Universitat Jaume I (Castellón de la Plana, España)

Degree in Video Game Design and Development

FINAL DEGREE PROJECT’S FINAL REPORT

INTERACTIVE VIRTUAL ENVIRONMENT FOCUSED ON GAMIFIED TOURISM

Authored by: Francisco Alfaro Moscardó

Tutored by: Cristina Rebollo Santamaría

A mi hermano y a mis padres.
Por inspirar. Por luchar.
Por creer y crear por encima de todo.
Por las lecciones en cada caída.

A Aure, a José María y a Nacho.
Por la confianza depositada en mí.
Por vuestro apoyo infinito.
Por las sabias palabras en los momentos más complicados.

Esto ha sido posible gracias a vosotros.
Gracias, de corazón.

"... Es el momento de hacer algo."

Misión cumplida.

Summary

The present document constitutes the Final Degree Project's Technical Report of the Video Games Design and Development Degree offered by the Universitat Jaume I. The work to be developed consists in creating an interactive virtual gamified environment focused on tourism, through geolocation and augmented reality tools based on smartphones. It covers aspects about the planning, decision making and conceptual design of the final product and implementation details involved in the development of the complete application during the course of a few months by *Francisco Alfaro Moscardó*, student of the before mentioned Degree.

The aim of the project is providing both residents and visitors some knowledge about the main monuments and places of interest of the Valencia city historical centre (Spain). The application has been developed using Unity 3D game engine combined with a set of current libraries that allow to have a better performance and management of the event interaction and Augmented Reality. The geolocated system is based on the Global Positioning System (GPS) and is fully integrated with two different augmented reality mini games that are part of the whole experience and give players some added challenges before giving them the historical information about the monuments. Furthermore, the application communicates remotely with Google servers to provide services such as the modification of the database where the areas data is stored or the use of the Google user account for unlocking achievements and be part of a worldwide ranking.

Keywords

Geolocation, GPS, Augmented Reality, tourism, Unity3D.

Index

Figures Index

1. Technical Proposal

1.1. Introduction

This project aims at designing, creating and learning about a geolocated augmented reality application for smartphones by mixing both a touristic and educational experience and casual minigame playing.

In this introductory section, several aspects of the project will be covered as an overview, such as the context it was created in and the main motivations that led me to develop this concept. In addition, the objectives I set myself up when facing this project and the justification of why this project was a good proposal to develop and enhance some of the skills I learned and improved during the last four years as student are also exposed.

1.2. Context

Tourism is a sector that attracts thousands of people every day to Valencia, but, over the years, the use of conventional leisure systems and attractions to visit a city have been degraded and new ways of knowing an urban environment have appeared. Technology is in many cases related to these new trends, and facilitates the inclusion of new users in the concern to know the environment they visit or live, through various systems of the so-called Smart Cities [1].

This situation is highly related to a change on the paradigm of the access to the information that we are living though the new generations. Nowadays, books have been deprecated in favour of other useful online tools that provide people with all the information about anything they need at the moment in an extremely simple way. However, we have to be aware that new generations who are born in a pure technological bosom have fully assimilated this access and many times they find it difficult or even tedious to navigate through pages and pages with letters and images expressed in the same way as their predecessors, textbooks. This is why we, as developers, have the great opportunity and responsibility for improving the way we pass on knowledge (not only information) to young people in order to generate the interest on the different fields of study that the current society requires.

1.3. Motivation

In recent years there has been an exponential growth of general interest in new visualization technologies, such as virtual reality, augmented reality or the mixed reality [2], both by large companies in the sector and by the established niches of players who are looking for expanding their gaming experience to other areas. In this context, it is not surprising that a large number of new devices have been proposed and started to be developed by leading technology companies representing the industry: Google, Microsoft, Lenovo, Xiaomi and recently Apple are some of them.

This situation has brought a large number of developers begin to create new products adapted to the so-called technological demand. Some examples of these applications are the innovative toy called AR Terminator, which uses a physical gun with an adapter for the smartphone, or AR City, which wants to revolutionize navigation using augmented reality maps (Figure 1). Nowadays, developers are using these new tools to generate original content, not only video games, but content applied to different fields, such as medicine, education and culture, construction, hospitality, catering, etc. The interest in the nature of these new technologies is obvious and inevitable, in addition to the unimaginable amount of uses and wealth that can be given to the society.

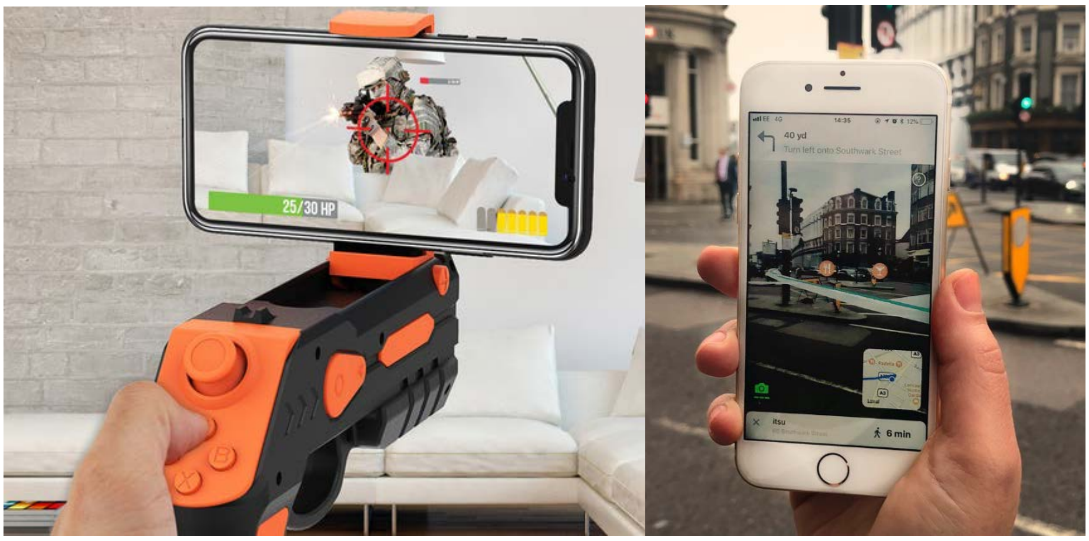


Figure 1: Augmented Reality is living an era of experimentation and transformations.

For the reasons expressed above, today there is a strong trend at the level of development that advocates to expand those market niches in new technologies and give them practical uses adapted to the current social reality. One of the sectors that demand this type of technological restructuring the most is education, since, as we have seen, knowledge can be transmitted in a very varied and equally plausible ways. One of them is making use of contemporary innovations like these. With all this in mind, in this project we will focus on the transmission of cultural knowledge through augmented reality, focusing on two main purposes: education and tourism.

Perhaps Pokémon Go [3] is the clearest example of an augmented reality and geolocation based application (Figure 2). Its system of spontaneous capture of Pokémon [4] and visit of certain places to collect different utilities captivated millions of people and managed to make them leave their homes to capture new Pokémon while visiting places physically. This project aims to follow that line, but adding a more playful section to the fact of going to the place and taking an action. The user must, apart from going to the point of interest, face a challenge there, in the form of a videogame. The user has to play (many times, play concerted) and after that, he/she will always be awarded with some interesting information about that place, so that the entire system will act as a single piece of learning.

Figure 2: Pokémon GO was one of the mass phenomena worldwide that showed the possibilities of geolocated augmented reality.

Thus, in this project it has been researched how to know Valencia's city centre from another point of view and how to achieve producing more people's curiosity to know the historical patrimony where they live or they are visiting, through several gamified systems and an attractive augmented reality context, and bring common gamers another ways of playing outside an screen, by using the physical environment as their main game board to develop and improve their gaming skills.

1.4. Objectives

The main objective of the project is to develop a smartphone application that manages geolocated information combined with augmented reality to transmit the most relevant information about certain places of the city centre of Valencia while playing video games within a narrative context.

To achieve this, several ways of motivating users to go out and get to know new places have been applied by many applications. However, there are not many geolocated applications in the mobile market based on augmented reality and at the same time focused on the preservation and transmission of information related to cultural heritage from divulgation and education.

Personally, I believe that geolocated augmented reality has a great potential to exploit in this field and with the correct creation of a gamified system it is possible to achieve an increase in the people's curiosity about the cultural heritage. This is the final goal of the project: to raise awareness about the importance of cultural heritage, transmit information while playing and promote curiosity about what they visit.

1.4.1. Project justification

Throughout this section, the skills and abilities to be developed during the project creation process will be covered. These are the fundaments why I think it is a good job to finish my Bachelor's academic studies in the field of video game development. Firstly, the skills required for the correct performance of the subject to which the present project belongs will be presented, and later some details about the skills developed will be given, classified in each of the subjects of the degree that are most related to the creation of this game.

As the teaching guide establishes, the Final Degree's Project subject expects a set of results that every student must obtain based on some skills developed before and during this subject. These are detailed below:

- **Skill 1:** Ability to individually develop, present and defend in front of a university tribunal an original exercise, consisting of a game design and development related project of professional nature, in which the student synthesizes and integrates the skills acquired during his/her studies.

- **Result 1:** Individually plan and implement an original game design and development related project of professional nature, in which the student synthesizes and integrates the skills acquired during his/her studies.

- **Result 2**: Write a technical report in English, presenting and defending in front of a university tribunal an original game design and development related project of professional nature, in which the student synthesizes the skills acquired during his/her studies.

1.4.2. Related subjects

- **VJ1208 - Programming II**

This course gave me the basics about the POO and the C # programming language, currently the most widespread for Unity scripting.

- **VJ1211 – Mathematics II**

As a basic competence in the fields of computer science and videogames, this course allowed me to develop my knowledge of trigonometry and algebra.

- **VJ1220 - Databases**

This subject gave me a wide knowledge about the conceptualization, implementation and management of relational databases.

- **VJ1222 - Conceptual Videogame Design**

Thanks to this subject I acquired a more critical and analytical vision of the game, as well as the elements that make it up and how they relate to each other.

- **VJ1227 - Game Engines**

In this subject, I developed my second integral project in Unity 3D and I learned the basics of a game engine, its execution cycle and its differentiating aspects with respect to others.

- **VJ1234 – Advanced Interaction Techniques**

This course allowed me to start learning techniques related to machine learning. I was able to better understand the pattern recognition algorithms thanks to this subject.

1.5. Software and tools

Unity 3D 2017.3.0f3 (64-bit) - Unity 2018.1.0f2 (64-bit) [5]

Unity3D is nowadays among the most important game engines and it is used to design and develop interactive applications and video games. It offers a good option for the development of the project in question because of its extensive active user community on platforms such as the Unity Forum, YouTube or GitHub, the technical resources (in its own Asset Store) and the learning. It also helps the ease to export to different platforms (such as Android [6], iOS [7] or Windows Phone [8]), the native support for these platforms, the good complementation with the official APIs of Google services and the simple handling of input data from external devices, such as the accelerometer, gyroscope, vibration or GPS [9] . In addition, the experience that I have acquired with the game engine during the last years of the Bachelor studies and through personal projects has been really useful.

Visual Studio Community 2017 [10]

Unity3D offers a good integration with multiple Integrated Development Environments (IDE). However, this Microsoft's IDE has the most complete support in this field and is widely used, thanks to its fast debug tools and comfortable movement between classes, functions and, in general, the POO-based structure of different interconnected C# scripts.

SketchUp Make 2017 (and extensions) [11]

This tool of graphic design and modelling in three dimensions is usually used to generate models based on architecture, civil engineering or industrial design thanks to its rapid conceptual prototyping and its use of Boolean structures based on faces. In addition, it allows, in a very simple and precise way, measuring at real size and reproducing them. This has been very useful when modelling certain buildings or points of interest in each of the most relevant areas. As we will see later, its millions of extensions and the extensive library of geolocated 3D models directly connected to Google Maps [12] has also been useful in certain aspects of the project. The tool could have a better integration with Unity, since it is somewhat limited in certain aspects and the handling of complex models is not good. However, for the purpose of the project it is more than enough, since it meets all the requirements and it has not been necessary to create figures with more level of detail.

Git and GitHub [13]

As one of the most powerful Git-based version control platforms on the market, GitHub is a good option to create code with caution and have better control over what is being developed. Despite carrying this project out alone, the tool is fantastic when it comes to organizing and provides a great help and time saving when having to discard the changes in the code for some unexpected reason.

Adobe Photoshop CC 2018 [14]

Without a doubt, this image processing tool (or similar) is essential in any work that has a certain visual component, since it allows with some speed or ease to edit any texture or sprite that is required and integrate it using the `.psd` file in Unity itself. Photoshop has been used since the modification of certain textures in visual effects or shaders (such as transitions or animated background textures) as well as others related to robot face animation or the creation of some Head-up Display (HUD) elements and dialogues.

Trello [15]

This project managing software based on agile development methodologies was used to keep track of the development of the project and task planning. Even for only one person, it is really useful, as it allows to identify and check all the little tasks in an organized and quick way.

Firebase [16]

For the development of the backend system, the Google's Firebase platform has been used, since thanks to its Firestore service [17], it was possible to store the information via the web in a secure manner and then request it through the application. Firebase acts as a real-time database that also allows great integrations with external API's written in languages such as JavaScript or Go.

Angular 6 and Angular CLI [18]

Angular framework, perfectly integrated with firebase, was used to create the web client that communicates with the server. The website and service controller was generated by using Angular CLI and was used to communicate with the Firebase data. Then, the component and the web template were created so that the developer could add and remove areas from a simple visual interface.

Microsoft Word 2013 [19]

This text-processor was used to write this memory, integrated with the **Mendeley Desktop** app [20]. Thanks to both, it was easier to correctly make citations and bibliographic references using the IEEE referencing standard.

External frameworks

In order to streamline and optimize certain parts of the project's development, it was investigated how to implement various basic functionalities that the design required and was conscientiously evaluated which tools to use to cover these needs. The Unity3D community is extensive and usually feeds on valuable resources. In addition, the Asset Store has lots of good free downloadable packages that save a lot of effort in terms of implementation.

As professionals of the sector, we must be attentive and up-to-date with all these tools provided by the community of users that sometimes make life easier and allow us to focus on developing time, among other things beyond mere implementation. Some of them were known through personal research and others thanks to recommendations from people involved in games development. All of them were tested and evaluated before implementing them at last.

The main external needs that appeared when planning and designing the project were the following:

1. **Create a stable dialog system that allows to interact with the characters and transmit textual or visual information.**

   The research on how to make the dialog system was a bit complex, because although there are different tools in the market, the initial purpose was to make one's own or at least sufficiently extend some of them to create the dialogues from an external document (such as JSON, txt or csv) and then load the text in the game. However, for the final purpose of the game, this was not too relevant and it was decided to look for integrated alternatives. The options were to use Ink [21], Yarn [22] or Fungus [23]. All three are great tools to create dialogue systems, but after testing them all, it was decided to use Fungus; since, in addition to dialogues, it offered a complete impressive event management system to create the game flow that was needed within each area.

2. **Find the right tool for the implementation of the augmented reality system based on the available resources.**

   After observing different alternatives, the SDK offered by Vuforia [24] was used, since it offers a series of varied tools, with good performance and support for a multitude of platforms. Details about the different Vuforia tools are detailed in "Technical limitations and Work environment" section.

1.6. Planning

1.6.1. Task list

Below is a list of the tasks that were planned and have been carried out chronologically throughout the development of the project, bearing in mind that a work rate of between 35 and 40 hours per week was approximated. Each of the rows in the table corresponds to a task with its description and the estimated time to carry it out.

Task	Description	Hours
1	Planning of the game design and main concept. This includes: sketching the initial class diagram, researching about how to implement the areas system and delimit the physical location where the project is based.	**20**

2	Initial implementation of the geolocation classes and the base architecture of the app, following a Singleton-based pattern design.	**15**
3	Geolocation input research. This includes: searching for reference apps that use this type of implementation and testing several methods of getting the coordinates data.	**15**
4	Scene persistence methods research and implementation.	**20**
5	Creation and depuration of the monuments' 3D models. Integration and animation on each area scene.	**25**
6	Research of pattern recognition methods and implementation of symbols minigame.	**20**
7	Integration of Vuforia SDK.	**10**
8	Modelling and animation of K.A.I's main features. Research and integration of the hologram shader.	**20**
9	Implementation of memory minigame.	**15**
10	Research of different ways to implement dialog systems and testing of some libraries.	**20**
11	Integration of Fungus and creation of all the flowcharts and graphic event-management diagrams.	**30**
12	Creation of the areas status and integration with flowchart variables and the new data structures.	**20**
13	Design, implementation and animation of the full map system.	**20**
14	Implementation of the functionalities to saving and loading the game data into a file.	**10**
15	Progress percentage implementation, cel shading and user interface improvements.	**10**
16	Addition of the Firebase backend and the Angular web.	**20**

17	Addition of the Google Play Games Services.	**20**

1.7. Technical limitations and work environment

The platform for which the project will be developed will necessarily be Android (from 4.3 Jelly Bean) or iOS operating systems in smartphones.

The complete video game creation has been developed on a PC with the following technical specifications:

- Intel Core i7-6700HQ 2.60GHz
- RAM 8 GB
- SO Windows 10 x64

And the final application has been tested in a featured smartphone:

- Xiaomi Mi5
- Internal storage 32 GB
- Snapdragon 820 (1,8 Ghz) // Adreno 530
- RAM 3 GB
- MIUI 7 on Android 6.0
- Battery 3000 mAh

The size of the final application is 68 MB. This value will not increase even if new areas are activated from the web application, since the fixed areas are created in the project's own build interchangeably. On the other hand, the application has been developed entirely in Spanish, so initially it will be destined to tourism carried out by Spanish speakers. In the near future it is expected to implement a wider range of languages.

1.7.1. Augmented reality features and justification

Because of its final scope, the project was designed to cover as amount of target audience as possible, always taking into account the limitations of the technologies used and its performance on the supported devices. Is for that reason that initially was needed a research to delimit which functionalities could be covered and developed and which devices would support that. Currently, there are a lot of middle-high range devices that support Augmented Reality features, both in Android and iOS operative systems, but depending on which AR tool or platform is used, they will allow some functionalities or other. Down below these aspects will be covered deeply. Because of these original limitations, some design aspects were iterative and had to be adjusted when adding AR, as they was strongly linked to the final results and functionality.

In order to avoid to make this point too extensive, first it will be explained some of the most common functionalities that exist in the current market of the AR and that delimit the possibilities for the generation of content with this technology (Figure 3) and next there will be observed the different tools that implement them and that were investigated in order to decide which one fitted well the objectives and the scope of the project.

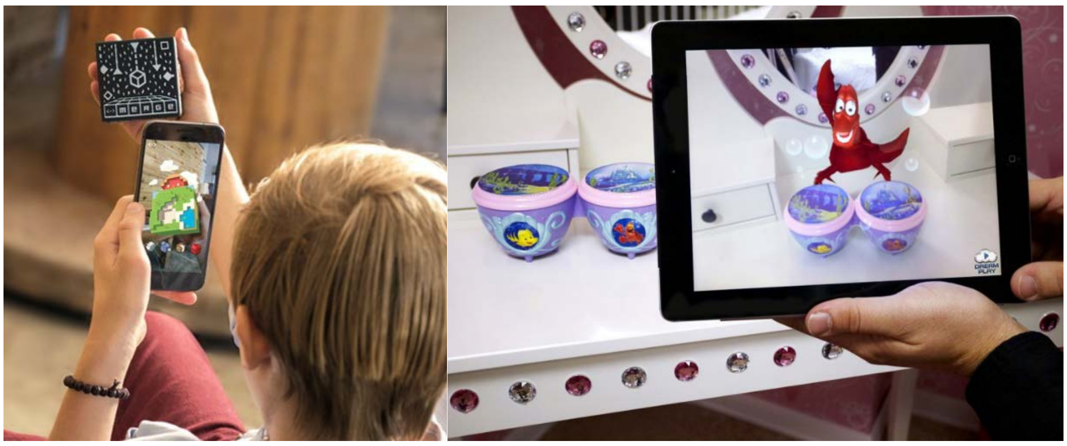


Figure 3: Augmented reality can be used by tracking both images and even physical objects

Currently there are different ways to show content in augmented reality. Since Vuforia integrates the vast majority of these, it has been decided to directly use the explanation about its features it provides, as it is clear and concise [25]. Some common uses are the following:

With targets

- **Image Targets**

  Images that the AR SDK is able to detect and track comparing its significant features against a known target resource database. Once the Image Target is detected, the SDK will track the image as long as it is at least partially in the camera’s field of view.

- **Multi Targets**

  Multi Targets are for objects with flat surfaces and multiple sides, or that contain multiple images. Just as Image Targets allow a developer to choose an image ahead of time that the app recognizes, User-Defined Targets allow an end user to pick an image at runtime. In outdoor scenarios the results are sometimes unpredictable.

- **Object Targets**

  Object Targets are a digital representation of the features and geometry of a physical object. They are distinct from image based target types, such as Image Target, Multi Targets and Cylinder Targets that require the use of a planar source image. An Object Target is created by scanning a physical object using some scanner tool.

- **Model Targets**

  Model Targets allow you to recognize objects by shape using pre-existing 3D models. Firstly, the physical object has to be modelled by hand or by using photogrammetry technology. After that, the model has to be treated in an external editor and imported to the game engine.

Without targets

- **Plane detection**

Plane detection (Ground Plane in Vuforia) enables digital content to be placed on horizontal surfaces in your environment, such as floors and table tops, without needing physical image targets. It supports the detection and tracking of horizontal surfaces, and also enables to place content in mid-air using Anchor Points.

At this point and knowing some of the possibilities offered by augmented reality, it was necessary to decide which SDK to use and define the scope of the project. The features and performance of the exposed functionalities vary slightly depending on which SDK is used, but there were other circumstances that were more relevant and had to be taken into account. On the one hand, as mentioned before, the preference was always to reach an as broad as possible user base, which is determined by using a tool with support for as many devices as possible. On the other hand, we also had to take into account some limiting factors, such as that the application should be able to run without problems on the outside and in the largest time slot as possible without having tracking issues.

In the current market there are different augmented reality SDKs (Figure 4). The most famous are: Vuforia (Qualcomm and after PTC), ARKit (Apple) [26], ARCore (Google) [27], Wikitude (Wikitude GmbH) [28] and Kudan computer vision (Kudan) [29].

Figure 4: Most used augmented reality SDKs in 2018.

At the time of choosing which SDK to use, the limitations came marked by the support that each one gave to the devices, observing if the smartphone in which they were going to carry out most of the testing (Xiaomi Mi5 32 GB) supported these functionalities. Finally, the three best performing SDKs were ARKit, ARCore and Vuforia, after having individually tested each of them and observed how they act in different environments. ARKit was discarded immediately, because it is facing devices with iOS and, although Unity3D is integrating it as a hybrid tool with ARCore, it is still in an experimental phase and could provide unexpected results. On the other hand, the still very limited list of devices supported by ARCore [30] was verified and it could be checked that the used smartphone was not among them. Therefore, the possibilities were reduced to Vuforia, which is possibly the AR software with the most variety of devices supported by its different tools.

With Vuforia, the different options offered were observed. Some require a complete scan of a physical object and others directly need a 3D model with exact dimensions for execution. Unfortunately, there was a lack of some necessary resources to apply it, such as a photogrammetry-based scanner or 3D models 1:1 scale of each one of the street elements where AR would be used. Therefore, by the tools previously seen, it was decided that what could be more interesting for the level of integrity in the development was to use either simple targets with Image Targets, or to do the same but without requiring a physical image to show the model in AR (preferable), with Plane Detection. The differences between them can be appreciated down below in the Figure 5. The other tools, although also very interesting and not too difficult to implement, were discarded because of their unpredictable performance outdoors. The answer between these two would again provide the support given for the devices, so it was investigated. Ground Planes in Vuforia, sadly, is only supported in the Xiaomi brand for the Redmi 3S, Redmi Note 3 and Redmi Note 4 [31], so it is not supported by the testing smartphone (was checked by installing the Unity testing package they offer). Therefore, using Image Targets and doing it in an original way was the definitive solution, planning the use of the planes detection in a future expansion.

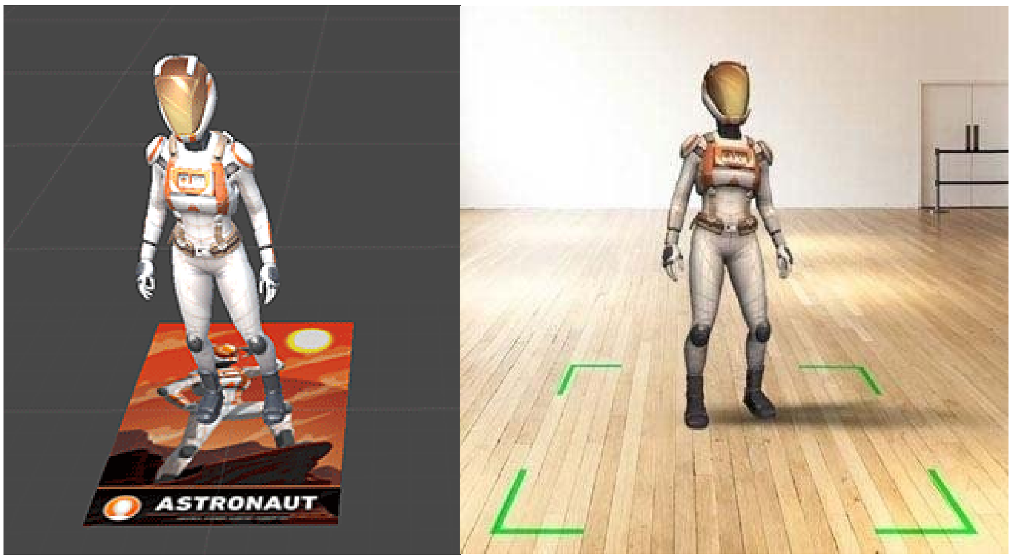


Figure 5: Image Target detection vs. Ground Plane detection in Vuforia

The picture used to act as an image target was an icon that represented both the geolocated component and the access to information through a fun disk (Figure 6). This design was kept as the rating of singular points to give the best possible result was high, thanks to its irregularities in the design and key points of the image.

