HELL BOUND

Juegos Interactivos

Escuela Politécnica Nacional

Jefferson David Chileno Manobanda
David Alejandro Quille Llumiguano

Ingeniería de Software
2025-B

Parte 1 – Conceptualización y Resumen Ejecutivo (High Concept)

Título provisional: HELL-BOUND

Género: FPS (First Person Shooter) / Survival Horror Retro

Elevator Pitch:
HELL-BOUND es un descenso a la locura en primera persona donde un soldado condenado debe purgar los 9 círculos del infierno. Armado solo con 5 balas sagradas y su voluntad, el jugador deberá gestionar cada disparo y utilizar el entorno para sobrevivir a hordas demoníacas. No es solo un shooter; es un puzzle de supervivencia donde errar un tiro significa la muerte.

Referencia (X meets Y): DOOM (1993/2016) meets Quake (por ritmo arena) – experiencia de combate directo y frenético.

Público objetivo: Jugadores de acción/retro-FPS y cursos de desarrollo de juegos interesados en mecánicas de combate, balance y diseño de niveles de ritmo alto.

USPs (Puntos Únicos de Venta):

1. **Estética "Pixel-Crunch"**: Gráficos 3D modernos con renderizado pixelado para emular la era de los 32-bits.
2. **Economía de la Muerte:** Munición extremadamente limitada. Cada bala fallada acerca al jugador al "Game Over".
3. **Narrativa Ambiental Diegética:** La interfaz (la cara del protagonista) cuenta la historia de su sufrimiento físico en tiempo real.

Parte 2 – Mecánicas y Gameplay (El Núcleo MDA)

Core Loop (bucle principal repetible):

1. Entrar en el cuarto/arena → 2. Eliminar enemigos/abrir rutas → 3. Recoger pickups (salud/armadura/ammo) y llaves → 4. Acceder a siguiente zona / volver al hub → 5. Mejoras / guardar / repetir.

Mecánicas clave:

- Movimiento
  - Caminar: 5.0 m/s.
  - Sprint: 7.0 m/s (consume estamina; duración 3 s antes de enfriamiento).
  - Strafing: movimiento lateral con 0% penalización de precisión.
  - Salto: permitidos (salto simple 1.2 m), con posibilidad de salto en cadena para sortear plataformas (verticalidad moderada).

- Salud / Armadura
  - Salud máxima: 100.
  - Armadura máxima: 100 (funciona como buffer que absorbe X% daño según tipo).

- Armas
  - Pistola (munición común): daño base 15 por disparo.
  - Escopeta (corto alcance): daño por impacto 65, mejor contra grupos cercanos.
  - Chaingun (alta cadencia): DPS alto, consumo de munición alto.
  - Lanzacohetes / BFG (arma pesada): proyectil pesado / área.
  - Mecánica de switching instantáneo con prioridad de arma (weapon swap rápido).

- Interacción con pickups
  - Salud pequeña +10, mediana +25, grande +50.
  - Armadura +25 o +50.
  - Munición por tipo con stacks limitados.

- Inputs (propuesta)
  - Movimiento: WASD, Mouse look.
  - Disparo primario: Click izquierdo. Secundario/alterno: Click derecho.
  - Correr: Shift, Recargar/usar: R / E.

Critierios de aceptación / pruebas:

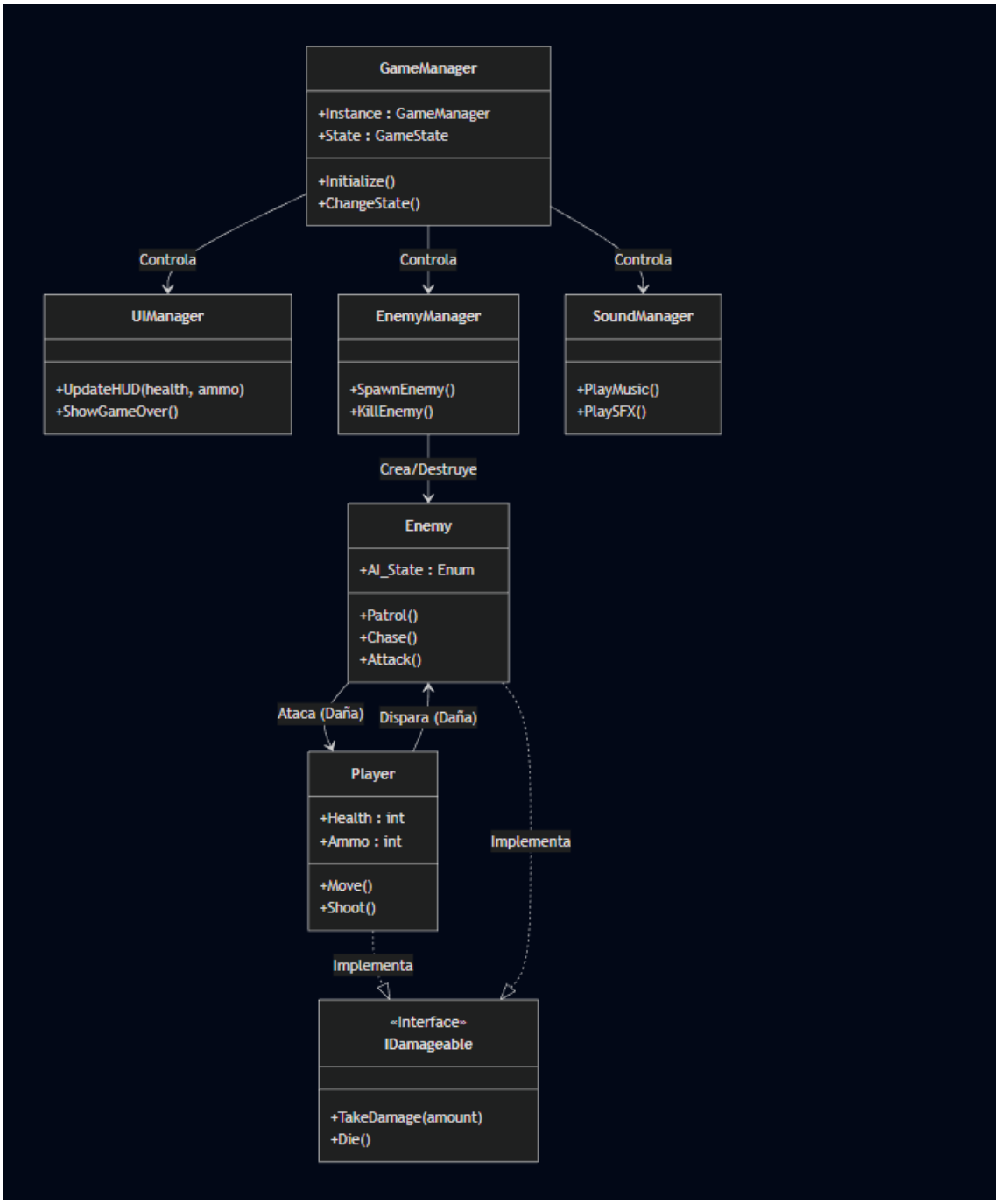
- PlayerController con colisiones sólidas (sin atravesar mallas), velocidades medibles.
- Armas disparan, consumen munición, aplican daño y activan efectos VFX/SFX.

- Pickups funcionan y actualizan UI inmediatamente.

Parte 3 — Dinámicas, Sistemas y “Economía” de juego

Sistemas principales:

- **Spawn & Wave System**: spawns controlados por triggers; dificultad escala por número y tipo de enemigos.
- **Pool de entidades** (Object Pooling) para balas, efectos y enemigos para mantener rendimiento.
- **IA de enemigos**: estados FSM (Patrulla → Perseguir → Atacar → Buscar cobertura) con comportamientos sencillos pero efectivos (rush, flanker, ranged).
- **Loot placement rules**: reglas para colocar pickups (ej.: tras matar mini-encuentro, en rutas riesgosas, o como recompensa por puzzle/secret).



