Informe Taller de Estructura de Datos

“Videojuego Onitama”

Nombres: Benjamin Bustamante
Joaquin Carrasco
Profesor : Jose Luis Veas.
Fecha de entrega : 20/06/2025

indice

Objetivo

El propósito principal de este proyecto es diseñar e implementar un juego de tablero inspirado en Onitama que ponga en práctica los conocimientos adquiridos en la asignatura de Estructuras de Datos y Algoritmos, integrando además una IA basada en Minimax con poda alfa-beta y una interfaz gráfica interactiva

En que consiste

Onitama es un juego de tablero para dos jugadores que se desarrolla sobre una cuadrícula de 5×5, donde cada bando dispone de un Maestro colocado en la casilla central de su fila inicial y dos Discípulos a su lado. Al comenzar, cada jugador recibe dos cartas de movimiento y deja una boca arriba en el centro; en cada turno, escoge una de sus cartas, desplaza una única de sus piezas según el patrón marcado

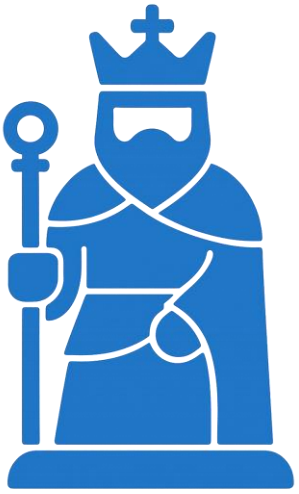
La condición de victoria puede alcanzarse de dos maneras: capturando al Maestro rival moviendo una pieza a su casilla, o trasladando tu propio Maestro hasta la casilla inicial del Maestro contrario (conocida como “Templo”), lo que otorga la victoria instantánea

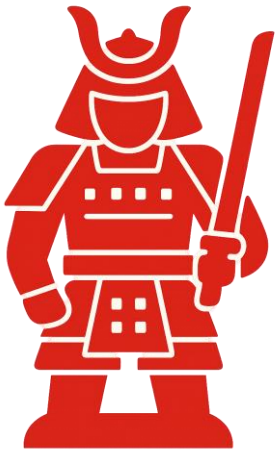
Diseño

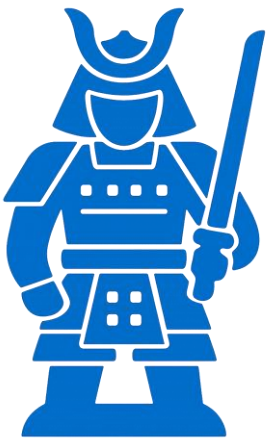
Cartas

Tablero

Fichas







Librerías ocupadas

En nuestro proyecto ocupamos la biblioteca SFML ,es una biblioteca de código abierto diseñada para facilitar el desarrollo de aplicaciones multimedia y juegos en C++. Proporciona módulos para gestionar gráficos 2D, sonido, eventos de ventana y entrada de usuario. En este proyecto, SFML la utilizamos para:

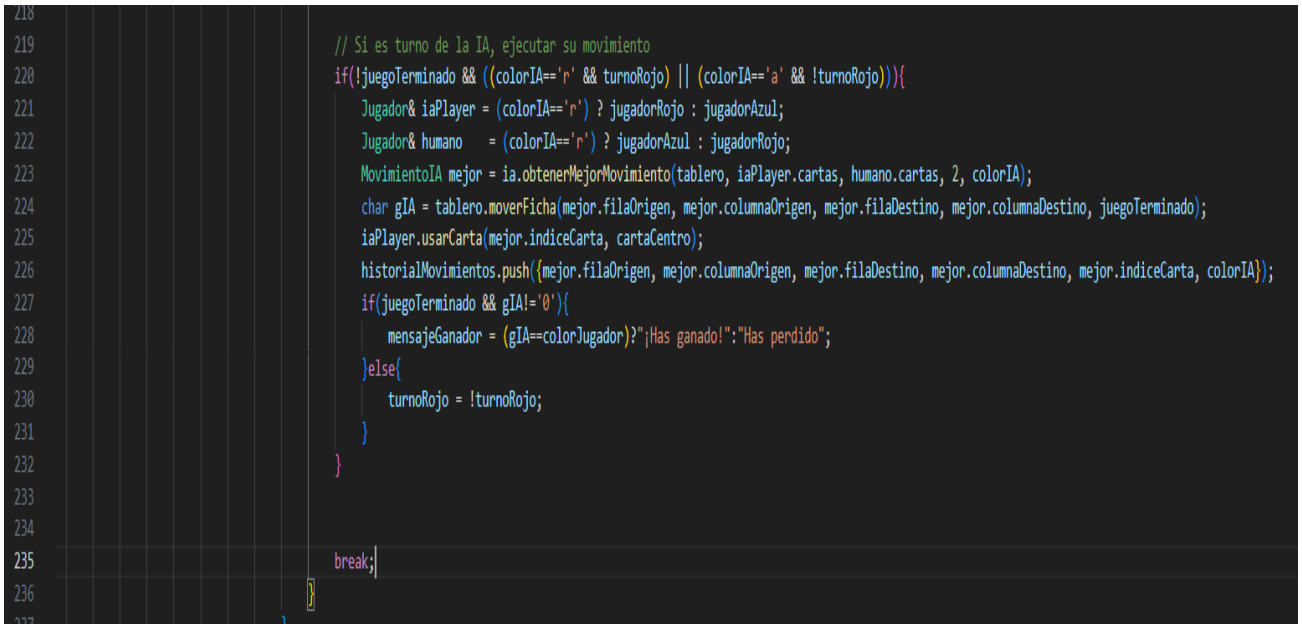
- Crear y gestionar ventanas (sf::RenderWindow).
- Dibujar elementos gráficos como textos y formas (sf::Text, sf::RectangleShape).
- Cargar y reproducir sonidos o música (sf::Music).
- Detectar eventos de teclado, mouse y cierre de la ventana (sf::Event).

Implementacion de estructura de datos en el juego:

- **punteros para guardar las fichas en la matriz**
- **pilas en cartas**

Nosotros utilizamos pilas en Historial de Movimientos , la pila historial de movimientos se va guardando un registro ordenado de todas las jugadas, colocando la más reciente en la parte superior.

```
26  struct RegistroMovimiento {
27      int filaOrigen;
28      int columnaOrigen;
29      int filaDestino;
30      int columnaDestino;
31      int indiceCarta;
32      char jugador;
33  };
```



```
// Si es turno de la IA, ejecutar su movimiento
if(!juegoTerminado && ((colorIA=='r' && turnoRojo) || (colorIA=='a' && !turnoRojo))){
    Jugador& iaPlayer = (colorIA=='r') ? jugadorRojo : jugadorAzul;
    Jugador& humano   = (colorIA=='r') ? jugadorAzul : jugadorRojo;
    MovimientoIA mejor = ia.obtenerMejorMovimiento(tablero, iaPlayer.cartas, humano.cartas, 2, colorIA);
    char gIA = tablero.moverFicha(mejor.filaOrigen, mejor.columnaOrigen, mejor.filaDestino, mejor.columnaDestino, juegoTerminado);
    iaPlayer.usarCarta(mejor.indiceCarta, cartaCentro);
    historialMovimientos.push({mejor.filaOrigen, mejor.columnaOrigen, mejor.filaDestino, mejor.columnaDestino, mejor.indiceCarta, colorIA});
    if(juegoTerminado && gIA!='0'){
        mensajeGanador = (gIA==colorJugador)?"¡Has ganado!":"Has perdido";
    }else{
        turnoRojo = !turnoRojo;
    }
}


break;
}
```

implementación de la ia minimax:

Inteligencia Artificial – IA.cpp

Esta IA controla al jugador rojo utilizando el algoritmo Minimax con poda alfa-beta. Evalúa todos los movimientos posibles en el tablero según las cartas del jugador.

evaluarPoscicion considera estrategias ofensivas como:

- Capturar al rey enemigo
- Ocupación del dojo rival
- Eliminación de piezas enemigas

Y estrategias defensivas como:

- Proteger al propio rey
- Alejarse de amenazas
- Defender el dojo propio

evaluarEstado se encarga de saber si el juego ha terminado:

si con un movimiento termina le asigna un valor elevado para que el metodo obtenerMejorMovimiento lo escoja.

El algoritmo minimax simula turnos alternos:

profundiza en turno posibles entre el jugador rojo y azul, buscando siempre maximizar el puntaje del jugador rojo y minimizar el puntaje del jugador azul

El método obtenerMejorMovimiento determina el movimiento que la ia tomara:

recorre todas las jugadas posibles, evalúa el estado del tablero con cada una y selecciona la más favorable.