TECHNICAL MANUAL FOR THE “REINHERIT A MUSEUM” AUGMENTED REALITY MOBILE APPLICATION

1 Summary

This document presents a short description for the setting up and usage of the source code developed to support the immersive performance for the ReInHerit project that took place at the Bank of Cyprus Cultural Foundation on the 19th of May 2023. The purpose of the document is to facilitate users who wish to recreate a similar experience to their own premises. The code for the application can be accessed from the following GitHub repository link: https://github.com/CYENS/ReInHeritArApp.

2 Contents

1 Introduction

“ReInHerit a Musuem” was a performance that took place on May 19th 2023, at the Bank of Cyprus Cultural Foundation in Nicosia, Cyprus. The performance included live instrumental musician, projection mapping with generative visuals, and an Augmented Reality application that visitors used to view a virtual exhibitions with artefacts that are not in the permanent public exhibition of the Foundation.

The Augmented Reality application was developed using the Unity Game Engine and the Lightship Augmented Reality Developer Kit (ARDK) and deployed to the Google Play store and Apple App store through which visitors could download it. The application provided information for 12 artefacts from the private collection of the Foundation, which were also 3D-scanned and presented as interactable 3D virtual models. Before the performance, and within the exhibition areas another 12 phones each associated with one of the artefacts were placed at certain locations along with a paired Bluetooth speaker. When visitors viewed and interacted with the virtual artefacts, these interactions were tracked and stored at a cloud-based database. The additional phones that were placed within exhibition space, run an application (base station) which listened to the events on the database, and whenever a new interaction was observed, it triggered an event which played music specifically composed for the event. Each of the base stations included six music tracks, that each started playing in order after an interaction was triggered. The tracks did not play in a loop.

2 Guidelines for reuse

These are the proposed steps to be followed when setting up a system for such a performance:

- The mobile phones that contain the Base Station application should be hidden so that no one can interfere with them. Also, visitors should not be able to reach them or see them so that the immersion of the project is not broken. Also, if it is possible, the devices should include locking mechanisms to avoid theft.
- The Bluetooth speakers should also be hidden from the eyes of the visitors and placed near points of interest. This will make the exhibition more interactive and engaging.
- When every mobile phone and Bluetooth speaker is set up, the Base Station application should be set up. First, the mobile phone must be connected to the internet in order to be able to connect to the project’s database and receive or send data to it. Then, inside the application, the number of the exhibit must be configured. Once this is done, the application will receive updates once certain events occur during the performance.
- If a user views or interacts with an exhibit, the project’s database records the data for further analysis. The interaction event for each exhibit is also sent to the corresponding Base Station application. When this happens and for every interaction, a specific sound starts playing that is previously loaded in the application.

It is important to mention that data collected during the event do not contain any sensitive private information about the visitors or the performers. The application tracks the following data:

1. Interactions. When the user interacts with an exhibit, the action is recorded along with the time it happened.
2. Views. When a user starts or stops viewing an exhibit the application records both events and creates two timestamps. This is to determine the duration that the user was viewing each exhibit.

More details regarding the specific app will be provided in dedicated sections of the ReInHerit digital hub with manuals, instructions, guidelines and links to the source code for the recreation of the experience, in a more “personalized” context for other potential cultural heritage sites across Europe.

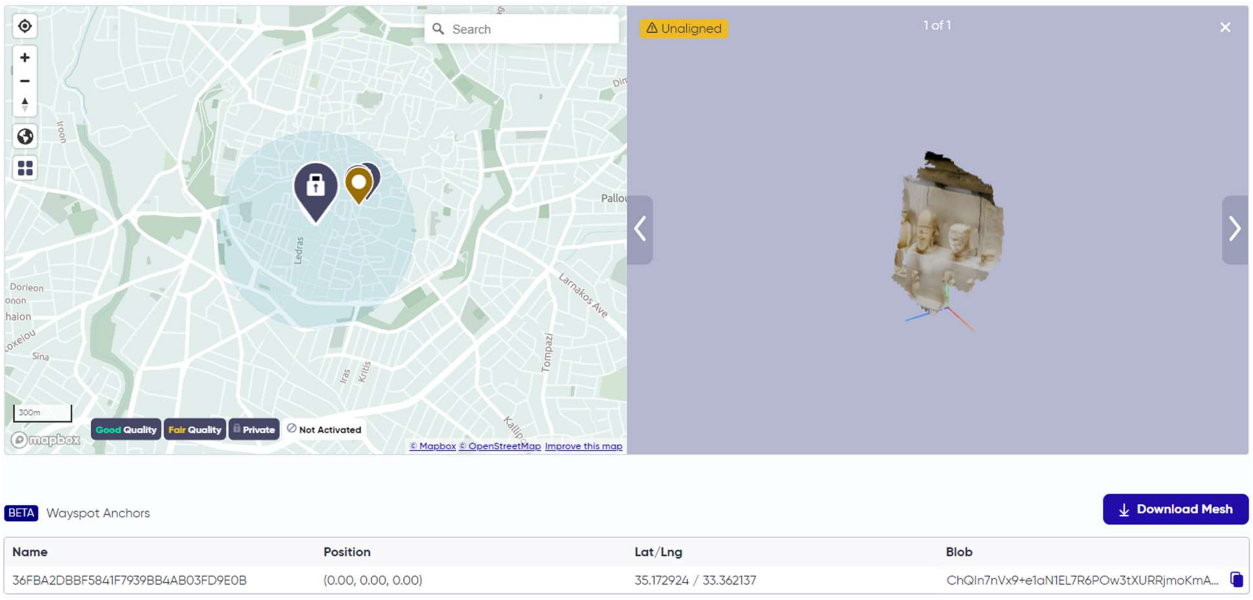
3 Technical Guidelines

“ReInHerit a Museum” is a location-based augmented reality application for mobile devices (Android and iOS) that assists with the promotion of cultural heritage sites, by offering an interactive experience to guests.

3.1 Lightship Scans

The application uses the Lightship SDK for Unity in order to recognize specific areas and project the artefacts. The areas used to create the AR experience must be scanned first.

To scan those areas, we need to use the Wayfarer application which is currently available only on Android mobile phones (https://wayfarer.nianticlabs.com/). Through it, the user can use the mobile phone’s camera and record videos of the area that is going to be used in the AR experience. Then the video is uploaded on the Niantic servers and processed to create a mesh that will later be used to place the AR objects in the scene. Depending on the traffic on the servers, this process can last from 1 to 5 days. Once the mesh is created, it can be downloaded and imported to Unity. Afterwards anchors can be placed around this mesh which represents the AR object that will appear on the application. Lastly, all the anchors along with their location data and the AR object are stored in a Manifest file that Lightship SDK is using. Each area that is going to be used corresponds to one Manifest file. All these files are then loaded into the Location Manifest Manager. By calling the LoadWayspotAnchors function, the application can load all the AR objects for that specific area and are able to be viewed through the mobile phone’s camera.



Name	Position	Lat/Lng	Blob
36FBA2DBBF5841F7939BB4AB03FD9E0B	(0.00, 0.00, 0.00)	35.172924 / 33.362137	ChQIn7nVx9+e1aN1EL7R6POw3tXURRjmoKmA...

Figure 1. Wayfarer application.

