Escuela Politécnica Superior de Jaén

Grado En Ingeniería Informática

Universidad de Jaén

Escuela Politécnica Superior (Jaén)

Trabajo de Fin de Grado

Modelado Urbano con Texturización asistida de fachadas

Alumno/a: Fernando José Salazar Ortega

Tutor/a: Dña. Lidia Ortega Alvarado

Dpto.: Departamento de Informática

Mayo, 2019

Universidad de Jaén

Escuela Politécnica Superior de Jaén

Departamento de Informática

Dña. Lidia Ortega Alvarado, tutora del Proyecto Fin de Carrera titulado: Modelado urbano con texturización asistida de fachadas, que presenta Fernando José Salazar Ortega, autoriza su presentación para defensa y evaluación en la Escuela Politécnica Superior de Jaén.

Jaén, mayo de 2019

El alumno: Los tutores:

Fernando José Salazar Ortega Dña. Lidia Ortega Alvarado

Agradecimientos

En este capítulo, quiero transmitir mis sinceros agradecimientos a todas las personas que han contribuido positivamente en el transcurso del desarrollo de este trabajo fin de grado. Una cordial mención para:

Mi familia que durante todos los días me han estado apoyando de forma incondicional y sugiriendo ideas con el objetivo de realizar este proyecto de la mejor forma posible. También a mis amigos que me han asistido en la corrección de la aplicación con sus consejos y en la organización de esta documentación.

A mí tutora, Lidia Ortega Alvarado, quien me ofreció este proyecto fin de grado y gracias a los cuales he aprendido a la vez que mejorado en un gran número de conocimientos que he puesto en práctica hasta la finalización de este trabajo. Su participación, dedicación, así como disponibilidad para resolver las dudas que le he ido planteando y gracias a ello, se ha podido conseguir un resultado final novedoso a la vez que competitivo.

Muchas gracias a todos.

CONTENIDO

ÍNDICE DE IMÁGENES

ÍNDICE DE TABLAS

ÍNDICE DE ECUACIONES

RESUMEN

Un modelo 3D es la representación de un conjunto de elementos y propiedades que, tras un proceso de renderización y con la utilización de un programa de dibujo, se convierte en una imagen. Estos modelos son en realidad una colección de datos (vértices, aristas…) representando una multitud de polígonos conectados entre sí, como triángulos, cuadrados, etc. Estos modelos nos permiten visualizar de forma realista los elementos del mundo real de manera virtual, al contrario que los modelos 2D en los que las dimensiones no se asemejan tanto a nuestra realidad. Una ventaja de usar modelos tridimensionales es su facilidad para crearlos, por ejemplo, escaneando los objetos deseados con un escáner 3D o mediante diversas herramientas como Unity.

Entre las posibles aplicaciones de estos entornos virtuales están, por ejemplo, entrenar al usuario en ciertas tareas o realizar acciones de la vida cotidiana. Todo esto ofreciendo al usuario una experiencia realista sin necesidad de tener que hacerlo en el mundo real. Con el fin de conseguir ese objetivo en el proyecto se requiere utilizar de modelos 3D y texturas realistas, siendo esta última una imagen de mapa de bits que envuelve a un objeto virtual de dos o tres dimensiones.

El proyecto presenta una herramienta que asiste al usuario con el objetivo de texturizar los edificios con las texturas de ejemplo o las añadidas por él, con las cuales podrá ver cómo quedaría su casa o negocio basándose en su diseño. Esta herramienta servirá para obtener una representación visual de los diferentes edificios con las texturas elegidas por el usuario. Esto le permitirá traspasar posteriormente este boceto a la realidad.

En este proyecto se podrá explorar un entorno 2.5D con construcciones previamente situadas, cuyo objetivo es tener representados edificios que tras ser texturizados se perciban de forma realista. Otra característica que contendrá la aplicación es la posibilidad de que el usuario pueda explorar el entorno virtual, así

como funciones para facilitar dicha navegación. Además, podrá interactuar con los diversos elementos que compone el proyecto como los menús, los edificios, etc.

ABSTRACT

A 3D model is the representation of a set of elements and properties, that after a rendering process and with the utilization of a drawing program, becomes an image. These models are actually a collection of data (vertex, edges ...) representing a multitude of polygons connected each other, such as triangles, squares, etc. These models allow us to realistically visualize the elements of the real world in a virtual way, unlike the 2D models in which the dimensions do not resemble our reality so much. One advantage of using three-dimensional models is their ease in creating them, for example, by scanning the desired objects with a 3D scanner or using several tools such as Unity.

Among the possible applications of these virtual worlds are, for example, to train the user in certain tasks or perform actions of daily life. All this offering the user an immersive experience without having to do it in the real world. In order to achieve this goal in the project, it is necessary to use 3D models and realistic textures, the latter being a bitmap image that surrounds a virtual object of two or three dimensions.

The project presents a tool that assists the user with the objective of texturing the buildings with sample textures or those added by him, which he will can see how his house or business would be based on his design. This tool will serve to obtain a visual representation of the different buildings with the textures chosen by the user. This will allow you to later transfer this sketch to reality.

In this project you can explore a 2.5D world with previously located buildings, whose objective is to have represented buildings that after being textured are perceived realistically. Another feature that will contain the application is the possibility that user can explore the virtual world, as well as functions to facilitate such navigation. In addition, you will can interact with the various elements that compose the project such as menus, buildings, etc.

DEFINICIONES Y ABREVIATURAS

Software: Conjunto de componentes lógicos que se encargan de hacer tareas específicas y se encarga de mandar instrucciones al hardware.

Hardware: Componentes físicos de un sistema informático como componentes eléctricos, electromecánicos, cables, gabinetes y los distintos periféricos.

TFG: Trabajo que ha de realizar el estudiante para obtener su titulación, debiendo ser un trabajo original y estar orientado por un tutor/a.

BBDD: Biblioteca virtual de documentos que pertenecen a un mismo contexto que son almacenados para su posterior uso.

SmartGit: Es un software que gestiona las diversas versiones que contiene un proyecto. Su propósito es la de llevar un registro de los cambios que se realice en el proyecto y la persona que los ha realizado.

Redmine: Esto es una herramienta para la gestión de los proyectos mediante distintas funcionalidades como gestor de usuarios, diagramas de Gantt, etc.

Mipmapping: Es una colección de imágenes preferiblemente de tamaño múltiplo de dos que acompañan a una textura original siendo estos múltiplos de dos menores a la original.

Scrum: Nos referimos a una metodología de trabajo para la correcta realización del proyecto que consiste en implementar pequeños incrementos y mostrarlos al cliente para obtener retroalimentación.

Sprint: Consiste en una iteración del proyecto para realizar un entregable para comprobar su funcionamiento, el sprint puede comprender de 2 semanas a dos meses como mínimo y máximo.

SQLITE: Es un sistema gestor de base de datos en el que el servidor está integrado en el programa.

Android: Sistema operativo para móvil basado en el código fuente de Linux.

CF: Costes fijos del proyecto.

CV: Costes variables del proyecto.

COCOMO: Es un Modelo Constructivo de Costos que sirve para la estimación de costos de una manera empírica, existiendo distintos modelos que se han de adaptar al proyecto.

UML: Lenguaje unificado de modelado, consiste en visualizar, construir y documentar un proyecto mediante una imagen detallando cómo son los pasos para la realización del proyecto.

Diagrama de secuencia: Diagrama que muestra la secuencia de vida de los distintos elementos del proyecto mostrando la relación entre ellos.

RawImagen: Una imagen no interactiva que puede utilizar cualquier formato de imagen que haya en el editor de Unity.

Sprite: Textura de uso 2D que contiene técnicas especiales para mejorar su eficiencia.

Image: Elemento que muestra un “Sprite” en la interfaz.

CAPÍTULO 1: INTRODUCCIÓN

1.1 Motivación

El mundo está cambiando y cada vez es más necesario ayudar a los usuarios y a las empresas para afrontar las nuevas necesidades que aportan las nuevas tecnologías. Es necesario crear nuevos programas que aporten una solución a las necesidades que aún estén sin resolver. Las distintas tecnologías que utilizan entornos virtuales han ayudado a recrear una visión 3D de objetos de uso habitual, como mobiliario para edificios, simuladores de experiencias gracias a herramientas diversas como son las gafas 3D ya existentes en el mercado.

Teniendo en cuenta las distintas zonas que contiene una ciudad, con esta aplicación se desea dar un soporte que modele las mismas para que el usuario pueda añadir texturas a su gusto a los edificios hasta encontrar un entorno virtual con el que se sienta satisfecho.

Como la población está en aumento y con ello el número de negocios que abren, estos emprendedores tienen a su disposición nuevas tecnologías en las que apoyarse para comenzar dicho negocio sin necesitar la asistencia de profesionales externos para diseñar su imagen de marca.

En este proyecto se plantea hacer un prototipo con el cual, cualquier usuario pueda elaborar una texturización en los modelos 3D de edificios que aporta la aplicación, teniendo así una visión gráfica de cómo quedaría esa calle en el mundo real en base al modelo [1].

1.2 Objetivo General

Para este Trabajo de Fin de Grado titulado “Modelado Urbano con Texturización Asistida de Fachadas” se va a describir el desarrollo de los modelos 3D creados. En este ámbito de investigación y desarrollo cuyo objetivo es la creación de una aplicación para dispositivos móviles, en la cual se utilicen las diversas texturas almacenas en la BBDD para colocarlas en los edificios que el usuario precise de un modo accesible y rápido.

Gracias al diseño de esta herramienta se facilita la navegación por el entorno virtual, que asistirá al usuario en el proceso de texturización de los distintos edificios, proporcionando facilidades a la hora de añadir nuevas texturas con unos pocos toques.

Como resultado final, se requiere que haya una correspondencia entre el mundo real y virtual con el menor coeficiente de error posible. Debido a estos requisitos, se ha realizado un Modelo 3D con unos edificios simulados, que serán texturizados posteriormente por el usuario.

1.3 Antecedentes y evolución tecnológica

Antes de comenzar, debemos definir que es un entorno virtual, generalmente conocido como la recreación de un entorno tridimensional que simula al real en su topografía, condiciones sociales, económicas y de comunicación, pero contando con limitaciones dadas por la potencia de los computadores.

Ofrecen al usuario un mundo en el que pueda interactuar y explorar un lugar real o imaginarios, gracias a la recreación de estos como se observa en la imagen siguiente un modelo de New York Ilustración 1:

Ilustración 1: Modelo 3D de una ciudad con Google Earth

Estos mundos ofrecen una experiencia como la vida misma en la que pueden incluir la amistad, economía, la guerra y la política. Ofreciendo al usuario poder hacer negocios, formarse académicamente, viajar y realizar muchas actividades.

La primera máquina creada para dar esa sensación de un entorno virtual fue la máquina Sensorama [2] que transmitía información al usuario mediante la visión, el sonido, el equilibrio y hasta el tacto con el fin de realizar una simulación de la vida misma. Trataba de engañar al sistema perceptivo para hacernos sentir inmerso en el universo creado introduciendo lo que conocemos como realidad virtual. Obteniendo más realismo cuando el usuario tiene la capacidad de moverse por el entorno pudiendo comprobar lo que ofrece y como puede interactuar con los diferentes elementos.

Unos ejemplos de los campos recreados por la realidad para su empleo han sido:

- Social: Como se ha dicho al principio de este apartado las diversas utilidades que se consiguen con este tipo de aplicaciones son importante para la comunicación y mejora la interacción con los demás usuarios mientras

pasas un buen rato explorando cada rincón del entorno. Algunos ejemplos de aplicaciones son: Second Life y Active Worlds.

- Médico: Se puede utilizar este tipo de tecnología con finalidad terapéutica, como por ejemplo en aquellas personas que sufren de algún problema como puede ser el autismo, se pueden crear entornos para que se sienta cómodos consiguiendo interacciones experimentales que de ninguna otra manera pueda obtener óptimos resultados Ilustración 2 [3].

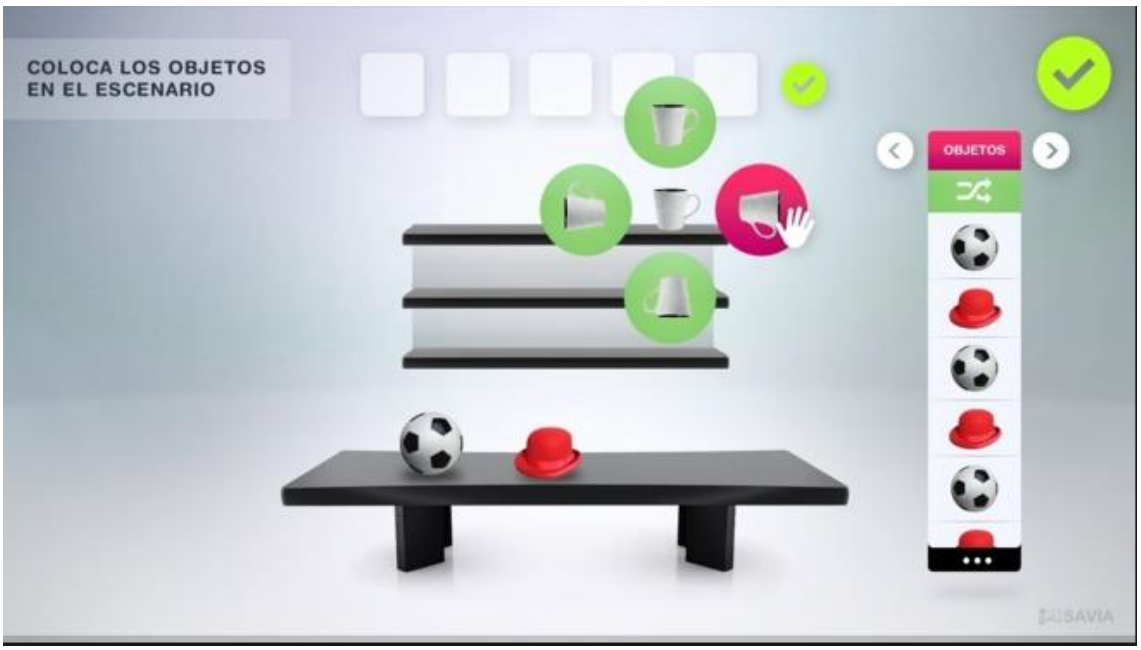

Ilustración 2: Entorno virtual que ayuda a los niños con autismo

- Comercial: Como mundo que es, no se libra de los comercios feroces como el ejemplo de Apple que ha creado una tienda virtual en la aplicación Second Life mencionada anteriormente. Usando estos mundos las empresas pueden recibir retroalimentación de parte de los clientes para el desarrollo de sus proyectos.

1.4 Contexto del problema

La finalidad del porqué se va a realizar este proyecto reside en que no se han encontrado aplicaciones encargadas de obtener una previsualización del exterior de cualquier edificio. Por este motivo se creó esta aplicación, para utilizar diversas texturas en las paredes, escaparate, etc. Google Maps es una aplicación que muestra una “previsualización” del exterior de los edificios, aunque tiene el problema de que algún objeto se interponga entre la cámara y el edificio, sin poder observar la fachada de manera clara. Otro problema trata sobre la actualización del sistema, del cual es necesario que el coche de Google Maps actualice las calles toma un tiempo considerable.

Buscando aplicaciones específicas para la texturización de fachadas, no se han encontrado aplicaciones relacionadas al objetivo de este proyecto sino otras con objetivos distintos. Por tanto, esta aplicación ayudará a resolver el problema con la decoración del exterior de las casas o tiendas de los usuarios.

Por ejemplo, hay aplicaciones que ayudan al usuario a decorar el interior del inmueble como es Home Design 3D, resultando viable que también existan maneras de obtener una decoración de las paredes. Para ello, se ha pensado en crear una herramienta con el objetivo de que el usuario pueda colocar las texturas en las fachadas de manera sencilla y asistida. Con el fin de visualizar como quedaría las distintas texturas con un resultado realista y luego se pueda guardar el proceso en la Base de datos para su posterior recuperación.

Por último, la aplicación tratará obtener entornos similares a la realidad para conseguir en un futuro traspasar el boceto creado al mundo real. Además, el usuario podrá agregar más texturas consiguiendo recrear la casa o tienda que anhela, obteniendo así una imagen 3D de un edificio real con su fachada texturizada.

1.5 Planteamiento del problema

Con el objetivo de satisfacer las necesidades del proyecto, sería necesario una correcta gestión y organización de la Base de datos sobre la distintas texturas y muros texturizados. Esa es una de las tareas primordiales si lo que se busca es una buena calidad a la hora de usar todos los servicios que aportará la aplicación. Como entorno virtual se ha optado por la recreación de tres calles con los edificios separados entre sí para la validación de las diversas pruebas. Se va a realizar de esta manera, con el motivo de que el usuario pueda observar y texturizar los laterales de los edificios.

La solución propuesta de este proyecto es escalable a cualquier tipo de edificio que se construya en el futuro, dando lugar a que se pueda representar no solo barrios, sino ciudades enteras con la ayuda de esta aplicación.

En el caso de este proyecto nos debemos centrar en las imágenes que serán necesarias para la texturización de los distintos modelos 3D de los edificios. Para ello tendremos que plantear varios factores como la de obtener una imagen comprobando su tamaño con el fin de comprobar cuantas veces podríamos subdividirla sin que quede mal en un edificio. Tener un identificador para encontrar dicha imagen cuando se requiera, guardar la propia imagen, etc. Estas características serán necesario para poder finalizar el proyecto satisfactoriamente.

Uno de los aspectos importantes de este proyecto, es buscar hacer más fácil el trabajo de los usuarios, ayudarles a perder el menor tiempo posible hasta completar sus objetivos. Ambos requerimientos son los retos marcado por los distintos desarrolladores de aplicaciones y con esta herramienta podrán decorar a su gusto las distintas paredes de las viviendas mientras explora las calles de la ciudad de una forma interactiva e intuitiva.

1.6 Metodología

La metodología a usar está enfocada al modelo incremental Scrum basado en el libro: *MÉTODOS ÁGILES: SCRUM, KANBAN, LEAN* [4]. Que consiste en la realización de un Sprint teniendo preparado un ejecutable cada dos/tres semanas de cada versión del proyecto para ir viendo cómo va prosperando el trabajo, si se está realizando bien los requisitos o si hay algún posible problema poder tratarlo.

Este proceso va seguido de distintas revisiones todas ellas comprobadas y corregidas con la tutora proporcionando una retroalimentación al proyecto. Para guiar cada iteración, se ha realizado un historial con todas las tareas que se deben tener implementadas en ese prototipo viéndose más detalladamente en el apartado de diseño. En dicho documento se incluyen las distintas funcionalidades que se han añadido, modificaciones realizadas al diseño, etc. Será revisado por la tutora de manera que se priorice tener los elementos más importantes del proyecto listos tan pronto como sea posible.

La finalidad del uso de esta metodología es la de poder gestionar las expectativas de la tutora de manera periódica, con el fin de comprobar si todas las decisiones tomadas en cada iteración han sido acertadas hasta alcanzar la última revisión, completando el proyecto. Dividiendo en tres iteraciones de dos a tres semanas como se dijo previamente y añadiendo nuevas funcionalidades en cada iteración. En cada incremento se realizarán las cuatro fases: análisis, diseño, implementación y pruebas.

Tras completar cada una de las iteraciones se obtendrá una versión beta con la que la tutora de este trabajo podrá examinar y usar la aplicación. El uso de este modelo ha sido útil en los casos en los que no se tiene una especificación completa de los diversos requerimientos del sistema que pueden variar en las primeras iteraciones del proyecto. Esta naturaleza incremental nos da la posibilidad de ofrecer pequeñas soluciones hasta definir el producto final. Las principales ventajas que aporta esta metodología son las siguientes descritas:

- Al aportar soluciones parciales en cada iteración, se reduce el tiempo dedicado al desarrollo inicial de la aplicación.

- La tutora comprobará periódicamente las diversas versiones de la aplicación ofreciendo retroalimentación al desarrollador de la aplicación, aportando que el desarrollo obtenga menos errores.

- Al estar dividido el proyecto en secciones resulta más sencillo realizar pequeños cambios que mejoren una de las secciones.

1.7 Objetivos Específicos

El objetivo de este trabajo es realizar una aplicación móvil que permita decorar las paredes exteriores de los edificios que están dentro del entorno virtual. Para ello será necesario la realización de los siguientes procedimientos:

- Para poder empezar a realizar estos objetivos, primero es necesario la recreación de un entorno virtual 3D que contenga edificios y calles.

- El usuario podrá navegar libremente por el entorno construido y podrá interactuar con varios elementos.

- Se podrá interactuar con las diversas paredes de los edificios para su posterior adición de las texturas.

- Tras interactuar con las paredes de la manera mencionada en el punto anterior, el usuario podrá colocar de manera automática las diferentes texturas que tenga a su disposición.

