Technical University of Crete

School of Electrical and Computer Engineering

PentAgRam: A Collaborative Augmented Reality Application For Music Composition

Dimitris Marinopoulos

Chania|Athens , May 2021

Abstract

This thesis presents a mobile Collaborative Augmented Reality application for music composition. The main focus of this implementation is to enable the remote collaboration of users on an Augmented Reality environment. This thesis acts a prototype for remote collaboration. Users are able to cooperate from anywhere in the world in order to interact with a same virtual collaboration space while on different physical areas. In this setting, users can coordinate and compose a music piece in real time. The challenge of this thesis was the technical implementation of Collaborative Augmented Reality, as there is little prior work on this field. Our implementation begins by the detection of the users' vertical physical surroundings, which then creates a shared virtual pentagram that each user can work on. The users are able to start composing by adding and removing notes on the shared virtual pentagram. The composition continues while both users are connected and new pentagrams are spawned on the shared virtual space when necessary. The Music Notation System we used on our approach composes of the treble clef, clear notes and the 'four quarters' time signature.

This application was developed using the Unity3D platform for the development of 3D interactive applications, as well as, the required programming tools in order to achieve a mobile Collaborative AR experience based on the AR Foundation API. AR features such as, plane detection, precise 3D model spawning, device tracking etc were implemented. More over, in order to implement a stable connection between the different users and to ensure a smooth collaboration, the Photon Unity Networking (PUN) was imported and further developed. As a result, we managed to establish a much needed real-time positional synchronization between the user's AR models and interfaces. By combining music composition and collaboration through Augmented Reality, we propose an innovative way to create a music piece, that includes the physical surrounding of the users which is digitally enhanced and enables the collaborative aspect of Augmented Reality.

Acknowledgements

Initially, i would like to express my sincere gratitude to my supervisor associate professor Aikaterini Mania for her advice and motivation throughout the whole development of the thesis. Even after the start of the pandemic of Covid-19, her continuous remote support played a fundamental role in overcoming this project's difficulties.

Secondly, i would like to thank my colleagues from the SURREAL Lab for their valuable ideas and useful instructions. Specifically, i would like to thank Yiannis Kritikos and Grigoris Daskalogrigorakis for their time and their contribution.

Afterwards, i would like to express my love and my gratitude to my friends who mentally supported me during the composure of this thesis. Each one of them contributed in their own way through all these years of friendship. I cant thank enough Chris Stavgiannoudakis for his guidance and his infinite patience and Katerina Voukelatou for her daily assistance.

Finally, i would like to deeply thank my family members: My father Thodoris, my mother Katerina and my litle sister Eirini for their limitless and selfless support. Without them and their relentless efforts i would not have reached my goals.

Contents

List of Figures

Chapter 1

Introduction

Music is the art of arranging sounds in time to produce a composition through the elements of melody, harmony, rhythm, and timbre. It is one of the universal cultural aspects of all human societies. Various systems of music's visual representations were formed through the years, defining the musical notation. The most known is the standard notation on 5-line staves which now days can be seen not only in papers but on every computer - mobile.

Augmented Reality (AR) has been getting into our lives steadily the last decade. The rapidly growing smartphone market contributed the maximum at AR's popularity. Since its first appearance, from the first AR head-mounted display to AR glasses and smartphones, mobility and accessibility has increased dramatically. At the same time, engagement and interaction with the real world provides a richer user experience. Thus, AR is no longer a laboratory technology, as it is used to a great extent by a number of different fields (gaming, weather, architecture, etc.). However, despite the wider applications of AR in different areas, music and composition application using AR technology are very rare.

This thesis focuses on a multiplayer AR experience for music composition. Collaborative Augmented Reality introduces a new experience for the AR users, who not only interact with the real and virtual world but with each other as well. That enables two types of collaboration: Face to face and remote.

1.1 Brief Description

In this thesis, we present the development of a mobile remote Collaborative AR application for music purposes. Two users in different areas are composing into the same pentagram, using the walls as paper and their mobile phone screen as a pen.

In more recent applications, the AR industry introduces multiplayer experiences, allowing users to collaborate with each other either face to face or remotely. This aspect of AR creates an enchanted, with 3D models, physical world. Multi-users are able to move around, making use of different angles of the models, examine every dimension and at the same time communicate.

This thesis aims to investigate previous single-player and multi-player AR applications and introduce this innovative technology of Collaborative AR into music. Users have the opportunity to compose together at the same virtual pentagram in the physical world. A note's model can be spawned on the virtual pentagram by a single tap on our mobile phone's screen, creating a music piece. In addition, new musicians can use this application along with their teachers. In this educational aspect, teachers are able to explain the visualised notes and their values to their students, into an AR environment while sounds will make the process interactive with the real world.

In more detail, this application was developed in Unity3D game engine using PUN Photon Networking Engine and Unity's package AR Foundation. AR Foundation was introduced by Unity's developers and it allows to work with augmented reality platforms within Unity, using ARCore and ARKit SDKs (Android , IOS). Photon Engine allows us to connect between two mobile phones and interact with each other. Our application starts by scanning the area for vertical objects such as walls, where our pentagram is spawned. Using PUN's features the first user creates a room where the second one joins. By the time both users are ready and the pentagrams have spawned, composition begins. At this phase, both users can add and delete notes at the same time affecting both pentagrams. This remote interaction between them leads to a unique composure which can be listened and edited any time during the composing procedure.

1.2 Structure of the Thesis

This section provides the detailed structure of our thesis. There are six chapters that are set out as follows.

Chapter 1 introduces a brief description of music composition and Collaborative Augmented Reality. It presents the current state of the AR and the reasons we immersed it with music.

Chapter 2 presents the research made around the AR fields, the history of Augmented Reality and the Collaborative additions. Furthermore, the chapter explores the AR Hardware Devices and AR Software Development Kits available. It introduces the reader to the scientific field of AR and the well known applications around it, while focusing on mobile based applications. The chapter continues with the AR and non-AR music applications and the way they inspired our thesis, and finally, presents the research between game engines and the reason we decide to use Unity.

Chapter 3 contains basic music information and knowledge, focusing on music composition. It offers the opportunity to the reader to learn about the history of music and the rules of music composition. Definitions such as pentagram and notes values are getting clarified regarding the way they have been implemented in our application.

Chapter 4 provides a presentation of our application from the user's view. Functional directions have been provided as a guide in order for the user to navigate through the application. Every scene is presented and analyzed until the composition is finalized.

Chapter 5 includes the full implementation process along with the technological terminology of Unity Game Engine. It refers to every software tool has been used into our application's structure. This is the most detailed chapter in our thesis since it covers every aspect of the development procedure.

Chapter 6 summarizes our thesis and the reflects upon the experience gained by this implementation. We present the evaluation method of our application, the feedback we have gained during this time and we list the known issues. Finally, we propose future improvements and ideas for our thesis and the role of music into Augmented Reality in general.

Chapter 2

Research Overview

2.1 Introduction

This thesis is a result of continual research on various topics. The main topics of this research includes Augmented Reality, music composition and Game Engines. In more detail, we tried to cover and understand AR technology's field in depth so the history of AR had to be researched. Since we developed a Collaborative AR application, it was critical to define AR without omitting the multiplayer aspect. It is also of importance to highlight the existing AR Hardware and AR SDKs. Furthermore, AR experiences formed a great topic of our research in order to understand AR implementations. In this chapter we also present our research for music applications, AR and non-AR. This topic is quite niche so we had to search for every application in order to realize how music composition can be accomplished through them. Finally, our application had to be developed into a Game Engine, so we introduced their advantages and disadvantages and what lead us to Unity3D.

2.2 Augmented Reality

2.2.1 Introduction to AR

Augmented Reality (AR) is the technology that allows overlays from computer graphics to blend with information that is input from the real world. During an AR experience users interact with the real-world environment which, at the same time, is enhanced with computer-generated elements (visual, auditory etc). There is often a misunderstanding between Augmented and Virtual reality. In order for this to be avoided, we need to mention that VR allows a person to interact with an artificial three-dimensional environment using electronic devices. In contrary to AR, VR refers exclusively to a computer generated virtual world.

The entire AR process is a result of three main steps:

- **Recognition** of an image or object
- **Tracking** of an image or object in space
- **Mixing** of the virtual media with real-world image or object by superimposition.

During the first step, sensors and cameras of the device recognize any object or space in the real-world. This data will be used in the second step, tracking, in order to find the place on which the virtual object will be superimposed. Finally, the mixing of virtual and real world takes place and virtual objects spawn in form of video, 3D and 2D models, etc.

2.2.2 Brief history of AR

Augmented Reality seems to be at the very first steps of its era. A great amount of well developed applications has already been published the last few years and it is safe to assume that there are more to come. As the technology grows rapidly, new 3D graphics are being introduced and more portable new devices with great computational power are being developed which makes AR's potential seem endless. Despite the fact that AR has gained its reputation recently, the technology was introduced approximately 50 years ago.

In 1968, Ivan Sutherland, dubbed by some "The Father of Computer Graphics" created the first Head Mount Display, called "The Sword of Damocles". It uses an optical see-through head-mounted display that is tracked by one of two different 6DOF trackers: a mechanical tracker and an ultra sonic tracker. Due to the limited processing power of computers at that time, only very simple wire-frame drawings could be displayed in real time.[1]

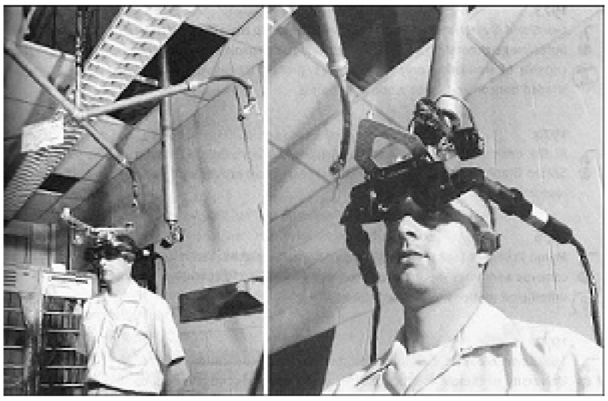


Figure 2.1: Ivan Sutherland: "The Sword of Damocles"

In 1992 the term "Augmented Reality" was first introduced by Tom Caudell and David Mizell to refer to overlaying computer-presented material on top of the real world. Caudell and Mizell discuss the advantages of AR versus VR such as requiring less processing power since less pixels have to be rendered. They also acknowledge the increased registration requirements in order to align real and virtual. [2]

Later on 1994, Paul Milgram and Fumio Kishino write their seminal paper "Taxonomy of Mixed Reality Visual Displays" in which they define the Reality-Virtuality Continuum. Milgram and Kishino describe a continuum that spans from the real environment to the virtual environment. In between there are Augmented Reality, closer to the real environment and Augmented Virtuality, which is closer to the virtual environment. Following their lead , Ronald Azuma presents the first survey on Augmented Reality.In his publication, Azuma provides a widely acknowledged definition for AR, as identified by three characteristics:

• it combines real and virtual
• it is interactive in real time
• it is registered in 3D.
Today Milgram's Continuum and Azuma's definition are commonly accepted as defining Augmented Reality.[3] [4]

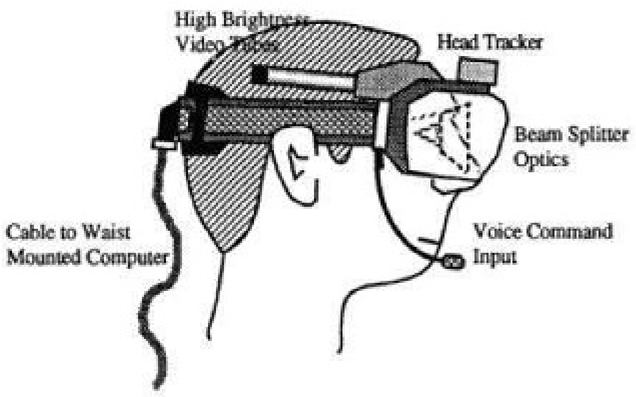


Figure 2.2: Caudel and Mizeli coining AR

Older technologies consisted of Head-Mounted Displays(or HMDs), eyeglasses or contact lenses that showed virtual objects in front of the user's eyes. This posed a multitude of problems because the tolerance of the technology was low when tracking sudden movements of the user and the precision of the then available instruments could not match it, thus users experienced frequent nausea and disorientation.

The aforementioned issues lead AR applications from HMDs to handheld tablets and smartphones. As users get distanced from the virtual screen, graphics are more visible into the real world and the experience becomes more interesting. Furthermore, via the portability that handheld devices provide, users are able to navigate through the real world and the application becomes way more interactive. The first Mobile AR System was introduced in 1997, by Steve Feiner. "Touring Machine" uses a see-through head-worn display with integral orientation tracker; a backpack holding a computer, differential GPS and digital radio for wireless web access; and a hand-held computer with stylus and touch-pad interface.[5]

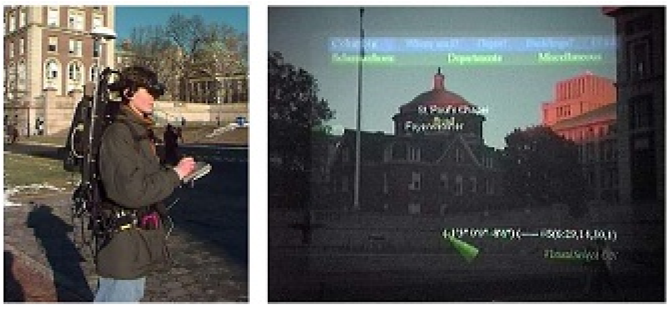


Figure 2.3: Steve's Feiner: "Touring Machine"

The AR's era as we know it today, started at the end of the 20th century with the presentation of "ARToolkit" by Hirokazu Kato and Mark Billinghurst. ARToolKit is a pose tracking library with six degrees of freedom, using square fiducials and a template-based approach for recognition. ARToolkit uses a simpler marker-based approach, with additional support for Natural Feature Markers. ARToolkit tracks the markers while in view, calculating which markers are in view, their orientation relative to the camera as well as their depth from the camera.The most commonly used markers are 2D bar-codes and 3D boxed pattern bar-codes which can often look out-of-place if they are not hidden correctly. ARToolKit is available as open source and is still very popular in the AR community.[6]

Approximately at the same time, Gaming industry joins into the development of AR applications. In 2000, Bruce Thomas et al. present AR-Quake, an extension to the popular desktop game Quake. ARQuake is a first-person perspective application which is based on a 6DOF tracking system using GPS, a digital compass and visionbased tracking of fiducial markers. Users are equipped with a wearable computer system in a backpack, an HMD and a simple two-button input device. The game can be played in- or outdoors where the usual keyboard and mouse commands for movement and actions are performed by movements of the user in the real environment and using the simple input interface. A great amount of AR game has been developed till then and they are about to be mentioned at the upcoming sections of our thesis, since they formed great inspiration for us. [7]

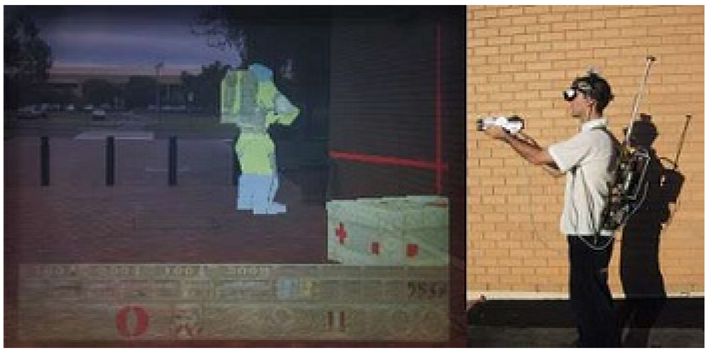


Figure 2.4: ARQuake: The first AR game

Later on 2004, while the Internet has started becoming a daily thing for humanity, the first multiplayer AR application is shown at SIGGRAPH 2004 Emerging Technologies, called the ”Invisible Train”. The Invisible Train is the first real multi-user Augmented Reality application for handheld devices (PDAs). Unlike other projects, in which wearable devices were merely used as thin-clients, while powerful (PC-based) servers performed a majority of the computations (such as graphics rendering), the software runs independently on of-the-shelf PDAs - eliminating the need for an expensive infrastructure.[8] Early on 2005 , following Invisible Train's lead Anders Henrysson ports ARToolKit to Symbian. Based on this technology he presents the famous AR-Tennis game, the first collaborative AR application running on a mobile phone. ARTennis was awarded the Independent Mobile Gaming best game award for 2005, and the technical achievement award.[9]

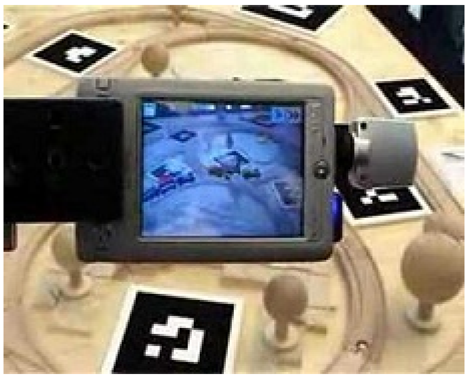


Figure 2.5: Invisible Train

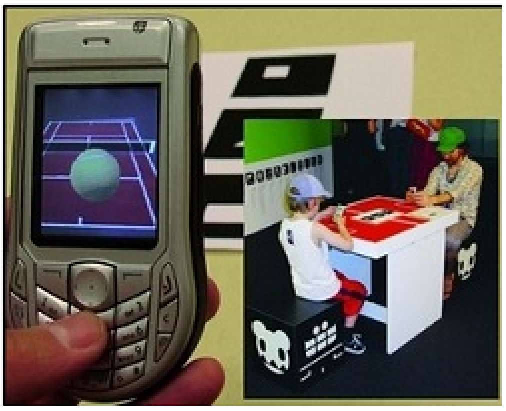


Figure 2.6: AR-Tennis: Best game award for 2005

Through the next years a great amount of honorable AR applications have been developed, making it nearly impossible to mention them all. In this thesis, we focus and we develop a Collaborative AR application so the last two projects are a major landmark for our research.

2.2.3 Introduction to Collaborative AR

Augmented Reality was initially introduced as a new technology well-known for its interactivity. The Collaborative AR approach enhances interactivity, since it allows multi-users to simultaneously share a real and a virtual world while maintaining the interaction with each other. In contrast with the existing collaborative technologies, with collaborative AR, physical objects are more than a source of information. Furthermore, with other collaborative applications, communication has been compartmentalised since they either lack visual content (audio apps) or spatial cues (video chats). Thus, the potential of creating new collaborative interfaces that will change our daily life, makes multi-user AR technology radical and is starting be prioritized among the developers and the industry.[10]

The entire AR collaborative process can be separated in two categories:

- Face-to-face collaboration
- Remote collaboration.

Face-to-face collaboration requires physical presence by both users During that process, people use speech, gestures and gaze as the AR application runs. Every possible object, enhances the AR environment and blends on both user interfaces simultaneously. [11]

Remote collaboration gives the ability to its users to engage from anywhere in the world. During that process multi-users of AR applications may not be sharing the same physical area but only a virtual one. Despite that, every user is able to edit in real-time each other's environment using 3D models etc.[12]

In this thesis we choose to develop our AR application using Remote Collaboration. One of the first reasons for this choice was that music is a universal art and it doesn't require physical presence in order to be composed. Covid's-19 pandemic situation had a key role on our decision, since social distancing became a necessity.

2.3 AR Hardware Devices

Augmented Reality was introduced to enhance the real world with computer-graphics. Therefore, special hardware devices have been used or created through all these years. Wearable glass masks, head mounted displays and smartphones are just a few examples which can provide AR experiences to the user. AR masks and HMD's tend to standalone exclusively for AR. On the other hand, quite a few Software Development Kits have been developed to improve smartphone's AR applications.

In our research we separate the hardware devices into two categories: i) Head-Mounted Devices , ii) Mobile Devices. A short presentation of both of the categories, along with the reasons that lead us to choose smartphones for our implementation is given further in the thesis.

2.3.1 AR Head-Mounted Display

Head-mounted displays (HMDs) are small displays or projection technology integrated into eyeglasses or mounted on a helmet or hat. Depends on the number of the display optics (one or two) HMDs are grouped into Monocular and Binocular. A typical HMD has one or two small displays, with lenses and semi-transparent mirrors embedded in eyeglasses (also termed data glasses), a visor, or a helmet. The display units are miniaturized and may include cathode ray tubes (CRT), liquid-crystal displays (LCDs), liquid crystal on silicon (LCos), or organic light-emitting diodes (OLED). Some vendors employ multiple micro-displays to increase total resolution and field of view. Another type of HMD is the optical one which is using and optical mixer that is made out of partly silvered mirrors.It can reflect artificial images, and let real images cross the lens, and let a user look through it.

HMDs are being used both on Augmented as in Virtual Reality. The difference between them are whether they can display only computer-generated imagery (CGI), or only live imagery from the physical world, or combination. Most HMDs can display only a computer-generated image, sometimes referred to as virtual image. Some HMDs can allow a CGI to be superimposed on real-world view. On those HMDs we mainly interested in, since they are the Augmented Reality HMDs.

AR HMDs are expected to see a rapid growth into Augmented Reality's industry. Using Virtual's Reality masks as an inspiration and compining them with the spatial scanning technologies, AR HMDs has been formed to create a new standard for the AR research. The potential of this product has lead great companies as Microsoft and Google to invest in order to massively produce them. The new era of HMDs introduce us to a fully 3D visualised environment where the standard computer functionality is integrated into a mask. However these devices are still quite new and the innovative form that they have, requires much more optimization and improvement until they become stable. Below we present some honorable AR HMDs that is believed to be the most widespread during this period.

Microsoft HoloLens

Microsoft HoloLens: developed by Microsoft back in 2016. One of the first AR HMDs to be announced and sell their prototypes, although in limited editions. Microsoft's company tried to integrate Windows in AR environment using Kinect's tracking technology.

Packed with the processing power of an average laptop and a multitude of sensors Hololens aims for precision tracking and world scanning around the user. On the software side, it uses an optimized version of the already trained and tested Microsoft Kinect's Neural Network for tracking and environmental scanning. Microsoft highlights numerous industries for which the HoloLens is suitable, from manufacturing and retail to healthcare and education. This self-contained, holographic computer allows for increased productivity, collaboration, and 3D design.[13]

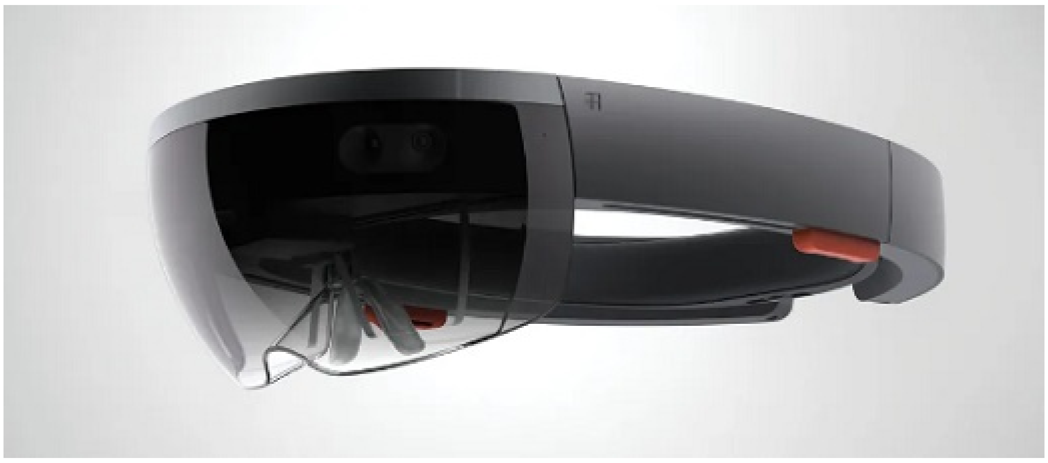


Figure 2.7: Microsoft HoloLens

Microsoft HoloLens II

Microsoft HoloLens: developed by Microsoft back in 2019. It was released to upgrade the experience of its "ancestor", Microsoft HoloLens. There is still no wide use of the product as it used only for researches and for authorised customers.

HoloLens II headset achieved some great improvements compared to its predecessor, such as : i) Processing power: the HoloLens 2 is more powerful than its predecessor. ii) Field of View (FOV): at 52 degrees, the HoloLens 2's field of view is larger, offering a more immersive AR experience for the user. The original HoloLens only has an FOV of 30 degrees. iii) Battery life: the HoloLens 2 features a 3-hour battery life, while the HoloLens 1 has a 2.5-hour battery life. 4) Design and fit: according to Microsoft, the HoloLens 2 offers a lighter and more ergonomic fit than the original HoloLens. The HMD also now features a flip-up visor which allows users to enter/exit augmented reality more quickly.[13]

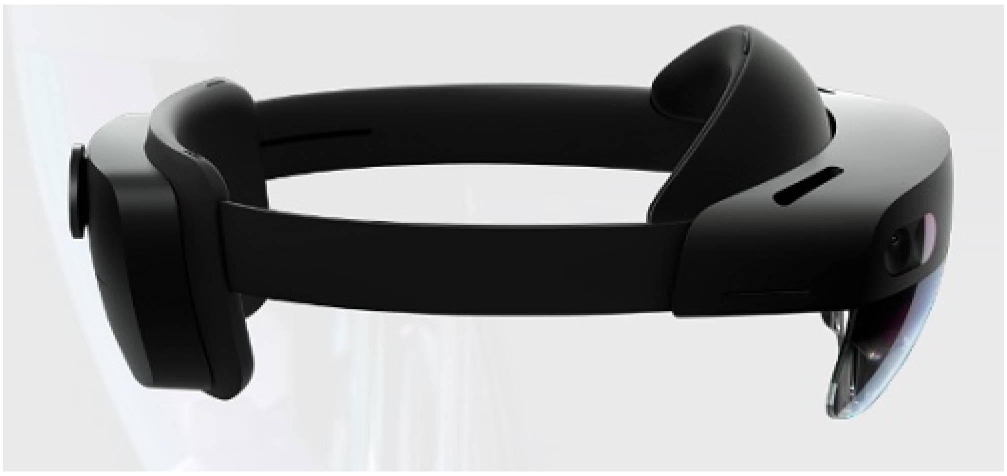


Figure 2.8: Microsoft HoloLens II

Magic Leap One

The Magic Leap One is a popular standalone AR headset made by Magic Leap, Magic Leap One uses Machine vision to thoroughly scan the environment around the user and make virtual objects context sensitive to the world around them. Furthermore, virtual objects are not only visually immersive but also use spatial audio with increasing depending on the distance from virtual objects.

