UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Centre de la Imatge i la Tecnologia Multimèdia

Creación de un Environment 3D en Unreal Engine 4

David Lira Martí

Director: Carles Sora

Grado: Diseño y Desarrollo de Videojuegos

Curso: 2022-23

Universidad: CITM-UPC

Índice

Resumen

Este trabajo consiste en la creación de un entorno 3D para el motor de videojuegos *Unreal Engine 4.* Actualmente, hay muchos programas y herramientas distintas para la creación de entornos 3D, adentrarse en este mundo puede ser complejo para aspirantes a artistas digitales 3D o para algunos desarrolladores independientes que quieran realizar sus propios videojuegos que funcionen en un motor 3D como *Unreal Engine.*

El proyecto intentará dar la respuesta a cuál es la mejor metodología, programas a utilizar, así como la solución a los problemas más comunes que pueden surgir durante el desarrollo del mismo. Para ello se ha hecho un trabajo de investigación sobre la creación de entornos en videojuegos, los distintos *softwares* en el mercado con sus respectivas posibilidades y los distintos estilos artísticos que se utilizan en la actualidad. Además, se explicará la gestión y metodología para el proyecto que se va a realizar en concreto, junto a un diario de desarrollo, donde se explicará con más detalle los pasos para obtener el resultado final esperado, desde la conceptualización del entorno, pasando por la creación de *assets*, hasta la publicación en internet para obtener visibilidad por el trabajo realizado.

Como resultado práctico del proyecto se ha creado un pequeño entorno con piezas modulares de un castillo flotante que sirve como confirmación de que el uso de la metodología y las herramientas seleccionadas son las correctas. El resultado ha servido para concluir que una sola persona puede llegar a crear un entorno 3D de una calidad aceptable con las herramientas, conocimiento y metodologías adecuadas, sin obviar algunas de las limitaciones que puede haber irremediablemente como el nivel de detalle o la calidad técnica de algunos *assets* que irremediablemente requieren de mucho tiempo y práctica, o en algunos casos, incluso de expertos en temas específicos como la creación de texturas.

Además, el proyecto finaliza con la publicación del *environment* en internet para dar a conocer el trabajo al mundo, gracias a esto se han valorado nuevas posibilidades para mejorar el proyecto en un futuro gracias al *feedback* recibido de otros artistas.

Palabras clave

Arte

Modelado

Unreal Engine

Entorno

Texturización

Iluminación

Creatividad

Enlaces

Artstation:

https://www.artstation.com/artwork/ob6akJ

Video:

https://drive.google.com/file/d/1j-NTNm9uTAL9oCfC_npJ5VfH4TWHbl76/view?usp=sharing

Índice de tablas

Índice de figuras

Glosario

***Environment*:** Obra artística de vanguardia hecha de distintos elementos repartidos en un espacio por el que se puede transitar.

Juego AAA: Videojuego realizado por grandes compañías con una gran inversión monetaria detrás, así como un grupo importante de desarrolladores centrados en tareas concretas.

Juego *Indie*: Son videojuegos creados por grupos pequeños de desarrolladores con bajo presupuesto para su desarrollo.

Motor: Nos referimos a un entorno de desarrollo que proporciona herramientas para crear videojuegos, es importante para el resultado final de un *environment* gracias a la iluminación y *shaders*.

***Software 3D*:** Cualquier herramienta utilizada para el desarrollo 3D, ya sea para crear modelos, texturas o materiales.

Modelo: Es el conjunto de polígonos que forman un objeto en 3D.

Textura: Son imágenes cuadradas que aportan información sobre la superficie de un material.

Material: Referencian las texturas a usar, le dan el acabado final a los *assets* en el motor de juego.

***UV map*:** Técnica que se utiliza para asignar texturas y colores a un modelo tridimensional, se despliega la geometría del modelo en un plano para poder pintar en 2D el modelo 3D.

***Asset*:** Es cualquier elemento que forma el conjunto de un videojuego, en este caso se le llamará *asset* a un elemento 3D terminado con su material aplicado.

***High Poly - Low Poly*:** Un modelo es *high poly* cuando su cantidad de polígonos es muy elevada, en cambio decimos que es *low poly* cuando la cantidad de polígonos que usa es baja, normalmente el *low poly* es el que se añade al motor mientras que el *high poly* es que se usa para crear la textura con más detalle.

Retopología: Es el proceso por el cual un modelo *high poly* pasa a ser *low poly*, se realiza normalmente a mano usando un *software* 3D como *Maya*.

***Blueprints*:** Son sistemas de comandos visuales basados en añadir y conectar nodos, se pueden usar, por ejemplo, para crear materiales en *Unreal Engine* o *Substance Designer.*

***Blockout*:** Es un boceto 3D con el que montar la escena antes de añadir los *assets* definitivos, se usan modelos simples y sin texturas.

1. Introducción

Este trabajo consiste en la creación de un entorno virtual en el motor destinado para la creación de videojuegos *Unreal Engine 4*, se explicará cómo se desarrolla el *environment* desde la conceptualización artística, metodologías de trabajo, el diferente uso de programas de modelado y texturización, implementación de los modelos a Unreal Engine 4 hasta los diferentes problemas y soluciones a los mismos.

En este proyecto se realizará un entorno de fantasía medieval protagonizado por un gran castillo flotante rodeado de varias torres mágicas que harán que el castillo se mantenga en el cielo. El tipo de estilo artístico utilizado será el estilizado, se eligió este estilo porque el estilizado es una técnica que se puede hacer por una sola persona, un estilo más realista requeriría de un tiempo y personal muchísimo más amplio que no es posible asumir por una sola persona. Además, este estilo es bastante popular dentro del desarrollo del videojuego actualmente ya que, ahorra bastantes recursos por el bajo poligonaje que supone y el menor detalle en las texturas.

Algunos ejemplos de videojuegos que usan un estilo 3D estilizado en la actualidad:

Figura 1: Nintendo, (2017), *The Legend of Zelda: Breath of the Wild*

Figura 2: Andrew Shouldice, (2022), *TUNIC*

Figura 3: Epic Games, (2017), *Fortnite: Battle Royale*

1.1. Motivación

Desde muy pequeño he quedado fascinado por los mundos de fantasía en los que me he sumergido gracias a los videojuegos, de todos los componentes que se utilizan para adentrar al jugador a un nuevo mundo, tales como la música, las mecánicas, la narrativa, etc. El que más me llegó a impactar eran los grandes escenarios que algunos juegos podían llegar a tener, me quedaba horas explorando estos escenarios y me fascinaba cómo podía llegar a crearse algo tan inmersivo.

Los videojuegos de rol japoneses (JRPG) fueron los principales causantes del interés sobre este tema, grandes sagas como *Final Fantasy, Dragon Quest, Xenoblade Chronicles*, entre otras muchas, gozan de extraordinarios escenarios como los que se muestran a continuación. Mi inspiración para el proyecto proviene de todos estos mundos de fantasía a los que he estado expuesto desde mi infancia.

Todo esto se suma a mi afición por el modelaje 3D que he descubierto durante el grado de Diseño y Desarrollo de Videojuegos que estoy acabando de cursar. Creo que este proyecto es el ideal para mí, ya que aparte de mejorar mi técnica con programas en los que ya tengo experiencia como *Maya*, *Zbrush* o *Substance Painter*, aprenderé a usar herramientas que nunca he usado, tales como *Unreal Engine 4* y *Substance Designer*.

Por esta razón decidí adentrarme en la creación de videojuegos, de todos estos entornos con los que quedé fascinado iban surgiendo ideas en mi cabeza de cómo yo haría estos escenarios. Varios años después, tengo la oportunidad de plasmar una de esas ideas en un trabajo de fin de grado.

Al final del trabajo espero poder haber mejorado mis habilidades como artista 3D y poder tener un porfolio que presentar para poder entrar a una empresa dedicada al desarrollo de videojuegos.

Figura 4: Monolith Soft, (2020), *Xenoblade: Definitive Edition*

Figura 5: Square Enix, (2017), *Dragon Quest XI*

Figura 6: Square Enix, (2016), *Final Fantasy XV*

1.2. Formulación del problema

Crear un entorno en 3D para un videojuego necesita un gran equipo profesional para cada aspecto del mismo (iluminación, modelados, entornos, texturizado, etc.), esto es algo asumible para un gran equipo de desarrollo para juegos AAA, pero para el caso de los videojuegos *indies*, si se quiere hacer un juego en 3D atractivo visualmente lo más probable es que los pocos artistas que habrá en el estudio se encuentren con diferentes problemas con los tiempos de desarrollo y la calidad del resultado final.

En este trabajo se intentarán solucionar estos problemas que se pueden encontrar al crear un entorno 3D tales como, ¿Qué estilo artístico es el adecuado para mi proyecto? ¿Qué programas puedo usar para cada aspecto del desarrollo? ¿Cómo puedo reutilizar *assets* sin que se vea reflejado en el resultado final?, entre otras preguntas que puedan surgir durante el desarrollo del *environment* 3D. También se hablará sobre la optimización en un motor de videojuegos, gracias a la retopología entre otras técnicas que se utilizarán durante el proyecto, la organización a seguir para llegar a las fechas propuestas y hacia la parte final de proyecto se comentarán los distintos métodos para darse a conocer como artista y técnicas para lucir mejor tu trabajo en páginas web como *Artstation*.

La idea es que a partir de este trabajo es crear una metodología de trabajo óptima para que personas o estudios pequeños puedan crear sus propios *environment* 3D. Podemos encontrar una gran cantidad de contenido sobre todo en YouTube donde se explica como hacer determinadas tareas sobre el desarrollo 3D, canales como el de "*3dEx*" muestran el desarrollo de *assets* y algunas técnicas de implementación en motores como *Unreal*, otros canales se centran en mostrar elementos más concretos del desarrollo 3D como la creación de materiales o el uso de *ZBrush*.

La gran diferencia de este proyecto respecto a este tipo de información en internet es que aquí se mostrará todo el desarrollo desde cero, identificando los problemas que pueden ir surgiendo, así como sus soluciones, además será una guía mucho más adecuada para alguien con poco conocimiento sobre el tema, principalmente por abarcar el tema desde la conceptualización del trabajo hasta la publicación en el porfolio.

1.3. Objetivos generales del TFG

Los objetivos generales para este trabajo son los siguientes:

- Crear un entorno con el motor *Unreal Engine 4*
- Crear *assets* y texturas de calidad
- Diseñar un escenario de fantasía medieval
- Crear una metodología de trabajo para este tipo de proyectos
- Presentar el trabajo correctamente en un porfolio

1.4. Objetivos específicos del TFG

Los objetivos específicos que busca cumplir este trabajo son:

- Aprender a usar *Unreal Engine 4* para el desarrollo 3D
- Mejorar mis técnicas de modelado con *Maya* y *Zbrush*
- Mejorar y ampliar mi conocimiento con *Substance Painter*
- Aprender a usar *Substance Designer*
- Aprender a usar *Nodos* en *Unreal Engine 4*
- Optimizar el proyecto usando correctamente la retopología
- Mejorar mis habilidades como artista digital

1.5. Alcance del proyecto

Como se ha mencionado previamente, este proyecto irá dirigido a pequeños estudios que están empezando ahora con el desarrollo de videojuegos y necesiten una metodología para crear entornos 3D con poco personal disponible.

También puede usarlo cualquier persona con interés por el arte 3D y quiera una guía con las mejores opciones y herramientas para poder empezar sus proyectos. El público objetivo del trabajo está entre los 16-22 años, que es más o menos donde se considera que podría estar un aspirante a artista 3D júnior, esto no quita que personas de cualquier edad puedan llegar a usar el proyecto como guía de desarrollo para sus propios proyectos.

Gracias a que se detallarán los pasos a seguir para cada tarea a realizar, además de diferentes justificaciones del *software* seleccionado, así como la solución de varios problemas que se pueden ir encontrando, este trabajo será de mucha utilidad para el tipo de persona que he mencionado. El trabajo puede convertirse en una buena herramienta para pequeños estudios *indies*, escuelas o universidades que quieran una guía general para el desarrollo de *environments*, incluso para individuos que quieran aprender de arte 3D en solitario, por lo que podemos decir que este trabajo será beneficioso para este tipo de personas en concreto.

2. Estado del arte/Marco Teórico/Contextualización/Estudio de Mercado

Para este proyecto se ha decidido centrar el estado del arte en tres apartados diferenciados, lo primero será analizar los diferentes estilos artísticos que existen en el mundo de los videojuegos actualmente. El segundo apartado consistirá en ver que se ha hecho previamente en el sector sobre el tema que se quiere tratar, se analizarán diferentes juegos 3D *indies* y lo que han llegado a realizar con pocos recursos en el ámbito del *environment* 3D. Por último, se analizarán los diferentes *softwares* disponibles y se seleccionará el más adecuado para cada apartado del proyecto.

2.1. Estilo Artístico

Antes de comenzar con el análisis de estilos artísticos cabe mencionar que aquí no se escribirá sobre recursos 2D como el *pixel art*, vectorizado, ilustraciones, entre otros, este análisis se centrará en estilos artísticos para el 3D, que es el tipo de proyecto que se quiere desarrollar. Por otro lado, sí se hablará de tecnologías para el desarrollo 3D que nos servirá para ver el panorama actual en la industria del videojuego.

2.1.1. Arte Retro

Intenta replicar los videojuegos antiguos cuando usaban *sprites* pixelados, la mayoría de juegos con arte retro son en 2D, pero hay varios casos como el de *A Short Hike* en la imagen de abajo, donde como se puede apreciar se intenta dar un estilo de *pixel art* a los elementos en 3D ya sea bajando la resolución o con cualquier otro tipo de técnica siempre que intente obtener esta estética retro.

Figura 7: Adam Robinson-Yu, Adamgryu, (2019), *A Short Hike*

2.1.2. Arte Cartoon

Es un tipo de estilo que luce como una ilustración o dibujo animado, también puede ser 2D. En 3D hay varios ejemplos de juegos con este tipo de arte como puede ser el de *The Legend of Zelda: Wind Waker*, que tiene este estilo tan característico gracias al diseño de personajes y escenarios y la técnica *cell shading*.

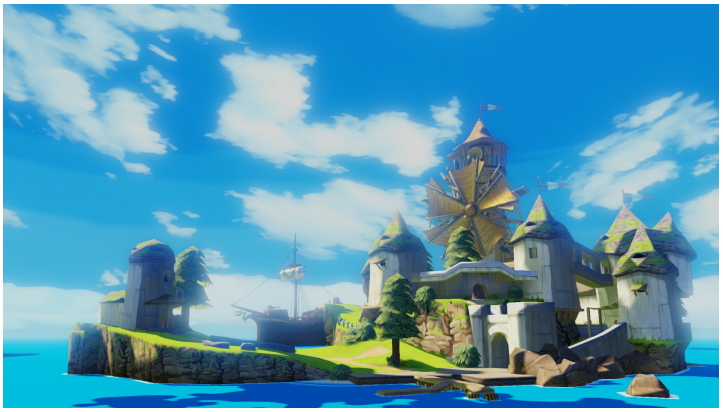


Figura 8: Nintendo, (2013), *The Legend of Zelda: Wind Waker HD*

2.1.3. Arte Único

El arte único intenta diferenciarse del resto de videojuegos mediante un estilo artístico totalmente diferente a lo visto anteriormente, se busca un resultado original, distinto y novedoso. Cuando un juego de estas características se hace conocido, suelen surgir otros que intentan copiar el estilo visual del mismo.

Figura 9: Playdead, (2010), *Limbo*

2.1.4. Arte Realista

Este estilo pretende imitar la realidad lo mejor posible, no es necesario que la ambientación sea algo realista, puede tener elementos fantásticos, como el ejemplo de *Final Fantasy XVI* abajo, pero requiere que los elementos en pantalla se sientan como en la vida real.

Figura 10: Square Enix, (2023), *Final Fantasy XVI*

2.1.5. Arte Estilizado

Se podría describir al arte estilizado como un punto intermedio entre el estilo *cartoon* y el estilo realista, se pueden considerar diferentes grados de estilizado, dependiendo del realismo que se le quiera dar a la estética. Como se ha mencionado en la introducción, este estilo de arte es el seleccionado para realizar el proyecto, respecto al grado de estilizado, será más cercano al cartoon que al realismo, algo entre *The Legend of Zelda: Wind Waker* y *Alice: Madness Returns*.

Figura 11: Spicy Horse, (2011), *Alice: Madness Returns*

2.2. Videojuegos Indies en 3D

Una gran parte de videojuegos creados por estudios *indies* están hechos en 2D, esta tendencia se debe a las limitaciones técnicas de este tipo de estudios con poco personal, crear un entorno 3D no es tarea fácil y no es viable para tan pocas personas. A continuación se mostrarán diferentes ejemplos de videojuegos *indies* que optaron por el diseño 3D, se analizará qué recursos utiliza cada uno para llegar a lucir bien sin un gran equipo detrás.

2.2.1. Untitled Goose Game

Untitled Goose Game es un videojuego creado por *House House*, un pequeño estudio situado en Melbourne, Australia, compuesto por cuatro miembros, Nico Disseldorp, Jake Strasser, Stuart Gillespie-Cook y Michael McMaster. Como se puede ver en la imagen el juego tiene una vista isométrica con gráficos 3D, lo primero que salta a la vista son las texturas, o más bien la falta de ellas, para desarrollar el juego utilizaron colores planos como textura, dándole ese estilo tan característico.

Además de este truco se puede apreciar que todos los modelados son muy sencillos y de poco poligonaje, incluyendo a los personajes que no tienen ningún tipo de detalle en la cara, gracias a esto los desarrolladores fueron capaces de construir el escenario y personajes del juego rápidamente.

Figura 12: House House, (2019), *Untitled Goose Game*

2.2.2. Neon White

Otro buen ejemplo de estudio pequeño capaz de crear un videojuego con escenarios 3D con muy pocos recursos, es el caso de *Angel Matrix* con *Neon White*, en la imagen de abajo se puede apreciar que usa la misma técnica que usaba *Untitled Goose Game*, pero en este caso se va más allá y se eliminan las texturas parcialmente para dejar las paredes del escenario totalmente blancas y lisas.

Este enfoque es muy interesante, ya que no solo se están ahorrando tiempo y dinero en crear un *environment* 3D, sino que además han creado su propio arte único muy reconocible visualmente, añadiendo elementos a color que destacan sobre el blanco para mejorar el propio *gameplay*.

Figura 13: Angel Matrix, (2022), *Neon White*

2.2.3. Weird West

El caso de *Weird West,* creado por *WofEye Studios*, tiene poco que ver con lo que se ha visto anteriormente, en vez de usar colores planos para hacer las texturas, se crean texturas, pero en vez de tener un tono realista, se dibujan con este estilo de cómic tan característico que hace que el escenario luzca genial desde arriba. Este estilo demuestra que lo que importa no es el detalle de la textura, sino el cómo a veces es mejor seguir un estilo visual simple y agradable que intentar ir a un estilo realista y fallar en el intento.

Figura 14: WolfEye Studios, (2022), *Weird West*

2.2.4. Signalis

Signalis es otro buen ejemplo del tipo de arte que hemos visto en el apartado de estilos artísticos, arte retro, el juego a simple vista parece ser en 2D, pero realmente mezcla tanto elementos 2D como 3D. Bajar la resolución de un objeto 3D hasta que quede pixelado como un juego de los 90 puede ser la solución, no necesariamente debe tener una cámara fija como el caso de *Signalis*, también puede tener un entorno totalmente en 3D y que todo el escenario funcione a una menor resolución como en el caso de *A Short Hike* del apartado anterior.

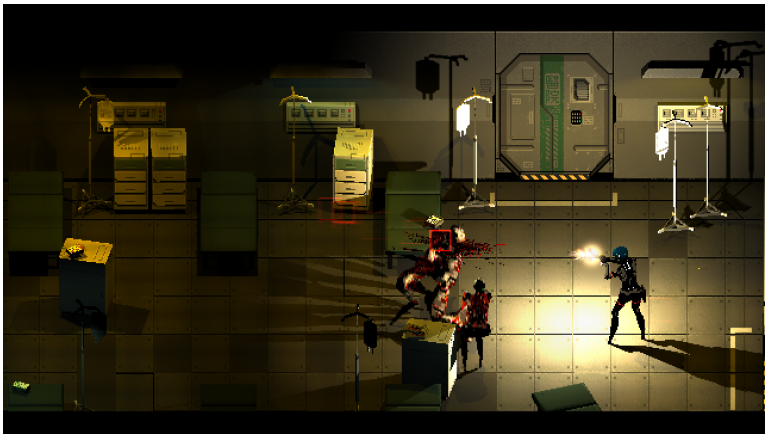


Figura 15: rose-engine, (2022), *Signalis*

2.2.5. Danganronpa V3: Killing Harmony

Por último, en esta lista de ejemplos se ha decidido incluir *Danganronpa V3: Killing Harmony*, un videojuego desarrollado por *Spike Chunsoft* con la característica de mezclar el 3D y el 2D directamente. El juego tiene una estética, anime y esto es lo que hace que se puedan añadir ciertos elementos como los personajes y los objetos de alrededor totalmente en 2D. Para realizar esta técnica hace falta que los objetos en 2D siempre apunten en la dirección que está mirando el cono de visión del jugador, ya que si no fuese así, el jugador podría rodear el objeto y perdería la inmersión del escenario.

Figura 16: Spike Chunsoft, (2017), *Danganronpa V3: Killing Harmony*

Volviendo al proyecto que se va a realizar, cabe comentar que tipo de técnicas se van a usar para el *environment* propuesto. Lo más importante es que el estilo va a ser estilizado, como se ha comentado anteriormente, esto quiere decir que se acercará bastante a lo que se ha visto en la imagen de ejemplo de *Alice: Madness Returns*, pero intentando mantener unas texturas más simplificadas sin llegar a la simplificación en exceso de *Untitled Goose Game*. Tampoco se obviará la técnica que se usa en *Danganronpa V3: Killing Harmony,* a la que se la dará uso en este proyecto, este recurso donde se usan elementos 2D puede funcionar muy bien para, por ejemplo, el césped en Unreal Engine 4, en el que se usará una textura en 2D repetida en grandes cantidades para dar este efecto de detalle del césped en una superficie.

Por lo general, se podría decir que el proyecto bebe de varios conceptos vistos en diferentes aplicaciones del estilo estilizado, desde lo mencionado anteriormente, hasta la creación de texturas planas al estilo cartoon de *Weird West* o el nivel de detalle muy bien escogido en los modelos en *Neon White*. El objetivo es conseguir un estilo atractivo intentando analizar lo que se ha hecho anteriormente en otros proyectos para crear un estilo propio.

2.3. Análisis del Software Disponible

2.3.1. Modelado 3D

A continuación se ha realizado una lista detallada con todos los programas de modelado 3D que se han considerado relevantes relacionados con el proyecto. Se han descartado programas como *SketchUp* y *Rhinoceros 3D*, ya que no han sido considerados relevantes para el proyecto por su orientación más arquitectónica y menos artística. Una vez aclarado este tema, la lista de programas es la siguiente.

2.3.2. Blender

Blender es una aplicación de código abierto, es decir que es totalmente gratuita para el público, además es bastante versátil, lo que nos permite crear modelos 3D y animaciones de todo tipo de complejidad. Otros puntos a favor pueden ser su interfaz intuitiva y sobre todo su gran comunidad debido a que puede usarla cualquier persona gratuitamente. Sin embargo, hay varios puntos en contra a analizar como puede ser su curva de aprendizaje algo elevada si no se ha usado antes y sus limitaciones a la hora de hacer ciertos modelos con más detalle como fibras y texturas de los materiales.

