Escuela Superior
de Ingeniería y Tecnología
Universidad de La Laguna

Trabajo de Fin de Grado

Regreso a La Laguna del siglo XVI

Back to 16th century La Laguna

Pablo Torres Albertos

La Laguna, 10 de septiembre de 2022

D. **Isabel Sánchez Berriel**, con N.I.F. 42.885.838-S profesora Contratada Doctora adscrita al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como tutora

D. **Fernando Pérez Nava**, con N.I.F. 42.091.420-V profesor Titular de Universidad adscrito al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como cotutor

C E R T I F I C A (N)

Que la presente memoria titulada:

"Regreso a La Laguna del siglo XVI"

ha sido realizada bajo su dirección por D. **Pablo Torres Albertos**, con N.I.F. 51147935-Y.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 19 jul 2022

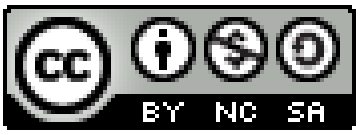
Agradecimientos

A mis tutores Isabel Sánchez Berriel y Fernando Pérez Nava por la dedicación y esfuerzo que han aportado a la hora de guiarme en la realización de este trabajo.

A los profesores del grado por compartir los conocimientos necesarios para que pudiese llegar hasta aquí.

A mi familia y amigos por su apoyo incondicional tanto en la realización de este trabajo de fin de grado como durante toda la carrera.

Licencia



RESUMEN

El objetivo de este trabajo ha sido el de desarrollar una aplicación donde el usuario sea capaz de poder comprobar cómo ha cambiado la ciudad de La Laguna desde el siglo XVI hasta la actualidad. El resultado de este desarrollo será una aplicación didáctica que mostrará al usuario la evolución a la que se ha sometido San Cristóbal de La Laguna durante estos últimos 5 siglos.

Se pretende adaptar un modelado 3D de la ciudad en el siglo XVI elaborado por el grupo de trabajo de Computación Gráfica de la Universidad de La Laguna a una aplicación para teléfonos móviles Android. Para ello se utilizará la herramienta de desarrollo de videojuegos Unity, donde se implementarán todas las funcionalidades necesarias para el desarrollo de una aplicación con fines educativos. Esto se conseguirá tomando las coordenadas GPS del dispositivo y ubicando al usuario en las mismas coordenadas en el modelo de la ciudad. A partir de lo anterior y de la capacidad de poder mirar a su alrededor mediante el movimiento del dispositivo, el jugador podrá apreciar in situ, tanto los edificios como los caminos, árboles, vegetación e incluso personas que habitaban La Laguna hace 5 siglos.

Por último, también será posible obtener información de edificios emblemáticos, personas y demás puntos de interés al mirarlos dentro de la aplicación.

Palabras clave: San Cristóbal de La Laguna, siglo XVI, GPS, Unity, modelado 3D, edificios emblemáticos, puntos de interés.

Abstract

The goal of this project will be to develop an application with the purpose of checking the past condition of the city of La Laguna and how it has changed since the 16th century. The result of this development will be a didactic application that will show the evolution that San Cristóbal de La Laguna has undergone during the last five centuries.

The intention will be to adapt a 3D model of the city in the 16th century, developed by the Graphic Computation work group of the University of La Laguna to a mobile application for Android devices. Through the Unity video game development tool, all the necessary functionalities will be applied for the development of an application to be used and implemented for educational purposes. For this purpose, the GPS coordinates will be used to locate the user in the same coordinates of the 3D city model. Taking into account the previous feature, joined to the ability to look around by moving the device, the user will be able to appreciate in situ both buildings and roads, trees, vegetation and even people who inhabited La Laguna five centuries ago.

Finally, it will also be possible to obtain information about emblematic buildings, people and other points of interest by looking inside the application.

Índice general

Índice de Figuras

Índice de Ecuaciones

Capítulo 1

1.1 Justificación

Las zonas que constituían los alrededores de San Cristóbal de La Laguna estuvieron habitadas desde la época de los guanches, sobre todo la zona del valle de Aguere, ya que en ese lugar se encontraba una gran laguna, de ahí el nombre que tomaría la ciudad posteriormente.

La propia ciudad fue fundada oficialmente al terminar la conquista del archipiélago Canario en 1496. Fue elegida capital por el Cabildo de Tenerife debido entre otras cosas a su lejanía de la costa, lo que evitaba los ataques de piratas. Esto permitió que su plan urbanístico careciera de murallas, ya que no era necesario defenderse de ningún ataque que procediera del mar. Esto llevó a La Laguna a ser la primera ciudad colonial en no tener fortificaciones. En consecuencia, el 2 de diciembre de 1999 fue declarada Patrimonio de la Humanidad por la UNESCO. [1]

Se trata por lo tanto de una ciudad con un valor turístico, cultural e histórico incalculable. Sus edificios emblemáticos cuentan la historia de una ciudad que ha preservado muchos de sus elementos a lo largo de su historia. Cualquier persona puede ir andando de un edificio histórico a otro siguiendo las mismas rutas que seguían las personas en la época en la que fue construida, ya que no ha sufrido apenas transformaciones en el trazado de sus calles y caminos.

Por lo tanto, podría aprovecharse el hecho de que nos movemos por los mismos caminos que usaban nuestros antepasados y viendo los mismos edificios que ellos veían para implementar una solución tecnológica que fuese capaz de mostrarnos cómo ha cambiado la ciudad, cuando las fachadas de las edificaciones se levantaban con las técnicas de la época. Esta aplicación también podría usarse para enseñar un poco de la historia de La Laguna y así promover el Patrimonio histórico de la ciudad.

1.2 Antecedentes y estado actual

La idea de una aplicación o programa que mezcla la propia realidad con la generada por la tecnología no es algo nuevo ya que existen algunos ejemplos de instituciones públicas y privadas que adaptan estos medios para facilitar tareas educativas:

- El proyecto **Penguin NAVI** [2] del Sunshine Aquarium se trata de una aplicación de realidad aumentada en la que unos pingüinos modelados en 3D te guían por la ciudad hasta el propio acuario. La parte de aprendizaje viene dada en la forma en la que caminan estos pingüinos virtuales, la cual imita la manera de caminar de esta especie en la vida real, lo que permite a los usuarios aprender un poco más sobre estos animales aparte de constituir una muy buena forma de guiar a posibles clientes.

Figura 1.2.1: Demostración de la aplicación Penguin NAVI del Sunshine Aquarium [2].

- El programa **Nintendo 3DS guide Louvre** para la consola 3DS ofrece a los usuarios una audioguía de las obras de arte que pueden encontrarse en el museo Louvre, en Francia. Además también ofrece diversos modelados en 3D y fotografías de las obras en exposición, con las cuales se puede interactuar usando los controles de la consola. El enfoque de esta aplicación es el de guiar a los visitantes del museo por sus instalaciones y también el de simular visitas virtuales. Si el usuario no está en el museo, puede usar un sistema con los planos de las plantas para situarse donde quiera

y ver las obras que se encuentren en ese lugar. Por otro lado, si se trata de un visitante, este será guiado usando el mismo sistema de planos.

- La empresa VewAR ofrece un servicio de instalación de guías virtuales para museos y otras instalaciones llamado **GuideBOT** [3] el cual se trata de una aplicación de guiado configurable a las inmediaciones del cliente. Lo más interesante de GuideBOT es la capacidad de mostrar información de un objeto de la exposición con solo apuntar con la cámara del dispositivo que se esté utilizando. Esto permite a la institución que instale GuideBot mostrar a los visitantes de las instalaciones la información.

1.3 Objetivos

Se han establecido los siguientes objetivos para el desarrollo de este TFG:

- Importar un modelo 3D de La Laguna en el siglo XVII facilitado por el grupo de trabajo de Computación Gráfica de la Universidad de La Laguna y realizado en formato Blender para su uso en Unity. Es necesario adaptar este modelo de forma que facilite el desarrollo de la aplicación.
- Posicionar al jugador en el modelo de la ciudad. Para esta tarea es necesario desarrollar un sistema que transforme las coordenadas GPS del dispositivo a coordenadas internas en Unity.
- Realizar las transformaciones y rotaciones necesarias para que la dirección de la cámara en la aplicación sea la misma a la que apunta el teléfono.
- Mejorar el rendimiento de la aplicación utilizando técnicas de renderizado.
- Implementar un sistema que permita al usuario obtener información sobre el punto de interés al que está mirando.
- Incorporar personajes cuya apariencia refleje las diferentes clases sociales y profesiones de la época.
- Implementar un sistema de objetivos que guíe al usuario al siguiente punto de interés.
- Exportar y probar la aplicación en diferentes dispositivos.

Capítulo 2

Herramientas y tecnologías usadas

2.1 Blender

Blender [4] es un programa informático multiplataforma desarrollado por la fundación Blender y que está enfocado al desarrollo de todos los procesos involucrados en el modelado 3D (modelado, rigging, animación, simulación, renderizado de luces…) El programa está optimizado para funcionar tanto en Linux como en Windows y Macintosh. Además, es software libre.

Se ha usado para modificar y adaptar el modelo de la ciudad para que así cumpla diferentes requisitos necesarios para facilitar el desarrollo de la aplicación.

Se ha utilizado este programa ya que los modelos provistos utilizaban el formato .blend, que es el formato utilizado por defecto en Blender.

2.2 Unity

Unity [5] es una plataforma de desarrollo de aplicaciones y videojuegos gratuita para uso personal o comercial hasta una cifra de beneficios. Se trata de una de las aplicaciones de su tipo más usadas, ya que es gratuita y cuenta con una comunidad muy grande, lo que facilita el aprendizaje de la propia herramienta debido a que existen muchos tutoriales, tanto de la compañía desarrolladora como de personas externas.

Unity permite el desarrollo de juegos y aplicaciones en 2D, 3D y realidad virtual o aumentada. Para esto, pone a disposición del usuario diversas herramientas, como un sistema de simulación de físicas, diferentes técnicas de iluminación y renderizado, gestor de objetos en escenas, objetos básicos para el desarrollo de videojuegos… También existen herramientas creadas por los usuarios para facilitar tareas más específicas. Además, Unity permite exportar

de forma fácil los proyectos a diferentes plataformas, como son consolas de videojuegos, ordenadores con diferentes sistemas operativos y móviles.

