Hall of the Red Earl
Augmented Reality Documentation

Introduction:

The goal of this project is to create an augmented reality application which re-creates the Hall of the Red Earl in Co. Galway so that upon entering the Hall of the Red Earl the user of the application can view the Hall of the Red Earl as it was many years ago in augmented reality on their device rather than its current state, allowing users to experience what the building once was when they scan the image which will be displayed in the hall on their device. The hall is the oldest building in Galway and has great historical significance in the city. Therefore attracts many tourists from all over the world who come and view the building upon visiting Galway. The building in its current state is in ruin with just pillars and rocks remaining from the original site. Leaving it hard for tourist to reimagine what the building is like with only drawn up pictures on the wall to help them visualize it as there was no photos to be taken back then. Therefore the owner of the hall has requested us to recreate the hall so that tourists can see what the building was like in its original state and also to create more interest in the building due to the augmented reality application being created. The owner intends on having us present the application during a medieval festival to gain interest to the application and the hall.

Design:

The application will be designed using unity and ARCore using importable packages which will allow us to recreate the Hall of the Red Earl in its original state back in 1271. We intend to recreate the inside of the Hall of the Red Earl as well as allow users to access 3D overhead birds eye model of the hall when pointing their camera to a specific image inside of the hall. The owner of the hall stated he just wanted a recreation of the hall as it was, rather than a full tour of the hall, which had been our original idea along with re-creating the hall, i.e. have all artefacts in the hall display information about them if the user pointed their device at them. But the owner of the hall stated that there was another team working on just a tour of the hall using just image detection so we re-designed to just focus on the hall itself in augmented reality. The aim is to allow users to enter the hall and download the application and upon opening the application be greeted with a screen telling them to scan the image which is needed to spawn the model of the hall in Augmented reality on their device. When spawned the user should be able to navigate around the hall by walking around with their device and be able to “Exit” the hall and be able to view the exterior of the hall as it once was.

Implementation:

We used ARCore to implement this project, as it is the SDK we found to have the best features which suited our project and the most stable choice and user friendly experience of from the SDK's we tested. Unity is used to implement the ARCore packages to a project which from there we use the ARCore's augmented image scene to create the augmented reality project. We use visualiser scripts and controller scripts which are implemented in the Controller which allows for the recreation of the hall in augmented reality. This is done by creating a model of the hall and importing it into the unity project, it is then attached to an image in a database which maps the model to the image. The model is pulled into the controller as an object. The controller then displays the model it needs to display when an image is detected on the device when it is run. When the model is placed anchors hold the model in place and allow the user to walk around it and inside it.

Developer's Guide - Clone project from git repository into folder, open project in unity, go to file, build settings, ensure android is selected as the platform, press build and run with your intended device to run application on (must be android and have version 7.0 or greater)

User's Guide - The user needs to download the apk to their device (Application will be put on app store at a later date). When downloaded the user should open the application and when opened the user is greeted with a screen welcoming them and telling them to locate the image which is needed to spawn the building. After being located the user must fit the image inside the scanning layout displayed on the device and the building will then spawn. When spawned the user can explore the building inside and out. The application is a single user experience only. Although other people can be seen on your camera in the background e.g. through windows etc...

Research:

Research was a major area of our project as we were totally new to augmented reality and had no experience with using any of the SDK's needed to implement augmented reality. In fact we were also very unfamiliar with unity as we had only done two weeks of a module on it as part of a graphics program module.

SDK's: We needed to use an SDK to implement the augmented reality into unity, so we had a choice to make of which SDK to use. After three weeks of research and testing where we tried them out extensively to see which were the best SDK's out there that would suit our project, we chose the SDK we thought would be best to use.

Kudan - After researching Kudan we discovered that it would have been a better SDK when it comes to creating a mapping of the area in augmented reality for reconstructing rather than creating a reconstruction of the hall directly e.g. placing it for the user fully finished, Kudan may have been useful were we trying to get the user to create a reconstruction of an

area or object themselves and not create the hall for the user for when they use the application.

