TEMIRBOULATOV KOUREICH p1707160
BENZIANE  ABDELDJALLIL p11919796
MA SHENGQI p12108131

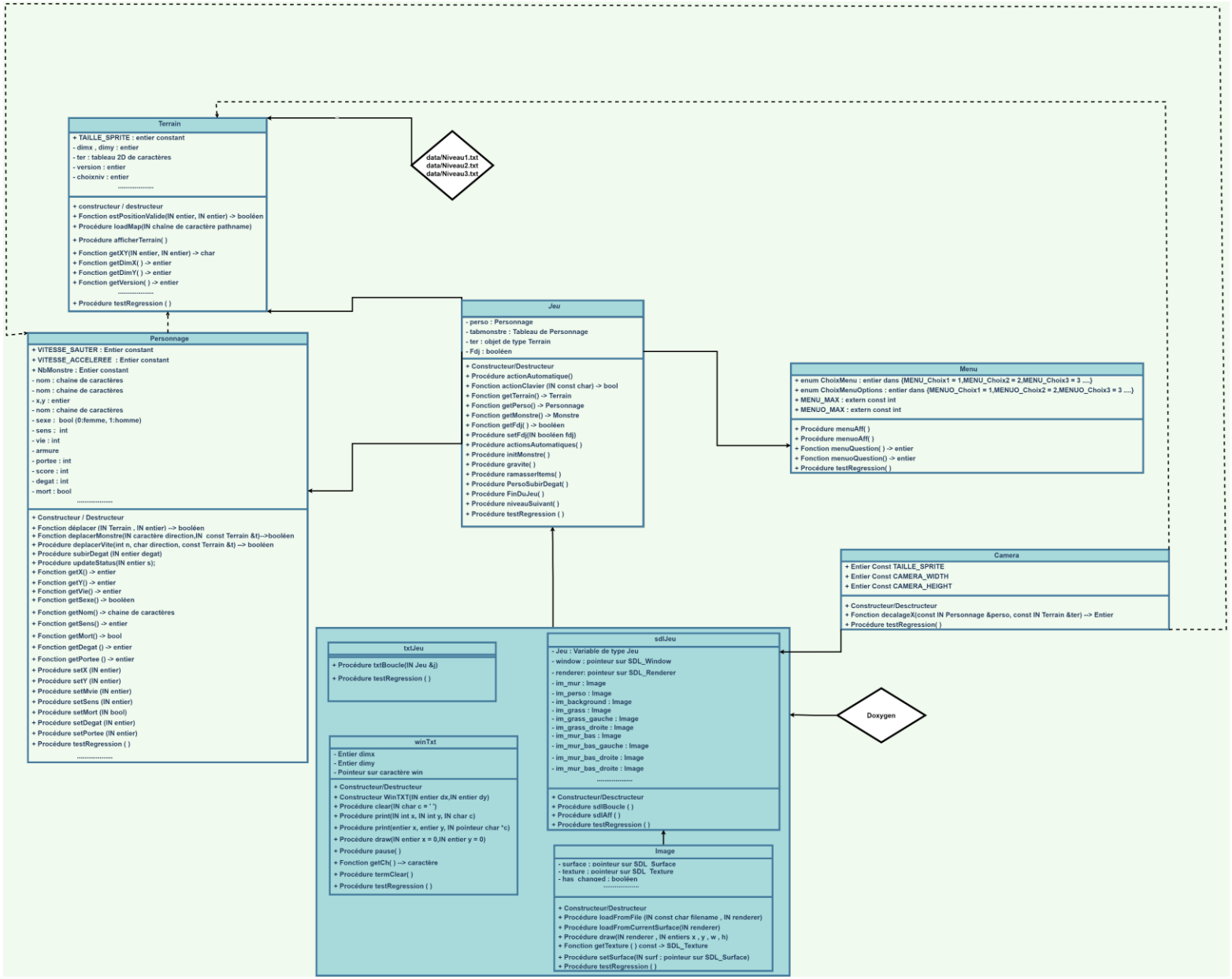
https://forge.univ-lyon1.fr/p1707160/projet-lifap4

# Présentation projet

## ULTIMATE-LIFKAS

## Jeu de plateforme 2D C/C++  SDL2

Diagramme des classes UML à jour :
- 9 classes



# Présentation en détails des classes Jeu , Personnage, SdlJeu et txtJeu

## Classe jeu

La classe Jeu est la classe principale qui gère les interactions entre les différents éléments du jeu , c'est-à-dire collisions , initialisation du personnage , des monstres , les dégâts , la gravité , le ramassage d'items …

