GNG 1103

Design Project User and Product Manual

Waste Management Solution: Geco

Submitted by:

Synthetic Solutions C01

Ben Leeder, 300073282

Lilly Ferrier, 300104934

Camila Escalante, 300073687

Angie Orabi, 300071028

April 18th, 2021

University of Ottawa

Table of Contents

List of Figures

List of Tables

List of Acronyms and Glossary

Table 1. Acronyms

Acronym	Definition
USB	Universal Serial Bus
UI	User Interface
VS	Visual Studio
TM	Teachable Machine
URL	Uniform Resource Locator

1 Introduction

This User and Product Manual (UPM) provides the information necessary for anyone who wishes to recreate this product to effectively use the application Geco and for prototype documentation. The purpose of this project is to encourage the habit of recycling amongst people, especially in the Ottawa region. The product proposed consists of a mobile app and a website to enable efficient recycling methods. The application was created in Unity and the website incorporates Teachable Machines.

This manual lists all the components of the product, which includes the app and the website and describe the design process and the illustration behind its execution. In addition, there are detailed explanations of every component of the app and the website. The explanations include how to operate the app and website, the design approach, the design process of each segment, as well as how to recreate the product for anybody who may be interested. Other details such as the project plan, bill of materials, list of equipment, troubleshooting, support and most importantly the instructions on how to use the app and website are also part of the user manual. Moreover, the user manual contains many figures depicting multiple features of the app and website.

2 Overview

The main purpose behind the creation of this product was to design a user-friendly, interactive application that helped educate and inform users about proper recycling. The reasoning for the product was significant because it directly influenced what was created in pursuit of meeting the desires of the client.

The fundamental needs that were met in the creation of this app include:

- An accurate item scanner
- User friendliness
- An interactive section
- Recycling education and information
- An application that can be easily expanded

In order for the user to have a pleasant experience with the application, several basic requirements were met. Firstly, it was important for the app to be easily navigable and esthetically pleasing. It was also vital to have working sections of the app, especially regarding the minigames and item scanner. Education pertaining to sustainable recycling practices was also emphasized to ensure that the user would gain knowledge from the application.

The product that was created is unique for several reasons. Firstly, it has a scanner image database of approximately 3500 photos, that can easily be added to at any time. The scanner is also its own entity, so it can be used independently from the rest of the app if desired. Regarding the UI, it is an easily navigable application with a natural theme that is comprised of multiple components including two games and an educational page.

The main page for the final product and the final website for the item scanner can be seen in Figures 1 and 2 below respectfully:

Figure 2- Item Scanner

Figure 1- Home Screen

Our product differentiates from other competitors in the market. Firstly, due to its ability to scan items using AR technology. Secondly, its abundance in being able to scan almost everything without being confined or limited to an item's barcode, the ultimate material definer essentially. What also makes this app stands out is its content and texture which accommodates users of all ages. Anybody with basic technological literacy is guaranteed to be able to navigate their way around this app with ease. Finally, the information we provide for our application comes from the Government of Canada's website, making it reliable and accurate.

2.1 Cautions & Warnings

Since this is a mobile application in its early stages, there are no cautions or warnings to provide at this time. However, there are some limitations to the application. In its current state, the application follows the City of Ottawa recycling practices and may not be applicable in other cities, provinces, and countries.

3 Getting started

For the Application to be set up for use, the app must be built to whatever platform it is going to be used on. To be built for Android devices, first the unity project must be built to that specific project. To do this first go to Files> Build Settings > Android > Switch platforms.

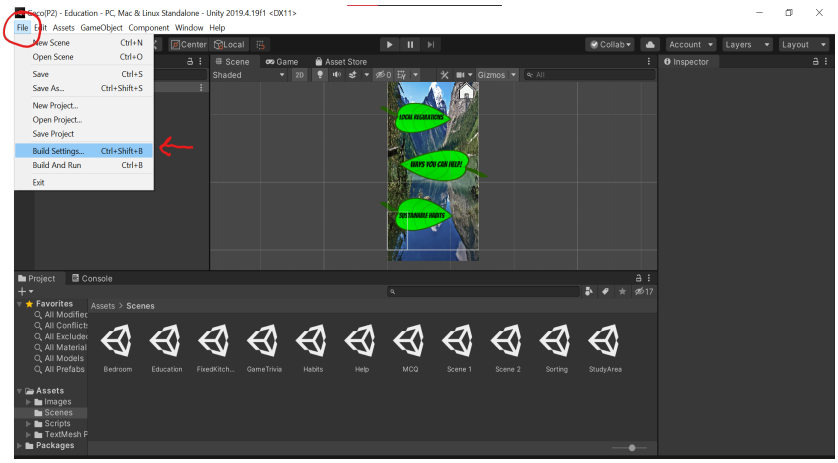

Figure 3-Build Settings Location

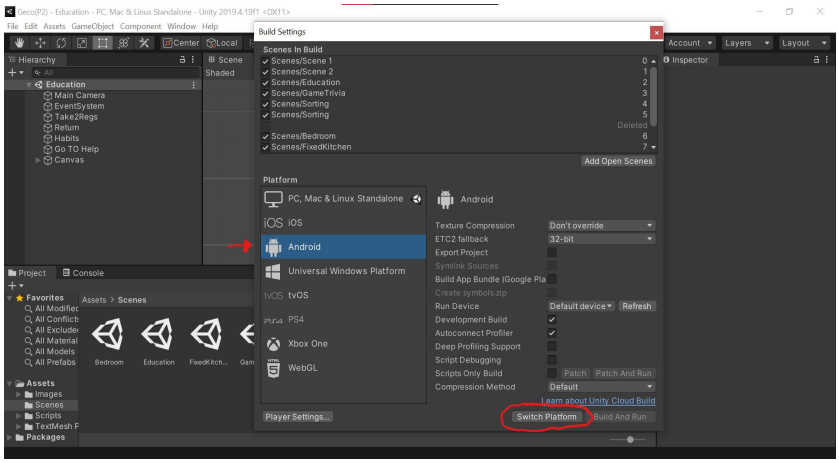

Figure 4- Build Settings

Once the program is built to a specific platform it can be exported to a specific android device for testing. To do this plug the device into the computer via USB and select it from the run

device in the drop down menu(in the Figure Below the device is called Samsung SM). Make sure Development Build and Autoconnect Profiler Buttons are selected. Once Build and Run is selected, the app will be compiled and open on the selected device.

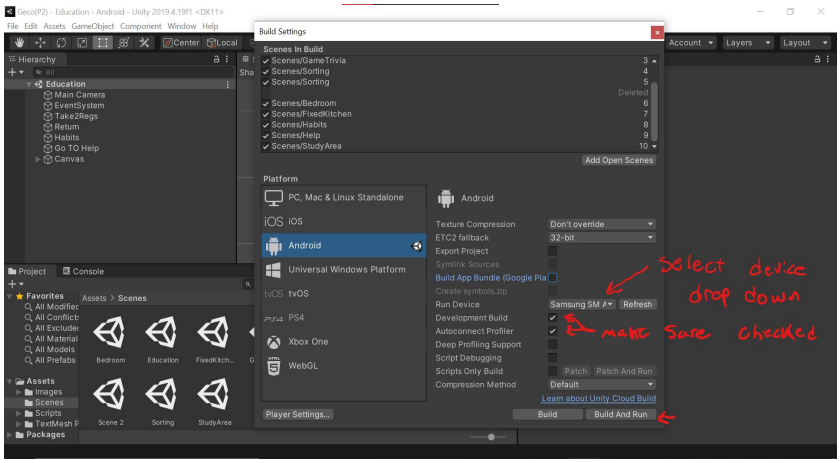

Figure 5-Test Settings

To add the app to the google play store, first make sure the project is built for Android. Next open Build Settings> Player Settings. From here select Player> Publishing Settings> Keystore Manager.

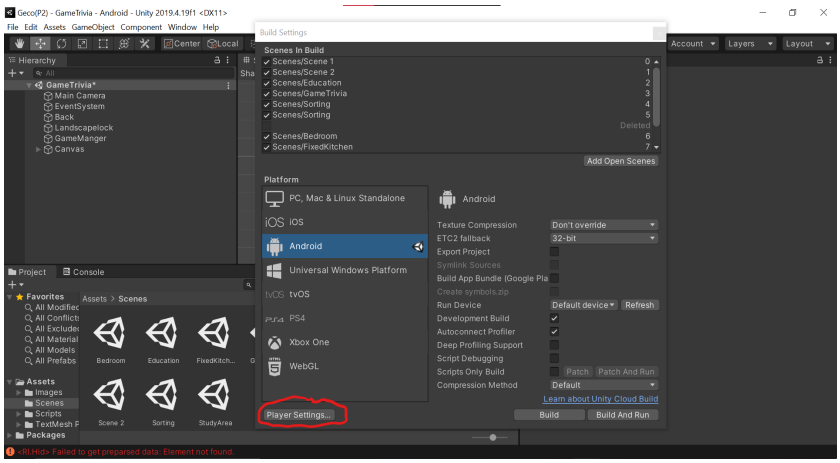

Figure 6- Player Setting Location

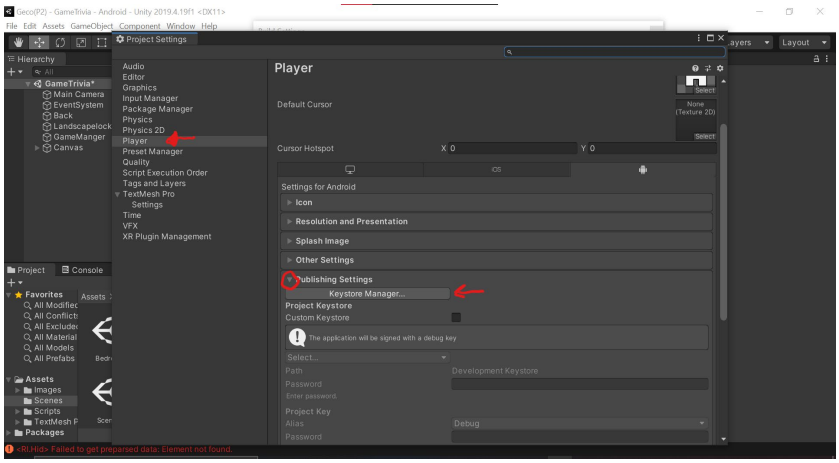

Figure 7-Player Settings

Next select Keystore and can choose the save location. It is recommended to save it in a folder with your project. This Key will be needed to publish the application. Create a password for the key and have that saved as well. Create another key with the same password as the one used before. Then preform a build of the project from the build settings.