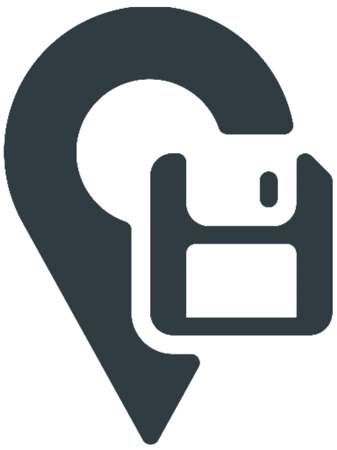


Figure 6: K.A.I.'s game image target.

2. Game Design Document

2.1. Overview

What would happen if, overnight, the whole history of our city was broken down and fragmented into small units of time and space? Even more, what if, for a mistake or on purpose, all these pieces of history were lost and there was only one person able to put back in order and recover that historical heritage legacy? From pre-Roman times, through the Kingdom of Valencia and reaching the Transition of the twentieth century, Valencia has been a very changing city in social, artistic, economic and religious aspects. All that knowledge stored in documents and captured in its streets has quickly degraded and it is in our hands to recover it.

The previous paragraph summarizes the context in which the action of this project is located. The idea that has been developed consists on an interactive game environment that combines the physical reality with an Augmented Reality interface on the screen, to create an implicit motivation in the player to discover new places and overcome the challenges he/she will find in them. Therefore, it is, in terms of classification in the different game genres, a game for exploration and continuous action, since the user must be attentive to the environment and he/she must to be able to find specific goals and surpass them in order to visit and recover all the information distributed in different areas.

To create this heroic environment in which the player is the protagonist, a narrative context was created that encourages action. Any day, a card falls into the hands of the player, through which it is possible to communicate with a robot from the year 4025 called *Knowledgekeeper Artificial Intelligence* (better known by its initials, K.A.I.). The robot will explain that it has been sent by scientists of the future to carry out an important mission: Recompose the pieces of the history of Valencia that have been either destroyed or forgotten by the people of their time. However, K.A.I. cannot do it alone. The player will be the one who must guide and take it to the sites to scan and retrieve the information. Not everything will be so simple, since all the information of the present is digitally encrypted for security reasons. The player will have to demonstrate his/her worth by facing the challenges of each monument, which are brief minigames, to overcome this protection system and save the cultural heritage in the future.

2.2. Target platforms

The project is focused on designing an application for smartphone, mainly in the Android operating system as well as iOS, since smartphones are an ideal platform to receive geolocated data and be able to estimate in real time the position of the device. It is intended from the start that the product reaches as many people as possible.

Tools such as the different aspects of the Java SDK combined with the Unity configuration offer great support for Android. In addition, the publication in the Play Store marketplace [32] is cheap and fast. For iOS, the process is similar, but a little more complex, since after having made the final build in Unity, it is necessary to configure it again in Xcode [33] with the appropriate version to add the registration and identity information before publishing. The App Store [34] requires, in addition, a review of the application and an important payment, so the publication process is considerably longer and more expensive.

2.3. Target audience

The application is not defined for a specific target audience, although a minimum experience in the use of the smartphone is required, as well as the capacity for reading and practical understanding. The age range, therefore, could be considered from 8 years old. The profile of the players, however, must be open and with a tolerance to the challenges; since the levels will increase in difficulty depending on the progress of the player. It should be remembered that the project will be used by both regular players who are not worried about the city's tourism and travellers who are not attracted by video games.

2.4. Aims

The objective of the game is to find and complete all the given areas of Valencia. The player has to complete all the challenges of an area to complete it. The data and information about the places is saved locally, so they players are able to consult it whenever they want, and all the levels are also replayable as many times as desired once the corresponding area has been completed. To retrieve the information or play the minigames, the player must physically return to the place where he first encountered them.

2.5. Main game flow

When the player initializes the game, the avatar menu appears with all the statistics achieved so far. At this time, only when the application is running, the GPS will start working and from now on, it will count the travelled distance (with a margin of error) and detect if there are areas of nearby elements with which to interact. It will also notify when a new area is entered, by means of a small warning. In short, the smartphone acts as a scanning device, as it allows the player to move and unlock the elements by playing minigames on the mobile phone. There are two types of games: a mini-game of repetition of memory sequences and another of skill and reflexes by drawing gestures. Further details of all this will be noted in the document.

The flow of actions carried out by the user now is as follows:

1. When the user moves within an area, there is a notification in the app that allows the player trigger the “connection” and load the Augmented Reality screen in order to play the corresponding level.

2. Then, when the player presses the "Start communication" button, the phone's camera is activated and the interaction with augmented reality begins. A message tells the player to place his card in front of the camera of the device.

3. When the image target associated with the card is detected, the talking robot K.A.I. appears. After a brief introduction, there is the possibility of playing the minigame to "decrypt" the information of the area (Figure 7).

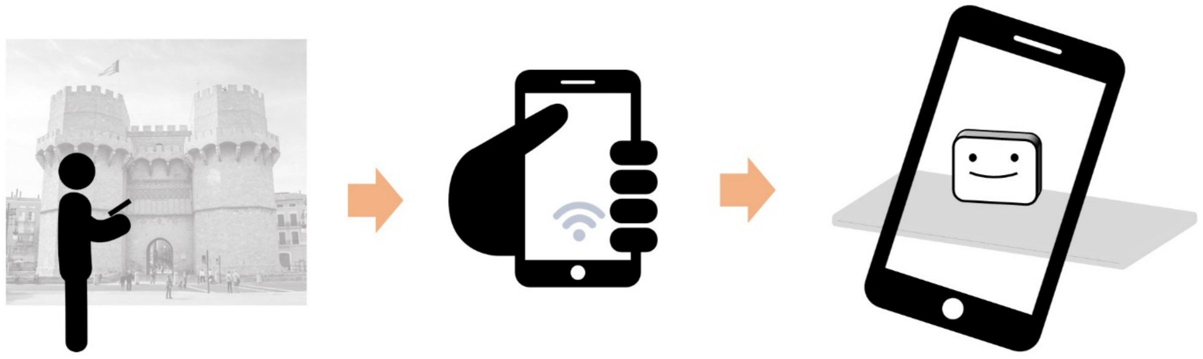


Figure 7: The player reaches an area and starts the augmented reality system.

4. The user plays the minigame associated with that area, which can be the memory minigame or the gestures minigame. He/she has as many opportunities as desired to surpass all the proposed levels and consequently complete the minigame (Figure 8).

5. Once the player solves it, he/she completes that area and unlocks the information about that element.

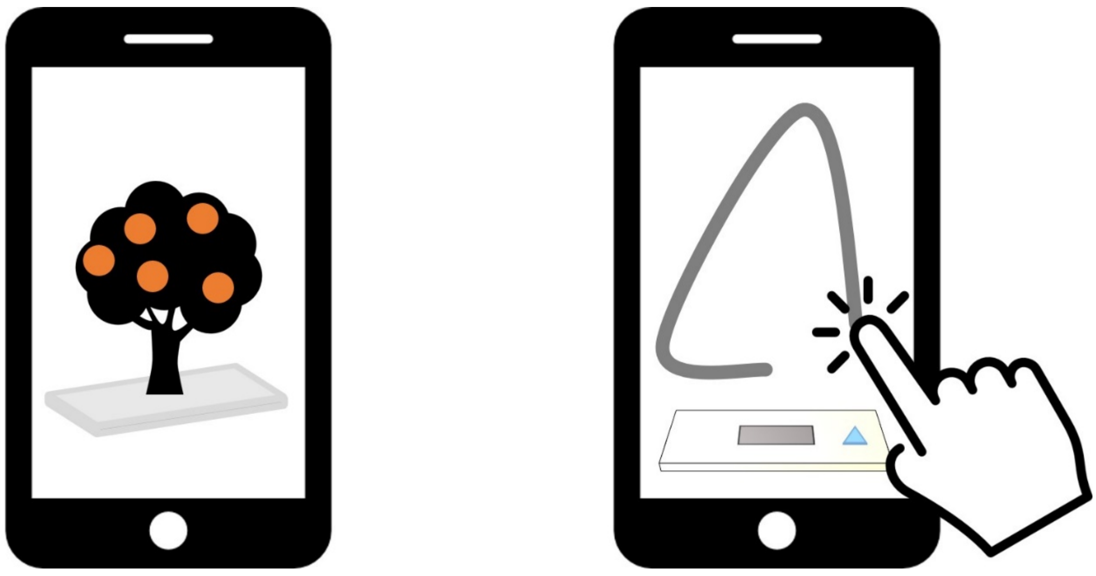


Figure 8: One of the two different minigames appears before unlocking the touristic information.

6. The 3D model of the hologram corresponding to that area's monument becomes visible. After an animation, several intractable points appear around the building (Figure 9).

7. When the user touches any of these points, an old photo (between 1930 and 1960) of the monument appears along with a brief historical description related to that particular point in the context of the monument and the time of the photo (Figure 9).

8. At this time, the user has the possibility of returning to a main menu through a contextual button. There he/she can choose whether to play the mini-game again or to see the information again.

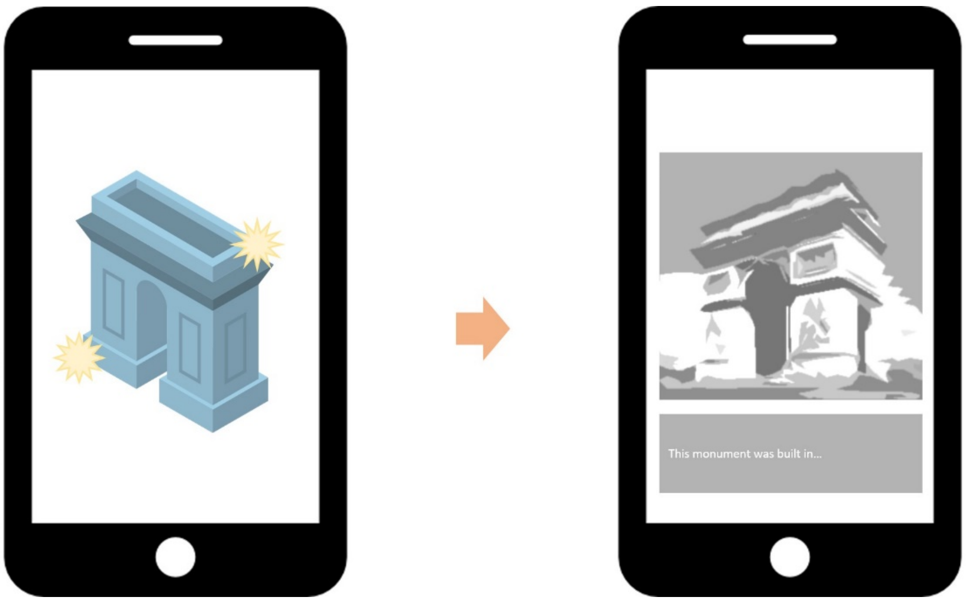


Figure 9: The monument building with a hologram look is showed and provides the player the information.

It is important to note that, at any time, the user can return to "radar mode" and leave the augmented reality functionality of the area, even if he/she is inside it. However, if he/she remains within the interaction with AR and physically leaves the area, the application will return to the radar automatically and it will show a message pointing out that it is not in any zone to interact with, until the user reaches another physical range.

2.6. Kind of challenges

2.6.1. Exploration challenge

This is the main global challenge that surrounds every aspect of the game, as from the beginning the foundation of the project is promoting exploration and getting information through overcoming different levels with progressive increasing difficulty, so it is necessary to get access to new areas with new challenging levels. As we will see later, thanks to the web app, it is possible to attach hidden levels or hidden areas in order to provoke more curiosity and immersion in players. Promoting curiosity is essential on this challenge, as the player has to be motivated to keep on playing and exploring new areas around the city centre. The percentage score and the areas status, always visible on the map screen, allow the player to know which areas have been visited, which areas have been completed and what is his/her score so far.

2.6.2. Atomic challenges

In order to reach the information hidden in each area, player has to overcome some obstacles, which will correspond to two different kind of little games that will prove their memory and reflexes. An area will only be considered as “completed” when the player passes all the levels of that game. These games constitute a sort of encryption on each area and are the immediate challenges of the game, as player only can play it when he/her is within the area. Minigames can be played again once completed, but only if the user is in the corresponding area. In the analysis section, a more detailed view about each minigame is provided.

2.7. Resources and entities

2.7.1. Map

The main map is a fundamental entity that helps the player to know where is him/her and which zones are available, visited or completed. It will fulfil an informative function, as the user is able to check at any time what is the status of the areas and what is the percentage of the game that has been completed. In the following analysis section, a more detailed description of the map is provided.

2.7.2. Score percentage

The percentage of completed game is the value that the player must take into account if he/she wants to reach 100% of the challenges offered by the application. For this, it will be necessary to visit all the places and overcome all the minigames in them to be able to see the tourist information hidden in each one.

2.7.3. Areas

The areas are the minimum game units of the project. They are located in the physical space of the world and, to access them, it is necessary to be within their geolocated range. Once inside, the player is able to interact with the elements of the environment through augmented reality. Each area also has a stored status for each player that indicates his/her situation in the game: Unknown, available, visited or completed. As mentioned previously, the player must get all the areas to have a completed status and be able to complete the game.

2.7.4. Touristic information

The tourist information is transmitted through text and relevant images. This type of information is hidden in each of the points of the 3D models of the monuments in holographic form, and will only be accessible once the player has completed the minigame of the area.

To develop this part of the project, it has been necessary to investigate about the most relevant historical information of each monument or of each zone. To do this, it has been searched different blogs of culture and historical heritage, such as the main website of Valencia tourist guide [35], and the acquired information has been synthesized. Then, the historical heritage information about the monument was expressed in terms closer to the usual language and summarized in a comfortable way to be inserted into the game.

2.8. Types of actions or mechanics

Since the platform to which the project is destined is an smartphone, all the actions done by the player will be carried out through interacting on a touch screen, either by a simple "tap", or by various gestures recognized by the program that allow to trigger the status of play at different points. Beyond a simple interface where the user interacts, there have also been presented augmented reality mechanics where the perspective and movement of the camera around the object play an important role for the progress along the game (always combined with touching on the right place).

2.9. Construction and articulation of the levels

The levels of the game can be understood from two widely related but also differentiated spheres. They are going to be distinguished, as has been done throughout this document, depending on the functionalities that are integrated in the game: Geolocation services and the use of augmented reality. From a general plane, the game system is built by the list of areas that the player must visit, with their corresponding parameters, and all of them framed in a delimited physical space that corresponds to the historic centre of the city of Valencia. Within each area, the levels have a structure and a very similar flow of actions, although each of them is developed in a particular way and has its peculiarities and its singularities. Next, you can see how the areas are distributed in the physical space of the centre of the city of Valencia in the Figure 10.

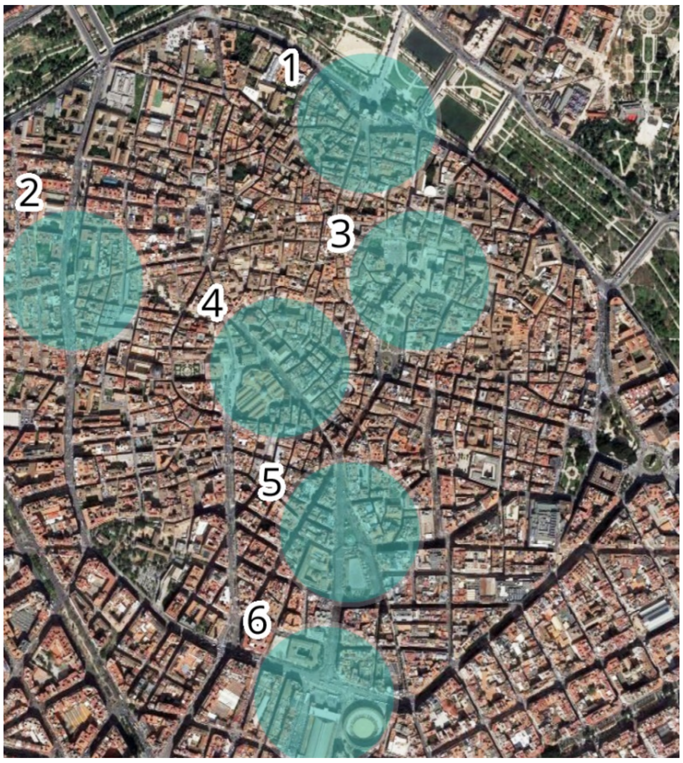


Figure 10: Geographical situation of all areas in Valencia city centre.

List of areas (from north to south):

1. Torres de Serranos (Serranos Towers).
2. Torres de Quart (Quart Towers).
3. Micalet and Valencia's Cathedral.
4. Lonja de la Seda.
5. Ayuntamiento de Valencia (Valencia's city council).
6. Estación del Norte (North Station).

3. Analysis

Next, each of the blocks that make up the project will be covered from a technical point of view and at a high level. Fundamentally, it has been decided to divide this section into two main sections, which refer to the two major technical aspects that have been studied and developed for the creation of the application: the geolocation system and the interaction in augmented reality when reaching a certain area.

3.1. Geolocation system

The geolocation system is the basis that gives the application its foundation. Through the GPS satellite connection, received thanks to the native geolocation services of the smartphone, it is possible to calculate with a certain precision the position of a player on the Earth's surface and, thus, to check if he/she is in a determined point. Nowadays there are different ways to create a stable geolocated system. Depending on the desired concept to implement, it is necessary to consider a robust prior design and find the right tools to do so, either through well-known APIs that manage the information received on time, such as the variety offered by certain Google services, or through other custom systems.

One example for a custom system is to compare the data of the current position received with a series of predefined areas and check if the player is in one of these areas to trigger the corresponding event. This approach is the one that has been decided to implement. In an iterative way, the system has been improved as the project progressed, from the insertion of different status in each area until finally the use of a small back-end to connect to a dynamic database using Firebase to create the areas and update them automatically in the application. This systems follow the logic of a well-known technique called *geo-fencing* [36], which consists on define an imaginary closed area in a specific physical space in order to use it, for example, in applications related to geolocation.

Each of the points will be detailed in a chronological order of execution (not necessarily of implementation) to facilitate the reader's understanding of the system.

3.1.1. Geo-fencing: Shape's issue

First of all, it was necessary to define the way in which the zones would have to be distributed along the terrain and how was it going to be divided. Generally, there are two ways of doing it: delimit the areas by simple regular or not regular polygons or do it by circles. (Figure 11) This first choice was to determine where the design of the system would be addressed, since, for example, making the circular zones implied that there would be certain places where no zone existed or, on the contrary, that there would be areas that overlapped inevitably, which always implies less precision and control. The advantage of doing it this way and not by polygons was the simplicity of calculating the distances between positions by observing the relative distance of the player to the centre of the area. A method much less complex that maintains integrity and the intuitive factor.

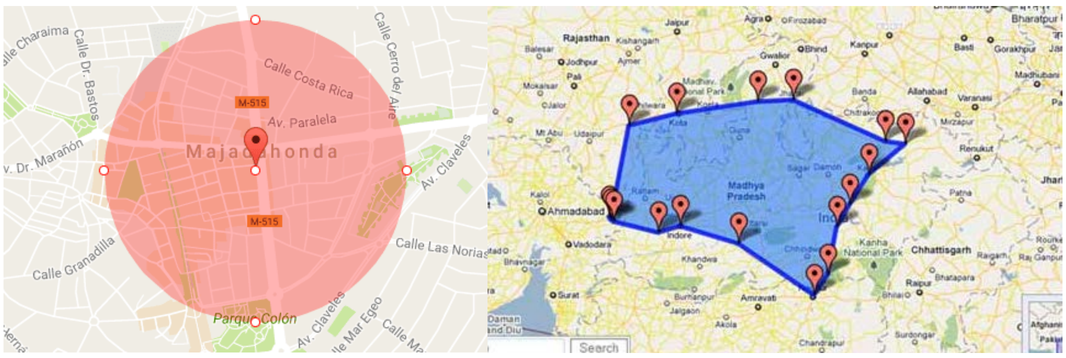


Figure 11: Radial vs. polygonal Geo-fencing.

3.1.2. Geolocation data storage

The next step when creating the system was to decide how the information of each area would be stored. At the conceptual level, it should be an expandable system that would allow areas to be created quickly and in a personalized manner, since the main idea was linked to the principle of scalability and flexibility. To do this, we defined the main parameters that should be composed by each area, which were: name, radius, latitude and longitude.

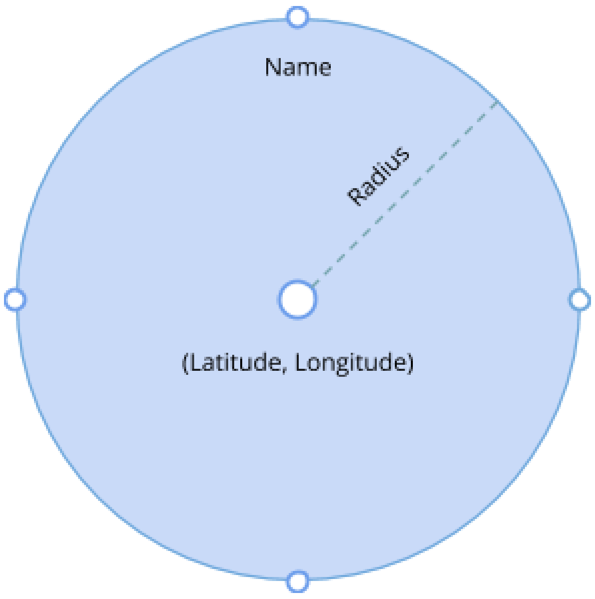


Figure 12: Definition of an area.

The question that had to be answered then was how to create them, and the response was outlined along a constant debugging process in which this storage went away from the code to make it more comfortable, high level and closer to the user. Without going into details of implementation, what began as tedious lines written by hand with the coordinates has ended in the online insertion of data visually with the help of Firebase and an extension of Google Maps.

3.1.3. Receiving and managing GPS data

Receiving GPS data in the device is one of the most delicate tasks and that most compromises the integrity of the project, since certain factors must be taken into account. As we have seen, GPS is a signal that is determined through a set of 24 to 32 satellites and by the mathematical method known as trilateration. The accuracy of the position data calculated in a moment of time will depend on external factors that we cannot control a priori (at least, within the scope of this project). Normally, there are three key factors that can interfere in the connection and generate a certain error: the state of the satellite, the state of the propagation medium and the state of terrestrial reception. In addition, another type of factors that usually influence are the signal delay in some layers of the atmosphere, the rebound of the signal in buildings and mountains, errors in the satellite's orbit, the number of satellites visible from the reception point in a precise moment and some possible local error in the GPS clock.

Another disadvantage that must be taken into account when working with mobile devices is the excessive battery consumption that currently causes the continuous use of GPS. Being aware of this, it was necessary to create a design that would allow the device to be turned off at certain moments of inactivity and that this would not have a direct consequence on the player, but would be integrated with the game system naturally. Besides, care was taken to update the responsible reception and with a minimum of five to ten seconds between updates of the current position. This has advantages such as battery saving and an accuracy margin when entering an area.

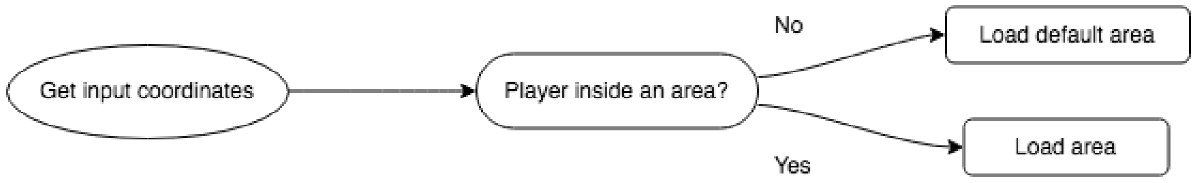


Figure 13: When current coordinates are updated, system checks if player is inside an area.

3.1.4. Areas status

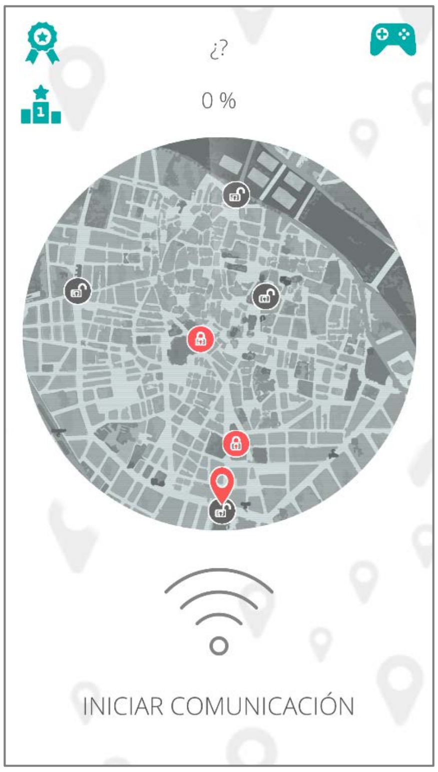
In order to have greater control over the actions the player has made in the areas and have the ability to hide them or make them available, a specific status has been assigned to each area relative to the player. This status can have four different values:

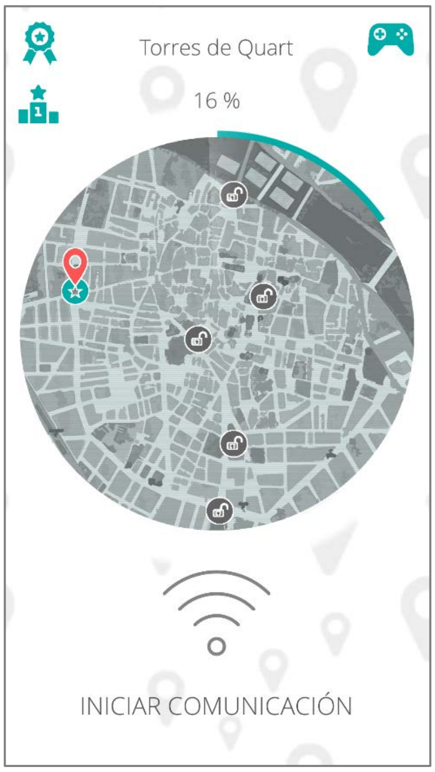
- **Locked** There is an area in a determined site but the access is not allowed for the player at that point in the game. By default, there will be areas will be blocked which will be unlocked after the player has overcome some other areas.

