Department of Computer Science
The University of Hong Kong

Final Year Project
VR/MR Simulation
Individual Final Report

Ho Kwan Kit (3035093173)
Supervisor: Dr. Vincent Lau

Abstract

Memory Palace, also commonly known as mind palace or the method of loci, is a memory system based on utilizing spatial memory. It has been proven to be an effective way for having quicker and stronger memory. However, the lack of ways to actualize the imagined scene results in a couple of difficulties in using this technique.

The intention of this project is to develop a Virtual Reality (VR) application to let users build their memory palace. The application will be run on Android phone and used together with a head-mounted display (HMD) such as Google Cardboard. Also uSens Fingo will be used for tracking hand motions. The VR application was developed with Unity 3D with a couple of external SDK.

This application can allow users to actualize their imaginary memory palace. People no longer have to rely purely on imagination ability but be able to see and interact with it. It is hoped that the application can help solving some of the major problems of Memory Palace System, as well as enhancing its effectiveness.

Acknowledgement

I would like to thank my supervisor Dr. Vincent Lau and project examiner Dr. Beta Yip for their support and advice of this project. I also want to thank my English class instructor Mr. Cazan Cezar for his teaching on writing this report.

Table of Content

List of Figures

Section I – Introduction

1.1 Background

1.1.1 Memory Palace

Memory Palace, also known as the method of loci, journey method or mind palace, is a memory system based on utilizing spatial memory. When using the system, people will visualize things they want to memorize, and mentally place the imagined objects into a place. It is found that humans are much better in memorizing spatial and graphical information than texts or concepts because our spatial memory is very powerful [1] but it is not utilized in remembering texts. Memory Palace aims to enhance memory by the utilization of spatial memory.

Memory Palace has been proven to be an effective memory enhancement technique. It is found that 90% of superior memorizers have used memory palace to aid in their memorization. [2]

Here is the detail of how the system is used:

1. Create a list of things to be memorized, e.g. a phone number, a list of names, etc.

2. Choose a familiar place (e.g. a street, home)

3. Pick locations in the place or on the route, each location for one items on the list

4. Find a symbol for each of the item on the list. The symbol is an object that has connection with the thing to memorize.
Here is an example for memorizing the phone number of HKU CS department:
2 - a tool like a hammer
8 – octopus card
5 – fire
9 – line
2 – tool
1 - won, represented by a trophy
8 - octopus card
0 - egg

5. Use imagination to visualize the symbols on the selected locations. For example, a hammer (2 - tool) on the bed, an octopus card on the desk (8 - octopus), etc.

To retrieve the information, go through the memory palace again, and recall the items one by one.

Although Memory Palace has proven to be effective, plenty of people have found difficulties in using the techniques, which has affected its effectiveness:

1. Pure imagination is not enough for visualization
The effectiveness of Memory Palace hugely relies on imagination ability of individuals. Whether the information memorized can become long-term memory depends on the strength of the sensory memory – touch, sight, smell, hearing and taste. [3] This implies that it is necessary to visualize the virtual scene as real as possible so as to deceive the brain that they are real. This is difficult, especially for people who have poor imaginary ability.

2. The memory palace is difficult to record
Although spatial memory is powerful, people can still forget the memory palace that have created. It would be better to record it down for future revisions. Drawings or texts can be used. However not everybody is able to draw and the visualization would be lost if using texts.

3. Running out of space to build memory palace
After using the system for a period of time, people might found that they have used up all of the places that they are familiar with. Some olds places would have to be reused which can confuse old memory. Some people have to go out and find a new place for building memory palace, which is inconvenient and time-consuming.

1.1.2 Virtual Reality

Virtual Reality (VR) is the technology to allow users to immerse themselves into a 3D digital world. Users can explore the digital world from a first person perspective similar to how they look at things in reality. The digital world is delivered through a headset. Common headset device includes Oculus Rift, Microsoft Hololens. Google Cardboard is also used together with mobile phone to act as a headset. Users may even be able to have interaction with things inside the digital world with the help of the controller of headsets or hand motion tracking device like Leap Motion and uSens Fingo.

VR is still a young technology that needs to be improved. Currently it has some limitations which have obstructed its prevalence. [4] For example, users might have the feeling of nausea and eye strain when using VR application. The major reason is the algorithm for head motion tracking is not ideal yet.

At this stage, VR is applied mostly to video games. Its use on other aspects such as business and data analytics have been discussed, yet most of them are still at experimental stage and not widely adopted on market.

1.2 Motivation

When the research for choosing the subject of the VR application in this project is being conducted, it is found that VR technology is advanced and has gained a lot of hype from the public, but there are opinions that VR is just for fanciness without being able to solve existing problem and improving working efficiency. The situation that VR's usage is mainly limited to game do provide a strong argument for those criticisms.

In this project, it is hoped that the VR application can have actual impact and solve some existing problems. It is noted that memory palace is a subject that really can use the help of VR to make significant improvement. VR presents itself as a possible tool to enhance its effectiveness by providing a realization of a virtual scene.
The detail of how VR can solve the above mentioned problems is discussed below:

1. Pure imagination is not enough for visualization
Everything can be visualized in the digital world. With the help of VR, memory palace users no longer need to rely on imagination to visualize a virtual scene. VR allows them to actually see the things and even interact with it. This can greatly strengthen memory through sensory.

2. The memory palace is difficult to record
The virtual scene is actualized inside VR. The virtual scenes created can be stored, retrieved and modified easily.

3. Running out of place to build memory palace
VR can provide affluent virtual places to be used for building memory palace. Those virtual places can be those exist in real world or created. It avoids the troubles to travel around physically to find new places.

1.3 Objectives

The objective of this project is to build a VR application for building memory palace. The intention of the application is to provide users an easy way to build and view the virtual scene created for using Memory Palace system. Moreover, the application can provide a way to allow users to actually see the memory palace that have built so as to strengthen users' memory. Currently, some people abandoned Memory Palace because they find it too hard or too time consuming to use and doubt its effectiveness. Hopefully, this application can help removes those current limitation. Moreover, this project is also meant to demonstrate a practical use of VR technology in the aspect other than games.

1.4 Scope

The application would provide a complete sets of features for user to freely use any place and any object to create memory palace.

Firstly, for place selection, the application provide a map for user to choose the location they want and the application would form a 3D environment with the photos of the place. Besides pictures from map, the application also allow user to upload a panorama photo of their own places.

Secondly, for objects in memory palace, the application provide a rich 3D object library. User can search for the object they want and freely import into their memory palace. The application also allow the user to upload pictures of the their own object.

When building memory palace, user can freely drag the object and place them into any location. User can input labels for the objects. After the creation, user can save the created memory palace and view or alter them later.

1.5. Previous Works in the field

There is a project called Macunx VR which is labelled as "a platform for building memory palaces in 3D and Virtual Reality”. The platform is still under development so the exact feature is unknown yet. It can be anticipated that it would provide similar functionality as the application.

1.6. Report Outline

The remainder of this report will provide details of the project. Then the methodology of implementation including hardware and development details will be given. Next, a complete review of the implemented application will be given. After that the potential future works will be outlined. This report will be closed with a the project working schedule and conclusion.

Section II – Methodology

2.1 Overview

The application run on Android platform and is used together with head mounted display and uSens Fingo. The main platform of development is Unity 3D. A couple of external software development kit (SDK) for Unity and Android Native development are used in the development process. This section will discuss the methodology of implementation in details.

2.2 Hardware

Figure 5 shows the complete hardware setup for the VR application.

1. Mobile Phone with Android 5.0 or above.

2. Head mounted display, Google Cardboard is currently used for development.

3. uSens Fingo, a hand motion tracking device to provide user interaction.

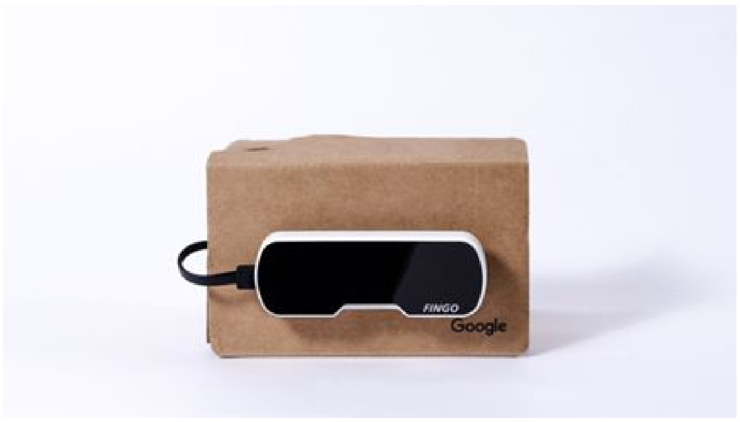


Figure 1 – Complete Hardware Setup of Google Cardboard and uSens Fingo

2.3 Development Environment

Unity 3D is the development platform. It is a 3D game engine which is popularly used for games and VR applications development. C# is used to write scripts in Unity.

Google VR SDK is integrated into the project to assist development. It provides some VR specific developer tools such as VR emulator on Windows and Instant Preview on Android to speed up development time.

Figure 2 - Unity 3D

Figure 3 - Google VR

2.4 Place Selection

The application provide a map for users to select places to build memory palace. The map is built with **Google Place API**. Since Google Place API doesn't provide support for Unity, API for Android is used and it is implemented as a plugin to be integrated into Unity project.

Figure 4 - Google Place

Google Place API provides a place picker (Figure 5). After the user has chosen the place, the API would return a JSON data of the information of the place such as address, latitude and longitude (Figure 6). The data is cached for later use in retrieving the images of the place.

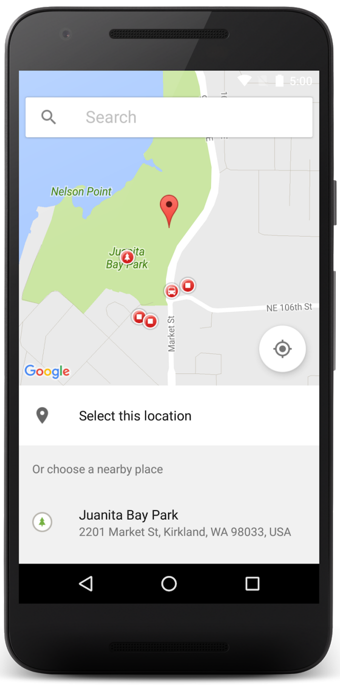


Figure 5 - Place Picker

```
[Place: Id=ChIJlzW0J4T_AzQR2MhvzdP1SCw,
Address=Pok Fu Lam, Hong Kong, Attrubutions=,
Name=The University of Hong Kong,
PhoneNumber=+852 2859 2111, Locale=,
PlaceTypes=University,PointOfInterest,Establishment,
PriceLevel=-1, Rating=4.4, Location=lat/lng:
(22.2829989,114.1370848), Viewport=[LatLngBounds
SW: lat/lng: (22.2799241,114.135058), NE: lat/lng:
(22.2848377,114.1412664)],
WebsiteUrl=http://www.hku.hk/]
```

Figure 6 - Place JSON Data Example

2.5 Upload Photo

The application allows users to upload photos of their own places and objects from the local storage of their phone. **Fantom Android Native Plugin**, which is an SDK downloaded from Unity Asset Store, is used to access the gallery of the Android phone. Once the user has selected the photo to be uploaded, the photo would be return as an byte stream. The stream of byte is encoded into PNG format using Unity API and is stored into the application's directory on local storage persistently.

Figure 7 - Fantom Android Native Plugin

2.6 Place 360 Degree Image

The images of the place need to be rendered as 360 degree 3D environment. To achieve that, a sphere is created in the scene and the images are used to form the material of the sphere.

Figure 8 - Google Street View

In the case that the place is chosen from map, the cached data of the place as mentioned in 2.4 would be used to request for the images with **Google Street View API**. The images of the 6 direction: front, back, left, right, up, top (Figure 9) are requested in runtime. The the images are used to create a cubemap. The cubemap is then used to create a new material and is applied onto the environment sphere (Figure 10).

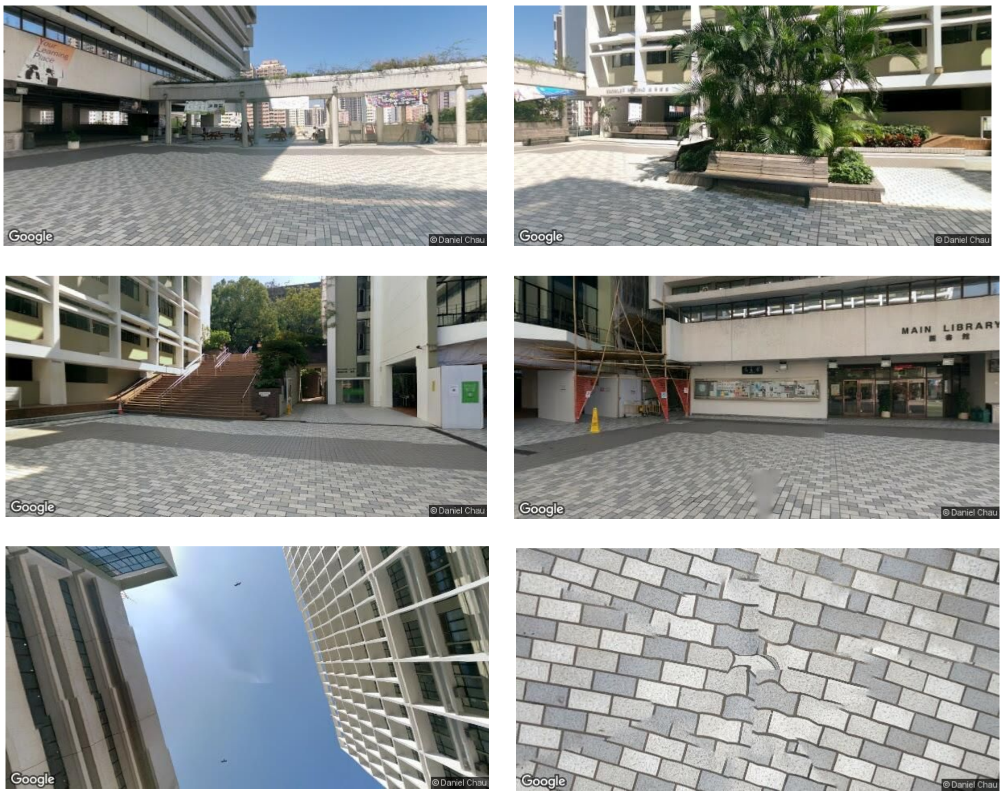


Figure 9 - Images of 6 directions: front, right, back, left, top, bottom

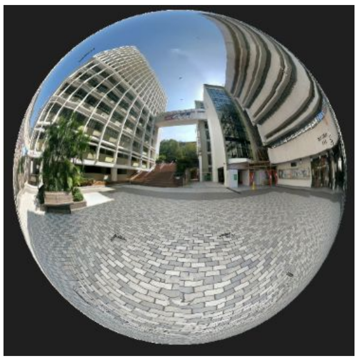


Figure 10 - Environment Sphere

In the case that the user upload photo of his own place, it is required that the photo needs to be a panorama photo (Figure 11). Taking panorama photo is very simple and common that most phone support natively. There are also a lot of Apps such as Google Cardboard Camera that provide this feature. The panorama photo would be scaled into 2:1 aspect so as to fit the size of the surface area of sphere. A shader has been implemented to render the panorama photo as material of the environment sphere.