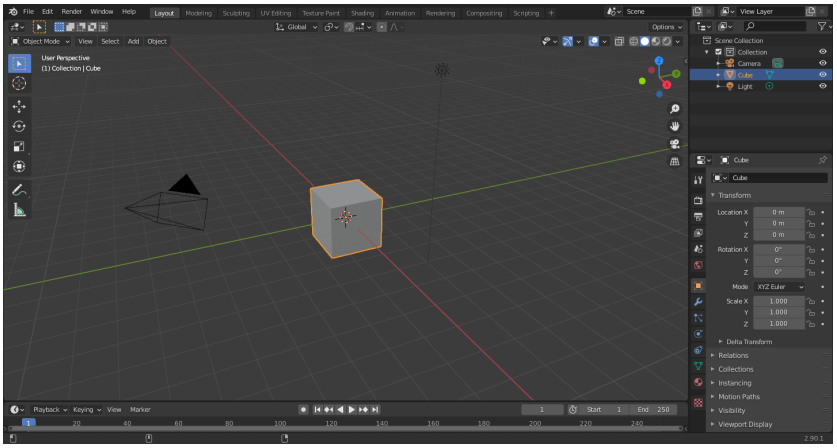


Figura 17: Captura de *Blender*

2.3.3. ZBrush

La principal característica de *ZBrush* es poder crear modelados muy detallados con facilidad como se puede ver en la imagen de abajo, ofrece un detalle muy muy elevado. La siguiente ventaja es la capacidad de importar modelos de otros programas como Blender o Maya para poder darles el detalle que queramos a partir del modelo creado previamente, además es uso de capas nos permitirá crear un *workflow* de trabajo que nos facilitará la creación de diferentes versiones de un modelo a la vez.

Por desgracia, *ZBrush* requiere de una suscripción de pago mensual, además de un ordenador de alto procesamiento y RAM, lo que supondrá una gran inversión para poder usarlo, su curva de dificultad es algo elevada, pero una vez se tienen conocimientos previos se convierte en una herramienta muy cómoda para el artista.

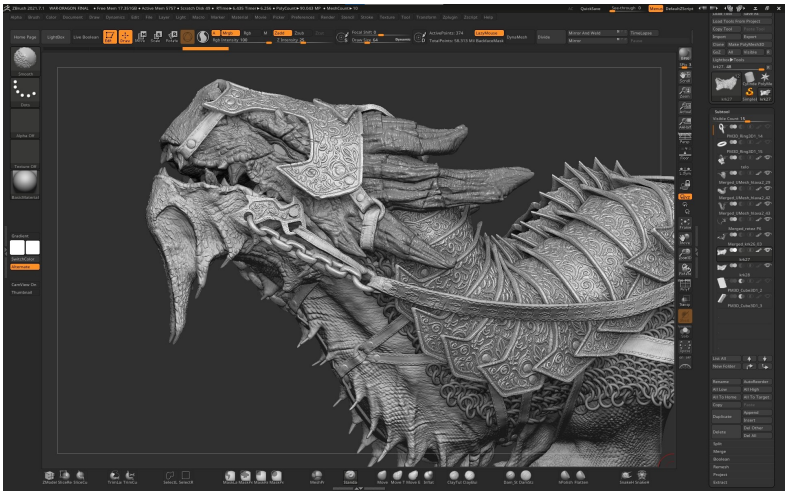


Figura 18: Captura de *Zbrush*

2.3.4. Cinema 4D

Cinema 4D quizás sea la herramienta con más opciones de efectos visuales e iluminación de la lista, como el resto de programas que aparecen aquí, no es muy diferente en cuanto a la interfaz como se puede ver abajo. Seguramente no sea el *software* más adecuado a lo que se quiere enfocar el proyecto, ya que en este caso los efectos visuales e iluminación vendrán del motor de juego. Debido a este enfoque más cinemático tiene alta compatibilidad con *Photoshop* y *After Effects*.

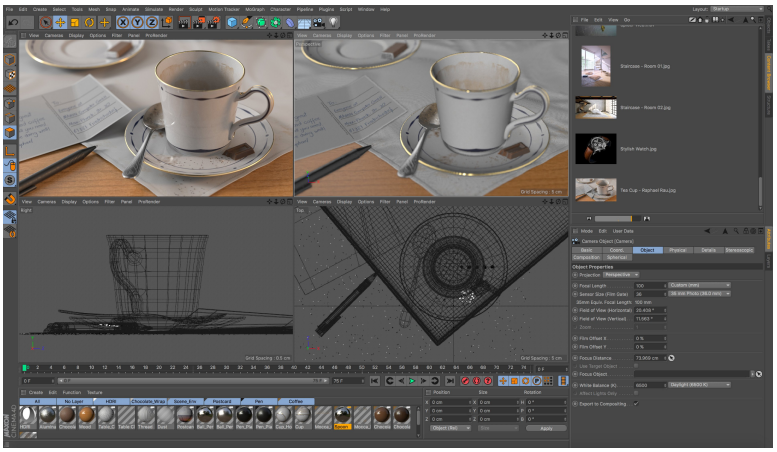


Figura 19: Captura de *Cinema 4D*

2.3.5. Autodesk Maya

Maya es un *software* 3D muy versátil y se pueden hacer varias cosas, desde animar, modelar o crear efectos visuales cinemáticos, *Maya* es muy popular y al igual que *Blender* cuenta con una comunidad muy grande de usuarios, por lo que se puede esperar mucha información sobre su uso en la red. Por desgracia, *Maya* no es gratuito y requiere de un pago mensual bastante elevado debido al soporte y la profesionalidad del programa, aún así, podría ser la mejor opción para lo que se quiere conseguir en el proyecto para hacer modelos básicos.

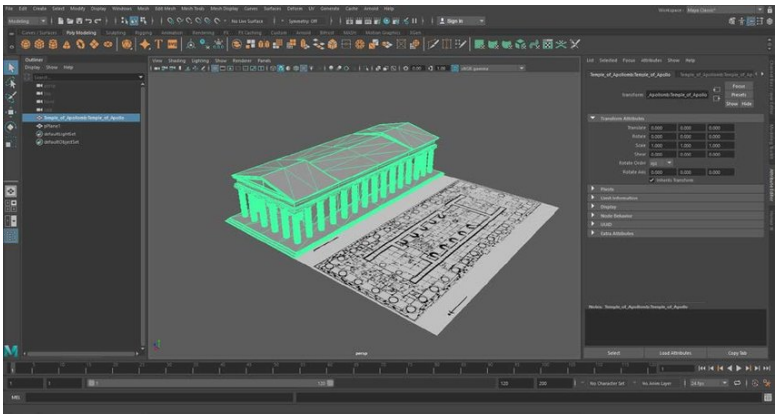


Figura 20: Captura de *Autodesk Maya*

2.3.6. Autodesk 3ds Max

3ds Max también es de *Autodesk* por lo que las posibilidades de ambos programas son muy parecidas, por esta razón es más conveniente poner su principal diferencia que se trata de los flujos de trabajo o *workflows*. *3ds Max* está más orientado a arquitectura, ingeniería y construcción, mientras que Maya está más centrado en multimedia y entretenimiento, por lo que para este proyecto el programa seleccionado sería *Maya*.

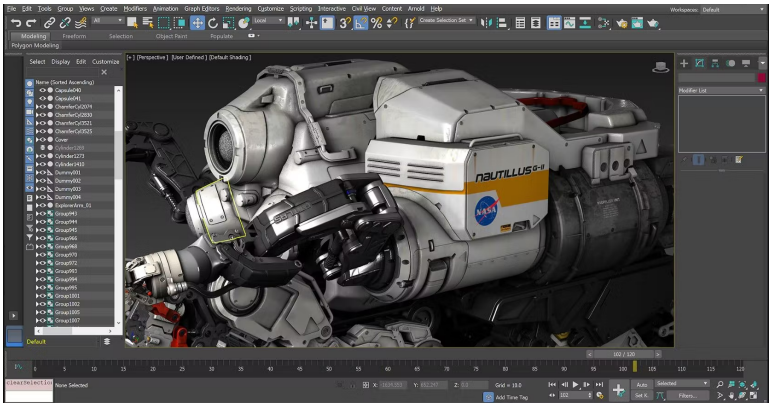


Figura 21: Captura de *Autodesk 3ds Max*

Después de este análisis de *software* 3D se decidió usar dos programas para la realización de este trabajo, el primero que se usará será *Maya*, se ha decidido usar este *software,* ya que, a pesar de que *Blender* tiene varias opciones que pueden resultar una mejor opción, como la versatilidad y el uso gratuito, con *Maya* el creador del proyecto tiene mucha más experiencia, por lo que valía la pena ahorrarse esa curva de aprendizaje de *Blender*. Además, con *Maya* la idea es modelar el *blockout*, objetos simples y hacer retopología de los modelos *high poly* por lo que lo más importante es que el desarrollador se sienta cómodo con la herramienta que está usando más que su complejidad.

Para los modelos *high poly* se usará *ZBrush*, de nuevo el creador del proyecto tiene más experiencia con este programa, pero en este caso valorando el resto de opciones se consideró que *ZBrush* era la opción correcta por delante del resto por la gran variedad de pinceles y opciones para crear un modelo 3D de calidad. Además, la potencia de procesamiento no será un problema, ya que se dispone de un ordenador de grandes características técnicas.

2.3.7. Texturizado

La mayoría de *softwares* mencionados en el apartado de Modelado 3D incluyen su propio método de texturización para el modelo que estamos realizando, pero esto no es suficiente para que el trabajo a realizar luzca lo mejor posible, por eso se usará otro *software* por separado para texturizar los modelos 3D. Además, necesitaremos uno de estos programas para crear materiales que nos permitirán aplicar una textura en nuestro entorno 3D.

2.3.8. Substance Designer

Substance Designer permite a los usuarios crear texturas y materiales complejos de manera rápida y eficiente utilizando una variedad de herramientas y efectos. Utiliza nodos para construir materiales y texturas, lo que permite a los usuarios crear materiales personalizados con un alto grado de precisión y control. Además, puede generar texturas procedurales utilizando una variedad de efectos y herramientas, lo que permite a los usuarios crear texturas de manera rápida y eficiente. *Substance Designer* es probablemente el mejor programa del mercado para crear materiales y texturas.

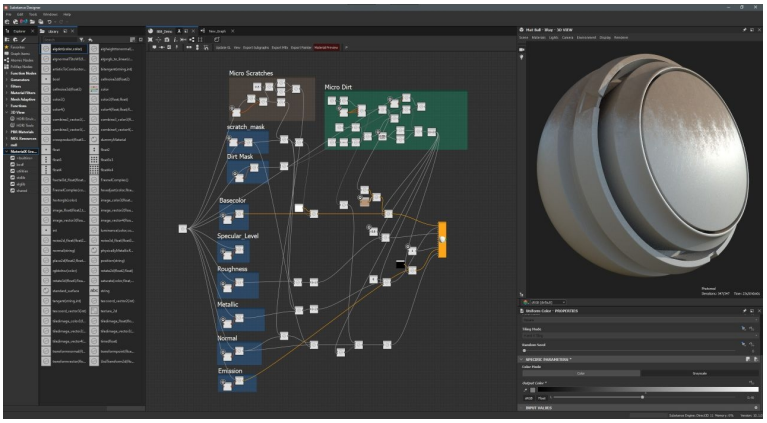


Figura 22: Captura de *Substance Designer*

2.3.9. Substance Painter

Al contrario que *Substance Designer* este programa y el resto de los que veremos a continuación no se centran en la creación de materiales, sino en el pintado de los objetos 3D para generar los diferentes mapas de textura y poder añadirlos al motor correspondiente. *Substance Painter* nos permite trabajar con varios mapas de textura a la vez, por lo que facilita la creación de texturas complejas y detalladas. Además, podemos pintar en tiempo real el modelo, lo que nos permite ver con claridad cómo está quedando el resultado mientras trabajamos.

Este *software* también tiene una gran cantidad de materiales disponibles que nos permitirán ahorrarnos tiempo para la creación de materiales más básicos. Por desgracia, esta herramienta es de pago mensual, pero se puede comprar junto a *Substance Designer*, algo muy positivo a contemplar para el proyecto.

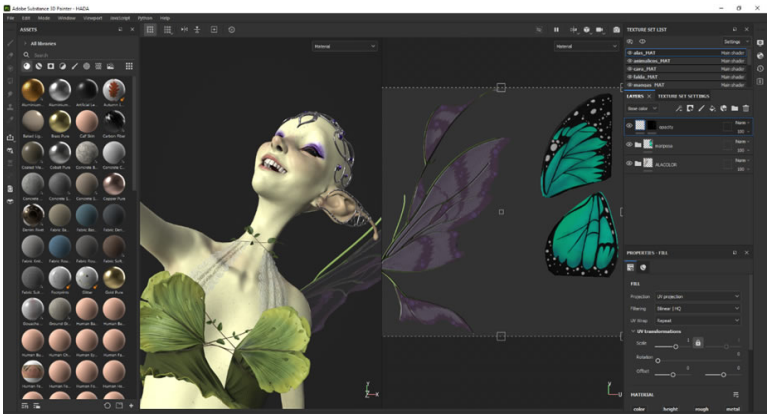


Figura 23: Captura de *Substance Painter*

2.3.10. Adobe Photoshop

Photoshop es una herramienta de *software* muy popular para texturizar modelos 3D debido a su capacidad para crear texturas detalladas y personalizadas, sin embargo, también tiene algunas limitaciones. Debido a su popularidad, hay muchos tutoriales de la comunidad que pueden ayudar con el aprendizaje del *software* para texturizar en 3D, pero esto no le libra de algunas importantes desventajas, como la proyección de texturas que puede ser bastante difícil en algunos modelos más complejos, entre otros problemas que supone trabajar en un *software* pensado para 2D para texturizar objetos 3D.

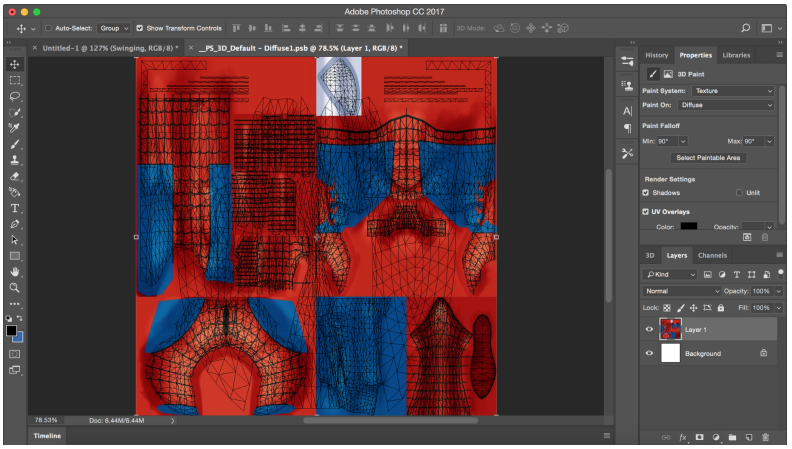


Figura 24: Captura de *Adobe Photoshop*

2.3.11. Mari

Al contrario de *Photoshop*, *Mari* sí que está pensado para texturizar modelos 3D, por lo que ofrece muchas más opciones que este, además es mucho mejor para texturizar modelos 3D muy detallados, cosa que con Photoshop no puede competir. *Mari* es un buen programa para texturizar modelos, pero al tener opciones tan avanzadas puede requerir de una curva de aprendizaje bastante elevada, por lo que si no se tiene conocimiento sobre el mismo, requerirá de un tiempo de aprendizaje para dominarlo.

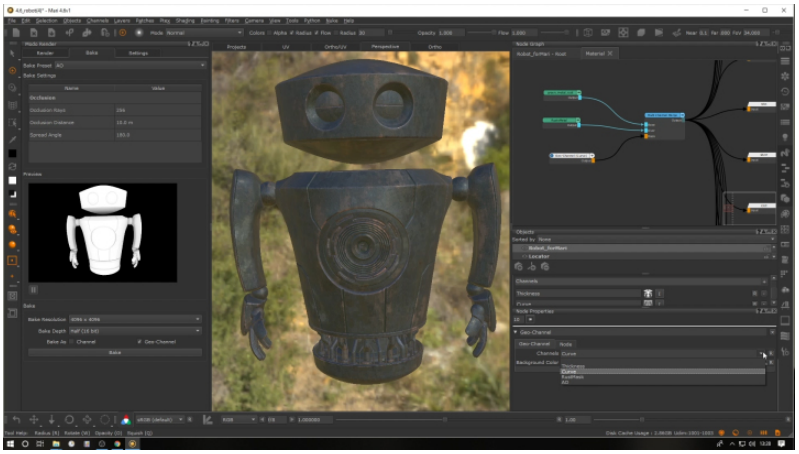


Figura 25: Captura de *Mari*

Para las texturas se ha decidido usar la gama de productos *Substance*, ya que de esta manera tenemos un programa dedicado para crear texturas específicas para modelos y otro para lo que sería la creación de materiales. Además, usar estos dos programas tiene el beneficio de que únicamente requerirá comprar el pack de texturización de *Substance* y tendríamos los dos, al ser de la misma compañía sus interfaces comparten muchas cosas en común, esto es bueno si se va a ir intercalando uno con otro.

2.3.12. Motores 3D

En la selección de motores solo se han añadido dos, *Unity* y *Unreal Engine 4*, se ha considerado que estos dos son los más adecuados para el trabajo a realizar, obviamente hay muchísimos más motores en el mercado, pero no tiene mucho sentido analizar todos los disponibles. Se ha optado por esta pequeña selección donde aparecen los dos más comunes y usados entre los desarrolladores.

2.3.13. Unity

Unity es conocido por su facilidad de uso y su flexibilidad, lo que permite a los desarrolladores crear juegos con una amplia gama de estilos y géneros. Además, *Unity* ofrece una gran cantidad de herramientas y recursos, a destacar, la *Unity Asset Store*, una tienda donde se pueden descargar *assets* gratuitos y de pago con una gran variedad de temas.

Por su popularidad, *Unity* también dispone de una comunidad muy grande de usuarios que hacen tutoriales y es mucho más fácil resolver problemas de esta forma. Como punto negativo estaría su potencia, los siguientes motores que veremos a continuación son capaces de mover entornos de mucha más calidad, por lo que en ese sentido, *Unity* se está quedando atrás.

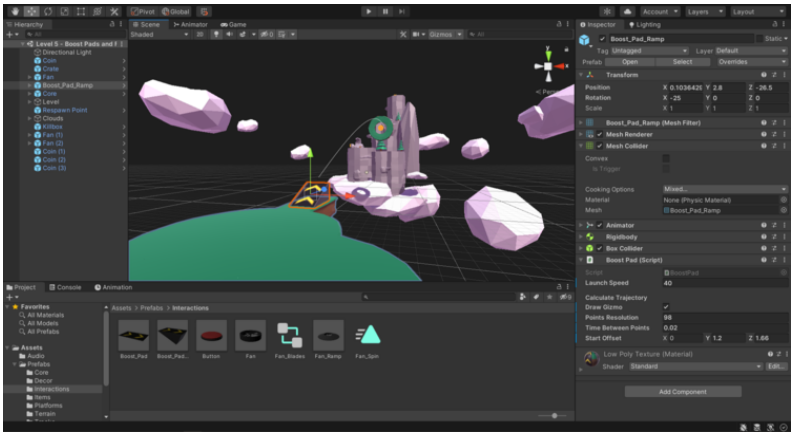


Figura 26: Captura de *Unity*

2.3.14. Unreal Engine 4

Unreal Engine 4 es un motor de juego de código abierto desarrollado por *Epic Games*, es decir, que es gratuito y accesible para cualquier persona con un ordenador que lo soporte. *Unreal* destaca por su capacidad para crear gráficos avanzados y efectos visuales de alta calidad, lo que lo convierte en una grandísima opción si lo que se quiere es crear un entorno y no un videojuego como en este caso. Además, ofrece una amplia gama de herramientas y recursos para ayudar a los desarrolladores a crear sus propios mundos de manera rápida y eficiente, incluyendo un editor visual completo, una biblioteca de *assets* parecida a la *Unity Asset Store*, una comunidad activa de desarrolladores y una extensa documentación disponible para todo el mundo.

Figura 27: Captura de *Unreal Engine 4*

Obviamente, por el título del proyecto, se ha escogido *Unreal Engine 4* como motor en el que presentar el *environment*, además de ser gratuito, tiene un buen balance entre accesibilidad para el usuario y gran potencia gráfica. *Unreal* es usado tanto por el público general como por el usuario común, es una herramienta perfecta que ayudará a mejorar la presentación del *environment*.

3. Gestión del proyecto

Antes de explicar cómo va a ser la gestión de este proyecto lo primero que hay que dejar claro es lo que se quiere llegar a conseguir, en este caso, se trata de la creación de un *environment* 3D en *Unreal Engine 4* por lo que debemos tener en cuenta diferentes cosas.

Lo primero a lo que se debe atender en este tipo de trabajo es a la complejidad del mismo, debido al tiempo limitado para crear el proyecto se debe ser cauteloso con lo que se quiere llegar a conseguir. El objetivo es que el *environment* quede atractivo y se pueda presentar en un porfolio, no debemos exagerar la carga de trabajo, ya que no es viable para una sola persona crear un escenario que por lo general está creado por un grupo de artistas especializados. Así que el primer objetivo para el proyecto será encontrar el balance entre la calidad y la carga de trabajo para obtener el mejor resultado posible al cierre del mismo.

Para que el proyecto sea viable y consistente se han organizado diferentes fases del desarrollo con el objetivo de que se llegue a tiempo al objetivo esperado:

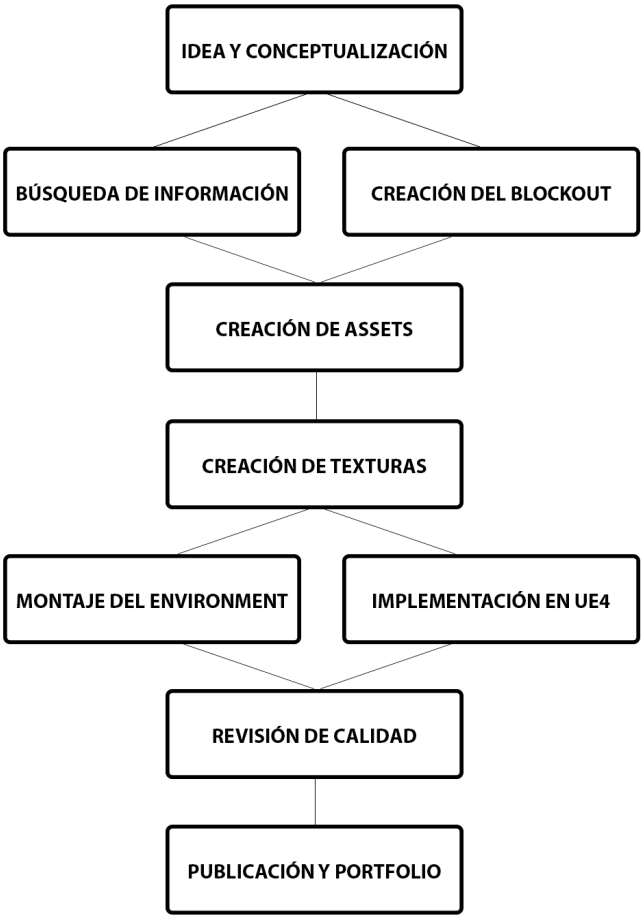


Figura 28: Organigrama del proyecto

En la introducción se ha mencionado que el objetivo es tener un *environment* completo al final del proyecto, para llegar a esto debemos empezar la construcción del mismo desde los cimientos, se crearán los *assets* en orden de prioridad (Ejemplo: crear antes una columna del castillo que un arbusto decorativo). Lo más importante al terminar el proyecto es tener lo que se conceptualizó al inicio del proyecto, es decir, un castillo flotante en el cielo rodeado de varias torres, con su lago y montañas rodeándolo, esto es lo esencial para que el trabajo se considere terminado.

Se ha realizado un calendario que nos ayudará a tener en cuenta ciertas fechas límite para no desviarnos de nuestro objetivo y llegar a finales de junio con el trabajo terminado. El calendario se ha dividido en los cuatro meses de trabajo principal y se ha usado un código de colores para poder ver claramente lo que se debería tener hecho en cada fecha.