El lenguaje de programación usado por este motor de videojuegos es C#. Este ha sido desarrollado por la empresa Microsoft y guarda especial similitud con los lenguajes de programación C++ y Java.

Se denomina como frame a la imagen generada por el motor cuando se renderizan los objetos en pantalla. Al renderizar imágenes una tras otra, dan lugar al movimiento de la aplicación, y por lo tanto, a la ejecución de esta. La frecuencia en la que se renderizan estos frames en un segundo es comúnmente llamada frames por segundo y determina la velocidad en la que se ejecuta la aplicación. La velocidad de generación de estos frames depende de cuánto se tarde en terminar de procesar las instrucciones necesarias para esto.

Las instrucciones necesarias para generar un frame se separan en las necesarias para renderizar los objetos y la lógica que estos siguen. En Unity y la mayoría de motores, primero se ejecutan las instrucciones relacionadas con el comportamiento de los objetos y luego las relacionadas con la renderización en pantalla. La lógica del programa viene dada por el bucle del motor que distingue dos tipos de métodos, cuya diferencia reside en el número de frames que tardan en volver a ejecutarse. Existen métodos internos que se ejecutan cada frame, otros que se producen solo al principio de la ejecución y otros que solo se llevan a cabo si se produce cierta acción.

2.2.1 Paquetes adicionales utilizados

A parte de los paquetes básicos de Unity, también se han usado los siguientes paquetes adicionales:

- **NuGetForUnity:** un administrador de librerías .NET similar a npm para NodeJs o rpm para ruby. Permite instalar dichas librerías en Unity usando el **nuget package** correspondiente [6]. Utilizaremos esta herramienta para instalar la librería CoordinateSharp, la cual permite hacer conversiones entre coordenadas.

- **Speech And Text in Unity iOS and Unity Android:** permite usar las herramientas de texto a voz y voz a texto ofrecidas en Android Studio en Unity [7].
- **TextMeshPro:** facilita más opciones para la configuración de textos en Unity, además de aportar mejoras visuales para estos.
- **Unity Remote:** permite depurar la aplicación sin necesidad de instalarla o actualizarla en el dispositivo móvil, usando una conexión por USB [8].

2.3 Teléfonos móviles con Android

Android es uno de los sistemas operativos más utilizados por los smartphones actuales. En el año 2021, el 67.21 por ciento de teléfonos móviles en Europa utilizaban Android [9]. Unity es capaz de exportar cualquier tipo de proyecto a este sistema, pero tiene que estar adaptado a éste de las siguientes maneras:

- **Interfaz adaptada a funciones táctiles:** la mayoría de teléfonos móviles usan tecnologías táctiles para que el usuario interactúe con el dispositivo. Por lo tanto, es necesario que los proyectos de Unity enfocados a estos sistemas cuenten con interfaces que puedan ser usadas de esta forma.
- **Rendimiento de la aplicación:** se necesita adaptar los requerimientos computacionales de la aplicación a los teléfonos móviles, los cuales tienen, en su mayoría, menor potencia que las consolas de videojuegos u ordenadores.
- **Utilización de sensores específicos:** es posible utilizar los sensores y actuadores incorporados en los móviles, los cuales no suelen estar presentes en otros dispositivos. Por lo tanto, se pueden diseñar experiencias que solo son posibles en teléfonos móviles.

Capítulo 3

Desarrollo

3.1 Formación inicial

No fue necesario un periodo de formación extenso ya que estaba bastante familiarizado con la herramienta Unity, tanto como por haberla usado en proyectos personales como por haberla usado a lo largo de la asignatura de Interfaces Inteligentes. Más bien, la mayoría del tiempo de investigación para esta herramienta estuvo enfocado en cómo trasladar los sistemas necesarios para el desarrollo de la aplicación a Unity. Por otro lado, también fue necesario el estudio de conceptos más complejos y avanzados del programa, los cuales se explicarán en detalle más adelante.

El uso de Blender solo fue necesario para tareas muy específicas, por lo que la preparación en está cuestión derivó en simples búsquedas para encontrar cómo hacer operaciones concretas.

3.2 De Blender a Unity

Los profesores encargados de la tutorización de este trabajo se han encargado de suministrar el propio modelado 3D de la ciudad en el siglo XVI. Este modelo ha sido elaborado por el grupo de trabajo de Computación Gráfica de la Universidad de La Laguna para el proyecto Reconstrucción Histórica Virtual de San Cristóbal de La Laguna. El modelo se encuentra en el formato blend, por lo que será necesario exportar a formato fbx para así poder usarlo en Unity. En este proceso de exportación hay que poner especial atención en los materiales, con el objetivo de que se apliquen de forma adecuada a los diferentes objetos. Como resultado, la mayoría de las texturas se importan de manera satisfactoria al motor, pero algunas habrá que asociarlas a la superficie correspondiente de manera manual.

3.3 Posicionamiento por GPS

El posicionamiento por GPS del usuario se trata de una de las funcionalidades que más afectará al uso de la aplicación, por lo que es necesario diseñarlo y depurarlo de tal forma que ofrezca una interacción rápida y libre de errores.

Para poder localizar un punto dentro de la superficie terrestre es necesario que éste tenga unas coordenadas únicas basadas en un sistema de referencia. Un problema que surge a raíz del concepto anterior es que no existe forma de representar la tierra de forma fidedigna sin realizar deformaciones o proyectando su superficie con diferentes fórmulas [10]. Usando la fórmula de la proyección transversa de Mercator se obtienen coordenadas UTM, expresadas en metros. El modelado de la ciudad se ha elaborado teniendo en cuenta este sistema. Al exportar de Blender a Unity, se transformarán estas coordenadas a las que usa el propio programa, de tal manera que el centro de la ciudad, que está en las coordenadas UTM E: 371197.41 N: 3151901.76, estará en la posición x: 0, y: 0, z:0 en Unity. Debido a esto, se puede asumir que para transformar coordenadas UTM a unidades de Unity se usan las siguientes fórmulas:

$$x = EsteUTM - 371197.41 \quad (1)$$

$$y = NorteUTM - 3151901.76 \quad (2)$$

Sin embargo, las coordenadas que pueden ser leídas a partir del GPS del dispositivo móvil utilizando la clase Input son geográficas. Las coordenadas geográficas o GPS permiten situar cualquier punto de la superficie terrestre usando dos unidades angulares expresadas en grados sexagesimales:

- **Latitud:** ángulo entre el plano ecuatorial y la línea que pasa por el punto elegido y el centro de la Tierra. Se podría considerar como la coordenada y en un sistema de referencia 2D.
- **Longitud:** ángulo entre el meridiano de referencia de Greenwich y el meridiano que pasa por ese punto. Los meridianos son líneas verticales imaginarias que pasan por los

dos polos. Se podría considerar a la longitud como la coordenada x en un sistema de referencia 2D.

- **Altitud:** específica la elevación de dicho punto y requiere de otro modelo diferente que produzca diferentes valores para la altitud.

Se tiene que tener en cuenta que este modelo toma a la Tierra como una esfera perfecta.

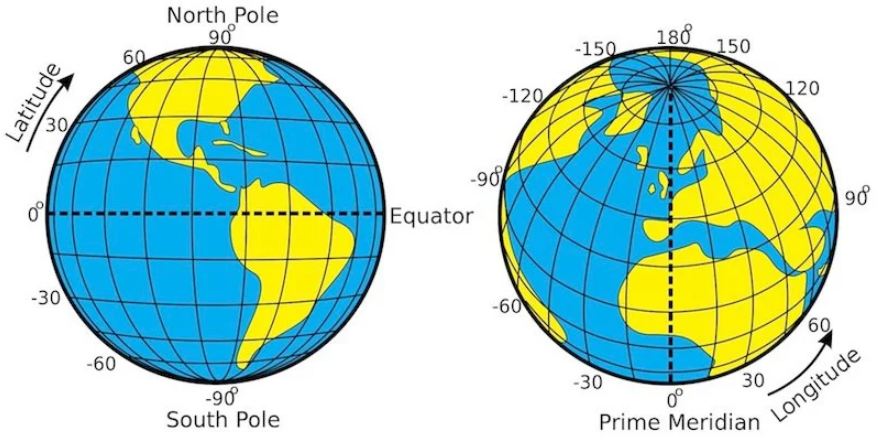


Figura 3.3.1: Representación visual de las coordenadas geográficas [11].

Por lo tanto es necesario transformar de coordenadas geográficas a UTM, para lo que se utilizará la librería .NET **CoordinateSharp**, que permite utilizar diferentes métodos que se ocuparán de las transformaciones. No obstante, no se pueden transformar las coordenadas en cada frame, ya que las llamadas a las funciones de CoordinateSharp son costosas para el rendimiento de la aplicación. Para solucionar esta cuestión, se harán estas tareas sólo cuando las coordenadas geográficas del frame actual sean diferentes a las del frame anterior. Si se da el caso de que sean diferentes, aparte de realizarse las transformaciones necesarias, se almacenarán las coordenadas obtenidas por el GPS en ese frame para usarlas en el siguiente como las coordenadas del frame anterior. Por lo tanto, será necesario almacenar en una nueva variable las coordenadas obtenidas en el anterior frame. Esta variable se inicializará con el valor (0, 0), el cual sería imposible para coordenadas obtenidas en La Laguna, ya que se necesita transformar las coordenadas en el primer frame.

Es posible facilitar estas comparaciones creando una clase que almacene dicha información y sobrecargando los símbolos de comparación.

3.3.1 Errores del GPS

Uno de los principales problemas de posicionamiento por GPS es que no es del todo preciso. Al hacer algunas pruebas con la aplicación se ha podido observar que el error medio de las coordenadas obtenidas suele ser de 1 o 2 metros. Además, no se obtienen nuevas coordenadas en tiempo real, sino cada 1 o más segundos [12], aunque también dependen del dispositivo y las velocidades de procesamiento de Unity. En conclusión, con la tecnología actual es imposible que un dispositivo disponible para el consumidor sea capaz de obtener coordenadas con una precisión fiel a la realidad.