- Si no hay textura que convenza al usuario, este podrá añadir más imágenes desde su dispositivo móvil con el uso de la cámara en la calle o desde la galería del usuario.

- Luego de que el usuario haya obtenido las fotografías de las diversas texturas con su cámara o desde fotos obtenidas con su ordenador. El usuario podrá realizar la texturización mientras esté en la calle o desde su casa.

- Este proyecto está enfocado a un posible uso comercial desarrollando una aplicación que ayude en el proceso de decoración de fachadas del entorno virtual para en un futuro disponer de un esbozo trasladable a la realidad.

1.8 Estructura de la memoria

Esta documentación está dividida en 7 capítulos presentando todos los hechos relacionados con este proyecto en función de la característica que describe siguiendo una metodología incremental. Este documento se desglosa, por tanto, en los siguientes apartados:

Para empezar, se introduce el proyecto en el Capítulo 1: INTRODUCCIÓN con una descripción con el motivo por el cual se ha elegido este TFG. Además, se define su contexto y se realiza una visión del problema a resolver y la metodología usada, explicando los objetivos generales y específicos establecidos en el proyecto.

En el Capítulo 2: TECNOLOGÍAS Y HERRAMIENTAS, se contemplan las distintas herramientas usadas en la implementación del proyecto, centrándose en Unity y en el lenguaje de programación C#. También se realiza una explicación sobre por qué se optó por utilizar Unity y no otros.

Durante el Capítulo 3: PLANIFICACIÓN, se describirán todas las tareas que componen el proyecto a través de distintos gráficos mostrando las relaciones entre las diversas actividades, el tiempo que tardan las tareas y quien las realiza. Otro apartado que se hablará es sobre la estimación de los costes, contando

con los elementos usados en el proyecto como son los costes hardware, software y humanos, etc.

Para el Capítulo 4: ANÁLISIS DEL SISTEMA, se muestran los distintos requisitos funcionales y no funcionales contemplados en la aplicación. Además, se explica la planificación de los requisitos a través de las diversas iteraciones de la aplicación de una manera más precisa.

A continuación, en el Capítulo 5: DISEÑO DEL SISTEMA, se explican los elementos que contiene este proyecto mediante distintos esquemas relacionales entre las distintas clases, como diagramas UML, de secuencia y con descripciones del contenido de la interfaz del proyecto.

Tras obtener el diseño del proceso, prosigue el Capítulo 6: IMPLEMENTACIÓN DEL SISTEMA, el cual muestra el desarrollo del proyecto de los distintos elementos que componen el proyecto mediante explicaciones y código.

Para finalizar este documento con el Capítulo 7: CONCLUSIONES explica las deducciones finales a las que se han llegado durante el desarrollo de este proyecto y cómo, en un futuro, esta aplicación puede ser más útil para los usuarios con las nuevas mecánicas pero que se salen del planteamiento inicial del proyecto.

Como Anexo A: MANUAL DE INSTALACIÓN se explicará el cómo instalar los elementos necesarios para poner en marcha Unity y el SDK de Android para poder empezar con este proyecto.

Como Anexo B: IMÁGENES EN ALTA RESOLUCIÓN, Se podrá observar las imágenes con mejor detalle que de otro modo no se vería o entendería correctamente.

CAPÍTULO 2: TECNOLOGÍAS Y HERRAMIENTAS

2.1 Estudio de herramientas alternativas y viabilidad

El primer paso para realizar este proyecto es plantear en qué sistema operativo para dispositivos móviles se va a realizar la aplicación, tratando de conseguir que la mayor cantidad de usuarios puedan utilizar el programa. Tras realizar un análisis del uso de los diferentes sistemas, se realizará la aplicación para los dispositivos Android. Como se muestra en la siguiente imagen se ve la diferencia en cantidad de usuarios de los distintos sistemas operativos Ilustración 3 [5]:

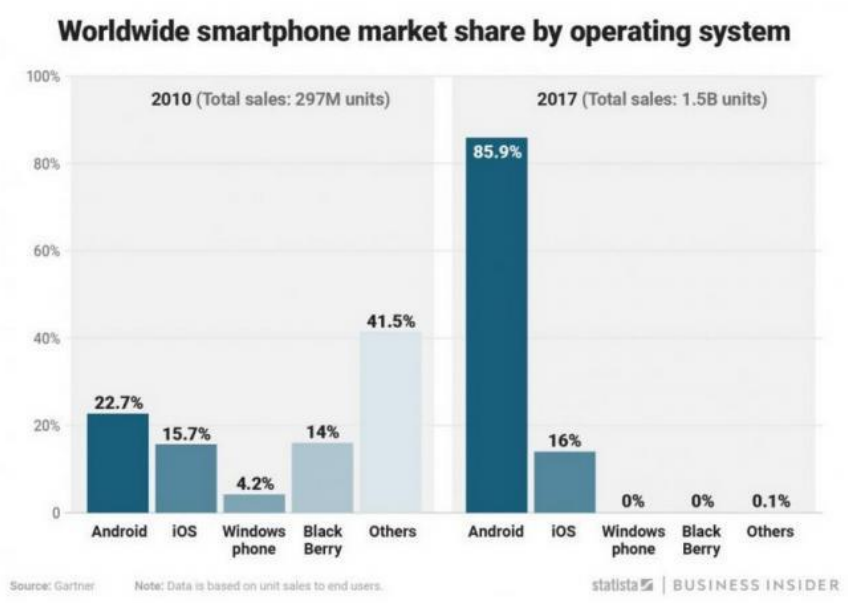

Ilustración 3: Utilización de los diferentes sistemas operativos 2010 y 2017

Como se puede comprobar en esta imagen, es recomendable programar para dispositivos Android por la cantidad de usuarios que lo utilizan.

En el planteamiento del proyecto, también se ha realizado un estudio sobre las distintas herramientas para comprobar las diversas ventajas e inconvenientes en

la creación de esta aplicación. Entre ellas se analizó su curva de aprendizaje, características que hacen que la herramienta se adapte a este proyecto y la facilidad de manejo de la interfaz. En este apartado se hablará sobre la herramienta que se ha descartado y posteriormente la que ha sido utilizada en este proyecto.

Para la realización del proyecto, en un principio se pensó utilizar Android Studio [6], que es un potente entorno de desarrollo integrado (IDE) en la elaboración de aplicaciones para dispositivos Android, con una interfaz que separa la parte visual de la aplicación del código Ilustración 4 [7].

Ilustración 4: Icono de Android Studio

La interfaz que nos aporta Android Studio nos permite observar el diseño de la aplicación directamente sin necesidad de compilar el trabajo en un dispositivo físico. Así conseguimos realizar un esbozo fácil y rápido del resultado final de la aplicación con los diversos elementos que aporta la plataforma como se puede observar en la siguiente imagen Ilustración 5 [8].

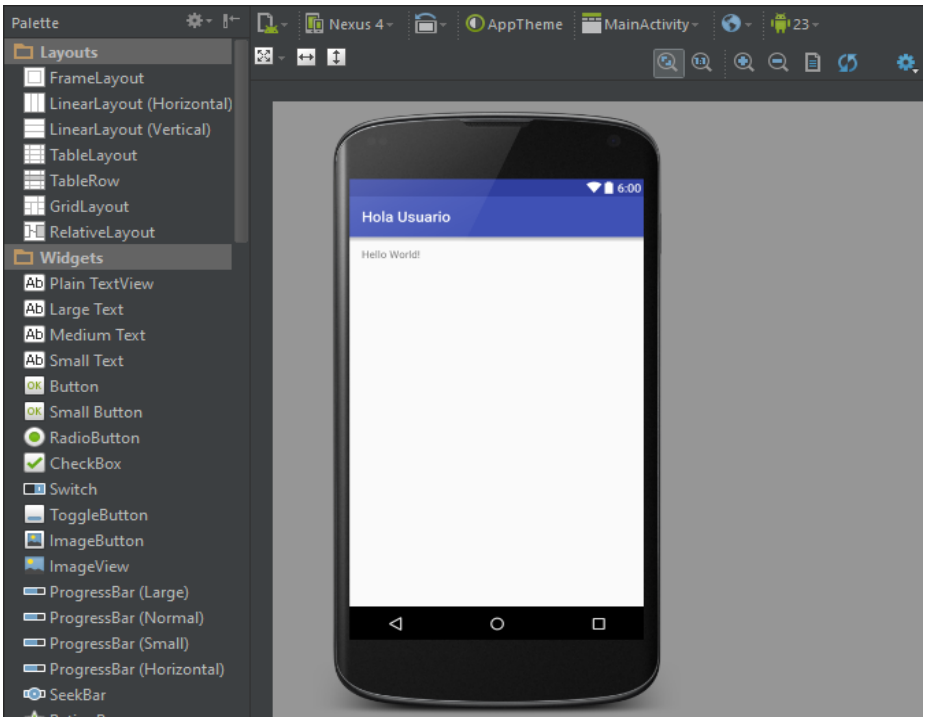

Ilustración 5: Prueba de contacto con Android Studio

La interfaz gráfica de usuario de Android Studio es similar a las de Visual Studio y Netbeans, haciendo más intuitiva la realización del proyecto debido a la experiencia previa de los programadores. Además, esta herramienta utiliza Java, lenguaje con el cual los programadores están familiarizadas como se muestra en la siguiente fotografía Ilustración 6 [9].

```
import android.support.v7.app.AppCompatActivity;
import android.widget.Button;

public class NiceFormattedActivity extends AppCompatActivity {

    private Button mButton;

    public static Intent newInstance(@NonNull final Context context) {
        return new Intent(context, NiceFormattedActivity.class);
    }

    @Override
    protected void onCreate(@Nullable final Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_hello_world);
        mButton = findViewById(R.id.button);
    }

    @Override
    protected void onResume() {
        super.onResume();
    }

    @Override
    protected void onPause() {
        super.onPause();
    }
```

Ilustración 6: Prueba de contacto de programación con Android Studio

Otra característica positiva encontrada en el programa es que puedes decidir la retrocompatibilidad con distintas versiones software de los sistemas operativos Android, a coste de un mayor tamaño de aplicación debido a la necesidad de descargar más plantillas. Queda a decisión del programador equilibrar la cantidad de usuarios que puedan utilizar la aplicación frente al tamaño de la misma Ilustración 7 [10].

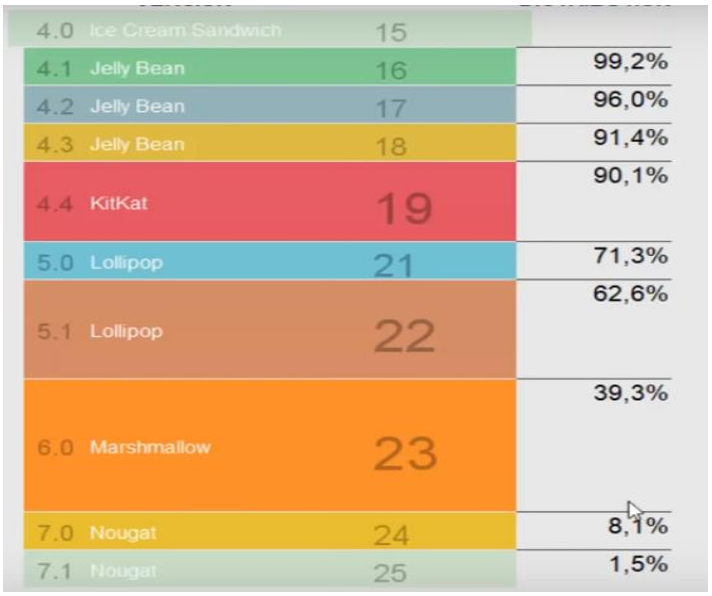

Ilustración 7: Porcentaje de usuarios en las distintas versiones Android

El motivo principal para desestimar esta herramienta fue la falta de componentes de desarrollo en 3D, siendo necesarias herramientas externas para este propósito, agregando al proyecto un gran coste de tiempo y esfuerzo para aprender a realizar el entorno virtual. Debido a esta problemática se decidió utilizar como entorno de desarrollo Unity3D, el cual ofrece mejor soporte para modelado 3D.

2.2 Motor de desarrollo Unity3D

Ilustración 8: Icono de Unity

Después del análisis previamente realizado, se decidió emplear la herramienta de videojuegos Unity (Ilustración 8) [11], por su sencillez en la creación y manejo de las diversos modelos 3D de las calles y construcciones. Con la ayuda del conocimiento obtenido de la asignatura de Desarrollo de Videojuegos y gracias al estudio de varios libros sobre Unity y procesamiento de imágenes, no se empieza desde cero en el aprendizaje de esta herramienta [12] [13] [14].

Unity3D es propiedad de *Unity Technologies,* y es uno de los motores de juego más importantes y extendidos, permitiendo utilizar *JavaScript* y C# (entre otros), pero más importante es el soporte técnico, provisto por la gran cantidad de usuarios de la que dispone la herramienta. Además, se pueden encontrar diversos plugins que ayudan a complementar algunos elementos ausentes en la versión original de Unity3D mediante la tienda online accesible desde la propia herramienta.

Se utilizará Unity3D para la realización de las pruebas de texturización explicadas en el apartado 4.4 de este documento. Al principio se planteó el uso de un archivo .cat, usando las coordenadas de los distintos puntos de una calle. Las polilíneas creadas a partir del archivo requerirían de una limpieza manual haciendo costosa la carga inicial del proyecto. A causa de esto, se propuso construir un modelo de una calle virtual con modelos 2.5D para la texturización de edificios Ilustración 9.

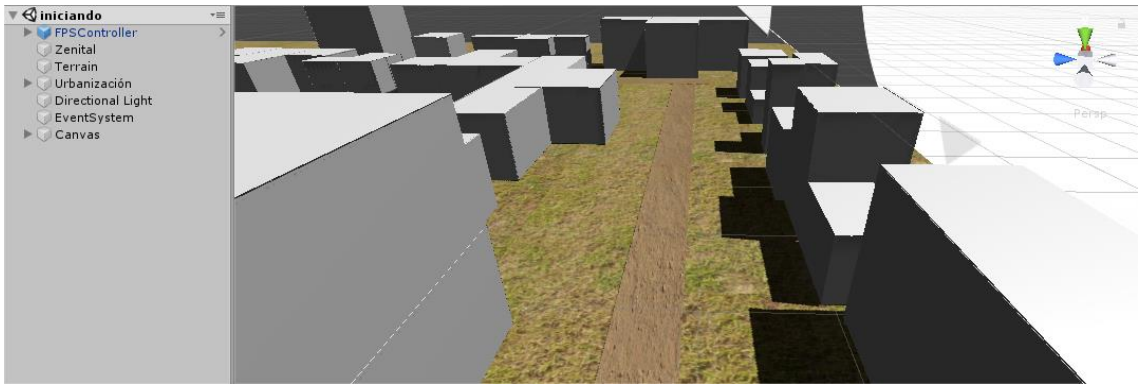

Ilustración 9: Calles de prueba de Unity [Autor]

Otro aspecto positivo de Unity es la compatibilidad con el SDK de Android para la creación de programas en dispositivos compatibles, poniendo a disposición del programador un simulador virtual para probar el programa. De esta manera, se evita la necesidad de realizar las pruebas en un dispositivo físico, y se añade la capacidad de poder disponer de distintas herramientas como la depuración modo ”debug” para simplificar la corrección de errores en la aplicación. Algo negativo que mencionar sobre la instalación de ambas herramientas simultáneamente es que ambas se actualizan de forma independiente, pudiendo dar lugar a incompatibilidades con algunas versiones, teniendo entonces que descargar una versión antigua de Unity o del SDK.

2.3 Lenguajes de programación

Tras elegir Unity3D, había que pensar en qué lenguaje de programación debía utilizar. Mis principales opciones abarcaban usar JavaScript o C#. Ambos lenguajes de programación están bastante respaldados por la comunidad facilitando así encontrar posibles soluciones a los problemas que se puedan presentar.

Habiendo desarrollado durante la carrera con la sintaxis de Java, C++ y C# en la asignatura de Videojuegos, opté por usar C#. Otro motivo de usar ese lenguaje es debido a que la mayoría de los foros y vídeos así lo recomiendan. Además, es más aconsejable cuando la implementación se basa en una aplicación de dispositivos móviles en Unity con el SDK de Android.

Para la conexión de Unity3D con un dispositivo Android comprobé las funcionalidades administradas por el SDK Android, utilizando sus diversas opciones y características con el fin de simplificar algunas de las tareas del proyecto. Por ejemplo, bloquear la orientación de pantalla en horizontal, el uso del modo “debug” para detección de errores y administrar hasta qué punto la aplicación tendrá retrocompatibilidad. Otra característica que nos ofrece es la posibilidad de descargar un simulador virtual para la realización de las diversas pruebas de diseño y corrección de errores del proyecto.

CAPÍTULO 3: PLANIFICACIÓN

3.1 Identificación de las tareas

En este apartado se nombrarán las distintas tareas necesarias para poder completar el proyecto. Como se muestra en la imagen el trabajo dispondrá de cinco versiones Ilustración 10:

Versión	Version Predeterminada	Fecha	Descripción
Plataforma de desarrollo y base de datos (v 0.2)		2019-02-28	aplicación y BBDD conectada a dispositivo Android
Modelado de edificios realizado (v 0.5)		2019-04-05	Texturización de los edificios correctamente implementado
Interfaz de usuario y servidor (v 0.7)		2019-05-03	se implementará una interfaz de usuario simple y conexión con un servidor externo
Proyecto finalizado		2019-05-31	TFG terminado sin errores
versión con posibles errores (v 0.8)		2019-05-31	proyecto terminado y en revisión

Ilustración 10: Versiones del proyecto [Autor]

Cabe mencionar, que las dos últimas versiones son de documentación y revisión de los problemas detectados. En esos casos no habrá iteraciones en la sección de requisitos, ni en el de diseño, aunque se mencionará los problemas encontrados durante la implementación.

A continuación, se describirán las diferentes versiones y tareas que se han de realizar para finalizar el proyecto.

Plataforma de desarrollo y base de datos (v 0.2)

Esta versión consiste en una toma de contacto con las herramientas a utilizar y las conexiones entre ellas. Se subdivide en:

- Estudio sobre las herramientas a utilizar.
- Creación de la Base de datos de texturas.

- Conexión de la herramienta de desarrollador con el dispositivo Android.
- Conexión de la base de datos a la herramienta y al dispositivo móvil.

Modelado de edificios (v 0.5)

En este apartado, se recrearán y prepararán los distintos edificios para añadirles texturas y realizar una preparación para la posibilidad de añadir texturas mipmapping previamente en la BBDD dividida en:

- Creación de una calle virtual.
- Realización de mipmapping de las texturas.
- Preparación los edificios para la adición de texturas.

Interfaz de usuario (v 0.7)

Una interfaz contendrá las distintas ventanas y opciones para añadir imágenes. Consta de los siguientes partes:

- Añadir fotos desde la galería.
- Modelado asistido de texturas.
- Almacenar texturas mediante fotos.

Versión con posibles errores (v 0.8)

En esta sección, se realizará una búsqueda de posibles errores encontrados en el proyecto que se solucionarán. También de los aspectos importantes que se han debido modificar debido a las optimizaciones.

Proyecto finalizado (v1.0)

Como su nombre indica la implementación del proyecto y documentación se ha completado y está lista para entregarse para su posterior defensa.

En la siguiente imagen, podemos observar las distintas versiones y tareas previamente mencionadas de manera más visual Ilustración 11:

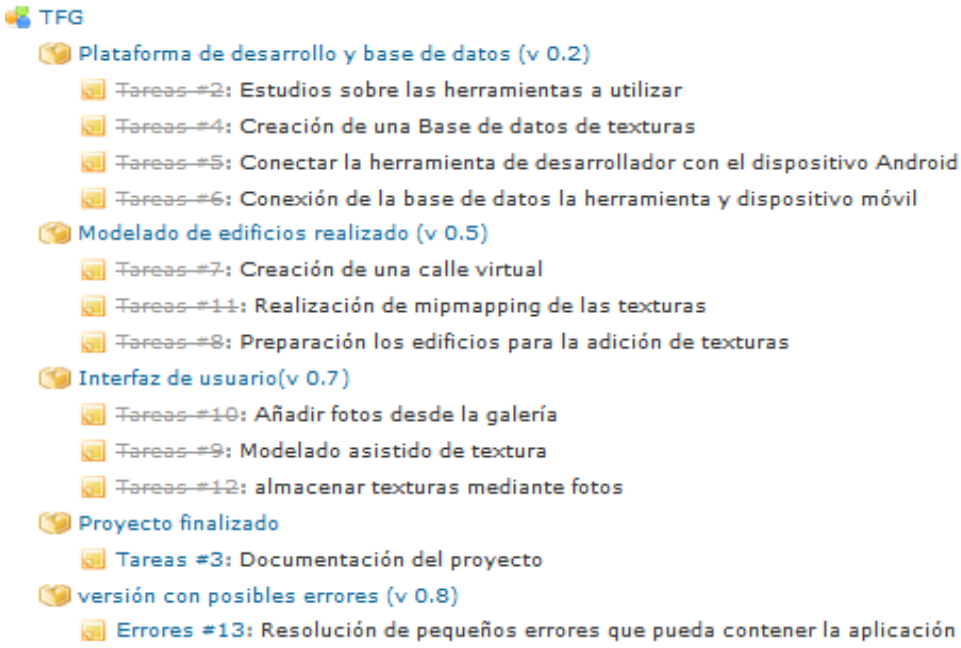

Ilustración 11: Tareas específicas de cada versión [AUTOR]

.

