
LIF-FIGHTER
MORGENSTERN EMMA & DJAMAKORZIAN SASHA

INTRODUCTION

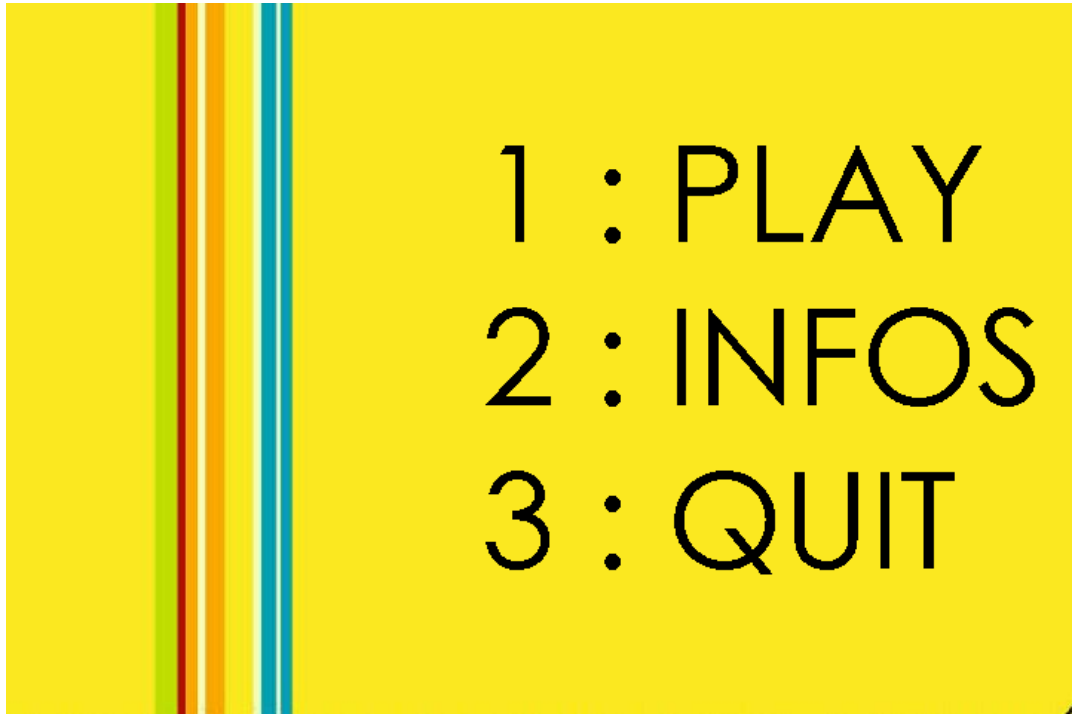
Notre projet consistera à imaginer et développer un jeu de combat 2D, similaire à Street Fighter, en C++ avec la bibliothèque graphique SDL2 et ses extensions.

Le jeu :

- se déroule dans un environnement 2D où 2 joueurs s'affrontent en utilisant différentes techniques de combat.

- Le but du jeu est de vaincre l'adversaire en lui infligeant des dégâts jusqu'à ce que sa barre de vie soit épuisée.

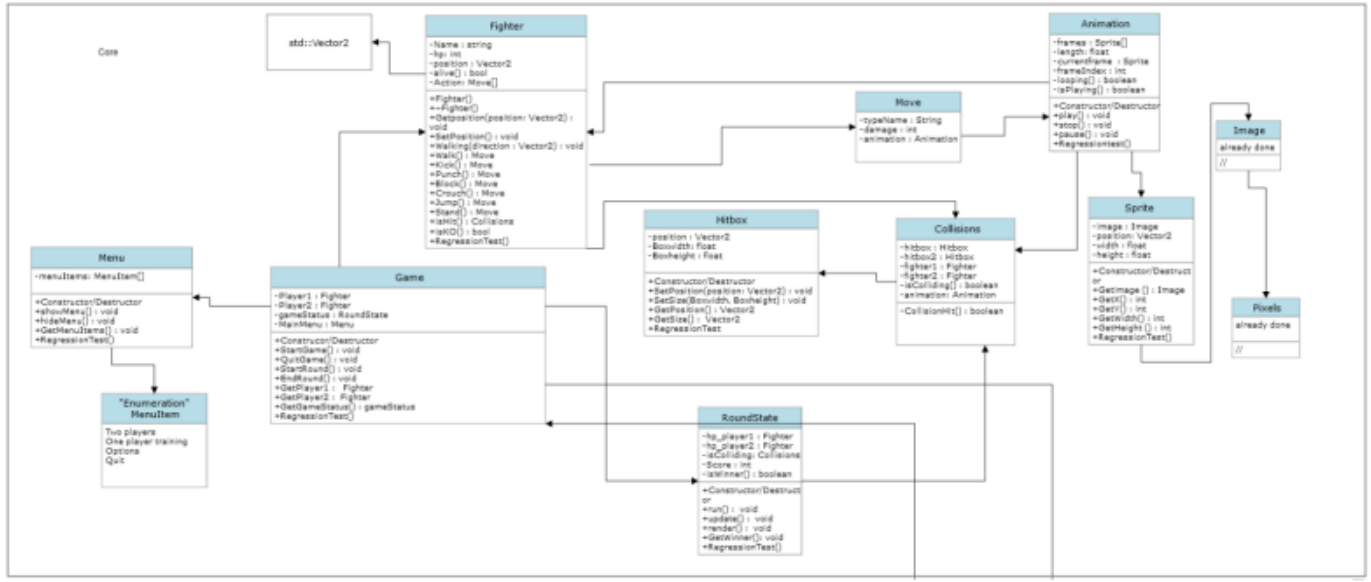
- Chaque joueur a le choix entre différents personnages pour vaincre son adversaire.
- Chaque personnage a deux attaques de base, un coup de poing et un coup de pied.
- Le jeu se déroule avec une musique de fond et différents bruitages en fonction des attaques.



