

Memoria descriptiva del programa Time Madness.

1. Introducción

La presente memoria descriptiva tiene como objetivo documentar de manera técnica y exhaustiva el sistema Time Madness, un videojuego de estrategia en tiempo real (RTS) desarrollado con fines académicos y de investigación. Este documento proporciona una visión integral de la arquitectura, funcionalidades, proceso de instalación y gestión estratégica del sistema.

El sistema permite a los usuarios experimentar diferentes enfoques estratégicos mediante mecánicas asimétricas que distinguen a cada civilización en términos de unidades, edificios, recursos y capacidades especiales.

Adicionalmente, TIME MADNESS constituye un entorno ideal para el entrenamiento y evaluación de sistemas de inteligencia artificial. La complejidad de sus mecánicas, que incluyen gestión de recursos, construcción estratégica, combate en tiempo real y toma de decisiones multi-objetivo.

Time Madness es un sistema interactivo complejo que integra:

- Gestión de perfiles de usuario con persistencia local
- Tres modos de juego diferenciados (Historia, Individual, LAN)
- Sistema de recursos multi-variable con balance económico
- Motor de inteligencia artificial con dificultad adaptativa
- Combate táctico en tiempo real sobre mapas 3D
- Sistema de configuración personalizable por perfil

2. Características técnicas del sistema.

Time Madness incorpora las siguientes características técnicas:

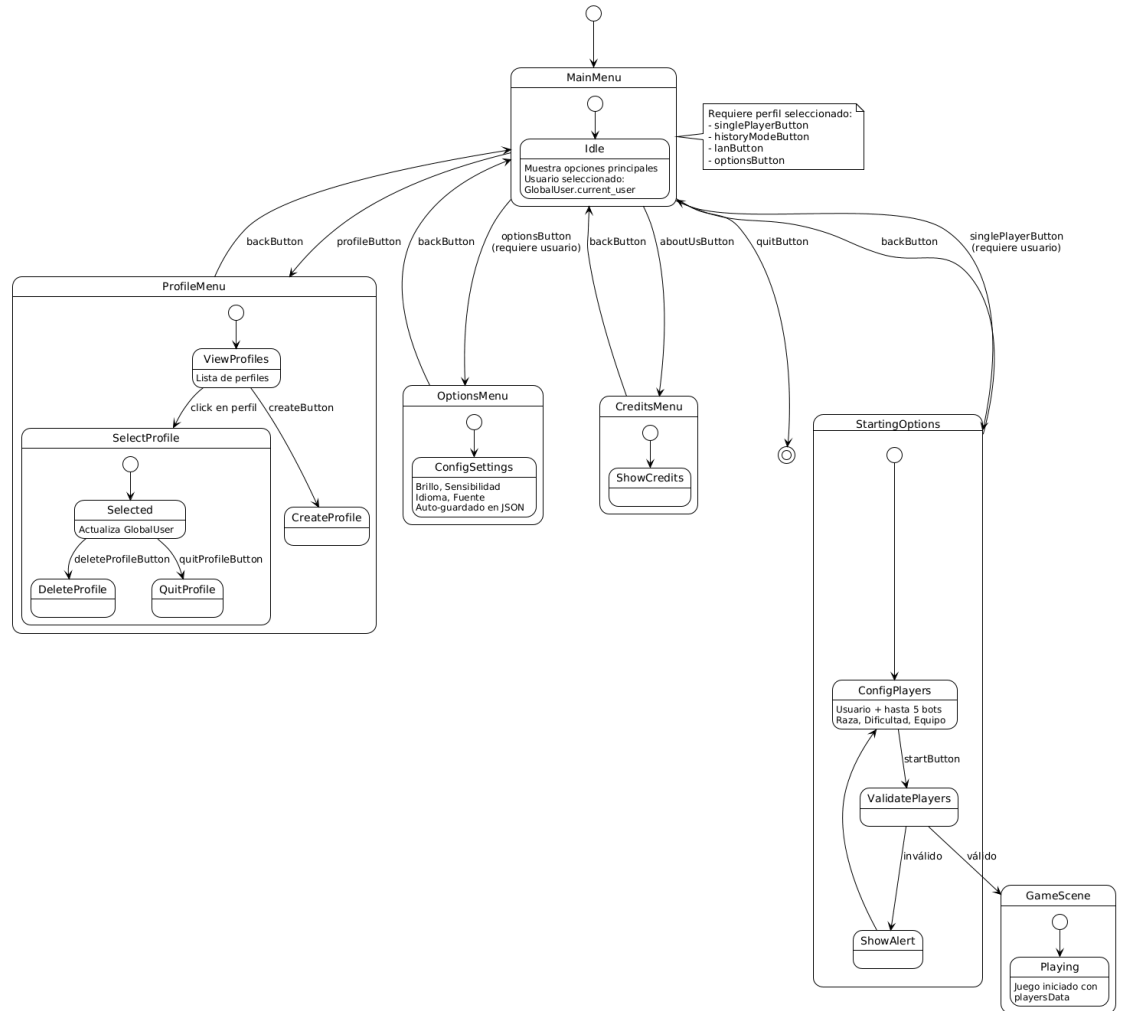
Arquitectura del Sistema

- Motor de juego: Godot Engine 4.3+
- Lenguajes de programación: GDScript (90.9%), GDShader (9.1%)
- Sistema de persistencia: Serialización en formato JSON para guardado y carga de partidas
- Máquina de estado para el manejo de las acciones de los Bots.

