Plan de Desarrollo Avanzado: Space Invaders Renovado
Pensado para un intrépido demonio de Tasmania con ojos morados y un tranquilo zorro plateado de mirada verde.

¡Bienvenidos, valientes exploradores, al emocionante y renovado universo de *Space Invaders*! Siguiendo la esencia clásica de jugabilidad simple pero adictiva, este plan integra nuevas mecánicas estratégicas de manera progresiva, al más puro estilo del "Método Nintendo". A lo largo de estas secciones, encontrarán consejos y sugerencias sobre cómo llevar el proyecto a cabo en *Unity* y cómo utilizar herramientas de animación como *LeanTween*, todo sin escribir todavía líneas de código concretas. ¿Listos para emprender la misión de crear un juego tan fascinante como sus propias leyendas?

---

1. Mecánicas Fundamentales (Alta Prioridad)

1.1 Controles del Jugador

Movimiento Horizontal

- En *Unity*, pueden asignar el movimiento de la nave a través del componente *Input System* o las funciones *Input.GetAxis* heredadas del sistema clásico.
- Para mejorar la respuesta, ajusten la sensibilidad en el *Project Settings* o añadan una curva de aceleración, de modo que sea suave y preciso a la vez.
- Si quieren pulir aún más las transiciones de movimiento, consideren usar *LeanTween* para interpolar la posición de la nave con animaciones fluidas y naturales (por ejemplo, para un ligero bamboleo al moverse).

Disparo Básico

- Para disparar, es tan sencillo como detectar la tecla (barra espaciadora) o el botón del mando en el *Update()* del script principal.
- Agreguen un sistema de *prefabs* para los proyectiles y manéjenlos con *Object Pooling* para optimizar el rendimiento (especialmente útil si el demonio de Tasmania se emociona y dispara sin parar).

Personalización de Controles

- Creen un menú de opciones con *Canvas* y *UI Buttons* en Unity.
- Para dar una sensación de fluidez al abrir y cerrar el menú, podrían utilizar *LeanTween* en los paneles, escalándolos o desvaneciéndolos suavemente.

1.2 Enemigos

Movimiento en Cuadrícula

- Utilicen un *GameObject* padre que contenga a todos los enemigos en filas y columnas. Con cada actualización, el objeto padre se mueve hacia un lado y, al colisionar con un límite virtual, invierte su dirección y desciende ligeramente.
- Para un efecto de “caída” más elegante, *LeanTween* puede interpolar la posición del contenedor de enemigos cuando cambian de dirección.

Incremento de Velocidad

- A medida que los enemigos se van eliminando, ajusten un multiplicador de velocidad o reduzcan el tiempo de *WaitForSeconds* en las corrutinas de movimiento.

Comportamiento Avanzado

- Integre patrones de ataque usando *ScriptableObjects* o variables públicas en los scripts de cada enemigo, definiendo así diferentes comportamientos (disparo teledirigido, ráfagas múltiples, movimientos zigzag, etc.).

1.3 Colisiones

Sistema de Daño

- En *Unity*, pueden aprovechar la detección de colisiones con *OnTriggerEnter2D* (o *OnCollisionEnter2D*) si usan físicas 2D, o sus equivalentes en 3D.
- Para mejorar la retroalimentación, reproduscan partículas de explosión cuando un enemigo o bunker es alcanzado. Con *LeanTween*, también se puede animar la escala de la nave enemiga al recibir impacto, dándole un efecto de “sacudida” visible.

---

2. Interfaces de Usuario (Alta Prioridad)

2.1 Pantalla Inicial

Menú Animado

- Estructuren la escena de menú con un *Canvas* principal y paneles para cada sección (Jugar, Opciones, Salir).
- *LeanTween* ofrece funciones como `LeanTween.move(...)` o `LeanTween.alpha(...)` para lograr transiciones fluidas en la interfaz. Así, cada opción podría aparecer flotando suavemente con un ligero retardo o “fade in”.

2.2 Selección de Naves (Minijuego de Elección)

- Creen un escenario corto en Unity donde el jugador pruebe los distintos tipos de nave antes de iniciar la partida real.
- Cada nave (Equilibrada, Rápida, Lenta) puede presentarse como un *prefab* distinto con sus propias propiedades de velocidad, daño y habilidad especial.