Le jeu pourra donc se jouer à 2.

→A l'ouverture du jeu, un menu apparaît, il suffit de choisir le mode de jeu pour lancer une partie.

DIAGRAMME UML DE DÉBUT DE PROJET



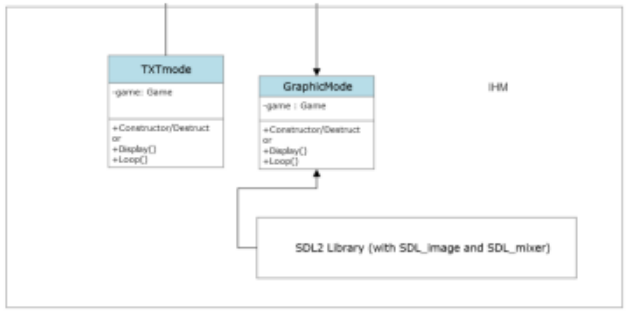
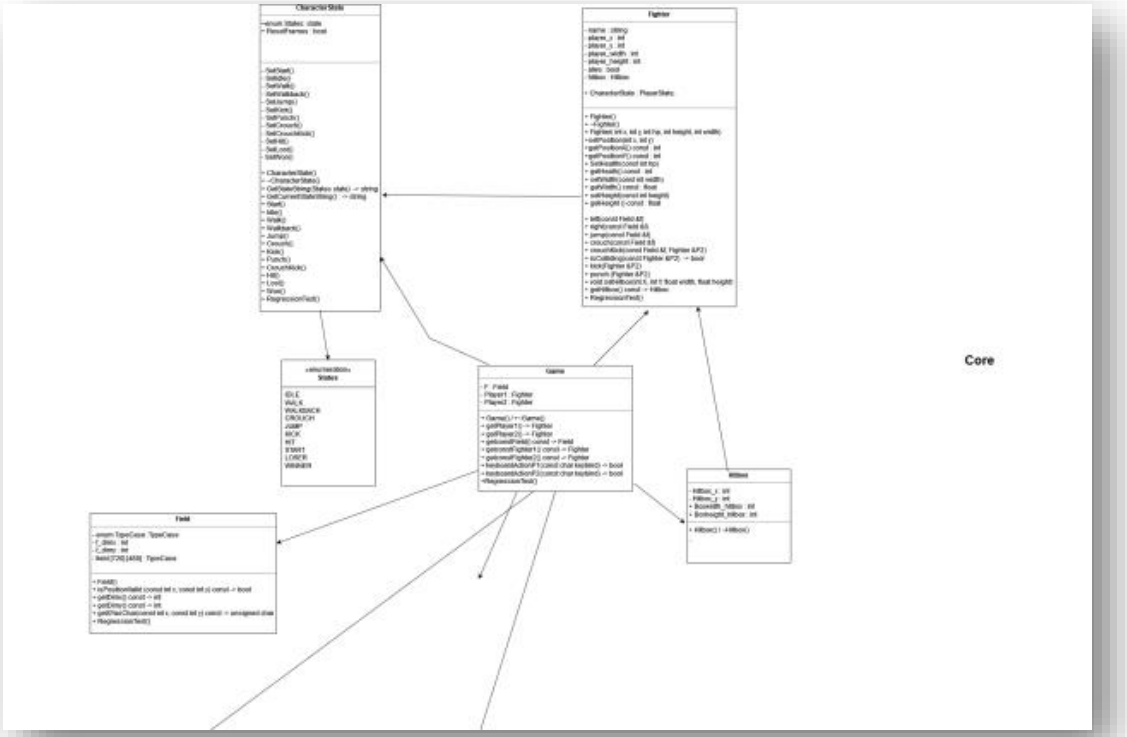
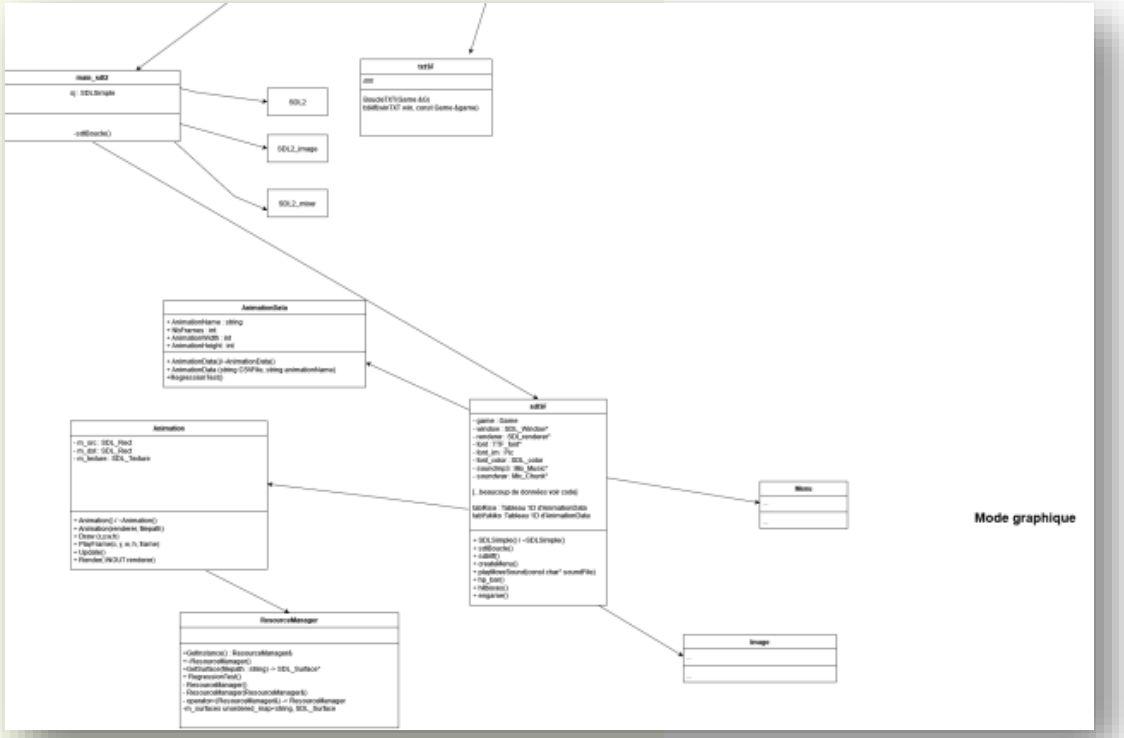


