Universitat Oberta de Catalunya

uoc.edu

Space Engineer

Luis José de Ahumada Falcó
Grado en Multimedia
Videojuegos

Rafel Pérez Vidal
Joan Arnedo Moreno

05/01/2021

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Space Engineer*
Nombre del autor:	*Luis José de Ahumada Falcó*
Nombre del consultor/a:	*Rafel Pérez Vidal*
Nombre del PRA:	*Joan Arnedo Moreno*
Fecha de entrega (mm/aaaa):	01/2021
Titulación:	*Grado Multimedia*
Área del Trabajo Final:	*Videojuegos*
Idioma del trabajo:	*Castellano*
Palabras clave	*Unity, Oculus, VR*

Resumen del Trabajo:

La presente memoria describe el proceso de desarrollo de la experiencia Space Engineer, tratando todas las fases: preproducción, producción y posproducción.

Se ha creado bajo el motor de videojuego Unity Engine y con plataforma de destino Oculus Quest, que funciona bajo el sistema operativo Android.

Space Engineer, un *Escape Room VR* que nos pone en el papel de Ellie, una ingeniera que ha sido despertada de su *hipersueño* por una serie de fallos críticos en la nave. Ellie deberá arreglar todos los daños de la nave antes de que sea demasiado tarde y llegue al colapso. Durante la experiencia, nos acompañará SAI (Sistema de Ayuda Integrado), que nos ayudará a resolver las distintas misiones si nos quedamos atascados, funcionando a modo de sistema de pistas.

El proyecto trata de poner en jaque al jugador, teniendo que completar un total de 8 misiones e interactuar con más de 50 objetos del escenario, haciendo que tenga que desenvolverse en un espacio tridimensional desconocido, aprendiendo a usar los objetos e interpretando la información que recibe para realizar las misiones. Todo ello, bajo la presión de un porcentaje de integridad de la nave que no para de bajar conforme pasa el tiempo.

A nivel técnico, lo más destacable del proyecto es el gestor de misiones, el sistema de carga de las distintas zonas del escenario y un sistema propio de *culling*.

Abstract:

This report describes the development process of the Space Engineer experience, covering all phases: pre-production, production and post-production.

It has been created under the Unity Engine video game engine and with the Oculus Quest target platform, which works under the Android operating system.

Space Engineer, an Escape Room VR that puts us in the role of Ellie, an engineer who has been awakened from her hyper sleep by a series of critical failures in the ship. Ellie must fix all the damage to the ship before it's too late and collapses. During the experience, we will be accompanied by SAI (Integrated Help System), which will help us solve the different missions if we get stuck, working as a hint system.

The project tries to put the player in check, having to complete a total of 8 missions and interact with more than 50 objects on the stage, making him have to navigate in an unknown three-dimensional space, learning to use the objects and interpreting the information he receives to perform missions. All this, under the pressure of a percentage of integrity of the ship that does not stop decreasing as time passes.

On a technical level, the most remarkable thing about the project is the mission manager, the loading system for the different areas of the stage and its own culling system.

Índice

Figuras

1. Introducción

1.1 Contexto y justificación del Trabajo

Los videojuegos están cada vez más presentes en nuestro día a día. Tenemos entretenimiento en cualquier plataforma, sobre todo desde el *boom* de los smartphones. Tanto es así, que hasta los chips que se usan en los smartphones, han sido llevados a dispositivos creados específicamente para el entretenimiento, como las Oculus Quest, que funcionan bajo el Snapdragon 835 abriendo un nuevo mundo posibilidades de desarrollo en VR, plataforma en el que se apoya este TFG.

Como usuario de Oculus Quest y, por lo tanto, queriendo aprovechar la oportunidad para explorar las capacidades de las gafas y desarrollar una experiencia en el sector de la realidad virtual, sector que ha renacido gracias a este dispositivo debido a su calidad/precio.

El proyecto pertenece al género de los *Escape Room*, una actividad que está de moda en los últimos años. Gracias al tipo de dispositivo, se puede experimentar en crear una experiencia para crear esas sensaciones de coger objetos y poder manipularlos de forma virtual. Al mismo tiempo, también estoy interesado a nivel técnico en el rendimiento del dispositivo, por lo que quiero exprimirlo para ver cómo se comporta bajo una carga importante de procesado, siempre manteniendo unos límites aceptables de *frame rate*.

1.2 Objetivos del Trabajo

Los objetivos del TFG son varios, incluso algún objetivo primigenio tuvo que ser descartado por las limitaciones técnicas del dispositivo:

- Probar el sistema de *tracking inside-out* en un mismo espacio físico, pero con salas separadas.
- Crear un gestor que permita controlar el estado de las misiones sin necesidad de compilar.
- Crear un sistema de carga y descarga de las distintas zonas de escenario para mejorar el rendimiento del dispositivo.
- Crear un sistema de *culling* para el renderizado según donde mire el jugador.
- Crear una experiencia donde el jugador tenga que discurrir sobre el camino a seguir y cómo interpretar la información recibida implícita y explícitamente.
- Adquirir y experimentar con la interacción-usabilidad en la VR.

- Aprender sobre el SDK de Oculus VR integrado en Unity.

1.3 Enfoque y método seguido

El rumbo establecido desde el comienzo ha sido crear una experiencia nueva y no optar por continuar una existente. El simple hecho de tomar esta decisión, implica que debemos partir desde los cimientos de un proyecto, como es la etapa de conceptualización y preproducción. Le sigue la etapa de producción, donde se encuentra el grueso del trabajo y la mayor carga en coste temporal. El proyecto finaliza con la etapa de postproducción, donde se realizan los últimos ajustes del proyecto para tener una versión lo más estable y pulida. Si bien es cierto que hay que tener en cuenta que estamos ante un TFG en 2020 con lo que el COVID-19 ha supuesto en nuestras vidas, por lo que la fase de *testing* con jugadores no ha sido tan exhaustiva como se esperaba, sin embargo, ha sido satisfactoria.

Lo bueno de realizar un proyecto desde cero, es que la curva de aprendizaje es más agresiva y las primeras decisiones son las más vitales, por lo que exige tener mucho más claro el camino que va a llevar el proyecto, ya que tiene una fecha máxima y no puede haber giros 360 de última hora.

1.4 Planificación del Trabajo

Partiendo de la base que el proyecto se crearía desde cero, había que estudiar muy bien todas las etapas por la que iba a pasar el proyecto, ajustarlas al tiempo y a los recursos disponibles, siendo lo más realistas posibles y dejando de lado la ambición desmedida.

El proyecto tenía dos posibles caminos, A, usar el sistema de *tracking inside-out* y hacer una experiencia sobre un espacio físico separado por salas. B, realizar una experiencia a pie parado, usando un sistema de movimiento por *teleport* o *joysticks*. La opción A era la primera por su carácter innovador y por originalidad, ya que no había nada igual en el mundo. Más tarde, descubrí porqué era así, y es que, efectivamente, las Oculus Quest realizan un reposicionamiento virtual cuando percibe cambios de luz y de profundidad de entorno. Es decir, que en un espacio abierto (un pabellón) no habría ningún problema, pero en un espacio cerrado con múltiples salas (una casa) sí iba a aparecer este problema técnico.

El mayor inconveniente a esta traba es el problema que supone a nivel de seguridad para el usuario, ya que dichos reposicionamientos virtuales hacen que el espacio físico y virtual dejen de encajar, pudiendo provocar golpes tangibles por el usuario.

Finalmente, por motivos obvios, se optó por la opción B, siendo la opción desarrollada en el proyecto. La planificación se ha dividido en cinco fases: preproducción, producción I, producción II, postproducción, testeo y retoques finales. A continuación, se detalla las tareas programadas para cada fase de la planificación:

Fase 1 – Preproducción

Es la fase inicial del proyecto, marcará el camino a seguir a la fase de producción. Consta de los siguientes objetivos:

- Diseño de juego
  - Trama
  - Menú Inicio, Victoria, Derrota, Tutorial
  - Secuenciación de eventos
- Misiones y puzles
  - Escribir las misiones y puzles a realizar
  - Cómo se resuelven
  - Reutilizar puzles con ligeras variaciones
  - Mecánica de puzle
- Diseño de niveles
  - Diseño sobre plano de las salas
  - Distribución de *props*
  - Separación de alturas
- Búsqueda de recursos
  - Entorno 3D
  - *Props* 3D
  - VFX
  - SFX
  - Dev Tools
- Pruebas de SDK, física y recursos
  - Pruebas SDK Oculus
  - Testing *tracking inside-out*
  - Pruebas de físicas de gravedad, interacción entre capas
  - Testing de resources
- Testeo F1

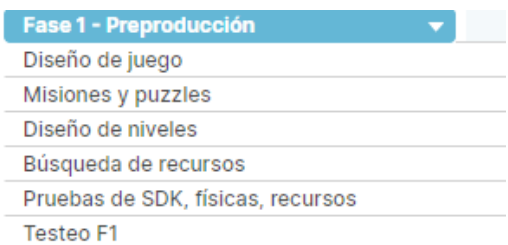

Figura 1: Fase 1 Preproducción

Fase 2 – Producción I

En esta segunda fase, entramos de lleno en el grueso de proyecto. Si bien es cierto que al final se obtuvieron los recursos del *environment* desde el Unity Asset Store y ahorramos tiempo en la tarea de *Modelado 3D environment*.

- Modelado 3D *environment* (finalmente composición de nivel)
  - Montar nivel con los *assets* descargados
  - Comprobación de materiales y resolución de texturas
- Modelado 3D *props*
  - Modelado en 3DS Max
  - UVW del modelo
  - Texturizado del modelo en Substance Painter
  - Importación a Unity
- Mecánicas base
  - Locomoción en VR
  - Acciones en el Menú Principal
  - Pantalla de Precarga
  - Sistema de pistas para SAI
  - Sistema de carga de zonas de escenario
  - Sistema propio de *culling*
- Testeo F2

Figura 2: Fase 2 Producción 1

Fase 3 – Producción II

Esta tercera fase va anidada a la segunda, pero esta se centra más en los puzles y misiones.

- Puzles
  - Scripting de misiones
  - Manager de misiones
  - Adaptación del *asset* de Switches VR a Oculus
  - Contador de tiempo
- Ambientación
  - GI de ambiente
  - Cambios de GI por eventos
- SAI
  - Escribir textos

  - Generador de voz de los textos
  - Modificación de la voz en Audacity
  - Importación en formato .ogg a Unity
  - Implementación en sistema de pistas
- Textos misiones
  - Implementar en script de misiones
  - Acoplar a la Tablet
- Testeo F3

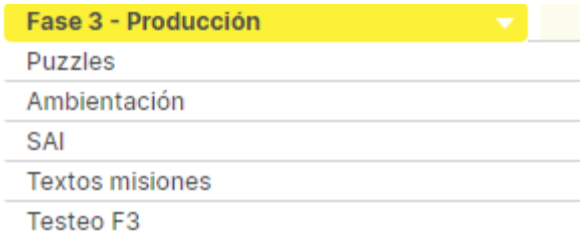

Figura 3: Fase 3 Producción 2

Fase 4 – Postproducción

Esta fase abarca los puntos de efectos especiales, efectos sonoros y de la GUI.

- VFX
  - Sistemas de partículas
- SFX
  - Efectos de sonido
  - Sonido ambiente
- GUI
  - Textos 3D en entorno mediante la herramienta Text Mesh Pro
- Testeo F4

Figura 4: Fase 4 Postproducción

Fase 5 – Testeo y retoques finales

En esta última fase abarcamos los aspectos de pulido del proyecto y de un testeo mucho más exhaustivo que los anteriores. Sin embargo, dada la situación de este 2020 provocada por la pandemia a nivel mundial del COVID-19, no se ha podido llevar a cabo este testeo como se esperaba. En este último tramo también se implementan sugerencias de los *testers*.

- Testeo completo
- Retoques necesarios
- Retoques opcionales
- Pruebas externas

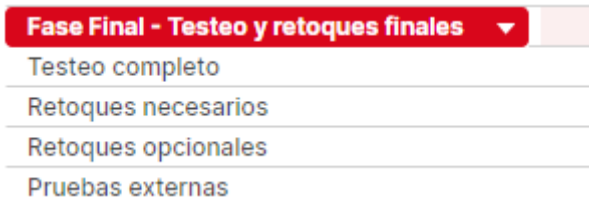

Figura 5: Fase 5 Testeo y retoques finales

Diagrama de Gantt completo:

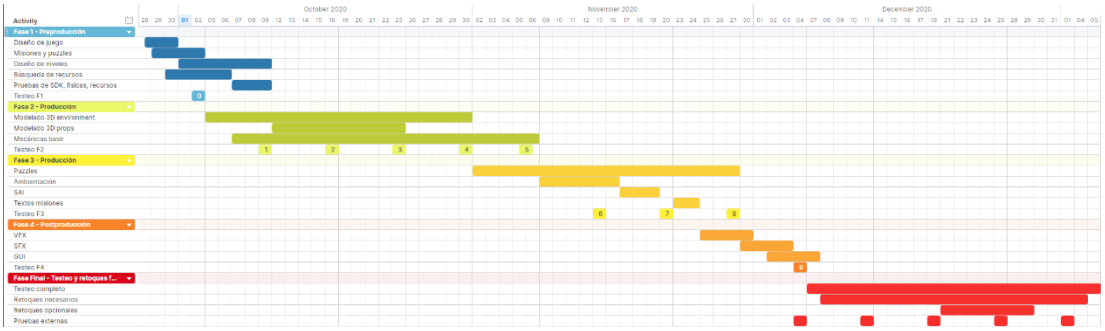

Figura 6: Diagrama de Gantt completo

1.5 Breve sumario de productos obtenidos

- 11 compilaciones de prueba
- Compilación Gold Master APK para Oculus Quest
- Video/Tráiler promocional
- 3 Gameplays

1.6 Breve descripción de los otros capítulos de la memoria

En el capítulo 2 se exponen las características que comparten los juegos del estilo Escape Room, tanto a nivel físico como virtual.

En el capitulo 3 nos adentramos en la definición del juego, la interacción con el jugador, la historia, las misiones. A fin de cuentas, trata sobre qué va el juego y qué contiene.

En el capítulo 4 profundizamos en apartados más técnicos como el sistema de carga, gestor de misiones, el *culling*, la arquitectura de juego entre otros aspectos. Es la parte más técnica de la memoria.

En el capítulo 5 se muestran las distintas zonas del escenario y habla de su modularidad, aspecto muy importante para la creación del juego y la escalabilidad que ofrece.

En el capítulo 6 se expone el manual del usuario, explicando la funcionalidad de las pantallas y los inputs de los controladores.

En el capítulo 7 se mostrarán los resultados de los tests con usuarios, así como los resultados obtenidos.

En el capítulo 8 se expondrán las conclusiones del proyecto, tanto a nivel personal como de planificación y de futuras mejoras.

En el capítulo 9 encontramos un breve glosario con algunos de los términos más relevantes de todo el documento.

2. Estado del arte

2.1 Género del juego

Space Engineer pertenece al género de los juegos de Realidad Virtual (VR) ya que, además de los controles de juego, su única plataforma de destino son las Oculus Quest, unas gafas de VR autónomas o *standalone*. Pero dentro de este gran género al cual pertenece por su plataforma final y tecnología empleada, su subgénero es el de Escape Room.

En los últimos años los Escape Room físicos han estado muy de moda, pero esto no es algo nuevo en el mundo de los videojuegos. Encontramos miles de juegos de este subgénero en el que es difícil marcar una fecha de inicio, ya que en antaño se encontraba en los juegos de mesa, como hoy en día también lo sigue haciendo la exitosa saga EXIT, de Devir.

Figura 7: Juegos Escape Room EXIT

Todos los juegos del estilo Escape Room comparten un mismo fin, encontrar la salida o el objetivo final resolviendo una serie de puzles durante el juego. Además de esa finalidad, este género se corresponde con una serie de mecánicas, elementos y entornos que refuerzan su género, como:

- Resolución de puzles
- Desbloqueo de sala/entornos/zonas de escenario
- Sistema de pistas
- Búsqueda de información útil en el entorno para descifrar puzles
- Usar ítems del entorno para
- Comprender, analizar el entorno y fusionar ideas

- Muerte si el contador principal llega a 0, suponiendo el fracaso del juego completo. A esto se le conoce como *permadeath* tanto en el mundo de los juegos de mesa como digitales.

Algo curioso a destacar, es que el subgénero de los Escape Room está muy presente en la VR, y eso se debe a la forma de poder interactuar con los objetos sumando inmersión, haciendo que sea más una experiencia que un videojuego. Es un tipo de juego muy agradecido para las plataformas de VR por el grado de inmersión que ofrecen con el entorno, que suelen ser habitaciones cerradas con mayor o menor detalle, por lo que a nivel de rendimiento no es tan exigente como puede ser un juego de otro género.

