Technical Report

João Henriques
j.tiago.henriques@tecnico.ulisboa.pt

Instituto Superior Técnico, Universidade de Lisboa
Advisor: Prof. Rui Prada
Co-Advisor: Prof. Patrícia Gouveia

Table of Contents

1 Documentation Of Our work

During this thesis, I created an Augmented Reality App to interact the Cathode Ray Artifact of the Faraday Museum. In this section we present the documentation of this work. To understand better the context of the work and the reasons take a look on my thesis.

1.1 Releases

There are two main repositories that should be taken as reference:

- My repository: contains all the work done for the thesis, the documents, the code, the important releases apk and necessary files, targets, etc.;
- And the official Extended Play At Faraday Museum Repository, that contains all the work done in terms of code, with releases and apks too. It contains the work from everyone that already worked on this thesis.

The most important releases are:

- WebGL Cathode Ray app[1]: An WebGL app of the Cathode Ray augmentation;
- Android Image Target Cathode Ray app[2]: An Android app of the Cathode Ray augmentation in English, that works with a image target;
- Android Model Target Cathode Ray app[3]: An Android app of the Cathode Ray augmentation in English, that works with a image target;
- Android Extended Play At Faraday Museum v3.1.0 app[4]: An Android app with all the work done so far. Cathode Ray augmentation in Portuguese. Notice that however, this is the most advanced release of the thesis, is far for being over. Since, the UI doesn't match between each game.

Notice that, the Android Image Target Cathode Ray app and the Android Model Target Cathode Ray app, only contains my work, and is the most advance and cleaned repository of this thesis. i.e., some code was rewritten to improve readability and efficiency and the repository was cleaned, so it doesn't contains unnecessary assets,files and scenes.

The Extended Play At Faraday Museum release contain all the work done so far, however since my project had major changes in terms of interface, the interfaces between the projects don't match. There is no documentation to their work, or what target device they where aiming, the only info is that it was android. So, it is very hard to quickly change their UI.

[1] https://github.com/Toscan0/IST-Thesis-FaradayMuseum/releases/tag/v1.0.0_WebGL

[2] https://github.com/Toscan0/IST-Thesis-FaradayMuseum/releases/tag/v2.0.0_Android_ImageTarget_CathodeRay

[3] https://github.com/Toscan0/IST-Thesis-FaradayMuseum/releases/tag/v2.0.0_Android_ModelTarget_CathodeRay

[4] https://github.com/ruiprada/FaradayMuseumAR/releases/tag/ExtendedPlayAtFaradayMuseum_v3.1.0

Note also, that it is possible to encounter some bugs, since some alterations in the Android Releases couldn't be tested due to COVID-19. However, I hope that this document will help to solve any problem found in my project

This documentation covers the code of the Android Image Target / Model Target Cathode Ray app.

1.2 Version

In term of versions:

- WebGL Cathode Ray app: Unity3D 2018.4.27f1 (LTS);
- Android Image Target Cathode Ray app: Unity3D 2019.4.15f1 (LTS) with Vuforia 9.5.4;
- Android Model Target Cathode Ray app: Unity3D 2019.4.15f1 (LTS) with Vuforia 9.5.4;
- Android Extended Play At Faraday Museum v3.1.0 app: Unity3D 2018.4.27f1 (LTS) with Vuforia 8.3.8.

1.3 Expertise Level

The SettingsManager.cs file contains the expertise level of the user in the electromagnetism field. There are 3 possible values: Expert, medium, beginner, where the default value is beginner. This expertise level is used to define what content the pop-ups should have. See Subsection 1.9.

At this time, we are only using the beginner level. And we don't give an option to the user change this. However, the code is thought to be an option in the start menu.

1.4 Targets

In terms of Vuforia Targets, there are two that make sense in the context of this thesis, that are: image target and model target.

They work in a very similar way, namely from the user point-of-view, where it is just pointing the device to the target, and for the developer point of view where it is only generating the target. Regarding the image target, it is generated online through Vuforia developer portal[5] and the model target through the Model Target Generator[6].

After generating the pretended target, import the target database. On the target game object is necessary to add the MyTrackableEventHAndler.cs, and in older versions the Game.cs. Check 2.1 to see in detail what to do.

