TEC | Tecnológico de Costa Rica
Anthony Fabricio Montiel López - 2025132603
Luis Diego Murillo - 2025069058
Ingeniería en computadores - Documentación proyecto Laberinto

DOCUMENTACIÓN DEL PROYECTO

1 - Atributo de Análisis de Problema

Problema complejo de ingeniería

El sistema aborda el diseño de un entorno interactivo en tiempo real donde un jugador y un conjunto de cazadores se desplazan dentro de un laberinto generado aleatoriamente. El problema combina varios principios fundamentales: matemáticos (uso de estructuras matriciales, recorrido en grafos mediante BFS y DFS, generación pseudoaleatoria), ciencias naturales (modelos de energía limitada y recuperación gradual), y ciencias de la ingeniería (modularidad orientado a objetos, manejo de eventos, control de estados, sincronización entre procesos). Todo esto debe coexistir bajo un enfoque sostenible, privilegiando algoritmos eficientes y una arquitectura mantenible, reduciendo la complejidad futura del sistema.

La dificultad radica en asegurar que cada partida produzca un mapa funcional, que los enemigos tomen decisiones coherentes bajo reglas distintas para cada modo, que el jugador pueda desplazarse sin bloqueos, y que las trampas, salidas y puntajes operen correctamente sin afectar la estabilidad del juego.

Análisis del contexto y variables del problema

El laberinto está representado por una matriz donde cada celda define restricciones particulares: caminos, muros, túneles y lianas. El jugador y los enemigos se ven afectados de manera distinta por estos terrenos, por lo que deben existir reglas claras de desplazamiento. La generación del mapa debe asegurar siempre un camino válido, obligación que introduce algoritmos de validación y corrección automática.

Plan de solución y principios de sostenibilidad

El plan para resolver el problema se basa en una arquitectura orientada a objetos que separa responsabilidad entre módulos: Mapa, Jugador, Enemigo, Terrenos, SoundManager, SpriteManager y el controlador JuegoApp. La generación del laberinto utiliza DFS para crear rutas y BFS para verificar conectividad, asegurando un mapa funcional sin intervención manual. Se integran mecanismos de manejo de energía, detección de colisiones, temporizadores y respawn de enemigos para garantizar jugabilidad estable.

Desde la sostenibilidad del software, se priorizó evitar redundancias, minimizar cálculos costosos y mantener las clases independientes. Esto permite que futuras modificaciones no requieran rehacer el sistema completo. Asimismo, la interfaz modular de Tkinter evita ventanas nuevas e innecesarias, permitiendo reutilizar frames y optimizar recursos.

Evaluación de soluciones: pros y contras

Pros:
– Mapas siempre funcionales gracias al uso conjunto de DFS y BFS.
– Arquitectura clara y escalable basada en POO.
– IA diferenciada para cada modo, lo que aumenta la profundidad del juego.
– Sistema de trampas y energía que añade estrategia sin aumentar complejidad.
– Manejo eficiente de recursos: sonidos, imágenes y lógica de juego están encapsulados.

Contras:
– Tkinter no ofrece advertencias cuando los callbacks se registran mal, causando fallos silenciosos.
– Los enemigos pueden atascarse en configuraciones específicas del laberinto, requiriendo heurísticas adicionales.
– La reproducción de audio depende del hardware; equipos lentos pueden desincronizar sonido y movimiento.
– El sistema de puntajes puede generar registros corruptos si no se valida el archivo antes de escribir.

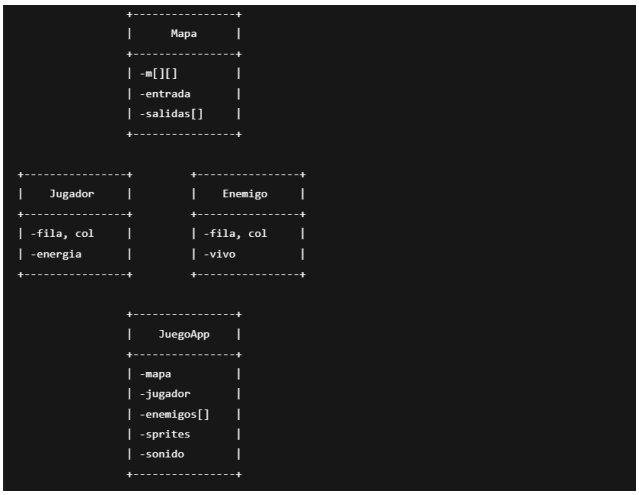
Estos aspectos se mitigaron en su gran medida mediante pruebas continuas, refactorización de funciones y ajustes a la lógica de movimiento, cargas de recursos y validaciones, exceptuando el sistema de trampas y energía, que no se implementó, si existe dentro de la lógica, solo que no es funciona, al menos no del todo.

