

JUGADOR 1



PUNTUACIÓN MÁS ALTA 2500



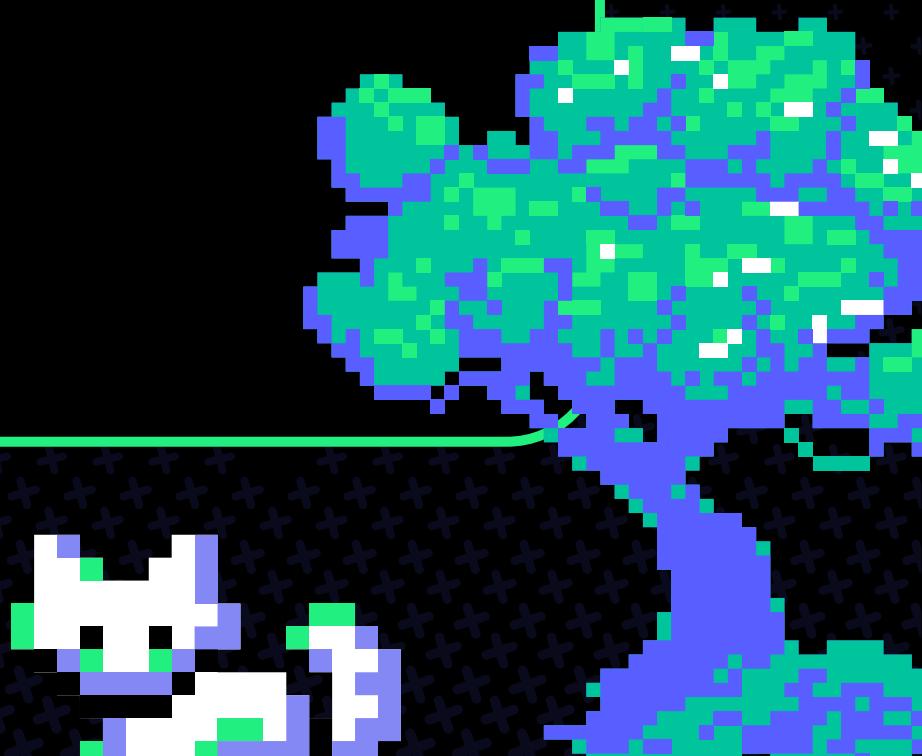
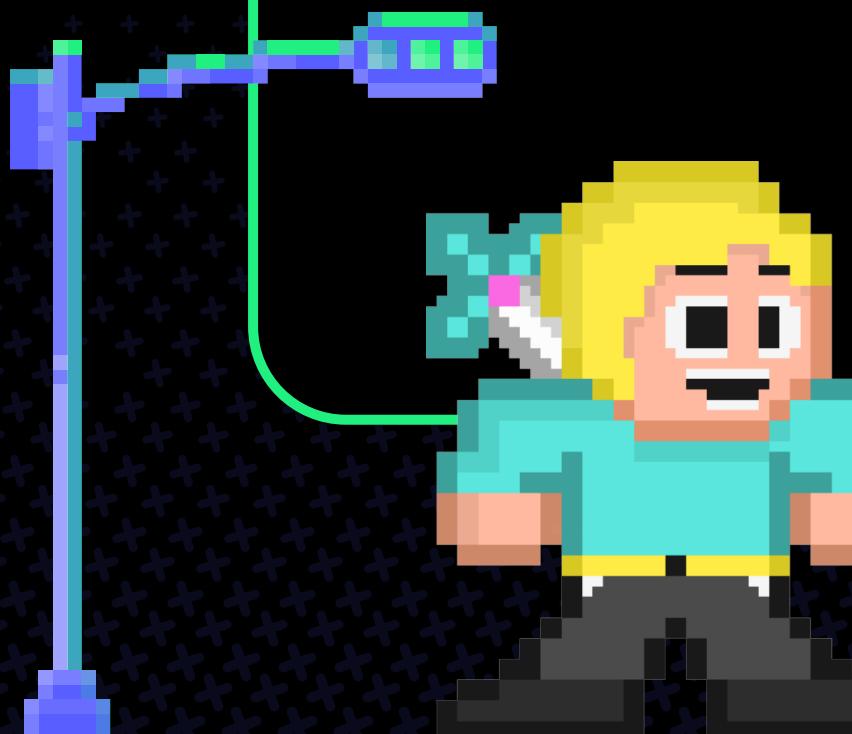
JUGADOR 2

