TEC | Tecnológico de Costa Rica

Documentación del proyecto 2: Juego Escapa/Cazador

Centro Académico de Alajuela
Joselyn Tatiana Nuñez Miranda - 2025121993
Mariana González Céspedes - 2025077428

1. Atributo de Análisis de Problema:

Problema Complejo de Ingeniería: Diseñar e implementar un sistema de navegación inteligente para agentes dentro de un laberinto dinámico, donde cada agente debe tomar decisiones de movimiento en tiempo real utilizando heurísticas ligeras. El diseño debe mantener un equilibrio entre eficiencia computacional, comportamiento inteligente y una jugabilidad justa, todo dentro de buenas prácticas de sostenibilidad en el desarrollo.

Principios de Ingeniería Aplicados

- Operaciones vectoriales.
- Relaciones trigonométricas.

Para calcular direcciones óptimas de movimiento durante la persecución y el escape, permitiendo un comportamiento coherente y predecible.

- Diseño algorítmico para generar el mapa de forma procedural.
- Modelos simples de toma de decisiones para agentes autónomos.
- Ajustes de balance de juego para asegurar una experiencia justa y estable.

Análisis de Contexto y Variables

El proyecto se desarrolla dentro de un entorno dinámico: un laberinto generado que cambia en cada partida. Este contexto obliga tanto al jugador como a los cazadores a adaptarse constantemente, tomando decisiones en función de reglas específicas establecidas para cada rol dentro del juego.
Una de las variables centrales del sistema está en las reglas de movimiento. Las lianas sólo pueden ser utilizadas por los roles cazadores, mientras que los túneles están reservados exclusivamente para los roles de los que escapan. Esta diferencia genera diversidad estratégica y evita que ambos roles se comporten de la misma forma. Este diseño favorece un entorno de juego justo, con reglas transparentes y bien delimitadas.

Una de las variables centrales es la de movimientos. El código establece que solo ciertos personajes pueden atravesar determinadas estructuras. Esta diferencia es producto de

condiciones programadas dentro de los métodos de movimiento. Esto crea variedad en la partida y mantiene una experiencia equilibrada.

Otra variable clave es la lógica de escape. Cuando el personaje controlado por el código debe decidir hacia qué dirección moverse (persecución o huida), se calcula una medida basada en la distancia.

La generación del mapa también es totalmente controlada por el código: mediante funciones recursivas se construye un laberinto distinto cada vez, lo que añade variedad sin necesidad de diseñar mapas manualmente.

Finalmente, el sistema de puntuación también está definido explícitamente en el código. La pérdida de 50 puntos por escape, la ganancia de 100 por captura y la suma adicional al activar trampas son reglas programadas que buscan mantener un equilibrio entre riesgo y recompensa.

Plan de Solución:

La solución se diseñó siguiendo un enfoque modular y orientado a objetos. En primer lugar, se definieron las clases principales del sistema (Jugador, Enemigo, Mapa), cada una con atributos y responsabilidades específicas. Esta modularidad aumenta la cohesión del código y facilita la depuración y el mantenimiento.

Posteriormente, se implementaron las restricciones asimétricas del movimiento. La variable *modo_juego* se integró directamente en los métodos de desplazamiento para determinar si un agente puede o no utilizar una liana o un túnel, asegurando que cada rol respete sus limitaciones.

El paso más relevante fue el desarrollar la distancia a la amenaza y la distancia a la salida. La distancia a la amenaza se multiplica por 2.5 para aumentar su influencia. Así, el agente no solo busca escapar, sino que evalúa qué tan peligrosa es su proximidad al cazador.
Este enfoque garantiza un movimiento coherente y evita que el agente tome decisiones ilógicas.

Evaluación de Soluciones

Ventajas de la Solución Implementada:

- Eficiencia: utiliza cálculos matemáticos simples, lo que permite procesar decisiones de movimiento en tiempo real sin afectar el rendimiento del juego.
- Comportamiento predecible y ajustable: Los parámetros como el factor de ponderación (2.5) pueden modificarse fácilmente para ajustar el comportamiento de los agentes sin alterar la estructura del código, facilitando el balance del juego.

- Arquitectura modular: La separación en clases independientes (Jugador, Enemigo, Mapa, Casilla) permite extender funcionalidades sin afectar componentes existentes, reduciendo costos de mantenimiento y facilitando el trabajo colaborativo.
- Generación del mapa: Al crear laberintos únicos en cada partida mediante algoritmos recursivos, se garantiza rejugabilidad sin necesidad de diseñar y almacenar múltiples mapas manualmente, optimizando el uso de memoria.
- Diferenciación estratégica: Las restricciones de movimiento (lianas para cazadores, túneles para escapistas) crean dinámicas de juego variadas que evitan soluciones únicas y promueven la experimentación.

