UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

Desarrollo de un videojuego para el diagnóstico de fobia social

Trabajo de final de grado
de la especialidad de
Ingeniería del software

Memoria

Alejandro Gómez Ochoa de Retana
Directora: María José Casany

Agradecimientos

En primer lugar, quiero agradecer a mi tutora Maria José Casany, quien con sus conocimientos y experiencia me guió durante cada una de las etapas del proyecto para poder alcanzar los resultados que buscaba, así como a Manel Rello Saltor que me ayudó a resolver varios problemas técnicos durante el desarrollo de este.

También quiero agradecer a la psicóloga psicoanalista Lucía Blanco por dedicarme su tiempo para ayudarme a comprender mejor el campo de la psicología y a elaborar correctamente todo lo que plantea este proyecto.

Por último, me gustaría agradecer a mi pareja Alba Gilabert, graduada en psicología, por su ayuda y apoyo durante toda la realización de este trabajo, especialmente en la decisión sobre el enfoque de este y en la elaboración de los diálogos.

Resumen

Hoy en día utilizamos la informática como una herramienta para implementar soluciones de cualquier tipo para cualquier problema que podamos afrontar. Sin embargo, existe un campo que apenas ha podido beneficiarse de esta área cada vez más imprescindible: la psicología. En este trabajo utilizo mis conocimientos informáticos para crear un prototipo de videojuego que pueda reforzar este ámbito. Sin embargo, la psicología es muy extensa por lo que me centro en el campo de las fobias, concretamente en la fobia social.

El videojuego está destinado a aquellas personas que no estén diagnosticadas de fobia social pero que, sin embargo, consideren que puedan padecerla. Para conseguir esto se han representado en el videojuego una serie de escenas para observar cómo el jugador reacciona ante ellas y poder realizar un diagnóstico final.

Con el objetivo de desarrollar un producto fiable, he colaborado con la psicóloga psicoanalista Lucía Blanco, la cual me ha ayudado a definir las escenas a plasmar en el videojuego y a establecer un sistema para valorar las reacciones de los jugadores ante estas.

Este proyecto se ha llevado a cabo principalmente mediante el motor de videojuegos Unity. También se ha hecho uso de servicios externos con otros fines como la obtención de objetos 3D, el modelado de estos, animaciones, etcétera.

El objetivo final no es otro que el de aportar una herramienta para aquellas personas que no reciben ayuda profesional, debido a que probablemente no saben que la necesitan. Con la ayuda de este proyecto, estas personas podrán recibir un diagnóstico que les sirva como guía y acudir a profesionales si lo necesitan.

Resum

Avui dia utilitzem la informàtica com una eina per implementar solucions de qualsevol tipus per a qualsevol problema que puguem fer front. No obstant això, hi ha un camp que amb prou feines ha pogut beneficiar-se d'aquesta àrea cada cop més imprescindible: la psicologia. En aquest treball faré servir els meus coneixements informàtics per crear un prototip de videojoc que pugui reforçar aquest àmbit. Tanmateix, la psicologia és molt extensa, per això em centraré en el camp de les fòbies, concretament en la fòbia social.

El videojoc estarà destinat a aquelles persones que no estiguin diagnosticades de fòbia social però que, tanmateix, considerin que puguin patir-la. Per aconseguir això s'han representat al videojoc una sèrie d'escenes per observar com el jugador reacciona davant d'elles i poder fer un diagnòstic final.

Amb l'objectiu de desenvolupar un producte fiable, he col·laborat amb la psicòloga psicoanalista Lucía Blanco, la qual m'ha ajudat a definir les escenes a plasmar al videojoc i a establir un sistema per valorar les reaccions dels jugadors davant d'aquestes.

Aquest projecte s'ha fet principalment mitjançant el motor de videojocs Unity. També s'ha fet ús de serveis externs amb altres finalitats com l'obtenció d'objectes 3D, modelar-los, animacions, etcètera.

L'objectiu final no és altre que aportar una eina per a aquelles persones que no reben ajuda professional, ja que probablement no saben que la necessiten. Amb l'ajuda d'aquest projecte, aquestes persones podran rebre un diagnòstic que els serveixi com a guia i acudir a professionals si ho necessiten.

Abstract

Today we use computing as a tool to implement solutions of any kind for any problem that we may face. However, there is one field that has hardly been able to benefit from this increasingly essential area: psychology. In this work, I will use my computer skills to create a video game prototype that can reinforce this area. However, psychology is very extensive so I will focus on the field of phobias, specifically on social phobia.

The video game will be intended for those people who are not diagnosed with social phobia but who, nevertheless, consider that they may suffer from it. To achieve this, a series of scenes have been represented in the video game to observe how the player reacts to them and to be able to make a final diagnosis.

In order to develop a reliable product, I have collaborated with the psychoanalyst psychologist Lucía Blanco, who has helped me define the scenes to capture in the video game and establish a system to assess the reactions of the players to them.

This project has been carried out mainly using the Unity video game engine. External services have also been used for other purposes such as obtaining 3D objects, modeling them, animations, and so on.

The ultimate goal is none other than to provide a tool for those who do not receive professional help because they probably do not know they need it. With the help of this project, these people will be able to receive a diagnosis that will serve as a guide and turn to professionals if they need it.

Índice

1. Introducción **7**

2. Contextualización **8**

3. Objetivo **9**

4. Stakeholders **10**

5. Metodología **11**
5.1. Herramientas de seguimiento 13
5.2. Herramientas de desarrollo 15

6. Alternativas existentes a Unity **17**

7. Planificación temporal **19**
7.1. Tareas 19
7.2. Estimación inicial y Gantt 20

8. Gestión de riesgos: Planes alternativos y obstáculos **24**

9. Presupuesto **25**
9.1. Elementos 25
9.1.1. Recursos humanos 25
9.1.2. Costes genéricos 26
9.1.3. Contingencias 27
9.1.4. Imprevistos 27
9.2. Control de Gestión 28
9.3. Planificación final: Desviaciones 28

10. Informe de sostenibilidad **31**
10.1. Autoevaluación 31
10.2. Análisis del proyecto 31
10.2.1. Dimensión Ambiental 31
10.2.2. Dimensión Económica 32
10.2.3. Dimensión Social 32

11. Requisitos **34**
11.1. Obtención de requisitos 34
11.2. Requisitos funcionales 34
11.3. Requisitos no funcionales 35

12. Diseño del sistema **36**
12.1. Patrones 36
12.2. Diseño de Personajes 36
12.2.1. Protagonista 37
12.2.2. NPC 37
12.3. Diseño de Escenas 38
12.1.1. BeginScene 38

12.1.2. ClassScene 38
12.1.3. BarScene 40
12.1.4. QuestionnaireScene 41
12.1.5. ResultsScene 42

13. Implementación 46
13.1. Unity 46
13.2. Servicios Utilizados 49
13.1. Elementos utilizados 52
13.1.1. Canvas 52
13.1.2. Animator Controller 54
13.1.3. WayPoints 56
13.1.4. Cinemachine 58
13.1.5. Coroutines 60
13.2. Componentes de los Personajes 62
13.2.1. Protagonista 62
13.2.2. NPC 63
13.3. Producción 63

14. Pruebas 65

15. Conclusiones 66

16. Referencias 68

17. Anexo A 70

1. Introducción

Conforme pasan los años, podemos apreciar cómo la informática (y más abiertamente la tecnología) influye cada vez más en prácticamente todos los campos: en las empresas, el uso de herramientas tecnológicas es una necesidad estratégica en la que gran parte de organizaciones a nivel global están invirtiendo, optimizando los procesos, proporcionando una mayor productividad, mejor comunicación, etc (Oliva, 2019). En nuestra vida cotidiana, podemos utilizarla para organizarnos mejor, aprender cosas nuevas, llevar un registro de nuestras metas y demás (*La Influencia De La Tecnología En Nuestra Vida Cotidiana*, n.d.). Son escasos los escenarios en los que la informática no ha tenido una influencia relevante, pero entre estos podemos encontrar uno muy interesante que es el que trataremos en este trabajo: la psicología.

Sin embargo, la psicología es una disciplina extremadamente extensa, por lo que me centraré en un aspecto más concreto: las fobias. Estas son un tipo de trastorno de ansiedad, un temor fuerte e irracional de algo que representa poco o ningún peligro real (*Fobias*, 2021). Pero ¿puede la informática de alguna manera ayudarnos a diagnosticar ciertos tipos de fobias?

2. Contextualización

Actualmente solo existe una única solución posible para superar una fobia: el tratamiento psicológico. Aquí se utilizan una serie de herramientas y mecanismos que ayudan a resolver el problema sin la necesidad de ayuda farmacológica.

Por otro lado, numerosos estudios han demostrado que los videojuegos pueden ser mucho más que mero entretenimiento: pueden ser una potente herramienta para favorecer el desarrollo cognitivo del cerebro, además de ser una altamente efectiva herramienta de aprendizaje (UPM, 2015).

He aquí mi intención para este trabajo, intentar aplicar algunos aspectos de los videojuegos al diagnóstico de ciertas fobias, creando uno que pueda ayudar a aquellas personas que quizás padezcan de este trastorno.

Para ello, en este trabajo me enfocaré en una única fobia: la fobia social. Esta es, definida brevemente, un problema de salud mental que provoca un temor intenso y persistente de ser observado y juzgado por otros.

Si bien sí existen ciertas soluciones tecnológicas para ayudar al tratamiento de ciertas fobias, estas normalmente se basan en una proyección en realidad virtual que sitúa al paciente en aquella situación que teme, realizando así una terapia de exposición. Ahora bien, ese no es el objetivo de mi trabajo.

La intención de este trabajo no trata de ser una ayuda terapéutica de aquellas personas que padecen de dicha fobia, sino de ser una ayuda diagnóstica para aquellas personas que puedan padecerla. Es decir, ¿cómo distinguiría una persona tímida si sufre de fobia social o si, sin embargo, sufre únicamente de cierto grado de timidez? ¿Es simplemente una forma de ser o más bien un trastorno que precisa de ayuda psicológica? Ayudar al usuario a responder a estas preguntas es el objetivo principal de este trabajo.

Figura 1: Videojuegos y educación

3. Objetivo

Mi objetivo es elaborar un videojuego cuya funcionalidad sea aportar una ayuda diagnóstica para aquellas personas que crean que pueden padecer de fobia social, un miedo persistente e irracional ante situaciones que puedan involucrar el escrutinio y juzgamiento por parte de los demás, como en fiestas u otros eventos sociales.

Para lograr este objetivo plasmaremos ciertas situaciones que se consideran incómodas para los padecedores de dicha fobia en el videojuego. Para ello, colaboraré con un psicólogo experto en la materia que me ayudará a identificar estas situaciones. Una vez hecho esto, elaboraremos un sistema con el que valorar la toma de decisiones que el paciente habrá hecho para poder aportar un resultado final, decidiendo si el paciente sí que podría realmente padecer de dicha fobia o si, por el contrario, se trataría de un caso distinto.

Para poder desarrollar este proyecto haré uso de la herramienta Unity, la mejor plataforma de desarrollo de videojuegos (Asensio, 2019).

4. Stakeholders

En cuanto a las partes interesadas del proyecto, listaré a continuación aquellas personas a las que les interesaría el videojuego, las afectadas o los participantes en el desarrollo.

- *Desarrolladores*: encargados de llevar a cabo la codificación del videojuego.
- *Jugadores*: personas que padecen de fobia social o crean que pueden padecerla.
- *Psicólogo psicoanalista*: encargado de colaborar con las situaciones a recrear en el videojuego.
- *Administración*: encargado de la gestión de riesgos.

5. Metodología

La metodología de trabajo es una herramienta muy potente para definir las pautas y procedimientos de una empresa o proyecto. Está comprobado que ayuda a optimizar los recursos de la empresa, mejora la calidad del trabajo, reduce los riesgos de los proyectos, establece prioridades, etc. Existen diversos tipos de metodologías, sin ser una mejor que otra, estando cada una pensada para distintos tipos de problemas y entornos de trabajo.

Para este proyecto me he decantado por hacer uso de la metodología *Agile*. Esta permite adaptar la forma del trabajo a las condiciones del proyecto, proporcionando así flexibilidad e inmediatez en la comunicación con el cliente para amoldar el proyecto y su desarrollo a las circunstancias específicas del entorno (Pursell & Rohn, 2021).

Por *'agile'* entendemos métodos de trabajo donde los requisitos y las soluciones evolucionan con el tiempo (no se quedan estancados en la planificación inicial) según la necesidad de cada proyecto, los trabajadores se organizan de manera multidisciplinar y autoorganizada en equipos eficientes y flexibles para planear el flujo de trabajo, existe un proceso compartido de toma de decisiones y los proyectos se gestionan de forma flexible, autónoma, eficaz, reduciendo los costes e incrementando la productividad.

Figura 2: Agile

Desde sus inicios, la metodología Agile reivindica 4 valores:

- Las interacciones de las personas sobre los procesos y las herramientas.
- Un software en funcionamiento frente a documentación exhaustiva.
- La participación activa del cliente durante todo el proceso de desarrollo.
- La capacidad de respuesta ante los cambios e imprevistos.

Además, consta de 12 principios fundamentales. Estos son:

1. Lograr la satisfacción del cliente a través de la entrega continua de software.
2. No tener miedo a realizar cambios.
3. Entregar software funcional en una escala de tiempo menor.
4. Desarrolladores y gerencia deben trabajar juntos.
5. Desarrollar proyectos en torno a personas motivadas.
6. Interactuar cara a cara es el modo de comunicación más eficiente y efectivo
7. Un software que funciona es la medida principal del progreso.
8. Los procesos ágiles promueven el desarrollo sostenible.

9. La atención continua a la excelencia técnica y al buen diseño mejora la agilidad.
10. La simplicidad es esencial.
11. Las mejores arquitecturas, requisitos y diseños emergen de equipos autoorganizados.
12. Inspeccionar y adaptar.

Esta metodología fue creada originalmente para los procesos de desarrollo de software, pero puede ser aplicable también a una gran empresa, un negocio pequeño o si trabajas por tu cuenta. *Agile* es apto para cualquier organización que esté dispuesta a cambiar paradigmas y lograr una cadena de valor, desde el trabajo interno de sus equipos hasta el usuario y cliente final.

Su objetivo es desarrollar la capacidad de crear nuevos flujos de trabajo que te permitan optimizar y mejorar el rendimiento del personal que labora en tu organización.

En metodologías ágiles la adaptación a los cambios e imprevistos es fundamental. Esta es la clave para conseguir el éxito en proyectos complejos, donde los requisitos son cambiantes o poco definidos y en donde la adaptación, la innovación, la complejidad y flexibilidad son fundamentales.

Entramos ahora en la parte más importante de esta metodología, el *Sprint*.

Sprint es el nombre que va a recibir cada uno de los ciclos o iteraciones que vamos a tener dentro de un proyecto Agile.

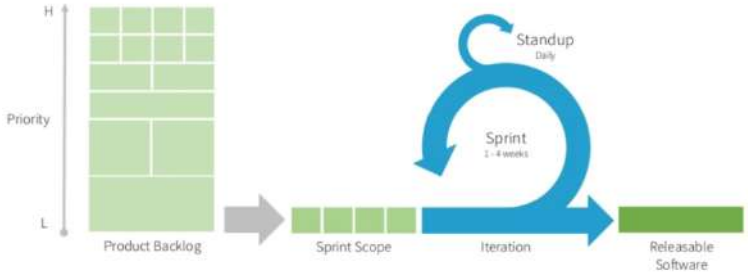


Figura 3: Sprint

Nos van a permitir tener un ritmo de trabajo con un tiempo prefijado, siendo la duración habitual de un Sprint unas cuatro semanas, aunque lo que la metodología dice es que debería estar entre dos semanas y un máximo de dos meses.

En cada Sprint o cada ciclo de trabajo lo que vamos a conseguir es lo que se denomina un entregable o incremento del producto, que aporte valor al cliente.

En la imagen vemos una pila o *product backlog*, que serían todos los requisitos que nos pide el cliente, es decir, el año completo de trabajo. La idea es ir seleccionando esos requisitos en los que tenemos la pila dividida, y los vamos a ir haciendo en diferentes Sprints.

Esto se haría seleccionando esos requisitos y realizando todos los pasos que conforman un Sprint, es decir, la toma de requisitos, diseño, implementación, pruebas y despliegue en el plazo establecido, y así vamos a tener siempre un software que sea válido y funcionando

5.1. Herramientas de seguimiento

Para realizar un correcto seguimiento del proyecto he hecho uso de conocidos servicios cuya funcionalidad no es otra que asegurar una correcta administración de este. Los principales programas que he utilizado son:

Figura 4: Trello Logo

Es una aplicación multidispositivo y multiplataforma que permite gestionar proyectos de forma sencilla y muy visual. Con Trello puedes crear tableros y listas para organizar cualquier proyecto en el que estemos trabajando, utilizarlos individualmente o invitar a compañeros a que colaboren, personalizar flujos de trabajo para distintos proyectos, añadir listas de tareas pendientes en tarjetas y muchas opciones más.

