**ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA**

**GRADO EN DISEÑO Y DESARROLLO DE VIDEOJUEGOS**

Curso Académico 2023/2024

Trabajo Fin de Grado

# DESARROLLO DE SISTEMA DE INTERACCIÓN PARA JUEGOS EDUCATIVOS EN ENTORNOS DE BAJO PRESUPUESTO

**Autor**: David López López

**Tutor**: Aarón Sújar Garrido

# Agradecimientos

A mi familia y amigos por su apoyo de inicio a fin de este proyecto.

# Resumen

Este proyecto fue desarrollado por David López López con el objetivo de crear una librería de código abierto que permita el desarrollo de videojuegos o experiencias inmersivas con la utilización de un proyector y una cámara. La librería está diseñada para ser totalmente accesible, eliminando la dependencia de *hardware* costoso y especializado.

Actualmente en la industria existen muchos tipos de experiencias inmersivas y videojuegos que utilizan *hardware* no convencional para crear interacciones fuera de lo común. Cada vez es más común ver museos que tienen exposiciones que utilizan proyectores o pantallas para impulsar el uso de la tecnología en un ámbito cultural, expandiendo el medio y consiguiendo experiencias imposibles de conseguir con medios convencionales.

Por otro lado, en la industria del videojuego siempre ha habido ejemplos de innovación de la forma de jugar e interactuar con la consola, evitando por todos los medios el mando convencional con el uso del cuerpo, los pies, los ojos o incluso la voz. Este campo en el entretenimiento lleva unos años perdiendo fuerza, hasta el punto que en muchos casos estos aparatos han ido desapareciendo o cuentan con un precio inaccesible para la gran mayoría de público.

El objetivo de este proyecto es democratizar la tecnología inmersiva, haciéndola disponible para instituciones con recursos limitados, como colegios y museos. Al proporcionar una herramienta gratuita y versátil, se abre un mundo de posibilidades para la educación y el entretenimiento, permitiendo a los desarrolladores a sumergirse en la creación entornos virtuales sin la necesidad de pantallas tradicionales ni un *hardware* limitante ni específico.

El principal problema que se busca solventar se encuentra en la dificultad de iniciarse en el desarrollo de este tipo de experiencias y videojuegos por la falta de medios, por lo que una librería abierta y gratuita que permitiese la creación y desarrollo de experiencias con el menor número de limitaciones posible puede suponer un gran cambio en la industria.

Este proyecto no solo busca innovar en campo tecnológico, sino también aportar y facilitar un acceso a herramientas que hasta ahora estaban reservadas para organizaciones con grandes recursos, un paso adelante hacia una tecnología inmersiva como herramienta común en la educación y en la cultura.

Finalmente, el desarrollo de esta librería se completa con la publicación en la *Asset Store de Unity* haciendo que se pueda descargar desde cualquier parte del mundo y aplicar a cualquier proyecto del motor de videojuegos.

Palabras clave:

- Unity
- C#
- OpenCV
- Accesibilidad
- Educación y entretenimiento
- Inclusión Tecnológica

# Contenido


**1. INTRODUCCIÓN ..... 12**

1.1. CONTEXTO Y ALCANCE ..... 12

*1.1.1. Aportaciones ..... 12*

*1.1.2. Impacto potencial ..... 12*

1.2. CARACTERÍSTICAS DEL PROYECTO ..... 13

**2. OBJETIVOS ..... 16**

2.1. DESCRIPCIÓN DEL PROBLEMA ..... 16

2.2. ESTUDIO DEL PROBLEMA ..... 16

*2.2.1. Estado del arte ..... 17*

*2.2.2. Elecciones para la solución ..... 20*

*2.2.3. Problemas de Hardware ..... 20*

2.3. METODOLOGÍA APLICADA ..... 22

**3. DESCRIPCIÓN INFORMÁTICA ..... 25**

3.1. ESQUEMA DE HARDWARE ..... 25

3.2. DISEÑO ..... 27

*3.2.1. Evolución del diseño ..... 27*

*3.2.2. Diseño final ..... 29*

3.3. IMPLEMENTACIÓN ..... 31

*3.3.1. Esquema UML ..... 31*

*3.3.2. Escalabilidad ..... 32*

3.4. CLASES ..... 32

*3.4.1. WebCam ..... 32*

*3.4.2. Calibration ..... 32*

*3.4.3. ContourFinder ..... 33*

*3.4.4. Interactor ..... 33*

*3.4.5. InteractiveObject ..... 34*

*3.4.6. Limits ..... 34*

3.5. DESARROLLO ..... 35

*3.5.1. Estructura de una escena ..... 35*

*3.5.2. Escena de calibración ..... 36*

*3.5.3. Escena de juego ..... 37*

**4. CASOS DE USO ..... 38**

*4.1 Búsqueda de estrellas ..... 38*

*4.2. Caza-Topos ..... 39*

*4.3. Fantasmas ..... 40*

*4.4. Recoge Frutas* ..... *41*

**5. RESULTADOS** ..... **42**

5.1. PUBLICACIÓN EN UNITY ASSET STORE ..... 42

5.2. REPOSITORIO EN GITHUB ..... 43

**6. CONCLUSIONES** ..... **44**

6.1. PRODUCTO FINAL ..... 44

6.2. LIMITACIONES ..... 44

*6.2.1. Hardware* ..... *44*

*6.2.2. Limitaciones físicas* ..... *45*

6.2. TRABAJOS FUTUROS ..... 45

**7. BIBLIOGRAFÍA** ..... **47**

**8. ANEXO** ..... **48**

¿CÓMO FUNCIONA? ..... 48

*Detección de la cámara* ..... *48*

*Contour Finder* ..... *48*

*Interactuables* ..... *48*

*¿Cómo usarlo?* ..... *49*

CONFIGURACIÓN ..... 50

# Índice de imágenes

Imagen 1 Lü Interactive Playground [1] ..... 17
Imagen 2 Kinems, videojuegos en el aula [2] ..... 18
Imagen 3 Software de *Kinect* para *Windows* [3] ..... 19
Imagen 4 Esquema desarrollo en cascada ..... 22
Imagen 5 Esquema Hardware ..... 26
Imagen 6 Primer esquema de diseño ..... 27
Imagen 7 Segundo esquema de diseño ..... 28
Imagen 8 Tercer esquema de diseño ..... 29
Imagen 9 Esquema Final de diseño ..... 30
Imagen 10 Esquema UML ..... 31
Imagen 11 Prefab Canvas ..... 35
Imagen 12 Estructura prefab Canvas ..... 36
Imagen 13 Prefab Interactor ..... 36
Imagen 14 Calibration Scene ..... 37
Imagen 15 Demo encontrar estrellas ..... 38
Imagen 16 Demo encontrar estrellas 2 ..... 39
Imagen 17 Demo topos ..... 39
Imagen 18 Game Over topos ..... 40
Imagen 19 Demo fantasmas 1 ..... 40
Imagen 20 Demo fantasmas 2 ..... 41
Imagen 21 Demo recoger frutas ..... 41
Imagen 22 Estructura librería ..... 42

# 1. Introducción

## 1.1. Contexto y alcance

Actualmente, en el mundo de los videojuegos, y sobre todo del *Serious Gaming*, la interacción sin pantallas es un área muy desarrollada. Sin embargo, sigue teniendo numerosas limitaciones que privan al público general y a los desarrolladores de adentrarse en este mundo.

Estas limitaciones suelen ser económicas, ya que un equipo de interacción sin pantallas como mínimo requiere de un proyector y una cámara, así como un software desarrollado específicamente para la función que se va a llevar a cabo, lo cual supone un gran aumento de costes que hace que no todo el mundo pueda permitírselo.

El principal objetivo de este proyecto es conseguir un entorno de desarrollo y de utilidad mucho más accesible y económico para entidades más humildes y desarrolladores independientes. A partir de la librería creada en este proyecto se podrían crear experiencias y videojuegos de todo tipo desde el software de Unity, con el uso de una cámara y un proyector que se utilice para colocar la imagen del juego en una pared, interactuando con una pequeña lámpara o fuente de luz.

### 1.1.1. Aportaciones

El desarrollo de una librería gratuita y de código abierto con estas características podría impulsar en gran medida el desarrollo de *Serious Games* ya que, por una parte, agilizaría el desarrollo simplificando toda su interacción y permitiría que cualquier equipo con cualquier hardware hiciese uso de los proyectos que se creen con la librería, todo ello sin requerir un hardware específico.

### 1.1.2. Impacto potencial

La implementación de una librería con estas características marcaría un hito en la democratización del acceso a tecnologías avanzadas en el ámbito del *Serious Gaming*. Al proporcionar las herramientas necesarias para crear experiencias interactivas sin la dependencia de hardware especializado y costoso, se abre un abanico de posibilidades para innovar en la forma en que interactuamos con los videojuegos.

En última instancia, la librería podría transformar significativamente el panorama del *Serious Gaming*, haciendo que la educación y el entretenimiento interactivos sean más accesibles y enriquecedores para una audiencia global.

## 1.2. Características del proyecto

El proyecto se caracteriza por su enfoque en la accesibilidad y la innovación en el ámbito del *Serious Gaming*. Estos serían los hitos que se cumpliría con la librería:

- **Una amplia compatibilidad:** Este trabajo está diseñado para ser compatible con una amplia gama de dispositivos, la librería garantizará que los desarrolladores puedan trabajar con hardware de diferentes niveles de rendimiento, desde equipos de alta gama hasta dispositivos más modestos.
- **Coste reducido:** Al eliminar la necesidad de equipos costosos, el proyecto reducirá significativamente la barrera económica, abriendo el camino para que una audiencia más amplia experimente y desarrolle videojuegos y experiencias más allá de los controles estándar.
- **Soporte Comunitario:** Con el lanzamiento de la librería como un recurso gratuito y de código abierto, se fomentaría una comunidad activa de usuarios y desarrolladores que contribuirán al crecimiento y mejora continua del proyecto, ya que todo el código se encontraría abierto en todo momento en un repositorio público de GitHub.
- **Documentación Exhaustiva:** El proyecto cuenta con una documentación detallada y tutoriales paso a paso para facilitar la adopción y el uso de la librería, asegurando que los usuarios puedan maximizar su potencial, esta se encuentra dentro de la misma y se podría descargar fácilmente desde el repositorio previamente nombrado.
- **Escalabilidad:** La arquitectura del proyecto estará diseñada para ser escalable, permitiendo la incorporación de nuevas funcionalidades y mejoras sin comprometer la estabilidad del sistema existente. Un sistema al que se entrará en detalle más avanzado el documento y que se pone a prueba en los ejemplos desarrollados.
- **Interacción Natural:** La librería aprovecha tecnologías avanzadas de visión artificial para crear una experiencia de interacción natural y fluida, acercando a los usuarios a una inmersión total en el juego dentro de un espacio controlado.

Estas características están alineadas con el objetivo principal del proyecto: democratizar el acceso a la tecnología de *Serious Gaming* y empoderar a los desarrolladores para que innoven y creen experiencias de juego significativas y educativas.

# 2. Objetivos

## 2.1. Descripción del problema

Actualmente, el desarrollo de videojuegos en muchas ocasiones se encuentra limitado por el hardware, y eso se lleva al extremo en la interacción sin pantallas, hubo una época en el que la industria abordaba este campo en busca de otra perspectiva, fue entonces cuando surgieron aparatos como *Kinect* o *PSMove*, estos proyectos abordan la idea de conseguir otro tipo de interacción para desarrollar sus videojuegos, algunos de ellos consiguieron bastante éxito.

