Université d'Ottawa University of Ottawa

uOttawa

L'Université canadienne
Canada's university

GNG1103 – Project A.R.C.

User Manual

By: Caine Myrah 5722319
Melanie Pinto 300117030
Mike Shepperd 300166172
Aunonto Bhuiya 300115942

Table of Contents

1. Getting Started

1.1 To upload a model:

1) open the project through the Unity hub.
2) Once the project is loaded, in the left-hand window(Hierarchy), navigate to the appropriate building section.
3) Select all of the components under that heading(ie: under “Building(Structure)”), be sure you know what material they are using (listed in right-hand window(inspector). Once the models under the desired section are selected,
4) Delete the selected models from the scene.
5) If you still have the original models for the ones being replaced or no longer care about the ones being replaced, they too can be deleted from the asset manager(bottom left window).
6) From your file explorer(windows), drag and drop a copy of the new building file in to the project’s asset folder. (search for the file name in the explorer if you don’t know the exact location)
7) Now that the new model is in the asset folder, Unity will not automatically import it. You can now drag and drop an instance of this model in to the hierarchy in the left-hand window. If this does not work, drop it directly in to the scene and then in the hierarchy window make sure to drag and drop it in to its appropriate building model type.
8) Select the relevant model, and in the right-hand window there should be a material property that can be selected. Click on the small bullseye beside

whatever is named there and in the new window select the material you remembered from before, or something equally appropriate.

9) Repeat for the scenes “TryingHarder”, “ARTime”, and “ARTimeB”

1.2 Building the APK

1) From the file menu, select build settings.
2) At the bottom hit build.
3) Call it what you want, and specify a convenient desination folder.
4) Drag the apk file from your chosen folder to a folder on your phone.
5) From your phone, find and select the apk file you named and ignore any warnings and give the app any permissions it seeks.

It is not harmful, at least as I have submitted it. However, others may modify it in the future, I am not responsible for any damage that may be incurred.

2. On the Phone

2.1 Loading Up

The app loads and prompts the user to choose either the AR environment or the 3D environment. The AR provides an overlay of the 3D model over the phone’s camera feed while the 3D displays the model in a fully virtual environment.

2.2 Main POV Camera

In both the AR and 3D environments, once in the model environment(after initial prompt) the user is free to look and move around the 3D model. The interface here allows the user to activate and deactivate different building layers using the toggle buttons at the top of the screen, go to the annotations menu(top right), go to the main menu(top left) or change elevations.

To move around in AR, just move and rotate the phone to change the view and position of the camera around the model.

To move in 3D, the left side of the screen will read a touch input for player movement. (up = forward, down = back, right = strafe right, left = strafe left). To look around, touch inputs on the right side of the screen will be read, simply drag the screen in the direction you want the environment to rotate. To look up, drag the building down. Changing elevations is handled using the up and down arrows in this mode.

2.3 Main Menu

Here the user can navigate to the settings, exit the app, resume, or go to the save/update menu(not implemented for .fbx).

2.3 Settings

Here the user can change the transparency of models in the environment. (this is very picky with material and model type). The user can also switch environments from this menu by hitting either the AR or 3D buttons depending on which environment they are in.

2.4 Save Update

This effectively destroys the existing iteration of aa rendered model and replaces it with a new one. Only works on OBJ files though.

2.5 Annotations

From the POV screen, after hitting the button in the top right, the user is brought to a screen where they can either take a screenshot of what they are currently observing. Then they are prompted to share using gmail, or a plethora of other 3rd party services. The user can then hit done to proceed to the next menu, where they can pinpoint the area of interest on a floorplan that updates based on which floor they are on or their elevation. They can then take another screenshot to share the same way as above, and continue to be brought back to the exploration environment.

Appendix A: Developer's Manual

A1. Application Framework

The application was built around the 3D model viewing environment, scene named "TryingHarder", and natively boots up in this scene. This scene contains all the same basic features as the AR scene "ARTime". From here the user can navigate to the AR scene and back and access all the menus and functions.

A2. The Character Controller

The virtual embodiment of the user within the simulation environment. A blank gameObject was created within the hierarchy pane, and renamed to "1stPersonPlayer".

With this selected, a character controller component was added from the bottom of the inspector window. Here is also where most of the game scripts are added, governing everything from movement to menu controls and interface canvases. The main camera within the hierarchy window was also dragged into this gameObject. The touch input module was required in order to get the first person controller working and all of the buttons working with mouse-click and touch(module forced active). Should be a fairly simple solve to switch this to the standaloneInputModule, but this one works fine for now.

