```
#include "View.h"
#include "Model.h"
#include "GraphicElement.h"
#include "SlidingBackground.h"
#include "AnimatedGraphicElement.h"
#include "Bonus.h"
#include <sstream>
#include <iostream>
#include <SFML/Audio.hpp>

using namespace std;

//======================================
// Constructeur
//======================================
View::View(int w, int h): _w(w), _h(h){
    _window = new sf::RenderWindow(sf::VideoMode(w, h, 32), "Runner", sf::Style:
:Close);
    _window->setFramerateLimit(60);

    if (!_backgroundTexture.loadFromFile(CITY1_IMAGE)) // On regarde si on peut
charger l'image correctement
        std::cerr << "ERROR when loading image file: " << CITY1_IMAGE << std::en
dl;//sinon on affiche ce message.

    else { //si Ã§a marche  on initialise un slidingBackground
        _backgroundSprite = new SlidingBackground(_backgroundTexture, 800, 600,_
w, _h, 2);
    }

    if (!_backgroundTexture2.loadFromFile(CITY2_IMAGE)) // On regarde si on peut
 charger l'image correctement
        std::cerr << "ERROR when loading image file: " << CITY2_IMAGE << std::en
dl;//sinon on affiche ce message.

    else { //si Ã§a marche on initialise un SlidingBackground
        _background2 = new SlidingBackground(_backgroundTexture2, 800, 600,_w, _
h, 4);

    }

    if (!_backgroundTexture3.loadFromFile(CITY3_IMAGE)) // On regarde si on peut
 charger l'image correctement
        std::cerr << "ERROR when loading image file: " << CITY3_IMAGE << std::en
dl;//sinon on affiche ce message.

    else { //si Ã§a marche on initialise un SlidingBackground
        _background3 = new SlidingBackground(_backgroundTexture3, 800, 600,_w, _
h, 2);
    }

    if (!_backgroundTexture4.loadFromFile(CITY4_IMAGE)) // On regarde si on peut
 charger l'image correctement
        std::cerr << "ERROR when loading image file: " << CITY4_IMAGE << std::en
dl;//sinon on affiche ce message.

    else { //si Ã§a marche on initialise un SlidingBackground
        _background4 = new SlidingBackground(_backgroundTexture4, 800, 600,_w, _
h, 4);
    }

    if (!_ballTexture.loadFromFile(BALLS_IMAGE)) // On regarde si on peut charge
r l'image correctement
        std::cerr << "ERROR when loading image file: " << BALL_IMAGE << std::end
l;//sinon on affiche ce message.
    else //si Ã§a marche on initialise un AnimatedGraphicElement pour la balle
    {
        vector<sf::IntRect> rectlect;
        RectLecture(rectlect);
```



```
        _ballSprite = new AnimatedGraphicElement(rectlect, _ballTexture, 30, 30,
 _w, _h);
    }

    if (!_font.loadFromFile(TEXT) || !_font2.loadFromFile(TEXT2)){ // On regarde
 si on peut charger l'image correctement
        std::cerr << "ERROR when loading image file: " << TEXT << std::endl;//si
non on affiche ce message.
        std::cerr << "ERROR when loading image file: " << TEXT2 << std::endl;}//
sinon on affiche ce message.
    else //si Ã§a marche on initialise 3 texte liÃ©s au score et la vie
    {
        _text.setFont(_font);
        _text.setString("Score : ");
        _text.setPosition(450, 535);
        _text.setCharacterSize(50);
        _text.setColor(sf::Color::Black);
        _text.setStyle(sf::Text::Bold);

        _tscore.setFont(_font2);
        _tscore.setPosition(605, 545);
        _tscore.setCharacterSize(40);
        _tscore.setColor(sf::Color::Black);
        _tscore.setStyle(sf::Text::Bold);

        _tvie.setFont(_font);
        _tvie.setString("Vie : ");
        _tvie.setPosition(50, 535);
        _tvie.setCharacterSize(50);
        _tvie.setColor(sf::Color::Black);
        _tvie.setStyle(sf::Text::Bold);
    }

    if (!_obstacleTexture.loadFromFile(OBSTACLE)) // On regarde si on peut charg
er l'image correctement
        std::cerr << "ERROR when loading image file: " << OBSTACLE << std::endl;
//sinon on affiche ce message.
    else //si Ã§a marche on initialise un Obstacle
    {
        _obstacleSprite = new Obstacles(100, 50, 20, 20, 0, 3);

    }
    if (!_logoTexture.loadFromFile(LOGO)) // On regarde si on peut charger l'ima
ge correctement
        std::cerr << "ERROR when loading image file: " << LOGO << std::endl;//si
non on affiche ce message.