3.2 Planificación temporal de las tareas

Con tal de analizar el tiempo necesario hasta terminar el proyecto se ha optado por usar un diagrama de Gantt. Es una herramienta de planificación de tareas que nos presenta de forma visual el tiempo entre las distintas labores necesarias para terminar el proyecto, quien las debe realizar y las relaciones entre ellas.

En la planificación se ha analizado qué tareas preceden a otras y cuales se pueden hacer de manera concurrente para optimizar el tiempo de realización del proyecto. Esto se puede observar en la imagen en el anexo B (Ilustración 69):

3.3 Herramientas de análisis software

Con el objetivo de una correcta gestión de tiempo y del trabajo de las distintas personas que hacen el proyecto, se ha pensado en usar la herramienta online Redmine. Entre sus características se encuentran la realización de los diagramas de Gantt que muestran las distintas tareas de una manera simple y así comprobar las distintas relaciones entre las tareas, quién las debe realizar y el tipo de trabajador, entre otros. Por último, permite crear diversos informes de los datos que requiera el usuario.

La herramienta utilizada para el control de versiones será SmartGit. Esta es una aplicación para la gestión de los distintos proyectos, capaz de organizar las subidas de los archivos del grupo desarrollador en las distintas etapas del proyecto. Otro elemento importante es que cuenta con un historial de subida por cada parte que ha sido desarrollada, un usuario para ver qué persona ha realizado un apartado del proyecto y cuándo, para incluirlo en el informe. [15]

Con la finalidad de realizar una estimación de los costes se ha optado por usar el Modelo COCOMO para estimar cuánto tiempo tardará en realizarse el proyecto desde la siguiente página web [16].

Para usar este modelo, necesitamos hallar los valores que serán usados, teniendo en cuenta la dificultad, números de líneas, etc. Por lo tanto, la métrica que se usará es del tipo básico y orgánico, ya que tendremos a programadores experimentados en la herramienta. El proyecto contendrá solo unos pocos miles de líneas y se usarán los valores que corresponda de la tabla siguiente Tabla 1.

Estos valores a, b, c y d representan el esfuerzo del proyecto, siendo estimaciones para obtener la información de cuantas personas son necesarias para finalizar el proyecto en la fecha acordada.

Modo de desarrollo	***a***	***b***	***c***	***d***
Orgánico	3.2	1.05	2.5	0.38
Semi-acoplado	3.0	1.12	2.5	0.35
Acoplado	2.8	1.20	2.5	0.32

Tabla 1: Valores COCOMO del modelo básico [AUTOR]

Se usarán las siguientes ecuaciones para saber el esfuerzo del personal y el tiempo de desarrollo estimado Ecuación 1:

$K_m = a * S_k^b$

$T_d = c\ K_m^d$

Dónde:

K_m: Las personas/mes necesarias en el proyecto.

S_k: Líneas de código fuente expresado en miles.

T_d: El tiempo de desarrollo en meses.

Ecuación 1: Ecuación del modelo Orgánico

3.4 Presupuesto

En esta sección analizaremos los distintos recursos que han sido necesarios para la realización del proyecto. Se analizarán los costes software, hardware, coste del personal y otros costes que no corresponden a ninguna de las categorías anteriores.

3.4.1 Costes de recursos humanos

Para empezar, analizaremos el recurso más valioso de una empresa, los empleados encargados en la realización de las diversas tareas del proyecto:

Primera iteración: Plataforma de desarrollo y base de datos Tabla 2.

Tarea	Personal	Tiempo estimado
Estudios sobre las herramientas a utilizar	Jefe de proyecto	4 semanas
Creación de una Base de datos de texturas	Analista y programador	2 semanas
Conexión de la herramienta de desarrollo con el dispositivo Android	Programador	1 semana
Conexión de la base de datos con la herramienta y el dispositivo móvil	Programador	1 semana

Tabla 2: Personal del proyecto de la iteración 1 [AUTOR]

Segunda iteración: Modelado de edificios realizado Tabla 3.

Tarea	Personal	Tiempo estimado
Creación de una calle virtual.	Programador	2 semanas
Realización de mipmapping de las texturas.	Programador	2 semanas
Preparación de los edificios para la adicción de texturas.	Programador	3 semanas

Tabla 3: Personal del proyecto de la iteración 2 [AUTOR]

Tercera iteración: Interfaz de usuario Tabla 4.

Tarea	Personal	Tiempo estimado
Añadir fotos desde la galería	Programador	2 semanas
Modelado asistido de textura	Programador	2 semanas
Almacenar texturas mediante fotos	Programador	3 semanas

Tabla 4: Personal del proyecto de la iteración 3 [AUTOR]

Cuarta iteración: Versión con posibles errores Tabla 5.

Tarea	Personal	Tiempo estimado
Documentación del proyecto	Jefe de proyecto	7 semanas

Tabla 5: Personal del proyecto de la iteración 4 [AUTOR]

Quinta iteración: Proyecto finalizado Tabla 6.

Tarea	Personal	Tiempo estimado
Resolución de pequeños errores que pueda contener la aplicación	Analista y programador	3 semanas

Tabla 6: Personal del proyecto de la iteración 5 [AUTOR]

Por consiguiente, el gasto semanal y de cada año serían Tabla 7:

Trabajador	Nº Semanas	Salario semanal	Salario Anual	Total
Jefe de proyecto	11 semanas	730,3€	35.056€	8.033,3€
Analista	5 semanas	573,4€	27.521€	2.867€
Programador	21 semanas	479€	23.000€	10.059€
			TOTAL	20.959,3€

Tabla 7: Presupuesto total del personal del proyecto [AUTOR]

3.4.2 Costes Hardware

En el caso de los costes de hardware se compondrá de las distintas herramientas usadas durante los cinco meses en la realización del proyecto para el correcto desarrollo de la implementación, corrección y pruebas del proyecto. Por tanto, en este caso los costes de la parte de implementación del proyecto serían Tabla 8:

Concepto	Importe	Coste por mes	Uso en el proyecto	Coste para el proyecto
Asus X550CC Intel(R) Core(TM) i7-3537U	1.100€	91,7€	5 meses	458,5€
Samsung Galaxy J4+	239€	99,6€	5 meses	52€
		TOTAL		510,5 €

Tabla 8: Costes Hardware [AUTOR]

3.4.3 Costes Software

A continuación, se mostrará los costes software derivado de la utilización de las distintas herramientas de nuestros programas para completar el proyecto Tabla 9:

Concepto	Importe	Coste por mes	Uso en el proyecto	Coste para el proyecto
Navegador Mozilla Firefox	0€	--	--	0€
Entorno de Desarrollo Unity 3D	1.825€	152€	5 meses	760 €

Concepto	Importe	Coste por mes	Uso en el proyecto	Coste para el proyecto
Sistema operativo Windows 10 Professional	56,90€	4,7 €	5 meses	23,5€
JDK	0€	--	--	0€
SDK ANDROID	0€	--	--	0€
Licencia de Smartgit	74€	6,2€	5 meses	31€
Gestor de Proyectos Redmine	0€	--	--	0€
		TOTAL		814,5€

Tabla 9: Coste software [AUTOR]

3.4.4 Otros Costes

Otros costes que no se catalogan en los apartados anteriores serían Tabla 10:

Concepto	Coste por mes	Uso en el proyecto	Coste para el proyecto
Fibra de 100M durante los 5 meses del proyecto	20,15€	5 meses	100.75€
Material de oficina	7€	5 meses	35€
Costo de luz del proyecto	65€	5 meses	325€
Total			460,75€

Tabla 10: Otros costes [AUTOR]

3.4.5 Gastos totales

Y teniendo todos los precios anteriores podemos concluir que la fase de desarrollo costará de Tabla 11:

Categoría	Coste
Coste hardware	510,5€
Coste software	814,5€
Coste de personal	20.959,3€
Otros costes	460,75€
Total	29.745.05€

Tabla 11: Costes totales del proyecto [AUTOR]

CAPÍTULO 4: ANÁLISIS DEL SISTEMA

4.1 Estudio del contexto y alcance

Tras haber hecho un estudio de mercado no se han encontrado herramientas con características parecidas que ayuden a los usuarios en texturizar los edificios, mediante modelos 3D virtuales.

La aplicación más parecida que podemos destacar es Google Maps por su utilidad para guiar a los usuarios a su destino, pero con una calidad de imagen no tan buena como se desearía. Además, tiene problemas de oclusión por culpa de los árboles en el caso de que se quiere ver los distintos edificios y la actualización de una zona tarda su tiempo por tener que realizarse en coche con una cámara.

Con esa aplicación en mente, el proyecto se centrará en la obtención de texturas mediante la realización de fotos o imágenes que luego el usuario podrá retocar con algunas herramientas externa y así poder texturizar los diferentes edificios del entorno virtual. Evitando así, que ocurra dicha oclusión. Tras este planteamiento es necesario identificar qué elementos formarán parte de la aplicación:

- Por una parte, tendremos a los usuarios que están interesados en usar esta aplicación. El usuario podría no tener ninguna experiencia en el uso de programas de este estilo, por lo cual sería necesario conseguir que sea entendible y fácil de usar para que a cualquier usuario, sin importar experiencias previas, le resulte intuitiva de manejar con solo un simple vistazo.

- Por otra parte, y para finalizar, tenemos que representar todos los mecanismos y requerimientos necesarios para obtener todas las

funcionalidades. También se incluye en esta parte las distintas interfaces que muestran información al usuario de una manera gráfica.

- Esta aplicación móvil ha sido desarrollada en toda su totalidad mediante C# con el SDK de Android con el uso de Unity3D para así aprovechar todas las ventajas de accesibilidad y portabilidad de dicho lenguaje tal y como se explicó en el punto 2.3 del proyecto.

Durante todas las etapas de este proyecto se ha seguido una metodología Scrum. Esta facilita distintos incrementos del proyecto con unas específicas mecánicas implementadas para cada una, las cuales son revisadas y evaluadas para una progresión adecuada del proyecto. Cada prototipo contiene el incremento anterior. En esta división se ha realizado en 5 partes cada una con las distintas fases de análisis diseño e implementación Tabla 12:

	Nombre	Duración	Fecha Inicio	Fecha Fin	Versión
Iteración 1	Plataforma de desarrollo y base de datos	1 mes	1 de febrero de 2019	28 de febrero de 2019	Versión 0.2
Iteración 2	Modelado de edificios realizado	1 mes y 5 días	1 de marzo de 2019	5 de abril de 2019	Versión 0.5
Iteración 3	Interfaz de usuario	1 mes	6 de abril de 2019	4 de mayo de 2019	Versión 0.7
Iteración 4	Versión con posibles errores	1 mes	5 de mayo de 2019	31 de mayo de 2019	Versión 0.8
Iteración 5	Proyecto finalizado	4 meses	1 de febrero de 2019	31 de mayo de 2019	Versión 1.0

Tabla 12: Iteraciones del proyecto [AUTOR]

4.2 Análisis de requisitos

La fase de requisitos es aquella que define los pasos para afrontar un problema, las distintas mecánicas que tendrá el proyecto, así como sus restricciones. Esta es una etapa primordial, puesto que dará paso a la codificación de la aplicación.

Para analizar un requisito tenemos que definirlo y según IEEE:

1- *Una condición o capacidad requerida por un usuario para resolver un problema o alcanzar un objetivo.*

2- *Una condición o capacidad que debe cumplir o poseer un sistema o componente de sistema para satisfacer un contrato, estándar, especificación, o cualquier otro documento impuesto formalmente.*

Existiendo dos tipos de requisitos:

1. Requisitos funcionales: Especifican qué servicios o comportamientos debe tener la aplicación para satisfacer al usuario.

2. Requisitos no funcionales: Observa las distintas restricciones para el sistema a la hora de realizar los requisitos.

En los siguientes párrafos se mostrarán los distintos comportamientos que deben tener la aplicación de los dos tipos de requisitos previamente nombrados.

4.2.1 Requisitos funcionales

- Entorno virtual: Creación de un entorno virtual con el que el usuario interactuará.

- Menú Textura: Menú encargado de mostrar al usuario las distintas texturas disponibles para colocarlas sobre el edificio elegido.

- Elegir edificio: Mediante dos toques a la pared de un edificio cualquiera, poder cambiar su textura.

- Submenú de texturas: Encargado de dar distintos tamaños a la textura previamente elegida.

- Menú de añadir Texturas: Compuesto de un menú con la capacidad de añadir texturas de dos maneras diferentes.

- Añadir mediante foto: Con la realización de una foto se inserta directamente al menú de textura.

- Elegir textura mediante galería: Se añaden las distintas texturas usando una carpeta especifica de la galería creada por el programa.

4.2.2 Requisitos no funcionales

- En un principio se ha de pensar en qué tipo de plataforma se debería desarrollar la aplicación y con qué herramienta se obtendría mejores resultados.

- La apariencia de la aplicación debe ser adaptable para los distintos tamaños de los móviles.

- El tiempo de respuesta de la aplicación debe ser aceptable, aun conteniendo una gran cantidad de texturas.

- La aplicación debe ser sencilla e intuitiva para toda clase de usuario, independientemente de la edad o problema que tenga el usuario, para conseguir usar todo el potencial de la aplicación.

- Las imágenes admitidas por la base de datos han de ser “png” o “jpg”. Tras procesar, será necesario hacer un encogimiento de la misma a la potencia de dos más cercana facilitando así cálculos posteriores.

4.3 Análisis de los elementos del proyecto

Gestión de la Base de Datos

Para esta primera versión del proyecto, se trata de investigar las herramientas que se utilizarán en el proyecto y como conectarla entre ellas. Inicialmente buscamos una herramienta para realizar un modelado 3D que permita usar una base de datos para la optimización de las búsquedas de las texturas y edificios (que contengan texturas en ellas). Planteándose usar mySQL o SQLite para dicha implementación en el proyecto, tras la búsqueda de información se encontró que había una BBDD SQLite para dispositivos Android que se puede usar en Unity. Como conclusión de esta sección sobre la base de datos se pensó en los distintos parámetros y tablas para la realización de este proyecto.

Se necesitará la creación de una base de datos que contenga las distintas texturas. Esta base de datos debe tener los siguientes parámetros:

- Id: Es el identificador de la imagen, además, es un entero, clave primaria y será usado para indicar la posición de la Textura en la implementación, es un valor auto incrementable.

- Nombre: Es el nombre de la foto almacenado como cadena de caracteres, este campo será usado para la realización de pruebas y comprobar que se obtiene la foto requerida.

- Dimensiones de la foto: Es el ancho y alto de la foto guardados como enteros, estos dos parámetros son útiles cuando se haga el proceso de mipmapping explicado detenidamente en el documento.

- La imagen: Es la textura que ha sido almacenada.

- Nº de veces que se ha realizado mipmapping: Este parámetro significa el número de veces que una imagen introducida en la base de datos se ha encogido, la imagen se va encogiendo en base a potencia de dos.

Además, de otra tabla, definida como pared, la cual es necesaria para saber que muros han sido texturizados y cuál es la imagen que se debe colocar con objeto de poder realizar el proceso de cargado del entorno virtual:

- Ruta de la pared: Es una cadena de caracteres que identifica la pared del edificio que ha sido modificado.

- La ruta de la imagen: Otra cadena de caracteres que indica la posición de la textura que se ha utilizado en dicho edificio.

Por último, se usará otra tabla que nos ayudará en el nombrar las diversas texturas que se nos almacene en las tablas y contiene:

- Id: numeración para nombrar las diversas fotos, para el caso de que coincida el nombre.

Quedándose así la organización de las distintas tablas realizadas en esta iteración Ilustración 12.

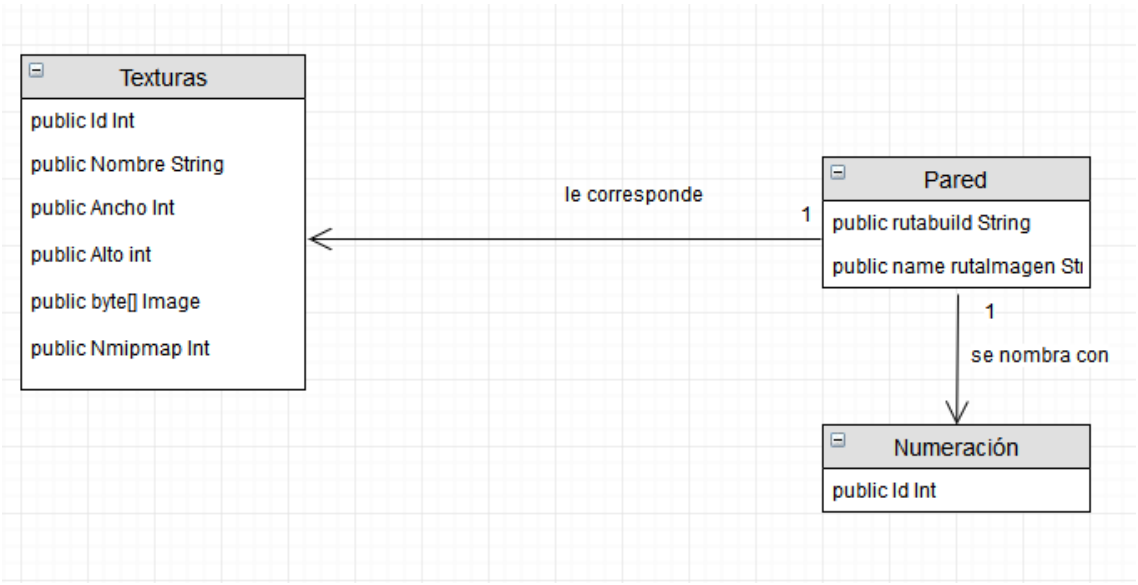

Ilustración 12: Esquema de BBDD [AUTOR]

Paso de elementos entre Unity y Android

Otra cuestión que era importante en el proyecto es el cómo pasar archivos de Unity al dispositivo Android ya que entre ellos tienen distintas rutas como diversas maneras de guardar archivos. Después de una búsqueda exhaustiva se encontró que se puede crear una carpeta llamada StreamingAssets o Resources que permite dicho traspaso. Pero para facilitar la búsqueda se planteó usar StreamingAssets ya que existe un Script llamado BetterStreamingAssets que ofrece una búsqueda de archivos más sencillos, estando disponible desde la tienda de Unity o en el git [15].

Además, se busca la manera de conectar las diferentes herramientas antes mencionadas consiguiendo los requerimientos del proyecto. Para acabar esta iteración, hay que investigar maneras de depurar la aplicación cuando se realicen las pruebas en un dispositivo móvil usando el SDK de Android.

4.4 Preparación del entorno virtual y texturas

Texturas y mipmapping

En el caso de la realización de mipmapping, tenemos que tener en cuenta las distintas dimensiones que puedan hacer las distintas cámaras de los móviles.

Después hay que buscar la potencia de dos de esas dimensiones para facilitar los posteriores cálculos en la implementación de varios métodos, también hay que pensar en qué punto se considera dejar de comprimir la foto. Se ha considerado que una textura de 64 pixeles de ancho o alto ya es lo suficiente pequeña.

Como se puede comprobar en las imágenes Ilustración 13 e Ilustración 14, se pierde mucha calidad cuando llega a 64 píxeles.

Ilustración 13: Textura de alta calidad [AUTOR]

Ilustración 14: Textura de baja calidad [AUTOR]

Recreación del mundo virtual

Para la creación de la calle virtual se ha usado un diseño de varias calles con tal de dar un entorno virtual al usuario, para a continuación, poder llenarlos de edificios con el objetivo de que el usuario pueda interactuar.

Tras obtener todo el entorno creado con todos los edificios, se comprobó que en un primer momento las texturas que se añadían al edificio se situaban en todas las caras del edificio. Para solucionar este problema, se ha optado por crear los edificios mediante paredes separadas y debiéndose colocar de forma que la textura salga correctamente en todas las caras. Además, con la adicción de la componente RawImage aportada por Unity, servirá para colocar las diferentes texturas que contenga el proyecto.

Para ayudar a la implementación, Unity asiste al desarrollador con la utilización de etiquetas que se le pueden poner al objeto, en pos de realizar una búsqueda rápida de una pared. La etiqueta que se ha puesto para las distintas paredes se ha llamado “pared”.