2-Atributo de Herramientas de Ingeniería

Técnicas, recursos, herramientas y métodos utilizados

El proyecto utiliza Python y Tkinter como base del desarrollo, complementado con random, time y collections.deque. Se aplican estructuras matriciales, colas FIFO, programación orientada a objetos, algoritmos DFS/BFS, manejo de eventos por teclado, temporización en tiempo real mediante after() y gestión modular de sprites y sonido. Estas herramientas se ajustan a la naturaleza del problema complejo porque permiten controlar múltiples elementos dinámicos dentro de un ambiente interactivo.

Diagrama del código:



Aplicación de herramientas en el proyecto

La matriz del mapa se construyó usando listas anidadas y DFS para generar sus caminos. BFS se empleó para buscar rutas válidas y para la persecución enemiga. Tkinter controla ventanas, canvas y eventos, mientras que after() implementa el ciclo del juego sin usar bucles bloqueantes. El sistema de puntajes usa archivos de texto y lectura ordenada para mantener registros fiables. El gestor de sprites y de sonido carga elementos externos y los encapsula para simplificar su uso dentro del controlador principal.

Adaptación de técnicas, recursos, herramientas y métodos

Durante el desarrollo fue necesario adaptar diversas técnicas para resolver problemas surgidos al integrar lógica, interfaz e imágenes. La creación de ventanas en Tkinter requirió reorganizar frames, ajustar posiciones, corregir callbacks y asegurarse de que cada función estuviera correctamente invocada mediante self. Un llamado mal definido no genera errores visibles, por lo que muchos fallos debieron identificarse mediante pruebas manuales.

También se adaptó el manejo de imágenes y audio: fue necesario implementar un sistema de carga flexible para evitar fallos silenciosos cuando un archivo no existía o tenía dimensiones incompatibles. El SpriteManager y SoundManager surgieron como respuesta a estos problemas.

En el modelo de juego, Jugador fue sencillo de integrar, pero Enemigo debió ajustarse varias veces. Su IA combinaba persecución y huida dentro de la misma clase, lo cual generaba conflictos cuando ambos comportamientos interactuaban. Se agregaron mecanismos anti-atascos y validaciones adicionales antes de mover cada enemigo. Estas pruebas se realizaron primero en entornos separados para evitar perturbar la interfaz completa.

El controlador JuegoApp también exigió ajustes: la lógica del registro y lectura de puntajes debía evitar sobrescrituras involuntarias; la presentación de ventanas debía reestructurarse para que no desaparecieran elementos al cambiar de modo; los botones debían envolver correctamente las funciones para garantizar sonido y acción; y la sincronización entre sonido, colisiones, energía y spawn de enemigos requirió reorganizar varios métodos.

Problemas encontrados

Problemas	Solución
No dejaba hacer commits	Guardar y después hacer el commits
Asignación de tecla de velocidad	No encontramos solución
Objetos del mapa se contraponían uno sobre el otro	Cambio de atributos en la clase laberinto
Movimientos en los enemigos	Cambios en las funciones que se encargaban del movimiento
Fusión de las ramas con el main	Hacerlo de forma manual
Dificultad de encontrar gifts de enemigos y jugador	Dibujar los gifts de enemigod y jugador
Reproducción de audios por ser largos	Recortar los audios para no tener problemas y no se escuchen audios sobre otros audios

Cada fallo requería revisión antes de continuar, por cerca que estuviera la solución final siempre fue necesario revisar. Base a esta situación se denota una mejora constante.