# Guía de Personalización de Space Invaders EPN

Este documento detalla cómo modificar la **velocidad de los disparos**, el **tamaño de letra**, el **tamaño de pantalla**, el **color de letra**, las **vidas**, el **tamaño de la nave**, el **tamaño de los enemigos**, el **sonido** y la **posición de las paredes** en el código de *Space Invaders EPN*, desarrollado en C++ con SFML. Las ubicaciones específicas están basadas en los archivos proporcionados.

## 1. Velocidad de los Disparos

### Ubicación

- **Archivo**: `main2.cpp`
- **Funciones**: `UpdatePlayer` (balas del jugador) y `UpdateEnemies` (balas de los enemigos).

### Código Relevante

- **Bala del Jugador**:

```
Bullet bullet(player.Pos().x + 24, player.Pos().y + 12, spritesheet,
IntRect(75, 122, 32, 32), -15);
```

  - El valor `-15` es la velocidad vertical (negativo porque sube).
- **Bala de los Enemigos**:

```
Bullet bullet = Bullet(enemigo.Pos().x + 9, enemigo.Pos().y - 12,
spritesheet, IntRect(43, 122, 32, 32), 15);
```

  - El valor `15` es la velocidad vertical (positivo porque baja).

### Cómo Cambiarlo

- Reemplaza `-15` (jugador) o `15` (enemigos) por un nuevo valor:
  - Más lento: `-10` o `10`.
  - Más rápido: `-20` o `20`.
- Compila y prueba el juego para ajustar la dificultad.

## 2. Tamaño de Letra

### Ubicación

- **Archivo**: `HUD.cpp`, `Menu.cpp`, `Win.cpp`, `Lose.cpp`
- **Método**: `setCharacterSize` en los constructores.

### Código Relevante

- **HUD.cpp**:

```
vidasText.setCharacterSize(24);
scoreText.setCharacterSize(24);
```

- **Menu.cpp**:

```
main_menu[0].setCharacterSize(50);
main_menu[1].setCharacterSize(50);
```

- **Win.cpp** (ejemplo):

```
gameWinText.setCharacterSize(60);
main_Game_Win[0].setCharacterSize(40);
main_Game_Win[1].setCharacterSize(40);
```

- **Lose.cpp** (similar):

```
gameOverText.setCharacterSize(60);
main_Game_Lose[0].setCharacterSize(40);
main_Game_Lose[1].setCharacterSize(40);
```

### Cómo Cambiarlo

- Busca `setCharacterSize` y ajusta los valores (e.g., `30` para más grande, `18` para más pequeño).
- Asegúrate de que el texto sea legible y encaje en la pantalla.

## 3. Tamaño de Pantalla

### Ubicación

- **Archivo**: `main2.cpp`
- **Líneas**: Creación de `menuWindow` y `window`.

### Código Relevante

- **Ventana del Menú**:

```
RenderWindow menuWindow(VideoMode::getDesktopMode(), "MENU SPACE INVADERS
EPN", Style::Fullscreen);
```

- **Ventana del Juego**:

```
RenderWindow window(VideoMode::getDesktopMode(), "Space Invaders EPN",
Style::Fullscreen);
```

  - Usa `VideoMode::getDesktopMode()` con `Style::Fullscreen` para pantalla completa.

### Cómo Cambiarlo

- Para un tamaño fijo, reemplaza con:

```
RenderWindow menuWindow(VideoMode(800, 600), "MENU SPACE INVADERS EPN");
RenderWindow window(VideoMode(800, 600), "Space Invaders EPN");
```

- Ajusta `800` (ancho) y `600` (alto) a las dimensiones deseadas (e.g., `1280, 720`).
- Modifica las posiciones de elementos (e.g., enemigos, muros) si es necesario.

## 4. Color de Letra

### Ubicación

- **Archivo**: `HUD.cpp`, `Menu.cpp`, `Win.cpp`, `Lose.cpp`
- **Método**: `setFillColor` en los constructores.

