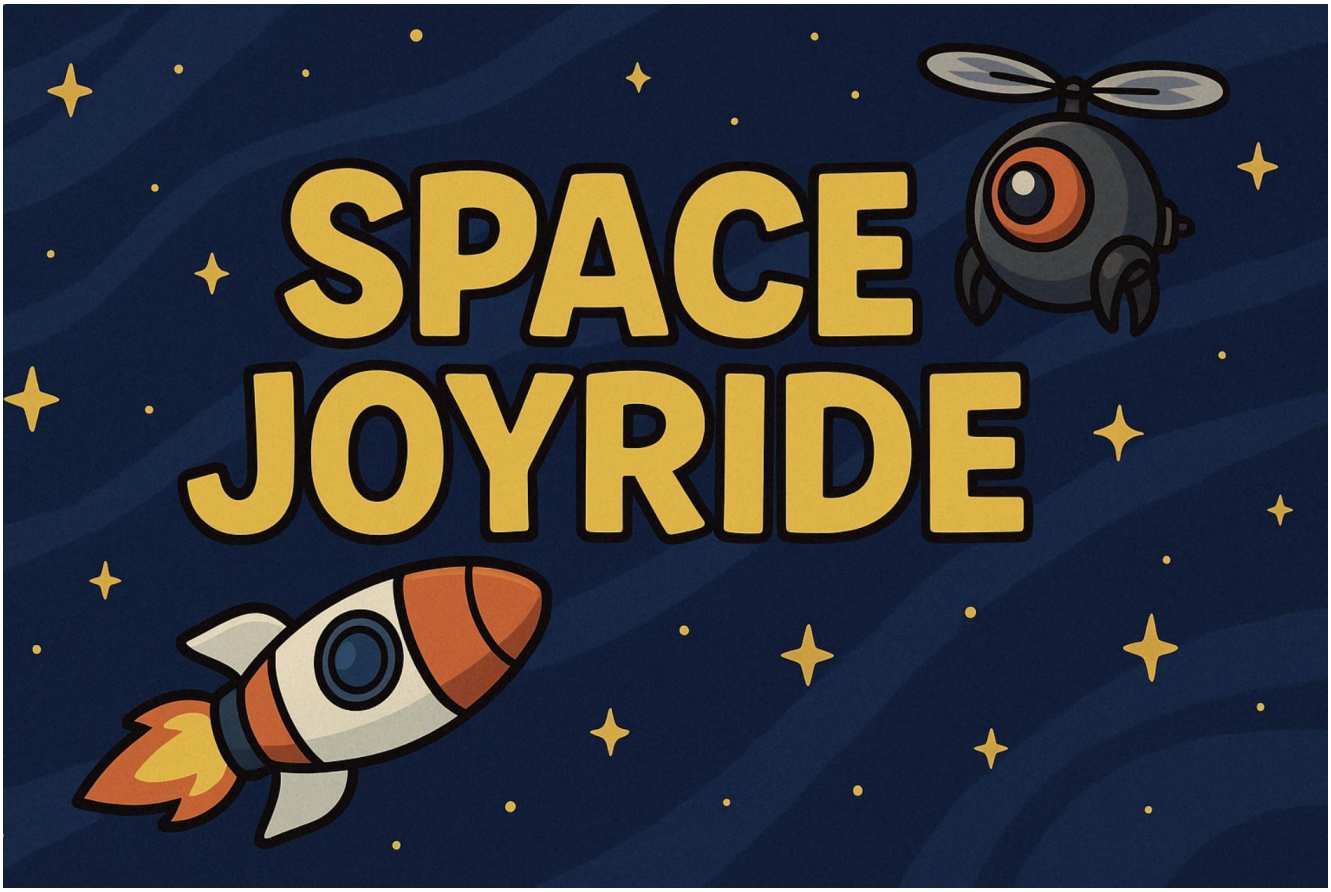

Álvaro Rodríguez Mesa

Claudia Pérez Cuadra

Descripción del juego

El proyecto consiste en el desarrollo de un videojuego tipo runner lateral en 2 5D, creado con Unity, ambientado en un entorno espacial en el que el jugador controla un cohete que avanza automáticamente mientras debe evitar obstáculos y enemigos para sobrevivir el mayor tiempo posible. El control se basa en mantener pulsado para ascender y soltar para descender, incrementándose gradualmente la velocidad a medida que avanza la partida. Durante el recorrido, el jugador recoge estrellas que aumentan su puntuación y le permiten acceder a cajas sorpresa que activan poderes temporales capaces de alterar la jugabilidad (como invertir los controles, reducir la velocidad o modificar la gravedad). El escenario genera de forma continua láseres, drones y misiles que suponen un peligro constante, mientras la puntuación se calcula en función de la distancia recorrida y las estrellas obtenidas. El juego dispone de una dinámica de dificultad progresiva, una estética espacial y un diseño orientado a la acción y la precisión.

