2025/2026

Jarein López Ruiz, Miguel de Pedro Olagaray

Universidad de la rioja

FactoryFall

Profesión de ingeniero en informática

**Resumen**

*FactoryFall* es un videojuego de plataformas 2D desarrollado con el motor **Godot** durante un período aproximado de diez semanas, como proyecto realizado por dos integrantes, para la asignatura de **Profesión de ingeniero en informática**. El juego consta de **tres niveles** progresivos en los que el jugador controla a un robot equipado con un jetpack, cuya mecánica de funcionamiento simula el sistema de salto clásico de juegos como *Mario Bros*.

El objetivo principal del juego es recorrer y superar los distintos niveles del mundo de *FactoryFall*, enfrentándose a escenarios gradualmente más difíciles a la par que más deteriorados. Cada nivel representa un entorno cada vez más afectado por la **contaminación provocada por una fábrica en ruinas**, siendo el segundo nivel el interior de dicha fábrica. Esta progresión se refleja visualmente en el cambio del color del cielo, que pasa de tonos azules a amarillentos y finalmente verdosos, transmitiendo el detrimento del entorno.

Además del apartado visual, el juego utiliza la **música como elemento narrativo**, adaptando la banda sonora de cada nivel para reflejar un tono cada vez más triste y opresivo, en concordancia con el aumento de la dificultad y el nivel de contaminación del mundo. De esta manera, *FactoryFall* busca combinar mecánicas clásicas de plataformas con una ambientación coherente que explica la historia del personaje y refuerza la experiencia del jugador.

Edificio con letrero en frente y tienda al lado

El contenido generado por IA puede ser incorrecto.

Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.**Introducción**

Este proyecto ha sido efectuado como trabajo final de la asignatura **Profesión de ingeniero en informática**. El desarrollo del videojuego se llevó a cabo a lo largo de aproximadamente 10 horas de trabajo (1 hora por semana).

Desde el inicio, se planteó la creación de un **videojuego de plataformas 2D**, inspirado en títulos clásicos como *Donkey Kong* y *Mario Bros*, que marcaron la infancia de los miembros del equipo, y sirvieron como referencia conceptual. En un principio, la idea inicial envolvía la integración de temáticas relacionadas con la **ingeniería informática**; sin embargo, debido a las limitaciones de tiempo y al hecho de partir desde cero en el aprendizaje del motor **Godot**, esta idea fue descartada. En su lugar, se optó por priorizar la creación de un videojuego completo y funcional. No obstante, como consecuencia de esta idea inicial, el personaje principal del juego fue diseñado como un **robot**, manteniendo así un vínculo indirecto con el ámbito tecnológico.

Dado el alcance limitado del proyecto y la falta de experiencia previa con Godot, se destinó aproximadamente el **30 % del tiempo total** a la **formación y aprendizaje** de la herramienta. Esta fase se llevó a cabo principalmente mediante vídeos y tutoriales disponibles en YouTube, utilizando como referencia dos canales especializados en desarrollo de videojuegos con Godot:

* [findemor](https://www.youtube.com/watch?v=-_LiMyZGoXw)
* [LuisCanary](https://www.youtube.com/watch?v=F3T_ZhllzJs&list=PLNEAWvYbJJ9nNOpe6fun7m6L_M8xslYnT&index=1)

Estos recursos sirvieron como base para comprender la lógica del motor, la estructura de escenas, la programación de scripts y la implementación de mecánicas fundamentales. A partir de ellos, se adaptaron y modificaron distintas mecánicas para ajustarlas a las necesidades específicas del proyecto *FactoryFall*.

Una vez adquiridos los conocimientos básicos y desarrollado un **primer nivel funcional**, se procedió a ampliar el juego creando dos niveles adicionales. Este proceso se aceleró gracias al descubrimiento del **TileMap**, que permitió reutilizar la lógica ya implementada y acelerar el diseño de escenarios manteniendo una coherencia en las acciones y mecánicas del jugador.

Finalmente, el **20 % restante del tiempo de desarrollo** se dedicó a la incorporación de elementos transversales, tales como la música, animaciones adicionales, efectos de sonido, ajustes de diseño y corrección de errores. Esta fase permitió pulir el proyecto y alcanzar un resultado final coherente, jugable y alineado con los objetivos iniciales del trabajo.

**Estructura de archivos**

Como el proyecto no se podía realizar en paralelo (problemas de versionado de archivos), el equipo prefirió modularizar cada tarea de forma que no se trabajara en labores de desarrollo al mismo tiempo.

De esta forma, mientras un miembro realizaba una acción del programa (desarrollo de mecánicas, diseño de niveles, implementación de animaciones…), el otro se encargaba de tareas relevantes, para la continuación del alcance: obtención de recursos, organización de las fuentes de información, aclaración del informe, etc.

