ESCUELA SUPERIOR DE INGENIERÍA

GRADO EN INGENIERÍA INFORMÁTICA

SISTEMA DE REALIDAD AUMENTADA ESPACIAL CON FUNCIONES ENFOCADAS A LA REHABILITACIÓN DE PACIENTES DE ALZHEIMER

José María Correro Barquín

27 de noviembre de 2020

ESCUELA SUPERIOR DE INGENIERÍA

GRADO EN INGENIERÍA EN INFORMÁTICA

SISTEMA DE REALIDAD AUMENTADA ESPACIAL CON FUNCIONES ENFOCADAS A LA REHABILITACIÓN DE PACIENTES DE ALZHEIMER

Director: José Miguel Mota Macías
Autor: José María Correro Barquín

Cádiz, 27 de noviembre de 2020

A mis padres, a los que les debo todo.

Os quiero.

José

Resumen

La **realidad aumentada** es la integración de información virtual en el mundo real. Para que se considere realidad aumentada, la información ha de tener un vínculo fuerte con el entorno real, el cual es más bien una relación espacial entre lo aumentado y la realidad. Entre las posibilidades de la realidad aumentada se encuentran nuevos métodos de visualización que hacen uso de elementos ópticos como reflejos en espejos, pantallas transparentes, hologramas o proyectores. A estas variaciones se las conoce como **realidad aumentada espacial**.

En 2009, Pranav Mistry mostraba al mundo la tecnología **Sixth Sense** que él mismo había desarrollado. Esta tecnología, ejemplo de realidad aumentada espacial, permite mostrar en el mundo real información procesada en un sistema informático proyectando imágenes y que el usuario interaccione con ella sin necesidad de utilizar interfaces típicas como pantallas o teclados. En lugar de ello, utiliza sus propias manos con gestos y señalizaciones sobre las imágenes proyectadas, siendo el sistema capaz de reconocer estos gestos y señalizaciones capturándolos con una cámara.

En este proyecto se trata de replicar esta tecnología utilizando distintas herramientas para desarrollar los componentes necesarios e integrarlos en un único sistema que se ejecute en un dispositivo móvil. A partir de él se crearán distintas aplicaciones enfocadas a ser utilizadas en la rehabilitación de pacientes de enfermedades neurológicas.

Palabras clave

SixthSense Project, Realidad Aumentada Espacial, Proyector, OpenCV, Unity, Alzheimer

Enlace a GitHub

https://github.com/JoseCorrero/SixthSenseTFG

Índice general

I Prolegómeno **1**

1. Introducción **3**
1.1. Motivación . . . 3
1.2. Objetivos . . . 4
1.3. Alcance . . . 5
1.4. Organización del documento . . . 6

2. Estado del arte **9**
2.1. Realidad aumentada espacial . . . 9
2.2. Tecnología SixthSense . . . 10
2.2.1. Funcionamiento . . . 11
2.2.2. Aplicaciones . . . 12
2.3. VirtualRehab . . . 16

3. Planificación **19**
3.1. Metodología de desarrollo . . . 19
3.2. Diagrama de Gantt . . . 20
3.3. Costes . . . 20
3.4. Gestión de riesgos . . . 21

II Desarrollo **23**

4. Análisis de requisitos del sistema **25**
4.1. Necesidades de negocio . . . 25
4.2. Catálogo de actores . . . 25
4.3. Requisitos funcionales . . . 26
4.4. Modelo conceptual . . . 30
4.5. Requisitos no funcionales . . . 35

5. Diseño del sistema **37**
5.1. Arquitectura del Sistema . . . 37
5.1.1. Arquitectura Física . . . 37
5.1.2. Arquitectura Lógica . . . 38
5.2. Diseño de la interfaz de usuario . . . 38

6. Construcción del sistema **43**
6.1. Tecnologías . . . 43
6.2. Algoritmo de detección de color . . . 44
6.3. Implementación en Unity . . . 48

7. Pruebas del sistema **53**
7.1. Pruebas unitarias . . . 53
7.2. Pruebas de integración . . . 54
7.3. Pruebas de sistema . . . 55

III Epílogo **57**

8. Conclusiones **59**
8.1. Objetivos alcanzados . . . 59
8.2. Conocimientos y valores adquiridos . . . 59
8.3. Trabajo futuro . . . 60

Apéndices **63**

A. Modelado e impresión 3D **65**
A.1. OpenSCAD . . . 65
A.2. Primer modelo . . . 67
A.3. Segundo modelo . . . 75
A.4. Impresión . . . 84

B. Manual de instalación **85**
B.1. Inventario de componentes . . . 85
B.2. Requisitos previos . . . 85
B.3. Procedimientos de instalación . . . 85

C. Manual de usuario **87**
C.1. Colocación y conexión del hardware . . . 87
C.2. Uso del sistema . . . 88

D. Manual de desarrollador **93**
D.1. Añadir colores al detector . . . 93
D.2. Interaccionar con objetos . . . 95

Índice de tablas

3.1. Costes 21
3.2. Riesgo 1. Error en la detección de los colores. 21
3.3. Riesgo 2. Falta de potencia computacional. 21
3.4. Riesgo 3. Error en la calibración de las posiciones de los colores. 22

6.1. Script ColorDetection: proceso de sincronización con barreras. 48

A.1. Errores encontrados en el primer modelo diseñado y sus soluciones. 72

Índice de figuras

2.1. Pranav Mistry utilizando WuW. 11
2.2. Utilizando SixthSense para dibujar. 12
2.3. Utilizando SixthSense para consultar un mapa. 12
2.4. Utilizando SixthSense para tomar una foto. 13
2.5. Utilizando SixthSense para realizar una llamada telefónica. 13
2.6. Utilizando SixthSense para obtener información de un vuelo. 14
2.7. Utilizando SixthSense para obtener información de una persona. 14
2.8. Utilizando SixthSense para ver la hora. 15
2.9. Juegos de VirtualRehab: dianas. 16
2.10. Juegos de VirtualRehab: utensilios de cocina. 17
2.11. Juegos de VirtualRehab: mantener a flote. 17

3.1. Diagrama Gantt de la planificación temporal. 20

4.1. Diagrama de casos de uso. 26
4.2. Modelo conceptual del sistema. 30
4.3. Escenas de Unity. 30
4.4. Modelo de color HSV. 31
4.5. Clase ColorRange. 31
4.6. Clase ColorDetector. 32
4.7. Clase ColorDetectorCV. 32
4.8. Clase Calibrator. 33
4.9. Clase ColorPosition. 33
4.10. Clase ColorDetection e interfaz ICamera. 34

5.1. Diseño de componentes. 38
5.2. Interfaz de usuario: escena de calibración. 39
5.3. Interfaz de usuario: menú principal. 39
5.4. Interfaz de usuario: juego de dianas. 40
5.5. Interfaz de usuario: tiempo empleado en prueba. 40
5.6. Interfaz de usuario: juego de utensilios de cocina. 41
5.7. Interfaz de usuario: juego de vestir al muñeco. 42

6.1. Detección de color: conversión a formato HSV. 44
6.2. Detección de color: binarización de imagen. 45
6.3. Detección de color: mejora de la máscara. 46
6.4. Detección de color: contornos de los colores. 46

6.5. Detección de color: centro del contorno. 47

7.1. Pruebas de integración: detección de color y Unity en Android. 55

A.1. Ejemplo de modelado 3D con OpenSCAD. 66
A.2. Ejemplo de modelado 3D con OpenSCAD (2). 66
A.3. Soporte físico para dispositivos hardware (parte frontal). 67
A.4. Soporte físico para dispositivos hardware (parte trasera). 68
A.5. Modelado 3D: acceso a conexiones cableadas. 69
A.6. Modelado 3D: modelo 1. 70
A.7. Modelado 3D (modelo 1): hendiduras para parte trasera y encajes de parte trasera. 70
A.8. Modelado 3D (modelo 1): lengüeta y soporte de espejo. 71
A.9. Modelado 3D (modelo 1): hendidura para lengüeta. 71
A.10. Modelado 3D: modelo 2. 76
A.11. Modelado 3D (modelo 2): hendiduras para parte trasera y encajes de parte trasera. 77
A.12. Modelado 3D (modelo 2): hendidura para soporte de espejo y soporte de espejo. 77
A.13. Modelado 3D (modelo 2): huecos en la parte trasera y topes. 78
A.14. Modelado 3D (modelo 2): perfilado del soporte. 78
A.15. Ejemplo de uso de Ultimaker Cura. 84

C.1. Manual de usuario: uso de soporte para dispositivos hardware. 87
C.2. Manual de usuario: escena de calibración. 88
C.3. Manual de usuario: menú principal. 89
C.4. Manual de usuario: tutorial de juego. 90
C.5. Manual de usuario: tiempo empleado en terminar juego. 90
C.6. Manual de usuario: prueba de dianas. 91
C.7. Manual de usuario: prueba de utensilios de cocina. 91
C.8. Manual de usuario: prueba de vestir al muñeco. 92

D.1. Manual de desarrollador: GameObjets de la detección de colores. 93
D.2. Manual de desarrollador: ventana Inspector de un color (1). 94
D.3. Manual de desarrollador: ventana Inspector de un color (2). 94
D.4. Manual de desarrollador: ventana Inspector del detector de color. 95
D.5. Manual de desarrollador: ventana Inspector de objeto que interacciona. . . 95

Parte I

Prolegómeno

Capítulo 1

Introducción

En este capítulo se describe la motivación, los objetivos y el alcance del proyecto, así como la organización que seguirá este documento.

1.1. Motivación

Este proyecto nace con la idea de recrear la tecnología desarrollada en el proyecto ***SixthSense*** (Pranav Mistry, 2009) [21]. Conocer sus nociones básicas es necesario para entender la motivación de este proyecto.

Su objetivo es llevar al mundo real información del mundo digital y permitir al usuario interaccionar con ella con sus propias manos a través de gestos y señalizaciones, eliminando la necesidad de utilizar pantallas y controladores. La tecnología utiliza como componentes hardware una cámara para captar el entorno, un dispositivo computacional móvil, como un ordenador portátil o un smartphone, que recibe la imagen que le transmite la cámara, procesa la información que pueda haber en ella y genera nueva información, y un proyector que muestra dicha información al entorno.

Son muchas las utilidades para las que se ha desarrollado esta tecnología. Algunos ejemplos son el poder ver un mapa e interactuar con él, tomar fotos o realizar una llamada telefónica, entre otros.

Con este proyecto buscamos recrear la tecnología *SixthSense* utilizando herramientas distintas, como la librería de visión artificial ***OpenCV*** [9] y el entorno de desarrollo de aplicaciones multiplataforma ***Unity*** [11].

Uno de los posibles usos que se le puede dar a la tecnología es el de ayudar a las personas que sufren enfermedades neurológicas, como Alzheimer[1] o Parkinson[2] a realizar ejercicios

[1] El Alzheimer es la forma más común de demencia, un término general que se aplica a la pérdida de memoria y otras habilidades cognitivas que interfieren con la vida cotidiana. La enfermedad de Alzheimer es responsable de entre un 60 y un 80 por ciento de los casos de demencia. El Alzheimer no es una característica normal del envejecimiento. El factor de riesgo conocido más importante es el aumento de la edad, y la mayoría de las personas con Alzheimer son mayores de 65 años [15].

[2] La enfermedad de Parkinson es una enfermedad progresiva del sistema nervioso que afecta el movimiento. Los

y actividades de rehabilitación que pueden mejorar su estado. Algunos de estos ejercicios consisten en señalar dianas de colores o realizar ejercicios de memoria. Estos ejercicios se pueden realizar utilizando la tecnología *SixthSense*, eliminando la necesidad de disponer de los elementos físicos necesarios para ello.

La realidad aumentada ya ha sido utilizada en actividades de rehabilitación previamente.

> La rehabilitación de realidad aumentada es un complemento interesante y útil de la terapia tradicional al aprovechar los beneficios del entrenamiento del mundo real y virtual, como proporcionar una experiencia inmersiva para los usuarios, cuantificación objetiva del proceso de entrenamiento y una forma motivadora de utilizar la práctica masiva. Al desarrollar un sistema de rehabilitación de realidad aumentada, la coordinación en el mundo real del usuario se unifica con el mundo virtual. Esto permite que los pacientes se sientan como si los objetos virtuales de asistencia, que se muestran para ayudarlos a realizar sus ejercicios, estuvieran realmente presentes y pertenecieran al mundo real en lugar de estar separados en una pantalla separada [22].

Por ello, las pruebas o juegos que se implementen en el sistema para probar su correcto funcionamiento se diseñarán de forma que puedan simular los ejercicios y actividades anteriormente mencionados.

1.2. Objetivos

El fin de este proyecto es estudiar el proyecto *SixthSense* [21] y crear un sistema de realidad aumentada espacial similar utilizando distintas tecnologías en el desarrollo y creando nuevas aplicaciones de uso sanitario en el ámbito de la rehabilitación de enfermedades neurológicas.

Para ello, podemos identificar los siguientes objetivos:

- **Obtención de librería de reconocimiento de manos.** El usuario interactuará con el sistema mediantes gestos o señalando con sus propias manos. Por ello, es necesario disponer de una librería que permita al sistema reconocer o detectar la posición de las manos del usuario.
- **Integración de los distintos componentes.** El sistema está compuesto por varios componentes que han de trabajar conjuntamente. Estos componentes son la librería de reconocimiento de manos, la cámara, el proyector y la aplicación desarrollada en Unity. Integrar todos estos componentes para
- **Estudio de la disposición óptima de los componentes hardware.** El sistema utiliza una cámara para captar el entorno, un dispositivo computacional que recibe la imagen de la cámara y procesa información y un proyector para mostrar la información procesada. Estos dispositivos son portados por el usuario durante la

síntomas comienzan gradualmente. A veces, comienza con un temblor apenas perceptible en una sola mano. Los temblores son habituales, aunque la enfermedad también suele causar rigidez o disminución del movimiento [4].

ejecución, y deben estar colocados de forma que puedan ejercer su función de la manera más eficiente posible. Son varias las posiciones y la orientaciones en las que pueden situarse, así como los métodos de conexión (por cable o inalámbrica) entre los dispositivos, por lo que hay que determinar cuáles se consideran mejores.

- **Obtención de soportes para los componentes hardware.** Como se menciona en el punto previo, los dispositivos que utiliza el sistema deben ser portados por el usuario, por lo que se necesitan soportes que le permitan hacerlo.

- **Estudio de las aplicaciones a desarrollar.** Se han de estudiar las posibilidades de la tecnología y definir los usos y aplicaciones que se desarrollarán. La rehabilitación de los pacientes de enfermedades neurológicas cuenta con distintos procesos entre los que se encuentra la realización de actividades de coordinación y movimiento. Es aquí donde se centra este proyecto: estudiar dichas actividades y desarrollar un sistema que ayude a llevarlas a cabo, así como proponer nuevas formas de obtener los logros que se persiguen con ellas. Es por ello que se han de analizar dichas actividades y determinar cuáles se podrán implementar con el sistema.

- **Desarrollo de funcionalidades.** A partir de los datos recabados en el estudio que se explica en el punto anterior, se diseñarán e implementarán distintas funcionalidades.

1.3. Alcance

El producto final de este proyecto será una aplicación para smartphones *Android* que permita al usuario interaccionar con ella señalando los elementos de la aplicación, la cual es proyectada en una superficie, preferiblemente una pared lisa de color claro, aunque pueden desarrollarse aplicaciones que destinadas a proyectarse en otras superficies.

- **Diseño e implementación de librería de reconocimiento de manos mediante *OpenCV*.** La posición y los gestos de las manos del usuario se reconocerán mediante la detección de cintas de colores en los dedos. Para ello, se diseñará e implementará una librería de reconocimiento de colores utilizando *OpenCV*. Esta librería ofrecerá una serie de funciones que permitirán seleccionar los colores a detectar y obtener la posición más aproximada posible de ellos mediante el uso de un algoritmo de tratamiento de imágenes. Las librerías de *OpenCV* permiten su uso tanto en sistemas operativos de escritorio como sistemas operativos móviles, con algunas diferencias entre ellos. Por ello, se desarrollará una versión de la librería para escritorio y otra para *Android*.

- **Modelado e impresión 3D de soportes necesarios.** Se hará uso de la impresora 3D de la que dispone el departamento de Ingeniería Informática para construir los soportes necesarios para permitir al usuario portar los dispositivos. Estos soportes serán modelados mediante la herramienta de modelado 3D *OpenSCAD* [10], la cuál ofrece un compilador para modelar los objetos 3D mediante código, lo que ofrece una gran precisión. Para desarrollar dicho código, se estudiarán y aprenderán las nociones básicas que sean necesarias de la documentación oficial de *OpenSCAD* y de

una wiki realizada por el doctor en robótica Juan González Gómez para un seminario en la Universidad Carlos III de Madrid titulado *Diseño de piezas con OpenSCAD* [18].

- **Selección de actividades a implementar.** Disponemos de una serie de actividades realizadas en la rehabilitación. Se analizarán y se escogerán aquellas que puedan ser implementadas eficazmente debido a la naturaleza del sistema. Éstas serán aquellas que sólo requieran de la cognición o del movimiento de las manos del paciente.
- **Diseño e implementación de las actividades mediante *Unity*.** El sistema será desarrollado bajo la herramienta de desarrollo de aplicaciones multiplataforma en tiempo real *Unity*. Será empleada como motor computacional y de implementación de juegos o pruebas. En ella se unirán las interfaces de las diferentes tecnologías y componentes de los que consta el sistema (librerías de reconocimiento de colores con *OpenCV*, captura de imagen de la cámara y proyección).

1.4. Organización del documento

El presente documento está compuesto por tres partes principales y una serie de apéndices. Cada una de las partes consta de una serie de capítulos, los cuales, a su vez, se componen de varias secciones, al igual que los apéndices.

A continuación, se procede a describir brevemente el contenido de cada uno de los capítulos y apéndices.

Parte I - Prolegómeno

En el **capítulo 1** se explica la motivación que da lugar a este proyecto, los objetivos que se pretenden alcanzar con su realización (el qué) y el alcance que tendrá (el cómo), así como la organización que seguirá este documento.

En el **capítulo 2** se estudia la tecnología *SixthSense* y sus aplicaciones, los proyectos precedentes en los que se origina y su desarrollo y los conceptos e ideas que los fundamentaron.

En el **capítulo 3** se describen los aspectos relativos a la gestión del proyecto: metodología, planificación, costes y riesgos.

Parte II - Desarrollo

En el **capítulo 4** se presentan las necesidades de negocio que le dan sentido al proyecto, el catálogo de requisitos del sistema y su modelo conceptual.

En el **capítulo 5** se recoge la arquitectura física y lógica del sistema de información y el diseño de la interfaz de usuario.

En el **capítulo 6** se tratan los aspectos relacionados con la implementación del sistema en código y los distintos entornos de desarrollo utilizados.

En el **capítulo 7** se presentan los diferentes tipos de pruebas que se han llevado a cabo.

Parte III - Epílogo

En el **capítulo 8** se detallan las lecciones aprendidas tras el desarrollo del presente proyecto y se identifican las posibles oportunidades de mejora sobre el software desarrollado.

Apéndices

En el **apéndice 1** se detallan las características de los soportes del hardware y los procesos de modelado e impresión 3D.

En el **apéndice 2** se indican los requisitos previos y el proceso de instalación del software.

En el **apéndice 3** se detallan instrucciones de conexión y uso del sistema.

En el **apéndice 4** se muestra cómo modificar la aplicación desde la plataforma de desarrollo *Unity*.

Capítulo 2

Estado del arte

El proyecto se basa en la tecnología *SixthSense*, la cual se puede considerar un tipo de realidad aumentada espacial. A continuación, se detalla este concepto y se explica en qué consiste dicha tecnología. Además, se muestran las aplicaciones creadas en la plataforma *VirtualRehab* para la rehabilitación de pacientes de Alzheimer o Parkinson.

2.1. Realidad aumentada espacial

Se puede definir la realidad aumentada (**RA**) como un caso de realidad virtual (**RV**) en el que el entorno juega un papel fundamental de la experiencia. El usuario no se sumerge en un mundo artificial, sino que es la parte artificial la que se introduce en el mundo real. Esta diferencia, a simple vista, puede hacer parecer que la RA es más simple que la RV, puesto que ya hay un entorno que no ha de ser creado. Pero no es así, ya que controlar dicho entorno real es mucho más difícil que controlar uno artificial.

A partir de lo descrito previamente, la RA puede definirse como la integración de información en un entorno real. La información debe estar estrechamente enlazada con el entorno en el que se integra. El enlace es prácticamente una relación espacial entre lo creado y lo real.

Ivan Sutherland estableció los fundamentos teóricos de la realidad virtual en 1965, describiendo lo que, en su opinión, debería ser un dispositivo de realidad aumentada:
"El dispositivo definitivo sería una habitación en la cual un ordenador puede controlar la existencia de materia. Una silla desplegada en dicha habitación sería lo suficientemente buena para sentarse. Unas esposas desplegadas en dicha habitación serían capaces apresarte, y una bala sería mortal. Con la programación adecuada, dicho dispositivo podría ser, literalmente, el País de la Maravillas en el que Alicia estuvo [23]."

Sutherland enfatizó en que el usuario debía ser capaz de interactuar con el entorno virtual. A partir de ello, creó el primer **Casco de Realidad Virtual** (Head-Mounted Display, o HDM), el cual supuso el nacimiento de la RA. Con un conjunto de espejos, hacía posible que el usuario viese los elementos del mundo virtual y los del mundo real a la vez. Además, usó sensores mecánicos y de ultrasonidos posicionados en la cabeza del usuario para poder

situarla en el espacio.

A día de hoy, el principal reto sigue siendo la consistencia espacial en la integración entre los elementos artificiales y el entorno real. Para ello, el sistema debe determinar la posición del usuario continuamente si éste está en movimiento.

Otros nuevos paradigmas de visualización combinan reflejos, pantallas transparentes, hologramas o proyectores para establecer la integración entre elementos. Esta variante tecnológica se conoce como realidad aumentada espacial (RAE). La RAE soluciona algunas de las limitaciones de la RA tradicional. La RAE se ha convertido en una tecnología de interés en la que investigar e invertir esfuerzos [17].

