Nombre del grupo	Bonfire Studio
Titulación	Software D y Computadores A
Grupo reducido	GR1
Número de grupo	03
Repositorio Github	https://github.com/alvarolh33/marcelo
Espacio Trello	https://trello.com/invite/b/28mBT12B/ATTI98661c3078d615b0004ebe57ae78b524CE17531A/bonfire-studio
Memoria del proyecto	https://docs.google.com/document/d/1ISR2ngOB4Au9SglrSnYng9WFSheibC2c-5RWcu3rXZA/edit?usp=sharing

Número	**Nombre**	**Apellidos**	**Correo UMA**
1	Álvaro	Leal Huertas	alealhuer995@uma.es
2	Yago	Salas Trujillo	yagosalast@uma.es
3	Ignacio	González González	ignaciogg@uma.es
4	Jesús	Quiñones de las Peñas	jesusqdlp2003@uma.es

Nuestro proyecto: *La mazmorra de Marcelo*

Nuestro equipo, al que hemos apodado *Bonfire Studio*, va a desarrollar un videojuego llamado *La mazmorra de Marcelo*. Al ser un videojuego, el principal objetivo es entretener al usuario. Queremos ofrecer un videojuego sencillo y al que sea fácil aficionarse.

Tenemos en mente tres perfiles de usuarios: el jugador, el padre o madre que controlará los controles parentales de un jugador menor y el creador de niveles.

Roles

- Product Manager: Yago y Jesús
- Project Manager: Álvaro e Ignacio
- Diseño audiovisual y de interfaces gráficas: Ignacio y Jesús
- Control del personaje y diseño de enemigos y escenarios: Álvaro y Yago

Gestión de riesgos

Riesgos tecnológicos

Limitaciones del motor gráfico elegido que den lugar a problemas de rendimiento en el juego.

- Probabilidad: baja, ya que el motor gráfico elegido es una herramienta con un gran número de usuarios y con capacidad para desarrollar juegos de magnitudes muy superiores a la de nuestro proyecto.
- Efectos: catastróficos, puesto que un problema causado por el motor requeriría un gran número de cambios a nuestro proyecto, ya que el código estaría estructurado acorde al motor elegido.
- Estrategia: en el caso de presentarse una limitación del motor que nos impida desarrollar el proyecto de acuerdo a nuestra visión, podríamos mitigar dicha limitación modificando el código fuente del motor, ya que es un motor de código abierto y, por tanto, se puede modificar libremente.

Riegos de personal

Falta de conocimientos técnicos necesarios para la implementación de la funcionalidad requerida para el desarrollo del proyecto.

- Probabilidad: Alta, ya que somos un grupo formado por estudiantes y no tenemos experiencia previa con el motor y el lenguaje de programación escogidos.
- Efectos: serios, ya que una falta de conocimientos podría enlentecer considerablemente el desarrollo.
- Estrategia: hacer uso de la documentación oficial ofrecida por el proyecto de Godot, donde se explica la funcionalidad completa del motor gráfico, así como

Riesgos de organización

Fallos en la comunicación entre los miembros del equipo debidos a una dificultad usando las herramientas de control de versiones como Git y Github debido a nuestra falta de experiencia.

- Probabilidad: moderada, ya que aunque nos falte experiencia no son herramientas difíciles de entender y usar.
- Efectos: tolerables, ya que la escala del proyecto no es tan grande como para que errores en el control de versiones supongan graves consecuencias.
- Estrategia: utilizar los recursos disponibles en internet para aprender la forma correcta de utilizar las herramientas de control de versiones, así como practicar lo aprendido en repositorios de prueba.

Riesgos de herramientas

Las herramientas escogidas no son eficientes o tienen *bugs*, tenemos problemas a la hora de trabajar con distintas herramientas de desarrollo, ya sea por problemas de compatibilidad de formatos o de integración con el motor.