Actualmente estos proyectos se encuentran fuera del mercado y no se sigue dando soporte para el desarrollo o su utilización, por lo que el desarrollo de este tipo de proyectos se complica por la falta de medios.

Durante los últimos años, en los colegios y museos se va implantando cada vez más el uso de las tecnologías para evolucionar la forma de aprendizaje y encajar mejor con una sociedad cada vez más tecnológica. El desarrollo de este tipo de productos se ve muy limitado por factores como los económicos, ya que se necesita una inversión grande y muy específica en un tipo de hardware y en el desarrollo de un software muy concreto para conseguir este tipo de experiencias.

## 2.2. Estudio del problema

Tras el análisis exhaustivo del problema, se presentan numerosas opciones para encontrar una solución, una de las más llamativas podría ser el desarrollo en Realidad Virtual, sin embargo, el uso de gafas a edades tempranas no está recomendado porque puede resultar muy perjudicial para la vista de los niños, por lo que el uso de una librería con Realidad Virtual se vería limitado en gran medida por el decremento de los potenciales usuarios de la misma.

Por otra parte, según numerosos estudios, el uso de pantallas a edades tempranas puede resultar muy perjudicial para el desarrollo de los niños, es por eso que existen muchos casos de museos y escuelas, que queriendo hacer uso de los beneficios de implementar la tecnología, han optado por introducir en aulas y exposiciones algún tipo de forma de interacción sin pantallas. Es en ese momento en el que entra la Realidad Mixta y sus numerosos beneficios.

La Realidad Mixta permite, entre otras cosas, crear un entorno interactivo sin el uso de pantallas, cuya limitación se encuentra en el espacio necesario para el desarrollo de estas actividades.

Dentro de todas las opciones que ofrece la Realidad Mixta, la que más potencial tiene sería la posibilidad de crear entornos interactivos con el uso de un proyector, una cámara y un ordenador. Este tipo de proyectos son beneficiosos en gran medida, ya que amplían de forma significativa las opciones a la hora de crear una experiencia sin la necesidad de tener un espacio físico muy complejo, a la vez que se deja de depender de un hardware específico. Asimismo, se consigue desarrollar habilidades en los usuarios que no se utilizarían con un entorno interactivo más estándar.

### 2.2.1. Estado del arte

Actualmente y durante los últimos años, la industria del videojuego y la tecnológica ha tenido un crecimiento exponencial sin freno, causando que sus beneficios y aplicaciones lleguen a todo tipo de campos y se utilicen abiertamente para facilitar y mejorar las tareas cotidianas e incluso descubrir nuevas formas de entretenimiento y ocio.

A día de hoy, existen numerosos proyectos que consiguen introducir la tecnología en colegios y sus aulas, un claro ejemplo se encuentra en *Lü Interactive Playground*, una empresa que introduce una nueva forma de aplicar la tecnología en los gimnasios de los colegios. Utiliza un proyector y con la ayuda de una cámara y pelotas de colores que se lanzan contra la pared se crea una interacción con la que se juega a distintos juegos.

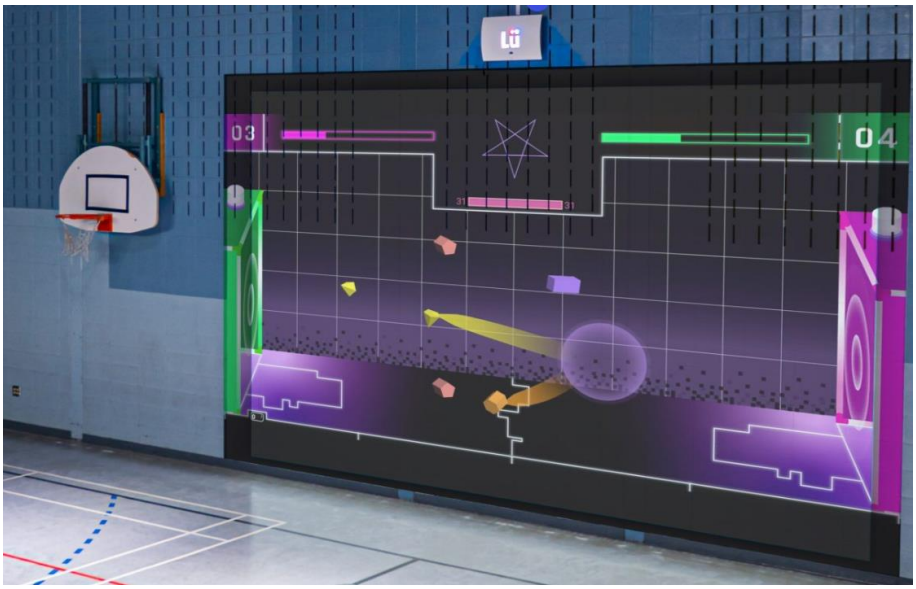



**Imagen 1 Lü Interactive Playground [1]**

Esta es una forma de innovar a la hora de dar clase y al mismo tiempo, conseguir nuevas experiencias desde una interacción totalmente natural como puede ser lanzar una pelota. Este proyecto resulta interesante porque ayuda a romper un estigma contra las tecnologías en las aulas y aporta un valor añadido a las clases, así como abre muchas oportunidades y posibilidades a la hora de hacer proyectos únicos para esta plataforma.

Por otra parte, se encuentra *Kinems*, una empresa que une aún más el concepto de videojuegos y aulas. En este caso se utiliza un *hardware* más convencional, sin embargo, explora otra forma de introducir la tecnología en el aula.

**Imagen 2 Kinems, videojuegos en el aula [2]**

Se utiliza un sistema de interacción como *Kinect* en unos videojuegos específicamente diseñados y desarrollados para la plataforma, estos videojuegos se utilizan en aulas con cada alumno y con los resultados que se obtienen se realiza un informe de los alumnos y su rendimiento en los juegos.

Este es un modelo diferente del uso de la tecnología, un uso como herramienta más que como medio, con la utilización de este tipo de proyectos se consigue una medida cuantificable que de formas más convencionales sería imposible de conseguir.

En el lado de los videojuegos como ocio, también se encuentran numerosos hitos en el desarrollo de formas de interacción diferentes, la industria ha ido evolucionando sin parar y este tipo de proyectos pasaron por una época de gran utilización con la llegada de *Kinect* por parte

de Microsoft y de *PSMove* por parte de PlayStation. Ambos dispositivos, sin embargo, ya no están en la vanguardia del desarrollo de las empresas y actualmente están descatalogados en ambas tiendas.

El caso de Kinect tuvo especial éxito a la hora de usarse en museos y experiencias educativas gracias a sus cámaras de profundidad y las muchas posibilidades que ofrecen. Sin embargo, su difícil acceso actualmente complica el acceso a este tipo de tecnologías y ha provocado que este tipo de proyectos sean mucho menos frecuentes a lo que eran anteriormente.

**Imagen 3 Software de *Kinect* para *Windows* [3]**

Por otro lado, PSMove no apareció tan ligado a su utilización en museos o experiencias fuera del ocio del videojuego, su objetivo como producto consistía en conseguir experiencias en videojuegos que no se podían conseguir con un mando clásico, como hacer magia o usar pistolas. Para ello hacía uso de unas bolas de luz pegadas a un mando alargado y una cámara que llevaría el seguimiento de esos colores para conseguir una interacción, esa tecnología es la principal inspiración de este proyecto.

En conclusión, diferentes tipos de interacción se han investigado a lo largo de la historia de la industria, algunos con muy buenos resultados y proyectos. Sin embargo, dentro de la industria de los videojuegos ya no existen ejemplos en activo donde prevalezcan este tipo de desarrollos,

y fuera de la industria, se encuentran unos nichos que ofrecen grandes proyectos, pero con una inversión que en algunos casos resulta imposible de afrontar.

### 2.2.2. Elecciones para la solución

Se investigaron algunas opciones para conseguir una solución, pero la que mejor encajaba con el planteamiento de la misma era desarrollar una librería para el motor de videojuegos Unity. Con esta decisión se conseguiría llegar a un mayor número de desarrolladores, puesto que Unity es en la actualidad uno de los softwares de desarrollo de videojuegos y de experiencias interactivas más utilizados alrededor del mundo.

Por otra parte, para conseguir una interacción física, se determinó que la librería hiciera uso de la tecnología de OpenCV [4] para el análisis y procesado de las imágenes capturadas con la cámara del ordenador. Siendo OpenCV una de las bibliotecas más importantes de visión artificial y que además de ser libre, cuenta con la existencia de un paquete que implementa su funcionalidad dentro de Unity.

De esta forma, se podría construir una librería abierta para todo el mundo con la cual poder iniciar este tipo de proyectos, con los beneficios del uso de tecnologías previamente existentes y mundialmente conocidas como base para el desarrollo.

### 2.2.3. Problemas de Hardware

Como se nombraba anteriormente, no se utiliza una tecnología sin explorar, sin embargo, la realidad es que en la actualidad este tipo de tecnologías no son de fácil acceso, ya sea por encontrarse descatalogadas o por el alto precio que supone un equipo completo de interacción.

En el caso de los equipos de interacción concretos, necesitan un desarrollo enfocado y organizado para *hardware* específico y requiere de una especialización en este.

Con esta solución, desaparecerían todos los problemas de *hardware* debido a que los requerimientos para el uso de la librería serían de una cámara, un ordenador, un proyector y una lámpara o fuente de luz; un equipo que prácticamente todas las escuelas o museos son capaces de asumir en caso de no contar con ello, de forma que además se elimina la barrera económica existente.

El fin de este proyecto es conseguir un entorno generalizado para cualquier tipo de hardware y, sobre todo, que no requiera de equipos de última generación y máxima potencia para un correcto funcionamiento.

## 2.3. Metodología aplicada

La metodología aplicada para este proyecto fue el desarrollo en cascada, un enfoque metodológico que divide el desarrollo en una serie de etapas secuenciales en las que se va avanzando según se van completando las etapas.

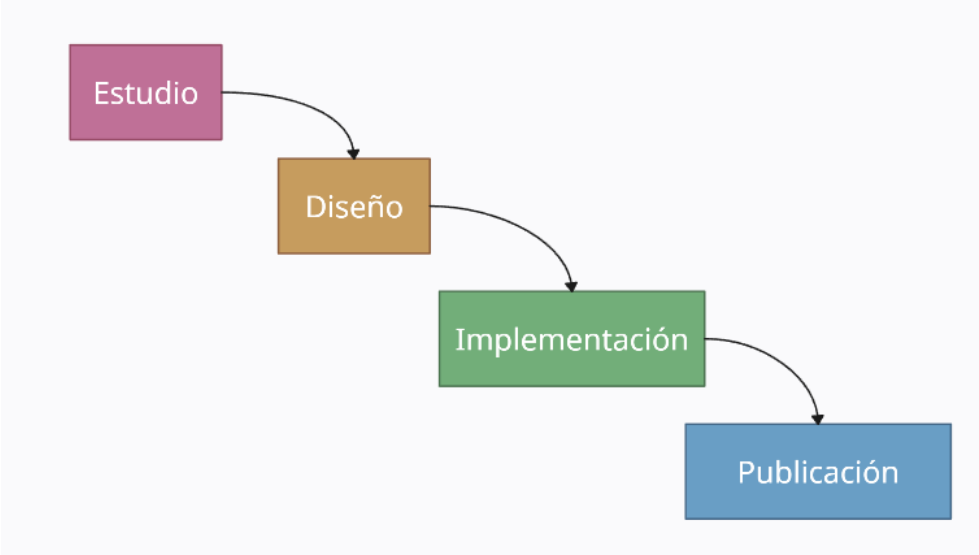



**Imagen 4 Esquema desarrollo en cascada**

El desarrollo en cascada o secuencial se basa en dividir el desarrollo del proyecto en etapas independientes, e ir resolviendo una a una sin mezclarse, no se inicia una etapa hasta que se ha completado la anterior. Cada una de las secciones puede repetirse hasta que se da por finalizada, y dentro de cada una de ellas hay diferentes tareas que se realizan, como por ejemplo en la fase de implementación, se encuentra las tareas de desarrollo y las tareas de depuración.

