Understanding the use of Blender and Unity as key tools in implementing a virtual reality memory room

Lukasz Wysocki[b, g], **Umar Ibrahim**[b, e], **Marc Auf Der Heyde**[a,d], **Kshitiz Bisht**[a, c], **and Andrew Parry**[a, f]

[a]Unity Developer; [b]3D Modeler; [c]Scrum master; [d]Product owner; [e]Team manager; [f]UX designer; [g]Unit tester

This manuscript was compiled on March 30, 2021

The authors were a part of the CS34 team, which worked with Viarama in order to build a VR application that would serve as a bespoke memory room, particularly targeting older people in society who may be reserved to care homes and those suffering from mental inhibitions such as dementia. It hopes to create a sense of immersion into a room whereby they can roam freely and experience activities no longer feasible in their current state, such as walking, listening to childhood music, and view virtual-ized images of their childhood to name a few. This application serves as a basis for what we hope will cause an increase in development in this area in hopes of making more seamless the ability to create therapeudic experiences. This paper specifically will elucidate the use of Blender and Unity for their respective purposes.

Source code downloads

To download the complete source code for the Desktop Virtual Reality Memory Room Client, please click here. To download the source code for the Virtual Reality Memory Room Client, please click here.

Final build downloads

To download the finished build for the Desktop Virtual Reality Memory Room Client, please click here. To download the finished build for the Virtual Reality Memory Room Client, please click here.

The power of Blender

As developers of this project, it is important that we incorporate the use of software that effectively solves the problem at hand, but is also cost-efficient. Blender is one of few applications that is the best of both worlds.

> **"Blender is a free and open-source 3D computer graphics software toolset used for creating animated films, visual effects, art, 3D printed models, motion graphics, interactive 3D applications, virtual reality, and computer games."**

Both authors had virtually no experience with using Blender, so as is expected, there is a long and steep initial learning curve. However, once passed, the tool can offer a very powerful workflow for building models for VR applications.

Use of Blender

The question has to be asked: **What does Blender aim to do with regards to the project?** In essence, the VR application consists of various objects, themed to replicate an older time period. Although various models already exist online, this brings about several issues:

1. Appropriately licensed objects are difficult to come across.
2. The variety of models is limited.
3. The given model may not coincide with our workflow, creating inconsistencies in the repository.

In reference to **point 3**, as modellers there is a specific workflow that has been employed. **First**, create the raw model with the required meshes. **Second**, create and apply the necessary materials for your model. **Third**, copy and paste the model, and for the copied version, select all components and join into a single object via *Right-click + Join* or *Ctrl + J*. **Fourth**, unwrap the model and bake the textures. **Fifth**, copy and paste the single-object model and remove all materials. **Sixth**, add a new material

(**Principled BSDF**) and connect an image texture to the colour with the image created from the baking procedure. Steps 4 - 6 will be broken down below as it is a delicate procedure.

Unwrapping [UV] and baking

UV unwrapping is the procedure of creating a UV map, a flat representation, for a 3D model. U and V refer to the horizontal and vertical axes of the 2D space respectively, as X, Y and Z are already being used in the 3D space. Due to this, there is no notion of a 3D texture, as they're truly a 2D image representation of the 3D model. This process is mandatory as 3D models do not directly translate with their respective materials into a game engine, such as Unity in our case. Once the UV map has been made, the baking procedure, as the name vaguely suggests, applies the colours to the 2D representation, which can then be transferred as a material to a blank version of the model. Below will explain, in detail, the workflow for unwrapping and baking a given model: **Note**: *It would be wise to prepare the workspace to make the following steps as seamless as possible. Split the window in half vertically and then split the left-half in half further horizontally. Set the top-left window to the shader editor, the bottom-left to the UV editor, and the right-most window to the default object mode.*

