

---

This is the **published version** of the master thesis:

Ortiz Rodríguez de Liébana, Miguel; Carrabina Bordoll, Jordi , dir. Localization, tracking and guidance with Android app. 2019. 94 pag. (1170 Màster Universitari en Enginyeria de Telecomunicació / Telecommunication Engineering)

---

This version is available at <https://ddd.uab.cat/record/259409>

under the terms of the  license



Master's Thesis

**Master in Telecommunication Engineering**

---

# Localization, Tracking and Guidance with Android app

Miguel Ortiz Rodríguez de Liébana

---

Supervisor: Jordi Carrabina Bordoll

*Departament de Microelectrònica i Sistemes Electrònics*

**Escola Tècnica Superior d'Enginyeria (ETSE)**

**Universitat Autònoma de Barcelona (UAB)**

September 2019



El sotasignant, *Nom del Professor*, Professor de l'Escola Tècnica Superior d'Enginyeria (ETSE) de la Universitat Autònoma de Barcelona (UAB),

CERTIFICA:

Que el projecte presentat en aquesta memòria de Treball Final de Master ha estat realitzat sota la seva direcció per l'alumne *Miguel Ortiz Rodríguez de Liébana*.

I, perquè consti a tots els efectes, signa el present certificat.

Bellaterra, 09/09/2019.

Signatura:      Jordi Carrabina Bordoll

**Acknowledgments**

I would like to express my special appreciation to my thesis supervisors, Phd Jordi Carrabina and Prf. Marc Codina. They have provided me their support on this project. I would also thank my colleagues from work for their support, kind advice and patience to resolve my questions. Furthermore, I would like to give a special thanks to my family and friends for their comprehension and affection during this year.

**Resum:**

*Avui en dia la evacuació total o parcial d'un edifici en cas d' alarma es basa en el coneixement dels usuaris del pla d'evacuació.*

*En aquest projecte es proposa una solució que complementi el pla d'evacuació i ajudi als usuaris durant el procés d'evacuació a arribar al punt de reunió amb l'ajuda de tecnologies mòbils i de localització indoor. Per això es proposa el desenvolupament d'una aplicació mòbil de Realitat Augmentada per fer servir en smartphones amb Android OS. Aquesta aplicació detecta la posició de l'usuari mitjançant tecnologia de posicionament indoor i el guia per l'edifici mitjançant senyals visuals per arribar a un punt de reunió.*

**Resumen:**

*Hoy en día la evacuación total o parcial de un edificio en caso de alarma se basa en el conocimiento de los usuarios del plan de evacuación.*

*En este proyecto se propone una solución que complemente al plan de evacuación y ayude al usuario durante el proceso de evacuación a llegar al punto de reunión con la ayuda de tecnologías móviles y de localización indoor. Para ello se propone el desarrollo de una aplicación móvil de Realidad Aumentada para utilizar en smartphones con Android OS. La aplicación detecta la posición del usuario mediante tecnología de posicionamiento indoor y le guía a través del edificio mediante señales visuales para llegar a un punto de reunión.*

**Summary:**

*Nowadays the total or partial evacuation of a building in the event of an alarm is based on the user knowledge of the evacuation plan.*

*The purpose of this project is to provide a complementary solution to the emergency evacuation plan. It will help the user during the evacuation process to reach the designated assembly area through mobile technologies and indoor location. Therefore, an Augmented Reality mobile application has been developed for smartphones with Android OS. The application detects the user position by indoor positioning technology and guides him through the building by visual signals in order to reach the designed assembly area.*

## Contents

1	Introduction.....	1
1.1	Motivation .....	1
1.2	Objectives .....	2
1.3	Thesis structure.....	2
2	State of Art.....	5
2.1	Mobile Operating System.....	5
2.2	Programming Languages .....	5
2.3	Mobile devices – smartphones and smart glasses .....	6
2.4	Indoor Positioning .....	7
3	Problem and Solution.....	8
3.1	Problem.....	8
3.2	Solution.....	8
4	Android .....	14
4.1	History and features.....	14
4.2	Android Studio .....	16
5	Java .....	17
5.1	History and features.....	17
5.2	Java API.....	17
5.3	Java - TIOBE Index .....	18
6	Augmented Reality & JPCT 3D engine.....	19
6.1	History .....	19
6.2	Applications.....	19
6.3	JPCT 3D engine.....	20
7	Indoor Localization System .....	21
7.1	History and features.....	21
7.2	Applications.....	23
8	Planning and resources .....	24
8.1	Planning - Gantt Chart.....	24
8.2	Costs and resources .....	25
9	Analysis.....	27
10	Application Development .....	29
10.1	Workflows and algorithms .....	29
10.2	Virtual World.....	30
10.3	Development.....	34
10.4	Functions .....	36
10.5	North Magnetic alignment.....	42
11	Tests and Results.....	47
11.1	Tests.....	47
11.2	Results .....	47

12	Conclusions and Future Works .....	51
12.1	Conclusions .....	51
12.2	Future steps.....	51
13	Bibliography and References .....	53
14	Annex - Code .....	54

## List of Figures

Figure 1. Smartphones companies .....	6
Figure 2. Google Glass.....	7
Figure 3. Museo del Prado map by Rafael Moneo.....	8
Figure 4. Signals.....	9
Figure 5. Interactive Kiosk.....	9
Figure 6. Virtual Maps .....	10
Figure 7. SIGUE application.....	10
Figure 8. Android versions 1.....	11
Figure 9. Android versions 2.....	12
Figure 10. Android Studio.....	12
Figure 11. Java .....	12
Figure 12. JPCT 3D engine .....	13
Figure 13. Xiaomi mi A1 .....	13
Figure 14. Android devices .....	14
Figure 15. Market OS distribution. Information source: Gartner Inc. ....	15
Figure 16. Android experience.....	15
Figure 17. Android Studio interface.....	16
Figure 18. Java API.....	18
Figure 19. TIOBE index 2019.....	18
Figure 20. AR: industry applications .....	19
Figure 21. GPS vs. IPS.....	21
Figure 22. Inertial positioning system flow diagram .....	22
Figure 23. a)Time of arrival approach; b)angle of arrival approach; c)hybrid ToA/AoA approach; d)received signal strength and fingerprint approach .....	22
Figure 24. IPS techniques combined.....	23
Figure 25. Gantt chart.....	24
Figure 26. Logical workflow diagram.....	29
Figure 27. Virtual reference axis.....	30
Figure 28. Android Studio dependencies .....	31
Figure 29. Virtual World map .....	32
Figure 30. Virtual World view .....	33
Figure 31. Virtual World 1 .....	33
Figure 32. Virtual World 2.....	33



Figure 33. Virtual World 3 .....	34
Figure 34. Virtual World axis .....	34
Figure 35. Android Studio configuration .....	35
Figure 36. Android Vritual Device Manager .....	35
Figure 37. Android Manifest - App name .....	36
Figure 38. onSurfaceCreated .....	39
Figure 39. onSurfaceChanged .....	39
Figure 40. onDrawFrame .....	39
Figure 41. onTouch .....	40
Figure 42. IsInWorld .....	40
Figure 43. PointsInvisibles .....	40
Figure 44. IsWayOut .....	40
Figure 45. CalcPoint_RoomCorr.....	41
Figure 46. Calc_Room .....	41
Figure 47. CalcRuta.....	41
Figure 48. onSensorChanged .....	42
Figure 49. onAccuracyChanged .....	42
Figure 50. panBy .....	42
Figure 51. Azimuth - Roll - Pitch.....	43
Figure 52. Azimuth angle monitoring .....	43
Figure 53. Camera rotation.....	44
Figure 54. Virtual World and Magnetic heading directions.....	45
Figure 55. User 1 - application.....	48
Figure 56. User 2 - application.....	48
Figure 57. Device 1. - Android 4.4 kitkat .....	48
Figure 58. Device 1.2 - Android 4.4. kit kat .....	49
Figure 59. Virtual World.....	49
Figure 60. Checkpoint signal Room 1 .....	49
Figure 61. Checkpoint signal Room 2.....	50

## List of Tables

Table 1. Project costs .....	26
Table 2. JPCT Classes .....	31

## Abbreviations

<b>IPS</b>	Indoor Positioning System
<b>GPS</b>	Global Positioning System
<b>IDE</b>	Integrated Development Environment
<b>API</b>	Application Programming Interface
<b>OS</b>	Operating System
<b>OOP</b>	Object-Oriented Programming
<b>UI</b>	User Interface
<b>SoC</b>	System on Chip
<b>LPS</b>	Local Positioning System
<b>BLE</b>	Bluetooth Low Energy
<b>NFC</b>	Near Field Communication
<b>JVM</b>	Java Virtual Machine
<b>JDK</b>	Java Development Kit
<b>OpenGL</b>	Open Graphics Library
<b>GNSS</b>	Global Navigation Satellite System
<b>RSSI</b>	Received Signal Strength Indicator
<b>IMU</b>	Inertial Measurement Units
<b>INS</b>	Inertial Navigation System
<b>WP</b>	Work Package
<b>WF</b>	Workflow
<b>APP</b>	Application (SW Mobile device)
<b>AR</b>	Augmented Reality
<b>AI</b>	Artificial Intelligence
<b>SW</b>	Software
<b>HW</b>	Hardware
<b>DVP</b>	Design Verification Plan



# **1 Introduction**

This first chapter explains the motivations for choosing this topic and working on this project. It also exposes the different objectives to be achieved. Finally, a brief explanation of the structure of this thesis will be introduced.

## ***1.1 Motivation***

Currently building evacuation depends on the knowledge the users have of the emergency evacuation plan and space distribution of a building. Psychological factors such as stress due to the alarming situation can cause mental block into the user and remain difficult the evacuation process. Even worse, what would happen if the user did not know at all the space distribution of the building?

This project pretends to design a solution based on mobile devices, Augmented Reality (AR) and IPS (Indoor Position System) in order to help the user, find the designated assembly area timely during the evacuation.

The solution is adapted in an emergency evacuation context, but it can be used for other purposes, for instance to move inside a building from a current position A to another position B. Never mind that the building is a shopping mall or an office, the user can easily be guided through the spaces of the building without losing time to find a plan or an information desk.

The main reason for being part of this project is for me an opportunity to develop a solution for a real issue working with different technologies and more precisely the mobile ones.

During those last years, mobile applications have given solutions to real problems in education, finance, mobility to mention just a few examples and making life easier for people. As an engineer I am willing to solve social problems with technology and the knowledge that I have acquired during my master's degree in engineering.

Thanks to this project, I am delighted to introduce myself to Android mobile applications development, to work with Java programming language and to investigate 3D virtual environments.

## **1.2 Objectives**

The aim of this project is to build an Augmented Reality application that shows the way out to the assembly area during the evacuation process.

For that purpose, it is mandatory that the following steps be developed:

- A 3D virtual environment of a building or part of it
- Fundamental functions for guidance in a virtual environment and the receipt of the position signal sent by the indoor localization system.
- Development of an Android application to be installed on smartphone with Android OS.

To achieve these objectives two prototypes will be created:

1. First prototype - virtual solution to verify the viability of the application.
  - 1.1 Development of a virtual map for the building prototype\_1
  - 1.2 Development of a navigation system in 3D virtual environment
  - 1.3 Development of a reference system for the virtual and real-world environments
  - 1.4 Development of the Way-Out function\_1
  - 1.5 Development of the signalization functions
  - 1.6 Test of prototype\_1
2. Second prototype - integration of prototype\_1 to the real-world environment.
  - 2.1 Development of a virtual map for the building prototype\_2
  - 2.2 Development of the Way-Out function\_2
  - 2.3 Development of the camera function to merge the virtual and real-world views
  - 2.4 Development of data acquisition system and adaptation to the indoor positioning
  - 2.5 Test of prototype\_2

## **1.3 Thesis structure**

The project is developed in 14 chapters, there is a short description of each one of them.

### **Chapter 1 - Introduction**

Definition of key concepts of the development and definition of the projects, motivations, objectives and structure.

## **Chapter 2 - State of Art**

Introduction to the different technologies used to develop the solution.

## **Chapter 3 - Problem and solution**

Explanation of the identified problem and the conceptual and technical solution proposed.

## **Chapter 4 - Android**

Chapter focused on explaining why the Android OS had been chosen and his principals characteristics. Also, a review of the IDE Android Studio used for the development of Android applications.

## **Chapter 5 - Java**

Chapter dedicated to the history of the programming language Java, the motivations why it has been selected and his impact in the current market.

## **Chapter 6 - Augmented Reality & JPCT 3D engine**

History of the Augmented Reality, his main applications and the organizations that used this technology. Also, comment on the 3D library used in this project.

## **Chapter 7 - Indoor positioning system (IPS)**

Point dedicated to explaining the IPS systems and opportunities that offer.

## **Chapter 8 - Planning and resources**

Description of the planning to develop the solution of the project. Following to the introduction to the resources that have been used for the project.

## **Chapter 9 – Analysis**

Definition of the projects scope.

## **Chapter 10 – Application Development**

Study of the main functions and development of the virtual world, the logical of the functions, the workflows developed, and the integration of the hardware used.

## **Chapter 11 – Test and Results**

Description of the verification and validation plan where all tests are defined.

Presentation of the tests results.

## **Chapter 12 - Conclusions and Future Works**

This chapter closes the project doing an evaluation of the results obtained and proposing future works.

## **Chapter 13 - Bibliography and References**

References to all the articles, books, blogs and websites consulted and mentioned for this project.

## **Chapter 14 - Annex**

Application code



## **2 State of Art**

Since the arrival of the mobile OS, the world has lived a digital revolution. These systems have permitted that a large range of services could be concentrated in only one device, a smartphone, a smartwatch, a tablet or smart glasses to mention only that.

Those last few years a significant number of solutions have emerged from the fusion of mobile technologies and identification technologies, positioning or information technologies, thanks to the successful evolution on the electronic area, specifically in electronic mobile devices.

Another significant point is the number of professionals with technology background who contributes to the exponential evolution of some programming languages and technologies. For that reason, a lot of technologies as Augmented Reality has shown a huge increase of their applications in different sectors: industry (receive instructions to repair a machine), advertising (outside advertising campaigns), tourism (show information of spaces or places), video games (PokemonGo), museums (giving life to drawing paint), medical (visualization of the data of the patient), architecture (showing the process of construction of a virtual house), academical (showing information on an interactive way).

### **2.1 Mobile Operating System**

It is about programs than facilitate the interaction between the user and the mobile device programs. There are several mobile systems in the market, but more than 98% of all the devices use Google Android OS or Apple IOS. This sector is in constant evolution, this past August the last mobile operating system was introduced as a rival of both mentioned previously. It is called HarmonyOS and is developed by the Chinese company Huawei.

These operating systems are oriented specifically to the wireless connectivity and multimedia format for mobile phones.

On the Spanish market, the OS which predominates is Android with almost 90% of the part of the market.

### **2.2 Programming Languages**

Since the creation of computing systems that allows programming and creating software, many programming languages have been created. Object-oriented programming (OOP) languages are the most used in those last years. Most of these programming languages have the capacity to

develop multiplatform solutions, to group their functions in APIs and few of them have the support of big corporations and communities. Their main advantages are to be able to recycle the components, easy maintenance and modular structure. The OOP most used are Java, C#, C++, Python, Objective-C y PHP.

Nowadays, there are used for the development of software in helpdesk applications, in backend solutions, in UI...

### ***2.3 Mobile devices – smartphones and smart glasses***

Mobile devices are electronic devices which combine the benefits of computers and mobile phones. These devices have the following characteristics: mobile operating system, SoC with high processing capacity, connection to network, memory, GPS, touch screen, integrated battery, cameras and wearable.

Between all the mobile devices the smartphones stand out. They revolutionized all sectors since their apparition ten years ago. Smartphones are very popular in all generations and accessible to all social categories. They have become an indispensable tool for professionals and personal uses. They also allow to install some applications as agenda, calendar, chats, mailbox, web browser, video games. All these applications make life easier. Since their arrival, these devices have modified our habits as shopping or social relationships.

The use of Smartphones has benefited from the improvement of OS and technology evolutions as SoCs (System on Chips), touch screens and batteries. Today these devices have the support of big corporations like Apple, Google, Huawei, Xiaomi, Oppo, ZTE or Samsung.



*Figure 1. Smartphones companies*

Thanks to continuous improvement of those devices, new technologies have gained importance through the years like AR and AI.

Regarding smart glasses, these devices had a huge boom in 2012. It is the incorporation of smartphone to glasses. The most well-known smart glasses are the “Google Glass”. There are also other options as for instance “HoloLens” by Microsoft.

The arrival of smart glasses had a positive reputation in the professional world due to the virtual reality and the development of customized applications in certain fields.



*Figure 2. Google Glass*

## **2.4 Indoor Positioning**

Through the years, outdoor positioning provides a high-quality solution thanks to GPS and GLONASS, but the indoor positioning solutions are still improving their technologies.

There are two positioning systems: **Global** as GPS which uses the satellites to calculate the position. **Local** systems called Local Positioning System (LPS) which calculates the position using local mechanisms like WIFI stations and telephony stations.

The system used to obtain the position in indoor places is the Indoor Position System (IPS). The IPS are specific cases of LPS. These systems are focused on indoor positioning in spaces no exposed to the outdoor.

The most important technologies used for indoor positioning are WIFI and Bluetooth.

The Bluetooth Low Energy (BLE) can reach a high precision in the positioning. BLE uses a hardware (called beacons) much more economical than a WIFI router moreover it had a reduced operation range. Normally They use the fingerprint technique.

### 3 Problem and Solution

#### 3.1 Problem

There are several reasons for what a person will need to move into a building to one point to another easily and quickly.

