UNIVERSITAT POLITÈCNICA DE VALÈNCIA

MIARFID

Departamento de Sistemas Informáticos y Computación

Universitat Politècnica de València

Integración dinámica de objetos sintéticos en escenas reales

Trabajo Fin de Máster

Máster en Inteligencia Artificial, Reconocimiento de Formas e Imagen Digital

Autor: Esther María Sánchez García

Tutor: Francisco José Abad Cerdá

2018/2019

Resumen

Desde los comienzos de la Realidad Aumentada se ha intentado conseguir una inclusión convincente de los elementos virtuales en las escenas reales de forma que un usuario no sea capaz de distinguir si los objetos sintéticos son reales o no. Para conseguir este objetivo, se abordan tres cuestiones diferentes, que el objeto tenga coherencia temporal, coherencia espacial y coherencia lumínica. En el presente Trabajo Final de Máster se pretende realizar un estudio de los métodos que existen en la actualidad que abordan el problema de la inconsistencia lumínica aportando finalmente un sistema que avanza en la resolución de este problema, mediante la técnica de *Iluminación basada en Imagen*.

Palabras clave: Realidad Aumentada, Iluminación en Gráficos, Objetos sintéticos, Coherencia lumínica, Iluminación basada en imagen.

Abstract

Since the beginning of the Augmented Reality, we have tried to achieve a complete inclusión of the vitual elements in the real scenes so that a user in no table to distinguish whether the synthetic objects are real or not. To achieve this goal, three diferents issues are addressed, the object has temporal coherence, spatial coherence and luminance coherence. In the present Final Master's Project we intend to carry out a study of the methods that currently exist that address the problema of the light inconsistency, finally providing a system that aims to solve this problema through Image Based Lighting

Keywords : Augmented Reality, Ilumination, Synthetic Objects, Luminance Coherence.

Agradecimientos

Agradecer en primer lugar a mi tutor, Paco, por su ayuda y dedicación ya que sin él esto no sería posible. Por responder a todas mis dudas en tiempo récord y saber encauzarme en el proyecto cuando yo no veía la luz.

A mis padres, Elena y Ramón y a mis hermanas Tamara y Eli ya que a pesar de la distancia he sentido su enorme apoyo y ánimos.

A mis amigos Jesu, Raquel y Adri por sus comentarios, apoyo y granitos de arena aportados y en especial a Kevin por saber aguantarme, no es buena compañía alguien centrado en su TFM.

Un sitio para cada cosa, cada cosa en su sitio.

Tabla de contenidos

Índice de Figuras

1. Introducción

1.1. Motivación

Desde que se generaron las primeras imágenes por ordenador, uno de los objetivos que se han perseguido, es el de conseguir que los elementos virtuales parezcan lo más auténticos posibles. Un campo que ha generado un gran interés en la industria del cine, la televisión o la realidad aumentada, es el de la inclusión de objetos sintéticos en escenas reales, de forma que, un usuario no consiga distinguir entre lo que es real y lo que ha sido generado por ordenador. De aquí surgen varias vertientes a estudio, que hacen que un objeto sintético parezca inmerso en un entorno real:

- La coherencia espacial → Los objetos virtuales deben de ajustarse a la perspectiva de la escena real.
- La coherencia lumínica → Los objetos sintéticos deben de presentar una iluminación acorde con la de la escena real (reflexiones y proyecciones de sombras como si estuviesen en el entorno)
- La coherencia temporal → Los objetos sintéticos se deben de mover y coordinar adecuadamente con el entorno.

Este Trabajo Final de Máster se centra en mejorar uno de esos aspectos: el de la coherencia lumínica de los objetos sintéticos en tiempo real. Los objetos sintéticos deberán de presentar una iluminación acorde con la del entorno real de forma dinámica, de manera que, si una fuente de luz cambia alguna de sus características (como la posición, color o intensidad con la que emite la luz), los cambios deberán de verse reflejados en los elementos virtuales en tiempo real.
Basándonos en la técnica de iluminación basada en imagen que consiste en usar una imagen panorámica de la escena como mapa de entorno (imagen que se mapea en una esfera que rodea a la escena virtual), se propone un sistema capaz de iluminar los objetos virtuales generando sombras y luces dependiendo de las condiciones lumínicas del entorno en tiempo real.

1.2. Objetivos

El objetivo principal de este trabajo es el análisis, desarrollo e implementación de un sistema que sea capaz de iluminar objetos sintéticos ubicados en escenas reales de forma que tengan una iluminación coherente con el entorno y en tiempo real. Para conseguir dicho objetivo, se han seguido los siguientes pasos:

- Estudiar los campos donde se incluyen objetos sintéticos en entornos reales.
- Realizar un estudio de los diferentes métodos que existen de iluminación por ordenador centrándonos en la técnica de la iluminación basada en imagen.
- Implementar una solución que sea capaz de iluminar los objetos virtuales generando sombras y reflexiones dependiendo de las condiciones lumínicas del entorno y que funcione en tiempo real.
- Realizar una serie de pruebas para sacar conclusiones de la solución presentada.

1.3. Descripción de la solución propuesta

El sistema que se propone deberá de generar objetos sintéticos en entornos reales que tengan una iluminación acorde con la del entorno. Esto quiere decir que los elementos virtuales deberán de generar sombras en función de las luces que se encuentran en la escena real y deberán de reflejarla.

Para capturar el entorno, se hace uso de una cámara de 360º de bajo coste que permite recoger información panorámica de la escena en tiempo real. La captura devuelta por la cámara es una imagen de Bajo Rango Dinámico (LDR) que puede devolver zonas subexpuestas o sobrexpuestas en la captura. De esta forma, se debe de hacer una conversión a una imagen de Alto Rango Dinámico (HDR) ya que representa con más exactitud el extenso rango de intensidad de la escena real. La conversión se realiza utilizando una técnica de Inverse Tone Mapping.

La imagen HDR servirá para dos cosas: como mapa de entorno y como imagen para detectar las fuentes de luz. El mapa de entorno será el que aporte a los objetos sintéticos reflejos, tanto especulares como difusos. Por otro lado, para poder proyectar sombras que sean acordes con las que se proyectarían en la realidad, es necesario detectar en la imagen HDR las fuentes de luz reales y colocar en el escenario virtual fuentes de luz sintéticas. De esta forma, las fuentes virtuales estarán ubicadas en los puntos de donde

proviene la luz en el entorno real y por lo tanto los objetos virtuales proyectarán la misma sombra que proyectarían los objetos reales en ese entorno.

Una vez se han iluminado los objetos sintéticos estos se representan en el mundo real. Cabe señalar que tanto las fuentes de luz sintéticas como el mapa de entorno no se representan en la imagen final si no que estos solo sirven para iluminar los objetos que serán ubicados en el entorno real. En la imagen de la Figura 1 se puede ver un esquema de la solución propuesta.

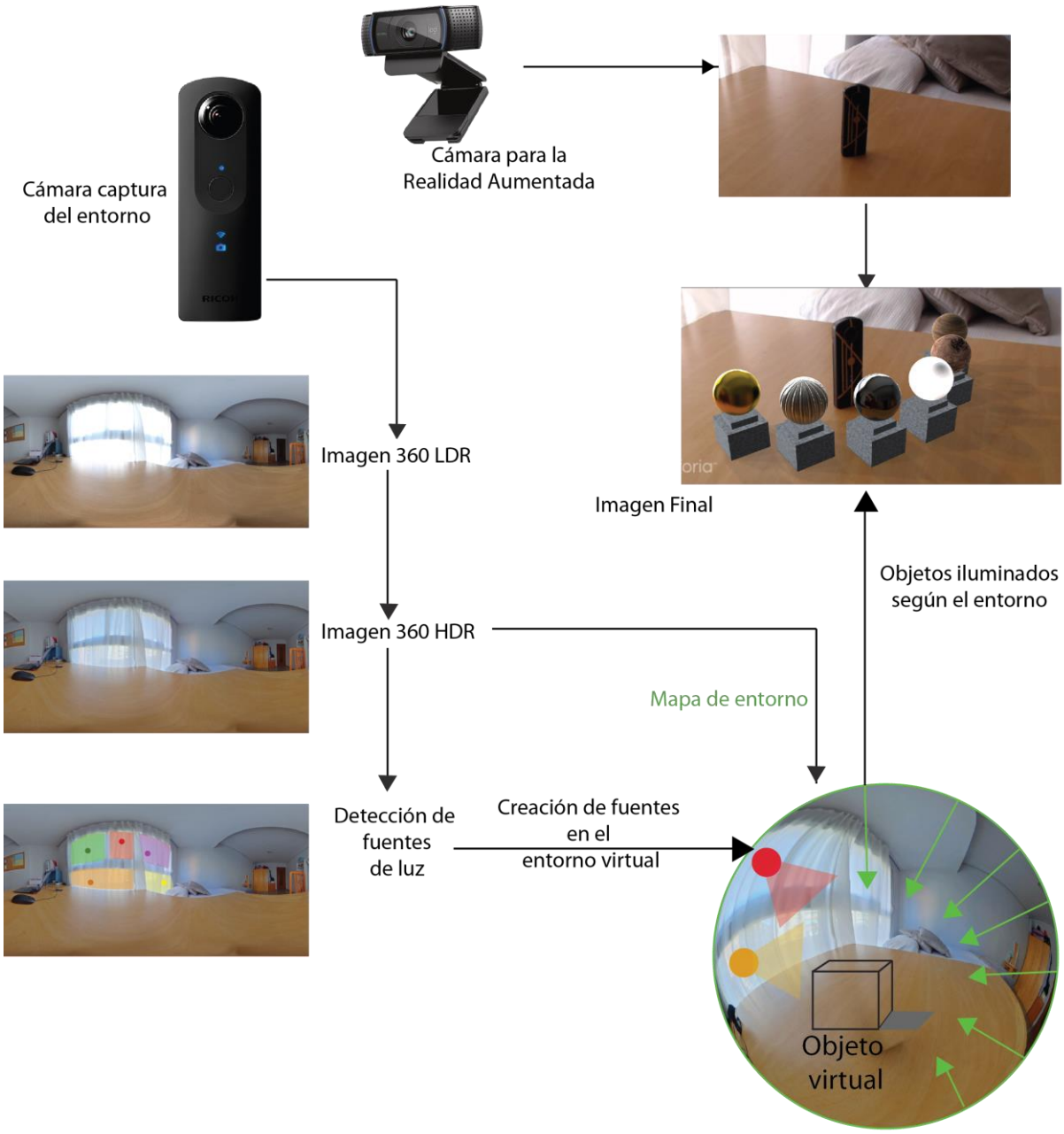


Figura 1. Esquema del sistema propuesto para iluminar objetos sintéticos que se situarán en entornos reales

1.4. Estructura de la memoria

El documento se encuentra estructurado en las siguientes siete partes:

- Introducción → En este apartado se describe la motivación del proyecto, los objetivos, así como una breve descripción de la solución propuesta y la estructura de la memoria.
- Estado del Arte → Aquí se hace un recorrido por los diferentes campos que han incluido objetos sintéticos en escena reales, centrándonos en el campo de la Realidad Aumentada, ya que al final se utiliza en el sistema. A continuación, se realiza un análisis de los métodos y tecnologías existentes de iluminación por gráficos por ordenador para finalmente hablar de la técnica de Iluminación basada en Imagen y hacer una crítica al Estado del Arte.
- Tecnologías empleadas → Se detallan las tecnologías, así como los dispositivos utilizados a lo largo de la elaboración del proyecto.
- Diseño e Implementación de la solución → En esta sección se define la solución propuesta y se especifican algunos detalles de la implementación. En esta sección también se presentan algunas pruebas que se han realizado para ajustar determinados parámetros del sistema.
- Resultados→ Se presentan una serie de imágenes y videos demostrativos para mostrar el correcto funcionamiento de la solución propuesta.
- Conclusiones → En este apartado, se comentan algunas reflexiones sobre los objetivos que se han conseguido con la elaboración de este trabajo.
- Trabajos Futuros → Basándonos en los puntos anteriores de la memoria, se detallan posibles mejoras en este TFM.
- Anexos → Es esta sección se añade información adicional relacionada con algunos aspectos del trabajo, así como algunos resultados de las pruebas realizadas para ajustar parámetros.

2. Estado del Arte

Con el fin de conseguir imágenes generadas por computador que no se puedan distinguir de videos o fotografías de una escena real, la Informática Gráfica se ha centrado desde sus inicios en generar imágenes sintéticas fotorrealistas. La búsqueda del realismo visual de las imágenes construidas de forma sintética se ha establecido en multitud de campos como los del cine, la televisión, la publicidad, los videojuegos o la realidad aumentada.

Uno de los objetivos principales en la creación de imágenes sintéticas, es lograr simular de forma verosímil las complejas interacciones entre los objetos de una escena real y las fuentes que los iluminan ya que la correcta iluminación es un requisito básico para alcanzar dichos niveles de realismo. Con este objetivo, se han desarrollado diferentes modelos de iluminación que simulan la propagación de la luz. Un método que se ha desarrollado para conseguir dicho realismo es el de la visualización basada en imagen (Image Based Rendering) que consiste en generar imágenes sintéticas a través de imágenes tomadas de una escena real.

De esta forma, en los siguientes apartados se presentan algunos campos como el cine, la televisión y la realidad aumentada para demostrar la integración realista de objetos sintéticos en imágenes reales. Dado que el Trabajo se desarrolla para una pequeña aplicación de Realidad Aumentada, este apartado es más extenso. Se expondrán los algoritmos y técnicas más comunes. Por último, se presenta una crítica al estado del arte donde, analizando lo expuesto anteriormente, se determinan las bases de las que parte este TFM.

2.1. Imágenes Sintéticas en el cine y la televisión

La Informática Gráfica en la industria del cine, se ha centrado en desarrollar algoritmos y software para la creación de los efectos especiales, entendiéndose estos como el conjunto de técnicas que crean personajes o escenarios que no existen en la realidad o que no se pueden grabar.

El precursor de dichas técnicas fue Alfred Clark quien, en 1895, demostró que una actriz podía ser decapitada sin perder la cabeza en la película "*The Execution of Mary Queen of Scots*". En ella utilizó el método que luego George Meliès utilizaría en las películas "*Viaje a la Luna*" o "*El viaje imposible*", el "Stop Trick". En este caso, lo que

realizó fue parar la cámara en el momento de la ejecución, cambiar a la actriz por un muñeco y seguir rodando justo cuando un hacha le corta la cabeza. A partir de aquí, empiezan a surgir otras técnicas como el "Matte Shot" donde se componían en la misma cámara dos tomas diferentes: se cubre una parte del objetivo con una máscara opaca durante la filmación de la primera toma, se rebobina la película, se descubre la zona que ya estaba cubierta y se filma la segunda toma. Cabe destacar que dicha técnica dio paso posteriormente al "Chroma Key", técnica usada hoy en día, que consiste en rodar una acción o actores sobre un fondo monocromático que luego es sustituido por un paisaje o escenario. Otras técnicas interesantes fueron el "Stop Motion" (se tomaban sucesivas fotografías donde en cada plano se variaba ligeramente una miniatura y que a la hora de componerlo daba la sensación de movimiento) o el "Optical Printing" (donde se usaba un dispositivo que consistía en una cámara que tenía enlazado un proyector y que permitía combinar múltiples elementos de diferentes tomas). [1]

Sin embargo, no fue hasta la década de los 80 donde las imágenes generadas por ordenador (CGI) empiezan a cobrar importancia en las producciones cinematográficas. Las dos películas en las que el CGI intervino por primera vez fueron "*Tron"*, en 1982 y en "*Last Starfighter"* (1984) aunque el primer personaje creado mediante una computadora fue por Pixar para la película "Las aventuras del joven Sherlock Holmes". No obstante, todas estas películas no lograron que el CGI persuadiera a la industria del cine, hasta el año 1989, cuando "*The Abyss*" ganó el Premio de la Academia de Hollywood en la categoría de Efectos Visuales gracias a la escena en la que una criatura de agua imitaba la cara de la protagonista. Posteriormente películas como "*Terminator 2: Judgment Day*" de 1991 y "Jurassic Park" de 1993, cambiaron radicalmente la percepción de la industria, ya que consiguieron integrar muy bien personajes ficticios en secuencias reales y viceversa. Cabe señalar, que muchas de las incrustaciones que se hacen de personajes reales en escenarios fantásticos, es mediante la técnica del Chroma Key donde el fondo es sustituido por un escenario generado digitalmente. [2]

A principios del 2000, el CGI domina el campo de los efectos especiales ya que las imágenes sintéticas realistas lograron impactar al público por salvar la brecha entre la fantasía y la realidad abriendo puertas para crear narraciones, personajes y mundos que de otra manera no podrían haberse logrado. La tecnología ha avanzado hasta tal punto que es posible sustituir digitalmente a los actores reales por actores virtuales como en "*Rogue One: una historia de Star Wars*" donde el actor Peter Cushing que ya había fallecido antes de que se rodara la película, fue sustituido por CGI. En la imagen de la Figura 2 se puede ver a la izquierda al actor real y a la derecha el generado por ordenador.

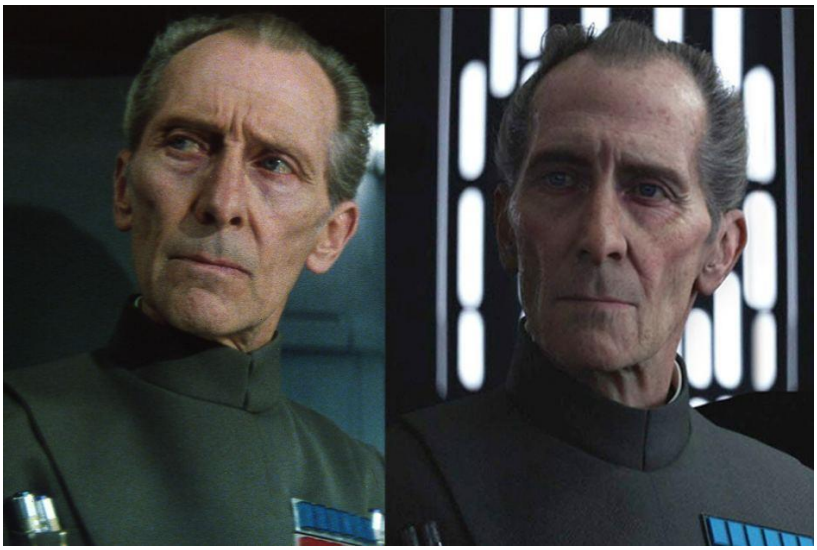


Figura 2. El actor Peter Cushing a la izquierda y a la derecha su doble generado por ordenador [1]

Cabe señalar que uno de los problemas que se tenían a la hora de animar personajes era la capacidad de dotar movimientos y gestos en la cara que fueran suficientemente reales. Para ello se creó la técnica del "Motion Capture" que consiste en grabar el movimiento de los gestos de la cara para luego trasladarlo a un modelo 3D y que por ejemplo fue usada por James Cameron en la película de "*Avatar*" [3]. Hay varias películas que han sido nombradas y galardonadas estos últimos años por la generación de un CGI espectacular. Ejemplo de ello son "*Gravity*", donde los actores están inmersos en el espacio o "*La Vida de Pi*" donde los animales que acompañan al personaje en el naufragio parecen reales.

En el mundo de la Televisión, los efectos especiales han ido a la par con los de la industria cinematográfica. En este caso, sobre todo se han centrado en generar escenarios realistas donde poder situar a presentadores o concursantes mediante la técnica del "Chroma Key". Otra vertiente del CGI en los programas de difusión como lo son las noticias, está en los gráficos con los que el presentador interactúa para mostrar información o datos. [4] Un ejemplo es el software que provee la compañía Brainstorm multimedia que permiten crear gráficos 3D para mostrar los resultados electorales de forma que sean más llamativos para el espectador. En la Figura 3 se puede ver un ejemplo del software usado por la cadena de Televisión Antena 3 para mostrar los resultados electorales. A pesar de que estas son las vertientes que más se han utilizado

[1] Imagen obtenida de: *https://www.abc.net.au/news/2017-01-19/real-tarkin-vs-rogue-one-cgi-tarkin/8194824*

en lo que efectos especiales en televisión se refiere, ya se están generando otro tipo de efectos como por ejemplo, en el programa de noticias Chino de la compañía Xinhua, donde el presentador es totalmente virtual y mueve la boca en función de la noticia que debe leer.

Figura 3. Gráficos generados por la compañía Brainstorm para presentar los datos electorales en Antena 3 Noticias[2]

Cabe destacar que hemos entrado en una generación en la que si se dispone de un elevado presupuesto es posible hacer que la fantasía ocupe el lugar de la realidad en la pantalla. Sin embargo, el coste medio tan elevado que supone crear CGI en el cine y la televisión hacen que sean pocos los contenidos generados por ordenador que parezcan reales. De esta forma, se siguen haciendo progresos en las CGI que van altamente acompañadas de los avances que se hacen en la Informática Gráfica y que mejoran cada vez más con el objetivo de conseguir el máximo realismo posible, con costes inferiores y que se puedan generar en tiempo real para evitar el coste añadido de la posproducción.

