Motor unificado para Tetris y Snake usando BrikScript

Autores:

Sebastián Valdivieso Zapata

Ángel Sebastián Cuaran Cruz

Brigid Vanesa Toro Males

Presentado a:

Profesor Fernan Alonso Villa Garzón

Asignatura:

Teoría de Lenguajes de Programación

Universidad Nacional de Colombia

Facultad de Minas

Medellín, Colombia

2025

1. Introducción

Este documento presenta la arquitectura, diseño, implementación y funcionamiento del motor de videojuegos 2D desarrollado para la asignatura TLP. El sistema integra dos juegos distintos —Tetris y Snake— utilizando un motor común implementado en C++ con SDL2, y un lenguaje de scripts propio llamado BrikScript, el cual permite definir entidades y comportamientos de manera declarativa.

El motor provee la infraestructura necesaria para crear mundos, dibujar entidades, procesar eventos del teclado, administrar colisiones básicas, actualizar el estado del juego y ejecutar bucles dependientes de scripts externos. Este informe está diseñado con un estilo académico orientado a documentar completamente el proyecto para su evaluación formal.

2. Objetivos del Proyecto

2.1. Objetivo general

Diseñar e implementar un motor sencillo que permita ejecutar múltiples juegos escritos en un lenguaje scripting propio, utilizando C++ como backend principal.

2.2. Objetivos específicos

- Implementar un intérprete básico para BrikScript.
- Diseñar un motor común para múltiples juegos.
- Implementar Tetris y Snake completamente mediante scripts.
- Garantizar portabilidad del ejecutable final.
- Manejar gráficos básicos mediante SDL2.
- Utilizar una arquitectura modular, limpia y mantenible.

3. Arquitectura General del Sistema

El sistema se compone de tres módulos principales. A continuación se describe la estructura de carpetas base:

```
src/
            engine/api.cpp
            engine/api.h
            interpreter/script_interpreter.cpp
            interpreter/script_interpreter.h
            integration_main.cpp
            games/
                 snake.json
                 tetris.json
```

Listing 1: Estructura de archivos

3.1. Módulo Engine (C++ + SDL2)

Es responsable de la inicialización de SDL2, manejo de ventanas y renderizado, gestión de entidades, lógica de movimiento, bucle principal del juego, entrada por teclado, gestión del puntaje y la API expuesta al intérprete.

3.2. Intérprete BrikScript

El intérprete ejecuta archivos .json que contienen scripts con instrucciones como `spawnBlock`, `moveEntity`, `wait` y loops. El intérprete corre linealmente el script, ejecutando instrucciones en orden mientras el motor actualiza visualmente el estado.

3.3. Scripts JSON de Juegos

Cada juego (Tetris y Snake) se escribe en un archivo JSON. De esta forma, el motor nunca se modifica para añadir nuevos juegos.

```
{
  "name": "Snake",
  "scripts": [
    {
      "loop": true,
      "instructions": [
        { "call": "moveEntity", "args": ["snake", 0, 1] },
        { "wait": 100 }
      ]
    }
  ]
}
```

Listing 2: Ejemplo de script JSON

4. Motor: Diseño Técnico

Esta sección profundiza en el funcionamiento interno del motor (`engine/api.cpp`).

4.1. Representación de Entidades

Cada entidad posee un ID, posición en la grilla (gx, gy), tamaño (w, h) y un tipo (string).

```
struct Entity {
    int id;
    int gx, gy;      // posici n en la grilla
    int w, h;        // tama o en tiles
    std::string type;
```

```
};
```

Listing 3: Struct de Entidad

Los tipos admitidos incluyen las piezas de Tetris (Ï”, ”L”, ”T”, etc.), componentes de Snake (”Snake”, ”Food”) y bloques genéricos (”Fixed”).

4.2. Representación del Mundo

El motor mantiene variables globales para la lista de entidades (`gEntities`), segmentos de la serpiente, IDs activos de Tetris y Snake, puntaje y estado del juego.

4.3. Ciclo de ejecución

El bucle del motor procesa eventos SDL, ejecuta instrucciones del script, aplica lógica interna (movimiento específico de juegos) y renderiza el frame repetidamente.

5. Implementación del Juego Tetris

El Tetris está implementado sobre el motor, pero su lógica interna se procesa dentro de `moveEntity()` para movimientos verticales, detección de fondo y congelamiento.

5.1. Generación de piezas

Cada vez que no hay una pieza activa, el script genera una nueva:

```
{ "call": "spawnBlock", "args": ["T", 5, 0] }
```

5.2. Lógica de caída y borrado

La función `moveEntity` hace caer la pieza. Si llega al fondo, cambia su tipo a ”Fixedτ permite generar una nueva. Cuando una fila se llena (coincide con el ancho del tablero), se marca para eliminar y las entidades superiores descienden.

6. Implementación del Juego Snake

Snake usa un modelo completamente distinto dentro del mismo motor.

6.1. Movimiento y Frutas

El motor mantiene una dirección global (`gSnakeDirX`, `gSnakeDirY`). La cabeza se mueve en esa dirección y el cuerpo la sigue. El motor garantiza que siempre exista una fruta (`ensureFoodExists`); al comerla, crece el puntaje y se genera un nuevo segmento.

6.2. Colisiones

El juego termina si la serpiente choca contra su propio cuerpo. No se aplica colisión contra bordes (wrap-around).

7. Lenguaje de Scripts: BrikScript

Los scripts se cargan desde archivos JSON y soportan instrucciones como:

- **Call:** Ejecuta una función de la API (ej. `moveEntity`).
- **Wait:** Pausa el script por un tiempo determinado.
- **Loop:** Ejecuta un bloque de instrucciones repetidamente.

8. Uso del Motor

El ejecutable final permite elegir entre Tetris y Snake mediante consola.

- **Tetris:** Flechas para mover (caída automática por script).
- **Snake:** W A S D para dirección (movimiento automático).

9. Limitaciones y Trabajo Futuro

Actualmente, no hay soporte completo para rotación en Tetris, las colisiones son básicas y SDL2 está vinculado estáticamente. Como trabajo futuro, se plantea implementar rotación completa, sonido, separación total de lógica y motor, y compilación para plataformas de bajos recursos (ej. Windows 98).

10. Conclusiones

El proyecto demuestra la implementación de un motor 2D, un lenguaje scripting propio e integración de dos juegos distintos bajo una arquitectura modular. El motor cumple con los requisitos fundamentales de la entrega, incluyendo la ejecución desde scripts y renderizado básico.

11. Anexos

11.1. Estructura de carpetas final

```
Entrega3/
            bin/motor.exe
            src/
            games/
                    tetris.json
                    snake.json
            documentacion/
                    informe.pdf
                    README.txt
            Integrantes.txt
```