First, we must consider what could happen if a person does not have the knowledge of the distribution of the indoor spaces of a building where he is. Following examples of places where this situation could happen:

1. Shopping Malls
2. Public buildings as school university or museums.
3. Hospital centers
4. Corporate buildings

Other reasons for moving into a building quickly is in the case of emergency alarm. In this case is mandatory that the person knows:

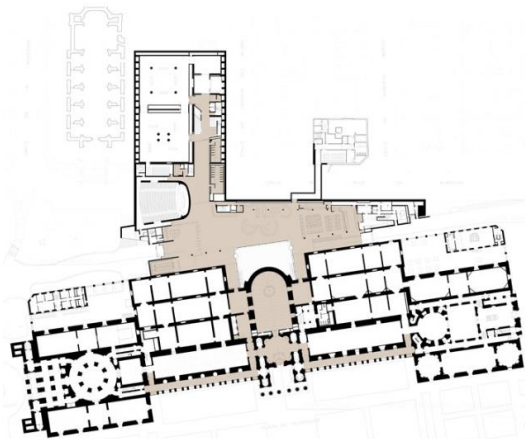
1. Where He/She must go
2. The route knowledge to go until the assembly area

#### 3.2 Solution

Nowadays a few solutions exist to resolve indoor positioning problems.

#### **BUILDING MAP**

The first solution is building maps. The user must have the physical map of the building under his eyes and be able to read the map. It is not a simple solution because there is not a lot of maps in a building and there are in some determined points. Everybody also does not have the same ability to interpret properly a map.

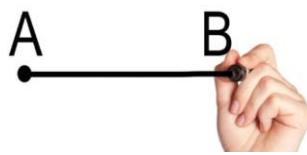


*Figure 3. Museo del Prado map by Rafael Moneo*

## INFORMATION DESK - EXPERT

Another option is to ask at the information desk to a professional or a person who knows the building. This solution supposes that these two persons have enough knowledge of the building. After asking how to move to the final point all depends on the quality and the reliability of the information provided. It is a solution usually used in hospitals, museums and office buildings but this solution has some disadvantages:

1. Find the right person at this specific moment when we need to know the way
2. Rely on the information provided by the person with the knowledge of the building
3. Be able to interpret the information received



*Figure 4. Signals*

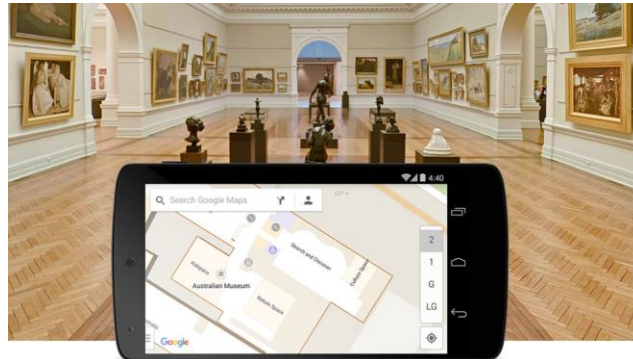
## VIRTUAL MAP

This following solution is a virtual map of the building. On this kind of maps spaces and including the ways are shown as the user must follow to go from initial point to the final point. There are two options to see the virtual maps. The first one is the **Interactive Kiosk**, it is a touch screen located in a determined point in the building and the user can consult the way to go to a place in the building.



*Figure 5. Interactive Kiosk*

The second one is a **Virtual Map** that the user can consult in his own mobile device for instance a smartphone. They are working the same as interactives Kiosks, but the difference is that ones are not in a determined point.



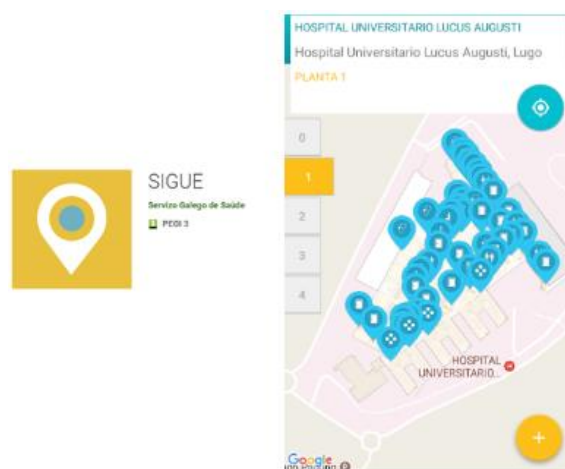
*Figure 6. Virtual Maps*

Therefore, the virtual map solution allows to show the way to the user from one determined point (kiosk) or in one determined moment (mobile device). The disadvantages of both options are the following points:

- The user must remember the way indicated in the screen of the kiosk.
- The mobile device offers a solution no connected with the real-world environment.

That is why these solutions are not complete.

The solution based on virtual map developed by the Public Galician Service of Saúde deserves to be highlighted. It is an application called “SIGUE” available on Play Store of Android and AppStore of Apple. Different services are centralized in this application and can show to the user a way to move through the hospitals of the Galician Health Service.



*Figure 7. SIGUE application*

## PROJECT SOLUTION

The solution proposed in this project improves the previous solutions exposed. This one brings a high added value solution with a reduced developing cost.

In this project, Android will be used as an operating system. This OS allows us to develop an application programmed with JAVA. This application can be used in some Android mobile device. Through the camera of the mobile device, the application captures the pictures of the building from the position of the user and shows him the necessary signals in real time to be guided intuitively to the final point.

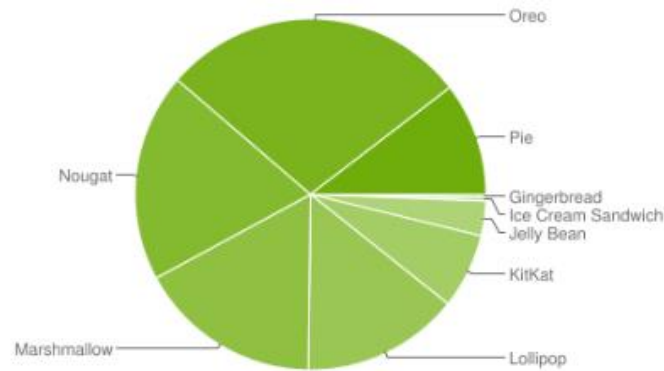
Android permits developing applications for mobile devices. This fact allows avoiding the process of design a customized hardware. The selection of Android is due to the following points:

- Very extended: there are a lot of devices commercialized which use OS
- Developers license is economical, and it is one-time payment
- Easy to download and install on a mobile device
- Updating the application is easy and fast
- Many code libraries
- Available for a different type of mobile devices (tablets, smartphones...)

The minimal version for the developed application to be operative is Android 4.4 KitKat which meets with the API level 19. The reason for selected this minimal version is because 96.2% of Android OS commercialized devices are compatible with this version.

Version	Codename	API	Distribution
2.3.3 - 2.3.7	Gingerbread	10	0.3%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	0.3%
4.1.x	Jelly Bean	16	1.2%
4.2.x		17	1.5%
4.3		18	0.5%
4.4	KitKat	19	6.9%
5.0	Lollipop	21	3.0%
5.1		22	11.5%
6.0	Marshmallow	23	16.9%
7.0	Nougat	24	11.4%
7.1		25	7.8%
8.0	Oreo	26	12.9%
8.1		27	15.4%
9	Pie	28	10.4%

Figure 8. Android versions I



Data collected during a 7-day period ending on May 7, 2019.  
Any versions with less than 0.1% distribution are not shown.

Figure 9. Android versions 2

The IDE (Integrated Development Environment) selected for the development of the application is **Android Studio**. It is the most important IDE for the development of Android applications. This option is totally free and can run on Windows 10. It is also supported by a huge community of developers.



Figure 10. Android Studio

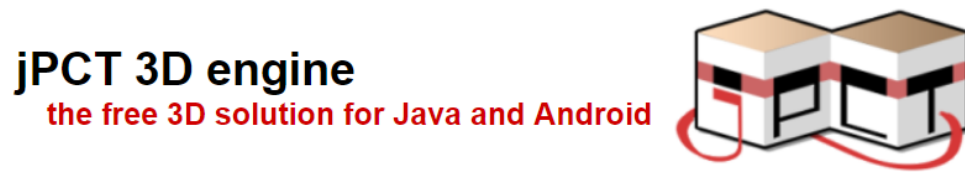
The programming language in the development of the application code is **Java**. There are two reasons to select this option. From one side is a very powerful programming language and from the other side it is one of the native languages to develop Android applications. As the most popular language in the development of Android applications, Java benefits a huge supporting community.



Figure 11. Java



The library chosen for developing the 3D environment of the spaces is **JPCT**. This library is compatible with Android Studio.



*Figure 12. JPCT 3D engine*

The physical device used to install the application is the smartphone Xiaomi mi A1. It is my own mobile device. It is a smartphone made by the Chinese company Xiaomi. Xiaomi mi A1 runs with Android One OS in his Android 9 Pie version.



*Figure 13. Xiaomi mi A1*

## 4 Android

### 4.1 History and features

In 2003 Android Inc company was built by Andy Rubin, Rich Miner, Nick Sears and Chris White with the aim to develop mobile devices which a knowledge of position and user preferences. Later in 2005 Google bought Android Inc and started to develop a mobile platform based on the Kernel of Linux. It was only in 2008 that the first version of the operating system of Android called Android 1.0 Apple Pie was commercialized.

Nowadays there is a huge number of versions and the last one was launched last August 2019 and is called Android 10.0 or Android Q.

Several operating systems exist but the difference between Android and the others is based on Linux, a totally free and multi-platform nuclear operating system.



*Figure 14. Android devices*

For the other side Android allows to programming applications in Java through the virtual engine Dalvik. That is why Android provides necessary interfaces to develop applications with smart device functions (sensors, GPS, agenda).

This operating system has a significant penetration in the international market as a worldwide leading OS. The following graphic shows the evolution of the trimestral sales of mobile devices separated by operating systems from 2007 to 2018.

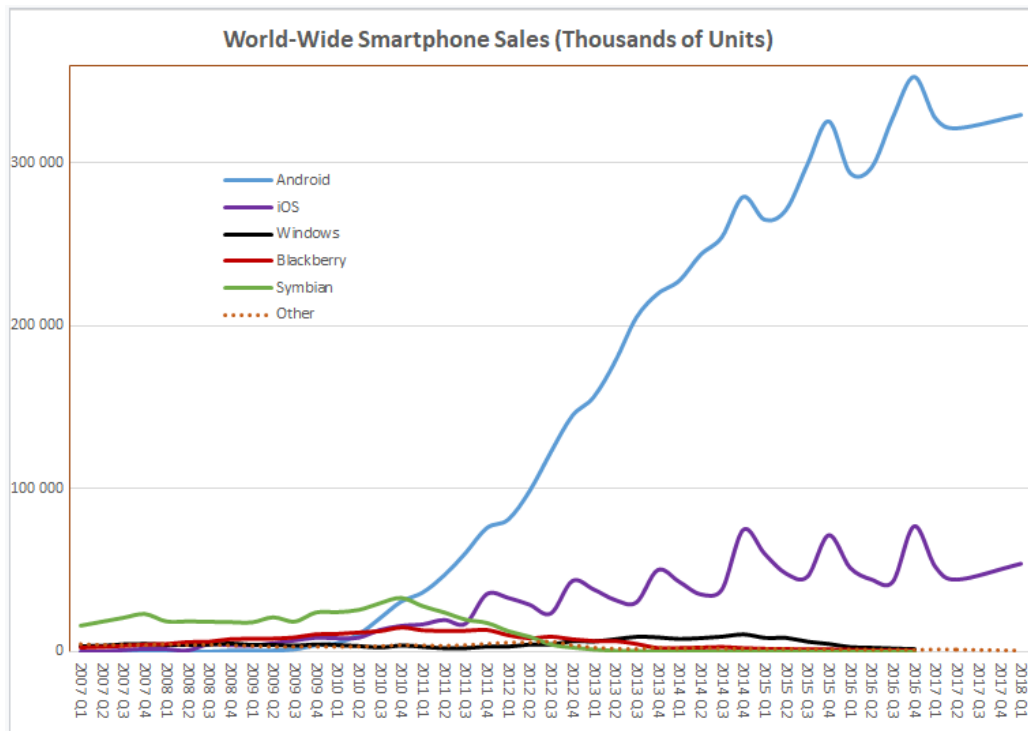


Figure 15. Market OS distribution. Information source: Gartner Inc.

It is important to underline that Android has a big community behind the system and support all the professionals through forums, blogs, websites and meetups.

Android has converted the smartphone as a device able to make high level photos, to be informed of the position of the user by GPS coverage, to make NFC (Near Field Communications) payments, to consult internet sites or to enjoy an experience based on virtual reality and so much more. Android also provides an IDE called Android Studio.



Figure 16. Android experience

## 4.2 Android Studio

Android Studio is the official integrated environment of development for Android platform. It was announced on May 16 of 2013 and replaced the Eclipse as the official IDE for the development of applications for Android. The first stable version was published in December 2014. Now the version 3.5 of Android Studio is available. Following the graphical presentation of the IDE.

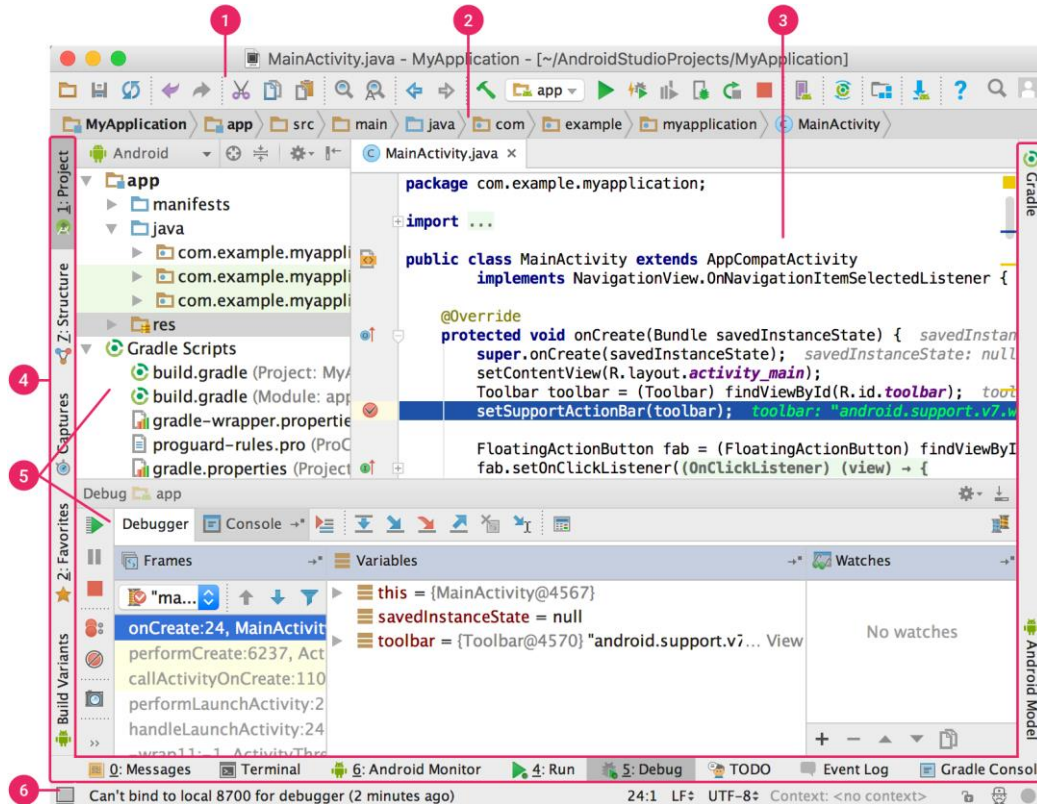


Figure 17. Android Studio interface

It is important to configure in the correct format the Android Manifest. It is an XML file that contains the description of all the Android application characteristics. Including the minimal version used for Android, the permissions, the building blocks and the presentation configuration (landscape or portrait presentation).

## **5 Java**

The selection of Java programming language has two factors:

1. It is one of the Official Android languages.
2. There is a considerable support community.

### **5.1 History and features**

TIOBE Quality Indicator presented Java as the most used programming language (object oriented) known in the world. Since 1995, it has been commercialized by Sun Microsystems (today Oracle). A number of applications and websites have been built in Java for his reliability, security and speed.

Java is deployed in a large number of devices: from laptops to data centers, from gaming consoles to supercomputers, from mobile devices to Internet.

The reason why Java is used on multiple devices is due to the fact that Java applications are compiled into bytecode (Java class), which can execute in any Java (JVM) virtual engine regardless of the computer architecture.

As key figures:

- 97% of corporation desktop run Java
- More than 9 million of Java developers worldwide
- More than 3000 million mobile phones execute Java
- 5000 million of Java cards activated
- 125 million of TV devices execute Java

Java has the support of a large community of software developers.

### **5.2 Java API**

API means application programing interface. It is basically a tool set that programmers can use to develop software. API of Java is a website where is published a list of all the classes that are part of the Java development kit (JDK). It includes all Java packages, classes, and interfaces, along with their methods, fields, and constructors. These prewritten classes provide a tremendous amount of functionality to a programmer. Following a screenshot of the website.

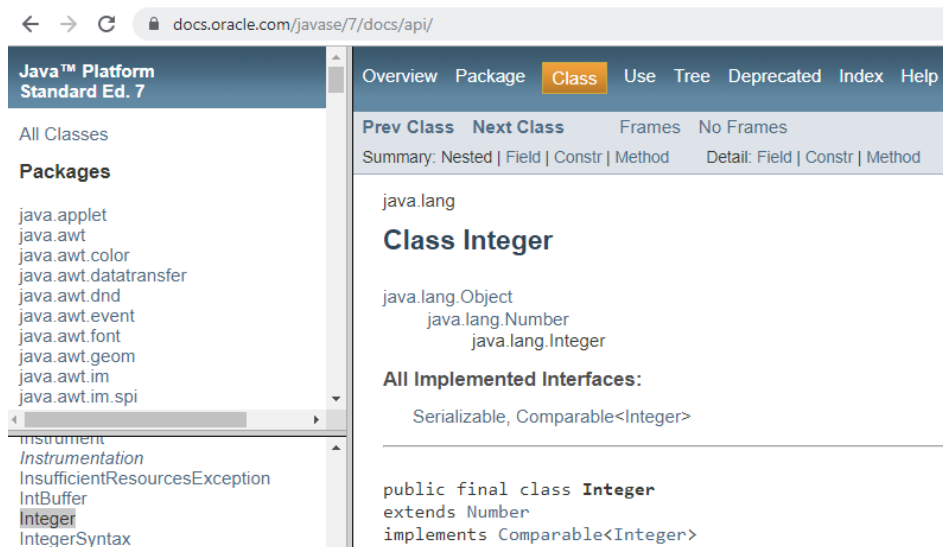


Figure 18. Java API

The current API of Java is in the edition 7 and is called “Java™ Platform, Standard Edition 7 API Specification”.