Una vez se inicia una etapa nueva, no se suele volver a las anteriores, ya que se promueve una metodología de trabajo efectiva en la que todas y cada una de las etapas se completan de forma satisfactoria y sin errores.

En la fase inicial de la investigación, se realizó un análisis exhaustivo de proyectos existentes en el ámbito de entornos interactivos educativos. Un caso destacado fue el de *Lü Interactive Playground* [1], una innovadora empresa canadiense que integra tecnología interactiva en espacios educativos. Su sistema, compuesto por un proyector, una cámara y un conjunto de luces y sonido, permite la interacción mediante el lanzamiento de pelotas de colores contra superficies proyectadas.

Este modelo sirvió como punto de partida conceptual para el desarrollo. Sin embargo, se identificó la necesidad de adaptar y evolucionar este concepto para crear un producto más universal y accesible. El desafío principal radicaba en lograr una interacción confiable y compatible con una amplia gama de dispositivos. La solución emergió del estudio de OpenCV, donde se determinó que el contraste de luz era el método más efectivo y accesible para la interacción, independientemente de la calidad de la cámara utilizada.

Tras el análisis y estudio de las posibilidades se inició la etapa de diseño. El objetivo de esta fase consiste en la realización y definición del diseño del *software*, en el que se definirán todas las clases y funciones que se desarrollarán en la siguiente etapa. Se inició con la elaboración de un esquema conceptual del *software,* del cual hubo varias fases de diseño que se detallarán más adelante. El hito para dar por finalizada la etapa era encontrar un diseño de la librería que cumpliese todos los objetivos que se querían desarrollar.

Siguiendo la definición del enfoque encontrado en la fase anterior, se procedió a la etapa de desarrollo. Iniciando con una fase de prototipado para validar la viabilidad técnica de las interacciones basadas en contraste de luz, en lo que se entrará en detalle más adelante. Tras confirmar su eficacia, se avanzó hacia la implementación y refinamiento del sistema, asegurando su funcionalidad en diversos entornos y con distintos dispositivos.

Tras el desarrollo funcional, se desarrollaron una serie de pequeñas demostraciones para aplicar la librería, con el objetivo de tener un prototipo completo sobre el cual poder hacer un estudio final de resultados. Todas las demostraciones y sus desarrollos serán analizadas más adelante en el documento.

Finalmente, con todo el desarrollo completado, el último de los periodos consistía en la publicación de la librería para que todos los objetivos del proyecto se puedan completar, para ello se utilizó la tienda de *Unity*, *Unity Asset Store.* Para poder publicar es necesario cumplir unos estándares introducidos por la empresa para mantener la calidad de la tienda además de seguir un proceso estructural y de revisión personal para cada librería.

Una vez completada esta etapa, se podría añadir en un futuro la fase de mantenimiento, un ciclo que suele estar presente en este tipo de metodologías, el cual consiste en realizar actualizaciones periódicamente que mantengan la calidad del software que se ha desarrollado.

# 3. Descripción informática

## 3.1. Esquema de Hardware

Al hacer el análisis se determinó buscar una solución aplicable a los dispositivos que se podrían encontrar en cualquier colegio o aula. Actualmente los colegios están tomando una perspectiva más tecnológica y adoptando unas clases con dispositivos electrónicos que aportan mucho valor a las clases.

En prácticamente todas las aulas es posible la adquisición y utilización de un proyector. Es por eso que se decidió como requisito mínimo de *hardware* para esta librería la posibilidad de utilizarse con un ordenador de cualquier gama, un proyector, una webcam y una pequeña fuente de luz. Unos requisitos que probablemente cualquier colegio podría cumplir sin ningún problema.

No existe una colocación específica para garantizar el correcto funcionamiento, sin embargo, es recomendable que la cámara y el proyector se encuentren lo más cerca posible y apunten hacia el mismo lugar.

De este modo, se consigue un entorno de interacción más natural gracias a que la sombra que se proyecta y el lugar donde se producen las interacciones en el mundo virtual se encuentran en el lugar más parecido posible.

Por otro lado, este enfoque consigue minimizar al máximo los posibles problemas de distorsión y mala calibración producidos por la cámara.

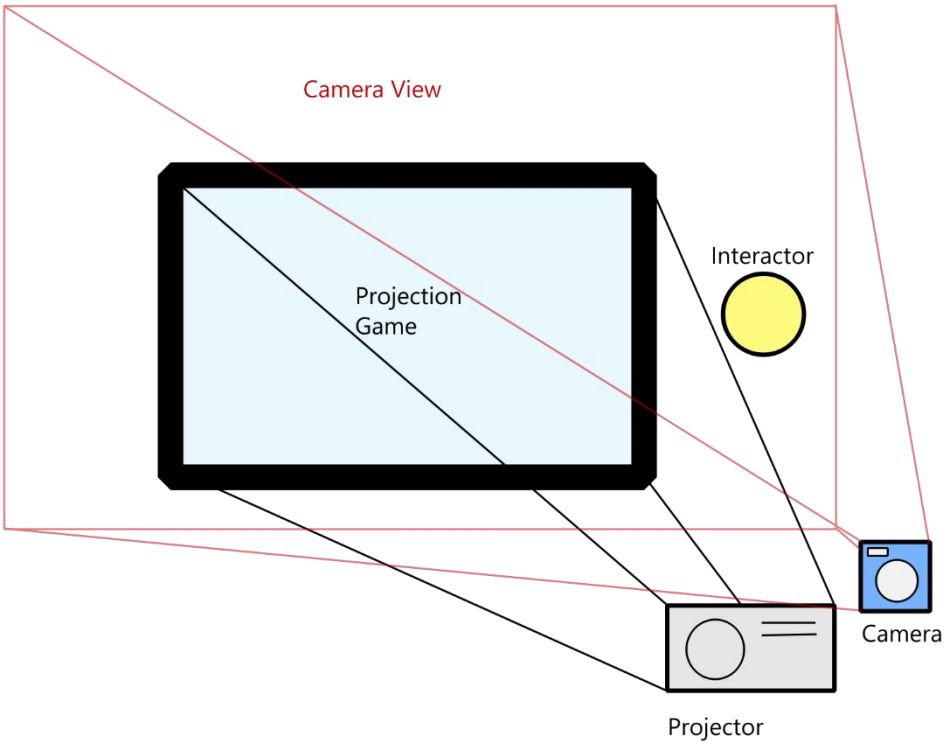



**Imagen 5 Esquema Hardware**

Como se puede apreciar en el esquema del hardware, el espacio que detecta la cámara es más grande a la proyección, de ahí la necesidad de calibrar el tamaño de la imagen proyectada y ajustar las posiciones del *interactor* en función de estos límites.

La configuración y posición mostrada en el esquema no es la única válida, pero si es la más optima. El proyector y la cámara se encuentran cercanos, por lo que la proyección y la vista de la cámara cubren la mayor área común posible. Aun con ello, esta configuración está lista para adaptarse a cualquier colocación de los elementos de hardware.

## 3.2. Diseño

### 3.2.1. Evolución del diseño

En cuanto al diseño informático de la librería, se plantea un modelo simple, pero a la vez funcional y muy escalable debido a la naturaleza del proyecto. En el primer modelo propuesto, la librería contaría con una clase que tomase y gestionase toda la información de la cámara para después enviar esos datos al motor de juego *Unity* y utilizarlos ahí.

De esta forma, entraría a la clase la imagen de la cámara sin modificar, estos datos se utilizarían dentro de la clase y con *OpenCV* se realizaría todo el cómputo necesario para conseguir los puntos de interacción. Tras eso, la información de la imagen procesada se enviaría a todos los objetos que lo necesiten en el juego.

Este modelo planteaba varios problemas, al colocar todo el cómputo en la misma clase, el programa acabaría perdiendo demasiado rendimiento, haciéndolo imposible de usar.

Asimismo, llamar a los métodos de cámara cada iteración del programa resulta muy costoso y todo ello ralentiza la ejecución.

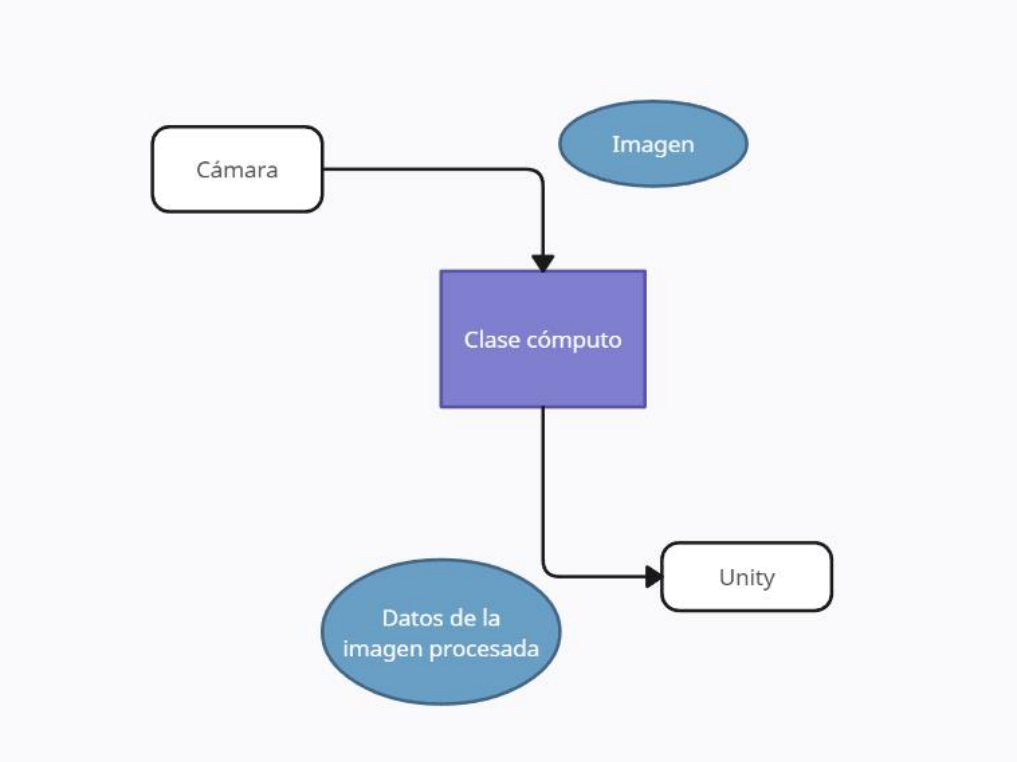



**Imagen 6 Primer esquema de diseño**

Tras eso, se planteó modularizar la clase cómputo en una serie de clases que realizaran el cómputo de forma asíncrona y limitar los pasos de la imagen procesada a *Unity*. Con la creación de una clase asíncrona se conseguían resolver numerosos problemas, como el de rendimiento y el de sincronización entre clases.