2.2. Imágenes Sintéticas en la Realidad Aumentada

Un campo donde ha supuesto un gran interés generar imágenes sintéticas fotorrealistas es la Realidad Aumentada. La Realidad Aumentada es el término que se usa para definir la inclusión, en tiempo real, de elementos virtuales dentro del mundo físico de forma que el usuario ve a través de una pantalla el mundo real y superpuesto visualiza objetos virtuales. El responsable de acuñar el término de Realidad Aumentada

[2] Imagen obtenida de: http://www.brainstorm3d.com/solutions/elections-graphics/

fue Tom Caudell en 1992, que por aquel entonces trabajaba como investigador de "Boening" donde logró superponer textos, diagramas y gráficos en el visor que utilizaban los operarios para revisar varias partes de un avión. Hasta ese entonces la Realidad Aumentada no se distinguía de la Realidad Virtual entendiéndose esta última como la sensación de inclusión de una persona en un entorno completamente virtual. [5]

2.2.1. Aplicaciones de Realidad Aumentada

Actualmente, existen numerosas aplicaciones de Realidad Aumentada, que abarcan desde industrias como la automovilística, la medicina o la educación hasta industrias que ofrecen contenido multimedia para el entretenimiento como pueden ser el cine o la televisión, el marketing y los videojuegos.

En la industria automovilística se están generando una gran multitud de aplicaciones tanto para los conductores como para los fabricantes o mecánicos. Un ejemplo de aplicación para los conductores es la que pretende implementar Continental 1 (proveedor de componentes para automóviles), empresa que lleva tiempo desarrollando dispositivos de HUD basados en Realidad Aumentada (proyección de información sobre el parabrisas). Este sistema consiste en paneles transparentes colocados en el cristal delantero del automóvil donde se proyectan imágenes sintéticas en perspectiva para ofrecer al conductor información sobre la carretera. [3]

En medicina existen métodos como los rayos X, la tomografía computerizada (TC), las resonancias… que brindan a los médicos la posibilidad de ver órganos internos de los pacientes, hacer diagnósticos y planificar cirugías. Actualmente existen estudios que investigan la posibilidad de utilizar la realidad aumentada, junto con otras tecnologías para evitar realizar una inspección invasiva en los pacientes y ayudar a las personas que trabajan en el sector sanitario a localizar órganos, huesos.... Un ejemplo de ello es el sistema que están desarrollando en la Universidad de Alberta, donde se incorpora un proyector, cámaras infrarrojas y marcadores que se colocan en ubicaciones estratégicas de los cuerpos de los pacientes, para proyectar imágenes en la piel que muestren la ubicación concreta de huesos, músculos u órganos. En esta aplicación sería de vital importancia que exista un registro fino, es decir que el lugar donde se colocan los órganos generados sintéticamente se corresponda con el lugar donde se encuentran realmente situados [6].

[3] Continental the Future in Motion, «AR HUD» [En línea]. Available: http://continental-head-up-display.com/ar-hud/. [Último acceso: 2018 Septiembre 06].

Por otro lado, también se han desarrollado multitud de aplicaciones en el ámbito de la educación. En algunas de ellas, es importante que los gráficos se muestren lo más fotorrealistas posibles para que el aprendizaje mediante objetos virtuales se corresponda de forma precisa con los objetos de la realidad. En estos casos, aplicaciones desarrolladas para estudiantes de áreas relacionadas con la ingeniería o la medicina, se basan en ofrecer elementos gráficos con gran parecido a los elementos de la realidad. Ejemplo de ello puede ser la aplicación 4DAnatomy[4] que muestra partes de la anatomía humana de forma realista.

En la industria del marketing o la publicidad se precisa que ciertos elementos parezcan que están realmente incluidos en la escena. Ejemplo de ello puede ser la ya conocida y difundida aplicación de IKEA[5] donde se permite a los usuarios colocar muebles virtuales (idénticos a los productos reales) en el entorno donde le gustaría posicionarlos una vez adquiridos. La finalidad de la aplicación es ofrecer a los usuarios una herramienta para visualizar en la distribución de la estancia un determinado mueble y evitar posibles devoluciones futuras. Obviamente, cuanto más realistas sean los muebles virtuales y su presencia en la escena real, más se corresponderán con el resultado final. En la Figura 4 se muestra un ejemplo de una mesa sintética correspondiente a la que ofrece la compañía colocada en una escena real.

[4] 4D Interactive Anatomy, «4DAnatomy,» [En línea]. Available: https://www.4danatomy.com/. [Último acceso: 2018 Septiembre 06].

[5] IKEA, «Ikea place aumented,» [En línea]. Available: https://highlights.ikea.com/2017/ikea-place/. [Último acceso: 2018 Septiembre 06].

Figura 4. Ejemplo de la aplicación de RA de la tienda IKEA[6]

Otro campo donde siempre se intenta mejorar la calidad gráfica de forma que esta sea lo más realista posible es en los videojuegos. Con el continuo crecimiento de la Realidad Aumentada, se han comenzado a desarrollar videojuegos con esta tecnología. La aplicación de PokemonGo[7] es un ejemplo de ello. Es cierto que, a la hora de ubicar a los Pokemon en la escena, no se hace teniendo en cuenta el entorno, pero Niantic, la empresa responsable de dicho juego, ya ha publicado que en futuras actualizaciones los personajes del juego se situarán teniendo en cuenta las superficies y que además estos tendrán un aspecto mucho más realista.

Por otro lado, cabe destacar que la Realidad Aumentada podría ser una gran solución a los efectos especiales ya que supondrían la generación de contenido sintético en escenas reales en tiempo real y se podrían ahorrar los elevados costes de la postproducción. Hay empresas como la mencionada anteriormente, Brainstorm Multimedia que ya ofertan paquetes de software que genera gráficos de realidad aumentada en televisión como los mostrados en la Figura 3.

2.2.2. Tecnología de los dispositivos de Realidad Aumentada

Respecto a la tecnología de los dispositivos que usa la Realidad Aumentada, puede ser de dos tipos: óptica o vídeo. La tecnología óptica se basa en la utilización de visores que permiten la visión a través de ellos del entorno real y superponer directamente los

[6] Imagen obtenida de: https://mobile-ar.reality.news/news/apple-ar-houzz-arkit-app-beats-ikea-app-store-0180132/

[7] Niantic, «PokemonGo App,» [En línea]. Available: https://pokemongolive.com/es/. [Último acceso: 2018 Septiembre 06].

objetos sintéticos. Un ejemplo puede ser las Google Glass o el HUD que pretende implementar Continental en los automóviles. En la Figura 5 se puede ver de forma esquematizada dicho tipo de tecnologías.

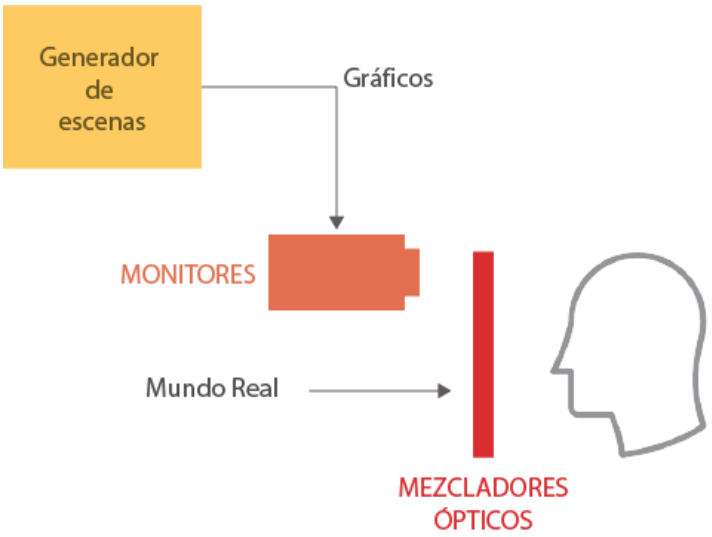


Figura 5. Esquema de la tecnología óptica en AR

Por el contrario, los dispositivos de Realidad Aumentada basados en vídeo no permiten ninguna visión del exterior y por lo tanto capturan el entorno, normalmente por cámaras de video y luego lo fusionan con los elementos virtuales, mostrando el resultado en un monitor o pantalla [7]. En la Figura 6 se puede ver de forma esquematizada los componentes principales de esta tecnología.

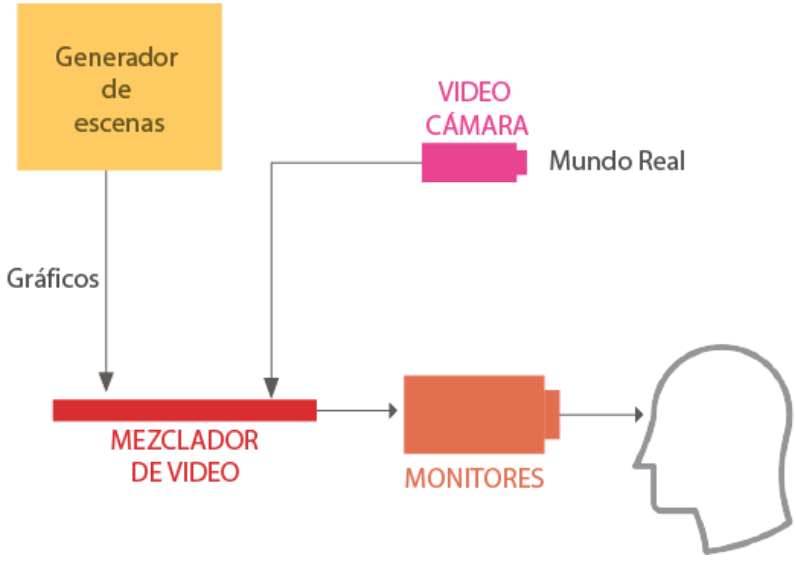


Figura 6. Esquema de la tecnología de video en AR

Hoy en día, prácticamente todo el mundo dispone de un smartphone con cámara incorporada, de un ordenador que tenga algún tipo de webcam o de tablets que también incorporen algún elemento de captación del entorno. Dado el uso extendido de estos dispositivos en la sociedad, actualmente las aplicaciones más populares de la Realidad

Aumentada se están implementando en dispositivos caracterizados por la tecnología de video.

Una de las ventajas que proporciona los sistemas basados en vídeo frente a la tecnología óptica, es que pueden utilizar métodos de registro 3D que consisten en asegurar que el mundo virtual se alinee de la mejor forma posible al mundo real. Para poder realizar un registro preciso, es necesario que la cámara virtual que se usa para generar los gráficos se corresponda lo máximo posible con la cámara real. Para ello, se necesita conocer tanto los parámetros intrínsecos (como la distancia focal o distorsión de la lente) y extrínsecos (como la posición o la orientación) de la cámara. El proceso de calcular el valor de los parámetros intínsecos recibe el nombre de Calibración. Cabe señalar que, para poder conocer la ubicación e inclinación de la cámara (parámetros extrínsecos) en la escena real y así saber desde qué punto de vista se está visualizando el mundo, es necesario un proceso que se denomina Localización o Tracking. Para localizar la cámara, típicamente se han usado marcadores (imágenes con mucho contraste, símbolos…) en la escena, pero hoy en día se están llevando a cabo soluciones que no necesitan marcadores artificiales. Según Vicent Lepetit, profesor de la universidad de Bordeux en visión por computador[8] los métodos de localización se pueden clasificar dependiendo de si se tiene información previa 3D del entorno o no y de si se consideran características naturales o si se usan marcadores visuales. En la Figura 7 se puede ver una representación de dicha clasificación.

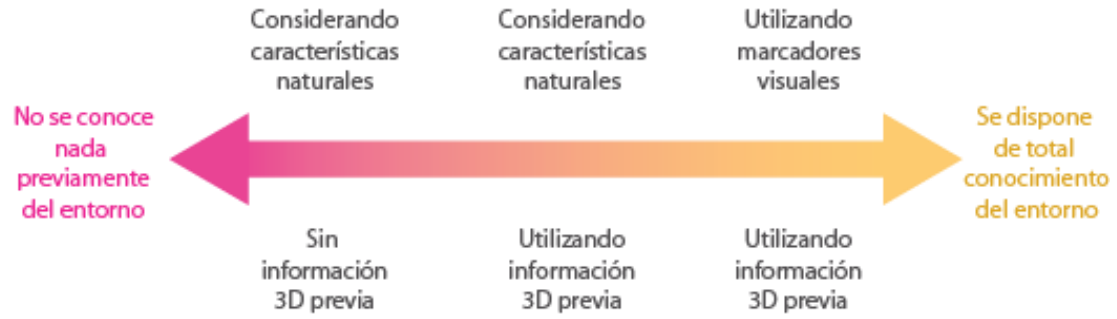


Figura 7. Clasificación de sistemas de localización o tracking

En caso de tener información 3D previa del entorno y de usar características naturales del mismo, se suelen usar técnicas de tracking basadas en características como, por ejemplo, las técnicas de SIFT y SURF [8] y que extraen puntos distintivos de los objetos que son invariantes a cambios de iluminación y puntos de vista. Por otro lado, en esta categoría también encajan los métodos basados en modelos 3D conocidos, donde se realiza una estimación de la posición y orientación.

[8] Información disponible en: *http://icwww.epfl.ch/~lepetit/*

Cuando no se tiene información 3D previa del entorno y se siguen considerando características naturales se puede hablar de técnicas de SLAM o PTAM [9] [10] que permiten crear un mapa 3D del entorno para localizar la posición y orientación de la cámara respecto a dicho mapa.

Por otro lado, la última clasificación que se realiza es la de no considerar características naturales, si no marcadores visuales ya conocidos por el sistema. De esta forma, se pueden realizar métodos de tracking basados en regiones donde se detectan cambios de color o intensidad, o basados en contornos, donde se detectan fronteras.

2.2.3. Software y plataformas de Realidad Aumentada

Actualmente existen múltiples librerías, software y plataformas que permiten a los desarrolladores de aplicaciones de Realidad Aumentada crear contenido con facilidad de forma que solo se tengan que preocupar de este y no de los cálculos que conllevan los procesos de registro, proyección y localización. A continuación, se mencionan algunas herramientas que se han hecho populares los últimos años.

- ARCore y ARKit.

  ArCore es la herramienta de creaciones de aplicaciones de Realidad Aumentada desarrollada por Google y ArKit es la plataforma desarrollada por Apple. Estas plataformas utilizan dos tecnologías claves para integrar los objetos o contenidos virtuales en el mundo real: Motion Tracking, Enviroment Understanding y Surface detection. El Motion Tracking consiste en rastrear el movimiento del usuario en el mundo real. Para ello, utiliza la cámara de los dispositivos con la que detecta los puntos característicos distintivos del mundo real basándose en un método Odométrico. A medida que la cámara comienza a moverse, la herramienta rastrea el cambio de la posición de todos los puntos característicos de la escena de forma que el punto que está cerca de la cámara se moverá menos en la pantalla en comparación con el punto que se encuentra más lejano. De esta forma, basándose en estos datos, la herramienta es capaz de detectar la posición de la cámara en relación con el resto de los objetos de la escena real. Para la detección de la orientación, la herramienta utiliza los sensores inerciales que están incorporados en los smartphones. Por otro lado, la herramienta incorpora la funcionalidad del Surface Detection que busca constantemente los puntos característicos distintivos que parecen estar en superficies horizontales comunes, como pueden ser mesas, suelos o escritorios. También proporciona los límites del plano mediante la detección de los bordes en las imágenes de la cámara.

Cabe destacar que Google presentó, como añadido en ArCore, una función que no tienen otras herramientas. Dicha funcionalidad se denomina Light Estimation. Lo que realiza esta función es detectar condiciones de luz promedio de las escenas y proporcionar a los desarrolladores datos e información de la iluminación de la escena para dotar a los elementos virtuales de sombras y brillos según dicha información. A pesar de que los resultados que presenta esta funcionalidad son bastante buenos, cabe señalar que al realizar un promedio de la intensidad de la luz de la escena y de no proporcionar la intensidad de luz en cada punto del espacio tridimensional, la herramienta es incapaz de generar superficies perfectamente reflectantes del mundo real.[9] Como ejemplo se muestra la Figura 8, donde se observa el personaje de un león generando sombras según la luz del entorno.

Figura 8. Personaje generado con ARCore que se muestra iluminado mediante la estimación de la luz del entorno.

- Vuforia

  Vuforia es un kit de desarrollo de software (SDK) basado en Realidad Aumentada que permite crear aplicaciones para Android, IOS o UWP (Universal Windows Platform). Esta herramienta permite usar fácilmente funciones avanzadas de visión por computador para poder gestionar los procesos de localización y registro sin tener la necesidad de que un desarrollador lo tenga que implementar por su cuenta. Según el esquema presentado en la sección

[9] ARCore, «Fundamental Concepts» [En línea]. Available: *https://developers.google.com/ar/discover/concepts*. [Último acceso: 2018 Septiembre 06].

anterior, Vuforia utiliza para la localización métodos basados en información 3D del entorno que es conocida previamente. Por un lado, provee la funcionalidad de generar marcadores basados en contornos (denominados Targets y que pueden ser imágenes creadas por los usuarios) y por otro lado ofrece la posibilidad de reconocer objetos 3D en el entono basándose en las características naturales de estos.

Al contrario que ARCore, actualmente Vuforia no ofrece ninguna función para iluminar los objetos virtuales según la información lumínica del entorno real.

2.3. Iluminación en gráficos por computador

Una de las vertientes que ha seguido la computación gráfica, es la de generar imágenes fotorrealistas que simulen la interacción física de la luz en el entorno. La simulación de la luz tiene en cuenta cómo la luz se refleja en los objetos y se dispersa en el entorno así que conocer las propiedades de las superficies de los objetos geométricos (como el color o las características de reflexión) y los elementos volumétricos como la niebla o el humo, son necesarios.

Hay diferentes métodos que permiten modelar la interacción de la luz que se pueden dividir en dos grupos: métodos que se basan en la iluminación local y métodos que se basan en la iluminación global. Los métodos de iluminación local solo tienen en cuenta la forma en la que los objetos responden a la luz directa, como la reflejan, la dispersan…. En contraste, los métodos de iluminación global se enfocan en el problema de cómo el entorno es iluminado como un todo, es decir, incluyen la iluminación directa de las fuentes de luz y la indirecta que alcanzan las superficies mediante las reflexiones y las dispersiones de otros objetos del entorno.

Para modelar la interacción directa de los objetos con la luz, en los motores gráficos se utilizan fuentes de luz. Las fuentes pueden ser o bien puntuales, direccionales o de área. Las fuentes puntuales se caracterizan por poseer una emisión por igual en todas las direcciones y tienen una localización determinada en el espacio. Sirven para modelar lámparas, chispas o explosiones. Las fuentes de luces direccionales se encuentran a una distancia infinita por tanto sus rayos se consideran paralelos, de forma que no se necesita indicar su posición, pero si la dirección de los rayos. Sirven para simular luces distantes, como la luz del sol.

Tanto las fuentes de luz puntuales como las direccionales, son muy utilizadas en aplicaciones en tiempo real, sin embargo, tienen una limitada aplicación en los gráficos

realistas ya que las fuentes de luz reales suelen incidir en la superficie con diferentes ángulos. De aquí surgen las fuentes de luz de área. Al iluminarse un objeto desde diferentes direcciones a la vez, el sombreado tiende a ser más suave y realista por lo que se puede utilizar para crear una farola realista, un banco de luces… El inconveniente que presenta este tipo de luces son los elevados tiempos de ejecución, por lo que aún hoy en día son difíciles de usar en tiempo real. [11]

La luz incidente en una superficie no solo hace que un objeto se ilumine y genere sombras si no que, dependiendo de las características del material, la luz se puede reflejar o dispersar en mayor o menor medida. Las reflexiones se pueden clasificar en dos tipos básicos: las reflexiones difusas o las especulares. Las reflexiones difusas ocurren cuando una luz incidente es dispersada uniformemente en todas las direcciones. Por otro lado, las superficies especulares son aquellas en las que la luz reflejada lo hace cerca de la dirección del reflejo del espejo y provoca que el objeto se vea con brillo.

Por otro lado, las técnicas de iluminación global tienen en cuenta también las reflexiones o dispersiones de la luz que llegan de otros objetos. Para el cálculo de la iluminación global, existen diferentes métodos o algoritmos como la radiosidad [12], el raytracing [13] o el photon mapping [14] entre otros. Sin embargo, uno de los problemas que presentan algunas de estas técnicas, son los grandes tiempos de cómputo que precisan al renderizar las escenas, lo que hace que su uso en tiempo real sea complicado.

