introducción a la programación

Grupo 51

Docente:

Diego Andrés Mora Rojas
Jeff Schmidt Peralta

Integrantes

Bryan Charpentier Paredes
Isaac Solís Alvarado

28 de noviembre del 2025

1. Atributo de Análisis de Problema

Identificación del Problema Complejo de Ingeniería

El proyecto *Escapa del Laberinto* aborda un problema complejo de ingeniería centrado en el diseño e implementación de un sistema interactivo capaz de generar, gestionar y simular la experiencia de navegación en un laberinto dinámico. El reto combina elementos de matemáticas (algoritmos de grafo, generación procedural, búsqueda espacial), ciencias de la computación (simulación, máquinas de estados, IA básica para enemigos) y principios de ingeniería del software (modularidad, patrones de diseño, abstracción y reutilización).

Además, el proyecto integra aspectos de desarrollo sostenible, considerando:

- uso eficiente de recursos computacionales,
- diseño modular que facilita mantenimiento y escalabilidad,
- reutilización de componentes y algoritmos,
- buena organización del código para su evolución futura,
- enfoque en accesibilidad y experiencia del usuario.

Análisis del Contexto y Variables del Problema

El proyecto se desarrolla en el contexto de un software educativo/lúdico donde el usuario debe escapar de un laberinto generado automáticamente. Las variables críticas del problema incluyen:

1. Generación del mapa: tamaño, complejidad, transiciones entre tiles, accesibilidad.
2. Movimiento del jugador: restricciones físicas (energía), reglas del terreno, validación de posiciones.
3. Comportamiento del enemigo: estados, decisiones, interacción con el entorno.

4. Elementos del entorno: trampas, túneles, muros, lianas, zonas transitables.
5. Dificultad del juego: parámetros variables según nivel elegido.
6. Interfaz gráfica: rendimiento, renderizado del mapa, retroalimentación visual.
7. Sostenibilidad en software:
    - Arquitectura dividida en paquetes (modelo, lógica, modos, sistema, GUI).
    - Reutilización de clases.
    - Reducción de acoplamiento mediante interfaces y abstracciones.
    - Facilita mantenimiento a largo plazo.

Plan de Solución Integrado con Sostenibilidad

La solución se basa en un modelo orientado a objetos, que divide el sistema en módulos especializados:

1. Generación procedural del laberinto
   – Desarrollada mediante la clase GeneradorMapa, con parámetros de dificultad.
   – Optimiza uso de memoria al representar el mapa como matriz de Tile.
2. Modelo del mundo del juego
   – Entidades como Jugador, Enemigo, Trampa, Mapa contienen lógica independiente.
   – Uso de clases abstractas (Tile) y enums para claridad y control del estado.
3. Simulación y modos de juego
   – GameModeEscapa y GameModeCazador integran jugador, mapa y enemigos.
   – Favorece sostenibilidad mediante separación de responsabilidades (SRP).

4. Interfaz gráfica modular
   – PantallaBase y sus derivadas simplifican añadir nuevas pantallas.
   – RenderizadorMapa desacopla la lógica del renderizado.
5. Sistema de puntuaciones
   – ScoreBoard y Puntaje aseguran trazabilidad de resultados.

El proyecto utiliza principios de ingeniería sostenible al priorizar:

- mantenibilidad,
- modularidad,
- escalabilidad,
- bajo acoplamiento,
- reutilización de componentes,
- claridad del diseño.

Evaluación de Soluciones (Pros y Contras)

Solución	Pros	Contras
Generación procedural basada en objetos (Mapa + Tile)	Reutilizable, escalable, baja memoria, fácil agregar nuevos tipos de Tile.	Complejidad inicial mayor.
IA de enemigos basada en estados	Código limpio, predecible, fácil extender estados (huir, perseguir).	IA más simple frente a enfoques avanzados.
Arquitectura por paquetes	Claridad, orden, sostenibilidad a largo plazo, facilita pruebas.	Más archivos y clases para gestionar.
Pantallas modulares	Fácil cambiar GUI sin tocar lógica del juego.	Requiere disciplina de diseño.

Solución	Pros	Contras
Sistema de trampas y dificultad configurable	Aumenta rejugabilidad, se adapta a diferentes usuarios.	Mayor interacción entre módulos.