# CAZADOR DE MONSTROS

START

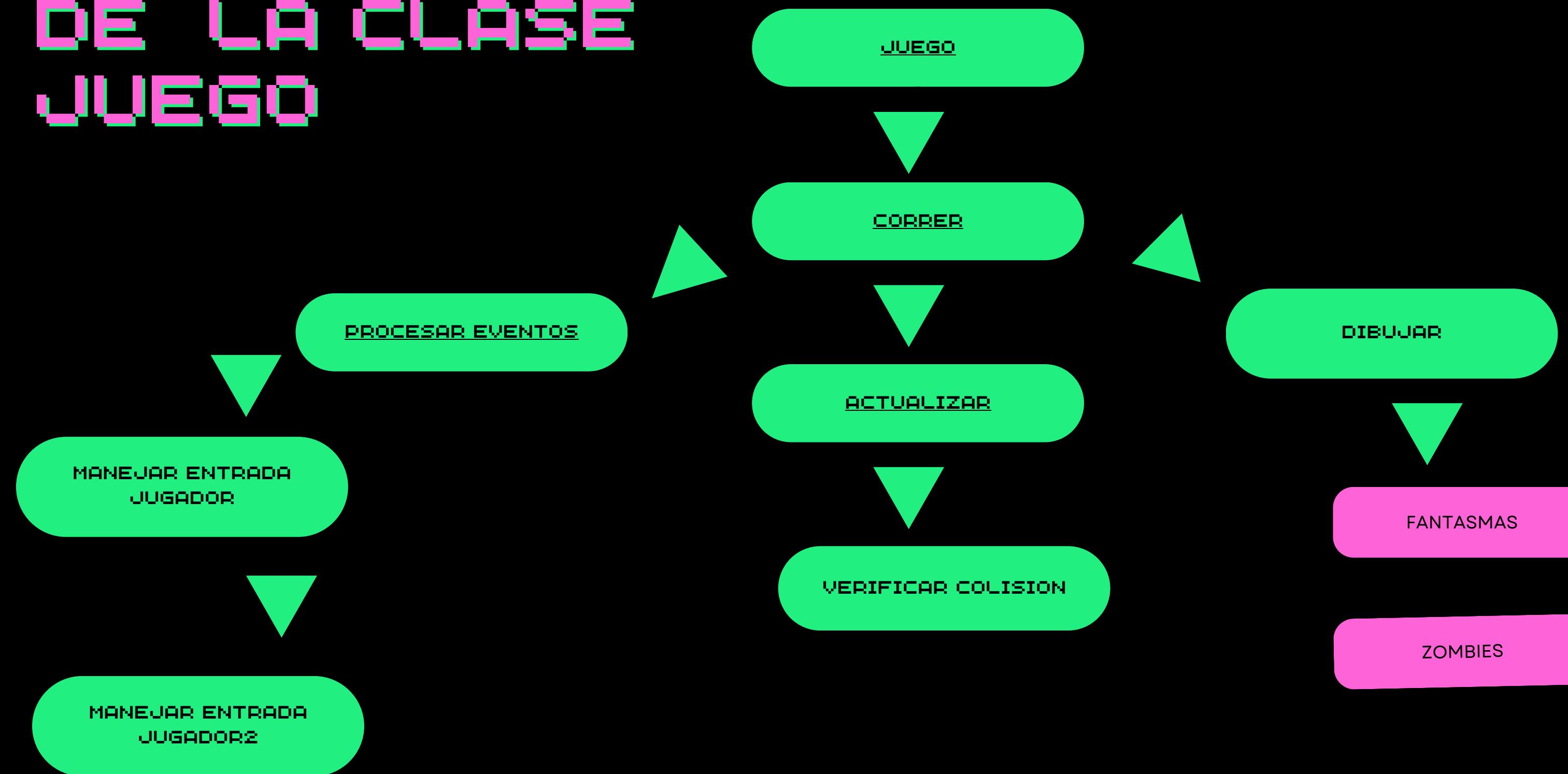
MENU

SIGN IN



◆ UTILIZANDO C++ Y SFML

# ESTRUCTURA DE LA CLASE JUEGO



# ATRIBUTOS DE LA CLASE JUEGO

```
class Juego {  
    Juego& operator=(const Juego&) = delete;  
    Juego();  
  
    void procesarEventos();  
    void actualizar();  
    void dibujar();  
    void manejarEntradaJugador(sf::Keyboard::Key tecla, bool presionada);  
    void manejarEntradaJugador2(sf::Keyboard::Key tecla, bool presionada);  
  
    void cargarJugadorTexturas();  
    void cargarJugador2Texturas();  
    bool verificarColision(const sf::Sprite& jugador);  
  
private:  
    Mapa mapa; ////////////////para mapaaaaaa  
  
    sf::Sprite jugador;  
    sf::Texture jugadorTex[4];  
    sf::Sprite jugador2;  
    sf::Texture jugador2Tex[4];  
    float velocidad;  
  
    sf::RenderWindow ventana;  
  
    sf::View vista;  
  
    sf::Clock reloj;  
    std::vector<sf::Vector2f> direcciones;  
    const float distanciaMaxima;  
  
    const float distanciaMaxima2; //Para los zombies  
    std::vector<std::unique_ptr<Zombie>> zombies;//Nueva funcionalidad  
