PLAYFOOD AR

by Carlos Requena Moreno

Final Work Degree

Bachelor's Degree in Video Game Design and Development

Universitat Jaume I

June, 2020

To the fools who dream

Acknowledgments

Thanks to my parents for betting on me all these years.
To my girlfriend for having supported me at all times.
And, of course, my friends and teachers who helped me to improve as a student and as a person. Without all of you, this job would not exist.

Abstract

This document tries to explain a new application focused on catering, powered by the most innovative technologies and accompanied by the idea of using gamification as a form of loyalty through the video games.

INDEX

1.INTRODUCTION

1.1 Work motivation

The motivation that led me to work in this app was the feeling that many times when a client reads the menu of a restaurant and sees an appetizing dish, but he is not convinced because he does not fully understand how does the plate is going to look like , its quantities and what allergens it may contain, so this app end up solving your doubts at a glance.

On the other hand, there is the part of introducing multiplayer video games as a form of using them to create a gamified loyalty system. The decision of its creation arises from the two moments that occur just before and after the meal.

Making them a moment of fun with other diners or even a reason to make the customer come back either because they liked it or because they have obtained some kind of reward that will replace the classic discount coupons.

1.2 Objectives

The objective is therefore the creation of a mobile application that allows users to access the restaurant's menu and see the prices of the dishes, their allergens, their appearance with great fidelity thanks to Augmented Reality and the photogrammetry[1].

And in turn, a section of mini games where one has a system of online ranking incorporating a number of rewards depending on the position and another one multiplayer confronting players directly in real-time, also called PvP (Player vs. Player).

[1]Technique applied in Computer Graphics to study and define precisely the shape, dimensions and position in space of any object, using measurements made from various photographs of that object.

1.3 Environment and initial state

This app is born from an original idea, which after investigating, it has turned out that part of the app, that corresponds to what would be the menu in Augmented Reality already exists (menuar.ru). Despite this, *Playfood AR* presents, among many other differences, that of making the models appear freely and without being restricted to an image and having mini-games that allow the promotion and loyalty of the brand.

The other part of the app, which would be a ranking mini-game, comes up as its own idea based on the coupon system used by *McDonald's* or *Burger King* and an old PlayStation game called "Pepsiman".

The game of "Pepsiman" (see **Fig. 1**) had as a game mechanic to dodge objects from the environment and collect as many *Pepsi* cans (points) as the player can until reaching a Pepsi vending machine. This game was a tool with aggressive advertising marketing intent.

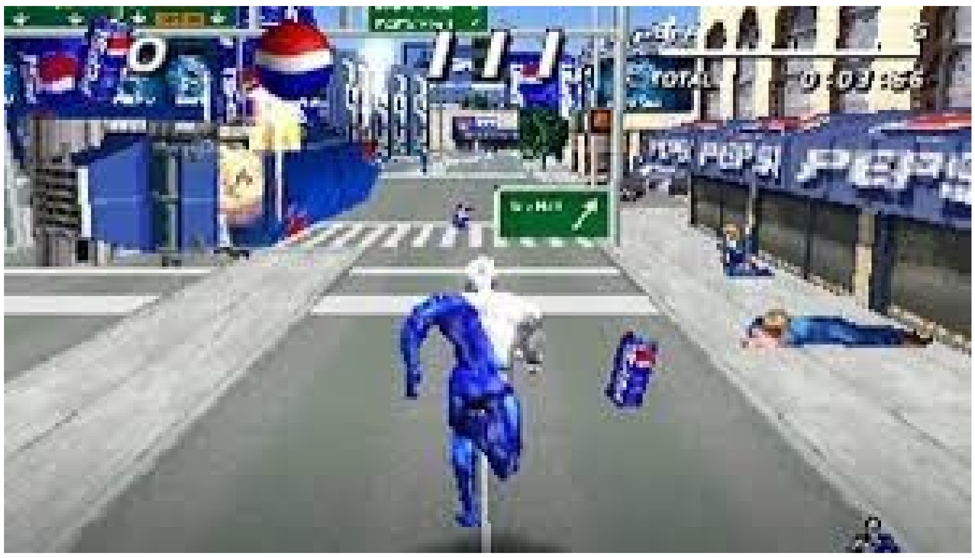


Fig.1. Pepsiman screenshot

Therefore, the idea is to combine a competitive video game with a strong brand image, while trying not to be too intrusive, and with a reward system similar to the coupon one.

So, if for example the app was for *McDonald's*, the mini game with online ranking system, could have a spaceship with the branding colors navigating between hamburguers, fries, etc. and collecting their logo as an example of points.

The final score would be the amount of all the points collected and it would be saved in a highscore table. This highscore would have all the players ranked by their scores and , for example, the top three at the end of a week would have a free menu and the top 20 several discounts in some of their products.

In conclusion, the loyalty system is based on the experience of making people play a funny game, alone or with others, and at the end of certain period of time, making them have some rewards so they come back to the restaurant for eating and having some fun.

2.PLANNING AND RESOURCES EVALUATION

2.1 Planning

The initial planning of the project is divided around the three points that the application initially revolved around and always taking into account the amount of target time of about 235 hours, with some flexibility.

In **Table 1, Table 2 and Table 3** we can see the tasks, subtasks, the estimated time dedicated to each subtask based on the 245 target hours and the actual hours dedicated to each subtask. For easier viewing, a Gantt chart has been made **Fig. 2**

Note that, finally, and for reasons that are explained throughout section 4.2 of this document together with the Gantt chart representing real time expend in the app (**Fig. 38),** the PvP minigame has not been performed.

◦ **MENU CARD IN AUGMENTED REALITY**

Task	Subtask	Time estimation	Real time
Perform photogrammetry	Research	7 hours	7 hours
	Planning with photographer	2 hours	2 hours
	Implementation	5 hours	5 hours
	Adjust	1 hour	2 hours
Perform AR System	Research	-	8 hours
	Implementation	4 hours	10 hours
	Adjust	1 hour	2 hours
Perform DB System	Investigation	2 hours	10 hours
	Planning	1 hour	2 hours
	Implementation	4 hours	20 hours
	Adjust	1 hour	2 hours
Perform HUD	Planning	1 hour	2 hours
	Implementation	5 hours	6 hours
	Adjust	1 hour	2 hours

Table 1. Table of tasks, subtasks and times of the letter.

◦ **MINIGAME TYPE RANKING**

Task	Subtask	Time estimation	Real time
Perform procedural generation system	Research	7 hours	10 hours
	Planning	1 hour	2 hours
	Implementation	10 hours	35 hours
	Adjustment	2 hours	2 hours
Perform the power-ups, HUD and obstacles system	Investigation	3 hours	5 hours
	Planning	1 hour	2 hours
	Implementation	10 hours	20 hours
	Adjustment	2 hours	5 hours
Perform online ranking system	Investigation	7 hours	10 hours
	Planning	1 hour	2 hours
	Implementation	50 hours	50 hours
	Adjust	1 hour	2 hours
Make 3D models	Planning	1 hour	2 hours
	Implementation	3 hours	5 hours
	Adjust	1 hour	3 hours

Table 2. Table of tasks, subtasks and times of the ranking mini-game.

◦ **MULTIPLAYER MINI GAME**

Task	**Time estimation**	**Real time**
Conceptual design of the mini game	5-hour	-
Programming of themultiplayer system	55-hour	-
Programming of the multiplayer mini game	30-hour	-
Creation and implementation of 3D models	10 hours	-

Table 3. Table of tasks and times of the type minigame multiplayer.

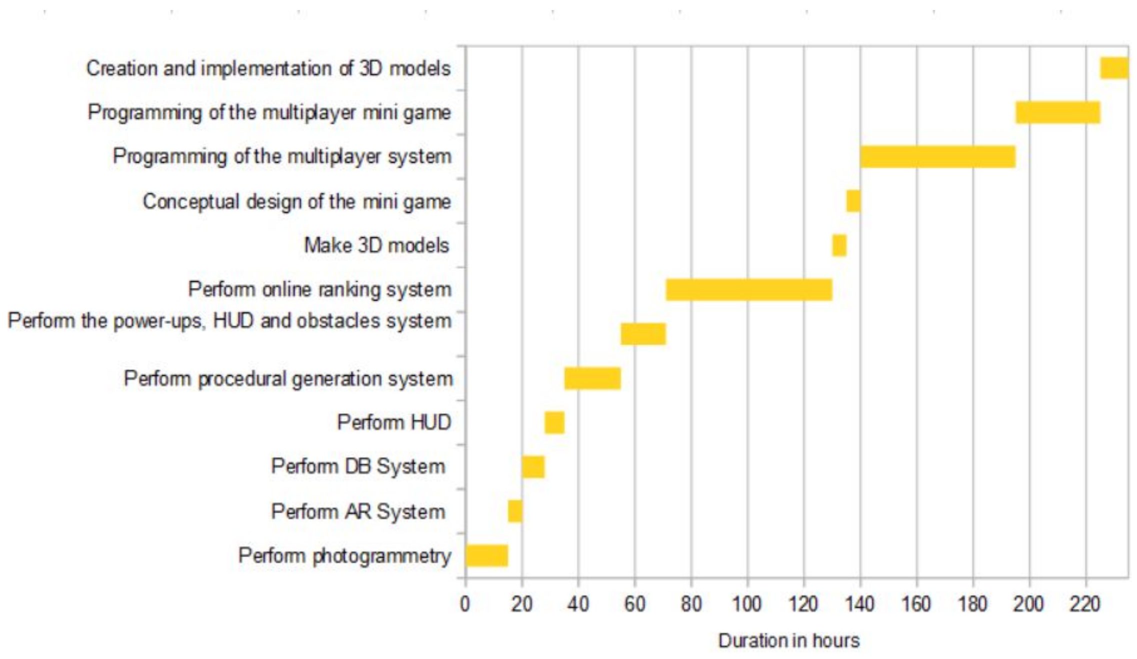


Fig. 2. Gantt chart of estimated time working on the app.

2.2 Resources Evaluation

The equipment needed in technology to create this application (see **Table 4**)

Item	Price
Computer (medium-high range)	900-1200 euros
Smartphone(supporting Arcore)	150 euros
Camera Photo	300 euros
Arcore	Free
Blender	Free
Substance Painter *	127 euros *
Meshroom	Free

Table 4. Table of necessary elements and prices.

Regarding the human team (see **Table 5**)

Professionals	**Salary**
Programmer	1500 euros / month **
Artist	1700 euros / month **
Photographer	11.89 euros / hour **

Table 5. Table of professionals and their salaries

* License perpetual supported by the digital video game platform Steam

** According to https://www.indeed.es/salaries/programador-junior-Salaries

https://www.indeed.es/salaries/dise%C3%B1ador-3d-Salaries

https://www.indeed.es/salaries/fot%C3%B3grafo-Salaries

3.SYSTEM ANALYSIS AND DESIGN

3.1 Requirement analysis

The requirements will be divided into two types: the functional requirements, which refer to all those requirements with data input and output as well as a established behavior. And the non-functional requirements, which cover all the conditions characteristic of the design and implementation of the project.

3.1.1.Functional requirements

All the functional requirements of the application are represented in the following tables (see **Table 6, 7, 8, 9, 10, 11, 12 and 13**).

Input	App section name
Output	Interface and camera
The home screen displays two buttons representing two different sections of the application. One of them is the menu, which when pressed will activate the camera and display the interface with which the user will later interact.	

Table 6. App section name input.

Input	Dish name
Output	Dish model

The menu has different names of dishes. When the user requests to see the model, the rear camera of the mobile device already activated, will allow the user to simply touch the screen again to make the 3D model appear. The 3D model will appear superimposed on a point of the image captured by the rear camera in real time.

Table 7. Dish name input.

Input	Name of the minigame section
Output	Sign in or registration interface
The home screen shows two buttons representing two different sections of the application. One of them is the minigame section, which when pressed will display the interface with which the user will later interact, deciding whether they want to register as a new user or if they have already registered to sign in.	

Table 8. Name of the minigame section input.

Input	Username and password
Output	Data in the database
The data inserted in the fields of the registration interface, corresponding to the username and password, will be registered in an SQL database.	

Table 9. Username and password input.

Input	Username and password
Output	Minigame
The data inserted in the fields of the access interface, corresponding to the username and password, will be verified to be in the SQL database and if so, the minigame will be accessed.	

Table 10. Username and password input.

Input	Touch screen control
Output	Character movement
Sliding your finger across the touch screen from side to side will allow the character to move according to your finger position.	

Table 11. Touch screen control input.

Input	Character
Output	Power-up, end of the game and score
In case the character collides with power-ups, you will get one of these by updating the interface and being able to use it by tapping on the screen. However, if the character collides with other elements of the environment it will be the end of the game and will show the score of that game.	

Table 12. Character input.

Input	Score
Output High	Tables
When the game ends, if the score obtained by the player is higher than the one previously obtained, the score will be updated in the database. Finally, a table with the highest scores will be shown	

Table 13. Score input.