Para ello se creó una clase auxiliar que guardase en su interior la información de la imagen procesada, y solo se actualizase cada vez que se consiguiera una imagen nueva. De esa forma no se necesita esperar a conseguir el cómputo para seguir con el juego, ya que los objetos toman la información de la última imagen guardada.

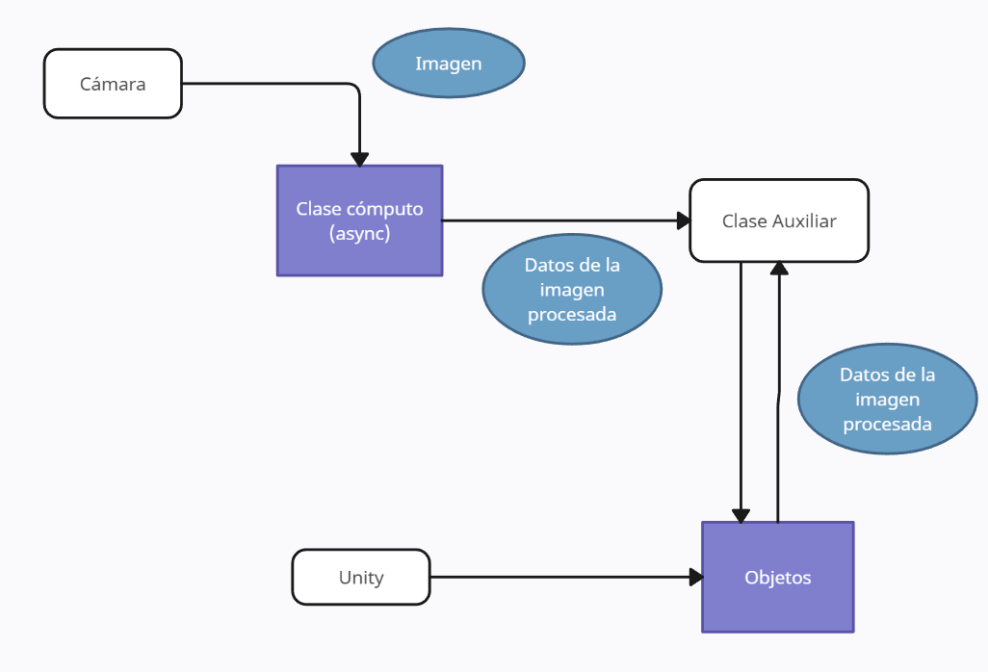



**Imagen 7 Segundo esquema de diseño**

Sin embargo, este diseño seguía contando con un grave problema de rendimiento arrastrado por la gran cantidad de llamadas a la clase Cámara necesarias para actualizar constantemente la información de la imagen a procesar.

Para suplir este inconveniente, el siguiente concepto enviaba todo el cómputo de *OpenCV* al interior de una clase heredada de cámara, de esta manera, la extracción de la imagen de la cámara y su cómputo quedaría en el interior de una misma clase, por lo que desaparecerían todas las operaciones de llamadas entre clases que provocaban una bajada considerable del rendimiento de la librería.

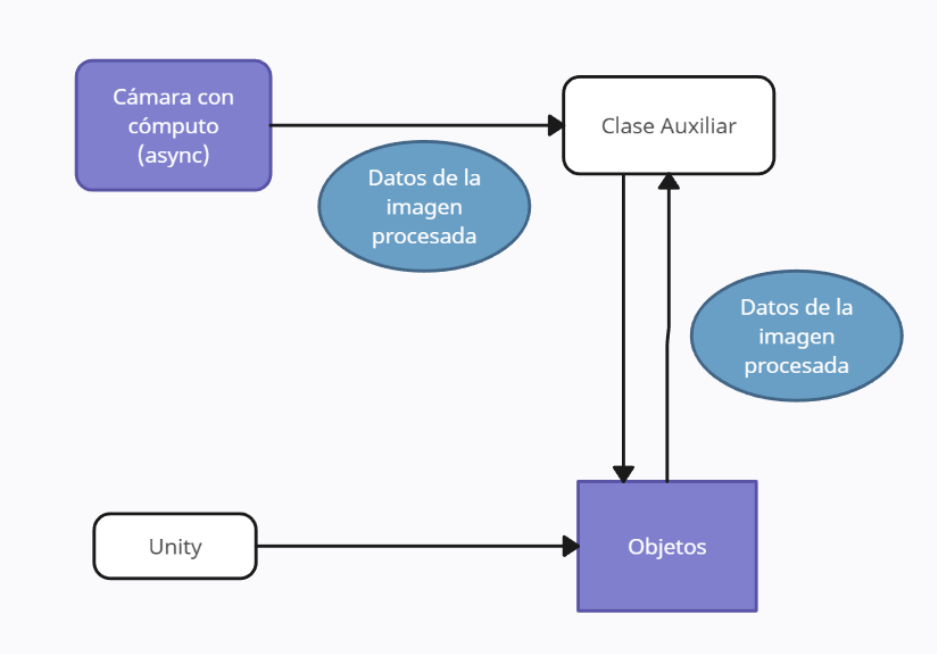



**Imagen 8 Tercer esquema de diseño**

Con este enfoque, las llamadas y la dependencia entre distintas clases se minimizaban, ya que el cómputo no dependía de llamadas a clases externas. Todo el procesamiento de las imágenes quedaba encapsulado dentro de la misma clase, y de esta se extraía únicamente la información de la imagen procesada necesaria para cada objeto.

### 3.2.2. Diseño final

Finalmente, con este último diseño se conseguía un buen rendimiento, pero se podía simplificar aún más y conseguir una eficacia mayor a la hora de conseguir una interacción. Asimismo, el modelo anterior limitaba bastante la forma de aplicar un *interactor* en el videojuego.

Todos los diseños anteriores utilizaban los propios datos de la imagen para determinar si un objeto debería interactuar o no. Se tomaba el valor de la posición de cada objeto y se comprobaba si tenía o no tenía luz en la imagen tomada de la Cámara. Esto resultaba en un diseño totalmente funcional, pero no ideal, ya que no se aprovechaba las ventajas de un motor como *Unity* y su sistema de físicas y colisiones integrado.

Por estos motivos, el último diseño cambia la forma de enfocar la interacción entre la cámara y el mundo virtual de *Unity*, para conseguir unos intercambios de información más ágiles y livianos había que deshacerse de enviar los datos de la imagen completa, para ello se desarrolló el cómputo dentro de la cámara para que calculase la posición del centro del objeto *interactor* de la cámara.

Con la información de la posición del centro del *interactor* los intercambios de información se simplifican, y la interacción se vuelve más general y personalizable. Esta posición se pasaría a una clase *Interactor*, la cual funcionará como un cursor dentro del juego y con la que se podrá actuar sobre los objetos interactivos dentro de cada escena de *Unity* mediante el sistema de físicas y colisiones.

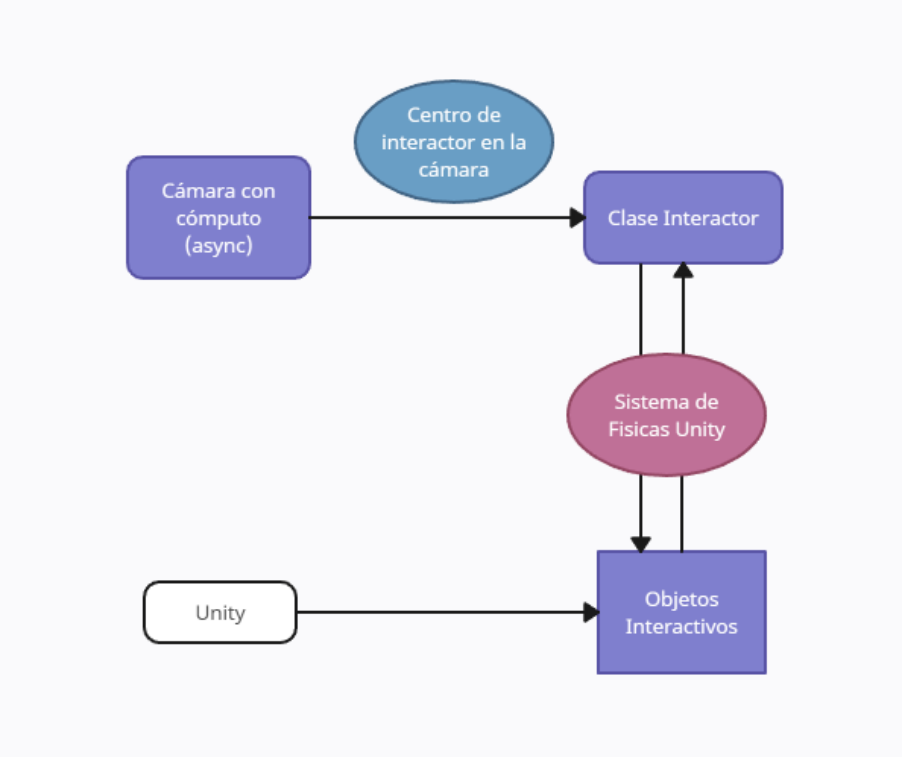



**Imagen 9 Esquema Final de diseño**

Con este último diseño, se consiguió una estructura de la librería tan funcional como escalable, mientras se aprovechan todas las ventajas que un motor como *Unity* puede aportar a un proyecto como este.

## 3.3. Implementación

La implementación de la librería queda con abstracciones que permiten una gran ampliación con una base sólida para el desarrollo. Destacan las clases más abstractas *WebCam* e *InteractiveObject* que sirven como estructura para añadir más adelante una funcionalidad específica en clases heredadas.

A continuación, se muestra un esquema UML que resume las clases con las que cuenta la librería y sus principales funciones y variables (excluyendo variables de control y funciones internas de Unity para mejorar la claridad). En el esquema se puede apreciar cómo la mayoría de clases heredan de las clases abstractas utilizando sus funciones. Además, se muestra dónde se encontrarían los objetos específicos que el desarrollador quiera introducir en el juego.