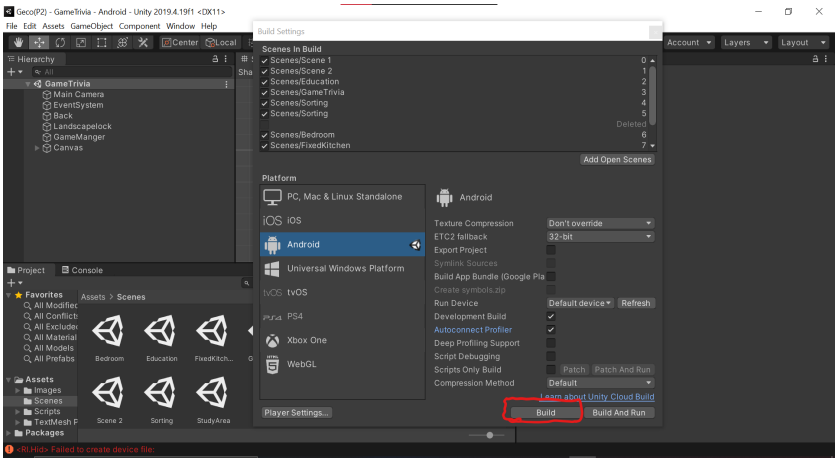

Figure 8-Build Settings Build

After this is done go to https://play.google.com/apps/publish and either log in or create a google play developer account. Click on +Add New Application to be able to add Geco. Fill out the

required boxes and then select the APK button. From there the APK file of the build can be uploaded to the PlayStore. Once the Build is approved, it will be added to the PlayStore.

For Release to Apple devices, some slight changes must be made. A MacBook and XCode are needed for testing and implementation of the app for IOS devices. For full guide on how to release for IOS please follow the guide linked below. https://learn.unity.com/tutorial/publishing-for-ios?signup=true#

The item scanner does not need any set up because it is a public website and is therefore simply connected with the link that is in the application. However, if editing the image database or website in any way was desired, then certain programs would need to be downloaded in order to make alterations to the code. Firstly, one would need visual studio, visual studio code, Git, and GitHub. Visual studio and visual studio code can be downloaded via https://visualstudio.microsoft.com/. Git is a bit more complicated to install and involves coding in the terminal or command line of your computer. The instructions about how to do this can be followed at https://git-scm.com/downloads. Regarding GitHub, one would just need to make an account at https://github.com/.

After these are set up, there are two options about how the website information could be accessed and edited. Firstly, one could obtain the existing information connected to the item scanner website by downloading it from the websites GitHub repository. First search Ferrlill.github.io in GitHub and press the green button seen in Figure 9 and download the ZIP file.

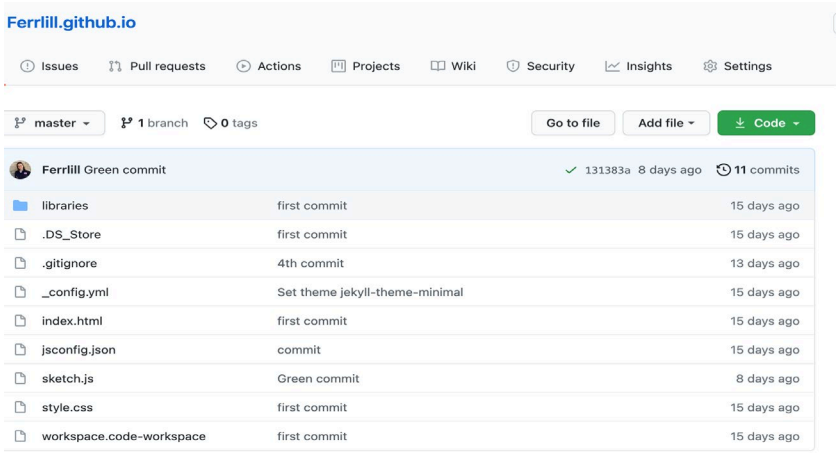

Figure 9- GitHub Download

That file then contains all of the existing code connected to the website. After this, simply unzip the file and open VS code and open the unzipped file with it. The code could then be altered in the VS code desktop app. The downside of this option is that the user would need to create a new repository in order to access their updated website. The other option is to simply edit the code by proposing changes to the existing repository. This is done using pull requests and is outlined here https://docs.github.com/en/github/managing-files-in-a-repository/editing-files-in-another-users-repository.

If editing the image database was desired, one would need to use Teachable Machine by Google (https://teachablemachine.withgoogle.com/). If one wanted to restart with a new image database then they could just begin by uploading images, if this was the case then minor changes would need to be made to the code so that it was referencing the correct URL and image category. If one wanted to simply build on to the existing image database, then the present Teachable Machine files and URL. These files are currently one a Google Drive of one of the members, so that would need to be shared in order to make alterations to the database. To upload images, simply click on the upload button for the desired category and add the images. The TM page can be seen in Figure 10 below.

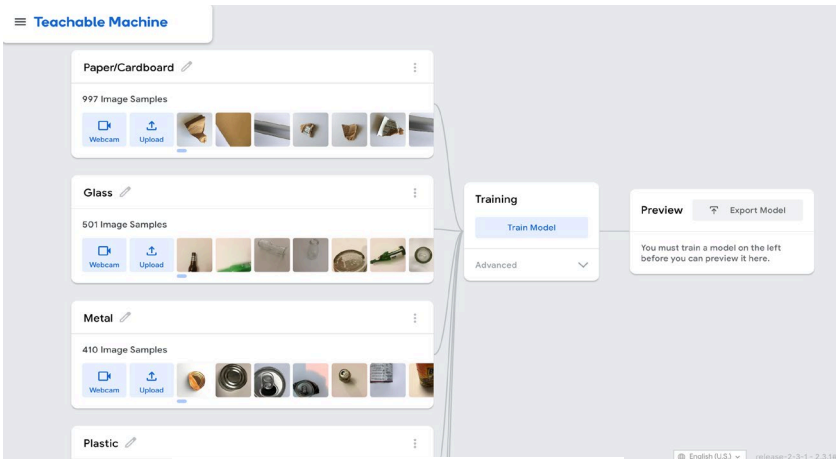

Figure 10- Teachable Machine Page

Once the changes were made, one would train the model using the button below and then export the model. The only thing that would need to be altered in the code for this edit would be the new URL that would be attached to the updated TM database.

3.1 Set-up Considerations

Some considerations to be considered is to make changes to the Application is that the Unity Editor must be downloaded on a computer. To make release versions for Apple devices, one must both have a MacBook as well as the program XCode. Since the item scanner is a website, it will work on all devices with camera access.

3.2 User Access Considerations

Currently there are no user restrictions except for language and camera access. The current app and website were only built to display text in English. No current system exists in the project to change language settings. In order to overcome this, the text could be translated, and another version of the app could then be made available. The item scanner only has the ability to use the devices camera to scan items. This means that an item cannot be scanned if the device does not have

a camera, or the scanner does not have camera access. In order to overcome this, code could be added so that the user would have the option to upload an image for processing.

3.3 Accessing the System

Everything for system access can be done from the unity project file with no special care needed such as passwords or user IDs. If UnityHub is downloaded the application should operate and be editable without difficulty. The only special action required is the project currently is built in Unity 2019.4.19f. To access the project this version must be installed on the desired device. This version can be gotten from the Unity website under older editor versions. Once this version is downloaded the project can be updated to more recent versions of Unity.

3.4 System Organization & Navigation

The system is organized starting from a central home screen. From the home screen, users can take many pathways to the rest of the app. The scanner is connected based on the Camera Button. The Games are connected based on the game controller and the Education is connected by the book icon.

From the game selection page there is a home icon, which will return users the home page. On the Games selection page there are the option to go to Trash Trivia or Recycling Madness. Trash Trivia has a back button which allows users to return to the game selection page.

After selecting Recycling Madness, users can select a level or click on the home button to return to the home screen. They can also click on the back button and return to the game selection screen. After selecting a screen, each level features a back button to return to level selection.

Once a user selects the book icon the education page opens. There is a home button in the corner to allow users to return to the home screen. The Local Regulations button will take users out of the app to their chosen browser. The other two buttons will open their respective infographic and they both have a back button to return to the education page.

3.5 Exiting the System

Users can stop both the app and scanner in the same way as any application or webpage. They can just close the app like any other and the application will stop running. The scanner, users can close the webpage and it will stop running.

4 Using the System

4.1 Application

The application features a home screen, two mini games, and an education section.

4.1.1 Home Screen

The home screen has features three buttons which the users can provide input with. The Camera Icon when clicked will take users to the website where the Item Scanner is hosted. The Game Controller Icon when clicked will take users to the game selection screen. Finally, the book Icon will take users to the education page.

4.1.2 The Games Page

The game Selection page allows users to select between the games Trash Trivia and Recycling Madness. When clicking on either button, the associated game will open.

4.1.2.1 Trash Trivia

After clicking on the Trash trivia button, a True or False game will open, and the screen will autorotate to landscape mode, and a question will be randomly selected for the user to play. The user can then click on the true or false button. After clicking on either button will then preform an animation and the user will be informed if their response was correct or not. In the top left corner of the app is a button that will allow users to navigate back to the game selection screen.

4.1.2.2 Recycling Madness

Once the Recycling Madness button is clicked (in the games page), a scene panel will open showing three different locations: kitchen, bedroom, and study area. The user will have to choose from the three options to start playing. After selecting the desired location, the game will start

showing approximate ten items to be dragged and sort into their correct recycling bins. As the item is dragged to a bin, it will inform the user if their selection was correct or not. If the item is sorted correctly, a “correct” sound effect will automatically play, while if the answer is wrong, an “incorrect” audio will play, returning the item to its initial place. Both the game and the scene selection pages have a *back* button, in the top left corner, that will allow the users to return their respective previous page.

4.1.3 Education Page

When opening the education page, three buttons are presented to the user. The first button, Local Regulations, when pressed will open the Ottawa recycling page. Here users can see everything trash and recycling related in the city of Ottawa. The second button, Ways You Can Help, will open an infographic with a few practices people can make to help with recycling and the environment. Finally, the last button, Sustainable Habits, opens another infographic with 5 sustainable habits users should start in their life.