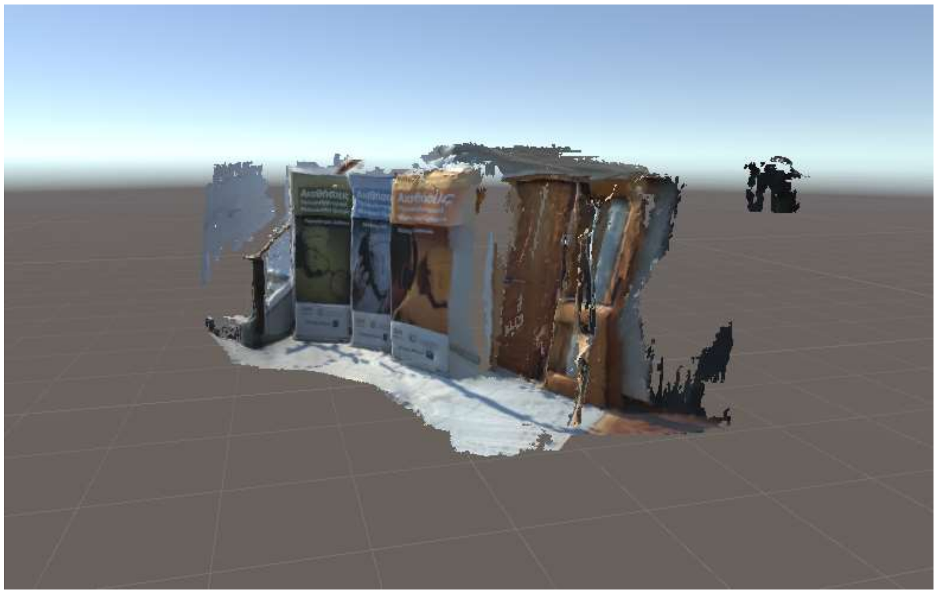


Figure 2. Mesh inside Unity.

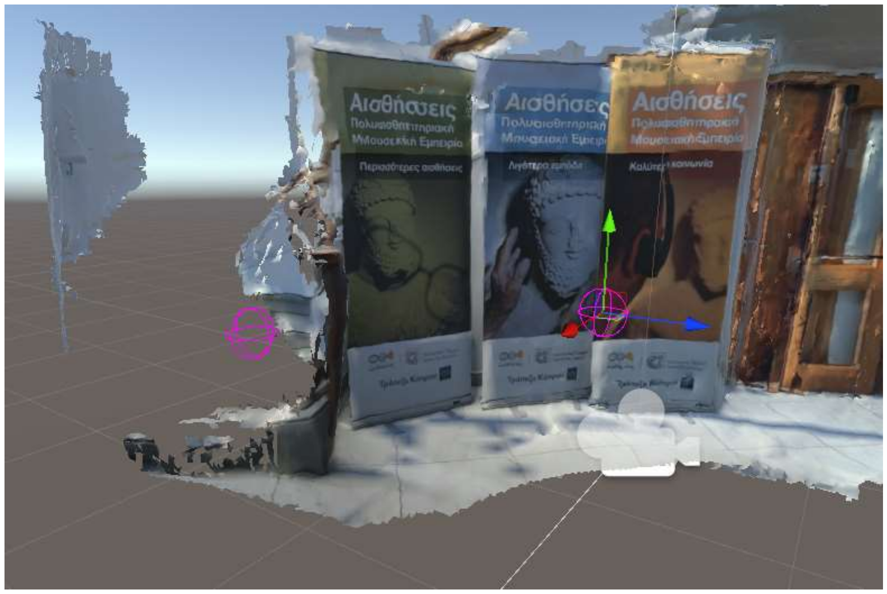
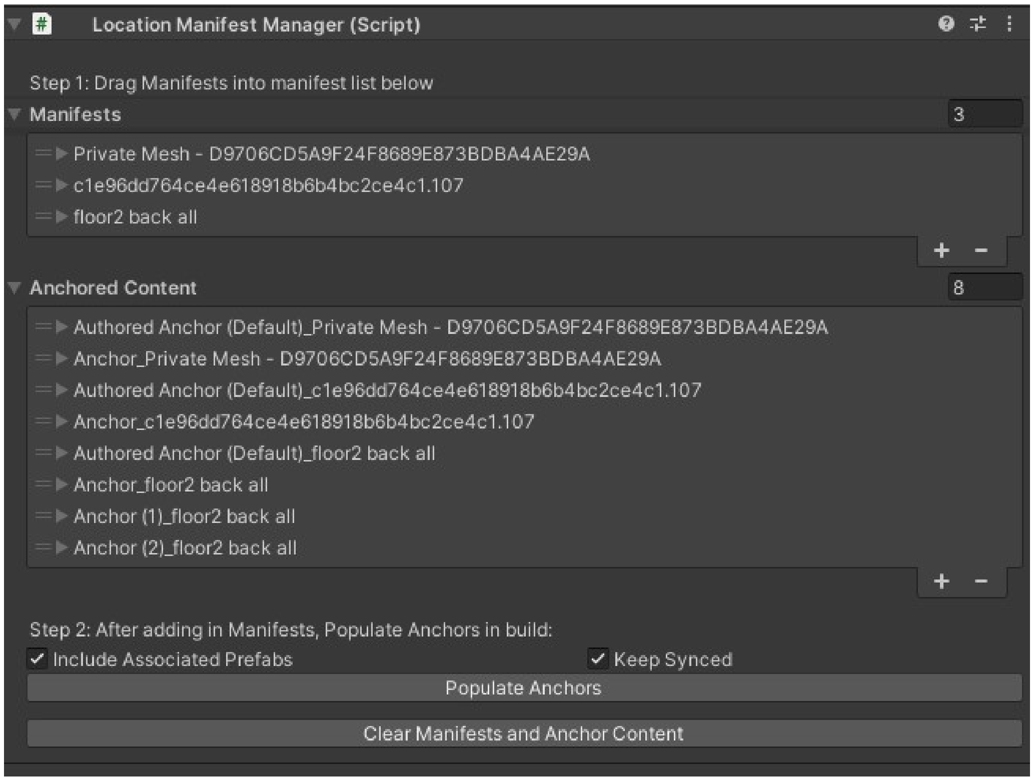

Αισθήσεις
Πολυαισθητηριακή Μουσειακή Εμπειρία
Περισσότερες αισθήσεις
Αισθήσεις
Πολυαισθητηριακή Μουσειακή Εμπειρία
Λιγότερα εμπόδια
Αισθήσεις
Πολυαισθητηριακή Μουσειακή Εμπειρία
Καλύτερη κοινωνία

Figure 3. The anchor is placed manually around the mesh.


Location Manifest Manager (Script)

Step 1: Drag Manifests into manifest list below

Manifests 3

- Private Mesh - D9706CD5A9F24F8689E873BDBA4AE29A
- c1e96dd764ce4e618918b6b4bc2ce4c1.107
- floor2 back all

Anchored Content 8

- Authored Anchor (Default)_Private Mesh - D9706CD5A9F24F8689E873BDBA4AE29A
- Anchor_Private Mesh - D9706CD5A9F24F8689E873BDBA4AE29A
- Authored Anchor (Default)_c1e96dd764ce4e618918b6b4bc2ce4c1.107
- Anchor_c1e96dd764ce4e618918b6b4bc2ce4c1.107
- Authored Anchor (Default)_floor2 back all
- Anchor_floor2 back all
- Anchor (1)_floor2 back all
- Anchor (2)_floor2 back all

Step 2: After adding in Manifests, Populate Anchors in build:

✓ Include Associated Prefabs ✓ Keep Synced

Populate Anchors

Clear Manifests and Anchor Content

Figure 4. The Location Manifest Manager holds the Manifests and their corresponding anchors.