    void generarZombies(std::vector<sf::Vector2f >&); //Generar zombies  
    sf::Texture zombiesTex[4];//texturas para los zombies  
  
    void generarFantasmas(std::vector<sf::Vector2f>& posicionesFantasmas);  
    sf::Texture monstruoTex[4];  
  
    std::vector<std::unique_ptr<Fantasma>> fantasmas;
```

MENU

01

07

12

# HEADERS



```
#ifndef MONSTRUO_H
#define MONSTRUO_H

#include <SFML/Graphics.hpp>
#include <vector>
#include <cmath>
#include <iostream>
#include <stdexcept>

class Monstruo { //Esta es la clase abstracta, de la que se dirabaran mas clases posteriormente
public:

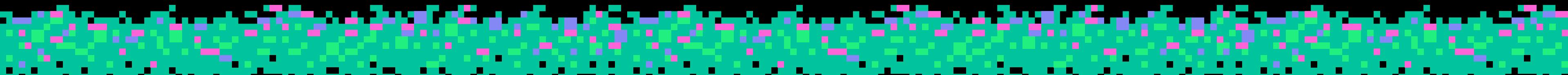
    //Todo esto se dibuja en al pantalla
    sf::Sprite monstruo;
    sf::Vector2f seEstaMoviendo;
    sf::RectangleShape barraVidaMonstruo;
    sf::Vector2f direccion;

    int vida;
    bool persiguiendo;
    float velocidad;

    //Las variables booleanas son para establecer una textura dependiendo del movimiento
    bool moverArriba;
    bool moverAbajo;
    bool moverDerecha;
    bool moverIzquierda;
    bool mostrar;
```