2.2. Tecnología SixthSense

Kedar Kanel realizó en mayo de 2014 una tesis, en el grado de Tecnología de la Información, en la Universidad de Ciencias Aplicadas de Centria, en Kokkola, Finlandia, titulada ***Sixth Sense Technology*** [20]. En ella, estudia el desarrollo de la tecnología SixthSense y los conceptos que la rodean.

A continuación, se resumen las ideas contenidas en esta tesis.

El concepto de sexto sentido (o percepción extrasensorial) implica poder percibir información sin utilizar los cinco sentidos primarios: vista, oído, tacto, gusto y olfato. Esto no parece posible para el ser humano a nivel biológico, pero sí tecnológicamente. La idea de un dispositivo que pueda percibir información de su entorno, procesarla y transmitirla es lo que da lugar a la tecnología *SixthSense*, y modificar el entorno de una persona para permitirle estar conectado con el mundo digital sin dejar de estarlo con el real es el objetivo principal.

La primera vez que se habló de la tecnología *SixthSense* fue en 1994, cuando el profesor Steve Mann desarrolló el primer prototipo de un dispositivo portátil capaz de permitir a dos usuarios situados en diferentes partes del mundo comunicarse sin necesidad de usar palabras. Uno de ellos portaba una cámara que transmitía la imagen de lo que tenía delante al otro. Este último señalaba con un puntero láser un punto en la imagen y el primero veía el lugar exacto al que apuntaba. Mann llamó a este prototipo *Telepointer*.

Tras esto, en 2009, Pranav Mistry presentó lo que en un principió se llamó *Wear ur World (WuW)*, y que hoy conocemos como tecnología *SixthSense*, la cual es un ejemplo de RAE. Mistry desarrolló un prototipo compuesto por una dispositivo computacional con acceso a internet, una cámara, un proyector, marcadores de colores, un micrófono y un espejo (ver figura 2.1).

Figura 2.1: Pranav Mistry utilizando WuW.

2.2.1. Funcionamiento

El prototipo funciona de la siguiente forma: el usuario pone en sus dedos unos marcadores de colores, los cuales sirven para representar la posición de los dedos. Los colores utilizados pueden ser los que quiera el usuario, siempre y cuando no se repitan, ya que cada color representa un dedo. Los marcadores pueden ser de papel, de cinta, dedales, etcétera. La cámara graba el entorno y las manos del usuario y detecta los marcadores. Un sensor utiliza dichos marcadores para reconocer los gestos que el usuario realiza. El dispositivo computacional contiene todo el software necesario. Recibe las imágenes de la cámara y utiliza internet para generar una respuesta a esas imágenes. La información generada es mostrada a través del proyector, el cual cuelga hacia abajo y proyecta hacia el espejo, reflejando la proyección hacia delante. La proyección necesita una superficie en la que ser mostrada. Esta puede ser una pared o incluso la propia mano del usuario. El usuario puede interactuar con la imagen proyectada a través de nuevos gestos.

El software utilizado es open source y está escrito en C, C++, C# y Java. Utiliza algoritmos de visión artificial (computer vision) para reconocer los elementos del entorno y generar respuestas. Está desarrollado para *Windows* y hace uso de *Microsoft DirectX*, *Adobe Flash Player* y *Microsoft Outlook*.

2.2.2. Aplicaciones

Varias aplicaciones han sido desarrolladas para su uso y aplicación en tareas cotidianas:

Dibujar. La aplicación permite dibujar con los dedos al usuario, detectando el sistema los marcadores de colores y mostrando el dibujo con la proyección. Una vez finalizado, el dibujo se guarda en el dispositivo (ver figura 2.2).

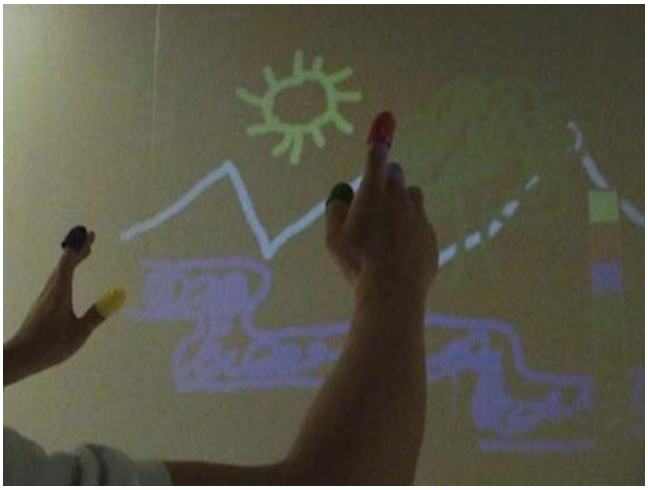


Figura 2.2: Utilizando SixthSense para dibujar.

Consultar un mapa. El usuario puede acceder a un mapa realizando un gesto. Una vez desplegado el mapa, puede interaccionar con él mediante gestos y señalizaciones (ver figura 2.3).

Figura 2.3: Utilizando SixthSense para consultar un mapa.

Tomar una foto. Con un simple gesto del usuario, el sistema guarda la imagen grabada por la cámara en ese instante y la guarda en el dispositivo, habiendo realizado así una fotografía. Ésta se puede ver o compartir por internet (ver figura 2.4).

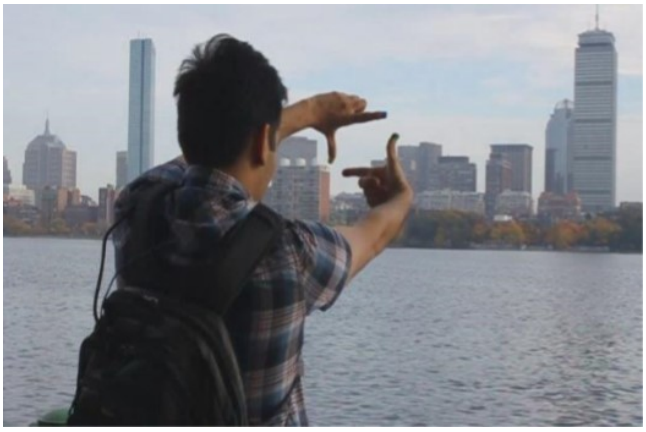


Figura 2.4: Utilizando SixthSense para tomar una foto.

Realizar una llamada telefónica. Un teclado numérico es proyectado en una superficie (puede ser la propia mano del usuario). Se marcan los números señalándolos con los dedos que tienen marcadores de colores. El sistema reconoce los marcadores y su posición, detectando los números que señala para realizar la llamada al número telefónico marcado (ver figura 2.5).

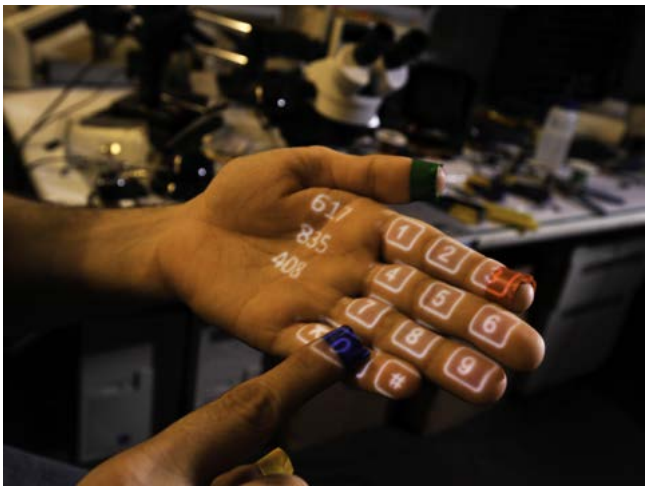


Figura 2.5: Utilizando SixthSense para realizar una llamada telefónica.

Mostrar información. Quizás, la más interesante de las aplicaciones desarrolladas. El sistema reconoce objetos e incluso personas del entorno y muestra información sobre ello. La información puede ser obtenida mediante internet (ver figuras 2.6 y 2.7).

Figura 2.6: Utilizando SixthSense para obtener información de un vuelo.

Figura 2.7: Utilizando SixthSense para obtener información de una persona.

Ver la hora. Si el usuario no dispone de un reloj, puede realizar un gesto en su muñeca para que el sistema le muestre uno funcional que muestra la hora actual (ver figura 2.8).

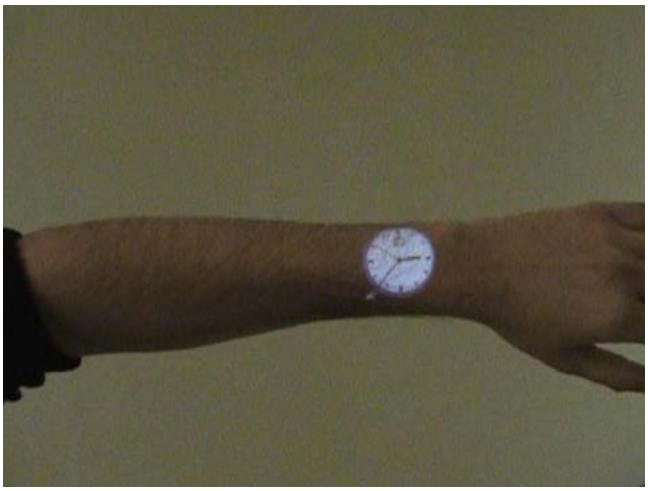


Figura 2.8: Utilizando SixthSense para ver la hora.

Además de esto, esta tecnología ofrece posibilidades en otros entornos más complejos:

Industria. En algunas industrias, como la metalurgia, se utilizan máquinas cuyo manejo puede suponer un peligro para los trabajadores inexpertos. Diversos problemas pueden surgir durante el propio proceso de aprendizaje de estas máquinas que supongan enormes riesgos para sus usuarios o quienes le rodeen [6]. Con *SixthSense* se puede apoyar a los usuarios con el despliegue de información sobre la máquina, sus controles y la situación en la que se encuentre en tiempo real.

Pero no sólo puede ser útil para prevenir riesgos. También puede serlo para mejorar la productividad y eficiencia de los procesos industriales. Esto se conoce como **Industria 4.0** [7].

La **Industria 4.0** es un avance en el que se integran las distintas tecnologías presentes en los procesos y técnicas industriales para que interactúen entre sí. De esta forma se pueden recopilar y analizar datos durante la producción, agilizar los procesos y aumentar la eficiencia para producir mayor valor con menos. La Industria 4.0 se basa en nueve pilares tecnológicos fundamentales, entre los que se encuentra la realidad aumentada. Los trabajadores pueden recibir instrucciones sobre cómo realizar sus tareas mediante la visualización de información en el entorno.

Ayudar a personas discapacitadas. Por ejemplo, puede implementarse un traductor de lenguaje de signos. El sistema reconocería los gestos y transmitiría la información, ya sea a través de texto con la proyección o de sonido con unos altavoces que se podrían agregar al conjunto de dispositivos.

2.3. VirtualRehab

VirtualRehab [14] es un producto de rehabilitación de pacientes que presentan algún tipo de discapacidad física debido a enfermedades neurológicas que emplea entornos de realidad virtual. Consiste en un software que utiliza *Microsoft*® *Kinect* o *Leap Motion* para capturar el movimiento del paciente. El sistema contiene pruebas o juegos que, al realizarlos, ayudan en la rehabilitación de las extremidades superiores e inferiores o de las manos de pacientes con enfermedades neurológicas como el Alzheimer o el Parkinson.

VirtualRehab captura el movimiento del usuario y lo representa en el entorno virtual. Contiene una serie de juegos que sirven como entrenamiento del sistema motor y de la cognición del paciente. A continuación, se muestran algunos ejemplos:

Dianas

En este juego aparecerán consecutivamente unas dianas que el paciente deberá alcanzar con las manos o los pies, dependiendo de la altura a la que se encuentren. Las dianas azules aparecerán a la izquierda y las rojas a la derecha. También aparecerán cuadrados en el suelo que el paciente deberá pisar (ver figura 2.9).

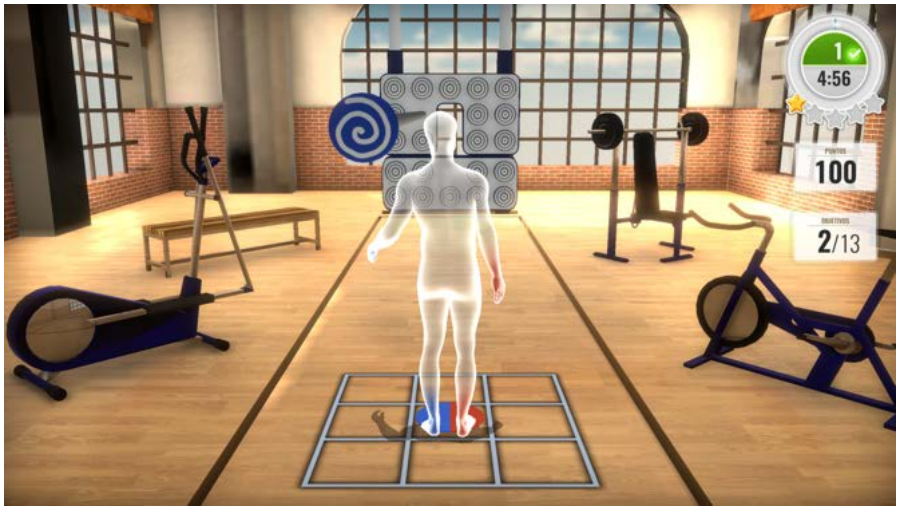


Figura 2.9: Juegos de VirtualRehab: dianas.

Utensilios de cocina

En este juego, el paciente verá frente a sí una cocina con varios utensilios. El paciente deberá alcanzar el utensilio que se le indique y, en algunos casos, colocarlos en otro lugar. Los utensilios azules se alcanzarán con la mano izquierda y los rojos con la derecha (ver figura 2.10).

Figura 2.10: Juegos de VirtualRehab: utensilios de cocina.

Mantener a flote

En este juego, el paciente se encuentra en un bote que se está hundiendo. Para evitarlo, deberá tapar los agujeros del casco del bote para evitar que entre agua. Los agujeros azules se taparán con la mano izquierda y los rojos con la mano derecha (ver figura 2.11).

Figura 2.11: Juegos de VirtualRehab: mantener a flote.

Capítulo 3

Planificación

En este capítulo se describen todos los aspectos relativos a la gestión del proyecto: metodología, planificación, costes y riesgos.

3.1. Metodología de desarrollo

El producto final del proyecto es complejo. El hecho de que el usuario tenga que portar físicamente una serie de componentes hardware hace que necesite ser probado a medida que se desarrolla para garantizar, no sólo que es útil, sino también cómodo (usable). La única forma de probarlo es generar versiones funcionales durante el proyecto para comprobar que los requisitos se cumplen. Las pruebas realizadas provocarán que los requisitos iniciales puedan ir cambiando a lo largo del proyecto, así como también que puedan surgir algunos nuevos.

Las características de este proyecto hacen necesario que se siga una **metodología de desarrollo ágil**.

Estas metodologías facilitan la incorporación de nuevos requisitos y la fácil modificación de los ya existentes en cualquier fase del proyecto. Se centran en la comunicación continua con el cliente o el usuario final, proporcionando versiones funcionales del software cada cierto tiempo (semanas) para que pueda probarlas y dar sus impresiones, así como sugerir cambios en los requisitos. De esta forma aumentan las probabilidades de que el producto final satisfaga las necesidades y resulte exitoso, dado que se ha ido aprobando durante todo el proceso de desarrollo del proyecto.

El proyecto se dividirá en etapas (o sprints) durante las cuales se desarrollarán distintos requisitos u objetivos. En cada etapa se analizarán los requisitos a desarrollar, se diseñará una solución y se implementará. Al finalizar la etapa, habrá un producto funcional. En base a los resultados obtenidos de este producto, si es necesario se realizarán cambios en los requisitos o se añadirán nuevos.

- Sprint 1. Definición de objetivos, alcance y requisitos del proyecto.
- Sprint 2. Diseño y modelado del primer soporte.

- Sprint 3. Análisis, diseño e implementación de detección de color con OpenCV.
- Sprint 4. Creación del proyecto Unity e integración con librería de detección de color en Windows.
- Sprint 5. Creación APK Android de la librería de detección de color e integración con Unity.
- Sprint 6. Diseño y modelado del segundo soporte.
- Sprint 7. Análisis, diseño e implementación de escenas.
- Sprint 8. Arreglos, mejoras y optimización del sistema.
- Sprint 9. Análisis, diseño e implementación de nuevo sistema de control gestual.

3.2. Diagrama de Gantt

En la figura 3.1 se muestra el diagrama de Gantt con los tiempos de desarrollo finales del proyecto.

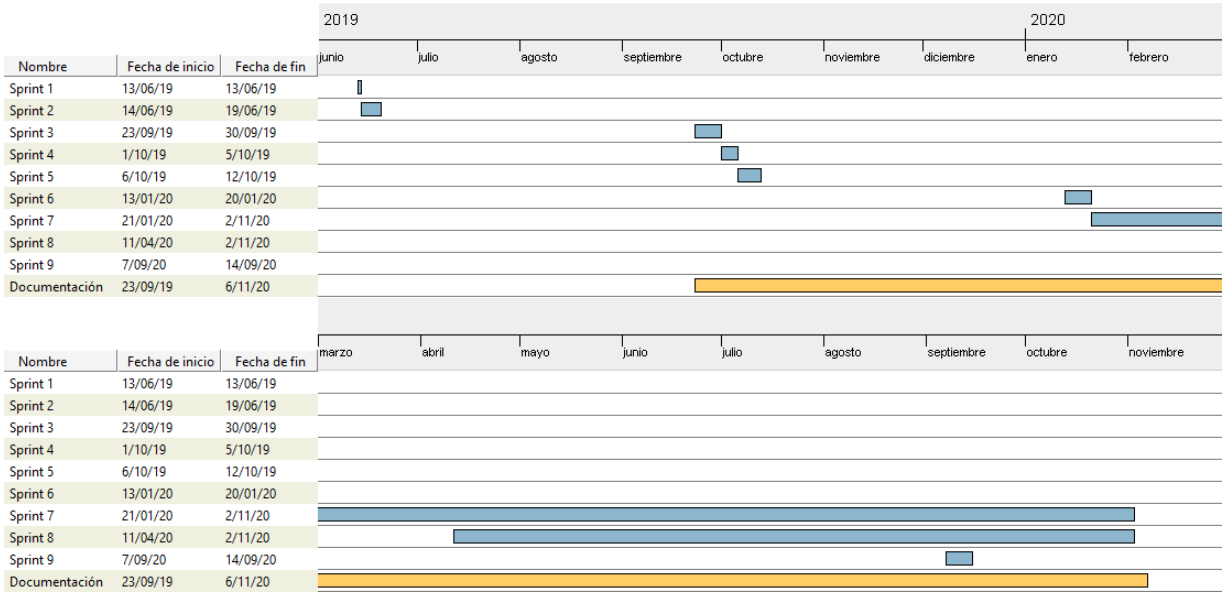


Figura 3.1: Diagrama Gantt de la planificación temporal.

3.3. Costes

A continuación, en el cuadro 3.1, se indica el gasto invertido en el desarrollo del proyecto.

Descripción	Precio (€)
Smartphone	300
Proyector	250
Impresora 3D	435
Material impresión 3D	5
Accesorios dispositivos hardware	35
Costes de desarrollo*	14.400
Total	15.425

Cuadro 3.1: Costes

* Los costes de desarrollo se han obtenido calculando el salario medio anual de un ingeniero informático junior en la provincia de Cádiz.

3.4. Gestión de riesgos

A continuación, se enumerarán los riesgos identificados con el desarrollo del proyecto, la probabilidad de que ocurran, el nivel de impacto, las consecuencias que tendrían, y cómo actuar para prevenirlos o reducir sus efectos.

Riesgo	Error en la detección de los colores
Probabilidad	Alta
Impacto	Crítico
Consecuencias	El sistema quedaría inservible puesto que el método principal de interacción con el usuario depende de ello.
Actuación	Este error puede deberse a que las condiciones lumínicas del entorno no son adecuadas, a que los colores de los marcadores no son los esperados o a la existencia, en el entorno capturado por la cámara, de elementos con los colores esperados que no son los marcadores de colores. Solucionar estos problemas debería acabar con el error.

Cuadro 3.2: Riesgo 1. Error en la detección de los colores.

Riesgo	Falta de potencia computacional
Probabilidad	Media
Impacto	Grave
Consecuencias	La ejecución de la aplicación mostraría lag, sobre todo en la detección de colores.
Actuación	Este error se dará si el smartphone utilizado no tiene suficiente potencia para ejecutar todos los procesos que se llevan a cabo durante la ejecución de la aplicación, sobre todo en los relacionados con la detección de colores, los cuales son los más exigentes. Utilizar un smartphone con más potente pondría fin al error.

Cuadro 3.3: Riesgo 2. Falta de potencia computacional.

Riesgo	Error en la calibración de las posiciones de los colores
Probabilidad	Alta
Impacto	Crítico
Consecuencias	El sistema quedaría inservible puesto que el método principal de interacción con el usuario depende de ello.
Actuación	Para que la calibración sea lo más precisa posible la imagen proyectada ha de estar lo más centrada posible con el objetivo de la cámara. Procurar que la proyección esté, al menos, a la misma altura que el objetivo de la cámara y repetir la calibración debería solucionar el problema.

Cuadro 3.4: Riesgo 3. Error en la calibración de las posiciones de los colores.

Parte II

Desarrollo

Capítulo 4

Análisis de requisitos del sistema

En este capítulo se presentan las necesidades de negocio, el catálogo de requisitos del sistema y los actores. Finalmente, se describen las diferentes alternativas tecnológicas.

4.1. Necesidades de negocio

El motivo principal por el que desarrollado este proyecto es crear aplicaciones de apoyo en actividades de rehabilitación de pacientes, proporcionando una nueva plataforma con la que realizar los ejercicios que conforman dichas actividades. De esta forma se demuestra que esta tecnología puede ser más que un simple método de interacción con un sistema, y que realmente puede llegar a ofrecer ayuda a personas que realmente la necesitan.

Además de ello, puede servir para dar un paso más en la *computación ubicua*.