DIAGRAMME UML ACTUEL





1ÈRE CLASSE: CHARACTERSTATE.CPP/.H

```cpp
string CharacterState::GetStateString(States state)
{
    switch(state)
    {
    case States::IDLE:
       return "idle";
       break;
    case States::WALK:
        return "walk";
        break;
    case States::WALKBACK:
        return "walkback";
        break;
    case States::CROUCH:
        return "crouch";
        break;
    case States::JUMP:
        return "jump";
        break;
    case States::PUNCH:
        return "punch";
        break;
    case States::KICK:
        return "kick";
        break;
    case States::HIT:
        return "hit";
        break;
    case States::START:
        return "start";
        break;
    case States::LOSER:
        return "loser";
        break;
    case States::WINNER:
        return "winner";
        break;
    default: break;
    }
    return "aucune action";
}
```

```cpp
void CharacterState::SetIdle()
{
    if(state == States::IDLE)
    {
        return;
    }
    state = States::IDLE;
    ResetFrames = true;
}
```

```cpp
void CharacterState::Idle()
{
    SetIdle();
}
```

```cpp
//Initialisation des animations de Rise
tabRise[CharacterState::IDLE] = AnimationData(CSVFile: "data/Rise/AnimationsRise.csv",animationName: "idle");
tabRise[CharacterState::WALK] = AnimationData(CSVFile: "data/Rise/AnimationsRise.csv",animationName: "walk");
tabRise[CharacterState::WALKBACK] = AnimationData(CSVFile: "data/Rise/AnimationsRise.csv",animationName: "walkback");
tabRise[CharacterState::CROUCH] = AnimationData(CSVFile: "data/Rise/AnimationsRise.csv",animationName: "crouch");
tabRise[CharacterState::JUMP] = AnimationData(CSVFile: "data/Rise/AnimationsRise.csv",animationName: "jump");
tabRise[CharacterState::PUNCH] = AnimationData(CSVFile: "data/Rise/AnimationsRise.csv",animationName: "punch");
tabRise[CharacterState::KICK] = AnimationData(CSVFile: "data/Rise/AnimationsRise.csv",animationName: "kick");
tabRise[CharacterState::LOSER] = AnimationData(CSVFile: "data/Rise/AnimationsRise.csv",animationName: "loser");
tabRise[CharacterState::WINNER] = AnimationData(CSVFile: "data/Rise/AnimationsRise.csv",animationName: "winner");
tabRise[CharacterState::START] = AnimationData(CSVFile: "data/Rise/AnimationsRise.csv",animationName: "start");
tabRise[CharacterState::HIT] = AnimationData(CSVFile: "data/Rise/AnimationsRise.csv",animationName: "hit");
// Ending fight
im_EndingRise.loadFromFile("data/Rise/Ending_R.png", renderer);
im_NameRise.loadFromFile("data/Rise/Name_R.png", renderer);
```

2ÈME CLASSE: ANIMATIONDATA.CPP/.H

```cpp
#include "AnimationData.h"
#include <fstream>
#include <sstream>
#include <vector>
#include <string>

using namespace std;


AnimationData::AnimationData(){};

AnimationData::AnimationData(string CSVFile, string animationName)
    {
        AnimationName = animationName;
        vector<string> data;
        string line, value;
        ifstream file(CSVFile);

        while (getline(file,line))
        {
            if(line.rfind(animationName,0) != 0) //cherche animationName
            {
                continue;
            }