3.1.2.Non-functional requirements

The project is guided by the following characteristics:

▪ Efficiency in terms of interaction and visualization, being possible to use in real time with a frame rate minimum of 24 fps, even with the rear camera active.

▪ Ease and simplicity, allowing the user to understand the interface of the application quickly.

▪ Usability, allowing the user to interact with the application with just one hand.

3.2 System design

The system design of the Menu AR is represented in **Fig.3**.

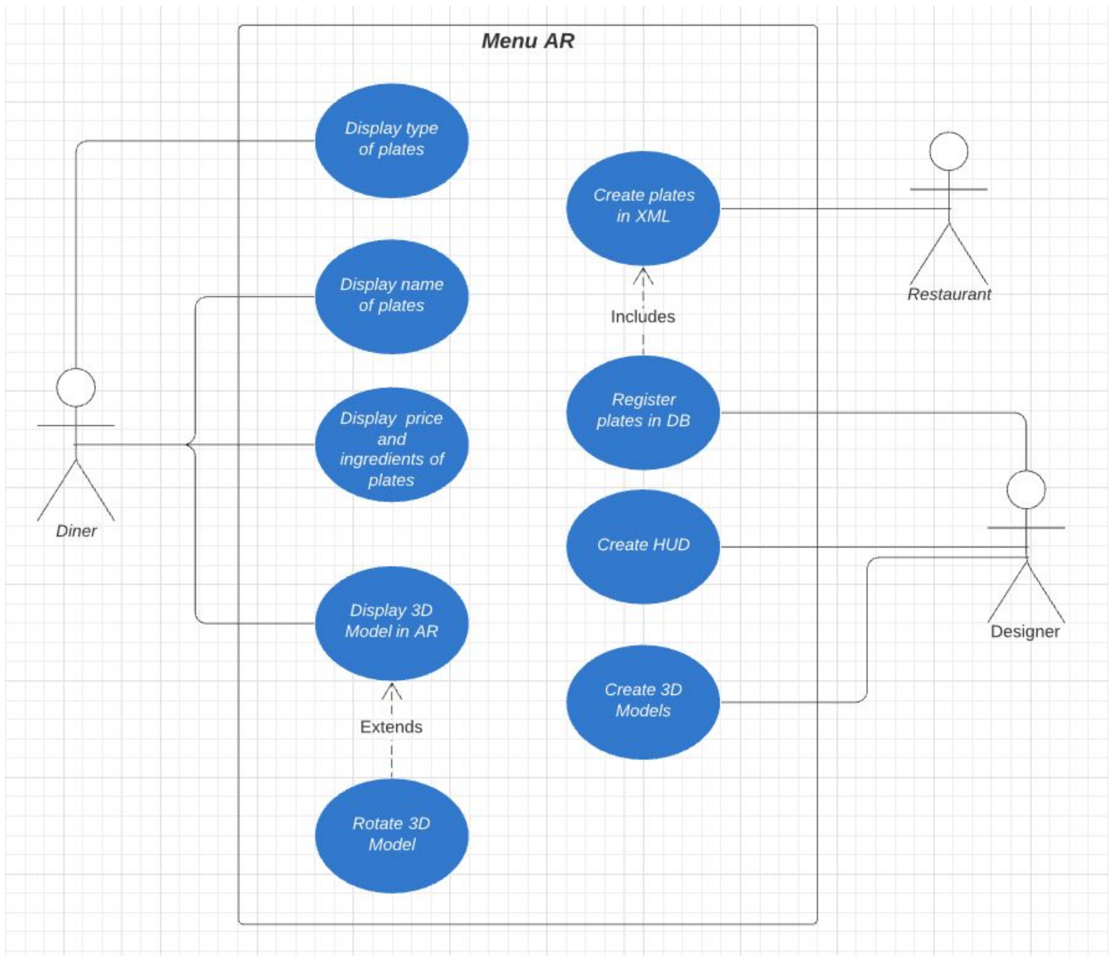


Fig. 3 . Case use diagram of Menu AR

3.3 System architecture

The system architecture must be primarily suitable for being able to use ***ARCore***[2] **(see 1.12).**

That is why the mobile device must have passed *Google*'s strict quality control, in regards to hardware. In it the camera, the motion sensors and the architecture design, in which a powerful CPU is of vital importance to be able to carry out the calculations in real time in an effective and precise way.

Regarding the software, it is important that it was originally launched with the *Google Play Store*, at least with version 7.0 of the *Android* operating system and support with *OpenGL ES* 3.0.

That is for the Chinese market there is an exception, since despite the fact that the *Google Play Store* is not launching, it can be purchased through the device's brand app.

And in case the device has an *iOS* operating system, *ARCore* supports all mobile devices that are compatible with *ARKit*[3] and that have version 11.0 or higher of the operating system.

For testing, an Oppo Realme X2 mobile device has been used, with the *Google Play Store* launch, the *Android* operating system in its version 9.0, a *Qualcomm* SDM730G AIE eight-core processor and 8 GB of RAM.

[2]It is the platform created by Google to create augmented reality experiences.

[3]It is the platform created by Apple to create augmented reality experiences.

3.4 Interface design

All the Playfood AR interface is designed to be used with the mobile device in vertical position and controlled with one hand. The colors of the interface in this case are blue (see in **Fig. 4**), but in case the app was for a restaurant this would be changed.

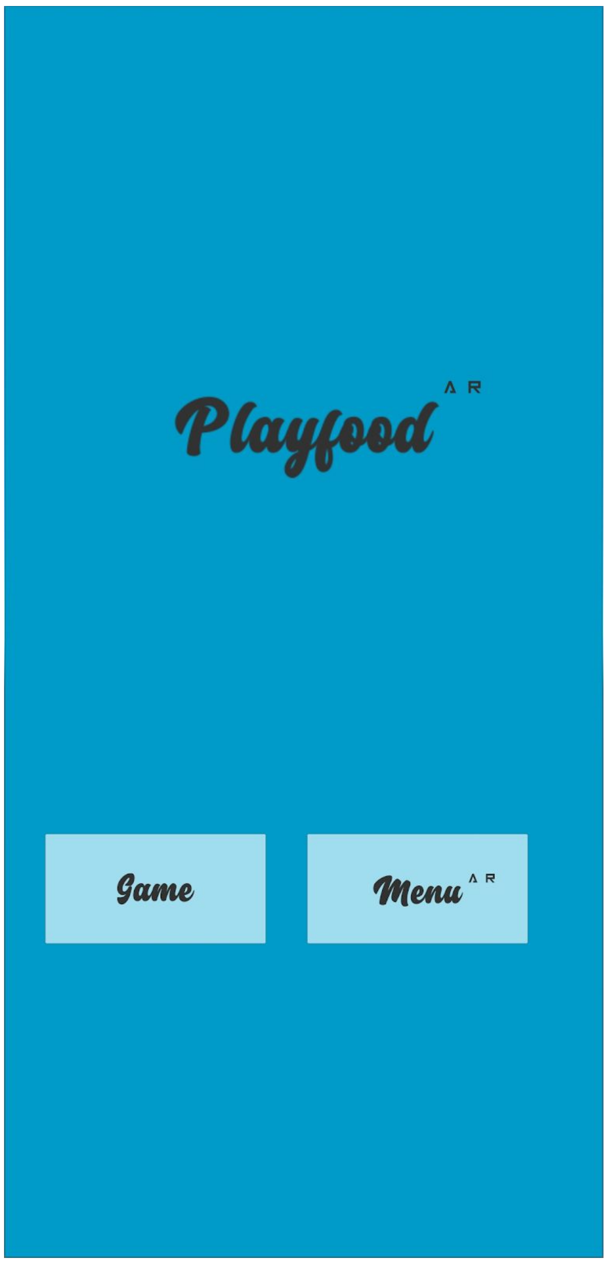


Fig.4. First Screen screenshot

4.WORK DEVELOPMENT AND RESULTS

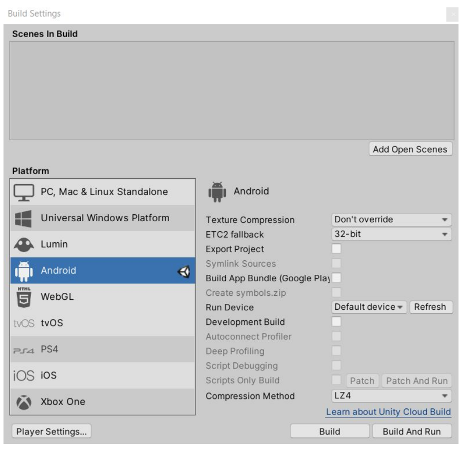
4.1 Development Work

As the application is divided in two parts: menu card and minigames, this chapter will be divided in two.

◦ **Menu card in Augmented Reality**

Since one of the main objectives is the use of augmented reality, the first thing was to try to implement augmented reality through *Vuforia*[4] [1]and the use of a reference image that can have any aspect, allowing the system to recognize a fast surface on which to display the 3D model in augmented reality.

To do this, an initial project was created in a unity with a default 3D scene.



```
Android
Texture Compression          Don't override
ETC2 fallback                32-bit
Export Project               ☐
Symlink Sources              ☐
Build App Bundle (Google Play) ☐
Create symbols.zip           ☐
Run Device                   Default device
Development Build            ☐
Autoconnect Profiler         ☐
Deep Profiling               ☐
Script Debugging             ☐
Scripts Only Build           ☐
Compression Method           LZ4
```

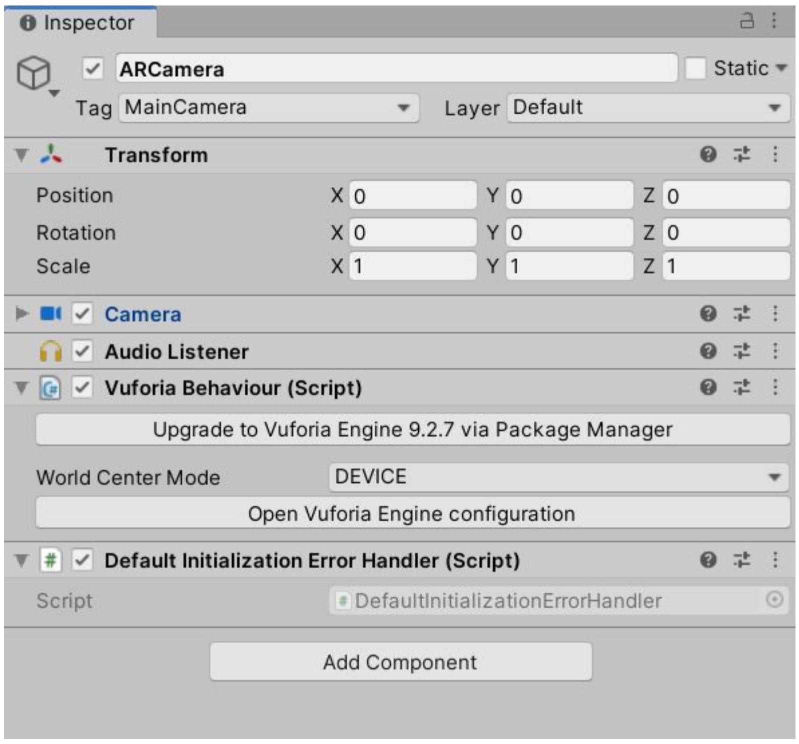
Later the project was configured to be able to run it on Android and make Vuforia work in the project (see **Fig. 5**).

Fig. 5. Build settings of the project configured for Android

[4]It is an augmented reality software development kit for mobile devices that allows the

creation of augmented reality applications.

The initial camera was then removed from the scene and replaced with the *Vuforia AR camera* **(see Fig. 6)**.


Inspector

ARCamera — Static

Tag: MainCamera — Layer: Default

Transform

	X	Y	Z
Position	0	0	0
Rotation	0	0	0
Scale	1	1	1

Camera

Audio Listener

Vuforia Behaviour (Script)

Upgrade to Vuforia Engine 9.2.7 via Package Manager

World Center Mode: DEVICE

Open Vuforia Engine configuration

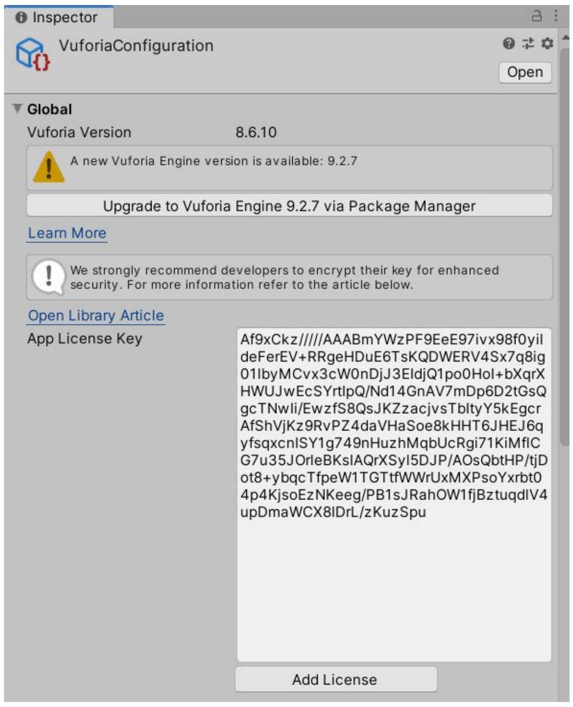
Default Initialization Error Handler (Script)

Script: DefaultInitializationErrorHandler

Add Component

Fig. 6. Vuforia AR camera components.