La computación ubicua trata sobre los ordenadores tomando presencia en todos y cada uno de los aspectos de nuestra vida, en cualquier lugar y momento, rodeándonos y comunicándonos. El término fue acuñado por primera vez por Mark Weiser, el cual es considerado el padre de la computación ubicua, y establece ciertos paralelismos con otras tareas de nuestra vida diaria, las cuales realizamos sin mayor esfuerzo debido a la costumbre que hemos desarrollado hacia ellas. Pone como ejemplo la escritura, la cuál fue un revolución en el momento de su creación, pero que a día de hoy está tan implementada en nuestras vidas que ni siquiera le damos importancia. Weiser considera que los ordenadores son la escritura de nuestra era, aunque todavía no hemos llegado a tal punto. Llegaremos a él cuando la integración los ordenadores en nuestra vida sea tan grande que no necesitemos realizar apenas esfuerzos y siquiera prestar atención para utilizarlos, como si sólo estuviésemos escribiendo [16].

4.2. Catálogo de actores

El sistema sólo cuenta con un tipo de actor: el ***usuario***.

Éste tendrá acceso a todas las funcionalidades del sistema.

4.3. Requisitos funcionales

Descripción de la funcionalidad del sistema

El usuario interaccionará con el sistema utilizando moviendo los dedos delante de la cámara, como si de un panel táctil se tratase, pero sin la necesidad de tocar físicamente la superficie donde se muestra la proyección. Cuando mueva los dedos, unos punteros que los representan se moverán en la escena y permitirán al usuario interaccionar con los distintos elementos.

Para detectar los dedos del usuario se utilizarán marcadores de colores que el usuario colocará en sus dedos. El sistema reconocerá los colores y obtendrá su posición.

El usuario se coloca los marcadores de colores en los dedos y el conjunto de dispositivos (teléfono móvil, proyector y espejo) en el torso. A continuación, se posiciona a, aproximadamente, un metro de distancia de la pared en la que se va realizar la proyección.

Una vez iniciado el sistema, éste mostrará al usuario dos puntos de referencia para que los señale. De esta forma, se establecerá la zona en la que se capturarán las manos del usuario.

El menú principal permitirá al usuario comenzar las distintas pruebas desarrolladas, cambiar el tamaño de los elementos de la interfaz para hacerla más accesible, o cerrar la aplicación.

Las pruebas y funcionalidades desarrolladas contendrán elementos que podrán ser seleccionados o deslizados con los punteros.

El usuario podrá interaccionar con el sistema a través de la pantalla táctil del teléfono móvil en lugar de con la detección de colores si así lo desea.

Se identifican los siguientes **casos de uso** (ver figura 4.1):

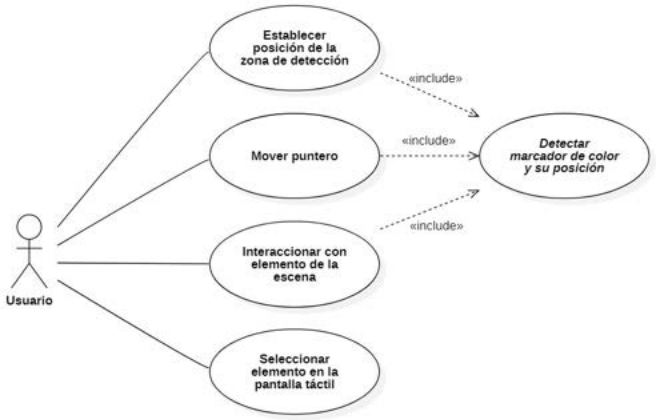


Figura 4.1: Diagrama de casos de uso.

Caso de uso I - Detectar marcador de color y su posición. (Abstracto)

Descripción. El sistema deberá detectar en la imagen capturada por la cámara el marcador del color indicado y calcular su posición en cada momento.

Precondición. El color que se va a detectar ha sido indicado y la cámara funciona.

Secuencia.
Paso 1. La cámara captura y transmite la imagen al sistema.

Paso 2. El sistema detecta todas las secciones de la imagen que sean del color indicado.

Excepción 1. El sistema no detecta ninguna sección de la imagen que sea del color indicado. Los pasos siguientes no se realizan.

Paso 3. El sistema determina cuál de estas secciones es la más prometedora para corresponder con el marcador de color. Se elegirá aquella con mayor área.

Paso 4. El sistema calculará las coordenadas cartesianas del centro de dicho área.

Postcondición. El sistema tiene la posición del marcador en el instante actual.

Caso de uso II - Establecer posición de la zona de detección.

Descripción. El sistema deberá determinar la posición central de la zona en la que se detectaran las manos del usuario. (4.4)

Precondición. La detección de colores está activa, el usuario tiene los marcadores de colores colocados y la escena se está proyectando.

Secuencia.
Paso 1. El sistema muestra dos puntos de referencia en el centro de la escena, uno a la izquierda y otro a la derecha.

Paso 2. El usuario señala el punto de cada lado con el dedo de la mano correspondiente a dicho lado.

Paso 3. El sistema detecta las posiciones señaladas.

Paso 4. El sistema determina que las posiciones se han detectado correctamente.

Excepción 1. El sistema determina que las posiciones no se han detectado correctamente. Se vuelve al paso 1.

Paso 5. El sistema determina que las posiciones detectadas son válidas.

Excepción 2. El sistema determina que las posiciones detectadas no son válidas. Se vuelve al paso 1.

Paso 6. El sistema calcula la posición central de la zona en la que se detectaran las manos del usuario.

Postcondición. La posición de la zona de detección queda establecida.

Caso de uso III - Mover puntero.

Descripción. El sistema deberá permitir al usuario interaccionar con el puntero que representa a su dedo.

Precondición. La detección de colores está activa, el usuario tiene los marcadores de colores colocados y la escena se está proyectando.

Secuencia.
Paso 1. El usuario mueve el dedo frente a la cámara.

Paso 2. El sistema detecta la posición del dedo y mueve el puntero de igual manera que lo ha hecho el dedo.

Excepción 1. El sistema no detecta la posición del dedo. El sistema oculta el puntero en la escena y acaba el caso de uso.

Postcondición. El puntero es movido al igual que el dedo del usuario.

Caso de uso IV - Interaccionar con elemento de la escena.

Descripción. El sistema deberá permitir al usuario interaccionar con elementos de la escena. Para ello, el usuario controlará unos punteros en la escena con el movimiento de sus dedos frente a la cámara.

Precondición. La detección de colores está activa, el usuario tiene los marcadores de colores colocados y la escena se está proyectando.

Secuencia.
Paso 1. El usuario mueve el puntero hacia un elemento de la imagen proyectada.

Paso 2. El sistema resalta el elemento seleccionado.

Paso 3. El usuario mantiene el puntero sobre el elemento.

Excepción 1. El usuario deja de mantener el puntero sobre el elemento. El sistema deja de resaltar el elemento y acaba el caso de uso.

Paso 4. El sistema realiza la acción correspondiente a ese elemento.

Postcondición. La acción correspondiente al elemento seleccionado se lleva a cabo.

Caso de uso V - Seleccionar elemento en la pantalla táctil.

Descripción. El sistema deberá permitir al usuario seleccionar elementos de la escena tocándolos en la pantalla táctil del teléfono móvil.

Precondición. Ninguna.

Secuencia.
Paso 1. El usuario selecciona un elemento de la escena en la pantalla táctil del teléfono móvil.

Paso 2. El sistema realiza la acción correspondiente a ese elemento.

Postcondición. La acción correspondiente al elemento seleccionado se lleva a cabo.

4.4. Modelo conceptual

A continuación, en la figura 4.2, se muestra el modelo conceptual que compone el sistema.

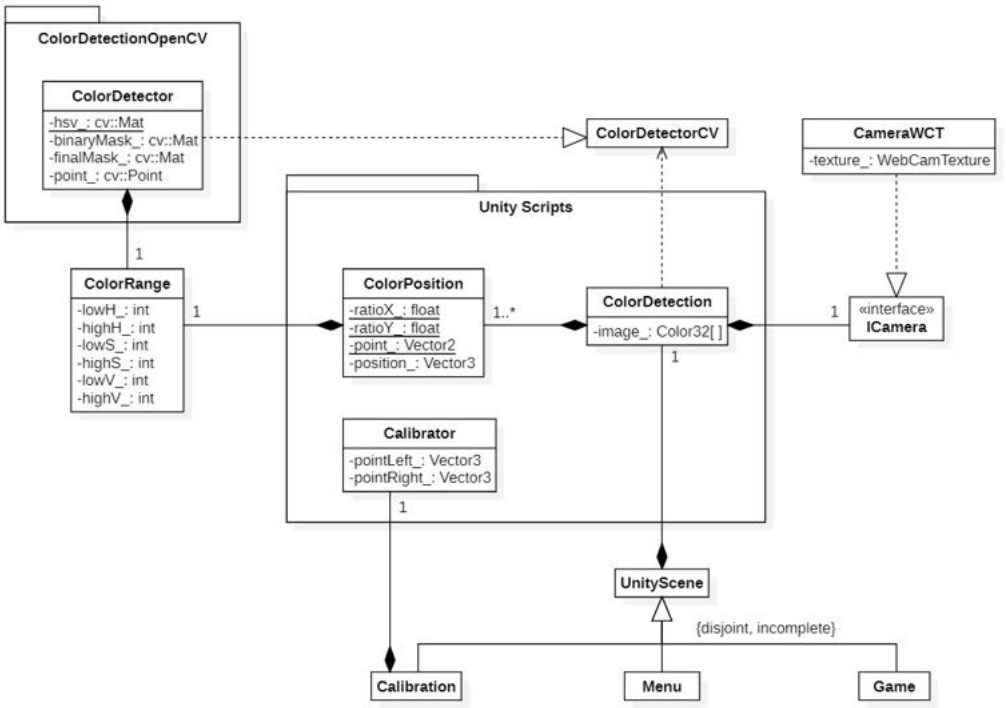


Figura 4.2: Modelo conceptual del sistema.

Escenas de Unity

Se han incluido en el modelo conceptual escenas de *Unity* en forma de especialización y compuestos (ver figura 4.3). Esto es tan sólo una forma de representar distintos tipos de escena que se han desarrollado. **Las escenas de *Unity* no son clases.**

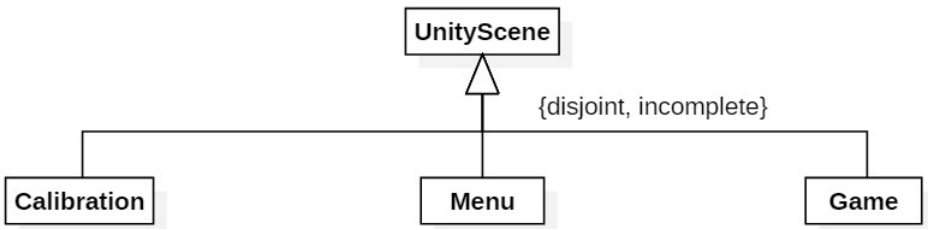


Figura 4.3: Escenas de Unity.

Intervalo de color

El sistema necesita detectar la posición de los dedos del usuario. Para ello, hace uso de marcadores de colores situados en los dedos del usuario. El sistema no detecta las manos del usuario, sino estos marcadores, y los utiliza para saber su posición.

Para indicarle al sistema los colores que se van a utilizar y que, por tanto, ha de detectar, se utilizará el modelo de color HSV (Hue, Saturation, Value – Matiz, Saturación, Valor) (ver figura 4.4).

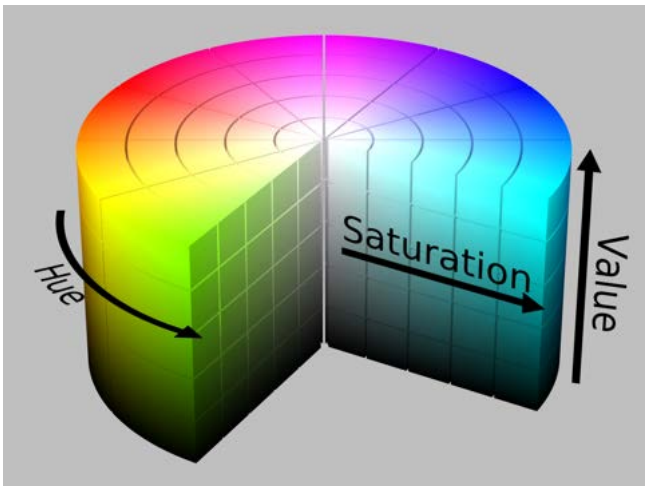


Figura 4.4: Modelo de color HSV.

Debido a que los colores captados por la cámara cambian constantemente por la luz que incide en ellos, cada color se le indicará al sistema en forma de intervalo, con un valor mínimo y un valor máximo de cada componente HSV (ver figura 4.5), cubriendo así un espectro de color suficiente para detectar los colores deseados.

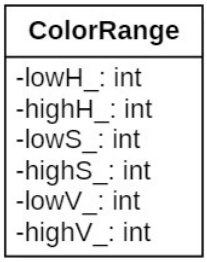


Figura 4.5: Clase ColorRange.

Detector de color

La detección de colores se realiza mediante un algoritmo que hace uso de las librerías de *visión artificial* (*computer vision*) que ofrece *OpenCV* [9]. Estas librerías contienen funciones con las que tratar la imagen captada por la cámara y detectar los colores que deseemos.

Para ello, el algoritmo utiliza distintas estructuras de datos de *OpenCV* (ver figura 4.6) donde contener la imagen original, realizar los tratamientos y almacenar la posición del color.

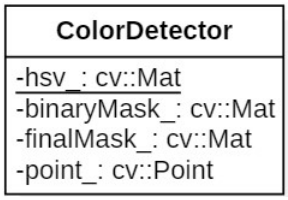


Figura 4.6: Clase ColorDetector.

Las librerías de *OpenCV* están disponibles en *C++* y *Python*. Para poder utilizar el algoritmo (escrito en *C++*) en *Unity*, que utiliza como lenguaje de programación *C#*, se exporta el código en *C*. Se utiliza lo que podríamos llamar una clase de envoltura que importa las funciones en *C* y permite su utilización en *C#* (ver figura 4.7).

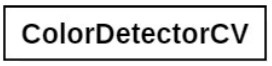


Figura 4.7: Clase ColorDetectorCV.

Calibrador

El usuario moverá los punteros moviendo sus propios dedos frente a la cámara. Para ello, hay que determinar la posición frente a la cámara en la que realizará estos movimientos.

Para ello, se obtiene los ratios de las coordenadas X e Y de la siguiente forma:

$$ratioX = \frac{resolucionHorizontal_{escena}}{resolucionHorizontal_{camara} * 0{,}9}$$

$$ratioY = \frac{resolucionVertical_{escena}}{resolucionVertical_{camara} * 0{,}9}$$

La posición central de la zona de detección será la media de las posiciones detectadas de los dedos mientras señalaban los puntos de referencia (ver figura 4.8).

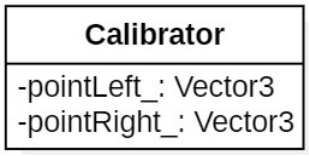


Figura 4.8: Clase Calibrator.

Posición del color

Para determinar la posición de la escena en la que debe estar el puntero se utiliza el punto central de referencia, los ratios de las coordenadas X e Y calculados previamente y la posición detectada de los marcadores de colores (ver figura 4.9).

Si la cámara está grabando horizontalmente, se intercambian las coordenadas X e Y y se invierte la coordenada X.

$$posicion_{escena}(x, y) =$$
$$(ratioX * (coordY_{detectada} - coordX_{puntoRef}), ratioY * (-coordX_{detectada} - coordY_{puntoRef}))$$

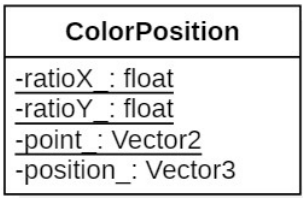


Figura 4.9: Clase ColorPosition.

Detección de color

Para realizar todo el proceso de detección de colores, es necesario gestionar la captura de imagen de la cámara, la selección de los colores a detectar, su indicación al detector de color, las llamadas a las funciones que componen el algoritmo de detección de color y la actualización de las posiciones detectadas.

La imagen capturada es transferida al algoritmo y éste devuelve la posición de los colores indicados.

Para la gestión de la cámara, se utiliza la clase de *Unity WebCamTexture*. En caso de que se quiera utilizar otro método, se implementa una interfaz de adaptación para permitir convertir la interfaz de dicho método en la esperada (ver figura 4.10).

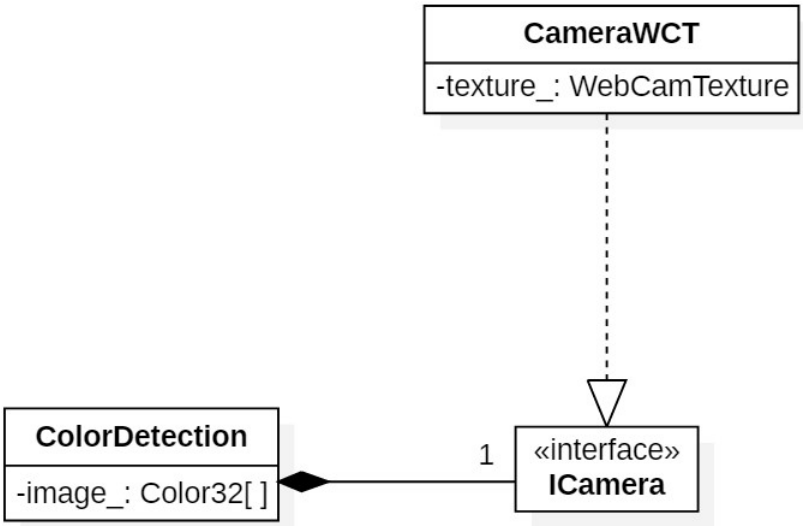


Figura 4.10: Clase ColorDetection e interfaz ICamera.

4.5. Requisitos no funcionales

Requisito no funcional I - Portabilidad a otros sistemas operativos.

Descripción. La aplicación se desarrollará para dispositivos móviles *Android* y *Windows*, principalmente. Su desarrollo en *Unity* facilitará el desarrollo para otros sistemas en el futuro.

Requisito no funcional II - Rendimiento de la detección de colores.

Descripción. El algoritmo de detección de colores ha de ser eficiente para asegurar que la aplicación funcione fluidamente y que la interacción del usuario con el sistema sea correcta. Se deben optimizar los procesos encargados de detectar los colores.

Requisito no funcional III - Mantenibilidad.

Descripción. La aplicación desarrollada debe poder ser ampliada, mediante la adición de nuevas funcionalidades; mejorada, optimizando los algoritmos o refactorizando lo posible; o reparada, en caso de que se encuentren errores en su funcionamiento. Se ha de procurar que todos los componentes desarrollados sean los más mantenibles posible, realizando un buen diseño del sistema, escribiendo un código limpio, claro y, si es necesario, bien comentado.

Capítulo 5

Diseño del sistema

En este capítulo se recoge la arquitectura general del sistema de información y el diseño de la interfaz de usuario.

5.1. Arquitectura del Sistema

En esta sección se define la arquitectura general del sistema de información, especificando la infraestructura tecnológica necesaria para dar soporte al software y la estructura de los componentes que lo forman.

5.1.1. Arquitectura Física

En este apartado describimos los principales elementos hardware que forman la arquitectura física de nuestro sistema:

Teléfono móvil. El software es ejecutado en un teléfono móvil con sistema operativo Android 4.1 ó superior que disponga de una cámara. Los smartphones multicámara deben funcionar correctamente, según muestran las pruebas de sistema realizadas.

El sistema ha sido probado con un ***Xiaomi Mi 9 SE***:

- Procesador *Qualcomm® Snapdragon™ 712* 2.3GHz
- Memoria RAM 6 GB LPDDR4x (1866 MHz) de doble canal
- Triple cámara trasera
- *Android 10* (las primeras pruebas se realizaron con *Android 9*)

Proyector. La proyección se realiza mediante un proyector portátil con conexión inalámbrica. Si no dispone de conexión inalámbrica o esta es inestable, será necesario conectar el teléfono móvil al proyector por cable. Por ejemplo, mediante un adaptador USB - HDMI, en el caso de que ambos dispositivos dispongan de este tipo de conectores I/O.

En este caso, el sistema ha sido probado con un proyector ***LG PH150G***.

5.1.2. Arquitectura Lógica

El software del sistema está formado por dos componentes: la detección de color y las escenas de *Unity* (ver figura 5.1).

Detección de color. Desarrollada en *C++* con *Visual Studio 15* y *Android Studio*, utilizando la librería de visión artificial *OpenCV*. Se compila de manera que las funciones desarrolladas sean exportadas en *C* y dando como resultado un *dll* para su uso en *Windows* y un *apk* para su uso en dispositivos *Android*.

Escenas de Unity. Desarrolladas en *C#* con *Visual Studio 15* y *Visual Studio 17*. Importa las funciones desarrolladas para la detección de color desde el *dll* o desde la *apk* generadas previamente, dependiendo de si la aplicación va a ser compilada para *Windows* o *Android*, respectivamente.

Se tratará con más detalle el proceso de compilación y exportación/importación en el capítulo 6.2 **Construcción del sistema**.

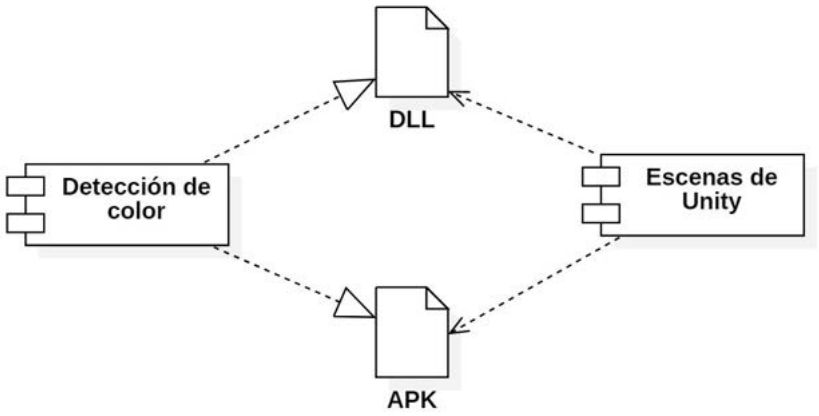


Figura 5.1: Diseño de componentes.