4.2 Item Scanner and Website

When the item scanner button is clicked, it automatically opens up the item scanner in your browser. If it is the first few times using the website, the website will ask to use your camera. Once that is accepted, then the view from the camera is displayed on screen and it will prompt that the item should be placed in the frame. If it recognizes the item as something in its database, it will inform the user of what the item is detected to be and how to properly dispose of it. If the image does not show up in the database, then It will say image not recognized.

5 Troubleshooting & Support

Data recovery is made easy with Unity and specifically Unity Teams. In the event of issues, previous project versions can be restored. This can be done by clicking on the Collab Button (1), located on the top bar, right of the Play/Pause group. Then play clicking on Version History (2), the Collab History (3) will appear where previous versions can be reverted back to.

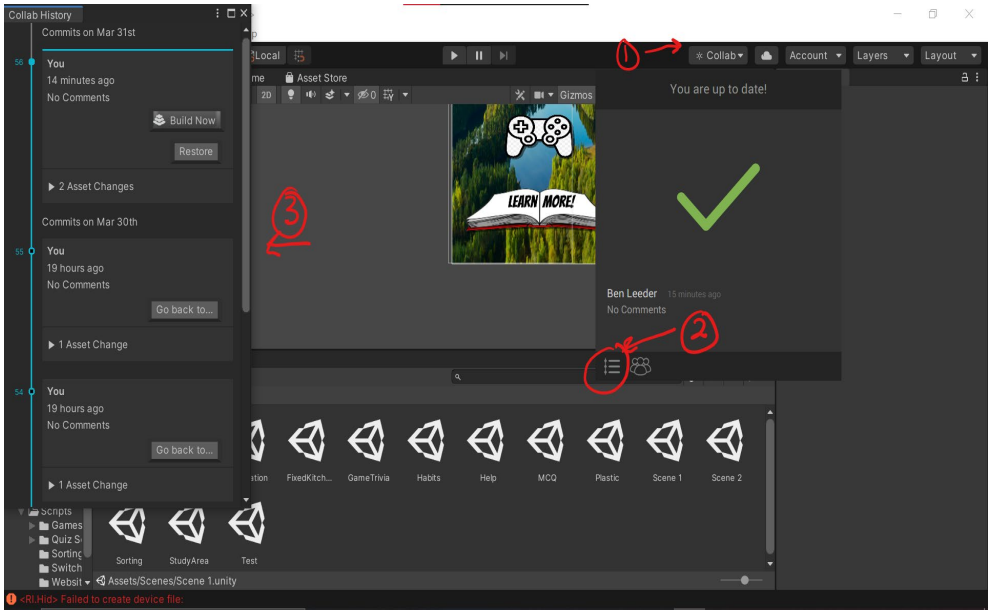

Figure 11-Version Restore

5.1 Error Messages or Behaviors

Most common error messages in the Unity Editor will be based around either scenes not being added to the build list or errors in code. Screenshots for both errors are referenced below.

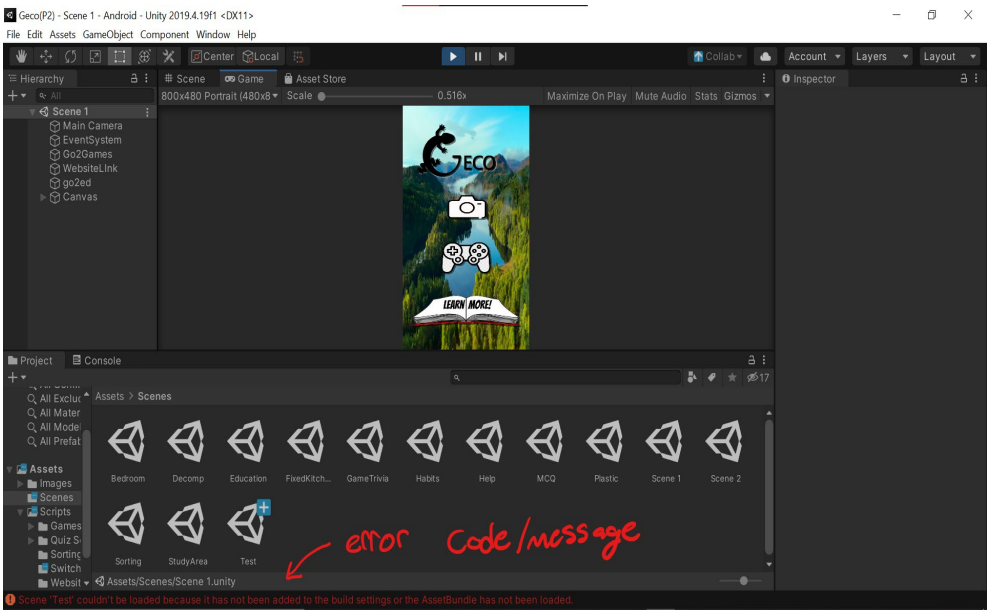

Figure 12-Scene Error Message

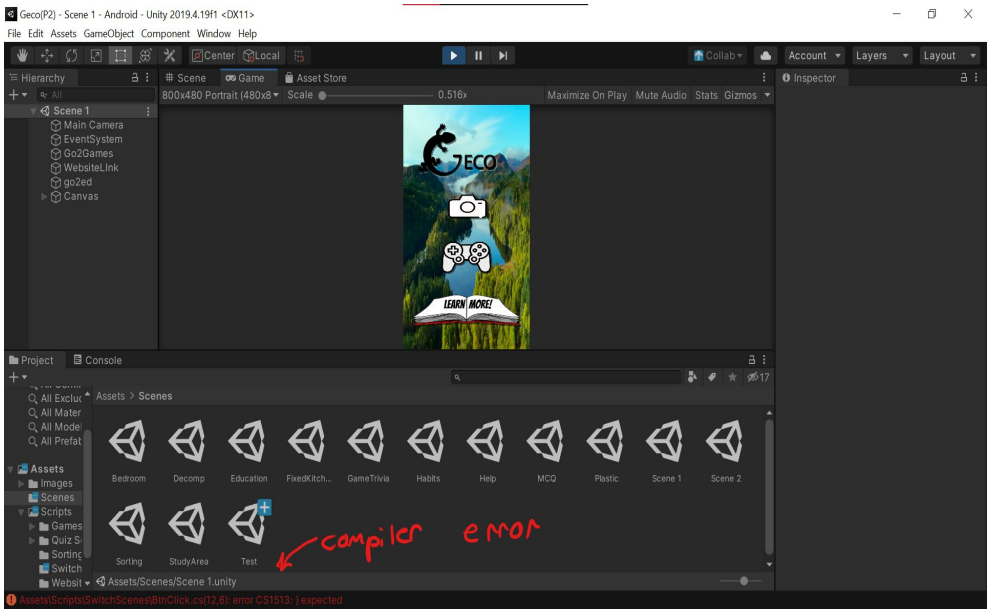

Figure 13-Error Message Code

Regarding the item scanner, error messages about the code in VS are the most likely as well. If there is an error in the code, VS will notify you what the problem is and what line it is referring to. This allows for a good starting point on how to solve the problem. A visual of what would show

up given an error in VS is depicted in Figure 14 below. In this case, the problem in question is on line 25 and caused by having an extra ‘ } ’ symbol. Another indication that there is an error in the code is if the *Go Live* button does not work.

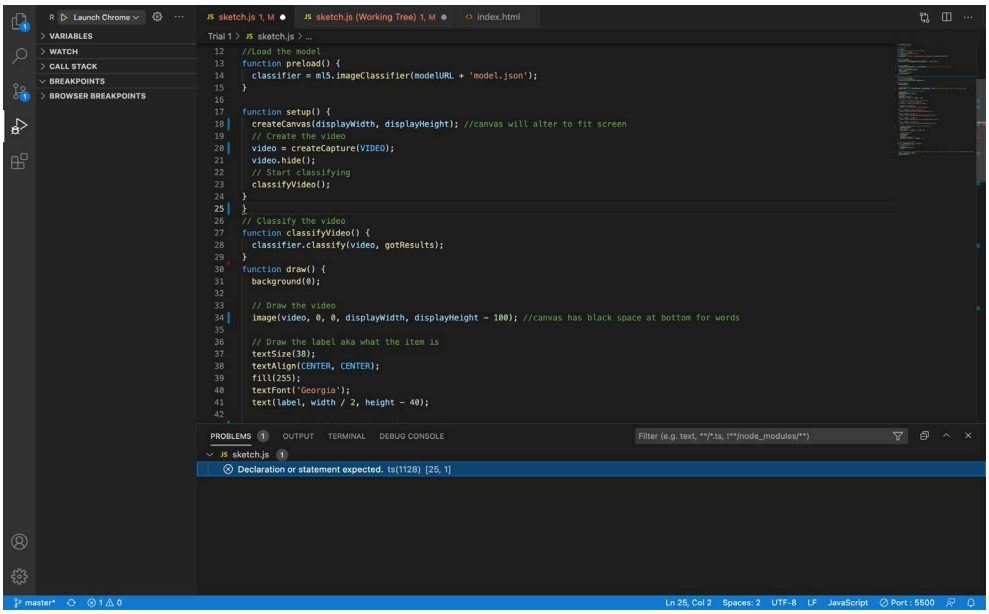

Figure 14- Error Message in VS

5.2 Special Considerations

When Troubling shooting issues with implementation in Unity, pay very close attention the error codes. These messages will make solving issues much easier. Error messages will appear in error at the bottom of the screen in the Unity Editor.

Regarding the item scanner, the activation of the website using GitHub did not work at first. After looking at trouble shooting recommendations, it was determined that a theme for the website

needed to be selected in order for the website to launch correctly. Figure 15 shows the *theme chooser* found in the GitHub repository's settings under the GitHub Pages section.

