Principado de Asturias Consejería de Educación	IES N1 Gijón	PROYECTO DAM	Cofinanciado por la Unión Europea

INSTITUTO DE EDUCACIÓN SECUNDARIA NÚMERO 1 DE GIJÓN
FAMILIA PROFESIONAL DE INFORMÁTICA Y COMUNICACIONES

Ciclo Formativo Desarrollo de Aplicaciones Multiplataforma
2º Curso

Proyecto de Desarrollo de Aplicaciones Multiplataforma

Modalidad Distancia
Tipo: Desarrollo de aplicación. Modalidad Videojuegos

Alegoría de la Depresión

Un Viaje Introspectivo

Documento-Memoria

Autor/a: Mario Galán Pedrayes
Fecha: 26/11/2024

Índice

1 Introducción

1.1 Presentación del alumno

Este proyecto de desarrollo ha sido creado por Mario Galán Pedrayes para el Ciclo Formativo de Grado Superior Desarrollo de Aplicaciones Multiplataforma cursado en el I.E.S. Nº 1 en el año 2023/2024 siendo el tutor Don Jose Antonio Priego Pil.

1.2 Título del proyecto y tipo de proyecto elegido

El proyecto se titula “Alegoría de la depresión” con el subtítulo “Un viaje introspectivo”. El proyecto consiste en un videojuego 2D (2 dimensiones) en el que el usuario podrá experimentar una pequeñísima parte de lo que sufre una persona con depresión, siendo perseguida por problemas y pensamientos negativos e intrusivos que solo podrá superar mediante la ayuda de sus amigos, fármacos, familiares u otro tipo de apoyo.

Se desarrollará en el framework Unity mediante el lenguaje de programación C#, es un videojuego arcade que consiste en partidas rápidas donde deberemos ir pasando niveles mientras somos perseguidos por enemigos a los que debemos esquivar si no queremos perder puntuación, la cual podemos obtener mediante PowerUps o eliminando a estos mismos enemigos.

El videojuego cuenta con un ranking que guardará las 5 mejores puntaciones, permitiendo exportar e importar estas en JSON por si se quiere compartir con amigos.

2 Definición del proyecto

2.1 Descripción general

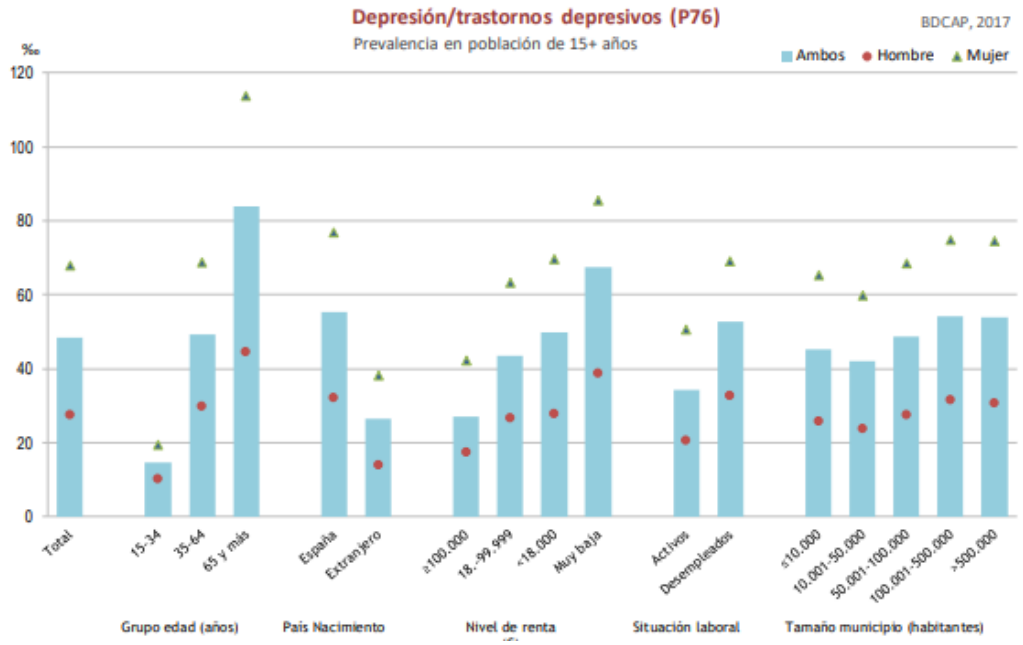
El proyecto consiste en un videojuego donde el objetivo es pasar una serie de niveles esquivando (o atacando mediante mejoras llamadas “PowerUps”) a elementos hostiles. La

historia pretende ser una alegoría de los obstáculos por los que pasa una persona con depresión, teniendo que sortear o enfrentar obstáculos diarios para pasar un día más.

En cuanto a la trama del videojuego se intercalarán entre los niveles espacios de texto, explicando o exponiendo la trama del videojuego buscando una mejor comprensión de la situación y la metáfora de este.

El videojuego se realizará en Unity, usando assets gratuitos de la biblioteca de este o creando los sprites basándose en otros como ejemplo y utilizando lo aprendido durante el curso en las diferentes asignaturas, usando lo aprendido, por ejemplo en desarrollo de Interfaces (para el correcto y buen desarrollo de la interfaz del videojuego, buscando una simpleza de uso necesaria para que un usuario poco experimentado en el uso del ordenador pueda jugar), programación multimedia y dispositivos móviles (Uso de la tecnología Unity y la programación del juego), programación de servicios y procesos (para la necesaria optimización y uso de procesos simultáneos), Acceso a datos para el guardado de datos en JSON y serializados, etc.

El videojuego va dirigido para un público adulto, siendo los clientes potenciales aquellas personas que quieran comprender un poco del gran problema que supone la depresión en la vida diaria de las personas que la sufren. Según el ministerio de sanidad una gran parte de la población está afectada por esta enfermedad metal que cada vez es más frecuente.



Ministerio de Sanidad | Subdirección General de Información Sanitaria

Además, para hacer el videojuego más atrayente, tiene un componente de competición en el que los usuarios que lo jueguen pueden luchar por llegar a ser los primeros en puntuación, teniendo un concepto de base arcade, permitiendo juntar una mecánica tan típica como efectiva (solo hay que ver como después de tantos años este tipo de juegos siguen triunfando) junto con un tema de actualidad y una estética oscura.

2.2 Definición de requisitos

- Diseño e implementación del modelo de datos: Los datos del usuario se guardarán en ficheros de bytes serializados, guardando el nombre del usuario introducido al empezar una nueva partida (con su correspondiente control para evitarlo vacío) y su puntuación, permitiendo que existan datos coincidentes en nombres y puntuación. Los datos solo se guardarán al acabar la partida como ocurre en los juegos arcade.

Los datos se guardarán en un array de 5 (permitiendo de esta forma guardar solo 5 posiciones) que contendrán 5 objetos de una clase con los atributos nombre y puntuación. El ranking mostrará los 5 mejores jugadores y sus puntuaciones ordenadas.

- Inserción y modificación de datos: En caso de que un usuario mejore la puntuación de otro al finalizar el juego, este tendrá la opción de guardar sus datos, lo que permitirá modificar los datos ya existentes. Para ello haremos uso de un array auxiliar que nos permitirá modificar las posiciones ya existentes, eliminando la más baja del array y colocando la nueva en la posición indicada.

- Exportación e importación de datos: El ranking se podrá exportar e importar en JSON permitiendo compartir datos entre usuarios en distintos equipos. El fichero resultando será un fichero con extensión “.json” que contendrá el array de objetos en texto plano, indicando el nombre y la puntuación.

- Visualización de datos: Los datos de puntuación y nombre se guardarán en un objeto que colocaremos en el menú e indicaremos mediante script que no se destruya al pasar de escena (si no Unity los destruye por defecto). Al pulsar la opción de juego nuevo el script cogerá el nombre introducido por el usuario y reiniciará la puntuación.

Para la visualización de los datos se creará un objeto nuevo llamado “Puntuación” que ira sumando los puntos obtenidos en ese nivel y cuando el jugador llegue al objetivo se los pase a los datos de puntuación del usuario, sumándolos para ir acumulando los puntos de

cada nivel. Los datos de puntuación y tiempo gastado en el nivel se mostrarán a través de un canvas con sus determinados scripts que irán actualizando los textos del canvas.

- Control del ranking: El ranking se controlará mediante el uso de dos arrays, un que es el que mostrará y se serializará para guardar los datos y otro auxiliar que usaremos en caso de que sea necesario actualizar y modificarlo porque un usuario a superado la puntuación de otro.

El array resultante tras comprobar si la puntuación se debe guardar se serializará y guardará en el directorio suministrado por Unity para estos casos mediante el método “application.persistentdatapath”, que da un directorio donde guardar los datos. Estos datos tendrán la extensión “.datos”.

- Escena tutorial. Aprendizaje y familiarización con los controles: Realizaremos una escena que contendrá un nivel de prueba en el que el usuario no sumará puntuación y podrá comprobar el funcionamiento del personaje, de los PowerUps y de los enemigos, así como el objetivo de llegar a la meta para acabar el nivel. Para evitar que sume puntuación todos los objetos usados en el tutorial contendrán en su etiqueta la palabra “tutorial” que indicará que a ese objeto no se suma puntuación, pero si actúa igual que el resto.

Tampoco se usará contador de tiempo para evitar poner nervioso al usuario y podrá acceder a este tutorial sin necesidad de introducir el nombre.

- Diseño y elaboración de niveles: Diseñaremos y elaboraremos los primeros niveles donde se usará un canvas para visualizar la puntuación del usuario, su nombre y el tiempo en el nivel.

- Diseño de menú y elaboración de escenas de historia: Se diseñará y elaborará el menú con las siguientes funcionalidades:

    - Un InputField para que el usuario introduzca el nombre que pasaremos a los niveles a través de un GameObject que no se destruirá al cargar la siguiente escena (mediante el método “DontDestroyOnLoad()”), donde también enlazaremos la puntuación con métodos getters y setters para poder modificar y obtener el nombre y la puntuación. Existirá un control para evitar que el usuario meta un nombre vació o rellenado con espacios (mediante el método Trim()).

- Un botón de juego nuevo que reiniciará las puntuaciones ya guardadas a 0 y el nombre al introducido por el usuario.
- Un botón para visualizar el Ranking de las 5 mejores puntuaciones ordenadas.
- Un botón de contexto que sencillamente mostrará una escena indicando la finalidad del juego y el motivo de la creación de este.
- Un botón de tutorial que nos llevará a un nivel sencillo donde se explicarán las mecánicas del videjuego.
- Un botón de salir que cerrará la aplicación.
- Dos botones, uno de importar datos en JSON y otro de exportarlos, que realizarán las funcionalidades descritas arriba para estos casos.