MapBox – MapBox again like Kudan mainly focuses on map creation but have their own AR section which they have available to developers for creating objects in AR. MapBox has a map created of the whole of Earth already mapped out and based on the users entered location may recreate that area using buildings etc… It works well in built up areas e.g. New York City but not as well in Galway (though it still recreates a map of the area). This could be used to map the building of the hall and recreate it but the problem we encountered is that the MapBox SDK is very complex to use with very little documentation available on how to achieve what is possible on the SDK and merely states that these ideas they show are possible on the SDK. Also, the documentation that is available is partially outdated in comparison with the current version of the SDK available, leaving it difficult to understand how the SDK works as they intended in their new updates released with many elements stated as being necessary to achieve something having been removed in the current update at the time of using. Also, the documentation that was available was very vague with many tutorials leaving us unsure as to what the actual tutorial intended to achieve in the final product or how useful it maybe rather than just implying this is possible using the SDK. For example, many tutorials had just pictures of buttons to click in their SDK modifiers for objects spawned in Unity rather than explain what or why we should be activating these options or stating what they are intended for. It appears MapBox's AR SDK has no image detection also which was important for us. After a while of playing around and testing the SDK, we decided that it may be best to try another. MapBox would've have been more suited to a project if our objective was trying to create an application similar to that of "Pokémon Go", requiring location object spawning rather than fixed building placement.

Vuforia- Vuforia was the SDK which we looked at first at the recommendation of our supervisor who gave us this task. The SDK is very user friendly and has very good documentation available, we spent about two weeks working with Vuforia as well as sampling other SDK's and comparing them with Vuforia. One problem we encountered with Vuforia is that the extended tracking is not very good when it comes to large scale objects which was the main element of our project seeing as we need the hall recreated in real life scaling. Another issue is with the stability of the tracking in Vuforia, Small movements by the user were stable but sudden or sharp movements by the user caused the positioning of the objects to be moved or in some cases disappear completely. Vuforia uses an image target to place objects which is very useful as we could place the hall for the user by looking at a point of interest in the hall e.g. a plaque.

Wikitude – Wikitude was an SDK we looked into which was very promising, it had all the features we were looking for and more. Including instant tracking, location based tracking, cloud markers, multiple image targeting and extended tracking. It also included a scene recogniser which would take multiple images e.g. 30, of an area you wish to recognise and from those created a whole reconstruction of the area which would then be recognised by the camera of the user when looking at the scene through their device and implement the objects in the area where they were placed in the scene creator in unity. This worked really well in tests and would have been ideal for the hall as we could've generated a scene from

the photos we had taken, but the issue we had with Wikitude was that they only offered a free trial to users, which wouldn't be applicable to an application we intend to place on the app store for both iOS and android. To use this SDK we would have to purchase a licence which would cost around €4490 per year to get the features we wanted and they did not offer a student usage licensing package from our research and got no reply from their team when we enquired about it.

ARCore – ARCore is an SDK by Google and has a lot of documentation and tutorials available to help understand the capabilities of the SDK. The main selling point of ARCore for us is that it is very stable when objects are placed, they stay in place even when moving completely them out of frame/view of the user camera e.g. so when user turns around and walks away, then looks in the same position they had placed the object, the object remains in that area. ARCore can use ground plane detection to achieve this resulting in that, the user must scan the ground to have the object placed or they can use image detection which places the object when the image is found and it stays in position using anchors. Image detection is the method we wish to use as we don't want the user to have to walk around the hall and scan the ground first. Also, when walking inside the object ARCore was a much better experience for the user than Vuforia due to these anchors. The problem with ARCore is it is more difficult to get working in comparison with Vuforia, but the stability of the objects makes it a much better SDK for the Hall reconstruction and as a result we chose this as our SDK to use for the project.

3D modeling Software: In order to create the model of the building we had to look into modeling software to use to try and create a model, luckily throughout our development, a team helped us by giving us a model they had been working on for their project of Galway, but the model they gave us was a model of the hall in ruins i.e. No roof and holes all over the walls, no pillars existed etc… But this gave us the outline of the positioning of pillars and walls etc... So we still had to import the model and edit it to make the building as it was in its original state.

Blender - Blender is an open source software which allow you to develop 3D models. It provides a very simple and easy to use platform which was good for us as we had very little modelling experience. We used this originally and created some models on it and exported them to unity.

Cinema 4D - Cinema 4D is a modeling platform allowing you to develop 3D models also. It is not open source and as a result one of our team members emailed them about our project and asked could we get a student educational version with which they obliged. We swapped to Cinema 4D as it is a much better model creator as you'd expect as it costs a lot of money to purchase a full licence. This was really helpful when editing the model we received from the team mentioned above and when we needed to swap model types to stop drifting.

Technologies used:

- Unity: Unity was used as the main platform to create the project and implement the SDK. It also creates the apk which allows the user for each device platform by compiling the project out to an apk format. We used this as we had previously had a module where we used unity and we were curious to learn more about it. Also it was the easiest way and was the way which had had the most support for helping us when we would need to research for help. We could have done the project using the unreal engine or in java, but opted for unity.

- Android Studios: Android studios was needed in order to download the different versions of android to be able to run the project on the proper version of android that the device needs to be run on.

