Álvaro Fernández Ferriz Steven Hernan Soriano Pérez

Institut Puig Castellar
Projecte / Crèdit de síntesi
Document funcional

DOCUMENTO FUNCIONAL DEL PROYECTO

ALUMNO/GRUPO: Alvaro Fernández y Steven Soriano

1. Introducción y contexto

Objetivo: explicar de qué trata el proyecto y por qué es necesario.

Fruit Quest, es un videojuego 2D desenvolupado con Godot y programado con GDscript. El protagonista es una especie de gato, que tiene que recoger frutas a lo largo de los diferentes niveles para obtener puntos y superar los niveles que cada vez son más complicados.

Describe:

- **El problema o necesidad que se quiere resolver.**
  Los jugadores actualmente buscan experiencias de juego accesibles y entretenidos, como los juegos clásicos pero con mecanismos modernos. Muchos juegos de ahora, son complejos o necesitan un tiempo para aprender a jugar.

  Eso es lo que no queremos con Fruit Quest, ya que resolvemos estas necesidades haciendo el juego sencillo pero requiriendo habilidad, rapidez y reflejos.

- **Quién será el usuario o cliente final.**
  Los usuarios serán jugadores a partir de 8 años, casuales y que sean entusiastas de juegos retro de plataformas. El juego será en PC, en cualquier sistema operativo.

- **Qué solución se propone y con qué propósito.**
  La solución es desenvolupar un videojuego 2D funcional con estas características:
  - **Mecanismos de movimientos fluidos:** desplazamiento y sistema de saltos.
  - **Sistema de recogida y puntuación:** frutas coleccionables que den puntos.
  - **Múltiples niveles:** mínimo 4-5 niveles con dificultad progresiva.
  - **Estética pixel art:** Gráficos sencillos pero coherentes y atractivos.
  - **Experiencia sonora:** Música de fondo y efectos de sonido.
  - **Interficie intuitiva:** Menús claros y HUD informativo.

Álvaro Fernández Ferriz Steven Hernan Soriano Pérez

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 jugador podrá mover al personaje hacia la izquierda y la derecha utilizando las teclas de dirección.
RF2	El jugador podrá hacer saltar al personaje utilizando la barra espaciadora.
RF3	El personaje colisionará correctamente con plataformas, paredes, techos y obstáculos.
RF4	El sistema permitirá recoger frutas al tocarlas, eliminándolas del mapa y sumando puntos.
RF5	El juego mostrará en pantalla la puntuación actual del jugador en tiempo real.
RF6	El juego contendrá al menos 3-5 niveles con dificultad progresiva.
RF7	Al completar un nivel, el sistema avanzará automáticamente al siguiente nivel.
RF8	El juego reproducirá efectos de sonido en acciones clave: saltar, recoger frutas, colisionar con obstáculos.
RF9	El juego incluirá música de fondo durante las partidas.
RF10	Existirá un menú principal con opciones para iniciar partida, consultar controles y salir del juego.
RF11	El juego mostrará una pantalla de "Nivel Completado" con la puntuación obtenida.
RF12	El jugador podrá pausar el juego en cualquier momento con la tecla ESC.

RF13	El sistema guardará las mejores puntuaciones de cada jugador en una base de datos.
RF14	El sistema guardará el progreso del jugador para poder continuar posteriormente la partida.
RF15	El juego mostrará un ranking con las 10 mejores puntuaciones históricas.
RF16	El jugador podrá introducir su nombre antes de empezar a jugar.

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 se ejecutará a un mínimo de 60 FPS en hardware básico (procesadores Intel i5/AMD Ryzen 3 o superior).
RNF2	La interfaz será intuitiva y no requerirá tutoriales extensos para comprender las mecánicas básicas.
RNF3	Los gráficos mantendrán un estilo pixel art coherente en todos los elementos visuales (personaje, frutas, plataformas, fondos).
RNF4	El juego será completamente jugable con teclado sin necesidad de ratón.
RNF5	Los tiempos de carga entre niveles no superarán los 2 segundos.
RNF6	El código estará comentado y organizado para facilitar el mantenimiento y posibles ampliaciones futuras.
RNF7	El juego será accesible para jugadores sin experiencia previa en videojuegos de plataformas.