One very important aspect, is the TargetID field on the MyTrackableEventHAndler.cs, this ID should be unique, and is case sensitivity to the pop-ups, BLE and achievements. For future reference, the target ID of the Cathode Ray, is CR.

[5] https://developer.vuforia.com/

[6] https://library.vuforia.com/articles/Solution/model-target-generator-user-guide.html

1.5 BLE

To connect the mobile phone to the BLE device we use the asset Arduino Bluetooth Plugin[7]. We didn't testes with museum device, since it's not available yet, but it works with our ESP32 Arduino.

The connection is managed in the BLEManager.cs and if connected it waits for a message in a string format.

It only trys to connect if it's tracking an target. To decide what BLE to connect or not, there is a function in the BLEManager.cs that checks if the target ID connected is equal or not to one to any id there. Is yes, connects to the respective BLE device, if not nothing happen.

If receives something, it calls the parser function in the Parser.cs file. The parser, splits the message by ";" and checks to each target is related by the ID. In my case, checks what values are received and respectively update the Control Panel in the respective scripts (RotationUI.cs, IntensityUI.cs, TensionUI.cs).

The BLEManager.cs and Parser.cs are both in the BluetoothManager component. The BLEManager.cs file has a Serialize bool named connectToBLE that defines if the app should try to connect or not. In our work this bool should be checked for Model Target and not checked for Image Target.

1.6 The control Panel

The control panel is where the user can interact with the Cathode Ray values. Depending on which version (WebGL / Android) are you working with, it can have 3 input formats (WebGL): Slide, buttons and input text, or two (Android): buttons and input text.

All these elements are connected to their respective script (RotationUI.cs, IntensityUI.cs, TensionUI.cs), and if a value is changed, either by the user or by Bluetooth, an event is launched saying that this value has changed and to which value it has changed.

The event is received in the ManageImput.cs the value is stored and then the shape is calculated and the e-bean drawn.

The connection between the control panel scripts and the manage input, is made by a an event due to the fact that there are two targets that are interested on this event. So, at first, we have one event instead of two function calls, and if one of the targets is disable or didn't exist there is no problem, contrary to what happened with functions. Thus, is possible to disable one target if only one is necessary to the apk.

1.7 e-bean

To draw the e-bean we use the Line Render Component, the line render basically works by interpolation of points. so, we established the points and the component

[7] https://assetstore.unity.com/packages/tools/input-management/arduino-bluetooth-plugin-98960

draw a line between them. To represent the e-bean we use blue material, however this should be changed by someone how understand art, to something more close to the real e-bean aspect.

We use the formulas given by professor Carlos Ferreira Fernandes, however this may need an approximation to reality. We couldn't test if they are accurate due to the fact that the artifact is in reparation.

The calculation of shapes takes place in the CalculateShape.cs script, that also checks the achievements. Then, calls the respective function on the draw.cs.

Due to the fact that the image target is smaller than the model target, a scale factor is applied between them. Making certain values, have to be specific to the target we want. This values are present in CalculateShape.cs or CheckAchivements.cs, denpending of your version, and in the Draw.cs.

The augmented part, i.e. the fade blue line is calculated by a shader, that basically receives the ampule position, and when is drawing the material created with that shaded, checks the position of the vertex with the position of the ampule and sees if is in our out, giving a blue or a fade blue color. To the circle is need a second step, that's breaking the line after the first point out of the circle, and draw it on DrawAux.cs that is very similar to the draw, but only draws with the fade blue line.

1.8 Electric Augmentation

The electric augmentation, is accomplished by an particle system. Basically, it creates the particles in a circular shape, and we give an yellow color than at the end is transparent. So, the particles are born with a yellow color and die with a transparent colour. What we accomplish with that, is that, when the lasts particles of the circle are born (with yellow color) the last ones are transparent (that are right on their side), and so on, giving an idea of movement. That represents the direction of the electric current.

This representation, updates their size with the intensity value. When the intensity changes, the ManagerInput.cs triggers an event to the CoilsManager.cs that convert this value the for an appropriate scale for coils size.

1.9 Pop-ups

The are four types of pop-ups: initial explanation, explanations, hints, and game end.