### 3.3.1. Esquema UML

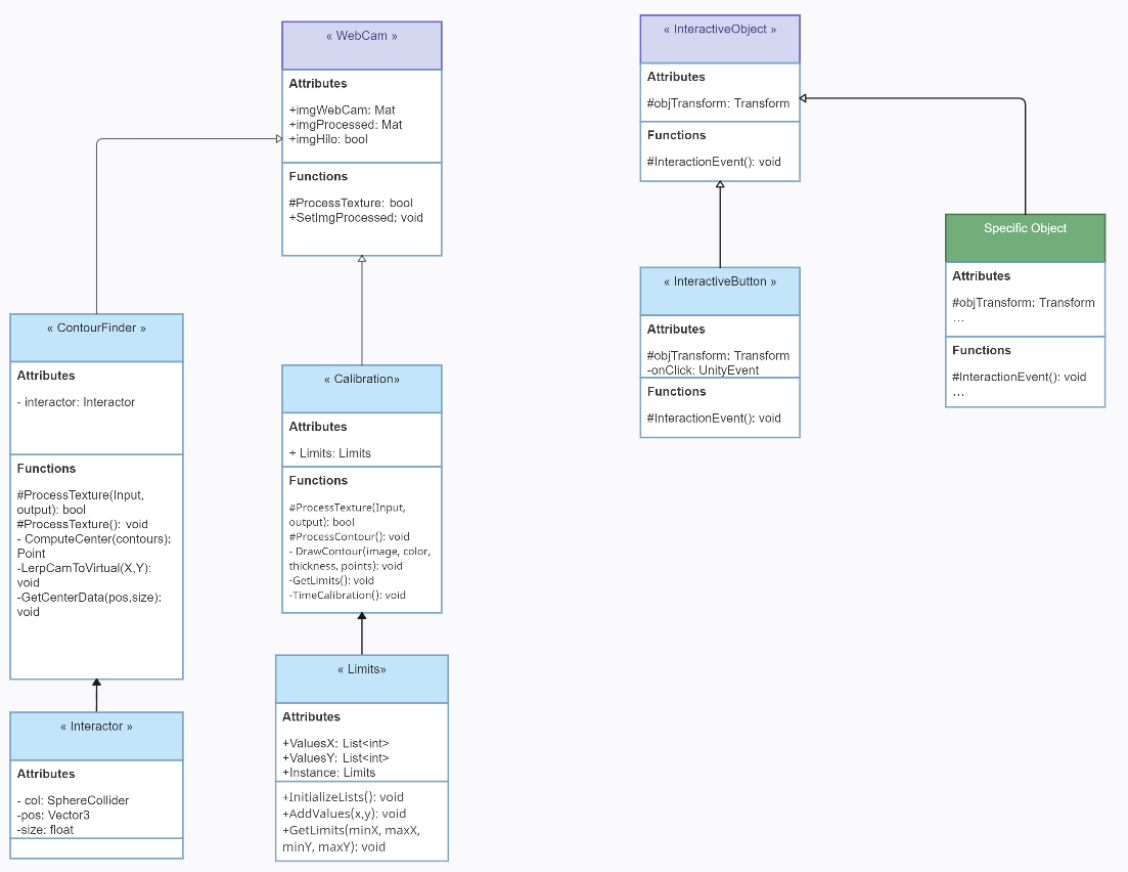



**Imagen 10 Esquema UML**

### 3.3.2. Escalabilidad

Utilizando como base las clases desarrolladas y descritas a continuación, la creación de experiencias interactivas y de videojuegos no se ve limitada por la librería. A razón de que se busca un modelo muy escalable, todas las clases del proyecto aceptan ampliaciones y modificaciones para adaptarse al producto que se busque desarrollar sin perder ningún tipo de funcionalidad ni rendimiento.

Por la propia naturaleza de la librería y su estructura interna, el enfoque de un *interactor* que utilice el sistema de físicas de *Unity* para provocar eventos en objetos permite la creación de todo tipo de videojuegos y experiencias sin suponer ningún tipo de limitación al desarrollo. Por otro lado, la adaptabilidad a todo tipo de *hardware* y su enfoque general y flexible puede permitir realizar proyectos de gran tamaño con esta librería.

## 3.4. Clases

En este apartado se entrará en detalle de las clases que contiene la librería y su funcionalidad.

### 3.4.1. WebCam

Esta clase hereda de la clase *WebCamera* de la librería de *OpenCV* [5]. Su funcionalidad consiste en tomar la imagen de una cámara conectada al ordenador y preparar la imagen como una textura para más adelante ser procesada con *OpenCV*. Se requiere la creación de esta clase y no la utilización de *WebCamera* para conseguir el funcionamiento asíncrono en la librería. Para ello se crean unas variables que servirán como semáforos para la preparación de la imagen, algo que no se encuentra contemplado en el diseño original de la librería de *OpenCV*

Esta clase sirve como puente entre el espacio físico y el virtual, uniéndolo por medio de la imagen entrante de la cámara. Y más adelante se utilizará como clase abstracta para añadirle el procesamiento de la textura en las clases definidas debajo.

### 3.4.2. Calibration

Esta clase hereda de *WebCam* y su funcionalidad consiste en utilizar la imagen entrante desde la cámara para calibrar los límites de la proyección. La clase procesa la imagen de la cámara para convertirla a una máscara de color, en este caso verde, que como se ha nombrado anteriormente, es el color utilizado para los cuadrados de la escena de calibración. De esta forma se toma de manera eficaz las posiciones de las esquinas de la proyección.

La máscara se realiza con dos límites con valores HSV, uno superior y otro inferior y se toman todos los valores de la imagen entre ambos límites. El color verde es fácil de conseguir contrastar con el fondo para facilitar la máscara.

Una vez procesada la imagen y creado la máscara, la clase tomará los valores mínimos y máximos de las apariciones como las esquinas de la proyección y lo guardará en un patrón *singleton* llamado *Limits* para utilizarlo más adelante durante toda la ejecución.

### 3.4.3. ContourFinder

Esta clase es la presente en los juegos y es la encargada de procesar la imagen y conseguir una interacción. Hereda de *WebCam* para poder modificar la imagen como textura y así conseguir realizar los cálculos y trabajos necesarios.

Esta clase no realiza una máscara de color para evitar problemas con diferentes lámparas utilizadas como *interactor*. En su defecto, esta clase hace una máscara por umbral de luminosidad, toma los valores más luminosos y transforma la imagen a blanco y negro, donde el negro son todos los valores por encima del umbral y el blanco el resto. Con esta información, se realizan contornos que recogen las áreas negras de la imagen procesada. Tras eso, de todos los contornos se toma el contorno con mayor área, ya que con casi total seguridad se va a tratar de la lámpara, y finalmente se calcula el centro de ese contorno que se utilizará como la posición del objeto que actuará como cursor en los juegos.

Esta clase trabaja de forma asíncrona, tiene un hilo que sirve como control para no hacer un procesamiento desmedido de la imagen a cada iteración. Este hilo permite el cómputo cuando existe algo en la imagen de la cámara que podría considerarse un *interactor* y cuando se ha realizado todo el cómputo anterior.

### 3.4.4. Interactor

La clase *Interactor* es la encargada de proporcionar a la librería un objeto con el que poder interaccionar con el mundo, esta clase es la que se introduce en el objeto con el que se calculan las colisiones con los objetos interactivos y el que se mueve por el mundo en función de la luz detectada por la cámara.

Esta clase se encarga de tomar desde el *ContourFinder* el centro y tamaño de la fuente de luz, con esa información, en cada iteración del *update* se modifica la posición del objeto y el tamaño del *collider*.

Al añadir el script a un objeto, se genera un *SphereCollider* requerido para el funcionamiento, el cual es el encargado de iniciar los eventos interactivos de los objetos.

### 3.4.5. InteractiveObject

*InteractiveObject* es una clase abstracta que se encarga de añadir interactividad a los objetos que se quiera que reaccionen al *interactor*. Todos los objetos que se quiera que desarrollen una interacción con la luz deberán heredar la funcionalidad de esta clase.

Dentro de su funcionalidad, se encuentra la función abstracta *InteractionEvent*, la cual se deberá añadir una funcionalidad concreta en los objetos que hereden esta clase. Esto otorga a la librería una gran capacidad de adaptarse a cada proyecto.

La función *InteractionEvent* se llama cuando entra en contacto con el *interactor* y se llama al *OnTriggerEnter* del script.

### 3.4.6. Limits

*Limits* es una clase que sigue el patrón *Singleton*, es decir, que se mantiene durante toda la ejecución. Este código guarda los valores de los límites de la proyección para utilizarlos a lo largo de las diferentes escenas.

Estos límites se utilizan en *ContourFinder* para calcular la interpolación entre la posición de la cámara y la posición en el mundo virtual. Gracias a esa interpolación se consigue calcular correctamente la posición del *interactor*.

## 3.5. Desarrollo

Para ello partiendo de un proyecto en blanco se introdujo la librería gratuita que implementaba *OpenCV en Unity*. Esta librería se encuentra libre para su acceso en la *Asset Store de Unity*. [5].

Esta librería de *OpenCV* añade clases y funcionalidades de visión artificial para que puedan utilizarse dentro de *Unity*, las que se utilizarán dentro de este proyecto son las relacionadas con la identificación de formas y contrastes de colores a partir de una imagen de una *Webcam*.

### 3.5.1. Estructura de una escena

Para el correcto funcionamiento de la librería, se hace uso de las funcionalidades del *Canvas* de *Unity*. Se debe crear un *canvas* y unirlo a la cámara para que se mantenga dentro del mundo, este *canvas* será el espacio donde se desarrollarán las interacciones y conformará el espacio de juego. Dentro de él se colocarán todos los objetos que quieran ser visibles.

Para facilitar la creación de un espacio interactivo, se ha creado un *prefab* en *Unity* que contiene todo lo necesario para añadirlo a la escena y poder interaccionar con el proyector y la fuente de luz.

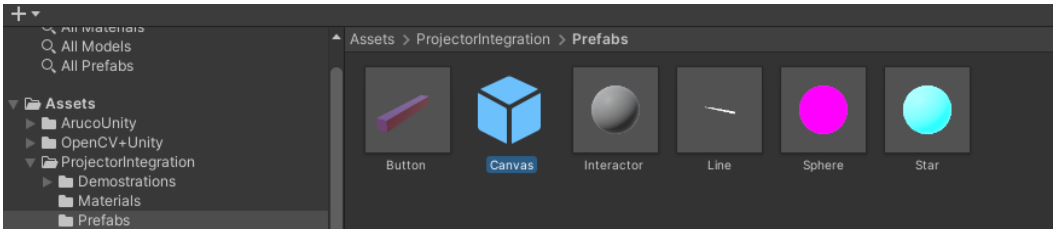



**Imagen 11 Prefab Canvas**

Este *prefab* añade el *canvas* citado anteriormente, un *gameobject* que muestra la imagen de la cámara y que contiene el script *ContourFinder*. Dentro de *ContourFinder* se encuentra todo el cómputo para detectar la fuente de luz en la cámara y transmitirla al espacio virtual.

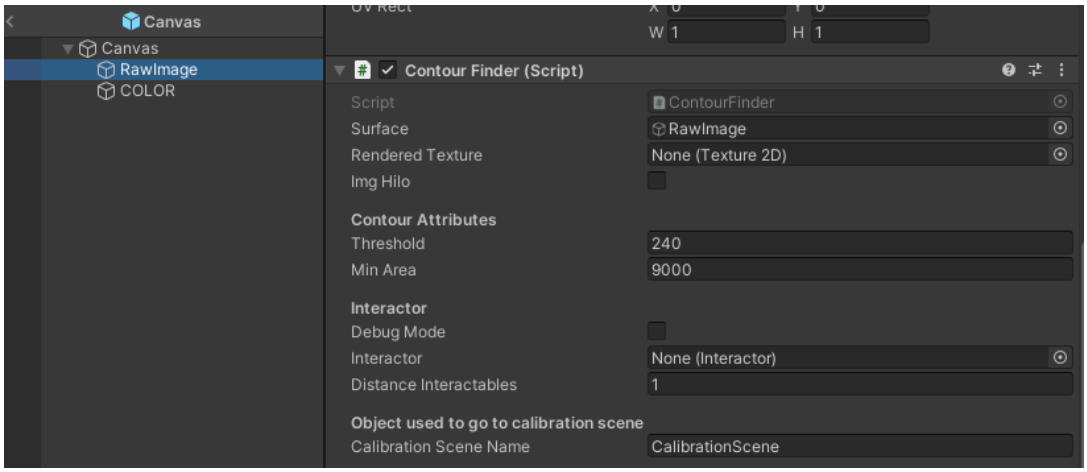



**Imagen 12 Estructura prefab Canvas**

Asimismo, para conseguir la interacción hay que añadir el *prefab Interactor* en la escena y ajustarlo para que se encuentre en el mismo espacio en el que se encuentra el *canvas* y se encontrarán los objetos interactivos.