2.3. Restricciones

Condiciones o limitaciones del proyecto.

- Lenguajes o tecnologías obligatorias:
  - Motor del juego: Godot con soporte para GDscript
  - Lenguaje de programación: Gdscript
  - Control de versiones: Github
  - Entorno de desenvolupamiento: Visual Studio Code o editor integrado de Godot.

- Recursos disponibles (tiempo, equipo, materiales):
  - **Tiempo total:** Octubre - Mayo = 135 horas.
    - Octubre-Enero: 3 horas semanales
    - Febrero-Mayo: 6 horas semanales
  - **Equipo**: 2 personas
  - **Material**: Ordenadores del centro y personales cuando avancemos en casa.
- Dependencias o limitaciones técnicas:
  - Sprites gratuitos de plataformas como itch.io, OpenGameArt o Kenney.nl.
  - Musica y efectos de sonido libres de derechos: Freesound.org, OpenGameArt, Incompetech.
  - El juego tiene que ser ejecutable sin instalación compleja.

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
jugador	Usuario que juega a Fruit Quest en su ordenador personal. No hace falta registro, es accesible sin barreras de entrada.	Controla el personaje, navega por los menús, inicia y pausa las partidas, puede consultar la puntuación.
desenvolupador	Alvaro y Steven, los creadores del juego.	Acceso completo al código fuente, diseño y modificación de niveles, configuración de parámetros del juego, pruebas y depuración.

Álvaro Fernández Ferriz Steven Hernan Soriano Pérez

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	juego de inicio	jugador	El jugador abre el juego y selecciona “jugar” al menú principal.	El juego carga el primer nivel y comienza la partida.
CU2	mover personaje	jugador	El jugador utiliza las teclas de dirección para mover al personaje.	El personaje se mueve suavemente a la dirección indicada con animaciones fluidas.
CU3	saltar	jugador	El jugador presiona la barra de espacio para que el gato salte.	El personaje salta y cae con realismo.
CU4	Recoger fruta	jugador	El jugador mueve al gato hasta tocar una fruta.	La fruta desaparece, suena un efecto de sonido y se suman puntos en el marcador.
CU5	Completar nivel	jugador	El jugador llega al final del nivel (punto de meta o recogiendo todas las frutas).	Aparece una pantalla de "Nivel Completado" con la puntuación y la opción de avanzar.
CU6	Pausar juego	jugador	El jugador pulsa ESC durante la partida.	El juego se pausa y aparece un menú con opciones (continuar, reiniciar, menú principal).

Álvaro Fernández Ferriz Steven Hernan Soriano Pérez

5. Modelo de datos o estructura de la información

1. El Jugador

Que necesitamos saber sobre el jugador?

- ★ **Dónde está?** → Posición en el mapa (coordenadas X, Y)
- ★ **A que velocidad se mueve?** → Velocidad de movimiento
- ★ **Está saltando o está en el suelo?** → Si toca una plataforma o está en el aire
- ★ **Cuantos puntos lleva?** → Puntuación actual
- ★ **Cuántas vidas tiene?** → Número de vidas

2. Las Frutas

Què necesito saber de cada fruta?

- ★ **Dónde está?** → Posición en el mapa
- ★ **Que tipo de fruta es?** → Cereza, manzana…
- ★ **Cuantos puntos vale?** → Puntuación que da cada fruta.
- ★ **Asegurarse que se ha recogido** → Si el jugador ya lo ha cogido o sigue estando en el mapa

3. Las Plataformas

Qué necesito saber de cada plataforma?

- ★ **Dónde está?** → Posición en el mapa
- ★ **Cuál es su medida?** → Ancho/Alto
- ★ **Se puede atravesar? o se tiene que saltar?** → El jugador puede caminar a través o se tiene que saltar.
- ★ **Qué tipo es?** → Normal, movil, trampa

4. Los niveles

Qué necesitamos saber de cada nivel?