The pop-ups are created by Scriptable Objects, that are stored in the Folder AssetsScriptableObjects>CathodeRay. This works as database for the pop-ups.

They contain: artifactId, title, description, expertise level and a ID. Only the title and description are showed to the user. The ID only exists in the hints and explanations, and should match the ID of the achievements.

Each pop-up has a GameObject on MainCanvas that is automatic populate by the respective scriptable object when enabled.

For example, when the target is scanned for the first time, the InitialExaplantion is enabled for that artifact. So, the InitialExplanationDisplay.cs checks what

is the target and the expertise level, and load the scriptable obejct to the UI. In the case of explanations, it works on the same way, however, before enabling the GameObject is necessary to set the ID.

In our work, we only give explanations when one achievement is completed, so this ID needs to match the respective achievement.

Hints The hints are created in the same way the explanations are. However, instead of being enabled when an achievement is completed, they have a timer of 1 minute that start to count when the user complete the first achievement, if the user don't complete the second achievement in 1 minute, the hint is showed for the second achievement. When the user completes the achievement the timer is reseted.

1.10 Achievements

The achievements system was not created by me, however here is what I found most important to know.

The achievements are scriptable objects, and are kept in the file Assets>Data>Achievements>Achievement Database.asset

There is a file called Achivement.cs in the folder Scripts, that contains a public enum called Achievements, with all the achievements existing in the App. This script is generante automaticly and an example can be:

```
public enum Achievements {CR0,CR1,CR2,CR3,CR4,CR5,}
```

The AchievementManager.cs is where is the most important logic in terms of using the achievement system.

To complete on achievement use the following function:

```
//Where <achievements> can be Achievements.CR0
achievementManager.IncrementAchievement(<achievements.id>);
```

This function returns a bool.

There is also a complementary function, which is:

```
//Where <achievements£>£ can be Achievements.CR0
AchivemententUnlocked(<achivement.id>)
```

This function creates an explanation if the achievement has an explanation associated, and resets the timer of the hint.

1.11 LogTests

Basically this systems records logs of the user interaction. It records on a Web-Server and on the Android device

There are 3 files that create the system.

The EventData.cs contains an public enum LogEventType, with the types of events that we want to record.

The UsabilityTestsSingleton.cs is the main class of the LogTest system. To record an event, there are two functions

```
public void AddGameEvent(LogEventType eventType)
public void AddGameEvent(LogEventType eventType, string objectName)
```

In my case we use ObjectName for additional information. Both of this functions save automatically the information in the android device and keep the information on a List for later send to WebServer. When an Event is recorded is recorded whit a time stamp.

To send the information to server we use UnityWebRequest, and the information is sent every 5 seconds.

The information is sent to the url defined in the DEVELOPER_ULR_ADDRESS variable. This url should be pointing to a php file with the following content:

```
<?php
        $file_name = $_POST["_fileName"];
        $content = $_POST["_content"];

        /* Create the file if doesn't exist, if file exists add
        * the content to the already existing content
        * (no over write)
        */
        file_put_contents("Data/" . $file_name, $content, FILE_APPEND);
?>
```

The file name is defined in the logFileName variable, and is created every time the app starts. The name consists in the starting date of the app (Year-Month-Day_Hour-minute) and followed by 14 random digits. A possible example is:

```
        logFileName = "2020-11-09\_15-27\1721184\_4526535.txt";
```

The LoadSceneSingleton.cs is where the file name is created.

In older versions is possible that the code doesn't has this organization, however it follows the same logic.

To record an event use the following code:

```
public static UsabilityTestsSingleton singleton =
    UsabilityTestsSingleton.Instance();

singleton.AddGameEvent(<logEventType>);
//or
singleton.AddGameEvent(<logEventType>, <aditionalInfo>);
```

Take in mind that the WebServer is public access, and anyone can see the information saved. Also, it's not necessary to check for credentials, so, anyone can send or retrieve information from the server.

1.12 Animations

In terms of animations I use 2 types of animations Fade In and Fade Out with 40 sec of duration and Slide with 30 sec of duration.