### 5.3 Java - TIOBE Index

The TIOBE Programming Community index is an indicator of the popularity of programming languages. The index is updated once a month. The ratings are based on the number of skilled engineers world-wide, courses and third-party vendors. Popular search engines such as Google, Bing, Yahoo!, Wikipedia, Amazon, YouTube and Baidu are used to calculate the ratings. It is important to note that the TIOBE index is not about the best programming language or the language in which most lines of code have been written. TIOBE Index for August 2019 described Java as the programming Language more popular.

Aug 2019	Aug 2018	Change	Programming Language	Ratings	Change
1	1		Java	16.028%	-0.85%
2	2		C	15.154%	+0.19%
3	4	▲	Python	10.020%	+3.03%
4	3	▼	C++	6.057%	-1.41%
5	6	▲	C#	3.842%	+0.30%

Figure 19. TIOBE index 2019

## 6 Augmented Reality & JPCT 3D engine

Augmented reality allows to combine real-world elements with virtual environment in real time. This capacity to overlay virtual information on top of the real-world through our smartphones pretends to create an added value experience to the user.

### 6.1 History

While developing one of the most famous planes “Boeing 747” in 1992, the scientist and researcher Thomas P. Caudell faced a manufacturing problem. The workers who assembled the different aircraft parts spent too much time trying to understand the instructions. He worked on a solution which provides information on a screen in order to guide the workers during the installation process.

### 6.2 Applications

Augmented reality has seen an increasing impact in important in almost all the sectors:

- **Industry:** Receiving instructions during an installation
- **Advertising:** Visualization of Outdoor Advertising Campaign
- **Turism:** Orientate a tourist providing relevant information
- **Gaming:** “PokemonGo”
- **Museums:** Giving life to the masterpieces.
- **Medical:** Visualization of patient information.
- **Architecture:** Visualization of the alterations in an apartment.
- **Sports:** The sporty person can explore new paths and visualize relevant information.
- **Academic:** Show to the student information in an interactive way.
- **Automotive:** Show route to the conductor using GPS.



Figure 20. AR: industry applications

An important group of companies works currently with AR and Android mobile technologies. To only mentioned a few of them: Amaxperteye, Wizzan, Icarus, VMware, Ubimax, Picavi, Eyesucceed, Hodei Technology, Chironix, Cvision. Most of the companies mentioned have their headquarters in the US.

### **6.3 JPCT 3D engine**

JPCT is a 3D engine is a library for desktop Java and Google's Android. It will work on Windows, Linux, Mac Os X, Solaris x86 and on Android mobile phone or Tablet. It supports OpenGL via LWJGL and JOGL and uses OpenGL ES 1.x and ES 2.0 on Android.

JPCT for Android called JPCT-AE has an API which offers to programmers all the features they need to code for mobile 3D games, simulation application for Google's Android platform.

The main Features:

- It is optimized for the Android mobile platform
- Supports **OpenGL** ES 1.x and 2.0 on Android 1.5 or higher
- Render to texture
- Build-in primitives like cones, cubes, spheres...
- Transparency and fog effects
- On a Tablet or smartphone

OpenGL (Open Graphics Library) is a specific standard who defines a multilingual and multi-platform API to describe applications that produce 2D and 3D graphics.



## 7 Indoor Localization System

### 7.1 History and features

Since 2000 the Geolocation systems are present in most mobile devices thanks to GNSS (Global Navigation Satellite System), of which the best known is the GPS.

Now, these systems have been adapted to the mobile devices and guide us in the car, by foot or on bike, as city or country side. Nevertheless, some zone still resisting to the positioning technology, especially indoor spaces, most known in the scientific area as indoor environment. In these environments the GPS fails and that is the reason why it is necessary other alternatives.

GPS technology is not working in indoor environments for these following reasons:

- The satellite signal is not able to reach the necessary intensity in indoor space
- In case of the satellite signal reaches, the building maps should be public and in mobile device format

It for these reasons that another type of indoor positioning is used as previously commented in the State-of-the-Art section: Indoor Positioning Systems (IPS). The following figure illustrates the different concepts in both technics GPS vs. IPS.



Figure 21. GPS vs. IPS

Nowadays, the next indoor positioning technologies are used:

### Bluetooth

These systems have relied on the use of received signal strength (RSSI) measurements to estimate the distance between Bluetooth devices that are part of the system. Using this technique, positioning systems can achieve meter-level accuracy when determining the location

of a specific device. As of Bluetooth 5.1, since angle to the antenna can be measured, it can be used to determine inside position accurate to centimeter level.

### Inertial Positioning systems

This technique works thanks to Inertial Measurement Units (IMU) such as accelerometer, gyroscope and geomagnetic field sensor in smartphone. Although the precision of Inertial Navigation System (INS) is limited due to the drift of IMU along the time as well as cumulative errors, no external signals are necessary, and this robustness and invulnerability make it a powerful complementary to others indoor positioning technologies in short term.

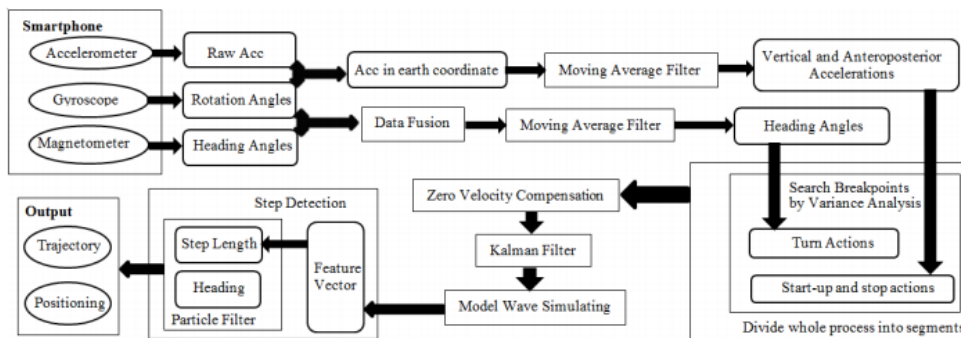


Figure 22. Inertial positioning system flow diagram

### Wifi positioning systems

The most common and widespread localization technique used for positioning with wireless access points is based on measuring the intensity of the received signal (received signal strength indication or RSSI) and the method of "fingerprinting". The accuracy depends on the number of nearby access points whose positions have been entered into the database.

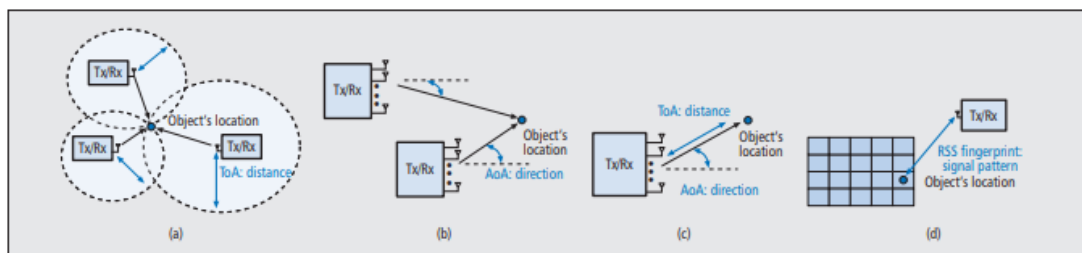


Figure 23. a)Time of arrival approach; b)angle of arrival approach; c)hybrid ToA/AoA approach; d)received signal strength and fingerprint approach

They can be combined with different technologies to reach a better positioning capacity. The possible solution is introduced in the following figure.

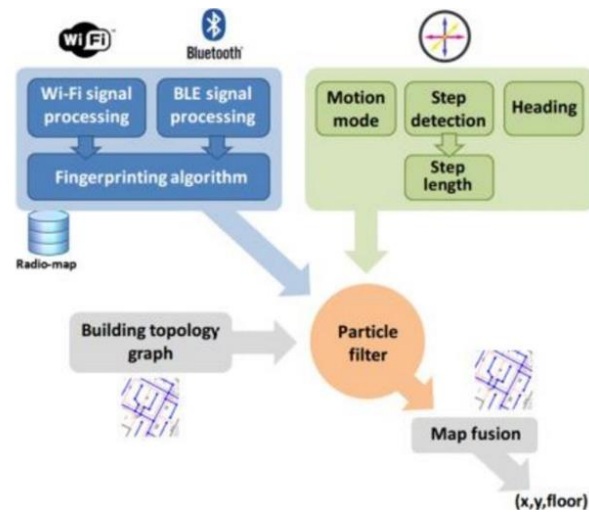


Figure 24. IPS techniques combined

## 7.2 Applications

The IPS applications are very extended. It will depend on the goal that is given to them. These following useful applications stand out:

- To control crowded spaces
- To localize someone into a space
- To show personalized advertising according to the user

Outstanding sectors:

Marketing	Space management
Sport and Music events	Industry
Education sector	Health
Hotel industry	Augmented reality
Retail	Aviation management

## 8 Planning and resources

### 8.1 Planning - Gantt Chart

The first step to develop the project is identifying the stages and the planning as resources and staff as in work packages called WP with the time perspective.

All the stages must be marked in the Gantt chart to have a better perspective. This chart is managed by the project manager and he will be responsible for it updating and identify all the eventual risks.

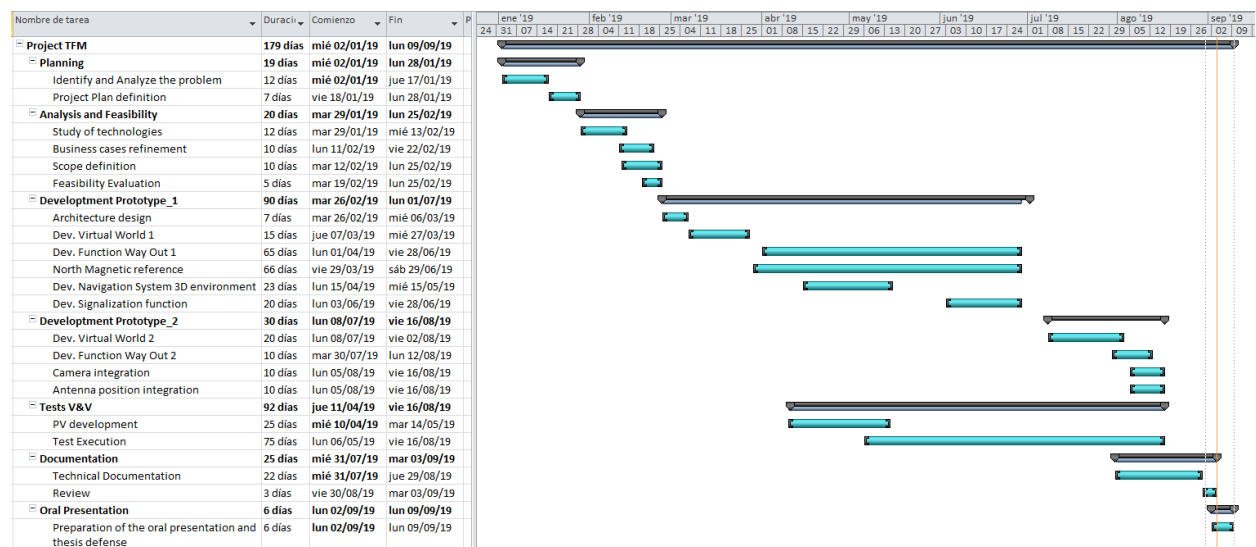


Figure 25. Gantt chart

#### WP\_1: Planning

Identify the needs and develop the project

Definition of a Project plan

Gantt Chart

#### WP\_2: Analysis and Feasibility

Analysis of the state of art of the components and technologies to understand technical concepts

Scope definition

Product Requirements

Business cases refinement

Risk evaluation

Feasibility report

### **WP\_3: Development prototype\_1**

Architecture design

Application development code

Build Android application

### **WP\_4: Development prototype\_2**

### **WP\_5: V&V - Tests**

Define a collection of tests to verify the correct construction of the application

Define collection of tests to validate that the application performs as is expected

Integration Test Report, System Test Report, SW Test Report, Acceptance Test Report.

### **WP\_6: Documentation**

Technical documentation

Review

### **WP\_7: Oral Presentation**

Preparation of the oral presentation

Thesis defense

## **8.2 *Costs and resources***

### **RESOURCES**

To calculate the costs of the project for developing the Prototype\_1 the following resources are required.

#### **SW Tools**

Android Studio

Developer account on Google Play

#### **HW Tools**

Android Smartphone

Windows Computers

Indoor Antennas position Systems

## Human Resources

Android developer

**COSTS** - The project cost is described in the following table.

Type	Concept	Cost	Period	Total Cost
Sw	Android Studio	-	-	-
Sw	Google Play acc.	25 €	one-time	25 €
Hw	Android smartphone	200 €	one-time	200 €
Hw	Windows computer	600 €	one-time	600 €
Hw	Indoor Antennas System	-	-	-
Hm	Android Developer	25 €/h	18 weeks	11250 €
-	-	-	-	<b>12075 €</b>

*Table 1. Project costs*

Indoor antennas Systems services are provided by an external company.

Android developer has a cost 25€/hour and the work day is of 5 hours.

## 9 Analysis

As it was mentioned in the section 1.2 the project was divided in two prototypes. For each one of them it will be defined their requirements for defining the scope of the project. Normally in a SW project the requirements are developed by the product owner or the responsible of each department (SW, HW, System). The collection of requirements is described next.

### PROTOTYPE 1

1. First prototype - virtual solution to verify the viability of the application
  - 1.1 Development of a virtual map for the building prototype \_1
  - 1.2 Development of a navigation system in 3D virtual environment
  - 1.3 Development of a reference system for the virtual and real-world environments
  - 1.4 Development of the Way-Out function \_1
  - 1.5 Development of the signalization functions
  - 1.6 Test of prototype\_1

#### Requirements - Prototype\_1:

- Req\_1. The app shall be support for minimum Android API level 19 (kitkat)
- Req\_2. The app shall be available in .apk format
- Req\_3. The app can be installed in any smartphone with Android version 4.4 or greater (until Android 9.0)
- Req\_4. The name of the app is “TFM\_Prototype1\_VX.Y”. The nomenclature X.Y is the reference for the version control of the app (X-major, Y-minor)
- Req\_5. The app shall work on smartphones with the screen resolution FullHD (1,920 x 1,080 pixels) or FullHD+ (2.340 x 1.080 pixels)
- Req\_6. The .apk file shall be less than 10 MBytes
- Req\_7. The device shall have at least the next sensors: accelerometer, gyroscope and geomagnetic field sensor
- Req\_8. The device shall have a maximum weight of 200gr
- Req\_9. The device shall be equal or greater than 5 inches
- Req\_10. The device shall have Android 4.4 kitkat (or greater)
- Req\_11. The device shall be able to process the rendering images
- Req\_12. The devices shall be able to do screen refresh at least 25 frames/second in order to have a fluid vision experience
- Req\_13. The app shall perform as described in the Workflow “App Workflow diagram”

Req\_14. The app shall update the direction of the camera when rotation movement on Y-axis is detected.

Req\_15. The app shall update the position camera when some touch screen movement is detected.

Req\_16. The app shall perform lateral and lineal movements of the camera.

Req\_17. The app shall introduce the virtual world on screen orientation “landscape”

Req\_18. The app shall set the camera 90° from the smartphone magnetic heading

## **PROTOTYPE 2**

2. Second prototype - integration of prototype\_1 to the physical-real world

2.1 Development of a virtual map of the real building prototype\_2

2.2 Development of the Way-Out function\_2

2.3 Development of the camera function to merge the virtual and real- world views

2.4 Development of data acquisition system and adaptation to the indoor positioning

2.5 Test of prototype\_2

### **Requirements - Prototype\_2:**

All the requirements of Prototypes\_1 are applied to the Prototype\_2 and the next ones.

Req\_19. The device shall have rear camera

Req\_20. The device shall have Bluetooth connectivity

Req\_21. The device shall have Wifi connectivity

Req\_22. The device shall have 3G/4G connectivity

Req\_23. The device shall connect with the server to get the position

Req\_24. The app shall update the position in less than Xms (TBD) after the position was received

Req\_26. The app shall have a virtual map with a scale 1:1 according real-world environment



## 10 Application Development

### 10.1 Workflows and algorithms

The next diagram shows the logical sequence used on the app.

