Documentación de *Rival Frontiers 2*

Integrantes

- Campos Ticona, José Gabriel
- Estrella Camacho, Juan Martin
- Norena Paredes, Steven Daniel
- Briceno Villegas, Leonardo Fabian

Información general

Nombre del Proyecto	*RIVAL FRONTIERS 2*
Tipo de Proyecto	Juego de Estrategia por Turnos en Consola
Lenguaje de Programación	C++
Plataforma	Windows/Linux/macOS - CLION
Fecha de Lanzamiento	30/11/25
Versión	2.3
Tiempo de Desarrollo	3 semanas

Especificaciones técnicas

Requisitos recomendados del sistema:

- SO: Windows 10, Linux, macOS
- IDE: CLion
- RAM: 256 MB
- ROM: 10 MB

Librerías utilizadas:

- #include <iostream>
- #include <vector>
- #include <string>
- #include <fstream>

Instalación del juego

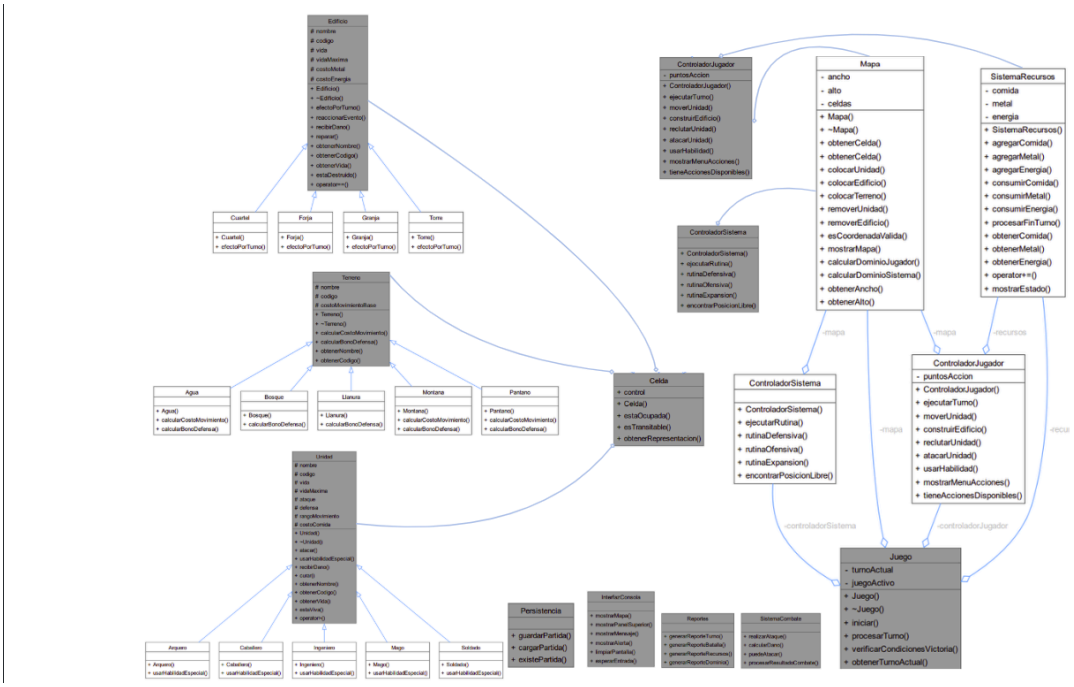
Extraer del repositorio de github todos los directorios y archivos e importarlos como un nuevo proyecto en el IDE de CLion, no aseguramos el correcto funcionamiento del juego en otros IDEs.

Arquitectura del proyecto

Estructura de Directorios:

```
RIVALFRONTIERS2/
├── src/                                    // Carpeta principal
│   ├── main.cpp                            // Punto de entrada que maneja el menú
│   ├── juego/
│   │   └── Juego.h/cpp                     // Maneja turnos y el flujo de la partida
│   ├── modelos                             // Carpeta de todas las clases en el terreno
│   │   ├── Terreno.h/cpp                   // Costos de movimiento y bonos de defensa
│   │   ├── Unidad.h/cpp                    // Vida, defensa, y habilidad de las unidades
│   │   ├── Edificio.h/cpp                  // Vida, y efecto por turno
│   │   └── Mapa.h/cpp                      // Manejo de celdas y elementos en el territorio
│   ├── controladores/
│   │   ├── ControladorJugador.h/cpp        // Acciones del jugador
│   │   └── ControladorSistema.h/cpp        // Acciones de la CPU
│   ├── sistemas/
│   │   ├── SistemaRecursos.h/cpp           // Gestión de recursos para el jugador
│   │   └── SistemaCombate.h/cpp            // Sistema de combate
│   ├── utilidades
│   │   ├── Utilidades.h/cpp                // Funciones de utilidad
│   │   └── Persistencia.h/cpp              // Guardar/cargar partidas (incompleto)
│   └── interfaz/
│       ├── InterfazConsola.h/cpp           // Manejo de impresión en consola de reportes
│       └── Reportes.h/cpp                  // Resumen de la partida por turno
│
----------------------------------------------------
```

Diagrama UML



Link del gráfico en mayor resolución:
https://drive.google.com/file/d/127xIZUF7tNM2qi1obqgJUFY9cZq4N4X3/view?usp=sharing

Características Implementadas

POO:

- Herencia: Jerarquías de Terreno, Unidad, Edificio
- Polimorfismo: Comportamiento dinámico según tipo
- Encapsulamiento: Atributos privados con getters/setters
- Sobrecarga de Operadores: `<<`, `>`, `+=`, `==`
- Relación entre clases

Mecánicas del Juego:

- Mapa de 6x6 celdas con 5 tipos de terreno
- 12 turnos totales con 2 puntos de acción por turno para jugador y CPU
- Recursos: Comida, Metal, Energía
- Sistema de control territorial
- Sistema básico de ataque/defensa

Interfaz de Usuario:

- Menú principal interactivo
- Visualización de mapa en tiempo real
- Interacción con tropas y edificios aliados y enemigos
- Opción de refrescar

Reglas del juego

Objetivo:
Controlar al menos 60% del territorio en 12 turnos o menos.

Mecánicas Clave:
- Cada turno: 2 puntos de acción
- Movimiento: 1 punto por unidad
- Construcción: 1 punto por edificio
- Reclutamiento: Requiere Cuartel + recursos

Condiciones de Victoria:
- Victoria: ≥60% dominio territorial
- Derrota: Sistema alcanza 60% dominio
- Derrota: Sin comida ni metal
- Derrota: Turno 12 sin ≥60% dominio territorial.

Flujo del programa

1. Inicio: Menú principal con 4 opciones
2. Juego: Inicialización de componentes
3. Bucle de Turnos: 12 iteraciones máximo
4. Fase Jugador: 2 acciones por turno
5. Fase Sistema: Rutinas automáticas
6. Evaluación: Verificación condiciones
7. Fin: Mensaje de resultado

Oportunidades de extensión

- Inteligencia Artificial Mejorada mediante algoritmos de búsqueda
- Finalizar el sistema de guardado
- Interfaz gráfica mejorada y sin usar consola que permiten mejores efectos visuales
- Creación de escenarios personalizados
- Sistema de Logros: Desbloqueables por jugador
- Implementación de efectos de sonido básicos

Licencia

- Tipo: Educativa/Académica
- Uso: Proyecto de curso universitario
- Restricciones: No comercial, atribución requerida
- Año: 2025