5.2. Diseño de la interfaz de usuario

En esta sección se muestran las escenas que componen la interfaz gráfica de usuario.

- Escena de calibración (ver figura 5.2).
- Menú principal (ver figura 5.3).
- Juego de dianas (ver figura 5.4).
- Vista de tiempos (ver figura 5.5).
- Juego de utensilios de cocina (ver figura 5.6).
- Juego de vestir al muñeco (ver figura 5.7).

Figura 5.2: Interfaz de usuario: escena de calibración.

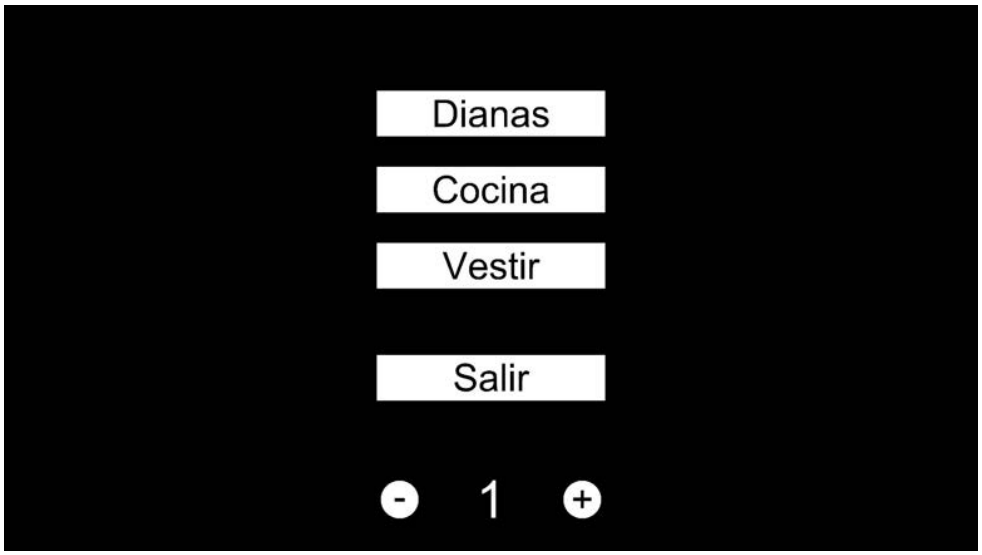


Figura 5.3: Interfaz de usuario: menú principal.

(a) Tutorial.

(b) Diana azul.

(c) Diana roja.

Figura 5.4: Interfaz de usuario: juego de dianas.

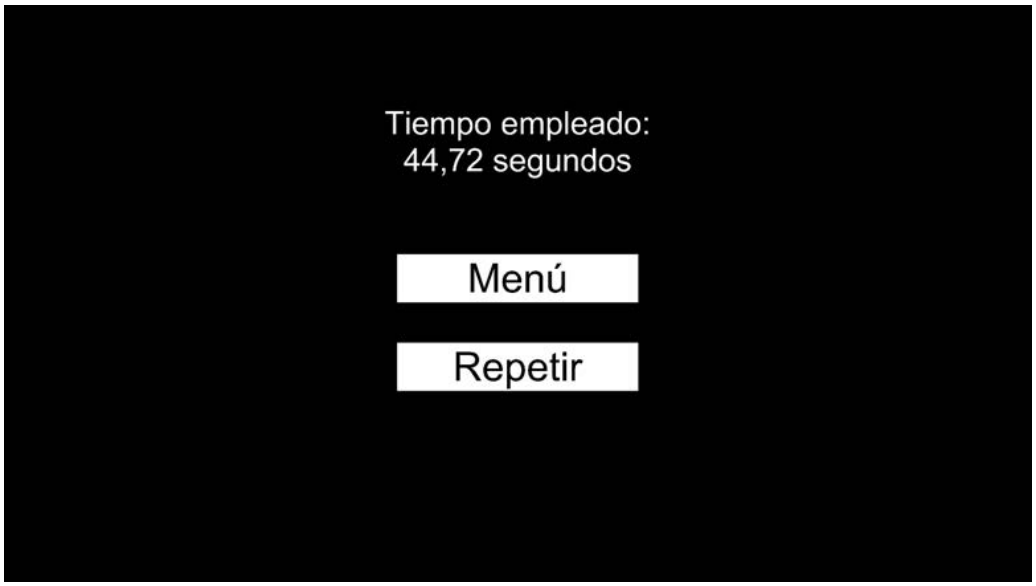
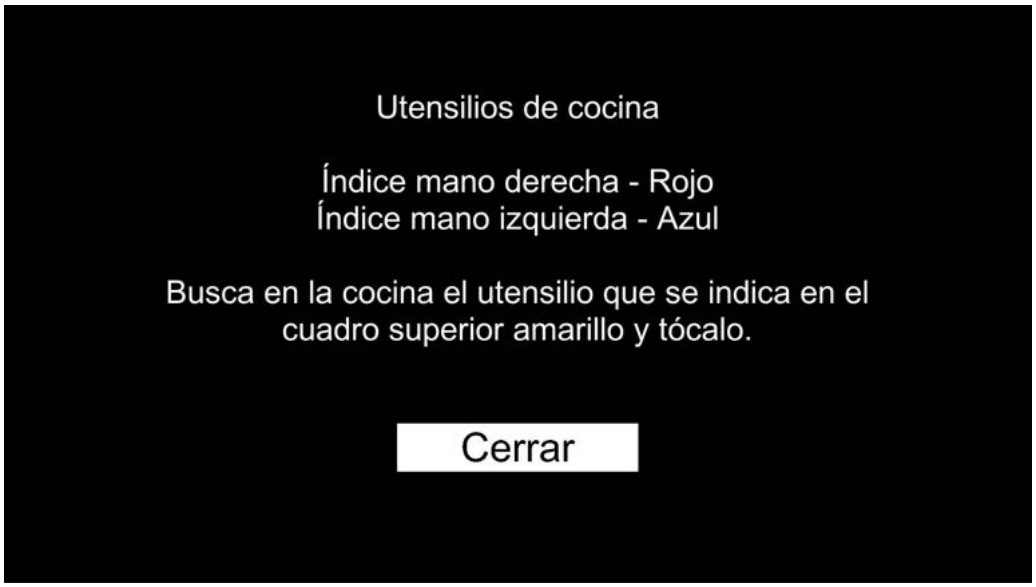
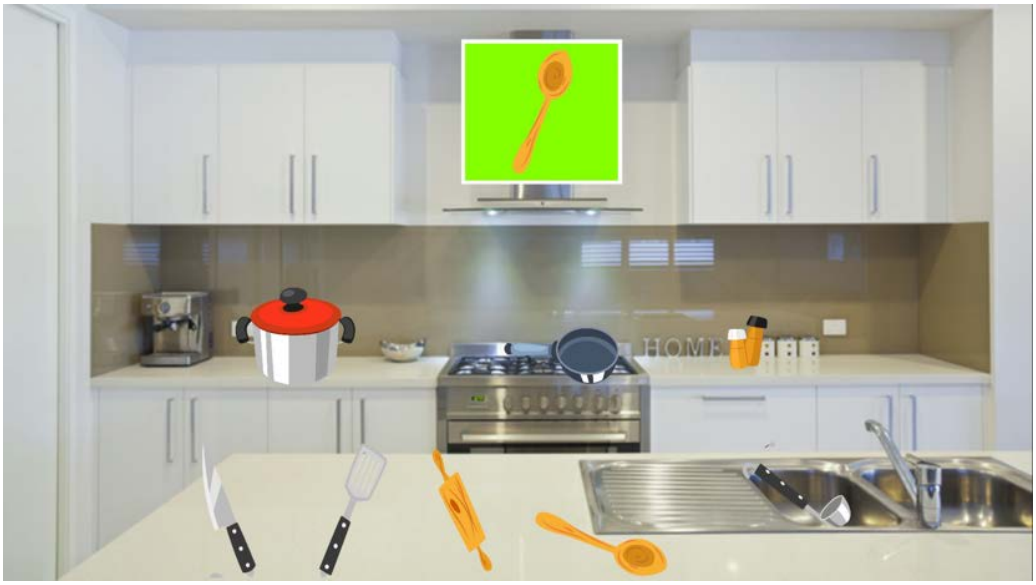


Figura 5.5: Interfaz de usuario: tiempo empleado en prueba.

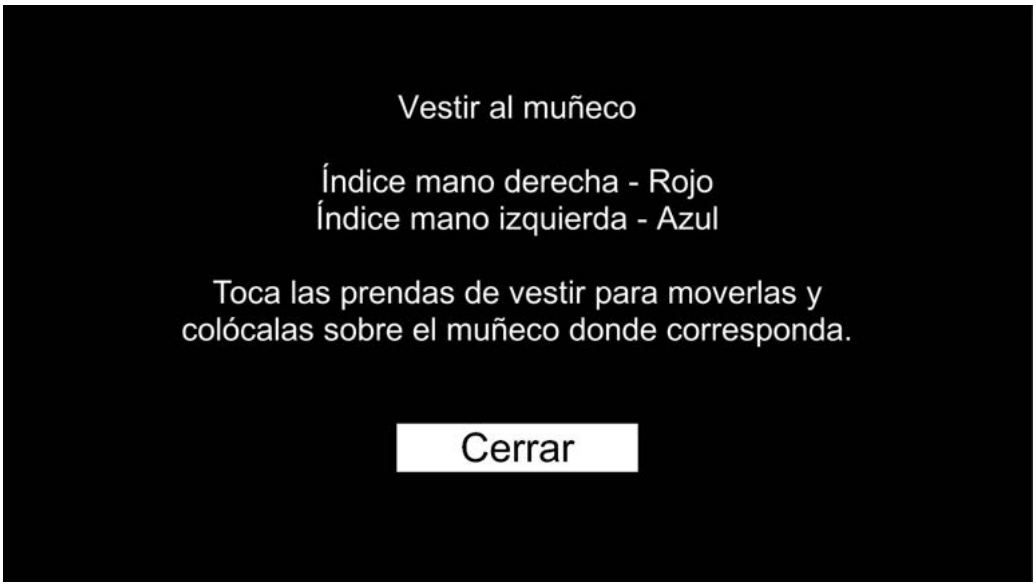


(a) Tutorial.

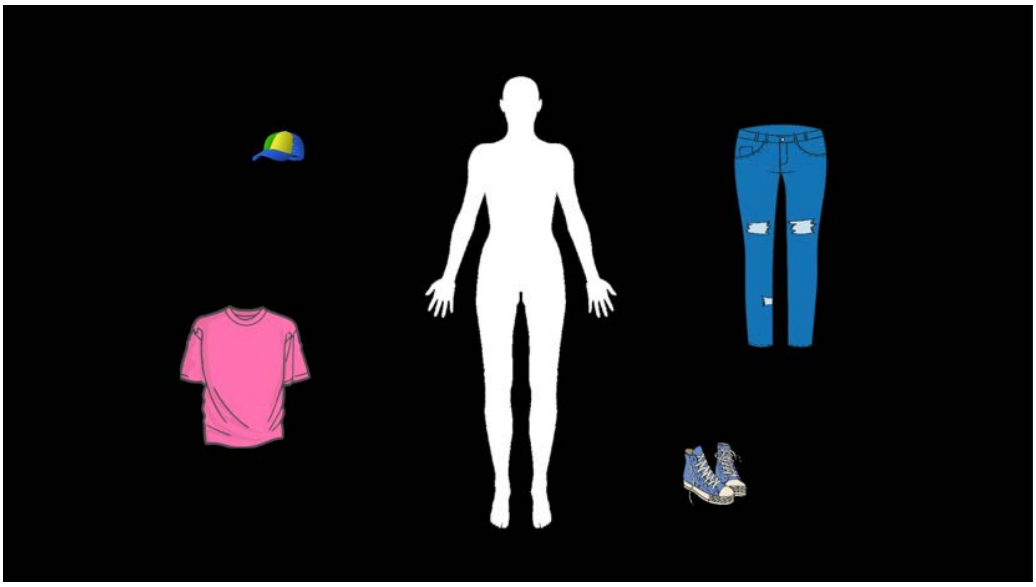


(b) Juego.

Figura 5.6: Interfaz de usuario: juego de utensilios de cocina.



(a) Tutorial.



(b) Juego.

Figura 5.7: Interfaz de usuario: juego de vestir al muñeco.

Capítulo 6

Construcción del sistema

Este capítulo trata sobre todos los aspectos relacionados con la implementación del sistema en código, haciendo uso de un determinado entorno tecnológico.

6.1. Tecnologías

La implementación del sistema puede dividirse en dos fases diferenciadas: la detección de colores y el conjunto de escenas que conforman la aplicación. Para la implementación de estas fases se han utilizado la librería de visión artificial *OpenCV* y *Unity*, respectivamente.

OpenCV ofrece un conjunto de librerías open source con las que tratar una imagen, de manera que, mediante el uso de funciones específicas que ofrecen estas librerías, se pueda desarrollar un algoritmo capaz de detectar colores presentes en esa imagen y obtener la posición en la que se encuentran en tiempo real.

OpenCV for Unity es un plug-in oficial para *Unity* que incluye detección de colores y muchas otras funcionalidades, pero de forma no gratuita, motivo por el que se decide implementar manualmente el sistema de detección de colores. Con el desarrollo manual, además, se dispone de esta funcionalidad para otras actividades académicas.

La elección de *Unity* como entorno de desarrollo de la aplicación final es debido a la facilidad que ofrece para desarrollar aplicaciones. Permite compilación multiplataforma, por lo que, aunque *Android* haya sido elegido como sistema operativo para aplicación final, poder utilizar la aplicación en *Windows* agiliza el proceso de desarrollo, al ser la compilación mucho más rápida para este último, además de facilitar la ampliación a otros sistemas operativos, como *iOS*, en un futuro.

Otra tecnología que se ha planteado aplicar a la aplicación es *Vuforia*, software de desarrollo de aplicaciones de realidad aumentada que dispone de integración con *Unity*. *Vuforia* utiliza marcadores para mostrar elementos en la imagen. Detecta la posición de estos marcadores y muestra los elementos en esa posición. El objetivo de utilizar *Vuforia* en la aplicación era emplearlo en el proceso de calibración, sustituyendo la señalización del usuario de las esquinas de la imagen proyectada y la detección de los colores de los marcadores situados en sus dedos por los marcadores empleados por *Vuforia*. Se situarían los

marcadores en las esquinas de la imagen proyectada y *Vuforia* detectaría la posición. En el momento de implementarlo se encontró un problema: *Vuforia* utiliza su propia interfaz para la obtención de la imagen a través de la cámara y el formato de la imagen obtenida no correspondía con el utilizado por el sistema de detección de colores implementado con *OpenCV*. *Unity* ofrece múltiples funciones para tratar imágenes y cambiar el formato de éstas, pero, tras varias pruebas, no se logró cambiar el formato original al deseado.

6.2. Algoritmo de detección de color

El proceso de detección de colores se realiza mediante las funciones de tratamiento de imágenes y visión artificial de *OpenCV* en su versión en *C++*. Con estas funciones se ha desarrollado un algoritmo que detecta en la imagen todos las figuras de un color indicado, calcula el área de cada una de ellas y devuelve el centro de la mayor.

Esta versión del sistema está pensada para ser utilizada con una superficie clara para la proyección, sin elementos del mismo color que los de los marcadores de colores frente a la cámara. Aun así, debido a las condiciones de luz que puedan darse, pueden detectarse los colores de los marcadores de colores en lugares donde estos colores no están presentes, pero formando siempre figuras muy pequeñas. Los marcadores de colores serán elementos más grandes que estas falsas figuras. Este es el motivo por el cual se elige aquella figura con mayor área.

El algoritmo está encapsulado en una clase ***ColorDetector***. Cada instancia de esta clase detectará un color.

Paso I - Conversión a formato HSV

El primer paso de este algoritmo es convertir la imagen obtenida de la cámara al formato *HSV* (ver figura 6.1). Este paso es necesario para detectar todos y cada uno de los colores indicados. Las imágenes se representan mediante matrices. Dado que la matriz resultado de este paso es la misma para todos los colores a detectar, puesto que no depende de estos, se realiza una sola vez, de manera estática, con el fin de ahorra y optimizar recursos. El resultado se almacena en una matriz común (estática) a todas las instancias de la clase.

(a) Imagen de la cámara.

(b) Imagen en formato HSV.

Figura 6.1: Detección de color: conversión a formato HSV.

Paso II - Binarización de imagen

Binarizar una imagen significa convertir el color de todos sus píxeles a negro (0) o blanco (255). Se realiza comprobando si el color de cada píxel se encuentra entre los valores de un rango dado. Si lo está, se convierte en blanco. Si no, en negro.

El rango indicado es el del color que se quiere detectar. Se utiliza un rango y no un valor exacto para permitir cierta holgura en la detección. De la otra forma, sólo detectaría una tonalidad muy concreta del color, lo cual no sería funcional, ya que, al incidir la luz sobre el color, este ofrece muchas tonalidades a la vez. Cuando mayor sea el rango indicado, mayor será la sensibilidad de la detección, y viceversa, por lo que es importante elegir correctamente el rango a detectar.

En el formato *HSV*, los valores de *H* se representan mediante grados, siendo el mínimo 0 y el máximo 360. *OpenCV* los representa de igual forma, pero estrechando el rango a la mitad, siendo el mínimo 0 y el máximo 180.

Por ejemplo, el color azul, en formato *HSV*, se encuentra alrededor del valor 240 para *H*. Se elige el rango 220-260, detectando así un espectro mayor de azules, encontrándose la tonalidad de azul que se desea detectar entre estos valores. Para adaptarlos a la representación del formato *HSV* en *OpenCV*, se reducen los valores a la mitad, quedando el rango 110-130.

Una vez aplicada la binarización, quedará como resultado una matriz, a la que se llamará **máscara**, cuyos valores serán 0 y 255 (negro y blanco, respectivamente). Los píxeles en blanco serán aquellos en los que se ha detectado el color deseado (ver figura 6.2).

(a) Máscara del color azul.

(b) Máscara del color rojo.

Figura 6.2: Detección de color: binarización de imagen.

Paso III - Mejora de la máscara

Las figuras formadas por los píxeles en blanco tendrán imperfecciones fruto de la luz incidente en la superficie de los objetos, que provoca brillos blancos que impiden que se detecte el color. Para calcular el área de estas figuras primero hay que eliminar estas imperfecciones, tratando de formar polígonos, es decir, figuras geométricas cerradas. De no ser así, la función que calcula el área tomará como área de las figuras el de la imagen completa.

La supresión de estas imperfecciones se lleva a cabo mediante funciones que erosionan y dilatan las figuras, realizando transformaciones morfológicas avanzadas (ver figura 6.3).

(a) Máscara mejorada del color azul.

(b) Máscara mejorada del color rojo.

Figura 6.3: Detección de color: mejora de la máscara.

Una vez las figuras de la máscara están libres de imperfecciones, se obtienen sus bordes exteriores (ver figura 6.4). Para ello, se hace uso de una función que utiliza el algoritmo de detección de bordes de John F. Canny. [1]

(a) Contornos del color azul.

(b) Contornos del color rojo.

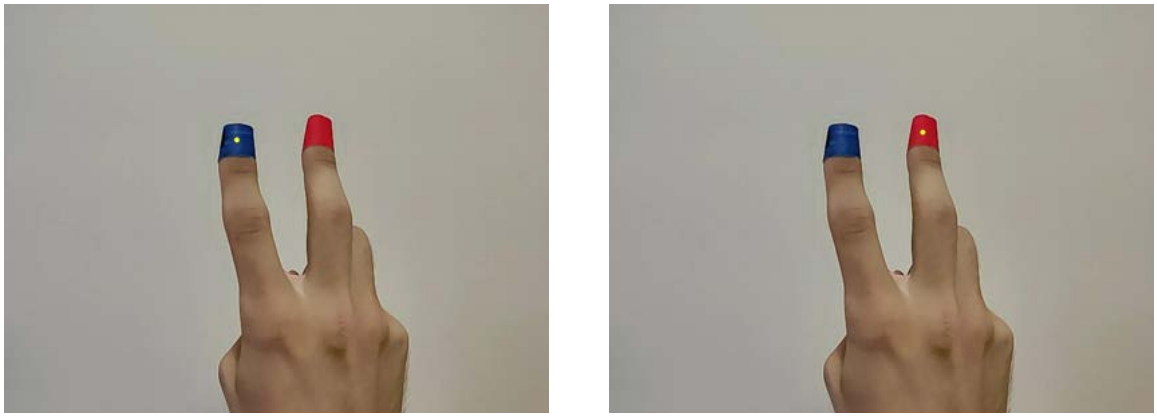
Figura 6.4: Detección de color: contornos de los colores.

Paso IV - Encontrar el mejor contorno

Se obtienen los puntos de la máscara (píxeles) que forman cada uno de los bordes de las figuras y se agrupan en conjuntos de puntos a los que llamaremos **contornos**. Estos contornos se utilizan para calcular el área de las figuras a través de una función. Se compara el valor del área de cada figura y se selecciona aquella con mayor valor.

Paso V - Centro del contorno

Por último, se halla el centro aproximado del contorno (ver figura 6.5), el cual representará la posición del dedo del usuario en el sistema final. El centro es hallado creando un rectángulo alrededor de los límites del contorno y hallando el centro de este rectángulo.



(a) Centro del mejor contorno azul.

(b) Centro del mejor contorno rojo.

Figura 6.5: Detección de color: centro del contorno.

Compilación y exportación

Como se ha mencionado en apartados anteriores, el código del detector de colores está escrito en *C++*, siendo este incompatible con el lenguaje utilizado en *Unity*, *C#*. Esto hace necesario exportar todas las funciones implementadas a *C*, lenguaje con el que ofrecen compatibilidad tanto *C++* como *C#*.