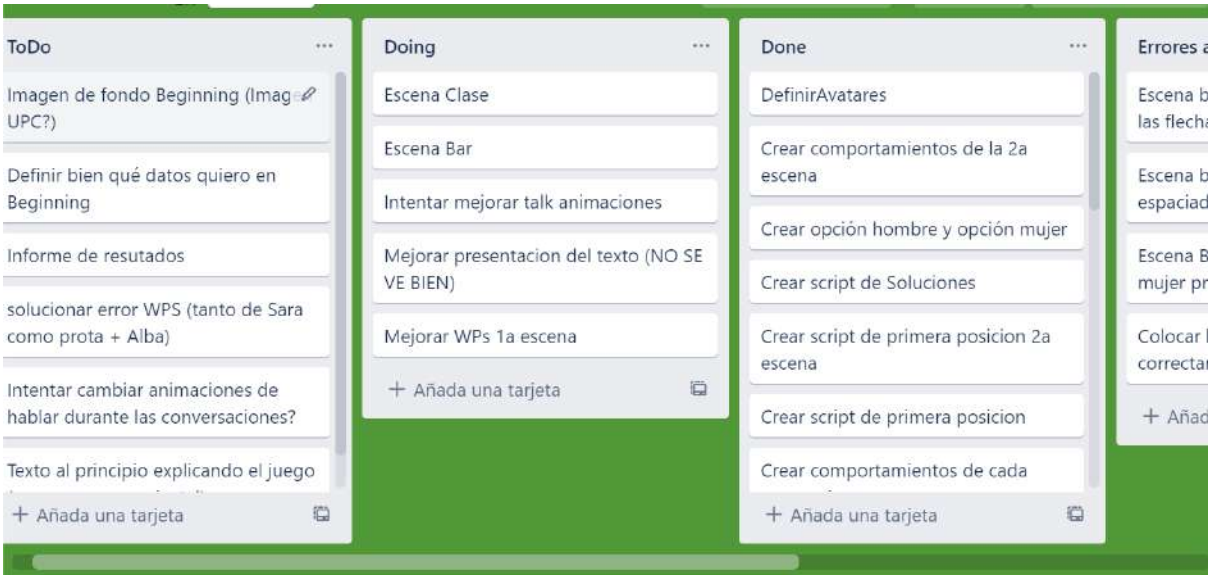


Figura 5: Trello

Figura 6: Github Logo

Es un portal creado para alojar el código de las aplicaciones de cualquier desarrollador. Está creado para que los desarrolladores suban el código de sus aplicaciones y herramientas. La web utiliza un sistema de control de versiones con el que los desarrolladores pueden administrar su proyecto. Las principales características de la plataforma es que ofrece las mejores características de este tipo de servicios sin perder la simplicidad, y es una de las más utilizadas del mundo por los desarrolladores. Es multiplataforma, y tiene multitud de interfaces de usuario.

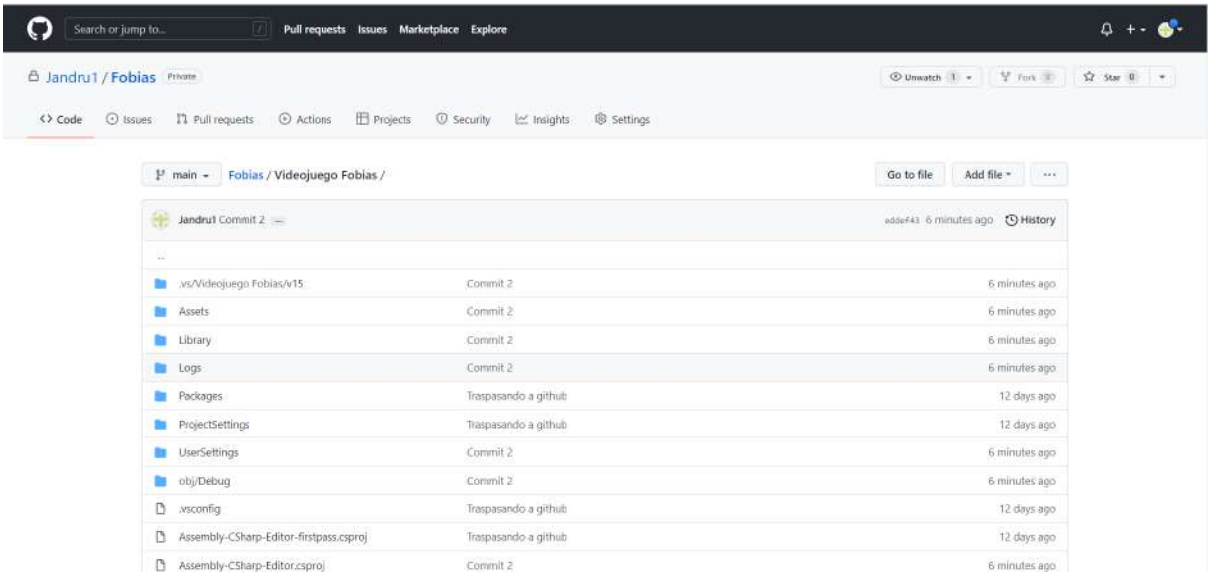


Figura 7: Github

Figura 8: Github Desktop Logo

GitHub Desktop es una aplicación que te habilita para interactuar con GitHub utilizando una GUI (Interfaz Gráfica de Usuario) en vez de la línea de comandos o de un buscador web. Fomenta que nosotros y nuestro equipo colaboremos utilizando las mejoras prácticas con Git y GitHub. Podemos utilizar GitHub Desktop para completar la mayoría de los comandos de Git desde nuestro ordenador con confirmaciones visuales para los cambios. Podemos subir, extraer y clonar repositorios remotos con GitHub Desktop y utilizar herramientas colaborativas tales como atribuir confirmaciones y crear solicitudes de extracción.

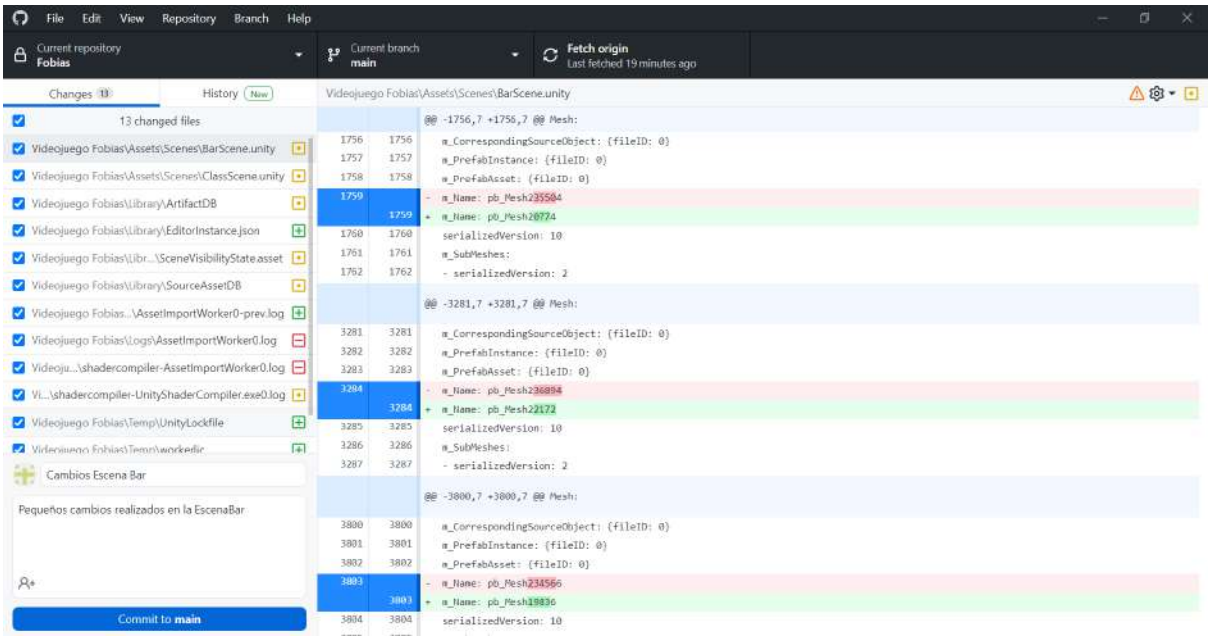


Figura 9: Github Desktop

5.2. Herramientas de desarrollo

Figura 10: Unity Logo

Unity es un software que centraliza todo lo necesario para el desarrollo de videojuegos. Es una herramienta que te permite crear videojuegos para diversas plataformas (PC, videoconsolas, móviles, etc.) mediante un editor visual y programación vía scripting, y pudiendo conseguir resultados totalmente profesionales. Prueba de ello son juegos muy famosos creados con Unity; tales como “Monument Valley”, “Gris” o “Cuphead”. Además, es muy utilizado en la mayoría de los desarrollos de videojuegos para móvil.

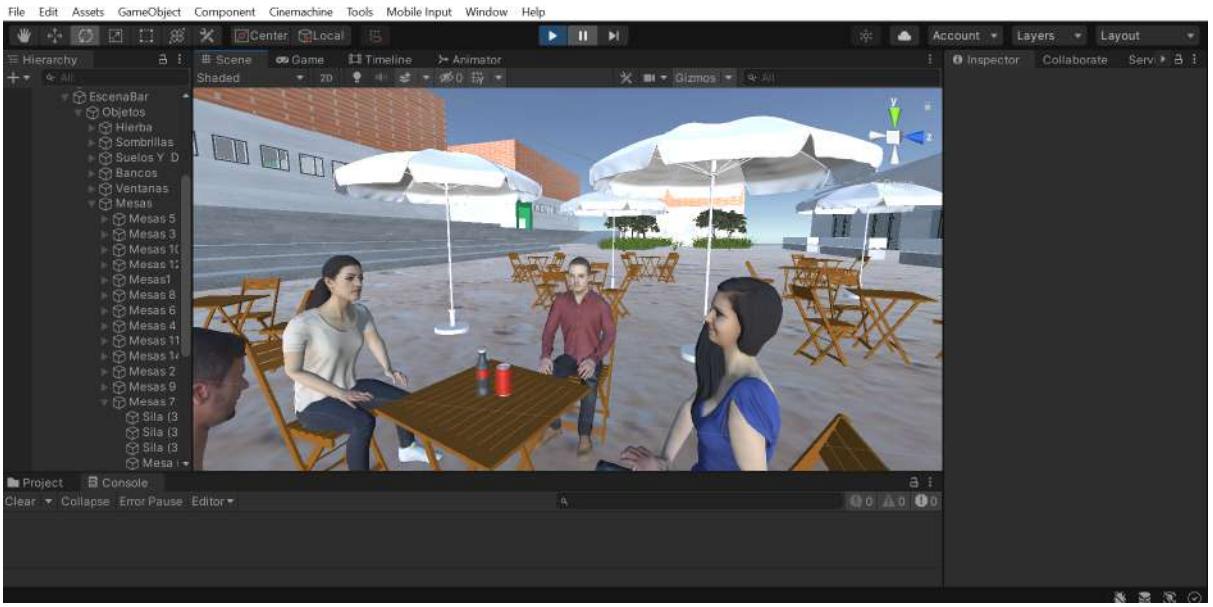


Figura 11: Unity

Figura 12: Visual Studio Logo

Visual Studio es un conjunto de herramientas y otras tecnologías de desarrollo de software basado en componentes para crear aplicaciones eficaces y de alto rendimiento, permitiendo a los desarrolladores crear sitios y aplicaciones web, así como otros servicios web en cualquier entorno que soporte la plataforma. Es compatible con múltiples lenguajes de programación, tales como C#, C++, Visual Basic .NET, Java y más.

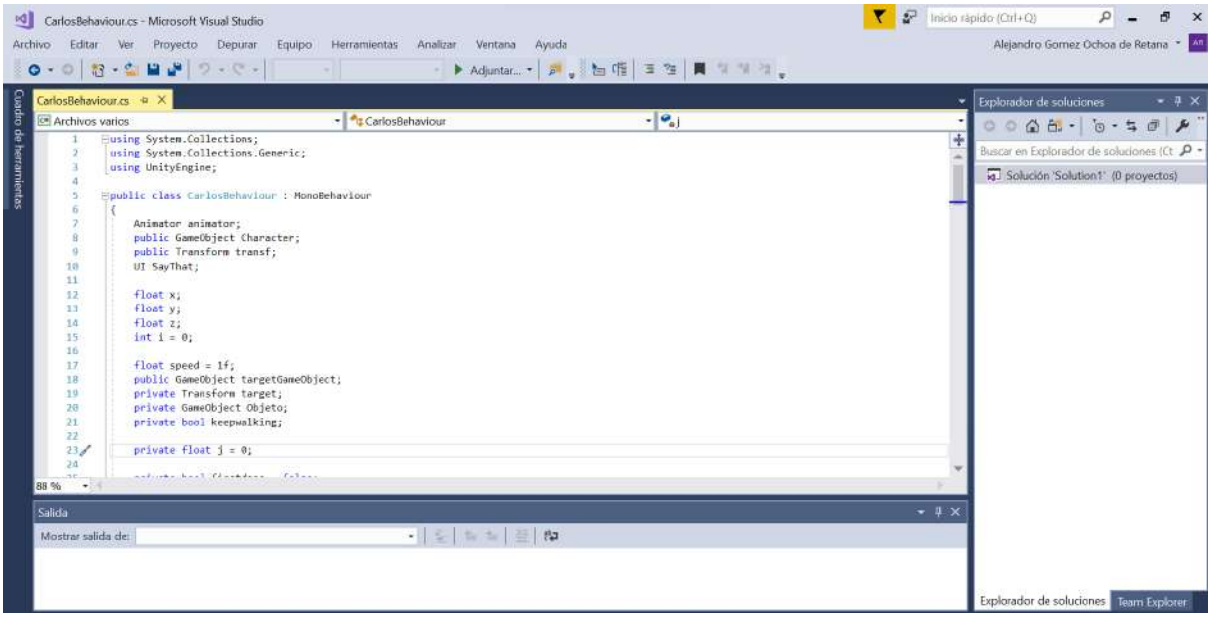


Figura 13: Visual Studio

6. Alternativas existentes a Unity

Uno de los grandes puntos fuertes que tiene Unity es la gran comunidad de usuarios que tiene. Esto permite tener acceso a multitud de documentación, foros y comunidades donde se preguntan y resuelven dudas, donde se explican diferentes métodos y técnicas nuevas, etc. Además, es uno de los motores predilectos para aprender a desarrollar videojuegos, ya que supone una puerta de acceso perfecta para aquellos que quieren incursionar en esta industria.

Existe únicamente un competidor: Unreal Engine, otro gran motor de desarrollo. Sin embargo, comparando ciertos aspectos de cada uno de ellos, podríamos decir que lo más conveniente para realizar este proyecto sería hacer uso de Unity. Los vemos a continuación:

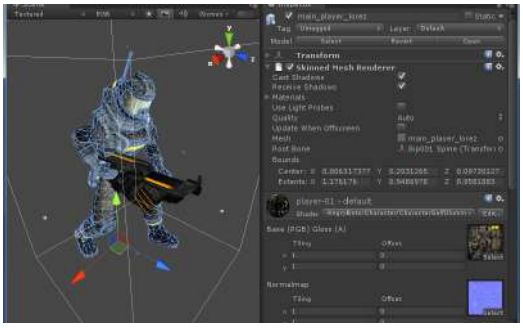


Figura 14: Ejemplo Unity

En cuanto a las plataformas de desarrollo, Unity es el motor multiplataforma líder en la industria por la versatilidad que ofrece a la hora de desarrollar videojuegos. Además, tiene mucha más ventaja a la hora de realizar desarrollos para dispositivos móviles; tanto por virtud propia, como por defecto de Unreal; debido a la poca optimización en los proyectos orientados a móviles. Aunque en principio la intención del proyecto no es lanzar el videojuego para dispositivos móviles, esto no deja de ser un factor diferenciador.

Generalmente, alguien que empieza desde cero suele empezar a trabajar de manera más rápida con Unity; ya que tiene una curva de aprendizaje muy fácil tanto por la estructura de su editor como por el uso de un lenguaje de programación sencillo como puede ser C#. Además, Unity tiene una gran cantidad de documentación disponible, tanto en su página oficial, como en multitud de foros y páginas; ya que cuenta con una comunidad de usuarios muy grande, activa y comprometida. Por el contrario, Unreal puede tener una curva de aprendizaje más alta al inicio, en parte debido a que no hay tanta documentación como en el caso de Unity y también por el propio editor y su lenguaje de programación, el cual puede ser algo más abrumador para alguien que recién empieza.

Unity utiliza C# (C Sharp), un lenguaje de programación orientado principalmente a objetos. Por otro lado, Unreal Engine permite programar tanto con el lenguaje C++, como mediante Blueprints. Los Blueprints se organizan entre nodos, y permiten programar y crear videojuegos de una forma más sencilla y visual. Aunque en este aspecto no se pueda decir que uno es mejor que el otro, sí que es cierto que en mi caso particular Unity me será de mayor facilidad dado que mi conocimiento en Blueprints es muy escaso comparado con el lenguaje C#.

Por último, ambos motores ofrecen una potencia y calidad con los mayores estándares de calidad (Asensio, n.d.).

	Unity	Unreal Engine
Plataformas de desarrollo	✓	
Curva de aprendizaje	✓	
Lenguajes de programación	✓	
Potencia y calidad	✓	✓

Tabla 1: Unity vs Unreal

7. Planificación temporal

7.1. Tareas

Antes de analizar las distintas tareas de las que consta el proyecto, debemos definir una serie de objetivos: la cantidad de trabajo diario, la fecha límite deseada y la fecha prevista de lectura.

Dado que la fecha de lectura de nuestro proyecto se dará a finales de enero, estableceremos esta como el día límite para la realización del trabajo. Para ello, comenzaremos a trabajar a partir del 1 de septiembre de 2021 asumiendo un trabajo diario de 4 horas al día y 6 días a la semana.