Para ello, fue creada una carpeta compartida en Drive, dónde se subía la actualización de cada versión del videojuego (en formato .zip), y otros recursos tales como música, sonidos, texturas (fondos, tileSet, sprites, y demás).

**Imagen que contiene Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.**

Imagen que contiene Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.Por otro lado, el proyecto en Godot tenía su propia estructura dónde se agrupaban las escenas correspondientes a los actores principales, o los recursos necesarios.

**Diseño del Videojuego**

**Concepto del juego**

Se trata de un juego de plataformas 2D. El número de niveles es 3, que aumentan progresivamente su dificultad junto con su estilo más degradado. El objetivo del jugador es pasar los niveles. Las 3 formas que tiene el jugador para perder son: cayéndose del escenario, agotando todas sus vidas (con el choque de barriles), o una vez terminado el tiempo en el contador del nivel. Por otro lado, para ganar y pasar de nivel, debe alcanzar la bandera roja situada en el punto final del mismo.

**Ambientación**

El diseño de los mapas se ha hecho principalmente con el siguiente TileSet:

**Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.**

Este ha sido empleado para estructurar los suelos y adornar los distintos mapas a lo largo del videojuego. Gracias a él, se puede crear la distribución del mapa, a la vez que se decora de forma complementaria.

Imagen en blanco y negro

El contenido generado por IA puede ser incorrecto.Vista desde lo alto de una montaña

El contenido generado por IA puede ser incorrecto.Dibujo de un edificio

El contenido generado por IA puede ser incorrecto.Para los fondos, por su parte, se emplearon imágenes generadas por Gemini, manteniendo la premisa de que a mayor dificultad del nivel, mayor contaminación mostraría el fondo.

**Personaje principal**

Para obtener un diseño coherente del robot principal, se optó por pedir un sprite a Gemini para el mismo. Además, se obtuvo también el **Sprite\_Antirobot** (con los ojos naranjas), dónde se simula una energía malvada en el personaje. Realmente este segundo solo fue usado para la animación de idle, dónde se cambia el color de los ojos del personaje si transcurren 2s en posición estática.

La rejilla fue dividida mediante el motor de Godot, para poder usar cada una de las posiciones del personaje refiriéndola como su número de (fila, columna).

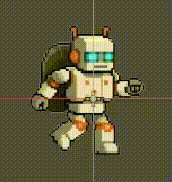
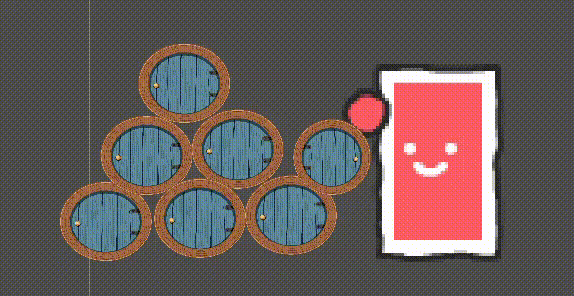
Imagen que contiene diferente, tabla, foto, artículos

El contenido generado por IA puede ser incorrecto.Imagen que contiene diferente, sushi, tabla, juguete

El contenido generado por IA puede ser incorrecto.

**Animaciones**

Se trata del movimiento de los elementos del juego. En este caso, se centraron en los dos personajes principales: el jugador y el boss.

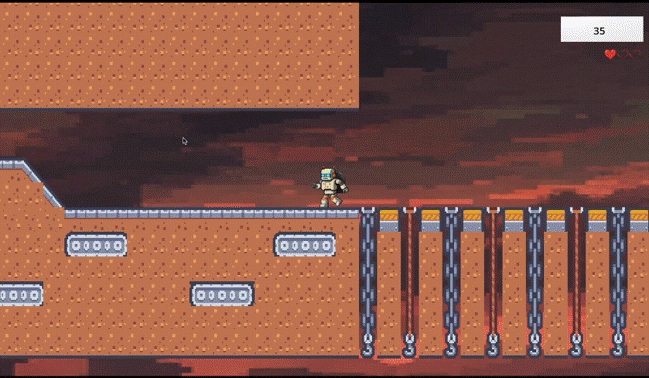
Para realizarlas, se empleó un nodo **AnimationPlayer**, que permite ordenar las secuencias de tiempo, y el aspecto que adoptará cada personaje ante una acción determinada (como el movimiento o el salto).

Concretamente, se hicieron animaciones para el lanzamiento de barriles por parte del boss; y las acciones idle (estático), andar y salto, en el jugador.

**Diseño de niveles**

Si bien no había una idea rígida sobre el diseño del recorrido, se fue estructurando según convenía para sacar máximo partido a las mecánicas logradas y a los efectos más visuales de los niveles.