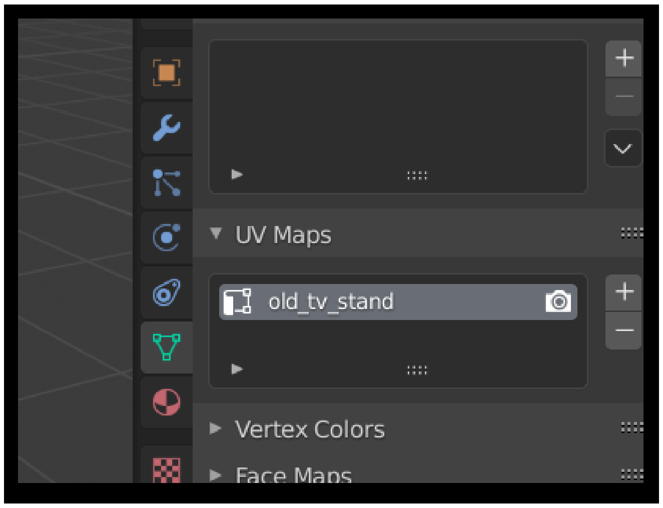


Fig. 1. UV maps window in Blender. The '+' and '-' can add a new map or remove a selected one.

First, given a completed model (i.e: up until but not including step for in lines 41 onward), navigate to object data properties and create a UV map. By default, Blender includes a 'UVMap', which should be renamed for convenience.

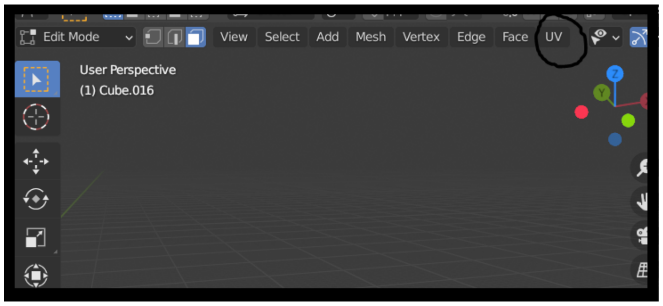


Fig. 2. UV button is available via 'Edit mode'.

Second, copy and paste your model for the first time as described in lines 43-45. Select the pasted model, head into edit mode and ensure every part is seledcted by pressing *a* on your keyboard and UV unwrap via the '*UV*' button. Preferably, select the Smart UV Project option, with an angle of 89 and an island margin of 0.03.

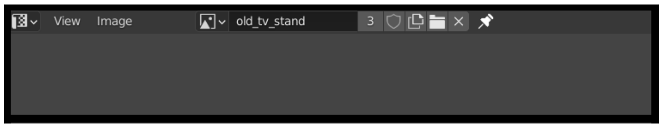


Fig. 3. Image button on the top-left creates a new image for the unwrap.

Third, in the bottom left window, the '*image*' button will highlighted with an asterisk, create a new image by clicking on it. The size should be 2048 x 2048 pixels and turn off the '*Alpha*' feature.

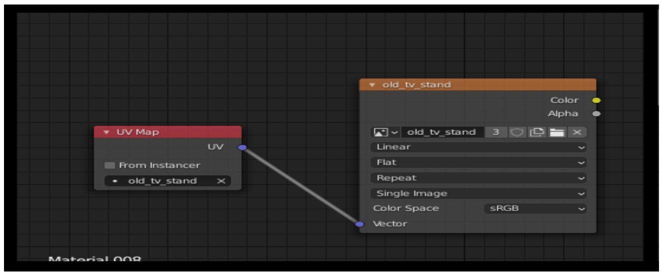


Fig. 4. How to connect the UV map and image modules in the shader editor.

Fourth, for all materials used by the selected 3D model, create a connected image-UV map module in the shader editor as shown above.

Fifth, navigate to '*Render properties*', by clicking the icon that looks like a camera. In this tab, select the Cycles engine, by default the Eevee engine will be used, but this does not have any baking capacity. Then, scroll down to the bake sub-tab, and select the options as shown in Figure 5. Note the baking procedure can take anywhere from a minute to half an hour, with the time largely depending on two factors: computer specifications and the complexity of the model.

