JUEGO DE LABERINTOS 3D

VISIÓN POR COMPUTADORA + VISUALIZACIÓN 3D EN TIEMPO REAL

INTEGRANTES

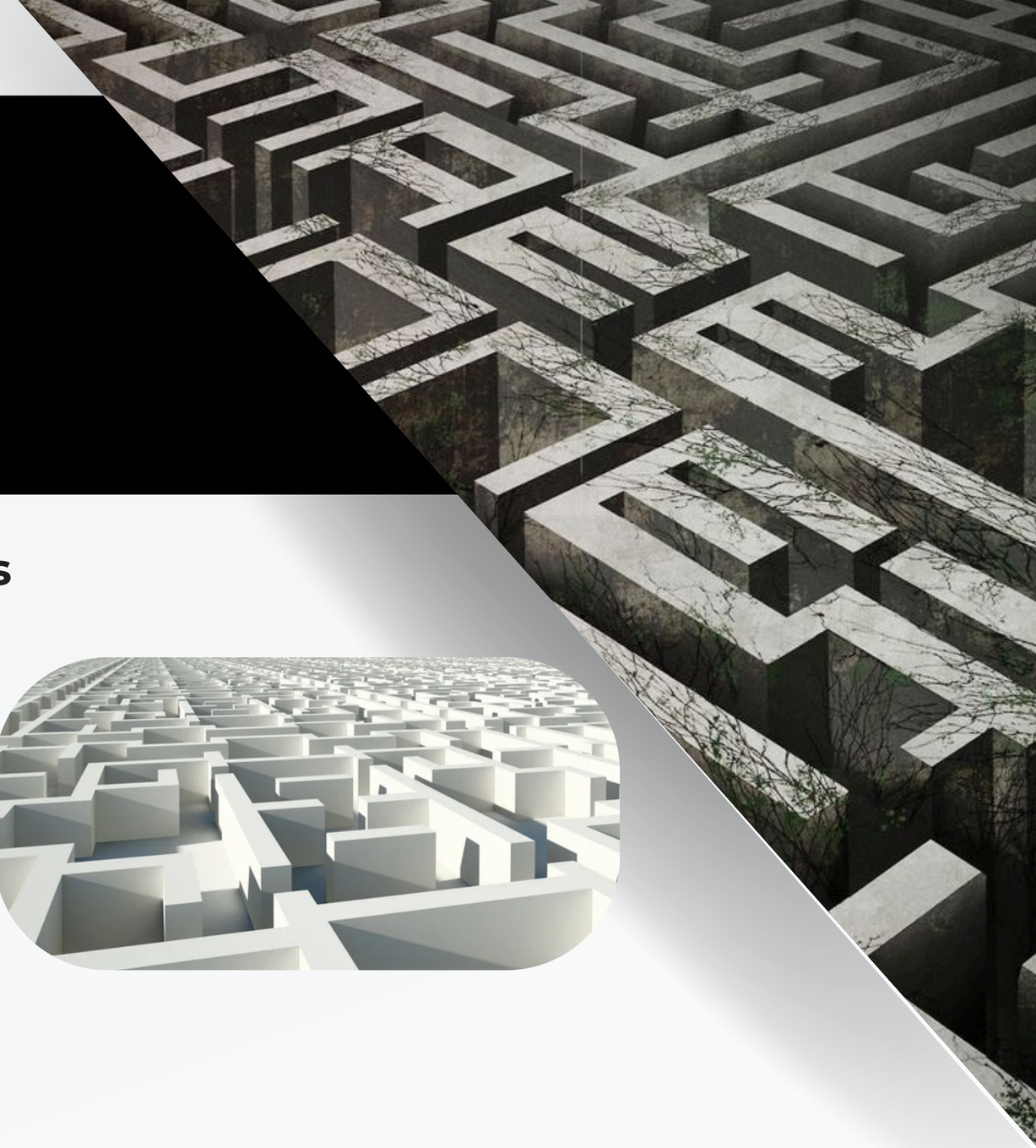
Cristian Daniel Montañez Pineda

Sergio Alejandro Ruiz Hurtado

Daniel Felipe Soraicpa Torres

Juan Jose Medina Guerrero

JUEGO DE LABERINTOS 3D



Análisis de Funcionalidades en consideracion

- Generación aleatoria del laberinto
- Recolección de llaves
- IA cazadora con visión simulada
- Meta: escapar antes de ser atrapado

