Instituto Tecnológico de Costa Rica

Unidad de Computación

Juego: Escapa del Laberinto

Introducción a la programación

Johan Josué Araya Rojas y Santiago Víquez Zamora

GR51, Sede San Carlos

28 de noviembre, 2025

Atributo de Análisis de Problema

a. Identificación del problema complejo y desarrollo sostenible

El problema fue diseñar un juego 2D donde un jugador y varios cazadores se mueven en tiempo real dentro de un laberinto distinto en cada partida. El mapa es una matriz de celdas y siempre debe existir al menos un camino desde el inicio hasta la salida, mientras el sistema controla energía, trampas, enemigos, puntaje, tiempo y dos modos de juego sin que la interfaz se congele. Lo relacionamos con desarrollo sostenible en dos sentidos: en la mecánica, el jugador administra un recurso limitado (la energía) y decide cuándo correr o caminar; en el desarrollo, buscamos un código modular y reutilizable a largo plazo, de modo que en el futuro se puedan agregar terrenos, reglas o demás características sin empezar de cero o usar sus métodos en otros programas que les sean de utilidad.

b. Análisis del contexto y variables

El juego se ejecuta en un entorno aleatorio: cada partida genera un laberinto distinto, de modo que la lógica debe funcionar para cualquier distribución de caminos, muros y terrenos especiales. Las variables centrales son la matriz del mapa, las posiciones del jugador y de los enemigos y la forma en que medimos distancias; para simplificarlo usamos coordenadas enteras y una distancia basada en cuántas casillas difieren en filas y columnas.

También influyen el tiempo de juego, la energía disponible, el número y la velocidad de enemigos y la dificultad elegida, que ajusta qué tan exigente es la partida. A nivel de software nos importa la modularidad para mantener el código legible, lo cual promueve la sostenibilidad técnica al facilitar el mantenimiento futuro y reducir el desperdicio de recursos en reprogramación.

c. Plan de solución

La solución se planteó dividiendo el problema en partes y aplicando Programación Orientada a Objetos. Definimos la clase Mapa, encargada de generar el laberinto de forma automática: se parte de una matriz llena de muros y, desde un punto inicial, se va “abriendo paso” de forma recursiva para formar los pasillos y una salida conectada, de modo que el mapa siempre sea jugable.

Luego creamos una clase base Casilla y varios tipos de terreno (Camino, Muro, Liana, Tunel), cada uno con reglas sobre quién puede entrar. Para las entidades definimos las clases Jugador y Enemigo: el jugador gestiona energía, fatiga y trampas, y los enemigos usan la información del mapa para decidir hacia dónde moverse según el modo de juego. Finalmente, la clase JuegoTK coordina mapa, entidades, puntaje e interfaz gráfica, y MenuPrincipal se encarga del menú inicial, la selección de dificultad y las tablas de puntajes.

d. Pros y Contras

Entre los aspectos positivos, el código quedó organizado por módulos: mapa, entidades, juego y menú están separados, lo que facilita entender y modificar el proyecto, la generación automática de laberintos aumenta la rejugabilidad y evita diseñar niveles a mano, y el uso de estructuras sencillas, como la matriz de casillas y los recorridos sobre ella para los enemigos, permite que el juego corra de forma fluida.

Como desventajas, la clase principal del juego concentra mucha lógica, lo que podría complicar su mantenimiento si el proyecto creciera más. Además, el uso de Tkinter limita el apartado gráfico a una estética sencilla en 2D, y la inteligencia de los enemigos, aunque adecuada para el curso, sigue siendo básica, de hecho para el sonido nos vimos obligados a ayudarnos de Pygame por las limitaciones de Tkinter.