Solucionado el problema de la compatibilidad, surge otro: *C* no tiene clases, lo que impide la exportación de la clase ***ColorDetector***. La solución que se le ha dado a este problema es la implementación de una serie de funciones que permiten la creación, modificación y eliminación de instancias de esta clase, así como el acceso a los métodos públicos de estas instancias. A cada instancia se le asigna un identificador, pudiendo manipular la instancia deseada en cada momento. Estas funciones están definidas en *C*.

6.3. Implementación en Unity

Para manejar el detector de color, se importan las funciones mediante el espacio de nombres *System.Runtime.InteropServices* de *Microsoft .NET* [5]. Este espacio de nombres proporciona métodos para operar con códigos de API's externas.

Script ColorDetection

Un script se encarga de conectar la cámara y recibir su imagen, emplear las funciones de creación, modificación y eliminación de instancias de ***ColorDetector*** y de llamar a los métodos de éstas. Con el fin de agilizar las llamadas a las funciones, se han implementado hilos de ejecución (clase *Thread* [3]) que se encargarán de las llamadas a los métodos de cada una de las instancias creadas. Se usan barreras (clase *Barrier* [2]) para sincronizar la ejecución de los hilos con la actualización de la imagen de la cámara.

El proceso de sincronización con barreras es el siguiente:

<table>
<tr><th>Proceso principal</th><th>Cada hilo</th></tr>
<tr><td colspan="2">Inicialización de barreras e hilos.</td></tr>
<tr><td>Inicializa dos barreras
(StartBarrier y EndBarrier)
con un participante cada una.</td><td></td></tr>
<tr><td></td><td>Añade un participante a
cada una de las barreras.</td></tr>
<tr><td colspan="2">Proceso de ejecución.</td></tr>
<tr><td>Captura la imagen de la cámara.</td><td rowspan="2">Señala la barrera StartBarrier y
espera a que los demás participantes
(los demás hilos y el proceso principal)
la alcancen.</td></tr>
<tr><td>Señala las dos barreras y
espera a que los demás participantes
(los hilos) las alcancen.</td></tr>
<tr><td rowspan="2"></td><td>Realiza su proceso.</td></tr>
<tr><td>Señala la barrera EndBarrier y
espera a que los demás participantes
(los hilos) la alcancen.</td></tr>
<tr><td colspan="2">Se repite el proceso de ejecución.</td></tr>
</table>

Cuadro 6.1: Script ColorDetection: proceso de sincronización con barreras.

La interacción con la cámara se realiza mediante una interfaz que declara los métodos necesarios para conectar la cámara y recibir la imagen capturada en tiempo real. Utilizando esta interfaz se pueden implementar clases que utilicen distintas formas de manipular la cámara y adaptar todas sus interfaces en una común.

Script ColorPosition

Este script se encarga del movimiento de un *GameObject* (elemento en una escena de *Unity*) que representa el punto señalado por el usuario. Cada instante en el que el script *ColorDetection* detecta la posición de un color seleccionado en la imagen de la cámara le indica las coordenadas a *ColorPosition*.

En este script se aplican los valores resultados de la calibración explicados previamente en apartado 4.4 **Calibrador**.

```
position_.Set(Values.ratioX * (y - Values.refPoint.x),
              Values.ratioY * (-x - Values.refPoint.y),
              posZ_);
```

Las variables x e y son las coordenadas que el script *ColorDetection* indica al script *ColorPosition* mediante los argumentos de una función.

El *GameObject* asociado al script dispone de un colisionador (*Collider*) que se activa al entrar en contacto con otros *GameObjects* que también dispongan de un colisionador. La activación da lugar a acciones implementadas en otros scripts asociados a estos *GameObjects*.

Scripts de botones

La selección de un botón por parte del usuario activa una animación de aumento de tamaño del propio botón para indicarle al usuario que lo está seleccionando. Si la selección del botón persiste durante un tiempo preestablecido, se activa la acción implementada para ese botón (cambio de escena, cerrar la aplicación, cambiar color de dibujado...). Si el botón es deseleccionado antes de que pase el tiempo, vuelve a su tamaño original.

Este comportamiento se ha implementado en el script *TouchHover*.

```
public abstract class TouchHover : MonoBehaviour, CollidingTimer
{
    public float maxX, maxY, speed;

    protected float time_ = 0f,
                    triggerTime_;

    protected bool triggered_ = false;

    private float x_, y_,
                  varX_, varY_;

    protected void Awake()
    {
        Debug.Log("TouchHover.Awake");
        x_ = transform.localScale.x;
        y_ = transform.localScale.y;
```

```
18
19          varX_ = (maxX - x_) / (maxY - y_) * speed;
20          varY_ = speed;
21      }
22
23      void Update()
24      {
25          if(triggered_)
26          {
27              time_ += Time.deltaTime;
28
29              if (transform.localScale.x < maxX && transform.localScale.y < maxY)
30                  if(transform.localScale.x + Time.deltaTime * varX_ > maxX ||
    transform.localScale.y + Time.deltaTime * varY_ > maxY)
31                      transform.localScale = new Vector3(maxX, maxY, 0.1f);
32                  else
33                      transform.localScale = new Vector3(transform.localScale.x +
    Time.deltaTime * varX_, transform.localScale.y + Time.deltaTime * varY_, 0.1
    f);
34          }
35          else
36              if (transform.localScale.x > x_)
37                  if (transform.localScale.x - Time.deltaTime * varX_ < x_ ||
    transform.localScale.y - Time.deltaTime * varY_ < y_)
38                      transform.localScale = new Vector3(x_, y_, 0.1f);
39                  else
40                      transform.localScale = new Vector3(transform.localScale.x -
    Time.deltaTime * varX_, transform.localScale.y - Time.deltaTime * varY_, 0.1
    f);
41
42          if (time_ > triggerTime_)
43          {
44              triggered_ = false;
45              time_ = 0f;
46
47              Action();
48          }
49      }
50
51      void OnTriggerEnter(Collider coll)
52      {
53          triggered_ = true;
54      }
55
56      void OnTriggerExit(Collider coll)
57      {
58          triggered_ = false;
59          time_ = 0f;
60      }
61
62      public float getTime()
63      {
64          return triggerTime_;
65      }
66
67      protected abstract void Action();
68  }
```

MonoBehaviour es la clase de la que heredan los scripts de *Unity*.

CollidingTimer es una interfaz creada para indicar el tiempo necesario para activar la acción del botón.

Este script está definido como abstracto, siendo en los scripts herederos en los que se define la acción a realizar y el tiempo de espera.

```
public class ExitButton : TouchHover
{
    void Start()
    {
        triggerTime_ = 3f;
    }

    protected override void Action()
    {
        Application.Quit();
    }
}
```

Capítulo 7

Pruebas del sistema

En este capítulo se presentan los diferentes tipos de pruebas que se han llevado a cabo.

7.1. Pruebas unitarias

Se han realizado pruebas unitarias a las distintas unidades funcionales del sistema.

Detección de color

Primero, se ha comprobado que el componente de detección de color realiza todas sus funciones de manera correcta. Esto es necesario, ya que el resto de componentes del sistema dependen de ello en mayor o menor medida, puesto que la detección de color es la base de la interacción con el sistema.

En la librería de *OpenCV* hay funciones que permiten visualizar imágenes (representadas mediante matrices). Mediante estas funciones se puede comprobar si cada uno de los pasos del algoritmo de detección de color realiza su función correctamente o si existen errores en el procedimiento.

```
1 cv::namedWindow(name, CV_WINDOW_AUTOSIZE);
2 cv::imshow(name, mat);
```

Unity

Para realizar las pruebas de las escenas y scripts desarrollados en *Unity*, primero hay que realizar pruebas de integración con el componente de detección de color. Estas pruebas se detallan en la sección 7.2 **Pruebas de integración**, más adelante.

Las pruebas de *Unity* se han realizado compilando las escenas y los scripts para *Windows* y ejecutándolos en el propio editor de *Unity*. Cada cambio es compilado en el momento en cuestión de segundos y puede ser testeado más ágilmente.

7.2. Pruebas de integración

Las pruebas de integración comprueban, valga la redundancia, la integración de los distintos componentes del sistema para comprobar que funcionan conjuntamente de manera correcta.

La integración del componente de detección de color y las escenas de *Unity* se ha comprobado implementando una función que utiliza la función de visualización de imágenes de *OpenCV* utilizada para probar la detección de color y exportándola para poder utilizarla desde *Unity*.

```
1  void showWindow(const char* id)
2  {
3    auto pair = colorDetectors.find(string(id));
4    if (pair != colorDetectors.end())
5    {
6      cd::ColorDetector& colorDetector = pair->second;
7      Mat mat(colorDetector.getFinalMask().clone());
8      circle(mat, { colorDetector.getPoint().x, colorDetector.getPoint().y }, 7,
       (255, 255, 255), -1);
9      putText(mat, string(id), { colorDetector.getPoint().x, colorDetector.
       getPoint().y },
10       FONT_HERSHEY_PLAIN, 1, Scalar(120, 120, 255));
11
12     string name(id);
13     namedWindow(name, cv::WINDOW_AUTOSIZE);
14     imshow(name, mat);
15   }
16 }
```

Este método permite probar la correcta integración de ambos componentes en *Windows*, pero no es válido para *Android*. Para comprobarlo en *Android*, el método empleado es más complicado. Consiste en añadir a la escena un cuadro con la imagen capturada por la cámara para comprobar que ésta captura correctamente. La detección de colores se ha comprobado añadiendo *logs* en pantalla que indican si los colores a detectar se han añadido correctamente a la clase *ColorDetector* y las coordenadas de la posición que detectan (ver figura 7.1).

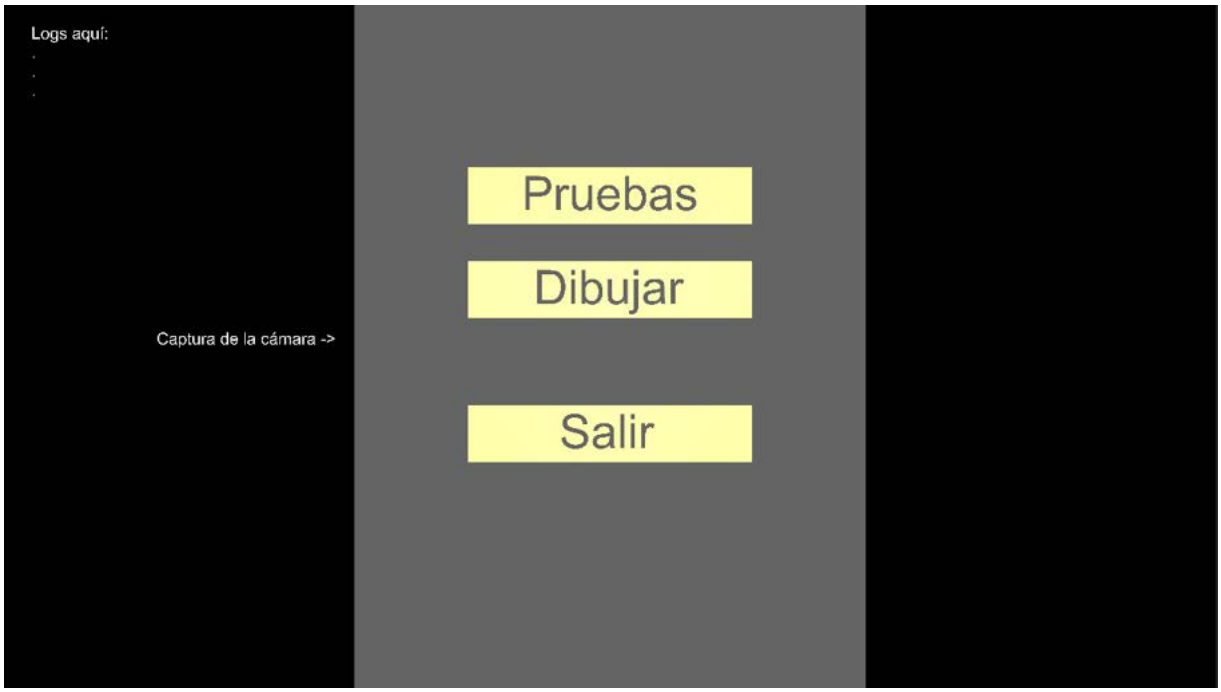


Figura 7.1: Pruebas de integración: detección de color y Unity en Android.

7.3. Pruebas de sistema

Estas pruebas consisten en la comprobación del correcto funcionamiento de la aplicación completa con todos sus componentes y se han llevado a cabo ejecutando la aplicación *Android* con el conjunto de componentes hardware que componen el sistema (smartphone y proyector junto con los soportes necesarios y el espejo).

Durante la ejecución se han probado todas las escenas implementadas de forma consecutiva, comprobando así si realizan su función correctamente, y la conexión inalámbrica entre smartphone y proyector, la cual se realiza a través de *Bluethooth*.

Con estas pruebas se han detectado errores en la disposición de los componentes hardware, los cuales se mueven demasiado al portarlos sobre el torso como para que resulte cómodo utilizar el sistema, en cómo se llevan a cabo determinados procesos durante la ejecución de la aplicación, como puedan ser el método de calibración, que fue rediseñado, o el método de interacción con algunos elementos de las escenas, y en la conexión inalámbrica, la cual no es lo suficientemente rápida, por lo que se recomienda utilizar conexión cableada.

Parte III

Epílogo

Capítulo 8

Conclusiones

En este último capítulo se detallan las lecciones aprendidas tras el desarrollo del presente proyecto y se identifican las posibles oportunidades de mejora sobre el software desarrollado.

8.1. Objetivos alcanzados

Se han alcanzado los objetivos principales marcados al principio del proyecto:

- Se ha obtenido una librería con funciones que permiten detectar colores determinados en una imagen con la cual se puede simular un sistema de **reconocimiento de manos** detectando unos marcadores de colores situados en los dedos. Las posiciones de estos colores en la imagen se toma como posición de los dedos de la mano.
- Se ha logrado la **integración** de esta librería con el entorno de desarrollo *Unity*, desde el cual se gestiona la captura de cámara y la comunica a la librería de detección de colores. La conexión entre dispositivos hardware (smartphone y proyector) se realiza desde las interfaces de estos mismos de forma sencilla.
- Se han pensado y probado varias **disposiciones de los dispositivos** hardware a lo largo del proyecto.
- Se han realizado los modelos 3D de algunos de los **soportes necesarios** para satisfacer las distintas disposiciones de los dispositivos hardware.
- Se han diseñado y **desarrollado distintas funcionalidades** para el sistema con la intención de simular ejercicios llevados a cabo en actividades de rehabilitación.

8.2. Conocimientos y valores adquiridos

Con este proyecto he adquirido numerosos conocimientos técnicos que seguro me serán útiles:

- He mejorado mi nivel de programación en *C++* y *C#* y he aprendido a integrar códigos de ambos lenguajes.

- He desarrollado una aplicación desde cero en *Unity*.
- He trabajado con librerías de visión artificial y he comprendido su funcionamiento a nivel básico.
- He aprendido cómo se modelan objetos 3D para imprimirlos.

En él he querido hacer el mejor trabajo posible, y ello me ha llevado a adquirir altos niveles de paciencia cuando las cosas no salían como quería y también de desesperación cuando, además, no conseguía averiguar por qué durante días. Pero también me ha aportado muy buenos momentos de satisfacción cuando lograba superar esas situaciones y ver el resultado. He aprendido que, a veces, hay que saber poner límites y pisar el freno, saber priorizar y conseguir que lo que realmente necesitas sea bueno. Para ello, he tenido muy presente el *principio de Pareto* [12], el cual acostumbro a cumplir en su parte negativa. Afrontar un proyecto completo en solitario ha sido una experiencia complicada, ya que todo depende de ti mismo y te obliga a buscar soluciones en cualquier medio: libros, foros (mención especial a la comunidad de *StackOverflow*, siempre echando una mano en tus peores momentos), APIs, wikis, etc. Aunque esto no es algo nuevo, ya que aprobar parte de la carrera se basa en ello.

8.3. Trabajo futuro

Muchas de las partes del proyecto han sido llevadas a cabo con unos conocimientos básicos de la materia, aprendidos sobre la marcha. Siendo el objetivo principal de este proyecto desarrollar los componentes necesarios para replicar la tecnología *SixthSense* e integrarlos, hay ciertas mejoras aplicables al producto final que no han sido llevadas a cabo y que pueden ser desarrolladas en un futuro proyecto.

Mejoras como:

Scripts optimizables. La aplicación se ha construido con unos conocimientos básicos de desarrollo en *Unity*. Hay procedimientos en los scripts de código que son optimizables y que mejorarían la eficiencia de la aplicación.

Atractivo de la interfaz gráfica. Siguiendo con el punto anterior, la interfaz gráfica de usuario es sencilla y funcional, pero puede resultar poco atractiva visualmente. Aplicar conocimientos un poco más avanzados de diseño gráfico o de modelado con herramientas como *Blender* o *Google Sketchup* que hagan la aplicación más vistosa mejoraría su usabilidad y aumentaría el incentivo de utilizarla.

Mejor reconocimiento de manos. El reconocimiento de colores como vía para reconocer manos es ingenioso, pero el método empleado requiere muchos recursos y un equipo potente para ser totalmente funcional. La tecnología tendrá muchas más opciones si se mejora este apartado, ya sea mediante la actualización u optimización del algoritmo empleado o mediante la utilización de un sistema distinto, como pueda ser *Leap Motion* [8].

Desarrollo de nuevas pruebas y funcionalidades. Evidentemente, crear nuevas experiencias para esta tecnología es el punto más interesante y el que mejor permite ver la posibilidades que tiene.

Apéndices

Apéndice A

Modelado e impresión 3D

El usuario ha de portar los dispositivos hardware (smartphone y proyector) que utiliza el sistema sobre su torso para grabar y proyectar delante de él.

Durante el desarrollo del proyecto se han ideado distintas posiciones y orientaciones de los dispositivos, buscando el mejor desempeño posible de éstos. La disposición de los dispositivos se ha ido mejorando en base a los resultados obtenidos en cada una de las iteraciones de las pruebas realizadas del sistema.

Para portar los dispositivos se han diseñado unos soportes, utilizando la herramienta de modelado 3D ***OpenSCAD***.

(Los soportes están diseñados para portar los modelos de proyector y smartphone de los que se ha dispuesto durante el desarrollo del proyecto.)

A.1. OpenSCAD

OpenSCAD es una herramienta de modelado 3D open source, disponible para sistemas operativos Linux/UNIX, Windows y Mac OS X.

Su principal diferencia con otras herramientas de modelado 3D, y el motivo por el cual se ha decidido utilizarla, es la forma de modelar objetos 3D. En lugar de modelar de forma interactiva, la herramienta lee el código programado por el usuario, el cual contiene la descripción del objeto a modelar, lo compila y renderiza el modelo 3D.

La ventaja de este método sobre los demás es la precisión milimétrica (y mayor, aunque dependerá de la precisión de la impresora 3D) que ofrece, ya que permite al usuario indicar la posición, medida y rotación exacta de cada uno de los elementos que forman parte del objeto diseñado.

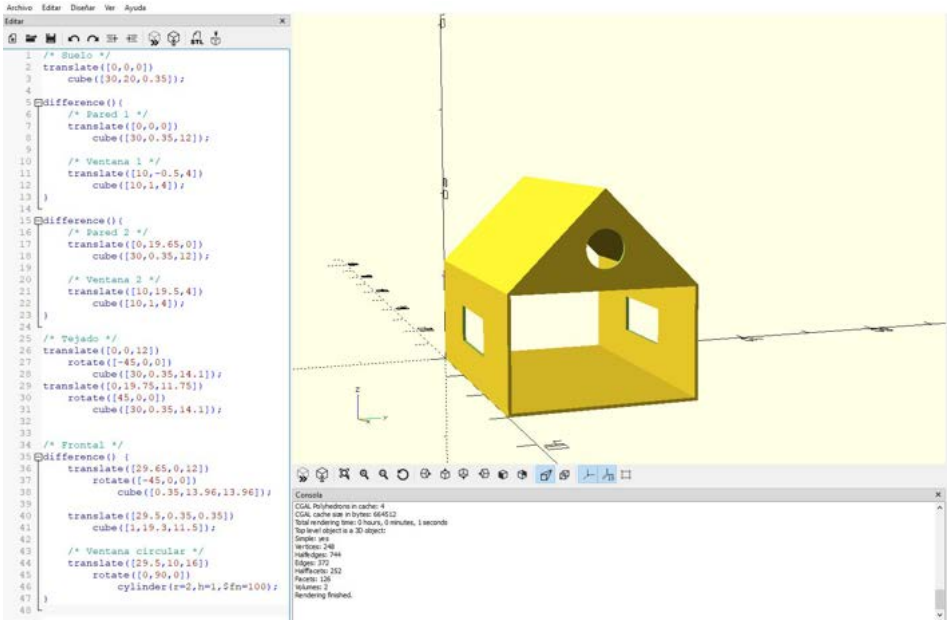


Figura A.1: Ejemplo de modelado 3D con OpenSCAD.

Cuando se modela un objeto 3D, hay una condición que se ha de tener en cuenta: toda superficie necesita una base para ser impresa. Si no la hay, se imprime una base de relleno más débil y menos refinado para facilitar su extracción del modelo impreso. Por ello, se debe diseñar el modelo minimizando las superficies sin base. En el ejemplo presentado en la figura A.1, el tejado y la parte superior de las ventanas de la casa modelada no tienen una base sobre la que imprimirse. Una de las soluciones es cambiar la orientación en la que se imprime, como se ve en la figura A.2.