TEMAS TRATADOS



# CONSTRUCTOR

```
Juego::Juego()
: ventana(sf::VideoMode(1000, 1000), "Mi Juego")
, distanciaMaxima(1000)
, jugador()
, jugador2()
, rayoPotencialDireccion{ 1.f, 0.f, 0.f }
, rayoPotencialDireccion2{ 1.f, 0.f, 0.f }
, velocidad(5.f)
, mostrarFantasma(true)
, vidaFantasma(5)
, moviendoArriba(false)
, moviendoAbajo(false)
, moviendoIzquierda(false)
, moviendoDerecha(false)
, yaDisparo(false)
, disparando(false)
, moviendoArriba2(false)
, moviendoAbajo2(false)
, moviendoIzquierda2(false)
, moviendoDerecha2(false)
, yaDisparo2(false)
, disparando2(false)
, vidaJugador(10)
, vidaJugador2(10)
, distanciaMaxima2(1500)
```

```
ventana.setFramerateLimit(60);
jugador.setPosition(400.f, 300.f);
jugador2.setPosition(450.f, 300.f);

fondo.setSize(sf::Vector2f(10000.f, 10000.f));
fondo.setFillColor(sf::Color(
    9, 56, 5));

vista.setSize(ventana.getSize().x + 300, vista.getSize().y + 300);

//////////////////////////////para mapaaaaaa
auto crearCamino = std::make_unique<crearcamino>();
auto crearPasto = std::make_unique<crearpasto>();
auto crearFlor = std::make_unique<crearflor>();
auto crearFlor2 = std::make_unique<crearflor2>();
auto crearPasto1 = std::make_unique<crearpasto1>();
auto crearPasto2 = std::make_unique<crearpasto2>();
auto crearTumba1 = std::make_unique<creartumba1>();
auto crearTumba2 = std::make_unique<creartumba2>();
auto crearTumba3 = std::make_unique<creartumba3>();
auto crearTumba4 = std::make_unique<creartumba4>();
auto crearCasa = std::make_unique<crearcasa>();
auto crearLago = std::make_unique<crearlago>();

/////////////////////////////
```

[VUELVE A LA  
PÁGINA AGENDA](#)



AVANT

MENU

01

07

12



# CONSTRUCTOR



## INICIALIZACIÓN DE VARIABLES

- VENTANA::VIDEOMODE(1000, 1000), "MI JUEGO")
- DISTANCIAMAXIMA[1000]
- JUGADOR()
- JUGADOR2()
- RAYOPOTENCIALDIRECCION{1.F, 0.F, 0.F}
- RAYOPOTENCIALDIRECCION2{1.F, 0.F, 0.F}
- VELOCIDAD{5.F}
- MOSTRARFANTASMA[TRUE]
- VIDAFANTASMA{5}
- MOVIENDOARRIBA[FALSE]
- MOVIENDOABAJO[FALSE]
- MOVIENDOIZQUIERDA[FALSE]
- MOVIENDODERECHA[FALSE]
- YADISPANO[FALSE]
- DISPARANDO[FALSE]
- MOVIENDOARRIBA2[FALSE]
- MOVIENDOABAJO2[FALSE]
- MOVIENDOIZQUIERDA2[FALSE]
- MOVIENDODERECHA2[FALSE]
- YADISPANO2[FALSE]
- DISPARANDO2[FALSE]
- VIDAJUGADOR{10}
- VIDAJUGADOR2{10}
- DISTANCIAMAXIMA2[1500]

## CONFIGURACIÓN DE LA VENTANA Y JUGADORES

- VENTANA.SETFRAMERATELIMIT(60)
- JUGADOR.SETPOSITION(400.F, 300.F)
- JUGADOR2.SETPOSITION(450.F, 300.F)

## CONFIGURACIÓN DEL FONDO Y VISTA

- FONDO.SETSIZE(SF::VECTOR2F{10000.F, 10000.F})
- FONDO.SETFILLCOLOR(SF::COLOR{9, 56, 5})
- VISTA.SETSIZE(VENTANA.GETSIZE().X + 300, VENTANA.GETSIZE().Y + 300)

