

*Instituto Tecnológico de Costa Rica – sede limon*

*Escuela de Ingeniería en Computación*

*Curso: Introducción a la Programación - Live Learning*

*Proyecto 2: Escapa del Laberinto / Cazador*

*Documentación Técnica del Proyecto*

*Estudiante:*

*Brandon Steven Álvarez Franco*

*Carnet:*

*2025103576*

*Fecha de Entrega: 28/11/2025*

## 1. Documentación Técnica del Sistema

### 1.1. Arquitectura y Módulos

El proyecto utiliza una arquitectura modular basada en el paradigma de **Programación Orientada a Objetos (POO)**. El sistema se divide en clases especializadas que gestionan el entorno (mapa), los agentes (jugador y enemigos) y la lógica del juego (modos y puntuación).

Archivo	Función Principal
main.py	Controla el flujo principal del juego, los modos ( <i>Escapa / Cazador</i> ), la interfaz (HUD) y la gestión de puntuaciones.
world.py	Contiene la clase World. Responsable de la generación procedural del mapa mediante un algoritmo <b>DFS</b> y de las comprobaciones de colisión con los tiles.
tiles.py	Define la jerarquía de herencia del terreno. La clase abstracta Casilla es heredada por Muro, Tunel, Liana, etc., para aplicar polimorfismo en las reglas de movimiento.
player.py	Contiene la clase Player. Gestiona la entrada del usuario, el movimiento físico, el sistema de vida (hp) y la barra de energía.
enemy.py	Contiene la clase Enemy. Implementa una máquina de estados para la IA (Patrulla, Persecución/Huida) utilizando <b>cálculos vectoriales</b> para la navegación.
constants.py	Almacena todas las variables de configuración, como TILE_SIZE, ENEMY_SPEED y MAX_HEALTH, para centralizar la dificultad y el balance.

### 1.2. Algoritmos Clave

- **Generación de Laberintos:** Se emplea un algoritmo de Búsqueda en Profundidad (DFS - Depth-First Search) para garantizar que el laberinto sea aleatorio pero que siempre tenga un camino válido entre el inicio y la salida.

- **Inteligencia de Agentes:** Los enemigos utilizan lógica de distancia euclidiana y normalización de vectores para determinar si deben perseguir (Modo Escapa) o huir (Modo Cazador) del jugador de manera eficiente

### 1.3. Reglas de Puntuación (Según Requisitos)

El esquema de puntuación está diseñado para fomentar el riesgo y la eficiencia del jugador en cada modo:

- **Modo Huida:**

<i>Evento</i>	<i>Fórmula de Puntuación</i>	<i>Variables Clave</i>
<i>Victoria</i>	<b>Puntuación Final</b> $= \frac{1000 \times (\text{Dificultad Base})}{\text{Tiempo de Escape (s)}}$ <b>+ Bonos</b>	<i>La dificultad base es una constante multiplicada por la cantidad de cazadores (NUM\_ENEMIES) y su velocidad.</i>
<i>Bono Trampa</i>	<i>+50 puntos por cada cazador eliminado con una trampa.</i>	<i>Premia el uso estratégico del entorno.</i>

- **Modo Cazador:**

<i>Evento</i>	<i>Puntos</i>	<i>Regla</i>
<i>Atrapar Enemigo</i>	<i>+50 puntos.</i>	<i>Es la ganancia principal del modo.</i>
<i>Enemigo escapa</i>	<i>-50 puntos.</i>	<i>Si un enemigo llega a la salida del mapa, se resta esta cantidad para crear presión.</i>

## 2. Diagrama de Clases (Modelo de Objetos)

El siguiente Diagrama de Clases UML modela la estructura de objetos utilizada, detallando los atributos (variables) y métodos (funciones) clave de cada componente del sistema.

Diagrama de clase  
Poncho Soto, Álvaro López 2 de febrero de 2020