Magic leap one introduces us to more features such as advanced eye-tracking. Its technology allows the device to know where the users are looking and track their gaze, even with blink services. Magic leap differs from every other HMDs since its provides gesture controls and voice commands as well. In addition, Magic Leap presents something more than a HMD since the masks comes with a steero head-set. All the processing power resides in a tethered Lightpack, which is a trademarked high-powered processing and graphics portable mini PC.It also comes bundled with a controller with 6-DoF (Degrees of freedom) of movement called Magic Leap Control.[14]

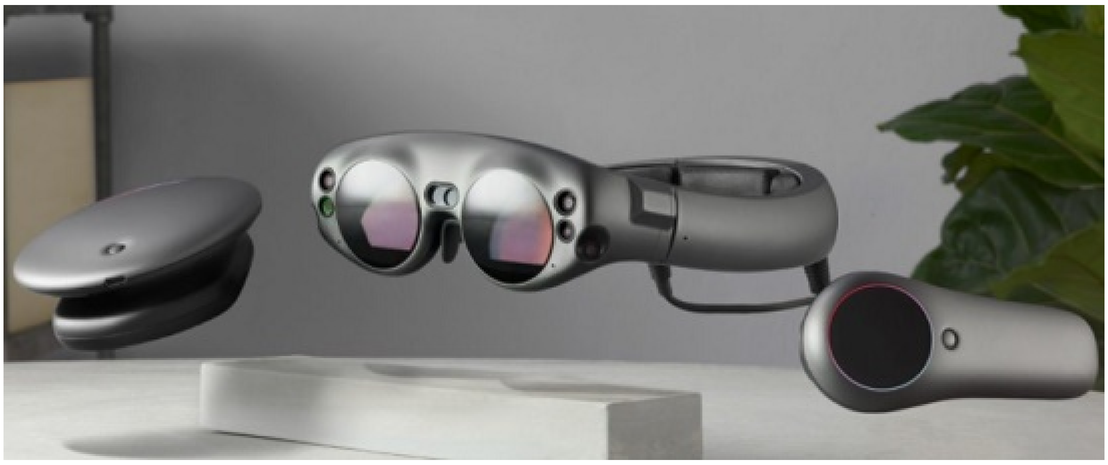


Figure 2.9: Magic Leap Mask, Lightpack and Control

2.3.2 Mobile Augmented Reality

Mobile Augmented Reality introduces AR technology into the widely growth industry of smartphones and tablets. AR takes advantage of those powerful platforms that have currently been developed in order to fulfill its great potential. At this point, the distinction between a portable and a mobile AR device needs to be mentioned. Portable AR devices, such us laptops, can be moved from place to place relatively easily. In our case though, smartphones and tablets are completely mobile devices, that are being carried on and used daily. The given lightweight and their pocket size helps AR applications blend easily in our life.

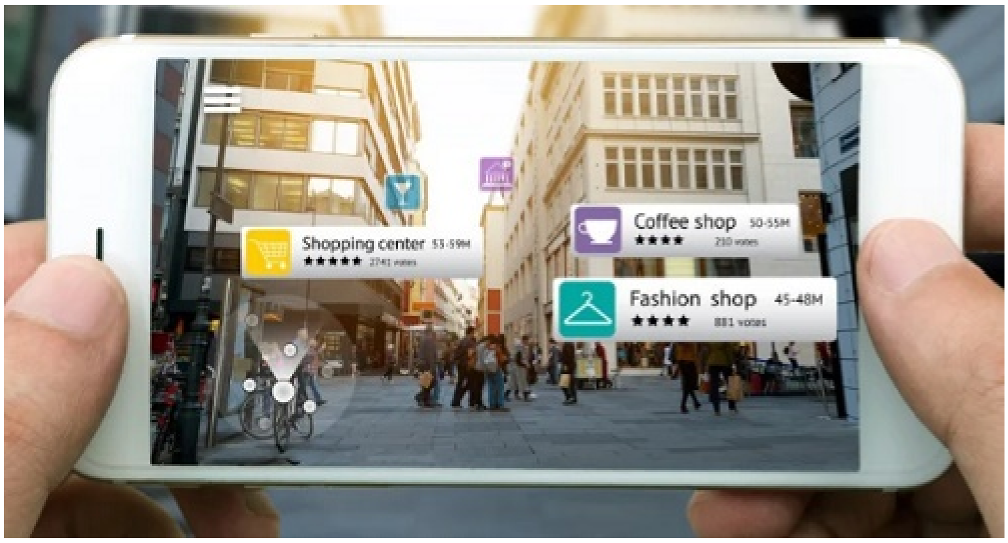


Figure 2.10: AR infos into a City

Smartphones and tablets have been introduced to us with innovative technologies, which are still getting upgraded, such us mobile processing, image recognition, object tracking, display technology and GPS tracking. AR's hardware requirements have been completely covered with everything being said. As an outcome, mobile devices provide a suitable environment for AR applications. In addition, compared to every other standalone AR device which has to be bought, smartphone's cost is significantly less.

AR apps utilize the sensors and cameras already present in modern computers or, more commonly, smartphones in order to gather information from the real world and allow virtual graphics to blend into the natural environment seen through a camera.[15] In order to develop an AR application, developers frequently use a pre-built Software Development Kit, or SDK for short, which provides them with pre-made tools useful for AR. These tools vary from automating simple jobs, like setting up a new AR scene, to complicated algorithms like using Machine Vision to scan the environment and extract data like marker detection.

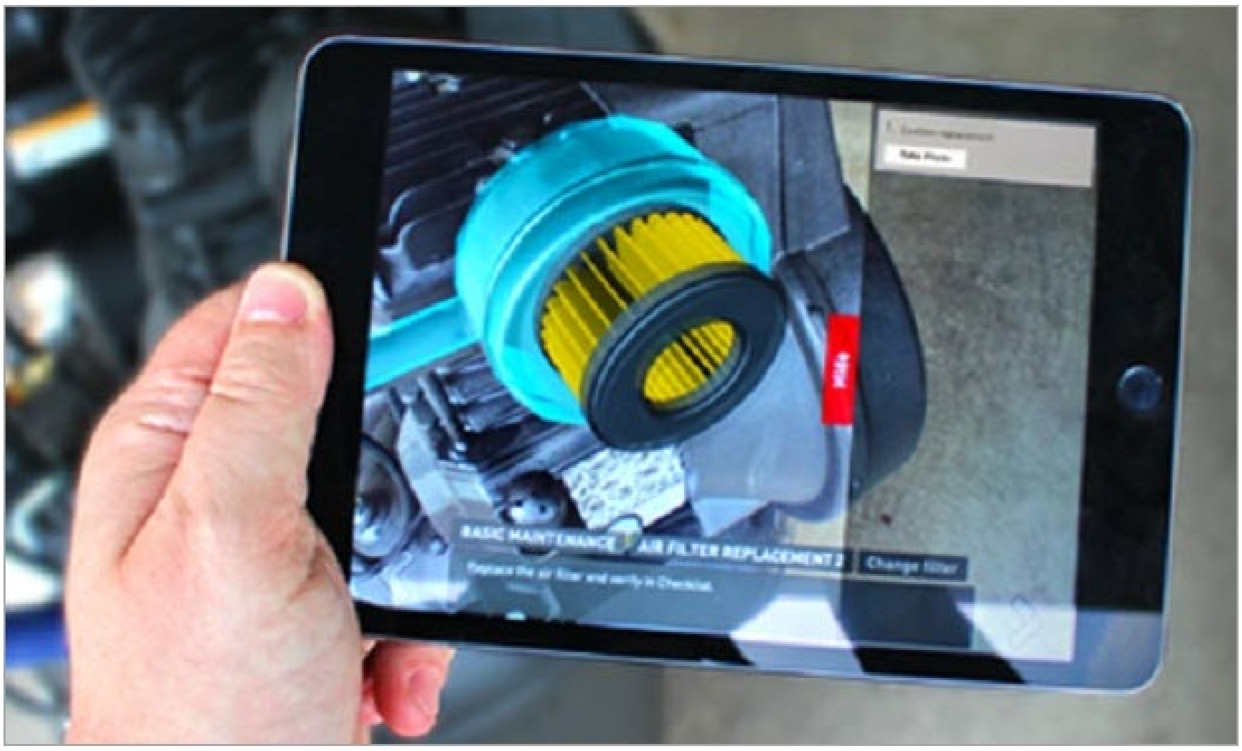


Figure 2.11: AR app for Car Service

Augmented Reality Software Development Kits (2.4) equip developers with a great amount of libraries and development tools making the build process easier and faster. In more details, an AR SDK includes the three main steps of Augmented reality process, as they have been mentioned at 2.2.1 subjection of our thesis: AR recognition, AR tracking, AR mixing-rendering.[16] A list of tools, such as AR cameras, AR plane detection, AR markers and many more are given to the developers in order for the main steps to be fulfilled. Augmented Reality SDKs can be organized in these broad categories: Geo-located AR Browsers, Marker based, Natural Feature Tracking. AR Browser SDKs allows users to create geo-located augmented reality applications, using the GPS and IMU available on today's mobile and wearable devices. Marker based SDKs employ special images, markers, to create augmented reality experiences. Natural Feature Tracking SDKs rely on the features that are actually present in the environment to perform the augmentation by tracking planar images, based on a SLAM (Simultaneous Location and

Mapping) approach.

In our thesis, mobile AR technology is being implemented and followed. Firstly, as we have already mentioned smartphone approach provides us an easily accessible way for every user. HMDs on the other hand aren't so popular and they require a decent amount of money in order to be obtained. Furthermore, we focus on smartphones since collaboration between two users wouldn't be possible with HMDs, as the existed SDKs don't support those features.

2.4 AR Software Development Kits

Augmented Reality Software Development Kit is a collection of software development tools which can be used to develop AR applications for different hardware devices. The existing number of SDKs varies from device to device. We are developing our application in Unity, so we imported the ARFoundation package. ARFoundation presents an interface for Unity developers to use, in order to work with augmented reality platforms. For target devices such as Android and iOS, ARFoundation requires their SDKs, ARCore and ARKit, to be imported.

2.4.1 ARCore

ARCore is a Google developed SDK for building AR experiences for Androids.[17] Some of the APIs are available across Android and iOS to enable shared AR experiences. ARCore uses three key capabilities to integrate virtual content with the real world as seen through your phone's camera:

• **Motion tracking** allows the phone to understand and track its position relative to the world.

• **Environmental understanding** allows the phone to detect the size and location of all type of surfaces: horizontal, vertical and angled surfaces like the ground, a coffee table or walls.

• **Light estimation** allows the phone to estimate the environment's current lighting conditions.

Fundamentally, ARCore is doing two things: tracking the position of the mobile device as it moves, and building its own understanding of the real world. ARCore's motion tracking technology uses the phone's camera to identify interesting points, called features, and tracks how those points move over time. With a combination of the movement of these points and readings from the phone's inertial sensors, ARCore determines both the position and orientation of the phone as it moves through space. In addition to identifying key points, ARCore can detect flat surfaces, like a table or the floor, and can also estimate the average lighting in the area around it. These capabilities combine to enable ARCore to build its own understanding of the world around it.

ARCore's understanding of the real world lets you place objects, annotations, or other information in a way that integrates seamlessly with the real world. You can place a napping kitten on the corner of your coffee table, or annotate a painting with biographical information about the artist. Motion tracking means that you can move around and view these objects from any angle, and even if you turn around and leave the room, when you come back, the kitten or annotation will be right where you left it.

2.4.2 ARKit

ARKit is a Apple developed SDK for building AR experiences for iOS.[18] ARKit is an augmented reality framework included in Xcode that is compatible with iPhones and iPads. ARKit lets developers place digital objects in the real world by blending the camera on the screen with virtual objects, allowing users to interact with these objects in a real space.

ARKit uses a technology called Visual Inertial Odometry in order to track the world around the iPad or iPhone. Using the iOS device's camera, accelerometers, gyroscope and context awareness, ARKit performs environment mapping as the device is moved. Sensor fusion of the inertial sensor data with the data from the camera allows for highly accurate location awareness and mapping. The software picks out visual features in the environment such as planes and tracks motion in conjunction with information from the inertial sensors. The camera is also used to determine light sources by which AR objects are lit. Apple's solution to the increased detail and therefore memory usage is a sliding map where old data disappears for new. Users can place anchors to mark creations they want to save.This enables the iOS device to sense how it moves in a room. The user doesn't have to do any calibration, that's a breakthrough in this space.

2.4.3 Other SDKs

The aforementioned SDKs are part of the new era of AR SDKs, their use is massive and their developers are supporting them consistently with new features. Before the wide growth of those two SKDs, Vuforia and Wikitude were among the most used.

Vuforia is an augmented reality software development kit (SDK) for mobile devices that enables the creation of augmented reality applications. Originally developed by Qualcomm, it has been acquired by PTC Inc. The Vuforia SDK supports a variety of 2D and 3D target types including 'markerless' Image Targets, 3D Model Target, and a form of addressable Fiducial Marker, known as a VuMark.[19] Additional features of the SDK include 6 degrees of freedom device localization in space, localized Occlusion Detection using 'Virtual Buttons', runtime image target selection, and the ability to create and reconfigure target sets programmatically at runtime.

Wikitude is a mobile augmented reality (AR) technology provider based in Salzburg, Austria. Founded in 2008, Wikitude initially focused on providing location-based augmented reality experiences through the Wikitude World Browser App.[20] In 2012, the company restructured its proposition by launching the Wikitude SDK, a development framework utilizing image recognition and tracking, and geolocation technologies.

2.5 AR Experiences Research

Before we started developing our thesis, a significant amount of time was spent researching different AR applications. The initial focus was on modern Single-player AR applications, since the approaches of the developers of the industry are constantly changing This resulted in acquiring critical knowledge for the current state of AR. The second topic we had to research was about the multi-user feature into AR. While there was far less documentation available for multi-user applications, it was crucial in the development of our thesis.

2.5.1 Single-player AR Experiences

• IKEA Studio App - WAIVER: 'View in Room'

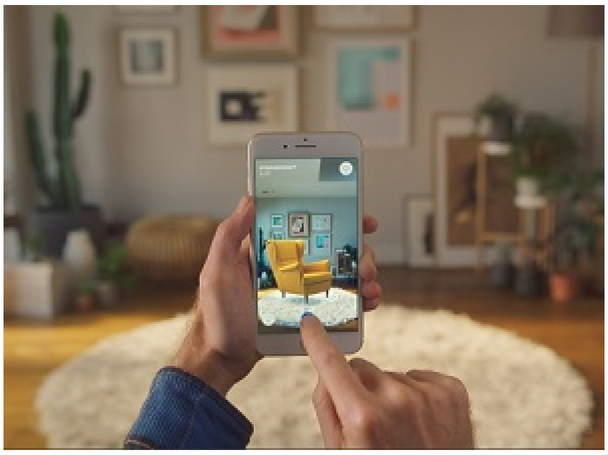


Figure 2.12: IKEA Studio Application

Both furniture retailers have heavily invest in Augmented Reality by releasing two innovative augmented reality application. Those applications are focusing on home decor. They provide a brand new feature to their customer where virtual furniture can be dragged directly to their room.

LiDEAR technology has been implemented in both applications enabling users to capture entire 3D room plans and re- designed them. In that way, customers are able to change the lighting among with the wall colours in order to realistically test new furniture. In addition, they can stack products everywhere just like the real world. Researchers say that users enjoy the given opportunity and they are most likely to buy a furniture that has been already tested into their room.

- ADIDAS - more virtual sneakers

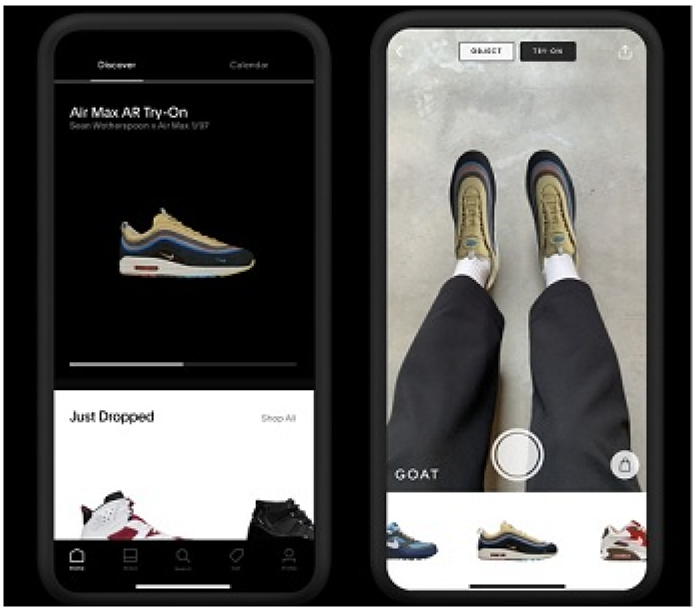


Figure 2.13: More virtual sneakers

Adidas was among the first brands that introduced the try-on AR applications. Covid-19 hit the whole world after a few months, making the number of similar applications increase.

In November 2019, Adidas added the feature to its iOS app, helping shoppers to decide on a purchase without ever entering a store – something that was soon to become unavoidable for all consumers. Created in partnership with computer vision platform, Vyking, the AR app tracks foot movements, enabling users to see how sneakers look on their feet in real-time, with or without shoes.

- Toyota Hybrid AR

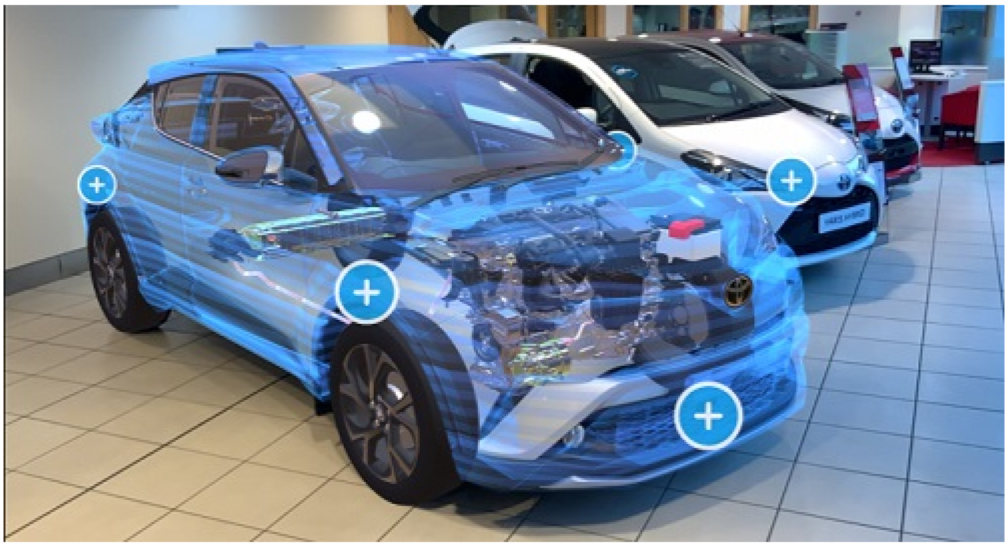


Figure 2.14: Toyota Hybrid AR

The Toyota Hybrid AR application uses Augmented Reality technology and objective recognition software to overlay graphics of the inner workings of the Hybrid drivetrain onto physical vehicles, helping customers to gain a better understanding of how the system works.

The 3D experience also features a number of ‘hotspots’ which, when clicked on, enable customers to get in-depth information on key features of the system, such as the motor, battery and fuel tank. Brandwidth worked with Toyota to develop the Hybrid AR app as part of an ongoing drive to educate the brand’s customers on hybrid technology and the benefits it offers. The app currently supports the Toyota C-HR model, across all grades and colours.

- Google Translate

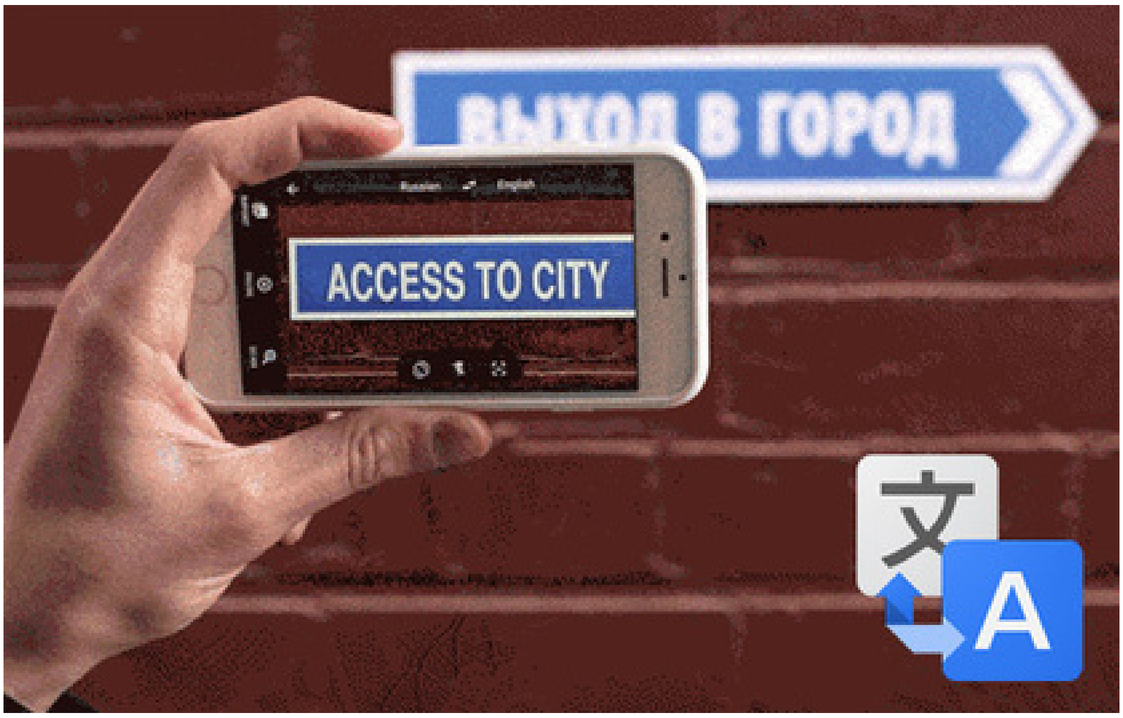


Figure 2.15: Google Translate AR

Google's Translate app is one of the most useful applications of augmented reality technology so far. It can translate text in an image from one language to another, allowing you to read signs, packaging, and even memes in other languages. Just open the app, capture the text, and wait for the translation. It also includes more pedestrian translation tools, which are the best free tools available.

2.5.2 Multi-player AR Experiences

- Just a Line

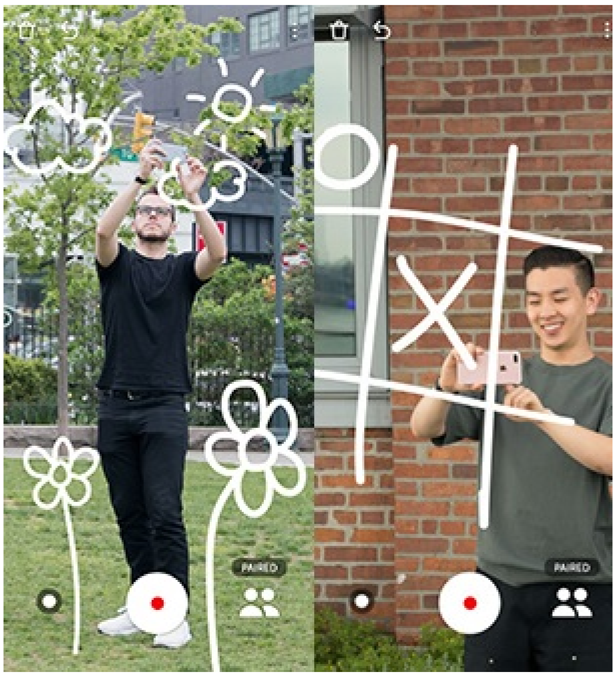


Figure 2.16: Just a Line

Just a line is a Face-to-Face Multi-Player application designed for Experiments by Google. In this project, users are able to pair with their friends, who share the same space, and make simple drawings in Augmented Reality. Once they finished the drawings they are able to share their creation in a short video. Just a Line is developed with ARCore Cloud Anchors , which allows users to collaborate between Android and iOS.[21]

- Pokemon GO

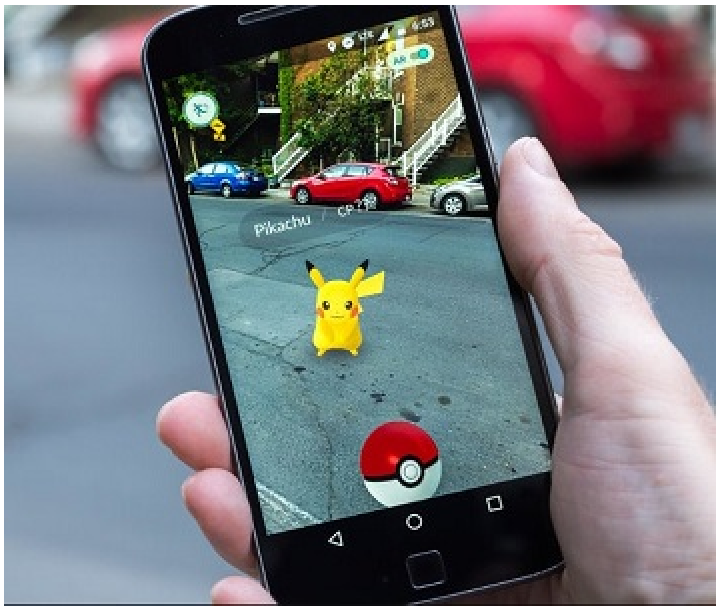


Figure 2.17: Pokemon GO

Augmented Reality has been introduced in gaming industry with a great amount of applications. Pokemon GO is one of the most popular and the one that counts the most users. Pokemon GO is a free Face-to-Face and Remote Augmented Reality application, developed by Niantic and published by The Pokemon Company.

