

Rapport PuruPuruDigger

M2103-AP2 Programmation orientée objet C++

CHARDAN Anaël, DAMEY Jérémy - 5 mai 2014



Sommaire

Rapport PuruPuruDigger	1
M2103-AP2 Programmation orientée objet C++	1
Sommaire	2
Introduction	3
Contexte et outils utilisés	3
Cahier des charges	4
Objectifs principaux	4
Objectifs supplémentaires	4
Objectifs bonus	4
Dossier de conception	5
Gestion de projet	7
Documentation :	9
Intégration de la documentation du manuel utilisateur	9
Manuel de l'interface en mode console :	9
Manuel de l'interface en mode SFML :	10
Conclusion	11
Annexes	12

Introduction

Le PuruPuruDigger est un jeu déjà existant sur la plateforme <http://www.jeux.fr/jeu/puru-puru-digger>. Il consiste en réalité à déplacer le personnage principal appelé « Digger » dans la grille. Il se déplace de manière horizontale, verticale ou diagonale d'un nombre de cases N défini par le chiffre inscrit dans sa case voisine, que l'utilisateur sélectionne par le biais d'un clic.



La grille contient 3 types de cases : Des cases numérotées de 1 à 6, des cases bombes, des cases trésors, une case contenant le mineur. A chaque fois que le mineur est déplacé dans une direction, la valeur inscrite dans la case est ajoutée au compteur (et un bonus s'il traverse une case bonus pendant son déplacement). Ce compteur est le score du joueur pour le niveau. Pour chaque niveau, lorsque le joueur atteint un score donné (l'objectif), le niveau est gagné et on passe au suivant.

Contexte et outils utilisés

L'objet de ce projet est de recoder ce jeu en binôme dans le cadre du cours M2103 Programmation Orienté Objet . Le but de ce projet est bien sûr de coder le jeu et cela en utilisant une architecture modèle vue. Certaines contraintes étaient donc à respecter :

Temps donné : 01/02/2014 au 23/05/2014

Langage : C++

Bibliothèque : SFML 1.6, STL

Patron de conception : Modèle / Vue

A cela s'ajoutent les choix que mon binôme et moi-même avons fait.

Gestionnaire de version : GIT via Github / Bitbucket

IDE : Xcode et CodeBlocks

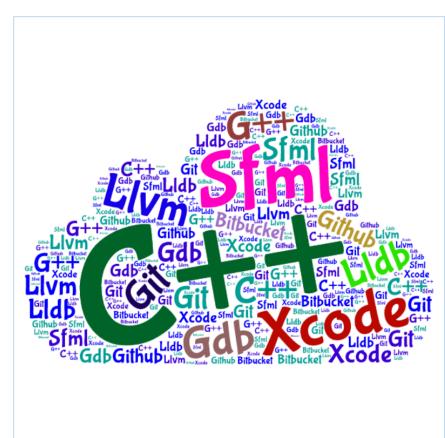
Tests Unitaires : Boost

Compilateur : LLVM et G++

Déboggeur : LLDB et GDB

Documentation : Doxygen

Patron de conception : Modèle / Vue , Observer Pattern et Pattern Singleton



Cahier des charges

Plusieurs étapes niveaux de fonctionnalités nous étaient proposés, nous devions les réaliser les uns après les autres pour les objectifs principaux et supplémentaires. Les fonctionnalités bonus étaient à faire en dernier lieu et étaient non-exhaustifs, nous pouvions rajouter autant de bonus que nous le voulions.

Celles que nous avons effectuées sont dotées d'un tag de couleur verte, les autres disposent d'un tag de couleur rouge.

Objectifs principaux

- Ecran d'introduction de jeu
- Écran de menu permettant via des boutons de lancer et de quitter le jeu
- Afficher un écran de jeu, un arrière plan, un mineur, des bombes, des cases numérotées, les scores
- Déplacement du mineur après un clic sur une case numérotée et valide [voisine]
- 3 niveaux jouables
- Des écrans de transition pour la fin de partie, et la fin de niveau

Objectifs supplémentaires

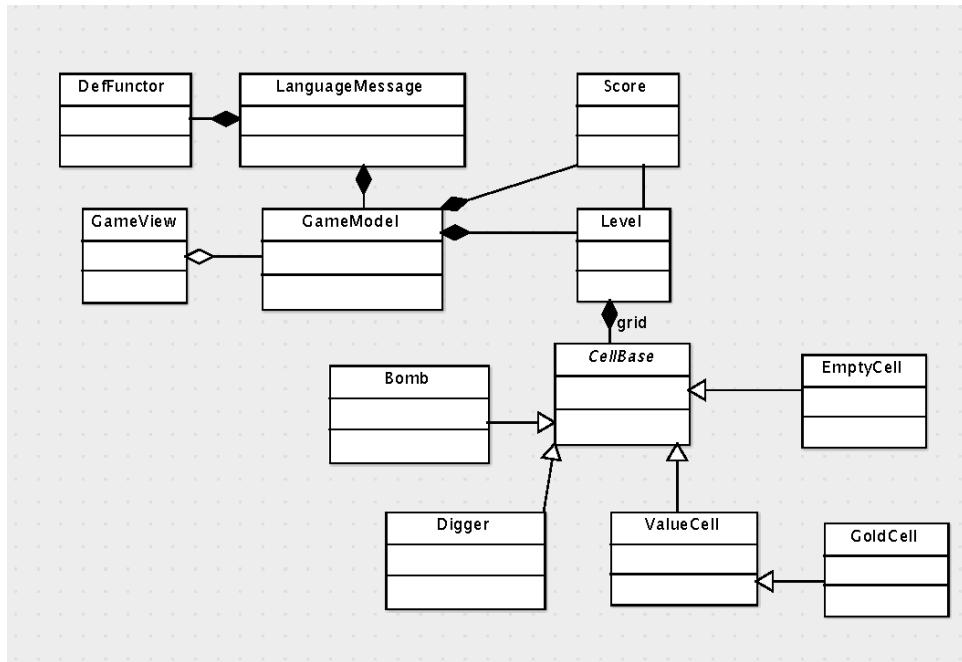
- Gestion du temps (l'objectif du niveau doit être réalisé en un temps donné)
- Animation des déplacement
- Ajout de bonus dans les cases numérotées
- Gestion des meilleurs scores avec un affichage d'un écran des meilleurs scores depuis le menu.
- Ajout de son

Objectifs bonus

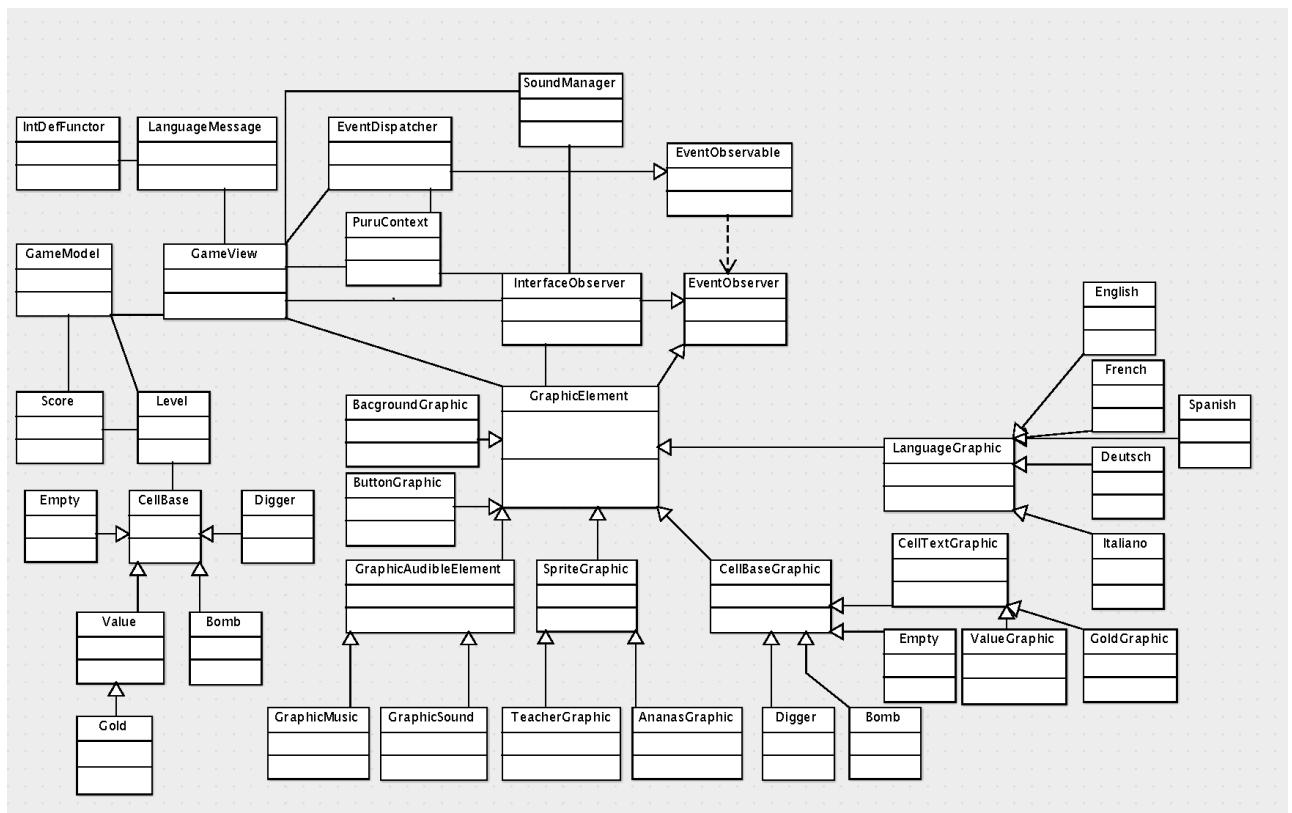
- Générateur aléatoire de niveau
- Nombre indéfini de niveau
- Changement de la charte graphique
- Support multi langues
- Éditeur de niveau
- Vérificateur de faisabilité de niveau

Dossier de conception

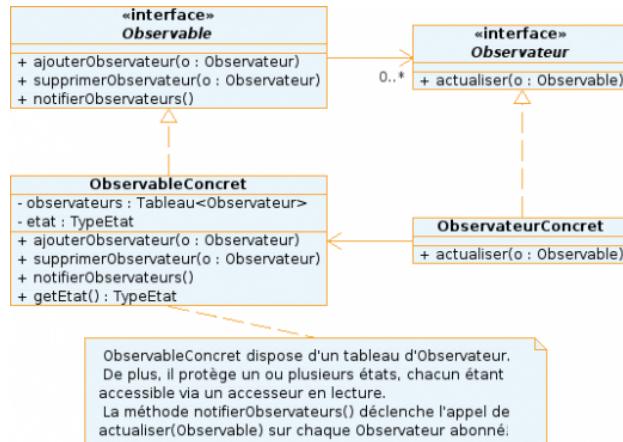
Tout d'abord, nous devions faire un rendu en mode Console, dont voici notre modèle de conception.



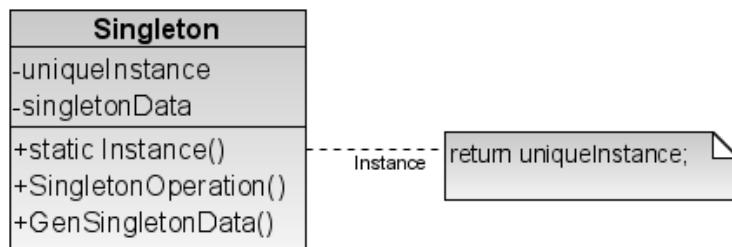
Mais après mûre réflexion, lorsque nous sommes en mode SFML nous avons ajouté des choses à notre conception que voici ci dessous.



Nous avons opté pour un Pattern Observer en plus du Modèle / Vue pour permettre de faire en sorte que les images réagissent elles-mêmes aux événements. De ce fait nous pouvons imaginer de gérer un événement par Observer. Et comme tout nos objets graphiques dérivent de la classe EventObserver, il nous est très facile de contrôler les réactions aux événements.

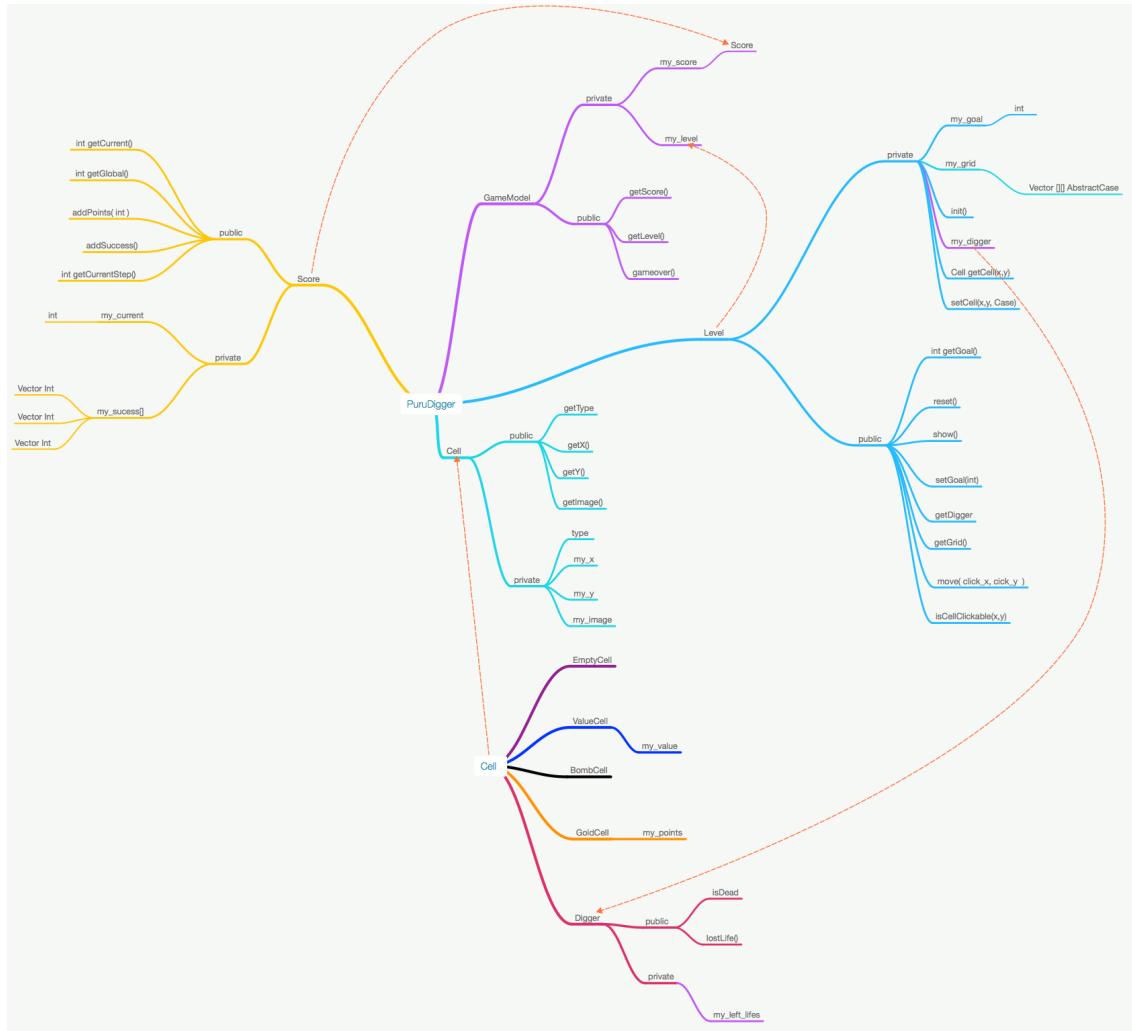


Ensuite nous avons décidé d'implémenter un Pattern Singleton afin de gerer une unique instance de la classe. Nous nous servons donc de ce modèle de conception pour faire un Manager de son afin de pouvoir jouer les sons beaucoup plus facilement et n'importe où dans le code. On sait alors où sont regroupes tous nos sons.



Gestion de projet

Au tout départ nous avons commencé par la conception sous la forme d'une bête à corne :



Ensuite Jérémy a retranscrit toutes nos informations dans Argo-UML afin de générer notre Diagramme de classe pour le rendu de conception comme vous avez pu le voir dans la partie dossier de conception.

Pour ce qui concerne le code, nous n'avions pas vraiment d'organisation, on a essayé d'avancer dans l'ordre et l'on se tenait au courant de ce que chacun devait faire.

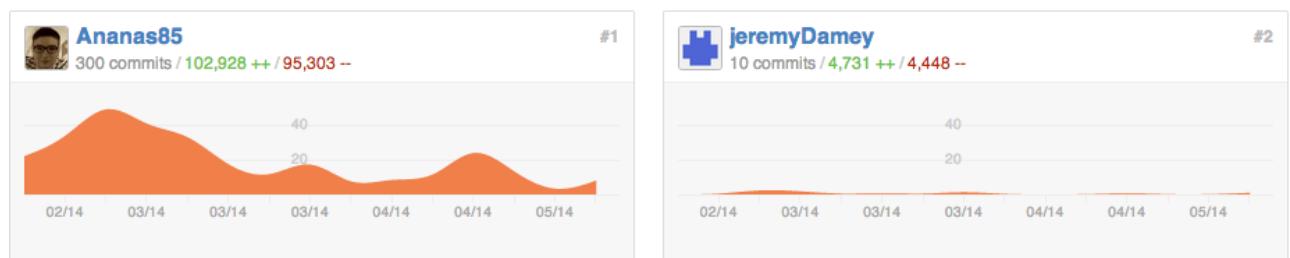
Ensuite la documentation fut fournie par Jérémy sous forme de commentaire Doxygen puis générée par Anaël et hébergée sur son site Web : <http://ananascorp.plopix.net/projet/untitled/html/index.html>.

Etant donné que nous nous sommes servis de GitHub comme service web d'hébergement et de gestion de développement de logiciel nous sommes en mesure de vous fournir des diagrammes fournis par GitHub.

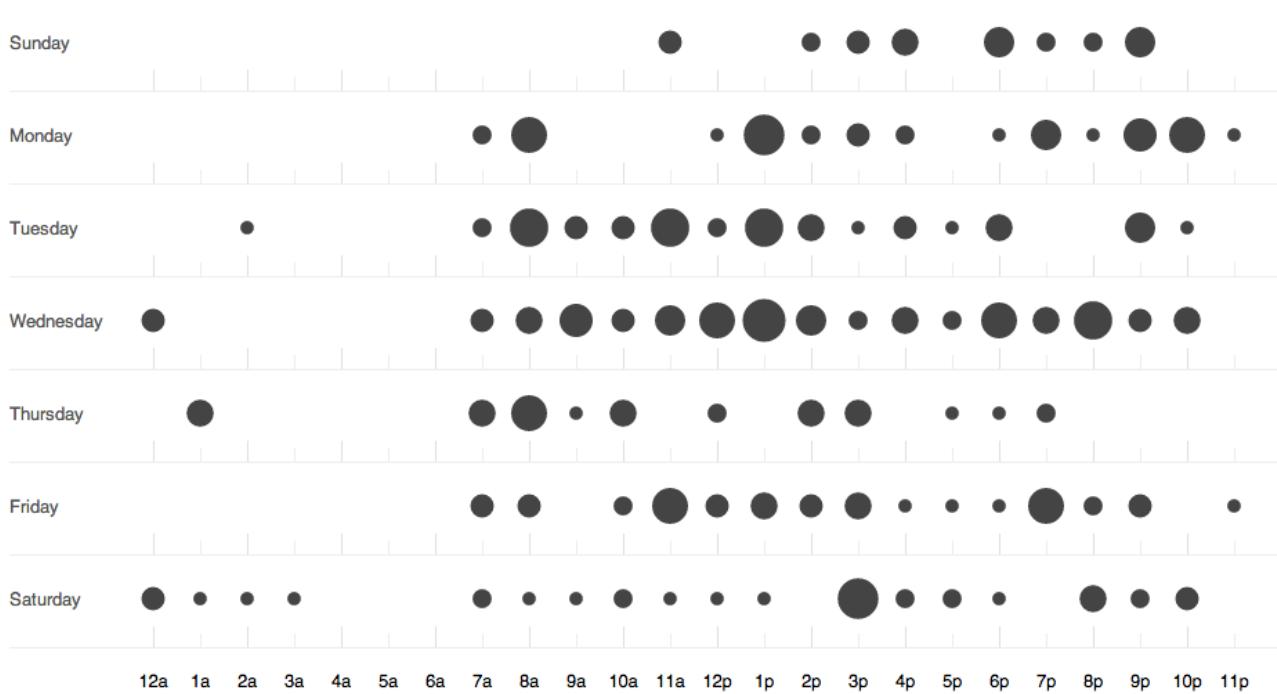
Vous pouvez voir ci dessous la courbe de travail, nous avons commencé très vite et très fort afin de ne pas être enseveli sous un charge de travail trop importante.



Contributeurs



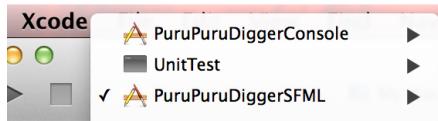
Punch Cards



Documentation :

Intégration de la documentation du manuel utilisateur

Notre jeu fut conçu avec deux vues différentes, une en mode console et une en mode SFML. Si nous vous donnons notre code à compiler les deux modes sont dans l'interface de votre IDE. Par exemple sous Xcode :



Manuel de l'interface en mode console :



Toutes les fonctionnalités avaient étées implémentées dans le mode console. Dès le démarrage on arrive sur le menu principal. Vous pouvez donc ici aller voir les meilleurs scores, quitter, ou encore jouer.