- Diseño, movimiento, control, colisión y comportamiento del personaje: El personaje se moverá mediante variaciones de su vector en el eje X e Y. Para su movimiento usaremos las teclas W, A, S y D, ya que son las que se usan habitualmente en juegos de control por teclado, por cada pulsación variaremos la posición del personaje, usando de esta manera un movimiento lineal en vez de un movimiento aplicando fuerzas, ya que al aplicar fuerza e inercia se vuelve más complicado y frustrante para el usuario el control preciso del personaje.

El personaje detectará las colisiones y dependiendo de la etiqueta del objeto con el que colisione realizará una función u otra:

- Si la etiqueta contiene tutorial realizará las mismas funciones que describiremos a continuación, pero evitando sumar la puntuación.
- Si la etiqueta con la que colisiona contiene “pw” (de PowerUp) tendrá que sumar la puntuación, destruir ese objeto y comprobar con qué tipo de PowerUp ha colisionado, comprobando las etiquetas completas y actuando en consecuencia:
  - Si colisiona con un objeto con la etiqueta “pwInvencible” tendrá que dar un indicador visual de que es invencible e inicializar el temporizador que irá sumando segundo hasta alcanzar el tiempo límite de invencible, tras lo que volverá a ser posible golpearlo. Para

que el enemigo detecte si es invencible usamos un booleano que indicará si el personaje es invencible o no.

- Si colisiona con un objeto con la etiqueta “pwAgresivo” tendrá que dar un indicador visual de que es agresivo e inicializar el temporizador que irá sumando segundos hasta alcanzar el tiempo límite de agresivo, tras lo que dejará de destruir a los enemigos al contacto. Para que el enemigo se destruya usaremos un booleano que si esta activado eliminará al enemigo y sumará la puntuación correspondiente.

- Si la etiqueta con la que colisiona contiene “enemigo” comprobará si el usuario está invencible o agresivo, en caso de que esté invencible sencillamente no ocurrirá el efecto de choque, la disminución de vida ni el efecto visual de golpeo. Si está agresivo destruiremos el enemigo (objeto con el que colisiona) y sumaremos puntuación y si ninguno de los dos está activo disminuiremos la barra de vida y restaremos la puntuación.

Al disminuir la barra de vida se comprobará si esta vale menos de 0.01 (no podemos igualar a 0 porque si el número de división es impar siempre habrá resto) finalizará el juego cargando una escena de “Game Over” (juego terminado). En este caso no se guardará la puntuación.

- Indicador de vida: Para indicar la vida del personaje usaremos un Sprite alargado y morado que irá disminuyendo por cada golpe. En principio el script recibirá un valor externo, aunque por defecto será 3 (pudiendo aumentarlo si modificamos este valor), a continuación, la barra restará su ancho menos el ancho total de la barra entre el número de golpes que se le permitirá al usuario:

  barraVida = barraVida – (barraVidaInicial / númeroGolpesPermitidos).

Por último, esta barra de vida se posicionará siempre encima del personaje controlado por el usuario, moviéndose para mantenerse en la zona superior de este.

- Diseño, movimiento, colisión y comportamiento de los enemigos o NPCs (Non – Player Character): Los enemigos tendrán la etiqueta “Enemigo” y buscarán la posición del personaje controlado por el jugador a partir de cierta cercanía con este y dejará de perseguirle si el jugador se vuelve alejar. Aunque el enemigo puede atravesar obstáculos para complicar el videojuego, tiene que colisionar con el

personaje controlado por el usuario. Además, debe haber algún indicador de la colisión tanto visual como sonoro para que el usuario se dé cuenta de esta colisión.

Por último, se indica que esto solo se produzca al colisionar, ya que si se le resta vida mientras colisionamos puede provocar la frustración del usuario debido a la falta de tiempo para escapar. De esta forma damos el suficiente tiempo al jugador como para reaccionar y escapar.

El control de la colisión se realizará en el script asociado al personaje controlado por el jugador.

- Diseño y elaboración de efectos de los PowerUps: Los PowerUps mejorarán al personaje dándole ciertas habilidades y tendrán la etiqueta “pwAtributo” donde el atributo será la modificación que le realizará al personaje (las modificaciones del personaje se controlarán en el script del personaje mediante la colisión). Los existentes son los siguientes:
    - PowerUp Invencible: Permite al jugador no recibir daño, aunque colisione con un enemigo, deshabilitando las respuestas visuales y la disminución de puntuación.
    - PowerUp Agresivo: Permite que sea ahora el jugador el que puede eliminar a los enemigos colisionando con ellos, además, sumará puntos al jugador por eliminar un enemigo.
- Pruebas técnicas: Se comprobará el correcto funcionamiento de la aplicación, si se detectasen errores, se solucionarían y continuaría probando.

3 Planificación del proyecto

3.1 Planificación de actividades y tareas. Temporización. Diagrama de Grantt

El videojuego estará desarrollado en distintos niveles que se cargarán tras leer un pequeño texto situacional.

- Diseño de implementación del modelo de datos.

    - Serializar y deserializar datos de prueba.
    - Crear una clase con los datos del usuario.
    - Serializar y deserializar un array de objetos con datos de prueba del usuario.

- Inserción y modificación de datos.
    - Insertar datos para comprobar su funcionamiento.
    - Modificar los datos para comprobar su funcionamiento y ver cómo afecta la actualización a los datos anteriores.
- Exportar e importar datos.
    - Exportar datos en JSON.
    - Importar datos en JSON.
- Visualización de datos.
    - Visualizar los datos del archivo serializado recorriendo el array de objetos y mostrando en pantalla el nombre y la puntuación.
- Control del Raking.
    - Controlar que los datos visualizados anteriormente se muestren en orden descendente.
    - En caso de modificación de datos, eliminar el de menor puntuación.
- Escena tutorial.
    - Plantear el tutorial.
    - Diseñar una escena tutorial.
    - Elaborar el tutorial.
- Diseño y elaboración de niveles.
    - Diseñar los niveles.
    - Elaborar los niveles.

- Diseñar el menú y escenas de historia.
    - Diseñar el menú.
    - Elaborar el menú.
    - Dar funcionalidad a las opciones del menú.
    - Plantear las escenas de la historia.
    - Elaborar las escenas de la historia.
- Diseño, movimiento, control de colisión y comportamiento del personaje.
    - Diseñar el personaje.
    - Implementar el movimiento.
    - Implementar y controlar las colisiones dependiendo de las etiquetas.
    - Definir el comportamiento del personaje ante colisiones.
- Diseño, movimiento, control y comportamiento de los enemigos.
    - Diseñar a los enemigos.
    - Implementar su movimiento automático centrado en el personaje.
    - Implementar y controlar la colisión.
    - Definir el comportamiento del enemigo con el entorno y el personaje.
- Diseño y efecto de los PowerUps.
    - Definir los powerUps.
    - Diseñar los powerUps.
    - Controlar su efecto al personaje o entidad afectada.
- Pruebas técnicas.
    - Control de posibles errores.
    - Control de errores al pasar datos entre escenas.

- Control de errores al colisionar el personaje con enemigos, barrearas, PowerUps o la meta.
- Control de errores al colisionar el enemigo con PowerUps.
- Control de errores al cerrar el juego forzándolo.
- Control de errores al importar y exportar datos al existir o estar vacíos estos.
- Control de errores visuales en diseños o animaciones.

- Elaboración de documentación.
  - Elaboración de la memoria.
  - Elaboración de un archivo relatando el progreso del desarrollo.

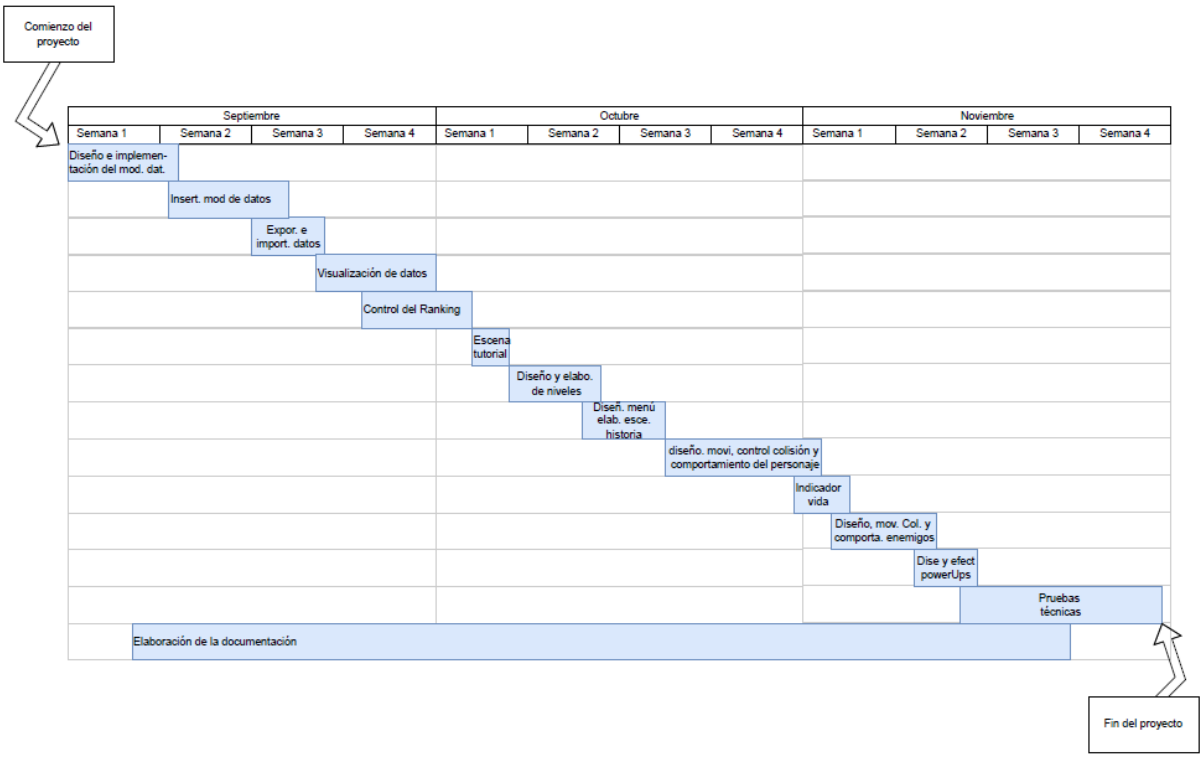


Diagrama de Grantt. Herramienta: Diagrams.net

Fecha inicio: 01 de Septiembre de 2024.