Otra herramienta para optimizar la búsqueda es el “layer” o capa, que he llamado tocable con la finalidad que sea sencillo de entender, se explicará en el punto 5.3 del proyecto con más detalle. Como se ven en las siguientes imágenes Ilustración 15 e Ilustración 16.

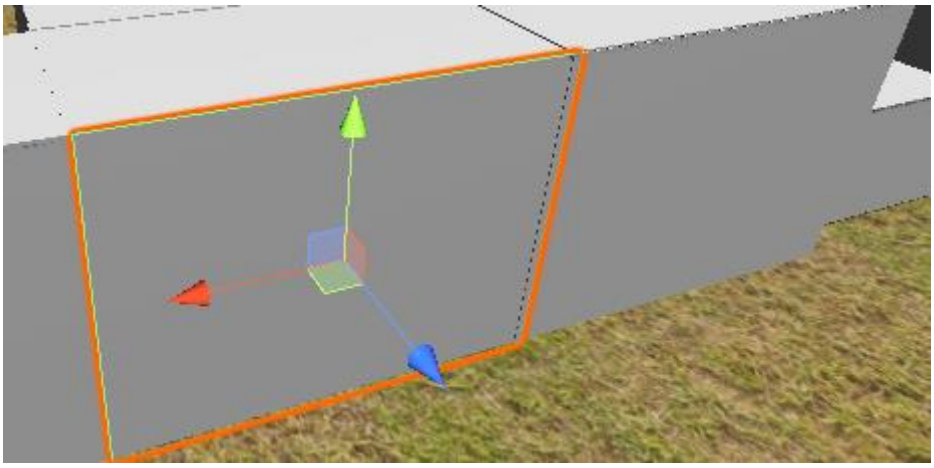

Ilustración 15: Componentes de una pared (1) [AUTOR]

Ilustración 16: Componentes de una pared (2) [AUTOR]

Por último, lo que he considerado para esta iteración es ofrecer al usuario una vista cenital con tal de poder ver todo el mundo, facilitándole que de un solo vistazo pueda observar qué edificios han sido texturizados poniendo el techo de color amarillo (siendo su etiqueta “techo”). Y siendo el punto rojo la posición del usuario Ilustración 17.

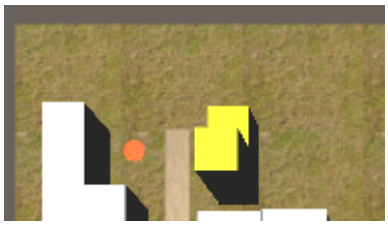

Ilustración 17: Posición del usuario y comprobación de edificios texturizados [AUTOR]

Con estos elementos, ya se tendría los edificios preparados para colocar las texturas en los edificios.

4.5 Adición de las texturas

Texturización

Tras tener los edificios preparados y las texturas iniciales listas, en esta tercera iteración se debe gestionar el proceso de colocar las distintas imágenes en las paredes de los edificios. Dando dos toques en una pared del edificio, esta se pondrá en amarillo indicando al usuario que se ha seleccionado correctamente y con ello se podrá añadir la textura que el usuario elija desde el menú. Solo cambiando la componente de textura del RawImagen de la pared Ilustración 18 como se mencionó en la iteración anterior.

Ilustración 18: Prueba de la texturización de una pared [AUTOR]

Interfaz de los menús

Otro elemento importante a implementar es la opción de guardar el proceso realizado en el entorno virtual desde el menú de opciones. Permitiendo guardar las paredes que han sido texturizadas en nuestra base de datos cuando el usuario lo requiera Ilustración 19.

Ilustración 19: Menú de opciones [AUTOR]

Para acabar con la implementación del proyecto se ha pensado en obtener fotos previamente almacenadas por el usuario con la intención de usarlas en la aplicación. Se podrán almacenar las fotos en una carpeta creada en la galería llamada “Imagen”. Al usuario se le avisa mediante un mensaje de texto cuando se crea la carpeta con éxito o se han añadido correctamente las distintas texturas.

En esta foto se muestra el caso de que la carpeta no estaba creada, por tanto, se crea Ilustración 20.

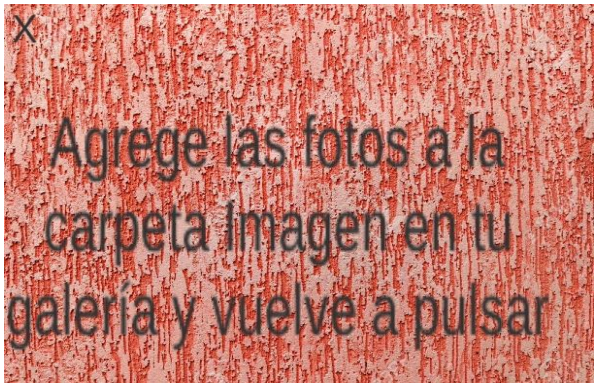

Ilustración 20: Creación de la carpeta Imagen en la galería [AUTOR]

Aquí se muestra cómo se ve en el álbum la nueva carpeta creada en la galería Ilustración 21:

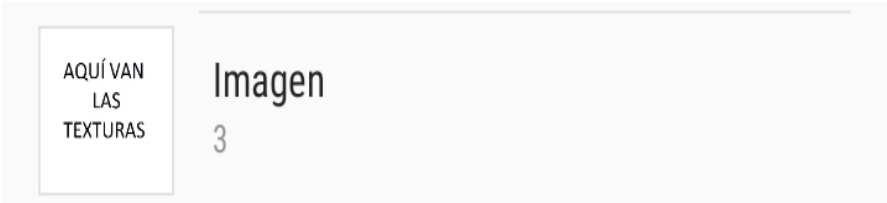

Ilustración 21: Carpeta Imagen creada [AUTOR]

En este último caso se accede a las distintas fotos que se introducirán a la base de datos Ilustración 22:

Ilustración 22: Se añadió las texturas [AUTOR]

Otro modo más sencillo y directo es la de sacar una foto con la cámara consiguiendo almacenar las texturas directamente al menú Ilustración 23.

Ilustración 23: Menú para la adicción de texturas [AUTOR]

CAPÍTULO 5: DISEÑO DEL SISTEMA

5.1 Diseño de los datos

5.1.1 Gestión de los datos en memoria

Como en cualquier proyecto es necesario guardar los diversos elementos en una base de datos, con la intención de conseguir poder acceder a dichos elementos cuando se requiera.

Se optó por usar una base de datos SQL para realizar las búsquedas de las texturas y de las distintas paredes de los edificios que han sido almacenados. Se ha tratado de guardar de manera óptima los datos. Un problema que no se pudo solucionar fue guardar la imagen de forma que sea compatible directamente con Unity, teniendo que transformar la imagen cada vez que necesite el usuario utilizarla.

5.2 Diseño de la interfaz de usuario

Para la correcta realización de la interfaz, se pensó en los posibles consumidores a los que les será útil esta aplicación. La utilización principal estaría enfocada a los emprendedores que quieran saber cómo quedaría su propio negocio. Otro uso que se le puede dar, sería modelar una ciudad virtual obteniendo un boceto de cómo quedaría.

Es necesario pensar en posibles perfiles, edad, problemas físicos, visuales, culturales. Con el fin de facilitar una herramienta accesible y sencilla de emplear por todas las personas sin que haya elementos que puedan resultar ofensivos para alguien.

Se pensó en buscar una manera de que el usuario, con una interfaz intuitiva y simple y mediante unos pocos toques obtenga resultados positivos. Además, se utilizarán elementos visuales para ayudar a simplificar el aprendizaje de esta aplicación.

La implementación de la interfaz se empezó a crear tras tener los principales elementos que debe contener el mundo que son el terreno y los edificios. Ambos necesarios para que haya una base en el proyecto con la que empezar a hacer la interfaz, esta debe contener los distintos menús que guiarán al usuario y poder alcanzar sus objetivos de modelado.

Y para mostrar lo intuitiva que es la interfaz obtenida se mostrará distintas imágenes mostrando una breve explicación de los elementos:

En esta primera imagen se muestra cómo es el entorno virtual creado. Se le da pistas al usuario para que vaya probando controles de forma intuitiva, como es ver las palancas en las esquinas inferiores derecha e izquierda que por el uso de otras aplicaciones entenderá que son de movimiento y de la cámara. Con ayuda de los iconos y los distintos menús el usuario sabrá el uso de cada uno de los elementos sin tener que probarlos. No obstante, el elemento menos intuitivo de la aplicación, que es la de tocar dos veces un edificio se ha puesto de manera directa con texto como aparece en la (Ilustración 70) y se explicará mejor en la sección 5.2.

A continuación, se observará el mismo mundo tras pulsar el botón de la cámara, pero con una vista cenital. Como se puede ver en la imagen el foco de luz roja es la posición donde está el usuario y en esta vista tocando la pantalla hace que el usuario se mueva a dicha posición. Otra característica que ofrece esta vista es saber qué edificios contienen texturas dando a conocer que se ha texturizado y que no, como se muestra en la siguiente Ilustración 24:

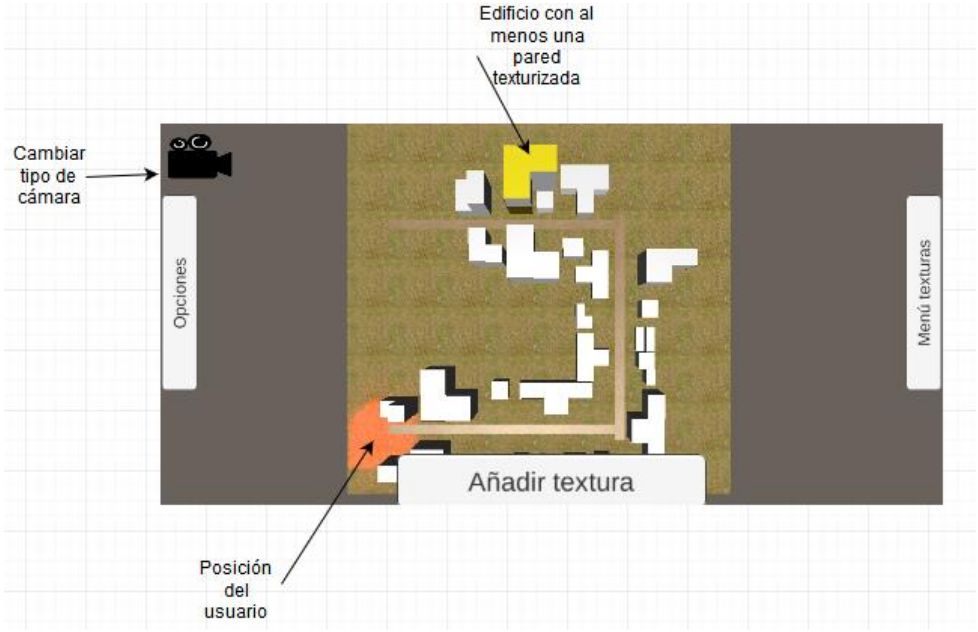

Ilustración 24: Interfaz de la vista cenital del entorno virtual [AUTOR]

A continuación, se enseñará el menú con las distintas texturas Ilustración 25:

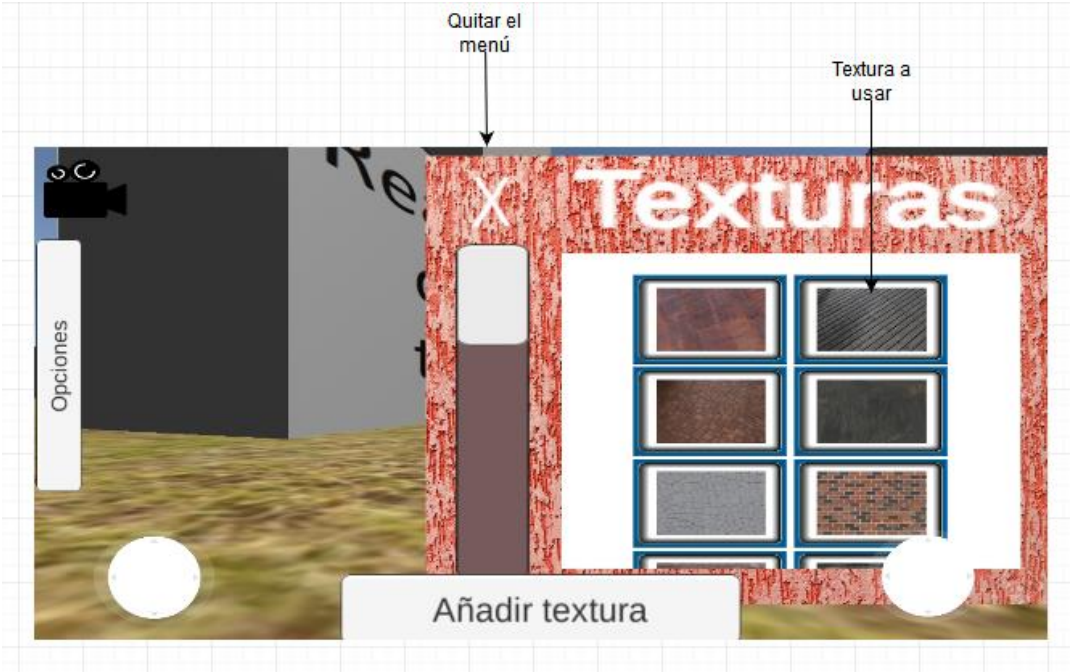

Ilustración 25: Interfaz del menú de texturas [AUTOR]

En el menú tras elegir una textura se obtiene la misma, pero esta tiene diferentes tamaños para que el usuario elija el que más le conviene y usarla

simplemente tocando la imagen cuando tenga un edificio este seleccionado Ilustración 26:

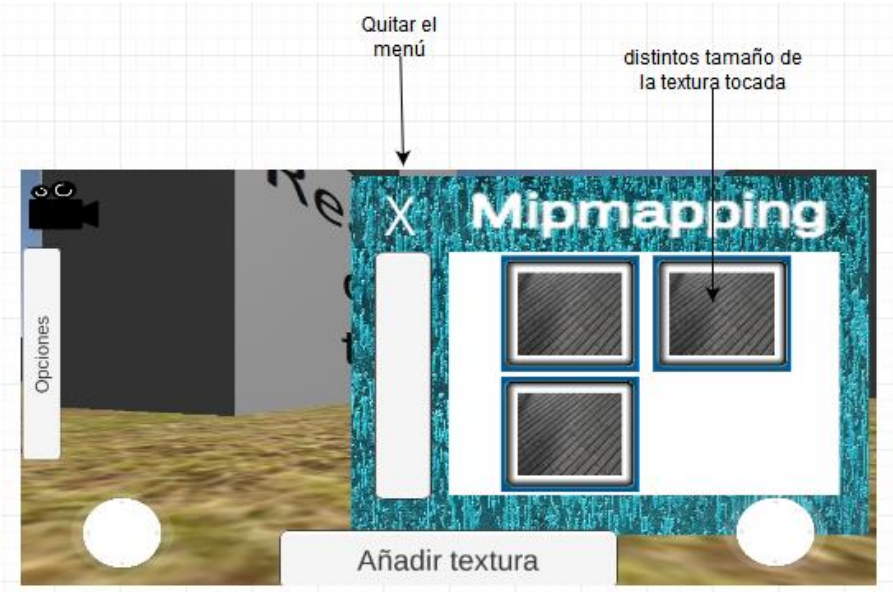

Ilustración 26: Interfaz del menú Mipmapping [AUTOR]

El siguiente menú muestra la obtención de las imágenes. Como se puede observar, se puede hacer de dos maneras: mediante la realización de una foto o desde la galería como se mencionó en el apartado de análisis Ilustración 27.

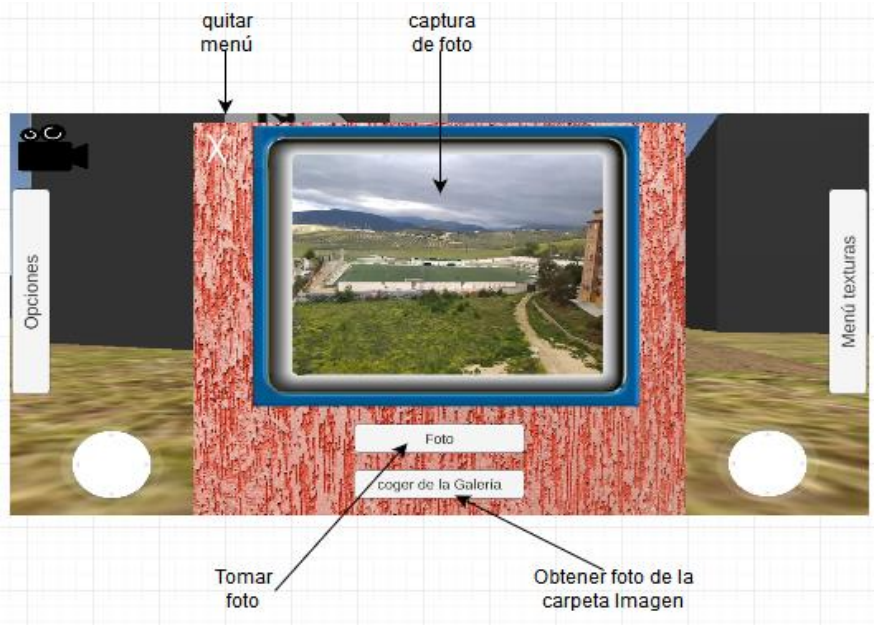

Ilustración 27: Interfaz menú para añadir textura [AUTOR]

Por último, se mostrará la interfaz del menú de opciones. Este contendrá dos botones: uno que guarda el estado del proyecto (las texturas que tiene cada edificio) y otro que las carga Ilustración 28:

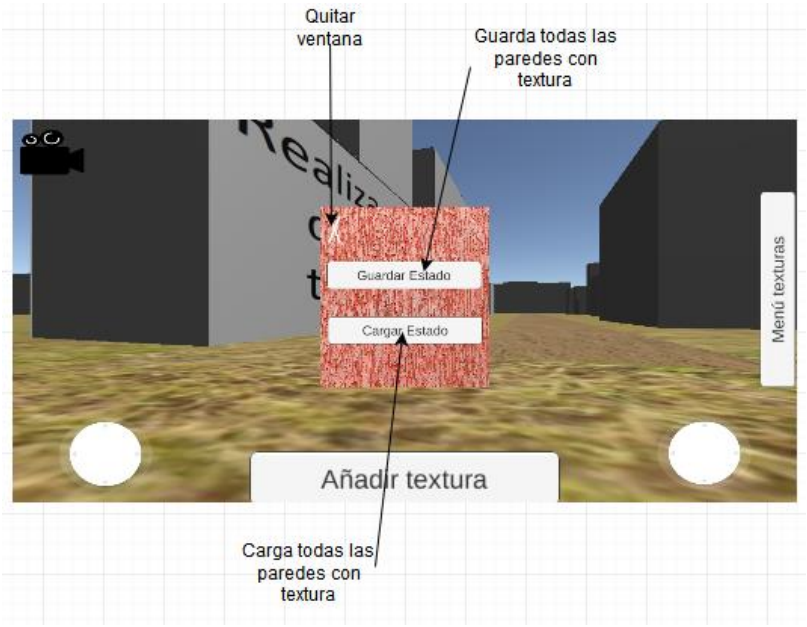

Ilustración 28: Interfaz del menú de guardado y carga del proyecto [AUTOR]

5.2.1 Guía de estilos: Usabilidad

Unos de los factores más importantes a tener en cuenta es el de la usabilidad de la aplicación, nos referimos a la facilidad con la que los usuarios podrán alcanzar un objetivo concreto.

La interfaz está pensada para que el usuario sepa desde un primer momento todas las propiedades que tiene a aplicación solo con iniciarla. Siendo así fácil de manejar y accesible para los usuarios. Constará de tres menús diferentes: un apartado para añadir texturas vía foto o galería, guardar/cargar estado del mundo y por último y más importante un menú para colocar las texturas a las distintas paredes de los edificios. Se da el suficiente espacio para que resulte sencillo pulsar sobre la pared de un edificio y para ver bien la textura cuando el usuario la ponga. En cuanto al texto se usará color blanco para que sea fácil de leer.

5.2.2 Esquemas

En este apartado se mostrarán los distintos esquemas lógicos relacionados con la parte del análisis de requisitos previamente realizados, tras hablar con la tutora sobre el proyecto y que elementos debería contener para su correcto funcionamiento.

A continuación se mostrará un UML y varios diagramas de secuencia para mostrar los diversos pasos en los que se han realizado los procesos para la realización de este proyecto Ilustración 29 e Ilustración 30.