Included C# Scripts:

- elevationController.cs
    - implements an incremental change to the character controllers y-position, using the methods goinUp and goinDown. Both are called from the appropriate arrow buttons on the POV canvas
- ActiveMenu.cs
    - This runs an update method that ensures that only one menu is active at a time. When adding this to the character controller, the developer can specifically select the canvas they want to correspond to the trigger. The methods “xxxxsetActive” will set the specified canvas selected in the inspector window dropdown active. (eg:setMainActive will activate the main menu and deactivate all others)
- FirstPersonController
    - This one takes left and right touch inputs based on where the user is touching the screen. To use it, you need to specify the main camera

transform, the character controller used as well as camera sensitivity(1 recommended), Move speed (7 works well), and input deadzone(12 works fairly well, lower numbers increase the deadzone.

    - It modifies the position of the user by calling on the character controller's move function, applying a vector2 taken from the left input on the screen for direction and applying the movement speed.
    - The look input is taken from the right, only this time the camera transform is directly rotated using the transform.localRotation. Passing a Quaternion the clamped and updated camera pitch.
    - This function is deactivated separately to the menu, but using the same button. Any menu buttons pointing back to the POV canvas must reactivate this control using the activateControl method of First Person Controller.

- IMG2Sprite (Not used)
- SceneChgr
    - This opens a new scene. Any scenes added will require their own methods to call them. Code in the script is pretty obvious.
- ShareButton
    - This feature takes a screenshot, encodes it to .png format and then prompts the user to select a 3rd party app (such as google or discord) to share the screenshot. An elegant solution to the locked down mobile filing system.
- ScreenCap2 (not used, but quite useful)

    - Also takes a screenshot, but saves the file locally. These cant easily be accessed by the user until this app is supported by a firebase project or similar. (for now only 2 screenshots will be saved using this, to save on app data and lack of access)
- mapDisplay
    - This is tied to the annotationsx canvas, and has the main camera transform selected. (it uses this value to decide which floorplan to display)
- moveCursor
    - This script allow the user to move the cursor in the annotations menu, in order to pinpoint areas of interest for the annotation.
    - Like mapDisplay, this script in added to the annotationsx canvas, and targets both an image object(map) as well as a sprite(cursor)

A3. The 3D Models

There are 4 empty gameObjects named as building(Structure) etc., denoting the models that correspond to those building components. These game objects are the ones handled within the POV canvas, setActive is called by the four toggles with Icons on the screen.

- New materials had to be added to the FBX files after they were imported.

A4. Canvases

These are the core element of the application's interface. All of them are linked to the main camera or AR camera of the scene, or should be. These provide selective

overlays that allow the user to interact with them without having to leave that particular scene.

Note that their distances from the camera have been staggered to make editing easier. For final implementations though, it would be best to collapse them down close to the camera. Otherwise they can get blocked by in-game entities.

A5. AR Scenes

A few differences here, no need for touch movement and look controls as that is all handled by the AR camera. So that script was not added, and all the menu buttons that activated them had to be adjusted as this function is not desired here. The elevation controller was also incompatible within the AR scene. A quick solution was to implement another AR scene that was a copy of this base one, only with the building now at a different elevation.

A6. Update from file

We managed to successfully implement a script that essentially allows the user to import and replace any 3d object within the scene with a freshly imported .obj model. However, it was restricted to only .obj files. Able to import from websites capturing the file through a datastream.

Appendix B: Health and Safety Statement

Mission Statement

PolarDevs is committed to protecting the health, and safety of our employees, customers We recognize that by integrating health, and safety management practices into all aspects of our business, we can offer technologically innovative products and services. PolarDevs strives for continuous improvement in our health and safety management systems and in the quality of our products, processes, and services.

Guiding Principles

We strive to meet or exceed all applicable safety requirements for the app store and the play store. The app is made strictly from our company that allows the user to upload their build sites onto the app. All protocols for the app store and play store were met, no content was stolen.

Safety Precautions

- When Handling the product keep in mind of any hazards in the vicinity of the room or work space.
- For safety of the users company do not share annotation screenshots outside of the company.
- If using AR on any device, be aware of its temperature, especially on older devices, as high temperatures can increase device wear.
- Be acutely aware of your surroundings when navigating around in AR, just because it looks like there is a floor there on camera, does not mean it is.

We strive to create products that are safe in their intended use, throughout the product life cycle including design, manufacture, use, and end-of-life management.