

---

# Informe Técnico del Proyecto: PrimerJuego2D

## 1. Arquitectura del Proyecto

El proyecto ha evolucionado hacia una arquitectura robusta y modular, separando claramente la vista (rendering/loop) de la lógica y los datos (estado del mundo).

### Estructura General

- **Núcleo (main / nucleo):**
- **PanelJuego.java (VISTA/LOOP):** Es el contenedor gráfico (JPanel) y responsable exclusivamente del **Game Loop**.
- **Game Loop:** Mantiene la ejecución a **60 FPS** usando “Delta Time”.
- **Responsabilidad:** Delegar el `update()` a `MundoJuego` y el `draw()` a los componentes visuales. No mantiene estado del juego.
- **MundoJuego.java (MODELO/LÓGICA):** Es el **corazón** del juego. Contiene:
  - El estado actual (`gameState`: menú, jugando, pausa, game over).
  - Los arrays de entidades (`npcs`, `proyectiles`, `objs`).
  - Referencias a todos los subsistemas (`TileManager`, `DetectorColisiones`, `GestorRecursos`, `Estadisticas`).
  - Lógica central de actualización (`update()`) y orquestación.
- **Sistema de Entidades (entidad):**
  - Usa herencia y polimorfismo (`Entidad` -> `NPC` -> `Bat`, `Slime`, `Orco`, `Ghoul`).

- **Jugador:** Gestiona entrada, movimiento y estados del héroe. Incorpora el seguimiento del nivel de equipamiento para interactuar con el sistema de botín.
  - **NPC:** Clase base abstracta con IA básica, máquinas de estados y **Object Pooling**.
  - **Gestión de Recursos (mundo / tiles):**
  - **TileManager:** Dibuja el mapa estático basado en archivos de texto.
  - **GestorRecursos:** Responsable de la **generación procedural** de enemigos y del control de drops en los cofres.
- 

## 2. Sistema de Botín y Progresión de Ítems (Nueva Funcionalidad)

Los cofres ya no son entidades con un efecto estático, sino contenedores interactivos que ejecutan un sistema de *Loot* o botín progresivo al ser abiertos.

### Catálogo de Ítems (objetos)

Dentro de un cofre, el jugador puede encontrar los siguientes artefactos:

- **Consumibles Tácticos:** Bebida energética (boost de velocidad temporal), TNT (daño en área), Carbón (material/combustible).
- **Artefactos Mágicos:** Espada mágica, Libro mágico, Gema de regeneración, Anillo de regeneración.
- **Equipamiento Defensivo:** Armaduras.

### Lógica de Progresión Escalada

Los ítems de equipamiento y amuletos cuentan con un sistema de **Tiers (Niveles de calidad)**.

- **Mecánica:** El sistema evalúa el estado actual del inventario del Jugador. Si el jugador abre un cofre y el RNG (Random Number Generator) decide otorgar una armadura, el juego verifica el nivel de la armadura actual.
- **Ejemplo:** Si el jugador posee una *Armadura de Hierro*, el sistema la actualiza y genera un *drop* de una *Armadura de Oro*.

- **Impacto Arquitectónico:** Esto requiere un acoplamiento controlado donde el objeto instanciado dentro del cofre depende del estado guardado en las Estadísticas o en la clase Jugador.
- 

### 3. Generación y Gestión de Enemigos (Análisis Profundo)

El sistema ha sido reescrito para maximizar el rendimiento y escalar la dificultad.

#### A. Gestión de Memoria (Object Pooling)

El juego utiliza un **Object Pool** masivo de **1000 NPCs** (`GestorRecursos.POOL_TOTAL`).

- **Separación Suave (Soft Collisions):**
    - Se eliminaron las colisiones rígidas (cajas invisibles) entre enemigos, que causaban bloqueos y cuellos de botella.
    - Nuevo sistema de **Vectores de Repulsión**: Cada enemigo calcula un vector para alejarse de sus vecinos cercanos (usando la `GrillaEspacial`) y otro para perseguir al jugador.
1. **Cero new en Gameplay:** Se instancian al inicio. **NUNCA** se instancia un nuevo NPC durante la partida.
  2. **Ciclo de Vida:** Activación/Desactivación a través del Pool.
  3. **Beneficio:** Eliminación de lag spikes por Garbage Collection.

#### B. Algoritmo de Dificultad Dinámica

La lógica en `GestorRecursos` ajusta la dificultad en tiempo real mediante un **Sistema de Oleadas** usando una Función Escalonada para la cantidad y composición (Murciélagos, Slimes, Orcos, Ghouls).

#### C. Escalado del Jugador y Estadísticas

El jugador recibe mejoras tangibles al subir de nivel (+10 HP, +30% Curación, +2 Ataque), balanceado contra las estadísticas base de las distintas entidades enemigas.

## D. Optimizaciones de Rendimiento (Engine)

- 1. Spatial Hash Grid (`GrillaEspacial`)
- 2. Logic Culling y Frustum Culling
- 3. Rectángulos Pre-allocados para colisiones.
- 4. Reciclaje Agresivo (Desaparición) de enemigos en la retaguardia.

## 4. Estructura de Paquetes y Componentes

El proyecto está organizado en 12 paquetes especializados:

Paquete	Descripción	Contenido Clave
nucleo	Motor principal	<code>PanelJuego</code> (Loop), <code>Main</code> (Entry)
mundo	Estado y datos	<code>MundoJuego</code> (State), <code>GestorRecursos</code> (Spawning)
entidad	Actores del juego	<code>Jugador</code> , <code>NPC</code> (IA), <code>Orco</code> , <code>Slime</code> , <code>Bat</code>
interfaz	GUI y Menús	<code>InterfazUsuario</code> , <code>HUD</code> , <code>PantallaSeleccion</code>
tiles	Mapa	<code>TileManager</code> (Renderizado del mundo)
entrada	Input	<code>GestorEntrada</code> (Teclado/Mouse unificado)
colision	Física	<code>DetectorColisiones</code> (Hitbox logic)
objetos	Sistema de botín	<code>Cofres</code> (Contenedores), Clases de Ítems (Bebida, Espada, Gema, TNT, Armadura Progresiva, etc.)
audio	Sonido	<code>GestorAudio</code>
configuracion	Constantes	<code>Configuracion</code> (Resolución, Flags)

Paquete	Descripción	Contenido Clave
estadísticas	Meta-juego	Estadísticas (Nivel, XP, Highscore)
utilidades	Tools	Herramientas (Escalado de imagen), Notificación