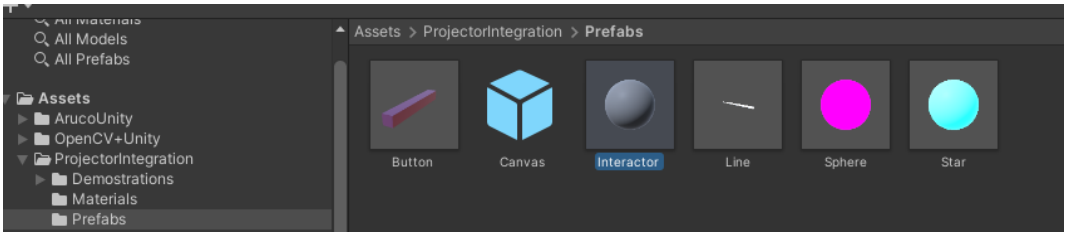



**Imagen 13 Prefab Interactor**

Una vez introducidos ambos *prefabs*, la escena estaría lista para su funcionamiento, y quedaría añadir los objetos interactivos y la funcionalidad deseada.

## 3.5.2. Escena de calibración

Antes de iniciar cualquier experiencia es necesario calibrar el tamaño de la proyección en la cámara, para ello se utiliza una escena predeterminada que se llama al iniciar la aplicación y sirve para esto mismo.

Esta escena cuenta con dos cubos verdes en las esquinas superior izquierda e inferior derecha. Estos cubos sirven para marcar en la proyección los límites de la imagen, y así más adelante se podrá calcular la posición del *interactor* en el mundo con una simple interpolación.

Dentro del script *ColorMask*, se realiza una máscara de color para tomar el color verde de la pantalla e ignorar el resto de gama de colores que capture la cámara. La información de la

posición de las esquinas se guarda en *Limits*, que mantendrá los valores de los límites durante toda la ejecución.

Tras calibrar la posición de la proyección, automáticamente se inicia otra escena que se selecciona en la propia escena de calibración.


**Imagen 14 Calibration Scene**

### 3.5.3. Escena de juego

Finalmente, una vez completados todos los pasos anteriores, se llega a una escena de juego, esta librería permite hacer escenas de juego muy personalizables y adaptables a casi cualquier tipo de experiencia.

Para conseguir una escena funcional se debe colocar el *prefab* del *canvas* para tener un espacio de interacción, donde se colocarán todos los objetos que tengan que tener algún tipo de interacción con la luz. Todos estos objetos deben colocarse con el script o un script que herede de la clase *InteractiveObject*.

Hacer los objetos *InteractiveObject* permite crear un comportamiento cuando entra en contacto con el *interactor*, y este tipo de comportamientos son los que crean cada experiencia personalizable y adaptable a los requerimientos.

También se debe añadir el *prefab* del *interactor*, el cual tomará la información de la posición de la fuente de luz y la traspasará a posición en el mundo. Con esos datos y la interacción del sistema de físicas y colisiones de Unity se crean las interacciones entre el *interactor* y los objetos interactivos.

Por otro lado, el *interactor* es totalmente editable para añadirle funcionalidad y conseguir cualquier tipo de efecto, como se puede ver en los ejemplos más adelante en el documento.

# 4. Casos de uso

Para exponer la funcionalidad de la librería, se han desarrollado 4 pequeñas experiencias y videojuegos que explotan la interactividad que proporciona el proyecto.

Estas experiencias están enfocadas a aprovechar de diferentes formas las opciones que otorga el uso de una interacción sin pantallas.

## 4.1 Búsqueda de estrellas

En este minijuego se sitúa la cámara en el interior de una cúpula como si fuera un observatorio. En la pantalla se encuentran dos botones que sirven para navegar entre constelaciones, y se utiliza el *interactor* para pulsarlos. Los botones sirven para navegar a través de las diferentes constelaciones y las coordenadas y nombres de las constelaciones van cambiando en función de donde apunte la cámara.

Tras elegir la constelación, con el uso de la luz de la lámpara usada como *interactor* se puede ir "pintando" las estrellas de la pantalla, una vez tocadas todas estas se hacen más grandes y finalmente la constelación aparece y se completa para poder observar cómo se vería.

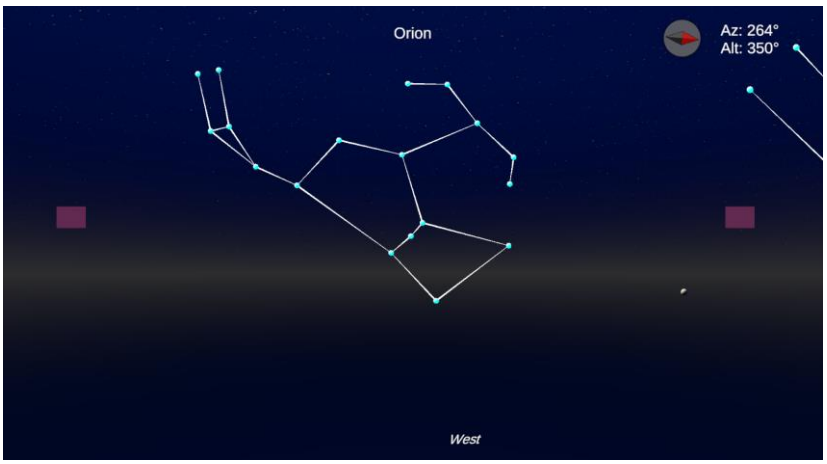



**Imagen 15 Demo encontrar estrellas**

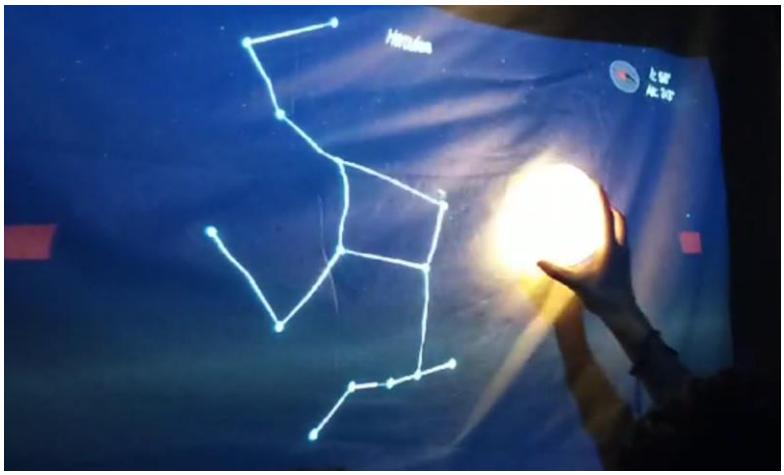


**Imagen 16 Demo encontrar estrellas 2**

Las constelaciones se encuentran en la posición en el cielo aproximada en la que se encuentran, por lo que serviría para dar una clase o exposición básica de astronomía. Estas coordenadas se muestran con azimut y altitud, que muestra la orientación horizontal y la vertical respectivamente.

## 4.2. Caza-Topos

Este minijuego se basa en el clásico juego de feria, donde aparecen topos de agujeros y con una maza debes golpearlos.

En este caso la maza es el *interactor*, los topos van apareciendo aleatoriamente por la pantalla y hay que golpearlos con la luz, recibiendo así una cantidad de puntos en relación al tiempo que lleva en pantalla el topo.

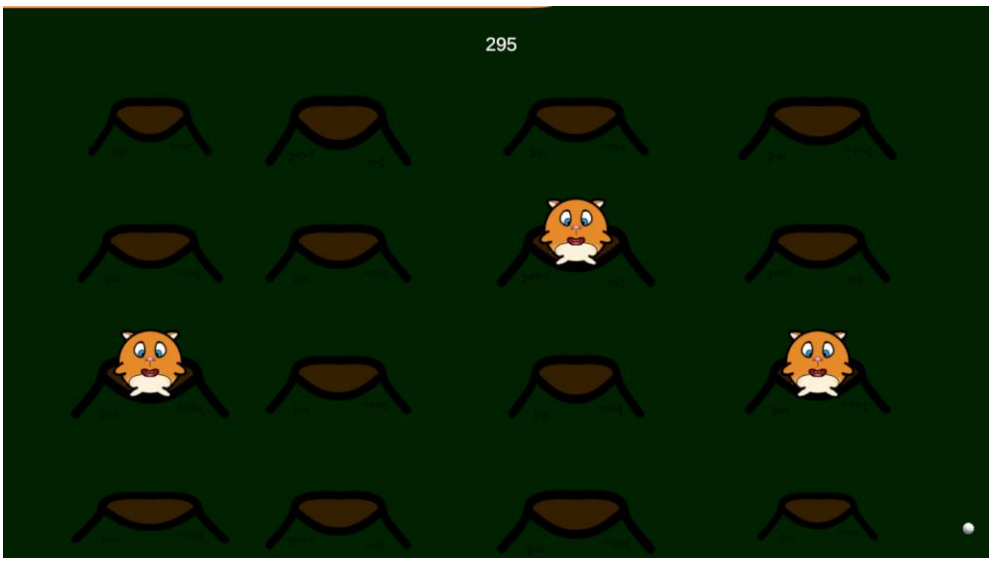



**Imagen 17 Demo topos**

Tras un tiempo predefinido, la partida finaliza y se muestra la puntuación final. Este tipo de juegos encaja en aulas de psicomotricidad de colegios y similares, donde se entrenaría con un ejercicio activo, divertido y ágil.

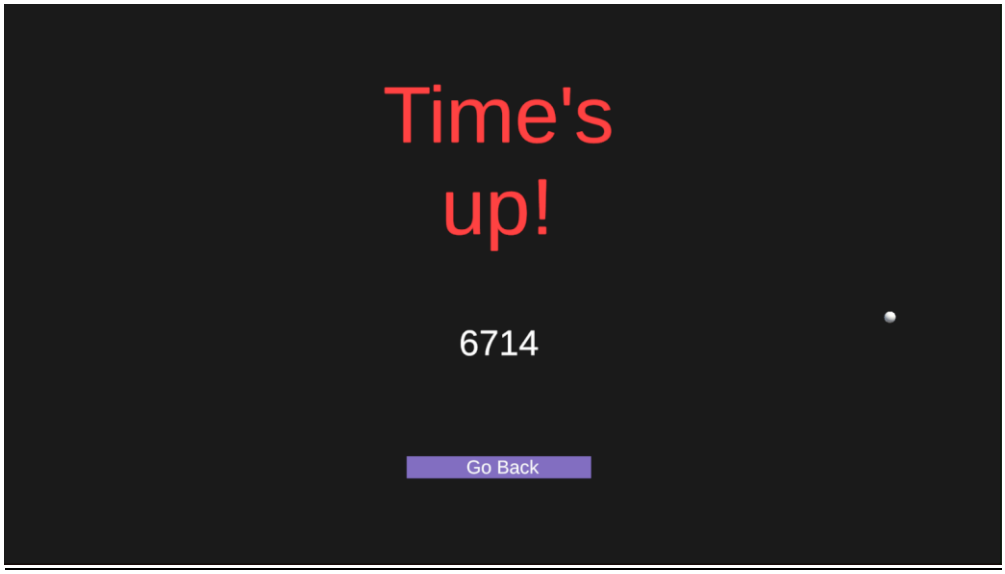



**Imagen 18 Game Over topos**

## 4.3. Fantasmas

Este minijuego se utiliza como muestra de la capacidad que tiene el *interactor* de cambiar en función de la experiencia que se quiera transmitir.