Based on the old Pokemon games, GO, blends old features into the real world. Users firstly create an avatar of their self and then the hunt begins. The application is setting up user's current location and letting him know about the closest spawns of Pokemons. Special events are taking place as well, in popular locations all over the world, such as galleries, memorials, museums, etc. Furthemore,the real world becomes a field where trainers are able to fight each other using their own Pokemons.

- Nissan AR Multiplayer

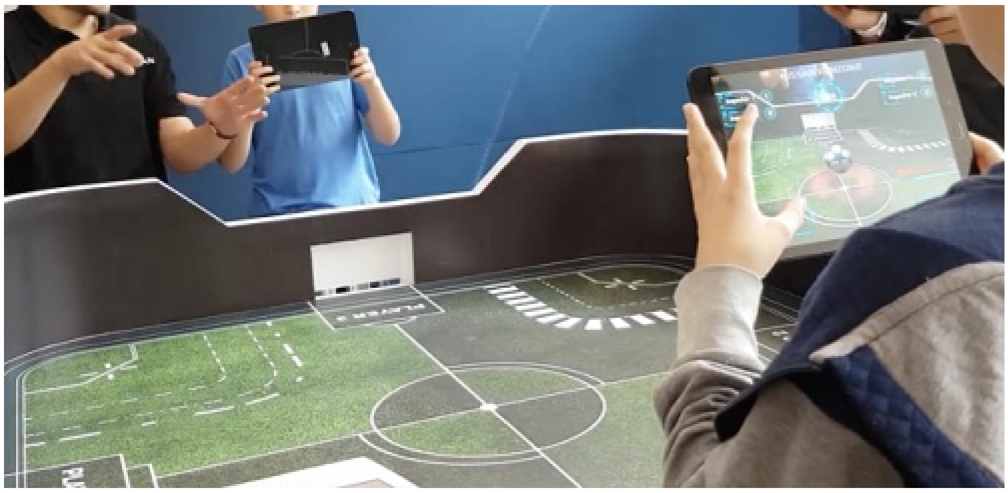


Figure 2.18: Nissan AR Multiplayer

A mini football game with twisted rules. The football players are the Nissan superstar car: the GTR. Four players have to play at the same time, this means four cars, four goals and one ball. We use a real mini football stadium built for an AR experience, its field is the tracker. The game runs on an Android tablet, use Vuforia SDK for AR and Photon OnPremise server for networking.

2.6 Music Experiences Research

A period of time was spent researching on Music Notation applications. This helped enrich our knowledge around the notation topics and determine the specific features that we were going to implement in our thesis.

There are two subsections at this topic since we tried to achieve a well-rounded research. Firstly, we were interested in the non-AR applications for music notation, with a significant amount already available. Next, we focused on finding AR projects centered around music in order to enrich our knowledge on how to combine it into the AR environment. The research concluded that were no similar applications developed regarding music integration in AR.

2.6.1 Applications for Music Notation

Software applications for music notation and composition vary nowadays. Rapidly growing technology provides the opportunity for old hand-written documents to be digitized. Apps for music notation have been developed, making it easier to create pieces while universally sharing them with other musicians, artists, etc.

Bellow we present few examples of the most characteristic music notation and composition programs:

- **Sibelius**

Sibelius is a scorewriter program developed and released by Sibelius Software Limited (now part of Avid Technology). It is the world's largest selling music notation program. Users create, edit and print music scores through Sibelius and at the same time can also play the music back. It supports virtually all music notations and it can cover every music category, even the most complex one. Sound libraries has been implemented giving the composer a variety of sounds for the score's playback. Furthermore, cloud publishing feature is available. Thus, every user can publish their own scores via the internet using PCs, laptops, tablets, etc, and at the same time users, with the software, Sibelius Scortch, can view and edit these scores.

- **Notion6**

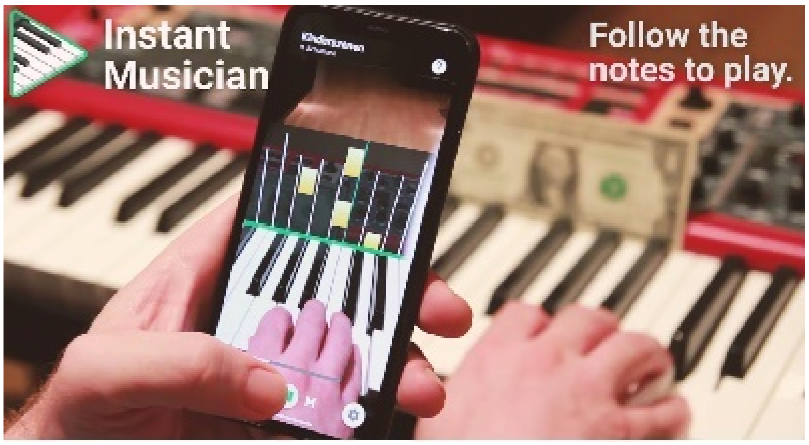
Notion6 is another computer software program for music composition and performance, created by NOTION Music, now owned by PreSonus. Among with Sibelius, they are considered to be the top tier notation programs. Notion6 supports composition using mouse/keyboard, MIDI keyboard, MIDI guitar, MIDI file, etc. Furthermore, for the first time hand-writting recognition is being introduced. Through this process musicians can scan their hand-writtend scores and implement them into Notion6.

There are many more applications that we can refer to, such as MuseScore, Noteflight, Finale PrintMusic, Forte Home and more. Every single notation application includes great complexity in order to cover every possible note's combination. In our thesis thought, in order to focus on the collaborative features, we keep the notation aspect at the lowest level, using the most basic case scenario.

2.6.2 AR Applications in Music Notation

In our approach, we aim to blend one very simplified use-case composition scenario with Augmented Reality. Our research around the AR aspect of music notation didn't lead us to any specific application. While we kept researching around the topic, we managed to find one AR application for music in general.

- **Instant Musician**



Instant Musician is a mobile AR application focused on Piano players. Using Apple's ARKit technology, the Instant Musician app superimposes augmented reality indicators directly on the piano or keyboard that show to the user which notes to play, when to play them, and for how long to hold them. Musicians can see through their screen as their hands play along.[22]

2.7 Game Engines

A Game Engine, also known as a game architecture, is a software-development environment designed primarily for the creation and development of video games and applications.

Historically game engines have sometimes been closely tied to the games themselves. In 1987, Ron Gilbert, along with some help from Chip Morningstar, created the SCUMM, or Script Creation Utility for Maniac Mansion game engine, while working at Lucasfilm Games, which was later renamed LucasArts. SCUMM is a great example of a game engine that was custom-made for a specific type of game. The "MM" in SCUMM stands for Maniac Mansion, which was a critically acclaimed adventure game and the first to use the point-and-click style interface, which Gilbert also invented. In mid-1991, at a company named id Software a seismic shift in the industry occurred. The 21-year-old John Carmack built a 3D game engine for a game called Wolfenstein 3D. Up until then, 3D graphics were generally limited to slow-moving flight simulation games or games with simple polygons, because the available computer hardware was too slow to calculate and display the number of surfaces necessary for a fast-paced 3D action game. Carmack was able to work around the current hardware limitations by using a graphics technique called Raycasting.

Modern-day game development studios such as Bethesda Game Studios and Blizzard Entertainment often have their own in-house, proprietary game engines. If a game development company doesn't have an in-house engine, they typically use an open-source engine, or license a third-party engine. Nowadays, with the high demand on smartphone applications there a lot of Game Engines on the market, the most powerful ones are Unity and Unreal. These Game Engines have proven to be a great tool for developers around the world, who don't need to create their own Game Engine and can focus on creating their application.

2.7.1 Unity vs Unreal

Unity, since 2005, is an extremely flexible and powerful Game Engine that affords a huge number of advantages over other game engines available in the market today, long supporting 2D and 3D graphics. Unity offers a visual workflow with drag-and-drop capabilities and supports scripting, with a very popular programming language, C#. With emphasis on portability Unity provides toolsets for sophisticated user-friendly experience with each release and it offers cross-platform support for 27 different platforms while taking advantage of graphics APIs specific to the system architecture, including Direct3D, OpenGL, Vulkan, Metal, and several others. Unity also comes with an integrated physics engine (nVidias PhysX) and an Asset Store, an online storefront where content creators and developers can upload content to be bought and sold.[23]

Unreal Engine(UE), initially released on 1998, is a complete suite of game development tools, powering hundreds of games,simulations and visualizations. It is one of the most advanced engines to date due to delivering top quality visuals while providing its users with a large variety of tools to work with. Because of its capabilities, efficient design, Visual Scripting and ease of use it is well-appreciated engine from hobbyists to development studios. Developers can also port their projects to mobile devices, both Android and iOS. Finally, UE also has a marketplace, similar to Unity providing art assets, models, sounds, environments, code snippets, and other features that others could purchase, along with tutorials and other guides.[24]

2.7.2 Why Unity

For our thesis, taking into account the advantages and disadvantages of each engine, it was decided to continue the development of this thesis with Unity, due to the following reasons:

First of all, Unity offers scripting with C# in comparison with Unreal's Visual Scipting. Scripting was the ideal due to changes that had to be made through code.

Secondly, Unity offers building and exporting in multiple platforms, making it useful for the future of this thesis in the sector of Augmented Reality.

Thirdly, Unity has managed over the years to create a very strong community, which proved to be very helpful during the thesis. People all over the world are connected through Unity's official forum or other mediums.

Fourthly, being able to implement AR Foundation and PUN2:Photon Engine provided us every possible component we wanted, in order for the collaboration AR to be achieved.

Last, Unity's Asset Store helped us to only focus on our thesis while having elegant assets inside the project.

Chapter 3

Musical Composition-Notation

3.1 Introduction to Music Composition

In this thesis, besides the AR technology and the collaborative options that we developed, the purpose of the final product is to form a music piece. In order for something to be considered a music piece, elements like harmony, melody and rhythm are required. Music composition is the process of creating or writing a new piece of music. The full meaning of composition can be found at Jean-Benjamin de Laborde's quote:

> *Composition consists in two things only. The first is the ordering and disposing of several sounds...in such a manner that their succession pleases the ear. This is what the Ancients called melody. The second is the rendering audible of two or more simultaneous sounds in such a manner that their combination is pleasant. This is what we call harmony and it alone merits the name of composition.*

3.2 Introduction to Musical Notation

Musical notation is any system used to create visual representation of a composition or music. Since the very first report of musical notation in the Ancient Near East, till nowadays and the standard notation, a lot of changes have been noticed. Clay tablets have taken the place of cuneiform ones in Ancient Greece and text syllables were representing notes. During Byzantine empire we encounter the first introduction to the notes, as we know them today, and it was what caused the Modern Europe Notation to rise.

Despite the great amount of the existed notations, such as tablatures, lead shits and graphics, the modern music notation on musical staves, is most common and the one we chose for our thesis.

3.3 Modern Music Notation

Modern Music Notation requires a musical stave, which acts as a framework upon which pitches are indicated by placing oval notes on, under or above the staff lines. In order for the note to be defined and named it needs to be placed on the pentagram. At this point, it needs to be mentioned that notation systems are quite complex and their syntax may involve a great amount of symbols. During our implementation we had to maintain the simplicity of our application since it works on an AR environment, so we used a basic notation method which is explained on the forthcoming sections.

3.3.1 Notes

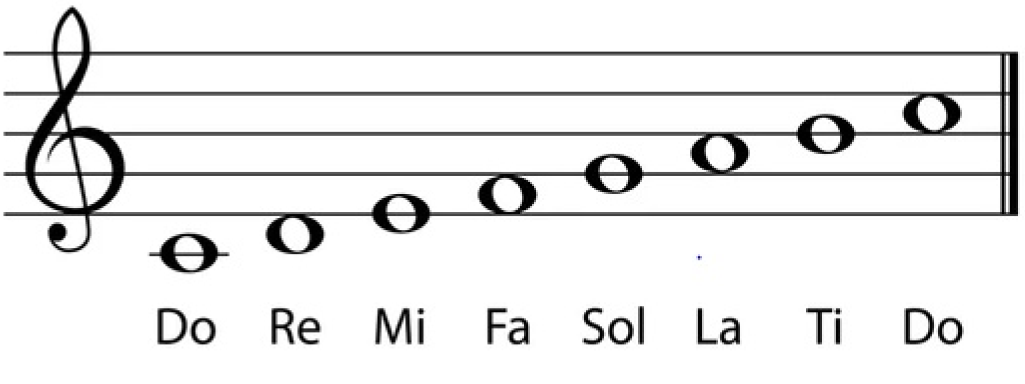


Figure 3.1: Notes

The names and the order of the notes are: Do, Re, Si, Fa, So, La, Si and they can be modified by accidentals. Accidental is a note that is a note of a pitch (or pitch class) that is not a member of the scale or mode indicated by the most recently applied key signature. Furthermore, notes have their own values which can be indicated by the note-head being a stemless hollow oval (a whole note or semibreve), a hollow rectangle or stemless hollow oval with one or two vertical lines on either side (double whole note or breve), a stemmed hollow oval (a half note or minim), or solid oval using stems to indicate quarter notes (crotchets) and stems with added flags or beams to indicate smaller subdivisions, and additional symbols such as dots and ties which lengthen the duration of a note. In our presented thesis, we use only clear notes and their values are whole, double, half, quarter and half-quarters.

3.3.2 Clefs

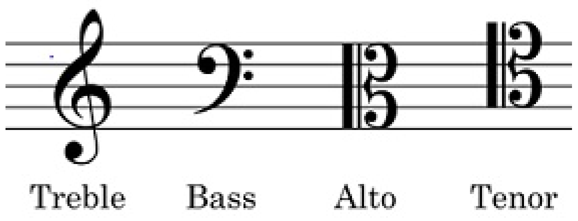


Figure 3.2: Clefs

The first thing music notation requires is for the Clef to be settled. Clefs indicate the position of one particular note on the staff and by extension, a fiducial for every other note, since every note is following the order mentioned before. There are many clefs, such as G (Sol), F (Fa), C (Do) and their use differs depending on the instrument a musician is playing. For example, Clef of Fa is used for bass instruments , Sol for treble and C for middle. The most common one is the Treble Clef, which we use in our application.

3.3.3 Key Signature

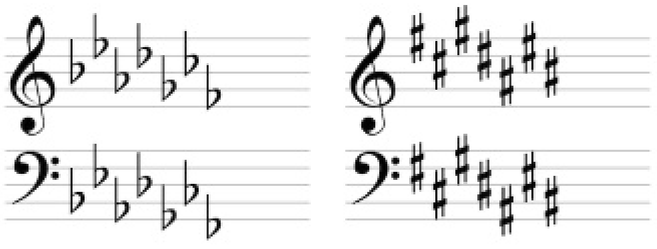


Figure 3.3: Key Signature

The Key Signature follows the Clef and it defines the Key of the song, indicating which note is sharp or flat. In our implementation we have chosen to use only clear notes and omit accidentals to minimise complexity for the user.

3.3.4 Time Signature

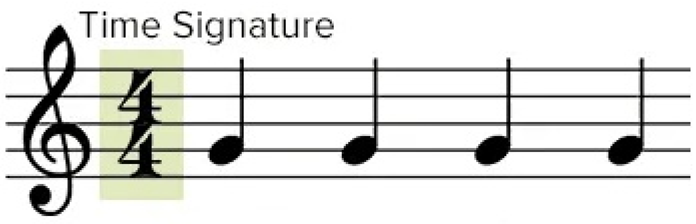


Figure 3.4: Time Signature

Time Signature is the last thing affecting basic Musical Notation. Time Signature consists of two numbers, first one contains the number of the beats of each measure and the second one, the note value which is equivalent to each beat. The most common one of them is the "four quarters", but in general those numbers vary, adding rhythm and complexity to songs. In our implementation, for the sake of simplicity and since the users are not necessarily experienced musicians, we used four quarter's signature.

Chapter 4

End user experience

4.1 Introduction to the application

In this chapter we will examine the way the application is experienced by the users. "Pentagram" is a collaborative augmented reality application for music composition. In short, users login in into the application with their usernames, the first one creates a room where the second joins and the connection between each other has been established. Then, they both scan their places in order to find and choose a vertical surface for the pentagram to be spawned. When the users are ready, a 3D pentagram spawns on the user's designated surface and the composition begins. Users add or remove 3D notes in order to create a music piece, while every user's change happens simultaneously on both pentagrams. The chapter is divided according to the application's Scenes.

The first-login Scene starts with the username requirement. The composers need to choose a username in order to login into the application. By the time the name has been chosen, if they are connected to a network, the login button leads them to the menu Scene. In case there is no network available, there is a waiting Scene that is asking them to find one and the retry to login.

The second-Menu Scene provides the choice to the users to either start the AR experience or enter the Help Screen.

The third-Help Scene works as a tutorial for the user. This application can be used by experienced or non experienced musicians since we have kept the notation part at a basic level as we explain on this Scene. Besides the music experience our user may have, the given instructions are necessary for people with no experience in Augmented reality.

There are three slides where we explain the functionality of the application. The first one provides basic information about our application regarding the collaborative part. As the user goes to the second slide, he learns about the virtual and the physical environment of the application and for the button usage. The last slide is a game-play tutorial, where the user observes a picture of the AR Scene with explanation points on every function.

The forth Scene is the AR Scene of our application, where the composition is taking place. The "Start" button in the Menu Scene activates the camera of our user's mobile phone and the AR experience starts. Users have to move around and scan their area with their mobile phones in order for the camera to detect vertical surfaces. By the time plane detection process is done, a Treble Clef will be spawned as an indicator for the location of the pentagram. Users can move the Treble Clef freely inside the scanned area, so as to choose where they want their pentagram to spawn. The "Place" button has been enabled and when it is pressed the plane detection process stops, and the Treble Clef is being placed at its final location. Afterwards, both of the users have to press the ready button in order for the pentagram to spawn. At the left side of the screen users can select the proper note, according to its value, and with the aiming dot this note can be placed on the suitable position on top of the pentagram. Our user becomes a composer who can listen every placed note until the whole music piece is created.

The AR Scene is the last one of our application and the one that our users can quit anytime they want. If any of the users loose their connection during the AR Scene, both users have to restart the application and re-establish their connection.

4.2 Login Scene

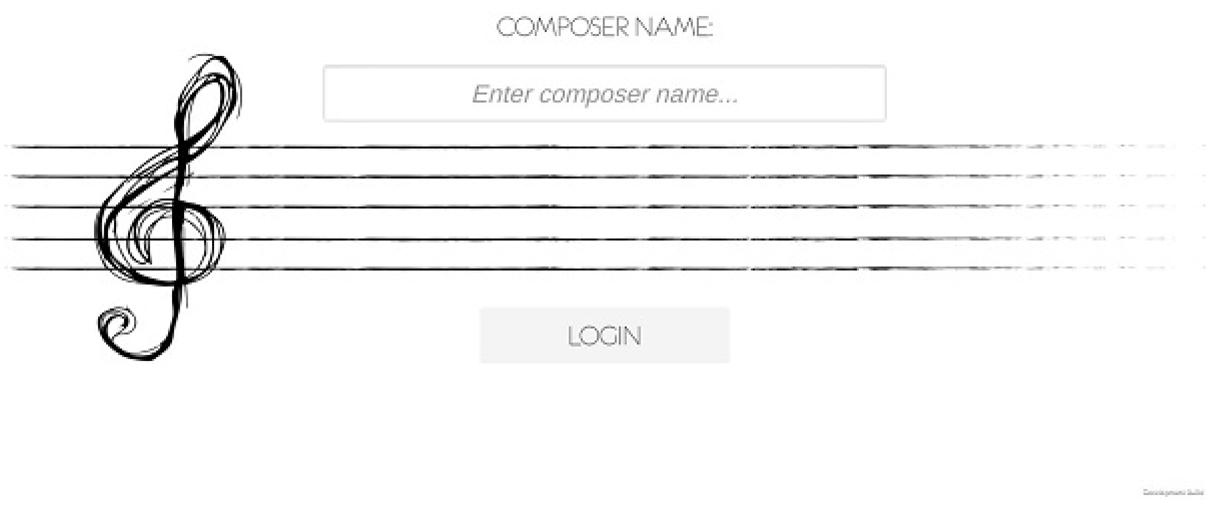


Figure 4.1: Login Scene

As users start our application they are asked to choose a username in order to connect to the application's server. There are two prerequisites for the application to continue. The username's field has to be filled and users have to be connected to the internet. The login button can be pressed after the username is ready, leading the user to the Menu Scene. If users face network issues, a waiting scene pops, informing them about their network state as we can see below:

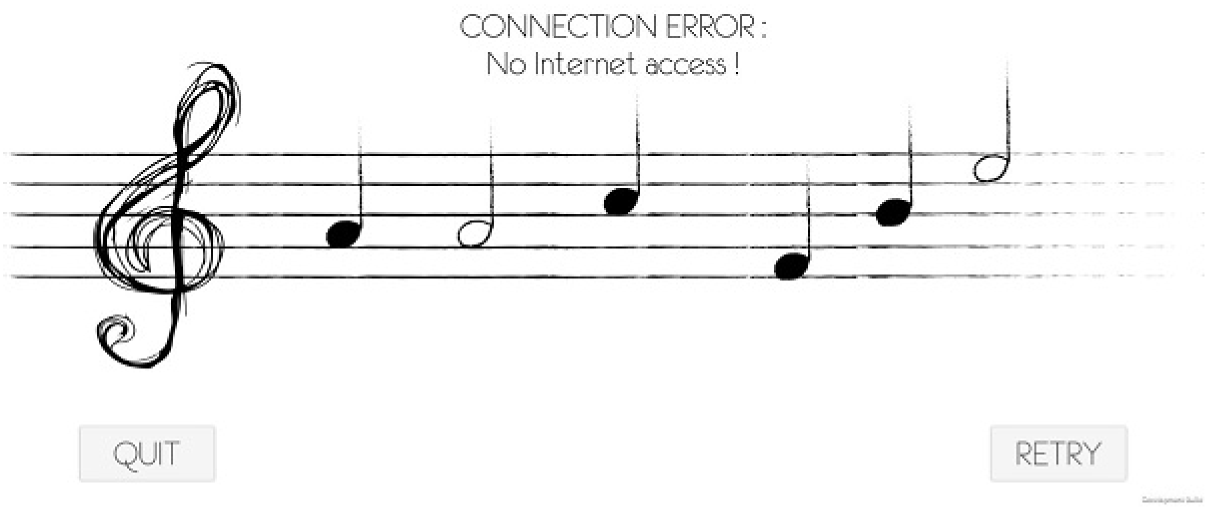


Figure 4.2: Connection Error Scene

The "Retry" button will check if the network issues have been solved, sending the user back to the Login Scene when the connection has been re-established. The "Quit" button allows the user to end the application if the network issues persist. This repetitive process secures the requirements for the application to be started.

4.3 Menu Scene

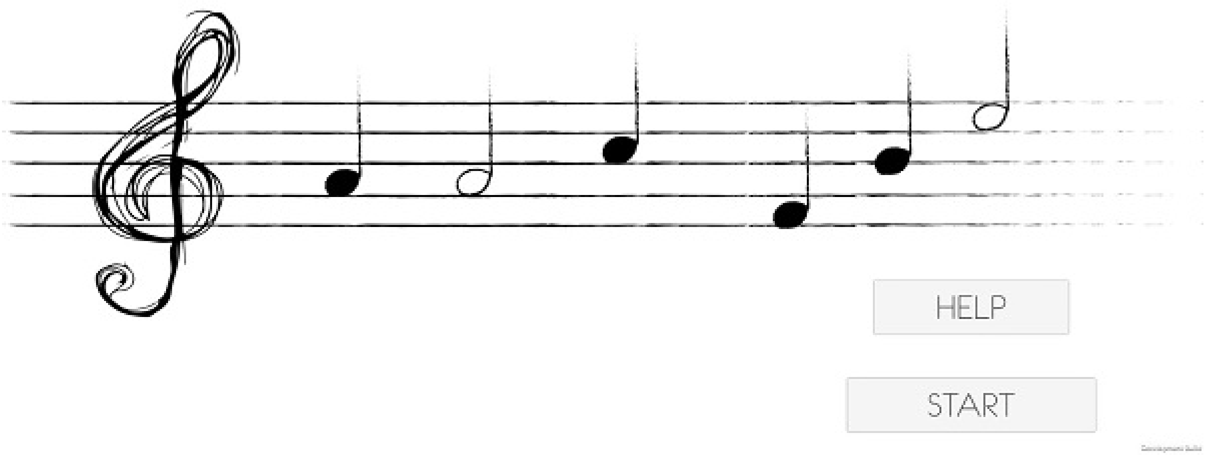
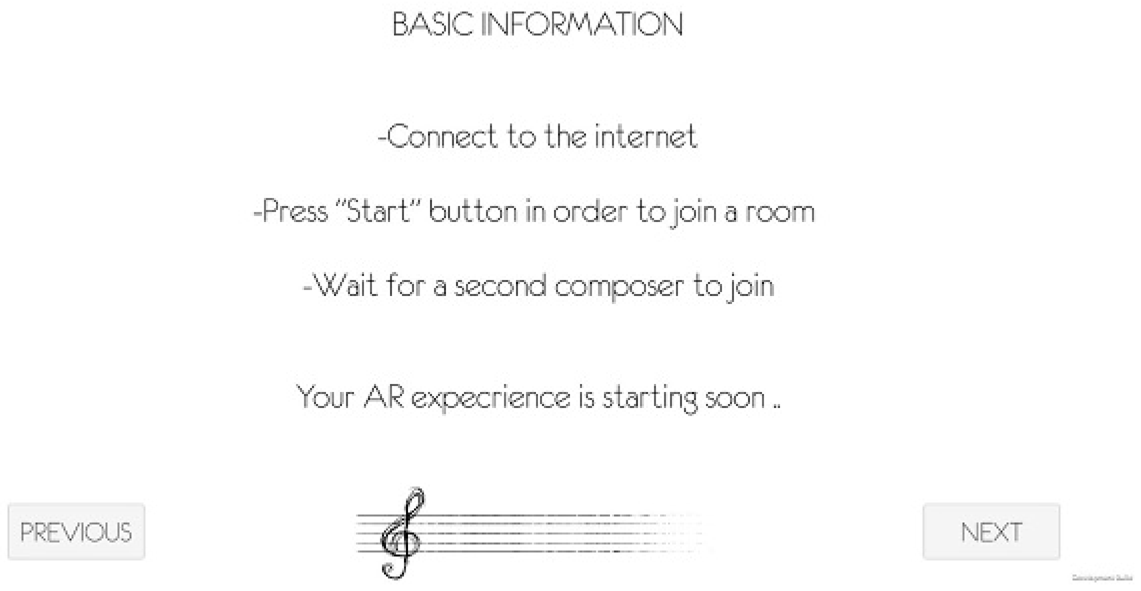


Figure 4.3: Menu Scene

At this point, users get to choose between our instructive tutorial and the start of the application by pressing "Help" or "Start" button. The "Start" button will enable their phone camera, leading them to the Augmented World. By pressing the "Help" button, they will be able to observe three images which will guide them through the application, as explained on the next subsection.

