Manual Técnico: Space Invaders EPN Este manual describe los componentes clave del proyecto Space Invaders EPN, un juego desarrollado en C++ utilizando la biblioteca SFML. El juego incluye un menú principal, un jugador, enemigos, balas, muros protectores, un HUD y pantallas de victoria y derrota. A continuación, se detalla cada archivo del proyecto, destacando su propósito y las partes más importantes.

1. Archivo: main2.cpp Propósito Archivo principal que inicializa el juego, maneja el menú principal y controla el bucle principal del juego. Gestiona la lógica de actualización y renderizado de los elementos del juego, como el jugador, enemigos, balas y muros. Componentes Clave

Variables Globales: spritesheet, spritesheetmuro: Texturas para los sprites del juego. bulletActive: Indica si la bala del jugador está activa. bulletsEnemies: Vector que almacena las balas disparadas por los enemigos. vida, score: Variables para rastrear las vidas y el puntaje del jugador. canciones, musicaJuego: Manejo de música de fondo aleatoria.

Funciones Principales: ReproducirCancionAleatoria: Selecciona y reproduce una canción aleatoria de la lista. UpdatePlayer: Actualiza la posición y el disparo del jugador. UpdateBulletPlayer: Maneja la lógica de la bala del jugador, incluyendo colisiones con enemigos y actualización del HUD. UpdateEnemies: Actualiza el movimiento de los enemigos. UpdateBulletsEnemies: Gestiona las balas enemigas y colisiones con el jugador. UpdateMuro: Maneja colisiones de balas (jugador y enemigos) con los muros protectores.

Bucle Principal: Inicializa la ventana en modo pantalla completa (RenderWindow). Crea un menú (Menu) y permite al usuario seleccionar "Play" o "Exit". Al seleccionar "Play", inicializa el jugador, enemigos, muros y HUD, y entra en el bucle del juego. Renderiza los elementos y actualiza el estado del juego.

2. Archivo: versionDemo.cpp Propósito Versión alternativa del archivo principal, similar a main2.cpp, pero sin el menú y con una configuración más simple. Sirve como una demostración básica del juego. Componentes Clave

Similar a main2.cpp, pero sin el manejo del menú ni la música de fondo. Inicializa el jugador, enemigos, balas y muros directamente. Contiene las mismas funciones de actualización (UpdatePlayer, UpdateBulletPlayer, UpdateEnemies, UpdateBulletsEnemies, UpdateMuro). No incluye HUD ni pantallas de victoria/derrota.

3. Archivo: Jugador.hpp y Jugador.cpp Propósito Define y gestiona la clase Player, que representa al jugador en el juego, incluyendo su movimiento, disparo y estado de vida. Componentes Clave

Clase Player: Atributos: sprite: Sprite del jugador. vida: Número de vidas (inicialmente 3). vel: Velocidad de movimiento (5 píxeles por frame). shoot, shootBuffer, shootSound: Gestionan el disparo y su sonido.

Métodos'. Player(int x, int y, Texture &texture): Constructor que inicializa el sprite y carga el sonido de disparo. Update(): Maneja el movimiento del jugador usando teclado (izquierda/derecha) o joystick. Shoot(): Detecta la acción de disparo (tecla Espacio o botón de joystick) y reproduce el sonido. QuitarVida(): Reduce la vida del jugador. Vivo(): Verifica si el jugador sigue vivo (vida != 0). Pos(): Devuelve la posición actual del sprite. draw(): Dibuja el sprite en la ventana.

4. Archivo: Bala.hpp y Bala.cpp Propósito Define y gestiona la clase Bullet, que representa las balas disparadas por el jugador o los enemigos. Componentes Clave

Clase Bullet: Atributos: sprite: Sprite de la bala. vel: Velocidad de la bala (positiva o negativa según la dirección). ubi: Almacena la posición previa en Y.

Métodos: Bullet(int x, int y, Texture &texture, IntRect intRect, int v): Constructor que inicializa el sprite, posición y velocidad. Update(): Mueve la bala verticalmente según su velocidad. Pos(): Devuelve la posición actual de la bala. getPreviousY(): Devuelve la posición previa en Y. draw(): Dibuja la bala en la ventana.

5. Archivo: Enemie.hpp y Enemie.cpp Propósito Define y gestiona la clase Enemie, que representa a los enemigos en el juego, incluyendo su movimiento, disparo y sonidos. Componentes Clave

Clase Enemie: Atributos: sprite: Sprite del enemigo. vel: Velocidad de movimiento horizontal. state, timer, cadencia: Controlan la animación y el movimiento. point: Coordenadas en el spritesheet. shoot, contadorDisparo: Gestionan el disparo del enemigo. shootBuffer, shootSound, explosionBuffer, explosionSound: Sonidos de disparo y explosión.

Métodos: Enemie(int x, int y, Texture &texture, Vector2f p): Constructor que inicializa el sprite y sonidos. Update(): Actualiza la posición y animación del enemigo. ChangeDir(): Invierte la dirección del movimiento y baja al enemigo. AumentarCadencia(): Reduce el tiempo entre movimientos. ActivarDisparo(): Activa el disparo y reproduce el sonido. UpdateDisparo(): Gestiona el estado del disparo. Pos(): Devuelve la posición del enemigo. draw(): Dibuja el sprite del enemigo.