2.3.1. Iluminación Basada en Imagen

Una técnica que está teniendo gran aceptación entre la comunidad científica y que además permite iluminar objetos virtuales con imágenes captadas de escenas reales es la de iluminación basada en imágenes [15]. Este modelo consiste en ofrecer la posibilidad de iluminar una escena virtual teniendo en cuenta un entorno real y sin tener un gran coste computacional. Dicha técnica usa mapas de entorno, que se pueden considerar como una envoltura alrededor de una escena que simula el cielo. Dichos mapas se suelen capturar con fotografías de la escena real. Las fotografías se suelen convertir en mapas cilíndricos o Cubemap (imágenes del entorno que se mapean en seis caras de un cubo). Para obtener los mapas se han propuesto diferentes técnicas como la de usar una bola reflectante para capturar fácilmente la luz de una escena o las Imágenes de Alto Rango Dinámico (HDRI) [16]

HDRI es un formato de imagen que permite un mayor rango dinámico de luminancias que las imágenes normales (Low Dinamic Range Image, LDRI). Uno de los

problemas que caracteriza a las LDRI es que dependiendo de las condiciones lumínicas de la escena y de algunos parámetros de la cámara, las imágenes pueden aparecer con zonas sobrexpuestas (al sensor le llega una gran cantidad de luz) o subexpuestas (al sensor de la cámara le llega poca luz y se ven zonas oscuras). Al contrario, las HDRI representan con más exactitud el extenso rango de niveles de intensidad de las escenas reales.

Actualmente se han propuesto diferentes métodos para generar las HDRI. Uno de los más difundidos es el de fotografiar varias veces la misma escena con diferentes tiempos de exposición y luego combinar las fotografías de forma que se obtenga un mayor rango de radiancias. Al realizar una fotografía, el CCD (*charge-coupled device)* de la cámara, responde a variaciones en la exposición. La respuesta es una función no lineal descrita por la curva de características de cada CCD que tiene en cuenta la respuesta que se produce cuando no hay exposición y la saturación producida a altas exposiciones. De esta forma, dependiendo del tiempo de exposición que se establezca en la cámara, las respuestas variarán ya que a mayor tiempo recogiendo luz más saturada estará la imagen y a menor tiempo, más subexpuesta. Dado que la curva de respuesta de la cámara nos ofrece información sobre cómo la exposición se proyecta a la luminosidad del píxel, la inversa de la función nos permitirá reproducir la radiancia real de la escena. Actualmente existen diferentes herramientas que permiten hacer esta conversión y obtener una HDRI dadas varias LDR. En la imagen de la Figura 9 se muestra un ejemplo de esta técnica. Se puede ver 16 fotografías tomadas con una apertura de diafragma de f/8 con tiempos de exposición que van desde 30 a 1/1000 segundos.

Figura 9. Resultado de tomar 16 fotografías con apertura de diafragma de f/8 y exposiciones de 30 a 1/1000 segundos[10]

[10] Imágenes obtenidas de [18]

El problema de este tipo de imágenes es que los monitores o los proyectores actuales tienen un rango dinámico limitado que no es capaz de reproducir el rango dinámico completo de las HDRI. El Tone Mapping es una técnica que resuelve el problema permitiendo aproximar la apariencia de las imágenes de alto rango dinámico en un medio que tiene un rango dinámico limitado. Esta conversión la realiza conservando los detalles de la imagen y el aspecto del color ya que son importantes para apreciar el contenido original de la escena. En la actualidad existen numerosos algoritmos de Tone Mapping como el que presentan en [17] donde realizan un ajuste del histograma de luminancias mapeándolo de forma eficiente para mostrar valores que preserven el contraste de la imagen. También proponen una segunda fase de Tone Mapping donde utilizan modelos de brillo y sensibilidad del color, para reproducir las imperfecciones de la visión humana. En la Figura 10 se puede ver a la izquierda, la imagen LDR tomada por una cámara convencional, en el centro la imagen HDRI con un Tone Mapping basado en histograma y en la imagen de la derecha el Tone Mapping añadiendo efectos en la imagen para simular la visión del ojo. Dichas imágenes fueron obtenidas de [18].

Figura 10. Imagen LDR a la derecha, imagen HDR con tone mapping basado en el histograma y imagen HDR con ToneMapping basado en histograma y caracterísitcas de la visión humana[9]

Algunos trabajos como el que se presenta en [19], se han centrado en demostrar que, realizando un mapeo de tonos inversos, se puede reconstruir un HDRI a través de un único LDRI. Esta técnica recibe el nombre de Inverse Tone Mapping. En [20] usan un operador de Inverse Tone Mapping que aplican de forma independiente a cada píxel. Dado un valor de entrada RGB, primero determinan la luminosidad como:

$$L_i = 0.3 \cdot R_i + 0.59 \cdot G_i + 0.11 \cdot B_i$$

Ecuación 1. Luminosidad de un píxel

Donde R_i, G_i y B_i son las componentes rojas, verdes y azul de la imagen de entrada. A partir de aquí se determina un factor de escala k basado en la luminosidad de entrada como:

$$k = 10 \cdot L_i^{10} + 1.8$$

Ecuación 2. Factor de escala

De esta forma la salida de la imagen se determina mediante la siguiente fórmula:

$$[R_o, G_o, B_o] = k \cdot [R_i, G_i, B_i]$$

Ecuación 3. Valor en HDR de la imagen

De esta forma, el Inverse Tone Mapping nos permite reconstruir una HDRI partiendo de una única imagen LDR, al contrario que la técnica mencionada anteriormente donde era preciso tener varias LDRI. De esta forma este proceso se podría utilizar para generar HDRI en tiempo real dada su facilidad de cálculo. Cabe destacar que la HDRI obtenida es una buena aproximación, pero no tiene la misma calidad ni conserva toda la información que tendría una HDRI real. Sin embargo, se ha demostrado en [20] que la imagen obtenida ofrece buenos resultados a la hora de usarse como mapa de entorno.

Tras establecer una HDRI como un mapa de radiancia, se procede a realizar el cálculo de la iluminación. Existen numerosos métodos que proponen iluminar los objetos sintéticos mediante los mapas de entorno como el propuesto por Ramamoorthi [21] donde utiliza los armónicos esféricos[11] para aproximar las propiedades difusas del mapa de radiancia.

Sin embargo, todos estos métodos solo iluminan los objetos sintéticos en tiempo real, pero no producen sombras, para ello se precisan de fuentes de luz. Así que para poder tener los objetos sintéticos completamente iluminados y que generen sombras hay que hacer una mezcla entre algún método de iluminación y alguno que calcule dónde se encuentran las fuentes reales. De esta forma, para poder generar sombras acordes con la realidad, se añaden fuentes de luces posicionadas donde se encuentra la luz en los mapas de entorno. Hay imágenes de entorno en las cuales localizar los focos de luz es un proceso fácil que se podría realizar a simple vista, pero hay otras en las que localizar el punto de donde proviene la luz, es más complicado. Además, este “truco” solo se

[11] Funciones armónicas que representan la variación espacial de un conjunto ortogonal de soluciones de la ecuación de Laplace. Son útiles para aproximar la iluminación difusa de una fuente de luz en un punto de la superficie.

podría hacer con mapas de entornos estáticos donde la iluminación de la escena no cambiase o lo hiciese de forma predecible. Se han desarrollado algunas técnicas para como la propuesta en [22] donde utilizan un algoritmo basado en la cuantización del color, el Median Cut, para localizar los puntos de luz en la imagen. Cabe señalar que, en este caso las luces calculadas también se utilizan para iluminar y no proporcionar únicamente sombras. Dicho algoritmo se basa en ir dividiendo el mapa de entorno en secciones que contengan una energía lumínica similar, de forma recursiva. De esta forma, la imagen se irá dividiendo en 2^n regiones de la siguiente forma:

1. Se agrega toda la imagen como una única región
2. Para cada región, se subdivide a lo largo de la dimensión más larga (ancho o alto) de tal forma que la energía lumínica de ambas secciones se divida de la forma más uniforme posible.
3. Si el número de iteraciones es menor que n, se vuelve al paso 2.

Cuando la imagen es dividida, se calcula el centro de cada región que determinará la posición de la fuente de luz. De esta forma, el número de luces que se acaba añadiendo a la escena se corresponde con el número de regiones creadas. En la siguiente imagen (Figura 11) se pueden ver los resultados que se obtienen cuando se calculan 16, 64, 256 y 4096 luces.

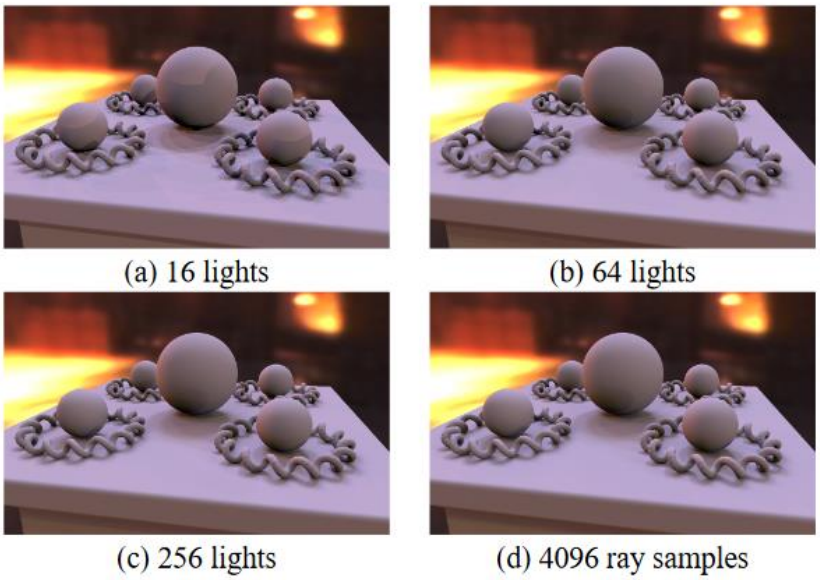


Figura 11. Median Cut con 16, 64, 256 y 4096 luces calculadas[9]

Esta técnica como se puede ver presenta unos resultados bastante realistas, sin embargo el número de luces que se precisa para que funcione correctamente es muy elevado y el tiempo de cálculo de sus posiciones, también lo es. Esto supone que este

método sea complicado implementarlo con mapas de entorno que varíen, como podría ser un video 360 HDR grabado o una emisión en streaming en 360 del mapa de entorno.

De aquí surgen otros métodos como el presentado en [20]. El objetivo de este algoritmo consiste en detectar zonas de pixels con un gran valor de luminosidad y colocar en esas zonas, luces direccionales. Para detectar estas regiones, el primer proceso que se realiza es el de binarizar la imagen de forma que ponga a cero todos aquellos pixels cuya luminancia se encuentre por debajo de un umbral y poniendo a uno todos aquellos que se encuentren por encima de este. Así se obtiene una máscara para detectar todos aquellos pixels pertenecientes a una fuente de luz. La luminancia umbral para la binarización se calcula en función de la luminancia promedio que haya en la imagen HDR. Es decir, que un mapa de entorno con una luz dominantemente fuerte requerirá de un valor de umbral alto y uno con un promedio de luces más difusas, requerirá de un valor de umbral mucho más bajo. De esta forma el cálculo del umbral se hará mediante la media y la desviación típica de la siguiente forma:

$$Y_t = \mu + 2\sigma$$

Ecuación 4. Cálculo del umbral de binarización

Este umbral se ha propuesto a través de pruebas que se describen en [22]. Cabe señalar, que al binarizar la imagen se producen dos efectos que no son deseados. El primero de ellos es que se crean pequeñas zonas de píxels solitarios que superan el umbral pero que no se podrían clasificar como fuentes de luz (por ejemplo, las generadas por brillos en una superficie). El segundo efecto es que se detectan muchas regiones en la zona inferior de la imagen que no se corresponden a una fuente de luz en el entorno real, si no a reflexiones en alguna superficie. Para solucionar el primer inconveniente proponen aplicar un filtro gaussiano (blur) a la imagen HDRI con el fin de que los pixels aislados se detecten como regiones. Por otro lado, para reducir el número de reflexiones detectadas como fuentes de luces, se propone trabajar con la mitad superior de la imagen sacrificando el detectar posibles fuentes de luz provenientes de zonas bajas del mapa de entorno.

Cuando se detectan las regiones, se procede a realizar el cálculo de las fuentes de luz. Cabe señalar que cada zona detectada se corresponderá con una fuente de luz. Para hallar las propiedades de la fuente, el primer paso consiste en calcular la contribución a la irradiancia de cada píxel de la región mediante la Ecuación 5, donde Y_t es el umbral de la luminancia de los píxeles de toda la imagen, Y_p es el valor de luminancia del pixel, L_p su radiancia y Ω_p el ángulo sólido subtendido por dicho píxel.

$$E_p = \Omega_p \cdot (L_p - L_p \cdot min(1, \frac{Y_t}{Y_p}))$$

Ecuación 5. Contribución a la irradiancia de un pixel

El ángulo sólido[12] viene determinado por la Ecuación 6 donde ϕ_1 y ϕ_2 son las latitudes que limitan un píxel y θ_1 y θ_2 son las longitudes que lo limitan.

$$\Omega_p = |\phi_1 - \phi_2| \cdot (\cos\theta_2 - \cos\theta_1)$$

Ecuación 6. Cálculo del ángulo sólido

A partir de aquí, se calcula la irradiancia total de la fuente de luz como el sumatorio de todos los píxeles pertenecientes a esa región (Ecuación 7)

$$E_l = \sum E_p$$

Ecuación 7. Irradiancia de la fuente de luz

A partir de aquí, se puede determinar la posición de las fuentes en coordenadas esféricas con la Ecuación 8, donde $Y(E_p)$ es la luminancia de un píxel, $Y(E_l)$ la luminancia de todos los pixeles y $< \theta_p, \phi_p >$ las coordenadas esféricas del píxel.

$$< \theta_l, \phi_l > = \frac{1}{Y(E_l)} \sum_{pixels} Y(E_p) \cdot < \theta_p, \phi_p >$$

Ecuación 8. Posición de las fuentes de luz en coordenadas esféricas

Como sabemos, el mapa de entorno es una imagen que se mapea en una esfera que rodea a la escena. Al final los píxeles de la imagen ocupan una posición en el espacio, por eso se puede hallar su ubicación en coordenadas esféricas. Para calcular las coordenadas esféricas de un pixel, es necesario conocer su fila y columna en la imagen según la Ecuación 10. El mapeo se realiza teniendo en cuenta que los pixeles horizontales van desde 0 hasta 2π y las coordenadas de los píxeles verticales van desde -π hasta π.

$$latitud = (columna \,/\, alto\ de\ imagen) \cdot \pi$$
$$longitud = (fila \,/\, ancho\ de\ imagen) \cdot 2\pi$$

Ecuación 9. Coordenadas esféricas de un píxel

[12] Ángulo espacial medido en estereorradianes, que abarca un objeto visto desde un punto dado y que se corresponde con la zona del espacio limitada por las rectas proyectantes desde el objeto hacia el observador.

La ventaja del algoritmo propuesto es la rapidez de cálculo de las posiciones de las fuentes de luz en los mapas de radiancia, lo que supone que se puede usar en tiempo real. En este caso, los mapas de radiancia podrían consistir en videos o emisiones en directo de una escena, por lo que se podrían incluir objetos sintéticos en tiempo real iluminados en función de las características lumínicas del entorno.

2.4. Crítica al Estado del Arte

Como se mencionó anteriormente, la inclusión de objetos sintéticos fotorrealistas en escenas reales dinámicas es de interés para diferentes industrias como el cine, la televisión o para algunas aplicaciones de Realidad Aumentada. A la hora de incluir los elementos virtuales, un aspecto importante es la iluminación de estos que debe de ser acorde con la luz que hay en el entorno para ayudar a crear la sensación de que están en el mundo real.

En el estudio que presenta Paul Devebec en [23], incluye objetos sintéticos en escenas reales. Para ello se basa en la técnica del Image Based Lighting y usa el software Radiance para calcular las luces y sombras de los objetos en función de un mapa de entorno de la escena real. El mapa de entorno en este caso es una fotografia que no varía y por lo tanto incluye los objetos en una escena estática. Por el contrario, se ha visto que el estudio que se presenta en [22], sí que incluye la integración de objetos sintéticos en entornos que varían, pero estos son videos grabados y no escenas en tiempo real.

Por otro lado, también podemos encontrar trabajos que permiten iluminar objetos virtuales en escenas dinámicas en tiempo real. Un ejemplo es el que se presenta en [24], donde se emplean redes neuronales convolucionales para estimar la luz y las sombras de los objetos sintéticos ubicados en la realidad. También se comentó que ya existen herramientas como ArCore que permiten a los desarrolladores crear aplicaciones de Realidad Aumentada donde las sombras y las luces de los objetos virtuales se calculan a partir de las imágenes que capta la cámara del dispositivo móvil. De esta forma, ambas técnicas permiten iluminar los objetos sintéticos mediante una estimación basada en una única imagen que no captura toda la información del entorno y a pesar de que no obtienen malos resultados, no proveen la iluminación de los objetos basándose en la iluminación del entorno completo.

Consecuentemente, en el presente trabajo se pretende implementar un sistema que sea capaz de iluminar los objetos sintéticos ubicados en escenas dinámica y en tiempo real que además no se base en estimar la luz mediante una imagen de una parte del entorno, si no en la información que provee una captura de 360 grados de este. Dados los buenos resultados que se han presentado con la técnica del Image Based Lighting, se presenta un sistema que es capaz de iluminar los elementos virtuales en tiempo real.

3. Tecnologías empleadas

Para el desarrollo de este TFM, se han usado las herramientas que se exponen a continuación.

3.1. Vuforia

Como se comentó anteriormente, Vuforia es un kit de desarrollo de software de Realidad Aumentada. En este caso, se ha utilizado para no tener que implementar por nuestra cuenta un sistema de tracking y de calibración [25].

3.1.1. Arquitectura de Vuforia

Una aplicación desarrollada con Vuforia está compuesta principalmente por los siguientes elementos:

- Una cámara -> Es la que captura la imagen del mundo real para que el Tracker procese cada fotograma que esta devuelva.
- Una base de datos -> La base de datos donde se almacenan una serie de Targets para ser reconocidos por el Tracker. La base de datos puede ser creada utilizando el Target Manage que provee Vuforia y puede ser una base de datos local o una almacenada en la nube.
- Target (marcadores) -> Son los que utiliza el Tracker para reconocer objetos reales y así poder ubicar y orientar los objetos virtuales en el mundo real. Existen diferentes tipos los cuales se describen posteriormente.
- Tracker -> Se encarga de analizar la imagen de la cámara y detectar objetos del mundo real con el fin de encontrar coincidencias en la base de datos.

De esta forma, la arquitectura de Vuforia se puede ver esquematizada en la Figura 12.

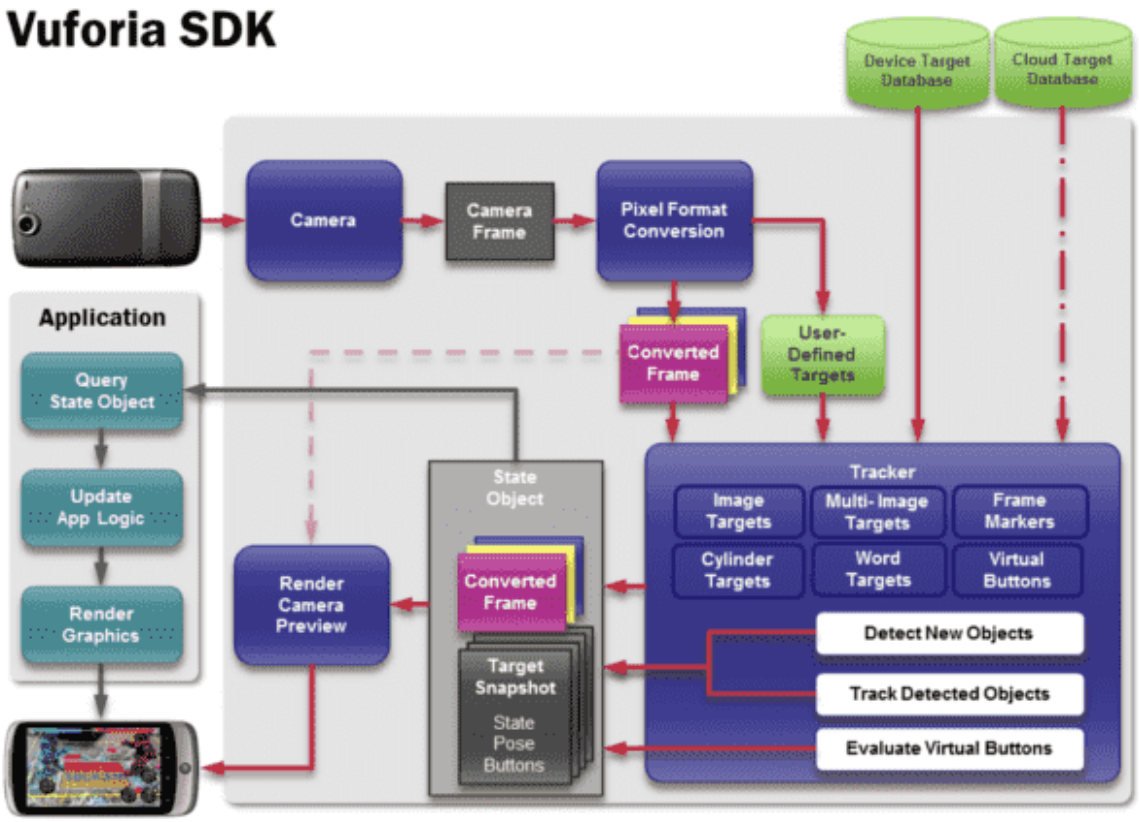


Figura 12. Esquema de la arquitectura del SDK de Vuforia[13]

Por lo tanto, el funcionamiento de Vuforia es el siguiente.

