Code source

PROJET LA BAGARRE

Quentin Le Guellanff | Anthime Louvet | Emmanuelle Zenou | Miguel de Oliveira
PROFESSEUR REFERENT : MME KELLER | IUT ORSAY DUT INFORMATIQUE AS |

Table des matières

main.cpp

```
#include "GestionFenetre.h"
//#include "Player.h"

#include <iostream>
#include <SFML/Graphics.hpp>
#include <SFML/Window.hpp>
#include <SFML/System.hpp>
#include <SFML/Audio.hpp>

using namespace std;

int main()
{
	GestionFenetre fenetre;

	while(fenetre.getWindow().isOpen())
	{
		fenetre.action();
	}
	return 0;
}
```

gestionFenetre.cpp

```
#include "GestionFenetre.h"

using namespace std;

GestionFenetre::GestionFenetre()
{
        window.create(sf::VideoMode(1920,1080),"la Bagarre",sf::Style::Fullscreen);
        window.setFramerateLimit(120);
        window.setMouseCursorVisible(0);

scene=Scene(window);
selecEcran=0;

menuPrinc= new MenuPrincipal(window.getSize().x,window.getSize().y);
menuSel= new MenuSelection(window);
menuCommandes= new MenuCommandes(window);
        menuBackground= new MenuBackground(window);

        if(!readyF.loadFromFile("background/SdHUDAtlas.png")) {
   std::cout<<"erreur background/SdHUDAtlas.png";
}

joueur1= new Player(1,window);
joueur2= new Player(2,window);

selecChamp_P1=-1;
selecChamp_P2=-1;

for(int i=0;i<=6;i++)
{
        _tabActionCombat.push_back(false);
}

readyFight.scale(2,2);

if (!musique.openFromFile("musique/theme_menu_princ.ogg")){
   std::cout<<"erreur musique";
}
```

```
   musique.setVolume(30.f) ;
   musique.play();
   musique.setLoop(true);
}

sf::RenderWindow& GestionFenetre::getWindow()
{

            return window;
}

void GestionFenetre::action()
{
            window.clear();
            sf::Event event;

            switch(selecEcran)        //menu principal
            {
            case -1:
                        window.close();
                        break;
   case 0:
     gestionMenuPrinc(event);
     break;
   case 1:
            gestionSelecPerso(event);
            break;
   case 2:
            combat(event);
            break;
   case 3:
            gestionMenuCommande(event);
            break;
```

```
    case 4:
             gestionSelecScene(event);
             break;
    }
}

void GestionFenetre::gestionMenuPrinc(sf::Event& event)
{
             while (window.pollEvent(event))
        menuPrinc->bouger(selecEcran,event,window);

    menuPrinc->draw(window);
    window.display();

    if(selecEcran==1)
    {
             if (!musique.openFromFile("musique/theme_menu_perso.ogg")){
           std::cout<<"erreur musique";
        }
        musique.play();
        musique.setLoop(true);

        menuSel->resetClock();
    }
}

void GestionFenetre::gestionSelecPerso(sf::Event& event)
{
             while (window.pollEvent(event))
    {
        if (event.type == sf::Event::Closed)
           window.close();
           menuSel->bouger_P1(event,window);
        menuSel->bouger_P2(event,window);
        selecEcran = menuSel->validationPerso(event,selecChamp_P1,selecChamp_P2);
    }

    window.clear();
    menuSel->draw(window);
    window.display();

    if(selecEcran==0)
    {
            if (!musique.openFromFile("musique/theme_menu_princ.ogg")){
        std::cout<<"erreur musique";
      }
      musique.play();
      musique.setVolume(30.f) ;
      musique.setLoop(true);

    }
}

void GestionFenetre::gestionMenuCommande(sf::Event& event)
{
            while (window.pollEvent(event))
      menuCommandes->retourMenu(selecEcran,event);
    menuCommandes->draw(window);
    window.display();
}

void GestionFenetre::gestionSelecScene(sf::Event& event)
{
            while (window.pollEvent(event))
    {
      menuBackground->retourMenu2(selecEcran,event, *menuSel,window);
    menuBackground->bouger(event, window);
    menuBackground->selectionner(event, window, selecEcran, scene,musique);
}

if(selecEcran==2)
{
    if(selecChamp_P1==0)
    {
        champion_P1=new Greg(-1,scene,window);
    }else if(selecChamp_P1==1)
    {
        champion_P1=new Dhalsim(-1,scene,window);
    }else if(selecChamp_P1==2)
    {
        champion_P1=new Ryu(-1,scene,window);
    }

    if(selecChamp_P2==0)
    {
        champion_P2= new Greg(1,scene,window);
    }else if(selecChamp_P2==1)
    {
        champion_P2=new Dhalsim(1,scene,window);
    }else if(selecChamp_P2==2)
    {
        champion_P2=new Ryu(1,scene,window);
    }

    if(selecChamp_P1!=-1 && selecChamp_P2!=-1)
    {
        joueur1->setChampion(champion_P1);
        joueur2->setChampion(champion_P2);
    }
    }



    menuBackground->draw(window);
    window.display();
}

void GestionFenetre::combat(sf::Event& event)
{
            /* Gestion de la fermeture de la fenetre */
            while (window.pollEvent(event))
    {
        if (event.type == sf::Event::Closed)
            window.close();

        joueur1->peutAttaquerP1(event,window);
        joueur2->peutAttaquerP2(event,window);
    }

            /* lancement des animations de début de combat */
            if(!_tabActionCombat[0] || !_tabActionCombat[1])
            {

                        readyFight.setTexture(readyF);
        readyFight.setTextureRect(sf::IntRect(666,435,300,74));
        readyFight.setPosition(sf::Vector2f(window.getSize().x/2-
readyFight.getGlobalBounds().width/2,scene.getBottom()*0.5));

                        if(!_tabActionCombat[0])
                                    _tabActionCombat[0]=joueur1->lancerApparition();
                        if(!_tabActionCombat[1])
                                    _tabActionCombat[1]=joueur2->lancerApparition();
        clockReadyFight.restart();
        }else if( (joueur1->getPV()<=0 || joueur2->getPV()<=0) && ( ( joueur1->getChampion()->auSol() && joueur2->getChampion()->auSol() ) || _tabActionCombat[4] ) )
  {
      readyFight.setTexture(readyF);
      readyFight.setTextureRect(sf::IntRect(377,384,287,128));
      readyFight.setPosition(sf::Vector2f(window.getSize().x/2-readyFight.getGlobalBounds().width/2,scene.getBottom()*0.5));

        _tabActionCombat[4]=true;
        if(_tabActionCombat[5]==false)
                _tabActionCombat[5]=joueur1->finPartie();
        if(_tabActionCombat[6]==false)
                _tabActionCombat[6]=joueur2->finPartie();

        if(_tabActionCombat[5]==true && _tabActionCombat[6]==true)
        {
        finCombat();
      }
  }else
        {
      readyFight.setTextureRect(sf::IntRect(0,401,373,107));
      readyFight.setPosition(sf::Vector2f(window.getSize().x/2-readyFight.getGlobalBounds().width/2,window.getSize().y*0.4));
      if(clockReadyFight.getElapsedTime().asSeconds() > 1) {
         readyFight.setTextureRect(sf::IntRect(0,2,0,0));

      }
                /* Récuperation des actions à effectuer */
                if(_tabActionCombat[2])
                {
                        joueur1->recuperationCommandesP1(*joueur2);
                }
                if(_tabActionCombat[3])
                {
                        joueur2->recuperationCommandesP2(*joueur1);
                }

                /* Lancement des animations Player 2*/
                _tabActionCombat[3]=joueur2->lancerActions(*joueur1);

                /* Lancement des animations Player 1*/
                _tabActionCombat[2]=joueur1->lancerActions(*joueur2);

        }

        /* Gestion de la fermeture de la fenetre */
        while (window.pollEvent(event))
  {
    if (event.type == sf::Event::Closed)
      window.close();
  }

  /* affichage des élements graphiques */
  affichageCombat();
}

void GestionFenetre::affichageCombat()
{
        window.draw(scene.getSprite());

  joueur1->afficherInfos(window);
        joueur2->afficherInfos(window);

  joueur1->affichageChampion(window);
```

```
    joueur2->affichageChampion(window);


    joueur1->afficherHitspark(window);
    joueur2->afficherHitspark(window);


    window.draw(readyFight);
    window.display();
}

void GestionFenetre::finCombat()
{
            for(int i=0;i<=6;i++)
    {
            _tabActionCombat[i]=false;
    }
    _tabActionCombat[2]=true;_tabActionCombat[3]=true;


            selecEcran=0;
            joueur1->getChampion()->resetHitbox();
            joueur2->getChampion()->resetHitbox();
            menuSel->reset(window);


            if (!musique.openFromFile("musique/theme_menu_princ.ogg")){
        std::cout<<"erreur musique";
    }
    musique.setVolume(50.f);
    musique.play();
    musique.setLoop(true);
}
```

menu.cpp

```
#include <iostream>

#include "menu.h"


using namespace std;


MenuPrincipal::MenuPrincipal(float width,float height){



   if(!menuFond.loadFromFile("background/menu.jpg")){

      std::cout<<"erreur fond"<<endl;

   }


   spriteFond.setTexture(menuFond);

   spriteFond.setScale(sf::Vector2f(width/1920.f,height/1080.f));


// Titre du jeu


   if (!font.loadFromFile("BebasNeue-Regular.ttf")){

      std::cout<<"erreur texte";

   }

   titre.setFont(font);

   titre.setString("La Bagarre");

   titre.setFillColor(sf::Color(220,60,60));

   titre.setStyle(sf::Text::Bold);

   titre.setCharacterSize(170);

   titre.setLetterSpacing (0.8);

   titre.setOutlineColor(sf::Color::Black);

   titre.setOutlineThickness (1.5);
```

```
    titre.setScale((width/1920)*2.5f,(height/1080)*1.f);
    titre.setPosition(sf::Vector2f(width/10,height/21.6));




// Tableau de sprites avec toutes les cases utilisables

    if(!textureCase.loadFromFile("sprites/spriteMenu.png")) {
        std::cout<<"erreur case";
    }
    textureCase.setSmooth(true);

    spriteMenu[0].setPosition(sf::Vector2f(width/2.74,height/3.08));
    spriteMenu[0].setTexture(textureCase);
    spriteMenu[0].setTextureRect(sf::IntRect(13, 12, 779, 180));
    spriteMenu[0].setScale((width/1920)*0.7f,(height/1080)*0.7f);

    spriteMenu[1].setPosition(sf::Vector2f(width/2.74,height/1.86));
    spriteMenu[1].setTexture(textureCase);
    spriteMenu[1].setTextureRect(sf::IntRect(10, 337, 779, 180));
    spriteMenu[1].setScale((width/1920)*0.7f,(height/1080)*0.7f);

    spriteMenu[2].setPosition(sf::Vector2f(width/2.74,height/1.33));
    spriteMenu[2].setTexture(textureCase);
    spriteMenu[2].setTextureRect(sf::IntRect(10,656, 779, 180));
    spriteMenu[2].setScale((width/1920)*0.7f,(height/1080)*0.7f);

    spriteMenu[3].setPosition(sf::Vector2f(width/2.9,height/3.10));
    spriteMenu[3].setTexture(textureCase);
    spriteMenu[3].setTextureRect(sf::IntRect(816, 12, 779, 180));
    spriteMenu[3].setScale((width/1920)*0.8f,(height/1080)*0.8f);
```

```
   spriteMenu[4].setPosition(sf::Vector2f(width/2.9,height/1.88));
   spriteMenu[4].setTexture(textureCase);
   spriteMenu[4].setTextureRect(sf::IntRect(816, 337, 779, 180));
   spriteMenu[4].setScale((width/1920)*0.8f,(height/1080)*0.8f);

   spriteMenu[5].setPosition(sf::Vector2f(width/2.9,height/1.35));
   spriteMenu[5].setTexture(textureCase);
   spriteMenu[5].setTextureRect(sf::IntRect(816,656, 779, 180));
   spriteMenu[5].setScale((width/1920)*0.8f,(height/1080)*0.8f);

// Les sprites des cases utilisées à l'ouverture du menu

   spriteMenux[0]=spriteMenu[3];

   spriteMenux[1]=spriteMenu[1];

   spriteMenux[2]=spriteMenu[2];

   _selection=0;
}

MenuPrincipal::~MenuPrincipal(){
}

void MenuPrincipal::draw(sf::RenderWindow &window){
     window.draw(spriteFond);
     window.draw(titre);
     window.draw(spriteMenux[0]);
     window.draw(spriteMenux[1]);
     window.draw(spriteMenux[2]);
```

```
  }

// Recupérer les intructions de l'utilisateur
void MenuPrincipal::bouger(int& selecEcran, sf::Event event,sf::RenderWindow& window)
{
  bool peutmonter = true, peutdescendre = true;
  if (sf::Joystick::isConnected(0))
  {
    sf::Time elapsed = clockAttenteJoystick.getElapsedTime();
    int timeAttente = elapsed.asMilliseconds();
    if(timeAttente>150)
    {
      joystick0_axisX = sf::Joystick::getAxisPosition(0, sf::Joystick::X);
      joystick0_axisY = sf::Joystick::getAxisPosition(0, sf::Joystick::Y);

      if( (joystick0_axisX>-80 && joystick0_axisX<80) && (joystick0_axisY<-30))
        moveUp();
      else if( (joystick0_axisX>-80 && joystick0_axisX<80) && (joystick0_axisY>30))
        moveDown();
      else if(sf::Joystick::isButtonPressed(0,0) && (_selection==0))
        selecEcran=1;
      else if(sf::Joystick::isButtonPressed(0, 0) && (_selection==1))
        selecEcran=3;
      else if(sf::Joystick::isButtonPressed(0, 0) && (_selection==2))
        selecEcran=-1;
      clockAttenteJoystick.restart();
    }
  }else
  {
    while (window.pollEvent(event))
    {
        switch ( event.type ){


        case sf::Event::Closed:
                window.close( );
                break;
        case sf::Event::KeyReleased:
            switch (event.key.code){
            case sf::Keyboard::Z:
              peutmonter = true;
              break;
            case sf::Keyboard::S:
                peutdescendre=true;
                break;}
        }
    }
    if(sf::Keyboard::isKeyPressed(sf::Keyboard::Up)){
        if(peutmonter){
            moveUp();
            peutmonter = false;
        }
    }
    if(sf::Keyboard::isKeyPressed(sf::Keyboard::Down)){
        if(peutdescendre){
            moveDown();
            peutdescendre = false;
        }
    }else if ((event.type==sf::Event::KeyReleased && event.key.code==sf::Keyboard::Enter) &&
(_selection==0))
    {
        selecEcran=1;
        if (!_effetSon.openFromFile("musique/menu_selec.ogg")){
                std::cout<<"erreur musique";
        }
        _effetSon.play();

    }else if ((event.type==sf::Event::KeyReleased && event.key.code==sf::Keyboard::Enter) &&
(_selection==1))
    {
        selecEcran=3;
        if (!_effetSon.openFromFile("musique/menu_selec.ogg")){
                std::cout<<"erreur musique";
        }
        _effetSon.play();

    }else if ((event.type==sf::Event::KeyReleased && event.key.code==sf::Keyboard::Enter) &&
(_selection==2))
    {
        if (!_effetSon.openFromFile("musique/menu_retour.ogg")){
                std::cout<<"erreur musique";
        }
        _effetSon.play();
        window.close();
    }
}

}

//Monter dans le menu
void MenuPrincipal::moveUp(){
if (_selection==1){
        spriteMenux[_selection]=spriteMenu[1];
        _selection=_selection -1;
        spriteMenux[_selection]=spriteMenu[3];
```

```
   }
   if (_selection==2){
         spriteMenux[_selection]=spriteMenu[2];
         _selection=_selection -1;
         spriteMenux[_selection]=spriteMenu[4];


   }
}

//Descendre dans le menu
void MenuPrincipal::moveDown()
{
   if (_selection==1){
         spriteMenux[_selection]=spriteMenu[1];
         _selection=_selection+1;
         spriteMenux[_selection]=spriteMenu[5];
   }
   if (_selection==0){
         spriteMenux[0]=spriteMenu[0];
         _selection=1;
         spriteMenux[1]=spriteMenu[4];


   }




}
```

```
MenuSelection::MenuSelection(sf::RenderWindow& window)
{
  if(!menuFond.loadFromFile("background/menu.jpg")){
    std::cout<<"erreur fond"<<endl;
  }

  spriteFond.setTexture(menuFond);
  spriteFond.setScale(sf::Vector2f(window.getSize().x/1920.f,window.getSize().y/1080.f));
  hauteurPerso=window.getSize().y*0.8;
  hauteurTexte=window.getSize().y*0.83;

  if (!textureVS.loadFromFile("background/VS.png"))
  {
    cout << "ERREUR : chargement d'image VS" << endl;
  }
  spriteVS.setTexture(textureVS);
  spriteVS.setPosition(sf::Vector2f(window.getSize().x*0.44,window.getSize().y*0.42));
  spriteVS.setTextureRect(sf::IntRect(0,0,324,277));
  //spriteVS.scale(window.getSize().x/1920.f,window.getSize().x/1080.f);

  if (!fontMenu.loadFromFile("MenuSelection/atari.ttf"))
  {
    cout << "ERREUR : chargement de police atari.ttf" << endl;
  }

  if (!texturePersos.loadFromFile("sprites/menuSelection.png"))
  {
    cout << "ERREUR : chargement d'image personnage : browli.png" << endl;
  }

  spriteP1.setTexture(texturePersos);
```

```
spriteP1.setPosition(sf::Vector2f(window.getSize().x*0.15, hauteurPerso-127*3.5));
spriteP1.setTextureRect(sf::IntRect(293,315,117,241));
spriteP1.setScale(sf::Vector2f(2,2));

spriteP2.setTexture(texturePersos);
spriteP2.setPosition(sf::Vector2f(window.getSize().x*0.85, hauteurPerso-220*1.8));
spriteP2.setTextureRect(sf::IntRect(205,19,141,220));
spriteP2.setScale(sf::Vector2f(-1.9,1.9));

if((choixJ1 == -1) || (choixJ2 == -1)) {
   //texte : sélection des personnages
   titre.setFont(fontMenu);
   titre.setString("Selection des personnages");
   titre.setCharacterSize(90);
   titre.setFillColor(sf::Color::Red);
   titre.setPosition(sf::Vector2f(window.getSize().x*0.3,window.getSize().y*0.05));
   titre.setScale(window.getSize().x/1920.f,window.getSize().x/1080.f);
}

if((choixJ1 == -1) || (choixJ2 == -1)) {

   //texte : Joueur 1
   j1.setFont(fontMenu);
   j1.setString("Joueur 1");
   j1.setCharacterSize(60);
   j1.setFillColor(sf::Color::White);
   j1.setStyle(sf::Text::Bold);
   j1.setPosition(sf::Vector2f(window.getSize().x*0.18,window.getSize().y*0.20));
   j1.setScale(window.getSize().x/1920.f,window.getSize().x/1080.f);

   //texte : Joueur 2
```

```
        j2.setFont(fontMenu);

        j2.setString("Joueur 2");

        j2.setCharacterSize(60);

        j2.setFillColor(sf::Color::White);

        j2.setStyle(sf::Text::Bold);

        j2.setPosition(sf::Vector2f(window.getSize().x*0.73,window.getSize().y*0.20));

        j2.setScale(window.getSize().x/1920.f,window.getSize().x/1080.f);



        //texte : nomPersoJ1

        nomPersoJ1.setFont(fontMenu);

        nomPersoJ1.setString("Greg");

        nomPersoJ1.setCharacterSize(40);

        nomPersoJ1.setFillColor(sf::Color::White);

        nomPersoJ1.setStyle(sf::Text::Bold);

        nomPersoJ1.setPosition(sf::Vector2f(window.getSize().x*0.20, hauteurTexte));

        nomPersoJ1.setScale(window.getSize().x/1920.f,window.getSize().x/1080.f);



        //texte : nomPersoJ2

        nomPersoJ2.setFont(fontMenu);

        nomPersoJ2.setString("Dhalsim");

        nomPersoJ2.setCharacterSize(40);

        nomPersoJ2.setFillColor(sf::Color::White);

        nomPersoJ2.setStyle(sf::Text::Bold);

        nomPersoJ2.setPosition(sf::Vector2f(window.getSize().x*0.76,hauteurTexte));

        nomPersoJ2.setScale(window.getSize().x/1920.f,window.getSize().x/1080.f);

    }

}


void MenuSelection::draw(sf::RenderWindow &window)
```

```
{
    window.draw(spriteFond);
    window.draw(titre);
    window.draw(j1);
    window.draw(j2);
    window.draw(spriteVS);
    window.draw(spriteP1);
    window.draw(spriteP2);
    window.draw(nomPersoJ1);
    window.draw(nomPersoJ2);
    window.draw(annulerChoixJ1);
    window.draw(annulerChoixJ2);

}

void MenuSelection::persoSuivant_P1(int& etatPerso,sf::RenderWindow& window)
{
  switch(etatPerso)
  {
  case 0:
    etatPersoJ1 = 1;
    spriteP1.setTextureRect(sf::IntRect(205,19,141,220));
    spriteP1.setScale(sf::Vector2f(1.9,1.9));
    spriteP1.setPosition(sf::Vector2f(window.getSize().x*0.15, hauteurPerso-220*1.8));
    break;
  case 1:
    etatPersoJ1 = 2;
    spriteP1.setTextureRect(sf::IntRect(367,25,123,245));
    spriteP1.setScale(sf::Vector2f(1.8,1.8));
    spriteP1.setPosition(sf::Vector2f(window.getSize().x*0.15, hauteurPerso-220*1.8));
    break;
    case 2:
        etatPersoJ1 = 0;
        spriteP1.setTextureRect(sf::IntRect(293,315,117,241));
        spriteP1.setScale(sf::Vector2f(1.9,1.9));
        spriteP1.setPosition(sf::Vector2f(window.getSize().x*0.15, hauteurPerso-127*3.5));
        break;
    }
}

void MenuSelection::persoSuivant_P2(int& etatPerso,sf::RenderWindow& window)
{
    switch(etatPerso)
    {
    case 0:
        etatPersoJ2 = 1;
        spriteP2.setTextureRect(sf::IntRect(205,19,141,220));
        spriteP2.setScale(sf::Vector2f(-1.9,1.9));
        spriteP2.setPosition(sf::Vector2f(window.getSize().x*0.85, hauteurPerso-220*1.8));
        break;
    case 1:
        etatPersoJ2 = 2;
        spriteP2.setTextureRect(sf::IntRect(367,25,123,245));
        spriteP2.setScale(sf::Vector2f(-1.8,1.8));
        spriteP2.setPosition(sf::Vector2f(window.getSize().x*0.85, hauteurPerso-220*1.8));
        break;
    case 2:
        etatPersoJ2 = 0;
        spriteP2.setTextureRect(sf::IntRect(293,315,117,241));
        spriteP2.setScale(sf::Vector2f(-1.9,1.9));
        spriteP2.setPosition(sf::Vector2f(window.getSize().x*0.85, hauteurPerso-127*3.5));
        break;
    }
}

void MenuSelection::persoPrecedent_P1(int& etatPerso,sf::RenderWindow& window)
{
    switch(etatPerso)
    {
    case 0:
        etatPersoJ1 = 2;
        spriteP1.setTextureRect(sf::IntRect(367,25,123,245));
        spriteP1.setScale(sf::Vector2f(1.8,1.8));
        spriteP1.setPosition(sf::Vector2f(window.getSize().x*0.15, hauteurPerso-220*1.8));
        break;
    case 1:
        etatPersoJ1 = 0;
        spriteP1.setTextureRect(sf::IntRect(293,315,117,241));
        spriteP1.setScale(sf::Vector2f(1.9,1.9));
        spriteP1.setPosition(sf::Vector2f(window.getSize().x*0.15, hauteurPerso-127*3.5));
        break;
    case 2:
        etatPersoJ1 = 1;
        spriteP1.setTextureRect(sf::IntRect(205,19,141,220));
        spriteP1.setScale(sf::Vector2f(1.8,1.8));
        spriteP1.setPosition(sf::Vector2f(window.getSize().x*0.15, hauteurPerso-220*1.8));
        break;
    }
}

void MenuSelection::persoPrecedent_P2(int& etatPerso,sf::RenderWindow& window)
{
    switch(etatPerso)
    {
    case 0:
        etatPersoJ2 = 2;
        spriteP2.setTextureRect(sf::IntRect(367,25,123,245));
        spriteP2.setScale(sf::Vector2f(-1.8,1.8));
        spriteP2.setPosition(sf::Vector2f(window.getSize().x*0.85, hauteurPerso-220*1.8));
        break;
    case 1:
        etatPersoJ2 = 0;
        spriteP2.setTextureRect(sf::IntRect(293,315,117,241));
        spriteP2.setScale(sf::Vector2f(-1.9,1.9));
        spriteP2.setPosition(sf::Vector2f(window.getSize().x*0.85, hauteurPerso-127*3.5));
        break;
    case 2:
        etatPersoJ2 = 1;
        spriteP2.setTextureRect(sf::IntRect(205,19,141,220));
        spriteP2.setScale(sf::Vector2f(-1.9,1.9));
        spriteP2.setPosition(sf::Vector2f(window.getSize().x*0.85, hauteurPerso-220*1.8));
        break;
    }
}

// Recupérer les intructions de l'utilisateur
void MenuSelection::bouger_P1(sf::Event event,sf::RenderWindow& window)
{
    //Selection j1
    if(choixJ1 == -1)
    {
        if (sf::Joystick::isConnected(0))
        {
            sf::Time elapsed = clockAttenteJoystick.getElapsedTime();
    int timeAttente = elapsed.asMilliseconds();
    if(timeAttente>150)
    {
      joystick0_axisX = sf::Joystick::getAxisPosition(0, sf::Joystick::X);
      joystick0_axisY = sf::Joystick::getAxisPosition(0, sf::Joystick::Y);

      if( (joystick0_axisX > 40) && (joystick0_axisY < 70 && joystick0_axisY > -55))
        persoSuivant_P1(etatPersoJ1,window);
      else if( (joystick0_axisX < -40) && (joystick0_axisY < 70 && joystick0_axisY > -40))
        persoPrecedent_P1(etatPersoJ1,window);

      clockAttenteJoystick.restart();
    }
  }else
  {
      bool peutGauche = true, peutDroite = true;

    while (window.pollEvent(event))
    {
      switch ( event.type )
      {
      case sf::Event::KeyReleased:
        switch (event.key.code)
        {
        case sf::Keyboard::Right:
         peutDroite = true;
         break;
        case sf::Keyboard::Left:
          peutGauche=true;
          break;
        }
        }
    }

    if(sf::Keyboard::isKeyPressed(sf::Keyboard::D))
    {
        if(peutDroite)
        {
            persoSuivant_P1(etatPersoJ1,window);
            peutDroite = false;
        }
    }

    if(sf::Keyboard::isKeyPressed(sf::Keyboard::Q))
    {
        if(peutGauche)
        {
            persoPrecedent_P1(etatPersoJ1,window);
            peutGauche = false;
        }
    }
}

switch(etatPersoJ1)
{
    case 0: nomPersoJ1.setString("Greg");
            break;
    case 1: nomPersoJ1.setString("Dhalsim");
            break;
    case 2: nomPersoJ1.setString("Ryu");
            break;
}
```

```
    }


    //Retour
    if( ( (sf::Event::KeyReleased && event.key.code == sf::Keyboard::B) ||
appuiBouttonManette(0,1,clockAttenteBoutton) ) && choixJ1 >= 0)
    {
        choixJ1 = -1;
        nomPersoJ1.setFillColor(sf::Color::White);
        annulerChoixJ1.setString("");
    }
}


void MenuSelection::bouger_P2(sf::Event event,sf::RenderWindow& window)
{
    //Selection j2
    if(choixJ2 == -1)
    {
        //if (sf::Joystick::isConnected(1))
        //{
            sf::Time elapsed = clockAttenteJoystick.getElapsedTime();
            int timeAttente = elapsed.asMilliseconds();
            if(timeAttente>142)
            {
                joystick0_axisX = sf::Joystick::getAxisPosition(1, sf::Joystick::X);
                joystick0_axisY = sf::Joystick::getAxisPosition(1, sf::Joystick::Y);


                if( (joystick0_axisX > 40) && (joystick0_axisY < 70 && joystick0_axisY > -55))
                    persoPrecedent_P2(etatPersoJ2,window);
                else if( (joystick0_axisX < -40) && (joystick0_axisY < 70 && joystick0_axisY > -40))
                    persoSuivant_P2(etatPersoJ2,window);
        clockAttenteJoystick.restart();
    }
//}else
//{
    bool peutGauche2 = true, peutDroite2 = true;

    while (window.pollEvent(event))
    {
        switch ( event.type )
        {
        case sf::Event::KeyReleased:
            switch (event.key.code)
            {
            case sf::Keyboard::D:
              peutDroite2 = true;
              break;
            case sf::Keyboard::Q:
                peutGauche2=true;
                break;
            }
        }
    }

    if(sf::Keyboard::isKeyPressed(sf::Keyboard::Right))
    {
        if(peutDroite2)
        {
            persoSuivant_P2(etatPersoJ2,window);
            peutDroite2 = false;
        }
    }
```

```
            if(sf::Keyboard::isKeyPressed(sf::Keyboard::Left))
            {
                if(peutGauche2)
                {
                    persoPrecedent_P2(etatPersoJ2,window);
                    peutGauche2 = false;
                }
            }
        //}

        switch(etatPersoJ2)
        {
            case 0: nomPersoJ2.setString("Greg");
                    break;
            case 1: nomPersoJ2.setString("Dhalsim");
                    break;
            case 2: nomPersoJ2.setString("Ryu");
                    break;
        }
    }

    //Retour
    if( ( (sf::Event::KeyReleased && event.key.code == sf::Keyboard::BackSpace) ||
appuiBouttonManette(1,1,clockAttenteBoutton) ) && choixJ2 >= 0)
    {
        choixJ2 = -1;
        nomPersoJ2.setFillColor(sf::Color::White);
        annulerChoixJ2.setString("");
    }
}
```

```
int MenuSelection::validationPerso(sf::Event event,int& selecChamp_P1, int& selecChamp_P2)

{

    //Validation du choix de personage pour Joueur 1


   if( (sf::Event::KeyReleased && event.key.code == sf::Keyboard::Space) ||
