DOCUMENTO FUNCIONAL DEL PROYECTO

ALUMNO/GRUPO: **Shaim y Jansen**

---

1. Introducción y contexto

Objetivo: Se trata de un juego 2D

Describe:

- El problema o necesidad que se quiere resolver.

  En el mercado actual de videojuegos existen pocas opciones accesibles que combinan entretenimiento, agilidad mental y reto progresivo sin requerir conocimientos técnicos previos por parte del jugador. La mayoría de juegos casuales carecen de un sistema de desafío real que mantenga al usuario motivado a lo largo del tiempo. Existe una necesidad de proporcionar una experiencia de juego simple en su aprendizaje pero estimulante en su ejecución, donde cada partida suponga un reto genuino para el jugador.

- Quién será el usuario o cliente final.

  El juego está dirigido al público general, sin restricción de edad. Tanto jugadores ocasionales que buscan una experiencia rápida y entretenida, como jugadores más competitivos que quieran superar sus propios récords de tiempo, son el perfil objetivo.

- Qué solución se propone y con qué propósito.

  Se propone el desarrollo y publicación de un videojuego 2D de plataformas y recolección en el que el jugador debe recoger todas las monedas de cada nivel para avanzar al siguiente, evitando trampas que pondrán a prueba sus reflejos y capacidad de planificación. El juego incorpora un contador de tiempo que añade una capa adicional de tensión y rejugabilidad, incentivando al jugador a mejorar su rendimiento en cada intento.

2. Análisis de requisitos

2.1. Requisitos funcionales (RF)

Qué debe hacer el sistema.
Enumera las funciones principales, numeradas como RF1, RF2, etc.

Código	Descripción del requisito funcional
RF1	El juego requerirá conexión a internet para poderse jugar en el navegador si no quieren descargarlo.
RF2	El juego registrará automáticamente los datos de cada partida (monedas recogidas y nivel alcanzado)
RF3	El juego dispondrá de un contador de tiempo visible durante la partida

2.2. Requisitos no funcionales (RNF)

Cómo debe comportarse el sistema.
Incluye aspectos como rendimiento, seguridad, compatibilidad o facilidad de uso.

Código	Descripción del requisito no funcional
RNF1	El juego no será compatible con tablets o móviles a menos que uses un teclado conectado con un adaptador
RNF2	El juego deberá ser compatible con los principales navegadores o sistemas operativos en su versión de distribución

2.3. Restricciones

Condiciones o limitaciones del proyecto.

- Lenguajes o tecnologías obligatorias:

  GDScript,Godot Engine, Github, HTML para subir el juego en el navegador y itch.io, pagina para encontrar los assets

- Recursos disponibles (tiempo, equipo, materiales):

  Pues se necesita 6 o 7 meses para acabar el juego, tener un repositorio de GitHub y ordenador con Godot instalado (la ultima version), y tener el Visual Studio Code que nos permite hacer la página web

- Dependencias o limitaciones técnicas:

  Dependencias:

  Godot Engine 4.6: El juego depende directamente de la versión 4 del motor.

  Assets externos gratuitos: El juego utiliza recursos gráficos y de audio obtenidos de plataformas gratuitas como en itch.io.

  Limitaciones técnicas:

  Compatibilidad de plataformas: El juego está diseñado y optimizado exclusivamente para PC con Windows y Linux. No se garantiza su correcto funcionamiento en macOS, dispositivos móviles o consolas.

  Sin soporte multijugador en tiempo real: El juego es de carácter individual.

---

3. Análisis de usuarios y roles

Objetivo: identificar quién usará el sistema y qué podrá hacer.

Describe los distintos tipos de usuario, sus necesidades y sus permisos.

Rol	Descripción	Permisos principales
Administrador	Persona responsable de gestionar y mantener el sistema del juego, incluyendo usuarios, niveles y datos de partidas.	Puede crear, modificar y eliminar usuarios, niveles, registros de partidas y configuraciones del sistema.
Usuario	Cualquier persona que accede y juega sin necesidad de registro ni cuenta.	Puede jugar todos los niveles, ver el contador de tiempo, recoger monedas y consultar el ranking global.
Tester	Persona encargada de probar el juego para detectar errores, bugs o desequilibrios en la dificultad de los niveles.	Puede acceder a todos los niveles en cualquier estado (incluso no publicados), registrar errores y reportar fallos, pero no puede modificar nada.