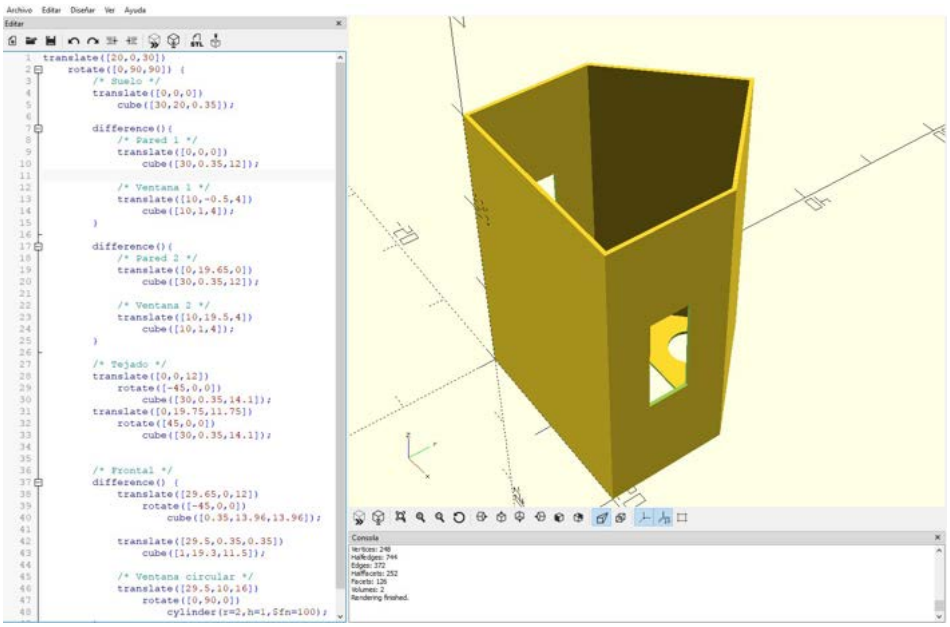


Figura A.2: Ejemplo de modelado 3D con OpenSCAD (2).

De esta forma, sólo el lateral de las ventanas quedaría sin base y se reduciría la cantidad de relleno necesario.

En otros casos, será necesario modelar el objeto por piezas y unirlas. En otros, no habrá más remedio que imprimir una cantidad considerable de relleno.

A.2. Primer modelo

El planteamiento inicial de la disposición de los dispositivos hardware se centraba en situar la lente de la cámara y el foco del proyector los más cerca posible el uno del otro, con el objetivo de disminuir la distancia a la estarían los dedos del usuario de la cámara.

El soporte tiene forma de caja rectangular que cubre parcialmente la superficie del proyector. En su interior se introduce el proyector, verticalmente, con el foco hacia arriba, y el smartphone, situado en la base del proyector, con la cámara hacia el proyector (ver figura A.3).

Figura A.3: Soporte físico para dispositivos hardware (parte frontal).

El espejo en el que se refleja la proyección para mostrarla frontalmente se coloca frente

al foco del proyector. Debido a que el soporte sólo cubre hasta la mitad de la altura del proyector, se ha diseñado una lengüeta que se añade a la parte trasera del soporte y al que se adhiere el espejo, permitiendo colocarlo en la posición correcta.

En los laterales del soporte se han añadido unas rejillas para permitir el flujo de aire por el sistema de ventilación del proyector.

Además, se ha añadido un hueco para permitir el acceso al botón de encendido/apagado del proyector y otro par de huecos para poder pasar por ellos una cinta o cuerda para poder colgar el conjunto del cuello o del pecho del usuario (ver figura A.4).

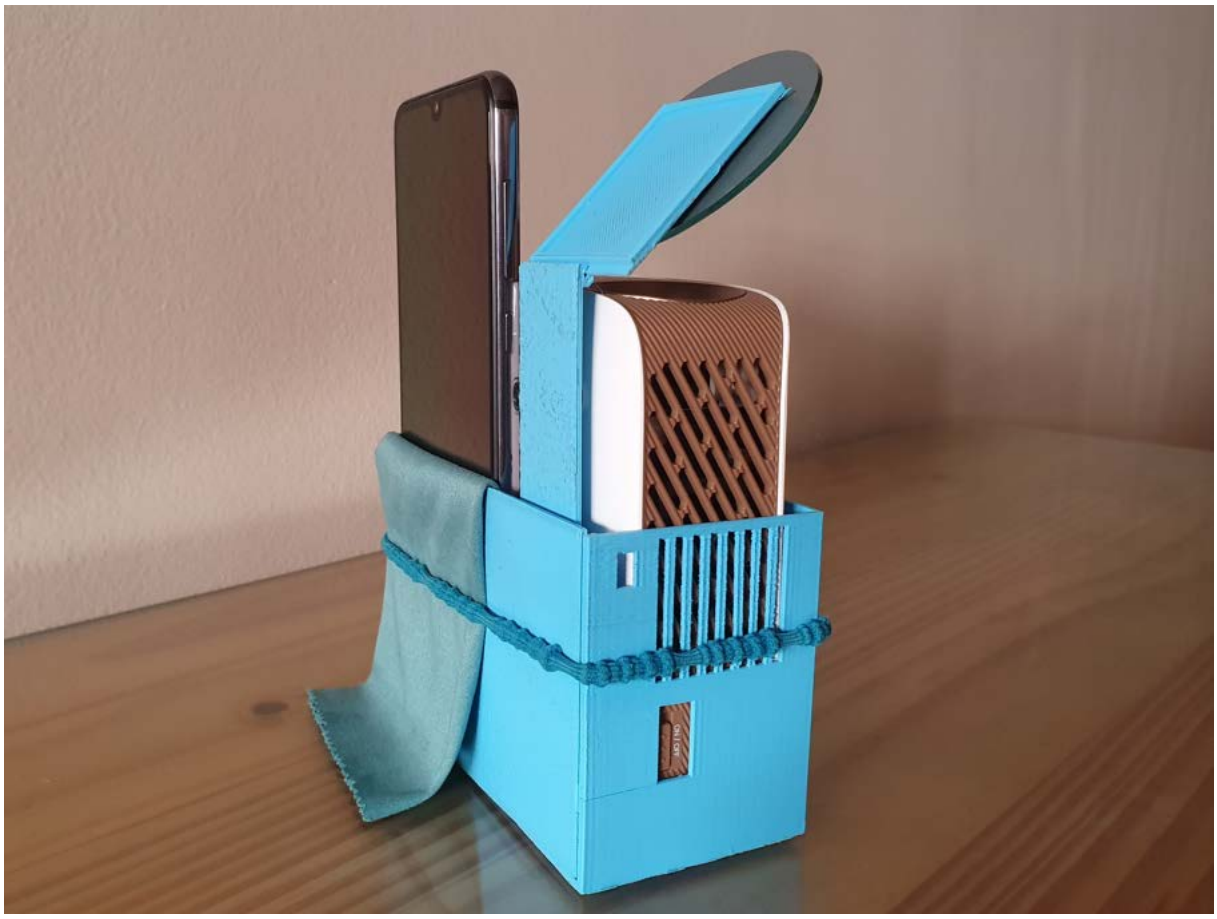


Figura A.4: Soporte físico para dispositivos hardware (parte trasera).

En la base del soporte se ha habilitado espacio para permitir el acceso a las conexiones cableadas del proyector y del smartphone (ver figura A.5), en caso de que sea necesario conectar ambos vía HDMI, en lugar de utilizar conexión inalámbrica, o conectar alguna fuente de energía para aumentar al autonomía de los dispositivos.

Figura A.5: Modelado 3D: acceso a conexiones cableadas.

La parte trasera del soporte puede retirarse para facilitar la colocación de los dispositivos en su interior. En el interior de dicha parte trasera se han añadido una serie de salientes para situar el smartphone en la posición correcta (ver figura **??**).

Detalles del diseño

El conjunto del modelo está formado por cuatro partes. Cada una se puede encajar en otra hasta formar el conjunto completo (ver figura A.6).

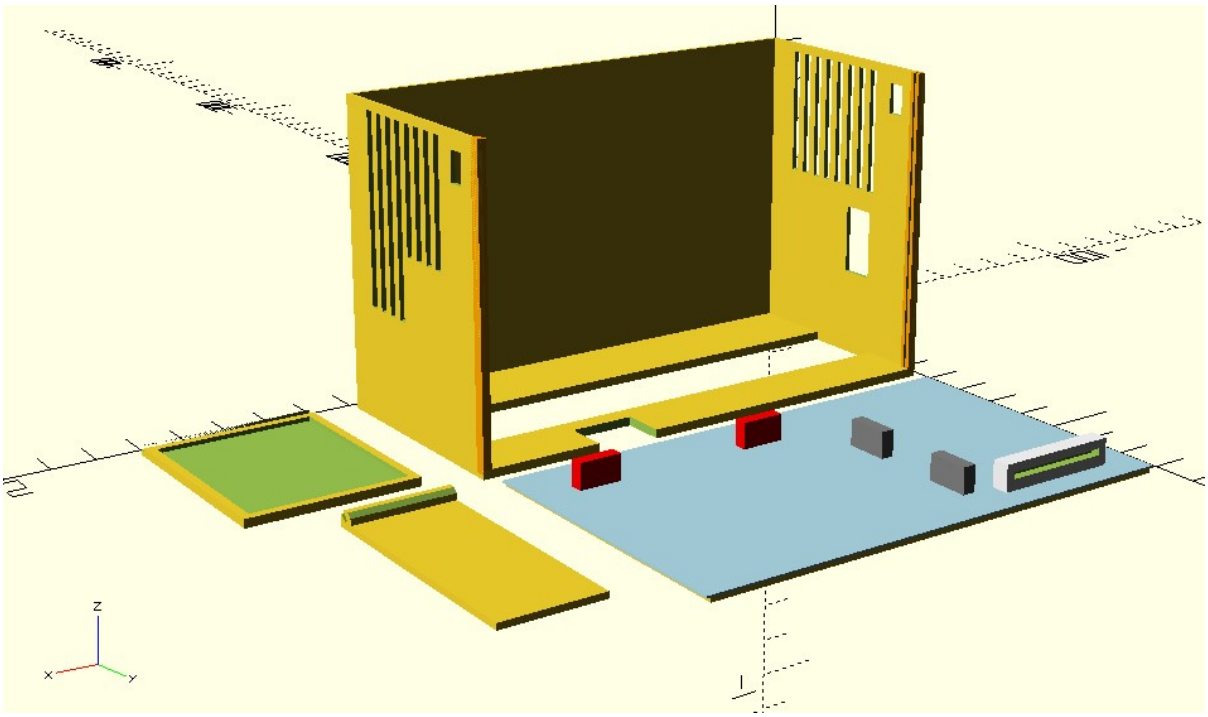
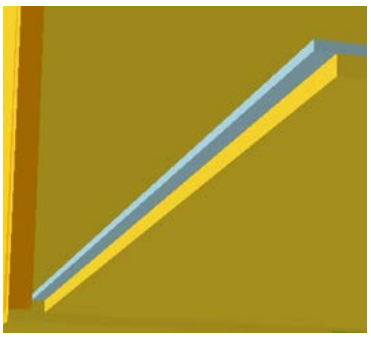


Figura A.6: Modelado 3D: modelo 1.

Como se menciona previamente, la parte trasera del soporte puede ser retirada para facilitar la colocación de los dispositivos en su interior. Para permitirlo, se han diseñado unas hendiduras en la parte principal del soporte que actúan como raíles por los que se desliza la parte trasera, la cual tiene los bordes laterales más finos (0,5mm de anchura) para que pasen por las hendiduras (ver figura A.7).

(a) Hendiduras para parte trasera.



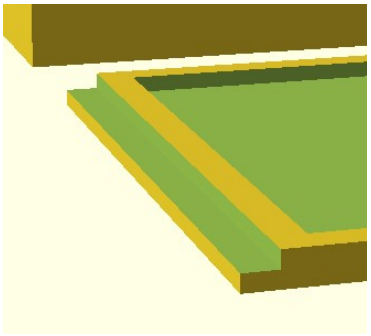
(b) Encajes de parte trasera.

Figura A.7: Modelado 3D (modelo 1): hendiduras para parte trasera y encajes de parte trasera.

La lengüeta a la que se adhiere el espejo está dividida en dos partes (ver figura A.8). La primera se encaja en la parte trasera del soporte y dispone, a su vez, de una hendidura inclinada en la que se encaja la segunda parte. La segunda tiene un borde más fino (2mm de anchura) para encajarla en la hendidura de la primera y a ella se adhiere el espejo.

El motivo por el cual se ha diseñado la lengüeta de esta forma y como pieza separada del conjunto total es permitir cambiar la altura e inclinación a la que se sitúa el espejo simplemente diseñando e imprimiendo nuevas lengüetas, más largas o cortas, con mayor o menor ángulo de inclinación de la hendidura, en lugar de tener que imprimir todo el conjunto de nuevo.

(a) Lengüeta.



(b) Soporte de espejo.

Figura A.8: Modelado 3D (modelo 1): lengüeta y soporte de espejo.

Para encajar la lengüeta en la parte trasera se ha diseñado un saliente con una hendidura (ver figura A.9).

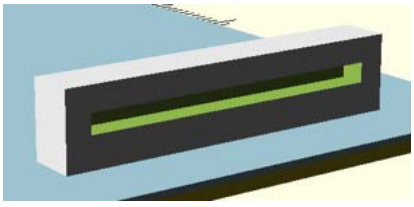


Figura A.9: Modelado 3D (modelo 1): hendidura para lengüeta.

La orientación del modelo diseñado reduce la cantidad de relleno necesario para realizar la impresión.

Resultados y conclusiones

Este soporte es el primer modelo 3D que se realiza y ha servido como aprendizaje. Tras su uso, se han identificado una serie de errores, los cuales se listan junto a su solución en la tabla A.1.

Error	Solución
El grosor de las paredes del soporte es de 1mm, excepto en la base, la cual es de 2mm de grosor .Esto provoca sensación de endeblez del soporte.	Aumentar el grosor.
Las piezas que conforman el soporte no encajan tan bien como se esperaba, debido a que la precisión requerida (0,05mm - 0,1mm) es mayor que la de la impresora 3D.	Diseñar hendiduras y encajes con mayor holgura.
El peso del espejo hace que la lengüeta tiemble cuando el usuario se mueve, aunque lo haga mínimamente.	Modelar un soporte que cubra el proyector por completo, prescindiendo de la lengüeta para poder colocar el espejo.
En el caso de que sea necesario conectar cables a los dispositivos, éstos quedan fuera del soporte, a la vista.	Modelar un soporte más grande que disponga de espacio para contener los cables.

Cuadro A.1: Errores encontrados en el primer modelo diseñado y sus soluciones.

Código OpenSCAD

```
/* Base */
translate([0,0,-1]) {
    cube([118,20,2]);

    translate([0,40,0])
        difference() {
            cube([118,14.5,2]);

            /* Hueco USB */
            translate([74.15,4.5,-0.5])
                cube([15,10.1,3]);
        }
}

/* Lado izquierdo */
difference() {
    union() {
        translate([0,0,-1])
            cube([1,54.5,72]);

        translate([0,51.9,0])
            color("orange")
                cube([2,2.6,71]);
    }

    /* Hueco On/Off */
    translate([-0.5,30,19])
        cube([2,8,16]);

    /* Respiraderos */
    translate([-0.5,9,40])
        cube([2,2,29]);

    translate([-0.5,13,40])
        cube([2,2,29]);
```

```
36
37      translate([-0.5,17,40])
38          cube([2,2,29]);
39
40      translate([-0.5,21,40])
41          cube([2,2,29]);
42
43      translate([-0.5,25,40])
44          cube([2,2,29]);
45
46      translate([-0.5,29,40])
47          cube([2,2,29]);
48
49      translate([-0.5,33,40])
50          cube([2,2,29]);
51
52      translate([-0.5,37,40])
53          cube([2,2,29]);
54
55      /* Asa */
56      translate([-0.5,44,60])
57          cube([2,4,7]);
58
59      /* Rail */
60      translate([1,52.9,1])
61          cube([2,0.6,71]);
62  }
63
64  /* Lado derecho */
65  translate([117,0,0]) {
66      difference() {
67          union() {
68              translate([0,0,-1])
69                  cube([1,54.5,72]);
70
71              translate([-1,51.9,0])
72                  color("orange")
73                      cube([2,2.6,71]);
74          }
75
76          /* Respiraderos */
77          translate([-0.5,9,25])
78              cube([2,2,44]);
79
80          translate([-0.5,13,25])
81              cube([2,2,44]);
82
83          translate([-0.5,17,25])
84              cube([2,2,44]);
85
86          translate([-0.5,21,25])
87              cube([2,2,44]);
88
89          translate([-0.5,25,40])
90              cube([2,2,29]);
91
92          translate([-0.5,29,40])
93              cube([2,2,29]);
```

```
94
95          translate([-0.5,33,40])
96              cube([2,2,29]);
97
98          translate([-0.5,37,40])
99              cube([2,2,29]);
100
101         /* Asa */
102         translate([-0.5,44,60])
103             cube([2,4,7]);
104
105         /* Rail */
106         translate([-1.1,52.9,1])
107             cube([1.1,0.6,71]);
108     }
109 }
110
111 /* Frontal */
112 cube([118,1,71]);
113
114 /* Respaldo */
115 translate([1,60,-1]) {
116     /* Fino */
117     translate([0,0,1])
118         color("lightblue")
119             cube([115.9,70,0.5]);
120
121     /* Grueso */
122     translate([1,0,0])
123         cube([113.9,70,1]);
124
125     /* Soportes smartphone */
126     translate([96,11,0.5])
127         color("red")
128             cube([10,3,7]);
129
130     translate([54.3,11,0.5])
131         color("red")
132             cube([10,3,7]);
133
134     translate([41,50,0.5])
135         color("grey")
136             cube([3,10,7]);
137
138     translate([41,25,0.5])
139         color("grey")
140             cube([3,10,7]);
141
142     /* Soporte espejo */
143     difference() {
144         translate([8,60,0.5])
145             color("white")
146                 cube([28,4,7]);
147
148         translate([9.975,62,4])
149             cube([24.05,4,1.55]);
150     }
151 }
```

```
152
153 /* Piezas espejo */
154 /* Cuadro */
155 translate([130,0,-1])
156     difference() {
157         cube([40,50,2]);
158
159         /* Encaje */
160         translate([-0.5,-0.1,1])
161             cube([41,2.1,2]);
162
163         /* Hueco espejo */
164         translate([2,4,1])
165             cube([36,44,2]);
166     }
167
168 /* Lengueta */
169 translate([130,60,-1])
170     difference() {
171         union() {
172             cube([24,58,1.5]);
173
174             cube([24,4,4]);
175         }
176
177         /* Ranura */
178         translate([-0.5,1.75,1.5])
179             rotate(a=[45,0,0])
180                 cube([25,1.05,2.05]);
181
182         /* Rebordes */
183         translate([-0.5,0,2.6])
184             rotate(a=[45,0,0])
185                 cube([25,2.5,2]);
186
187         translate([-0.5,4,3])
188             rotate(a=[45,0,0])
189                 cube([25,3,2]);
190     }
```

A.3. Segundo modelo

Con este segundo diseño se solucionan los errores listados anteriormente, aumentando la holgura de las hendiduras y los encajes y el grosor de las paredes del soporte, el cual cubre por completo el proyector y dispone de espacio para situar el cableado, en caso de ser necesario.

- El grosor medio ha sido aumentado de 1mm a 3mm, aportando al soporte una robustez mucho mayor que en el primer modelo.
- El espacio entre las hendiduras y los encajes ha sido aumentado de 0,05mm-0,1mm a 0,3mm, aumentado la holgura entre éstos.

- El soporte cubre totalmente la superficie del proyector, permitiendo situar el soporte del espejo sin necesidad de usar la lengüeta.
- Bajo el proyector y el smartphone hay espacio cubierto por el soporte para almacenar los cables que pueden colocarse en las conexiones de ambos dispositivos.

Debido a un problema externo a este proyecto con la impresora 3D, ésta dejó de estar disponible en el momento en el que se diseñó este modelo, impidiendo su impresión. Para cuando la impresora 3D volvió a estar disponible, se había decidido cambiar la disposición de los dispositivos hardware, haciendo innecesario el soporte.

Detalles del diseño

De nuevo, el modelo está formado por varias partes. En este caso, son tres, debido a que el soporte del espejo está formado por una única pieza.

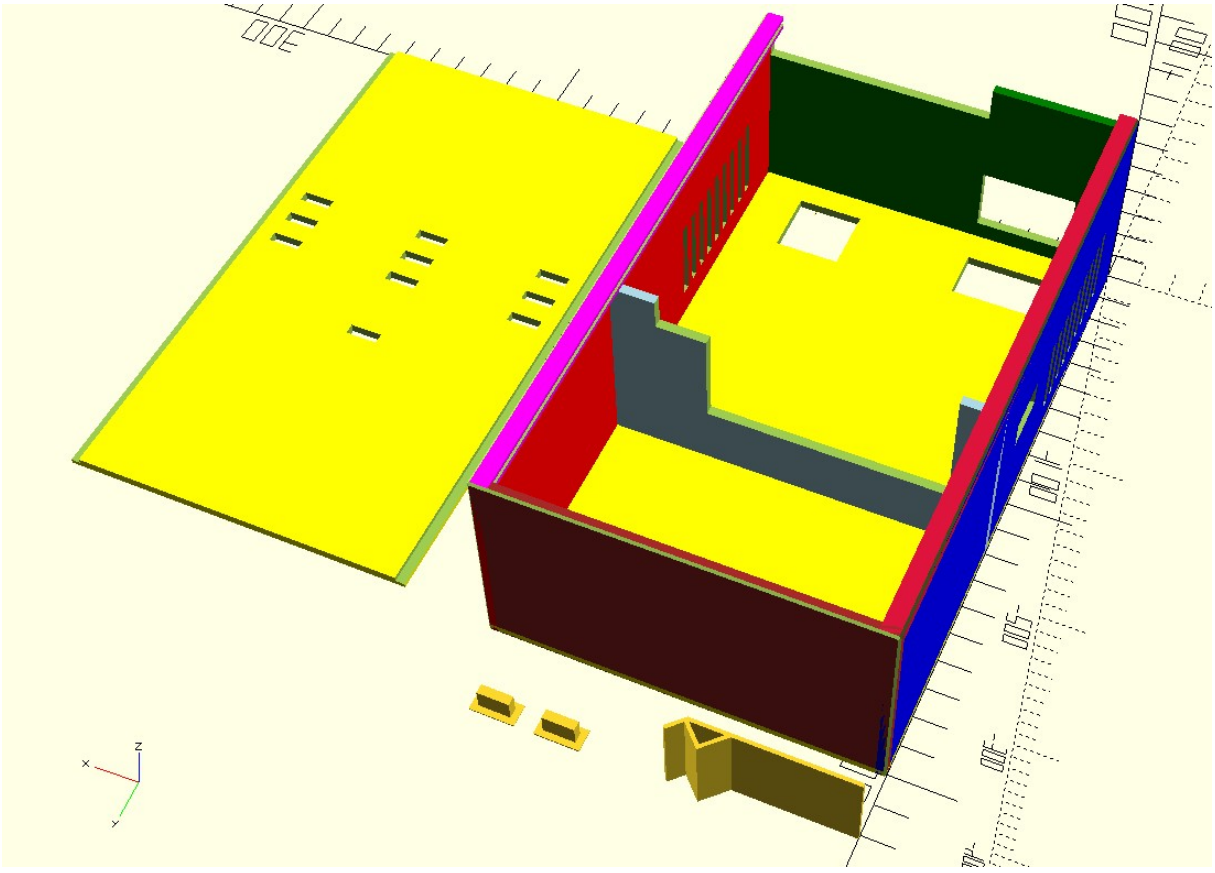
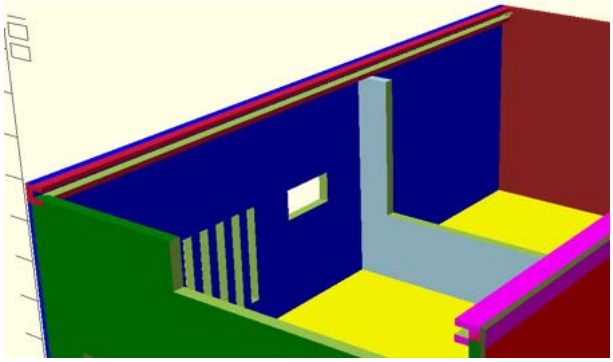
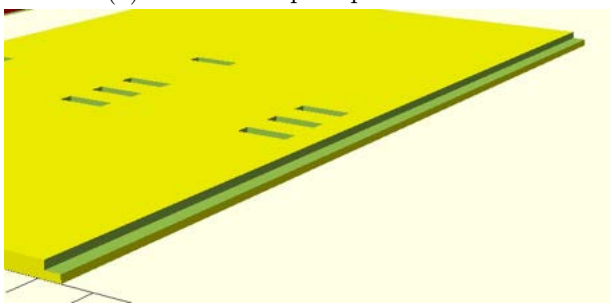


Figura A.10: Modelado 3D: modelo 2.