INTELIGENCIA ARTIFICIAL CAZADORA

01 **Patrulla**

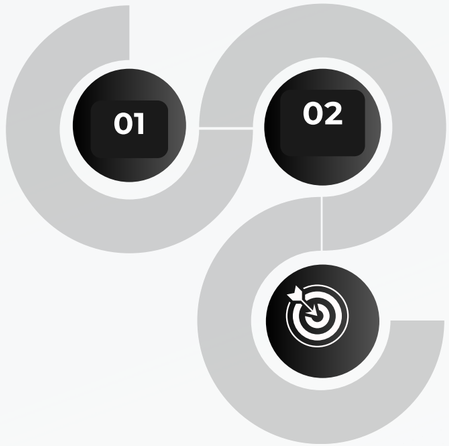
- Ruta predefinida o aleatoria entre nodos del laberinto.
- Escanea visualmente mientras se mueve.
- Usa una cámara virtual para “ver” desde su punto de vista.

02 **Persecución**

- Entra en este estado si detecta al jugador dentro de su campo de visión.
- Calcula ruta óptima hacia el jugador con A* (o similar).

03 **Búsqueda**

- Explora sistemáticamente las zonas cercanas a donde lo vio por última vez.
- Puede usar patrones de búsqueda tipo espiral o BFS local.



CÓMO VE Y ENTIENDE LA IA EL ENTORNO 3D

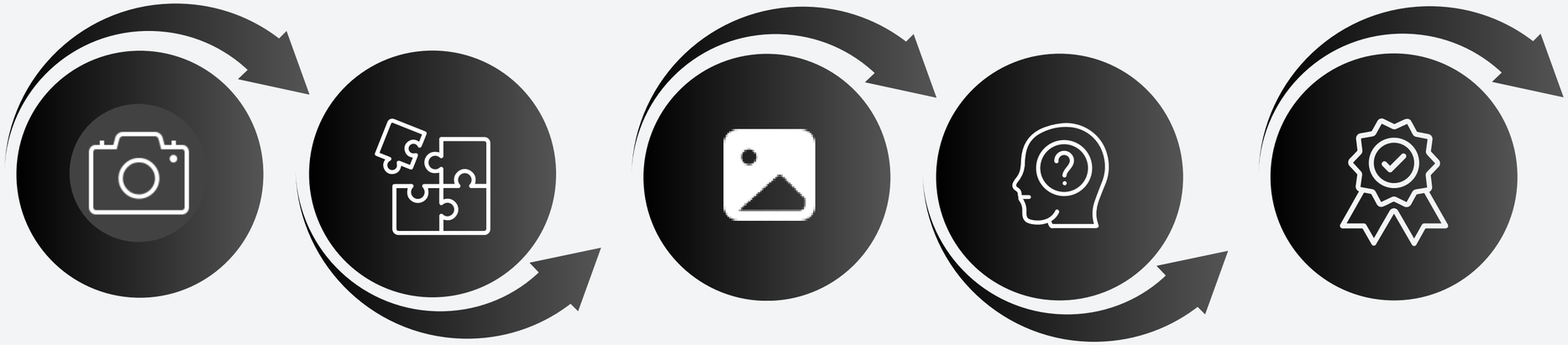
Cámara virtual (Three.js)

Renderización en buffer oculto (WebGLRender Target)

Imagen 2D resultante (frame)

Análisis con visión por computadora (OpenCV.js o TensorFlow.js)

Detección de jugador → acción de IA (seguir, buscar, etc.)