- ★ **Qué número de nivel es?** → Nivel 1, 2, 3…
- ★ **Cuántas plataformas tiene?** → Mirar antes de todo cuantas plataformas encontramos en el mapa.
- ★ **Cuántas frutas tiene?** → Antes de comenzar el nivel, fijarse en el nº de frutas.
- ★ **Qué dificultad tiene el nivel?** → Fácil, medio o difícil

5. El gestor del juego

Qué necesitamos saber sobre el estado general del juego?

- ★ **En qué nivel estamos?** → Nivel actual
- ★ **Saber nuestra puntuación total?** → Suma de todos los puntos que hemos acumulado
- ★ **Qué está haciendo el jugador?** → **Es**tá en el menú, está jugando, ha perdido...
- ★ **Cuál es el récord del juego?** → Máxima puntuación conseguida en ese mapa

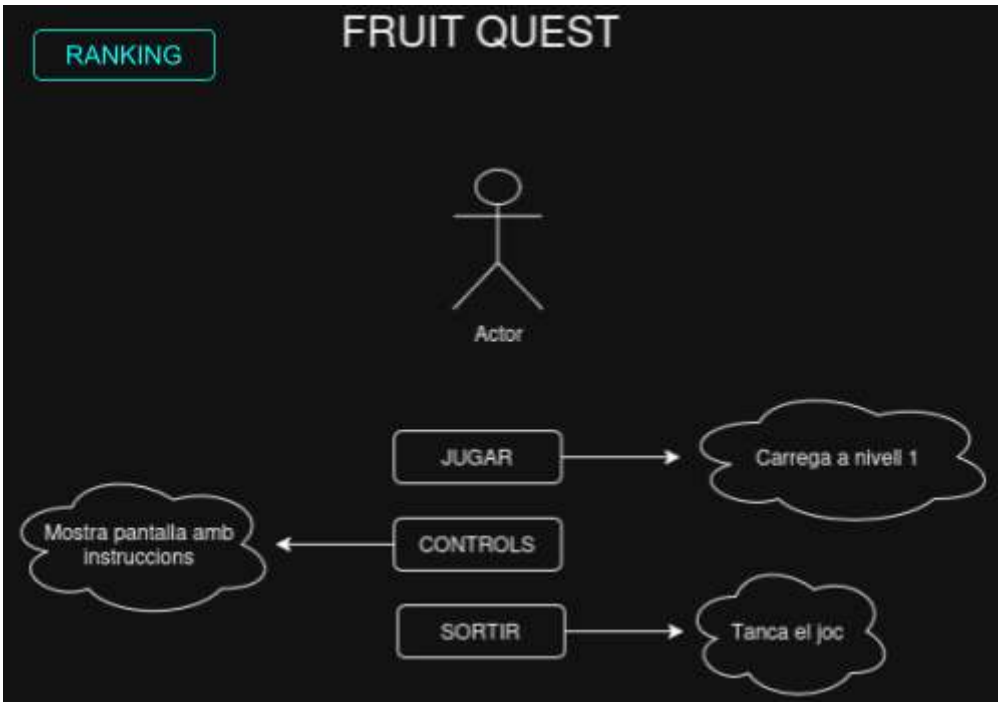
6. Diseño de la interfaz

Objetivo: visualizar la estructura y navegación del sistema antes de desarrollarlo.*

Incluye bocetos o capturas de las pantallas principales y una breve descripción de su función.

