uc3m | Universidad Carlos III de Madrid

Grado en Ingeniería en Tecnologías de Telecomunicación
Curso académico 2016-2017

Trabajo Fin de Grado

"Simulación de efectos electromagnéticos utilizando la tecnología de realidad aumentada"

Paloma Gracia Ballesteros

Tutora

María Blanca Ibáñez Espiga

Título
Simulaciones de efectos electromagnéticos utilizando la tecnología de realidad aumentada

Autora
Paloma Gracia Ballesteros
Directora
María Blanca Ibáñez Espiga

Composición del tribunal

Presidente
María Carmen Guerrero López
Secretario
Ascensión Gallardo Antolín
Vocal
Pedro Martín Mateos

Realizado el acto de defensa y lectura del Proyecto Fin de Carrera el día 14 de marzo de 2017

En el municipio de la Comunidad de Madrid de Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid

Agradecimientos

En primer lugar, agradecer a mi tutora de este Trabajo de Fin de Grado por todo el tiempo y paciencia que ha tenido conmigo.

En segundo lugar, a mis padres y hermano por haberme apoyado durante todos mis años en la universidad, en los momentos buenos y en los malos, y haberme dado la oportunidad de poder disfrutar de seis meses de Erasmus, que fueron una experiencia única en la vida.

En tercer lugar, a mis compañeros de la universidad, tanto los que conozco desde el primer día como los que han ido llegando después, y sin los cuales quizás no habría llegado hasta donde estoy, y sin los cuales mi paso por la universidad no habría sido tan llevadero. Sobre todo, agradecer a Patricia y A. Escudero por su infinita paciencia conmigo durante todos estos años en los que me han ayudado en todo lo que han podido y por haber pasado conmigo un insufrible número de horas trabajando duro. Sin olvidarme de Álvaro, que me ha dado su apoyo incondicional en todo momento durante mi último año en la universidad y que me hizo aprender a valorarme.

Por último, agradecer a todas mis amigas y, en general, a toda la gente tanto de dentro como de fuera de la universidad que ha hecho que mi paso por la misma haya sido mucho mejor de lo que habría sido sin ellos. En especial, a mis amigas Sandra, María y Laura por haber estado siempre ahí; a mis compañeros y entrenadores del gimnasio a los cuales echare de menos cuando ya no esté en la universidad; y a los hermanos Montero que, aunque no estuvieron siempre ahí, me han ayudado mucho en mis últimos pasitos por la Universidad Carlos III de Madrid.

Índice

1. Introducción 10
1.1. Motivación 10
1.2. Objetivos 12
1.3. Marco regulador y socio-económico 13
1.3.1. Marco regulador 13
1.3.2. Marco socio-económico 14
1.4. Estructura de la memoria 15
2. Planteamiento del problema 16
3. Estado del arte 19
3.1. Dispositivos móviles 20
3.2. Aplicaciones móviles 22
3.3. Realidad aumentada 23
3.4. Herramientas de desarrollo 26
3.4.1. Unity 26
3.4.1.1. Escena 27
3.4.1.2. Objetos 3D 27
3.4.1.3. UI (User Intergace) y Canvas 28
3.4.1.4. Particle System 28
3.4.1.5. Material 28
3.4.1.6. Collider y Trigger 28
3.4.1.7. Rigid Body 29
3.4.1.8. Touch 29
3.4.1.9. Script 29
3.4.1.10. Librerías de Unity 29
3.4.1.11. Trail Renderer 30
3.4.1.12. Shader 30
3.4.2. Vuforia 31
3.4.2.1. Image Target y Frame Marker 32
3.4.2.2. AR Camera 33
3.4.3. Microsoft Visual Studio: C# 34
3.4.4. Blender 35
3.5. Sistemas operativos móviles 36

3.5.1. Android 38
3.5.2. Otros 40
4. Análisis de la aplicación 41
4.1. Diseño de la aplicación 42
4.2. Implementación de la aplicación 52
4.2.1. Implementación de la aplicación en C# 53
4.2.1.1. Clases y métodos utilizados en Unity 54
4.2.1.2. Desarrollo de la aplicación en C# 57
4.2.2. Implementación de pruebas 65
4.2.3. Empaquetado de la aplicación 69
5. Conclusiones 71
5.1. Conclusiones del proyecto 71
5.2. Valoración personal 72
6. Presupuesto y planificación 73
6.1. Presupuesto 73
6.2. Planificación 76
7. Líneas futuras de trabajo 79
8. Anexos 80
8.1. Abstract 81
8.2. Bibliografía 87

Lista de figuras

Figura 1. Dos hilos rectos largos que conducen corrientes paralelas. 16
Figura 2. Ley de la mano derecha o del tornillo. 17
Figura 3. Distancia entre los dos hilos (r_1). 17
Figura 4. Situación creada por la acción de las corrientes eléctricas. 18
Figura 5. Evolución del uso de dispositivos móviles según “ditrendia”. 20
Figura 6. Realidad aumentada proyectada sobre código QR. 24
Figura 7. Gafas de realidad aumentada. 24
Figura 8. Ejemplo de jerarquía entre objetos. 27
Figura 9. Escenario de Vuforia para realidad aumentada. 31
Figura 10. Image Target: Stones. 32
Figura 11. Frame Marker ID 0. 32
Figura 12. Uso de los diferentes sistemas operativos móviles. 37
Figura 13. Arquitectura Android. 38
Figura 14. Diseño base de las escenas (BS). 42
Figura 15. Escena de Log In. 43
Figura 16. Escena 2. 46
Figura 17. Enunciado mostrado en la Escena 2. 46
Figura 18. Escena 3. 47
Figura 19. Escena 4. 48
Figura 20. Escena 5. 49
Figura 21. Escena 6. 50
Figura 22. Escena 7. 51
Figura 23. Estructura de un proyecto C# en Unity. 54
Figura 24. Ejemplo de uso de método OnGUI(). 56
Figura 25. Diagrama de flujo botón “Corriente 1”. 58
Figura 26. Diagrama de flujo botón “Corriente 2”. 59
Figura 27. Diagrama de flujo botón “Simulación”. 60
Figura 28. Diagrama de flujo botón “Continuar”. 61
Figura 29. Script OnTrigger. 63
Figura 30. Método OnGUI() para la creación de las ventana. 64
Figura 31. Botón para activar el Play Mode (1) y consola (2). 66
Figura 32. Aspecto del texto de los enunciados. 68
Figura 33. Pasos para el empaquetado de la aplicación. 69
Figura 34. Diagrama de Gantt. 78

Lista de tablas

Tabla 1. Duración en días de las fases del proyecto. 74
Tabla 2. Presupuesto de los recursos materiales. 74
Tabla 3. Presupuesto de los recursos humanos..... 75
Tabla 4. Presupuesto total del proyecto. 75
Tabla 5. Detalle de las tareas del diagrama de Gantt. 76

1. Introducción

1.1. Motivación

Uno de los motores principales del ser humano es y ha sido siempre el deseo de avanzar y comprender. De estos anhelos surge la creación de las universidades en el siglo XIII, con el fin de llegar a comprender mejor la vida y la misma esencia del propio ser humano.

Algunos ejemplos de este afán de conocimiento son el desarrollo de los diferentes algoritmos, ecuaciones y teorías llevados a cabo en dichas universidades, o el avance en economía, medicina o, en nuestro caso, tecnología.

Pero además de conseguir estos avances, también se ha buscado siempre una mejora en las técnicas de aprendizaje con el fin de optimizar el trabajo de los estudiantes.

El campo tecnológico ha sido uno de los que más ha avanzado en las últimas siete décadas, donde ha habido un creciendo exponencial desde el momento en el que se crea el primer ordenador en 1953, en el que IBM cambió la historia de la tecnología con el lanzamiento de la primera computadora, el IBM 701 EDPM. [1] [2]

Desde entonces, no sólo ha aumentado la cantidad de tecnología que tenemos a nuestro alcance, sino que, además, ha habido un gran aumento de la necesidad, e incluso dependencia, de todas estas tecnologías de las que disponemos hoy en día y que están cada día más presentes en nuestras vidas.

Uno de los ejemplos más claros de esta presencia de nuevas tecnologías en nuestro día a día es, sin lugar a dudas, el uso de los *Smartphones* y, por supuesto, de la gran cantidad de aplicaciones que hay para dichos dispositivos.

La educación no es inmune a la irrupción de estas tecnologías hoy día y los educadores buscan la forma en la que estas tecnologías pueden ser integradas en los procesos de aprendizaje para motivar a los estudiantes y facilitarles la adquisición de nuevas competencias.

En nuestro caso concreto, y con esta misma finalidad, se pensó que sería una buena idea juntar el temario impartido en la asignatura de Física en el Grado en Ingeniería Informática sobre electromagnetismo con uno de los últimos avances en tecnología: la realidad aumentada.

Con esta unión de la física y la realidad aumentada, se pretende crear una aplicación con el fin de ayudar a los estudiantes de la asignatura a visualizar en un espacio tridimensional efectos de electromagnetismo, ya que éstos son difíciles de mostrar sobre una pizarra (espacio bidimensional), y por tanto causan dificultad a aquellos estudiantes con poca capacidad de visión espacial.

Por tanto, se elige el uso de la realidad aumentada debido a que permite la proyección en tres dimensiones, en tiempo real y desde cualquier perspectiva objetos reales a los que se les superpone una determinada información. De este modo, el estudiante podrá manipular marcadores que representarán cables conductores, simular el paso de la corriente a través de ellos y observar desde la perspectiva que desee los cambios que se producen en los vectores de fuerza y campo magnético que ocurrirían en un sistema real.

1.2. Objetivos

A continuación, vamos a especificar los objetivos del proyecto. Para ello, vamos a definir 2 tipos de objetivos: unos objetivos parciales, que son aquellos objetivos concretos que se deben conseguir para el buen desarrollo del proyecto y que son necesarios para llegar al resultado final, y unos objetivos principales, que se refieren a la finalidad del proyecto, es decir, que definen cuál es el resultado final al que se pretende llegar con la realización del mismo.

Los **objetivos parciales** son los que se definen a continuación:

- Conocer qué es la realidad aumentada y los diferentes usos y ventajas que puede suponer, en este caso, en relación a la educación y la enseñanza.
- Conocer y aprender el funcionamiento de la herramienta de diseño: Unity. Esta herramienta nos ayudará a poder llevar a cabo el diseño de la aplicación a través de elementos como objetos 3D, Canvas, materiales…
- Conocer las funcionalidades que nos proporciona Vuforia a través de Unity para poder introducir la realidad aumentada.
- Aprender a utilizar el lenguaje de programación necesario para llevar a cabo la aplicación, C#, y su entorno de programación, Microsoft Visual Studio. Esto incluye tareas como, por ejemplo, saber cómo se define cada tipo de dato, conocer las funciones que incluyen las librerías de Unity para este lenguaje de programación y saber cómo usar cada una de ellas o conocer las herramientas de depuración de la plataforma de programación. Gracias a ellos, podemos programar la aplicación para que tenga las funcionalidades deseadas.
- Aprender a diseñar objetos 3D con Blender, ya que nos hará falta para poder construir los objetos digitales necesarios en la simulación que no nos proporciona Unity.

Los **objetivos principales** son los siguientes:

- Proporcionar al estudiante una aplicación para un dispositivo Android con un ejercicio que le permitan conocer y entender de una forma más clara ciertos conceptos de la asignatura gracias a la realidad aumentada.
- Facilitar a los docentes de la asignatura la enseñanza de dichos conceptos a través de la aplicación.

1.3. Marco regulador y socio-económico

En este apartado se procederá a la realización de un análisis tanto del marco regulador como de la situación socio-económica actual en la cual se ve inmersa la tecnología de realidad aumentada.

1.3.1. Marco regulador

Para la realización de este proyecto, se han usado únicamente herramientas de *software* gratuito, por lo que no aplica un marco regulador específico para las mismas.

Sin embargo, y a pesar de que también es gratuito, se necesita obtener una licencia para el uso de Vuforia a través del *License Manager* de la página web de Vuforia (*Vuforia Developer Portal*).

Además, hemos aceptado las políticas de privacidad y condiciones de uso de las diferentes herramientas de *software*, las cuales debemos aceptar a la hora de descargar dichas herramientas. Algunos de los puntos que se indican en estas políticas son, en otros, la violación de los derechos de autor por parte del usuario, la responsabilidad de éste de no utilizar el *software* con fines que vayan en contra de las leyes o la privacidad de los datos del usuario por parte de los responsables de la herramienta.

También se debe tener en cuenta que, al inicio de la aplicación, se recogen los datos personales de los alumnos (su nombre, apellidos y número de identificación de alumno). Respecto a esto, se deberá respetar la protección de datos de los estudiantes y asegurar la confidencialidad de los mismos.

Además, la aplicación hace uso de la cámara del dispositivo, por lo que se deberá tener en cuenta su normativa respecto a ello.

1.3.2. Marco socio-económico

La situación de la economía global se va a ver afectada por las políticas desarrolladas por las principales potencias económicas en los próximos años. De las alianzas, los bloqueos comerciales y la estabilidad que se genere desde las administraciones de dichas potencias dependerá mucho el crecimiento a nivel internacional.

En Europa, la situación económica actual viene marcada por la decisión de una de las grandes potencias de abandonar el marco de libre circulación de personas, bienes y servicios. Sin embargo, el Brexit no ha sido hasta el momento una fuerte inestabilidad profunda. De producirse finalmente la desvinculación total durante el 2017, no se prevén fuertes sacudidas para la Unión Europea.

A nivel nacional, la economía española finalizó el 2016 con un crecimiento anual del PIB del 3,2%. Las predicciones apuntan a que España será una de las economías principales de Europa que más crecerá en el año 2017, se calcula un crecimiento en torno al 2,3% gracias en parte a la reducción del endeudamiento, la disminución de la tasa de desempleo o a la reestructuración de los sistemas bancarios, sobre todo el de las cajas de ahorros. [3]

El modelo productivo español tiende en la actualidad hacia un modelo de producción con un fuerte componente tecnológico. Todos estos datos resultan ventajosos especialmente para los sectores con mayor presencia en España, entre los que se encuentran las nuevas tecnologías.

Respecto al mercado de la realidad aumentada, los datos son muy esperanzadores ya que se estima que éste sufrirá un crecimiento de más del 120% entre los años 2013 y 2018, según se publicó en un estudio conocido como *Global Augmented Reality Market 2014-2018*. [4]

En España, ya hay más de 140 empresas dedicadas a la realidad aumentada, y otras muchas están invirtiendo en dicha tecnología. Entre los sectores que trabajan con realidad aumentada, cabe destacar el de las tecnologías TIC, *Social Media*, *Marketing* o desarrollo de *software*. [5]

Este crecimiento se debe al amplio abanico de campos en los cuales puede ser utilizada dicha tecnología, así como al incremento del mercado de dispositivos con la tecnología necesaria para soportar la realidad aumentada y realidad virtual que están lanzando muchas de las grandes compañías a nivel mundial.

1.4. Estructura de la memoria

En los puntos que siguen, se procederá a explicar los aspectos más relevantes en el desarrollo del proyecto. Estos puntos serán los siguientes:

- **Planteamiento del problema:** en este apartado se explicará qué es lo que se pretende hacer en la aplicación, haciendo hincapié en la física de los ejercicios a representar en la misma.
- **Estado del Arte:** donde se explicará la situación actual de los aspectos más relevantes que conciernen al proyecto, tales como los *Smartphones* y *tablets*, la realidad aumentada, las aplicaciones móviles o los lenguajes de programación; además, se explicarán las diferentes herramientas que han sido necesarias para el desarrollo de la aplicación, se expondrán los sistemas operativos móviles y se presentarán trabajos anteriormente realizados en relación al proyecto.
- **Análisis de la aplicación:** aquí se expondrá todo el proceso que se ha llevado a cabo para la realización de la aplicación, incluyendo aspectos como el diseño, la programación o la elección de los elementos que se han utilizado de entre aquellos que nos proporcionan las herramientas de *software* utilizadas. En este apartado también se irán incluyendo alternativas que podrían haberse hecho para el desarrollo de la aplicación.
- **Conclusiones:** las cuales mostrarán un resumen de las conclusiones a las que se ha podido llegar tras la finalización del proyecto, desde un punto de vista tanto objetivo como subjetivo.
- **Presupuesto y planificación:** donde estará recogida toda la información relacionada con el presupuesto necesario para la realización del proyecto y la planificación del mismo.
- **Líneas futuras de trabajo:** aquí se recogerán diferentes posibilidades que puede haber en relación al proyecto de cara a utilidades del mismo en un futuro.

Para finalizar, se incluye un **anexo** con las referencias bibliográficas que han sido necesarias para recoger toda la documentación e información para el desarrollo tanto del proyecto como de la memoria. Además, se incluye un resumen en lengua inglesa de todo lo recogido en esta misma memoria.

2. Planteamiento del problema

Para el desarrollo de este proyecto, se propuso realizar una aplicación para dispositivos que soporten un sistema operativo Android y que dispongan de cámara y pantalla táctil. Dicha aplicación debe mostrar un ejercicio en relación a la asignatura de física, para observar fenómenos electromagnéticos a través de la tecnología de realidad aumentada.