## CONFIGURACIÓN DEL MAPA, TEXTURAS Y PERSONAJES

1. CREACIÓN DE ELEMENTOS DEL MAPA
  - CREAR CAMINOS, PASTO, FLORES, TUMBAS, CASA, LAGO USANDO STD::MAKE\_UNIQUE
  - EJEMPLOS:
    - AUTO CREARCAMINO = STD::MAKE\_UNIQUE<CREARCAMINO>()
    - AUTO CREARPASTO = STD::MAKE\_UNIQUE<CREARPASTO>()
  - AGREGAR ELEMENTOS AL MAPA:
    - MAPA.CREACION(STD::MOVE(CREARLAGO))
    - MAPA.CREARELEMENTO({{700.F, 5200.F}})
    - REPETIR PARA OTROS ELEMENTOS
2. CARGA DE TEXTURAS
  - CARGARJUGADORTEXTURAS()
  - CARGARJUGADOR2TEXTURAS()
  - ZOMBIESTEX[0].LOADFROMFILE("./TEXTURAS/ZOMBIEBAJANDO.PNG")
  - REPETIR PARA OTRAS TEXTURAS DE ZOMBIES, RAYO Y MONSTRUOS
3. CONFIGURACIÓN DE FANTASMA
  - FANTASMA.SETTEXTURE(ZOMBIESTEX[0])
  - FANTASMA.SETSCALE{2.F, 2.F}
  - FANTASMA.SETPOSITION(800.F, 400.F)
4. CONFIGURACIÓN DE BARRAS DE VIDA
  - BARRAVIDAJUGADOR.SETSIZE(SF::VECTOR2F{100.F, 10.F})
  - BARRAVIDAJUGADOR2.SETSIZE(SF::VECTOR2F{100.F, 10.F})
  - BARRAVIDAJUGADOR.SETFILLCOLOR(SF::COLOR::GREEN)
  - BARRAVIDAJUGADOR2.SETFILLCOLOR(SF::COLOR::BLUE)
  - BARRAVIDAJUGADOR.SETPOSITION(JUGADOR.GETPOSITION().X, JUGADOR.GETPOSITION().Y)
  - BARRAVIDAJUGADOR2.SETPOSITION(JUGADOR2.GETPOSITION().X, JUGADOR2.GETPOSITION().Y)
5. CONFIGURACIÓN DE LA BARRA DE VIDA DEL FANTASMA
  - BARRAVIDAFANTASMA.SETSIZE(SF::VECTOR2F{40.F, 5.F})
  - BARRAVIDAFANTASMA.SETFILLCOLOR(SF::COLOR::RED)
  - BARRAVIDAFANTASMA.SETPOSITION(FANTASMA.GETPOSITION().X + 10, FANTASMA.GETPOSITION().Y - 5)
6. GENERACIÓN DE FANTASMAS Y ZOMBIES
  - STD::VECTOR<SF::VECTOR2F> POSICIONESFANTASMAS = {{50.F, 50.F}}
  - STD::VECTOR<SF::VECTOR2F> POSICIONESZOMBIES = {{50.F, 350.F}}
  - GENERARFANTASMAS(POSICIONESFANTASMAS)
  - GENERARZOMBIES(POSICIONESZOMBIES)

# CONSTRUCTOR

```
mapa.creacion(std::move(crearLago));
mapa.crearelemento({ {700.f, 5200.f} });

mapa.creacion(std::move(crearCasa));
mapa.crearelemento({ {900.f, 203.f} });

mapa.creacion(std::move(crearPasto1));
mapa.crearelemento({ {409.f, 5106.f} });

mapa.creacion(std::move(crearPasto2)); //E
mapa.crearelemento({ {827.f, 385.f} });

mapa.creacion(std::move(crearTumba1)); //I
mapa.crearelemento({ {7892, 3459}
| });

mapa.creacion(std::move(crearTumba2));
mapa.crearelemento({
| {220.f, 0.f}
| });

mapa.creacion(std::move(crearTumba3));
mapa.crearelemento({ {5003.f, 48.f}
| });

mapa.creacion(std::move(crearTumba4));
mapa.crearelemento({ {8020.f, 1000.f} });

mapa.creacion(std::move(crearFlor)); //hecl
mapa.crearelemento({
| {50.f, 50.f}
```

```
cargarJugadorTexturas();
cargarJugador2Texturas();

zombiesTex[0].loadFromFile("./texturas/zombieBajando.png");//Generar Zombies
zombiesTex[1].loadFromFile("./texturas/zombieSubiendo.png");
zombiesTex[2].loadFromFile("./texturas/zombieIzquierda.png");
zombiesTex[3].loadFromFile("./texturas/zombieDerecha.png");

fantasma.setTexture(zombiesTex[0]);
fantasma.setScale(2.f, 2.f);
fantasma.setPosition(800.f, 400.f);

barraVidaJugador.setSize(sf::Vector2f(100.f, 10.f));//Hecho
barraVidaJugador2.setSize(sf::Vector2f(100.f, 10.f));
barraVidaJugador.setFillColor(sf::Color::Green);
barraVidaJugador.setFillColor(sf::Color::Blue);
barraVidaJugador.setPosition(jugador.getPosition().x, jugador.getPosition().y);
barraVidaJugador2.setPosition(jugador2.getPosition().x, jugador2.getPosition().y);

barraVidaFantasma.setSize(sf::Vector2f(40.f, 5.f));
barraVidaFantasma.setFillColor(sf::Color::Red);
barraVidaFantasma.setPosition(fantasma.getPosition().x + 10, fantasma.getPosition().y - 5);

rayoTexturas[0].loadFromFile("./texturas/ataqueArriba.png");
rayoTexturas[1].loadFromFile("./texturas/ataqueAbajo.png");
rayoTexturas[2].loadFromFile("./texturas/ataqueIzquierda.png");
rayoTexturas[3].loadFromFile("./texturas/ataqueDerecha.png");

monstruoTex[0].loadFromFile("./texturas/fantasmaSubiendo.png");
monstruoTex[1].loadFromFile("./texturas/fantasmaBajando.png");
monstruoTex[2].loadFromFile("./texturas/fantasmaIzquierda.png");
monstruoTex[3].loadFromFile("./texturas/fantasmaDerecha.png");

std::vector<sf::Vector2f> posicionesFantasmas = { {50.f, 50.f} };
std::vector<sf::Vector2f> posicionesZombies = {
| {50.f, 350.f}
};

generarFantasmas(posicionesFantasmas);
generarZombies(posicionesZombies);
```