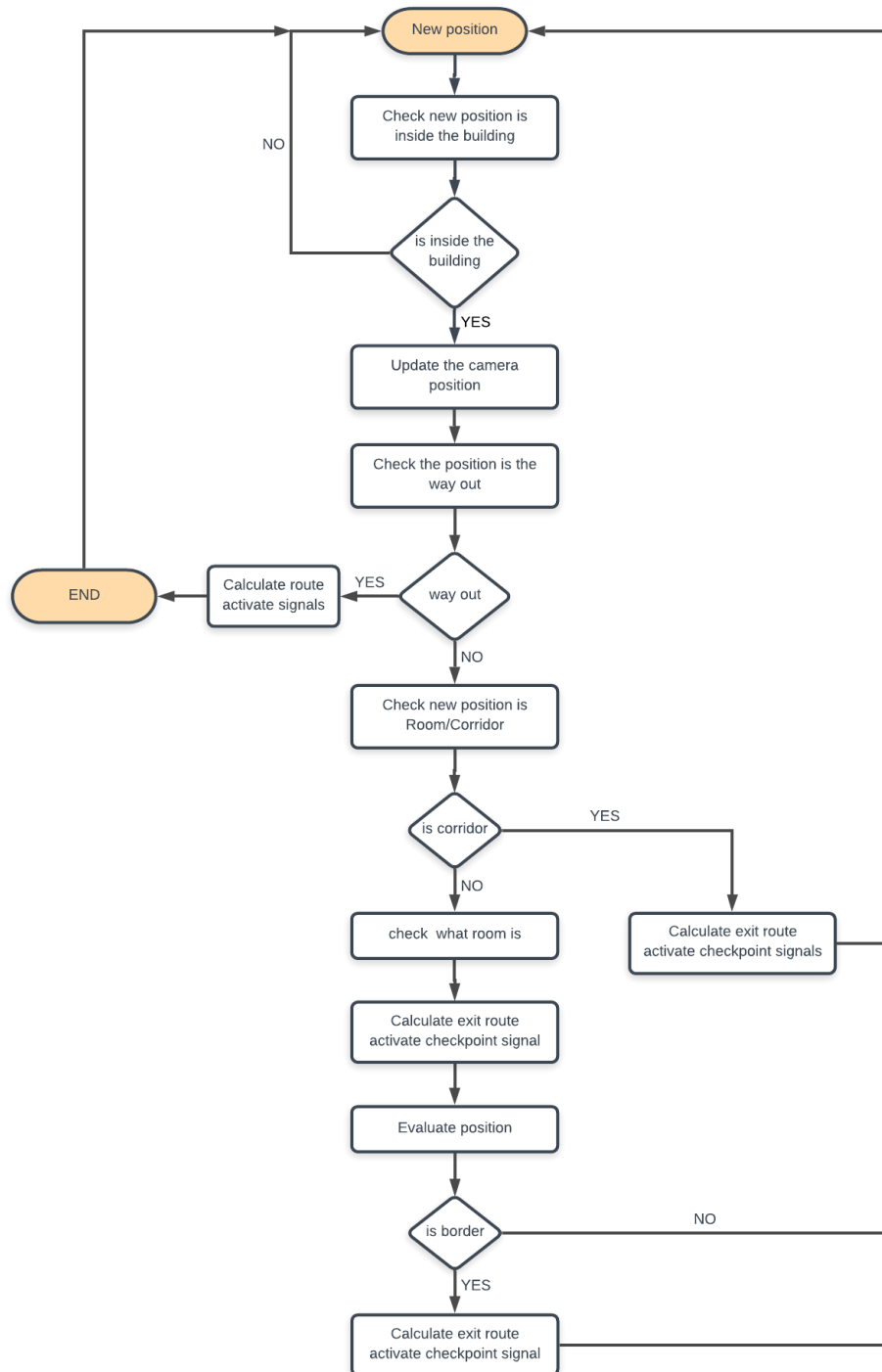


Figure 26. Logical workflow diagram

The workflow defined previously is usable for both subproject (prototype\_1 and prototype\_2).

The aim of this workflow is based on the evaluation of every new point entered into the system

in order to determine the position in the building (“WayOut”, “Room” or “Corridor”). In consequence the logical of the app modifies the checkpoints signals features (visibility or texture).

The idea is to add a series of visual elements depending on the user's position. The logical workflow is built for the prototype\_1 but it can serve for the prototype\_2. Although some modification will be necessary for the introduction of a logical part focused on the management of the position signal provided by the indoor positioning system.

**Important:** From this point we will focus on the development of the solution based on the prototype\_1.

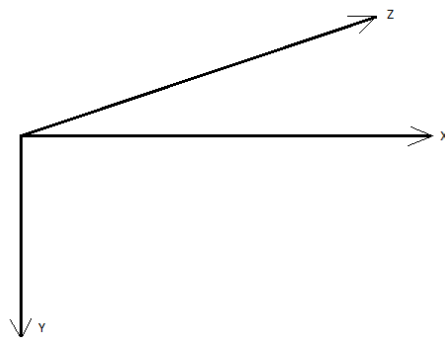
## ***10.2 Virtual World***

To guide the user inside of the building is necessary to develop the signals that will serve as reference points. These signals are situated in the space included into the building, therefore a virtual world must be built.

The virtual world is the virtual recreation of the inside of the building. It is mandatory to create all the elements to build the virtual world as walls, floors, doors, stairs and distributing them properly inside the world.

The construction of the virtual world is possible thanks to the 3D JPCT library. This library has an API where the necessary basic elements are described to build the world.

The axis reference of JPCT is visualize as below.



*Figure 27. Virtual reference axis*

The classes used to develop the virtual world are shown in the next table.

<b>World</b>	Define the space where the elements will be added
<b>FrameBuffer</b>	Provide a buffer into which JPCT renders the scene
<b>Light</b>	Allow to handle the light source
<b>Object3D</b>	Allow to create 3D objects
<b>Primitives</b>	Allow to create some predefined 3D objects
<b>ExtendedPrimitives</b>	Provide some methods to create basic 3D objects that can be used within a JPCT world
<b>SimpleVector</b>	Represent a basic three-dimensional vector (used by Object3D and ExtendedPrimitives)

Table 2. JPCT Classes

The next step is the IDE configuration dependencies for Android Studio to recognize the JPCT library classes.

Then, the “build.gradle” file in dependencies section has to create a reference to the file “libs/jpct\_ae.jar”. This is the way for Android Studio to allow developers using the JPCT library.

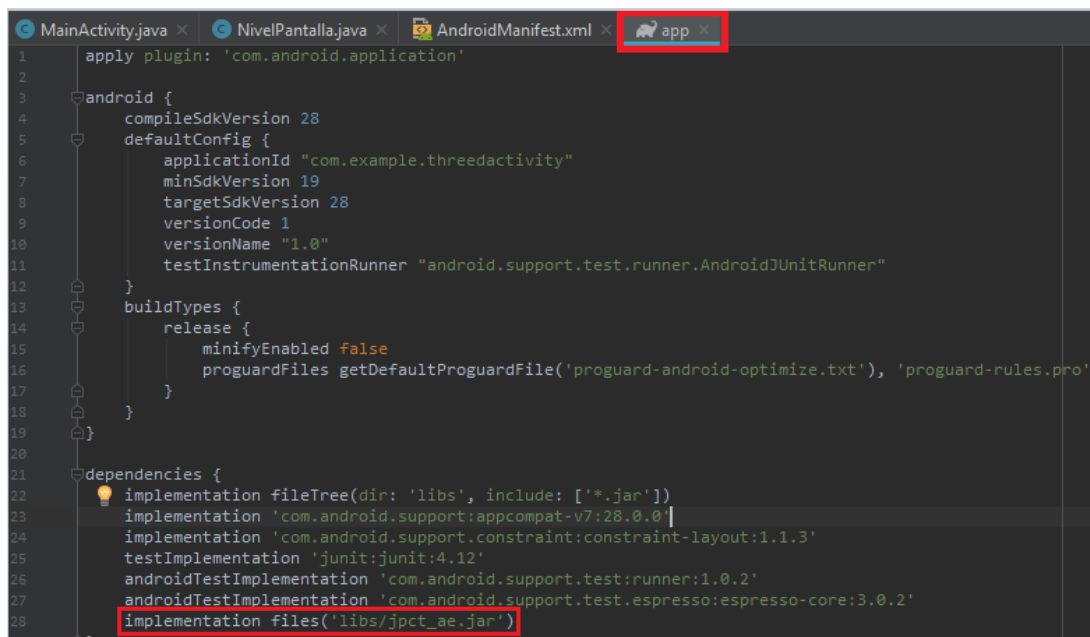


Figure 28. Android Studio dependencies

To create the building is necessary to follow the next steps:

1. Create a world and a light source
2. Create objects (define the size and give them texture, move them to a position)
3. Add objects to the world

To define the objects, it must set some features: height, thickness, length.

To define the texture, we must instantiate the texture class and modify their features. New textures can be defined as pictures or defined textures can be represented with color (red, blue, green...).

To move an object to an exact position into the world is necessary to have a reference in the central position of the object and it will move according to the reference point of the coordinate axis. For further information about connected data with the development of the virtual world is recommended to go to the [Annex](#) section where the application code is.

The developed world for this prototype\_1 is shown in the following figures. It is an approximate representation of a segment of offices floor of the engineer university UAB.

It is composed by a hallway and 6 rooms, 3 on each side of the hallway. The elements created are walls, floor and signals elements.

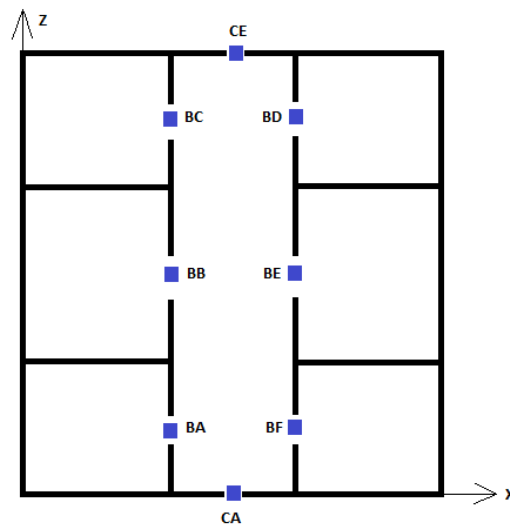


Figure 29. Virtual World map

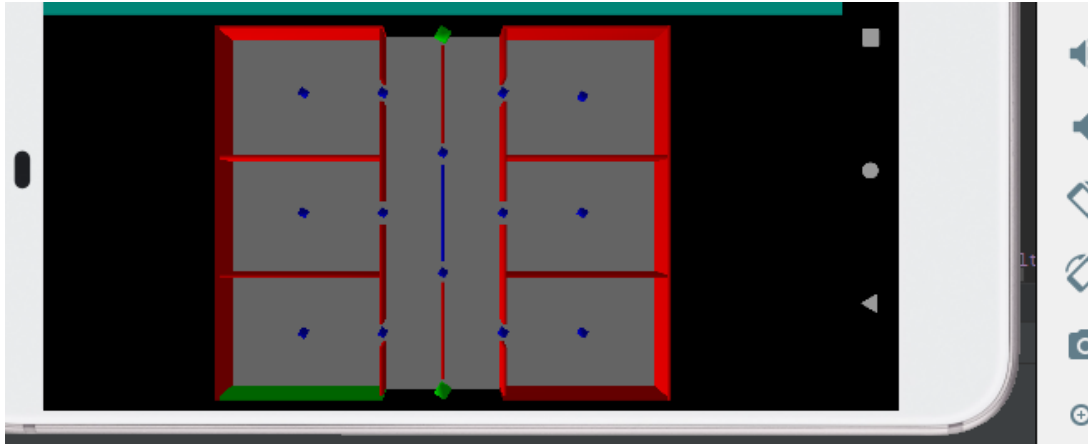


Figure 30. Virtual World view

These screenshots as below show different perspective of the virtual world developed.

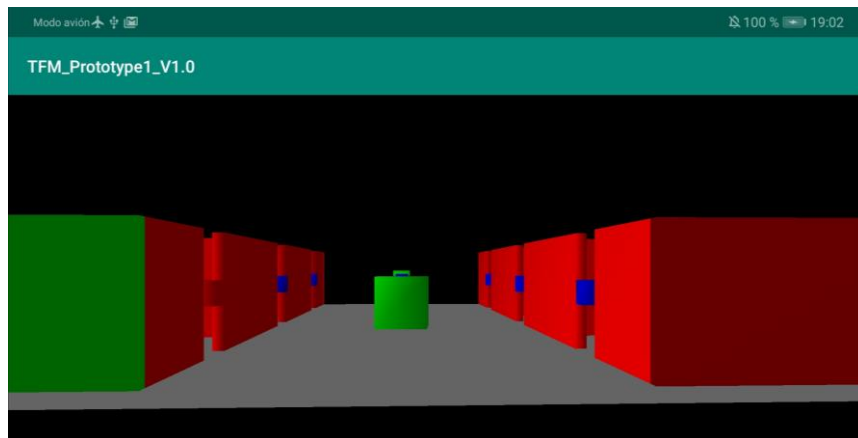


Figure 31. Virtual World 1

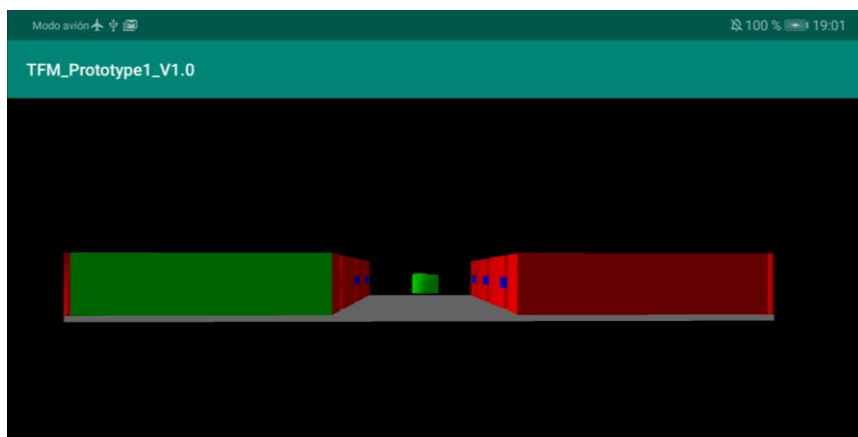


Figure 32. Virtual World 2

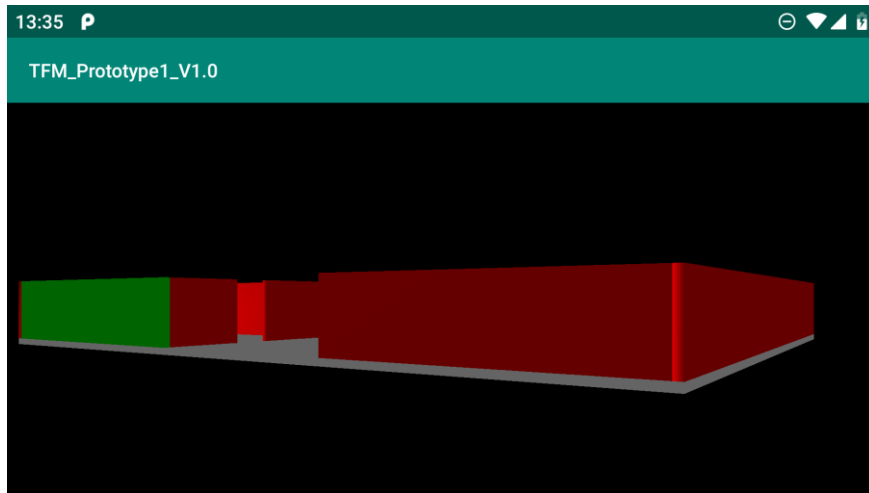


Figure 33. Virtual World 3

**Note:** The wall with the green texture is a reference. One of the green wall's vertex meets with the point (0,0,0). This fact makes easy the test process for the developer.

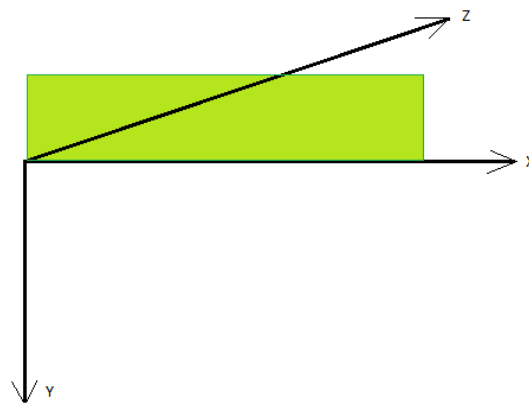
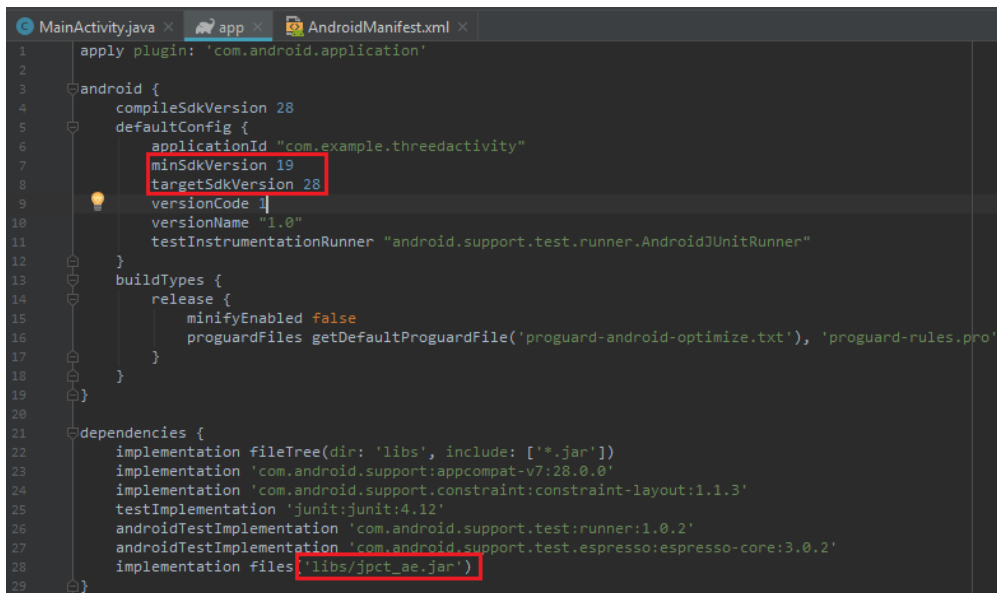


Figure 34. Virtual World axis

### 10.3 Development

Before starting the code development is mandatory to make a correct Android Studio configuration.