4.3.1 Help Scene


BASIC INFORMATION

-Connect to the internet

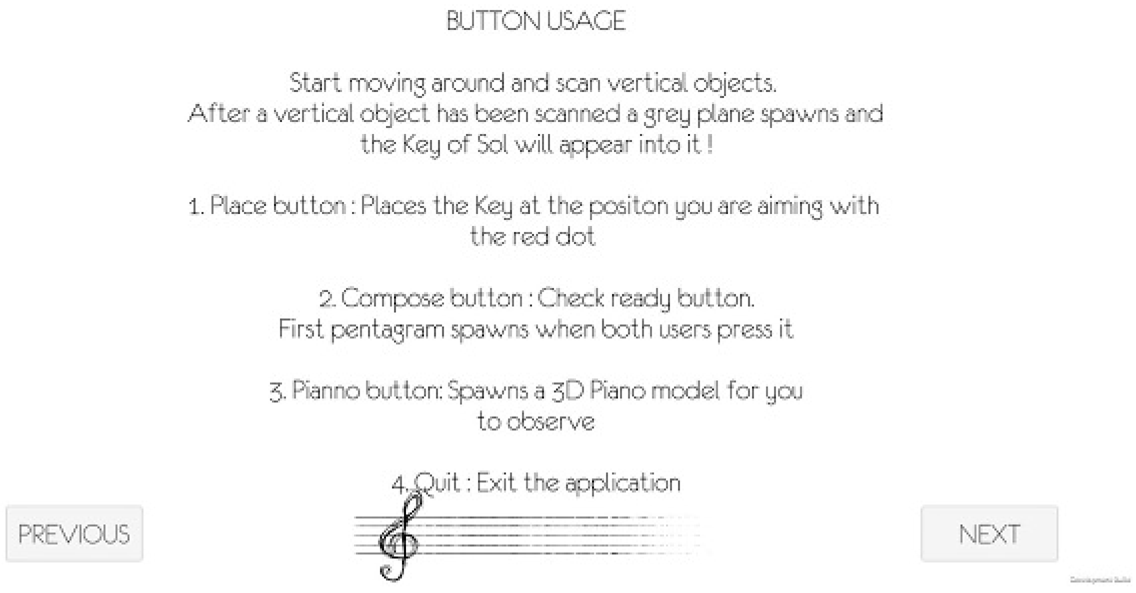
-Press "Start" button in order to join a room

-Wait for a second composer to join

Your AR expecrience is starting soon ..

Figure 4.4: Basic information

As users fade into the tutorial's slides, basic information about our application and its functionality are being presented. Since we have implemented a multiplayer application, we are pointing out the importance of the network access. Furthermore, inexperienced, with the multiplayer applications, users learn how to join a virtual room, in order to collaborate with the second composer. By joining the room, users not only match with the second user, but they are already into the Augmented Reality world.


BUTTON USAGE

Start moving around and scan vertical objects.
After a vertical object has been scanned a grey plane spawns and the Key of Sol will appear into it !

1. Place button : Places the Key at the positon you are aiming with the red dot

2. Compose button : Check ready button.
First pentagram spawns when both users press it

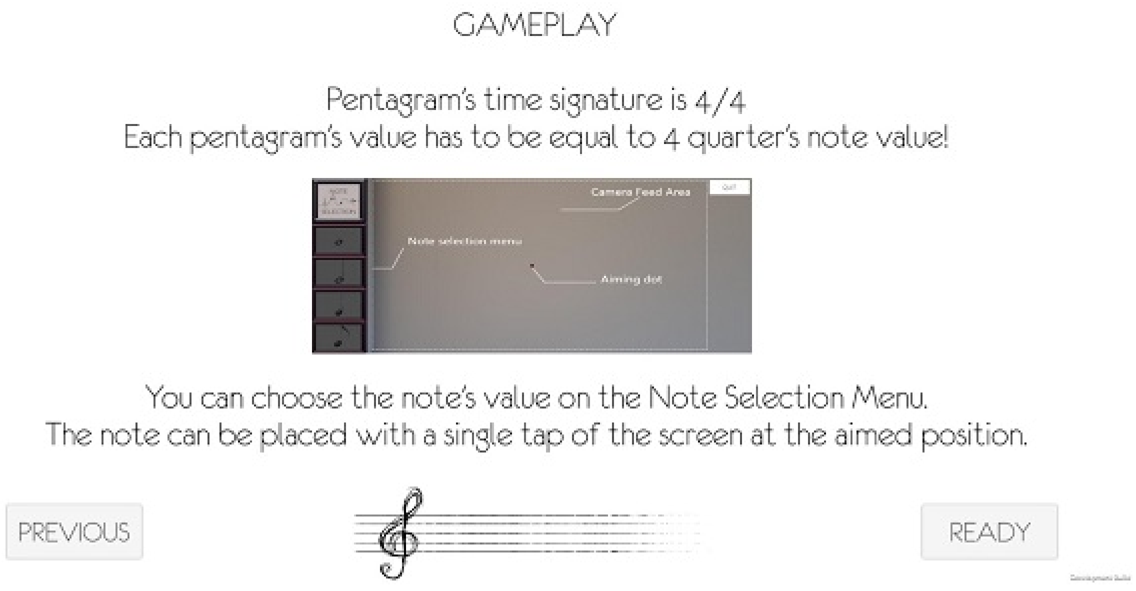
3. Pianno button: Spawns a 3D Piano model for you to observe

4. Quit : Exit the application

Figure 4.5: Button usage

The second slide provides necessary information about the button usage of our application and it is a must-see image for every user. Users are getting to know the required steps in order to reach to the Composing Scene. They are getting introduced to the plane detection process, and the way they will establish the location of their pentagram. The "Compose" button is the final step for both users in order for the pentagrams to be spawned.

Creating a stable Augmented Reality environment is quite challenging, since the technology is still under research. Thus, for the stability of our application, the aforementioned buttons and their functions can be used only once.



GAMEPLAY

Pentagram's time signature is 4/4
Each pentagram's value has to be equal to 4 quarter's note value!

You can choose the note's value on the Note Selection Menu.
The note can be placed with a single tap of the screen at the aimed position.

Figure 4.6: Game play

The last slide of the tutorial is going through a presentation about the Composing Scene. The given image from the user's point of view, contains useful information about the composing procedure and helps the user familiarize with the AR environment. Users learn about the music notation system we use in our application, which secures a proper composure. At this point, our tutorial has come to an end. The "Ready" button will lead the user back to the Menu Scene in order for the AR experience to get started. During this tutorial users can move between slides with the "Next" and "Previous" buttons.

4.4 Composing-AR Scene

Users are now ready to start composing. In order for this to be possible the application needs permission from them for the camera to be activated. As the camera is enabled, the blending between the real and the virtual world starts.

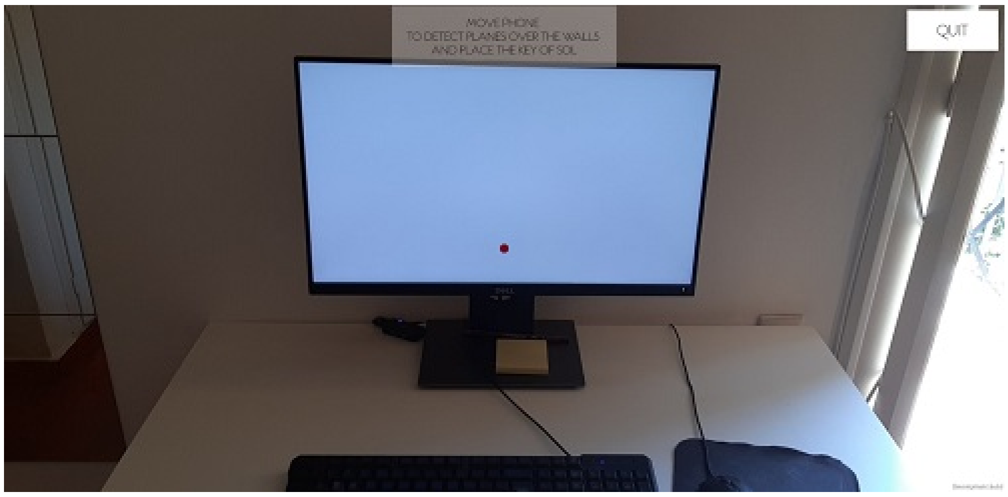
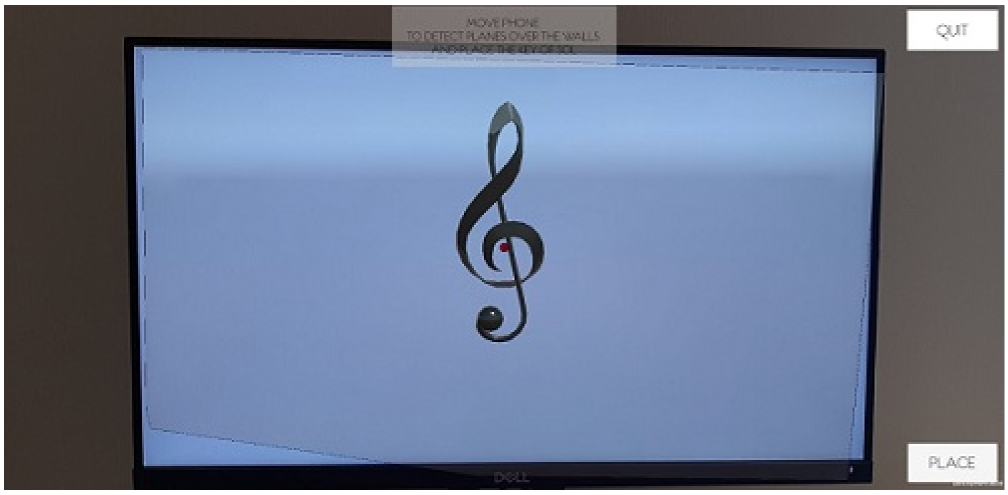


Figure 4.7: Plane Detection

The plane detection process is the first task users have to do on the start of the AR Scene. They have to move with the phone and scan the surrounding area for vertical and horizontal objects. As users move with their phone, scanning the field, grey planes will be instantiating if the application spots a surface. This is an important process for our application since the 3D Treble Clef spawns on top of those detected surroundings as we can see bellow :



MOVE PHONE
TO DETECT PLANES OVER THE WALLS
AND PLACE THE KEY OF SOL

Figure 4.8: Treble Clef Placement

The "Place Button" stops the plane detection and places the Treble Clef at the intended location. This exact location will provide a placeholder where our first pentagram will spawn. The spawned key cannot be moved after its final placement so users have to place it wisely. As the application runs, users have to choose between a 3D model piano's observation or the main composing event.

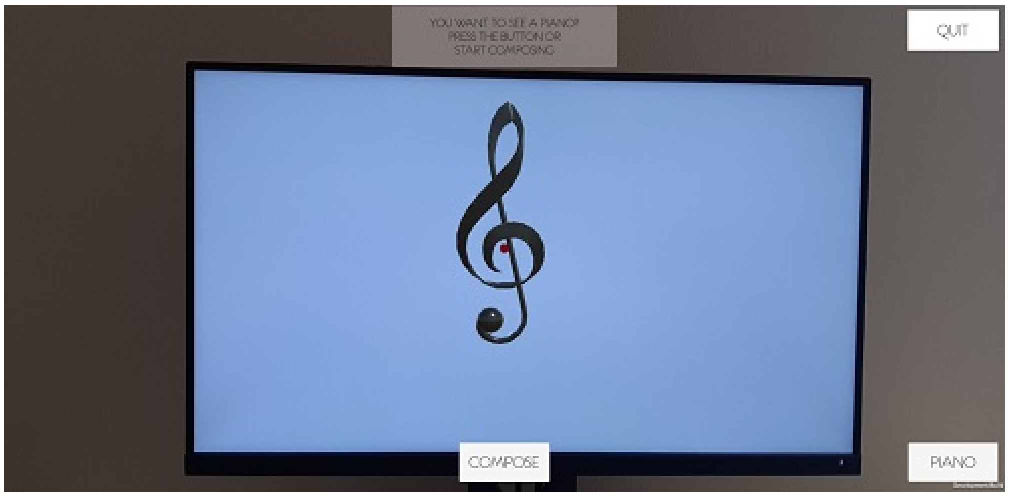


Figure 4.9: Piano or Composing

The "Piano" Button will spawn the Piano's 3D model, which can be moved anywhere in the physical world by the users by touching the screen of their phone. This AR Feature is not a part of the collaborative aspect of our application, but aims to familiarize the users with the Augmented Reality in general. Users are able to observe every point of view around the Piano by placing it anywhere in the physical world.

The "Compose" Button is the beginning of the Collaboration part of our thesis. By pressing this button our user joins a virtual room in order to match with the other composer. If there is no room then the first user tapping the compose button creates one. At the same time, the Note's selection UI pops on the mobile phone's screen, on which our users will be selecting their notes.

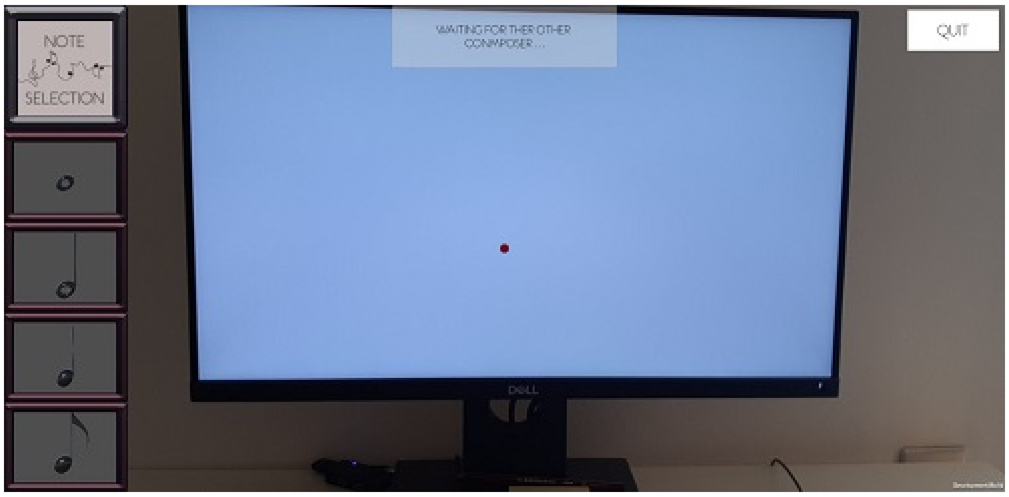


Figure 4.10: Waiting Composer

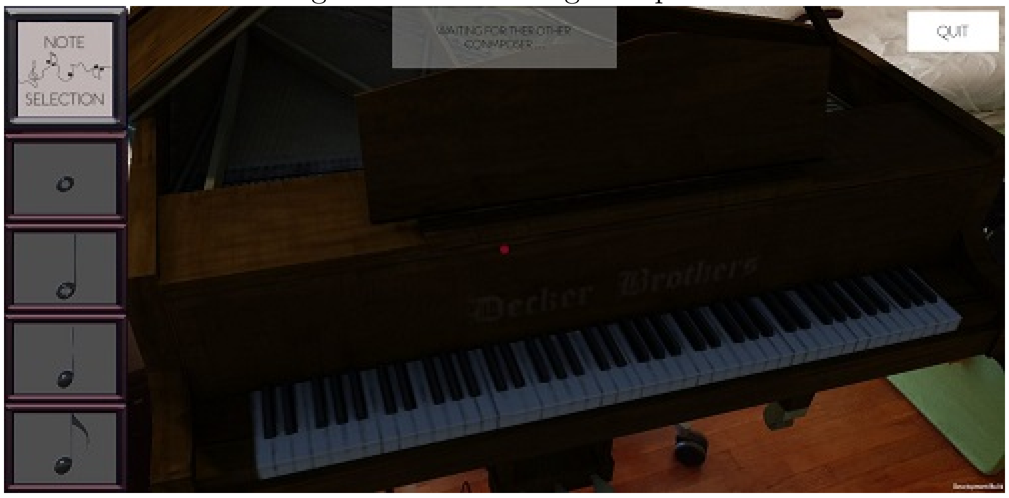


Figure 4.11: Piano

As it can be noticed in the image above, the key of sol has peen deleted and the user is now waiting for the other composer. This waiting scene is a result of one pressed compose button without a follow up from the second user. Since the waiting time depends exclusively on the second user, we implemented the 3D piano in order for the AR experience to be continuous. As the user waits, he can enhance the real world around him by moving the piano at the proper position, in order to create a music room. By the time the second user joins, the first pentagram will be spawned at the position they key of sol was.

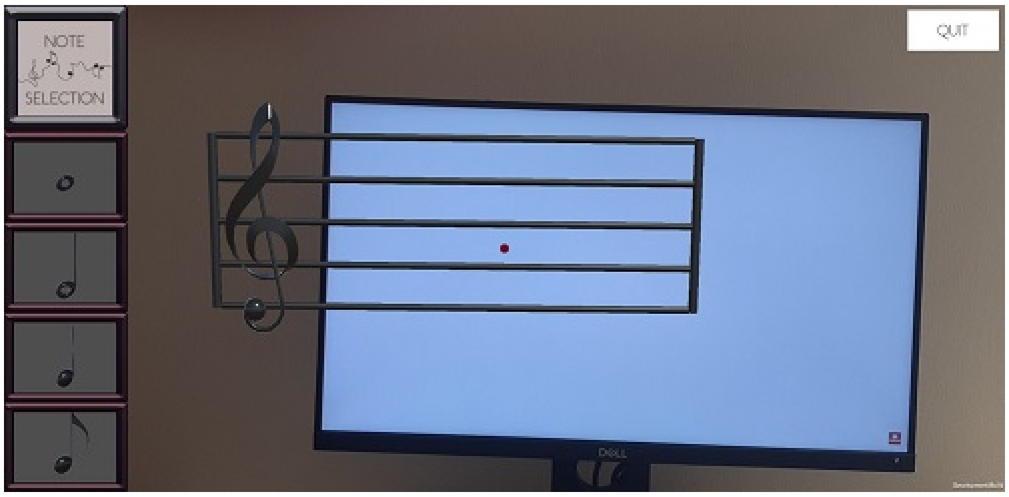


Figure 4.12: First Pentagram

At this point, the Composing scene is taking place and our users are able to compose with each other. By touching the screen where the note selection UI is, composers select the desired note. The red dot in the middle of the screen points at the location users will be spawning their notes. Hovering the dot on top of the pentagram will change the line's colour for the position of the note to be designated. As the AR experience continues, users instantiate notes simultaneously on both pentagrams until the limit of the 4 out 4 value is completed. A new pentagram is about to be spawned when the users achieve this number.

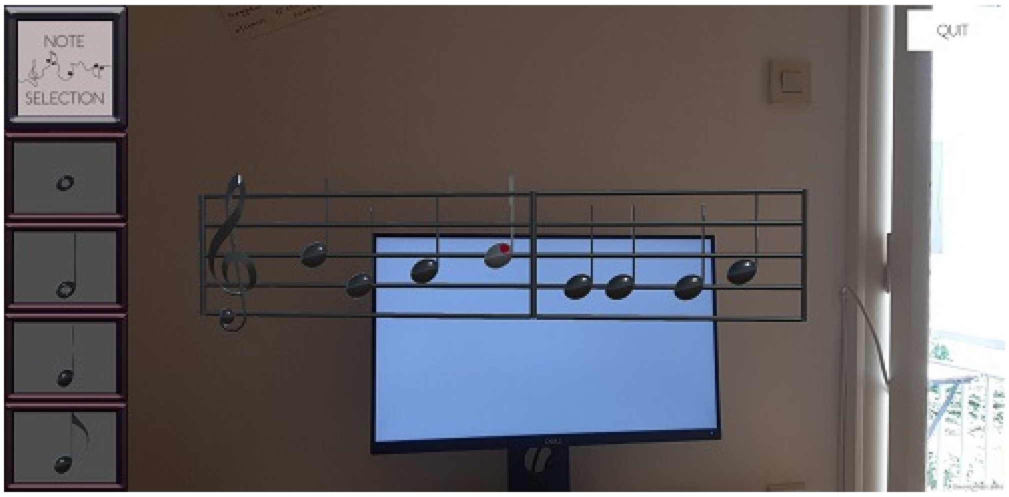


Figure 4.13: Note Spawn

During the Composing experience users are able to listen to the spawning notes in order to secure the harmony of the music piece. In case of a music error, users can delete any non desired note and replace it with others as long as they secure the 4 out of 4 value of each pentagram. Users have to hover with the aiming dot the non desired note and by the time its colour changes, a single screen touch is enough. A message pops in the middle of the screen, informing both of the users that a pentagram is not full. This is a repetitive process which secures that every time a user edits a pentagram, the composure will be paused until the value of the edited pentagram is set. Both users have discretion on the matter of editing regardless of who deletes the note.

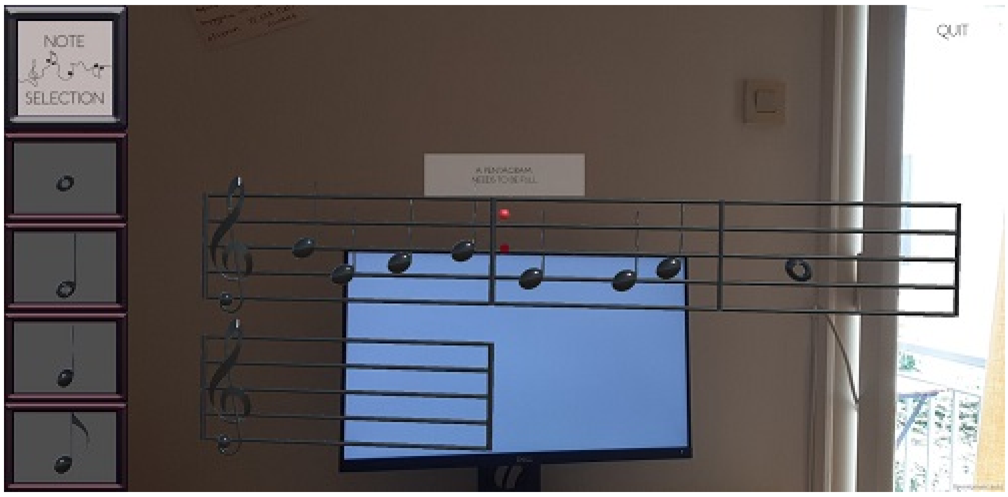


Figure 4.14: Edit feature

As users complete the pentagrams more will spawn, and the collaboration will continues until they choose to quit. Experienced musician's goal is to compose freely into the 3D pentagrams until the music piece is over. On the other hand, new musicians are able to familiarize with the notes, especially if they collaborate with their teachers.

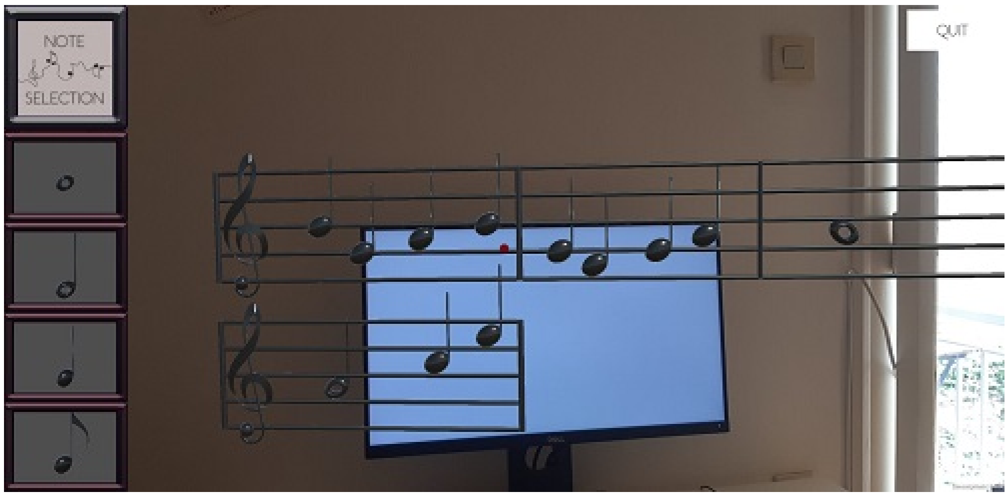


Figure 4.15: Composure

Chapter 5

Implementation

5.1 Introduction

In this section, the technical implementation will be explained based on the developer's point of view. The stages of the developing phase will be presented in this chapter, as we breakdown the development process.