- **Available** The area has been unlocked and now the player can access it and start the quest by communicating with K.A.I.

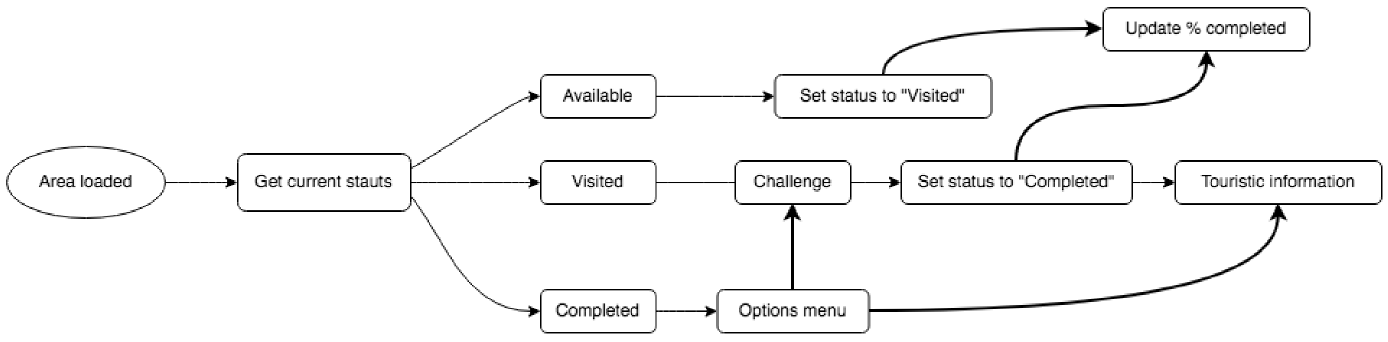
- **Visited** The player has entered the area but has left without overcoming the challenge, so it remains pending. A visited area is considered "partially completed".

- **Completed** The area has been visited and the challenge has been overcome, so that the area has been completed. If the player returns to the physical site, the area can be accessed again and revisited as many times as desired, either to repeat the corresponding minigame or to see again the tourist information of the place.





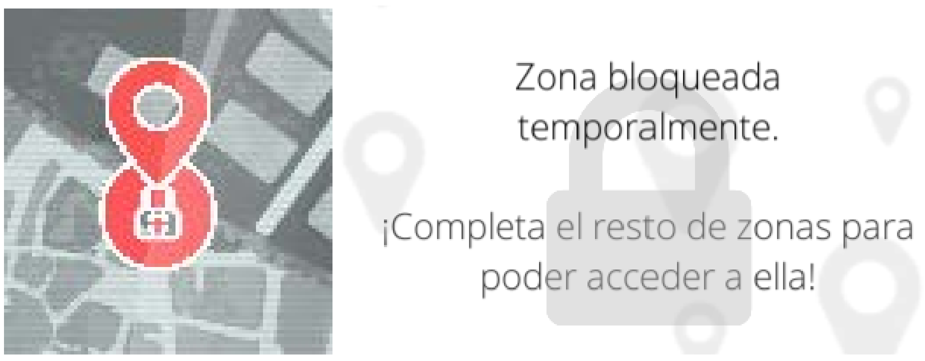
Changes in the status of a particular area occur during the course of its execution. As soon as an available area is accessed, it automatically changes its status to “Visited” and starts posing the challenge to the player. If this challenge is overcome, the corresponding tourist information will be displayed and the status will be “Completed”. Since areas in the “Locked” status cannot be accessed, they will never be loaded. Therefore, it is not necessary checking that state when the player enters the area, because it will never happen. In Figure 14, it is possible to observe how status changes in an area as the player executes actions.



https://www.draw.io/?page=1 - G1crimHDOFrDyOvzKaKUK0YH3ngz9oTne9

Figure 14: Diagram that shows how an area check and update its status.

When a new game starts, there is a certain percentage of areas that remain locked and to which the player does not have access. That percentage can be configured from the implementation. If the player is within the range of a blocked area, a message will appear on the screen indicating it (. When the player completes an area, automatically one of the blocked areas (if there are any) is randomly selected and it is unlocked. This is, its status changes from "Locked" to "Available" and becomes accessible to the player.



Zona bloqueada temporalmente.

¡Completa el resto de zonas para poder acceder a ella!

Figure 15: Locked area notification.

The status of an area is automatically saved in the local data file each time it is changed. Thus, the information is always up to date and can be accessed from any part of the program. When any status changes, the global variable of the completed game percentage is also updated. This information is saved online in the player's Google Play Games account, so all players are able to check the leader boards and see all others' scores.

3.1.5. Map displaying

The map is a valuable resource that is only displayed when the player is not interacting within an area. This, as we will later see, only occurs in the default scene. The map is largely related to the previous point, since it is where the status of the areas are visually displayed and this allows the player to know his/her current both game and geographical situation. The map shows all the areas symbolically through an icon with a drawing and a colour that express its status ("Locked, "Available", "Visited" or "Completed"). If the player is within the range of an area, the placeholder will appear and indicate where it is. Also, if the area has been viewed, its name will appear at the top of the screen.

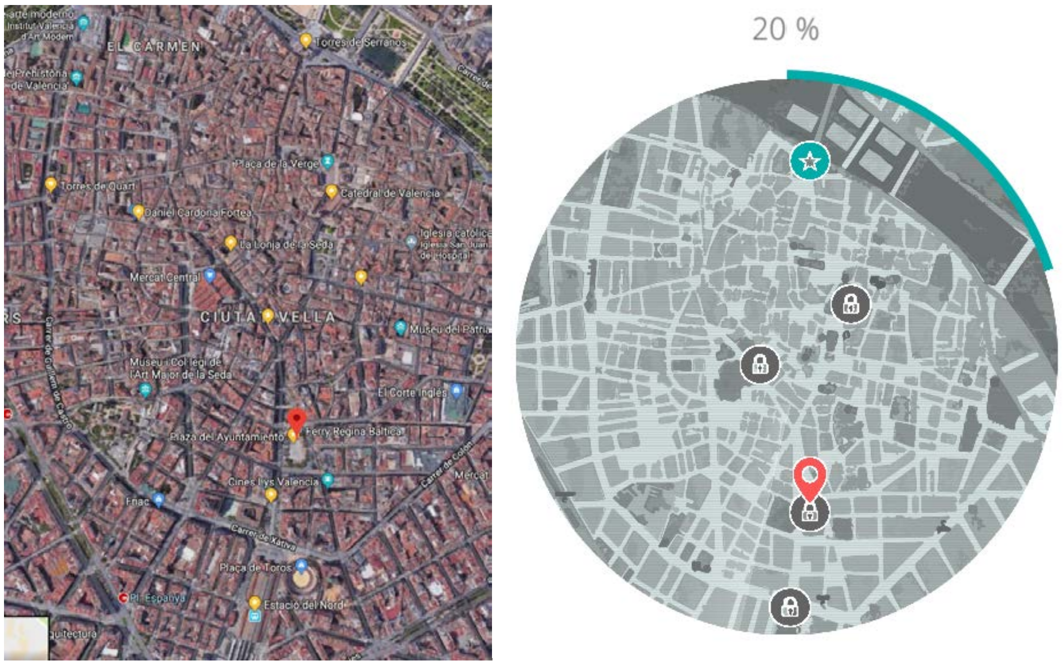


Figure 16: Points where the areas were defined. Left: Google Maps. Right: Map HUD.

3.1.6. Storing the areas' information on the cloud

To scale the application or modify it without upload a new version each time to the publishing market, it was necessary to create a database in the cloud capable of sending data in a format such as JSON so that it can be read from the application in Unity each time it is started and then update the local data from it. To create the web application site and store the data with the parameters of the areas in the cloud, Firebase was used as the main data storage database. The web client was created using Angular together with some libraries, as will be detailed later in the implementation section. The following Figure 17 shows the process by which the information of the areas is generated on the web and each device receives the data of the areas and synchronizes them at the beginning of the execution. All these processes will be detailed later.

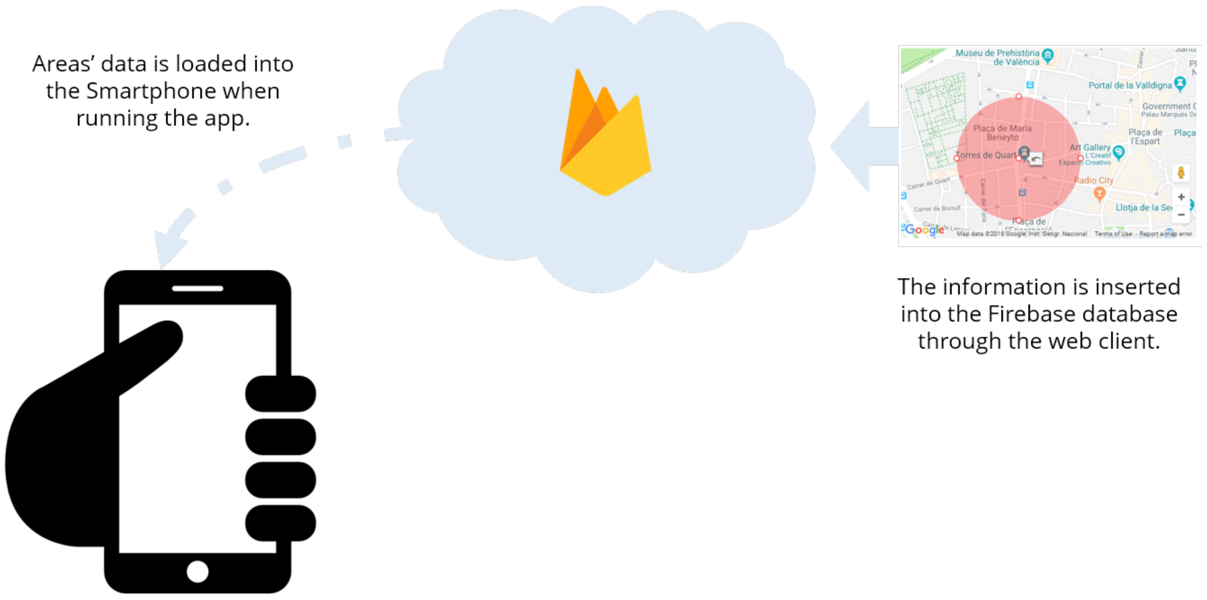

Areas' data is loaded into the Smartphone when running the app.
The information is inserted into the Firebase database through the web client.

Figure 17: Flow of the areas data, from the web to the app.

3.2. Augmented Reality

Augmented reality is the second main component of the application and the one that provides the player a more direct and close connection with the surrounding environment. When an area is reached, a contextual message appears on the screen and invites the player to "initiate communication with K.A.I"; or, what in terms of the game means, interact with that area and initiate augmented reality functionality. When pressed, the screen immediately the image received on the device's camera is showed and a message indicating the instructions to follow to make the friendly robot appearing is displayed.

As previously mentioned, there are different platforms and tools that offer implementation services for augmented reality functions. On a practical level, the complexity of implementation of each other does not change much, but since in this project one of the objectives was to have a solid base of users and to open it to the maximum number of possible devices, it was decided to finally use the *image targets* method integrated into Vuforia and make the models and dialogs appear only when the player has tracked the card with the pre-established image. For more information about this, please see Technical limitations and work environment section.

There are three main stages or components in the area cycle: K.A.I., which acts as speech moderator and links the actions; the minigames, which are related to the direct interaction by the player in an environment with defined rules and, at last, the model of the monument with the corresponding texts, that is, the informative part of the area.

Next, each of the three stages how the player interacts with them will be described. This has been addressed to enhance the use of augmented reality from different perspectives: Conversational / narrative with a character set in its own context, interactive games with experimental mechanics and spatial exploration of a three-dimensional model.

3.2.1. K.A.I.

The small robot K.A.I. is an adventure mate, experienced guide that will help the player to reach his goal and complete all the levels. It appears at the beginning of each area to *scan* the environment in which the player is located and gives the guidelines to introduce the corresponding minigame, in case it is the first time the area is visited (Figure 18). At the end, he is the one who generates the three-dimensional model of the emblematic monument of the place so that the player can interact with him. It is important to note that in this game the main character of the adventure is the player, since he/her has the responsibility to complete all the levels in order to recover the lost information. However, the robot serves every time as support and help and gives player a weighty reason to fulfil his mission at the same time.

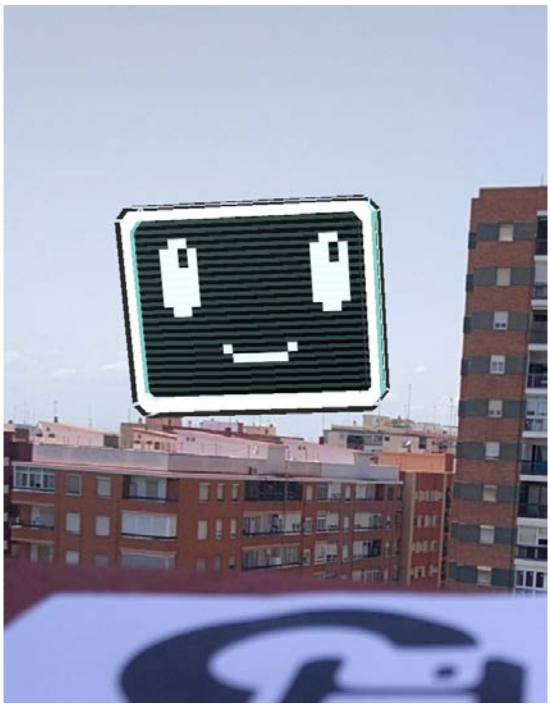


Figure 18: K.A.I. after scanning the environment.

3.2.2. Minigames

3.2.2.1. Memory game

This minigame has its foundations in the well-known electronic memory game called Simon, created by Ralph Baer and Howard J. Morrison in 1978 [37] (Figure 19). As in its original influence, a series of figures will be illuminated following a sequential order created randomly. The player will have to repeat the sequence in exactly the same order as it was generated, touching the corresponding figures on the screen.

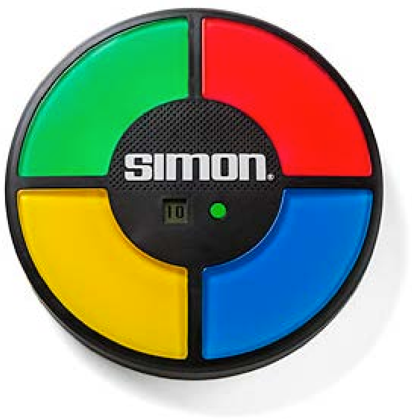


Figure 19: Version of Simon electronic game.

However, the difference is in the distribution of these figures in space, since this time it will be in a three-dimensional space that will be observed by the player in augmented reality. In this way, spatial perception also becomes a relevant factor, in addition to memory, to overcome each level offered by the game. To make the game more integrated in the Valencian context, it was decided that each of the objects that the player would have to touch to complete the sequence would be the oranges of a typical orange tree. The following Figure 20 shows the final result of the game in augmented reality.

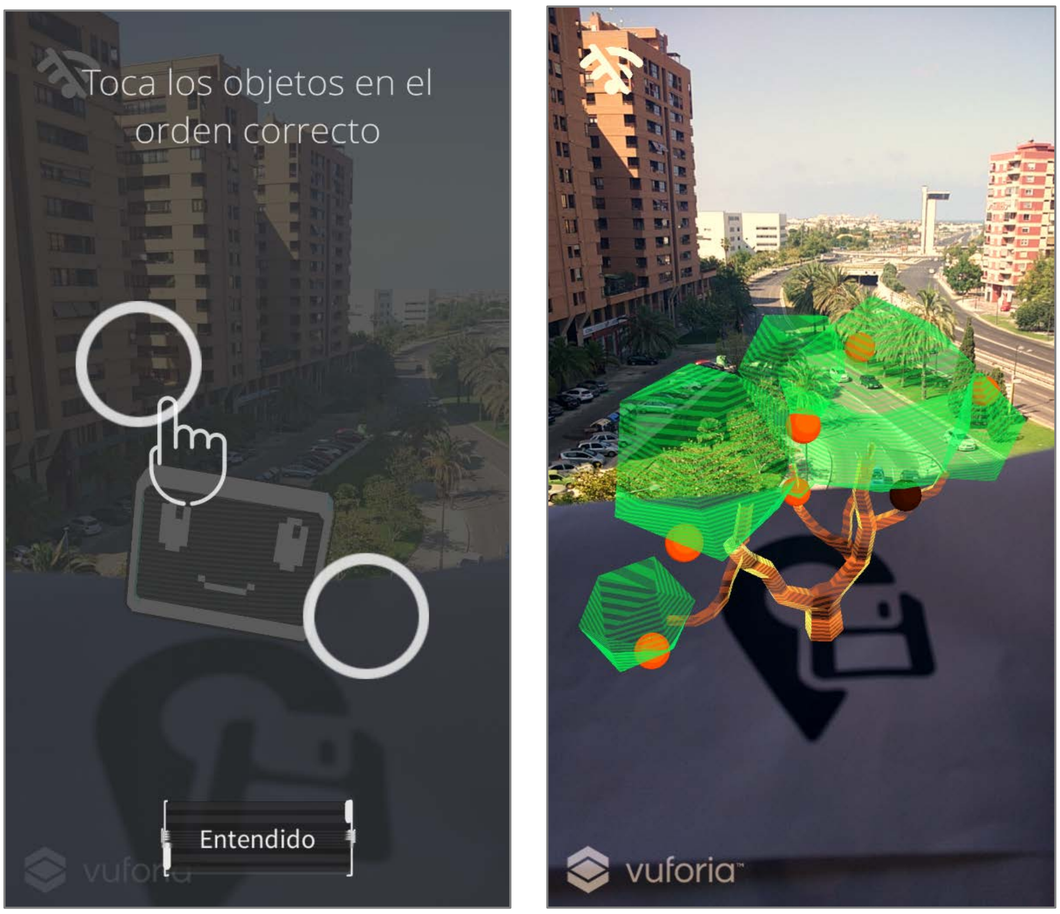


Figure 20: AR memory game

Once a sequence is overcome, the game will increase its difficulty by adding one or several figures to the sequence and generating another one in a random way. As usual, the player manages to reach the end of the game when all the levels are passed. The difficulty of the minigame and the number of levels depend on the area in which the player is and the progress of the game. Next, a diagram with the game execution cycle in the Figure 21 is shown.

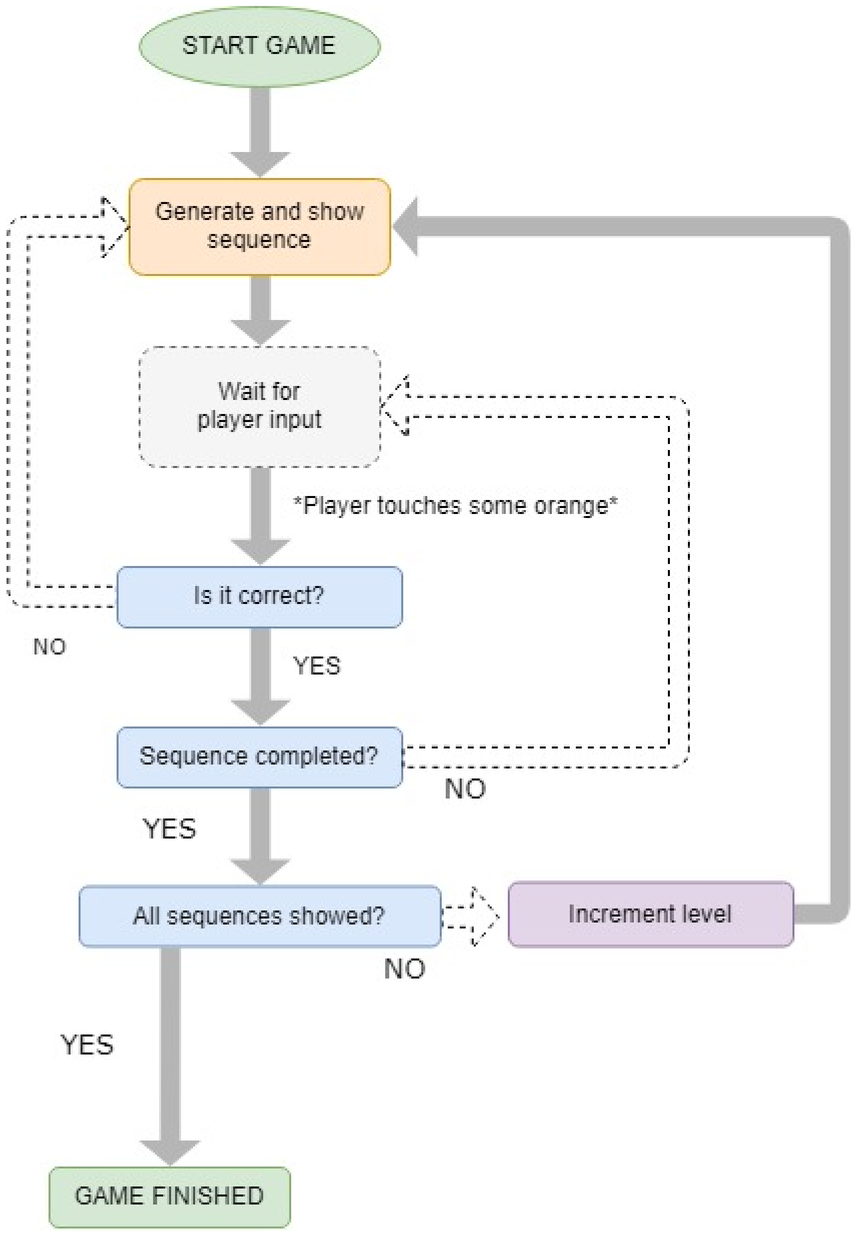


Figure 21: Complete memory minigame game cycle-

3.2.2.2. Pattern recognition game

In this minigame, it is essential to be attentive and put as much concentration as possible in the reflexes and speed. The symbols appear in a pseudo-random way on all sides of the screen, and the player must make the corresponding figure at the precise moment it is over the colored range. The range will change its size throughout the game, so it will not always be so easy to hit. The following Figure 22 shows the final result of the game in augmented reality.

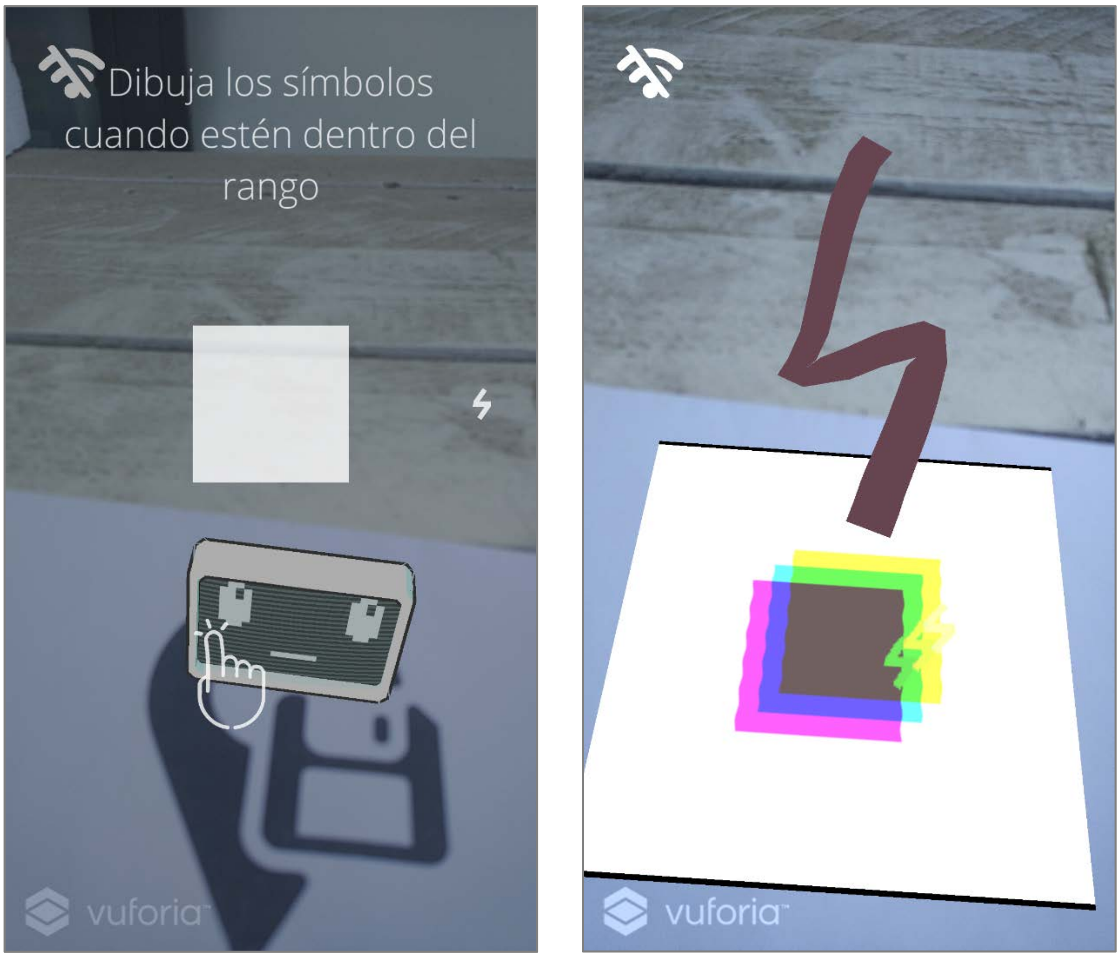
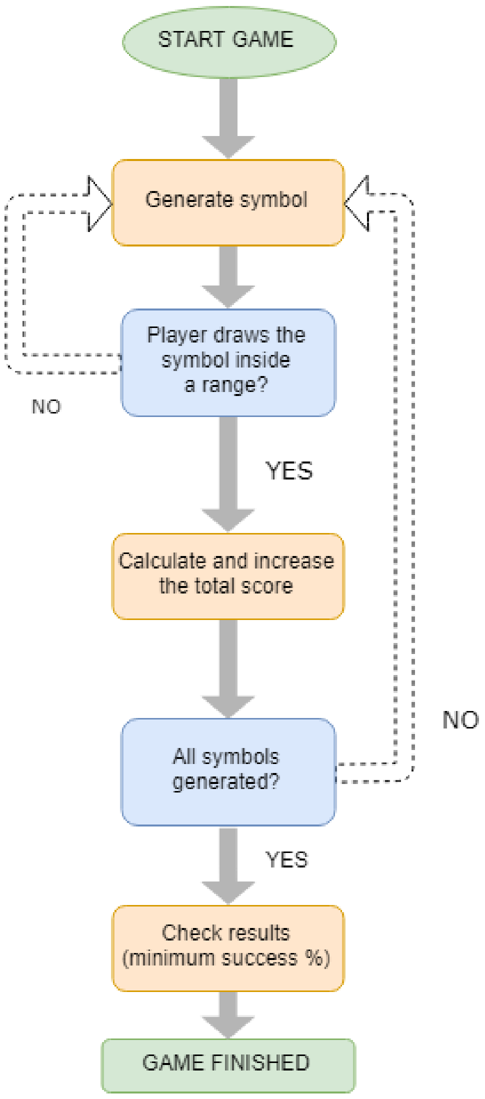

Dibuja los símbolos cuando estén dentro del rango

Figure 22: Pattern recognition game.