A continuación, expondré las distintas tareas planificadas para la realización del proyecto:

Tareas de Gestión:

- *TG1* - Definir Alcance: Esta primera tarea consiste en definir el alcance del proyecto en el contexto de su estudio, indicando el objetivo, la razón del proyecto, los medios, etc. - *10h*

- *TG2* - Planificación Temporal: Planificación en el tiempo de nuestro proyecto. Proporcionar una descripción de las diferentes fases del proyecto y la metodología llevada a cabo para cada una de ellas. - *5h*

- *TG3* - Gestión Económica y Sostenibilidad: Autoevaluación y análisis sobre la sostenibilidad, además de tratar la dimensión económica, hacer un presupuesto, reflexión final, etc. - *10h*

- *TG4* - Planificación de las escenas: Reunión individual con una psicóloga experta para poder definir las escenas a crear en el videojuego. El objetivo es realizar varias reuniones a lo largo del proyecto para asegurar el correcto desarrollo de las escenas. Dado que es un proyecto que siempre se va a poder mejorar y completar en mayor detalle y que el tiempo que tenemos es limitado, desarrollaremos únicamente 3 niveles. - *30h*

- *TG5* - Integración del documento final: Elaborar un documento final que reúna una introducción y contextualización, justificación, alcance, gestión económica, planificación temporal, etc. Es decir, reunir todos los aspectos tratados hasta ahora. - *20h.*

- *TG6* - Memoria: Elaboración de la memoria final a entregar. - *30h*

Tareas de Desarrollo:

- *TD1* - Planificación de la primera escena: Analizar y decidir la representación más adecuada de la primera escena del proyecto. - *20h*

- *TD2* - Planificación de la segunda escena: Analizar y decidir la representación más adecuada de la segunda escena del proyecto. - *20h*

- *TD3* - Planificación de la tercera escena: Analizar y decidir la representación más adecuada de la tercera escena del proyecto. - *20h*

- *TD4* - Planificación del informe de resultados: Analizar y definir el sistema para valorar el comportamiento del jugador. - *20h*

- *TD5* - Búsqueda: Búsqueda tanto de conocimientos sobre todos aquellos programas requeridos, tanto antes como durante el proyecto, como de objetos 3D a utilizar. - *50h*

- *TD6* - Elaboración de la primera escena: Desarrollar, mediante la herramienta Unity, la primera escena del juego. Incluye revisión, profundización en los detalles, prueba de errores, etc. Al finalizar, la escena debe quedar terminada y con una funcionalidad perfecta. - *50h*

- *TD7* - Elaboración de la segunda escena: Desarrollar, mediante la herramienta Unity, la segunda escena del juego. Incluye revisión, profundización en los detalles, prueba de errores, etc. Al finalizar, la escena debe quedar terminada y con una funcionalidad perfecta. - *50h*

- *TD8* - Elaboración de la tercera escena: Desarrollar, mediante la herramienta Unity, la tercera escena del juego. Incluye revisión, profundización en los detalles, prueba de errores, etc. Al finalizar, la escena debe quedar terminada y con una funcionalidad perfecta. - *50h*

- *TD9* - Elaboración del informe de resultados: Desarrollar, mediante la herramienta Unity y el sistema de valoración decidido, el informe de resultados. - *50h*

- *TD10* - Pruebas: Comprobación individual y global para cada una de las escenas creadas. - *20h*

- *TD11* - Unificación de todas las escenas existentes: Conectar todas las escenas del videojuego, de forma que al terminar una comience la siguiente. Al terminar esta tarea, el videojuego ya debería tener una forma prácticamente definitiva. - *15h*

7.2. Estimación inicial y Gantt

Para poder realizar el proyecto de forma correcta y ordenada he dividido todas las tareas mencionadas en 4 Sprints.

En el primer Sprint trataré de realizar las tareas Definir Alcance, Planificación Temporal, Gestión Económica y de Sostenibilidad, Planificación de Escenas y, por último, Integración del Documento Final. He estimado que la duración de este Sprint será de aproximadamente un mes, comenzando a principios de septiembre y acabando a principios de octubre.

El segundo Sprint comenzará con la Planificación de la primera escena, seguido de la segunda, de la tercera y del informe de resultados. También constará el tiempo invertido en la búsqueda de conocimiento sobre todo aquello que necesitaremos para el trabajo. El tiempo estimado para este Sprint es de aproximadamente 7 semanas, desde la primera semana de octubre hasta principios de diciembre.

Con la Elaboración de la primera escena empezará el tercer Sprint, seguido de la Elaboración de la segunda, de la tercera y del informe de resultados. Para terminar, realizaremos una serie de pruebas para garantizar que cada una de las escenas funcione correctamente. El tiempo previsto para este Sprint es de un mes.

Para finalizar, el cuarto y último Sprint abarcará la Unificación de todas las escenas existentes y la elaboración final de la Memoria, con un tiempo estimado de aproximadamente dos semanas, terminando a mediados de enero.

ID	Tarea	Horas	Dependencias	Sprint
TG1	Definir Alcance	10	-	1
TG2	Planificación Temporal	5	TG1	1
TG3	Gestión Económica y Sostenibilidad	10	TG2	1
TG4	Planificación de las escenas	30	TG2	1
TG5	Integración del documento final	20	TG4	1
TG6	Memoria	30	TD8	4
TD1	Planificación de la primera escena	20	TG5	2
TD2	Planificación de la segunda escena	20	TD1	2
TD3	Planificación de la tercera escena	20	TD2	2
TD4	Planificación del informe de resultados	20	TD3	2
TD5	Búsqueda	50	TD3	2
TD6	Elaboración de la primera escena	50	TD1	3

TD7	Elaboración de la segunda escena	50	TD2	3
TD8	Elaboración de la tercera escena	50	TD3	3
TD9	Elaboración del informe de resultados	50	TD4	3
TD10	Pruebas	20	TD9	3
TD11	Unificación de todas las escenas existentes	15	TD10	4
Horas totales		**470**		

Tabla 2: Estimación inicial de las tareas

A continuación, en el siguiente diagrama, podemos ver la representación de la planificación de todas las tareas descritas en un diagrama de Gantt, en base a las horas planificadas para cada tarea, las horas de trabajo al día mencionadas y la fecha de inicio del proyecto.

Figura 15: Diagrama de Gantt

8. Gestión de riesgos: Planes alternativos y obstáculos

Para el correcto desarrollo del proyecto y prevenir cualquier tipo de infortunio, debemos ser capaces de analizar los distintos riesgos a los que nos vamos a enfrentar y anticipar una solución a cada uno de ellos.

El principal riesgo que podemos detectar es la presencia de *bugs* en nuestro código. En la gran mayoría de proyectos con alto nivel de código suele haber presencia de ellos. Para evitarlo haremos uso de *testing*.

Otro riesgo que podemos detectar es el incumplimiento con la planificación del tiempo. Actualmente sí que podemos cumplir con los requisitos dados, pero probablemente, durante el periodo establecido, se dé el caso de que no pueda ser así, ya sea por las demás asignaturas universitarias, el trabajo, o demás. Para poder evitar esto en la medida de lo posible, intentaremos compensar esas horas 'perdidas' otro día en el que podamos permitírnoslo, además de haber cuadrado el tiempo estimado por tarea teniendo esto en cuenta.

9. Presupuesto

9.1. Elementos

9.1.1. Recursos humanos

Para definir el coste de cada una de las tareas establecidas en el diagrama de Gantt, necesitaremos calcular el coste de personal que supondrá el total de cada una de ellas.
Definiremos los distintos roles de los empleados que necesitaremos para llevar a cabo el proyecto calculando el coste de cada uno de ellos, conociendo el número de horas que trabajarán y su respectivo salario (*Sueldos De La Empresa*, n.d.).

Rol	Salario / hora (€ bruto)	Salario / hora (€ bruto) + Seguridad Social (x1.3€)
Desarrollador backend senior (DB)	17.37	22.59
Desarrollador frontend senior (DF)	15, 79	20,53
Diseñador gráfico (DG)	9.21	12
Psicólogo psicoanalista (PP)	13.85	18
Gestor del proyecto (GP)	15.96	20,75

Tabla 3: Costes estimados por rol / hora

Ahora identificaremos el trabajo de cada uno de ellos en cada una de las distintas tareas definidas en el diagrama de Gantt y calcularemos el coste final de su totalidad.

ID	Tarea	Horas	Horas / empleado					Coste (€)
			DB	DF	DG	PP	GP	
TG1	Definir Alcance	10				2	8	202
TG2	Planificación Temporal	5					5	103,75
TG3	Gestión Económica y Sostenibilidad	10					10	207,5
TG4	Planificación de las escenas	30					30	622.5
TG5	Integración del documento final	20					20	415

TG6	Memoria	30					30	622.5
TD1	Planificación de la primera escena	20				20		360
TD2	Planificación de la segunda escena	20				20		360
TD3	Planificación de la tercera escena	20				20		360
TD4	Planificación informe de resultados	20				20		360
TD5	Búsqueda	50			50			600
TD6	Elaboración de la primera escena	50	15	15	20			886,8
TD7	Elaboración de la segunda escena	50	15	15	20			886,8
TD8	Elaboración de la tercera escena	50	15	15	20			886,8
TD9	Elaboración del informe de resultados	50	15	15	20			886,8
TD10	Pruebas	20	10	10				431,2
TD11	Unificación de todas las escenas existentes	15	7,5	7,5				323,4
Coste Total (CPA)								**8515,05**

Tabla 4: Costo estimado de las tareas (CPA)

9.1.2. Costes genéricos

A continuación, deberemos contemplar también los costes más generales, aquellos relacionados con el Hardware o Software o con el coste del entorno de trabajo, material, etc.

Para calcular la amortización de los productos Hardware y Software, dividiremos su respectivo precio entre su vida útil. Dado que queremos finalizar el proyecto en diciembre,

esta corresponderá a los 135 días de estos 4 meses y medio (aproximadamente). El precio del paquete básico de Office 365 es de 50,4€ anuales por usuario. Puesto que somos 5 empleados, el precio total será de 252€. Para calcular el precio de la luz, supondremos que utilizaremos 3 fluorescentes (18W aproximadamente cada uno) y un portátil (150W) a un precio de 0.22618 €/kWh, con 4h de trabajo al día. El precio de Internet hoy en día es de 77,50€ al mes.

Recurso	Coste (€)	Amortización (€)
Office 365	252	
Unity	Gratuito	
Escritorio	70	0,52
Silla de escritorio	30	0,23
Lámpara	20	0,15
Ordenador	1.000	7,4
Internet	387,5	
Gantt Project	Gratuito	
Electricidad	25	
Coste	**1784,5**	
Coste Total (CPA+CG)	**10299,55**	

Tabla 5: Coste genérico estimado (CG)

9.1.3. Contingencias

El cálculo del coste de las contingencias se refiere a aquel cuyo propósito es cubrir el coste de los riesgos previstos del proyecto. En aquellos pertenecientes al sector informático, como es este, suele oscilar entre el 10% y el 20%, por lo que supondremos un coste adicional del 15% del total.

Coste Contingencia (CC) = (CPA+CG) * 0,15 = 1544,94 €
Coste total (CPA+CG+CC) = 11844,5 €

9.1.4. Imprevistos

Para calcular el coste de los imprevistos que se puedan llegar a dar, supondremos una probabilidad para cada uno de ellos y calcularemos el coste que supondría la solución para estos. Dicho coste, multiplicado por su probabilidad, nos dará el coste por imprevisto a calcular. El coste total será la suma de todos y cada uno de ellos.

Riesgo	Probabilidad %	Horas totales	Horas					Coste Estimado (€)	Coste (€)
			DB	DF	DG	PP	GP		
Bugs	45	20	10	10	10			551,2	248,04
Planificación	25	20					20	415	103,75
Total									**351,79**

Tabla 6: Coste de imprevistos estimado (CI)

Coste total (CC+CG+CC+CI) = 11844,5 + 351,79 = 12196,3 €

9.2. Control de Gestión

Necesitaremos un sistema de control del presupuesto, para supervisar las posibles desviaciones de los costes durante la ejecución del proyecto. Para ello, calcularemos la diferencia entre el tiempo real empleado y el tiempo previsto para cada una de las tareas establecidas. Con este objetivo implantamos las siguientes métricas:

Desvío en coste por tarifa:

Desvío de mano de obra en precio: (coste estimado - coste real) * consumo de horas real

Desvío en eficiencia:

Desvío de mano de obra en consumo: (consumo horas estimadas - consumo horas reales) * coste estimado

Desvío en totales:

Desvío total en mano de obra: total coste estimado mano de obra – total coste real mano de obra.

9.3. Planificación final: Desviaciones

Una vez realizado el proyecto, viendo la planificación inicial me doy cuenta de que las estimaciones de ciertas tareas no han sido acertadas. Las principales tareas en las que se ha visto una mayor desviación son:

- *TD4*: Búsqueda - Ha sido una tarea que realmente no terminó hasta que terminé con el proyecto, ya que ha habido pequeños cambios que he realizado durante el transcurso de este que no estaban planificados en un principio y que requerían de búsqueda adicional de información.

- *TD5*: Elaboración de la primera escena - Tarea en la que mayor desviación ha habido. Al ser la primera escena es donde he cometido más errores de primerizo que han provocado que le dedique un tiempo bastante mayor al que estimé.

- *TD6*: Elaboración de la segunda escena - Si bien no cometí tantos errores como en la anterior, ciertas dudas sobre cómo recrear con exactitud la situación, probablemente debidas a una planificación con falta de exactitud, provocaron que también le dedicara tiempo de más a esta tarea.

Tarea	Horas planificadas	Horas reales	Diferencia
Búsqueda	50	~75	+25
Elaboración de la primera escena.	50	~80	+30
Elaboración de la primera escena.	50	~60	+10
Total	**150**	**215**	**65**

Tabla 7: Desviaciones de la planificación (horas)

Esto obviamente altera el presupuesto que se hizo en un principio. Con respecto a recursos humanos, recordando el coste de estas tareas en la Tabla 4 del punto 9.1.1., vemos que este es:

ID	Tarea	Horas	Horas / empleado					Coste
			DB	DF	DG	PP	GP	
TD4	Búsqueda	50			50			600
TD5	Elaboración de la primera escena	50	15	15	20			886,8
TD6	Elaboración de la segunda escena	50	15	15	20			886,8

Tabla 8: Coste estimado inicialmente de las tareas

Ahora, con el tiempo real de estas tareas, los datos quedarían así:

ID	Tarea	Horas	Horas / empleado					Coste (€)
			DB	DF	DG	PP	GP	
TD4	Búsqueda	75			75			900

TD5	Elaboración de la primera escena	80	24	24	32			1698,88
TD6	Elaboración de la segunda escena	60	18	18	24			1274,16

Tabla 9: Desviaciones de la planificación (coste)

ID	Tarea	Coste inicial (€)	Coste final (€)	Desviación (€)
TD4	Búsqueda	600	900	300
TD5	Elaboración de la primera escena	886,8	1698,88	812,08
TD6	Elaboración de la segunda escena	886,8	1274,16	387,36
Total				**1499,44**

Tabla 10: Desviación final de la planificación (coste)

10. Informe de sostenibilidad

10.1. Autoevaluación

Entendemos como un sistema sostenible aquel sistema en el que se conjugue la sostenibilidad ambiental, económica y social. Son distintos aspectos que hemos ido tratando durante estos últimos años, incorporándose en distintas ocasiones en la elaboración de distintos proyectos.

Bajo mi punto de vista, el fundamento principal de la sostenibilidad ambiental es conservar y proteger el medio ambiente de forma indefinida, satisfaciendo las necesidades actuales sin comprometer las de las generaciones futuras. Entre las 3 que vamos a ver, es de la que menos conozco y la que menos he trabajado. Desconozco la repercusión que puede tener este proyecto en el aspecto medioambiental y, por ende, posibles soluciones para remediarlo.

Respecto a la sostenibilidad social, sí que la he trabajado en más de una ocasión. En una de ellas desarrollamos una aplicación cuyo objetivo era ofrecer ayuda a las personas vulnerables al covid-19 en época de pandemia. Creo que con este ejemplo se puede apreciar aspectos positivos de dicha sostenibilidad, al ofrecer una mejora en la calidad de vida de las personas que utilizaran dicha aplicación y conectarlas entre ellas.

También hemos considerado a menudo la sostenibilidad económica en diversos proyectos. En uno de ellos planificamos el número de empleados y el gasto que esto supondría, además de otros gastos como el enfocado al entorno de trabajo, al análisis y prevención de riesgos, a la inversión, etc. De todas formas, opino que aún tengo bastantes lagunas en este aspecto.

A pesar de haber tratado dichas dimensiones de sostenibilidad a lo largo de la carrera, unas más que otras como ya he mencionado, considero que mi conocimiento sobre todas ellas sigue siendo escaso, y que aún queda mucho por aprender.

10.2. Análisis del proyecto

10.2.1. Dimensión Ambiental

Referente a PPP: ¿Has estimado el impacto ambiental que tendrá la realización del proyecto? ¿Te has planteado minimizar el impacto, por ejemplo, reutilizando recursos?

A la hora de plantear el proyecto no sé estimó el impacto que éste podría llegar a tener en el medio ambiente. Cabe destacar que la finalidad de este proyecto corresponde al mero desarrollo software de un videojuego, por lo que carecería de residuos electrónicos o demás posibilidades de producir realmente un impacto ambiental.

Referente a la Vida Útil: ¿Cómo se resuelve actualmente el problema que quieres abordar (estado del arte)? ¿En qué mejorará ambientalmente tu solución a las existentes?

Actualmente no existe ninguna solución para el problema que quiero tratar, o al menos ninguna tecnológica. La única solución real para aquella persona que se vea en esta situación es asistir al psicólogo para recibir un diagnóstico real. Al ser un videojuego, el paciente puede hacer uso de él desde cualquier lugar, sin necesidad de desplazarse a ninguna consulta, lo que podría acarrear consecuencias negativas para el medio ambiente.

10.2.2. Dimensión Económica

Referente a PPP: ¿Has estimado el coste de la realización del proyecto (recursos humanos y materiales)?

Para poder llevar a cabo correctamente la planificación y gestión de un proyecto grande, debemos calcular previamente el presupuesto que será necesario para llevarlo a cabo, para ayudarnos a considerar si vale la pena invertir en este o no. Para ello, hemos estimado el coste que llegarían a tener los empleados a nuestra disposición en función de su salario y las horas de trabajo estimadas, así como otro tipo de costes como los genéricos, el de contingencia y aquel destinado a la solución de posibles imprevistos.

Referente a la Vida Útil: ¿Cómo se resuelve actualmente el problema que quieres abordar (estado del arte)? ¿En qué mejorará económicamente tu solución a las existentes?

Actualmente no existe ninguna solución para el problema que quiero tratar, o al menos ninguna tecnológica. La única solución real para aquella persona que se vea en esta situación es asistir al psicólogo para recibir un diagnóstico real. Se podría decir que la única diferencia económica en comparación con la mencionada sería que mi proyecto sería totalmente gratuito, a disposición de todo aquel que quisiera hacer uso de él.

10.2.3. Dimensión Social

Referente a PPP: ¿Qué crees que te va a aportar a nivel personal la realización de este proyecto?

A nivel personal, gracias a la realización de este proyecto puedo llegar a mejorar mis habilidades para el desarrollo de software, el área de la informática más interesante para mí. Además, puedo enriquecer mis conocimientos sobre la psicología, campo muy distinto pero que a su vez siempre me ha sido de gran interés.

Referente a la Vida Útil: ¿Cómo se resuelve actualmente el problema que quieres abordar (estado del arte)? ¿En qué mejorará socialmente (calidad de vida) tu solución a las existentes?