This thesis is the result of research around the Augmented Reality technology. By the time our research was over and the topic of our thesis had been established, we had to work with the development tools that Unity3D provided. Microsoft Visual Studio and Android Studio were attached to Unity platform allowing us to code and compile our application for Android smartphones.

The first step into learning the technology of AR was to develop a Test Scene and experiment with some AR Features such as observing a 3D cube and interacting with it. For this to happen, we had to import the following packages into our Unity's project; AR Foundation and ARCore XR Plugin.

As soon as we figured out the way to blend the virtual with the physical world, we worked on the multiplayer aspect of our application. We were searching for a way to make two remote users interact on the same 3D model. We achieved the collaborative part by importing Photon Unity Networking (PUN) which, under the hood, uses Photon's features.

Having the basic features set up, it was time to start developing the main experience of our application, the music composition. The testing cubes and the basic 3D models were replaced by our final assets such us, treble clefs, notes, note's sounds and pentagrams which we found into the Unity's asset store. Everything had to be scripted in order to provide a well rounded composing experience.

Below, we precisely explain every aforementioned step. Furthermore, we mention every used software during our implementation and the way we immerse it into our application.

5.2 Development platform

5.2.1 Unity3D

PentAgRam was exclusively developed in Unity 3D game engine. The version we used was Unity 2019.2. By the time we started developing our application, this version was already stable and functional. Research showed that the software and the packages we wanted to import were fully compatible with this version, since the minimum requirement was Unity's version 2019.1. We avoided further upgrades of our development platform since the changes, would not improve neither our AR functions nor the quality of our application in general. During this section, we will not delve into Unity's basics so an overall knowledge of Unity's UI is recommended.[23]

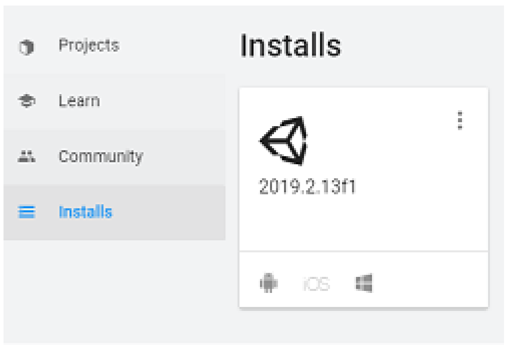
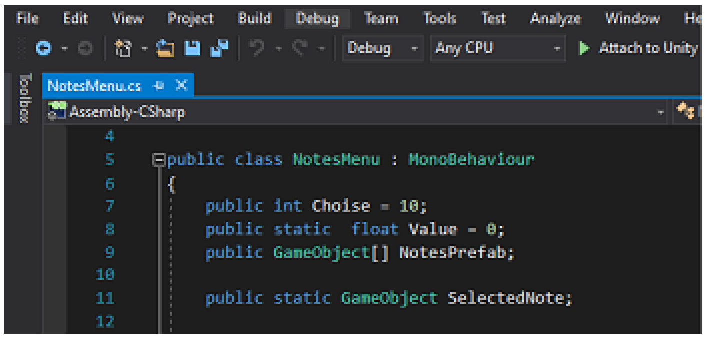


Figure 5.1: Unity 2019.2

5.2.2 Microsoft Visual Studio

Unity3D provided us with every needed tool in order to put our AR world together. Microsoft Visual is attached to Unity and has been used in our thesis as an external script editor. Microsoft Visual Tool for Unity is a Visual Studio extension that allows developers to write scripts for Unity projects using C#. The debugging capabilities of this editor along with the great connection with our game engine, lead us to write all of our scripts through Microsoft Visual Studio.

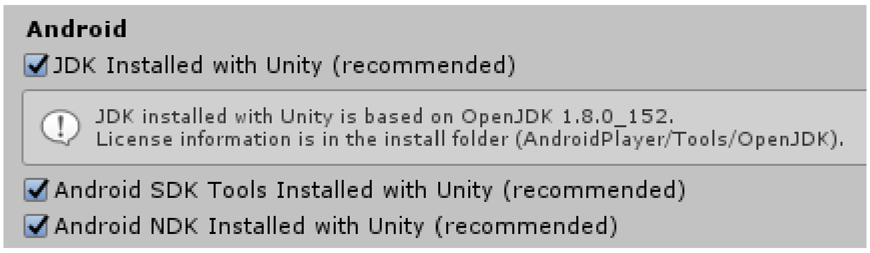


```
public class NotesMenu : MonoBehaviour
{
    public int Choise = 10;
    public static  float Value = 0;
    public GameObject[] NotesPrefab;

    public static GameObject SelectedNote;

```

Figure 5.2: Microsoft Visual Studio

5.2.3 Android Build Support

Our application's target devices are Android Smartphones. In order to build and run for Android we had to install the Unity Android Build Support module. Essentials packages as Android Software Development Kit(SDK) and the Native Development Kit(NDK) were needed as well. Unity, by default, installs a Java Development Kit which in our case was JKD 1.8.0. Until the end of the development phase, all those packages gave us a solid build and run process. The minimum requirement for our Android's version was 7.0+, since ARCore SDK is not compatible with older versions.



Android

☑ JDK Installed with Unity (recommended)

JDK installed with Unity is based on OpenJDK 1.8.0_152.
License information is in the install folder (AndroidPlayer/Tools/OpenJDK).

☑ Android SDK Tools Installed with Unity (recommended)

☑ Android NDK Installed with Unity (recommended)

Figure 5.3: Android Build Support

5.3 Software and Packages

5.3.1 ARCore

An Augmented Reality application for Android Devices requires the ARCore package to be installed on the targeted device. In the present time, while our thesis is being composed, the version we use is the ARCore 1.26. Google ARCore can be installed via Google Play Store on every supported device, allowing it to run a great amount of applications which have been developed on top of the ARCore's SDK.

ARCore uses different Application Programming Interfaces (API's) which enable the device to understand where it is relative to the world and interact with the given information. Google's ARCore provides the ARCore SDK along with some useful guides. In our implementation, we did not build directly on top of ARCore but instead, we import Unity's ARFoundation package which includes the ARCore Extension. ARFoundation will be explained precisely on the upcoming subsection.[17]

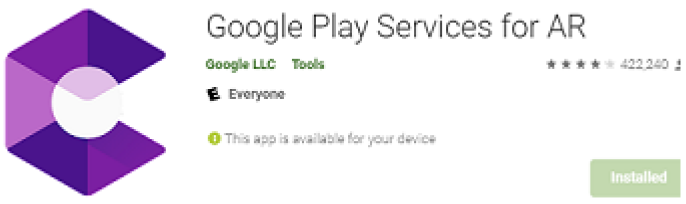

Google Play Services for AR
Google LLC
Tools
422,240
Everyone
This app is available for your device
Installed

Figure 5.4: Google's ARCore

5.3.2 AR Foundation

AR Foundation allows us to work with augmented reality platforms in a multi-platform way within Unity. This package presents an interface for Unity developers to use, but doesn't implement any AR features itself. To use AR Foundation on a target device, we also need separate packages for the target platforms such as ARCore XR Plugin for Androids or ARKit XR Plugin for iOS.

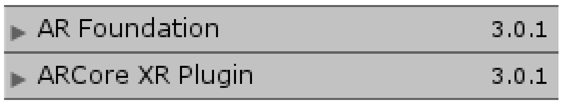
AR Foundation is a set of 'MonoBehaviours'(Unity scripts) and APIs wich were critical for our device to support the following concepts:

- Device tracking: track the device's position and orientation in physical space.

- Plane detection: detect horizontal and vertical surfaces.

- 3D object tracking: detect 3D objects.

- Raycast: queries physical surroundings for detected planes and feature points.

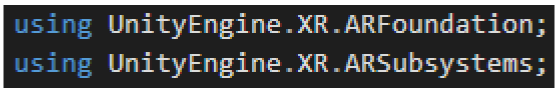


▶ AR Foundation	3.0.1
▶ ARCore XR Plugin	3.0.1

Figure 5.5: ARFoundation - ARCore

ARFoundation supports a various numbers of AR concepts that we did not implement on our application since they would not enhance our AR experience they way we design it. Unity's documentation explains extensively every feature we did not use.

AR Foundation is built on subsystems. A subsystem is a platform-agnostic interface for surfacing different types of information. The AR-related subsystems are defined in the AR Subsystems package, which we had to import into our scripts, and use the namespace UnityEngine.XR.ARSubsystems. Each subsystem runs the aforementioned functionalities(Plane detection, Device Tracking, etc).



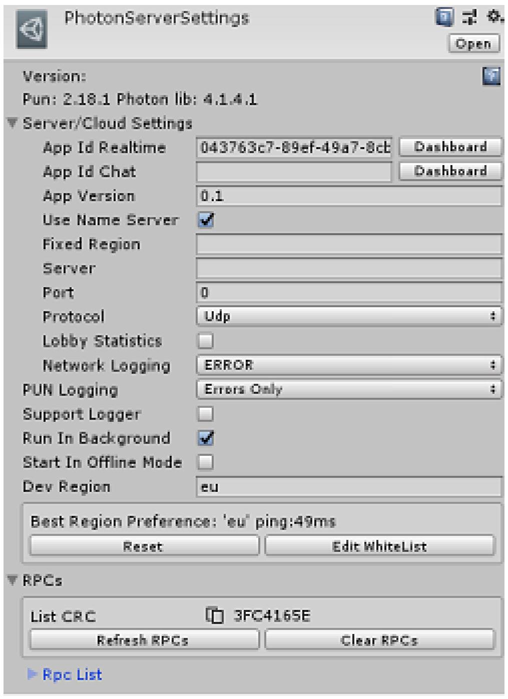
```
using UnityEngine.XR.ARFoundation;
using UnityEngine.XR.ARSubsystems;
```

Figure 5.6: ARSubsystems

Importing all these packages and using them on our scripts allows us to create a functional AR application on top of ARCore SDK. Our thesis has been built using AR-Foundation 3.0.0 and ARCore 3.0.0. Those versions are compatible with each other and during the construction of this project were the most stable. Further upgrades did not provide us with further capabilities, on the contrary, caused some compatibility errors. As long as the functionality and the quality of our application remained unchanged, we chose not to upgrade those versions.[25]

5.3.3 PUN:Photon Engine

Developing the Collaborative feature of our application was the most challenging part of our project. After spending a great amount of time researching around this topic, we ended up importing Photon Unity Networking. PUN is a Unity package for multiplayer games supported by Photon Engine. Flexible matchmaking gets your players into rooms where objects can be synced over the network. RPCs, Custom Properties or "low level" Photon events are just some of the features. The fast and seemingly reliable communication is done through dedicated Photon server(s), so clients don't need to connect one to one.



PhotonServerSettings

Version:
Pun: 2.18.1 Photon lib: 4.1.4.1

Server/Cloud Settings

App Id Realtime	043763c7-89ef-49a7-8cb
App Id Chat	
App Version	0.1
Use Name Server	✓
Fixed Region	
Server	
Port	0
Protocol	Udp
Lobby Statistics	☐
Network Logging	ERROR
PUN Logging	Errors Only
Support Logger	☐
Run In Background	✓
Start In Offline Mode	☐
Dev Region	eu

Best Region Preference: 'eu' ping:49ms

RPCs

List CRC 3FC4165E

Figure 5.7: Photon Settings

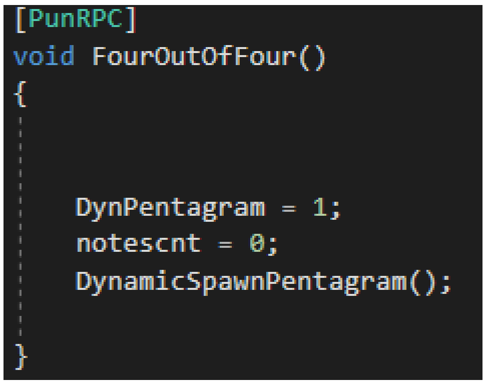
PUN was one of the greatest tools in our disposal in order to complete our application because of its structure and its documentation. The PUN wraps up three layers of APIs:

- The highest level is the PUN code, which implements Unity-specific features like networked objects, RPCs and so on.

• The second level contains the logic to work with Photon servers, do matchmaking, callbacks and such. This is the Realtime API and it can be used on it's own already.

• The lowest level is made up of DLL files, which contain the de/serialization, protocols and such.

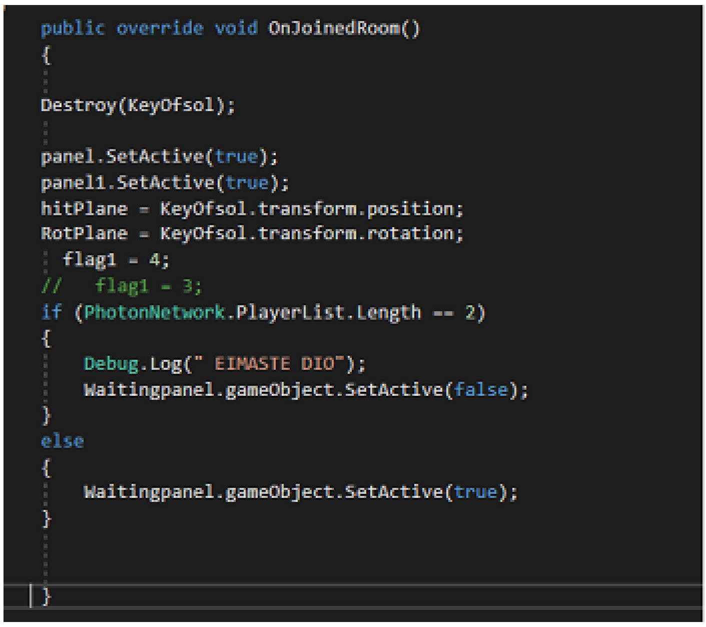
Besides the matchmaking feature, our implementation relies on the Remote Procedure Calls and on the Callbacks methods. RPCs are an exclusive feature of PUN which is not included in other Photon packages. RPCs are exactly what the name implies: method-calls on remote clients in the same room. Every method that affects the collaboration aspect is being cast as an RPC. For example, if a pentagram has reached the four of out of four value, an RPC is being called in order to spawn the next pentagram on both users.



```
[PunRPC]
void FourOutOfFour()
{


    DynPentagram = 1;
    notescnt = 0;
    DynamicSpawnPentagram();

}
```

Figure 5.8: Photon RPC

Callback is any executable code that is passed as an argument to other code; that other code is expected to call back (execute) the argument at a given time. Photon uses callbacks to let us know when the client established the connection, joined a room, etc. By the time this happens, we execute another function as below.



```
public override void OnJoinedRoom()
{

Destroy(KeyOfsol);

panel.SetActive(true);
panel1.SetActive(true);
hitPlane = KeyOfsol.transform.position;
RotPlane = KeyOfsol.transform.rotation;
  flag1 = 4;
//   flag1 = 3;
if (PhotonNetwork.PlayerList.Length == 2)
{
    Debug.Log(" EIMASTE DIO");
    Waitingpanel.gameObject.SetActive(false);
}
else
{
    Waitingpanel.gameObject.SetActive(true);
}


}
```

Figure 5.9: Photon Callback Method

The version we imported in our project was Pun: 2.18.1. Later in this section we proceed to a detailed analysis of how we implement Photon Unity Network's features in our development.[26]

5.3.4 Rest of Packages

At this subsection, we have listed the rest of the packages we have used during our development. Compared to the previous packages we analyzed, those packages are not critical for our Collaborative Augmented Reality Experience but they contribute to the creation of a completed application. Most of the following packages have being imported automatically by Unity, during the creation of our project, in order to secure its smooth functionality or because they were prerequisites for other packages. The rest were added for the User Interface design including menus, buttons, 2D pictures etc.

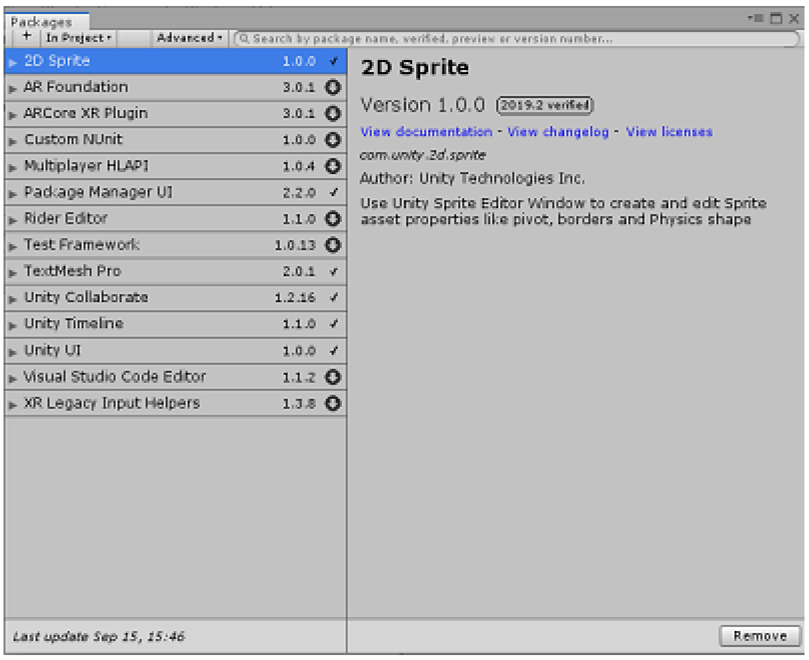
Some of the packages are listed below:

• 2D Sprite: Use Unity Sprite Editor Window to create and edit Sprite asset properties like pivot, borders and Physics shape

• Unity UI: is a UI toolkit for developing user interfaces for games and applications. It is a GameObject-based UI system that uses Components and the Game View to arrange, position, and style user interfaces.

• Test Framework: enables Unity users to test their code in both Edit Mode and Play Mode, and also on target platforms such as Standalone, Android, iOS, etc.

• Unity's Timeline: creates cinematic content, game-play sequences, audio sequences, and complex particle effects.


Packages
+
In Project
Advanced
Search by package name, verified, preview or version number...
Remove

2D Sprite	1.0.0 ✓
AR Foundation	3.0.1
ARCore XR Plugin	3.0.1
Custom NUnit	1.0.0
Multiplayer HLAPI	1.0.4
Package Manager UI	2.2.0 ✓
Rider Editor	1.1.0
Test Framework	1.0.13
TextMesh Pro	2.0.1 ✓
Unity Collaborate	1.2.16 ✓
Unity Timeline	1.1.0 ✓
Unity UI	1.0.0 ✓
Visual Studio Code Editor	1.1.2
XR Legacy Input Helpers	1.3.8

Last update Sep 15, 15:46

2D Sprite

Version 1.0.0 2019.2 verified

View documentation - View changelog - View licenses

com.unity.2d.sprite

Author: Unity Technologies Inc.

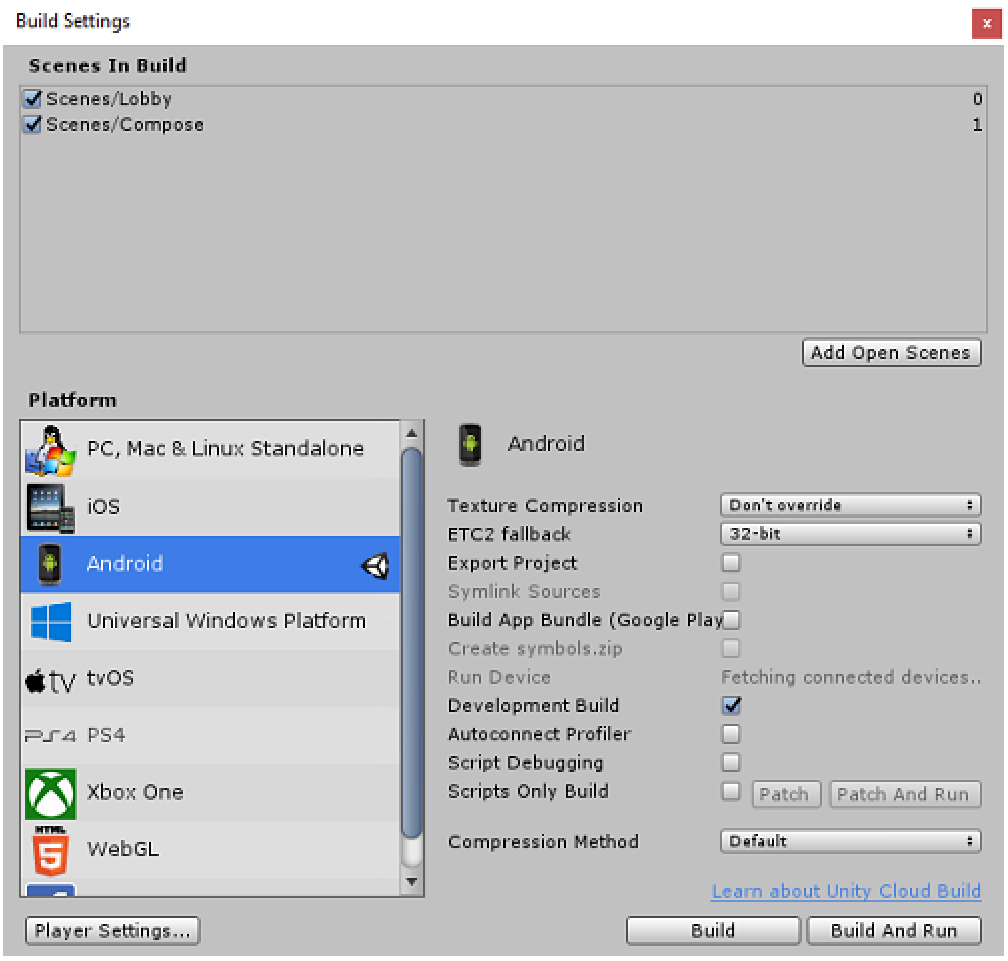
Use Unity Sprite Editor Window to create and edit Sprite asset properties like pivot, borders and Physics shape

Figure 5.10: Rest of Packages

5.4 Structure of the application

In this section we present the basic structure of our application in Unity game engine. Furthermore, we show and explain the connection between our scenes and the way they are being built into our target device.

In the next figure, are the scenes that we have built in Unity in order to run on our Android device. We have created two main scenes for the Unity to build, First/Lobby Scene and Second/Compose Scene. Into the nest of the First Scene we have implemented three sub-Scenes for the user: The Login Scene, the Menu Scene and the Help Scene.



Build Settings

Scenes In Build

✓ Scenes/Lobby	0
✓ Scenes/Compose	1

Platform: Android

Texture Compression	Don't override
ETC2 fallback	32-bit
Export Project	☐
Symlink Sources	☐
Build App Bundle (Google Play	☐
Create symbols.zip	☐
Run Device	Fetching connected devices..
Development Build	✓
Autoconnect Profiler	☐
Script Debugging	☐
Scripts Only Build	☐
Compression Method	Default

Figure 5.11: Scenes of the Application

Our application launches on the Login Scene and if the user inserts his/her credential correctly, the Menu Scene is loaded. Afterwards, based on the user's preferences one scene between the Compose/AR Scene or the Help Scene will be loaded as we have explained in User Experience section. When the user chooses to enter the Compose/AR Scene, the Scene will remain loaded until the end of the session. Below we display a diagram with the structure of our project and the connection between our scenes.