AR camera properties, more specifically *Vuforia* engine configuration, were accessed and a license key was inserted (see **Fig.7**).



Vuforia Version 8.6.10

A new Vuforia Engine version is available: 9.2.7

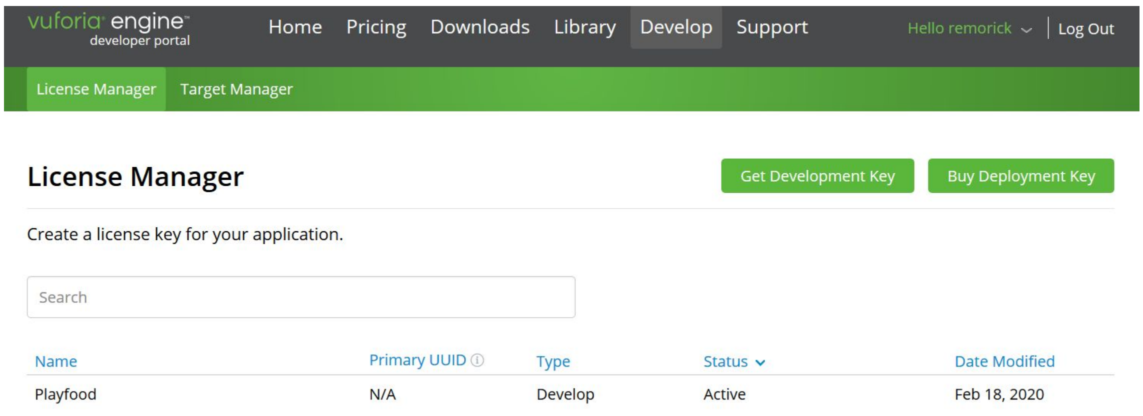
We strongly recommend developers to encrypt their key for enhanced security. For more information refer to the article below.

App License Key

```
Af9xCkz/////AAABmYWzPF9EeE97ivx98f0yil
deFerEV+RRgeHDuE6TsKQDWERV4Sx7q8ig
01IbyMCvx3cW0nDjJ3EIdjQ1po0HoI+bXqrX
HWUJwEcSYrtIpQ/Nd14GnAV7mDp6D2tGsQ
gcTNwIi/EwzfS8QsJKZzacjvsTbItyY5kEgcr
AfShVjKz9RvPZ4daVHaSoe8kHHT6JHEJ6q
yfsqxcnISY1g749nHuzhMqbUcRgi71KiMfIC
G7u35JOrIeBKsIAQrXSyI5DJP/AOsQbtHP/tjD
ot8+ybqcTfpeW1TGTtfWWrUxMXPsoYxrbt0
4p4KjsoEzNKeeg/PB1sJRahOW1fjBztuqdIV4
upDmaWCX8IDrL/zKuzSpu
```

Fig. 7. License Key inserted in Unity Project

In order to obtain it, the access to the website (was accessed https://developer.vuforia.com)and a company registration was made in order to obtain a free license key(see **Fig. 8 and 9**).

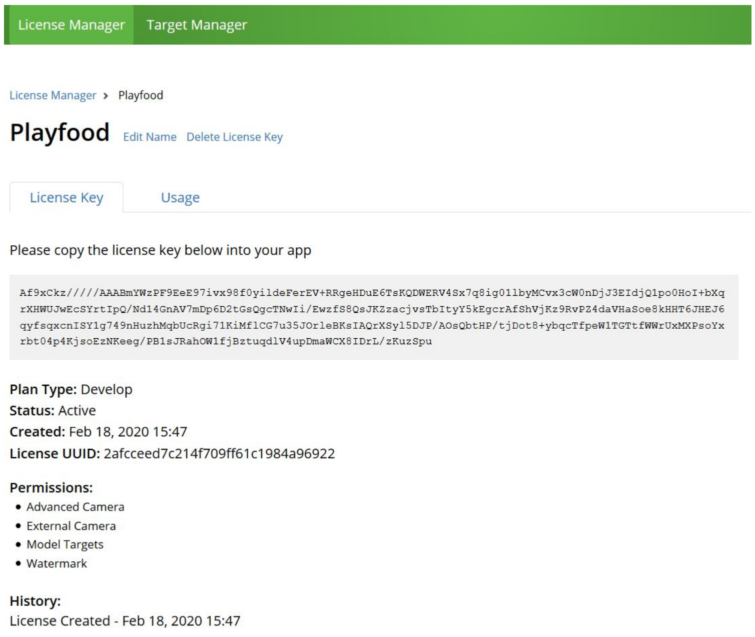


License Manager

Create a license key for your application.

Name	Primary UUID	Type	Status	Date Modified
Playfood	N/A	Develop	Active	Feb 18, 2020

Fig. 8. Project created in the developers website of Vuforia.



Playfood

Please copy the license key below into your app

```
Af9xCkz/////AAABmYWzPF9EeE97ivx98f0yildeFerEV+RRgeHDuE6TsKQDWERV4Sx7q8ig01lbyMCvx3cW0nDjJ3EIdjQ1po0HoI+bXq
rXHWUJwEcSYrtIpQ/Nd14GnAV7mDp6D2tGsQgcTNwIi/EwzfS8QsJKZzacjvsTbItyY5kEgcrAfShVjKz9RvPZ4daVHaSoe8kHHT6JHEJ6
qyfsqxcnISY1g749nHuzhMqbUcRgi71KiMflCG7u35JOrleBKsIAQrXSyl5DJP/AOsQbtHP/tjDot8+ybqcTfpeW1TGTtfWWrUxMXPsoYx
rbt04p4KjsoEzNKeeg/PB1sJRahOW1fjBztuqdlV4upDmaWCX8IDrL/zKuzSpu
```

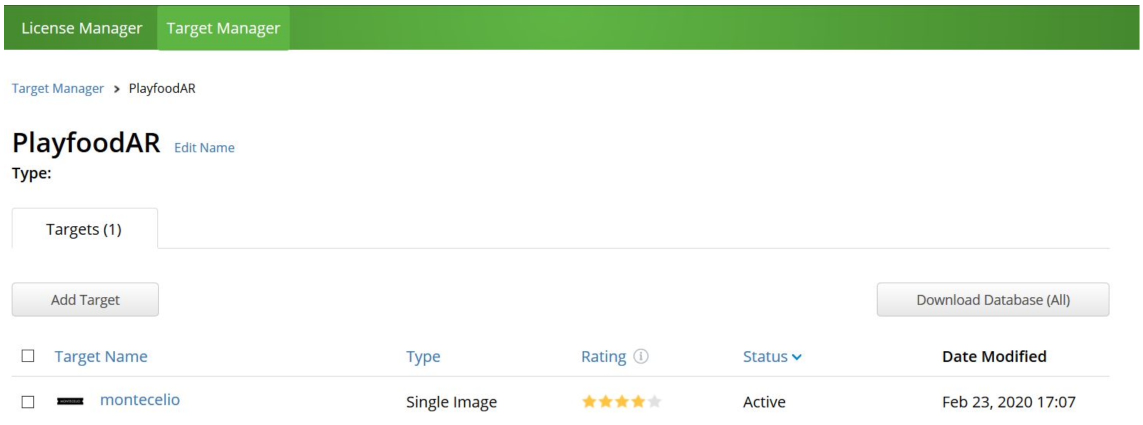
Plan Type: Develop
Status: Active
Created: Feb 18, 2020 15:47
License UUID: 2afcceed7c214f709ff61c1984a96922

Permissions:
- Advanced Camera
- External Camera
- Model Targets
- Watermark

History:
License Created - Feb 18, 2020 15:47

Fig. 9. License key obtained for the project

Through the same website, a database was created for the project's reference images (see **Fig. 10**), which were downloaded(see **Fig. 11**) as a unity package and imported into the project.

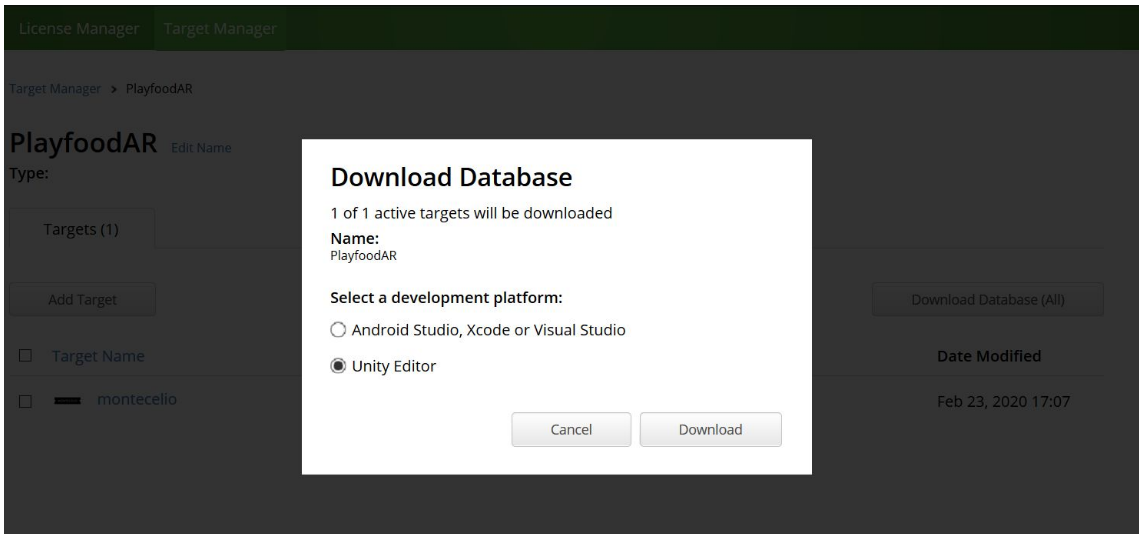


PlayfoodAR

Type:

☐	Target Name	Type	Rating	Status	Date Modified
☐	montecelio	Single Image	★★★★☆	Active	Feb 23, 2020 17:07

Fig. 10. Database of targets of the project.


License Manager | Target Manager

Target Manager > PlayfoodAR

PlayfoodAR Edit Name

Type:

Targets (1)

Add Target | Download Database (All)

	Target Name	Date Modified
☐	montecelio	Feb 23, 2020 17:07

Download Database

1 of 1 active targets will be downloaded

Name:
PlayfoodAR

Select a development platform:

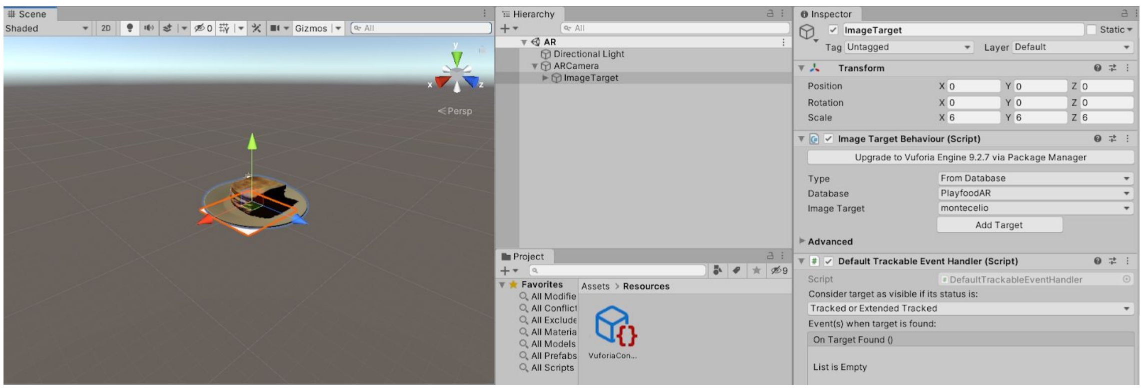
○ Android Studio, Xcode or Visual Studio

● Unity Editor

Cancel | Download

Fig. 11. Database download for Unity

Finally, the imported reference image was included as Image Target of the scene, a free 3D model was downloaded for testing and inserted into the scene in the same position as the image and hierarchically below in Unity (see **Fig. 12**).