- Probabilidad: muy baja, ya que la gran mayoría del desarrollo se hará dentro del propio motor gráfico. Las únicas partes del proyecto que se desarrollarán con herramientas ajenas al motor serán los *sprites* y sonidos.
- Efectos: insignificantes, ya que los formatos de imágen y vídeo gozan de una alta capacidad de conversión. Si alguna incompatibilidad se produjese, sería sencillo solventarla.
- Estrategia: para evitar bugs, usaremos versiones estables de las herramientas escogidas (en contraposición a las versiones *nightly*) y trataremos de limitar el número de herramientas esenciales para el desarrollo del proyecto a una cantidad razonable.

Riesgos de requisitos

Vamos a emplear un modelo incremental, de manera que los requisitos cambiarán a lo largo del desarrollo del proyecto. Esto puede llevar a que un nuevo requisito del proyecto sea incompatible o requiera efectuar grandes cambios a la estructura del proyecto.

- Probabilidad: moderada, ya que los requisitos pueden cambiar pero la visión global del proyecto será estable, es decir, no pretendemos cambiar la idea general del proyecto.
- Efectos: tolerables, ya que el motor elegido ofrece mucha flexibilidad a la hora de cambiar la estructura de un proyecto. El motor utiliza una metodología basada en clases y objetos, cosa que facilita una posible reestructuración de parte del proyecto
- Estrategia: llevar un control estricto sobre qué requisitos nuevos pueden incluirse en el proyecto y considerar detenidamente si su implementación colisiona de alguna manera con los sistemas ya implementados.

Riesgos de estimación

No hacemos una correcta valoración del tiempo necesario para desarrollar el proyecto. Invertimos más tiempo del estimado en labores que no contribuyen o contribuyen poco en la finalización del proyecto.

- Probabilidad: alta, ya que es el primer proyecto que desarrollamos en equipo y con las herramientas escogidas, es muy probable que nuestras estimaciones de tiempo sean erróneas.
- Efectos: serios, ya que tenemos una fecha de entrega con la que tenemos que cumplir.
- Estrategia: empezar el desarrollo del proyecto con suficiente tiempo de antelación para poder así acostumbrarnos a las nuevas herramientas y metodologías de trabajo, así como mantener bajo control la escala del proyecto, teniendo en cuenta el número de personas involucradas y la experiencia.

Planificación

Hemos decidido implementar un modelo de desarrollo incremental. Consideramos que es el modelo que mejor se ajusta al desarrollo de un videojuego, ya que la separación en versiones nos permite probar mecánicas de manera gradual y cambiar el diseño de aspectos del juego de acuerdo a nuestras impresiones al jugarlo.

El desarrollo estará basado en versiones iterativas, de manera que el proyecto se irá refinando de manera gradual. Cada sistema o mecánica del juego deberá tener una implementación inicial que se irá mejorando de manera iterativa.

Hemos decidido clasificar las tareas en cuatro grupos principales:

🟥 Interfaz de usuario

- Menú del juego: Un menú navegable por el usuario con un conjunto de opciones relativas a los gráficos, el audio y los controles parentales.
- Gestión de inventarios: Implementar una interfaz para la gestión de objetos y equipamiento por parte del jugador.
- HUD (Heads-Up Display): La implementación del HUD, la interfaz gráfica que puede ver el jugador durante el ciclo normal de juego. Muestra la vida del jugador, las monedas, la puntuación, el tiempo de juego, etc.

🟩 Controles del usuario

- Control del personaje: Desarrollar el sistema de control de movimiento y ataque del personaje.
- Control del menú: Sistema de navegación por las diferentes interfaces gráficas como los menús o el inventario.

- Navegación entre salas: Implementación de la navegación entre las diferentes salas y escenarios del juego, basado en puertas que llevan de un escenario a otro.