---

4. Casos de uso / Escenarios de uso

Objetivo: mostrar cómo interactúan los usuarios con el sistema.

Selecciona de tres a cinco casos principales y descríbelos brevemente.

Código	Nombre del caso de uso	Actor principal	Descripción	Resultado esperado
CU1	Iniciar partida.	Jugador	El jugador abre el juego y pulsa el botón de jugar para comenzar el primer nivel.	El juego carga el nivel 1 con las monedas, trampas y el contador de tiempo activo.
CU2	Recoger monedas y avanzar de nivel	Jugador	El jugador recorre el nivel recogiendo todas las monedas mientras evita las trampas.	Al recoger la última moneda, el juego desbloquea y carga el siguiente nivel automáticamente.

CU3	Morir y reiniciar nivel	Jugador	El jugador toca una trampa y pierde la vida, provocando el fin de la partida en ese intento.	El juego muestra una pantalla de misión fallida y ofrece al jugador la opción de reiniciar el nivel desde el principio.

---

5. Modelo de datos o estructura de la información

Entidades principales:

Jugador: guarda el estado actual del jugador durante la partida, como el nivel en el que se encuentra, las monedas recogidas hasta el momento.

Moneda: representa cada objeto coleccionable que el jugador debe recoger para completar el nivel. Tiene una posición fija dentro del mapa y un estado (recogida o no recogida).

Trampa: representa los obstáculos estáticos del nivel que eliminan al jugador al contacto. Tiene un tipo (pinchos, fuego, sierra, etc.) .

Enemigo: guarda las características de cada enemigo móvil del nivel, como su tipo, velocidad de movimiento, patrón de desplazamiento y el daño que causa al jugador

Relaciones:

Un jugador puede jugar varios niveles a lo largo de una sesión de juego.

Cada nivel contiene un conjunto de monedas, trampas estáticas y enemigos móviles.

El jugador recoge monedas para completar el nivel y desbloquear el siguiente.

Un nivel puede contener uno o varios enemigos, cada uno con su propio patrón de movimiento.

6. Diseño de la interfaz

Pantallas principales del juego:

- **Pantalla de inicio / menú principal**

  Función:Tiene 3 opciones (ver desafío, jugar y salir), la pantalla de inicio es el personaje del juego sosteniendo a los enemigos como títeres.

- **Pantalla de juego (nivel)**

  Función: pantalla principal donde transcurre la acción. Muestra el mapa del nivel, el personaje, las monedas, las trampas, los enemigos y el contador de tiempo en pantalla. Casos de uso relacionados: CU2 (Recoger monedas y avanzar de nivel), CU3 (Morir y reiniciar nivel).

- **Pantalla de pausa**

  Función: permite al jugador pausar la partida en cualquier momento, con opciones para salir, reiniciar el nivel o volver al menú principal, para continuar deberás de clicar de nuevo la tecla del menú pausa qué es “esc”.

- **Pantalla de resultados / fin de partida**

  Función: enseña la puntuación final, monedas obtenidas y botón para reiniciar o volver al menú.

- **Pantalla de resultados / siguiente nivel**

  Función: enseña la puntuación una vez pasado el nivel con un mensaje de “misión completada” y abajo el resultado de las monedas recogidas

7. Planificación técnica

Lenguajes y frameworks:

- Godot Engine (GDScript): desarrollo del juego y su lógica principal.
- HTML: el juego se ha subido al navegador gracias al lenguaje de HTML
- GitHub: control de versiones y almacenamiento del proyecto.

Herramientas de diseño o edición:

- Programa Gnu: Nos ha permitido editar al personaje y colorear para que se identifique bien en los diferentes frames

Reparto de tareas:

- Shaim: programación de todos los assets, creación de todos los objetivo, trampas, enemigos y personajes, creación del mapa entero, sistema de daño y sonido, menú principal y pausa, contador de tiempo y animaciones para el personaje y los assets, y por último creacion de pagina web y subir el juego en 2 plataformas + edicion de pagina del itch.io + memoria juego
- Jansen: tester + reworks de los niveles + memoria del juego + Pruebas de jugabilidad y dificultad de los niveles + Ideas para nuevas mecánicas + Control de los bugs general del proyecto y seguimiento + Pruebas finales y grabación del gameplay del videojuego y editarlo + buscar canciones de algunos niveles + Verificación de que no existan objetos atravesables o errores de mapa. (Esto es de momento)