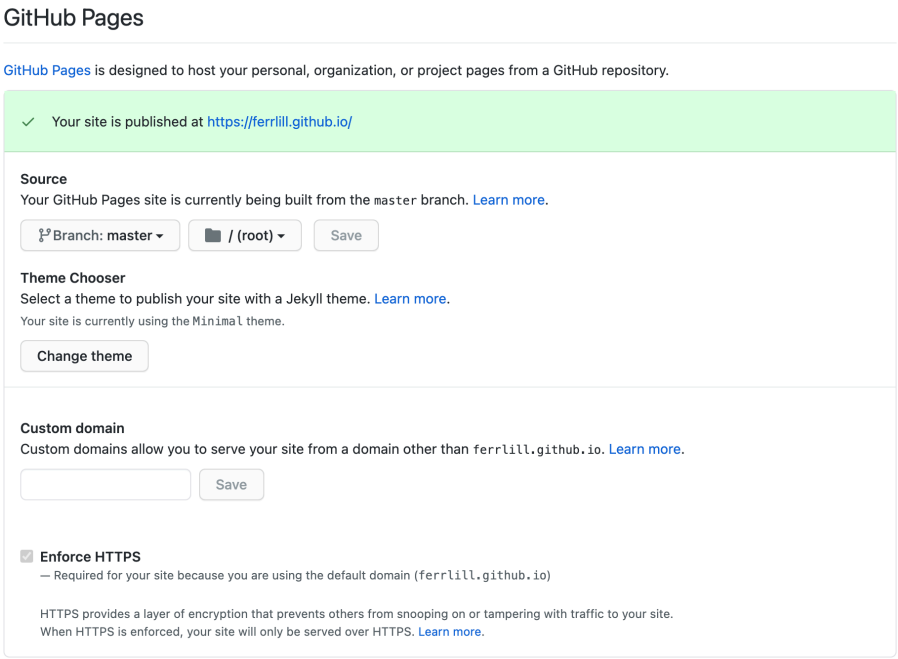

Figure 15- GitHub Theme Chooser

5.3 Maintenance

The application should undergo regular testing as well as optimization for a wide variety of electronic devices. This is due to the large number of devices being released. This maintenance will ensure compatibility in the large number of mobile devices in use now and in the future. As more recycling images become accessible, more images could be added to the TM image database to increase the accuracy of the item scanner.

5.4 Support

For any issues with unity when making changes to the application, one can consult Unity's support at https://unity.com/support-services. For the item scanner, contact support p5.js, Visual Studio support, or GitHub support depending on the situation. Stacked overflow is also a good resource for solving coding and GitHub problems.

6 Product Documentation

6.1 General Application Features

6.1.1 BOM (Bill of Materials)

All materials and equipment used to develop the system were completely free.

6.1.2 Equipment list

Table 2. Equipment list for General Features

Type	Name
Application Developer	Unity Hub
Account	Student Account
Coding App	Visual Studios
PNG Images	Application icons and Background

6.1.3 Instructions

To create the application, start by downloading and installing the unity desktop application into your computer. Unity Hub will make it easier to manage multiple projects simultaneously, even with numerous different versions, and allows you to share your project with whom you wish to work. To download, go to https://store.unity.com/download-nuo , click on first-time users (if applicable), review the terms and conditions, and click on download. During the installation process, you will be asked to create a UnityID. This ID will grant access to the unity editor option, unity store, and upload your project to the cloud.

Once the application is installed, download and install the latest Unity version (Unity 2019.4.21f1 (LTS)) using the ADD tab from the Installs section. You will also need to install

Microsoft Visual Studio software, to develop C sharp scripts in unity and a build supporter for your computer (Mac Build Support, Window Build support, etc.) to create your new project.

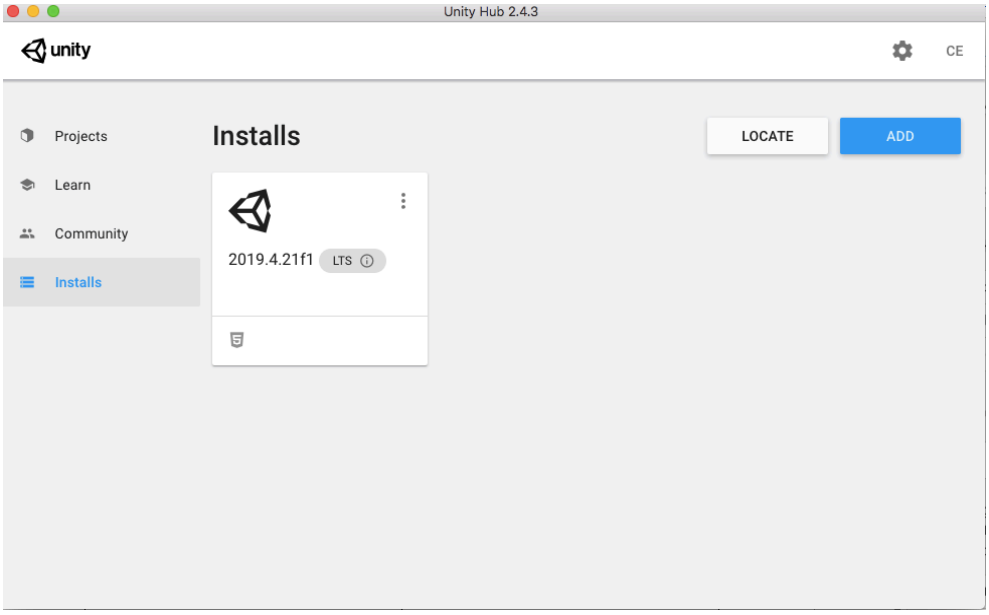

Figure 16- Installs section to add the latest version of unity

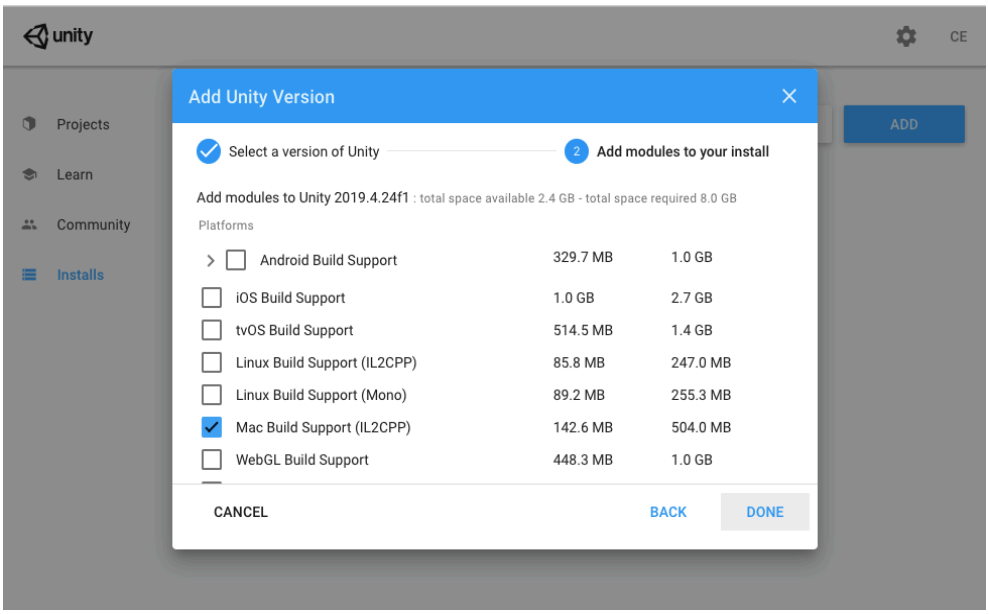

Figure 17- Downloading a computer Build Support

The general app features consist of three main scenes: Home Page, Games Page, and Educational Page. This section will further explain how each main page was created and linked to its corresponding secondary pages. Firstly, create a new scenes b right-clicking in the assets folder and select *scene* in the *create* option. Immediately unity will open an empty work scene that you will name for its corresponding use. We will start by creating the home page; to add any component to the scene, only use the Hierarchy section, and right-click to select such desired addition. In this case, do right-click, select canvas in the UI section as shown in Figure 18 to have our initial setup. On the Inspector page, under “canvas scaler”, select “Scale with Screen Size”.

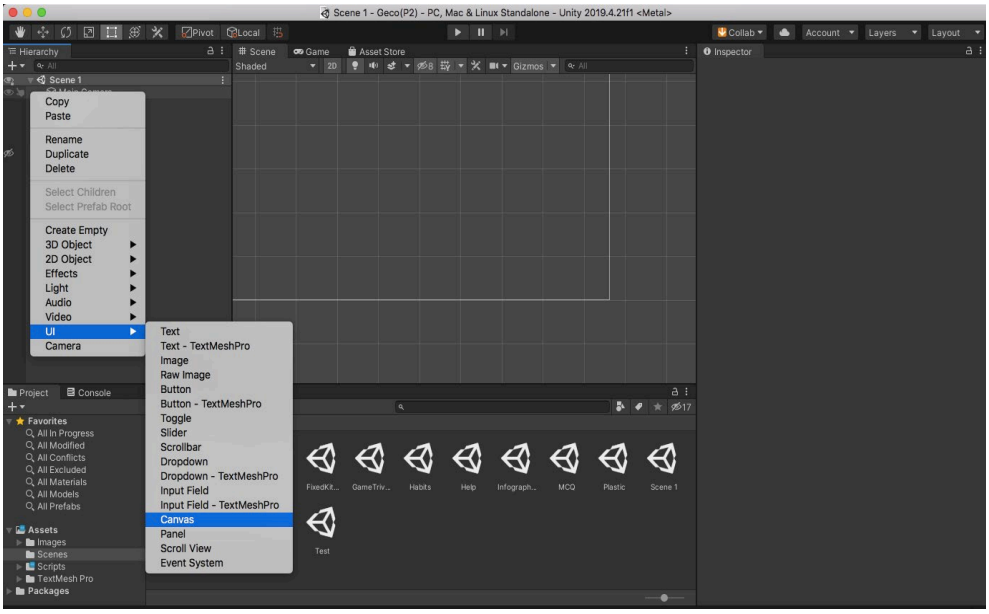

Figure 18- Creating a canvas

Now that the workspace is ready add an image to the canvas to create the Home page background. Fit the image to the canvas using the inspector tab. At this point, you will want to import (drag) an images folder from your desktop to the unity project. Ideally, you would have a prepared folder with all the background images and png images for any icons your app will need. However, if this is not the case, you can import the photos individually to the project. Once the

images are imported, under *Image* in the Inspector tab, select the source image for your desired background. The next step will be to add the desired buttons to the home page. In this case, we will add three buttons to link the scanner, games page, and educational page. To add the button, you will select the button option under UI in the hierarchy window and, duplicate the button twice for a total of three. Using the Unity editor options in the top left of the screen, move and place the buttons around the canvas to nicely fit them all in your preferred arrangement.