Además, una de las temáticas principales por la que se suelen decantar las desarrolladoras es por la del terror. Los Escape Room están ligados a que hay que escapar de algo o alguien, hay un elemento se cual sea su forma (entidad, finalidad) que debemos evitar. No precisamente tienen que haber sustos o un ente malvado, pero las escenas donde la oscuridad prevalece, aumentan la tensión y la inmersión en el jugador, sobre todo si es en VR.

2.2 Revisión sobre la tecnología

Actualmente existen dos grandes motores gráficos para la creación de contenido para VR: Unity, Unreal Engine y Cryengine. Ambos son dos opciones totalmente recomendables y válidas para realizar este proyecto. En el caso de Space Engineer, fue desarrollado bajo Unity, ya que su curva de aprendizaje es mucho más rápida, la documentación y comunidad es mucho más grande. Siempre ha habido rivalidades en la comunidad de desarrolladores entre ambos motores, sobre todo con el comentario hacia Unreal Engine de que “se ve mejor gráficamente de primeras que Unity”. Algo que, en realidad, es cierto. Sin embargo, con Unity se pueden lograr los mismos resultados o muy similares si se tienen los conocimientos necesarios.

Cryengine es un motor con una potencia desmesurada y bajo ningún control. Es un motor complicado, con una comunidad insignificante y que además no cuenta con soporte las Oculus Quest, sin embargo, si tuvo soporte para el SDK 0.8 de Oculus en 2015 [1].

A pesar de ello, Unity creo que es la decisión más acertada. Es todo un referente a nivel profesional y para este tipo de proyecto es perfecto, ya que tampoco estamos buscando una calidad Triple A. Su interfaz es muy intuitiva, fácil de manejar y cuenta con una gran cantidad de recursos en su tienda, el Asset Store.

3. Definición del juego

3.1 Subgénero y referencias a videojuegos existentes

Space Engineer pertenece al género de Realidad Virtual, y dentro del mismo al de Escape Room. El principal motivo que hace que pertenezca a este subgénero es porque su mecánica base es la resolución de puzles con un objetivo final, en este caso salvar la nave. Las referencias a videojuegos existentes no son muchas, pero sí me gustaría destacar dos juegos que me han hecho decantarme por este género.

Figura 8: Logo Mechanical Box

Mechanical Box (Caja Mecánica), es un juego de móvil que consiste en descifrar lo que hay dentro de una caja superando todos los niveles. Sus puzles son muy interesantes a nivel de lógica además de su estética, a pesar de tratarse de un juego en 2D. Lo que más destacaría de este juego es la complejidad de los puzles.

Figura 9: Logo Keep Talking and Nobody Explodes

Keep Talking And Nobody Explodes también es un gran referente a nivel de puzles. Este juego está disponible en varias plataformas, entre ellas las Oculus Quest. El juego consiste en desactivar una bomba que sólo puede ver un jugador, mientras tanto, el resto de jugadores deben descifrar los puzles (gracias al manual proporcionado por el juego) con la ayuda de las pistas que les del jugador de la bomba.

Mechanical Box y Keep Talking And Nobody Explodes pertenecen al género, pero no al estilo de juego. Por lo que tuve que buscar referencias que se asemejaran más a las mecánicas que quería conseguir una vez empezada la fase 1. Un gran ejemplo fue el de The Room, que a pesar de que este no es la versión en VR, sí que fue una gran inspiración en cuanto a referencia de puzles. Si bien es cierto que este juego sigue una interacción *point-and-click*, en Space Engineer este tipo de mecánica se hace al usar los interruptores para activar elementos del juego.

Figura 10: Puzle The Room

Obviamente no se pudo pasar por alto a su hermano en VR, The Room VR. Aquí me fijé más en la forma de interactuar con los objetos, si bien es cierto que utilizan un sistema de poses de mano distinto al que yo usé, que es mediante la detección de colisiones.

Figura 11: Puzle The Room VR

3.2 Tipo interacción juego-jugador

La interacción entre el juego y el jugador se establece mediante el contacto entre los objetos del escenario. Se ha querido mantener ese toque de realismo para dificultar la experiencia. Los elementos con los que se puede interactuar se perciben de forma explícita, que no hay elementos que puedan generar una confusión excesiva. Gracias a los mandos, podrá interactuar con todos (la gran mayoría) los objetos del escenario. La interacción principal se percibe cuando al acercarse a un objeto este se encuentra iluminado o con elementos destacables dado a entender que es manejable por parte del mismo. Todos los puzles dan

una señal de acierto cuando se están realizando, así el jugador sabrá si va por buen camino o no. El feedback por parte del juego es muy importante para que el jugador conozca su progreso.

Cuando se completa una misión siempre suena un sonido y la información de la Tablet cambia, al igual que en la sala centra la misión pasa a ponerse en color verde, previo estado en color rojo. Sumado a que SAI, avanza en su monólogo en relación a la misión. Otro punto importante es que las puertas se van desbloqueando conforme el jugador va completando misiones, así se evita que tenga que perder tiempo buscando por zonas que aún no tienen relevancia y evitar que caiga en la frustración.

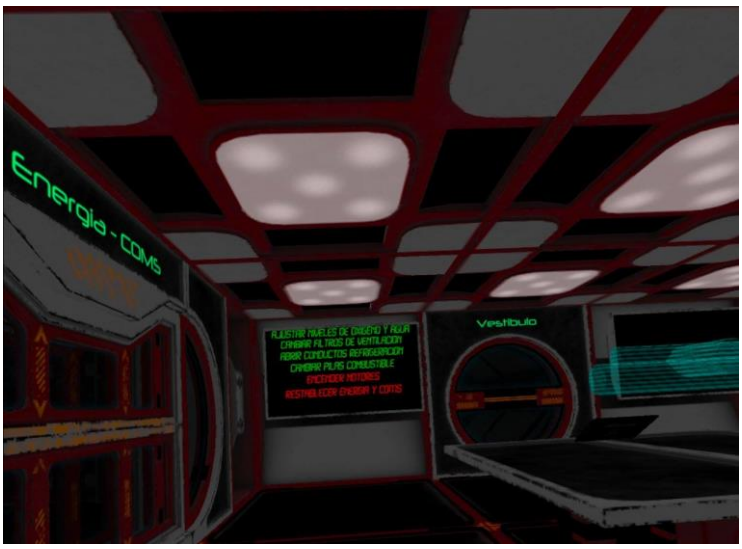

Figura 12: Sala central

3.3 Plataforma de destino

La plataforma de destino son las Oculus Quest, que funcionan bajo el sistema operativo de Android. Al principio la idea era usar el sistema de *tracking inside-out* pero como se dijo anteriormente, por limitaciones tecnológicas del dispositivo no ha podido ser posible.

El simple hecho de compilar para Android ya nos obliga a que el motor de juego tenga compatibilidad con el SDK de Android, esta es una de las grandes limitaciones que tiene Cryengine por ejemplo, sumado a la enorme complicación del motor.

3.4 Historia y ambientación

Los recursos en la Tierra se están agotando, por eso se crearon las naves colonizadoras con el fin de encontrar nuevos planetas habitables y enviarles las coordenadas de vuelta. El juego nos sitúa en una de las naves que han

abandonado la Tierra con el fin de colonizar otros planetas. Pero en la nave Igriv-II ha habido un fallo crítico que ha hecho despertar a Ellie, la ingeniera encargada de las reparaciones a bordo y, así, resolver los problemas para que pueda seguir su trayecto como estaba planeado.

La ambientación de Space Engineer es en el interior de la nave, si bien es cierto que tendremos vistas al espacio en algunas ventanas y la posibilidad de subir al casco de la nave, en este caso estaríamos en el espacio, para completar alguna misión. La mayoría de las zonas estarán iluminadas, salvo la primera que, estarán las luces de emergencia activadas (primera misión) y alguna otra zona que estamos a oscuras e iremos con una linterna.

La escalabilidad del juego se deja aquí abierta, ya que como se menciona en la historia que cuenta al inicio del juego, hay más naves por el espacio, por lo que puede haber más misiones en futuras actualizaciones y/o entregas. Sin embargo, dado del tiempo disponible, sólo se ha creado una de las naves. La historia, por lo tanto, continuaría con otros ingenieros del resto de naves, con nuevos puzles y misiones, creando además una trama entre todas ellas. En el caso de la Igriv-II, el satélite de comunicaciones es el culpable de todo, haciendo que Ellie tenga que tomar la decisión de lanzarse al espacio profundo con el satélite para salvar la nave y que sigua rumbo hacia su destino.

3.5 Definición de personajes

En Space Engineer no hay una trama/relación entre personajes, como podría ser el caso de The Last Of Us. Aquí sólo llevaremos a Ellie, la protagonista de esta historia. Eso no significa que estemos solos en la nave, SAI (Sistema de Ayuda Integrado) es una voz en off algo extrovertida que nos acompañará durante toda la aventura. SAI se encargará de darnos un informe de la situación de la nave, así como de información sobre la misión en la que estemos. Otra funcionalidad de SAI es que será el sistema de pistas, es decir, nos ayudará a resolver las misiones dándonos sugerencias.

3.6 Definición de elementos

Durante el juego vamos a tener que interactuar con diversos elementos, que, según su función, reaccionarán de una forma u otra. Esta es la lista de elementos con los que podremos interactuar:

- **Coger objetos para el inventario de utilidades**
    - Jeringuilla
        - Misión inicial
    - Tablet
        - Información de misión actual

  - Linterna
    - Ayuda en entornos oscuros
  - Cinta aislante
    - Posibilidad de arreglar fallos y aumentar integridad nave
  - Hacha
    - Empujar objetos
- **Coger objetos para el inventario de herramientas**
  - Soplete
    - Quita remaches misión final
  - Destornillador
    - Desatornilla en placas con tornillos
    - Atornilla de nuevo a la placa los tornillos
- **Coger objeto extra de misión**
  - Traje espacial
  - Vitaminas
  - Alicates
  - Martillo
  - Pistola silicona
- **Botón tipo Switch**
  - Activar/Desactivar
- **Botón tipo Push**
  - Activar/Desactivar
- **Dial**
  - Seleccionar un valor
- **Teclado numérico**
  - Poner código para desbloquear puertas
- **Filtros**
  - Verde, Rojo o Azul
    - Sustituir en posición de filtro roto
  - Roto
    - Quitar de su posición
  - Los filtros aumentan la integridad de la nave en un 1%
- **Pilas de combustible**
  - Nueva
    - Sustituir en posición de pila rota
  - Rota
    - Quitar de su posición y lanzar al vacío
  - Las pilas de combustible no pueden colisionar con absolutamente nada en la nave, sólo puede tocar la mano del jugador.
- **Baterías**
  - Se colocan en la zona de COMS
  - Aumentan en 1% la integridad de la nave
- **Elementos flotantes (miscelánea)**

  - Bombonas
  - Alicates
  - Tapas de caja contenedora
- **Elementos del almacén**
  - Placas de pila de fusión
  - Tornillos para placa de pila de fusión
  - Pila de fusión
- **Pila de fusión**

3.7 Secuencia de misiones

En el siguiente punto se detalla la secuencia lineal de Space Engineer en cuanto a la resolución de las misiones:

1. Ellie despierta del hipersueño y avanza por un pasillo oscuro. Llega a una sala (vestíbulo) en la que SAI le presenta la situación. Debe coger todos los elementos de la mesa, colocarse el traje, inyectarse la adrenalina y reactivar la energía. Una vez hecho todo esto, la puerta se desbloqueará.
2. En la siguiente sala (sala central) debe pulsar el botón de la mesa y realizar el diagnóstico de la nave. En una de las esquinas hay un rack con 4 switches que deberá activar. Ahora ya le aparece la información la Tablet.
3. Entrará a la sala de Oxígeno y Refrigeración, girar los 4 diales hasta el 100% para completar la misión.
4. Entrará a la siguiente sala y colocar los filtros en su posición. 5 filtros rojos, 3 verdes y 3 azules. En la planta superior se encuentran más filtros por colocar, no es necesario, peri si se quiere aumentar la integridad de la nave cada filtro lo hará en 1%.
5. Seguirá avanzando y deberá bajar las escaleras. Aquí tendrá que poner el código para desbloquear la puerta, que será la suma hasta llegar a 21, por ejemplo 10+10+1 o 5+5+3+2+4+1+1, cualquier combinación es posible mientras el resultado sea 21. En la sala central se encuentra la pista apuntada en la mesa. Una vez la puerta está desbloqueada, deberá girar los diales hasta el valor correcto, una luz verde se encenderá si ha dado con dicho valor. En los tubos están apuntados los valores a mano a modo de pista.
6. Ahora deberá dirigirse a la sala de motores para cambiar las pilas de combustible. Primero deberá quitar las pilas rotas, lanzarlas al espacio a través de la sala de vacío y luego colocar las pilas nuevas que están en el almacén. Todo ello con mucho cuidado de no tocar ninguna superficie ya que pueden explotar, tanto las rotas como las nuevas.

7. Colocadas las pilas de combustible, toca arrancar los motores. Pero primero será necesario cambiar la pila de fusión que se encuentra en la cubierta de la nave (exterior). Para ello, tendrá que ir al almacén y coger las herramientas que se encuentran dentro de una caja si no lo ha hecho ya. También deberá coger una pila de fusión del almacén. Esta es una misión que requiere habilidad y organización, además de precisión, ya que los tornillos y la placa se pueden perder en el espacio debido a la falta de gravedad. Por eso hay de recambio en el almacén, pero unidades limitadas. Una vez cambiada la pila de fusión, que SAI se lo dirá, deberá bajar a la sala de arranque y activar todos los switches para que el botón de arranque funcione. Posteriormente, presionar el botón de arranque.
8. Al presionar el botón de arranque, habrá un fallo eléctrico y deberá dirigirse a la sala de COMS y Energía. Allí tendrá que colocar al menos 10 baterías que se encuentran en dos cajas repartidas en la sala. Puede colocar todas si quiere y aumentar la integridad de la nave en 1% por batería. También deberá reiniciar los 6 servidores de comunicaciones repartidos por la sala. Podrá guiarse por el sonido, ya que estos emiten un pitido con configuración espacial 3D. Cuando llegue a cada servidor, deberá pulsar el switch que parará el pitido y arreglará el servidor.
9. Una vez puestas las 10 baterías y reiniciado los 6 servidores, sonará una explosión del exterior. Ellie tendrá que salir a comprobar qué sucede. El satélite habrá sufrido daños graves, y con habilidad de no caer en el espacio, tendrá que llegar hasta él teniendo que desacoplarlo para salvar la nave. 30 segundos más tarde, termina el juego y pasará a la pantalla de victoria.

3.8 Objetivos planteados al jugador

El juego, al estar en el subgénero de los Escape Room, se le irán desbloqueando nuevas misiones y tareas que podrá consultar en su Tablet. Tendrá que ir completando cada una de las misiones para poder acceder a la siguiente. El objetivo principal del juego es que el jugador sea capaz de desenvolverse en un entorno desconocido lleno de tareas que completar, todo dentro de un tiempo.

Es por eso que el juego no tendrá modo de guardado, el jugador debe completar todo el nivel de una sola tirada, por ese motivo los niveles no tendrán una duración excesiva. Este primer nivel, a tiro hecho, dura unos 30 minutos.