Si vous choisissez de jouer la langue vous est demandée, en effet le jeu est disponible en français, en anglais, en italien, en espagnol et en allemand.

1 : Français 2 : English 3 : Deutsch 4 : Espanol 5 : Italiano
CHOICE : █

Ensuite vous arrivez sur la fenêtre de jeu, où l'on retrouve une grille avec plusieurs cases. Notre Digger est représenté en blanc, les bombes en rouges, les cases vides avec rien à l'intérieur, les cases bonus en rose et les cases numérotées classiques en rouge.

En bas, vous retrouvez les informations sur le niveau courant, le score global, le score courant, l'objectif de déplacement, les déplacements en cours, la vie restante de notre Digger, la position du digger, le temps restants, et les choix de déplacements que vous avez sachant que :

Il est interdit d'aller sur une case vide ou une bombe à côté de vous. Le nombre de déplacements que vous effectuerez sera en fonction du numéro inscrit dans la case qui est dans la direction choisie.



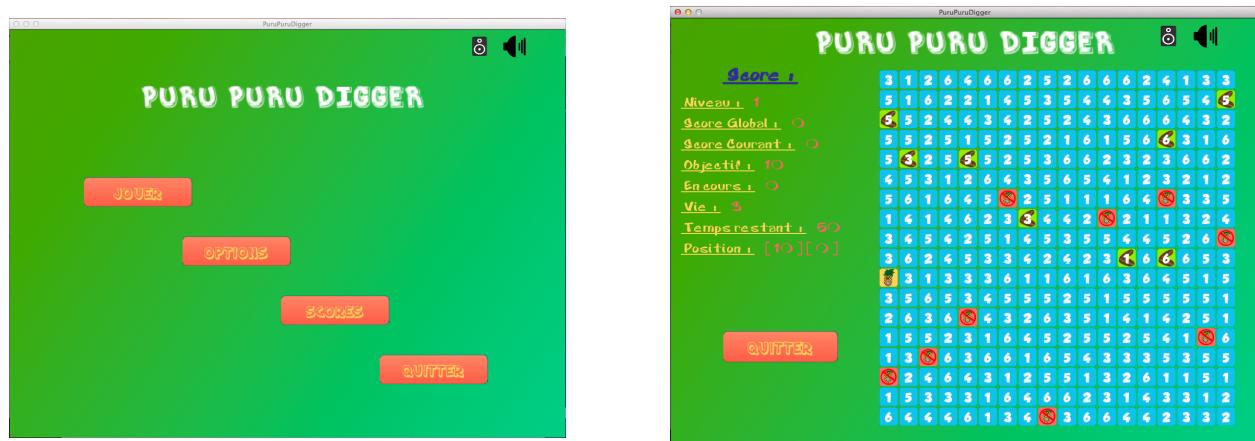
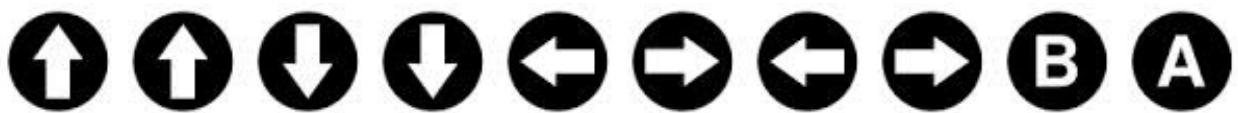
Si pendant le déplacement vous touchez une bombe, une case vide ou sortez du plateau vous perdez une vie et êtes téléporté dans un nouveau niveau de même objectif.

Si vous croisez une case bonus, vous pouvez gagner des points, retrouver la totalité de votre temps, ou gagner une vie.

A la fin de la partie il vous sera demandé de taper votre nom afin qu'il soit inscrit dans le top si vous avez fait dans les 5 meilleurs scores.

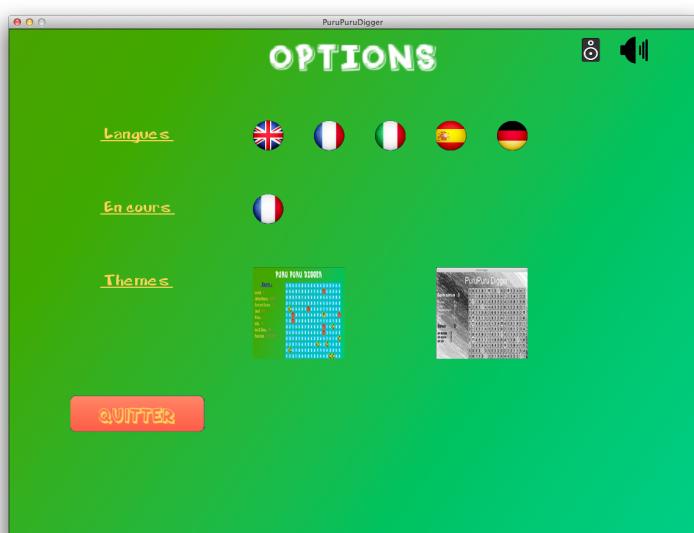
Manuel de l'interface en mode SFML :

Ici, nous allons juste vous montrer l'interface du jeu ainsi que le menu des options, puisque le principe du jeu reste le même mis à part une petite option. Vous pouvez tricher au jeu grâce au code Konami que vous pouvez taper en cours de jeu, il vous mènera directement au niveau suivant.



Voici le menu principal, vous pouvez voir ici que le menu est quasiment semblable mis à part le menu des options et les deux icônes en haut à droites qui vous permettent de jouer avec la musique et avec les sons.

Le jeu en lui-même est juste plus « friendly-user » et beaucoup plus aisément à prendre en main.



Pour ce qui concerne les options :

Vous pouvez voir ici que l'on peut donc toujours changer la langue par le biais des drapeaux de chacun des pays possibles.

Et vous remarquez également que le thème peut être changé.

Il y a deux modes : « l'ananas mode » celui que nous avons conçu et « le teacher mode » celui que notre professeur a conçu.

Conclusion

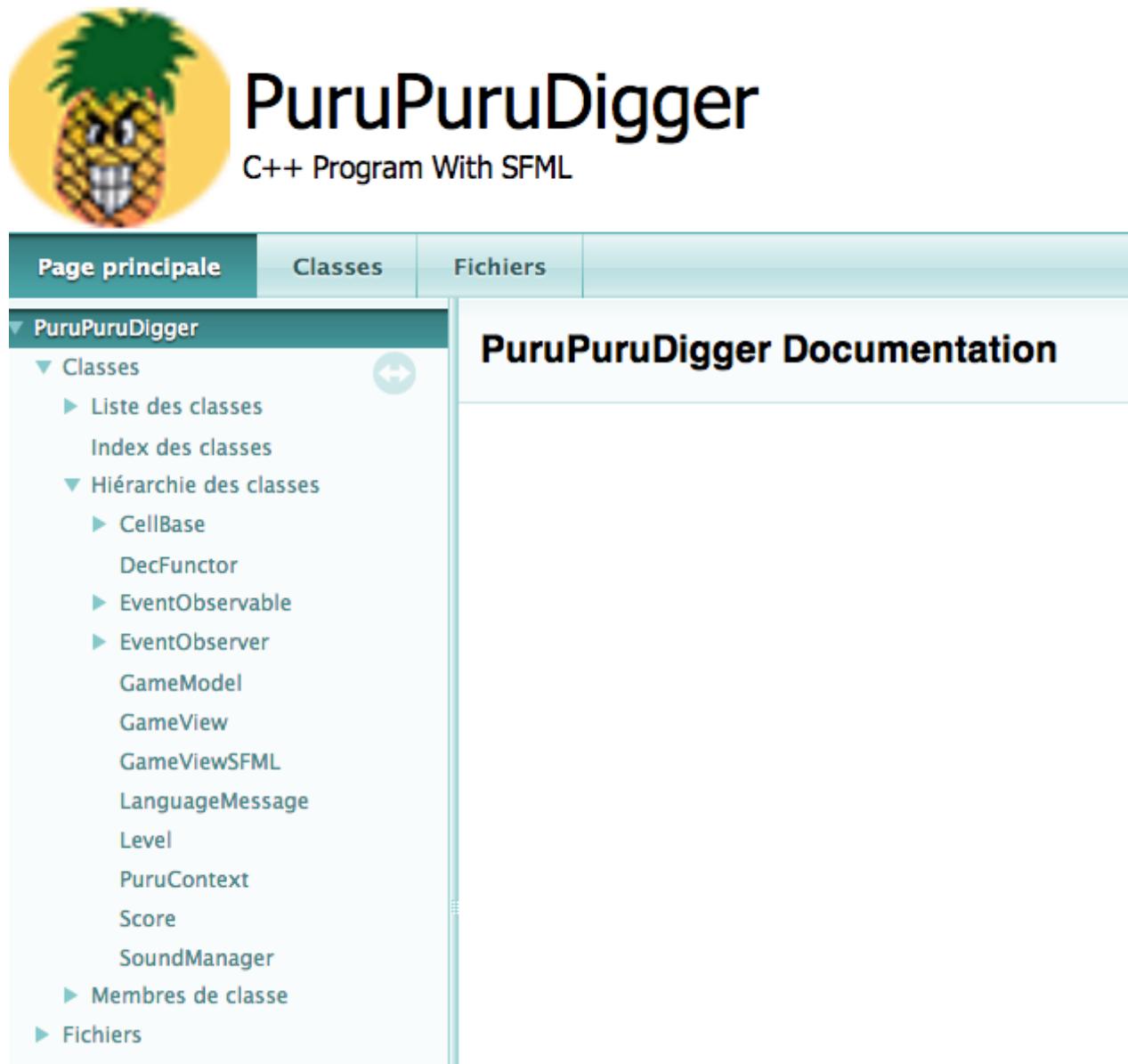
Du point de vue de notre cahier des charges, toutes les contraintes sont respectées. Tous les objectifs principaux ont été atteints comme nous pouvons le voir dans le cahier des charges. Nous disposons également de toutes les fonctionnalités supplémentaires ainsi que quelques fonctionnalités bonus.

Cependant si nous voulions pousser plus loin le concept quelques idées pourraient être implémentées comme le mode multi-joueur. Il pourrait même être joué en réseau puisque la SFML permet une communication par socket. Nous pourrions aussi faire l'éditeur de niveau même si dans un jeu comme celui-ci son intérêt est moindre. Nous pourrions aussi imaginer un jeu en 3 dimensions avec les fonctionnalités que l'OpenGL permet.

Annexes

Doxygen

Nous avons également fait une Doxygen qu'Anaël a hébergé sur son site à l'adresse suivante :
<http://ananascorp.plopix.net/projet/untitled/html/index.html>



The screenshot shows a Doxygen-generated documentation page for a C++ program named PuruPuruDigger, which uses SFML. The page features a header with a yellow circular logo containing a cartoon pineapple face, followed by the title "PuruPuruDigger" and the subtitle "C++ Program With SFML". Below the header is a navigation bar with tabs: "Page principale" (highlighted in green), "Classes", and "Fichiers". The main content area is titled "PuruPuruDigger Documentation". On the left, a sidebar contains a tree view of the project's structure:

- PuruPuruDigger
 - Classes
 - Liste des classes
 - Index des classes
 - Hiérarchie des classes
 - CellBase
 - DecFunctor
 - EventObservable
 - EventObserver
 - GameModel
 - GameView
 - GameViewSFML
 - LanguageMessage
 - Level
 - PuruContext
 - Score
 - SoundManager
 - Membres de classe
 - Fichiers