            stringstream stream(line);
            while(getline(stream,value,';'))
            {
                data.push_back(value); //rajoute dans le vecteur
            }
            NbFrames = stoi(data[1]);
            AnimationWidth = stoi(data[2]);
            AnimationHeight = stoi(data[3]);
            break;
        }

    }

AnimationData::~AnimationData(){};
```

3ÈME CLASSE: SDLSF.CPP/.H (PARTIE ANIMATION ET HITBOX)

```
Hitbox Player1Hitbox = player.gethitbox();
Hitbox Player2Hitbox = player2.gethitbox();
Player1Hitbox.Boxwidth_hitbox = AnimationWidth/NbFrames;
Player1Hitbox.Boxheight_hitbox = AnimationHeight;
Player2Hitbox.Boxwidth_hitbox = AnimationWidth2/NbFrames2;
Player2Hitbox.Boxheight_hitbox = AnimationHeight2;

SDL_Rect rectHitbox = {player.getPositionX()*SIZE_SPRITE,21*SIZE_SPRITE-AnimationHeight,Player1Hitbox.Boxwidth_hitbox,Player1Hitbox.Boxheight_hitbox};

SDL_Rect rectHitbox2 = {player2.getPositionX()*SIZE_SPRITE,21*SIZE_SPRITE-AnimationHeight2,Player2Hitbox.Boxwidth_hitbox,Player2Hitbox.Boxheight_hitbox};

if(SDL_HasIntersection(&rectHitbox,&rectHitbox2) && (player.PlayerState.state == CharacterState::States::PUNCH || player.PlayerState.state == CharacterState::States::KICK) && player2.PlayerState.state != CharacterState::States::HIT)
{
    player2.setHealth(hp: player2.getHealth()-1);
    player2.PlayerState.state = CharacterState::States::HIT;
}else if(SDL_HasIntersection(&rectHitbox,&rectHitbox2) && (player2.PlayerState.state == CharacterState::States::PUNCH || player2.PlayerState.state == CharacterState::States::KICK) && player.PlayerState.state != CharacterState::States::HIT)
{
    player.setHealth(hp: player.getHealth()-1);
    player.PlayerState.state = CharacterState::States::HIT;
}
SDL_SetRenderDrawColor(renderer, 255, 0, 0, 255);
```

```
/*Mise en place de l'animation du Joueur 1*/
Animation A(renderer,"data/Rise/spritesheets/spritesheetR_" + player.PlayerState.GetCurrentStateString() + ".png");
A.Draw(x: player.getPositionX()*SIZE_SPRITE,y: 21*SIZE_SPRITE-AnimationHeight,w: AnimationWidth/NbFrames,h: AnimationHeight);
A.PlayFrame(x: 0,y: 0,w: AnimationWidth/NbFrames,h: AnimationHeight,frame: frameNumber);
A.Render(renderer);

frameNumber++;
if(frameNumber>NbFrames-1){
    if(player.PlayerState.state == CharacterState::States::START || player.PlayerState.state == CharacterState::States::HIT)
    {
        game.keyboardActionP1(keybind: '\0');
    }
    frameNumber= 2;
}


/*Mise en place de l'animation du Joueur 2*/
Animation A2(renderer,"data/Yukiko/spritesheets/spritesheetY_" + player2.PlayerState.GetCurrentStateString() + ".png");
A2.Draw(x: player2.getPositionX()*SIZE_SPRITE,y: 21*SIZE_SPRITE-AnimationHeight2,w: AnimationWidth2/NbFrames2,h: AnimationHeight2);
A2.PlayFrame(x: 0,y: 0,w: AnimationWidth2/NbFrames2,h: AnimationHeight2,frame: frameNumber2);
A2.Render(renderer);

frameNumber2++;
if(frameNumber2>NbFrames2-1){
    if(player2.PlayerState.state == CharacterState::States::START || player2.PlayerState.state == CharacterState::States::HIT)
    {
        game.keyboardActionP2(keybind: '\0');
    }
    frameNumber2= 2;
}
```