1.13 App Guide Wire

The way or game is organized if the following:

- The app starts in a warning scene;
- The user makes the scan of the target;
- Appears the 3D model with buttons, control panel and the Initial Explanation;
- If the user completes the first achievement (increase the Intensity);
- Appears the first Explanation;
- If the user completes the second achievement (Create a line, Rotation = 0^{o}, 180^{o} or 360^{o});
- Appears the second Explanation;
- If the user completes the third achievement (Create a circle, Rotation = 90^{o});
- Appears the third Explanation;
- If the user completes the fourth achievement (Create another circle);
- Appears the fourth Explanation;
- If the user completes the fifth achievement (Create a spiral);
- Appears the fifth Explanation;
- If the user completes the sixth achievement (Create another spiral);
- Appears the game end pop-up;

This order can't be violated, i.e., the user needs to create this achievements in this order or nothing happens, it doesn't matter if the user creates a circle and then a line, it only receives the line achievement. To complete the circle achievement, needs to make the circle again.

Also, if the user in the third achievement created a circle with a radius bigger than the ampule in the fourth achievement needs to create one with a radius smaller than the ampule and vice versa.

2 Wiki How

2.1 How to create a target

As said in Subsection 1.4, there are two important types of targets. The image target is generated online through Vuforia developer portal[8] and the model target through the Model Target Generator[9].

After generating the target load the package.

2.2 Image Target

To create an Image target right click the scene hierarchy then select Vuforia Engine>Image Target (Figure 1).

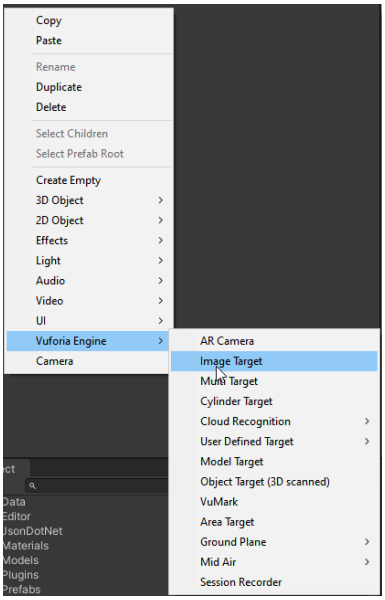

Fig. 1: Create image target.

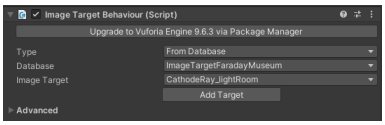

Fig. 2: Image Target Behaviour component.

Then on the component Image Target Behaviour select the database and target that you want (Figure 2).

Then add the component MyTrackableEventHAndler.cs, and fill the necessary fields (Figure 3). Where UI is the game interface which must be shown when tracking the target. If there isn't a UI, don't add any.The initial explanation is the initial explanation game object from the Main Canvas. The target manager and artifact ID are self explanatory. Make sure to check the Is image Target Boolean.

2.3 Model Target

To create a Model target right click the scene hierarchy then select Vuforia Engine> Model Target (Figure 4).

[8] https://developer.vuforia.com/

[9] https://library.vuforia.com/articles/Solution/model-target-generator-user-guide.html

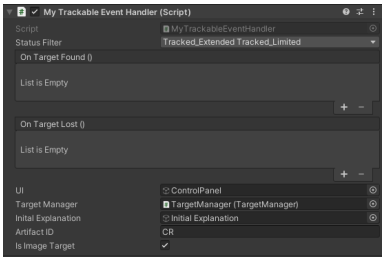

Fig. 3: My trackable event handler component.

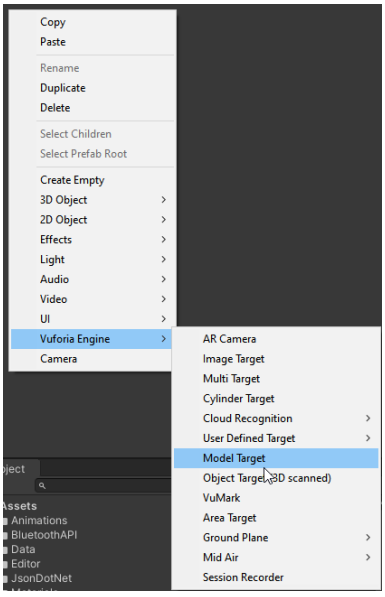

Fig. 4: Create model target.

