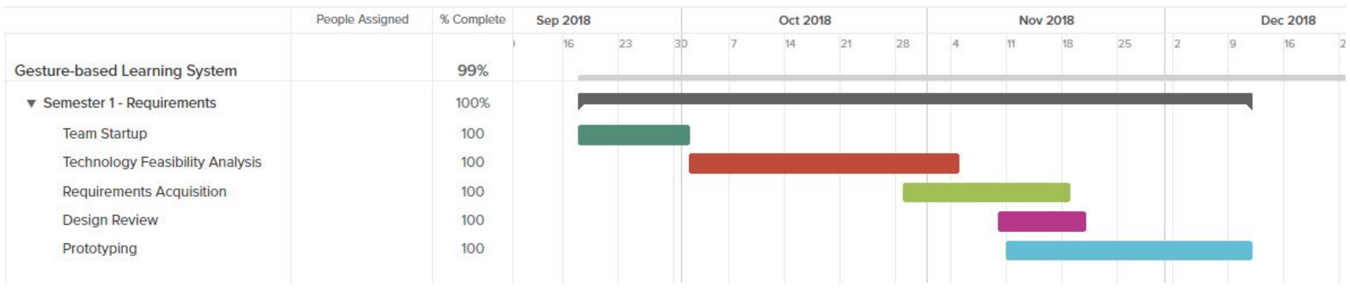
Pictured here is one of the examples of usability testing that we performed. Our usability testing primarily consisted of test groups of different ages who interacted with our system. This included actions such as navigating the menus, creating a new account, and playing at least one of the activities provided. Ease of navigation and overall satisfaction was what the team was testing for. We have created two surveys for navigability and overall satisfaction to give to users testing our system.

The above testing provided information on bugs and user experience that the team used to refine the system. The unit and integration testing showed a few bugs that interfered with the math game, database scores, and score counter for the interview game. This resulted in the team reviewing the logic of the interview game to ensure that the counter works for all interviewees. The database system was fixed to ensure that high scores were being associated with the correct username during play of the math game. Then, the math game code was revisited to make sure that items were destroyed when users interacted with them.

The feedback from our user testing included providing better tutorials, editing the interview game button interactions, and changing the spacing and text size for some of the buttons. The team implemented the feedback for the interview game by disabling the continue button when the user has to choose a person before moving on to the next question. We also display a prompt to choose a person, so that the user remembers to click on a name. The written tutorial also had a quick script change to make it easier to understand controls. The button changes suggested by the survey were also changed in the system.

6. Project Timeline

During the first semester of capstone, the team first established roles for each member before focusing on technology feasibility and requirements acquisition. After deciding on the technologies we used and working on detailing the exact requirements of our system, we created a thorough design review. By the end of capstone, we had a rough prototype of the system that proved the feasibility of our project.

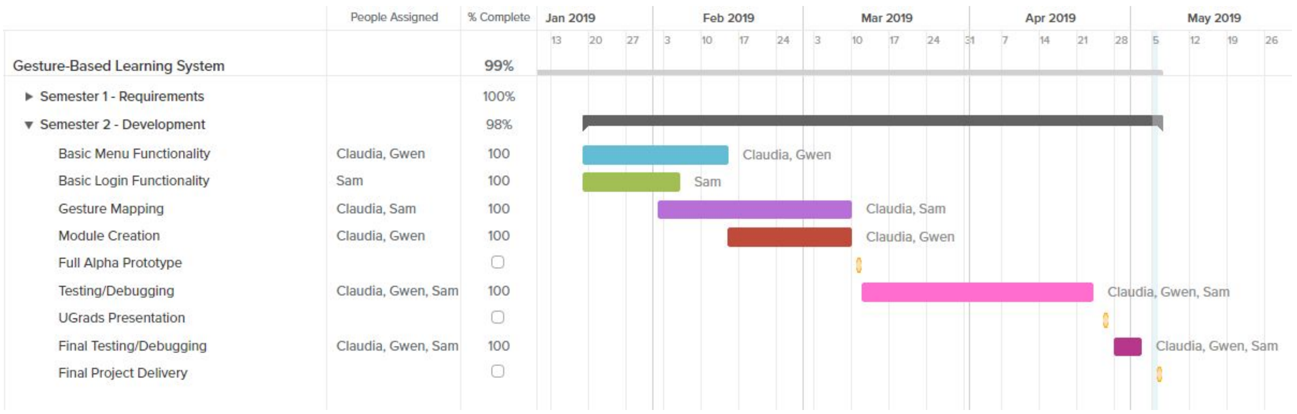


	People Assigned	% Complete
Gesture-based Learning System		99%
▼ Semester 1 - Requirements		100%
Team Startup		100
Technology Feasibility Analysis		100
Requirements Acquisition		100
Design Review		100
Prototyping		100