Datos/Balance:

```
{
  "weapons": [
    {"id":"pistol","damage":15,"clip":12},
    {"id":"shotgun","damage":65,"pellets":8,"spread":12}
  ],
  "pickups":[
    {"id":"health_small","heal":10},
    {"id":"armor_large","armor":50}
  ]
}
```

Parte 4 — Narrativa, Mundo y Diseño de Niveles

Tipo de narrativa: ambiental y fragmentada — notas de la UAC, terminales, entorno que cuenta la invasión. No se requieren cinemáticas largas; la historia se sugiere por objetos y escenarios.

Premisa corta: una brecha dimensional en la estación UAC permite la entrada de demonios; el protagonista (marine) debe cerrar la brecha y detener la fuente.

Diseño de Niveles (Nivel 1 — "Sector de Mantenimiento")

- **Inicio (Safe Room):** armas básicas, tutorial.
- **Threshold:** compuerta de acceso a pasillo industrial.
- **Arena principal:** sala con rutas elevadas y cajas, patrullas de imps y un mini-encuentro con spawn en oleadas.
- **Goal:** panel de control para desbloquear la puerta al siguiente sector.
- **Extracción / Checkpoint:** rejilla de ventilación / ascensor.

Elementos de mapa:

- Secrets (habitaciones ocultas) con munición/armadura.
- Shortcuts que reducen tiempo si se dominan.
- Volumen de trigger para cambiar música y aumentar tensión.

Greyboxing: diseñar niveles con cubos y placeholders primero; comprobar flujo y tiempos antes de añadir arte.

Parte 5 — Estética, UI/UX y Arte Técnico

Dirección artística: industrial-horror, mezcla de metales fríos con rojo demoníaco; alto contraste para que enemigos y pickups sean legibles a distancia.

Paleta sugerida: grises/metalizados + acentos rojos y naranjas para peligro; verdes/amarillos para pickups.

UI / HUD (minimalista):

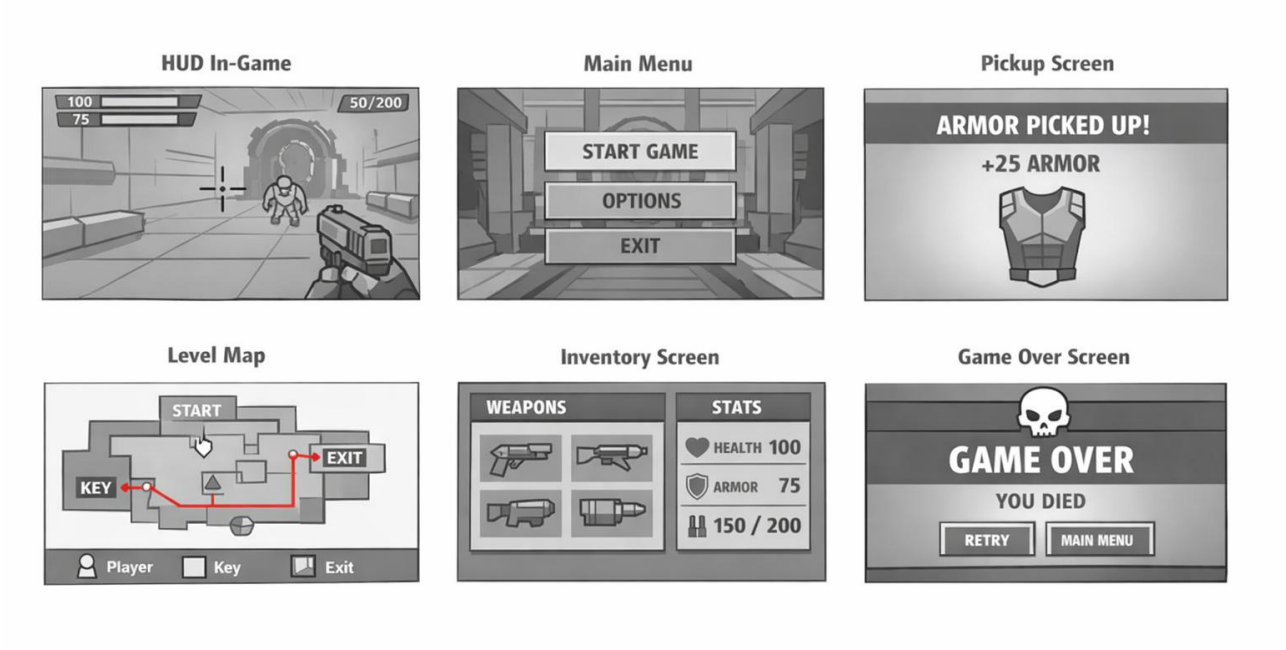
- Salud (esquina superior izquierda).