Muchas de las aplicaciones del mercado utilizan otros medios en conjunto con el GPS para obtener estos datos. Por ejemplo, los servicios de localización de Google utilizan redes Wi-Fi cercanas, redes móviles y otros sensores del mismo dispositivo [13]. Utilizar este tipo de técnicas será muy complicado y llevaría mucho tiempo, por lo que se han diseñado diferentes sistemas para sobrellevar estos problemas.

Para empezar, como la actualización de las coordenadas obtenidas solo ocurre cada cierto tiempo, el jugador cambiará de posición dentro del mapa de manera muy brusca, lo que se podría describir como un teletransporte. Esto tiene un gran impacto negativo en la experiencia del usuario, ya que es posible que se desoriente y confunda con los movimientos rápidos en la posición de la cámara. Para mejorar este aspecto, se usa el método moveTowards de la clase Vector3, que calcula cuánto tiene que moverse el objeto cada frame para llegar desde la posición inicial hasta la final en el tiempo establecido. La coordenada inicial corresponde a las obtenidas por el GPS en el frame anterior mientras que las finales son las obtenidas en el frame actual. El tiempo establecido tiene en cuenta que lo más plausible es que el jugador no atraviese grandes distancias en el tiempo que se tarda en obtener nuevas coordenadas.

Otro error más preocupante reside en la posibilidad de que el jugador sea posicionado dentro de un edificio por un error en la toma de coordenadas. La solución a este problema requiere en primer lugar establecer un sistema de colisiones. En Unity, se define qué objetos tienen la capacidad de colisionar e interactuar físicamente con otros mediante el uso del

componente collider. Un collider define la forma física que tiene mediante un objeto geométrico que lo envuelve y es utilizado para realizar los cálculos de intersección con otras mallas en la escena.

Los colliders permiten dos tipos de interacción entre objetos, la colisión y la activación:

- **Colisión:** determina si dos objetos han chocado. Para que esto ocurra, los dos tendrán que tener un collider y al menos uno de ellos un componente de tipo Rigid Body, un componente que permite al objeto usar simulaciones físicas.
- **Activación:** determina si el collider de alguno de los objetos está dentro del otro. Para este tipo de interacción sólo es necesario que los objetos tengan un collider y que tengan activado la opción trigger. La colisión está libre de simulación física.

Por otro lado, existen diferentes tipos de collider, pero los dos más usados en este proyecto son:

- **Box collider:** Collider en forma de caja.
- **Mesh collider:** Collider que intenta adaptarse automáticamente a la forma del objeto al que está asociado. Puede adaptarse exactamente a la forma de los polígonos del modelo 3D del objeto o de manera que el collider sea menos fiel a estos polígonos pero tenga forma convexa. Está última es útil cuando se tienen partes vacías, ya que si no es convexo, el collider no introducirá colisiones en estos huecos. Un ejemplo de esto es el propio modelo de La Laguna, cuyas casas mayoritariamente están huecas, de forma que los mesh colliders no convexos sólo se adaptaban a las paredes del edificio y no incluían el interior.

Estos componentes otorgan a las clases asociadas a los objetos la opción de usar métodos que se ejecutan en los frames en los que se produzca una interacción de colisión o de activación con otro objeto con collider, siguiendo las reglas especificadas anteriormente. De esta manera es posible programar la reacción de un objeto a la colisión con otro objeto, teniendo en cuenta factores como el punto de colisión, el tipo de objeto que era, la etiqueta que poseía este, etc.

En la fase de adaptación del modelado en Blender se modificó el modelado de tal manera que cada edificio fuese un objeto individual e independiente de los demás. De esta forma, se pueden seleccionar todas las edificaciones e insertarles el componente mesh collider a la vez. Si se selecciona la opción de collider convexo, el componente se adaptaría a las estructuras de forma correcta, ya que estos en su mayoría poseen formas bastante simples. En cambio, las construcciones singulares y públicas, como pueden ser el Convento de Santo Domingo o la Iglesia de la Concepción, tienen formas más complejas y las cuales no es posible adaptar usando solo mesh colliders.

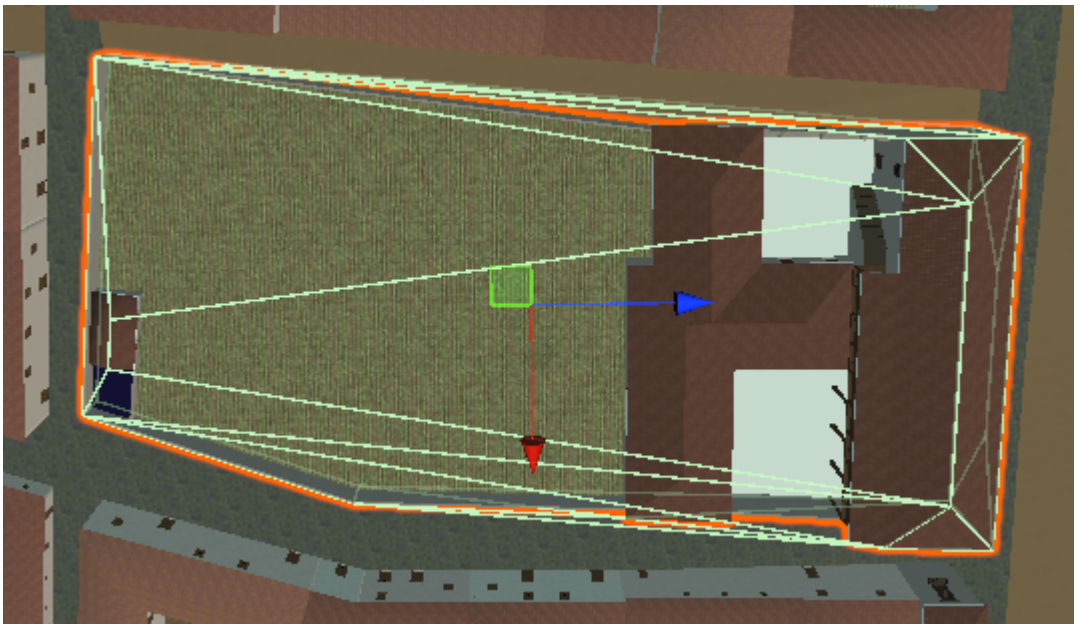


Figura 3.3.1.1: Intento de usar mesh colliders en la Iglesia del Convento de Santa Catalina de Siena.

En la anterior figura, la línea naranja marca los límites del objeto que representa esta edificación, mientras que las verdes muestran las caras del collider, que es una figura convexa. Como se puede apreciar, el collider no es correcto ya que supera los límites del edificio, ocupando parte de los caminos adyacentes.

Para este tipo de caso, es mejor utilizar varios objetos cuadrados los cuales tienen cada uno un box collider e intentar adaptar la forma de la estructura.

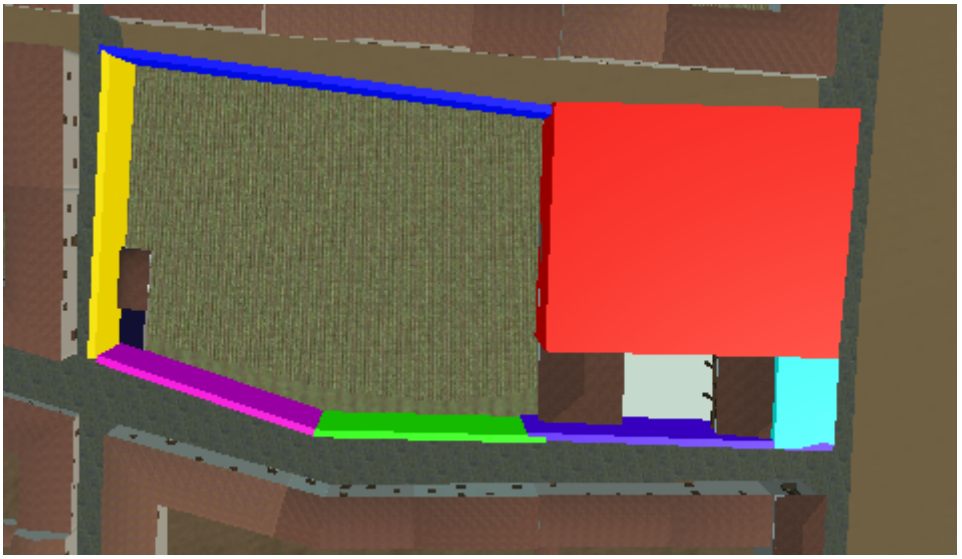


Figura 3.3.1.2: Representación visual del uso de box colliders para especificar de forma precisa las colisiones de objetos complejos.

Al haber realizado todos estos preparativos, se podrá evitar transportar al jugador dentro de un edificio enviando antes un objeto que compruebe si existe una colisión o una activación (en este caso, no existen diferencias entre los dos). Si se ha producido, se puede confirmar que existe una edificación, por lo que no se enviará al jugador a las coordenadas especificadas.

3.3.2 Posicionamiento vertical

Es posible obtener también los datos relacionados con la altitud a la que se encuentra el individuo para situarlo en el mapa, pero como se ha mencionado anteriormente, estos datos están sujetos a cierto grado de error, el cual debe ser tratado. No existe forma de obtener la altitud y la longitud del dispositivo de otra forma, pero si se puede obtener una aproximación bastante fiable de la altitud usando las coordenadas geográficas y un elemento muy común en el desarrollo de videojuegos denominado Raycast. Se designa como Ray casting a las técnicas que implican el lanzamiento de “rayos” capaces de chocar contra superficies y devolver el conjunto de objetos con el que se intersecta a lo largo de su trayectoria. De esta forma se puede obtener todo tipo de información relevante a esta colección de objetos. En Unity, estos

rayos solo chocarán con objetos que cuenten con un componente collider. A partir de esta colisión, se establece que código ejecutar mediante comprobaciones con el nombre del objeto, su etiqueta, algún componente que tenga, etc.