Then on the component Model Target Behaviour select the database and the target that you want (Figure 5).

On the Guide View Mode, you can select between No Guide View, Guide View 2D or Guide View 3D. To appear a preview of the object so the user can have a guide of what the object looks like and how should it match the screen with the artifact, select Guide View 2D, and then select which guide view you want in the Guide View Attribute. (Need to have created a Guide View on the Model Target Generator)

Then add the component MyTrackableEventHAndler.cs, and fill the necessary fields (Figure 6). Where UI is the game interface which must be shown when tracking the target. If there isn't a UI, don't add any.The initial explanation is the initial explanation game object from the Main Canvas. The target manager

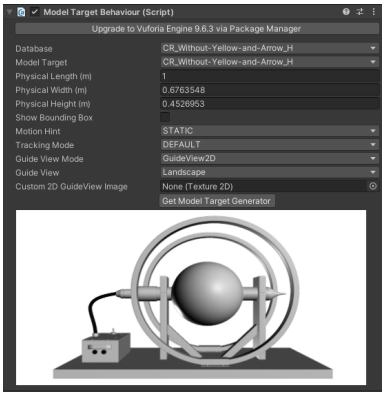

Fig. 5: Model Target Behaviour component.

and artifact ID are self explanatory. Make sure not to check the Is image Target Boolean.

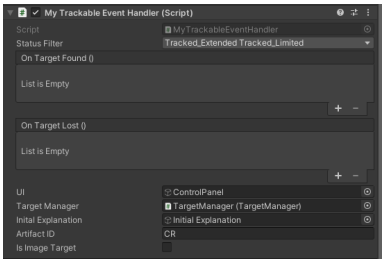

Fig. 6: My trackable event handler component.

2.4 Targets in older versions

In older versions, in addition to add MyTrackableEnventHAndler.cs it's also necessary to add the Game.cs. Also, the MyTrackableEnventHAndler.cs have more fields. Some of them, we don't know what they do since their are part of another ongoing project. However, follow the Figure 7.

2.5 How to create a initial explication

On the project window, on Assets>ScriptableObjects>CathodeRay>Initial Explanation right click on mouse and then select create>ScriptableObjects>initialExplication (Figure 8).

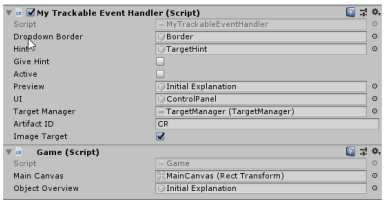

Fig. 7: Target components in older versions.

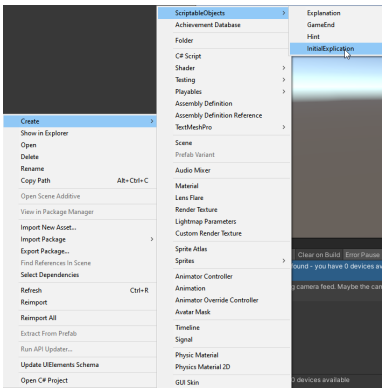

Fig. 8: Create a Initial explanation.

The the initial explanation is created, fill with the pretended values (Figure 9). The title and description are the only information visible to the user. The Artifact ID and expertise level are case sensitive.

Fig. 9: Initial explanation example.

Then add the initial explanation to the initial explanations list on the Initial Explication Display component on the Initial Explanation Game Object, inside the Main Canvas Game Object (Figure 10).

Then add the target to the targets list in the Target Managers Game Object

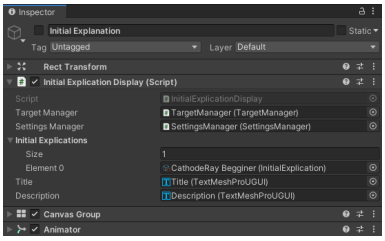

Fig. 10: Initial explanation Game Object.

2.6 How to create a explication

On the project window, on Assets>ScriptableObjects>CathodeRay>Explanation right click on mouse and then select create>ScriptableObjects>Explanation (Figure 11).

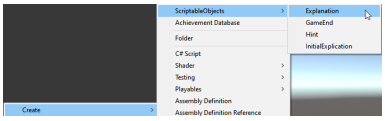

Fig. 11: Create an explanation.