- Armadura (justo debajo).
- Munición actual / total (esquina inferior derecha).
- Retícula dinámica (cambia al apuntar o en lock).
- Menús: pausa, opciones, pantalla de muerte con respawn o restart.

Constraints técnicos (web/PC):

- Formato/modelos: .glb / .fbx.
- Polycount sugerido: personaje (player arms + weapon): ≤ 8k tris; enemigos pequeños ≤ 5k.
- Texturas: 1024×1024 para personajes principales, 512×512 para props.

Implementación UI: UIManager que escucha eventos de cambio de salud/armadura/ammo y actualiza la HUD (separación lógica/presentación).



Parte 6 – Arquitectura Técnica y Stack Tecnológico

Stack Tecnológico:

Categoría	Herramienta	Justificación
Motor Gráfico	Babylon.js 6.0+	Framework nativo web robusto, ideal para accesibilidad sin instalación.

Categoría	Herramienta	Justificación
Lenguaje	TypeScript	Tipado estricto necesario para mantener una arquitectura escalable y sin errores de tipo.
Control de Versiones	Git / GitHub	Gestión de ramas (Main, Develop, Features) y Pull Requests.
IDE	VS Code	Con extensiones de ESLint y Prettier para estandarización de código.

Estructura de carpetas propuesta:

1. /managers
   - (Scripts encargados de la gestión visual, enemigos y construcción del entorno 3D)
2. /systems
   - (Lógica central del juego, máquina de estados, control de inputs y gestión de la interfaz)
3. /texturas
   - (Todos los recursos gráficos: Sprites 2D, texturas de paredes/suelos y assets de la UI)
4. /mp3
   - (Todos los recursos de audio: Música de fondo y efectos de sonido)
5. /docs
   - (Documentación técnica del proyecto, GDD y reportes)

Patrones y decisiones claves:

- **Singleton:** GameManager, AudioManage.
- **State Machine:** para estados del juego y estados de IA.
- **Object Pooling:** para balas, efectos y enemigos.
- **Event System / Observer:** desacoplar UI de lógica

GameManager (resumen): singleton que controla estados (MENU, PLAYING, PAUSED, GAMEOVER), carga/descarga de escenas y manejo de checkpoints.

Parte 7 – Mantenimiento del GDD Vivo y Planificación de Sprints

Changelog: GDD v0.1 – Base del high concept y core loop; v0.2 – mecánicas y primer prototipo movement + shooting.

Roadmap de Sprints:

Sprint 1: MVP
Objetivo: Tener un personaje jugable que pueda disparar y enemigos básicos funcionales en un entorno de prueba.

ID	Historia de Usuario	Criterios de Aceptación (Definition of Done)	Puntos
HU-01	**Movimiento del Jugador** Como jugador, quiero moverme por el escenario usando WASD para explorar el nivel.	1. El personaje se desplaza en 4 direcciones. 2. Se detectan colisiones con paredes. 3. La cámara responde al mouse.	5
HU-02	**Disparo Básico** Como jugador, quiero disparar mi arma y ver un efecto visual para saber que ataqué.	1. Clic izquierdo activa animación del arma. 2. Se reproduce sonido de disparo.	8

ID	Historia de Usuario	Criterios de Aceptación (Definition of Done)	Puntos
		3. Se descuenta munición del contador.	
HU-03	**IA de Persecución** Como jugador, quiero que los enemigos me sigan al verme para sentir peligro.	1. El enemigo rota hacia el jugador. 2. El enemigo se desplaza hacia la posición del jugador. 3. Si toca al jugador, causa daño.	8
HU-04	**Feedback de Estado (UI)** Como jugador, quiero ver la cara de mi personaje cambiar según mi salud.	1. Cara "Normal" al 100% de vida. 2. Cara "Herida" al recibir daño. 3. Cara "Muerta" al llegar a 0 HP.	5
HU-05	**Muerte de Enemigo** Como jugador, quiero ver cuando elimino a un enemigo.	1. El enemigo reproduce sonido de muerte. 2. El enemigo desaparece visualmente del escenario.	3

ID	Historia de Usuario	Criterios de Aceptación (Definition of Done)	Puntos
		3. El contador de enemigos baja.	