For this particular home page design, text is not added to most of the buttons except for the "Learn more" button. Instead, you will replace each button with a descriptive icon for each specific purpose. Select the first button and in the source image section, pick the corresponding icon (which was earlier imported in the images folder) for each of the three buttons in the setup. For example, Figure 1 (in the overview section) shows our simplistic home page design with a camera icon for the scanner, a game controller for the mini-game section and, a book for the learn more page of our application. To add the text component to the learn more button, click the button and select *Text* from the UI section through the hierarchy window. Under *Text Input* in the Inspector tab, write "Learn More!" and adjust the text's size and alignment to fit the button best, as shown in Figure 19. Also, you can change and choose the font through the Font asset option and the font style (e.g., bold, italics, underline, etc.).

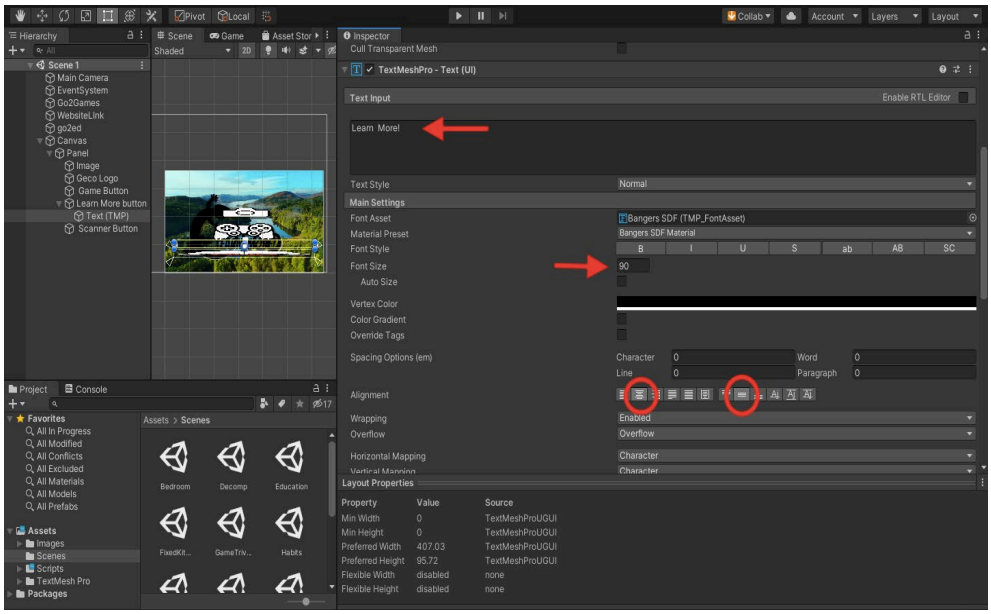

Figure 19- Button's text adjustments

Similarly, you will create and design both games and educational pages. For these two scenes, you will need to add a “back” button option, which will allow the user to return to the home page. Preferably, position the button in a well visible place, such as in the top left or right corner of the canvas. The games page will have a total of three buttons, one for each mini game: Trash Trivia and Recycling Madness (further explanation on their creation on following sections). The Educational page is aimed to have three linking buttons: Local regulations, Ways you can help and, sustainable habits, plus the back button to the home page. The figures down below reference how each scene should look like.

Figure 20- Eduactional Page Design

Figure 21- Games Page Design

The additional two buttons in the Lean more page, Ways you could Help and Sustainable Habits, takes you to some infographics. Infographics were self-made using photoshop app and canvas. Once again, you will create a new scene, and open a new canvas for each button. In the Inspector tab, set the canvas to fit the screen size. Add an image and select your source image to be your desire infographic. With unity tools, scale the image to the screen size as well. Finally add a back button to each scene, to return to the Educational page of the application.

Additionally, you will need to do some coding to connect each scene to its corresponding button. As of now, you will be able to link just the three home screen buttons to its scanner website, the games page, and the educational page, respectively. Start by connecting the camera icon to the scanner website (further details on the scanner and website creation on section 6.2). To do so, create a c-sharp script by right-clicking the assets tab and selecting *C# script* in the *Create* section, as shown in Figure 22. Immediately, name your script "WebsiteTEst" or any descriptive name of your preference, and double click to open in VS. You will notice that VS gives you an already started code with all the primary functions used for unity. Erase everything below "`public class`

`WebsiteTEst: Monobehaviour`" and replace it with a pair of open brackets {}. In between, call `public void Open URL ()`, and open another set of brackets. Lastly, write `Application.OpenURL ("insert your website link here");` in between the brackets. Follow Figure 23 for more doubts.

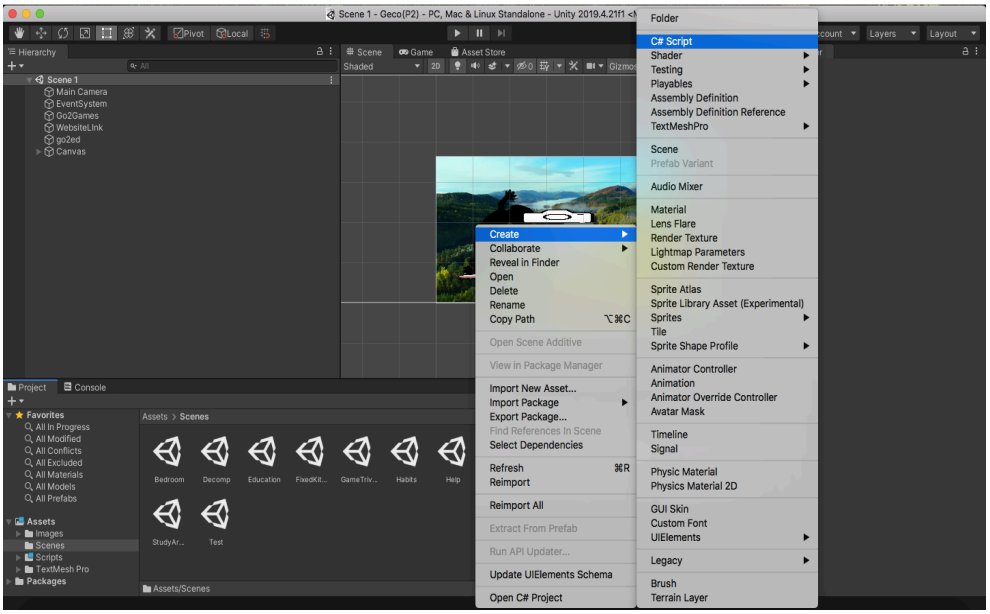

Figure 22- Creating a Script

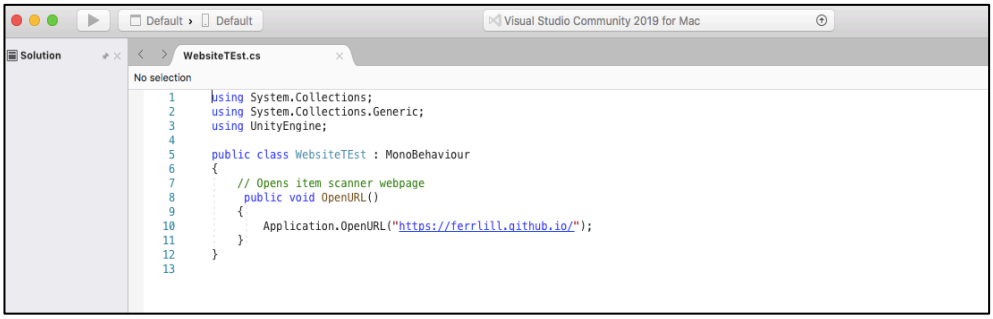

Figure 23- Website Linking Code

Save your code and return to your unity project. Through the Hierarchy window, create an Empty Object and name it "WebsiteLink". Drag your script to the Websitelink object in the hierarchy window to incorporate the link into your scene. Finally, you will select the camera icon

button and scroll down to the *Button* section in the Inspector Tab. You will find an option called *"On Click()"*; click the + icon that will open a small running code menu. Drag the "WebsiteLink" object to the second space to the left of the menu. You will notice that the third button (upper right corner in the menu) will now change, click on it and select the `OpenURL()` function under the *WebsiteTEst* section, as shown in Figure 24. Save your work, and the application should now be fully functioning to open the scanner website. Similarly, you will link the Ottawa regulations website to the Local Regulations button on the Educational page.

To link the scenes within unity (game and educational) to the buttons, it is fairly similar to the previously detailed code. Unlike the website code, you will now have to add a unity function to call said scene and allow the application to swap screens. You will create a new script for each button and follow the VS function setup detailed before. Under the last function given in the first four lines of the code, add "`using UnityEngine.SceneManagement;`". Open the public void with a new name, for example, `Go2Education()`, to link the education page to its button. Replace the function written after the public void with `SceneManager.LoadScene("Education")`. Continue to save and complete the process of calling the code in the *OnClick* function in unity. Follow the same instructions for the games page button and any other linking scenes during the project completion.

Note: we named our Educational page "Education", and don't forget to add all your scenes to the Build settings.

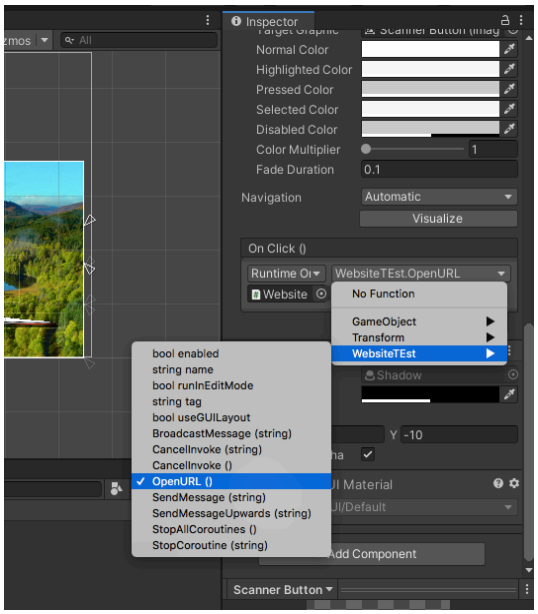

Figure 24- Selecting OnClick() Options

6.2 Item Scanner

6.2.1 BOM

Everything used to create the item scanner was completely free.

