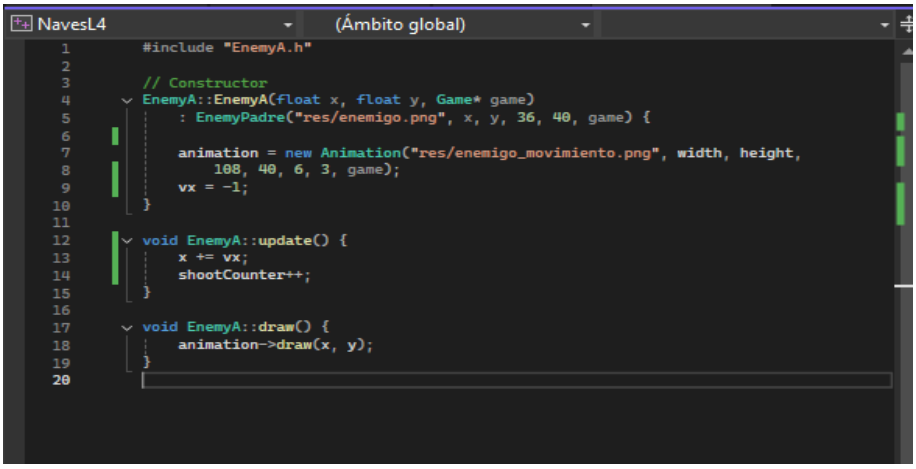
Informe de ampliaciones

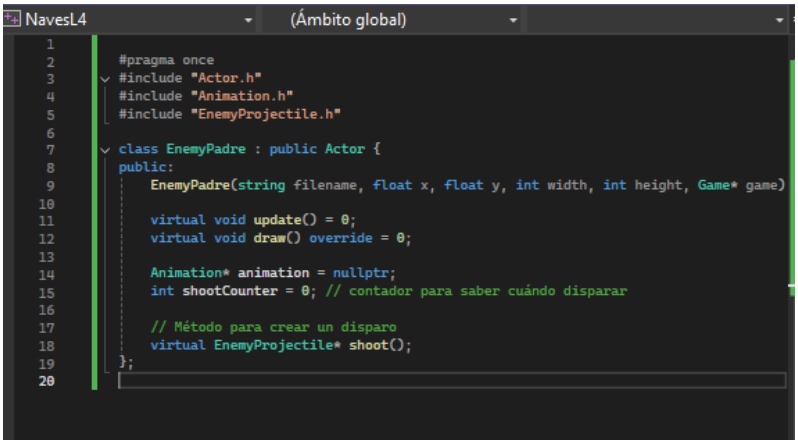
:)

Ikram El Mabrouk Morhnane

- **Ampliación 1: Nuevo tipo de enemigo**

Cree una clase abstracta EnemyPadre que sirve como base para los diferentes tipos de enemigos. Esta clase contiene atributos y métodos comunes, como la animación ,aunque en mi caso no hubo animación , y el comportamiento básico de actualización y dibujo.