appuiBouttonManette(0,0,clockAttenteBoutton))

   {

       choixJ1 = etatPersoJ1;


       nomPersoJ1.setFillColor(sf::Color::Red);


       //texte : annulation du choix J1

       annulerChoixJ1.setFont(fontMenu);

       annulerChoixJ1.setString("Touche B pour annuler la selection");

       annulerChoixJ1.setCharacterSize(20);

       annulerChoixJ1.setFillColor(sf::Color::White);

       annulerChoixJ1.setStyle(sf::Text::Italic);

       annulerChoixJ1.setPosition(sf::Vector2f(650,hauteurTexte+15));


       if (!_effetSon.openFromFile("musique/perso_selec.ogg")){

             std::cout<<"erreur musique";

       }

       _effetSon.play();

   }


   //Validation du choix de personage pour Joueur 2

   if( (sf::Event::KeyReleased && event.key.code == sf::Keyboard::Enter) ||
appuiBouttonManette(1,0,clockAttenteBoutton))

   {

         choixJ2 = etatPersoJ2;
```

```
        nomPersoJ2.setFillColor(sf::Color::Red);

        //texte : annulation du choix J1
        annulerChoixJ2.setFont(fontMenu);
        annulerChoixJ2.setString("Touche BackSpace pour annuler la selection");
        annulerChoixJ2.setCharacterSize(20);
        annulerChoixJ2.setFillColor(sf::Color::White);
        annulerChoixJ2.setStyle(sf::Text::Italic);
        annulerChoixJ2.setPosition(sf::Vector2f(1000,hauteurTexte+15));

        if (!_effetSon.openFromFile("musique/perso_selec.ogg")){
            std::cout<<"erreur musique";
        }
        _effetSon.play();
    }

    if(choixJ1 != -1 && choixJ2 != -1)
    {
      selecChamp_P1=choixJ1;
      selecChamp_P2=choixJ2;
      return 4;
    }
    else if( ( (sf::Event::KeyReleased && event.key.code == sf::Keyboard::Escape) ||
appuiBouttonManette(0,3,clockAttenteBoutton) ) && choixJ1 == -1 && choixJ2 == -1)
    {
      if (!_effetSon.openFromFile("musique/menu_retour.ogg")){
          std::cout<<"erreur musique";
      }
      _effetSon.play();
      return 0;
    }else
        return 1;
}

void MenuSelection::reset(sf::RenderWindow& window)
{
    choixJ1=-1;choixJ2=-1;
    etatPersoJ1=0;etatPersoJ2=1;

    annulerChoixJ1.setString("");
    annulerChoixJ2.setString("");

    nomPersoJ1.setFillColor(sf::Color::White);
    nomPersoJ1.setString("Greg");
    nomPersoJ2.setFillColor(sf::Color::White);
    nomPersoJ2.setString("Dhalsim");

    spriteP1.setPosition(sf::Vector2f(window.getSize().x*0.15, hauteurPerso-127*3.5));
    spriteP1.setTextureRect(sf::IntRect(293,315,117,241));
    spriteP1.setScale(sf::Vector2f(1.9,1.9));

    spriteP2.setPosition(sf::Vector2f(window.getSize().x*0.85, hauteurPerso-220*1.8));
    spriteP2.setTextureRect(sf::IntRect(205,19,141,220));
    spriteP2.setScale(sf::Vector2f(-1.8,1.8));
}

void MenuSelection::initValidationPerso()
{
    choixJ2 = -1;
    nomPersoJ2.setFillColor(sf::Color::White);
    annulerChoixJ2.setString("");
```

```
}

void MenuSelection::resetClock()
{
   clockAttenteJoystick.restart();
   clockAttenteBoutton.restart();
}



MenuCommandes::MenuCommandes(sf::RenderWindow& window)
{

   if(!menuFond.loadFromFile("background/menu.jpg")){
      std::cout<<"erreur fond"<<endl;
   }

   spriteFond.setTexture(menuFond);
   spriteFond.setScale(sf::Vector2f(window.getSize().x/1920.f,window.getSize().y/1080.f));

   if (!fontCommandes.loadFromFile("MenuSelection/atari.ttf"))
   {
      cout << "ERREUR : chargement de police atari.ttf" << endl;
   }
   if(!texturej1.loadFromFile("sprites/commandej1.png")) {
      std::cout<<"erreur manette";
   }
   if(!texturej2.loadFromFile("sprites/commandej2.png")) {
      std::cout<<"erreur clavier";
   }
   j1.setFont(fontCommandes);
   j1.setString("Joueur 1");
```

```
j1.setCharacterSize(50);
j1.setFillColor(sf::Color(255,0,0));
j1.setPosition(sf::Vector2f(100,100));
j1.setScale(sf::Vector2f(window.getSize().x/1920.f,window.getSize().y/1080.f));


j2.setFont(fontCommandes);
j2.setString("Joueur 2");
j2.setCharacterSize(50);
j2.setFillColor(sf::Color(255,0,0));
j2.setPosition(sf::Vector2f(1000,100));
j2.setScale(sf::Vector2f(window.getSize().x/1920.f,window.getSize().y/1080.f));


spriteCommandes[0].setPosition(sf::Vector2f(window.getSize().x*0.1, window.getSize().y*0.2));
spriteCommandes[0].setTexture(texturej1);
spriteCommandes[0].setTextureRect(sf::IntRect(0, 0,  679, 415));
spriteCommandes[0].setScale(sf::Vector2f(window.getSize().x/1920.f,window.getSize().y/1080.f));

spriteCommandes[1].setPosition(sf::Vector2f(window.getSize().x*0.55, window.getSize().y*0.2));
spriteCommandes[1].setTexture(texturej2);
spriteCommandes[1].setTextureRect(sf::IntRect(0, 0,515, 515));
spriteCommandes[1].setScale(sf::Vector2f(window.getSize().x/1920.f,window.getSize().y/1080.f));

retour.setFont(fontCommandes);
retour.setString("Appuyez sur echap pour revenir au menu");
retour.setCharacterSize(30);
retour.setFillColor(sf::Color::White);
retour.setStyle(sf::Text::Italic);
retour.setPosition(sf::Vector2f(window.getSize().x*0.40, window.getSize().y*0.9));
retour.setScale(sf::Vector2f(window.getSize().x/1920.f,window.getSize().y/1080.f));
```

```
    //ligne delim

    ligneDelim.setSize(sf::Vector2f(window.getSize().x*0.005,window.getSize().y*0.6));

    ligneDelim.setPosition(sf::Vector2f(window.getSize().x/2, window.getSize().y/5));

    ligneDelim.setFillColor(sf::Color::Black);

}


void MenuCommandes::retourMenu(int& selecEcran,sf::Event event)

{

    if( (sf::Event::KeyReleased && event.key.code == sf::Keyboard::Escape) ||
appuiBouttonManette(0,3,clockAttenteBoutton) )

    {

        selecEcran=0;

    }

}


void MenuCommandes::draw(sf::RenderWindow &window)

{

    window.draw(spriteFond);

    window.draw(spriteCommandes[0]);

    window.draw(spriteCommandes[1]);

    window.draw(retour);

    window.draw(j1);

    window.draw(j2);

    window.draw(ligneDelim);

}


MenuBackground::MenuBackground(sf::RenderWindow& window)

{


    selection=0;
```

```
if(!menuFond.loadFromFile("background/menu.jpg")){
   std::cout<<"erreur fond"<<endl;
}

spriteFond.setTexture(menuFond);
spriteFond.setScale(sf::Vector2f(window.getSize().x/1920.f,window.getSize().y/1080.f));

if (!fontBackground.loadFromFile("MenuSelection/atari.ttf"))
{
   cout << "ERREUR : chargement de police atari.ttf" << endl;
}
if(!bg[0].loadFromFile("background/toit.png")) {
   std::cout<<"erreur fond toit";
}
if(!bg[1].loadFromFile("background/futur.jpg")) {
   std::cout<<"erreur fond futur";
}
if(!bg[2].loadFromFile("background/ring_xenoverse_V2.jpg")) {
   std::cout<<"erreur fond xenorverse";
}
if(!bg[3].loadFromFile("background/skulls.jpg")) {
   std::cout<<"erreur fond skulls";
}
if(!bg[4].loadFromFile("background/SanFran.png")) {
   std::cout<<"erreur fond brazil";
}
if(!bg[5].loadFromFile("background/avion.png")) {
   std::cout<<"erreur fond brazil";
}
```

```
spritebg[0].setTexture(bg[0]);

spritebg[0].setPosition(sf::Vector2f(window.getSize().x*0.025,window.getSize().y*0.25));

spritebg[0].setScale(0.3f,0.3f);

spritebg[0].scale(window.getSize().x/1920.f,window.getSize().y/1080.f);


spritebg[1].setTexture(bg[1]);

spritebg[1].setPosition(sf::Vector2f(window.getSize().x*0.35,window.getSize().y*0.25));

spritebg[1].setScale(0.3f,0.3f);

spritebg[1].scale(window.getSize().x/1920.f,window.getSize().y/1080.f);


spritebg[2].setTexture(bg[2]);

spritebg[2].setPosition(sf::Vector2f(window.getSize().x*0.675,window.getSize().y*0.25));

spritebg[2].setScale(0.3f,0.3f);

spritebg[2].scale(window.getSize().x/1920.f,window.getSize().y/1080.f);


spritebg[3].setTexture(bg[3]);

spritebg[3].setPosition(sf::Vector2f(window.getSize().x*0.025,window.getSize().y*0.61));

spritebg[3].setScale(0.3f,0.3f);

spritebg[3].scale(window.getSize().x/1920.f,window.getSize().y/1080.f);


spritebg[4].setTexture(bg[4]);

spritebg[4].setPosition(sf::Vector2f(window.getSize().x*0.35,window.getSize().y*0.61));

spritebg[4].setScale(0.3f,0.3f);

spritebg[4].scale(window.getSize().x/1920.f,window.getSize().y/1080.f);


spritebg[5].setTexture(bg[5]);

spritebg[5].setPosition(sf::Vector2f(window.getSize().x*0.675,window.getSize().y*0.61));

spritebg[5].setScale(0.3f,0.3f);

spritebg[5].scale(window.getSize().x/1920.f,window.getSize().y/1080.f);
```

```
    titre.setFont(fontBackground);
    titre.setString("Choix de la Map");
    titre.setCharacterSize(90);
    titre.setFillColor(sf::Color::Red);
    titre.setPosition(sf::Vector2f(window.getSize().x*0.38,window.getSize().y*0.05));
    titre.setScale(window.getSize().x/1920.f,window.getSize().y/1080.f);

    retour.setFont(fontBackground);
    retour.setString("Appuyez sur echap pour revenir au menu");
    retour.setCharacterSize(30);
    retour.setFillColor(sf::Color::White);
    retour.setStyle(sf::Text::Italic);
    retour.setPosition(sf::Vector2f(window.getSize().x*0.40, window.getSize().y*0.95));
    retour.setScale(sf::Vector2f(window.getSize().x/1920.f,window.getSize().y/1080.f));

    for(int i=0; i<6;i++){
        rect[i].setSize(sf::Vector2f(window.getSize().x*0.32,window.getSize().y*0.32));
        rect[i].setFillColor(sf::Color(255,0,0));

    }
    rect[0].setPosition(sf::Vector2f(window.getSize().x*0.015,window.getSize().y*0.24));
    rect[1].setPosition(sf::Vector2f(window.getSize().x*0.34,window.getSize().y*0.24));
    rect[2].setPosition(sf::Vector2f(window.getSize().x*0.665,window.getSize().y*0.24));
    rect[3].setPosition(sf::Vector2f(window.getSize().x*0.015,window.getSize().y*0.6));
    rect[4].setPosition(sf::Vector2f(window.getSize().x*0.34,window.getSize().y*0.6));
    rect[5].setPosition(sf::Vector2f(window.getSize().x*0.665,window.getSize().y*0.6));

}

void MenuBackground::draw(sf::RenderWindow& window)
```

```
{
        window.draw(spriteFond);
        window.draw(rect[selection]);
        window.draw(spritebg[0]);
        window.draw(spritebg[1]);
        window.draw(spritebg[2]);
        window.draw(spritebg[3]);
        window.draw(spritebg[4]);
        window.draw(spritebg[5]);
        window.draw(retour);
        window.draw(titre);
    }

void MenuBackground::retourMenu2(int& selecEcran,sf::Event event,MenuSelection&
m,sf::RenderWindow& window)
{
    if( (sf::Event::KeyReleased && event.key.code == sf::Keyboard::Escape) ||
appuiBouttonManette(0,3,clockAttenteBoutton) )
    {
        selecEcran=1;
        m.initValidationPerso();
    }
}

void MenuBackground::bouger(sf::Event event, sf::RenderWindow& window)
{
    bool peutGauche = true, peutDroite = true;

    while (window.pollEvent(event))
    {
        switch ( event.type )
        {
```

```
    case sf::Event::KeyReleased:
        switch (event.key.code)
        {
        case sf::Keyboard::Right:
          peutDroite = true;
          break;
        case sf::Keyboard::Left:
            peutGauche=true;
            break;
        }
    }
}

if(sf::Joystick::isConnected(0))
{
    sf::Time elapsed = clockAttenteJoystick.getElapsedTime();
    int timeAttente = elapsed.asMilliseconds();
    if(timeAttente>150)
    {
        joystick0_axisX = sf::Joystick::getAxisPosition(0, sf::Joystick::X);
        joystick0_axisY = sf::Joystick::getAxisPosition(0, sf::Joystick::Y);

        if( (joystick0_axisX > 40) && (joystick0_axisY < 70 && joystick0_axisY > -55))
            moveRight();
        else if( (joystick0_axisX < -40) && (joystick0_axisY < 70 && joystick0_axisY > -40))
            moveLeft();

        clockAttenteJoystick.restart();
    }

}else if(sf::Keyboard::isKeyPressed(sf::Keyboard::Right))
```

```
    {
        if(peutDroite)
        {
            moveRight();
            peutDroite = false;
        }
    }else if(sf::Keyboard::isKeyPressed(sf::Keyboard::Left))
    {
        if(peutGauche)
        {
            moveLeft();
            peutGauche = false;
        }
    }
    if ((selection>=0) && (selection<3) && (sf::Keyboard::isKeyPressed(sf::Keyboard::Down)))
        selection=selection+3;

    if ((selection>=3) && (selection<6) && (sf::Keyboard::isKeyPressed(sf::Keyboard::Up)))
        selection=selection-3;
}

void MenuBackground::moveRight()
{
    if (selection<5)
    {
        selection=selection+1;
    }
}

void MenuBackground::moveLeft(){
    if (selection>0)
    {
        selection=selection-1;
    }

}
void MenuBackground::selectionner(sf::Event event, sf::RenderWindow& window, int& selecEcran,
Scene& s,sf::Music& son)
{
    bool go = true;
    while (window.pollEvent(event))
    {
        if(sf::Event::KeyReleased && event.key.code==sf::Keyboard::Enter)
        {
            go=true;
            break;
        }
    }

    if(sf::Joystick::isConnected(0))
    {
        if(appuiBouttonManette(0,0,clockAttenteBoutton))
            valider(window,selecEcran,s,son);
    }else if(sf::Keyboard::isKeyPressed(sf::Keyboard::Enter))
    {
        if(go)
        {
            valider(window, selecEcran, s,son);
            go = false;
        }
    }
```

```
}


void MenuBackground::valider(sf::RenderWindow& window, int& selecEcran, Scene& s,sf::Music&
son)
{
    if (selection==0)
        s.chargementToit(son);
    if (selection==1)
        s.chargementFutur(son);
    if (selection==2)
        s.chargementXenoverse(son);
    if (selection==3)
        s.chargementSkulls(son);
    if (selection==4)
        s.chargementSanFran(window,son);
    if (selection==5)
        s.chargementAvion(son);
    selecEcran=2;
}



bool appuiBouttonManette(int numJoy,int numBoutton,sf::Clock& clockAttente)
{
  sf::Time elapsed = clockAttente.getElapsedTime();
  int timeAttente = elapsed.asMilliseconds();
  if(timeAttente>150)
  {
    if (sf::Joystick::isButtonPressed(numJoy,numBoutton))
    {
      clockAttente.restart();
      return true;
```

```
        }
    }
    return false;
}
```

player.cpp

```
#include "IncludeManager.h"
#include "Player.h"
#include <string>
#include <vector>

using namespace std;

Player::Player(int n,sf::RenderWindow& window)
{
	double largeurFenetre=window.getSize().x;
	ratioScale=largeurFenetre/1920;

	_PV=100;
	_energie=0;

	_posHorizontale=0;
	_posVerticale=0;
	_action=-1;
	_actionFini=true;

	for(int i=0;i<=11;i++)
		_tabActions.push_back(false);
	for(int i=0;i<=4;i++)
		_tabPeutAction.push_back(true);
```

```
if (!_textureBI.loadFromFile("background/lifeBar_V2.png"))
{
   std::cout<<"Erreur au chargement du sprite";
}
_barreInfos.setTexture(_textureBI);

for(int i=0;i<2;i++)
{
            sf::RectangleShape temp;
            _barrePV.push_back(temp);
            _barreEnergie.push_back(temp);
}

_barrePV[0].setSize(sf::Vector2f(_PV*7.15*ratioScale,38*ratioScale));
_barreEnergie[0].setSize(sf::Vector2f(100*2.76*ratioScale,40*ratioScale));

_barrePV[0].setFillColor(sf::Color(90,37,37));
_barreEnergie[0].setFillColor(sf::Color(210,254,254));

_barreEnergie[1].setFillColor(sf::Color(10,255,255));
_barreEnergie[1].setScale(ratioScale,ratioScale);

if(n==1)
{
            _barreInfos.setScale(ratioScale,ratioScale);
            _barreInfos.setPosition(sf::Vector2f(0,10*ratioScale));

            for(int i=0;i<2;i++)
            {
                        _barrePV[i].setPosition(sf::Vector2f(177*ratioScale,19*ratioScale));
                        _barreEnergie[i].setPosition(sf::Vector2f(177*ratioScale,63*ratioScale));
                }
        }else
        {
                _barreInfos.setScale(-1*ratioScale,1*ratioScale);
                _barreInfos.setPosition(sf::Vector2f(window.getSize().x,10*ratioScale));

                for(int i=0;i<2;i++)
                {
                        _barrePV[i].scale(-1,1);
                        _barrePV[i].setPosition(sf::Vector2f(window.getSize().x-
177*ratioScale,18*ratioScale));
                        _barreEnergie[i].scale(-1,1);
                        _barreEnergie[i].setPosition(sf::Vector2f(window.getSize().x-
177*ratioScale,63*ratioScale));
                }

        }
}

void Player::resetPlayer()
{
        resetAttributs();
        _prendCoup=0;
        _energie=0;
        _PV=100;
}

void Player::resetAttributs()
{
        _posHorizontale=0;
        _posVerticale=0;
```

```
	_action=-1;
	_actionFini=true;

	for(int i=0;i<=11;i++)
		_tabActions[i]=false;
}

void Player::setChampion(Personnage* perso)
{
	_champion=perso;
	resetPlayer();

	if(_barreInfos.getScale().x>=0)
	{
		_portrait=_champion->getIcone();
		_portrait.setPosition(0,10);
	}else
	{

		_portrait=_champion->getIcone();
		_portrait.setPosition(_barreInfos.getPosition().x,10);
		_portrait.scale(-1,1);
	}
}

Personnage* Player::getChampion()
{
	return _champion;
}

void Player::recuperationAttaqueLancee()
```

```
{
            for(int i=7;i<_tabActions.size();i++)
            {
                        if(_tabActions[i]==true)
                                    _tabPeutAction[i-7]=false;
            }
}

void Player::peutAttaquerP1(sf::Event& event, sf::RenderWindow& window)
{
   if (sf::Joystick::isConnected(0))   // Commandes pour manette
            {
      if(event.type==sf::Event::JoystickButtonReleased && event.joystickButton.joystickId==0 )
      {
         switch (event.joystickButton.button)
         {
         case 0:
           _tabPeutAction[2] = true;
           break;
         case 1:
           _tabPeutAction[2] = true;
           break;
         case 2:
            _tabPeutAction[0] = true;
            _tabPeutAction[4] = true;
            break;
         case 3:
           _tabPeutAction[0] = true;
           _tabPeutAction[4] = true;
           break;
         case 4:
          _tabPeutAction[1] = true;
          break;
        case 5:
          _tabPeutAction[1] = true;
          break;
        }

        if(event.joystickMove.axis==sf::Joystick::Z || event.joystickMove.axis==sf::Joystick::R)
        {
          if(event.joystickMove.position<10)
                  _tabPeutAction[3] = true;
        }

      }
    }else
    {
          if(event.type==sf::Event::KeyReleased )
      {
        switch (event.key.code)
        {
        case sf::Keyboard::A :
          _tabPeutAction[0] = true;
          break;
        case sf::Keyboard::E :
           _tabPeutAction[2]=true;
           break;
        case sf::Keyboard::R :
          _tabPeutAction[4] = true;
          break;
        }
      }
```

```
   }
}

void Player::recuperationCommandesP1(Player& ennemi)    // Commandes pour le player 1
{
        resetAttributs();

        if (sf::Joystick::isConnected(0))   // Commandes pour manette
        {
                /* gestion des attaques */
                _tabActions[9] =( (sf::Joystick::isButtonPressed(0, 0) ||
sf::Joystick::isButtonPressed(0, 1) ) && _tabPeutAction[2]);
                _tabActions[7] =( (sf::Joystick::isButtonPressed(0, 2) ||
sf::Joystick::isButtonPressed(0, 3) ) && _tabPeutAction[0]);   // touche pour mettre un coup de poing

                joystick0_axisZ =sf::Joystick::getAxisPosition(0, sf::Joystick::Z);   // touche pour super
kick
                joystick0_axisR =sf::Joystick::getAxisPosition(0, sf::Joystick::R);   // touche pour super
kick
                _tabActions[10] =( (joystick0_axisZ > 40 || joystick0_axisR>40)  &&
_tabPeutAction[3]);

                _tabActions[8]=( (sf::Joystick::isButtonPressed(0, 4) || sf::Joystick::isButtonPressed(0,
5))  && _tabPeutAction[1]);

                if( ( (joystick0_axisX > 40 || joystick0_axisX < -40) && joystick0_axisY > 40) &&
(sf::Joystick::isButtonPressed(0, 2) || sf::Joystick::isButtonPressed(0, 3)) && _tabPeutAction[4] )
                        _tabActions[11] =true;

                /* gestion des deplacements */
                joystick0_axisX = sf::Joystick::getAxisPosition(0, sf::Joystick::X);
                joystick0_axisY = sf::Joystick::getAxisPosition(0, sf::Joystick::Y);
                //cout<<"x : "<<joystick0_axisX<<"\t y : "<<joystick0_axisY<<endl;
                if( (joystick0_axisX > 40) && (joystick0_axisY < 70 && joystick0_axisY > -55) )
                        _tabActions[0] =true;
                else if( (joystick0_axisX < -40) && (joystick0_axisY < 70 && joystick0_axisY > -40) )
                        _tabActions[1] =true;
                else if( (joystick0_axisX>-80 && joystick0_axisX<80) && (joystick0_axisY<-40) )
                        _tabActions[3] =true;
                else if( (joystick0_axisX<=100 && joystick0_axisX>=80) && (joystick0_axisY<-40) )
                {
                        _tabActions[0] =true;
                        _tabActions[3] =true;
                }
                else if( (joystick0_axisX>=-100 && joystick0_axisX<=-80) && (joystick0_axisY<-40) )
                {
                        _tabActions[1] =true;
                        _tabActions[3] =true;
                }
                else if( (joystick0_axisX>-80 && joystick0_axisX<80) && (joystick0_axisY>40) )
                        _tabActions[2] =true;


        }else   // Commandes clavier au cas ou manette absent
        {
                _tabActions[1] =sf::Keyboard::isKeyPressed(sf::Keyboard::Q);    // touche pour
reculer:    Q
                _tabActions[0] =sf::Keyboard::isKeyPressed(sf::Keyboard::D);    // touche pour
avancer:    D
                _tabActions[3] =sf::Keyboard::isKeyPressed(sf::Keyboard::Z);                 // touche
pour sauter:     Z
                _tabActions[2] =sf::Keyboard::isKeyPressed(sf::Keyboard::S);     // touche pour
accroupir:  S
                _tabActions[7] =( sf::Keyboard::isKeyPressed(sf::Keyboard::A) && _tabPeutAction[0]);
                // touche pour puncher:    A
                _tabActions[9] =( sf::Keyboard::isKeyPressed(sf::Keyboard::E) && _tabPeutAction[2]);
                // touche pour kicker:     E
                _tabActions[11]=( sf::Keyboard::isKeyPressed(sf::Keyboard::R) &&
_tabPeutAction[4]);                   // touche pour spécial 1:  R
```

```
            }


            recuperationAttaqueLancee();
            gestionDesCommandes(ennemi);
}



void Player::peutAttaquerP2(sf::Event& event, sf::RenderWindow& window)
{
   if (sf::Joystick::isConnected(0))   // Commandes pour manette
            {
       if(event.type==sf::Event::JoystickButtonReleased && event.joystickButton.joystickId==1 )
       {
          switch (event.joystickButton.button)
          {
          case 0:
            _tabPeutAction[2] = true;
            break;
          case 1:
            _tabPeutAction[2] = true;
            break;
          case 2:
             _tabPeutAction[0] = true;
             _tabPeutAction[4] = true;
             break;
          case 3:
            _tabPeutAction[0] = true;
            _tabPeutAction[4] = true;
            break;
          case 4:
            _tabPeutAction[1] = true;
            break;
          case 5:
            _tabPeutAction[1] = true;
            break;
          }

          if(event.joystickMove.axis==sf::Joystick::Z || event.joystickMove.axis==sf::Joystick::R)
          {
            if(event.joystickMove.position<10)
                      _tabPeutAction[3] = true;
          }

       }
   }else
   {
            if(event.type==sf::Event::KeyReleased )
       {
          switch (event.key.code)
          {
          case sf::Keyboard::P :
            _tabPeutAction[0] = true;
            break;
          case sf::Keyboard::M :
             _tabPeutAction[2]=true;
             break;
          case sf::Keyboard::L :
            _tabPeutAction[4] = true;
            break;
          }
       }
   }
```

```
}


void Player::recuperationCommandesP2(Player& ennemi)    // Commandes pour le player 2

{

            resetAttributs();


            if (sf::Joystick::isConnected(1))   // Commandes pour manette

            {

                        /* gestion des attaques */

                        _tabActions[9] =( (sf::Joystick::isButtonPressed(1, 0) ||
sf::Joystick::isButtonPressed(1, 1)) && _tabPeutAction[2]);

                        _tabActions[7] =( (sf::Joystick::isButtonPressed(1, 2) ||
sf::Joystick::isButtonPressed(1, 3)) && _tabPeutAction[0]);   // touche pour mettre un coup de poing


                        joystick1_axisZ =sf::Joystick::getAxisPosition(1, sf::Joystick::Z);   // touche pour super
punch

                        joystick1_axisR =sf::Joystick::getAxisPosition(1, sf::Joystick::R);   // touche pour super
kick

                        _tabActions[10] =( (joystick1_axisZ > 40 || joystick1_axisR>40) &&
_tabPeutAction[3]);


                        _tabActions[8]=( (sf::Joystick::isButtonPressed(1, 4) || sf::Joystick::isButtonPressed(1,
5)) && _tabPeutAction[1]);


                        if( ( (joystick1_axisX > 40 || joystick1_axisX < -40) && joystick1_axisY > 40) &&
(sf::Joystick::isButtonPressed(1, 2) || sf::Joystick::isButtonPressed(1, 3)) && _tabPeutAction[4])

                                    _tabActions[11] =true;


                        /* gestion des deplacements */

                        joystick1_axisX = sf::Joystick::getAxisPosition(1, sf::Joystick::X);

                        joystick1_axisY = sf::Joystick::getAxisPosition(1, sf::Joystick::Y);

                        //cout<<"x : "<<joystick0_axisX<<"\t y : "<<joystick0_axisY<<endl;

                        if( (joystick1_axisX > 40) && (joystick1_axisY < 70 && joystick1_axisY > -55) )

                                    _tabActions[0] =true;
```

```
                else if( (joystick1_axisX < -40) && (joystick1_axisY < 70 && joystick1_axisY > -40) )
                        _tabActions[1] =true;
                else if( (joystick1_axisX>-80 && joystick1_axisX<80) && (joystick1_axisY<-40) )
                        _tabActions[3] =true;
                else if( (joystick1_axisX<=100 && joystick1_axisX>=80) && (joystick1_axisY<-40) )
                {
                        _tabActions[0] =true;
                        _tabActions[3] =true;
                }
                else if( (joystick1_axisX>=-100 && joystick1_axisX<=-80) && (joystick1_axisY<-40) )
                {
                        _tabActions[1] =true;
                        _tabActions[3] =true;
                }
                else if( (joystick1_axisX>-80 && joystick1_axisX<80) && (joystick1_axisY>40) )
                        _tabActions[2] =true;


        }else   // Commandes clavier au cas ou manette absent
        {
                _tabActions[0]=sf::Keyboard::isKeyPressed(sf::Keyboard::Right);         // touche
pour _tabActions[1]:    Right
                _tabActions[1]=sf::Keyboard::isKeyPressed(sf::Keyboard::Left);          // touche
pour _tabActions[0]:    Left
                _tabActions[3]=sf::Keyboard::isKeyPressed(sf::Keyboard::Up);                    //
touche pour sauter:     Up
                _tabActions[2]=sf::Keyboard::isKeyPressed(sf::Keyboard::Down);          // touche
pour accroupir:  Down
                _tabActions[7]=sf::Keyboard::isKeyPressed(sf::Keyboard::P)&& _tabPeutAction[0];
                        // touche pour puncher:    P