3.9 Concept Art

Estas son algunas de las referencias que sirvieron de inspiración a la hora de componer el apartado gráfico:

Figura 13: Echo VR

Figura 14: Solaris

Figura 15: Tacoma

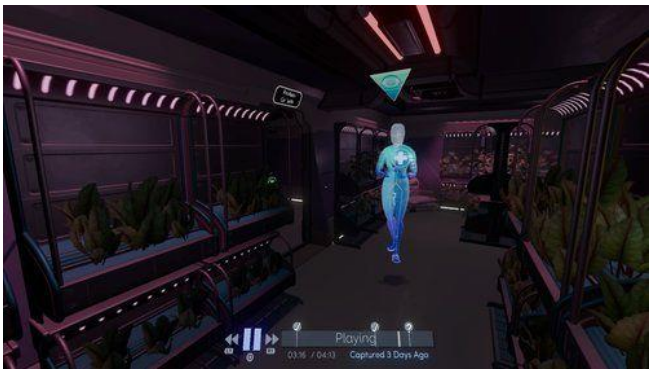

Figura 16: Tacoma

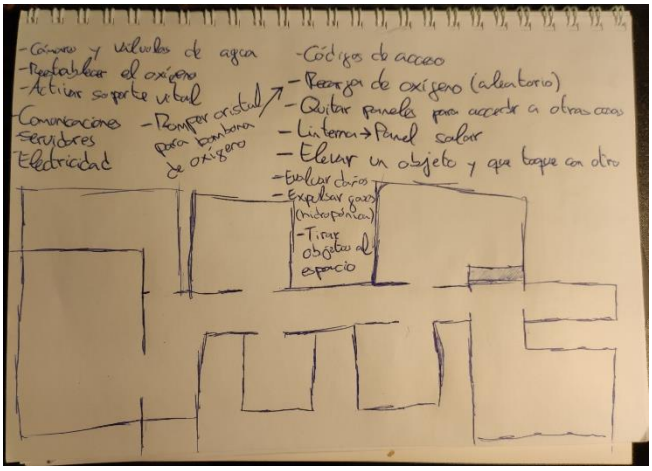

Figura 17: Concept tracking inside-out

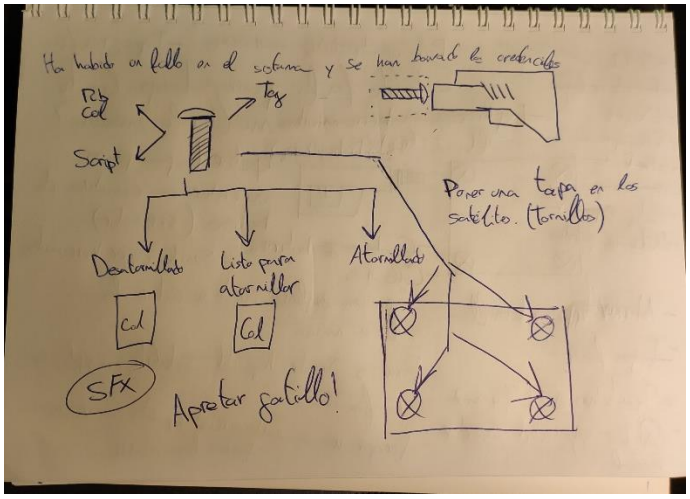

Figura 18: Concept destornillador

4. Diseño técnico

4.1 Entorno escogido

En cuanto al entorno escogido para el desarrollo de videojuego ha sido el motor de juego Unity, y más concretamente la versión Unity 2019.1.4f1, ya que ofrece una serie de ventajas frente a sus competidores:

- Curva de aprendizaje sencilla y rápida
- Programación en C#, del cual dispongo conocimientos
- Interfaz sencilla e intuitiva
- El motor con la mayor comunidad en internet
- Gran cantidad de recursos en su tienda virtual
- Integración perfecta con el SDK de Oculus

Unity es un motor muy versátil y fácil de manejar a nivel principiante y medio, lo cual supone una gran ventaja a encontrar problemas debidos al desconocimiento como podría suceder en Unreal Engine, que además funciona bajo el lenguaje C++, del cual tengo conocimientos muy básicos.

4.2. Requerimientos técnicos del entorno de desarrollo

Para obtener los requisitos a nivel de software y hardware he consultado la web oficial de Unity [2] para la versión que se ha usado en el proyecto (2019.1.4.f1), que pertenece a los requisitos de la serie 2019.1:

System	Minimum requirements
Operating system	**Windows:** 7 SP1+, 8, 10, 64-bit versions only **macOS:** 10.12+ **Linux:** Fixed at: Ubuntu 16.04, 18.04 and CentOS 7 Server versions of Windows and OS X are untested.
CPU	SSE2 instruction set support.
GPU	Graphics card with DX10 (shader model 4.0) capabilities.
Devices	
iOS	Mac computer running minimum macOS 10.12.6 and Xcode 9.4 or higher.
Android	Android SDK and Java Development Kit (JDK). IL2CPP scripting backend requires Android NDK.
Universal Windows Platform	Windows 10 (64-bit), Visual Studio 2015 with C++ Tools component or later and Windows 10 SDK.

Este proyecto ha sido desarrollado bajo el sistema operativo Windows 10 64-bit, usando el SDK de Android y Visual Studio Community 2017.

4.3 Inventario y breve descripción de herramientas empleadas

En asociación con el motor Unity, estos han sido los programas que también han formado parte del desarrollo del proyecto:

Autodesk 3DS Max 2020 → Software de modelado 3D, que nos permite crear cualquier objeto tridimensional y crear su mapa de UWV posteriormente.

Substance Painter 2019 → Software de texturizado y pintado digital, permite pintar el modelo exportado desde un software de modelado 3D. Es sencillo de utilizar y el texturizado es muy rápido.

Audacity → Software de edición de audio, gratuito y muy sencillo de manejar. Permite editar audios, recortar, añadir efectos, componer varios clips… Se ha usado sobre todo para modular la voz de SAI.

Microsoft Visual Studio Community 2017 → Es el software por excelencia de edición código de Microsoft. En el instalador de Unity, da la opción de su instalación. Hace uso del *intelliJ* con Unity, lo que significa tiempo de ahorro a la hora de escribir código con la API de Unity en C#.

Adobe Photoshop CC → Software de edición de imágenes de adobe.

Draw.io → Software gratuito para la creación de diagramas de flujo.

Tom's planner → Software en línea para la creación de diagramas de Gantt.

Google Drive → Medio de almacenamiento en red para subir las copias de seguridad del proyecto.
Voz Loquendo → Software en línea que permite generar voces desde un texto.

Oculus Quest → Dispositivo de realidad virtual *standalone* que cuenta con sensores internos que permiten el tracking de los controladores y del medio físico. Funciona bajo el sistema operativo Android y cuenta con un SoC Snapdragon 835, 4GB de RAM y 64 GB de ROM.

4.4 Inventario de assets y recursos utilizados

4.4.1 Modelos 3D *environment*

- **Modular sci-fi Environment A** de Ogasoda, obtenido desde el Asset Store
- **Modular sci-fi Environment B** de Ogasoda, obtenido desde el Asset Store
- **Modular sci-fi Environment D** de Ogasoda, obtenido desde el Asset Store
- **Modular sci-fi Environment E** de Ogasoda, obtenido desde el Asset Store
- **Modular sci-fi Environment F_C1** de Ogasoda, obtenido desde el Asset Store
- **Modular sci-fi Environment G_C1** de Ogasoda, obtenido desde el Asset Store

4.4.2 Modelos 3D *props*

- **Sci-fi cargo container**, obtenido desde el Asset Store

Figura 19: Sci-fi cargo container

- **MSFMC - Radar Dish**, obtenido desde el Asset Store

Figura 20: MSFMC - Radar Dish

- **Free Tools Kit**, obtenido desde el Asset Store

Figura 21: Free Tools Kit

- **Energy Cells Free**, obtenido desde el Asset Store

Figura 22: Energy Cells Free

- **Sci-fi Crate**, obtenido desde el Asset Store

Figura 23: Sci-fi Crate

- **HiTech SciFi Energy Cell**, obtenido desde el Asset Store

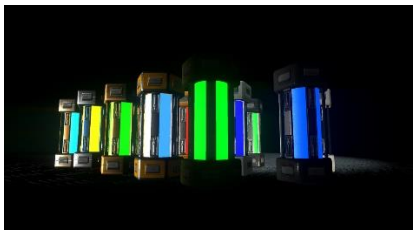

Figura 24: HiTech SciFi Energy Cell

- **Sci-Fi Tablet Computer**, obtenido desde el Asset Store

Figura 25: Sci-Fi Tablet Computer

- **Fire Axe Free**, obtenido desde el Asset Store

Figura 26: Fire Axe Free

- **Free Laboratory Pack**, obtenido desde el Asset Store

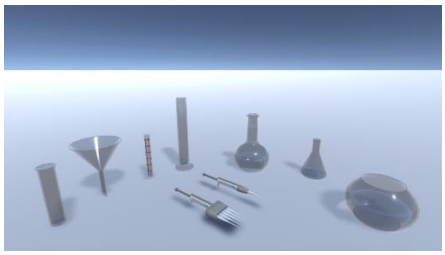

Figura 27: Free Laboratory Pack

- **Power Cell AAA**, obtenido desde el Asset Store

Figura 28: Power Cell AAA

- **LowPoly Scifi Pistol**, obtenido desde el Asset Store

Figura 29: LowPoly Scifi Pistol

- **SciFi Handlight Free [HQ, Lowpoly, PBR]**, obtenido desde el Asset Store

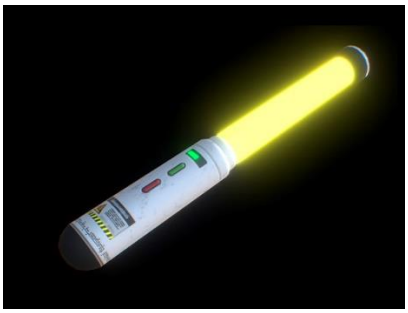

Figura 30: SciFi Handlight Free [HQ, Lowpoly, PBR]

- **Buttons and Switches**, obtenido desde el Asset Store, con adaptación propia a las Oculus Quest

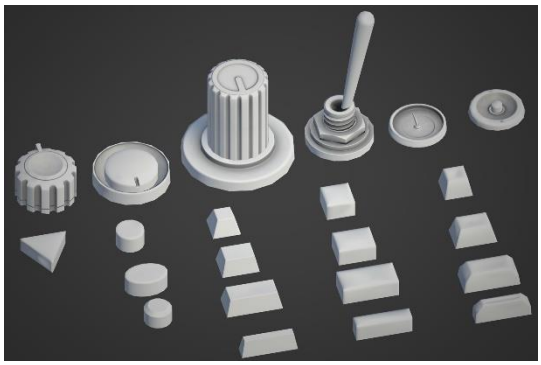

Figura 31: Buttons and Switches

- **Astronaut White Suite**, obtenido desde Sketchfab

Figura 32: Astronaut White Suite

- **Estación base misión final**, creado en 3DS Max y texturizado en Substance Painter. A continuación, pasamos a explicar el proceso de creación.
  Para comenzar a modelar empezamos con formas primitivas, como lo puede ser un cubo o un cilindro. Lo primero es hacer un modelo en bajo poligonaje o low poly, esto sirve para tener una forma básica y sencilla del modelo para posteriormente añadirle más detalle. En realidad, lo que se hace es añadirle un modificador llamado Turbosmooth, gracias a este la

malla del objeto aumenta según nuestras preferencias y se obtiene el modelo high poly, aunque en este caso este paso no ha sido necesario.

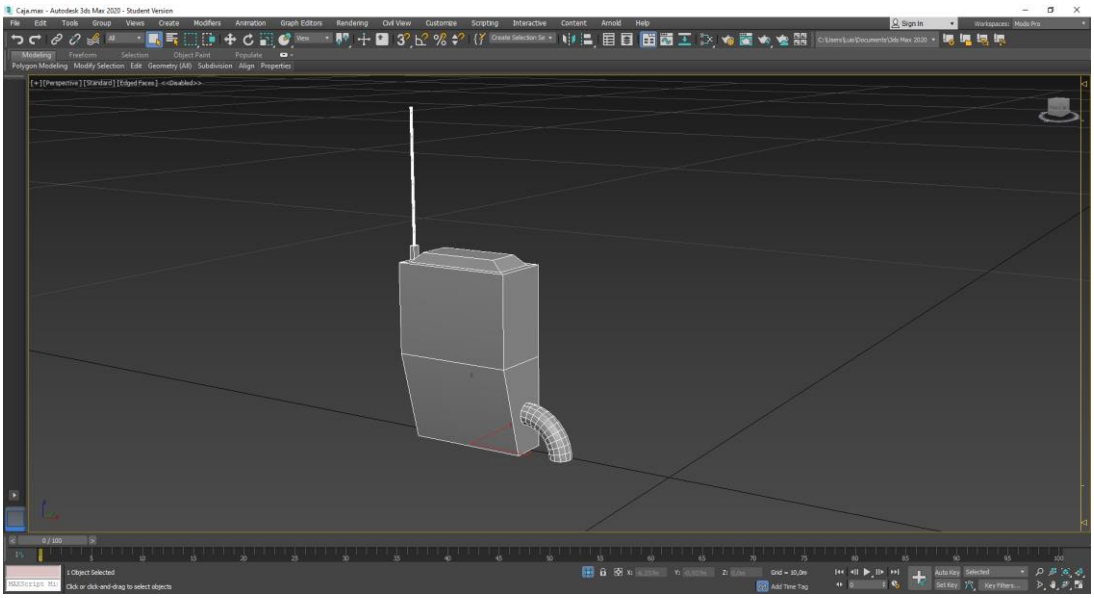

Figura 33: Proceso modelado 3D

Mi flujo de trabajo en el modelado 3D es siempre modelar pensando en cómo se va a pintar. A la hora de modelar pienso en ahorrar el mayor tiempo posible para la parte del texturizado, ya que Substance Painter te permite seleccionar geometría u objetos dentro de la propia geometría del modelo, esto es algo muy positivo que tiene este programa ya que con un par de clicks podemos tener un modelo prácticamente pintado al 100%. Por eso mismo, mientras se esté modelando o al final del modelado conviene separar aquellas partes que sepamos con certeza (o no, no pasa nada) que vayan a ir con distintos materiales o colores. Lo bueno es que todo eso estará dentro de un mismo modelo 3D y por tanto será el mismo mapa de UV.

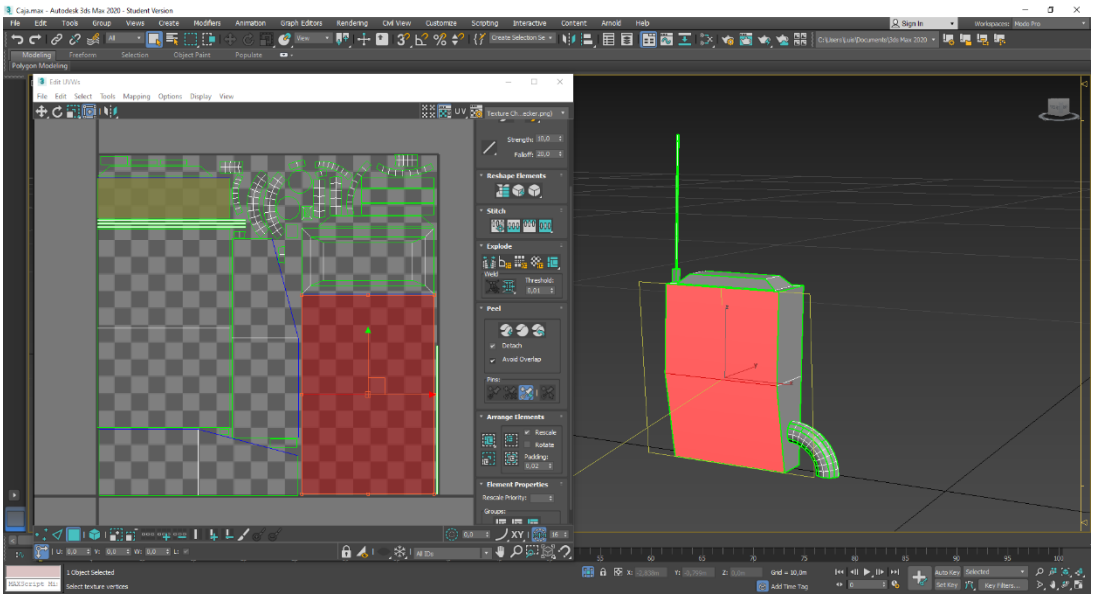

Figura 34: Proceso UWV

Sin embargo, para que el resultado sea lo óptimo posible y consigamos la mejor calidad, el mapa de UV debe estar bien trabajado. Para hacerlo bien, lo que tenemos que hacer es separar bien las piezas y extender en el mapa de UV aquellas que sean las más se van a ver, no es lo mismo un tornillo que una cara, por ejemplo.
Como se ha mencionado anteriormente, a la hora de trabajar los UVs hay que tener muy claro que objetos son más importantes que otros. A veces y, si la malla no es compleja, se pueden hacer unos cortes automáticamente. El problema es que esto ocurre en raras ocasiones y siempre hay alguna/s pieza/s que hay que modificar. Lo positivo de este procedimiento es que hay ciertas piezas de la malla que a lo mejor no son tan relevantes, son pequeñas y/o llevan colores planos, en ese caso nos dará igual cómo se haya hecho el corte.

Gracias al tipo de modelado empleado (malla de objetos) podemos pintar de una manera muy rápida aplicando los materiales deseados. Una vez aplicados se ajustan al gusto.