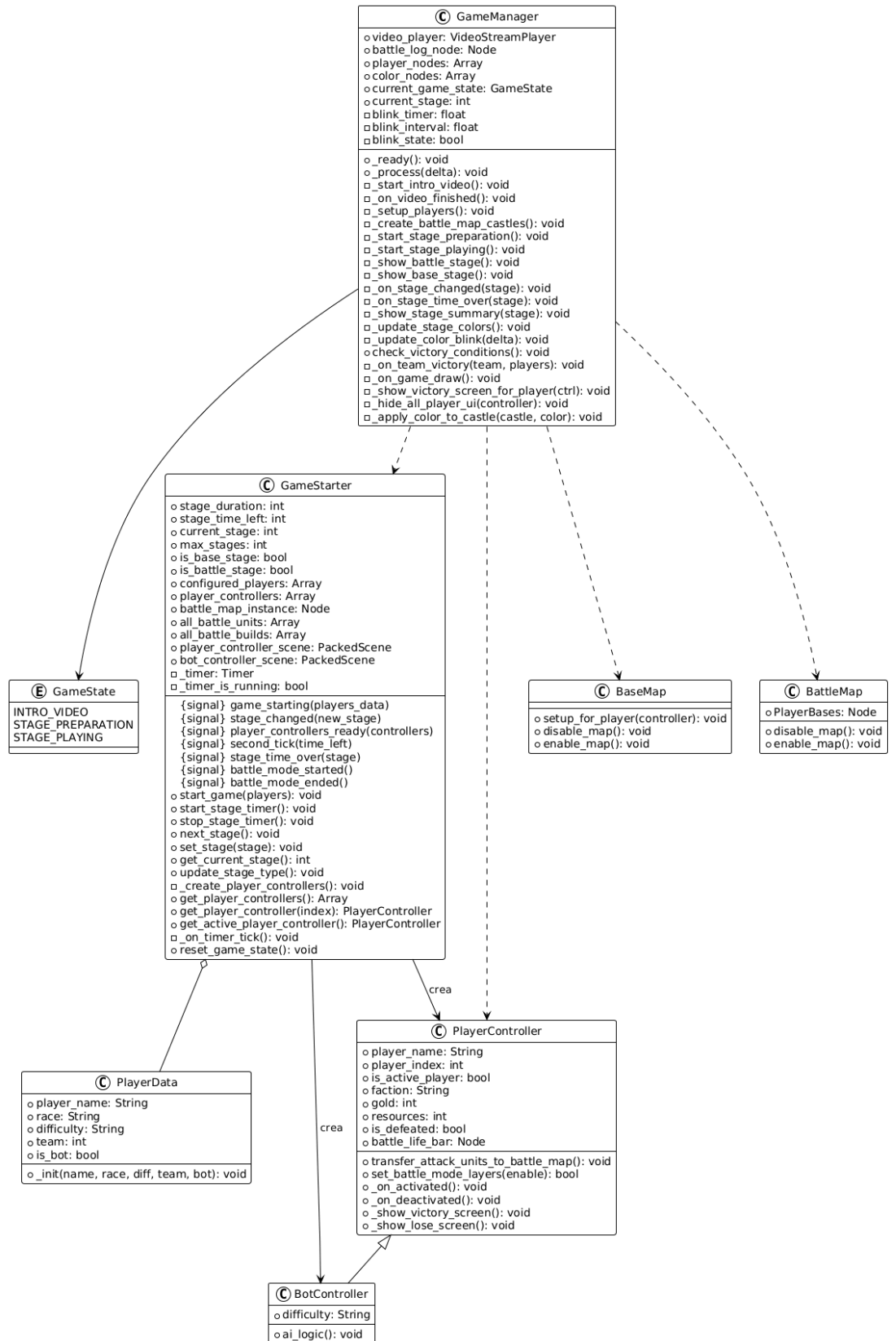
Subsistemas principales

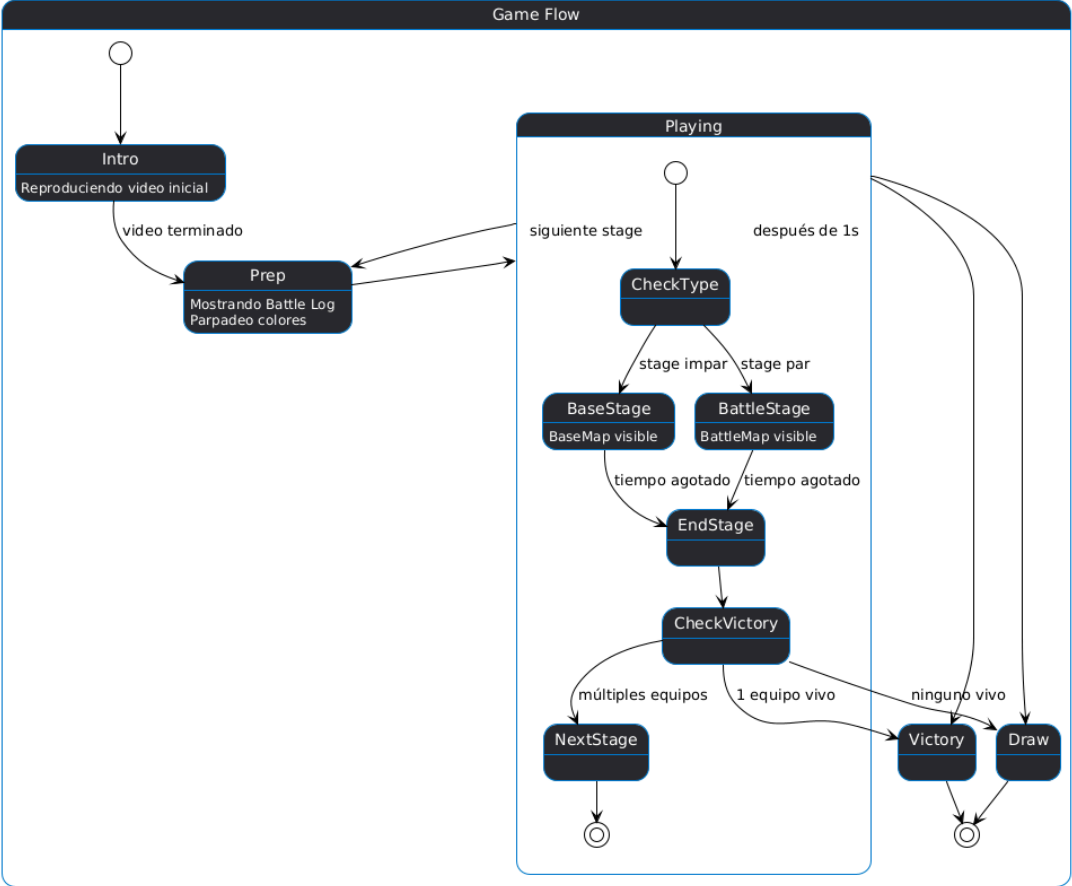
El sistema está estructurado en módulos funcionales que gestionan diferentes aspectos del juego:

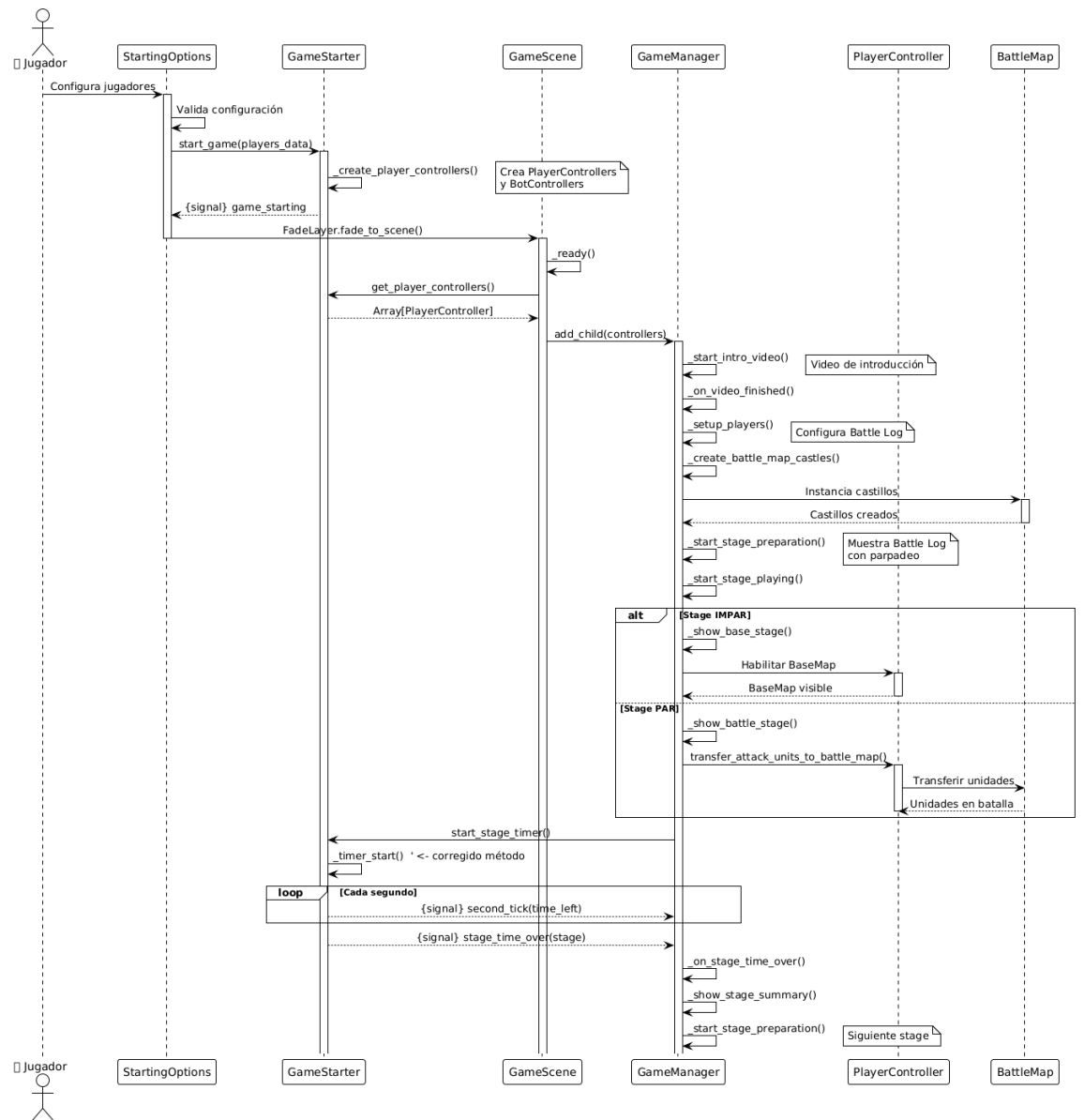
- **Menú e Interfaz Gráfica (GUI System)**