Para empezar, se ha establecido "Suelo" como etiqueta de todos los objetos en los que el jugador puede posicionarse encima. También se ha incorporado un componente box collider a todos estos, ya que un mesh collider no sería capaz de reflejar los cambios en la elevación existentes en los terrenos. Se lanzará un rayo que apuntando hacia abajo y desde el centro del objeto que representa al jugador, obteniendo el punto de colisión para utilizar en el cálculo de coordenada *y* en la que tiene que posicionarse usando la siguiente fórmula:

$$yPos = hitpoint.y + height \quad (3)$$

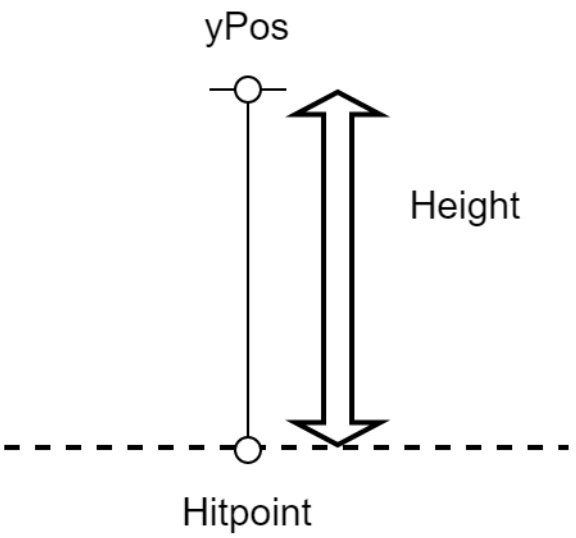


Figura 3.3.2.1: Representación gráfica de la fórmula de posición vertical

Donde:

- yPos: la coordenada de la altura en la que se posicionará el jugador.
- hitpoint.y: la altura en la que el rayo hizo colisión con el suelo.
- height: variable que representa la altura del jugador y que puede ser cambiada por este.

Será necesario calcular esta altura cada vez que se mueve al jugador, incluyendo también en cada frame en el que se mueva entre dos coordenadas usando el método

moveTowards mencionado anteriormente. De esta forma el cambio de altura se realizará de forma progresiva teniendo en cuenta la elevación del propio terreno.

3.4 Movimiento de la cámara

Una vez ubicados en el mapa, el usuario deberá poder explorar y mirar su entorno. Para lograr esto, se ha optado por hacer que la rotación de la cámara imite a la del propio teléfono, por lo que el jugador estará orientado en la misma dirección tanto en el juego como en la realidad.

En primer lugar, se ha usado el giroscopio para obtener los datos de rotación actuales del teléfono. La clase interna de Unity **Input** posee una propiedad llamada gyro, la cual almacena los datos de orientación del dispositivo en una variable de tipo Quaternion denominada **attitude**. Unity representa todas las rotaciones usando **cuaterniones** (Quaternion), siendo estos una forma de representar rotaciones en el plano físico. Los cuaterniones están formados por un eje real w y 3 ejes imaginarios x, y, z, teniendo en cuenta que todos los ejes son dependientes entre sí, por lo que no se puede cambiar el valor de un eje sin cambiar el de los demás [14]. Los cuaterniones pueden expresarse de la siguiente forma:

$$q = w + x * i + y * j + z * k \quad (4)$$

Es posible expresar las rotaciones de un ángulo Θ sobre un eje $\vec{u}$ usando cuaterniones de la siguiente manera:

$$q = cos(\Theta/2) + (u_X * i + u_y * j + u_z * k) * sin(\Theta/2) \quad (5)$$

Unity ofrece los métodos suficientes para simplificar el uso de cuaterniones, por ejemplo, permitiendo obtener cuaterniones a partir de vectores de 3 valores que representan ángulos de Euler, los cuales son más fáciles de entender.

Sin embargo, no es posible utilizar directamente los datos obtenidos a partir de la propiedad attitude, ya que estos están representados siguiendo **coordenadas a la izquierda** (Left-handed Coordinates) mientras que Unity utiliza **coordenadas a la derecha** (Right-handed Coordinates). Estos conceptos afectan principalmente a la dirección a la que apuntan sus ejes.

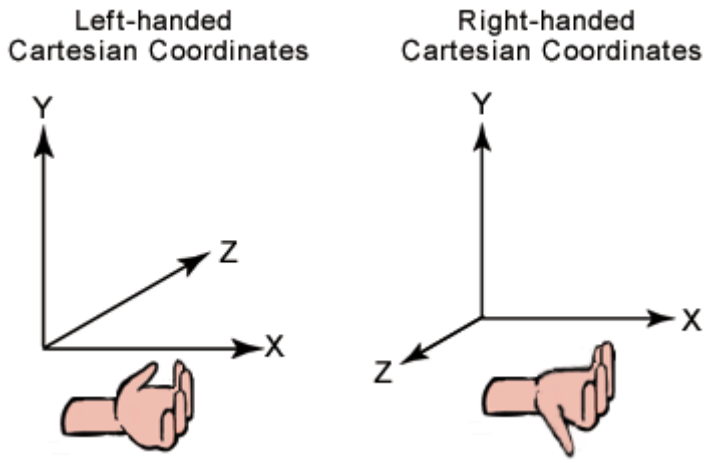


Figura 3.4.1: Representación de las reglas de la izquierda y la derecha utilizando coordenadas cartesianas para simplificar [15].

A partir de la experimentación en el propio motor y consultar la documentación, es posible entender que la propiedad attitude anterior es tomada a partir de un sistema de referencia basado en un cuaternión sin rotación (cuaternión identidad) que al transformarlo a ángulos de Euler tiene su eje z siguiendo la dirección de la pantalla del dispositivo hacia fuera. Es posible invertirlo para que el eje z siga la dirección a la que apunta la cámara si se multiplica por otro cuaternión con la ecuación:

$$q = 0 + 0 * i + 0 * j + 1 * k \qquad (6)$$

Para transformar de un sistema de coordenadas a otro, se usan los conceptos derivados de la jerarquía de objetos de Unity. Un objeto hijo seguirá siempre la rotación y posición del padre. Por lo tanto, si se establece como padre de la cámara un objeto intermedio que sirva

para alinear el sistema de coordenadas a la izquierda y el de la derecha, la cámara podría seguir sin problema la rotación obtenida a través del giroscopio, ya que el objeto padre se encargará de realizar la transformación entre sistemas y la cámara realizará sus rotaciones en función de la rotación del padre.

Para evitar cambios bruscos en la rotación de la cámara y así mejorar la experiencia del usuario, se suavizará el movimiento de la cámara usando el método Lerp de la clase Quaternion. Esta función pasará de una rotación inicial a una final en un tiempo especificado, lo que hará menos bruscos los cambios rápidos de dirección.

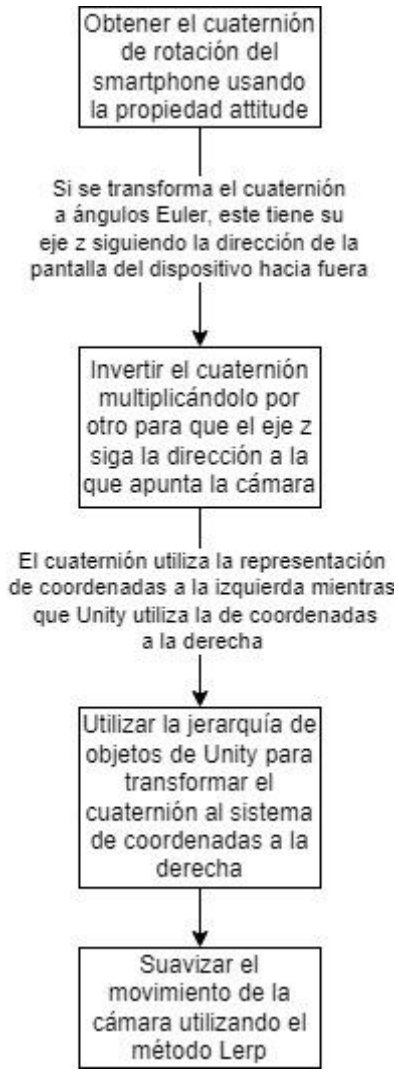


Figura 3.4.2: Diagrama resumen de los pasos para lograr el movimiento de cámara

3.5 Calibración de la cámara

El giroscopio no tiene en cuenta la dirección con la que empieza el jugador, por lo que si este empieza mirando a otra dirección que no sea el norte, las rotaciones realizadas no serán las mismas. Debido a esto es necesario una mecánica dentro de la aplicación que pida al usuario mirar al norte antes de empezar. Para este fin, se desarrolló una fase de calibrado dentro de la aplicación, donde se pide al jugador que alinee una aguja con el norte magnético, ayudándose de una rosa de los vientos. La manecilla cambia el valor del eje z de su rotación en ángulos Euler de la propiedad compass.trueHeading de la clase Input. Dicha propiedad medirá el ángulo que crea la dirección a la que apunta la parte superior del móvil y el norte magnético.

El resultado obtenido permite que la dirección de la aguja copie la del teléfono móvil. Una vez realizada la fase de calibración, se sabrá con certeza que el jugador está mirando al norte, por lo cual la posición inicial del giroscopio será la correcta, ya que este también empieza orientado hacia el norte. En consecuencia, todas las demás rotaciones realizadas estarán alineadas con las del dispositivo. También se le ofrecerá al usuario la posibilidad de volver a repetir la calibración si por alguna razón esta no ha sido correcta.

3.6 Optimización del rendimiento

Aunque hoy en día existen teléfonos móviles de gama muy alta, los cuales pueden rivalizar incluso con la potencia computacional de algunas consolas de videojuegos, la mayoría de los usuarios tienen dispositivos menos capaces. Si a esto se suma que el modelado 3D posee mucho detalle, se entenderá la necesidad de optimizar la aplicación para que sea capaz de generar un rendimiento adecuado en este tipo de dispositivos.

3.6.1 Occlusion culling

La eliminación selectiva de oclusión (occlusion culling) es una técnica usada en desarrollo de videojuegos para mejorar el rendimiento que consiste en no renderizar un objeto

si este no es visible por la cámara que se esté usando. Para poder detectar esto, el motor de videojuegos dividirá la escena en celdas y generará información que describe la geometría de las celdas y la visibilidad entre celdas adyacentes. Estos datos son cargados en memoria cuando se ejecute la aplicación, y serán consultados cada frame para determinar qué objetos se han de renderizar.