### Código Relevante

- **HUD.cpp**:

```
vidasText.setFillColor(sf::Color::White);
scoreText.setFillColor(sf::Color::White);
```

- **Menu.cpp**:

```
main_menu[0].setFillColor(Color::Green);
main_menu[1].setFillColor(Color::Red);
```

- **Win.cpp** (ejemplo):

```
gameWinText.setFillColor(Color::Yellow);
main_Game_Win[0].setFillColor(Color::Green);
main_Game_Win[1].setFillColor(Color::Red);
```

- **Lose.cpp** (similar):

```
gameOverText.setFillColor(Color::Blue);
main_Game_Lose[0].setFillColor(Color::Green);
main_Game_Lose[1].setFillColor(Color::Red);
```

### Cómo Cambiarlo

- Reemplaza los colores (e.g., `sf::Color::White` por `sf::Color::Red`, `sf::Color::Blue`, o `sf::Color(255, 0, 0)` para rojo personalizado).
- Usa `setOutlineColor` y `setOutlineThickness` para bordes (e.g., en `Win.cpp` con `gameWinText`).

## 5. Vidas

### Ubicación

- **Archivo**: `Jugador.cpp`, `main2.cpp`
- **Clase**: `Player`

### Código Relevante

- **Inicialización en `Jugador.cpp`**:

```
Player::Player(int x, int y, Texture &texture)
{
    // ... (otro código)
    vida = 3; // Número inicial de vidas
    // ...
}
```

- **Actualización en `main2.cpp`**:

```
int vida = 3; // Declaración global en main2.cpp
hud.updateVidas(vida); // Actualiza el HUD
```

### Cómo Cambiarlo

- En `Jugador.cpp`, cambia `vida = 3` a otro valor (e.g., `5` para 5 vidas).
- En `main2.cpp`, ajusta `int vida = 3` al mismo valor inicial.

## 6. Tamaño de la Nave

### Ubicación

- **Archivo**: `Jugador.cpp`
- **Clase**: `Player`

### Código Relevante

- **Constructor en `Jugador.cpp`**:

```
Player::Player(int x, int y, Texture &texture)
{
    sprite.setTexture(texture);
    sprite.setTextureRect(IntRect(0, 118, 32, 32)); // Región del
spritesheet
    sprite.setPosition(x, y);
    sprite.setScale(3, 3); // Escala de la nave
    // ...
}
```

  - `sprite.setScale(3, 3)` define el tamaño (3x el tamaño original de 32x32 píxeles).

### Cómo Cambiarlo

- Ajusta `sprite.setScale(3, 3)` a un nuevo valor (e.g., `2, 2` para más pequeño o `4, 4` para más grande).
- Ajusta `playerRect` en `UpdateBulletsEnemies` (en `main2.cpp`) si es necesario:

```
playerRect = IntRect(player.Pos().x, player.Pos().y + 9, 48, 15);
```

## 7. Tamaño de los Enemigos

### Ubicación

- **Archivo**: `Enemie.cpp`
- **Clase**: `Enemie`

### Código Relevante

- **Constructor en `Enemie.cpp`**:

```
Enemie::Enemie(int x, int y, Texture &texture, Vector2f p)
{
    sprite.setTexture(texture);
    sprite.setTextureRect(IntRect(point.x, point.y, 32, 32)); // Región del
spritesheet
    sprite.setPosition(x, y);
    sprite.setScale(2.5, 2.5); // Escala de los enemigos
    // ...
}
```

  - `sprite.setScale(2.5, 2.5)` define el tamaño (2.5x el tamaño original de 32x32 píxeles).