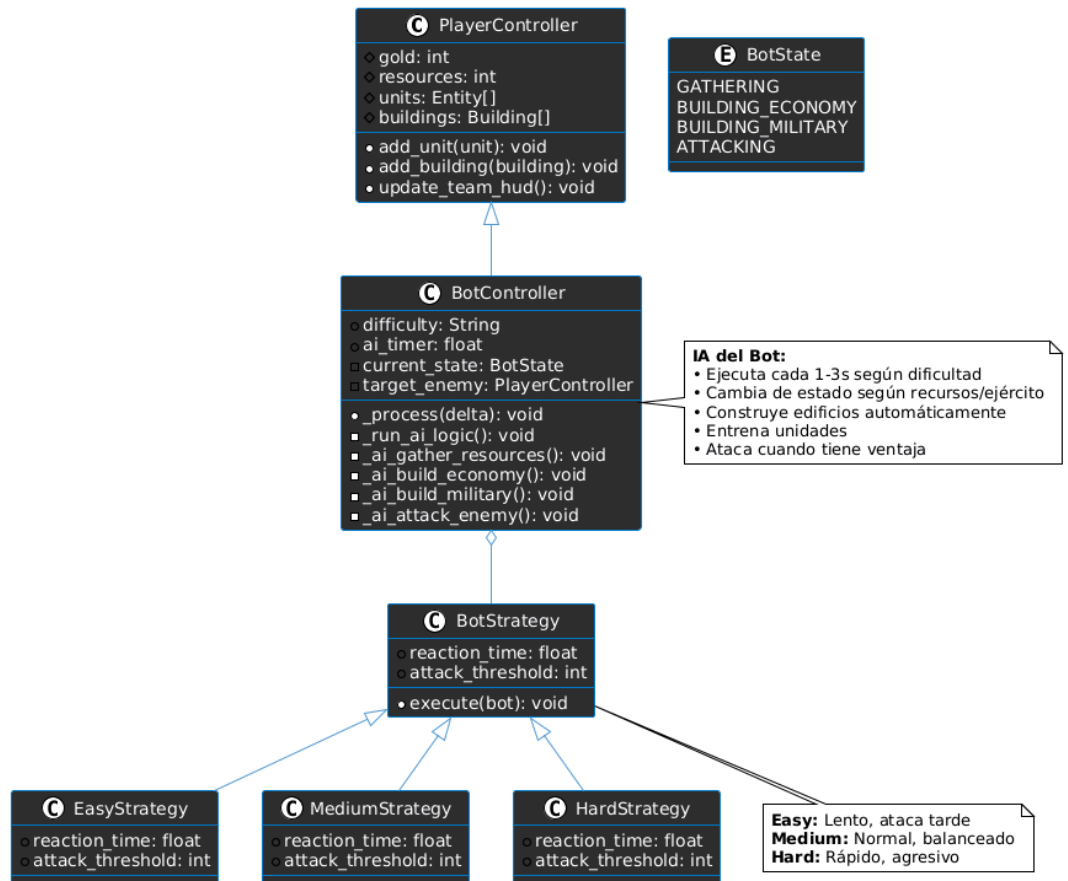
- **Player Controller**

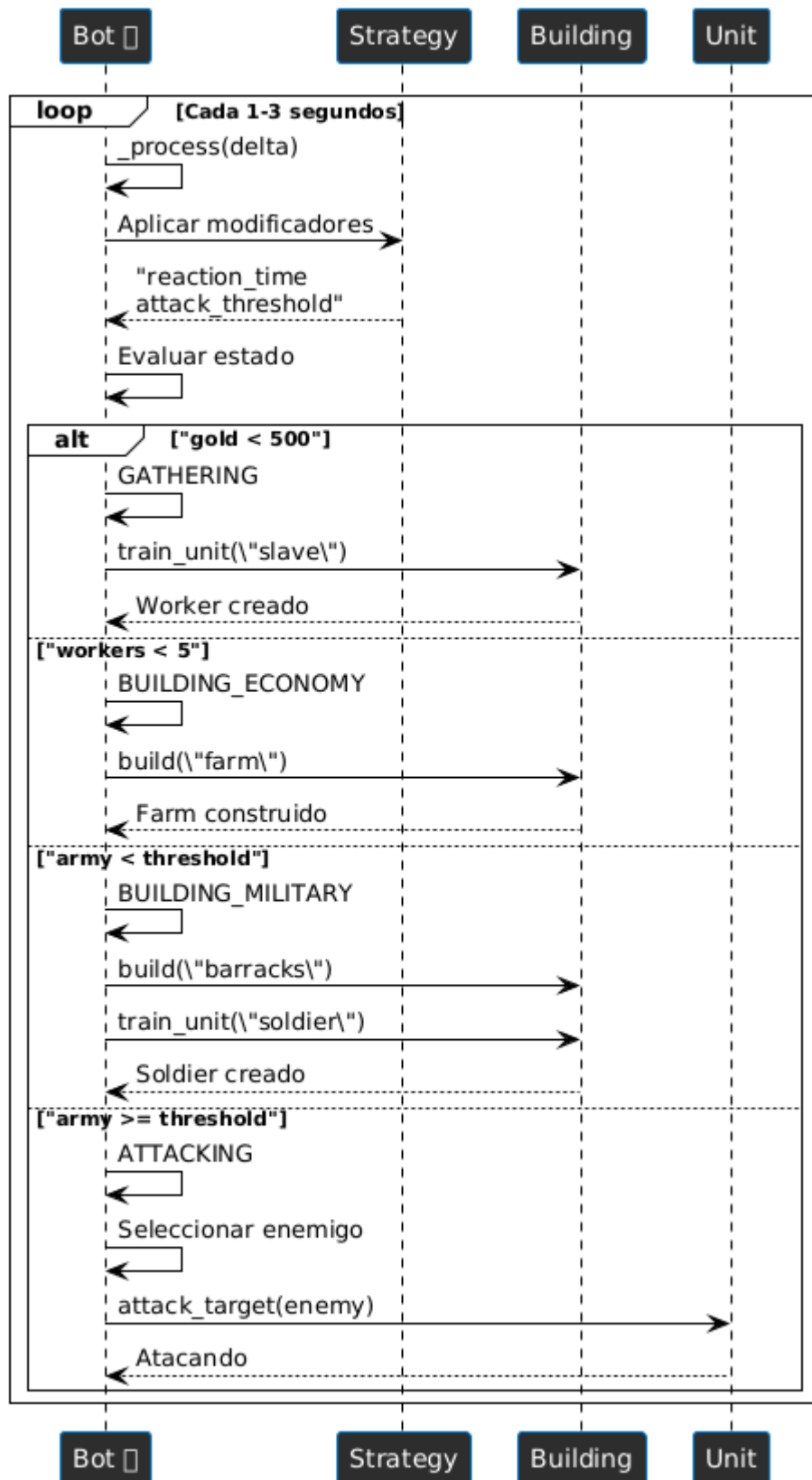