1. Set up the Android version: minSdkVersion:19 and targetSdkVersion:28
2. Set up the dependencies (libs\jpcet.jar) as explained in the last section 10.2



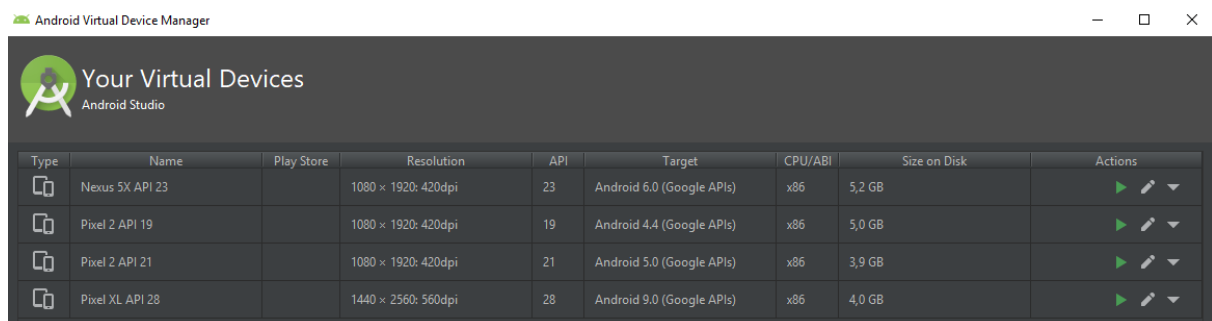
```

1  apply plugin: 'com.android.application'
2
3  android {
4      compileSdkVersion 28
5      defaultConfig {
6          applicationId "com.example.threedactivity"
7          minSdkVersion 19
8          targetSdkVersion 28
9          versionCode 1
10         versionName "1.0"
11         testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
12     }
13     buildTypes {
14         release {
15             minifyEnabled false
16             proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'
17         }
18     }
19 }
20
21 dependencies {
22     implementation fileTree(dir: 'libs', include: ['*.jar'])
23     implementation 'com.android.support:appcompat-v7:28.0.0'
24     implementation 'com.android.support.constraint:constraint-layout:1.1.3'
25     testImplementation 'junit:junit:4.12'
26     androidTestImplementation 'com.android.support.test:runner:1.0.2'
27     androidTestImplementation 'com.android.support.test.espresso:espresso-core:3.0.2'
28     implementation files('libs/jpct_ae.jar')
29 }

```

Figure 35. Android Studio configuration

It is possible to emulate a smartphone using the Android Studio emulator. Android Virtual Device Manager allows the developer to use the smartphone desired. Although the emulator has some limitations. For this reason, is a good option to install the apk file in a real smartphone, especially for heavy computation applications.



Android Virtual Device Manager

Your Virtual Devices  
Android Studio

Type	Name	Play Store	Resolution	API	Target	CPU/ABI	Size on Disk	Actions
	Nexus 5X API 23		1080 × 1920: 420dpi	23	Android 6.0 (Google APIs)	x86	5,2 GB	
	Pixel 2 API 19		1080 × 1920: 420dpi	19	Android 4.4 (Google APIs)	x86	5,0 GB	
	Pixel 2 API 21		1080 × 1920: 420dpi	21	Android 5.0 (Google APIs)	x86	3,9 GB	
	Pixel XL API 28		1440 × 2560: 560dpi	28	Android 9.0 (Google APIs)	x86	4,0 GB	

Figure 36. Android Virtual Device Manager

In the AndroidManifest.xml file is possible to modify the application name changing the value name of “android:label”. Then the application will show the name on the screen when it has been executed in the smartphone.

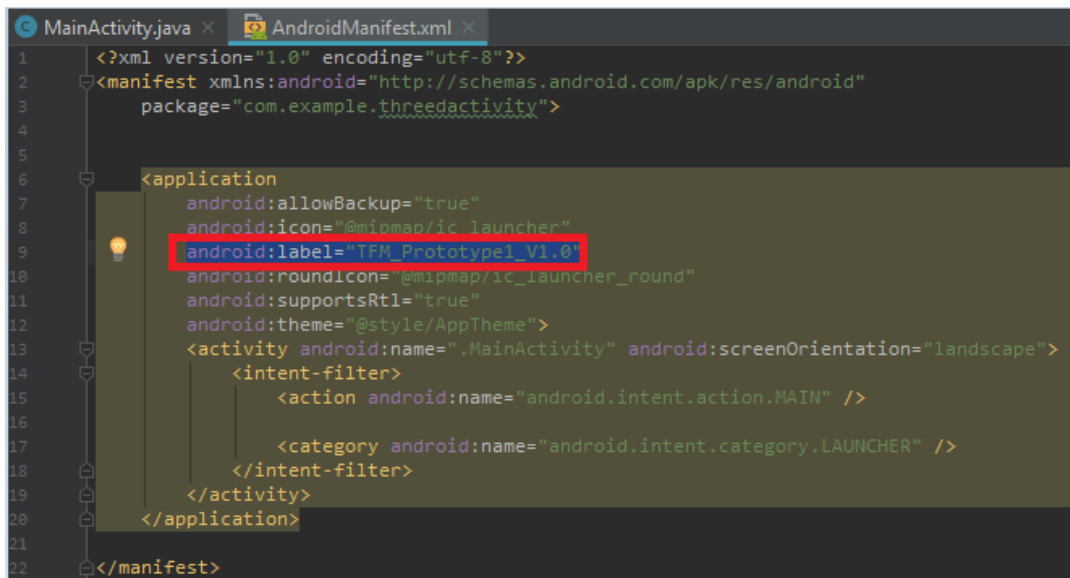


Figure 37. Android Manifest - App name

## 10.4 Functions

This sub chapter is the most important part of the project. The functions and methods developed in the code app were described next.

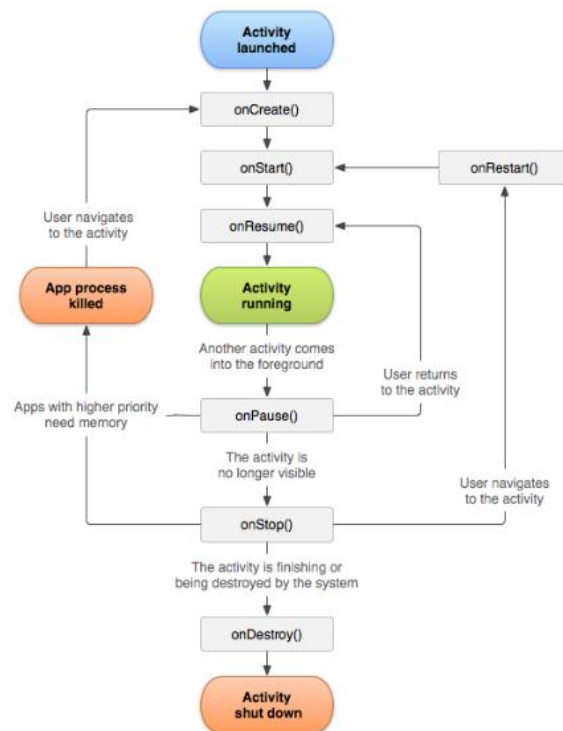
### Activity

The “Activities” are necessary to visualize on the screen the application content. Each screen shown by the application is an Activity. That means if an application has three screens, the application has 3 Activities. The activities are built into two parts: The logical and the graphical. The **graphical** part is a file XML which contains all the elements shown on the screen.

The activities accomplish a lifecycle. The activity class provides a core set of lifecycle callbacks: onCreate(), onStart(), onResume(), onPause(), onStop(), and onDestroy().

The system invokes each of these callbacks as an activity enters a new state. The workflow of these states is described next.





Then, the programmer shall develop some of these methods on the code application. For more information check the code in the [Annex](#) section and [here](#).

The **logical** part is a file with .java format. In this file is where the application Java code is and where the main functions are developed.

## Import

All the packages imported are mentioned in this section

```

1 package com.example.threedactivity;
2
3 import android.graphics.Color;
4 import android.hardware.Sensor;
5 import android.hardware.SensorEvent;
6 import android.hardware.SensorEventListener;
7 import android.hardware.SensorManager;
8 import android.opengl.GLSurfaceView;
9 import android.os.Bundle;
10 import android.support.v7.app.AppCompatActivity;
11 import android.view.MotionEvent;
12 import android.view.View;
13
14 import com.threed.jpct.Camera;
15 import com.threed.jpct.FrameBuffer;
16 import com.threed.jpct.Light;
17 import com.threed.jpct.Logger;
18 import com.threed.jpct.Object3D;
19 import com.threed.jpct.Polyline;
20 import com.threed.jpct.Primitives;
21 import com.threed.jpct.RGBColor;
22 import com.threed.jpct.SimpleVector;
23 import com.threed.jpct.Texture;
24 import com.threed.jpct.TextureManager;
25 import com.threed.jpct.World;
26 import com.threed.jpct.util.ExtendedPrimitives;
27
28 import javax.microedition.khronos.egl.EGLConfig;
29 import javax.microedition.khronos.opengles.GL10;
  
```

## MainActivity class

It is the class instantiated to execute the application.

```
31 public class MainActivity extends AppCompatActivity implements GLSurfaceView.Renderer, SensorEventListener, View.OnTouchListener {
32
33     private GLSurfaceView glView;
34
35     private World world;
36     private FrameBuffer fb;
37     private Light light;
```

## onCreate

Called when the activity is starting.

```
150 @Override
151 protected void onCreate(Bundle savedInstanceState) {
152     super.onCreate(savedInstanceState);
153
154     // Get an instance of the SensorManager
155     miManager=(SensorManager) getSystemService(SENSOR_SERVICE);
156
157     // Create our Preview view and set it as the content of our Activity
158     glView = new GLSurfaceView( context: this);
159     glView.setEGLContextClientVersion(2);
160     glView.setRenderer(this);
161     setContentView(glView);
162
163     glView.setOnTouchListener(this);
```

## onResume

State in which the application interacts with the user. The SensorManger features are coded (type of sensor, sampling period).

```
190 @Override
191 protected void onResume() {
192     super.onResume();
193     glView.onResume();
194
195     miManager.registerListener( listener: this, miSensorPr, samplingPeriodUs: 100000);
196     miManager.registerListener( listener: this, miSensorAc, samplingPeriodUs: 100000);
197     miManager.registerListener( listener: this, miSensorGi, samplingPeriodUs: 100000);
198     miManager.registerListener( listener: this, miSensorMa, samplingPeriodUs: 100000);
199 }
```

## onPause

It is called when the user no longer actively interacts with the Activity, but it is still visible on screen.

```
202 @Override
203 protected void onPause() {
204     glView.onPause();
205     super.onPause();
206     miManager.unregisterListener(this);
207 }
```

## onSurfaceCreated

Here is where the world, light and camera are instantiated. The textures used are also defined. The objects are added to the world. They can also be positioned in the virtual environment. Some objects features are settled.

```
210      @Override
211      public void onSurfaceCreated(GL10 gl, EGLConfig config) {
212
213          // INSTANCE OF WORLD CLASS - CAMERA CLASS - TEXTURE CLASS
214          world = new World();
215
216          Camera a = world.getCamera();
217
218          a.setPosition( (x: 57, (y: -7, (z: -25);
219
220          a.rotateCameraY((float) Math.toRadians(0));
221
222          //TEXTURES
223          Texture t1 = new Texture( width: 64, height: 64, RGBColor.BLUE);
224          TextureManager.getInstance().addTexture( name: "t1", t1);
225          Texture t2 = new Texture( width: 64, height: 64, RGBColor.RED);
226          TextureManager.getInstance().addTexture( name: "t2", t2);
227          Texture t3 = new Texture( width: 64, height: 64, RGBColor.WHITE);
228          TextureManager.getInstance().addTexture( name: "t3", t3);
229          Texture t4 = new Texture( width: 64, height: 64, RGBColor.GREEN);
230          TextureManager.getInstance().addTexture( name: "t4", t4);
231
232
233          // ***** ROOM POINTS *****
234
235          RA = Primitives.getCube( scale: 1f);
236          RA.translate(PRA);
237          RA.setTexture("t1");
```

Figure 38. onSurfaceCreated

## onSurfaceChanged

This sets up the FrameBuffer and the World. Called after the surface is created and whenever the OpenGL ES surface size changes.

```
444      @Override
445      public void onSurfaceChanged(GL10 gl, int width, int height) {
446          fb = new FrameBuffer(width, height);
447      }
448
```

Figure 39. onSurfaceChanged

## onDrawFrame

This is the render method, called by Android in the render thread.

```
450      @Override
451      public void onDrawFrame(GL10 gl) {
452          fb.clear();
453          //
454
455          // CHECKPOINTS ROTATION
```

Figure 40. onDrawFrame

## onTouch

Intercept the touch events in an Activity or a View.

```
private float lastX, lastY; // Aux.Var. to keep the last position

@Override
public boolean onTouch(View v, MotionEvent event) {

    touch_click++;
```

Figure 41. onTouch

## IsInWorld

This function evaluates for any new position if it is inside the area defined as building. As a result, return a true (is in the area) or false (is not in the area).

```
623 // ***** Check whether new point is inside the world *****
624
625 public boolean IsInWorld(float x, float y, float z){
626
627     if((x <= (ground_x + world_error) && x >= (-world_error)) && (y >= (-wall_height) && y <= -1) && (z >= (-world_error) && z <= (ground_z + world_error)))
628     {
629         return true;
630     }
631     else{
632         return false;
633     }
634 }
635 }
```

Figure 42. IsInWorld

## PointsInvisibles

PointsInvisible sets the feature “setVisibility” to false for all checkpoint signals. That means, all the checkpoints signals become invisibles.

```
628 public void PointsInvisible()
629 {
630     BA.setVisibility(false);
631     BB.setVisibility(false);
632     BC.setVisibility(false);
633     BD.setVisibility(false);
634     BE.setVisibility(false);
635     BF.setVisibility(false);
```

Figure 43. PointsInvisibles

## IsWayOut

Evaluate for any new position if is the Way out or not. In consequence the function modifies the features of some checkpoint objects. The checkpoint marks as way out change his texture to green color. All the next checkpoint objects become invisibles.

```
652 public boolean IsWayOut(float x, float y, float z){
653
654     if(WayOut == 0 ){ // Way Out is Point CA
655
656         if((x < (30 + wall_width + wall_long41)) && ( x > (wall_width -
657         {
658
659             PointsInvisible();
660             CA.setTexture("t4"); //GREEN
661             CA.setVisibility(true);
662
663             return true;
664         }
665     }
```

Figure 44. IsWayOut

## CalcPoint\_RoomCorr

Check if the current position is in some room area or in the corridor area.

```
680 //**** Calculate the current position is room or corridor ****
681
682 public int CalcPoint_RoomCorr(float pos_x, float pos_y, float pos_z){
683
684     // Check if position is in the corridor surface
685     if( (pos_x < (wall_width/2+wall_long40+wall_width*2+30) && pos_x > wall_long40+wall_width) && (pos_y > -wall_height && pos_y < -1) && (pos_z < ground_z && pos_z > 0)){
686
687         Logger.Log( msg: "Current Position is Corridor");
688
689         return 1;
690     }
691     else{
692
693         Logger.Log( msg: "Current Position is Room");
694
695         return 0;
696     }
697 }
698
699 }
```

Figure 45.CalcPoint\_RoomCorr

## Calc\_Room

Calc\_Room can find in which room is the new camera position. Evaluate the point for each room area and return a codification integer value. This codification is:

RoomA – 1	RoomB – 2	RoomC – 3	RoomD – 4	RoomE – 5	RoomF- 6
-----------	-----------	-----------	-----------	-----------	----------

```
727 public int Calc_Room(float pos_x, float pos_y, float pos_z){
728
729     // Room_A (1)
730     if( (pos_x < wall_long40+wall_width && pos_x > wall_width) && (pos_y > -wall_height && pos_y < -1) && (pos_z < ground_z && pos_z > 0)){
731
732         Logger.Log( msg: "1 - Room_A");
733
734         return 1;
735     }
736     // Room_B (2)
737     else if((pos_x < wall_long40+wall_width && pos_x > wall_width) && (pos_y > -wall_height && pos_y < -1) && (pos_z < ground_z && pos_z > 0)){
738
739         Logger.Log( msg: "2 - Room_B");
740
741         return 2;
742     }
743 }
```

Figure 46.Calc\_Room

## CalcRuta

This function manages the features of the checkpoint objects to show the signal to the user.

```
789 public void CalcRuta(int position){
790
791     if(position == 1)
792     {
793
794         PointsInvisible();
795         BA.setVisibility(true);
796         //CE.setVisibility(true);
797     }
798     else if(position == 2)
799     {
800 }
```

Figure 47.CalcRuta

## onSensorChanged

Detect when a sensor event is executed thanks to the SensorManager.

```
806      @Override
807      public void onSensorChanged(SensorEvent event) {
808
809          if(event.sensor.getType() == Sensor.TYPE_ACCELEROMETER){
810              //Logger.log("The value of " + event.sensor.getName() + " is: " + event.values[0] +
811              }
812
813          if (event.sensor.getType() == Sensor.TYPE_ROTATION_VECTOR) {
814
```

Figure 48. onSensorChanged

## onAccuracyChanged

Called when the accuracy of the registered sensor has changed.

```
861      @Override
862      public void onAccuracyChanged(Sensor sensor, int accuracy) {
863      }
864
```

Figure 49.onAccuracyChanged

## panBy

This is the most important function.

In panBy the camera position is moved in a lateral or lineal movement.

In panBy the camera direction is rotated.

In panBy the alignment process is done between the real world and the virtual one. It means that both worlds are aligned from the north magnetic as a reference.

```
486      private void panBy(float dx, float dy, int ang, int app) {
487
488          Camera c = world.getCamera();
489
490          vector1 = c.getPosition();
491          vector_array = vector1.toArray(); // Returns a float[3]-array containing the components of the vector
492          Logger.log( msg: "Camera vector is: " + vector_array[0] + " / " + vector_array[1] + " / " + vector_array[2]);
493
494          // Check if the point is inside the world
495          if(IsInWorld(vector_array[0], vector_array[1], vector_array[2])){
496
497              if(CalcPoint_RoomCorr(vector_array[0], vector_array[1], vector_array[2]) == 1){
498
499                  ground.setTexture("t3"); //White t3 - antes t1 BLUE - CORRIDOR antes
500                  Logger.log( msg: "Es pasillo");

```

Figure 50. panBy

## 10.5 North Magnetic alignment

One of the most important points on this project is to get the alignment between the virtual and real world. To get this goal two references have been taken. The first one will serve for both worlds share a common point (0,0,0) calculate by fingerprint technique. The second one reference uses the north magnetic as a reference. In this point we must remind that the

smartphone used has the next sensors: accelerometer, gyroscope, and geomagnetic field sensor as required. Thus, we can use the geomagnetic field sensor in combination with the accelerometer to determine a device's position relative to the magnetic north pole. Azimuth, Roll, Pitch show the angle rotation movement according to the three-axis x, y, z.