Más concretamente, se debe realizar con esta tecnología un ejercicio consistente en dos hilos o cables rectos largos que conducen corrientes paralelas (cada una por uno de los hilos) tal y como se muestra en la Figura 1 [6]:

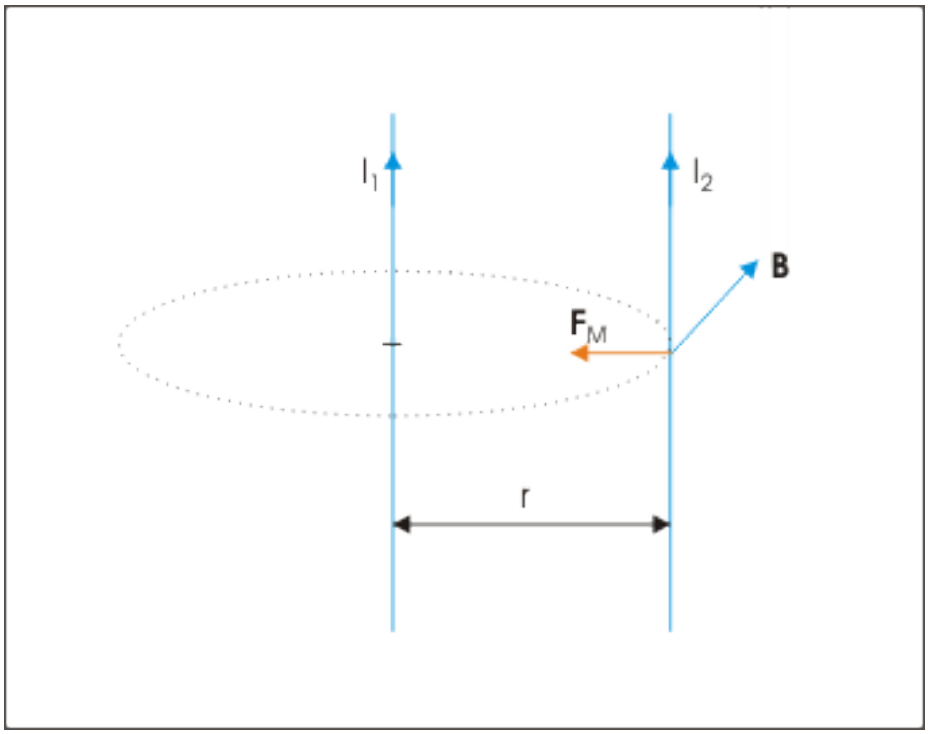


Figura 1. Dos hilos rectos largos que conducen corrientes paralelas.

Con esta aplicación se pretende que el estudiante vaya siguiendo una serie de pasos y respondiendo a algunas cuestiones que se le van planteando sobre los conocimientos que ha ido adquiriendo en las sesiones teóricas de la asignatura. Dichas cuestiones y pasos a seguir se explicarán más extensamente en la sección 4 de esta memoria.

Tal y como dictan las leyes de electromagnetismo, el hecho de que haya dos corrientes paralelas crea un **campo magnético** que será:

- concéntrico a cada uno de los hilos.
- con radio igual a la distancia entre ambos.
- cuya dirección de giro dependerá de la corriente que circule por el hilo de acuerdo a la "ley de la mano derecha" o del tornillo (es decir, sentido antihorario si es ascendente y sentido horario si es descendente) tal y como se muestra en la Figura 2:

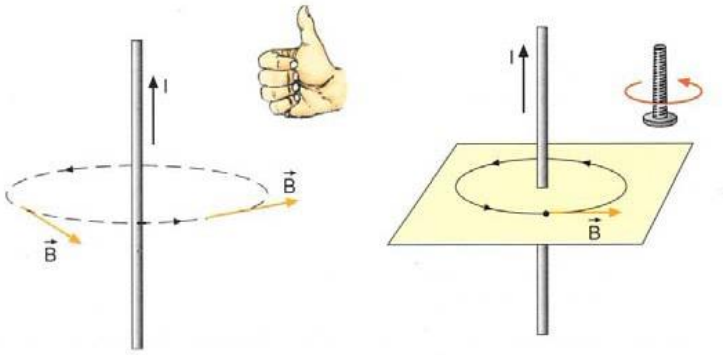


Figura 2. Ley de la mano derecha o del tornillo.

- con un vector de campo cuya intensidad es inversamente proporcional a la distancia entre los hilos y tangente al campo. Este vector de campo atiende a la siguiente expresión:

$$B_1 = (\mu_0 I_1) / (2 \pi r_1) \quad [T]$$

donde μ0 es la permeabilidad del vacío o del espacio libre y su valor es de $4 \pi 10^{-7}$ Tm/A, I1 es el valor de la corriente que atraviesa al cable 1 y r_1 es la distancia entre los cables y que en este caso será el valor de la hipotenusa que forma la distancia "a" a la que se encuentran los cables en los ejes X e Y, es decir, $\sqrt{a^2 + a^2}$.

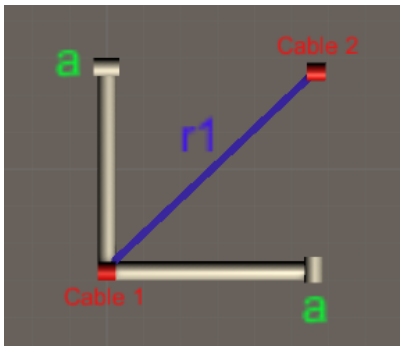


Figura 3. Distancia entre los dos hilos (r_1).

Además, esta situación crea una **fuerza magnética** cuyo vector:

- tiene una intensidad que dependerá del valor de la corriente y del campo magnético y la longitud del hilo, según la expresión:

$$F_2 = (I_2\ L) \times B_1$$

donde I_2 es la corriente que pasa por el cable 2 y L es un vector cuyo módulo es la longitud del hilo, su dirección es paralela a la corriente que circula por dicho hilo y su sentido es el de la corriente.

- va en dirección centrípeta al hilo si la dirección de sus corrientes es la misma, o en dirección opuesta a la centrípeta si la dirección de la corriente de los hilos es opuesta.

Esta misma fuerza, además, es la que hará que los hilos se acerquen si tienen corrientes en sentidos opuestos, o se alejen si tienen corrientes en el mismo sentido.

Por tanto, obtenemos un escenario como el mostrado en la siguiente imagen:

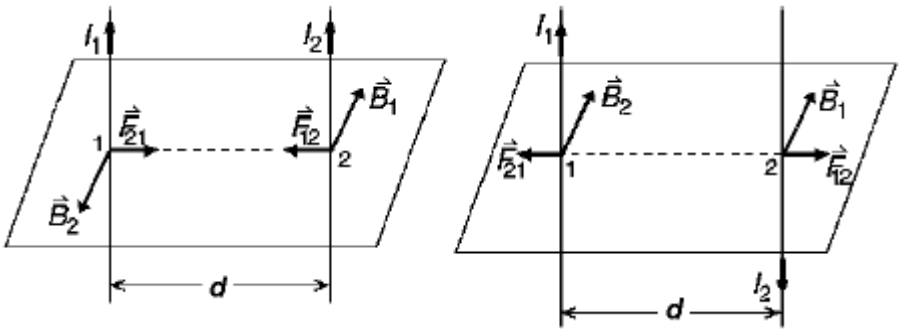


Figura 4. Situación creada por la acción de las corrientes eléctricas.

Todo lo anteriormente indicado, se le presentará al estudiante en una aplicación para dispositivos con sistema operativo Android y que el estudiante deberá resolver como un problema en un laboratorio de la asignatura de Física. El estudiante recibirá un enunciado y se le indicarán las opciones de interacción que tiene con la herramienta, de forma que pueda observar todo lo anteriormente explicado como una simulación en realidad aumentada en un espacio tridimensional para su mejor compresión.

3. Estado del arte

En esta sección, se va a hacer referencia al estado de los medios tecnológicos necesarios para poder llevar a cabo este proyecto y algunos ejemplos de aplicación de realidad aumentada, así como explicar qué puede aportar su uso.

Para ello, hablaremos del estado actual de:

- Dispositivos móviles: donde se hará referencia a la situación actual y presencia de los teléfonos inteligentes y sus capacidades.
- Aplicaciones móviles: donde se analizará el gran aumento que han sufrido en los últimos años debido a la proliferación de los dispositivos móviles donde son utilizadas.
- Realidad aumentada: donde se comentará la situación actual y los avances de la realidad aumentada.
- Herramientas de desarrollo: donde se explicará cada una de ellas y algunos de sus elementos más relevantes. Entre ellas, cabe destacar las siguientes:
    - Unity 3D
    - Blender
    - Visual Studio
- Sistemas operativos móviles: donde se expondrá una breve referencia a los sistemas operativos móviles y la situación actual de ellos. Entre ellos se encuentra los siguientes:
    - Android
    - iOS
    - Otros

3.1. Dispositivos móviles

Actualmente, los dispositivos móviles de los cuales ha incrementado en mayor medida su uso debido al avance de sus características y que, por tanto, pueden considerarse como los principales objetivos de uso, son los teléfonos móviles inteligentes o *Smartphones* y las *tablets*.

En los últimos años, ha había un incremento exponencial de los usuarios que hacen uso tanto de *Smartphones* como de *tablets* y las tecnologías que utilizan para proporcionar servicios a estos usuarios ha avanzado con gran rapidez, tal y como se muestra en la Figura 5. [7]

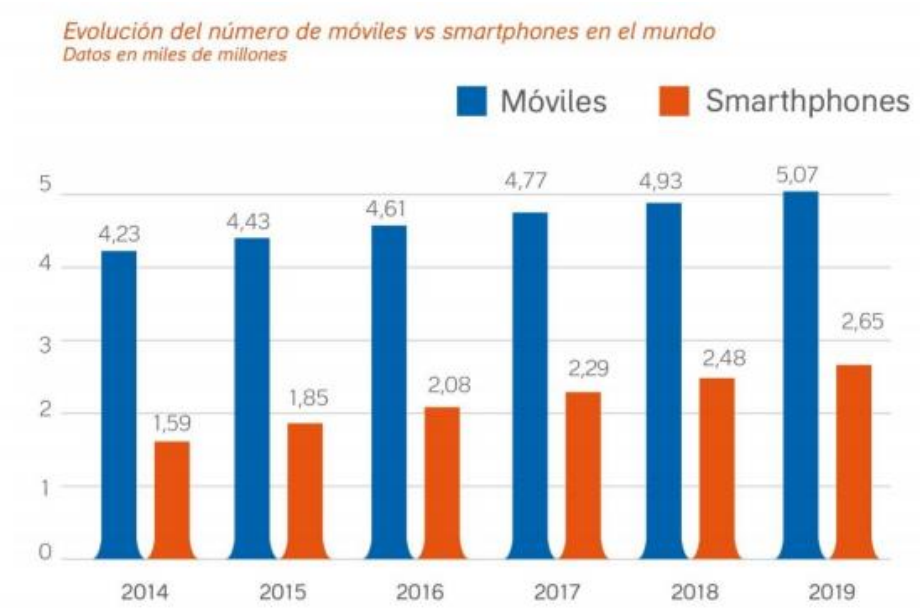


Figura 5. Evolución del uso de dispositivos móviles según "ditrendia".

Por una parte, los S*martphones* son la evolución de los antiguos teléfonos móviles, a los cuales se les fue añadiendo cada vez más elementos para mejorar la experiencia del usuario (como las pantallas táctiles, una mayor capacidad de almacenamiento o la conexión a internet) hasta llegar a los actuales dispositivos inteligentes, es decir, los *Smartphones*.

Por otra parte, las *tablets* son una evolución de los ordenadores portátiles con la finalidad de proporcionar al usuario una mayor movilidad del dispositivo, ya que la *tablet* es más reducida en tamaño y peso que un ordenador portátil y, además, permite el uso de aplicaciones, aunque en algunas de las últimas versiones de sistemas operativos para ordenadores también se permite la instalación de aplicaciones.

Algunos de los elementos más relevantes que son comunes a ambos dispositivos son: pantalla táctil, cámara de fotos y vídeo, conexión a internet, posibilidad de instalar aplicaciones, gran capacidad de almacenamiento y sistema operativo adecuado que permita el uso de estas herramientas como pueden ser iOS, Android, Windows Phone, Firefox OS o Tizen entre otros.

Todo esto lleva a que ambos dispositivos sean ideales para el uso de aplicaciones de realidad aumentada, donde se necesita: una pantalla táctil para poder interactuar con los objetos de dicha realidad aumentada a través de un *Touch* en la pantalla; una cámara digital que permita detectar elementos como las *Image Target* o los *Frame Marker* sobre los que se van a proyectar los objetos de nuestra realidad aumentada; capacidad de almacenamiento, ya que nuestra aplicación tiene un tamaño aproximado de 80 MB; y posibilidad del uso de aplicaciones, de tal forma que podamos instalar la aplicación en el dispositivo correspondiente.

3.2. Aplicaciones móviles

Las aplicaciones son una adaptación de los programas informáticos para poder ser usados sobre *Smartphones* o *tablets* y que permiten al usuario realizar una determinada tarea.

El uso de las aplicaciones ha ido aumentando desde el lanzamiento de los *Smartphones* y ha ido ganándole terreno a los programas que tradicionalmente se han usado siempre en ordenadores, debido a que su formato es más adecuado en prestaciones e interfaz gráfica para estos nuevos dispositivos.

Gracias a ellas, es posible tener acceso a la realidad aumentada en *Smartphones* o *tablets* ya que nos permiten aunar en un solo archivo tanto el diseño como la programación de dicha realidad aumentada sin necesidad de disponer de las diferentes herramientas de *software* que se necesitan para su desarrollo.

3.3. Realidad aumentada

La Realidad Aumentada, o AR (del inglés *Augmented Reality*), permite aumentar la percepción de nuestros sentidos mediante la superposición en el mundo físico de información digital, permitiendo al usuario percibir los cambios en tiempo real. [8]

Además de la realidad aumentada, existen otro tipo de tecnologías que guardan relación con ésta. Estas tecnologías son: la **realidad virtual**, en la cual, a diferencia de la realidad aumentada, el entorno virtual no es proyectado sobre el real, sino que en este caso dicho entorno real es sustituido por el virtual; y la **realidad mixta**, donde se mezclan características de realidad aumentada y virtual, ya que dicha realidad mixta proyecta elementos virtuales sobre el entorno real como sucede en realidad aumentada, pero además permite al usuario interactuar con objetos físicos del mundo real como elementos dentro del entorno virtual. [9]

Como ya se había comentado en el apartado anterior, para poder hacer uso de la realidad aumentada necesitamos:

- Un **dispositivo** que contenga el *software* adecuado para ejecutar la aplicación concreta de realidad aumentada.
- Una **pantalla** táctil para que el usuario pueda interactuar mediante un Touch (o sin táctil si no se requiere esta funcionalidad) con el entorno virtual que nos proporcione la aplicación.
- Una **cámara digital** donde podamos ver los objetos 3D de la realidad aumentada.
- Y un **elemento de proyección**, en general algún tipo de marcador impreso en papel, que será detectado por el dispositivo que se esté utilizando y sobre el que se proyectarán los objetos del entorno virtual.

Estos elementos para representar la realidad aumentada pueden ser de distintos tipos. Algunos de los más relevantes son los siguientes:

- **Códigos QR o marcadores:** consiste en proyectar los elementos virtuales sobre un código QR o un marcador (en nuestro caso será un *Frame Marker*). Podemos observar un ejemplo de ello en la Figura 6 [10].

Figura 6. Realidad aumentada proyectada sobre código QR.

- **Imágenes u objetos del entorno real:** consiste en reconocer estos objetos o imágenes (en nuestro caso será el *Image Target* de las piedras), de tal forma que se pueda representar una realidad aumentada determinada sobre ellos.

- **GPS:** el GPS también es utilizado para darnos información en forma de realidad aumentada a cerca del lugar en el que nos encontramos gracias a la señalización recogida por la señal GPS del dispositivo.

- **Gafas de realidad aumentada:** uno de los últimos avances en realidad aumentada, es poder incorporarla en gafas que nos permitan su proyección. Esto transmite al usuario una mayor sensación de poder interactuar con el entorno virtual. Ejemplos de esto son las *Google Glass* o las *3D Glasses* y *VR Gear Glasses* diseñadas para *Smartphones*. [11] [12]

Figura 7. Gafas de realidad aumentada.

Actualmente, el uso de la realidad aumentada es cada vez mayor y sus aplicaciones abarcan campos muy diversos. A continuación, se presentan algunos ejemplos de sus diferentes aplicaciones en algunos de los campos en los cuales es usada.

En concreto, en este estado del arte nos centraremos en la aplicación de realidad aumentada en materia de educación, ya que es donde se engloba nuestro proyecto.

En el campo de la educación se han llevado a cabo muchas aplicaciones en relación a diversos campos del conocimiento, todas ellas con la finalidad de mejorar la enseñanza de cada una de las materias a las que se refieren dichas aplicaciones.

Por ejemplo, se han realizado implementaciones en las cuales dos o más usuarios pueden observar el mismo objeto desde diferentes perspectivas (dependiendo del punto de referencia en el que se encuentre cada uno) de representaciones tridimensionales de datos en forma de gráfica. [13]

Otro ejemplo de aplicación podría ser en el campo de la química, donde se ha desarrollado un entorno con el cual el usuario puede observar la reacción de aminoácidos por medio de la interacción entre *Frame Markers* y objetos físicos del mundo real. [14]

En relación a este proyecto, el Departamento de Ingeniería Telemática de la Universidad Carlos III de Madrid ha trabajado con realidad aumentada para realizar aplicaciones con ejercicios de electromagnetismo. En ellos, se mostraba a los estudiantes una serie de ejercicios consistentes en la representación de un cable, una batería y unos imanes, a través de los cuales el estudiante podía comprender de una forma más interactiva principios sobre fenómenos eléctricos. [15] [16]

Con esto, se pretendía estudiar la motivación y concentración de los alumnos mediante la realización de simulaciones de electromagnetismo a través de la tecnología de realidad aumentada. El resultado al que se llegó fue que el aprendizaje de los estudiantes es mayor en el caso de enseñanza con realidad aumentada que en el caso de enseñanza mediante diapositivas teóricas. [17]

3.4. Herramientas de desarrollo

En este apartado, se presentarán todas las herramientas de *software* que han sido usados en el desarrollo e implementación de la aplicación, y se hará una explicación de cada uno de ellas.

3.4.1. Unity

Unity fue creada en el año 2005 en Dinamarca por la compañía Unity Technologies. Es un motor de juego multiplataforma y totalmente integrado que permite crear entornos interactivos tanto en 2D como en 3D, y que es usado principalmente para el desarrollo de videojuegos para Windows, Mac, Linux, iOS, Android, tvOS, Tizen, Xbox, Play Station y Samsung TV.

Es la herramienta de *software* principal usada para el diseño e implementación de la aplicación. Gracias a ella, podemos crear el entorno virtual que formará parte de la realidad aumentada, y programar dicho entorno para permitirle al usuario la interacción con la aplicación.

Dispone de una interfaz gráfica que nos permite ver de forma sencilla y ordenada todo lo que estamos usando y diseñando en cada momento.

En este proyecto se ha utilizado la versión 5.3.4 de Unity ya que era la última versión existente en el momento de la descarga (mayo de 2016) y suponía una mejora sobre el resto de versiones existentes hasta el momento.

