# CHOMP CRAWLER

*Adaptación al Entorno Web del Pac-Man Clásico*

**Autor:** Diego Sota Rebollo

**Tutor:** Dr. Jesús María González Barahona

Grado en Ingeniería en Sistemas Audiovisuales y Multimedia

Universidad Rey Juan Carlos · Curso 2025/2026

# ESTRUCTURA DE LA PRESENTACIÓN

# INTRODUCCIÓN

01

*Contexto y Motivación*

## Pregunta

*¿Es posible crear un videojuego inspirado en Pac-Man cuyo objetivo sea resolver un laberinto real, diferente en cada partida?*

## Contexto

Navegadores como plataformas de ejecución

Pac-Man (1980): Mecánicas icónicas vs. Tablero predecible

## Objetivo Principal

- Juego completo y rejugable
- Laberintos procedurales
- Ejecución en navegador web

# STACK TECNOLÓGICO

02

*Tecnologías empleadas en el desarrollo*

**TypeScript**

Motor lógico y aplicación web

**Python**

Generador procedural de laberintos

**WebAssembly**

Ejecución de Python en el navegador (Pyodide)

**WebGL**

Renderizado 3D acelerado por hardware

**React + Next.js**

Arquitectura de componentes y despliegue web

**React Three Fiber**

Escena 3D declarativa sobre WebGL/Three.js

**Zustand**

Estado global reactivo (UI ↔ motor)

**Tailwind CSS**

Estilización de la interfaz

# ¿QUÉ HACE?

*Resumen funcional de la aplicación*

03

**Pac-Man + Rejugabilidad + Navegador**

## Sin instalación

Corre íntegramente en el navegador

## Generación Procedural

Cada partida produce una mazmorra única

## Vista 3D

Renderizado tridimensional con WebGL

## Enemigos

Comportamiento autónomo

# MECÁNICAS DE JUEGO

03

*Cómo se juega a Chomp Crawler*

### Exploración

Mazmorra interconectada.
Objetivo: Alcanzar la salida.

### Ecos (enemigos)

Entidades que persiguen al jugador.
Sistema de detección.

### Generación procedural

Mazmorras únicas.
Dificultad escalable.

### Esferas de esencia

Combustible para medallones
Puntuación
Dash

### Medallones

Objetos recolectables.
Mejoras progresivas.
Nivel 5 → Habilidad especial.

### Esferas de ruido blanco

Power Pellets
Ahuyentan enemigos temporalmente.

# ARQUITECTURA DEL SISTEMA

04

*Visión general de la implementación*

## GENERADOR DE NIVELES

*Python + WebAssembly (Pyodide)*

· Generación en cliente
· Tilemap ASCII como salida

## MOTOR LÓGICO ECS

*TypeScript*

· Ciclo de partida completo
· Estado frío (imperativo)

## APLICACIÓN WEB

*Next.js · React Three Fiber · Zustand*

· Escena 3D (React Three Fiber)
· HUD / Menus (React)