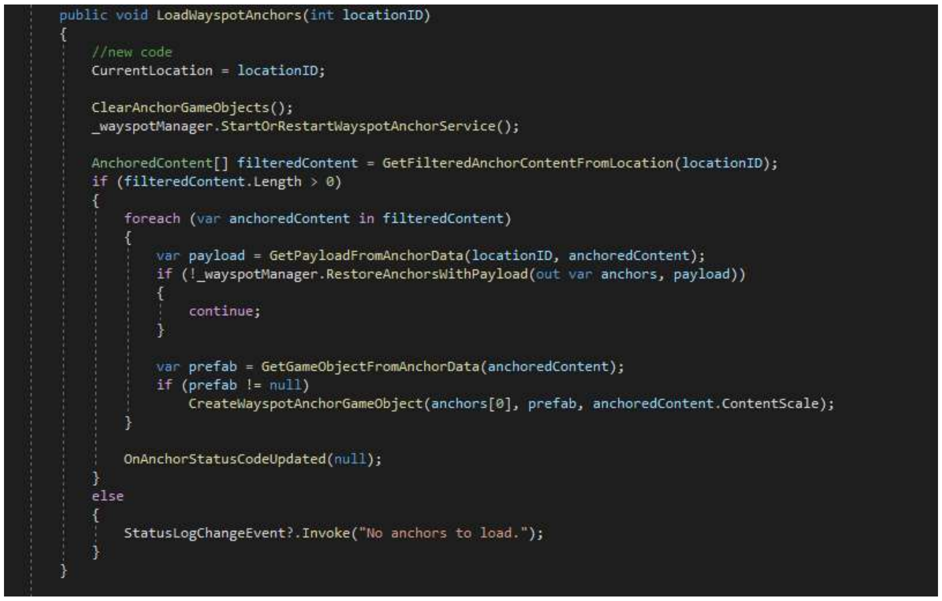

```
public void LoadWayspotAnchors(int locationID)
{
    //new code
    CurrentLocation = locationID;

    ClearAnchorGameObjects();
    _wayspotManager.StartOrRestartWayspotAnchorService();

    AnchoredContent[] filteredContent = GetFilteredAnchorContentFromLocation(locationID);
    if (filteredContent.Length > 0)
    {
        foreach (var anchoredContent in filteredContent)
        {
            var payload = GetPayloadFromAnchorData(locationID, anchoredContent);
            if (!_wayspotManager.RestoreAnchorsWithPayload(out var anchors, payload))
            {
                continue;
            }

            var prefab = GetGameObjectFromAnchorData(anchoredContent);
            if (prefab != null)
                CreateWayspotAnchorGameObject(anchors[0], prefab, anchoredContent.ContentScale);
        }

        OnAnchorStatusCodeUpdated(null);
    }
    else
    {
        StatusLogChangeEvent?.Invoke("No anchors to load.");
    }
}
```

Figure 5. LoadWayspotAnchors function.

3.2 Loading Screen

The application's loading screen can be used to customize the application's appearance and helps initialize it. This screen contains two sections, the logo area and the partners area. There are two images which can be changed according to each event.

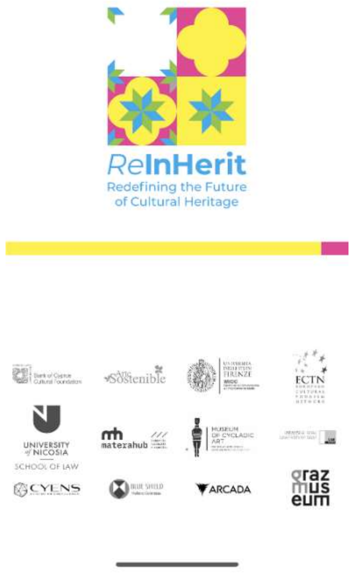
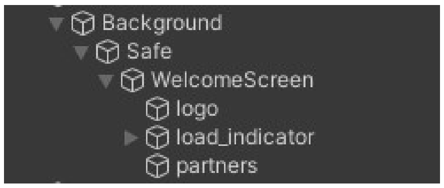

ReInHerit
Redefining the Future of Cultural Heritage
UNIVERSITY of NICOSIA SCHOOL OF LAW
CYENS
ARCADA
graz mus eum
ECTN

Figure 6. Loading Screen with the ReInHerit logo.

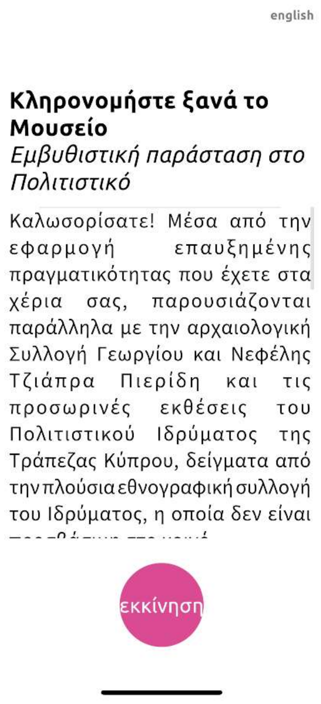


```
▼ Background
  ▼ Safe
    ▼ WelcomeScreen
        logo
      ▶ load_indicator
        partners
```

Figure 7. The logo and the partners images can be switched in Unity.

3.3 UI and Navigating through the Application

In this section we go through the screens that appear when the user starts the application. The first screen is an introduction to the event/exhibition. It contains a simple text with scrollbar. Throughout the application, every text is handled by the Localization package from Unity which is used to configure localization settings and adds support for multiple languages (for this event English and Greek). Each text that needs to appear in several languages needs to have the “Localize String Event” component. At the top of the screen there is also a small discreet button that switches between English and Greek. Lastly, when the user presses the circular pink button at the bottom of the screen, the main application menu loads.



english

Κληρονομήστε ξανά το Μουσείο
Εμβυθιστική παράσταση στο Πολιτιστικό

Καλωσορίσατε! Μέσα από την εφαρμογή επαυξημένης πραγματικότητας που έχετε στα χέρια σας, παρουσιάζονται παράλληλα με την αρχαιολογική Συλλογή Γεωργίου και Νεφέλης Τζιάπρα Πιερίδη και τις προσωρινές εκθέσεις του Πολιτιστικού Ιδρύματος της Τράπεζας Κύπρου, δείγματα από την πλούσια εθνογραφική συλλογή του Ιδρύματος, η οποία δεν είναι

Figure 8. Introduction screen with scrollable text. On the top right the user can switch between English and Greek.

The second screen contains a list of all the 3D objects that were scanned from the museum and are used in the AR application. For the ReInHerit event, 12 objects were used and could be displayed. By clicking on one of the objects, a new window appears which shows a preview of the object along with a small description about it. Also, as indicated by a small icon next to object, the user can rotate the object and view it from different angles. Finally, when the user is ready to start exploring, the top right yellow button with the camera icon leads to the area selection screen. That button appears only when the user is close to the location of the event.

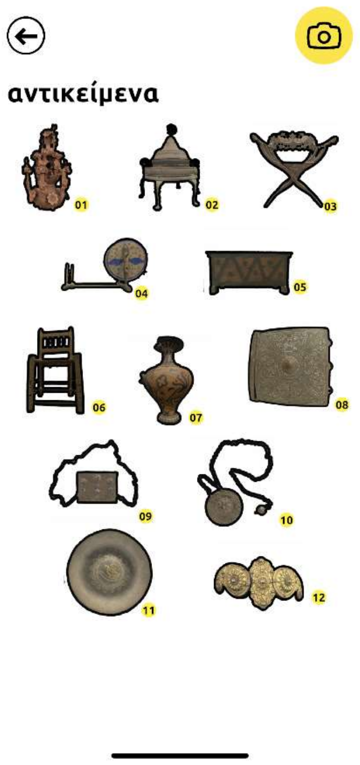
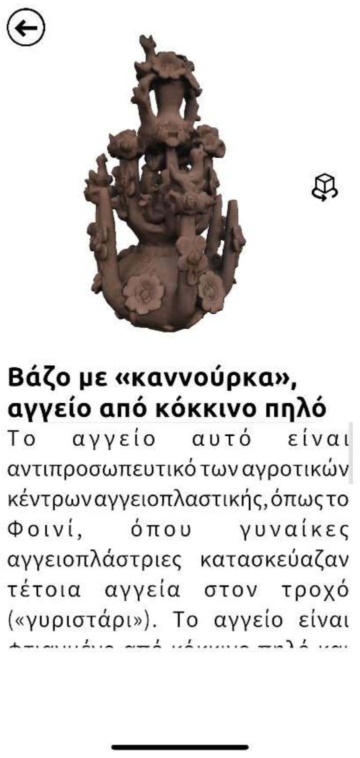


Figure 9. Screen that lists all the objects available in the application.