Atributo de Herramientas de Ingeniería

a. Técnicas, recursos y herramientas utilizadas

Utilizamos Python como lenguaje principal, para la interfaz gráfica empleamos Tkinter y para la música y efectos de sonido, Pygame debido a las limitaciones que nos presentó Tkinter. El diseño se basó en POO, creando clases para el mapa, los tipos de casilla, el jugador, los enemigos y la lógica del juego, la generación del laberinto usa un algoritmo recursivo que va abriendo caminos sobre la matriz, mientras que el movimiento de los enemigos se apoya en una búsqueda por capas para encontrar rutas válidas respetando los muros y tomando en cuenta al jugador, y como herramienta de apoyo usamos GitHub para el control de versiones y para registrar el avance del trabajo en equipo.

b. Aplicación de las técnicas y métodos

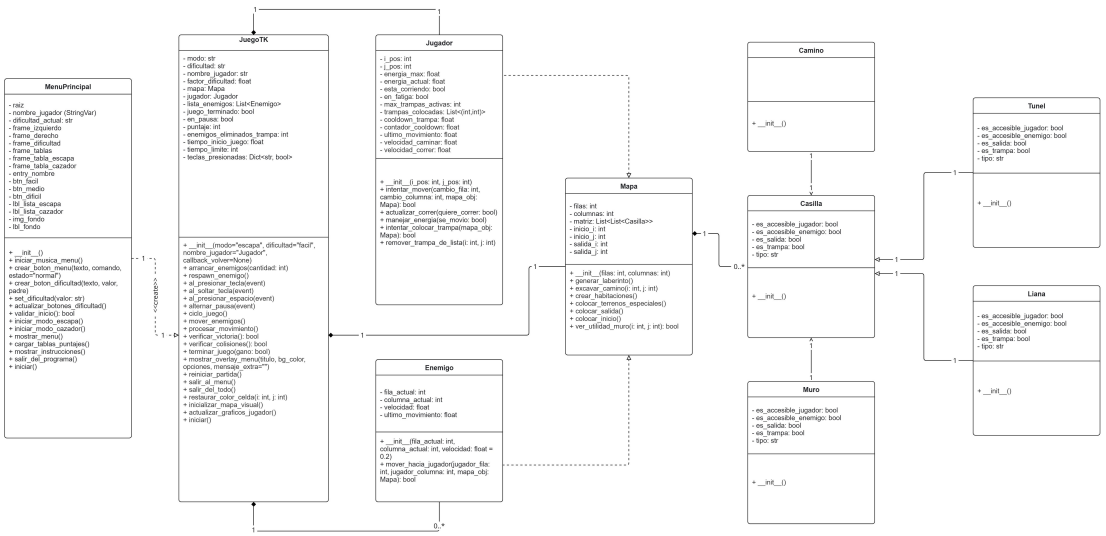
En la ejecución, la clase JuegoTK crea el mapa, el jugador y la lista de enemigos, y utiliza el método after de Tkinter para actualizar periódicamente el juego sin bloquear la ventana, el mapa se refleja en una grilla de Frame coloreados según el tipo de terreno, y se actualiza cuando el jugador o los enemigos se mueven, mientras que las teclas se manejan mediante eventos de Tkinter como KeyPress y KeyRelease, cambiando banderas internas que luego se leen en el ciclo de juego para mover al personaje, gastar o recuperar energía y colocar trampas. El algoritmo recursivo del Mapa se ejecuta al inicio de la partida; después, los enemigos van recalculando caminos sobre esa misma estructura cuando necesitan perseguir o huir.

c. Adaptación de técnicas durante el desarrollo

Durante el desarrollo hicimos varios ajustes:

- El movimiento de enemigos empezó siendo casi aleatorio y luego se mejoró usando búsqueda por capas, agregando algo de aleatoriedad para que los caminos no fueran siempre idénticos.
- En el modo Cazador reutilizamos la misma técnica de búsqueda, pero cambiando el objetivo: normalmente se dirigen a la salida; si el jugador está muy cerca, se busca una zona lejana para simular que huyen de él.
- Ajustamos varias veces las probabilidades de aparición de túneles y lianas, así como los tiempos de energía, trampas y respawn de enemigos, hasta encontrar un balance razonable entre dificultad y jugabilidad.

d. Diagrama de clases del modelo de objetos



Ver en alta calidad en cualquiera de los siguientes enlaces adjuntos:

- Web: Diagrama de clases url
- Imagen: Diagrama de clases UML.jpg