1. La cámara capta una escena (video en vivo).
2. El SDK de Vuforia crea un frame (una imagen en particular dentro de una sucesión de imágenes) de la escena capturada y convierte dicho frame a una resolución determinada para que el Tracker pueda tratar la imagen correctamente
3. A continuación, se analiza la imagen a través del Tracker y se busca coincidencias en la base de datos, la cual está compuesta por los Targets
4. Tras detectar el Target, la aplicación se encarga de renderizar contenido virtual (imágenes, videos, modelos, etc) en la pantalla del dispositivo, y así crear una escena con elementos virtuales combinados con los reales.

[13] Imagen obtenida de: [25]

3.1.2. Targets de Vuforia

Para el proceso de calibración en Realidad Aumentada, Vuforia permite desarrollar diferentes tipos de Targets (Image Targets, Multi-targets, VuMarks, Model Target…). En el presente trabajo se usó el Model Target, que consiste en un marcador que admite la calibración mediante el reconocimiento de un objeto 3D. Vuforia reconoce el objeto haciéndolo corresponder con los elementos que tiene en su base de datos. Por lo tanto, hay que subir a dicha base, el modelo 3D del objeto para que al escanearlo en la escena real sea reconocido. Para ello Vuforia provee la aplicación Model Target Generator que permite cargar una representación 3D digital del objeto y generar una base de datos para poder reconocerlo.

3.2. Unity

A pesar de ser una herramienta para el desarrollo de videojuegos multiplataforma (Windows, Linux, IOS, Android...), Unity ya dispone de funcionalidades que facilitan el desarrollo de todo tipo de aplicaciones gráficas. Dicha plataforma contiene un editor visual muy útil y completo donde mediante unos pocos cliks se puede importar modelos 3D, texturas, sonidos… al igual que permite implementar funcionalidades mediante C#.

Además, Unity incluye un motor gráfico propio que soporta una gran cantidad de plataformas como Direct3D, OpenGL e interfaces propietarias en función de cada plataforma (Wii, Playstation, Xbox…).

Por otro lado, Unity también permite la creación de shaders. Para ello hace uso de ShaderLab Language. Este lenguaje soporta diferentes métodos de programación de Shadders, además de ofrecer soporte a otros lenguajes, como GLSL o Cg.

Cabe desatacar que, en las últimas versiones de la plataforma, ya se incluye el kit de Vuforia para poder implementar de forma fácil aplicaciones de realidad aumentada. Consecuentemente se ha decidido usar esta herramienta por la facilidad que proporciona crear aplicaciones de realidad aumentada. [26]

3.2.1. Iluminación en Unity

El motor de gráficos de Unity soporta tanto iluminación local como global. Para la iluminación local, ofrece diferentes tipos de fuentes de luz como son las direccionales, puntuales, focos y luces de área. Para calcular las sombras proyectadas por un objeto 3D, Unity necesita conocer la intensidad, dirección y color de las fuentes de luz que inciden sobre el objeto. Cabe señalar que el algoritmo de generación de sombras que

implementa Unity es el shadow mapping. Este algoritmo se basa en dos pasos: en el primero se produce un mapa de sombras y el segundo lo aplica a la escena. Para generar el mapa de sombras, la escena se representa desde el punto de vista de la luz y se ve que partes son visibles y cuáles no, guardando la información en el buffer de profundidad. En el segundo pase, se renderiza la escena desde el punto de vista de la cámara teniendo en cuenta el mapa de sombras. Primero se calcula la distancia del punto a renderizar hasta la cámara y se compara con la distancia almacenada en el mapa de sombras, si la distancia es mayor, el punto entonces se encuentra en sombra. En la Figura 13 se muestra en la imagen de la izquierda como se obtiene la profundidad desde el punto de vista de la fuente de luz. En la imagen de la derecha se observa que el punto A tiene la misma profundidad y por lo tanto no se encuentra en sombra, en cambio el punto B es visible desde la cámara pero no desde la luz, por lo tanto se encuentra en sombra.

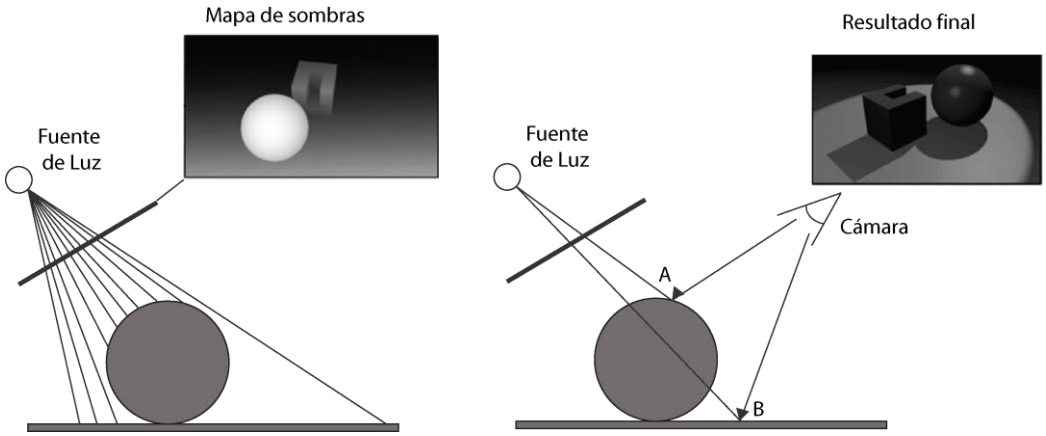


Figura 13. Esquema representativo del shadow mapping

Por otro lado, las sombras de los objetos tienden a hacerse menos visibles cuanto más alejados están estos de la cámara. De esta forma, Unity permite establecer una distancia de sombras, es decir, que los objetos que se encuentren más allá de esta distancia (desde la cámara) no generen sombras ya que estas no suelen ser importantes y supondría una buena optimización del tiempo de cómputo.

Con respecto a la iluminación global, Unity dispone de dos técnicas que la calculan, la “Baked GI” y la “Precomputed Realtime GI”. La técnica del “Baked GI” consiste en generar mapas de luz tradicionales antes de la ejecución del proyecto y que luego no cambien en tiempo de ejecución. Por otro lado, la “Precomputed Realtime GI” realiza un cálculo previo de iluminación donde se almacena el cálculo del rebote de luz alrededor de una geometría estática dentro de una escena. De esta forma no se crean mapas de luces que permanecen estáticos durante la ejecución, si no datos que contienen la

información necesaria para generar y actualizar mapas de luz de baja resolución en tiempo real.

Una forma de recoger los datos para generar los mapas, es mediante la técnica del Probe Lighting que muestrea la iluminación entrante en un punto específico del espacio 3D y se codifica en una esfera usando los armónicos esféricos. Estos coeficientes tienen un bajo costo de almacenamiento y se pueden "desempaquetar" rápidamente y usarlos dentro de la escena para aproximar la iluminación de la superficie. En Unity la herramienta que proporciona esta funcionalidad es el Light Probe. Una de las limitaciones es que es difícil representar la iluminación de alta frecuencia en todo el rango esférico sin aumentar el orden de los armónicos. Dado que el coste en tiempo de cómputo aumenta considerablemente, los Light Probe se limitan a armónicos esféricos de orden inferior, por lo que únicamente almacenan iluminación difusa.

Cabe señalar que Unity también ofrece la posibilidad de crear mapas de entorno con dos intenciones: crear escenarios lejanos que simulen un cielo o paisajes distantes y calcular la luz que proviene de la escena. Dicha luz es ambiental y mediante los Light Probe se puede recoger información sobre ella, pero como es de suponer no genera sombras. Para ello, habría que seguir alguna de las técnicas nombradas en el Estado del Arte o colocar fuentes de luz a mano en la ubicación de la procedencia de la luz en el mapa de entorno. En Unity, los mapas de entorno reciben el nombre de Skybox y soportan texturas panorámicas o cubemaps. De esta forma, se precisa crear un material que tenga asociado un shader de tipo skybox.

Unity también ofrece la posibilidad de recoger reflejos de las escenas. Para ello usa los Reflection Probe que en este caso actúa como una cámara que captura una vista esférica de su entorno en todas las direcciones. La imagen se almacena como un mapa de entorno que se puede aplicar a objetos con materiales reflectantes. Una de las características de los Reflection Probe es que las imágenes esféricas del entorno se pueden actualizar en tiempo de ejecución por lo que pueden devolver el reflejo de los objetos que se mueven de forma dinámica en la escena. A parte de reflejar los objetos que se encuentran en la escena, también se puede reflejar el Skybox.

3.3. Ricoh Tetha S

Para capturar el mapa de entorno en tiempo real, se usa la cámara 360 Ricoh Tetha S, cuyas especificaciones técnicas se incorporan en el Anexo I de este documento. Cabe destacar que esta cámara también está pensada para desarrolladores, de manera

que tienen una amplio foro[14] donde se han compartido multitud de desarrollos. La cámara Ricoh Tetha S dispone de una API[15] que está basada en el protocolo OSC (Open Spherical Camera protocol) que proporciona un conjunto de comandos que permite grabar video, captar imágenes, video en tiempo real, modificar parámetros de la cámara como el ISO, la velocidad de obturación… Este protocolo está diseñado para cámaras 360 que tienen WiFi integrado de forma que la comunicación se realiza a través de la red local de la cámara que debe implementar un servidor HTTP 1.1 y así poder hacer solicitudes GET o POST.

3.4. Blender

Para la construcción del modelo 3D para el ModelTarget, se usó Blender, software libre multiplataforma de modelado y animación 3D que principalmente incluye en sus funciones el modelado, texturizado, animación, edición de video, renderizado…

[14] https://developers.theta360.com/en/forums/viewforum.php?f=4

[15] https://developers.theta360.com/en/docs/

4. Diseño e Implementación

En este capítulo se presenta la solución propuesta para iluminar objetos virtuales en tiempo real mediante la técnica del Image Based Lighting. Se comenzará presentando un esquema general con todas los dispositivos y software implicados y a continuación, se especificarán los detalles de la implementación de cada uno de ellos.

4.1. Esquema General

En el esquema de la Figura 14 se muestran las partes de las que está compuesta la solución propuesta:

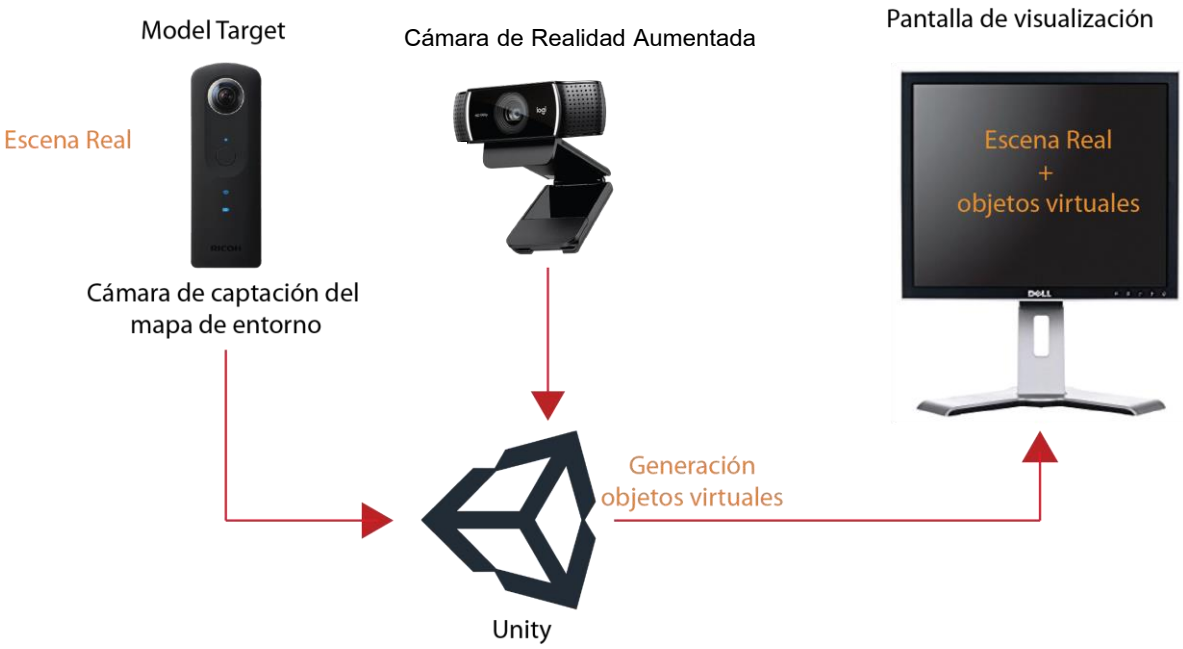


Figura 14. Esquema general de la solución

El sistema está compuesto de:

- Cámara de captación del mapa de entorno → Para poder generar el mapa de entorno, se hace uso de la cámara Ricoh Theta S, que como se ha visto permite recoger imágenes 360º del entorno en tiempo real. La imagen que captura la cámara es LDR por lo que tras hacerle una transformación a HDR, se utiliza como mapa de entorno para iluminar la escena.
- Model Target → Para la localización y ubicación de los objetos virtuales en el entorno real, se ha decidido utilizar la funcionalidad de Vuforia para detectar objetos 3D. En este caso, como el sistema precisa de la cámara 360º para

captar el mapa de entorno, se ha decidido usar la misma como modelo 3D y ahorrarnos el tener que usar otro tipo de marcador.

- Cámara de Captura del entorno real → En este caso se decidió usar una webcam convencional como dispositivo de captura final. A lo largo de la elaboración del proyecto se han utilizado otras cámaras como las de un smartphone, por lo que este componente del sistema es totalmente sustituible y por lo tanto no se especifican sus características.
- Pantalla → En ella se recoge la imagen final, resultante de fusionar los elementos virtuales y la escena real con la iluminación de esta.
- Motor de Unity → Aquí se realizan varios procesos:
    - Se crean los objetos virtuales finales que aparecerán en la escena. Se les dota de diferentes materiales para ver la respuesta que presentan frente a la iluminación de diferentes entornos.
    - Se crea el mapa de entorno con la imagen captada por la cámara 360 en tiempo real y transformada a HDR.
    - Mediante Scripts se crea un algoritmo que identifique la proveniencia de la luz en el entorno real para poder colocar fuentes de luz en Unity que generen sombras.
    - Se crea el dibujo de las sombras sobre una superficie transparente para que estas parezcan que están dibujadas en el entorno real.

  Además, Unity se encarga junto con el motor de render de fusionar la imagen capturada por la webCam y superponer los elementos virtuales en ella.

De esta forma y como cabe esperar, el flujo de trabajo de la solución propuesta es el siguiente:

1. La webcam capta la escena real, al mismo tiempo que la cámara 360 capta las imágenes del mapa de entorno.
2. La imagen captada por la webcam se encarga de localizar el modelTarget para que Vuforia realice los procesos de calibración y tracking.
3. Con la imagen 360 captada por la cámara Ricoh Tetha, se crea el mapa de entorno que servirá para que los objetos reflejen el entorno real.
4. Con la imagen 360 también se calcula la proveniencia de las fuentes de luz en la escena para colocar en su lugar fuentes de luces direccionales que generen las sombras para los objetos sintéticos.
5. Se dibujan los objetos sintéticos sobre la imagen del entorno real capturado con la webcam con las sombras y reflexiones correspondientes.
6. Se repite el primer paso.

4.2. Conexión de la cámara de captación del mapa de entorno

Se decidió conectar por WiFi la cámara Ricoh Tetha S para poder tener flexibilidad a la hora de colocarla en el entorno. Para poder comunicar la cámara 360 con Unity, fue preciso crear un script donde, mediante JSON se intercambiase la información de forma unidireccional (es decir, desde la cámara hacia Unity).

A partir de aquí, mediante la API que proporciona la cámara y usando el comando *camera._getLivePreview* se consiguió obtener imagen de la cámara en tiempo real. La imagen en formato panorámico devuelta por la cámara, se le pasó a una textura en Unity para que se fuese actualizando conforme llegaba información. Este tipo de texturas en Unity se denominan RenderTexture y son generalmente usadas cuando se necesitan texturas que se actualizan en cada frame.

La imagen panorámica devuelta por la cámara 360 es la que se utilizará para texturizar el mapa de entorno. Sin embargo, para obtener buenos resultados de iluminación, se precisa que dicha imagen sea HDR y no LDR que es como nos la proporciona la cámara, así que se realiza una conversión. La transformación se realiza mediante la técnica del Inverse Tone Mapping que se explica en el Estado del Arte de esta memoria (página 29). Como la conversión es un proceso muy costoso ya que se debe de realizar para todos los pixels de la imagen de forma independiente, se decidió crear un shader que tuviese implementadas las ecuaciones Ecuación 1 y Ecuación 2 y que las aplicase a todos los pixels de la imagen. De esta forma, se le pasa al shader la Render Texture que genera como salida la cámara y se obtiene una textura HDRI que se le aplica al Skybox y que se usará para la detección de las fuentes de luz. En la Figura 15 se puede ver de forma esquemática lo comentado.

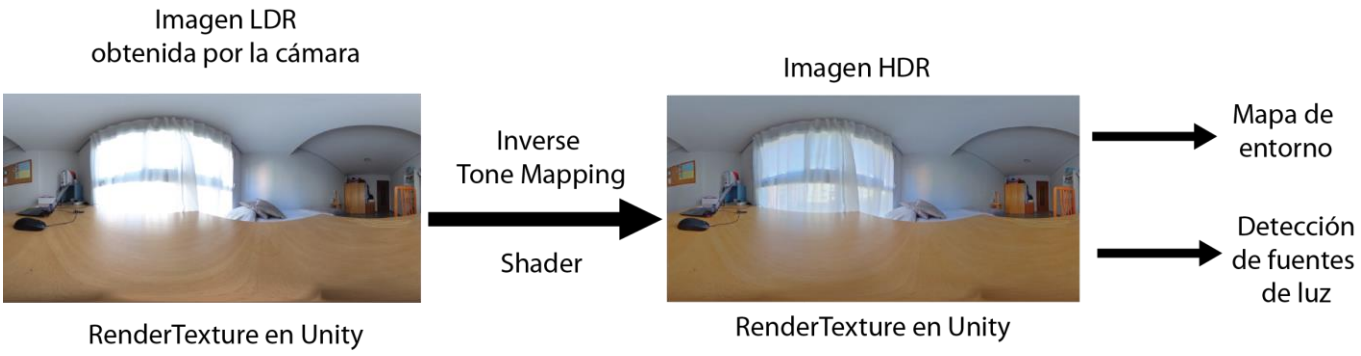


Figura 15. Esquema de transformación de LDR a HDR en Unity

4.3. Model Target

Un dispositivo esencial en la solución propuesta es la cámara Ricoh Tetha S, que además estará incluida en las escenas o se encontrará inmersa en ellas para poder capturar el entorno en 360 grados. Además, como se verá más adelante, es importante que esta se encuentre cerca de las superficies especulares, para generar los reflejos de la forma más realista posible. Sabiendo que es un objeto que siempre se encontrará ubicado en alguna parte de la escena, se ha decidido utilizar como marcador evitando así usar marcadores adicionales.

Para reconocer la cámara se usan los ModelTarget de Vuforia, que permiten usar objetos físicos como marcadores, para la localización de la cámara. Para que Vuforia pueda reconocer el objeto es preciso que este disponga de las siguientes características.

- Que sean objetos estáticos que no se muevan una vez son detectados por primera vez. La cámara se puede mover alrededor de estos, pero estos no podrán moverse.
- Que sean objetos que no presenten un único color o textura si no que dispongan de varios colores.
- Que los objetos tengan suficiente geometría ya que, si no, podrían confundirse con otros objetos de la escena. Por ejemplo, un cubo que es una geometría muy simple podría ser fácilmente confundido con otro objeto de la escena.
- Que los objetos no puedan cambiar su geometría dado que si pudiesen cambiar, estos dejarían de corresponderse con el modelo 3D generado.