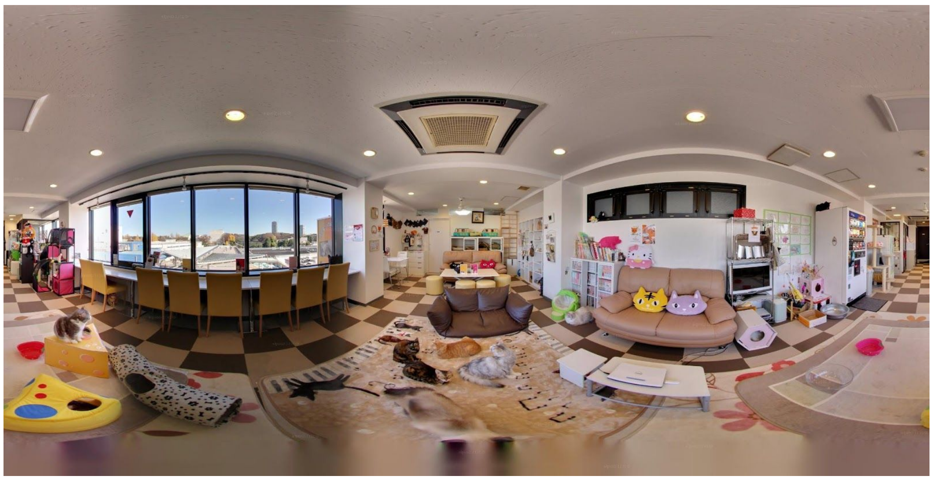


Figure 11 - Example Panorama Photo

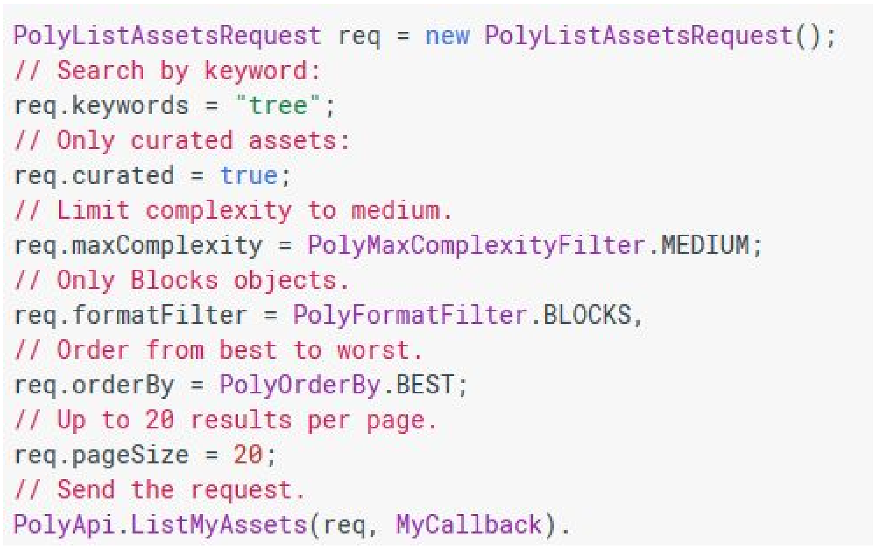
2.7 3D Objects Library

The application provides a 3D Objects library for user to search for the objects they want. The library is built with **Google Poly API**. Google Poly is a relatively new 3D model library website launched on November 2017. The API provides the feature of real-time import of 3D models into the scene of an application. Real time connection to Google Poly is preferred over including models statically into the application so as to leverage the growing content of Google Poly.

Poly

Figure 12 - Google Poly

In the application, when the user search with a keyword, the list model api is first called to fetch the names, thumbnails and other information of the models that match the searching request (Figure 13). The data is cached by wrapped into a PolyAsset class. After the user has selected the object, the get model api is called with the cached data to import the chosen model.



```
PolyListAssetsRequest req = new PolyListAssetsRequest();
// Search by keyword:
req.keywords = "tree";
// Only curated assets:
req.curated = true;
// Limit complexity to medium.
req.maxComplexity = PolyMaxComplexityFilter.MEDIUM;
// Only Blocks objects.
req.formatFilter = PolyFormatFilter.BLOCKS,
// Order from best to worst.
req.orderBy = PolyOrderBy.BEST;
// Up to 20 results per page.
req.pageSize = 20;
// Send the request.
PolyApi.ListMyAssets(req, MyCallback).
```

Figure 13 - Example List Asset Rquest

To avoid the blocking of user-interface, listing and getting models are executed parallelly with the main thread. For better performance, only model with low or middle complexity would be offered to the user since model with high complexity often take more than 1 to import, while middle complexity models take no more than 5 seconds.

Figure 14 - Example 3D Models on Google Poly

2.8 User Interaction

Usens Fingo is used in this project to achieve user interaction. Fingo is a hand motion tracking device to render the motion of hands in real time.

The implementation of the user interaction is achieved with **Fingo SDK**, which support Unity naturally. There are mainly 2 types of action in the application: clicking buttons and grabbing and placing 3D objects.

Figure 15 - Fingo SDK

To let objects be touchable by Fingo hands, collider and rigidbody are attached to them. An invisible sphere collider is attached to the fingertip of the Fingo hand and the collision between sphere and objects is detected to achieve clicking. For grabbing and placing, The distance and angle between fingers and palm is calculated to detect if an object is being grabbed, then the position of objects would be changed according to the position of hands to achieve placing objects in desired location.

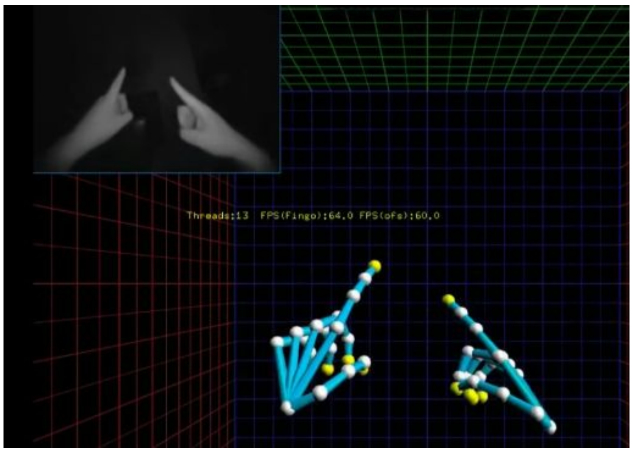


Figure 16 - Hand Motion Tracking

2.9 User Input

Input text is needed in 2 areas in the application: searching 3D models and labelling objects. 2 input method are supported: virtual keyboard and voice input.

Virtual keyboard is rendered as an object in the application (Figure 17). The original plan is to make the keyboard clickable with Fingo hands. Yet, it is found that it is difficult to use since the size of keys are small and Fingo Hand is not delicate enough to track the clicking precisely. Eventually, eye-gazing is used as the input method of the keyboard. The key that is in the center of the gaze would be focused. Then user can press the button on the top of Google Cardboard, which emulate clicking the screen of phone to type the key.

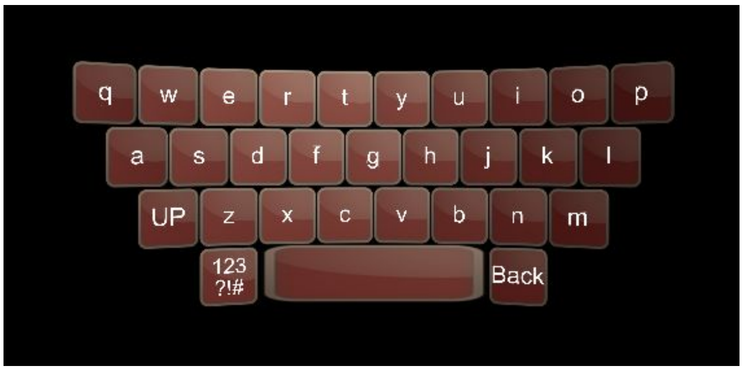


Figure 17 - Virtual Keyboard

After taking the advice from supervisor, voice input is also implemented to provide better user experience. Voice input is achieved with the access to the text-to-speech(TTS) engine built in Android phone. Phone with Android 5.0 or above all support speech input natively. There are a lot of TTS engine on the market. Common engine include Google TTS, Acapela and Pico TTS.

Figure 18 - Google Text-to-Speech

Figure 19 - Acapela

2.10 User Interface

A good-looking and user-friendly user interface is essential to the success of an application. Due to the lack of skill in graphic design, some external resources are utilized to overcome the difficulties.

UI - Builder is an SDK purchased on Unity asset Store. It is a customizable UI kit that provide ready-made UI components with good design and style. The asset is used in making layout and components such as input box and buttons.

Inventory Engine is another asset purchased on Unity asset Store. It is used to build the 3D object library. Modification has been made to support interaction with Fingo hand and new features such as pagination and searching.

Figure 20 - UI Builder

Figure 21 - Inventory Engine

2.11 Storage

The data of the created memory palace and the uploaded photos need to be stored persistently for future retrieval. At the time of making project plan, it is planned to build a Node.JS web server connected to a NoSQL database such as MongoDB to handle to storage and retrieval. Yet due to new works with higher priority found and the limitation of time, external database is not implemented at this stage. Currently the data is stored on the local storage space on the phone.

The storage of uploaded photos is mentioned in section 2.5. The data of the created memory palace is used to create new Objects. The objects are serialized into binary stream and stored in storage space. It can be deserialized later when needed. Details of implementation will be discussed in next chapter.

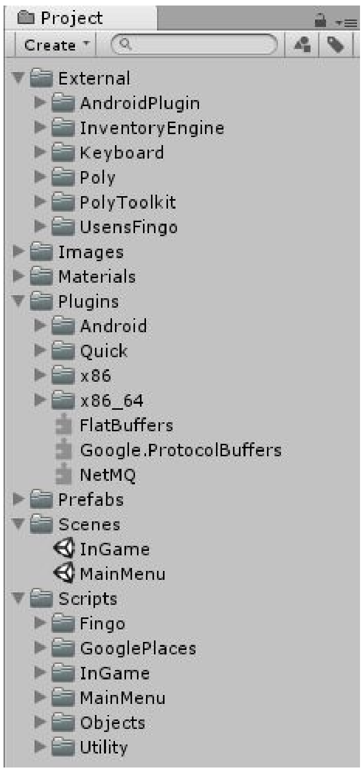
Section III - Implementation Details

3.1 Overview

This section will explain the detail of the Unity project of the application. The organization and structure of the project and some source code of the key component will be explained.

3.2 Project Structure

The file organization of the Unity project is shown in figure 22. There are 2 scenes in the "Scenes" folder: "MainMenu" scene is for the user menu and the google map, while "InGame" scene is for the building of memory palace in VR. The "External" folder stores the external library used in the project such as Poly SDK and Fingo SDK. The Plugins folder stored the Android plugin which would be explained later in this chapter and the plugin imported by external SDK such Fingo. "Prefabs" folder stores the reusable and dynamically created game objects in the scenes. "Scripts" folder store the scripts that are attached to the game objects.



```
Project
Create
▼ External
    ▶ AndroidPlugin
    ▶ InventoryEngine
    ▶ Keyboard
    ▶ Poly
    ▶ PolyToolkit
    ▶ UsensFingo
▶ Images
▶ Materials
▼ Plugins
    ▶ Android
    ▶ Quick
    ▶ x86
    ▶ x86_64
      FlatBuffers
      Google.ProtocolBuffers
      NetMQ
▶ Prefabs
▼ Scenes
      InGame
      MainMenu
▼ Scripts
    ▶ Fingo
    ▶ GooglePlaces
    ▶ InGame
    ▶ MainMenu
    ▶ Objects
    ▶ Utility
```

Figure 22 - Project Structure

3.3 Google Map

The Google Map is integrated into the project as an Android native plugin. Android native plugin in Unity is Android-specific native code library. C# api of Unity does not support system call to Android, but Unity has specification on how Android native feature can be imported as plugin.

Google Place API for Android is used and built as a Java project with Android Studio. Then the generated file such as JAR, AAR file and the AndroidManifest.xml file are placed under the **Plugins/Android** folder as specified by Unity. After that C# scripts are written to call the Java api (Figure xx). Those files are placed under Scripts/GooglePlaces folder.