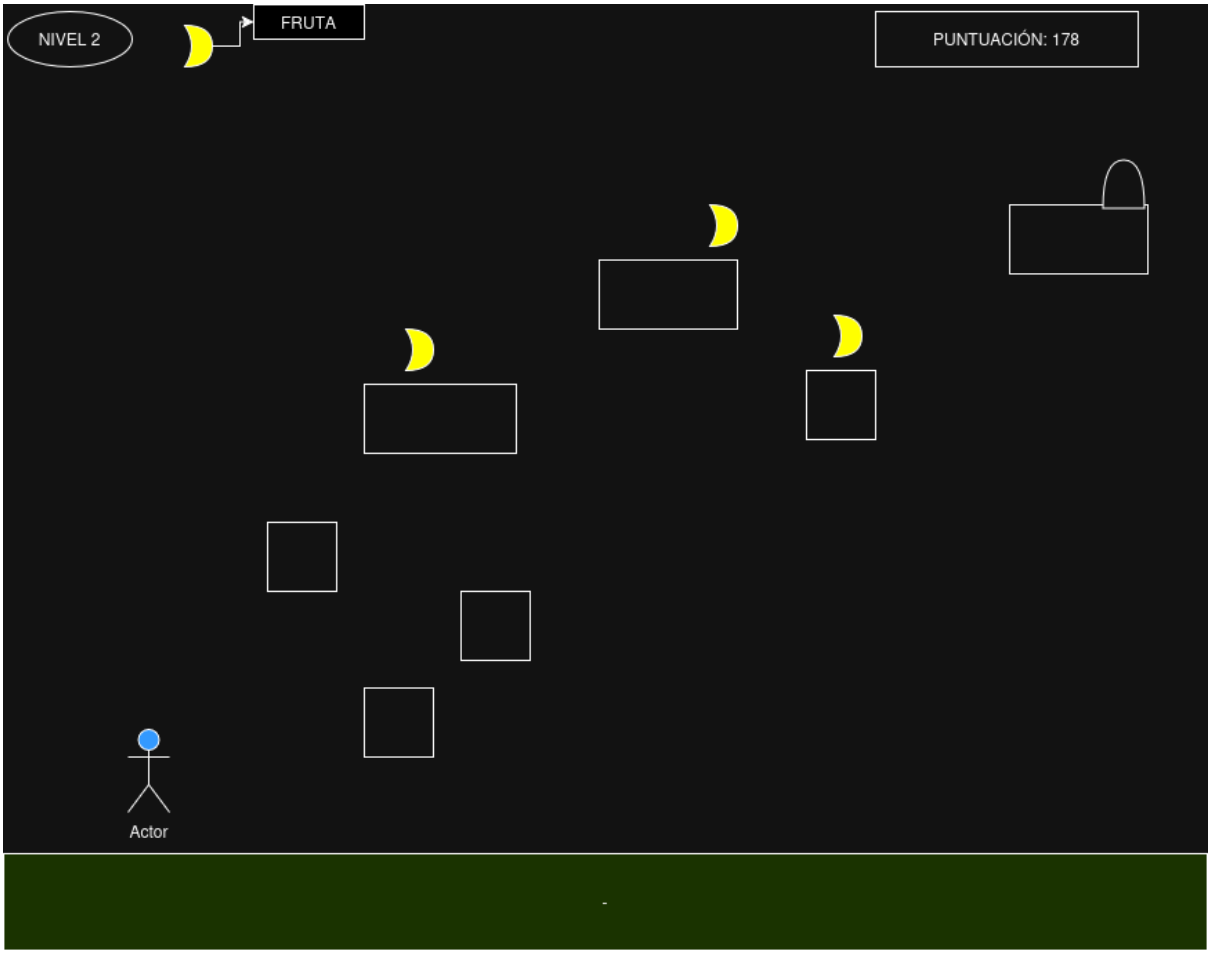
Para cada pantalla, indica:

- PANTALLA 1:
- Funcionalidad principal: Pantalla d'inici de joc
- Casos de uso relacionados: CU1



Álvaro Fernández Ferriz Steven Hernan Soriano Pérez

- PANTALLA 2:
- Funcionalidad principal: Interficie visible durante la partida.
- Casos de uso relacionados: CU2, CU3, CU4, CU5



- PANTALLA 3:
- Funcionalidad principal: Menú que aparece al presionar ESC
- Casos de uso relacionados: CU6

Álvaro Fernández Ferriz Steven Hernan Soriano Pérez

- PANTALLA 4:
- Funcionalidad principal: Aparece en completar un nivel
- Casos de uso relacionados: CU5

- PANTALLA 5:
- Funcionalidad principal: Aparece SI EL JUGADOR PIERDE TODAS LAS VIDAS
- Casos de uso relacionados:

Álvaro Fernández Ferriz Steven Hernan Soriano Pérez

7. Planificación técnica

Objetivo: planificar el desarrollo del proyecto.*

Indica las tecnologías y herramientas que se utilizarán, y cómo se organizará el trabajo.