Figure 3: First Semester Schedule

Figure 3 depicts the first semester of capstone, from the team startup phase to the prototyping phase.

Our team focused on product development during the second semester. Because this product was being built from the ground up, we broke the base of this project down into basic login and menu functionality and gesture mapping. After the groundwork of our system was laid out, we worked with our client to decide what modules should be created. We have created a simple math game for groups K-5 and an interview game for users in college or in the general community. This was done before our full alpha prototype on March 11, 2019. Our team then focused on thoroughly testing the system before the undergraduate symposium. We are currently finishing some last minute adjustments before finally delivering our product to our client.



	People Assigned	% Complete
Gesture-Based Learning System		99%
▶ Semester 1 - Requirements		100%
▼ Semester 2 - Development		98%
Basic Menu Functionality	Claudia, Gwen	100
Basic Login Functionality	Sam	100
Gesture Mapping	Claudia, Sam	100
Module Creation	Claudia, Gwen	100
Full Alpha Prototype		☐
Testing/Debugging	Claudia, Gwen, Sam	100
UGrads Presentation		☐
Final Testing/Debugging	Claudia, Gwen, Sam	100
Final Project Delivery		☐

Figure 4: Second Semester Schedule

Figure 4 is a Gantt chart that depicts the second semester of capstone, from creating basic functionalities to the final project delivery.

7. Future Work

One of the goals our sponsor has for this project to continue on after capstone. Since this project will be a system used in the Flagstaff community there is still future work to be done. Some of this future work includes further module development, converting to a cloud database, and the expansion of gesture support.

Through our capstone, we developed modules as a platform to hold future STEM content. With this project we added a small amount of content to show proof of feasibility however, there is future work to be done in expanding existing modules This can be through adding content to modules that the team has already created or by adding new modules and providing new content for those modules. The details of the content would be up to the developer and sponsor of the project but would be stored in the navigation system.

Another item that can be worked on in the future is developing a cloud database. Since the focus of our capstone project was to develop a local database for one system, we understand in the future there will be several of these systems in the community. To ensure that personal profiles can be accessed by any of these systems that are around the community, we would like to convert into using a cloud database instead of the local one that is currently being used.

The gesture recognition system used for this project, Nuitrack, was very limited on the amount of gestures we could use. For new gestures to be configured, developers would essentially need to make their methods from scratching using Nuitrack’s API. This means that there is a potential future work to be done here. When this project expands there should also be the expansion of gesture support as far as what is allowed. Currently, we only have clicking and skeletal recognition, but in the future developers could possibly add additional gestures like swiping capabilities.

Overall, this semester our team worked on providing the framework to hold this gesture-based learning system and it will be up to future developers to continue our work.

8. Conclusion

Scikids is a team that was formed with a goal to provide STEM outreach services to the Flagstaff K-12 and community individuals alongside our sponsor Elizabeth Glass. We have created a gesture-based learning system that includes different modules, interactive fun, and personalization. The current problem is that STEM is a quickly growing field without enough individuals going into it. The solution in the Flagstaff community is to create this gesture-based learning system to be easily accessible and portable for K-12, college, and community members. Our client has decided on this solution when seeing how effective gestures can be in helping users retain information and how there is a lack of gesture-based technologies that are currently available. This gesture-based system allows users to interact with the system with gestures such as open and closed fists as well as grab-and-drag movements (to substitute click-and-drag gestures). They will be able to navigate the menu and engage in age-based modules through these gestures.

9. Appendix A

9.1 Hardware

Our team used the Intel RealSense D435 camera to help us develop this product. These cameras are compatible with Nuitrack (discussed in section 9.2), along with a variety of other cameras. The team has purchased 3 cameras for $179 each. For other compatible cameras, please visit the Nuitrack website (nuitrack.com).