- Para una pequeña animación al elegir cada nave, usen *LeanTween* para rotar o hacer un *scale up/down* del modelo en la pantalla de selección.

2.3 Patrones de Enemigos Aleatorios

- Configuren *prefabs* diferentes para enemigos con variaciones de velocidad, resistencia y tipo de disparo.
- Al cargar cada nivel, un script generará filas de enemigos aplicando lógica aleatoria para la disposición (espacios vacíos, enemigos raros en posiciones sorpresivas, etc.).

2.4 Pantallas de Transición

Mensajes Animados

- "Nivel Completado" o "Game Over" pueden implementarse como paneles que aparezcan sobre la pantalla de juego. Para un efecto más profesional, se podría animar la escala de cada mensaje con *LeanTween*, enfatizando su entrada triunfal.

Estadísticas del Nivel

- Muestren puntaje, precisión y enemigos eliminados. Pueden usar un script que calcule la precisión dividiendo el número de disparos acertados por el total de disparos realizados, redondeando para mostrar un porcentaje.

2.5 Selección del Modo Bunker

- Tras elegir la nave, disparar a un bunker destruido (sin bunkers) o reparado (con bunkers) puede hacerse con *Raycast* si se trabaja en 3D o *Raycast2D* en 2D.
- Al habilitar el "Modo Bunkers", spawneen cuatro bunkers al inicio y permitan al jugador configurarlos (defensivo, curativo o mixto).

  Ejemplo Divertido:
  Imagina al demonio de Tasmania disparándole frenéticamente al bunker destruido porque quiere un reto máximo sin defensas; mientras, el zorro plateado, más precavido, dispara al bunker reparado para proteger sus naves con un plan a largo plazo.

---

3. Modificaciones Obligatorias (Prioridad Media)

3.1 PowerUps Aleatorios

- Cuando un enemigo es destruido, hay una probabilidad de que suelte un PowerUp. Para resaltarlo, se puede animar la rotación del objeto usando *LeanTween* o un simple *script* de rotación continua.
- Tipos de PowerUps:

    - Reparación de bunkers.
    - Aumento de velocidad de disparo.
    - Escudo temporal.
    - Inversión de controles (¡perfecto para confundir al jugador!).
    - Munición explosiva.

3.2 Implementación de Bunkers Tipo A y Tipo B

- **Bunker Tipo A (Escudos para la Nave)**
    - Al activarse, muestren un efecto visual (campo de energía alrededor del bunker). Con *LeanTween*, pueden expandir o pulsar un objeto tipo “esfera de energía” para enfatizar que el escudo está activo.
- **Bunker Tipo B (Cura para Estructuras o la Nave)**
    - Visualicen una especie de rayo o partícula que fluya hacia los objetos reparados. También podrían usar sonidos chisporroteantes o un zumbido futurista.

3.3 Jerarquías de Dificultad

- Para cada nivel, definan la disposición y tipo de enemigo en filas superiores (más fuertes) e inferiores (más débiles pero más numerosos).
- Con *Listas Dobles* o cualquier estructura dinámica, pueden manejar la lógica de generación de formaciones que van ajustándose a la progresión del jugador.

---

4. Innovaciones Adicionales (Opcional)

4.1 PowerUps Avanzados

- **Drones Aliados:**
    - Pequeñas naves de apoyo que siguen al jugador. Podrían detectarse con *OverlapCircle* en 2D (o su análogo en 3D) para disparar a enemigos cercanos.
    - Animen su movimiento con *LeanTween*, haciéndolos orbitar la nave del jugador o desplazarse en un patrón que subraye su carácter “robótico”.

4.2 Sistema Bunker Tower Defense

- **Gestión de Recursos:**
    - Para cada bunker protegido, se suman puntos o “energía”. Luego, se utilizan para mejorar los bunkers (más resistencia, mayor frecuencia de disparo, etc.).
- **Defensas Activas:**
    - Añadan bunkers ofensivos (que disparan de forma automática), recolectores (recogen puntos al destruir enemigos) y curativos (restauran estructuras).