- Tecnologías seleccionadas:

- Motor de juego: Godot Engine
- Lenguaje de programación: Gdscript
- Control de versiones: GitHub
- Editor de código: Visual Studio Code
- Editor de audio: godot

Porque hemos elegido Godot?

- Motor de código abierto y gratuito
- Diseñado específicamente para juegos 2D
- Soporte nativo para C#
- Multiplataforma
- Ligero y rápido de compilar

Porque hemos elegido lenguaje C#?

- Lenguaje moderno y potente
- Fácil de aprender si no lo entiendes
- Buena integración con Godot, nuestro programa para crear el juego

Álvaro Fernández Ferriz Steven Hernan Soriano Pérez

Cronograma de nuestro proyecto: Lo hemos hecho todo juntos, sin dejar ningún apartado para hacerlo individualmente, ya que tenemos que saber ambos como hacemos cada cosa, para luego poder defenderla.

Investigación de Godot i C#	Álvaro Y Steven	Octubre
Diseño de niveles en papel/digital	Álvaro Y Steven	Noviembre-Diciembre
Selección y adaptación de sprites	Álvaro Y Steven	Noviembre
Programación de mecánicas básicas de movimiento	Álvaro Y Steven	Enero-Febrero
Programación del sistema de frutas y puntuación	Álvaro Y Steven	Febrero
Implementación de niveles en Godot	Álvaro Y Steven	Marzo
Implementación de efectos de sonido	Álvaro Y Steven	Marzo
Pruebas y corrección de errores	Álvaro Y Steven	Abril
Redacción de la memoria	Álvaro Y Steven	Abril
Preparación de la presentación y defensa	Álvaro Y Steven	Mayo

Álvaro Fernández Ferriz Steven Hernan Soriano Pérez

8. Análisis de riesgos

Objetivo: identificar posibles problemas y cómo se afrontarán.*

8.1. Identificación de riesgos

Ejemplos:

- Falta de tiempo o mala planificación.
- Problemas técnicos con godot o Gdscript.
- Dificultad de comunicación entre los miembros.
- Perdida de datos.
- Cambios en los requisitos.
- Dificultad para encontrar sprites y sonidos coherentes
- Incompatibilidad entre versiones de Godot
- Bugs críticos fuera del juego

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	grave	Dividir tareas y fijar entregas intermedias.
Problemas técnicos con godot o GDscript.	Media	Media	Probar tecnologías antes de programar.
Dificultad de comunicación entre los miembros.	Baja	Alta	Reuniones presenciales o virtualmente.
Perdida de datos.	Baja	grave	Hacer copias de seguridad semanales en github, hacer una sincronicación con drive.
Cambios en los requisitos.	Media	grave	Validar constantemente con el profesor, utilizar metodología ágil que permita pequeños ajustes sin desestabilizar el proyecto.

Dificultad para encontrar sprites y sonidos coherentes	Media	Baja	Buscar diferentes recursos desde el primer mes, así tener alternativas más preparadas.
Incompatibilidad entre versiones de Godot	Baja	Media	Utilizar la misma versión de godot en los dos ordenadores, asi documentar la versión exacta utilizada.
Bugs críticos fuera del juego	Alta	Media	Hacer pruebas constantes depsues de cada cambio, mantener el codigo limpio y comentado.