Le code source

```
#ifndef __PuruPuruDigger__SoundManager__
#define __PuruPuruDigger__SoundManager__

/***
 * \file SoundManager.h
 * \brief Notre classe SoundManager
 * \author CHARDAN Anaël
 * \author DAMEY Jérémy
 * \date 09/03/2014
 */

#include <iostream>
#include <SFML/Audio.hpp>
#include "../PuruContext.h"

/*! \class SoundManager
 * \brief Classe pour la gestion des sons
 */

class SoundManager {

public:
    /**
     * \brief Retourne une instance
     */
    static SoundManager* getInstance() {
        static SoundManager instance; // Instantiated when this function
                                      // is called
        return &instance;
    }

    /**
     * \brief Active ou non le son
     */
    void clickButton();

    /**
     * \brief Joue le son du clic
     */
    void clickCell();

    /**
     * \brief Active la musique
     */
    void playMusic();

    /**
     * \brief Met la musique en pause
     */
    void pauseMusic();

    /**
     * \brief Stop entièrement la musique
     */
}
```

```
void stopMusic();

/*
 * \brief Permet de savoir si l'on a perdu un niveau
 *
 */
void youLoose();

/*
 * \brief Permet de savoir si on a gagné un niveau
 *
 */
void youWin();

/*
 * \brief Permet de savoir si l'on a perdu la partie
 *
 */
void gameOver();

/*
 * \brief Permet de savoir si une touche est appuyé
 *
 */
void touchPress();

/*
 * \brief Modifie la valeur de context
 *
 */
void setContext( PuruContext *context );

private:
/*
 * \brief Constructeur par defaut
 *
 * Constructeur de la classe SoundManager
 *
 */
SoundManager(); // constructor is private

/*
 * \brief Constructeur par copie
 *
 * Constructeur de la classe SoundManager
 *
 */
SoundManager(SoundManager const&); // copy constructor
    is private

/*
 * \brief Operateur d'affectation
 *
 */
SoundManager& operator=(SoundManager const&); // assignment operator
    is private

/*
 * \brief Destructeur
 *
 */
```

```
* Destructeur de la classe SoundManager
*
*/
~SoundManager(); // destructor is
private

PuruContext *my_context; /*!< conetext */

sf::SoundBuffer *my_textBuffer; /*!< son quand on écrit du texte */
sf::Sound *my_textSound;

sf::SoundBuffer *my_gameOverBuffer; /*!< son quand on perd */
sf::Sound *my_gameOverSound;

sf::SoundBuffer *my_clickableBuffer; /*!< son quand on clique */
sf::Sound *my_clickableSoundCell;

sf::SoundBuffer *my_isNotClickableBuffer; /*!< son quand on déclique */
sf::Sound *my_isNotClickableSound;

sf::SoundBuffer *my_loseLevelBuffer; /*!< son quand perd un niveau */
sf::Sound *my_loseLevelSound;

sf::SoundBuffer *my_buttonBuffer; /*!< son quand on clique sur un
bouton */
sf::Sound *my_buttonSound;

sf::SoundBuffer* my_winBuffer;
sf::Sound* my_winSound;

sf::Music *my_musicLevel; /*!< music quand on joue un niveau */

};

#endif /* defined(__PuruPuruDigger__SoundManager__) */
```

```
/**\n * \file SoundManager.cpp\n * \brief Notre classe SoundManager\n * \author CHARDAN Anaël\n * \author DAMEY Jérémy\n * \date 09/03/2014\n */\n\n#include "SoundManager.h"\nusing namespace sf;\n\nSoundManager::~SoundManager() {\n\n    delete my_loseLevelBuffer;\n    delete my_buttonBuffer;\n    delete my_buttonSound;\n    delete my_musicLevel;\n\n    delete my_textBuffer;\n    delete my_textSound;\n    delete my_gameOverBuffer;\n    delete my_gameOverSound;\n    delete my_clickableBuffer;\n    delete my_clickableSoundCell;\n    delete my_isNotClickableBuffer;\n    delete my_isNotClickableSound;\n    delete my_loseLevelSound;\n    delete my_winBuffer;\n    delete my_winSound;\n\n}\n\nSoundManager::SoundManager() {\n\n    my_loseLevelBuffer = new SoundBuffer();\n    my_buttonBuffer= new SoundBuffer();\n    my_buttonSound = new Sound();\n    my_musicLevel = new Music();\n\n    my_textBuffer = new SoundBuffer();\n    my_textSound = new Sound();\n    my_gameOverBuffer = new SoundBuffer();\n    my_gameOverSound = new Sound();\n    my_clickableBuffer = new SoundBuffer();\n    my_clickableSoundCell = new Sound();\n    my_isNotClickableBuffer = new SoundBuffer();\n    my_isNotClickableSound = new Sound();\n    my_loseLevelSound = new Sound();\n    my_winBuffer = new SoundBuffer();\n    my_winSound= new Sound();\n\n#ifndef __linux__\n    if (!my_buttonBuffer->LoadFromFile("Ressources/Music/soundButton.wav") ||\n        !my_textBuffer->LoadFromFile("Ressources/Music/soundEnterText.wav") ||\n        !my_gameOverBuffer->LoadFromFile("Ressources/Music/soundGameOver.wav")\n        ||\n        !my_clickableBuffer->LoadFromFile("Ressources/Music/\n            soundIsClickable.wav") ||
```

```
!my_isNotClickableBuffer->LoadFromFile("Ressources/Music/
    soundIsNotClickable.wav") ||
!my_musicLevel->OpenFromFile( "Ressources/Music/gridMusic.wav" ) ||
!my_loseLevelBuffer->LoadFromFile("Ressources/Music/
    soundLoseLevel.wav") ||
!my_winBuffer->LoadFromFile("Ressources/Music/youWin.wav")
) {
    std::cout << "Error when loading font" << std::endl;
}
#else
if ( !my_buttonBuffer->LoadFromFile("soundButton.wav") ||
    !my_textBuffer->LoadFromFile("soundEnterText.wav") ||
    !my_gameOverBuffer->LoadFromFile("soundGameOver.wav") ||
    !my_clickableBuffer->LoadFromFile("soundIsClickable.wav") ||
    !my_isNotClickableBuffer->LoadFromFile("soundIsNotClickable.wav") ||
    !my_musicLevel->OpenFromFile( "gridMusic.wav" ) ||
    !my_loseLevelBuffer->LoadFromFile("soundLoseLevel.wav") ||
    !my_winBuffer->LoadFromFile("youWin.wav")
) {
    std::cout << "Error when loading font" << std::endl;
}
#endif
else {
    my_textSound->SetBuffer(*my_textBuffer);
    my_gameOverSound->SetBuffer(*my_gameOverBuffer);
    my_clickableSoundCell->SetBuffer(*my_clickableBuffer);
    my_isNotClickableSound->SetBuffer(*my_isNotClickableBuffer);
    my_loseLevelSound->SetBuffer(*my_loseLevelBuffer);
    my_buttonSound->SetBuffer(*my_buttonBuffer);
    my_winSound->SetBuffer(*my_winBuffer);
    my_musicLevel->SetLoop( true );
}
}

void SoundManager::setContext( PuruContext *context ) {
    my_context = context;
}

void SoundManager::clickButton() {
    if ( my_context->isEnableSound() )
        my_buttonSound->Play();
}

void SoundManager::clickCell() {
    if ( my_context->isEnableSound() )
        my_clickableSoundCell->Play();
}

void SoundManager::playMusic() {
    my_musicLevel->Play();
}

void SoundManager::pauseMusic() {
    my_musicLevel->Pause();
}

void SoundManager::stopMusic() {
    my_musicLevel->Stop();
}
```

```
void SoundManager::youLoose() {
    if ( my_context->isEnableSound() )
        my_loseLevelSound->Play();
}

void SoundManager::youWin() {
    if ( my_context->isEnableSound() )
        my_winSound->Play();
}

void SoundManager::gameOver() {
    if ( my_context->isEnableSound() )
        my_gameOverSound->Play();
}

void SoundManager::touchPress() {
    if ( my_context->isEnableSound() )
        my_textSound->Play();
}
```

```
#ifndef __PuruPuruDigger__InterfaceObserver__
#define __PuruPuruDigger__InterfaceObserver__

/***
 * \file InterfaceObserver.h
 * \brief Notre classe Interface observer
 * \author CHARDAN Anaël
 * \author DAMEY Jérémy
 * \date 09/03/2014
 */

#include <iostream>
#include "EventObserver.h"
#include "../Graphics/ButtonGraphic.h"
#include "../Constantes.h"
#include "../Graphics/LanguageGraphic.h"
#include "../LanguageMessage.h"
#include "../Graphics/AnanasSprite.h"
#include "../Graphics/ValueGraphic.h"
#include "../Graphics/GoldGraphic.h"
#include "../Graphics/TeacherSprite.h"
#include "../Graphics/BackgroundGraphic.h"
#include "../GameModel.h"
#include <string>
#include <SFML/System.hpp>
#include <SFML/Graphics.hpp>
#include <SFML/Window.hpp>
#include "../Graphics/CellBaseGraphic.h"
#include "../Graphics/GraphicMusic.h"
#include "../Graphics/GraphicSound.h"

typedef char KonamiCode[10]; /*!< Typedef pour faciliter l'écriture */

/*! \class InstanceObserver
 * \brief Classe pour la gestion de l'interface
 */
class InterfaceObserver : public EventObserver {

private:
    sf::RenderWindow* my_window; /*!< La fenêtre */
    GameModel* my_model; /*!< Le modèle */

    ButtonGraphic* my_playButton; /*!< Le bouton play */
    ButtonGraphic* my_settingButton; /*!< Le bouton setting */
    ButtonGraphic* my_bestButton; /*!< le bouton des meilleurs scores */
    ButtonGraphic* my_quitButton; /*!< Le bouton quitter */

    GraphicMusic *my_musicIcon; /*!< Le bouton de la musique */
    GraphicSound *my_soundIcon; /*!< Le bouton des sons */

    std::map< Language, LanguageGraphic* >* my_languageToSprite; /*!< La map
        des langues vers les sprites */

    AnanasSprite *my_ananasSprite; /*!< L'image de l'ananasMode */
    TeacherSprite *my_teacherSprite; /*!< L'image du teacher Mode */
    BackgroundGraphic *my_background; /*!< L'image de fond */

    sf::Font* my_fontScore; /*!< la font des score */
    sf::Font* my_fontTitle; /*!< La font du titre */
}
```

```
sf::Font* my_bestScoreFont; /*!< La font des meilleurs scores */

sf::String* my_titleScoreString; /*!< Titre des scores*/
sf::String* my_scoreString; /*!< Enoncé dans la grille*/
sf::String* my_scoreNum; /*!<Les scores en numéro*/
sf::String* my_titleString; /*!< Les titres de pages*/
sf::String* my_bestScoreString; /*!<Pour l'affichage des meilleurs
scores*/

LanguageMessage my_messages; /*!< La bibliothèque de message de notre
partie */

KonamiCode my_konamiCode; /*!< le code konami que l'on modifie */
KonamiCode my_trueKonamiCode; /*!< le bon code konami */
bool my_cheater; /*!< savoir si on on triché */

std::map< std::string, CellBaseGraphic* > my_stringToSprite;

GoldCell* ptr_goldCell; /*!< Le pointeur de gold celle */
ValueCell* ptr_valueCell; /*!< Le pointeur de valueCell */
GoldGraphic* ptr_goldGraphic; /*!< Le pointeur de gold graphic */
ValueGraphic* ptr_valueGraphic; /*!< Le pointeur de value graphic */

std::string player; /*!< Le nom de notre joueur */

sf::Clock pause; /*!<La clock pour la pause */

/*
 * \brief Affichage Menu principal
 */
void showPresentation();

/*
 * \brief Affichage Menu principal
 */
void resetLanguageNorm();

/*
 * \brief donne un text à un string le positionne et l'affiche
 * param[in] text le string
 * param[in] s le string qui va recevoir le texte
 * param[in] x la position en x
 * param[in] y la position en y
 * param[in] useSizeRectX pour savoir si on veut centrer selon x
 */
void setTextAndDraw( sf::String* s, std::string text, int x, int y, bool
useSizeRectX );

/*
 * \brief Affichage choix des langues
 */
void showLanguage();

/*
*/
```

```
* \brief Affichage choix des sprites
*/
void showSpriteChoice();

/*!
 * \brief Affichage des options
 */
void showOption();

/*!
 * \brief Affichage le nom de notre joueur
 * param[in] player le string de notre joueur
 */
void showIsEnteringABestScore( std::string player );

/*!
 * \brief Affichage des scores
 */
void showScore();

/*!
 * \brief Affichage de la grille
 */
void showGrid();

/*!
 * \brief Affichage du level
 */
void showLevel();

/*!
 * \brief Affichage du message de perte
 */
void showLoseLevel();

/*!
 * \brief Affichagedu message de gain
 */
void showWinLevel();

/*!
 * \brief Clear le screen
 */
void newScreen();

/*!
 * \brief Affichage des instructions de déplacement
 */
void showInstruction() ;

/*!
 * \brief Affichage des meilleurs scores
 */
void showBestScore();

/*!
 * \brief Réalisation de l'animation du digger
 */
```

```
void toAnimate();

<賓!
 * \brief On entre le score
 */
void enterScore() const;

<賓!
 * \brief On vérifie le code Konami
 */
bool konamiCode( sf::Event event );

<賓!
 * \brief Initialisation de notre code Konami
 */
void initKonamiCode();

public:

<賓!
 * \brief Constructeur
 */
InterfaceObserver(
    sf::RenderWindow* window,
    GameModel * model,
    ButtonGraphic *play,
    ButtonGraphic *setting,
    ButtonGraphic *best,
    ButtonGraphic *quit,
    GraphicMusic *music,
    GraphicSound *sound,
    std::map< Language, LanguageGraphic*> * languageToSprite
    ,
    AnanasSprite *ananas,
    TeacherSprite *teacher,
    BackgroundGraphic *background
);

<賓!
 * \brief Destructeur
 */
virtual ~InterfaceObserver();

<賓!
 * \brief Evenement mouse move
 * param[in] event un événement
 */
virtual void mouseMoved(sf::Event event);

<賓!
 * \brief Evenement keypressed
 * param[in] event un événement
 */
virtual void keyPressed(sf::Event event);

<賓!
```

```
* \brief Evenement textentered
* param[in] event un événement
*/
virtual void textEntered(sf::Event event);

/*!
* \brief Evenement button pressed
* param[in] event un événement
*/
virtual void mouseButtonPressed(sf::Event event);

/*!
* \brief Préparation de l'affichage
*/
virtual void preDisplay();

/*!
* \brief fin de l'affichage
*/
virtual void postDisplay();

/*!
* \brief changement de theme
*/
virtual void changeTheme( std::string theme );

};

#endif /* defined(__PuruPuruDigger__InterfaceObserver__) */
```

```
/**\n * \file InterfaceObserver.cpp\n * \author CHARDAN Anaël\n * \author DAMEY Jérémy\n * \date 09/03/2014\n */\n\n#include "InterfaceObserver.h"\n#include "../Graphics/EnglishGraphic.h"\n#include "../Graphics/FrenchGraphic.h"\n#include "../Graphics/SpanishGraphic.h"\n#include "../Graphics/DeutschGraphic.h"\n#include "../Graphics/ItalianoGraphic.h"\n#include "../Utils.h"\n#include <sstream>\n#include <fstream>\n#include "../Graphics/DiggerGraphic.h"\n#include "../Graphics/EmptyGraphic.h"\n#include "../Graphics/ValueGraphic.h"\n#include "../Graphics/BombGraphic.h"\n#include "../Graphics/GoldGraphic.h"\n#include "../IntDecFunctor.h"\n\nusing namespace std;\nusing namespace sf;\n\nInterfaceObserver::InterfaceObserver(\n    sf::RenderWindow* window,\n    GameModel* model,\n    ButtonGraphic *play,\n    ButtonGraphic *setting,\n    ButtonGraphic *best,\n    ButtonGraphic *quit,\n    GraphicMusic *music,\n    GraphicSound *sound,\n    std::map< Language, LanguageGraphic*>*\n        languageToSprite,\n    AnanasSprite *ananas,\n    TeacherSprite *teacher,\n    BackgroundGraphic *background\n) :\n    my_window( window ),\n    my_model( model ),\n    my_playButton( play ),\n    my_settingButton( setting ),\n    my_bestButton( best ),\n    my_quitButton( quit ),\n    my_musicIcon( music ),\n    my_soundIcon( sound ),\n    my_languageToSprite( languageToSprite ),\n    my_ananasSprite( ananas ),\n    my_teacherSprite( teacher ),\n    my_background( background )\n{\n    my_fontScore = new Font();\n    my_fontTitle = new Font();\n}
```

```
my_bestScoreFont = new Font();

my_scoreString = new String();
my_titleScoreString = new String();
my_scoreNum = new String();
my_titleString = new String();
my_bestScoreString = new String();

player = "";

my_stringToSprite["Digger"]    = new DiggerGraphic();
my_stringToSprite["EmptyCell"] = new EmptyGraphic();
my_stringToSprite["GoldCell"]  = new GoldGraphic();
my_stringToSprite["ValueCell"] = new ValueGraphic();
my_stringToSprite["Bomb"]      = new BombGraphic();

///On initialise le vrai konamiCode

my_trueKonamiCode[0] = my_trueKonamiCode[1] = 'U';
my_trueKonamiCode[2] = my_trueKonamiCode[3] ='D';
my_trueKonamiCode[4] = my_trueKonamiCode[6] = 'L';
my_trueKonamiCode[5] = my_trueKonamiCode[7] = 'R';
my_trueKonamiCode[8] = 'B';
my_trueKonamiCode[9] = 'A';

///On initialise le cheater
my_cheater = false;

initKonamiCode();

}

//Constructeur
InterfaceObserver::~InterfaceObserver() {
    delete my_fontScore;
    delete my_fontTitle;
    delete my_bestScoreFont;

    delete my_scoreString;
    delete my_scoreNum;
    delete my_titleScoreString;
    delete my_bestScoreString;
    delete my_titleString;

    for ( map<string, CellBaseGraphic*>::const_iterator it = my_stringToSprite
        .begin() ; it!=my_stringToSprite.end(); ++it) {
        delete my_stringToSprite[ it->first ];
    }
}

void InterfaceObserver::initKonamiCode() {
    //On initialise notre konamiCode
    for ( unsigned int i = 0; i < 10; ++i ) {
        my_konamiCode[i] = 'Z';
    }
}
```

```
void InterfaceObserver::resetLanguageNorm() {
    for ( std::map<Language, LanguageGraphic*>::const_iterator it =
        my_languageToSprite->begin() ; it!=my_languageToSprite->end(); ++it) {
        (*my_languageToSprite)[ it->first ]->reset();
    }
}

void InterfaceObserver::newScreen() {
    my_window->Clear();
    my_background->draw(my_window);
    my_musicIcon->draw(my_window);
    my_soundIcon->draw(my_window);
}

void InterfaceObserver::showPresentation() {
    //On affiche un écran "propre"
    newScreen();
    //On affiche tout sce dont on a besoin
    my_titleString->SetColor(Color(255,255,255));
    my_titleString->SetSize(60);
    setTextAndDraw( my_titleString, "PURU PURU DIGGER ", ( WINDOWWITDH / 2 ),
        100, true );
    my_playButton->setSpriteAndDraw(PLAYX, PLAYY, my_window, my_messages
        [my_context->getLanguage()][play]);
    my_settingButton->setSpriteAndDraw(OPTIONX, OPTIONY, my_window,
        my_messages[my_context->getLanguage()][setting]);
    my_bestButton->setSpriteAndDraw(BESTX, BESTY, my_window, my_messages
        [my_context->getLanguage()][best]);
    my_quitButton->setSpriteAndDraw(QUITX, QUITY, my_window, my_messages
        [my_context->getLanguage()][stop]);
}

void InterfaceObserver::showOption() {
    newScreen();
    //Le titre de la page
    setTextAndDraw( my_titleString, my_messages[my_context->getLanguage()]
        [setting], ( WINDOWWITDH / 2 ), 10, true );
    //L'énoncé langue
    setTextAndDraw( my_scoreString, my_messages[my_context->getLanguage()]
        [language], QUITONX + 50, CHOICELANGUEHIGH, false );
    setTextAndDraw( my_scoreString, my_messages[my_context->getLanguage()]
        [actual], QUITONX + 50, MYLANGUEY, false );
    setTextAndDraw( my_scoreString, my_messages[my_context->getLanguage()]
        [theme], QUITONX + 50, CHOICESPRITEY, false );
    showLanguage();

    //On place notre langue en cours
    (*my_languageToSprite)[my_context->getLanguage()]->setSpriteAndDraw
        (MYLANGUEX, MYLANGUEY, my_window);
    showSpriteChoice();
    my_quitButton->setSpriteAndDraw(QUITONX, QUITONY, my_window, my_messages
        [my_context->getLanguage()][stop]);
}

void InterfaceObserver::showLanguage() {
    (*my_languageToSprite)[english]->setSpriteAndDraw(ENGLISHX,
        CHOICELANGUEHIGH, my_window);
```

```
(*my_languageToSprite)[francais]->setSpriteAndDraw(FRENCHX,
    CHOICELANGUEHIGH, my_window);
(*my_languageToSprite)[espanol]->setSpriteAndDraw(SPANISHX,
    CHOICELANGUEHIGH, my_window);
(*my_languageToSprite)[deutsch]->setSpriteAndDraw(DEUTSCHX,
    CHOICELANGUEHIGH, my_window);
(*my_languageToSprite)[italiano]->setSpriteAndDraw(ITALIANOX,
    CHOICELANGUEHIGH, my_window);

}

void InterfaceObserver::showSpriteChoice() {
    my_ananasSprite->setSpriteAndDraw(CHOICEANANASX, CHOICESPRITEY, my_window)
    ;
    my_teacherSprite->setSpriteAndDraw(CHOICETEACHERX, CHOICESPRITEY,
        my_window);
}

void InterfaceObserver::showBestScore() {
    newScreen();

    ifstream scoreLect(FILEBESTSCORE.c_str(), ios::in );
    if ( scoreLect ) {
        string line;

        my_titleString->SetColor(Color(255,255,255));
        my_titleString->SetSize(60);

        //Le titre de la page
        setTextAndDraw( my_titleString, my_messages[my_context->getLanguage()]
            [score], ( WINDOWWITDH / 2 ), 10, true );
        int i = 200;
        //Le contenu de notre fichier
        while ( getline(scoreLect, line) ) {
            //Pour garantir la plus grande taille
            setTextAndDraw(my_bestScoreString, line, ( WINDOWWITDH / 2 ), i,
                true );
            i += 100;
        }

        //On affiche le bouton quitter avec son string
        my_quitButton->setSpriteAndDraw(QUITONX, QUITONY, my_window,
            my_messages[my_context->getLanguage()][stop]);
        scoreLect.close();
    } else {
        cerr << " Error when program is opening text file " << endl;
    }
}

//Cette méthode sert à mettre un texte à un string, le positionner, et le
//dessiner
void InterfaceObserver::setTextAndDraw( sf::String* s, string text, int x, int
y, bool useSizeRectX ) {

    s->SetText(text);

    ///Si le texte est plus grand que la fenêtre, on le réduit
    if ( s->GetRect().GetWidth() > WINDOWWITDH ) {
        while ( s->GetRect().GetWidth() > WINDOWWITDH )
            s->SetSize( s->GetSize() - 5 );
    }
}
```

```
}

///Si l'on veut centrer
if ( useSizeRectX )
    x -= ( ( s->GetRect().GetWidth() ) / 2 );

s->SetPosition(x, y);
my_window->Draw(*s);
}

void InterfaceObserver::showIsEnteringABestScore( string player ) {
    newScreen();
    setTextAndDraw( my_bestScoreString, my_messages[my_context->getLanguage()]
        [by], ( WINDOWWITDH / 2 ), 10, true );
    setTextAndDraw( my_bestScoreString, my_messages[my_context->getLanguage()]
        [name], ( WINDOWWITDH / 2 ), 100, true );
    setTextAndDraw( my_bestScoreString, player, ( WINDOWWITDH / 2 ) ,
        WINDOWHEIGHT / 2, true );
}

void InterfaceObserver::showGrid() {
    if ( my_context->isInAnimation() )
        toAnimate();
    else {
        for ( int i = 0; i < LIGNE ; i++ ) {
            for ( int j = 0; j < COLONNE; j++ ) {
                //On dessine le contenu de la case
                if ( my_model->getLevel()->getGrid()[i][j]->getType() ==
                    "GoldCell" ) {
                    ptr_goldCell = dynamic_cast<GoldCell*>(my_model->getLevel()
                        ()->getGrid()[i][j]);
                    ptr_goldGraphic = dynamic_cast<GoldGraphic*>
                        (my_stringToSprite["GoldCell"]);
                    ptr_goldGraphic->setSpriteAndDraw( convertIndiceXToPixel(
                        j ), convertIndiceYToPixel( i ), my_window, intToString
                        (ptr_goldCell->getValue()) );
                } else if ( my_model->getLevel()->getGrid()[i][j]->getType() ==
                    "ValueCell" ) {
                    ptr_valueCell = dynamic_cast<ValueCell*>(my_model->
                        getLevel()->getGrid()[i][j]);
                    ptr_valueGraphic = dynamic_cast<ValueGraphic*>
                        (my_stringToSprite["ValueCell"]);
                    ptr_valueGraphic->setSpriteAndDraw( convertIndiceXToPixel(
                        j ), convertIndiceYToPixel( i ), my_window, intToString
                        (ptr_valueCell->getValue()) );
                } else
                    my_stringToSprite[my_model->getLevel()->getGrid()[i][j]->
                        getType()]->setSpriteAndDraw(convertIndiceXToPixel( j )
                        , convertIndiceYToPixel( i ), my_window);
            }
        }
    }
}

void InterfaceObserver::showLoseLevel() {
    newScreen();
    my_titleString->SetSize(40);

    my_titleString->SetColor(Color(0,0,0));
```

```
if ( !my_context->isTimeOver() && !my_context->isOver() ) {
    setTextAndDraw( my_titleString, my_messages[my_context->getLanguage()]
                    [looselevel], ( WINDOWWITDH / 2 ), WINDOWHEIGHT / 2, true ) ;
} else if ( my_context->isTimeOver() ) {
    setTextAndDraw( my_titleString, my_messages[my_context->getLanguage()]
                    [timeup], ( WINDOWWITDH / 2 ), WINDOWHEIGHT / 2, true );
} else if ( my_context->isOver() ) {
    setTextAndDraw( my_titleString, my_messages[my_context->getLanguage()]
                    [loosegame], ( WINDOWWITDH / 2 ), WINDOWHEIGHT / 2, true );
} else if ( my_context->isTimeOver() && my_context->isOver() ) {
    setTextAndDraw( my_titleString, my_messages[my_context->getLanguage()]
                    [loosegame], ( WINDOWWITDH / 2 ), WINDOWHEIGHT / 2, true );
}
}

void InterfaceObserver::showWinLevel() {
    newScreen();

    my_titleString->SetSize(40);
    my_titleString->SetColor(Color(0,0,0));

    setTextAndDraw( my_titleString, my_messages[my_context->getLanguage()]
                    [winlevel], ( WINDOWWITDH / 2 ), WINDOWHEIGHT / 2, true ) ;

    if ( my_cheater ) {
        setTextAndDraw( my_titleString, my_messages[my_context->getLanguage()]
                        [cheater] , ( WINDOWWITDH / 2 ), 70 + WINDOWHEIGHT / 2, true ) ;
    }
}

void InterfaceObserver::showScore() {
    //Le titre
    setTextAndDraw( my_titleScoreString, my_messages[my_context->getLanguage()]
                    [score] + " : ", 100, 80, false);
    //Level et son num
    setTextAndDraw( my_scoreString, my_messages[my_context->getLanguage()]
                    [level] + " : ", 20, 140, false);
    setTextAndDraw( my_scoreNum, intToString(my_model->getScore()->
        getCurrentStep() ), my_scoreString->GetRect().GetWidth() + 40, 140,
        false );
    //Score Total
    setTextAndDraw( my_scoreString, my_messages[my_context->getLanguage()]
                    [global] + " : ", 20, 180, false);
    setTextAndDraw( my_scoreNum, intToString(my_model->getScore()->getGlobale
        () ), my_scoreString->GetRect().GetWidth() + 40, 180, false );
    //Score en cours
    setTextAndDraw( my_scoreString, my_messages[my_context->getLanguage()]
                    [current] + " : ", 20, 220, false);
    setTextAndDraw( my_scoreNum, intToString(my_model->getScore()->getCurrent
        () ), my_scoreString->GetRect().GetWidth() + 40, 220, false );
    //Objectif
    setTextAndDraw( my_scoreString, my_messages[my_context->getLanguage()]
                    [goal] + " : ", 20, 260, false);
    setTextAndDraw( my_scoreNum, intToString(my_model->getLevel()->getGoal() )
        , my_scoreString->GetRect().GetWidth() + 40, 260, false );
    //En cours
    setTextAndDraw( my_scoreString, my_messages[my_context->getLanguage()]
                    [step] + " : ", 20, 300, false);
    setTextAndDraw( my_scoreNum, intToString(my_model->getLevel()->
        getCurrentMove() ), my_scoreString->GetRect().GetWidth() + 40, 300 ,
```

```
        false);
//La vie
setTextAndDraw( my_scoreString, my_messages[my_context->getLanguage()]
    [life] + " : ", 20, 340, false);
setTextAndDraw( my_scoreNum, intToString(my_model->getLevel()->getDigger()
    ->getLife()), my_scoreString->GetRect().GetWidth() + 40, 340, false );

//Le temps
setTextAndDraw( my_scoreString, my_messages[my_context->getLanguage()]
    [ltime] + " : ", 20, 380, false);
setTextAndDraw( my_scoreNum, intToString( my_model->getLevel()->leftTime()
    ) , my_scoreString->GetRect().GetWidth() + 40, 380 , false);

//La position
setTextAndDraw( my_scoreString, my_messages[my_context->getLanguage()]
    [position] + " : ", 20, 420, false);
setTextAndDraw( my_scoreNum, "[" + intToString( my_model->getLevel()->
    getDigger()->getX() ) + " ] [ " + intToString( my_model->getLevel()->
    getDigger()->getY() ) + " ] " , my_scoreString->GetRect().GetWidth() +
    40, 420, false );
}

void InterfaceObserver::showLevel() {

    if ( !my_context->isInAnimation() ) {
        newScreen();
        my_titleString->SetColor(Color(255,255,255));
        my_titleString->SetSize(60);
        setTextAndDraw( my_titleString, " PURU PURU DIGGER " , ( WINDOWWITDH /
            2 ), 10, true );
        //On dessine le score
        showScore();
        my_quitButton->setSpriteAndDraw(QUITONX, QUITONY, my_window,
            my_messages[my_context->getLanguage()][stop]);
    }
    //On dessine la grille
    showGrid();
}

void InterfaceObserver::enterScore() const{
    ifstream scoreLect(FILEBESTSCORE.c_str(), ios::in );
    if ( scoreLect ) {
        string line;
        int scoreligne;
        string nomligne;
        map< int, string, DecFunctor> Scores;
        int scorePlayer = ( my_model->getScore() )->getGlobale() ;

        while ( !scoreLect.eof() ) {
            //On lit le score et on le stocke dans une map
            scoreLect >> scoreligne >> nomligne;
            Scores[scoreligne] = nomligne.c_str();
        }

        //On ajoute notre joueur à la map
        Scores[scorePlayer] = player;

        scoreLect.close();
    }
}
```

```
ofstream scoreEcr(FILEBESTSCORE.c_str(), ios::out | ios::trunc );

map< int, string>::iterator i;
if ( Scores.size() < 5 ) {
    i = Scores.end();
} else {
    i = Scores.begin();
    for ( int cpt = 0 ; cpt < 5; cpt ++ ) ++i;
}

for ( map< int, string >::const_iterator it = Scores.begin() ; it!=i ;
++it) {
    scoreEcr << it->first;
    scoreEcr << " ";
    scoreEcr << it->second;
    scoreEcr << endl;
}

scoreEcr.close();
} else {
    cerr << " Error when program is opening text file " << endl;
}
}

void InterfaceObserver::toAnimate() {

if ( ( convertIndiceXToPixel( my_model->getLevel()->getDigger()->getY() )
- my_stringToSprite["Digger"]->getXPos() ) == 0 &&
convertIndiceYToPixel( my_model->getLevel()->getDigger()->getX() )
- my_stringToSprite["Digger"]->getYPos() ) == 0 )
my_context->setAnimation( false );

else {

//Premier temps, on place la case vide

if ( my_model->getMovement() == South || my_model->getMovement() ==
East || my_model->getMovement() == SEast )
my_stringToSprite["EmptyCell"]->setSpriteAndDraw(
    convertIndiceXToPixel( convertXPixel( my_stringToSprite["Digger"]
"->getXPos() ) ), convertIndiceYToPixel( convertYPixel(
my_stringToSprite["Digger"]->getYPos() ) ), my_window );

else if ( my_model->getMovement() == North || my_model->getMovement() ==
NEast )
my_stringToSprite["EmptyCell"]->setSpriteAndDraw(
    convertIndiceXToPixel( convertXPixel( my_stringToSprite["Digger"]
"->getXPos() ) ), convertIndiceYToPixel( convertYPixel(
my_stringToSprite["Digger"]->getYPos() + CASEHEIGHT + 1 ) ),
my_window );

else if ( my_model->getMovement() == West || my_model->getMovement() ==
SWest )
my_stringToSprite["EmptyCell"]->setSpriteAndDraw(
    convertIndiceXToPixel( convertXPixel( my_stringToSprite["Digger"]
"->getXPos() + CASEWITDH + 1 ) ), convertIndiceYToPixel(
convertYPixel( my_stringToSprite["Digger"]->getYPos() ) ),
my_window );

else if ( my_model->getMovement() == Nwest )
```

```
my_stringToSprite["EmptyCell"]->setSpriteAndDraw(
    convertIndiceXToPixel( convertXPixel(my_stringToSprite["Digger"]
    ]->getXPos() + CASEWITDH ) ), convertIndiceYToPixel(
    convertYPixel( my_stringToSprite["Digger"]->getYPos() + 
CASEHEIGHT ) ), my_window );

//Second temps, on place le digger
switch ( my_model->getMovement() ) {
    case South :
        my_stringToSprite["Digger"]->setSpriteAndDraw(
            my_stringToSprite["Digger"]->getXPos(),
            my_stringToSprite["Digger"]->getYPos() + 2, my_window );
        break;

    case North :
        my_stringToSprite["Digger"]->setSpriteAndDraw(
            my_stringToSprite["Digger"]->getXPos(),
            my_stringToSprite["Digger"]->getYPos() - 2, my_window );
        break;

    case West :
        my_stringToSprite["Digger"]->setSpriteAndDraw(
            my_stringToSprite["Digger"]->getXPos() - 2,
            my_stringToSprite["Digger"]->getYPos(), my_window );
        break;

    case East :
        my_stringToSprite["Digger"]->setSpriteAndDraw(
            my_stringToSprite["Digger"]->getXPos() + 2,
            my_stringToSprite["Digger"]->getYPos(), my_window );
        break;

    case SEast :
        my_stringToSprite["Digger"]->setSpriteAndDraw(
            my_stringToSprite["Digger"]->getXPos() + 2,
            my_stringToSprite["Digger"]->getYPos() + 2, my_window );
        break;

    case SWest :
        my_stringToSprite["Digger"]->setSpriteAndDraw(
            my_stringToSprite["Digger"]->getXPos() - 2,
            my_stringToSprite["Digger"]->getYPos() + 2, my_window );
        break;

    case NEast :
        my_stringToSprite["Digger"]->setSpriteAndDraw(
            my_stringToSprite["Digger"]->getXPos() + 2,
            my_stringToSprite["Digger"]->getYPos() - 2, my_window );
        break;

    case NWest :
        my_stringToSprite["Digger"]->setSpriteAndDraw(
            my_stringToSprite["Digger"]->getXPos() - 2,
            my_stringToSprite["Digger"]->getYPos() - 2, my_window );
        break;

    default :
        break;
}
```

```
}

bool InterfaceObserver::konamiCode( Event event ) {

    bool check;
    int lastElement = 0;

    while ( my_konamiCode[lastElement] != 'Z' ) {
        lastElement++;
    }

    switch (event.Key.Code) {
        case Key::A :
            my_konamiCode[lastElement] = 'A';
            check = true;
            break;

        case Key::B :
            my_konamiCode[lastElement] = 'B';
            check = true;
            break;

        case Key::Up :
            my_konamiCode[lastElement] = 'U';
            check = true;
            break;

        case Key::Down :
            my_konamiCode[lastElement] = 'D';
            check = true;
            break;

        case Key::Left :
            my_konamiCode[lastElement] = 'L';
            check = true;
            break;

        case Key::Right :
            my_konamiCode[lastElement] = 'R';
            check = true;
            break;

        default:
            check = false;
            break;
    }

    if ( check ) {
        unsigned int z = 0;

        while ( check && z <= lastElement ) {
            if ( my_konamiCode [z] != my_trueKonamiCode[z] ) {
                check =false;
            } else
                z++;
        }
    }
}
```

```
}

if ( check == false || lastElement == 9 ) {
    for ( unsigned int i = 0; i < 10; ++i )
        my_konamiCode[i] = 'Z';
}

return ( check && lastElement == 9 );
}

/** Events Subscriber */

void InterfaceObserver::mouseMoved(sf::Event event) { }

void InterfaceObserver::keyPressed(sf::Event event) {

    ///Si l'on joue, on peu faire le code konami
    if ( my_context->isPlaying() ) {
        if ( konamiCode( event ) ) {
            my_model->getLevel()->winLevel();
            my_cheater = true;
        }
    }

    ///Si l'on est en train de taper notre nom
} else if ( my_context->isEnterABestScore() ) {
    switch (event.Key.Code) {
        ///Entrer pour valider
        case Key::Return :
            if ( player.length() > 0 ) {
                my_context->setEnterABestScore( false );
                my_context->setViewingBestScore( true );
                enterScore();
                player = "";
            }

            }
            break;

        ///BackSpace pour effacer des lettres
        case Key::Back :
            if ( player.length() > 0 )
                player.erase( player.length() - 1, 1 );
            break;

        default :
            break;
    }
}

void InterfaceObserver::textEntered(sf::Event event) {
    ///Si l'on est en train de rentrer notre nom
    if ( my_context->isEnterABestScore() ) {
        ///On bloque les caractères selon la table ascii
        if ( event.Text.Unicode >= 48 && event.Text.Unicode <127 && player.
            length() < 25 ) {
            ///On transtype et on ajoute au bout de notre nom de joueur
            player += static_cast<char>(event.Text.Unicode);
            SoundManager::getInstance()->touchPress();
        }
    }
}
```

```
    }

void InterfaceObserver::mouseButtonPressed(sf::Event event) {
    if ( my_quitButton->isInZone( event.MouseButton.X, event.MouseButton.Y ) )
    {
        initKonamiCode();
        SoundManager::getInstance()->stopMusic();
        my_context->setMusic( my_context->isEnableMusic() );
    }
}

void InterfaceObserver::preDisplay() {

    if ( my_context->isInPresentation() ) {
        showPresentation();

    } else if ( my_context->isChoosingOption() ) {
        showOption();

    } else if ( my_context->isViewingBestScore() ) {
        showBestScore();

    } else if ( my_context->isPlaying() ) {

        //On check le temps, et l'on peut perdre à cause de lui.
        if ( my_model->getLevel()->timeIsUp() ) {
            my_model->getLevel()->lostLevel();
            my_context->setTimeOver( true );
        }

        if ( my_model->gameOver() ) {
            if ( !my_context->isInBreak() ) {
                pause.Reset();
                SoundManager::getInstance()->gameOver();
            }
            my_context->setOver( true );
            my_context->setInBreak( true );
            my_context->setAnimation( false );

        }

        if ( my_model->getLevel()->lose() ) {
            if ( !my_context->isInBreak() ) {
                pause.Reset();
                SoundManager::getInstance()->youLoose();
            }
            showLoseLevel();
            my_context->setInBreak( true );
            my_context->setAnimation( false );

        } else if ( my_model->getLevel()->win() ) {
            if ( !my_context->isInBreak() ) {
                pause.Reset();
                SoundManager::getInstance()->youWin();
            }
            showWinLevel();
            my_context->setInBreak( true );
            my_context->setAnimation( false );
        } else {
    }
```

```
        showLevel();
    }
} else if ( my_context->isEnterABestScore() ) {
    showIsEnteringABestScore( player );
}

}

void InterfaceObserver::postDisplay() {
    if ( my_context->isInBreak() ) {

        if ( pause.GetElapsedTime() > 1.5 ) {

            my_model->getLevel()->resetLose();
            my_model->getLevel()->resetWin();
            my_cheater = false;
            my_model->getLevel()->resetTime();
            my_context->setInBreak( false );
            my_context->setTimeOver( false );
            pause.Reset();
            if ( my_model->gameOver() ) {
                initKonamiCode();
                my_context->setPlaying( false );
                my_context->setEnterABestScore( true );
                my_context->setOver( false );
                if ( my_context->isEnableMusic() ) {
                    SoundManager::getInstance()->pauseMusic();
                }
            }
        }
    }
}

void InterfaceObserver::changeTheme( std::string theme ) {
    std::string mypath;
#ifdef __linux__
    mypath = "Ressources/Font/" + theme;
#else
    mypath = theme;
#endif
    string fontScore = mypath + "_scoreFont.ttf";
    string fontTitle = mypath + "_titleFont.ttf";
    string bestScoreFont = mypath + "_BestFont.ttf";

    if ( theme == "teacher" ) {
        fontScore = mypath + "_arial.ttf";
        fontTitle = mypath + "_arial.ttf";
        bestScoreFont = mypath + "_arial.ttf";
    }

    if ( !my_fontScore->LoadFromFile( fontScore.c_str() ) || !my_fontTitle->
        LoadFromFile( fontTitle.c_str() ) || !my_bestScoreFont->LoadFromFile(
        bestScoreFont.c_str() ) ) {
        cout << "Error when loading fonts" << endl;
    } else {
        for ( map<string, CellBaseGraphic*>::const_iterator it =
            my_stringToSprite.begin(); it!=my_stringToSprite.end(); ++it) {
            my_stringToSprite[ it->first ]->changeTheme( theme );
        }
    }
}
```

```
my_titleScoreString->SetFont( *my_fontScore );
my_scoreString->SetFont( *my_fontScore );
my_scoreNum->SetFont( * my_fontScore );
my_titleString->SetFont( *my_fontTitle );
my_bestScoreString->SetFont( *my_bestScoreFont );

my_titleScoreString->SetStyle(String::Underlined | String::Bold |
    String::Italic );
my_scoreString->SetStyle(String::Underlined);

my_scoreString->SetSize(30);
my_titleScoreString->SetSize(40);
my_bestScoreString->SetSize(28);

if ( theme == "ananas" ) {

    my_titleScoreString->SetColor(Color(50,50,150));
    my_scoreString->SetColor(Color(251,210,98));
    my_scoreNum->SetColor(Color(255,100,100));
    my_bestScoreString->SetColor(Color(49,140,231));
} else {
    my_titleScoreString->SetColor(Color(255,255,255));
    my_scoreString->SetColor(Color(255, 255,255));
    my_scoreNum->SetColor(Color(255,255,255));
    my_bestScoreString->SetColor(Color(255,255,255));
}
}
```

```
#ifndef __PuruPuruDigger__EventObserver__
#define __PuruPuruDigger__EventObserver__

/***
 * \file EventObserver.h
 * \brief Notre classe abstraite EventObserver
 * \author CHARDAN Anaël
 * \author DAMEY Jérémy
 * \date 09/03/2014
 */

#include <string>
#include <set>
#include <iostream>
#include <SFML/Graphics.hpp>
#include "../PuruContext.h"
#include "../Manager/SoundManager.h"

/*! \class EventObserver
 * \brief Classe abstraite pour décrire un observer
 */
class EventObserver {
protected:
    PuruContext *my_context; /*!< Le context de notre jeu */

public:
    /*!
     * \brief Destructeur de la classe
     */
    virtual ~EventObserver();

    /*!
     * \brief Réaction de l'observer selon l'événement mouseMoved
     * param[in] un événement
     */
    virtual void mouseMoved(sf::Event event) = 0;

    /*!
     * \brief Réaction de l'observer selon l'événement keyPressed
     * param[in] un événement
     */
    virtual void keyPressed(sf::Event event) = 0;

    /*!
     * \brief Réaction de l'observer selon l'événement textEntered
     * param[in] un événement
     */
    virtual void textEntered(sf::Event event) = 0;

    /*!
     * \brief Réaction de l'observer selon l'événement boutonPressed
     * param[in] un événement
     */
    virtual void mouseButtonPressed(sf::Event event) = 0;

    /*!
     * \brief predDisplay
     */
    virtual void preDisplay() = 0;
```

```
/*!
 * \brief postDisplay
 */
virtual void postDisplay() = 0;

/*!
 * \brief changement de theme
 */
virtual void changeTheme( std::string theme ) = 0;

/*!
 * \brief injection de dépendance context
 */
void setContext( PuruContext *context );

};

#endif /* defined(__PuruPuruDigger__EventObserver__) */
```

```
/**\n * \file EventObserver.cpp\n * \brief Notre classe abstraite EventObserver\n * \author CHARDAN Anaël\n * \author DAMEY Jérémy\n * \date 09/03/2014\n */\n#include "EventObserver.h"\n\nEventObserver::~EventObserver() { }\n\nvoid EventObserver::setContext( PuruContext *context ) {\n    my_context = context;\n}
```

```
#ifndef __PuruPuruDigger_EventObservable__
#define __PuruPuruDigger_EventObservable__


/***
 * \file EventObservable.h
 * \brief Notre classe EventObservable
 * \author CHARDAN Anaël
 * \author DAMEY Jérémy
 * \date 09/03/2014
 */

#include <iostream>
#include <string>
#include <set>
#include "EventObserver.h"

/*! \class EventObservable
 */

class EventObservable {

private:
    std::set<EventObserver*> list_observers; /*!< Liste des observers
abonnées */

public:
    /**
     * \brief Destructeur de la classe
     */
    virtual ~EventObservable();

    /**
     * \brief notification d'un événements aux abonnés
     */
    void notify( sf::Event event ) const;

    /**
     * \brief preDisplay
     */
    void preDisplay() const;

    /**
     * \brief postDisplay
     */
    void postDisplay() const;

    /**
     * \brief Changement de theme
     */
    void changeTheme( std::string theme ) const;

    /**
     * \brief Ajouté un abonne
     */
    virtual void addObserver( EventObserver* observer );

    /**
     * \brief Retirer un abonne
     */
    virtual void removeObserver( EventObserver* observer );
};
```