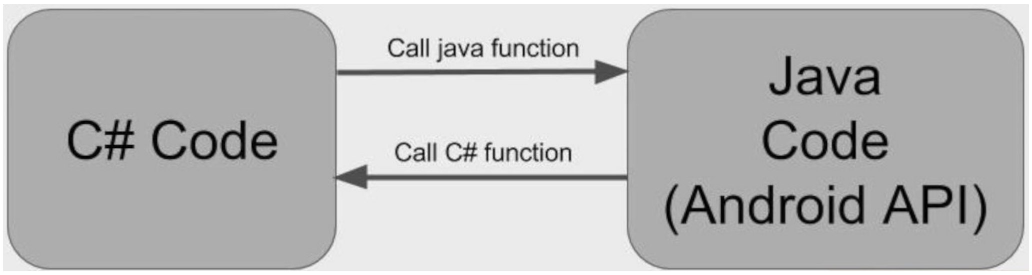
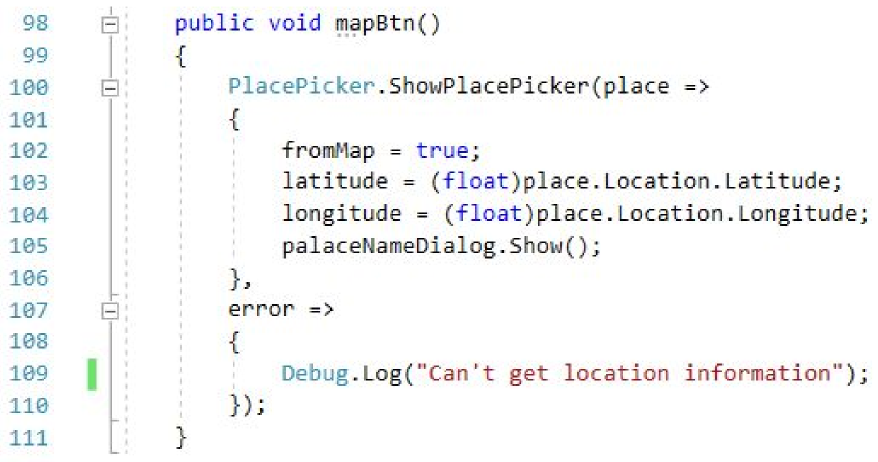


Figure 23 - System Call to Android Native Library

Once the user has selected the location, the data is returned in JSON format as mentioned in section 2.4. The JSON data is parsed to an object with the api from **MiniJSON.cs**. The data now is able to be used by the C# script to create a street view (Figure 24).

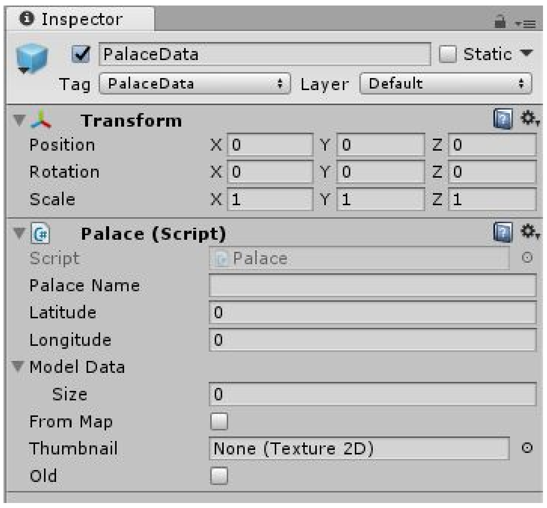


```
public void mapBtn()
{
    PlacePicker.ShowPlacePicker(place =>
    {
        fromMap = true;
        latitude = (float)place.Location.Latitude;
        longitude = (float)place.Location.Longitude;
        palaceNameDialog.Show();
    },
    error =>
    {
        Debug.Log("Can't get location information");
    });
}
```

Figure 24 - Code Snippet for Using the Parsed Location Data to Create Street View

3.4 Load Street View

After the user has chosen the location, the application would switch from **MainMenu** scene to **InGame** scene. To bring the data of the location across the scenes, a game object named **PalaceData** (Figure 25)is created before the switch. **“DontDestroyOnLoad”** method from Unity api is called to make **PalaceData** be able to persist across different scenes.



Inspector

PalaceData ✓ | Static
Tag PalaceData | Layer Default

Transform

Position	X 0	Y 0	Z 0
Rotation	X 0	Y 0	Z 0
Scale	X 1	Y 1	Z 1

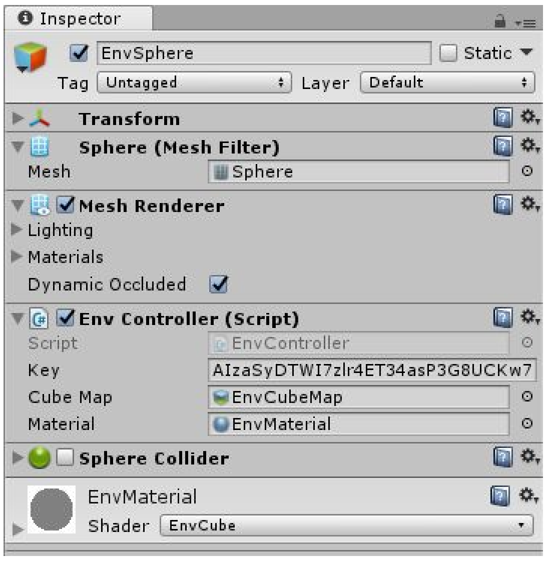
Palace (Script)

Script	Palace
Palace Name	
Latitude	0
Longitude	0
Model Data	
Size	0
From Map	☐
Thumbnail	None (Texture 2D)
Old	☐

Figure 25 - PalaceData Game Object

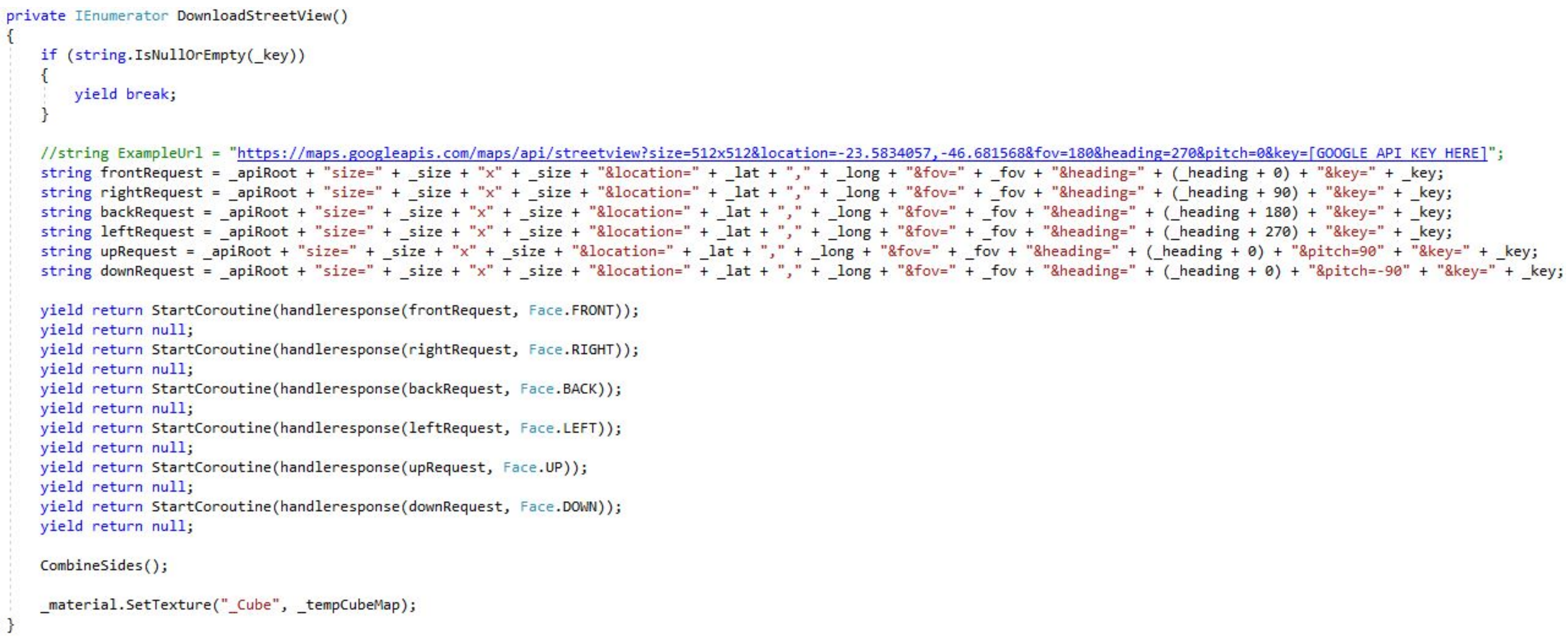
In the **InGame** scene, there is a game object called **EnvSphere** (Figure 26) responsible for rendering the street view. Once the **InGame** scene is loaded, the data of the location would be extracted from **PalaceData** and used to download the pictures for the 6 sides of the street view (Figure 27).

Since there is network transfer overhead in downloading images, the download function both implemented as IEnumerator to avoid the user interface from being blocked. The execution of IEnumerator functions are delayed while staying alive in the background. The execution would be resumed once the data is ready.



```
Inspector
EnvSphere                         Static
Tag  Untagged        Layer  Default
Transform
Sphere (Mesh Filter)
Mesh                 Sphere
Mesh Renderer
Lighting
Materials
Dynamic Occluded     ✓
Env Controller (Script)
Script               EnvController
Key                  AIzaSyDTWI7zlr4ET34asP3G8UCKw7
Cube Map             EnvCubeMap
Material             EnvMaterial
Sphere Collider
EnvMaterial
Shader  EnvCube
```

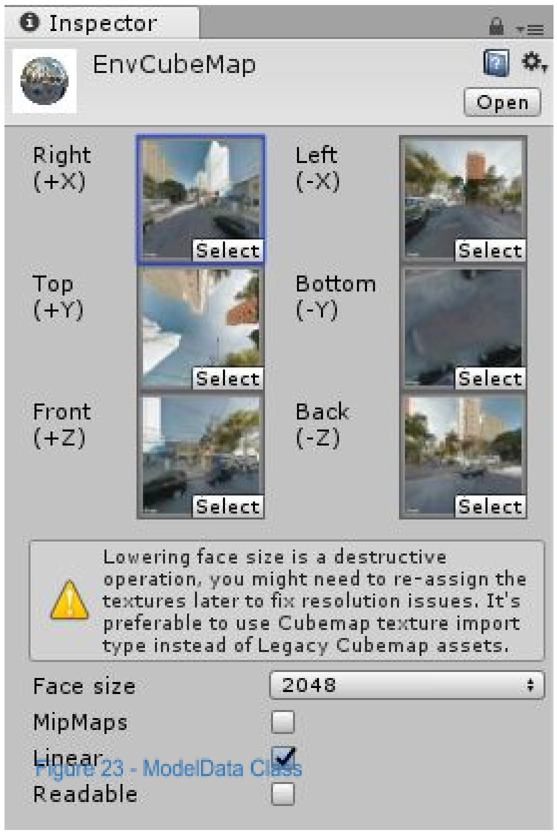
Figure 26 - EnvSphere Game Object



```
private IEnumerator DownloadStreetView()
{
    if (string.IsNullOrEmpty(_key))
    {
        yield break;
    }

    //string ExampleUrl = "https://maps.googleapis.com/maps/api/streetview?size=512x512&location=-23.5834057,-46.681568&fov=180&heading=270&pitch=0&key=[GOOGLE API KEY HERE]";
    string frontRequest = _apiRoot + "size=" + _size + "x" + _size + "&location=" + _lat + "," + _long + "&fov=" + _fov + "&heading=" + (_heading + 0) + "&key=" + _key;
    string rightRequest = _apiRoot + "size=" + _size + "x" + _size + "&location=" + _lat + "," + _long + "&fov=" + _fov + "&heading=" + (_heading + 90) + "&key=" + _key;
    string backRequest = _apiRoot + "size=" + _size + "x" + _size + "&location=" + _lat + "," + _long + "&fov=" + _fov + "&heading=" + (_heading + 180) + "&key=" + _key;
    string leftRequest = _apiRoot + "size=" + _size + "x" + _size + "&location=" + _lat + "," + _long + "&fov=" + _fov + "&heading=" + (_heading + 270) + "&key=" + _key;
    string upRequest = _apiRoot + "size=" + _size + "x" + _size + "&location=" + _lat + "," + _long + "&fov=" + _fov + "&heading=" + (_heading + 0) + "&pitch=90" + "&key=" + _key;
    string downRequest = _apiRoot + "size=" + _size + "x" + _size + "&location=" + _lat + "," + _long + "&fov=" + _fov + "&heading=" + (_heading + 0) + "&pitch=-90" + "&key=" + _key;

    yield return StartCoroutine(handleresponse(frontRequest, Face.FRONT));
    yield return null;
    yield return StartCoroutine(handleresponse(rightRequest, Face.RIGHT));
    yield return null;
    yield return StartCoroutine(handleresponse(backRequest, Face.BACK));
    yield return null;
    yield return StartCoroutine(handleresponse(leftRequest, Face.LEFT));
    yield return null;
    yield return StartCoroutine(handleresponse(upRequest, Face.UP));
    yield return null;
    yield return StartCoroutine(handleresponse(downRequest, Face.DOWN));
    yield return null;

    CombineSides();

    _material.SetTexture("_Cube", _tempCubeMap);
}
```

Figure 27 - Code Snippet for Downloading Street View Images

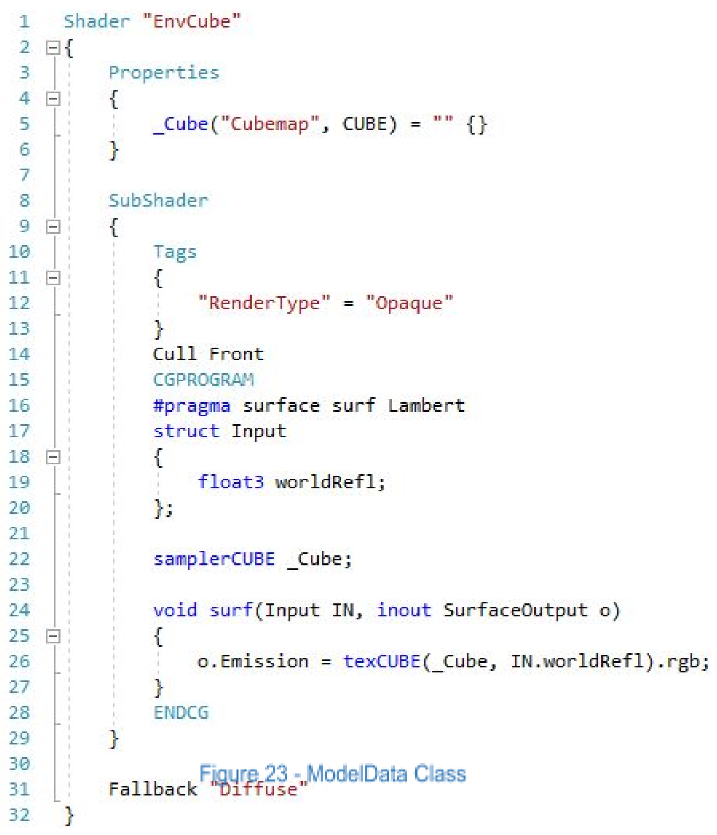
The downloaded images are used to build a cubemap (Figure 28) and be applied on **EnvSphere**. The main camera which correspond to user's point of view is placed at the center of **EnvSphere.** In order for the street view to be viewed from the inside of the sphere, the new Shader is implemented (Figure 29) to create a special material for **EnvSphere.**



Lowering face size is a destructive operation, you might need to re-assign the textures later to fix resolution issues. It's preferable to use Cubemap texture import type instead of Legacy Cubemap assets.