Aquí el *interactor* se convierte en una linterna, con esta linterna hay que buscar unos fantasmas que aparecen en una posición aleatoria de la pantalla. Para la realización de este proyecto, se cambia el *prefab* del *interactor* para que proyecte una luz sobre la escena. Esta es una de las posibilidades que permite una librería tan modificable.

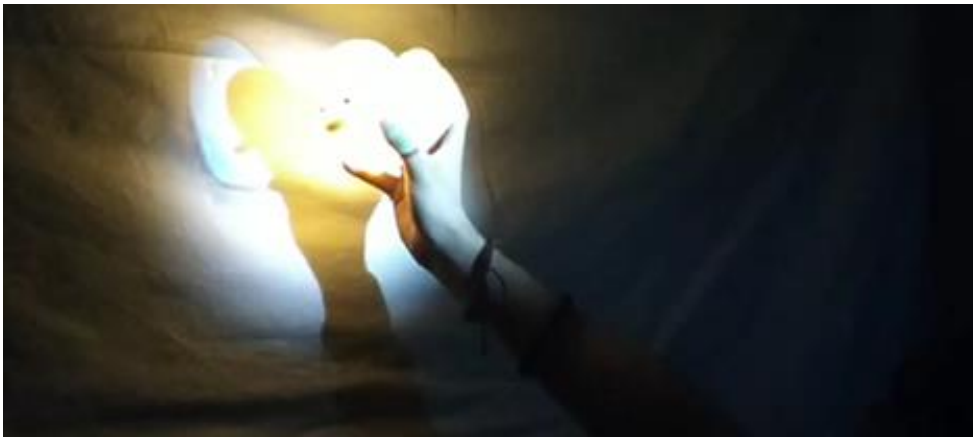


**Imagen 19 Demo fantasmas 1**

La pantalla se encuentra en completa oscuridad y se ilumina en las zonas según pasa el *interactor*, cuando se encuentra un fantasma, este desaparece y se marcha a otra posición de la pantalla, al cabo de un tiempo, el minijuego finaliza y se muestra la puntuación.

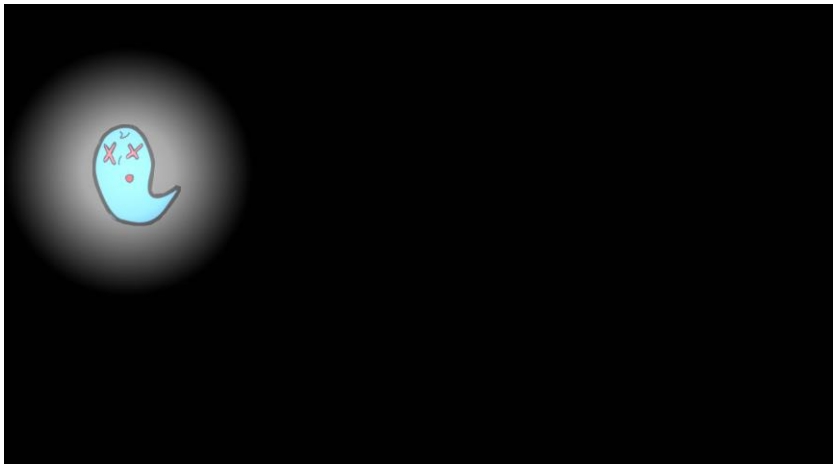


**Imagen 20 Demo fantasmas 2**

## 4.4. Recoge Frutas

Este minijuego consiste en recoger las frutas que van cayendo del cielo con una cesta que se controla con unos botones en la parte inferior de la pantalla. Todo ello mientras evitas las bombas que también caen del cielo.

**Imagen 21 Demo recoger frutas**

Este ejemplo permite revisar la capacidad de esta librería de desarrollar proyectos más estáticos en los que el movimiento se limita a tres puntos concretos.

# 5. Resultados

## 5.1. Publicación en Unity Asset Store

Para concluir, esta librería se ha publicado en la *Asset Store de Unity*, en la página del paquete se encuentra toda la información para su instalación y uso. [6]

Para ello se desarrolló una documentación específica para la descarga y se tuvo que completar el paquete de Unity de forma que cumpliese los estándares requeridos por *Unity* para ser aceptado en la tienda. Un extracto de la documentación se encuentra en el apartado de anexo. Una vez completados todos los requisitos, se envía a la plataforma y se inicia un periodo de revisión para ser aceptado o denegado como producto en la tienda.

Los requisitos de *Unity* consisten en unas pautas de estructura y de forma que implementar tanto dentro de cada uno de los códigos incluidos en el paquete como dentro de *Unity*. Entre otras, se pide explícitamente que todos los códigos se encuentren dentro de un *namespace*, esto es útil para poder diferenciar cada paquete, así como importarlos en otras clases únicamente importando el espacio de nombres.

También se requiere que dentro de la propia jerarquía del motor se encuentre un documento de texto que contenga la documentación y tutoriales para utilizar la librería, la documentación publicada se puede encontrar más adelante en el apartado anexo de este documento.

Finalmente, los requisitos exigen una estructura interna de la librería que contenga todo lo necesario en el interior de una carpeta raíz, y dentro de esa carpeta se encontrará todo el contenido desarrollado para el paquete y que se importará cuando se descargue desde la tienda. La librería dentro de *Unity* queda de la siguiente forma:

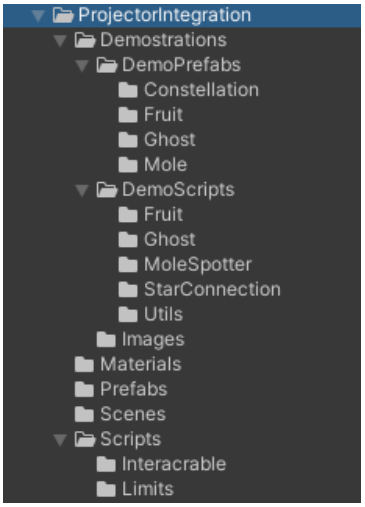



**Imagen 22 Estructura librería**

Una vez creada la librería, con todos los requisitos cumplidos, se debe crear una cuenta de desarrollador dentro de la tienda de *Unity*, esta cuenta utiliza la propia dirección de correo y nombre de la cuenta personal del motor, a la que se le añade un nombre profesional que aparecerá junto a los paquetes que se publiquen.

Este paquete se puede descargar de forma gratuita para todos los usuarios de *Unity* y listo para su uso una vez descargado. El paquete cuenta con todas las demostraciones nombradas anteriormente y con todos los objetos necesarios para su correcto funcionamiento.

En la propia página se encuentra toda la información del proyecto, así como las dependencias extras que requiere como, por ejemplo, la integración de *OpenCV* en *Unity*. [5]

## 5.2. Repositorio en GitHub

Asimismo, como parte de ser proyecto de código abierto, todo el código utilizado para el desarrollo de la librería se encuentra guardado en un repositorio público de *GitHub* [7], donde se puede apreciar todo el desarrollo, e incluso descargar el proyecto completo con el que se ha desarrollado este TFG.

El repositorio de *GitHub* permite dar una capa de transparencia al proyecto y podría permitir en un futuro un desarrollo más colaborativo donde los usuarios tomarían un papel más importante con la detección y corrección de errores, así como ampliación y mejora de la propia librería.

La utilización de GitHub en trabajos de desarrollo de *software* permite a los proyectos tener un control de versiones que guarda todos los cambios que se realizan desde el inicio al final del desarrollo. Y una vez finalizado el proyecto, permite tener una plataforma donde los usuarios y desarrolladores que usan la librería exponen sus problemas y errores con el uso y da pie a un mantenimiento colaborativo de la librería.

# 6. Conclusiones

## 6.1. Producto final

El producto final conseguido resulta en una librería pública de código abierto, que cualquier persona puede instalar en sus proyectos de *Unity* y utilizarlo al momento.

La publicación de la librería en la *Unity Asset Store* permite que el alcance del proyecto llegue a todos los usuarios de uno de los motores de desarrollo de videojuegos más utilizado en la industria.

Tiene una estructura suficientemente escalable como para aportar una capacidad de desarrollo con pocas limitaciones, las demostraciones nombradas anteriormente cubren las interacciones más básicas y unas modificaciones simples de las que partir para crear un producto completo y una experiencia inmersiva innovadora y atractiva.

Se ha conseguido crear una librería que permite el desarrollo de experiencias interactivas con un hardware totalmente genérico y accesible a la mayoría de entidades y desarrolladores, un hecho mejora enormemente el alcance que puede tener una tecnología como esta. Por otro lado, permite un espacio de trabajo libre y con la capacidad de mejorar en función de cada usuario.

## 6.2. Limitaciones

Se ha conseguido suplir todos los problemas que se planteaban al inicio del proyecto, sin embargo, cuenta con una serie de limitaciones que restringe el uso de la librería.

### 6.2.1. Hardware

Uno de los principales objetivos del desarrollo de esta tecnología era democratizar el uso y desarrollo de experiencias interactivas, y uno de las principales barreras se encontraba en el campo del *hardware* específico para este tipo de proyectos. Con esta librería, esa barrera desaparece, sin embargo, no por completo.

Para el correcto desarrollo de este tipo de interacciones, es imposible deshacerse por completo de la dependencia de hardware, es por eso que se ha minimizado al máximo el impacto del mismo en el proyecto. Sigue siendo una limitación, en algunos casos concretos determinante, que para su uso se requiera un proyector, una pequeña fuente de luz y un ordenador; pero para este tipo de tecnologías es la menor limitación posible.

Por otro lado, la implementación con base en un hardware totalmente genérico puede ser muy limitante para proyectos más ambiciosos, donde una tecnología más específica y desarrollada para un proyecto en particular otorgaría muchas más posibilidades de desarrollo.

### 6.2.2. Limitaciones físicas

Para el correcto uso de esta librería se necesita de un lugar suficientemente espacioso como para poder albergar todo el *hardware* necesario, un espacio de juego cómodo y un espacio en el que poder proyectar de forma efectiva el juego.

Por otra parte, debido a la naturaleza del cálculo de la interacción, para un mejor funcionamiento y unos mejores resultados de la librería, es necesario que esta estancia se encuentre con bastante oscuridad y con la menor interferencia lumínica posible debido a que la interacción depende de la luz.

Esta última limitación se puede reducir en la mayoría de los casos con la capacidad de aumentar el umbral a partir del cual se calculan las interacciones, sin embargo, siempre será mejor un espacio oscuro para el desarrollo de las experiencias inmersivas desarrolladas con esta tecnología.

## 6.2. Trabajos futuros

El desarrollo de este tipo de tecnologías se plantea como una visión optimista hacia el desarrollo e impulso de *Serious Games* y proyectos interactivos culturales que buscan enriquecer su contenido y hacerlo más accesible y atractivo para su público. La implementación de esta tecnología en entornos con recursos limitados, como pequeñas exposiciones y aulas escolares, puede abrir un mundo de posibilidades para la educación interactiva y la divulgación cultural.

La capacidad de transformar cualquier superficie en una interfaz interactiva mediante el uso de un hardware genérico y accesible puede revolucionar la forma en que se presentan y se interactúa con los contenidos. Esto puede ser especialmente útil en aulas, donde los recursos pueden ser limitados, pero la necesidad de involucrar a los estudiantes en su aprendizaje es primordial.