};

#endif /* defined(__PuruPuruDigger__EventObservable__) */

```
/**\n * \file EventObservable.cpp\n * \author CHARDAN Anaël\n * \author DAMEY Jérémy\n * \date 09/03/2014\n */\n\n#include "EventObservable.h"\nusing namespace sf;\n\nEventObservable::~EventObservable() { }\n\nvoid EventObservable::notify( Event event ) const {\n    for (std::set<EventObserver*>::const_iterator it = list_observers.begin();\n         it != list_observers.end(); ++it) {\n\n        switch (event.Type) {\n\n            case Event::MouseMoved:\n                (*it)->mouseMoved(event);\n                break;\n\n            case Event::KeyPressed:\n                (*it)->keyPressed(event);\n                break;\n\n            case Event::TextEntered:\n                (*it)->textEntered(event);\n                break;\n\n            case Event::MouseButtonPressed:\n                (*it)->mouseButtonPressed(event);\n                break;\n\n            default:\n                break;\n        }\n    }\n}\n\nvoid EventObservable::preDisplay() const {\n    for (std::set<EventObserver*>::const_iterator it = list_observers.begin();\n         it != list_observers.end(); ++it) {\n        (*it)->preDisplay();\n    }\n}\n\nvoid EventObservable::postDisplay() const {\n    for (std::set<EventObserver*>::const_iterator it = list_observers.begin();\n         it != list_observers.end(); ++it) {\n        (*it)->postDisplay();\n    }\n}\n\nvoid EventObservable::changeTheme(std::string theme) const {\n    for (std::set<EventObserver*>::const_iterator it = list_observers.begin();\n         it != list_observers.end(); ++it) {\n        (*it)->changeTheme( theme );\n    }\n}
```

```
}

void EventObservable::addObserver(EventObserver* observer) {
    list_observers.insert(observer);
}

void EventObservable::removeObserver(EventObserver* observer) {
    list_observers.erase(observer);
}
```

```
#ifndef __PuruPuruDigger__EventDispatcher__
#define __PuruPuruDigger__EventDispatcher__

/***
 * \file EventDispatcher.h
 * \brief Notre classe EventDispatcher
 * \author CHARDAN Anaël
 * \author DAMEY Jérémy
 * \date 09/03/2014
 */

#include <iostream>
#include "EventObservable.h"
#include "../PuruContext.h"
#include "../Manager/SoundManager.h"

/*! \class EventDispatcher
 */
class EventDispatcher : public EventObservable {
private:
    PuruContext *my_context; /*!< Le context */
public:
    /*!
     * \brief Destructeur
     */
    virtual ~EventDispatcher();

    /*!
     * \brief Ajouté un abonne
     */
    void addObserver( EventObserver* observer );

    /*!
     * \brief Retirer un abonne
     */
    void removeObserver( EventObserver* observer );

    /*!
     * \brief Constructeur
     */
    EventDispatcher( PuruContext *context );
};

#endif /* defined(__PuruPuruDigger__EventDispatcher__) */
```

```
/**\n * \file EventDispatcher.cpp\n * \brief Notre classe EventDispatcher\n * \author CHARDAN Anaël\n * \author DAMEY Jérémy\n * \date 09/03/2014\n */\n\n#include "EventDispatcher.h"\n#include <SFML/Graphics.hpp>\n\nEventDispatcher::EventDispatcher( PuruContext *context ) : my_context( context ) { }\n\nEventDispatcher::~EventDispatcher() { }\n\nvoid EventDispatcher::addObserver( EventObserver* observer ) {\n    observer->setContext( my_context );\n    EventObservable::addObserver( observer );\n}\n\nvoid EventDispatcher::removeObserver( EventObserver* observer ) {\n    observer->setContext( NULL );\n    EventObservable::removeObserver( observer );\n}
```

```
#ifndef __PuruPuruDigger__GraphicElement__
#define __PuruPuruDigger__GraphicElement__

/***
 * \file GraphicElement.h
 * \brief Notre classe GraphicElement
 * \author CHARDAN Anaël
 * \author DAMEY Jérémy
 * \date 09/03/2014
 */

#include <iostream>

#include <SFML/System.hpp>
#include <SFML/Graphics.hpp>
#include <SFML/Window.hpp>
#include <SFML/Audio.hpp>
#include "../Observers/EventObserver.h"

/*! \class GraphicElement
 * \brief Classe pour l'affichage des éléments graphiques
 */

class GraphicElement : public EventObserver {
protected :
    sf::Sprite my_sprite; /*!< sprite choisi */

public :
    /*!
     * \brief Destructeur
     *
     * Destructeur de la classe GraphicElement
     */
    virtual ~GraphicElement();

    virtual void setImageToSprite() = 0; // On associe l'image à son
                                         // sprite

    /*!
     * \brief Configurer le sprite par rapport à l'image
     *
     * Affichage de l'image choisie aux coordonnées choisies
     *
     * \param[in] int x
     * \param[in] int y
     * \param[in] sf::RenderWindow * _window
     */
    virtual void setSpriteAndDraw( int x, int y, sf::RenderWindow *
                                  _window );

    /*!
     * \brief Configurer le sprite par rapport à l'image
     *
     * Affichage de l'image choisie
     *
     * \param[in] sf::RenderWindow* _window
     */
    virtual void draw( sf::RenderWindow* _window ) const;

}

```

```
* \brief Boolean qui permet de savoir si l'image est dans la zone
*
* Savoir si l'image se trouve dans la zone voulue
*
* \param[in] int x
* \param[in] int y
*/
bool isInZone( int x, int y ) const;

/*!
* \brief Affichage des coordonnees
*
* Affichage des coordonnées en x de l'image
*/
int getXPos() const;

/*!
* \brief Affichage des coordonnees
*
* Affichage des coordonnées en y de l'image
*/
int getYPos() const;

/*!
* \brief Exécuter le mouvement choisi à la souris
*
* \param[in] sf::Event event
*/
virtual void mouseMoved(sf::Event event);

/*!
* \brief Executer le mouvement choisi au clavier
*
* \param[in] sf::Event event
*/
virtual void keyPressed(sf::Event event);

/*!
* \brief Ecrire le texte
*
* \param[in] sf::Event event
*/
virtual void textEntered(sf::Event event);

/*!
* \brief Executer le mouvement choisi à la souris
*
* \param[in] sf::Event event
*/
virtual void mouseButtonPressed(sf::Event event);

/*!
* \brief Gestion de l'evenement avant le click
*/
virtual void preDisplay();

/*!
* \brief Gestion de l'evenement après le click
*/