Face size 2048
MipMaps ☐
Linear ☑
Readable ☐

Figure 23 - ModelData Class



```
Shader "EnvCube"
{
    Properties
    {
        _Cube("Cubemap", CUBE) = "" {}
    }

    SubShader
    {
        Tags
        {
            "RenderType" = "Opaque"
        }
        Cull Front
        CGPROGRAM
        #pragma surface surf Lambert
        struct Input
        {
            float3 worldRefl;
        };

        samplerCUBE _Cube;

        void surf(Input IN, inout SurfaceOutput o)
        {
            o.Emission = texCUBE(_Cube, IN.worldRefl).rgb;
        }
        ENDCG
    }

    Fallback "Diffuse"
}
```

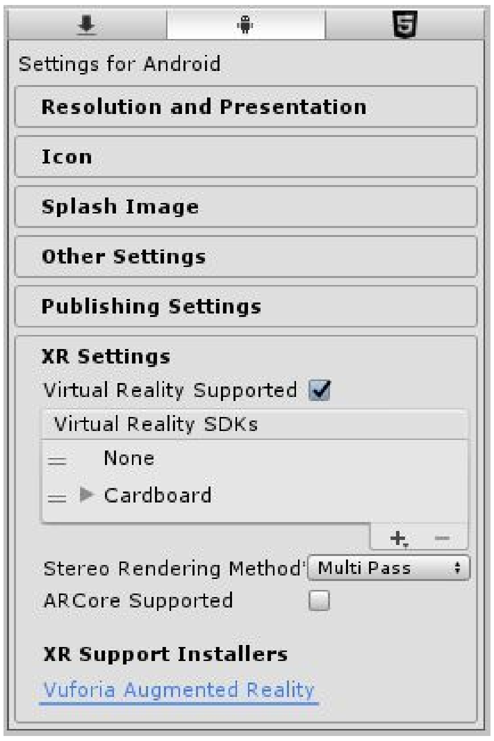
Figure 23 - ModelData Class

Figure 28 - CubeMap for EnvSphere

Figure 29 - Shader for the Material of EnvSphere

3.5 Switching Between normal and VR mode

The application is a hybrid app that it starts as normal mode in the main menu and enters VR mode when the user starts building memory palace. To achieve that, in the build setting of the project, Virtual Reality support is set to be true and 2 SDKs are selected: **“None”** and **“Cardboard”** (Figure 30). **“None”** would be executed just like normal mode.

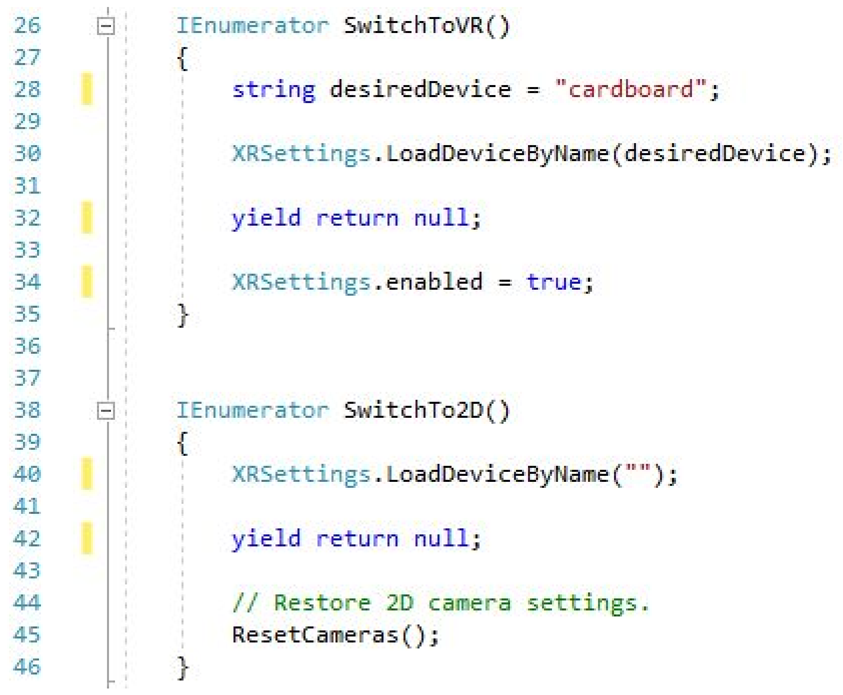


Settings for Android

Resolution and Presentation
Icon
Splash Image
Other Settings
Publishing Settings
XR Settings
Virtual Reality Supported ☑
Virtual Reality SDKs
None
Cardboard
Stereo Rendering Method Multi Pass
ARCore Supported ☐
XR Support Installers
Vuforia Augmented Reality

Figure 30 - VR Setting of the Project

The 2 SDKs are loaded accordingly with IEnumerator function (Figure 31).



```
IEnumerator SwitchToVR()
{
    string desiredDevice = "cardboard";

    XRSettings.LoadDeviceByName(desiredDevice);

    yield return null;

    XRSettings.enabled = true;
}


IEnumerator SwitchTo2D()
{
    XRSettings.LoadDeviceByName("");

    yield return null;

    // Restore 2D camera settings.
    ResetCameras();
}
```

Figure 31 - Code Snippet for Switching Mode

3.6 Google Poly

Google Poly SDK is used for building the 3D object library. The api allow applications to dynamically import 3D objects into the project at runtime. The function calls to the api are implemented as a chain of callback function as explained below.

For displaying objects in the library (Figure 32), **ListAssests** function is called and **ListCallback** is passed as callback function. Inside **ListCallback**, the **FetchThumbnail** function is called to get the picture of the asset and **getThumbnailCallback** is passed as the callback function to display the picture of the model in the library. Importing model work similarly as a chain of callback (Figure 33).

```
private IEnumerator ListingAsset()  // List Model, implemented as IEnumerator to avoid blocking UI
{
    yield return null;
    PolyApi.ListAssets(request, ListCallback);  //Get a list of 3D Models
}

void ListCallback(PolyStatusOr<PolyListAssetsResult> result)
{
    if (result.Value.nextPageToken != null && page + 1 == pageTokens.Count) pageTokens.Add(result.Value.nextPageToken);
    foreach (PolyAsset asset in result.Value.assets)
    {
        PolyApi.FetchThumbnail(asset, getThumbnailCallback);   //Fetch the thumbnail picture of each 3D models
    }
}

void getThumbnailCallback(PolyAsset asset, PolyStatus status)  //Add the 3D objects to the library
{
    BaseItem item = new BaseItem();
    item.ItemID = asset.name;
    item.TargetInventoryName = "ModelInventory";
    item.ItemName = asset.displayName;
    item.ShortDescription = asset.description;
    item.Description = asset.description;
    Rect rectangle = new Rect(0, 0, asset.thumbnailTexture.width, asset.thumbnailTexture.height);
    item.Icon = Sprite.Create(asset.thumbnailTexture, rectangle, new Vector2(0.5f, 0.5f));
    //Use the thumbnail to create a sprite and display on the UI
    item.MaximumStack = 10;
    inventory.AddItem(item, 1);
}
```

Figure 32- Code Snippet for Displaying Objects in the Library

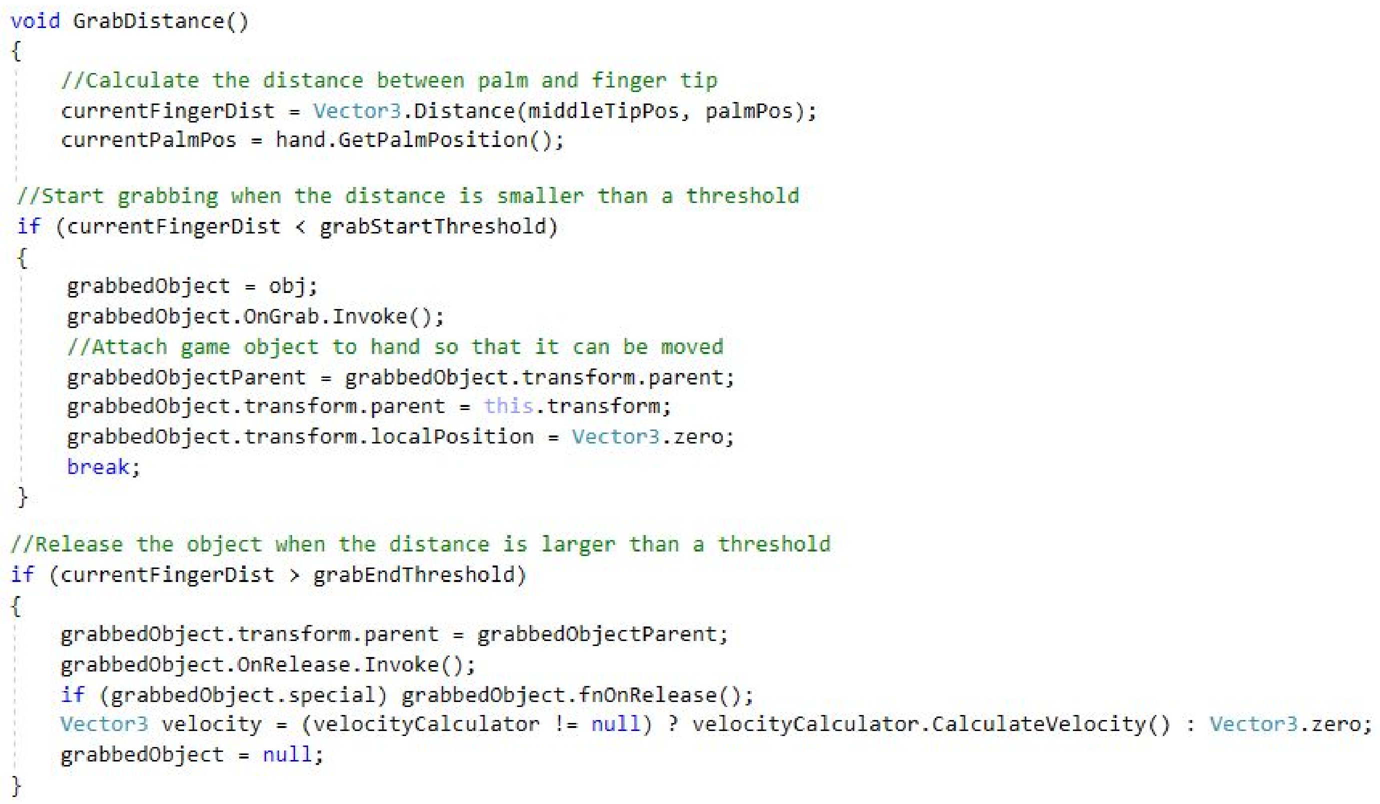
```
// Import model into Memory Palace, implemented as IEnumerator to avoid blocking UI
private IEnumerator importModel(string ID)
{
    yield return null;
    PolyApi.GetAsset(ID, getAssetCallback);  // Use the stored ID to get the data of model
}

void getAssetCallback(PolyStatusOr<PolyAsset> result)
{
    PolyImportOptions option = PolyImportOptions.Default();
    //Setting scale and size for the model to be imported.
    option.rescalingMode = PolyImportOptions.RescalingMode.FIT;
    option.desiredSize = 0.1f;
    PolyApi.Import(result.Value, option, importCallback);  // Import model
}

void importCallback(PolyAsset asset, PolyStatusOr<PolyImportResult> result)
{
    //Wrap the imported Model with a ModelPrefab gameobject.
    GameObject model = Instantiate(ModelPrefab, new Vector3(0, 0, 0.5f), Quaternion.identity, ModelHolder.transform);
    model.GetComponent<ModelController>().poly = true;
    model.GetComponent<ModelController>().polyIdOrFileName = asset.name;
    //Set the position of the model to be right in front of user
    model.transform.RotateAround(new Vector3(0, 0, 0), Vector3.up, GameObject.FindWithTag("MainCamera").transform.eulerAngles.y);
    result.Value.gameObject.transform.SetParent(model.transform);
    result.Value.gameObject.transform.localPosition = new Vector3(0, 0, 0);
}
```

Figure 33- Code Snippet for Importing Objects

3.7 Grabbing and Moving Objects

Fingo SDK is used for implementing the interaction between the Fingo hand and game objects. **FingoHands** game objects is attached to Main Camera so that it would always stay in front of the user no matter how the point of view shift and rotate.

