Space Invaders EPN

Manual de Estándares del Proyecto

Este proyecto automatiza la compilación de un juego arcade desarrollado en C++ utilizando la biblioteca SFML. Se organiza bajo una estructura modular que facilita el mantenimiento, la expansión y la legibilidad del código.

Compilador y Opciones

Se define el compilador y las banderas de compilación que aseguran compatibilidad con C++17 y mejores advertencias al programar.

- `CXX = g++` → Compilador GNU C++.
- `CXXFLAGS = -std=c++17 -Wall -Wextra` → Usa estándar C++17 y muestra advertencias comunes y adicionales.

Librerías SFML Necesarias

Se especifican las bibliotecas necesarias para correr el juego, incluyendo gráficos, sonido y eventos.

- `-lsfml-graphics` → Renderizado 2D.
- `-lsfml-window` → Ventana y entrada del teclado.
- `-lsfml-system` → Tareas generales del sistema.
- `-lsfml-audio` → Reproducción de música y efectos.

Directorios del Proyecto

Organización de archivos por función para mantener orden y escalabilidad:

- `src/` → Archivos fuente (.cpp).
- `include/` → Cabeceras (.h o .hpp).
- `build/` → Objetos compilados (.o).
- `resources/` → Recursos multimedia.
  - `textures/` → Imágenes del juego.
  - `music/` → Efectos y música.
  - `fonts/` → Tipografías utilizadas.
- `bin/` → Ejecutable compilado final.

Archivos Fuente

Listado de módulos principales del juego:

- `main.cpp` → Punto de entrada del juego.
- `Jugador.cpp` → Control del jugador.

- `Bala.cpp` → Disparo y manejo de balas.
- `Enemie.cpp` → Lógica de enemigos.
- `InputHandler.cpp` → Gestión de entradas del teclado.
- `Menu.cpp` → Interfaz de menú.
- `Muro.cpp` → Obstrucciones en el mapa.
- `Win.cpp` → Pantalla de victoria.
- `Lose.cpp` → Pantalla de derrota.
- `HUD.cpp` → Interfaz gráfica del jugador.

Convenciones para Variables

Las variables deben usar nombres descriptivos en camelCase.

Las constantes deben declararse con const y en UPPER_CASE.

Evitar abreviaciones como sc, pl, x1 fuera de bucles o cálculos temporales.

Las variables globales deben evitarse salvo que sean necesarias para el flujo del juego (por ejemplo: score, vida, screenWidth).

Ejemplos:

Copiar código

- int screenWidth; // Bien
- bool bulletActive; // Bien
- int sw; // Mal
- Vidas = 3; // Bien

Generar objetos correspondientes

- OBJ_FILES = $(patsubst $(SRC_DIR)/%.cpp, $(OBJ_DIR)/%.o, $(SRC_FILES))

Nombre del ejecutable

- TARGET = $(BIN_DIR)/SpaceInvadersEPN

Regla principal

- all: $(TARGET)

Enlaza todos los objetos en el ejecutable final

- $(TARGET): $(OBJ_FILES)
  - @mkdir -p $(BIN_DIR)
  - @echo Enlazando objetos...
  - $(CXX) $(OBJ_FILES) -o $@ $(LDFLAGS)

- @echo Compilación finalizada: $(TARGET)

Compila los archivos fuente en objetos

- $(OBJ_DIR)/%.o: $(SRC_DIR)/%.cpp
  - @mkdir -p $(OBJ_DIR)
  - @echo Compilando $<...
  - $(CXX) $(CXXFLAGS) -I$(INC_DIR) -c $< -o $@

Limpieza

- clean:
  - @echo Eliminando archivos de compilación...
  - rm -rf $(OBJ_DIR) $(BIN_DIR)

Paquete de distribución ZIP

- package:
  - @echo Empaquetando proyecto...
  - zip -r SpaceInvadersEPN.zip $(SRC_DIR) $(INC_DIR) $(RES_DIR) $(BIN_DIR) README.md

---

Ayuda

```
- @echo "Comandos disponibles:"
- @echo "  make         → Compila el juego"
- @echo "  make clean   → Elimina binarios y objetos"
- @echo "  make package → Crea un ZIP con el proyecto"
- @echo "  make help    → Muestra esta ayuda"
```

Créditos del equipo

- Proyecto desarrollado por:
- Jheramy Acurio
- Nicolás Bohórquez
- Alexander Cando
- Joseph Castro
- Esteban Chávez
- Alex Coral