    - A medida que avanza la partida, el jugador puede desbloquear más bunkers (hasta 10) si cumple objetivos específicos.

4.3 Estrategia Híbrida

- **Método Tower Defense:**
    - Bunkers especializados (Guerrero, Mago, Curandero, Ingeniero) que introducen variedad de estrategias.
- **Método Arkanoid:**
    - Bloques destructibles que, al romperse, sueltan PowerUps o generan nuevas oleadas de enemigos.
- **Método Clickers:**
    - Permite mejorar bunkers con clics o pulsaciones rápidas, ofreciendo recompensas inmediatas.
- **Método Space Invaders Tradicional:**
    - Mantiene el desplazamiento en cuadrícula, asegurando la esencia retro.

---

5. Resumen

1. **Fase 1: Gameplay Básico**
    - Crear controles pulidos en Unity (movimiento y disparo).
    - Programar movimiento y ataque de enemigos en cuadrícula.
    - Implementar colisiones funcionales.
2. **Fase 2: Modificación Obligatoria**
    - Integrar PowerUps visualmente atractivos.
    - Añadir jerarquías y evoluciones de enemigos.
    - Generar patrones de formación con estructuras dinámicas (Listas Dobles).
3. **Fase 3: Menús Funcionales**
    - Diseñar pantallas iniciales y de transición.
    - Incluir el minijuego de selección de nave y modo bunker.
4. **Fase 4: Extras Creativos**
    - Añadir PowerUps avanzados (drones, transformaciones de nave).
    - Mejorar efectos audiovisuales con partículas y animaciones.
    - Combinar mecánicas de Tower Defense, Arkanoid, Clickers, etc.

Sugerencia Inspiradora:
Dejen que el demonio de Tasmania con ojos morados pruebe la *Fase 1* a toda velocidad, disparando a cuanto bicho se mueva. Luego, soliciten la opinión del zorro plateado con ojos verdes para ajustar la dificultad y optimizar la estrategia en las fases posteriores.

---

6. Evaluación Detallada

1. **Juego Completo (4 puntos)**
    - Incluir controles bien implementados, enemigos funcionales y colisiones.
2. **Modificación Obligatoria (3 puntos)**
    - PowerUps aleatorios y patrones de enemigos dinámicos con bunkers especializados.
3. **Menús Funcionales (2 puntos)**
    - Pantallas claras, transiciones animadas y un minijuego de selección de nave/bunker.
4. **Extras Creativos (3 puntos)**
    - Integración de sistemas adicionales (PowerUps avanzados, elementos de Tower Defense y Arkanoid, etc.).
5. **Código Modular (2 puntos)**
    - Scripts separados por función, permitiendo una fácil reutilización y mantenimiento.

Beneficios de la Modularidad

- **Reutilización de Scripts:** El script para el movimiento de enemigos puede servir para futuros proyectos espaciales.
- **Mantenimiento Sencillo:** Cada funcionalidad aislada facilita la detección y corrección de errores.
- **Escalabilidad:** Añadir nuevas mecánicas, como nuevos tipos de bunkers o PowerUps, resulta más práctico y seguro.

Entrega Final

- **GitHub:**
    - Subir el proyecto con etiquetas de versiones (v1.0, v2.0...) y mensajes de *commit* claros.
- **Archivo Comprimido:**
    - Incluir la carpeta de Unity y el ejecutable (.exe) para facilitar la prueba sin necesidad de la versión original de Unity.

¡Y eso es todo, intrépidos aventureros! Con estos lineamientos, podrán combinar la magia retro de *Space Invaders* con dinámicas modernas y entretenidas. Utilicen *Unity* para la estructura principal del juego y anímense a dar toques más suaves y pulidos con *LeanTween*. De este modo, tanto el demonio de Tasmania con ojos morados como el zorro plateado de ojos verdes podrán disfrutar de una experiencia envolvente, llena de acción, estrategia y ese encanto inconfundible de los clásicos reinventados.

¡A disparar y a proteger bunkers, estrellas del cosmos! Que la victoria galáctica sea suya.