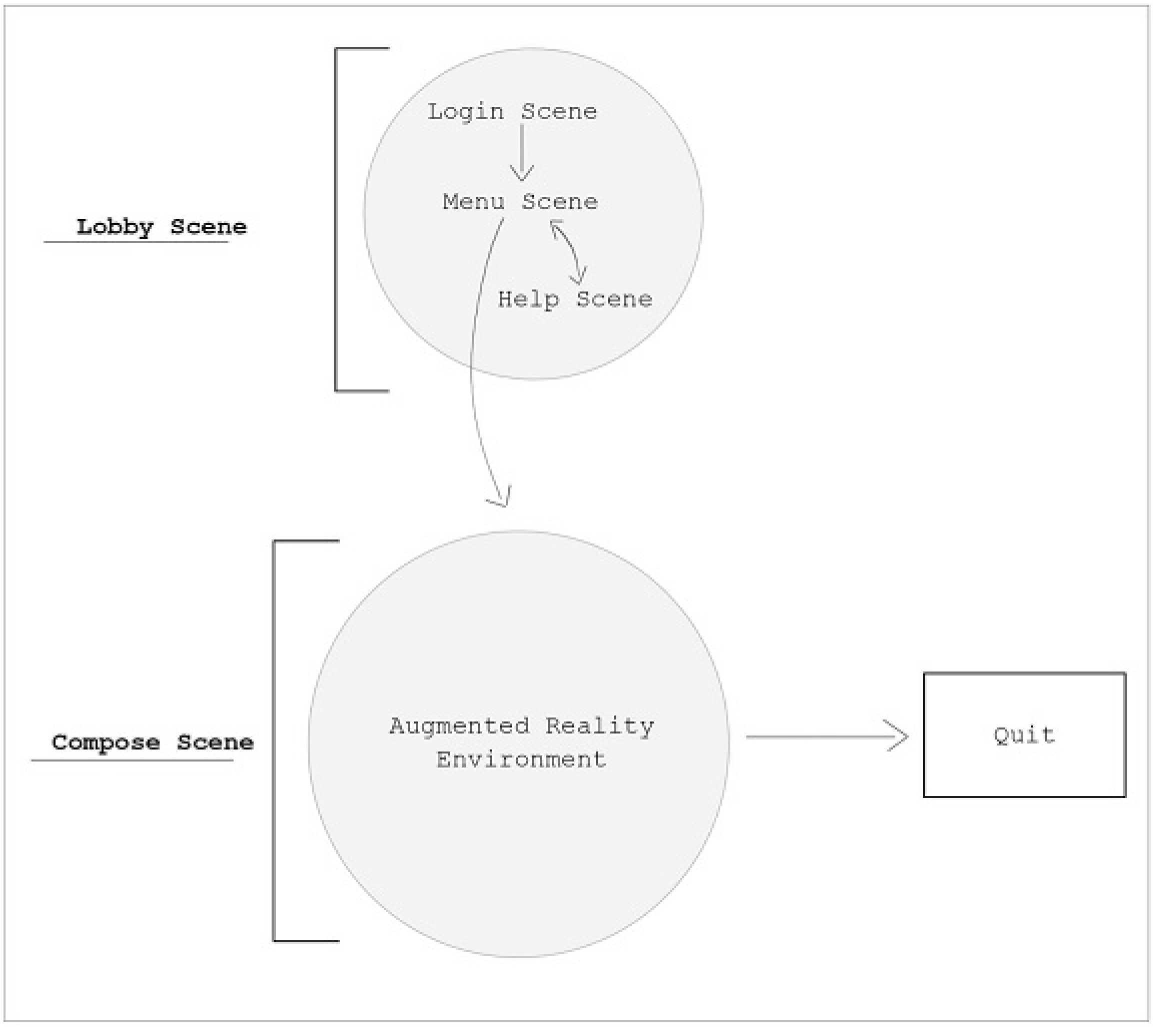
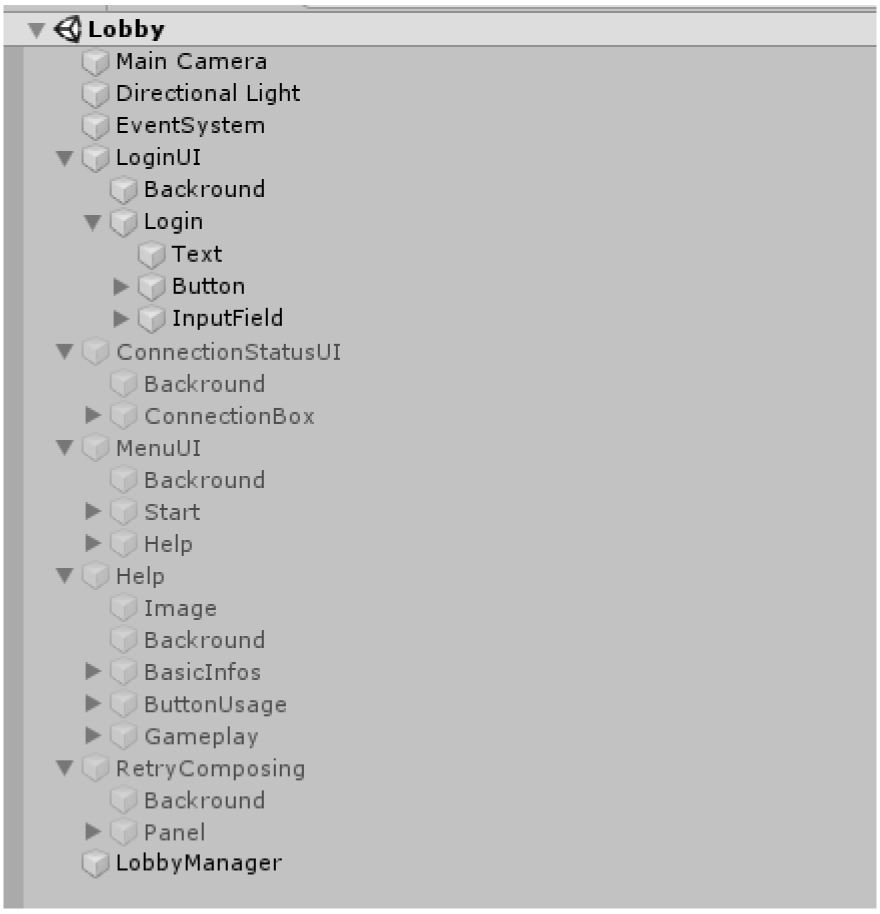


Figure 5.12: Scenes Diagram

5.5 Lobby Scene

The Lobby Scene is the first main scene of our application which consists of the Login Scene, the Menu Scene and the Help Scene. Technically, the Login, Menu, and Help Scenes are not scenes, as defined by Unity, but imaged canvases with different functions. Even though we could have formed standalone scenes for each canvas, we choose not to, since it adds unnecessary computational complexity to the implementation.

In the next figure we present the components we created for the Lobby Scene:



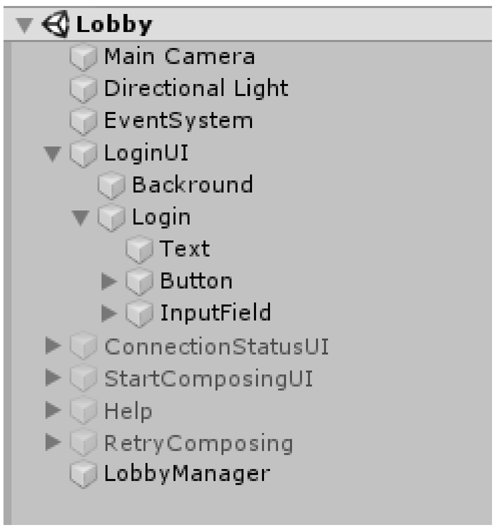
```
▼ Lobby
    Main Camera
    Directional Light
    EventSystem
  ▼ LoginUI
      Backround
    ▼ Login
        Text
      ▶ Button
      ▶ InputField
  ▼ ConnectionStatusUI
      Backround
    ▶ ConnectionBox
  ▼ MenuUI
      Backround
    ▶ Start
    ▶ Help
  ▼ Help
      Image
      Backround
    ▶ BasicInfos
    ▶ ButtonUsage
    ▶ Gameplay
  ▼ RetryComposing
      Backround
    ▶ Panel
    LobbyManager
```

Figure 5.13: Lobby's Scene Components

This scene contains mainly UI elements such us, canvases, buttons, texts, images and input fields. The interactions between them and the users will be analyzed during the next subsections, according to the scene they belong. We have added a LobbyManager which is an empty game object with a script attached to it, in order to handle these interactions. In addition, this scene includes a Main Camera object and an EventSystem object. Every scene requires at least one Main Camera, since it is responsible for the rendering of the scene. The 'EventSystem' is responsible for processing and handling events in a Unity scene. A scene should only contain one 'EventSystem'. The 'EventSystem' works in conjunction with a number of modules and mostly just holds state and delegates functionality to specific, overridable components. This object is generated automatically.

5.5.1 Login Scene - Server Connection

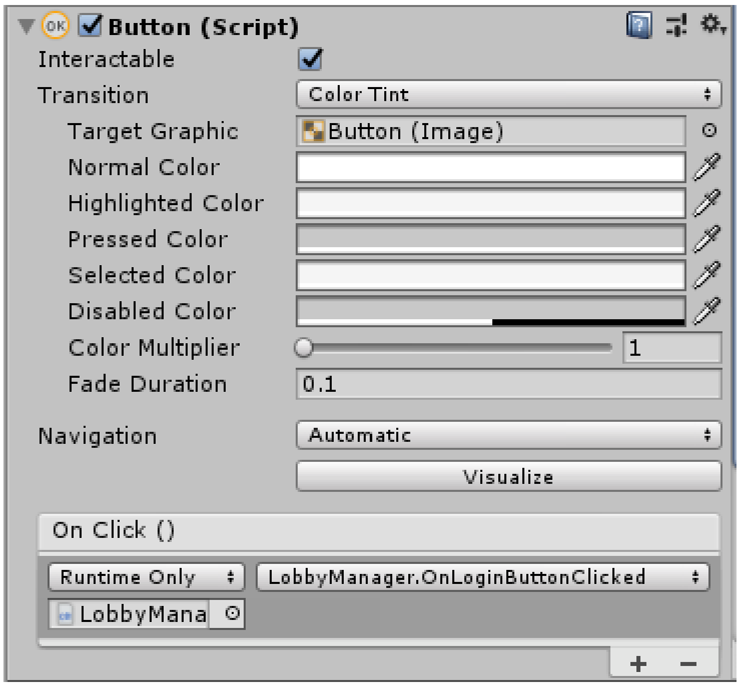
The Login Scene, figure 4.1, is the first Scene our user sees at the time our application is initially launched. To begin with, into this panel there are the background, the login button, a text and the Username's input field, as we can see below in Unity's hierarchy.



```
▼ Lobby
    Main Camera
    Directional Light
    EventSystem
  ▼ LoginUI
      Backround
    ▼ Login
        Text
      ▶ Button
      ▶ InputField
  ▶ ConnectionStatusUI
  ▶ StartComposingUI
  ▶ Help
  ▶ RetryComposing
    LobbyManager
```

Figure 5.14: LoginUI Hierarchy

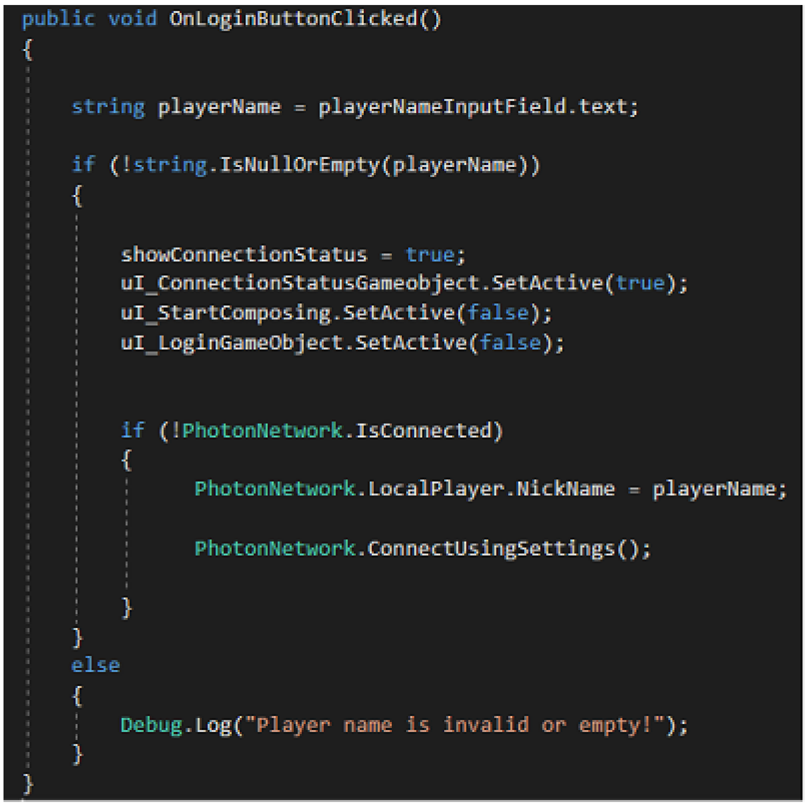
During this scene our user sets their username and afterwards they have to press the Login Button in order to connect into the Photon Server. To do so, we have developed the 'LobbyManager' script. This is a MonoBehaviour script, which means it works the scene locally and it has to be attached to an object. This script includes certain UI and PUN callback methods that secure the user's connection and transition to the Menu Scene.



Button (Script)
Interactable ✓
Transition: Color Tint
Target Graphic: Button (Image)
Normal Color
Highlighted Color
Pressed Color
Selected Color
Disabled Color
Color Multiplier: 1
Fade Duration: 0.1
Navigation: Automatic
Visualize
On Click ()
Runtime Only | LobbyManager.OnLoginButtonClicked
LobbyMana

Figure 5.15: Login Button's OnClick Script

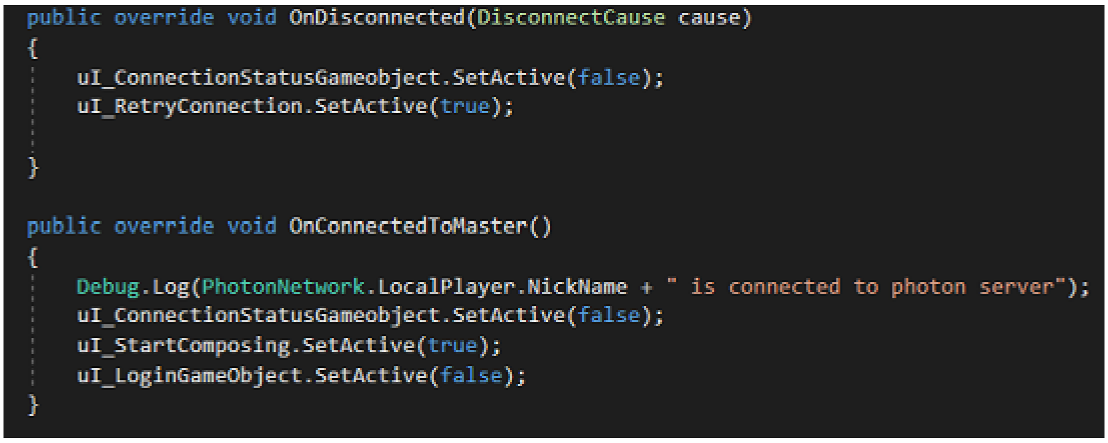
At the time the user presses the Login Button, a UI callback the method 'OnLogin-ButtonClicked' runs. This method secures that the user has filled the Username's input field and in that case, attempts to achieve the connection between them and the Photon Server. This happens with the PUN method Photon.ConnectwithSettings since we have already set up the Photon Settings as we have presented on figure 5.7.



```
public void OnLoginButtonClicked()
{

    string playerName = playerNameInputField.text;

    if (!string.IsNullOrEmpty(playerName))
    {

        showConnectionStatus = true;
        uI_ConnectionStatusGameobject.SetActive(true);
        uI_StartComposing.SetActive(false);
        uI_LoginGameObject.SetActive(false);


        if (!PhotonNetwork.IsConnected)
        {
              PhotonNetwork.LocalPlayer.NickName = playerName;

              PhotonNetwork.ConnectUsingSettings();

        }
    }
    else
    {
        Debug.Log("Player name is invalid or empty!");
    }
}
```

Figure 5.16: OnlogginButton Script

Our application requires a network connection which occasionally might be an issue for our users. During the Login phase if the connection fails the PUN callback method OnDisconnected activates a pop up panel. This panel includes a text that is describing the network issue and is asking the user to choose between a transition into the Login Scene to try to reconnect or to exit the application. As soon as the user is connected to the Photon Server the PUN callback method OnConnectedtoMaster is enabled, leading the user to the Menu Scene.

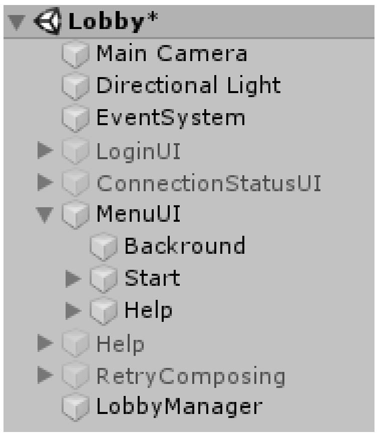


```
public override void OnDisconnected(DisconnectCause cause)
{
    uI_ConnectionStatusGameobject.SetActive(false);
    uI_RetryConnection.SetActive(true);

}

public override void OnConnectedToMaster()
{
    Debug.Log(PhotonNetwork.LocalPlayer.NickName + " is connected to photon server");
    uI_ConnectionStatusGameobject.SetActive(false);
    uI_StartComposing.SetActive(true);
    uI_LoginGameObject.SetActive(false);
}
```

Figure 5.17: PUN Callback methods

5.5.2 Menu Scene - Help Scene

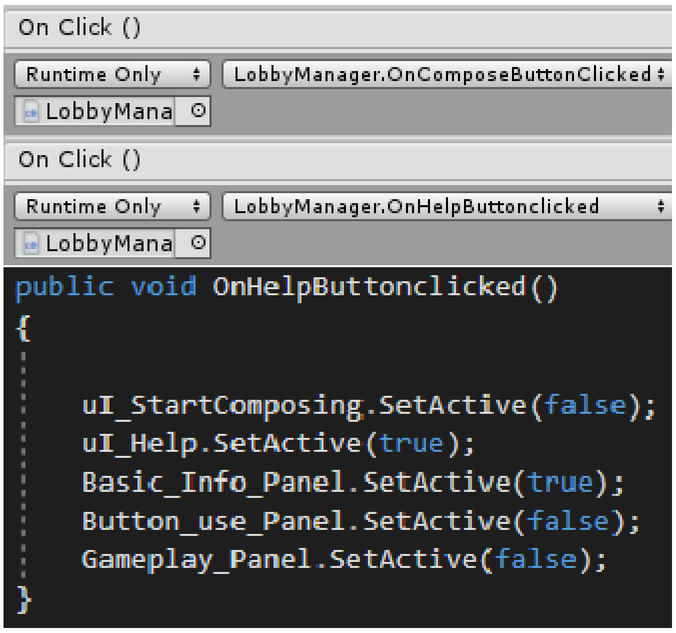
The Menu Scene, figure 4.3, constitutes a middle scene that secures the smooth transition of the connected user to the main AR experience. Furthermore, this scene provides the opportunity to the inexperienced users to reach our tutorial through the Help Scene.



```
▼ Lobby*
    Main Camera
    Directional Light
    EventSystem
  ▶ LoginUI
  ▶ ConnectionStatusUI
  ▼ MenuUI
      Backround
    ▶ Start
    ▶ Help
  ▶ Help
  ▶ RetryComposing
    LobbyManager
```

Figure 5.18: MenuUI Hierarchy

The implementation and the hierarchy of this Scene is pretty similar with the Login Scene, since we use UI callback methods while the buttons are being pressed. The methods have been developed in the same 'LobbyManager' script.


On Click ()
Runtime Only
LobbyManager.OnComposeButtonClicked
LobbyMana
On Click ()
Runtime Only
LobbyManager.OnHelpButtonclicked
LobbyMana

```
public void OnHelpButtonclicked()
{

    uI_StartComposing.SetActive(false);
    uI_Help.SetActive(true);
    Basic_Info_Panel.SetActive(true);
    Button_use_Panel.SetActive(false);
    Gameplay_Panel.SetActive(false);
}
```

Figure 5.19: Start and Help Onclick Script Buttons

The 'OnHelpButtonclicked' method disables the MenuUI canvas and enables the HelpUI canvas. This leads our user into the Help Scene where three panels, with valuable information, can be found (figures: 4.4, 4.5, 4.6).

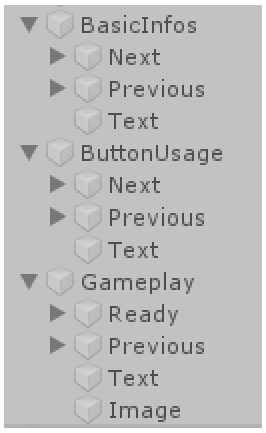


Figure 5.20: Help Scene Components

The 'BasicInfos' is the first panel of the Help Scene, followed by the 'ButtonUsage' and the 'Gameplay' is the last one. Users can navigate through those panels by pressing the 'Next' and the 'Previous' Buttons as we have already explained in the previous Chapter. The methods that are attached to the 'Previous' and 'Next' buttons remain similar to the previous 'Onclick' callback methods in this section. The Menu and Help Scenes are connected in a way that the user can go back and forth between them as many times as they want before they enter the Compose/AR Scene. Therefore, below we present a diagram between the Menu Scene and the three panels of the Help Scene, in order to clarify their connection.

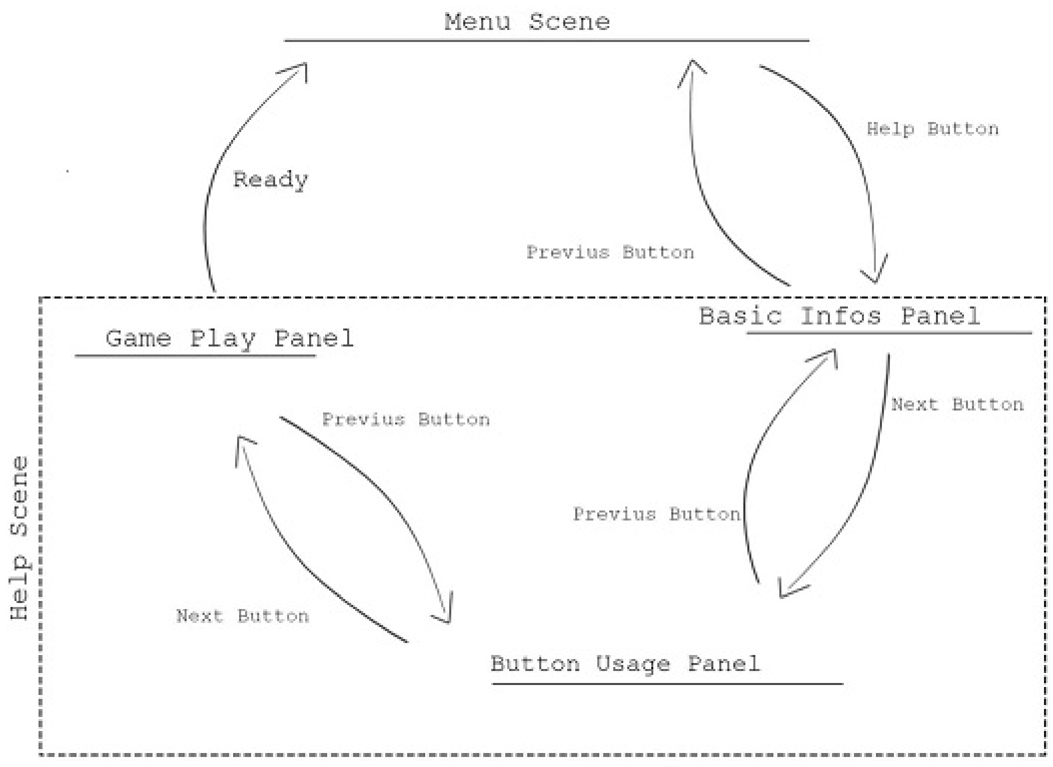


Figure 5.21: Menu and Help Scene Diagram

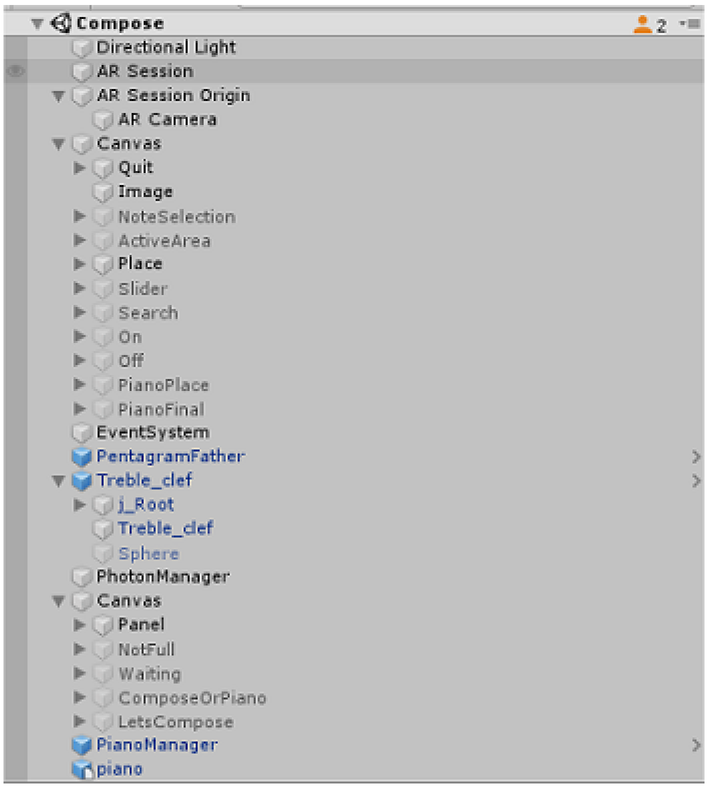
The moment the 'Start' Button is pressed, the 'OnComposeClickButton' runs. This method loads the Compose/AR Scene with the SceneManager.LoadScene instruction. Occasionally, using SceneManager.LoadScene may cause frame stuttering and pauses since the scene is loaded in the next frame and not immediately. In our case, we did not deal with any issues, since we transit from a 2D into an AR environment and the AR scene needs some time to be loaded inherently, because of the camera activation.

5.6 Compose-AR Scene

The Compose Scene, figure 4.15, is where our users experience the collaborative Augmented Reality feature. In this section we analyze the development and the functionality of this scene. Furthermore, we present our 3D models and their set up in order to work together for a smooth AR composing experience. This section will be divided according to the methods and the technologies our users meet during their session.

5.6.1 AR Scene Components

The components of our Compose Scene are being presented below. Since it is an AR Scene, Unity's game editor is empty and every test had to be ran into our Android Device.

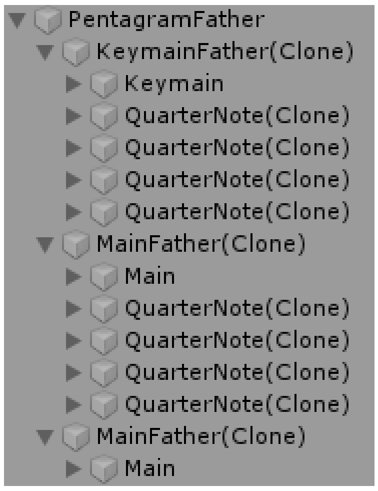


```
Compose                                  2
  Directional Light
  AR Session
  AR Session Origin
    AR Camera
  Canvas
    Quit
    Image
    NoteSelection
    ActiveArea
    Place
    Slider
    Search
    On
    Off
    PianoPlace
    PianoFinal
  EventSystem
  PentagramFather
  Treble_clef
    j_Root
    Treble_clef
    Sphere
  PhotonManager
  Canvas
    Panel
    NotFull
    Waiting
    ComposeOrPiano
    LetsCompose
  PianoManager
  piano
```

Figure 5.22: Compose Scene Components

To begin with, 'AR Session' and 'AR Session Origin' are the two components the AR Foundation requires in order to activate its AR features. The AR Session controls the life-cycle of an AR experience, enabling or disabling AR on the target platform. The AR Session Origin converts the AR session's space into Unity world space. It represents the user during the session of our application since the user sees and interacts with the virtual environment using this object. Rotation or movements of this object represent rotation and movement of the user in the scene. The attached AR Camera is the camera to associate with the target device, rendering what the user sees.