Sprint 2: Expansión de Contenido y Sistemas
Objetivo: Implementar la estructura del nivel completo y mejorar la IA.

ID	Historia de Usuario	Criterios de Aceptación	Puntos
HU-06	**Sistema de Estamina** Como jugador, quiero correr con Shift pero cansarme para gestionar mi huida.	1. Barra de energía visible. 2. Correr reduce la barra; dejar de correr la regenera. 3. Sin energía no se puede correr.	5
HU-07	**Enemigo a Distancia** Como diseñador, quiero un enemigo que dispare proyectiles para variar el combate.	1. IA que mantiene distancia del jugador. 2. Instancia proyectiles hacia el jugador. 3. Tiene menos vida que el enemigo básico.	8

ID	Historia de Usuario	Criterios de Aceptación	Puntos
HU-08	**Recolección de Items** Como jugador, quiero recoger cajas de munición para recargar.	1. Objeto "Caja" detectable en el suelo. 2. Al pasar por encima, aumenta munición. 3. Sonido de "Recarga" al recoger.	3
HU-09	**Diseño de Nivel 1** Como level designer, quiero construir el "Pasillo Infernal" con zonas seguras y peligrosas.	1. Importar mapa final (paredes, suelo, techo). 2. Colocar luces atmosféricas. 3. Definir puntos de spawn de enemigos.	13

Sprint 3: Jefes y Pulido Final
Objetivo: Implementar el desafío final (Boss) y cerrar el ciclo de juego.

ID	Historia de Usuario	Criterios de Aceptación	Puntos
HU-10	**Lógica de Jefe (Boss)** Como jugador, quiero enfrentar a un enemigo gigante con mucha vida al final del nivel.	1. Modelo de Jefe escalado x3. 2. Barra de vida exclusiva para el Jefe.	13

ID	Historia de Usuario	Criterios de Aceptación	Puntos
		3. Al morir, activa la victoria.	
HU-11	**Ataques Especiales de Jefe** Como jugador, quiero que el Jefe tenga patrones de ataque únicos.	1. Fase 1: Embestida física. 2. Fase 2: Lanzamiento de fuego en área. 3. Cambio visual al cambiar de fase.	8
HU-12	**Pantallas de Flujo** Como jugador, quiero menús de Inicio, Victoria y Derrota claros.	1. Botón "Jugar" inicia la escena. 2. Pantalla "You Died" permite reiniciar. 3. Pantalla "Victory" muestra estadísticas.	5
HU-13	**Audio Espacial** Como jugador, quiero escuchar de dónde vienen los enemigos.	1. Sonidos de pasos en 3D (izquierda/derecha). 2. Música dinámica que sube de volumen en combate.	5

QA y Criterios de Calidad

- Testing de Rendimiento: El juego debe correr a 60 FPS estables en un navegador Chrome estándar en una laptop de gama media.
- Testing de Jugabilidad: La dificultad debe ser "Desafiante pero Justa". El jugador promedio debe morir al menos 2 veces antes de completar el nivel por primera vez.
- Bugs Críticos (Showstoppers):
  1. El jugador atraviesa paredes (Collision failure).
  2. La munición baja de 0 (Negative integer error).
  3. El juego no carga los assets de audio.

Reporte

Para el control de calidad, cualquier error encontrado debe reportarse utilizando obligatoriamente este formato de ticket:

Ticket de Bug (Plantilla)

- **ID:** [Ej: BUG-001]
- **Gravedad:** [Alta (Rompe el juego) / Media (Molesto) / Baja (Visual)]
- **Descripción:** [Breve explicación del error]
- **Pasos para reproducir:**
  1. Ir a la zona X.
  2. Realizar acción Y.
  3. Observar el error Z.
- **Estado:** [Abierto / En Progreso / Resuelto]