6. Archivo: Muro.hpp y Muro.cpp Propósito Define y gestiona la clase Muro, que representa los muros protectores que pueden ser destruidos por balas. Componentes Clave

Clase Muro: Atributos: sprites: Vector de sprites para las partes del muro. state: Vector que almacena el estado de daño de cada parte.

Métodos: Muro(int x, int y, Texture &texture): Constructor que inicializa 5 sprites para el muro. Update(): Actualiza los sprites según el estado de daño. Pos(vector<pair<int,Vector2f>> &pos): Devuelve las posiciones de las partes no destruidas. Colision(int indice, bool up): Incrementa el daño de una parte del muro. draw(): Dibuja las partes no destruidas del muro.

7. Archivo: HUD.hpp y HUD.cpp Propósito Define y gestiona la clase HUD, que muestra la información de vidas y puntaje en pantalla. Componentes Clave

Clase HUD: Atributos: vidasText, scoreText: Textos para mostrar vidas y puntaje. vidas, score: Valores actuales de vidas y puntaje.

Métodos: HUD(Font &font): Constructor que inicializa los textos con la fuente proporcionada. updateVidas(int v): Actualiza el texto de vidas. updateScore(int s): Actualiza el texto de puntaje. draw(): Dibuja los textos en la ventana.

8. Archivo: Menu.hpp y Menu.cpp Propósito Define y gestiona la clase Menu, que muestra el menú principal con las opciones "Play" y "Exit". Componentes Clave

Clase Menu: Atributos: main_menu: Array de textos para las opciones ("Play" y "Exit"). main_menu_selected: Índice de la opción seleccionada. font: Fuente para los textos. menuMusic: Música de fondo del menú.

Métodos: Menu(float width, float height): Constructor que inicializa las opciones y la música. draw(): Dibuja las opciones del menú. MoveUp(), MoveDown(): Cambian la selección del menú. getSelectedOption(): Devuelve la opción seleccionada. PlayMusic(), StopMusic(): Controlan la música del menú.

9. Archivo: Lose.hpp y Lose.cpp Propósito Define y gestiona la clase Lose, que muestra la pantalla de derrota con las opciones "Play Again" y "Exit". Componentes Clave

Clase Lose: Atributos: gameOverText: Texto "GAME OVER". main_Game_Lose: Array de textos para las opciones. main_Game_Lose_selected: Índice de la opción seleccionada. font: Fuente para los textos. LoseMusic: Música de la pantalla de derrota.

Métodos: Lose(float width, float height): Constructor que inicializa los textos y la música. draw(): Dibuja los textos en la ventana. MoveUp(), MoveDown(): Cambian la selección de opciones. GameLosePressed(): Devuelve la opción seleccionada. PlayMusic(), StopMusic(): Controlan la música.

10. Archivo: Win.hpp y Win.cpp Propósito Define y gestiona la clase Win, que muestra la pantalla de victoria con las opciones "Play Again" y "Exit". Componentes Clave

Clase Win: Atributos: gameWinText: Texto "WELL DONE HERO!!!". main_Game_Win: Array de textos para las opciones. main_Game_Win_selected: Índice de la opción seleccionada. font: Fuente para los textos. WinMusic: Música de la pantalla de victoria.

Métodos: Win(float width, float height): Constructor que inicializa los textos y la música. draw(): Dibuja los textos en la ventana. MoveUp(), MoveDown(): Cambian la selección de opciones. GameWinPressed(): Devuelve la opción seleccionada. PlayMusic(), StopMusic(): Controlan la música.

11. Archivo: InputHandler.hpp Propósito Define la enumeración InputResult y la función manejarInputMenu para manejar la entrada del usuario en los menús. Componentes Clave

Enumeración InputResult: Define posibles acciones: Nada, Subir, Bajar, Seleccionar.

Función manejarInputMenu: Procesa eventos de teclado (Arriba, Abajo, Enter) y joystick (eje Y, botón A) para navegar y seleccionar opciones en el menú.

12. Archivo: Joystick.cpp Propósito Maneja la entrada de un joystick, actualizando y mostrando información sobre sus ejes y botones. Componentes Clave

Funciones: updateIdentification: Actualiza el nombre del joystick. updateAxes: Actualiza las posiciones de los ejes. updateButtons: Actualiza el estado de los botones. updateValues: Llama a las funciones anteriores para reflejar el estado del joystick.

Estructura: Utiliza un mapa (Texts) para almacenar etiquetas y valores de texto. Procesa los ejes (X, Y, etc.) y botones del joystick.

Notas Adicionales

Estructura del Juego: El proyecto sigue una arquitectura orientada a objetos, con clases que heredan de sf::Drawable para facilitar el renderizado. Recursos: Utiliza un spritesheet (spritesheet.png) para los gráficos y archivos de audio para disparos, explosiones y música. Interacción: Soporta entrada por teclado y joystick, con soporte para menús interactivos. Mejoras Potenciales: Corregir errores en la inicialización de muros en main2.cpp y versionDemo.cpp (número inconsistente de muros: 3 vs. 4). Optimizar la gestión de colisiones para mejorar el rendimiento. Agregar documentación adicional en el código para mejorar la mantenibilidad.