Figura 29: Calendario del proyecto

Este calendario se creó al finalizar la primera rúbrica del trabajo, en la segunda rúbrica se han realizado algunos cambios para el correcto desarrollo del proyecto, en esta primera como se puede ver se tenía pensado crear dos iteraciones para los modelos, esto se ha descartado, ya que se decidió que la prioridad era tener montado el *environment* para la rúbrica 2 y a partir de ahí comenzar a sustituir los modelos del *blockout* por los modelos finales.

A pesar de que se hayan producido estos cambios, esto no debería afectar a los costes propuestos más adelante, porque es únicamente un cambio de orden la planificación, no se añaden o se descartan tareas, más allá de ahorrar el tiempo de los *assets* de prueba, cambiados por un *blockout* mejor planificado.

Así quedaría el nuevo calendario:

Figura 30: Calendario del proyecto (2a Iteración)

Para ayudarnos a llegar a las fechas indicadas en el calendario se usará la herramienta *Trello*, que sirve para poder añadir las tareas que queramos en diferentes estados de desarrollo. Esta herramienta será muy útil, ya que este trabajo usará la gestión por sprints, para cada entrega o tarea importante se creará un nuevo sprint en Trello con tres etapas, planificación, en desarrollo y hecho. Con esta herramienta podemos organizar perfectamente las tareas y subtareas de cada sprint y será mucho más fácil seguir el ritmo del proyecto.

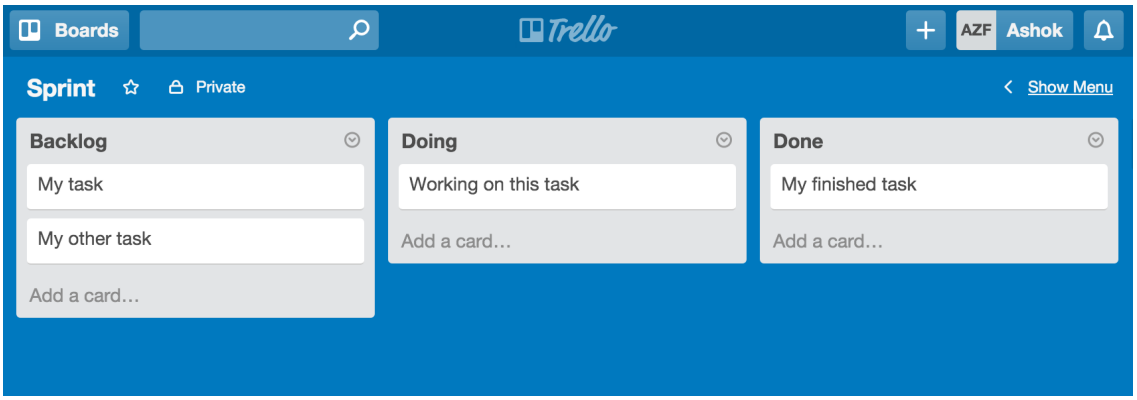


Figura 31: Ejemplo de uso de *Trello*

Además de esta herramienta también se ha realizado un diagrama de Gantt basándonos en las fechas del calendario que nos sirve para saber cuando empezar cada tarea y cuál es nuestro tiempo límite, el siguiente diagrama es una simplificación de las tareas del calendario anteriormente mostrado, pero gracias a los dos elementos podemos realizar un correcto seguimiento de las tareas a lo largo del tiempo.

Figura 32: Diagrama de Gantt del proyecto

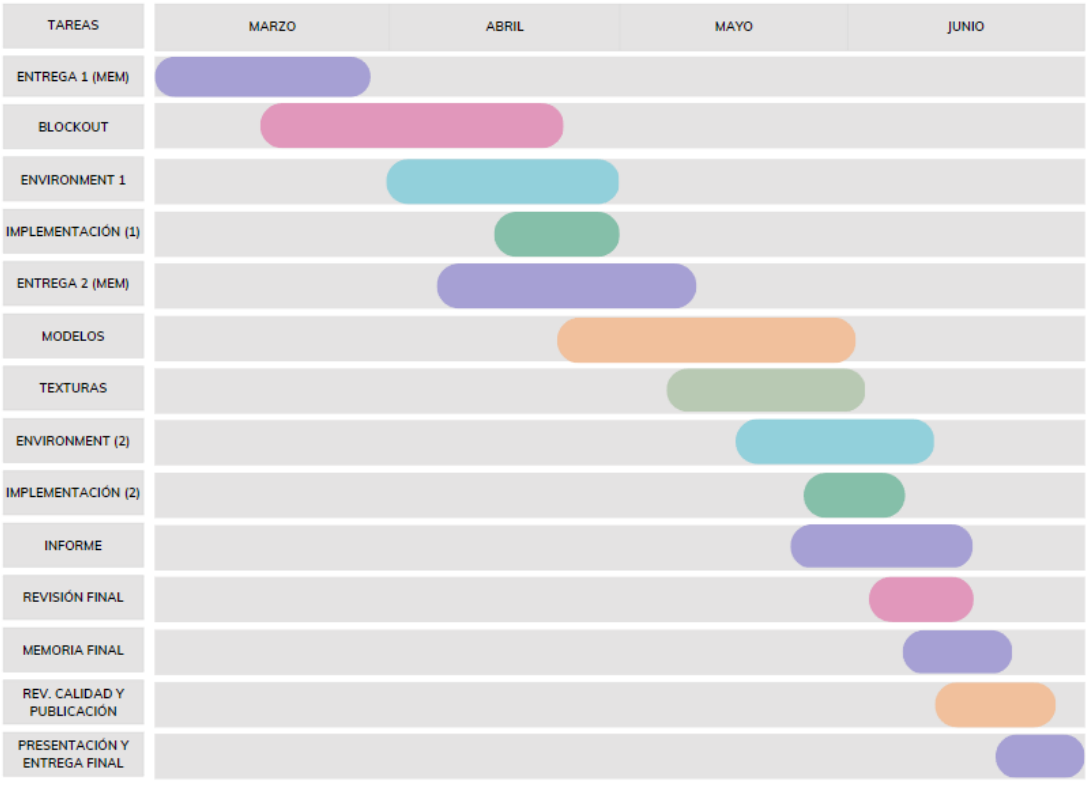


Figura 33: Diagrama de Gantt del proyecto (2a Iteración)

Como se puede ver en la segunda iteración del diagrama de Gantt se han eliminado las dos fases de creación de texturas y modelados para así crear dos únicas fases de larga duración que se juntarán con la creación de la segunda iteración del *environment*, donde se añadirán estos modelos progresivamente directamente en el motor, sustituyendo las piezas del *blockout.*

3.1. DAFO

Esta es la tabla DAFO, que nos dará una idea general sobre las virtudes y problemas que puede tener el proyecto, está dividida en cuatro sectores, Fortalezas, Debilidades, Oportunidades y Amenazas.

	Positivos	**Negativos**
Origen Interno	**Fortalezas** - Trabajo muy visual, muy gráfico al momento de mostrar imágenes del desarrollo - Conocimiento previo de algunos programas de modelado 3D	**Debilidades** - Poca teoría para desarrollar sobre el tema - Difícil innovación artística en este sector - Requiere de habilidad y conocimiento de todas los aspectos del desarrollo de *environments* 3D
Origen Externo	**Oportunidades** - El resultado final puede llegar a ser más espectacular visualmente que en algún otro proyecto, ya que es más completo - Puede abrir varias puertas a futuro gracias a la polivalencia en diferentes ámbitos artísticos	**Amenazas** - El resultado final puede no destacar en ningún área en concreto por la cantidad de trabajo - Tiempo limitado para terminarlo, por lo que puede no llegarse al nivel de calidad deseado

Tabla 1: DAFO

3.2. Riesgos y plan de contingencias

Los posibles riesgos identificados de este proyecto y sus correspondientes soluciones son las siguientes, ordenadas de menor a mayor importancia:

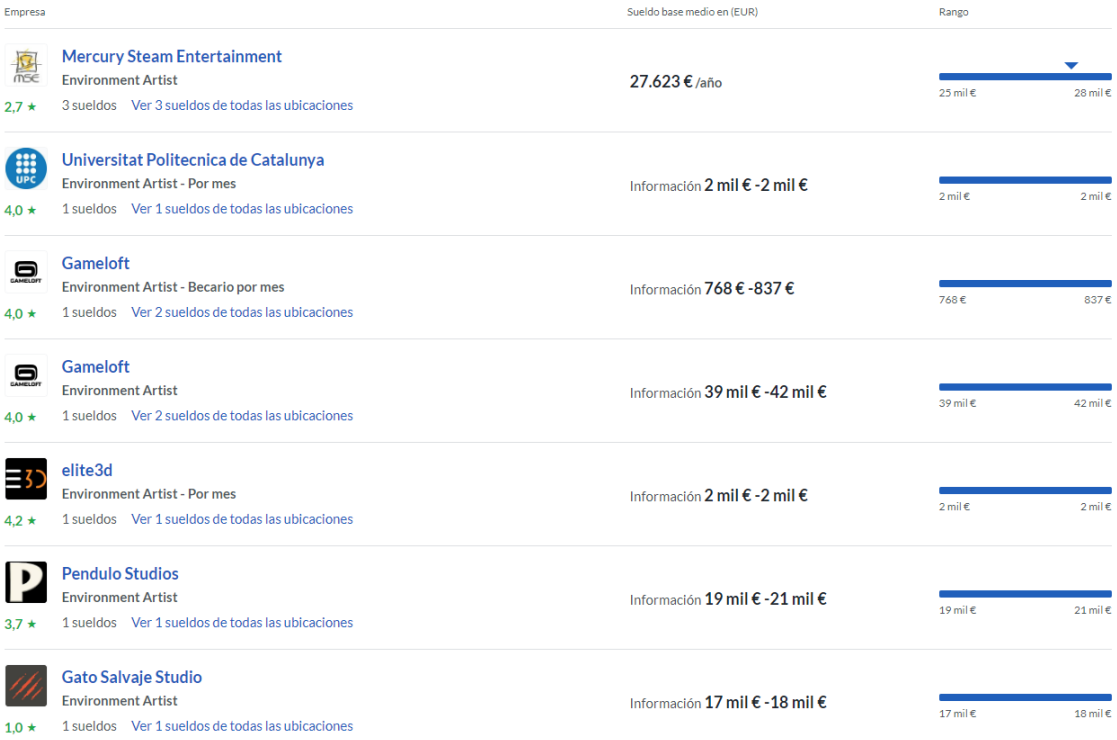
Riesgo	Solución
No lograr terminar a tiempo el trabajo	Planificación anticipada del trabajo a seguir gracias a una metodología y gestión adaptadas al tiempo establecido.
Falta de conocimiento de Unreal Engine 4	El creador de este trabajo no ha usado nunca Unreal Engine 4, esto podría ser un gran problema para la creación del *environment* por el proceso de aprendizaje. Para evitar este problema se han seguido una serie de tutoriales de aprendizaje sobre el motor para que no sea necesario aprender durante la producción.
Intentar abarcar demasiado	Es posible que al inicio del trabajo se aspire a hacer un *environment* que no es posible para una sola persona con el tiempo establecido, para solucionar esto se debe tener claro qué es lo que queremos hacer y visualizar la viabilidad del proyecto, para llegar a tiempo en este caso se ha optado por un estilo visual estilizado como el que se mostraba en la introducción
Environment poco variado	Al hacer el *environment* podríamos darnos cuenta de que para una persona hacer tantos *assets* puede llegar a ser un problema por el tiempo establecido. Se puede evitar esto camuflando el problema con diferentes técnicas como por ejemplo el *Vertex Paint* que nos dará mucha más variedad en las texturas.

Tabla 2: Tabla de Riesgos-Soluciones

3.3. Análisis inicial de costes

Para hacer el análisis inicial de costes se ha seguido la siguiente estructura: sueldos, *software* y *hardware*.

En la parte de sueldos se ha decidido incluir únicamente el salario de un *Environment Artist* en España, que será el rol que desempeñará el creador de este TFG durante estos cuatro meses. Para calcular el salario mensual se ha usado la información disponible de las empresas con este puesto de trabajo en España.



Empresa	Sueldo base medio en (EUR)	Rango
Mercury Steam Entertainment Environment Artist 2,7 ★ 3 sueldos Ver 3 sueldos de todas las ubicaciones	27.623 € /año	25 mil € – 28 mil €
Universitat Politecnica de Catalunya Environment Artist - Por mes 4,0 ★ 1 sueldos Ver 1 sueldos de todas las ubicaciones	Información 2 mil € -2 mil €	2 mil € – 2 mil €
Gameloft Environment Artist - Becario por mes 4,0 ★ 1 sueldos Ver 2 sueldos de todas las ubicaciones	Información 768 € -837 €	768 € – 837 €
Gameloft Environment Artist 4,0 ★ 1 sueldos Ver 2 sueldos de todas las ubicaciones	Información 39 mil € -42 mil €	39 mil € – 42 mil €
elite3d Environment Artist - Por mes 4,2 ★ 1 sueldos Ver 1 sueldos de todas las ubicaciones	Información 2 mil € -2 mil €	2 mil € – 2 mil €
Pendulo Studios Environment Artist 3,7 ★ 1 sueldos Ver 1 sueldos de todas las ubicaciones	Información 19 mil € -21 mil €	19 mil € – 21 mil €
Gato Salvaje Studio Environment Artist 1,0 ★ 1 sueldos Ver 1 sueldos de todas las ubicaciones	Información 17 mil € -18 mil €	17 mil € – 18 mil €

Figura 34: Sueldos de *Environment Artist* en diferentes empresas

Teniendo en cuenta estos datos y considerando que se va a trabajar en todas las etapas del desarrollo 3D, se determinó que aproximadamente el sueldo sería unos 2083,33 €.

Para la parte de *software* se ha buscado información para cada programa específico, en el caso de *Unreal Engine*, es gratis hasta que el desarrollador gane más de 3000 € con su obra, a partir de aquí *Epic* se queda con un 5% de los beneficios. *Substance* tiene un precio mensual de su paquete de programas para texturizar por 19,35 € en España. *Maya* y *Zbrush* usan un modelo de negocio simple mensual en el que hay que pagar 279 € y 32,96 € respectivamente.

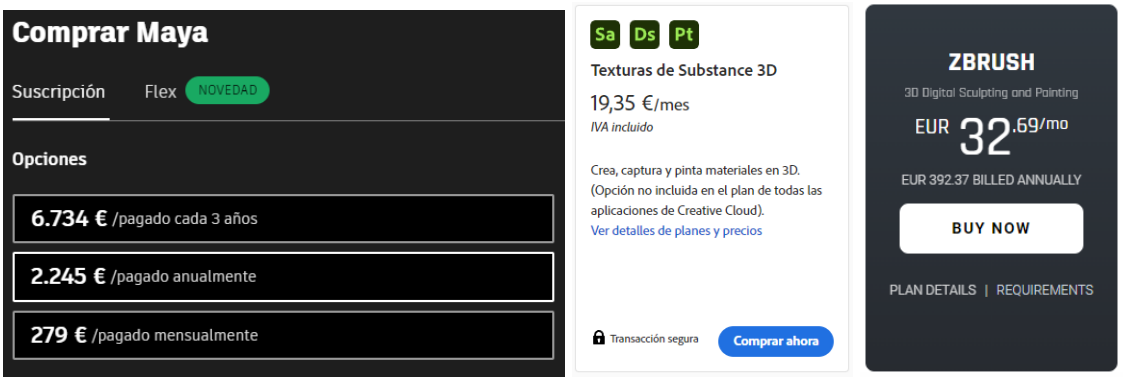


Figura 35: Precios de los programas a utilizar

Para el hardware se han tenido en cuenta cuatro elementos principales, el primero es la oficina, en este caso se ha considerado que el precio de oficina será nulo, ya que se trabajará desde casa. Lo siguiente sería el PC y sus periféricos, teniendo en cuenta el tipo de programas de gran potencia que se van a usar y la comodidad para trabajar se consideró que el PC debía costar unos 2500 € teniendo en cuenta el precio actual de elementos como las tarjetas gráficas, en el caso de los periféricos se consideraron dos monitores, teclado y ratón y una tableta gráfica, muy importante para modelar y hacer las texturas, se estimó que aproximadamente serían unos 1500 € en total, todos estos gastos serán solo en el primer mes, ya que no son pagos mensuales.

Por último debemos tener en cuenta la energía requerida, teniendo en cuenta el tipo de ordenador con el que se va a trabajar y su gasto por hora, se determinó que aproximadamente se gastaría 0.15 €/kWh, aproximadamente se dedicarán unas 3 horas al día en lo que respecta al desarrollo con estos programas por lo que podemos intuir que el coste energético será de unos 12,55 € al mes.

	Marzo	Abril	Mayo	Junio
Sueldo Environment Artist	2.083,33 €	2.083,33 €	2.083,33 €	2.083,33 €
TOTAL SUELDO	2.083,33 €	2.083,33 €	2.083,33 €	2.083,33 €
Unreal Engine 4	0,00 €	0,00 €	0,00 €	0,00 €
Substance Painter/Designer	19,35 €	19,35 €	19,35 €	19,35 €
Maya	279,00 €	279,00 €	279,00 €	279,00 €
ZBrush	32,69 €	32,69 €	32,69 €	32,69 €
TOTAL SOFTWARE	331,04 €	331,04 €	331,04 €	331,04 €
Oficina	0,00 €	0,00 €	0,00 €	0,00 €
Energía	12,55 €	12,55 €	12,55 €	12,55 €
PC	2.500,00 €	0,00 €	0,00 €	0,00 €
Periféricos	1.500,00 €	0,00 €	0,00 €	0,00 €
TOTAL HARDWARE	4.012,55 €	12,55 €	12,55 €	12,55 €
COSTES MENSUALES	6.426,92 €	2.426,92 €	2.426,92 €	2.426,92 €
COSTES TOTALES	13.707,68 €			

Tabla 3: Costes del proyecto

4. Metodología

Para este proyecto se usará una metodología basándonos en la Preproducción, Producción y Postproducción, probablemente sea la más adecuada para este tipo de proyecto dividido en muchas fases para llegar al objetivo final.

Previamente en el apartado de gestión ya se pudieron ver los pasos a seguir del proyecto, a continuación se muestra cómo se dividirá en las tres fases mencionadas:

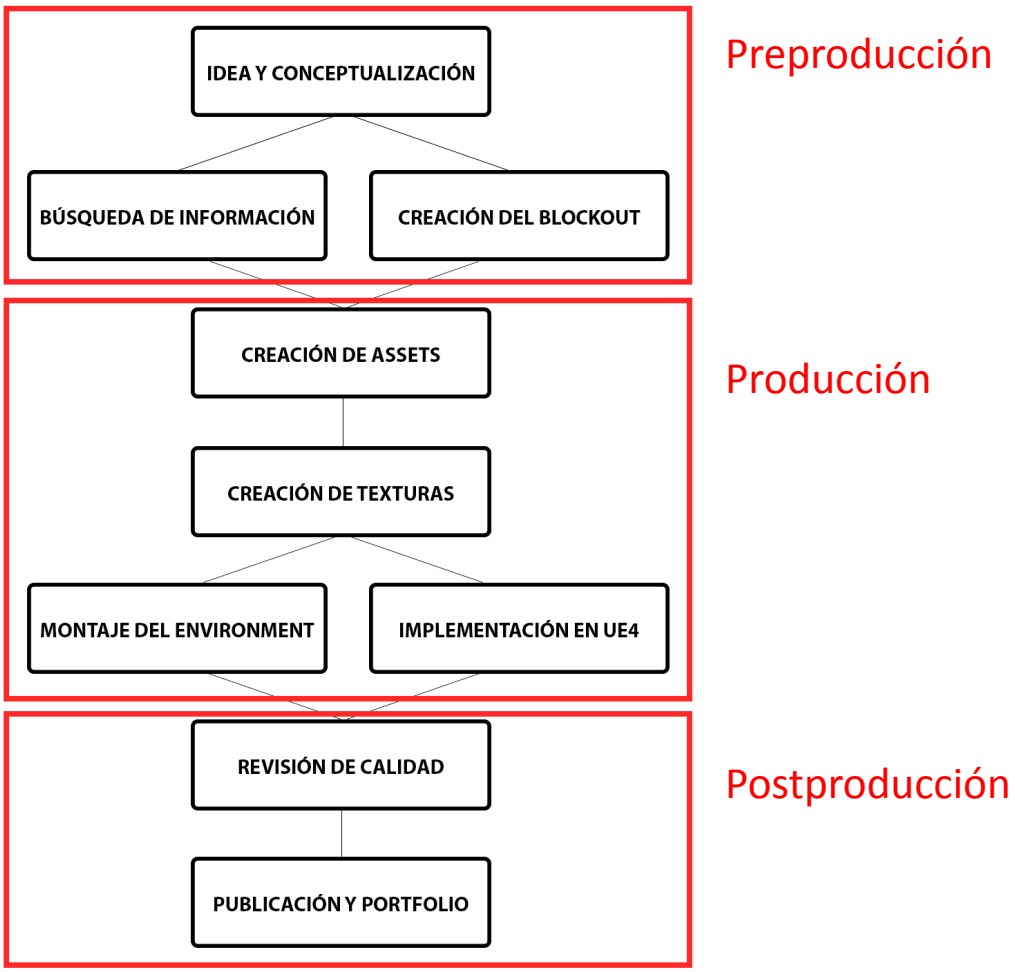


Figura 36: Metodología de organigrama

Preproducción

Idea y Conceptualización

La primera fase del proyecto consiste en la idea para el mismo, debemos responder a la pregunta ¿Qué queremos hacer?, esta fase determinará cómo será nuestro proyecto al final del mismo. También es importante tener en cuenta desde este mismo momento cuál será el tiempo estimado a dedicar para que el proyecto pueda estar listo en el límite de tiempo.

La conceptualización también tiene un punto importante aquí, deberemos buscar referencias que nos resulten atractivas para inspirar nuestro proyecto.

Algunas referencias que se han seleccionado en este caso:

Figura 37: 1a Referencia

Figura 38: 2a Referencia

Figura 39: 3a Referencia

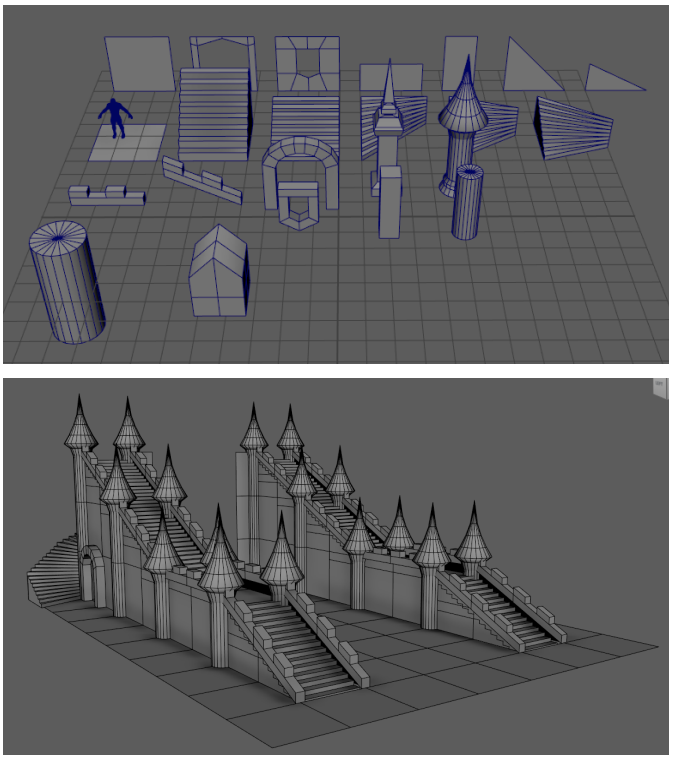
Búsqueda de información

Esta fase consiste en informarse sobre todo lo necesario para completar el proyecto. Nuestro objetivo es tener todas las herramientas para lograr hacer lo que queremos, estas van desde aprender a usar los programas/motores con los que queramos trabajar, informarse de técnicas de modelado y texturizado para mejorar lo que ya sabemos, hasta encontrar técnicas específicas para lo que queremos hacer como la construcción modular.