Fecha fin: 27 de septiembre de 2024

Número de días dedicados al proyecto: 80 días.

Horas por día: 3 – 4 horas.

3.2 Estimación de costes

Se ha calculado el coste de los siguientes perfiles en base al sueldo mínimo plasmado en el respectivo convenio:

	Cargo	Salario €/hora	Horas invertidas	Total
	Analista	10	16	160
	Programador	7	200	1400
	Diseñador gráfico	6,25	40	250
	Jefe de proyecto	14,6	40	584
Total				2394

Para el uso de software se ha intentado seleccionar software de uso libre, aunque para la modificación de imagen se ha usado programas del paquete de Adobe, por lo que el gasto es el siguiente:

	Software	Coste licencia €/mes	Meses utilizado	Total
	Creative Cloud	36,29	3	108,87
	Unity	0	3	0
	Visual Studio Code	0	3	0
	Teams	0	3	0
	Git	0	3	0
	Pixel Studio	0	3	0
Total				108,87

En cuanto a la publicación y venta del videojuego la mejor opción es la publicación en la tienda virtual “Steam”. La publicación en dicha plataforma tiene un gasto inicial de 100$ reembolsables una vez el juego alcance unas ganancias de 1.000$ brutos. La elección de Steam es debido a que es un líder en el mercado de ventas de juegos de forma digital, llegando a tener más de 34 millones de usuarios conectados de forma simultánea y 132 millones conectados mensualmente a la plataforma.

Se calcula además de que el gasto medio por cada usuario es de 23$, generando en el primer semestre de 2022 3.100 millones de dólares y lanzando en este mismo espacio de tiempo 6.000 juegos.

Además, para aumentar los ingresos existe la posibilidad vender también el juego en la plataforma digital “Epic Games” donde hay que pagar otros 100 dólares. Epic Games tiene más de 230 millones de usuarios registrados y, dependiendo del método de pago que se use, el creador puede llevarse entre el 88 y el 100% del valor de la venta.

	Publisher	Precio
	Steam	100
	Epic Games	100
Total		200

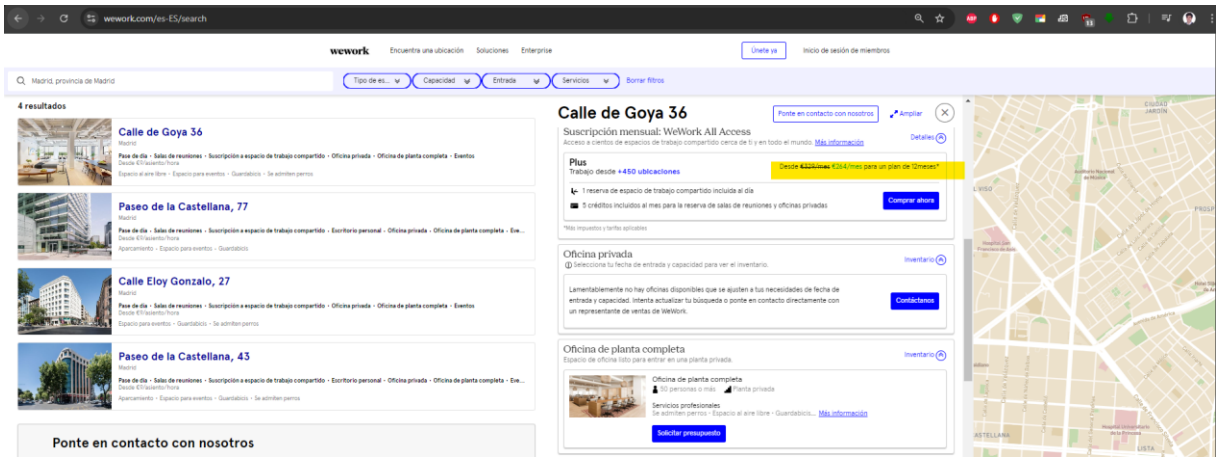
En cuanto al uso de hardware se han seleccionado equipos con buenas especificaciones en cuanto a RAM, GPU y CPU, ya que el uso de un equipo sin buenas características puede ralentizar el desarrollo e incluso bloquearlo, aumentando los costes:

	Hardware	Precio €	Cantidad	Total
	Monitor	84,99	4	339,96
	Teclado	7,38	4	29,52
	Ratón	6	4	24
	Torre	879	4	3516
	Tableta gráfica	29,99	1	29,99
Total				3939,47

El total global asciente a 6.442,34 euros.

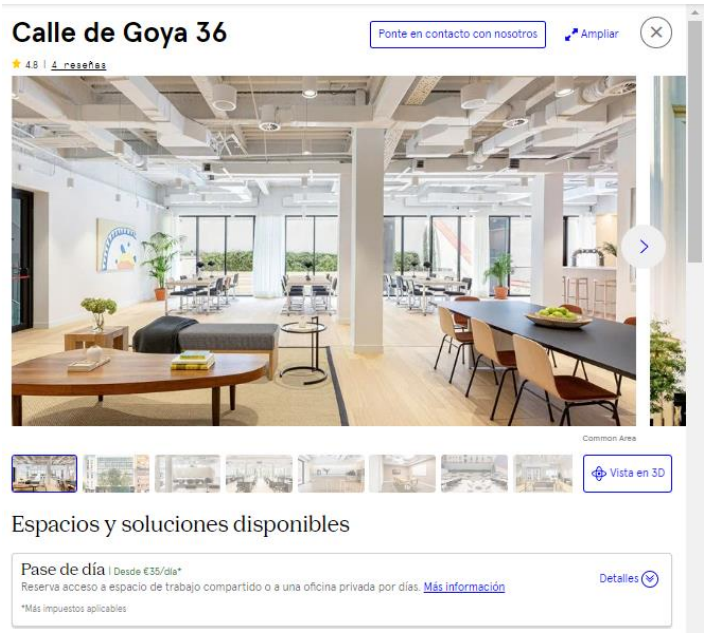
Otra opción es el uso de una oficina alquilada. Ante el encarecimiento de la vida y la imposibilidad de muchas personas de emprender han surgido web como https://www.wework.com/ que ofrecen la posibilidad de alquilar por horas, días, meses o incluso un año una oficina con equipo propio (aunque con espacio compartido). Analizando precios hemos encontrado que sale muchos más económico alquilar una oficina que comprar todo el equipo necesario más los posibles gastos de espacio en caso de no disponer de oficina o lugar privado para trabajar.

En la siguiente captura se puede observar cómo incluso tienen un servicio de suscripción anual:



Portal wework.com

Se puede optar a alquilar una oficia privada por 35 euros al día, dando resultado en un total de 3.150 euros por su uso durante 3 meses. La reserva no indica si se proporciona solo 1 equipo o varias (la web hace incapie en que es una oficina). Habría que consultar con el arrendatario para especificar cuantos equipos se pueden usar.



3.3 Previsión de riesgos del proyecto

- Perdida de codificación: Para evitar la posible pérdida de codificación se usarán varios métodos:
    - Uso de un repositorio en Git.
    - Uso del guardado en la nube de Unity.

- Creación de copias de seguridad y exportación de Assets cada semana.

- Problemas de optimización: Para evitar los problemas de optimización se programará siguiendo estándares de calidad, evitando repetir código y sobrecargar al proyecto con Sprites u GameObjects sin necesidad.

- Documentación ineficiente: Para evitar la pérdida de datos de desarrollo y la documentación reiterativa o ineficiente se han creado dos archivos, uno que se usará de memoria y otro que se irá rellenando diariamente con las modificaciones e implementaciones tanto en el código como en el diseño, indicando el perfil que lo ha modificado y la fecha.

- Código poco claro: Se documentará todo el código indicando que hace, como lo hace y a que parte del proyecto afecta.

Probab.\Consec.	**Grave**	**Importante**	**Leve**
Alta		Código poco claro	Documentación inef.
Media	Perdida de codificación	Problemas de optimización	
Baja			

Siguiendo la siguiente gráfica de riesgos se debe priorizar primero instaurar las medidas necesarias para evitar los riesgos altos:

Probabilidad		**Muy probable**	**Probable**	**Improbable**	**Altamente improbable**
Consecuencias	**Fatalidad**	**Alto**	**Alto**	**Alto**	**Medio**
	Lesiones importantes	**Alto**	**Alto**	**Medio**	**Medio**
	Lesiones leves	**Alto**	**Medio**	**Medio**	**Bajo**
	Lesiones insignificantes	**Medio**	**Medio**	**Bajo**	**Bajo**

Fuente: https://safetyculture.com/

4 Análisis y Diseño del proyecto

4.1 Diseño y arquitectura

4.1.1 Diseño

4.1.1.1 Decisiones de diseño

Los colores elegidos para el proyecto son los siguientes (de más claro a más oscuro): #d8d8ee, #c4c4dd, #a8a9d1, #3b2e6b, #1b1339 y #0a0000.

Se han seleccionado colores armónicos, es decir, aquellos que pertenecen a una misma sección del círculo cromático (en nuestro caso, a los colores “Fríos”).

Herramienta: https://color.adobe.com/es/create/color-wheel

En cuanto a la cámara del personaje, se ha decidido que la vista sea cenital (la cámara es perpendicular al suelo), ya que además de facilitar el modelado de los sprites, permite posicionar visualmente mucho más fácilmente los enemigos y el personaje, permitiendo a los usuarios pensar más rápidamente sus acciones.

4.1.1.2 Sprites

En lo referente a los sprites en principio se pensó en usar una estética de punto de cruz para dar personalidad al proyecto, pero se descartó debido a la lentitud y la complejidad de usarlos como modelo, aunque se elaboraron algunos bocetos como prueba copiando los sprites de videojuegos famosos.

Diseño de algunos sprites en punto de cruz

Finalmente, los sprites se han elaborado con las herramientas Photoshop, After Effects y PixelStudio, pero debido a nuestro desconocimiento en cuanto al uso de estas herramientas se han seguido tutoriales (todos referenciados en la bibliografía) y copia sprites modificando ciertos puntos para adaptarlo a nuestro proyecto. Los sprites son los siguientes.

- Enemigos: El Sprite de los enemigos es un vórtice animado mediante Unity. El vórtice hace referencia al bucle que comúnmente se produce al tener pensamientos negativos, ya que estos mismos pensamientos alientan tener más, provocando entrar en una espiral.

- PowerUp Invencible: El sprite de este PowerUp representa a los fármacos que se suministran a los pacientes que padecen de depresión, bloqueando los efectos negativos, pero siendo tarea de la persona que la padece seguir hacia delante y mejorar.