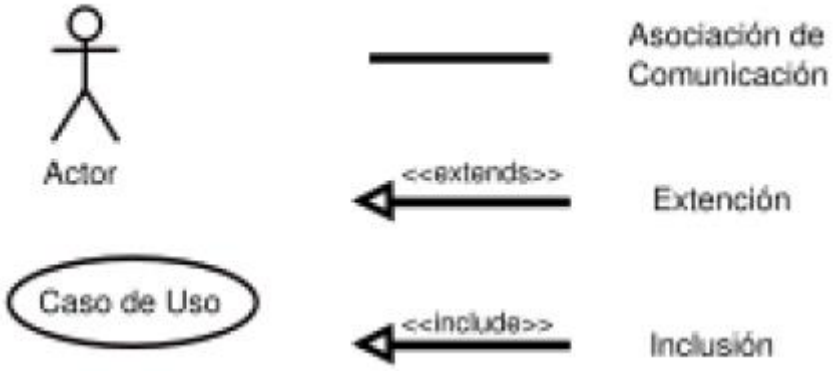

Ilustración 29: Leyenda del UML [AUTOR]

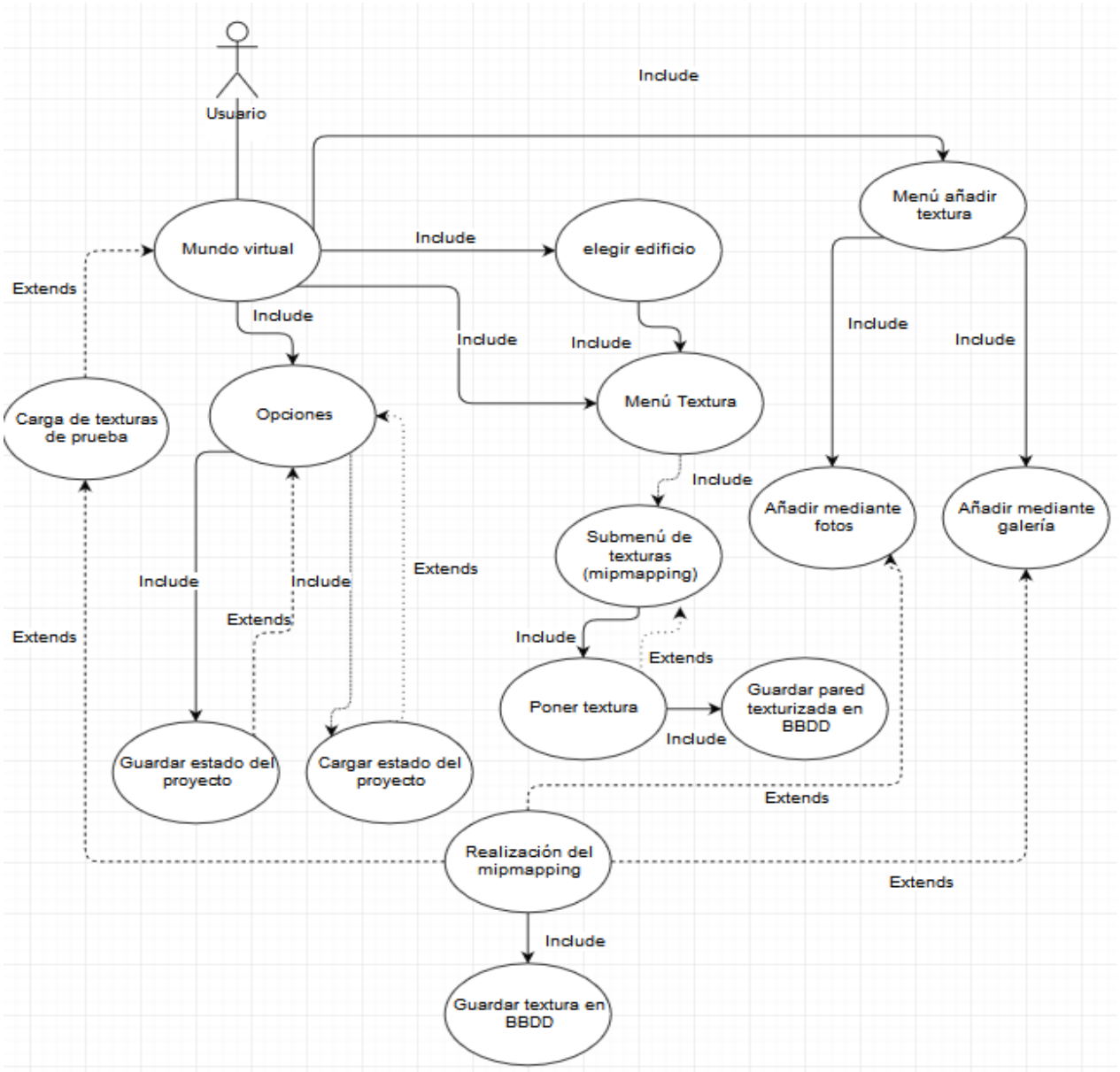

Ilustración 30: UML del proyecto [AUTOR]

Para la realización del diseño se ha optado por hacer unos diagramas de secuencia en los que se podrá ver todas las interacciones y el tiempo de vida del programa, cómo se ejecutarían los distintos comandos en cada una las iteraciones realizadas por el usuario.

El primer diagrama de secuencia explicará de forma gráfica como se puede añadir una nueva textura y como se almacena en la base de datos. En la imagen se observan diversas maneras de obtener las texturas. En un caso se puede tomar una foto. En el otro caso se necesitaría guardar la imagen en la galería. Pueden ocurrir

tres cosas, que no esté creado la carpeta para meter las texturas y se crea, el segundo caso es cuando no se han añadido nuevas texturas y el último caso se añaden las diferentes texturas Ilustración 31.

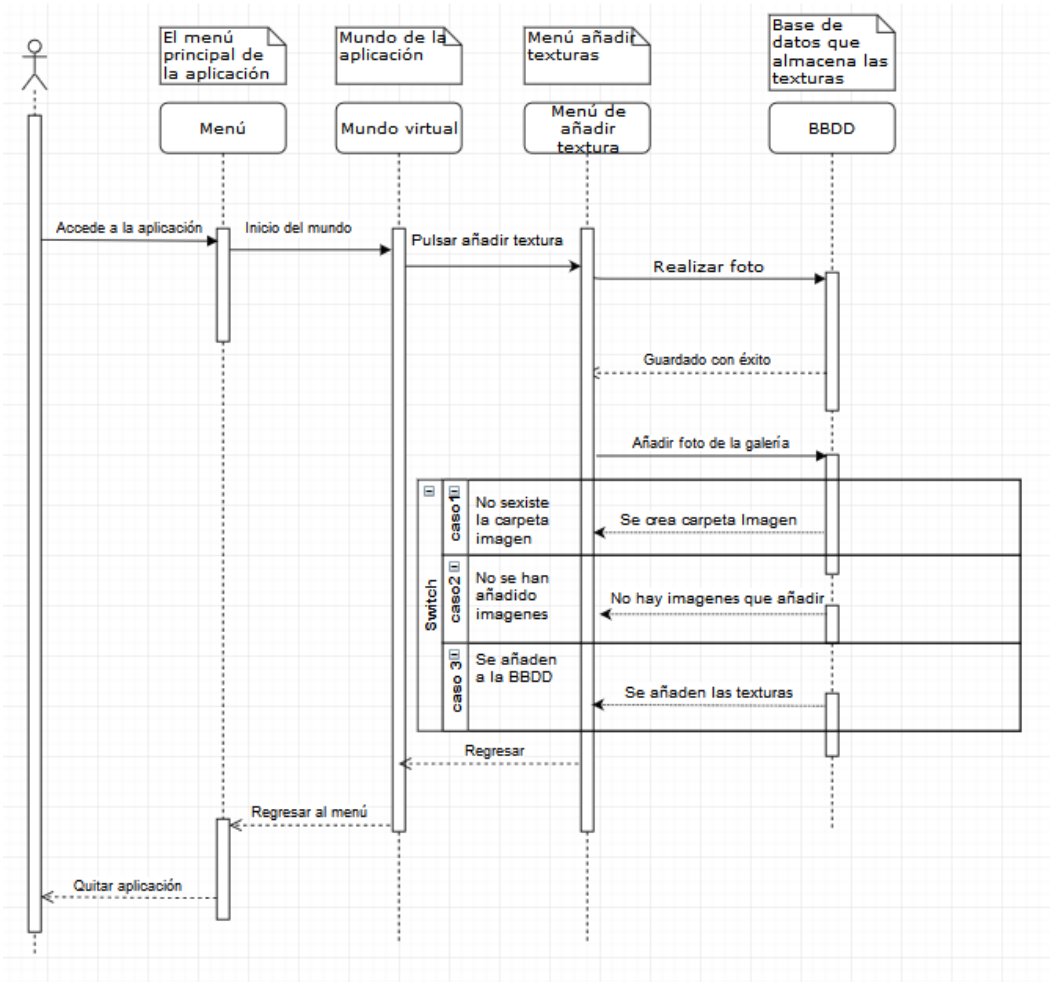

Ilustración 31: Diagrama de secuencias (añadir texturas) [AUTOR]

A continuación, se visualiza la manera de guardar y recuperar el estado del proyecto. Tras pulsar el menú de opciones se puede pulsar los botones guardar o cargar el proyecto Ilustración 32.

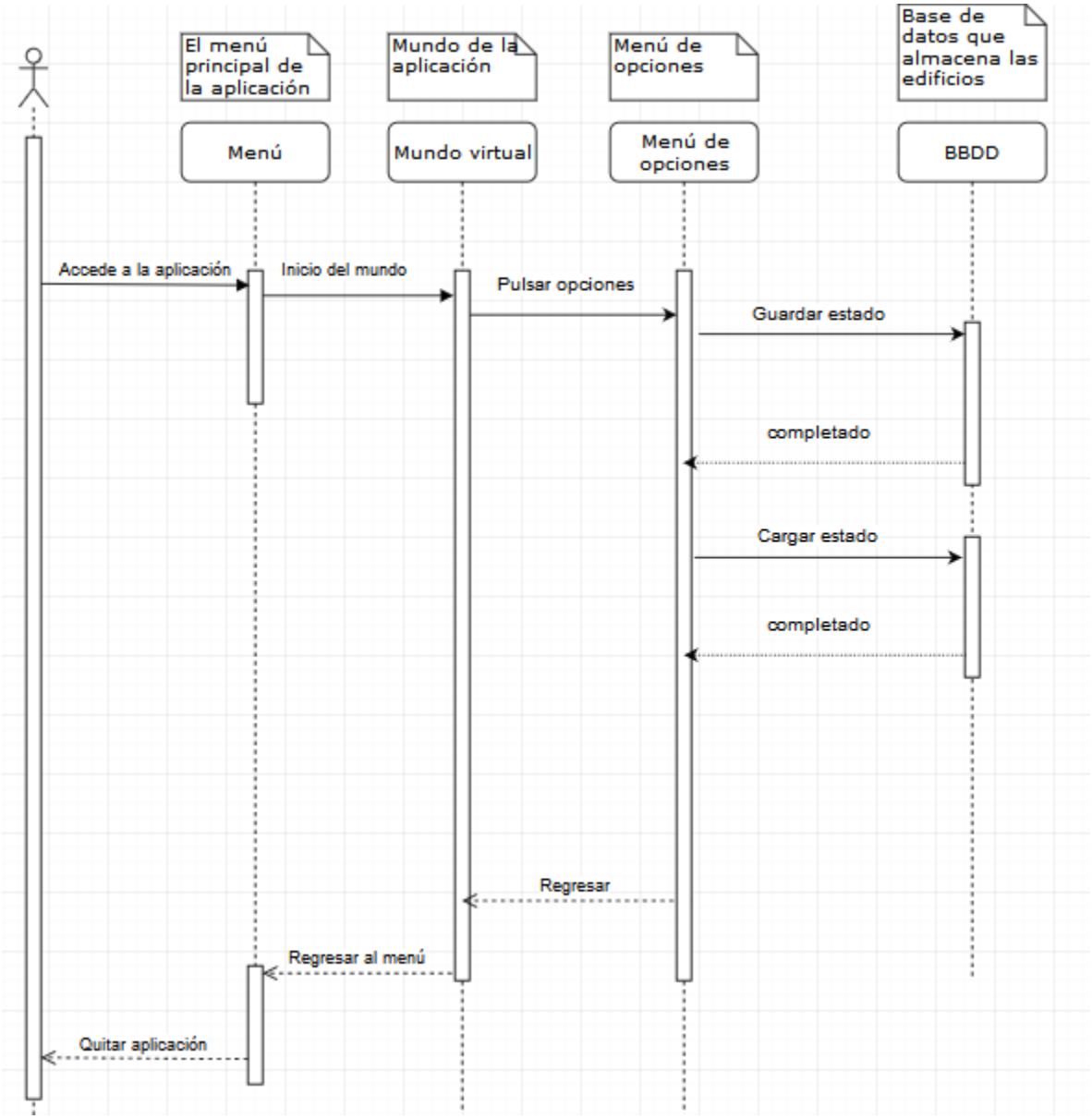

Ilustración 32: Diagrama de secuencia (guardado/carga del estado de los edificios) [AUTOR]

Por último, se muestra como se añade una textura a alguna de las paredes de este proyecto. Tras darle al menú textura se podrá elegir cualquier textura para poder ver sus diferentes tamaños y elegir una de ellas para colocarla en un muro Ilustración 33.

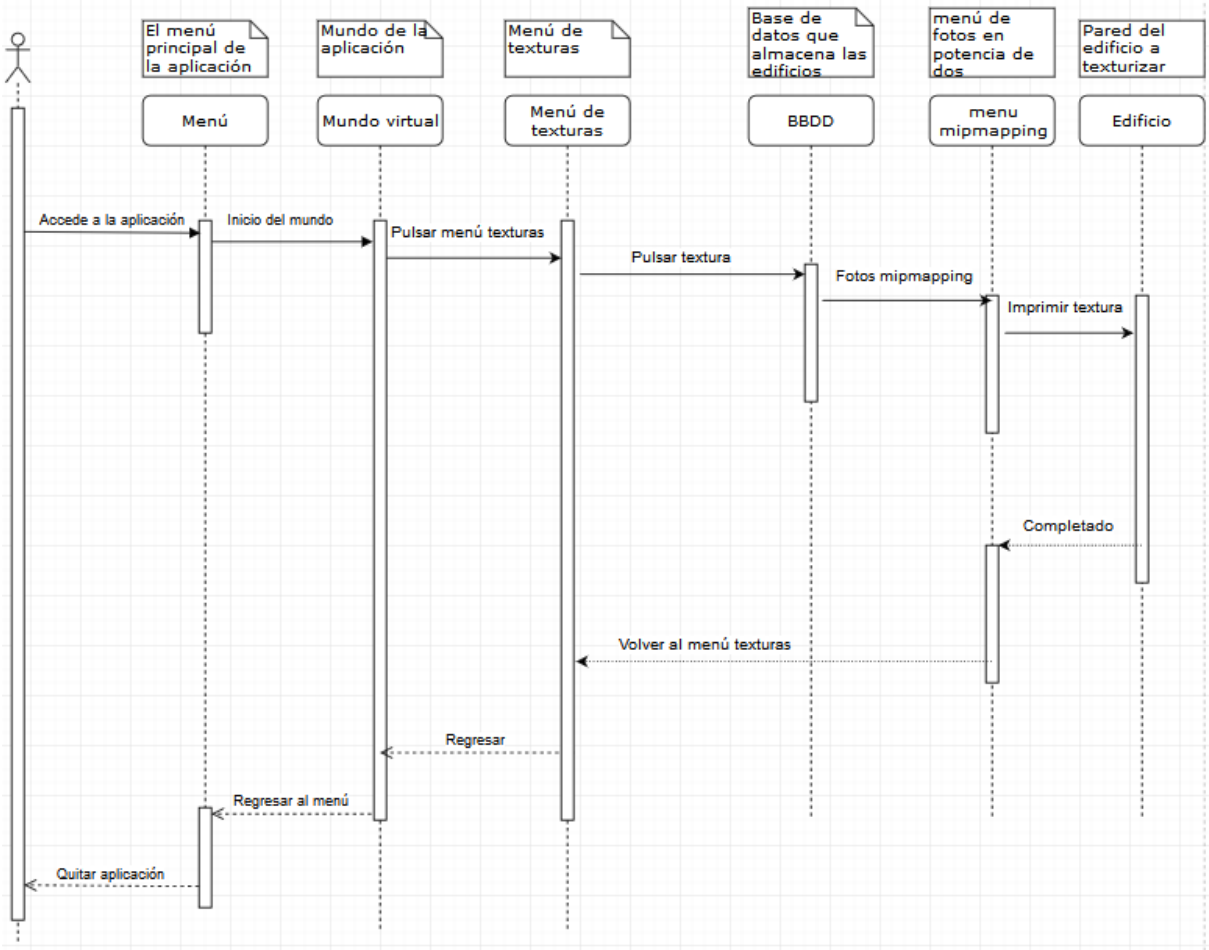

Ilustración 33: Diagrama de secuencia (poner textura a una pared) [AUTOR]

5.3 Preparación del entorno virtual y de las texturas

Entorno virtual

Tras tener las herramientas listas, es necesario preparar una calle de prueba, con la cual se podría empezar a texturizar los primeros edificios con las imágenes de base en el proyecto para la realización de pruebas.

Para la construcción de varias calles se ha usado el editor de Unity con el objetivo de tener un entorno virtual de ejemplo de forma rápida y simple para la realización de varias pruebas, quedando como resultado la Ilustración 34 e Ilustración 35:

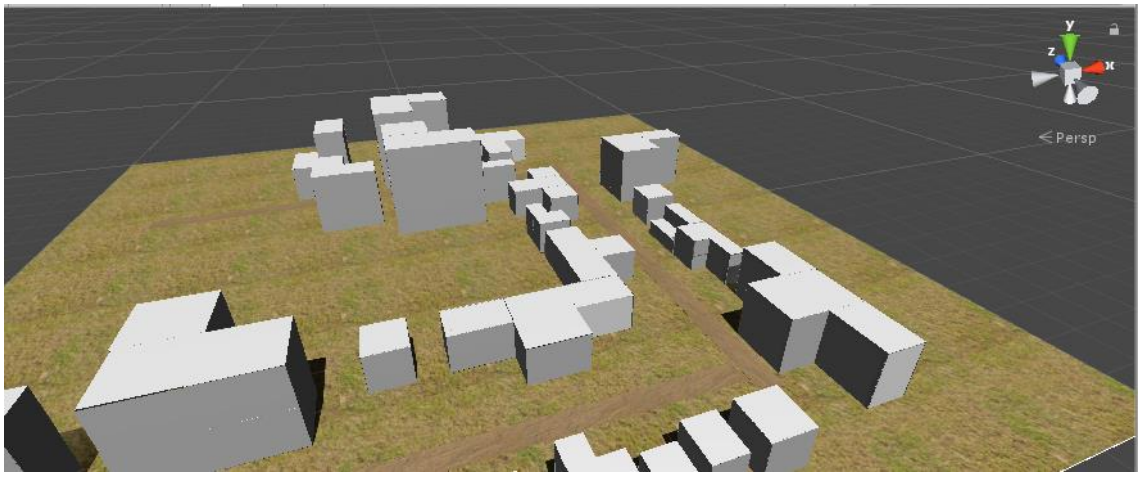

Ilustración 34: Vista aérea del entorno virtual [AUTOR]

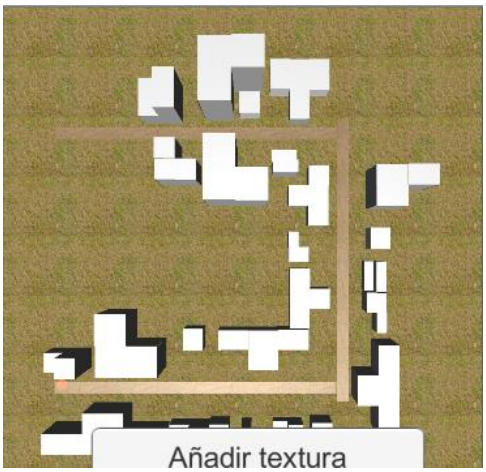

Ilustración 35: Vista cenital del entorno virtual [AUTOR]

Para añadir texturas se ha optado por colocar una componente de Unity llamada “RawImagen” en las paredes del edificio, siendo más fácil añadir las texturas mediante Script buscando dicho elemento. Quitando pasos intermedios, como al utilizar otra componente llamada “Image” que se debe transformar el Texture2D en un “Sprite”. Otra optimización para la búsqueda de los edificios es utilizar una capa (o “layer” en inglés), la cual Unity buscará solo los elementos de dicha capa e ignorará otros. Consiguiendo así ahorrar tiempo de búsqueda y además se mejora con el uso de la etiqueta (o “tag”) para que al tocar un edificio dos veces realice la búsqueda mediante la “layer” y “tag”.