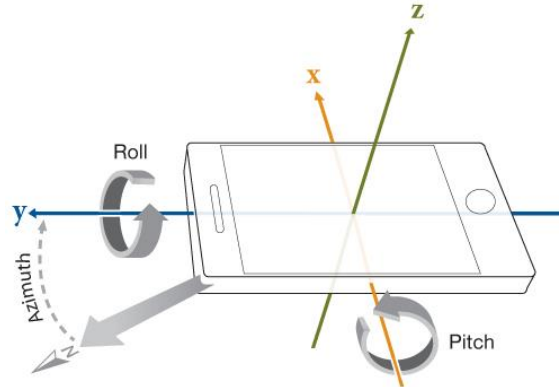


Figure 51. Azimuth - Roll - Pitch

**-Azimuth** (degrees of rotation around the z axis).

This is the angle between magnetic north and the device's Y-axis.

For example, if the device's Y-axis is aligned with magnetic north this value is 0°, and if the device's Y-axis is pointing south this value is 180°.

Likewise, when the Y-axis is pointing east this value is 90° and when it is pointing west this value is 270°.

The next points show the steps followed to achieve the alignment process.

## 1. Azimuth angle monitoring

The sensor data taken from sensors is processed by `SensorManger.getOrientation`. As a result, the Azimuth angle value is calculated. It is necessary to translate the angle to degrees. Then the data is sent to `panBy` function. The next figure shows the code developed for it.

```
mAzimuth = (int) ( Math.toDegrees( SensorManager.getOrientation( rotationMatrix, orientationAngles )[0] ) + 360 ) % 360;
mPitch = (int) ( Math.toDegrees( SensorManager.getOrientation( rotationMatrix, orientationAngles )[1] ) + 360 ) % 360;
mRoll = (int) ( Math.toDegrees( SensorManager.getOrientation( rotationMatrix, orientationAngles )[2] ) + 360 ) % 360;

Logger.log( msg: "Compass: " + " X: " + mAzimuth + " / Y: " + mPitch + " / Z: " + mRoll);

glView.queueEvent(new Runnable()
{
    @Override
    public void run()
    {
        panBy( dx: 0, dy: 0, mAzimuth, app: 2);
    }
});
```

Figure 52. Azimuth angle monitoring

## 2. Camera rotation

In panBy function, the camera is rotated from Y-axis. The rotated angle meets with the variation between the current Azimuth angle and the previous Azimuth angle measured. For this reason, the camera can rotate in clockwise and counterclockwise directions. Then if the difference is positive the camera rotates in clockwise and if the different is negative the camera rotates counterclockwise.

```
// Rotation Movement Camera
if(app == 2) {
    angulo1 = ang - aux;
    aux = ang;
    c.rotateCameraY((float) Math.toRadians(angulo1));
}
else if(app == 1){
    c.moveCamera(Camera.CAMERA_MOVEIN, speed: dy*0.1f);
    c.moveCamera(Camera.CAMERA_MOVERIGHT, speed: dx*0.05f);
}
```

Figure 53. Camera rotation

## 3. Reference initial angle

Previously it was commented that the rotation movement considers the current and previous Azimuth values and in consequence the camera can rotate.

In this point is necessary to talk about the variable “aux”, this variable is used in the process to calculate rotation angle. The alignment between virtual world and real-world environments is achieved by this variable. In the next step is explained.

## 4. Calibration process

This following point deals with calibration system to achieve the alignment between the virtual and real-world environments. A range of values will be introducing into the “aux” variable and for each value it will be checked if the worlds are aligned. Although it is not necessary to test all 360 possible values. An estimation can be made for taking a range of minor values. For this, the following picture shows the relation between the mobile device and the representation of virtual world.



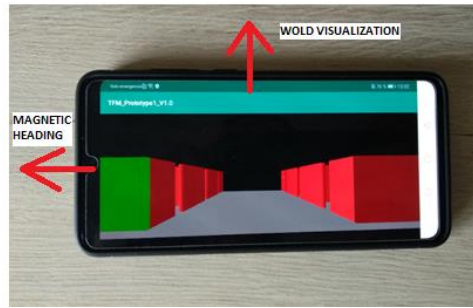


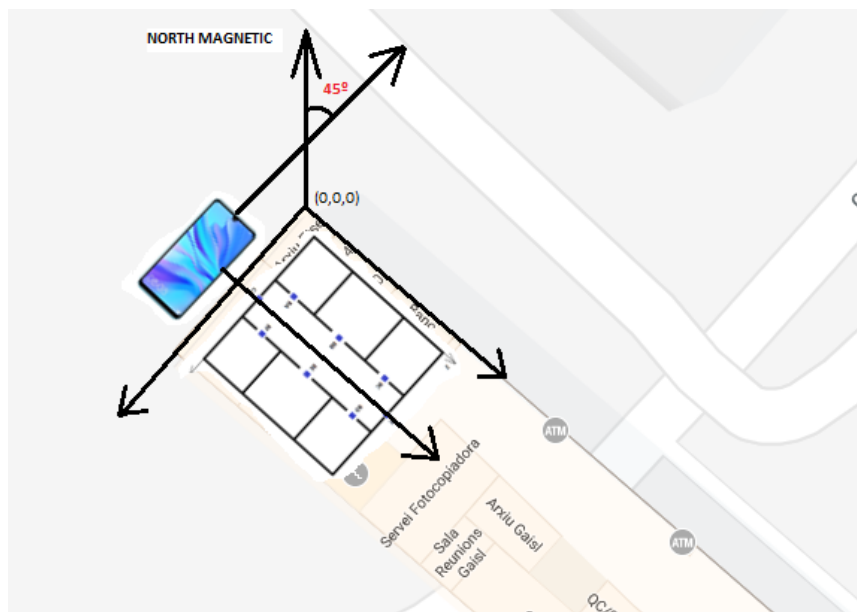
Figure 54. Virtual World and Magnetic heading directions

There is a difference of  $90^\circ$  degrees between the direction that the smartphone used to detect the north magnetic respecting to the world visualization. This fact is due to the screen orientation is  $90^\circ$  from the magnetic heading smartphone.

It is necessary to consider that the mobile phone cannot rotate according to Z-axis. It has to stay on the landscape position with the magnetic heading on the left side where the user takes the phone.

As an example, we will use the UAB engineering school building. The point reference (0,0,0) is shown in the next image.

**Note:** It will be supposed as a correct the North magnetic direction marked.



Therefore, the variation between the mobile phone Z-Axis and the north magnetic is  $45^\circ$  which means the Azimuth value is  $45^\circ$  too. Then, the variable “aux” will be evaluated in the range of  $30^\circ$ -  $60^\circ$  in order to get the alignment of both worlds. As a simple test it will be positioned at the entrance of the building (where it is situated on the smartphone in the picture below) and

be visualized if the virtual hallway matches with the real building. From this point, a complete 360° rotation will be done. When the rotation will be completed, we must orientate in the same direction as we started. This test proves that both worlds matches, and the calibration process can be considered concluded.

## **11 Tests and Results**

### **11.1 Tests**

The Verification and Validation process is mandatory in all SW project. It is necessary to ensure the system satisfies the specification. To verify and validate the correct functionality of the application and as well the correct construction of the application a group of tests are defined. Normally the tests are defined per categories as Unit tests, Component tests, Integration test, System tests, Reliability Tests.

The Test plan document allows all members of SW team to analyze some important KPIs:

1. Requirements coverage (tests coverage)  
Check the scope of the project is achieved
2. State of the application (tests results)  
Shows to developers, testers and product owners the project evolution

This Test plan is composed by a group of tests.

#### **Installation Tests:**

Test – The app is installed on an Android 4.4 device correctly and it is functional

Test – The app is installed on an Android 9.0 device correctly and it is functional

Test – The app is installed on an Android Full HD screen and it is functional

Test – The app is installed on an Android FullHD+ screen and it is functional

Test – The app is called “TFM\_Prototype1\_VX.Y”

#### **Functional Test:**

Test – The walls shown on the screen are built in red color

Test – The floor shown on the screen are built in white color

Test – The camera rotates 360° in the virtual world

Test – The camera rotates 360° in the virtual world and meets with 360° real rotation movement.

Test – The camera position moves in lateral ways (right and left)

Test – The camera position moves in linear ways (front and behind)

Test – There are not checkpoint signals shown when the position is no in the building.

Test – The function CalcRuta is well executed taking as initial position the room 1.

Repeat the last test for all the rooms

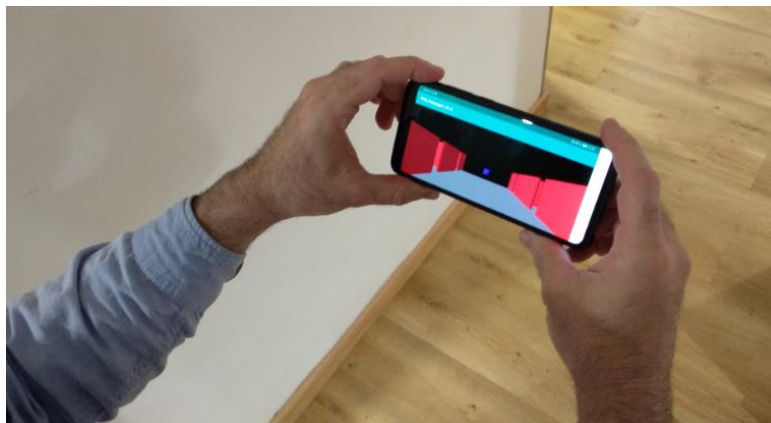
### **11.2 Results**

All the test described in section 11.1 were executed and the result was OK.

In the next pictures are shown some captures of the test execution. The user can execute the application on his smartphone with FullHD+ screen. The checkpoint signals are shown when the position is inside the building area.

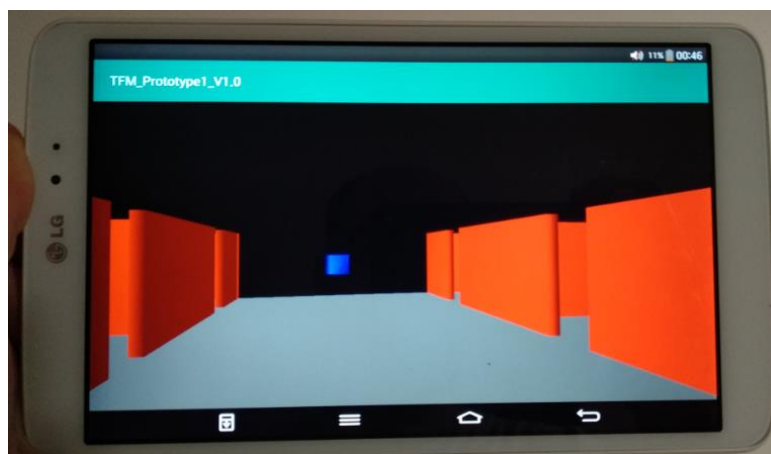


*Figure 55. User 1 - application*



*Figure 56. User 2 - application*

In the next test we validate the application runs in a device with Android minimum version kit kat.



*Figure 57. Device 1. - Android 4.4 kitkat*

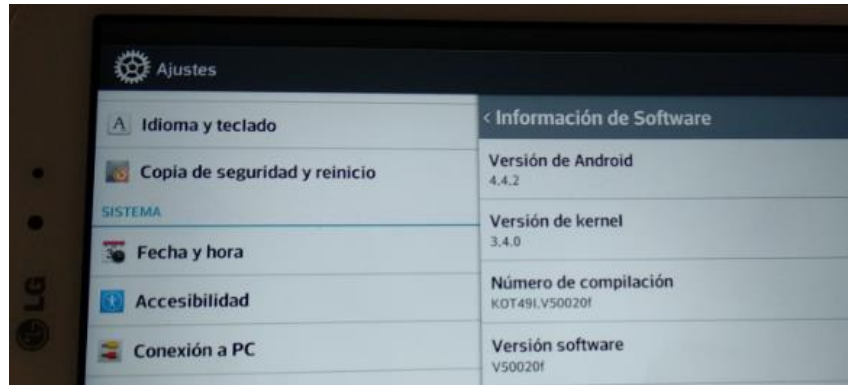


Figure 58. Device 1.2 - Android 4.4, kit kat

In the next test we validate the world is shown on the phone screen and the checkpoint signals are not shown. At the same time the name of the application is validated too.

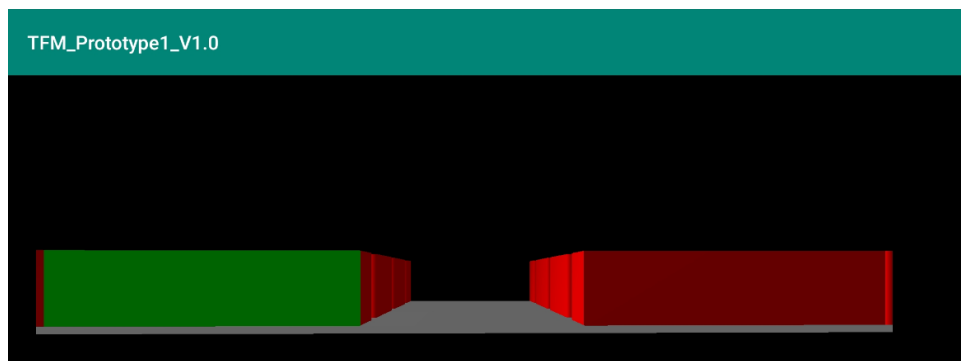


Figure 59. Virtual World

Execute a test where the initial position is inside a room. The checkpoint signal to find the way out of the room is shown in blue color.

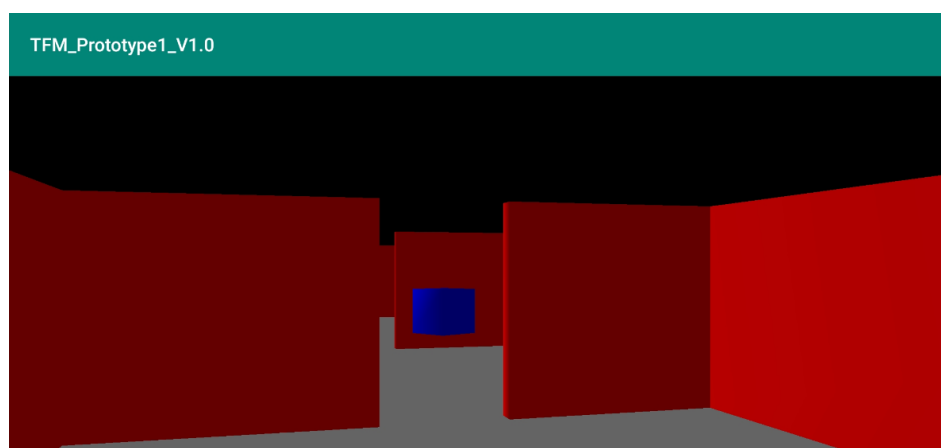
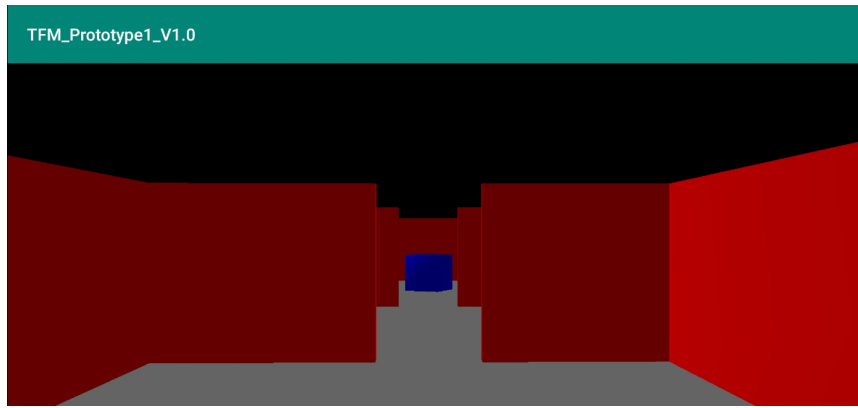
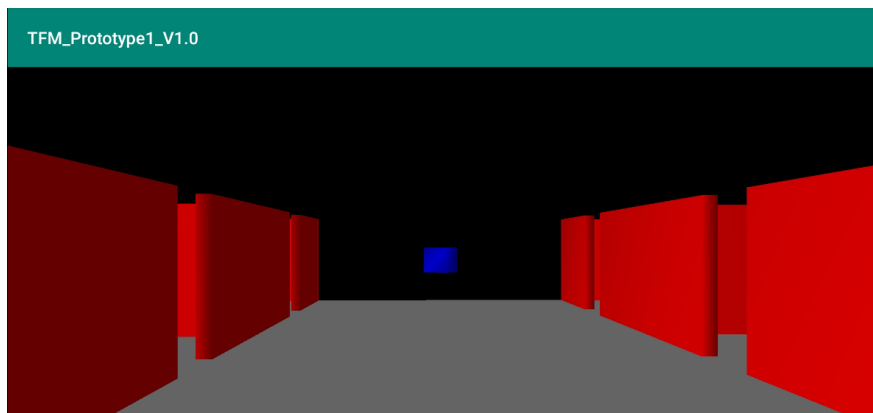


Figure 60. Checkpoint signal Room 1

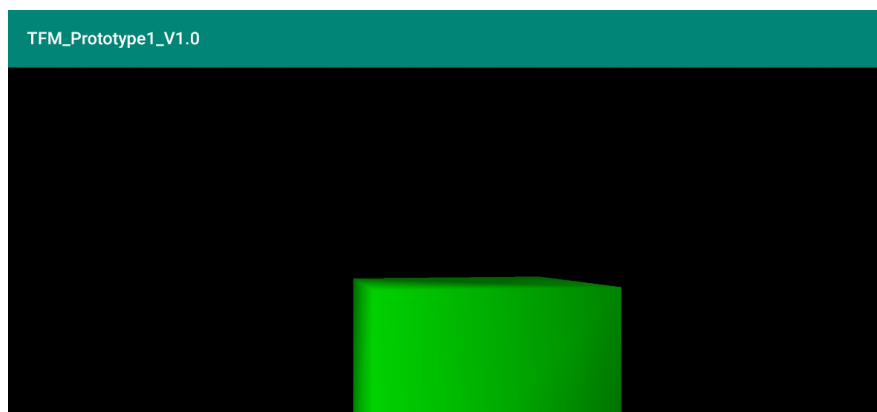


*Figure 61. Checkpoint signal Room 2*

When the user is in the corridor the next checkpoint signal is shown with blue color.