                _tabActions[9]=sf::Keyboard::isKeyPressed(sf::Keyboard::M) && _tabPeutAction[2];
                        // touche pour kicker:     M
                _tabActions[11]=sf::Keyboard::isKeyPressed(sf::Keyboard::L) && _tabPeutAction[4];
                        // touche pour spécial 1:  L
```

```
	}

	recuperationAttaqueLancee();
	gestionDesCommandes(ennemi);
}



void Player::gestionDesCommandes(Player& ennemi)
{
	if(_tabActions[7] && (_tabActions[0] || _tabActions[1]))
	{
		_tabActions[7]=true;
		_tabActions[0]=false;
		_tabActions[1]=false;
	}else if(_tabActions[8] && (_tabActions[0] || _tabActions[1]))
	{
		_tabActions[9]=true;
		_tabActions[0]=false;
		_tabActions[1]=false;
	}else if(_tabActions[9] && (_tabActions[0] || _tabActions[1]))
	{
		_tabActions[9]=true;
		_tabActions[0]=false;
		_tabActions[1]=false;
	}else if(_tabActions[10] && (_tabActions[0] || _tabActions[1]))
	{
		_tabActions[10]=true;
		_tabActions[0]=false;
		_tabActions[1]=false;
	}else if(_tabActions[11] && (_tabActions[0] || _tabActions[1] || _tabActions[2] ||
_tabActions[3]))
```

```
{
        _tabActions[11]=true;
        _tabActions[0]=false;
        _tabActions[1]=false;
        _tabActions[2]=false;
        _tabActions[3]=false;
}

if(_tabActions[0] && _tabActions[3])
{
        _posHorizontale=1;
        _posVerticale=1;
}else if (_tabActions[1] && _tabActions[3])
{
        _posHorizontale=-1;
        _posVerticale=1;
}



/* Gestion des attributs */

if(_tabActions[0])
        _posHorizontale=1;
else if(_tabActions[1])
        _posHorizontale=-1;
else
        _posHorizontale=0;



if(_tabActions[3])
```

```
		_posVerticale=1;
	else if(_tabActions[2])
		_posVerticale=-1;
	else
		_posVerticale=0;

	if(ennemi.getAction()>0 && _champion->getOrientation()==_posHorizontale)
		_action=0;
	else if(_tabActions[11])
		_action=5;
	else if(_tabActions[7])
		_action=1;
	else if(_tabActions[8])
		_action=2;
	else if(_tabActions[9])
		_action=3;
	else if(_tabActions[10])
		_action=4;
	else
		_action=-1;
}

bool Player::lancerApparition()
{
	return _champion->apparition(_effet);
}

bool Player::lancerActions(Player& jEnnemi)
{
	if(_prendCoup!=0)
	{
```

```
            if(_action==0)
                _actionFini=_champion->parade(&_prendCoup,_effet);


            else
            {
                if(_prendCoup>0)
                    setDegats(_prendCoup);
                _posHorizontale==0;_posVerticale==0;_action=-1;
                _actionFini=_champion->prendCoup(&_prendCoup,_effet,_energie);
            }
        }


        else if(_action==0)
        {
            if( (_champion->getOrientation()==-1 && jEnnemi.getChampion()->getPosX() <
    _champion->getPosX() + _champion->getSprite().getGlobalBounds().width) || (_champion-
    >getOrientation()==1 && jEnnemi.getChampion()->getPosX() > _champion->getPosX()-_champion-
    >getSprite().getGlobalBounds().width) )
                _champion->garde();
            else
                _champion->reculer();
        }


        else if(_posHorizontale==1 && _posVerticale==1)
        {
            if(_champion->getOrientation()==-1)
                _actionFini=_champion->sauterAvant(*jEnnemi.getChampion());
            else
                _actionFini=_champion->sauterArriere(*jEnnemi.getChampion());
        }


        else if(_posHorizontale==-1 && _posVerticale==1)
```

```
	{
		if(_champion->getOrientation()==-1)
			_actionFini=_champion->sauterArriere(*jEnnemi.getChampion());
		else
			_actionFini=_champion->sauterAvant(*jEnnemi.getChampion());
	}

	else if(_posHorizontale==1)
	{
		if(_champion->getOrientation()==-1)
			_champion->avancer(*jEnnemi.getChampion());
		else
			_champion->reculer();
	}

	else if(_posHorizontale==-1)
	{
		if(_champion->getOrientation()==-1)
			_champion->reculer();
		else
			_champion->avancer(*jEnnemi.getChampion());
	}

	else if(_posVerticale==1)
		_actionFini=_champion-
>sauter(_action,*jEnnemi.getChampion(),jEnnemi.getPrendCoup(),_energie);

	else if(_posVerticale==-1)
		_champion->accroupi(_action==0);

	else if(_action==1)
```

```
                        _actionFini=_champion-
>punch(*jEnnemi.getChampion(),jEnnemi.getPrendCoup(),_energie);

            else if(_action==2)
                        _actionFini=_champion-
>punchSP(_effet,*jEnnemi.getChampion(),jEnnemi.getPrendCoup(),_energie);

            else if(_action==3)
                        _actionFini=_champion-
>kick(*jEnnemi.getChampion(),jEnnemi.getPrendCoup(),_energie);

            else if(_action==4)
                        _actionFini=_champion-
>kickSP(*jEnnemi.getChampion(),jEnnemi.getPrendCoup(),_energie);

            else if(_action==5)
                        _actionFini=_champion-
>SP(_effet,*jEnnemi.getChampion(),jEnnemi.getPrendCoup(),_energie);

            else
                        _champion->statique(*jEnnemi.getChampion());

            if(_posVerticale!=-1)
                        _champion->resetCptAccroupi();



            return _actionFini;
}

bool Player::finPartie()
{
            _posVerticale=0;_posHorizontale=0;_action=-1;
            _effet.setTextureRect(sf::IntRect(0,0,0,0));
```

```
	if(_PV>0)
		return _champion->victoire();
	else
		return _champion->mort();
}

int Player::getAction()
{
	return _action;
}

int Player::getPV()
{
	return _PV;
}

void Player::setDegats(int degats)
{
	_PV-=degats;
}

void Player::afficherInfos(sf::RenderWindow& window)
{
	/* gestion de la barre de points de vie */

	if(_PV>66)
		_barrePV[1].setFillColor(sf::Color(0,250,0));
	else if(_PV>33)
		_barrePV[1].setFillColor(sf::Color(255,165,0));
	else
		_barrePV[1].setFillColor(sf::Color(255,0,0));
	if(_PV<0)
		_PV=0;
	_barrePV[1].setSize(sf::Vector2f(_PV*7.15*ratioScale,38*ratioScale));

	/* gestion de la barre d'energie */

	sf::Time elapsed = _clockPasAssez.getElapsedTime();
int timePA = elapsed.asMilliseconds();

if(timePA>200)
		_barreEnergie[0].setFillColor(sf::Color(210,254,254));

	if(_energie>=0 && _energie<=100)
		_sauvegardeEnergie=_energie;

	if(_energie==-100)
	{
		_energie=_sauvegardeEnergie;
		_barreEnergie[0].setFillColor(sf::Color(255,30,30));
		_clockPasAssez.restart();
	}

	else if(_energie<0 && _energie!=-100)
		_energie=0;
	else if(_energie>100)
		_energie=100;

	_barreEnergie[1].setSize(sf::Vector2f(_energie*2.76*ratioScale,44*ratioScale));

	/* affichage des barres */
```

```
        for(int i=0;i<2;i++)
        {
                window.draw(_barrePV[i]);
                window.draw(_barreEnergie[i]);
        }

        window.draw(_barreInfos);
        window.draw(_portrait);
}


void Player::afficherHitspark(sf::RenderWindow& window){
  _champion->affichageEffet(window);
}


void Player::affichageChampion(sf::RenderWindow& window)
{
  window.draw(_champion->getSprite());
  window.draw(_effet);
  //window.draw(_champion->getHurtbox());
  //window.draw(_champion->getHitbox());
        //window.draw(_champion->getGardebox());
}

int* Player::getPrendCoup()
{
        return &_prendCoup;
}
```

scene.cpp

```
#include "IncludeManager.h"


using namespace std;


Scene::Scene(sf::RenderWindow& w)
{
	_tailleWindow=w.getSize();
}


void Scene::chargementXenoverse(sf::Music& sonScene)
{
	if(!_textureScene.loadFromFile("background/ring_xenoverse_V2.jpg")){cout<<"Erreur
chargement de Scene"<<endl;}
	else{
		_textureScene.setSmooth(true);
		_spriteScene.setTexture(_textureScene);
		_spriteScene.setScale(_tailleWindow.x/1920.f,_tailleWindow.y/1080.f);
	}


	_hauteurSol=75.f*(_tailleWindow.y/1920);
	_limiteSol=_tailleWindow.y-_hauteurSol;
	_largeurWindow=_tailleWindow.x;


	_solScene.setSize(sf::Vector2f(1920.f, _hauteurSol));
	_solScene.setPosition(0.f, _limiteSol);
	_solScene.setFillColor(sf::Color(250,250,250,0));
	_solScene.setOutlineThickness(2.f);
	_solScene.setOutlineColor(sf::Color(250, 130, 1));
```

```
            _wallLeft.setSize(sf::Vector2f(5.f, _tailleWindow.y));
            _wallLeft.setPosition(0.f, 0.f);
            _wallLeft.setFillColor(sf::Color(50,250,60,1));


            _wallRight.setSize(sf::Vector2f(5.f, _tailleWindow.y));
            _wallRight.setPosition(_tailleWindow.x-5, 0.f);
            _wallRight.setFillColor(sf::Color(50,250,60,1));


            if (!sonScene.openFromFile("musique/World_tournament_arena_stage.ogg")){
          std::cout<<"erreur musique";
   }
   sonScene.play();
   sonScene.setVolume(40.f) ;
   sonScene.setLoop(true);
}


void Scene::chargementFutur(sf::Music& sonScene)
{
            if(!_textureScene.loadFromFile("background/futur.jpg")){cout<<"Erreur chargement de
Scene"<<endl;}
            else{
                        _textureScene.setSmooth(true);
                        _spriteScene.setTexture(_textureScene);
       _spriteScene.setScale(_tailleWindow.x/1920.f,_tailleWindow.y/1080.f);
            }


            double temp=_tailleWindow.x;


            _hauteurSol=75.f*(temp/1920);
            _limiteSol=_tailleWindow.y-_hauteurSol;
            _largeurWindow=_tailleWindow.x;
            _solScene.setSize(sf::Vector2f(1920.f, _hauteurSol));

            _solScene.setPosition(0.f, _limiteSol);

            _solScene.setFillColor(sf::Color(250,250,250,0));

            _solScene.setOutlineThickness(2.f);

            _solScene.setOutlineColor(sf::Color(250, 130, 1));


            _wallLeft.setSize(sf::Vector2f(5.f, _tailleWindow.y));

            _wallLeft.setPosition(0.f, 0.f);

            _wallLeft.setFillColor(sf::Color(50,250,60,1));


            _wallRight.setSize(sf::Vector2f(5.f, _tailleWindow.y));

            _wallRight.setPosition(_tailleWindow.x-5, 0.f);

            _wallRight.setFillColor(sf::Color(50,250,60,1));


            if (!sonScene.openFromFile("musique/theme_future.ogg")){

          std::cout<<"erreur musique";

    }

    sonScene.play();

    sonScene.setVolume(40.f) ;

    sonScene.setLoop(true);

}

void Scene::chargementToit(sf::Music& sonScene)

{

            if(!_textureScene.loadFromFile("background/toit.png")){cout<<"Erreur chargement de
Scene"<<endl;}

            else{

                        _textureScene.setSmooth(true);

                        _spriteScene.setTexture(_textureScene);

      _spriteScene.setScale(_tailleWindow.x/1920.f,_tailleWindow.y/1080.f);

            }
            double temp=_tailleWindow.x;

            _hauteurSol=200.f*(temp/1920);
            _limiteSol=_tailleWindow.y-_hauteurSol;
            _largeurWindow=_tailleWindow.x;

            _solScene.setSize(sf::Vector2f(1920.f, _hauteurSol));
            _solScene.setPosition(0.f, _limiteSol);
            _solScene.setFillColor(sf::Color(250,250,250,0));
            _solScene.setOutlineThickness(2.f);
            _solScene.setOutlineColor(sf::Color(250, 130, 1));

            _wallLeft.setSize(sf::Vector2f(5.f, _tailleWindow.y));
            _wallLeft.setPosition(0.f, 0.f);
            _wallLeft.setFillColor(sf::Color(50,250,60,1));

            _wallRight.setSize(sf::Vector2f(5.f, _tailleWindow.y));
            _wallRight.setPosition(_tailleWindow.x-5, 0.f);
            _wallRight.setFillColor(sf::Color(50,250,60,1));

            if (!sonScene.openFromFile("musique/theme_japon.ogg")){
        std::cout<<"erreur musique";
    }
    sonScene.play();
    sonScene.setVolume(40.f) ;
    sonScene.setLoop(true);
}
void Scene::chargementSanFran(sf::RenderWindow& window,sf::Music& sonScene)
{
```

```
            if(!_textureScene.loadFromFile("background/SanFran.png")){cout<<"Erreur chargement de
Scene"<<endl;}

            else{

                        _textureScene.setSmooth(true);

                        _spriteScene.setTexture(_textureScene);

       _spriteScene.setScale(_tailleWindow.x/1920.f,_tailleWindow.y/1080.f);

            }


            double temp=_tailleWindow.x;


            _hauteurSol=70.f*(temp/1920);

            _limiteSol=_tailleWindow.y-_hauteurSol;

            _largeurWindow=_tailleWindow.x;


            _solScene.setSize(sf::Vector2f(1920.f, _hauteurSol));

            _solScene.setPosition(0.f, _limiteSol);

            _solScene.setFillColor(sf::Color(250,250,250,0));

            _solScene.setOutlineThickness(2.f);

            _solScene.setOutlineColor(sf::Color(250, 130, 1));


            _wallLeft.setSize(sf::Vector2f(5.f, _tailleWindow.y));

            _wallLeft.setPosition(0.f, 0.f);

            _wallLeft.setFillColor(sf::Color(50,250,60,1));


            _wallRight.setSize(sf::Vector2f(5.f, _tailleWindow.y));

            _wallRight.setPosition(_tailleWindow.x-5, 0.f);

            _wallRight.setFillColor(sf::Color(50,250,60,1));


            if (!sonScene.openFromFile("musique/theme_brazil.ogg")){

          std::cout<<"erreur musique";

   }
   sonScene.play();

   sonScene.setVolume(40.f) ;

   sonScene.setLoop(true);

}

void Scene::chargementSkulls(sf::Music& sonScene)

{

            if(!_textureScene.loadFromFile("background/skulls.jpg")){cout<<"Erreur chargement de
Scene"<<endl;}

            else{

                        _textureScene.setSmooth(true);

                        _spriteScene.setTexture(_textureScene);

      _spriteScene.setScale(_tailleWindow.x/1920.f,_tailleWindow.y/1080.f);

            }


            double temp=_tailleWindow.x;


            _hauteurSol=75.f*(temp/1920);

            _limiteSol=_tailleWindow.y-_hauteurSol;

            _largeurWindow=_tailleWindow.x;


            _solScene.setSize(sf::Vector2f(1920.f, _hauteurSol));

            _solScene.setPosition(0.f, _limiteSol);

            _solScene.setFillColor(sf::Color(250,250,250,0));

            _solScene.setOutlineThickness(2.f);

            _solScene.setOutlineColor(sf::Color(250, 130, 1));


            _wallLeft.setSize(sf::Vector2f(5.f, _tailleWindow.y));

            _wallLeft.setPosition(0.f, 0.f);

            _wallLeft.setFillColor(sf::Color(50,250,60,1));


            _wallRight.setSize(sf::Vector2f(5.f, _tailleWindow.y));
            _wallRight.setPosition(_tailleWindow.x-5, 0.f);
            _wallRight.setFillColor(sf::Color(50,250,60,1));

            if (!sonScene.openFromFile("musique/theme_skulls.ogg")){
          std::cout<<"erreur musique";
   }
   sonScene.play();
   sonScene.setVolume(40.f) ;
   sonScene.setLoop(true);
}

void Scene::chargementAvion(sf::Music& sonScene)
{
            if(!_textureScene.loadFromFile("background/avion.png")){cout<<"Erreur chargement de
Scene"<<endl;}
            else{
                        _textureScene.setSmooth(true);
                        _spriteScene.setTexture(_textureScene);
       _spriteScene.setScale(_tailleWindow.x/1920.f,_tailleWindow.y/1080.f);
            }

            double temp=_tailleWindow.x;

            _hauteurSol=50.f*(temp/1920);
            _limiteSol=_tailleWindow.y-_hauteurSol;
            _largeurWindow=_tailleWindow.x;

            _solScene.setSize(sf::Vector2f(1920.f, _hauteurSol));
            _solScene.setPosition(0.f, _limiteSol);
            _solScene.setFillColor(sf::Color(250,250,250,0));
            _solScene.setOutlineThickness(2.f);
            _solScene.setOutlineColor(sf::Color(250, 130, 1));


            _wallLeft.setSize(sf::Vector2f(5.f, _tailleWindow.y));
            _wallLeft.setPosition(0.f, 0.f);
            _wallLeft.setFillColor(sf::Color(50,250,60,1));


            _wallRight.setSize(sf::Vector2f(5.f, _tailleWindow.y));
            _wallRight.setPosition(_tailleWindow.x-5, 0.f);
            _wallRight.setFillColor(sf::Color(50,250,60,1));


            if (!sonScene.openFromFile("musique/theme_skulls.ogg")){
          std::cout<<"erreur musique";
   }
   sonScene.play();
   sonScene.setVolume(40.f) ;
   sonScene.setLoop(true);
}
sf::Sprite Scene::getSprite() const
{
            return _spriteScene;
}



int Scene::getBottom() const
{
            return _limiteSol;
}


int Scene::getLeftLimit() const
{
            return _wallLeft.getSize().x;
```

```
}

int Scene::getRightLimit() const
{
	return _largeurWindow-_wallRight.getSize().x;
}

sf::RectangleShape Scene::getSol() const
{
	return _solScene;
}
```

personnage.cpp

```
#include "../IncludeManager.h"

using namespace std;

Personnage::Personnage(){
}

void Personnage::setScene(const Scene& s){
  _scene=s;



	if(_orientation==1)
		_posX=100.f;
	else
		_posX=_scene.getRightLimit()-100.f;

	_posY=_scene.getBottom()-_tailleSprite.y;
```

```
            _sprite.setPosition(_posX,_posY);

            keepInWalls();



    if(_orientation==-1)

    {

                    _hurtbox.setScale(-1,1);

                    _hitbox.setScale(-1,1);

            }

            _cptAnimEffet = 0;

            _hitSpark = false;

            _peutHitSpark = true;

            _effetEnCours = false;


    if(!_textureEffet.loadFromFile("sprites/hitsparks.png")){

        std::cout<<"Erreur au chargement du sprite";

            }

            _spriteHitSpark.setTexture(_textureEffet);

            _spriteHitSpark.setScale(2,2);

}


sf::Sprite Personnage::getSprite()

{

            return _sprite;

}



void Personnage::setSprite(int n1, int n2, int i1, int i2)

{

            _tailleSprite.x=i1*_scale;_tailleSprite.y=i2*_scale;

            _sprite.setTextureRect(sf::IntRect(n1, n2,i1,i2));
```

```
}

sf::Sprite Personnage::getIcone()
{
   return _icone;
}

sf::RectangleShape Personnage::getHurtbox()
{
            return _hurtbox;
}

sf::RectangleShape Personnage::getHitbox()
{
            return _hitbox;
}

void Personnage::resetHitbox()
{
   _hitbox.setSize(sf::Vector2f(0,0));
}

sf::RectangleShape Personnage::getGardebox()
{
   return _gardebox;
}

bool Personnage::collisionCoup(Personnage& ennemi)
{
   return _hitbox.getGlobalBounds().intersects(ennemi.getHurtbox().getGlobalBounds());
}
```

```
void Personnage::collision(Personnage& ennemi, int& deplacement)
{
   if( (_orientation==1 && _posX+_tailleSprite.x+deplacement*2 >= ennemi.getPosX()-
ennemi.getHurtbox().getGlobalBounds().width)
           || (_orientation==-1 && _posX-_tailleSprite.x-deplacement*2 <=
ennemi.getPosX()+ennemi.getHurtbox().getGlobalBounds().width))
   {
      deplacement=0;
   }

   _sprite.setPosition(_posX,_posY);
}

void Personnage::keepInWalls()
{
        if(_orientation==-1)
        {
                if(_posX-_tailleSprite.x<_scene.getLeftLimit())
                        _posX=_scene.getLeftLimit()+_tailleSprite.x;
                else if(_posX>_scene.getRightLimit())
                        _posX=_scene.getRightLimit();
        }else if(_orientation==1)
        {
                if(_posX<_scene.getLeftLimit())
                        _posX=_scene.getLeftLimit();
                else if(_posX+_tailleSprite.x>_scene.getRightLimit())
                        _posX=_scene.getRightLimit()-_tailleSprite.x;
        }
        if(_posY+_tailleSprite.y>_scene.getBottom())
                        _posY=_scene.getBottom()-_tailleSprite.y;
        _sprite.setPosition(sf::Vector2f(_posX,_posY ));
}



void Personnage::rotate(Personnage& ennemi)

{

   if( (_orientation==1 && _hurtbox.getPosition().x > ennemi.getHurtbox().getPosition().x) ||
(_orientation==-1 && _hurtbox.getPosition().x< ennemi.getHurtbox().getPosition().x) )

           {

      //cout<<"_orientation :\t"<<_orientation<<endl<<"moi.x
:\t"<<_hurtbox.getPosition().x<<endl<<"lui.x :\t"<<ennemi.getHurtbox().getPosition().x<<endl;

      _orientation=_orientation*-1;

      if(_orientation==-1)

      {

         _hurtbox.setScale(-1,1);

         _hitbox.setScale(-1,1);

      }else

      {

         _hurtbox.setScale(1,1);

         _hitbox.setScale(1,1);

      }

                      _posX=_posX-_tailleSprite.x*_orientation;

                      _sprite.setPosition(_posX,_posY);

                      _sprite.setScale(_orientation*_scale,_scale);

           }

}


bool Personnage::auSol()

{

   return(_posY+_tailleSprite.y>=_scene.getBottom()-5);

}


int Personnage::getOrientation() const
```

```
{
        return _orientation*-1;
}

void Personnage::setPosX(int n)
{
  _posX=n;
}

int Personnage::getPosX()
{
  return _posX;
}

void Personnage::setPosY(int n)
{
  _posY=n;
}

int Personnage::getPosY()
{
  return _posY;
}

void Personnage::resetCptAccroupi()
{
        _cptAccroupi=0;
}

void Personnage::collisionsaut(Personnage& ennemi,int& deplacement)
{
  if(_hurtbox.getGlobalBounds().intersects(ennemi.getHurtbox().getGlobalBounds()))
  {
      float positionGauche = _hurtbox.getGlobalBounds().left;
      float positionDroite = _hurtbox.getGlobalBounds().left + _hurtbox.getGlobalBounds().width;
      float positionBasse = _hurtbox.getPosition().y + _hurtbox.getGlobalBounds().height;
      float positionHaute = _hurtbox.getPosition().y;


      float positionGaucheEnnemi = ennemi.getHurtbox().getGlobalBounds().left;
      float positionDroiteEnnemi = ennemi.getHurtbox().getGlobalBounds().width +
ennemi.getHurtbox().getGlobalBounds().left;
      float positionHauteEnnemi = ennemi.getHurtbox().getPosition().y;
      float positionBasseEnnemi = ennemi.getHurtbox().getPosition().y +
ennemi.getHurtbox().getGlobalBounds().height;


      /*
          _scene.getBottom() est la position du sol
          le perso verifie si l'autre perso est au sol ou en saut, le comportement est different en saut et
au sol
          normalement les personnages ne peuvent pas sortir de la fenetre, lorsqu'ils se croisent en saut
leurs vitesses = 0,
          un personnage qui saute et qui retombe sur un autre perso ira a sa droite ou à sa gauche en
fonction de sa position en x
          par rapport à l'autre personnage



      */
      //on verifie que le perso cible est le seul à être en saut
      if(positionBasse < positionBasseEnnemi && positionBasseEnnemi >= _scene.getBottom())
      {
          if(_orientation == -1)
          {
              //on ne fait rien si il n'y a pas collision
```

```
            if(!(positionGauche + deplacement > positionDroiteEnnemi && positionBasse >
positionHauteEnnemi)){
                //on verifie si le perso cible dépasse l'autre si oui on le deplace à droite, sinon à gauche
                if((positionDroite + deplacement >= positionDroiteEnnemi && positionBasse >
positionHauteEnnemi && positionDroiteEnnemi < _scene.getRightLimit() -
ennemi.getHurtbox().getGlobalBounds().width/2)
                    || positionGauche < 0 && positionGaucheEnnemi  <
_hurtbox.getGlobalBounds().width/2 && positionBasse > positionHauteEnnemi)
                {
                    _posX+=(positionDroiteEnnemi - positionGauche);
                    deplacement = 0;
                }
                else if(positionDroite + deplacement > positionGaucheEnnemi && positionBasse >
positionHauteEnnemi)
                {
                    _posX+=(-positionDroite + positionGaucheEnnemi - deplacement*2);
                    deplacement = 0;
                }
            }
        }
        else if(_orientation == 1)
        {
            //même chose mais avec une orientation différente
            if(!(positionDroite + deplacement < positionGaucheEnnemi && positionBasse >
positionHauteEnnemi)){
                if((positionGauche + deplacement <= positionGaucheEnnemi && positionBasse >
positionHauteEnnemi && positionGaucheEnnemi > ennemi.getHurtbox().getGlobalBounds().width/2)
                    || positionDroite > _scene.getRightLimit() && positionDroiteEnnemi >
_scene.getRightLimit() - _hurtbox.getGlobalBounds().width/2 && positionBasse >
positionHauteEnnemi)
                {
                    _posX+=(-positionDroite+positionGaucheEnnemi - deplacement);
                }
                else if(positionGauche + deplacement < positionDroiteEnnemi && positionBasse >
positionHauteEnnemi)
```

```
                {
                    _posX+=(positionDroiteEnnemi - positionGauche - deplacement*2);
                }
            }
        }
    }
    //si les deux personnages sont en saut on les empêche de se confondre et on stoppe leur vitesse
    else if(positionBasse < _scene.getBottom() && positionBasseEnnemi < _scene.getBottom())
    {
        if(_orientation == -1)
        {
            if(positionDroite + deplacement >= positionGaucheEnnemi)
            {
                _posX+=(-deplacement);
                deplacement = 0;
            }
        }
        else if(_orientation == 1)
        {
            if(positionGauche + deplacement <= positionDroiteEnnemi)
            {
                _posX+=(-deplacement);
                deplacement = 0;
            }
        }
    }
    _sprite.setPosition(_posX,_posY);
    keepInWalls();
  }
}
```

```
void Personnage::affichageEffet(sf::RenderWindow& window){
sf::Time elapsed = _clockEffet.getElapsedTime();
int timeAnim = elapsed.asMilliseconds();
int decalageX;
  if(_hitSpark && _peutHitSpark){
    _peutHitSpark = false;
    _effetEnCours = true;
    _clockEffet.restart();
  }
  if(_effetEnCours){
    _hitSpark = false;
    if(_orientation == -1){
      _spriteHitSpark.setScale(-1,1);
    }
    else{
      _spriteHitSpark.setScale(1,1);
    }
    switch(_cptAnimEffet){
      case 0:
        _spriteHitSpark.setTextureRect(sf::IntRect(1,1,142,220));
        decalageX = 0;
        break;
      case 1:
        _spriteHitSpark.setTextureRect(sf::IntRect(147,1,145,220));
        decalageX = -10*_orientation;
        break;
      case 2:
        _spriteHitSpark.setTextureRect(sf::IntRect(296,1,196,220));
        decalageX = 20*_orientation;
        break;
```

```
          case 3:
              _spriteHitSpark.setTextureRect(sf::IntRect(496,1,196,220));
              decalageX = +20*_orientation;
              break;
          case 4:
              _spriteHitSpark.setTextureRect(sf::IntRect(696,1,196,220));
              decalageX = +20*_orientation;
              break;
          case 5:
              _spriteHitSpark.setTextureRect(sf::IntRect(896,1,184,220));
              decalageX = +60*_orientation;
              break;
        }
        if(timeAnim > 40){
          _cptAnimEffet +=1;
          _clockEffet.restart();

_spriteHitSpark.setPosition(sf::Vector2f(_spriteHitSpark.getPosition().x+decalageX,_spriteHitSpark.ge
tPosition().y));
        }
        if(_cptAnimEffet > 5){
          _cptAnimEffet = 0;
          _effetEnCours = false;
          _peutHitSpark = true;
        }
        window.draw(_spriteHitSpark);
    }
}
```

dhalsim.cpp

```
#include "../IncludeManager.h"


using namespace std;


Dhalsim::Dhalsim(int orientation,Scene& s,sf::RenderWindow& window)
{
    double largeurFenetre=window.getSize().x;
    _scale=4*(largeurFenetre/1920);


    _orientation=-orientation;



_cptStatic=0;_cptAvancer=0;_cptReculer=0;_cptSauter=0;_cptApparition=0;_cptAction=0;_cptAccrou
pi=0;_cptPrendCoup=0;
    _vsaut = -40;


    if (!_texture.loadFromFile("sprites/sprite_dhalsim.png"))
    {
        std::cout<<"Erreur au chargement du sprite";
    }
    _sprite.setTexture(_texture);
    _sprite.scale(_orientation*_scale,_scale);