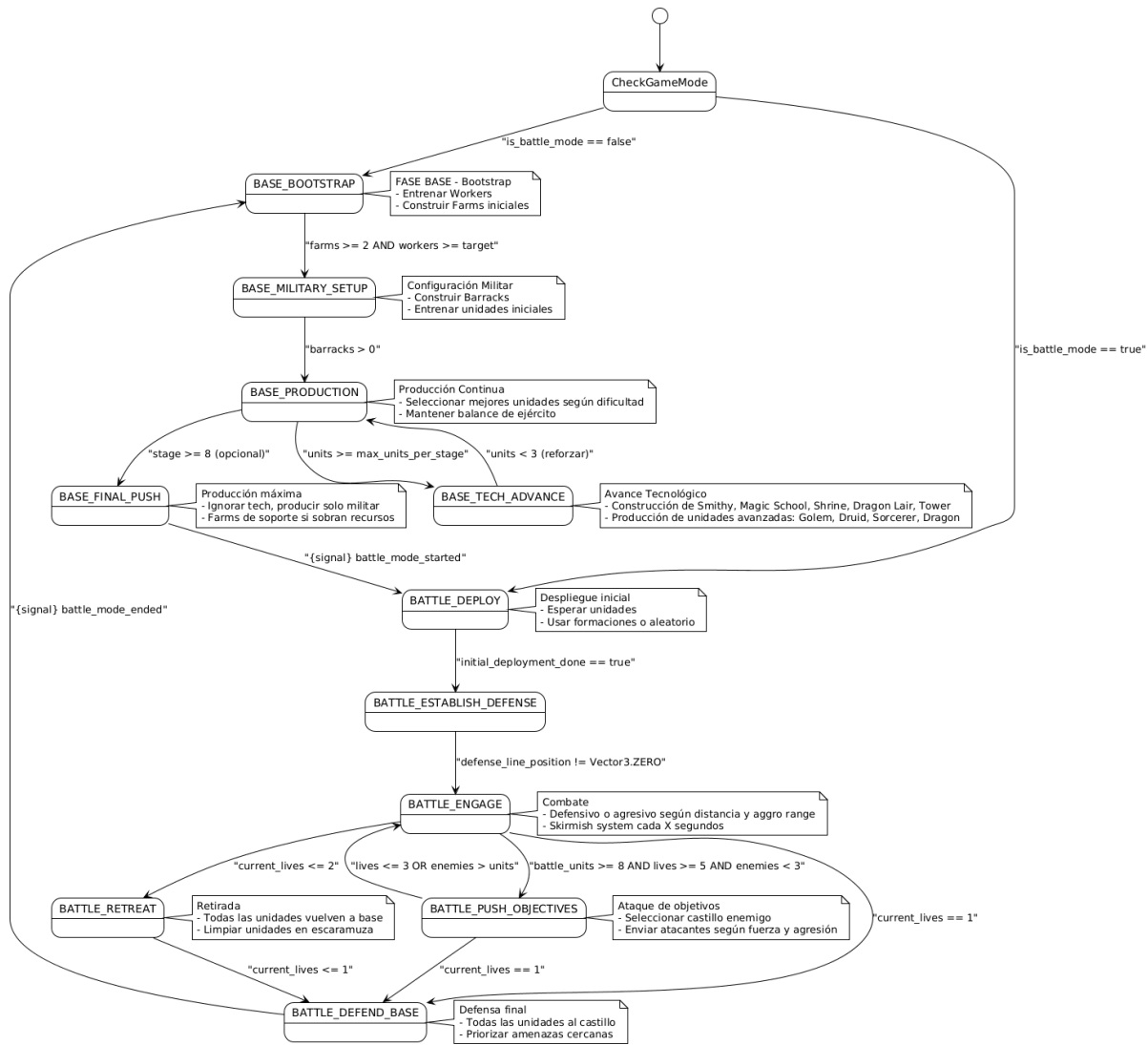


- Bot Controller (IA)