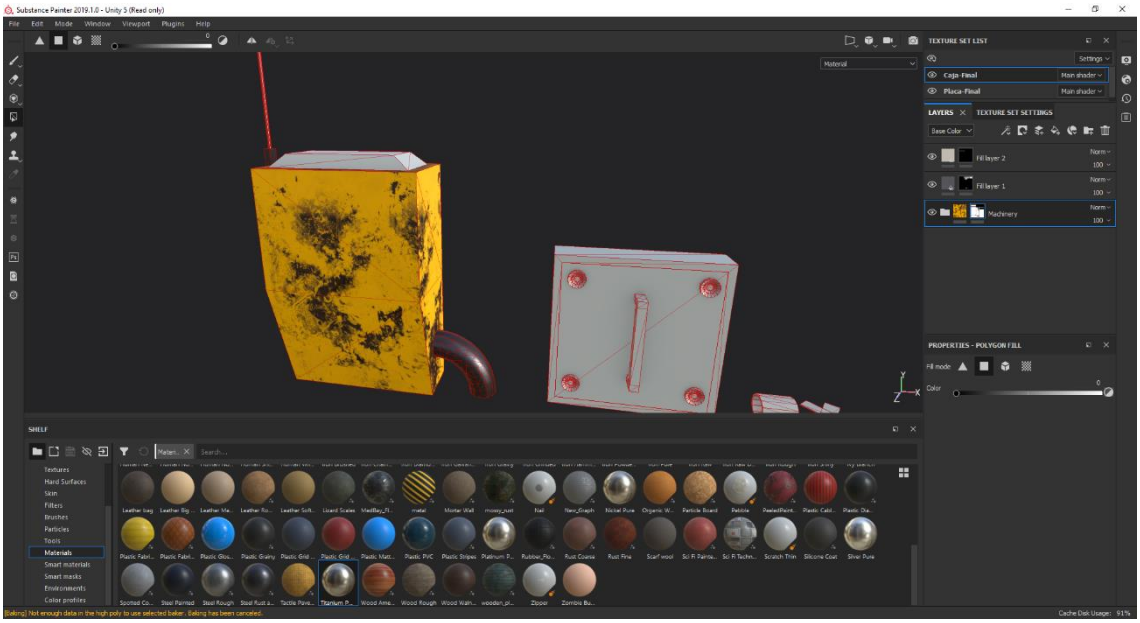

Figura 35: Proceso texturizado 1

Dando detalles con Hardsurface:

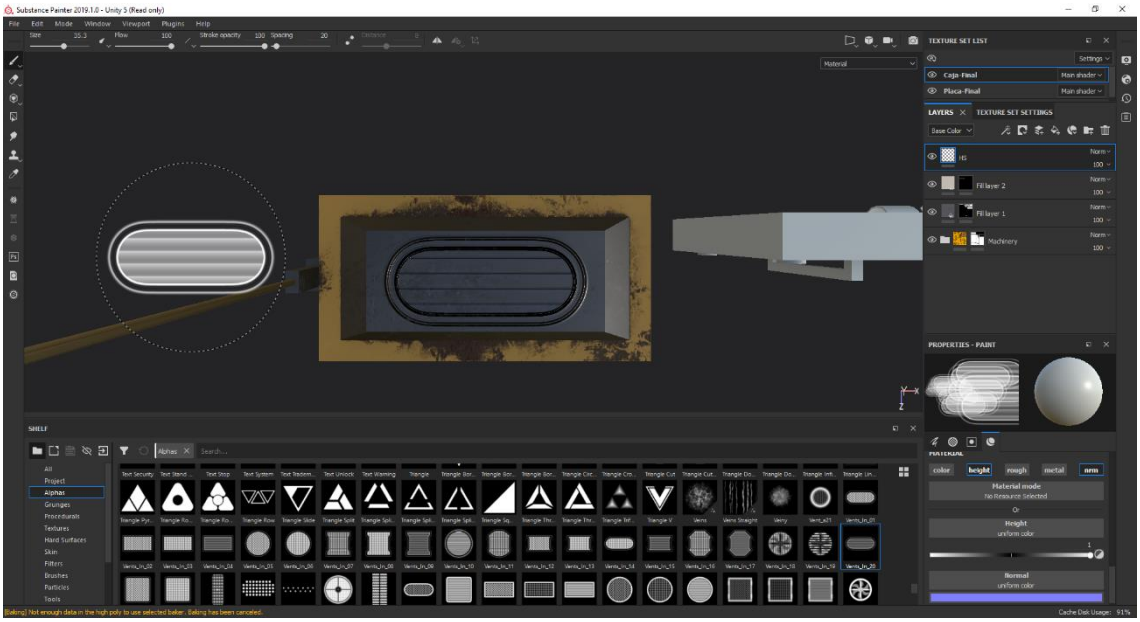

Figura 36: Proceso texturizado 2

Añadiendo detalles de textura:

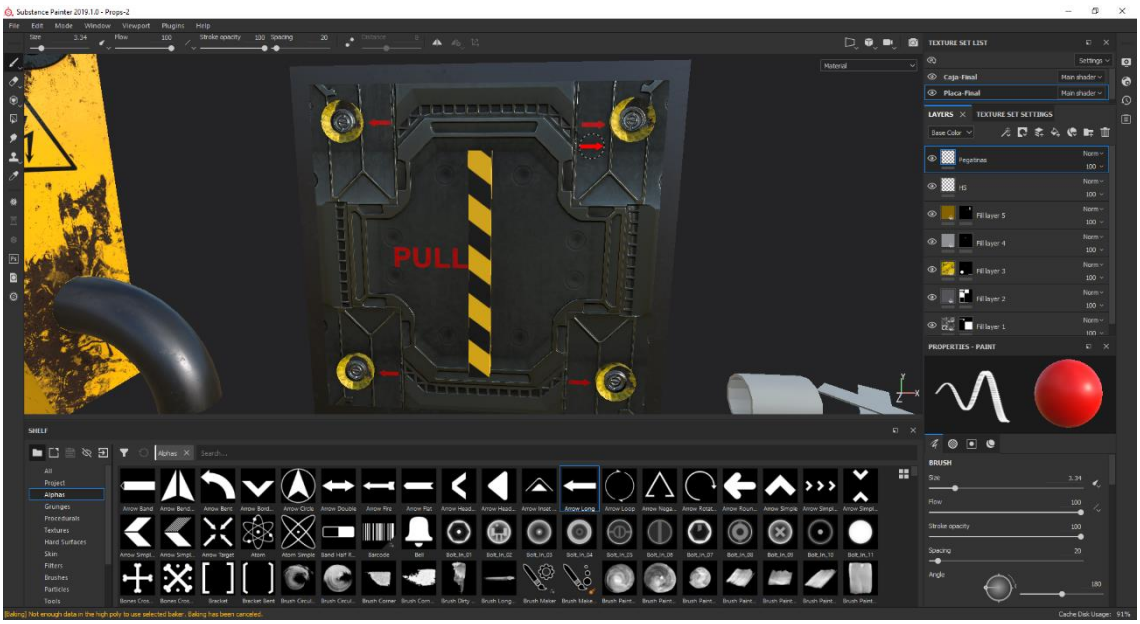

Figura 37: Proceso texturizado 3

Finalmente exportamos las texturas con el preset de Unity que tiene Substance Painter.

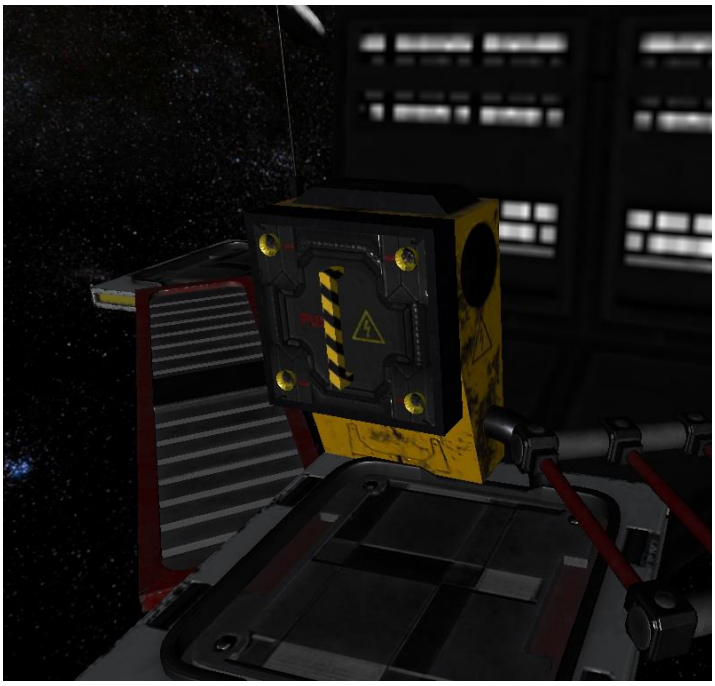

Figura 38: Importación Unity

- **Servidores modulares**, creados en 3DS Max y texturizados en Substance Painter. Hay hasta 6 variantes del modelo, gracias a su proceso de creación modular.

Figura 39: Servidor

4.4.3 Sonidos

- **Error-Alerta**, *TheLibrarybyMTC_UI_Computer_094* de TheLibrarybyMTC, GDC Game Audio Bundle 2016

- **Mision-Completada**, *TheLibrarybyMTC_UI_Alert_01* de TheLibrarybyMTC, GDC Game Audio Bundle 2016
- **Ambiente**, *Spaces - Deep Space Rumble_01* de SoundMorph, GDC Game Audio Bundle 2017

- **Destornillador Start y Destornillador Loop**, *electrical-screwdriver* de NWSP, Freesound.org

- **Explosion-Pila-Combustible**, *Heavy Magical Explosion_SI 03* de Mechanical Wave, GDC Game Audio Bundle 2016

- **Jeringuilla**, *Gore,Slime,Creature,Meat,Blood,Short,Movement,Various13* de Mattia Cellotto - The Borax Experiment, GDC Game Audio Bundle 2016

- **Diagnostico**, *Modular UI - Data Transmissions-001* de SoundMorph by Richard Devine, GDC Game Audio Bundle 2016

- **CrugidosNave-1**, *TheLibrarybyMTC_Robo_Metal_HatchDeep_006* de TheLibrarybyMTC, GDC Game Audio Bundle 2016

- **CrugidosNave-2**, *metal, heavy creaks, shipwreck, submarine deformation under pressure 07* de Alexander Kopeikin - Black Metal, GDC Game Audio Bundle 2016

- **Pitido-Servidor**, *Science fiction spaceship alarm 2* de ZapSplat

- **Susto-1-m7**, *MUG_Metal_TV_Tray_Drops_Series002* de Griffin Sound Design – Mugs, GDC Game Audio Bundle 2017

- **Explosion-M8**, *Spaces - Space Explosions_01* de SoundMorph – Spaces, GDC Game Audio Bundle 2017

- **Explosion-Final**, *explosion_large_08* de Gamemaster Audio - Explosion Sound Pack, GDC Game Audio Bundle 2017

- **Puerta-1**, *5302 - Electric Starship Door 02* de The Sound Pack Tree - The Sound Pack Tree, GDC Game Audio Bundle 2017

- **Puerta-2**, *5366 - Space Lift Door 02* de The Sound Pack Tree - The Sound Pack Tree, GDC Game Audio Bundle 2017

- **Puerta-3**, *Door - Sci-Fi – 04* de SoundMorph – Portals, GDC Game Audio Bundle 2017

- **Puerta-4**, *Door - SpaceShip_hatchway_06* de SoundMorph – Portals, GDC Game Audio Bundle 2017

- **Puerta-5**, *door_SpaceShip_hatchway_04* de SoundMorph - Solar Sky, GDC Game Audio Bundle 2016

- **Soplete-Start y Soplete-Loop**, *Afterburner* de TiesWijnen, Freesound.org

- **Steam-Pipe**, *Steam Pipe Blowout* de magnuswaker, Freesound.org

- **SAI**-(…), los 40 archivos de SAI han sido generados mediante la herramienta en línea de Voz Loquendo. Se ha escogido el idioma español y la voz de la chica "Mónica". Posteriormente se ha capturado el audio interno desde Audacity (modo WASAPI). Una vez capturado, se ha amplificado la señal, se ha duplicado la pista, y la pista duplicada se ha desplazado 26 milisegundos para dar ese efecto de voz robótica.

4.4.4 VFX

- Sistemas de partículas de los **Standard Assets** de Unity, obtenido desde Asset Store
- **Milky Way SkyBox**, de Adam Bielecki, obtenido desde Asset Store
- **Sci-fi UI Arsenal** (Sci-Fi MEGA PACK), de JFekb, obtenido desde Asset Store

4.4.5 Fuentes

- **LCD Feb. mono**, de Samuel Reynolds, DaFont.com
- **Neuropol X**, de Typodermic Fonts, DaFont.com
- **New Slender Mans Writing**, de Paul Kerssenbrock, DaFont.com
- **Potra**, de Alejo Bergmann, DaFont.com
- **Commodore 64**, de Devin Cook, DaFont.com

4.4.6 Extras

- Oculus Integration, de Oculus, Asset Store

4.5 Arquitectura del juego

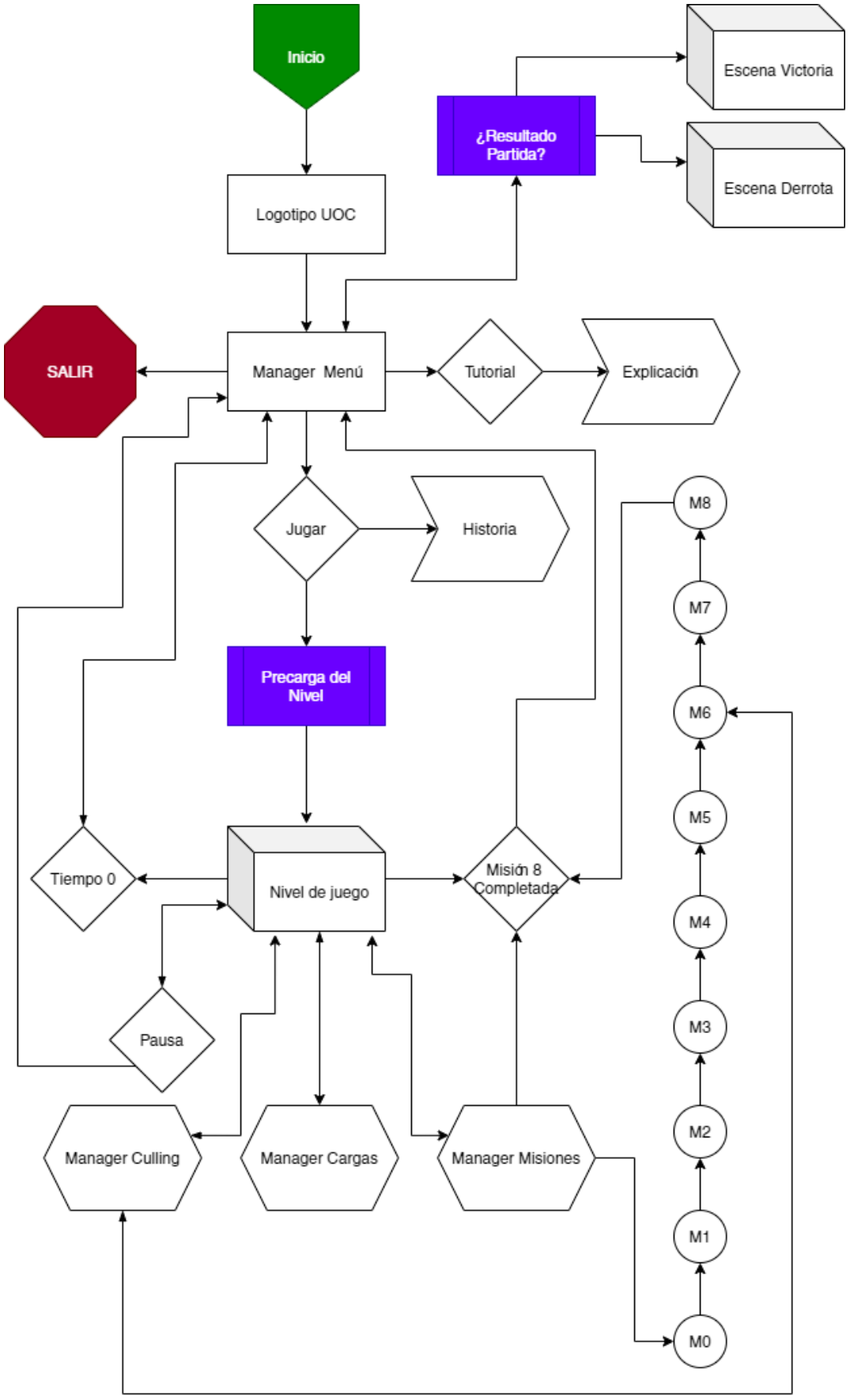

Figura 40: Arquitectura juego

4.6 Manager de Misiones

Para gestionar las misiones se ha creado un script que es el controlador o *manager* que se encarga de revisar y actualizar el estado de cada una de ellas. Este script está comunicado con el script específico de cada misión, y este último es quién gestiona su misión de forma interna, comunicándose a su vez con los mecanismos que completan la misión (switches, botones, diales, filtros…). Cuando el script de misión comprueba que se han completado los requisitos para cumplirla, este le envía un "mensaje" al manager de misiones diciéndole que su misión ha sido completada. Acto seguido, el manager de misiones cierra esa misión completada y abre la siguiente misión.