Βάζο με «καννούρκα», αγγείο από κόκκινο πηλό

Το αγγείο αυτό είναι αντιπροσωπευτικό των αγροτικών κέντρων αγγειοπλαστικής, όπως το Φοινί, όπου γυναίκες αγγειοπλάστριες κατασκεύαζαν τέτοια αγγεία στον τροχό («γυριστάρι»). Το αγγείο είναι

Figure 10. Preview of the object along with a brief description.

3.4 GPS

The application should be working only in the area where the event should take place. This is why before the user can check the areas of the exhibits, the application uses the mobile phone’s GPS to check its location. Then the application compares the latitude and longitude of the current location and the location of the event which is stored in the application. The distance of the two locations is then compared with a minimum threshold that can be set beforehand. The GPS should be activated once the application starts (by using Unity’s function Input.location.Start) just to get the initial location of the user and then be deactivated (Input.location.Stop) to save battery.

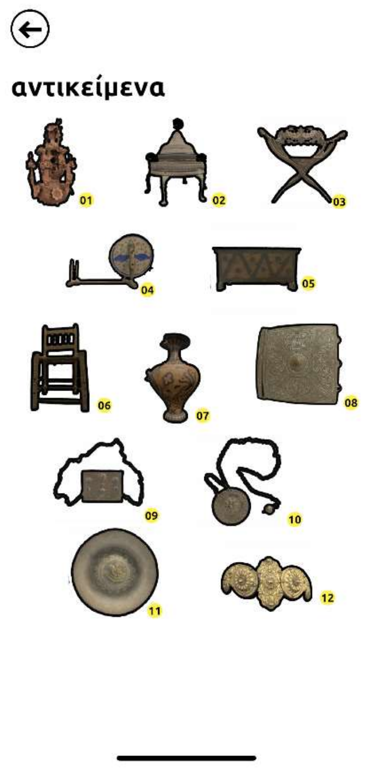
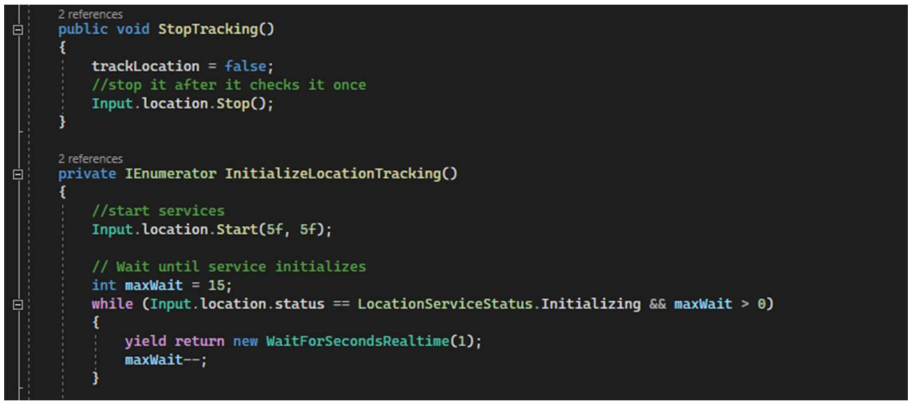


Figure 11. If the application is out of range the yellow button with camera icon disappears.



```
2 references
public void StopTracking()
{
    trackLocation = false;
    //stop it after it checks it once
    Input.location.Stop();
}

2 references
private IEnumerator InitializeLocationTracking()
{
    //start services
    Input.location.Start(5f, 5f);

    // Wait until service initializes
    int maxWait = 15;
    while (Input.location.status == LocationServiceStatus.Initializing && maxWait > 0)
    {
        yield return new WaitForSecondsRealtime(1);
        maxWait--;
    }
```

Figure 12. Code sample that starts and stops the GPS function.

The third screen of the application contains the list of all the areas that are available to be scanned. When the user touches one thumbnail, a new smaller window pops up with a wider image of the area. The user should stand roughly in the same position as shown in the image and then touch again the camera button at the bottom of the window to scan the area and begin the search for the AR Objects.

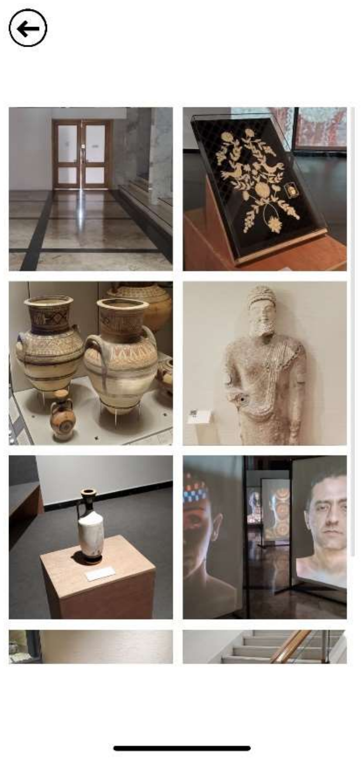


Figure 13. List of the areas that are available to be scanned.

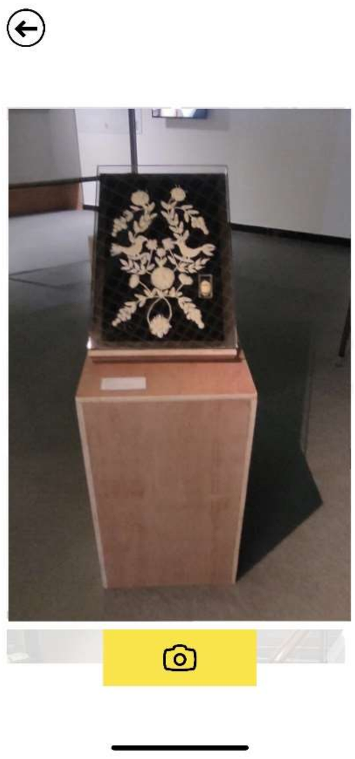


Figure 14. Larger image of the selected area.

3.5 Exhibition Locations Manager

The Exhibit Locations Manager holds all the info about the locations that are going to be used for the AR application. Every new location needs to be added to the list along with the following data:

- Name: The name of the area. It is used to distinguish between the areas that are in the application.
- Index in Manifest List: The index of the Manifest that is going to be selected from the Manifests list.
- Index in UI: The index of location in the Area selection screen.
- Number of Anchors interacted with: The number of anchors (AR Objects) that the user has interacted with so far. At the start of the application this variable is 0.
- Number of Anchors at location: The anchors (AR Objects) that are available to be viewed at that location.
- Thumbnail: The image that is used is the Area selection screen.
- Sprite: The image that is used when an Area is selected in the Area selection screen. It helps the user to find the correct location to stand.
- Temporary Overlay: An outline image of the area that helps the user to hold the mobile phone’s camera in the correct angle.
- Anchors: These are all the prefabs that are spawned in the area and are responsible for the AR objects that will be shown.