GrabHand.cs script is implemented to handle the grabbing and moving of game objects. The grabbing is implemented by calculating the distance between finger tips and the palm to decide if the user is making the grab gesture (Figure 34). Once grabbing is detected, the game objects would be attached to **FingoHands** as a child so that the game object's position will change accordingly with the hands to achieve moving.

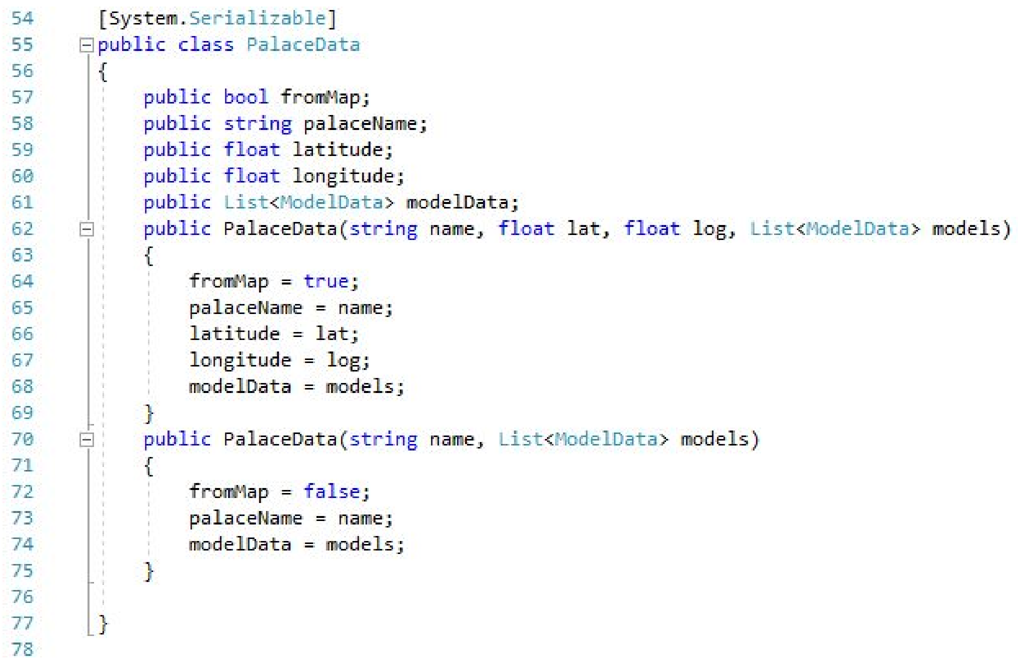


```
void GrabDistance()
{
    //Calculate the distance between palm and finger tip
    currentFingerDist = Vector3.Distance(middleTipPos, palmPos);
    currentPalmPos = hand.GetPalmPosition();

 //Start grabbing when the distance is smaller than a threshold
 if (currentFingerDist < grabStartThreshold)
 {
     grabbedObject = obj;
     grabbedObject.OnGrab.Invoke();
     //Attach game object to hand so that it can be moved
     grabbedObjectParent = grabbedObject.transform.parent;
     grabbedObject.transform.parent = this.transform;
     grabbedObject.transform.localPosition = Vector3.zero;
     break;
 }

//Release the object when the distance is larger than a threshold
if (currentFingerDist > grabEndThreshold)
{
    grabbedObject.transform.parent = grabbedObjectParent;
    grabbedObject.OnRelease.Invoke();
    if (grabbedObject.special) grabbedObject.fnOnRelease();
    Vector3 velocity = (velocityCalculator != null) ? velocityCalculator.CalculateVelocity() : Vector3.zero;
    grabbedObject = null;
}
```

Figure 34 - Code Snippet for Grabbing and Moving Objects

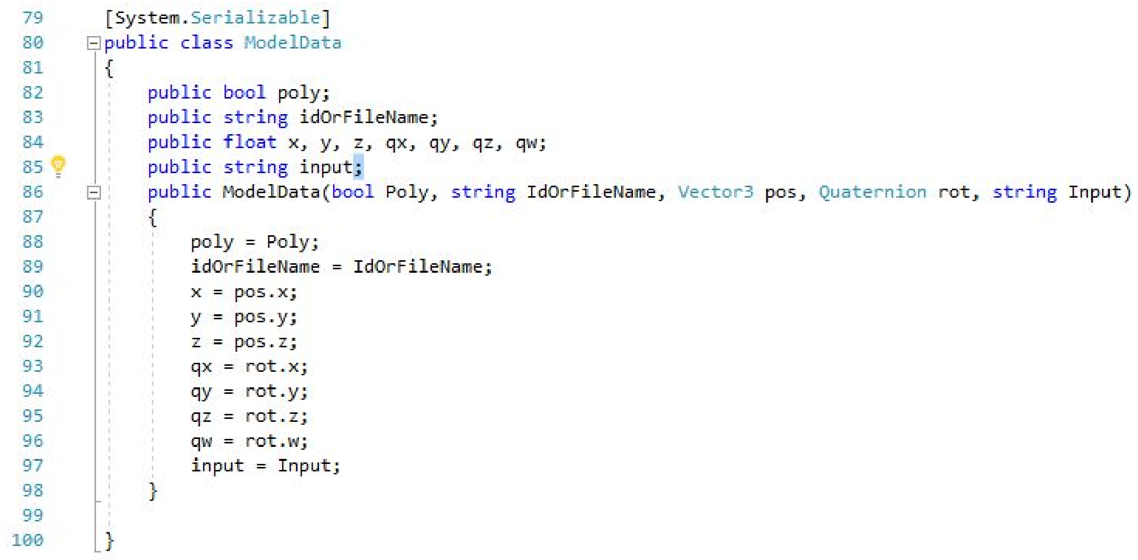
3.8 Saving Data

To save the created memory palace, an instance of **PalaceData** is created (Figure 35) which the name, location and 3D model data are stored. For each 3D models, the data is wrapped into an ModelData object (Figure 36). Data such as ID on Poly library or the location of the image on local storage, position and rotation are stored .



```
[System.Serializable]
public class PalaceData
{
    public bool fromMap;
    public string palaceName;
    public float latitude;
    public float longitude;
    public List<ModelData> modelData;
    public PalaceData(string name, float lat, float log, List<ModelData> models)
    {
        fromMap = true;
        palaceName = name;
        latitude = lat;
        longitude = log;
        modelData = models;
    }
    public PalaceData(string name, List<ModelData> models)
    {
        fromMap = false;
        palaceName = name;
        modelData = models;
    }

}
```

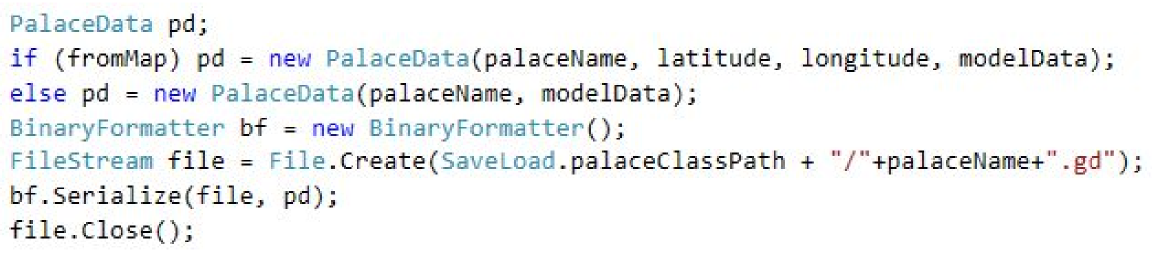
Figure 35 - Code Snippet for PalaceData Class



```
[System.Serializable]
public class ModelData
{
    public bool poly;
    public string idOrFileName;
    public float x, y, z, qx, qy, qz, qw;
    public string input;
    public ModelData(bool Poly, string IdOrFileName, Vector3 pos, Quaternion rot, string Input)
    {
        poly = Poly;
        idOrFileName = IdOrFileName;
        x = pos.x;
        y = pos.y;
        z = pos.z;
        qx = rot.x;
        qy = rot.y;
        qz = rot.z;
        qw = rot.w;
        input = Input;
    }

}
```

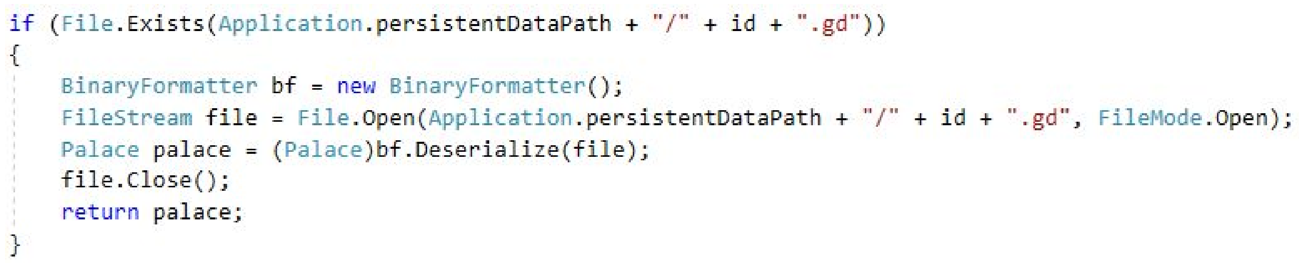
Figure 36 - Code Snippet for ModelData Class

The created object is serialized into binary file with C#'s **BinaryFormatter** class (Figure 37). The file is stored under **Application.persistentDataPath** directory which is specified by Unity api. The binary file can be deserialized later when loaded. (Figure 38)



```
PalaceData pd;
if (fromMap) pd = new PalaceData(palaceName, latitude, longitude, modelData);
else pd = new PalaceData(palaceName, modelData);
BinaryFormatter bf = new BinaryFormatter();
FileStream file = File.Create(SaveLoad.palaceClassPath + "/"+palaceName+".gd");
bf.Serialize(file, pd);
file.Close();
```

Figure 37 - Code Snippet for Serializing and Saving ModelData



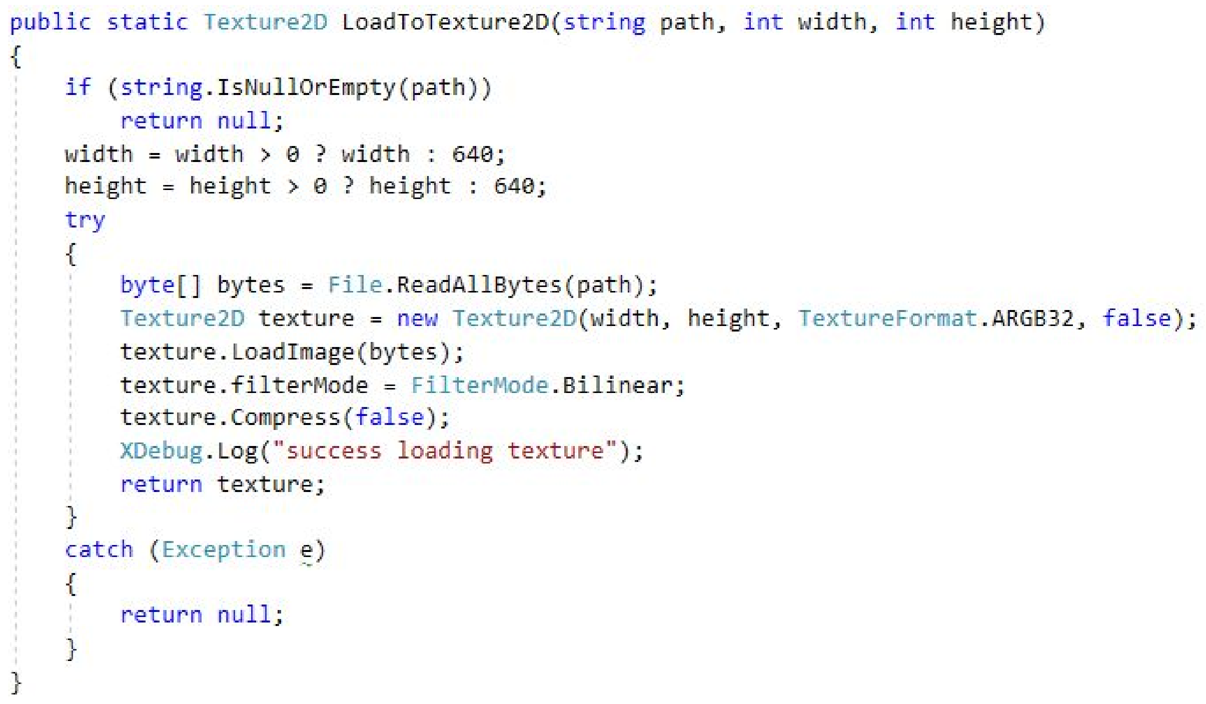
```
if (File.Exists(Application.persistentDataPath + "/" + id + ".gd"))
{
    BinaryFormatter bf = new BinaryFormatter();
    FileStream file = File.Open(Application.persistentDataPath + "/" + id + ".gd", FileMode.Open);
    Palace palace = (Palace)bf.Deserialize(file);
    file.Close();
    return palace;
}
```

Figure 38 - Code Snippet for Deserializing and Loading ModelData

3.9 Upload and Save Images

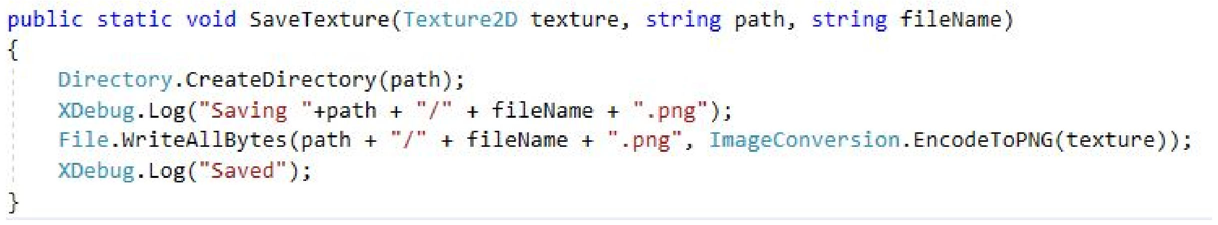
The upload and save of images are implemented with the api of Fantom Android Native Plugin. Fantom is a plugin similar to the Google Map plugin mentioned in section 3.3. When the user has chosen the image, Fantom api would return the path of the image. Then the file at the path returned is loaded into a byte array(Figure 39). The byte array is loaded in a **Texture2D** object. The texture is then applied onto the game objects in the scene as a material so as to be rendered.