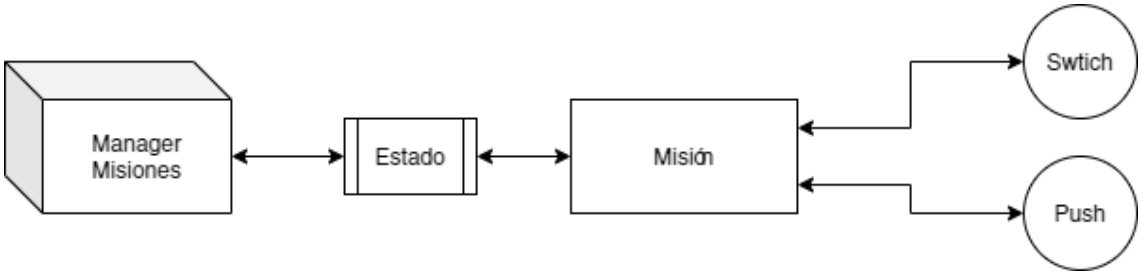

Figura 41: Esquema manager misiones

Cada script específico de misión tiene sus parámetros que debe cumplir para cambiar su condición a **completada**, estos parámetros se ven afectados por sus actuadores. Como observamos en la *Figura 41*, el *switch* y el *push* deben devolver su valor para que el script de misión lo interprete como correcto o incorrecto.

La gran ventaja de haber creado esto es que no es necesario compilar para ver si las misiones están funcionando correctamente. Recordemos que estamos trabajando para unas gafas de VR, por lo que es necesario compilar y probar para corroborar que todo va según lo previsto. En mi caso, una compilación tardaba entre 45 y 60 minutos, un tiempo en el que el PC quedaba inutilizable debido a la carga de la CPU y el disco duro, por eso realizar esto era muy valioso en términos de tiempo.

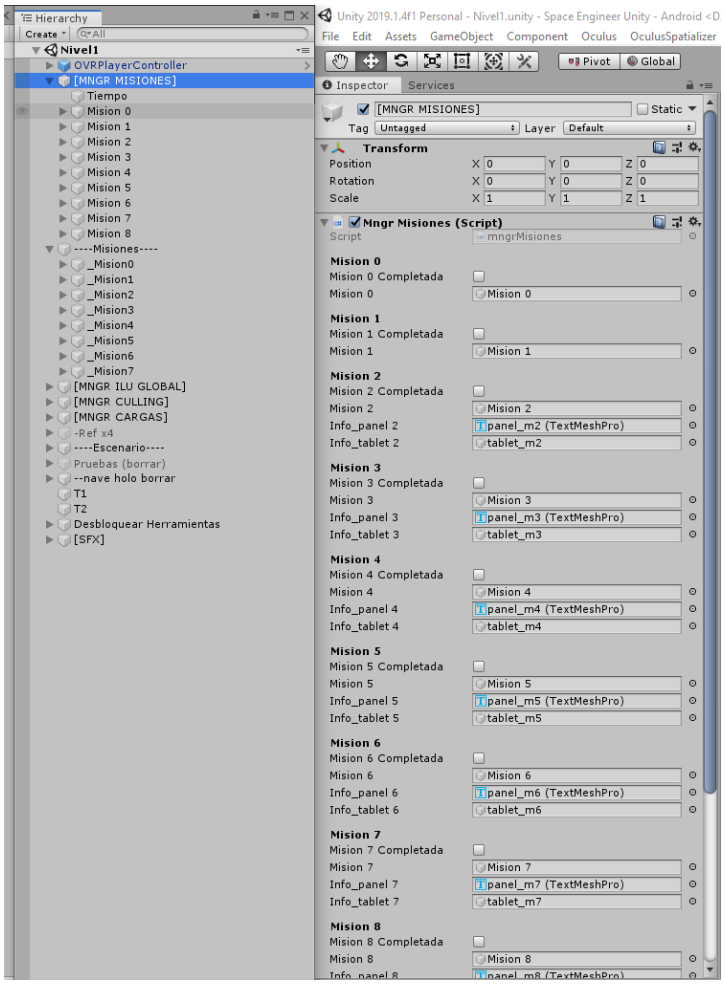

Figura 42: Inspector manager misiones

4.7 Actuadores de misión

Durante la experiencia, encontraremos distintos elementos con los que poder interactuar para realizar una misión. Estos actuadores son:

- **Coger objeto** → recoge el objeto y suma 1 al script correspondiente.
- **Switch** → según su estado, true o false, suma o resta 1 respectivamente. en algunos casos, además, puede activar algún *GameObject.*
- **Push** → botón que devuelve true o false, pudiendo activar además *GameObjects* como el switch.
- **Pos_Filtro** → lugar donde se coloca un filtro. Primero se quita el roto, y luego se coloca el nuevo. Esto suma 1 a su script correspondiente y añade un 1% extra al script Tiempo
- **Pos_Combustible** → lugar donde se coloca la pila de combustible. Primero hay que quitar la pila rota, tirarla al vacío donde detectará que ha sido sacada de la nave con seguridad, y luego colocar la pila nueva.
- **Pila fusión** → es de las mecánicas que más habilidad requieren. Se debe usar las herramientas Destornillador y Atornillador. Primero se deben quitar los tornillos (suma 1 tornillo fuera y resta 1 tornillo puesto en el

estado de la placa), luego la placa y sacar la pila de fusión. Posteriormente se debe hacer el proceso inverso, colocando la nueva pila, luego la placa se imanta para su fácil atornillado, y finalmente se atornillan los tornillos.

- **Placa extraíble (soplete)** → es la placa en la que se puede usar el soplete, simplemente quita los remaches a la placa (resta 1 al estado de la placa) y luego se expulsa.
- **Pos_Baterías** → las baterías se enganchan a su posición, sumando 1 a su script y aumentando en 1% la integridad de la nave al script Tiempo por cada batería colocada.

Lo cierto es que estos scripts han sido diseñados para cada misión, pero si el juego tuviera una proyección de realizar más niveles, habría que hacer scripts genéricos o que fueran más modulares para no tener tantos scripts diferentes. Hay un total de 34 scripts entre los managers individuales de misión y sus actuadores.
Este es un ejemplo del script **mngrMision0**:

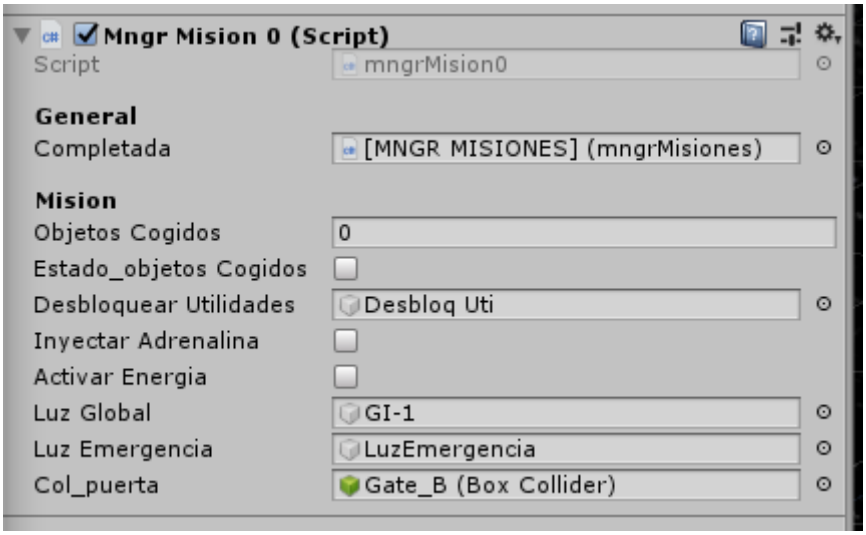

Figura 43: Inspector manager misión 0

Observamos que está conectado a al script mngrMisiones, donde le enviará el estado de la misión cuando se complete.
Objetos cogidos tiene que ser 10 para que Estado_objetosCogidos se ponga true, entonces desbloqueará las utilidades. Ahora podremos inyectarnos la adrenalina, que pasará a true si lo hacemos.
ActivarEnergia se pondrá en true cuando su actuador (switch) se ponga en su estado pertinente, lo que provocará que la luz global cambie y la luz de emergencia desaparezca.
Si todo lo anterior se ha completado, entonces Col_puerta pasará a estar true, y podremos continuar.

4.8 Manager de cargas

Space Engineer cuenta con un sistema de cargas propio. En el caso de este juego siempre estamos en la misma escena, pero los objetos son cargados y descargados según la ubicación en la que estemos. Esto es posible gracias un script que detecta cuando el jugador está en una zona por un *trigger*, cargando las zonas más próximas y descargando las más lejanas. Todo esto es realizado por scripts independientes que operan en sincronía según donde se encuentre el jugador. Por lo tanto, el manager en realidad está compuesto por todas las zonas de carga, donde cada una tiene los scripts que necesita: ActivarZona o DesactivarZona.

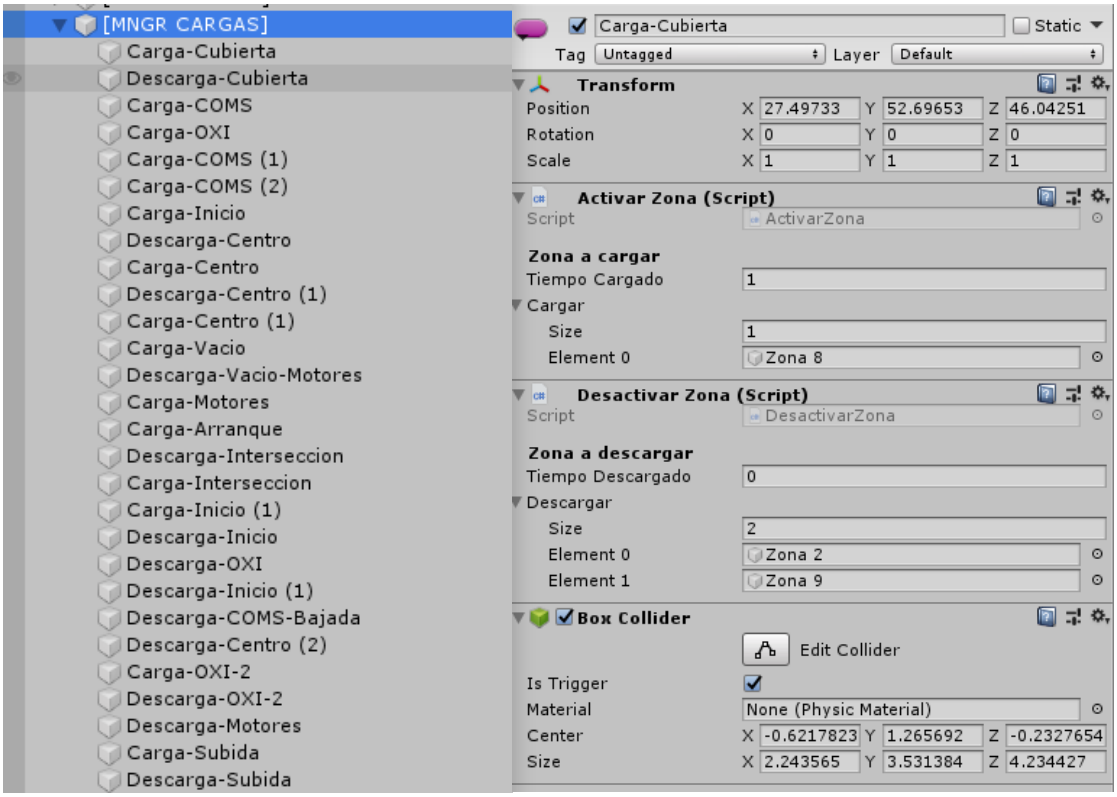

Figura 44: Inspector manager cargas

En los inicios del desarrollo hice especial hincapié en la parte de la carga y la descarga, incluso se llegaron a crear los scripts para la carga de sectores y carga de consumo: ´

Sectores: el juego se divide en partes más grandes que las zonas, que es lo que llamo un sector. Un sector recoge varias zonas que están muy próximas y elementos que son invisibles al jugador (colisionadores, por ejemplo). Además, nos aseguramos que una zona lejana a nuestra posición se descargue por si hubiera ocurrido algún bug de carga.

Consumo: a carga y descarga de consumo se refiere a aquellos objetos que se cargan con una zona cercana pero que están consumiendo recursos innecesarios, como pueden ser las luces o partículas. Con este script se gestiona la carga y descarga de esos elementos si se está en la zona contigua, pero manteniendo otros elementos, como puede ser el modelado.

Estos dos scripts finalmente no han sido necesarios, ya que los propios managers de cada zona de carga se encargan de gestionar la funcionalidad de los dos anteriores, ahorrando tiempo de procesamiento.

4.9 Sistema de *culling*

Además del sistema de cargas, también se ha creado un sistema de *culling* propio. Este mecanismo lo que hace es mostrar en pantalla lo que ve el jugador desde su posición. Es decir, lo que está detrás nuestra no lo vemos. Aunque Unity tiene su propio sistema de Occlusion Culling, sumado al Dynamic Culling que ya tiene por defecto (marcar opción en el objeto o no), este sistema que se ha creado es mucho más agresivo, beneficiando en aquellas zonas que por ejemplo tienen una alta densidad de polígonos, como en la sala de servidores.
Rayos azules miran al frente y cargan. Los rojos dan a la espalda y descargan:

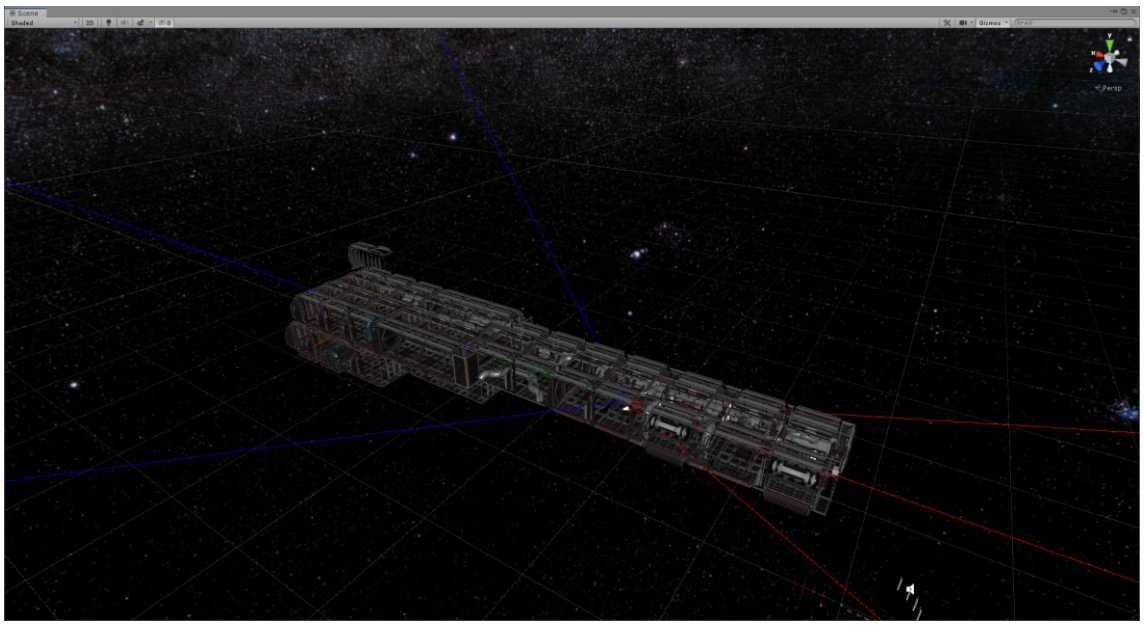

Figura 45: Muestra culling 1

Cuando nos giramos 180 grados respecto a la anterior imagen:

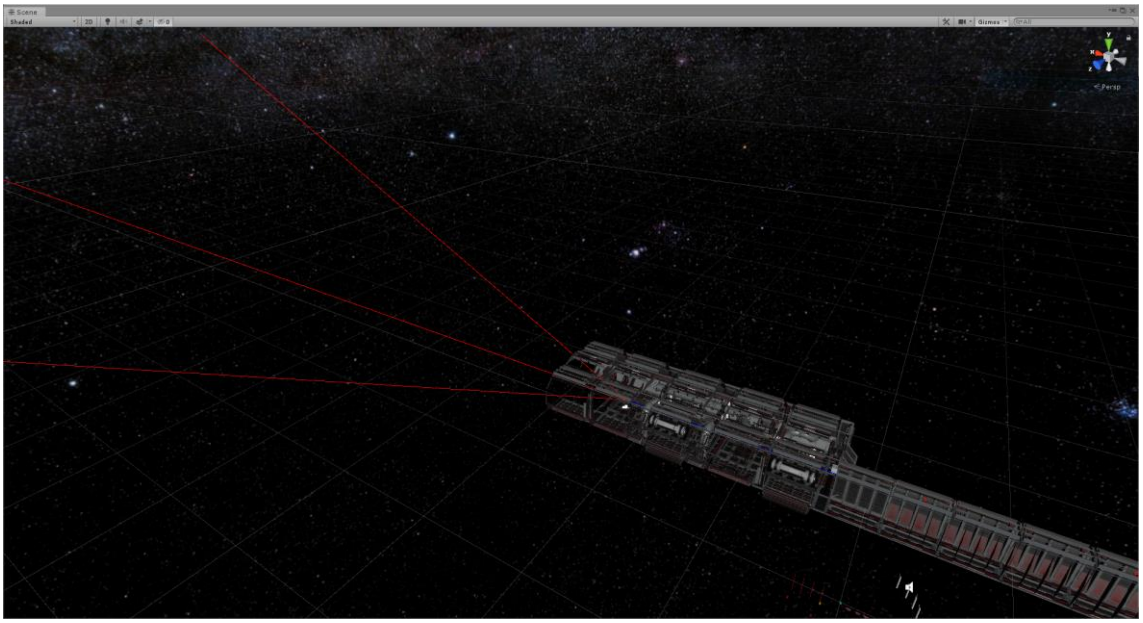

Figura 46: Muestra culling 2

Se puede observar que la parte donde apuntan los rayos rojos ya no está presente.