- PowerUp Agresivo: El Sprite de este PowerUp representa la salud y estabilidad mental que se necesita para enfrentar a los pensamientos negativos y eliminarlos.

- PowerUp Multiplicar: El Sprite de este PowerUp representa la utilidad de escribir los sentimientos y situación tanto a distintas personas como de forma personal.

- Meta: Este Sprite representa la capacidad de pasar un nuevo día.

- Personaje: Este Sprite representa a una persona con distintas animaciones tanto para andar como para estar estático.

- Otros: Se han creado Sprites de setos (distintos tamaños y formas) para delimitar el mapa y flores para adornar el fondo.

4.1.1.3 Título

Para el título se estuvieron barajando varios nombres como “Grietas”, “Cristal”, “Depressio” o “Tempo”, pero al final el elegido es “Alegoría”, ya que la RAE define esta palabra como:

“Ficción en virtud de la cual un relato o una imagen representan o significan otra cosa diferente.”

Por ello, y como el juego pretende ser una alegoría del sentimiento de la depresión, se ha considerado que es el correcto y el más afín.

4.1.1.4 Referentes

Para la elaboración del proyecto se han tomado como referentes varios juegos que han sido clave para la cultura del videojuego.

Como referencia para la jugabilidad se ha usado el videojuego “Haunted Maze”, de la PlayStation 1, el cual consiste en un protagonista consiguiendo puntos obteniendo ítems a medida que es perseguido por zombis. Debe conseguir X ítems para avanzar y en el menor tiempo posible para obtener más puntuación.

Fuente: https://www.youtube.com/@RetroPixelLizard

Visualmente se ha intentado asemejarse o referenciarse en los videojuegos “Limbo” o “The end is nigh”, juegos oscuros que tratan temas serios pero su jugabilidad es simple y entretenida, siendo plataformas e historias lineales.

Portadas LIMBO y THE END IS NIGH

Como inspiración temática, aparte de experiencias vitales, se ha elegido como referentes los juegos “Fear and Hunger” y “Dont Starve”, juegos oscuros y complicados que tratan temas complejos y delicados como puede ser la depresión, la locura o la soledad.

Portada FEAR AND HUNGER

Portada DON’T STARVE

4.1.1.5 Música de fondo

Para la música de fondo se ha utilizado música clásica ya que está libre de derechos de autor. Son las siguientes melodías:

- Greensleeves: Autor desconocido, aunque las leyendas dicen que su autor es Enrique VIII.
- Concierto para 2 violines de Bach.

- Nocturno Op9 Nº1 de Chopin.
- Silencio de Beethoven.
- Faust de Wagner.
- Canon en D mayor de Pachebel.

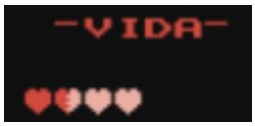
4.1.2 Arquitectura

En cuanto a la arquitectura del proyecto lo utilizado diversas tecnologías.

En lo referente al motor del videojuego se ha utilizado Unity debido a su versatilidad y a lo aprendido durante el curso. Se ha intentado que no existan valores fijos (más allá de los estrictamente necesarios) permitiendo siempre pasarlos gracias a declararlos públicos en el script asociado al GameObject.

Para los golpes que puede recibir el personaje se barajaron varias opciones:

- Que aparecieran unos cerebros en la parte superior de la pantalla representando cuantos golpes podía recibir el personaje utilizando como referencia a los videojuegos de Zelda que lo representan mediante corazones.



Vida en Zelda

Se descarto debido a que obligar al usuario a vigilar tanto la zona de juego como la zona de la vida puede saturar al usuario, por lo que representamos la vida mediante una barra superior.

- Para no tener que definir los golpes de vida directamente en el código se ha usado la siguiente fórmula matemática:

barraVida = barraVida – (barraVidaInicial / númeroGolpesPermitidos).

De esta forma, cuando el resultado de la barra de vida sea inferior a 0.1 (debido a que los números impares dejan decimales) el personaje se morirá, saltando a la escena de “Game Over”.

Para la conservación de datos se ha decidido serializar los datos, ya que muchos juegos con pocos datos recurren a está técnica en vez de uso de bases de datos (BBDD) debido a que estas consumen muchos recursos. Además, los datos se pueden exportar e importar en JSON permitiendo al usuario modificarlos y enviarlos a otro usuario para poder compartir sus puntuaciones. Para definir la ruta de serialización se usa la ruta suministrada por Unity mediante el método “Application.persistentDataPath”, que devuelve una ruta distinta dependiendo del sistema operativo del usuario.

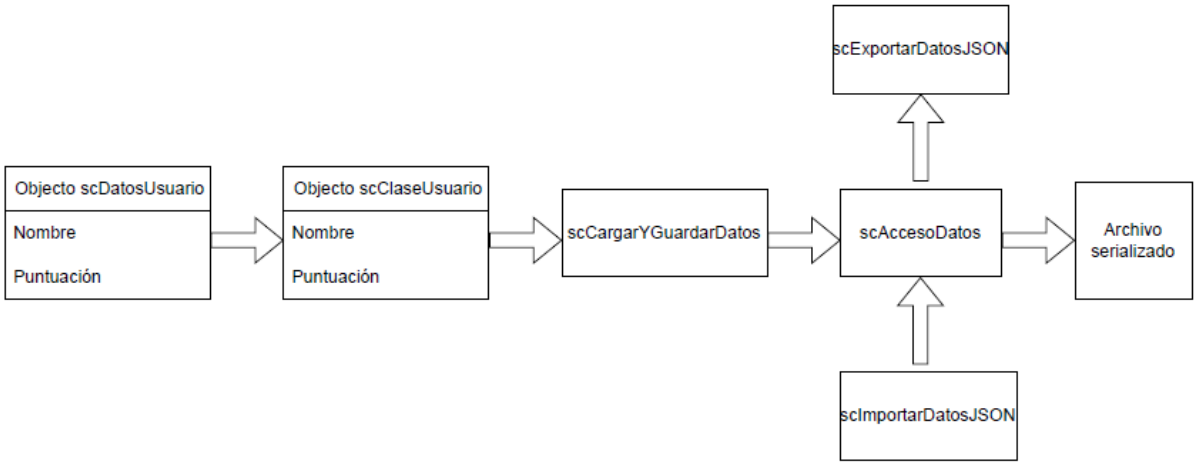
Los datos se van guardando según se va jugando en un GameObject que se no se destruye entre escenas (mediante el método “DontDestroyOnLoad()”) permitiendo ir guardando y sumando las puntuaciones por nivel del usuario y reiniciando sus datos al pulsar en “Juego Nuevo”. El objeto se ha llamado “DatosUsuario” y es una clase que permite crear objetos con los atributos “nombre” y “puntuacion”, permitiendo guardar los datos de los usuarios en objetos y serializarlos para guardarlos.

A continuación, surgió el problema de que no se podían serializar las clases que heredan de “MonoBehaviour” y es un requisito indispensable que hereden de esta para poder ir asociados con GameObjects por lo que se usó una clase auxiliar que tiene los mismos atributos que “DatosUsuario” pero que no hereda de “MonoBehaviour” permitiendo así su serialización.

Finalmente, al serializar solo se guarda un datos u objeto, por lo que, para poder añadir un ranking, se ha decidido serializar un array de las 5 mejores puntuaciones, creando así un array de objetos con los datos de los mejores usuarios. Al intentar guardar partida, deserializará estos datos, comprobará si es superior o igual a la peor puntuación ya guardada, y en caso de que lo sea y mediante la ayuda de un array auxiliar modificará el array para guardar la nueva puntuación.

En cuanto a los scripts, se ha intentado aislar lo máximo posible cada funcionalidad con el objetivo de que sea más fácil interpretar su funcionamiento y uso.

4.2 Modelo de datos



Herramienta: Diagrams.net

El objeto que se encarga de recoger el nombre y el usuario pertenece a la clase “scDatosUsuario” y recibe el nombre de “DatosUsuario”. Este objeto no se destruye entre escenas y va guardando la puntuación total y el nombre inicial del usuario. Al estar unido a un GameObject debe heredar de la clase “MonoBehaviour” impidiendo así su serialización, por lo que se usa una clase auxiliar “scClaseUsuario” que no hereda de esta y le permite serializar.

Una vez recibida la orden de guardar los datos, estos se pasan a “scCargarYGuardarDatos” que se encarga de comprobar si esos datos son lo suficientemente altos como para superar la peor puntuación de los ya existentes, en caso de que así sea, guarda los datos en el array de objetos ya existente deserializando el previamente guardado y serializando el nuevo con la peor puntuación eliminada.

Finalmente llama a “scAccesoDatos” que se encarga de guardar o cargar los datos desde un archivo, serializando o deserializando el array de objetos existente en este y guardarlo en un array de tipo “scClaseUsuario”.

Exportar e importar JSON permite usar el script “scAccesoDatos” para deserializar los datos y o guardarlos en un archivo JSON (previo paso a esta estructura) o pasar los datos de un JSON a un array de objetos “scClaseUsuario” que posteriormente se pasa al método para sobreescribir el array.