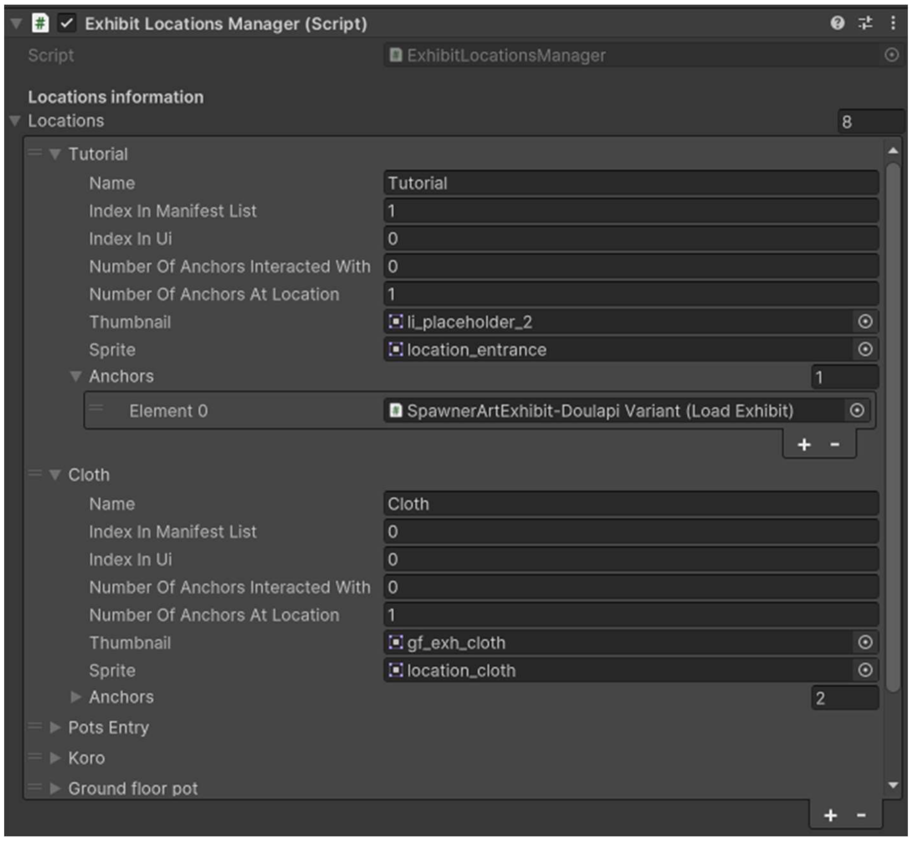


Exhibit Locations Manager (Script)

Script	ExhibitLocationsManager

Locations information

Locations 8

Tutorial

Name	Tutorial
Index In Manifest List	1
Index In Ui	0
Number Of Anchors Interacted With	0
Number Of Anchors At Location	1
Thumbnail	li_placeholder_2
Sprite	location_entrance
Anchors	1
Element 0	SpawnerArtExhibit-Doulapi Variant (Load Exhibit)

Cloth

Name	Cloth
Index In Manifest List	0
Index In Ui	0
Number Of Anchors Interacted With	0
Number Of Anchors At Location	1
Thumbnail	gf_exh_cloth
Sprite	location_cloth
Anchors	2

Pots Entry

Koro

Ground floor pot

Figure 15. The Exhibition Locations Manager script. It shows two locations named Tutorial (debug area) and the Cloth area.

3.5.1 Outline Overlays

To make it easier for the user to position better in the museum and view the exhibitions, some area overlays were created. When the user first scans the area with the camera, an overlay of it appears on the screen. By matching the outlines of the overlay to the scanned area, we help the user to align correctly with the exhibits and interact with them. To create these overlays, we took pictures of these areas beforehand under ideal lighting conditions. Then Photoshop was used to isolate the most significant features of the images and create their outlines. It is also important to mention that these images are also the ones that are used in the area selection menu which also helps the users to orient themselves in the museum.

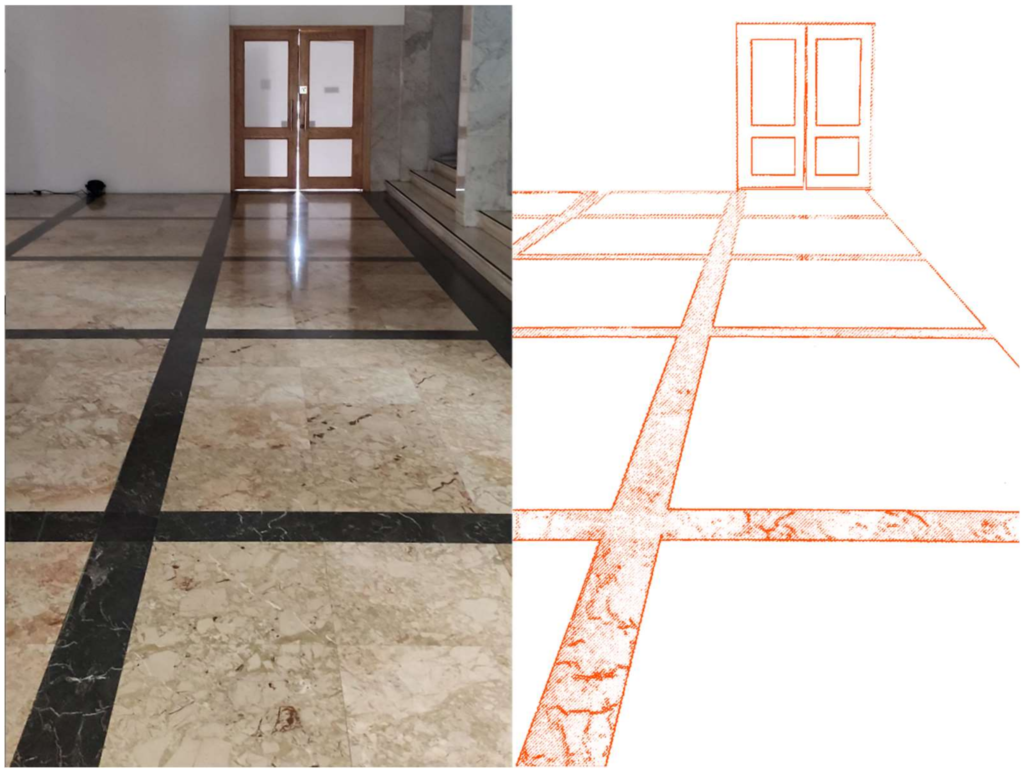


Figure 16. Image of the area that can be scanned (left) and the overlay of it that was created in Photoshop (right).

3.6 Exhibit Models Manager

The Exhibit Models Manager is responsible for spawning the correct models from a list of prefabs. For these prefabs we use the Addressable Asset System from Unity. It allows the developers to ask for an asset via its address. The asset can reside locally or remotely and then can be loaded whenever it is needed. For the ReInHerit exhibition, the assets are hosted remotely on a server. That means that any changes in the models can be made without the need to rebuild the application or the user to download the application again.

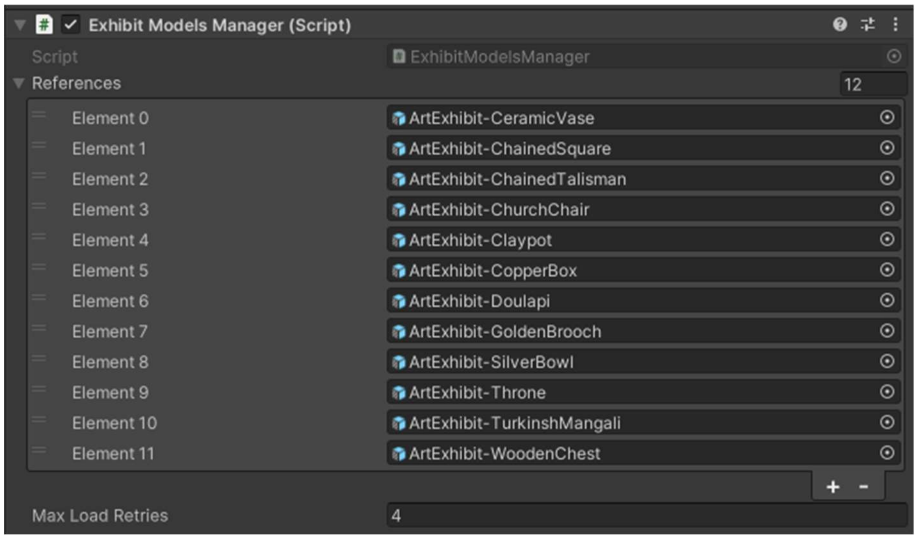


Exhibit Models Manager (Script)

Script	ExhibitModelsManager
References	12
Element 0	ArtExhibit-CeramicVase
Element 1	ArtExhibit-ChainedSquare
Element 2	ArtExhibit-ChainedTalisman
Element 3	ArtExhibit-ChurchChair
Element 4	ArtExhibit-Claypot
Element 5	ArtExhibit-CopperBox
Element 6	ArtExhibit-Doulapi
Element 7	ArtExhibit-GoldenBrooch
Element 8	ArtExhibit-SilverBowl
Element 9	ArtExhibit-Throne
Element 10	ArtExhibit-TurkinshMangali
Element 11	ArtExhibit-WoodenChest
Max Load Retries	4

Figure 17. The Exhibit Models Manager and its list of AR Objects that can be spawned.