Creación del Blockout

En este punto ya empezamos con la creación 3D, nuestro objetivo es crear las piezas básicas que compondrán el *environment*, estas piezas se realizarán en este caso con el programa Maya, que nos permitirá usar geometría básica para construir nuestra *build* modular y empezar con las primeras piezas para crear el castillo.

Figura 40: Programa *Maya*



Figuras 41 y 42: Imágenes del *blockout* en desarrollo

Producción

Creación de Assets / Texturas

Esta fase consiste en crear los *assets* que queremos basándonos en el *blockout* que hemos hecho para dar vida al entorno. Para crear el modelo *high poly* usaremos el programa *ZBrush*, luego, para crear las UV que usaremos para texturizar más adelante usaremos Maya, igual que para el *blockout*, si el poligonaje es muy elevado habrá que hacer también una retopología del *high poly*, también se usará *Maya* durante este proceso.

Figura 43: Programa *ZBrush*

Una vez tengamos los modelos con sus UV pasaremos a hacer lo que serán las texturas, para ello usaremos el programa *Substance Painter* en el caso de que queramos hacer texturas detalladas para un objeto en concreto. Se usará *Substance Designer* en el caso de que la textura que queramos hacer no sea para un objeto en concreto sino para uso general (paredes, suelos, tierra, etc.).

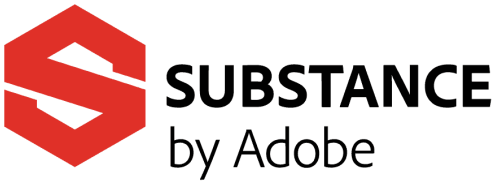


Figura 44: Paquete de programas *Substance*

Montaje del Environment / Implementación en UE4

Respecto al montaje, este paso consiste en crear el entorno donde vamos a poner el castillo en *Unreal Engine 4,* usaremos las herramientas que nos da el motor para la creación de superficies y líquidos, ya que la idea es que el castillo flote en una isla sobre el agua. Obviamente, también se implementará el castillo con todas las piezas ya acabadas sobre la superficie que se haya creado, así como las texturas necesarias para cada elemento, esto también sería la parte de implementación.

En esta fase también será importante el añadido de temas como la iluminación, partículas, efectos, vegetación y varios detalles que nos puede dar el motor para lucir mucho más el trabajo realizado.

Figura 45: Motor *Unreal Engine 4*

Postproducción

Revisión de calidad

En este apartado se reevaluará el trabajo realizado una vez lo tengamos todo terminado en fase de producción, en este punto ya deberíamos tener todos los *assets* que queríamos hechos y montados en el motor, así como todo el *environment* totalmente terminado. El objetivo en este momento es identificar qué aspectos del acabado no dan la talla, encontrar la solución necesaria o añadir pequeños detalles para que el acabado general se vea de la calidad que queremos.

Uno de los problemas que podría darse en este punto puede ser la repetición de texturas o modelos, en este caso se podría solucionar añadiendo nuevas texturas y modelos o simplemente modificando los mismos si no se tuviese tanto tiempo.

Publicación y Porfolio

Por último se deberá publicar nuestro trabajo en internet, esto puede parecer fácil a primera vista, pero el objetivo aquí será encontrar la mejor forma de mostrarlo profesionalmente y puede resultar difícil mostrar adecuadamente todo el trabajo en el que hemos invertido tanto tiempo.

Gracias a la *ArtStation* actualmente los artistas tienen mejores herramientas para mostrar sus trabajos, esta será la web que usemos para presentar el nuestro, en esta web miles de artistas alrededor del mundo enseñan sus proyectos, es la mejor forma actualmente de promocionar tu trabajo por internet, ya que es muy fácil añadir tu link de *ArtStation* a tu porfolio. Se aprenderá a como usar *ArtStation* y las técnicas para que nuestro proyecto pueda ser visto por el máximo de personas posibles, como por ejemplo presentar el proyecto en diferentes etapas de desarrollo, especificando los programas que usemos, entre otros consejos.

Figura 46: Web *ArtStation*

5. Desarrollo del proyecto

Siguiendo los pasos especificados en la gestión del proyecto y la metodología, se han descrito los pasos concretos que se han seguido, así como los problemas encontrados en cada caso. Este punto del proyecto sirve como un diario de desarrollo distribuido en diferentes puntos referentes al anterior apartado de Metodología.

5.1. Idea y Conceptualización

Antes de empezar a desarrollar el proyecto debemos tener claro que se quiere queremos hacer exactamente, para ello debemos conceptualizar nuestras ideas para más adelante darles vida. De momento no hablaremos de la selección de programas para cada tarea, lo importante de esta fase es elegir qué queremos hacer y hasta donde se quiere llegar, esto último es muy importante sobre todo si se tiene un límite de tiempo como es en este caso. En este punto tenemos dos opciones, o dibujar nuestro propio *concept art* y darle vida más adelante en 3D o buscamos referencias y hacemos el proyecto inspirándose en las mismas. En este caso se ha optado por la segunda opción, ya que se consideró que si no se tienen altas capacidades de dibujo lo mejor es optar por basarse en *concepts* de artistas profesionales.

En el apartado de metodología se mostraron algunos *concepts* consultados para la creación del *environment*, pero en este apartado de desarrollo se ha decidido mostrar dos de los más importantes que han servido como inspiración en este caso, son los siguientes:

Figura 47: Referencia para la idea

Figura 48: Referencia arquitectónica

La primera referencia sirvió para tener la idea de las cadenas, en ese caso las cadenas provienen del suelo y parece que intentan mantener al castillo cerca del suelo, pero en el caso del *environment* que se va a realizar el concepto de las cadenas es otro. Habrá cinco pequeñas islas con sus respectivos torreones que mantendrán flotando a la gran isla central que es donde se encontrará el gran castillo. La segunda referencia ha servido sobre todo a nivel arquitectónico, es la referencia que se eligió para algunos aspectos de la arquitectura, como el aspecto de las torres y sus tejados que se verán reflejados desde el *blockout*, aunque también servirán como inspiración para los detalles de los *assets* que se irán viendo más adelante del desarrollo.

Con estas dos ideas en mente se decidió crear una pequeña referencia en 3D de lo que sería en un futuro la estructura del entorno, esta prueba se hizo en el programa *Autodesk Maya,* aunque no es muy relevante el programa usado en este caso, ya que, como he mencionado, es solo una prueba de geometría muy básica para tener una referencia de lo que se quiere conseguir en este proyecto. Tras varias pruebas se acabó teniendo lo que se muestra a continuación.

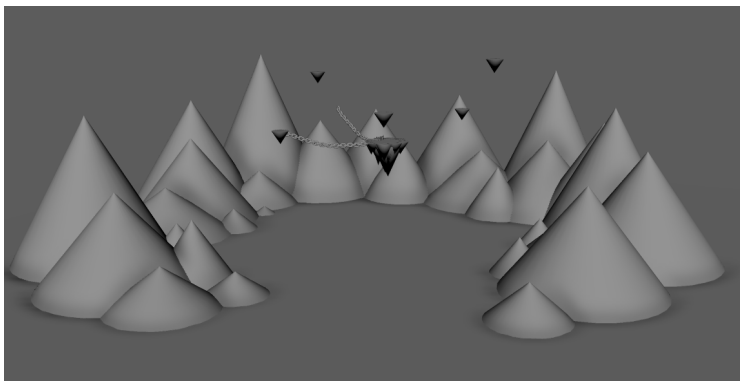


Figura 49: Concept del *environment*

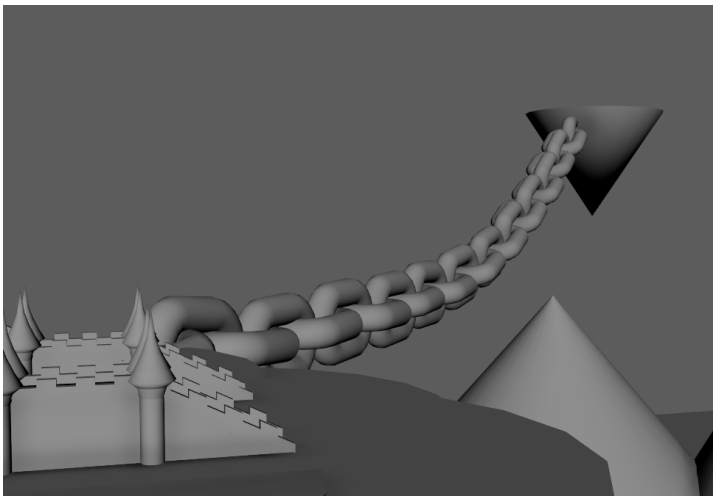


Figura 50: Concept de las islas

Como podemos ver en la primera imagen, a nivel composicional hay unas montañas rodeando la escena, lo que le dará más empaque al escenario en general, ya que si no quedaría muy vació. El plano de abajo debe considerarse que será un gran océano, esto se hará directamente en el motor, *Unreal Engine 4* tiene algunas herramientas que facilitarán mucho este trabajo.

En la segunda imagen se muestra más de cerca una de las grandes cadenas que unirán la gran isla con las pequeñas torres mencionadas anteriormente. El tamaño de las cadenas es intencionado, como se puede ver en comparación con las escaleras del castillo, son exageradamente grandes, igual que en la referencia seleccionada, tiene bastante sentido, ya que deben estar unidas a grandes trozos de tierra de gran peso, además hacerlas de este tamaño le darán al proyecto un punto más distintivo.

Con las referencias y este primer esbozo de lo que se quiere hacer, podemos comenzar a hacernos las preguntas de como haremos cada cosa del *environment* y valorar las opciones que tenemos en cada caso, por lo que se puede pasar a la siguiente fase del proyecto.

5.2. Búsqueda de Información

En el apartado de metodología y en el estado del arte ya se han mencionado las herramientas que van a usarse para cada parte de desarrollo, este apartado servirá para hacer un repaso de las elecciones tomadas y el porqué de cada una. Obviamente, dentro de este apartado también entraría toda la información sobre el desarrollo de *environments* 3D que se ha obtenido gracias al estado del arte y a investigación independiente sobre el tema.

Empezaremos por los programas de modelado 3D, en el caso de la creación del *blockout* se usará *Autodesk Maya*, ya se han mencionado otros programas 3D como *3ds Max* o *Blender*, pero se justificó esta elección basándose en la experiencia del creador de este proyecto con el programa, este programa se utilizará, para el *blockout* así como para la retopología de algunos *assets* a los que se le bajará el poligonaje, otro uso muy importante que se le dará a este *software* es el desplegado de UVs, podremos alterar las UVs de nuestros modelos para más adelante trabajar correctamente en la texturización.

Otro programa muy importante de modelado 3D que se usará en el trabajo es *Zbrush*, en este caso se usará para la creación de modelos mucho más complejos que no se podrían hacer en *Maya* con las herramientas de modelado que ofrece. *Zbrush* es una herramienta ideal para esta tarea, se habían llegado a contemplar otros programas como *Blender* o *Cinema 4D,* pero se consideró que *Zbrush* ofrecía mejores herramientas, además de que ya había sido utilizada por el creador del trabajo, por lo que es tiempo de aprendizaje que se puede ahorrar en este caso.

Para la texturización no había demasiadas dudas, el paquete de programas *Substance* parece la única opción correcta, en este caso, se llegó a explorar alguna otra opción como *Photoshop* o *Mari*, pero el ecosistema *Substance* pasaba muy por encima de estas opciones gracias a la facilidad y conectividad que tienen los *softwares* entre ellos. Con *Substance Designer* se crearán los materiales, más adelante se hablará de cómo se han creado los materiales estilizados, el software usa *blueprints* y gracias a sus miles de combinaciones y personalización posible, se pueden crear materiales de todo tipo que darán mucha más vida a los modelados.

Substance Designer se combina a la perfección con *Substance Painter*, en este software se pueden usar directamente los materiales creados en *Substance Designer* así como sus propiedades y aplicarlas al modelo para conseguir el resultado deseado en cada caso. Trabajar con estos dos programas es muy cómodo y nos ahorra problemas de compatibilidad que podría haber de usar otros, por lo que no es una elección muy difícil.

El motor también se ha mencionado anteriormente, se ha hablado de algunos como *Unity*, pero *Unreal Engine 4* es el escogido en este caso. *Unreal* es el equilibrio perfecto entre potencia y facilidad para el usuario, es un motor muy intuitivo para el desarrollador, además de potente a nivel técnico, de ahí que actualmente es el motor más usado por las compañías y estudios indies ganando cada vez más y más terreno a Unity.

Se contempló usar *Unreal Engine 5,* pero hay dos motivos relevantes por la cual se descartó, el primer motivo es el poco tiempo que lleva en el mercado, esto dificulta la búsqueda de alguna información relevante así como problemas que pueda haber por estar en una versión de prueba, el segundo motivo y más importante es que el motor necesita mucha más capacidad de procesamiento y el ordenador del que se disponía marcado además en el presupuesto no llega a mover el motor de una forma tan fluida como *Unreal Engine 4*. Este problema podría intensificarse mucho más en la recta final cuando empiece a haber una carga alta de polígonos y podría generar un problema serio incapacitando el poder terminar el proyecto.

Por último, es importante destacar donde se va a publicar el resultado, no es necesario hacer demasiada investigación en este aspecto, para publicar este tipo de proyecto la página ideal es *ArtStation*, en ella hay muchos artistas digitales mostrando su trabajo y tiene su propia red social que permite contactar con otros artistas o empresas que buscan empleados, por lo que es la mejor manera de darse a conocer. También se comentará cómo darse a conocer por otras redes sociales importantes como *Twitter* o *LinkedIn*, esenciales para encontrar trabajo en esta disciplina.

Una vez tenemos claro todas las herramientas que vamos a usar y tenemos en mente el conocimiento necesario para el desarrollo de *environments* es el momento de comenzar el desarrollo real del proyecto.

5.3. Creación del Blockout

La primera gran fase del desarrollo consiste en crear el *blockout* del *environment*, se contemplaron dos posibilidades para este caso, la primera era usar la técnica del *whiteboxing* y *greyboxing*. La técnica del *whiteboxing* consiste en tener una estructura a base a figuras geométricas básicas como se ve en la imagen de abajo, la idea del *whiteboxing* es tener una estructura básica de lo que va a ser el *environment*, podríamos decir que es algo similar a lo que se ha hecho en el apartado de Idea y Conceptualización, donde se crea una estructura básica para ver cómo funciona visualmente antes de comenzar con el desarrollo de *assets*.

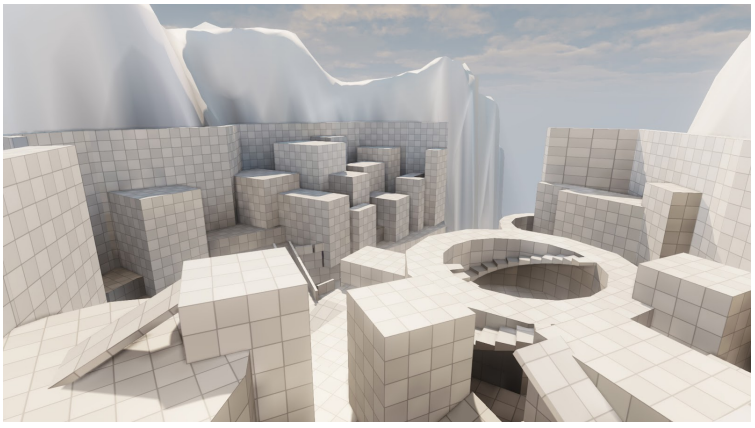


Figura 51: Ejemplo de *whiteboxing*

La siguiente fase sería el *greyboxing*, en este caso se usan modelos lo suficientemente detallados como para diferenciarlos entre ellos, en esta fase ya se diferencian, puertas, casas, árboles, etc. En esta fase del desarrollo ya se debería tener del todo claro cómo será el escenario definitivo. Normalmente, esta metodología se suele usar más en videojuegos, ya que los escenarios llevan un tiempo de desarrollo y sería un problema crear un escenario acabado antes de hacer los test y tener que cambiarlo.

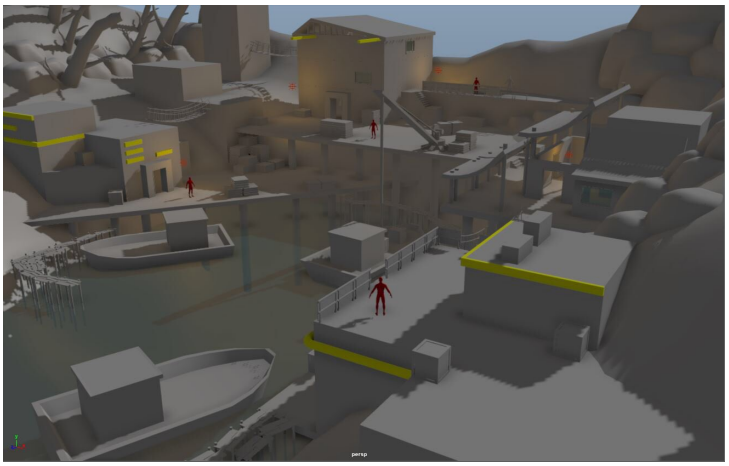


Figura 52: Ejemplo de *greyboxing*

Ya que en este caso el escenario es meramente artístico y no habrá *gameplay* se decidió usar la segunda técnica que es básicamente crear una build modular. Consiste básicamente en crear piezas que formarán la estructura completa de un *environment*, el nivel de detalle depende en cada caso, en este caso se ha elegido usar un nivel de detalle cercano a lo que sería el *greyboxing* visto anteriormente.

Como se ha mencionado anteriormente el programa utilizado para la creación del *blockout* es *Autodesk Maya,* cuando se ha empezado a desarrollar el *blockout* se han encontrado diferentes problemas, la escala, los pivotes de las piezas y la *grid* para montar el escenario. Para la escala se ha usado el modelo proporcionado por *Epic Games*, *Unreal Mannequin*, este modelo 3D nos dará una referencia perfecta para la escala en comparación con un ser humano promedio.

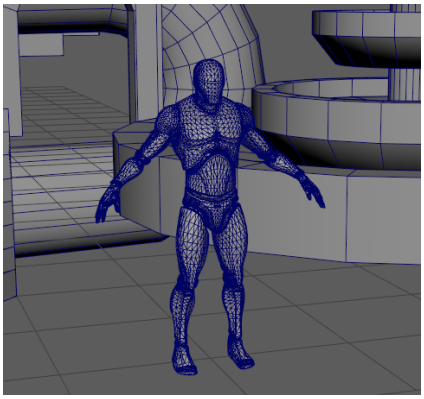


Figura 53: *Unreal Mannequin*

Además, desde *Maya* deberemos configurar la escala en metros para tener una referencia con la *grid* del programa, para ello habrá que cambiar las preferencias del programa a metros para poder trabajar con la escala adecuada desde un principio, ya que cambiarla después puede dar diferentes problemas. Además de estos ajustes es importante preparar la *grid* para que el montaje sea lo más sencillo posible, para ello hay que seleccionar el primer imán en la parte superior de *Maya* que nos anclará los pivotes de las piezas del *blockout* a la *grid*. A parte, de esto para poder tener una modular *build* del todo funcional faltaría una parte esencial que es modificar los pivotes de las piezas para que encajen entre sí, es importante que las paredes y los suelos tengan las mismas proporciones, por ejemplo en este caso se han hecho en metros cuadrados y los pivotes se han colocado en las esquinas para unir con facilidad las piezas, en el caso de columnas u otros elementos varía la posición del pivote para que encaje lo mejor posible en el montaje del castillo.

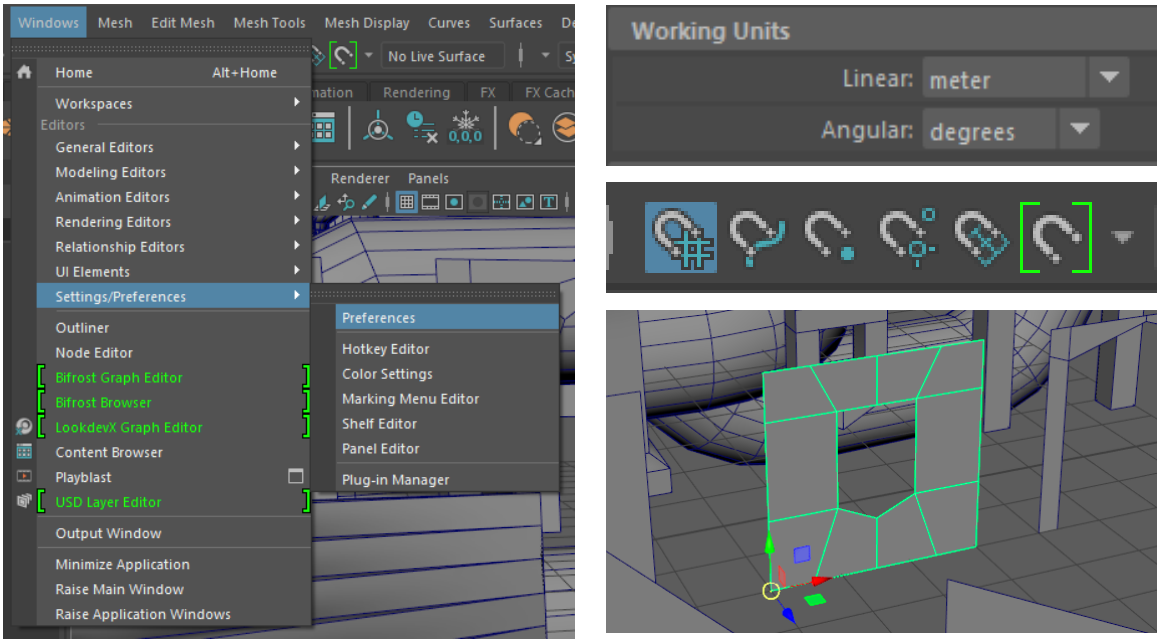


Figura 54: Capturas de *Maya*

Una vez tenemos todas las piezas que necesitamos, con más o menos detalle depende lo que hayamos decidido para este *blockout*, se organizan todas ellas para ver con claridad nuestras herramientas para construir el *environment*. A continuación, el resultado de esta primera modular *build*.

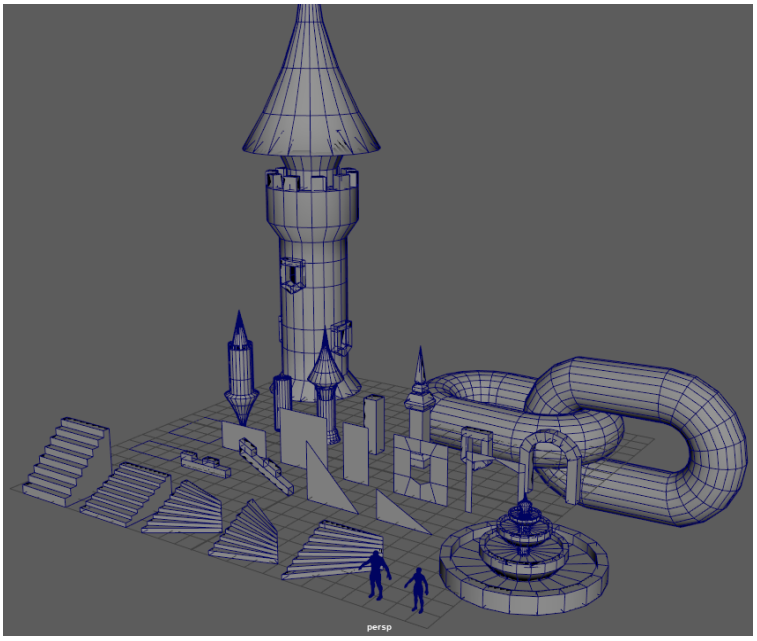


Figura 55: *Modular Build* terminada

Lo único que nos faltaría es montar lo que queramos con nuestras piezas, es posible que durante el montaje te des cuenta de que faltaría alguna pieza para que el resultado sea el deseado, pero siempre se pueden crear nuevas piezas para seguir construyendo el castillo, otra técnica muy útil es re-escalar los elementos ya creados obteniendo unos más grandes que parezcan nuevos, esto nos permitirá reutilizar *assets* en el futuro sin que el resultado sea muy repetitivo visualmente. Tras un buen rato montando el castillo, el resultado quedó de la siguiente manera.