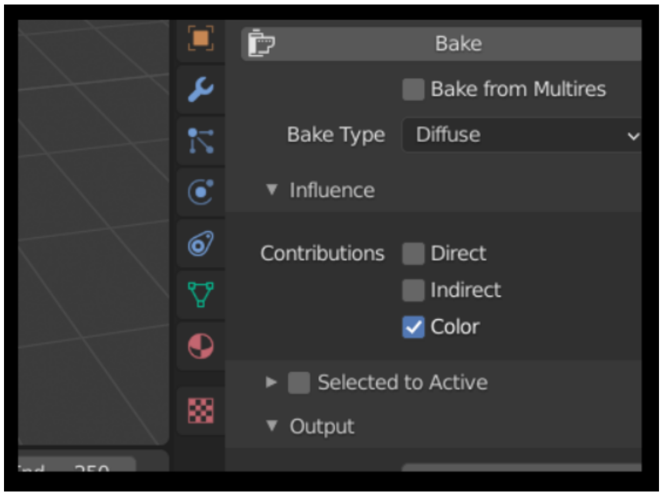

Bake
Bake from Multires
Bake Type: Diffuse
Influence
Contributions: Direct, Indirect, ✓ Color
Selected to Active
Output

Fig. 5. Baking settings to choose.

Applying baked texture

Once the baking procedure has completed, you will be required to save the image once again in the bottom-left window in a directory of your choosing. Then, follow steps *5-6* in lines 46-51. Following this, select the image-textured model, and export as an *FBX*, ensuring the '*Selected objects only*' option is clicked. The FBX object can now be imported into the Unity scene with appropriate shading/texturing.

Our 3D models

If you draw your attention to the working directory you can arrive at the Blender workspace by navigating in the following steps: **Root** $\longrightarrow$ **Assets** $\longrightarrow$ **BlenderAssets**. The '*.../BlenderAssets*' directory contains two sub-directories of interest: **.blend** and **fbx**. The former contains the raw models, available to the customer to be modified as needed. The latter contains the ready-to-import files. Recall the initial room plan:

All entities in the room plan have been constructed with 3 variations in their appropriate directories. Additionally, the corresponding materials, which are the result of the baking procedure, are saved within the '*.../blenderAssets/fbx/Materials*' directory.

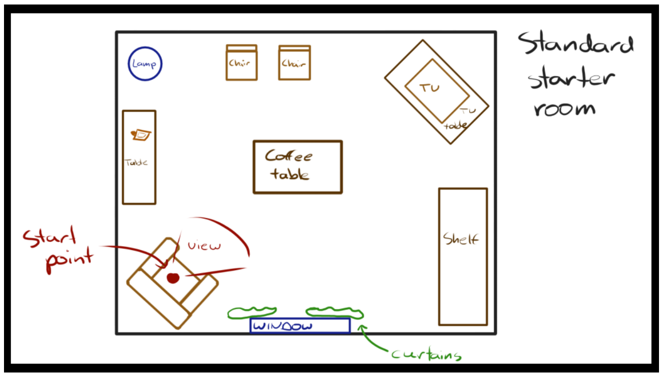


Fig. 6. The agreed upon room plan set in the initial stages of the project.

Further information

The authors strongly urge the reader to refer to the official Interface manual and Modelling guide. Additionally, one may refer to Ninja textures for image files of different materials should pre-existent textures be preferred. However, a disclaimer worth mentioning is that the variety in materials is often limited and so objects may not achieve the intended effect.