3.7 Prefabs and Exhibit models

For each exhibit model that is going to be used in the application, a specific prefab needs to be created to be used in the Exhibit Models Manager. That prefab holds every functionality that is needed for the AR experience. Specifically, the “ArtExhibit” prefab has two important scripts on it, the “ExhibitItem” and the “TriggerHandlers”.

- ExhibitItem: It holds references to some parts of the prefab and controls the minimum and the maximum scale range that the model can reach when the user interacts with it.
- TriggerHandlers: This script handles most of the interactions that the user can perform. It checks if the user has viewed or has interacted with the AR object. From here, the minimum distance for interaction can be set. It is also responsible for the object’s texture. If the user is not at the minimum distance from the object, the texture’s color becomes dark. As the user gets closer the texture becomes clearer. If the user then interacts with it, the inverted texture is used to change the color of the object. Finally, the sphere collider that is on this prefab should be customized so that the developers can change the interaction area.

For future events, new AR objects can be created by simply adding a new model to “ArtExhibit” prefab under its child root called “GraphicRoot”.

3.8 Views and Interactions

As it was mentioned before the script “TriggerHandlers” handles most of the events and interactions that the user can perform on an exhibit. Before an interaction with an exhibit can begin, the user needs to be a certain distance from it. This is controlled by a sphere collider in Unity that can be adjusted for each exhibit.

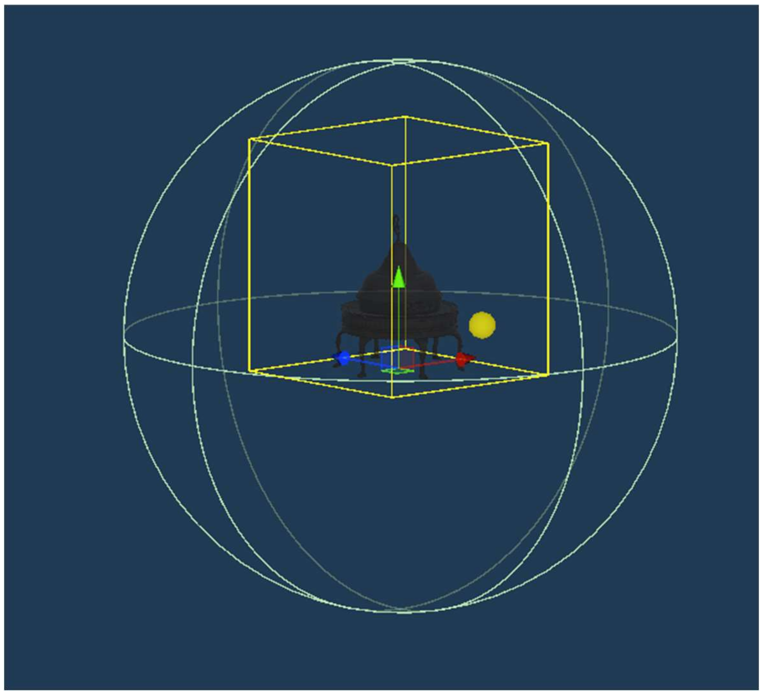
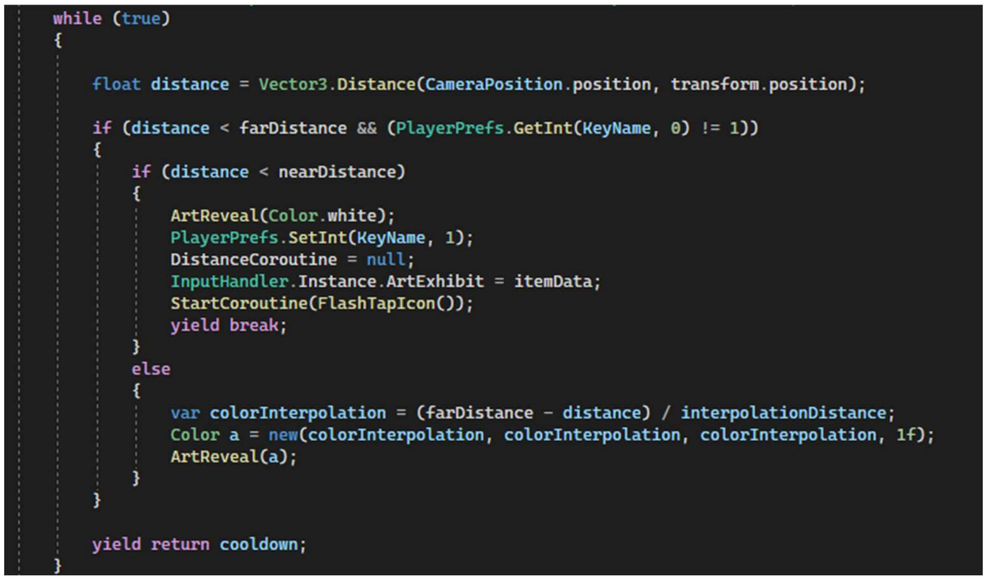


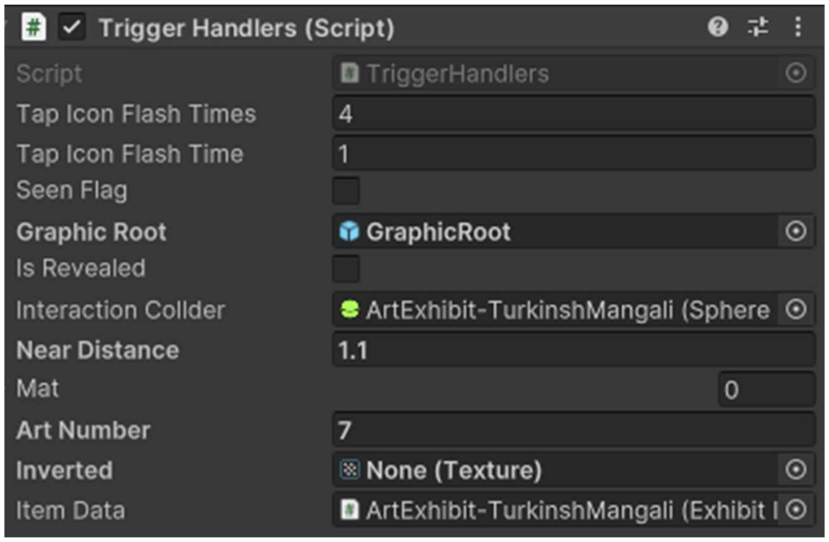
Figure 18. Green sphere collider that defines the limits that the user must be to begin any interactions.

At first, the exhibits appear hidden with a black texture. As the user gets closer to them, the texture gradually appears. When the whole texture is revealed a small tap icon appears which signals the user that he can now interact with the exhibit by touching it and change its texture to negative colors. The sphere collider and the distance the user must cover to reveal the exhibit’s texture are customizable through Unity’s editor.



```
while (true)
{

    float distance = Vector3.Distance(CameraPosition.position, transform.position);

    if (distance < farDistance && (PlayerPrefs.GetInt(KeyName, 0) != 1))
    {
        if (distance < nearDistance)
        {
            ArtReveal(Color.white);
            PlayerPrefs.SetInt(KeyName, 1);
            DistanceCoroutine = null;
            InputHandler.Instance.ArtExhibit = itemData;
            StartCoroutine(FlashTapIcon());
            yield break;
        }
        else
        {
            var colorInterpolation = (farDistance - distance) / interpolationDistance;
            Color a = new(colorInterpolation, colorInterpolation, colorInterpolation, 1f);
            ArtReveal(a);
        }
    }

    yield return cooldown;
}
```

Figure 19. Coroutine that checks the user's distance from the exhibit and slowly reveals it as the user gets closer.