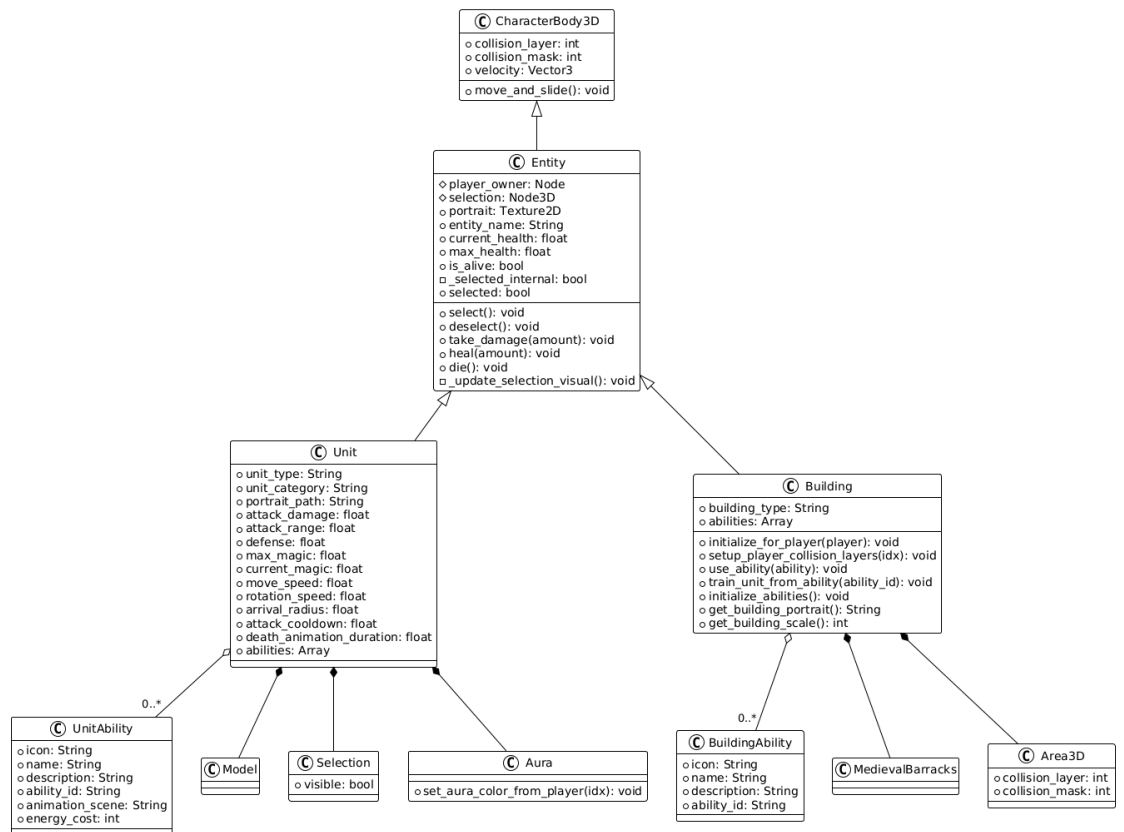


Super Bot AI - Máquina de Estados Completa



● Building & Unit System

Sistema de Entidades - Unidades y Edificios



Sistema de Recursos y Economía

El sistema económico del juego se basa en tres elementos principales:

Oro

- Recurso primario para la construcción de edificios y entrenamiento de unidades
- Generación: 1 unidad de oro por segundo por cada trabajador activo
- Sin límite de acumulación

Recursos Materiales

- Necesarios para construcciones avanzadas y unidades especializadas
- Generación: 0.5 unidades de recursos por segundo por cada trabajador activo
- Sin límite de acumulación

Trabajadores (Workers)

- Unidades productivas que generan oro y recursos pasivamente
- Se obtienen mediante la construcción de Granjas (Farms)
- No tienen límite de cantidad

Mantenimiento (Upkeep)

- Representa el costo de sostenimiento de unidades militares
- Cada unidad consume un valor específico de upkeep
- Límite inicial: 10 unidades de upkeep (ampliable mediante edificios)

Interfaz de Usuario

La interfaz del sistema está organizada en las siguientes secciones:

Panel Superior

- Visualización en tiempo real de recursos disponibles (oro, materiales)
- Contador de trabajadores activos
- Indicador de upkeep (usado/máximo)
- Temporizador de partida

Área de Visualización Principal

- Vista tridimensional del mapa de juego
- Representación de unidades, edificios y elementos del terreno
- Indicadores visuales de selección y rangos de acción

Panel de Minimapa

- Representación esquemática del mapa completo
- Indicadores de posición de unidades y edificios
- Navegación rápida por el área de juego

Panel de Acciones

- Botones de comandos principales (Mover, Atacar, Construir)
- Información detallada de la entidad seleccionada
- Acceso a habilidades especiales

Sistema de Combate y Terreno

El sistema implementa mecánicas de combate que consideran múltiples factores:

Atributos de Unidades

- Puntos de Vida (HP)
- Daño de Ataque
- Defensa
- Velocidad de Movimiento

- Rango de Ataque
- Velocidad de Ataque

Condición de Victoria

El sistema determina la victoria mediante el mecanismo de invasores:

- Una unidad se considera "invasor" cuando ocupa una casilla adyacente al territorio base del oponente
- La condición de victoria se cumple al conseguir 5 invasores simultáneamente
- El sistema valida continuamente el estado de victoria durante la fase de combate

Alcance y Limitaciones

Alcance del Sistema

- Soporte para hasta 6 jugadores en modo local
- Mapas con dimensiones variables desde 64x64m hasta extensiones mayores
- Sistema de IA con tres niveles de dificultad
- Tres civilizaciones completamente jugables con mecánicas diferenciadas

