

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%)
- Patrón arquitectónico: Model-View-Controller (MVC)
- Sistema de persistencia: Serialización en formato JSON para guardado y carga de partidas
- WebSockets para la opción multijugador.

Componentes Principales

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

- **GameManager** - Controla el flujo general del juego, la transición entre fases y el estado global del sistema
- **PlayerController** - Gestiona las acciones, recursos y estado de cada jugador
- **UnitManager** - Administra el ciclo de vida, comportamiento y acciones de las unidades
- **ResourceSystem** - Controla la generación, consumo y balance de recursos económicos
- **MapGrid** - Gestiona la representación del terreno, validación de movimientos y efectos de tipo de suelo
- **AIController** - Implementa comportamientos de inteligencia artificial con heurísticas adaptativas

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

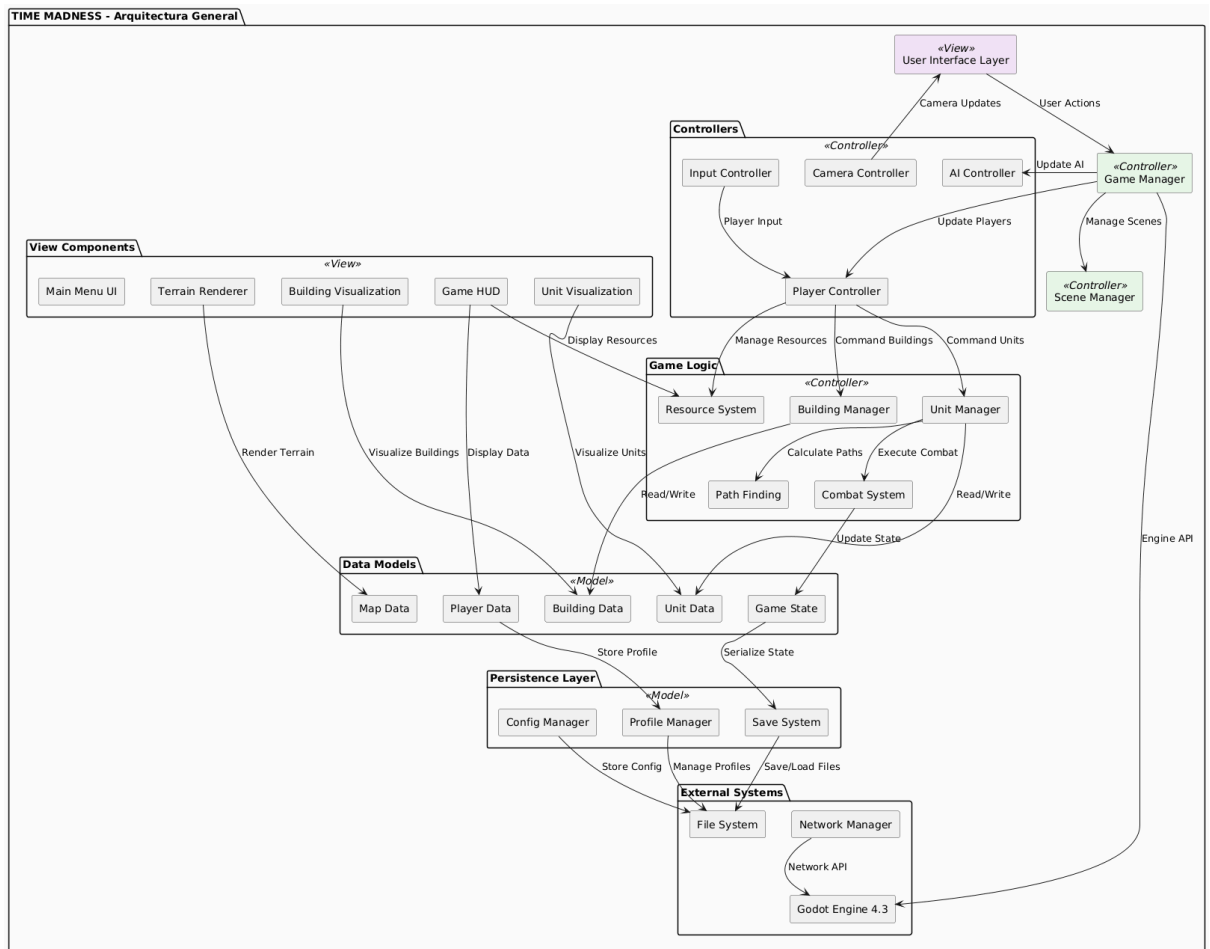
El proyecto siguió una metodología ágil adaptada a desarrollo académico:

- Sprints de 2 semanas 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 MVC

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



4. Funcionalidades del sistema

Gestión de Perfiles de Usuario

- Creación de perfiles con almacenamiento persistente local
- Validación de nombres (máximo 20 caracteres, alfanuméricos)
- Auto-incremento de nombres duplicados (Usuario, Usuario_1, Usuario_2)
- Almacenamiento de configuraciones personales
- Registro de estadísticas de juego (partidas jugadas/ganadas/perdidas, tiempo total)
- Guardado de progreso en modo historia
- Selección rápida de perfil desde menú principal
- Funciones de cambio de nombre y eliminación de perfil
- Estructura de datos en formato JSON

Modo Individual

- Partidas personalizadas contra IA
- Selección de civilización propia y enemiga
- Tres niveles de dificultad (Fácil, Normal, Difícil)
- Configuración de tamaño de mapa (Pequeño, Mediano, Grande)
- Ajuste de recursos iniciales (Escaso, Estándar, Abundante)

- IA con comportamiento adaptativo según dificultad
- Sin restricciones narrativas

Sistema de configuración.

Sistema de economía por jugador.

Sistema de combate y planificación de ejércitos.

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 indicara 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.