- ARCore: The SDK used to carry out the augmented reality part of the project. The reason we chose this is outlined in the research part of the project.

- Blender/Cinema 4D - These were used in the creation of the model and editing of models as well as changing the formats that the model were saved in. These were important as we needed to edt models and save them in different formats to prevent drifting of models.

Methodology:

The software testing methodology that we used is agile. We used this to break the project up into what might be considered mini projects. Each week we would take an aspect of the project and focus on getting that done for that week. This helped a lot as rather than bouncing between different aspects of the project. We just focused on one aspect each week. Ensuring we had fully implemented that part of the project before moving to another part, which would leave that aspect with flaws had we not finished it. If we had been just moving in and out of different parts of the project, I'm sure there would've been areas where we would've forgot about and towards the end of our project would have left us struggling to finish the project due to disorganisation..

Issues:

Versioning- became a problem for us with creating an augmented reality application as many of the SDK's require the user to have a certain version of their specific hardware/operating system to run the application e.g. for ARCore the user of an Android device must have a later version of android than 7.0 (One member of our team had to purchase a new phone to carry out the project as we used ARCore and their current Device only was able to upgrade as far as android 6.0.1). For example, running ARCore on our team member's phone that had insufficient requirements resulted in the application launching with a purple screen. Another example is that, iOS users using ARCore require an ARKit compatible device running iOS 11.0 or later. Similar requirements are needed to run other SDK's on devices also.

Hardware- SDK's require that the device also has certain requirements to pass certification, components such as the camera, motion sensors, and the design architecture are tested to ensure that the device can run the application, also a powerful CPU is needed to integrate with the hardware design to ensure good performance and accurate real-time calculations are taken. Without the device being of a certain criteria the application may not work on that device.

Drifting- Drifting was a problem we experienced for a while throughout the creation of the project. Drifting can occur for may different reasons. By drifting, It's referring to the the model moving with the user and not staying in the position that it was originally placed in. In small occurrences of it, the model may move slightly towards the user as they move and then stop moving, we found this occurred a lot if the image had been scanned on an uneven surface or the image was scanned at an angle or if it was just a bad sacan of the image in general. This also occured in situations where the lighting in a room wasn't so good or a glare may have been on the image when scanning it. Overall this was a minor issue we had to deal with and wasn't going to cause an issue for our project as we intend the scan image to be mounted and placed in a well lighted area. The large drift movements were a big issue for us as this would cause the whole model to drift with user continuously. For example if the user entered the model, they would be unable to exit the model on their camera screen as the model would drift with the user on their screen even when walking long distances from the original placement. We were unsure what was causing this issue to begin with as some test models we used we not having this issue and other were having major issues with drifting. This became a big issue for us towards the end of the project as earlier in the project when testing we just avoided using models that were having this issue as we assumed it may have been that the assets weren't compatible with our type of project and just used the working ones. Then we we created our model for the hall, it had major drift issues. So we had to find a solution to this problem. The main issue we found that causes this, is the type that the model is saved as (e.g. .obj or .3ds etc….) So we experimented with different types of objects by using an external model creating software and exporting as different types until we found a model type which had no drift ultimately we used a .obj.

Transparency - Another issue which arose for us was transparency, by this I mean having one wall of the hall be transparent from one viewing point while not being transparent from another. We had hoped to allow users who are viewing the hall from the above railings to look inside of the hall and when they are inside of the hall when looking at the wall have it not be transparent. Although adding transparency was very easy to a wall, we struggled to find a way of making it so that it worked as we hoped. Through research we could only find a way of doing it in games e.g. 2D and 3D, but couldn't find a way to implement it into an augmented reality project. We used the user's current position relative to the building position but didn't work, as it needed to be just that wall of the building rather than the full building. Although If we had more time it is likely possible to implement.

Limitations:

Object disappearance - This was an issue of ARCore and other SDK's is that when a some circumstances occurs the object would disappear completely and need to be rescanned. An example of this occurs when the user's camera becomes covered when the application has scanned the image and spawned the model, the application loses track of the model and it needs to be rescanned. This also occurs if the users screen "goes to sleep". The only solution to solve both of these issues at once would be to have the model placed permanently on a cloud server using coordinates and spawn the model when user comes within a certain range. This would be an expensive alternative and the timeframe for the project was not sufficient to achieve such a solution.

Object movement - Sometimes a bad scan, scan of a moving target or a scan at a bad angle would also cause the object spawned to move itself in a direction. It would just fly rapidly off in a direction and get smaller and smaller. This issue mainly occurred if the scan was read really quickly when at an angle i.e. if the user moves onto the image fast and off again or maybe partially view the image quickly. Not a major issue as doesn't happen when an image is stable which for our project won't be an issue when set up in the hall.

