DOCUMENTACIÓN DEL PROYECTO – JUEGO "ESCAPA / CAZADOR"

1. Atributo de Análisis de Problema

El proyecto consiste en el diseño y construcción de un videojuego interactivo desarrollado en Python con Tkinter, cuyo propósito es simular un entorno donde un jugador debe escapar o cazar enemigos dentro de un laberinto generado aleatoriamente. El problema complejo de ingeniería que se aborda implica combinar generación de mapas, inteligencia artificial básica para enemigos, control de energía, interacción en tiempo real y un objetivo dependiendo del modo.

Desde una perspectiva matemática e ingenieril, el proyecto requiere garantizar que cada mapa generado contenga **al menos un camino válido entre el jugador y la meta**. Además, se integran elementos de ciencias de la computación como la simulación de movimiento, prioridades de colisión, gestión de eventos en tiempo real y uso eficiente de estructuras de datos. Desde el punto de vista del desarrollo sostenible, se consideran prácticas que minimicen el uso innecesario de recursos computacionales, como mantener una interfaz ligera, evitar cómputos repetitivos y asegurar que la inteligencia artificial opere con reglas simples y optimizadas.

El análisis del contexto revela que el usuario interactúa con un entorno donde las decisiones deben ser rápidas y la dificultad se ajusta mediante elementos como energía, velocidad, trampas y patrones de movimiento. Las variables clave incluyen el tamaño del mapa, la cantidad de enemigos, sus velocidades, la tasa de regeneración de energía y la ubicación de casillas especiales (túneles y lianas). Integrar sostenibilidad implica priorizar la simplicidad computacional, optimizar tiempos de actualización en pantalla y evitar estructuras que recarguen la CPU, manteniendo así accesibilidad en computadoras con recursos limitados.

El plan de solución se fundamentó en dividir el problema en componentes independientes:

- Generación de mapa: Matriz numérica con cuatro tipos de casilla (camino, muro, túnel, liana). El sistema asegura que la ruta principal sea transitable.
- Movimiento del jugador: Física básica con velocidad normal y modo de correr, limitado por una barra de energía regenerativa.
- Inteligencia artificial de enemigos: Persiguen al jugador en modo Escapa y huyen de él en modo Cazador, respetando restricciones de terreno.
- Gestión de trampas y temporizadores: Máximo 3 trampas activas, recarga de 5 s y respawn del enemigo en 10 s.
- Sistema de puntajes: Se evalúa tiempo, enemigos atrapados, penalizaciones y bonificaciones dependiendo del modo.
- Interfaz gráfica: Construida con Tkinter, ligera, accesible y adaptable a pantallas estándar.

En términos de sostenibilidad, la solución eligió un enfoque simple y eficiente: el movimiento enemigo se basa en comparaciones directas, el mapa usa enteros para minimizar memoria y la interfaz evita animaciones costosas para priorizar la accesibilidad.

En cuanto a los pros y contras de la solución:

Pros

- Sistema de mapa ligero y sostenible.
- Evita cálculos pesados y usa IA simple, lo que disminuye consumo energético.
- Interfaz accesible sin requerir librerías externas.
- Balance adecuado entre dificultad, jugabilidad y claridad.

Contras

- La calidad gráfica es básica debido a la optimización.
- IA sencilla: no utiliza pathfinding avanzado por razones de eficiencia.
- El laberinto depende de un generador pseudoaleatorio, lo que requiere validación adicional.

En conclusión, el proyecto resuelve el problema mediante técnicas de ingeniería efectiva, diseño modular, sostenibilidad computacional y mecánicas claras y equilibradas para el usuario final.

2. Atributo de Herramientas de Ingeniería

Para resolver el problema complejo planteado, se utilizaron herramientas y recursos acordes con el diseño modular del sistema. El lenguaje principal fue **Python**, apoyado por **Tkinter** para la interfaz gráfica, lo que permite una implementación multiplataforma, eficiente y con bajo consumo de recursos. La representación del mapa como una matriz de enteros facilitó manipulación rápida, detección de colisiones y conservación de memoria.
El proyecto integra técnicas como:

- Algoritmos de búsqueda (DFS) para validar caminos transitables.
- Uso de clases para modelar los elementos clave del juego: jugador, enemigo, trampa, casilla y mapa.
- Temporizadores de Tkinter para animación ligera y control de eventos.
- Manipulación de estados (modo Escapa / Cazador) para modificar reglas del juego dinámicamente.
- Interfaz minimalista para favorecer accesibilidad y eficiencia energética.

Durante el desarrollo, las herramientas se adaptaron conforme se identificaban problemas: el sistema de túneles y lianas se diseñó inicialmente con reglas fijas, pero posteriormente se ajustó para invertir roles en modo Cazador, mejorando la coherencia del gameplay. El generador de mapas también se refinó para asegurar rutas completables, respondiendo a pruebas de jugabilidad.

El control de versiones se hizo mediante **GitHub**, por favor leer el archivo de texto del repositorio, documentación interna y estructura ordenada de carpetas. Esto aseguró trazabilidad, transparencia y sostenibilidad del trabajo colaborativo.

En conjunto, las herramientas y técnicas empleadas permiten un proyecto robusto, sostenible, accesible y claro tanto para jugadores como para futuros desarrolladores que requieran extender el sistema.