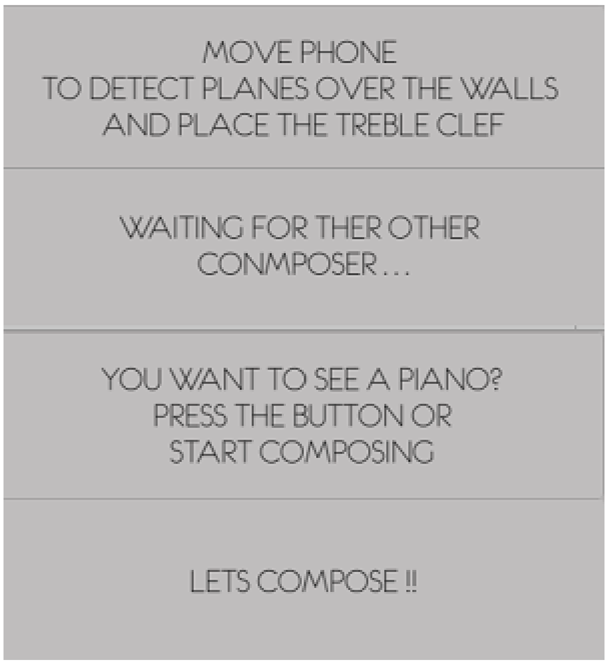
The 'PentagramFather' is an empty game object where our 3D pentagram and notes will be spawned under. This game object will become the father of every other model into our AR Scene. The Parenting process is critical for our implementation since our 3D models are being spawned dynamically in the world space. 'PentagramFather' as a parent game object forms a landmark for the nested child objects, providing important information about their position and their rotation.



```
▼ PentagramFather
  ▼ KeymainFather(Clone)
    ▶ Keymain
    ▶ QuarterNote(Clone)
    ▶ QuarterNote(Clone)
    ▶ QuarterNote(Clone)
    ▶ QuarterNote(Clone)
  ▼ MainFather(Clone)
    ▶ Main
    ▶ QuarterNote(Clone)
    ▶ QuarterNote(Clone)
    ▶ QuarterNote(Clone)
    ▶ QuarterNote(Clone)
  ▼ MainFather(Clone)
    ▶ Main
```

Figure 5.23: Parenting in Unity

In order to create an interactive environment for the user we imported two Canvases, which are responsible for the User's interface. The 'ButtonCanvas' contains the buttons and their functions with which our users interact during the session. Through these buttons our users join the virtual rooms, select, spawn and delete their notes. The 'InfoCanvas' has messages that appear under certain circumstances, providing the user with useful information.


MOVE PHONE
TO DETECT PLANES OVER THE WALLS
AND PLACE THE TREBLE CLEF

WAITING FOR THER OTHER
CONMPOSER...

YOU WANT TO SEE A PIANO?
PRESS THE BUTTON OR
START COMPOSING

LETS COMPOSE !!

Figure 5.24: Pop up messages

The AR feature of our thesis is inextricably linked with the music composition, therefore we use music notation's 3D models. Unity's Asset Store impact was great since it provided us with the Treble Clef, the notes and the piano 3D model. The two types of 3D pentagrams (Keymain,Main) were being created by us in Unity's editor.

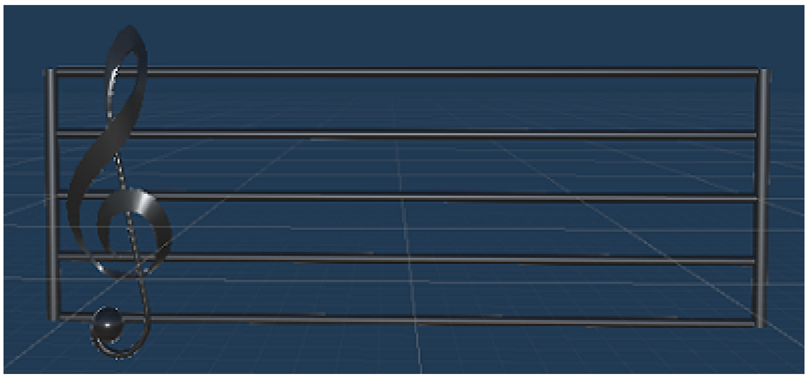
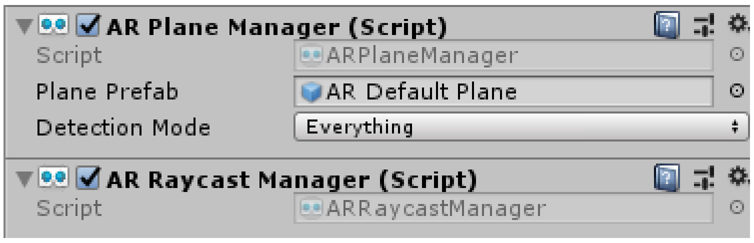


Figure 5.25: 'Keymain' Pentagram

5.6.2 Treble Clef's Placement

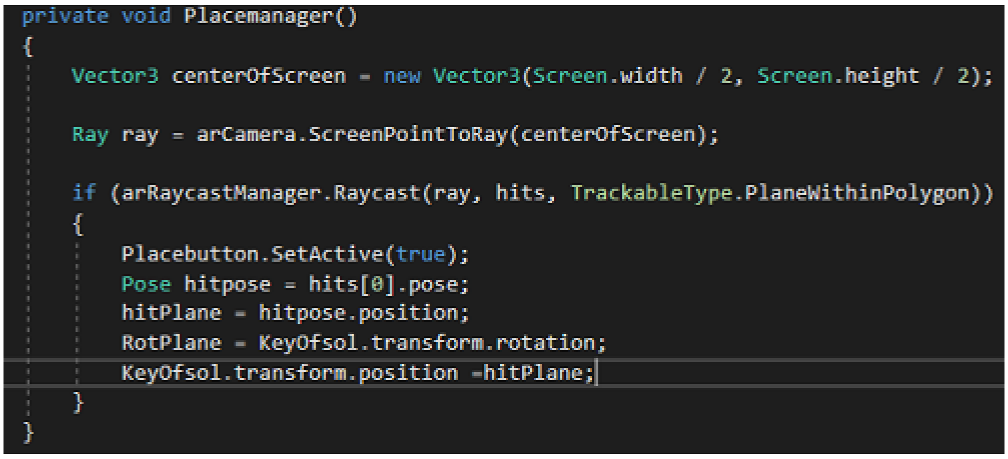
At the time our AR Scene is loaded, users have to move around in order to enable the plane detection process. Plane detection finds horizontal or/and vertical planes in user's physical surroundings. Having the AR Plane manager imported into our implementation creates a new game object every time a new plane is detected or updates the existing ones.



AR Plane Manager (Script)
Script ARPlaneManager
Plane Prefab AR Default Plane
Detection Mode Everything
AR Raycast Manager (Script)
Script ARRaycastManager

Figure 5.26: AR Plane and Raycast manager

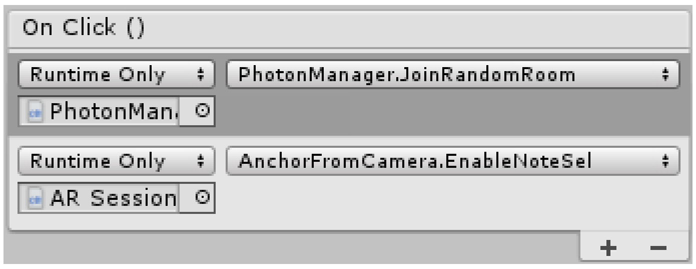
AR planes are 'trackables', which means that they can be tracked and detected by an AR device. AR ray casting allows us to determine where, and if, a ray intersects with a 'trackable'. By the time a ray hits a plane in the physical environment, the Treble Clef 3D model moves at the exact position of the hit (figure 4.8)



```
private void Placemanager()
{
    Vector3 centerOfScreen = new Vector3(Screen.width / 2, Screen.height / 2);

    Ray ray = arCamera.ScreenPointToRay(centerOfScreen);

    if (arRaycastManager.Raycast(ray, hits, TrackableType.PlaneWithinPolygon))
    {
        Placebutton.SetActive(true);
        Pose hitpose = hits[0].pose;
        hitPlane = hitpose.position;
        RotPlane = KeyOfsol.transform.rotation;
        KeyOfsol.transform.position =hitPlane;
    }
}
```

Figure 5.27: PlaceManager Method

Our users can move the treble clef until they decide its final position, which is being established by pressing the 'Place' Button. In addition, 'Place' Button's on click method activates two more buttons. Firstly, the 'Piano' Button leads the users to a standalone AR experience in which our user can interact with a 3D Piano, by moving it in the physical surrounding through their touchscreen. 'Compose' button's on click method activates the Note's Selection Panel and starts the matchmaking process.



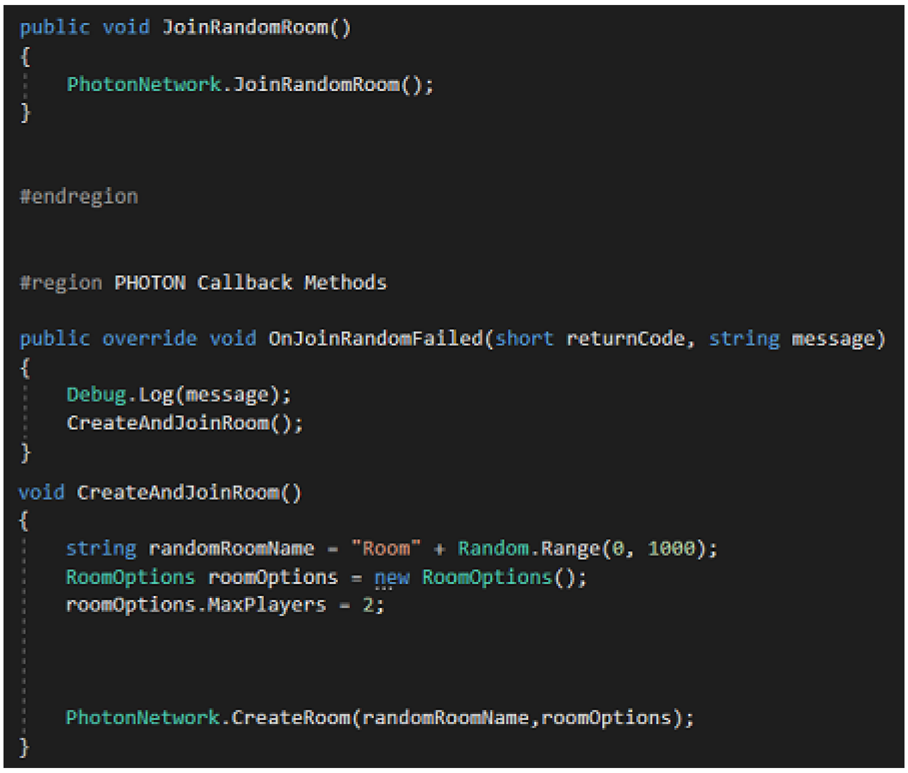
On Click ()

Runtime Only	PhotonManager.JoinRandomRoom
PhotonMan	
Runtime Only	AnchorFromCamera.EnableNoteSel
AR Session	

Figure 5.28: On Compose Button Click

5.6.3 Matchmaking

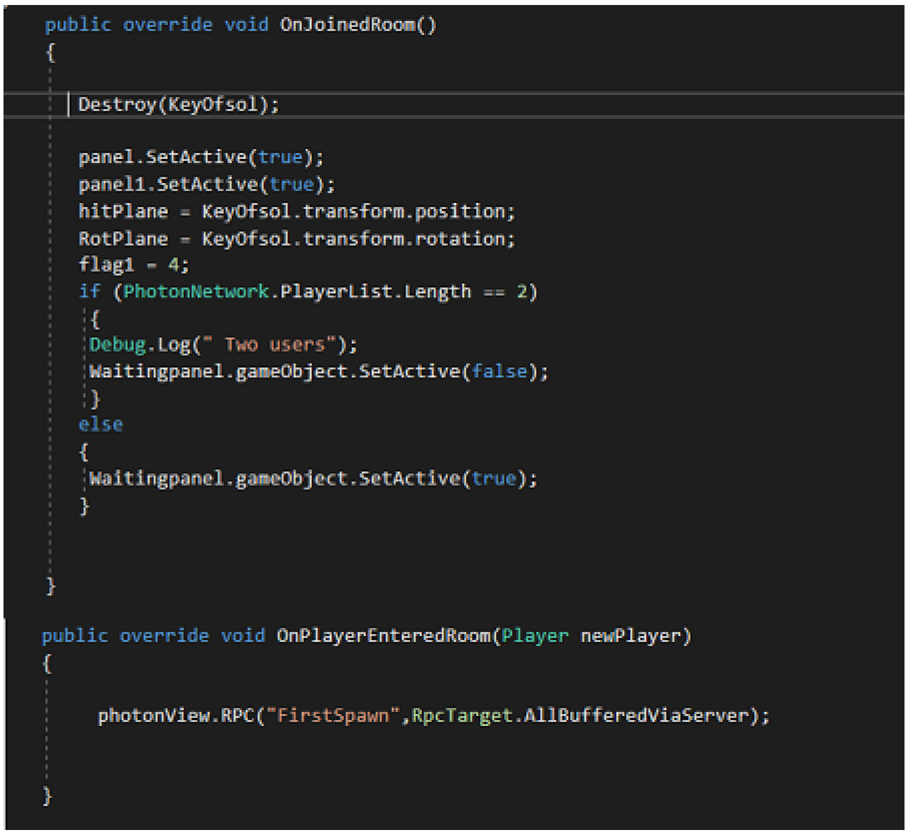
As we mentioned above, with the 'Compose' button being pressed, Photon's JoinRandomRoom method runs. Clients either joins a room successfully or creates one if there is none. In our application, the first of the two users will create the room which will be joined by the second. During the room creation, we set its maximum capacity up to two since every session requires only two users.



```
public void JoinRandomRoom()
{
    PhotonNetwork.JoinRandomRoom();
}


#endregion


#region PHOTON Callback Methods

public override void OnJoinRandomFailed(short returnCode, string message)
{
    Debug.Log(message);
    CreateAndJoinRoom();
}
void CreateAndJoinRoom()
{
    string randomRoomName = "Room" + Random.Range(0, 1000);
    RoomOptions roomOptions = new RoomOptions();
    roomOptions.MaxPlayers = 2;



    PhotonNetwork.CreateRoom(randomRoomName,roomOptions);
}
```

Figure 5.29: Matchmaking methods

As each user joins the room, Photon's Callback method OnJoinedRoom is being activated, in order to locally destroy the Treble Clef and save its position. The user who creates the room waits for the second user to join. By the time that happens Photon's Callback method OnPlayerEnterRoom is being called. Simultaneously, the waiting panel is being disabled and two pentagrams spawn locally on both users. The music composition feature is about to begin.

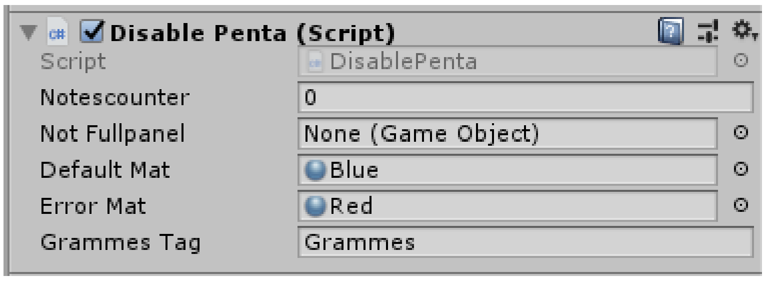


```
public override void OnJoinedRoom()
{

    Destroy(KeyOfsol);

    panel.SetActive(true);
    panel1.SetActive(true);
    hitPlane = KeyOfsol.transform.position;
    RotPlane = KeyOfsol.transform.rotation;
    flag1 = 4;
    if (PhotonNetwork.PlayerList.Length == 2)
     {
     Debug.Log(" Two users");
     Waitingpanel.gameObject.SetActive(false);
     }
    else
    {
     Waitingpanel.gameObject.SetActive(true);
    }



}
public override void OnPlayerEnteredRoom(Player newPlayer)
{

     photonView.RPC("FirstSpawn",RpcTarget.AllBufferedViaServer);


}
```

Figure 5.30: First Pentagram Spawns

5.6.4 Pentagram's Dynamic spawn

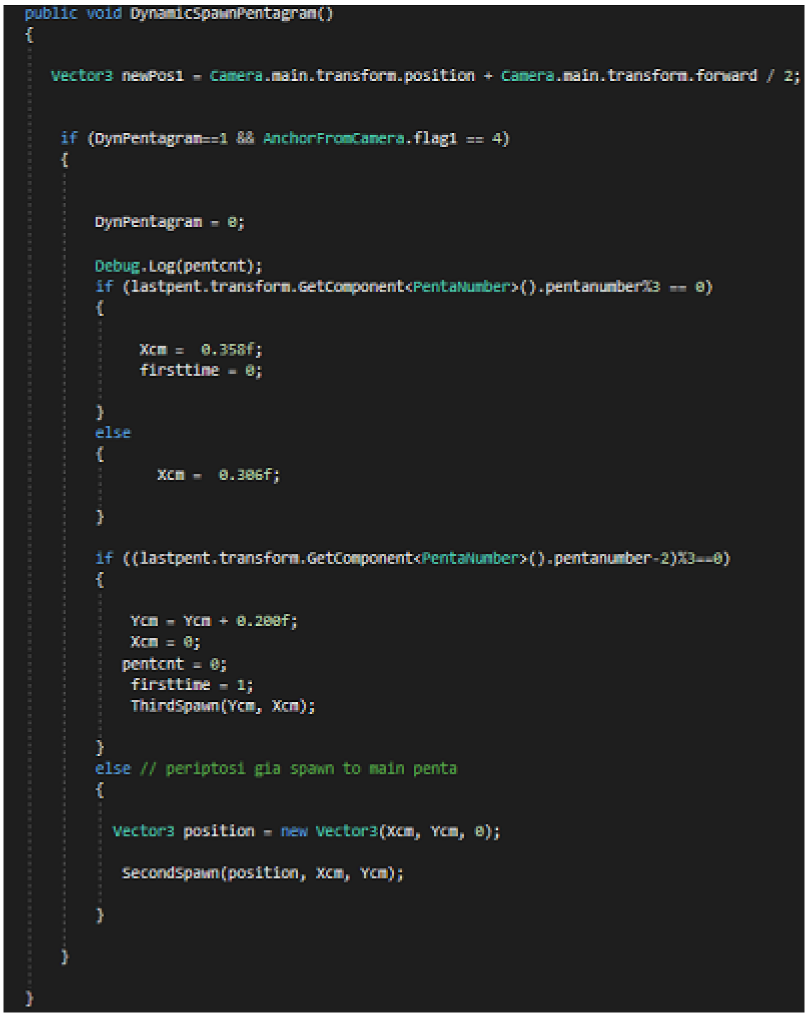
Pentagrams are the game objects where the users spawn their notes. Users can spawn notes on top of the pentagrams until the summary of the note's values equals to four. Every pentagram is attached with the 'DisablePenta' script which indicates the value of the pentagram.



Disable Penta (Script)	
Script	DisablePenta
Notescounter	0
Not Fullpanel	None (Game Object)
Default Mat	Blue
Error Mat	Red
Grammes Tag	Grammes

Figure 5.31: Pentagram's Value

At the time a pentagram reaches the maximum value, its colliders are being disabled in order for the user to not able to spawn another note on it and an RPC is being cast. This RPC calls the 'DynamicSpawnPentagram' method which will locally spawn a pentagram on both users. The position of the new pentagram can either be next to the previous pentagram if the horizontal pentagrams are less than three or below from the first pentagram, creating a new line of pentagrams.



```
public void DynamicSpawnPentagram()
{

   Vector3 newPos1 = Camera.main.transform.position + Camera.main.transform.forward / 2;


    if (DynPentagram==1 && AnchorFromCamera.flag1 == 4)
    {


        DynPentagram = 0;

        Debug.Log(pentcnt);
        if (lastpent.transform.GetComponent<PentaNumber>().pentanumber%3 == 0)
        {

             Xcm =  0.358f;
             firsttime = 0;

        }
        else
        {
               Xcm =  0.386f;

        }

        if ((lastpent.transform.GetComponent<PentaNumber>().pentanumber-2)%3==0)
        {

            Ycm = Ycm + 0.200f;
            Xcm = 0;
           pentcnt = 0;
            firsttime = 1;
            ThirdSpawn(Ycm, Xcm);

        }
        else // periptosi gia spawn to main penta
        {

          Vector3 position = new Vector3(Xcm, Ycm, 0);

           SecondSpawn(position, Xcm, Ycm);

        }

    }

}
```

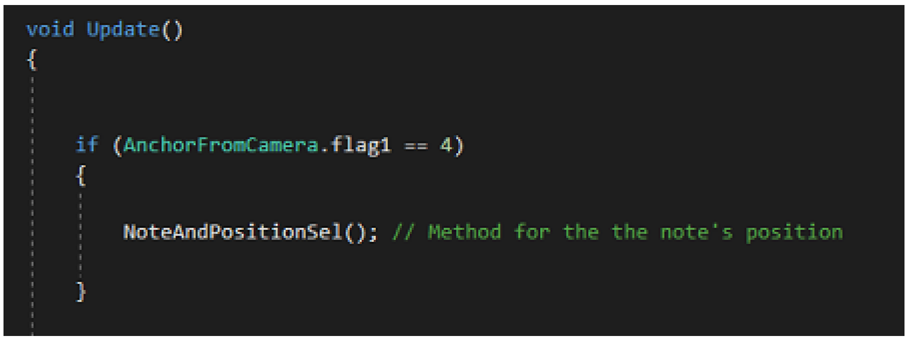
Figure 5.32: DynamicSpawnPentagram Method

In our application we have created two types of pentagrams, the one attached with the Treble Clef which is spawned only as the first pentagram of the line, and the Main Pentagram which continues horizontally in the line.

5.6.5 Note's Update Method

In line with the previous subsections, we have already achieved the connection of our two users. As this connection is being established, both users have their first pentagram spawned at the local position of their Treble Clef. Each local position can be fundamentally different for each user so we had to establish a way for the position of every upcoming note to be synchronized on both pentagrams. The position's synchronization of the upcoming notes, which will be spawned by both users, was our biggest and most challenging issue.