Hierarchy

- AR
  - Directional Light
  - ARCamera
    - ImageTarget

Inspector

ImageTarget ☑ | Static ☐

Tag Untagged | Layer Default

Transform

Position	X 0	Y 0	Z 0
Rotation	X 0	Y 0	Z 0
Scale	X 6	Y 6	Z 6

Image Target Behaviour (Script)

Upgrade to Vuforia Engine 9.2.7 via Package Manager

Type	From Database
Database	PlayfoodAR
Image Target	montecelio

Add Target

Advanced

Default Trackable Event Handler (Script)

Script DefaultTrackableEventHandler

Consider target as visible if its status is:

Tracked or Extended Tracked

Event(s) when target is found:

On Target Found ()

List is Empty

Fig. 12. Scene result in Unity

But after implementing it, it generated problems when displaying the 3D model, because of the light reflections on the reference image cause recognition failures. On the other hand, needing a reference image seemed impractical and elementary. Therefore, a more elegant and powerful solution was sought thanks to ***ARCore (see Fig. 13)***.

Fig. 13. *ARCore* Logo

And it is that *ARCore*[2] has very interesting capabilities such as: Motion Tracking, Environmental Understanding, Light Estimation, User interaction, Oriented Points, Anchors and Trackables, Augmented Images and Sharing.

➢ MOTION TRACKING

As a mobile phone moves around the world, *ARCore* uses a process called, simultaneous location and mapping, or SLAM (), to understand where the phone is relative to the world around it. *ARCore* visually detects these points to compute their location changes. Visual information combined with inertial measurements from the IMU (Inertial Measuring Unit), a device that measures and reports on the speed, orientation, and gravitational forces of a device, using a combination of accelerometers and gyroscopes of the device to estimate the position and camera orientation relative to the world over time.

By aligning the position and orientation of the virtual camera that renders 3D content with the position and orientation of the device camera provided by *ARCore*, developers can render virtual content from the correct perspective. This allows the rendered virtual image to be superimposed on the image taken by the device's camera making it appear that the virtual content is part of the real world.

➢ ENVIRONMENTAL UNDERSTANDING

ARCore is constantly improving its understanding of the real-world environment by detecting feature points and planes.

ARCore searches for groups of characteristic points that appear to be on common horizontal surfaces or vertical surfaces, such as tables or walls, making these surfaces available to the application as planes. *ARCore* can also determine the plane boundaries and provide that information to the app. This allows this information to be used to place virtual objects on flat surfaces.

The downside is that because *ARCore* uses feature points to detect planes, flat and un textured surfaces, how a white wall might not be detected correctly.

➢ LIGHT ESTIMATION

ARCore can detect information about the lighting in your environment and provide the average intensity and color correction of a certain camera image. This information allows the virtual objects to be illuminated in the same conditions as the surrounding environment, increasing the feeling of realism.

➢ USER INTERACTION

ARCore uses the shock test to take a coordinate (x, y) corresponding to the phone screen (provided by a touch or any other interaction the application wants to support) and projects a ray into the world view of the camera, returning any plane or characteristic points that intersect with it, along with the position and orientation of the intersection with world space. This allows users to select or interact with objects in the environment.

➢ ORIENTED POINTS

Oriented points allow virtual objects to be placed on angled surfaces. When you perform an impact test that returns a characteristic point, *ARCore* will search for nearby characteristic points and use them to try to estimate the angle of the surface belonging to the given characteristic point.

Finally *ARCore* will return the position and orientation also taking into account the angle obtained.

That is, because *ARCore* uses groups of characteristic points to detect the angle of the surface, it is possible that surfaces without texture, present detection problems as it happens in understanding the environment.

➢ ANCHORS AND TRACKABLES

Position and orientation may change as *ARCore* improves understanding of your own position and that of your surroundings. When you want to place a virtual object, you must define an anchor to make sure that *ARCore* tracks the position of the object over time. Many times an anchor is created based on the position and orientation returned by an impact test, as described in the user interaction section.

The fact that position and orientation can change means that ARCore must update the position of environmental objects such as planes or characteristic points over time. Planes and points are a special type of object called trackable. As the name suggests, these are objects that *ARCore* will track over time. Therefore, you can pin virtual objects to specific trackables to ensure that the relationship between your virtual object and the trackable remains stable even when the mobile device is moved. This means that if a virtual model is placed on, for example, a table, if *ARCore* then adjusts the position and orientation of the plane associated with the table, the virtual model will continue to appear on the table.

➢ AUGMENTED IMAGES

Augmented imagery is a feature that enables you to create AR applications that can respond to specific 2D images, such as product packaging or movie posters. Allowing you to trigger AR experiences when you point your phone's camera at specific images; for example, pointing your phone's camera at a movie poster and having a movie trailer play.

ARCore also tracks moving images, such as a moving bus billboard.

Images can be compiled to create a reference image database, as with *Vuforia*, or even add individual images in real time from the device. Once registered, *ARCore* will detect these images, the limits of the images and return a corresponding position and orientation.

➢ SHARING

The *ARCore* Cloud Anchor API allows you to create collaborative or multiplayer applications for Android and iOS devices.

With Cloud Anchors, a device sends an anchor and feature points close to the cloud to virtually host them. These anchors can be shared with other users on Android or iOS devices in the same environment. Allowing applications to represent the same 3D objects attached to these anchors, making users able to have the same AR experience simultaneously.

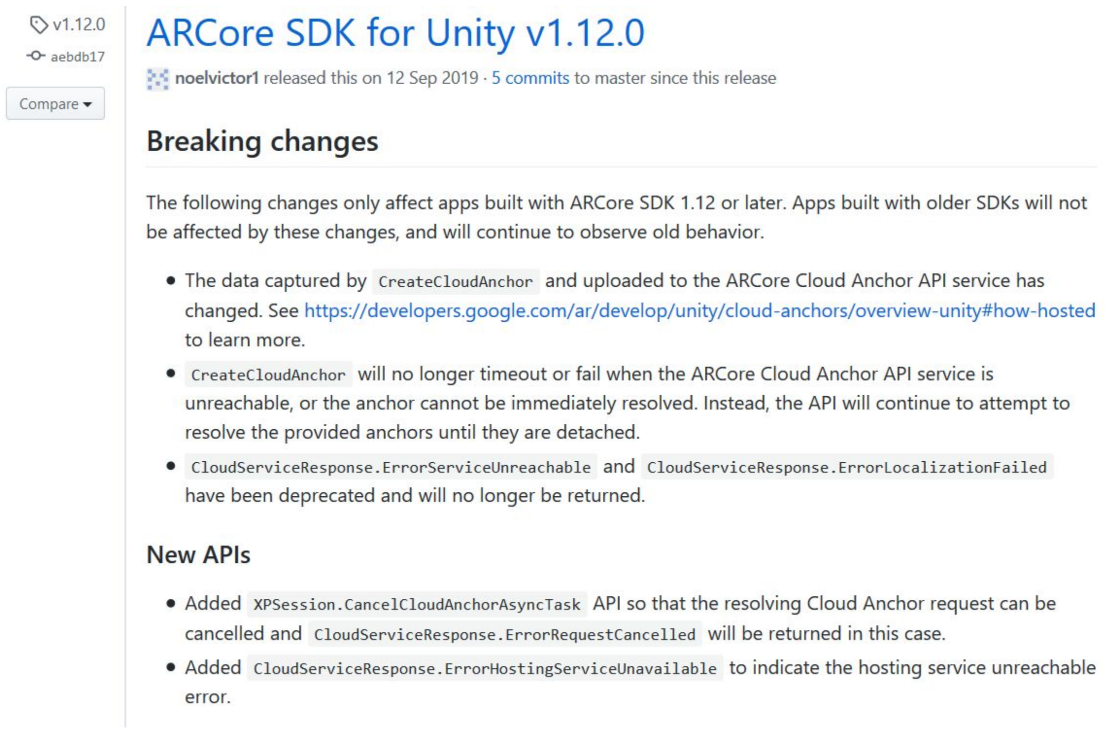
This is more than interesting functionality and that for the future development of the application allows the creation of mini-games.

All these capabilities, a small introduction for developers and the existence of an SDK * for the Unity video game engine, make *ARCore*[3] the best option to develop the application.

---

*An SDK is a collection of software used for developing applications for a specific device or operating system

This is why the 2018.3.5 version of *Unity* was first downloaded, a new project was installed and created. Subsequently, the *ARCore* SDK[4] for Unity version 1.12.0 was downloaded and all the content was imported into the previously created project. The project settings were changed so that it could run on the Android operating system and the *ARCore* functionalities (see **Fig. 14**) could be carried out.



v1.12.0
aebdb17
Compare

ARCore SDK for Unity v1.12.0

noelvictor1 released this on 12 Sep 2019 · 5 commits to master since this release

Breaking changes

The following changes only affect apps built with ARCore SDK 1.12 or later. Apps built with older SDKs will not be affected by these changes, and will continue to observe old behavior.

- The data captured by `CreateCloudAnchor` and uploaded to the ARCore Cloud Anchor API service has changed. See https://developers.google.com/ar/develop/unity/cloud-anchors/overview-unity#how-hosted to learn more.
- `CreateCloudAnchor` will no longer timeout or fail when the ARCore Cloud Anchor API service is unreachable, or the anchor cannot be immediately resolved. Instead, the API will continue to attempt to resolve the provided anchors until they are detached.
- `CloudServiceResponse.ErrorServiceUnreachable` and `CloudServiceResponse.ErrorLocalizationFailed` have been deprecated and will no longer be returned.

New APIs

- Added `XPSession.CancelCloudAnchorAsyncTask` API so that the resolving Cloud Anchor request can be cancelled and `CloudServiceResponse.ErrorRequestCancelled` will be returned in this case.
- Added `CloudServiceResponse.ErrorHostingServiceUnavailable` to indicate the hosting service unreachable error.

Fig. 14. *ARCore* SDK 1.12 for *Unity* features and changes.

From here, all the components belonging to an example scene from the SDK, called *HelloAR*, were added to a new scene(see **Fig.15**).

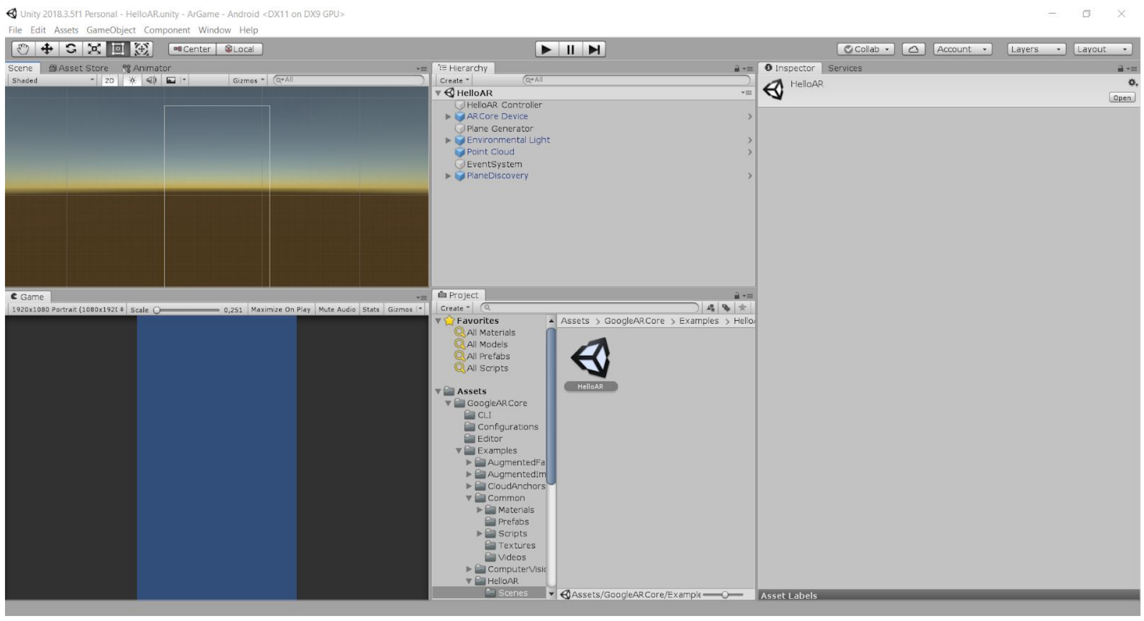
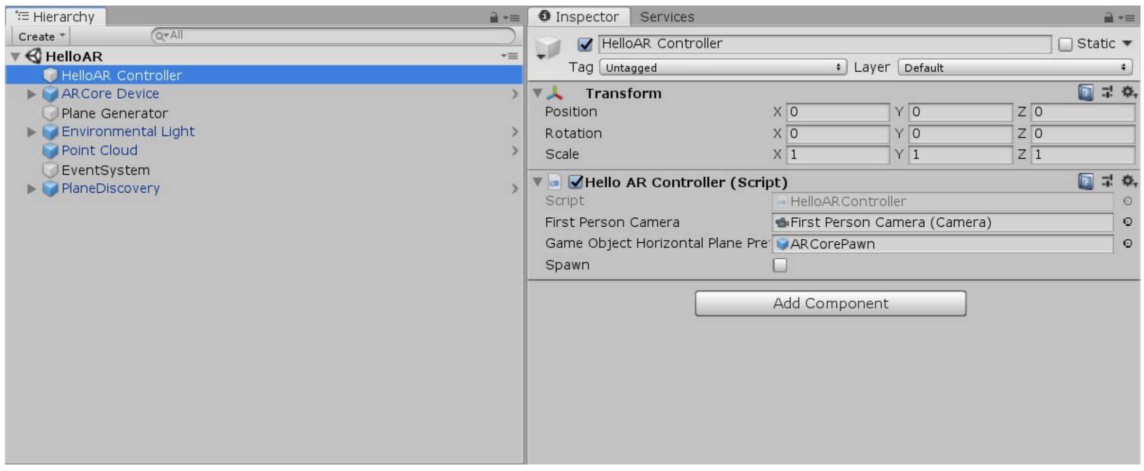


Fig. 15. *HelloAR* Scene

Among them is the *HelloAR Controller* (see **Fig.16**), a basic script to be able to interact with reality, what it does is make reference to the camera and prefabs that will be shown if the plane is detected and clicked on it.