5. Diseño de niveles

Space Engineer se monta sobre un diseño de niveles modular. Los 5 *assets* adquiridos para el entorno son modulares, lo que permite crear libremente el *environment* uniéndolos, ya que encajan perfectamente. La unión de estos *assets* ha dado lugar a la Igriv-II.

Cabe destacar, aunque ya se ha mencionado en apartados anteriores, que la experiencia iba a funcionar sobre un espacio físico, por lo que la locomoción sería la propia inercia del jugador. Sin embargo, dadas las limitaciones del dispositivo, esto no fue posible y hubo que replantear todo el diseño de nivel del juego. Para recrear el espacio físico tuve que ir marcando con cinta de carrocero el suelo de mi casa, y una vez marcado todo el suelo medí cada tramo de cinta para posteriormente pasar esas medidas al 3DS Max y “levantar las paredes”. Así se modeló el espacio físico:

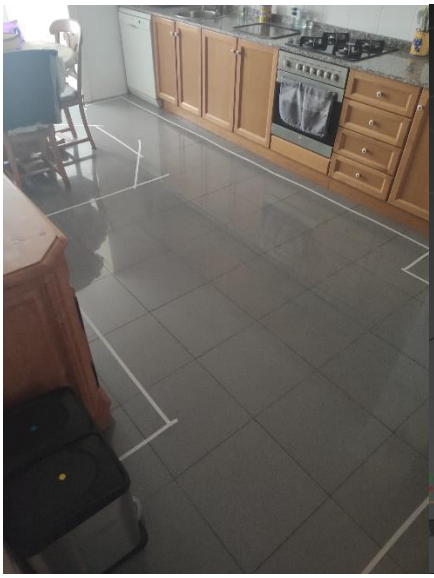

Figura 47: Delimitación espacio físico

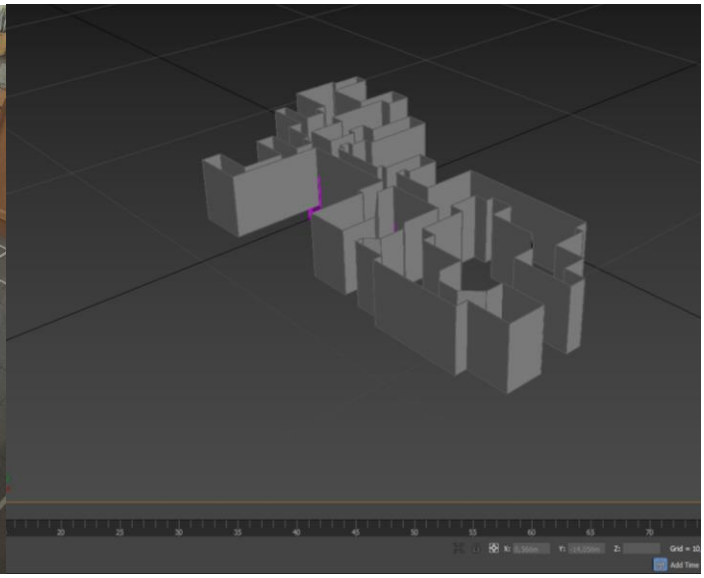

Figura 48: Creación límites físicos en 3D

Y así se pudo ver dentro de Unity en las primeras semanas de desarrollo:

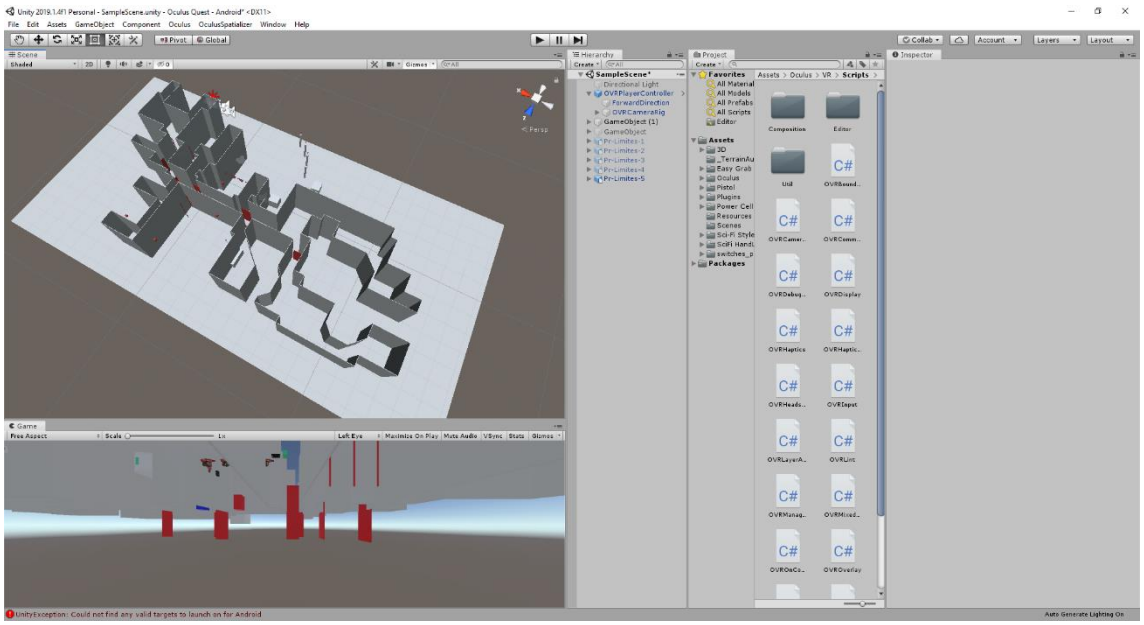

Figura 49: Importación límites físicos en 3D a Unity

Como comentaba, finalmente esta idea tuvo que ser descartada. Así que se tuvieron que buscar otras soluciones teniendo en cuenta el tiempo que ya se había consumido. En un principio, tal y como se ha mostrado, el entorno iba a ser propio, sin embargo, se decidió para no comprometer el desarrollo del proyecto optar por un escenario ya previamente creado y modular. Optar por un escenario modular traía consigo grandes beneficios, ya que la cantidad de materiales iba a ser muy reducida y las estructuras eran copias repetidas, lo que ahorraría recursos de renderizado.

El juego se divide en 10 zonas, cada una de ellas se comunica con sus contiguas mediante puertas, que a nivel interno permite bloquear el paso del jugador y gestionar las cargas.

- **Zona 1**: vestíbulo, es la zona en la que empieza la experiencia. Se realizará la primera misión (misión 0).

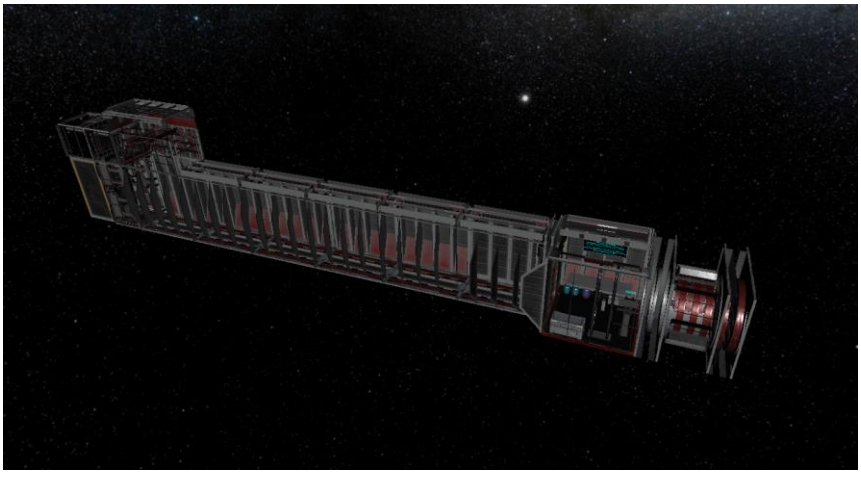

Figura 50: Zona 1

- **Zona 2**: sala central, es la zona que comunica con Oxígeno-Refrigeración, Almacén, Motores-Arranque, Cubierta, COMS-Energía y Vestíbulo. Tendrá lugar la misión 1. Dentro de esta zona está el almacén donde podremos acceder a coger objetos y herramientas para las misiones.

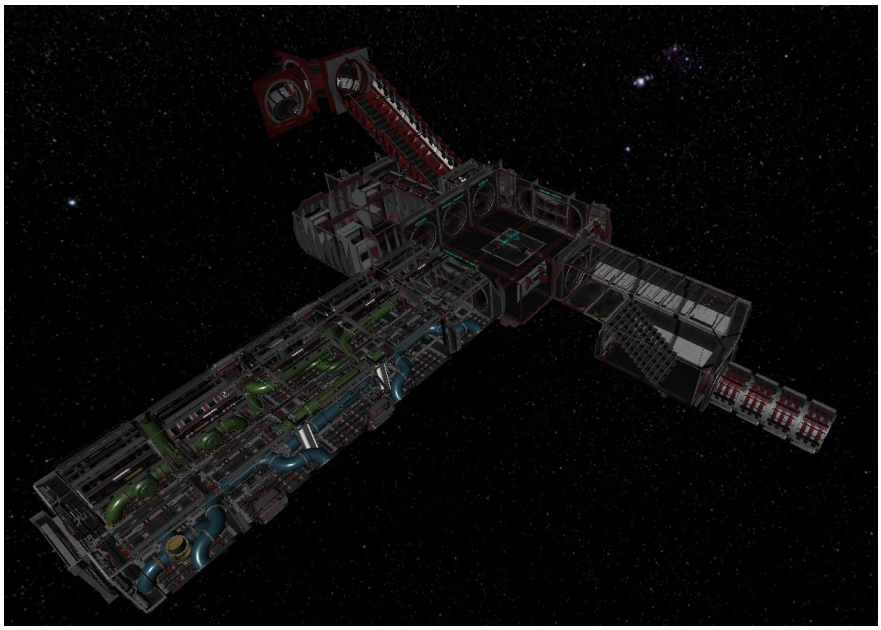

Figura 51: Zona 2

- **Zona 3**: sala de los filtros y de refrigeración, para las misiones 3 y 4. Los filtros estarán flotando por el ambiente.

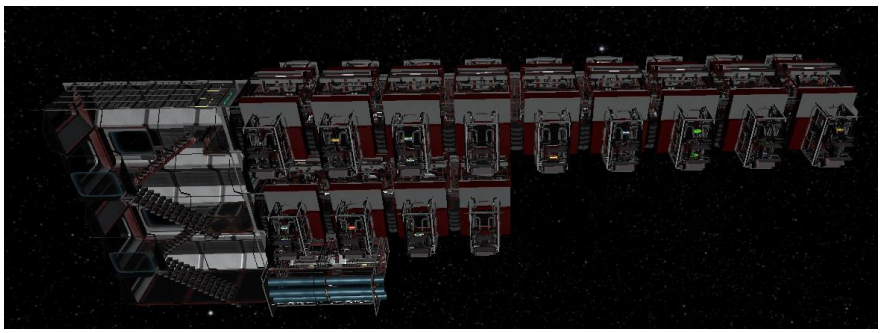

Figura 52: Zona 3

- **Zona 4**: es la zona que hace de puente de carga entre la zona 2 y la zona 5 y 6.

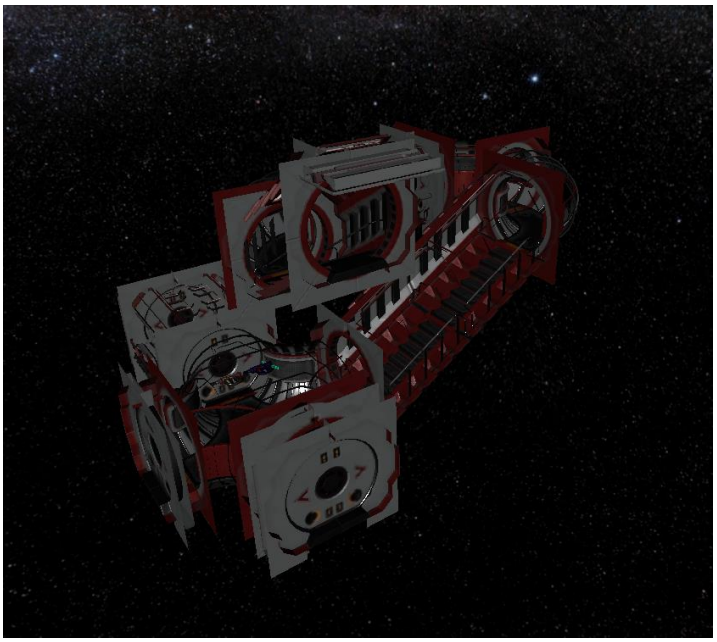

Figura 53: Zona 4

- **Zona 5**: es la zona de vacío, donde se podrán tirar las pilas de combustible rotas al espacio.

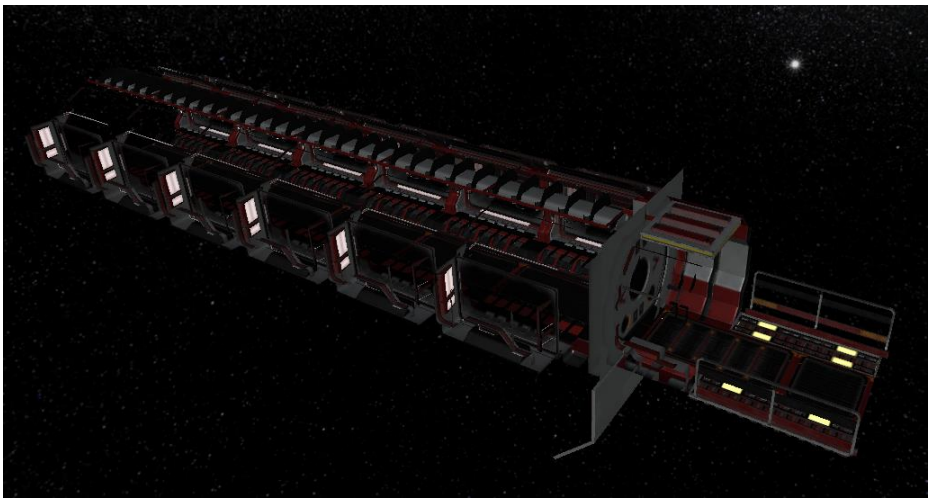

Figura 55: Zona 5

- **Zona 6**: es la zona de motores, dentro encontramos la habitación con las pilas de combustible y el almacén de motores. Por el ambiente habrán flotando *props* para dificultar la extracción y colocación de las pilas de combustible.