Partiendo de las condiciones que propone Vuforia, podríamos afirmar que la cámara cumple con las dos primeras. En primer lugar, la cámara no dispone de ningún componente que pueda cambiar su geometría y, además se puede mantener en un lugar fijo de forma que no se rote ni se desplace durante el transcurso de la sesión. Sin embargo, el modelo dispone de una geometría demasiado simple y de pocos colores que además no proporcionan un gran contraste entre ellos. Como solución, se añadió mediante adhesivos una textura extra a la cámara con un color llamativo para que detectase más puntos característicos. El modelo final de la cámara se puede ver en la siguiente imagen (Figura 16):

Figura 16. Cámara 360º con geometría extra para el ModelTarget

Para que Vuforia sea capaz de detectar el objeto 3D, hay que crear una representación digital de este y añadirlo a una base de datos. Cuando se realice la detección del objeto, Vuforia detectará si es el mismo que se ha añadido a la base de datos. Para realizar una mejor detección hay que intentar modelar el objeto digital de forma que se parezca lo más posible al real. Vuforia también aconseja algunas características que deben tener la representación digital:

- El modelo 3D no deberá de contener grietas en la malla
- El modelo 3D debe de tener todas sus normales en la dirección correcta
- El modelo 3D debe de tener los colores correspondientes asociados a cada parte del objeto. Cualquier etiqueta o patrón de superficie ayuda a Vuforia a mejorar la detección.
- Evitar texturas fotorrealistas, solamente materiales que cambien el color de la superficie.

La representación del objeto digital se realizó con Blender. Primero se modeló el cuerpo de la cámara para a continuación modelar los elementos añadidos para mejorar el tracking. Una vez se realizó el modelado, se procedió a dar color a cada una de las partes de la cámara. Todo este proceso se realizó teniendo en cuenta los puntos detallados anteriormente. El modelo 3D final de la cámara es el que se muestra en la Figura 17.

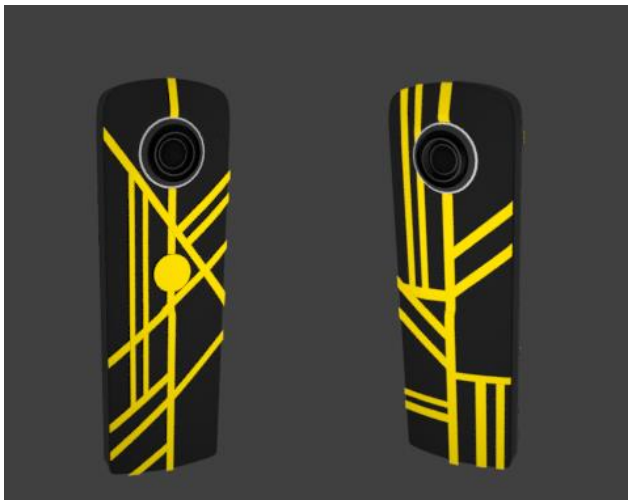


Figura 17. Cámara modelada en Blender. A la izquierda se muestra la parte delantera y a la derecha la trasera.

Tras realizar el modelo hay que crear un dataset para que Vuforia pueda reconocer la cámara. Para ello Vuforia dispone de la herramienta Model Target Generator. El primer paso es dar nombre y ubicar en algún directorio el dataset y subir el modelo 3D. Cabe destacar que la aplicación admite diferentes formatos, como el .dae, .3ds entre otros. En este caso se utilizó el formato .dae para importar el modelo al Model Target Generator desde Blender. Cuando se completa la importación, hay que proceder a cambiar las unidades de las dimensiones del modelo haciéndolas corresponder con las del objeto real (si estas se miden en metros, en decímetros o centímetros). A continuación, hay que decidir si la vista del objeto (cámara, smatphone…) en la que se reconocerá el objeto estará posicionada vertical u horizontalmente. En este caso se decidió usar la vista horizontal. Aquí nos aparecerán unos marcos representando dicha vista y se deberá de ubicar y rotar el objeto en la posición en la que se va a detectar mediante las herramientas que aparecen a la izquierda de la Figura 18 donde se ve la interfaz de la aplicación.

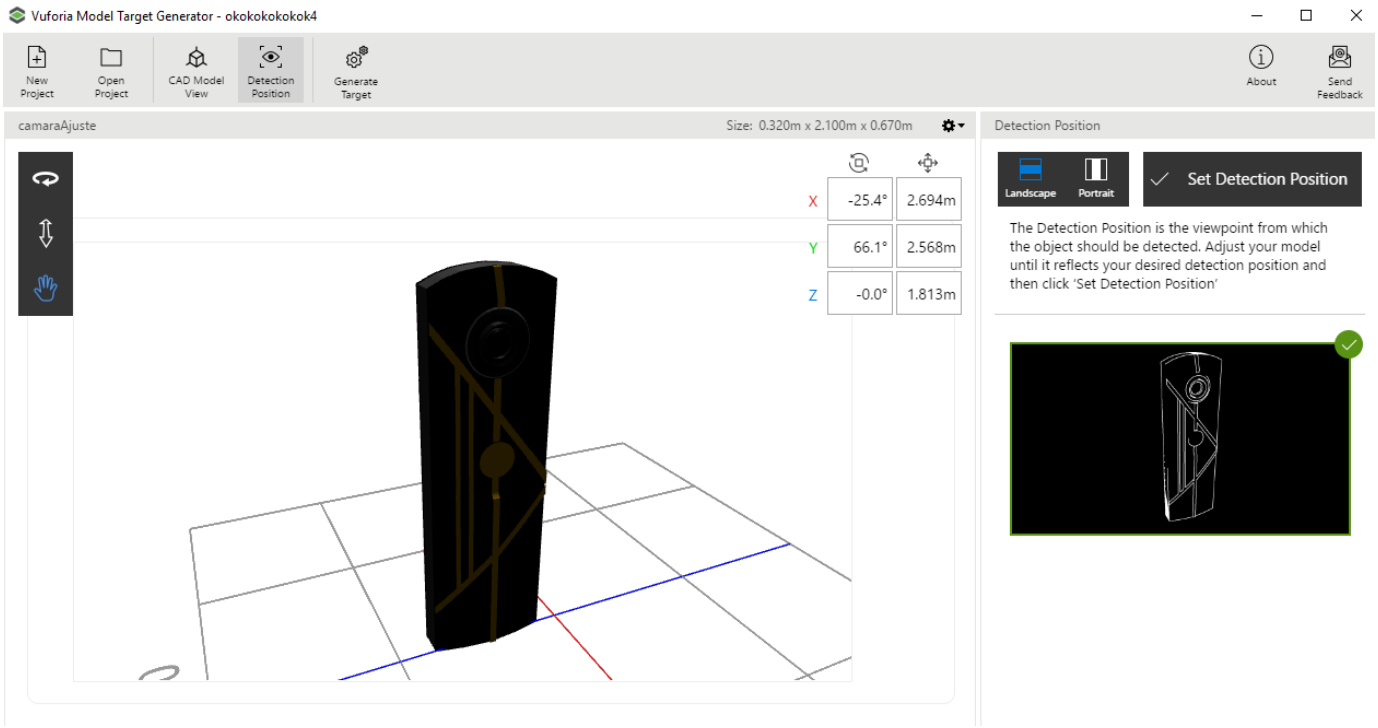


Figura 18. Interfaz de Vuforia Model Target

A la derecha se muestra una imagen binaria donde se ve una representación del modelo desde la perspectiva en la que será detectado. A la hora de ejecutar la aplicación se recomienda utilizar dicha representación para guiar a los usuarios a ver el objeto de la perspectiva correcta. Hay que señalar que esta es una primera detección que una vez hecha, el usuario puede moverse alrededor del objeto siendo este aún detectado. Como se puede ver en la Figura 15 se decidió usar una perspectiva en picado de la cámara ya que la cámara de detección del entorno se supone que estará más elevada que el objeto.

4.4. Cálculo y configuración de las reflexiones

4.4.1. Creación del mapa de entorno

Tal y como se comentó, para poder calcular las componentes difusas y especulares de los objetos situados en la escena dependiendo de lo que ocurre en el entorno real, se hace uso de la técnica del Image Based Lighting. Para ello es preciso establecer un mapa de entorno, que como ya sabemos, Unity puede implementar con facilidad y que denomina Skybox. Al Skybox se le asigna un material con una textura panorámica, que en este caso, es la imagen que nos devuelve la cámara 360 transformada a HDR. Para crear dicha textura se siguen los siguientes pasos:

- Se crea un nuevo material al que se le asigna el shader propio de Unity. En este caso se selecciona que el shader del Skybox sea una imagen panorámica.

- A continuación, se le asigna la RenderTexture que proporciona la salida del shader que hace la conversión de la imagen LDR devuelta por la cámara 360 a HDRI.
- Aquí se puede cambiar la exposición del mapa de entorno que se decide dejar a 1 porque se considera que la imagen tiene una exposición adecuada.
- También se varía la rotación de la textura en el mapa de entorno para que esta coincida con la imagen del entorno real.
- Otra característica a definir es que el tipo de imagen panorámica consistía en una de 360 grados y no en 180.

En la Figura 19 se puede ver la configuración final del material en Unity.

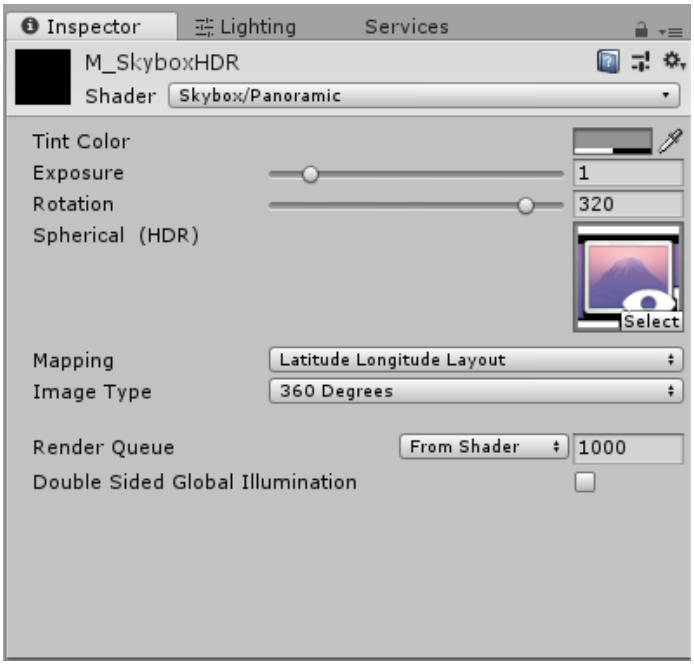


Figura 19. Configuración del material asociado al Skybox en Unity

Para asignar el material configurado al Skybox, hay que dirigirse al panel de iluminación donde además, se permiten hacer algunos ajustes. En este caso, los ajustes y pasos que se siguieron para la creación del Skybox fueron los siguientes:

- Primero se asignó el material previamente creado como Skybox. Se nos permite escoger una fuente de luz que simule al sol en la escena. Como nuestras fuentes de luz se crearán en tiempo real, no seleccionamos ninguna fuente como tal.
- Se nos permite hacer algunos ajustes con respecto a la iluminación ambiental.
    - Hay que especificar que la fuente de luz ambiental será el Skybox no un color o un gradiente.

- La intensidad de la luz ambiente se mantendrá a 1 que es el parámetro por defecto.
- Por último, hay que especificar que la iluminación ambiental se actualiza en tiempo real.

- También se nos permite configurar las reflexiones del entorno.
  - El primer parámetro nos permite especificar si utilizamos el skybox para reflejos o no. En este caso marcamos que sí.
  - A continuación, especificamos el grado de visibilidad de la fuente de reflejos en los objetos.
  - Por último, se nos permite especificar cuantos rebotes de reflejos queremos que se tengan en cuenta, es decir que si lo asignamos a 1 entonces solo tendremos en cuenta el reflejo inicial que sería el de los objetos cercanos y el Skybox, sin embargo, si queremos también el reflejo que ha generado otro objeto cercano, entonces debemos de aumentar el número a 2. En este caso este parámetro se fue modificando dependiendo del tipo de escena que se creó.

En la Figura 20 se puede visualizar el panel de iluminación con las configuraciones realizadas en Unity.

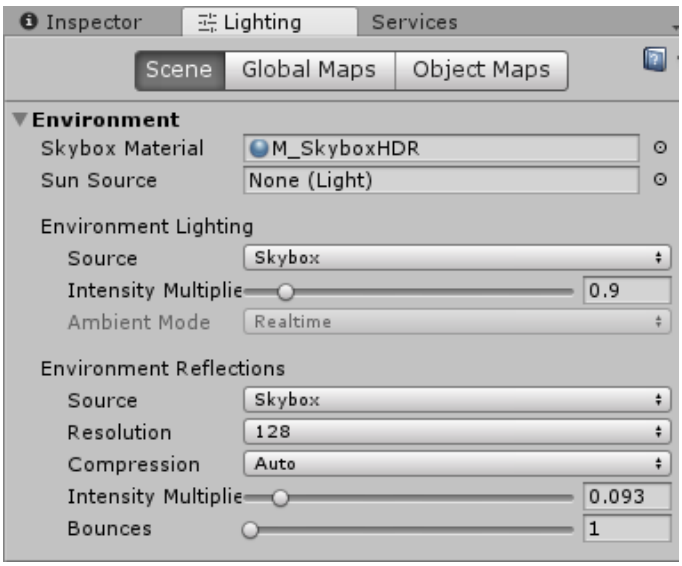


Figura 20. Configuración del panel de Iluminación en Unity

4.4.2. Reflexiones difusas. Configuración de los Light Probe

Como se comentó en el apartado de Tecnologías empleadas, al tener que calcular los mapas de entorno en tiempo real, es preciso disponer de una serie de cálculos

posteriores a la creación de estos. Estos cálculos se obtenían mediante el uso de los Light Probe. En las escenas en Unity, los Light Probe quedan representados como matrices formadas por pequeñas esferas que captarán información sobre la luz en el lugar donde estén ubicadas. En la Figura 21 se puede ver un ejemplo de como se ven los Light Probe en el entorno de Unity y como las esferas han sido colocadas en lugares estratégicos donde se requiere conseguir información de la iluminación difusa.

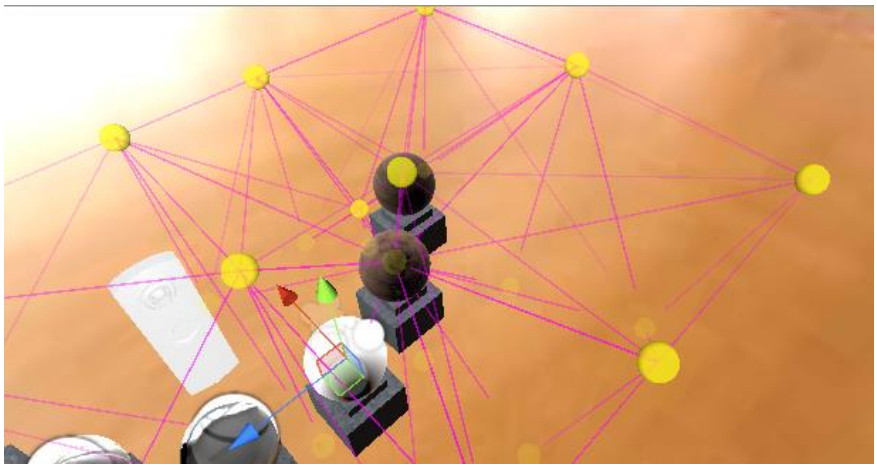


Figura 21. Configuración y visualización de los Light Probe en el entorno de Unity

De esta forma, en la creación y configuración de los Light Probe se siguieron los siguientes pasos:

- Primero se creó y añadió a la escena un Light Probe.
- A continuación, dependiendo del tipo de escena, se colocaron las sondas en diferentes lugares donde se vio que era interesante calcular la luz. Cabe destacar que aquí también se tuvo especial cuidado ya que cada sonda de luz consume grandes tiempos de cómputo.

Por otro lado, para que un objeto reciba iluminación de un sistema de Light Probes, hay que habilitar la opción en el objeto, tal y como se ve en la Figura 22 donde además se observa qué sondas son las que afectan al objeto seleccionado.

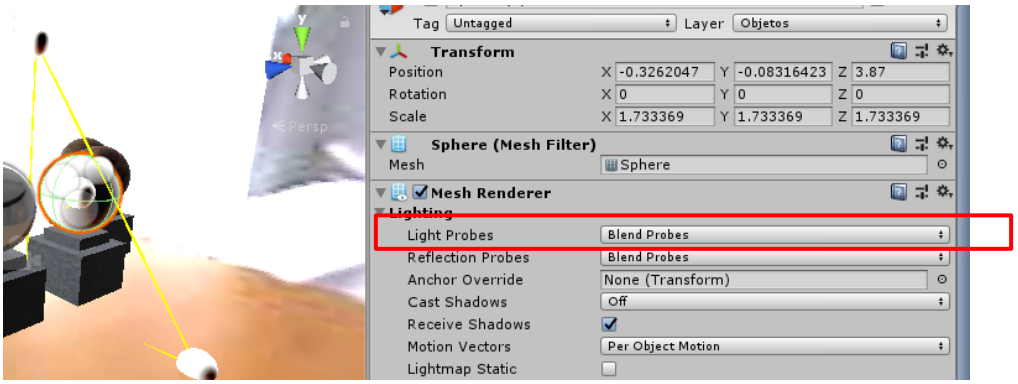


Figura 22. Configuración de los Light Probe en los objetos

4.4.3. Reflexiones Especulares. Configuración de los Reflection Probe

Para proporcionar reflexiones de la escena en tiempo real, se usaron los Reflexion Probe que también se especificaron en el apartado de Tecnologías empleadas. En este caso, los Reflection Probe nos interesan que tengan reflexiones en tiempo real de los objetos que se encuentran en el entorno y del Skybox. De esta forma, los pasos que se siguieron para la creación y configuración del Reflection Probe fueron las siguientes:

- Se añadió a la escena un Reflection Probe al cual se le ajustó su tamaño y posición en función del objeto al que se le quería aplicar la reflexión. Los Reflexion Probe en Unity están modelados como una esfera totalmente especular que la envuelve una malla cúbica la cual proporciona los límites a los que afectará el Reflexion Probe. Cabe señalar que se puede usar un mismo Reflection Probe para varios objetos ya que se ha comprobado que añadiendo varios a la escena el sistema se ralentiza.
- Con respecto al tipo, se marcó que fuera en tiempo real para que se actualizase para cada frame.
- Otro aspecto que se puede modificar es la Resolución. En este caso depende del tipo de efecto que queramos conseguir en el material del objeto. Si disponemos de un material que su superficie no es perfectamente pulida si no que tiene una superficie más rugosa o que tiene algún tipo de textura aplicada, podríamos aplicar una resolución menor para el mapa ya que no necesitamos que se definan o identifiquen a la perfección los objetos que se reflejan. En cambio, si se dispone de una superficie que se quiera que disponga de reflexiones perfectas, entonces sí que habría que disponer de una resolución mayor.

- También hay que marcar la opción del “Clear Flags” como Skybox. Esta opción es la que permite especificar cómo se llenarán las áreas en las que no se encuentren objetos sintéticos en la escena, si será con un color sólido o con el Skybox.

En el caso de los Reflection Probe no hace falta habilitar ninguna opción en los objetos a los que se quiere que afecten las reflexiones que calcula. En este caso, basta con que el objeto se incluya dentro de los “límites” marcados.

4.5. Cálculo de fuentes de luz para la generación de sombras

Para el cálculo de las fuentes de luz y así poder generar sombras en la escena, es preciso colocar alguna fuente de luz en el entorno de Unity. En este caso la posición de las fuentes debe de ser coherente y corresponder con la procedencia de las luces en el entorno para que los objetos sintéticos generen sombras como lo harían si fueran reales. Además, las fuentes de luz se deberán de actualizar en cada frame. El tipo de fuentes que utilizaremos en este caso para generar sombras, son fuentes direccionales, tal y como indican en [22], estudio en el que nos basamos para la detección de la procedencia de la luz. En él detectan las zonas de los pixeles del mapa de entorno con un alto valor de luminosidad y los convierten en luces direccionales. A continuación, se muestra en la Figura 23 un diagrama con las partes más importantes del algoritmo que se ejecuta para cada frame y que posteriormente se irán comentando.