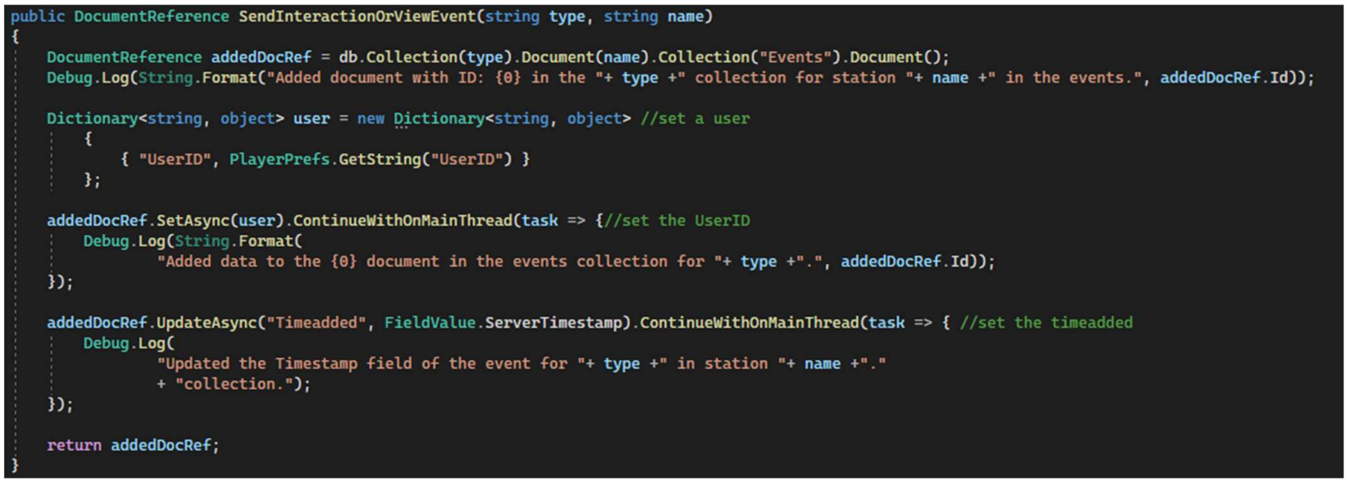
Trigger Handlers (Script)

Script	TriggerHandlers
Tap Icon Flash Times	4
Tap Icon Flash Time	1
Seen Flag	
Graphic Root	GraphicRoot
Is Revealed	
Interaction Collder	ArtExhibit-TurkinshMangali (Sphere
Near Distance	1.1
Mat	0
Art Number	7
Inverted	None (Texture)
Item Data	ArtExhibit-TurkinshMangali (Exhibit I

Figure 20. Adjustable settings in the Unity editor such as the activation distance.

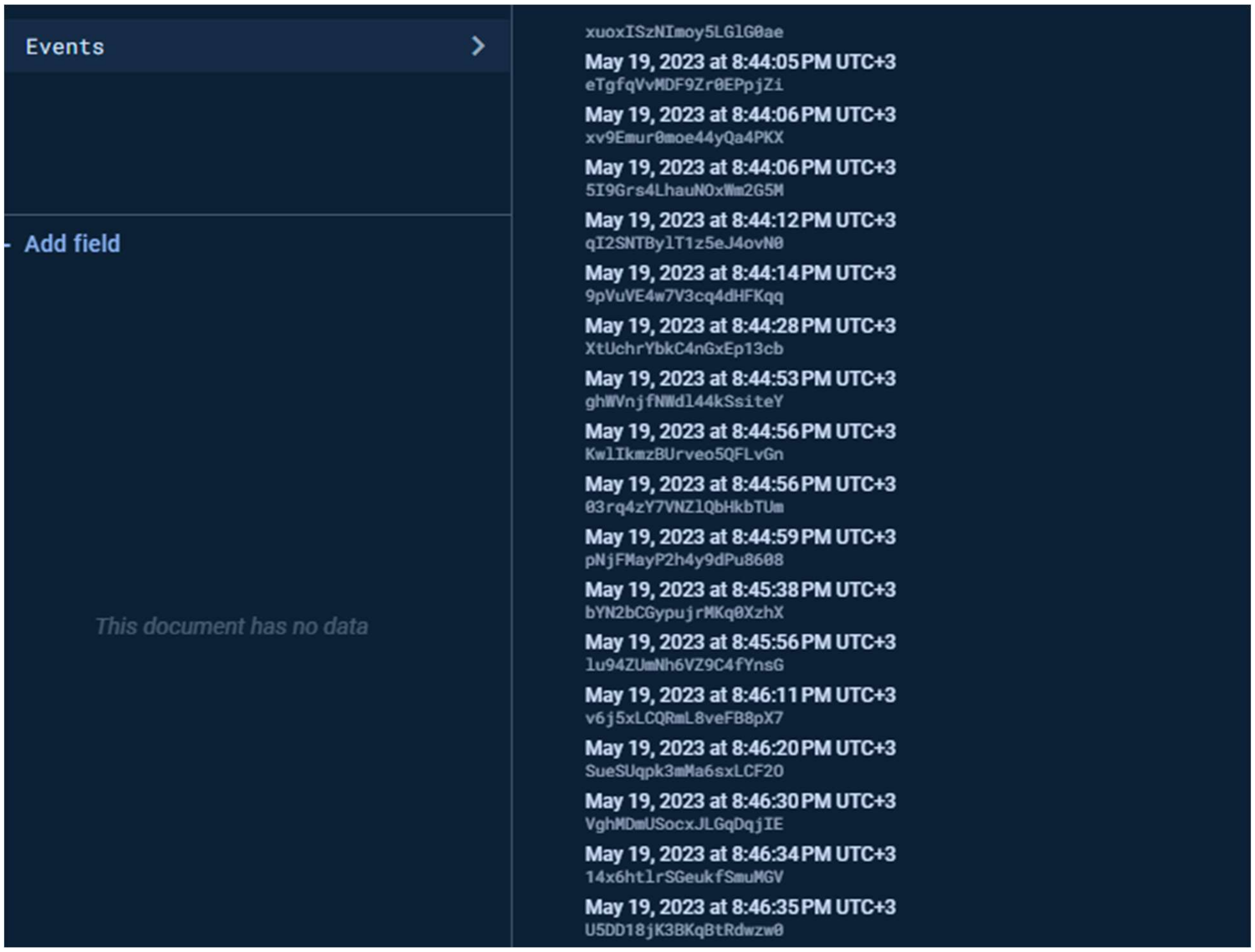
3.9 Firebase and Data Handling

The two types of data that are stored are how many users have viewed (and for how long) an exhibit and if they managed to interact with it. Both are stored online by using Google’s online Database solution called Firebase and specifically the Firestore database. The views are recorded when the user enters the aforementioned sphere collider that exists in every exhibit and are updated with an exit time when the user exits the collider (in order to calculate the duration of viewing). Interactions are recorded if the users tap the exhibit when prompted by the application and change the exhibit’s texture.



```
public DocumentReference SendInteractionOrViewEvent(string type, string name)
{
    DocumentReference addedDocRef = db.Collection(type).Document(name).Collection("Events").Document();
    Debug.Log(String.Format("Added document with ID: {0} in the "+ type +" collection for station "+ name +" in the events.", addedDocRef.Id));

    Dictionary<string, object> user = new Dictionary<string, object> //set a user
        {
            { "UserID", PlayerPrefs.GetString("UserID") }
        };

    addedDocRef.SetAsync(user).ContinueWithOnMainThread(task => {//set the UserID
        Debug.Log(String.Format(
                "Added data to the {0} document in the events collection for "+ type +".", addedDocRef.Id));
    });

    addedDocRef.UpdateAsync("Timeadded", FieldValue.ServerTimestamp).ContinueWithOnMainThread(task => { //set the timeadded
        Debug.Log(
                "Updated the Timestamp field of the event for "+ type +" in station "+ name +"."
                + "collection.");
    });

    return addedDocRef;
}
```