Hierarchy

- HelloAR
  - HelloAR Controller
  - ARCore Device
  - Plane Generator
  - Environmental Light
  - Point Cloud
  - EventSystem
  - PlaneDiscovery

Inspector — Services

HelloAR Controller — Static
Tag: Untagged — Layer: Default

Transform

	X	Y	Z
Position	0	0	0
Rotation	0	0	0
Scale	1	1	1

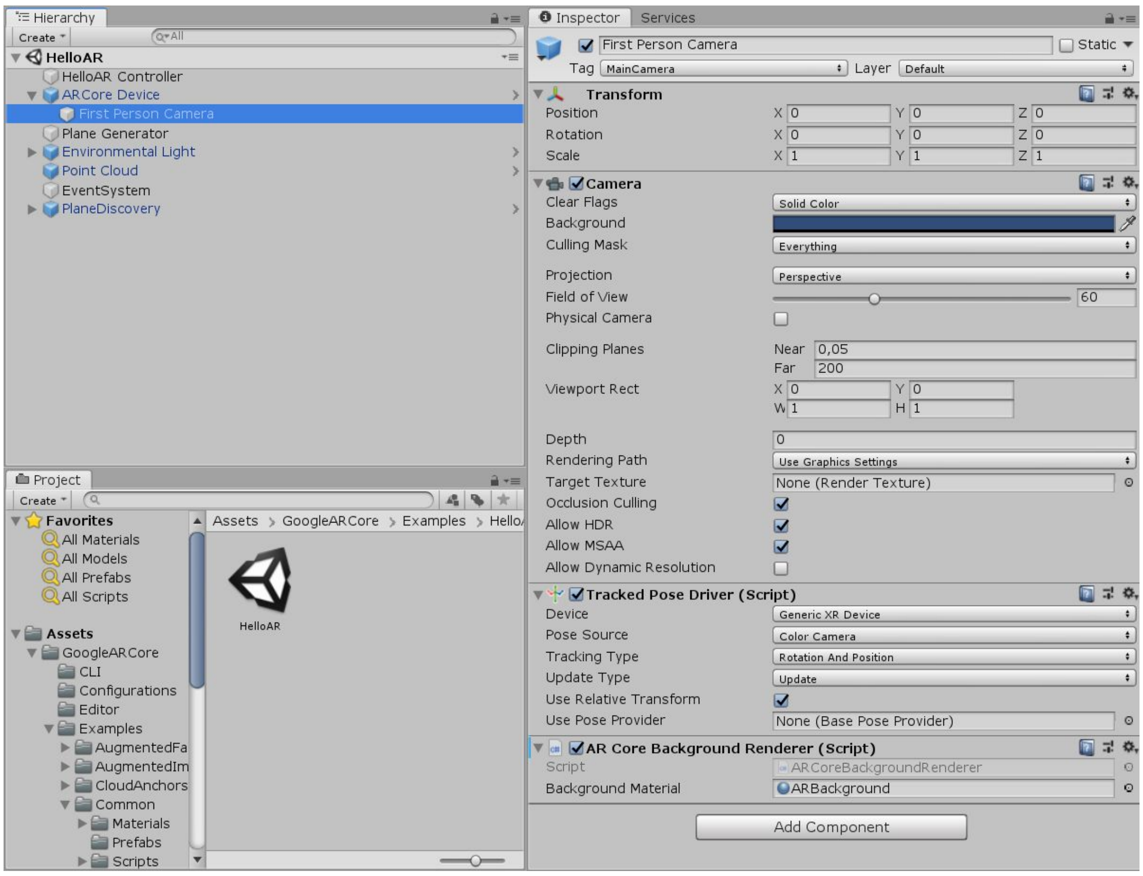
Hello AR Controller (Script)

Script	HelloARController
First Person Camera	First Person Camera (Camera)
Game Object Horizontal Plane Pre	ARCorePawn
Spawn	☐

Add Component

Fig.16. *Hello AR Controller* components

The *ARCore Device* (see **Fig. 17**)that presents a script that allows us to decide which camera will be used. Inside the *ARCore Device* is the camera that presents a script called *Tracked Pose Driver* that will allow to follow the position of the planes that are detected in the scene and at the same time move the camera in reference to that environment.


Hierarchy
Create
HelloAR
HelloAR Controller
ARCore Device
First Person Camera
Plane Generator
Environmental Light
Point Cloud
EventSystem
PlaneDiscovery

Project
Create
Favorites
All Materials
All Models
All Prefabs
All Scripts
Assets
GoogleARCore
CLI
Configurations
Editor
Examples
AugmentedFa
AugmentedIm
CloudAnchors
Common
Materials
Prefabs
Scripts
Assets > GoogleARCore > Examples > Hello/
HelloAR

Inspector Services
First Person Camera Static
Tag MainCamera Layer Default

Transform
Position X 0 Y 0 Z 0
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1

Camera
Clear Flags Solid Color
Background
Culling Mask Everything
Projection Perspective
Field of View 60
Physical Camera ☐
Clipping Planes Near 0,05
Far 200
Viewport Rect X 0 Y 0
W 1 H 1
Depth 0
Rendering Path Use Graphics Settings
Target Texture None (Render Texture)
Occlusion Culling ✓
Allow HDR ✓
Allow MSAA ✓
Allow Dynamic Resolution ☐

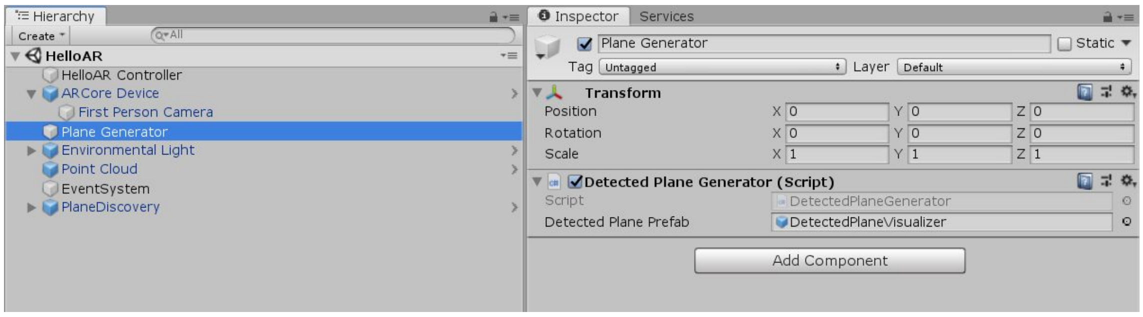
Tracked Pose Driver (Script)
Device Generic XR Device
Pose Source Color Camera
Tracking Type Rotation And Position
Update Type Update
Use Relative Transform ✓
Use Pose Provider None (Base Pose Provider)

AR Core Background Renderer (Script)
Script ARCoreBackgroundRenderer
Background Material ARBackground

Add Component

Fig.17. *ARCore Device* components

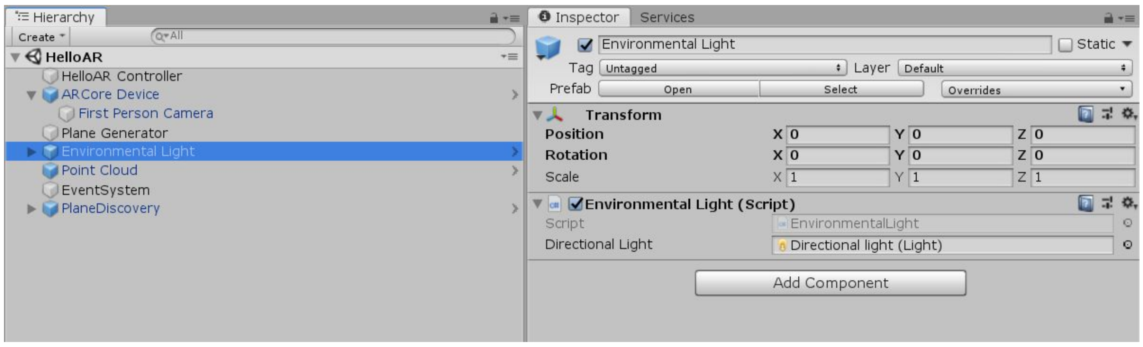
The *Plane Generator* component (see **Fig. 18**) presents another script, *Detected Plane Generator*, which as its name indicates will be in charge of showing the points and later the plane detected as they are generated.


Hierarchy
Create ▾ Q▾All
HelloAR
HelloAR Controller
ARCore Device
First Person Camera
Plane Generator
Environmental Light
Point Cloud
EventSystem
PlaneDiscovery

Inspector Services
Plane Generator ☐ Static
Tag Untagged Layer Default
Transform
Position X 0 Y 0 Z 0
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1
Detected Plane Generator (Script)
Script DetectedPlaneGenerator
Detected Plane Prefab DetectedPlaneVisualizer
Add Component

Fig.18. *Plane Generator* components

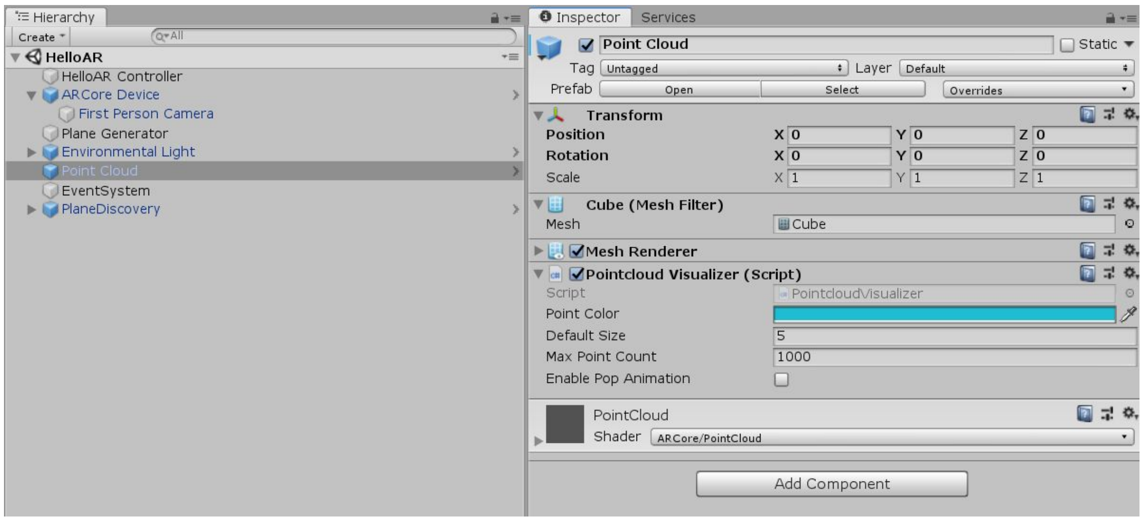
Environmental Light (see **Fig. 19**) with a script that allows you to change the direction of the light depending on the position of the camera with respect to the environment.


Hierarchy
Create ▾ Q▾All
HelloAR
HelloAR Controller
ARCore Device
First Person Camera
Plane Generator
Environmental Light
Point Cloud
EventSystem
PlaneDiscovery

Inspector Services
Environmental Light ☐ Static
Tag Untagged Layer Default
Prefab Open Select Overrides
Transform
Position X 0 Y 0 Z 0
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1
Environmental Light (Script)
Script EnvironmentalLight
Directional Light Directional light (Light)
Add Component

Fig. 19. *Environmental Light* components

The *Point Cloud* (see **Fig. 20**) is a component that, through its script, *Point Cloud Visualizer*, generates the cloud of characteristic points for creating planes.