6.2.2 Equipment List

Type	Name
Dataset	Categorized Recycling Images
Website Tool	Teachable Machine (by Google)
Coding App	Visual Studios
Account	GitHub
Version Control System	Git

Table 3- Equipment List for the Item Scanner

6.2.3 Instructions

The recycling item scanner as seen in the final product was created through a multiple step process that will be detailed in this section. Firstly, a categorized recycling image database was

downloaded from WaDaBa (http://wadaba.pcz.pl/). This was then uploaded onto Teachable Machine using following categories: Paper/Cardboard, Garbage, Plastic, Glass, Metal, and Image Not Recognized. For the image not recognized section, background pictures and images of people's faces were uploaded so that the scanner would not misidentify those images. After this, the machine was trained using the button that can be seen in Figure 25 below. Once the machine was trained, it

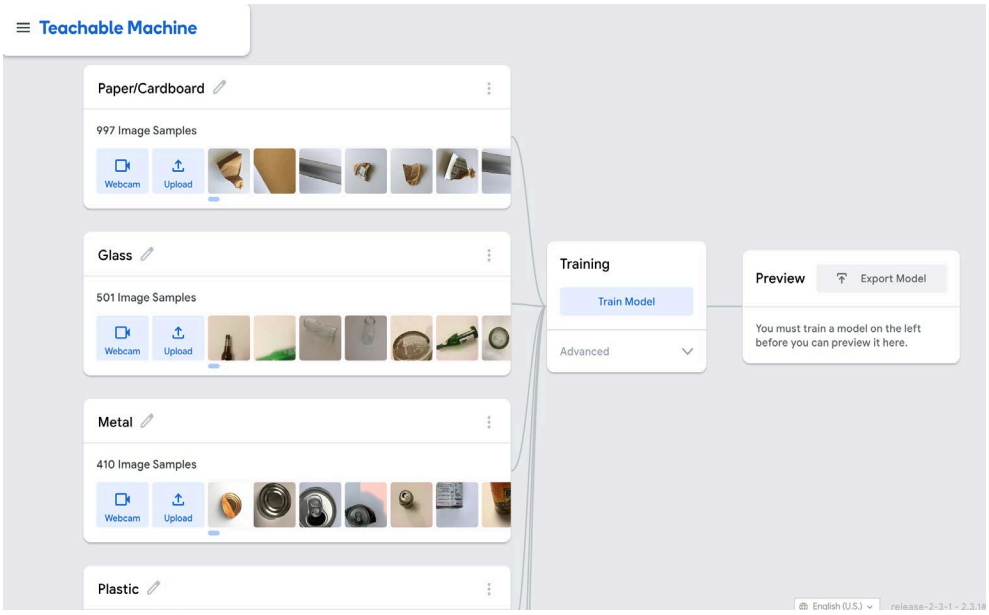

Figure 25- Teachable Machine

was exported and given a permanent URL. The template and YouTube video made by *The Coding Train* was used as a starting point to create the code for the website (https://www.youtube.com/watch?v=kwcillcWOg0). The following code seen in Figure 26 was created in the VS desktop app and first tested using the *Go Live* button seen in the bottom right

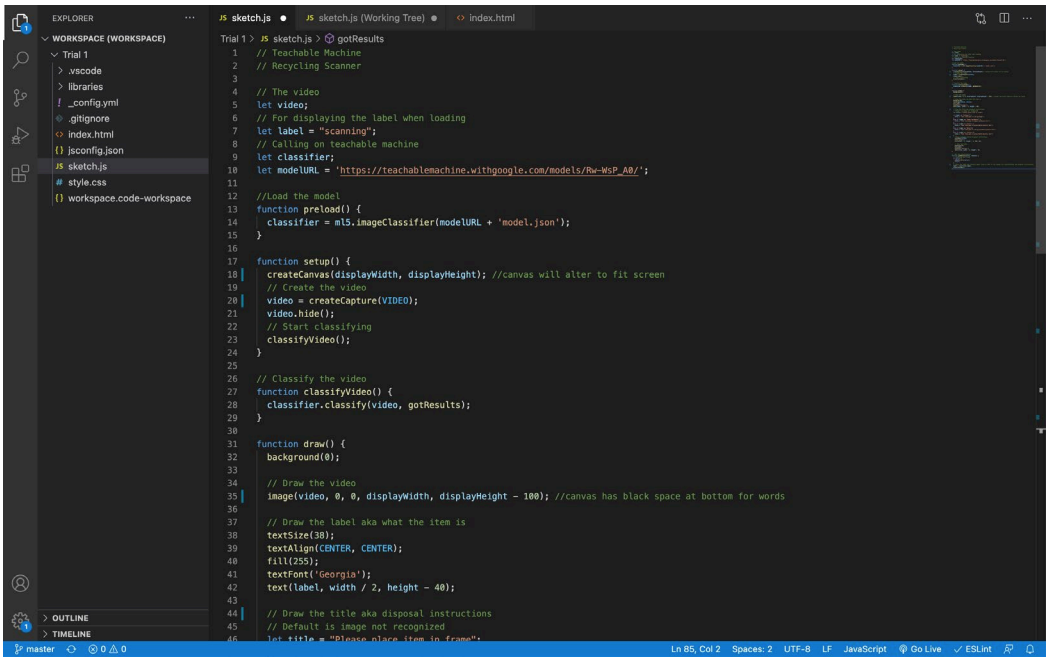

Figure 26- VS Code Page

corner. It is important to note this is only a partial picture of the code created to bring the website to live and that a JavaScript p5 extension was added to the code to allow it to run on VS. Even though the *Go Live* button would open up the item scanner in the browser, it was still running locally on the computer. So in order to allow anyone to access it, a permanent website needed to be created.

The most cost effective way to create a website is by using GitHub. In order to begin this process, GitHub repository needed to be created. Firstly, an account was made on the GitHub website and Git was installed. After this, a repository was created using the instructions outlined on GitHub docs (https://docs.github.com/en/github/getting-started-with-github/create-a-repo). Once this was done, the code could be pushed to the repository on GitHub and a permanent website was created using GitHub pages, the steps that were followed are outlined on GitHub guides (https://guides.github.com/features/pages/). This completed the steps followed to make item scanner website. If changes needed to be made to the website, the code had to first be altered in the

VS desktop. Once this was done, the changes needed to be committed to the repository in order for the website to undergo the desired alterations.

6.3 Games

6.3.1 BOM (Bill of Materials)

Everything used to create the item scanner was completely free.

6.3.2 Equipment list

Table 4. Equipment List for Games

Type	Name
Application Developer	Unity Hub
Coding App	Visual Studios
Audio Clips	Sound Effects
PNG Images	Application icons and Background

6.3.3 Instructions

The mini-games section, Trash Trivia and Recycling Madness, was a lengthy process integral to our application. Follow the detailed steps described previously (in the general app feature section), to add a new scene to the project and, create a canvas for each game. This section will start describing the development of the trivia game followed by the sorting game.

6.3.3.1 Trash Trivia

This trivia game was created based on a tutorial video found on YouTube (https://youtu.be/g_Ff1SPhidg). You will firstly be making and setting up the scene for the game. Open a new canvas and add a panel. In this panel, you will be incorporating a question box and buttons for true and false. For the question box, you will use the panel previously inserted. Scale the square on both the x and y-axis to create a long and thin rectangular at the top of the canvas. Play around with colours, found the inspector tab, under *image*, to have the panel slightly lighter than the canvas. The question panel will need a text component that will be a placeholder for the moment, displaying new questions through the code. Make sure to scale the text, nicely fit the panel, and choose font and color. Add the buttons similar to the previous one you've added to all the scenes created before. Align the buttons with the question panel's edge to have a symmetrical quiz setup as shown in figure 27. To clarify the quiz options, you will insert a text component, under the UI section, to the button and write True for one button and False for the other under the *Text* section of the inspector tab. You can change the font, size, and colour of the text to your preference (as well as the colour of each button). You can name your panel Question Panel and the two buttons True and False, respectively, through the hierarchy window for better organization.

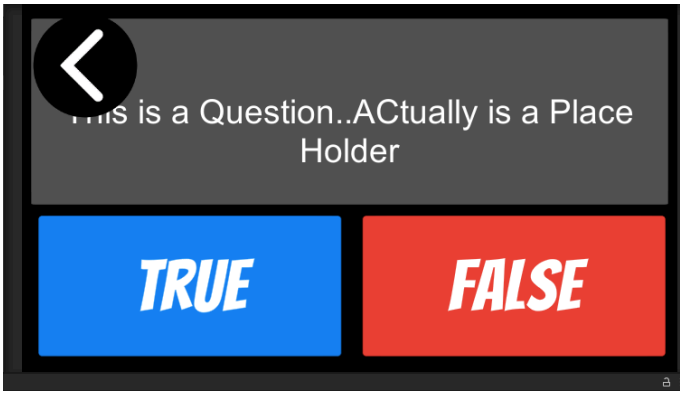

Figure 27. Trivia quiz setup

Once you have the quiz setup ready in Unity, you can start writing up the code for your game. Create a c-script and name it “Manager”. In the hierarchy window, create an *EmptyObject*, name it “QuizManager”, and drag your script to it. This Empty object will be hosting the script. Essentially all the game logic: loading a question, choosing a random question, check if it's true or false, and load the next question. You will also need to create another c- script named “Questions” that will store all the information for your quiz. In your `Question` script, you will have an empty class to start with, where you will set it as a serialized system, which will inform Unity that it is storing information. Create a `public string question` that will have the information for your question bundle and a `public bool isTrue` which will hold whether your questions are true or false.

For your `QuizManager`, you will create a list of questions using an array "`public void Question[] questions;`" and a `private static List<Question>;` that will store the number of questions. You will also create an `if` loop in a `public void start` called every time you start the quiz or reload the scene. You will aim to have a randomized quiz, so you will need to write void `SetCurrentQuestion()` to detail the question choosing process. To access the question, you will write `factText.text = currentQuestion.fact;` inside the `SetCurrentQuestion` function. You can follow the script in the figures down below or use this link https://www.youtube.com/watch?v=5CW1yGsVg4k to follow the tutorial used to create the code. Lastly, you can add some animations when the answer is chosen correctly by introducing the option in your code as a `private Animator`. You'll need to add another text panel, in your unity project, to appear and inform if the answer is correct. Look for the *Window* option in the top bar of your computer and select *animation* to create a clip that will slide over the text when necessary.