The scoring system of this minigame follows a simple logic: the player must achieve a certain hit ratio (percentage of success), which is different depending on the level of difficulty. The parameters of each level will be explained in more detail below. The player will get points every time he draws a symbol correctly. These points will depend on factors such as the speed of the symbol and the size of the range at the moment in which this symbol has been successful, based on a small arbitrary formula. In this way, the game is fairer when it comes to scoring. Next, a diagram with the game execution cycle in the Figure 23 is shown.



START GAME
Generate symbol
Player draws the symbol inside a range?
NO
YES
Calculate and increase the total score
NO
All symbols generated?
YES
Check results (minimum success %)
GAME FINISHED

Figure 23: Complete pattern recognition minigame game cycle.

3.2.3. Touristic information

Observing the three-dimensional models of monuments, buildings or objects of interest in each area in augmented reality is a great step forward for the user in the playable experience, as it allows him/her to explore areas and details of the model beyond the limitations of a two-dimensional display. Visualizing the body in space makes user more curious about the all the details the model has, such as a hidden stairway that had a certain function several centuries ago, the exact width and height of a famous medieval gate, the situation of each of the faces of a building, etc.

When a mini-game is completed for the first time or the option of showing the information is selected, the model of the building or object of interest is displayed and then the player can appreciate it. After a brief animation, several points appear around the model that show the possible interactions to see the information. When the user touches any of these points, K.A.I. briefly explains something related to the place you are visiting and usually also related to that part of the model or object (going back to the previous example, an information about what happened with that gate or that stairway in medieval times). In addition, this information will always be visually supported by a photo taken between 1930 and 1960, to convey more tangibly the change that this area has given over time and encourage the user to think about it.

4. Implementation details

4.1. Running the app: Game Manager

As you can see, a modular attitude has been taken dividing the project code. The `GameManager`, on this occasion, has a set of attributes and more general functions that need to be accessed from some parts of the program. Some attributes are the name of the default area, the URL from which to obtain the information of the areas via the web and the set of local areas. This script serves as a link between local data, cloud data and the connection between the main scripts. From here you can communicate with the `SaveManager` in order to save, load, reset the data or obtain the status of the current area.

The first thing you do when you start the app is to create the Singleton instances of the different managers (`GameManager`, `GeoLocManager`, `SaveManager`, `MapManager`, etc.). Immediately after this, the list of areas in the `GameManager` is created with the default area as the only element and then an asynchronous method is executed that downloads through a provided URL the data of all the areas that we have defined and introduced in the web. The program flow, as it should be, does not operate with the areas until this operation has been performed.

4.1.1. Synchronizing the areas data

Once the data has been obtained, it is synchronized with the locally stored data. At this point the saved data is loaded and two situations can occur:

a) It is the first time that the player executes the application:

There is no saved data, so the `SaveManager` must be responsible for creating a new `GameData` object to save the data corresponding to the areas that have just been loaded into the application. Once it has been done, the newly saved data is loaded and it is not necessary to do any other operation, since obviously they are already updated.

b) It is not the first time that the player executes the application:

Therefore, a file of saved data already exists and it is necessary to verify that the local data is synchronized with the data of the cloud[1]. Web data will always have preference over local data. This means that if, for example, a local area has been deleted on the web, as soon as the synchronization process is finished, this area will be locally removed without affecting the others and the percentage of the game completed by the user will be updated in relation to the current existing areas. The exact same thing will happen if a new area is added.

The algorithm used to perform this operation is as follows:

1. Add the areas that are in the cloud and are not locally to the local area array.

2. The areas that are locally and not in the cloud are searched and their name is saved as an identifier in order to be able to eliminate them later.

3. The areas to be removed are traversed and removed from the local area array. Now the system has been synchronized.

The following Figure 24 shows how the data from the local areas and in the cloud do not coincide. It may have been assumed that a new area (yellow) has been inserted into the cloud storage database. In addition, in the cloud it has been previously decided to eliminate another area (orange). As soon as the application starts, this will be checked and both data will be synchronized giving priority to the cloud list. The yellow area will be added to the local list and the orange area will be removed from it.

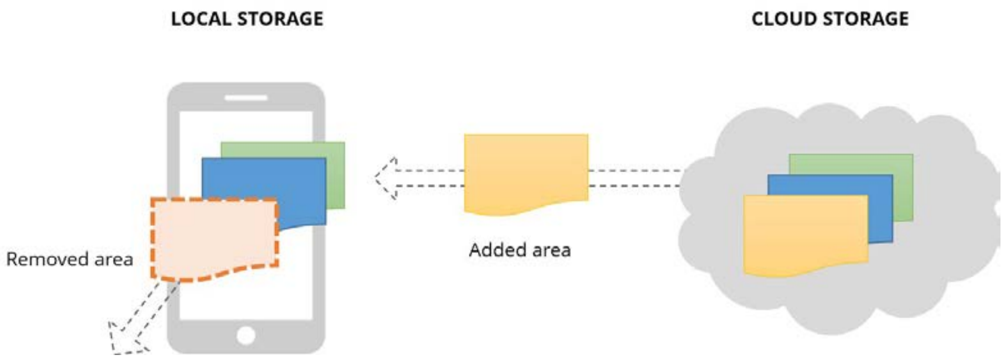


Figure 24: Example of data synchronization between the cloud and local areas lists.

[1] Various situations can occur due to which the data of the cloud have been modified and do not correspond to the locally stored data. Due to its nature as a database, a new area may have been added, and an existing area modified or eliminated.

After synchronization, it is time to call the event that will start geo-location services thanks to the `GeoLocManager` and get the player's current position.

4.2. Geolocation system

`GeoLocManager` is the class in charge of managing everything related to obtaining data through the sensors of the device and its internal management. It has information about the current coordinates and the area in which the player is located, as well as the time (approximate, remember that it is never exact) of updating the coordinates. Its function is to periodically observe where the player is in the world and, based on that, check if it is within an area defined in the `GameManager`. If it is affirmative, the player will be visually notified so that he/she can access to upload it whenever he/she wishes.

It is important to note that two support classes have been created to store the geolocation information: `Area` and `GeoLocCoodinates` (Figure 25).

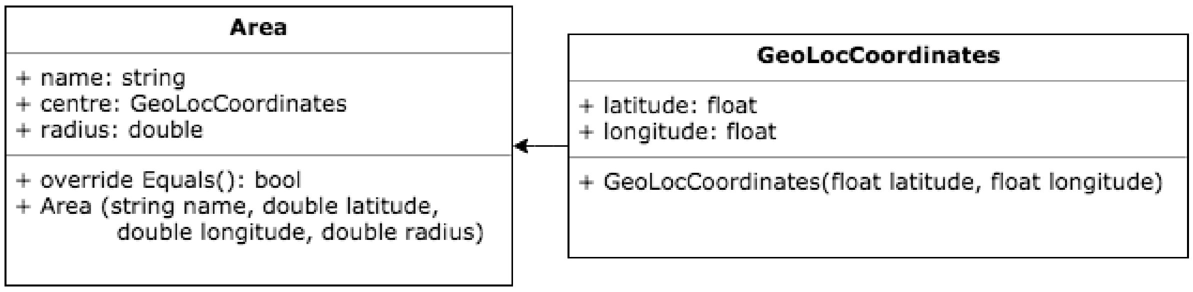

Area
+ name: string
+ centre: GeoLocCoordinates
+ radius: double
+ override Equals(): bool
+ Area (string name, double latitude, double longitude, double radius)
GeoLocCoordinates
+ latitude: float
+ longitude: float
+ GeoLocCoordinates(float latitude, float longitude)

Figure 25: Area and GeoLocCoordinates C# classes.

Once the `GameManager` has synchronized the areas, the `GeoLocManager` starts to work. The current area is set to the default area, as, in order to make the system more robust, it was decided that the user must always pass through this intermediary area before having the possibility of accessing another new area. The coordinates of the default area have been arbitrarily set to latitude = 0 and longitude = 0. The asynchronous method that will update the coordinates every elapsed time is then started.

4.2.1. Updating coordinates and updating the current area

The method `UpdateCoords()` is called every certain elapsed time and the coordinates are updated. After that, `UpdateArea()` is called to check if the player is within an area in the current position that has just been updated.

Once the area is updated, there are three possible situations:

a) **Player has not reached any area**: He/she was in the default scene and is still there, so there is nothing to load.

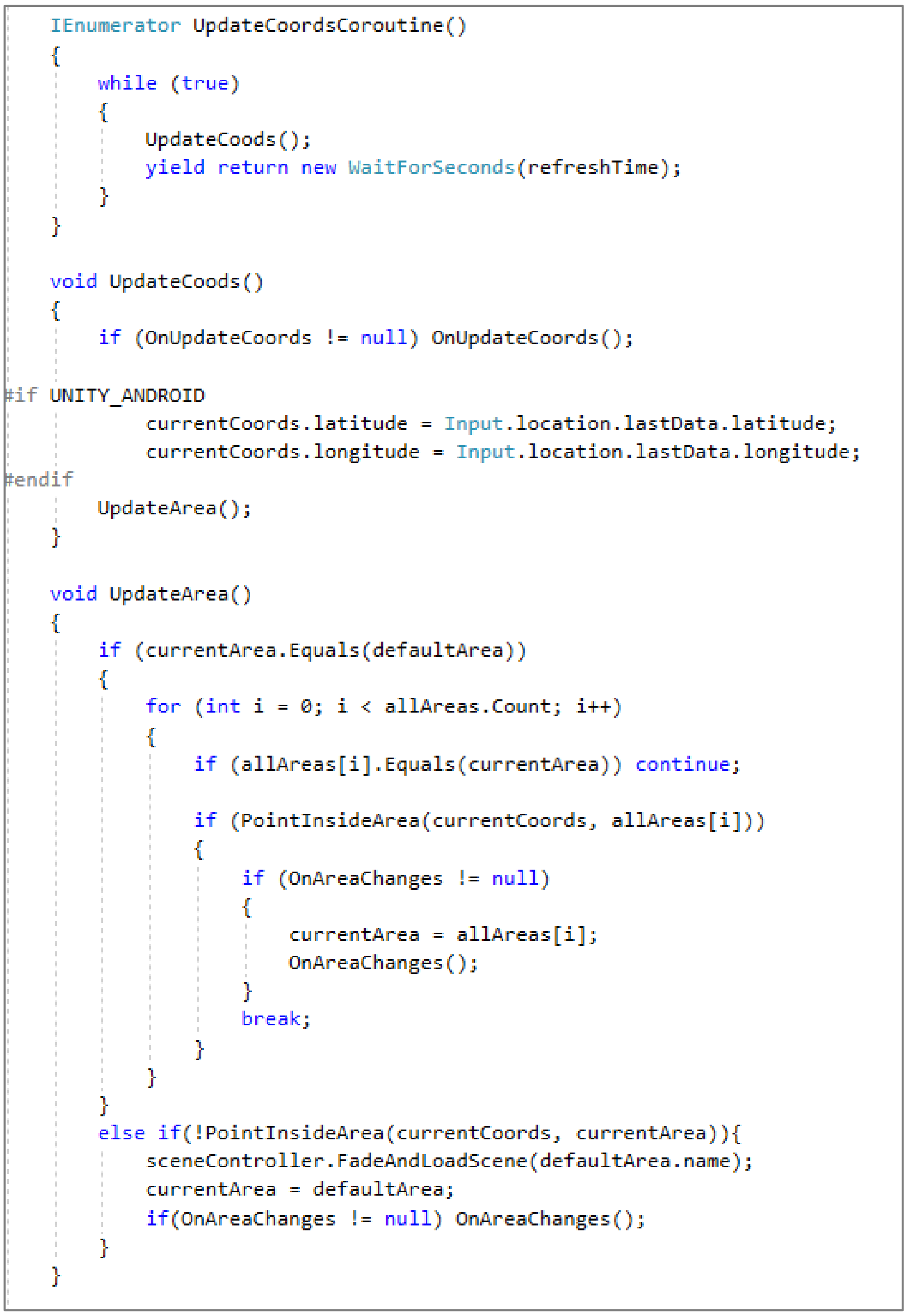
b) **Player has just reached an area**: He/she was in the default scene and it is necessary to inform that there is an available area to be loaded at this time.

c) **Player has just left an area**: He/she was within the range of an area and has just come out of it. In that case, the default scene will be loaded again.

The algorithm to update the area is not complex to understand. First it checks if in the previous update of coordinates the player was in the default area (this is, he/she was out of some area). If so, it is calculated if its position is inside any of the areas of the main array (detailed below) and if so, the current area is updated and the `OnAreaChanges()` event is called so that, external methods subscribed to it in the rest of the code, perform the corresponding action (such as, for example, displaying the "Start communication" button in the default scene to allow the player entering the area).

If it is not in the default area, we check if it is still in an area with the new update. If not, it means that he/she has just left the area he/she was in and, therefore, it is necessary to load the default scene.

In order to get a more specific idea of how the player's current position refresh and the checking works, the C# implementation of the previous functions is shown in the Figure 26.



```
    IEnumerator UpdateCoordsCoroutine()
    {
        while (true)
        {
            UpdateCoods();
            yield return new WaitForSeconds(refreshTime);
        }
    }

    void UpdateCoods()
    {
        if (OnUpdateCoords != null) OnUpdateCoords();

#if UNITY_ANDROID
            currentCoords.latitude = Input.location.lastData.latitude;
            currentCoords.longitude = Input.location.lastData.longitude;
#endif
        UpdateArea();
    }

    void UpdateArea()
    {
        if (currentArea.Equals(defaultArea))
        {
            for (int i = 0; i < allAreas.Count; i++)
            {
                if (allAreas[i].Equals(currentArea)) continue;

                if (PointInsideArea(currentCoords, allAreas[i]))
                {
                    if (OnAreaChanges != null)
                    {
                        currentArea = allAreas[i];
                        OnAreaChanges();
                    }
                    break;
                }
            }
        }
        else if(!PointInsideArea(currentCoords, currentArea)){
            sceneController.FadeAndLoadScene(defaultArea.name);
            currentArea = defaultArea;
            if(OnAreaChanges != null) OnAreaChanges();
        }
    }
```

Figure 26: C# implementation of updating and checking the current position methods.

4.2.2. Checking if a point is inside an area

The methods that are going to be detailed below constitute a key part of the geolocalized system process, since its function is to delimit if the player is inside an area or not. For this, it was necessary to perform a previous investigation on the mathematical basis that requires calculating the distance between two points on a spherical surface such as the Earth to, later, apply a basic mathematical theorem on how to find out if a point is inside of a circle or not.

According to Geodesy[2] principles, it is defined as great-circle distance (also called orthodromic distance) the shortest distance between two points on the surface of a sphere, in this case, on the surface of the Earth (Figure 27). A certain level of abstraction is necessary when calculating this distance, since an ideal situation is assumed where the surface of the Earth does not have irregularities in the terrain. In addition, the algorithm does not understand elevation changes, since it is only based on latitude and longitude measurements. When talking about spaces with curvature, straight lines that defined the distance between two points in Euclidean space are replaced by geodesics, which are circles on the sphere whose centres coincide with the centre of the sphere. They are also called great circles.

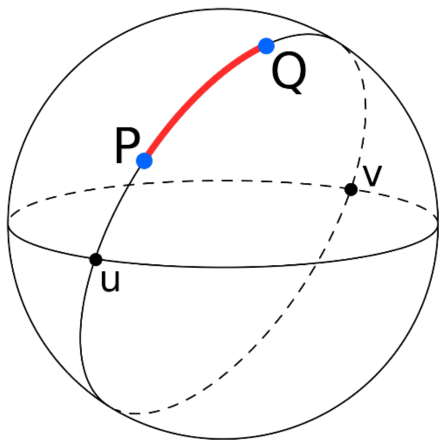


Figure 27: Representation of a geodesic arc formed by two points P and Q. The segment that joins them is their great-circle distance.

[2] Geodesy is the science that accurately measures and understands some Earth properties such as its spherical shape, its orientation in space and its gravitational field.

The mathematical theorem used for such a procedure is the Haversine formula. For any two points on a sphere, the Haversine of the central angle between them is given by:

$$\operatorname{hav}\left(\frac{d}{r}\right) = \operatorname{hav}(\varphi_2 - \varphi_1) + \cos(\varphi_1)\cos(\varphi_2)\operatorname{hav}(\lambda_2 - \lambda_1)$$

Where **hav** is the Haversine function:

$$\operatorname{hav}(\theta) = \sin^2\left(\frac{\theta}{2}\right) = \frac{1-\cos(\theta)}{2}$$

- **d** is the distance between the two points along the sphere's surface,
- **r** is the radius of the sphere, in this case the Earth.
- **φ1**, **φ2**: latitude of point 1 and latitude of point 2, expressed in radians,
- **λ1**, **λ2**: longitude of point 1 and longitude of point 2, expressed in radians.

Next, the mathematical development that had to be based to simplify the formula and be able to write it correctly in the code of the program is going to be exposed. Starting from the definition of the Haversine function, it was chosen to develop it using the squared sine, the first of the expressions of Haversine (with the power of two in the sine operation). Thus, clearing the distance:

$$hav\left(\frac{d}{r}\right) = hav\left(\varphi_2, \varphi_1\right) + \cos\left(\varphi_1\right)\cos\left(\varphi_2\right) hav\left(\lambda_2 - \lambda_1\right)$$

$$\sin^2\left(\frac{d/r}{2}\right) = \sin^2\left(\frac{\varphi_2 - \varphi_1}{2}\right) + \cos\left(\varphi_1\right)\cos\left(\varphi_2\right)\sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right)$$

✓ To simplify and improve legibility, this expression is parameterized.

$$d_3 = \sin^2\left(\frac{\varphi_2 - \varphi_1}{2}\right) + \cos\left(\varphi_1\right)\cos\left(\varphi_2\right)\sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right)$$

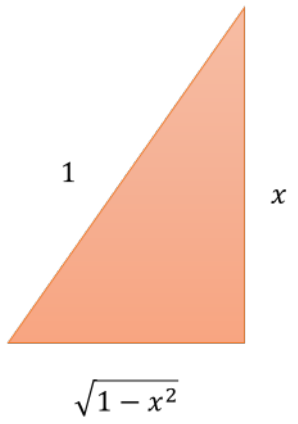
$$\sin^2\left(\frac{d}{2r}\right) = d_3 \Leftrightarrow \sin\left(\frac{d}{2r}\right) = \sqrt{d_3} \Leftrightarrow \frac{d}{2r} = \arcsin\left(\sqrt{d_3}\right) \Leftrightarrow$$
$$d = 2r\arcsin\left(\sqrt{d_3}\right)$$

Now is the time to look for the relationship between the arcsine of a given value (x) and the corresponding arctangent. It can be easily demonstrated by using a right triangle of hypotenuse equal to 1 and a leg equal to the value (x).

$$\sin\alpha = \frac{x}{1} = x \Leftrightarrow \alpha = arc\sin(x)$$

$$\tan\alpha = \frac{x}{\sqrt{1-x^2}} \Leftrightarrow \alpha = \arctan\left(\frac{x}{\sqrt{1-x^2}}\right)$$

$$\arcsin(x) = \arctan\left(\frac{x}{\sqrt{1-x^2}}\right)$$



Therefore, the previous expression is equivalent to the following forms:

$$d = 2r\arctan\left(\frac{\sqrt{d_3}}{\sqrt{1-\left(\sqrt{d_3}\right)^2}}\right) = 2r\arctan\left(\frac{\sqrt{d_3}}{\sqrt{1-d_3}}\right)$$

It was necessary to use the function **`atan2`**, also called the inverse tangent, that takes two arguments (not both equal to zero) and represent the coordinates of an arbitrary point (y, x) in the X/Y plane. Without going into deep mathematical details, this multi-valued function is widely used in programming to avoid the error that the **`atan`** function of an argument produces when it must distinguish between diametrically opposed directions. With the **`atan2`** function, only one arctangent value is calculated from two variables, which symbols determine the resulting quadrant.

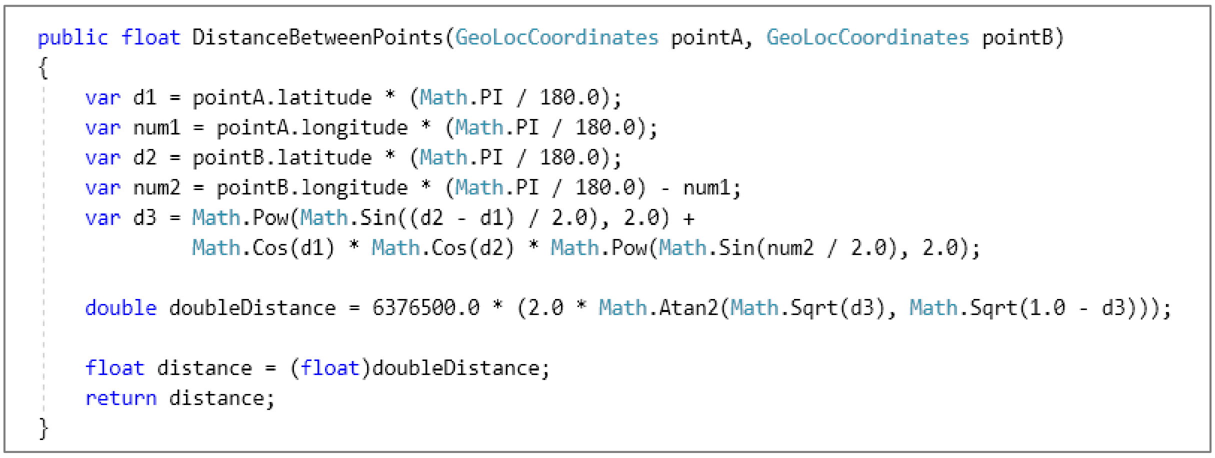
The correspondence between trigonometric functions can be given in several ways:

$$\arctan\left(\frac{x}{y}\right) = \arctan 2\,(x, y) \qquad \arcsin(x) = \arctan 2\left(x, \sqrt{1-x^2}\right)$$

The resulting expression, which was written in the function of C#, was:

$$d = 2r\arctan 2\left(\sqrt{d_3}, \sqrt{1-d_3}\right)$$

And, as can be seen at Figure 28 the result on the code was:



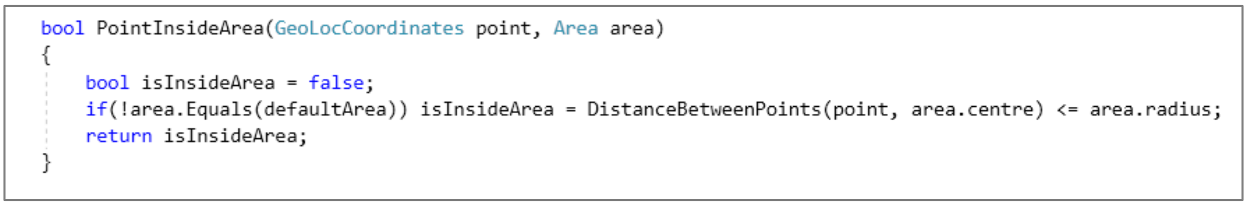
```csharp
public float DistanceBetweenPoints(GeoLocCoordinates pointA, GeoLocCoordinates pointB)
{
    var d1 = pointA.latitude * (Math.PI / 180.0);
    var num1 = pointA.longitude * (Math.PI / 180.0);
    var d2 = pointB.latitude * (Math.PI / 180.0);
    var num2 = pointB.longitude * (Math.PI / 180.0) - num1;
    var d3 = Math.Pow(Math.Sin((d2 - d1) / 2.0), 2.0) +
             Math.Cos(d1) * Math.Cos(d2) * Math.Pow(Math.Sin(num2 / 2.0), 2.0);

    double doubleDistance = 6376500.0 * (2.0 * Math.Atan2(Math.Sqrt(d3), Math.Sqrt(1.0 - d3)));

    float distance = (float)doubleDistance;
    return distance;
}
```

Figure 28: C# implementation of Haversine formula.