Historias de usuario

JUGADOR

1.Movimiento básico

Como jugador, quiero mantener pulsado para subir y soltar para bajar, para esquivar obstáculos.

Objetivo

Permitir el control vertical básico del personaje.

Requisitos extraídos

- El jugador debe subir al pulsar y bajar al soltar.
- El movimiento debe ser fluido.
- El jugador debe detectar colisiones con elementos del escenario.

Tareas

- Implementar script PlayerController.
- Añadir lógica de movimiento pulsar/soltar.
- Ajustar físicas o interpolación.
- Implementar detección de colisiones.

2. Velocidad incremental

Como jugador, quiero que la velocidad aumente con el tiempo, para que el desafío sea progresivo.

Objetivo

Incrementar la dificultad a medida que avanza la partida.

Requisitos

- La velocidad horizontal aumenta con el tiempo.
- La velocidad afecta al desplazamiento del escenario.

Tareas

- Variable de velocidad inicial.
- Función que modifique la velocidad de forma progresiva.
- Aplicar velocidad al fondo.

3. Recoger estrellas

Como jugador, quiero recoger estrellas para aumentar mi puntuación.

Objetivo

Añadir recompensas y progresión.

Requisitos

- Las estrellas deben detectarse al tocarlas.
- La puntuación debe aumentar al recogerlas.

Tareas

- Crear el objeto estrella.
- Detectar colisiones con el jugador.
- Incrementar contador y notificar al GameManager.

4. Conseguir caja sorpresa

Como jugador, quiero que al llegar a X estrellas aparezca una caja sorpresa para obtener un poder especial.

Objetivo

Desbloquear power-ups durante la partida.

Requisitos

- La caja sorpresa se genera al alcanzar X estrella.
- La caja debe aparecer en una zona válida aleatoria.

Tareas

- Implementar contador de estrellas.
- Crear objeto de caja sorpresa.
- Generar la caja cuando se cumpla el umbral.

5. Activar un power-up

Como jugador, quiero obtener habilidades temporales de 20 segundos para tener ventaja momentáne al tocar la caja sorpresa.

Objetivo

Introducir modificaciones temporales en el personaje.

Requisitos

- Activar un power-up aleatorio.
- Duración de 20 segundos.
- Restaurar el estado original al finalizar.

Tareas

- Implementar PowerUpManager.
- Aplicar efectos al jugador.
- Gestionar temporizador y restauración.

ENEMIGOS / OPONENTES CON IA

6. Evitar drones

Como jugador, quiero evitar un dron que me persigue, para no morir si me alcanza.

Objetivo

Introducir un enemigo con comportamiento dinámico.

Requisitos

- El dron debe seguir la posición del jugador.
- El dron debe tener varios estados (patrulla, persecución, ataque).
- Contacto con el misil que lanza el dron implica Game Over.

Tareas

- Crear objeto de dron.
- Implementar máquina de estados (FSM).
- Añadir movimiento hacia el jugador.
- Implementar colisión con el jugador.

7. Esquivar misiles

Como jugador, quiero esquivar los misiles que dispara el dron, para sobrevivir más tiempo.

Objetivo

Aumentar la presión del enemigo sobre el jugador.

Requisitos

- El dron debe disparar misiles periódicamente hacía la posición del jugador.
- El misil debe desplazarse en dirección adecuada.
- La colisión del misil debe matar al jugador.

Tareas

- Crear objeto de misil.
- Implementar trayectoria (recta o teledirigida).
- Detectar colisiones.
- Gestionar disparo desde el dron.

8. Evitar láseres

Como jugador, quiero evitar láseres fijos generados aleatoriamente para no morir al tocarlos.

Objetivo

Añadir obstáculos letales estáticos.

Requisitos

- Generación de láseres en posiciones aleatorias.
- Contacto con el láser implica muerte.
- Destrucción del láser al salir de pantalla.

Tareas

- Crear objeto de láser.
- Implementar spawner aleatorio.
- Añadir animación o efecto visual.
- Detección de colisiones.

3. GAMEPLAY / PROGRESIÓN

9. Dificultad creciente

Como jugador, quiero que aparezcan más láseres con el tiempo, para aumentar el desafío.

Objetivo

Escalar la dificultad global del juego.

Requisitos

- Incremento gradual de la cantidad de láseres.
- Modificación dinámica de la frecuencia de spawn.

Tareas

- Implementar sistema de dificultad.
- Ajustar frecuencia de spawners.

10. Power-up de velocidad reducida

Como jugador, quiero un modo que reduzca mi velocidad durante tiempo limitado.

Objetivo

Reducir temporalmente la dificultad de movimiento.

Requisitos

- Reducir la velocidad del jugador.

- Restaurar velocidad posterior.

Tareas

- Implementar multiplicador de velocidad.
- Añadir lógica en PowerUpManager.

11. Power-up de tamaño pequeño

Como jugador, quiero hacerme más pequeño para esquivar mejor.

Objetivo

Modificar el tamaño del personaje durante un tiempo limitado.