    _icone.setTexture(_texture);
    _icone.setTextureRect(sf::IntRect(990,6490,97,104));
    _icone.scale(largeurFenetre/1920,largeurFenetre/1920);


    _hurtbox.setFillColor(sf::Color(255,255,255,0));
    _hurtbox.setOutlineColor(sf::Color::Green);
    _hurtbox.setOutlineThickness(4);
    _hitbox.setFillColor(sf::Color(255,255,255,0));
    _hitbox.setOutlineColor(sf::Color::Red);
    _hitbox.setOutlineThickness(4);

    _gardebox.setFillColor(sf::Color(255,255,255,0));
    _gardebox.setOutlineColor(sf::Color::Blue);
    _gardebox.setOutlineThickness(4);

    _spriteHitSpark.setColor(sf::Color(130,255,130,255));

    setScene(s);
}



bool Dhalsim::victoire()
{
    sf::Time elapsed = _clockAnim.getElapsedTime();
    int timeAnim = elapsed.asMilliseconds();
    int delai=70;
    bool fini=false;
    _hitbox.setSize(sf::Vector2f(0,0));

    if(timeAnim>delai)
    {
        switch (_cptApparition)
        {
        case 0:
            _cptApparition ++;
            _clockAnim.restart();
            setSprite(24,5634,50,126);

            _hurtbox.setSize(sf::Vector2f(0,0));

            if (!_effetSonore.openFromFile("musique/Dhalsim/victoire.ogg"))
                std::cout<<"erreur musique";
            _effetSonore.play();
            break;
        case 1:
            _cptApparition ++;
            _clockAnim.restart();
            setSprite(82,5634,50,126);
            break;
        case 2:
            _cptApparition ++;
            _clockAnim.restart();
            setSprite(140,5634,56,126);
            break;
        case 3:
            _cptApparition ++;
            _clockAnim.restart();
            setSprite(204,5634,60,126);
            break;
        case 4:
            _cptApparition ++;
            _clockAnim.restart();
            setSprite(272,5634,65,126);
            break;
        case 5:
            _cptApparition ++;
            _clockAnim.restart();
```

```
            setSprite(345,5634,63,126);
            break;
        case 6:
            _cptApparition++;
            _clockAnim.restart();
            setSprite(417,5634,64,126);
            break;
        }

        _posY=_scene.getBottom()-_tailleSprite.y;
        _sprite.setPosition(_posX,_posY);
    }

    if(_cptApparition==7 && timeAnim>2000)
    {
        _clockAnim.restart();
        _cptApparition=0;
        fini=true;
    }

    keepInWalls();
    return fini;
}

bool Dhalsim::mort()
{
    sf::Time elapsed = _clockAnim.getElapsedTime();
    int timeAnim = elapsed.asMilliseconds();
    int delai=100,deplacementX=_scene.getRightLimit()/15;
    bool fini=false;
    _hitbox.setSize(sf::Vector2f(0,0));
```

```
if(timeAnim>delai)
{
    switch (_cptApparition)
    {
    case 0:
        _cptApparition ++;
        _clockAnim.restart();
        setSprite(102,5112,90,111);
        _hurtbox.setSize(sf::Vector2f(0,0));
        _posX-=deplacementX*_orientation;

        if (!_effetSonore.openFromFile("musique/Dhalsim/mort.ogg"))
            std::cout<<"erreur musique";
        _effetSonore.play();
        break;
    case 1:
        _cptApparition ++;
        _clockAnim.restart();
        setSprite(200,5112,93,111);
        _posX-=deplacementX*_orientation;
        break;
    case 2:
        _cptApparition++;
        _clockAnim.restart();
        setSprite(301,5112,130,111);
        _posX-=deplacementX*_orientation;
        break;
    case 3:
        _cptApparition++;
        _clockAnim.restart();
```

```
            setSprite(439,5112,127,111);
            break;
        case 4:
            _cptApparition++;
            _clockAnim.restart();
            setSprite(300,5328,141,39);
            break;

        }

        if(_cptApparition >=3)
            _posY=_scene.getBottom()-_tailleSprite.y;
        _sprite.setPosition(_posX,_posY);

    }

    if(_cptApparition==5 && timeAnim>2000)
    {
        _clockAnim.restart();
        _cptApparition=0;
        fini=true;
    }



    keepInWalls();
    return fini;
}

bool Dhalsim::parade(int* degats,sf::Sprite& effet)
{
    bool fini=false;
```

```
_cptSauter=0;
_cptAction=0;
effet.setTextureRect(sf::IntRect(0,0,0,0));
_hurtbox.setSize(sf::Vector2f(0,0));

sf::Time elapsed = _clockAnim.getElapsedTime();
int timeAnim = elapsed.asMilliseconds();
int delai=120;

if(_cptPrendCoup==0)
{
   setSprite(125,4747,63,100);
   _cptPrendCoup++;
}
else if(timeAnim > delai)
{
   if(_cptPrendCoup==1)
   {
      _clockAnim.restart();
      _cptPrendCoup++;
   }
   else
   {
      _clockAnim.restart();
      _cptPrendCoup=0;
      fini=true;
      *degats=0;
   }
}

sf::Time elapsedDep = _clockMove.getElapsedTime();
```

```
    int timeDep = elapsedDep.asMilliseconds();
    int delaiDep=20,deplacement=_scene.getRightLimit()/200*_orientation;

    if(timeDep>delaiDep)
    {
        _clockMove.restart();
        _posX-=deplacement;
        _sprite.setPosition(_posX,_posY);
    }

    keepInWalls();
    return fini;
}

bool Dhalsim::prendCoup(int* degats,sf::Sprite& effet,int& energie)
{
    *degats=-1;
    bool fini=false;
    _cptSauter=0;
    _cptAction=0;
    _vsaut=-40;
    effet.setTextureRect(sf::IntRect(0,0,0,0));
    sf::Time elapsed = _clockAnim.getElapsedTime();
    int timeAnim = elapsed.asMilliseconds();
    int delai=70;
    _hurtbox.setSize(sf::Vector2f(0,0));
    _gardebox.setSize(sf::Vector2f(0,0));

    if(timeAnim > delai)
    {
        switch(_cptPrendCoup)
```

```
{
case 0:
   _clockAnim.restart();
   _cptPrendCoup++;
   setSprite(24,4996,85,100);
   _posX-=10*_scale*_orientation;

   energie+=5;

   if (!_effetSonore.openFromFile("musique/Dhalsim/degat.ogg"))
      std::cout<<"erreur musique";
   _effetSonore.play();
   break;
case 1:
   _clockAnim.restart();
   _cptPrendCoup++;
   setSprite(117,4996,90,100);
   _posX-=10*_scale*_orientation;
   break;
case 2:
   _clockAnim.restart();
   _cptPrendCoup++;
   setSprite(215,4996,93,100);
   _posX-=10*_scale*_orientation;
   break;
case 3:
   _clockAnim.restart();
   _cptPrendCoup++;
   setSprite(117,4996,90,100);
   break;
case 4:
```

```
            _clockAnim.restart();
            _cptPrendCoup=0;
            setSprite(24,163,96,103);
            fini=true;
            *degats=0;
            break;
        }
    }

    _posY=_scene.getBottom()-_tailleSprite.y;
    _sprite.setPosition(_posX,_posY);
    keepInWalls();
    return fini;
}


bool Dhalsim::apparition(sf::Sprite& bandeau)
{
    sf::Time elapsed = _clockAnim.getElapsedTime();
    int timeAnim = elapsed.asMilliseconds();
    bool fini=false;
    int delai=200;

    if(_cptApparition==0)
    {
        bandeau.setTexture(_texture);
        bandeau.setTextureRect(sf::IntRect(0,0,0,0));
        bandeau.setScale(_orientation*_scale,_scale);

        setSprite(24,32,51,115);
        _cptApparition ++;
    _posY=_scene.getBottom()-_tailleSprite.y;
    _sprite.setPosition(_posX,_posY);

    if (!_effetSonore.openFromFile("musique/Dhalsim/apparition.ogg"))
        std::cout<<"erreur musique";
    _effetSonore.play();
}
else if(timeAnim>delai)
{
    switch(_cptApparition)
    {
    case 1:
        _cptApparition ++;
        _clockAnim.restart();
        setSprite(83,32,52,115);
        _posX-=1*_scale*_orientation;
        break;
    case 2:
        _cptApparition ++;
        _clockAnim.restart();
        setSprite(143,32,53,115);
        _posX-=1*_scale*_orientation;
        break;
    case 3:
        _cptApparition ++;
        _clockAnim.restart();
        setSprite(204,32,59,115);
        _posX-=6*_scale*_orientation;
        bandeau.setTextureRect(sf::IntRect(462, 70,78,77));
        bandeau.setPosition(_posX-(_tailleSprite.x*_orientation),_posY);
```

```
      break;
   case 4:
      _cptApparition ++;
      _clockAnim.restart();
      setSprite(271,32,64,115);
      _posX-=5*_scale*_orientation;
      bandeau.setTextureRect(sf::IntRect(542, 70,78,77));
      break;
   case 5:
      _cptApparition ++;
      _clockAnim.restart();
      setSprite(343,32,58,115);
      _posX+=6*_scale*_orientation;
      bandeau.setTextureRect(sf::IntRect(622, 70,78,77));
      break;
   case 6:
      _cptApparition ++;
      _clockAnim.restart();
      setSprite(409,32,51,115);
      _posX+= 13*_scale*_orientation;

      bandeau.setPosition(_posX-(_tailleSprite.x*_orientation),_posY);
      break;
   }
   _posY=_scene.getBottom()-_tailleSprite.y;
   _sprite.setPosition(_posX,_posY);
}

if(_cptApparition>=7)
{
   _cptApparition++;
```

```
        bandeau.setPosition(_posX-
((_tailleSprite.x+_cptApparition*3)*_orientation),_posY+_cptApparition);

    }

    if(_cptApparition==70)

    {

        bandeau.setTextureRect(sf::IntRect(0,0,0,0));

        _cptApparition=0;

        fini=true;

        _sprite.setPosition(_posX,_posY);

    }



    return fini;

}


void Dhalsim::statique(Personnage& champEnnemi)

{

    sf::Time elapsed = _clockAnim.getElapsedTime();

    int timeAnim = elapsed.asMilliseconds();

    int delai=150;


    if(timeAnim>delai)

    {

        switch (_cptStatic)

        {

        case 0:

            _cptStatic ++;

            _clockAnim.restart();

            setSprite(24,163,96,103);

            break;

        case 1:
```

```
   _cptStatic ++;
   _clockAnim.restart();
   setSprite(128,163,97,103);
   break;
case 2:
   _cptStatic ++;
   _clockAnim.restart();
   setSprite(233,163,94,103);
   break;
case 3:
   _cptStatic ++;
   _clockAnim.restart();
   setSprite(335,163,94,103);
   break;
case 4:
   _cptStatic ++;
   _clockAnim.restart();
   setSprite(437,163,93,103);
   break;
case 5:
   _cptStatic ++;
   _clockAnim.restart();
   setSprite(538,163,92,103);
   break;
case 6:
   _cptStatic=0;
   _clockAnim.restart();
   setSprite(638,163,91,103);
   break;
}
_posY=_scene.getBottom()-_tailleSprite.y;
```

```
        _sprite.setPosition(_posX,_posY);
    }

    _hurtbox.setSize(sf::Vector2f(_tailleSprite.x*0.4,_tailleSprite.y*0.9));
    _hurtbox.setPosition(_posX+_tailleSprite.x*0.2*_orientation,_posY+_tailleSprite.y*0.1);
    _hitbox.setSize(sf::Vector2f(0,0));
    _gardebox.setSize(sf::Vector2f(0,0));



    rotate(champEnnemi);
    int n=0;
    collision(champEnnemi,n);
    keepInWalls();
}



void Dhalsim::garde()
{
    _cptStatic=0;

    sf::Time elapsed = _clockAnim.getElapsedTime();
    int timeAnim = elapsed.asMilliseconds();
    int delaiAnim=70;

    if(timeAnim>delaiAnim)
    {
        _clockAnim.restart();
        setSprite(125,4747,63,100);
        _gardebox.setSize(sf::Vector2f(_tailleSprite.x*0.2,_tailleSprite.y));
        _gardebox.setPosition(_posX+_tailleSprite.x*0.8*_orientation,_posY);
    }
    _posY=_scene.getBottom()-_tailleSprite.y;
    _sprite.setPosition(_posX,_posY);
    keepInWalls();
}
void Dhalsim::avancer(Personnage& champEnnemi)
{
    sf::Time elapsed2 = _clockMove.getElapsedTime();
    int timeMove = elapsed2.asMilliseconds();
    int deplacement=8;

    _posY=_scene.getBottom()-_tailleSprite.y;

    collision(champEnnemi,deplacement);

    if(timeMove>10)
    {
        _posX= _posX+deplacement*_orientation;
        _clockMove.restart();
    }

    sf::Time elapsed = _clockAnim.getElapsedTime();
    int timeAnim = elapsed.asMilliseconds();
    int delai=70;

    if(deplacement==0)
    {
        statique(champEnnemi);
    }
    else if(timeAnim>delai)
    {
```

```
switch (_cptAvancer)
{
case 0:
   _sprite.setPosition(_posX,_posY);
   _cptAvancer ++;
   _clockAnim.restart();
   setSprite(24,284,75,101);

   break;
case 1:
   _cptAvancer ++;
   _clockAnim.restart();
   setSprite(107,284,68,101);
   break;
case 2:
   _cptAvancer ++;
   _clockAnim.restart();
   setSprite(183,284,61,101);
   break;
case 3:
   _cptAvancer ++;
   _clockAnim.restart();
   setSprite(252,284,58,101);
   break;
case 4:
   _cptAvancer ++;
   _clockAnim.restart();
   setSprite(318,284,67,101);
   break;
case 5:
   _cptAvancer ++;
```

```
            _clockAnim.restart();
            setSprite(393,284,67,101);
            break;
        case 6:
            _cptAvancer++;
            _clockAnim.restart();
            setSprite(468,284,63,101);
            break;
        case 7:
            _cptAvancer=0;
            _clockAnim.restart();
            setSprite(539,284,66,101);
            break;
        }
    }
    collision(champEnnemi,deplacement);
    _hurtbox.setSize(sf::Vector2f(_tailleSprite.x*0.5,_tailleSprite.y));
    _hurtbox.setPosition(_posX+_tailleSprite.x*0.2*_orientation,_posY);
    rotate(champEnnemi);
    keepInWalls();
}


void Dhalsim::reculer()
{
    _cptStatic=0;

    sf::Time elapsed2 = _clockMove.getElapsedTime();
    int timeMove = elapsed2.asMilliseconds();
    int deplacement=6;
```

```
_posY=_scene.getBottom()-_tailleSprite.y;

if(timeMove>10)
{
   _posX= _posX-deplacement*_orientation;
   _clockMove.restart();
}

sf::Time elapsed = _clockAnim.getElapsedTime();
int timeAnim = elapsed.asMilliseconds();
int delai=70;

if(timeAnim > delai)
{
   switch (_cptReculer)
   {
   case 0:
      _sprite.setPosition(_posX,_posY);
      _cptReculer ++;
      _clockAnim.restart();
      setSprite(614,282,69,103);
      _posX-=_orientation*deplacement;
      _sprite.setPosition(_posX,_posY);
      break;
   case 1:
      _cptReculer ++;
      _clockAnim.restart();
      setSprite(691,282,63,103);
      _posX-=_orientation*deplacement;
      _sprite.setPosition(_posX,_posY);
      break;
```

```
case 2:
   _cptReculer ++;
   _clockAnim.restart();
   setSprite(762,282,60,103);
   _posX-=_orientation*deplacement;
   _sprite.setPosition(_posX,_posY);
   break;
case 3:
   _cptReculer ++;
   _clockAnim.restart();
   setSprite(830,282,63,103);
   _posX-=_orientation*deplacement;
   _sprite.setPosition(_posX,_posY);
   break;
case 4:
   _cptReculer ++;
   _clockAnim.restart();
   setSprite(901,282,62,103);
   _posX-=_orientation*deplacement;
   _sprite.setPosition(_posX,_posY);
   break;
case 5:
   _cptReculer ++;
   _clockAnim.restart();
   setSprite(971,282,57,103);
   _posX-=_orientation*deplacement;
   _sprite.setPosition(_posX,_posY);
   break;
case 6:
   _cptReculer++;
   _clockAnim.restart();
```

```
            setSprite(1036,282,60,103);
            _posX-=_orientation*deplacement;
            _sprite.setPosition(_posX,_posY);
            break;
        case 7:
            _cptReculer=0;
            _clockAnim.restart();
            setSprite(1104,282,63,103);
            _posX-=_orientation*deplacement;
            _sprite.setPosition(_posX,_posY);
            break;
        }
    }
    _gardebox.setSize(sf::Vector2f(0,0));
    _hurtbox.setSize(sf::Vector2f(_tailleSprite.x*0.5,_tailleSprite.y));
    _hurtbox.setPosition(_posX+_tailleSprite.x*0.2*_orientation,_posY);
    keepInWalls();
}


bool Dhalsim::sauter(int& lancerAttaque,Personnage& champEnnemi,int* degats,int& energie)
{
    _cptStatic=0;

    sf::Time elapsed2 = _clockMove.getElapsedTime();
    int timeMove = elapsed2.asMilliseconds();
    float v_grav = 1.7;

    if(timeMove > 10)
    {
        _vsaut += v_grav;
    _posY += _vsaut;
    _clockMove.restart();

    _sprite.setPosition(_posX,_posY);
    _hurtbox.setSize(sf::Vector2f(_tailleSprite.x*0.5,_tailleSprite.y*0.8));
    _hurtbox.setPosition(_posX+_tailleSprite.x*0.2*_orientation,_posY+_tailleSprite.y*0.1);
}

sf::Time elapsed = _clockAnim.getElapsedTime();
int timeAnim = elapsed.asMilliseconds();
int delai=70;
bool fini=false;

if(lancerAttaque!=-1)
{
    bool enAttaque=false;

    if(lancerAttaque==1)
        enAttaque=sautPunch(champEnnemi,degats,energie);
    else if(lancerAttaque==2)
        enAttaque=sautKick(champEnnemi,degats,energie);

    if(enAttaque)
    {
        lancerAttaque=-1;
        if(_cptSauter<4)
            _cptSauter=7-_cptSauter;
    }

}
else
```

```
{

   if(_cptSauter==0)
   {
      setSprite(974,1705,50,126);
      _clockAnim.restart();
      _cptSauter++;

      if (!_effetSonore.openFromFile("musique/Dhalsim/saut.ogg"))
         std::cout<<"erreur musique";
      _effetSonore.play();
   }
   else
   {
      int n=0;
      collisionsaut(champEnnemi,n);

      if(_cptSauter<8 && timeAnim>delai)
      {
         _cptSauter ++;
         _clockAnim.restart();
      }

      switch (_cptSauter)
      {
      case 1:
         setSprite(1084,1730,57,101);
         break;
      case 2:
         setSprite(1148,1744,60,87);
         break;
```

```
            case 7:
                setSprite(1084,1730,57,101);
                break;
            case 8:
                setSprite(974,1705,50,126);
                if(_posY + _tailleSprite.y + _vsaut >= _scene.getBottom())
                {
                    _cptSauter ++;
                }
                break;
            case 9:
                _cptSauter =0;
                setSprite(24,163,96,103);
                _posY=_scene.getBottom()-_tailleSprite.y;
                _vsaut = -40;
                fini = true;
                break;
            }
        }
    }

    keepInWalls();
    return fini;
}



bool Dhalsim::sauterAvant(Personnage& champEnnemi)
{
    _cptStatic=0;

    sf::Time elapsed2 = _clockMove.getElapsedTime();
```

```
int timeMove = elapsed2.asMilliseconds();
float v_grav = 1.7;
int deplacementX=15;

if(timeMove > 10)
{
    _vsaut += v_grav;
    _posY += _vsaut;
    collisionsaut(champEnnemi,deplacementX);

    if(_cptSauter!=8)
        _posX += deplacementX*_orientation;

    _clockMove.restart();

    _hurtbox.setSize(sf::Vector2f(_tailleSprite.x*0.6,_tailleSprite.y*0.8));
    _hurtbox.setPosition(_posX+_tailleSprite.x*0.2*_orientation,_posY+_tailleSprite.y*0.1);
}

sf::Time elapsed = _clockAnim.getElapsedTime();
int timeAnim = elapsed.asMilliseconds();
int delai=60;
bool fini=false;

if(timeAnim > delai)
{
    switch(_cptSauter)
    {
    case 0:
        _clockAnim.restart();
        _cptSauter++;
        setSprite(108,1720,68,111);


        if (!_effetSonore.openFromFile("musique/Dhalsim/saut.ogg"))
            std::cout<<"erreur musique";
        _effetSonore.play();
        break;
    case 1:
        _clockAnim.restart();
        _cptSauter++;
        setSprite(108,1720,68,111);
        break;
    case 2:
        _clockAnim.restart();
        _cptSauter++;
        setSprite(254,1756,68,75);
        break;
    case 3:
        _clockAnim.restart();
        _cptSauter++;
        break;
    case 4:
        _clockAnim.restart();
        _cptSauter++;
        setSprite(418,1770,79,61);
        break;
    case 5:
        _clockAnim.restart();
        _cptSauter++;
        setSprite(505,1770,57,61);
        break;
    case 6:
```

```
            _clockAnim.restart();
            _cptSauter++;
            setSprite(570,1784,103,47);
            break;
        case 7:
            _clockAnim.restart();
            setSprite(108,1720,68,111);
            if(_posY + _tailleSprite.y + _vsaut >= _scene.getBottom())
            {
                _cptSauter ++;
                setSprite(24,1720,82,111);
            }
            break;
        case 8:
            _clockAnim.restart();
            _cptSauter=0;
            _vsaut=-40;
            fini=true;
            rotate(champEnnemi);
            break;
        }
    }

    _sprite.setPosition(_posX,_posY);
    keepInWalls();
    return fini;
}

bool Dhalsim::sauterArriere(Personnage& champEnnemi)
{
    _cptStatic=0;
```

```
sf::Time elapsed2 = _clockMove.getElapsedTime();
int timeMove = elapsed2.asMilliseconds();
float v_grav = 1.7;
int deplacementX=15;

if(timeMove > 10)
{
   _clockMove.restart();

   _vsaut += v_grav;
   _posY += _vsaut;
   collisionsaut(champEnnemi,deplacementX);

   if(_cptSauter!=8)
      _posX -= deplacementX*_orientation;

   _sprite.setPosition(_posX,_posY);

   _hurtbox.setSize(sf::Vector2f(_tailleSprite.x*0.6,_tailleSprite.y*0.8));
   _hurtbox.setPosition(_posX+_tailleSprite.x*0.2*_orientation,_posY+_tailleSprite.y*0.1);
}

sf::Time elapsed = _clockAnim.getElapsedTime();
int timeAnim = elapsed.asMilliseconds();
int delai=70;
bool fini=false;

if(timeAnim > delai)
{
   switch(_cptSauter)
```

```
{
case 0:
   _clockAnim.restart();
   _cptSauter++;
   setSprite(108,1720,68,111);

   if (!_effetSonore.openFromFile("musique/Dhalsim/saut.ogg"))
      std::cout<<"erreur musique";
   _effetSonore.play();
   break;
case 1:
   _clockAnim.restart();
   _cptSauter++;
   setSprite(108,1720,68,111);
   break;
case 2:
   _clockAnim.restart();
   _cptSauter++;
   setSprite(570,1784,103,47);
   break;
case 3:
   _clockAnim.restart();
   _cptSauter++;
   setSprite(505,1770,57,61);
   break;
case 4:
   _clockAnim.restart();
   _cptSauter++;
   setSprite(418,1770,79,61);
   break;
case 5:
```

```
            _clockAnim.restart();

            _cptSauter++;

            setSprite(330,1785,80,46);

            break;

        case 6:

            _clockAnim.restart();

            _cptSauter++;

            setSprite(254,1756,68,75);

            break;

        case 7:

            _clockAnim.restart();

            setSprite(108,1720,68,111);

            if(_posY + _tailleSprite.y + _vsaut >= _scene.getBottom())

            {

                _cptSauter ++;

                setSprite(24,1720,82,111);

            }

            break;

        case 8:

            _clockAnim.restart();

            _cptSauter=0;

            _vsaut=-40;

            fini=true;

            break;

        }

    }


    keepInWalls();

    return fini;

}
```

```
void Dhalsim::accroupi(bool garde)
{
  _cptStatic=0;
  sf::Time elapsed = _clockAnim.getElapsedTime();
  int timeAnim = elapsed.asMilliseconds();
  int delai=35;
  if(_cptAccroupi==0)
  {
    if(timeAnim>delai)
    {
      _clockAnim.restart();
      _cptAccroupi++;
      setSprite(24,1424,82,95);
      _posY=_scene.getBottom()-_tailleSprite.y;
      _sprite.setPosition(_posX,_posY);
      _hurtbox.setSize(sf::Vector2f(_tailleSprite.x*0.9,_tailleSprite.y));
      _hurtbox.setPosition(_posX,_posY);
    }
  }
  else if(_cptAccroupi==1)
  {
    if(timeAnim>delai)
    {
      _clockAnim.restart();
      _cptAccroupi++;
      setSprite(114,1424,61,95);
      _hurtbox.setSize(sf::Vector2f(_tailleSprite.x*0.8,_tailleSprite.y*0.9));
      _hurtbox.setPosition(_posX+_tailleSprite.x*0.1,_posY+_tailleSprite.y*0.1);
    }
  }
  else
```

```
    {
        if(timeAnim>delai)
        {
            _clockAnim.restart();
            if(garde==true)
            {
                setSprite(263,4776,59,71);
            }
            else
                setSprite(183,1424,60,95);
        }
    }
}


bool Dhalsim::punch(Personnage& champEnnemi, int* degats,int& energie)
{
    _cptStatic=0;
    sf::Time elapsed = _clockAnim.getElapsedTime();
    int timeAnim = elapsed.asMilliseconds();
    int delai=50;
    bool fini=false;

    if(timeAnim > delai)
    {
        switch (_cptAction)
        {
        case 0:
            _cptAction ++;
            _clockAnim.restart();
            setSprite(24,419,82,117);
            _hurtbox.setSize(sf::Vector2f(_tailleSprite.x*0.5,_tailleSprite.y*0.8));
            _hurtbox.setPosition(_posX+_tailleSprite.x*0.2*_orientation,_posY+_tailleSprite.y*0.1);

            if (!_effetSonore.openFromFile("musique/Dhalsim/coup_poing.ogg"))
                std::cout<<"erreur musique";
            _effetSonore.play();
            break;
        case 1:
            _cptAction ++;
            _clockAnim.restart();
            setSprite(114,419,74,117);
            break;
        case 2:
            _cptAction ++;
            _clockAnim.restart();
            setSprite(197,419,108,117);
            _hitbox.setSize(sf::Vector2f(40*_scale,20*_scale));
            _hitbox.setPosition(_posX+68*_scale*_orientation,_posY+56*_scale);
            _spriteHitSpark.setPosition(_posX+68*_scale*_orientation,_posY+56*_scale);
            break;
        case 3:
            _cptAction ++;
            _clockAnim.restart();
            setSprite(313,419,108,117);
            break;
        case 4:
            _cptAction =0;
            _clockAnim.restart();
            setSprite(429,419,75,117);
            fini=true;
```

```
            _hitbox.setSize(sf::Vector2f(0,0));
            break;
        }
    }

    if(collisionCoup(champEnnemi))
    {
        if(_peutHitSpark)
            _hitSpark=true;
        *degats=5;
        energie+=10;

        if(champEnnemi.getPosX()==_scene.getRightLimit())
            _posX-=25*_scale*_orientation;
        else if(champEnnemi.getPosX()<=5)
            _posX+=25*_scale;
    }

    keepInWalls();
    return fini;
}

bool Dhalsim::sautPunch(Personnage& champEnnemi,int* degats,int& energie)
{
    _cptStatic=0;
    sf::Time elapsed = _clockAnim.getElapsedTime();
    int timeAnim = elapsed.asMilliseconds();
    int delai=70,deplacement=125;
    bool fini=false;

    if(timeAnim>delai)
```

```
{
    switch(_cptAction)
    {
    case 0:
        _cptAction ++;
        _clockAnim.restart();
        setSprite(896,1847,69,95);

        _hurtbox.setSize(sf::Vector2f(_tailleSprite.x*0.6,_tailleSprite.y*0.9));
        _hurtbox.setPosition(_posX+_tailleSprite.x*0.2*_orientation,_posY+_tailleSprite.y*0.1);

        if (!_effetSonore.openFromFile("musique/Dhalsim/coup_poing.ogg"))
            std::cout<<"erreur musique";
        _effetSonore.play();
        break;
    case 1:
        _cptAction ++;
        _clockAnim.restart();
        setSprite(973,1847,58,95);
        break;
    case 2:
        _cptAction ++;
        _clockAnim.restart();
        setSprite(1039,1847,105,95);

        _hitbox.setSize(sf::Vector2f(_tailleSprite.x*0.4,_tailleSprite.y*0.5));
        _hitbox.setPosition(_posX+_tailleSprite.x*0.6*_orientation,_posY+_tailleSprite.y*0.2);
        break;
    case 3:
        _cptAction=0;
        _clockAnim.restart();
```

```
            setSprite(973,1847,58,95);
            fini=true;

            _hitbox.setSize(sf::Vector2f(0,0));
            break;
        }
    }

    if(collisionCoup(champEnnemi))
    {
        *degats=5;
        energie+=10;

        if(champEnnemi.getPosX()==_scene.getRightLimit())
            _posX-=25*_scale*_orientation;
        else if(champEnnemi.getPosX()<=5)
            _posX+=25*_scale;
    }

    keepInWalls();
    return fini;
}

bool Dhalsim::punchSP(sf::Sprite& inutile,Personnage& champEnnemi, int* degats,int& energie)
{
    if(energie<20)
    {
        energie=-100;
        return true;
    }
```

```
_cptStatic=0;
sf::Time elapsed = _clockAnim.getElapsedTime();
int timeAnim = elapsed.asMilliseconds();
int delai=30;
bool fini=false;

if(timeAnim > delai)
{
  switch (_cptAction)
  {
  case 0:
    _cptAction ++;
    _clockAnim.restart();
    setSprite(24,660,86,92);

    _posX+=2*_scale*_orientation;

    if (!_effetSonore.openFromFile("musique/Dhalsim/punch_sp.ogg"))
      std::cout<<"erreur musique";
    _effetSonore.play();
    break;
  case 1:
    _cptAction ++;
    _clockAnim.restart();
    setSprite(118,665,95,87);
    break;