🟦 Diseño de mecánicas

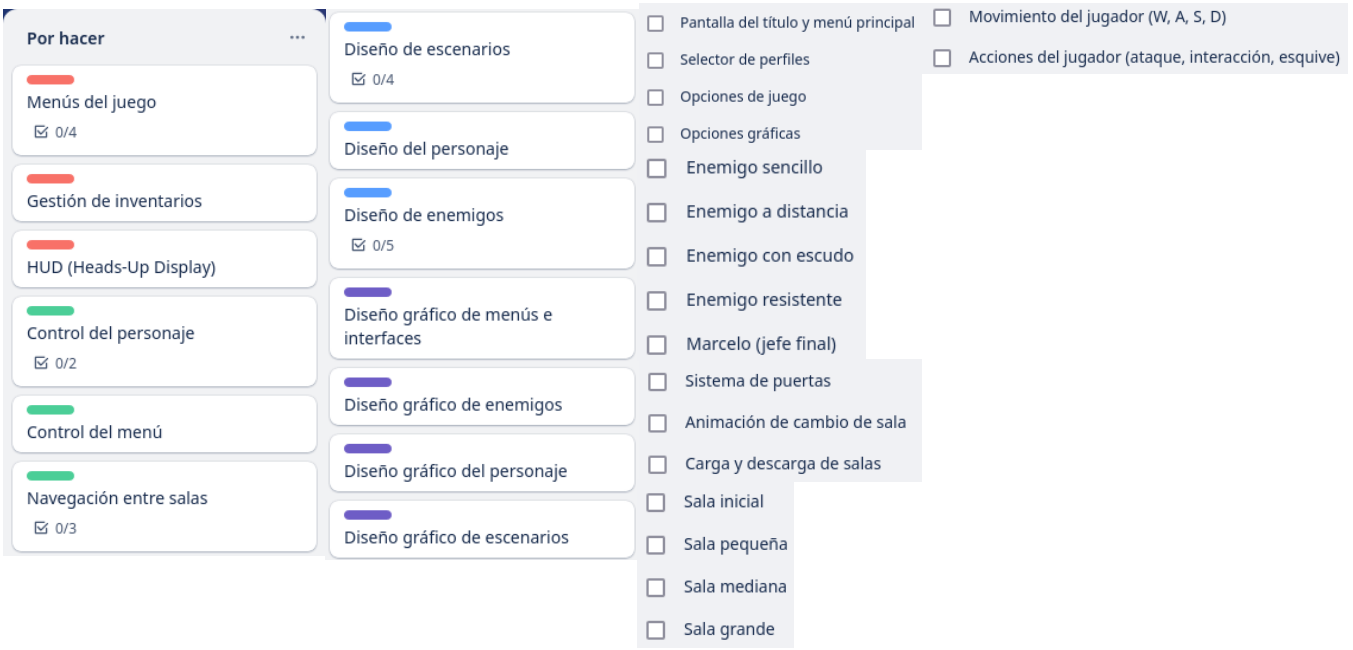
- Diseño de niveles y escenarios: La distribución de salas, tipos de salas, y su contenido, así como la progresión del jugador.
- Diseño del personaje principal: El desarrollo de su diseño mecánico, como lo serían sus diferentes habilidades, poderes o características del mismo.
- Diseño de enemigos: Diseño de su dificultad, formas de derrotarlos y principalmente su función de obstáculo ante el jugador.

🟪 Diseños gráficos

Esta categoría incluye el desarrollo y diseño de los diferentes sprites y elementos gráficos o visuales requeridos. Estos serían los diseños gráficos de:

- Menús e interfaces
- Enemigos
- Personaje
- Escenarios y gráficos (e.g. partículas, efectos visuales, etc.)

Planificación en Trello



Herramientas usadas para el desarrollo del software:

- Godot: Es el motor gráfico elegido para desarrollar nuestro videojuego. Utiliza un sistema de escenas y objetos con scripts asociados, los cuales usan un lenguaje de *scripting* propio llamado GDScript. Es un motor de código abierto, bajo la licencia permisiva MIT.
- Aseprite y Gimp: Son las herramientas escogidas para el desarrollo de los sprites y gráficos de nuestro juego.
- Git y Github: Hemos escogido Git como software de control de versiones de nuestro proyecto y el repositorio será *hosteado* en GitHub.
- Discord: Es el medio de comunicación informal que vamos a usar cuando queramos hablar acerca del proyecto.
- Trello: Vamos a usar Trello como forma de organizar nuestras tareas pendientes y en proceso.