After verifying that the formula had been well expressed and the result offered fitted with that granted by other online tools such as Google Maps or calculators of spherical distances, the last step was to check if the point was within the radius of the area. A trivial solution was made to solve this: simply check if that distance is less than the radius of the defined area. If it is, it means that the point is within the area and therefore the player has that available area. If it is greater than the radius value, the player is outside the area. In the following Figure 29 can be appreciated the C# code and in Figure 30 the main idea is explained graphically.



```csharp
bool PointInsideArea(GeoLocCoordinates point, Area area)
{
    bool isInsideArea = false;
    if(!area.Equals(defaultArea)) isInsideArea = DistanceBetweenPoints(point, area.centre) <= area.radius;
    return isInsideArea;
}
```

Figure 29: C# implementation of the function that checks if the player is inside an area.

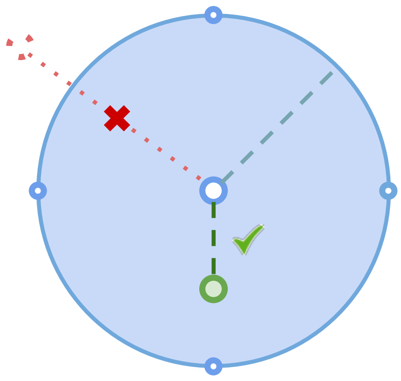


Figure 30: A player is inside an area if the distance from their position to the area's centre is less than the area's radius.

4.3. Scene managing: Persistence and runtime loading

All the actions related to geolocation are carried out by events that are triggered based on a constant updating and checking of the input data provided by the sensors of the mobile device. It is for this reason that it was necessary to devise a way to maintain certain information at a global level (such as game managers, geolocation to update coordinates and save and load data), while another should be loaded or downloaded depending on where the player was. The ideal solution for this was to implement a scene manager based on one persistent scene.

There is a particular default area that is treated as a further area, of coordinates (0, 0) as a "zero zone". This allows that, if the player is not in any area, goes to the default status simply following the same logic that is being followed with the other scenes. Greater robustness and homogeneity in the code is achieved in this way. As we will see later, this default scene is always the same one and does not require to be saved, so neither it will be created in the persistent data file nor it will be specified through the web app. The program will be responsible for automatically creating and loading it at the beginning of the execution (Figure 31).

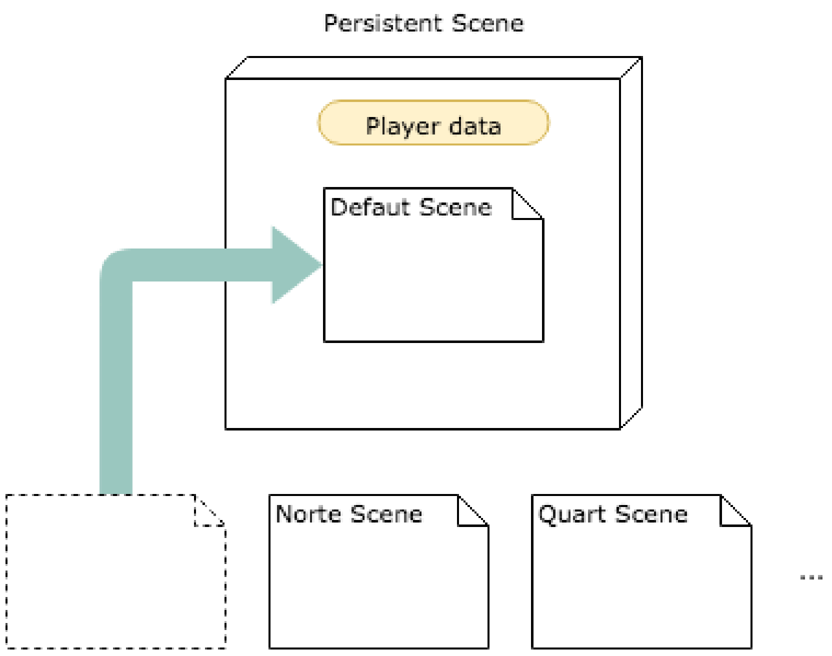


Figure 31: Default scene is loaded when the app starts running.

The persistent scene acts as a "scene container". When the player is within the range of an available area and decides to interact with it, the script responsible for the scene management unloads asynchronously the default scene (the one loaded at that moment) and loads the new scene, as it was a layer system, into a stack over the persistent scene.

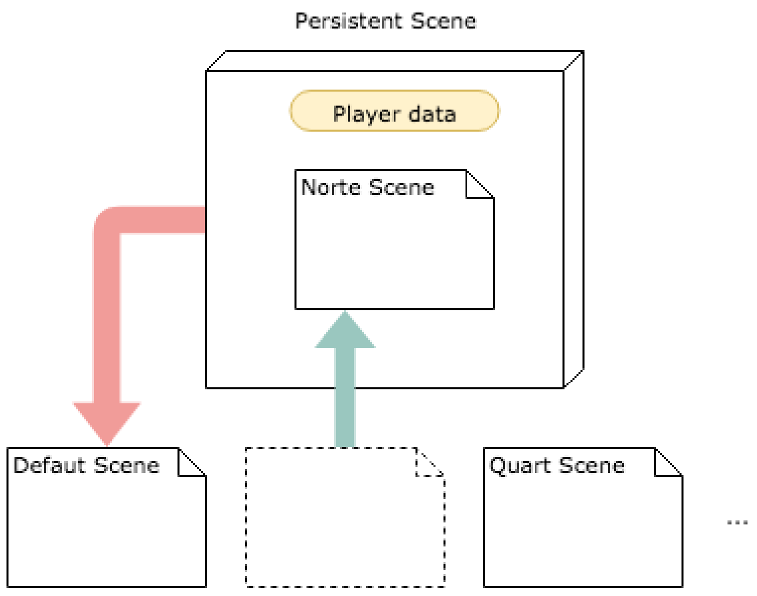


Figure 32: When the player selects to interact with an area, the default scene is unloaded and the area scene is loaded.

It may seem to the reader with knowledge of Unity3D that the solution of implementing each interaction of the areas in different scenes was not optimal, being that the scenes usually resemble each other a lot and it may seem a priori a greater workload both computational and development. The decision has been made for ease of implementation and flexible scalability. Separating each area in a different Unity scene allowed a better personalized control with the dialog system of each place and with the 3D models that are shown. If, for example, the developers want to add a level to be unlocked as a DLC, it would be enough to create a new scene from the previous ones and add the photos, information and the corresponding models. It would be necessary to carry out a similar process to insert new minigames.

4.4. Saving and loading data locally

As we have observed, player progress data is locally stored. This process is carried out by the serialization[3] of the data of a `GameData` type instance, which contains three unique parameters: the status of each area, the percentage of the game completed by the player and the names of the current locked areas. In the following Figure 33 this class is shown more detailed.

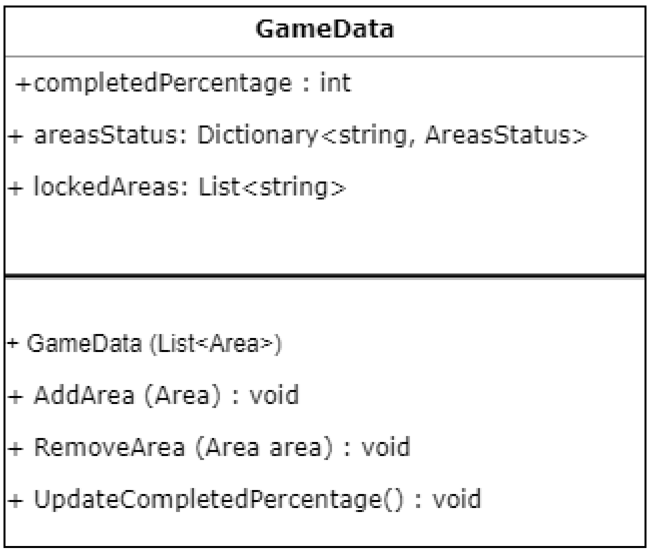

GameData
+completedPercentage : int
+ areasStatus: Dictionary<string, AreasStatus>
+ lockedAreas: List<string>
+ GameData (List<Area>)
+ AddArea (Area) : void
+ RemoveArea (Area area) : void
+ UpdateCompletedPercentage() : void

Figure 33: GameData C# Class.

[3] Serialization is a process to encode a set of data as a series of bytes or in a more universal format to be saved on the hard disk or transmitted.

Note that a string key dictionary and `enum AreaStatus` value has been used to store the status of each area. This has been decided so that the access and modification of these values is O (1) and also much clearer in the writing and reading of the code. Please, see the section Updating the areas status" for more details.

The saved and loaded data corresponds to the `SaveManager` class. At the beginning of this section we saw how the `GameManager` used the `LoadGame` and `SaveGame` functions to interact with this script and save the new `GameData` instance with the updated data synchronized with the server. Let's see more about how these methods work in the following Figure 34:

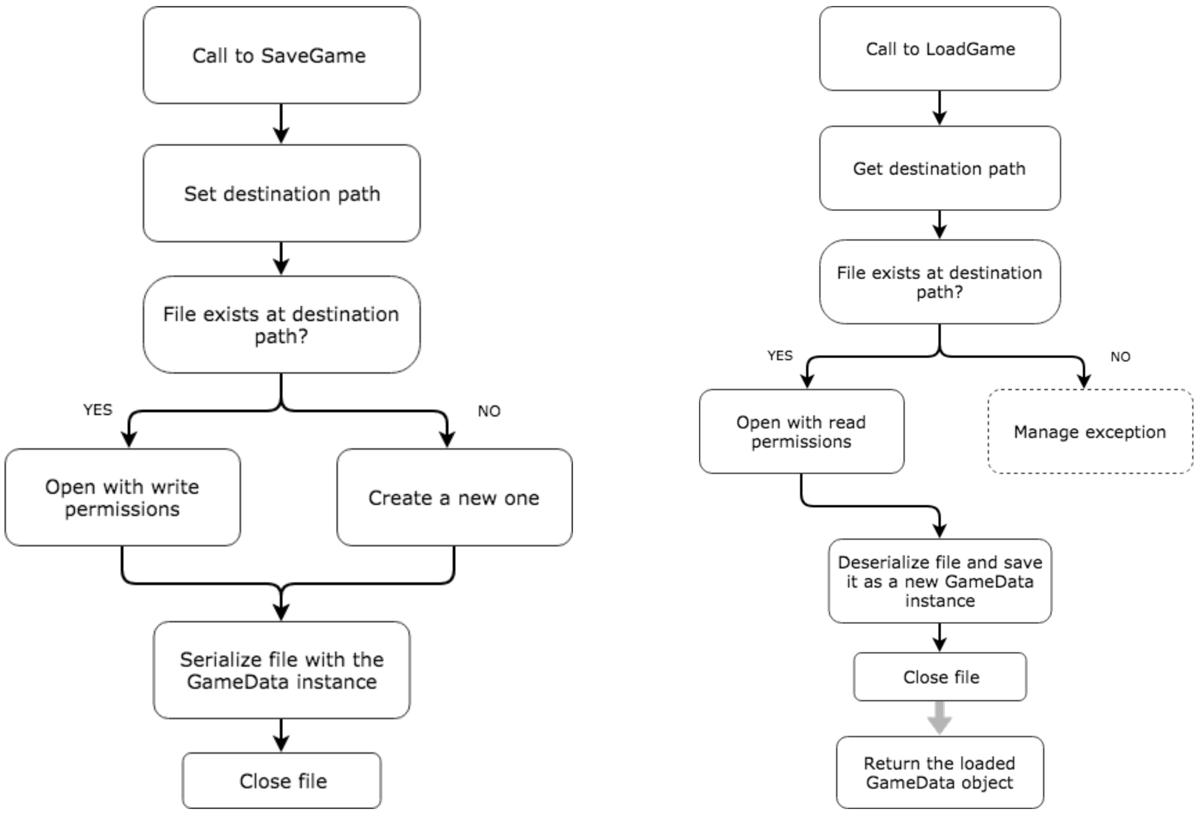

Call to SaveGame
Set destination path
File exists at destination path?
YES
NO
Open with write permissions
Create a new one
Serialize file with the GameData instance
Close file
Call to LoadGame
Get destination path
File exists at destination path?
YES
NO
Open with read permissions
Manage exception
Deserialize file and save it as a new GameData instance
Close file
Return the loaded GameData object

Figure 34: SaveGame and LoadGame methods diagrams.

Later we will see that the method of saving data is called every time that the status of some area is updated. The current instance of `GameData` is modified before saving it: the new status of the area in the dictionary is searched and updated and then the new percentage of completed game is calculated.

Resetting all the status is trivial. Just creating and saving a new default instance of `GameData` (with the list of areas that we want to save as a parameter), the constructor will directly mark them as "unknown" or "available" (as desired).

4.5. Game flow inside an area

Fungus was the tool I decided to use for creating the dialog system within each of the areas. It is important to note that Fungus is not just a tool that allows you to easily display text by screen or contextual menus of options. It's much more than that. By means of the extension of the tool and a little ingenuity to connect it with the external scripts, all kinds of events can be controlled.

4.5.1. Fungus flowchart and event management

All the logic of Fungus is found in a visual environment called flowchart and flows through interconnected block trees. These blocks are sequences of actions of all kinds that are reproduced one after the other. The actions of the blocks can call other blocks and branch depending on the parameters entered or the options that the player has chosen on the screen.

Apart from this, you can also call blocks by events, which are circumstances that occur at a certain time and that trigger actions. Fundamentally, in any area there are three types of events that must be taken into account:

a) **When the game starts**

The block is executed immediately after starting the scene.

➔ In the areas, you must wait for the player to track the image by the camera to activate the augmented reality, so when you start the scene a message indicating this will appear.

b) **When a message is received**

When an event occurs at a certain time, the script that reads it send a broadcast message identified by a name to the flowchart. If a block is configured to receive a message with that identifier, it will execute its corresponding actions when it receives it.

➔ This type of logic is used for delimiting when the AR image has been found and we must show the model. The Vuforia manager script sends a broadcast message and the corresponding block captures it to check the status of the area. When ending a minigame, this type of message is also used.

c) **When an object is touched**

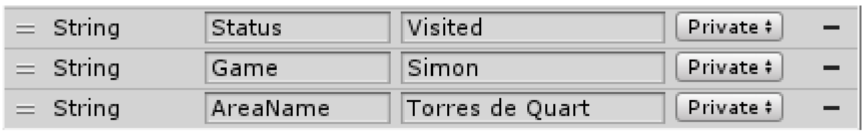
The logic of this type of events keeps a close relationship with the previous ones, but now the message triggers when an object is touched by the user. Then, the corresponding block will execute its actions.

➔ The touch of objects in the areas is produced by touching the disk to load the hologram or each time one an information point is touched.

4.5.2. Updating the areas status

Each flowchart has a set of own variables that can be obtained from certain actions to create a logical flow between the blocks. These variables have been the key aspect to relate the status of the game areas with the flowchart of each of the scenes. The trick was to create an updated copy of the status of the area in an internal variable of the flowchart. When the area is accessed, the status is obtained and the variable is copied. If an event that produces the modification of that status occurs (the game is completed, for example), this status is updated in the game's global data and in turn it is updated in the variable of the flowchart. In this way, we can call blocks or others comfortably depending on the value of this variable.

The following Figure 35 shows the main variables that have been used in each flowchart to define their parameters. In this case, the only dynamic variable is "Status", since the rest are treated as constants. The variable "Game" allows to define what type of minigame is going to be played in that area and the name is defined by its label.



String	Status	Visited	Private
String	Game	Simon	Private
String	AreaName	Torres de Quart	Private

Figure 35: Main flowchart variables used in each area.

Now the Fungus way of working has been contextualized, it is time to observe in greater depth how has this been logically applied to each area. If one block ends and does not call another, the flow of actions of the flowchart stops until the player performs the next action.

The execution cycle of an area is divided into two parts to make it easier to visualize. As shown in Figure 36, when the area is loaded the event that shows the instructions that the player must follow to activate the augmented reality (that is, put the card in front of the camera of the phone) is triggered. Once the identifier of the target image is detected, a broadcast message is sent to the flowchart again and the status of the area is obtained by the `GameManager`. This state will overwrite the flowchart variable to work more efficiently. Depending on the state, as seen in Figure x, some actions or others will be executed. If it is the first time that area is visited or the game has not yet been completed, the monument information will not be accessible. After a brief dialogue, the game will be started. Otherwise, it will be showed the options menu that allows playing the minigame or seeing the monument again.

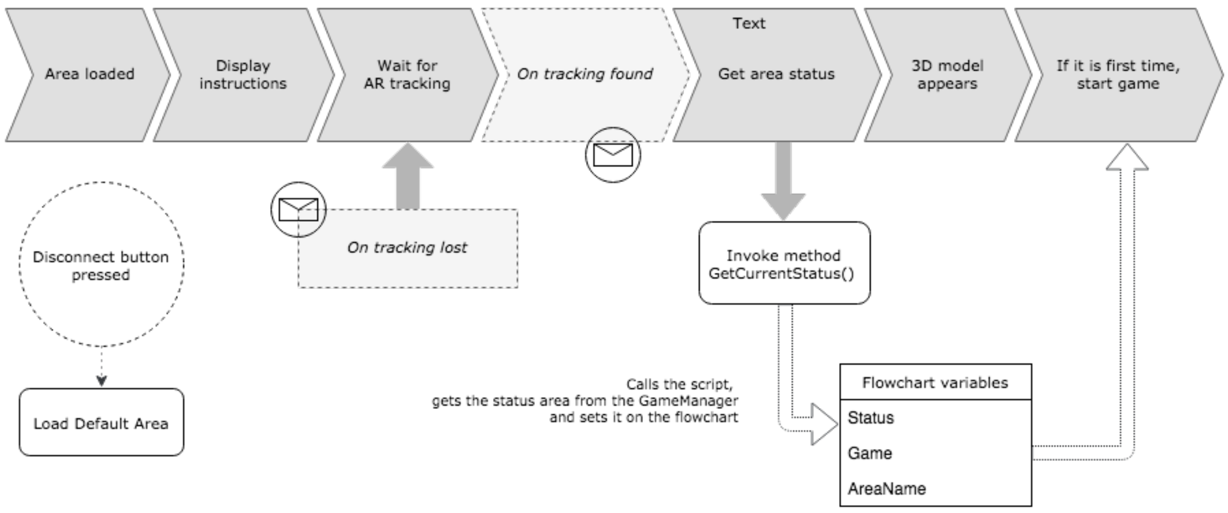

Area loaded
Display instructions
Wait for AR tracking
On tracking found
Text
Get area status
3D model appears
If it is first time, start game
Disconnect button pressed
Load Default Area
On tracking lost
Invoke method GetCurrentStatus()
Calls the script, gets the status area from the GameManager and sets it on the flowchart
Flowchart variables
Status
Game
AreaName

Figure 36: Diagram of the complete implementation cycle of an area (I).

The part in which the minigame is executed is going to be circumvented, since the execution cycle of both minigames has already been detailed at the point of analysis of the application. The attention will now be focused on what happens when the minigame ends. In any case, a broadcast message will be sent to the flowchart. If the player has not passed the game, the game restarts. If it has done so, it is checked again if the area was completed (that is, if the minigame was accessed from the options menu) by means of the variable of the flowchart that was established when entering the area. If so, it returns to this menu. If not, the status is updated in both the flowchart and the `GameManager` dictionary and the disk with the information of the building's hologram is displayed. When the player touches the disc, another event is triggered that makes the monument appear.

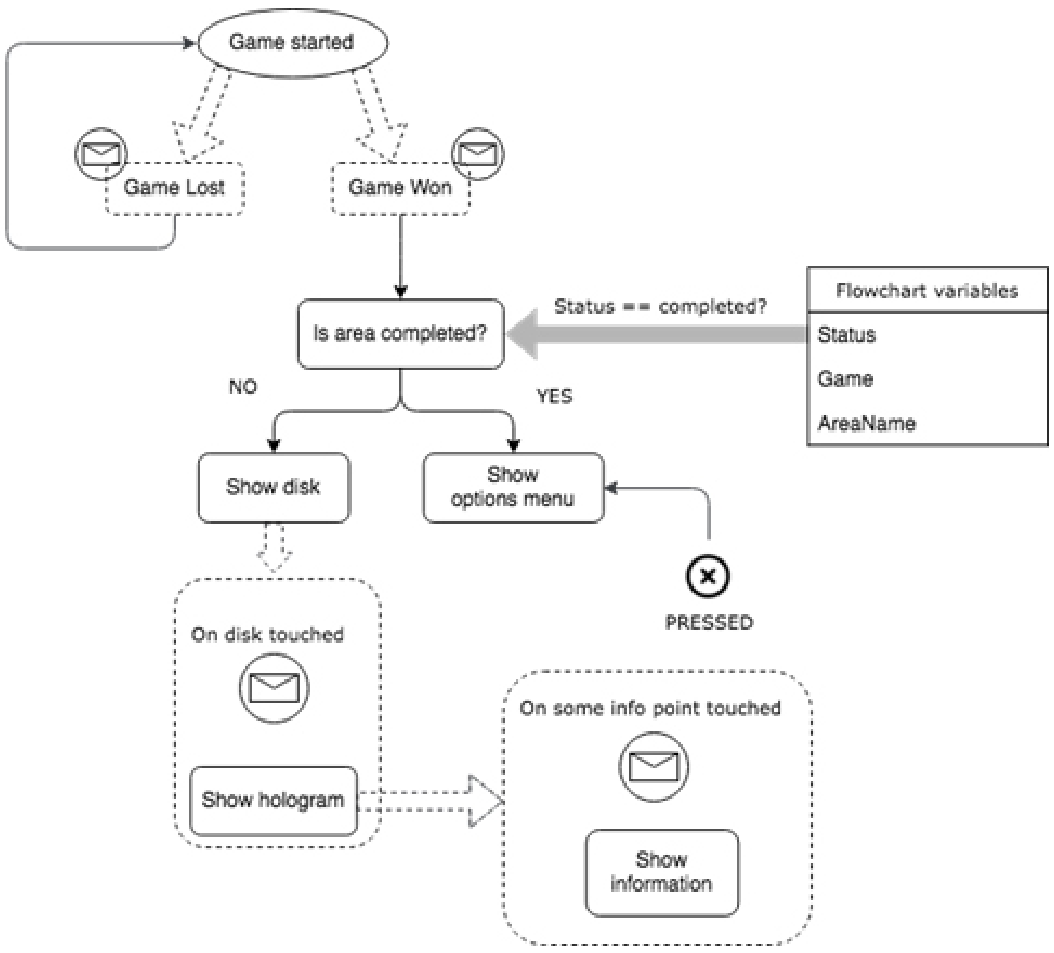


Figure 37: Diagram of the complete implementation cycle of an area (II).

4.6. Minigames

In this section, each of the augmented reality minigames implemented will be explained, as well as the particular techniques that have been used to build their main mechanics will be further detailed.

4.6.1. Memory Game

4.6.1.1. Overview

In the memory minigame, a sequence of illuminated objects is displayed initially, and the player must repeat it exactly in the same order. The difficulty increases progressively, since there will be more objects that are added to the final sequence and that have to be remembered. When one sequence is failed, another is randomly generated from the same number of illuminated objects, this is, the same level.

This game is highly inspired by the classic "Simon says", but with the peculiarity that the two-dimensional space is broken to make way for augmented reality in three dimensions (Figure 38). This is a factor of difficulty (and fun) extra for the player, since he/she must also take into account the spatial component.

The implementation has been strongly parameterized in order to give the game as scalability as possible. This allows infinite levels of difficulty, since, in addition, the arrangement of the objects in the space is totally customizable. Some of the parameters are:

- Total number of objects.
- Elapsed time between when an object and the next one light up.
- Initial level.
- Number of game rounds.