Actualmente no existe ninguna solución para el problema que quiero tratar, o al menos ninguna tecnológica. La única solución real para aquella persona que se vea en esta situación es asistir al psicólogo para recibir un diagnóstico real. Con mi proyecto, las personas en duda sobre qué es exactamente lo que les sucede, podrán obtener un poco más de claridad sobre esto y, por ejemplo, valorar o no si necesitan de ayuda médica, psicológica o demás.

Referente a la Vida Útil: ¿Existe una necesidad real del proyecto?

Existen muchas personas con dificultades a la hora de socializar, pero esto puede ser debido a muchas cosas, no solo a una fobia social, como puede ser la depresión o el autismo. Con la ayuda de mi proyecto, estas personas podrán identificarse más en una situación o en otra y ver con más claridad qué les sucede con respecto a la sociabilidad.

11. Requisitos

11.1. Obtención de requisitos

Para establecer los requisitos funcionales del proyecto se han mantenido reuniones con la psicóloga psicoanalista, la cual me ha ayudado con aquellas funcionalidades referentes a las situaciones planteadas en el videojuego y la toma de decisiones del jugador respecto a estas. A su vez me ha ayudado a establecer un sistema para la valoración de dichas decisiones y realizar así el informe de resultados.

Por otro lado, un objetivo para que el proyecto funcione correctamente es hacer sentir al jugador que experimenta los escenarios en primera persona. Para ello solicito ciertos datos del jugador, como su propio nombre, para poder utilizarlo dentro del videojuego.

11.2. Requisitos funcionales

Entre los requisitos funcionales de este proyecto encontramos los siguientes, especificados en forma de historias de usuario:

- **Historia de usuario**: Añadir información inicial.
  Descripción: Como usuario quiero insertar mi nombre, edad y sexo.
  Criterios de aceptación:
  1. El nombre debe tener valor.
  2. El sexo debe tener valor.
  3. La edad debe tener valor.
  4. El valor de la edad debe ser un número entero.

- **Historia de usuario**: Presentarme.
  Descripción: Como usuario quiero tener la oportunidad de presentarme.
  Criterios de aceptación:
  1. El usuario debe pulsar la barra espaciadora cuando se le dé la oportunidad.

- **Historia de usuario**: Intervenir en conversación sobre la distancia hasta la universidad.
  Descripción: Como usuario quiero tener la oportunidad de intervenir en la conversación sobre la distancia hasta la universidad y el madrugar aportando mi opinión.
  Criterios de aceptación:
  1. El usuario debe seleccionar una de las dos opciones visibles, pulsando la tecla correspondiente a la flecha izquierda o la derecha.

- **Historia de usuario**: Intervenir en conversación sobre el motivo por el que se escoge esta carrera.
  Descripción: Como usuario quiero tener la oportunidad de intervenir en la conversación sobre el motivo por el que cada uno de nosotros hemos escogido estudiar esta carrera.
  Criterios de aceptación:

1. El usuario debe seleccionar una de las dos opciones visibles, pulsando la tecla correspondiente a la flecha izquierda o la derecha.

➢ **Historia de usuario**: Llamar al camarero.
Descripción: Como usuario quiero tener la oportunidad de llamar al camarero cuando me lo piden.
Criterios de aceptación:
1. El usuario debe pulsar la barra espaciadora cuando se le dé la oportunidad.

➢ **Historia de usuario**: Puntuar emociones.
Descripción: Como usuario quiero poder evaluar las emociones que se me presentan cuando me lo piden.

➢ **Historia de usuario**: Generar informe de resultados.
Descripción: Como usuario quiero poder obtener el informe de resultados para ver mis resultados correspondientes a la toma de decisiones realizadas.
Criterios de aceptación:
1. El usuario debe haber completado el juego entero.

11.3. Requisitos no funcionales

Entre los requisitos no funcionales vemos:

➢ **Requisito**: Tener una mecánica de juego sencilla.
Categoría: 11a) Requisitos de Capacidad de Uso y Humanidad - Facilidad de Utilización.

➢ **Requisito**: Intuitivo y sencillo de jugar.
Categoría: 11c) Requisitos de Capacidad de Uso y Humanidad - Aprendizaje.

➢ **Requisito**: Causar una sensación de primera persona al jugador.
Categoría: 10b) Requisitos de Percepción - Estilo.

➢ **Requisito**: El videojuego estará disponible únicamente para Windows, en un principio.
Categoría: 14c) Requisitos de Preservación y Soporte - Adaptabilidad.

➢ **Requisito**: El diseño debe ser agradable.
Categoría: 10a) Requisitos de Percepción - Apariencia.

12. Diseño del sistema

12.1. Patrones

El patrón Entity-Component forma parte de la arquitectura base de Unity. Es un patrón de diseño donde primero definimos la jerarquía de los elementos que componen la aplicación (entidades) y luego definimos las características y los datos que cada uno contendrá (componentes).

Este es un buen patrón para aliviar los problemas de herencia múltiple, donde una estructura de clase compleja puede introducir problemas como el problema de diamante donde una clase D, heredando dos clases, B y C, con la misma clase base A, puede introducir conflictos porque B y C modifican las características de A de manera diferente. Este tipo de problemas pueden ser comunes en el desarrollo de juegos donde la herencia se usa a menudo extensivamente.

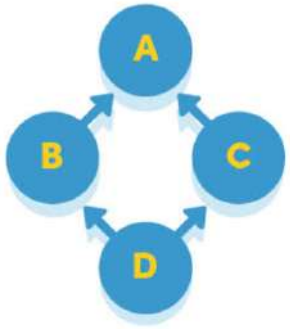


Figura 16: Ejemplo EC

Unity es una plataforma de desarrollo basada en EC (Entity-Component), donde todas las entidades son instancias Game Object y las características que las hacen ser “visibles”, “móviles”, “interactivas”, etc., son proporcionadas por las clases que se extienden.

El panel de jerarquía y el panel Inspector del editor de Unity proporcionan una forma poderosa de ensamblar nuestra aplicación, adjuntar componentes, configurar su estado inicial y arrancar nuestro juego con mucho menos código fuente de lo que lo haría normalmente.

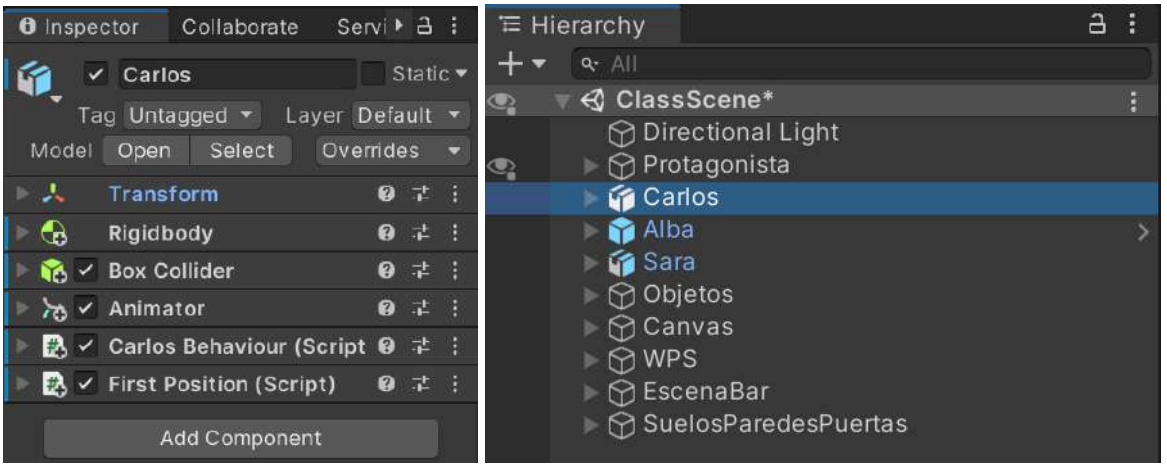


Figura 17: Inspector y Jerarquía

12.2. Diseño de Personajes

Existen únicamente cinco personajes en nuestro videojuego: nuestro protagonista y cuatro NPC (*Non Playable Character*).

12.2.1. Protagonista

El videojuego dispone de dos protagonistas, dependiendo del sexo del jugador. En la Figura 18 vemos el diseño del personaje si el jugador fuera una mujer o en caso de que este fuera un hombre.

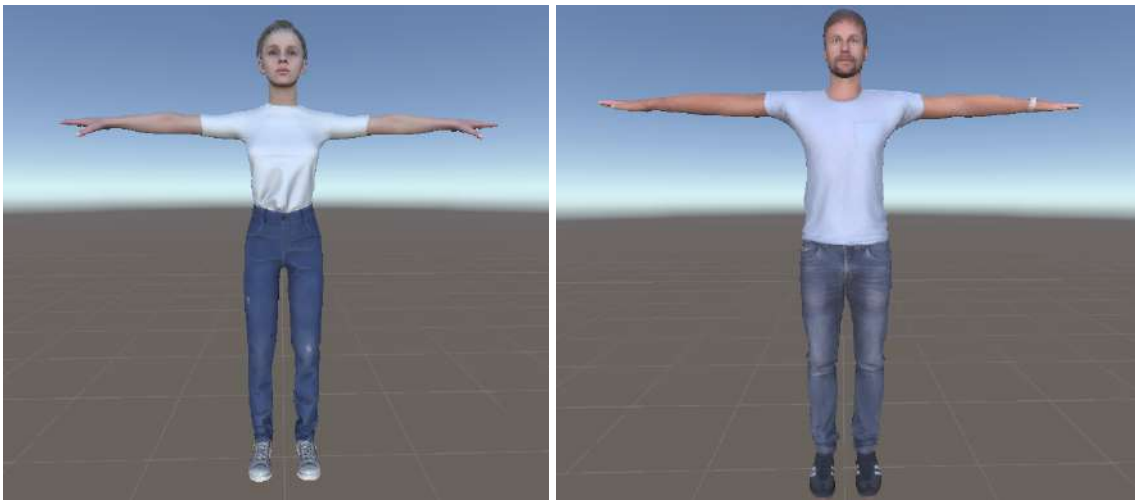


Figura 18: Mujer y Hombre Protagonistas

El nombre de estos personajes será el nombre del propio jugador que habrá introducido en la primera escena.

12.2.2. NPC

Como he mencionado anteriormente, dispondremos de cuatro NPC. La función de tres de ellos es intentar socializar con el protagonista y ayudar al jugador a sumergirse en el juego. El NPC restante se trata del camarero, cuya funcionalidad veremos más adelante.

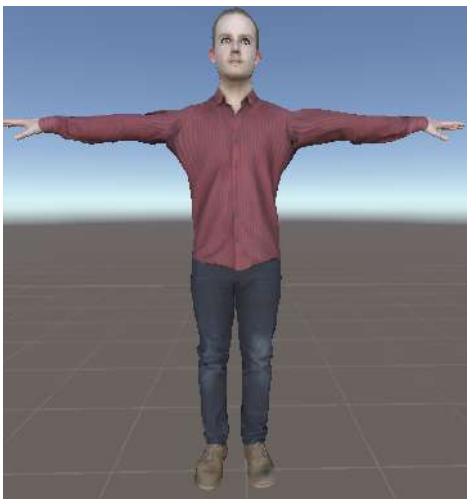


Figura 19: Carlos

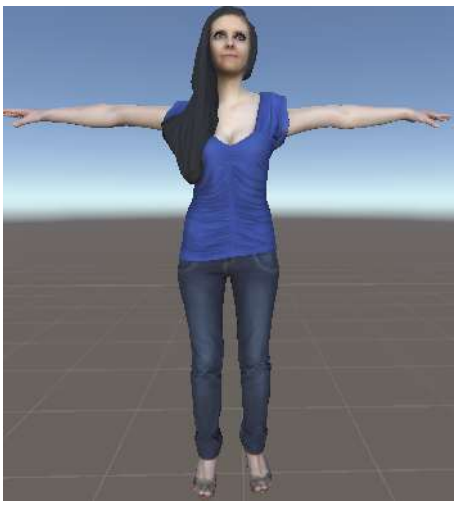


Figura 20: Alba

Figura 21: Alba

Figura 22: Camarero

12.3. Diseño de Escenas

12.3.1. BeginScene

La primera escena se trata de la pantalla principal. Aquí se le solicitará al jugador su nombre, edad y sexo. Una vez introducidos estos datos, podrá iniciar el videojuego pulsando el botón de Comenzar.

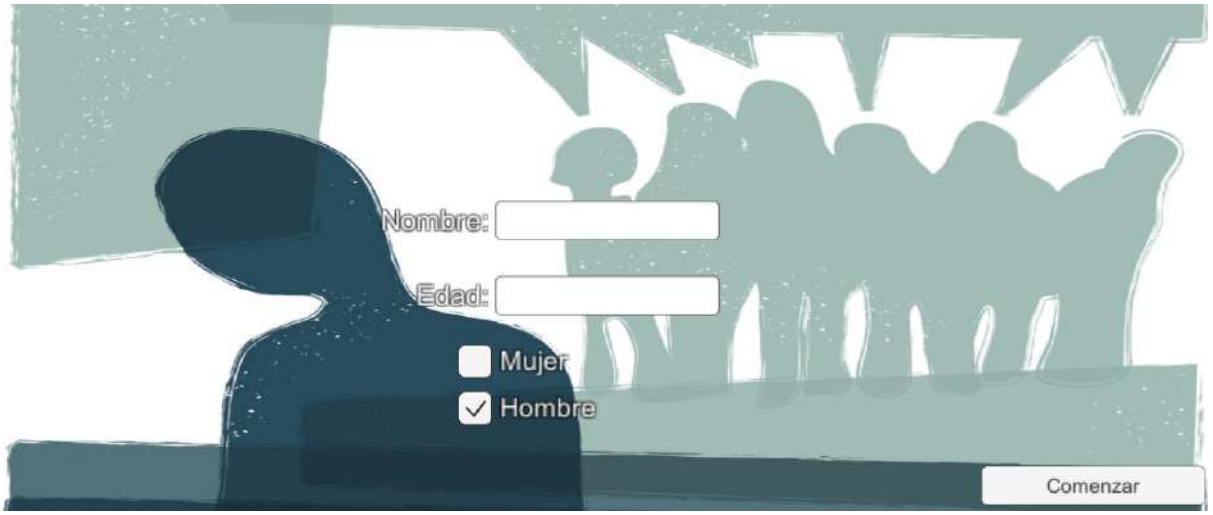


Figura 23: BeginScene

12.3.2. ClassScene

La segunda escena se presenta al finalizar una clase. Está ambientada en el primer día de universidad del primer año. Normalmente los estudiantes en esta situación intentan conocer a gente nueva y socializar ya que pasarán con esa gente los próximos años. Al comienzo de la escena, el protagonista recibe una invitación para ir a tomar algo. Dado el objetivo de este proyecto, este no tiene la posibilidad de rechazar la invitación, la aceptará para poder observar cómo reacciona más tarde a las situaciones programadas. Una vez aceptada la invitación pondrán camino al bar.

Figura 24: ClassScene muestra 1

En esta escena al protagonista solo se le presenta una toma de decisión: presentarse a sí mismo o no. Según lo investigado, y confirmado por la psicóloga, a las personas que padecen de fobia social se les hace muy complicado dar este paso, y más aún cuando no se lo han preguntado de forma directa. Por ello, se le da la oportunidad al jugador de decir su nombre justo después de que Alba se presente, mientras mantiene una conversación con esta. Sin embargo, Alba no le ha preguntado por su nombre directamente.

Figura 25: ClassScene muestra 2

En la situación se presenta la posibilidad de presentarse y un Timer. Pulsando la barra espaciadora, como se ve en la Figura 25, el jugador se presentará. El Timer especifica el tiempo que tiene el jugador para tomar la decisión. En caso de que no pulse la barra espaciadora, el jugador no se presentará. En cualquier caso, al finalizar el Timer y cruzar la puerta al salir, comenzará la tercera escena.

12.3.3. BarScene

Esta escena está ambientada en el bar de la universidad. Aquí se presentan los personajes ya sentados y habiendo pedido ya lo que quisieran pedir.

Dependiendo de la decisión tomada en la escena anterior, la escena comienza de una forma o de otra. En caso de no haberse presentado, Alba pregunta por el nombre del protagonista directamente, a lo que este responde. En caso contrario, este momento no se da y comienza realmente la escena.

Aquí se darán tres situaciones:

1. Los NPC hablan sobre un tema general, en este caso sobre la lejanía de su residencia a la universidad y sobre el hecho de madrugar para asistir a esta. Realmente el tema de conversación no es relevante. El quid de la cuestión es comprobar si el jugador participa en una conversación en la que no se le ha preguntado nada directamente. Llegado el momento se le presentará una decisión:

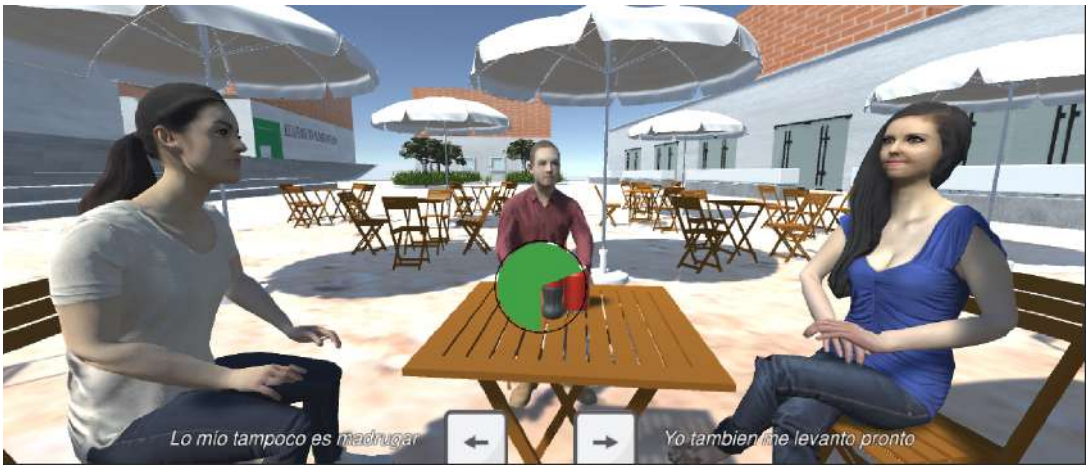


Figura 26: BarScene muestra 1

   El jugador tendrá tres opciones: escoger la opción de la izquierda, escoger la opción de la derecha o no responder. De nuevo habrá un Timer que especificará el tiempo restante para hacerlo. En caso de que se escoja una de las dos primeras opciones o el Timer se acabe, este desaparecerá junto con las opciones anteriores y la escena continuará habiendo registrado la respuesta.