Our team has also used Windows 10 to develop this product however, Nuitrack and Unity (discussed in section 9.2) are compatible with a wider range of systems. Nuitrack is compatible with Android ARM v7 and Linux x64. Unity is compatible with 64-bit versions of Windows 7 SP1+, 8, 10 and Mac computers with macOS 10.12.6 or higher.

9.2 Toolchain

Our team used Unity as the primary development platform. Unity is a popular and versatile game development engine that allowed the team to quickly create GUIs with backend support. During development, the team decided on using Unity version 2018.3.5. Older versions cannot support newer versions, so we decided to not update Unity during development to eliminate confusion between team members. Unity uses C# for backend processes, so we decided to use Visual Studio 2017 to create the backend.

We have also used Nuitrack, a 3D skeletal tracking middleware, to help us map gestures to actions. Nuitrack offers compatibility with a selection of cameras and is continuously updating their software, so the team is hopeful for advancements in gesture mapping in the future.

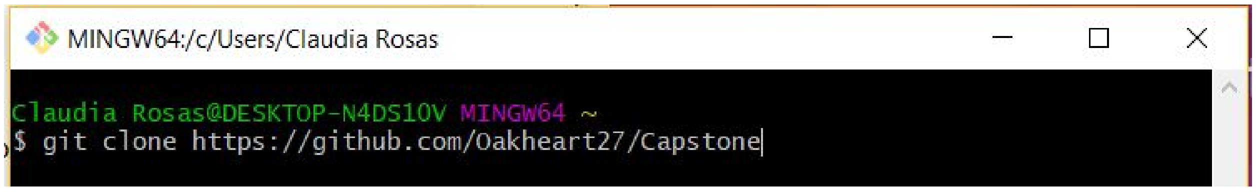
Our team also used GitHub as a means of quickly sharing development between team members, keeping track of issues, and ensuring that work between members merges together.

Finally, we have implemented a local SQLite database to save user information, scores, etc. We have done this to further encourage users to continue using our system.

9.3 Setup

9.3.1 Setting up the Environment

1. *Clone the project from GitHub using GitBash.* To do this, create a new folder on your system and initialize an empty Git repository using “git init”. Then, use a git clone command with https://github.com/Oakheart27/Capstone



```
Claudia Rosas@DESKTOP-N4DS10V MINGW64 ~
$ git clone https://github.com/Oakheart27/Capstone
```

Figure 5: git clone

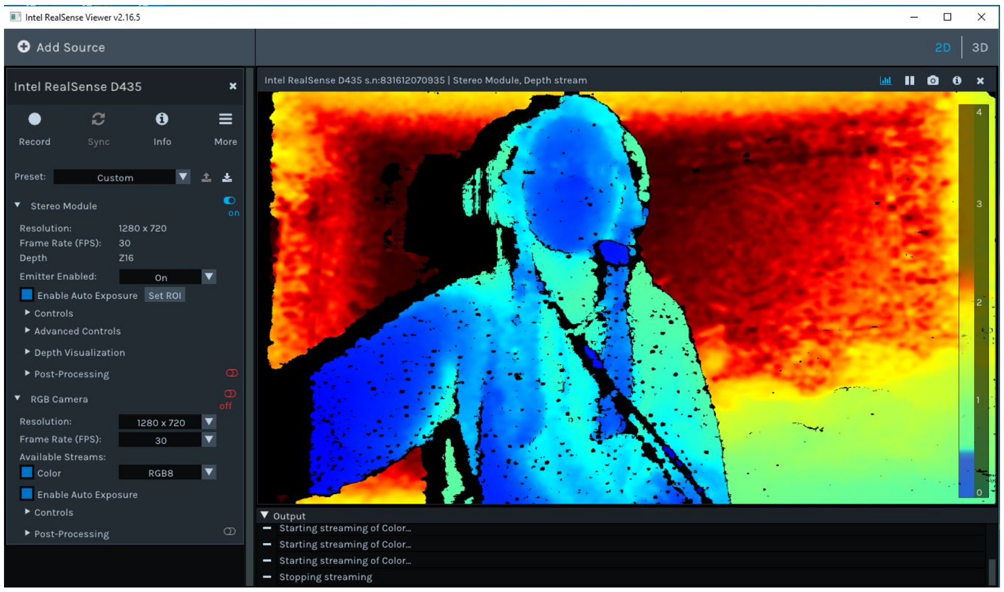
2. *Download Unity and Visual Studio*. Go to Unity’s website to download the Unity installer. Our team used Unity version 2018.3.5 to develop this product. The product can work with future versions of Unity, however teams who work on this project should be careful when updating Unity while developing. When downloading Unity, the Unity installer will offer the option to download Visual Studio 2017 as well. Make sure this option is checkmarked.
3. *Set up the Intel RealSense SDK 2.0* . This step is required for Nuitrack’s software. The download link can be found at https://github.com/IntelRealSense/librealsense/releases. This download also includes Intel’s Depth Quality Tool and the Intel RealSense Viewer.
4. *Set up the Nuitrack SDK*. Our team has already purchased three copies of Nuitrack PRO for Windows, and they are activated with the provided cameras.
    a. Download Nuitack from http://download.3divi.com/Nuitrack/platforms/