Para crear una aplicación con Unity, el proyecto sobre el que trabajemos debe dividirse en escenas. A estas escenas se le irán añadiendo determinados componentes. Por ejemplo, podemos introducir objetos 3D, a los cuales se les otorgará un comportamiento específico mediante la adición de ***Components***. En particular, los *Components* más relevantes en nuestro proyecto han sido los *Scripts*, que permitirán a los usuarios de la aplicación la interacción con los objetos virtuales de la misma.

3.4.1.1. Escena

Las escenas son instancias que se crean por separado y en las que se introducen todos los elementos que después podrán observarse en la aplicación. Son un elemento muy importante dentro del desarrollo y diseño de la aplicación ya que son las que van a permitir crear todas y cada una de las diferentes pantallas que se visualizarán en la aplicación.

3.4.1.2. Objetos 3D

Los objetos 3D son cada uno de los objetos que forman la realidad aumentada de la aplicación.

Son los objetos digitales que introduciremos en las escenas, y que pueden ser creados a partir de un menú básico de figuras geométricas proporcionadas por Unity o mediante la importación de objetos digitales más complejos desarrollados en herramientas de diseño gráfico como Blender.

Una de las características más importantes a tener en cuenta de los objetos 3D es la de ***Parenting***. Esta característica permite atribuir a un determinado objeto uno o más objetos, de manera que se establece una jerarquía en la cual cada objeto pasa a ser padre o hijo de otro u otros objetos en caso de que se necesite una dependencia entre ellos. Por ejemplo, si queremos que el objeto X se desplace siguiendo a otro objeto Y que también se esté desplazando, debemos hacer a X hijo de Y.

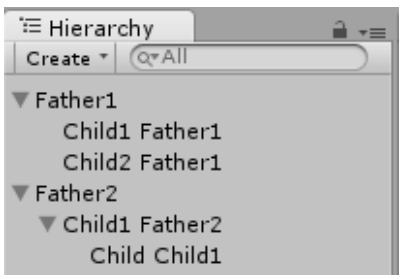


Figura 8. Ejemplo de jerarquía entre objetos.

Un problema derivado de la jerarquía de objetos, es que el sistema de referencia de rotación, posición y escala de los hijos, ya no depende del eje de coordenadas absoluto dentro del mundo Unity, sino que dependerán del sistema de referencia del padre. Para evitar estos problemas, se pueden usar los objetos vacíos, que son otro tipo de objetos 3D que viene predefinido en Unity. Por tanto, estos objetos vacíos facilitan que las cualidades del hijo no varíen en caso de que se modifiquen las del padre.

3.4.1.3. UI (User Intergace) y Canvas

La herramienta UI de Unity permite introducir elementos que no forman parte de la realidad aumentada, sino que son elementos que se ven fijos en la pantalla del dispositivo y con los cuales el usuario puede interactuar.

Cada vez que queramos introducir un elemento UI, se crea automáticamente un *Canvas*. El *Canvas* es el área dentro del cual van a estar colocados todos los elementos UI que se vayan introduciendo. Por consecuencia, todos estos elementos serán hijos del *Canvas*.

En concreto, en este proyecto se han utilizado elementos UI tales como **Text**, que muestra un texto en la pantalla del dispositivo, o **Input Field**, que sirve para que el usuario pueda introducir un texto mediante teclado en un cuadro de texto.

3.4.1.4. Particle System

Un *Particle System* es un efecto que se le puede añadir a un objeto 3D y que consiste en un conjunto de partículas que emanan de él.

3.4.1.5. Material

Un *Material* es un elemento de renderizado que se puede atribuir a un objeto 3D para que la superficie de éste tome un determinado color, brillo, transparencia u opacidad, entre otros.

3.4.1.6. Collider y Trigger

El ***Collider*** es una característica que presentan todos los objetos 3D, aunque puede desactivarse en caso de que no sea necesaria. Es una estructura imaginaria que rodea a los objetos y que, en general, tiene la misma forma que el objeto, aunque su forma puede editarse y modificarse. Suelen utilizarse para detectar el contacto entro dos o más objetos.

Además, podemos hacer que el *Collider* también sea un ***Trigger***. En este caso, no sólo detectará el choque con otro objeto, sino que también detectará si el objeto está dentro de un determinado espacio.

3.4.1.7. Rigid Body

Rigid Body es una característica física que se le puede introducir a un objeto. Por defecto, todos los objetos 3D son entes que pueden ser atravesados por otros objetos. Al añadir la característica del *Rigid Body*, el objeto pasará a comportarse como un físico, por ejemplo, dispondrá de masa, de forma que, si otro objeto colisiona con él, éste no le atravesará, sino que chocará con él.

3.4.1.8. Touch

El *Touch* es una interacción con el usuario a través del dispositivo móvil que permite detectar si éste ha tocado la pantalla del dispositivo.

Con el *Touch* podemos obtener información acerca de la posición en la que ha tocado el usuario en la pantalla o si el *Touch* se ha producido sobre algún elemento, de forma que se pueda producir algún tipo de evento o acción.

3.4.1.9. Script

Un *Script* es un fichero en el que viene recogido el código de programación que permite introducir en la aplicación todas y cada una de las funcionalidades que se necesitan para que ésta funcione.

3.4.1.10. Librerías de Unity

Gracias a las librerías proporcionadas por Unity, podemos usar determinados elementos y funciones incorporados en el lenguaje de programación que nos van a facilitar darle los efectos pertinentes a los objetos del entorno virtual.

Las librerías de Unity facilitan a la programación el uso de objetos tales como Game Object, Particle System, Text, etc.

Además, proporcionan a la programación una serie de funciones que van a facilitar la búsqueda de los objetos dentro de una escena o la creación o acción sobre alguno de los objetos de la misma.

3.4.1.11. Trail Renderer

El *Trail Renderer* es un efecto que se le puede añadir a un objeto y que consiste en que éste, al moverse o desplazarse, dejará tras de sí un rastro en forma de línea a lo largo de todo el recorrido realizado por el objeto.

3.4.1.12. Shader

Cada objeto 3D tiene un *Material* asignado. Este *Material* está compuesto por una serie de texturas y un *Shader* que le dan un aspecto característico.

El *Shader* es, por tanto, el elemento de renderizado dentro del Material de un objeto que tiene como función darle un aspecto más allá de un color o una textura.

Gracias al *Shader*, podemos darle al objeto un aspecto específico que debe ser previamente establecido mediante un código de programación específico.

3.4.2. Vuforia

Vuforia es un SDK (*Software Development Kit*) creado por la compañía Qualcomm. Permite el reconocimiento de imágenes y la proyección de realidad aumentada mediante dicho reconocimiento de imágenes.

En este caso, se ha usado la versión 5.0.5 de Vuforia, la cual se debe importar en Unity para poder utilizar la funcionalidad de realidad aumentada que proporciona Vuforia.

Para poder crear con Vuforia dicho entorno de realidad aumentada, se van a necesitar dos elementos básicos, sin los cuales no podremos acceder a la realidad aumentada: un *Image Target* y un *Frame Marker* y la *AR Camera*.

En la Figura 9 [18], podemos observar un esquema de un entorno de simulación de realidad aumentada a la hora de trabajar con Vuforia.

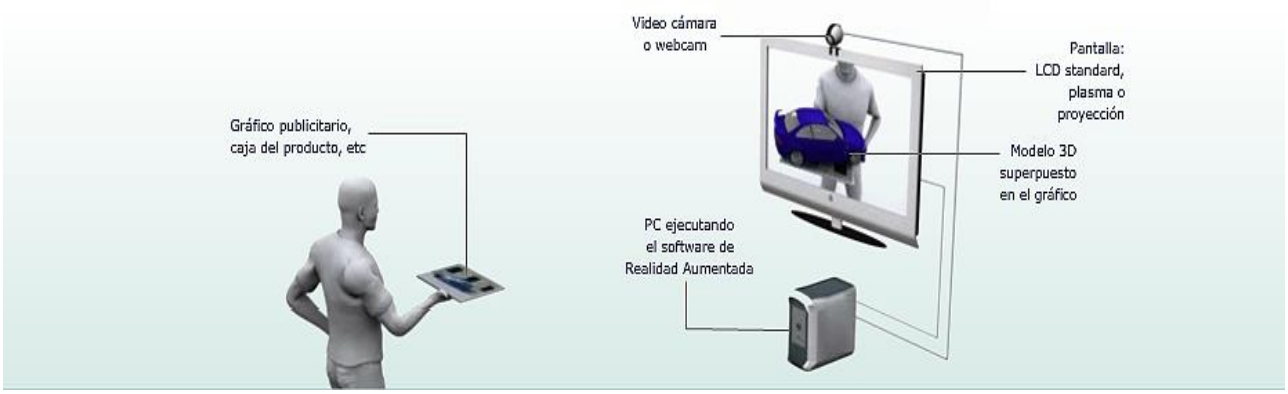


Figura 9. Escenario de Vuforia para realidad aumentada.

3.4.2.1. Image Target y Frame Marker

En primer lugar, el ***Image Target*** es cualquier imagen que Vuforia permite reconocer a través de la cámara del dispositivo móvil y que actuará como base sobre la que se proyectarán las escenas de la aplicación. En este caso, se han usado las *Stones* (Figura 10 [19]) incluidas en Vuforia como *Image Target*.

En segundo lugar, un ***Frame Marker*** es un marcador que contiene un patrón determinado, que será reconocido por Vuforia al igual que pasaba en el caso del *Image Target* y que nos permite mover objetos específicos dentro de una escena.

Cada *Frame Marker* tiene un número de identificación o ID asociado y un patrón característico que le permite diferenciarse del resto de *Frame Markers*. En este caso concreto, hemos usado el *Frame Marker* con ID 0 (Figura 11 [20]).

Figura 10. Image Target: Stones.

Figura 11. Frame Marker ID 0.

3.4.2.2. AR Camera

En Unity siempre hay una cámara, la *Main Camera*, que es el dispositivo dentro de las escenas encargado de transmitir la imagen de un determinado escenario a través de la pantalla del dispositivo sobre el que se esté observando el videojuego o aplicación desarrollado en Unity.

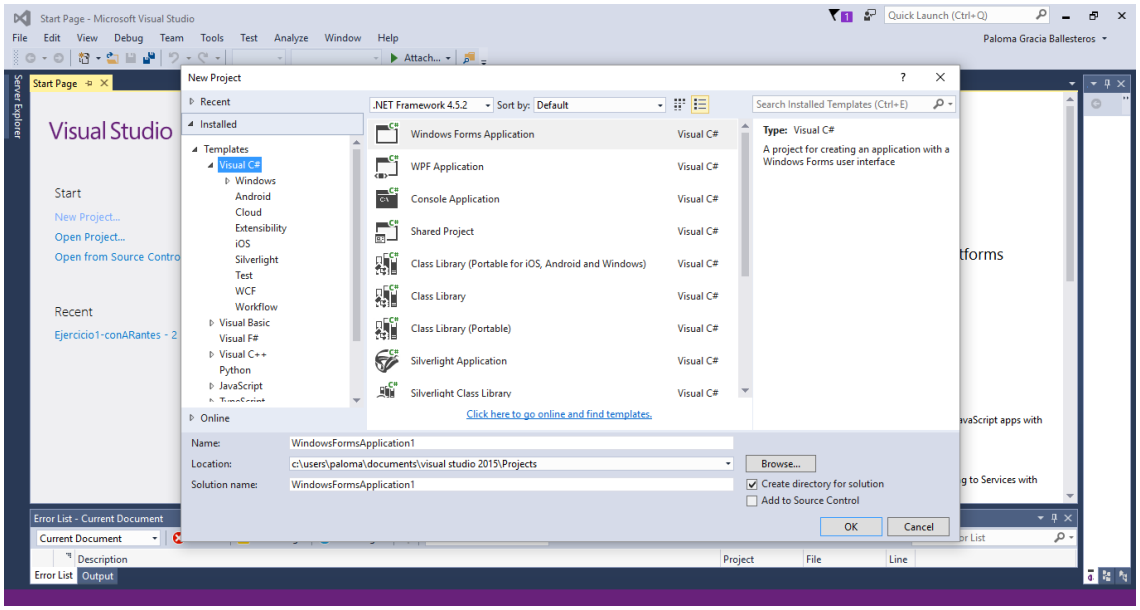
Sin embargo, cuando se requiere realidad aumentada, la *Main Camera* es sustituida por la *AR Camera*. Ésta será la encargada de reconocer los objetos del entorno real a través de la cámara del dispositivo para poder así reconocer determinados patrones sobre los cuales proyectar la realidad aumentada. Además, es la que proporciona algunos aspectos como la calidad del renderizado o el número máximo de objetos del entorno real que deben ser reconocidos.

3.4.3. Microsoft Visual Studio: C#

MS Visual Studio es la herramienta de *software* utilizada para la depuración y desarrollo del código de programación de los *scripts*. Facilita estas tareas ya que, por una parte, permite la depuración de forma sencilla permitiendo incluso depurar el código sobre la escena correspondiente de Unity y, por otra parte, ayuda a la escritura del código prediciendo en tiempo real el nombre de las funciones y variables y advirtiendo de errores que haya en el código.

Unity nos permite trabajar principalmente con C# y JavaScript, aunque también pueden ser usadas algunas implementaciones de otros lenguajes como Boo, que es una implementación de Python, u otros lenguajes .NET.

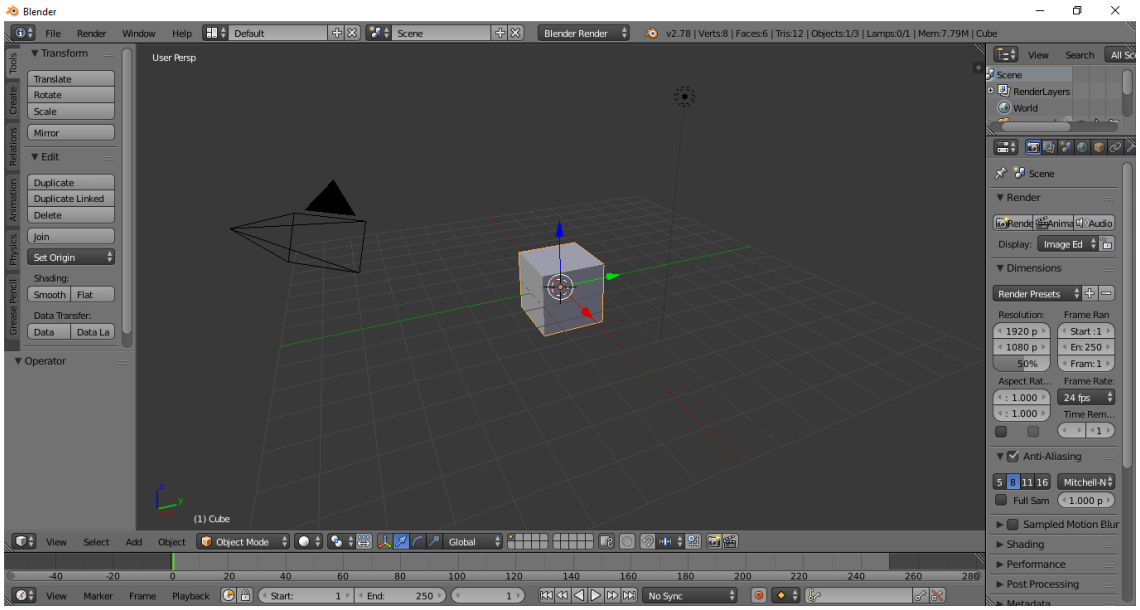
Más concretamente, para este proyecto decidimos usar **C#** ya que los manuales de Unity en su mayoría están orientados a dicho lenguaje y, además, el rendimiento de ambos será muy similar.



3.4.4. Blender

Blender es una plataforma de modelado y creación de objetos 3D que facilita la disponibilidad de objetos 3D diseñados por nosotros mismo y que podemos usar en Unity una vez creados en caso de no existir un objeto 3D con las características que necesitemos.

Unity admite varios formatos para la importación de objetos 3D modelados. Entre ellos, se encuentra el formato propio de proyectos modelados en Blender, que es una plataforma de uso libre que fue elegida para el diseño de los objetos 3D requeridos en nuestro trabajo.



3.5. Sistemas operativos móviles

Un sistema operativo es un junto de órdenes y programas que controlan los procesos básicos de un dispositivo de computación y permiten el funcionamiento de otros programas implementados por el mismo desarrollador o por terceros. [21]

Un sistema operativo gestiona los recursos de *hardware* y provee servicios a los programas de aplicación de *software*, ejecutándose en modo privilegiado respecto de los restantes.

La principal característica de los sistemas operativos móviles es que estos son más ligeros debido a las limitaciones *hardware*, energéticas y disipación del calor que éstos tienen frente a otros dispositivos de computación como los grandes servidores o incluso un ordenador de sobremesa de propósitos generales.

Un sistema operativo móvil suele partir de un sistema operativo de propósitos generales el cual ha sido adaptado para que este realice únicamente funciones específicas ahorrando así recursos o aligerándola de carga de servicios.

Entre las funciones principales de los sistemas operativos móviles se encuentran:

- Facilitar al usuario la resolución de determinados problemas y operaciones mediante la ejecución de aplicaciones móviles.
- Hacer que el usuario pueda tener una experiencia de uso del *hardware* lo más fácil y cómoda posible.
- Proporcionar al usuario un uso eficiente de los recursos del dispositivo móvil.

En el mercado de sistemas operativos móviles en la actualidad podemos encontrar dos sistemas operativos móviles muy utilizados y varios más que son tienen una cuota de mercado muy baja y luchan por hacerse un hueco en él.

Android es el sistema operativo móvil más utilizado en la actualidad, actualmente algo menos de 8 de 10 dispositivos móviles vendidos vienen con dicho sistema operativo, le sigue iOS este sistema operativo diseñado específicamente y en exclusiva para dispositivos vendidos por Apple vende algo más de 1 de cada 10 dispositivos móviles con este sistema operativo, luego el resto son repartidos entre Windows Phone, Blackberry, Firefox OS, Ubuntu y Tizen, de estos sistemas operativos móviles los más destacables y los cuales tienen presencia en el mercado, aunque muy pequeña, son Microsoft Windows Phone y Blackberry.

Los sistemas operativos móviles que hemos nombrado son los que mayor presencia cuentan en el mercado actualmente, hace algunos años, no tantos, en el mercado existían otros sistemas operativos móviles el cual uno de ellos dominaba el mercado estando presente en la mayoría smartphones de la época, actualmente casi desaparecido Symbian, y Bada, una alternativa a él. Ambos sistemas operativos móviles estaban basados en Linux.