The the explanation is created, fill with the pretended values (Figure 12). The title and description are the only information visible to the user. The Artifact ID, ID (Achievement ID) and expertise level are case sensitive.

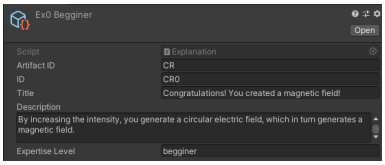

Fig. 12: Explanation example.

Then add the explanation to the explanations list on the Explanation Display component on the Explanation Game Object, inside the Main Canvas Game Object (Figure 13).

Then add the explanation to the explanations list on the Explanation Display component on the Arrow Container Game Object, inside the Main Canvas Game Object (Figure 14).

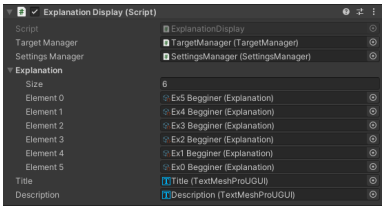

Fig. 13: Explanation Game Object.

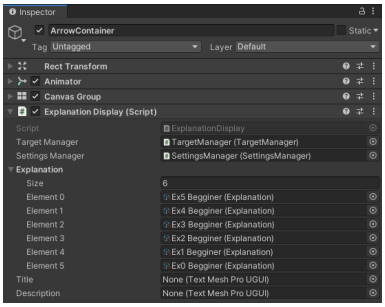

Fig. 14: Arrow Container Game Object.

2.7 How to create a hint

On the project window, on Assets>ScriptableObjects>CathodeRay>Hint right click on mouse and then select create>ScriptableObjects>Hint (Figure 15).

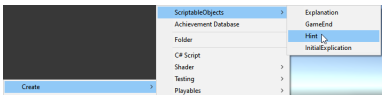

Fig. 15: Create an Hint.

The the hint is created, fill with the pretended values (Figure 16). The title and description are the only information visible to the user. The Artifact ID, ID (Achievement ID) and expertise level are case sensitive.

Then add the hint to the hints list on the Hint Display component on the Hint Game Object, inside the Main Canvas Game Object (Figure 17).

2.8 How to create the end-game explication

On the project window, on Assets>ScriptableObjects>CathodeRay>GameEnd right click on mouse and then select create>ScriptableObjects>GameEnd (Figure 18).

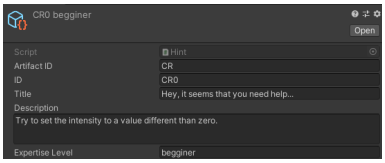

Fig. 16: Hint example.

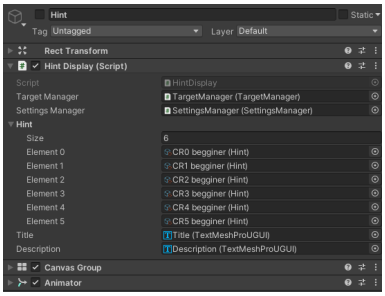

Fig. 17: Hint Game Object.

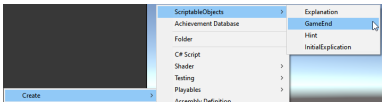

Fig. 18: Create a end-game explication.

The the end-game explanation is created, fill with the pretended values (Figure 19). The title and description are the only information visible to the user. The Artifact ID and expertise level are case sensitive.

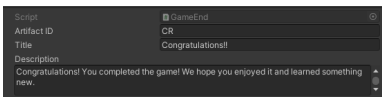

Fig. 19: End Game explication example.

Then add the End Game explication to the game end list on the Game End Display component on the Game Completed Game Object, inside the Main Canvas Game Object (Figure 20).

2.9 How to add or delete an achievement

The achievement database is situated in Assets>Data>Achievements>Achievement Database, and can be visualized in the Figure 21.

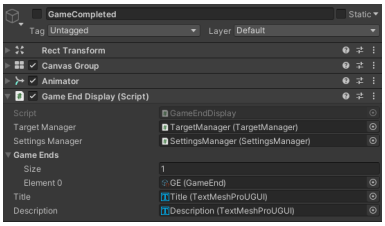

Fig. 20: Game Completed Game Object.