VUELVE A LA  
PÁGINA AGENDA



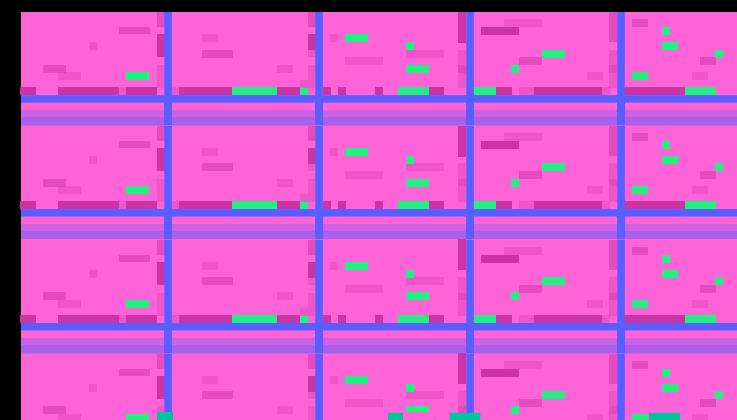
# FRACTI<sup>O</sup>N INICIAL DE MAPA



# JUGADOR 1



```
class ParametrosCreacion {
public:
    std::vector<sf::RectangleShape> caminos;
    std::vector<sf::RectangleShape> pastos2;
    std::vector<sf::Sprite> flores;
    sf::Texture texturaFlor;
    std::vector<sf::Sprite> flores2;
    sf::Texture texturaFlor2;
    std::vector<sf::Sprite> tumba1;
    sf::Texture texturaTumba1;
    std::vector<sf::Sprite> tumba2;
    sf::Texture texturaTumba2;
    std::vector<sf::Sprite> tumba3;
    sf::Texture texturaTumba3;
    std::vector<sf::Sprite> tumba4;
    sf::Texture texturaTumba4;
    std::vector<sf::Sprite> piedral;
    sf::Texture texturaPiedral;
    std::vector<sf::Sprite> piedra2;
    sf::Texture texturaPiedra2;
    std::vector<sf::Sprite> pasto1;
    sf::Texture texturaPasto1;
    std::vector<sf::Sprite> pasto3;
    sf::Texture texturaPasto3;
    std::vector<sf::Sprite> pasto2;
    sf::Texture texturaPasto2;
    std::vector<sf::Sprite> casa;
    sf::Texture texturaCasa;
    std::vector<sf::Sprite> lago;
    sf::Texture texturalago;
};
```