4.3 Modelo de procesos

El diagrama de casos de uso es el siguiente:

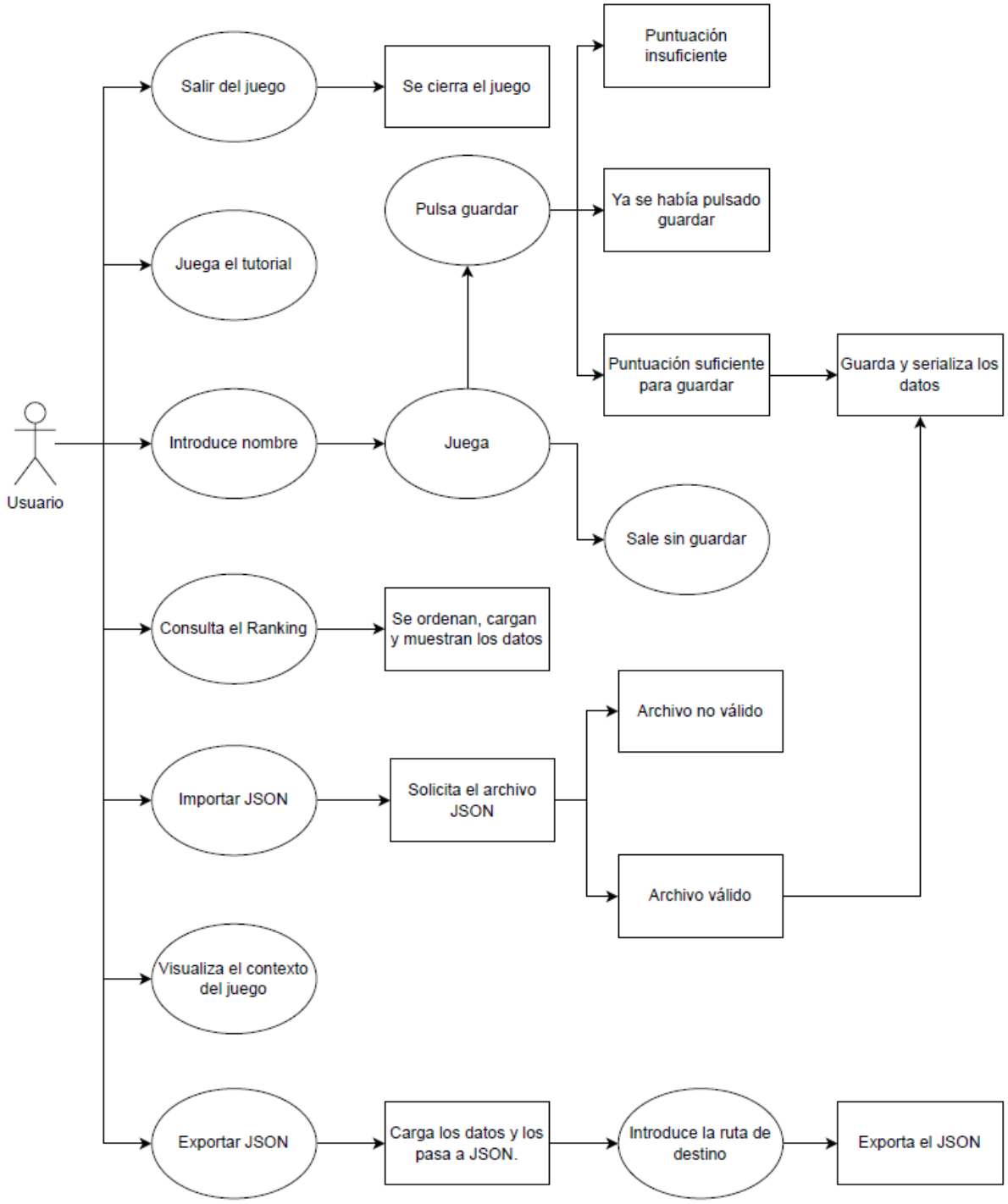


Diagrama de casos de uso. Herramienta: Diagrams.net

Ahora profundizaremos en los casos de uso más completos, siendo estos el proceso de jugar del usuario, de exportar y de importar JSONs.

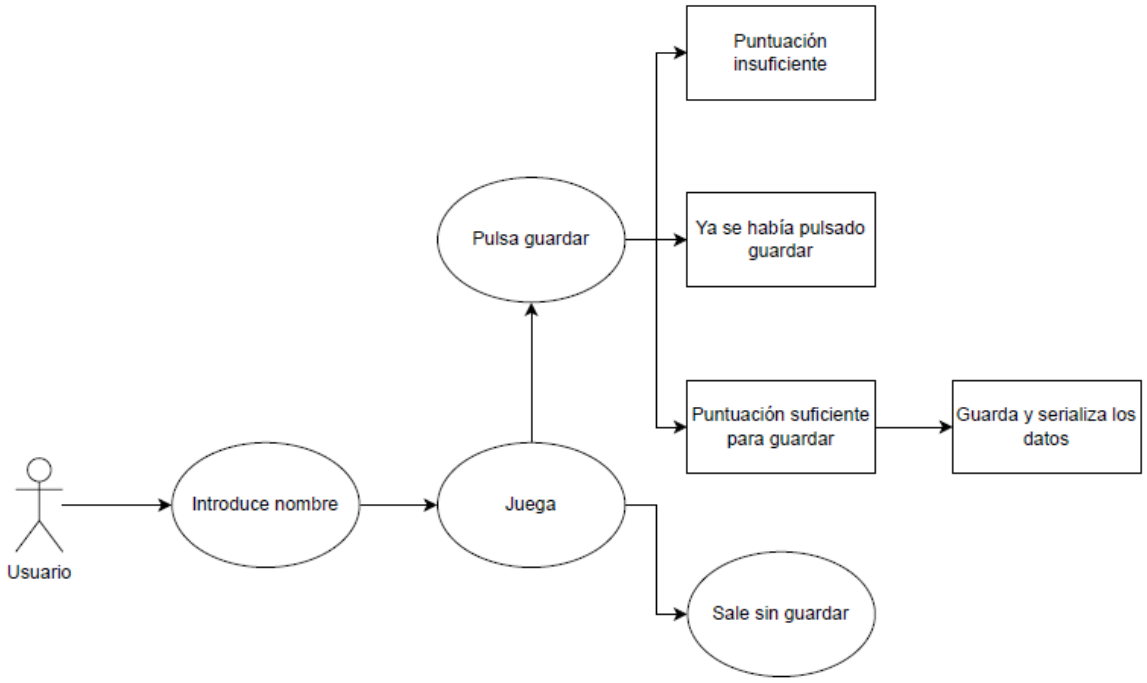


Diagrama de caso de uso: Jugar. Herramienta: Diagrams.net

Diagrama de caso de uso para jugar:

1º. El usuario introduce un nombre y pulsa jugar: El usuario introduce un nombre, pulsa en jugar y va jugando los niveles obteniendo la puntuación necesaria, en caso de que pierda o cierre el juego antes de llegar al final no podrá guardar.

2º. El usuario pulsa guardar: En caso de que el usuario ya hubiera pulsado guardar o la puntuación fuera insuficiente se mostrará una retroalimentación indicando el caso. Si la puntuación es suficiente (comprobado mediante la carga de datos ya existentes y recorriendolos mientras los comparar con la nueva puntuación) la introduce en la puntuación guardada, elimina la menor y serializa los datos.

3º. El usuario vuelve al menú inicial.

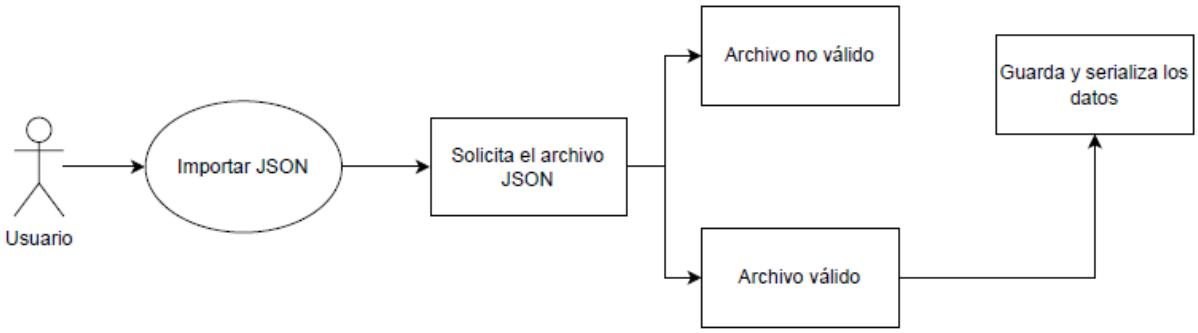


Diagrama de caso de uso: Importar JSON. Herramienta: Diagrams.net

Diagrama de caso de uso para importar JSON:

1º. El usuario pulsa importar JSON: Abre un explorador de archivos donde pide seleccionar un archivo JSON.

2º. El usuario selecciona un archivo JSON: En caso de que no seleccione uno correcto (o cierre la ventana) se mostrará una retroalimentación indicando que el archivo no es válido.

3º. Si el archivo es correcto pasa los datos a un array de objetos con los datos de los usuario y llama al método encargado de guardar los datos pasandole el array y guardandolo.

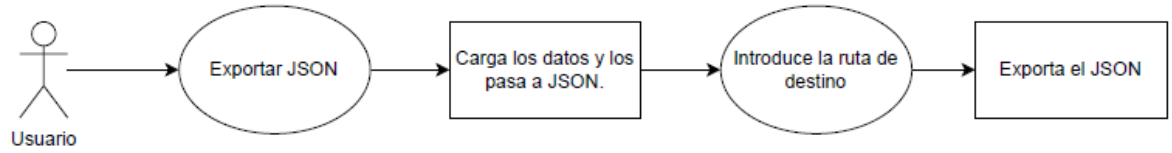


Diagrama de caso de uso: Exportar JSON. Herramienta: Diagrams.net

Diagrama de caso de uso para exportar JSON:

1º. El usuario pulsa exportar JSON: Abre un explorador de archivos donde pide seleccionar una ruta para el archivo JSON.

2º. El usuario seleciona una ruta: En caso de no ser una ruta válida o de cancelar la ruta se mostrará una retroalimentación indicando el problema.

3º. Si el archivo es correcto carga y deserializa los datos guardados y lo guardo en un archivo formato JSON, exportandolo en la ruta seleccionada previamente.

5 Construcción del proyecto

5.1 Codificación

El siguiente método devuelve la puntuación final restando a la puntuación el 10% del tiempo gastado.

```
public float getPuntuacionFinalNivel(){

    return Mathf.Round(puntuacion - temporizadorNivel/10); //Retorna la puntuación final restando el 10% del tiempo gastado en el nivel y redondeando

}
```

El siguiente método resta la vida del personaje y comprueba cuando debe saltar la pantalla de “Game Over”.

```
private void DisminuirBarraVida(){

    tamannoBarraVidaActual -= tamannoBarraVidaInicial/golpesRestantes; //Voy restando el tamaño de la barra

                                barraVida.transform.localScale = new Vector2(tamannoBarraVidaActual,anchoBarraVida); //Y actualizo

    if(tamannoBarraVidaActual < 0.01){ //Si la vida es menor de 0,01 (no puedo usar 0 por numeros impares)

        SceneManager.LoadScene("GameOver");  //salta a la pantalla de gameOVer

    }

}
```

Al recibir públicamente la variable “golpesRestantes” podemos indicar desde Unity cuantos golpes aceptará el personaje antes de morir sin tener que tocar el código.