b. Install Nuitrack for your platform. For Windows 10, create an environment variable with the name of NUITRACK_HOME and value of <*install-folder*>\nuitrack. Then, add <*install-folder*>\nuitrack\bin to the PATH environment variable. For other operating systems, please visit the Nuitrack API.

5. *Open project through Unity Hub*. Unity projects are typically accessed through the Unity Hub. If the project is not automatically displayed in the hub, click Open and navigate to the folder than contains the project.

9.3.2 Setting up the Hardware

As stated prior, the team has purchased 3 Intel RealSense D435 cameras. Connect the camera into a USB-3 port on your computer before testing the product (either in the editor or in runtime). To ensure that the camera is properly connected, open the Intel RealSense Viewer (see section 9.3.1, step 3).



Intel RealSense D435 s.n:831612070935 | Stereo Module, Depth stream

Preset: Custom

Stereo Module on
Resolution: 1280 x 720
Frame Rate (FPS): 30
Depth Z16
Emitter Enabled: On
Enable Auto Exposure Set ROI

RGB Camera off
Resolution: 1280 x 720
Frame Rate (FPS): 30
Available Streams:
Color RGB8
Enable Auto Exposure

```
Starting streaming of Color...
Starting streaming of Color...
Starting streaming of Color...
Stopping streaming
```

Figure 6: Intel RealSense Viewer

If the viewer can view both a stereo module and the RGB camera, as seen above in figure 6, the camera is properly connected. Another indication of the camera properly working is a small red light in the biggest lens.

9.4 Production Cycle

9.4.1 Importing an External Scene Into Project.

First, export the scene and all dependencies. On the upper menu of the Unity Editor, go to Assets -> Export Package.

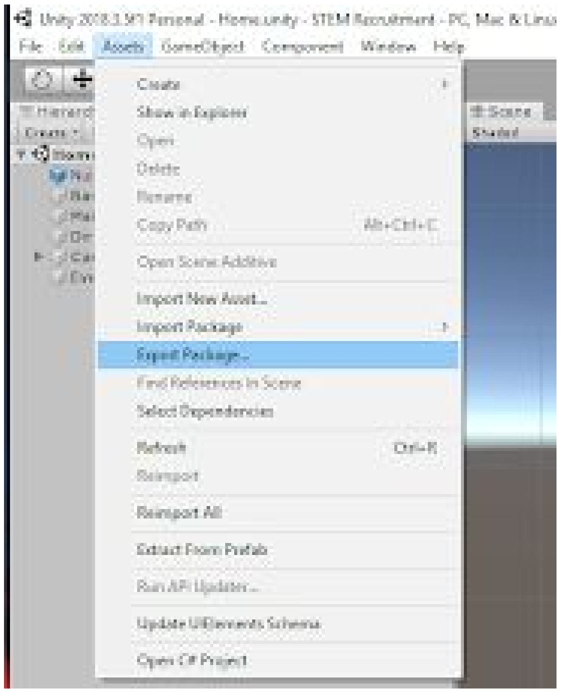


Figure 7: Export Package

Make sure to include all dependencies of the scene (prefabs, scripts, etc.). Then, click “export” and save the Unity package in the desired folder.