La forma de encajar la parte trasera vuelve a ser mediante hendiduras que actúan como raíles en la parte principal por las que se deslizan los bordes más finos de la parte trasera (ver figura A.11).



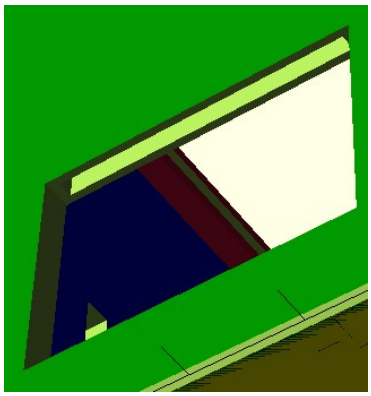
(a) Hendiduras para parte trasera.



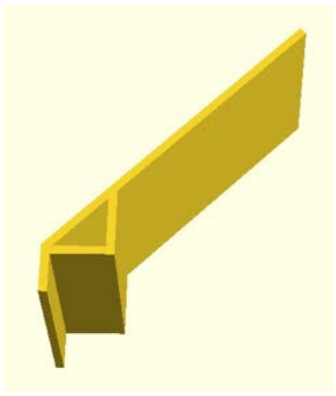
(b) Encajes de parte trasera.

Figura A.11: Modelado 3D (modelo 2): hendiduras para parte trasera y encajes de parte trasera.

El nuevo soporte para el espejo se encaja en una hendidura situada en el hueco para el foco del proyector. El soporte cuenta con un refuerzo que aumenta su estabilidad (ver figura A.12).



(a) Hendidura para soporte de espejo.

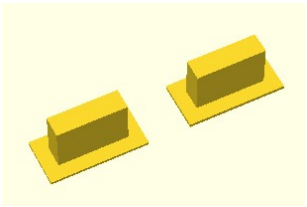


(b) Soporte de espejo.

Figura A.12: Modelado 3D (modelo 2): hendidura para soporte de espejo y soporte de espejo.

Para permitir poner el smartphone en distintas alturas, se han colocado una serie huecos en la parte trasera del soporte (ver figura A.13). En ellos se insertan unos topes sobre los que reposa el smartphone.

(a) Huecos en la parte trasera.



(b) Topes para reposar el smartphone.

Figura A.13: Modelado 3D (modelo 2): huecos en la parte trasera y topes.

Para estilizar un poco el soporte, se han perfilado todos los bordes de la parte principal (ver figura A.14).

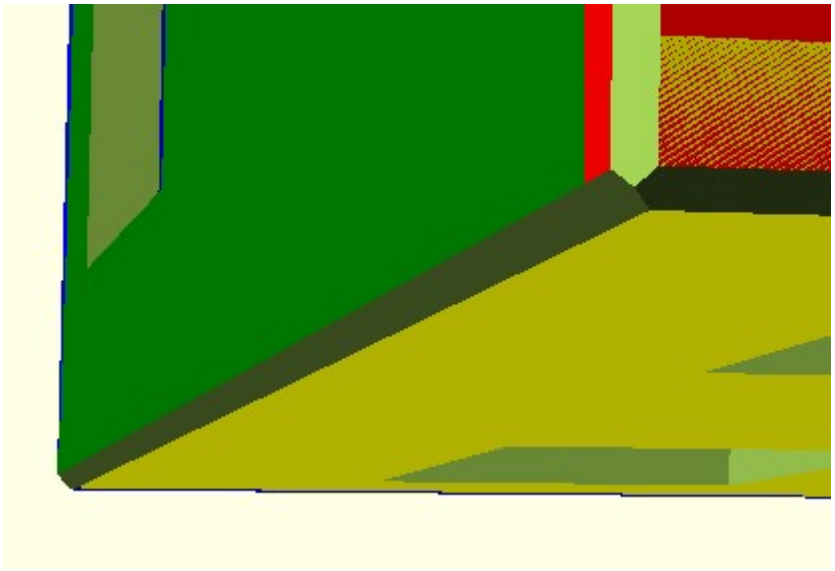


Figura A.14: Modelado 3D (modelo 2): perfilado del soporte.

Código OpenSCAD

```
/* Principal */
difference() {
    union() {
        /* Frontal */
        translate([0,0,0])
            difference() {
                color("yellow")
                    cube([121.2,201.2,3]);

                /* Focus */
                translate([13,13,-0.5])
                    cube([30.5,16,4]);

                /* Control */
                translate([82,11,-0.5])
                    cube([21,21,4]);
            }
```

```
18
19          /* Top */
20          translate([0,0,0])
21              difference() {
22                  color("green")
23                      cube([121.2,3,66.4]);
24
25                  /* Lente */
26                  translate([14,-0.5,9])
27                      cube([30.6,4,22]);
28
29                  /* Ranura espejo */
30                  translate([15,0.75,30.5])
31                      cube([28.6,1.5,5.5]);
32
33                  /* Espacio smartphone */
34                  translate([45,-0.5,55])
35                      cube([73.2,4,12]);
36              }
37
38          /* Base proyector */
39          translate([0,118.2,0])
40              difference() {
41                  color("lightblue")
42                      cube([121.2,3,64.9]);
43
44                  /* Conexiones proyector */
45                  translate([13,-0.5,23.4])
46                      cube([75.2,4,42]);
47
48                  /* Hueco USB */
49                  translate([88,-0.5,55])
50                      cube([16,4,12]);
51              }
52
53          /* Base */
54          translate([0,198.2,0])
55              color("brown")
56                  cube([121.2,3,69.4]);
57
58          /* Lado izquierdo */
59          translate([0,0,0])
60              difference() {
61                  color("blue")
62                      cube([3,201.2,69.4]);
63
64                  /* Respiraderos */
65                  translate([-0.5,20,12])
66                      cube([4,3,28]);
67
68                  translate([-0.5,26,12])
69                      cube([4,3,28]);
70
71                  translate([-0.5,32,12])
72                      cube([4,3,28]);
73
74                  translate([-0.5,38,12])
75                      cube([4,3,28]);
```

```
76
77                  translate([-0.5,44,12])
78                      cube([4,3,28]);
79
80                  translate([-0.5,50,12])
81                      cube([4,3,28]);
82
83                  translate([-0.5,56,12])
84                      cube([4,3,28]);
85
86                  translate([-0.5,62,12])
87                      cube([4,3,28]);
88
89                  translate([-0.5,68,12])
90                      cube([4,3,28]);
91
92                  /* Hueco On/Off */
93                  translate([-0.5,85,31])
94                      cube([4,17.5,9]);
95              }
96
97          /* Lado derecho */
98          translate([118.2,0,0])
99              difference() {
100                 color("red")
101                     cube([3,201.2,69.4]);
102
103                 /* Respiraderos */
104                 translate([-0.5,20,12])
105                     cube([4,3,28]);
106
107                 translate([-0.5,26,12])
108                     cube([4,3,28]);
109
110                 translate([-0.5,32,12])
111                     cube([4,3,28]);
112
113                 translate([-0.5,38,12])
114                     cube([4,3,28]);
115
116                 translate([-0.5,44,12])
117                     cube([4,3,28]);
118
119                 translate([-0.5,50,12])
120                     cube([4,3,28]);
121
122                 translate([-0.5,56,12])
123                     cube([4,3,28]);
124
125                 translate([-0.5,62,12])
126                     cube([4,3,28]);
127
128                 translate([-0.5,68,12])
129                     cube([4,3,28]);
130             }
131
132         /* Ranura izquierda */
133         translate([0,0,64.9])
```

```
            difference() {
                color("crimson")
                    cube([6,201.2,4.5]);

                translate([3,-0.5,1.5])
                    cube([4,198.7,1.75]);
            }

        /* Ranura derecha */
        translate([115.2,0,64.9])
            difference() {
                color("magenta")
                    cube([6,201.2,4.5]);

                translate([-1,-0.5,1.5])
                    cube([4,198.7,1.75]);
            }
    }

    /* Bordes */
    /* Inferior izquierda */
    translate([-0.71,-0.5,0])
        rotate(a=[0,45,0])
            cube([1,300,1]);

    /* Inferior derecha */
    translate([121.2,0,0])
        translate([-0.71,-0.5,0])
            rotate(a=[0,45,0])
                cube([1,300,1]);

    /* Superior izquierda */
    translate([0,0,69.4])
        translate([-0.71,-0.5,0])
            rotate(a=[0,45,0])
                cube([1,300,1]);

    /* Inferior derecha */
    translate([121.2,0,69.4])
        translate([-0.71,-0.5,0])
            rotate(a=[0,45,0])
                cube([1,300,1]);

    /* Inferior delantero */
    translate([-1,0,-0.71])
        rotate(a=[45,0,0])
            cube([124,1,1]);

    /* Inferior trasero */
    translate([0,201.2,0])
        translate([-1,0,-0.71])
            rotate(a=[45,0,0])
                cube([124,1,1]);

    /* Inferior trasero */
    translate([0,201.2,69.4])
        translate([-1,0,-0.71])
            rotate(a=[45,0,0])
```

```
                cube([124,1,1]);

    /* Delantero izquierdo */
    translate([0,-0.71,0])
        rotate(a=[0,0,45])
            cube([1,1,72]);

    /* Delantero derecho */
    translate([121.2,0,0])
        translate([0,-0.71,0])
            rotate(a=[0,0,45])
                cube([1,1,72]);

    /* Trasero izquierdo */
    translate([0,201.2,0])
        translate([0,-0.71,0])
            rotate(a=[0,0,45])
                cube([1,1,72]);

    /* Trasero derecho */
    translate([121.2,201.2,0])
        translate([0,-0.71,0])
            rotate(a=[0,0,45])
                cube([1,1,72]);
}

/* Tapa */
translate([150,0,0])
    difference() {
        color("yellow")
            cube([114.2,198,3]);

        /* Carril izquierdo */
        translate([-0.5,-0.5,1.45])
            cube([3.5,200,2]);

        /* Carril derecho */
        translate([111.2,-0.5,1.45])
            cube([3.5,200,2]);

        /* Hueco superior izquierdo */
        translate([8,73.2,-0.5])
            cube([10.5,3.5,4]);

        /* Hueco superior central */
        translate([52,73.2,-0.5])
            cube([10.5,3.5,4]);

        /* Hueco superior derecho */
        translate([95,73.2,-0.5])
            cube([10.5,3.5,4]);

        /* Hueco medio izquierdo */
        translate([8,83.2,-0.5])
            cube([10.5,3.5,4]);

        /* Hueco medio central */
        translate([52,83.2,-0.5])
```

```
            cube([10.5,3.5,4]);

        /* Hueco medio derecho */
        translate([95,83.2,-0.5])
            cube([10.5,3.5,4]);

        /* Hueco inferior izquierdo */
        translate([8,93.2,-0.5])
            cube([10.5,3.5,4]);

        /* Hueco inferior central */
        translate([52,93.2,-0.5])
            cube([10.5,3.5,4]);

        /* Hueco inferior derecho */
        translate([95,93.2,-0.5])
            cube([10.5,3.5,4]);

        /* Hueco extra */
        translate([52,118.2,-0.5])
            cube([10.5,3.5,4]);
    }

/* Soporte espejo */
translate([0,220,0]) {
    cube([50,1.2,28]);

    translate([50,0,0])
        rotate(a=[0,0,45])
            cube([7,1.2,28]);

    translate([51,1,0])
        rotate(a=[0,0,135])
            cube([10,1.2,28]);

    translate([37.555,0,0])
        rotate(a=[0,0,45])
            cube([10,1.2,28]);
}

/* Topes */
translate([80,220,0]) {
    cube([14,7,0.5]);

    translate([2,2,0])
        cube([10,3,7]);

    translate([20,0,0]) {
        cube([14,7,0.4]);

        translate([2,2,0])
            cube([10,3,7]);
    }
}
```

A.4. Impresión

Terminado el diseño, *OpenSCAD* compila el código y genera un archivo *.stl* (de las siglas en inglés *STereoLithography*).

De este archivo hace uso el último programa que se ha utilizado para la impresión 3D: ***Ultimaker Cura***. Este programa contiene una base de datos de impresoras 3D y múltiples opciones de modificación de los parámetros de impresión (ver figura A.15). En él, se selecciona la impresora 3D y los parámetros de impresión que se consideren oportunos. En este caso, grosor de cada capa impresa (por altura) y calidad de la base de impresión.

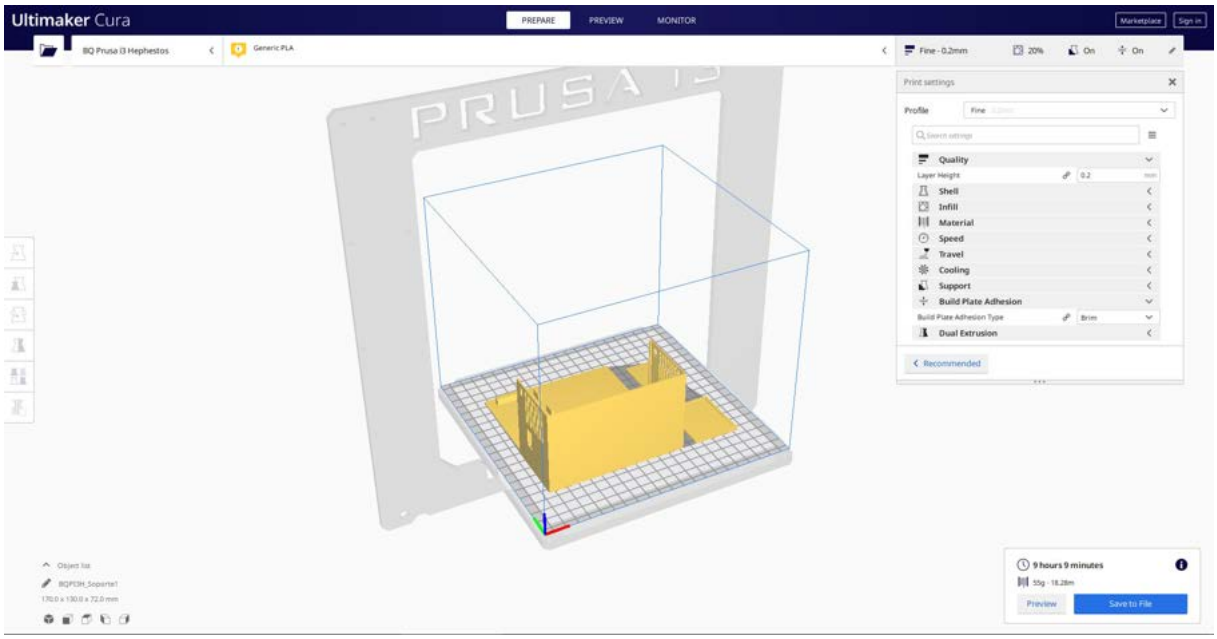


Figura A.15: Ejemplo de uso de Ultimaker Cura.

Ultimaker Cura genera un archivo *.gcode* que utilizará la impresora 3D para imprimir el modelo. También realiza una estimación del tiempo que empleará la impresora 3D en imprimir el modelo.

Apéndice B

Manual de instalación

A continuación, se detallan los requisitos previos y el procedimiento de instalación del software.

B.1. Inventario de componentes

El conjunto de componentes que forman el sistema está formado por:

- Dispositivo móvil con sistema operativo Android 4.1 ó superior.
- Proyector portátil.
- Soportes o accesorios que permitan portar los anteriores dispositivos en el torso.
- Espejo pequeño (sólo en caso de que el foco del proyector portátil no pueda apuntar hacia el frente del usuario mientras lo porta en el torso).

B.2. Requisitos previos

- Esta *release* del software funciona en un dispositivo con sistema operativo Android 4.1 ó superior. Comprueba la versión de Android de tu dispositivo antes de intentar instalar el software. Para ello, ve a **Ajustes** en tu dispositivo y escribe "**android**" en el buscador. Te aparecerá una opción para comprobar la versión de Android.
- Si vas a instalar la aplicación vía ***APK***, antes tendrás que habilitar esta opción. La forma de hacerlo varía entre versiones de Android y dispositivos.
- Si vas a conectar el dispositivo móvil y el proyector de forma inalámbrica, necesitarás conexión ***Wifi*** y ***Bluetooth***.

B.3. Procedimientos de instalación

Para instalar la aplicación, selecciona el **archivo *APK*** y acepta la instalación. Si todo va bien, la aplicación se instalará en tu dispositivo con el nombre "**SixtSense**".

Tendrás que darle a la aplicación permisos para utilizar la cámara de tu dispositivo. Para ello, ve a **Ajustes** en tu dispositivo y busca "**permisos**" en el buscador. Te aparecerá una opción para darle permisos a las aplicaciones. Deberás buscar los permisos de la cámara y activar la aplicación **SixthSense** o buscar la aplicación **SixthSense** y darle los permisos de la cámara.

Apéndice C

Manual de usuario

En este capítulo se detallan las instrucciones de uso del sistema.

C.1. Colocación y conexión del hardware

Los dispositivos hardware se colocan sobre el torso. Necesitarás el soporte diseñado para ello o algún otro tipo de soporte que te permita portarlos.

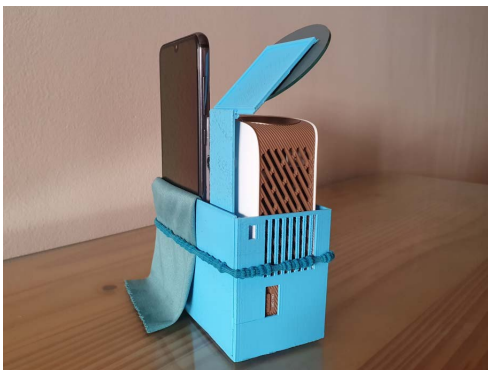
1. Si dispones del soporte, coloca el proyector en su interior con el foco apuntando hacia arriba y la base hacia la parte trasera del soporte.

2. Si vas a conectar el proyector y el smartphone con un cable HDMI, conéctalo ahora a ambos dispositivos.

3. La tapa trasera desmontable del soporte tiene unos salientes pequeños sobre los que puedes colocar el smartphone, con la cámara apuntando al frente y la pantalla hacia atrás, y un saliente alargado con una hendidura en el que colocar el soporte del espejo.

   En la figura C.1 se muestra el resultado:

(a) Parte frontal.



(b) Parte trasera.

Figura C.1: Manual de usuario: uso de soporte para dispositivos hardware.

4. Enciende el proyector y selecciona el método de conexión que vas a usar: HDMI o *ScreenShare*.
5. Para la segunda, activa la funcionalidad *ScreenShare* en el smartphone. Tendrás que estar conectado a una red wifi y tener activado el *bluetooth*.
6. Se buscarán los dispositivos disponibles. Si aparece el identificador del proyector, selecciónalo y espera a que se realice la conexión. Si no reconoce el dispositivo o la conexión falla, tendrás que repetir el proceso.

Con esto, proyector y smartphone estarán conectados y podrás ver proyectada la pantalla del smartphone mediante el proyector.

C.2. Uso del sistema

Calibración

Índice mano izquierda - Marcador azul
Índice mano derecha - Marcador rojo

Cuando inicies la aplicación, tras el logo de *Unity*, verás la escena de calibración (ver figura C.2), compuesta por dos cuadrados que cambian de color.

Figura C.2: Manual de usuario: escena de calibración.

Deberás señalar cada uno de ellos con el dedo correspondiente al color inicial de éstos (por defecto, azul el izquierdo y rojo el derecho).

Pasados unos segundos, cambiará de color. El color verde indica que es ahora cuando el sistema está capturando la posición que señalas. Si determina que la captura no se ha realizado adecuadamente, se repetirá el proceso.

Una vez realizada con éxito la calibración, podrás controlar dos punteros de colores (rojo y azul) con el movimiento de tus dedos frente a la cámara. Con estos punteros podrás interactuar con los distintos elementos de la interfaz que se te mostrarán.

Menú principal

La escena siguiente a la calibración es el menú principal (ver figura C.3) desde el cual se puede acceder a los juegos, cerrar la aplicación o cambiar el tamaño de la interfaz.