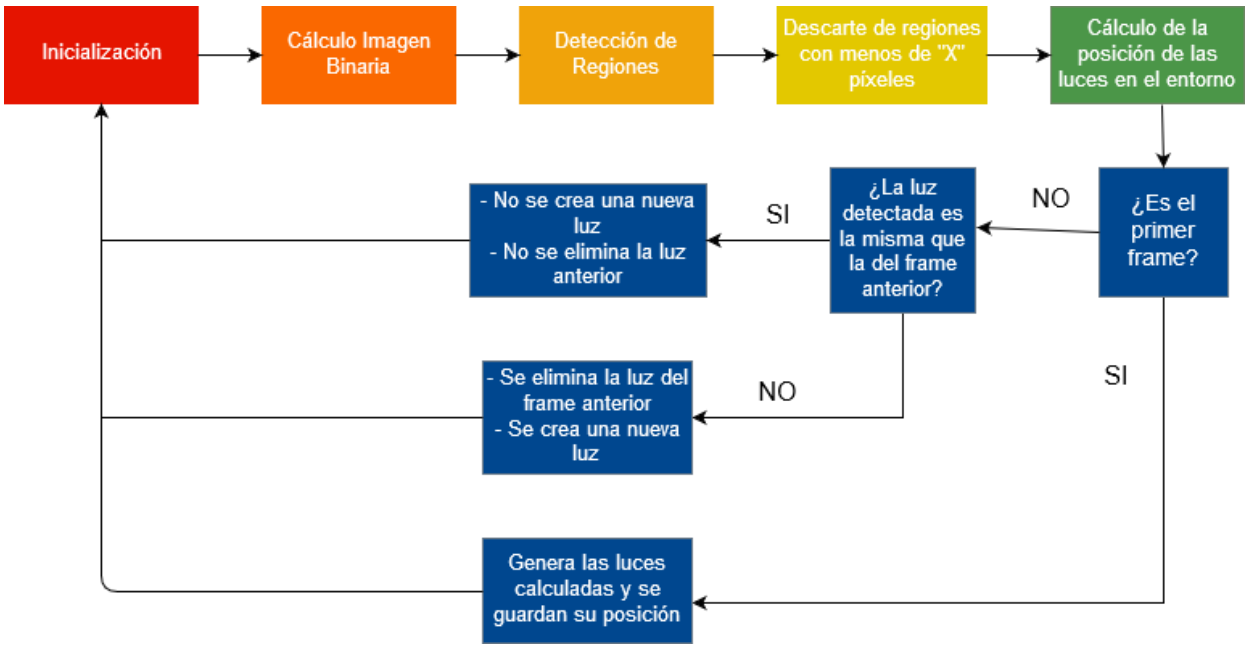


Figura 23. Diagrama del algoritmo de detección de luces

4.5.1. Inicialización.

El primer paso consiste en recibir la textura del mapa de entorno. Recordemos que esta es devuelta por la cámara 360 y que se le ha hecho una transformación a HDRI mediante Inverse Tone mapping. Esta textura consistirá en una RenderTexture, pero en este caso necesitamos que sea una textura 2D para poder extraer los valores de los pixels, así que se realiza una transformación de una a otra. Tal y como comentan en [20] a la hora de binarizar la imagen, había píxeles sueltos que se consideraban fuentes de luz. Este problema lo solucionaban aplicando un filtro Blur a la imagen HDRI y trabajando con ella a continuación, así que en realidad la imagen con la que se parte, no es la textura del mapa de entorno en sí, si no esta misma pasada por un filtro.

Cabe señalar que la resolución de la imagen que devuelve la cámara es de 1920x960 píxeles. Más adelante se verá que hay que realizar cálculos para todos los pixels de la imagen lo que supone un proceso costoso. De esta forma, en vez de trabajar con la imagen completa se trabaja con un mipmap[16]. En este caso el mipmap con el que se decide trabajar es de orden dos, lo que supone reducir a la mitad el tamaño de la imagen dos veces, por lo que nos quedamos con una textura final de 480x240 pixels, de forma que reducimos también los tiempos de cómputo.

En [22], analizan que la solución propuesta detecta posibles fuentes de luz en superficies donde la luz del entorno es altamente reflejada, como por ejemplo en mesas. Como solución proponen que la luz solo se pueda detectar de la mitad de la imagen hacia arriba donde se suele encontrar la mayoría de los focos de luz sacrificando posibles fuentes que vengan desde abajo. Consecuentemente, al final solo se trabajará con imágenes de 480x120 pixels quedándonos únicamente con la mitad superior.

4.5.2. Binarización

A partir de la imagen HDR se crea una máscara para todos los pixels pertenecientes a una fuente de luz. Para saber si un pixel pertenece a una fuente de luz este tendrá que superar el umbral dado por la Ecuación 4. De esta forma se crea una imagen binaria donde todos los pixels que han superado el umbral se ponen a uno y el resto se quedan a cero. Esta imagen es la que servirá como punto de partida para calcular los puntos de donde proviene la luz.

[16] Colecciones de imágenes de mapas de bits que acompañan a una textura principal donde cada imagen es una versión reducida de la textura principal.

4.5.3. Detección de las Regiones

El siguiente paso consiste en clasificar los píxeles que han superado el umbral en diferentes regiones, ya que cada una de ellas se corresponderá con una fuente de luz. Cabe señalar que en este caso se entiende que, una región es un subconjunto de pixels que han superado el umbral y que están conectadas. Para saber si un píxel pertenece o no a una región, se ha implementado un algoritmo de "Connected-component labeling" donde los pixels conectados entre sí se marcan con una misma etiqueta. De esta forma, se construirá una estructura donde los pixels que han superado el umbral tengan la etiqueta de la región a la que pertenecen. El algoritmo implementado recibe el nombre de "Two-Pass" o "Hoshen-Kopelman algorithm" [27] que itera a través de estructuras bidimensionales binarias, como en este caso la imagen binaria que calculamos anteriormente. El algoritmo realiza dos pasadas sobre ella: la primera es para asignar etiquetas temporales a los pixels y guardar las equivalencias entre ellas y la segunda es para reemplazar esas etiquetas temporales por la etiqueta más pequeña entre sus equivalentes.

Para comprobar si un pixel está conectado a otro, se verifica las etiquetas de los pixeles vecinos. En el algoritmo implementado, se comprueban las etiquetas de 8 pixeles vecinos.

En caso de que uno de los píxeles vecinos esté etiquetado, al píxel actual se le asigna esa etiqueta. Hay ocasiones en las que un píxel puede tener dos o más vecinos con diferentes etiquetas. En esa situación, todas esas etiquetas pertenecen a una misma región por lo tanto deberán de fusionarse, se registra que son equivalentes y más adelante se les asignará a todas la misma etiqueta (la de menor valor). Por otro lado, si se encuentra un píxel cuyos vecinos no tienen etiquetas a este se le asigna una nueva.

Cuando todos los píxeles tienen una etiqueta asignada, se realiza la fusión de aquellos que se han marcado como equivalentes. Para esto se utilizan las funciones Union y Find. La función Find se utiliza para determinar si una etiqueta es la misma que otra, es decir, si dos etiquetas pertenecen a una misma región. Por otro lado, la función Union se usa para unir dos subconjuntos en un único conjunto es decir, dos etiquetas en una.

A continuación se presenta el pseudocódigo de esta sección.

```
deteccionRegiones(imagenBinaria)

     etiquetasVinculadas = Píxeles que contienen una etiqueta determinada
```

```
     imgEtiquetada = Es una imagen con la misma dimensión que la imagen
binaria inicializada a cero.
     etiqueta= 1 // Nombre de la etiqueta
    //Primer pase
    For fila in imagenBinaria
      For columna in imagenBinaria
        If imagenBinaria[fila][columna] == 1:
           Vecinos = pixels conectados
           If vecinos es vacío
              etiqueta = etiqueta +1 // Pasamos a la siguiente etiqueta
              etiquetasVinculadas[etiqueta] = set que contiene esta etiqueta
              imgEtiquetada[fila][columna] = etiqueta
           else
           //buscamos la etiqueta menor:
              L = etiquetas de vecinos
              imgEtiquetada[fila][columna] = min(L)
              for etiqueta in L
                etiquetasVinculadas[etiqueta] = Union(etiquetasVinculadas, L)

    //Segundo pase
    For fila in imagenBinaria
      For columna in imagenBinaria
         If imagenBinaria[fila][columna] == 1:
         imgEtiquetada[fila][columna] = Find(imgEtiquetada[fila][columna])

return imgEtiquetada
```

4.5.4. Descarte de regiones muy pequeñas

En las pruebas realizadas, cuando las regiones eran detectadas en la imagen binaria, se daba el caso de que a veces se obtenían algunas con muy pocos píxeles. Estas zonas detectadas se debían a reflejos en alguna superficie de los verdaderos focos de luz (en entornos cerrados) o simplemente a pequeños puntos de luz que se

detectaban a la distancia (en entornos abiertos) y realmente no deberían de afectar a la generación de sombras del objeto.

Para eliminarlas y no tenerlas en cuenta como posibles focos de luz, se ha implementado un parámetro que establece el límite de número mínimo de píxeles que deberán de tener las regiones. Por lo tanto, para cada región detectada es necesario conocer cuántos píxeles tiene, ya que, si no cumple el límite establecido, se eliminará. Se ha comprobado que para los diferentes escenarios este parámetro puede variar por lo que se le permite al usuario cambiarlo dependiendo del escenario en el que se encuentre. Cabe señalar que, en el apartado de ajustes de parámetros de esta memoria, se presentan una serie de pruebas que se han realizado en diferentes escenarios y se propone un valor del mínimo número de píxeles por región.

4.5.5. Cálculo de la posición de las fuentes de luz virtuales

Tras detectar las regiones que finalmente se consideran fuentes de luz, el siguiente paso consiste en calcular sus propiedades. Cabe recordar que se usan fuentes direccionales, por lo que su posición en el mundo no importa, sino la dirección de sus rayos. El otro parámetro a calcular es su intensidad. Para ello se siguientes los siguientes pasos:

1. Para cada píxel que se considere una región:
    a. Se determina su ángulo solido mediante la Ecuación 6.
    b. Se calcula la contribución a la irradiancia de la fuente de luz mediante la fórmula de la Ecuación 5.
    c. Se calcula la posición del píxel en coordenadas esféricas según la ecuación Ecuación 9.
2. Para cada región:
    a. Se calcula el sumatorio de las irradiancias de todos los pixeles pertenecientes a la región.
    b. Se calcula la posición de la fuente de luz según la Ecuación 8 en coordenadas esféricas
    c. Se calcula el valor de la intensidad o dureza que tendrá la fuente de luz y que vendrá dado por el sumatorio comentado en 2.a.

Cabe destacar que el valor de la intensidad de la luz es multiplicado por un factor de escala ya que si no las sombras se dibujarían muy intensas en relación con las sombras que se generarían en el entorno real. Este factor de escala es otro parámetro que es ajustable por el usuario ya que dependerá en gran medida del tipo de entorno en el que

se ubique la escena. Aun así en la siguiente apartado, se determina un valor estimado para cada entorno que se ha calculado realizando pruebas en diferentes escenarios.

4.5.6. Creación de las fuentes de luz

En las primeras versiones que se diseñaron de esta parte del algoritmo, se construían las fuentes de luz para cada frame y se destruían las detectadas en frames anteriores. Sin embargo, esto generaba un parpadeo en el dibujo de las sombras que no era realista. Por esa razón se ha diseñado un algoritmo para la creación de las fuentes de luz donde se tiene en cuenta las fuentes detectadas en el frame anterior. En este caso, si nos encontramos en el primer frame de la escena que se renderiza, se crean las luces encontradas. Si por el contrario, nos encontramos en otro frame entonces hay que tener en cuenta si la luz nueva que se ha detectado es la misma o muy parecida a la detectada anteriormente ya que si es nueva, se dibuja, si no, se queda la del frame anterior. Para saber si una fuente de luz calculada en el frame actual es la misma que alguna del frame anterior, se mira si la posición de la nueva luz se encuentra entre unos límites. Los límites vienen marcados por el centroide de la fuente de luz anterior y un radio. El radio es otro parámetro que se permite modificar ya que depende en gran medida de si tenemos en el entorno luces que se mueven o que permanecen estáticas. Por ejemplo, si una linterna se mueve alrededor de un objeto sintético, el radio debe de ser pequeño para que se puedan detectar las variaciones de movimiento de forma fluida. Si por el contrario nos encontramos con una luz que permanece estática en la escena un radio mayor sería el recomendado para que no se produjecen parpadeos indeseados.

Cada vez que se crea una nueva fuente de luz direccional se siguen los siguientes pasos:

1. A partir de las coordenadas esféricas halladas en el paso anterior, se calculan las coordenadas en el espacio 3D.
2. Se crea un objeto de fuente de luz direccional que apunte al origen del sistema de coordenadas, que es donde se ubicará la escena.
3. Se coloca en la posición calculada en el punto 1.
4. Se le da la intensidad calculada en el paso anterior

Para calcular las coordenadas x,y y z se hace uso de las formulas de la Ecuación 10 donde φ es el azimut y θ la colatitud calculadas anteriormente y el radio es la distancia

al centro. Este radio no es significativo ya que la distancia de las fuentes direccionales a los objetos no es relevante a la hora de iluminarlos ya que afecta de la misma forma a menor o mayor distancia.

$$x = radio \cdot \sin\theta \cdot \cos\varphi$$
$$y = radio \cdot \cos\theta$$
$$z = radio \cdot \sin\theta \cdot \sin\varphi$$

Ecuación 10. Cálculo de las coordenadas X, Y, Z de las luces en el entorno de Unity

Cabe señalar que el origen del centro de coordenadas del sistema se encuentra justo en la posición del ModelTarget. De esta forma el sistema de coordenadas y la dirección de los ejes queda definido de la forma que se muestra en la Figura 24.

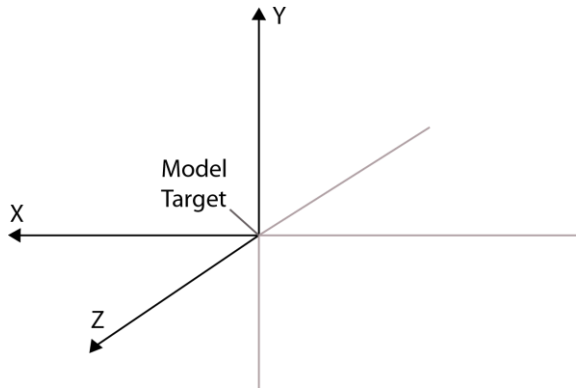


Figura 24. Sistema de coordenadas y dirección de los ejes.

4.6. Proyección de las sombras

A la hora de generar sombras, es necesaria una superficie donde poder proyectarlas. Por ello, como los objetos virtuales se muestran sobre superficies reales, es necesario introducir una superficie ficticia donde generarlas. En este trabajo las sombras se proyectan en un plano que tendrá asociado un material que hará que su superficie sea transparente, pero que sí dibuje las sombras. Al ser el plano transparente, dejará ver lo que hay debajo, es decir, la superficie del entorno real y las sombras sintéticas se compondrán en la superficie del entorno real. Cabe señalar que en este caso se incluye un plano pero podría generarse otra geometría en la que se quiera proyectar las sombras en el entorno. El problema surge que se desconoce tanto las superficies reales donde deben proyectarse como los materiales de estas, por eso en este trabajo se presupone que todas las superficies donde se dibujarán los objetos sintéticos son planas y no tendrán ningún elemento a su alrededor que distorsione el dibujo de la sombra. En futuros trabajos se podría usar alguna técnica para reconocer

la geometría del entorno y los materiales de estas, y así proyectar las sombras en función de la información recogida.

4.7. Ajuste de parámetros

A lo largo de esta sección, se han mencionado tres parámetros cuyo valor varía dependiendo del tipo de entorno en el que se quieran renderizar los objetos. Estos tres parámetros son: el factor de escala que multiplica la dureza con la que se dibujan las sombras, el límite del número de píxeles que se pueden considerar una región y el radio que determina si hay un nuevo foco de luz o es igual que alguno detectado en el frame anterior. A continuación, se muestran las pruebas realizadas, se comentan los diferentes resultados y se analiza cual es el valor óptimo para cada parámetro dependiendo del tipo de escenario.

4.7.1. Dureza del dibujo de las sombras

Hay dos valores que están ligados a la dureza con la que se dibujan las sombras: la irradiancia de la fuente de luz en el entorno real y la luminosidad que existe en el entorno. Por ejemplo, una sombra que se proyecta por un único punto de luz es más fuerte que la sombra que se genera en un entorno al aire libre. Como se comentó anteriormente, la irradiancia de la fuente es la que nos provee la intensidad, pero hay que aplicarle un factor de escala ya que ese valor por si solo dibuja sombras muy duras y oscuras.

Se ha decidido analizar para diferentes escenas qué valor de umbral de luminancia se tiene y para ese umbral qué factor de escala dibuja sombras más realistas. De esta forma, se recoge información de los ocho escenarios diferentes que se muestran en la Figura 25.

(a) Escena completamente exterior

(b) Escena semi exterior

(c) Escena interior con ventana y foco de luz

(d) Escena interior oscura con foco de luz

(e) Escena interior con ventana

(f) Escena interior con ventana alejada

(g) Escena interior con iluminación natural

(h) Escena interior con iluminación artificial

Figura 25. Escenarios donde se realizaron las pruebas.

Para evaluar la cantidad de iluminación que hay en el entorno, se ha decidido usar el umbral que se calcula para binarizar la imagen ya que es una buena medida de la irradiancia que generan las fuentes principales de luz en las escenas. De esta forma, para cada tipo de escenario se han recogido 100 valores del umbral y se ha sacado la media de todos ellos los cuales se representan en la Tabla 1.

Escenario	**A**	**B**	**C**	**D**	**E**	**F**	**G**	**H**
Media del umbral	6,8154	8,8037	5,7497	4,1644	6,7234	2,6245	6,9206	3,0897

Tabla 1. Valores de las medias de los umbrales para 8 entornos diferentes

De la Tabla 1 podemos deducir que los entornos A, E y G, que tienen un umbral parecido, son escenas donde hay bastante iluminación. La escena B, que presenta un umbral elevado en comparación con el resto, nos informa de que en ese escenario había una gran cantidad de iluminación, al contrario que la escena F, donde el umbral es muy bajo por la distancia que existe con la principal fuente de luz de la escena (ventana muy alejada). Por otro lado, el umbral de las escenas C, D y E, que se encuentran entre 3 y 5, nos informa de que la cantidad de iluminación de la escena es notable pero que hay algún foco de luz artificial. Estas conclusiones se han corroborado comprobando que esto es así para otras escenas donde se simulaban situaciones parecidas.

El siguiente paso fue generar imágenes compuestas de cada escenario variando el valor del factor de escala. Un ejemplo son las imágenes que se muestran en el Anexo II, donde el factor de escala se fue variando para el entorno “g” mostrado en la Figura 25.

Para cada escenario, se buscó el valor del factor de escala que quedaba mejor comparando las capturas generadas con la captura de la escena sin objetos virtuales, únicamente con la cámara 360 de forma que se pudiese comparar la sombra de esta con la de los elementos sintéticos. De esta forma, los resultados obtenidos fueron los que se muestran en la siguiente tabla, donde se recogen los valores de escala que se proponen para cada tipo de escenario.

Escenario	**A**	**B**	**C**	**D**	**E**	**F**	**G**	**H**
Umbral	6,8154	8,8037	5,7497	4,1644	6,7234	2,6245	6,9206	3,0897
Factor de escala	0.08	0.04	0.08	0.6	0.08	0.04	0.08	0.3

Tabla 2. Resultados de los valores seleccionados del factor de escala de la intensidad para diferentes escenarios

De todos estos resultados, se dedujo que para variaciones del umbral que se encuentran entre 2 y 3 el factor de escala de las sombras sea de 0.04, para escenas donde el umbral esté entre 3 y 5 que el factor sea de 0.5, para entornos con un umbral entre 5 y 8 que el factor sea de 0.08. Cabe señalar que a pesar de que este tipo de medidas se han realizado en diferentes entornos, el mundo real es muy imprevisible y las casuísticas pueden ser tantas que esto es solo una pequeña aproximación. En el apartado de resultados de la memoria se muestran los objetos en otros entornos donde se puede ver que los parámetros hallados no siempre son los ideales y por lo tanto se precisa realizar un ajuste por parte del usuario.

4.7.2. Límite del número de píxeles de una región

Hay escenarios en los que, debido a superficies reflectantes, el algoritmo detecta pequeñas regiones como fuentes de luz. Además, también hay ocasiones en las que se detectan píxeles sueltos como regiones. Para solucionarlo, se propone establecer un número mínimo de píxeles por región.

Para establecer ese valor, se han extraído de las escenas el número de regiones que se detectan con la cantidad de píxeles que tienen y si estas regiones son o no fuentes de luz. Estas medidas quedan recogidas en el Anexo III.