To import the scene into the base system, open the STEM Recruitment project and go to Assets -> Import Package -> Custom Package.

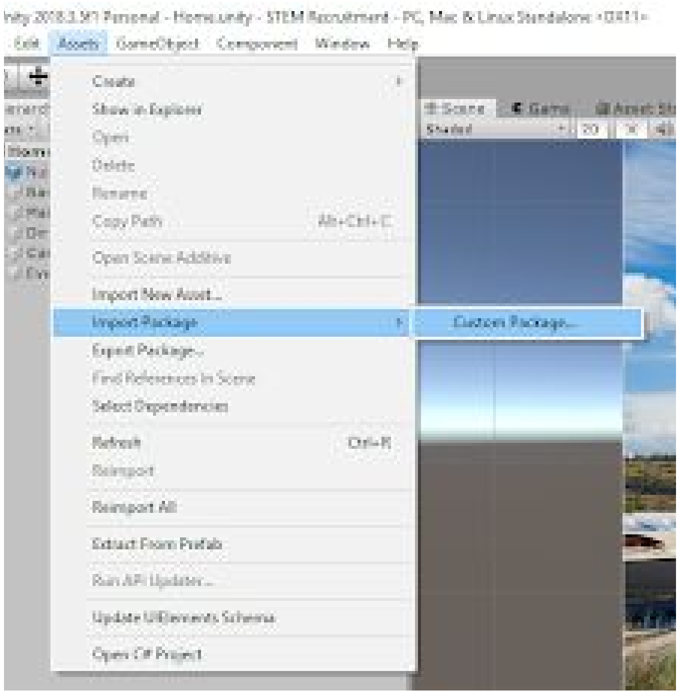
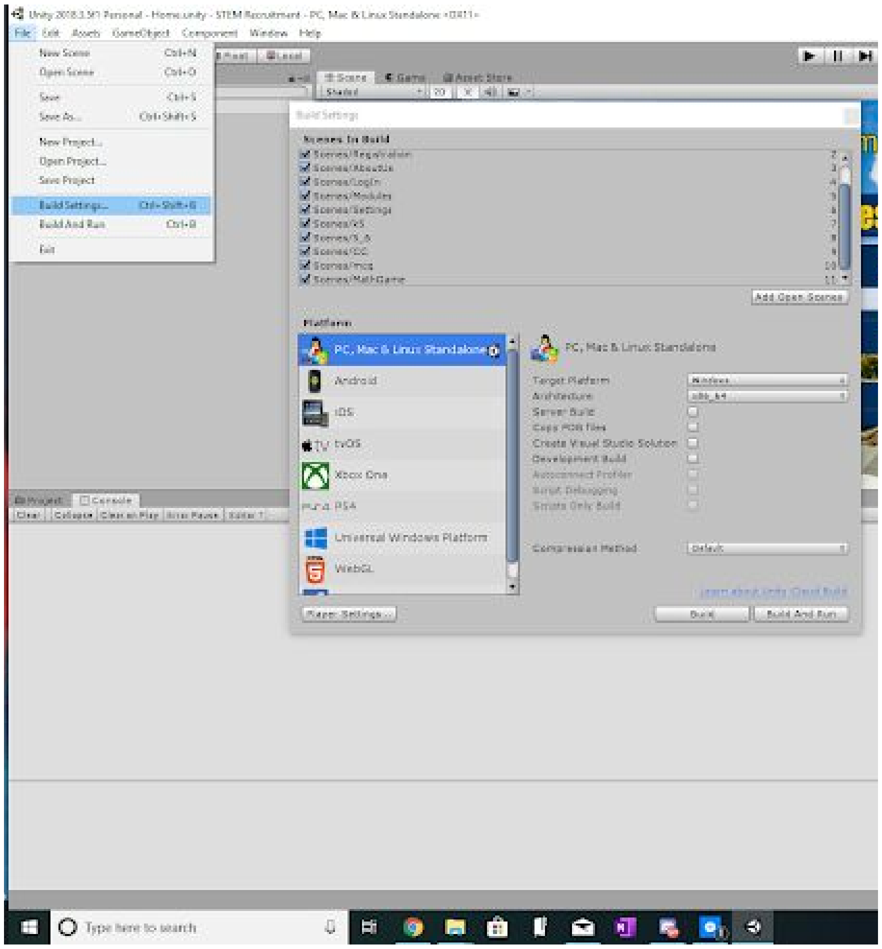


Figure 8: Import-Package

Navigate to your Unity package and import it into the project. Unity will automatically organize your dependencies into their respective folders. In order for the project to have access to your scene, it must be added to the build. Open your scene in the editor and go to File -> Build Settings. Then click “Add Open Scenes”.