    else { //si Ã§a marche on initialise un GraphicElement
        _logo = new GraphicElement(_logoTexture, 200, 100, 0, 0);
    }
    if (!_rulesTexture.loadFromFile(RULES)) // On regarde si on peut charger l'i
mage correctement
        std::cerr << "ERROR when loading image file: " << RULES << std::endl;//s
inon on affiche ce message.

    else { //si Ã§a marche on initialise un GraphicElement
        _rules = new GraphicElement(_rulesTexture, 0, 0, 800, 600);
    }
    if (!_menuTexture.loadFromFile(MENU)) // On regarde si on peut charger l'ima
ge correctement
        std::cerr << "ERROR when loading image file: " << MENU << std::endl;//si
non on affiche ce message.

    else { //si Ã§a marche on initialise un GraphicElement
        _menu = new GraphicElement(_menuTexture, 0, 0, 800, 600);
    }
    if (!_ennemies.loadFromFile(OBSTACLE1)) // On regarde si on peut charger l'i
mage correctement
        std::cerr << "ERROR when loading image file: " << OBSTACLE1 << std::endl
```

```
;//sinon on affiche ce message.

    else { //si Ã§a marche on initialise un Obstacle
        _ennemiesSprite = new Obstacles(100, 150, 0, 0, 1, 3);
    }
    if (!_ennemies2.loadFromFile(OBSTACLE2)) // On regarde si on peut charger l'
image correctement
        std::cerr << "ERROR when loading image file: " << OBSTACLE2 << std::endl
;//sinon on affiche ce message.

    else { //si Ã§a marche on initialise un Obstacle
        _ennemies2Sprite = new Obstacles(100, 150, 0, 0, 2, 3);
    }

    if (!_pieceTexture.loadFromFile(PIECES)) // On regarde si on peut charger l'
image correctement
        std::cerr << "ERROR when loading image file: " << PIECES << std::endl;//
sinon on affiche ce message.

    else { //si Ã§a marche on initialise un Bonus
        _piece = new Bonus(50, 50, 10,10, 4, 3);
    }

    if (!_bonuslifeTexture.loadFromFile(BONUSLIFE)) // On regarde si on peut cha
rger l'image correctement
        std::cerr << "ERROR when loading image file: " << BONUSLIFE << std::endl
;//sinon on affiche ce message.

    else { //si Ã§a marche on initialise un Bonus
        _bonuslife = new Bonus(50, 50, 10,10, 1, 3);
    }

    //On initialise la barre de vie
    barreDeVie.setPosition(sf::Vector2f(150.f, 550.f));
    barreDeVie.setFillColor(sf::Color(183,31,102,255));



}


//=====================================
// Fonction de dessin
//=====================================


void View::draw(){
    _window->clear();
    switch(play) //selon la valeur de play, on dessine:
    {
    case 0: //On dessine le menu
        drawMenu();
        break;
    case 1: //On dessine le jeu
        drawGame();
        break;
    case 2: //On dessine le gameOver
        drawEnd();
        break;
    case 3: //On dessine les regles
        drawRules();
        break;
    }
    _window->display();

}

void View::drawMenu(){
    _window->clear(sf::Color::White);
```



```
    _model->setScore(0); //le score reste Ã  0
    _model->setVie(100); //la vie reste Ã  100

    time = clock.getElapsedTime();
    if(time.asMilliseconds() > 0 && time.asMilliseconds() < 2000)
    { //Si le temps ecoulÃ© est superieur Ã  0 et inferieur Ã  2000
        //On affiche l'ecran d'introduction
        _backgroundSprite->draw(_window);

        sf::Text textDebut;
        textDebut.setPosition(120, 250);
        textDebut.setString("Projet de Programmation");
        textDebut.setCharacterSize(50);
        textDebut.setColor(sf::Color::Black);
        textDebut.setStyle(sf::Text::Bold);
        textDebut.setFont(_font);
        _window->draw(textDebut);
    }
    else{
        //sinon on dessine le menu
        _menu->draw(_window);
    }

}

void View::drawGame(){
    _window->clear();


    _window->setMouseCursorVisible(false); // on rend le curseur invisible

    //Si le temps Ã©coulÃ© est inferieur Ã  20000
    //on affiche le premier fond, sinon on affiche
    //le deuxieme.