HelloAR
- HelloAR Controller
- ARCore Device
  - First Person Camera
- Plane Generator
- Environmental Light
- Point Cloud
- EventSystem
- PlaneDiscovery

Point Cloud — Static
Tag: Untagged — Layer: Default
Prefab: Open | Select | Overrides

Transform
Position: X 0, Y 0, Z 0
Rotation: X 0, Y 0, Z 0
Scale: X 1, Y 1, Z 1

Cube (Mesh Filter)
Mesh: Cube

Mesh Renderer

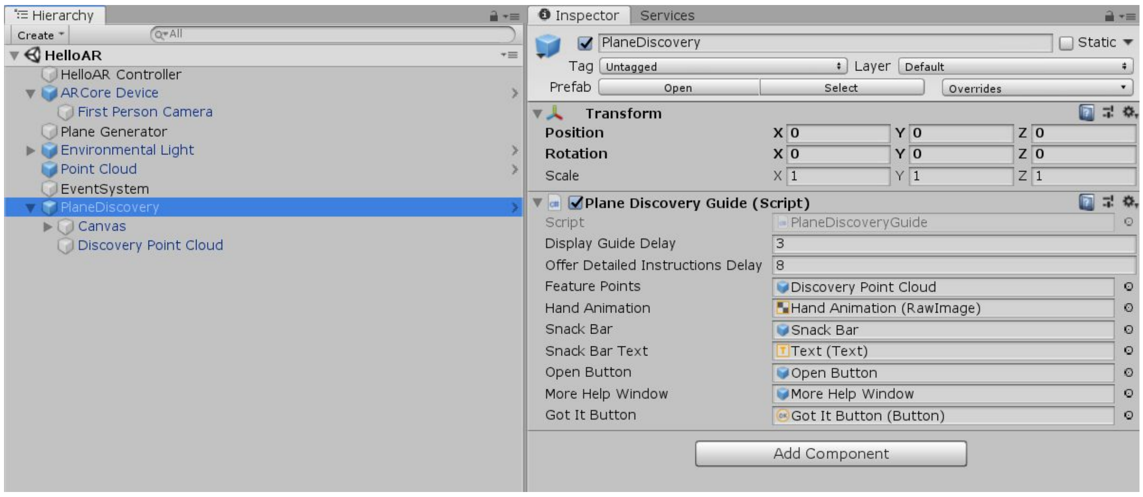
Pointcloud Visualizer (Script)
Script: PointcloudVisualizer
Point Color:
Default Size: 5
Max Point Count: 1000
Enable Pop Animation: ☐

PointCloud
Shader: ARCore/PointCloud

Fig. 20. *Point Cloud* components

The Event System is used to interact with the elements that create the scene or the canvas.

Plane Discovery (see **Fig. 21**)is in charge of managing everything related with the detection of planes.



HelloAR
- HelloAR Controller
- ARCore Device
  - First Person Camera
- Plane Generator
- Environmental Light
- Point Cloud
- EventSystem
- PlaneDiscovery
  - Canvas
  - Discovery Point Cloud

PlaneDiscovery — Static
Tag: Untagged — Layer: Default
Prefab: Open | Select | Overrides

Transform
Position: X 0, Y 0, Z 0
Rotation: X 0, Y 0, Z 0
Scale: X 1, Y 1, Z 1

Plane Discovery Guide (Script)
Script: PlaneDiscoveryGuide
Display Guide Delay: 3
Offer Detailed Instructions Delay: 8
Feature Points: Discovery Point Cloud
Hand Animation: Hand Animation (RawImage)
Snack Bar: Snack Bar
Snack Bar Text: Text (Text)
Open Button: Open Button
More Help Window: More Help Window
Got It Button: Got It Button (Button)

Fig. 21. *Plane Discovery* components

From here the scene was configured to be able to show a basic model once you click on a detected plane.

Next, a database was created for the dishes. But with one objective: that the database was easy to understand how to create, making the businessman who requests a menu in augmented reality, participate in the process in a very simple way.

Therefore it has to use XML documents in which is provided an example that the employer can open in Excel. In this the type of the dish is requested (Tapa, Salad, Burger, etc.), the name of the dish, ingredients and its price.

Thus, to implement it, the folder was created Resources within the project, for greater accessibility.

Inside this, first the Item script was created, which will allow to read the attributes such as: the type of dish, name of the dish, ingredients and price, written in the XML and transform them into those of an Item object. Then a test XML document creating different dishes with different attributes. A script called ItemContainer that will be in charge of reading the entire XML document generating a list of different Items each with their attributes. And, finally, a script called ItemLoader was created where the list created by ItemContainer will be loaded and that will facilitate the work when creating the HUD, since it will provide all the information.

Once the XML database reading system was implemented in the app, it was passed to the creation of the HUD.

The goal is for the HUD to be as integrated with reality as possible, as intuitive as possible, and usable with just one hand.

This will consist of a canvas with different components that will be turning and off while displaying the information available in XML

That is why when the menu button on the home screen is pressed, it will appear things shown from the camera back and the types of dishes (see **Fig. 22**) on sliders at the bottom of the screen, if you press a type they will appear more sliders at the bottom of the screen with the name of the dishes (see **Fig. 23**).

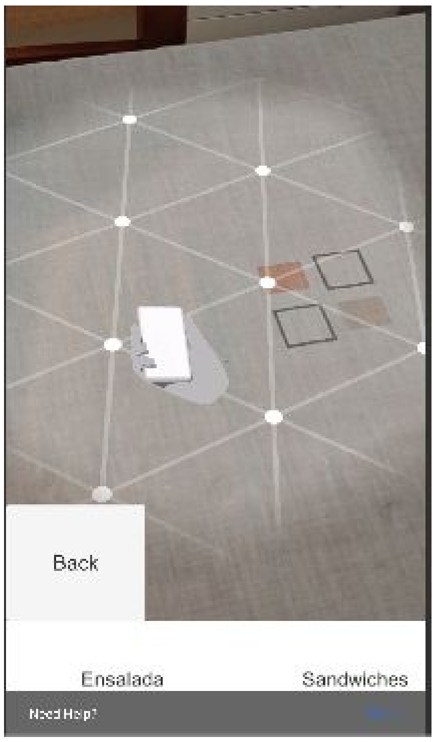


Fig. 22. HUD of types of plates

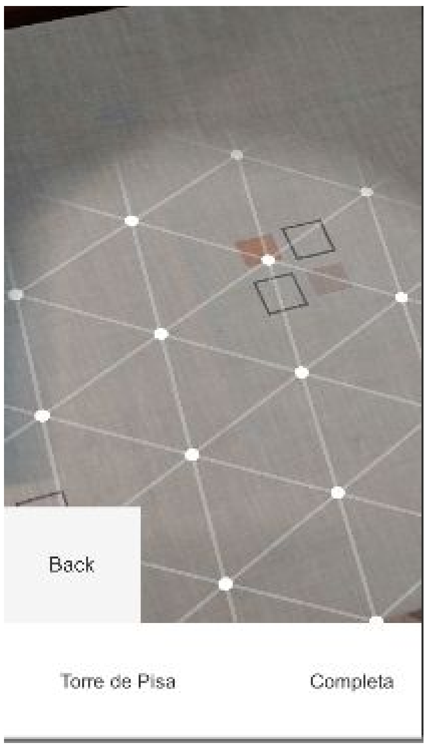


Fig. 23. HUD of name of plates

Finally, if one of them is pressed, its price will appear in the upper right corner, with the ingredients at the bottom of the screen, and the user will be allowed to press the screen on a scanned surface with a 3D model. In case you want to see it better, the 3D model can be rotated by moving your finger from side to side (see **Fig. 23**).

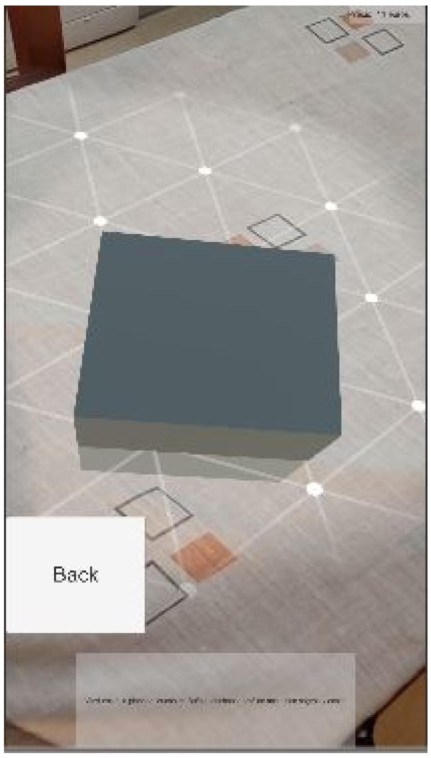


Fig. 24. 3D model displayed and HUD of ingredients and price.

In favor of the realism of augmented reality, photogrammetry has been used, thus allowing the greatest possible fidelity between the 3D model of the dish to choose and the one that the diner can receive at his table.

For this, first a real plate has been placed on a table and some 50 photographs have been taken from various points and angles. Then, through the use of a photo editing program such as *Adobe Photoshop* (see **Fig. 25**), touch-ups have been made, mainly removing everything other than the plate and its contents.

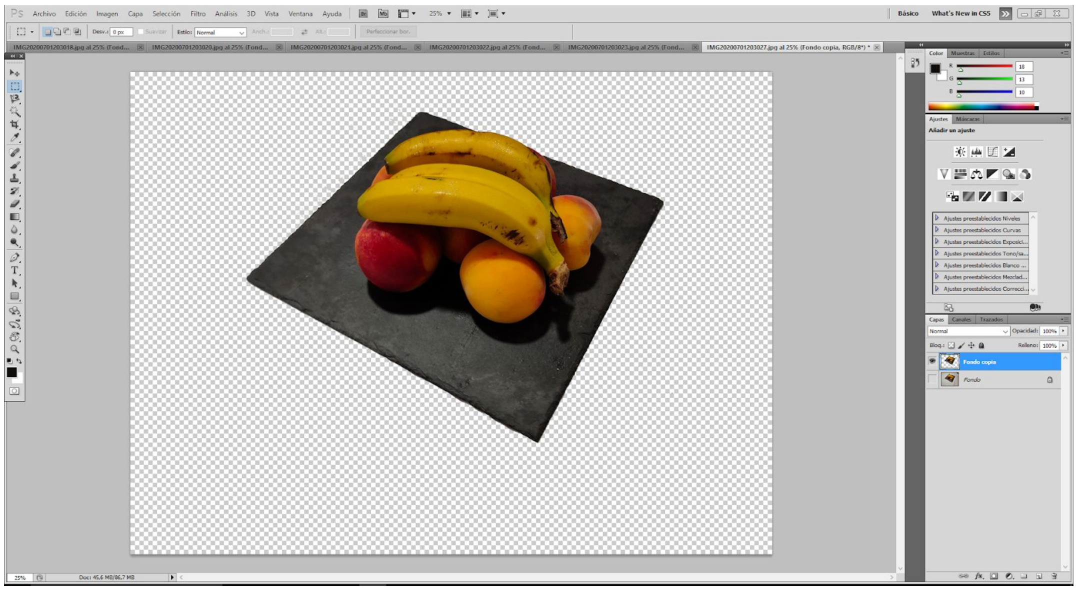


Fig.25. Photo in Adobe Photoshop

These photos were then imported into *Meshroom*(see **Fig.26**) that generated a 3D model with its texture and normal map. And finally, this model was exported to the free Blender program, which from there was exported to Unity for the final result.

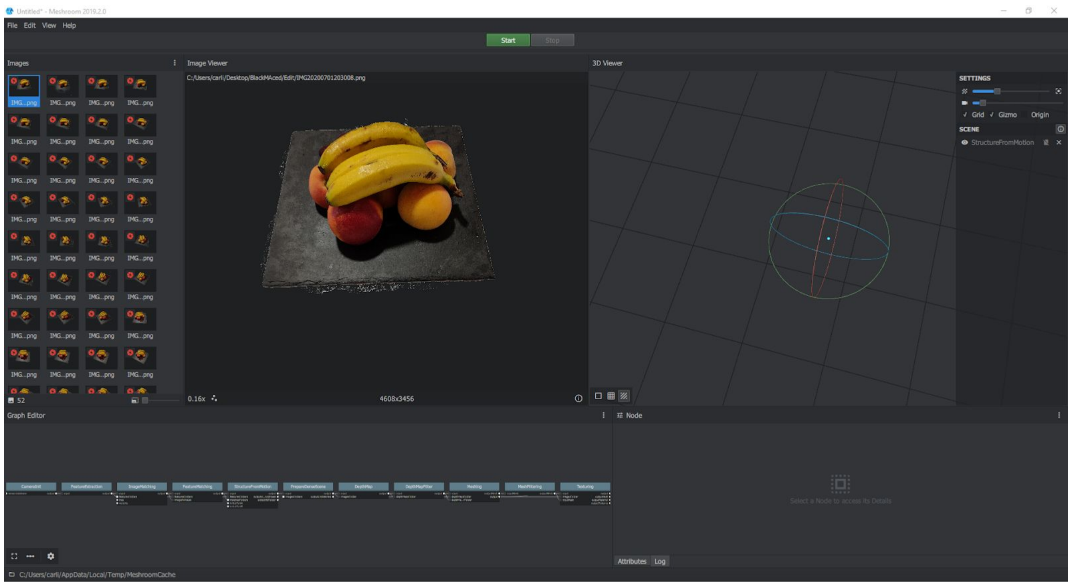


Fig. 26. Edited photographs in *Meshroom*

But after having some issues with the reconstruction of the mesh, I decided to use some free photoscan 3D models from Sketchfab[5].