  case 2:
    _cptAction ++;
    _clockAnim.restart();
    setSprite(221,688,143,64);
    break;
```

```
case 3:
   _cptAction ++;
   _clockAnim.restart();
   setSprite(372,688,271,64);

   _hitbox.setSize(sf::Vector2f(_tailleSprite.x*0.6,_tailleSprite.y*0.3));
   _hitbox.setPosition(_posX+_tailleSprite.x*0.4*_orientation,_posY+_tailleSprite.y*0.1);
   _spriteHitSpark.setPosition(_posX+_tailleSprite.x*0.4*_orientation,_posY+_tailleSprite.y*0.1);
   break;
case 4:
   if(timeAnim>delai*4)
   {
      _cptAction++;
      _clockAnim.restart();
      setSprite(651,688,143,64);
      _hitbox.setSize(sf::Vector2f(0,0));
   }
   break;
case 5:
   _cptAction++;
   _clockAnim.restart();
   setSprite(802,688,147,64);
   break;
case 6:
   _cptAction++;
   _clockAnim.restart();
   setSprite(957,671,96,81);
   break;
case 7:
   _cptAction++;
   _clockAnim.restart();
```

```
        setSprite(1061,665,95,87);
        break;
    case 8:
        _cptAction++;
        _clockAnim.restart();
        setSprite(1164,663,86,89);
        break;
    case 9:
        _cptAction++;
        _clockAnim.restart();
        setSprite(1258,660,84,92);
        break;
    case 10:
        _cptAction=0;
        _clockAnim.restart();
        setSprite(24,163,96,103);
        fini=true;
        energie-=25;
        _posX+=2*_scale*_orientation;
        break;
    }
    _posY=_scene.getBottom()-_tailleSprite.y;
    _sprite.setPosition(_posX,_posY);
}

if(collisionCoup(champEnnemi))
{
    if(_peutHitSpark)
        _hitSpark=true;
    *degats=10;
```

```
        if(champEnnemi.getPosX()==_scene.getRightLimit())
            _posX-=25*_scale*_orientation;
        else if(champEnnemi.getPosX()<=5)
            _posX+=25*_scale;
    }

    return fini;
}

bool Dhalsim::kick(Personnage& champEnnemi, int* degats,int& energie)
{
    _cptStatic=0;
    sf::Time elapsed = _clockAnim.getElapsedTime();
    int timeAnim = elapsed.asMilliseconds();
    int delai=55;
    bool fini=false;

    if(timeAnim > delai)
    {
        switch (_cptAction)
        {
        case 0:
            _cptAction ++;
            _clockAnim.restart();
            setSprite(24,768,77,113);

            _hurtbox.setSize(sf::Vector2f(_tailleSprite.x*0.5,_tailleSprite.y*0.8));
            _hurtbox.setPosition(_posX+_tailleSprite.x*0.2*_orientation,_posY+_tailleSprite.y*0.1);

            if (!_effetSonore.openFromFile("musique/Dhalsim/coup_pied.ogg"))
                std::cout<<"erreur musique";
    _effetSonore.play();
    break;
case 1:
    _cptAction ++;
    _clockAnim.restart();
    setSprite(109,768,56,113);
    _posX+= 18*_scale*_orientation;
    break;
case 2:
    _cptAction ++;
    _clockAnim.restart();
    setSprite(173,768,126,113);
    _posX-=8*_scale*_orientation;

    _hitbox.setSize(sf::Vector2f(39*_scale,22*_scale));
    _hitbox.setPosition(_posX+87*_scale*_orientation,_posY+51*_scale);
    _spriteHitSpark.setPosition(_posX+87*_scale*_orientation,_posY+51*_scale);
    break;
case 3:
    _cptAction ++;
    _clockAnim.restart();
    setSprite(307,768,122,113);
    break;
case 4:
    _cptAction ++;
    _clockAnim.restart();
    setSprite(437,768,56,113);
    _posX+=8*_scale*_orientation;

    _hitbox.setSize(sf::Vector2f(0,0));
    break;
```

```
        case 5:
            _cptAction =0;
            _clockAnim.restart();
            setSprite(501,768,77,113);
            _posX-=18*_scale*_orientation;
            fini=true;
            break;
        }
        _sprite.setPosition(_posX,_posY);
    }

    if(collisionCoup(champEnnemi))
    {
        if(_peutHitSpark)
            _hitSpark=true;
        *degats=7;
        energie+=10;

        if(champEnnemi.getPosX()==_scene.getRightLimit())
            _posX-=25*_scale*_orientation;
        else if(champEnnemi.getPosX()<=5)
            _posX+=25*_scale;
    }

    keepInWalls();
    return fini;
}

bool Dhalsim::sautKick(Personnage& champEnnemi, int* degats,int& energie)
{
    sf::Time elapsed = _clockAnim.getElapsedTime();
```

```
int timeAnim = elapsed.asMilliseconds();
int delai=80,deplacement=_scene.getBottom()/6;
bool fini=false;

if(timeAnim>delai)
{
  switch(_cptAction)
  {
  case 0:
    _cptAction ++;
    _clockAnim.restart();
    setSprite(601,2319,53,110);

    _hurtbox.setSize(sf::Vector2f(_tailleSprite.x*0.6,_tailleSprite.y*0.9));
    _hurtbox.setPosition(_posX+_tailleSprite.x*0.2*_orientation,_posY+_tailleSprite.y*0.1);

    if (!_effetSonore.openFromFile("musique/Dhalsim/coup_pied.ogg"))
      std::cout<<"erreur musique";
    _effetSonore.play();
    break;
  case 1:
    _cptAction ++;
    _clockAnim.restart();
    setSprite(662,2319,63,110);
    break;
  case 2:
    _cptAction ++;
    _clockAnim.restart();
    setSprite(733,2319,99,110);

    _hitbox.setSize(sf::Vector2f(_tailleSprite.x*0.6,_tailleSprite.y*0.3));
```

```
            _hitbox.setPosition(_posX+_tailleSprite.x*0.4*_orientation,_posY+_tailleSprite.y*0.25);

_spriteHitSpark.setPosition(_posX+_tailleSprite.x*0.4*_orientation,_posY+_tailleSprite.y*0.25);
            break;
        case 3:
            _cptAction =0;
            _clockAnim.restart();
            setSprite(840,2319,70,110);
            fini=true;

            _hitbox.setSize(sf::Vector2f(0,0));
            break;
        }
    }

    if(collisionCoup(champEnnemi))
    {
        *degats=10;
        energie+=10;

        if(champEnnemi.getPosX()==_scene.getRightLimit())
            _posX-=25*_scale*_orientation;
        else if(champEnnemi.getPosX()<=5)
            _posX+=25*_scale;
    }

    keepInWalls();
    return fini;
}

bool Dhalsim::kickSP(Personnage& champEnnemi, int* degats,int& energie)
```

```
{
   if(energie<20)
   {
      energie=-100;
      return true;
   }

   _cptStatic=0;
   sf::Time elapsed = _clockAnim.getElapsedTime();
   int timeAnim = elapsed.asMilliseconds();
   int delai=50;
   bool fini=false;

   if(timeAnim > delai)
   {
      switch (_cptAction)
      {
      case 0:
         _cptAction ++;
         _clockAnim.restart();
         setSprite(24,1165,77,113);

         _posX+=2*_scale*_orientation;

         if (!_effetSonore.openFromFile("musique/Dhalsim/punch_sp.ogg"))
            std::cout<<"erreur musique";
         _effetSonore.play();
         break;
      case 1:
         _cptAction ++;
         _clockAnim.restart();
```

```
    setSprite(109,1159,97,119);
    break;
case 2:
    _cptAction ++;
    _clockAnim.restart();
    setSprite(214,1173,56,105);
    break;
case 3:
    _cptAction ++;
    _clockAnim.restart();
    setSprite(278,1173,126,105);
    break;
case 4:
    _cptAction++;
    _clockAnim.restart();
    setSprite(412,1173,232,105);

    _hitbox.setSize(sf::Vector2f(_tailleSprite.x*0.8,_tailleSprite.y*0.3));
    _hitbox.setPosition(_posX+_tailleSprite.x*0.2*_orientation,_posY);
    break;
case 5:
    if(timeAnim>delai*2)
    {
        _cptAction++;
        _clockAnim.restart();
        setSprite(652,1173,126,105);

        _hitbox.setSize(sf::Vector2f(0,0));
    }
    break;
case 6:
        _cptAction++;
        _clockAnim.restart();
        setSprite(924,1173,56,105);
        break;
    case 7:
        _cptAction++;
        _clockAnim.restart();
        setSprite(988,1173,100,105);
        break;
    case 8:
        _cptAction=0;
        _clockAnim.restart();
        setSprite(24,163,96,103);
        fini=true;
        energie-=25;
        _posX+=2*_scale*_orientation;
        break;
    }
    _posY=_scene.getBottom()-_tailleSprite.y;
    _sprite.setPosition(_posX,_posY);
}

if(collisionCoup(champEnnemi))
{
    if(_peutHitSpark)
        _hitSpark=true;
    *degats=10;

    if(champEnnemi.getPosX()==_scene.getRightLimit())
        _posX-=25*_scale*_orientation;
    else if(champEnnemi.getPosX()<=5)
```

```
        _posX+=25*_scale;
    }

    return fini;
}

bool Dhalsim::SP(sf::Sprite& bouleFeu,Personnage& champEnnemi, int* degats,int& energie)
{
    if(energie<50)
    {
        energie=-100;
        return true;
    }

    _cptStatic=0;
    sf::Time elapsed = _clockAnim.getElapsedTime();
    int timeAnim = elapsed.asMilliseconds();
    int delai=70;
    bool fini=false;

    if(timeAnim > delai)
    {
                switch (_cptAction)
                        {
                        case 0:
                            _cptAction ++;
                            _clockAnim.restart();
                            setSprite(24,3233,76,120);
                            _posX-=10*_scale*_orientation;

                            if (!_effetSonore.openFromFile("musique/Dhalsim/yoga_fire.ogg"))
```

```
              std::cout<<"erreur musique";
                  _effetSonore.play();


            bouleFeu.setTexture(_texture);
                  bouleFeu.setTextureRect(sf::IntRect(0,0,0,0));

bouleFeu.setPosition(bouleFeu.getPosition().x+20*_orientation,_posY+(_tailleSprite.y/2));
                  break;
          case 1:
            _cptAction ++;
            _clockAnim.restart();
            setSprite(108,3233,81,120);
            _posX-=6*_scale*_orientation;
                  break;
          case 2:
            _cptAction ++;
            _clockAnim.restart();
            setSprite(197,3233,58,120);
            _posX+=28*_scale*_orientation;
                  break;
          case 3:
            _cptAction ++;
            _clockAnim.restart();
            setSprite(263,3233,92,120);
            _posX-=4*_scale*_orientation;


                  bouleFeu.setTextureRect(sf::IntRect(357,3355,38,25));
                  bouleFeu.setScale(_orientation*_scale,_scale);

bouleFeu.setPosition(_posX+(_tailleSprite.x*_orientation/2),_posY+(_tailleSprite.y/3));
                  break;
          }
```

```
        _sprite.setPosition(_posX,_posY);
}

if(_cptAction>=4 && _cptAction<8)
{
        _cptAction ++;
        bouleFeu.setTextureRect(sf::IntRect(310,3355,39,25));

bouleFeu.setPosition(bouleFeu.getPosition().x+20*_orientation,_posY+(_tailleSprite.y/2));
}else if(_cptAction>7 && _cptAction<11)
{
        _cptAction ++;
        bouleFeu.setTextureRect(sf::IntRect(263,3355,39,25));

bouleFeu.setPosition(bouleFeu.getPosition().x+20*_orientation,_posY+(_tailleSprite.y/2));
}else if(_cptAction>10 && _cptAction<15)
{
        _cptAction ++;
        bouleFeu.setTextureRect(sf::IntRect(217,3355,38,25));

bouleFeu.setPosition(bouleFeu.getPosition().x+20*_orientation,_posY+(_tailleSprite.y/2));
}else if(_cptAction>14 && _cptAction<20)
{
        _cptAction ++;
        bouleFeu.setTextureRect(sf::IntRect(167,3355,42,25));

bouleFeu.setPosition(bouleFeu.getPosition().x+20*_orientation,_posY+(_tailleSprite.y/2));
}else if(_cptAction>19 && _cptAction<24)
{
        _cptAction ++;
        bouleFeu.setTextureRect(sf::IntRect(119,3355,40,25));

bouleFeu.setPosition(bouleFeu.getPosition().x+20*_orientation,_posY+(_tailleSprite.y/2));
```

```
        }else if(_cptAction>23 && _cptAction<28)
        {
                _cptAction ++;
                bouleFeu.setTextureRect(sf::IntRect(71,3355,40,25));

        bouleFeu.setPosition(bouleFeu.getPosition().x+20*_orientation,_posY+(_tailleSprite.y/2));
        }else if(_cptAction>27)
        {
                _cptAction ++;
                bouleFeu.setTextureRect(sf::IntRect(24,3355,39,25));

        bouleFeu.setPosition(bouleFeu.getPosition().x+20*_orientation,_posY+(_tailleSprite.y/2));
        }


        if(_cptAction>4)
        {
                if( (_orientation==1 && bouleFeu.getPosition().x>=_scene.getRightLimit()) ||
    (_orientation==-1 && bouleFeu.getPosition().x<=_scene.getLeftLimit()) )
                {
                        bouleFeu.setTextureRect(sf::IntRect(0,0,0,0));
                        fini=true;
                        energie-=50;
                        _cptAction=0;
                }


                if(collisionCoup(champEnnemi))
                {
                        *degats=30;


                        if(champEnnemi.getPosX()==_scene.getRightLimit())
                                _posX-=25*_scale*_orientation;
                bouleFeu.setTextureRect(sf::IntRect(0,0,0,0));
                fini=true;
                energie-=50;
                _cptAction=0;
            }
        }

    _hurtbox.setSize(sf::Vector2f(_tailleSprite.x*0.5,_tailleSprite.y*0.8));
    _hurtbox.setPosition(_posX+_tailleSprite.x*0.2*_orientation,_posY+_tailleSprite.y*0.1);


_hitbox.setSize(sf::Vector2f(bouleFeu.getGlobalBounds().width,bouleFeu.getGlobalBounds().height))
;
    _hitbox.setPosition(bouleFeu.getPosition().x,bouleFeu.getPosition().y);



    keepInWalls();
    return fini;
}
```

greg.cpp

```
#include "../IncludeManager.h"

using namespace std;

Greg::Greg(int orientation,Scene& s,sf::RenderWindow& window)
{
    double largeurFenetre=window.getSize().x;
    _scale=4.2*(largeurFenetre/1920);
```

```
            _orientation=-orientation;

_cptStatic=0;_cptAvancer=0;_cptReculer=0;_cptSauter=0;_cptApparition=0;_cptAction=0;_cptAccrou
pi=0;_cptPrendCoup=0;
    _vsaut = -40;

            if (!_texture.loadFromFile("sprites/sprite_greg.png"))
            {
                std::cout<<"Erreur au chargement du sprite";
            }
            _sprite.setTexture(_texture);
            _sprite.scale(_orientation*_scale,_scale);

            _icone.setTexture(_texture);
            _icone.setTextureRect(sf::IntRect(859,5579,119,108));
    _icone.scale(largeurFenetre/1920,largeurFenetre/1920);

    _hurtbox.setFillColor(sf::Color(255,255,255,0));
    _hurtbox.setOutlineColor(sf::Color::Green);
    _hurtbox.setOutlineThickness(4);

    _hitbox.setFillColor(sf::Color(255,255,255,0));
    _hitbox.setOutlineColor(sf::Color::Red);
    _hitbox.setOutlineThickness(4);

    _gardebox.setFillColor(sf::Color(255,255,255,0));
    _gardebox.setOutlineColor(sf::Color::Blue);
    _gardebox.setOutlineThickness(4);

    _spriteHitSpark.setColor(sf::Color(130,130,255,255));

    setScene(s);
```

```
}



bool Greg::victoire()//ok
{
            sf::Time elapsed = _clockAnim.getElapsedTime();
   int timeAnim = elapsed.asMilliseconds();
   int delaiAnim=150;
   bool fini=false;
   _hitbox.setSize(sf::Vector2f(0,0));

            if(timeAnim>delaiAnim)
            {
               switch (_cptApparition)
               {
               case 0:
                           _cptApparition ++;
                           _clockAnim.restart();
                                  setSprite(420,5374,67,94);

                                  _hurtbox.setSize(sf::Vector2f(0,0));

                                  if (!_effetSonore.openFromFile("musique/Greg/victoire.ogg"))
             std::cout<<"erreur musique";
          _effetSonore.play();
                        break;
               case 1:
                           _cptApparition ++;
                           _clockAnim.restart();
                           setSprite(977,5367,67,101);
                        break;
```

```
case 2:
            _cptApparition ++;
            _clockAnim.restart();
            setSprite(1115,5367,67,101);
        break;
case 3:
            _cptApparition ++;
            _clockAnim.restart();
            setSprite(1253,5367,67,101);
        break;
case 4:
            _cptApparition ++;
            _clockAnim.restart();
            setSprite(1391,5367,67,101);
        break;
case 5:
            _cptApparition ++;
            _clockAnim.restart();
            setSprite(1529,5367,67,101);
        break;
case 6:
            _cptApparition ++;
            _clockAnim.restart();
            setSprite(1667,5367,67,101);
        break;
case 7:
            _cptApparition ++;
            _clockAnim.restart();
            setSprite(1834,5367,116,101);
            _posX-=49*_scale*_orientation;
        break;
```

```
            case 8:
                    _cptApparition ++;
                    _clockAnim.restart();
                    setSprite(2364,5367,67,101);
                    _posX+=49*_scale*_orientation;
                  break;
                  }

                  _posY=_scene.getBottom()-_tailleSprite.y;
                  _sprite.setPosition(_posX,_posY);
          }

          if(_cptApparition==9 && timeAnim>1000)
          {
                  _clockAnim.restart();
                  _cptApparition=0;
                  fini=true;
          }

          return fini;
}

bool Greg::mort()//ok
{
          sf::Time elapsed = _clockAnim.getElapsedTime();
  int timeAnim = elapsed.asMilliseconds();
  int delaiAnim=100,deplacementX=_scene.getRightLimit()/12;
  bool fini=false;
  _hitbox.setSize(sf::Vector2f(0,0));

          if(timeAnim>delaiAnim)
```

```
 {
    switch (_cptApparition)
    {
    case 0:
            _cptApparition ++;
            _clockAnim.restart();
                setSprite(1,4753,65,97);


                _hurtbox.setSize(sf::Vector2f(0,0));


if (!_effetSonore.openFromFile("musique/Greg/mort.ogg"))
   std::cout<<"erreur musique";
_effetSonore.play();
          break;
    case 1:
            _cptApparition ++;
            _clockAnim.restart();
            setSprite(1,4965,80,77);


            _posX-=deplacementX*_orientation;
            _posY-=11*_scale;
          break;
    case 2:
            _cptApparition ++;
            _clockAnim.restart();
            setSprite(82,4961,105,44);


            _posX-=deplacementX*_orientation;
          break;
    case 3:
            _cptApparition ++;
            _clockAnim.restart();
            setSprite(188,4975,73,65);


            _posX-=deplacementX*_orientation;
            break;
case 4:
            _cptApparition ++;
            _clockAnim.restart();
            setSprite(278,4980,120,46);
         break;
case 5:
            _cptApparition ++;
            _clockAnim.restart();
            setSprite(399,5024,125,41);
         break;
         case 6:
            _cptApparition++;
            _clockAnim.restart();
            setSprite(651,5025,123,41);
            break;
         case 7:
            _cptApparition++;
            _clockAnim.restart();
            setSprite(775,5024,133,34);
            break;
         }


         if(_cptApparition >=4)
                  _posY=_scene.getBottom()-_tailleSprite.y;


         _sprite.setPosition(_posX,_posY);
```

```
            }

            if(_cptApparition==8 && timeAnim>2000)
            {
                        _clockAnim.restart();
                        _cptApparition=0;
                        fini=true;
            }

            keepInWalls();
            return fini;
}


bool Greg::parade(int* degats,sf::Sprite& effet)//ok
{
            bool fini=false;
            _cptSauter=0;_cptAction=0;
            effet.setTextureRect(sf::IntRect(0,0,0,0));
            _hurtbox.setSize(sf::Vector2f(0,0));

            sf::Time elapsed = _clockAnim.getElapsedTime();
    int timeAnim = elapsed.asMilliseconds();
    int delaiAnim=120;

    if(_cptPrendCoup==0)
    {
            setSprite(70,4648,68,102);
            _cptPrendCoup++;
    }else if(timeAnim > delaiAnim)
    {
```

```
                        if(_cptPrendCoup==1)
                        {
                                    _clockAnim.restart();
                                    _cptPrendCoup++;
                        }else{
                                    _clockAnim.restart();
                                    _cptPrendCoup=0;
                                    fini=true;
                                    *degats=0;
                        }
    }

    sf::Time elapsedDep = _clockMove.getElapsedTime();
    int timeDep = elapsedDep.asMilliseconds();
    int delaiDep=20,deplacement=_scene.getRightLimit()/200*_orientation;

    if(timeDep>delaiDep)
    {
            _clockMove.restart();
            _posX-=deplacement;
            _sprite.setPosition(_posX,_posY);
    }

    keepInWalls();
    return fini;
}


bool Greg::prendCoup(int* degats,sf::Sprite& effet,int& energie)//ok
{
            *degats=-1;
```

```
          bool fini=false;
          _cptSauter=0;_cptAction=0;
          effet.setTextureRect(sf::IntRect(0,0,0,0));
          sf::Time elapsed = _clockAnim.getElapsedTime();
int timeAnim = elapsed.asMilliseconds();
sf::Time elapsed2 = _clockMove.getElapsedTime();
int timeMove = elapsed2.asMilliseconds();
int delaiAnim=70;
int deplacement = 20;
_hurtbox.setSize(sf::Vector2f(0,0));
_gardebox.setSize(sf::Vector2f(0,0));

if(timeMove > 20){
    switch (_cptPrendCoup)
             {
             case 1:
       _posX -= deplacement * _orientation;
       _clockMove.restart();
                    break;
             case 2:
                       _posX -= deplacement * _orientation;
       _clockMove.restart();
                    break;
             }
}


if(timeAnim > delaiAnim)
{
          switch(_cptPrendCoup)
          {
```

```
     case 0:
            _clockAnim.restart();
            _cptPrendCoup++;
    setSprite(574,4748,69,102);
    if (!_effetSonore.openFromFile("musique/Greg/degat.ogg"))
      std::cout<<"erreur musique";
    _effetSonore.play();


            energie+=5;
            break;
     case 1:
            _clockAnim.restart();
            _cptPrendCoup++;
            setSprite(325,4750,73,99);
            break;
     case 2:
            _clockAnim.restart();
            _cptPrendCoup++;
            setSprite(574,4748,69,102);
            break;
     case 3:
            _cptPrendCoup=0;
              _clockAnim.restart();
              fini=true;
              *degats = 0;
            break;
     }
}
_sprite.setPosition(_posX,_posY);
keepInWalls();
return fini;
```

```
}



bool Greg::apparition(sf::Sprite& inutile)//ok
{
	sf::Time elapsed = _clockAnim.getElapsedTime();
    int timeAnim = elapsed.asMilliseconds();
    bool fini=false;
    int delaiAnim=200;

	if(_cptApparition==0)
	{
		setSprite(9,216,61,113);
		_cptApparition ++;

        if (!_effetSonore.openFromFile("musique/Greg/apparition.ogg"))
             std::cout<<"erreur musique";
          _effetSonore.play();
	}else if(timeAnim>delaiAnim)
	{
		switch(_cptApparition)
		{
		case 1:
		    _cptApparition ++;
		    _clockAnim.restart();
		    setSprite(80,216,61,113);
			break;
		case 2:
		    _cptApparition ++;
		    _clockAnim.restart();
		    setSprite(152,216,67,113);
```

```
            break;
case 3:
   _cptApparition ++;
   _clockAnim.restart();
   setSprite(227,216,70,113);
            break;
case 4:
   _cptApparition ++;
   _clockAnim.restart();
   setSprite(306,216,64,113);
            break;
case 5:
   _cptApparition ++;
   _clockAnim.restart();
   setSprite(379,216,61,113);
            break;
case 6:
   _cptApparition ++;
   _clockAnim.restart();
   setSprite(449,216,61,113);
            break;
case 7:
   _cptApparition ++;
   _clockAnim.restart();
   setSprite(519,216,61,113);
            break;
case 8:
   _cptApparition ++;
   _clockAnim.restart();
   setSprite(589,216,61,113);
            break;
```

```
                    case 9:
                       _cptApparition=0;
                       _clockAnim.restart();
                       setSprite(654,216,68,113);
                              fini=true;
                              break;
                    }
          }
          keepInWalls();
          return fini;
}

void Greg::statique(Personnage& champEnnemi)//ok
{
   sf::Time elapsed = _clockAnim.getElapsedTime();
   int timeAnim = elapsed.asMilliseconds();
   int delaiAnim=50;
   if(timeAnim>delaiAnim)
   {
             switch (_cptStatic)
             {
             case 0:
                       _cptStatic ++;
                       _clockAnim.restart();
                    setSprite(2,360,66,105);
                    _posY=_scene.getBottom()-_tailleSprite.y;
                              _sprite.setPosition(_posX,_posY);
                    break;
             case 1:
                       _cptStatic ++;
                       _clockAnim.restart();
```

```
                setSprite(71,360,66,105);
            break;
    case 2:
                _cptStatic ++;
                _clockAnim.restart();
                setSprite(140,360,66,105);
            break;
    case 3:
                _cptStatic ++;
                _clockAnim.restart();
                setSprite(209,360,66,105);
            break;
    case 4:
                _cptStatic ++;
                _clockAnim.restart();
                setSprite(279,360,64,105);
            break;
    case 5:
                _cptStatic=0;
                _clockAnim.restart();
                setSprite(347,360,66,105);
            break;
    }
}

_hurtbox.setPosition(_posX+_tailleSprite.x*0.2*_orientation,_posY+_tailleSprite.y*0.1);
_hurtbox.setSize(sf::Vector2f(_tailleSprite.x*0.6,_tailleSprite.y*0.9));
_hitbox.setSize(sf::Vector2f(0,0));
_gardebox.setSize(sf::Vector2f(0,0));

int n=0;
```

```
            collision(champEnnemi,n);
   rotate(champEnnemi);
   keepInWalls();
}



void Greg::garde()
{
            _cptStatic=0;
            _posY=_scene.getBottom()-_tailleSprite.y;
            sf::Time elapsed =
            _clockAnim.getElapsedTime();
   int timeAnim = elapsed.asMilliseconds();
   int delaiAnim=70;
   if(timeAnim>delaiAnim)
   {
            _clockAnim.restart();
            setSprite(3,4658,65,92);
            _gardebox.setSize(sf::Vector2f(_tailleSprite.x*0.2,_tailleSprite.y));
            _gardebox.setPosition(_posX+_tailleSprite.x*0.8*_orientation,_posY);
   }
   _posY=_scene.getBottom()-_tailleSprite.y;
   _sprite.setPosition(_posX,_posY);
   keepInWalls();
}

void Greg::avancer(Personnage& champEnnemi)//ok
{
            _posY=_scene.getBottom()-_tailleSprite.y;
            _cptStatic=0;
            sf::Time elapsed1 = _clockAnim.getElapsedTime();
```

```
int timeAnim = elapsed1.asMilliseconds();
sf::Time elapsed2 = _clockMove.getElapsedTime();
int timeMove = elapsed2.asMilliseconds();
int delai=70;
int deplacement=12;

collision(champEnnemi,deplacement);
if(_cptAvancer > 5){
   _cptAvancer = 0;
}

         if(deplacement==0)
{
         statique(champEnnemi);
}

if(timeMove > 10){
   if(_cptAvancer < 6){
      _posX= _posX+deplacement*_orientation;
            _clockMove.restart();
   }
}
if(timeAnim>50){
   _cptAvancer ++;
   _clockAnim.restart();
}
switch (_cptAvancer)
{
case 0:
   if(timeAnim>20){
      _cptAvancer ++;
```

```
            _clockAnim.restart();
        }
        setSprite(-3,626,72,104);
        break;
    case 1:
        setSprite(70,626,69,104);
        break;
    case 2:
        setSprite(143,626,69,104);
        break;
    case 3:
        setSprite(212,626,68,104);
        break;
    case 4:
        setSprite(281,626,69,104);
        break;
    case 5:
        setSprite(350,626,71,104);
        break;
    }

    _sprite.setPosition(_posX,_posY);
    _hurtbox.setPosition(_posX+_tailleSprite.x*0.2*_orientation,_posY+_tailleSprite.y*0.1);
    _hurtbox.setSize(sf::Vector2f(_tailleSprite.x*0.6,_tailleSprite.y*0.9));