Limitaciones de la Solución:

- Situaciones sin salida: Los agentes pueden quedar atrapados en rincones del laberinto donde todas las direcciones están bloqueadas
- Ausencia de planificación a largo plazo: El sistema evalúa únicamente las casillas adyacentes inmediatas, sin calcular rutas óptimas completas hacia el objetivo, lo que puede resultar en caminos ineficientes o movimientos repetitivos.
- Interacciones limitadas entre enemigos: Múltiples enemigos pueden ocupar el mismo espacio o bloquearse mutuamente, afectando el realismo del comportamiento colectivo.

Consideraciones de Sostenibilidad:

La solución prioriza la eficiencia de recursos computacionales, minimizando el procesamiento innecesario y permitiendo que el juego funcione en hardware modesto. El código limpio y bien documentado reduce el tiempo de desarrollo futuro, disminuyendo el costo ambiental asociado al trabajo remoto y uso de servidores.

2. Atributo de Herramientas de Ingeniería:

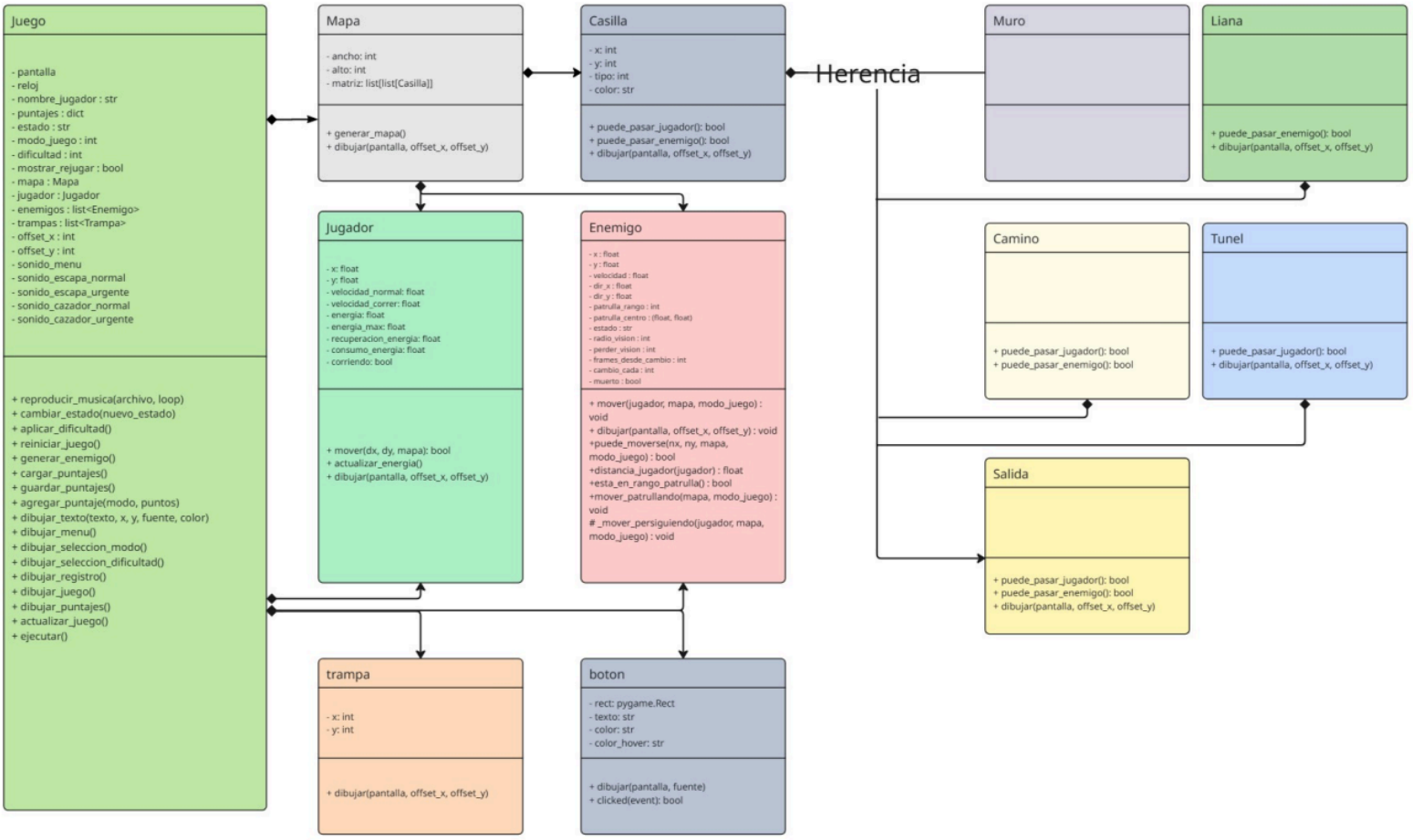
Herramientas de Ingeniería Utilizadas:

El desarrollo se realizó utilizando *Pygame* se utilizó para la visualización, manejo gráfico, control del bucle principal del juego y la gestión de eventos.

La *Programación Orientada a Objetos* permitió estructurar las entidades del sistema de forma coherente, encapsulando sus comportamientos y atributos, lo cual facilita la escalabilidad y el mantenimiento.

Se aplicó una técnica de cálculo de distancia ponderada, basada en conceptos matemáticos como la distancia euclidiana y operaciones vectoriales simples, para determinar la dirección de movimiento de los agentes dentro del juego. La generación del mapa se desarrolló mediante técnicas recursivas, lo que permite crear estructuras variadas y coherentes sin aumentar la complejidad del código ni afectar el rendimiento.

Diagrama de clases:



Aplicación y Adaptación:

La técnica de distancia ponderada se aplica en el método encargado de escoger la dirección de movimiento del enemigo. El cálculo del puntaje considera la distancia a la amenaza multiplicada por 2.5, a la cual se le resta la distancia hacia la salida. La dirección elegida es aquella que maximiza este valor, priorizando así la seguridad antes que la progresión hacia el objetivo.

Durante las primeras pruebas, la falta de un factor de ponderación adecuado provocaba comportamientos erróneos : la presa se dirigía hacia la salida incluso cuando el cazador estaba peligrosamente cerca. Ajustar el factor a 2.5 solucionó este problema sin necesidad de recurrir a algoritmos pesados.

Esquema de Puntaje:

El sistema de puntaje funciona diferente según el modo de juego, las acciones del jugador, el tiempo utilizado y la dificultad seleccionada.

Escapar:

En el modo Escapa el objetivo del jugador es llegar a la salida sin ser atrapado. Si el jugador llega a la salida, el puntaje se calcula usando una fórmula que parte de un valor base de diez mil puntos y resta diez puntos por cada segundo transcurrido. A ese resultado se le añade un bono de quinientos puntos por cada enemigo que quede vivo en el mapa. La fórmula utilizada es: puntos = max(0, 10000 – tiempo_total * 10 + cantidad_de_enemigos * 500). Además, cada vez que el jugador atrapa a un enemigo con una trampa obtiene cincuenta puntos adicionales.

Sin embargo, si el jugador es atrapado por un enemigo en el modo Escapa, su puntaje final queda automáticamente en cero. No importa cuánto haya avanzado ni cuánto tiempo haya sobrevivido; cualquier atrapada fuerza el puntaje final a cero antes de guardarlo.

Cazador:

En el modo Cazador el jugador debe atrapar enemigos. Cada enemigo atrapado suma cien puntos. Si un enemigo logra llegar a la salida, el jugador pierde cincuenta puntos y se incrementa un contador de enemigos que escaparon. El sistema no permite que el puntaje baje de cero, por lo que después de cualquier resta se asegura que el valor mínimo posible sea cero.

Cuando el tiempo de la partida llega a cero, el puntaje acumulado hasta ese momento se guarda tal cual, tanto en modo Escapa como en modo Cazador. No se otorgan puntos adicionales por finalizar por tiempo.

Antes de guardar el puntaje se aplica un multiplicador basado en la dificultad seleccionada. En dificultad fácil el puntaje se multiplica por 0.8. En dificultad normal el puntaje se deja igual con un multiplicador de 1.0. En dificultad difícil el puntaje se incrementa en un veinte por ciento mediante un multiplicador de 1.2. El puntaje final que se guarda es la versión entera del resultado después de aplicar el multiplicador.

Cada puntaje incluye el nombre del jugador con la dificultad entre paréntesis, el puntaje final y la fecha y hora de la partida. El sistema guarda únicamente los cinco mejores puntajes para cada modo de juego.

Este sistema de puntaje busca reflejar el desempeño del jugador, premiar llegar a la meta, penalizar ser atrapado en el modo Escapa y ajustar la puntuación final dependiendo de la dificultad seleccionada.