De estos resultados podemos deducir que un buen valor para eliminar regiones con pocos píxeles es un valor mínimo de 120. Sin embargo, establecer un valor para eliminar las regiones que son reflejos producidos en escenas interiores es más complicado, por eso se establece un parámetro que permita modificar al usuario el número de píxeles mínimo que deberán de tener las regiones para que sean consideradas fuentes de luz. Hay situaciones, en las que incluso eliminar los reflejos es complicado. Un claro ejemplo es el del caso del escenario H, donde se puede ver en la Tabla del Anexo III que se detectan regiones de reflejos cuyo número de píxeles es muy parecido al de la luz que hay en la escena. En este caso el usuario debe de hacer un ajuste fino y establecer un valor de número de píxeles muy próximos a la región de la fuente de luz.

4.7.3. Radio para saber si hay un nuevo foco de luz.

Hay ocasiones en las que nos encontramos con regiones muy grandes donde la posición de la luz que se calcula varia ligeramente, lo que genera que se detecte una nueva

fuente de luz cuando realmente es la misma. Por ejemplo, en la Figura 26 se puede ver como en una misma región (en una parte de la ventana), se detecta un punto de luz en una posición para un frame y otro punto de luz para el siguiente frame.

Figura 26. Región con un punto de luz detectado en un frame y otro punto de luz diferente detectado en otro frame

Esto generaba parpadeos en el dibujo de las sombras en la aplicación. Para solucionar este problema se debe de establecer unos límites para que el algoritmo detecte que son la misma luz de dos frames consecutivos. Los límites vendrán marcados por el centroide (posición de la luz) en la región y un radio a su alrededor que, en este caso, es el parámetro que se calcula en esta sección. El radio dependerá de si lo que se encuentra en la escena son luces estáticas o estas se mueven. En caso de ser luces estáticas, entonces se puede marcar un radio bastante mayor en comparación con una luz que se mueve cuyo radio deberá de ser menor para detectar las variaciones de movimientos. Se ha establecido que, para escenarios donde las luces del entorno se mantienen fijas, el radio se puede establecer entre 20 y 50. Si hay alguna luz que se mueve por la escena, el radio en este caso podría estar entre 2 y 5. Para encontrar estos valores se estudió el funcionamiento del algoritmo para diferentes casos. En escenas donde había luz estática se comenzó dando valores al radio muy pequeños y se fueron aumentando hasta que se eliminó el parpadeo. Para entornos donde se movía una luz, se realizó el mismo experimento, dando valores pequeños al radio hasta que se comprobó que no se producían parpadeos y era capaz de detectar el movimiento de la luz.

5. Resultados

En este apartado, se muestran una serie de resultados donde se observa el sistema en funcionamiento con la iluminación de varias escenas en diferentes entornos. Con las imágenes solo se muestra si los objetos están correctamente iluminados. Para mostrar el funcionamiento del sistema en tiempo real, se aporta un video que se puede encontrar en el siguiente enlace *http://bit.ly/2PvEd0P*. En el video se ha grabado el sistema reaccionando a diferentes escenarios en tiempo real. Se muestra además una escena iluminada con un foco que se va moviendo alrededor de esta y como se generan las sombras correspondientes.

Una de las escenas para la que se va a comprobar el funcionamiento es la que se muestra en la Figura 27 donde se pueden ver seis esferas con diferentes texturas.

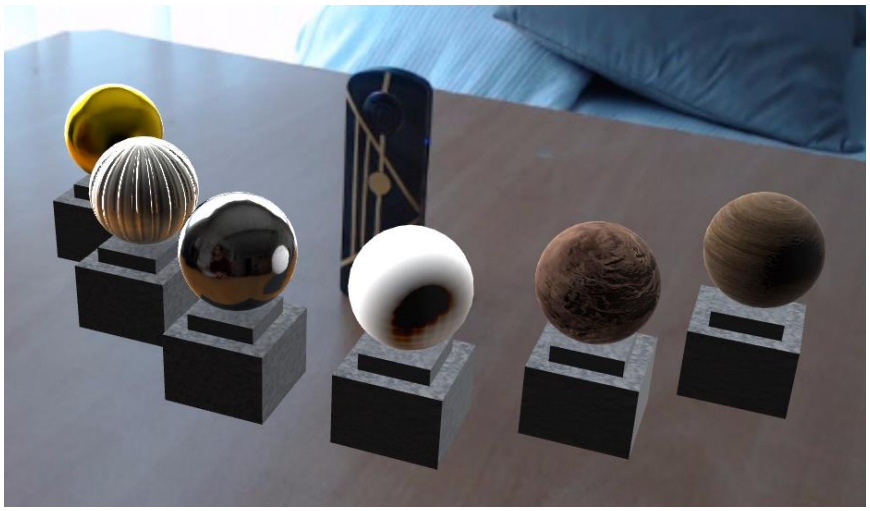


Figura 27. Escena que se ha utilizado para las pruebas y resultados (sin sombras)

Las esferas de la izquierda muestran materiales más especulares. La que está más atrás simula el material del oro, la de en medio un metal y la que se ubica delante de la cámara tiene asociado un material que simula una bola completamente reflectiva (como un espejo). Por otra parte, las esferas de la derecha muestran materiales más difusos. La esfera de atrás tiene un material que simula la madera, la esfera central un material que simula la piedra y la esfera delantera un material blanco completamente difuso.

Otra escena para la que se mostrará algunos resultados es la que se muestra en la Figura 28 donde se observa la escultura del Ángel Lucy de Standford. El ángel tiene un material completamente difuso asignado. En esta imagen no se muestran ni sombras ni

se tiene en cuenta la luz del entorno, es decir, se representa el objeto como lo haría en una aplicación normal de Realidad Aumentada.

Figura 28. Escultura del Ángel Lucy de Standford sin iluminación del entorno ni sombras.

Para comprobar que se recoge información de la iluminación difusa del entorno, se acerca un material completamente difuso, en este caso de color rojo. Se puede apreciar en la imagen de la derecha de la Figura 29 como la escultura cambia su tonalidad debido a la transferencia de color (color blending) en comparación con la imagen de la izquierda donde no se acerca ningún material y el objeto queda iluminado con la luz del entorno.

Figura 29. A la izquierda la escultura con la iluminación del entorno y a la derecha la escultura rodeada con un material completamente difuso de color rojo

En la imagen de la Figura 30 se puede comprobar como las esferas con materiales más especulares, también reflejan el entorno. Si nos fijamos en la esfera más especular podemos ver que esta se comporta prácticamente como un espejo ya que el entorno se ve reflejado totalmente en ella.

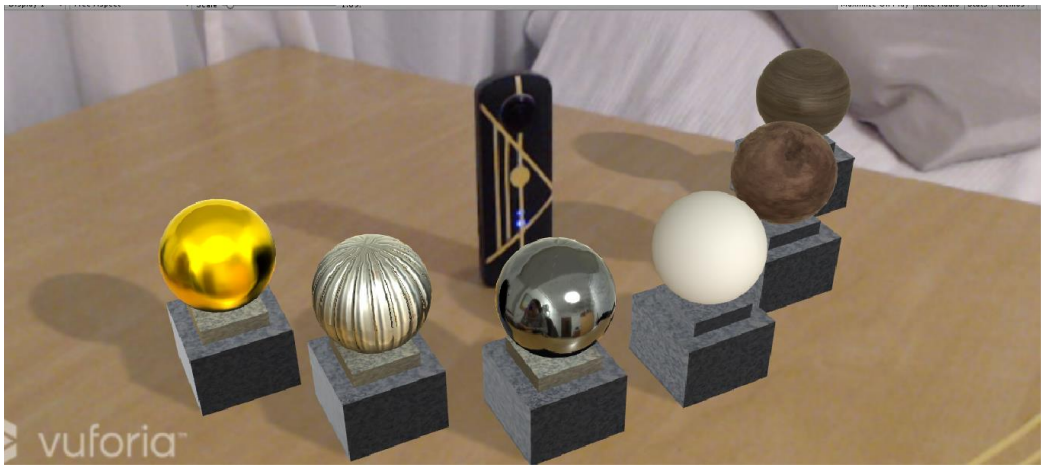


Figura 30. Imagen con escenas de materiales especulares que reflejan el entorno

Cabe mencionar que los objetos sintéticos que tienen un material altamente especular deberán de estar próximos a la cámara de captura del mapa de entorno ya que, si no, producirían reflejos que no son correctos. Esto se debe principalmente a que en el sistema que se presenta, no se aplica ningún tipo de offset a la imagen 360 para tener reflejos en objetos desplazados. Un ejemplo de ello se puede ver en la escena de la Figura 31 donde se observa que cuanto más alejadas se encuentran las esferas de la cámara, menos se corresponden los reflejos. Por ejemplo, la esfera que se encuentra ubicada en el aire, no refleja la figura de papel de frente (que sería lo correcto) si no que lo hace desde abajo tal y como capta la cámara 360 el entorno.

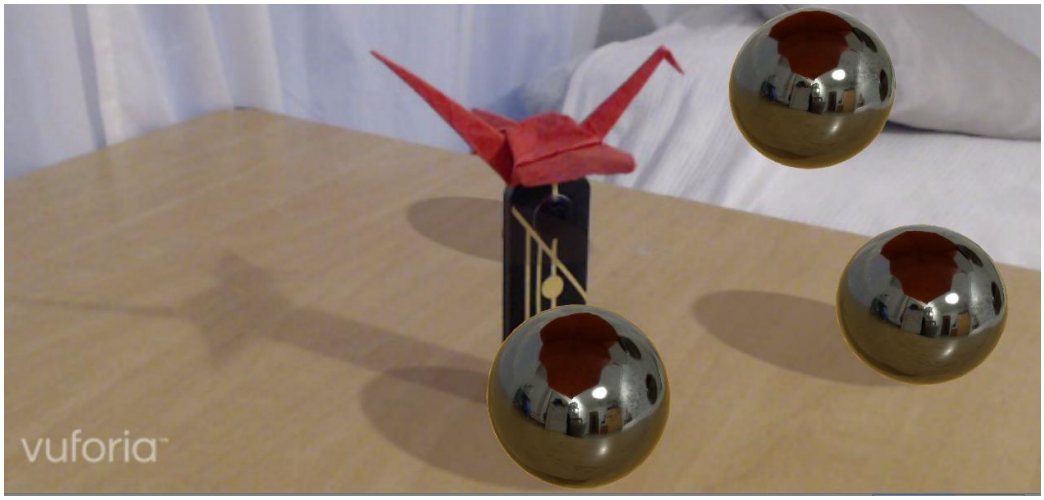


Figura 31. Esferas especulares con reflejos erróneos

En la Figura 31 también se puede apreciar otro de los aspectos que no se contemplan en este primer prototipo del sistema. La sombra proyectada por la esfera ubicada en el aire no tiene en cuenta la geometría de la cámara 360 que se encuentra ubicada en el entorno y no se dibuja detrás de esta, que sería lo que ocurriría en la realidad. Esto

ocurre porque en el sistema las sombras solo se proyectan en superficies planas y no tienen en cuenta la geometría del entorno.

Por otro lado, también se comprueba el correcto funcionamiento de las sombras en el entorno. En la Figura 32 se puede ver cómo quedan estas dibujadas con los valores de los parámetros propuestos en apartados anteriores (en este caso un valor del factor de escala de la dureza de las sombras de 0.08 , un límite de píxeles por región de 100 y un radio de 30).

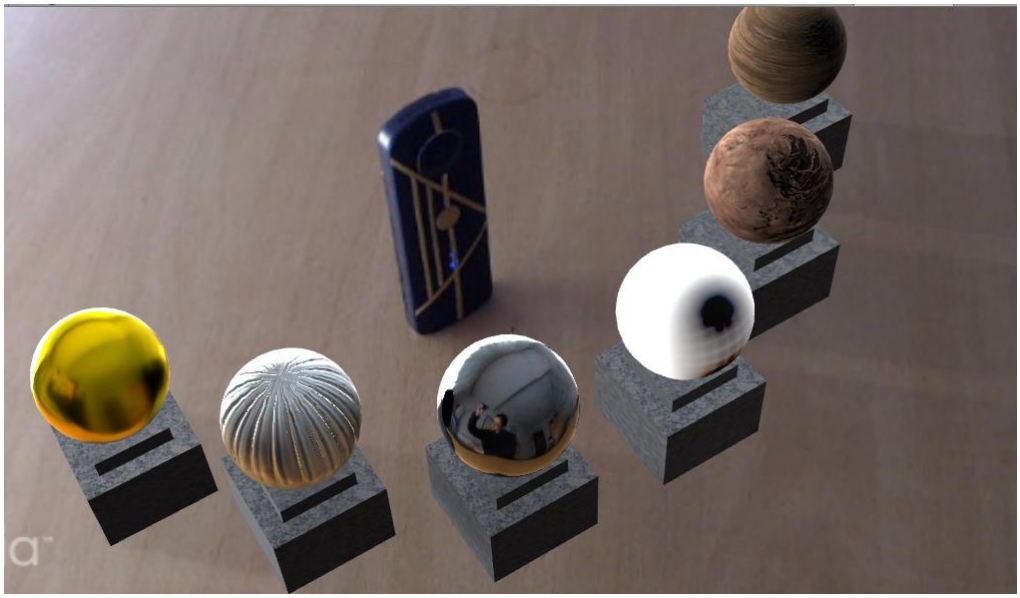


Figura 32. Sombras generadas con parámetros por defecto

En este caso, las sombras se generan con la misma dirección que la sombra que proyectarían los objetos reales (cámara 360 en la imagen) y con una intensidad similar. Cabe señalar que en la misma imagen se aprecia como la sombra de la cámara 360 es más intensa cerca de esta y más difusa cuanto más alejada está del objeto. Tal y como se puede ver en la Figura 32 las sombras sintéticas que se proyectan no generan este efecto.

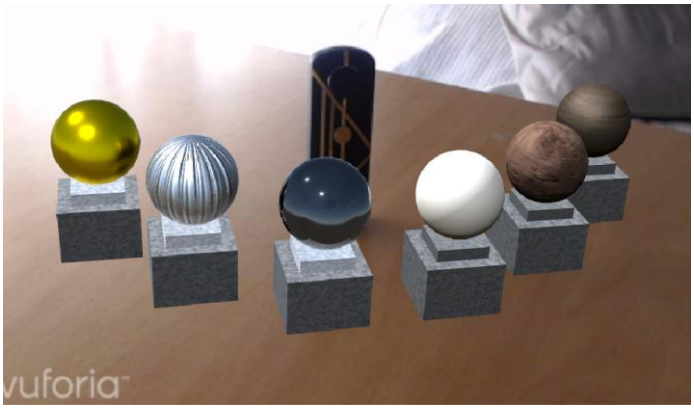
En Figura 33 se puede ver también como se dibujan las sombras para la escena del ángel con otras condiciones lumínicas. En este caso, se ha usado un factor de escala de la dureza de las sombras de 0.3, un valor mínimo del número de píxeles de 200 y un radio de 2, ya que para la captura de esta escena se estaba moviendo una fuente alrededor del objeto. Cabe destacar que el sistema generaba un factor de dureza de las sombras mucho más bajo y se tuvo que incrementar de forma manual para que las sombras fuesen más acordes con las que se generarían en el entorno real. Este es un claro ejemplo de que en trabajos futuros deberían de ajustarse los parámetros teniendo

en cuenta más escenas. Tal y como se aprecia en la imagen, en este caso se dibujan dos sombras dado que hay dos fuentes de luz en la escena.

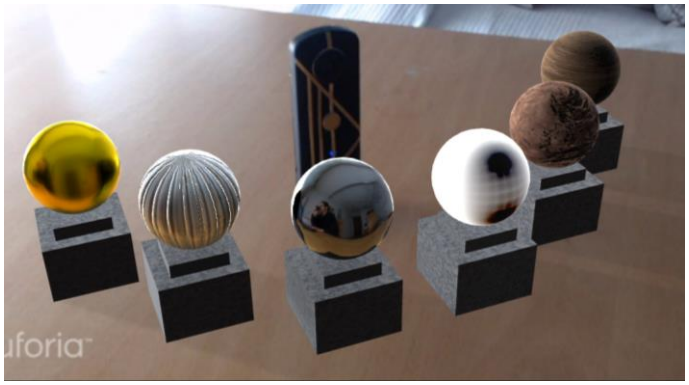
Figura 33. Sombras proyectadas con dos fuentes de luz reales

Nuevamente las sombras que proyecta la cámara 360 (objeto real en la escena), nos dan información en la imagen de cómo son las sombras para un entorno específico. Podemos ver que las sombras proyectadas por el objeto sintético se generan en la misma dirección, sin embargo, se puede comprobar que la sombra dibujada al lado derecho de la imagen debería de ser más intensa, como la que proyecta la cámara.

Si un objeto sintético se representase en un escenario real sin tener en cuenta las luces del entorno, posiblemente se dibujaría como se ve en la Figura 34 (a) y en la Figura 34 (c). Como se aprecia, el resultado es el de objetos que parecen que no están ubicados en la escena ya que no están iluminados de forma coherente con el entorno. Por el contrario, en la Figura 34 (b) y Figura 34 (d) se muestran los mismos objetos sintéticos ubicados en la misma escena pero con iluminación acorde con la del entorno. Tal y como se puede apreciar, al incluir una iluminación acorde se aumenta la sensación de que los objetos sintéticos se encuentran ubicados en la escena.



(a)



(b)

(c)

(d)

Figura 34. A la izquierda objetos sintéticos ubicados en la escena real sin tener en cuenta la iluminación del entorno y a la derecha los objetos iluminados según las luces reales

De todos estos resultados podemos deducir lo siguiente:

1. Las reflexiones difusas de los objetos del entorno real son recogidas y consideradas a la hora de iluminar los objetos sintéticos.
2. Se obtienen reflejos especulares del entorno real pero, los objetos con materiales especulares deben estar próximos a la cámara, ya que no se ha incluido un offset para poder trasladar esa reflexión.
3. Las sombras se generan en la dirección correcta, sin embargo, hay escenas donde se precisa una variación de parámetros por parte del usuario para que estas sean más realistas. Además, las sombras tienden a ser más difusas cuando se alejan de los objetos y las que se generan en este caso, se mantienen igual.
4. Por otro lado, las sombras también deberían de proyectarse teniendo en cuenta la geometría del entorno y que no solo se dibujasen en una superficie plana.

5. El sistema implementado ofrece resultados más realistas que los ofrecidos por un sistema convencional de Realidad Aumentada, ya que se ha conseguido que los objetos sintéticos tengan una iluminación acorde con la del entorno real.

6. Conclusiones

En este proyecto se ha realizado, en primer lugar, un estudio sobre las técnicas de composición de objetos sintéticos en escenas reales. Hay tres campos en los que se utilizan principalmente estas técnicas: la televisión, el cine y la Realidad Aumentada. Para alcanzar el nivel de calidad esperado en el medio, tanto en el cine como en la televisión se utilizan técnicas que permiten crear la ilusión de que los objetos sintéticos están en la escena real, y para ello simulan que están iluminados por las luces de dicha escena. Este proceso suele hacerse en postproducción, lo que incrementa los costes de generación de contenido. Por otro lado, en el campo de la Realidad Aumentada aún no existe un sistema que pueda construir una iluminación de los objetos sintéticos que se corresponda con la de la escena real, reflejando y proyectando sombras en función de esta.

En este Trabajo Fin de Máster se ha propuesto un sistema que visualiza objetos sintéticos ubicados en escenas reales y que presentan una iluminación acorde con la del entorno, reflejando la escena y proyectando sombras en tiempo real.

En los resultados presentados en este trabajo se muestra que el sistema propuesto ilumina a los objetos de forma coherente con la escena real, ya que reflejan los objetos reales del entorno y proyectan sombras sobre el mismo teniendo en cuenta la procedencia de las fuentes de luz. Esta coherencia de la iluminación de los objetos sintéticos con el resto de la escena permite crear la sensación de que realmente se encuentran ubicados en ella. La herramienta desarrollada, además funciona en tiempo real, como se ha mostrado en el vídeo que acompaña a este trabajo. Por todo lo anterior, consideramos que se han alcanzado los objetivos propuestos en la introducción.

Finalmente, del desarrollo de este proyecto se ha conseguido experiencia en el campo de la iluminación en gráficos por computador, ya que el sistema que se presenta se basa en una técnica (iluminación basada en imagen) de la que no se tenía conocimiento previo. Así mismo, también se ha conseguido experiencia en las herramientas que se han usado a lo largo de proyecto (Vuforia y Unity).

7. Trabajos Futuros

Tras el proyecto desarrollado, se presentan una serie de mejoras para posibles futuros trabajos.