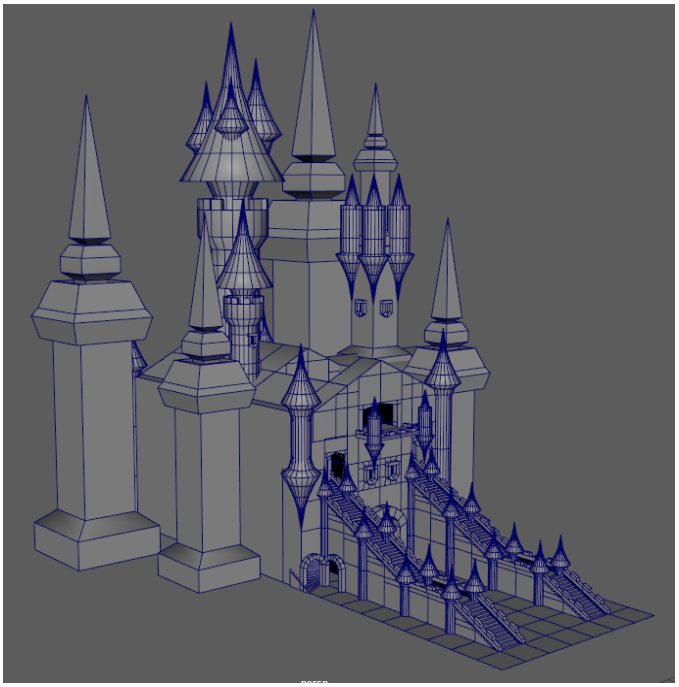


Figura 56: *Blockout* del castillo terminado

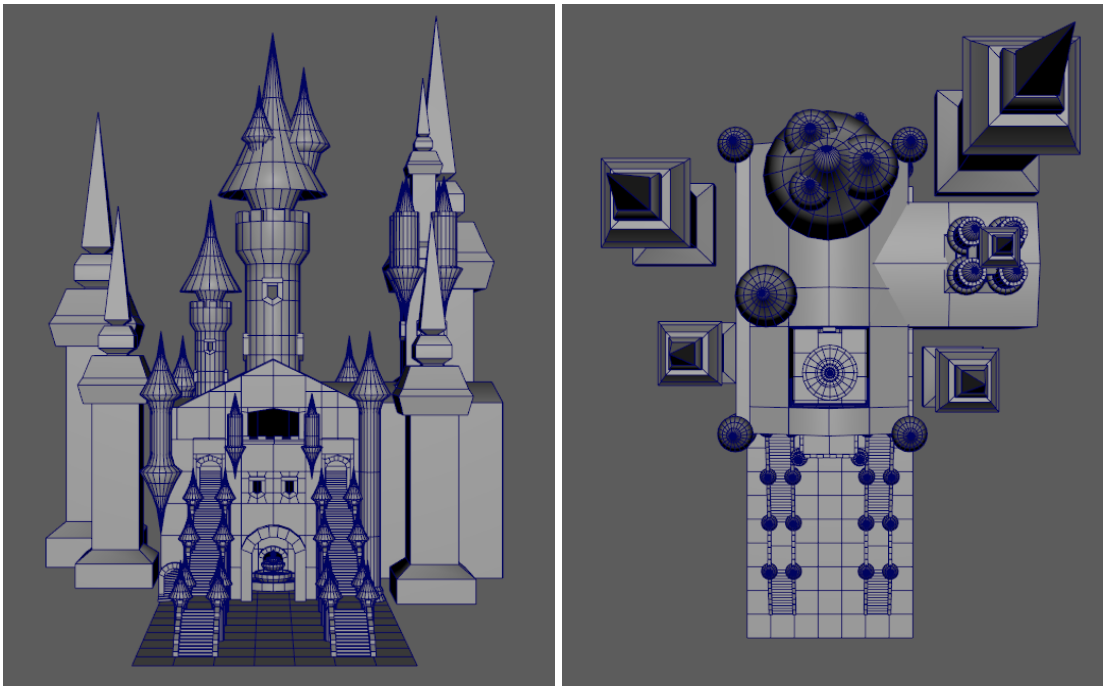


Figura 57 y 58: Vista frontal/cenital de la estructura

Una vez con esto terminado y las estructura del *environment* ya definida en el *concept*, podemos pasar a la siguiente etapa, donde vamos a crear los modelos que sustituyen a los del *blockout* que tenemos en la modular *build*.

5.4. Creación de Modelos

Para la creación de modelos en alta definición se utilizará la herramienta *ZBrush*, esta herramienta ofrece muchísimas posibilidades para crear modelos 3D, aunque aprender a usarla puede suponer una curva de dificultad algo más elevada que otros programas 3D por la inmensa cantidad de opciones que se nos ofrece. Por esta razón no se ahondará demasiado en cómo funciona el programa sino que se explicará la metodología de trabajo que se ha seguido en cada caso.

El primer modelo que se ha creado es el trozo de tierra que sostendrá al castillo, para este modelo se usaron primitivas de conos para así crear lo que será la base para el modelo como se puede ver en la siguiente imagen.

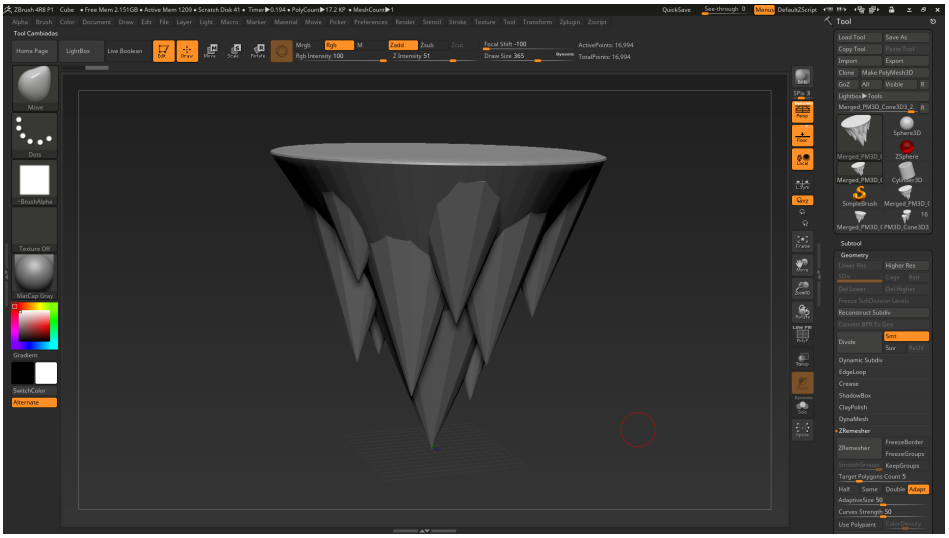


Figura 59: Base del castillo con primitivas

La siguiente fase es unir estas primitivas en una única *mesh* y hacer todas las divisiones necesarias para obtener el poligonaje que deseemos. Una vez hecho esto, le daremos detalle con los pinceles disponibles, hay una gran cantidad de pinceles y técnicas que pueden usarse en *ZBrush*, explicar todas y cada una de ellas sería imposible, por lo que en este trabajo se explicará cómo tener una maya bien hecha para no tener problemas con las UVs más adelante.

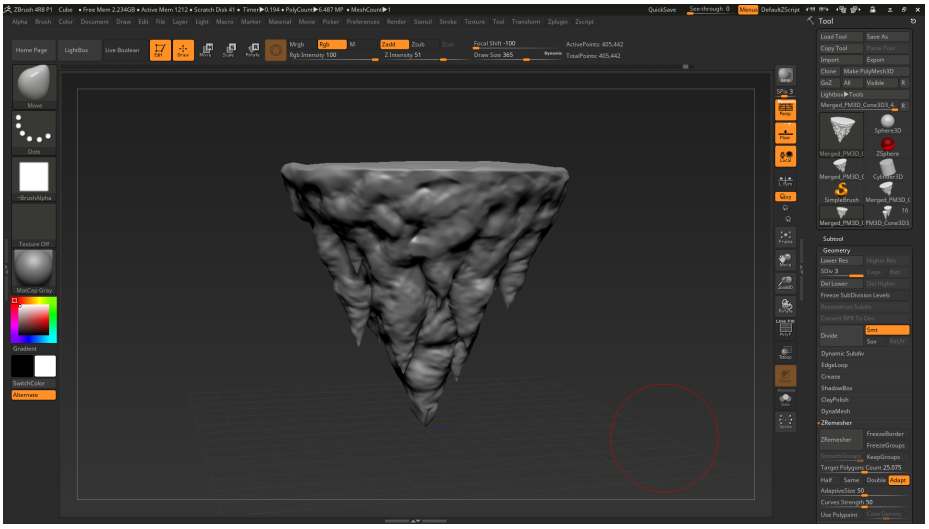


Figura 60: Base del castillo *high poly*

Una vez tenemos el modelo con la calidad deseada el siguiente paso será arreglar la *mesh* del modelo, al deformar el modelo también deformamos la *mesh,* esto puede generar irregularidades en la misma. Para arreglar estos problemas usaremos la herramienta *ZRemesher.*

Lo primero que debemos hacer es duplicar el modelo en su versión *Dynamesh*, es decir la no modificada, lo siguiente será usar la herramienta *ZRemesher* en esta copia, el objetivo es que tenga aproximadamente los mismos polígonos que la del *Dynamesh*, de esta manera el siguiente paso será mucho más sencillo, se pueden modificar las opciones del *ZRemesher* desde la propia herramienta.

Figura 61: Proceso de *Dynamesh* a *ZRemesher*

Una vez realizado este proceso tendremos un modelo *high poly* con todo detalle y con una *mesh* ordenada sin irregularidades que puedan causar problemas a futuro. Pero llevar este objeto directamente al motor no sería óptimo, ya que a pesar de haber mejorado la *mesh*, sigue teniendo demasiados polígonos que podrían afectar a la optimización, es aquí donde entra el siguiente paso, la retopología.

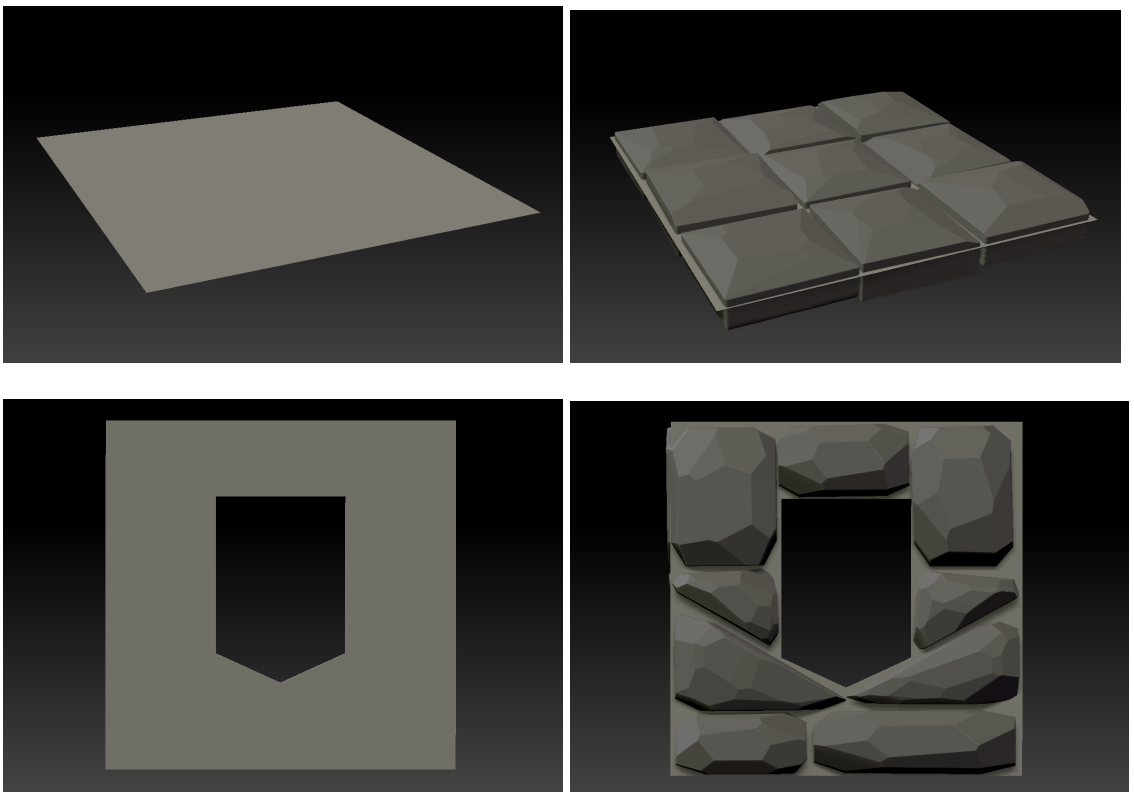


Figura 62: Ejemplos de piezas modulares del *blockout* a la versión final

Para realizar la retopología de los modelos *high poly* como se ha mencionado anteriormente se usará el *software Autodesk Maya*, este programa da muchas opciones al usuario para crear la retopología de modelos de alto poligonaje. El objetivo de la retopología es minimizar la cantidad de polígonos de una maya para así poder añadir el modelo al motor y facilitar la creación de *UV* para el texturizado de más adelante. *Maya* da diferentes opciones para obtener estos resultados, en este caso se han contemplado dos de ellas, una sería para crear la nueva maya automáticamente, esto se hace con la herramienta que nos ofrece *Maya* llamada *polyRetopo*. Gracias a esta herramienta se puede crear una nueva maya basada en una maya *high poly* con el poligonaje que se desee, este proceso se puede manipular con varias opciones hasta conseguir el resultado deseado.

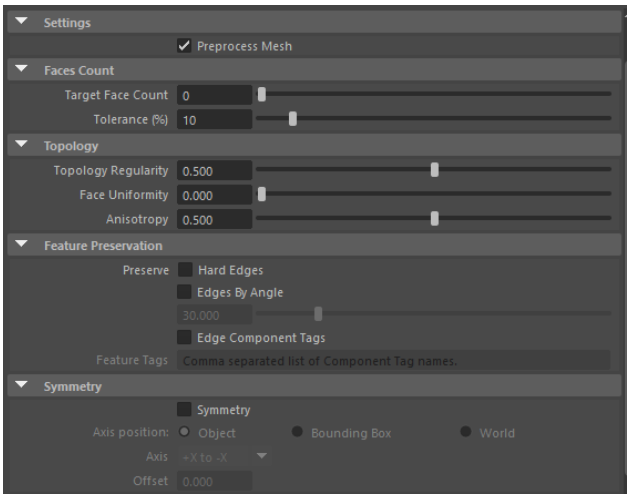


Figura 63: Opciones de la herramienta *polyRetopo*

A pesar de que *polyRetopo* puede ser útil en algunos casos, en otros se necesitará más precisión y se tendrá que crear una nueva *Maya* desde cero, para facilitar esta tarea existe la herramienta *Quad Draw*, seleccionando el imán para que se fije el objeto seleccionado podremos dibujar polígonos por encima del modelo *high poly*, en un principio puede resultar algo difícil, pero la curva de aprendizaje es muy sencilla y en pocos minutos se convierte en un trabajo muy mecánico.

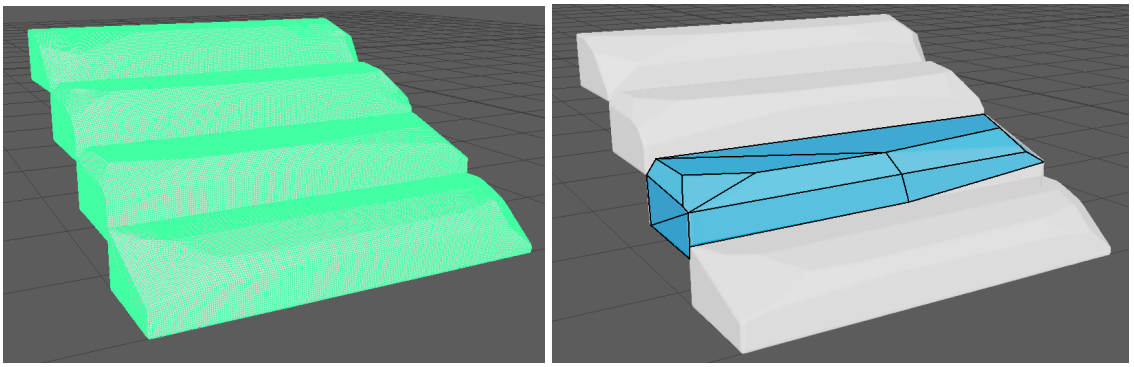


Figura 64: Proceso de retopología en *Maya* con la herramienta "*Quad Draw*"

Después de esto solo quedaría crear las *UV* de los modelos para poder texturizarlos en *Substance Painter*, como en el proceso de retopología *Maya* también da una opción para crear *UV* automáticas. De igual manera, esta opción también tiene sus problemas, como no hacer correctamente los cortes de las piezas o la posibilidad de que algunas de las piezas se lleguen entre ellas, generando problemas visuales durante el texturizado. Por eso mismo la opción óptima de crear *UV* es creándolas a mano, aun así la herramienta de *Maya* puede servirnos para modelos con más polígonos que supongan un trabajo mayor, realizarlos a mano, siempre revisando el resultado para que no haya problemas en el futuro.

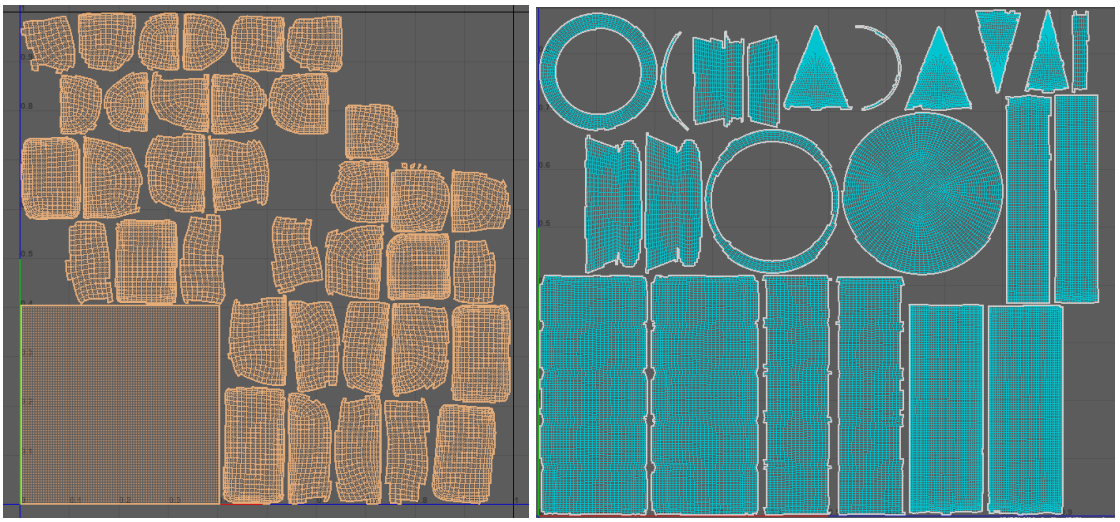


Figura 65: *UV* realizadas por la herramienta automática de *Maya*

Cuando se tengan todos los modelos necesarios creados, solo faltaría organizarlos, es recomendable crear carpetas separadas para cada *asset* con su modelo en *high poly*, que servirá para el siguiente apartado, Creación de Texturas. Es importante tener organizada la lista de *assets* que se van a utilizar principalmente porque exportaremos la carpeta de *assets* directamente al proyecto de *Unreal* junto a las texturas que se crearán a continuación. La lista de modelos en este caso es la siguiente:

Wall, Wall2, Wall3, Wall4, WallDoor, DoorFrame, WindowFrame, WallWindow, Ceiling, Floor, Column, ColumnBase, CoumnTop, Column2, Column2Base, Column2Top, ColumnBalcony, Railing, Railing2, Stairs, Fountain, MagicTower y *GiantRock.*

Al final de la memoria, en el apartado de anexos, se hará una lista detallada con imágenes de los *assets* ya texturizados para poder apreciar detalladamente cada uno individualmente.

5.5. Creación de Texturas

En esta sección se hablará de todo lo que corresponde al proceso de texturizado del proyecto, se ha dividido en dos secciones. En la primera se hablará sobre la creación de materiales en *Substance Designer* mientras que en la segunda se hablará de la creación de texturas en *Substance Painter*. Debido a que *Substance Designer* es un programa complejo de usar que se necesita de muchas horas de experiencia para poder dominarlo totalmente y conocer todas sus posibilidades, no se han creado los materiales del proyecto con el mismo, aun así, se hablará del programa y como se ha utilizado para la creación de algunos materiales básicos.

Lo primero importante antes de empezar con *Substance Designer* es saber que es lo que vamos a hacer con el programa, es importante destacar que en *Substance Designer* crearemos materiales que podrán ser usados para texturizar los objetos en *Substance Painter*, por lo que para empezar deberíamos usar una configuración determinada para que funciones en *Substance Painter* en el siguiente paso.

Lo primero que hay que hacer es crear un nuevo *Substance* y seleccionar los atributos del mismo, en este caso se ha decidido que la resolución del material será de 2048 x 2048 y se usará el *template* de *Metallic Roughness*, que es el *template* básico recomendado. Estos atributos serán los mismos en *Substance Painter* cuando pasemos a texturizar los objetos más adelante, así que es importante recordarlos.

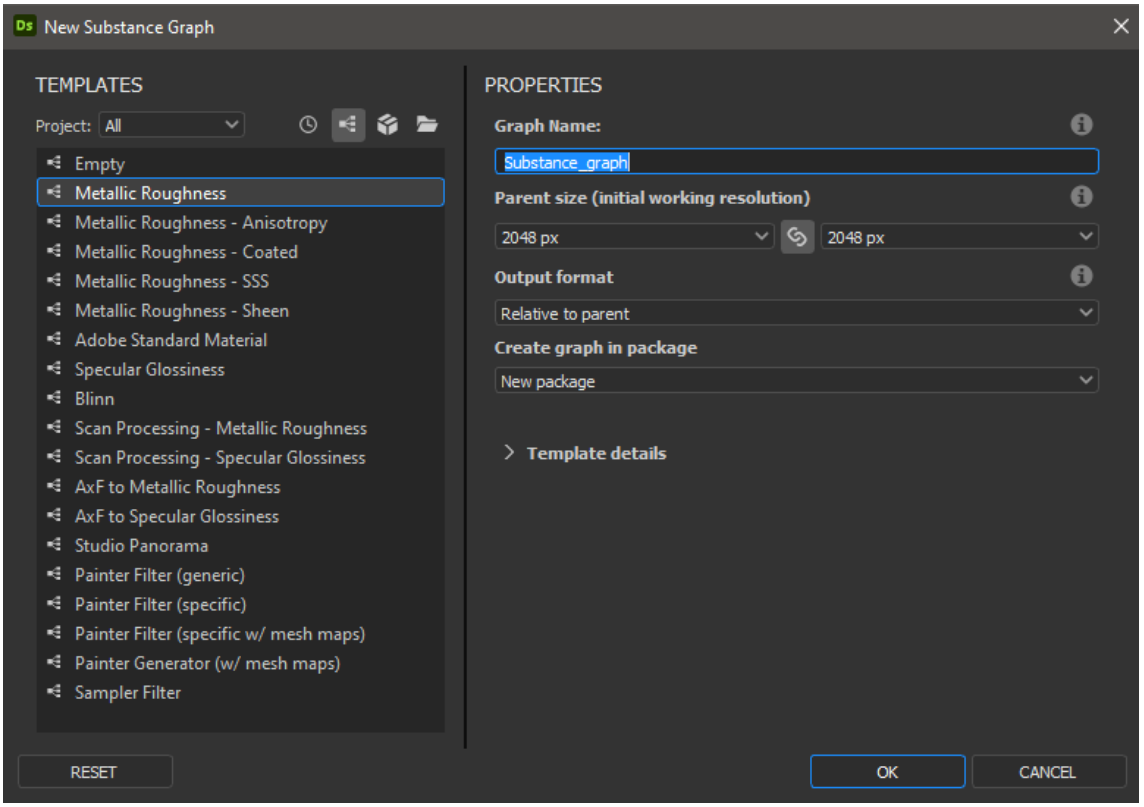


Figura 66: Captura de los atributos del proyecto de *Substance Designer*

Una vez creado el proyecto aparecerán diferentes ventanas en pantalla, la principal es el editor de *graphs*, en esta pantalla modificaremos, añadiremos y combinaremos elementos para crear el material. Como se ha mencionado se ha escogido el *Metallic Roughness* como *template*, por lo que el proyecto nos ha generado los siguientes atributos para el material, estos son: *Base Color, Normal, Roughness, Metallic, Ambient Occlusion* y *Height*. Cada uno de estos atributos *graphs* modificará una característica del material, por ejemplo el *base color* sería la textura 2D, el *height* es el mapa de alturas o las normales que dará un efecto 3D. En la pantalla de abajo podemos ver como va avanzando la creación del material en tiempo real, se puede cambiar qué elemento estamos visualizando así como la forma en la que visualizamos, cubos, esferas, planos, etc.