    timeB = cbackground.getElapsedTime();
    if(timeB.asMilliseconds() <= 20000){
        _background3->draw(_window);
        _background4->draw(_window);
    }
    else{
        _backgroundSprite->draw(_window);
        _background2->draw(_window);
    }

    timeB = cbackground.restart();

    //On dessine tous les elements du jeu
    _ballSprite->draw(_window);
    _tscore.setString(_model->writeScore(_model->getScore()));
    barreDeVie.setSize(sf::Vector2f(2*_model->getVie(),555));
    _window->draw(barreDeVie);
    _window->draw(_tvie);
    _window->draw(_text);
    _window->draw(_tscore);

    //Dessine les obstacles
    std::map<const MovableElement*, GraphicElement*> ::iterator i;
    for (i = _elementToGraphicElement.begin(); i != _elementToGraphicElement.end
(); i++)
    {
        i->second->draw(_window);
    }
}

void View::drawEnd(){
    _window->clear();
    _window->setMouseCursorVisible(true);

    sf::Text textGO;
```

```
    textGO.setString("Game Over");
    textGO.setPosition(80, 100);
    textGO.setCharacterSize(150);
    textGO.setColor(sf::Color::White);
    textGO.setStyle(sf::Text::Bold);
    textGO.setFont(_font);

    _window->draw(textGO);

}

void View::drawRules(){
    _window->clear();
    _rules->draw(_window);

    _model->setScore(0); //le score reste Ã  0
    _model->setVie(100); //la vie reste Ã  100

    sf::Text text;
    text.setString("Back");
    text.setPosition(680, 530);
    text.setCharacterSize(50);
    text.setColor(sf::Color::Yellow);
    text.setStyle(sf::Text::Bold);
    text.setFont(_font);

    _window->draw(text);

}

//=======================================
// Destructeur
//=======================================
View::~View(){
    if(_window!= NULL)
        delete _window;
    delete _ballSprite;
    delete _background2;
    delete _background3;
    delete _background4;
    delete _menu;
    delete _backgroundSprite;
    delete _rules;
}

//=======================================
// Accesseurs en Ã©criture
//=======================================

void View::synchronize()
{
    int x;
    int y;
    _model->getBallPosition(x, y);
    _ballSprite->setPosition(x, y);

    //Ajoute du score, +100 tous les 50milliseconds
    timeS = clock.getElapsedTime();
    if(timeS.asMilliseconds()%50 == 0 && timeS.asMilliseconds() > 3000 && play !
= 2)
    {
        _model->setScore(_model->getScore()+100);
    }

    time = clock.getElapsedTime();

    std::vector< MovableElement *> i = _model->getMovableElements();
    for (auto element: i)
    {
```



```
        //Suivant le type de l'obstacle, on ajoute la bonne texture avec les bon
nes coordonnees.
            if(element->getType() == 0){
                GraphicElement *obstacle = new GraphicElement(_obstacleTexture,
100, 100, 100, 50);
                _elementToGraphicElement[element] = obstacle;
            }
            else if (element->getType() == 1){
                GraphicElement *obstacle1 = new GraphicElement(_ennemies, 100, 4
00, 100, 150);
                _elementToGraphicElement[element] = obstacle1;
            }
            else if (element->getType() == 2){
                GraphicElement *obstacle2 = new GraphicElement(_ennemies2, 150,
250, 100, 150);
                _elementToGraphicElement[element] = obstacle2;
            }
            else if(element->getType() == 4){
                GraphicElement *piece = new GraphicElement(_pieceTexture, 50, 50
, 50, 50);
                _elementToGraphicElement[element] = piece;

            }
            else if(element->getType() == 5){
                GraphicElement *bonuslife = new GraphicElement(_bonuslifeTexture
, 50, 50, 50, 50);
                _elementToGraphicElement[element] = bonuslife;

            }
        }

    for (auto i : _elementToGraphicElement)
    {
        i.second->updatePosition(i.first->getPosition());
    }

    for (auto elem : _elementToGraphicElement) {
        std::vector<MovableElement *> elements = _model->getMovableElements();
        if (find(elements.begin(), elements.end(), elem.first) != elements.end()
) {
            x = elem.first->getX();
            y = elem.first->getY();
            _elementToGraphicElement[elem.first]->setPosition(x,y);
        }
        else {
            //On insert dans la poubelle les elements.