To persistently saved the uploaded image, a new copy of the image is made instead of just storing the location path so as to avoid the user removing the image on the phone. The texture loaded would be encoded into a PNG file with the **ImageConversion** api of Unity (Figure 40) and saved into the local space of the application. The PNG file would be decoded back to texture when loaded.



```
public static Texture2D LoadToTexture2D(string path, int width, int height)
{
    if (string.IsNullOrEmpty(path))
        return null;
    width = width > 0 ? width : 640;
    height = height > 0 ? height : 640;
    try
    {
        byte[] bytes = File.ReadAllBytes(path);
        Texture2D texture = new Texture2D(width, height, TextureFormat.ARGB32, false);
        texture.LoadImage(bytes);
        texture.filterMode = FilterMode.Bilinear;
        texture.Compress(false);
        XDebug.Log("success loading texture");
        return texture;
    }
    catch (Exception e)
    {
        return null;
    }
}
```

Figure 39 - Code Snippet for Creating Texture from the Path of Image File



```
public static void SaveTexture(Texture2D texture, string path, string fileName)
{
    Directory.CreateDirectory(path);
    XDebug.Log("Saving "+path + "/" + fileName + ".png");
    File.WriteAllBytes(path + "/" + fileName + ".png", ImageConversion.EncodeToPNG(texture));
    XDebug.Log("Saved");
}
```

Figure 40 - Code Snippet for Persistently Saving the Uploaded Image

Section IV - Result and Deliverable

4.1 Overview

A fully functional application has been implemented and fully tested. Modifications and additions of new features have been made as the progress of the project proceed. The application run on Android phone and used together with Google Cardboard and uSens Fingo. This chapter gives a detailed review of the implemented application.

4.2 Main Menu

The application starts in non-VR mode. Main Menu is shown in the starting. It would enter VR mode when user start building memory palace.

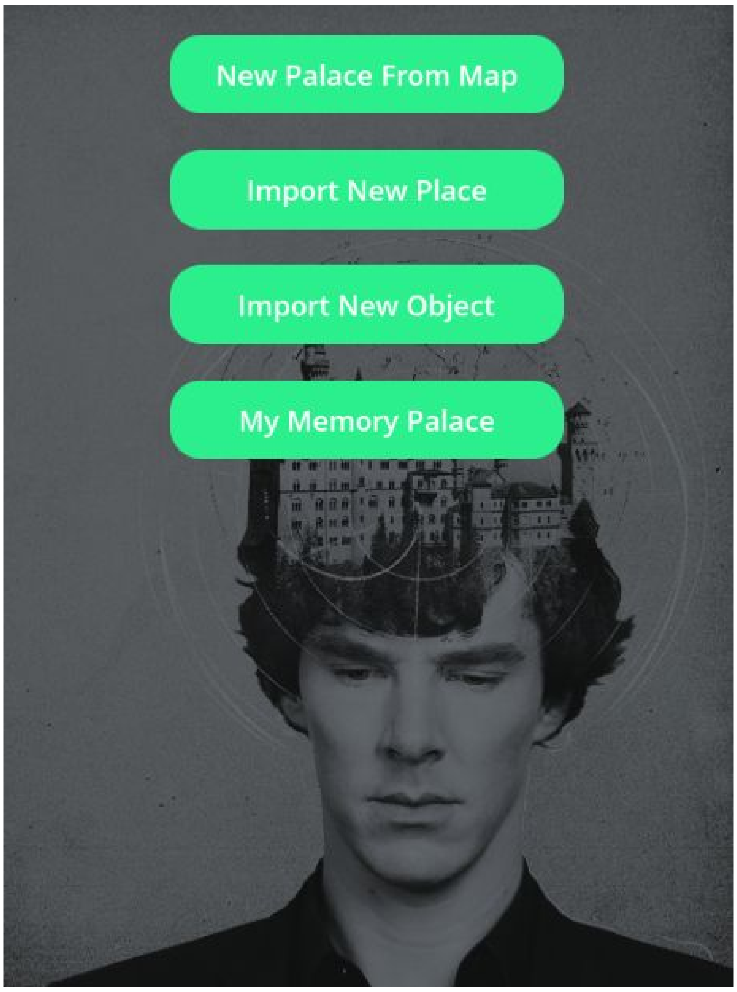
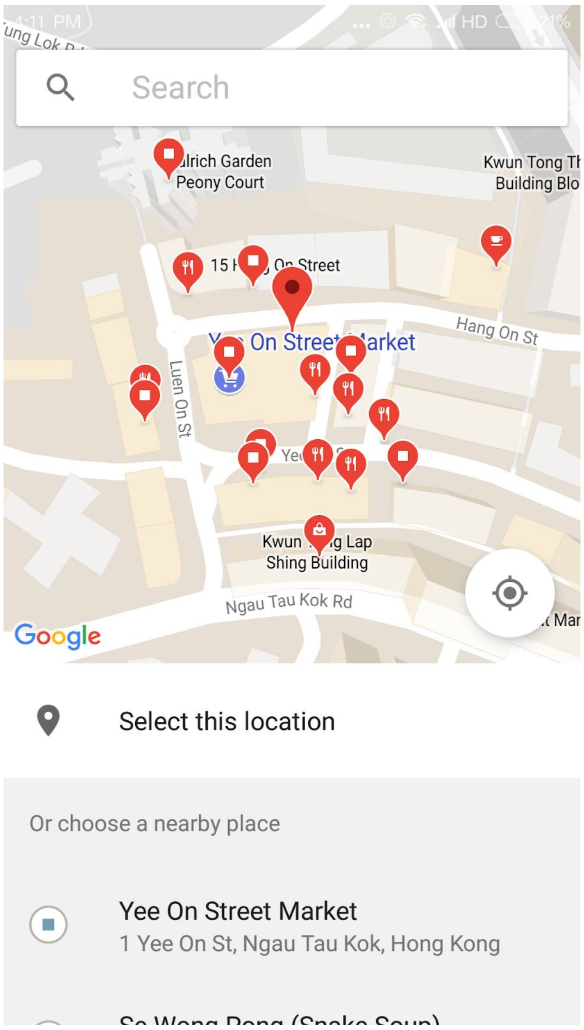


Figure 41 - Main Menu

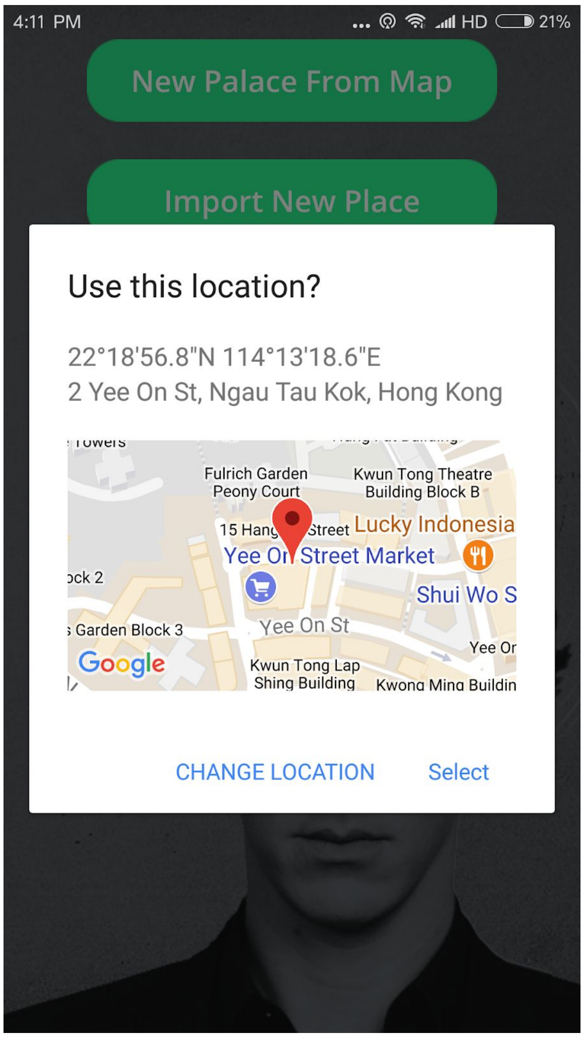
4.3 Place Selection

User can open the map to pick place by clicking “New Palace From Map” in main menu. User can drag the map to pick place (Figure 42, 43) or search with name (Figure 44). After the picking place, user is prompted to enter a name for the memory palace (Figure 45). After that, the application would be changed into VR mode and user should put on Google Cardboard.



Yee On Street Market
1 Yee On St, Ngau Tau Kok, Hong Kong

Figure 42 - Map



Use this location?

22°18'56.8"N 114°13'18.6"E
2 Yee On St, Ngau Tau Kok, Hong Kong

Figure 43 - Select Place

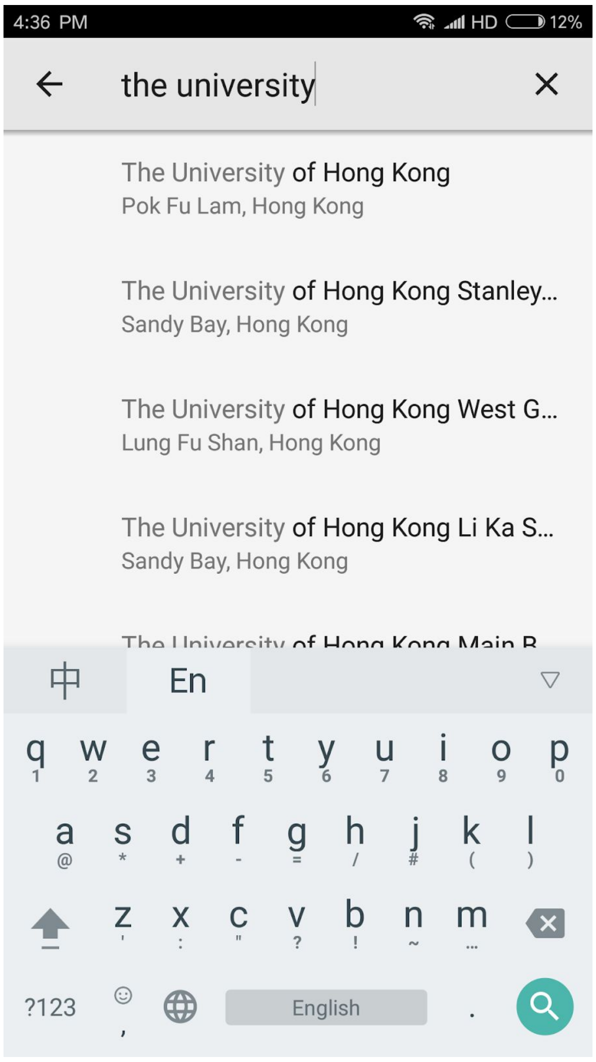

4:36 PM
HD 12%
the university
The University of Hong Kong
Pok Fu Lam, Hong Kong
The University of Hong Kong Stanley...
Sandy Bay, Hong Kong
The University of Hong Kong West G...
Lung Fu Shan, Hong Kong
The University of Hong Kong Li Ka S...
Sandy Bay, Hong Kong
English

Figure 44 - Search Place

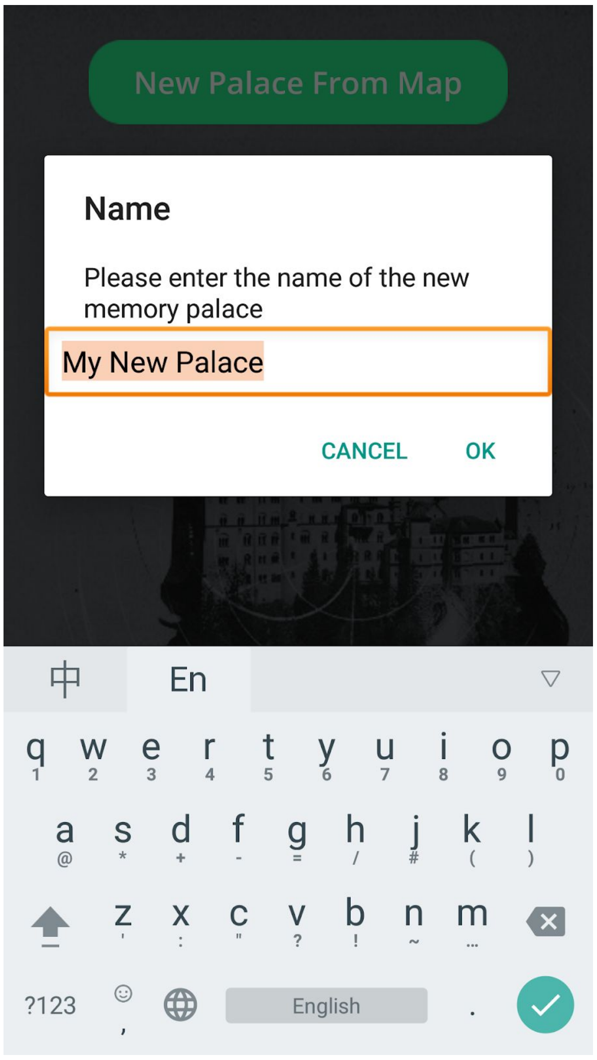


Figure 45 - Input Memory Palace Name

4.4 Upload place and object photo

By pressing “Import New Place” button on main menu, user would be able to upload panorama photo of place (Figure 46). Then user would be prompted to type name for memory palace.