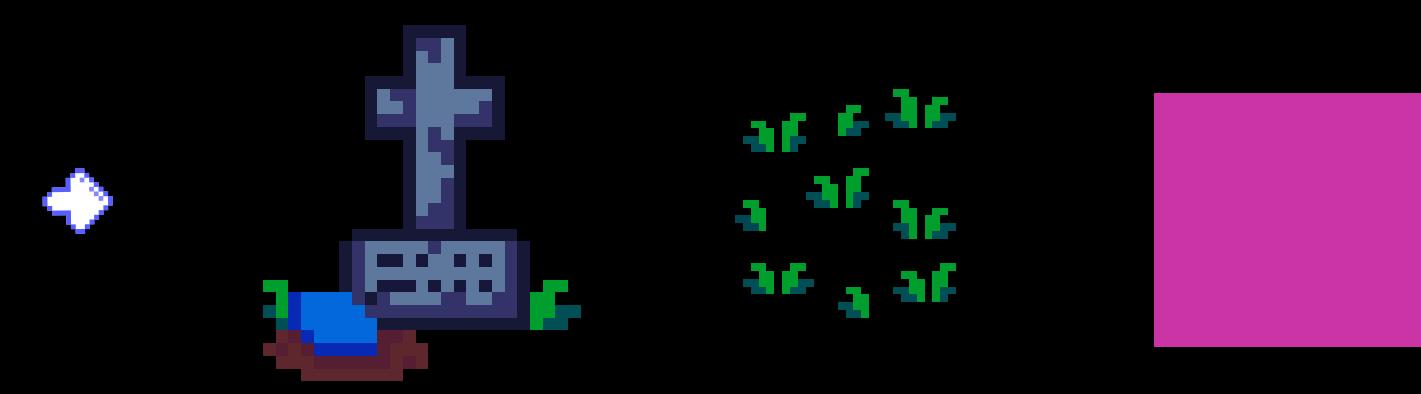


```
// Interfaz para crear elementos
class crearelemento {
public:
    virtual ~crearelemento() = default;
    virtual void crear(const std::vector<sf::Vector2f>& posiciones, ParametrosCreacion& parametros) const = 0;
};

// Clases concretas
class crearcamino : public crearelemento {
public:
    void crear(const std::vector<sf::Vector2f>& posiciones, ParametrosCreacion& parametros) const override;
};

class crearpasto : public crearelemento {
public:
    void crear(const std::vector<sf::Vector2f>& posiciones, ParametrosCreacion& parametros) const override;
};

class crearflor : public crearelemento {
public:
    void crear(const std::vector<sf::Vector2f>& posiciones, ParametrosCreacion& parametros) const override;
};
```