2. La segunda situación es muy parecida a la primera. El tema de conversación, en este caso, será el motivo por el que cada uno de ellos está estudiando esa carrera. De nuevo el objetivo es ver si el jugador intervendrá o no en la conversación. La diferencia radica en una simple pregunta que Sara formulará: “Bueno, ¿y vosotros por qué habéis escogido esta carrera?”, siendo una pregunta directa el resto de los personajes presentes, incluido el protagonista, pero sin ser una pregunta exclusivamente para él. Tendrá, otra vez, tres opciones a escoger siguiendo el mismo formato que en la situación anterior (Figura 26). Al escoger una de las opciones, o al finalizar el Timer, estos desaparecerán y la escena continuará.

3. La última situación es distinta. En este momento los personajes se dan cuenta de que quizás llegan tarde a la siguiente clase, por lo que quieren pedir la cuenta. Es entonces cuando aparece el camarero, a lo lejos, y se presenta la posibilidad de llamarlo para pedir la cuenta. Esta situación se da comúnmente y es muy incómoda para las personas que padecen de fobia social, hasta el punto de que muchas de ellas deciden no entrar en restaurantes, cafeterías, bares, y demás para evitarla.

   Esta situación será parecida a la vista en la ClassScene, cuando el protagonista decide si presentarse o no. Se mostrarán dos elementos, el Timer y la barra espaciadora. Si pulsa esta mientras aún está visible, el protagonista llamará al camarero, este se acercará, pedirá la cuenta y el camarero se marchará. En caso de no hacerlo, será Sara la que lo haga. Seguidamente la escena terminará.

Figura 27: BarScene muestra 2

12.3.4. QuestionnaireScene

Esta escena es diferente a las demás. No se presenta ninguna simulación, sino que se le pregunta al jugador cómo se ha sentido en cada una de las situaciones ya expuestas.

En muchas ocasiones, ni siquiera uno mismo está seguro de qué es lo que siente en según qué situaciones. Por ello, en esta escena se muestran las emociones más probables que puede sentir una persona con fobia social dependiendo de cada situación planteada. Para averiguar cuáles son estas he colaborado con la psicóloga psicoanalista.

El jugador deberá evaluar, del 0 al 10, cuánto ha sentido cada una de dichas emociones en cada situación, siendo 0 lo más bajo y 10 lo más alto.

Figura 28: QuestionnaireScene Ejemplo 1ª Situación

Figura 29: QuestionnaireScene Ejemplo 4ª Situación

12.3.5. ResultsScene

En esta última escena se presentará el diagnóstico final obtenido, variando éste en función de las decisiones tomadas por el jugador. Para que tenga cierta validez, he realizado este informe con la ayuda de la psicóloga. Los resultados se mostrarán de la siguiente forma:

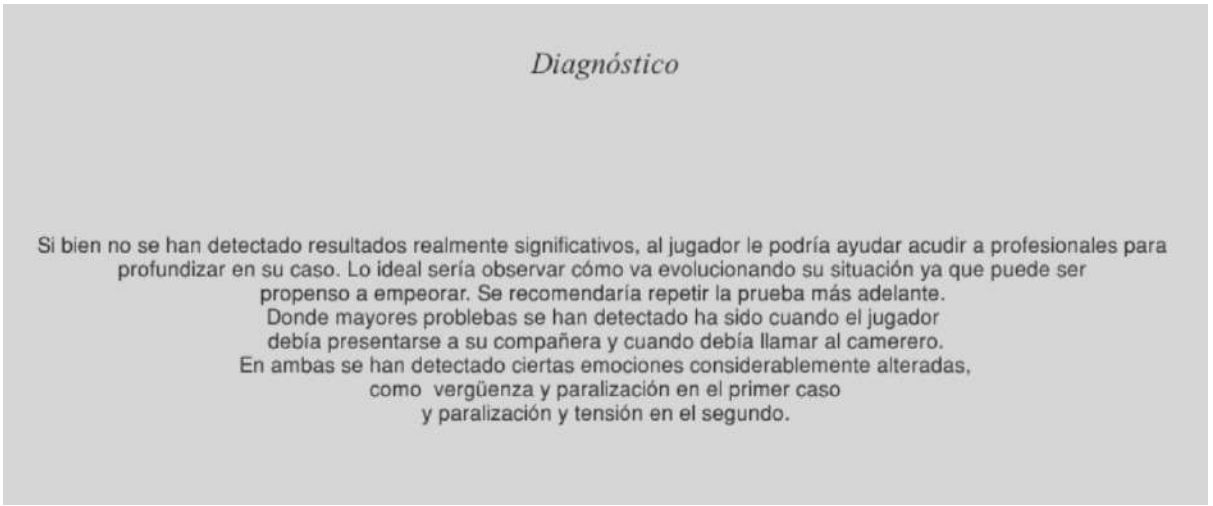


Figura 30: ResultsScene; Diagnóstico

También se mostrará un botón con la indicación ‘Jugar de nuevo’, que nos enviará de vuelta a la BeginScene.

A continuación, explicaré el método de evaluación que he establecido con ayuda de la psicóloga:

Por un lado, se evaluarán las decisiones tomadas hasta la BarScene y por el otro las puntuaciones registradas en la QuestionnaireScene. Diferenciaremos ambos en Parte 1 y Parte 2, respectivamente.

Para la Parte 1 se recogerá la siguiente información:

- ClassScene:
    - ¿Se ha presentado el jugador ante su compañera?
- BarScene:
    - ¿Interviene el jugador en la primera conversación?
    - ¿Interviene el jugador en la conversación en la que se le interpela?
    - ¿Llama el jugador al camarero?

Cada una de las respuestas se guardarán en formato booleano y se puntuarán como 0 o 1, siendo 0 si la respuesta es 'sí' y 1 si la respuesta es 'no'. Después se sumará el total y se clasificará el resultado, siendo este sobre 4. Cuanto mayor sea la puntuación, más síntomas presentará el jugador.

Evaluación de decisiones:

- 4/4: Puntuación elevada.
- 3/4: Puntuación elevada.
- 2/4: Puntuación media.
- 1/4: Puntuación baja.
- 0/4: Puntuación baja.

Para la Parte 2 el sistema de puntuación será un poco más complejo. Para cada una de las emociones presentadas en cada escena se establecerá el siguiente método de evaluación:

Evaluación de emociones:

- 8 - 10: Puntuación elevada.
- 5 - 7: Puntuación media.
- 0 - 4: Puntuación baja.

Si la escena cuenta con un mínimo de dos emociones con puntuación elevada, así se considerará la escena. Si por el contrario cuenta con un mínimo de dos emociones con puntuación media, la escena se considerará de puntuación media, y si no se cumple con ninguno de los casos anteriores se considerará de puntuación baja.

Evaluación de escenas:

- Emociones de puntuación elevada >= 2: Puntuación elevada.
- Emociones de puntuación media >= 2: Puntuación media.

- Emociones de puntuación elevada < 2 y Emociones de puntuación media < 2: Puntuación baja.

Ahora que ya tendremos una evaluación para cada escena, se necesitará una global para todo el cuestionario. Para ello utilizaremos la siguiente tabla:

Escenas (4)			**Puntuación global**
de puntuación alta	**de puntuación media**	**de puntuación baja**	
4	0	0	Elevada
3	1	0	Elevada
	0	1	Elevada
2	2	0	Media
	1	1	Media
	0	2	Media
1	3	0	Media
	2	1	Media
	1	2	Baja
	0	3	Baja
0	4	0	Media
	3	1	Media
	2	2	Baja
	1	3	Baja
	0	4	Baja

Tabla 11: método de evaluación de la QuestionnaireScene.

Daremos entonces valores numéricos a cada puntuación de la siguiente manera para terminar la evaluación de la parte 2:

- Puntuación global elevada: 4
- Puntuación global media: 2
- Puntuación global baja: 0

Ahora deberemos combinar ambas partes. Haremos esto de forma sencilla, sumando los valores de ambas y clasificando el resultado de la siguiente forma:

- 6 <= Resultado: <= 8: El diagnóstico final es negativo. El jugador muestra claramente síntomas de fobia social y es recomendable que consulte con un profesional y se plantee iniciar un tratamiento psicológico.
- 3 <= Resultado <= 5: Si bien el jugador no muestra tanta abundancia de sintomatología de fobia social como el grupo anterior, sí que aparecen algunos indicios o rasgos de esta, por lo que sería recomendable que consultara con un profesional.
- 0 <= Resultado <= 2: No hay síntomas de ansiedad social o bien los hay en escasa medida y no patológica.

13. Implementación

13.1. Unity

Para la elaboración de este proyecto, se ha necesitado hacer uso de varios servicios. El principal y que ya he mencionado anteriormente es Unity. Aquí es donde he desarrollado todo el proyecto: he creado las escenas, codificado los scripts, importado los objetos necesarios, creado animaciones, y demás.

La ventana principal del editor se compone de ventanas con pestañas que pueden ser ajustadas, agrupadas o desadjuntadas y minimizadas. Esto significa que el aspecto del editor puede ser diferente de un proyecto al otro, y de un desarrollador al siguiente, dependiendo de la preferencia personal y qué tipo de trabajo está haciendo.

El arreglo predeterminado de las ventanas nos da un acceso práctico a las ventanas más comunes. Si no estamos familiarizados todavía con las distintas ventanas de Unity, podemos identificarlas por su nombre en la pestaña.

La interfaz de Unity consta de 5 componentes principales, los cuales son: Vista de la escena, ventana de jerarquía, Inspector, ventana del proyecto y barra de herramientas (*Manual: Aprendiendo La Interfaz*, n.d.).

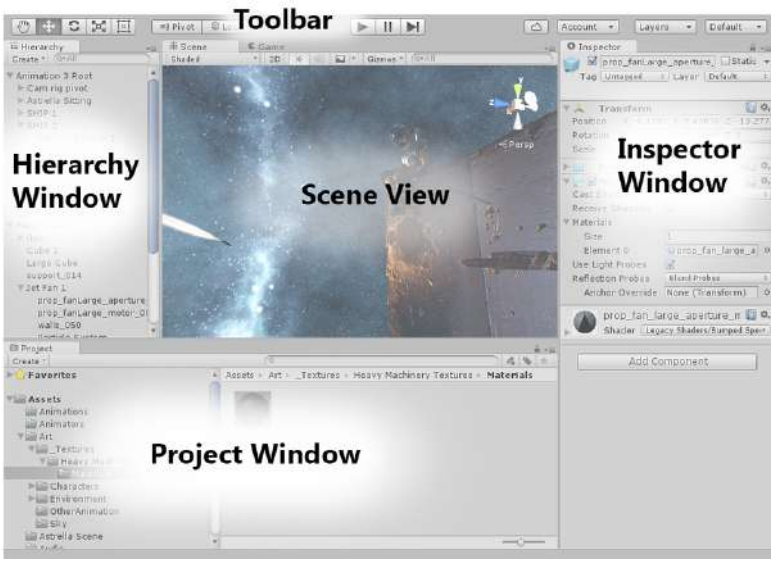


Figura 31: Interfaz

- Vista de la escena: Nos permite navegar y editar visualmente nuestra escena. La vista de escena puede ser de perspectiva 2D o 3D. Tiene varias herramientas para visualizar qué Game Objects tienen un determinado script, quitar y añadir sonidos, etc.

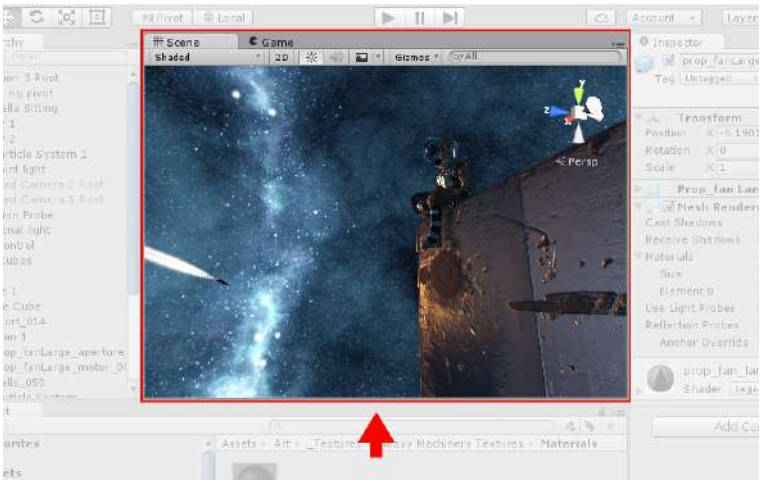


Figura 32: Vista de la Escena

- Ventana de Jerarquía: Es una representación jerárquica en forma de texto de todos los Game Objects de la escena. Revela la estructura de cómo los objetos están agrupados el uno al otro.

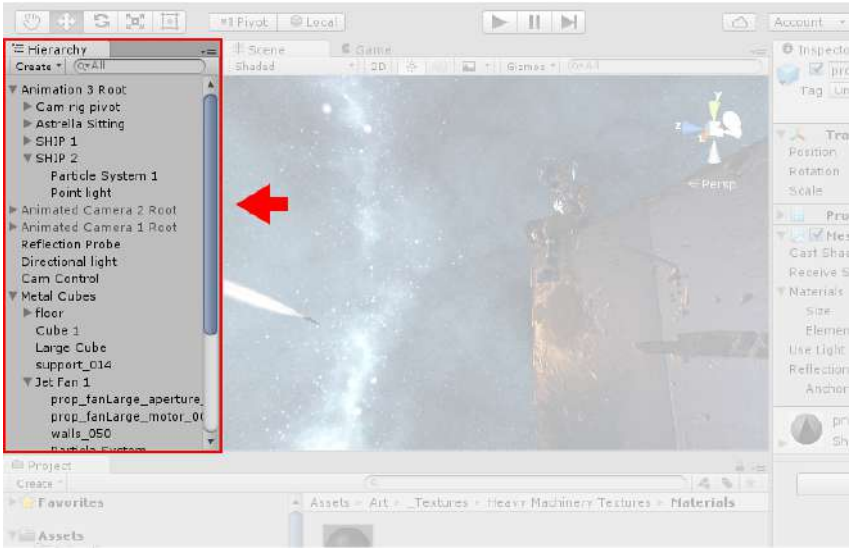


Figura 33: Ventana de Jerarquía

- Inspector: Permite ver y editar todas las propiedades del objeto seleccionado. Debido a que cada objeto tiene diferentes tipos de propiedades, el contenido del inspector va a ir variando.

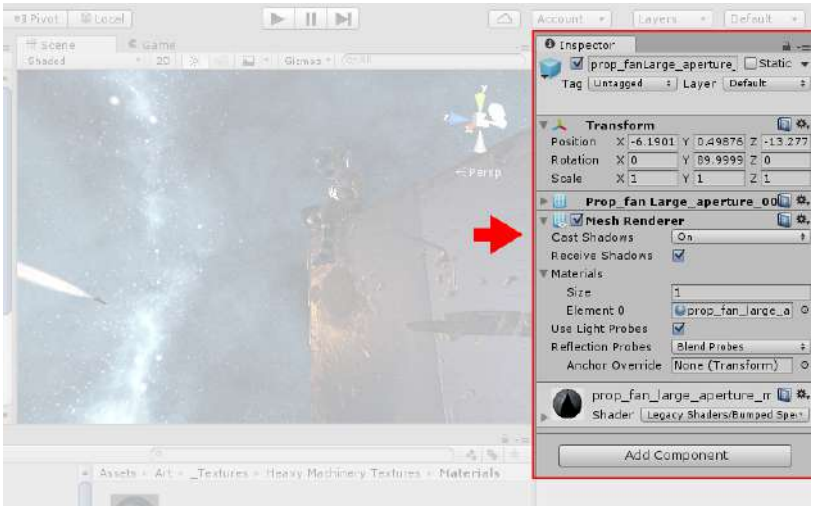


Figura 34: Inspector

- Ventana del proyecto: La ventana del proyecto muestra los assets que están disponibles para que los uses en tu proyecto. Aquí podemos añadir carpetas, imágenes, sonidos, scripts, materiales, modelos, etc. Cuando importamos assets a nuestro proyecto, estos aparecen aquí.

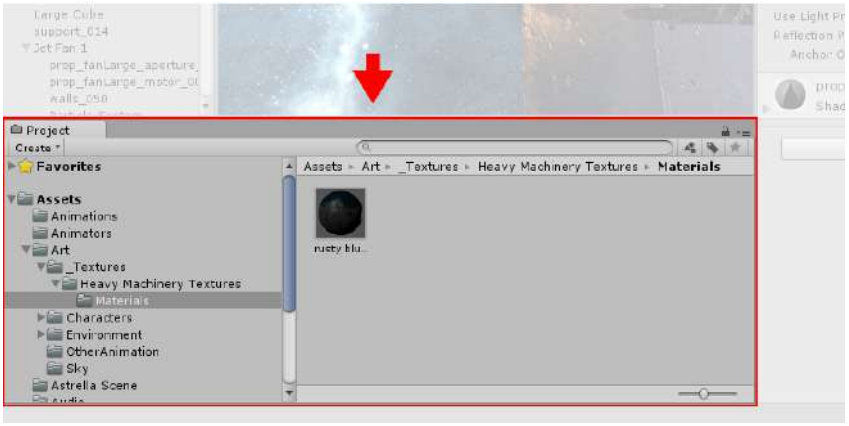


Figura 35: Ventana de proyecto

- En la izquierda encontramos las herramientas básicas para manipular la Vista de la escena y los objetos dentro de esta. En el centro vemos los controles de *play*, *pause* y *step*. Los botones a la derecha nos dan acceso a nuestros servicios de Unity Cloud y nuestra cuenta de Unity, seguido por un menú de visibilidad de capas y finalmente el menú del layout del editor (que proporciona algunos diseños alternativos para la ventana del editor, y nos permite guardar nuestros propios layouts personalizados).