### 0. Types de donnée membres

```
Personnage perso;
Personnage monst;
Personnage tabmonstre[18];
```

### 1. Initialisation de la classe

```
Jeu::Jeu() : ter(), perso()
{
        Fdj = false;
}
```

### 2. Initialisation des monstres selon la version de jeu : 1=SDL , 0=TXT

```
void Jeu::initMonstre()
{
        if (getTerrain().getVersion() == 1)
        {
                tabmonstre[0].setX(15 * TAILLE_SPRITE);
                tabmonstre[0].setY(3 * TAILLE_SPRITE);
                tabmonstre[1].setX(17 * TAILLE_SPRITE);
                tabmonstre[1].setY(3 * TAILLE_SPRITE);
                tabmonstre[2].setX(18 * TAILLE_SPRITE);
                tabmonstre[2].setY(3 * TAILLE_SPRITE);
                tabmonstre[3].setX(22 * TAILLE_SPRITE);
                tabmonstre[3].setY(3 * TAILLE_SPRITE);
                tabmonstre[4].setX(30 * TAILLE_SPRITE);
                tabmonstre[4].setY(3 * TAILLE_SPRITE);
                tabmonstre[5].setX(41 * TAILLE_SPRITE);
                tabmonstre[5].setY(3 * TAILLE_SPRITE);
                tabmonstre[6].setX(65 * TAILLE_SPRITE);
                tabmonstre[6].setY(6 * TAILLE_SPRITE);
                tabmonstre[7].setX(73 * TAILLE_SPRITE);
                tabmonstre[7].setY(3 * TAILLE_SPRITE);
                tabmonstre[8].setX(71 * TAILLE_SPRITE);
                tabmonstre[8].setY(3 * TAILLE_SPRITE);
                tabmonstre[9].setX(85 * TAILLE_SPRITE);
                tabmonstre[9].setY(3 * TAILLE_SPRITE);

                tabmonstre[10].setX(86 * TAILLE_SPRITE);
```

### 3. Gestion des action claviers pour déplacer le personnage . ActionAutomatique() sera appelée en boucle dans le main_txt ou main_sdl

```
bool Jeu::actionClavier(const int touche)
{
        switch (touche)
        {
        case 'q':
                perso.deplacerVite(VITESSE, touche, ter);
                break;
        case 'd':
                perso.deplacerVite(VITESSE, touche, ter);
                break;
        case 'z':
                perso.setSaute(true);
                break;
        case 's':
                perso.deplacerVite(VITESSE, touche, ter);
                break;
        case 'v':
                        for(int i = 0;i < NbMonstre;i++){
                        perso.attaquer(tabmonstre[i]);
                }
                break;
        }
        return false;
}

void Jeu::actionsAutomatiques()
{

        gravite();
        bougeAutoMonstre(ter);
        FinDuJeu();
        PersoSubirDegat();
        ramasserItems();
}
```

# Présentation en détails des classes Jeu , Personnage,SdlJeu et txtJeu

## Classe Personnage

La classe Personnage permet d'initialiser un Personnage tout en lui donnant un ensemble d'attributs , elle contient des accesseurs et mutateurs pour toutes les données membres .

### 0.Les attributs

```
class Personnage
{
private:
  int x, y;
  int vie;
  bool sexe;
  int sens;
  int sens_o;
  bool mort;
  std::string nom;
  int degat;
  int portee;
  int Armure;
  bool arme;
  int piece;
  int status;
  int status_o;
  int vitesse_gravite;
  bool saute;
```

### 1.Déplacements par rapport au terrain

```
bool Personnage::deplacer(char direction, const Terrain & t){

    if(direction == 'd' && t.estPositionPersoValide(x+1,y)){
      x++;
      updateStatus(2);//droite
      updateSens(1);
      return true;
    }else if(direction == 'q' && t.estPositionPersoValide(x-1,y)){
      x--;
      updateStatus(1);//gauche
      updateSens(-1);
      return true;
    }else if(direction == 's' && t.estPositionPersoValide(x,y+1)){
      y++;
      updateStatus(3);//tomper
      return true;
    }else if(direction == 'z' && t.estPositionPersoValide(x,y-1)){
      y--;
      updateStatus(4);//sauter
      return true;
    }else{
      updateStatus(0);//idle
      updateSens(sens);
      return false;
    }
}
```