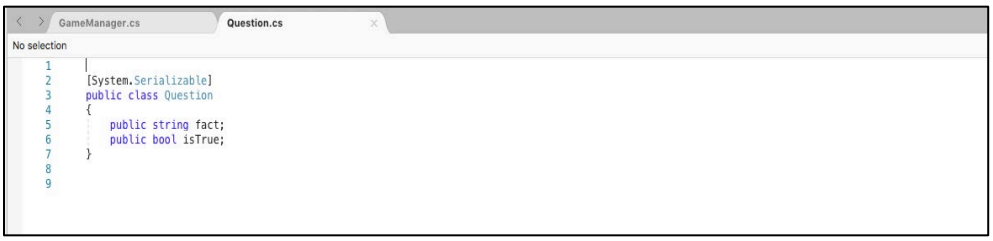

Figure 28. Question Script

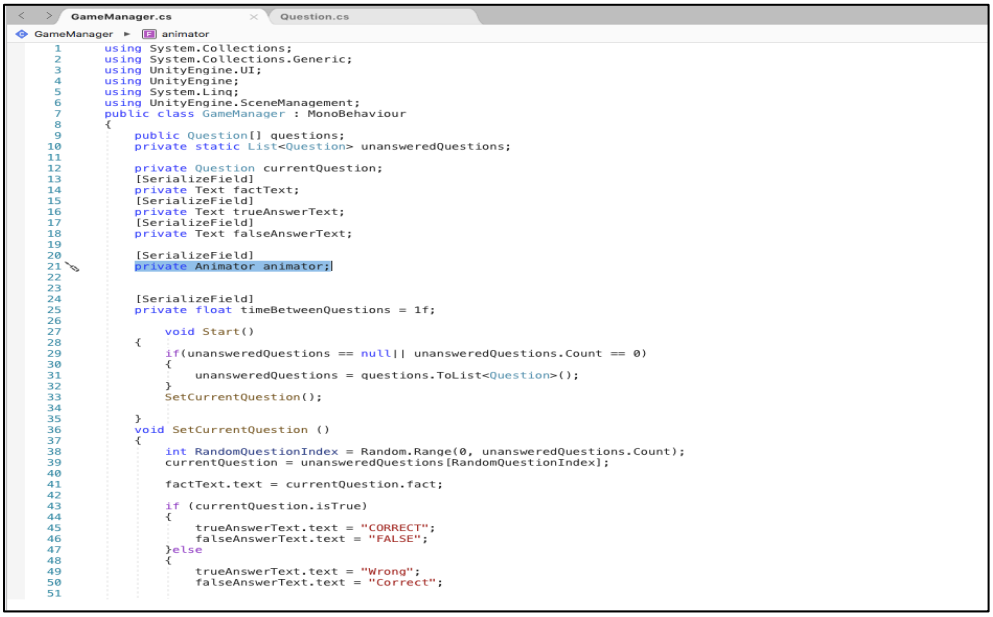

Figure 29. QuizManager script

Once you save your script, the “QuizManager” holder will drop down a menu containing a Questions drop-down list and some text options/ animation space to be filled. You will need to drag each corresponding text box from your hierarchy window to the inspector tab, which will allow the system to identify the functions of each text or animation. Under the Question drop-down menu, you will be able to determine how many questions your quiz will have and if it is true, as shown in the figure down below.

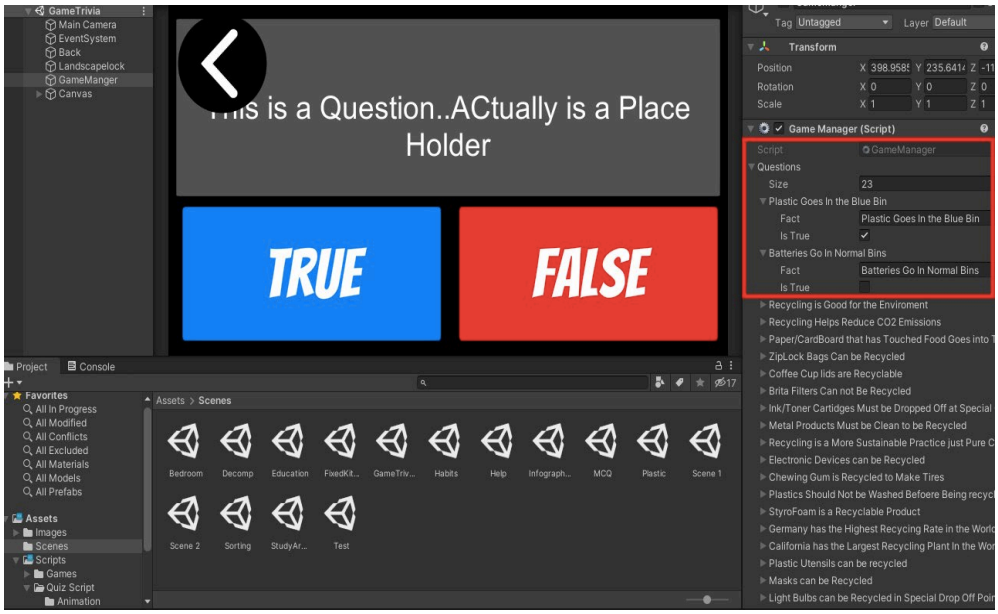

Figure 30. Questions drop-down list

6.3.3.2 Recycling Madness

Firstly, create a new scene, and select a background image for the household location menu for the users to choose from. In this case, the game will offer three different locations, for which you'll need three corresponding buttons. Add the buttons (through the Hierarchy window) to your canvas and adjust their dimensions and position to fit the screen best. As a recommendation, since the buttons will provide a preview of the location option, it is preferable to make the buttons squared shaped for better visibility of the image. In the Inspector tab, the source image selects the corresponding image for each button, which you imported to the project earlier within the images folder. In view of a very descriptive image of the location, it is unnecessary to write down the place's name within the button. Instead, you will add a text option to each button to insert the word "Play!", to create a more interactive page, as shown in figure 31. Remember that you can play around with the fonts, size, and color, and make sure to align and center the text. The last thing you will need for the scene to be complete is the back button to return to the home page.

Figure 31.Sorting game option scene

To create the first location option, in this case, the kitchen, you will open a new scene in your project, add a canvas, and set the " fit the screen size" option. The first step is to set the background image of a kitchen by inserting an image component to the canvas and selecting the source image option. Then, add another UI image to the canvas and duplicate it as many times as to match the number of items you will want the user to drag. For example, this scene will have a total of eight items to drag, plus three garbage bins, for a total of 11 images. You will now insert the sprite to each image component by either dragging it to the blank space or selecting the source image through the inspector tab. For the item's picture, try to incorporate things that would typically find in that section, e.g., the kitchen, such as a cereal box, yogurt tub, eggshell, etc. Resize the image's height and width to achieve a better appearance for each sprite. You can also move the items around the canvas to assign a specific place for the image to avoid a messy presentation. Lastly, add a back button in the top left corner of the canvas by adding the sprite into the button. You can use figure 32 and a reference for your sorting game scene.

Figure 32. Sample kitchen scene

Now that the initial scene aesthetics are done, it is time to code for a drag and drop action. This code was completed by following an instructional video on YouTube (https://www.youtube.com/watch?v=Zg2t9IOJRQ8&t=145s). Create a new script as shown in figure 22, and name it *KitchenManager*. As the script opens with VS, erase everything below "`public class KitchenManager: Monobehaviour`" and replace it with a pair of open brackets {}. Note that our items to be dragged in figure 32 are a ketchup bottle, yogurt tub, cereal box, etc. Start by introducing your game objects by writing "`public GameObject ketchup, yogurt, cereal,….,recyclingbin;`" in between the brackets. Then, open a `vector2` function just down below to state the name for your object's initial position, which you will use throughout the code.

For this drag and drop game, you'll have three main functions for your game to work correctly: start (), drag (), and drop (). The start function will specify that the game will begin when a specific object moves commanded by "`KetchupInitialPos = ketchup.transform.position;`". This transform component will allow you to store and

manipulate the position, scale and rotate your object. Write a similar statement between your `public void Start ()` function brackets for each object to be dragged in your game. The following function would be a void DragKetchup(), which will noticeably allow the object to be dragged around the canvas. To drag, the object's position has to be set up to be the same position as your mouse. For each object, write `public void drag ()` and add "`ketchup.transform.position = Input.mousePosition;`". Lastly, you have to make a drop function for each object being dropped to its corresponding bin. For this, you will need to open an `if` loop that will measure the distance of the selected object to its corresponding bin. If the distance between the object is minimal, the bin will take it, and the object will disappear. However, if the object is not dragged to the correct bin, the code will return the object to its initial position.

At this time, the drag and drop code is ready. You can add some extra functions to display some sound effects for both wrong and correct bin selection. To do this, add three public statements: `AudioSource source, AudioClip correct, and AudioClip incorrect`, right after your `vector2` function at the begging of the code. Once the audios have been stated in the code, go back to each drop function (one for each object dragged), and insert "`source.clip = correct;`" and "`source.play();`" to the `if` loop for correct answers. For the wrong answer sound effect to work, you will insert "`source.clip = incorrect;`" and "`source.Play();`" in the `else` loop. You can follow the figures below for more information and guidance on the functions and general code. Save your code and go back to the unity hub.

KitchenManager.cs

KitchenManager ▸ Start()

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

public class KitchenManager : MonoBehaviour
{
    public GameObject ketchup, yogurt, cereal, banana, oil, eggshell, plate, teabag, can, mug, trashbin, compostbin, recyclingbin;
    Vector2 ketchupInitialPos, yogurtInitialPos, cerealInitialPos, bananaInitialPos, oilInitialPos, eggshellInitialPos, plateInitialPos, teabagInitialPos, canI

    public AudioSource source;
    public AudioClip correct;
    public AudioClip incorrect;


    public void Start()
    {
        ketchupInitialPos = ketchup.transform.position;
        yogurtInitialPos = yogurt.transform.position;
        cerealInitialPos = cereal.transform.position;
        bananaInitialPos = banana.transform.position;
        oilInitialPos = oil.transform.position;
        eggshellInitialPos = eggshell.transform.position;
        plateInitialPos = plate.transform.position;
        teabagInitialPos = teabag.transform.position;
        canInitialPos = can.transform.position;
        mugInitialPos = mug.transform.position;
    }

    public void DragKetchup()
    {
        ketchup.transform.position = Input.mousePosition;
    }

    public void DragYogurt()
    {
        yogurt.transform.position = Input.mousePosition;
    }
```

Figure 33.Drag and Drop code example 1

```
public void DropKetchup()
{
        float Distance = Vector3.Distance(ketchup.transform.position, recyclingbin.transform.position);
        if (Distance < 50)
        {
            ketchup.transform.position = recyclingbin.transform.position;
            source.clip = correct;
            source.Play();
        }
        else
        {
            ketchup.transform.position = ketchupInitialPos;
            source.clip = incorrect;
            source.Play();