En la siguiente imagen se puede comprobar el diseño de cómo seleccionar un edificio, gracias a los elementos previamente mencionados Ilustración 36:

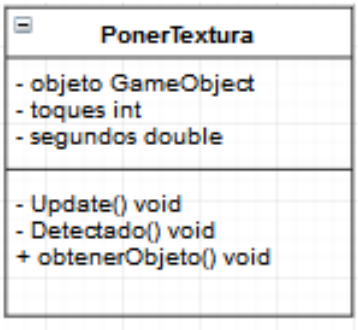

Ilustración 36: Diseño poner Textura

Vista cenital

También se añadió con la creación del mundo una vista cenital, la cual indica donde está el usuario en el mapa mediante una luz de color rojo. En esta vista el usuario se puede trasportar de sitio al usuario mediante un toque en la pantalla, cambiando así la posición del usuario. También se puede comprobar qué edificios han sido texturizados, cambiando la componente de los edificios a un tono amarillento cuando contengan al menos una textura. Como se muestra en la (Ilustración 24: Interfaz de la vista cenital del entorno virtual [AUTOR]) y en la siguiente Ilustración 37 mostrará sus distintos elementos:

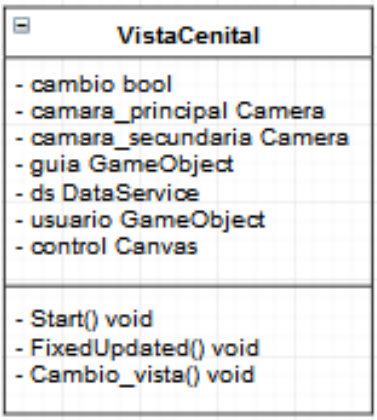

Ilustración 37: Diseño vista cenital [AUTOR]

Mipmapping

Empezando con la creación de las texturas, primero debemos crear la BBDD con el objetivo de guardar las texturas y edificios. Para ello se va a utilizar una base de datos SQLite por su sencillez en la conexión con Unity, además se añadirán texturas de prueba.

Para acabar esta iteración, lo más difícil era la implementación del mipmapping de las texturas. Se optó por guardar todas las texturas en una cadena, sabiendo que la textura original contiene un número de versiones de esa misma (reducida con el mipmapping) en potencia de dos. En primer lugar, encontramos la

primera potencia de dos más cercana por debajo y luego la encogemos hasta llegar a tener 64 pixeles de ancho o de alto. En consecuencia, sabiendo las veces que se ha encogido la imagen, siempre se sabe al tocar en el menú textura cuantas fotos mipmapping se hicieron de esa imagen, consiguiendo que el submenú contenga solo esas texturas para usarlas en los edificios con el diseño posterior Ilustración 38.

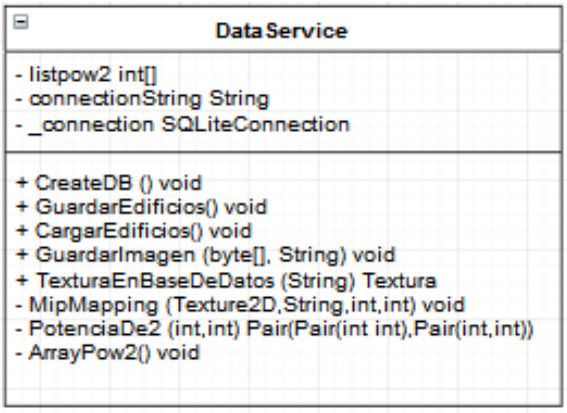

Ilustración 38: Diseño de la gestión de texturas y edificios [AUTOR]

El menú de ‘’texturas’’ y de ‘‘texturas con el mipmapping’’ previamente nombrados se observan en el apartado 5.2 del proyecto.

El problema de este apartado del mipmapping residía en conseguir un buen escalado. Para ello se usó una muestra de números enteros con distintas potencias de dos para encontrar la más cercana por debajo de nuestra textura original, pero al tratar de escalarla con el método “Scale” de texture2D ofrecida por Unity, la imagen se corrompía. Se encontró un script llamado TextureScale que resolvía dicho problema con los métodos “Point” o “Bilinear” (dependiendo de la calidad deseada se utiliza uno u otro). Se puede encontrar la referencia a ese archivo al final del documento en el apartado de bibliografía [18] y en la Ilustración 39 se puede observar sus elementos.

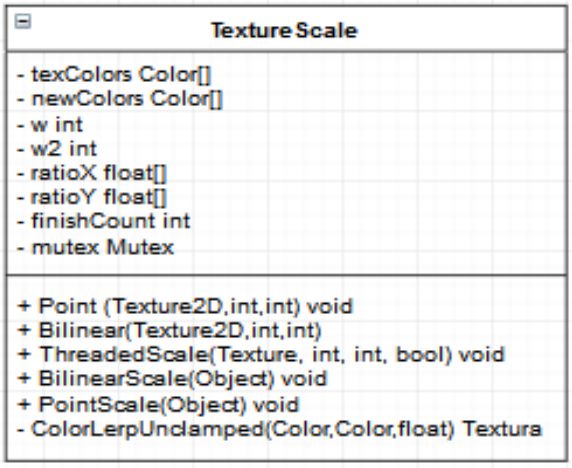

Ilustración 39: TextureScale [AUTOR]

Colocación de Texturas

Ya teniendo todo listo sobre los edificios en el apartado anterior, era necesario realizar mediante un Script la colocación de una textura en la pared que el usuario elija. Para ello, después de encontrar la pared mediante dos toques con el uso del “layer” y “tag” previamente mencionados. A continuación, tocamos la imagen realizando una búsqueda de la componente “RawImage” del edificio para modificar su textura y realizar dicho cambio.

Para terminar esta sección se mejoró el guardado de los edificios al almacenar previamente solo las paredes que han sido texturizadas, guardando la ruta de la pared y de la textura. Consiguiendo así no tener que realizar un guardado de todos los edificios sino solo los que el usuario ha modificado en ese momento. La clase encargada es la siguiente Ilustración 40:

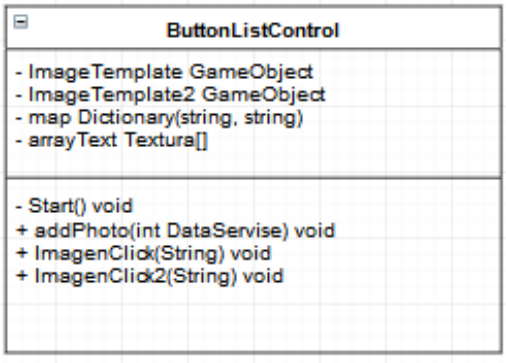

Ilustración 40: Clase para añadir texturas [AUTOR]

5.4 Adición de texturas

Teniendo el entorno creado se añadieron las distintas formas de almacenar las fotografías o texturas a nuestra aplicación. La primera forma que se va a explicar en esta sección es la de añadir fotos usando la galería Android.

En un primer momento se buscó la forma de poder acceder a la galería de fotos pulsando un botón desde nuestra aplicación. El problema que se encontró fue que se conseguía acceder a la galería, pero no se podía acceder a la foto por un motivo de incompatibilidad que tenían algunas versiones.

Para solucionar eso se pensó que la aplicación crea una carpeta en la galería en la que el usuario pudiera meter las fotos que quiera, ahorrando así tiempo de búsqueda de todas las fotos que contiene su móvil. Pero al crear dicha carpeta desde el móvil no se podía acceder a ella, debido a que el móvil no realiza una comprobación de los nuevos archivos hasta que se den los siguientes casos: Se conecta el dispositivo a un disco externo, se saca y se vuelve a meter la SD o si se reinicia el móvil.

La solución para este problema es forzar la búsqueda de nuevas fotos en la carpeta creada, consiguiendo así que el dispositivo Android detecte dicha carpeta e imágenes.

La otra manera de introducir texturas al proyecto es mediante el uso de la cámara. Este apartado es simple de realizar, ya que solo hace falta localizar la cámara trasera del móvil y luego realizar el proceso mipmapping de la foto obtenida. Hay que destacar que la foto sale con perspectiva, en el caso de que se requiera hacer una corrección en la foto se tendría que usar una aplicación de terceros.

El diseño encargado de realizar dichos procesos se mostrará a continuación en la Ilustración 41:

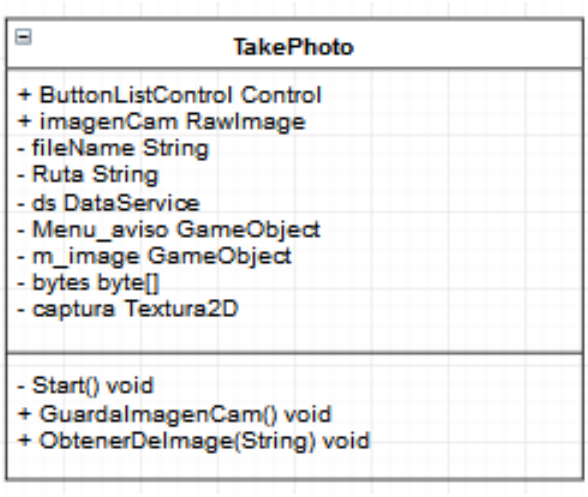

Ilustración 41: Clase tomar foto [AUTOR]

CAPÍTULO 6: IMPLEMENTACIÓN DEL SISTEMA

6.1 Preparación del entorno virtual y de las texturas

Entorno virtual

Para el caso de la creación del entorno virtual no hay parte de implementación, la herramienta Unity se encarga de asistir a la construcción de las calles, edificios, etc. También gestiona la utilización de los “layer” y “tag” y con ambos se puede encontrar las paredes de forma óptima con esta línea de código Ilustración 42 :

```
if (Physics.Raycast(Camera.main.ScreenPointToRay(Input.mousePosition), out hit, 50f, 1 << 8) && hit.transform.tag == "Pared")
```

Ilustración 42: Uso del "layer" y el "tag" [AUTOR]

Como se puede observar, se busca el objeto que ha sido tocado que contenga el “tag” pared y para saber que solo se buscará en el “layer” correspondiente se nombra de la siguiente manera *1 << 8* siendo el número 8 el “layer” donde se pretende buscar.

Con la utilización del método “Updated ()” se cuenta el número de toques que el usuario ha pulsado una pared en un intervalo de tiempo y lo vuelve de color amarillo si lo ha pulsado dos veces en menos de 0,5 segundos Ilustración 43:

```
if (Input.touchCount == 1 && Input.GetTouch(0).phase == TouchPhase.Began)
{
    tapCount++;
}
if (tapCount > 0)
{
    doubleTapTimer += Time.deltaTime;
}
if (tapCount >= 2)
{
    Detected();
    doubleTapTimer = 0.0f;
    tapCount = 0;
}
if (doubleTapTimer > 0.5f)
{
    doubleTapTimer = 0f;
    tapCount = 0;
}
```

Ilustración 43: Elegir edificio[AUTOR]

Por último, para terminar de preparar el entorno virtual, se implementa la manera de seleccionar un edificio con el objetivo de que se pueda colocar alguna de las texturas que haya en la aplicación Ilustración 44:

```
void Detectado()
{


    RaycastHit hit;
    if (objeto != null)
    {
        Renderer color1 = objeto.GetComponent<Renderer>();
        color1.material.color = Color.white;
    }

    if (!this.GetComponent<vistazenital>().getCambio())
    {
        if (Physics.Raycast(Camera.main.ScreenPointToRay(Input.mousePosition), out hit, 50f, 1 << 8) && hit.transform.tag == "Pared")
        {
            objeto = hit.collider.gameObject;
            Renderer color = objeto.GetComponent<Renderer>();
            color.material.color = Color.yellow;
        }

    }
}
```

Ilustración 44: Selección de una pared [AUTOR]

Vista cenital

En este caso se introdujo un icono de la cámara en la esquina superior izquierda para producir un sencillo cambio de una cámara a otra dependiendo de cuál esté activada en ese momento. Como se puede visualizar en el código de la siguiente Ilustración 45:

```
public void Cambio_vista() {
    if (!cambio)
    {
        camara.enabled = true;
        camara_principal.enabled = false;
        cambio = true;
        guia.SetActive(true);
        control.enabled = false;
    }
    else {
        camara.enabled = false;
        camara_principal.enabled = true;
        cambio = false;
        guia.SetActive(false);
        control.enabled = true;
    }
}
```

Ilustración 45: Cambio de vista [AUTOR]

Para la localización del usuario en esta vista se ha optado por manejar un foco que se moverá con el usuario y en el caso de querer navegar de manera rápida por el mapa solo se tiene que tocar la pantalla. La implementación se puede observar a continuación en el método “FixedUpdate ()” del Script vistaCenital Ilustración 46:

```
private void FixedUpdate()
{
    if (cambio)
    {
        if (Input.GetMouseButtonDown(0))
        {
            RaycastHit hit;
            Debug.DrawRay(camara.ScreenPointToRay(Input.mousePosition).origin, new Vector3(0,-500,0), Color.red, 20.0f);
            if (Physics.Raycast(camara.ScreenPointToRay(Input.mousePosition), out hit, 500f,1 << 8) && hit.transform.tag == "Terrain")
            {
                player.transform.position = new Vector3(hit.point.x, player.transform.position.y, hit.point.z);

            }

        }
    }
}
```

Ilustración 46: Transportar usuario [AUTOR]

Mipmapping

Para comenzar esta sección, primero hay que realizar la creación de la base de datos y cargar las primeras texturas con la implementación siguiente que leen la carpeta donde están las texturas en la Ilustración 47

```
_connection.CreateTable<Textura>();
_connection.CreateTable<Buildings>();
_connection.CreateTable<Numeracion>();

try
{
    if (this.GetTamtablaTexture() == 0)
    {
        if (Application.platform == RuntimePlatform.Android)
        {
            BetterStreamingAssets.Initialize();
            string[] dirs = BetterStreamingAssets.GetFiles("texturasInicial", "*.*");


            foreach (string nombreimage in dirs)
            {
                if (!nombreimage.Contains(".meta") && (nombreimage.Contains(".jpg") || nombreimage.Contains(".png")))
                {
                    string[] subfile = nombreimage.Split('/');
                    GuardarImagen(BetterStreamingAssets.ReadAllBytes(nombreimage), subfile[1]);
                }
            }
```

Ilustración 47: Creación de la BBDD [AUTOR]

La Ilustración 48 muestra cómo se añaden las distintas imágenes a la Base de datos.

```
    Texture2D bit = new Texture2D(2, 2, TextureFormat.ARGB32, false);
    bit.LoadImage(data);

    //Comprobamos que este mismo elemento no esté en la tabla y no lo añade

    Pair<Pair<int, int>, Pair<int, int>> anchoxalto = Potenciade2(bit.width, bit.height);
    string[] separador = nombre.Split('.');
    string nombrecorrecto = separador[0];

    if (this.TexturaEnBaseDeDatos(nombrecorrecto) == null)
    {

        var p = new Textura
        {
            Nombre = nombrecorrecto,
            Ancho = bit.width,
            Alto = bit.height,
            Image = data,
            //se guarda el numero de veces que se ha hecho mipmapping
            Nmipmap = Math.Min(anchoxalto.Second.First, anchoxalto.Second.Second)
        };
        _connection.Insert(p);
        MipMapping(bit, nombrecorrecto, anchoxalto.First.First, anchoxalto.First.Second);
    }
}
catch
{
    Debug.Log("fallo en la insercción de fotos");
}
```

Ilustración 48: Guardando texturas [AUTOR]

Después de tener la BBDD lista y el proceso de leer las imágenes de prueba, hay que realizar el mipmapping de manera que se escale correctamente la textura. Primero se ha de encontrar las dimensiones en potencia de dos correcta y también obtener las veces que se irá a encoger la imagen como se muestra en la siguiente Ilustración 49:

```
private Pair<Pair<int, int>, Pair<int, int>> Potenciade2(int widht, int heigh)
{
    ArrayPow2();
    Pair<int, int> a = new Pair<int, int>();
    Pair<int, int> b = new Pair<int, int>();
    Pair<Pair<int, int>, Pair<int, int>> valor = new Pair<Pair<int, int>, Pair<int, int>>();
    valor.First = a;
    valor.Second = b;
    bool secon_power_width = false;
    bool secon_power_height = false;

    for (int i = 0; i < listpow2.Length; i++)
    {
        if (!secon_power_width && listpow2[i] >= widht)
        {

            valor.First.First = listpow2[i - 1];
            secon_power_width = true;
            valor.Second.First = i;
        }

        if (!secon_power_height && listpow2[i] >= heigh)
        {

            valor.First.Second = listpow2[i - 1];
            secon_power_height = true;
            valor.Second.Second = i;
        }
    }
    return valor;
```

Ilustración 49: Obtención de las dimensiones de la texturas [AUTOR]

Con motivo de finalizar de realizar el mipmapping, se mostrará el método que gestiona las veces que es necesario escalar de las imágenes y se observa que el límite del proceso es hasta 64 pixeles como se mencionó en la sección 4.4. Además se mostrará la función que Unity no realiza correctamente que es el escalado como se puede comprobar en la Ilustración 50 e Ilustración 51 respectivamente:

```
private void MipMapping(Texture2D orig, string nombre, int ancho, int alto)
{
    //hacemos mipmapping
    //comprobamos si alguna de las imagenes mipmapping han sido borradas para restaurarlas
    string nombreaux = nombre;

    while (ancho > 64 && alto > 64)
    {
        nombreaux = nombre + " " + ancho.ToString() + " x " + alto.ToString();

        byte[] data;
        TextureScale.Point(orig, ancho, alto);
        data = orig.EncodeToJPG();

        var p = new Textura
        {
            Nombre = nombreaux,
            Ancho = ancho,
            Alto = alto,
            Image = data,
            Nmipmap = -1
        };
        _connection.Insert(p);
        alto = alto / 2;
        ancho = ancho / 2;
    }
}
```

Ilustración 50: Mipmapping [AUTOR]

```
public static void PointScale(System.Object obj)
{
    ThreadData threadData = (ThreadData)obj;
    for (var y = threadData.start; y < threadData.end; y++)
    {
        var thisY = (int)(ratioY * y) * w;
        var yw = y * w2;
        for (var x = 0; x < w2; x++)
        {
            newColors[yw + x] = texColors[(int)(thisY + ratioX * x)];
        }
    }

    mutex.WaitOne();
    finishCount++;
    mutex.ReleaseMutex();
}
```

Ilustración 51: Realizando el escalado de imágenes [AUTOR]

Colocación de Texturas

Teniendo un edificio seleccionado previamente se tendrá que utilizar el menú de textura para poder colocarlas en las distintas paredes que contiene el proyecto. Primero se tendrá que cargar el menú de textura con el método Añadir fotos Ilustración 52:

```
public void AñadirFoto(int valor_anterior, DataService ds)
{

    arrayText = ds.GetTextures();
    Transform[] trans = GameObject.Find("Canvas").GetComponentsInChildren<Transform>(true);
    GameObject button = null;
    foreach (Transform t in trans)
    {
        if (t.gameObject.name == "ButtonscrollList")
        {
            button = t.gameObject;
           button.SetActive(true);

        }
        if (t.gameObject.name == "Menu_texturas")
        {

            t.gameObject.SetActive(true);

        }
    }

    GameObject image = Instantiate(ImageTemplate) as GameObject;
    image.name = arrayText[valor_anterior].Id.ToString();
    image.SetActive(true);

    image.transform.SetParent(ImageTemplate.transform.parent, false);
    Texture2D prueba = new Texture2D(2, 2, TextureFormat.ARGB32, false);
    prueba.LoadImage(arrayText[valor_anterior].Image);
    image.transform.GetComponent<RawImage>().texture = prueba;

    button.SetActive(false);
}
```

Ilustración 52: Cargar menú texturas [AUTOR]

Estando en el menú de texturas el usuario tendrá que tocar una textura que quiera utilizar, con el objetivo de cargar el menú de mipmapping de dicha imagen Ilustración 53:

```
int id = int.Parse(text) - 1;
int mip = arrayText[id].Nmipmap;
Transform[] trans = GameObject.Find("Canvas").GetComponentsInChildren<Transform>(true);


foreach (Transform t in trans)
{
    if (t.gameObject.name == "ButtonscrollList2")
    {

        t.gameObject.SetActive(true);
        break;

    }
}
GameObject submenu = GameObject.Find("ButtonscrollList2");

for (int i = id + 1; i < id + mip; i++)
{

    GameObject image = Instantiate(ImageTemplate2) as GameObject;
    image.name = arrayText[i].Id.ToString();
    image.SetActive(true);

    image.transform.SetParent(ImageTemplate2.transform.parent, false);
    Texture2D prueba = new Texture2D(2, 2, TextureFormat.ARGB32, false);
    prueba.LoadImage(arrayText[i].Image);

    image.transform.GetComponent<RawImage>().texture = prueba;

}
this.gameObject.SetActive(false);
```

Ilustración 53: Textura elegida [AUTOR]