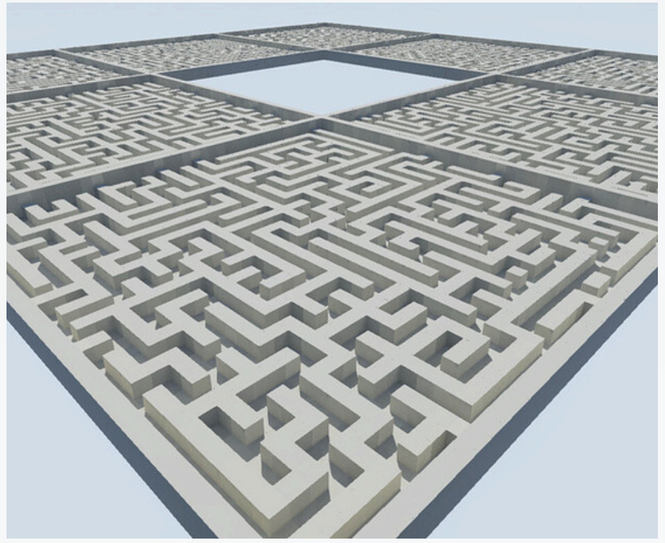
GENERACIÓN PROCEDURAL DEL LABERINTO

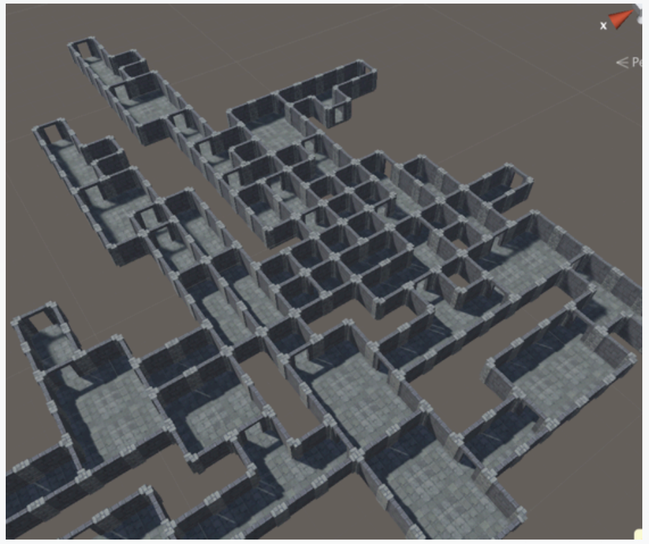
01 Algoritmos propuestos

- DFS Backtracking
- Prim
- Kruskal

02 Ventajas

- Laberintos únicos
- Dificultad ajustable
- Se garantiza una salida



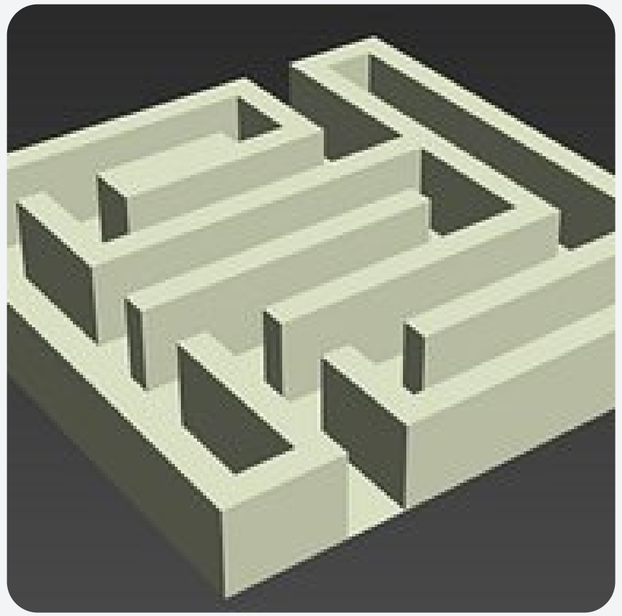


VISUALIZACIÓN 3D CON THREE.JS

01

Captura de la escena 3D

muros, luces, cámara en primera persona



02

Componentes

geometrías, texturas, luces

03

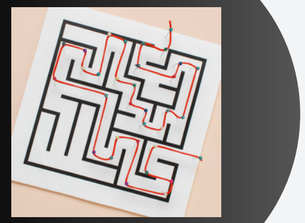
Elementos visuales:

llaves brillantes, sombras, ambientación

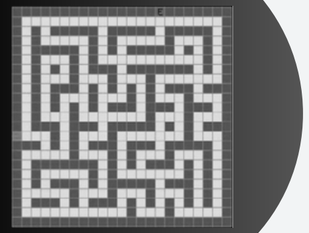
ENTRADA DESDE IMAGEN 2D

Procesamiento

Una imagen original del laberinto



Versión binarizada



Cuadrícula (matriz)

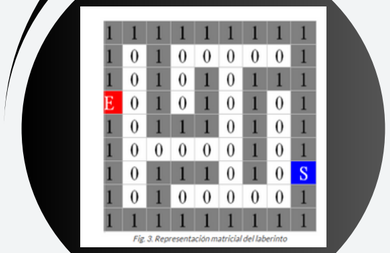


Fig. 3. Representación matricial del laberinto

escena en 3D (render en Three.js)

TECNOLOGÍAS UTILIZADAS

Visualizacion

- Three.js
- JavaScript
- WebGL

Vision por maquina

- Python + OpenCV (prototipo)
- opencv.js / TensorFlow.js
- WebGLRenderTarget

BORCELLE