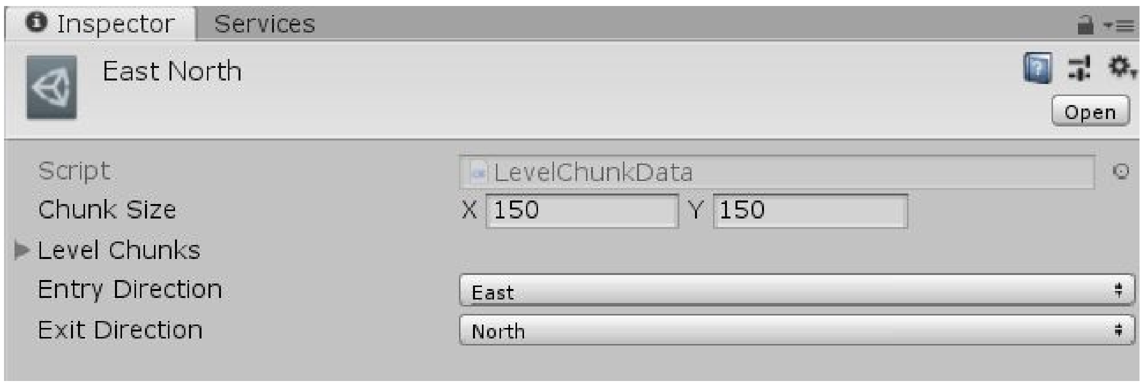
◦ **Ranking type mini game**

Once achieved, the next step was to create an endless runner[5], but trying to avoid its typical linear environments and its movement restricted to three invisible lanes.

So what I have called an "endless racer" was created.

In this the environment is not linear, but is generated procedurally by chunks[6] that due to certain conditions that they have and between them allows a random circuit to be generated and that will never end.

To do this, a *ScriptableObject*, was created to store the possible addresses, the size, the prefab to add, and the input and output addresses (see **Fig. 27 and 28**).



East North

Script	LevelChunkData
Chunk Size	X 150 Y 150
▶ Level Chunks	
Entry Direction	East
Exit Direction	North

Fig. 27. Chunk configuration.

[5]Genre where the player must irretrievably advance in the same direction, generally escaping from any enemy or danger and whose objective is to advance as much as possible before dying. The actions are generally jumping and dodging obstacles.

[6]Term used in video games that refers to map fragments made up of blocks with established measures, which as the character travels the world are generated when needed, generating up to a programmed number of them.

This allowed the creation of various types of chunks, causing them to have different input and output addresses (marked with colliders), different circuit shapes inside and even different objects within each one.

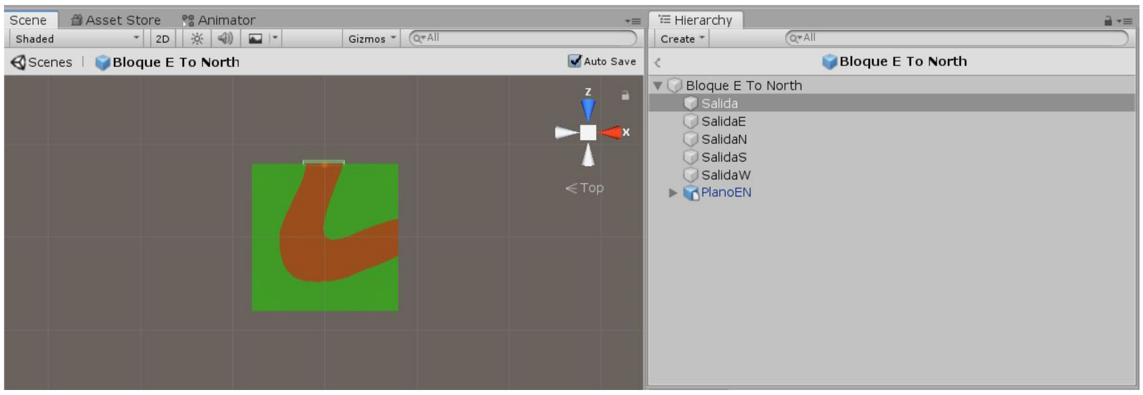


Fig. 28. Chunk from East to North design.

Next, to be able to build the circuit randomly created a script that took the different types of chunks and allowed to create, when initializing the game, as many as the programmer sees fit. Of course, respecting the condition that the chunks are always built northbound (Z Axis) (see **Fig. 29**).

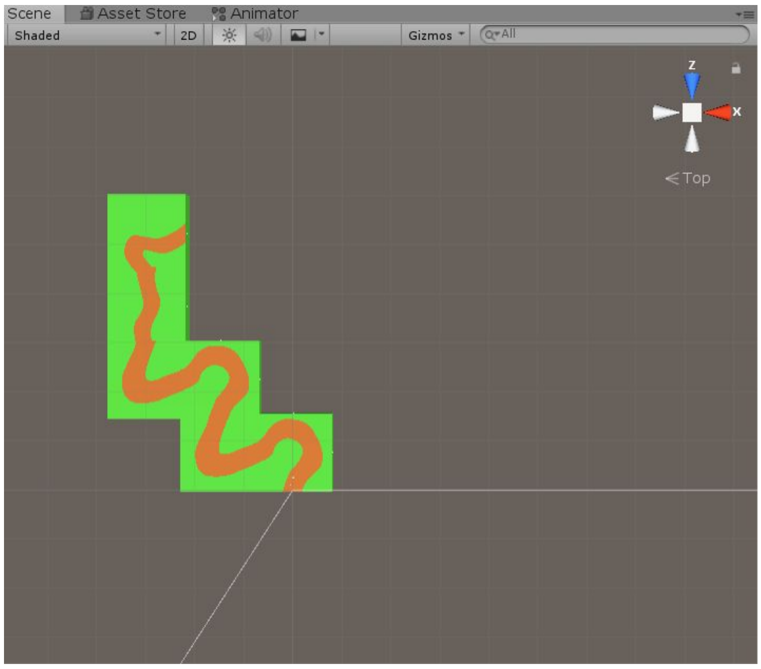


Fig. 29. Chunks created at start.

Once they are built and as the player passes and passes through the colliders of the different chunks, new ones will be generated and the ones already traveled will be eliminated.

But a problem appeared, and that is that after a long distance traveled, far from the origin, display problems began to be generated.
So a script was added to the main camera that, after a distance traveled, reset the player's position back to the origin.

As regards control, there are no invisible lanes, but the vehicle's turn, although restricted, to prevent the vehicle from turning, is free to make different lines.
Then when the gameplay, the target is kept in the loop as long possible as they longer are in it better our score.
The score meter was created from a canvas with text and a script that modified the text based on the time elapsed since the game started.

That is, 3 things will appear throughout the circuit: power-ups, coins and obstacles
Obstacles imply the end of the game when coming into contact with them.
For them, a script was created to manage that in case something of an obstacle type collides with the player, the value of the scoring script will be stored and it will go to the next scene.

Power-ups give us certain advantages when advancing along the circuit and even giving the player the possibility of avoiding obstacles. To obtain them, you will have to contact a 3D model that does not indicate what type of power-up you can touch, since this is totally random. Currently there is only one power-up that increases the character's speed for 3 seconds.
For them a *ScriptableObject* was created allowing them to have an image and access to modify player attributes to each of the power-ups. Along with this, a script was also created to detect the collision with the player and the random appearance on a specific canvas for power-ups.

And finally, there are the coins that come in contact with them increase our score, therefore, although they may apparently seem insignificant, an element that can make a difference. Its script is similar to that of obstacles except that in the event of a collision it increases the value of the score by 10 (see **Fig. 30**).

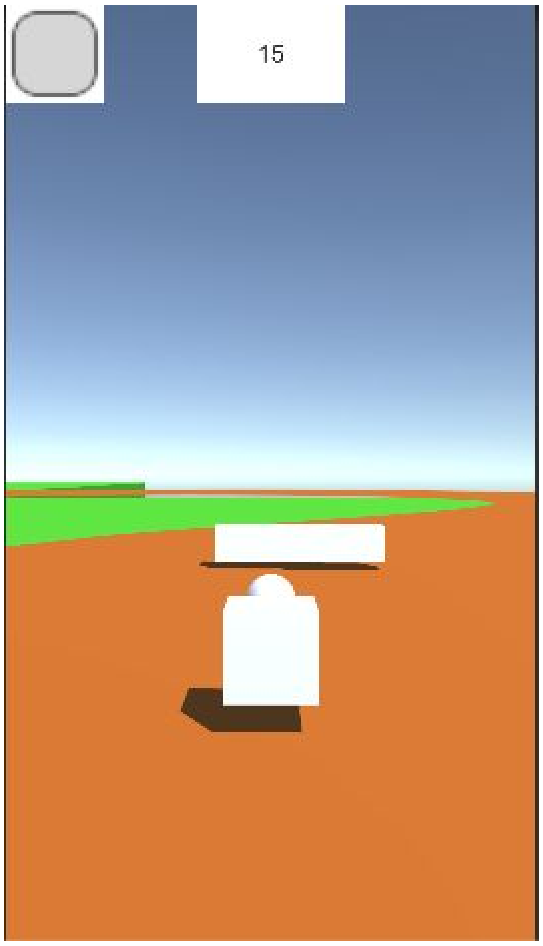


Fig. 30. Screenshot of mini-game with HUD (in development).

Next, a user registration system (see **Fig. 31**) was worked through the use of a SQL database and the XAMPP program in order to verify its correct operation using a local server. This system allows registering new users and their passwords or if they have already been created, check that they exist in the database and then update their score if necessary.

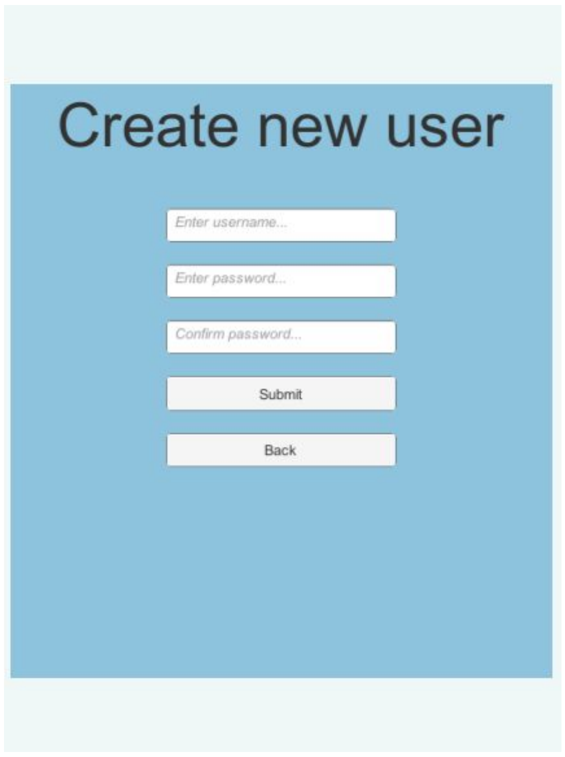


Fig. 31. Register interface

After, 3D models of the environment and character were imported from Sketchfab[5]. The environment of every chunk was created with all of the powerups and coin and a tap to start screen was created, so the player starts to play when he taps the screen. Although, a pause button was added at the upper right corner of the screen, making possible that the player pauses the game when he wants and if he wants to go to the main screen.
Finally, a game over screen was added ,so the player can see his final score, and a highscore panel was created, for displaying the highscores of all the players registered in the database.

4.2 Results

The result with respect to the initial idea has varied in certain respects

Without going any further, the use of *Vuforia* and its use of virtual reality through images by reference was considered optimal.

But after the problems of image recognition, in case, that the reference image is lost or that it is in one object with reflective properties, it was decided not to use it and opt for *ARCore*.

This is because *ARCore* allows 3D models to be arranged in augmented reality without the need of a reference image, but rather by scanning the space and making the 3D models display at one point in space and remain at that point. In addition to this, is added the ability that the objects shown will present greater realism thanks to the estimation of light in real time with the environment, and that for the future, the *ARCore* being developed by *Google* has great potential.