Partiendo del sistema actual, se podrían realizar las siguientes mejoras:

- Mejoras en las medidas de los parámetros. En el presente proyecto se han recogido medidas de ocho escenarios diferentes, pero para ajustar aún más los valores, sería conveniente recoger más medidas.
- Dibujado de las sombras en función de la superficie donde se encuentra la escena real. En el prototipo presentado las sombras solo se dibujan en superficies planas horizontales, de forma que si se quisiese proyectar las sombras de los objetos virtuales en otro tipo de superficie habría que conocer la geometría de esta. Por ello se propone usar alguna técnica para recocer la geometría del entorno real y proyectar las sombras en función de esta.
- En el presente proyecto, la cámara 360 trabaja con la configuración automática donde la cámara va ajustando parámetros como la velocidad de obturación o la apertura de diafragma. Sería conveniente hacer pruebas con diferentes parámetros de esta para ver si se consiguen mejorar los resultados obtenidos.
- Ya que los objetos siempre han de generarse cerca de donde se encuentra la cámara 360 para que no haya discordancia por ejemplo en los reflejos, se propone utilizar algún método que permita visualizar correctamente objetos a una cierta distancia de la cámara que captura el mapa de entorno.
- Optimización de algunas características del sistema. Se ha comprobado que al incluir en la escena diferentes Reflection Probe (que hacen que los objetos especulares reflejen el entorno) y Light Probe (que hacen que los objetos recojan información difusa del entorno), el sistema empeora su funcionamiento en tiempo real. Se propone buscar algún método que permita solucionar este inconveniente.

8. Bibliografía

[1] C. M. Aniceto, Video Digital. Efectos Especiales, Paradimage Soluciones, 2012.

[2] M. Pierson, «CGI in Hollywood science-fiction cinema 1982-95: the wonder years,» *Screen,* vol. 40, nº 2, pp. 158-176, 1999.

[3] Y. Furukawa y P. Jean, «Dense 3D Motion Capture for Human Faces,» 2009.

[4] G. Esteban y F. Cesáreo, «La escenografía virtual en la retransmisión de grandes eventos,» *Revista Latina de comunicación Social,* vol. 66, p. 063 to 078, 2011.

[5] J. Carmigniani, B. Furht, M. Anisetti, P. Ceravolo, E. Damiani y M. Ivkovic, «Aumented reality technologies, systems and applications,» *Multimedia Tools and Applications,* vol. 51, nº 1, pp. 341-377, 2011.

[6] K. Willis, «Universidad de Alberta,» Enero 2018. [En línea]. Available: https://www.ualberta.ca/science/science-news/2018/january/augmented-reality-tech-see-under-skin-without-scalpel. [Último acceso: 6 Septiembre 2018].

[7] R. T. Azuma, «A Survey of Augmented Reality,» *Teleoperators and Virtual Environments,* vol. 6, pp. 355-385, 1997.

[8] E. Rublee, V. Rabaud, K. Konolige y G. Bradski, «An efficient alternative to SIFT or SURF,» *International Conference on Computer Vision,* pp. 2564-2571, 2011.

[9] T. Bailey, «Simultaneous localization and mapping (SLAM),» *IEE Robotics and Automation Magazine,* vol. 13, nº 2, p. 108.

[10] G. Klein y D. Murray, «Parallel tracking and mapping for small AR workspaces,» *Proceedings of the International Symposium on Mixed and Aumented Reality,* 2007.

[11] N. Kurachi y M. Stark, The Magic of Computer Graphics, CRC Press, 2011.

[12] M. Cohen y J. R. Wallace, Radiosity and Realistic Image Synthesis, Academic Press, 1993.

[13] P. Shirley, Realistic Ray Tracing, A K Peter, 2000.

[14] H. Wann Jensen, Synthesis Using Photon Mapping, New York: A K Peters, 2001.

[15] P. Debevec, «Image-Based lighting,» *IEEE Computer Graphics and Application,* vol. 22, nº 2, pp. 26-34, 2002.

[16] E. Reinhard, W. Heidrich, P. Debevec, S. Pattanaik, G. Ward y K. Myszkowski, High Dynamic Range Imaging. 2nd Edition. Acquisition, Display, and Image-Based Lighting, Morgan Kaufmann, 2010.

[17] I. Rasool Khan, S. Rahardja, M. Murtaza Khan, M. Mobeen Movania y F. Abed, «A Tone-Mapping Technique Based on Histogram Using a Sensitivity Model of the Human Visual System,» *IEEE Transactions on Industrial Electronics,* vol. 65, nº 4, pp. 3469-3479, 2018.

[18] P. Debevec y J. Malik, «Recovering high dynamic range radiance maps from photographs,» *SIGGRAPH'97,* vol. 1, pp. 369-378, 1997.

[19] F. Banterle, P. Ledda, K. Debattista y A. Chalmers, «Inverse tone mapping,» *In Proceedings of the 4th international conference on Computer graphics and interactive techniques in Australasia and Southeast Asia,* pp. 349-356, 2006.

[20] T. Rhee, L. Petikam, B. Allen y A. Chalmers, «MR360: Mixed Reality Rendering for 360° Panoramic Videos,» *IEEE Transactions on Visualization and Computer Graphics,* vol. 23, nº 4, pp. 1379-1388, 2017.

[21] R. Ramamoorthi y P. Hanrahan, «An efficient representation for irradiance environment maps,» *Proceedings of the 28th annual conference on Computer graphics and interactive techniques. ACM,* pp. 497-500, 2001.

[22] P. Debevec, «A Median Cut Algorithm for Light Probe Sampling,» de *SIGGRAPH*, 2005.

[23] P. Debevec, «Debevec, P. (2008, August). Rendering synthetic objects into real scenes: Bridging traditional and image-based graphics with global illumination and high dynamic range photography.,» *SIGGRAPH ,* p. 32, 2008.

[24] D. Mandl, «Learning Lightprobes for Mixed Reality Illumination,» *IEEE International Symposium on Mixed and Augmented Reality (ISMAR),* vol. 25, pp. 82-89, 2017.

[25] Vuforia, «[Available] https://library.vuforia.com/getting-started/overview.html».

[26] M. d. Unity, «[Available] https://docs.unity3d.com/es/current/Manual/».

[27] L. He, X. Ren, Q. Gao, X. Zhao, B. Yao y Y. Chao, «The connected-component labeling problem: A review of state-of.the-art algorithms,» *Pattern Recognition,* vol. 70, pp. 25-43, 2017.

[28] J. R. Michael F.Cohen, Radiosity and Realistic Image Synthesis, United Kingdom, 1993.

[29] B. T. Phong, «Illumination for Computer Generated Pictures,» *ACM SIGGRAPH'75 Conference Proceeding,* vol. 13, nº 6, pp. 311-317, 1975.

[30] S. P. Ashikhmin M., «An anisotropic Phong BRDF model,» *Journal on Graphics Tools,* vol. 2, nº 5, pp. 25-32, 2000.

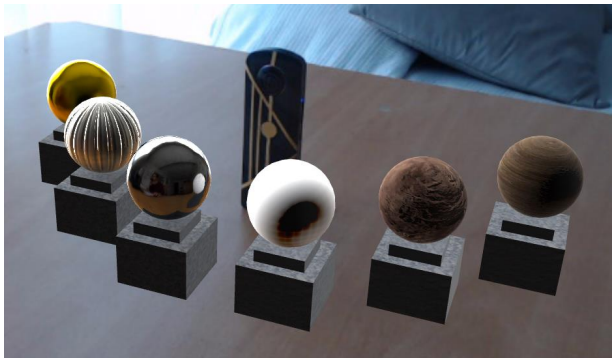
9. Anexos

Anexo I. Características de la cámara 360 Ricoh Theta S

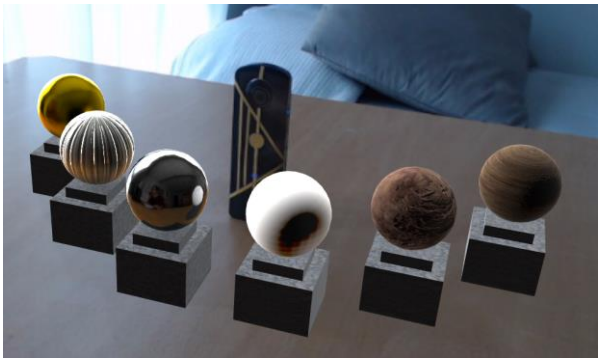
Distancia del objeto	**Aproximadamente 10 cm - ∞ (desde el frontal del objetivo)**
Modo captura	Imagen fija: Automático (Reducción de ruido/ Compensación de rango dinámico/ Renderizado de alto rango dinámico) *9, prioridad a la obturación, prioridad ISO *5, manual *5 Vídeo: Automático Emisión en vivo: Automático
Modo control de exposición	Programa AE, prioridad de la velocidad de obturación AE, sensibilidad de la ISO AE, exposición manual
Compensación de exposición	Imagen fija: Compensación manual (-2.0 - +2.0 EV, en pasos de 1/3 EV) *5
Sensibilidad ISO (sensibilidad de salida estándar)	Imagen fija: ISO100 - 1600 Vídeo: ISO100 - 1600 Emisión en vivo: ISO100 - 1600
Modo balance de blancos	Imagen fija: Automático, exterior, sombra, nublado, luz incandescente 1, luz incandescente 2, luz fluorescente de color luz del día, luz fluorescente del blanco natural, luz fluorescente blanca, luz fluorescente de color bombilla *5, Capturas con especificación de temperatura de color *9*10 Vídeo: Automático Emisión en vivo: Automático
Velocidad de obturación	Imagen fija: (sin incluir modo manual) de 1/6400 segundos a 1/8 segundos, (modo manual) de 1/6400 segundos a 60 segundos Vídeo: (L) de 1/8000 segundos a 1/30 segundos, (M) de 1/8000 segundos a 1/15 segundos Emisión en vivo: (USB) de 1/8000 segundos a 1/15 segundos, (HDMI) de 1/8000 segundos a 1/30 segundos
Otras funciones importantes de captura (imágenes fijas)	Captura a intervalos, Compuestas a intervalos *9, Temporizador automático *9, Capturas de horquillado múltiple *9
Número de fotos que se pueden grabar/tiempo *1	Imagen fija: (L) aprox. 1600 imágenes, (M) 9000 imágenes Vídeo (tiempo de grabación): Máx. 25 minutos o hasta el tamaño máximo de archivo de 4 GB *6 Vídeo (tiempo total de grabación): (L) aprox. 65 minutos, (M) aprox. 175 minutos *6
Tamaño de sensor de imagen	1/2.3 CMOS (x2)
Resolución de imagen fija	L: 5376 x 2688, M: 2048 x 1024
Resolución de vídeo/frecuencia de	L: 1920 x 1080/30fps/16 Mbps M: 1280 x 720/15fps/6 Mbps

fotogramas/velocidad de bits	
Resolución de emisión en vivo/frecuencia de fotogramas (USB)	L: 1920 × 1080/30fps M: 1280 × 720/15fps *La resolución y tasa de imágenes por segundo en Windows 7 es de 1280 × 720 y 15fps respectivamente.
Resolución de emisión en vivo/frecuencia de fotogramas (HDMI)	L: 1920 x 1080/30fps M: 1280 x 720/30fps S: 720 x 480/30fps *Cambia automáticamente para ajustar la visualización

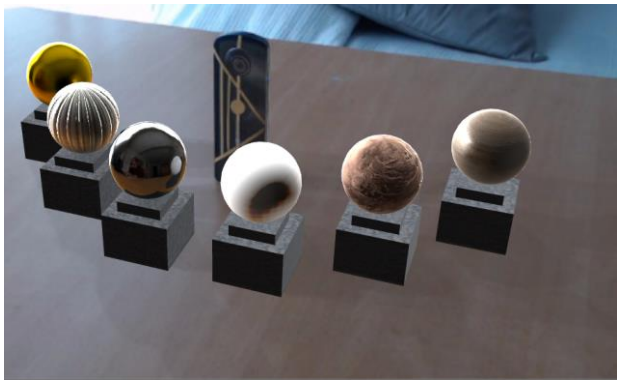
Anexo II. Ejemplos de las capturas realizadas para el escenario G con diferentes valores del factor de escala de la intensidad



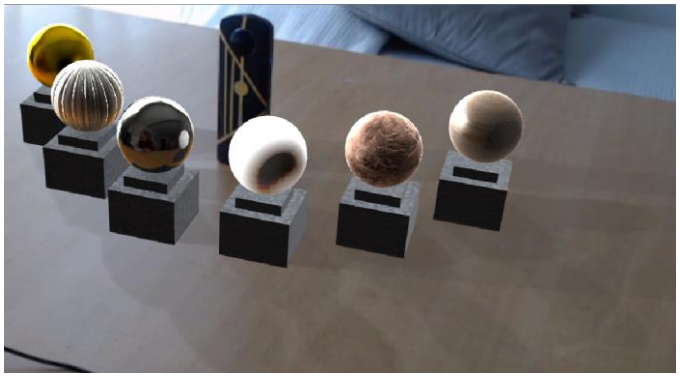
(a)Factor de escala a 0.02



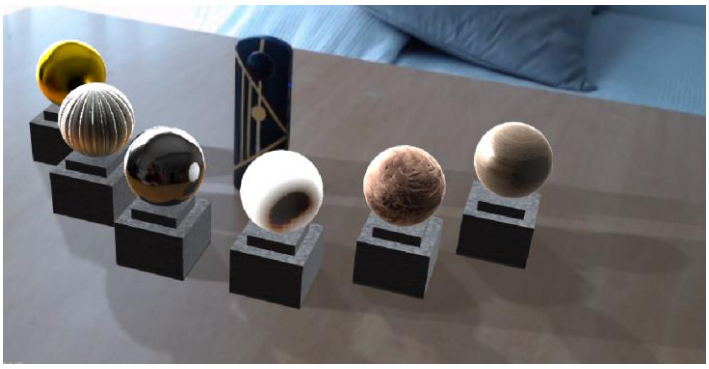
(b)Factor de escala a 0.04



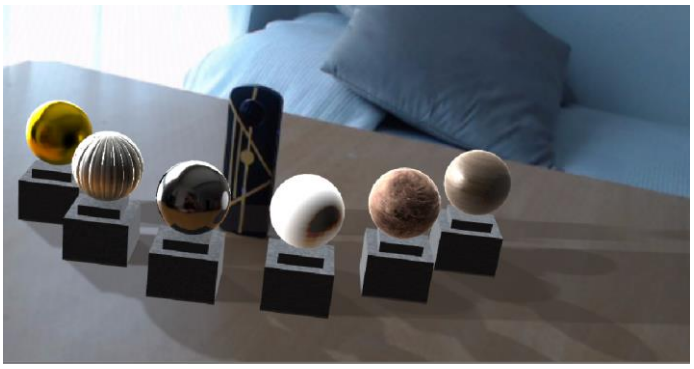
(c)Factor de escala a 0.08



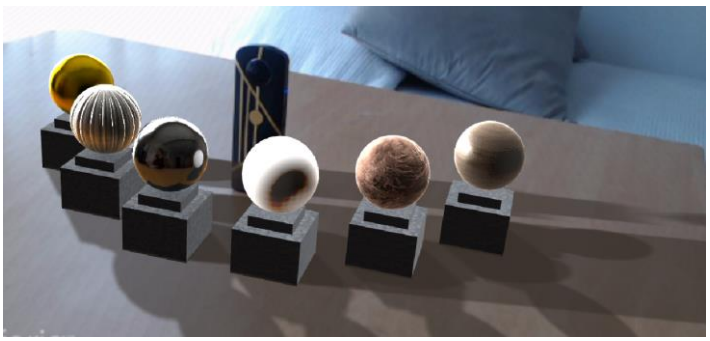
(d)Factor de escala a 0.1



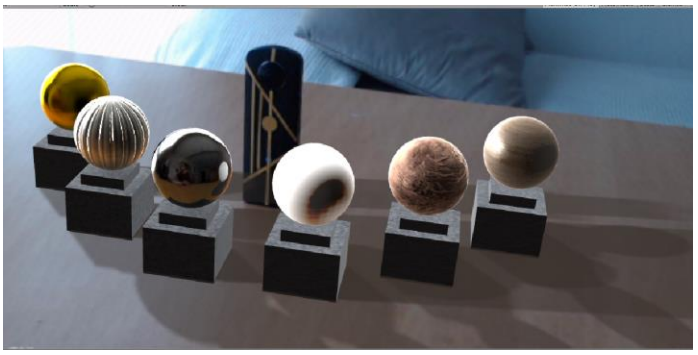
(e)Factor de escala a 0.2



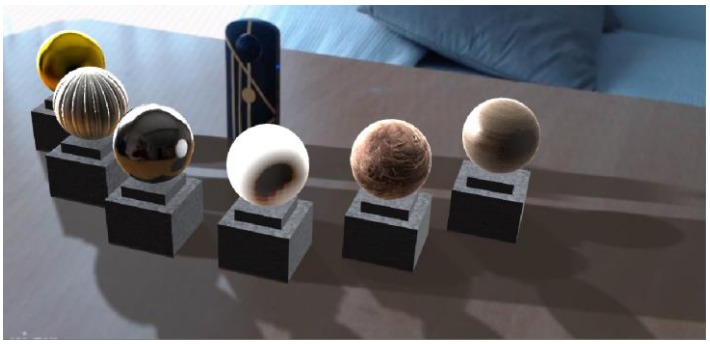
(f)Factor de escala a 0.3



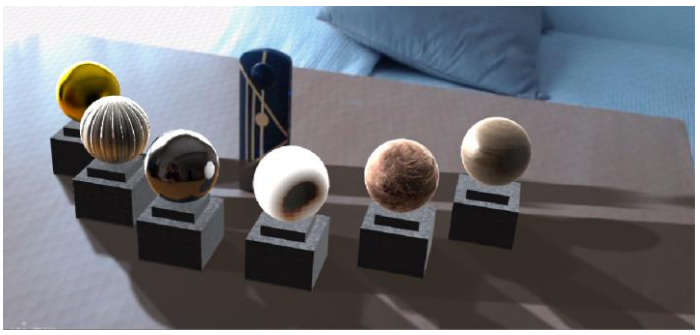
(g)Factor de escala a 0.4



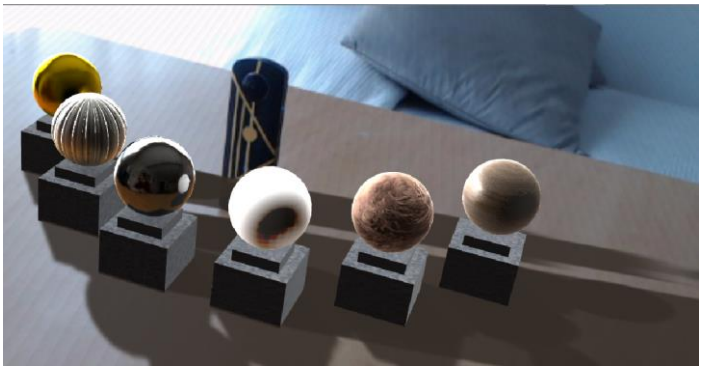
(h)Factor de escala a 0.5



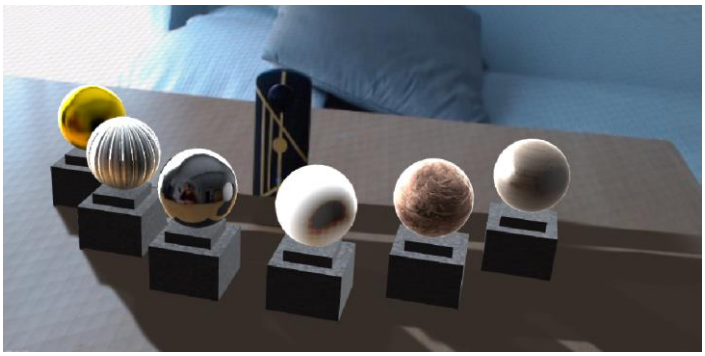
(i)Factor de escala a 0.6



(j)Factor de escala a 0.7



(k)Factor de escala a 0.8



(l)Factor de escala a 0.9

Anexo III. Resultados de las pruebas para determinar un valor límite del número de píxeles que debe de tener una región para que sea considerada una fuente de luz

Escena	Regiones	Cantidad de píxeles	¿Es un foco de luz?
A	1	3067	SI
	2	1954	SI
B	1	1129	SI
	2	1	NO
	3	109	NO
	4	438	SI
	5	2409	SI
	6	10	NO
	7	1	NO
	8	1	NO
	9	5	NO
C	1	1119	SI
	2	506	SI
	3	1487	SI
	4	2119	SI
	5	3	NO
	6	5	NO
	7	560	SI
D	1	2013	SI
E	1	2979	SI
F	1	1911	SI
	2	5	NO
	3	32	NO

	4	1	NO
	5	1	NO
	6	2	NO
	7	3	NO
	8	102	NO
	9	45	NO
G	1	1281	SI
	2	398	SI
	3	1314	SI
	4	2764	SI
H	1	77	NO
	2	1	NO
	3	395	NO
	4	707	SI
	5	606	NO
	6	1	NO