El siguiente método se encarga de comprobar su la puntuación es superior y de modificar el array para introducirla eliminando la menor puntuación.

```
 private void ComprobarPuntuacion(){ //Comprobamos la puntuacion


        int contador = 0;

        int contadorAux = 0;

        //El arrayAuxiliar nos servirá en caso de que ya estén guardados

        if(scAccesoDatos.CargarDatosUsuario() != null){ //Si existen ya datos de usuarios

           arrayUsuarios = scAccesoDatos.CargarDatosUsuario(); //Los cargamos

           arrayAuxiliar = scAccesoDatos.CargarDatosUsuario();



        }

        if(arrayUsuarios == null){ //Si no existen

            arrayUsuarios = scInicializarDatos.DatosIniciales(); //Los inicializamos

            arrayAuxiliar = scInicializarDatos.DatosIniciales();

        }


        Array.Sort(arrayUsuarios); //Lo ordenamos

        Array.Sort(arrayAuxiliar);

        if(!evitarRepetirDatos){ //Evita que compruebe cada vez que se pulsa guardar datos los datos, lo que provoca la exception de salirse del array

            foreach (scClaseUsuario usuario in arrayUsuarios){ //Recorremos el array
```

```
            if(usuario.puntuacion <= datosSerializados.puntuacion && !yaGuardado){ //Si la puntuacion del usuario es menor  o igual a la que debemos guardar

            guardar = true; //Le indicamos que se debe guardar

            yaGuardado = true; //Y que ya se ha guardado (evita que sobreescriba todos los puntajes por debajo de este)

             usuario.nombre = datosSerializados.nombre; //Sobreescribimos el valor de ese nombre

            usuario.puntuacion = datosSerializados.puntuacion; //Y se su puntuacion

            contadorAux = contador + 1;

                /*Ahora está el problema de que ha sobreescrito encima del primero inferior que encuentra, pero debemos ir sobreescribiendo

                los valores hacia abajo, para ello creamos el array auxiliar*/

          }

          contador++;



        }

        if(yaGuardado){ //Si ya se ha guardado

          try{

              for(; contadorAux < arrayAuxiliar.Length; contadorAux++){ //Por la duración de contador hasta el final del array auxiliar

                //Y sobreescribimos los usuarios, haciendo que bajen todos 1 en el ranking a partir del introducido

                arrayUsuarios[contadorAux].nombre = arrayAuxiliar[contadorAux-1].nombre;

                        arrayUsuarios[contadorAux].puntuacion  =  arrayAuxiliar[contadorAux-1].puntuacion;
```

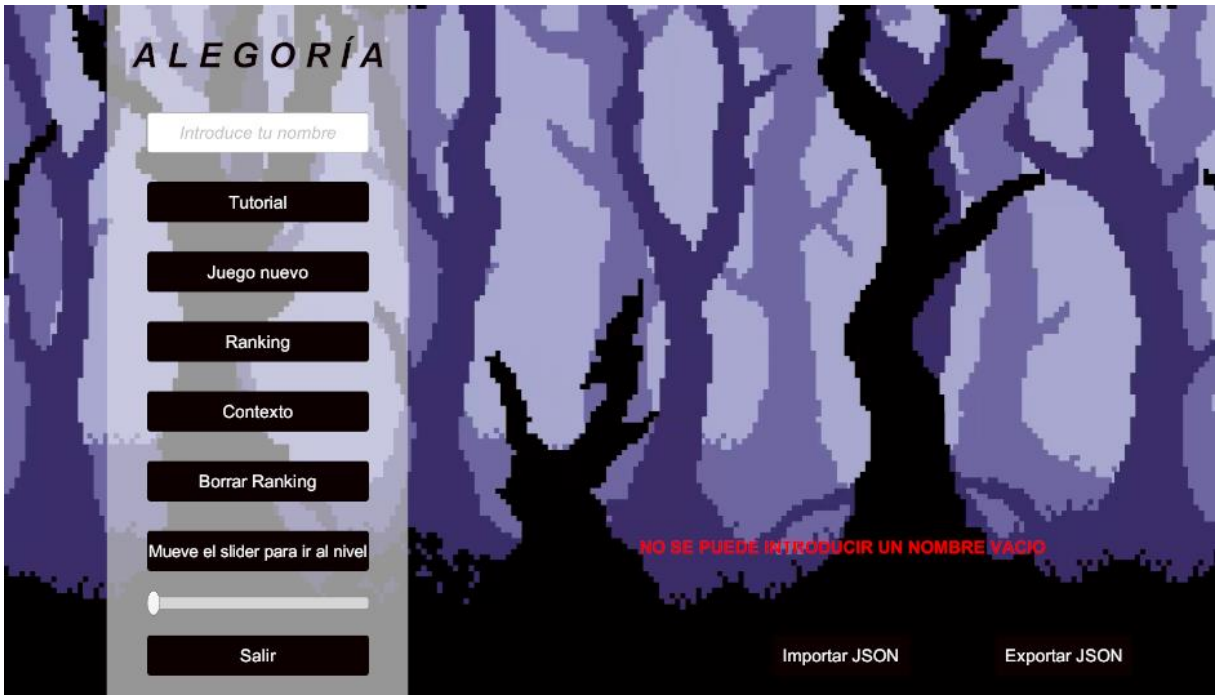
```
                }
            }catch(IndexOutOfRangeException fueraDeRango){
                        Debug.LogError("FUERA DEL RANKING, EXCEPTION: " + fueraDeRango.Message);
            }
        }
    }
}
```

Este fragmento de código es probablemente el que más problemas me ha dado del proyecto ya+, tras realizar el pseudocódigo y realizar pruebas no funcionaba. Tras investigar el funcionamiento de los arrays (ya que tras modificar el código y asegurar que funcionaba con otros lenguajes no entendía porque en este no) descubrí que si igualabas los arrays el segundo array creaba una referencia al primero, no creaba otro objeto, por lo que lo solucioné creando dos arrays e indicando la misma fuente de datos.

Se han utilizado la librería “StandaloneFileBrowser” para poder utilizar el explorador de archivos al importar y exportar los archivos JSON.

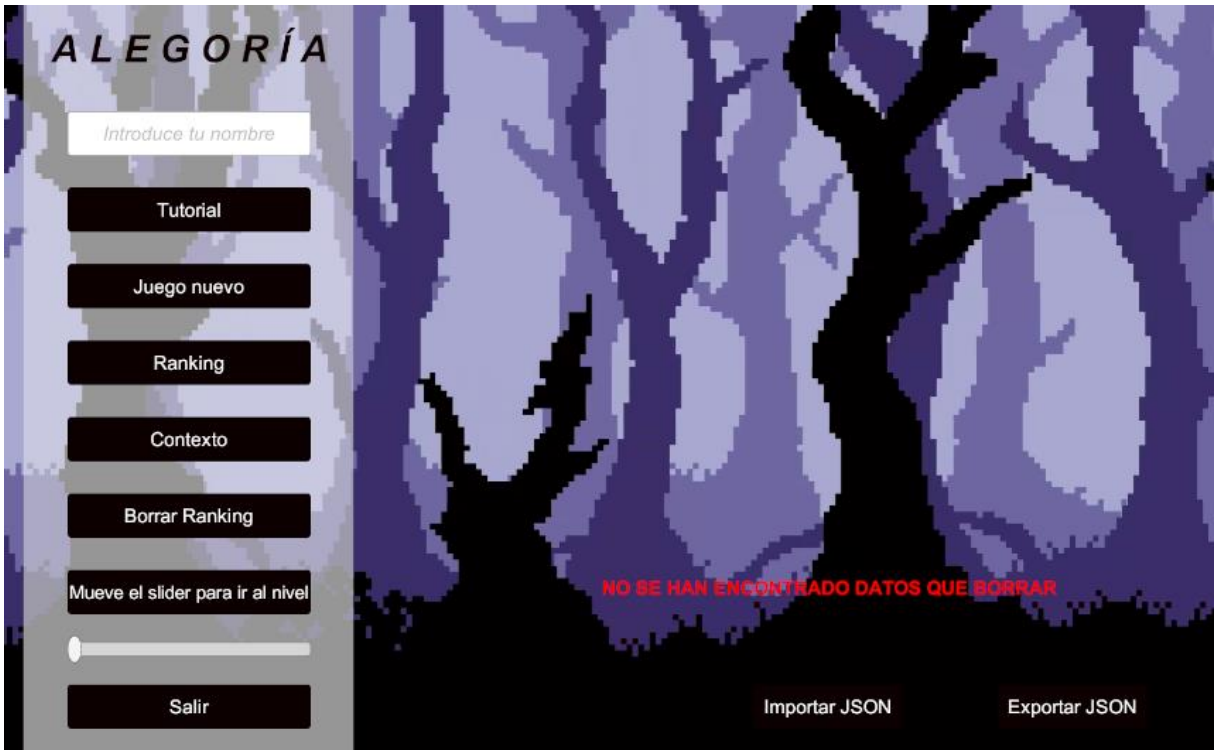
5.2 Pruebas

Se han realizado múltiples pruebas, siendo las más comunes la comprobación del correcto paso de datos entre escenas. También se ha comprobado que no se pueda jugar sin introducir un nombre vacío o espacios, en caso de que el usuario lo intente, mostrará un mensaje de error.



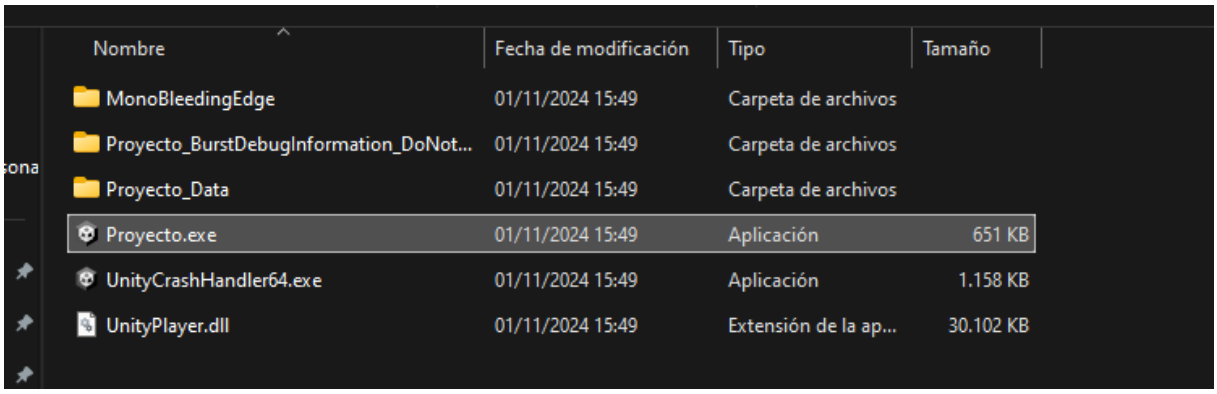
A la hora de importar y exportar JSONs también se han controlado los posibles errores como la cancelación de esta acción o la selección de un archivo con distinta extensión o estructura. Si cancelamos estas acciones:

En caso de que se intente borrar el ranking aunque no exista se mostrará el siguiente mensaje:



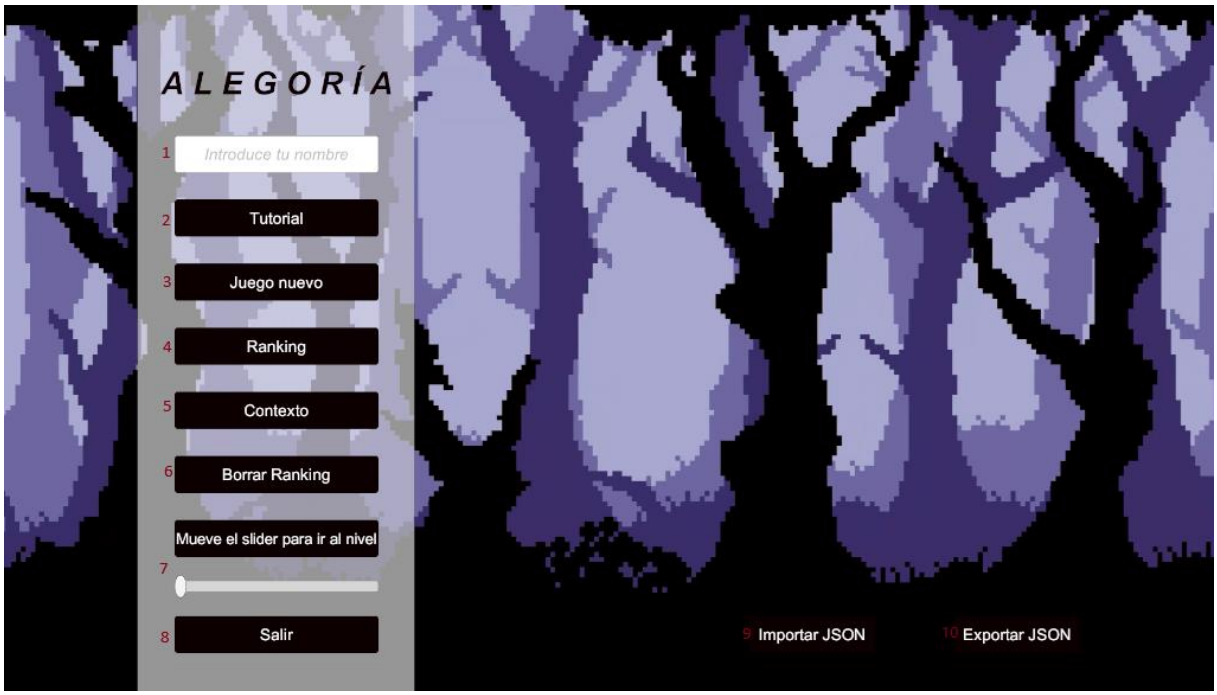
5.3 Manual de instalación

Para la instalación del videojuego solo será necesario seleccionar el archivo “proyecto.exe” para la ejecución de este.



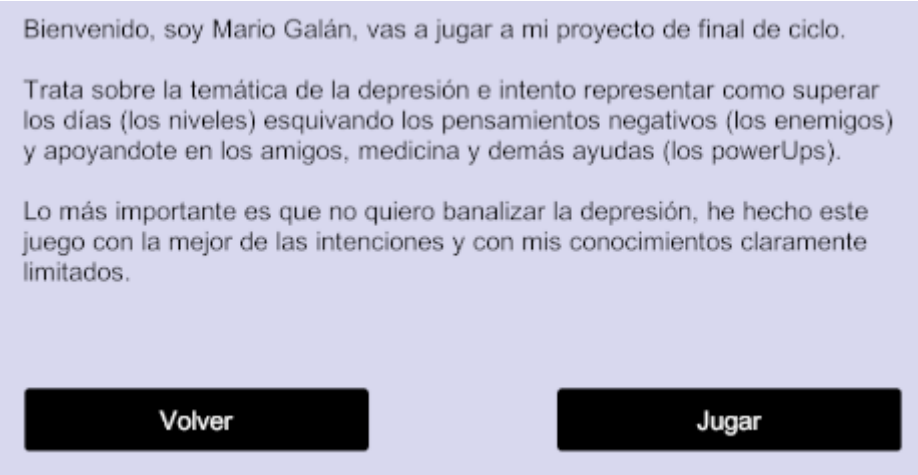
5.4 Manual de usuario

Al iniciar el videojuego nos saldrá el menú inicial donde podremos seleccionar varias opciones (numeradas del 1 al 10):



1. Campo para introducir el nombre del jugador.
2. Lleva a la escena del tutorial donde se explica la jugabilidad.
3. Permite iniciar un nuevo juego (tras comprobar que el nombre está rellenado).
4. Muestra las 5 mejores puntuaciones y los usuarios que las completaron.
5. Muestra una pantalla donde se explica la motivación del videojuego.
6. Borra el ranking actual.
7. Permite acceder directamente a un nivel del videojuego (tras comprobar que el nombre está rellenado) pero con puntuación 0.
8. Permite salir de la aplicación.
9. Abre el explorador de archivos permitiendo importar un JSON.
10. Abre el explorador de archivos permitiendo exportar el ranking actual en forma de JSON

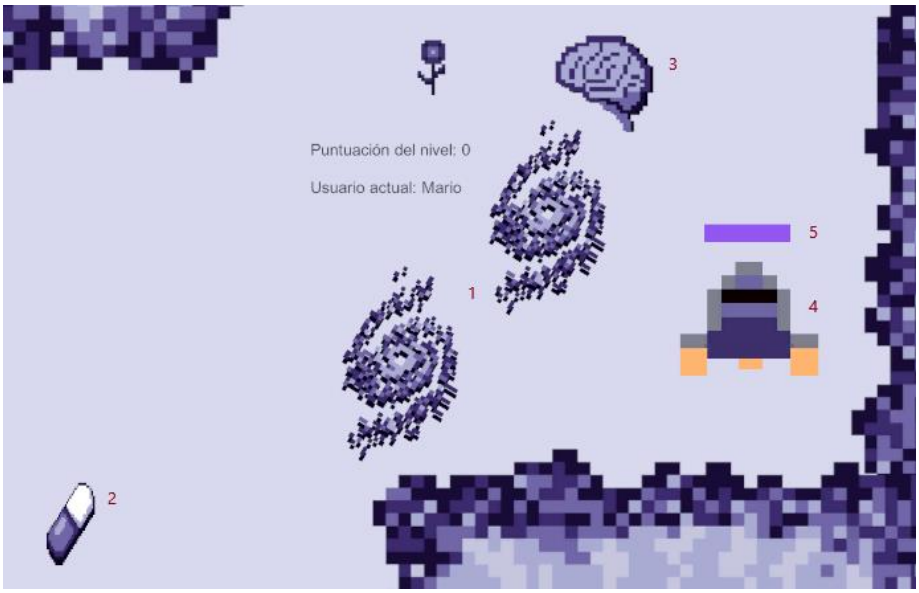
Una vez pulsemos en el juego nuevo mostrará la siguiente pantalla:



En esta pantalla se muestra una breve introducción, podemos volver al menú pulsando en "Volver" o jugar el primer nivel pulsando en "Jugar".

Al pulsar en jugar podremos iniciar el juego, donde tendremos que llegar hasta nuestra meta, representada por una cama para indicar que hemos conseguido superar un día más.

Controlaremos nuestro personaje y nos indicará su vida en una barra morada encima de él. Debemos evitar a los enemigos a la vez que cogemos los PowerUps para aumentar nuestra puntuación donde cada PowerUp nos dará ventajas como invulnerabilidad, multiplicar los puntos obtenidos o poder eliminar a los enemigos (sumando más puntuación). En caso de que los enemigos nos golpeen perderemos tanto vida como puntuación, y si nuestra vida llega a 0 perderemos.



En el siguiente pantallazo se pueden observar a los enemigos (1) y dos PowerUps distintos (2 y 3) además del personaje y su vida actual (4 y 5 respectivamente).

Ahora deberemos ir pasando niveles hasta llegar al último, donde, tras obtener nuestra puntuación, podremos elegir entre guardar la puntuación o volver al menú.

6 Evaluación final del proyecto

6.1 Evaluación del diseño, del proceso y del resultado

En el diseño inicial la estética era mucho más oscura, haciendo diseños garabateados e inconexos intentando igualar a la confusión que se produce en la mente humana. El problema surgió cuando a la hora de jugar todo era confuso y nada gratificante visualmente, por lo que se decidió por una estética PixelArt y con tonos morados.

6.2 Conclusiones y lecciones aprendidas

Se ha aprendido mucho sobre codificación en C# y el funcionamiento de GameObjects en Unity, permitiendo que la sensación al programar los atributos de estos en este lenguaje sea mucho más satisfactoria y resulte mucho más sencillo. Además, también se han aprendido características del lenguaje C# como su funcionamiento en arrays.

6.3 Posibles ampliaciones futuras

Para posibles ampliaciones futuras se puede implementar que la puntuación total de todos los usuarios vaya guardándose y que esta sirva para comprar mejoras desde la pantalla de menú que darán mejoras como multiplicadores de puntuación, más velocidad de movimiento, más vida, etc.