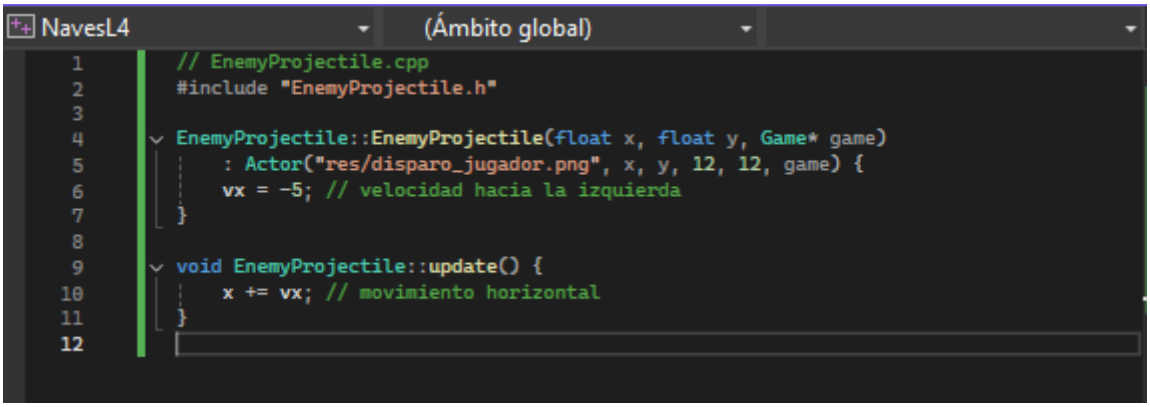


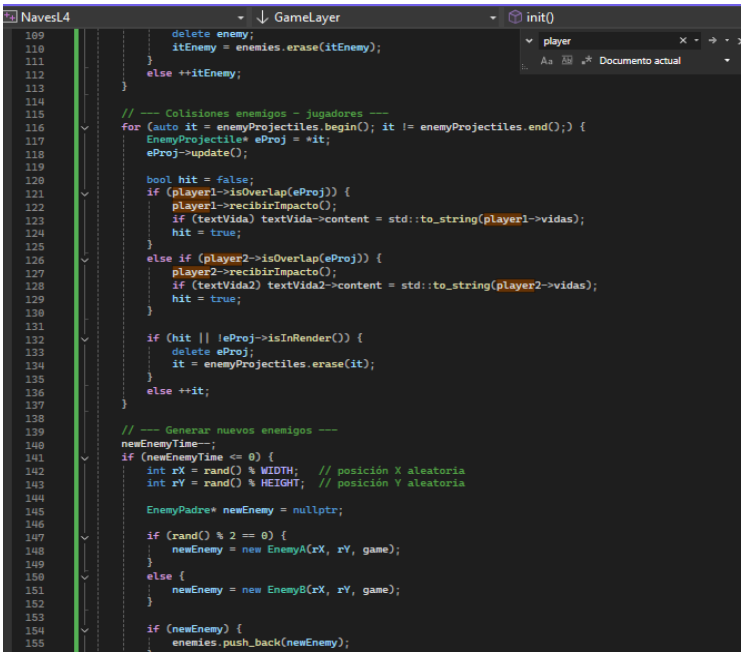


- **Ampliación 2: Enemigos con capacidad de disparo**

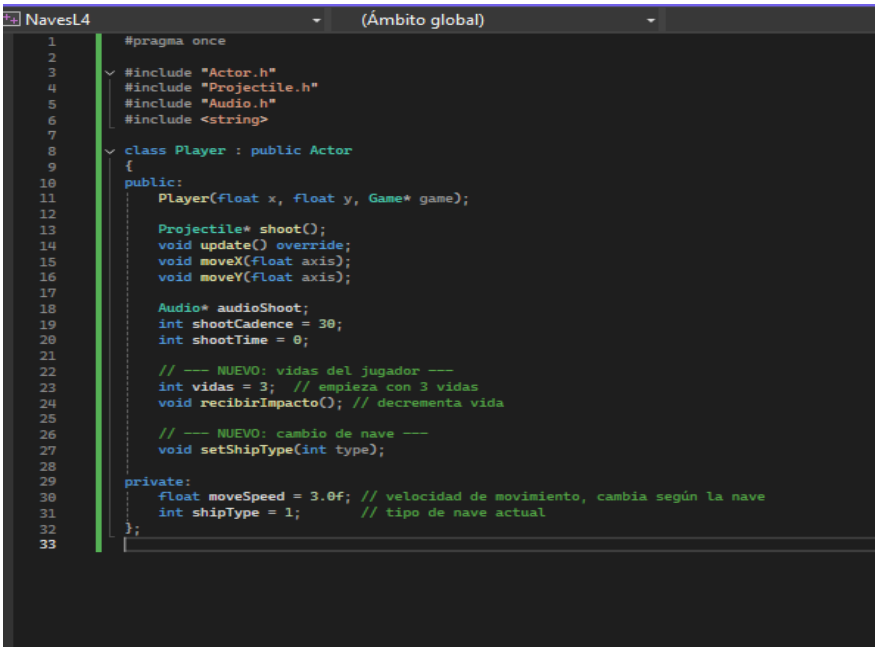
Se añadió a los enemigos la capacidad de disparar proyectiles hacia el jugador. Cada enemigo genera un disparo cada N iteraciones del juego, controladas mediante un contador interno (shootCounter).

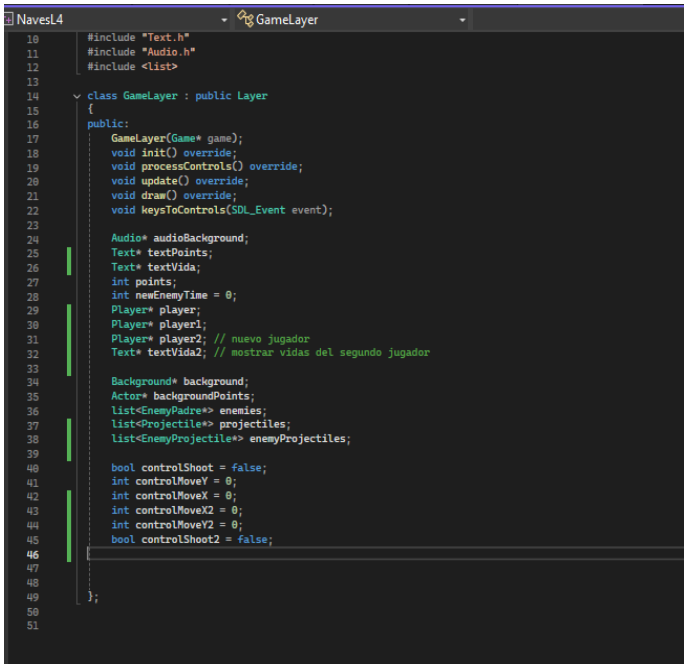
- Cada enemigo hereda de EnemyPadre y utiliza el método shoot() para generar proyectiles (EnemyProjectile).
- Los proyectiles se actualizan en cada iteración del juego y pueden colisionar con el jugador.
- La frecuencia de disparo se controla mediante un contador que se incrementa en cada update(). Cuando alcanza un valor definido (por ejemplo, cada 100 frames), el enemigo dispara.
- Los proyectiles se agregan a la lista de enemyProjectiles y se dibujan en pantalla.