```

```
 */
virtual void postDisplay();

<*/
 * \brief Changer le theme
 *
 * \param[in] std::string theme
 */
virtual void changeTheme( std::string theme );

};

#endif /* defined(__PuruPuruDigger__GraphicElement__) */
```

```
/**  
 * \file GraphicElement.cpp  
 * \brief Notre classe GraphicElement  
 * \author CHARDAN Anaël  
 * \author DAMEY Jérémy  
 * \date 09/03/2014  
 */  
  
#include "GraphicElement.h"  
  
void GraphicElement::setSpriteAndDraw(int x, int y, sf::RenderWindow *_window)  
{  
    my_sprite.SetPosition( x, y );  
    _window->Draw(my_sprite);  
}  
  
GraphicElement::~GraphicElement() {}  
  
bool GraphicElement::isInZone(int x, int y) const {  
    if ( x >= my_sprite.GetPosition().x && x <= ( my_sprite.GetPosition().x +  
        my_sprite.GetSize().x ) && y >= my_sprite.GetPosition().y && y <= (  
            my_sprite.GetPosition().y + my_sprite.GetSize().y ) )  
        return true;  
    else  
        return false;  
}  
  
void GraphicElement::draw(sf::RenderWindow *_window) const {  
    //On affiche à l'écran notre sprite  
    _window->Draw(my_sprite);  
}  
  
int GraphicElement::getXPos() const {  
    ///On retourne la position du sprite  
    return my_sprite.GetPosition().x;  
}  
  
int GraphicElement::getYPos() const {  
    return my_sprite.GetPosition().y;  
}  
  
/** Events Subscriber */  
  
void GraphicElement::mouseMoved( sf::Event event ) {}  
void GraphicElement::keyPressed( sf::Event event ) {}  
void GraphicElement::textEntered( sf::Event event ) {}  
void GraphicElement::mouseButtonPressed( sf::Event event ) {}  
void GraphicElement::postDisplay() {}  
void GraphicElement::preDisplay() {}  
void GraphicElement::changeTheme( std::string theme ) {}
```

```
#ifndef __PuruPuruDigger__CellBaseGraphic__
#define __PuruPuruDigger__CellBaseGraphic__

/***
 * \file CellBaseGraphic.h
 * \brief Notre classe CellBaseGraphic
 * \author CHARDAN Anaël
 * \author DAMEY Jérémy
 * \date 09/03/2014
 */

#include <iostream>
#include "GraphicElement.h"

/*! \class CellBaseGraphic
 * \brief Classe pour la gestion des cases numérotés
 */

class CellBaseGraphic : public GraphicElement {
protected :
    static sf::Image my_image; /*!< image de fond */

public :
    /*!
     * \brief Configurer le sprite par rapport à l'image
     *
     * Affichage de l'image
     */
    virtual void setImageToSprite();

    /*!
     * \brief Configurer le sprite par rapport à l'image
     *
     * Affichage du thème choisi, permet de changer de thème
     *
     * \param[in] std::string theme
     */
    virtual void changeTheme( std::string theme );
};

#endif /* defined(__PuruPuruDigger__CellBaseGraphic__) */
```

```
/**\n * \file CellBaseGraphic.cpp\n * \brief Notre classe CellBaseGraphic\n * \author CHARDAN Anaël\n * \author DAMEY Jérémy\n * \date 09/03/2014\n */\n\n#include "CellBaseGraphic.h"\n#include "../Constantes.h"\n\nsf::Image CellBaseGraphic::my_image;\n\nvoid CellBaseGraphic::changeTheme( std::string theme ) {\n    std::string myimage;\n#ifndef __linux__\n    myimage = "Ressources/Pictures/" + theme + "_case.png";\n#else\n    myimage = theme + "_case.png";\n#endif\n    if ( !my_image.LoadFromFile( myimage.c_str() ) ) {\n        std::cerr << " Error when loading case image " << std::endl;\n    } else {\n        setImageToSprite();\n    }\n}\n\nvoid CellBaseGraphic::setImageToSprite() {\n    my_image.CreateMaskFromColor(sf::Color(0, 55, 97));\n    my_sprite.setImage(my_image);\n    my_sprite.Resize( CASEWITDH, CASEHEIGHT );\n}
```

```
#ifndef __PuruPuruDigger__ValueGraphic__
#define __PuruPuruDigger__ValueGraphic__

/***
 * \file ValueGraphic.h
 * \brief Notre classe ValueGraphic
 * \author CHARDAN Anaël
 * \author DAMEY Jérémy
 * \date 09/03/2014
 */

#include <iostream>
#include "CellTextGraphic.h"

/*! \class ValueGraphic
 * \brief Classe pour l'affichage des chiffres dans les cases
 */

class ValueGraphic : public CellTextGraphic {
public :
    /*!
     * \brief Destructeur
     *
     * Destructeur de la classe ValueGraphic
     */
    virtual ~ValueGraphic();

    /*!
     * \brief Configurer le sprite par rapport à l'image
     *
     */
    virtual void setImageToSprite();
};

#endif /* defined(__PuruPuruDigger__ValueGraphic__) */
```

```
/**  
 * \file ValueGraphic.cpp  
 * \brief Notre classe ValueGraphic  
 * \author CHARDAN Anaël  
 * \author DAMEY Jérémy  
 * \date 09/03/2014  
 */  
  
#include "ValueGraphic.h"  
#include "../Constantes.h"  
  
ValueGraphic::~ValueGraphic() { }  
  
void  
ValueGraphic::setImageToSprite() {  
    CellTextGraphic::setImageToSprite();  
    my_sprite.SetSubRect( sf::IntRect ( VALUESX, SPRITECASEBEGIN, VALUEEX,  
        SPRITECASEHEIGHT ) );  
    my_sprite.Resize( CASEWITDH, CASEHEIGHT );  
}
```

```
#ifndef __PuruPuruDigger__DiggerGraphic__
#define __PuruPuruDigger__DiggerGraphic__

/***
 * \file DiggerGraphic.h
 * \brief Notre classe DiggerGraphic
 * \author CHARDAN Anaël
 * \author DAMEY Jérémy
 * \date 09/03/2014
 */

#include <iostream>
#include "CellBaseGraphic.h"

/*! \class DiggerGraphic
 * \brief Classe pour la gestion du digger
 */

class DiggerGraphic : public CellBaseGraphic {
public :
    /*!
     * \brief Configurer le sprite par rapport à l'image
     *
     * Affichage du thème choisi
     */
    virtual void setImageToSprite();

    /*!
     * \brief Configurer le sprite par rapport à l'image
     *
     * Affichage de l'image du digger
     */
    virtual ~DiggerGraphic();
};

#endif /* defined(__PuruPuruDigger__DiggerGraphic__) */
```

```
/**\n * \file DiggerGraphic.cpp\n * \brief Notre classe DiggerGraphic\n * \author CHARDAN Anaël\n * \author DAMEY Jérémy\n * \date 09/03/2014\n */\n\n#include "DiggerGraphic.h"\n#include "../Constantes.h"\n\nDiggerGraphic::~DiggerGraphic() { }\n\nvoid\nDiggerGraphic::setImageToSprite() {\n    CellBaseGraphic::setImageToSprite();\n    my_sprite.SetSubRect( sf::IntRect ( DIGGERSX, SPRITECASEBEGIN, DIGGEREX,\n        SPRITECASEHEIGHT ) );\n    my_sprite.Resize( CASEWITDH, CASEHEIGHT);\n}
```

```
#ifndef __PuruPuruDigger__SpriteGraphic__
#define __PuruPuruDigger__SpriteGraphic__

/***
 * \file SpriteGraphic.h
 * \brief Notre classe SpriteGraphic
 * \author CHARDAN Anaël
 * \author DAMEY Jérémy
 * \date 09/03/2014
 */

#include <iostream>
#include "GraphicElement.h"

/*! \class SpriteGraphic
 * \brief Classe pour l'affichage des images
 */

class SpriteGraphic : public GraphicElement {
protected :
    static sf::Image my_image ; /*!< image de fond */

public :
    /*!
     * \brief Configurer le sprite par rapport à l'image
     *
     */
    virtual void setImageToSprite();

    /*!
     * \brief Changer le theme
     *
     * \param[in] std::string theme
     */
    virtual void changeTheme( std::string theme );
};

#endif /* defined(__PuruPuruDigger__SpriteGraphic__) */
```

```
/**\n * \file SpriteGraphic.cpp\n * \brief Notre classe SpriteGraphic\n * \author CHARDAN Anaël\n * \author DAMEY Jérémy\n * \date 09/03/2014\n */\n\n#include "SpriteGraphic.h"\n#include "../Constantes.h"\n\nsf::Image SpriteGraphic::my_image;\n\nvoid SpriteGraphic::changeTheme( std::string theme ) {\n    std::string myimage;\n#ifndef __linux__\n    myimage = "Ressources/Pictures/choiceSprite.png";\n#else\n    myimage = "choiceSprite.png";\n#endif\n\n    if ( !my_image.LoadFromFile( myimage.c_str() ) ) {\n        std::cerr << " Error when loading choiceSprite image " << std::endl;\n    } else {\n        setImageToSprite();\n    }\n}\n\nvoid SpriteGraphic::setImageToSprite() {\n    my_image.CreateMaskFromColor(sf::Color(0, 55, 97));\n    my_sprite.SetImage(my_image);\n}
```

```
#ifndef __PuruPuruDigger__TeacherSprite__
#define __PuruPuruDigger__TeacherSprite__

/***
 * \file TeacherSprite.h
 * \brief Notre classe TeacherSprite
 * \author CHARDAN Anaël
 * \author DAMEY Jérémy
 * \date 09/03/2014
 */

#include <iostream>
#include "SpriteGraphic.h"

/*! \class TeacherSprite
 * \brief Classe pour l'affichage du jeu avec les sprites du professeur
 */

class TeacherSprite : public SpriteGraphic {
public :
    /*!
     * \brief Configurer le sprite par rapport à l'image
     */
    virtual void setImageToSprite();

    /*!
     * \brief Destructeur
     *
     * Destructeur de la classe TeacherSprite
     */
    virtual ~TeacherSprite();
};

#endif /* defined(__PuruPuruDigger__TeacherSprite__) */
```

```
/**\n * \file TeacherGraphic.cpp\n * \brief Notre classe TeacherGraphic\n * \author CHARDAN Anaël\n * \author DAMEY Jérémy\n * \date 09/03/2014\n */\n\n#include "TeacherSprite.h"\n#include "../Constantes.h"\n\nTeacherSprite::~TeacherSprite() { }\n\nvoid\nTeacherSprite::setImageToSprite() {\n    SpriteGraphic::setImageToSprite();\n    my_sprite.SetSubRect( sf::IntRect ( SPRITETEACHERSX, SPRITECHOICEBEGIN,\n        SPRITETEACHEREX, SPRITECHOICEHEIGHT ) );\n    my_sprite.Resize( SPRITECHOICEWIDTH, SPRITECHOICEHEIGHT );\n}\n
```

```
#ifndef __PuruPuruDigger__AnanasSprite__
#define __PuruPuruDigger__AnanasSprite__

/***
 * \file AnanasSprite.h
 * \brief Notre classe AnanasSprite
 * \author CHARDAN Anaël
 * \author DAMEY Jérémy
 * \date 09/03/2014
 */

#include <iostream>
#include "SpriteGraphic.h"

/*! \class AnanasSprite
 * \brief Classe pour l'Ananas mode
 */

class AnanasSprite : public SpriteGraphic {
public :
    /*!
     * \brief Configurer le sprite par rapport à l'image
     *
     * Affichage de l'Ananas mode
     */
    virtual void setImageToSprite();

    /*!
     * \brief Destructeur
     *
     * Destructeur de la classe AnanasSprite
     */
    virtual ~AnanasSprite();
};

#endif /* defined(__PuruPuruDigger__AnanasSprite__) */
```

```
/**\n * \file AnanasSprite.cpp\n * \brief Notre classe AnanasSprite\n * \author CHARDAN Anaël\n * \author DAMEY Jérémy\n * \date 09/03/2014\n */\n\n#include "AnanasSprite.h"\n#include "../Constantes.h"\n\nAnanasSprite::~AnanasSprite() { }\n\nvoid\nAnanasSprite::setImageToSprite() {\n    SpriteGraphic::setImageToSprite();\n    my_sprite.SetSubRect( sf::IntRect( SPRITEANANASSX, SPRITECHOICEBEGIN,\n        SPRITEANANASEX, SPRITECHOICEHEIGHT ) );\n    my_sprite.Resize( SPRITECHOICEWIDTH, SPRITECHOICEHEIGHT );\n}\n
```

```
#ifndef __PuruPuruDigger__LanguageGraphic__
#define __PuruPuruDigger__LanguageGraphic__

/***
 * \file LanguageGraphic.h
 * \brief Notre classe LanguageGraphic
 * \author CHARDAN Anaël
 * \author DAMEY Jérémy
 * \date 09/03/2014
 */

#include <iostream>
#include "GraphicElement.h"

/*! \class LanguagaGraphic
 * \brief Classe pour l'affichage de la charte graphique des langages
 */

class LanguageGraphic : public GraphicElement {
protected :
    static sf::Image my_image ; /*!< image de fond */

public :
    /*!
     * \brief Configurer le sprite par rapport à l'image
     *
     * Destructeur de la classe GraphicElement
     */
    virtual void setImageToSprite();

    /*!
     * \brief Change la couleur du sprite
     */
    void setHover();

    /*!
     * \brief Reinitialise la couleur de l'image
     */
    void reset();

    /*!
     * \brief Change la couleur du sprite en fonction de la position de la
     * souris
     */
    virtual void mouseMoved(sf::Event event);

    /*!
     * \brief Changer de theme
     *
     * \param[in] std::string theme
     */
    virtual void changeTheme( std::string theme );

};

#endif /* defined(__PuruPuruDigger__LanguageGraphic__) */
```

```
/**\n * \file LanguageGraphic.cpp\n * \brief Notre classe LanguageGraphic\n * \author CHARDAN Anaël\n * \author DAMEY Jérémy\n * \date 09/03/2014\n */\n\n#include "LanguageGraphic.h"\n#include "../Constantes.h"\n\nsf::Image LanguageGraphic::my_image;\n\nvoid LanguageGraphic::changeTheme( std::string theme ) {\n    std::string myimage;\n#ifndef __linux__\n    myimage = "Ressources/Pictures/";\n#endif\n    myimage += "languages.png";\n    if ( !my_image.LoadFromFile( myimage.c_str() ) ) {\n        std::cerr << " Error when loading languages image " << std::endl;\n    } else {\n        setImageToSprite();\n    }\n}\n\nvoid LanguageGraphic::setImageToSprite() {\n    my_image.CreateMaskFromColor(sf::Color(0, 55, 97));\n    my_sprite.SetImage(my_image);\n}\n\nvoid LanguageGraphic::setHover() {\n    my_sprite.SetColor(sf::Color(255,255,255,128));\n}\n\nvoid LanguageGraphic::reset() {\n    my_sprite.SetColor(sf::Color(255,255,255,255));\n}\n\n/** Events Subscriber */\n\nvoid LanguageGraphic::mouseMoved( sf::Event event ) {\n    if ( isInZone( event.MouseMove.X, event.MouseMove.Y ) ) {\n        setHover();\n    } else {\n        reset();\n    }\n}
```

```
#ifndef __PuruPuruDigger__SpanishGraphic__
#define __PuruPuruDigger__SpanishGraphic__

/***
 * \file SpanishGraphic.h
 * \brief Notre classe SpanishGraphic
 * \author CHARDAN Anaël
 * \author DAMEY Jérémy
 * \date 09/03/2014
 */

#include <iostream>
#include "LanguageGraphic.h"

/*! \class SpanishGraphic
 * \brief Classe pour l'affichage du jeu en espagnol
 */

class SpanishGraphic : public LanguageGraphic {
public :
    /*!
     * \brief Configurer le sprite par rapport à l'image
     */
    virtual void setImageToSprite();

    /*!
     * \brief Destructeur
     *
     * Destructeur de la classe SpanishGraphic
     */
    virtual ~SpanishGraphic();
};

#endif /* defined(__PuruPuruDigger__SpanishGraphic__) */
```

```
/**\n * \file SpanishGraphic.cpp\n * \brief Notre classe SpanishGraphic\n * \author CHARDAN Anaël\n * \author DAMEY Jérémy\n * \date 09/03/2014\n */\n\n#include "SpanishGraphic.h"\n#include "../Constantes.h"\n\nSpanishGraphic::~SpanishGraphic() { }\n\nvoid\nSpanishGraphic::setImageToSprite() {\n    LanguageGraphic::setImageToSprite();\n    my_sprite.SetSubRect( sf::IntRect ( SPANISHSX, SPRITELANGUEBEGIN,\n        SPANISHEX, SPRITELANGUEHEIGHT ) );\n    my_sprite.SetPosition(SPANISHX, CHOICELANGUEHIGH);\n    my_sprite.Resize( LANGUEWIDTH, LANGUEHEIGHT);\n}
```

```
#ifndef __PuruPuruDigger__ItalianoGraphic__
#define __PuruPuruDigger__ItalianoGraphic__

/***
 * \file ItalianoGraphic.h
 * \brief Notre classe Italiano
 * \author CHARDAN Anaël
 * \author DAMEY Jérémy
 * \date 09/03/2014
 */

#include <iostream>
#include "LanguageGraphic.h"

/*! \class ItalianoGraphic
 * \brief Classe pour l'affichage du jeu en Italien
 */

class ItalianoGraphic : public LanguageGraphic {
public :
    /*!
     * \brief Configurer le sprite par rapport à l'image
     */
    virtual void setImageToSprite();

    /*!
     * \brief Destructeur
     *
     * Destructeur de la classe ItalianoGraphic
     */
    virtual ~ItalianoGraphic();
};

#endif /* defined(__PuruPuruDigger__ItalianoGraphic__) */
```

```
/**\n * \file ItalianoGraphic.cpp\n * \brief Notre classe ItalianoGraphic\n * \author CHARDAN Anaël\n * \author DAMEY Jérémy\n * \date 09/03/2014\n */\n\n#include "ItalianoGraphic.h"\n#include "../Constantes.h"\n\nItalianoGraphic::~ItalianoGraphic() { }\n\nvoid\nItalianoGraphic::setImageToSprite() {\n    LanguageGraphic::setImageToSprite();\n    my_sprite.SetSubRect( sf::IntRect ( ITALIANOSX, SPRITELANGUEBEGIN,\n        ITALIANOEX, SPRITELANGUEHEIGHT ) );\n    my_sprite.SetPosition(ITALIANOX, CHOICELANGUEHIGH);\n    my_sprite.Resize( LANGUEWIDTH, LANGUEHEIGHT);\n}
```

```
#ifndef __PuruPuruDigger__EnglishGraphic__
#define __PuruPuruDigger__EnglishGraphic__

/***
 * \file EnglishGraphic.h
 * \brief Notre classe EnglishGraphic
 * \author CHARDAN Anaël
 * \author DAMEY Jérémy
 * \date 09/03/2014
 */

#include <iostream>
#include "LanguageGraphic.h"

/*! \class EnglishGraphic
 * \brief Classe pour l'affichage du jeu en anglais
 */

class EnglishGraphic : public LanguageGraphic {
public :
    /*!
     * \brief Configurer le sprite par rapport à l'image
     *
     * Affichage de l'image choisi
     */
    virtual void setImageToSprite();

    /*!
     * \brief Configurer le sprite par rapport à l'image
     *
     * Affichage du jeu en Anglais
     */
    virtual ~EnglishGraphic();
};

#endif /* defined(__PuruPuruDigger__EnglishGraphic__) */
```

```
/**\n * \file EnglishGraphic.cpp\n * \brief Notre classe EnglishGraphic\n * \author CHARDAN Anaël\n * \author DAMEY Jérémy\n * \date 09/03/2014\n */\n\n#include "EnglishGraphic.h"\n#include "../Constantes.h"\n\nEnglishGraphic::~EnglishGraphic() { }\n\nvoid\nEnglishGraphic::setImageToSprite() {\n    LanguageGraphic::setImageToSprite();\n    my_sprite.SetSubRect( sf::IntRect ( ENGLISHSX, SPRITELANGUEBEGIN,\n        ENGLISHEX, SPRITELANGUEHEIGHT ) );\n    my_sprite.SetPosition(ENGLISHX, CHOICELANGUEHIGH);\n    my_sprite.Resize( LANGUEWIDTH, LANGUEHEIGHT);\n}
```

```
#ifndef __PuruPuruDigger__FrenchGraphic__
#define __PuruPuruDigger__FrenchGraphic__

/***
 * \file FrenchGraphic.h
 * \brief Notre classe FrenchGraphic
 * \author CHARDAN Anaël
 * \author DAMEY Jérémy
 * \date 09/03/2014
 */

#include <iostream>
#include "LanguageGraphic.h"

/*! \class FrenchGraphic
 * \brief Classe pour l'affichage du jeu en français
 */

class FrenchGraphic : public LanguageGraphic {
public :
    /*!
     * \brief Configurer le sprite par rapport à l'image
     *
     * Affichage de l'image choisi
     */
    virtual void setImageToSprite();

    /*!
     * \brief Configurer le sprite par rapport à l'image
     *
     * Affichage du jeu en Français
     */
    virtual ~FrenchGraphic();
};

#endif /* defined(__PuruPuruDigger__FrenchGraphic__) */
```

```
/**  
 * \file FrenchGraphic.cpp  
 * \brief Notre classe FrenchGraphic  
 * \author CHARDAN Anaël  
 * \author DAMEY Jérémy  
 * \date 09/03/2014  
 */  
  
#include "FrenchGraphic.h"  
#include "../Constantes.h"  
  
FrenchGraphic::~FrenchGraphic() { }  
  
void  
FrenchGraphic::setImageToSprite() {  
    LanguageGraphic::setImageToSprite();  
    my_sprite.SetSubRect( sf::IntRect ( FRENCHSX, SPRITELANGUEBEGIN, FRENCHEX,  
        SPRITELANGUEHEIGHT ) );  
    my_sprite.SetPosition(FRENCHX, CHOICELANGUEHIGH);  
    my_sprite.Resize( LANGUEWIDTH, LANGUEHEIGHT );  
}
```

```
#ifndef __PuruPuruDigger__DeutschGraphic__
#define __PuruPuruDigger__DeutschGraphic__

/***
 * \file DeutschGraphic.h
 * \brief Notre classe DeutschGraphic
 * \author CHARDAN Anaël
 * \author DAMEY Jérémy
 * \date 09/03/2014
 */

#include <iostream>
#include "LanguageGraphic.h"

/*! \class DeutschGraphic
 * \brief Classe pour l'affichage du jeu en Allemand
 */

class DeutschGraphic : public LanguageGraphic {
public :
    /*!
     * \brief Configurer le sprite par rapport à l'image
     *
     * Affichage de l'image choisi
     */
    virtual void setImageToSprite();

    /*!
     * \brief Configurer le sprite par rapport à l'image
     *
     * Affichage de tout le jeu en Allemand
     */
    virtual ~DeutschGraphic();
};

#endif /* defined(__PuruPuruDigger__DeutschGraphic__) */
```

```
/**\n * \file DeutschGraphic.cpp\n * \brief Notre classe DeutschGraphic\n * \author CHARDAN Anaël\n * \author DAMEY Jérémy\n * \date 09/03/2014\n */\n\n#include "DeutschGraphic.h"\n#include "../Constantes.h"\n\nDeutschGraphic::~DeutschGraphic() { }\n\nvoid\nDeutschGraphic::setImageToSprite() {\n    LanguageGraphic::setImageToSprite();\n    my_sprite.SetSubRect( sf::IntRect ( DEUTSCHSX, SPRITELANGUEBEGIN,\n        DEUTSCHEX, SPRITELANGUEHEIGHT ) );\n    my_sprite.SetPosition(DEUTSCHX, CHOICELANGUEHIGH);\n    my_sprite.Resize( LANGUEWIDTH, LANGUEHEIGHT);\n}
```

```
#ifndef __PuruPuruDigger__GraphicSound__
#define __PuruPuruDigger__GraphicSound__

/***
 * \file GraphicSound.h
 * \brief Notre classe GraphicSound
 * \author CHARDAN Anaël
 * \author DAMEY Jérémy
 * \date 09/03/2014
 */

#include <iostream>
#include "GraphicAudibleElement.h"

/*! \class GraphicSound
 * \brief Classe pour l'affichage des sons
 */

class GraphicSound : public GraphicAudibleElement {
public :
    /**
     * \brief configurer le sprite par rapport à l'image
     *
     * Destructeur de la classe GraohicElement
     */
    virtual void setImageToSprite();

    /**
     * \brief Destructeur
     *
     * Destructeur de la classe GraohicSound
     */
    virtual ~GraphicSound();

    /**
     * \brief Inverse le son
     *
     * Si le son est activé, cela le désactive et inversement
     */
    virtual void reverse();
};

#endif /* defined(__PuruPuruDigger__GraphicSound__) */
```

```