Scenes In Build

- ✓ Scenes/Registration 2
- ✓ Scenes/AboutUs 3
- ✓ Scenes/LogIn 4
- ✓ Scenes/Modules 5
- ✓ Scenes/Settings 6
- ✓ Scenes/RS 7
- ✓ Scenes/S_A 8
- ✓ Scenes/DC 9
- ✓ Scenes/mcq 10
- ✓ Scenes/MathGame 11

PC, Mac & Linux Standalone

Target Platform	Windows
Architecture	x86_64
Server Build	☐
Copy PDB files	☐
Create Visual Studio Solution	☐
Development Build	☐
Autoconnect Profiler	☐
Script Debugging	☐
Scripts Only Build	☐
Compression Method	Default

Figure 9: Build Settings

9.4.2 Connecting Scenes Through a Button.

In the Assets/Prefabs folder, developers can find a button prefab that they can use to connect scenes. To add the button onto canvas, first open the scene from the STEM Recruitment project that you want your scene to be connected to. Drag the myButton prefab from the Assets/Prefab folder onto the canvas of the scene. This prefab contains a HoverToClick script that works with both a mouse and gesture recognition. Note that you can also change the color, text, etc. of the button prefab.

In the Inspector, add a new OnClick() task. The OnClick() task needs to reference the Canvas (as that is where the script to load a scene is located). Call the function

LoadSceneThroughButton.changeScene, and enter in the name of your scene into the input parameter.

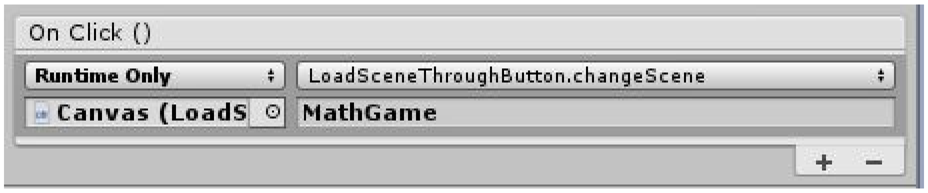

On Click ()
Runtime Only | LoadSceneThroughButton.changeScene
Canvas (LoadS | MathGame

Figure 10: Managing OnClick Events

9.4.3 Adding Gestures to New Scene

In the Assets/Scripts folder, a HandCursor script is available for developers to use. This script allows users to navigate through menus and click on buttons using gestures, as it maps the user’s hand to the mouse cursor. Add the HandCursor script to a button. This can be done by either finding the HandCursor script in the “Add Component” option of the Inspector or by dragging the script onto the button in the hierarchy menu.

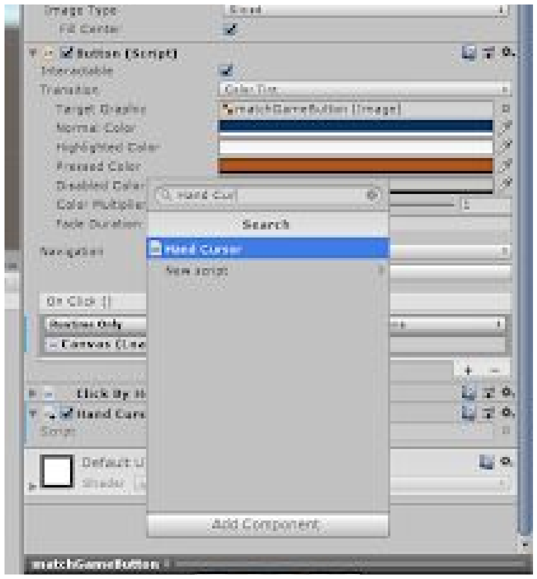


Figure 11: Adding HandCursor

Add the NuitrackScripts prefab onto canvas, which can be found in Assets/NuitrackSDK/Prefabs. Add it to the scene and make sure that the Gestures Recognizer Module and Hands Tracker Module are both checked in the Inspector.