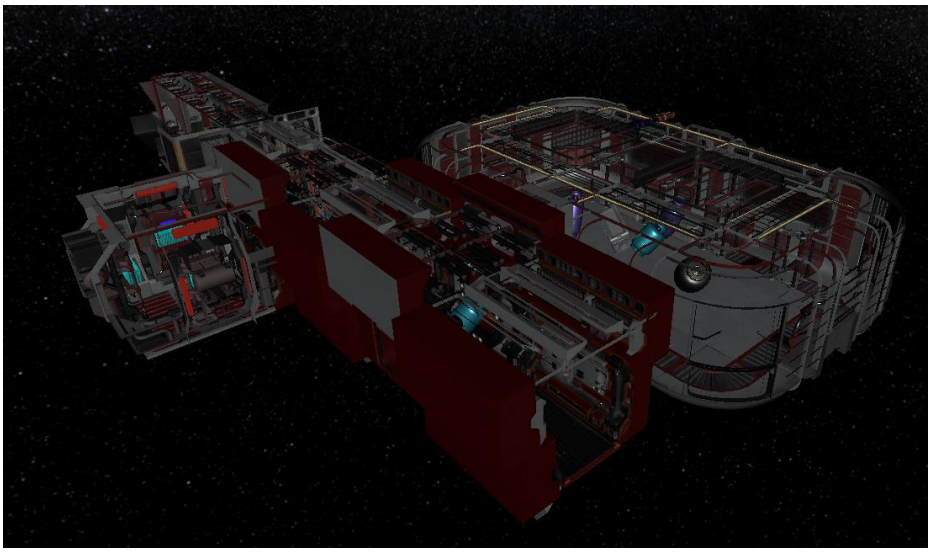

Figura 56: Zona 6

- **Zona 7**: es la zona de arranque, dentro tiene lugar parte de la misión 6.

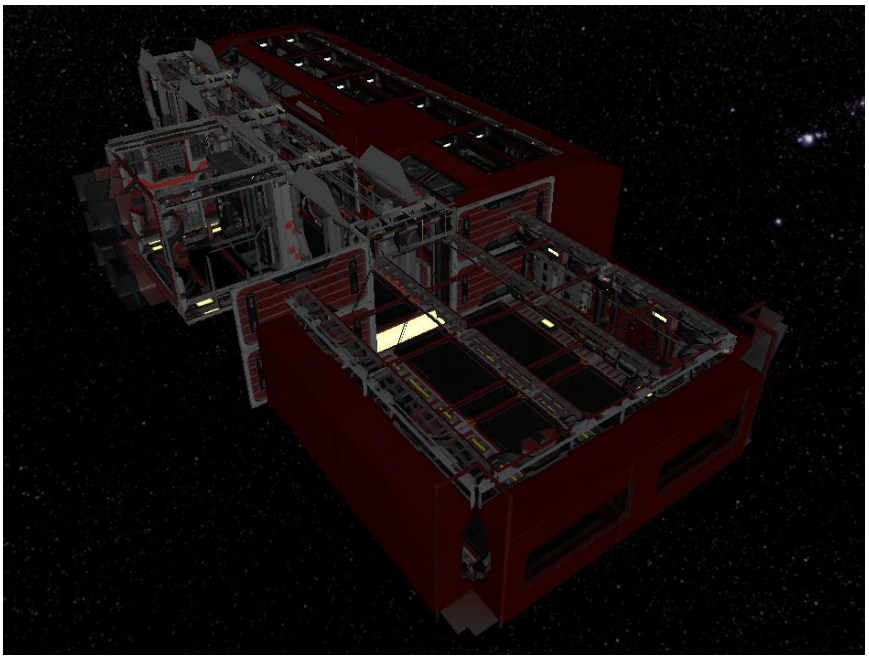

Figura 57: Zona 7

- **Zona 8**: es todo el casco de la nave, por donde podremos caminar libremente.

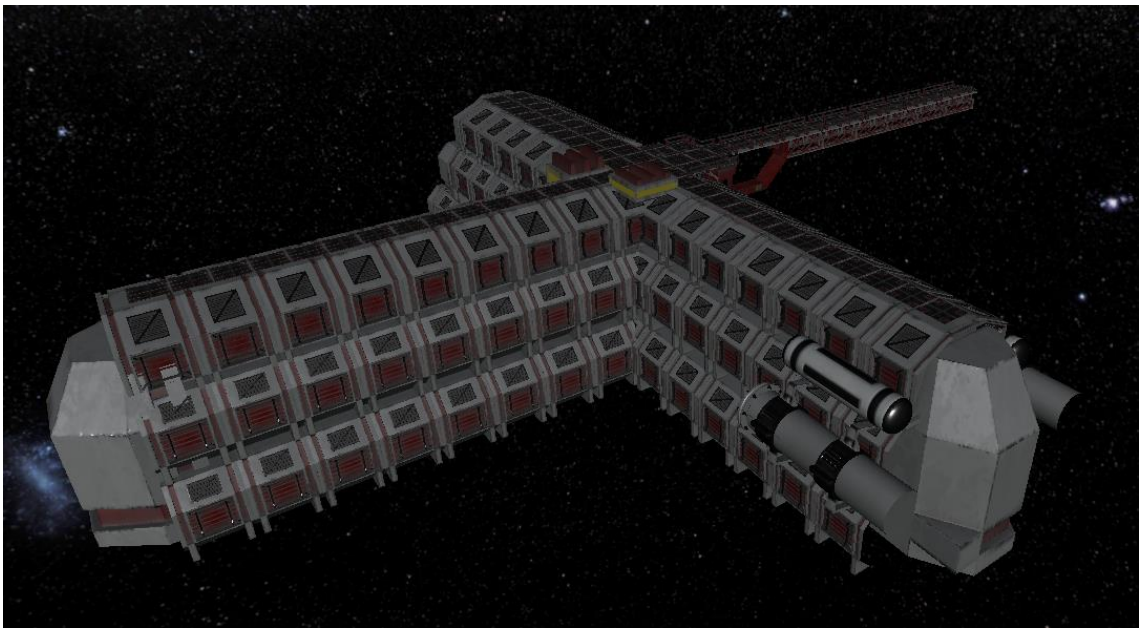

Figura 58: Zona 8

- **Zona 9**: es la sala de servidores, aquí están las baterías y los servidores defectuosos que habrá que arreglar en un ambiente oscuro y tenebroso. La linterna será muy útil.

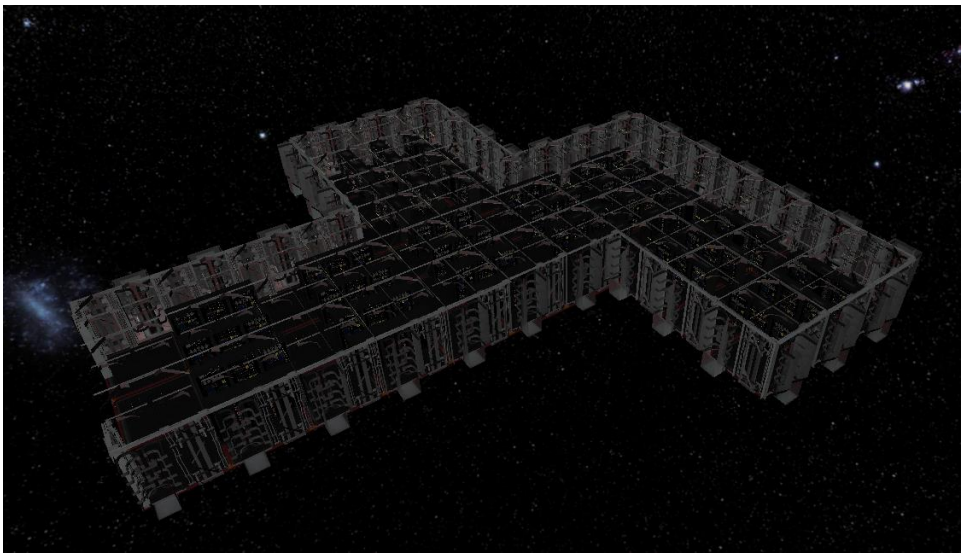

Figura 59: Zona 9

- **Zona 10**: es la zona que hace de puente entre la sala central y la cubierta.

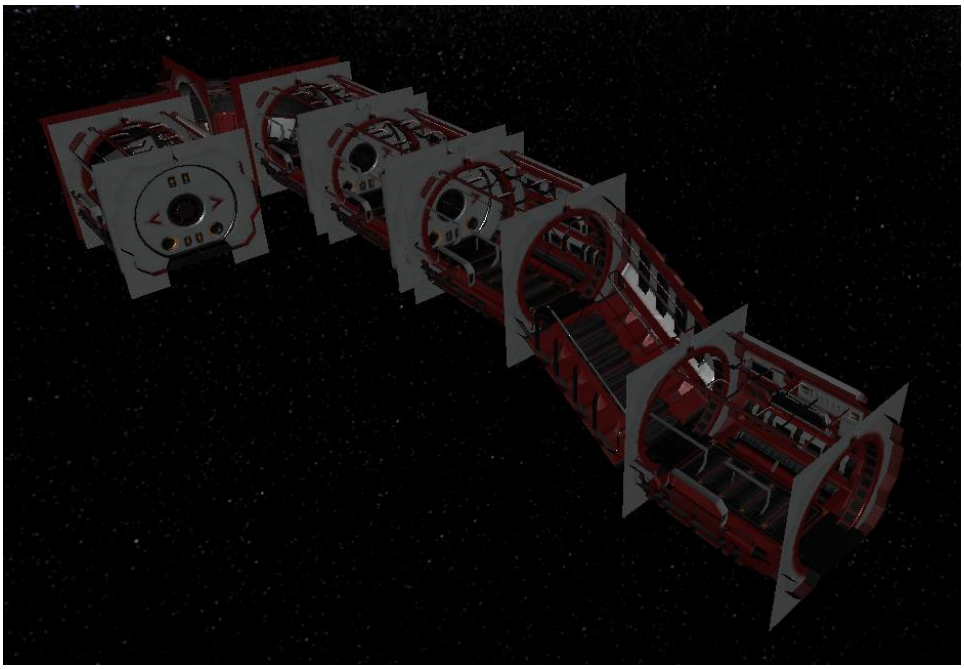

Figura 60: Zona 10

Las puertas tienen un papel importante dentro del juego, ya que separan las zonas de carga y además son los elementos desbloqueables para seguir avanzando, es decir, si una puerta no se abre es porque aún no tenemos que acceder a esa zona. Conforme vayamos avanzando en el juego las puertas se irán desbloqueando y podremos acceder a nuevas salas, algo muy propio de los Escape Room.

Al completar la misión 6, se irá la luz de la nave, obligándonos a usar la linterna. En ese momento se pretende hacer que el jugador este tenso, consiguiendo incluso la sensación de agobio y terror. Durante el trayecto hasta la sala de COMS – Energía, se reproducirán sonidos de crujidos de la nave, además de una válvula de vapor que producirá un escape justo cuando pase el jugador por una zona. Al llegar a la sala de COMS – Energía, tendrá que adentrarse en pasillos oscuros y guiarse por los pitidos. También habrá sonidos extraños cuando se pase por ciertas zonas.

En la misión final, el nivel habrá cambiado en la zona del satélite debido a la explosión, teniendo que pasar por los escombros con mucho cuidado de no caer al vacío.

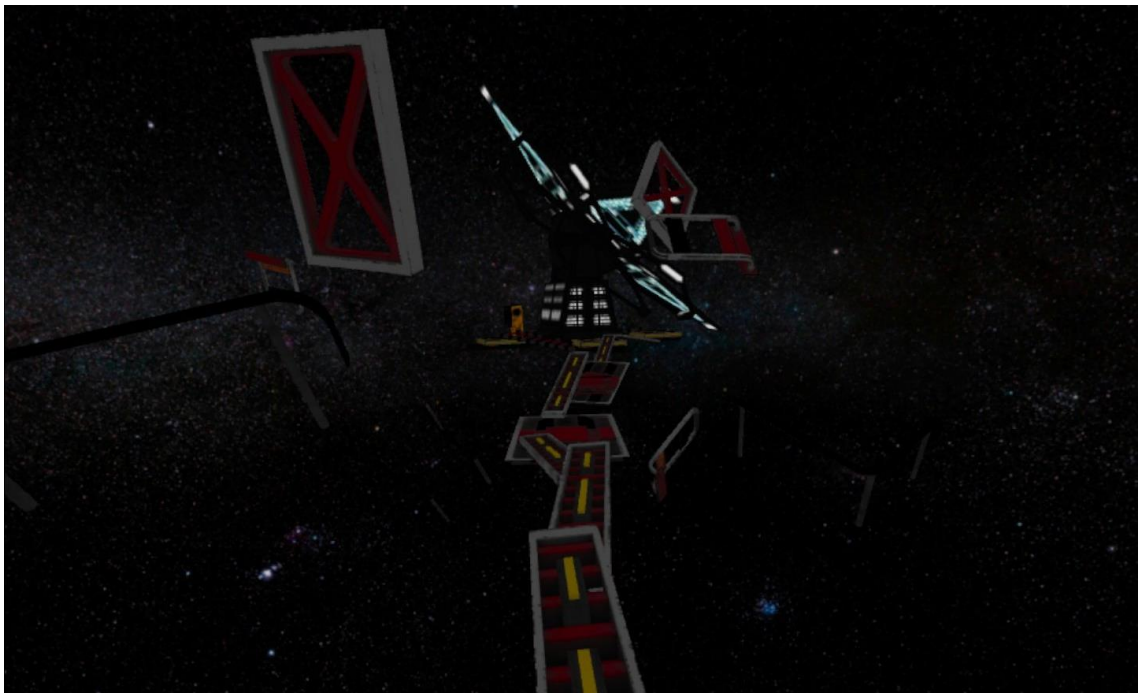

Figura 61: Satélite misión 8

6. Manual de usuario

6.1 Requisitos técnicos del equipo

Para jugar a Space Engineer será **estrictamente** necesario tener unas Oculus Quest del 2019, aunque entendemos que con las Oculus Quest del 2020 también será posible jugar ya que comparten mismo sistema operativo y arquitectura. Sin embargo, habría que probar por si hubiera un problema de compatibilidad con los controladores, aunque los botones siguen siendo los mismos.

6.2 Instrucciones del juego

Para jugar la experiencia, será necesario estar en un espacio de 2x2 metros, despejado de mobiliario y siguiendo siempre las directrices de seguridad que nos marca Oculus. Si bien es cierto que el juego se puede jugar en modo “Limite reducido” del sistema guardián de Oculus Quest, es recomendable que se juegue con el modo “Desplazamiento” para que la rejilla del sistema guardián interrumpa lo menos posible.

Menú del juego

Una vez lanzada la aplicación, empezaremos en el menú del juego, estaremos en un cubículo de 2x2 metros. Tendremos varias opciones a escoger:

1. **Jugar**: lanzará la explicación inicial de la trama, si se vuelve a pulsar podemos saltar dicha explicación y pasaremos directamente a la pantalla de precarga.
2. **Tutorial**: comenzará una explicación sobre la finalidad del juego, consejos y finalmente los controles, esto último acompañado de gráficos.
3. **Modo inmortal**: comenzará el modo de juego, sólo que esta vez la variable del tiempo no se tendrá en cuenta, por lo que nunca se podrá morir.
4. **Salir**: cerrará la aplicación.

En un lateral se encuentran los créditos y el logotipo de la UOC.

Dentro del juego

Una vez hayamos iniciado la partida, estaremos dentro de la nave, no hay GUI en la visión del jugador, tan solo en el brazo derecho el porcentaje de la integridad de la nave.