/**\n * \file GraphicSound.cpp\n * \brief Notre classe GraphicSound\n * \author CHARDAN Anaël\n * \author DAMEY Jérémy\n * \date 09/03/2014\n */\n\n#include "GraphicSound.h"\n#include "../Constantes.h"\n\nGraphicSound::~GraphicSound() { }\n\nvoid\nGraphicSound::setImageToSprite() {\n    GraphicAudibleElement::setImageToSprite();\n    my_sprite.SetSubRect( sf::IntRect( SOUNDSX, ICONSPRITEBEGIN, SOUNDONEX,\n        ICONSPRITEHEIGHT ) );\n    my_sprite.SetPosition(SOUNDX, ICONY);\n    my_sprite.Resize( ICONWIDTH, ICONHEIGHT);\n}\n\nvoid\nGraphicSound::reverse() {\n\n    if ( !my_context->isEnableSound() ) {\n        my_sprite.SetSubRect( sf::IntRect( SOUNDSX, ICONSPRITEBEGIN, SOUNDONEX\n            , ICONSPRITEHEIGHT ) );\n        my_context->setSound( true );\n    } else {\n        my_sprite.SetSubRect( sf::IntRect( SOUNDsx, ICONSPRITEBEGIN,\n            SOUNDOFFEX, ICONSPRITEHEIGHT ) );\n        my_context->setSound( false );\n    }\n}
```

```
#ifndef __PuruPuruDigger__GraphicMusic__
#define __PuruPuruDigger__GraphicMusic__

/***
 * \file GraphicMusic.h
 * \brief Notre classe GraphicMusic
 * \author CHARDAN Anaël
 * \author DAMEY Jérémy
 * \date 09/03/2014
 */

#include <iostream>
#include "GraphicAudibleElement.h"

/*! \class GraphicMusic
 * \brief Classe pour l'affichage des boutons representant la musique
 */

class GraphicMusic : public GraphicAudibleElement {
public :
    /*!
     * \brief Configurer le sprite par rapport à l'image
     */
    virtual void setImageToSprite();

    /*!
     * \brief Constructeur par defaut
     *
     * Constructeur de la classe GraphicMusic
     */
    GraphicMusic();

    /*!
     * \brief Inverse la musique
     *
     * Si musique est activé, cela la desactive et inversement
     */
    virtual void reverse();

    /*!
     * \brief Gestion de la musique avec la souris
     *
     * \param[in] sf::Event event
     */
    virtual void mouseButtonPressed(sf::Event event);
};

#endif /* defined(__PuruPuruDigger__GraphicMusic__) */
```

```
/**\n * \file GraphicMusic.cpp\n * \brief Notre classe GraphicMusic\n * \author CHARDAN Anaël\n * \author DAMEY Jérémy\n * \date 09/03/2014\n */\n\n#include "GraphicMusic.h"\n#include "../Constantes.h"\n\nGraphicMusic::GraphicMusic() {\n}\n\nvoid GraphicMusic::setImageToSprite() {\n    GraphicAudibleElement::setImageToSprite();\n    my_sprite.SetSubRect( sf::IntRect( MUSICONSX, ICONSPRITEBEGIN, MUSICONEX,\n        ICONSPRITEHEIGHT ) );\n    my_sprite.SetPosition(MUSICX, ICONY);\n    my_sprite.Resize( ICONWIDTH, ICONHEIGHT);\n}\n\nvoid GraphicMusic::reverse() {\n    if ( !my_context->isEnableMusic() ) {\n        my_sprite.SetSubRect( sf::IntRect( MUSICONSX, ICONSPRITEBEGIN,\n            MUSICONEX, ICONSPRITEHEIGHT ) );\n        my_context->setMusic( true );\n    } else {\n        my_sprite.SetSubRect( sf::IntRect( MUSICOFFSX, ICONSPRITEBEGIN,\n            MUSICOFFEX, ICONSPRITEHEIGHT ) );\n        my_context->setMusic( false );\n    }\n}\n\n/** Events Subscriber */\n\nvoid GraphicMusic::mouseButtonPressed( sf::Event event ) {\n    if ( isInZone( event.MouseButton.X, event.MouseButton.Y ) ) {\n        reverse();\n\n        if ( my_context->isPlaying() ) {\n            SoundManager::getInstance()->playMusic();\n            if ( !my_context->isEnableMusic() ) {\n                SoundManager::getInstance()->pauseMusic();\n            }\n        }\n    }\n}
```

```
#ifndef __PuruPuruDigger__GraphicAudibleElement__
#define __PuruPuruDigger__GraphicAudibleElement__

/***
 * \file GraphicAudibleElement.h
 * \brief Notre classe GraphicAudibleElement
 * \author CHARDAN Anaël
 * \author DAMEY Jérémy
 * \date 09/03/2014
 */

#include <iostream>
#include "GraphicElement.h"

/*! \class GraphicAudibleElement
 * \brief Classe pour le chargement des sons
 */

class GraphicAudibleElement : public GraphicElement {
protected :
    static sf::Image my_image ; /*!< image de pour la musique */

public :
    /*!
     * \brief Configurer le sprite par rapport à l'image
     *
     * Affichage de l'image choisie
     */
    virtual void setImageToSprite();

    virtual void reverse() = 0;

    /*!
     * \brief Configurer le sprite par rapport à l'image
     *
     * Affichage du thème choisi
     *
     * \param[in] std::string theme
     */
    virtual void changeTheme( std::string theme );

    /*!
     * \brief Changer le bouton s'il y a ou non le son
     *
     * Affichage du bouton demandé
     */
    virtual void mouseButtonPressed(sf::Event event);
};

#endif /* defined(__PuruPuruDigger__GraphicAudibleElement__) */
```

```
/**  
 * \file GraphicAudibleElement.cpp  
 * \brief Notre classe GraphicAudibleElement  
 * \author CHARDAN Anaël  
 * \author DAMEY Jérémy  
 * \date 09/03/2014  
 */  
  
#include "GraphicAudibleElement.h"  
#include "../Constantes.h"  
  
sf::Image GraphicAudibleElement::my_image;  
  
void GraphicAudibleElement::changeTheme( std::string theme ) {  
    std::string myimage;  
#ifdef __linux__  
    myimage = "Ressources/Pictures/icon.png";  
#else  
    myimage = "icon.png";  
#endif  
    if ( !my_image.LoadFromFile( myimage.c_str() ) ) {  
        std::cerr << " Error when loading icon image " << std::endl;  
    } else {  
        setImageToSprite();  
    }  
}  
  
void GraphicAudibleElement::setImageToSprite() {  
    my_image.CreateMaskFromColor(sf::Color(0, 55, 97));  
    my_sprite.SetImage(my_image);  
    my_sprite.Resize( ICONWIDTH, ICONHEIGHT );  
}  
  
/* Events Subscriber */  
  
void GraphicAudibleElement::mouseButtonPressed( sf::Event event ) {  
    if ( isInZone( event.MouseButton.X, event.MouseButton.Y ) ) {  
        reverse();  
    }  
}
```

```
#ifndef __PuruPuruDigger__GoldGraphic__
#define __PuruPuruDigger__GoldGraphic__

/***
 * \file GoldGraphic.h
 * \brief Notre classe GoldGraphic
 * \author CHARDAN Anaël
 * \author DAMEY Jérémy
 * \date 09/03/2014
 */

#include <iostream>
#include "CellTextGraphic.h"

/*! \class GoldGraphic
 * \brief Classe pour l'affichage du digger
 */

class GoldGraphic : public CellTextGraphic {
public :
    /*!
     * \brief Destructeur
     *
     * Destructeur de la classe GoldGraphic
     */
    virtual ~GoldGraphic();

    /*!
     * \brief Configurer le sprite par rapport à l'image
     *
     * Affichage de l'image choisie
     */
    virtual void setImageToSprite();
};

#endif /* defined(__PuruPuruDigger__GoldGraphic__) */
```

```
/**\n * \file GoldGraphic.pdf\n * \brief Notre classe GoldGraphic\n * \author CHARDAN Anaël\n * \author DAMEY Jérémy\n * \date 09/03/2014\n */\n\n#include "GoldGraphic.h"\n#include "../Constantes.h"\n\nGoldGraphic::~GoldGraphic() { }\n\nvoid\nGoldGraphic::setImageToSprite() {\n    CellTextGraphic::setImageToSprite();\n    my_sprite.SetSubRect( sf::IntRect ( GOLDSX, SPRITECASEBEGIN, GOLDEX,\n        SPRITECASEHEIGHT ) );\n    my_sprite.Resize( CASEWITDH, CASEHEIGHT );\n}\n
```

```
#ifndef __PuruPuruDigger__EmptyGraphic__
#define __PuruPuruDigger__EmptyGraphic__

/***
 * \file EmptyGraphic.h
 * \brief Notre classe EmptyGraphic
 * \author CHARDAN Anaël
 * \author DAMEY Jérémy
 * \date 09/03/2014
 */

#include <iostream>
#include "CellBaseGraphic.h"

#include "GraphicElement.h"

/*! \class EmptyGraphic
 * \brief Classe pour l'affichage d'une case vide
 */

class EmptyGraphic : public CellBaseGraphic {
public :
    /**
     * \brief Destructeur
     *
     * Destructeur de la classe EmptyGraphic
     */
    virtual ~EmptyGraphic();

    /**
     * \brief Configurer le sprite par rapport à l'image
     *
     * Affichage e l'image choisi
     */
    virtual void setImageToSprite();
};

#endif /* defined(__PuruPuruDigger__EmptyGraphic__) */
```

```
/**\n * \file EmptyGraphic.pdf\n * \brief Notre classe EmptyGraphic\n * \author CHARDAN Anaël\n * \author DAMEY Jérémy\n * \date 09/03/2014\n */\n\n#include "EmptyGraphic.h"\n#include "../Constantes.h"\n\nEmptyGraphic::~EmptyGraphic() { }\n\nvoid\nEmptyGraphic::setImageToSprite()\n{\n    CellBaseGraphic::setImageToSprite();\n    my_sprite.SetSubRect( sf::IntRect ( EMPTYSX, SPRITECASEBEGIN, EMPTYEX,\n        SPRITECASEHEIGHT ) );\n    my_sprite.Resize( CASEWITDH, CASEHEIGHT );\n}
```

```
#ifndef __PuruPuruDigger__CellTextGraphic__
#define __PuruPuruDigger__CellTextGraphic__

/***
 * \file CellTextGraphic.h
 * \brief Notre classe CellTextGraphic
 * \author CHARDAN Anaël
 * \author DAMEY Jérémy
 * \date 09/03/2014
 */

#include <iostream>
#include "CellBaseGraphic.h"

/*! \class CellTextGraphic
 * \brief Classe pour l'affichage du texte
 */

class CellTextGraphic : public CellBaseGraphic {
protected :
    static sf::Font my_font; /*!< police d'écriture */
    sf::String my_string; /*!< chaîne de caractère */

public :
    /*!
     * \brief Affiche le texte demander
     *
     * Affiche le texte demander
     *
     * \param[in] int x
     * \param[in] int y
     * \param[in] sf::RenderWindow * _window
     * \param[in] std::string _string
     */
    virtual void setSpriteAndDraw( int x, int y, sf::RenderWindow * _window, std::string _string );

    /*!
     * \brief Configurer le sprite par rapport à l'image
     *
     * Affichage de l'image choisi
     */
    virtual void setImageToSprite();

    /*!
     * \brief Configurer le sprite par rapport à l'image
     *
     * Affichage du thème choisi
     *
     * \param[in] std::string theme
     */
    virtual void changeTheme( std::string theme );
};

#endif /* defined(__PuruPuruDigger__CellTextGraphic__) */
```

```
/**\n * \file CellTextGraphic.cpp\n * \brief Notre classe CellTextGraphic\n * \author CHARDAN Anaël\n * \author DAMEY Jérémy\n * \date 09/03/2014\n */\n\n#include "CellTextGraphic.h"\n\nsf::Font CellTextGraphic::my_font;\n\nvoid CellTextGraphic::changeTheme( std::string theme ) {\n    std::string myfont;\n#ifndef __linux__\n    if ( theme == "teacher" ) {\n        myfont = "Ressources/Font/" + theme + "_arial.ttf";\n    } else {\n        myfont = "Ressources/Font/" + theme + "_value.ttf";\n    }\n#else\n    if ( theme == "teacher" ) {\n        myfont = theme + "_arial.ttf";\n    } else {\n        myfont = theme + "_value.ttf";\n    }\n#endif\n\n    if ( !my_font.LoadFromFile( myfont.c_str() ) ) {\n        std::cerr << " Error when loading value font " << std::endl;\n    } else {\n        my_string.SetFont(my_font);\n        if ( theme == "ananas" ) {\n            my_string.SetSize(23);\n            my_stringSetColor(sf::Color(255,255,255));\n        } else {\n            my_string.SetSize(20);\n            my_stringSetColor(sf::Color(0,0,0));\n            my_string.setStyle( sf::String::Bold );\n        }\n        CellBaseGraphic::changeTheme( theme );\n    }\n}\n\nvoid CellTextGraphic::setSpriteAndDraw(int x, int y, sf::RenderWindow* _window,\n, std::string _string) {\n    GraphicElement::setSpriteAndDraw( x, y, _window);\n    my_string.SetText(_string);\n    my_string.SetPosition( my_sprite.GetPosition().x + ( my_sprite.GetSize().x / 2 ) - ( my_string.GetRect().GetWidth() / 2 ), my_sprite.GetPosition().y + ( my_sprite.GetSize().y / 2 ) - ( my_string.GetRect().GetHeight() / 2 ) );\n    _window->Draw(my_string);\n}\n\nvoid\nCellTextGraphic::setImageToSprite() {\n    CellBaseGraphic::setImageToSprite();\n}
```

```
#ifndef __PuruPuruDigger__ButtonGraphic__
#define __PuruPuruDigger__ButtonGraphic__

/***
 * \file ButtonGraphic.h
 * \brief Notre classe ButtonGraphic
 * \author CHARDAN Anaël
 * \author DAMEY Jérémy
 * \date 09/03/2014
 */

#include <iostream>
#include "GraphicElement.h"

/*! \class ButtonGraphic
 * \brief Classe pour l'affichage des boutons
 */

class ButtonGraphic : public GraphicElement {
protected :
    static sf::Image my_image; /*!< image du bouton */
    static sf::Font my_font; /*!< fond du bouton */

    sf::String my_string; /*!< texte de l'image */

public :
    /*!
     * \brief Configurer le sprite par rapport à l'image
     *
     * Affichage du bouton
     */
    virtual void setImageToSprite();

    /*!
     * \brief Configurer le sprite par rapport à l'image
     *
     * Affichage du thème choisi
     *
     * \param[in] int x
     * \param[in] int y
     * \param[in] sf::RenderWindow * _window
     * \param[in] std::string
     */
    virtual void setSpriteAndDraw( int x, int y, sf::RenderWindow * _window,
                                  std::string );

    /*
     * \brief Affichage des informations du sprite demandé
     *
     * Affichage des informations du sprite demandé
     */
    void setHover();

    /*
     * \brief Affichage du sprite à sa position initiale
     *
     * Affiche le sprite demandé à sa position de départ
     */
    void reset();
}
```

```
/*
 * \brief Dessine le sprite et affiche le texte associé
 *
 * Dessine le sprite et affiche le texte associé
 */
virtual void draw(sf::RenderWindow* _window) const;

/*
 * \brief Change le boutton de couleur
 *
 * Le boutton change de couleur quand on passe dessus avec la souris
 */
virtual void mouseMoved(sf::Event event);

/*
 * \brief Ecoute d'un son
 *
 * Un son est associé en fonction du bouton
 */
virtual void mouseButtonPressed(sf::Event event);

/*
 * \brief Changement de thème
 *
 * Possibilité dans le menu option de changer de thème
 */
virtual void changeTheme( std::string theme );

};

#endif /* defined(__PuruPuruDigger__ButtonGraphic__) */
```

```
/**\n * \file ButtonGraphic.cpp\n * \brief Notre classe ButtonGraphic\n * \author CHARDAN Anaël\n * \author DAMEY Jérémy\n * \date 09/03/2014\n */\n\n#include "ButtonGraphic.h"\n#include "../Constantes.h"\n\nsf::Image ButtonGraphic::my_image;\nsf::Font ButtonGraphic::my_font;\n\nvoid ButtonGraphic::changeTheme( std::string theme ) {\n    std::string myimage;\n    std::string myfont;\n#ifndef __linux__\n    myimage = "Ressources/Pictures/" + theme + "_button.png";\n    myfont = "Ressources/Font/" + theme + "_buttonFont.ttf";\n\n    if ( theme == "teacher" ) {\n        myfont = "Ressources/Font/" + theme + "_arial.ttf";\n    }\n#else\n    myimage = theme + "_button.png";\n    myfont = theme + "_buttonFont.ttf";\n\n    if ( theme == "teacher" ) {\n        myfont = theme + "_arial.ttf";\n    }\n#endif\n    if ( !my_image.LoadFromFile( myimage.c_str() ) || ( !my_font.LoadFromFile(\n        myfont.c_str() ) ) ) {\n        std::cerr << " Error when loading button images and fonts " << std::endl;\n    } else {\n        my_string.SetFont( my_font );\n        my_string.SetSize(30);\n\n        if ( theme == "ananas" ) {\n            my_stringSetColor(sf::Color(251,210,98));\n        } else {\n            my_stringSetColor(sf::Color(0,0,0));\n        }\n\n        setImageToSprite();\n    }\n}\n\nvoid ButtonGraphic::setImageToSprite() {\n    my_image.CreateMaskFromColor(sf::Color(0, 55, 97));\n    my_sprite.setImage(my_image);\n    my_sprite.SetSubRect( sf::IntRect( BUTTONNORMSX, BUTTONCASEBEGIN,\n        BUTTONNORMEX, BUTTONCASEHEIGHT ) );\n    my_sprite.Resize( BUTTONWIDTH, BUTTONHEIGHT );\n}\n\nvoid ButtonGraphic::setHover() {
```

```
    my_sprite.SetSubRect( sf::IntRect( BUTTONHOVESX, BUTTONCASEBEGIN,
        BUTTONHOVEEX, BUTTONCASEHEIGHT ) );
}

void ButtonGraphic::reset() {
    my_sprite.SetSubRect( sf::IntRect( BUTTONNORMSX, BUTTONCASEBEGIN,
        BUTTONNORMEX, BUTTONCASEHEIGHT ) );
}

void ButtonGraphic::setSpriteAndDraw(int x, int y, sf::RenderWindow* _window,
    std::string _string) {
    GraphicElement::setSpriteAndDraw( x, y, _window );
    my_string.SetText(_string);
    my_string.SetPosition( my_sprite.GetPosition().x + ( my_sprite.GetSize().x
        / 2 ) - ( my_string.GetRect().GetWidth() / 2 ), my_sprite.GetPosition()
        .y + ( my_sprite.GetSize().y / 2 ) - ( my_string.GetRect().GetHeight()
        / 2 ) );
    _window->Draw(my_string);
}

void ButtonGraphic::draw(sf::RenderWindow *_window) const {
    _window->Draw(my_sprite);
    _window->Draw(my_string);
}

/** Events Subscriber */

void ButtonGraphic::mouseMoved( sf::Event event ) {
    if ( isInZone( event.MouseMove.X, event.MouseMove.Y ) ) {
        setHover();
    } else {
        reset();
    }
}

void ButtonGraphic::mouseButtonPressed( sf::Event event ) {
    if ( isInZone( event.MouseButton.X, event.MouseButton.Y ) ) {
        SoundManager::getInstance()->clickButton();
    }
    reset();
}
```

```
#ifndef __PuruPuruDigger__BombGraphic__
#define __PuruPuruDigger__BombGraphic__

/***
 * \file BombGraphic.h
 * \brief Notre classe BombGraphic
 * \author CHARDAN Anaël
 * \author DAMEY Jérémy
 * \date 09/03/2014
 */

#include <iostream>
#include "CellBaseGraphic.h"
#include "GraphicElement.h"

/*! \class BombGraphic
 * \brief Classe l'affichage des bombes
 */

class BombGraphic : public CellBaseGraphic {
public :
    /**
     * \brief Destructeur
     *
     * Destructeur de la classe BombGraphic
     */
    virtual ~BombGraphic();

    /**
     * \brief Configurer le sprite par rapport à l'image
     *
     * Affichage de l'image bomb
     */
    virtual void setImageToSprite();
};

#endif /* defined(__PuruPuruDigger__BombGraphic__) */
```

```
/**\n * \file BombGraphic.cpp\n * \brief Notre classe BombGraphic\n * \author CHARDAN Anaël\n * \author DAMEY Jérémy\n * \date 09/03/2014\n */\n\n#include "BombGraphic.h"\n#include "../Constantes.h"\n\nBombGraphic::~BombGraphic() { }\n\nvoid BombGraphic::setImageToSprite()\n{\n    CellBaseGraphic::setImageToSprite();\n    my_sprite.SetSubRect( sf::IntRect ( BOMBSX, SPRITECASEBEGIN, BOMBEX,\n        SPRITECASEHEIGHT ) );\n    my_sprite.Resize( CASEWITDH, CASEHEIGHT );\n}
```

```
#ifndef __PuruPuruDigger__BackgroundGraphic__
#define __PuruPuruDigger__BackgroundGraphic__

/***
 * \file BackgroundGraphic.h
 * \brief Notre classe BacgroundGraphic
 * \author CHARDAN Anaël
 * \author DAMEY Jérémy
 * \date 09/03/2014
 */

#include <iostream>
#include "GraphicElement.h"

/*! \class BacgroundGraphic
 * \brief Classe pour le changement de thème
 */

class BackgroundGraphic : public GraphicElement {
protected :
    static sf::Image my_image; /*!< image de fond */

public :
    /*!
     * \brief Configurer le sprite par rapport à l'image
     *
     * Affichage du thème choisi
     */
    virtual void setImageToSprite();

    /*!
     * \brief Permet de changer le thème du jeu
     *
     * Prise en compte du nouveau thème choisi
     *
     * \param[in] std::string theme
     */
    void changeTheme( std::string theme );
};

#endif /* defined(__PuruPuruDigger__BackgroundGraphic__) */
```

```
/**\n * \file BackgroundGraphic.cpp\n * \brief Notre classe BackgroundGraphic\n * \author CHARDAN Anaël\n * \author DAMEY Jérémy\n * \date 09/03/2014\n */\n\n#include "BackgroundGraphic.h"\n#include "../Constantes.h"\n\nsf::Image BackgroundGraphic::my_image;\n\n\nvoid BackgroundGraphic::changeTheme( std::string theme ) {\n    std::string myimage;\n#ifndef __linux__\n    myimage = "Ressources/Pictures/" + theme + "_wallpapper.png";\n#else\n    myimage = theme + "_wallpapper.png";\n#endif\n    if ( !my_image.LoadFromFile( myimage.c_str() ) ) {\n        std::cerr << " Error when loading background image " << std::endl;\n    } else {\n        setImageToSprite();\n    }\n}\n\nvoid BackgroundGraphic::setImageToSprite() {\n    my_sprite.SetImage(my_image);\n    my_sprite.Resize( WINDOWWIDTH, WINDOWHEIGHT );\n    my_sprite.SetPosition(0,0);\n}
```

```
#ifndef __PuruPuruDigger_Bomb__
#define __PuruPuruDigger_Bomb__

/***
 * \file Bomb.h
 * \brief Notre classe Bomb
 * \author CHARDAN Anaël
 * \author DAMEY Jérémy
 * \date 09/03/2014
 */

#include <iostream>
#include "CellBase.h"

/*! \class Bomb
 * \brief Classe modélisant ce qu'est une Bomb
 */

class Bomb : public CellBase {
public:

    /*!
     * \brief Constructeur
     *
     * Constructeur de la classe Bomb
     */
    Bomb();

    /*!
     * \brief Constructeur paramétré
     *
     * Constructeur paramétré de la classe Bomb
     * \param[in] int x
     * \param[in] int y
     */
    Bomb( int x, int y );

    /*!
     * \brief Constructeur par copie
     *
     * Constructeur par copie de la classe Bomb
     * \param[in] Bomb b
     */
    Bomb( const Bomb &b);

    /*!
     * \brief Destructeur
     *
     * Destructeur de la classe fille Bomb
     */
    virtual ~Bomb();

    /*!
     * \brief Opérateur d'affectation pour recopier une case
     *
     * \param[b] Bomb b : opérateur d'affectation pour recopier une
     * Bomb
     */
    virtual Bomb& operator=(const Bomb & b);
}
```