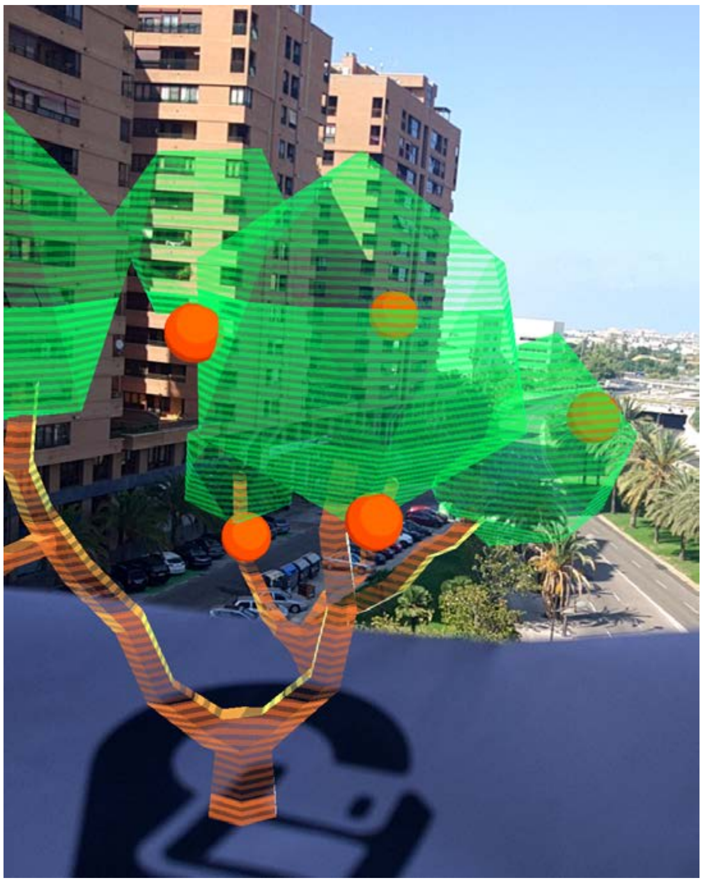
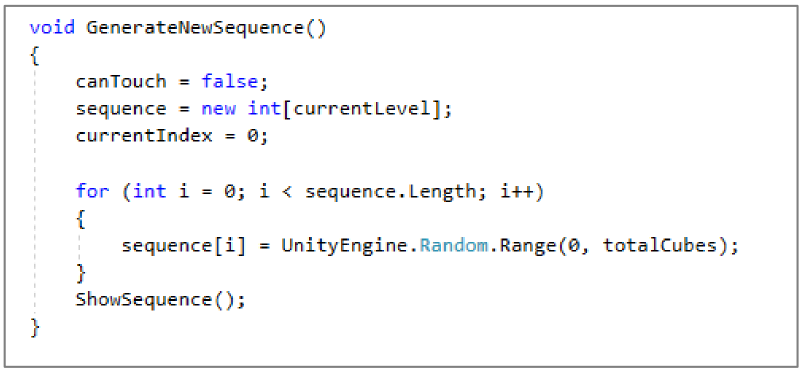


Figure 38: Memory game in augmented reality.

4.6.1.2. Random sequence generation

The generation of the sequences in a random way was implemented through the use of two arrays. To begin with, it was necessary to refer to all the objects in the scene that were going to be part of the sequence and that could be touched later to solve it. The best option was to save them in an array, with each index of that array identifying the object. The second array, of a length equivalent to the level of the game in which we find ourselves, stores the sequence of objects themselves based on a sequence of indexes (from array of objects). When generating a new sequence, a new array is simply created depending on the level and it is traversed assigning to each index a random integer value among all those corresponding to the playable objects. They can, of course, repeat values. Finally, a counter is updating the index of the array object that the player must press at that moment. In this way it is possible to check if the sequence succeeds. At the moment in which this index coincides with the length of the array, it means that all the objects (played in order) have been successful and must either generate a new sequence or end the game. If the player makes a mistake, the level will restart with a new sequence.



```
void GenerateNewSequence()
{
    canTouch = false;
    sequence = new int[currentLevel];
    currentIndex = 0;

    for (int i = 0; i < sequence.Length; i++)
    {
        sequence[i] = UnityEngine.Random.Range(0, totalCubes);
    }
    ShowSequence();
}
```

Figure 39: GenerateNewSequence() method C# implementation.

For the set of sequential animations of this game in particular, DoTween library [38] has been used, as it allows to manage this type of events one after another in a magnificent way. By nesting sequences of animations and parameterizing them, the effects of coloring the oranges and creating all the necessary visual feedback were achieved.

4.6.2. Pattern recognition game

4.6.2.1. Overview

During the pattern recognition mini-game, several icons will go through the screen from certain places. The player must destroy them by means of the symbol that indicates his figure in the precise moment in which these are within a delimited range that will vary in size. The player will win if he/she draws correctly a certain number of symbols, depending on the difficulty level and the total number of appearances.

This minigame was designed in such a way that it could be customized and expanded. Therefore, the difficulty levels have been parameterized based on the following values:

- Maximum and minimum symbol speed.
- Time elapsed between appearances of symbols.
- Total number of symbols that will appear.
- Percentage of minimum success necessary to overcome the level.
- List of symbols generation points.
- List of ranges for this level of difficulty.

With all the previous parameters, it is possible to define as many levels of difficulty as desired, and thus endow this game with a practically infinite variety when it appears in the areas.

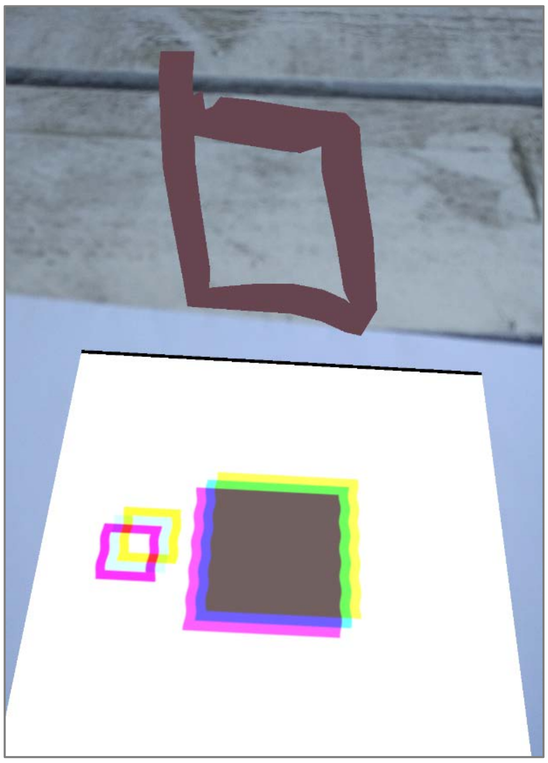


Figure 40: Gesture being drawn while a symbol comes to the range.

4.6.2.2. Randomness

The reader will have appreciated that certain values, such as the speed of symbols, appear as a bounded range. This is because the game was given a randomness factor in several aspects:

- The symbols appear at a random generation point.
- The speed of the symbols is between the two given values for that level of difficulty.

When a symbol is drawn and it is within a range, an event that checks by means of a recognition algorithm if the line drawn is correct and corresponds to the symbol within the range is launched. If successful, a score is arbitrarily weighted based on the speed of the successful symbol and the size of the range at the time is destroyed.

4.6.2.3. P-Dollar Algorithm

The pattern recognition algorithm that uses this little game is called *P-Dollar Point-Cloud Gesture Recognizer* (commonly abbreviated $P). It is attributed to Radu-Daniel Vatavu (University Stefan cel Mare of Suceava), Lisa Anthony, (University of Maryland-Baltimore County, currently at the University of Florida) and Jacob O. Wobbrock (University of Washington) [39].

As they explain, "In machine learning terms, $P is an instance-based nearest-neighbour classifier with a Euclidean scoring function." In the context of pattern recognition, it is necessary to extract certain characteristics of the sample to be classified on which it is going to work. In this case, this set is given by coordinates in a two-dimensional space, since the patterns are lines formed by the union of points. The classification procedure to be used is geometric (not statistical) and, as mentioned above, is based on clustering. Gestures are always compared as collections of points, using the supervised classification method of the nearest-neighbour (kNN with k = 1) [40].

The algorithm classifies a candidate gesture against a set of training samples that have previously been defined in a database (in this case, in an XML file). At the time of classifying it, it is compared with all the stored gestures looking for the one that provides the highest score.

In Figure 41 it is possible to see how the algorithm acts. Two sets of points which correspond to two different strokes, black and red, are represented. The red one corresponds to the set defined in the database (that is, part of the training samples). The black one corresponds to the stroke drawn by the user at a time of execution. While the player is defining the stroke, points are generated from time to time. When the user stops drawing, these two sets of points are compared and a quantitative value that expresses the similarity between the two is obtained. If this number is satisfactory (depending on how it was defined in the game), the candidate set is given as valid and, therefore, the gesture is identified as the gesture stored in the database.

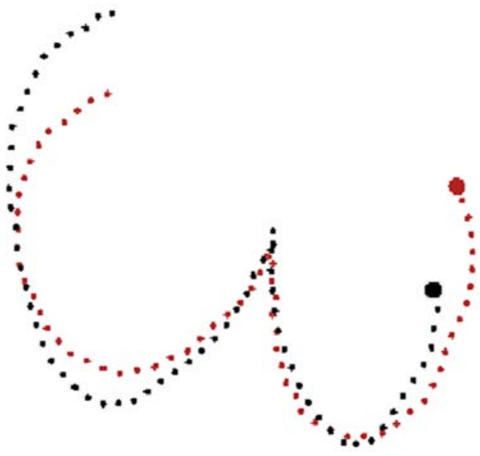


Figure 41: The black points set defines a candidate gesture that is classified against the red set, which was previously defined.

Since the game is going to run on a mobile device, it was decided to simplify the algorithm and implement a reduced version called $1 Unistroke Recognizer [41]. It maintains the same classification logic, but only with gestures made with a continuous line. As the game should be frenetic, it is logical that the symbols are the simplest and fastest to draw (as well as to recognize). The way it works is similar to the $P one.

4.7. Reactive web manager

As expressed before, during the development a lack of flexibility was detected in the project that could hinder the action of adding new areas in a simple and fast way, in order to allow the players accessing to new information. Due to the project's structure, the only way to make visible new areas was a new compilation with the consequent update in the digital market where the application was (in this case, Play Store). From the developer's side, this method means an uncomfortable task, since although locally the areas are stored as objects of code and can be easily edited thanks to Unity serialization and mechanisms such as Scriptable Objects, it was necessary to write in an imprecise and non-intuitive way the numerical data of the areas.

For this reason, the need to implement a small tool that allows remotely add, remove or modify game zones was evaluated. Ideally, the application would load that information reactive just at the beginning of each execution and the data of the areas would be updated and synchronized locally. The cloud data always has the priority, so the local data has to be modified if something changes.

The development of this functionality consisted of three basic components: a server in which to host the information, a controller in Unity capable of consuming it and processing it well and a web client capable of generating it. In Figure 42 it is observed how the three components are communicated.

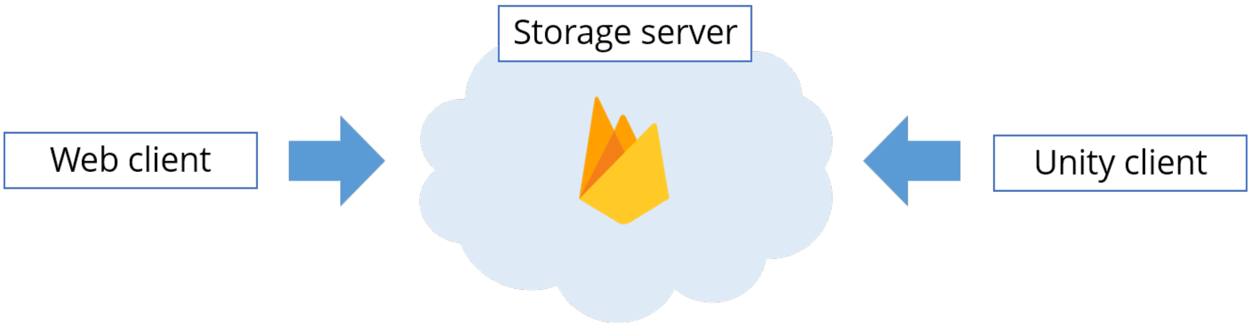
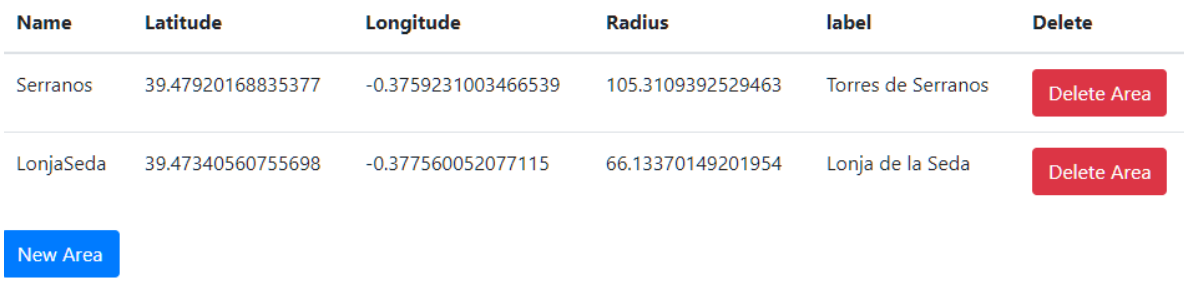


Figure 42: Relation between the both web and Unity clients with the cloud storage server.

The complete development of an area would be much more complex, since it would be necessary to include several fields with questionnaires associated with each type of minigame, 3D model to display and each type of information that will be transmitted in the area. These functionalities would require a more extensive research work in web development and storage, and integration with the current local system. However, it was decided to develop a basic version that would allow defining the areas by their basic attributes: Name, radius and coordinates expressed in latitude and longitude. In addition, a tag attribute was added to identify the area by its full name that would later be attached to the project. The great advantage of editing the areas in this way was that thanks to a Google Maps library it was possible to edit the coordinates in a visual way. This means that adding an area becomes much more intuitive. In the following Figure 43 the resultant user interface with a table structure is showed.



Name	Latitude	Longitude	Radius	label	Delete
Serranos	39.47920168835377	-0.3759231003466539	105.3109392529463	Torres de Serranos	Delete Area
LonjaSeda	39.47340560755698	-0.377560052077115	66.13370149201954	Lonja de la Seda	Delete Area

New Area

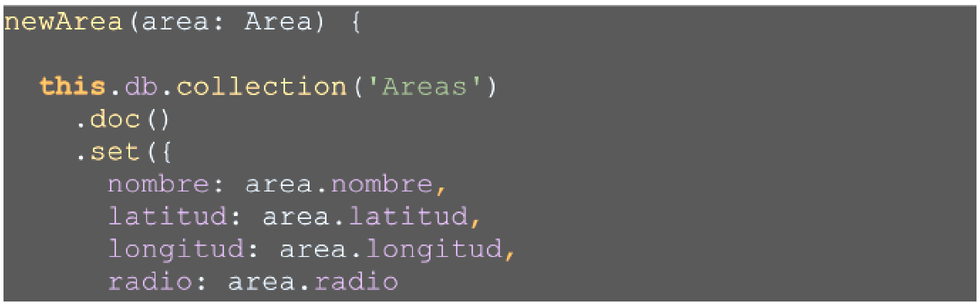
Figure 43: Table with all areas data. Each row represents an area and the columns mean its attributes.

After making a brief analysis of the available technologies, it was decided to use Firebase for the server where the data would be stored. Firebase is a technology offered by Google that allows control over a database in real time and, in addition, greatly facilitates the reactive programming[4] thanks to a JavaScript API. An updated version of Angular 4, also perfectly integrated with Firebase, is used for the client.

Next, the most important points of the implementation of both web clients, Angular and Unity, will be described.

4.7.1. Creating the data with Angular

For the generation of the Web, the Angular CLI library has been used. Angular is programmed with TypeScript[5], so once with the attributes of the conceptually defined areas, it was easy to implement them in a similar way to their structure in Unity thanks to this typed language. Next, a service controller was created that communicates with the database in Firebase. This is possible using the AngularFire2 library [42]. Internally, a code similar to the creation of an object is used to add a new area to the list (Figure 44).


```
newArea(area: Area) {

  this.db.collection('Areas')
    .doc()
    .set({
      nombre: area.nombre,
      latitud: area.latitud,
      longitud: area.longitud,
      radio: area.radio
```

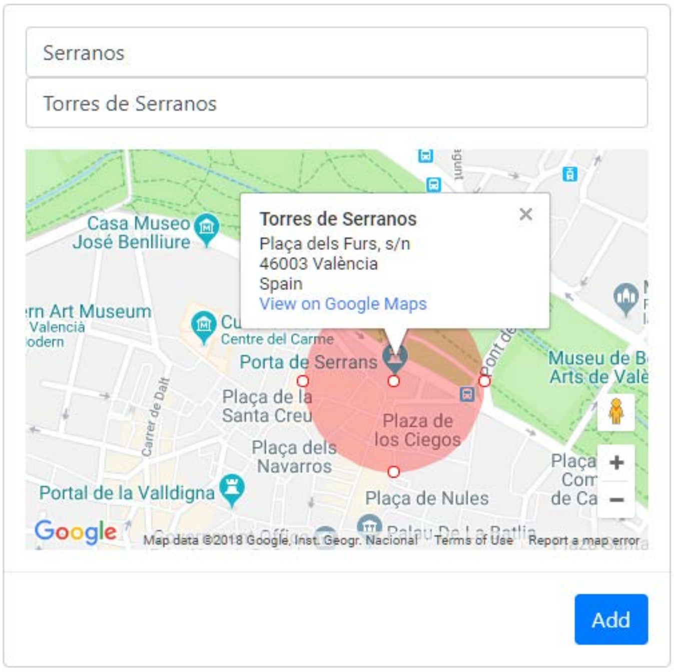
Figure 44: Constructor method for creating a new Area instance in TypeScript..

[4] Reactive programming is a set of patterns and techniques that allows current applications to react to events (event-driving), react to load (scalability), react to failures (resilience) and react to users (responsively).

[5] TypeScript is a language similar to JavaScript that can be compiled into JavaScript itself. It is a typed language, which means that it is possible to create classes and methods similar to C #. This was an advantage at the structural level, since the objects in the areas were created in a very similar way.

Once the class and the communication service were created, the last step was to create the component and the web template that allowed adding and deleting areas comfortably. It was decided to design it in the form of a table, with a column corresponding to each attribute. To obtain the coordinates data introducing them visually, it was used the AngularGoogleMaps library [43], which allows to display maps and perform simple operations with them.

The user, that is, the developer, will simply have to select the "Add" button and a pop-up will appear showing the map of the city centre (Figure 45). Then, by clicking on any game, an area with a modifiable range will be displayed. The user will choose how they want the area to be and will put a label on it. Once the operation is confirmed, the area will be added to the list with the established parameters. When the application opens next time, the new area will be available to access it.


Serranos

Torres de Serranos

Torres de Serranos
Plaça dels Furs, s/n
46003 València
Spain
View on Google Maps

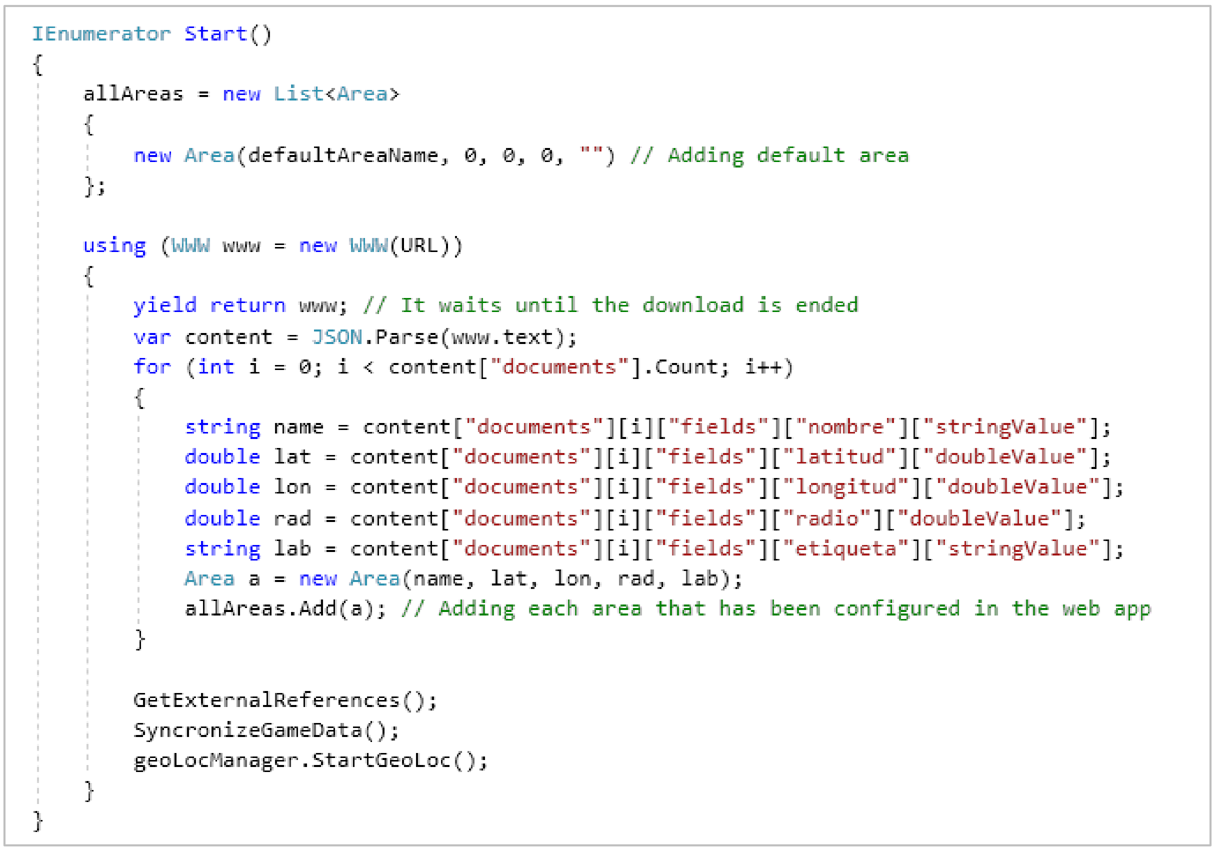
Figure 45: Google Maps pop-up used to add an area.

4.7.2. Getting the data with Unity

For the part of the integration in Unity, it was necessary to modify the `GameManager` class to add a coroutine to check if there had been modifications in the cloud as soon as the application was executed and, if so, to update and synchronize all the data of the areas. The algorithm used for such a procedure has been detailed in the "Synchronizing the areas data" section. In the Figure 46 can be observed how the callback call that collects the data in JSON format from the given URL and converts them to application data has been implemented in C#.

To carry out the web request, the Unity class called `WWW` was used. Due to the way the coroutines are executed, the request is made asynchronously, so this ensures that all the data will be loaded when the execution cycle continues.

With the created Area class, the challenge here was to fragment the JSON file into its different fields and then create Area objects depending on them and add them to the list to be compared with the local data in the `SyncronizeGameData()` method below. For the management of this response package, an external library called SimpleJSON [44] was used, which facilitates the work with JSON-type data structures and acts as a manager.



```
IEnumerator Start()
{
    allAreas = new List<Area>
    {
        new Area(defaultAreaName, 0, 0, 0, "") // Adding default area
    };

    using (WWW www = new WWW(URL))
    {
        yield return www; // It waits until the download is ended
        var content = JSON.Parse(www.text);
        for (int i = 0; i < content["documents"].Count; i++)
        {
            string name = content["documents"][i]["fields"]["nombre"]["stringValue"];
            double lat = content["documents"][i]["fields"]["latitud"]["doubleValue"];
            double lon = content["documents"][i]["fields"]["longitud"]["doubleValue"];
            double rad = content["documents"][i]["fields"]["radio"]["doubleValue"];
            string lab = content["documents"][i]["fields"]["etiqueta"]["stringValue"];
            Area a = new Area(name, lat, lon, rad, lab);
            allAreas.Add(a); // Adding each area that has been configured in the web app
        }

        GetExternalReferences();
        SyncronizeGameData();
        geoLocManager.StartGeoLoc();
    }
}
```

Figure 46: C# asynchronous request to the web service.

With all of the above, the reactive reception system for content areas generated in the cloud is complete. The developer now is able to edit the information of the areas that wants to apply to the game at any time, specifying where the areas will be physically. In all installed applications this information will be updated the next time they are executed, without avoiding to generate a new **.apk** file and upload it to the marketplace.

4.8. Google Play Games Services

To finalize the application development, Google Play Games Services (GPGS) [45] were incorporated, which give the game more versatility and allow it to have intrinsically a community of users based on their respective Google accounts. First it will be defined what Google play games Services are about and what they allow to add to the development and, later, it will be seen how they have been implemented to extend the application functionality.

The GPGS, as the name suggests, is the set of services offered by Google to connect an application with its platform for games for smartphones called Google Play Games. Access to this platform allow the application users sign in with their Google account and be part of a global ranking with the results obtained in the game (Figure 47). In addition, it also allows you to configure a wide set of unlockable achievements, cloud storage for saving data and multiplayer options. Using them, the life of the application and its game system is greatly extended, since it goes from being local to being hosted by a community of players that can communicate with each other.