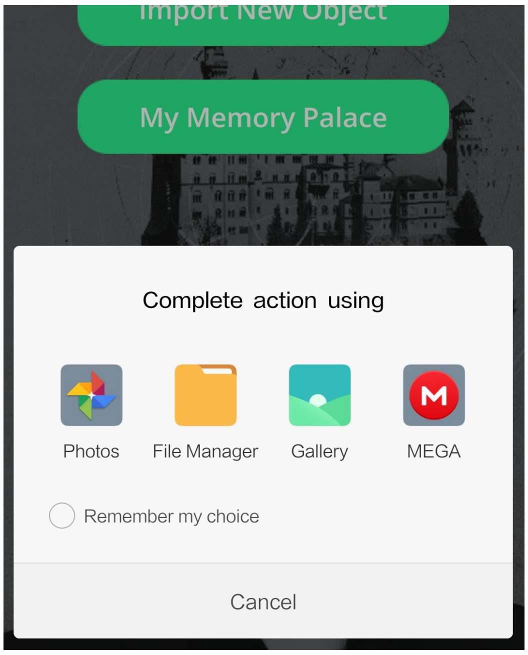


Figure 46 - Upload Place Panorama Photo

By pressing “Import New Object” button on main menu, user would be able to review the photos of objects imported into the application (Figure 47). User can upload new photo by pressing the upload button on top right corner and would be prompted to give a name after the upload (Figure 48). User can delete the photos by long pressing the photo and click the delete button (Figure 49).

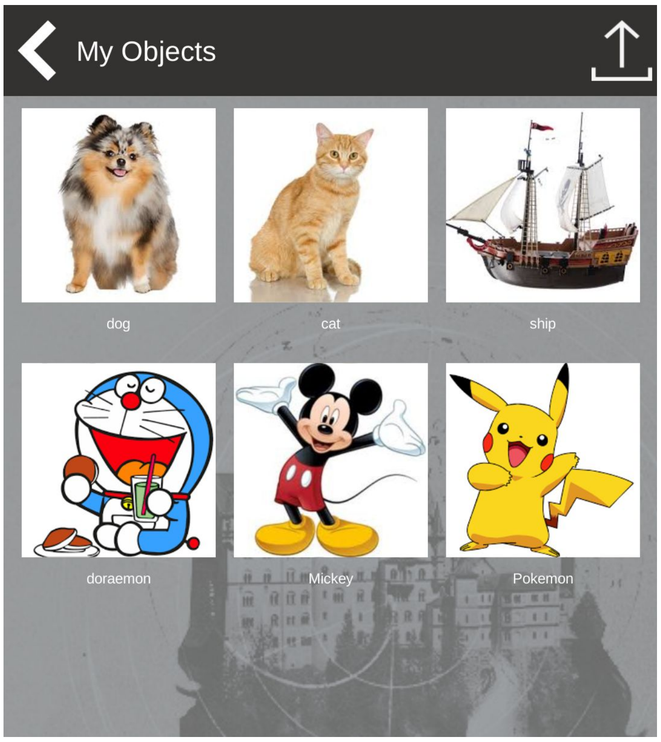


Figure 47 - Uploaded Photos

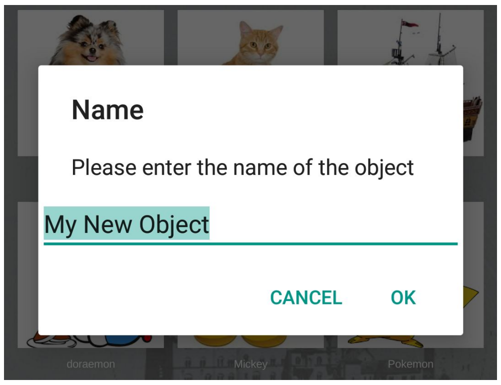


Figure 48 - Input Object Name

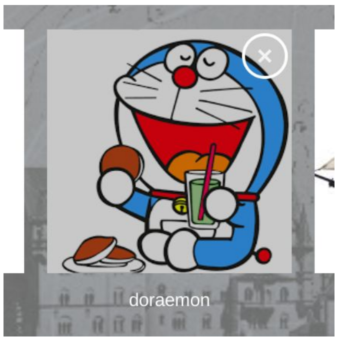


Figure 49 - Delete Object

4.5 Panorama View

The application would enter VR mode when using start editing memory palace. The image of the chosen location or the uploaded panorama photo would l be rendered as panorama environment (Figure 50). User should put on Google Cardboard and attached Fingo to phone.

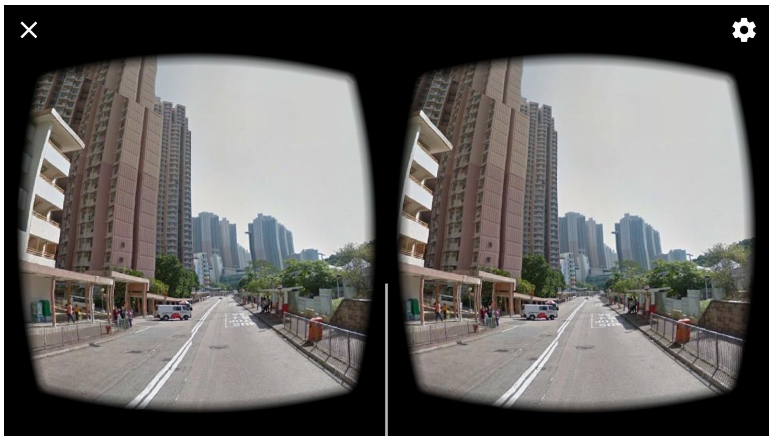


Figure 50 - Panorama View

4.6 3D Model Library

User can press the “Library” button in the menu to open 3D model library which is powered by Google Poly (Figure 51). User can click to select object and press “Import” button to import model (Figure 52). Pagination is also supported which user can click the 2 arrow on left and right side. User can press the “My Objects” button to import their self-uploaded objects (Figure 51).

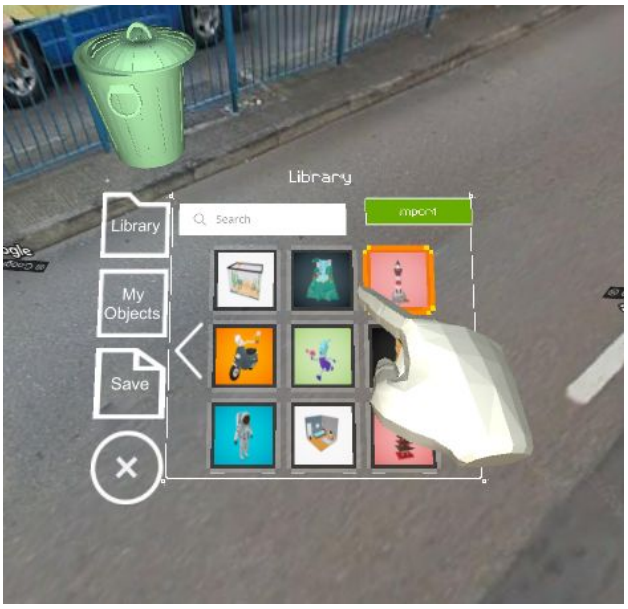


Figure 51 - Google Poly Library

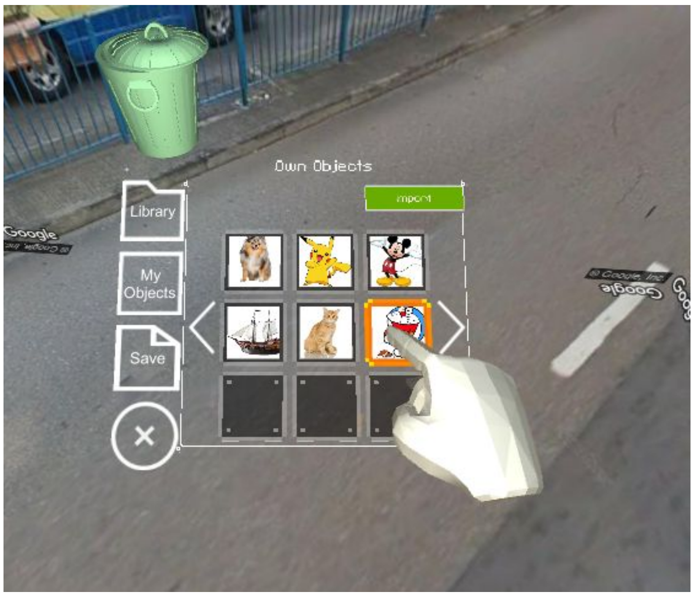


Figure 52 - Self-uploaded Object Library

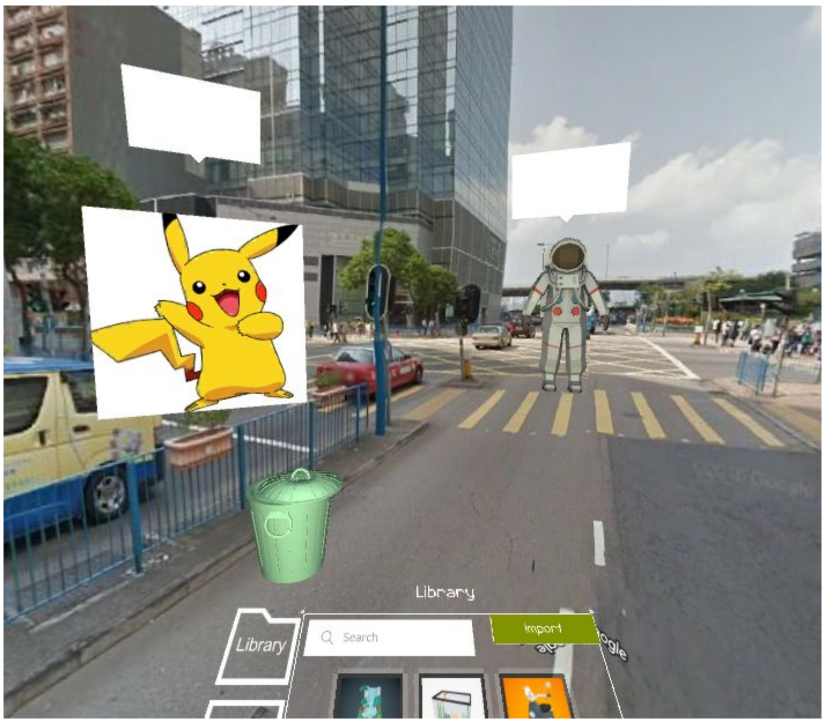


Figure 53 - Import Objects

4.7 Interaction with Objects

After the objects are imported, user can grab to rotate and place the objects into desirable location (Figure 54). The objects can be dragged to the trash bin to remove it from scene (Figure 55).

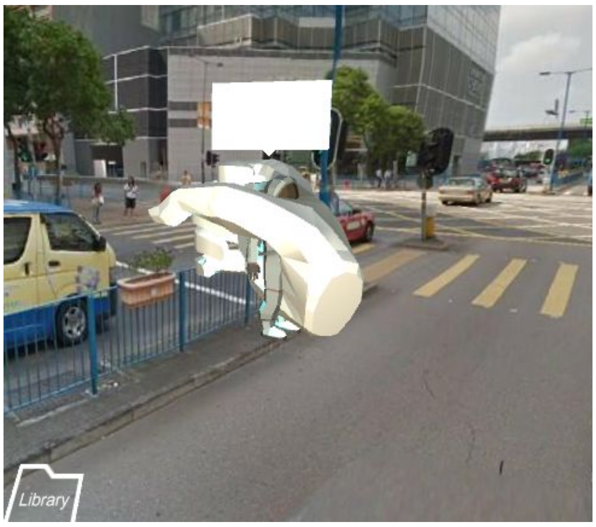


Figure 54 - Grabbing Object

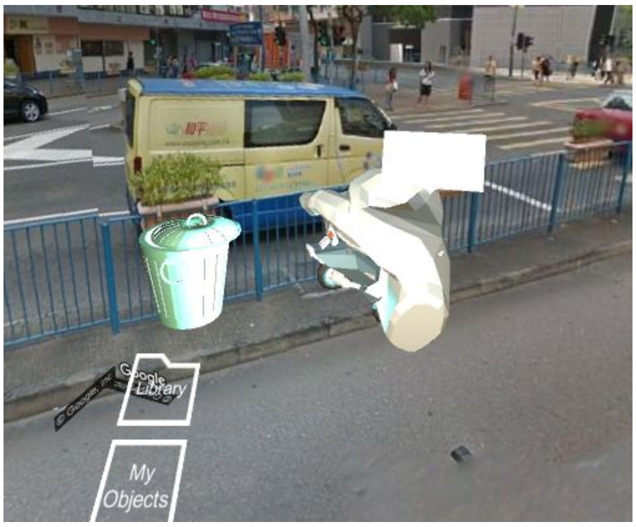


Figure 55 - Remove Object

4.8 Virtual Keyboard and Voice Input

There are 2 area that require user input: the searching in library and labelling 3D models. 2 input methods are supported: virtual keyboard and voice input. The detail of their implementation is mentioned in section 2.9.
User can press the search bar in the 3D model library with a point gesture (Figure 56) to type with virtual keyboard. When using virtual keyboard, user should use eye gaze to focus the keys and press the button on Google Cardboard (Figure 58) to type and search (Figure 59). User can press the search bar with a gun gesture (Figure 57) to search with voice input (Figure 60).