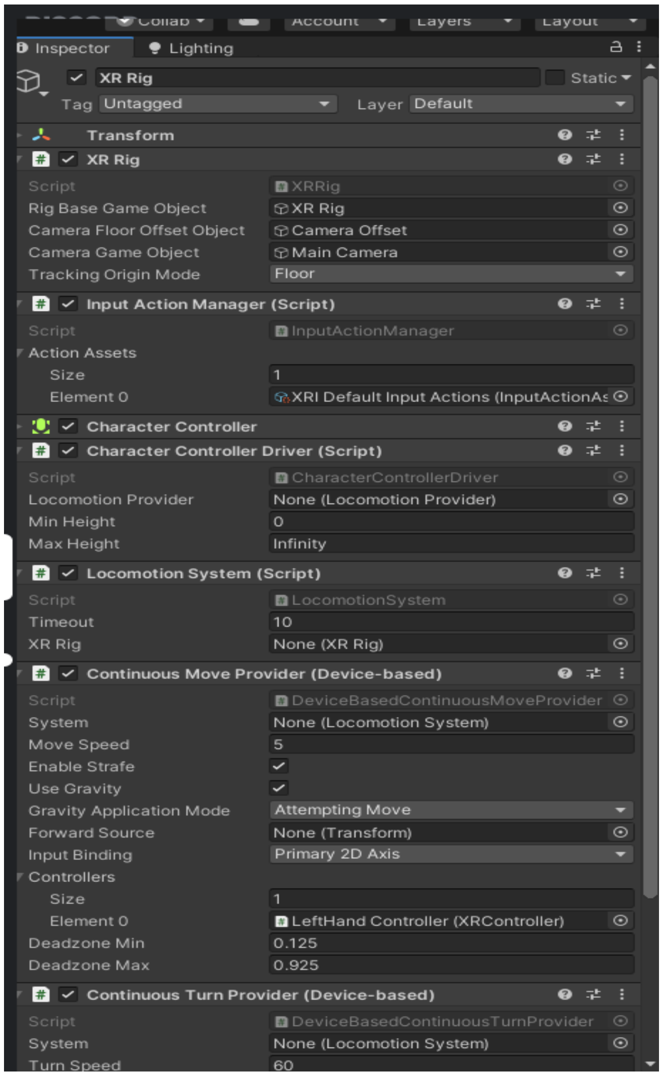
Creating VR experiences in Unity

To load the MemoryRoomVR in Unity, first download the entire MemoryRoomVR directory. Once downloaded, open Unity Hub and add a project. Select the MemoryRoomVR directory and ensure you are on Unity 2020+, with Android installed. Upon opening the project, if the compiler indicates any errors, make sure you have the XR Interaction Toolkit installed.

Unity's XR Interaction Toolkit

What will follow now is an introduction into the tools and packages used to port the desktop version of the virtual memory room to a room-scale VR experience, as well as an explanation of any custom scripts used in this process. The main technology used to complete this port was Unity's *XR Interaction Toolkit* a package which offers a cross-platform interaction management system for most major XR headsets currently available. This allowed us to develop for multiple platforms at the same time, and though we were focused on producing an Oculus Quest version of the VR application, it should be relatively straightforward to port to other devices within Unity,

using the XR Toolkit.
The XR Interaction toolkit supports basic controller interactions with basic interactable objects, as well as UI interaction with controllers. Furthermore, the toolkit allows you to access the controller's haptic and audio feedback, as well as a plug-in-and-play XR Camera Rig Solution, with a built-in locomotion system for continuous movement and continuous looking around. The Toolkit's simple interactables, make developing an interactive experience in a virtual reality environment extremely straight forward. In the next few sections, I will go into the Toolkit's specific components which allowed us to port the memory room to virtual reality. I will also expand on how these components can be changed to alter the virtual reality environment.



```
Inspector   Lighting
XR Rig                                   Static
Tag Untagged                 Layer Default
Transform
XR Rig
  Script                        XRRig
  Rig Base Game Object          XR Rig
  Camera Floor Offset Object    Camera Offset
  Camera Game Object            Main Camera
  Tracking Origin Mode          Floor
Input Action Manager (Script)
  Script                        InputActionManager
  Action Assets
    Size                        1
    Element 0                   XRI Default Input Actions (InputActionAs
Character Controller
Character Controller Driver (Script)
  Script                        CharacterControllerDriver
  Locomotion Provider           None (Locomotion Provider)
  Min Height                    0
  Max Height                    Infinity
Locomotion System (Script)
  Script                        LocomotionSystem
  Timeout                       10
  XR Rig                        None (XR Rig)
Continuous Move Provider (Device-based)
  Script                        DeviceBasedContinuousMoveProvider
  System                        None (Locomotion System)
  Move Speed                    5
  Enable Strafe                 ✓
  Use Gravity                   ✓
  Gravity Application Mode      Attempting Move
  Forward Source                None (Transform)
  Input Binding                 Primary 2D Axis
  Controllers
    Size                        1
    Element 0                   LeftHand Controller (XRController)
  Deadzone Min                  0.125
  Deadzone Max                  0.925
Continuous Turn Provider (Device-based)
  Script                        DeviceBasedContinuousTurnProvider
  System                        None (Locomotion System)
  Turn Speed                    60
```

Fig. 7. XR Rig Settings after correct set up.