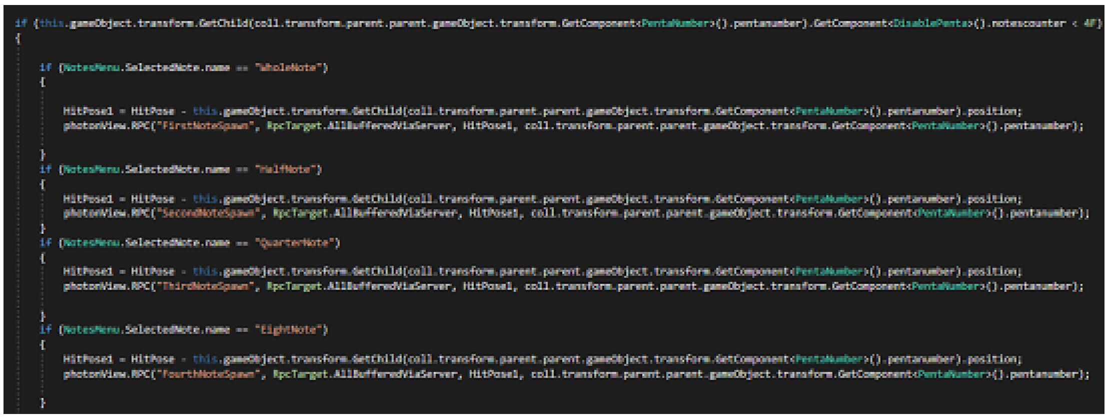
An AR Experience is inextricably linked with the physical world. The origin of the Scene is the position of the phone when the user initially launches the application. Our users are placing their Treble Clefs according to their own environment. Therefore, their first pentagram's local position differs, which affects the position of the note on the remote user. Users decide the position of the note on top of the pentagram by aiming with a red dot. As the user aims the lines and the spaces of the pentagram, a Unity's Raycast process runs. If the ray hits a line or space collider, it creates a hover effect and saves the position of the Raycast's 'hit'. The saved 'hit' represents a position in the 3D space. This method runs under the Update method, which means that it is called on every frame of our application.



```
void Update()
{


    if (AnchorFromCamera.flag1 == 4)
    {

        NoteAndPositionSel(); // Method for the the note's position

    }
```

Figure 5.33: Note's Position

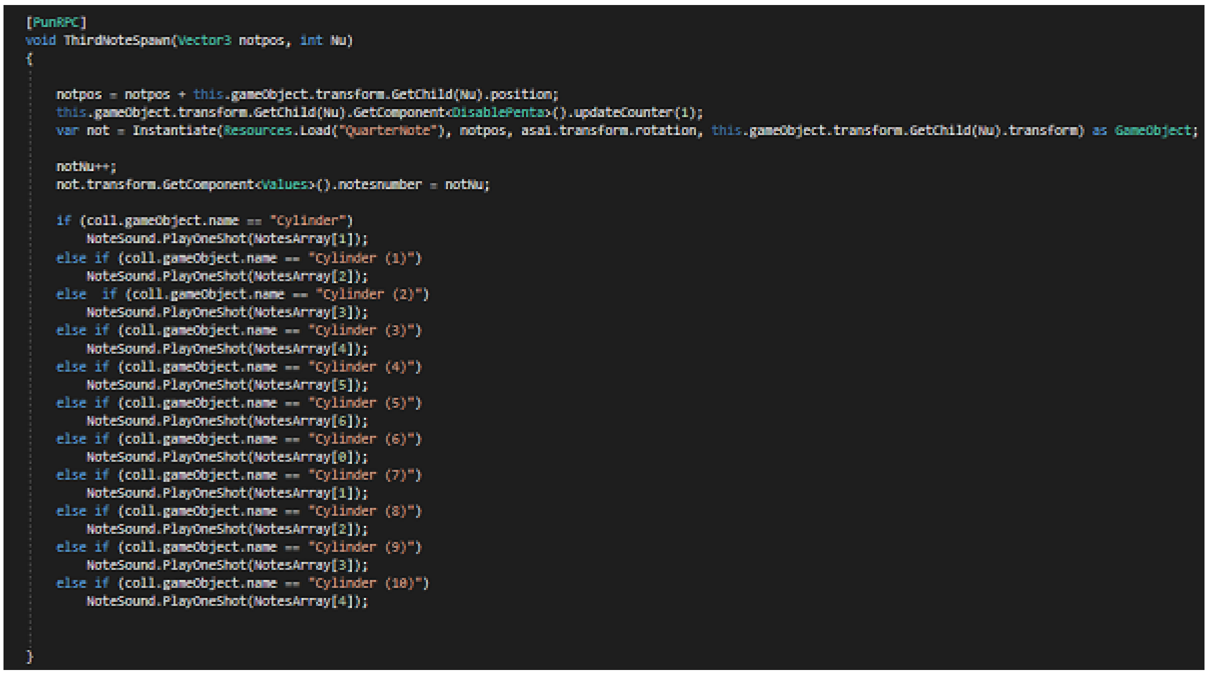
By the time one of our users has selected a note and has chosen its position, a touch on the screen will call the 'UpdateNote' method. The aforementioned Raycast's hit position contains the note's spawning position and the local pentagram's position of the first user. In order to solve the note's position synchronization issue we have to remove the local pentagram's position from the saved hit position and send only the note's spawning position to the remote user. The local pentagram position of the second user is then added to the received position. This results in the synchronised position of the note on the remote user.



```
if (this.gameObject.transform.GetChild(coll.transform.parent.parent.gameObject.transform.GetComponent<PentaNumber>().pentanumber).GetComponent<DisablePenta>().notescounter < 4F)
{

    if (NotesMenu.SelectedNote.name == "WholeNote")
    {

        HitPose1 = HitPose - this.gameObject.transform.GetChild(coll.transform.parent.parent.gameObject.transform.GetComponent<PentaNumber>().pentanumber).position;
        photonView.RPC("FirstNoteSpawn", RpcTarget.AllBufferedViaServer, HitPose1, coll.transform.parent.parent.gameObject.transform.GetComponent<PentaNumber>().pentanumber);

    }
    if (NotesMenu.SelectedNote.name == "HalfNote")
    {
        HitPose1 = HitPose - this.gameObject.transform.GetChild(coll.transform.parent.parent.gameObject.transform.GetComponent<PentaNumber>().pentanumber).position;
        photonView.RPC("SecondNoteSpawn", RpcTarget.AllBufferedViaServer, HitPose1, coll.transform.parent.parent.gameObject.transform.GetComponent<PentaNumber>().pentanumber);
    }
    if (NotesMenu.SelectedNote.name == "QuarterNote")
    {
        HitPose1 = HitPose - this.gameObject.transform.GetChild(coll.transform.parent.parent.gameObject.transform.GetComponent<PentaNumber>().pentanumber).position;
        photonView.RPC("ThirdNoteSpawn", RpcTarget.AllBufferedViaServer, HitPose1, coll.transform.parent.parent.gameObject.transform.GetComponent<PentaNumber>().pentanumber);

    }
    if (NotesMenu.SelectedNote.name == "EightNote")
    {
        HitPose1 = HitPose - this.gameObject.transform.GetChild(coll.transform.parent.parent.gameObject.transform.GetComponent<PentaNumber>().pentanumber).position;
        photonView.RPC("FourthNoteSpawn", RpcTarget.AllBufferedViaServer, HitPose1, coll.transform.parent.parent.gameObject.transform.GetComponent<PentaNumber>().pentanumber);

    }
```

Figure 5.34: UpdateNote method

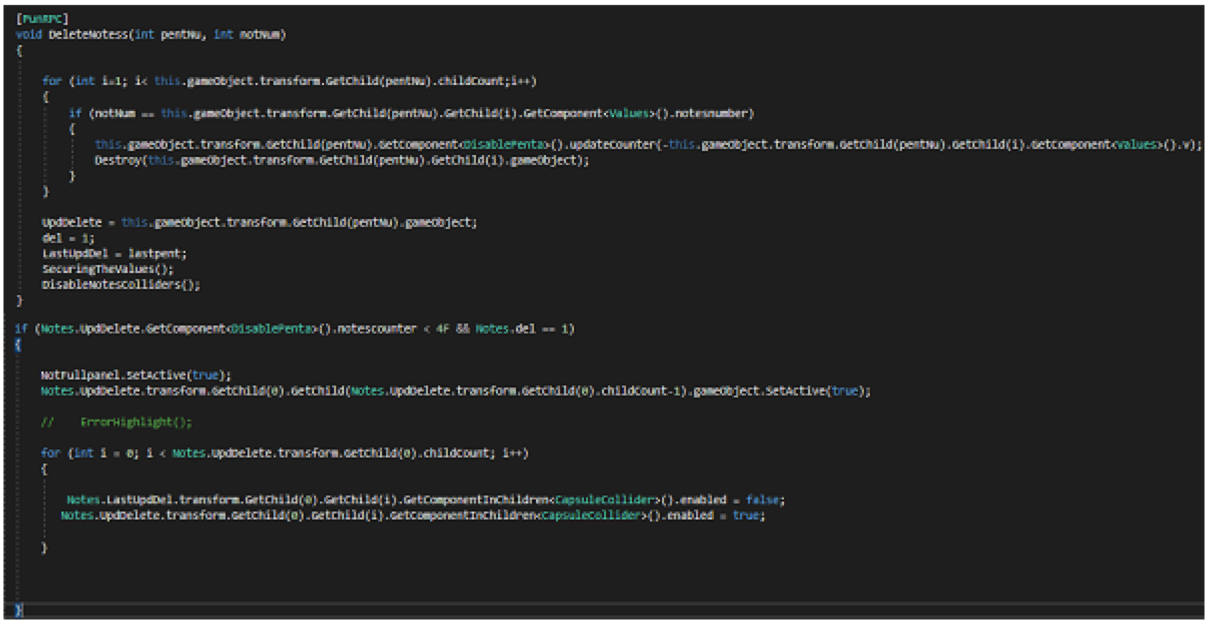
During the UpdateNote method we examine if the pentagram that our user wants to spawn the note on is not full and we cast a Remote Procedure Call. An RPC calls a method on remote clients in the same room. This RPC send as parameters, the spawning position of the note and the unique number of the aimed pentagram. The executed method adds the local position of the pentagram with the extracted spawning position of the note and spawns simultaneously the selected note at this position on both users. As the note is being spawned, a corresponding sound is played for each user.



```
[PunRPC]
void ThirdNoteSpawn(Vector3 notpos, int Nu)
{

    notpos = notpos + this.gameObject.transform.GetChild(Nu).position;
    this.gameObject.transform.GetChild(Nu).GetComponent<DisablePenta>().updateCounter(1);
    var not = Instantiate(Resources.Load("QuarterNote"), notpos, asai.transform.rotation, this.gameObject.transform.GetChild(Nu).transform) as GameObject;

    notNu++;
    not.transform.GetComponent<Values>().notesnumber = notNu;

    if (coll.gameObject.name == "Cylinder")
        NoteSound.PlayOneShot(NotesArray[1]);
    else if (coll.gameObject.name == "Cylinder (1)")
        NoteSound.PlayOneShot(NotesArray[2]);
    else  if (coll.gameObject.name == "Cylinder (2)")
        NoteSound.PlayOneShot(NotesArray[3]);
    else if (coll.gameObject.name == "Cylinder (3)")
        NoteSound.PlayOneShot(NotesArray[4]);
    else if (coll.gameObject.name == "Cylinder (4)")
        NoteSound.PlayOneShot(NotesArray[5]);
    else if (coll.gameObject.name == "Cylinder (5)")
        NoteSound.PlayOneShot(NotesArray[6]);
    else if (coll.gameObject.name == "Cylinder (6)")
        NoteSound.PlayOneShot(NotesArray[0]);
    else if (coll.gameObject.name == "Cylinder (7)")
        NoteSound.PlayOneShot(NotesArray[1]);
    else if (coll.gameObject.name == "Cylinder (8)")
        NoteSound.PlayOneShot(NotesArray[2]);
    else if (coll.gameObject.name == "Cylinder (9)")
        NoteSound.PlayOneShot(NotesArray[3]);
    else if (coll.gameObject.name == "Cylinder (10)")
        NoteSound.PlayOneShot(NotesArray[4]);


}
```

Figure 5.35: Quarter's Note RPC

The deletion of a note is a process quite similar to the spawning and it runs through the UpdateNote method as well. In order for a note to be deleted, our user has to aim it and touch the screen. An RPC is being called via UpdateNotes with the unique number of the note and the pentagram that the note was spawned. The method the RPC calls searches for the note, on both users pentagrams, with its unique number and destroy the note's game object if the search is successful. If our user decides to edit an already full pentagram, the composition process stops until this pentagram is back to the 4 out of 4 value.



```
[PunRPC]
void DeleteNotess(int pentNu, int notNum)
{

    for (int i=1; i< this.gameObject.transform.GetChild(pentNu).childCount;i++)
    {
        if (notNum == this.gameObject.transform.GetChild(pentNu).GetChild(i).GetComponent<Values>().notesnumber)
        {
            this.gameObject.transform.GetChild(pentNu).GetComponent<DisablePenta>().updatecounter(-this.gameObject.transform.GetChild(pentNu).GetChild(i).GetComponent<Values>().v);
            Destroy(this.gameObject.transform.GetChild(pentNu).GetChild(i).gameObject);
        }
    }

    UpdDelete = this.gameObject.transform.GetChild(pentNu).gameObject;
    del = 1;
    LastUpdDel = lastpent;
    SecuringTheValues();
    DisableNotesColliders();
}

if (Notes.UpdDelete.GetComponent<DisablePenta>().notescounter < 4F && Notes.del == 1)
{

    NotFullpanel.SetActive(true);
    Notes.UpdDelete.transform.GetChild(0).GetChild(Notes.UpdDelete.transform.GetChild(0).childCount-1).gameObject.SetActive(true);

    //    ErrorHighlight();

    for (int i = 0; i < Notes.UpdDelete.transform.GetChild(0).childCount; i++)
    {

         Notes.LastUpdDel.transform.GetChild(0).GetChild(i).GetComponentInChildren<CapsuleCollider>().enabled = false;
        Notes.UpdDelete.transform.GetChild(0).GetChild(i).GetComponentInChildren<CapsuleCollider>().enabled = true;

    }


}
```

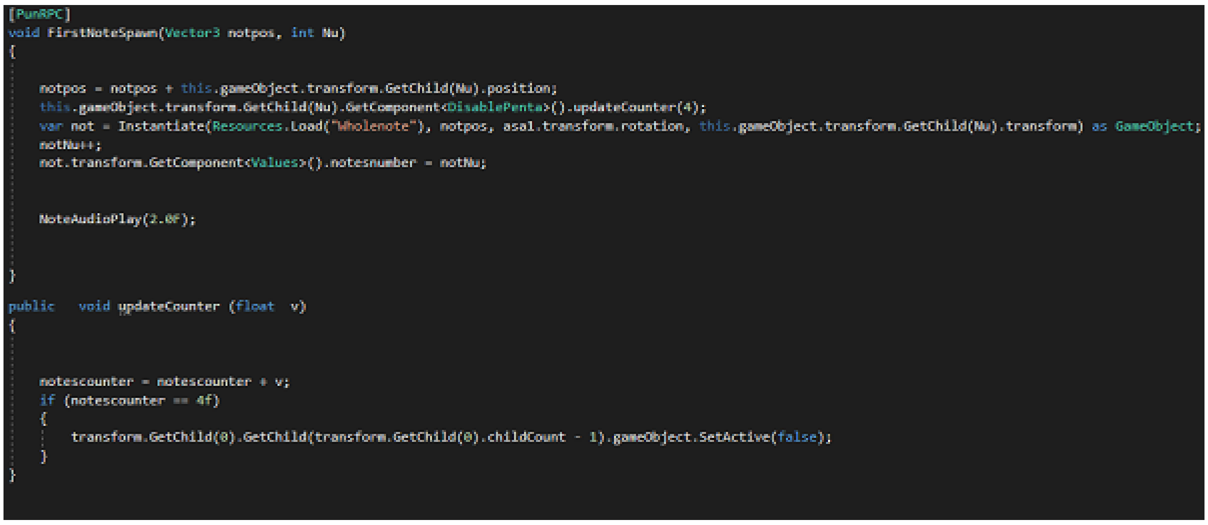
Figure 5.36: Deletion Process

5.7 Network Synchronization

At the time this thesis is being composed, the research around the AR field is still ongoing. AR technology has to deal with several stability issues in order to be fully functional. As a result, adding a network system that will allow the collaboration between two remote users had to be implemented respectfully. An unstable internet connection can be fatal for an AR experience because it will multiply the already mentioned stability issues.

Since we can not completely get rid of the lag problems we had to use the traits of the existing technology to provide our users with a synchronized collaboration. Our primary goal was to maintain each pentagram at the same state for each user during the whole AR session. The first thing that we had to establish was the order and the targets of our Remote Procedure Calls. In our implementation we use RPCs that are being buffered and sent through the server. When a user casts an RPC, this RPC won't be executed immediately. Instead, it will be sent through the server to both users, which means that the method will be executed simultaneously for both users. During a scenario that both users send an RPC at the same time, the order of their execution will be the same as their arrival order on the server. Furthermore, both users are being stalled for 1s after the time they cast an RPC, to avoid excess RPCs by being cast mistakenly. This secures a solid time for the call to 'travel' via the server to both users. We are aware that under high ping situations out-of-bounds note spawns may happen. To account for this, our implementation secures that our application will maintain its functionality regardless of the network conditions, as long as the users do not disconnect from the network.

In the case that an out-of-bounds note is spawned on a pentagram due to the aforementioned network conditions, our methods secure that the upcoming pentagram will be locked until the value of the previous pentagram is set. This is possible since when an RPC is called to spawn a new note, the value of the newly spawned note is added to the sum of the pentagram value. If the pentagram value is out-of-bounds, the user cannot continue composing until a correction is made.



```
[PunRPC]
void FirstNoteSpawn(Vector3 notpos, int Nu)
{

    notpos = notpos + this.gameObject.transform.GetChild(Nu).position;
    this.gameObject.transform.GetChild(Nu).GetComponent<DisablePenta>().updateCounter(4);
    var not = Instantiate(Resources.Load("Wholenote"), notpos, asal.transform.rotation, this.gameObject.transform.GetChild(Nu).transform) as GameObject;
    notNu++;
    not.transform.GetComponent<Values>().notesnumber = notNu;


    NoteAudioPlay(2.0F);


}

public   void updateCounter (float  v)
{


    notescounter = notescounter + v;
    if (notescounter == 4f)
    {
        transform.GetChild(0).GetChild(transform.GetChild(0).childCount - 1).gameObject.SetActive(false);
    }
}
```

Figure 5.37: Secured Pentagram Bounds

Chapter 6

Conclusion

6.1 Summary

In this thesis we presented the design of a mobile Collaborative Augmented Reality Application for music composition. This thesis's ultimate goal was to achieve a real-time collaboration between two remote users and make them enhance their physical surroundings with 3D models. Into this collaborative Augmented Reality environment our user's goal was to compose a music piece. By implementing the AR feature into the music composition we aimed to enhance this process in a way that the physical environment will turn into a digital music room. In order to minimize the complexity and focus on the functionality, our application maintains the music notation at a very basic level but our development was focused on being expandable. Further notation's features with great complexity aimed for expert musicians can be added on top of our prototype with the proper changes.

Music Composition and Augmented Reality could have various use cases and we wanted to present an idea combining the two fields, showing their capabilities and their future potential. During this process we had to overcome a great amount of unpredictable issues since AR technology has just reached the wider public. Networking on mobile phones while using AR technology was the most challenging part of our development since the processing power of the devices and the network issues were affecting our experience. Thus, our final result is inextricably linked with the available technologies. Furthermore, this thesis has been developed during the pandemic of Covid-19, therefore, the remote collaboration became a necessity for the vast majority of the population.

There is much research being done around the Augmented Reality topic and this technology is still rapidly growing. During the composure of this thesis, we were witnessing constant changes of the AR technology such as the addition of innovative algorithms and specialized mobile devices with great computing power. In addition, with the deployment of the fifth generation technology standard for broadband cellular networks, network systems are facing a radical upgrade which will reduce the latency and enhance the network's speed. The aforementioned, along with the availability and the technological advances of modern smartphones create and an ideal environment for the growth of the Collaborative Augmented Reality.

6.2 Evaluation Method

In order to test the finally developed thesis we organised private sessions with colleagues and friends. Since our application is collaborative, every session required two participants. We had the opportunity to run multiple remote tests with one participant being in Athens and the other one in Crete. Furthermore, a lot of sessions took place in the Technical University of Crete, using different rooms in the same building.

Because our thesis combines the AR technology with music, we chose to evaluate PentAgRam with both experienced AR users and musicians. During the evaluation period, we had the opportunity to form different 'pairs' of users and observe their experience. In that way, users provided us with a well-rounded feedback, covering a wider field of our implementation.

A fact of great importance for the evaluation process is the frequency that it was happening. Since the beginning of our application we tried to test every new functional addition. Having our thesis evaluated by users as it was getting developed, pointed us towards their needs for a smooth AR experience and also existing errors which we had to fix.

6.3 Evaluation Results

After an extended testing period by both us and our users we have gathered the evaluation results. The received feedback lead us to significantly improve the functionality of our application. The evaluation results are listed bellow:

• Inexperienced, with AR, users were facing difficulties to properly set up the AR Scene since the Plane Detection process was not known to them. In addition, we noticed that the aiming procedure for the note placement had to be further explained. On the contrary, users with basic knowledge around the AR technology were able to properly set up the AR Scene but they required more information about the composing scenario. The given solution was to implement the 'Help Scene', where we provide instructions about the AR features and precise information about the music notation.

• User's feedback about the implemented 3D models was quite positive. More specifically, the notes and the pentagrams seemed elegant to them while their simplicity helped them focus on the composing procedure. At this point, we added the note's sound on every note spawn, since some of the composers were expecting to be able to listen to their selected note. The 3D Piano looked quite interesting to them and they proposed for more interactions with it as a future work.

• The vast majority of our users faced no issues while they were navigating through our Scenes. The UI was pretty intuitive and the button's signs were specific and clear. Furthermore, the colors and the images were a nice touch, which contributed to the overall experience.

• Some of our users had drifting issues and elevation problems with the 3D models into the AR Scene. This is a known issue into the AR industry which can happen while users rotate or move their mobile phones fast. Additionally, while the user's device was loaded with a great amount of background processes some positioning issues were noticed.

• Expert musicians were not able to compose at the level of their expertise as our Music Notation system was kept at the lowest possible level for the sake of simplicity.

• While users were facing lag issues, we noticed a wrong order of events which was causing random note and pentagram spawns. This conflict happened because we were casting the RPCs locally where there is no lag. This issue was solved by buffering the RPC's via the server in order for them to be executed in the order off arrival on the server. Finally, the screen touch was limited to one per second to avoid massive 3D model spawns especially under networking errors.

The Evaluation method was critical for our thesis because the repetitive tests indicated the weak points of our application and directed us towards their improvement. Some of our users focused on the Composing experience and others on the AR functionality, according to their background knowledge. Overall, our users seemed enthusiastic with the blend of the music into AR's Technology. Most of them agreed that they had been through a unique and innovative AR experience which requires future work in order to become a final product.

We are aware that the Collaborative Augmented Reality technology needs improvements before it can adopted by a broader audience. With the available tools given, we believe that our application's prototype showcased the capabilities of Collaborative AR.

6.4 Future Work

The PentAgRam, the Collaborative Augmented Reality application presented in this thesis should be considered a proof of concept as it remains at a prototype phase. This application has been developed with the at the time available software and hardware. As we have already mentioned, the field of Augmented Reality is rapidly growing and new networking capabilities are rising. Thus, PentAgRam should follow this progress in order to become a finished product. Below we present some future work that has not be implemented due to lack of software/hardware and time, which we believe that will improve PentAgRam.

The first thing that could be improved is the Plane Detection process. For the time being, vertical objects are hard to be precisely detected. AR foundation relies on feature points (rapid changes in color, contrast, shape) in order to detect planes in the physical world. Vertical objects such as single-colored walls without enough feature points can not be detected which complicates the spawn of our Treble Clef. This is a known issue for the AR industry and further upgrades are being expected.

Secondly, Music Composure feature can be improved by adding an advanced notation system. Currently, PentAgRam contains very basic notation features which can be enhanced with more Clefs, Key Signatures and Time Signatures as we have mentioned at the Chapter 3. This improvement requires the creation of new 3D models that will visualize the aforementioned notation features.

The collaboration during our implementation is limited to two users per server room. It is a fact that plenty music pieces required more than two composers in order to complete. Therefore, on future approaches around our application, server rooms can be modified to accept more than two users. The changes on our code to achieve that can be easily made.

Furthermore, we believe that it would be interesting to implement PentAgRam on AR Head Mound Displays. Handheld AR will allow to our musicians to play their instruments while they are composing. When we first started researching on our approach, we examined HMDs as optional targeted devices for our application. This required a whole new different perspective on the developing phase and the usage of different SDKs which were not fully tested by then.

Last but not least, more AR Content can be added besides music composure. Different 3D music instruments with innovative features can be developed. During the interaction with them, the user would be able to listen to the composed music piece.

Altogether, the suggested improvements and the future work that could be done is inextricably linked with the AR technology progression. At the given state of AR technology, we believe that adding more notation features and visual context will enhance the AR experience and attract more users.

Bibliography

[1] Ivan E. Sutherland, A head-mounted three dimensional display, In AFIPS ’68 (Fall, part I), 1968. 7

[2] T. Caudell and D. Mizell, Augmented reality: An application of heads-up display technology to manual manufacturing processes, In System Sciences, 1992. 7

[3] P. Milgram and F. Kishino, A taxonomy of Mixed Reality Visual Displays, IEICE Transactions on Information and Systems vol. E77-D, 1994 8

[4] Ronald T. Azuma, A survey of Augmented Reality, In Presence: Teleoperators and Virtual Environments 6, 1997 8

[5] Hollerer S. Feiner, B. MacIntyre and A. Webster, A touring machine: prototyping 3d mobile augmented reality systems for exploring the urban environment, IEEE International Symposium on Wearable Computers, 1997. 8

[6] H. Kato and M. Billinghurst, Marker tracking and HMD calibration for a video-based Augmented Reality conferencing system, In Proceedings 2nd IEEE and ACM International Workshop on Augmented Reality (IWAR’99), 1999 9

[7] B. Thomas, B. Close, J. Donoghue, J. Squires, P. De Bondi, M. Morris, and W. Piekarski, Arquake: an outdoor/indoor augmented reality first person application, In Digest of Papers, Fourth International Symposium on Wearable Computers, 2000 9

[8] Clemens Arth, Raphael Grasset, Lukas Gruber, Tobias Langlotz, Alessandro Mulloni and Daniel A. Wagner, The history of mobile augmented reality developments in mobile AR over the last almost 50 years, 2015. 10

[9] Daniel Wagner, Thomas Pintaric, Florian Ledermann, and Dieter Schmalstieg, Towards massively multi-user augmented reality on handheld devices,In Hans W. Gellersen, Roy Want, and Albrecht Schmidt, editors, Pervasive Computing, Berlin, Heidelberg, 2005. 10

[10] Mark Billinghurst, Hirokazu Kato, Collaborative Augmented Reality, Communications of the ACM, 2002. 12

[11] A. Henrysson, M. Billinghurst, and M. Ollila, Face to face collaborative AR on mobile phones. In Fourth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'05), 2005. 12

[12] Lyn Pemberton and Marcus Winter, Collaborative augmented reality in schools, International Society of the Learning Sciences (ISLS), 2019 12

[13] Microsoft Hololens: 14, 15

`https://www.microsoft.com/en-us/hololens`

[14] Magic Leap: 16

`https://www.magicleap.com/en-us/magic-leap-1`

[15] Te-Lien Chou and Lih-Juan Chanlin, Augmented reality smartphone environment orientation application: A case study of the fu-jen university mobile campus touring system, Procedia - Social and Behavioral Sciences, 46:410–416, 12 2012. 17

[16] Dhiraj Amin and Sharvari Govilkar, Comparative study of augmented reality sdk's, International Journal on Computational Science Applications, 5:11–26, 2015. 18

[17] ARCore Documentation: 19, 58

`https://developers.google.com/ar/reference/`

[18] ARKit Documentation: 20

`https://developer.apple.com/documentation/arkit`

[19] Vuforia Documentation: 21

`https://library.vuforia.com/getting-started/vuforia-engine-10-api`

[20] Wikitude Documentation: 21

`https://www.wikitude.com/`

[21] Expirements with Google, Just A Line: 26

`https://experiments.withgoogle.com/justaline`

[22] Music Everywhere, Instant Musician: 31

https://www.music-everywhere.co/home

[23] Unity: 32, 56

https://unity.com/

[24] Unreal Engine: 33

https://www.unrealengine.com/en-US/unreal

[25] Unity's ARFoundation: 60

https://unity.com/unity/features/arfoundation

[26] Photon Unity Networking: 63

https://www.photonengine.com/pun