af & 36  
a af 36  
af a 36

# JUGADOR 1



```
class Mapa {
public:
    Mapa();
    void creacion(std::unique_ptr<crearelemento> generador);
    void crearelemento(const std::vector<sf::Vector2f>& posiciones);
    void dibujar(sf::RenderWindow& ventana);

    const std::vector<sf::RectangleShape>& obtenercaminos() const { return parametros.caminos; }
    const std::vector<sf::RectangleShape>& obtenerpastos2() const { return parametros.pastos2; }
    const std::vector<sf::Sprite>& obtenerflores() const { return parametros.flores; }
    const std::vector<sf::Sprite>& obtenerflores2() const { return parametros.flores2; }
    const std::vector<sf::Sprite>& obtenertumba1() const { return parametros.tumba1; }
    const std::vector<sf::Sprite>& obtenertumba2() const { return parametros.tumba2; }
    const std::vector<sf::Sprite>& obtenertumba3() const { return parametros.tumba3; }
    const std::vector<sf::Sprite>& obtenertumba4() const { return parametros.tumba4; }
    const std::vector<sf::Sprite>& obtenerpiedra1() const { return parametros.piedra1; }
    const std::vector<sf::Sprite>& obtenerpasto1() const { return parametros.pasto1; }
    const std::vector<sf::Sprite>& obtenerpasto3() const { return parametros.pasto3; }
    const std::vector<sf::Sprite>& obtenercasa() const { return parametros.casa; }
    const std::vector<sf::Sprite>& obtenerlago() const { return parametros.lago; }

private:
    void cargarTextura();
    ParametrosCreacion parametros;
    std::unique_ptr<crearelemento> generadorElemento;
};
```

```
Mapa::Mapa() {
    cargarTextura();
}

void Mapa::cargarTextura() {
    if (!parametros.texturaFlor.loadFromFile("./texturas/flor.png") ||
        !parametros.texturaFlor2.loadFromFile("./texturas/flor2.png") ||
        !parametros.texturaTumba1.loadFromFile("./texturas/tumbas_0001.png") ||
        !parametros.texturaTumba2.loadFromFile("./texturas/tumbas_0002.png") ||
        !parametros.texturaTumba3.loadFromFile("./texturas/tumbas_0003.png"))
```



```
void crearcamino::crear(const std::vector<sf::Vector2f>& posiciones, ParametrosCreacion& parametros) const {
    parametros.caminos.clear();
    for (const auto& pos : posiciones) {
        sf::RectangleShape camino(sf::Vector2f(300.f, 400.f));
        camino.setPosition(pos);
        camino.setFillColor(sf::Color(
            63, 49, 12));
        parametros.caminos.push_back(camino);
    }
}

void Mapa::creacion(std::unique_ptr<crearelemento> generador) {
    generadorElemento = std::move(generador);
}

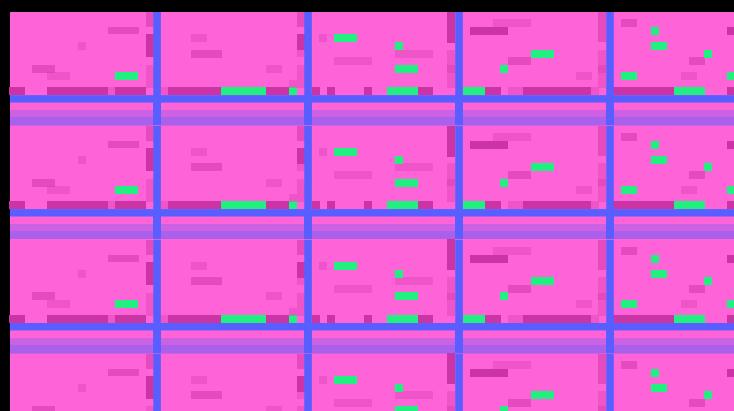
void Mapa::crearelemento(const std::vector<sf::Vector2f>& posiciones) {
    if (generadorElemento) {
        generadorElemento->crear(posiciones, parametros);
    }
}
```

JUGADOR 1 🌸

```
auto crearCamino = std::make_unique<crearcamino>();
auto crearPasto = std::make_unique<crearpasto>();
```

```
mapa.creacion(std::move(crearLago));
mapa.crearelemento({ {700.f, 5200.f}, {8600.f, 303.f}, {3000.f, 600.f} ,{ 9000.f, 7000.f } });
```

```
void Mapa::dibujar(sf::RenderWindow& ventana) {
    for (const auto& camino : parametros.caminos) {
        ventana.draw(camino);
    }
    for (const auto& pasto : parametros.pastos2) {
        ventana.draw(pasto);
    }
}
```



# SPRITES DE JUGADOR 1



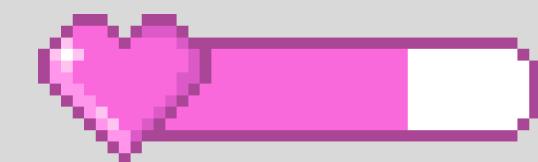
# SPRITES DE JUGADOR 2



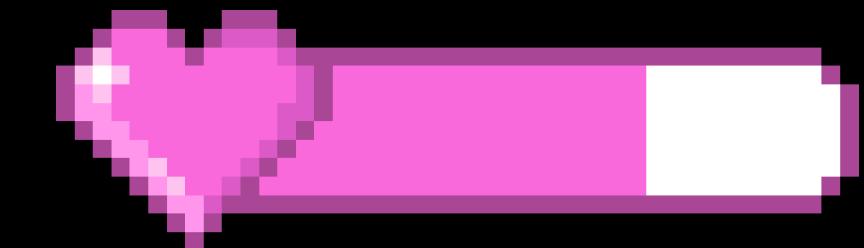
# SPRITES DE ATAQUE



# ENEMIGOS



# SPPRITES DE AMBIENTE



MENU



MUCHAS GRACIAS!