Figura 3.6.1.1: Ejemplo de occlussion culling: Al mirar a la parte derecha de una calle con edificios a ambos lados, solo se renderizan los edificios del lado que se esté mirando.

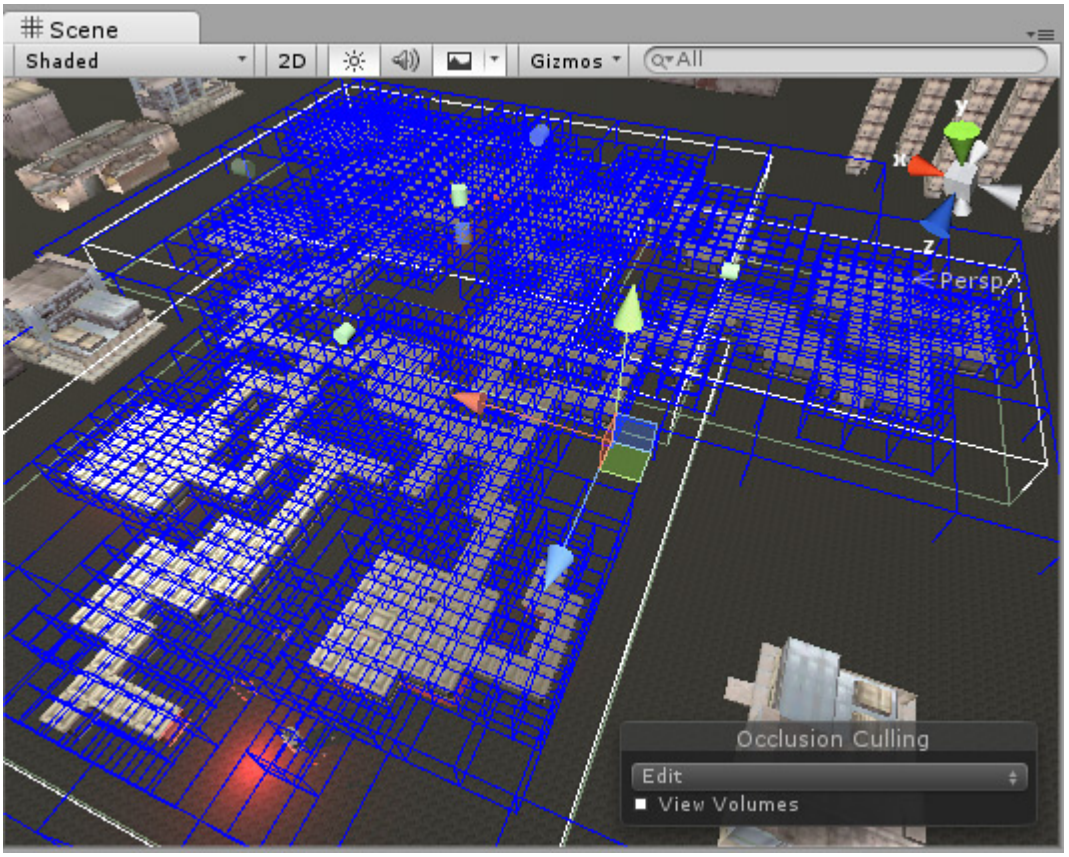


Figura 3.6.1.2: Representación de las celdas usadas en el occlusion culling [16].

Las dos opciones más importantes al configurar el occlusion culling son:

- **Smallest Occluder:** tamaño mínimo que tiene que tener un objeto para que pueda tapar a los demás, o lo que es lo mismo, que el objeto provoque que la cámara no pueda ver los objetos detrás de este.
- **Smallest Hole:** tamaño mínimo de los huecos por los cuales la cámara puede ver

Ambas opciones determinarán la calidad del resultado, así como el tiempo que tardará en realizarse y la cantidad de memoria que ocupará, ya que cuanto más pequeños sean los tamaños mínimos, más objetos se deben tener en cuenta, dependiendo de si el modelado contiene muchos huecos pequeños o muchos objetos pequeños. En el caso del modelo de La Laguna, no existen muchos objetos ni huecos de tamaño reducido por lo que el impacto no es significativo.

3.6.2 Distancia de renderizado

La distancia de renderizado dicta si un objeto se dibujará dependiendo de lo lejos que esté de la cámara usada. Los smartphones donde se ejecutará la aplicación tienen una pantalla de menor tamaño que un monitor habitual, por lo que es posible reducir la distancia de renderizado hasta cierto punto sin que esto tenga un gran impacto en la calidad gráfica.

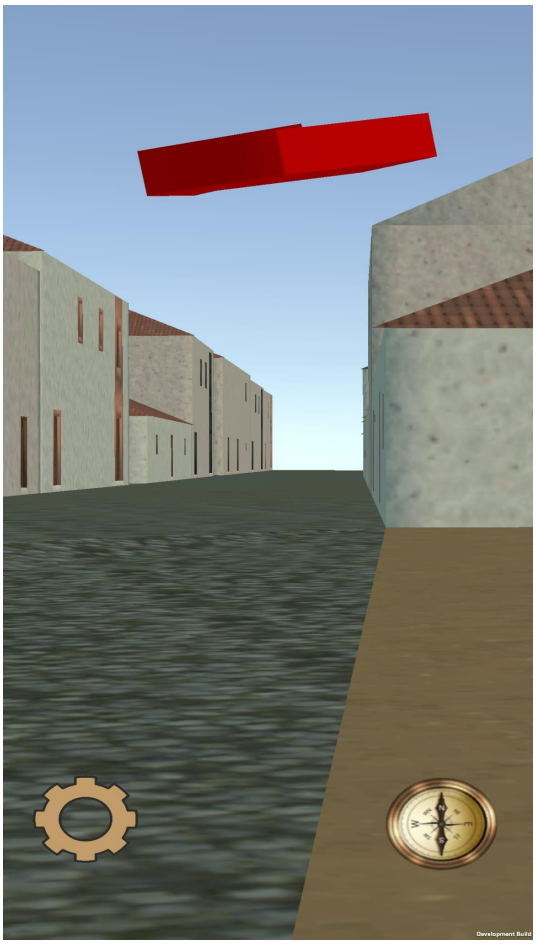


Figura 3.6.2.1: Ejemplo del efecto de reducir la distancia de renderizado.

3.6.3 Reducción del tamaño de las texturas

Las texturas son imágenes que se superponen encima de los modelos 3D para añadirles detalles, como pueden ser color, superficies, iluminación. Como cabe esperar, cuanto mayor sea la resolución de estas imágenes, más minuciosos serán estos detalles y más recursos se consumirá al renderizarlos. Por este motivo será necesario reducir la resolución de las texturas utilizando herramientas que provee Unity, ya que el rendimiento no llega a ser el

adecuado con el tamaño original de estas. Se ha reducido el tamaño máximo de las texturas a 128, buscando así el rendimiento de la aplicación frente a la calidad gráfica.

3.7 Sistema de guía

El objetivo de esta aplicación es el de ilustrar a los usuarios sobre la historia de la ciudad a través de sus lugares emblemáticos. Con este objetivo, se ha desarrollado un sistema de guía que permitirá a los jugadores obtener información de un punto de interés solo con mirarlo.

El primer problema que surge está relacionado con la manera en la que se almacena la información de los puntos de interés. Para que estos datos sean fáciles de crear y editar directamente en Unity se han usado Scriptable Objects. Los Scriptable Objects son un tipo de objeto en Unity que almacena información en forma de variables especificadas por el desarrollador. Este tendrá que definir en un script una clase que herede de ScriptableObject, donde se definen estos valores a almacenar, teniendo como resultado un nuevo tipo de objeto que es creado desde el navegador de archivos de Unity. Una vez creado el objeto, será posible editar los valores de dichas variables desde el propio inspector, agilizando así la manera de generar datos que comparten características. Para este proyecto, las variables serán el nombre del punto y su descripción.

En este caso, se usará además una clase nueva llamada Touristic Display cuyas instancias estarán asociadas a un punto de interés y se encargará de almacenar una referencia al Scriptable Object que representa a ese lugar.

Para poder visualizar esta información, se usará otra clase que tiene como fin disparar un Raycast en la dirección a la que esté mirando el jugador y comprobar si el objeto con el que colisionó tiene la etiqueta “Casa” y él mismo o su padre tienen asociado una instancia de Touristic Display. En caso de que se cumplan estas condiciones, se cargarán los datos almacenados en el ScriptableObject referenciado por el script y se mostrarán en un panel que estará visible sólo si el jugador aprieta un botón que aparecerá en pantalla. Como detalle a

tener en cuenta, solo se cargará la información en el panel si este no es visible, para evitar que nueva información se muestre sin haber cerrado la anterior.

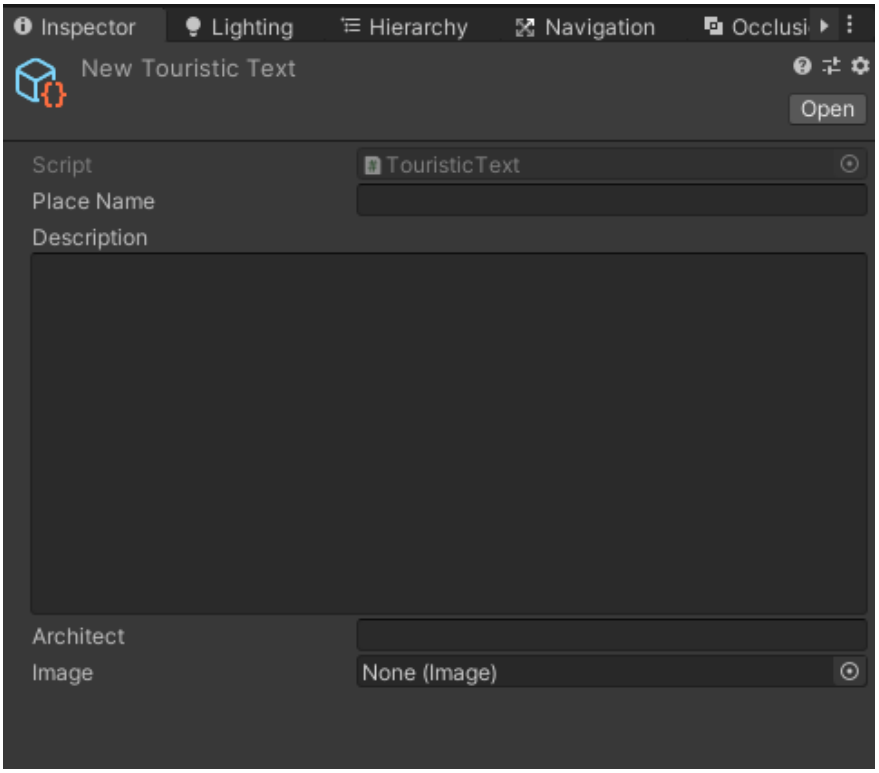


Figura 3.7.1: Menú que permite crear y editar un Scriptable Object con variables personalizadas. Architect e image no se usan en la versión actual pero se dejan preparadas para su futuro uso si fuese necesario.