Imagen que contiene computadora, tabla, monitor, grande

El contenido generado por IA puede ser incorrecto.Por supuesto, la influencia de los juegos *Donkey Kong* y *Mario Bros*, hacían que el diseño mantuviera mecánicas y escenarios muy similares a los suyos.

**Música y sonidos**

Estos fueron sacados principalmente de Pixabay y Uppbeat principalmente. Se trata de música libre de derechos, cuya licencia permite su uso en el videojuego.

Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.

Los sonidos se emplearon para el efecto en el salto del personaje y el golpeo con los barriles (con la consiguiente pérdida de vida).

**Interfaz de usuario**

En todas las pantallas existe un **Controlador**, encargado de gestionar los datos del jugador. Por simplicidad estos datos son las vidas del jugador, y si gana o pierde.

Un conjunto de letras blancas en fondo azul

El contenido generado por IA puede ser incorrecto.Estas vidas se muestran al usuario en la esquina superior derecha, y se actualizan en caso de que el jugador pierda una por el choque de un barril.

Imagen que contiene exterior, luz, tráfico, grande

El contenido generado por IA puede ser incorrecto.Además, se controla que el jugador gane (alcanzando la bandera) o pierda (se quede sin tiempo o sin vidas), mostrando un mensaje de “Victoria” o “Derrota respectivamente, y un botón que permitirá avanzar al siguiente nivel (si gana), o repetir el que no ha podido superar (si pierde).

**Implementación técnica**

En esta sección se comentarán las decisiones tomadas a la hora de realizar el programa y el código empleado.

**Cámara**

El nodo de cámara es ajustado por un objeto Nodo2D, que perseguirá. Gracias a ello, el personaje puede moverse libremente por el mapa, ya que la cámara lo acompañará, en lugar de mantenerse fija (comportamiento por defecto).

**Texto

El contenido generado por IA puede ser incorrecto.**

**Boss**

Encargado de lanzar los barriles, llama a su animación cada intervalo de tiempo para ello. Al terminar el nivel, tanto con victoria como derrota, deja de moverse y generar barriles (ya que de lo contrario podría colapsarse la pantalla con barriles generados tras el timeout).

**Texto

El contenido generado por IA puede ser incorrecto.**

**Barril**

Es la unidad que hace daño al personaje cuándo está en contacto con él. Además, cambia su aspecto a rojo y emite el sonido de golpeo, para indicar al jugador que ha sido impactado.

**Texto

El contenido generado por IA puede ser incorrecto.**

Por otro lado, si el barril cae muy abajo en el mapa, será destruido. Esta decisión de desarrollo permite la ejecución del juego sin problemas de rendimiento por la sobregeneración de barriles.

**Una captura de pantalla de un celular

El contenido generado por IA puede ser incorrecto.**

**Personaje**

Posee los parámetros que permiten crear efectos en el personaje (correr más rápido, reducir su salto, aumentar la gravedad…). Además, es en este script dónde se conectan las teclas del ordenador para ejecutar sus movimientos principales. También se enlazan con esos movimientos los sonidos y las animaciones, que permiten una dinámica más estilizada en su desplazamientos o tiempos de espera (como es el caso del idle).

**Texto

El contenido generado por IA puede ser incorrecto.**

**Controlador**

Esta clase tiene numerosas tareas. La primera de ellas es decidir si el jugador ha ganado/perdido la partida, y emitir una señal para informar al resto de nodos, para que puedan actuar en consecuencia.

**Texto

El contenido generado por IA puede ser incorrecto.**

Esto es posible, ya que es en el controlador dónde se calcula la interfaz de usuario en función de golpes recibidos y valor de las variables.

Texto

El contenido generado por IA puede ser incorrecto.

Además, la cuenta atrás que se muestra al usuario es actualizada mediante esta clase, encargada también si se pierde en el nivel porque ha llegado a cero.

Texto

El contenido generado por IA puede ser incorrecto.

Por último, es en el controlador dónde, una vez alcanzada la victoria en el nivel, se redirige a la siguiente pantalla, restableciendo las vidas y el temporizador.

**Texto

El contenido generado por IA puede ser incorrecto.**

**Conclusión**

La realización de *FactoryFall* ha puesto de manifiesto la importancia del trabajo en equipo, la organización eficiente del tiempo y la capacidad de adaptación en un proyecto con un plazo tan reducido.

A lo largo del desarrollo fue clave dividir tareas, priorizar objetivos alcanzables y apoyarse en herramientas de inteligencia artificial para la generación de recursos visuales, y mejoras de código.

En conjunto, el proyecto demuestra cómo, mediante una planificación adecuada y el uso responsable de nuevas tecnologías, es posible alcanzar los objetivos propuestos incluso en contextos de tiempo y recursos limitados.