Figura 36: Herramientas

Para codificar los scripts he utilizado el lenguaje C#. Si bien hacía un tiempo que no hacía uso de él, no me ha costado reengancharme por lo que esta parte no ha supuesto gran problema.

Para mantener un orden entre tantísimos elementos del proyecto, los he dividido entre los siguientes tipos:

- ➢ Animations: Guarda las animaciones, los controladores de animación y los *Avatar Mask* de todo el proyecto.
- ➢ Fuentes: Guarda los tipos de fuentes a utilizar para el diálogo.
- ➢ Imágenes: Guarda todas las imágenes a utilizar en el proyecto.
- ➢ Objetos: Guarda todos los objetos 3D del proyecto, menos los humanoides.
- ➢ Packages: Colección de archivos y datos de Unity.
- ➢ Personajes: Guarda los objetos 3D humanoides.
- ➢ SampleScenes: Escenas de ejemplo que aporta Unity.
- ➢ Scenes: Guarda todas las escenas del proyecto.
- ➢ Scripts: Guarda todos los scripts del proyecto.
- ➢ Texturas: Guarda todas las texturas de los objetos del proyecto.

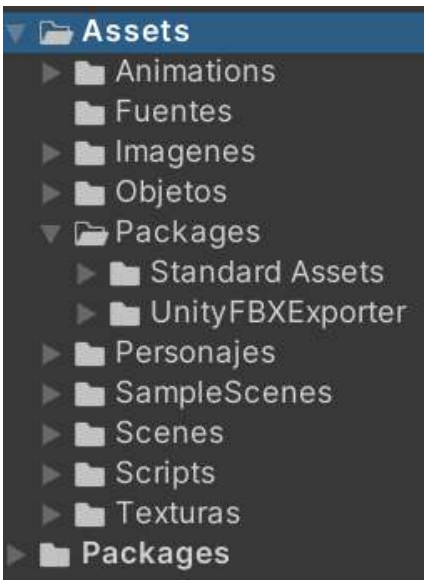


Figura 37: Assets

13.2. Servicios Utilizados

Además de Unity, he utilizado una serie de servicios diferentes, cada uno para una finalidad específica. Estos son:

- *Blender*: es un programa informático multiplataforma, dedicado especialmente al modelado, iluminación, renderizado, animación y creación de gráficos tridimensionales. Se ha utilizado para remodelar todos los objetos 3D de las escenas, de forma que tengan el tamaño, texturas, rotación y demás características correctamente.

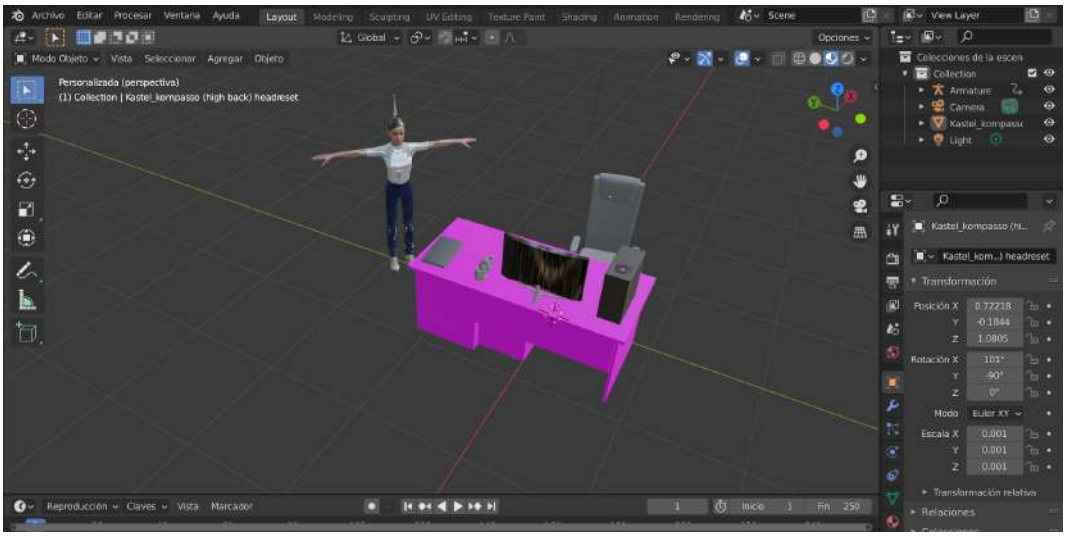


Figura 38: Blender

- *Anima 3D*. aplicación de animación de personas en 3D desarrollada específicamente para arquitectos y diseñadores, ideal para crear increíbles personas animadas en 3D de forma rápida y fácil. Se ha utilizado para descargar a las personas 3D que se utilizan, ya que apenas se pueden encontrar en internet.

Figura 39: Anima

- *Objetos 3D* vía web: Páginas web que ofrecen una gran cantidad de objetos 3D. Se han utilizado sólo aquellos objetos ofrecidos de forma gratuita. Se han utilizado para descargar todos los objetos 3D a excepción de los humanoides. Las principales páginas web utilizadas han sido:

    - Turbosquid
    - CGTrader
    - All3DP

Figura 40: TurboSquid, CGTrader & All3DP

- *Mixamo*: Es una empresa de tecnología de gráficos por computadora en 3D. De aquí he sacado todas las animaciones que he utilizado para el proyecto. También te ofrece un *auto-rigger* para los objetos humanoides que importas, lo que es de mucha utilidad.

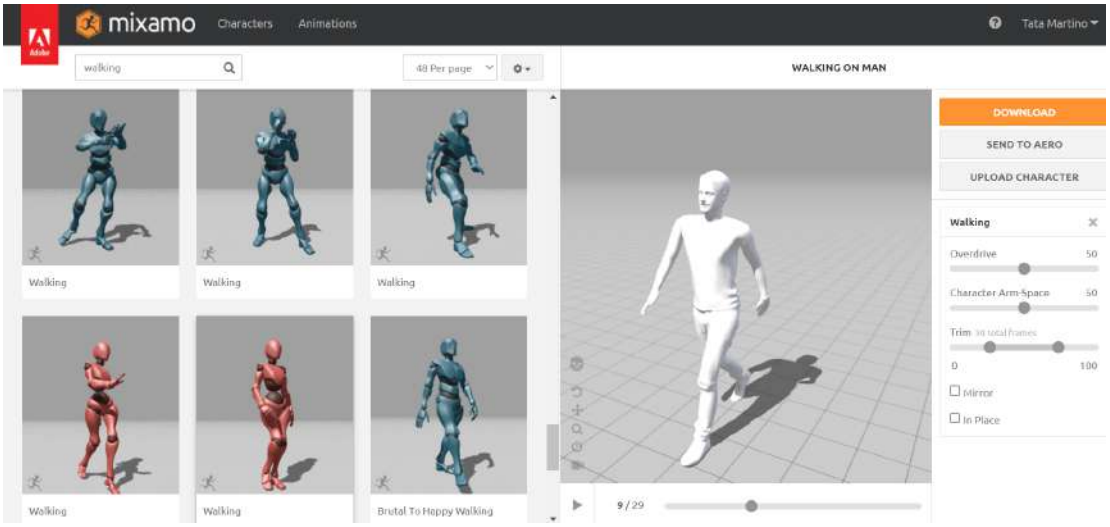


Figura 41: Mixamo

- *Unity Asset Store*: Biblioteca de assets comerciales y gratuitos creados por Unity Technologies y miembros de la comunidad. Hay una gran cantidad de assets disponibles, desde texturas, modelos y animaciones hasta ejemplos de proyectos completos, tutoriales y extensiones del editor. Estos assets son accesibles desde una interfaz simple dentro del editor de Unity y son descargados e importados directamente en sus proyectos.

  He probado varios assets (gratuitos todos) ofrecidos en esta tienda a lo largo del proyecto que me han ayudado a inspirarme en varios aspectos, pero al final solo he utilizado realmente uno del que hablaré más adelante: *Cinemachine*.

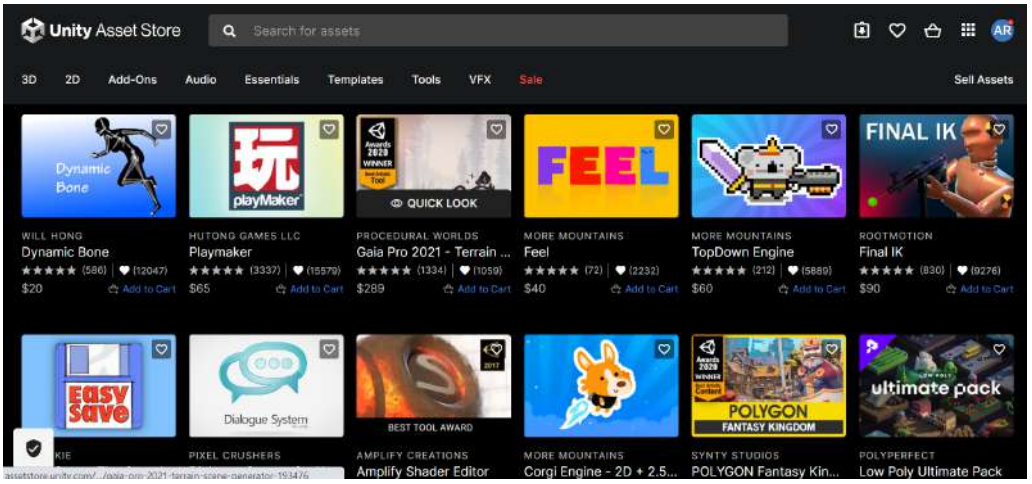


Figura 42: Unity Asset Store

13.1. Elementos utilizados

13.1.1. Canvas

Para la interacción del jugador con el videojuego he utilizado el elemento Canvas.

El Game Object Canvas es un contenedor donde podemos incluir todos los objetos que componen nuestro interfaz de usuario (textos, imágenes, menús, botones…) que aparecerán en pantalla (*Canvas - Unity Manual*, n.d.).

Figura 43: Canvas elementos

Nuestro Canvas consta de los siguientes elementos:

- FlechaIzqText: Texto correspondiente a la opción de la izquierda. Varía en función de la situación en la que estemos.
- FlechaDerText: Texto correspondiente a la opción de la derecha. Varía en función de la situación en la que estemos.
- FlechaIzqImage: Imagen que ilustra la tecla de la flecha izquierda. Sirve para dar a entender al jugador que debe pulsar dicha tecla para seleccionar la opción de la izquierda.

- FlechaDerImage: Imagen que ilustra la tecla de la flecha derecha. Sirve para dar a entender al jugador que debe pulsar dicha tecla para seleccionar la opción de la derecha.
- TextNPC: Texto correspondiente a los diálogos de los NPC. Todo lo que diga un NPC se escribirá aquí, comenzando el texto siempre por el nombre del NPC que pronuncie dicha frase y dos puntos.
- SpaceBarImage: Imagen que ilustra la tecla de la barra espaciadora. Sirve para dar a entender al jugador que debe pulsar dicha tecla para tomar una decisión.
- SpaceBarText: Texto correspondiente a la decisión que ofrece la barra espaciadora. Varía en función de la situación en la que estemos.
- TextoProtagonista: Texto correspondiente a los diálogos del protagonista. Sirve para diferenciar aquello que digan los NPC y lo que diga este.
- Timer: Temporizador. Sirve para representar el tiempo restante que tiene el jugador para tomar decisiones cuando deba. Se reinicia siempre que comience una nueva situación. Para crear este temporizador he hecho uso de dos imágenes: un círculo totalmente vacío y uno completamente verde. Siendo la primera imagen padre de la segunda, he utilizado el elemento *Fill Amount*. Esto nos permite variar la cantidad de la imagen que queremos enseñar.

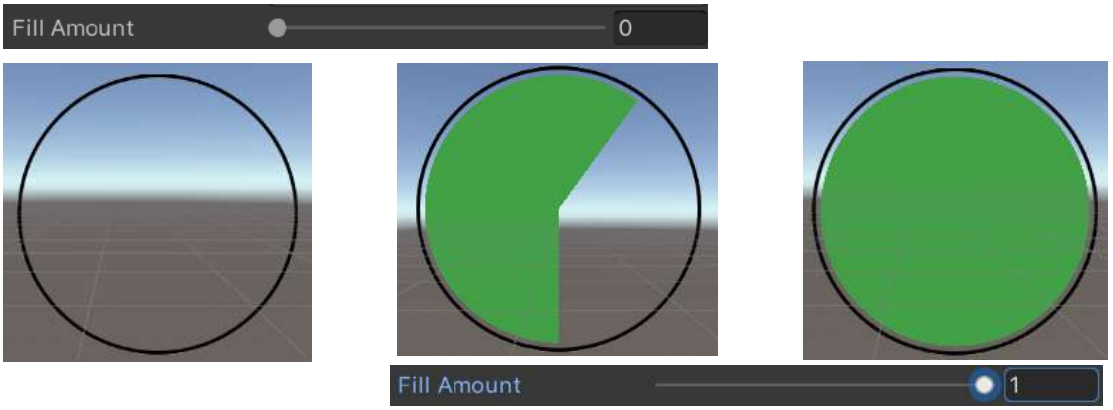


Figura 44: Timer muestra

- Panel: Panel totalmente negro. Si bien en la imagen no se aprecia, sirve para crear animaciones al inicio y al final de cada escena. Estas animaciones se llaman *fade in* y *fade out*. Se trata de oscurecer la pantalla cuando la escena termina y de hacer lo contrario cuando comienza. Para ello jugamos con el elemento *Transparencia* del color del panel, creando una animación FadeIn cambiando este elemento de 1 a 0 y viceversa para FadeOut, siendo 1 totalmente opaco y 0 completamente transparente.

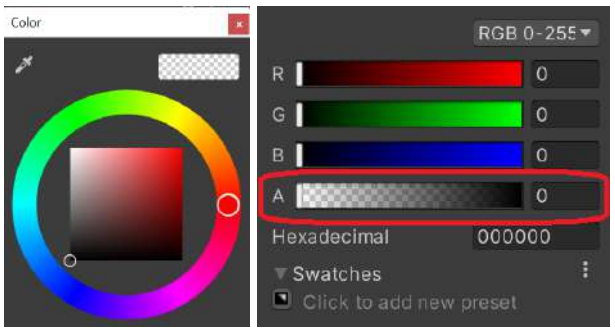


Figura 45: Panel, FadeIn - FadeOut

- Slider: permite al usuario seleccionar un valor numérico de un rango predeterminado arrastrando el ratón. Este elemento lo utilizaremos únicamente en la ResultsScene, para poder puntuar las emociones que se presenten en la pantalla anteriormente mencionadas.

13.1.2. Animator Controller

Un asset Animator Controller nos permite arreglar y mantener un conjunto de animaciones para un personaje u objeto. En la mayoría de situaciones, es normal tener múltiples animaciones y cambiarlas entre sí cuando ocurren ciertas condiciones del juego. De todas formas, incluso si sólo tuviéramos un clip de animación, aún necesitaríamos colocarlo en un animator controller para utilizarlo en un Game Object (*Manual: Animator Controller*, n.d.).

El controller tiene referencias a los clips de animación utilizados dentro y maneja los varios estados de las transiciones entre estos utilizando el *State Machine*, una especie de diagrama de flujo.

Para completar las transiciones entre animaciones utilizaremos los parámetros, variables definidas dentro de un Animator Controller que pueden ser accedidas desde scripts y cuyos valores pueden ser asignados por estos.

Para manejar *State Machine* complejos para diferentes partes del cuerpo Unity utiliza Animation Layers. Estos nos ayudan a activar distintas animaciones al mismo tiempo referenciando a qué parte del cuerpo queremos que afecten dichas animaciones. Para ello utilizamos las *Avatar Mask*, que nos permite descartar algunos datos de la animación dentro de un clip, permitiendo al clip animar solo las partes del objeto o personaje seleccionadas.

A continuación, un ejemplo del Animator Controller utilizado para el protagonista en la ClassScene. Se trata de un controller sencillo ya que no se necesitan muchas animaciones

Protagonista Animator Controller - Class Scene

Utilizamos dos Layers: un Base Layer que se encarga de la mayoría de las animaciones y transiciones y un Talk Layer cuya función consiste únicamente en activar o desactivar la *TalkAnimation* (animación correspondiente a la actividad de hablar). Para el TalkLayer creamos el *Avatar Mask Solo brazos* que referencia a dichas partes del personaje. No es necesario crear uno nuevo para el Base Layer ya que, al ser este el layer principal, automáticamente se queda con las partes no seleccionadas por los *Avatar Mask* existentes.