When the user arrived to the final checkpoint signal this object change to green color indicating to the user that is in the assembly point.



## **12 Conclusions and Future Works**

### ***12.1 Conclusions***

After the analysis of the project, it is time to conclude with the objectives that have been achieved and make an evaluation of all the stages of this project.

The prototype\_1 has been successfully finalized. It meets with the requirements of the project. However, there are limitations that explain the delay on the planning project.

1. The introduction to Android and JPCT environment was very slow. Understand how to build a world and how to manage the queue in the render process was very hard to plan.
2. The devices requirements used that mentioned are not correct. A new requirement must be created for the specification of a type of sensors. Since the Azimuth measure has contained errors, we have suffered significant variations making us doubt in the development. This fact makes us to review a new option to capture the information introducing an important deviation on the time planning.
3. The rotation system of the camera was complex to achieve. I tried to make complex solutions using the provided information by the sensors. After the investigation, I finally found out in the library the existence of a function that add rotations in the camera.

Due to these complications during the project, only the prototype\_1 has been developed. On the other hand, the development of prototype\_1 has permit the verification of the feasibility of the project and the possibility to face the prototype\_2.

As a conclusion, this project was very interesting. During the investigation period, I had the opportunity to read more about AR and Android projects and I was surprised to discover all the companies behind these technologies and the community of developers which works around these technologies.

### ***12.2 Future steps***

It is recommended that further works be undertaken in the following areas.

1. The project has clearly demonstrated that this custom-developed solution is adapted to limited group users. In particular, the users with high-acuity vision as the route signage is based on visual elements that guide the user through spaces at every moment. In

order to extend access to more users, it would be interesting to introduce audios that describe the next action to do to reach the way out (for instance “follow”, “turn right”, “turn left”, “stop”, “climb the stairs”, “go down stairs”). In addition, the route signage would be complemented with floor marking and checkpoint signals.

2. Regarding operating systems, a natural progression of this app solution is to develop it for IOS. As explained earlier, this current solution is based on Android for his low-cost OS in comparison of the others and his significant market position. However, it would be necessary to extend the solution to IOS users due to the fact that they represent more than 12% of the Spanish market and even more in other countries.
3. Make an exhaustive analysis of the needs in order to improve the requirements by modifying them or adding the new ones.



### 13 Bibliography and References

1. IEEE Standard Definitions of Terms for Antennas, Yi SUN, Yubin ZHAO, Jochen SCHILLER. 2015 IEEE Wireless Communications and Networking Conference (WCNC)
2. Viet-Cuong Ta. Smartphone-based indoor positioning using Wi-Fi, inertial sensors and Bluetooth. Machine Learning [cs.LG]. Université Grenoble Alpes, 2017. English. ffNNT : 2017GREAM092ff. fftel01883828f. <https://tel.archives-ouvertes.fr/tel-01883828/document>
3. Open Geospatial Consortium. OGC webpage. <http://www.opengeospatial.org/>.
4. Unigis – Servicio de Sistemas de Información Geográfica y Teledetección. Unigis official webpage. <https://www.unigis.es/posicionamiento-indoor/>
5. Android Developers. Android official webpage. <https://developer.android.com/>
6. Oracle Corp. API Java. Oracle official webpage. <https://docs.oracle.com/javase/7/docs/api/>
7. Oracle Corporation. Java webpage. <https://www.java.com/es/about/>
8. Universidad de la Rioja. Yanapay: evacuation system based on RFID technology and Android devices. Dialnet webpage. <https://dialnet.unirioja.es/servlet/articulo?codigo=5972717>
9. University of Pittsburgh. On Indoor Position Location with Wireless Lans. <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.127.3543&rep=rep1&type=pdf>
10. IEEE Survey of Wireless Indoor Positioning Techniques and Systems. Hui Liu, Houshang Darabi, Pat Banerjee, Jing Liu. 2007 IEEE Transactions on systems, Man and Cybernetics
11. JPCT 3D engine. JPCT official webpage. <http://www.jpct.net/>
12. Microsoft HoloLKens. HoloLens 2 offical webpage. <https://www.microsoft.com/en-us/hololens>
13. Google Glass. Glass official webpage. <https://www.google.com/glass/start/>
14. IndoorGML. IndoorGML official webpage. <http://www.indoorgml.net/>
15. Neosentec SL. Neosentec official webpage. <https://www.neosentec.com/realidad-aumentada/>
16. UOC. Indoor positioning. UOC blog. <http://informatica.blogs.uoc.edu/2016/04/21/posicionamiento-en-interiores-indoor-positioning/>
17. Air-fi. Air-fi official webpage. <http://www.air-fi.es/finalizacion-del-proyecto-sip-sistema-de-indoor-positioning/>
18. Geospatial World youtube - What is Indoor Positioning System and how does it work?. Youtube channel. Youtube channel video. [https://www.youtube.com/watch?v=rJGl6\\_crZmw](https://www.youtube.com/watch?v=rJGl6_crZmw)
19. Bluetooth SIG Inc. Bluetooth blog. <https://www.bluetooth.com/blog/bluetooth-positioning-systems/>

## 14 Annex - Code

```
package com.example.threedactivity;

import android.hardware.Sensor;
import android.hardware.SensorEvent;
import android.hardware.SensorEventListener;
import android.hardware.SensorManager;
import android.opengl.GLSurfaceView;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.MotionEvent;
import android.view.View;
import com.threed.jpct.Camera;
import com.threed.jpct.FrameBuffer;
import com.threed.jpct.Light;
import com.threed.jpct.Logger;
import com.threed.jpct.Object3D;
import com.threed.jpct.Primitives;
import com.threed.jpct.RGBColor;
import com.threed.jpct.SimpleVector;
import com.threed.jpct.Texture;
import com.threed.jpct.TextureManager;
import com.threed.jpct.World;
import com.threed.jpct.util.ExtendedPrimitives;
import javax.microedition.khronos.egl.EGLConfig;
import javax.microedition.khronos.opengles.GL10;

public class MainActivity extends AppCompatActivity implements GLSurfaceView.Renderer,
SensorEventListener, View.OnTouchListener
{
    private GLSurfaceView glView;
    private World world;
    private FrameBuffer fb;
    private Light light;
    private Object3D cube, cube2, cube3, cube4, ground;
```

```

private Object3D wall_1, wall_2, , wall_3, wall_4, wall_5, wall_6, wall_7, wall_8, wall_9,
wall_10, wall_11, wall_12, wall_13, wall_14, wall_15, wall_16, wall_17, wall_18;

private Object3D RA,RB,RC,RD,RE,RF,CA,CB,CC,CD,CE,BA,BB,BC,BD,BE,BF;


// WAY OUT zones
//private int WayOut = 0; // Exit is check point list CA
private int WayOut = 1; // Exit is check point list CE


// world size and error
private float ground_y = 1f;
private float ground_z = 94;
private float ground_x = 114;
private float world_error = 0;
private float wall_width = 1f;
private float wall_height = 10f;
private float wall_long12 = 12;
private float wall_long25 = 25;
private float wall_long40 = 40;
private float wall_long41 = 41;
private float wall_long94 = 94;


// CHECKPOINTS - room
private SimpleVector PRA = new SimpleVector(wall_width+wall_long40/2,-
wall_height/2,wall_width+30/2);
private SimpleVector PRB = new SimpleVector(wall_width+wall_long40/2,-
wall_height/2,2*wall_width+30+30/2);
private SimpleVector PRC = new SimpleVector(wall_width+wall_long40/2,-
wall_height/2,3*wall_width+30+30+30/2);
private SimpleVector PRD = new
SimpleVector(3*wall_width+wall_long40+30+wall_long40/2,-
wall_height/2,2*wall_width+30+30+30/2);
private SimpleVector PRE = new
SimpleVector(3*wall_width+wall_long40+30+wall_long40/2,-
wall_height/2,2*wall_width+30+30/2);

```

```

private          SimpleVector          PRF          =          new
SimpleVector(3*wall_width+wall_long40+30+wall_long40/2,-
wall_height/2,wall_width+30/2);

```

```

// CHECKPOINTS - corridor

```

```

private SimpleVector PCA = new SimpleVector(2*wall_width+wall_long40+30/2,-
wall_height/2,wall_width);

```

```

private SimpleVector PCB = new SimpleVector(2*wall_width+wall_long40+30/2,-
wall_height/2,wall_width+30/2);

```

```

private SimpleVector PCC = new SimpleVector(2*wall_width+wall_long40+30/2,-
wall_height/2,2*wall_width+30+30/2);

```

```

private SimpleVector PCD = new SimpleVector(2*wall_width+wall_long40+30/2,-
wall_height/2,3*wall_width+30+30+30/2);

```

```

private SimpleVector PCE = new SimpleVector(2*wall_width+wall_long40+30/2,-
wall_height/2,wall_long94-wall_width);

```

```

// CHECKPOINTS - borders

```

```

private SimpleVector PBA = new SimpleVector(wall_width/2+wall_width+wall_long40,-
wall_height/2,wall_width+30/2);

```

```

private SimpleVector PBB = new SimpleVector(wall_width/2+wall_width+wall_long40,-
wall_height/2,2*wall_width+30+30/2);

```

```

private SimpleVector PBC = new SimpleVector(wall_width/2+wall_width+wall_long40,-
wall_height/2,3*wall_width+30+30+30/2);

```

```

private          SimpleVector          PBD          =          new
SimpleVector(wall_width/2+2*wall_width+wall_long40+30,-
wall_height/2,3*wall_width+30+30+30/2);

```

```

private          SimpleVector          PBE          =          new
SimpleVector(wall_width/2+2*wall_width+wall_long40+30,-
wall_height/2,2*wall_width+30+30/2);

```

```

private          SimpleVector          PBF          =          new
SimpleVector(wall_width/2+2*wall_width+wall_long40+30,-
wall_height/2,wall_width+30/2);

```

```

// Wall sizes

```

```

private SimpleVector size_ground = new SimpleVector(ground_x,ground_y,ground_z);
private          SimpleVector          size_wall_1          =          new
SimpleVector(wall_width,wall_height,wall_long12);
private          SimpleVector          size_wall_2          =          new
SimpleVector(wall_width,wall_height,wall_long25);
private          SimpleVector          size_wall_3          =          new
SimpleVector(wall_width,wall_height,wall_long25);
private          SimpleVector          size_wall_4          =          new
SimpleVector(wall_width,wall_height,wall_long12);
private          SimpleVector          size_wall_5          =          new
SimpleVector(wall_width,wall_height,wall_long12);
private          SimpleVector          size_wall_6          =          new
SimpleVector(wall_width,wall_height,wall_long25);
private          SimpleVector          size_wall_7          =          new
SimpleVector(wall_width,wall_height,wall_long25);
private          SimpleVector          size_wall_8          =          new
SimpleVector(wall_width,wall_height,wall_long12);
private          SimpleVector          size_wall_9          =          new
SimpleVector(wall_long40,wall_height,wall_width);
private          SimpleVector          size_wall_10         =          new
SimpleVector(wall_long40,wall_height,wall_width);
private          SimpleVector          size_wall_11         =          new
SimpleVector(wall_long40,wall_height,wall_width);
private          SimpleVector          size_wall_12         =          new
SimpleVector(wall_long40,wall_height,wall_width);
private          SimpleVector          size_wall_13         =          new
SimpleVector(wall_width,wall_height,wall_long94);
private          SimpleVector          size_wall_14         =          new
SimpleVector(wall_width,wall_height,wall_long94);
private          SimpleVector          size_wall_15         =          new
SimpleVector(wall_long41,wall_height,wall_width);
private          SimpleVector          size_wall_16         =          new
SimpleVector(wall_long41,wall_height,wall_width);

```

```

private SimpleVector size_wall_17 = new
SimpleVector(wall_long41,wall_height,wall_width);
private SimpleVector size_wall_18 = new
SimpleVector(wall_long41,wall_height,wall_width);

```

```

// Aux. var. camera position in method panBy

```

```

private SimpleVector vector1;

```

```

private int aux = 210;

```

```

private int angulo1 = 0;

```

```

//private SimpleVector size_x = new SimpleVector(1,5,94);

```

```

private float[] vector_array = { 1000,1000,1000};

```

```

//***** SENSORS *****

```

```

private SensorManager miManager;

```

```

private Sensor miSensorAc;

```

```

private Sensor miSensorGi;

```

```

private Sensor miSensorPr;

```

```

private Sensor miSensorMa;

```

```

// Aux. var. new camera points

```

```

private int touch_click = 0;

```

```

private int evento = 0;

```

```

private int mCount;

```

```

private int mAzimuth = 0; // degree

```

```

private int mPitch = 0; // degree

```

```

private int mRoll = 0; // degree

```

```
//*****  
*****
```

```
//*****  
*****
```

```
@Override
```

```
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);
```

```
    // Get an instance of the SensorManager  
    miManager=(SensorManager) getSystemService(SENSOR_SERVICE);
```

```
    // Create our Preview view and set it as the content of our Activity  
    glView = new GLSurfaceView(this);  
    glView.setEGLContextClientVersion(2);  
    glView.setRenderer(this);  
    setContentView(glView);
```

```
    glView.setOnTouchListener(this);
```

```
//*****  
*****
```

```
    miSensorAc=miManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);  
    miSensorGi=miManager.getDefaultSensor(Sensor.TYPE_GYROSCOPE);  
    miSensorPr=miManager.getDefaultSensor(Sensor.TYPE_ROTATION_VECTOR);
```

```
    miSensorMa=miManager.getDefaultSensor(Sensor.TYPE_GEOMAGNETIC_ROTATION_  
    VECTOR);
```

```

if(miSensorAc == null) {
    Logger.log("Accelerometer is not available");
    finish(); // Close app
}

if(miSensorGi == null) {
    Logger.log("Gyroscope is not available");
    finish(); // Close app
}

if(miSensorPr == null){
    Logger.log("Rotation Vector is not available");
    finish(); // Close app
}

}

@Override
protected void onResume() {
    super.onResume();
    glView.onResume();

    miManager.registerListener(this, miSensorPr, 100000);
    miManager.registerListener(this, miSensorAc, 100000);
    miManager.registerListener(this, miSensorGi, 100000);
    miManager.registerListener(this, miSensorMa, 100000);
}

@Override
protected void onPause() {
    glView.onResume();

```



```

super.onPause();
miManager.unregisterListener(this);
}

```

@Override

```

public void onSurfaceCreated(GL10 gl, EGLConfig config) {

```

```

// INSTANCE OF WORLD CLASS - CAMERA CLASS - TEXTURE CLASS

```

```

world = new World(); //instancia mundo

```

```

Camera a = world.getCamera();

```

```

a.setPosition(57, -7, -25);

```

```

a.rotateCameraY((float) Math.toRadians(0));

```

```

//TEXTURES

```

```

Texture t1 = new Texture(64,64, RGBColor.BLUE); //instancia textura

```

```

TextureManager.getInstance().addTexture("t1", t1); //aplicar textura a

```

```

TextureManager

```

```

Texture t2 = new Texture(64,64, RGBColor.RED);

```

```

TextureManager.getInstance().addTexture("t2", t2);

```

```

Texture t3 = new Texture(64,64, RGBColor.WHITE);

```

```

TextureManager.getInstance().addTexture("t3", t3);

```

```

Texture t4 = new Texture(64,64, RGBColor.GREEN);

```

```

TextureManager.getInstance().addTexture("t4", t4);

```

```

// ***** ROOM POINTS

```

```

*****

```

```

RA = Primitives.getCube(1f);

```

```
RA.translate(PRA);
RA.setTexture("t1");
```

```
RB = Primitives.getCube(1f);
RB.translate(PRB);
RB.setTexture("t1");
```

```
RC = Primitives.getCube(1f);
RC.translate(PRC);
RC.setTexture("t1");
```

```
RD = Primitives.getCube(1f);
RD.translate(PRD);
RD.setTexture("t1");
```

```
RE = Primitives.getCube(1f);
RE.translate(PRE);
RE.setTexture("t1");
```

```
RF = Primitives.getCube(1f);
RF.translate(PRF);
RF.setTexture("t1");
```

```
//          *****          CORRIDOR          POINTS
```

```
*****
```

```
CA = Primitives.getCube(1.5f);
CA.translate(PCA);
CA.setTexture("t4");
```

```
CB = Primitives.getCube(1f);
CB.translate(PCB);
CB.setTexture("t1");
```

```

CC = Primitives.getCube(1f);
CC.translate(PCC);
CC.setTexture("t1");
CC.setCollisionMode(Object3D.COLLISION_CHECK_OTHERS);

```

```

CD = Primitives.getCube(1f);
CD.translate(PCD);
CD.setTexture("t1");

```

```

CE = Primitives.getCube(1.5f);
CE.translate(PCE);
CE.setTexture("t4");

```

```

//          *****          BORDER          POINTS
*****

```

```

BA = Primitives.getCube(1f);
BA.translate(PBA);
BA.setTexture("t1");

```

```

BB = Primitives.getCube(1f);
BB.translate(PBB);
BB.setTexture("t1");

```

```

BC = Primitives.getCube(1f);
BC.translate(PBC);
BC.setTexture("t1");

```

```

BD = Primitives.getCube(1f);
BD.translate(PBD);
BD.setTexture("t1");

```

```

BE = Primitives.getCube(1f);
BE.translate(PBE);
BE.setTexture("t1");