        }
}


public void DropYogurt()
{
    float Distance = Vector3.Distance(yogurt.transform.position, recyclingbin.transform.position);
    if (Distance < 50)
    {
        yogurt.transform.position = recyclingbin.transform.position;
        source.clip = correct;
        source.Play();
    }
    else
    {
        yogurt.transform.position = yogurtInitialPos;
        source.clip = incorrect;
        source.Play();

    }
}
```

Figure 34.Drag and Drop code example 2

After elaborating the code, you will have to introduce it into your unity scene. Create an *EmptyObject* , name it "*KitchenManager,*" and drag your script folder into the object. This step was further explained previously in section 6.1 of the manual. You will note that once the script has been added correctly to the scene, the inspector tab will drop a menu displaying all your

object's names and functions. This is for you to indicate which object is which in your scene. To do so, drag the object image name, under the hierarchy window, to its corresponding function in the inspector tab. Follow figure 35 for more guidance on this step. You will also need to select the audios source for the sound effects. You can download the audio clips from YouTube https://www.youtube.com/watch?v=qHx6qJBiMvA (for correct answer) and https://www.youtube.com/watch?v=2naim9F4010 (for incorrect), and later convert them to mp4 audios. Import the audios to unity and select "Audio source" under the *Add component* section of the inspector tab. This action will add more objects to the script menu: source, correct and incorrect. Drag each corresponding audio to the menu bar, and you are ready to test your game.

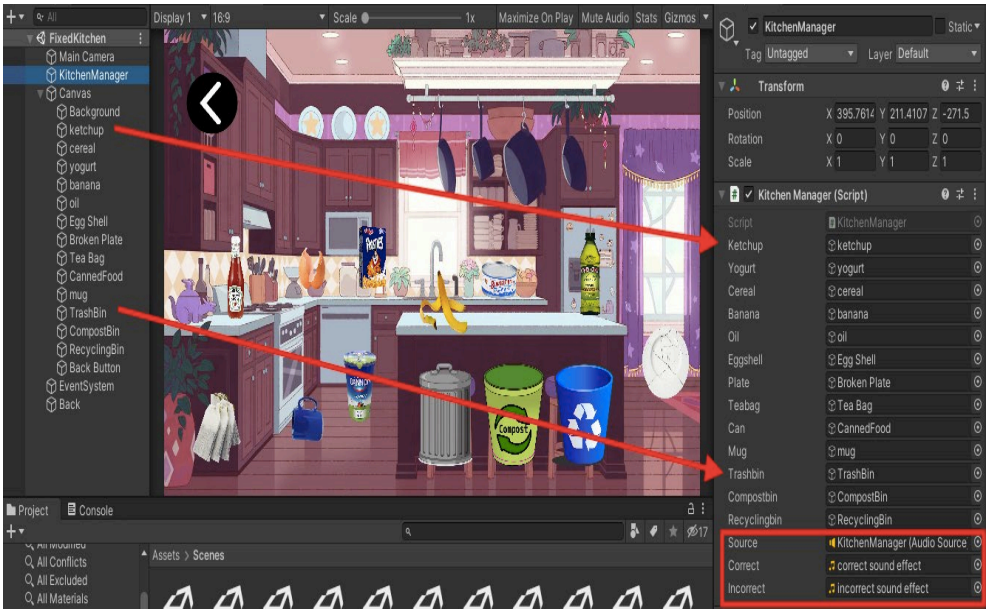

Figure 35.Script drop-down menu

Similarly, you will be creating the other two locations: a bedroom and a study area. The sound effects are the same for each scene, so there is no need to import more audio clips. Use the figures down below to recreate the scenes for each sorting game option. Don't forget to include a script to return to the games page and link each location scene to its corresponding button.

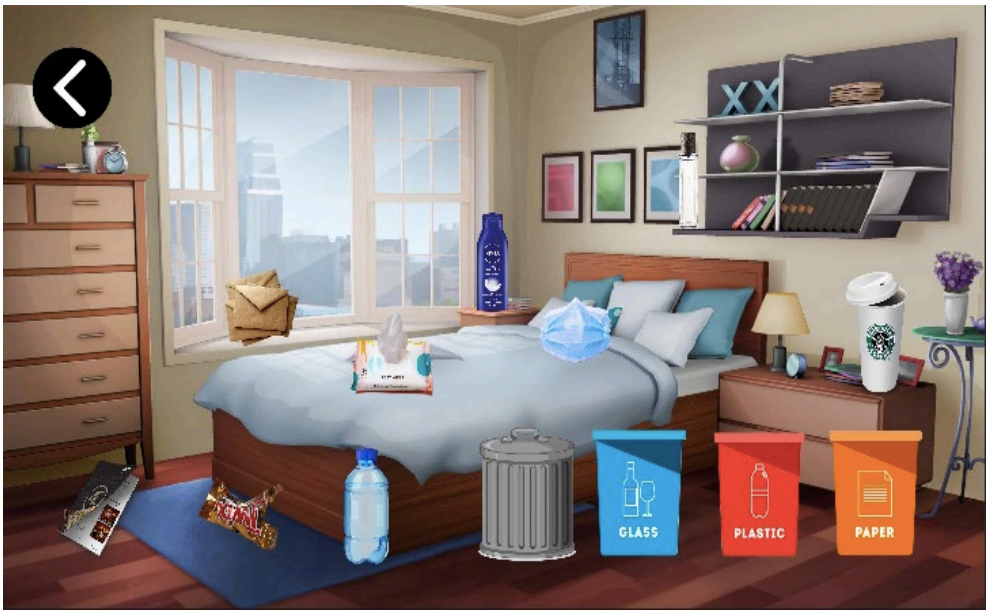

Figure 36.Bedroom scene

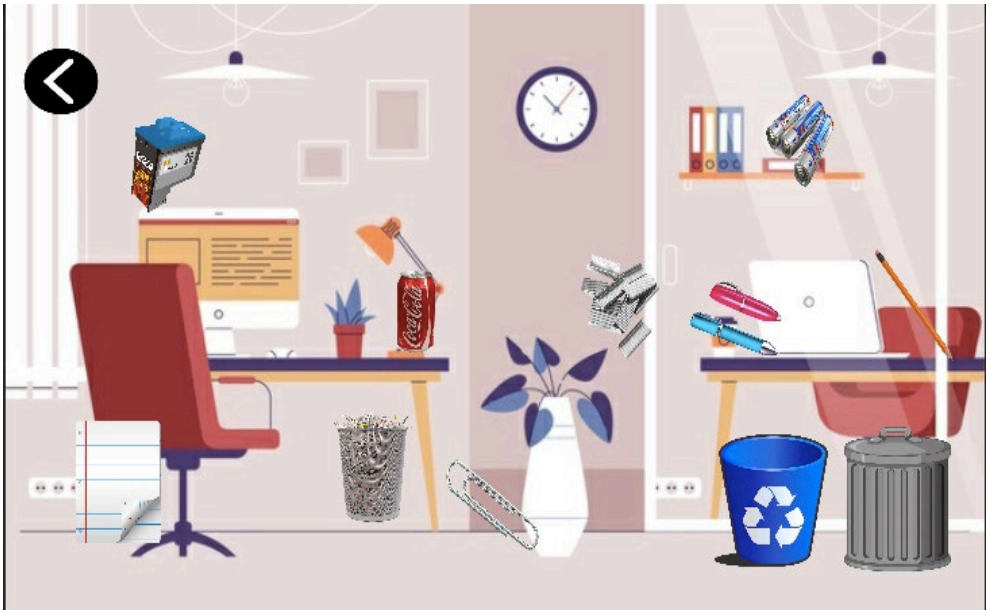

Figure 37.Study Area scene

6.4 Testing & Validation

Testing for the application was done on multiple Android phones and in the Unity Editor. The application was tested by loading it onto a device and allowing someone to use the app. They would make sure buttons were responsive and everything that could be interacted with would do as

intended. The item scanner was also tested on a laptop and multiple mobile devices. The accuracy of the item scanner was assessed using various different recycling items to determine if the scanner was correctly identifying the items.

Feedback was given mostly in needing to make aesthetic changes. Only issues that should be checked frequently is the compatibly on different screen sizes of the trivia game animations. Some screen sizes cause the animation to not properly reveal the correct answer to the given question. Regarding the item scanner, feedback on esthetics and accuracy were given.

7 Conclusions and Recommendations for Future Work

To conclude our waste management project, we believe that our application Geco proves to be a user-friendly one and a healthy promoter of proper recycling habits. It also successfully raises awareness about the consequences of improper recycling.

Many lessons were learned throughout the development process of this project. Being introduced to a highly reliable software like Unity and learning how to utilize it even in the future was a great advantage of this project. Learning how to have effective and respectful communication amongst the group is a life-long learning lesson. Knowing how important it is to incorporate project planning was definitely a game-changer. In addition, creating prototypes and testing them. We found that to be extremely essential in allowing us to discover flaws in our design which we didn't imagine could be there. The final lesson we learned is the importance of having a simple and clean user interface and not propagandizing your product. As our dear professor had once advised us, "Nothing matters as long as you can SELL your project!"

8 Bibliography

Unity laboratory Manual. Engineering Design Lab. University of Ottawa

APPENDICES

9 APPENDIX I: Design Files

Maker Repo Link: https://makerepo.com/Bleed087/810.geco-by-synthetic-solutions

Google Drive Link:
https://drive.google.com/drive/folders/1MUNgEZvc4CPK9VgG3swRlNEo1ci-zkpW

Table 5. Referenced Documents

Document Name	Document Location and/or URL
Visual Studio Download	https://visualstudio.microsoft.com/
Git Download	https://git-scm.com/downloads
GitHub	https://github.com/
Unity Download	https://store.unity.com/download-nuo
GitHub Pull Requests	https://docs.github.com/en/github/managing-files-in-a-repository/editing-files-in-another-users-repository
Teachable Machine	https://teachablemachine.withgoogle.com/
Image Database	http://wadaba.pcz.pl/
Creating a Repo (GitHub)	https://docs.github.com/en/github/getting-started-with-github/create-a-repo
GitHub Pages	https://guides.github.com/features/pages/
Trivia Game Tutorial Video	https://www.youtube.com/watch?v=5CW1yGsVg4k
Drag and Drop Game Tutorial Video	https://www.youtube.com/watch?v=Zg2t9IOJRQ8&t=145s
Coding Train TM Tutorial	https://www.youtube.com/watch?v=kwcillcWOg0
Unity Lab manual	https://uottawa.brightspace.com/d2l/le/dropbox/215316/130159/DownloadAttachment?fid=3178626