The XR Rig

The XR Interaction Toolkit provides us with an XR Rig, containing a main camera, a camera offset, and left and right controllers. When adding such an XR Rig to an existing scene, we need to take a few things into account. Any pre-existing cameras that were being used for firs-person character controlling should be removed, as well as any references to the FP controller. This includes any scripts used, a GameManager if there was one, etc. Once the XR Rig's main camera is the only main camera in the scene, we add a few components to our rig to allow it to move in virtual reality. We need the XR Interaction Toolkit's Input Action Manager, where we attach XRI Default Input Actions to the action assets of the input action manager. Furthermore, we need to add a Character Controller, a Locomotion System, a Continuous Move Provider and a Continuous Turn Provider to the XR Rig. An XR Interaction Manager should have been automatically added to your hierarchy when set up correctly. Your settings should look similar to those in Figure 7.

Now let's explore each of the hand controllers. They both come with ray casters on the loading screen, which enable interaction between the rays and the loading menu. In the main memory room, both controllers have a hand animation without ray casters. This hand animation works by using the Oculus Integration prefabs, together with some scripts written by VR with Andrew. These scripts are contained in */Assets/Scripts/Basic/* and */Assets/Scripts/Extra/*, and take care of the necessary animation and physics. If the hand animation needs to be tweaked, these files can be edited appropriately to change the transformations being applied. Each controller also contains a direct interactor for interacting directly with scene interactables. Both controllers use Grip for selecting items and interacting with them, and allow for fists to be formed, and other hand gestures.

The Loading Screen

The VR Loading Screen contains a 3D plane and a large input menu, containing a keypad and two buttons. The input menu is made up of a canvas component, Using the ray casters from either the left or right controller allows for input by gripping the controller by the grip button. The loading screen expects a 3-6 digit code, which is supplied by the desktop client when logging in and selecting a profile. Upon failing to enter a correct code, the system asks the user to try again through a debug text below the submission button. See figure 8.

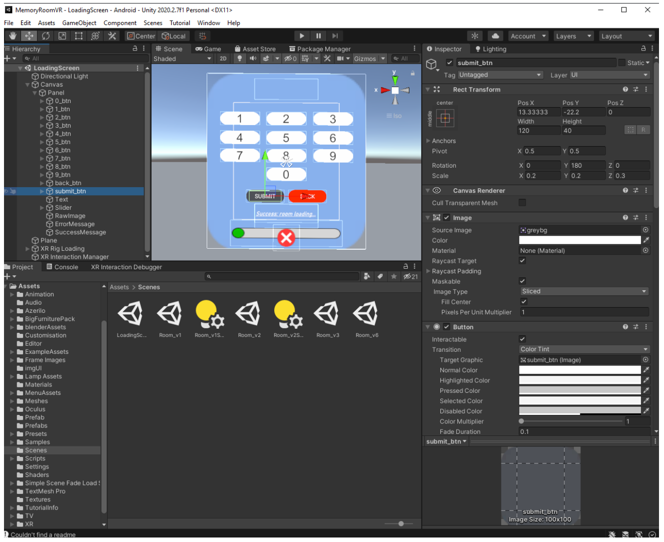


Fig. 8. The VR Loading Screen, in Unity's editor.

Memory Room Scene Set-up

In the VR Unity project, you will find that each variation of the memory room is saved as a separate scene. This ensures that referencing the scene is straightforward in the database, and makes loading the scenes asynchronously more simple. A simple button handler listening on the submit button checks the code entered and retrieves that room if a match is found. This also makes it easier to implement automatic customization's at some point in the future, given that the customization's can be stored in the database while referencing an already existing and accessible room stored in the APK.

The Virtual Reality Memory Room

The Virtual Reality Memory Room contains many of the same elements as the desktop version of the memory room, although requiring some additional tools from the XR Interaction Toolkit in order to make VR movement and interaction possible. Specifically, we require an XR Rig and an interaction manager, described above. Additionally, all models used in the virtual memory room require either a box or mesh collider in order to make the objects physical in the virtual environment.