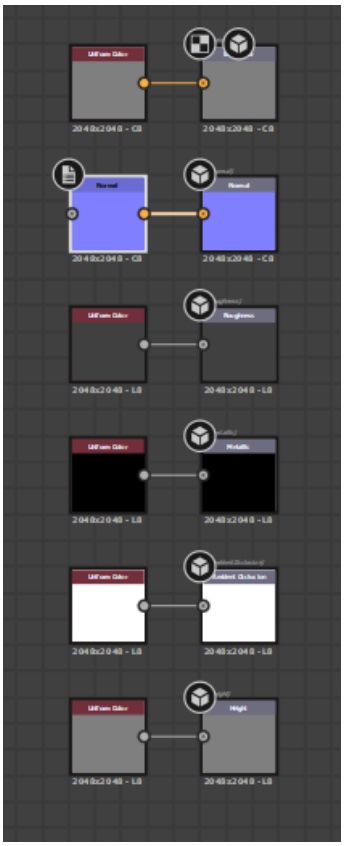


Figura 67 : Propiedades del material en *Substance Designer*

El programa da infinitas opciones para crear materiales y tiene una gran profundidad para hacer prácticamente cualquier cosa que se nos ocurra. Las funcionalidades del programa pueden entenderse viendo las posibilidades que nos dan los nodos, los responsables de modificar todos los aspectos del material que crearemos. Existen muchos tipos de nodos, los de tipo *Blend and Transform*, que modificarán el aspecto del *noise*, los de *Color, Grayscale* y *HDR*, que tienen que ver con el color y tono del material, los gradientes que son básicamente el degradado, etc. Todos ellos tienen mucha más profundidad de lo que parece, ya que el programa nos da infinitas opciones de escala, curvas, opacidad, etc. que nos permiten crear el material que teníamos en mente.

En este caso no se necesitarán este tipo de materiales, ya que en el proyecto que se está haciendo los detalles que se pueden crear con *Substance Designer* se han creado ya con modelos 3D, por lo que no necesitaremos texturas específicas. De todas maneras siempre es útil informarse de las posibilidades del programa para futuros proyectos.

La segunda parte para la creación de texturas es el uso de *Substance Painter*, al contrario que en *Designer*, este programa consiste en pintar los modelos *low poly* que teníamos para darles detalle, color y textura, no para crear los materiales para el pintado. Básicamente, pintaremos el objeto con materiales ya disponibles en *Substance Painter* o los que hayamos exportado desde *Designer*.

El primer paso es crear un nuevo proyecto y seleccionar el *FBX* en *low poly* con las *UV* creadas en *Maya*, es recomendable usar la *template PBR Metallic Roughness*, también podremos seleccionar la resolución con la que trabajaremos las texturas, con estas opciones seleccionadas creamos el proyecto.

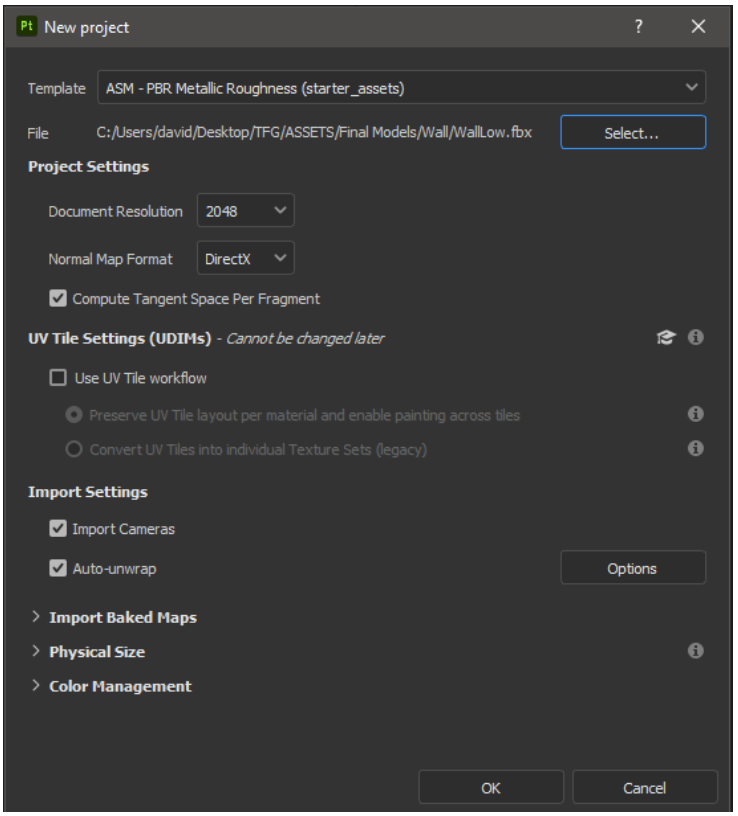


Figura 68: Creación del proyecto de *Substance Painter*

El siguiente paso es pasar la calidad del modelo en alta definición al modelo *low poly,* para esto tendremos que hacer un *bake* de los mapas del modelo *high poly*. En la parte izquierda de la pantalla debemos ir a *Texture Set Settings* y seleccionamos *Bake Mesh Maps*, a continuación se nos pedirá seleccionar el modelo *high poly* así como la resolución a la que se hará el mapeo. El resultado de esto pasa a ser como en la imagen de abajo, gracias a esto podremos usar modelos *low poly* con el mismo detalle que los de *high poly* en *Unreal Engine*.

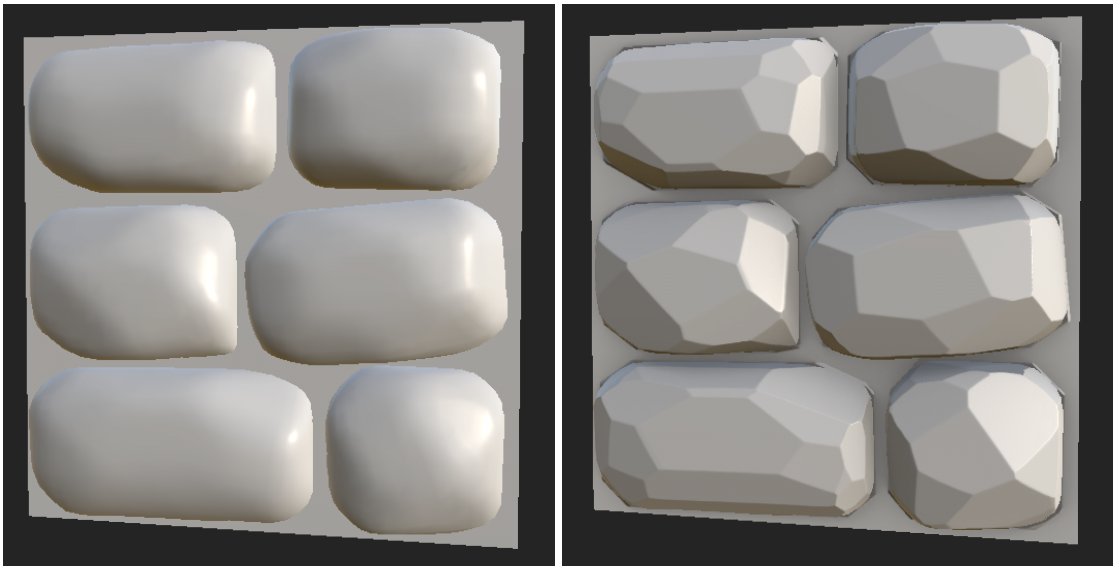


Figura 69: Comparación antes y después de *Bake Mesh Maps*

Una vez con el *bake* hecho solo faltaría comenzar a pintar el modelo, en este caso se ha pintado varias *fill layers* comenzando con el material *Steel Ruined*, que dará rugosidad a las rocas, el *Concrete Simple* dará el color y textura de la piedra. Con una *Black Mask* se ha añadido una capa de *Leather Damaged* que servirá para el fondo de la pared, con la herramienta de *Polygon Fill* se pueden pintar los polígonos del modelo sin miedo a pintar otra parte del mismo, es lo que se ha usado junto a la *Black Mask* es este caso. Por último, para dar más detalle se ha añadido un material oscuro que junto a una *Black Mask* y un *Generator*, en este caso *Metal Edge Wear* se ha conseguido este efecto en los bordes de las rocas.

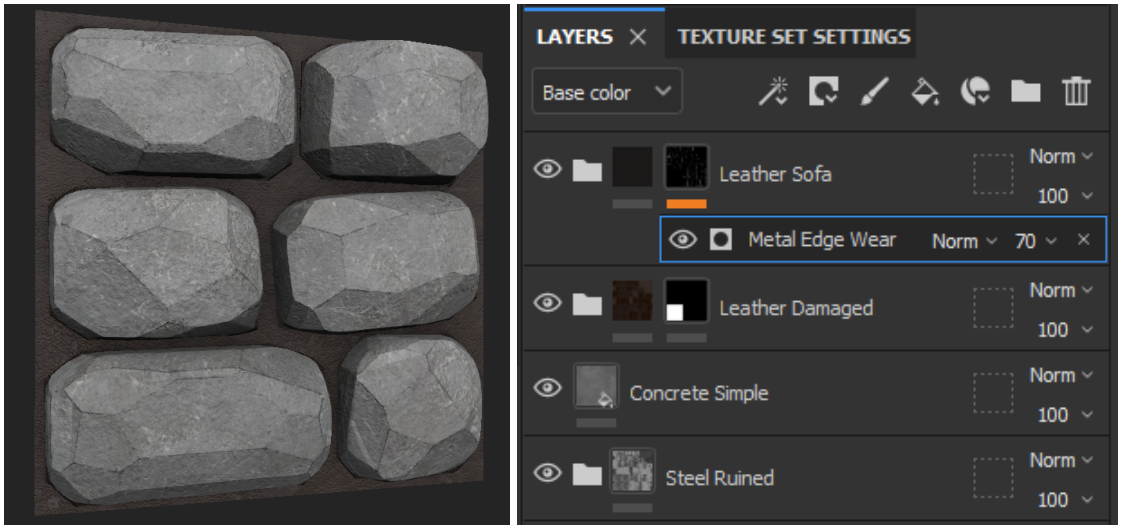


Figura 70: *Layers* y resultado del *asset Wall*

El resto de texturizado tiene un proceso similar, obviamente variando las capas de texturas y los generadores dependiendo del resultado a obtener en los diferentes *assets* del proyecto.

5.6. Montaje del Environment / Implementación en Unreal Engine 4

Esta fase del desarrollo se ha ido haciendo conforme se avanzaba en el resto de fases, por lo que estará dividida en diferentes momentos del desarrollo.

La primera fase consiste en crear un entorno básico con el primer *blockout* que tenemos y montar la estructura del escenario para ir mejorándolo poco a poco. Lo primero que tenemos que hacer es crear un proyecto de *Unreal* nuevo, lo podemos nombrar como queramos, pero es importante que seleccionemos algunas opciones obligatorias para el desarrollo de *environments* como el *target platform (Desktop/Console)* y la calidad al máximo. Si el ordenador que usemos lo soporta se puede activar la opción de *Raytracing*, una herramienta muy útil para mejorar la calidad de la iluminación de la escena de manera considerable, en este caso no se disponía de un ordenador que pueda usar esta tecnología, por lo que se desactivó esta opción para el proyecto.

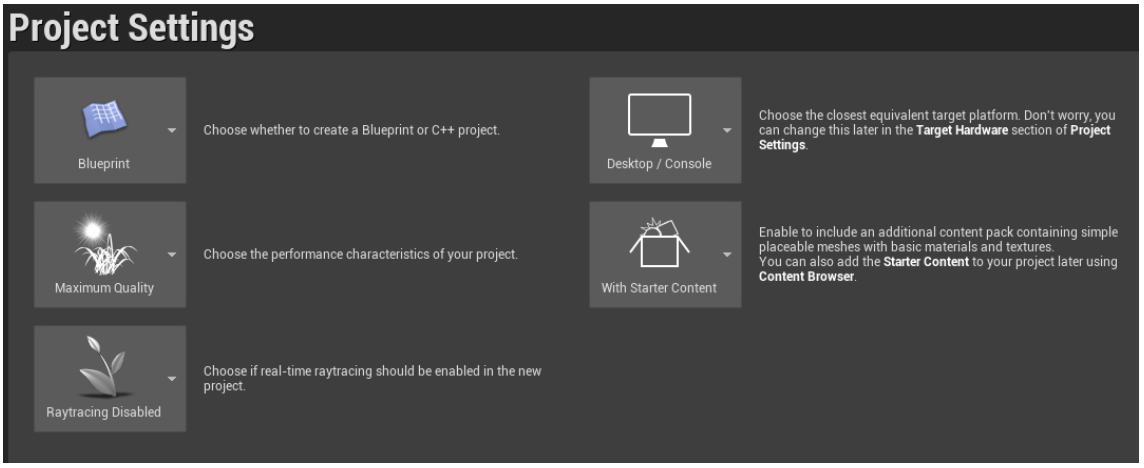


Figura 71: Opciones de proyecto de *Unreal*

Una vez con el proyecto creado lo normal sería crear un plano y modificarlo con la herramienta de terreno, en este caso se hará otro procedimiento por la naturaleza del *environment*. Se decidió crear un océano gigante usando el plugin gratuito de *Unreal "Water"*, este *plugin* nos permite crear diferentes masas acuáticas, desde ríos, lagos u océanos como en este caso.

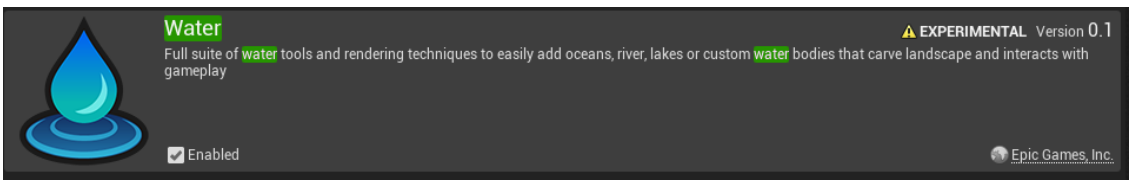


Figura 72: *Plugin* "Water" de *Unreal*

Con este *plugin* es muy fácil crear un océano con un solo clic sobre un plano, el océano tiene propiedades y puede modificarse de varias formas, pero de momento se dejará como está, ya que este tipo de modificaciones es mejor hacerlas cuando la escena esté totalmente terminada y no a mitad del desarrollo.

Como se ha dicho anteriormente, esta primera fase consiste únicamente en añadir los elementos que tenemos a *Unreal* para poder ir viendo el resultado del *environment* en tiempo real en nuestro motor, hay varias opciones que irán añadiendo como las nubes volumétricas que en el caso de esta temática funcionarán muy bien con el castillo flotante, pero lo inteligente es empezar por lo más básico.

Con el océano creado, lo primero que se debería añadir son las estructuras en sí, los primeros *assets* y el *blockout* del castillo. Se comenzó añadiendo el *blockout*, al ya estar escalado desde *Maya* nos daría una referencia de los tamaños del resto de elementos, una vez con el castillo en escena si incluyeron los *assets* de las islas flotantes y se escalaron de forma que encajasen con el castillo y los torreones de alrededor, también se utilizó este *asset* para la creación de las montañas de fondo que le dan un mejor empaque a la escena en general. Después de retocar varias posiciones y añadir las cadenas entre las islas, se le añadieron texturas básicas de *Unreal* a las estructuras para poder ver mucho mejor cómo será el resultado a futuro, estas texturas son *standby,* pero en el punto en el que estamos nos sirven para ver un poco a color como quedará la escena visualmente.

Por último, para esta fase, se ha creado una cámara estática en el motor que nos servirá para ver el avance del proyecto con la implementación de *assets*, texturas e iluminación, entre otras mejoras que va a ir teniendo.

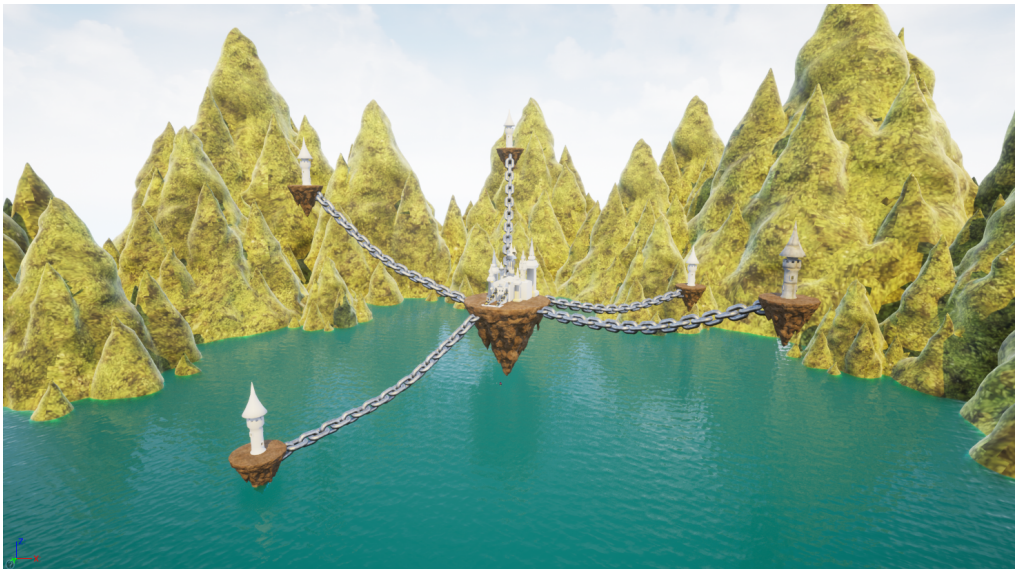


Figura 73: Primera iteración del escenario en *Unreal*

A partir de este momento la tarea consistirá en ir cambiando los *assets* del *blockout* por los finales hasta obtener el resultado deseado. La tarea es bastante simple, se deben añadir tanto los modelos como las texturas al proyecto de *Unreal Engine 4*, una

vez añadido se creará un material en blanco para el modelo 3D, a este material podemos acceder haciendo doble clic y se nos abrirá una ventana para poder editar los materiales. En esta ventana se deberán arrastrar las texturas que hemos importado de ese objeto, una vez las tengamos en la plantilla simplemente deberemos unir los nodos de la textura con los atributos del material, por ejemplo la textura del *base color* al atributo *base color,* etc. así hasta tener el material completo.

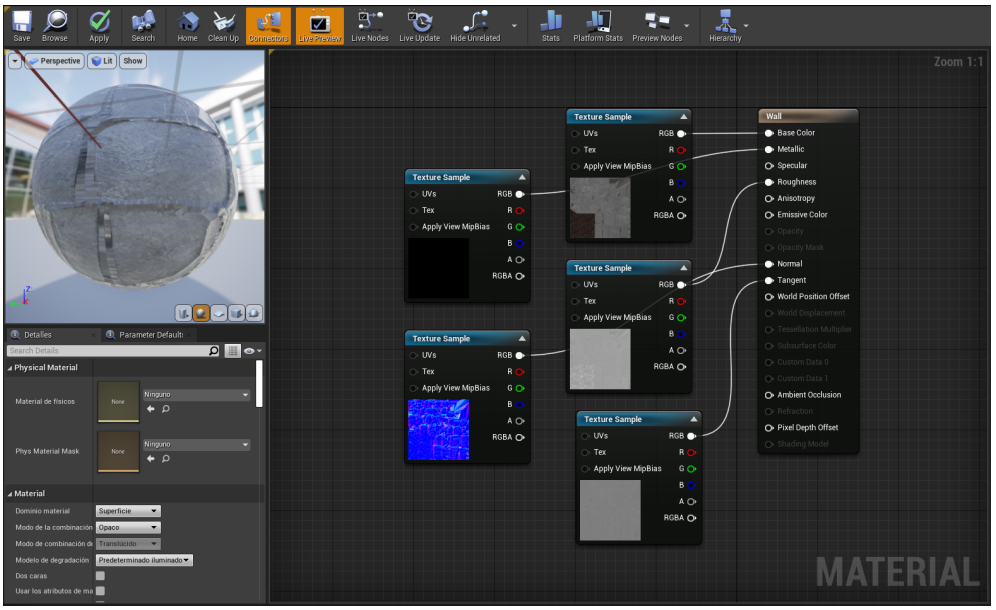


Figura 74: Ventana de edición de materiales de *Unreal Engine*

Con el material modificado, todos los modelos asignados a ese material cambiarán automáticamente, por lo que no hará falta hacerlo más veces. Es posible que el material no se va como se veía en *Substance Painter*, esto puede ser por dos razones principalmente, la primera es un tema de iluminación, puede ser que la iluminación del proyecto no esté configurada correctamente y esto haga que las texturas tengan demasiado brillo. Aparte de corregir la iluminación, también podría contemplarse retocar el material modificando el brillo del material desde el editor de materiales si la nueva iluminación no soluciona este problema. El otro problema puede ser externo, en la exportación de texturas en *Substance Painter*, hay varias opciones de exportación, si la opción predeterminada no acaba de funcionar se podría probar a volver a exportar las texturas con la configuración para *Unreal* que ofrece el programa. Con estas propuestas sobre la mesa, debería solucionarse cualquier problema de tonalidad de las texturas.