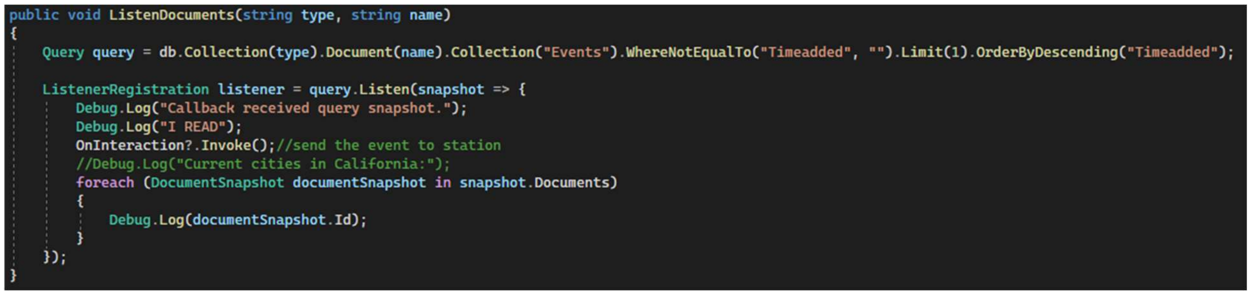
Figure 21. Code snippet of sending a view or interaction event to Firebase. Each event gets registered along with a unique userID.



```
xuoxISzNImoy5LGlG0ae
May 19, 2023 at 8:44:05 PM UTC+3
eTgfqVvMDF9Zr0EPpjZi
May 19, 2023 at 8:44:06 PM UTC+3
xv9Emur0moe44yQa4PKX
May 19, 2023 at 8:44:06 PM UTC+3
5I9Grs4LhauNOxWm2G5M
May 19, 2023 at 8:44:12 PM UTC+3
qI2SNTBylT1z5eJ4ovN0
May 19, 2023 at 8:44:14 PM UTC+3
9pVuVE4w7V3cq4dHFKqq
May 19, 2023 at 8:44:28 PM UTC+3
XtUchrYbkC4nGxEp13cb
May 19, 2023 at 8:44:53 PM UTC+3
ghWVnjfNWdl44kSsiteY
May 19, 2023 at 8:44:56 PM UTC+3
KwlIkmzBUrveo5QFLvGn
May 19, 2023 at 8:44:56 PM UTC+3
03rq4zY7VNZlQbHkbTUm
May 19, 2023 at 8:44:59 PM UTC+3
pNjFMayP2h4y9dPu8608
May 19, 2023 at 8:45:38 PM UTC+3
bYN2bCGypujrMKq0XzhX
May 19, 2023 at 8:45:56 PM UTC+3
lu94ZUmNh6VZ9C4fYnsG
May 19, 2023 at 8:46:11 PM UTC+3
v6j5xLCQRmL8veFB8pX7
May 19, 2023 at 8:46:20 PM UTC+3
SueSUqpk3mMa6sxLCF2O
May 19, 2023 at 8:46:30 PM UTC+3
VghMDmUSocxJLGqDqjIE
May 19, 2023 at 8:46:34 PM UTC+3
14x6htlrSGeukfSmuMGV
May 19, 2023 at 8:46:35 PM UTC+3
U5DD18jK3BKqBtRdwzw0
```

Figure 22. Records of view events inside the Firestore database in chronological order.

Lastly, the Firestore database sends any incoming interaction events to the Base stations so that a sound can be played. This is done automatically by the database if the Base station registers with that event for a specific exhibit.



```
public void ListenDocuments(string type, string name)
{
    Query query = db.Collection(type).Document(name).Collection("Events").WhereNotEqualTo("Timeadded", "").Limit(1).OrderByDescending("Timeadded");

    ListenerRegistration listener = query.Listen(snapshot => {
        Debug.Log("Callback received query snapshot.");
        Debug.Log("I READ");
        OnInteraction?.Invoke();//send the event to station
        //Debug.Log("Current cities in California:");
        foreach (DocumentSnapshot documentSnapshot in snapshot.Documents)
        {
            Debug.Log(documentSnapshot.Id);
        }
    });
}
```

Figure 23. Registering with a specific exhibit. Each time a new interaction happens, the database informs the base station.

3.10 Base station

The Base station application is responsible for playing various sounds when a user interacts with a specific exhibit. Each Base station can be assigned to one exhibit. Before the performance the exhibits should each be given a number (this is done through their prefab settings in the TriggersHandler script and assigning a value to the “Art Number ” variable). Then when the application is launched, assign the correct “Art Number”. After that the Base station will listen for interaction events that correspond to that exhibit ID and will play a sound if there is one added.

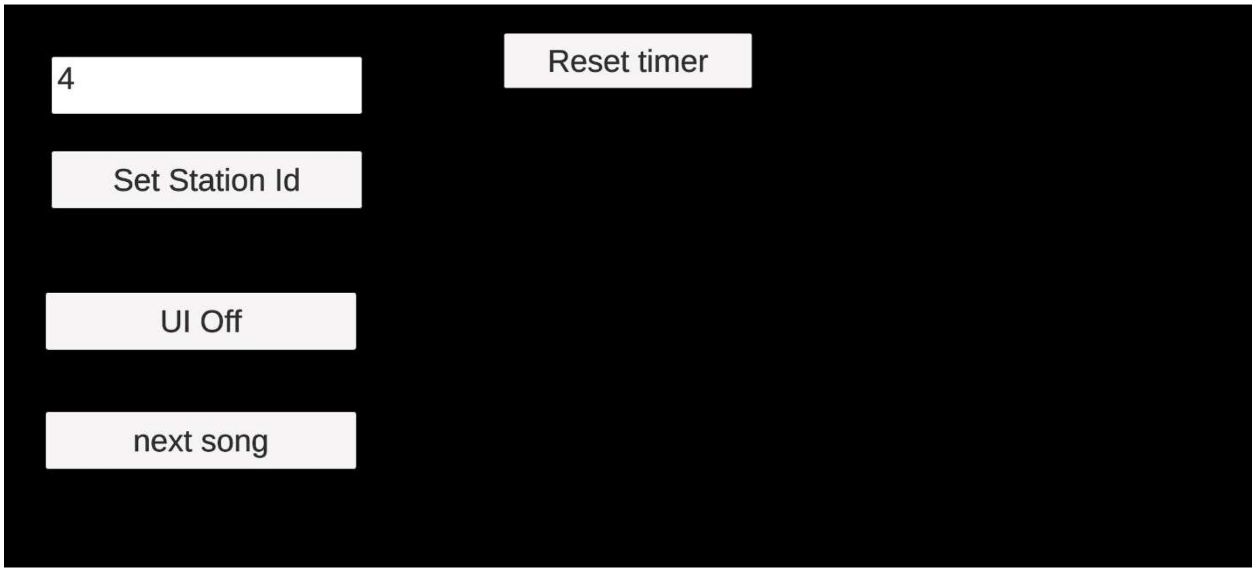


Figure 24.Base station UI. On the top left input field, the user specifies which exhibit this Base station is for. Pressing the "Set Station Id" the base station registers with the database and receives any new interactions that can occur.

It should be noted that the Base station needs to be connected to the internet during the performance to communicate with Firebase.