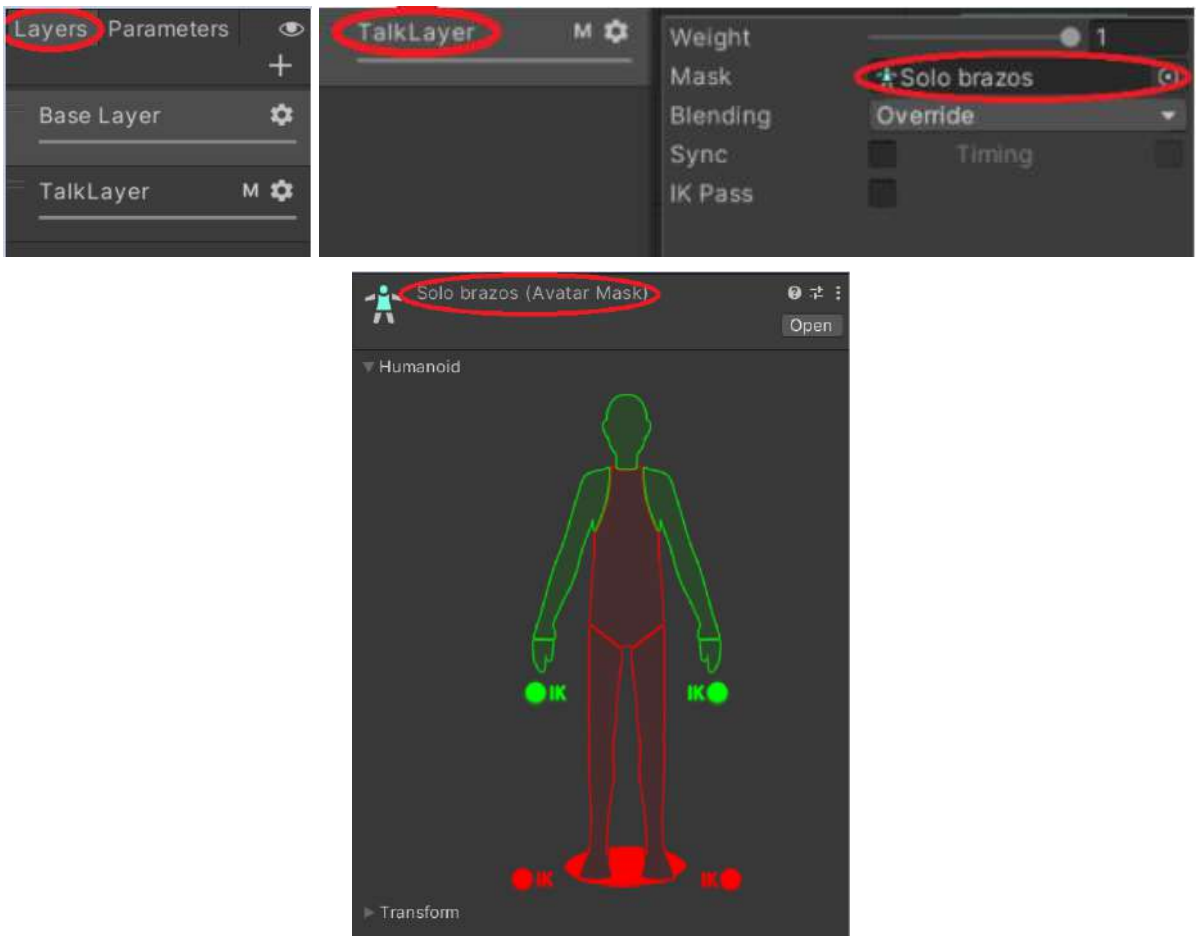


Figura 46: Animator Controller

Base Layer:

Este layer se encarga de la mayor parte del comportamiento del protagonista. Comienza sentado (*Sitting*) para más tarde levantarse de la silla (*StandUp*), seguidamente se queda en una posición inactiva (*Idle*) para, un algún momento, comenzar a andar (*WalkingF*). En caso de dejar de andar volvería a la posición inactiva.

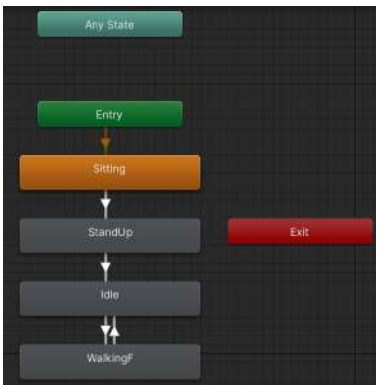
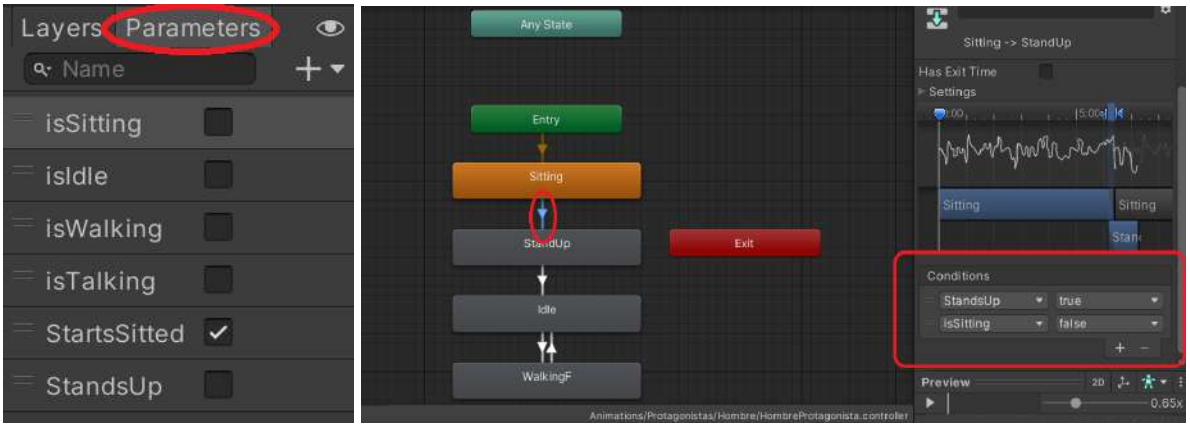


Figura 47: Animator Animations

Para realizar las transiciones debemos activar y desactivar las animaciones correspondientes. Esto lo conseguiremos mediante los parámetros mencionados anteriormente y accediendo a ellos desde el script.

Por ejemplo, como vemos en la siguiente imagen, para la transición *Sitting* → *StandUp* utilizaremos los parámetros booleanos *isSitting* y *StandsUp*, de forma que les daremos el valor *false* y *true* respectivamente desde el script. En la sección *Conditions* estableceremos que son estas las condiciones para realizar la transición.



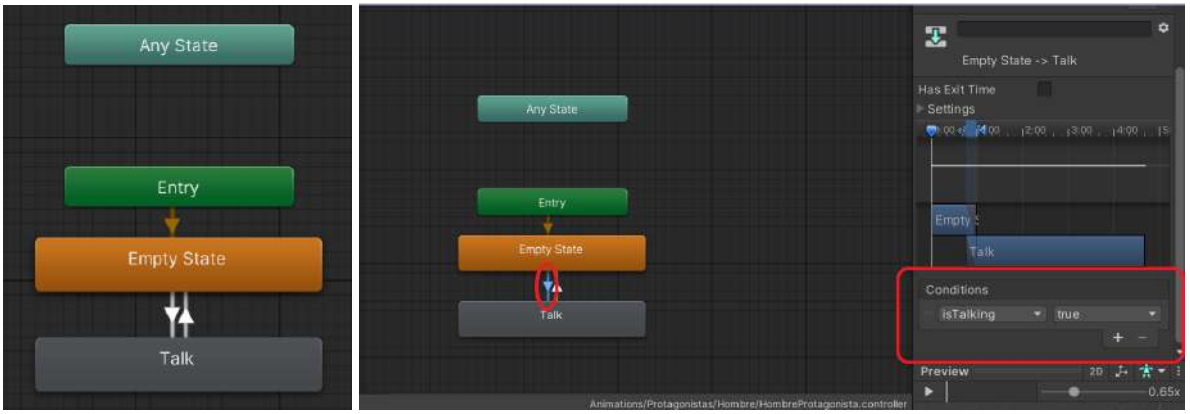

```
animator.SetBool("isSitting", false);
animator.SetBool("StandsUp", true);
```

Figura 48: Animator Parameters & Conditions - Base Layer

Talk Layer:

Este Layer es aún más sencillo. Únicamente consta de dos animaciones: una animación vacía y la correspondiente al acto de hablar *(Talk Animation*). La primera de ellas no tiene ninguna funcionalidad, sirve únicamente para desactivar la segunda animación.

Los parámetros son compartidos con el Layer anterior, pero en este caso solo necesitamos uno ya que únicamente realizamos una transición (bidireccional): *isTalking*. Repetimos el proceso: establecemos las condiciones y desde el script activamos o desactivamos las animaciones.




```
animator.SetBool("isTalking", true);
```

Figura 49: Animator Parameters & Conditions - Talk Layer

13.1.3. WayPoints

Durante el transcurso de la ClassScene, varios personajes deben moverse de un lugar a otro constantemente. Para definir los caminos que deben seguir utilizamos los WayPoints. Estos no son más que Game Objects vacíos que se encargan de definir el camino que recorrerá cada uno de los personajes en la escena (de Byl, 2021).

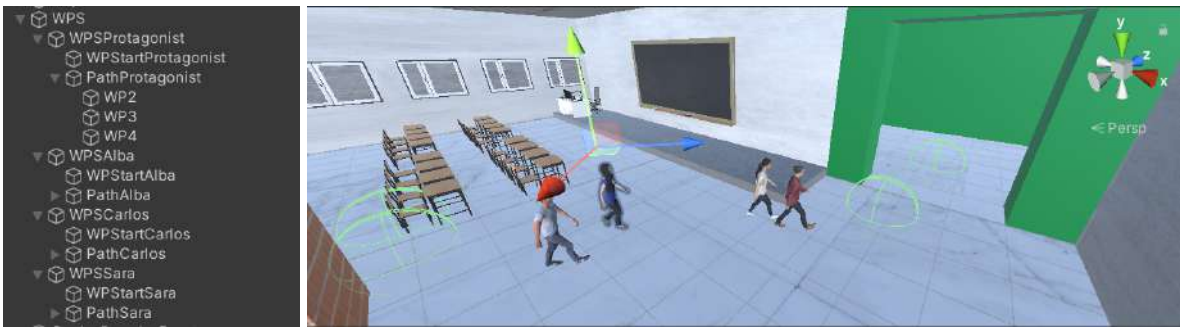


Figura 50: WayPoints

Cada personaje tiene su propio camino (*path*) por lo que cada uno tiene un sistema de WayPoints personalizado.

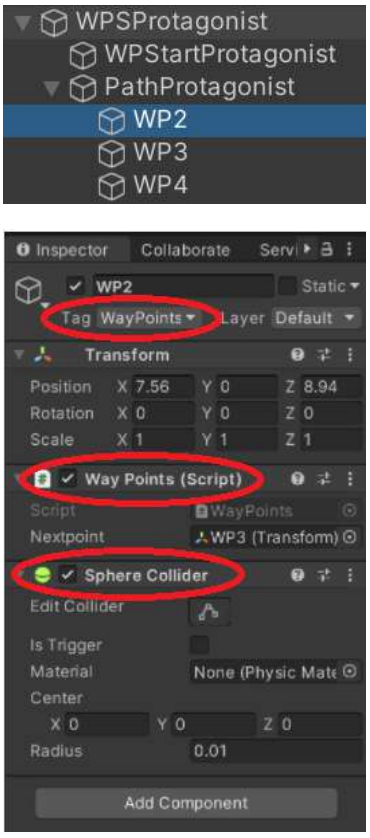


Figura 51: WayPoints Implementation

Para que este proceso funcione correctamente, cada WayPoint debe tener los siguientes componentes:

- Tag: “WayPoints”:

Utilizo esta etiqueta para saber con certeza que el personaje estará colisionando con un WayPoint y no un Game Object distinto.

- WayPoints script:

Este script es extremadamente sencillo. Únicamente cuenta con una variable pública para poder seleccionarla luego. Esta variable debe hacer referencia al WayPoint siguiente.

```
public class WayPoints : MonoBehaviour
{
    public Transform nextpoint;
}
```

Figura 52: WayPoints Script

- Sphere Collider:

Los colliders se usan para que sea posible detectar colisiones entre distintos objetos del juego. En otras palabras, es una estructura que nos permite establecer o delimitar una zona de contacto, esta zona de contacto nos permite saber si un objeto entró en contacto con otro o si acaba de dejar de estar en contacto con determinado objeto.

Yo utilizaré un *Sphere Collider*, una primitiva de colisión básica con la forma de una esfera. Si bien en la Figura 50 parece que tienen un diámetro exagerado, esto se debe a que aumenté el radio a 0.75 para representar correctamente la ilustración. En el videojuego el radio de cada una de las *Sphere Collider* es el mínimo posible, 0.01.

Además, cada personaje debe tener como componente el siguiente script:

Figura 53: WayPoints in Behaviour

- Behaviour

Este script contiene todo el comportamiento del personaje a lo largo de la escena. Con respecto a los WayPoints, lo importante son las variables públicas *Target Game Object* y *Transf*. La primera de ellas se encarga de iniciar el primer *target* de la secuencia de WayPoints y la segunda coloca al personaje en el WayPoint inicial después de levantarse de la silla.

Una vez quiera que comience el recorrido entre los WayPoints, esto se realizará mediante las siguientes líneas de código:

```
transform.lookAt(new Vector3(target.position.x, transform.position.y, target.position.z));
```

Figura 54: WayPoints LookAt

Recorreremos esta línea de código al inicio de la escena una única vez. Pondremos al personaje a mirar en dirección al *target* para posteriormente recorrer la siguiente función:

```
transform.Translate(new Vector3(0, 0, speed * Time.deltaTime));
```

Figura 55: WayPoints Translate

Recorreremos esta línea cada frame, para mover al personaje en la misma dirección que aquella a la que mira.

Por último, utilizaremos la siguiente función:

```
private void OnCollisionEnter(Collision collision)
{
    if (collision.gameObject.tag == "WayPoints")
    {
            target = collision.gameObject.GetComponent<WayPoints>().nextpoint;
            transform.LookAt(new Vector3(target.position.x, transform.position.y, target.position.z));
    }
}
```

Figura 56: WayPoints onCollisionEnter

La función OnCollisionEnter se llama cuando el colisionador del Game Object al que se adjunta el script ha comenzado a tocar otro colisionador. De esta forma, cada vez que se cree una colisión entre personaje y WayPoint, el *target* cambiará al siguiente WayPoint y el personaje mirará en dirección a este para llamar de nuevo a la línea de código de la Figura 55.

13.1.4. Cinemachine

Cinemachine es una suite de herramientas para cámaras dinámicas, inteligentes y sin código que permite que las mejores tomas emerjan según una composición e interacción de

escena, lo que permite afinar, modificar, experimentar y crear comportamientos de cámara en tiempo real (*Cinemachine*, n.d.).

He hecho uso de este *asset* en la BarScene para mostrar la escena en totalidad.

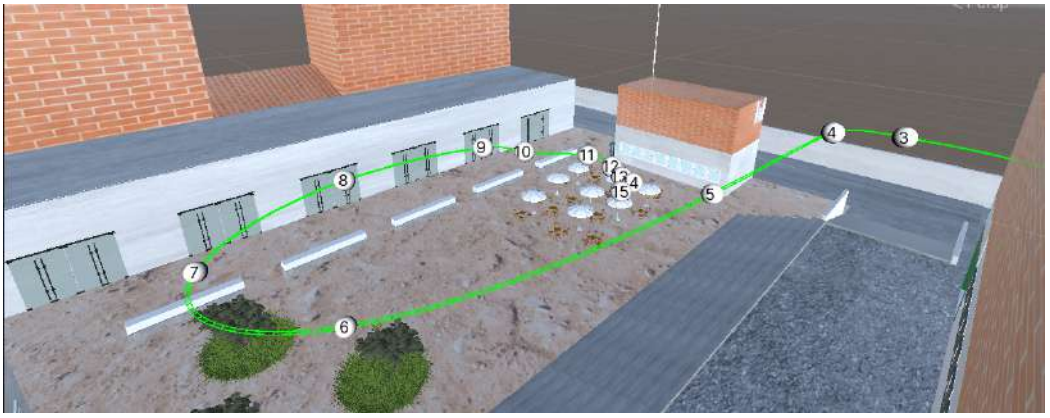


Figura 57: Cinemachine Path

Cinemachine te ofrece varias posibilidades, pero yo solo utilizaré *Create Dolly Camera with Track*. Se crearán entonces dos nuevos componentes, una nueva cámara virtual y un *DollyTrack*. Este último es el elemento que se utiliza para establecer el camino que seguirá la nueva cámara, como se puede ver en la Figura 57. Conseguiremos esto utilizando el componente *WayPoints* dentro del script *CinemachineSmoothPath* que encontramos en el *DollyTrack*.

Para terminar, deberemos crear un *Animation Track* para la nueva cámara, editando el *Path Position* (Waypoints del *DollyTrack*) y su rotación.

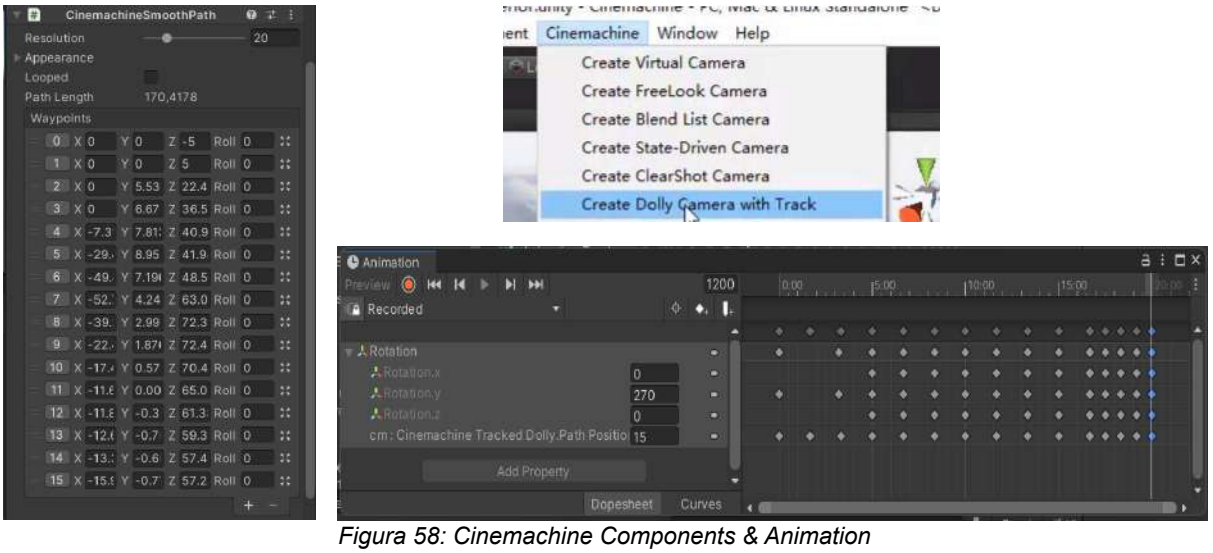


Figura 58: Cinemachine Components & Animation

Gracias a esta herramienta, podemos crear una cinemática mediante el movimiento de la cámara siguiendo los *WayPoints* establecidos.

Figura 59: Cinemachine muestra

13.1.5. Coroutines

Las Coroutines son una forma especial de hacer que la lógica suceda con el tiempo. Cuando llamamos a una función, esta se ejecuta en su totalidad antes de volver. Esto significa que cualquier acción en una función debe suceder en una sola actualización de frame.