Limitaciones Conocidas

- El modo multijugador está limitado a juego local (no incluye multijugador en red)
- El sistema requiere Godot Engine 4.3 o superior para su ejecución
- La IA no implementa aprendizaje automático avanzado, sino heurísticas predefinidas

Metodología de Desarrollo

El proyecto siguió una metodología ágil adaptada a desarrollo académico:

- Sprints de 1 semana con objetivos iterativos
- Desarrollo modular de componentes independientes
- Testing continuo de mecánicas y balance
- Documentación paralela al desarrollo

3. Arquitectura y plataforma tecnológica.

Patrón Arquitectónico

El sistema TIME MADNESS está estructurado siguiendo el patrón arquitectónico por capas adaptado a las particularidades del desarrollo de videojuegos con Godot Engine. Esta arquitectura separa claramente las responsabilidades del sistema en tres capas principales:

1. Capa Global (Singletons/Autoloads)

Gestiona el estado global del juego y proporciona servicios compartidos:

- GameStarter: Control de stages, timer y modo de juego (Base/Battle)
- Datos globales: UnitStats, UnitCosts, BuildingCosts, Teams
- Servicios: FadeLayer (transiciones), GlobalUser (perfiles)

2. Capa de Juego (Game Core)

Controla el flujo del juego y coordina los sistemas:

- GameManager: Máquina de estados del juego (Intro → Preparation → Playing)
- Mapas: BaseMap (construcción) y BattleMap (combate)
- Controllers: PlayerController (humano) y BotController (IA)

3. Capa de Entidades (Game Objects)

Representa los elementos interactivos del juego:

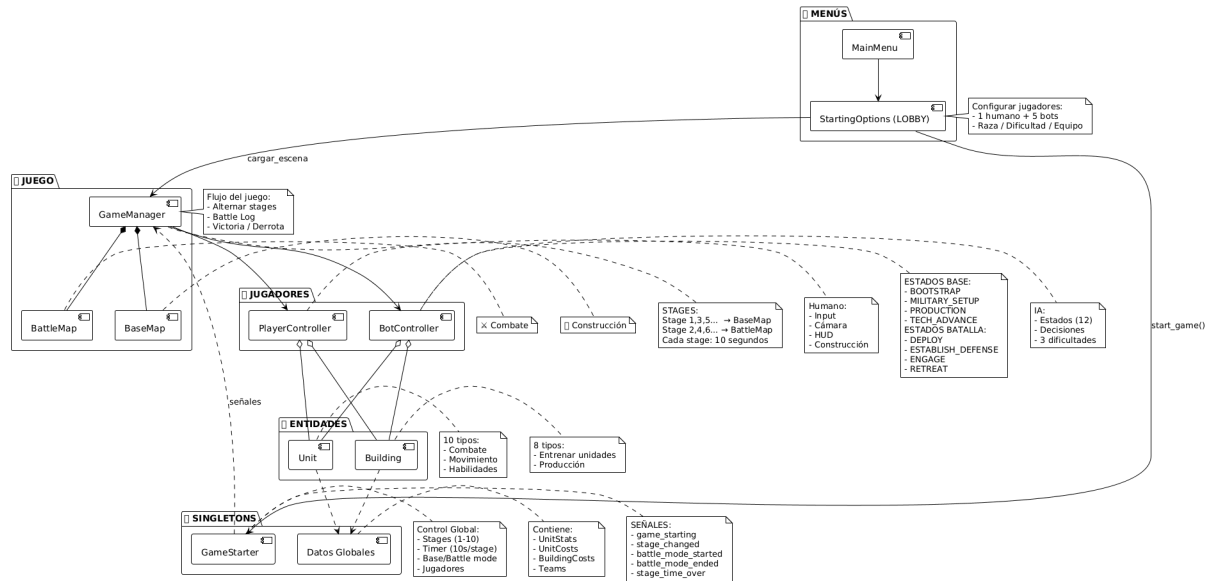
- Units: 10 tipos de unidades con sistemas de combate, movimiento y habilidades
- Buildings: 8 tipos de edificios con capacidad de producción y entrenamiento

Comunicación entre Capas

El sistema utiliza señales (signals) de Godot para la comunicación basada en eventos. La Capa Global emite señales como `stage_changed` y `battle_mode_started` que son escuchadas por la Capa de Juego (GameManager y Controllers). A su vez, la Capa de Juego controla la Capa de Entidades mediante llamadas directas a métodos como `add_unit()` o `attack_target()`.

Este diseño permite separación de responsabilidades, bajo acoplamiento entre capas, alta cohesión de componentes relacionados, y facilita la escalabilidad para agregar nuevas unidades, edificios o modos de juego.

ARQUITECTURA DEL JUEGO - Simplificada



4. Funcionalidades del sistema

a. Gestión de Perfiles de Usuario

- Creación y administración de perfiles

- Creación de perfiles con almacenamiento persistente local en formato JSON.
- Validación de nombres (máximo 20 caracteres alfanuméricos).
- Auto-incremento de nombres duplicados (Usuario, Usuario_1, Usuario_2).
- Selección rápida de perfil desde menú principal.
- Funciones de cambio de nombre y eliminación de perfil.
- Sistema de cierre de sesión sin eliminar el perfil.

- Almacenamiento de datos personalizados

- Configuraciones de usuario (brillo, sensibilidad del mouse, tamaño de fuente, idioma).
- Auto-guardado de cambios en opciones.
- Estadísticas de juego.
- Guardado de progreso en modo historia.
- Historial de civilizaciones más utilizadas.