### Cómo Cambiarlo

- Modifica `sprite.setScale(2.5, 2.5)` a un nuevo valor (e.g., `2, 2` o `3, 3`).
- Ajusta `enemieRect` en `UpdateBulletPlayer` (en `main2.cpp`) si es necesario:

```
enemieRect = IntRect(static_cast<int>(enemies[i][j].Pos().x),
static_cast<int>(enemies[i][j].Pos().y), 32, 32);
```

# 8. Sonido

## Ubicación

- **Archivo**: `Jugador.cpp`, `Enemie.cpp`, `main2.cpp`
- **Clases**: `Player`, `Enemie`, y gestión de música en `main2.cpp`

## Código Relevante

- **Sonido de Disparo del Jugador (Jugador.cpp)**:

```
if (!shootBuffer.loadFromFile("sounds/Bala.wav"))
{
    cerr << "Error: No se pudo cargar el sonido de disparo.\n";
}
else
{
    shootSound.setBuffer(shootBuffer);
}
```

- **Sonido de Disparo del Enemigo (Enemie.cpp)**:

```
if (!shootBuffer.loadFromFile("sounds/shootenemy.wav"))
{
    cerr << "Error: No se pudo cargar el sonido de disparo.\n";
}
else
{
    shootSound.setBuffer(shootBuffer);
}
```

- **Música de Fondo (main2.cpp)**:

```
vector<string> canciones = {"Music/Play1.ogg", "Music/Play2.ogg"};
if (!musicaJuego.openFromFile(canciones[cancionActual]))
{
    cout << "No se pudo cargar la canción: " << canciones[cancionActual] <<
endl;
}
```

### Cómo Cambiarlo

- **Cambiar Archivos de Sonido**:
  - Reemplaza `Bala.wav`, `shootenemy.wav`, u otros en `sounds` con nuevos `.wav`.
- **Ajustar Volumen**:
  - Añade después de `setBuffer`:

```
shootSound.setVolume(50); // 50% (0-100)
```

  - Para música en `main2.cpp`:

```
musicaJuego.setVolume(60); // 60%
```

- **Cambiar Música**:
  - Modifica `canciones` (e.g., `"Music/NuevaCancion.ogg"`) y verifica los archivos en `Music`.

## 9. Posición de las Paredes

### Ubicación

- **Archivo**: `main2.cpp`
- **Función**: Inicialización de los muros en el bucle principal o función `main`.

### Código Relevante

- **Creación de Muros en `main2.cpp`**:

```
vector<Muro> muros;
muros.push_back(Muro(screenWidth * 0.25f, screenHeight * 0.75f,
spritesheet));
muros.push_back(Muro(screenWidth * 0.50f, screenHeight * 0.75f,
spritesheet));
muros.push_back(Muro(screenWidth * 0.75f, screenHeight * 0.75f,
spritesheet));
```

  - Las posiciones iniciales son aproximadamente 25%, 50% y 75% del ancho de la pantalla (`screenWidth`) en la parte inferior (`screenHeight * 0.75f`).

### Cómo Cambiarlo

- Modifica las coordenadas `x` y `y` en `Muro(screenWidth * 0.25f, screenHeight * 0.75f, spritesheet)`:
  - **Coordenada X**: Cambia `screenWidth * 0.25f`, `screenWidth * 0.50f`, y `screenWidth * 0.75f` a nuevos valores relativos (e.g., `screenWidth * 0.10f` para mover más a la izquierda, o

`screenWidth * 0.90f` para más a la derecha).
    - **Coordenada Y**: Ajusta `screenHeight * 0.75f` a un valor diferente (e.g., `screenHeight * 0.60f` para subirlas, o `screenHeight * 0.90f` para bajarlas).
    - Ejemplo de cambio a posiciones más cercanas al centro y más altas:

```
muros.push_back(Muro(screenWidth * 0.30f, screenHeight * 0.65f,
spritesheet));
muros.push_back(Muro(screenWidth * 0.50f, screenHeight * 0.65f,
spritesheet));
muros.push_back(Muro(screenWidth * 0.70f, screenHeight * 0.65f,
spritesheet));
```

- Asegúrate de que las nuevas posiciones no se solapen con el jugador o los enemigos, y ajusta las colisiones si es necesario.

## Recomendaciones

- **Pruebas**: Compila con `g++` y ejecuta `./SpaceInvaders` tras cada cambio.
- **Coherencia**: Ajusta colisiones y posiciones si cambias tamaños o ubicaciones.
- **Copia de Seguridad**: Haz un respaldo antes de modificar.