A pesar del gran éxito que obtuvo Symbian, grandes empresas como son Google y Apple les ganando terreno con sus sistemas operativos Android e iOS respectivamente. [22]

A continuación, se presentarán algunos de los sistemas operativos para dispositivos móviles que más se han popularizado en los últimos años, de los cuales pondremos mayor atención en Android e iOS, que son los más utilizados, tal y como podemos observar en la Figura 12 [23].

Además, se hará hincapié en Android, ya que es el sistema operativo que necesitaremos para este proyecto.

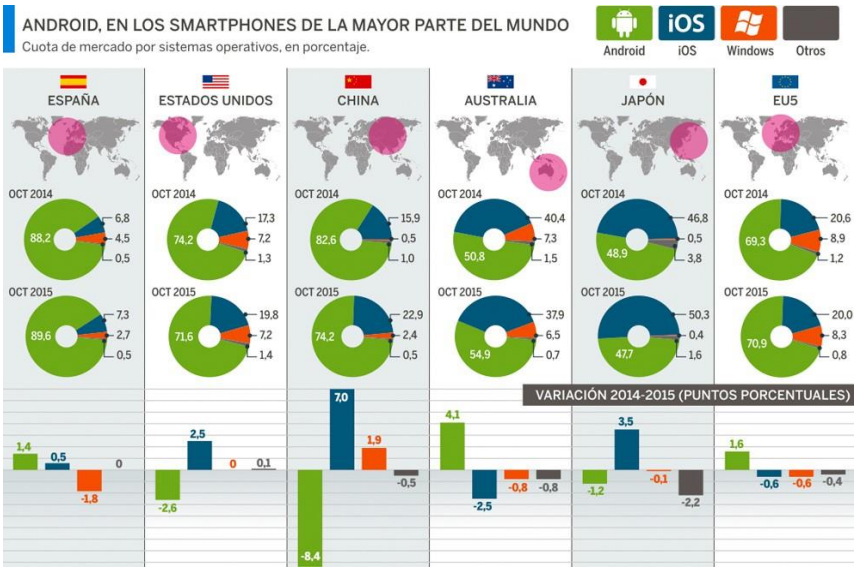


Figura 12. Uso de los diferentes sistemas operativos móviles.

3.5.1. Android

La primera versión de Android, Android 1.0, fue lanzada al mercado en noviembre del año 2007, cuando dicho sistema operativo se dio a conocer, aunque no fue hasta el año 2008 cuando su uso se expandió gracias al lanzamiento de los primeros dispositivos en los que iba incorporado, entre los que cabe destacar el HTC Dream, que fue el primer dispositivo comercializado con Android. [24]

Este sistema operativo fue creado por Google junto con la *Open Handset Alliance* y está basado en Lunix, ya que su kernel se fundamente en éste. [24]

Una de las características más importantes de este sistema operativo es que tiene carácter abierto, es decir, que sus derechos y código fuente son de dominio público. Fundamentalmente, esta característica ha sido la que ha hecho que actualmente sea el sistema operativo más usado a nivel global. [24]

Este sistema operativo tiene una arquitectura conformada principalmente por cinco capas, todas ellas de *software* libre, y cada una de las cuales tiene una función diferente. Estas capas son: kernel de Linux, librerías nativas, Android Runtime, Framework de aplicaciones y las aplicaciones en sí. [25]

Figura 13. Arquitectura Android.

El **kernel de Linux** conforma el núcleo del sistema operativo y es donde se llevan a cabo funciones indispensables en cualquier sistema operativo tales como la seguridad, la pila de protocolos o el control de los diferentes procesos que se lleven a cabo. Además, esta capa es dependiente del *hardware* ya que es la encargada de hacer de intermediaria entre dicho *hardware* y el resto de capas de la arquitectura.

Las **librerías nativas** son un conjunto de librerías necesarias para el buen funcionamiento del sistema operativo ya que algunas de ellas incluyen código referido a la representación gráfica tanto como en 2D como en 3D, al procesamiento de bases de datos, la navegación por internet, los protocolos de servicios de conexión segura como SSL (*Secure Socket Layer*) o al procesamiento de audio y vídeo.

El **Android Runtime** es el entorno de ejecución de cada una de las aplicaciones que están corriendo sobre el sistema operativo. El Runtime será, por tanto, el encargo de llevar a cabo la compilación del código de la aplicación a través de una máquina virtual capaz de correr en el dispositivo de acuerdo a sus limitaciones de memoria y velocidad de procesamiento.

Un **Framework** es un conjunto de normas definidas que estandarizan la estructura de las aplicaciones, de forma que facilitan el mantenimiento y desarrollo de las mismas a largo plazo y la reutilización del código. Concretamente, el Framework de aplicaciones de Android define estándares sobre el acceso a datos de las aplicaciones, la notificación de alertas provenientes de éstas, el ciclo de vida de cada una ellas y su visualización sobre el dispositivo Android.

La aplicación de este proyecto será utilizada sobre dispositivos con sistema operativo Android. Para que esto sea posible, el **Android SDK** debe ser instalado en el ordenador sobre el que se vaya a trabajar con Unity. De esta forma, se podrá construir el fichero con extensión .apk, que será el que permita instalar la aplicación en el dispositivo sobre el que se quiera desplegar.

3.5.2. Otros

Existen otros sistemas operativos móviles en el mercado, como pueden ser: iOS, perteneciente a la empresa Apple y diseñado para ser utilizado sobre sus dispositivos móviles tales como los iPhone o iPad Touch; Windows Phone, perteneciente a la compañía Microsoft y que actualmente se podría considerar como el más usado después de Android e iOS; u otros sistemas operativos como Firefoz, Ubuntu o Tizen, cuya relevancia es menor en el mercado ya que su uso es menos extendido.

4. Análisis de la aplicación

En esta sección, se analizarán el diseño y la implementación de todo lo relacionado con las fases del desarrollo de la aplicación.

Primeramente, nos centraremos en el diseño de cada una de las escenas y se explicará lo que el estudiante debe hacer en cada una de ellas.

Después, se expondrá la implementación del diseño que se ha llevado a cabo para conseguir las simulaciones y funcionalidades requeridas en el proyecto. Aquí, se incluirán diagramas de estados para explicar el funcionamiento de cada escena, y se explicarán los diferentes métodos y clases usados en el código de la implementación.

4.1. Diseño de la aplicación

La aplicación está dividida en varias escenas, donde cada una de ellas corresponde a una etapa de cada ejercicio experimental diferente a realizar por el estudiante.

Todas las escenas, a excepción de la primera y la última, parten de un mismo patrón el cual servirá de base de diseño de cada una de ellas. Tomaremos como diseño base el mostrado en la Figura 14, y debido a que va a ser nombrado en numerosas ocasiones, lo vamos a denominar como **BS** a partir de este momento.

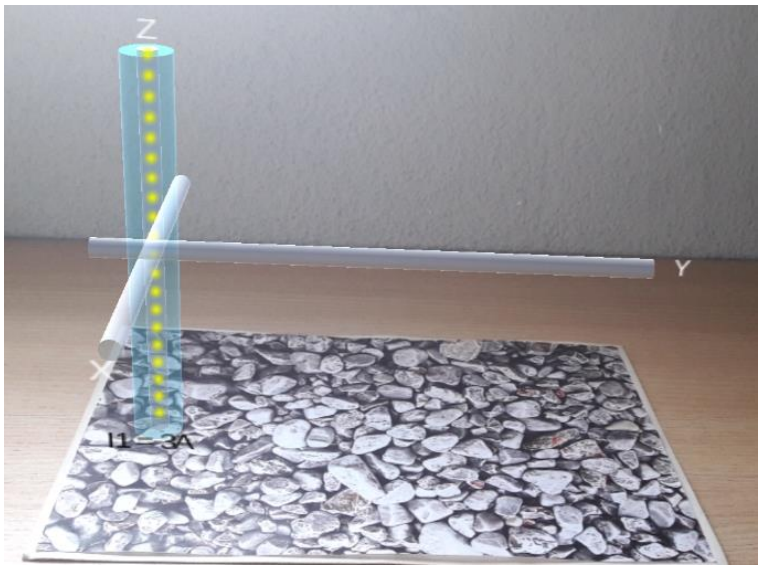


Figura 14. Diseño base de las escenas (BS).

Todas las escenas incluyen un *Canvas* con un elemento UI de tipo *Text* que permite incluir un texto que conforma el enunciado de cada ejercicio a realizar por el estudiante en cada una de ellas, y una serie de botones con los que éste deberá interactuar para llevar a cabo las actividades propuestas en cada escena.

A continuación, se explica cómo se ha llevado a cabo el diseño de las escenas y el ejercicio que el estudiante debe realizar en cada una de ellas.

Escena 0: Log In

El diseño de esta escena se compone únicamente de un objeto vacío al que se le ha atribuido un *script*. Este *script* será el que proporcione a la escena el diseño correspondiente.

Aquí, se mostrarán dos cuadros de texto en los cuales el estudiante deberá simplemente introducir su NIA, nombre y apellidos. Una vez introducidos, deberá proceder a la ejecución de la primera escena de ejercicios pulsando el botón de "*Log In*" situado debajo de los cuadros de texto, tal y como se muestra en la siguiente imagen.

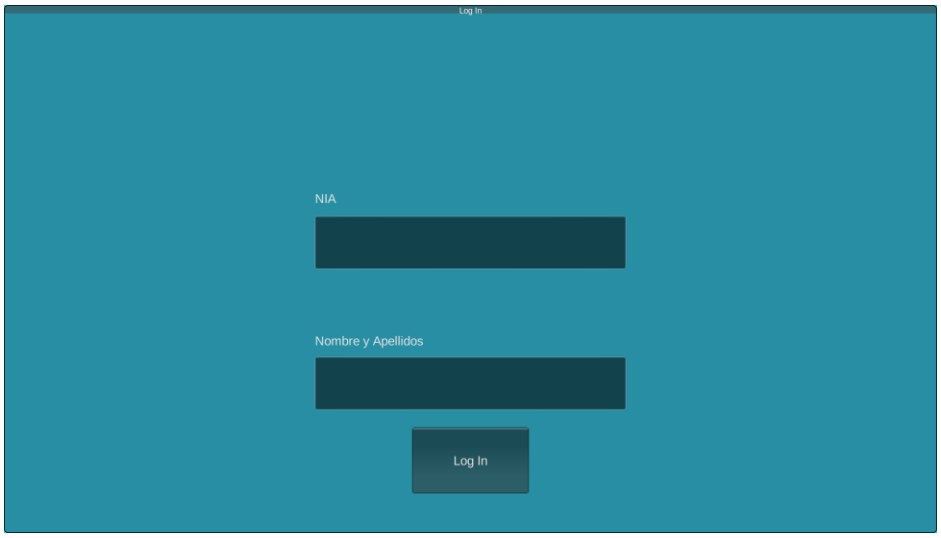


Figura 15. Escena de Log In.

Versiones posteriores de esta aplicación incluirán una base de datos para poder verificar que el estudiante que realice el *Log In* está matriculado en la asignatura de Física.

Escena 1: Familiarización con el entorno

Aquí, se le introduce al estudiante el diseño de la escena BS de forma que pueda familiarizarse con el entorno de los ejercicios que va a realizar a través de la realidad aumentada.

Dentro de este diseño, se le presenta al estudiante el cable 1, que está situado sobre el eje Z y por el que circula una corriente, la cual el estudiante puede cambiar de sentido para que sea ascendente o descendente mediante el botón “Corriente 1” presente en la pantalla.

Tal y como se ha mostrado en la Figura 15, este diseño se basa en la representación del eje de coordenadas de los hilos conductores (ejes X, Y, Z) y del cable 1 por el que circula una corriente.

Tanto los ejes como el cable 1 están representados por cilindros, a los cuales se les ha asociado un texto que indica el nombre del eje de cada uno de ellos y el valor de la corriente del cable 1 correspondientemente.

Además, la corriente que circula por el cable 1 está representada mediante un *Particle System* que emite partículas de color amarillo y que está situado en el extremo del hilo conductor.

Escena 2: Observar la dirección de los vectores de campo magnético B y el efecto que ejerce la fuerza

Primeramente, y partiendo de la escena BS a la que se le ha añadido una zona sombreada de color rojo, el estudiante, siguiendo las instrucciones incluidas en la escena, debe posicionar el marcador o *Frame Marker* sobre la zona sombreada roja.

Una vez se haya hecho esto, se detectará la colisión entre el plano que forma la zona sombreada y el *Frame Marker*. Al producirse esto, aparecerá el cable 2 por el que circula una corriente, diseñados del mismo modo que lo habíamos hecho para el cable 1 descrito en la escena BS. Además, aparecerá la línea de campo magnético que rodea al cable 1 y los vectores de campo y de fuerza posicionados sobre ella.

Por un lado, para representar los vectores de campo y fuerza magnética, se utilizó un objeto 3D con forma de flecha, ya que ésta nos permitirá representar de forma sencilla la dirección de dichos vectores.

Por otro lado, para el diseño de la línea de campo circular, se consideraron tres posibilidades de implementación, que son las siguientes:

- Línea de campo creada por un ***Trail Renderer*** de un objeto.

  Esta opción consiste en introducir en la escena un objeto al cual se le añade un *script* para hacer que éste realice un recorrido circular alrededor del cable 1. A este objeto además de le añade un *Trail Renderer* que dibujará la circunferencia alrededor del cable a medida que el objeto se desplaza en su movimiento circular.

  El problema lo tendríamos a la hora de querer hacer a un objeto hijo de la circunferencia, ya que ésta no es un objeto sino un efecto.
- Línea de campo creada por un ***Shader***.

  Esta opción consiste en introducir un objeto en forma de plano y añadirle un *Shader* que previamente se ha debido programar mediante código para dar al objeto un aspecto totalmente transparente y en el cual haya dibujada una circunferencia.

  El problema de este método es que la vista en planta de la aplicación no se vería bien ya que la zona que quedase por debajo del plano con el *Shader* no se observaría.

- Línea de campo formada por un **objeto 3D** creado en Blender.

  Este diseño consiste en la creación de un objeto tipo Torus en Blender, es decir, un objeto con forma de circunferencia cuyo interior está vacío.

  Esta fue la opción de diseño elegida y que nos evita los problemas previamente descritos en las otras opciones de diseño.

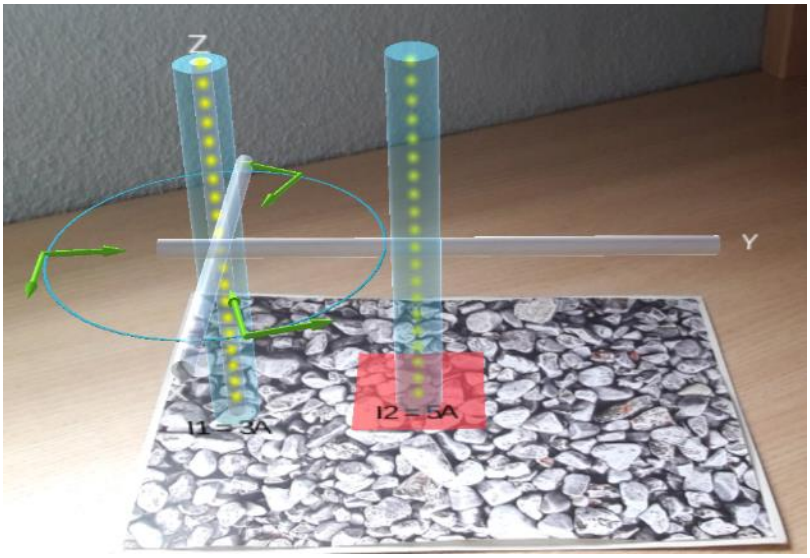


Figura 16. Escena 2.

En esta parte de los ejercicios el estudiante debe observar cómo cambia la dirección del vector de campo magnético B creada por los dos hilos al cambiar la dirección de las corrientes que por ellos circula mediante los botones “Corriente 1” y “Corriente 2” que aparecen en pantalla.

Después, se pide al alumno que pulse el botón “Simulación” una vez haya colocado las corrientes en las direcciones deseadas para que pueda observar el efecto que ejerce la fuerza magnética del cable 1 sobre el cable 2.

Al pulsar dicho botón, el cable 2 se alejará o se acercará al cable 1 en función del sentido de las corrientes tal y como se había descrito en el apartado 2 de esta misma memoria.

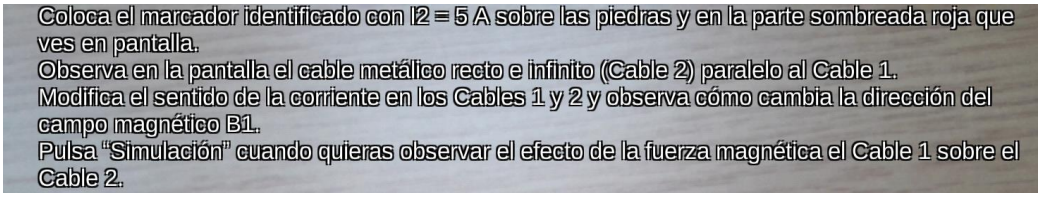


Figura 17. Enunciado mostrado en la Escena 2.

Escena 3: Observar la magnitud de los vectores de campo magnético B y el efecto que ejerce la fuerza

Este caso es parecido al que teníamos en la Escena 2. Sin embargo, aquí se introducen principalmente, no uno, sino tres zonas sombreadas rojas, cada una a una distancia diferente del cable 1. Además, el estudiante no sólo debe observar la dirección del vector del campo magnético y el efecto de la fuerza, sino que también debe observar cómo cambia la magnitud del mismo a medida que el cable 2 se acerca o se aleja del cable 1.

En esta escena se pide al estudiante que coloque el *Frame Marker* en la zona sombreada que prefiera en función de la distancia al cable 1 a la que desee observar el efecto de la fuerza y el cambio de la magnitud de los vectores. Para poder observarlo, al igual que en el caso anterior, debe pulsar el botón “Simulación” que aparece en pantalla cuando haya colocado el marcador y las corrientes en las posiciones deseadas.