7 Bibliografía/Webgrafía

BillWagner. (2024, 15 mayo). *Matrices - C# reference*. Microsoft Learn. https://learn.microsoft.com/es-es/dotnet/csharp/language-reference/builtin-types/arrays

¿Como mover a mi personaje segun la direccion a la que mire? How do I move my character according to the direction they're facing? (2023, 22 mayo). Unity Discussions.

https://discussions.unity.com/t/como-mover-a-mi-personaje-segun-la-direccion-a-la-que-mire-how-do-i-move-my-character-according-to-the-direction-theyre-facing/257951/2

Dan. (2021, 22 septiembre). *Software desde 0 – 7 – Como definir la funcionalidad y el Alcance. Historias y Escenarios*. https://d0a.dev/2021/09/22/software-desde-0-7-como-definir-la-funcionalidad/

Design Toolkit | Interfaces de juego: control. (s. f.). https://design-toolkit.recursos.uoc.edu/es/interfaces-de-juego-control/

Draw Pixel Art Online - pildora. (s. f.). Pixilart. https://www.pixilart.com/draw/pildora-616526f96d9bbe6

draw.io - free flowchart maker and diagrams online. (s. f.). https://app.diagrams.net/

El diseño de interfaz de usuario en un videojuego | Tokio School. (s. f.). Tokio School. https://www.tokioschool.com/noticias/diseno-interfaz-videojuego/

Epic Games. (2023, 9 marzo). *Epic Games Store lanza herramientas de autopublicación para desarrolladores y editoras*. store.epicgames.com. https://store.epicgames.com/es-ES/news/epic-games-store-launches-self-publishing-tools-for-game-developers-and-publishers

Fading in/out GUI text with C#? {SOLVED}. (2016, 20 enero). Unity Discussions. https://discussions.unity.com/t/fading-in-out-gui-text-with-c-solved/613416/2

Learn game development w/ Unity | Courses & tutorials in game design, VR, AR, & Real-time 3D | Unity Learn. (s. f.). Unity Learn. https://learn.unity.com/tutorial/como-implementar-la-continuacion-de-datos-entre-escenas?language=es#

Libro abierto con estilo pixel art | Vector Premium. (2021, 18 octubre). Freepik. https://www.freepik.es/vector-premium/libro-abierto-estilo-pixel-art_19580503.htm

Ministerio de Sanidad, Consumo y Bienestar Social - Portal Estadístico del SNS - Base de Datos Clínicos de Atención Primaria - BDCAP. (s. f.). https://www.sanidad.gob.es/estadEstudios/estadisticas/estadisticas/estMinisterio/SIAP/

Portilla, C. P. (2024, 4 marzo). *Steam alcanza nuevo récord de usuarios simultáneos con 34,6 millones*. www.latercera.com. https://www.latercera.com/mouse/steam-alcanza-nuevo-record-de-usuarios-simultaneos-con-346-

millones/#:~:text=Esta%20es%20la%20primera%20vez,6%20millones%20de%20usuarios%20conectados

Sign up & Log In - Moqups App. (s. f.). https://app.moqups.com/

Solución. (2024, 23 enero). Estadísticas de Steam que debe conocer [Actualización noviembre 2024]. *Scottmax.com*. https://scottmax.com/es/estadisticas-de-vapor/#How_much_revenue_did_Steam_generate_in_2022

Sprite de cama. (s. f.). www.pngegg.com. https://www.pngegg.com/es/png-nsbam

Steam Direct Product Submission Fee on Steam. (s. f.). https://store.steampowered.com/sub/163632?l=spain

Team, B. (2024, 14 junio). *Steam Usage and Catalog Stats*. Backlinko. https://backlinko.com/steam-users

Technologies, U. (s. f.). *Unity - Manual: Serialización JSON*. https://docs.unity3d.com/es/530/Manual/JSONSerialization.html

Usuarios de Youtube:

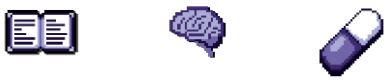
- KopDesarrollo.
- IndieNuggets.
- Nicosiored.
- OmelPixela.
- DiseñoyVideojuegos.
- Hanapix.

8 Anexos

8.1 Sprites utilizados en el videojuego.

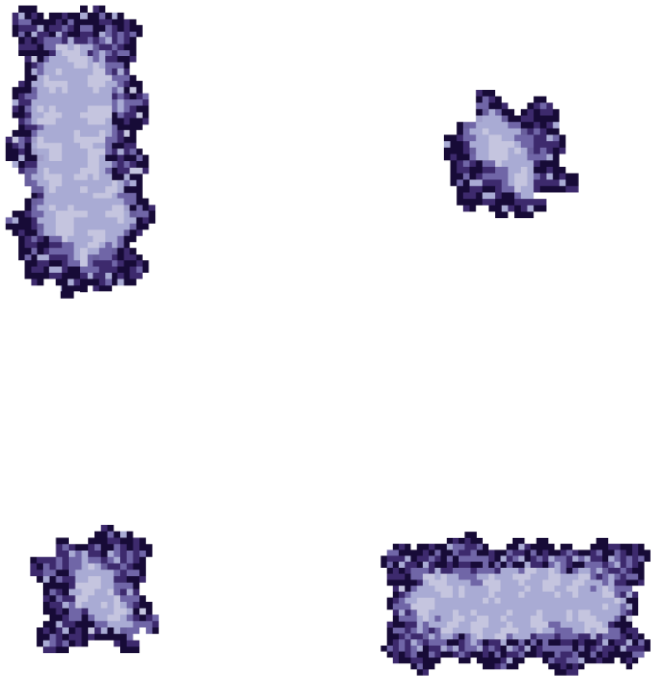
Enemigo

Meta

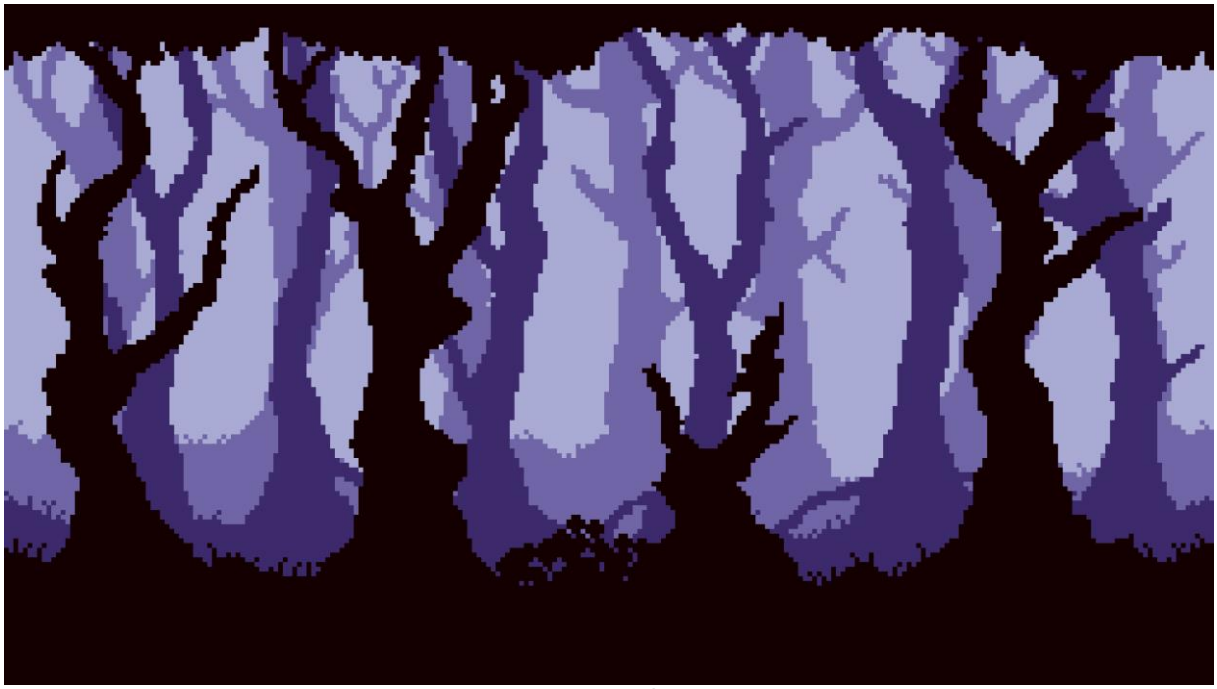


PowerUps multiplicar, agresivo e Invencible

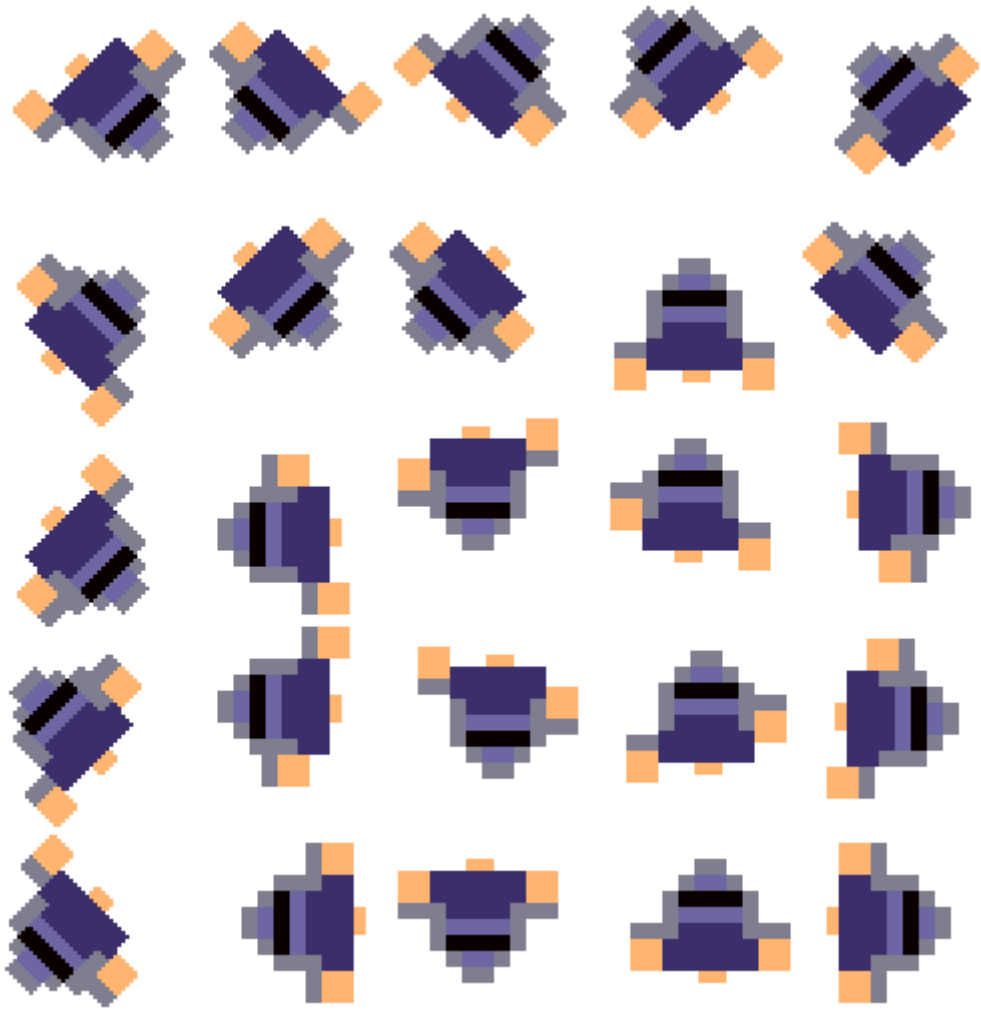
Versiones de flores



Versiones de setos



Fondo del menú



Sprite del personaje