};

#endif /* defined(__PuruPuruDigger__Bomb__) */

```
/**\n * \file Bomb.cpp\n * \brief Notre classe Bomb\n * \author CHARDAN Anaël\n * \author DAMEY Jérémy\n * \date 09/03/2014\n */\n\n#include "Bomb.h"\n\n/*=====\n Les Constructeurs\n=====*/\n\nBomb::Bomb() : CellBase() { }\n\nBomb::Bomb( int x, int y ) : CellBase(x,y) { }\n\nBomb::Bomb(const Bomb &b) {\n    CellBase::toClone(b);\n}\n\n/*=====\n Le Destructeur\n=====*/\n\nBomb::~Bomb() {} \n\n/*=====\n Les opérateurs\n=====*/\n\nBomb&\nBomb::operator=(const Bomb &b) {\n    if ( this != &b ){\n        CellBase::toClone(b);\n    }\n    return *this;\n}
```

```
#ifndef __purpurudigger__CellBase__
#define __purpurudigger__CellBase__

/***
 * \file CellBase.h
 * \brief Notre classe CellBase
 * \author CHARDAN Anaël
 * \author DAMEY Jérémie
 * \date 09/03/2014
 */

#include <iostream>
#include <string>
#include "../Utils.h"
#include "../Constantes.h"

/*! \class CellBase
 * \brief Classe modélisant ce qu'est une case
 */

class CellBase {

protected :

    std::string my_type;           /*!< le type de ma case */
    int my_x;                     /*!< Le x de ma case */
    int my_y;                     /*!< Le y de ma case */

    virtual void toClone(const CellBase &c);

public :

    /**
     * \brief Constructeur
     *
     * Constructeur de la classe CellBase
     */
    CellBase();

    /**
     * \brief Constructeur paramétré
     *
     * Constructeur paramétré de la classe CellBase
     */
    CellBase( int x, int y );

    /**
     * \brief Constructeur par copie
     *
     * Constructeur par copie de la classe CellBase
     */
    CellBase(const CellBase &c);

    /**
     * \brief Destructeur
     *
     * Destructeur de la classe mère CellBase
     */
    virtual ~CellBase();

    /**

```

```
* \brief retourne la position en X de la case
*
* \return l'entier positionnant notre case en X
*/
int getX() const;

<賓!
* \brief retourne la position en Y de la case
*
* \return l'entier positionnant notre case en Y
*/
int getY() const;

<賓!
* \brief Pour positionner notre case dans la grille
*
* param[in] int x : la position verticale de notre case
*/
void setX( int x );

<賓!
* \brief Pour positionner notre case dans la grille
*
* param[in] int x : la position horizontale de notre case
*/
void setY( int y );

<賓!
* \brief Opérateur d'affectation pour recopier une case
*
* param[c] CellBase c : opérateur d'affectation pour recopier une
* case
*/
virtual CellBase& operator=(const CellBase &c);

std::string getType();

};

#endif /* defined(__purpurudigger__CellBase__) */
```

```
/**\n * \file CellBase.cpp\n * \brief Notre classe CellBase\n * \author CHARDAN Anaël\n * \author DAMEY Jérémy\n * \date 09/03/2014\n */\n\n#include "CellBase.h"\n#include <typeinfo>\n\n/*=====*\nLes Constructeurs\n=====*/\n\nCellBase::CellBase() : my_x(0), my_y(0) { }\n\nCellBase::CellBase( int x, int y ) : my_x(x), my_y(y) { }\n\nCellBase::CellBase(const CellBase &c ) {\n    toClone(c);\n}\n\n/*=====*\nLe Destructeur\n=====*/\n\nCellBase::~CellBase() {} \n\n/*=====*\nLes fonctions non destiné à être redéfini dans les classes filles\n=====*/\n\nint\nCellBase::getX() const {\n    return my_x;\n}\n\nint\nCellBase::getY() const {\n    return my_y;\n}\n\nvoid\nCellBase::setX( int x ) {\n    my_x = x;\n}\n\nvoid\nCellBase::setY( int y ) {\n    my_y = y;\n}\n\n/*=====*\nLes fonctions destinés à être redéfinie dans les classes filles concernés\n=====*/\n\nCellBase&\nCellBase::operator=(const CellBase &c) {
```

```
if ( this != &c ) {
    toClone(c);
}
return *this;
}

std::string
CellBase::getType() {
    if ( my_type == "" ) {
        my_type = typeid(*this).name();
        while ( my_type[0] >= '0' && my_type[0] <= '9' ) {
            my_type.erase(0,1);
        }
    }
    return my_type;
}

void
CellBase::toClone(const CellBase &c) {
    my_x = c.my_x;
    my_y = c.my_y;
    my_type = c.my_type;
}
```

```
#ifndef __purpurudigger__Digger__
#define __purpurudigger__Digger__

/***
 * \file Digger.h
 * \brief Notre classe Digger
 * \author CHARDAN Anaël
 * \author DAMEY Jérémy
 * \date 09/03/2014
 */

#include <iostream>
#include "CellBase.h"

/*! \class Digger
 * \brief Classe modélisant ce qu'est un digger
 */
class Digger : public CellBase {
private :
    int my_life; /*!< le nombre de vie du digger */
    void toClone( const Digger &d);

public :
    //Les constructeurs
    /*!
     * \brief Constructeur
     *
     * Constructeur de la classe Digger
     */
    Digger();

    /*!
     * \brief Constructeur paramétré
     *
     * Constructeur paramétré de la classe Digger
     * \param[in] x
     * \param[in] y
     */
    Digger( int x, int y );

    /*!
     * \brief Constructeur par copie
     *
     * \param[in] Digger d
     * \param[out] Digger d
     */
    Digger(const Digger &d );

    /*!
     * \brief Destructeur
     *
     * Destructeur de la classe fille DiggerCell
     */
    virtual ~Digger();

    /*!
     * \brief Retourne la vie du Digger
     */
}
```

```
* \return my_life, la vie du digger pour éviter l'abstraction de la
* classe
*/
int getLife() const;

/*! \brief Ajouter une vie au digger
*
*/
void addLife();

/*! \brief Enlever une vie au digger
*
*/
void lostLife();

/*! \brief Remets à 3 la vie du digger
*
*/
virtual void resetLife();
/*! \brief Opérateur d'affectation pour recopier une Digger
*
* \param[in] Digger d : opérateur d'affectation pour recopier un
* Digger
*/
virtual Digger& operator=( const Digger &d );

};

#endif /* defined(__purpurudigger__Digger__) */
```

```
/**\n * \file Digger.cpp\n * \brief Notre classe Digger\n * \author CHARDAN Anaël\n * \author DAMEY Jérémy\n * \date 09/03/2014\n */\n\n#include "Digger.h"\n#include "../Constantes.h"\n\nusing namespace std;\n\n/*=====*\nLes Constructeurs\n******/\n\nDigger::Digger() : CellBase(), my_life(3) { }\n\nDigger::Digger( int x, int y ) : CellBase(x,y), my_life(3) { }\n\nDigger::Digger( const Digger &d ) {\n    toClone(d);\n}\n\n/*=====*\nLe Destructeur\n******/\n\nDigger::~Digger() { }\n\n/*=====*\nLes méthodes\n******/\n\nint\nDigger::getLife() const {\n    return my_life;\n}\n\nvoid\nDigger::addLife() {\n    if ( my_life < 3 )\n        my_life++;\n}\n\nvoid\nDigger::lostLife() {\n    if ( my_life >= 0 )\n        my_life--;\n}\n\nvoid\nDigger::resetLife() {\n    my_life = 3;\n}\n\nDigger&\nDigger::operator=( const Digger &d ) {\n    if ( this != &d ) {\n
```

```
        toClone(d);
    }
    return *this;
}

void
Digger::toClone( const Digger &d ) {
    CellBase::toClone(d);
    my_life = d.my_life;
}
```

```
#ifndef __PuruPuruDigger__EmptyCell__
#define __PuruPuruDigger__EmptyCell__

/***
 * \file EmptyCell.h
 * \brief Notre classe EmptyCell
 * \author CHARDAN Anaël
 * \author DAMEY Jérémy
 * \date 09/03/2014
 */

#include <iostream>
#include "CellBase.h"

/*! \class EmptyCell
 * \brief Classe modélisant ce qu'est une EmptyCell
 */
class EmptyCell : public CellBase {

public:

    /*!
     * \brief Constructeur
     *
     * Constructeur de la classe EmptyCell
     */
    EmptyCell();

    /*!
     * \brief Constructeur paramétré
     *
     * Constructeur paramétré de la classe EmptyCell
     */
    EmptyCell( int x, int y );

    /*!
     * \brief Constructeur par copie
     *
     * Constructeur par copie de la classe EmptyCell
     */
    EmptyCell( const EmptyCell &b);

    /*!
     * \brief Destructeur
     *
     * Destructeur de la classe fille EmptyCell
     */
    virtual ~EmptyCell();

    /*!
     * \brief Opérateur d'affectation pour recopier une case
     *
     * \param[b] EmptyCell b : opérateur d'affectation pour recopier
     * une EmptyCell
     */
    virtual EmptyCell& operator=(const EmptyCell & b);

};

#endif /* defined(__PuruPuruDigger__EmptyCell__) */
```

```
/**\n * \file EmptyCell.cpp\n * \brief Notre classe EmptyCell\n * \author CHARDAN Anaël\n * \author DAMEY Jérémy\n * \date 09/03/2014\n */\n\n#include "EmptyCell.h"\n\n/*=====*\n Les Constructeurs\n =====*/\n\nEmptyCell::EmptyCell() : CellBase() { }\n\nEmptyCell::EmptyCell( int x, int y ) : CellBase(x,y) { }\n\nEmptyCell::EmptyCell(const EmptyCell &b) {\n    CellBase::toClone(b);\n}\n\n/*=====*\n Le Destructeur\n =====*/\n\nEmptyCell::~EmptyCell() {} \n\n/*=====*\n Les opérateurs\n =====*/\n\nEmptyCell&\nEmptyCell::operator=(const EmptyCell &b) {\n    if ( this != &b ) {\n        CellBase::toClone(b);\n    }\n    return *this;\n}
```

```
#ifndef __PuruPuruDigger_GoldCell__
#define __PuruPuruDigger_GoldCell__


/***
 * \file GoldCell.h
 * \brief Notre classe GoldCell
 * \author CHARDAN Anaël
 * \author DAMEY Jérémy
 * \date 09/03/2014
 */

#include <iostream>
#include "ValueCell.h"

/*! \class GoldCell
 * \brief Classe modélisant ce qu'est une GoldCell
 */

class GoldCell : public ValueCell {

private :

    int my_bonus; /*!< Notre bonus */

    /*!
     * \brief Action toClone
     *
     * Action toClone pour copier une goldCell
     */
    virtual void toClone( const GoldCell& g);

public :

    /*!
     * \brief Constructeur
     *
     * Constructeur de la classe GoldCell
     */
    GoldCell();

    /*!
     * \brief Constructeur paramétré
     *
     * Constructeur paramétré de la classe GoldCell
     */
    GoldCell( int x, int y );

    /*!
     * \brief Constructeur par copie
     *
     * \param[in] GoldCell g
     * \param[out] GoldCell g
     */
    GoldCell( const GoldCell &g);

    /*!
     * \brief Destructeur
     *
     * Destructeur de la classe fille GoldCell
     */
}
```

```
virtual ~GoldCell();  
  
/*!  
 * \brief Retourne les points que va ajouter la case dans les scores  
 *  
 * \return my_points, retourne la valeur de la case  
 */  
virtual int getPoints() const;  
  
/*!  
 * \brief Opérateur d'affectation pour recopier une GoldCell  
 *  
 * \param[in] GoldCell g : opérateur d'affectation pour recopier une  
 * GoldCell  
 */  
virtual GoldCell& operator=( const GoldCell &g );  
  
};  
  
#endif /* defined(__PuruPuruDigger_GoldCell__) */
```

```
/**  
 * \file GoldCell.cpp  
 * \brief Notre classe GoldCell  
 * \author CHARDAN Anaël  
 * \author DAMEY Jérémy  
 * \date 09/03/2014  
 */  
  
#include "GoldCell.h"  
  
/*======  
 Les Constructeurs  
 ======*/  
  
GoldCell::GoldCell() : ValueCell(), my_bonus(randomNumber(MINVALB, MAXVALB)) {}  
  
GoldCell::GoldCell( int x, int y ) : ValueCell( x, y ), my_bonus(randomNumber(MINVALB, MAXVALB)) {}  
  
GoldCell::GoldCell( const GoldCell &g ) {  
    toClone(g);  
}  
  
/*======  
 Le Destructeur  
 ======*/  
  
GoldCell::~GoldCell() {}  
  
/*======  
 Les méthodes  
 ======*/  
  
//Pour connaître les points que vont rapporté les bonus  
int  
GoldCell::getPoints() const {  
    return ( ValueCell::getPoints() + my_bonus );  
}  
  
/*======  
 Les opérateurs  
 ======*/  
  
GoldCell&  
GoldCell::operator=(const GoldCell &v) {  
    if ( this != &v ){  
        toClone(v);  
    }  
    return *this;  
}  
  
void  
GoldCell::toClone(const GoldCell &g) {  
    ValueCell::toClone(g);  
    my_bonus = g.my_bonus;  
}
```

```
#ifndef __PuruPuruDigger__ValueCell__
#define __PuruPuruDigger__ValueCell__

/***
 * \file ValueCell.h
 * \brief Notre classe ValueCell
 * \author CHARDAN Anaël
 * \author DAMEY Jérémy
 * \date 09/03/2014
 */

#include <iostream>
#include "../Constantes.h"
#include "CellBase.h"
#include <cstdlib>

/*! \class ValueCell
 * \brief Classe modélisant ce qu'est une ValueCell
 */
class ValueCell : public CellBase {

protected :
    int my_value; /*!< Nos points */
    virtual void toClone( const ValueCell & v);

public :

    /**
     * \brief Constructeur
     *
     * Constructeur de la classe ValueCell
     */
    ValueCell();

    /**
     * \brief Constructeur paramétré
     *
     * Constructeur paramtré de la classe GoldCell
     */
    ValueCell( int x, int y);

    /**
     * \brief Constructeur par copie
     *
     * \param[in] ValueCell v
     * \param[out] ValueCell v
     */
    ValueCell( const ValueCell & v );

    /**
     * \brief Destructeur
     *
     * Destructeur de la classe fille ValueCell
     */
    virtual ~ValueCell();

    /**
     * \brief Retourne les points que va ajouter la case dans les scores
     *
     * \return my_value, retourne la valeur de la case
     */
}
```

```
 */
virtual int getValue() const;

/*
 * \brief Retourne les points que va ajouter la case dans les scores
 *
 * \return my_points, retourne la valeur de la case
 */
virtual int getPoints() const;

/*
 * \brief Opérateur d'affectation pour recopier une ValueCell
 *
 * \param[in] ValueCell v : opérateur d'affectation pour recopier une
 * GoldCell
 */
virtual ValueCell& operator=(const ValueCell &v);

};

#endif /* defined(__PuruPuruDigger__ValueCell__) */
```

```
/**  
 * \file ValueCell.cpp  
 * \brief Notre classe ValueCell  
 * \author CHARDAN Anaël  
 * \author DAMEY Jérémy  
 * \date 09/03/2014  
 */  
  
#include "ValueCell.h"  
#include <iostream>  
#include <string>  
  
/*======  
 Les Constructeurs  
 ======*/  
  
ValueCell::ValueCell() : CellBase(), my_value(randomNumber(MINVAL, MAXVAL)) {}  
  
ValueCell::ValueCell( int x, int y ) : CellBase(x,y), my_value(randomNumber(MINVAL, MAXVAL)) {}  
  
ValueCell::ValueCell( const ValueCell &v ) {  
    toClone(v);  
}  
  
/*======  
 Le Destructeur  
 ======*/  
  
ValueCell::~ValueCell() {}  
  
/*======  
 Les méthodes  
 ======*/  
  
int  
ValueCell::getValue() const {  
    return my_value;  
}  
  
int  
ValueCell::getPoints() const {  
    return my_value * 10;  
}  
  
/*======  
 Les opérateurs  
 ======*/  
  
ValueCell&  
ValueCell::operator=(const ValueCell &v) {  
    if ( this != &v ){  
        toClone(v);  
    }  
    return *this;  
}  
  
void
```

```
ValueCell::toClone(const ValueCell &v) {
    CellBase::toClone(v);
    my_value = v.my_value;
}
```

```
#ifndef purpurudigger_Constantes_h
#define purpurudigger_Constantes_h

/***
 * \file Constantes.h
 * \brief Les constantes
 * \author CHARDAN Anaël
 * \author DAMEY Jérémy
 * \date 09/03/2014
 */

//Relatif aux objets de notre grille
const int COLONNE = 18;
const int LIGNE = 18;
const int MINVAL = 1;
const int MAXVAL = 6;
const int MINOBJ = 8;
const int MAXOBJ = 10;
const int MINVALB = 10;
const int MAXVALB = 100;

//Relatif aux couleurs consoles
const int RED = 31;
const int GREEN = 32;
const int YELLOW = 33;
const int BLUE = 34;
const int PINK = 35;
const int CYAN = 36;
const int WHITE = 37;

//Relatif à tout ce qui est affichage
const int WINDOWWITDH = 1128;
const int WINDOWHEIGHT = 828;
const int BPP = 32;

//Pour la grille
const int MARGINLEFT = 400 ;
const int MARGINTOP = 100;
const int PADDINGRIGHT = 3;
const int PADDINGBOTTOM = 3;

const int CASEWITDH = 35;
const int CASEHEIGHT = 35;

//Pour nos feuilles de sprite ANANAS
const int SPRITECASEBEGIN = 6;
const int SPRITECASEHEIGHT = 56 ;
const int DIGGERSX = 0 ; //Début de la position en X, S pour Start
const int DIGGEREX = 50 ; //Fin de la position en X, E pour End
const int GOLDSX = 56 ;
const int GOLDEX = 106 ;
const int EMPTYSX = 112;
const int EMPTYEX = 162 ;
const int BOMBSX = 168 ;
const int BOMBEX = 218 ;
const int VALUESX = 224 ;
const int VALUEEX = 274 ;
```

```
//Pour la gestion des langues
const int TEXTPLACEX = 150;

const int SPRITELANGUEBEGIN = 12;
const int SPRITELANGUEHEIGHT = 122 ;

const int LANGUEWIDTH = 50;
const int LANGUEHEIGHT = 50;

const int CHOICELANGUEHIGH = 150;

const int MYLANGUEX = 400;
const int MYLANGUEY = CHOICELANGUEHIGH + 120;

const int ENGLISHX = 400;
const int ENGLISHSX = 7;
const int ENGLISHEX = 114;

const int FRENCHX = ENGLISHX + 100;
const int FRENCHSX = 123;
const int FRENCHEX = 230;

const int ITALIANOX = FRENCHX + 100;
const int ITALIANOSX = 240;
const int ITALIANOEX = 346;

const int SPANISHX = ITALIANOX + 100;
const int SPANISHSX = 360;
const int SPANISHEX = 463;

const int DEUTSCHX = SPANISHX + 100;
const int DEUTSCHSX = 472;
const int DEUTSCHEX = 578;

//Pour le choix des sprites
const int CHOICESPRITEY = MYLANGUEY + 120;

const int CHOICEANANASX = 400;
const int CHOICETEACHERX = 700;

const int SPRITECHOICEBEGIN = 0;
const int SPRITECHOICEHEIGHT = 150;
const int SPRITECHOICEWIDTH = 150;
const int SPRITEANANASSX = 0;
const int SPRITEANANASEX = 285;
const int SPRITETEACHERSX = 344;
const int SPRITETEACHEREX = 600;

//Pour les boutons
const int BUTTONWIDTH = 220;
const int BUTTONHEIGHT = 60;

const int BUTTONCASEBEGIN = 3;
const int BUTTONCASEHEIGHT = 68;

const int BUTTONGROWNSX = 2;
const int BUTTONGROWMEX = 143;
```

```
const int BUTTONHOVESX = 157;
const int BUTTONHOVEEX = 298;

//Pour les icônes de sons sur les sprites
const int ICONWIDTH = 50;
const int ICONHEIGHT = 50;

const int ICONSPRITEBEGIN = 0;
const int ICONSPRITEHEIGHT = 143;

const int MUSICONSX = 0;
const int MUSICONEX = 130;
const int MUSICOFFSX = 130;
const int MUSICOFFEX = 270;

const int SOUNDSX = 270;
const int SOUNDOFFEX = 370 ;
const int SOUNDONEX = 431;

//Pour le placement à l'écran des icônes
const int ICONY = 10;
const int MUSICX = WINDOWWITDH - ( 2 * ICONWIDTH ) - 100;
const int SOUNDX = MUSICX + ICONWIDTH + 20;

//L'emplacement des boutons à l'écran pour gérer les événements
const int PLAYX = 150;
const int PLAYY = 300;

const int OPTIONX = PLAYX + 200;
const int OPTIONY = PLAYY + 120;

const int BESTX = OPTIONX + 200;
const int BESTY = OPTIONY + 120;

const int QUITX = BESTX + 200;
const int QUITY = BESTY + 120;

const int QUITONX = 100;
const int QUITONY = 600;

//Relatif à notre fichier de meilleurs scores
#ifndef __linux__
const std::string FILEBESTSCORE = "Ressources/bestScores.txt";
#else
const std::string FILEBESTSCORE = "bestScores.txt";
#endif

/*
 * \enum Language
 * \brief Voici l'énumération des différents langues possibles
 */
enum Language{ francais, english, deutsch, espanol, italiano };

/*!
```

```
* \enum Message
* \brief Voici l'énumération des différents messages possibles
*
*/
enum Message{
    choice, move, nwest, north, neast, west, east, swest, south, seast, stop,
    score, level, global, current, goal, step, life, position,
    winlevel, looselevel, loosegame, name, ltime, timeup, by, play, best,
        setting, language, actual,
    theme, cheater
};

enum Movement {
    Nwest, NEast, North, South, SWest, SEast, West, East
};

#endif
```

```
#ifndef __purpurudigger__Game__
#define __purpurudigger__Game__

/***
 * \file GameModel.h
 * \brief Ce que représente une partie
 * \author CHARDAN Anaël
 * \author DAMEY Jérémy
 * \date 09/03/2014
 */

#include <iostream>
#include <vector>
#include "Level.h"
#include "Score.h"

/*! \class GameModel
 * \brief Classe modélisant une partie
 */

class GameModel {
private :
    Level* my_level; /*!< Nos Levels ( en vérité un mais infini ) */
    Score* my_score; /*!< Les scores de notre partie */
    Movement my_movement;

public :
    /*!
     * \brief Constructeur
     *
     * Constructeur de la classe GameModel
     */
    GameModel();

    /*!
     * \brief Destructeur
     *
     * Destructeur de la classe GameModel
     */
    ~GameModel();

    /*!
     * \brief Retourner notre score ( affichage )
     *
     * \return un pointeur constant sur le score
     */
    Score* const getScore();

    /*!
     * \brief Retourner notre Level ( affichage )
     *
     * \return un pointeur constant sur le level
     */
    Level* const getLevel();

    /*!
     * \brief Ordonner un mouvement à notre grille
     *
     * \param[in] le mouvement
     */
}
```

```
/*
void orderMovement( int depl );

/*
 * \brief Ordonner un mouvement à notre grille en fonction de la
 * position de la souris
 *
 * \param[in] int xclick
 * \param[in] int yclick
 */
void orderMovement( int xclick , int yclick );

/*
 * \brief Retourne le mouvement
 *
 */
Movement getMovement() const;

/*
 * \brief Savoir si la partie est terminée
 *
 * \return true si la partie est finie
 */
bool gameOver() const ;

void reset() ;
};

#endif /* defined(__purpurudigger__Game__) */
```

```
#ifndef __PuruPuruDigger__GameView__
#define __PuruPuruDigger__GameView__
/***
 * \file GameView.h
 * \brief Affichage de notre partie en mode terminal
 * \author CHARDAN Anaël
 * \author DAMEY Jérémy
 * \date 09/03/2014
 */

#include <iostream>

#include "GameModel.h"
#include "LanguageMessage.h"
#include <map>
#include <string>

/*! \class GameView
 * \brief Classe modélisant ce qu'est une vue
 */

class GameView {
private :
    Language my_language; /*!< La langue de notre partie */
    LanguageMessage my_messages; /*!< La bibliothèque de message de notre
                                   partie */
    std::map<std::string , std::string> my_typeToString;

    GoldCell* ptr_goldCell; /*!< Un pointeur de goldCell, pour que ce
                           soit plus pratique */
    ValueCell* ptr_valueCell; /*!< Un pointeur de valueCell, pour que ce
                           soit plus pratique */

    GameModel * my_model; /*!< La modèle de notre vue */
    /**
     * \brief Affichage Menu principal
     */
    void showPresentation() const;

    /**
     * \brief Affichage choix des langues
     */
    void showLanguage() const ;

    /**
     * \brief Affichage des scores
     */
    void showScore() ;

    /**
     * \brief Affichage de la grille
     */
    void showGrid() ;

    /**
     * \brief Affichage des instructions de déplacement
     */
    void showInstruction() ;

    /**