View - When the application is running and the object has been spawned, the views can sometimes be a bit of a let-down when it comes to immersion. For example if a person walks across in front of you, through the windows of the building you will see them and not inside with you. Which when a person is stood directly in front of your camera would make no sense. As augmented reality is unable to place them inside the building as its merely a view on your camera. It makes sense that its incapable of doing this, but with how fast augmented reality is developing, it may not be long until this isn't an limitation anymore. Another issue with the view is that sometimes when viewing an object it appears to be slightly moving when going around it, though that is not the case.

ARCore - Although being hard to mark as a limitation due to its fantastic ability, It is only at the beginning of its lifecycle. ARCore was only first released in march 2018 making it at the current time just over a year old. When looking at how good it is now, we can only imagine how good it will become over the next few years. Our application may be rendered obsolete/out of date in comparison with newer updates which ARCore may release in a couple of years. If we look at another SDK in Vuforia for example, it has been around for years and is not as stable as ARCore is, in only one year of existence. Due to ARCore being so young in its lifecycle, it also leaves ARCore lacking in some areas where other SDK's have had years to perfect them. For example, Again in comparison with Vuforia, when developing you can see the target image's position in the unity engine. While ARCore uses a visualiser which is invisible to the developer which makes it much harder for spawning objects relative to the position of the visualiser. In our project we had to move the visualizer and objects relative to each other as the target image cannot be placed in the site as it is marked as a historical monument.

Versioning/Hardware- As mentioned in other sections hardware and versions of operating systems are causing major issues for augmented reality developers as their audience is minimised to those with good/more expensive devices. This results in less companies investing in this field and less competition, so therefore companies who are interested in

augmented reality are not being pushed to improve as much/as quickly, limiting the progression of augmented reality in general. Although with google taking an interest via ARCore this trend is slightly changing.

Testing:

For testing, we tested the project in many different environments (e.g. in direct sunlight on the camera and image, uneven surfaces etc...) with mixed results. As mentioned in the issues/limitations. Augmented reality is still in the early stages and if the environment is not ideal then the application may not perform perfectly. Although on a stable surface the application works almost flawlessly. Luckily when set up in the hall, as it is inside and the image target will be placed on a stable surface, we have no issues running the application from within the hall. But we still decided to test the application in areas to test how our application performs in less than ideal situations. Testing was also carried out consistently throughout to ensure that new implementations to the project, didn't cause new issues like drift etc… and if they did occur then those issues were fixed when possible.

Future Recommendations:

This style of application is one which would definitely be one to look at in the future, as the augmented reality is getting bigger and bigger and constant improvements are coming very frequently. If we were to do this application again, we'd certainly like a larger timeframe to do it, as we were so new to everything in this project, we spent a large amount of time just learning the basics. If we had the experience like we do now going into the application, I feel like we could've expanded even further and could've added much more but we were restricted due to the requirements we were given by the client as well as the learning process. we could've done a full tour of the hall but the client only wanted to limit us to just creating a augmented reality version of the hall alone as other groups were focusing on the other areas of the hall. We would definitely spend more time working on transparency views as mentioned in the issues, as with more time we feel we would've got this implemented. The project could be made even better just due to the progression of ARCore in the future. We could also try to use different SDK's such as Wikitude that has a very expensive licence. We could try to build the hall without the need to detect an image. This could be done using the scene recogniser by taking loads of images of the hall and upon entering the hall, the application would recognise the area without need of scanning a single image. We could also place the building on a cloud server using coordinates and when entering the hall spawn the building as mentioned above in other sections. Other SDK's could be researched into at a later date and see if they have implemented new features which may suit the project. A multi-SDK project could also be used as some SDK's are compatible with each other, unfortunately for us, all other SDK's have limited uses for users besides ARCore so as a result right now, it's the best to go for in our opinion.

Conclusions:

We are very pleased with the overall outcome of the project. At the beginning of our project when we were assigned this task, we were thinking that the project would be more of a research project with a small implementation to show as an example of what we could do given a longer timeframe. But instead we have a finished product which our client has stated

he is extremely happy with and wants us to be the main showing at an event later on in the year when the application launches on the app store. We started off knowing nothing and would now consider ourselves to be pretty competent at creating augmented reality applications and how they work and would be happy to do another application like this in the future. Due to the fact that we successfully completed our aims for this project and despite the issues and limitations of the software we still managed to produce an augmented reality application which surpassed the expectations of our client. The application uses the current technologies available to us to create an aesthetically pleasing, stable and immersive augmented reality experience and as a result we would say the project was a success.