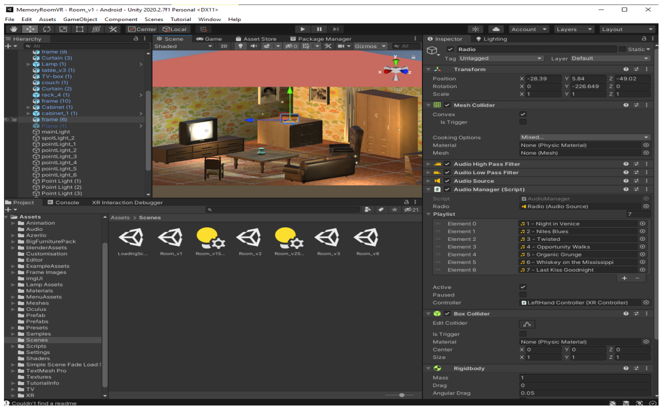


Fig. 9. The virtual reality memory room as it initially loads, in Unity's editor.

The interactable virtual radio

In each memory room there is a radio which represents the audio source for the music played in the memory room. This radio was made interactable using the XR Interaction Toolkit's simple interactable. The left dial on the radio has a listener attached when selecting the dial with the virtual hands, and pauses the music when triggered. The righ dial has the same type of listener and plays the music when triggered. The two press in buttons will skip to the next song when triggered. The radio is controlled by a script named AudioManager, found in */Assets/Audio/*, and has a reference to the audio source, the radio, and keeps track of whether the memory room is paused or not.

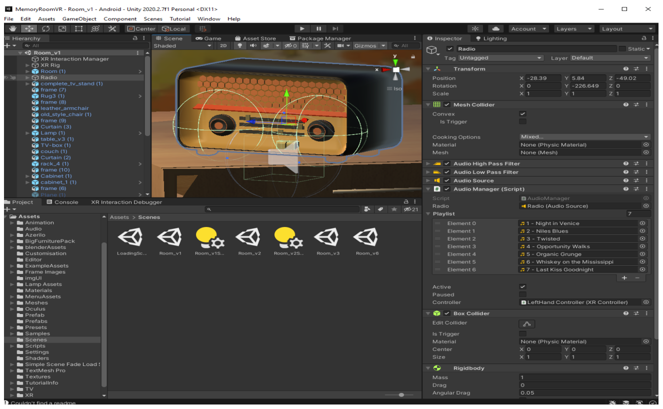


Fig. 10. The virtual reality memory room's interactable radio, in Unity's editor.

VR Picture Frames

In each memory room there is several empty picture frame game objects, which can display any images loaded in the projects assets. In order to apply an image to the frame, the image must first be converted to a material. The easiest way to do this, is to use a legacy render pipeline, such as diffuse, and to use the image as the selected texture. The materials will need to upgraded if you are using the universal render pipeline, and this can be done from Edit -> Render Pipeline -> Universal Render Pipeline -> Upgrade materials to URMaterials.

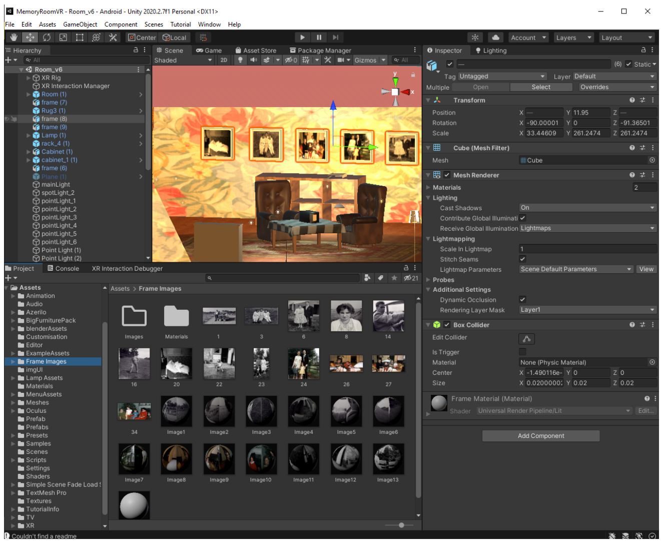


Fig. 11. The VR picture frames, in Unity's editor, along with the images and materials used to populate them.

Further information

The authors strongly urge the reader to refer to the official complete documentation for the XR Interaction Toolkit. The Oculus XR Plugin which allows you to build for Oculus devices using the XR Interaction Toolkit. Additionally, one may refer to Unity's official documentation for further topics.