---

9. Validación y criterios de éxito

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

Criterios de aceptación

Nuestro proyecto se considerará que ha salido bien, si los siguientes requisitos se cumplen:

Funcionalidad completa

- Todos los requisitos funcionales críticos RF1-RF12
- El jugador puede mover y hacer saltar al personaje sin ningún problema
- Las colisiones funcionan correctamente, el personaje no atraviesa plataformas ni objetos
- Se pueden recoger frutas y la puntuación se actualiza en tiempo real cuando la recoges
- Existen al menos 2-3 niveles jugables con dificultad diferente

Calidad técnica

- No hay errores críticos, como puede ser un crash en medio de la partida
- Los tiempos de carga no són muy grandes, si no al contrario
- El código está comentado y organizado para que se pueda entender un mínimo

Experiencia de usuario

- El menú principal y las transiciones entre niveles funcionan correctamente, sin ningún error
- El juego tiene efectos de sonido al recoger fruta o otras cosas
- La interfaz es clara y informativa, un HUD con la puntuación que llevamos
- Los controles son muy intuitivos y responden correctamente a su función

Álvaro Fernández Ferriz Steven Hernan Soriano Pérez

Documentación

- Memoria del proyecto completada y bien redactada
- El código estará disponible en un repositorio de GitHub.
- README en Github con instrucciones de instalación y uso del juego.
- Y por último, tener la presentación bien preparada para poder defender nuestro trabajo bien.

Pruebas previstas para saber que todo está correcto:

- Probar cada mecanica individualmente
- Probar cada nivel completo des del inicio hasta el final
- Verificar las transiciones entre niveles
- Comprobar el funcionamiento del menú

Pruebas de usabilidad con usuarios

- Pedir a algún compañero de clase que pruebe el juego
- Mirar si entienden bien los controles sin falta de explicarlos
- Recoger feedback sobre las dificultad de los niveles, etc.
- Identificar partes que pueden llegar a ser muy difíciles o confusas para la gente.

Pruebas de rendimiento

- Verificar que el juego funcione estable, sin que se pare y que todo funcione correctamente
- Comprobar el tiempo de carga entre niveles
- Probar la estabilidad en sesiones largas de juego

Pruebas de regreso

- Después de hacer algún cambio importante, volver a probar funcionalidades que previamente funcionaban
- Asegurarse que los cambios que hagamos, no provoquen nuevos errores

10. Conclusión

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

Resume las decisiones principales tomadas durante el análisis:

- Qué funciones tendrá el sistema.

  Movimiento del personaje, sistema de saltos, recogida de frutas, niveles múltiples con dificultad progresiva, sistema de puntuación y experiencia audiovisual completa.

- Qué tecnologías se utilizarán.

  Godot Engine con GDscript, control de versiones con GitHub, sprites, pixel art y audio libre de derechos.

- Qué valor aporta el proyecto.
  Aprender a desemvolupar videojuegos con herramientas profesionales, trabajar en equipo, crear un producto divertido y funcional, desemvolupar habilidades de programación y diseño.

Durante la elaboración de este documento funcional, hemos identificado varios riesgos con potencial, ajustar objetivos para hacerlos más realistas, aprender a planificar un proyecto complejo de manera estructurada y comprendido la importancia de la documentación antes de programar.

Lo vamos a hacer por diferentes fases

- Fase 1:
  - Instalar y configurar Godot con soporte C#.
  - Crear repositorio de GitHub y configurar Git.
  - Ver/completar códigos básicos de C# a Godot.
- Fase 2:
  - Crear la escena principal del juego.
  - Implementar los movimientos básicos del personaje.
  - Añadir plataforma y crear la primera fruta.
- Fase 3:
  - Seguir el cronograma hecho en el diagrama de Gantt.
  - Hacer un seguimiento diario utilizando Trello.
  - Realizar varias reuniones para ver el seguimiento y poder seguir avanzando.
- Fase 4:
  - Hacer una corrección de lo que no se ha podido implementar.
  - Redactar la memoria.
  - Empezar a preparar la presentación.
  - Ir haciendo ensayos de la presentación para ir seguros.

Ahora tenemos lo necesario para comenzar el desenrollamiento del proyecto con confianza. A partir de aquí comienza la creación de Fruit Quest

Álvaro Fernández Ferriz Steven Hernan Soriano Pérez