    Garbage.insert(elem);
        }
    }
    for(auto it: Garbage)
{
        //Pour tous les elements qui sont dans la poubelle, on les supprime.
        delete it.second;
        _elementToGraphicElement.erase(it.first);
    }
    Garbage.clear(); //On efface la map poubelle

    if (_model->getVie() <= 0) //Si la vie est inferieur ou egale a 0
        play = 2;                           //On passe en play = 2 (<=> drawEnd)
}

void View::setModel(Model * model){
    _model = model;
    play = 0;
}

// Gere l'animatedGraphicElement ( i est inferieur Ã  8 car le sprite de la ball
e contient 8 images)
void View::RectLecture(std::vector<sf::IntRect> &RectLecture)
```

```
{
    for(int i =0; i < 8; i++)
    {
        RectLecture.push_back(sf::IntRect(i*50,0,50,50));
    }
}

//======================================
// Traitement des evenements
//======================================
bool View::treatEvents(){
    bool result = false;
    sf::Vector2i mousepos = sf::Mouse::getPosition(*_window);
    if(_window->isOpen()){ //On regarde si la fenÃªtre est ouverte
        result = true;
        sf::Event event;
       while (_window->pollEvent(event)) {
            if ((event.type == sf::Event::Closed) ||
                    ((event.type == sf::Event::KeyPressed) && (event.key.code ==
 sf::Keyboard::Escape))) {
                _window->close();
                result = false;}
            else if (event.type == sf::Event::MouseButtonPressed) //Si on appuie
 sur un bouton de la souris
            {
                switch(event.key.code)
                {
                case sf::Mouse::Left: //Si on clique gauche sur la souris
                    if( mousepos.x >= 255 && mousepos.x <= 555 && mousepos.y >=
120 && mousepos.y <= 240)
                    { //Et que la position de la souris est au bon endroit
                        play = 1; //play = 1 <=> drawGame => dessine le jeu
                    }
                    else if(mousepos.x >= 255 && mousepos.x <= 555 && mousepos.y
 >= 275 && mousepos.y <= 395)
                    { //Et que la position de la souris est au bon endroit
                        _window->close(); //On ferme la fenÃªtre
                    }
                    else if(mousepos.x >= 230 && mousepos.x <= 570 && mousepos.y
 >= 425 && mousepos.y <= 545)
                    { //Et que la position de la souris est au bon endroit
                        play = 3; // play = 3 <=> drawRules => dessine les regle
s
                    }
                    else if(mousepos.x >= 680 && mousepos.x <= 800 && mousepos.y
 >= 530 && mousepos.y <= 600)
                    { //Et que la position de la souris est au bon endroit
                        play = 0; // play = 0 <=>drawMenu => dessine le menu
                    }
                    break;
                default:
                    break;
                }

            }
            else if (event.type == sf::Event::KeyPressed) //Si on appuie sur une
 touche
            {
                switch(event.key.code)
                {
                case sf::Keyboard::Left: //Si on appuie sur la fleche gauche
                    _model->moveBall(true); //la balle se depacera a gauche
                    break;
                case sf::Keyboard::Right: //Si on appuie sur la fleche droite
                    _model->moveBall(false); //La balle se deplacera a droite
                    break;
                case sf::Keyboard::Up: //Si on appuie sur la fleche du haut
                    _model->jumpBall(); //La balle va sauter
                    break;
```



```
                case sf::Keyboard::Add:// Si on appuie sur la touche +
                    _model->addElement("obstacle");// on ajoute un Ã©lÃ©ment de
type obstacle
                    break;
                default:
                    break;

                }
            }

        }
    }
    return result;
}
```