Unity lighting

A key component in any Unity scene is the presence of appropriate lighting. There are many avenues at one's disposal to achieve this objective. In particular, given the employment of Blender and Unity as two, key separate tools for development of the product, both are known to have support for scene lighting. When faced with the reality that both teams relied on Blender in order to construct individual models, as opposed to an entire scene, this tool would not suffice. On the contrary, Unity has support for scene lighting in the form of real-time and baked. The latter relies on static scenes, which then go through a baking phase, producing the scene appropriately lit. Real-time lighting is more suitable, as we expect user-based movement in the scene and would like the lighting to reflect this. There are several modules that assist with this goal: directional-lights, spot-lights, and point-lights to name a few. The example below illustrates one manifestation of this approach to scene lighting.

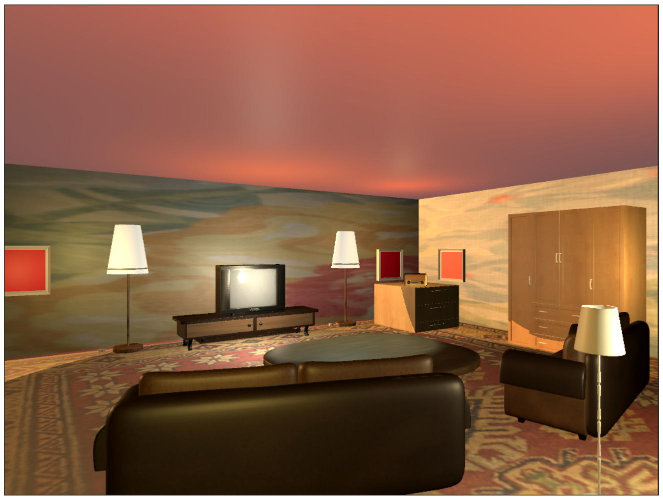


Fig. 12. One scene variation captured via the camera module.

Explanation of User Interface: Design Rules and Choices

The objective of this interface is to be simple and quick to navigate with a calm colour scheme evoking a clinical and professional feel to suit the environment in which it shall be used. The choice to smooth corners is inspired from modern interface design techniques and should be maintained throughout the interface. The colour scheme should be followed at all times as should the general style of the font faces chosen. All views should be centred with a title above and a blue background, all exit buttons should be red and all other buttons should be grey. Most font uses should be grey unless they are directly placed on a blue background in which they should be white.

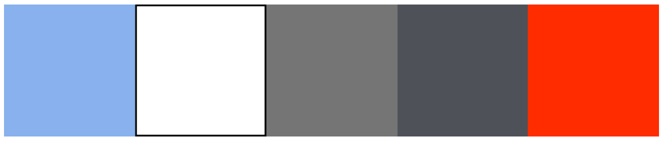


Fig. 13. Chosen colour scheme.

Asset Use within User Interface

There are two main asset types used in this interface: fonts and coloured materials. All fonts used are available with no royalties from google web fonts and the original font faces are included in the project files delivered. There are 2 fonts used, Oxygen Mono for titles above each view and Roboto in three different font weights for the content of each view. Titles for each view are in hex colour 757575 and font size 57. Buttons should generally use Roboto in bold when the button is large enough. Descriptions for input fields and any placeholder text should be in Roboto light. Roboto regular can be used for larger text such as code outputs or other pieces of information that should be highlighted for a user. Font sizes can be varied for effect though should generally not exceed the title font size of 57.

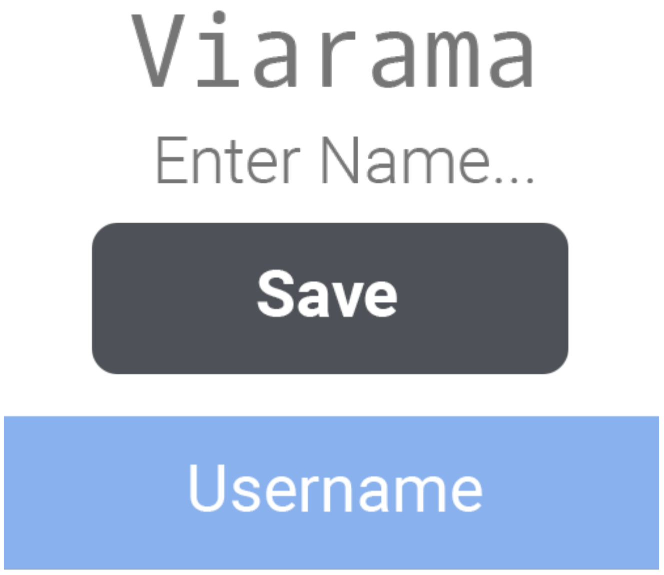


Fig. 14. Example Text Use.