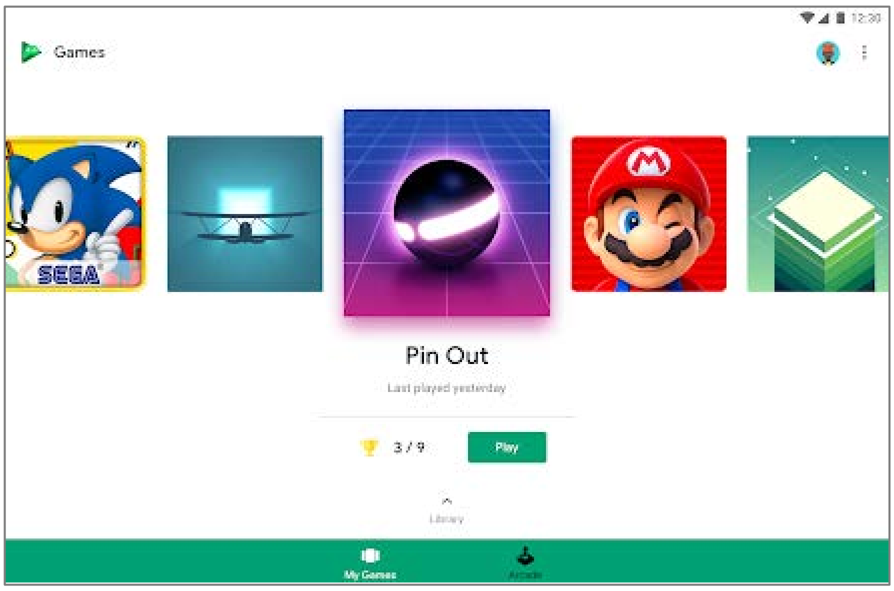
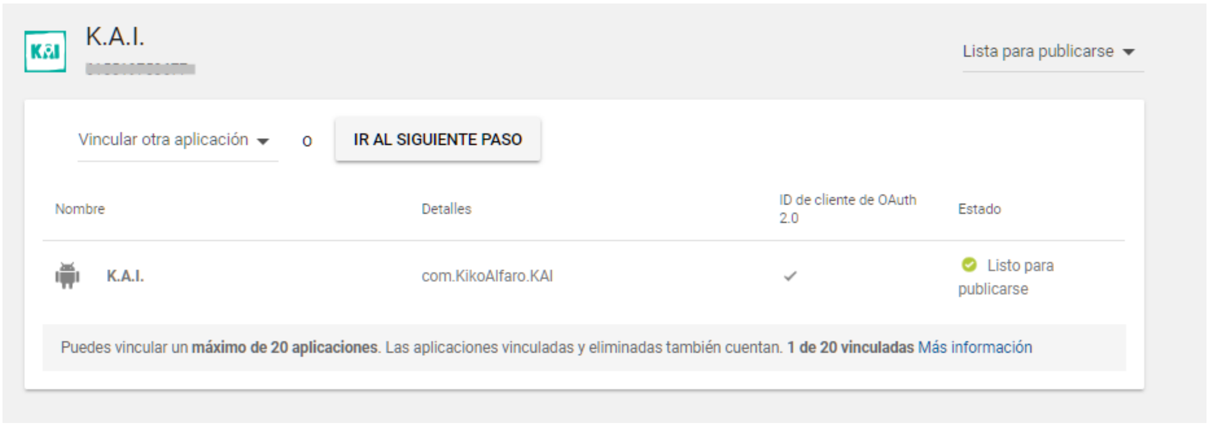


Figure 47: Google Play Games Services official app main menu.

To facilitate the visualization and edition of statistical services, such as Google Analytics [46], the creation of achievements and leaderboards and the tracking of the game data in the Play Store, the Google Play Console [47] is used. From this platform, it is possible to configure the resources that will be used in the application and then download them in XML format and integrate them into the game engine. This resources can be configured in the scripts related and there trigger some events and update the data in the cloud at run time.

GPGS is well integrated with Unity, as Google provided, a few years ago, a public API for developers hosted on the Github [48] that can be downloaded as a Unity Package and configured in the editor. The most complex and delicate part of the process is to configure the signature of the application, since it is necessary to have basic knowledge about the meaning of the identifiers (commonly called tockens) that are used to carry out this signature.

In order to explain this process briefly, a new Keystore was created for the application, which is a file that contains the identity that securely connects the game to the Google Play Console application. The Keystore is generated from a name, an alias and their respective passwords and it is signed with a key SHA1. This key must be accessed by command console and it is necessary to link the local application with the Google Play Console one (Figure 48).



K.A.I.

[redacted]

Lista para publicarse

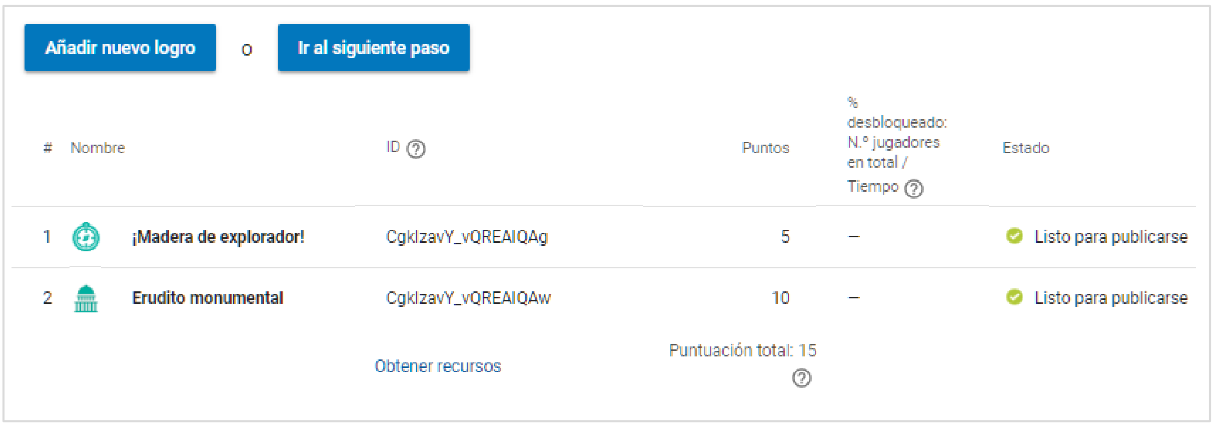
Vincular otra aplicación o IR AL SIGUIENTE PASO

Nombre	Detalles	ID de cliente de OAuth 2.0	Estado
K.A.I.	com.KikoAlfaro.KAI	✓	Listo para publicarse

Puedes vincular un **máximo de 20 aplicaciones**. Las aplicaciones vinculadas y eliminadas también cuentan. **1 de 20 vinculadas** Más información

Figure 48: "Linked applications" tab of the K.A.I.'s games services on the Google Play Console.

Once the base configuration is done, it is time to create the leaderboards and the achievements. The API allows to create both instantaneous achievements (they are only achieved once) and incremental achievements (a counter is added repeatedly until the full achievement is achieved), as shown in the list of the Figure 49. When everything is ready, the resources are obtained, they are incorporated in Unity and the functions of the library are implemented in order to execute the actions of identification of the user and of unlocking achievements from code.


Añadir nuevo logro o Ir al siguiente paso

#	Nombre	ID	Puntos	% desbloqueado: N.º jugadores en total / Tiempo	Estado
1	¡Madera de explorador!	CgkIzavY_vQREAIQAg	5	–	✓ Listo para publicarse
2	Erudito monumental	CgkIzavY_vQREAIQAw	10	–	✓ Listo para publicarse
		Obtener recursos	Puntuación total: 15		

Figure 49: "Achievements" tab of the K.A.I.'s games services on the Google Play Console.

For example, to report that an achievement has been completed and update it in the active user's account, the `ReportProgress` function of the API must to be called, passing as parameters the token corresponding to the achievement to be unlocked and the numerical amount of progress of that achievement. In the following Figure 50 there is an example of an implementation in C#.

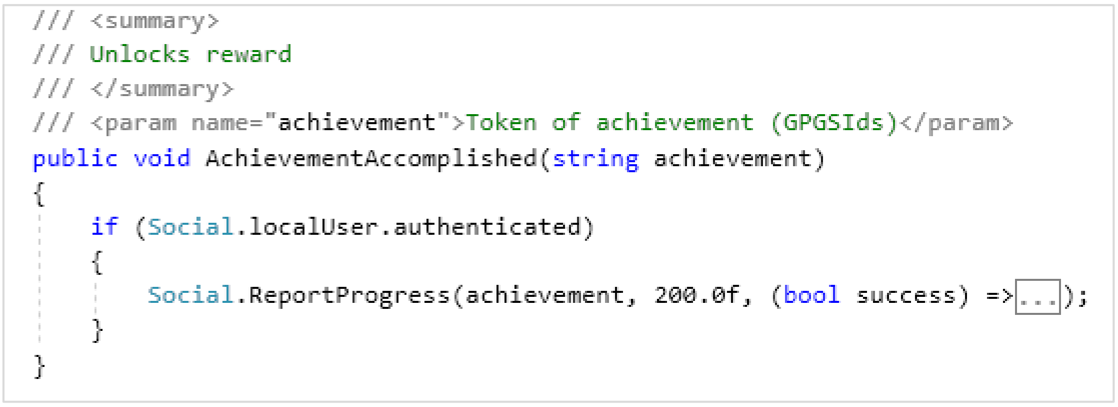
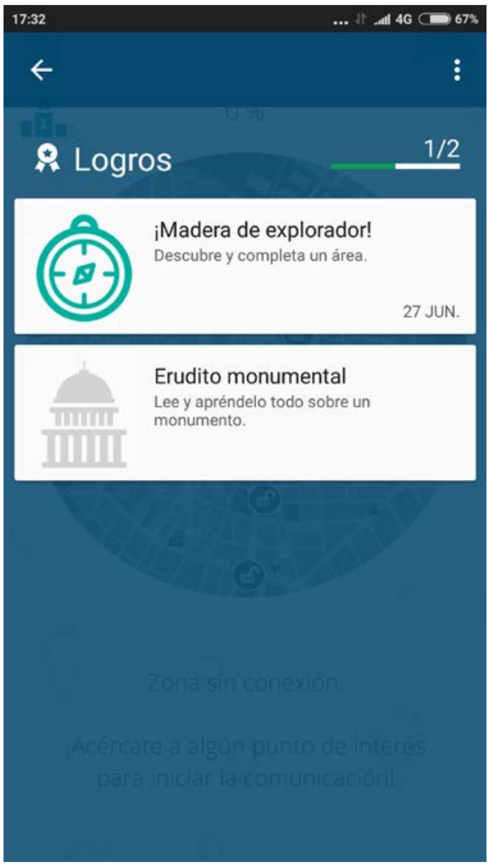

```
/// <summary>
/// Unlocks reward
/// </summary>
/// <param name="achievement">Token of achievement (GPGSIds)</param>
public void AchievementAccomplished(string achievement)
{
    if (Social.localUser.authenticated)
    {
        Social.ReportProgress(achievement, 200.0f, (bool success) =>...);
    }
}
```

Figure 50: C# example of call to `ReportProgress` function to indicate that an achievement has been accomplished.

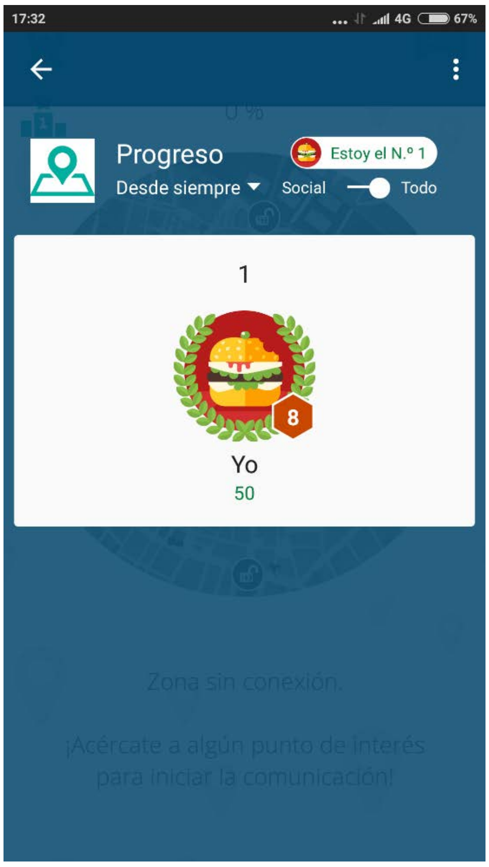
The final result of the implementation of these Google services in the application is shown in the following Figure 51. The user accesses this information through two buttons located in the upper left of the default scene and thus he/she can check its progress associated with his/her Google account. The player also has the option of sign in or sign out with a third additional button in the upper right corner (Figure 52).


17:32 ... 4G 67%

Logros 1/2

¡Madera de explorador!
Descubre y completa un área.
27 JUN.

Erudito monumental
Lee y apréndelo todo sobre un monumento.


17:32 ... 4G 67%

Progreso
Desde siempre
Estoy el N.º 1
Social Todo

1
Yo
50

Figure 51: Left: Achievents pop-up. Right: Leaderboards pop-up.

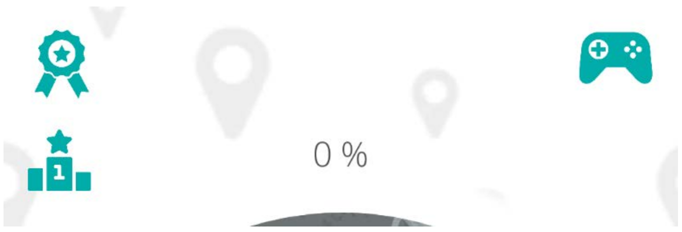


Figure 52: Google Play Services HUD buttons.

5. Artistic design

From the beginning, the artistic design of the project was strongly linked to give a distant futuristic atmosphere to the elements that we found, to transmit the sensation of contrast sought between ancient and past elements and its holographic figure represented by the robot. In this section, the most important points of artistic design and how they have been focused are exposed.

5.1. Colour schemes

In every visual project it is necessary to make a good choice of the colour palettes that are going to be used, since it will be crucial for the user that the contrast, harmony and chromatic coherence that they find throughout the work complement each other. In order to transmit a technological and futuristic feeling, a strong emphasis was placed on three grey tones (#626262, #9CA4A5 and #F3F3F3), since they contribute this mechanical and electronic extravagance and they imply an imbalanced situation and a non-complete happiness that must be resolved at the same time. In addition, other two more saturated colours join the scheme and create a contrast with the predominant grey: Persian green #00ACA9 and Texas rose #FC565A.

These last colours allow to highlight elements or relevant items in the context menus of the application. On the one hand, green is used as a synonym of progress in the game and at all times it is intended that the player identifies it. The green colour will be displayed when an action is performed correctly, as well as to show on the map that an area is completed or the bar of the completed game percentage. On the other hand, red will have opposite uses, generally. This colour will indicate on the map when an area is blocked and, therefore, is inaccessible. In addition, in the memory minigame, when the player fails the sequence, all the oranges light up with a very similar reddish colour and transmit that there has been an error. In addition to this, this colour is used in the map to indicate the current position of the player when it is within the range of some area. It was decided to wear this colour for its greater saturation with respect to the rest and by mere convention with the usual geolocation pin icon. This first colour scheme is mostly used in the map scene and it is shown in the Figure 53.

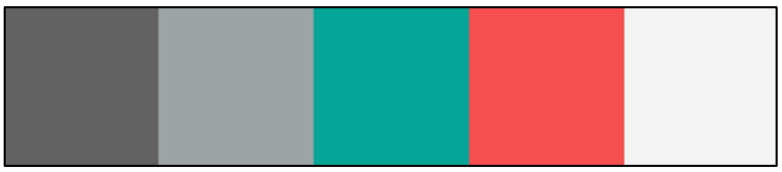


Figure 53: Map scene colour scheme.

A second important point regarding colour is the use that is given in holograms (Figure 54). The problem of holograms, as will be seen below, was matching a realistic and enough visible alpha channel. Transparency really gave the feeling of artificial generation that was sought, but sometimes it confused the way of interpreting the monument. The solution was to reach an intermediate point by using a semi-opaque cel shading effect, which used the light of the scene and the custom colour of the outer line in the models. So interesting marine green and yellow tones were obtained (#407060, #93BFB2, #9BF1BB, #F2F2BD and #F2F5F3). These tones are the ones that have finally defined the hologram models and the exterior of the K.A.I. model, since they provide a better readability of the models in space and are coherent with the rest of the application.

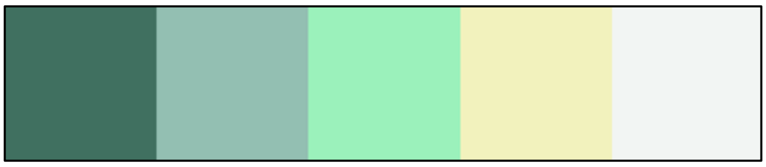


Figure 54: Cel shaded holograms and K.A.I. colour scheme.

5.2. The user interface

Commonly, the user interface (UI) is defined as the channel or medium that is between the computer program and the user, and allows him/her to communicate both by sending input information, as well as by consuming the output information generated by the game. The objective of a good UI is to be intuitive, easily recognizable and not saturated with too much items, in order to provide the user with that feeling of comfort when using the product. In this case, the UI is focused on the main screen of the map, with its respective elements, the dialogues, the information shown in each one of the tourist points of the city and the other contextual buttons of the app.

To design the interface of the map (Figure 55) several sketches with different forms were made: from a square map on the screen to finally ending in a round shape that stylized the scene better and was in line with the icons indicating the areas.

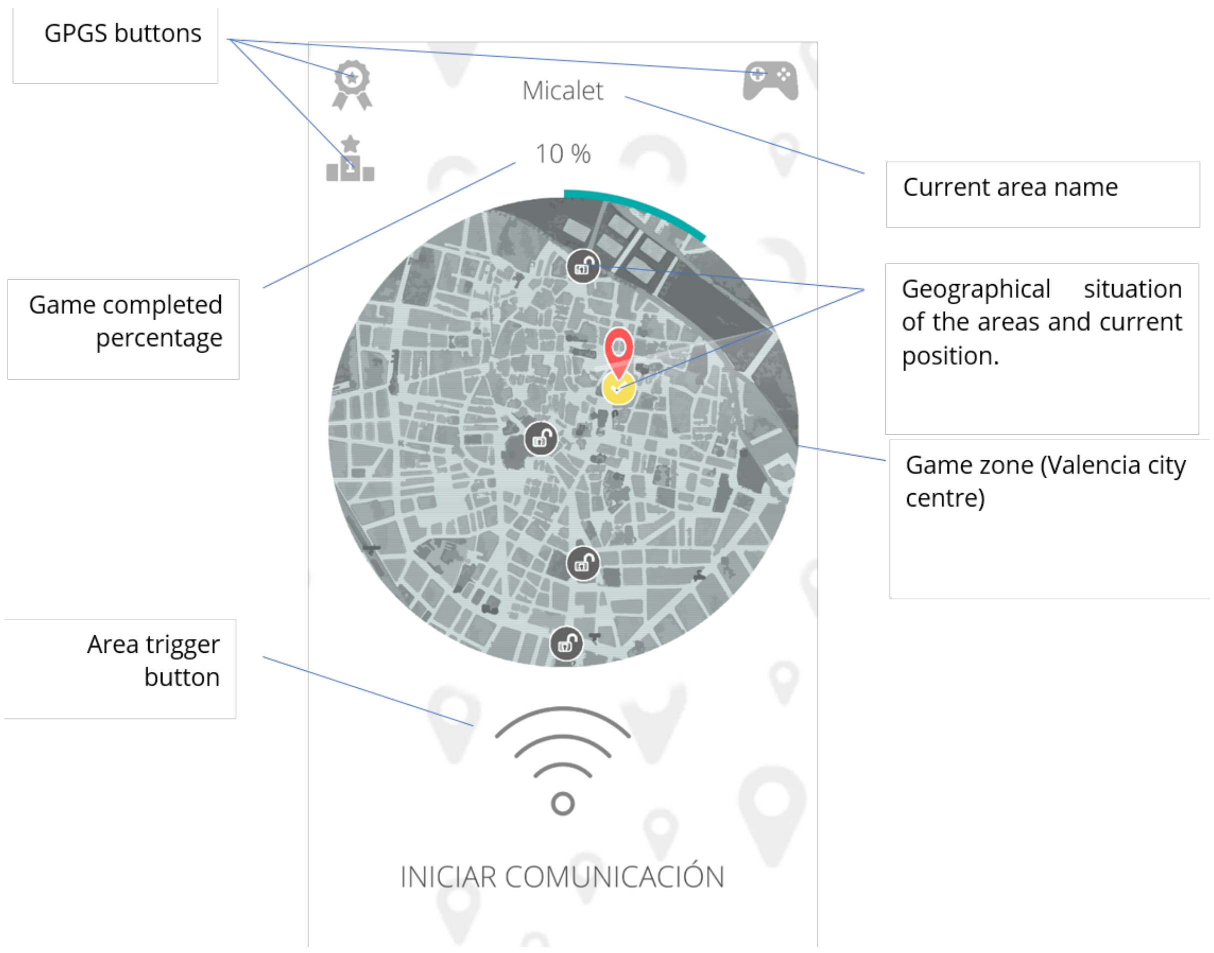

GPGS buttons
Micalet
10 %
Current area name
Game completed percentage
Geographical situation of the areas and current position.
Game zone (Valencia city centre)
Area trigger button
INICIAR COMUNICACIÓN

Figure 55: Default scene User Interface, which includes map, areas location and status, current player location and game completed percentage.

. The background image for the plane was created by taking a street map of the centre of Valencia and editing it in Photoshop using various filters and techniques to eliminate any traces of names of the streets and places and give it a simpler look. The result is a quartered cartoon style image that perfectly delineates streets and buildings without too much detail. In addition to this, a nice circular radar effect was added to the map and the hologram shader was gently applied to give it that subtle futuristic approach. To finish, the circle has a concentric radial slider on the outside that indicates the total percentage of the game completed by the player. When the scene starts, an animation is executed and the slider is filled to the current progress value.

A grated design simulating the technological aspect was also used for the dialog and menus interfaces (Figure 56). A holographic effect was added on the front side, as well as a dark background was applied to improve the readability.

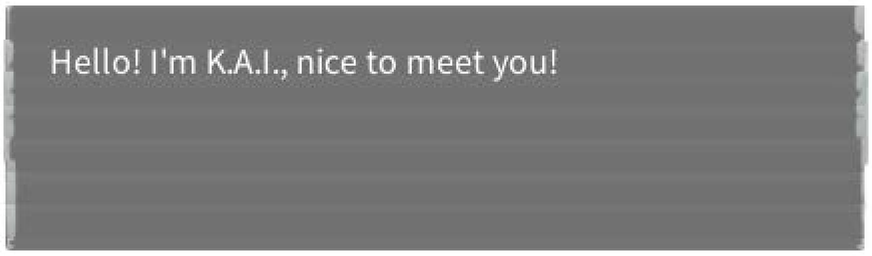

Hello! I'm K.A.I., nice to meet you!

Figure 56: Dialog box.

5.3. 3D modelling

For the modelling of the buildings, the character of K.A.I and other support elements such as the disk has been used exclusively SketchUp Make software. This program offers a series of powerful tools for creating models with a low number of polygons. In addition, it has specialized tools for linear and technical drawing, so it was ideal for designing the basic architecture of the buildings that were intended to be displayed (Figure 57).

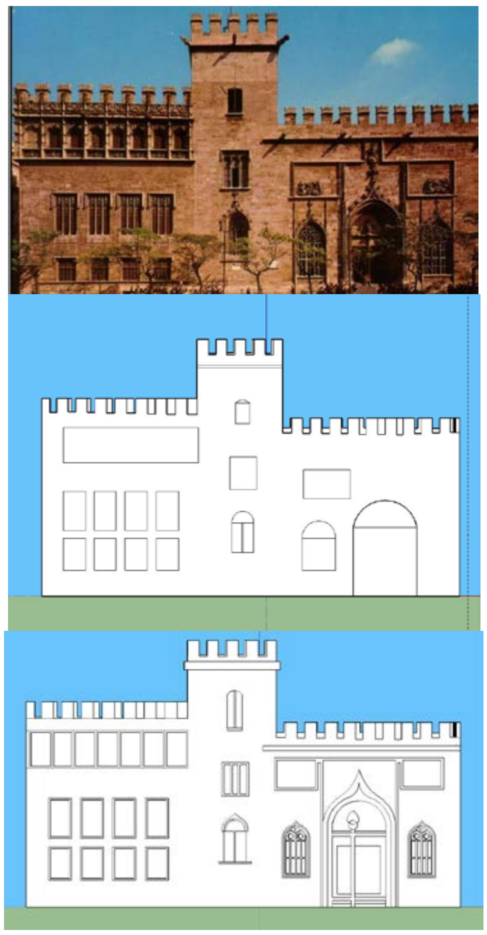


Figure 57: Process of modeling of Lonja de la Seda building.

Apart from this series of advantages, its integration with Google Maps allowed to observe some of the models that are part of the database of the SketchUp community, called 3D Warehouse [49]. Some of the base models were obtained from there and then fixed, detailed and customized depending on the sought characteristics. Once with the vertices, the faces and the profiled details, they could be passed to Unity to integrate the final materials and the corresponding shaders [50]. Below in the Figure 59 all the models of monuments available in the game are shown.

The aesthetic that was being sought was something adjusted to reality but not necessarily exact. That is why some shapes, mouldings and reliefs were exaggerated when modelling them. Thanks to this decision, the final aspect was much more cartoon and fun, and once hologram shader effects and cel shading were added, the edges became much more accentuated and the model became more defined.

In the following Figure 58 it can be seen the details of the mouldings of two monuments. The features and some faces were extruded and exaggerated to give more prominence to the model when it is visualized in augmented reality.

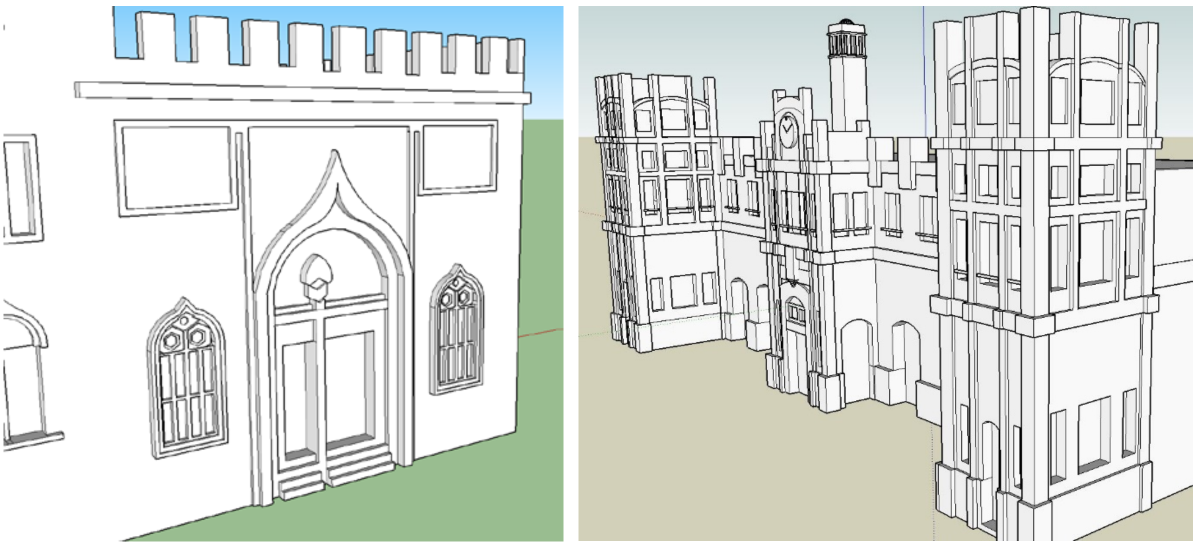


Figure 58: Molding details of the Lonja de la Seda and Estación del Norte models.