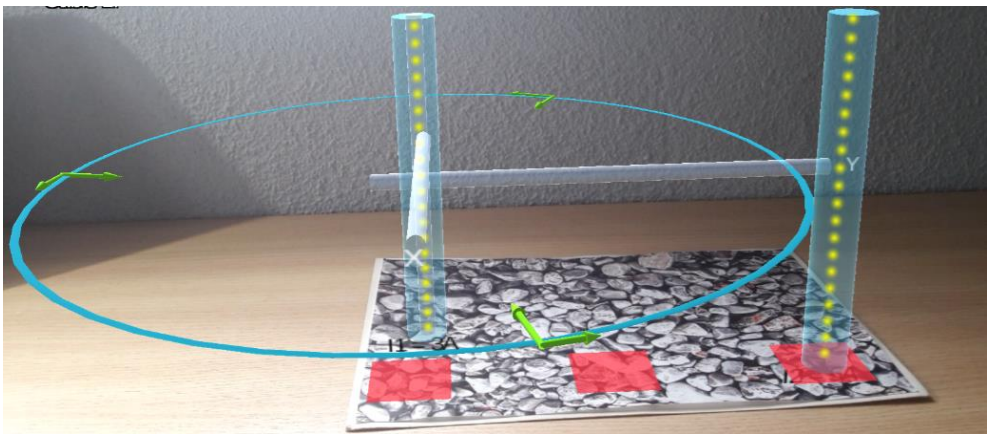


Figura 18. Escena 3.

Escena 4: señalar la dirección del vector de campo magnético B

En este caso partimos de la escena inicial BS a la cual se ha añadido una zona sombreada al igual que en el caso de la Escena 2.

Se pide al alumno que coloque el *Frame Marker* sobre la zona sombreada. Al hacerlo, aparecerá el cable 2 y la línea de campo concéntrica al cable 1. Sobre dicha línea, hay un punto negro que representa el origen del vector de campo magnético y ocho puntos azules alrededor de dicho punto de origen formando un cuadrado.

Con esta representación en pantalla, se le pide al alumno que toque sobre la pantalla del dispositivo en el punto azul que crea conveniente en función de la dirección del campo magnético en la que piense que éste debe estar. Al tocar sobre los puntos, aparecerá una flecha con origen en el punto negro y en dirección al punto azul que se tocó, y que representa el vector de campo magnético B.

No se podrá pasar al siguiente ejercicio hasta que se marque el punto correcto de la dirección del campo y aparecerá un mensaje de “Por favor, vuelve a intentarlo” cada vez que el estudiante toque sobre un punto incorrecto.

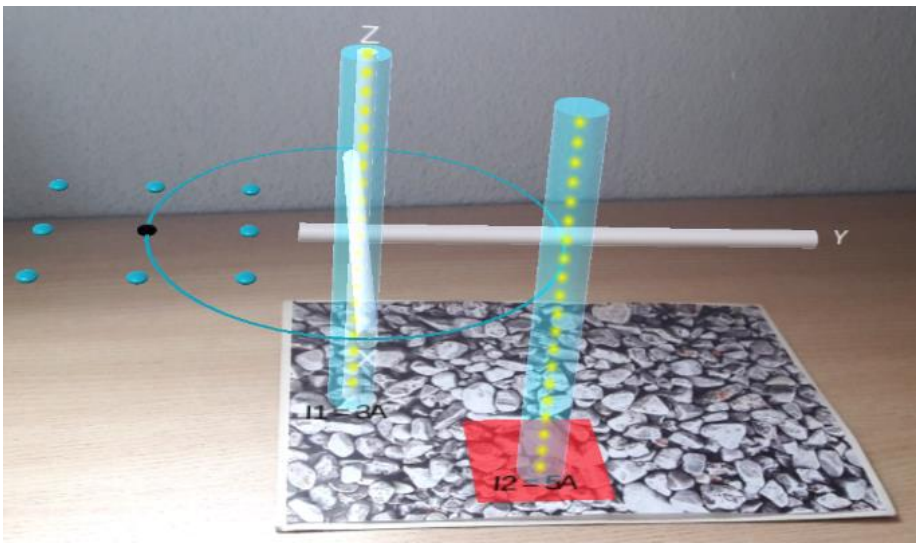


Figura 19. Escena 4.

Escena 5: señalar la dirección del vector de fuerza magnética

Esta escena consiste en lo mismo que la escena anterior, a diferencia de que en este caso no se debe señalar el punto de la dirección del vector de campo, sino el de fuerza magnética.

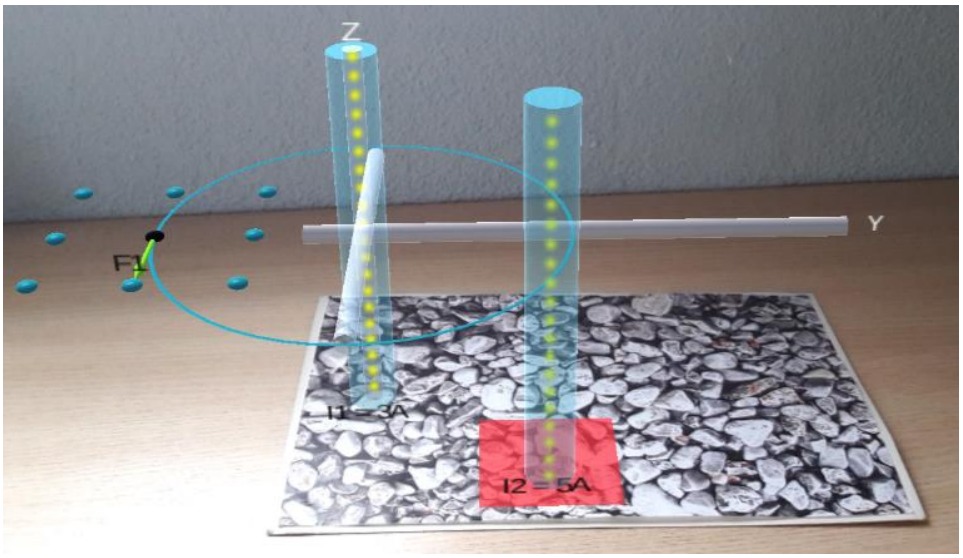


Figura 20. Escena 5.

Escena 6: fórmula de la fuerza magnética

El diseño de esta escena lo forma la escena de partida BS a la que se le ha añadido el cable 2 en la posición (0.4, 0.4, 0) del sistema de coordenadas de los cables conductores (valores expresados en metro). Además, aparecen representados la correspondiente línea de campo y las flechas que representan los vectores de fuerza y campo magnético creados por las corrientes de los hilos conductores.

También aparece en pantalla la fórmula correspondiente a la fuerza magnética. A esta fórmula le faltan dos de sus valores, los cuales se pide al estudiante que calcule e introduzca en dicha fórmula. El resultado final del cálculo, mediante la fórmula que se indicó en el apartado 2 de esta memoria, nos lleva a que se debe introducir como solución un valor de $-1.5 * 10^{-5}$ [î + ĵ] (N).

En caso de que el estudiante no introduzca los valores correctos, se irán mostrando unos diálogos de ayuda para facilitarle llegar al resultado correcto de forma guiada.

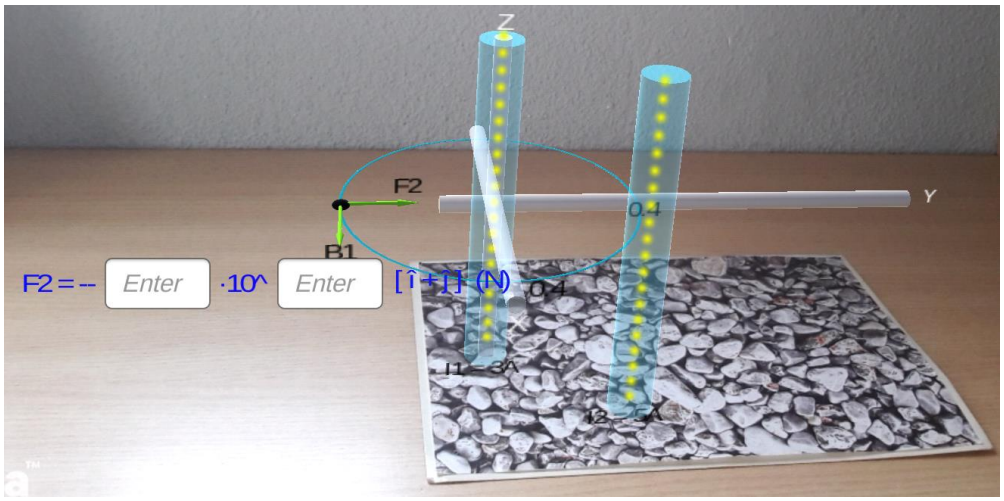


Figura 21. Escena 6.

Escena 7: fin de la aplicación

Esta escena corresponde con el final de la aplicación y la superación de todos los ejercicios y experimentos que debía superar el alumno.

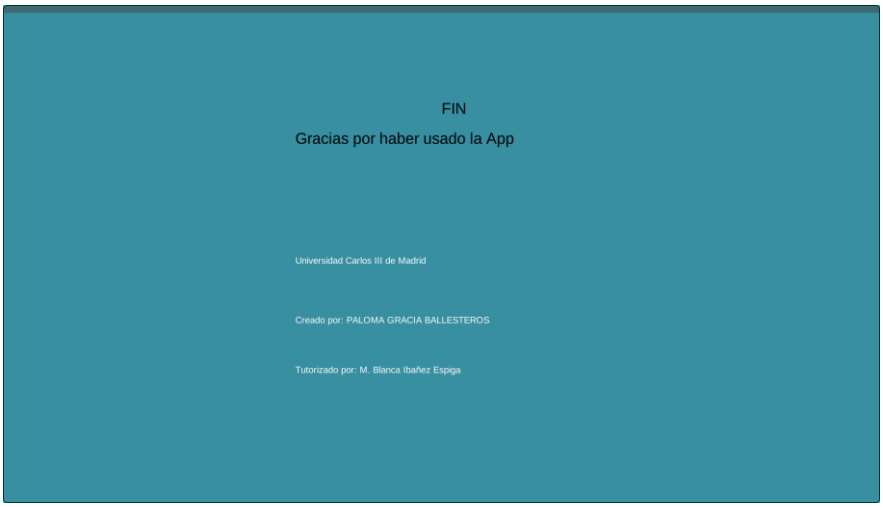


Figura 22. Escena 7.

4.2. Implementación de la aplicación

En esta sección, vamos a explicar todo lo relacionado con la implementación del sistema que se ha llevado a cabo mediante las herramientas de *software*.

Por una parte, se analizará todo lo relacionado con el código C#. Para ello, primero se explicará cómo se han estructurado los *scripts* de las diferentes escenas, y después se examinará más detenidamente el contenido de estos *scripts*.

Por otra parte, se expondrán los pasos que se han de seguir para empaquetar la aplicación a fin de que ésta pueda ser instalada en el dispositivo sobre el que se quiera ejecutar.

4.2.1. Implementación de la aplicación en C#

Para el perfecto funcionamiento de las escenas del proyecto, ha sido imprescindible el desarrollo de *scripts* de código de programación, este caso mediante el lenguaje C#.

En este apartado, se explicará cómo se ha llevado a cabo el desarrollo de estos *scripts* y qué es lo que nos ha aportado al proyecto cada uno de ellos.

Para que el lector tenga una mejor idea sobre la implementación de los mismo, primero se explicará el funcionamiento de la programación cuando trabajamos con Unity ya que se requiere una estructura específica y, después, se explicará la implementación concreta que se ha llevado a cabo en el proyecto.

En esta segunda parte donde explicaremos la implementación concreta del proyecto, haremos una distinción entre cuatro casos para los cuales necesitaremos los *scripts*, los cuales serán:

- Botones
- *Colliders*
- Escenas de *Log In* y fin

4.2.1.1. Clases y métodos utilizados en Unity

Cuando se realiza un proyecto en Unity, hay una estructura inicial que debemos seguir siempre a la hora de programar, la cual se muestra en la Figura 23.

```
using UnityEngine;
using System.Collections;

public class NewBehaviourScript : MonoBehaviour {

    // Use this for initialization
    void Start () {

    }

    // Update is called once per frame
    void Update () {

    }
}
```

Figura 23. Estructura de un proyecto C# en Unity.

Para poder comprender algunos de los conceptos que se explicarán posteriormente, primero debemos conocer qué es un ***frame***.

Un *frame* es el tiempo de refresco de cada uno de los píxeles que forman el entorno virtual, en este caso, de las representaciones de los objetos que observamos en la realidad aumentada. Éste, viene expresado en milisegundos y será mejor cuanto menor sea su valor ya que esto hará mejorar la calidad de las simulaciones.

A continuación, se explicarán las clases y métodos de la estructura inicial de los *scripts* tal y como aparecen en la Figura 23.

Clase MonoBehaviour

Cuando introducimos a un objeto 3D un componente de tipo *Script*, debe haber una conexión entre este componente y el objeto, de forma que se consiga la funcionalidad deseada para dicho objeto. Pues bien, la clase *MonoBehaviour* es la responsable de esta conexión entre el objeto y las funcionalidades desarrolladas en un *script*.

Método Start()

El método Start() es el primero que se ejecuta en un *script*. Es el método utilizado para la inicialización de los parámetros y variables que se vayan a utilizar en el mismo *script* en el que se encuentre dicho método. Éste, sólo se llama en caso de que el objeto al cual se haya atribuido esté activado.

Método Update()

Este método tiene como característica principal que se ejecuta cada *frame*, por lo que se debe tener especial precaución a lo que se escriba en él ya que se ejecutará de forma muy repetida y cada muy poco tiempo.

Al igual que ocurría con el método Start(), el método Update() sólo es llamado siempre y cuando el objeto en el que se encuentre esté activo.

Método FixedUpdate()

Como habíamos comentado antes, todo lo que éste incluido en el método Update() será ejecutado una vez por *frame*. Sin embargo, la frecuencia de los *frames* no es constante. Pero en ocasiones se necesita una mayor precisión en dicha frecuencia para poder llevar a cabo ciertas acciones sobre los objetos.

Para ello, Unity dispone del método FixedUpdate(), los cuales proporcionan que el código en ellos incluido sea ejecutado en intervalos de tiempo constantes.

Método OnGUI()

Este método nos permite introducir elementos que no forman parte de la realidad aumentada, sino que son creados sobre la pantalla del dispositivo. Por ejemplo, si queremos que en la pantalla aparezca una ventana con un determinado texto, cuadros de introducción de texto por teclado o botones debemos usar este método.

Para la mejor comprensión de este método, en la Figura 24 se incluye un ejemplo sencillo de código usando este método con el cual se crea un botón.

```
void OnGUI()
{
    if (GUILayout.Button("Press Me"))
        Debug.Log("Hello!");
}
```

Figura 24. Ejemplo de uso de método OnGUI().

Métodos Play() y Stop()

Estos métodos se usan para poder activar o desactivar respectivamente elementos del tipo *Particle System*.

Método SetActive(boolean x)

Gracias a este método, podemos activar (x=true) o desactivar (x=false) objetos 3D, es decir, hacerlos visibles o invisibles dentro de una escena.

Método LoadScene(string x)

Este método es el que nos va a permitir la aparición de la siguiente escena a aquella que se esté mostrando en el momento. Para ello, debemos introducir por parámetro el nombre de la escena a la que queramos cambiar mediante este método. Para su uso, debemos incluir la librería *Scene Management* en el *script* en el cual vaya a ser usado dicho método.

4.2.1.2. Desarrollo de la aplicación en C#

En el desarrollo de la aplicación, podemos distinguir entre cuatro tipos diferentes de script según su función: aquellos orientados a la creación de los botones con los que el usuario podrá interactuar, aquellos cuya funcionalidad es la de proporcionar las simulaciones del ejercicio, aquellos creados para conseguir obtener la colisión entre los planos de color rojo mostrados previamente en el diseño y el *Frame Marker* correspondiente y aquellos que proveen el código necesario para mostrar las escenas de *Log In* y fin de la aplicación.

Implementación de los botones

Para las escenas de la aplicación, fue necesaria la implementación de cuatro botones: “Corriente 1” y “Corriente 2” para permitir cambiar el sentido de las corrientes de los cables 1 y 2 respectivamente, “Simulación” para poder observar los efectos de las corrientes sobre los vectores de fuerza y campo magnéticos y “Continuar” para poder avanzar a la siguiente escena una vez finalizada la simulación de la escena actual en la que se encuentre el estudiante.

Para la creación de estos botones cabían dos posibles alternativas de diseño:

- Creación de botones mediante ***Canvas***

  Esta alternativa consiste en la introducción de un elemento UI de tipo *Button* dentro de un *Canvas*. A pesar de ser una alternativa más cómoda y rápida de implementar, ésta fue descartada ya que no permitía el movimiento de los cables a la hora de pulsar el botón “Simulación”. Esto es debido a que dichos botones sólo permiten la ejecución de una línea de código por cada vez que son pulsados, por lo que no permiten, por ejemplo, la ejecución de bucles dentro del código.

- Creación de botones mediante método **OnGUI()**

  Esta fue la alternativa elegida finalmente ya que con ella desaparecía el problema antes mencionado sobre la ejecución de bucles.

A continuación, se mostrará cómo fue implementado cada uno de los botones previamente mencionados.

- Botones “**Corriente 1**” y “**Corriente 2**”

  Tanto el cable 1 como el cable 2 están dotados de dos *Particle Systems*, uno al principio y otro al final de los mismos. Por tanto, la funcionalidad que nos deben proporcionar dichos botones será la de activar o desactivar dichos sistemas de partículas para conseguir así cambiar el sentido de las corrientes de los cables. Además, en los casos que sea necesario, deberá cambiar también la dirección de los vectores de campos y fuerza magnéticos al mismo tiempo que cambia la corriente.

  Para ello, se utilizarán las funciones Play() y Stop() para las corrientes y setActive(boolean x) para los vectores.

  En las Figuras 25 y 26, se muestra un diagrama de flujo del funcionamiento de dichos botones.