3.7.1 Narración por audio de la información

Usando la herramienta de texto a narración "Speech and text Unity Ios Android" se ha programado la capacidad para poder escuchar todos los datos de los puntos de interés sin necesidad de grabarlos previamente.

3.7.2 Selección y obtención de los datos de los puntos de interés

Para seleccionar qué lugares deberían incluirse en el sistema, así como sus datos descriptivos, se ha consultado la página web Patrimonio Cultural Cicop [17] la cual tiene una sección con los bienes culturales de la ciudad, con sus descripciones, historia, etc.

Al haber asignado anteriormente los colliders a cada edificio, lo único que será necesario para que el sistema funcione correctamente es asignar el script TouristicDisplay y la referencia al ScriptableObject que corresponda con el punto de interés. Añadir manualmente un número significativo de emplazamientos como se quiere hacer sería muy lento y tedioso. Para facilitar esta tarea se ha implementado una funcionalidad que mueve la vista del visor de escenas de Unity hasta las coordenadas geográficas especificadas, haciendo uso de la misma ecuación de conversión a coordenadas UTM que se usó anteriormente. De esta manera se puede buscar el inmueble en Google Maps, copiar las coordenadas geográficas y pegarlas en un formulario de Unity, llevando la vista directamente al edificio al que se tiene que asociar el script.

Figura 3.7.2.1: Uso de la herramienta de asignación de TouristicDisplay.

Algunos puntos de interés importantes no existían en el siglo XVI. Para poder incluir estos lugares, se introduce un ícono el cual representa el sitio. Este icono contiene un collider, la etiqueta y el script correspondiente, de tal manera que sea interactuable de la misma forma que los otros puntos.

3.7.3 Guía visual hacia los puntos de interés

La aplicación no cumplirá su cometido de informar al usuario si no puede encontrar los puntos de interés. Será necesario guiar al jugador para que este sea capaz de encontrarlos. Con este cometido se ha utilizado un modelo 3D de una flecha [18] que usando la clase ArrowGuide, cambiará su rotación de forma que apunte hacia el punto de interés más cercano. También se tendrá en cuenta si se ha visitado o no el punto de interés para así poder evitar guiar al usuario a un punto del cual ya ha leído la información.

Al principio de la ejecución, la clase ArrowGuide se encargará de encontrar y almacenar en una lista todos los objetos con la etiqueta “Casa”, discriminando los que no posean el script TouristicDisplay, ya que estas edificaciones no son puntos de interés. También se guardará un puntero a una variable booleana del script que representa si el jugador ya ha leído la información del lugar. Una vez hecho esto, se calculará el punto más cercano al jugador de entre todos los que estén en la lista que no hayan sido visitados, rotando la flecha hacia el elegido usando el método LookAt.

Para forzar a que unity renderice la flecha por encima de todos los objetos, se utilizará un shader personalizado [19]. Un shader se trata de código que es ejecutado en la GPU y modifica los niveles de color, luz y oscuridad en la renderización de un modelado 3D.

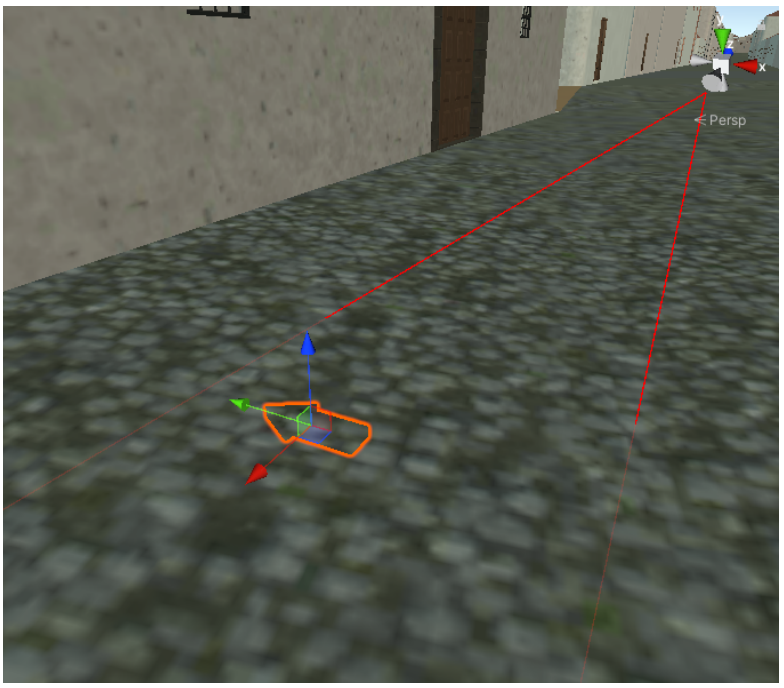


Figura 3.7.3.1: Antes de utilizar el shader.

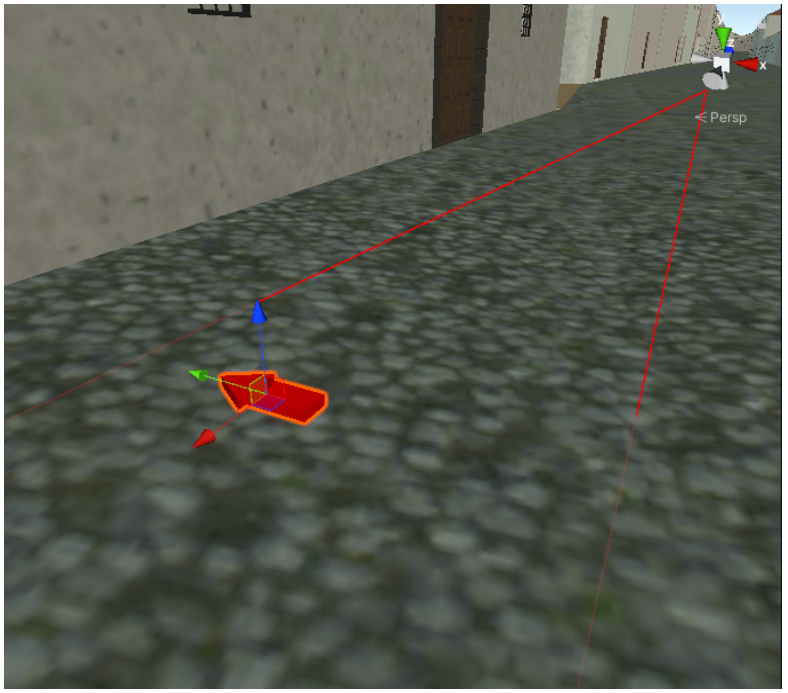


Figura 3.7.3.2: Utilizando el shader.

Figura 3.7.3.3: Captura de la aplicación donde la flecha guía al usuario a un punto de interés.

3.8 Personajes

Los tutores de este proyecto también facilitaron algunos modelos 3D de personas vestidas con la vestimenta de algunas ocupaciones y estamentos sociales de aquella época. Estos modelos fueron creados en el proyecto de reconstrucción de la ciudad de La Laguna [20]. En primer lugar, se adaptaron estos modelos en Blender, eliminando objetos que no se iban a utilizar, como la cámara de Blender y las luces y se unieron todos los objetos pertenecientes a un personaje en uno solo, para que sean más fáciles de procesar en Unity.

Una vez exportado este archivo a Unity, se posicionaron los personajes en el escenario, teniendo en cuenta dónde podrían estar dependiendo de que estén representando. Además, se

les ha incluido como punto de interés, añadiendoles un collider, el script TouristicDisplay junto a su referencia y una nueva etiqueta “Persona”. Se modificarán los scripts para que también tengan en cuenta esta etiqueta y se pueda consultar información de estos personajes de la misma forma que los demás lugares de interés.

Figura 3.8.1: Ejemplo de personaje.

3.9 Interfaz

Para poder asegurar que todos los sistemas desarrollados puedan ser usados de manera sencilla y adecuada, es necesario desarrollar una interfaz gráfica que resulte intuitiva para cualquier usuario.

Como punto de partida, se crearán 3 botones diferentes para poder acceder a los menús de información, configuración y calibración.

Figura 3.9.1: Botones.

El menú de calibración aparecerá en pantalla al abrir la aplicación y permitirá al usuario calibrar la brújula, aunque también será accesible en cualquier otro momento de la ejecución ya que es posible que este falle al realizar la primera calibración. En está sección se encuentra el diagrama de calibración y una escueta explicación sobre cómo usarlo.

Figura 3.9.2: Menú de calibración.

Como ya se ha comentado anteriormente, la velocidad de suavizado de movimiento y la altura del usuario deben de poder ser configurables. Por ello, uno de los botones, llevará a un menú con textos que representan estos valores y unos botones para aumentar o disminuir los mismos.

Figura 3.9.3: Menú de configuración.

El último menú es el que se encargará de mostrar las descripciones de los puntos de interés. Dichos textos de información deben mostrarse de manera correcta, permitiendo desplazarse por el texto si este llegase a contener más de un cierto número de carácteres. Para esta tarea, Unity aporta un tipo de objeto denominado Scroll View, que permite mover tanto imágenes como texto mediante el ratón, barras de desplazamiento o toques en la pantalla táctil.