# GENERACIÓN PROCEDURAL

04

*Cómo se construye cada mazmorra*

# GENERACIÓN PROCEDURAL

04

*Cómo se construye cada mazmorra*

## Capa Micro

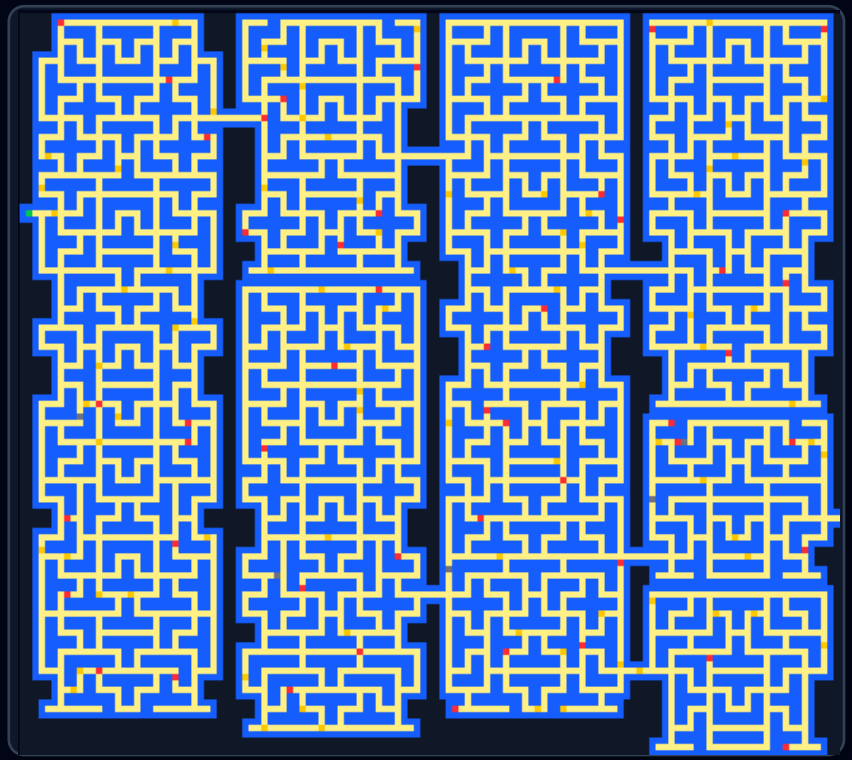
*Interior de cada sala*

- Algoritmo ShaunLeBron (Niveles tipo Pac-Man)
- Laberintos cíclicos (Sin callejones)
- Niveles simétricos, túneles asimétricos
- Python sobre WebAssembly (Pyodide)

# GENERACIÓN PROCEDURAL

04

*Cómo se construye cada mazmorra*



## Capa Macro

*Estructura de la mazmorra*

- División en columnas
- Partición en salas (Chunks)
- Algoritmo de Kruscal (Conectividad Total)
- Túneles laterales alineados

# MOTOR LÓGICO ECS

*Entity Component System*

04

## ENTIDAD

Identificador único
sin datos propios

## COMPONENTE

Datos puros
(Posición, Velocidad...)

## SISTEMA

Lógica pura
(opera sobre firmas)

# DISEÑO VISUAL Y ARTE



*Paleta de color y dirección de arte*

**Accent**

**Background**

**Main Dark**

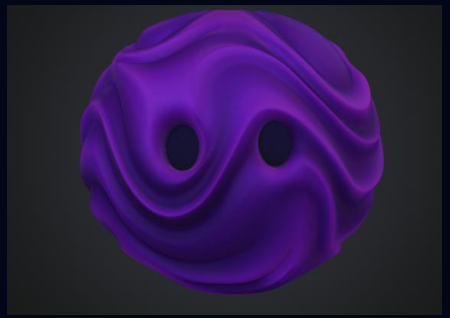
**Main**

**Text Main**

**Text Body**

**Success**

**Alert**



# EXPERIMENTOS Y VALIDACIÓN

05

| Experimento | Problema detectado | Solución aplicada | Resultado |
|---|---|---|---|
| Rendimiento de generación | Tiempos de minutos por nivel. | Banco de pruebas Python.<br>Identificar causas.<br>Relajar restricciones estéticas originales. | **Minutos → segundos.** |
| Renderizado 3D | Caídas severas de FPS en mazmorras grandes | Profiler del navegador.<br>Instanced meshes. | **FPS estables.** |
| Validación con usuarios (5) | Confusión de objetivo y dificultad plana para jugadores habituales | Mejora de señalización de salida.<br>Escalado de velocidad de ecos por nivel. | **Mejor jugabilidad.** |