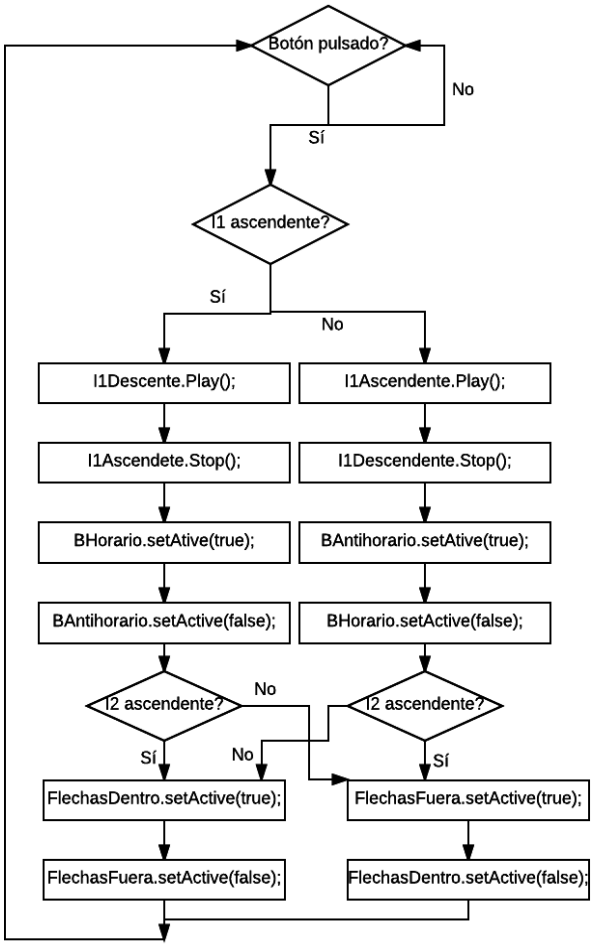


Figura 25. Diagrama de flujo botón “Corriente 1”.

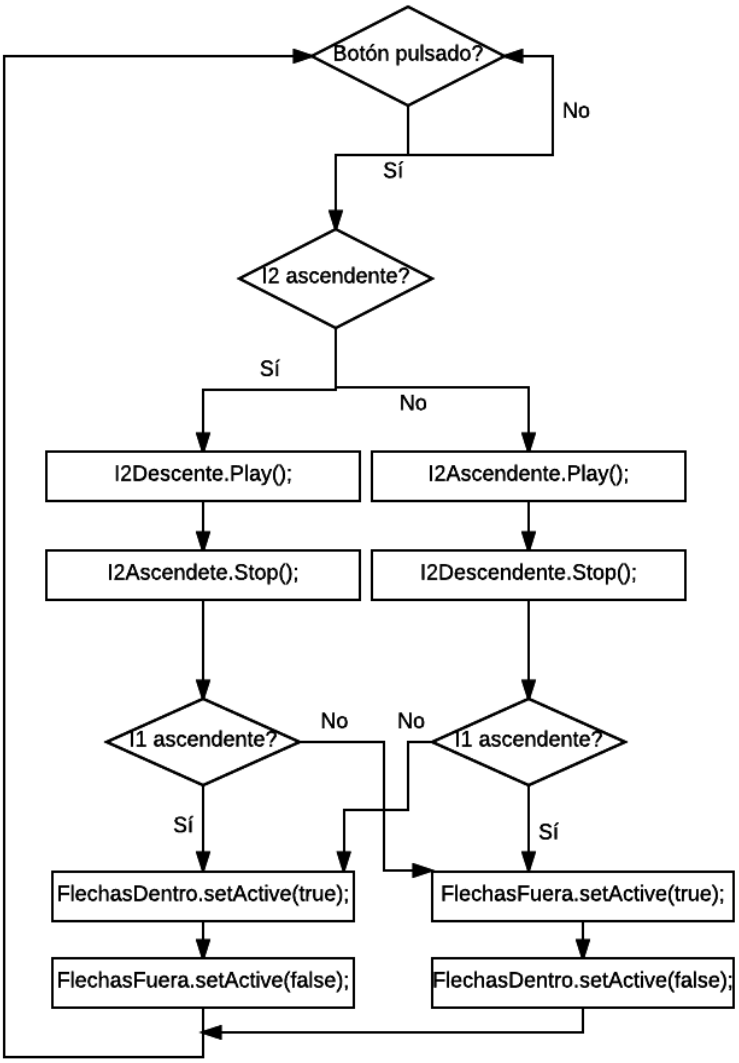


Figura 26. Diagrama de flujo botón "Corriente 2".

- Botón “**Simulación**”

  El botón “Simulación” está presente en las escenas 2 y 3.

  En el caso de la escena 2, el botón permite observar el efecto que ejerce la corriente del cable 1 sobre el cable 2, es decir, permitirá que el cable 2 se aleje del cable 1 si las corrientes de éstos circulan en el mismo sentido o se acerque al cable 1 en caso contrario.

  En el caso de la escena 3, el botón permite el mismo efecto que en caso anterior, pero ahora, además, hará que los vectores de fuerza y campo magnéticos aumenten o disminuyan a medida que el cable 2 se acerca o aleja del cable 1 (los vectores serán mayores cuanto más cerca se encuentren los cables).

  En la Figura 27 se muestra el diagrama de flujo del comportamiento que proporciona este botón al cable 2, a la línea de campo (Torus) y a las flechas de fuerza (F) y campo (B) magnético.

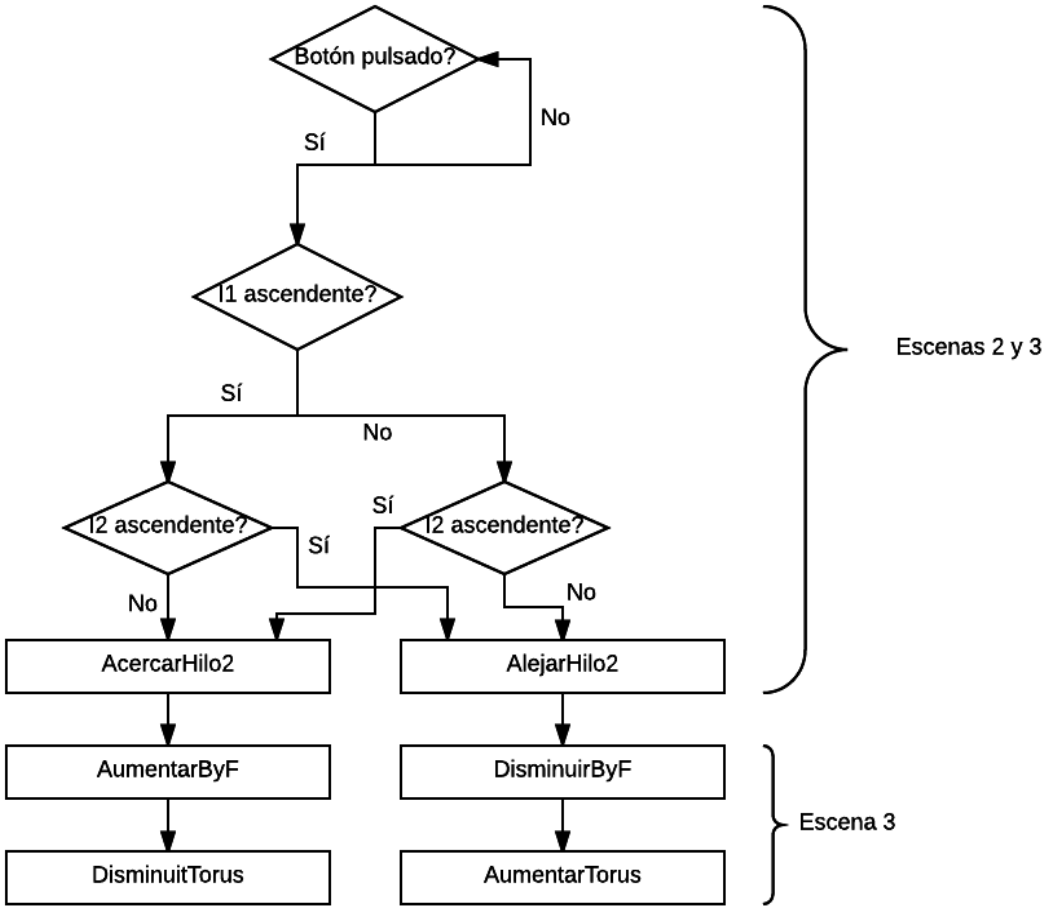


Figura 27. Diagrama de flujo botón “Simulación”.

- Botón “**Continuar**”

  Estos botones tienen como función permitir el cambio de una escena a la siguiente una vez el estudiante haya finalizado la simulación de cada de una de ellas.

  En la escena 0, este botón es sustituido por el botón de *Log In*, aunque es creado de la misma forma que el resto de botones de este tipo.

  El cambio de una escena a otra en Unity, se realiza mediante la librería *Scene Management* y a través del método LoadScene().

  No obstante, este botón tiene una funcionalidad especial en las escenas 4, 5 y 6, ya que se deberá comprobar antes de pasar a la siguiente escena que se cumpla una determinada condición. Esta condición, en el caso de las escenas 4 y 5 consiste en comprobar que el estudiante ha tocado sobre el punto correcto donde debe ir el vector correspondiente, y en el caso de la escena 6 la condición será que los valores a introducir en la fórmula de la fuerza sean los correctos, de tal forma que el diagrama de flujo sería como el mostrado en la Figura 28.

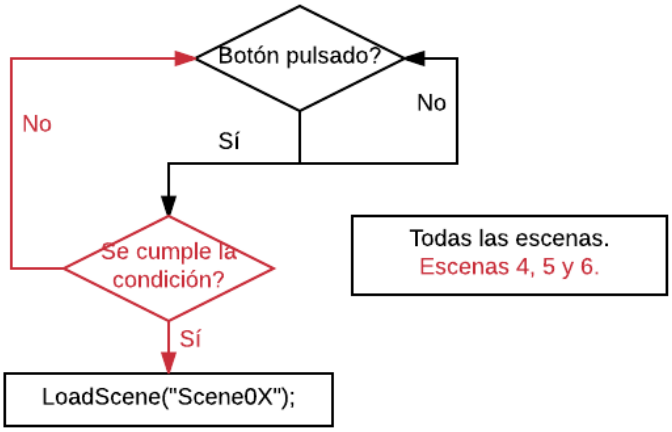


Figura 28. Diagrama de flujo botón “Continuar”.

Implementación de Colliders

Tanto en la escena 2 como en la escena 3, debemos usar el *Frame Marker* para hacer que aparezca el cable 2 junto con los vectores de campo y fuerza magnética.

Para ello, debemos realizar usar un script con el código necesario para que se pueda detectar la colisión entre el *Frame Marker* y la zona sombreada roja sobre la que se colocará el cable 2.

Para la correcta detección de la colisión entre ambos elementos, debemos realizar la implementación tanto del *Frame Marker* como de los planos que forman las zonas sombreadas rojas de la siguiente forma:

- Configuración del ***Frame Marker***

  Al *Frame Marker* debemos introducirle un *Collider* y ajustarlo de la forma más precisa posible para conseguir así una óptima detección a la hora de detectar la colisión.

- Configuración del **plano**

  Al igual que ocurre con el *Frame Marker*, al plano debemos introducirle un *Collider* de la misma que se ha explicado anteriormente.

  Además, debemos darle a dicho plano la propiedad de *Rigid Body*, para que el plano no pueda ser atravesado por el *Frame Marker,* sino que éste choque al encontrarse con el plano. Respecto a esta propiedad física, debemos también otorgarle la cualidad de ser "*Kinematic*". Gracias a esta propiedad, se consigue que el plano no se vea afectado si otro objeto ejerce sobre él una fuerza o colisión.

  También, debemos atribuirle un *script* de código, que será el responsable de hacer visible el cable 2, los vectores y la línea de campo cuando se produzca la colisión. Este script, denominado "OnTrigger", es el que se muestra en la Figura 29.

```
public class onTrigger : MonoBehaviour {

    private GameObject model;

    ParticleSystem sysDown2;

    // Use this for initialization
    void Start () {
        sysDown2 = GameObject.FindGameObjectWithTag("PS22").GetComponent<ParticleSystem>();
        model = transform.FindChild("Hilo2").gameObject;
        model.SetActive(false);
    }

    void OnTriggerEnter(Collider other) {
        Debug.Log("Enter Trigger");
        model.SetActive(true);
        sysDown2.Stop();
    }

    void OnTriggerStay(Collider other) {
        Debug.Log("Stay Trigger...");
    }

    void OnTriggerExist(Collider other) {
        Debug.Log("Exist Trigger");
        model.SetActive(false);
    }
}
```

Figura 29. Script OnTrigger.

En líneas generales, este código lleva a cabo lo siguiente: inicialmente, busca al cable 2 en la escena y lo hace invisible y, a continuación, tras haberse detectado la colisión del plano con el *Frame Marker*, hace al cable 2 visible. Como tanto la línea de campo como los vectores son hijos del cable 2, éstos se harán visibles o invisibles según lo sea o no su padre, por lo que no es necesario realizar la misma operación con ellos.

Implementación de las escenas de *Log In* y fin

La implementación de las escenas de *Log In* y de finalización de la aplicación se ha llevado a cabo del mismo modo, esto es, mediante la introducción de un *script* asociado un objeto vacío dentro de la escena. En este *script* se llevará a cabo la implementación de un método OnGUI().

Por una parte, en este método OnGUI() crearemos una ventana (*Window*) que ocupará el espacio completo comprendido por las dimensiones de la pantalla del dispositivo sobre el que se esté ejecutando la aplicación, tal y como se muestra en la Figura 30.

```
void OnGUI()
{
    GUIStyle mystyle = new GUIStyle();
    mystyle.fontSize = 34;
    estilo = mystyle;
    GUI.Window(0, windowRect, WindowFunction, "");
}
```

Figura 30. Método OnGUI() para la creación de las ventana.

Como podemos observar en la Figura 29, la creación de la ventana lleva asociado un método WindowFunction(). En este método, se escribirá el código necesario para introducir el texto que aparecerá dentro de la ventana o, en el caso del *Log In*, las barras de introducción de texto y el botón de *Log In* para pasar a la primera escena correspondiente a la primera parte del ejercicio.

4.2.2. Implementación de pruebas

En este apartado, procederemos a explicar las diferentes pruebas llevadas a cabo para comprobar el buen funcionamiento de la aplicación.

Estas pruebas que se han ido realizando a lo largo de todo el desarrollo del proyecto pueden dividirse en tres partes:

- Pruebas implementadas en Unity
- Pruebas implementadas en Microsoft Visual Studio
- Pruebas implementadas en un *Smartphone*

Gracias a todas estas pruebas realizadas a lo largo del proceso de implementación, se puede ir observando si el comportamiento de la aplicación cumple con los objetivos y requisitos que se persiguen para la misma.

4.2.2.1. Pruebas en Unity

Todos y cada uno de los cambios realizados en la implementación a lo largo del desarrollo de la aplicación se han ido probando en Unity, de forma que se pudiera comprobar si dichos cambios se habían implementado correctamente.

Las pruebas en Unity consisten en observar el comportamiento de la aplicación en lo que se conoce como ***Play Mode***.

El *Play Mode* es el estado en el que entra Unity cuando pulsamos el botón *Play* presente en él, y que podemos ver en la Figura 31. Gracias a éste, podemos probar la aplicación de realidad aumentada en la misma pantalla del ordenador a través de una *webcam* o de la misma cámara del ordenador, sin necesidad de pasar la aplicación a un dispositivo externo o a un simulador de dispositivo.

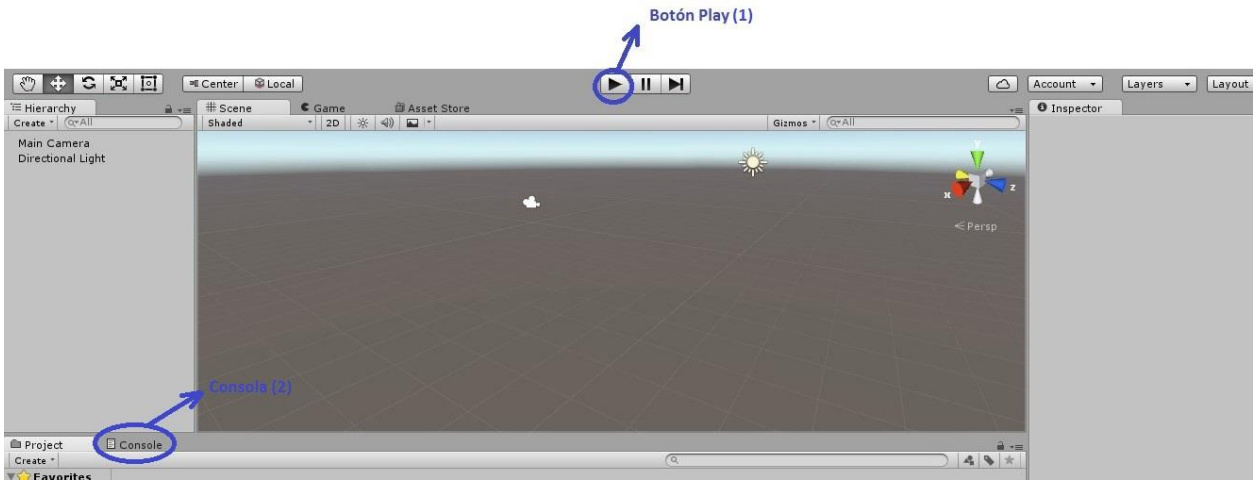


Figura 31. Botón para activar el Play Mode (1) y consola (2).

En nuestro caso, se ha utilizado un ordenador portátil con un sistema operativo Windows 10 y una capacidad de 8GB de memoria RAM. Esto será una ventaja a la hora de usar el *Play Mode* en Unity ya que éste tendrá un consumo máximo de aproximadamente 210 MB al hacer las simulaciones.

Además, Unity dispone de una **consola** donde se muestra información de posibles errores o *warnings* que pueda haber en el código de programación, lo cual nos ayudará a detectar de forma más rápida problemas en el código.

También a través de la consola, podemos ver mensajes que hayamos introducido en el código de forma que podamos tener acceso al valor de determinados parámetro y variables o saber qué sección de código se está simulando.

4.2.2.2. Pruebas en Microsoft Visual Studio

MS Visual Studio nos va a permitir llevar a cabo las pruebas de **depuración** del código implementado en cada uno de los *scripts* de los que dispongamos.

Una de las características más importantes y de mayor interés a la hora de trabajar con Unity y que MS Visual Studio nos va a proporcionar, es la opción "*Attach to Unity and Play"*. Esta opción permite aunar las pruebas en Unity previamente mencionadas, con la depuración del código de los *scripts*.