- Persistencia

- Archivos JSON individuales por perfil en directorio user://profiles/.
- Validación de integridad de archivos al cargar.

b. Sistema de Lobby (Configuración de Partida)

- **Configuración de jugadores**
 - Soporte para 1 jugador humano y hasta 5 bots simultáneos.
 - Activación/desactivación dinámica de ranuras de jugadores.
 - Selección individual de civilización por jugador (Medieval).
 - Configuración de equipos (alianzas de 1 a 6 jugadores).
 - Asignación de colores por equipo.
- **Configuración de bots**
 - Tres niveles de dificultad: Fácil, Normal, Difícil.
 - Personalización independiente de cada bot.
 - IA con comportamiento adaptativo según dificultad.
- **Validación de configuración**
 - Verificación de campos obligatorios (raza, equipo).
 - Alertas visuales para configuraciones incompletas.
 - Confirmación antes de iniciar partida.

c. Sistema de Configuración

- **Opciones gráficas**
 - Ajuste de brillo.
 - Calidad de sombras y texturas.
 - Resolución y modo de pantalla.
- **Opciones de control**
 - Sensibilidad del mouse.
 - Velocidad de cámara RTS.
 - Configuración de zoom (mínimo/máximo).
 - Límites de desplazamiento de cámara.
- **Opciones de interfaz**
 - Selección de tamaño de fuente (Pequeña, Mediana, Grande).
 - Configuración de idioma.
 - Visibilidad de elementos de HUD.
- **Persistencia**
 - Auto-guardado instantáneo de cambios.
 - Sincronización con perfil activo.
 - Restauración de configuración por defecto.

d. Sistema de Stages (Fases de Juego)

- 10 stages totales, 150 segundos cada uno.
- Alternancia automática entre fases de construcción y combate.
- Stages impares: Fase Base.
- Stages pares: Fase Battle.
- Pantalla de preparación entre stages (Battle Log).
- Resumen de estadísticas del stage anterior.
- Timer visible en HUD.

e. Sistema de Economía por Jugador

- Economía: Oro, Recursos, Upkeep.
- Gestión de workers y costos de edificios/unidades.
- Máximo de 6 unidades militares por stage (para los bots)
- HUD de economía en tiempo real.

f. Sistema de Combate y Planificación de Ejércitos

- Cálculo de daño, sistema de críticos, ventajas de tipo.
- Composición de ejércitos (Tanks, DPS, Support, Cavalry).
- Habilidades, formaciones tácticas, comandos de unidades.
- Sistema de selección individual y múltiple.
- Comportamiento de IA y sistema de vidas.

g. Sistema de Construcción de Edificios

- Modo de colocación con vista previa 3D.
- Validación de ubicación, detección de colisiones y terreno.
- Restricciones de construcción solo en Fase Base.
- Sistema de proximidad y árboles de tecnología.

h. Sistema de Victoria y Derrota

- Condiciones de victoria y derrota.
- Pantallas de resultado (WinScene, LoseScene, DrawScene).
- Estadísticas finales de partida.

i. Sistema de Inteligencia Artificial (IA de Bots)

- Máquina de estados finitos, Decision Tree y Utility AI.
- Estados en Fase Base y Battle.
- Sistemas de decisión y modificadores por dificultad.
- Selección inteligente de unidades.

j. Sistema de Cámara RTS

- Controles de cámara, configuración ajustable y límites de desplazamiento.
- Guardado de estado de cámara por jugador.

k. Sistema de HUD y UI

- HUD del jugador activo, botones de acción.
- Alertas y notificaciones.
- Battle Log con indicadores visuales y resumen de unidades.

l. Sistema de Persistencia

- Almacenamiento de perfiles en JSON.
- Auto-guardado y sincronización con perfil activo.
- Limpieza de estado al volver al menú principal.

5. Proceso de instalación

Requisitos del Sistema

Mínimos:

SO: Windows 10 (64-bit) / Ubuntu 20.04 / macOS 11.0

CPU: Intel Core i3 / AMD Ryzen 3 (2.5 GHz)

RAM: 4 GB

GPU: Intel HD Graphics 4000 / AMD Radeon R5

DirectX: Versión 11 (Windows)

Almacenamiento: 2 GB

Resolución: 1280x720

Recomendados:

SO: Windows 11 (64-bit) / Ubuntu 22.04 / macOS 13.0

CPU: Intel Core i5 / AMD Ryzen 5 (3.0 GHz+)

RAM: 8 GB

GPU: NVIDIA GTX 1050 / AMD RX 560

DirectX: Versión 12 (Windows)

Almacenamiento: 4 GB (SSD recomendado)

Resolución: 1920x1080

- Hacer clone del siguiente repositorio:
- Abrir el project.godot en la versión de Godot indicada y ejecutar (F5).
- También se puede encontrar los ejecutables en el siguiente drive.

<https://drive.google.com/drive/folders/1w4UGwnibImuQqDvzJspkicPAVboldrew?usp=sharing>

6. Conclusiones

TIME MADNESS representa un logro significativo como proyecto académico. Demuestra capacidad de diseñar, implementar y documentar un sistema interactivo complejo funcional.

El sistema cumple sus objetivos establecidos tanto como videojuego jugable como herramienta educativa. La documentación exhaustiva facilita su estudio y extensión por terceros.

La arquitectura modular y el código organizado permiten que el proyecto sirva como base para desarrollos futuros, tanto académicos como recreativos.

El compromiso con código abierto y documentación completa refleja principios de responsabilidad social universitaria, contribuyendo al conocimiento colectivo y facilitando el aprendizaje de futuros desarrolladores.