# PROBLEMAS PRINCIPALES

05

*Cómo se abordaron los retos técnicos*

## Generación procedural lenta

**Causa:** Restricciones estéticas heredadas del algoritmo de Pac-Man original → 59% de descartes innecesarios

→ **De minutos a segundos por nivel**

## Caídas de FPS en 3D

**Causa:** Miles de instancias individuales → miles de draw calls por frame

→ **FPS estables en cualquier tamaño de mazmorra**

## Doble naturaleza del movimiento

**Causa:** Jugador con movimiento continuo, enemigos con movimiento discreto.

→ **Movimiento fluido sin sacrificar la lógica de colisión discreta**

## Integración Python en el navegador

**Causa:** El generador de laberintos está implementado en Python; el navegador no lo ejecuta nativamente

→ **Generación 100% cliente-side, sin servidor**

# PROGRESO TEMPORAL

06

*4 sprints · Jul 2024 – Jun 2026*

S1

## Investigación y Generación Procedural

*Jul 2024 – May 2025 (~11 meses)*

Investigación algorítmica. Selección y reimplementación del algoritmo de ShaunLeBron en Python.

S2

## Prototipado y Viabilidad

*May – Sep 2025 (~4 meses)*

Primer prototipo jugable. Generación a escala Pac-Man. Primer fantasma básico.

S3

## Arquitectura y Flujo de Juego

*Sep – Dic 2025 (~3 meses)*

Implementación del motor ECS. Ciclo de partida completo. Núcleo lógico sólido.

S4

## Identidad Propia y Pulido

*Mar – Jun 2026 (~3 meses)*

Migración a Next.js. Diseño visual, modelos 3D, nuevos sistemas. Producto final.

# TRABAJOS FUTUROS

07

*Líneas de mejora y expansión*

### Online y Persistencia

Autenticación de usuarios, guardado en nube y rankings globales.

### Multijugador Cooperativo

Modo co-op para 2–4 jugadores vía WebRTC (P2P).

### Nuevos Enemigos y Jefes

Ecos con patrones únicos, batallas de jefe final y finales alternativos.

### Eventos Especiales

Salas singulares, misiones transitorias, objetos ofensivos

### Personalización Estética

Sistema de progresión con cosméticos desbloqueables para el protagonista.

# CONSECUCIÓN DE OBJETIVOS

07

| | |
|---|---|
| Objetivo principal — ALCANZADO | Motor lógico desde cero — ALCANZADO |
| Tecnología Web 3D — ALCANZADO | Generador procedural — ALCANZADO |
| Ejecución 100% cliente-side — ALCANZADO | Stack Next.js + React — ALCANZADO |

# CONCLUSIONES

07

1

## Producto funcional y accesible

Chomp Crawler es un videojuego web completo, jugable sin instalación desde chompcrawler.com, con identidad visual y mecánicas propias.

2

## Integración tecnológica heterogénea

El mayor logro de ingeniería: Python + TypeScript + WebAssembly + WebGL integrados en un único producto cliente-side.

3

## Generación procedural de calidad

Sistema de dos capas (macro + micro) con garantía algorítmica de conectividad. Ningún nivel se repite, todos son navegables.

4

## Motor ECS desde cero

Arquitectura robusta y extensible que separa datos y comportamiento, con adaptaciones innovadoras al contexto del proyecto.

# LECCIONES APRENDIDAS

07

- React moderno y arquitectura ECS no están en el temario del Grado — ambos aprendidos de forma autónoma y aplicados en el proyecto.
- TypeScript: la inversión en tipado estático previene horas de debugging en proyectos de esta complejidad.
- Integración prudente de IA: aprobar cambios atómicos genera ilusión de control — el entendimiento real exige leer el código en contexto completo.
- Sostenibilidad del ritmo: la exigencia desmedida es contraproducente. Las fases de reflexión entre sprints son tan valiosas como las de desarrollo activo.
- Diseño visual como disciplina de ingeniería: paleta de color, contraste y tipografía tienen impacto directo en la experiencia del usuario.

# CHOMP CRAWLER

**→ PLAY NOW ←**

*Gracias. Quedo a disposición del tribunal.*

Repositorio: github.com/yodak025/chomp-crawler

**chompcrawler.com**