De esta forma, podemos conocer lo que sucede en los *scripts* en tiempo real en cada *frame* de forma más detallada al interactuar con el diseño de la realidad aumentada durante el *Play Mode*.

Además, podemos introducir en los *scripts* elementos como los ***breakpoints***. Estos *breakpoints* harán que la ejecución del *script* se pare justo antes de alcanzar la línea de código en la que se encuentre. Al llegar a este punto, el *Play Mode* también se detendrá. Así, podremos realizar un estudio más detallado de lo que está ocurriendo en la simulación de la aplicación.

4.2.2.3. Pruebas en un Smartphone

Para testar el resultado de la implementación de nuestra aplicación, ésta ha sido instalada y probada en un dispositivo móvil, para poder así tener constancia de cómo será la experiencia para el usuario final de la aplicación.

Para estas pruebas, se ha utilizado un *Smartphone* con sistema operativo Android 6.0.1 que dispone de una pantalla táctil de 5.2'' (132.2 mm) de tamaño y una cámara digital CMOS de 13.0 MP de resolución.

Gracias a estas pruebas, se pudieron ajustar algunos elementos del diseño de forma más precisa que a través de las pruebas sobre Unity.

Los elementos que estas pruebas nos permitieron reajustar fueron los siguientes:

- **Botones**

  Los botones que aparecían en pantalla se veían demasiado pequeños a la hora de utilizar la aplicación sobre un *Smartphone*. Esto fue debido a que en el *Play Mode* de Unity, los elementos se observan con mayor tamaño. Así pues, se puedo proceder a cambiar el tamaño de dichos botones.

- **Texto para los enunciados de los ejercicios**

  Al igual que ocurría con los botones, el tamaño del texto en el que aparecía el enunciado de los ejercicios de cada escena no era lo suficientemente grande. Con estas pruebas, se pudo aumentar el tamaño y cambiar el color de dicho texto, al principio rojo, pero que después se cambió con el aspecto que aparece en la siguiente imagen de forma que pudiera verse mejor. Además, con eso se consiguió que dicho texto pudiera observarse de forma clara tanto si tenemos un fondo claro como si tenemos un fondo oscuro a través de la cámara del dispositivo.

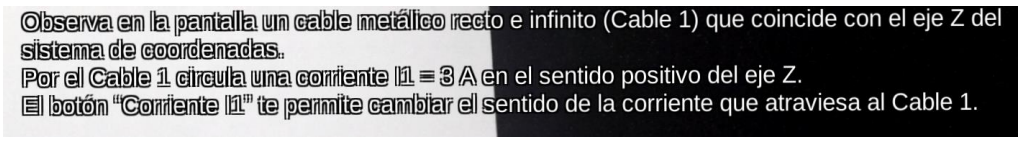


Figura 32. Aspecto del texto de los enunciados.

4.2.3. Empaquetado de la aplicación

Una vez hayamos finalizado todo el proceso de desarrollo de la aplicación en Unity, llega el momento de exportarla a un *Smartphone* o *tablet*, de forma podamos facilitársela al estudiante para que pueda instalarla en dicho dispositivo para que realice los ejercicios de la misma. Es en este punto donde debemos empaquetarla para poder después instalarla en el dispositivo móvil.

El término empaquetado se refiere a la creación de un paquete o fichero que contenga todos los archivos y ejecutables necesarios para poder llevar a cabo la instalación de la aplicación en el dispositivo móvil correspondientes.

Por tanto, en este fichero se encontrarán elementos tales como las escenas, los *scripts*, los objetos 3D que hayan sido importados o las librerías requeridas.

En Android, los ficheros resultantes del empaquetado son de extensión .apk (*App PacKage*), y serán los que permitan la instalación de la aplicación en el dispositivo.

Lo primero que debemos hacer para realizar el empaquetado es acceder en Unity a la sección ***File → Build Settings → Player Settings***. Aquí, aparecerá un panel como el mostrado en la Figura 33. Dentro de este panel y en la sección de Android, debemos configurar algunos de los parámetros de la aplicación.

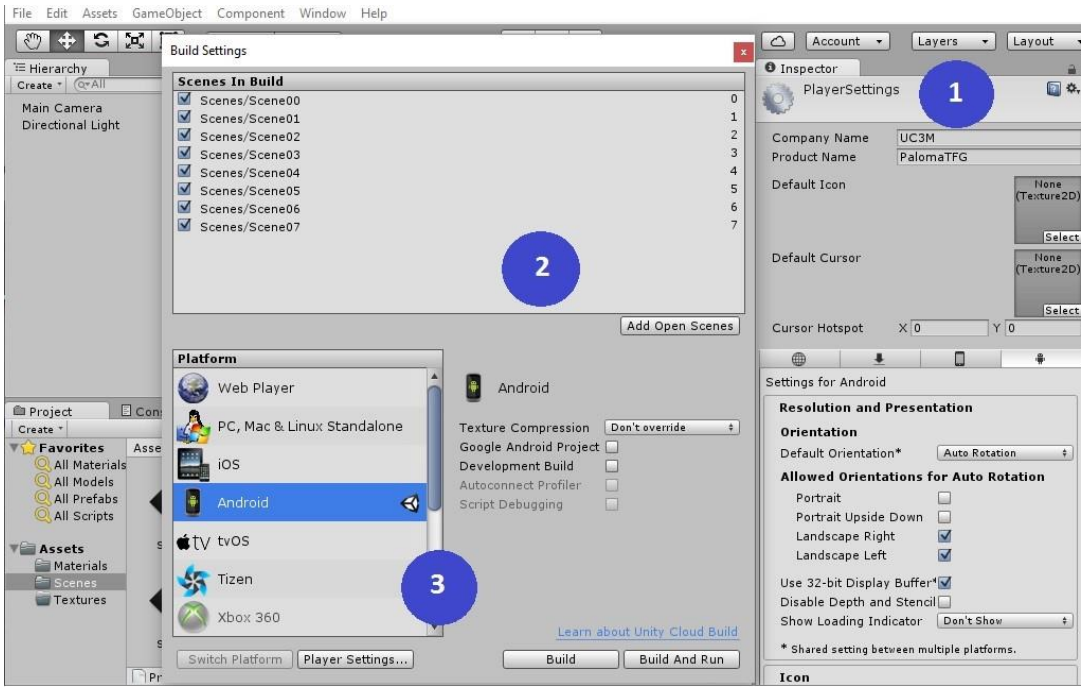


Figura 33. Pasos para el empaquetado de la aplicación.

Los parámetros más importantes a configurar dentro de *Player Settings* son los siguientes:

- **Nivel mínimo del API de Android** con el que se puede usar la aplicación.

  En este caso, se ha configurado la versión Android 2.3.1 como la más antigua para usar nuestra aplicación, que corresponde con el nivel 9 de API (recordemos que la última versión de Android es la Android 8.0, lanzada en febrero del año 2017, correspondiente al nivel 26 del API).

- **Versión** de la aplicación.

  Número de versión de nuestra aplicación. En nuestro caso, será la versión 1.0.

- **Bundle Identifier**

  El *Bundle Identifier* es un nombre que debemos darle a nuestro paquete de aplicación para identificarlo como nuestro. Tiene un formato por defecto que es com.Company.ProductName el cual debemos cambiar obligatoriamente para poder realizar en empaquetado. A este parámetro le hemos puesto el nombre de com.Paloma.TFG, de forma que podamos identificarlo de manera sencilla.

- **Orientación** de la pantalla.

  Debido a que la aplicación está diseñada para ser usada con el dispositivo en horizontal, restringiremos la orientación de rotación automática del dispositivo de tal forma que la aplicación sólo se pueda usar en posición de rotación horizontal hacia la derecha o hacia la izquierda.

Una vez hechos estos ajustes, debemos introducir en *Build Settings* cada una de las escenas de las que constará la aplicación por orden de aparición de las mismas.

Por último, seleccionamos la plataforma sobre la cual queremos ejecutar la aplicación, en este caso Android, y pulsamos sobre el botón “*Build*”, que será el responsable de la creación del empaquetado final de la aplicación.

5. Conclusiones

5.1. Conclusiones del proyecto

Tras la finalización de todas las fases en las que se ha ido desarrollando el proceso, podemos afirmar que se han conseguido los objetivos que se perseguían desde un principio, los cuales eran los siguientes:

- Usar la realidad aumentada para realizar una aplicación que pueda ayudar a los estudiantes a visualizar fenómenos físicos que no son perceptibles para nuestros sentidos.

- Aprovechar las facilidades interactivas de la realidad aumentada en el aprendizaje (“aprender haciendo”).

- Crear una aplicación con un ejercicio guiado en pequeños pasos para evitar dispersión de la atención, uno de los riesgos comprobados de la realidad aumentada.

Por todo ello, podemos decir que este proyecto:

- Muestra algunos principios de electromagnetismo, como la fuerza o el campo magnético, que en el mundo real no podemos observar.

- La aplicación es totalmente guiada.

- Se trata de una simulación interactiva (“aprender haciendo”).

5.2. Valoración personal

Bajo mi punto de vista, la realización de este proyecto ha sido entretenido, a la vez que laborioso y, en ocasiones, frustrante, ya que a veces las cosas no funcionaban como debían.

Al comienzo, no me sentía cómoda con la herramienta Unity, ya que nunca la había utilizado y no conocía su funcionamiento. Sin embargo, tras una labor de investigación de esta herramienta, y gracias a las sesiones de tutoría sobre los principios básicos de Unity y sobre la forma de realizar las diferentes interacciones con Vuforia que me proporcionó mi tutora, fui cogiendo soltura con Unity, hasta que me empecé a sentir más cómoda usándola e incluso me entretenía.

Además, al hecho de que Unity era totalmente nuevo para mí, hay que añadirle que tampoco había programado nunca en el lenguaje elegido, C#. Sin embargo, el hecho de tener experiencia previa en C y Java me hizo más llevadero aprender a programar con este lenguaje nuevo para mí.

A pesar de todo ello, ha sido muy satisfactorio haber podido realizar una aplicación para dispositivos Android, que es algo que siempre me había llamado la atención, y más si a ello le sumamos que dicha aplicación puede después ayudar a compañeros de otros Grados en sus estudios dentro de la universidad.

Además, el hecho de que la aplicación sea de realidad aumentada ha sido también muy satisfactorio ya que, bajo mi punto de vista, y como se explicó en apartados anteriores, es una de las tecnologías que está emergiendo actualmente en el mercado y que tendrá mucho uso en un futuro. Sin embargo, este mismo hecho también fue un impedimento en el desarrollo del proyecto ya que hay poca información actualizada disponible en la *web*. No obstante, hay muchos foros en internet donde la gente pregunta dudas relacionadas con Unity o Vuforia y que me han servido como ayuda a la hora de poder resolver dudas.

6. Presupuesto y planificación

6.1. Presupuesto

En esta sección, se va a realizar un análisis del presupuesto que ha conllevado la realización de este proyecto a lo largo de todo su desarrollo.

En este proceso de desarrollo del proyecto, podemos distinguir distintas fases, cada una con una finalidad concreta. Éstas son:

- **Fase 1**: en esta primera fase del proyecto, se llevó a cabo un análisis de las necesidades que se debían cubrir con la realización del mismo y de comprensión de la tecnología de realidad aumentada.

- **Fase 2**: a continuación, se inició un proceso de planificación de la estructura a seguir en el proyecto de acuerdo con los objetivos que se debían cumplir.

- **Fase 3**: para comenzar el proyecto, primero se llevó a cabo un proceso de aprendizaje de la herramienta Unity y de todas sus propiedades y funcionalidades, y también del lenguaje de programación C#.

- **Fase 4**: en esta fase, se procedió al diseño de las escenas, incluida la implementación de los scripts mediante lenguaje C# a través de MS Visual Studio.

- **Fase 5**: aquí, se realizaron las pruebas mediante la plataforma Unity a través de la cámara digital de ordenador portátil con el que se ha desarrollado el proyecto, a la vez que se llevaba a cabo la depuración del código mediante MS Visual Studio y la corrección de posibles errores que fueron surgiendo.

- **Fase 6**: en esta fase, se llevó a cabo el empaquetado de la aplicación una vez terminada, y la realización de pruebas sobre un dispositivo móvil con sistema operativo Android, corrigiendo así los diseños que se consideraron oportunos para la mejora del renderizado y diseño gráfico de los elementos virtuales presentes en la aplicación.

- **Fase 7**: por último, se procedió a la documentación y redacción de esta memoria.

Sin embargo, estas fases no se siguieron en un estricto orden cronológico, sino que, en ocasiones, se debió retroceder en el proceso para una mejor implementación de todas las tareas a realizar.

A continuación, se presenta una tabla con el número de días estimado que llevó la realización de cada una de las fases anteriores:

Fase del proyecto	Número estimado de días
Fase 1	3 días
Fase 2	3 días
Fase 3	10 días
Fase 4	90 días
Fase 5	30 días
Fase 6	10 días
Fase 7	17 días
	TOTAL: 163 días

Tabla 1. Duración en días de las fases del proyecto.

Suponiendo que la media de horas por día trabajado fue de 6 horas cada día, obtenemos que en total se han dedicado al proyecto **978 horas**.

Para obtener el presupuesto total del proyecto, se hará un cálculo de los recursos humanos y materiales que se han empleado a lo largo de todo el proceso, lo cual se mostrará con mayor detalle en las tablas incluidas a continuación.

Por una parte, se han calculado en la Tabla 2 los recursos materiales de todas aquellas herramientas tanto de *hardware* como de *software* que han sido necesarias para la correcta realización de las fases del proyecto.

Recursos Materiales			
	Precio (€)	**Amortización**	**Precio Final (€)**
Sistema operativo Windows 10	135	0,25	33,75
Ordenador portátil Asus F55L 1TB	655	0,25	163,75
Dispositivo móvil Samsung A5 2016	360	0,15	54
Unity 5.3.4	0		0
Blender	0		0
Microsoft Office 365	50	0,25	12,5
Vuforia 5.0.5	0		0
Adobe Acrobat Reader DC	180	0,5	90
Costes de consumo eléctrico	100		100
			TOTAL
			454

Tabla 2. Presupuesto de los recursos materiales.

Por otra parte, se ha recogido en la Tabla 3 el presupuesto de los recursos humanos empleados, es decir, del precio de las horas de trabajo que nos habría costado la realización del proyecto.

Recursos Humanos			
	Precio (€/día)	**Días**	**Precio Final (€)**
Analista	94	28	2.632
Programador	81	112	9.072
			TOTAL
			11.704

***Tabla 3*. Presupuesto de los recursos humanos.**

Por último, se ha hecho la suma en la Tabla 4 de los costes calculados en las tablas anteriores, con lo que obtenemos que el presupuesto total de proyecto ha sido de **12.158€**.

Recursos Totales	
	Precio(€)
Recusos Humanos Totales	11.704
Recusos Materiales Totales	454
	TOTAL
	12.158

Tabla 4. Presupuesto total del proyecto.

6.2. Planificación

En esta parte de la memoria, se incluye un diagrama Gantt con los detalles de la planificación de acuerdo a las fases que se indicaron en la Tabla 1 de este proyecto.

El diagrama de Gantt es una distribución conforme a un calendario, de tal manera que se pueda visualizar el periodo de duración de cada actividad de un proyecto, sus fechas de iniciación y terminación y el tiempo total requerido para la realización de cada una de las tareas de dicho proyecto. [26]

En la Tabla 5 que se muestra a continuación, se recoge la información detallada de las actividades que se llevaron a cabo en cada una de las diferentes fases del proyecto, indicando también el comienzo y finalización de dichas actividades.

Fase	Nombre de tarea	Duración	Comienzo	Fin
Fase 1	Análisi de las necesidades	1 día	vie 01/07/16	vie 01/07/16
Fase 1	Comprensión de Realidad Aumentada	2 días	lun 04/07/16	mar 05/07/16
Fase 2	Planificación de la estructura del proyecto	3 días	lun 11/07/16	mié 13/07/16
Fase 3	Aprendizaje de Unity	7 días	lun 18/07/16	lun 25/07/16
Fase 3	Aprendizaje de C#	5 días	mié 20/07/16	lun 25/07/16
Fase 4	Diseño de las escenas	25 días	lun 29/08/16	vie 25/11/16
Fase 4	Programación en C#	70 días	jue 01/09/16	mié 07/12/16
Fase 5	Pruebas en Unity	20 días	jue 01/09/16	mié 02/11/16
Fase 5	Depuración en MS Visual Studio	8 días	lun 03/10/16	mié 26/10/16
Fase 5	Corrección de errores	12 días	jue 13/10/16	mar 06/12/16
Fase 6	Pruebas en un Smartphone	5 días	lun 05/12/16	vie 09/12/16
Fase 6	Corrección del diseño	3 días	mar 06/12/16	jue 08/12/16
Fase 7	Documentación para la redacción de la memoria	5 días	lun 27/02/17	lun 06/03/17
Fase 7	Redacción de la memoria final	12 días	lun 27/02/17	mar 14/03/17

Tabla 5. Detalle de las tareas del diagrama de Gantt.

Dentro de esta planificación, debe recalcarse que la suma de los días indicados en las distintas actividades realizadas a lo largo proyecto no coincide con el número de días indicado en la Tabla 1. Esto es debido a que algunas de las actividades dentro de una misma fase se realizaron conjuntamente, por lo que, realmente, podrían considerarse una misma actividad, aunque en la gráfica se separan para tener una información más detallada.

Teniendo en cuenta las tareas y tiempo indicados en la Tabla 5, tenemos que en diagrama de Gantt resultante es el que se muestra en la Figura 34 que se incluye a continuación.