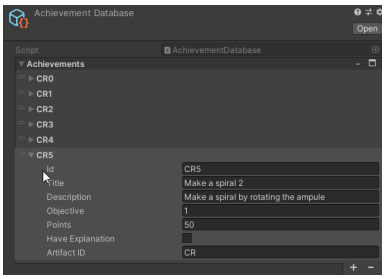

Fig. 21: Achievement Database.

To delete one achievement use the minus button, it will always delete the last achievement (in the case of Figure 21, it will delete the CR5 achievement).

To add a new achievement use the plus button, it always copy the last achievement (in the case of Figure 21, it will delete the CR5 achievement). After created, change the necessary attributes.

Regarding the achievements attributes:

- ID: case sensitive with explications and hints, not showed to the user;
- Title: title of the achievement, showed to the user;
- Description: description of the achievement, showed to the user;
- Objective: How many times need to be completed, not showed to the user;
- Points: How many points the achievement is worth, showed to the user;
- Have Explanation: If is supposed to show an explanation when the achievement is completed, not showed to the user;
- Artifact ID: case sensitive with target id, not showed to the user;

2.10 Scene Hierarchy

The scene hierarchy of our our project can be visualized in Figure 22. Where the Game Objects are organized by types inside of "===" Game Objects, were this have the tag "EditorOnly" so they don't take memory space in the build and consequently on the user's device (Figure 23). They are organized from top to bottom in relation to the possibility of change, i.e., the higher the position the less likely it is to be changed, which improves the visibility while working of the most likely, because they are closer to you.

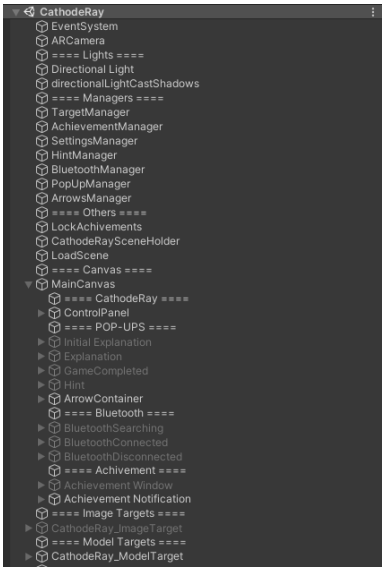

Fig. 22: Scene Hierarchy.

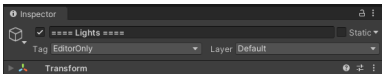

Fig. 23: Editor only tag.

3 Conclusion

This project is a project with immense capabilities. It arouses great interest in the youngest, the teenagers and even the elder people. It has been tested and used by different students, where very positive results have been obtained. This is a project that can be used for several years, and its development and maintenance for that to happen is extremely important.

However, it is only possible to develop a project for so many years, if the project is well maintained. So it's very important that every developer documents the code that he did.

It is also important that the project be consistent in terms of design, so it is also important that designers document the UI aspect. The size of the images, the colors scheme, the fonts and letter size, so that the next developers can follow this rules.

Every time that a user uses this project, the project records information on how it was used. The amount of information generated is still large for a project of this size, and should be maintained to be able to make more severe and comprehensive studies on the application. It's also very important that the next developers don't damage the work already done.

It is also important that this project continues to be developed with the Unity account of the gaming laboratory and Vuforia account of the museum, and not with the personal account of each student. This is the only way to ensure that data and databases are not lost when passing folders between students.

In order to avoid code loss problems, we also suggest that it should be created an repository in a version controller, such as Git Hub, where all the work will be developed

For all of this reasons, we strongly suggest that this project has someone as product owner (PO). That understand the code, and the tools used. This PO would be the owner of the repository, and would guarantee the quality of the code and the project. Each student could have his own branch, and the repository a pull request system, that the PO would manage.

One of the initial objectives of this project is that this application can be used at home, since some mini-games need interaction with the real object. It's necessary that the developers design the mini-game to be able to use with and without the interaction with the real object. For the museum to be able to use the work carried out in dissemination activities it is important that the entire application also works with image target. In conclusion, each developer must deliver at the end a fully functional release with model target, and one with image target.