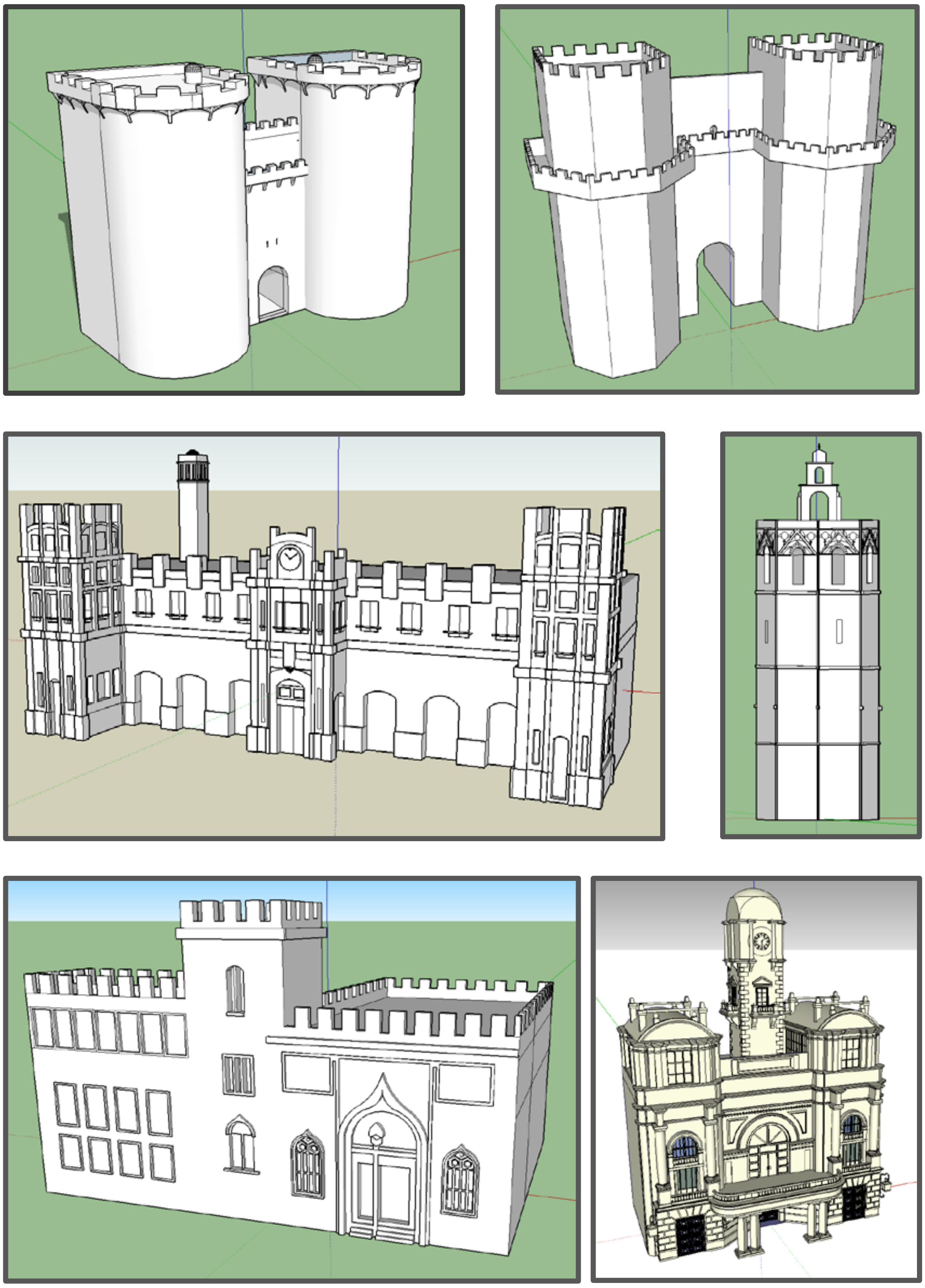
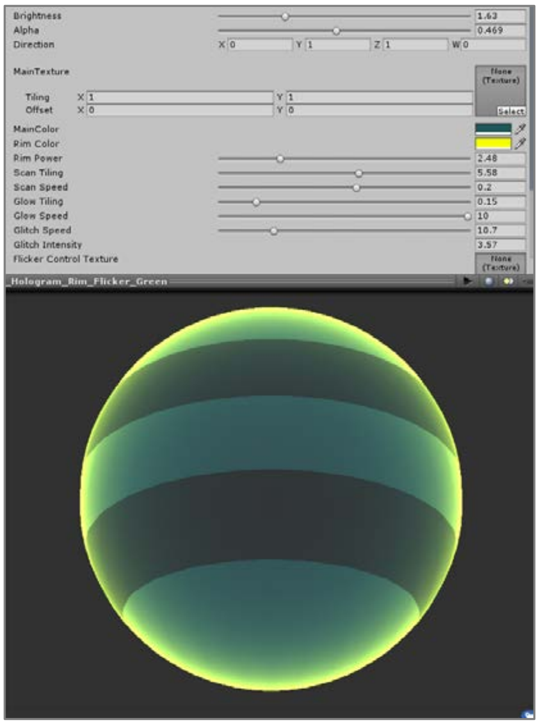


Figure 59: All monument 3D models on SketchUp editor.

5.4. Shaders and additional effects

5.4.1. Hologram shader

As it was desired to give it the most technological aspect possible, one of the visual resources that most helped achieve the effect was the use of a holographic shader. After researching about it, it was fortunate to find an Andy Duboc’s resource protected under a copyrighted MIT license with no restrictions of use [51]. This was the shader that was finally used to produce the holographic effect in the models of the monuments. The objective then was to obtain the highest possible visual quality. Therefore, it was decided avoid using materials or textures beyond those provided by the effects of the shaders. Once the models were clean and with the correct topology, in Unity the shader parameters were adjusted and reinforced with external directional lights (Figure 60).



```
Brightness                 1.63
Alpha                      0.469
Direction         X 0   Y 1   Z 1   W 0

MainTexture                                None (Texture)
  Tiling   X 1              Y 1
  Offset   X 0              Y 0            Select
MainColor
Rim Color
Rim Power                  2.48
Scan Tiling                5.58
Scan Speed                 0.2
Glow Tiling                0.15
Glow Speed                 10
Glitch Speed               10.7
Glitch Intensity           3.57
Flicker Control Texture                    None (Texture)
_Hologram_Rim_Flicker_Green
```

Figure 60: Shader parameters adjustment in the Unity's inspector view.

This shader was inserted in all the models of the buildings. The following Figure 61 shows the result of coupling it with different parameters in the Torres de Quart model.

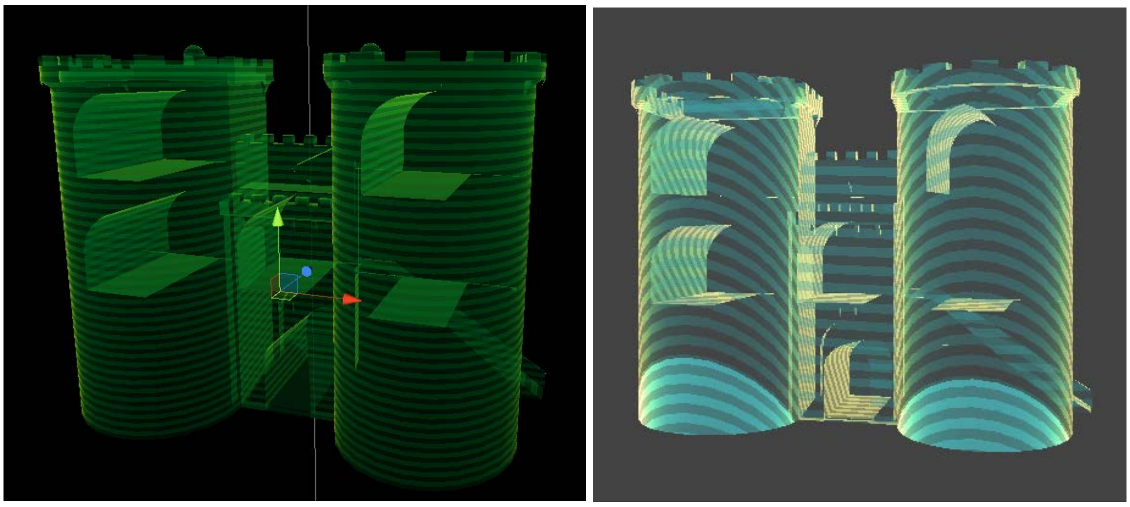


Figure 61: Hologram shader integrated with a building model.

5.4.2. Cel shading effect

The cel shading is a method of rendering images that try to imitate the drawings by hand or sketches with flat shadows, as if it were comics or cartoons. Since the project required a visually striking and pleasing component, it was looked for ways to highlight those objects in three dimensions that were to be displayed in augmented reality. The addition of this effect had a multiple function, since on the one hand it avoided photorealism, it delimited the objects shown in augmented reality with the rest of the environment and allowed to make the models more solid.

This shader has been applied fundamentally to the models of the monuments in combination with the hologram shader, creating a white base layer with the scaled edges that were then covered by the grated effects of the hologram.

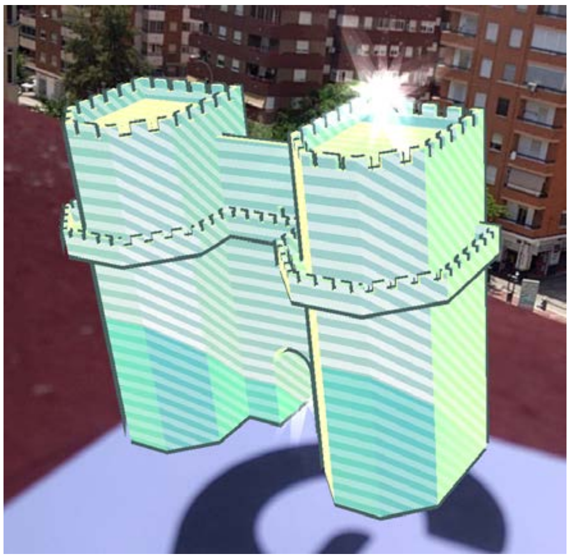


Figure 62: Cel shading effect applied to Torres de Serrano in augmented reality.

In addition, it has also been used in special cases that required a greater distinction of the elements, such as the oranges hung from the memory minigame tree. In that case, other materials and maps of relief and depth were used to give the pseudorealistic effect with flat shadows shown in the Figure 63.

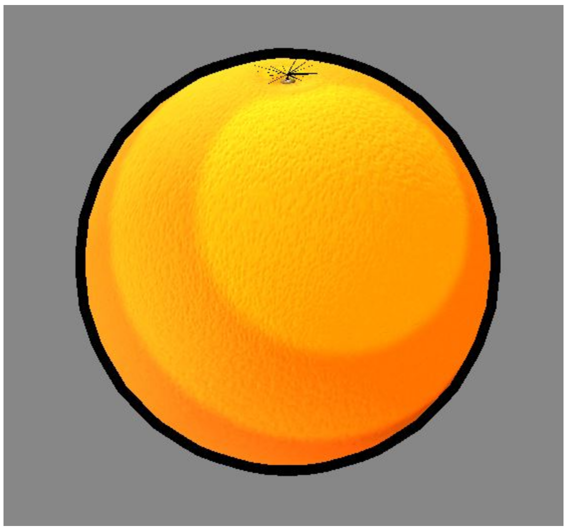


Figure 63: Orange from memory minigame with a cel shading effect applied.

5.4.3. K.A.I.’s design

The power of the transparency of the hologram shader was also used to create another instance of a material and insert it into a curved plane generated to give the retro-screen effect with stripes moving to the K.A.I.’s face. This allows to appreciate that, actually, the final entity of the robot is formed by three layers joined with a separate function each. We find in the frontal part this plane with the material of the shader, between both other plane with a series of associated images that correspond to each one of the gestures that the face of the robot can show and, finally, the model created in Sketch Up of the solid body of the device. The three layers can be seen clearly separated in the Figure 64.

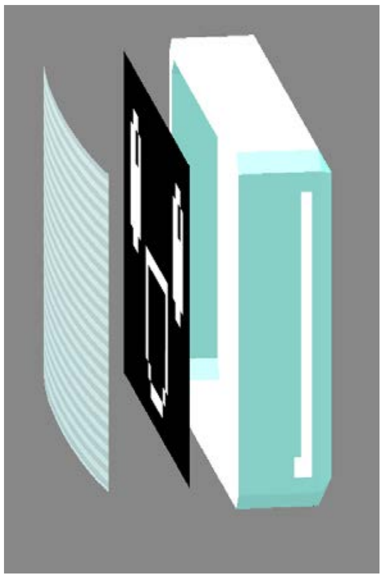


Figure 64: K.A.I. entity is divided in three separated layers.

The model of K.A.I. He suffered various modifications throughout the process. The most important thing is that it was not a model that overloaded the scene, since it was simply a support entity and should not have many details. The futuristic tablet format was created, which, paradoxically, had a floppy disk slot on its left side, in order to load the unlocked information and thus display the hologram that contained the disk. In Figure 65 a little further down you can see the development of the entity of the robot, from its model in Sketch Up to the tests in Unity with the three animated and running layers and the subsequent inclusion of the cel shading effect and lighting.

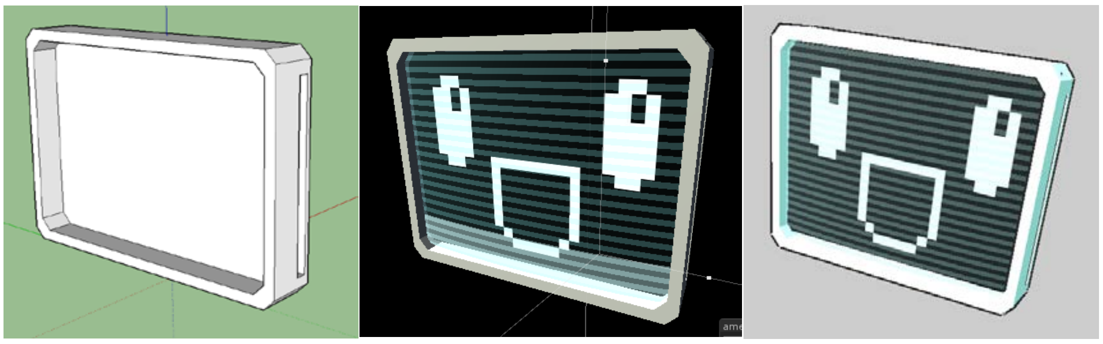


Figure 65: Left to right: Model of K.A.I. in SketchUp, shader hologram and face expressions included, cel shading included and final result.

5.4.4. Logotype

For the main logo design, it was looked for something minimalist that showed the idea of a geolocated application. That is why the classic GPS geolocation pin icon should be incorporated somewhere. The word chosen to represent the app was "KAI", this time without points for mere visual balance. The pin icon was incorporated in the centre of the letter "A" thanks to the use of the figure-background in an original way.

Following the most used design patterns nowadays in a wide range of apps, flat colours were used with a very small range: white for the background and the Persian green used throughout the project for the typography and was drawn an irregular frame surrounding the title, which made the icon less serious and more charismatic. Finally, variations were made adding a slight skybox with the most important monuments of the city in which the application is based, in this case Valencia, with the idea of expanding it in the future to more cities. The final result can be seen below, in the Figure 66.

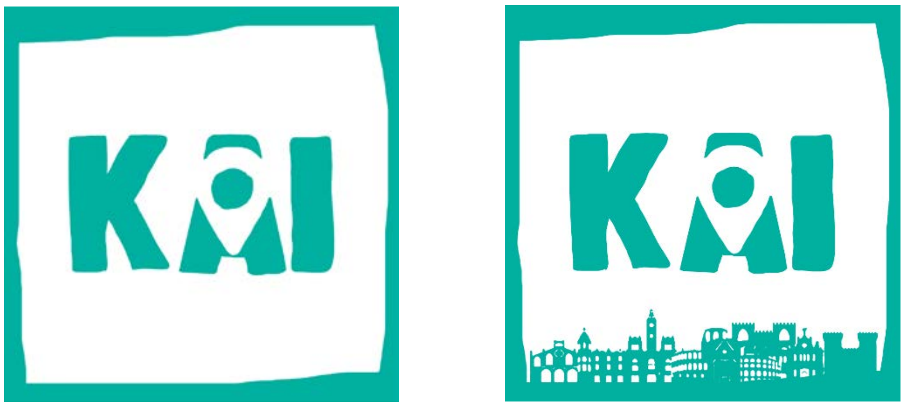


Figure 66: Left: Final logo design. Right: Variation with Valencia's skybox.

6. Results and conclusions

The development of this project has meant a huge personal learning in all aspects. From the beginning, the idea of the game to be developed was very much linked to creating an application that took advantage of the geolocation tools currently offered by mobile devices in a different way. The intention was to create an autonomous and scalable product that explored the interaction with the environment beyond the point of view of a videogame. That is why I decided to develop an interactive application focused on culture and education. Investigating and observing with my own eyes how applications are being used to transmit educational content in a more simple and effective way, I came up with the idea of using the physical environment so that the player could experience in a close way what he was observing. Since my place of residence during the development of the project has been Valencia and, as a lover of travel who likes to see what piece of history the rocky walls of the monuments of the cities have, I thought that there was no better occasion to know the cultural heritage of the city that making a game about it. Also, all that illusion could be shared in the way that I would love to receive it when I travel to a city or town.

Although the project was difficult to focus on in its beginnings, the final result has been satisfactory to a large extent, and a large part of what has been planned has been implemented. Development has been a very iterative process, since the critical vision of the projects themselves always makes us look for different ways to improve what has been done, either internally in terms of the architecture of the code, or at an external level. Raising constantly how the user will interact with the application and what are the strengths and weaknesses of a project is crucial to continue with this process of constant improvement and its consequent learning. This has been, independently of the results, the best feedback that I get from this process. Having faced for the first time such a broad project to develop from start to finish has been a challenge, since planning, research, decisions, implementation times and the level of demand that oneself make on several occasions the pressure play against. All this means a direct learning that, in the last pages of what has been the most extensive project at individual level that I have developed, I appreciate.

Creating a game that sometimes goes off the screen and links the player to the real world has been a great experience, and I sincerely hope that those who have the opportunity to run it on their smartphones and tour the city of Valencia observing every detail of what they have in front of them enjoy it with as much enthusiasm as I had creating it.

7. Further work

Although the objectives and ideas raised at the beginning regarding the total implementation of the project have been met, it is true that there are some application aspects of the application that would be perfect to continue and thus provide new possibilities for the geolocation system.

• One of them, fundamental, is the translation of the project into different languages, since apart from being an educational and informative application, it is conceived as a tourist application. Right now it fulfils its role with local tourism, but I would like to extend it and internationalize its use.

• Exporting the project to other cities beyond Valencia, using a similar system, would be fantastic, since it would allow exploring new areas and greatly expand the gameplay and the community.

• The creation of more different mini-games for each type of monument, as well as the direct relationship of these with what is being visited, seems to me a crucial point to take into account in the future development of the application. This is also linked to the inclusion of more achievements in GPGS to motivate users. Also, once people start testing it, feedback will be evaluated to improve interactivity and make the gaming experience more enjoyable.

• Finally, it would be very interesting if the user could have direct access to the information he has visited from a comfortable interface and without necessarily finding himself in the place. Thus, the key points of buildings and relevant data could be revised (also with augmented reality options to visualize them) as a comparison between the historical moments of their creation. This would be a much more enriching experience at cultural and tourist level.

I hope you have enjoyed reading this memory and that this approach has seemed interesting and inspiring. The borders of videogames have still a diffuse contour that is still to be defined, and, as mentioned at the beginning of this report, experimenting with new technologies to apply them in our society is a responsibility and a privilege that we have as developers nowadays. Maybe in the future digital gamification is much more present in our lives than it is even now. Perhaps the solutions to the most complex problems are found in the simplest programs, so let's never stop experimenting. Let's never stop creating.

8. Bibliography

[1] [Online]. Available: https://en.wikipedia.org/wiki/Smart_city.

[2] P. Milgram, H. Takemura, A. Utsumi and F. Kishino, "" Augmented reality: a class of displays on the reality-virtuality continuum, ",» Telemanipulator and Telepresence Technologies, vol. 2351, p. 282-293, 1995.

[3] «Pokemon GO,» [Online]. Available: http://www.pokemongo.com/

[4] Wikipedia. [Online]. Available: https://es.wikipedia.org/wiki/Pok%C3%A9mon.

[5] U. Technologies. [Online]. Available: https://unity3d.com/es.

[6] «Android,» [Online]. Available: https://www.android.com/

[7] «IOS (Apple),» [Online]. Available: https://www.apple.com/en/ios/ios-11/.

[8] «Windows Phone,» [Online]. Available: https://www.microsoft.com/es-es/store/b/mobile.

[9] United States government, "Global Positioning System," [Online]. Available: https://www.gps.gov/.

[10] M. V. S. Community. [Online]. Available: https://www.visualstudio.com/en/downloads/.

[11] S. Make. [Online]. Available: https://www.sketchup.com/es.

[12] «Google Maps,» [Online]. Available: https://www.google.es/maps

[13] GitHub. [Online]. Available: https://github.com/.

[14] «Adobe Photoshop,» [Online]. Available: https://www.adobe.com/products/photoshopfamily.html.

[15] Trello. [Online]. Available: https://trello.com/.

[16] F. (Google). [Online]. Available: https://firebase.google.com/?hl=es-419

[17] «Firestore,» [Online]. Available: https://firebase.google.com/docs/firestore/?hl=es-419

[18] Angular. [Online]. Available: https://angular.io/.

[19] M. W. 2013. [Online]. Available: https://products.office.com/es-es/word.

[20] «Mendeley,» [Online]. Available: https://www.mendeley.com/.

[21] Ink. [Online]. Available: https://www.inklestudios.com/ink/.

[22] Infinite Ammo Inc . [Online]. Available: https://github.com/InfiniteAmmoInc/Yarn.

[23] Fungus [Online]. Available: http://fungusgames.com/.

[24] «Vuforia (PTC),» [Online]. Available: https://www.vuforia.com/

[25] Vuforia (PTC), «Vuforia features,» 2018. [Online]. Available:

https://www.vuforia.com/features.html.

[26] «AR Kit,» [Online]. Available: https://developer.apple.com/arkit/

[27] «AR Core,» [Online]. Available: https://developers.google.com/ar/discover/

[28] «Wikitude,» [Online]. Available: https://www.wikitude.com/.

[29] «Kudan,» [Online]. Available: https://www.kudan.eu/.

[30] «ARCore Supported devices,» [Online]. Available:

https://developers.google.com/ar/discover/supported-devices

[31] «Ground Plane Supported Devices (Vuforia),» [Online]. Available:

https://library.vuforia.com/articles/Solution/ground-plane-supported-devices.html.

[32] «Google Play Store,» [Online]. Available: https://play.google.com/store/apps?hl=en

[33] «Xcode (Apple),» [Online]. Available: https://developer.apple.com/xcode/.

[34] «App Store,» [Online]. Available: https://www.apple.com/es/ios/app-store/.

[35] «Tourist Guide Valencia,» [Online]. Available: https://www.valencia-tourist-

guide.com/es/general/valencia-espana-historia.html.

[36] Wikipedia, «Geo-Fencing,» [Online]. Available: https://en.wikipedia.org/wiki/Geo-fence.

[37] «Simon Game,» [Online]. Available: http://www.freesimon.org/.

[38] «DoTween - Demigiant,» [Online]. Available: http://dotween.demigiant.com/.

[39] R.-D. Vatavu, L. Anthony and JO Wobbrock, "Gestures as point clouds: a $ P recognizer for user interface prototypes," by ICMI '12 Proceedings of the 14th ACM international conference on Multimodal interaction, Santa Monica, California, USA, October 22 - 26, 2012.

[40] Columbia EE, "Nearest Neighbor Classifiers," [Online]. Available: http://www.ee.columbia.edu/~vittorio/lecture8.pdf.

[41] University of Washington, "1 Unistroke Recognizer," [Online]. Available: http://depts.washington.edu/madlab/proj/dollar/.

[42] AngularFire2 - Github, [Online]. Available: https://github.com/angular/angularfire2.

[43] AngularGoogleMaps - Github, [Online]. Available: https://github.com/SebastianM/angular-google-maps

[44] Unity Community Wiki, «SimpleJSON,» [Online]. Available: http://wiki.unity3d.com/index.php/SimpleJSON.

[45] Google, «Google Play Games Services,» [Online]. Available: https://developers.google.com/games/services/.

[46] «Google Analytics,» [Online]. Available: https://www.google.com/analytics/

[47] «Google Play Console,» [Online]. Available: https://play.google.com/apps/publish/

[48] «GPGS for Unity - GitHub,» [Online]. Available: https://github.com/playgameservices/play-games-plugin-for-unity.

[49] «SketchUp 3D Warehouse,» [Online]. Available: https://3dwarehouse.sketchup.com/

[50] Unity, "Materials, Shaders and Textures," [Online]. Available: https://docs.unity3d.com/current/Manual/Shaders.html.

[51] Andi Duboc's hologram shader., «Github,» [Online]. Available: https://github.com/andydbc/HologramShader.