### 2.Exemples de fonctions …

```
void Personnage::subirDegat(int degat){
  if(degat <= vie)
    vie -= degat;
  else
    vie = 0;
  if(vie <= 0)
    setMort(true);
}


void Personnage::attaquer(Personnage & perso){
  if(getArme())
  {
    if((x + portee * sens >= perso.getX() && y == perso.getY())) perso.subirDegat(degat);
    updateStatus(5);
  }
}


void Personnage::sauter(const Terrain & ter){
  if(VITESSE_SAUTER-(vitesse_gravite/VITESSE_ACCELEREE) > 0){
    deplacerVite(VITESSE_SAUTER-(vitesse_gravite/VITESSE_ACCELEREE), 'z', ter);
  }else{
    deplacerVite(-(VITESSE_SAUTER-(vitesse_gravite/VITESSE_ACCELEREE)), 's', ter);
  }
  vitesse_gravite++;
}

void Personnage::gravite(const Terrain & ter){
  if(!saute){
    if(deplacerVite(vitesse_gravite/VITESSE_ACCELEREE, 's', ter)){
      vitesse_gravite++;
    }else{
      vitesse_gravite = VITESSE_ACCELEREE;
    }
  }else{
    sauter(ter);
    if(status == 0) saute = false;
  }
}
```

# Présentation en détails des classes Jeu , Personnage,SdlJeu et txtJeu

## Classe sdljeu

La classe sdlJeu est la classe qui gère l'interface graphique , elle initialise la SDL/SDL2 et permet de jouer des audios selon les niveaux et les déplacements du personnage.

0.Appels des bibliothèques SDL et classes nécessaires

```
#include <SDL2/SDL.h>
#include <SDL2/SDL_image.h>
#include <SDL2/SDL_ttf.h>
#include <SDL2/SDL_image.h>
#include <SDL2/SDL_mixer.h>

#include "../core/Jeu.h"
#include "Image.h"
#include "../core/Camera.h"

class sdlJeu
{
private:
  Jeu jeu;

  SDL_Window *window;
  SDL_Renderer *renderer;
  SDL_Surface *im_icon;
  SDL_Event event;



  Image im_menu;
  Image im_mortperso;
  Image im_mur;
  Image im_background;
  Image im_ciel;
  Image im_ter;
  Image im_grass;
  Image im_grass_gauche;
  Image im_grass_droite;
  Image im_mur_bas;
```

1.Pour pouvoir gérer les sons et la musique

```
  Mix_Chunk *soundpiece;
  Mix_Chunk *soundvie;
  Mix_Chunk *soundarmure;
  Mix_Chunk *soundarme;
  Mix_Chunk *soundattack;
  Mix_Chunk *finniveau;
  Mix_Chunk *sonpas1;
  Mix_Chunk *sonpas2;

  Mix_Music *musique;
  Mix_Music *musique2;
  Mix_Music *musique3;
  bool withSound;
  bool withmusique;

  Camera camera;

  int i = 0;

public:
  const int TAILLE_SPRITE = 36;

  const int FPS = 20;
  bool jeupause = false;
```

2.Fonctions principales

```
void sdlBoucle();

void sdlAff():

void drawTerrain();

void drawPersonnage();

void drawMonstre();

SDL_Event getEvent() con

void drawInfoPerso();
```

3.Gestion des sons et musiques selon les actionsclavier

```
while (!quit && !jeu.getFdj())
{


  while (SDL_PollEvent(&events)) {
                    if (events.type == SDL_QUIT) quit = true;
                    else if (events.type == SDL_KEYDOWN) {
          //   bool mangepiece = false;
                            switch (events.key.keysym.sym) {
                            case SDLK_UP:
                            //      mangepiece = jeu.actionClavier();

                                    break;
                            case SDLK_DOWN:
                            //      mangepiece = jeu.actionClavier();
                                    break;
                            case SDLK_LEFT:
                            //      mangepiece = jeu.actionClavier();
                                    Mix_PlayChannel(-1,sonpas2,0);

                                    break;
                            case SDLK_RIGHT:
                            //      mangepiece = jeu.actionClavier();
                                    Mix_PlayChannel(-1,sonpas1,0);
                                    break;
    case SDLK_v:
      Mix_PlayChannel(-1,soundattack,0);
      break;
    case SDLK_m:
      if (Mix_PausedMusic() == 1) // Si la musique est en pause
      {
        Mix_ResumeMusic(); // Reprendre la musique
      }
```