```

```

BF = Primitives.getCube(1f);
BF.translate(PBF);
BF.setTexture("t1");

```

```

//                      *****                      WORLD                      ELEMENTS

```

```

*****

```

```

ground = ExtendedPrimitives.createBox(size_ground);
ground.translate(57f,0.5f,47f);
ground.setTexture("t3");

```

```

wall_1 = ExtendedPrimitives.createBox(size_wall_1);
wall_1.translate(((wall_width/2)+wall_long40+wall_width),-wall_height/2f,((ground_z-
wall_width)-(wall_long12/2)));
wall_1.setTexture("t2");

```

```

wall_2 = ExtendedPrimitives.createBox(size_wall_2);
wall_2.translate(((wall_width/2)+wall_long40+wall_width),-wall_height/2f,(ground_z-
wall_width-30-(wall_width/2)));
wall_2.setTexture("t2");

```

```

wall_3 = ExtendedPrimitives.createBox(size_wall_3);
wall_3.translate(((wall_width/2)+wall_long40+wall_width),-
wall_height/2f,((1.5f*wall_width)+30));
wall_3.setTexture("t2");

```

```

wall_4 = ExtendedPrimitives.createBox(size_wall_4);
wall_4.translate(((wall_width/2)+wall_long40+wall_width),-
wall_height/2f,(wall_width+(wall_long12/2)));

```

```

wall_4.setTexture("t2");

wall_5 = ExtendedPrimitives.createBox(size_wall_5);
wall_5.translate(((wall_width/2)+wall_long40+wall_width*2+30),-
wall_height/2f,((ground_z-wall_width)-(wall_long12/2)));
wall_5.setTexture("t2");

wall_6 = ExtendedPrimitives.createBox(size_wall_6);
wall_6.translate(((wall_width/2)+wall_long40+wall_width*2+30),-
wall_height/2f,(ground_z-wall_width-30-(wall_width/2)));
wall_6.setTexture("t2");

wall_7 = ExtendedPrimitives.createBox(size_wall_7);
wall_7.translate(((wall_width/2)+wall_long40+wall_width*2+30),-
wall_height/2f,((1.5f*wall_width)+30));
wall_7.setTexture("t2");

wall_8 = ExtendedPrimitives.createBox(size_wall_8);
wall_8.translate(((wall_width/2)+wall_long40+wall_width*2+30),-
wall_height/2f,(wall_width+(wall_long12/2)));
wall_8.setTexture("t2");

wall_9 = ExtendedPrimitives.createBox(size_wall_9);
wall_9.translate((wall_width+(wall_long40/2)), -wall_height/2f, wall_width+30+30);
wall_9.setTexture("t2");

wall_10 = ExtendedPrimitives.createBox(size_wall_10);
wall_10.translate((wall_width+(wall_long40/2)), -wall_height/2f, wall_width+30);
wall_10.setTexture("t2");

wall_11 = ExtendedPrimitives.createBox(size_wall_11);
wall_11.translate(((3*wall_width)+(wall_long40/2)+wall_long40+30),-
wall_height/2f, wall_width+30+30);
wall_11.setTexture("t2");

```

```

wall_12 = ExtendedPrimitives.createBox(size_wall_12);
wall_12.translate(((3*wall_width)+(wall_long40/2)+wall_long40+30),-
wall_height/2f,wall_width+30);
wall_12.setTexture("t2");

wall_13 = ExtendedPrimitives.createBox(size_wall_13);
wall_13.translate(0.5f,-wall_height/2f,47f);
wall_13.setTexture("t2");

wall_14 = ExtendedPrimitives.createBox(size_wall_14);
wall_14.translate(113.5f,-wall_height/2f,47f);
wall_14.setTexture("t2");

wall_15 = ExtendedPrimitives.createBox(size_wall_15);
wall_15.translate(((wall_width)+(wall_long41/2)),-wall_height/2f,(wall_long94-
wall_width/2));
wall_15.setTexture("t2");

wall_16 = ExtendedPrimitives.createBox(size_wall_16);
wall_16.translate((30+wall_long41+(wall_width)+(wall_long41/2)),-
wall_height/2f,(wall_long94-wall_width/2));
wall_16.setTexture("t2");

wall_17 = ExtendedPrimitives.createBox(size_wall_17);
wall_17.translate(((wall_width)+(wall_long41/2)),-wall_height/2f,wall_width/2f);
wall_17.setTexture("t4");

wall_18 = ExtendedPrimitives.createBox(size_wall_18);
wall_18.translate((30+wall_long41+(wall_width)+(wall_long41/2)),-
wall_height/2f,wall_width/2f);
wall_18.setTexture("t2");

```

```
//*****  
*****
```

```
// ADD ALL ELEMENTS TO THE WORLD
```

```
world.addObject(RA);  
world.addObject(RB);  
world.addObject(RC);  
world.addObject(RD);  
world.addObject(RE);  
world.addObject(RF);
```

```
world.addObject(CA);  
world.addObject(CB);  
world.addObject(CC);  
world.addObject(CD);  
world.addObject(CE);
```

```
world.addObject(BA);  
world.addObject(BB);  
world.addObject(BC);  
world.addObject(BD);  
world.addObject(BE);  
world.addObject(BF);
```

```
world.addObject(ground);
```

```
world.addObject(wall_1);  
world.addObject(wall_2);  
world.addObject(wall_3);  
world.addObject(wall_4);
```

```
world.addObject(wall_5);
world.addObject(wall_6);
world.addObject(wall_7);
world.addObject(wall_8);
world.addObject(wall_9);
world.addObject(wall_10);
world.addObject(wall_11);
world.addObject(wall_12);
world.addObject(wall_13);
world.addObject(wall_14);
world.addObject(wall_15);
world.addObject(wall_16);
world.addObject(wall_17);
world.addObject(wall_18);
```

```
// INSTANCE OF LIGHT CLASS
```

```
light = new Light(world);
light.setIntensity(128,128,128);
}
```

```
@Override
```

```
public void onSurfaceChanged(GL10 gl, int width, int height) {
    fb = new FrameBuffer(width, height);          //INSTANCE OF BUFFER
}
```

```
@Override
```

```
public void onDrawFrame(GL10 gl) {
    fb.clear();          //Lights and transforms are applied to all stored objects
```



```
// CHECKPOINTS ROTATION
```

```
RA.rotateY(0.05f);
RB.rotateY(0.05f);
RC.rotateY(0.05f);
RD.rotateY(0.05f);
RE.rotateY(0.05f);
RF.rotateY(0.05f);
CA.rotateY(0.05f);
CB.rotateY(0.05f);
CC.rotateY(0.05f);
CD.rotateY(0.05f);
CE.rotateY(0.05f);
BA.rotateY(0.05f);
BB.rotateY(0.05f);
BC.rotateY(0.05f);
BD.rotateY(0.05f);
BE.rotateY(0.05f);
BF.rotateY(0.05f);
```

```
world.renderScene(fb);    //the scene is drawn on the frame buffer
```

```
world.draw(fb);          //Show the object with all their features
```

```
fb.display();            //printar en buffer el render
```

```
}
```

```
//***** panBy modify camera position; It is possible to modify the speed camera
movement *****
```

```
private void panBy(float dx, float dy, int ang, int app) {
```

```

Camera c = world.getCamera();

vector1 = c.getPosition();
vector_array = vector1.toArray(); // Returns a float[3]-array containing the components
of the vector

Logger.log("Camera vector is: " + vector_array[0] + " / " + vector_array[1] + " / " +
vector_array[2]);

// Check if the point is inside the world
if(IsInWorld(vector_array[0], vector_array[1], vector_array[2])){

    if(IsWayOut(vector_array[0], vector_array[1], vector_array[2])){

        }

        else{

            if(CalcPoint_RoomCorr(vector_array[0], vector_array[1], vector_array[2]) == 1){
//1-Corridor

                ground.setTexture("t3"); //White t3 - antes t1 BLUE - CORRIDOR antes
                Logger.log("Is Corridor");
                CalcRuta(7);

            }

            else if(CalcPoint_RoomCorr(vector_array[0], vector_array[1], vector_array[2]) ==
0){ //0-Rooms

                ground.setTexture("t3"); //WHITE - ROOM
                Logger.log("Is a Room");

                CalcRuta(Calc_Room(vector_array[0],    vector_array[1],    vector_array[2]));
//Check the room where is the position

            }

```

```

else{

    ground.setTexture("t4"); //GREEN - ERROR
    Logger.log("Is Out of the World");

}

}

}

else {

    ground.setTexture("t3"); //White - Antes t2 RED
    PointsInvisible();

}

// Rotation Movement Camera
if(app == 2) {

    angulo1 = ang - aux;
    aux = ang;
    c.rotateCameraY((float) Math.toRadians(angulo1));

}

else if(app == 1){

    c.moveCamera(Camera.CAMERA_MOVEIN, dy*0.1f);
    c.moveCamera(Camera.CAMERA_MOVERIGHT, dx*0.05f); //Mueve la cámara
derecha con una determinada velocidad

}

else{

```

```
}
```

```
}
```

```
private float lastX, lastY; // Aux.Var. to keep the last position used in "onTouch" function
```

```
@Override
```

```
public boolean onTouch(View v, MotionEvent event) {
```

```
    touch_click++;
```

```
    // It is necessary to count "2"
```

```
    // each action of pressing the touch screen 1-ACTION_DOWN and 2-ACTION_UP are  
counted 1 each one
```

```
    // so it is necessary to count until 2 to count 2 actions
```

```
    if(touch_click % 2 == 0)
```

```
    {
```

```
        evento++;
```

```
    }
```

```
    switch(event.getAction()){
```

```
        case MotionEvent.ACTION_DOWN:
```

```
            lastX = event.getX();
```

```
            lastY = event.getY();
```

```
            return true;
```

```
        case MotionEvent.ACTION_MOVE:
```

```
            final float dx = event.getX()-lastX;
```

```
            final float dy = event.getY()-lastY;
```

```
            lastX = event.getX();
```

```
            lastY = event.getY();
```

//Using queueEvent() provides a convenient way of executing a method in the rendering thread.

```
glView.queueEvent(new Runnable() {  
    @Override  
    public void run() {  
        panBy(dx, dy, 1000,1);  
    }  
});  
}  
return false;  
}
```

```
//      *****   Check   whether   new   point   is   inside   the   world  
*****
```

```
public boolean IsInWorld(float x, float y, float z){  
  
    if((x <= (ground_x + world_error) && x >= (-world_error)) && (y >= (-wall_height) &&  
y <= -1) && (z >= (-world_error) && z <= (ground_z + world_error))){  
  
        Logger.log("The point is inside the Area");  
        return true;  
  
    }  
    else{  
  
        Logger.log("The point is outside the Area");  
  
        return false;  
    }  
  
}
```

```

public void PointsInvisible()
{
    BA.setVisibility(false);
    BB.setVisibility(false);
    BC.setVisibility(false);
    BD.setVisibility(false);
    BE.setVisibility(false);
    BF.setVisibility(false);

    CA.setVisibility(false);
    CB.setVisibility(false);
    CC.setVisibility(false);
    CD.setVisibility(false);
    CE.setVisibility(false);

    RA.setVisibility(false);
    RB.setVisibility(false);
    RC.setVisibility(false);
    RD.setVisibility(false);
    RE.setVisibility(false);
    RF.setVisibility(false);

}

public boolean IsWayOut(float x, float y, float z){

    if(WayOut == 0 ){    // Way Out is Point CA

        if((x < (30 + wall_width + wall_long41)) && ( x > (wall_width + wall_long41)) && (z
< 12 && z > 0))    // Define the condition
        {

            PointsInvisible();
            CA.setTexture("t4");    //GREEN

```

```

        CA.setVisibility(true);

        return true;
    }

    else{
        CA.setTexture("t1");    //BLUE
        CE.setVisibility(false);
        return false;
    }

}

if(WayOut == 1 ){    // Way Out is Point CE

    if((x < (30 + wall_width + wall_long41)) && ( x > (wall_width + wall_long41)) && (z
< (ground_z) && z > (ground_z - 12))) // Define the condition
    {

        PointsInvisible();
        CE.setTexture("t4");    //GREEN
        CE.setVisibility(true);

        return true;
    }

    else{
        CE.setTexture("t1");    //BLUE
        CA.setVisibility(false);
        return false;
    }

}

```

```

else{

    return false;
}

}

//***** Calculate the current position is room or corridor
*****

public int CalcPoint_RoomCorr(float pos_x, float pos_y, float pos_z){

    // Check if position is in the corridor surface
    if( (pos_x < (wall_width/2+wall_long40+wall_width*2+30) && pos_x >
wall_long40+wall_width) && (pos_y > -wall_height && pos_y < -1) && (pos_z < ground_z
&& pos_z > 0)){

        Logger.log("Current Position is Corridor");

        return 1;

    }
    else{

        Logger.log("Current Position is Room");

        return 0;
    }

}

```



```

public int Calc_Room(float pos_x, float pos_y, float pos_z){

    // Room_A (1)
    if( (pos_x < wall_long40+wall_width && pos_x > wall_width) && (pos_y > -wall_height
&& pos_y < -1) && (pos_z < wall_width+30 && pos_z > wall_width)){

        Logger.log("1 - Room_A");

        return 1;

    }
    // Room_B (2)
    else if((pos_x < wall_long40+wall_width && pos_x > wall_width) && (pos_y > -
wall_height && pos_y < -1) && (pos_z < 2*wall_width+2*30 && pos_z >
2*wall_width+30)){

        Logger.log("2 - Room_B");

        return 2;

    }
    // Room_C (3)
    else if((pos_x < wall_long40+wall_width && pos_x > wall_width) && (pos_y > -
wall_height && pos_y < -1) && (pos_z < ground_z-wall_width && pos_z > ground_z-30-
wall_width)){

        Logger.log("3 - Room_C");

        return 3;

    }
    // Room_D (4)

```

```

        else if((pos_x < (ground_x-wall_width) && pos_x > ground_x-wall_long41) && (pos_y
> -wall_height && pos_y < -1) && (pos_z < ground_z-wall_width && pos_z > ground_z-30-
wall_width)){

            Logger.log("4 - Room_D");

            return 4;

        }
        // Room_E (5)
        else if((pos_x < (ground_x-wall_width) && pos_x > ground_x-wall_long41) && (pos_y
> -wall_height && pos_y < -1) && (pos_z < 2*wall_width+2*30 && pos_z >
2*wall_width+30)){

            Logger.log("5 - Room_E");

            return 5;

        }
        // Room_F (6)
        else if((pos_x < (ground_x-wall_width) && pos_x > ground_x-wall_long41) && (pos_y
> -wall_height && pos_y < -1) && (pos_z < wall_width+30 && pos_z > wall_width)){

            Logger.log("6 - Room_F");

            return 6;

        }
        else{

            Logger.log("No information");
            return 7;

        }

```

```
}
```

```
    /******* Calculate the new route; Precondition: the point_check_corr is verify previously  
    *****/
```

```
public void CalcRuta(int position){
```

```
    if(position == 1)
```

```
    {
```

```
        PointsInvisible();
```

```
        BA.setVisibility(true);
```

```
        //CE.setVisibility(true);
```

```
    }
```

```
    else if(position == 2)
```

```
    {
```

```
        PointsInvisible();
```

```
        BB.setVisibility(true);
```

```
        //CE.setVisibility(true);
```

```
    }
```

```
    else if(position == 3)
```

```
    {
```

```
        PointsInvisible();
```

```
        BC.setVisibility(true);
```

```
        //CE.setVisibility(true);
```

```
    }
```

```
    else if(position == 4)
```

```

{
    PointsInvisible();
    BD.setVisibility(true);
    //CE.setVisibility(true);

}
else if(position == 5)
{
    PointsInvisible();
    BE.setVisibility(true);
    //CE.setVisibility(true);

}
else if(position == 6)
{

    PointsInvisible();
    BF.setVisibility(true);
    //CE.setVisibility(true);

}
else if(position == 7)
{

    PointsInvisible();

    if(WayOut == 0){

        CA.setVisibility(true);
        CA.setTexture("t1");    //BLUE

    }
    else if(WayOut == 1){

```

```

        CE.setVisibility(true);
        CE.setTexture("t1");    //BLUE

    }
    else{

    }

}
else if(position == 0){

}
else{

}
}
}

```

```

@Override
public void onSensorChanged(SensorEvent event) {

    if(event.sensor.getType() == Sensor.TYPE_ACCELEROMETER){
        //Logger.log("The value of " + event.sensor.getName() + " is: " + event.values[0] + " /
" + event.values[1] + " / " + event.values[2]);
    }

    if (event.sensor.getType() == Sensor.TYPE_ROTATION_VECTOR) {

        // convert the rotation-vector to a 4x4 matrix. the matrix is interpreted by Open GL as
the inverse of the rotation-vector, which is what we want.

        // Rotation matrix based on current readings from accelerometer and magnetometer.
        final float[] rotationMatrix = new float[9];
    }
}

```

```

miManager.getRotationMatrixFromVector(rotationMatrix, event.values);

// Express the updated rotation matrix as three orientation angles.
final float[] orientationAngles = new float[3];
miManager.getOrientation(rotationMatrix, orientationAngles);

if (mCount++ > 0) {

    mCount = 0;

    mAzimuth = (int) ( Math.toDegrees( SensorManager.getOrientation( rotationMatrix,
orientationAngles )[0] ) + 360 ) % 360;
    mPitch = (int) ( Math.toDegrees( SensorManager.getOrientation( rotationMatrix,
orientationAngles )[1] ) + 360 ) % 360;
    mRoll = (int) ( Math.toDegrees( SensorManager.getOrientation( rotationMatrix,
orientationAngles )[2] ) + 360 ) % 360;

    Logger.log("Compass: " + " X: " + mAzimuth + " / Y: " + mPitch + " / Z: " + mRoll);

    glView.queueEvent(new Runnable()
    {
        @Override
        public void run()
        {
            panBy(0,0, mAzimuth, 2);
        }
    });
}
}
}

```

```
@Override  
public void onAccuracyChanged(Sensor sensor, int accuracy) {  
    }  
  
}
```