The coloured materials are used for the background of each view and the buttons. Materials are required to achieve the bevelled corners of most elements in the UI, though colour of each material can be changed maintaining the corners multiple coloured materials are included for convenience. These are: Blue 88b1ed, Red ff2c00 and grey 4e5157. These colours should suffice for all elements that could be added to this UI, as such this colour scheme should be maintained. Each material is a 100x100 pixel square with 20 pixel bevelled corners saved as a PNG file to maintain the effect of smoothed corners.

Preparing New Assets

New font assets can be brought in simply by clicking the import new asset option when viewing project files in Unity, all fonts used have been imported as TTF files which have automatically been initialised as Unity font assets, all new elements including text including buttons should use the Unity text mesh pro asset type as this achieves the most control over each asset.

New materials can be added easily using Unity 2D Sprite Editor. A new image must be created in any image manipulation software such as GIMP *OpenSource* or Adobe Photoshop *Proprietary*, this image should be in the format previously mentioned with a colour that fits into the current colour scheme. This new image can be brought in as an asset by following the same process as fonts though some extra changes must be made.

Once you have an image imported into unity you must perform a 9-split on the image. This allows the image to be scaled from its current scale to any scale or ratio i.e a background for all content. This is achieved by setting the texture type of this imported image to 'Sprite 2D and UI' in the asset inspector, this allows for the mesh type of the image to be set from tight to full rect allowing for scaling. To 9-split the image click the button 'Sprite Editor' in the inspector and apply changes. Once the sprite editor loads change all border values to 20 in the bottom right corner, this will scale our smoothed corners correctly as Unity now recognises that there are different parts of the image that must be scaled differently. Click 'Apply' and your asset is complete, to add a new background create a new Image element in your scene and set the source image to your new asset. To create a new button select 'Button - TextMeshPro' and set its source image to your asset.

Unity And Firebase Integration

This project uses Firebase for back-end. It takes the email-Id and password of the person to login and register. The profile management system used in project is similar to Netflix, meaning a person can store 6 profiles under same email-Id. The person can create a profile by typing add profile button and typing his/her name. Each profile ask the person the room to be selected, stores that preference in the

Firebase and shows the code to the person to enter into that particular room. The person can then type that code before entering the VR room and it will take the person to the room selected.

Basic Working of Firebase

The script FirebaseManager manages everything related to Firebase.

1. When the app is started it calls the method Awake which sets ups the Firebase with the help of the function InitializeFirebase. If there are any issues with initialization it logs an error.

2. The registration process is done with the help of register function.The registration screen takes the username, email and password. This function checks for all the type of common errors that can happen, for e.g., missing email id, missing password, mismatch while confirming password. If none of the error happens then, it successfully creates a profile and logs an error if any.

3. The login process is done with the help of Login function. It takes email-Id and password and checks in the database if there exist an account with that Id and confirms the password. Again the Login function deals with all the types of basic errors that can happen in login.

4. After the person logs in, it takes the person to the profiles screen, where the user can create and edit profile. Once the user creates a profile it saves the options selected by the user in Firebase's Realtime Database and shows the code to enter the room.

Credentials of Firebase Account

Use the EmailId and password in the attached .txt file to access the Firebase Database. The authentication section in Firebase shows all the accounts that have been created. The Realtime Database in the Firebase shows details of all the accounts, i.e., username, number of profiles under that username and details of each profiles.

Acknowledgements

This project spanned two semesters. The progress made to date would not be possible without the co-operation of every team member, our coach: Viktor Szendrei, course lecturer: Tim Storer, and most certainly, our customer: Billy Agnew, without whom this project would not be possible. His project proposal caught our attention immediately, and throughout the last few months, we have learned new, transferable skills and have had the honour of acquainting ourselves with a wonderful individual.