# Présentation en détails des classes Jeu , Personnage,SdlJeu et txtJeu

## Classe txtJeu

La classe txtJeu est la classe qui gère l'interface textuelle. L'affichage des informations (le USER INTERFACE) et les déplacements des éléments du jeu

0. Très peu de Fonctions membres

```
#ifndef _TXTJEU_H
#define _TXTJEU_H

#include "../core/Jeu.h"

void txtBoucle(Jeu &j);



void testRegression();

#endif
```

1. Fonction txtAff qui affiche le personnage et les monstres à l'écran

```
void txtAff(WinTXT &win, Jeu &jeu)
{

         Terrain& ter = jeu.getTerrain();
         Terrain terconst = jeu.getConstTerrain();
         Personnage& perso = jeu.getPerso();

         Personnage Tmonst[18];
         for (int i=0;i<NbMonstre;i++){ Tmonst[i] = jeu.getConstMonstre(i);}

        win.clear();

        int xp,xd=0;
        if(perso.getX()>20){xp = perso.getX()-20;}else{xp=0;}
        xd = perso.getX()+25;
        if(xp == perso.getX()+1) {xp++;}

         // Affichage des murs,des plateformes et des monstres selon le champ de vision du personnage
        if(xd){

                for(int x=xp;x<xd;++x)
                for(int y=0;y<ter.getDimY();++y)
                win.print(x,y,ter.getXY(x,y));
                for(int i=0;i<NbMonstre;i++){
                        if(Tmonst[i].getX()<=xd && Tmonst[i].getX()>=xp){
                win.print(Tmonst[i].getX(),Tmonst[i].getY(),'M');
                }
                }

        }else{
                for(int x=0;x<ter.getDimX()/2;++x)
                for(int y=0;y<ter.getDimY();++y)
                win.print(x,y,ter.getXY(x,y));
                for(int i=0;i<NbMonstre;i++){win.print(Tmonst[i].getX(),Tmonst[i].getY(),'M');}

        }

        cout << endl << endl<<"PSEUDO: "<<perso.getNom()<<" Vie:"<<perso.getVie()<<" Armure:"
        <<perso.getArmure()<<" Degats:"<<perso.getDegat()<<" Portee:"<<perso.getPortee()<<" Piece:"<<perso.getPiece()
        <<" positionPerso: {x:"<<perso.getX()<<",y:"<<perso.getY()<<"}"<< endl;

        cout <<endl<< "Appuyez sur X pour quitter et retourner au menu" << endl;
```

2. txtBoucle(Jeu &jeu) va appeler les fonctions txtAff , actionautomatique et actionCalvier en boucle

```
void txtBoucle(Jeu &jeu)
{
        // Creation d'une nouvelle fenetre en mode texte
        // => fenetre de dimension et position (WIDTH,HEIGHT,STARTX,STARTY)
        WinTXT win(jeu.getConstTerrain().getDimX(), jeu.getConstTerrain().getDimY());
        jeu.initMonstre();

        char c;

        do
        {
                txtAff(win, jeu);
                #ifdef _WIN32
                Sleep(100);
                #else
                usleep(100000);
                #endif // WIN32


                c = win.getCh();

                jeu.actionsAutomatiques();
                jeu.actionClavier(c);

        //      if(jeu.getFdj()) AffFin(win, jeu);

                if (c == 'x') {
                        jeu.setFdj(true);
                        //fflush(stdout);
                        }

        } while (!jeu.getFdj());

}
```

# Objectifs de projet

Fonctionnalités:

- Choix du sexe du personnage
- Sélection du niveau
- Déplacement du personnage dans la direction souhaitée
- Possibilité d'envoyer une attaque
- Nombre de points de vie visible
- Présence de boss en fin de niveau
- Obtention d'arme et pièce (score)
- Possibilité de jouer à plusieurs niveaux différents.
- Menu en version texte.
- Menu en version graphique.

Gestion des scores :

- Le score initial du jeu est de 0.
- Au fur et à mesure que le temps passe, le score du joueur augmente.
- Lorsque nous tuons des monstres, nous obtenons une petite quantité de pièces et de points.
- Lorsque nous tuons un boss, nous obtenons une petite quantité de pièces et de points.
- Réussir un niveau permet de gagner une grande quantité de pièces et de points (cela peut dépendre du temps que le joueur met à passer le niveau, plus le temps est court, plus le bonus est élevé).

Légende

Atteint.