Por ejemplo, si queremos una función que reduzca gradualmente el valor alfa de un objeto (opacidad) hasta que se convierta completamente invisible:

```
void Fade()
{
    for (float f = 1f; f >= 0; f -= 0.1f)
    {
        Color c = renderer.material.color;
        c.a = f;
        renderer.material.color = c;
    }
}
```

Figura 60: Coroutine example

Esta función no tendrá el efecto que buscamos. Para que el desvanecimiento sea visible, el valor alfa debe ser reducido durante una secuencia de frames para mostrar los valores intermedios siendo renderizados. Sin embargo, la función se ejecutará en su totalidad en una sola actualización de frame. Nunca veríamos los valores intermedios y el objeto desaparecería instantáneamente.

Para conseguir el efecto que queremos utilizaremos las *Coroutines*. Estas son una función que tienen la capacidad de pausar su ejecución y devolver el control a Unity para continuar su trabajo en el siguiente frame. Se declararía de la siguiente forma.

```
IEnumerator Fade()
{
    for (float f = 1f; f >= 0; f -= 0.1f)
    {
        Color c = renderer.material.color;
        c.a = f;
        renderer.material.color = c;
        yield return null;
    }
}
```

Figura 61: Coroutine example

Es esencialmente una función declarada con un tipo de retorno IEnumerator y con una instrucción de retorno yield incluida en algún lugar de su cuerpo. La línea con la instrucción de retorno yield es el punto en el cual la ejecución pausará y se reanudará en el siguiente frame. Para establecer la ejecución de la *Coroutine* necesitaremos usar la función *StartCoroutine*.

```
void Update()
{
    if (Input.GetKeyDown("f"))
    {
        StartCoroutine("Fade");
    }
}
```

Figura 62: Coroutine example

Además, el uso de las *Coroutines* nos ofrece una serie de posibilidades. En este proyecto he hecho bastante uso de dos de ellas:

- *WaitForSeconds*: Suspende la ejecución de la *Coroutine* durante la cantidad dada de segundos utilizando un tiempo escalado.
- *WaitUntil*: Suspende la ejecución de la *Coroutine* hasta que la condición proporcionada se evalúe como verdadera.

```
using UnityEngine;
using System.Collections;

public class WaitForSecondsExample : MonoBehaviour
{
    void Start()
    {
        //Start the coroutine we define below named ExampleCoroutine.
        StartCoroutine(ExampleCoroutine());
    }

    IEnumerator ExampleCoroutine()
    {
        //Print the time of when the function is first called.
        Debug.Log("Started Coroutine at timestamp : " + Time.time);

        //yield on a new YieldInstruction that waits for 5 seconds.
        yield return new WaitForSeconds(5);

        //After we have waited 5 seconds print the time again.
        Debug.Log("Finished Coroutine at timestamp : " + Time.time);
    }
}
```

```
using UnityEngine;
using System.Collections;

public class WaitUntilExample : MonoBehaviour
{
    public int frame;

    void Start()
    {
        StartCoroutine(Example());
    }

    IEnumerator Example()
    {
        Debug.Log("Waiting for princess to be rescued...");
        yield return new WaitUntil(() => frame >= 10);
        Debug.Log("Princess was rescued!");
    }

    void Update()
    {
        if (frame <= 10)
        {
            Debug.Log("Frame: " + frame);
            frame++;
        }
    }
}
```

Figura 63: WaitForSeconds & WaitUntil

13.2. Componentes de los Personajes

13.2.1. Protagonista

Los protagonistas utilizan los siguientes componentes:

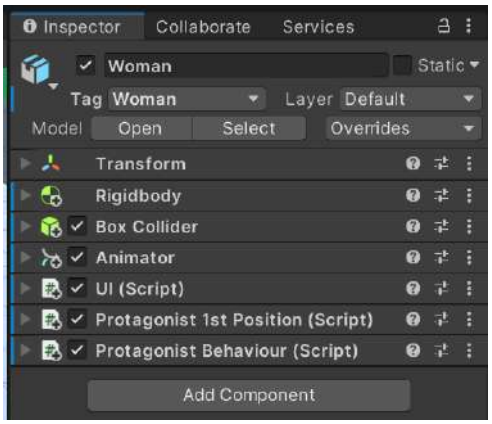


Figura 64: Protagonista Componentes

- Transform: Define la posición, orientación y escala del protagonista.

- Rigidbody: componente principal que permite el comportamiento físico del protagonista. Con este componente adjunto, el protagonista inmediatamente responderá a la gravedad. También reacciona ante colisiones.

- Box Collider: Primitiva básica de colisión en forma de cubo. La forma de este es editable.

- Animator: Define el *Animator Controller* que utilizará el protagonista.

- UI Script: Script creado que interactúa con el elemento Canvas de la escena.

- Protagonist 1st Position Script: Script creado que sitúa al protagonista en la posición que deseemos al comienzo de la escena, dependiendo de sus variables públicas.

- Protagonist Behaviour Script: Script creado que define el comportamiento global del protagonista. Destaca el uso de las *Coroutines*.

13.2.2. NPC

Los NPC utilizan los siguientes componentes:

Figura 65: NPC Componentes

- Transform: Define la posición, orientación y escala del NPC.

- Rigidbody: componente principal que permite el comportamiento físico del NPC. Con este componente adjunto, el NPC inmediatamente responderá a la gravedad. También reacciona ante colisiones.

- Box Collider: Primitiva básica de colisión en forma de cubo. La forma de este es editable.

- Animator: Define el *Animator Controller* que utilizará el NPC.

- Behaviour Script: Script creado que define el comportamiento global del NPC. Destaca el uso de las *Coroutines*.

- First Position Script: Script creado que sitúa al NPC en la posición que deseemos al comienzo de la escena, dependiendo de sus variables públicas.

13.3. Producción

Unity cuenta con una ventana que nos permite escoger la plataforma de destino, definir los ajustes y empezar el proceso de construcción (*Manual: Build Settings*, n.d.).

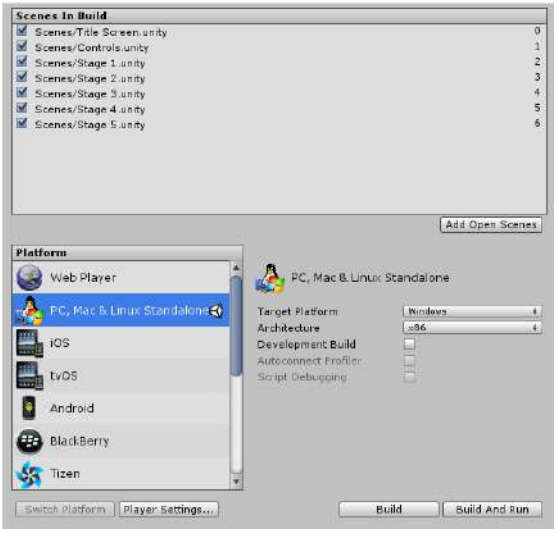


Figura 66: Unity Plataformas

Esta es la lista de todas las plataformas que están disponibles para descargar el proyecto:

- Web Player (Reproductor Web)
- PC, Mac & Linux Standalone
- iOS
- Android
- Tizen
- WebGL (Pre-visualización)
- Samsung TV
- Windows Store
- Windows Phone 8
- Específicas de la plataforma (otras)

Nosotros haremos uso de la segunda opción, concretamente de la versión para PC. Esta ofrece gran variedad de posibilidades:

Opción	Propósito
Target Platform (Plataforma destino)	
Windows	Construcción para Windows
Mac OS X	Construcción para Mac
Linux	Construcción para Linux
Architecture (Arquitectura)	x86
x86	32-bit CPU
x86_64	64-bit CPU
Universal	Todos los dispositivos CPU
x86 + x86_64 (Universal)	Todos los dispositivos CPU para Linux
Headless Mode	Construye juego para uso del servidor sin ningún elemento visual

Figura 67: Unity plataformas y propósitos

Para la demostración del videojuego, yo utilizaré la opción para Windows, pero siempre que lo queramos en otro formato Unity nos ofrece esta posibilidad de forma extremadamente sencilla.

14. Pruebas

Para poder garantizar la correcta realización del trabajo es necesario elaborar un sistema de pruebas. En este caso no se me ocurría uno más adecuado que seguir los siguientes dos puntos:

- Comprobar que todo funciona correctamente a medida que voy añadiendo elementos nuevos al proyecto. Unity cuenta con un sencillo botón de *play* que pone el videojuego en marcha. Siempre que añado un cambio al proyecto, inicio el videojuego un mínimo de 3 veces para comprobar que todo funciona como deseo. De esta forma me aseguro constantemente de que la elaboración de la escena se está realizando correctamente.

- Pedir a terceras personas que prueben el videojuego cada vez que añado un cambio considerable de forma que, en caso de haber un error o una posible mejoría, pueda solucionarlo. El número mínimo de personas debe ser tres para poder recoger un feedback que cumpla con cierto grado de fiabilidad. Por ejemplo, al unificar todas las escenas, pedí a compañeros que comprobaran su funcionamiento. Fueron las mujeres las que comprobaron que, escogiendo a la mujer como avatar, el sistema daba con diversos errores: la posición inicial no era la correcta, el orden del diálogo se descomponía, también las animaciones, y algunos más. También lo pedí siempre que terminaba alguna escena, para que probaran todas sus funcionalidades. En la mayoría de veces he tenido algo que corregir y/o mejorar, ya sean errores significativos como el que he mencionado o detalles más pequeños que ajustar.

15. Conclusiones

Objetivo: El objetivo de este proyecto era elaborar un videojuego para el diagnóstico de fobia social para aquellas personas que puedan padecerla. Podríamos decir que hemos conseguido este objetivo, creando dicho videojuego de la mano de una psicóloga experta con un informe de resultados final y mediante la elaboración de una serie de escenas basadas en los criterios diagnósticos de la fobia social. Todo ello mediante la herramienta Unity.

Competencias:

- CES1.3: Identificar, evaluar y gestionar los riesgos potenciales asociados a la construcción de software que pudiesen presentarse: Como riesgos detectados en este proyecto encontramos la posibilidad de bugs y la mala gestión de tiempo. Para ello he planteado una serie de soluciones que he seguido durante el transcurso del trabajo.

- CES1.7: Controlar la calidad y diseñar pruebas en la producción de software: Para garantizar que el programa funcione correctamente, hemos planificado una serie de pruebas a realizar, entre las que encontramos la comprobación constante de que el videojuego funcione correctamente cada vez que se añada un elemento nuevo y la opinión de una tercera persona después cada gran cambio.

- CES2.1: Definir y gestionar los requisitos de un sistema software: Para ello he evaluado los requisitos funcionales y no funcionales de mi proyecto.

- CES3.1: Desarrollar servicios y aplicaciones multimedia: Considero que, al tratarse este proyecto de la elaboración de un videojuego, ya se logra esta competencia.

Posibles mejoras: La finalidad de este proyecto no es otra que aportar un diagnóstico que ayude a orientar a aquellas que crean que pueden padecer de fobia social. Obviamente, cuanto más completo sea dicho diagnóstico, mejor. ¿Qué posibles mejoras podríamos hacer en caso de disponer de más tiempo? Entre otras:

- Mayor número de escenas: Disponer de más escenas significa representar un mayor número de situaciones, y cuántos más datos tengamos de estas más completo será nuestro resultado.

- Mejor diseño: Uno de los objetivos es aportar realismo al videojuego, de forma que el jugador se sienta sumergido en este. Probablemente, con la ayuda de un experto en diseño, o mejorando yo mis conocimientos sobre el tema, podríamos mejorar este aspecto

- Objetos más realistas: Los objetos 3D utilizados en este trabajo son totalmente gratuitos. Si dispusiéramos de fondos para objetos no gratuitos el videojuego podría tener una mejor estética más realista.

- Otros entornos: Como he dicho anteriormente, un objetivo es conseguir que el jugador se sienta dentro del videojuego. Al tener el tiempo reducido, para conseguirlo me he enfocado en posibles situaciones de un estudiante de universidad, concretamente de uno de la Facultad de Informática de Barcelona. Si dispusiéramos de más tiempo, se podrían estudiar nuevos entornos para lograr dicho fin.

Estas son varias posibles mejoras, pero seguramente existan muchas otras. En conclusión, cuanta mayor dedicación, más completo será el proyecto.

16. Referencias

Aprende C# con Unity - Corrutinas· Antonio Moon´s. (n.d.). Antonio Moon´s. Retrieved January 17, 2022, from https://moonantonio.github.io/post/2018/csharpunity/007/

Asensio, I. (n.d.). *Unity vs Unreal ¿Qué es mejor para crear videojuegos?* SoloEmpleo. Retrieved January 17, 2022, from https://www.soloempleo.com/diferencias-entre-unity-y-unreal-engine

Asensio, I. (2019, November 8). *Qué es Unity y para qué sirve*. MasterD. Retrieved January 17, 2022, from https://www.masterd.es/blog/que-es-unity-3d-tutorial

Canvas - Unity Manual. (n.d.). Unity - Manual. Retrieved January 17, 2022, from https://docs.unity3d.com/es/2019.4/Manual/UICanvas.html

Cinemachine. (n.d.). Unity. Retrieved January 17, 2022, from https://unity.com/es/unity/features/editor/art-and-design/cinemachine

de Byl, P. (2021, March 5). *Waypoints & Graphs*. Unity Learn. Retrieved January 17, 2022, from https://learn.unity.com/project/waypoints-graphs

Fobias. (2021, October 28). MedlinePlus. Retrieved January 17, 2022, from https://medlineplus.gov/spanish/phobias.html

La influencia de la tecnologia en nuestra vida cotidiana. (n.d.). Universidad Anáhuac. Retrieved January 17, 2022, from https://www.anahuac.mx/generacion-anahuac/la-influencia-de-la-tecnologia-en-nuestra-vida-cotidiana

"Los videojuegos son una herramienta de aprendizaje efectiva". (2015, April 20). Agencia SINC. Retrieved January 17, 2022, from https://www.agenciasinc.es/Entrevistas/Los-videojuegos-son-una-herramienta-de-aprendizaje-efectiva

Manual: Animator Controller. (n.d.). Unity - Manual. Retrieved January 17, 2022, from https://docs.unity3d.com/es/530/Manual/class-AnimatorController.html

Manual: Aprendiendo la Interfaz. (n.d.). Unity - Manual. Retrieved January 17, 2022, from https://docs.unity3d.com/es/530/Manual/LearningtheInterface.html

Manual: Build Settings. (n.d.). Unity - Manual. Retrieved January 17, 2022, from https://docs.unity.cn/2018.1/Documentation/Manual/BuildSettings.html

MVC simplificado.· Antonio Moon´s. (n.d.). Antonio Moon´s. Retrieved January 17, 2022, from https://moonantonio.github.io/post/2017/dev/014/

Oliva, V. (2019, December 26). *¿Cómo influye la tecnología en las empresas? - Admisión UTEM*. Admisión UTEM. Retrieved January 17, 2022, from https://admision.utem.cl/2019/12/26/como-influye-la-tecnologia-en-las-empresas/

Pursell, S., & Rohn, J. (2021, December 1). *Metodología Agile: qué es y cómo aplicarla a tu proyecto*. Blog de HubSpot. Retrieved January 17, 2022, from https://blog.hubspot.es/marketing/metodologia-agile

Sueldos de la empresa. (n.d.). Glassdoor. Retrieved January 17, 2022, from https://www.glassdoor.es/Sueldos/index.htm

17. Anexo A

Entrevista con la psicóloga:

Antes de la entrevista que concerté con la psicóloga, yo no tenía muy claro hacia donde iba a enfocar mi trabajo. En un principio mi intención era desarrollar un videojuego que funcionara como terapia para cierta fobia, pero sin saber aún en cuál podría enfocarme y cómo podría hacerlo exactamente.

Durante la reunión me explicó que este primer objetivo era demasiado complicado, por no decir imposible, ya que el tratamiento de una fobia depende en su gran mayoría por el origen de estas, y este varía para cada persona que la padece. Es decir, los tratamientos son mayoritariamente personalizados, por lo que no existiría nada que se pudiera plasmar en un videojuego y que garantizara que una gran cantidad de pacientes pudieran hacer uso de él.

Comenzó entonces a explicarme un poco más en profundidad lo que era una fobia, para ayudarme a comprender mejor el tema que intento tratar. Me explicó que esa emoción de miedo intenso, así como muchas otras, que surge en los pacientes está causada por un sinfín de sentimientos más concretos.

Así que después de meditar un rato qué clase de proyecto podría realizar, decidimos dar un paso atrás y cambiar el enfoque principal. Optamos por no verlo como una especie de tratamiento sino como un diagnóstico. Por tanto, cambiaríamos el público objetivo a aquellas personas que crean que pueda padecer de fobia social, no ya a los pacientes. El objetivo pasaría a ser ayudar a estas personas con una especie de diagnóstico que pueda orientarlas y meditar, entre otras cosas, si quizás necesitan ayuda profesional. Para ello intentaríamos traducir esos sentimientos que componen la emoción del miedo en situaciones dentro del videojuego para ver cómo reaccionan los jugadores.

Por último, me ayudó a crear las pautas para el informe de resultados dependiendo de los comportamientos de los pacientes en el videojuego. Sin ser, obviamente, un diagnóstico oficial, al venir de la mano de una experta sí que le otorga cierta fiabilidad.

ALEGRÍA	MIEDO	TRISTEZA
Agradecimiento	A la defensiva	Abatido
Alegría	Acomplejado	Afligido
Alivio	Acorralado	Agotado
Animado	Agobiado	Apagado
Apacible	Alarmado	Apenado
Buen humor	Amenazado	Avergonzado
Calmado	Angustiado	Conmovido
Contento	Ansioso	Debilitado
Deleite	Aprisionado	Decepcionado
Despreocupado	Aterrorizado	Deprimido
Dicha	Desesperado	Derrumbado
Entusiasmo	Desorientado	Desanimado
Esperanza	En pánico	Desdichado
Estímulo	Incrédulo	Desesperanzado
Euforia	Indeciso	Desgarrado
Fascinación	Inquieto	Desgraciado
Felicidad	Inseguro	Desmoralizado
Gozo	Intranquilo	Frustrado
Ilusión	Nervioso	Impotente
Orgullo	Paralizado	Infeliz
Reconocido	Perplejo	Melancólico
Satisfecho	Preocupado	Nostálgico
Triunfo	Sorprendido	Soledad/solo
Vital	Tenso	Triste

Figura 68: Emociones