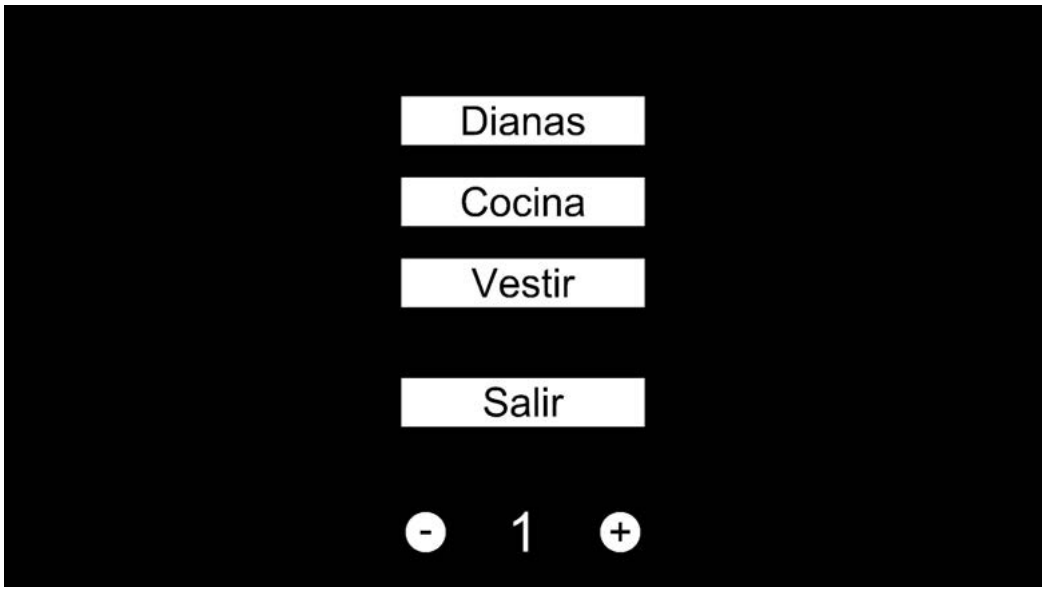


Figura C.3: Manual de usuario: menú principal.

Tan sólo tienes que seleccionar con alguno de los punteros los botones disponibles durante unos segundos para acceder a su función. Cuando selecciones un botón, éste se agrandará para indicarte que lo estás seleccionando correctamente.

Juegos

Al entrar en un juego, verás una pantalla tutorial en la que se explica qué tienes que hacer (ver figura C.4). Para pasar al juego, selecciona el botón **Cerrar**.

Figura C.4: Manual de usuario: tutorial de juego.

Cuando acabes un juego, se mostrará una pantalla con el tiempo que has empleado en completar el juego y dos botones: uno para volver al menú principal y otro para repetir ese juego (ver figura C.5).

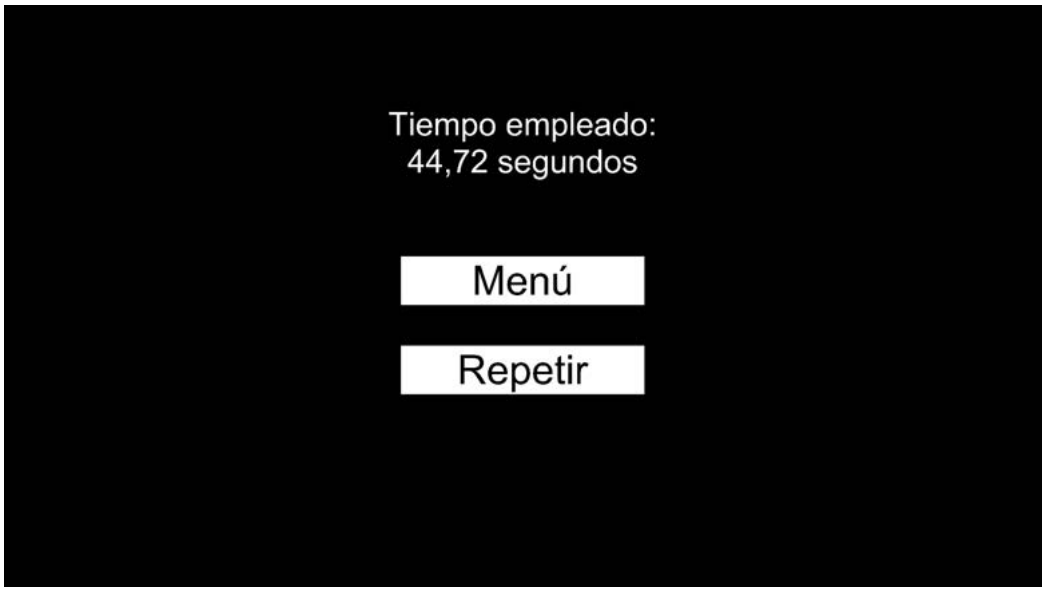


Figura C.5: Manual de usuario: tiempo empleado en terminar juego.

Juegos - Dianas

Índice mano izquierda - Marcador azul
Índice mano derecha - Marcador rojo

En este juego aparecen, una a una, dianas de colores que tendrás que seleccionar con el puntero correspondiente al color de la diana: mano izquierda para las dianas azules y mano derecha para las dianas rojas (ver figura C.6). Cuando selecciones correctamente una, ésta desaparecerá y aparecerá la siguiente.

(a) Diana azul.

(b) Diana roja.

Figura C.6: Manual de usuario: prueba de dianas.

Juegos - Utensilios de cocina

En este juego verás una cocina con distintos utensilios repartidos (ver figura C.7). En la parte superior, verás un cuadro amarillo en el que se indica el utensilio que debes seleccionar.

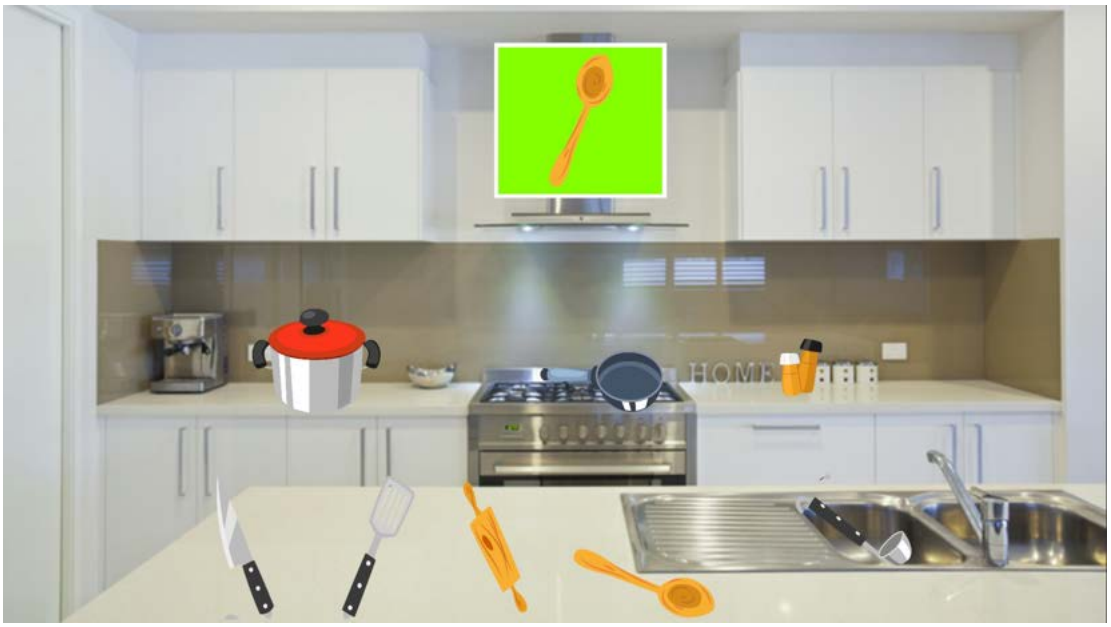


Figura C.7: Manual de usuario: prueba de utensilios de cocina.

Juegos - Vestir al muñeco

Índice mano izquierda - Marcador azul
Índice mano derecha - Marcador rojo

En este juego aparecen varias prendas de ropa con las que tendrás que vestir a un muñeco (ver figura C.8). Para ello, selecciónalas durante unos segundos y podrás moverlas. Sitúalas unos segundos sobre la parte del muñeco a la que correspondan y se quedarán fijadas.

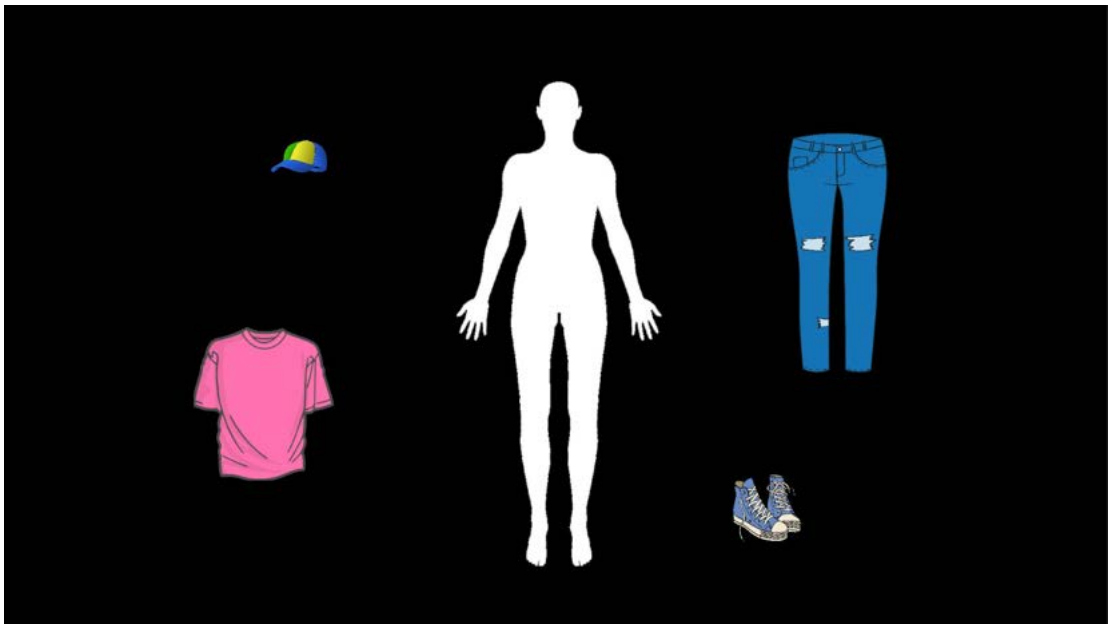


Figura C.8: Manual de usuario: prueba de vestir al muñeco.

Apéndice D

Manual de desarrollador

En este manual se mostrará cómo manipular el proyecto de *Unity* para modificar la aplicación. Para más información sobre cómo desarrollar en *Unity*, consultar la web oficial de aprendizaje [13]. También hay disponibles libros de aprendizaje [19] que se incluyen en la bibliografía de este documento.

D.1. Añadir colores al detector

En cada escena en la que se utiliza la detección verás, en la ventana **Hierarchy** del editor (ver figura D.1), un *GameObject* correspondiente al detector de colores llamado **ColorDetector** y un conjunto de *GameObjects* llamado **Colors** que contiene los colores que se detectan.

Figura D.1: Manual de desarrollador: GameObjets de la detección de colores.

Para añadir colores a la detección, copia y pega el *GameObject* uno de los ya existentes y cambia los valores del script **ColorPosition** en la ventana **Inspector** (ver figura D.2) del nuevo *GameObject*.

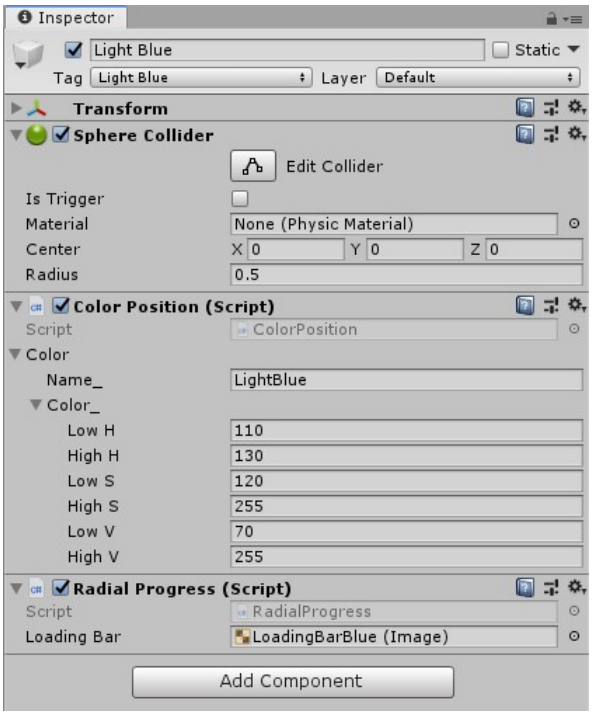


Figura D.2: Manual de desarrollador: ventana Inspector de un color (1).

En su interior hay otro *GameObject* llamado **CircularProgressBar**, el cual contiene los *GameObjects* que generan la animación de carga del puntero. El *GameObject* **Center** es el que representa el color del puntero. Cambia el color en el componente *Image* en la ventana **Inspector** (ver figura D.3) para cambiar el color del puntero.

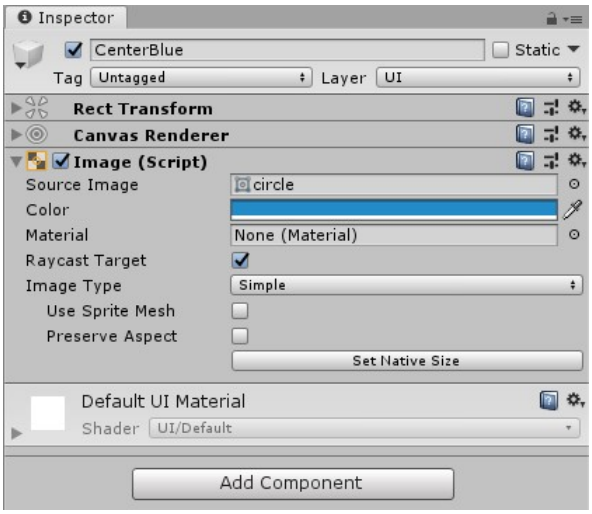


Figura D.3: Manual de desarrollador: ventana Inspector de un color (2).

Finalmente, añade el *GameObject* del nuevo color al detector de color (ver figura D.4).

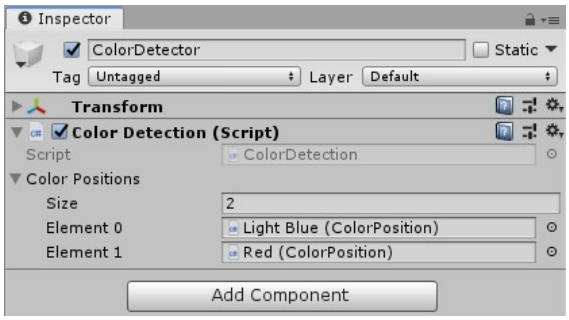


Figura D.4: Manual de desarrollador: ventana Inspector del detector de color.

Cada escena tiene su detector y colores propios, por los que tendrás que realizar estos pasos en cada escena en la que quieras añadir nuevos colores.

D.2. Interaccionar con objetos

Para interaccionar con los objetos de las escenas, los *GameObjects* de los colores deberán tener un **Collider**, como se muestra en la sección anterior, y los *GameObjects* de los objetos con los que interaccionan deberán tener **Collider** y **RigidBody**. Una vez añadidos, activa la opción **trigger** del **Collider** en la ventana **Inspector** del *GameObject* del objeto (ver figura D.5).

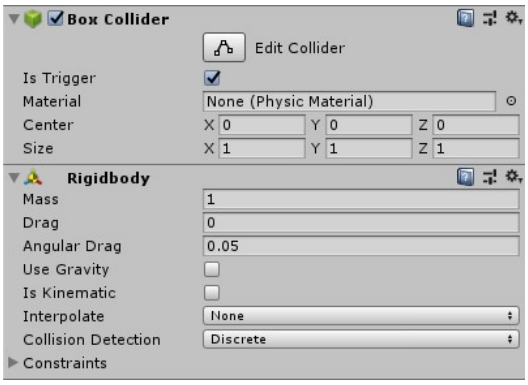


Figura D.5: Manual de desarrollador: ventana Inspector de objeto que interacciona.

Bibliografía

[1] Algoritmo de detección de bordes de Canny. (Octubre de 2019). https://docs.opencv.org/trunk/da/d22/tutorial_py_canny.html.

[2] API de la clase Barrier. (Octubre de 2019). https://docs.microsoft.com/es-es/dotnet/api/system.threading.barrier?view=netcore-3.1.

[3] API de la clase Thread. (Octubre de 2019). https://docs.microsoft.com/es-es/dotnet/api/system.threading.thread?view=netcore-3.1.

[4] Enfermedad de Parkinson - Mayo Clinic. (Septiembre de 2020). https://www.mayoclinic.org/es-es/diseases-conditions/parkinsons-disease/symptoms-causes/syc-20376055#:~:text=La%20enfermedad%20de%20Parkinson%20es,rigidez%20o%20disminuci%C3%B3n%20del%20movimiento.

[5] Espacio de nombres InteropServices de Microsoft .NET. (Octubre de 2019). https://docs.microsoft.com/es-es/dotnet/api/system.runtime.interopservices?view=netcore-3.1.

[6] How dangerous are cranes really? - Health Safety Training. (Agosto de 2020). https://www.hst.uk.com/news/how-dangerous-are-cranes-really/.

[7] Industry 4.0: The Future of Productivity and Growth in Manufacturing Industries. (Agosto de 2020). https://www.bcg.com/publications/2015/engineered_products_project_business_industry_4_future_productivity_growth_manufacturing_industries.

[8] Leap Motion / Developer. (Abril de 2020). https://developer.leapmotion.com/.

[9] OpenCV. Documentación web. (Octubre de 2019). https://opencv.org/about.

[10] OpenSCAD - The Programmers Solid 3D CAD Modeller. (Junio de 2019). https://www.openscad.org/index.html.

[11] Plataforma de desarrollo en tiempo real de Unity | Visualizaciones de VR y AR en 3D y 2D. (Octubre de 2019). https://unity.com/es.

[12] Principio de Pareto. (Abril de 2020). https://es.wikipedia.org/wiki/Principio_de_Pareto.

[13] Unity Learn website. (Octubre de 2019). https://learn.unity.com/.

[14] VirtualRehab - Evolv. (Noviembre de 2020). https://evolvrehab.com/es/virtualrehab/virtualrehab_body/.

[15] ¿Qué es el Alzheimer? - Alzheimer's Association. (Septiembre de 2020). https://www.alz.org/alzheimer-demencia/que-es-la-enfermedad-de-alzheimer?lang=es-MX.

[16] Aris, Aleks. *Ubiquitous Computing*. 2003. http://www.cs.umd.edu/grad/scholarlypapers/papers/AleksAris_UbiquitousComputing.pdf.

[17] Bimber, Oliver y Raskar, Ramesh. *Spatial Augmented Reality: Merging Real and Virtual Worlds*. CRC Press, 2005.

[18] González, Juan. Diseño de piezas con OpenSCAD - WikiRobotics. (Junio de 2019). http://www.iearobotics.com/wiki/index.php?title=Dise%C3%B1o_de_piezas_con_OpenScad#Ficha, year = 2011.

[19] Joe Hocking. *Unity in Action: Multiplatform game development in C*. Manning, 2018.

[20] Kanel, Kedar. *Sixth Sense Technology*. 2014. https://www.theseus.fi/bitstream/handle/10024/87120/final%20thesis_1_kedar.pdf.

[21] Mistry, Pranav. SixthSense. Integrating information with the real world. (Noviembre de 2020). http://www.pranavmistry.com/archived/projects/sixthsense.

[22] Mousavi Hondori, Hossein; Khademi, Maryam; Dodakian, Lucy; Cramer, Steven C.; Lopes, Cristina Videira. *A Spatial Augmented Reality Rehab System for Post-Stroke Hand Rehabilitation*. IOS Press. 2013. https://www.researchgate.net/profile/Hossein_Mousavi_Hondori/publication/235522337_A_Spatial_Augmented_Reality_rehab_system_for_post-stroke_hand_rehabilitation/links/0c9605177970643f38000000/A-Spatial-Augmented-Reality-rehab-system-for-post-stroke-hand-rehabilitation.pdf.

[23] Sutherland, Ivan E. *The Ultimate Display*. Proceedings of IFIP Congress. pp. 506–508. 1965. http://citeseer.ist.psu.edu/viewdoc/download;jsessionid=8C1253D47A22178EF18F28B3C667543C?doi=10.1.1.136.3720&rep=rep1&type=pdf.

GNU Free Documentation License

Version 1.3, 3 November 2008

<http://fsf.org/>

Preamble

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.
This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.
We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "**Document**", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "**you**". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.
A "**Modified Version**" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "**Secondary Section**" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "**Invariant Sections**" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "**Cover Texts**" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "**Transparent**" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "**Opaque**".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "**Title Page**" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

The "**publisher**" means any person or entity that distributes copies of the Document to the public.

A section "**Entitled XYZ**" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "**Acknowledgements**", "**Dedications**", "**Endorsements**", or "**History**".) To "**Preserve the Title**" of such a section when you modify the Document means that it

remains a section "Entitled XYZ" according to this definition.
The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or non-commercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.
You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.
If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.
If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.
It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections

Entitled “Acknowledgements”, and any sections Entitled “Dedications”. You must delete all sections Entitled “Endorsements”.

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.
You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an “aggregate” if the copyright resulting from the compilation is not used to limit the legal rights of the compilation’s users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.
If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document’s Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.
If a section in the Document is Entitled “Acknowledgements”, “Dedications”, or “History”, the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.
Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.
Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.
Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING

"Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A "Massive Multiauthor Collaboration" (or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.
"CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.
"Incorporate" means to publish or republish a Document, in whole or in part, as part of another Document.
An MMC is "eligible for relicensing" if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently

incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.
The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.