Al utilizar solo un panel en el que se cargan textos dinámicamente, el desplazamiento de un texto no se reiniciará automáticamente cuando se cargue el siguiente, debido a que no se están creando nuevos objetos de texto sino modificando el texto de un mismo objeto al que

se le está aplicando una nueva posición cada vez que se produce un evento que active el desplazamiento de Scroll View.. Para solucionar este inconveniente, el botón que cierra el panel también cambia el valor de desplazamiento del Scroll View, que se encuentra en la propiedad **Vertical Normalized Position**, para que el siguiente texto que se cargue esté en la posición de inicio.

Esta misma técnica se usa para visualizar los nombres de los puntos de interés pero adaptando un scroll horizontal en lugar de vertical.

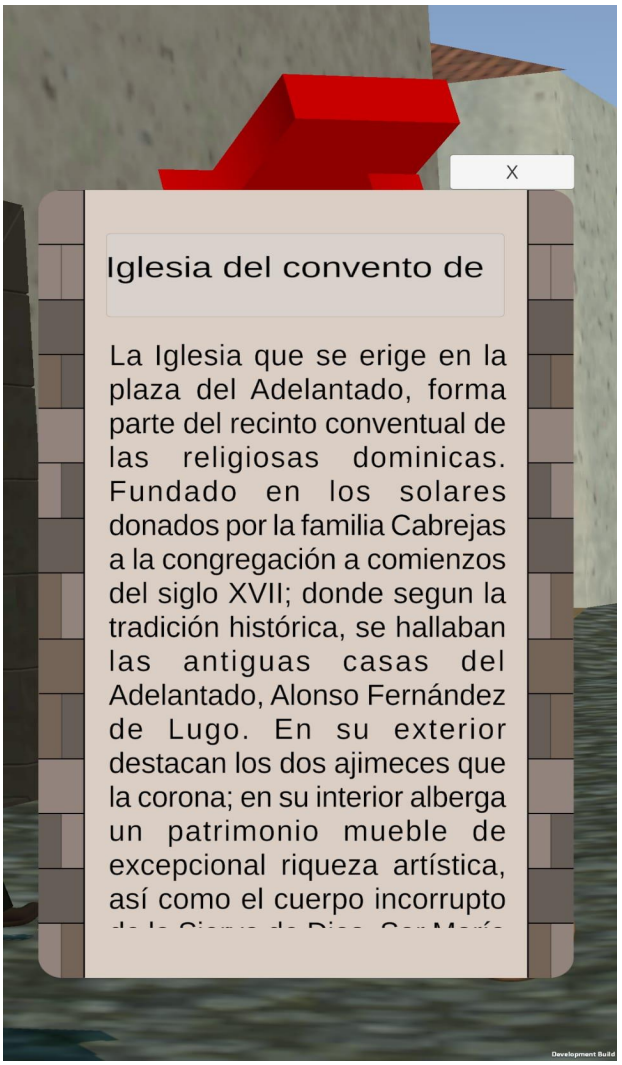


Figura 3.9.4: Menú de descripción de puntos de interés.

3.10 Diagramas de las clases utilizadas

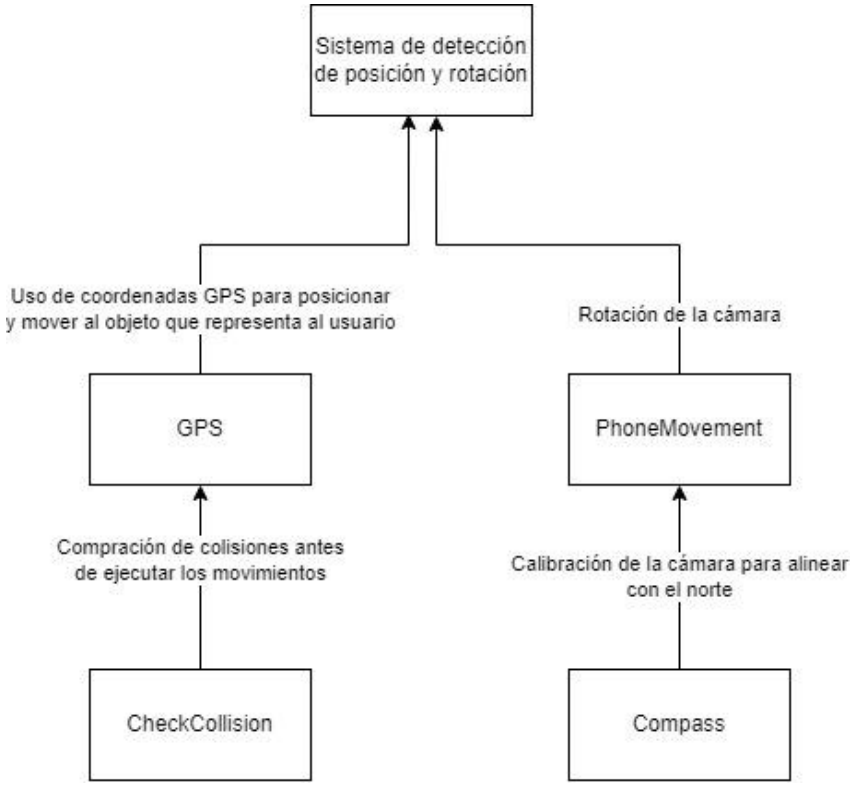


Figura 3.10.1: Diagrama de clases utilizadas para posicionar al usuario en el mapa y permitir el movimiento de cámara.

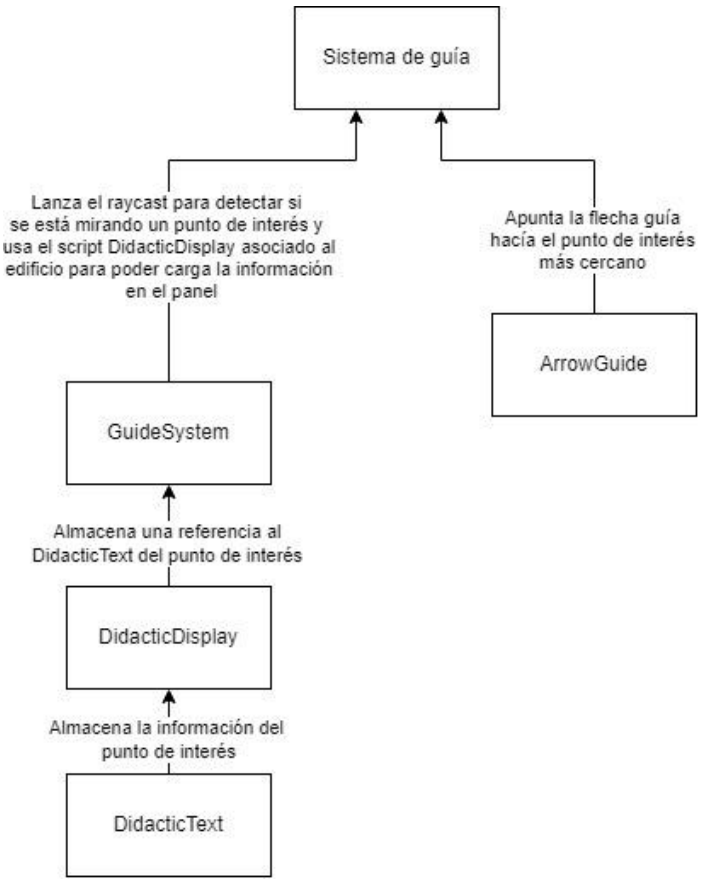


Figura 3.10.2: Diagrama de clases utilizadas para obtener la información de los puntos de interés.

3.11 Problemas surgidos durante el desarrollo del proyecto

La mayor dificultad encontrada estuvo relacionada con el desarrollo de las funcionalidades de posicionamiento por GPS, ya que se tuvieron que tener en cuenta los errores en las coordenadas obtenidas por GPS. Se tuvo que realizar un diseño de forma que estos errores no afectaran gravemente la experiencia del usuario.

Al principio se optó por mostrar el mensaje de error en coordenadas cada vez que se transportase al usuario dentro de edificaciones. Esta aproximación no generó buenos resultados debido a que si el jugador pasaba por un callejón estrecho, era muy probable que viese varias veces el mensaje de error. El sistema actual tiene en cuenta esto, de forma que el jugador podrá ver al menos lugares cerca de su situación, hasta que las nuevas coordenadas calculadas no se encuentren dentro de edificios.

Otro problema a destacar fue entender los cuaterniones para poder realizar las rotaciones necesarias de la cámara. Los cuaterniones poseen cierta complejidad debido a su estructura. Afortunadamente Unity facilita algunos métodos para que el uso de estos conceptos sea algo más simple e intuitivo.

Capítulo 4

Conclusiones y líneas futuras

4.1 Conclusiones

Para concluir, pasaremos a comentar los objetivos cumplidos y la importancia de estos.

Se ha adaptado un escenario 3D realizado en Blender para su uso como mapa en la plataforma Unity. Usando este como base, se ha diseñado un sistema de sincronización de posición y rotación de un dispositivo móvil a una escena de Unity, con el objetivo de que cuando el usuario se encuentre en la ciudad de San Cristóbal de La Laguna, pueda ver en la aplicación su situación actual pero en el siglo XVI. Para lograr este fin, se utilizaron datos provenientes de diferentes sensores del dispositivo, los cuales fueron tratados en scripts de C# para conseguir los efectos deseados.

Al ser tan detallado y grande el modelo de La Laguna, fue necesario utilizar diferentes técnicas de optimización del rendimiento para gráficos en 3D para que fuese posible ejecutar la aplicación en dispositivos móviles sin que estos tuviesen que ser de gama alta. Unity facilitó en gran manera la tarea gracias a sus herramientas y documentación, que permitieron configurar estos mecanismos de optimización de forma rápida y ágil.

Las modificaciones al escenario en Blender permitieron la implementación de colisiones en estructuras y superficies, las cuales fueron usadas para implementar un sistema de guía en pos del carácter educativo de la aplicación. Este sistema permite a los usuarios encontrar puntos de interés desde cualquier parte de la ciudad y consultar información sobre estos. Una vez hayan visitado uno de estos puntos, la propia aplicación les guiará al siguiente lugar más cercano.

Se pretende fomentar el conocimiento sobre nuestra historia mediante esta aplicación, que gracias al uso de sistemas accesibles y fáciles de utilizar por cualquier usuario, permite acceder al patrimonio histórico de una de las ciudades más importantes de las Islas Canarias.