Para finalizar, falta poner la textura con el tamaño elegido en la pared asignada como se puede observar en la Ilustración 54:

```
int id = int.Parse(text) - 1;
Debug.Log(arrayText[id].Nombre);
Texture2D texture = new Texture2D(2, 2, TextureFormat.ARGB32, false);
texture.LoadImage(arrayText[id].Image);
texture.name = arrayText[id].Nombre;
GameObject camara = GameObject.Find("FirstPersonCharacter");
PonerTextura p = camara.GetComponent<PonerTextura>();
if (p.obtenerObjeto() != null)
{

    GameObject objeto1 = p.obtenerObjeto();
    objeto1.transform.GetComponent<MeshRenderer>().material.mainTexture = texture;
```

Ilustración 54: Añadir textura en edificio [AUTOR]

6.2 Adición de texturas

Tras lo explicado en el apartado de diseño, primero se mostrará la implementación de obtener una foto mediante la cámara. Para conseguirlo, se debe activar la cámara trasera del móvil, realizar la captura y guarda la imagen pasando por el procedimiento de mipmapping y guardado de fotos Ilustración 55:

```
public void GuardaImagenCam()
{

    try
    {
        captura = new Texture2D(webCamTexture.width, webCamTexture.height);
        captura.SetPixels(webCamTexture.GetPixels());
        captura.Apply();
        bytes = captura.EncodeToJPG();
        int valor = ds.GetTamtablaTexture();
        ds.GuardarImagen(bytes, fileName + ds.Getnumeracion() + ".jpg");
        Control.AñadirFoto(valor, ds);
    }
    catch { Debug.Log("fallo"); }
}
```

Ilustración 55: Guardar textura mediante la cámara [AUTOR]

Para concluir esta sección, hay que gestionar las texturas que serán guardadas en la carpeta “Image” del dispositivo Android sea cual sea su versión. Para empezar, hay que crear la carpeta y cada vez que se guarden las texturas hacer que el dispositivo las analice para que aparezca ambos en la galería. A continuación, se observa cómo se introduce en la carpeta “Image” una fotografía llamada “Final.png” y como se fuerza una lectura con la clases ANDROIDJAVA Ilustración 56:

```
public void ObtenerDeImage()
{
    string path = Application.persistentDataPath.Substring(0, Application.persistentDataPath.IndexOf("/Android")) + "/Imagen";
    BetterStreamingAssets.Initialize();

    if (Application.platform == RuntimePlatform.Android)
    {
        if (!System.IO.Directory.Exists(path))
        {

            System.IO.Directory.CreateDirectory(path);
            string[] dirs = BetterStreamingAssets.GetFiles("Otratextura", "*.*");
            foreach (string nombreimage in dirs)
            {
                if (!nombreimage.Contains(".meta"))
                {

                    string[] subfile = nombreimage.Split('/');
                    Texture2D ss = new Texture2D(2, 2);
                    byte[] b = BetterStreamingAssets.ReadAllBytes(nombreimage);
                    ss.LoadImage(b);
                    File.WriteAllBytes(path + "/Final.png", b);
                    File.OpenRead(path + "/Final.png");
                    using (AndroidJavaClass jcUnityPlayer = new AndroidJavaClass("com.unity3d.player.UnityPlayer"))
                    using (AndroidJavaObject joActivity = jcUnityPlayer.GetStatic<AndroidJavaObject>("currentActivity"))
                    using (AndroidJavaObject joContext = joActivity.Call<AndroidJavaObject>("getApplicationContext"))
                    using (AndroidJavaClass jcMediaScannerConnection = new AndroidJavaClass("android.media.MediaScannerConnection"))
                    using (AndroidJavaClass jcEnvironment = new AndroidJavaClass("android.os.Environment"))
                    using (AndroidJavaObject joExDir = jcEnvironment.CallStatic<AndroidJavaObject>("getExternalStorageDirectory"))
                    {
                        jcMediaScannerConnection.CallStatic("scanFile", joContext, new string[] { path+"/Final.png" }, null, null);
                    }
```

Ilustración 56: Creación y carga de la carpeta “Image” [AUTOR]

Dependiendo del caso que se dé se crea un mensaje distinto, el primer caso muestra que no se añadido ninguna nueva textura en la carpeta Ilustración 57:

```
string[] dirs = System.IO.Directory.GetFiles(path);

if (dirs.Length == 1)
{
    Menu_aviso.SetActive(true);
    GameObject aviso = GameObject.Find("Aviso");
    aviso.GetComponent<Text>().text = "No hay fotos que agregar";
}
```

Ilustración 57: No hay Imágenes que añadir [AUTOR]

El siguiente caso muestra el proceso de guardado de todas las imágenes que se han encontrado en la carpeta Ilustración 58:

```
bool ninguna_nueva = true;
foreach (string nombreimage in dirs)
{
    if (!nombreimage.Contains(".meta") && (nombreimage.Contains(".png") || nombreimage.Contains(".jpg")) && !nombreimage.Contains("Final.png"))
    {
        string[] sub = nombreimage.Split('/');
        string[] separador = sub[sub.Length - 1].Split('.');
        string nombrecorrecto = separador[0];
        Textura textura = ds.TexturaEnBaseDeDatos(nombrecorrecto);
        if (textura == null) {
            ninguna_nueva = false;
            byte[] b = File.ReadAllBytes(nombreimage);
            int valor = ds.GetTamtablaTexture();

            ds.GuardarImagen(b,  sub[sub.Length - 1].ToString());
            Control.AñadirFoto(valor, ds);
        }
    }
}

Menu_aviso.SetActive(true);
GameObject aviso = GameObject.Find("Aviso");
if (!ninguna_nueva)
{
    aviso.GetComponent<Text>().text = "Se han añadido al menú de textura";
}
```

Ilustración 58: Añadiendo texturas [AUTOR]

Para finalizar, después de analizar si todas las texturas han sido analizadas y no hay nuevas se muestra un mensaje al usuario comunicándoselo Ilustración 59:

```
else {
    aviso.GetComponent<Text>().text = "No habia ninguna nueva foto que añadir";
}
```

Ilustración 59: No hay nuevas texturas que añadir [AUTOR]

6.3 Tipos de usuarios

Tras completar el proyecto, se ha pensado realizar un conjunto de casos de prueba. Se escogerán a tres tipos de usuarios con diversas experiencias con aplicaciones móviles, a estos usuarios se les realizará una encuesta para comprobar la aceptación de la aplicación. Las preguntas son las siguientes:

- Pregunta 1: ¿Crees que son realistas los edificios que han sido texturizados? Puntúe de 0 a 10.

- Pregunta 2: Pensando en la interfaz de usuario. ¿Has tenido algún problema al interactuar con las distintas funcionalidades del proyecto? Indique de 0 a 10 el grado de dificultad, 0 es complejo y 10 sencillo.

- Pregunta 3: Califica de 0 a 10, siendo 0 poco intuitivo y 10 muy intuitivo, el grado de accesibilidad que presenta el programa.

- Pregunta 4: ¿Te parece correcto la utilización de la combinación de colores del programa? Siendo 0 poco de acuerdo y 10 totalmente de acuerdo.

- Pregunta 5: Con respecto a la navegación del entorno virtual, puntúa de 0 a 10, ¿Crees que ha sido intuitivo el modo de mover la cámara y el usuario?

- Pregunta 6: Puntúa de 0 a 10, ¿Piensas que las maneras de obtener nuevas texturas son sencillas y ofrecen resultados realistas? siendo 0 no estar de acuerdo y 10 estar de acuerdo.

- Pregunta 7: Describe el grado de innovación que presenta esta aplicación móvil, 0 es poco innovador y 10 es bastante innovador.

Las personas escogidas para la realización de este proyecto están divididas en 3 niveles diferentes dependiendo de sus experiencias con este tipo de aplicaciones, habiéndolo separados en los niveles de conocimiento: principiante, intermedio y avanzado. En el siguiente apartado se realizará una breve descripción de cada uno de los usuarios y los resultados obtenidos de cada uno de ellos.

Para comenzar, definimos al usuario principiante como aquel que han tenido poco contacto con aplicaciones móviles siendo su conocimiento escaso para proceder solo en esta aplicación. Este usuario requiere de una explicación detallada de las distintas funcionalidades de la aplicación, teniendo que ayudar durante el desarrollo de cualquier actividad en concreto. Parte del reto del desarrollo del

programador consistirá en reducir el periodo de aprendizaje para que el usuario pueda familiarizarse con el entorno de manera rápida.

A continuación, el usuario intermedio tiene un conocimiento suficiente para entender con una simple explicación todas y cada una de las características del programa y realizar cualquier actividad con poca ayuda. En este nivel escogeremos a usuarios que han utilizados aplicaciones de dispositivos móviles pero que no esté acostumbrado a manejar herramientas de este estilo.

Por último, el usuario avanzado goza de un alto conocimiento sobre la tecnologías y aplicaciones de esta índole. Este tipo de usuarios formaron parte de otros proyectos profesionales similares y su opinión sobre esta herramienta es relevante pues serán más exigente sobre las características del programa. Su nivel les permite manejar la aplicación de manera que no requieran de ayuda.

Después de describir a los usuarios elegidos para la obtención de los resultados de esta encuesta, se presentarán la media de las calificaciones obtenidas en la siguiente Tabla 13:

	Usuario Principiante	**Usuario Intermedio**	**Usuario Avanzado**
Pregunta 1	8	8	7
Pregunta 2	7	8	9
Pregunta 3	7	7	9
Pregunta 4	8	9	8
Pregunta 5	7	9	9
Pregunta 6	7	8	6

	Usuario Principiante	Usuario Intermedio	Usuario Avanzado
Pregunta 7	10	9	10

Tabla 13: Resultados de la encuesta

Tras obtener las distintas respuestas de cada usuario se puede contemplar la aceptación de los usuarios en este programa gracias a sus altas calificaciones. Para poder analizar de forma conjunta se va a agrupar los datos con objetivo de ver el impacto que obtendría este prototipo en el caso de que salga al mercado **Error! Reference source not found.**.

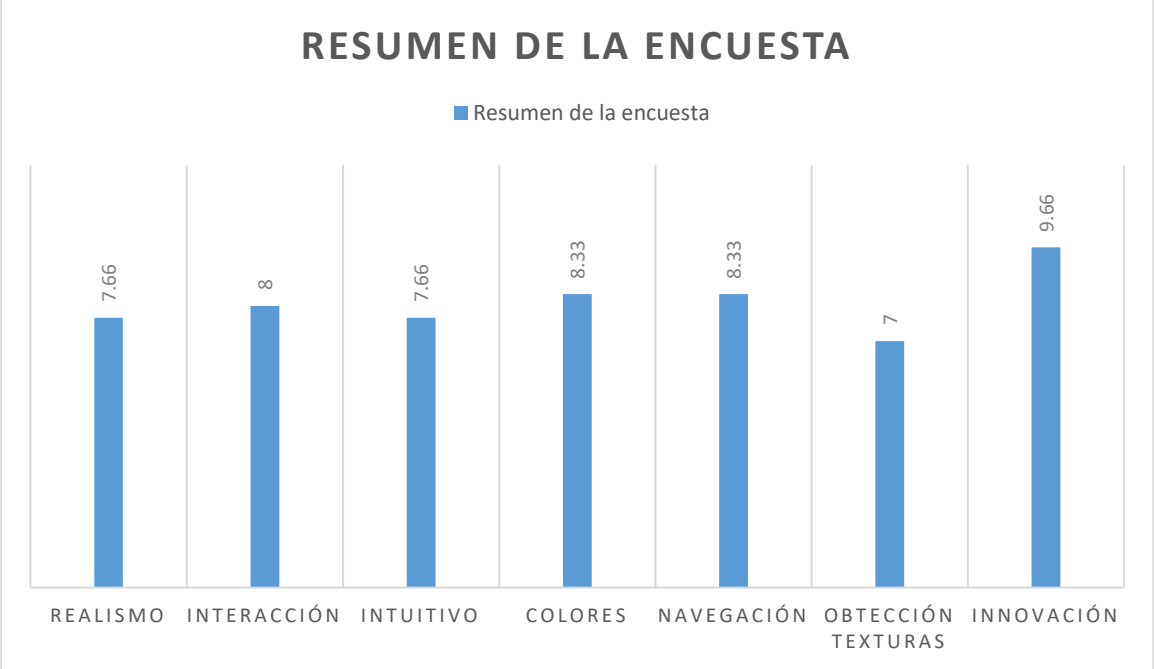

Ilustración 60: Resumen de la encuesta

Como podemos comprobar en la imagen, los usuarios tienen una buena impresión sobre la aplicación en todos los aspectos. Destacando en la innovación porque parece que los usuarios no han utilizado o desconocen herramientas parecidas. Aunque otros aspectos a mejorar son la manera como se obtiene textura y la sensación de realismo de la aplicación.

6.4 Observaciones finales de la implementación

Esta última parte se ha dejado para comentar las correcciones de los posibles errores que pueda contener el proyecto, mejorar/optimizar los distintos elementos y algunos comentarios en el transcurso del mismo:

La principal característica de este proyecto es la realización del mismo solo con una persona, haciendo que el proyecto no pueda realizarse de forma concurrente para ahorrar horas de trabajo.

Se ha cambiado durante las iteraciones la forma en la que se abren los menús, usando el método “Start ()” de Unity3D. Utilizándose para obtener una referencia al elemento del menú y poder llamarlo directamente en vez de tener que buscar el elemento una y otra vez cuando se pulse un menú. Con el motivo de ahorrar líneas de código y sea más legible el texto Ilustración 61:

```
public void Start()
{
    ButtonscrollList = GameObject.Find("ButtonscrollList");
    Menu_text = GameObject.Find("Menu_texturas");
    Opciones = GameObject.Find("Opciones");
    Menu_Opciones = GameObject.Find("Menu_Opciones");
    Menu_Opciones.SetActive(false);
    Anadir = GameObject.Find("Anadir");
    Menu_anadir = GameObject.Find("Menu_anadir");
    Menu_anadir.SetActive(false);
    Menu_aviso = GameObject.Find("Menu_aviso");
    Menu_aviso.SetActive(false);

}
```

Ilustración 61: Asignando los menús [AUTOR]

Otra deficiencia importante que hizo perder bastante tiempo es la realización de pruebas en el dispositivo móvil con el uso de Unity. Este tiene problemas de identificación con los dispositivos algo frecuentes y tras crecer el proyecto tardaba más la construcción de la aplicación en el dispositivo Android, llegando a durar de diez a quince minutos. Por cada línea que se cambie en Unity se ha de volver a empaquetar el proyecto para realizar las pruebas pertinentes tardando el tiempo previamente mencionado.

CAPÍTULO 7: CONCLUSIONES

7.1 Conclusión

Los avances tecnológicos que han ido aconteciendo estos años han traído muchas nuevas herramientas al mercado, facilitando aspectos de nuestras vidas.
Se han obtenido resultados realistas como si se hubiesen llevado al cabo las tareas en la realidad para la obtención de datos o tener la experiencia para poderlas hacer de verdad.

Desde un punto de vista empresarial, aún se pueden abarcar muchos proyectos en entorno 3D para mejorar la productividad de algunos trabajos que hoy por hoy usan métodos bastantes eficientes. Aunque con nuevas herramientas como esta pretende, se puede optimizar dicha eficiencia y además agilizar distintos procesos costosos en el tiempo.

En este trabajo de fin de grado se ha desarrollado una solución relativa a la decoración de edificios tratando de abordar distintas disciplinas del grado de ingeniería informática, para la resolución de las complicaciones que ocurrieron durante el desarrollo. Se han estado utilizando herramientas de gestión de proyectos con las cuales tener guardadas copias del proyecto y practicar para cuando se realicen trabajos en equipo. En el apartado de implementación para el modelado 3D de los objetos del entorno virtual se les dota de características a fin de que el usuario pueda interactuar con esos elementos consiguiendo realizar su tarea.

Todas y cada una de las asignaturas que engloba la ingeniería informática han sido cruciales para la realización de este trabajo gracias las aptitudes y conceptos obtenidos en el grado, pues han sido de vital importancia para la finalización de este proyecto. Cabe destacar que también ha sido útil la investigación principal en el caso de observar los diversos requisitos y herramientas para la elaboración de este trabajo.

Para finalizar este apartado voy a redactar una enumeración de los conocimientos y aptitudes que he ido aprendiendo a lo largo del proyecto.

Trabajar de manera iterativa me ha fomentado una buena actitud hacia el trabajo futuro. Gracias a presentar cada prototipo a la tutora he mejorado mi competencia comunicativa, siendo esta muy importante para el futuro como para pensar en mecánicas adicionales con las cuales mejorar la aplicación en un futuro. Todos estos pasos previos han ayudado a la realización de este proyecto para que los distintos usuarios puedan beneficiarse de esta simulación de decoración en un futuro.

7.2 Trabajo futuro

Para esta sección del documento se plantearán nuevas líneas de desarrollo con la visión de mejorar esta aplicación posteriormente. A pesar de todas las características generales que se reunieron en la versión final del prototipo, quedan disponibles nuevas áreas para seguir añadiendo nuevas mejoras con el propósito de ayudar a los usuarios con modelado.

Un cambio que se extendía demasiado de este TFG es la realización de un servidor externo por cual cada usuario podría observar el proyecto de otras personas. Así como realizar cambios con el objetivo de usar esta herramienta de forma colaborativa entre todos los usuarios, empresas, etc. Pero tras un análisis se comprobó que habría demasiados problemas:

- El principal problema es el de exclusión mutua, el cómo gestionar los distintos proyectos. Si se tiene un proyecto que luego otro usuario utilice ese proyecto y tras su uso trate de guardarlo. Hay que pensar si es necesario que haya redundancia de proyectos, es decir, que cada uno tenga en su usuario su propio proyecto sin que sustituya el del usuario original o al ser colaborativa si se pueda realizar el guardado, pero solo si tiene permiso.

- La gestión de los distintos usuarios que contenga la aplicación. Se deberían analizar varios rangos disponibles con distintos privilegios siendo considerados el modo invitado si el usuario no se registra.

  El caso de que se registre será un administrador de sus proyectos, siendo este capaz de controlar los privilegios de otros usuarios para darles niveles de libertad. Algunos solo podrán usar el proyecto y no podrán sustituir en el trabajo del usuario administrador. Los que este defina como amigos aparte de poder usar el proyecto tendrían derecho a poder guardar los cambios que haga. Haciendo esta medida útil para cooperar entre compañeros buscando la mejor forma de decoración.

  En pos de realizar esta versión mejorada del programa en un futuro, se piensa que es mejor realizarla tras haber recibido el conocimiento mediante el Master de Seguridad para ayudar en la realización de este punto, para gestionar la seguridad de los usuarios y de privilegios.

- Añadir más nivel de detalle para que el usuario pueda decorar más cosas, preparar varios objetos como ventanas, puertas, asfalto entre otros. Siendo también un buen planteamiento que el usuario pueda crear sus propios edificios y almacenarlos en una lista de edificios propios del usuario.

BIBLIOGRAFÍA

[1] *BBC*. (Abril de 2017). Recuperado el Marzo de 2019, de BBC.com: https://www.bbc.com/mundo/media-39655682

Android Studio. (s.f.). *Android*. Recuperado el Marzo de 2019, de https://developer.android.com/studio

Azuela, J. H. (2011). *Procesamiento y Análisis Digital de Imágenes.* Ra-Ma.

BBC. (Abril de 2017). Recuperado el Marzo de 2019, de BBC.com: https://www.bbc.com/mundo/media-39655682

Caballero, J. (Marzo de 2016). *Armadillo Amarillo*. Recuperado el Marzo de 2019, de https://www.armadilloamarillo.com/blog/novedades-de-android-studio-2-0/

Chacon, S., & Straub, B. (2014). *Pro Git.* Apress.

Dingman, H. (Marzo de 2015). *pcworld*. Recuperado el Abril de 2019, de https://www.pcworld.com/article/2892314/unity-5s-new-full-featured-personal-edition-is-completely-utterly-free-to-use.html

Domínguez, A., Navarro, F., & Javier, C. M. (2017). *Unity 2017.x.* Ra-Ma. Recuperado el Marzo de 2019

Folgado, R. (Abril de 2014). *El mundo*. Recuperado el Marzo de 2019, de https://www.elmundo.es/economia/2014/04/02/533af90022601dbe748b4589.html

Gómez, C. L., Álvarez García, A., & de las Heras del Dedo, R. (2017). *MÉTODOS ÁGILES: SCRUM, KANBAN, LEAN (2ª ED.).* ANAYA. Recuperado el Febrero de 2019

Gómez, J. (Abril de 2013). *Laboratorio*. Recuperado el Marzo de 2019, de IT: https://www.laboratorioti.com/2013/04/22/estimacion-de-costes-con-cocomo-81-ii/

gwiazdorrr. (2017). *github*. Recuperado el Marzo de 2019, de https://github.com/gwiazdorrr/BetterStreamingAssets

Haines, E. (Octubre de 2016). *unity3d*. Recuperado el Abril de 2019, de TextureScale: http://wiki.unity3d.com/index.php/TextureScale

Historia de la informática. (Enero de 2011). Recuperado el Abril de 2019, de https://histinf.blogs.upv.es/2011/01/07/mundos-virtuales/

Karim, I. (Marzo de 2018). *stack overflow*. Recuperado el Marzo de 2019, de https://stackoverflow.com/questions/49304874/what-is-the-difference-between-the-android-p-and-api-level-27-system-images

Loganathan, G. (2015). *Java helps*. Recuperado el Febrero de 2019, de https://www.javahelps.com/2015/01/install-android-studio-in-ubuntu-1404.html

Okita, A. (2014). *Learning C# programming with Unity 3D.* CRC press.