En la mano izquierda tendremos las utilidades, que son:

- **Jeringuilla**: para inyectarnos la adrenalina en la primera misión

- **Tablet**: para consultar la información de la misión actual
- **Linterna**: para iluminarnos en zonas oscuras
- **Cinta**: para arreglar pequeñas fugas y aumentar la integridad de la nave (futura implementación)
- **Hacha**: para golpear objetos (muy útil para quitar filtros altos)

En la mano derecha tendremos las herramientas, que son:

- **Soplete**: para quitar remaches (sólo para la última misión)
- **Desatornillador**: para quitar tornillos (sólo misión 6)
- **Atornillador**: para colocar tornillos (sólo misión 6)

Instrucciones de controles

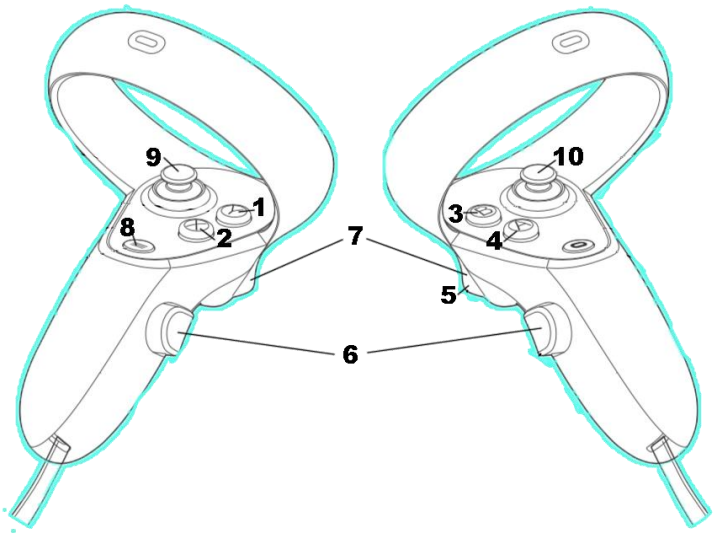

Figura 62: Inputs

1. En la mano izquierda donde tendremos las utilidades habrá que pulsar el botón Y para activar o desactivar las utilidades.
2. Pulsar el botón X para alternar entre las utilidades.
3. En la mano derecha donde tendremos las herramientas habrá que pulsar el botón B para activar o desactivar las herramientas.
4. Pulsa el botón A para alternar entre las herramientas.
5. Para activar la herramienta seleccionada presionar el gatillo derecho.
6. Para agarrar objetos usar los botones laterales.
7. Para accionar interruptores y botones presionar los gatillos. Para girar diales mantener el gatillo presionado y rotar la mano.
8. Pulsar el botón Start del controlador izquierdo para acceder al menú de pausa.
9. Movimiento personaje.
10. Rotación personaje.

Pausa del juego

Pulsando el botón Start accedemos al menú de pausa, pero el tiempo no parará de bajar. De hecho, el menú de pausa lo único que sirve es para volver al menú principal pulsando los dos gatillos al mismo tiempo. Ya se advierte en el tutorial que una vez se empieza no hay vuelta atrás.

Si bien es cierto que como posible mejora del futuro se pueden establecer dificultades, donde en la “Fácil” y “Normal” el tiempo se podría pausar. Simplemente habría que añadir al script de pausa el GameObject Tiempo y que su estado se ponga a *false* mientras su estado sea *true*.

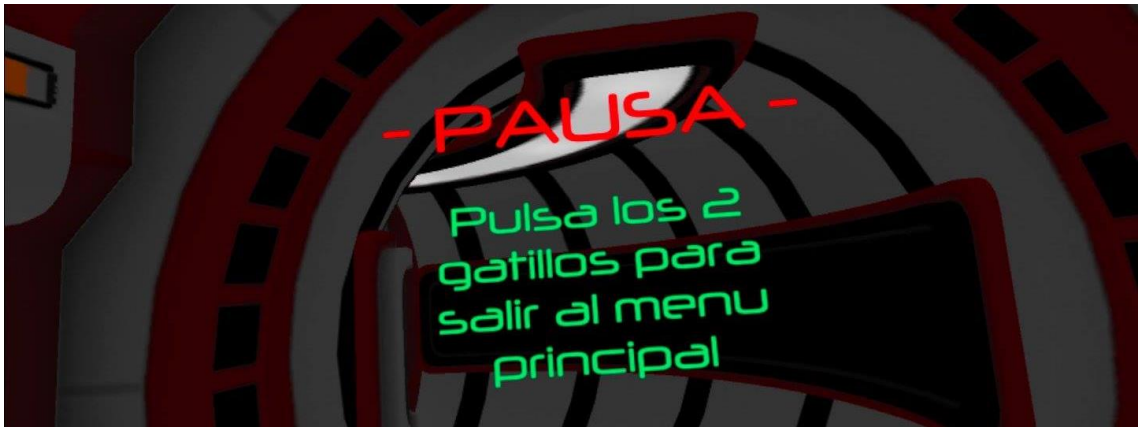

Figura 63: Menú pausa

Menú victoria

Si la última misión se ha completado, el tiempo parará de bajar y 10 segundos más tarde la aplicación cargará el menú de victoria. Este menú es idéntico en funcionalidades al menú principal, sólo que esta vez aparecerá un mensaje distinto en la pantalla y SAI nos dará un mensaje de enhorabuena.

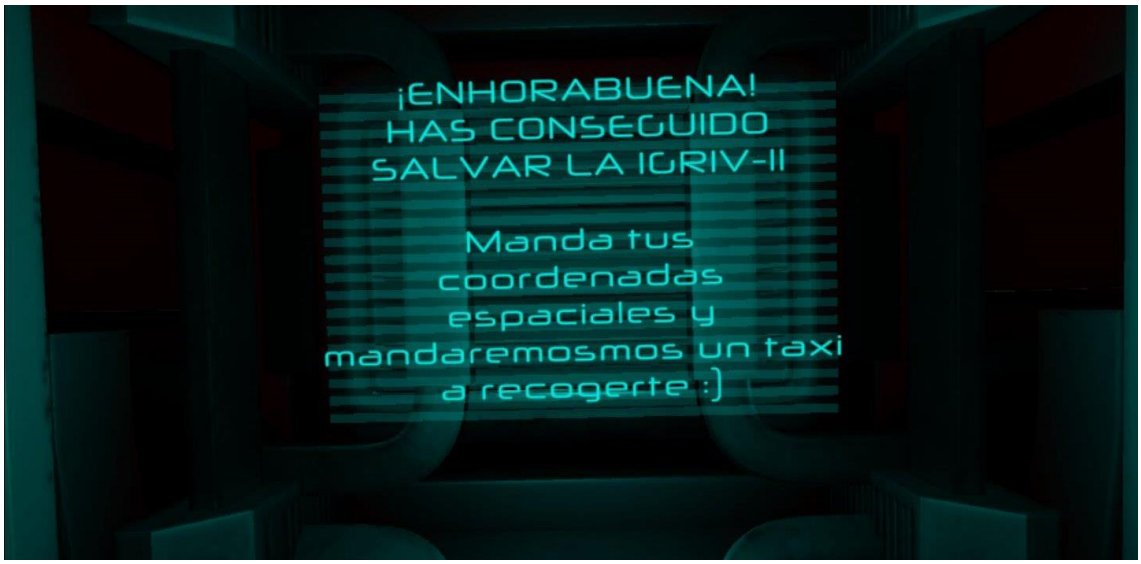

Figura 64: Menú victoria

Menú derrota

Si la integridad de la nave llega a 0%, explotamos una pila de combustible o caemos al vacío, se cargará un menú idéntico al principal, sólo que aparecerá otro mensaje en la pantalla y SAI nos dará ánimos para volverlo a intentar.

Figura 65: Menú derrota

7. Pruebas con usuarios

Dada la situación vivida en el 2020 a causa del COVID-19, las pruebas con usuarios no han sido las esperadas, sin embargo, sí se han podido recabar algunos datos.

El test con usuarios es una parte vital en el proceso de postproducción, ya que es donde se realizarán algunos retoques finales y mejoras sustanciales para la usabilidad del juego. Tratándose de una plataforma exclusiva y no muy común en mi círculo no tuve mucha muestra de pruebas, pero sí sirvieron para ayudar a mejorar ciertos aspectos. Mi Tutor del TF, Rafel Pérez Vidal, se implicó en este apartado para aumentar la cantidad de sujetos de prueba y obtener un *feedback* más rico.

Los sujetos

Los usuarios que han testeado el juego no tienen dispositivos de realidad virtual, por lo que no están muy familiarizados con el entorno. Sin embargo, si han probado la VR en otras ocasiones. Los 5 sujetos tienen entre 24 y 40 años, algunos con perfiles más *gamer*, pero no son jugadores habituales, los otros apenas juegan a videojuegos por lo que la tendencia a la sensación de sentirse perdido podía acentuarse.

Resultados

Después de realizar las pruebas y obtener su feedback estos fueron los problemas observados y que se tuvieron que corregir:

- Audio solapado entre la misión 0 y la misión 1.
- Problemas de usabilidad con las válvulas, no sabían cómo girarlas. Los sujetos que las probaron con diales no tuvieron problemas, por lo que se han quedado los diales.
- El tiempo baja muy rápido.
- Demasiados filtros para completar misión dudan de si hay que ponerlos todos y se agobian.
- Más pistas visuales.
- Sólo a uno el juego le falló una vez realizando una acción. No podía seguir avanzando.
- Textos flotando.
- Caracteres que se salen de sus límites.

Estos resultados ayudaron a obtener una versión mucho más pulida y jugable, siendo más agradable para el jugador. hay que tener en cuenta que estamos

ante un juego de VR, un género que no es el más extendido de todos y que para muchos usuarios no han tenido toma de contacto. La conclusión que puedo sacar al verlos jugar es que se dedican más a observar el entorno y los detalles que a vivir el propio juego, es decir, parece que les ha gustado más entretenerse con el entorno y explorar que la propia finalidad. Es cierto que la VR gana en sensaciones corporales frente a los juegos tradicionales de teclado y ratón o *gamepad*, por eso es algo más novedoso y que permite hacer cosas en un entorno seguro que se acerca más a la forma de interactuar en el mundo físico, un gran ejemplo en este juego es el de coger objetos y tirarlos en gravedad 0. La sensación de nictofobia también ha estado presente, al igual que la de vértigo en la última misión.

8. Conclusiones

8.1 Conclusión personal

Space Engineer ha sido un proyecto donde he tenido que enfrentarme a dificultados tanto técnicas como temporales. El mayor problema que encontré fue el de las limitaciones tecnológicas que tenía el propio dispositivo, teniendo que volver a plantear todo el diseño de niveles y secuencia de misiones del juego. Fue un problema que supe resolver con velocidad y diligencia valorando las distintas posibilidades, teniendo en cuenta el tiempo del cual iba a disponer más el que ya había invertido en probar el *tracking inside-out*. Esto al menos me sirvió para entender por qué no había ningún juego o experiencia como la que tenía planteada crear en un principio.

Me considero una persona que suele centrarse más en el apartado técnico, ya que es la que más me gusta. El tema de crear el sistema de cargas y de culling propio para que el juego tuviera que tener menos en cuenta la optimización a la hora de crear o escoger los modelos, ya que no se dispone mucho tiempo, consideré que era mejor que probar modelos o elegir una estética más plana o *cartoon* (menor poligonaje y texturas planas). Otro asunto que tuve que resolver y que también tenía un impacto importante en el avance del proyecto era que cada compilación tardaba alrededor de unos 45 minutos en completarse, dejando mi ordenador completamente inutilizable (debido al cálculo de compilado). Para ello cree una escena que era de pruebas, totalmente vacía y con las mecánicas principales para ir probándolas. Los cálculos de compilado se redujeron a los 5 minutos aproximadamente. Sin embargo, cuando el proyecto ya estaba avanzado, no había más remedio que jugarlo, por lo que la escena de pruebas ya no servía para este cometido, tenía que comprobar que todo fuera correctamente. Aquí es donde entra el gestor de misiones en el inspector de Unity, que me proporcionaba el estado de las misiones, y haciendo una simulación algo tosca, ya que en realidad sí que podemos movernos con las teclas WASD por el editor, iba comprobando si los eventos iban teniendo lugar.

En términos generales, creo que el proyecto ha cumplido con su cometido de ser una experiencia que supone un reto al usuario que trate de jugarla. Si bien es cierto, hay aspectos mejorables en el juego como la sincronía de ciertos elementos estéticos.

8.2 Conclusión de objetivos y planificación

En cuanto a los objetivos y la planificación, después del cambio que sufrió el proyecto debido a las limitaciones tecnológicas del dispositivo, hubo que hacer un replanteamiento que hizo perder poco más de una semana extra, ya que la idea permanecía intacta.

Creo que también, dada mi situación personal, prolongué alguna fase un poco más de lo establecido, como la de los puzles (Fase 3). Luego ya tuve más tiempo para seguir desarrollando sin problema el resto de fases. También las fases de testeo se vieron afectadas y no las cumplí del todo por el COVID-19, impidiéndome hacer el testing con usuarios durante el proceso de desarrollo. Algunas mecánicas no se vieron implementadas porque el juego parecía que se estaba volviendo demasiado complejo o trabajoso, pero que desde luego quedan para una versión futura y más avanzada que se comentará en el siguiente punto.

Para realizar las compilaciones, siempre se intentaba hacerlo justo antes de una parada, como la comida, cena o si tenía que salir ha realizar algún recado. Así conseguía aprovechar el tiempo que no estaba delante del PC para que compilara. En las compilaciones se probaban muchos cambios, y en ocasiones por algún fallo de alguno no pude probar los siguientes cambios. Cada compilación sé que tenía que andar con pies de plomo, porque no podía echar a perder tanto tiempo en probar. Para más inri, el juego a tiro hecho dura aproximadamente 30 minutos.

8.3 Conclusión sobre futuras implementaciones y mejoras

Space Engineer tiene las limitaciones propias de un TFG, siendo lo mas parecido a una demo o un *teaser*. Creo que desde que se empieza a plantear el proyecto hay que ser realista y saber lo que puedes hacer, lo que no, cuánto tiempo dispones, con qué recursos contamos… Son muchas variables a tener en cuenta que marcan la diferencia entre llegar al último día con un producto demostrable y jugable a no llegar por no ser realista y ambicioso.

Por ello, además de otras razones por sugerencias externas, no se han implementado todas las mecánicas o todos los niveles que realmente compondrían el juego comercial. Estas son algunas de las futuras implementaciones y mejoras:

- Modo fácil, normal, difícil y hardcore
- Alarmas de descompresión donde hay que dirigirse a una exclusa en un tiempo límite
- Recargar el oxígeno cuando se sale al espacio
- Añadir una trama más extensa y sólida durante el juego
- Más salas
- Aprovechar más los espacios
- Misiones secundarias (para la cinta y otras herramientas) que aumenten la integridad de la nave
- Más sustos no-orgánicos y momentos de oscuridad
- Realizar más pruebas de usabilidad con usuarios

9. Glosario

- **APK**: es el nombre de la extensión de archivo generado para el sistema operativo Android y, por consiguiente, la extensión del archivo del proyecto.
- **Asset**: hace referencia a los recursos utilizados por el juego, de la índole que sea (modelos 3D, materiales, partículas, animaciones…).
- **Compilación**: es el proceso de unir todos los recursos que son utilizados en el juego para crear un archivo final o conjunto de ellos que sea el resultado de todo el código fuente creado.
- **Culling**: es un método que consiste en ocultar aquello que no está viendo el jugador, como el Occlusion Culling propio de Unity.
- **GameObject**: es el objeto básico de Unity que se encuentra dentro de la escena de juego.
- **Gamepad**: dispositivo físico para controlar ya se por cable o de forma inalámbrica un programa informático, generalmente videojuegos.
- **Oculus**: empresa de Facebook dedicada a la realidad virtual.
- **OGG**: formato de archivo de audio comprimido, con mayor calidad que el MP3 y menor tamaño. Es el recomendado por Unity.
- **Permadeath**: término que hace referencia a la muerte permanente en la jerga de los videojuegos. Significa que si mueres debes volver a empezar desde el comienzo.
- **Prop**: es un tipo de Asset 3D que hace referencia a objetos que están repartidos por el escenario que no pertenecen a las estructuras (paredes, suelo, techo). Un ejemplo de prop son las cajas.
- **Standalone**: dispositivo que funciona de forma autónoma, como las Oculus Quest. No requiere de ningún otro sistema para funcionar, como las gafas VR de PC.
- **Tracking inside-out**: sistema de posicionamiento calculado desde las propias gafas gracias a sus 4 cámaras. No necesita sensores externos como algunas gafas de PC.
- **VR**: acrónimo que hace referencia a Virtual Reality.

10. Bibliografía

[1] Realovirtual.com. 2021. Cryengine 3.8.6 Cuenta Con Soporte Para Oculus SDK 0.8. [online] Available at: <https://www.realovirtual.com/noticias/2212/cryengine-3-8-6-cuenta-soporte-oculus-sdk-0-8> [Accessed 5 January 2021].

[2] Technologies, U., 2021. Unity - Manual: System Requirements For Unity 2019.1. [online] Docs.unity3d.com. Available at: <https://docs.unity3d.com/2019.1/Documentation/Manual/system-requirements.html> [Accessed 5 January 2021].