Requisitos

- Cambiar la escala del jugador.
- Ajustar colisionador.
- Restaurar tamaño.

Tareas

- Modificar escala.
- Ajustar colisionador.

12. Power-up de controles invertidos

Como jugador, quiero que el control se invierta durante tiempo limitado.

Objetivo

Alterar la forma de controlar al personaje.

Requisitos

- Invertir comportamiento del control.
- Restaurar al finalizar.

Tareas

- Variable de control invertido.
- Modificar lógica de movimiento.

13. Power-up Gravity Flip

Como jugador, quiero moverme solo por techo o suelo sin pasar por el medio.

Objetivo

Cambiar completamente la forma de moverse.

Requisitos

- Movimiento solo entre dos posiciones fijas (techo/suelo).
- No cruzar por el centro.
- Restauración posterior.

Tareas

- Definir posiciones top/bottom.
- Implementar los desplazamientos automáticos.
- Restauración de movimiento normal.

14. Power-up Toggle

Como jugador, quiero que al pulsar cambie entre techo y suelo instantáneamente.

Objetivo

Permitir cambios rápidos en la posición del jugador.

Requisitos

- Cambio instantáneo entre techo y suelo.
- No cruzar por el centro.
- Restaurar tras la duración.

Tareas

- Implementar salto instantáneo top/bottom.
- Integrar con el sistema de power-ups.

4. INTERFAZ / SISTEMA

15. Ver mi puntuación

Como jugador, quiero ver mi puntuación en pantalla en todo momento.

Objetivo

Mostrar información actual de la partida.

Requisitos

- Visualización del marcador.
- Actualización continua.

Tareas

- Crear UI de puntuación.
- Vincular con GameManager.

16. Ver estrellas recogidas

Como jugador, quiero ver cuántas estrellas llevo.

Objetivo

Informar al jugador del progreso hacia la caja sorpresa.

Requisitos

- Mostrar contador de estrellas.
- Actualización al recogerlas.

Tareas

- Crear UI del contador.
- Actualizarlo al recoger estrella.

17. Game Over

Como jugador, quiero una pantalla de derrota con mi puntuación final y un botón de reiniciar.

Objetivo

Completar el ciclo de juego con una salida clara.

Requisitos

- Pantalla de Game Over.
- Botón de reinicio.
- Mostrar puntuación final.

Tareas

- Crear pantalla de Game Over.
- Añadir botón "Reintentar".

- Mostrar puntuación final.

Implementación de patrones de diseño

Durante el desarrollo de Space Joyride se van a utilizar varios patrones de diseño con el objetivo de estructurar el sistema, evitar problemas comunes en arquitecturas de videojuegos y facilitar la escalabilidad futura. Hemos implmentado algunos patrones que tenemos predeterminados hacer, sin embargo, igual implementamos más en función de como vayamos desarrollando el juego.

Patrón Observer / Event-Driven Architecture

Uno de los patrones centrales en Space Joyride es Observer, implementado mediante el sistema de eventos de C# y Unity. Este patrón resulta especialmente adecuado cuando varios componentes del juego requieren ser notificados de un cambio sin que exista una dependencia directa entre ellos. En nuestro videojuego este problema aparece de forma natural: cuando el jugador recoge una estrella, el GameManager debe actualizar la puntuación global, el UIManager debe actualizar el marcador en pantalla, y el ItemSpawner debe comprobar si se han alcanzado las condiciones para generar una caja sorpresa. Mediante el uso de Observer, cada subsistema se suscribe únicamente a la información que necesita, sin conocer ni depender del resto.

Patrón State para los Power-Ups y el comportamiento del Dron

El patrón State ha sido aplicado tanto al sistema de power-ups del jugador como a la inteligencia del dron enemigo. Este patrón es especialmente útil cuando un objeto debe adoptar distintos comportamientos en función de su estado interno, evitando largas cadenas de condicionales que complican el mantenimiento del código.

En el caso del jugador, cada power-up (modo mini, velocidad reducida, controles invertidos, gravity flip y toggle) define un comportamiento distinto del movimiento durante un tiempo limitado. Al encapsular el comportamiento en clases de estado, cada power-up queda aislado, y el PowerUpManager controla únicamente la activación y desactivación, manteniendo el código modular y coherente.

Del mismo modo, el dron enemigo posee una máquina de estados con tres fases: *Patrol*, *Chase* y *Attack*. Cada fase define un comportamiento concreto (movimiento lineal, seguimiento del jugador, o disparo de misiles). El uso de State permite representar esta lógica de forma clara y extensible, lo que facilita la futura incorporación de nuevos comportamientos o ajustes de dificultad.

Patrón Singleton en el GameManager

Un uso controlado del Singleton es especialmente útil en el GameManager, encargado de controlar la puntuación, la gestión de estados generales y la comunicación con otros subsistemas. En Space Joyride solo se emplea para asegurar una única instancia accesible globalmente durante la ejecución. Esto evita inconsistencias y simplifica el acceso sin sacrificar la modularidad.