- **Ampliación 3: Jugador con Vida**
  - Se implementó un sistema de vidas para el jugador:
  - El jugador empieza con 3 vidas (vidas).
  - Cada vez que un enemigo o un proyectil enemigo lo alcanza, pierde 1 vida.
  - Las vidas restantes se muestran en pantalla mediante un texto (textVida).
  - El juego no se reinicia automáticamente al perder vida; solo se reduce la vida y se elimina el enemigo o proyectil que causó el impacto.





- **Ampliación 5: Cambio de Nave**
  - Se añadieron dos naves disponibles para el jugador.
  - Al pulsar la tecla 1 o 2, se cambia la nave controlada por el jugador.
  - Cada nave tiene imagen, tamaño, velocidad de movimiento y cadencia de disparo diferente.
  - Esto permite variar la jugabilidad según la nave seleccionada.

```
void GameLayer::keysToControls(SDL_Event event) {
    int code = event.key.keysym.sym;

    if (event.type == SDL_KEYDOWN) {
        switch (code) {
            // --- Jugador 1 ---
        case SDLK_1: player1->setShipType(1); break;
        case SDLK_2: player1->setShipType(2); break;
        case SDLK_w: controlMoveY = -1; break;
        case SDLK_s: controlMoveY = 1; break;
        case SDLK_a: controlMoveX = -1; break;
        case SDLK_d: controlMoveX = 1; break;
        case SDLK_SPACE: controlShoot = true; break;

            // --- Jugador 2 ---
        case SDLK_UP: controlMoveY2 = -1; break;
        case SDLK_DOWN: controlMoveY2 = 1; break;
        case SDLK_LEFT: controlMoveX2 = -1; break;
        case SDLK_RIGHT: controlMoveX2 = 1; break;
        case SDLK_RETURN: controlShoot2 = true; break;

        case SDLK_ESCAPE: game->loopActive = false; break;
        }
    }
```

- **Ampliación 6: Multijugador Local**
  - Permite que dos jugadores controlen naves simultáneamente en la misma pantalla.
  - Cada jugador tiene sus propias teclas de control para moverse y disparar.
  - Cada nave tiene vidas independientes y dispara proyectiles de forma separada.
  - Mantiene la mecánica de colisiones y puntuación por jugador.

NavesL4 GameLayer

```
#include "Text.h"
#include "Audio.h"
#include <list>

class GameLayer : public Layer
{
public:
    GameLayer(Game* game);
    void init() override;
    void processControls() override;
    void update() override;
    void draw() override;
    void keysToControls(SDL_Event event);

    Audio* audioBackground;
    Text* textPoints;
    Text* textVida;
    int points;
    int newEnemyTime = 0;
    Player* player;
    Player* player1;
    Player* player2; // nuevo jugador
    Text* textVida2; // mostrar vidas del segundo jugador

    Background* background;
    Actor* backgroundPoints;
    list<EnemyPadre*> enemies;
    list<Projectile*> projectiles;
    list<EnemyProjectile*> enemyProjectiles;

    bool controlShoot = false;
    int controlMoveY = 0;
    int controlMoveX = 0;
    int controlMoveX2 = 0;
    int controlMoveY2 = 0;
    bool controlShoot2 = false;


};
```

```
void GameLayer::init() {
    audioBackground = new Audio("res/musica_ambiente.mp3", true);
    audioBackground->play();

    points = 0;
    textPoints = new Text("0", WIDTH * 0.92, HEIGHT * 0.04, game);

    // --- Jugadores ---
    player1 = new Player(50, 70, game);
    textVida = new Text(std::to_string(player1->vidas), 50, 20, game);

    player2 = new Player(50, 200, game); // posición inicial jugador 2
    textVida2 = new Text(std::to_string(player2->vidas), 700, 20, game);

    background = new Background("res/fondo.png", WIDTH * 0.5, HEIGHT * 0.5, -1, game);
    backgroundPoints = new Actor("res/icono_puntos.png", WIDTH * 0.85, HEIGHT * 0.05, 24, 24, game);

    enemies.clear();
    projectiles.clear();

    enemies.push_back(new EnemyA(300, 50, game));
    enemies.push_back(new EnemyB(300, 200, game));

    // Inicializar controles jugador 2
    controlMoveX2 = 0;
    controlMoveY2 = 0;
    controlShoot2 = false;
}
```

FIN