DOCUMENTO DEL PROYECTO 2 – INTRODUCCIÓN A LA PROGRAMACIÓN

TEC | Tecnológico de Costa Rica

Proyecto 2 – Documento de Ingeniería

Este documento explica, de manera sencilla y natural, cómo analicé el problema del proyecto, qué decisiones tomé para resolverlo y qué herramientas utilicé durante el desarrollo. La redacción intenta ser clara y humana, como la escribiría un estudiante universitario.

---

1. Análisis del Problema

El objetivo del proyecto es construir un juego estilo laberinto con dos modos: Escapa y Cazador. Aunque al inicio parece un proyecto simple, implica trabajar con varios conceptos a la vez: generar un mapa aleatorio, manejar movimiento del jugador, programar enemigos que reaccionan, administrar energía, colocar trampas y guardar puntajes entre partidas.

Lo primero que analicé fue la estructura del terreno. Las casillas del mapa podían ser cuatro tipos diferentes: camino, muro, túnel o liana. Cada una afecta el movimiento de forma distinta, por lo que estas reglas influyen directamente en la estrategia del jugador y los enemigos.

Otro punto importante fue que el mapa debía ser aleatorio, pero no caótico. En las primeras pruebas el jugador y los enemigos quedaban atrapados en zonas sin salida. Por eso decidí equilibrar las probabilidades para que hubiera más caminos libres y menos casillas complicadas.

También consideré la claridad visual del juego. Como el proyecto es por consola, era importante que el jugador pudiera reconocer fácilmente cada casilla y la posición de los enemigos.

---

2. Plan de Solución

Para organizar mejor el programa, utilicé Programación Orientada a Objetos. Cada tipo de casilla (camino, muro, túnel y liana) tiene su propia clase. El jugador, los enemigos y el sistema de trampas también son clases independientes. Esto ayuda a mantener todo ordenado y fácil de entender.

El mapa se maneja como una matriz donde cada posición es un objeto que representa una casilla. Ajusté la generación aleatoria para que el mapa fuera jugable: 70% caminos y el resto distribuido entre muros, túneles y lianas.

El movimiento del jugador es directo, mientras que los enemigos toman decisiones basadas en la distancia: en modo Escapa persiguen al jugador, y en modo Cazador huyen de él. No usé algoritmos avanzados porque no eran necesarios para este nivel del curso.

El sistema de trampas funciona con un límite de trampas activas y un tiempo de espera antes de poder colocar otra. Para mantener puntajes y jugadores entre sesiones utilicé archivos JSON porque son simples y fáciles de manejar en Python.

---

3. Evaluación de la Solución

El proyecto final cumple con los requisitos principales: dos modos de juego funcionales, mapa aleatorio equilibrado, sistema de trampas, puntajes guardados y enemigos que reaccionan al jugador.

Entre los aspectos positivos destaco que el código quedó modular, ordenado y sencillo de actualizar. Además, la mejora en la distribución del mapa hizo que el juego se volviera más jugable.

Una desventaja es que los enemigos no utilizan búsqueda de rutas avanzada, por lo que a veces toman caminos poco eficientes. Sin embargo, esto está dentro del alcance del curso y funciona suficientemente bien para los objetivos del proyecto.

---

4. Herramientas y Técnicas Utilizadas

- Programación Orientada a Objetos para organizar casillas, jugador, enemigos y trampas.
- Uso de matrices para manejar el mapa del terreno.
- Aleatoriedad controlada para generar el mapa equilibrado.
- Archivos JSON para almacenar puntajes y jugadores registrados.
- GitHub para control de versiones, manteniendo commits frecuentes y mensajes claros.

Durante el desarrollo hice ajustes basados en las pruebas: cambié la probabilidad de aparición de cada casilla, simplifiqué la visualización del mapa y organicé mejor las clases para que fueran más fáciles de entender.

C A S I L L A (abstracta)

- tipo : str

+ puede_pasar_jugador() : bool

+ puede_pasar_enemigo() : bool

. * U L

Camino Muro Tunel Liana

+ puede_pasar_jugador() : bool (solo Camino y Tunel)

+ puede_pasar_enemigo() : bool (solo Camino y Liana)

J U G A D O R

- fila : int

- columna : int

- energia : int

- energia_maxima : int

+ mover(direccion, mapa, correr=False)

+ recuperar_energia()

ENEMIGO

- fila : int
- columna : int

+ mover(direccion, mapa)

SISTEMA DE TRAMPAS

- trampas : list
- tiempo_ultima_colocacion : float
- cooldown : int
- max_trampas : int

+ colocar_trampa(fila, columna)
+ verificar_enemigos(lista_enemigos) : list

M A P A

(la clase no existe formalmente, pero el sistema actúa así)

- matriz : lista de listas de Casilla

+ generar_mapa()

+ crear_camino_principal()

+ imprimir_mapa(jugador, enemigos, trampas)

J U E G O

(implementado mediante funciones)

+ modo_escapa(nombre)

+ modo_cazador(nombre)

+ menu_principal()

+ registrar_jugador()

+ guardar_puntaje()

+ cargar_puntajes()

+ mostrar_puntajes()

RELACIONES EN EL DIAGRAMA

- Jugador → depende del Mapa para moverse
- Enemigo → depende del Mapa y del Jugador (perseguir/huir)
- SistemaTrampas → interactúa con Jugador y Enemigos
- Modo Escapa / Cazador → controla el flujo del juego
- Puntajes → persiste datos del jugador