4.2 Líneas de Trabajo futuro

En su estado actual, la aplicación constituye un producto perfectamente funcional. Sin embargo y de cara al futuro, es posible mejorarla o modificarla en diferentes ámbitos:

- Los botones, menús y otros elementos que no pertenecen al propio modelo de la ciudad no fueron elaborados por un experto en el tema, por lo que el apartado visual y artístico no tiene la calidad necesaria para un producto profesional. Se podría mejorar este aspecto de la aplicación si se contratara a un diseñador gráfico para que se encargara de estas tareas. En el diseño de estos elementos visuales se tiene que tener en cuenta también la usabilidad de la aplicación.

- La aplicación no está disponible para teléfonos IPhone ya que no se tenía dispositivos de este tipo disponibles para hacer pruebas. Para que la aplicación estuviera disponible para estos smartphones sería necesario pasar por un periodo de pruebas.

- El sistema de guía podría adaptarse para usar puntos de interés relacionados con temas específicos. Por ejemplo, se podrían introducir los lugares necesarios para virtualizar el tour “La Laguna Oculta”. Con este fin, sería interesante desarrollar software externo que permitiese añadir o cambiar los puntos de interés utilizados en el sistema sin necesidad de tener conocimientos sobre Unity.

- Por falta de tiempo, se desechó la idea de que los personajes se movieran y realizarán acciones relacionadas con su profesión, ya que hubiese sido necesario crear animaciones y optimizarlas para que estas no tuviesen un efecto negativo en el rendimiento.
- Las personas ubicadas en el modelado de la ciudad cuentan con información algo escasa. Con ayuda de alguna persona experta en historia de Canarias, podría mejorarse este aspecto.

Capítulo 5

Conclusions and future lines

5.1 Conclussions

To conclude, we will comment on the objectives achieved and their importance.

A 3D scenario made in Blender has been adapted to the Unity platform to be function as a map. Using this as a base, a system of synchronization of position and rotation of a mobile device to a Unity scene has been designed with the intention of enabling users to view their current position in San Cristóbal de La Laguna as it would have looked in the sixteenth century. To accomplish this goal, we used data from different sensors of the device, treated in C# scripts to achieve the desired effects.

As the model of La Laguna was so detailed and large, it was necessary to use different performance optimization techniques for 3D graphics to make it possible to run the application on regular mobile devices without them having to be high-end. Unity made this task easier thanks to its tools and documentation, which made it possible to configure these optimization mechanisms in a fast and agile way.

The modifications to the Blender scene allowed the implementation of collisions in structures and surfaces, which were used to implement a guidance system for the educational character of the application. This system allows users to find points of interest from anywhere in the city and check information about them. Once they have visited one of these points, the application itself will guide them to the next closest place.

The aim is to promote knowledge about our history through this application, which, thanks to the use of accessible and easy-to-use systems, allows any user to access to the historical heritage of one of the most important cities in the Canary Islands.

5.2 Future lines

In its current state, the application is a perfectly functional product. However, keeping the future in mind, it is possible to improve or modify it in different areas:

- The buttons, menus and other elements that do not belong to the city model itself were not elaborated by an expert in the subject, so the visual and artistic section does not have the needed quality for a professional product. This aspect of the application could be improved by hiring a graphic designer to take care of these tasks. The design of these visual elements must also take into account the usability of the application.

- The application is not available for iPhone phones as there were no Apple devices available for testing. In order to make the application available for these smartphones, it would be necessary to go through a testing period.

- The guidance system could be adapted to use points of interest related to specific topics. For example, the places needed to virtualize the "La Laguna Oculta" tour could be introduced. At this point, it would be interesting to develop external software that would allow adding or changing the points of interest used in the system without requiring knowledge of Unity.

- Due to lack of time, the idea of having the characters move and perform actions related to their profession was discarded, since it would have been necessary to create animations and optimize them so that they would not have a negative effect on the performance of the application. The people located in the modeling of the city have limited information. With the help of an expert in the history of the Canary Islands, this aspect could be improved.

Capítulo 6

Presupuesto

Tipo	Descripción	Cantidad	Coste Unidad
Software	Software necesario para el desarrollo del proyecto	-	0,00€
Hardware	Ordenador Sobremesa	1	699,99€
Hardware	Smartphone	1	271,99€
Formación y análisis	Formación necesaria en Unity	30 h	407,70€
Investigación	Selección y recopilación de datos de puntos de interés	16 h	226,08€
Desarrollo	Implementación de los sistemas	90 h	1223,10€
Pruebas	Pruebas in situ de la aplicación	40 h	543,60€
Total			3372,46€

Tabla 5.1: Resumen del presupuesto

6.1 Justificación del presupuesto

En cálculo del presupuesto se han incluido todo el hardware y software necesario, así como las horas necesarias para la realización del proyecto.

El software utilizado en el proyecto es gratuito, por lo que no influye en el precio total.

Para el ordenador de sobremesa, se ha elegido un equipo de gama media, con el fin de no tener problemas de rendimiento cuando se modifique el modelo de la ciudad en Blender o Unity. En cuanto al teléfono móvil, se ha elegido uno que aparecía en un artículo sobre smartphones de gama media [21].

El coste de desarrollo y pruebas está basado en información encontrada sobre el precio medio por hora de un ingeniero informático [22] (13,59€/h).

El coste de investigación está basado en el salario medio de un historiador [23] (14.13€/h). Se necesita un historiador ya que este podrá identificar qué puntos de interés son relevantes.

Bibliografía

[1] Javier Ramos. “San Cristóbal de la Laguna: por algo es Patrimonio de la Humanidad (Tenerife) (Español)”. LugaresConHistoria. En (2016). url: https://www.lugaresconhistoria.com/san-cristobal-de-la-laguna-tenerife

[2] Karthic Prabu. “Japan Tourist Attraction uses digital penguins to increase visits. (Inglés)”. Phocus Wire. En (2013). url: https://www.phocuswire.com/Japan-tourist-attraction-uses-digital-penguins-to-increase-visits

[3] ViewAR. “Guide Bot Documentation. (Inglés)”. En (2020). url: https://documentation.viewar.com/docs/tutorials/template_tutorials/guidebot

[4] Blender Foundation. “Blender About Section (Inglés).” url: https://www.blender.org/about/

[5] Unity Technologies. “Made with Unity (Español).” url: https://unity.com/es/madewith

[6] GlitchEnzo. “NuGetForUnity (Inglés)”. En (2022) url: https://github.com/GlitchEnzo/NuGetForUnity

[7] j1mmyto9 “Speech and text Unity Ios Android (Inglés)”. En (2022) url: https://github.com/j1mmyto9/Speech-And-Text-Unity-iOS-Android

[8] Unity Technologies ApS “Unity Remote 5 (Inglés)”. En (2021) url: https://play.google.com/store/apps/details?id=com.unity3d.mobileremote&hl=es_419&gl=US

[9] Statista “Market share of leading mobile operating systems in Europe from 2010 to 2021 (Inglés).” En (2022). url: https://www.statista.com/statistics/639928/market-share-mobile-operating-systems-eu/#:~:text=As%20of%202021%2C%20Android%20was,percent%20of%20European%20mobile%20phones.

[10] Universitat Politècnica de València - UPV. “Coordenadas UTM | UPV (Español)”. En: (2014). url: https://www.youtube.com/watch?v=I0An2yVPhA8

[11] Djexplo “Illustration of geographic latitude and longitude of the earth (Inglés)”. En (2011). url: https://en.wikipedia.org/wiki/File:Latitude_and_Longitude_of_the_Earth.svg

[12] Sparkfun “GPS Basics (Inglés)”. En (2012). url: https://learn.sparkfun.com/tutorials/gps-basics/all#:~:text=Update%20Rate%20%2D%20The%20update%20rate,available%20in%20low%20cost%20modules.

[13] Google “Cómo utiliza Google la información de ubicación (Español)”. url: https://policies.google.com/technologies/location-data?hl=es-Es

[14] Leopoldo Armesto. “Cuaterniones | Sistemas Robotizados (Español)”. En: (2020). url: https://www.youtube.com/watch?v=EsThkmS1NlA

[15] Microsoft “Sistemas de coordenadas (Español)”. En (2022). url: https://docs.microsoft.com/es-es/windows/uwp/graphics-concepts/coordinate-systems

[16] Unity “Occlusion Culling (eliminación selectiva) (Español)”. En (2019). url: https://docs.unity3d.com/es/2018.4/Manual/OcclusionCulling.html

[17] Fundación Centro Internacional para la Conservación del Patrimonio CICOP “Gestor Patrimonio Cultural (Español)”. url: http://gestorpatrimoniocultural.cicop.com/Patrimonio-en-SAN_CRIST%C3%93BAL_DE_LA_LAGUNA

[18] davidonionball “Arrow Pointer (Inglés).” En (2018). url: https://www.cgtrader.com/free-3d-models/architectural/other/arrow-pointer

[19] Usuario Sinuosity en Reddit “Shader para renderizar por encima de todos los objetos”. En (2013). url: http://www.mediafire.com/file/3q8wcw9rxhw99cg/FrontShaders.zip/file

[20] Cecile Meier, Isabel Sanchez Berriel y Fernando Pérez Nava “Creation of a Virtual Museum for the Dissemination of 3D Models of Historical Clothing (Inglés)” En (2021). url: https://www.mdpi.com/2071-1050/13/22/12581

[21] Ricardo Aguilar “Los mejores móviles de gama media que puedes comprar en 2022 (Español).” En (2022). url: https://www.xatakamovil.com/guias-de-compra/mejores-moviles-gama-media-que-puedes-comprar-2021

[22] talent.com “Salario medio para Ingeniero Informático en España, 2022 (Español).” url: https://es.talent.com/salary?job=ingeniero+inform%C3%A1tico#:~:text=El%20salario%20ingeniero%20inform%C3%A1tico%20promedio,hasta%20%E2%82%AC%2035.000%20al%20a%C3%B1o.

[23] tusalario.es “Filósofos, historiadores y especialistas en ciencias políticas (Español).” url: https://tusalario.es/carrera/funcion-y-sueldo/filosofos-historiadores-y-cientificos-politicos