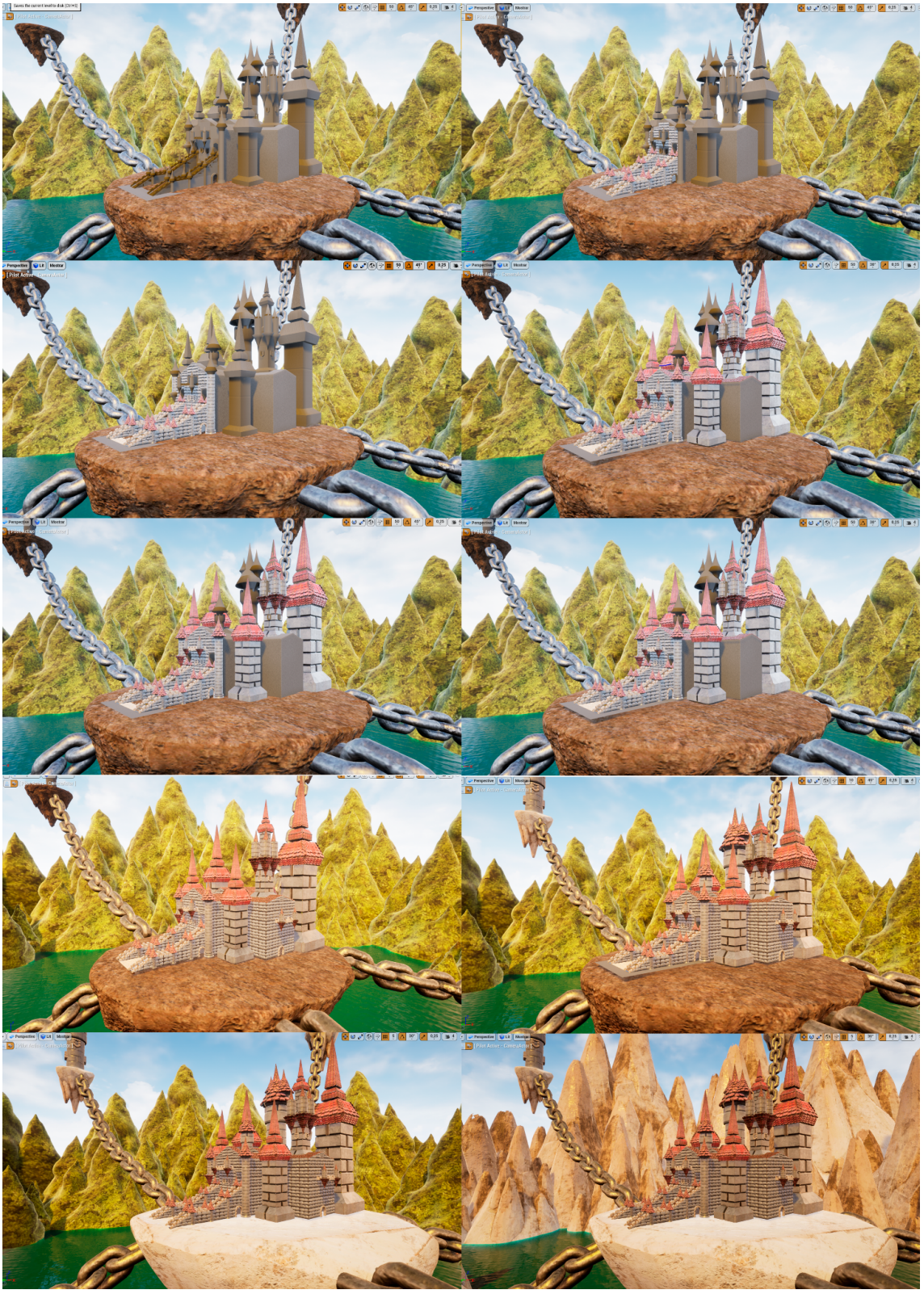


Figura 75 : Evolución del *Environment* en *Unreal*

Una vez montados todos los *assets* tenemos el escenario casi acabado, pero falta modificar algunos elementos que le darán al resultado final un acabado muy mejorado al actual.

5.7. Revisión de Calidad

Este apartado consiste en revisar el proyecto y modificar algunos aspectos que podrían mejorarse. En este caso se han intentado mejorar dos apartados en específico que mejorarán en gran medida la calidad general del proyecto, el primero es el suelo de las islas flotantes, al que le sentaría muy bien algo de césped para evitar que se vea una textura plana bajo el castillo, el segundo apartado es la iluminación, cambiar tanto el tono como la intensidad será necesario para lucir el trabajo de la mejor manera posible.

Para hacer el césped se ha utilizado la herramienta de follaje de *Unreal Engine*, aunque antes deberemos hacer un *asset* que nos permita utilizarla. En este caso se ha hecho un modelo 3D bastante simple en *Maya* que emulará lo que serían las briznas de césped.

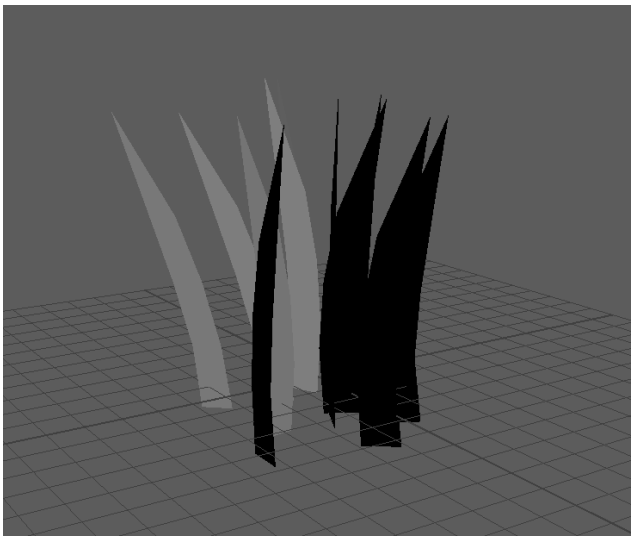


Figura 76: Modelo en *Maya* del césped

Con este modelo exportado, solo faltaría añadirlo al proyecto de *Unreal* y de ahí seleccionarlo como follaje para poder pintar por el mapa fácilmente, pero antes de esto es importante añadir un material al modelo, en este caso se le añadirá una propiedad especial. Se trata de un atributo llamado *SimpleGrassWind* que nos proporciona *Unreal* que le dará al *asset* un efecto de viento muy vistoso, para añadir este efecto simplemente hay que acceder al material del modelo y modificarlo como en la imagen de abajo. Como se puede ver, el efecto tiene diferentes atributos que se pueden modificar añadiendo las constantes deseadas hasta obtener el resultado que se busca.

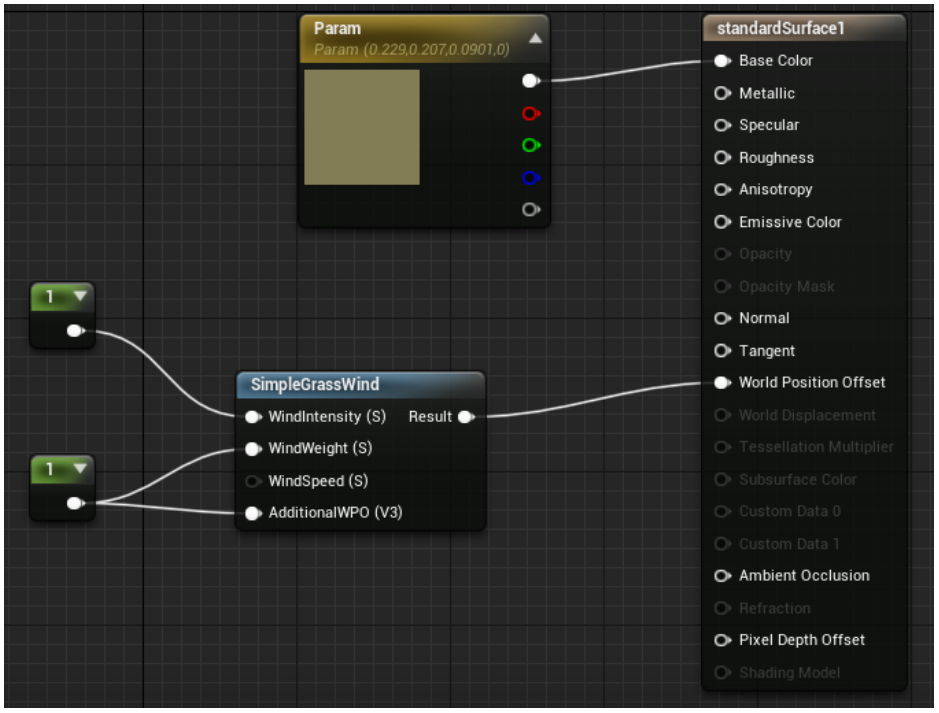


Figura 77: *SimpleGrassWind* en *Unreal Engine*

Una vez hecho esto solo hará falta modificar los parámetros de pintado del césped (principalmente la densidad y el tamaño) y pintar el terreno al gusto. El césped se ha añadido tanto en la base del castillo como en la base de las torres para tener un terreno coherente.

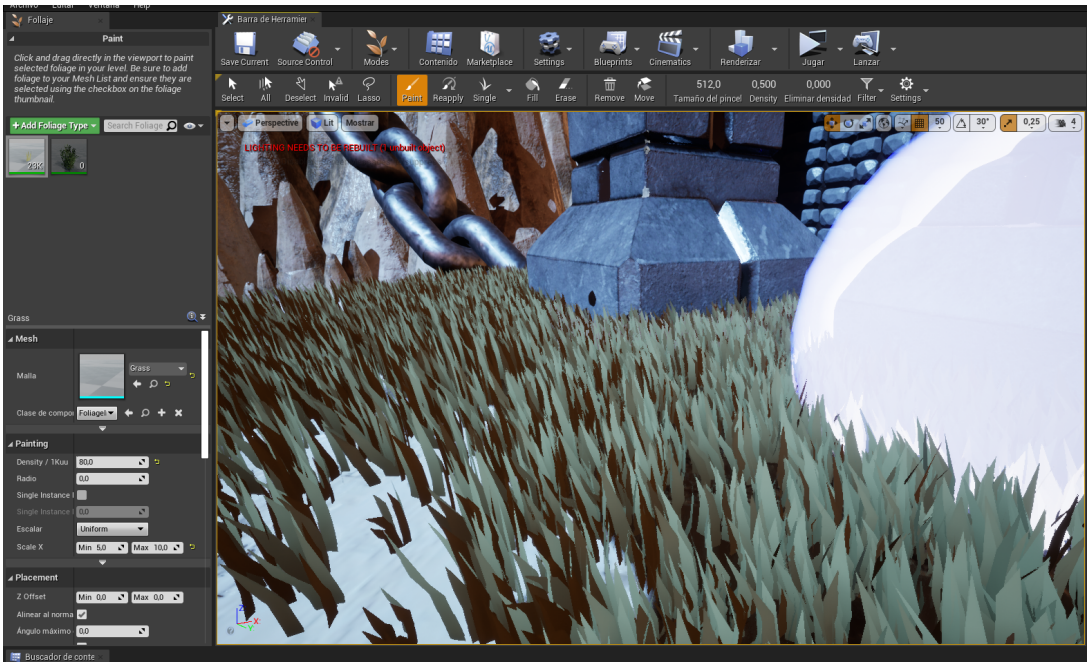


Figura 78: Opciones de follaje de *Unreal* y resultado del césped

El otro tema que había que mejorar en el proyecto es la iluminación, hasta el momento los *assets* se veían con demasiada luminosidad como se ha mencionado en el apartado anterior y la iluminación general no se adecuaba demasiado al ambiente del proyecto. Por decisión artística se ha decidido que el *environment* transcurrirá en el anochecer, para conseguir este efecto se han modificado tres atributos del proyecto de *Unreal*, en este caso los dos que tienen que ver con la iluminación del proyecto, la *Light Source* (*DirectionalLight*), la *Sky Light* y las *Sky Sphere*.

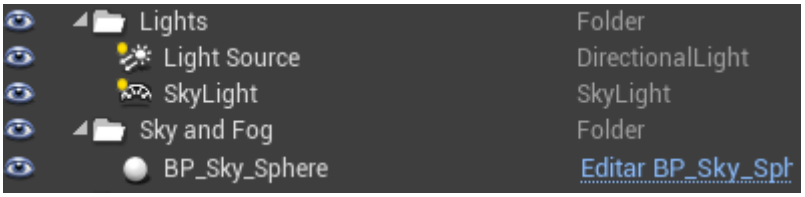


Figura 79: Luces en el editor de *Unreal*

Para la *Light Source* se han modificado los parámetros como en la imagen de abajo, con esto conseguiremos que el sombreado y la iluminación de los modelos tenga este efecto nocturno que buscamos, los valores de intensidad se pueden modificar hasta encontrar unos que se adecuen al resultado que queremos obtener, el más importante para que la luz parezca nocturna es el color de la luz.

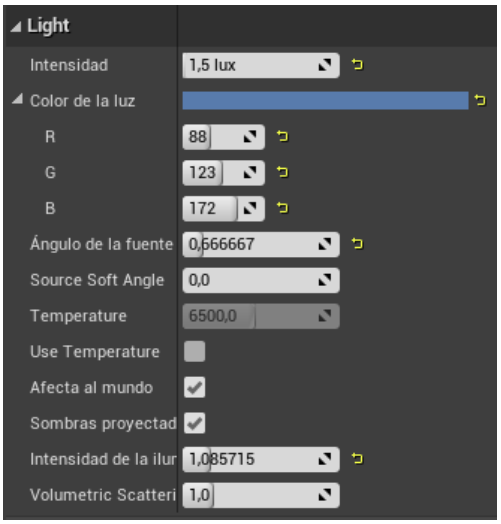


Figura 80: Atributos de *Light Source*

Para la *Sky Light* y la *Sky Sphere* se han modificado los siguientes atributos, el color de la luz en este caso es así debido a que se le quiere dar ese efecto de atardecer-anochecer que se intensificará con la modificación de la *Sky Sphere*. En el caso de esta, lo más importante a cambiar es la altura del sol, esta determinará el momento del día representado por el *Skybox*, además podemos cambiar la tonalidad para ajustar el color que queremos en la escena.

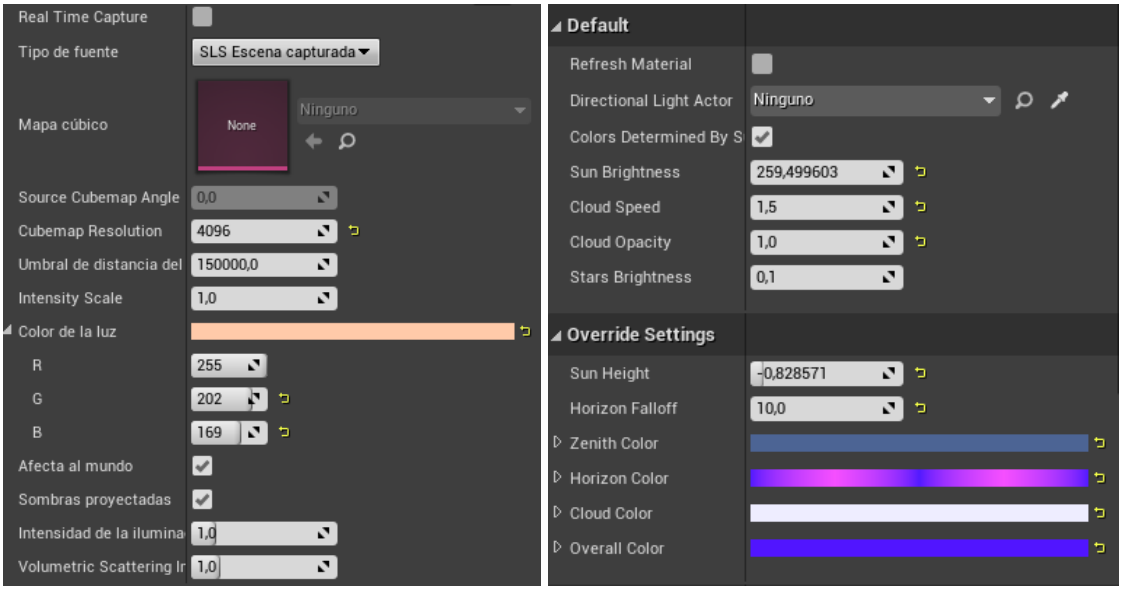


Figura 81: Atributos de *Sky Light* y *Sky Sphere*

Cabe recalcar que como este proyecto se ha hecho en Unreal Engine 4 es importante decir que la iluminación en gran parte no podemos verla con el resultado final hasta que hagamos un renderizado de las mismas, por lo que es importante no olvidarse de esto antes de comenzar con la presentación del *environment*.

5.8. Publicación y Portfolio

Como ya se ha mencionado en el apartado Metodología, el trabajo realizado lo presentaremos en *Artstation*, el link al *ArtStation* para ver en detalle el trabajo final está en el apartado de Links, casi al inicio de la memoria. Para hacer una buena presentación en *ArtStation* se han seguido algunas normas, no hay demasiadas imágenes, ya que un exceso puede desviar la atención de lo verdaderamente interesante.

En este caso se ha incluido un pequeño vídeo para mostrar el castillo mejor y el movimiento del césped, tres imágenes del proyecto final, tres imágenes de *assets* en detalle (dos texturizadas y una desde *ZBrush*), y por último un par de imágenes que muestran el *concept* y el *blockout* para añadir contexto. Es importante seguir este orden, ya que lo primero que debe ver el que va a visitar el proyecto debe ser lo mejor, en este caso el vídeo, las siguientes imágenes van de mayor a menor importancia dejando para el final imágenes del *blockout* para quien le interese ver en más detalle el proyecto. Toda esta información se ha obtenido viendo ejemplos de otros artistas en *ArtStation* y al final todos siguen la misma línea de presentación.

Para hacer el vídeo se ha utilizado la herramienta de *Unreal Engine Sequencer*, en la ventana de *Cinematics*. Esta herramienta nos permite grabar un clip de vídeo gracias a que se puede exportar un vídeo controlando cámaras de *Unreal* dándoles una posición final e inicial para así acabar teniendo un vídeo para mostrar el *environment*. En videojuegos la herramienta también se usa para crear cinemáticas, es muy útil y es fácil aprender a usarla, simplemente deberemos colocar la cámara donde queramos que inicie la escena y moverla donde queremos que acabe, dependiendo de la cantidad de *frames* entre una posición y otra, el vídeo durará más o menos. Además, se pueden hacer cortes en una misma secuencia añadiendo más cámaras, las posibilidades son infinitas, para lo que queremos hacer, es más que suficiente.

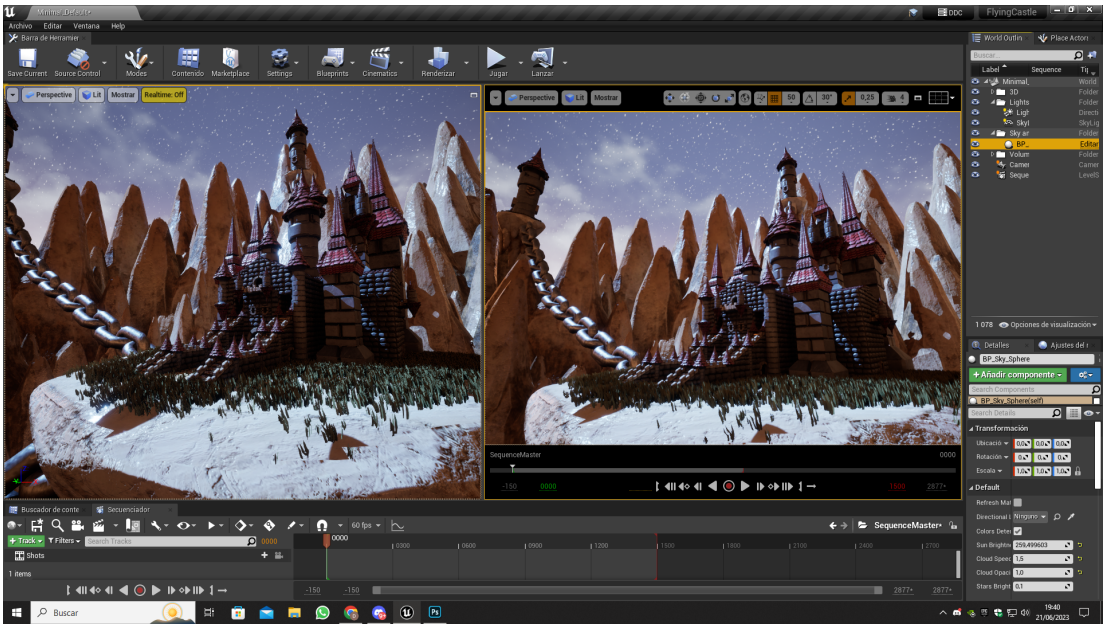


Figura 82: Uso de la herramienta *Sequencer* para crear el clip de vídeo

También se han añadido algunas descripciones de imágenes y lo más importante, la descripción general del proyecto, es importante que todo esté en inglés para llegar al máximo público posible. En la descripción del proyecto se ha añadido una breve presentación, una explicación del que se ve en pantalla, con qué herramientas se ha logrado conseguir el resultado, los problemas encontrados así como lo que he logrado aprender en el transcurso del proyecto.

6. Validación del proyecto

Para validar el proyecto se ha pedido opinión en el foro *Polycount*, el foro más famoso para mostrar tu arte 3D y recibir *feedback* de artistas más experimentados. Además, en *Polycount* se pueden consultar tutoriales específicos creados por la comunidad, y en general, la mayoría de temas de discusión son interesantes para enriquecer el conocimiento de arte 3D, por lo que es la web ideal para este caso.

La mayor crítica respecto al proyecto en el foro ha sido la repetición de patrones, principalmente en las paredes y el tejado, se ha recomendado cambiar el patrón de estos elementos, ya sea creando nuevos modelos o cambiando algunas texturas para que no se note tanto esta repetición. Para solucionar este problema al inicio del proyecto se comentó usar la técnica del *Vertex Paint* que básicamente consiste en pintar una nueva textura por encima de otra en los modelos que uno quiera, se podría haber creado otra textura y usar esta técnica, pero en la práctica dio problemas y el resultado no era el esperado por lo que se descartó.

También recomiendan rotar el mismo modelo para que parezca otro o invertirlo para que se vea distinto, este es un buen consejo que se usará para futuros proyectos, en este *environment* no se había planteado.

Otro consejo que no entraría en este proyecto, ya que sobrepasaría el tiempo estimado para completar el *environment* es crear gran variedad de modelos para estos elementos más repetitivos. Es una recomendación bastante obvia, pero es cierto que en este caso quitaría más tiempo del esperado y por eso se consideró usar el *Vertex Paint* antes que crear nuevos modelos, aun así es un consejo a valorar en futuros proyectos que se hagan con más tiempo y experiencia.

Por el momento no ha habido más respuestas en el foro, pero las observaciones que se han publicado han servido para hacer un pequeño análisis de lo que llegaba a flaquear más en el *environment*, la repetición de patrones.

7. Conclusiones

En general, podría decirse que la mayoría de objetivos del trabajo se han conseguido, algunos con más solvencia que otros. En primer lugar, se ha conseguido usar *Unreal Engine* para la creación de un entorno 3D, que era el objetivo principal del proyecto, se planteó que fuese de fantasía medieval y también ha sido así. También se han creado *assets* y texturas de calidad y se han hecho *renders* para apreciar mejor el detalle (en la sección de anexos está la lista completa de *assets* del proyecto), además todo el proceso ha quedado reflejado en la memoria, especificando los problemas encontrados y los métodos utilizados en cada caso, por lo que el objetivo de crear una metodología de trabajo también se ha cumplido. Además, el trabajo ha sido presentado en *ArtStation*, por lo que también se ha cumplido el apartado de presentar el proyecto en un porfolio correctamente.

Respecto a los objetivos más específicos del proyecto, algunos se han cumplido en mayor medida que otros. Se ha aprendido a usar *Unreal Engine 4* para el desarrollo 3D, la creación de *environments* en este caso. También se han aprendido técnicas de modelado en *ZBrush* y en *Autodesk Maya* gracias a un trabajo de investigación cómo qué opciones nos da *Maya* para la creación de retopología o el uso de algunos pinceles en *ZBrush*. Respecto a *Substance Painter* se ha ampliado el conocimiento respecto a algunos aspectos como el uso de generadores y opciones avanzadas, con *Substance Designer*, a pesar de haber aprendido las nociones básicas del programa, no se ha puesto en práctica el nivel que se había planteado en un principio, esto se debe a que es un programa algo más complejo de utilizar y la calidad de los materiales que se estaban intentando hacer no era la adecuada para el proyecto, por lo que finalmente no se llegó a usar en el proyecto aunque sí que ha quedado reflejado en la metodología.

También se ha aprendido algo de nodos para crear *shaders* en *Unreal Engine* en la creación de materiales, aunque se podría haber ahondado más en este aspecto. Se ha optimizado el proyecto usando retopología y se han usado varios métodos de los presentados en la metodología de trabajo, en general este proyecto ha mejorado en general mis habilidades como artista digital.

Respecto a los posibles problemas se ha dado solución a la mayoría de ellos, la planificación ha sido efectiva, ya que se ha completado el *environment* en los tiempos dados y se ha llegado a la fecha de entrega, a mitad del proyecto se replanteó la segunda parte de la planificación y fue un acierto para llegar a tiempo al final del proyecto. El problema de no saber usar *Unreal* también se solucionó porque antes de empezar con el proyecto se estudió sobre el motor, lo que permitió avanzar rápidamente cuando llegó el momento.