Non-atteint mais sachant faire

Non-atteint et abandonnée

# Objectifs de projet

Contraintes :

-Le projet doit être développé en 8 semaines.

-Le jeu sera développé en C/C++ sous Linux Les librairies utilisées seront SDL2 et une librairie texte pour une version alpha.

-Le code respectera le standard suivant : code indenté, variable ayant du sens, ...

-Le code sera géré et archivé sur la Forge Lyon 1.

-La documentation du code sera produite par Doxygen

-Un diagramme des classes permettra d'avoir une vision de haut niveau de l'implantation

Légende

Atteint.

Non-atteint mais sachant faire

Non-atteint

Non-atteint et abandonnée

# Déroulement du projet par rapport au diagramme de Gantt

## Ensemble des tâches :

Tâche 0 : Rédiger le cahier des charges
Tâche 1:  Définir le diagramme des classes
Tâche 2 : Implémentation d'un terrain en version texte, Déplacement du personnage, Apparition d'ennemi, Déplacement des ennemis automatiquement
Tâche 3 : Implémentation d'un terrain, déplacement du personnage et des ennemis, etc…
Tâche 4 : Conception de l'intelligence artificielle
Tâche 5 : Implémentation de l'Intelligence Artificielle (ennemi qui charge vers le personnage principal)
Tâche 6 : Test du jeu basique et fonctionnel en version texte
Tâche 7 : Conception du jeu dans une interface graphique
Tâche 8  : Implémentation du jeu dans une interface graphique
Tâche 9 : Test du jeu avec l'interface graphique (Toutes les fonctionnalités)
Tâche 10 : Optimisation du jeu (Valgrind)
Tâche 11 : Test du projet

Diagramme de Gantt

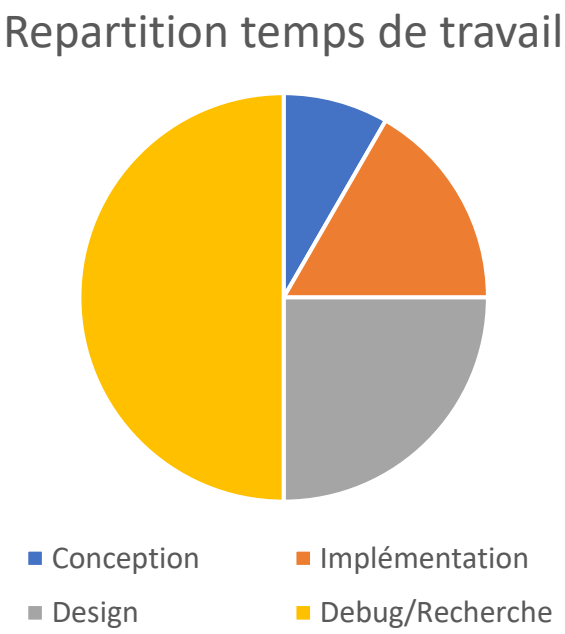
| Tâche/Semaine | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 : Conception des règles du jeu | ■ | ■ | | | | | | |
| 2 : Conception des fonctionnalités basiques | | ■ | ■ | | | | | |
| 3 : Implémentation des fonctions du jeu | | ■ | ■ | ■ | | | | |
| 4 : Conception de l'intelligence artificielle | | | ■ | ■ | ■ | | | |
| 5 : Implémentation de l'Intelligence Artificielle | | | ■ | ■ | ■ | | | |
| 6 : Test du jeu basique et fonctionnel | | | | | ■ | | | |
| 7 : Conception du jeu dans une interface graphique | | | | | ■ | ■ | ■ | |
| 8 : Implémentation du jeu dans une l'interface graphique | | | | | ■ | ■ | ■ | ■ |
| 9 : Test du jeu avec l'interface graphique | | | | | | | ■ | ■ |
| 10 : Optimisation du jeu | | | | | | | | ■ |
| 11 : Test du projet | | | | | | | | ■ |

<u>Légende</u>
Atteint à l'avance
Atteint à temps
Atteint en retard
Non-atteint et abandonnée

UCBL LIFAP4 – Printemps 2022

Université Claude Bernard Lyon 1

# Conclusion




-Tous les objectifs fixés au départ non pas étaient atteints
-Un temps pour le design a été consacré pour fournir une version plus agréable visuellement
-Plus de temps pour le développement nous aurait certainement permis d'aboutir le projet à 100%
-Beaucoup trop de temps consacré au debug …