PLANIFICACIÓN Y GESTIÓN DEL PROYECTO

1. Marco y documentación.

Marco: Ágil ligero (Scrum adaptado a videojuego académico)
Duración sugerida: 4 Sprints (1–2 semanas cada uno)

Elemento	Uso en tu proyecto
Backlog	Sale directamente del GDD
Historias de usuario	Funcionalidades jugables
Tareas técnicas	Implementación en Babylon.js
Assets tasks	Arte, sonido, VFX, UI
Definition of Done	Funciona, se ve, se prueba en navegador

2. Estructura del Backlog

EPIC 1 — Documentación (GDD)

Historia / Tarea	Tipo
Crear estructura base del GDD	Documentación
Completar sección de estilo visual y paleta	Diseño
Definir tipos de enemigos (básico, rápido, jefe)	Game Design
Especificar comportamiento del jefe final	Game Design

EPIC 2 — Core Gameplay

Historia de Usuario	Resultado Jugable
Como jugador quiero mover mi nave en 2 ejes	Control total nave
Como jugador quiero activar disparo automático con espacio	Sistema toggle
Como jugador quiero perder vidas al ser impactado	Sistema de vidas
Como jugador quiero que aparezcan enemigos por oleadas	Sistema de spawn
Como jugador quiero ganar al eliminar todos los enemigos	Condición de victoria

Como jugador quiero perder al quedarme sin vidas	Game Over

EPIC 3 — Sistema de Combate

Historia	Mecánica
Disparos del jugador destruyen enemigos	Colisiones bala-enemigo
Enemigos disparan proyectiles	Ataques enemigos
Enemigos desaparecen tras tiempo límite	Sistema de temporizador

EPIC 4 — Progresión y Dificultad

Historia	Diseño
Fase 1 con enemigos lentos	Introducción
Fase 2 con más frecuencia y menos tiempo en pantalla	Escalada
Fase 3 con enemigos rápidos y jefe final	Clímax
Sistema de puntuación por enemigo destruido	Score

EPIC 5 — Power-Ups

Historia	Sistema
Enemigos tienen 5% de probabilidad de soltar boost	Drop system
Boost de velocidad del jugador	Buff temporal
Boost de disparo mejorado	Buff ofensivo

EPIC 6 — UI & Feedback

Historia	Elemento
Mostrar vidas en pantalla	HUD
Mostrar puntuación	HUD
Mostrar mensaje de Game Over	UI estado
Mostrar mensaje de Victoria	UI estado

EPIC 7 — Assets

Tipo	Tareas

Visual	Nave jugador sprite
Visual	Enemigos (mínimo 2 tipos + jefe)
Visual	Balas jugador/enemigo
Visual	Fondo espacial
VFX	Explosión con partículas
Audio	Sonido disparo
Audio	Sonido explosión
Audio	Música fondo arcade

3. Plan por Sprints

Sprint 1 — Base jugable

- Movimiento nave
- Disparo automático toggle
- Bala del jugador
- Un tipo de enemigo
- Sistema de vidas
- Colisiones básicas

Sprint 2 — Combate completo

- Proyectiles enemigos
- Spawn por oleadas
- Temporizador de enemigos
- Sistema de puntuación
- HUD básico

Sprint 3 — Progresión y diseño

- Fases 1, 2 y 3
- Jefe final
- Power-ups
- Dificultad progresiva

Sprint 4 — Pulido

- Sonido
- Partículas de explosión
- Pantallas de victoria/derrota
- Ajuste de balance

- Pruebas

ARQUITECTURA TÉCNICA (Babylon.js)

1. Arquitectura General

Game

1. GameStateManager
2. SceneManager
3. EntityManager
   - Player
   - Enemies
   - Bullets
   - PowerUps
4. Systems
   - InputSystem
   - CombatSystem
   - SpawnSystem
   - CollisionSystem
   - ScoreSystem
   - UISystem

2. Máquina de Estados del Juego

Estado	Qué pasa
MENU	Pantalla inicial
PLAYING	Gameplay activo
GAME_OVER	Sin vidas
VICTORY	Nivel completado
PAUSED (opcional)	Juego detenido

3. Entidades Principales

Entidad	Componentes
Player	Sprite, Position, Movement, Shooter, Health
Enemy	Sprite, MovementPattern, Timer, Shooter

Bullet	Sprite, Velocity, Damage
PowerUp	Sprite, EffectType, Duration

4. Sistemas

- InputSystem
  - Detecta WASD/flechas
  - Detecta toggle de disparo

- CombatSystem
  - Genera balas
  - Controla daño

- SpawnSystem
  - Controla oleadas
  - Maneja fases del nivel

- CollisionSystem
  - Bala vs enemigo
  - Bala enemigo vs jugador
  - Nave vs enemigo

- EnemyTimerSystem
  - Cuenta tiempo de vida del enemigo
  - Lo elimina si expira

- ScoreSystem
  - +1 por enemigo
  - Bonus por eliminación total

- PowerUpSystem
  - Probabilidad de drop (5%)
  - Aplica buffs temporales

4. Estructura de Código

/src

1. core/
- Game.js
- GameStateManager.js
2. entities/
- Player.js
- Enemy.js
- Bullet.js
- PowerUp.js
3. systems/
- InputSystem.js
- SpawnSystem.js
- CollisionSystem.js
- CombatSystem.js
- UISystem.js
4. assets/
5. ui/

5. Flujo del Core Loop

1. Update Loop (Babylon Render Loop)
2. InputSystem
3. Movement Update
4. SpawnSystem
5. CombatSystem
6. CollisionSystem
7. EnemyTimerSystem
8. Score & UI Update