Además, esta tecnología puede ser una herramienta valiosa para las pequeñas exposiciones y museos que buscan ofrecer experiencias interactivas a sus visitantes, pero que pueden no tener los recursos para invertir en hardware costoso. Con esta librería de *Unity*, pueden crear

exposiciones interactivas que atraigan a los visitantes y les permitan interactuar con las exhibiciones de una manera completamente nueva.

# 7. Bibliografía


[1] Lü, «Play Lü,» [En línea]. Available: https://play-lu.com/. [Último acceso: Mayo 2024].

[2] S. Retalis, «How Schools are Overcoming the Challenges of Using Assistive Technology for Children with Multiple Disabilities,» 4 Enero 2022. [En línea]. Available: https://blog.kinems.com/how-schools-are-overcoming-the-challenges-of-using-assistive-technology-for-children-with-multiple-disabilities/. [Último acceso: Marzo 2024].

[3] Microsoft, «Learn Microsoft - Kinect para Windows,» 14 07 2023. [En línea]. Available: https://learn.microsoft.com/es-es/windows/apps/design/devices/kinect-for-windows. [Último acceso: Mayo 2024].

[4] OpenCV, «OpenCV,» 2024. [En línea]. Available: https://opencv.org/. [Último acceso: Mayo 2024].

[5] Paper Plane Tools, «Asset Store Unity,» Paper Plane Tools, 24 Enero 2019. [En línea]. Available: https://assetstore.unity.com/packages/tools/integration/opencv-plus-unity-85928. [Último acceso: 2023].

[6] D. López, «Projector Integration - Asset Store Unity,» 20 Mayo 2024. [En línea]. Available: https://u3d.as/3i7c.

[7] D. López, «Projector Integration Unity - Github,» 20 Mayo 2024. [En línea]. Available: https://github.com/david-3lm/ProjectorIntegrationUnity.

# 8. Anexo

Aquí se encuentra parte de la documentación desarrollada para la librería, contiene un tutorial más específico de cómo crear una escena y preparar su funcionalidad

## ¿Cómo funciona?

Primero de todo, se requiere una versión instalada de Unity, la librería está desarrollada en la versión 2021.3.10f1 del editor, pero cualquier versión de ahí en adelante debería funcionar correctamente.

Al instalar la librería habrá una carpeta con algunos *prefabs*.

En cuanto a su funcionamiento interno, podemos dividirlo en 3 grandes partes:

### Detección de la cámara

Al principio de la ejecución será dirigido a la escena de calibración, tendrá cuadrados verdes en las esquinas. Con ayuda de una máscara de color, se utiliza para calibrar los límites de la proyección y así conseguir estimar la posición del *interactor*.

### Contour Finder

Este apartado usa la librería de *OpenCV* para conseguir la imagen de la webcam y procesar la imagen, sigue estos pasos:

- Se toma la imagen de la cámara, sin ninguna modificación.
- Tras eso, se modifica la imagen utilizando un umbral y se procesa la imagen a blanco y negro, para conseguir que sea más fácil de calcular y analizar por el ordenador.
- Con esa información, el script analiza las zonas negras de la imagen procesada, las cuales corresponden con zonas iluminadas, y por tanto con el *interactor* luminoso que se utilizará. Estas zonas están divididas en diferentes contornos con *OpenCV* y se define si cumple los requisitos para ser considerado *interactor* (área y si se encuentra dentro de los límites).

### Interactuables

Se utilizan tres clases para conseguir una interacción funcional, estas serán las bases para cualquier objeto interactivo dentro del juego.

Se utiliza el sistema de Unity de colisiones y triggers para desarrollar esta funcionalidad.

***Interactor***: esta clase toma la información de *ContourFinder* y el centro del área negra más grande de la imagen procesada en el script como el centro del *collider*. Este objeto es un *GameObject* vacío con un *Collider* y un *RigidBody* que se mueve a través de la escena.

Este objeto será el que interactúe con los *trigger* de los objetos interactivos.

***InteractiveObject***: esta clase abstracta consiste de un método abstracto llamado *InteractionEvent* que será llamado dentro de su *OnTriggerEnter*. Todo objeto que se quiera que tenga una interacción deberá heredar de esta clase y tener un *Collider* en el *GameObject* que lo contenga.

***InteractiveButton***: esta clase hereda de *InteractiveObject* y desarrolla el comportamiento de un botón clásico, con un evento *OnClick*.

## ¿Cómo usarlo?

Para usar la librería de forma efectiva hace falta un proyector, una cámara y una fuente de luz (algo como una pequeña lámpara o bombilla) que servirá como *interactor*.

Es necesario colocar el proyector en una habitación con poca iluminación para que la interacción funcione de la mejor forma posible.

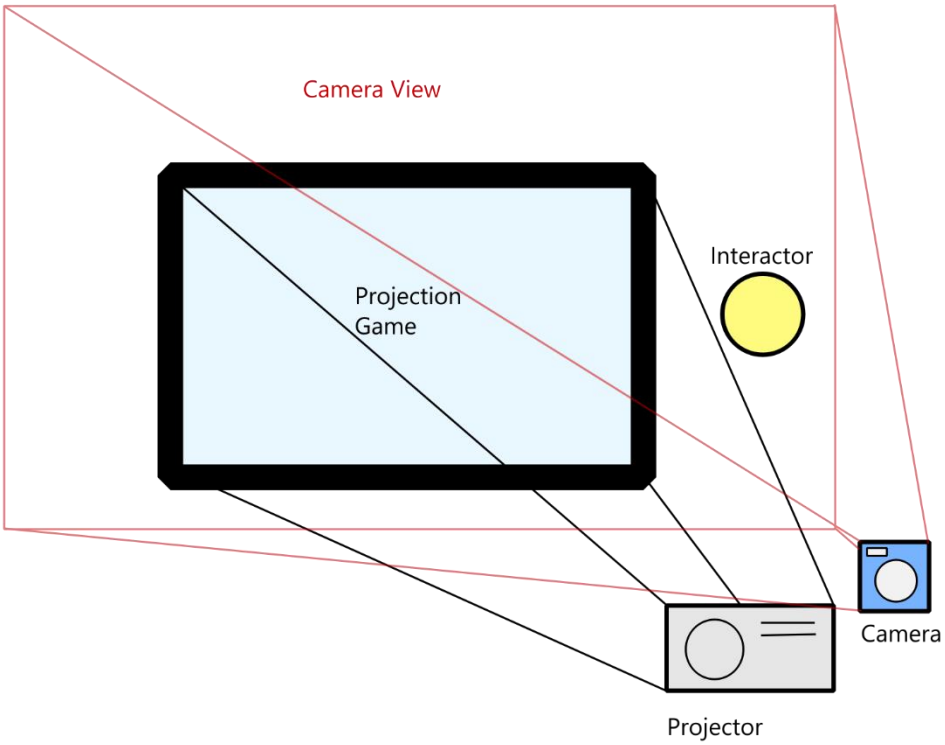
La cámara se debe colocar apuntando hacia la proyección (e idealmente cerca del proyector para una interacción más intuitiva) de forma que contenga toda la pantalla, no hace falta ajustar el tamaño porque ya se calibrará internamente la imagen para conseguir una calibración correcta.

En cuanto al desarrollo de los Scripts y de los objetos, todo objeto que queramos que sea interactivo, deberá heredar de la función *InteractiveObject* y definir un comportamiento específico a la función *InteractionEvent*.

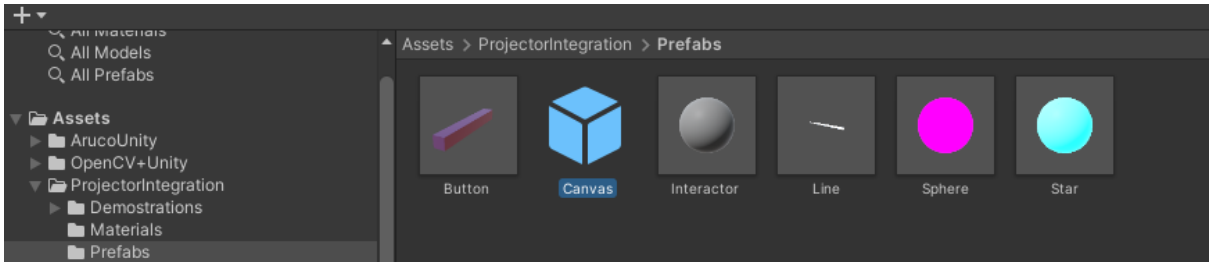
Ese script que se crea deberá colocarse en el objeto de Unity como componente y ya estaría listo para interactuar con el *interactor*.

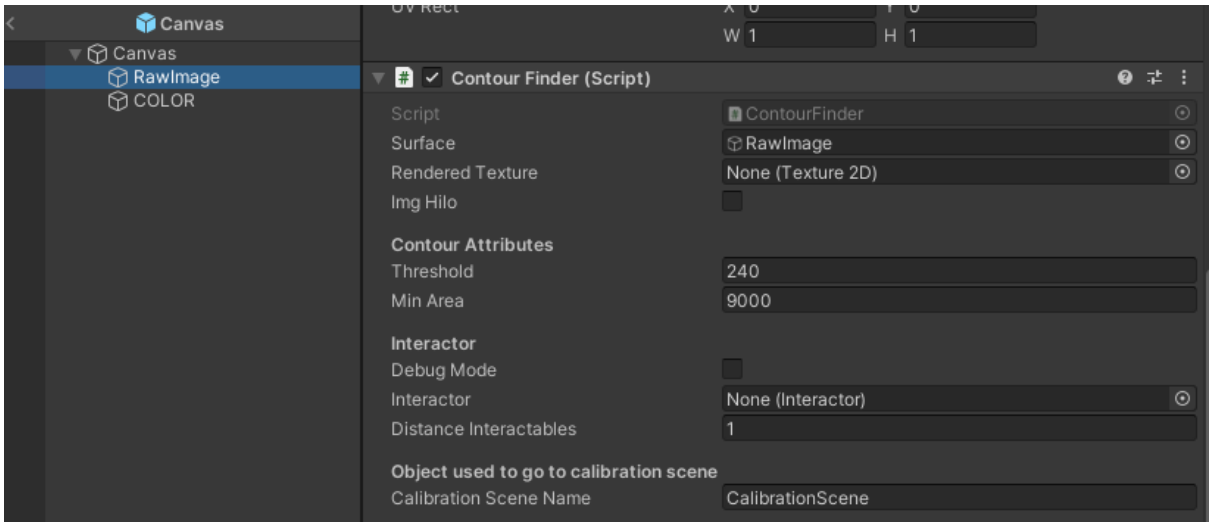
# Configuración

Debes preparar el proyector y la cámara apuntando hacia la imagen creada, cuanto más oscura se encuentre la estancia más sencilla será para la librería diferenciar interacciones.




Al instalar la librería existe una carpeta con ***prefabs***, en ella se encontrará el *canvas* utilizado para la interacción, en ese *canvas* se encuentra una *RawImage* que contendrá el script *ContourFinder*, éste es el que se encarga de la interacción.






Al iniciar el juego, se necesita calibrar, y para ello existe *CalibrationScene*, la cual no necesita ningún tipo de edición, utiliza una máscara de color verde para calibrar los límites de la proyección y así calcular la interacción en *ContourFinder* más adelante.

Una vez colocado el *canvas*, para crear objetos interactivos hay que añadir a sus scripts la herencia de *InteractableObject* y redefinir su *InteractionEvent* para conseguir el efecto deseado.

Finalmente hay que tomar el *prefab* de *Interactor* encontrado en la misma carpeta nombrada anteriormente y añadirlo a la escena en la que quieres que haya interacción.