Pascual, J. A. (Julio de 2018). *Computer hoy*. Recuperado el Febrero de 2019, de https://computerhoy.com/reportajes/industria/android-vs-iphone-guerra-smartphones-cifras-271447

Passsy. (Octubre de 2017). *Jlelse*. Recuperado el Marzo de 2019, de https://android.jlelse.eu/7-reasons-this-android-code-style-improves-your-productivity-65d196fa55f

robertohuertasm. (Diciembre de 2017). *Github*. Recuperado el Mayo de 2019, de https://github.com/robertohuertasm/SQLite4Unity3d

Rosso, R. (Octubre de 2015). *uptodown*. Recuperado el Febrero de 2019, de https://android-sdk.uptodown.com/windows

Unity Technologies. (2019). *Unity*. Recuperado el Febrero de 2019, de https://store.unity.com/es/products/unity-personal?_ga=2.157446200.309120779.1556205886-1575540075.1553284190

ANEXO A: MANUAL DE INSTALACIÓN

En el caso de esta iteración, se basa en la realización de un estudio sobre las herramientas a utilizar y como conectarlas entre sí. Al utilizar Unity el proceso es más simple, ya que automáticamente se encarga de conectar con Android. Pero para ello debemos instalar cada una de las herramientas que se va a usar.

Para empezar, se hablará de los pasos para instalar el SDK de Android necesaria para poder empezar a hacer aplicaciones móviles. Para ello, vamos a descargar el SDK manager que gestiona los distintos componentes que luego necesitamos, obtenido del siguiente enlace [19]. Tras descargarlo lo abrimos y le damos a siguiente hasta que te pida una ruta donde se descargará (recomiendo que sea en el escritorio para facilitar la búsqueda de la ruta) esperamos y cuando se descargue todo en la carpeta que hayas elegido lo ejecutamos.

Tras ejecutar aparecerá el siguiente menú para poder descargar los elementos necesarios Ilustración 62.

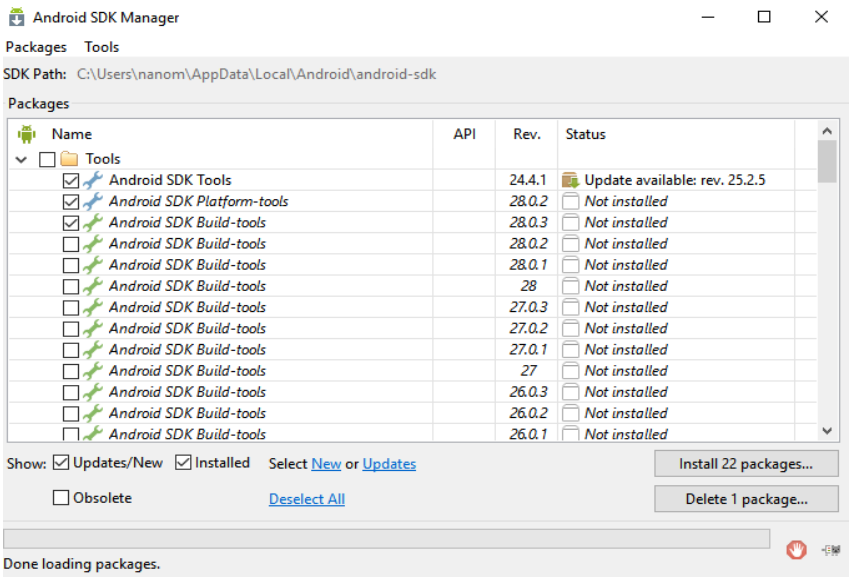

Ilustración 62: SDK manager [AUTOR]

Se tienen que descargar los siguientes paquetes:

- De la carpeta tools Ilustración 63:

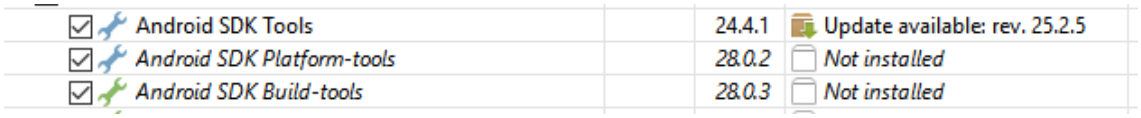

Ilustración 63: Elementos a descargar (carpeta tools) [AUTOR]

- De la carpeta Android 9 Ilustración 64:

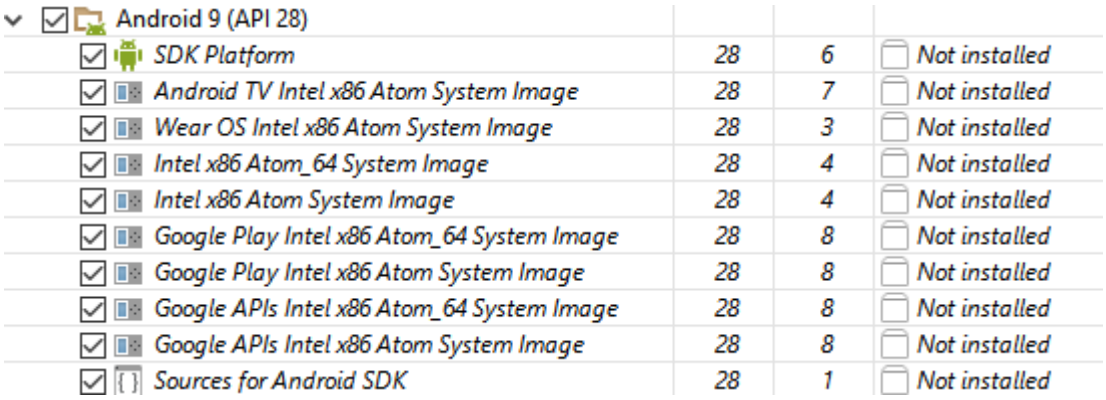

Ilustración 64: Elementos a descargar (carpeta Android 9) [AUTOR]

Ya teniendo preparado todo ello, es hora de descargar la herramienta Unity y para empezar hemos de volver a descargar la última versión desde su página en el siguiente enlace [20].

Tras descargarlo, lo instalamos hasta que accedemos a la ventana que aparece en la Ilustración 65.

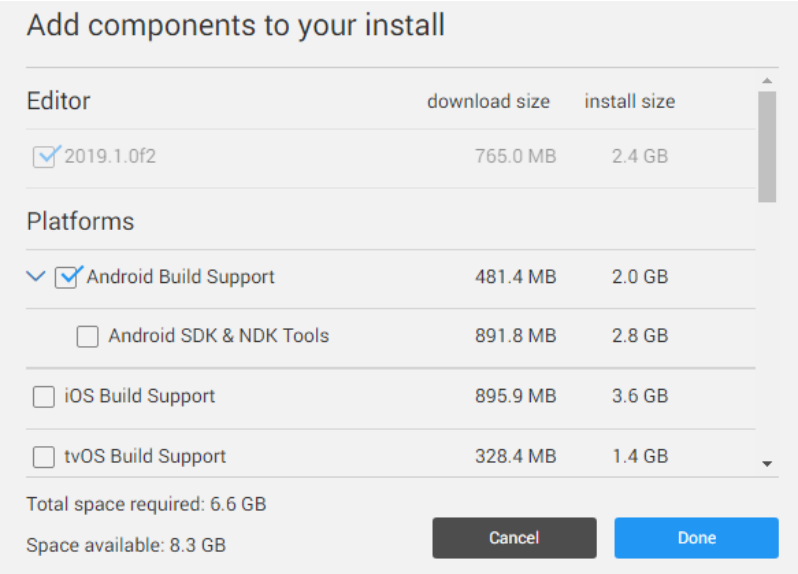

Ilustración 65: Instalación de Unity [AUTOR]

Ya teniendo unity instalado con el soporte de Android por un lado y el SDK instalado en el escritorio (se descarga de la manera descrita en el proyecto, a causa de que en las primeras pruebas no funcionaba el SDK teniendo que descargar una versión anterior).

Con Unity descargado y ejecutado, se crea un proyecto en edit->preferences->external Tools donde en la parte de Android buscamos el lugar donde hemos puesto el SDK y lo agregamos como esta en la siguiente Ilustración 66:

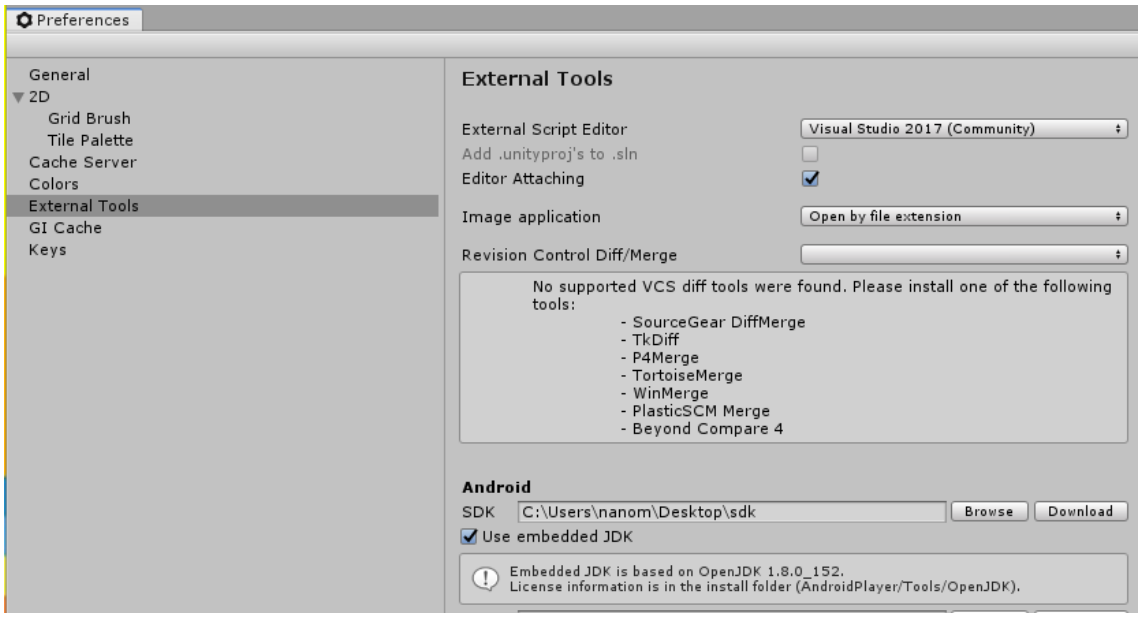

Ilustración 66: Integrando el SDK en Unity [AUTOR]

Otro detalle del apartado external Tools es que puedes modificar la herramienta para codificar los distintos Script en vez de utilizar Monodevelop y poder emplear Visual Studio como se ve en la imagen anterior.

Por último, para tener preparado Unity en el apartado File->built Settings, hay que pulsar el icono de Android para que Unity pueda compilar en el dispositivo. Para acabar, en la misma ventana pulsamos en Player Settings, nos aparecerá una pestaña en la cual tenemos que cambiar el nombre de la compañía y el producto al que el usuario desee. Pero en el sección Identificación tiene que ponerse con los nombres con el siguiente formato: com.(NombreCompañia).(NombreProducto) sino dará un error en la construcción. Se hace como a continuación en la Ilustración 67 e Ilustración 68.

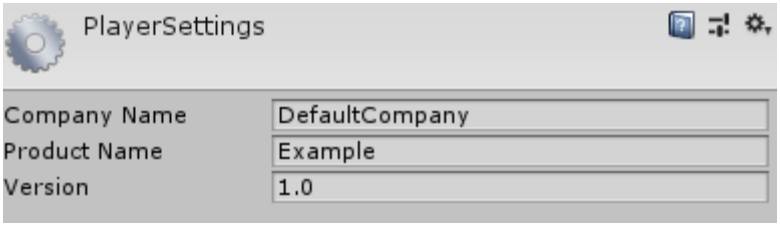

Ilustración 67: Identificar el proyecto (1) [AUTOR]

Ilustración 68: Identificar el proyecto (2) [AUTOR]

Ya teniendo las herramientas preparadas nos falta añadir la base de datos SQLite. Se ha optado por un script aportado en la página de Unity para facilitar dicha conexión y añadir los archivos a la carpeta StreamingAssets (o crearla si no existe). [21]

ANEXO B: IMÁGENES EN ALTA RESOLUCIÓN

En este apartado se encontrarán las fotos que necesiten visualizarse en alta calidad.

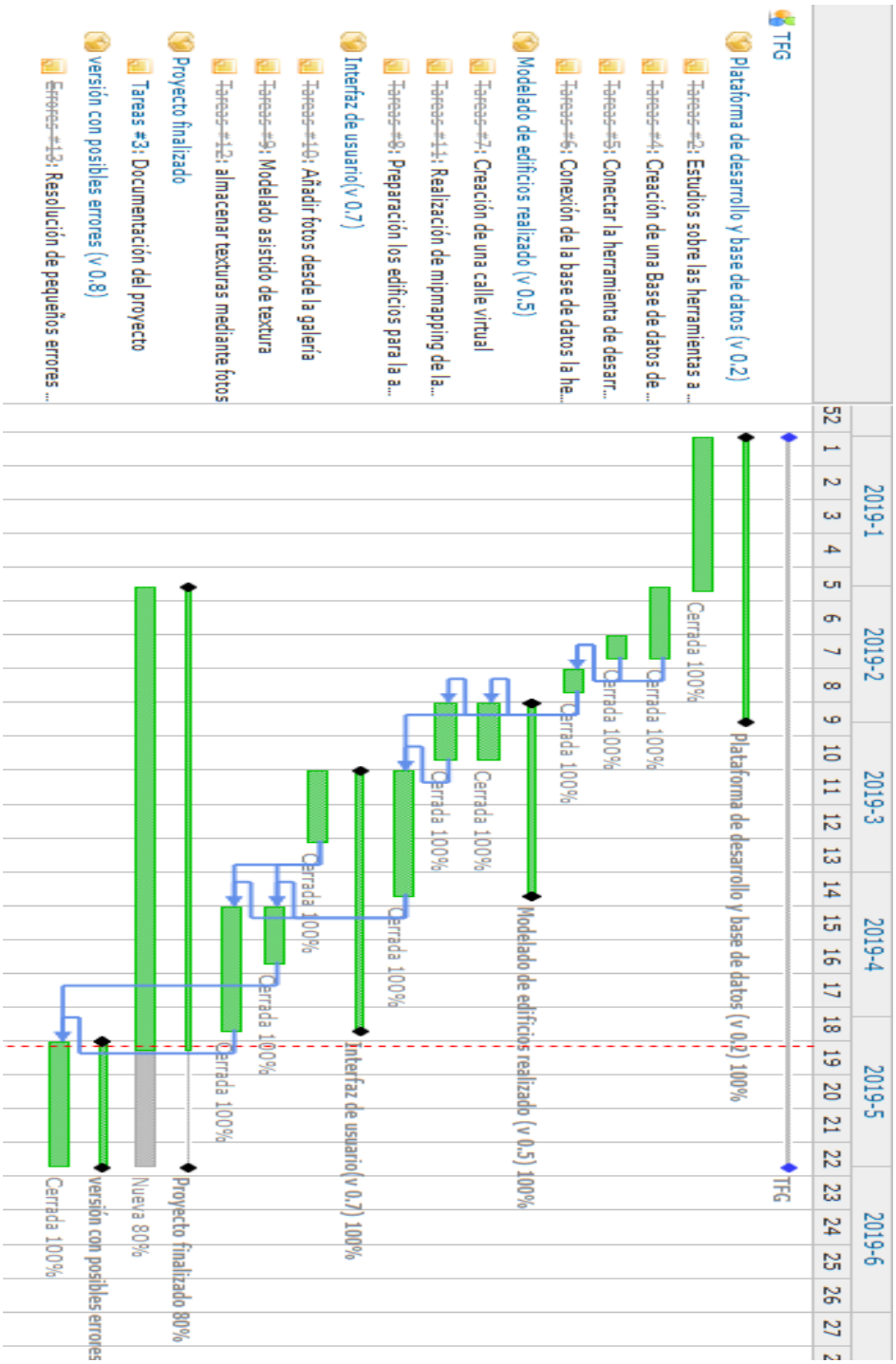

Ilustración 69: Diagrama de Gantt [AUTOR]

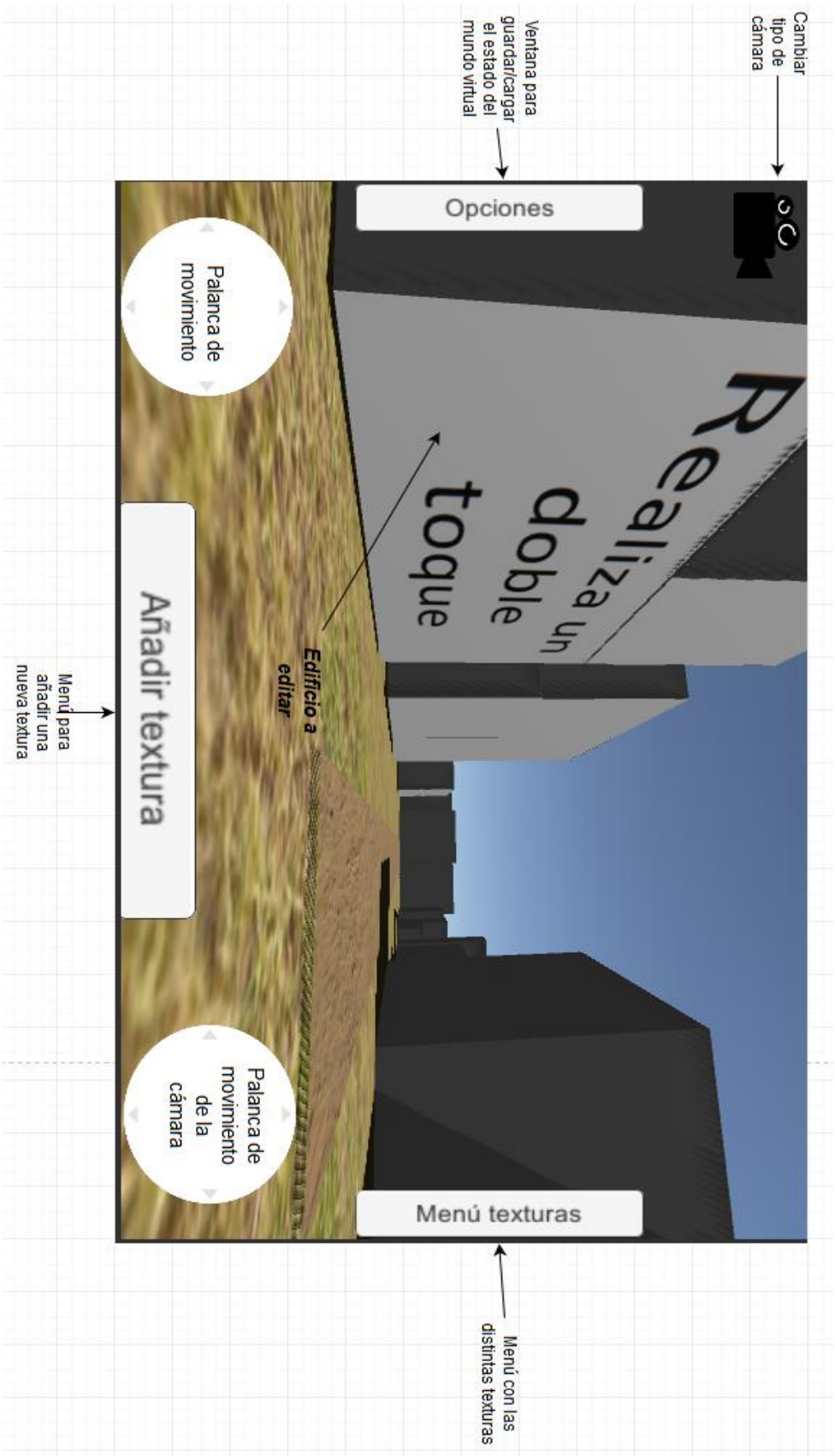

Ilustración 70: Interfaz del entorno virtual [AUTOR]