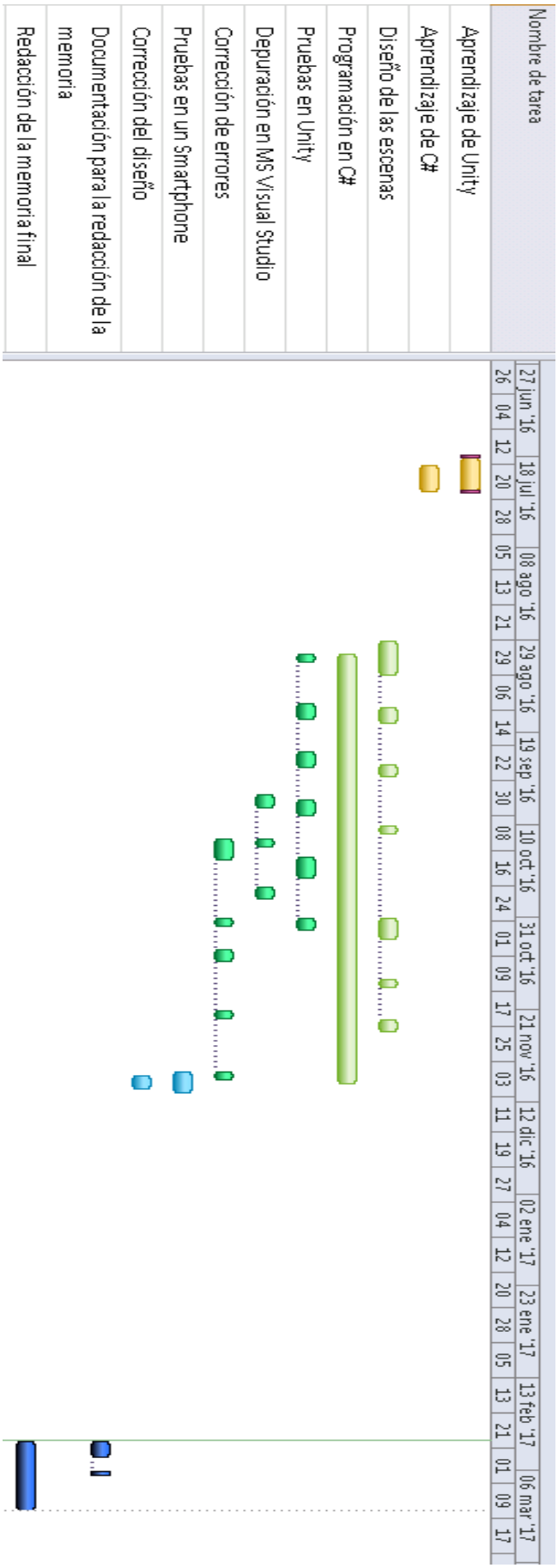


Figura 34. Diagrama de Gantt.

7. Líneas futuras de trabajo

La aplicación que hemos realizado, tiene diferentes diseños e implementaciones futuras que podrían realizarse a fin de mejorar algunos aspectos de la misma.

En primer lugar, se podría incorporar a la aplicación el acceso a una base de datos con la información correspondiente de los alumnos de la asignatura de Física del Grado en Ingeniería Informática, para los cuales va dirigida la aplicación. De esta forma, se podrá comprobar si los datos que introduce un determinado estudiante en la página de *Log In* coinciden con los datos pertenecientes a la base de datos con la información de los alumnos matriculados en la asignatura de Física de dicho Grado.

En segundo lugar, se podría extender el uso de la aplicación a otros sistemas operativos móviles para que, así, cualquier estudiante pueda utilizar la aplicación independientemente del tipo de dispositivo móvil que tenga a su disposición.

En tercer lugar, podría introducirse una nueva opción, en la misma página de *Log In* ya implementada, donde el estudiante pueda elegir el idioma en el que desea realizar el ejercicio de la aplicación, permitiendo así facilitar su uso a estudiantes internacionales.

En último lugar, se podría realizar una implementación del mismo ejercicio de tal forma que éste se pueda visualizar a través de gafas de realidad virtual, para poder así dar una mayor sensación a los estudiantes de inmersión con los elementos del entorno virtual.

8. Anexos

A continuación, se procederá a hacer un resumen en inglés, de carácter obligatorio para todos los estudiantes que pertenecemos al Plan de estudios 2011, con el contenido de todo lo anteriormente expuesto en esta misma memoria.

En él, procederé a explicar brevemente en lengua inglesa lo previamente analizado de los puntos que, bajo mi punto de vista, han sido los más relevantes para la correcta comprensión del proyecto realizado.

Para evitar repeticiones, las figuras incluidas en este resumen no se incorporarán a la lista de figuras mostrada al principio de este documento, ni se incluirán referencias bibliográficas.

Por último, se incluirá una lista con toda la bibliografía que ha sido necesaria en el proceso de investigación y documentación para la redacción de esta misma memoria.

8.1. Abstract

1. INTRODUCTION AND GOALS OF THE PROJECT

This Project looks for creating an application allowing the visualization of electromagnetic effects on a three-dimensional space, difficult to show on a bi-dimensional space, in order to facilitate its comprehension and learning, using for this purpose mobile devices such as smartphones or tablets on Android SO and the most recent advances on Augmented Reality technology.

Augmented Reality technology (AR) consists basically on the generation and management of visual elements combined or shown as virtual elements on a physical environment on real world, in such a way that the user can interact to get special sensitive experiences.

In this case, the use of Augmented Reality allows to show in three-dimensions, on real time and from any perspective, some specific information related to electromagnetic effects overlapping real objects. In this way, the student will be able to manipulate markers representing conductor wires, simulate the flow of a current on them and observe from the desired perspective the changes produced in the force or magnetic field vectors happening in a real system.

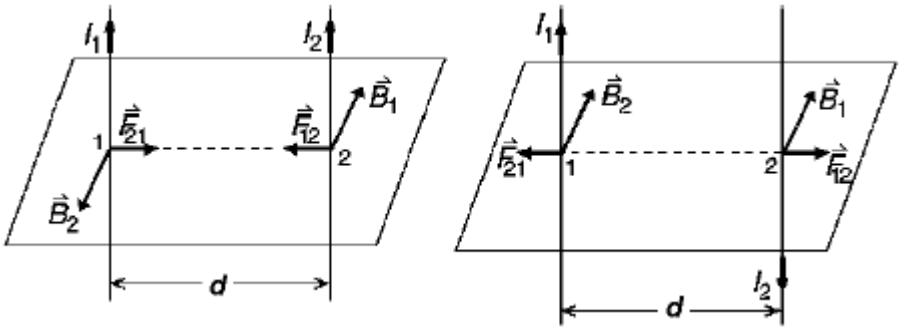
So the main goal of the project is to provide an application able to allow the visualization of electromagnetic concepts, to facilitate teaching these concepts. In order to achieve this aim, we should then include some complementary goals, associated to the knowledge and management of the AR concepts and the development tools needed to the main purpose, as:

- Understanding and management of AR and graphic 2D and 3D design concepts.
- C# language and Microsoft Visual Studio programming environment.
- Unity Games Engine.
- Vuforia Software Development Kit.
- 3D objects design with Blender (software program for images and graphics).

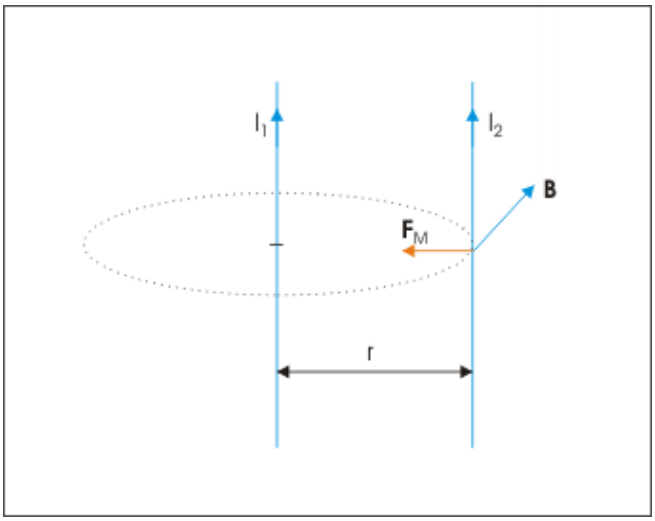
2. JOB PROPOSAL

The exercise to be reproduced in 3D images is the representation of the magnetic fields and their representative vectors, generated around two parallel straight wires carrying different intensity currents.

The tool should allow the student to introduce and modify the parameters associated to the direction and intensity of the currents and visualize the magnetic effect.



Magnetic and Force Vectors in the interaction of two parallel wires



Magnetic field generated by two parallel wires

All of these concepts will be presented to the student with an app on devices under Android OS. The student should solve the scenario as a lab exercise of Physics course. He will receive a formulation with the input interaction options the tool allows, in such a way that he may observe all the formerly explained as a simulation in Augmented Reality in a three-dimensional space for his better comprehension.

3. STATE OF ART

Using mobile devices matches very well with the Augmented Reality application looked for, since these devices:

- Use a tactile screen and a camera, that will allow to interact with Augmented Reality objects, and introduce inputs, and detecting elements as Image Targets or Frame Makers.

- Provide some storage capacity, also needed to hold the around 80 Mb required by the application.

- Support the installation of applications to be run.

3.1. Augmented Reality concepts

Our Augmented Reality (AR) application will basically need:

- A **device** with the software needed to run the AR application.
- A **tactile screen** for the user to interact by mean of Touch with the virtual environment provided by the app.
- A **digital camera**, which screen is going to be used to visualize the 3D AR objects.
- A **projector element**, such as any type of print paper marker, to be detected by the device and over which virtual environment objects will be projected or shown.

  These elements to represent AR may be of different types. Some of the more relevant are the following:

  - **QR codes or markers:** consisting on showing virtual elements over a QR code or any other kind of marker (a Frame Marker in our case).

AR show on QR code.

- **Real environment images or objects:** consisting on the recognition of such objects or elements (in our case, it will be the Stones Image Target), in such a way that the Augmented Reality environment can be represented over them.

- **GPS:** GPS may also be used to provide information in an Augmented Reality way about the location where we are, by mean of the GPS signal received by the device.

- **Augmented Reality glasses**: one of the last advances in AR, is to incorporate it on glasses in order to allow the vision of the projected images on them. Google Glass or 3D Glasses and VR Gear Glasses designed for smartphones are examples.

3.2. ANDROID O.S.

This project was conceived to be used under Android OS, as this Mobile Operative System is currently the more frequently used by mobile devices (80%). This OS was created by Google together with the *Open Handset Alliance*, and is based on Linux, on which kernel is supported.

One of the main characteristics on Android is its open character, as his rights and source code are of public domain. That allows the rapid growth of his use at global level.

The architecture of this O.S. is based on five layers, all of them free software. These five layers are: Linux kernel, native libraries, Android Runtime, Applications Framework, and the Applications itself.

The **Linux kernel** constitute the core of the operative system and is where main functions for any OS are running, such as security, protocol stack or the control of the different processes to run. This level is also hardware dependent, being the responsible to act as intermediate between such a hardware and the rest of the architecture layers.

Native Libraries is a set of libraries needed for the correct running of the OS as many of them include code referred to the graphic representation, data base processing, internet navigation, secure connection protocols as SSL or audio and video processing.

Android Runtime is the execution environment of each of the apps running over the OS. Runtime should compile the app code trough a virtual machine able to run on the device according to its memory limitations and processing speed.

A **Framework** is a set of defined rules standardizing the applications structure, in order to allow the maintenance and development of the apps and the code reuse.

In order to allow the development of this project for Android devices, the **SDK Android SDK** should be installed on the laptop where Unity is running. In this way, an .apk extension file may be built, allowing the students to be able to install the app on their devices.

3.3. SOFTWARE TOOLS USED IN THE PROJECT

The following software tools have been used in the development and implementation of this job:

I. UNITY

Unity was created by Unity Technologies in Denmark in 2005. It's a game development engine, multiplatform and fully integrated, that allows the creation of interactive environments both in 2D or 3D. It is mainly used for the development of videogames for Windows, Mac, Linux, iOS, Android, tvOS, Tizen, Xbox, Play Station and Samsung TV.

Unity, in version 5.3.4, is the main software tool used to design and implement the application. Thanks to it, we can create the virtual environment that will build the AR, and program such an environment to allow the user the interaction with the app. Unity provides a simple graphic interface that allows creating the AR environment in a simple way.

The project under Unity is divided into **Scenes** on top of which we add components, such as **3D objects**, with an assigned role by mean of the addition of **Components**. The main components in our project have been **Scripts**, that allow users the interaction with the virtual objects in the app.

We have used the User Interface (UI) tool in order to incorporate fixed elements that are not part of the AR but are fixed elements on the screen of the device. These elements facilitate the user interaction. For example, we can introduce UI of type **Text**, that shows a text on the screen, or elements of type **Input Field**, used for the user to introduce a text from the keyboard in a text frame.

When dealing with AR, Unity makes use of an **AR Camera** to recognize the objects of the real environment by mean of the device camera, and identify specific frames or models. It's also the element providing many aspects associated to the image processing like the rendering quality or the command of the maximum number of objects from the real environment to be recognized.

Many other Unity elements have been used, as described in the text.

II. VUFORIA

Vuforia is a SDK (Software Development Kit) created by the Qualcomm company in order to allow the recognition of images and the projection of AR on them. Vuforia 5.0.5, has been used in this project, imported on Unity to be able to use the Vuforia AR functionality.

Two Vuforia's elements are used: an Image Target and a Frame Marker, and the AR Camera:

- **Image Target** is an image that Vuforia allows to recognize through the camera of the mobile device, that will act as base on top of which to project or show the application scenes. In our case, we have used the Image Target Stones.

- A **Frame Marker** contains a determined frame or pattern, that will be recognized by Vuforia in order to allow moving specific objects into a scene. Each Frame Marker has its own associated ID. In our case we are using Frame Marker with ID 0.

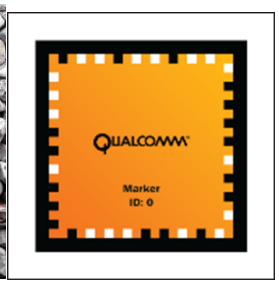


Image Target: Stones. ***Frame Marker ID 0.***

III. MICROSOFT VISUAL STUDIO: C#

MS Visual Studio is the software tool used to develop and debug scripts programming code. Among its functionality, it also allows debugging the code of Unity scenes, predicting in real time the name of functions and variables and detecting possible errors or warnings in the code.

Unity allows the work with C# or JavaScript or even other language implementations like Boo. C# is the one chosen for this project since most of the help manuals are C# oriented and performance seems to be quite similar in both cases.

IV. BLENDER

Blender is used in the project to facilitate the introduction of some figures in the Unity scenes.

8.2. Bibliografía

[1] La historia del ordenador y la evolución de la tecnología
http://www.ordenadores-y-portatiles.com/historia-del-ordenador.html

[2] Bashe, C. J., Johnson, L. R., Palmer, J. H., & Pugh, E. W. (1986). IBM's early computers. MIT press

[3] Los expertos prevén que la economía española marchará peor de lo esperado en 2017
http://www.expansion.com/economia/2016/08/14/57b04693268e3eb5338b4583.html

[4] Global Augmented Reality Market 2014-2018
http://www.technavio.com/report/global-augmented-reality-market-2014-2018

[5] Análisis sector realidad aumentada España
http://logic-fin.com/analisis-sector-realidad-aumentada/

[6] Esquema de dos hilos rectos largos que conducen corrientes paralelas
http://e-ducativa.catedu.es/44700165/aula/archivos/repositorio/3000/3232/html/4_fuerzas_entre_corrientes_paralelas_definicin_de_amperio.html

[7] Ditentria (Julio, 2016). Informe Mobile en España y en el Mundo 2016
http://www.ditrendia.es/wp-content/uploads/2016/07/Ditrendia-Informe-Mobile-en-España-y-en-el-Mundo-2016-1.pdf

[8] Azuma, R. T. (1997). A survey of augmented reality. Presence: Teleoperators and virtual environments, 6(4), 355-385

[9] Milgram, Paul; H. Takemura; A. Utsumi; F. Kishino (1994). "Augmented Reality: A class of displays on the reality-virtuality continuum"(pdf). Proceedings of Telemanipulator and Telepresence Technologies. pp. 2351–34

[10] Empleo de realidad aumentada proyectada sobre un código QR.
http://creeramonlaza.blogspot.com.es/2014/05/la-realidad-aumentada.html

[11] Fotografía de las Google Glass
http://www.euronics.es/blog/google-glasses-las-gafas-de-realidad-aumentada/

[12] Fotografía de las VR Gear
https://www.s7fanclub.com/2016/04/top-5-best-virtual-reality-vr-headsets-3d-galaxy-s7.html

[13] Szalavári, Z., Schmalstieg, D., Fuhrmann, A., & Gervautz, M. (1998). “Studierstube”: An environment for collaboration in augmented reality. Virtual Reality, 3(1), 37-48

[14] Chen, Y. C. (2006, June). A study of comparing the use of augmented reality and physical models in chemistry education. In Proceedings of the 2006 ACM international conference on Virtual reality continuum and its applications (pp. 369-372). ACM

[15] Ibáñez, M. B., Di-Serio, Á., Villarán-Molina, D., & Delgado-Kloos, C. (2015). Augmented reality-based simulators as discovery learning tools: An empirical study. IEEE Transactions on Education, 58(3), 208-213

[16] Ibáñez, M. B., Villarán, D., & Delgado-Kloos, C. (2015, July). Integrating Assessment into Augmented Reality-Based Learning Environments. In Advanced Learning Technologies (ICALT), 2015 IEEE 15th International Conference on (pp. 218-222). IEEE

[17] Ibáñez, M. B., Villarán, D., & Delgado-Kloos, C. (2015, July). Integrating Assessment into Augmented Reality-Based Learning Environments. In Advanced Learning Technologies (ICALT), 2015 IEEE 15th International Conference on (pp. 218-222). IEEE

[18] Escenario de trabajo con Vuforia
http://realidadaumentada.info/tecnologia/

[19] Image Target: Stones proporcionado por Vuforia
https://developer.vuforia.com/sites/default/files/sample-apps/targets/stones.pdf

[20] Frame Marker con ID 0 proporcionado por Vuforia
https://developer.vuforia.com/resources/dev-guide/frame-markers

[21] Silberschatz, A., Galvin, P. B., Gagne, G., & Silberschatz, A. (1998). Operating system concepts (Vol. 4). Reading: Addison-wesley

[22] Arroyo, N. (2013). Información en el móvil. Editorial UOC

[23] Gráfica del uso de los diferentes sistemas operativos móviles
http://www.expansion.com/economia-digital/companias/2015/12/09/56684be1ca474151018b4590.html

[24] Arroyo, N. (2013). Información en el móvil. Editorial UOC

[25] Arquitectura de Android
http://www.androidcurso.com/index.php/99

[26] Hinojosa, M. A. (2007). Diagrama de Gantt