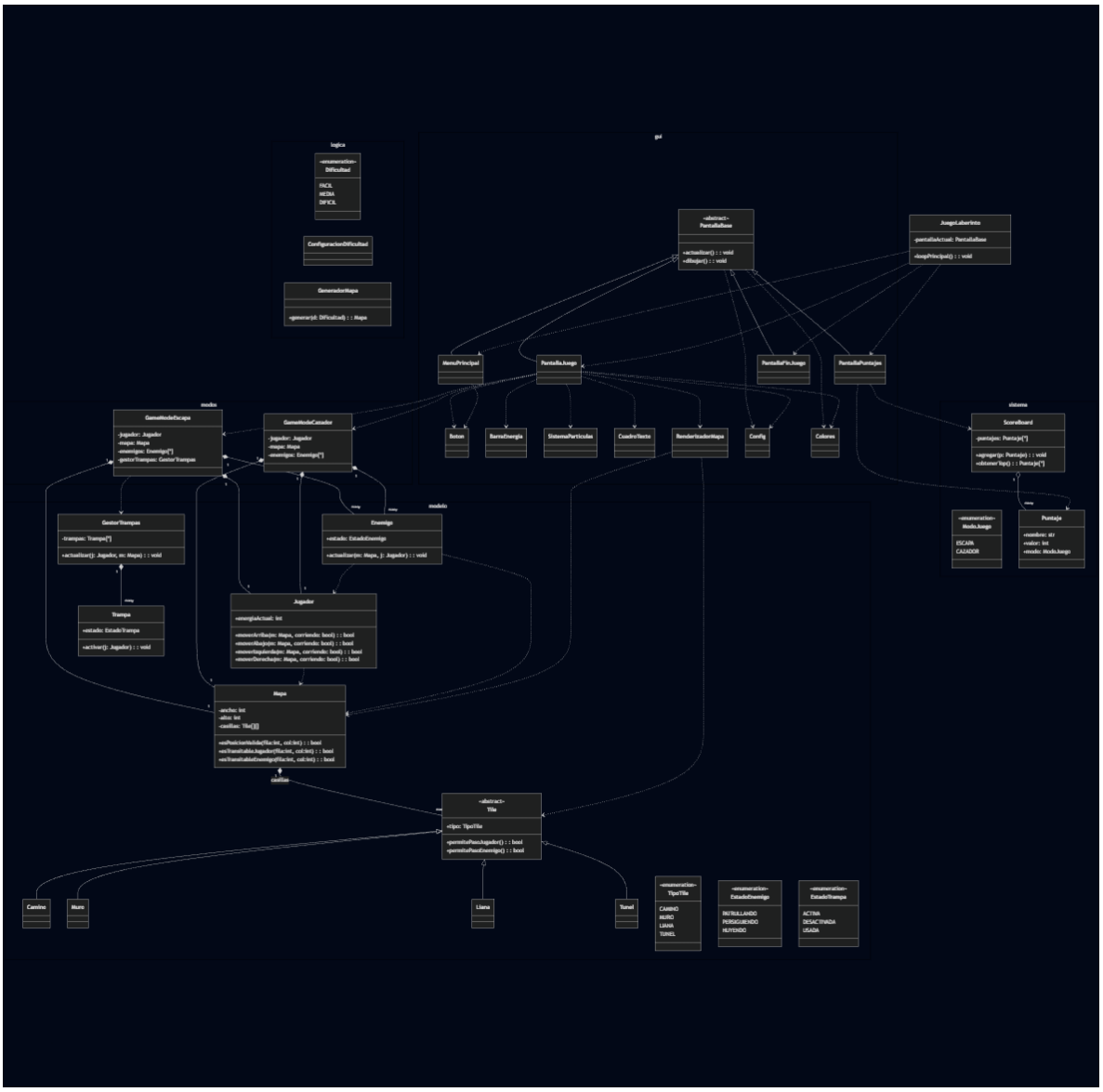
2. Atributo de Herramientas de Ingeniería

Técnicas, Recursos y Métodos Utilizados

Para resolver el problema complejo de ingeniería se aplican:

- Programación Orientada a Objetos (POO)
  – Encapsulación, herencia, polimorfismo, abstracción.
  – Clases como Tile, Mapa, Jugador, Enemy, GameModeEscapa.
- Modelado UML
  – Diagrama de clases para visualizar relaciones y dependencias.
- Algoritmos y estructuras matemáticas
  – Matrices para representar el mapa.
  – Algoritmos de búsqueda espacial.
  – Uso de máquinas de estados para enemigos.
- Principios SOLID
  – Especialmente: SRP, OCP y LSP.
- Diseño modular por paquetes
  – modelo, logica, modos, sistema, gui, raíz.
- Generación procedural (sistema sostenible de creación de niveles).
- Uso de enums para mejorar mantenibilidad y evitar valores mágicos.
- Simulación de sistemas (movimiento, energía, colisiones).

Diagrama de Clases del Modelo de Objetos



Cómo se aplican las herramientas en el proyecto

- POO facilitó separar responsabilidades y reutilizar funciones.
- UML + Modularización permitió agregar nuevas funciones (como Renderizador, Túneles, Pantallas nuevas) sin romper código viejo.
- Enums y Clases Abstractas hacen el sistema sostenible y expandible.
- Máquina de estados en Enemigo permitió IA clara, comprensible y eficiente.

Adaptación de Herramientas Durante el Desarrollo

Durante el proyecto se requirió adaptar técnicas:

- Ampliar el diseño inicial al agregar trampas, lo cual se resolvió extendiendo GestorTrampas.
- Optimizar el renderizado separando RenderizadorMapa de PantallaJuego.
- Incorporar nuevos tipos de Tile sin romper el sistema, aprovechando la herencia.
- Ajustar niveles de dificultad manipulando parámetros en ConfiguracionDificultad.
- Rediseñar relaciones débiles o ambiguas al observar el diagrama UML.
- Modularizar el sistema de pantallas para permitir navegabilidad limpia entre estados del juego.

Estas adaptaciones se hicieron preservando principios de ingeniería sostenible: mantenibilidad, claridad, divisibilidad, escalabilidad y orden lógico del software.