    rotate(champEnnemi);

            keepInWalls();
}
```

```
void Greg::reculer()//ok
{
   if(_cptReculer > 3){
      _cptReculer = 0;
   }
           _posY=_scene.getBottom()-_tailleSprite.y;
           _cptStatic=0;
           sf::Time elapsed1 = _clockAnim.getElapsedTime();
   int timeAnim = elapsed1.asMilliseconds();
   sf::Time elapsed2 = _clockMove.getElapsedTime();
   int timeMove = elapsed2.asMilliseconds();
   int delai=70;
   int deplacement=10;

   if(timeMove > 10){
      if(_cptReculer < 4){
         _posX -= deplacement*_orientation;
         _clockMove.restart();
      }
   }
   if(timeAnim > delai)
          {
             _cptReculer++;
      _clockAnim.restart();
          }

   switch (_cptReculer)
   {
   case 0:
      setSprite(427,624,63,106);
      break;
   case 1:
      setSprite(497,624,61,106);
      break;
   case 2:
      setSprite(564,624,55,106);
      break;
   case 3:
      setSprite(632,624,55,106);
      break;
   }

                _sprite.setPosition(_posX,_posY);
            _hurtbox.setPosition(_posX+_tailleSprite.x*0.2*_orientation,_posY+_tailleSprite.y*0.1);
            _hurtbox.setSize(sf::Vector2f(_tailleSprite.x*0.6,_tailleSprite.y*0.9));

   _gardebox.setSize(sf::Vector2f(0,0));

            keepInWalls();
}



bool Greg::sauter(int& lancerAttaque,Personnage& champEnnemi,int* degats,int& energie)//ok
{
   float v_grav = 1.7;
            _cptStatic=0;
            sf::Time elapsed = _clockAnim.getElapsedTime();
   int timeAnim = elapsed.asMilliseconds();
   sf::Time elapsed2 = _clockMove.getElapsedTime();
   int timeMove = elapsed2.asMilliseconds();

   int delaiAnim=100;
```

```
bool fini=false;

if(lancerAttaque!=-1)
{
        bool enAttaque=false;

                if(lancerAttaque==1)
                        enAttaque=sautPunch(champEnnemi,degats,energie);
        else if(lancerAttaque==2)
                enAttaque=sautKick(champEnnemi,degats,energie);

        if(enAttaque)
        {
                lancerAttaque=-1;
                if(_cptSauter<4)
                        _cptSauter=7-_cptSauter;
        }

}else
{
    if(timeMove > 10){
    _vsaut += v_grav;
     _posY += _vsaut;
    _clockMove.restart();
    }
    if(timeAnim > delaiAnim){
        if(_cptSauter < 6)
           _cptSauter++;
        _clockAnim.restart();
    }
    switch (_cptSauter)
```

```
{
case 0:
   if(timeAnim > 20){
      _cptSauter ++;
      _clockAnim.restart();


      if (!_effetSonore.openFromFile("musique/Greg/saut.ogg"))
         std::cout<<"erreur musique";
      _effetSonore.play();
   }
   setSprite(651,818,63,100);
   break;
case 1:
   setSprite(714,809,70,117);
   break;
case 2:
   setSprite(791,764,64,99);
   break;
case 3:
   setSprite(861,737,61,81);
   break;
case 4:
   setSprite(925,729,61,75);
   break;
case 5:
   setSprite(1000,739,64,97);
   break;
case 6:
   setSprite(1071,765,62,115);
   if(_posY + _tailleSprite.y + _vsaut >= _scene.getBottom()){
      _cptSauter ++;
```

```
            }
            break;
        case 7:
            _cptSauter =0;
            setSprite(2,423,66,108);
            _posY=_scene.getBottom()-_tailleSprite.y;
            _vsaut = -40;
            fini = true;
            break;
        }
    }
            _hurtbox.setSize(sf::Vector2f(_tailleSprite.x*0.6,_tailleSprite.y*0.8));
            _hurtbox.setPosition(_posX+_tailleSprite.x*0.2*_orientation,_posY+_tailleSprite.y*0.1);

            _sprite.setPosition(_posX,_posY);
    keepInWalls();
    return fini;
}



bool Greg::sauterAvant(Personnage& champEnnemi)//ok
{
    float v_grav = 1.7;
            _cptStatic=0;
            sf::Time elapsed = _clockAnim.getElapsedTime();
    int timeAnim = elapsed.asMilliseconds();
    sf::Time elapsed2 = _clockMove.getElapsedTime();
    int timeMove = elapsed2.asMilliseconds();

    int delaiAnim=70;
    int deplacementX=15;
```

```
bool fini=false;


if(timeMove > 10){
    _vsaut += v_grav;
    _posY += _vsaut;
    collisionsaut(champEnnemi,deplacementX);
    _posX += deplacementX*_orientation;;
    _clockMove.restart();
}
if(timeAnim > delaiAnim){
    if(_cptSauter < 8 && _cptSauter != 2){
        if(_cptSauter == 1){
            _posX += 25*_orientation;
            _posY -= 5;
        }
        _cptSauter++;
        _clockAnim.restart();
    }


    if(_cptSauter == 4){
        _posX -= 100*_orientation;;
        _posY += 50;
    }
    else if(_cptSauter == 5){
        _posX += 100*_orientation;;
        _posY -= 50;
    }
    else if(_cptSauter == 6){
        _posX -= 140*_orientation;;
        _posY += 100;
    }
    else if(_cptSauter == 7){
        _posX += 70*_orientation;;
        _posY -= 70;
    }
}
switch (_cptSauter)
{
    case 0:
        if(timeAnim > 20){
            _cptSauter ++;
            _clockAnim.restart();

            if (!_effetSonore.openFromFile("musique/Greg/saut.ogg"))
               std::cout<<"erreur musique";
           _effetSonore.play();
        }
    setSprite(651,820,63,98);
    _hurtbox.setSize(sf::Vector2f(_tailleSprite.x*0.6,_tailleSprite.y*0.8));
    _hurtbox.setPosition(_posX+_tailleSprite.x*0.2*_orientation,_posY+_tailleSprite.y*0.1);
    break;
case 1:
    setSprite(714,811,70,115);
    _hurtbox.setSize(sf::Vector2f(_tailleSprite.x*0.6,_tailleSprite.y*0.8));
    _hurtbox.setPosition(_posX+_tailleSprite.x*0.2*_orientation,_posY+_tailleSprite.y*0.1);
    break;
case 2:
    setSprite(1348,785,62,115);
    _hurtbox.setSize(sf::Vector2f(_tailleSprite.x*0.6,_tailleSprite.y*0.8));
    _hurtbox.setPosition(_posX+_tailleSprite.x*0.2*_orientation,_posY+_tailleSprite.y*0.1);
        if(timeAnim > 200){
            _cptSauter ++;
            _clockAnim.restart();
        }
    break;
case 3:
    setSprite(1488,927,65,90);
    _hurtbox.setSize(sf::Vector2f(_tailleSprite.x*0.6,_tailleSprite.y*0.8));
    _hurtbox.setPosition(_posX+_tailleSprite.x*0.2*_orientation,_posY+_tailleSprite.y*0.1);
    break;
case 4:
    setSprite(1410,760,96,46);
    _hurtbox.setSize(sf::Vector2f(_tailleSprite.x*0.6,_tailleSprite.y*0.8));
    _hurtbox.setPosition(_posX+_tailleSprite.x*0.2*_orientation,_posY+_tailleSprite.y*0.1);
    break;
case 5:
    setSprite(1510,737,53,82);
    _hurtbox.setSize(sf::Vector2f(_tailleSprite.x*0.6,_tailleSprite.y*0.8));
    _hurtbox.setPosition(_posX+_tailleSprite.x*0.2*_orientation,_posY+_tailleSprite.y*0.1);
    break;
case 6:
    setSprite(1568,768,120,52);
    _hurtbox.setSize(sf::Vector2f(_tailleSprite.x*0.4,_tailleSprite.y*0.8));
    _hurtbox.setPosition(_posX+_tailleSprite.x*0.3*_orientation,_posY+_tailleSprite.y*0.1);
    break;
case 7:
    setSprite(1689,738,70,103);
    _hurtbox.setSize(sf::Vector2f(_tailleSprite.x*0.6,_tailleSprite.y*0.8));
    _hurtbox.setPosition(_posX+_tailleSprite.x*0.4*_orientation,_posY+_tailleSprite.y*0.1);
    break;
case 8:
    setSprite(1071,765,62,115);
    _hurtbox.setSize(sf::Vector2f(_tailleSprite.x*0.6,_tailleSprite.y*0.8));
```

```
        _hurtbox.setPosition(_posX+_tailleSprite.x*0.2*_orientation,_posY+_tailleSprite.y*0.1);
        if(_posY + _tailleSprite.y + _vsaut >= _scene.getBottom()){
            _cptSauter ++;
        }
        break;
    case 9:
        _cptSauter =0;
        setSprite(2,423,66,108);
        _hurtbox.setSize(sf::Vector2f(_tailleSprite.x*0.6,_tailleSprite.y*0.8));
        _hurtbox.setPosition(_posX+_tailleSprite.x*0.2*_orientation,_posY+_tailleSprite.y*0.1);
        _posY=_scene.getBottom()-_tailleSprite.y;
        _posX += 45*_orientation;
        _vsaut = -40;
        rotate(champEnnemi);
        fini =  true;
        break;
    }
            _sprite.setPosition(_posX,_posY);
    keepInWalls();
    return fini;
}

bool Greg::sauterArriere(Personnage& champEnnemi)//ok
{
            float v_grav = 1.7;
            _cptStatic=0;
            sf::Time elapsed = _clockAnim.getElapsedTime();
    int timeAnim = elapsed.asMilliseconds();
    sf::Time elapsed2 = _clockMove.getElapsedTime();
    int timeMove = elapsed2.asMilliseconds();
int delaiAnim=50;
int deplacementX=15;
bool fini=false;

if(timeMove > 10){
    _vsaut += v_grav;
    _posY += _vsaut;
    collisionsaut(champEnnemi,deplacementX);
    _posX -= deplacementX*_orientation;;
    _clockMove.restart();
}

if(timeAnim > delaiAnim){
    if(_cptSauter < 8){
        if(_cptSauter == 1){
            _posX += 25*_orientation;
            _posY -= 5;
        }
        _cptSauter++;
        _clockAnim.restart();
    }
    switch(_cptSauter)
    {
    case 2:
        _posX -= 100*_orientation;
        break;
    case 3:
        _posX -= 50*_orientation;
        _posY += 50;
        break;
    case 4:
        _posX += 100*_orientation;
        _posY -= 50;
        break;
    case 5:
        _posX -= 50*_orientation;
        _posY += 50;
        break;
    case 6:
        _posX += 75*_orientation;
        _posY -= 50;
        break;
    case 7:
        _posY -= 50;
        break;
    }
}
switch (_cptSauter)
{
case 0:
    if(timeAnim > 20){
         _cptSauter ++;
         _clockAnim.restart();

        if (!_effetSonore.openFromFile("musique/Greg/saut.ogg"))
            std::cout<<"erreur musique";
        _effetSonore.play();
    }
    setSprite(651,818,63,100);
    break;
case 1:
     setSprite(791,764,64,99);
```

```
   break;
case 2:
   setSprite(1689,738,70,103);
   break;
case 3:
   setSprite(1568,768,120,52);
   break;
case 4:
   setSprite(1510,737,53,82);
   break;
case 5:
   setSprite(1410,759,93,47);
   break;
case 6:
   setSprite(1488,927,65,90);
   break;
case 7:
   setSprite(518,982,61,107);
   break;
case 8:
   setSprite(1071,765,62,115);
   if(_posY + _tailleSprite.y + _vsaut >= _scene.getBottom()){
      _cptSauter ++;
   }
   break;
case 9:
   _cptSauter =0;
   setSprite(2,423,66,108);
   _posY=_scene.getBottom()-_tailleSprite.y;
   _vsaut = -40;
   rotate(champEnnemi);
```

```
        fini = true;
        break;
    }
    _hurtbox.setSize(sf::Vector2f(_tailleSprite.x*0.6,_tailleSprite.y*0.8));
        _hurtbox.setPosition(_posX+_tailleSprite.x*0.2*_orientation,_posY+_tailleSprite.y*0.1);

        _sprite.setPosition(_posX,_posY);
    keepInWalls();
    return fini;
}

void Greg::accroupi(bool garde)//ok
{
        _cptStatic=0;
        sf::Time elapsed = _clockAnim.getElapsedTime();
    int timeAnim = elapsed.asMilliseconds();
    int delaiAnim=35;
    if(_cptAccroupi==0)
    {
        if(timeAnim>delaiAnim)
        {
            _clockAnim.restart();
            _cptAccroupi++;
            setSprite(73,530,62,82);

                _hurtbox.setSize(sf::Vector2f(_tailleSprite.x*0.9,_tailleSprite.y));
                _hurtbox.setPosition(_posX,_posY);
        }
    }else
    {
        if(timeAnim>delaiAnim)
```

```
            {
                        _clockAnim.restart();
                        if(garde==true)
                                    setSprite(212,4674,64,76);
                        else
                        {
                                    setSprite(142,537,62,75);
                                    _hurtbox.setSize(sf::Vector2f(_tailleSprite.x*0.8,_tailleSprite.y*0.9));
                                    _hurtbox.setPosition(_posX+_tailleSprite.x*0.1,_posY+_tailleSprite.y*0.1);
                        }
            }
   }
   _posY=_scene.getBottom()-_tailleSprite.y;
   _sprite.setPosition(_posX,_posY);
}

bool Greg::punch(Personnage& champEnnemi,int* degats,int& energie)//ok
{
            _cptStatic=0;
            sf::Time elapsed = _clockAnim.getElapsedTime();
   int timeAnim = elapsed.asMilliseconds();
   int delaiAnim=60;
   bool fini=false;

   if(timeAnim > delaiAnim)
   {
                        switch (_cptAction)
                        {
                        case 0:
                           _cptAction ++;
                           _clockAnim.restart();
```

```
                  setSprite(3,1310,74,102);


                  _hurtbox.setSize(sf::Vector2f(_tailleSprite.x*0.6,_tailleSprite.y*0.9));

         _hurtbox.setPosition(_posX+_tailleSprite.x*0.2*_orientation,_posY+_tailleSprite.y*0.1);


        if (!_effetSonore.openFromFile("musique/Greg/coup_poing.ogg"))
          std::cout<<"erreur musique";
        _effetSonore.play();
                        break;
                case 1:
                  _cptAction ++;
                  _clockAnim.restart();
                  setSprite(80,1308,102,104);
                  _hitbox.setSize(sf::Vector2f(40*_scale,20*_scale));
                  _hitbox.setPosition(_posX+60*_scale*_orientation,_posY+10*_scale);
                  _spriteHitSpark.setPosition(_posX+60*_scale*_orientation,_posY+10*_scale);
                        break;
                case 2:
                  _cptAction++;
                  _clockAnim.restart();
                  setSprite(3,1310,74,102);


                  _hitbox.setSize(sf::Vector2f(0,0));
                        break;
                case 3:
                        _cptAction=0;
                        _clockAnim.restart();
                        setSprite(2,360,66,105);
                        fini=true;
                        break;
                        }
            }

            if(collisionCoup(champEnnemi))
            {
                        *degats=5;
                        energie+=10;
       if(_peutHitSpark)
          _hitSpark = true;
                        if(champEnnemi.getPosX()==_scene.getRightLimit())
                                    _posX-=25*_scale*_orientation;
       else if(champEnnemi.getPosX()<=5)
          _posX+=25*_scale;
            }

            _posY=_scene.getBottom()-_tailleSprite.y;
   _sprite.setPosition(_posX,_posY);
            keepInWalls();
            return fini;
}

bool Greg::sautPunch(Personnage& champEnnemi,int* degats,int& energie)
{
            _cptStatic=0;
            sf::Time elapsed = _clockAnim.getElapsedTime();
   int timeAnim = elapsed.asMilliseconds();
   int delaiAnim=60,deplacement=125;
   bool fini=false;

   if(timeAnim>delaiAnim)
   {
    switch(_cptAction)
    {
    case 0:
            _cptAction ++;
              _clockAnim.restart();
              setSprite(150,1794,55,73);

              _hurtbox.setSize(sf::Vector2f(_tailleSprite.x*0.6,_tailleSprite.y*0.9));

  _hurtbox.setPosition(_posX+_tailleSprite.x*0.2*_orientation,_posY+_tailleSprite.y*0.1);

if (!_effetSonore.openFromFile("musique/Greg/coup_poing.ogg"))
   std::cout<<"erreur musique";
_effetSonore.play();
                    break;
            case 1:
              _cptAction ++;
              _clockAnim.restart();
              setSprite(206,1794,74,79);
                    break;
            case 2:
                    _cptAction++;
                    _clockAnim.restart();
                    setSprite(281,1794,98,72);

                    _hitbox.setSize(sf::Vector2f(_tailleSprite.x*0.4,_tailleSprite.y*0.5));

  _hitbox.setPosition(_posX+_tailleSprite.x*0.6*_orientation,_posY+_tailleSprite.y*0.2);
                    break;
            case 3:
              _cptAction=0;
              _clockAnim.restart();
```

```
                        setSprite(281,1794,98,72);
                        fini=true;

                        _hitbox.setSize(sf::Vector2f(0,0));
                                break;
                }
            }

            if(collisionCoup(champEnnemi))
            {
                    *degats=5;
                    energie+=10;

                    if(champEnnemi.getPosX()==_scene.getRightLimit())
                            _posX-=25*_scale*_orientation;
        else if(champEnnemi.getPosX()<=5)
            _posX+=25*_scale;
            }

    keepInWalls();
    return fini;
}

bool Greg::punchSP(sf::Sprite& inutile,Personnage& champEnnemi, int* degats,int& energie)
{
            if(energie<20)
            {
                    energie=-100;
                    return true;
            }
```

```
         _cptStatic=0;
         sf::Time elapsed = _clockAnim.getElapsedTime();
int timeAnim = elapsed.asMilliseconds();
int delaiAnim=50,deplacement=_tailleSprite.x/2;
bool fini=false;



if(timeAnim > delaiAnim)
{
                collisionsaut(champEnnemi,deplacement);

                switch (_cptAction)
                {
                case 0:
                   _cptAction ++;
                   _clockAnim.restart();
                   setSprite(8,3795,66,86);

                   _posY=_scene.getBottom()-_tailleSprite.y;
                   if (!_effetSonore.openFromFile("musique/Greg/shoryuken.ogg"))
                      std::cout<<"erreur musique";
                         _effetSonore.play();
                         break;
                case 1:
                   _cptAction ++;
                   _clockAnim.restart();
                   setSprite(83,3791,78,90);

                   _posY=_scene.getBottom()-_tailleSprite.y;

                   _hitbox.setSize(sf::Vector2f(_tailleSprite.x*0.2,_tailleSprite.y*0.4));
_hitbox.setPosition(_posX+_tailleSprite.x*0.8*_orientation,_posY+_tailleSprite.y*0.1);

_spriteHitSpark.setPosition(_posX+_tailleSprite.x*0.8*_orientation,_posY+_tailleSprite.y*0.1);
                         break;
                case 2:
                   _cptAction ++;
                   _clockAnim.restart();
                   setSprite(176,3754,62,129);

                   _posX+=deplacement*_orientation;
                   _posY=_scene.getBottom()-_tailleSprite.y;

                   _hitbox.setSize(sf::Vector2f(_tailleSprite.x*0.4,_tailleSprite.y*0.5));
                   _hitbox.setPosition(_posX+_tailleSprite.x*0.6*_orientation,_posY);
                         break;
                case 3:
                   _cptAction ++;
                   _clockAnim.restart();
                   setSprite(244,3686,55,121);

                   _posX+=deplacement*_orientation;
                   _posY-=_tailleSprite.y/2;

                   _hitbox.setSize(sf::Vector2f(_tailleSprite.x*0.4,_tailleSprite.y*0.5));
                   _hitbox.setPosition(_posX+_tailleSprite.x*0.6*_orientation,_posY);
                         break;
                case 4:
                         if(timeAnim>delaiAnim*2)
                         {
                                 _cptAction++;
                           _clockAnim.restart();
```

```
                    setSprite(315,3697,61,117);


                    _posX+=deplacement/2*_orientation;
                    _posY+=_tailleSprite.y/10;


                    _hitbox.setSize(sf::Vector2f(0,0));
                }
                break;
            case 5:
              _cptAction++;
              _clockAnim.restart();
              setSprite(380,3779,63,99);


              _posY=_scene.getBottom()-_tailleSprite.y;
                break;
            case 6:
              _cptAction=0;
              _clockAnim.restart();
              setSprite(2,433,66,98);
              fini=true;
              energie-=25;


              _posY=_scene.getBottom()-_tailleSprite.y;
              break;
            }


            if( (_orientation==1 && _posX+_tailleSprite.x >= champEnnemi.getPosX()-
champEnnemi.getHurtbox().getGlobalBounds().width)
        || (_orientation==-1 && _posX-_tailleSprite.x <=
champEnnemi.getPosX()+champEnnemi.getHurtbox().getGlobalBounds().width) )
         {
```

```
                    _posX=champEnnemi.getPosX()-
(champEnnemi.getHurtbox().getGlobalBounds().width+_tailleSprite.x+deplacement)*_orientation;
                }


                        _sprite.setPosition(_posX,_posY);



            _hurtbox.setPosition(_posX+_tailleSprite.x*0.2*_orientation,_posY+_tailleSprite.y*0.1);
                        _hurtbox.setSize(sf::Vector2f(_tailleSprite.x*0.6,_tailleSprite.y*0.9));
            }


            if(collisionCoup(champEnnemi))
            {
                if(_peutHitSpark){
          _hitSpark = true;
                }
                        *degats=10;


                        if(champEnnemi.getPosX()==_scene.getRightLimit())
                                    _posX-=25*_scale*_orientation;
       else if(champEnnemi.getPosX()<=5)
          _posX+=25*_scale;
            }


            keepInWalls();
            return fini;
}


bool Greg::kick(Personnage& champEnnemi,int* degats,int& energie)
{
            _cptStatic=0;
            sf::Time elapsed = _clockAnim.getElapsedTime();
```

```
int timeAnim = elapsed.asMilliseconds();
int delaiAnim=70;
bool fini=false;

if(timeAnim > delaiAnim)
{
            switch (_cptAction)
            {
            case 0:
              _cptAction ++;
              _clockAnim.restart();
              setSprite(497,2550,67,103);

              _hurtbox.setSize(sf::Vector2f(_tailleSprite.x*0.6,_tailleSprite.y*0.9));

     _hurtbox.setPosition(_posX+_tailleSprite.x*0.2*_orientation,_posY+_tailleSprite.y*0.1);

    if (!_effetSonore.openFromFile("musique/Greg/coup_pied.ogg"))
      std::cout<<"erreur musique";
    _effetSonore.play();
                    break;
            case 1:
              _cptAction ++;
              _clockAnim.restart();
              setSprite(566,2550,65,103);
                    break;
            case 2:
              _cptAction ++;
              _clockAnim.restart();
              setSprite(656,2550,118,103);
            _hitbox.setSize(sf::Vector2f(80*_scale,22*_scale));
              _hitbox.setPosition(_posX+36*_scale*_orientation,_posY);
              _spriteHitSpark.setPosition(_posX+80*_scale*_orientation,_posY);
                    break;
            case 3:
              _cptAction ++;
              _clockAnim.restart();
              setSprite(775,2550,65,103);


              _hitbox.setSize(sf::Vector2f(0,0));
                    break;
            case 4:
              _cptAction =0;
              _clockAnim.restart();
              setSprite(867,2550,65,103);
              fini=true;
                    break;
            }
    }


    if(collisionCoup(champEnnemi))
    {
      if(_peutHitSpark){
  _hitSpark = true;
      }
            *degats=7;
            energie+=10;


            if(champEnnemi.getPosX()==_scene.getRightLimit())
                    _posX-=25*_scale*_orientation;
else if(champEnnemi.getPosX()<=5)
```

```
            _posX+=25*_scale;
             }
             _posY=_scene.getBottom()-_tailleSprite.y;
    _sprite.setPosition(_posX,_posY);
             keepInWalls();
             return fini;
}

bool Greg::sautKick(Personnage& champEnnemi,int* degats,int& energie)
{
             sf::Time elapsed = _clockAnim.getElapsedTime();
    int timeAnim = elapsed.asMilliseconds();
    int delaiAnim=60,deplacement=_scene.getBottom()/6;
    bool fini=false;

             if(timeAnim>delaiAnim)
    {
                switch(_cptAction)
                {
                case 0:
                       _cptAction ++;
                          _clockAnim.restart();
                          setSprite(228,3022,58,117);

                          _hurtbox.setSize(sf::Vector2f(_tailleSprite.x*0.6,_tailleSprite.y*0.9));

             _hurtbox.setPosition(_posX+_tailleSprite.x*0.2*_orientation,_posY+_tailleSprite.y*0.1);

            if (!_effetSonore.openFromFile("musique/Greg/coup_pied.ogg"))
               std::cout<<"erreur musique";
            _effetSonore.play();
```

```
                        break;
            case 1:
                    _cptAction ++;
                       _clockAnim.restart();
                       setSprite(298,3013,59,98);
                                break;
                    case 2:
                       _cptAction ++;
                       _clockAnim.restart();
                       setSprite(367,3020,92,107);

                       _hitbox.setSize(sf::Vector2f(_tailleSprite.x*0.6,_tailleSprite.y*0.2));

_hitbox.setPosition(_posX+_tailleSprite.x*0.4*_orientation,_posY+_tailleSprite.y*0.35);
                                break;
                    case 3:
                                _cptAction =0;
                                _clockAnim.restart();
                                setSprite(472,3036,61,102);
                                fini=true;

                                _hitbox.setSize(sf::Vector2f(0,0));
                                break;
            }
        }

        if(collisionCoup(champEnnemi))
        {
                *degats=7;
                energie+=10;
```

```
                        if(champEnnemi.getPosX()==_scene.getRightLimit())
                                _posX-=25*_scale*_orientation;
        else if(champEnnemi.getPosX()<=5)
            _posX+=25*_scale;
            }

    keepInWalls();
    return fini;
}

bool Greg::kickSP(Personnage& champEnnemi, int* degats,int& energie)
{
        if(energie<20)
        {
                energie=-100;
                return true;
        }

        _cptStatic=0;
        sf::Time elapsed = _clockAnim.getElapsedTime();
    int timeAnim = elapsed.asMilliseconds();
    int delaiAnim=70,deplacementY=_scene.getBottom()/7,deplacementX=50*_orientation;
    bool fini=false;

    if(timeAnim > delaiAnim)
    {
                switch (_cptAction)
                {
                case 0:
                    _cptAction ++;
                    _clockAnim.restart();
                    setSprite(1,3039,71,110);
                    _posY=_scene.getBottom()-_tailleSprite.y;

            if (!_effetSonore.openFromFile("musique/Greg/tatsumaki.ogg"))
               std::cout<<"erreur musique";
            _effetSonore.play();
                                break;
                case 1:
                    _cptAction ++;
                    _clockAnim.restart();
                    setSprite(75,3036,61,87);
                    _posY-=deplacementY;
                                break;
                case 2:
                    _cptAction ++;
                    _clockAnim.restart();
                    setSprite(148,3025,54,68);
                    _posY-=deplacementY;
                                break;
                case 3:
                    _cptAction ++;
                    _clockAnim.restart();
                    setSprite(228,3022,58,77);
                    _posY-=deplacementY/2;
                                break;
                case 4:
                                _cptAction++;
                                _clockAnim.restart();
                                setSprite(298,3013,59,98);
                                _posY+=deplacementY;
                                break;
                case 5:
                                _cptAction++;
                    _clockAnim.restart();
                    setSprite(366,3020,93,108);
                    _posY+=deplacementY;


                    _hitbox.setSize(sf::Vector2f(_tailleSprite.x*0.6,_tailleSprite.y*0.3));

              _hitbox.setPosition(_posX+_tailleSprite.x*0.4*_orientation,_posY+_tailleSprite.y*0.4);