Referring to the HUD, the menu has been modified so that augmented reality has a greater presence and its use is easy with one hand, and this are the results (see **Fig. 33, 34 and 35**)

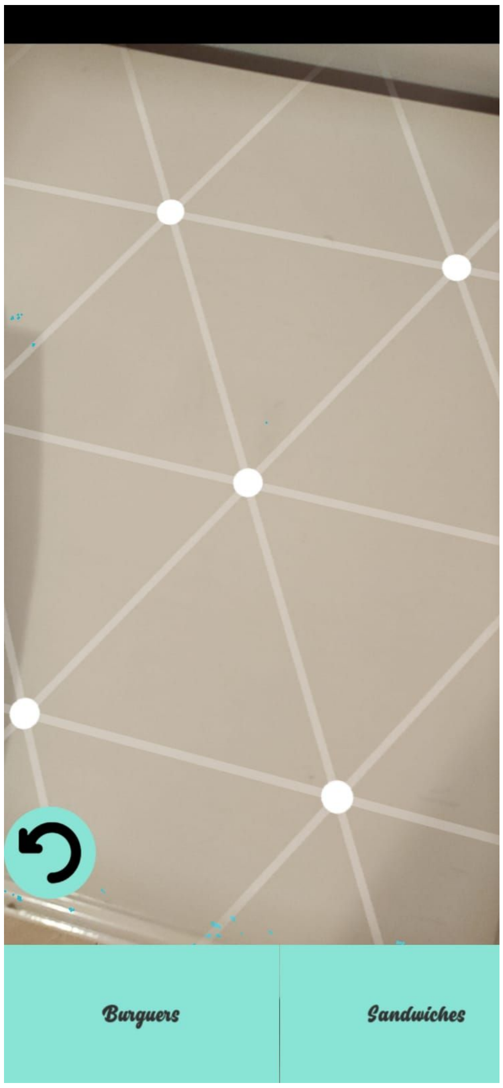


Fig. 32. Types of plates

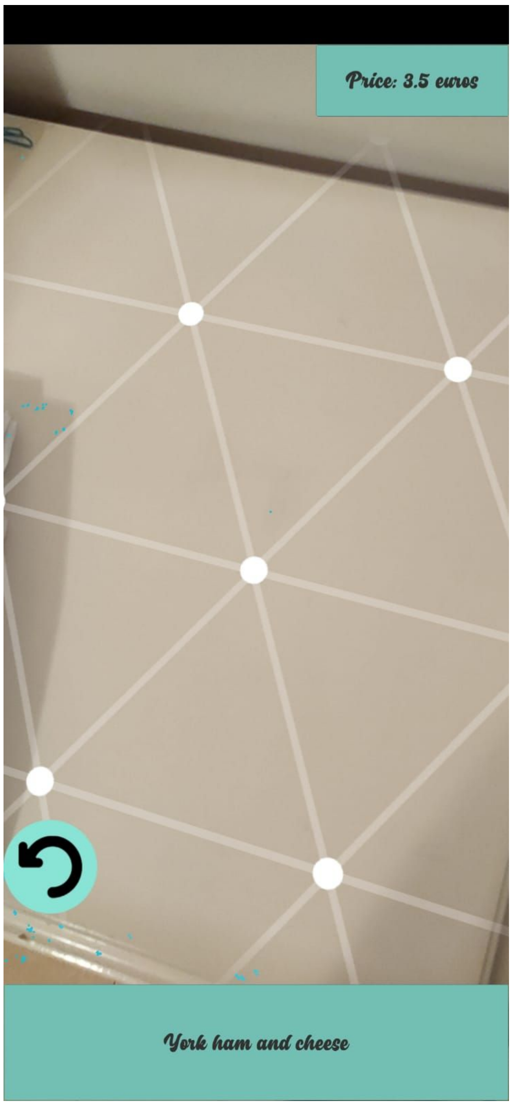


Fig. 33. Ingredients and price

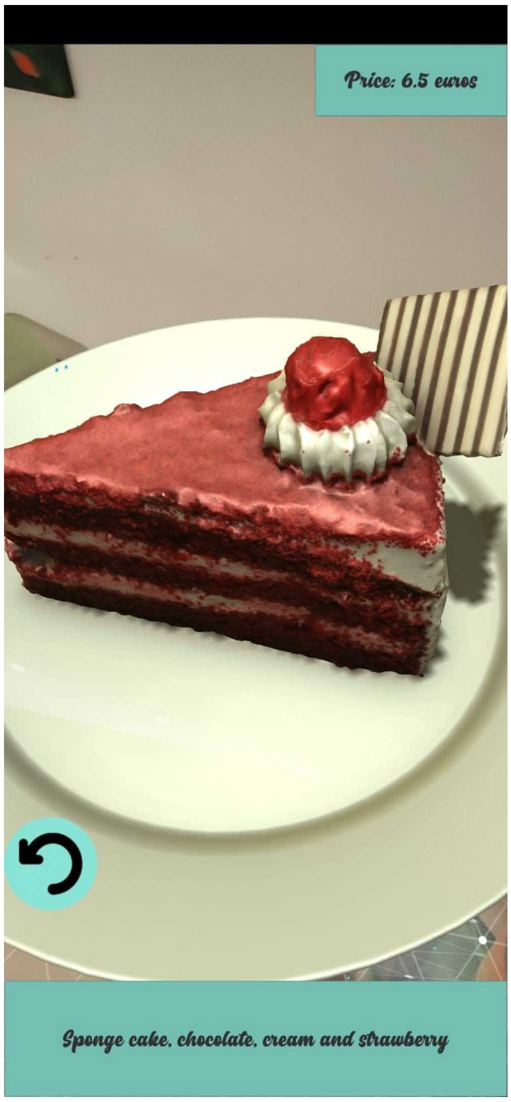


Fig 34. 3d model in AR

The multiplayer game has been removed from the initial idea because it has not been considered relevant since the main idea of this app is to modernize the menus of the restaurants and their loyalty system. To this, it is added that the project has to meet a few target hours and to create it, it would take more time.

But this are some screenshots of the results of the renking type mini game (see **Fig.35, 36 and 37**)

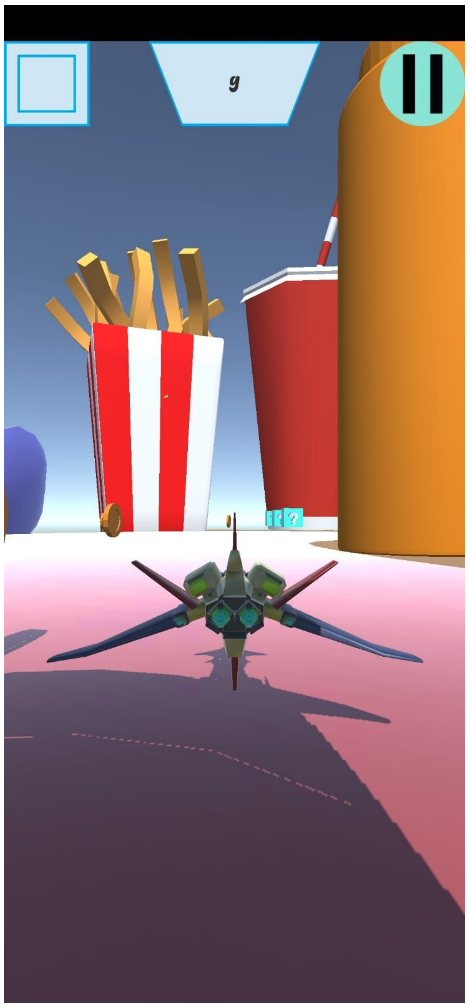
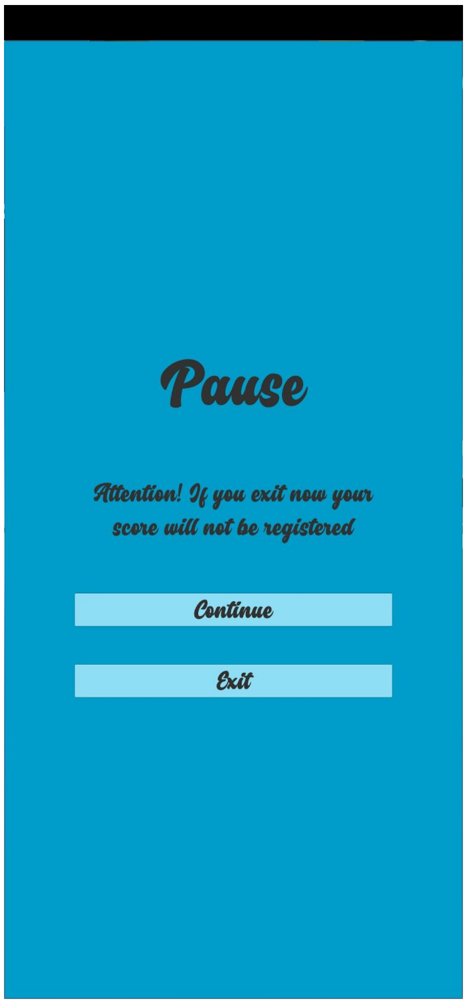


Fig. 35. In-game screenshot



Attention! If you exit now your score will not be registered

Fig. 36. Pause menu

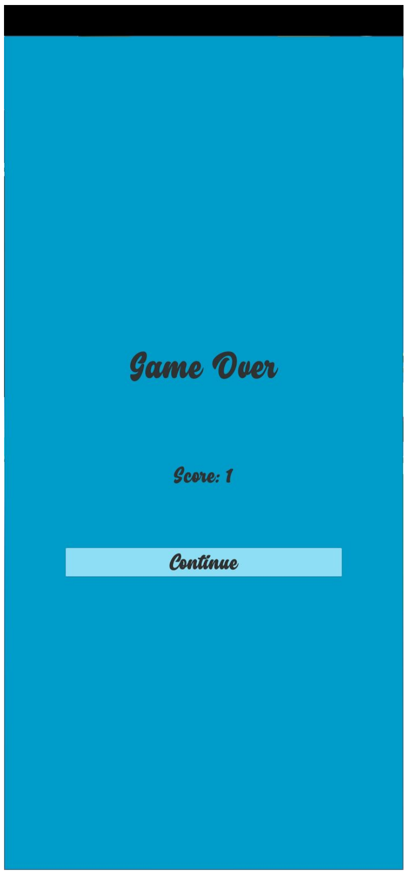


Fig. 37. Game Over menu

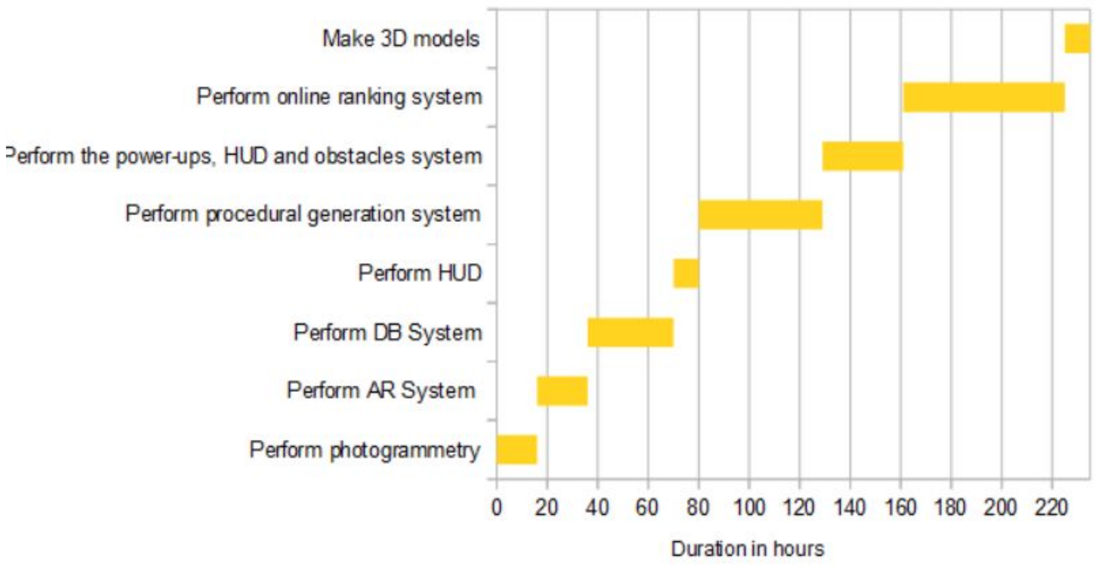


Fig. 38. Gantt chart of the real time expend in the app.

5.CONCLUSIONS AND FUTURE WORK

5.1 Conclusions

This project has been quite an experience, because it has allowed me to create something that I believe can be the future. Not only because the app makes use of the latest technology in AR for mobiles devices, empowering the fact that in the future we could do almost everything with them. It is the idea of creating, for the games, environments where they never had been before, and making them not only more accessible but much more necessary for business.

That is what I believe and I am proud because on a personal level this project has pushed me to limits that I did not think I could achieve.

5.2 Future work

I think it is an application that can have a great future, especially now facing a post-COVID-19 era, where diners will not have to choose their dishes in a way sung by a waiter or from a laminated and disinfected menu. But thanks to their smartphones they can access it without any problem and better than ever.

That is why in this application I can continue working, creating an environment when all the restaurants can be in this same app using geolocation, creating multiplayer games(even in AR) so that people enjoy more than ever being accompanied and new loyalty games such as fantasy sports games.

6.BIBLIOGRAPHY

[1]Vuforia developers**.** https://developer.vuforia.com

[2]ARCore concepts. https://developers.google.com/ar/discover/concepts

[3]ARCore quickstart for developers. https://developers.google.com/ar/develop/unity/quickstart-android#get_the_arcore_sdk_for_unity

[4]ARCore SDK**.** https://github.com/google-ar/arcore-unity-sdk/releases

[5] Sketchfab 3d models. https://sketchfab.com/