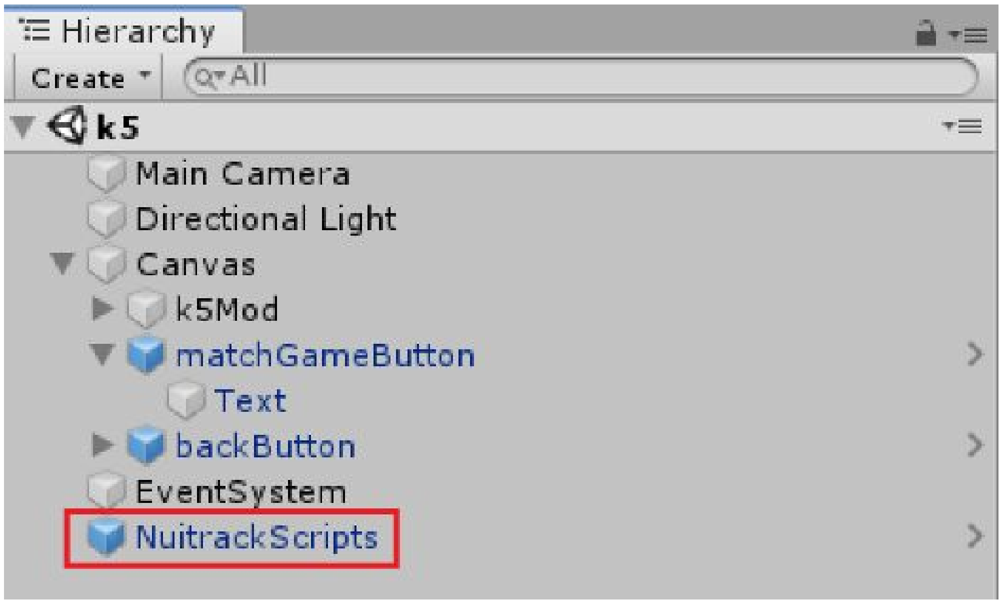
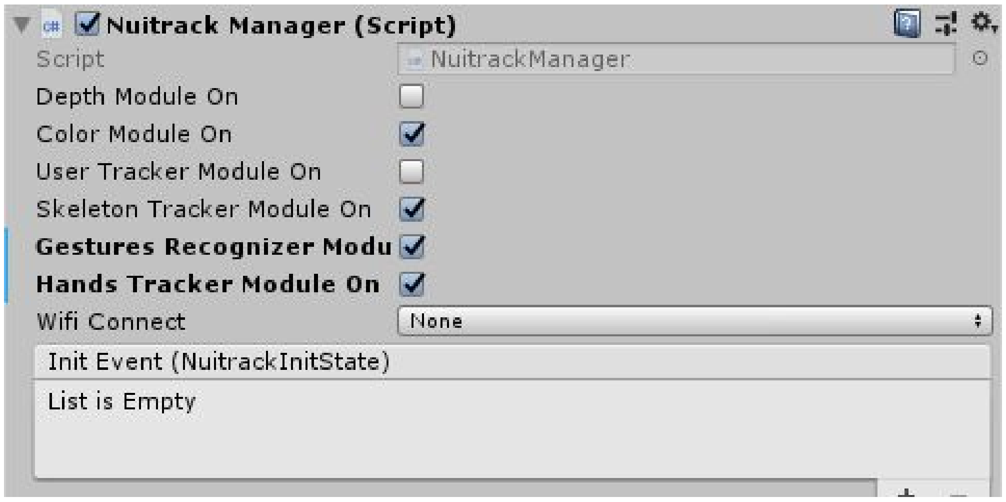


Figure 12: Adding NuitrackScripts Prefab



Nuitrack Manager (Script)

Script	NuitrackManager
Depth Module On	☐
Color Module On	✓
User Tracker Module On	☐
Skeleton Tracker Module On	✓
Gestures Recognizer Modu	✓
Hands Tracker Module On	✓
Wifi Connect	None

Init Event (NuitrackInitState)

List is Empty

Figure 13: Nuitrack Manager

To add more gestures, please visit the Nuitrack API. Here you can find tutorials using Unity along with a list of their classes, modules, etc.