El problema de intentar abarcar demasiado se solucionó desde la creación del *blockout,* ya que desde ese momento se determinó que cantidad de *assets* iban a crearse así como la estructura de los mismos, gracias a esto ha sido posible no desviarse de la idea inicial y tener el trabajo terminado. El último problema ha sido el más difícil, es cierto que el escenario tiene varios patrones de repetición, principalmente en las paredes, se planteó usar la técnica *Vertex Paint* para cambiar las texturas de la pared, pero el resultado no fue el esperado, por lo que descartó la idea inicial y se terminó por dejarlo de esta manera, igualmente gracias a la creación de un total de 23 *assets* este problema se pudo subsanar en cierta medida.

Dicho esto, se puede considerar que el problema planteado ha sido solucionado al final del proyecto, gracias a que se ha creado una metodología de trabajo accesible para todo el mundo que quiera iniciarse en el mundo del arte 3D para *Unreal Engine*.

7.1. Líneas de futuro

El proyecto puede mejorar bastante si se aplican diferentes técnicas que no se han llegado a utilizar, como el *Vertex Paint* que nos servirá para evitar la repetición de patrones en las paredes. El *environment* puede expandirse a nivel técnico, quizás añadir nuevos *assets* para los fondos, algunos detalles que mejoren el aspecto general o incluso llegado un punto podría crearse el interior del castillo.

Otro aspecto que no se ha explorado en este proyecto se trata de las partículas en *Unreal Engine* usando sistemas como *Cascade* o *Niagara* que ofrece el motor, en un futuro estaría bien aprender más sobre el tema, ya que es un conocimiento esencial para un *environment artist* y mejora en gran medida el resultado final del trabajo.

El objetivo principal después del proyecto es aprender a usar *Substance Designer* para poder crear materiales de calidad y mejorar futuros proyectos, esto no solo se aplica a *Designer*, en general se intentará mejorar la calidad general, mejorando en la creación de *assets* y aprendiendo nuevas técnicas. Considero que lo más importante ahora es mejorar técnicamente para tener nuevos proyectos en el porfolio y en un futuro añadir las mejoras al castillo para lograr mejorar la calidad del mismo.

8. Bibliografía

- *3dEx (Canal de Youtube especializado en arte 3D) www.youtube.com/@3dextrude. Consultado 7 Feb, 2023*
- *Documentación Zbrush http://docs.pixologic.com/. Consultado 23 Feb, 2023*
- *Documentación Substance Painter/Designer https://substance3d.adobe.com/documentation/home. Consultado 26 Feb, 2023*
- *Acerca de Blender https://docs.blender.org/manual/es/latest/getting_started/about/index.html. Consultado 2 Mar, 2023*
- *Acerca de Mari https://learn.foundry.com/mari/4.2/Content/learnhome/learn_mari.html. Consultado 4 Mar, 2023*
- *Acerca de Cinema 4D https://www.maxon.net/es/cinema-4d/features. Consultado 4 Mar, 2023*
- *Diferencias Maya/3ds Max https://www.autodesk.es/support/technical/article/caas/sfdcarticles/sfdcarticles/ESP/Comparison-of-3ds-Max-and-Maya.html. Consultado 6 Mar, 2023*
- *Sergio Santos. Masterclass: "Texturizado y Substance" https://lboxacademy.es/blog/texturizado-substance-sergio-santos/#:~:text=El%20Substance%20Painter%20se%20parece,hacer%20trazados%20todo%20es%20vectorial…. Consultado 8 Mar, 2023*
- *Documentación Unreal Engine 4 https://docs.unrealengine.com/4.27/en-US/. Consultado 12 Mar, 2023*
- *Documentación Unity https://docs.unity3d.com/Manual/index.html. Consultado 12 Mar, 2023*
- *Tutorial Substance Desinger https://www.artstation.com/learning/playlists/L6Bjv/substance-designer Consultado 8 May, 2023*
- *Pedro P. Fernández. 2018. Producción Visual en Videojuegos. Parte I: Dirección de Arte https://medium.com/@vancorso/producci%C3%B3n-visual-en-videojuegos-parte-i-492755789b68. Consultado 15 Mar, 2023*
- *Building The Art for What Remains Of Edith Finch | Inside Unreal https://www.youtube.com/watch?v=6mE9VnnG5lc&ab_channel=UnrealEngine. Consultado 20 Mar, 2023*

9. Anexos

En este apartado se ha añadido una lista de los assets utilizados para crear el proyecto final.

1. **Wall**

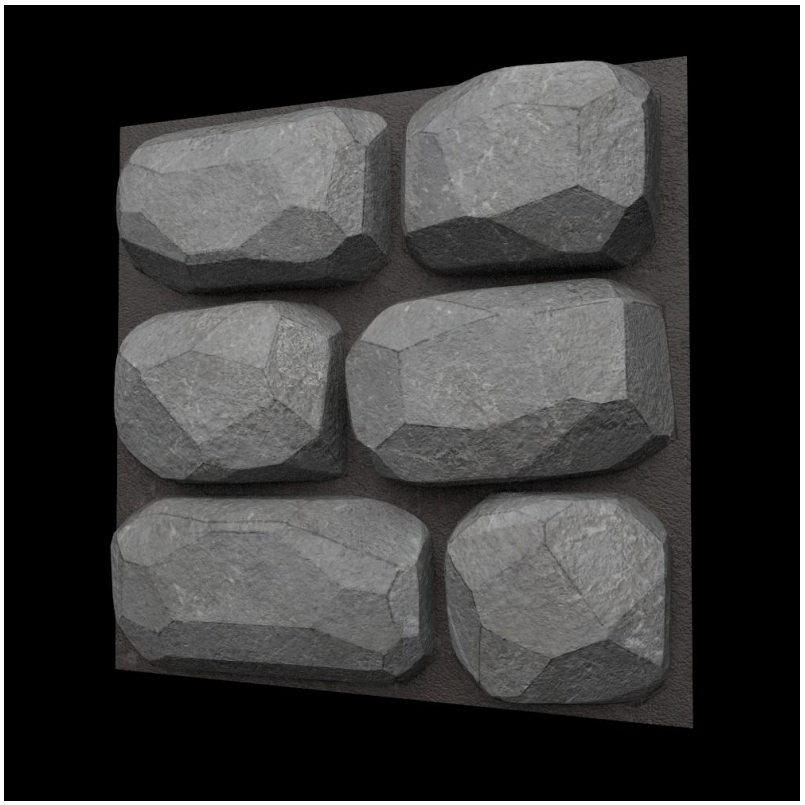


Figura 83: *Render* del *asset Wall*

2. Wall2

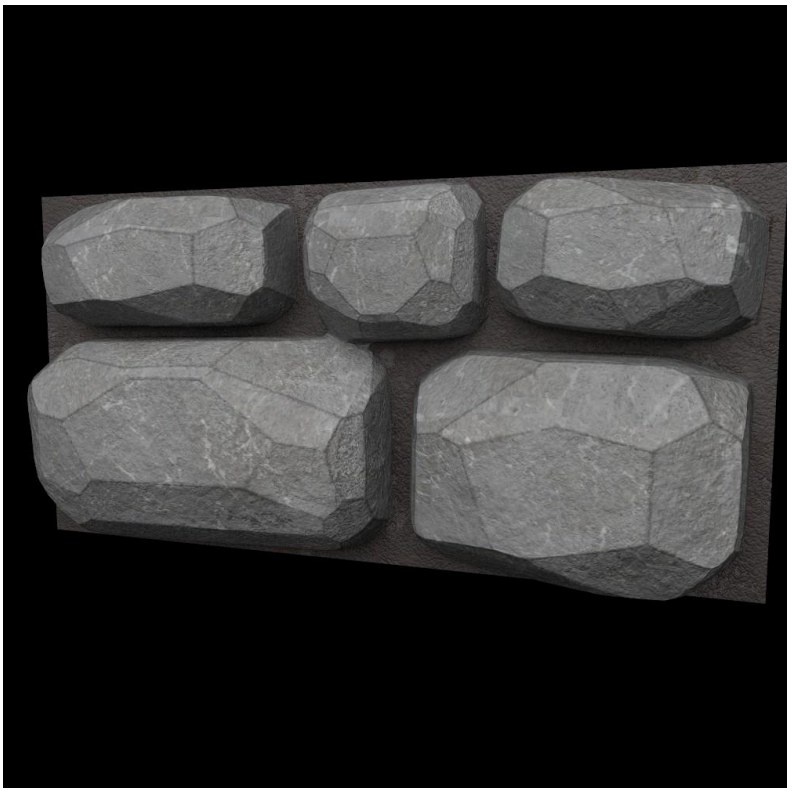


Figura 84: *Render* del *asset Wall2*

3. Wall3

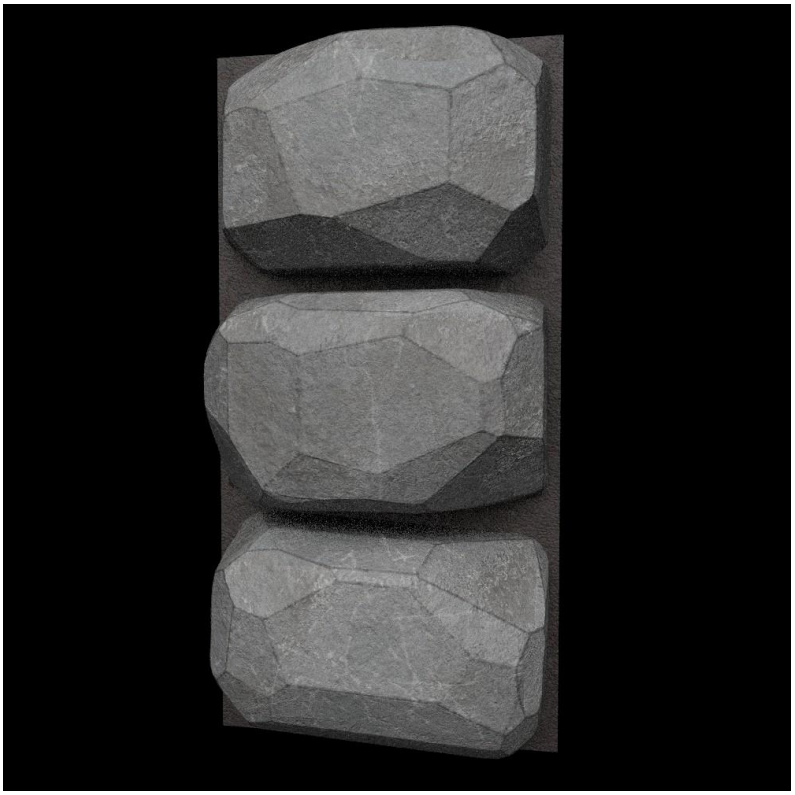


Figura 85: *Render* del *asset Wall3*

4. Wall4

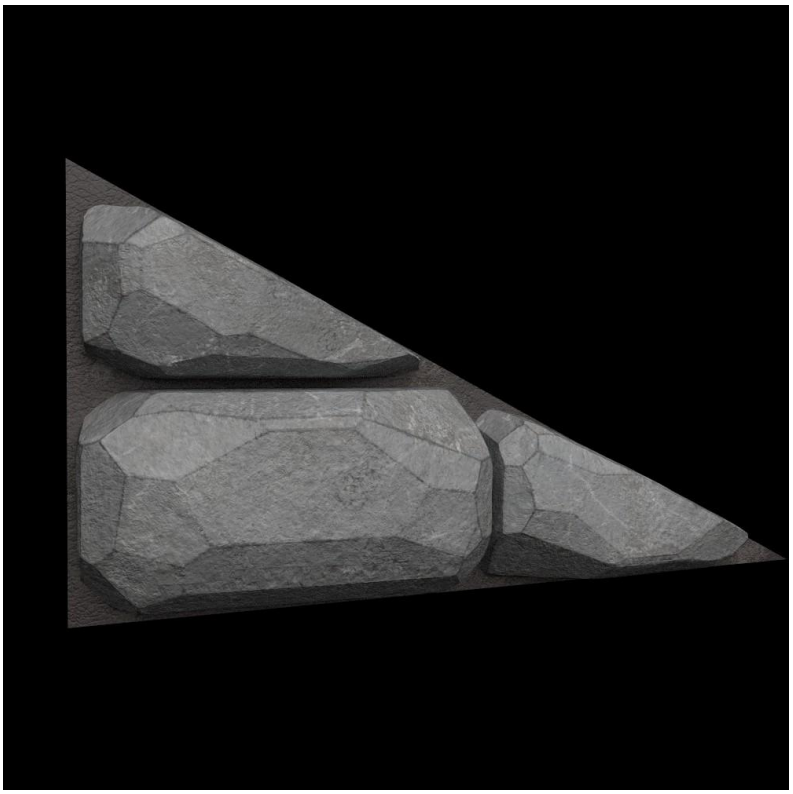


Figura 86: *Render* del *asset Wall4*

5. WallDoor

Figura 87: *Render* del *asset WallDoor*

6. DoorFrame

Figura 88: *Render* del *asset DoorFrame*

7. WindowFrame

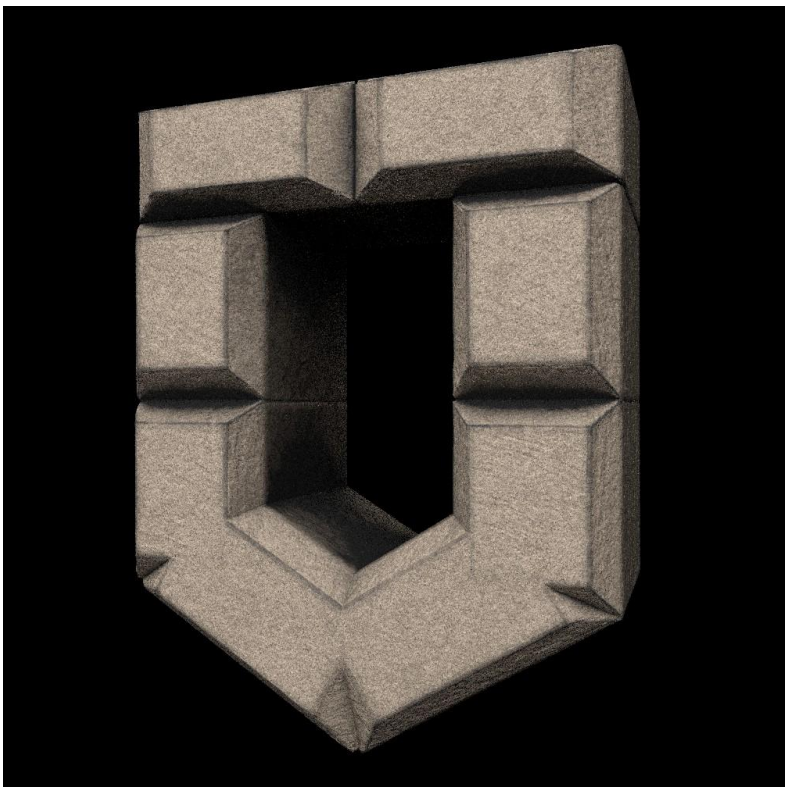


Figura 89: *Render* del *asset WindowFrame*

8. WallWindow

Figura 90: *Render* del *asset WallWindow*

9. Ceiling

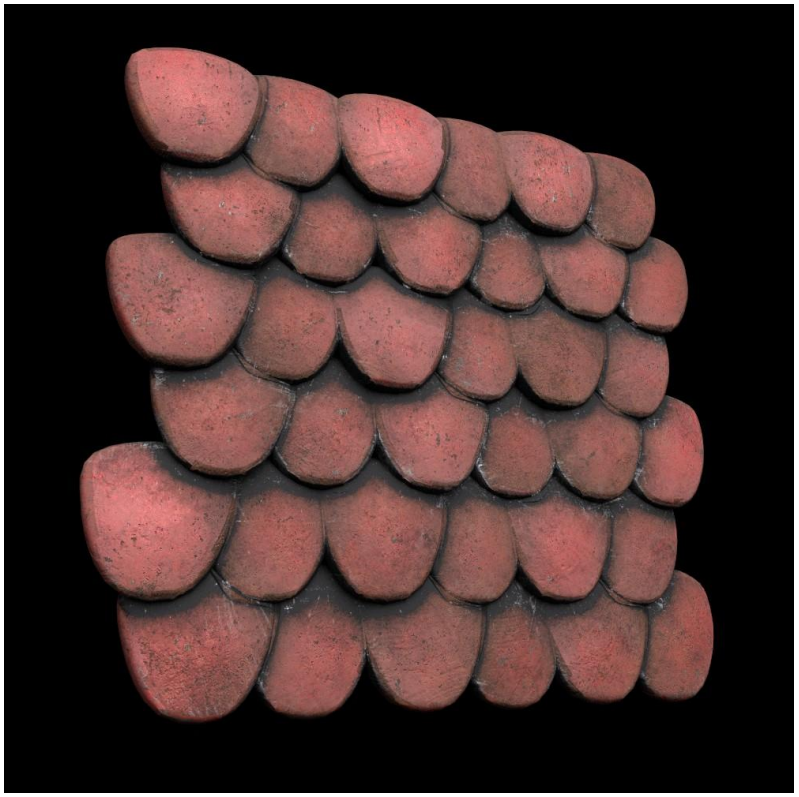


Figura 91: *Render* del *asset Ceiling*

10. Floor

Figura 92: *Render* del *asset Floor*

11. Column

Figura 93: *Render* del *asset Column*

12. ColumnBase

Figura 94: *Render* del *asset ColumnBase*

13. ColumnTop

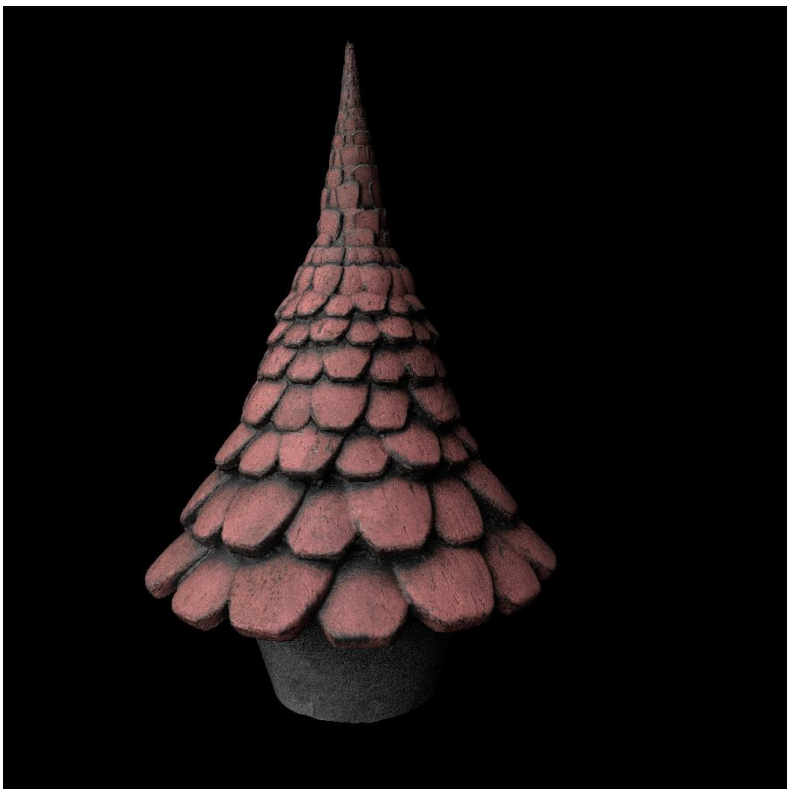


Figura 95: *Render* del *asset ColumnTop*

14. Column2

Figura 96: *Render* del *asset Column2*

15. Column2Base

Figura 97: *Render* del *asset Column2Base*

16. Column2Top

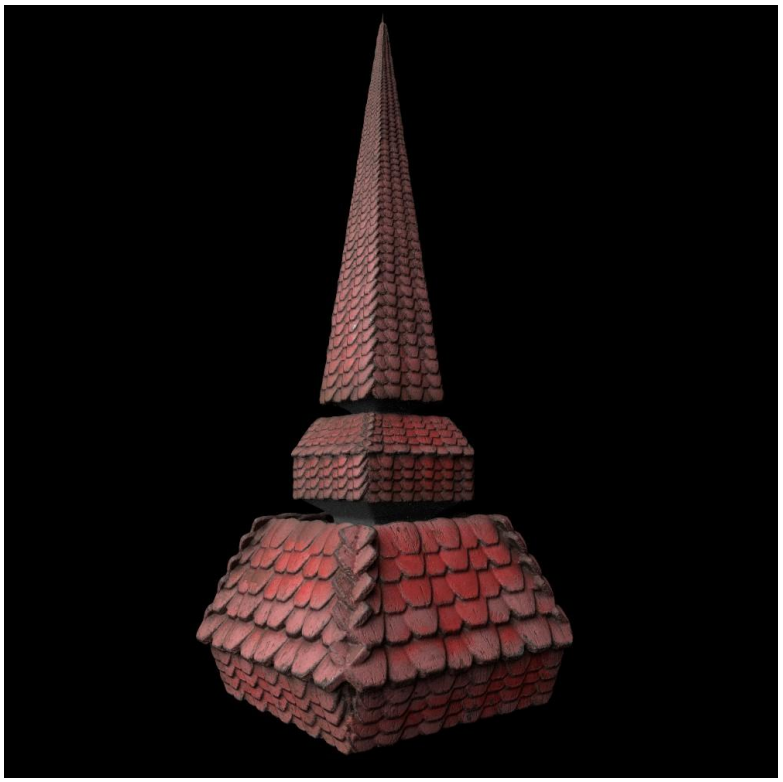


Figura 98: *Render* del *asset Column2Top*

17. ColumnBalcony

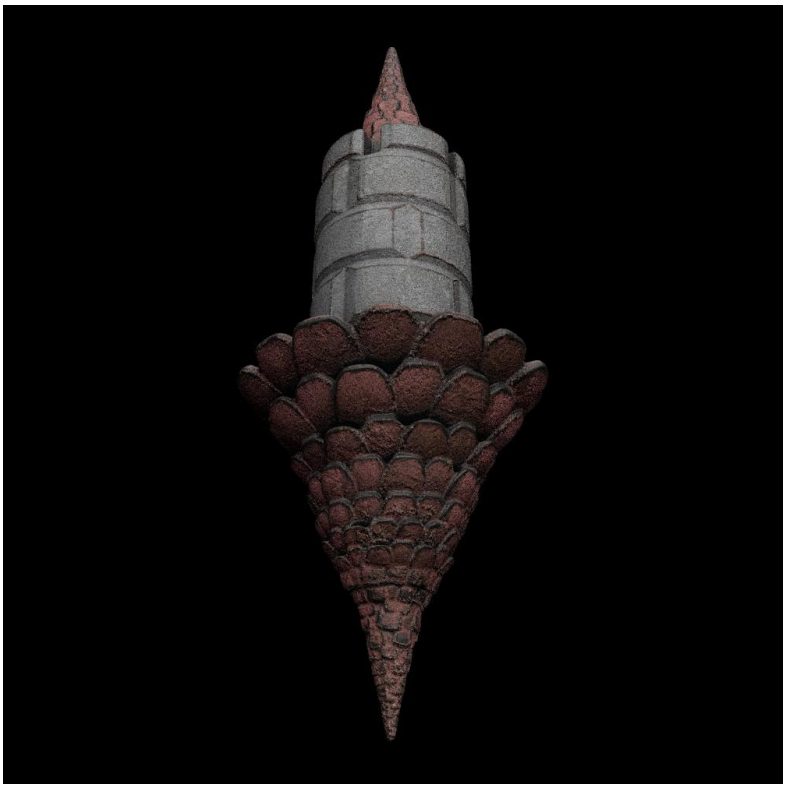


Figura 99: *Render* del *asset ColumnBalcony*

18. Railing

Figura 100: *Render* del *asset Railing*

19. Railing2

Figura 101: *Render* del *asset Railing2*

20. Stairs

Figura 102: *Render* del *asset Stairs*

21. Fountain

Figura 103: *Render* del *asset Fountain*

22. MagicTower

Figura 104: *Render* del *asset MagicTower*

23. GiantRock

Figura 105: *Render* del *asset GiantRock*