              _spriteHitSpark.setPosition(_posX+_tailleSprite.x*0.4*_orientation,_posY+_tailleSprite.y*0.4
      );
                                break;
                case 6:
                    _cptAction++;
                    _clockAnim.restart();
                    setSprite(472,3036,61,102);
                    _posY=_scene.getBottom()-_tailleSprite.y;
                    _hitbox.setSize(sf::Vector2f(0,0));
                                break;
                case 7:
                    _cptAction=0;
                    _clockAnim.restart();
                    setSprite(538,3057,63,89);
                    fini=true;
                    energie-=25;
                    _posY=_scene.getBottom()-_tailleSprite.y;
                                break;
                }
                _posX+=deplacementX;
                _sprite.setPosition(_posX,_posY);
              }
```

```
            if(collisionCoup(champEnnemi))
            {
               if(_peutHitSpark)
          _hitSpark = true;
                        *degats=10;

                        if(champEnnemi.getPosX()==_scene.getRightLimit())
                                    _posX-=25*_scale*_orientation;
       else if(champEnnemi.getPosX()<=5)
          _posX+=25*_scale;
            }

            keepInWalls();
            return fini;
}

bool Greg::SP(sf::Sprite& bouleFeu,Personnage& champEnnemi,int* degats,int& energie)
{
            if(energie<50)
            {
                        energie=-100;
                        return true;
            }

            _cptStatic=0;
            sf::Time elapsed = _clockAnim.getElapsedTime();
   int timeAnim = elapsed.asMilliseconds();
   int delaiAnim=70;
   bool fini=false;
```

```
if(timeAnim > delaiAnim)
{
            switch (_cptAction)
                    {
                    case 0:
                                if (!_effetSonore.openFromFile("musique/Greg/hadouken.ogg"))
                          std::cout<<"erreur musique";
                       _effetSonore.play();


                       _cptAction++;
                       _clockAnim.restart();
                       setSprite(10,3493,74,90);


                       bouleFeu.setTexture(_texture);
                                bouleFeu.setTextureRect(sf::IntRect(0,0,0,0));
      bouleFeu.setPosition(bouleFeu.getPosition().x+20*_orientation,_posY);
                                break;
                    case 1:
                       _cptAction ++;
                       _clockAnim.restart();
                       setSprite(890,3520,91,90);
                       _posX-=1*_scale*_orientation;
                                break;
                    case 2:
                       _cptAction ++;
                       _clockAnim.restart();
                       setSprite(986,3520,111,90);
                       _posX-=20*_scale*_orientation;
                                break;
                    case 3:
                       _cptAction ++;
                    _clockAnim.restart();
                    setSprite(742,3630,115,90);
                    _posX-=4*_scale*_orientation;
                    break;
                case 4:
                    _cptAction ++;
                    _clockAnim.restart();
                    setSprite(993,3630,117,90);
                    _posX-=2*_scale*_orientation;
                    break;
                case 5:
                    _cptAction ++;
                    _clockAnim.restart();
                    setSprite(1124,3636,98,85);
                    _posX+=20*_scale*_orientation;
                    break;
                case 6:
                    _cptAction ++;
                    _clockAnim.restart();
                    setSprite(1229,3635,119,86);
                    _posX+=2*_scale*_orientation;

                        bouleFeu.setTextureRect(sf::IntRect(1127,5553,142,126));
                        bouleFeu.setScale(_orientation,1);
                        bouleFeu.setPosition(_posX+(_tailleSprite.x*0.7*_orientation),_posY);
                        break;
                }
        }

sf::Time elapsedEffet = _clockEffet.getElapsedTime();
int timeEffet = elapsedEffet.asMilliseconds();
int delaiEffet=10;


if(timeEffet>delaiEffet)
{
   if(_cptAction>6 && _cptAction<11)
   {
      setSprite(413,3495,114,88);
      _cptAction ++;
      bouleFeu.setTextureRect(sf::IntRect(1395,5555,128,130));
      bouleFeu.setPosition(bouleFeu.getPosition().x+20*_orientation,_posY);
   }else if(_cptAction>10 && _cptAction<15)
   {
      _cptAction ++;
      bouleFeu.setTextureRect(sf::IntRect(1544,5556,99,133));
      bouleFeu.setPosition(bouleFeu.getPosition().x+20*_orientation,_posY);
   }else if(_cptAction>14 && _cptAction<19)
   {
      _cptAction ++;
      bouleFeu.setTextureRect(sf::IntRect(1680,5553,65,135));
      bouleFeu.setPosition(bouleFeu.getPosition().x+20*_orientation,_posY);
   }else if(_cptAction>18 && _cptAction<23)
   {
      _cptAction ++;
      bouleFeu.setTextureRect(sf::IntRect(1762,5557,99,133));
      bouleFeu.setPosition(bouleFeu.getPosition().x+20*_orientation,_posY);
   }else if(_cptAction>22 && _cptAction<27)
   {
      _cptAction ++;
      bouleFeu.setTextureRect(sf::IntRect(1887,5556,131,129));
      bouleFeu.setPosition(bouleFeu.getPosition().x+20*_orientation,_posY);
      if(_cptAction==27)
```

```
            _cptAction=7;
        }
        _clockEffet.restart();
    }

    if(_cptAction>6)
    {
        if( (_orientation==1 && bouleFeu.getPosition().x>=_scene.getRightLimit()) || (_orientation==-1
&& bouleFeu.getPosition().x<=_scene.getLeftLimit()) )
        {
            bouleFeu.setTextureRect(sf::IntRect(0,0,0,0));
            fini=true;
            energie-=50;
            _cptAction=0;
        }

        if(collisionCoup(champEnnemi))
        {
            *degats=30;
            bouleFeu.setTextureRect(sf::IntRect(0,0,0,0));
            fini=true;
            energie-=50;
            _cptAction=0;
        }

    }


            _hurtbox.setSize(sf::Vector2f(_tailleSprite.x*0.6,_tailleSprite.y*0.9));
            _hurtbox.setPosition(_posX+_tailleSprite.x*0.2*_orientation,_posY+_tailleSprite.y*0.1);
```

```
_hitbox.setSize(sf::Vector2f(bouleFeu.getGlobalBounds().width,bouleFeu.getGlobalBounds().height))
;
            _hitbox.setPosition(bouleFeu.getPosition().x,bouleFeu.getPosition().y);



            _posY=_scene.getBottom()-_tailleSprite.y;
    _sprite.setPosition(_posX,_posY);
            keepInWalls();
            return fini;
}
```

ryu.cpp

```
#include "../IncludeManager.h"

using namespace std;

Ryu::Ryu(int orientation,Scene& s,sf::RenderWindow& window)
{
    double largeurFenetre=window.getSize().x;
    _scale=4.2*(largeurFenetre/1920);

            _orientation=-orientation;

_cptStatic=0;_cptAvancer=0;_cptReculer=0;_cptSauter=0;_cptApparition=0;_cptAction=0;_cptAccrou
pi=0;_cptPrendCoup=0;
    _vsaut = -40;

            if (!_texture.loadFromFile("sprites/sprite_ryu.png"))
            {
                std::cout<<"Erreur au chargement du sprite";
            }
            _sprite.setTexture(_texture);
            _sprite.setScale(_orientation*_scale,_scale);

            _icone.setTexture(_texture);
            _icone.setTextureRect(sf::IntRect(824,5573,124,104));
    _icone.scale(largeurFenetre/1920,largeurFenetre/1920);

    _hurtbox.setFillColor(sf::Color(255,255,255,0));
    _hurtbox.setOutlineColor(sf::Color::Green);
    _hurtbox.setOutlineThickness(4);

    _hitbox.setFillColor(sf::Color(255,255,255,0));
    _hitbox.setOutlineColor(sf::Color::Red);
    _hitbox.setOutlineThickness(4);

    _gardebox.setFillColor(sf::Color(255,255,255,0));
    _gardebox.setOutlineColor(sf::Color::Blue);
    _gardebox.setOutlineThickness(4);

    setScene(s);
}



bool Ryu::victoire()
{
            sf::Time elapsed = _clockAnim.getElapsedTime();
    int timeAnim = elapsed.asMilliseconds();
    int delaiAnim=150;
    bool fini=false;
    _hitbox.setSize(sf::Vector2f(0,0));
```

```
if(timeAnim>delaiAnim)
{
   switch (_cptApparition)
   {
   case 0:
              _cptApparition ++;
              _clockAnim.restart();
                     setSprite(420,5374,67,94);


                     _hurtbox.setSize(sf::Vector2f(0,0));


                     if (!_effetSonore.openFromFile("musique/Ryu/ryu_victoire.ogg"))
                 std::cout<<"erreur musique";
                     _effetSonore.play();
           break;
   case 1:
              _cptApparition ++;
              _clockAnim.restart();
              setSprite(977,5374,67,94);
           break;
   case 2:
              _cptApparition ++;
              _clockAnim.restart();
              setSprite(1115,5374,67,94);
           break;
   case 3:
              _cptApparition ++;
              _clockAnim.restart();
              setSprite(1253,5374,67,94);
           break;
   case 4:
              _cptApparition ++;
              _clockAnim.restart();
              setSprite(1391,5374,67,94);
           break;
   case 5:
              _cptApparition ++;
              _clockAnim.restart();
              setSprite(1529,5374,67,94);
           break;
   case 6:
              _cptApparition ++;
              _clockAnim.restart();
              setSprite(1667,5374,67,94);
           break;
   case 7:
              _cptApparition ++;
              _clockAnim.restart();
              setSprite(1834,5374,116,94);
              _posX-=49*_scale*_orientation;
           break;
   case 8:
              _cptApparition ++;
              _clockAnim.restart();
              setSprite(2364,5374,67,94);
              _posX+=49*_scale*_orientation;
           break;
           }

           _posY=_scene.getBottom()-_tailleSprite.y;
           _sprite.setPosition(_posX,_posY);
```

```
            }

            if(_cptApparition==9 && timeAnim>1000)
            {
                        _clockAnim.restart();
                        _cptApparition=0;
                        fini=true;
            }

            return fini;
}

bool Ryu::mort()
{
            sf::Time elapsed = _clockAnim.getElapsedTime();
   int timeAnim = elapsed.asMilliseconds();
   int delaiAnim=100,deplacementX=_scene.getRightLimit()/12;
   bool fini=false;
   _hitbox.setSize(sf::Vector2f(0,0));

            if(timeAnim>delaiAnim)
            {
               switch (_cptApparition)
               {
               case 0:
                           _cptApparition ++;
                           _clockAnim.restart();
                                  setSprite(1,4763,65,87);

                                  _hurtbox.setSize(sf::Vector2f(0,0));
```

```
if (!_effetSonore.openFromFile("musique/Ryu/mort.ogg"))
   std::cout<<"erreur musique";
_effetSonore.play();
            break;
     case 1:
               _cptApparition ++;
               _clockAnim.restart();
               setSprite(1,4965,80,77);

               _posX-=deplacementX*_orientation;
               _posY-=11*_scale;
            break;
     case 2:
               _cptApparition ++;
               _clockAnim.restart();
               setSprite(82,4961,105,44);

               _posX-=deplacementX*_orientation;
            break;
     case 3:
               _cptApparition ++;
               _clockAnim.restart();
               setSprite(188,4975,73,65);

               _posX-=deplacementX*_orientation;
               break;
     case 4:
               _cptApparition ++;
               _clockAnim.restart();
               setSprite(278,4980,120,46);
            break;
   case 5:
               _cptApparition ++;
               _clockAnim.restart();
               setSprite(399,5024,125,41);
            break;
            case 6:
               _cptApparition++;
               _clockAnim.restart();
               setSprite(651,5025,123,41);
               break;
            case 7:
               _cptApparition++;
               _clockAnim.restart();
               setSprite(775,5024,133,34);
               break;
            }

            if(_cptApparition >=4)
                     _posY=_scene.getBottom()-_tailleSprite.y;

            _sprite.setPosition(_posX,_posY);
}

if(_cptApparition==8 && timeAnim>2000)
{
            _clockAnim.restart();
            _cptApparition=0;
            fini=true;
}

keepInWalls();
```

```
            return fini;
}



bool Ryu::parade(int* degats,sf::Sprite& effet)
{
            bool fini=false;
            _cptSauter=0;_cptAction=0;
            effet.setTextureRect(sf::IntRect(0,0,0,0));
            _hurtbox.setSize(sf::Vector2f(0,0));

            sf::Time elapsed = _clockAnim.getElapsedTime();
   int timeAnim = elapsed.asMilliseconds();
   int delaiAnim=120;

   if(_cptPrendCoup==0)
   {
            setSprite(70,4659,68,91);
            _cptPrendCoup++;
   }else if(timeAnim > delaiAnim)
   {
                        if(_cptPrendCoup==1)
                        {
                                    _clockAnim.restart();
                                    _cptPrendCoup++;
                        }else{
                                    _clockAnim.restart();
                                    _cptPrendCoup=0;
                                    fini=true;
                                    *degats=0;
                        }
   }

   sf::Time elapsedDep = _clockMove.getElapsedTime();
   int timeDep = elapsedDep.asMilliseconds();
   int delaiDep=20,deplacement=_scene.getRightLimit()/200*_orientation;

   if(timeDep>delaiDep)
   {
           _clockMove.restart();
           _posX-=deplacement;
           _sprite.setPosition(_posX,_posY);
   }

   keepInWalls();
   return fini;
}


bool Ryu::prendCoup(int* degats,sf::Sprite& effet,int& energie)
{
           *degats=-1;
           bool fini=false;
           _cptSauter=0;_cptAction=0;_vsaut=-40;
           effet.setTextureRect(sf::IntRect(0,0,0,0));
           sf::Time elapsed = _clockAnim.getElapsedTime();
   int timeAnim = elapsed.asMilliseconds();
   sf::Time elapsed2 = _clockMove.getElapsedTime();
   int timeMove = elapsed2.asMilliseconds();
   int delaiAnim=70;
   int deplacement = 20;
   _hurtbox.setSize(sf::Vector2f(0,0));
```

```
_gardebox.setSize(sf::Vector2f(0,0));

if(timeMove > 20){
   switch (_cptPrendCoup)
        {
        case 1:
     _posX -= deplacement * _orientation;
     _clockMove.restart();
              break;
        case 2:
                _posX -= deplacement * _orientation;
     _clockMove.restart();
              break;
        }
}


if(timeAnim > delaiAnim)
{
      switch(_cptPrendCoup)
      {
      case 0:
              _clockAnim.restart();
              _cptPrendCoup++;
     setSprite(574,4752,73,98);

     if (!_effetSonore.openFromFile("musique/Ryu/degat.ogg"))
        std::cout<<"erreur musique";
     _effetSonore.play();

              energie+=5;
```

```
                        break;
                case 1:
                        _clockAnim.restart();
                        _cptPrendCoup++;
                        setSprite(325,4752,73,98);
                        break;
                case 2:
                        _clockAnim.restart();
                        _cptPrendCoup++;
                        setSprite(574,4752,73,98);
                        break;
                case 3:
                        _cptPrendCoup=0;
                           _clockAnim.restart();
                           fini=true;
                           *degats = 0;
                        break;
                }
    }
    _sprite.setPosition(_posX,_posY);
    keepInWalls();
    return fini;
}


bool Ryu::apparition(sf::Sprite& inutile)
{
                sf::Time elapsed = _clockAnim.getElapsedTime();
    int timeAnim = elapsed.asMilliseconds();
    bool fini=false;
    int delaiAnim=200;
    if(_cptApparition==0)
    {
            setSprite(9,225,61,104);
            _cptApparition ++;

if (!_effetSonore.openFromFile("musique/Ryu/apparition.ogg"))
  std::cout<<"erreur musique";
_effetSonore.play();
    }else if(timeAnim>delaiAnim)
    {
            switch(_cptApparition)
            {
            case 1:
              _cptApparition ++;
              _clockAnim.restart();
              setSprite(80,225,61,104);
                    break;
            case 2:
              _cptApparition ++;
              _clockAnim.restart();
              setSprite(152,225,67,104);
                    break;
            case 3:
              _cptApparition ++;
              _clockAnim.restart();
              setSprite(227,225,70,104);
                    break;
            case 4:
              _cptApparition ++;
              _clockAnim.restart();
```

```
                setSprite(306,225,64,104);
                        break;
            case 5:
                _cptApparition ++;
                _clockAnim.restart();
                setSprite(379,225,61,104);
                        break;
            case 6:
                _cptApparition ++;
                _clockAnim.restart();
                setSprite(449,225,61,104);
                        break;
            case 7:
                _cptApparition ++;
                _clockAnim.restart();
                setSprite(519,225,61,104);
                        break;
            case 8:
                _cptApparition ++;
                _clockAnim.restart();
                setSprite(589,225,61,104);
                        break;
            case 9:
                _cptApparition=0;
                _clockAnim.restart();
                setSprite(654,225,68,104);
                        fini=true;
                        break;
            }
}
keepInWalls();
```

```
        return fini;
}

void Ryu::statique(Personnage& champEnnemi)
{
  sf::Time elapsed = _clockAnim.getElapsedTime();
  int timeAnim = elapsed.asMilliseconds();
  int delaiAnim=50;
  if(timeAnim>delaiAnim)
  {
          switch (_cptStatic)
          {
          case 0:
                  _cptStatic ++;
                  _clockAnim.restart();
                setSprite(2,433,66,98);
                _posY=_scene.getBottom()-_tailleSprite.y;
                        _sprite.setPosition(_posX,_posY);
                break;
          case 1:
                  _cptStatic ++;
                  _clockAnim.restart();
                  setSprite(71,433,66,98);
                break;
          case 2:
                  _cptStatic ++;
                  _clockAnim.restart();
                  setSprite(140,433,66,98);
                break;
          case 3:
                  _cptStatic ++;
```

```
                    _clockAnim.restart();
                    setSprite(209,433,66,98);
                break;
            case 4:
                    _cptStatic ++;
                    _clockAnim.restart();
                    setSprite(279,433,64,98);
                break;
            case 5:
                    _cptStatic=0;
                    _clockAnim.restart();
                    setSprite(347,433,66,98);
                break;
            }
        }

        _hurtbox.setPosition(_posX+_tailleSprite.x*0.2*_orientation,_posY+_tailleSprite.y*0.1);
        _hurtbox.setSize(sf::Vector2f(_tailleSprite.x*0.6,_tailleSprite.y*0.9));
        _hitbox.setSize(sf::Vector2f(0,0));
        _gardebox.setSize(sf::Vector2f(0,0));

        int n=0;
        collision(champEnnemi,n);
    rotate(champEnnemi);
    keepInWalls();
}


void Ryu::garde()
{
        _cptStatic=0;
```

```
            sf::Time elapsed = _clockAnim.getElapsedTime();
   int timeAnim = elapsed.asMilliseconds();
   int delaiAnim=70;

   if(timeAnim>delaiAnim)
   {
            _clockAnim.restart();
            setSprite(3,4658,65,92);
            _gardebox.setSize(sf::Vector2f(_tailleSprite.x*0.2,_tailleSprite.y));
            _gardebox.setPosition(_posX+_tailleSprite.x*0.8*_orientation,_posY);
   }

   _posY=_scene.getBottom()-_tailleSprite.y;
   _sprite.setPosition(_posX,_posY);
   keepInWalls();
}

void Ryu::avancer(Personnage& champEnnemi)
{
            _posY=_scene.getBottom()-_tailleSprite.y;
            _cptStatic=0;

            sf::Time elapsed1 = _clockAnim.getElapsedTime();
   int timeAnim = elapsed1.asMilliseconds();
   int delai=70;

   sf::Time elapsed2 = _clockMove.getElapsedTime();
   int timeMove = elapsed2.asMilliseconds();
   int deplacement=12;
```

```
collision(champEnnemi,deplacement);
if(_cptAvancer > 5){
   _cptAvancer = 0;
}

        if(deplacement==0)
{
        statique(champEnnemi);
}

if(timeMove > 10){
   if(_cptAvancer < 6){
      _posX= _posX+deplacement*_orientation;
              _clockMove.restart();
   }
}
if(timeAnim>50){
   _cptAvancer ++;
   _clockAnim.restart();
}
switch (_cptAvancer)
{
case 0:
   if(timeAnim>20){
      _cptAvancer ++;
      _clockAnim.restart();
   }
   setSprite(-3,634,72,96);
   break;
case 1:
   setSprite(70,634,69,96);
```

```
        break;
    case 2:
        setSprite(143,634,69,96);
        break;
    case 3:
        setSprite(212,634,68,96);
        break;
    case 4:
        setSprite(281,634,69,96);
        break;
    case 5:
        setSprite(350,634,71,96);
        break;
    }

    _sprite.setPosition(_posX,_posY);
    _hurtbox.setPosition(_posX+_tailleSprite.x*0.2*_orientation,_posY+_tailleSprite.y*0.1);
    _hurtbox.setSize(sf::Vector2f(_tailleSprite.x*0.6,_tailleSprite.y*0.9));

    rotate(champEnnemi);

            keepInWalls();
}



void Ryu::reculer()
{
    if(_cptReculer > 3){
        _cptReculer = 0;
    }
            _posY=_scene.getBottom()-_tailleSprite.y;
        _cptStatic=0;
        sf::Time elapsed1 = _clockAnim.getElapsedTime();
int timeAnim = elapsed1.asMilliseconds();
sf::Time elapsed2 = _clockMove.getElapsedTime();
int timeMove = elapsed2.asMilliseconds();
int delai=70;
int deplacement=10;

if(timeMove > 10){
   if(_cptReculer < 4){
      _posX -= deplacement*_orientation;
      _clockMove.restart();
   }
}
if(timeAnim > delai)
        {
           _cptReculer++;
   _clockAnim.restart();
        }

switch (_cptReculer)
{
case 0:
   setSprite(427,634,63,96);
   break;
case 1:
   setSprite(497,634,61,96);
   break;
case 2:
   setSprite(564,634,55,96);
   break;
   case 3:
      setSprite(632,634,55,96);
      break;
   }

            _sprite.setPosition(_posX,_posY);
         _hurtbox.setPosition(_posX+_tailleSprite.x*0.2*_orientation,_posY+_tailleSprite.y*0.1);
         _hurtbox.setSize(sf::Vector2f(_tailleSprite.x*0.6,_tailleSprite.y*0.9));

   _gardebox.setSize(sf::Vector2f(0,0));

         keepInWalls();
}



bool Ryu::sauter(int& lancerAttaque,Personnage& champEnnemi,int* degats,int& energie)
{
   float v_grav = 1.7;
         _cptStatic=0;
         sf::Time elapsed = _clockAnim.getElapsedTime();
   int timeAnim = elapsed.asMilliseconds();
   sf::Time elapsed2 = _clockMove.getElapsedTime();
   int timeMove = elapsed2.asMilliseconds();

   int delaiAnim=100;
   bool fini=false;

   if(lancerAttaque!=-1)
   {
         bool enAttaque=false;
```

```
                    if(lancerAttaque==1)
                            enAttaque=sautPunch(champEnnemi,degats,energie);
            else if(lancerAttaque==2)
                    enAttaque=sautKick(champEnnemi,degats,energie);

            if(enAttaque)
            {
                    lancerAttaque=-1;
                    if(_cptSauter<4)
                            _cptSauter=7-_cptSauter;
            }

}else
{
  if(timeMove > 10){
  _vsaut += v_grav;
   _posY += _vsaut;
  _clockMove.restart();
  }
  if(timeAnim > delaiAnim){
    if(_cptSauter < 6)
      _cptSauter++;
    _clockAnim.restart();
  }
  switch (_cptSauter)
  {
  case 0:
    if(timeAnim > 20){
      _cptSauter ++;
      _clockAnim.restart();
```

```
        if (!_effetSonore.openFromFile("musique/Ryu/saut.ogg"))
            std::cout<<"erreur musique";
        _effetSonore.play();
    }
    setSprite(651,829,63,89);
    break;
case 1:
    setSprite(714,818,70,108);
    break;
case 2:
    setSprite(791,775,64,88);
    break;
case 3:
    setSprite(861,748,61,70);
    break;
case 4:
    setSprite(925,739,61,65);
    break;
case 5:
    setSprite(1000,750,64,86);
    break;
case 6:
    setSprite(1071,765,62,115);
    if(_posY + _tailleSprite.y + _vsaut >= _scene.getBottom()){
        _cptSauter ++;
    }
    break;
case 7:
    _cptSauter =0;
    setSprite(2,433,66,98);
    _posY=_scene.getBottom()-_tailleSprite.y;
```

```
            _vsaut = -40;
            fini = true;
            break;
        }
    }
            _hurtbox.setSize(sf::Vector2f(_tailleSprite.x*0.6,_tailleSprite.y*0.8));
            _hurtbox.setPosition(_posX+_tailleSprite.x*0.2*_orientation,_posY+_tailleSprite.y*0.1);

            _sprite.setPosition(_posX,_posY);
    keepInWalls();
    return fini;
}


bool Ryu::sauterAvant(Personnage& champEnnemi)
{
    float v_grav = 1.7;
            _cptStatic=0;
            sf::Time elapsed = _clockAnim.getElapsedTime();
    int timeAnim = elapsed.asMilliseconds();
    sf::Time elapsed2 = _clockMove.getElapsedTime();
    int timeMove = elapsed2.asMilliseconds();

    int delaiAnim=70;
    int deplacementX=15;
    bool fini=false;

    if(timeMove > 10){
        _vsaut += v_grav;
        _posY += _vsaut;
        collisionsaut(champEnnemi,deplacementX);
    _posX += deplacementX*_orientation;;
    _clockMove.restart();
}
if(timeAnim > delaiAnim){
    if(_cptSauter < 8 && _cptSauter != 2){
        if(_cptSauter == 1){
            _posX += 25*_orientation;
            _posY -= 5;
        }
        _cptSauter++;
        _clockAnim.restart();
    }

    if(_cptSauter == 4){
        _posX -= 100*_orientation;;
        _posY += 50;
    }
    else if(_cptSauter == 5){
        _posX += 100*_orientation;;
        _posY -= 50;
    }
    else if(_cptSauter == 6){
        _posX -= 140*_orientation;;
        _posY += 100;
    }
    else if(_cptSauter == 7){
        _posX += 70*_orientation;;
        _posY -= 70;
    }
}
switch (_cptSauter)
```

```
{
   case 0:
      if(timeAnim > 20){
          _cptSauter ++;
          _clockAnim.restart();
      }
   setSprite(651,829,63,89);
   _hurtbox.setSize(sf::Vector2f(_tailleSprite.x*0.6,_tailleSprite.y*0.8));
   _hurtbox.setPosition(_posX+_tailleSprite.x*0.2*_orientation,_posY+_tailleSprite.y*0.1);

   if (!_effetSonore.openFromFile("musique/Ryu/saut.ogg"))
      std::cout<<"erreur musique";
   _effetSonore.play();
   break;
case 1:
   setSprite(714,818,70,108);
   _hurtbox.setSize(sf::Vector2f(_tailleSprite.x*0.6,_tailleSprite.y*0.8));
   _hurtbox.setPosition(_posX+_tailleSprite.x*0.2*_orientation,_posY+_tailleSprite.y*0.1);
   break;
case 2:
   setSprite(1348,794,62,106);
   _hurtbox.setSize(sf::Vector2f(_tailleSprite.x*0.6,_tailleSprite.y*0.8));
   _hurtbox.setPosition(_posX+_tailleSprite.x*0.2*_orientation,_posY+_tailleSprite.y*0.1);
      if(timeAnim > 200){
          _cptSauter ++;
          _clockAnim.restart();
      }
   break;
case 3:
   setSprite(1488,927,61,90);
   _hurtbox.setSize(sf::Vector2f(_tailleSprite.x*0.6,_tailleSprite.y*0.8));
   _hurtbox.setPosition(_posX+_tailleSprite.x*0.2*_orientation,_posY+_tailleSprite.y*0.1);
   break;
case 4:
   setSprite(1410,759,93,47);
   _hurtbox.setSize(sf::Vector2f(_tailleSprite.x*0.6,_tailleSprite.y*0.8));
   _hurtbox.setPosition(_posX+_tailleSprite.x*0.2*_orientation,_posY+_tailleSprite.y*0.1);
   break;
case 5:
   setSprite(1510,736,55,78);
   _hurtbox.setSize(sf::Vector2f(_tailleSprite.x*0.6,_tailleSprite.y*0.8));
   _hurtbox.setPosition(_posX+_tailleSprite.x*0.2*_orientation,_posY+_tailleSprite.y*0.1);
   break;
case 6:
   setSprite(1565,768,123,52);
   _hurtbox.setSize(sf::Vector2f(_tailleSprite.x*0.4,_tailleSprite.y*0.8));
   _hurtbox.setPosition(_posX+_tailleSprite.x*0.3*_orientation,_posY+_tailleSprite.y*0.1);
   break;
case 7:
   setSprite(1689,747,72,94);
   _hurtbox.setSize(sf::Vector2f(_tailleSprite.x*0.6,_tailleSprite.y*0.8));
   _hurtbox.setPosition(_posX+_tailleSprite.x*0.4*_orientation,_posY+_tailleSprite.y*0.1);
   break;
case 8:
   setSprite(1071,765,62,115);
   _hurtbox.setSize(sf::Vector2f(_tailleSprite.x*0.6,_tailleSprite.y*0.8));
   _hurtbox.setPosition(_posX+_tailleSprite.x*0.2*_orientation,_posY+_tailleSprite.y*0.1);
   if(_posY + _tailleSprite.y + _vsaut >= _scene.getBottom()){
      _cptSauter ++;
   }
   break;
case 9:
        _cptSauter =0;
        setSprite(2,433,66,98);
        _hurtbox.setSize(sf::Vector2f(_tailleSprite.x*0.6,_tailleSprite.y*0.8));
        _hurtbox.setPosition(_posX+_tailleSprite.x*0.2*_orientation,_posY+_tailleSprite.y*0.1);
        _posY=_scene.getBottom()-_tailleSprite.y;
        _posX += 45*_orientation;
        _vsaut = -40;
        rotate(champEnnemi);
        fini =  true;
        break;
    }
            _sprite.setPosition(_posX,_posY);
    keepInWalls();
    return fini;
}

bool Ryu::sauterArriere(Personnage& champEnnemi)
{
            float v_grav = 1.7;
            _cptStatic=0;
            sf::Time elapsed = _clockAnim.getElapsedTime();
    int timeAnim = elapsed.asMilliseconds();
    sf::Time elapsed2 = _clockMove.getElapsedTime();
    int timeMove = elapsed2.asMilliseconds();

    int delaiAnim=50;
    int deplacementX=15;
    bool fini=false;

    if(timeMove > 10){
        _vsaut += v_grav;
    _posY += _vsaut;
    collisionsaut(champEnnemi,deplacementX);
    _posX -= deplacementX*_orientation;;
    _clockMove.restart();
}

if(timeAnim > delaiAnim){
    if(_cptSauter < 8){
        if(_cptSauter == 1){
            _posX += 25*_orientation;
            _posY -= 5;
        }
        _cptSauter++;
        _clockAnim.restart();
    }
    switch(_cptSauter)
    {
    case 2:
        _posX -= 100*_orientation;
        break;
    case 3:
        _posX -= 50*_orientation;
        _posY += 50;
        break;
    case 4:
        _posX += 100*_orientation;
        _posY -= 50;
        break;
    case 5:
        _posX -= 50*_orientation;
        _posY += 50;
        break;
    case 6:
        _posX += 75*_orientation;
        _posY -= 50;
        break;
    case 7:
        _posY -= 50;
        break;
    }
}
switch (_cptSauter)
{
case 0:
    if(timeAnim > 20){
         _cptSauter ++;
         _clockAnim.restart();

         if (!_effetSonore.openFromFile("musique/Ryu/saut.ogg"))
            std::cout<<"erreur musique";
        _effetSonore.play();
    }
    setSprite(651,829,63,89);
    break;
case 1:
    setSprite(791,775,64,88);
    break;
case 2:
    setSprite(1689,747,72,94);
    break;
case 3:
    setSprite(1565,768,123,52);
```

```
   break;
case 4:
   setSprite(1510,736,55,78);
   break;
case 5:
   setSprite(1410,759,93,47);
   break;
case 6:
   setSprite(1488,927,61,90);
   break;
case 7:
   setSprite(518,982,61,107);
   break;
case 8:
   setSprite(1071,765,62,115);
   if(_posY + _tailleSprite.y + _vsaut >= _scene.getBottom()){
      _cptSauter ++;
   }
   break;
case 9:
   _cptSauter =0;
   setSprite(2,433,66,98);
   _posY=_scene.getBottom()-_tailleSprite.y;
   _vsaut = -40;
   rotate(champEnnemi);
   fini = true;
   break;
}
_hurtbox.setSize(sf::Vector2f(_tailleSprite.x*0.6,_tailleSprite.y*0.8));
        _hurtbox.setPosition(_posX+_tailleSprite.x*0.2*_orientation,_posY+_tailleSprite.y*0.1);
```

```
            _sprite.setPosition(_posX,_posY);
   keepInWalls();
   return fini;
}

void Ryu::accroupi(bool garde)
{
            _cptStatic=0;
            sf::Time elapsed = _clockAnim.getElapsedTime();
   int timeAnim = elapsed.asMilliseconds();
   int delaiAnim=35;
   if(_cptAccroupi==0)
   {
            if(timeAnim>delaiAnim)
            {
                        _clockAnim.restart();
                        _cptAccroupi++;
                        setSprite(73,555,62,72);

                                    _hurtbox.setSize(sf::Vector2f(_tailleSprite.x*0.9,_tailleSprite.y));
                                    _hurtbox.setPosition(_posX,_posY);
            }
   }else
   {
            if(timeAnim>delaiAnim)
            {
                        _clockAnim.restart();
                        if(garde==true)
                                    setSprite(212,4685,64,65);
                        else
                        {
```

```
                                    setSprite(142,562,62,65);
                                    _hurtbox.setSize(sf::Vector2f(_tailleSprite.x*0.8,_tailleSprite.y*0.9));
                                    _hurtbox.setPosition(_posX+_tailleSprite.x*0.1,_posY+_tailleSprite.y*0.1);
                            }
                    }
    }
    _posY=_scene.getBottom()-_tailleSprite.y;
    _sprite.setPosition(_posX,_posY);
}

bool Ryu::punch(Personnage& champEnnemi,int* degats,int& energie)
{
            _cptStatic=0;
            sf::Time elapsed = _clockAnim.getElapsedTime();
    int timeAnim = elapsed.asMilliseconds();
    int delaiAnim=60;
    bool fini=false;

    if(timeAnim > delaiAnim)
    {
                    switch (_cptAction)
                    {
                    case 0:
                        _cptAction ++;
                        _clockAnim.restart();
                        setSprite(3,1319,74,94);

                        _hurtbox.setSize(sf::Vector2f(_tailleSprite.x*0.6,_tailleSprite.y*0.9));

            _hurtbox.setPosition(_posX+_tailleSprite.x*0.2*_orientation,_posY+_tailleSprite.y*0.1);
```

```
if (!_effetSonore.openFromFile("musique/Ryu/coup_poing.ogg"))
   std::cout<<"erreur musique";
_effetSonore.play();
                        break;
             case 1:
                _cptAction ++;
                _clockAnim.restart();
                setSprite(80,1318,102,95);
                _hitbox.setSize(sf::Vector2f(40*_scale,20*_scale));
                _hitbox.setPosition(_posX+60*_scale*_orientation,_posY+10*_scale);
                _spriteHitSpark.setPosition(_posX+60*_scale*_orientation,_posY);
                        break;
             case 2:
                _cptAction++;
                _clockAnim.restart();
                setSprite(185,1319,74,94);


                _hitbox.setSize(sf::Vector2f(0,0));
                        break;
             case 3:
                        _cptAction=0;
                        _clockAnim.restart();
                        setSprite(2,433,66,98);
                        fini=true;
                        break;
             }
  }


  if(collisionCoup(champEnnemi))
  {
     if(_peutHitSpark)
```

```
        _hitSpark = true;
                *degats=5;
                energie+=10;

                if(champEnnemi.getPosX()==_scene.getRightLimit())
                    _posX-=25*_scale*_orientation;
      else if(champEnnemi.getPosX()<=5)
        _posX+=25*_scale;
            }

            _posY=_scene.getBottom()-_tailleSprite.y;
    _sprite.setPosition(_posX,_posY);
            keepInWalls();
            return fini;
}

bool Ryu::sautPunch(Personnage& champEnnemi,int* degats,int& energie)
{
            _cptStatic=0;
            sf::Time elapsed = _clockAnim.getElapsedTime();
    int timeAnim = elapsed.asMilliseconds();
    int delaiAnim=60,deplacement=125;
    bool fini=false;

    if(timeAnim>delaiAnim)
    {
            switch(_cptAction)
            {
            case 0:
                _cptAction ++;
                  _clockAnim.restart();
                setSprite(150,1794,55,73);


                _hurtbox.setSize(sf::Vector2f(_tailleSprite.x*0.6,_tailleSprite.y*0.9));

  _hurtbox.setPosition(_posX+_tailleSprite.x*0.2*_orientation,_posY+_tailleSprite.y*0.1);


if (!_effetSonore.openFromFile("musique/Ryu/coup_poing.ogg"))
   std::cout<<"erreur musique";
_effetSonore.play();
                        break;
            case 1:
                _cptAction ++;
                _clockAnim.restart();
                setSprite(206,1794,74,79);
                        break;
            case 2:
                        _cptAction++;
                        _clockAnim.restart();
                        setSprite(281,1794,98,72);


                        _hitbox.setSize(sf::Vector2f(_tailleSprite.x*0.4,_tailleSprite.y*0.5));

  _hitbox.setPosition(_posX+_tailleSprite.x*0.6*_orientation,_posY+_tailleSprite.y*0.2);
                        break;
            case 3:
                _cptAction=0;
                _clockAnim.restart();
                setSprite(281,1794,98,72);
                fini=true;


                _hitbox.setSize(sf::Vector2f(0,0));
                        break;
                }
            }

            if(collisionCoup(champEnnemi))
            {
                        *degats=5;
                        energie+=10;

                        if(champEnnemi.getPosX()==_scene.getRightLimit())
                                    _posX-=25*_scale*_orientation;
        else if(champEnnemi.getPosX()<=5)
           _posX+=25*_scale;
            }

    keepInWalls();
    return fini;
}

bool Ryu::punchSP(sf::Sprite& inutile,Personnage& champEnnemi, int* degats,int& energie)
{
            if(energie<20)
            {
                        energie=-100;
                        return true;
            }

            _cptStatic=0;
            sf::Time elapsed = _clockAnim.getElapsedTime();
    int timeAnim = elapsed.asMilliseconds();
    int delaiAnim=50,deplacement=_tailleSprite.x/2;
    bool fini=false;
    if(timeAnim > delaiAnim)
    {
                        collisionsaut(champEnnemi,deplacement);

                        switch (_cptAction)
                        {
                        case 0:
                            _cptAction ++;
                            _clockAnim.restart();
                            setSprite(8,3795,66,86);

                            _posY=_scene.getBottom()-_tailleSprite.y;
                            if (!_effetSonore.openFromFile("musique/Ryu/shoryuken.ogg"))
                                std::cout<<"erreur musique";
                                    _effetSonore.play();
                                    break;
                        case 1:
                            _cptAction ++;
                            _clockAnim.restart();
                            setSprite(83,3791,78,90);

                            _posY=_scene.getBottom()-_tailleSprite.y;

                            _hitbox.setSize(sf::Vector2f(_tailleSprite.x*0.2,_tailleSprite.y*0.4));
_hitbox.setPosition(_posX+_tailleSprite.x*0.8*_orientation,_posY+_tailleSprite.y*0.1);

_spriteHitSpark.setPosition(_posX+_tailleSprite.x*0.8*_orientation,_posY+_tailleSprite.y*0.1);
                                    break;
                        case 2:
```

```
   _cptAction ++;
   _clockAnim.restart();
   setSprite(176,3754,62,129);

   _posX+=deplacement*_orientation;
   _posY=_scene.getBottom()-_tailleSprite.y;

   _hitbox.setSize(sf::Vector2f(_tailleSprite.x*0.4,_tailleSprite.y*0.5));
   _hitbox.setPosition(_posX+_tailleSprite.x*0.6*_orientation,_posY);
        break;
case 3:
   _cptAction ++;
   _clockAnim.restart();
   setSprite(244,3686,55,121);

   _posX+=deplacement*_orientation;
   _posY-=_tailleSprite.y/2;

   _hitbox.setSize(sf::Vector2f(_tailleSprite.x*0.4,_tailleSprite.y*0.5));
   _hitbox.setPosition(_posX+_tailleSprite.x*0.6*_orientation,_posY);
        break;
case 4:
        if(timeAnim>delaiAnim*2)
        {
                _cptAction++;
           _clockAnim.restart();
           setSprite(315,3697,61,117);

           _posX+=deplacement/2*_orientation;
           _posY+=_tailleSprite.y/10;
```

```
                    _hitbox.setSize(sf::Vector2f(0,0));

                  }

                  break;

          case 5:

             _cptAction++;

             _clockAnim.restart();

             setSprite(380,3779,63,99);


             _posY=_scene.getBottom()-_tailleSprite.y;

                  break;

          case 6:

             _cptAction=0;

             _clockAnim.restart();

             setSprite(2,433,66,98);

             fini=true;

             energie-=25;


             _posY=_scene.getBottom()-_tailleSprite.y;

             break;

          }


          if( (_orientation==1 && _posX+_tailleSprite.x >= champEnnemi.getPosX()-
champEnnemi.getHurtbox().getGlobalBounds().width)

        || (_orientation==-1 && _posX-_tailleSprite.x <=
champEnnemi.getPosX()+champEnnemi.getHurtbox().getGlobalBounds().width) )

     {

        _posX=champEnnemi.getPosX()-
(champEnnemi.getHurtbox().getGlobalBounds().width+_tailleSprite.x+deplacement)*_orientation;

     }


          _sprite.setPosition(_posX,_posY);
```

```
            _hurtbox.setPosition(_posX+_tailleSprite.x*0.2*_orientation,_posY+_tailleSprite.y*0.1);
                        _hurtbox.setSize(sf::Vector2f(_tailleSprite.x*0.6,_tailleSprite.y*0.9));
            }

            if(collisionCoup(champEnnemi))
            {
               if(_peutHitSpark)
          _hitSpark = true;
                        *degats=10;

                        if(champEnnemi.getPosX()==_scene.getRightLimit())
                                    _posX-=25*_scale*_orientation;
       else if(champEnnemi.getPosX()<=5)
          _posX+=25*_scale;
            }

            keepInWalls();
            return fini;
}

bool Ryu::kick(Personnage& champEnnemi,int* degats,int& energie)
{
            _cptStatic=0;
            sf::Time elapsed = _clockAnim.getElapsedTime();
   int timeAnim = elapsed.asMilliseconds();
   int delaiAnim=70;
   bool fini=false;

   if(timeAnim > delaiAnim)
   {
        switch (_cptAction)
        {
        case 0:
          _cptAction ++;
          _clockAnim.restart();
          setSprite(497,2559,67,94);

          _hurtbox.setSize(sf::Vector2f(_tailleSprite.x*0.6,_tailleSprite.y*0.9));

 _hurtbox.setPosition(_posX+_tailleSprite.x*0.2*_orientation,_posY+_tailleSprite.y*0.1);

if (!_effetSonore.openFromFile("musique/Ryu/coup_pied.ogg"))
   std::cout<<"erreur musique";
_effetSonore.play();
                break;
        case 1:
          _cptAction ++;
          _clockAnim.restart();
          setSprite(566,2559,65,94);
                break;
        case 2:
          _cptAction ++;
          _clockAnim.restart();
          setSprite(656,2563,118,90);

        _hitbox.setSize(sf::Vector2f(80*_scale,22*_scale));
          _hitbox.setPosition(_posX+36*_scale*_orientation,_posY);
          _spriteHitSpark.setPosition(_posX+80*_scale*_orientation,_posY);
                break;
        case 3:
          _cptAction ++;
```

```
                              _clockAnim.restart();
                              setSprite(775,2559,65,94);


                              _hitbox.setSize(sf::Vector2f(0,0));
                                   break;
                            case 4:
                              _cptAction =0;
                              _clockAnim.restart();
                              setSprite(867,2559,65,94);
                              fini=true;
                                   break;
                            }
                     }


                     if(collisionCoup(champEnnemi))
                     {
                       if(_peutHitSpark)
                   _hitSpark = true;
                            *degats=7;
                            energie+=10;


                            if(champEnnemi.getPosX()==_scene.getRightLimit())
                                   _posX-=25*_scale*_orientation;
                 else if(champEnnemi.getPosX()<=5)
                   _posX+=25*_scale;
                     }
                     _posY=_scene.getBottom()-_tailleSprite.y;
               _sprite.setPosition(_posX,_posY);
                     keepInWalls();
                     return fini;
              }
```

```
bool Ryu::sautKick(Personnage& champEnnemi,int* degats,int& energie)
{
        sf::Time elapsed = _clockAnim.getElapsedTime();
  int timeAnim = elapsed.asMilliseconds();
  int delaiAnim=60,deplacement=_scene.getBottom()/6;
  bool fini=false;

        if(timeAnim>delaiAnim)
  {
          switch(_cptAction)
          {
          case 0:
                _cptAction ++;
                  _clockAnim.restart();
                  setSprite(228,3022,58,117);

                  _hurtbox.setSize(sf::Vector2f(_tailleSprite.x*0.6,_tailleSprite.y*0.9));

        _hurtbox.setPosition(_posX+_tailleSprite.x*0.2*_orientation,_posY+_tailleSprite.y*0.1);

       if (!_effetSonore.openFromFile("musique/Ryu/coup_pied.ogg"))
         std::cout<<"erreur musique";
       _effetSonore.play();
                      break;
          case 1:
                _cptAction ++;
                  _clockAnim.restart();
                  setSprite(298,3013,59,98);
                      break;
                case 2:
                  _cptAction ++;

                  _clockAnim.restart();

                  setSprite(367,3020,92,107);


                  _hitbox.setSize(sf::Vector2f(_tailleSprite.x*0.6,_tailleSprite.y*0.2));

_hitbox.setPosition(_posX+_tailleSprite.x*0.4*_orientation,_posY+_tailleSprite.y*0.35);

                      break;

                case 3:

                      _cptAction =0;

                      _clockAnim.restart();

                      setSprite(472,3036,61,102);

                      fini=true;


                      _hitbox.setSize(sf::Vector2f(0,0));

                      break;

          }

        }


        if(collisionCoup(champEnnemi))

        {

                *degats=7;

                energie+=10;


                if(champEnnemi.getPosX()==_scene.getRightLimit())

                      _posX-=25*_scale*_orientation;

  else if(champEnnemi.getPosX()<=5)

       _posX+=25*_scale;

        }


  keepInWalls();
```

```
    return fini;
}

bool Ryu::kickSP(Personnage& champEnnemi, int* degats,int& energie)
{
            if(energie<20)
            {
                        energie=-100;
                        return true;
            }

            _cptStatic=0;
            sf::Time elapsed = _clockAnim.getElapsedTime();
    int timeAnim = elapsed.asMilliseconds();
    int delaiAnim=70,deplacementY=_scene.getBottom()/7,deplacementX=50*_orientation;
    bool fini=false;

    if(timeAnim > delaiAnim)
    {
                        switch (_cptAction)
                        {
                        case 0:
                            _cptAction ++;
                            _clockAnim.restart();
                            setSprite(1,3039,71,110);
                            _posY=_scene.getBottom()-_tailleSprite.y;

          if (!_effetSonore.openFromFile("musique/Ryu/tatsumaki.ogg"))
             std::cout<<"erreur musique";
          _effetSonore.play();
                                    break;
```

```
case 1:
   _cptAction ++;
   _clockAnim.restart();
   setSprite(75,3036,61,87);
   _posY-=deplacementY;
           break;
case 2:
   _cptAction ++;
   _clockAnim.restart();
   setSprite(148,3025,54,68);
   _posY-=deplacementY;
           break;
case 3:
   _cptAction ++;
   _clockAnim.restart();
   setSprite(228,3022,58,77);
   _posY-=deplacementY/2;
           break;
case 4:
           _cptAction++;
           _clockAnim.restart();
           setSprite(298,3013,59,98);
           _posY+=deplacementY;
           break;
case 5:
           _cptAction++;
   _clockAnim.restart();
   setSprite(366,3020,93,108);
   _posY+=deplacementY;

   _hitbox.setSize(sf::Vector2f(_tailleSprite.x*0.6,_tailleSprite.y*0.3));
```

```
            _hitbox.setPosition(_posX+_tailleSprite.x*0.4*_orientation,_posY+_tailleSprite.y*0.4);

            _spriteHitSpark.setPosition(_posX+_tailleSprite.x*0.4*_orientation,_posY+_tailleSprite.y*0.4
);
                        break;
                case 6:
                   _cptAction++;
                   _clockAnim.restart();
                   setSprite(472,3036,61,102);
                   _posY=_scene.getBottom()-_tailleSprite.y;
                   _hitbox.setSize(sf::Vector2f(0,0));
                        break;
                case 7:
                   _cptAction=0;
                   _clockAnim.restart();
                   setSprite(538,3057,63,89);
                   fini=true;
                   energie-=25;
                   _posY=_scene.getBottom()-_tailleSprite.y;
                        break;
                }
                _posX+=deplacementX;
                _sprite.setPosition(_posX,_posY);
            }

            if(collisionCoup(champEnnemi))
            {
               if(_peutHitSpark)
           _hitSpark = true;

                *degats=10;
```

```
                    if(champEnnemi.getPosX()==_scene.getRightLimit())
                                _posX-=25*_scale*_orientation;
        else if(champEnnemi.getPosX()<=5)
           _posX+=25*_scale;
            }


            keepInWalls();
            return fini;
}


bool Ryu::SP(sf::Sprite& bouleFeu,Personnage& champEnnemi,int* degats,int& energie)
{
   if(energie<50)
            {
                    energie=-100;
                    return true;
            }


            _cptStatic=0;
            sf::Time elapsed = _clockAnim.getElapsedTime();
   int timeAnim = elapsed.asMilliseconds();
   int delaiAnim=70;
   bool fini=false;


   if(timeAnim > delaiAnim)
   {
              switch (_cptAction)
                     {
                    case 0:
                                if (!_effetSonore.openFromFile("musique/Ryu/hadouken.ogg"))
```

```
                  std::cout<<"erreur musique";
                _effetSonore.play();


                _cptAction++;
                _clockAnim.restart();
                setSprite(10,3493,74,90);


                bouleFeu.setTexture(_texture);
bouleFeu.setTextureRect(sf::IntRect(0,0,0,0));
                     bouleFeu.setPosition(bouleFeu.getPosition().x+20*_orientation,_posY);
                     break;
              case 1:
                _cptAction ++;
                _clockAnim.restart();
                setSprite(890,3520,91,90);
                _posX-=1*_scale*_orientation;
                     break;
              case 2:
                _cptAction ++;
                _clockAnim.restart();
                setSprite(986,3520,111,90);
                _posX-=20*_scale*_orientation;
                     break;
              case 3:
                _cptAction ++;
                _clockAnim.restart();
                setSprite(742,3630,115,90);
                _posX-=4*_scale*_orientation;
                break;
              case 4:
                _cptAction ++;
                _clockAnim.restart();
                setSprite(993,3630,117,90);
                _posX-=2*_scale*_orientation;
                break;
            case 5:
                _cptAction ++;
                _clockAnim.restart();
                setSprite(1124,3636,98,85);
                _posX+=20*_scale*_orientation;
                break;
            case 6:
                _cptAction ++;
                _clockAnim.restart();
                setSprite(1229,3635,119,86);
                _posX+=2*_scale*_orientation;

                    bouleFeu.setTextureRect(sf::IntRect(1130,5745,240,171));
                    bouleFeu.setScale(_orientation,1);
                    bouleFeu.setPosition(_posX+(_tailleSprite.x*0.7*_orientation),_posY);
                    break;
            }
    }

sf::Time elapsedEffet = _clockEffet.getElapsedTime();
int timeEffet = elapsedEffet.asMilliseconds();
int delaiEffet=10;

if(timeEffet>delaiEffet)
{

    if(_cptAction>6 && _cptAction<11)
```

```
            {
                        setSprite(400,3605,119,79);
                        _cptAction ++;
                        bouleFeu.setTextureRect(sf::IntRect(1426,5744,231,181));
                        bouleFeu.setPosition(bouleFeu.getPosition().x+20*_orientation,_posY);
            }else if(_cptAction>10 && _cptAction<15)
            {
                        _cptAction ++;
                        bouleFeu.setTextureRect(sf::IntRect(1713,5742,234,176));
                        bouleFeu.setPosition(bouleFeu.getPosition().x+20*_orientation,_posY);
            }else if(_cptAction>14 && _cptAction<19)
            {
                        _cptAction ++;
                        bouleFeu.setTextureRect(sf::IntRect(2003,5738,234,178));
                        bouleFeu.setPosition(bouleFeu.getPosition().x+20*_orientation,_posY);
            }else if(_cptAction>18 && _cptAction<23)
            {
                        _cptAction ++;
                        bouleFeu.setTextureRect(sf::IntRect(1130,5745,240,171));
                        bouleFeu.setPosition(bouleFeu.getPosition().x+20*_orientation,_posY);
                        if(_cptAction==23)
                                    _cptAction=7;
            }
    }


    if(_cptAction>6)
    {
       if( (_orientation==1 && bouleFeu.getPosition().x>=_scene.getRightLimit()) || (_orientation==-1
&& bouleFeu.getPosition().x<=_scene.getLeftLimit()) )
       {
          bouleFeu.setTextureRect(sf::IntRect(0,0,0,0));
```

```
            fini=true;
            energie-=50;
            _cptAction=0;
            cout<<endl<<"ici\t"<<bouleFeu.getPosition().x<<endl;
        }

        if(collisionCoup(champEnnemi))
        {
            *degats=30;
            bouleFeu.setTextureRect(sf::IntRect(0,0,0,0));
            fini=true;
            energie-=50;
            _cptAction=0;
        }
    }



            _hurtbox.setSize(sf::Vector2f(_tailleSprite.x*0.6,_tailleSprite.y*0.9));
            _hurtbox.setPosition(_posX+_tailleSprite.x*0.2*_orientation,_posY+_tailleSprite.y*0.1);


_hitbox.setSize(sf::Vector2f(bouleFeu.getGlobalBounds().width,bouleFeu.getGlobalBounds().height))
;
            _hitbox.setPosition(bouleFeu.getPosition().x,bouleFeu.getPosition().y);



            _posY=_scene.getBottom()-_tailleSprite.y;
    _sprite.setPosition(_posX,_posY);
            keepInWalls();
            return fini;
}
```