4ÈME CLASSE: MENU.CPP/.H

```
// creation de la fenetre pour le menu
windowMenu = SDL_CreateWindow("MENU LIF-Fighter", SDL_WINDOWPOS_CENTERED, SD

if (windowMenu == nullptr)
{
    cout << "Erreur lors de la creation de la fenetre : " << SDL_GetError()
    SDL_Quit();
    exit(Code: 1);
}

// renderer du menu
rendererMenu = SDL_CreateRenderer(windowMenu, -1, 0);

// chargement image background menu
im_background.loadFromFile("data/menu_background.png", rendererMenu);
im_background.draw(rendererMenu, 0, 0, dimx, dimy);

// chargement surface du PLAY
im_txtPlay.setSurface(TTF_RenderText_Solid(font, "1 : PLAY", fontColor));
im_txtPlay.loadCurrentSurface(rendererMenu);
TTF_SizeText(font, "1 : PLAY", &posPlay.w, &posPlay.h);
SDL_RenderCopy(rendererMenu, im_txtPlay.getTexture(), NULL, &posPlay);

// chargement surface INFOS
im_txtInfos.setSurface(TTF_RenderText_Solid(font, "2 : INFOS", fontColor));
im_txtInfos.loadCurrentSurface(rendererMenu);
TTF_SizeText(font, "2 : INFOS", &posInfos.w, &posInfos.h);
SDL_RenderCopy(rendererMenu, im_txtInfos.getTexture(), nullptr, &posInfos);

// chargement surface QUIT
im_txtQuit.setSurface(TTF_RenderText_Solid(font, "3 : QUIT", fontColor));
im_txtQuit.loadCurrentSurface(rendererMenu);
TTF_SizeText(font, "3 : QUIT", &posQuit.w, &posQuit.h);
SDL_RenderCopy(rendererMenu, im_txtQuit.getTexture(), nullptr, &posQuit);

SDL_RenderPresent(rendererMenu);

SDL_Event events;
bool quit = false;
```

```
    SDL_Event events;
    bool quit = false;

    while (!quit)
    {
        while (SDL_PollEvent(&events))
        {
            if (events.type == SDL_QUIT)
                quit = true;
            else if (events.type == SDL_KEYDOWN)
            {
                switch (events.key.keysym.scancode)
                {
                case SDL_SCANCODE_1:
                    stateM = menuState::PLAY;
                    SDL_HideWindow(windowMenu);
                    quit = true;
                    break;
                case SDL_SCANCODE_2:
                    stateM = menuState::INFOS;
                    displayInfos();
                    cout << "infos.png" << endl;
                    break;
                case SDL_SCANCODE_3:
                case SDL_SCANCODE_ESCAPE:
                    stateM = menuState::QUIT;
                    quit = true;
                    break;
                default:
                    break;
                }
            }
        }
    }
    SDL_RenderPresent(rendererMenu);
};
```

CONCLUSION