Fase	Descripción	Duración estimada
Octubre - Noviembre	Diseño inicial, propuesta y estructura base del juego	2 meses
Diciembre - Febrero	Programación de niveles, enemigos, sistema de monedas y jugador	3 meses
Marzo - Abril	Crear varios niveles para no verse aburrido y aumentando la dificultad	2 meses
Mayo	Finalización, presentación y entrega	1 mes

8. Análisis de riesgos

Objetivo: identificar posibles problemas y cómo se afrontarán.*

8.1. Identificación de riesgos

Posibles riesgos:

- Falta de tiempo por acumulación de tareas: intentaremos encontrar más tiempo de lo que tenemos acabando tareas del instituto lo antes posible
- Problemas técnicos con Godot o errores en el script: problemas que nos daría el script y podría romperse la estructura, usaremos IA si no entendemos el problema.
- Pérdida del progreso del proyecto o archivos dañados: Buscaremos los assets necesarios si perdemos los archivos
- Desmotivación o abandono de un miembro del grupo: 1 tendría que hacerlo todo

8.2. Valoración y respuesta

Clasifica cada riesgo según su probabilidad e impacto, e indica cómo se mitigará.

Riesgo	Probabilidad	Impacto	Plan de prevención o contingencia
Falta de tiempo	Alta	Alta	Repartir tareas y hacer entregas parciales cada semana.
Problemas técnicos	Media	Media	Buscar soluciones en foros de Godot y hacer pruebas frecuentes.
Pérdida de datos	Baja	Alta	Usar GitHub y guardar copias en la nube.
Desmotivación del grupo	Media	Alta	Mantener comunicación y repartir el trabajo de forma equilibrada.
Cambios en los requisitos	Media	Media	Adaptar el código modularmente para facilitar modificaciones.

---

9. Validación y criterios de éxito

Objetivo: definir cómo sabremos que el proyecto funciona correctamente.*

Criterios de aceptación:

1. El juego se ejecuta correctamente en PC con Windows y Linux sin errores de inicio.
2. El personaje puede moverse, saltar y recoger monedas sin errores ni comportamientos inesperados.
3. Al recoger todas las monedas del nivel, el juego carga el siguiente nivel automáticamente.
4. El contacto con una trampa o enemigo provoca la misión fallida y permite reiniciar el nivel correctamente.
5. El contador de tiempo funciona con precisión y se reinicia al comenzar un nuevo intento.
6. Los niveles aumentan su dificultad progresivamente en cuanto a disposición de trampas y comportamiento de enemigos.

Pruebas previstas:

- **Pruebas funcionales:** comprobar que todos los botones y acciones del juego funcionan correctamente, incluyendo el movimiento del personaje, la recogida de monedas, la detección de colisiones con trampas y enemigos, el cambio de nivel y el reinicio tras la misión fallida.

- **Pruebas de usuario:** sesiones de testeo con compañeros del instituto para evaluar la jugabilidad, la dificultad de los niveles y la experiencia general del jugador, recogiendo feedback para realizar ajustes.

- **Pruebas de compatibilidad:** ejecutar el juego en diferentes equipos con Windows y Linux para asegurar que el ejecutable funciona correctamente en distintas configuraciones de hardware.

Indicadores de calidad:

- Juego fluido y sin lag.

- Interfaz clara y accesible.

- Diversión y satisfacción del jugador (medida con encuestas).

---

10. Conclusión

Objetivo: cerrar el análisis y preparar la siguiente fase.*

Durante esta fase del proyecto se han definido los aspectos más importantes de *Collect The Coins*:

- Funciones principales: movimiento del personaje, recolección de monedas, mejora de habilidades y progresión de niveles.
- Tecnologías utilizadas: Godot con GDScript, Python, GitHub y programas de diseño gráfico.
- Valor del proyecto: crear un videojuego sencillo, divertido y desafiante que permita a los jugadores disfrutar mientras nosotros aprendemos programación y trabajo en equipo.

Próximos pasos:

- Configurar el entorno de desarrollo en Godot.
- Crear el repositorio en GitHub y la estructura básica del proyecto.
- Empezar la implementación de los niveles y el sistema de monedas.

Reflexión final:

El análisis funcional nos ha ayudado a planificar bien el proyecto y a tener una visión clara de lo que queremos lograr. A partir de aquí comienza la fase de desarrollo, donde nuestro objetivo será convertir la idea en un videojuego real y jugable.