

1. Atributo de Análisis de Problema:

El problema complejo de ingeniería en el proyecto plantea como problema complejo el diseño y construcción de un videojuego interactivo capaz de generar, gestionar y simular un laberinto, donde intervienen distintos enemigos con comportamientos distintos y un usuario o jugador.

Para esto es necesario aplicar principios de matemáticas como: matrices, cálculos de movimiento, ciencias naturales como: modelos simples de energía y desplazamiento y ciencias de la ingeniería como: modularidad, sistemas dinámicos, y optimización, con el propósito de generar un videojuego entretenido y fácil de utilizar, no obstante, también se busca la integración de un enfoque de desarrollo sostenible mediante la optimización de recursos computacionales y el diseño modular, con la intención de reutilizar elementos sin generar procesos innecesarios y obtener un rendimiento adecuado en cuanto a consumo energético y eficiencia.

Análisis del contexto y variables

El juego debe funcionar sobre un mapa representado por una matriz, donde cada casilla tiene reglas de movimiento distintas. Las variables más relevantes son: Tipos de terreno: afectan accesibilidad y rutas posibles. Comportamiento del jugador y enemigos: incluyen persecución, huida, energía y colisiones. Generación aleatoria del mapa: debe ser funcional y eficiente. Dificultad y puntajes: dependen de tiempos, velocidad y cantidad de cazadores.

Y para una integración del desarrollo sostenible, en el proyecto se buscó minimizar el uso innecesario del procesador, reutilizar elementos, a excepción de los exclusivos de cada modo de juego, para evitar código duplicado y garantizar un funcionamiento fluido incluso con varios enemigos interactuando a la vez.

Plan de solución

El plan de solución consiste en construir un sistema modular donde el mapa, los tipos de terreno, el jugador y los enemigos funcionen como componentes independientes, facilitando el mantenimiento y la sostenibilidad del proyecto. El mapa se genera de manera aleatoria, pero se garantiza que siempre exista un camino válido desde el inicio hasta la salida mediante la creación previa de una ruta directa sobre la cual no se colocan muros.

A partir de este camino, el resto del laberinto se forma con patrones equilibrados de muros, túneles y lianas, evitando sobrecargar el procesamiento. El movimiento del jugador y de los enemigos se gestiona con operaciones simples de distancia y dirección, junto con temporizadores para energía, trampas y reapariciones, lo que permite un flujo fluido sin consumo excesivo de recursos. Y así, la solución integra principios de ingeniería y sostenibilidad al optimizar el rendimiento, asegurar la reutilización del código y garantizar la buena eficiencia en el juego.

Evaluación de las soluciones (pros y contras)

Las soluciones planteadas ofrecen ventajas importantes, como una estructura modular que facilita el mantenimiento y reutilización del código, y una generación de mapa optimizada, que favorece la eficiencia energética y la sostenibilidad del sistema. El movimiento y las mecánicas del juego se manejan con lógica simple, lo que garantiza un buen rendimiento. Sin embargo, estas decisiones también tienen desventajas: los laberintos pueden ser menos variados y el balance de dificultad puede necesitar ajustes adicionales. Aun así, la solución logra un equilibrio adecuado entre funcionalidad, rendimiento y sostenibilidad.

2. Atributo de Análisis de Problema:

Para resolver el problema, se emplearon técnicas de programación orientada a objetos y generación procedural. El sistema se construyó mediante clases separadas para el jugador, los enemigos, el mapa y los distintos tipos de terreno, lo que permitió organizar mejor las responsabilidades y mantener un código claro y fácil de modificar. El mapa se generó con métodos aleatorios que aseguran un camino válido, mientras que el movimiento, la energía, las trampas y las reapariciones se manejan mediante varios cálculos y temporizadores.

Estas técnicas se aplicaron de forma incremental para la realización del proyecto, de manera que: primero se definió la estructura del mapa y las clases base, y luego se integraron las mecánicas dinámicas. Durante el desarrollo, las herramientas se adaptaron según las necesidades del juego, optimizando funciones que consumían recursos y ajustando la arquitectura para mantener un rendimiento estable y sostenible.

