Documentación del Proyecto 2: Escapa del Laberinto

Grupo:
Juan Pablo Zumbado
Pablo Fuentes

Semestre II, 2025

1. Atributo de Análisis de Problema

1.1 Indique de manera clara cuál es el problema complejo de ingeniería.
El problema consiste en desarrollar un videojuego 2D con dos modos (Escapa y Caza) sobre un laberinto aleatorio, garantizando que siempre exista solución, que las colisiones y reglas funcionen correctamente y que la dificultad sea jugable. Se integran probabilidad (para terrenos especiales), geometría discreta (mapa en matriz), movimiento en tiempo real y programación orientada a objetos. El enfoque sostenible se refleja en un diseño modular, eficiente y fácil de ampliar.

1.2 Realice un análisis del contexto y de las variables relacionadas con el problema.
El juego se crea como proyecto académico y se publica en GitHub. El mapa es una matriz 7×14 con IDs de casillas (camino, entrada, salida, trampa, liana, túnel y muro). El laberinto se genera con un algoritmo tipo Prim aleatorizado: se inicia con muros, se excavan caminos conectados usando una lista de paredes vecinas y se asegura conectividad. Luego, solo los muros restantes pueden convertirse en túneles o lianas con probabilidad 0,14 cada uno. Las restricciones dependen del modo: en Escapa el explorador no pasa por lianas/muros y el cazador no pasa por túneles/muros; en Caza se invierten roles. El puntaje es por tiempo en Escapa y por capturas (+200) en Caza. El flujo del juego se controla por estados (inicio, ingreso, podio, partida y final).

1.3 Describa un plan de solución para el problema complejo de ingeniería.
Se modelan casillas y personajes con POO. Se genera el laberinto con Prim aleatorizado y se asignan entrada/salida en esquinas opuestas; después se transforman algunos muros en túneles o lianas. Para cada modo se instancian personajes distintos y se calculan obstáculos según su rol. Finalmente se aplican reglas de victoria/derrota y se actualiza el podio con los puntajes obtenidos.

1.4 Evalúe pros y contras de las soluciones planteadas.
Prim aleatorizado: garantiza solución y produce mapas más variados; como contra, requiere calibración para evitar zonas simples. **Probabilidades fijas de túnel/liana:** ajustan dificultad sin romper el mapa, pero porcentajes altos pueden desbalancear. **Dos modos con reglas distintas:** aumenta rejugabilidad, aunque la IA sigue siendo básica y puede volverse predecible.

2. Atributo de Herramientas de Ingeniería

2.1 Explique las técnicas, recursos, herramientas o métodos que se utilizan.
Se usa Python 3 con Pygame para gráficos, eventos, colisiones y control de frames, y pygame_gui para el ingreso de nombre. El diseño es POO con herencia para casillas y personajes. El mapa se construye con Prim aleatorizado y probabilidades. Git/GitHub se usa para control de versiones y entrega.

2.2 Realice un diagrama de clases del modelo de objetos utilizado (formato textual).

```
Casilla
- x, y, ancho, altura, colision, splash, type_id
  Subclases: Camino, Entrada, Salida, Trampa, Liana, Tunel, Muro

Personaje
- x, y, sprite, forma
- mover_personaje(), imprimir_personaje()
  Subclases:
   - UExplorador (jugador Escapa)
   - UCazador (jugador Caza)
   - EnemigoCazador: seguir_jugador(), tocar_personaje()
   - EnemigoExplorador: buscar_salida(), tocar_personaje()

Usuarios
- nombre_usuario, puntuacion
- asociado al personaje jugable
```

2.3 Indique cómo aplicar las técnicas, recursos, herramientas o métodos.
El mapa numérico se convierte a objetos con `generar_matriz_mapa`. Se generan obstáculos según rol con `obst_escapa` y `obst_cazador`. En Escapa el enemigo persigue al jugador y se gana al llegar a salida; en Caza el explorador intenta escapar y el cazador suma puntos al capturarlo. El podio se actualiza al final de cada partida.

2.4 Explique cómo adapta las técnicas, recursos, herramientas o métodos durante el desarrollo.
Se reemplazó la generación anterior por Prim para mejorar la calidad del laberinto. Se añadieron clases nuevas para soporte completo del modo Caza, reutilizando la base POO. También se ajustaron velocidades, obstáculos y puntaje para balancear ambos modos, manteniendo el proyecto modular y sostenible.

Conclusión: El juego implementa dos modos funcionales sobre un laberinto resoluble y variado, usando POO, algoritmos aleatorizados y control de estados con bajo costo computacional.