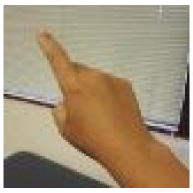


Figure 56- Point Gesture

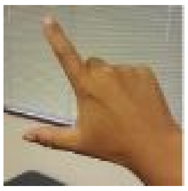


Figure 57- Gun Gesture

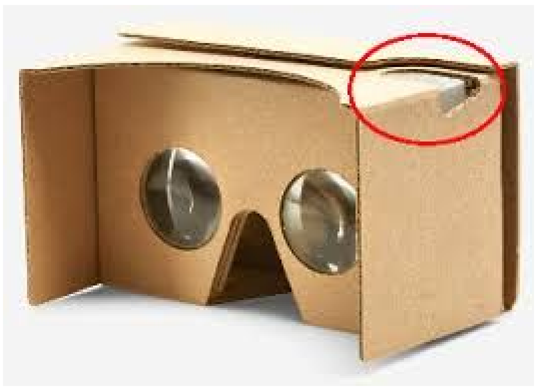


Figure 58 - Google Cardboard Button

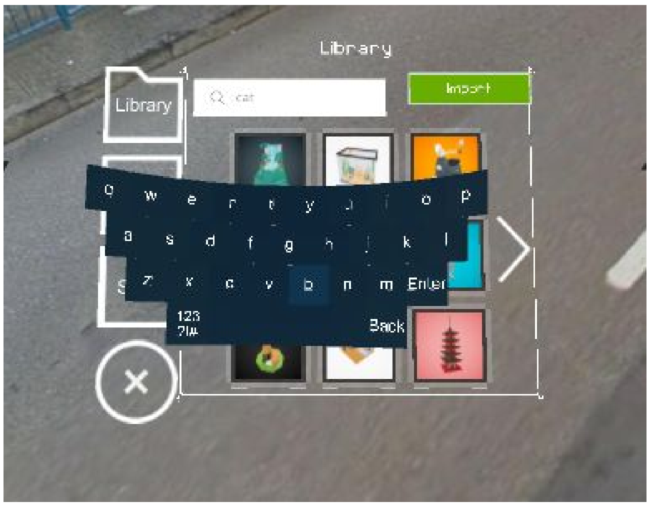


Figure 59 - Search with Virtual Keyboard

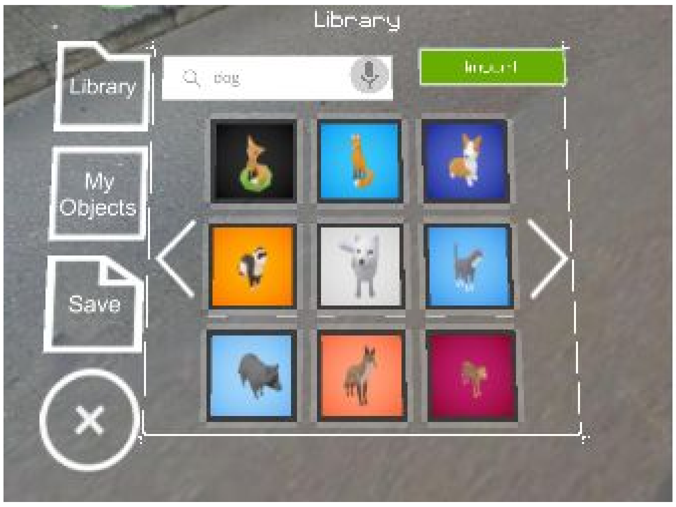


Figure 60 - Search with Voice Input

Typing label for objects works similar to typing searching keyboard. User can press the bubble on top of the object to input text (Figure 61).

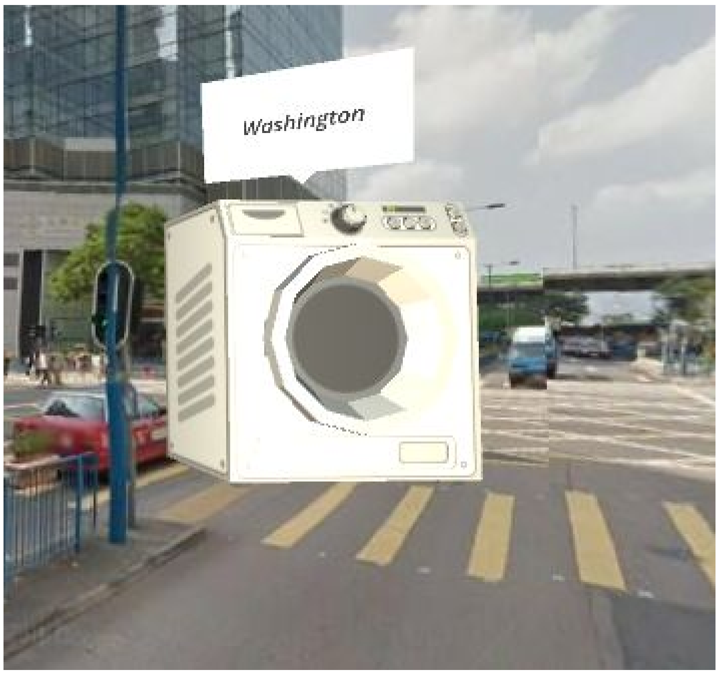


Figure 61- Labelling Object

4.9 Saving Data

User can click the “Save” button on user menu to save the created memory palace (Figure 62). The saved memory palace can be viewed and loaded by clicking “My Memory Palace” button in main menu (Figure 63).

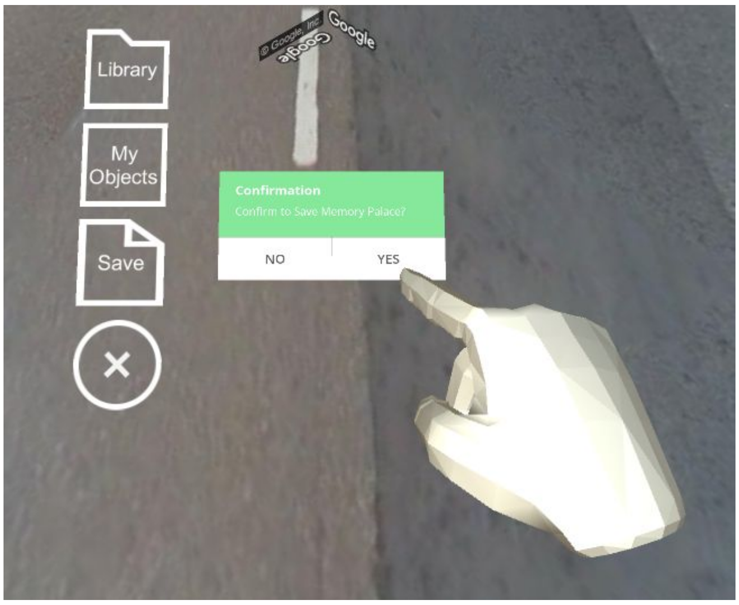


Figure 62 - Save Memory Palace

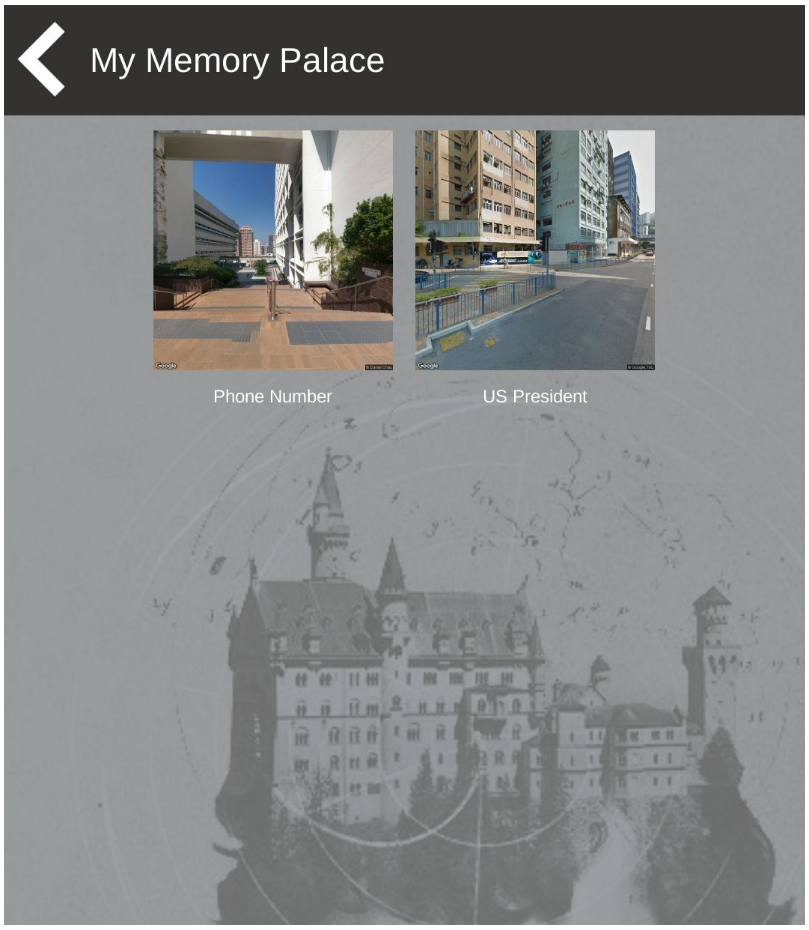


Figure 63 - View Saved Memory Palace

Section V - Future Works

5.1 Overview

At this stage, the application provide enough feature to reach the objective of this project, yet there are still some potential improvement can be made to provide extra features and better utilize the power of virtual reality so as to further enhance the effectiveness of memory palace technique.

5.2 Web Server and Database

As mentioned in section 2.11, the original plan is to implement an external database that the client would application request and store data via a web server. The advantage of using external database is to avoid lost of data due to accidentally delete or damage of device. It also make data portable across different devices. Currently the data is serialized into byte stream to be stored locally. To use external database, the storage form of data has to be changed. One possible way is to structure the data in form of JSON and store in an NoSQL database such as Cassandra and MongoDB.

Moreover, compared to loading data locally, there is overhead in network transfer. Optimization needs to be made such as minimizing data size and caching large data locally. Multi-thread is needed to handle the retrieving and storing of data to avoid blocking of user interface.

5.3 Sharing

An idea to enhance the feature of the application is to implement sharing between different users. A online community can be made that user can create personal account and share their memory palace either publicly or with their friends. This feature has huge potential that it can make the application very suitable for educational use. For example, memory class tutor can use the application to make teaching material and share them with students. Students can submit homework via the system. Normal school teachers can also use it to help daily teaching, for example school can use it to help students memorizing English vocabulary.

5.4 User Movement in VR

The potential of virtual reality is huge. This project has just implemented the basic immersive experience and user interaction. More enhancement can be made to further utilize the power of VR. One of the enhancement is let the user move around in the place and the street view would be changed accordingly.

The current implementation is using let the user stay in the middle of the place statically to build memory palace. Yet, it is very common for people to build a memory palace along a route from place A to place B. This is especially useful for memorizing data with order and in sequence.

5.5 Sound for Objects

According to research, it is found that strengthen of memory is closely related to strength of sense [5]. People usually just rely on looking to memorize information, yet other sense like hearing and touching is also very important to give more impression and stimulation to the brain in order to have strong memory. Human can easily remember things happened in daily life because we memorize it with the help of all 5 sense. Yet struggle to memorize knowledge on book because only eye is utilized.

Hearing sense is a potential feature that can be added to the application. One idea is to let user attach sound to the objects. The sound would be trigger when the object is touched.

Section VI - Working Schedule

Time	Task	Status
September	Research on topics and technology	Completed
	Project plan and website	
October	Study of Unity and other SDKs	
November	Implementation with Fingo	
December	Implementation of panorama view	
	Implementation of 3D models library	
January	Continuation on previous implementation	
	First presentation and intermediate report	
	Implementation of place selection	
February	Implementation of 3D model library	
	Implementation of user interface	
March	Implementation of optional feature	
	Finalize the implementation	
April	Complete testing	
	Final report	
	Final Presentation	In Progress
	Project Poster	
May	Project Exhibition	Not Started

Section VII - Conclusion

This report has described the background, motivation, objective, design, methodology and the complete review of the VR memory palace application. The application is aimed to help people who use Memory Palace system to actualize their memory palace. VR can solve the problem that Memory Palace system has as well as enhancing its effectiveness. This project is desirable because it demonstrates how VR can be used to solve problems but not just of fun and fanciness.

This application allows user to choose places and 3D objects to build their memory palace. The application runs on Android phone and is used together with Google Cardboard. uSens Fingo is used for hand motion tracking. The application is developed with Unity as the platform and some external SDKs are also used in development.

The application provides a map for users to choose their desired place to build memory palace. The chosen place would be rendered as a panorama environment. User can also upload their own panorama photos. The application also provide a rich 3D models library powered by Google Poly, as well as allowing users to upload their own photos. The imported object are interactable with Fingo hand. Virtual keyboard and voice input are supported to type texts in the VR mode.

At this stage, the application provides enough features to serve the objective of this project. Yet, some future works can be made to implement new features and better leverage the power of VR.

References

[1] Yael Shrager, Peter J. Bayley, Bruno Bontempi, Ramona O. Hopkins and Larry R. Squire, "Spatial memory and the human hippocampus", Proc Natl Acad Sci USA, Washington, DC, PMC1815289, Feb.2007.

[2] Anthony Metivier. (2017, Sep 22). Memory Palace Science: Proof That This Memory Technique Works[Online]. Available: https://www.magneticmemorymethod.com/memory-palace/ [Accessed: 2017, Oct 20]

[3] Ron White, Sherlock Holmes Mind Palace Secrets Revealed, 2010

[4] Henry Johnson. (2016, May 15). Virtual Reality in 2016: Its Power and Limitations. Available: https://medium.com/@_43614/virtual-reality-in-2016-its-power-and-limitations-ae214c894888 [Accessed: 2017, Oct 22]

[5] Michael Hopkin. (2004, May 31). Link proved between senses and memory. Available: https://www.nature.com/news/2004/040531/full/news040524-12.html [Accessed: 2018, March 25]

Appendices

1. Memory Techniques Wiki, Page. 7
   https://artofmemory.com/wiki/Main_Page
2. uSens Fingo, Page 11
   https://www.usens.com/fingo
3. Unity 3D, Page 11
   https://unity3d.com/
4. Google VR, Page 11
   https://vr.google.com/
5. Google Place API, Page 12
   https://developers.google.com/places/
6. Fantom Android Native Plugin, Page 12
   https://assetstore.unity.com/packages/tools/gui/android-native-dialogs-and-functions-plugin-106497
7. Google Street View API, Page 13
   https://developers.google.com/maps/documentation/streetview/
8. Google Poly, Page 14
   https://developers.google.com/poly/
9. UI-Builder
   https://assetstore.unity.com/packages/tools/gui/ui-builder-29757
10. Inventory Engine
    https://assetstore.unity.com/packages/tools/gui/inventory-engine-95550