```

```
* \brief Affichage des meilleurs scores
*/
void showBestScore() const;

/*!
 * \brief Entrée d'un nouveau score
 *
 * \param[in] nom le nom du joueur
 */
void enterScore(std::string nom) const ;

public:

GameView();

/*!
 * \brief Injection du modèle à la vue
 *
 * \param[in] model : le modèle à interpréter */
void setModel(GameModel * model);

/*!
 * \brief La boucle de jeu
 */
void treatGame();

};

#endif /* defined(__PuruPuruDigger__GameView__) */
```

```
/**\n * \file GameView.cpp\n * \brief Méthode liée à l'affichage de notre partie en mode terminal\n * \author CHARDAN Anaël\n * \author DAMEY Jérémy\n * \date 09/03/2014\n */\n\n#include "GameView.h"\n#include "Constantes.h"\n#include <fstream>\n#include "IntDecFunctor.h"\n#include <string>\n#include <sstream>\n#include <cstdlib>\n#include "Utils.h"\n#include "Cell/Digger.h"\n#include "Cell/Bomb.h"\n#include "Cell/EmptyCell.h"\n#include "Cell/ValueCell.h"\n#include "Cell/GoldCell.h"\nusing namespace std;\n\nGameView::GameView() {\n    //On set notre map pour l'affichage\n    my_typeToString["Digger"] = colorMessage( "DD" , WHITE );\n    my_typeToString["Bomb"] = colorMessage( "BB" , RED );\n    my_typeToString["EmptyCell"] = colorMessage( " " , WHITE );\n    my_typeToString["GoldCell"] = colorMessage( "*" , PINK );\n    my_typeToString["ValueCell"] = colorMessage( " " , CYAN );\n}\n\nvoid\nGameView::setModel(GameModel *model) {\n    my_model = model;\n}\n\nvoid\nGameView::showPresentation() const {\n    cout << colorMessage( "+=====\n", CYAN ) << endl ;\n    cout << colorMessage( "|      |", BLUE ) << endl ;\n    cout << colorMessage( "|      |", CYAN ) << endl ;\n    cout << colorMessage( "|      |", BLUE ) << endl ;\n    cout << colorMessage( "+=====|\n", CYAN ) << endl ;\n    cout << colorMessage( "|      |", BLUE ) << endl ;\n    cout << colorMessage( "|      |", CYAN ) << endl ;\n    cout << colorMessage( "|      |", BLUE ) << endl ;\n    cout << colorMessage( "|      |", CYAN ) << endl ;\n    cout << colorMessage( "|      |", BLUE ) << endl ;\n    cout << colorMessage( "|      |", CYAN ) << endl ;\n    cout << colorMessage( "|      |", BLUE ) << endl ;\n    cout << colorMessage( "|      |", CYAN ) << endl ;\n    cout << colorMessage( "|      |", BLUE ) << endl ;\n    cout << colorMessage( "|=====|", CYAN ) << endl ;\n}
```

```
+=====+", CYAN ) << endl << endl << endl;

cout << colorMessage( "                                PURU          DIGGER
", RED ) << endl << endl << endl;

cout << colorMessage( "           BY ANAEL CHARDAN   &   JEREMY
DAMEY           ", GREEN ) << endl << endl << endl;

cout << colorMessage( " 1 : START ", YELLOW ) << endl;
cout << colorMessage( " 2 : BEST SCORE ", PINK ) << endl;
cout << colorMessage( " 3 : QUIT  ", WHITE ) << endl << endl;

cout << colorMessage( " CHOICE : ", YELLOW );
}

void
GameView::showLanguage() const {
    cout << endl << colorMessage( " 1 : Francais  2 : English  3 : Deutsch
        4 : Espanol  5 : Italiano ", WHITE ) << endl << endl;
    cout << colorMessage( " CHOICE : ", YELLOW );
}

void
GameView::showGrid() {
    ///Affichage de la première ligne
    for ( int z = 0; z < (COLONNE * 5 + 3); z++ ) {
        if ( z%5 == 1 )
            cout << colorMessage( "+", YELLOW );
        else
            cout << colorMessage( "-", YELLOW );
    }

    cout << endl;

    ///Affichage de la grille selon le getType bien utile
    for ( int i = 0; i < LIGNE; i++ ) {
        cout << colorMessage( " | ", YELLOW );
        for ( int j = 0; j < COLONNE; j++ ) {

            cout << my_typeToString.at(my_model->getLevel()->getGrid()[i][j]->
                getType());

            if ( (my_model->getLevel()->getGrid()[i][j])->getType() ==
                "GoldCell" ) {
                ///On copie le contenu du pointeur donné, qui ne renverra
                // normalement pas nul
                ptr_goldCell = dynamic_cast<GoldCell*>(my_model->getLevel()->
                    getGrid()[i][j]);
                cout << colorMessage( intToString( ptr_goldCell->getValue() ).
                    c_str(), PINK);
            }
            else if ( (my_model->getLevel()->getGrid()[i][j])->getType() ==
                "ValueCell" ) {
                ///On copie le contenu du pointeur donné, qui ne renverra
                // normalement pas nul
                ptr_valueCell = dynamic_cast<ValueCell*>(my_model->getLevel()->
                    getGrid()[i][j]);
            }
        }
    }
}
```

```
        cout << colorMessage( intToString( ptr_valueCell->getValue() )
                           .c_str(), CYAN );
    }

    cout << colorMessage( " | ", YELLOW );
}

cout << endl;

///Affichage de la ligne en dessous de chaque case
for ( int z = 0; z < (COLONNE * 5 + 3); z++ ) {
    if ( z%5 == 1 )
        cout << colorMessage( "+", YELLOW );
    else
        cout << colorMessage( "-", YELLOW );
}
cout << endl;
}
cout << endl;
}

void
GameView::showScore() {
    cout << my_messages[my_language][score] << " : " << endl << endl;
    cout << my_messages[my_language][level] << my_model->getScore()->
        getCurrentStep() << endl;
    cout << my_messages[my_language][global] << my_model->getScore()->
        getGlobale() << endl;
    cout << my_messages[my_language][current] << ( my_model->getScore() )->
        getCurrent() << endl;
    cout << my_messages[my_language][goal] << ( my_model->getLevel() )->
        getGoal() << endl;
    cout << my_messages[my_language][step] << ( my_model->getLevel() )->
        getCurrentMove() << endl;
    cout << my_messages[my_language][life] << " Digger : " << ( ( my_model->
        getLevel() )->getDigger() )->getLife() << endl;
    cout << my_messages[my_language][position] << " Digger : [ " << my_model-
        >getLevel() )->getDigger() )->getX() << " ] [ " << my_model->getLevel() )->
        getDigger() )->getY() << " ] " << endl << endl;

    cout << my_messages[my_language][ltime]; for ( int i = 0; i < my_model->
        getLevel() )->leftTime(); i++ ) { cout << colorMessage(":", CYAN ); }
    cout << " " << my_model->getLevel() )->leftTime() << endl << endl;
}

void
GameView::showInstruction( ) {
    cout << my_messages[my_language][move] << " : " << endl << endl;
    cout << " 7 : " << my_messages[my_language][nwest] << " 8 : " <<
        my_messages[my_language][north] << " 9 : " << my_messages[my_language]
        [neast] << endl;
    cout << " 4 : " << my_messages[my_language][west] << " "
        << " 6 : " << my_messages[my_language][east] << endl
        ;
    cout << " 1 : " << my_messages[my_language][swest] << " 2 : " <<
        my_messages[my_language][south] << " 3 : " << my_messages[my_language]
        [seast] << endl << endl;

    cout << " 5 : " << my_messages[my_language][stop] << endl << endl;
```

```
        cout << my_messages[my_language][choice];
    }

void
GameView::showBestScore() const {
    ifstream scoreLect(FILEBESTSCORE.c_str(), ios::in );
    if ( scoreLect ) {
        string line;

        cout << endl << endl << " Best Scores " << endl << endl;

        while ( getline(scoreLect, line) ) {
            cout << line << endl;
        }

        scoreLect.close();

        cout << endl;
    } else {
        cerr << " Error when program is openning text file " << endl;
    }
}

void
GameView::enterScore( string nom ) const{
    ifstream scoreLect(FILEBESTSCORE.c_str(), ios::in );
    if ( scoreLect ) {
        string line;
        int scoreligne;
        string nomligne;
        map< int, string, DecFunctor> Scores;
        int scorePlayer = ( my_model->getScore() )->getGlobale() ;

        while ( !scoreLect.eof() ) {
            //On lit le score et on le stocke dans une map
            scoreLect >> scoreligne >> nomligne;
            Scores[scoreligne] = nomligne.c_str();
        }

        //On ajoute notre joueur dans la map
        Scores[scorePlayer] = nom;

        scoreLect.close();

        ofstream scoreEcr(FILEBESTSCORE.c_str(), ios::out | ios::trunc );

        map< int, string>::iterator i;
        if ( Scores.size() < 5 ) {
            i = Scores.end();
        } else {
            i = Scores.begin();
            for ( int cpt = 0 ; cpt < 5; cpt ++ ) ++i;
        }

        for ( map< int, string >::const_iterator it = Scores.begin() ; it!=i ;
              ++it) {
            scoreEcr << it->first;
            scoreEcr << " ";
            scoreEcr << it->second;
        }
    }
}
```

```
        scoreEcr << endl;
    }

    scoreEcr.close();
} else {
    cerr << " Error when program is opening text file " << endl;
}
}

void
GameView::treatGame() {
    bool isRunning = true;
    bool isPlaying = false;
    int menuChoice;
    int languechoice;
    int movechoice;
    string nom;
    while ( isRunning ) {
        showPresentation();
        cin >> menuChoice;
        while ( !(menuChoice>= 1 && menuChoice<=3 ) ) {
            showPresentation();
            cin >> menuChoice;
        }
        switch ( menuChoice ) {
            case 1 : isPlaying = true;
            break;
            case 2 : showBestScore();
            break;
            case 3 : isRunning = false;
        }
    }

    if ( isPlaying ) {
        showLanguage();
        cin >> languechoice;
        while ( !(languechoice >= 1 && languechoice<=5 ) ) {
            showLanguage();
            cin>>languechoice;
        }
        cout << endl;
        switch ( languechoice ) {
            case 1 : my_language = francais;
            break;
            case 2 : my_language = english;
            break;
            case 3 : my_language = deutsch;
            break;
            case 4 : my_language = espanol;
            break;
            case 5 : my_language = italiano;
            break;
        }
        my_model->reset();

        while (isPlaying) {
            showGrid();
            showScore();
            showInstruction();
        }
    }
}
```

```
cin >> movechoice;

//Pour ne pas rentrer des choses fausses
while ( !(movechoice >= 0 && movechoice <= 9) ) {
    showInstruction();
    cin >>movechoice;
}
//On vérifie le temps, s'il est écoulé
if ( my_model->getLevel()->timeIsUp() ) {
    cout << endl << endl;
    cout << my_messages[my_language][timeup] << endl;
    //on fait perdre un niveau
    my_model->getLevel()->lostLevel();
    //On vérifie si ce n'est pas gameOver
    if ( my_model->gameOver() ) {
        cout << my_messages[my_language][loosegame] << endl;
       .isPlaying = false;
    }
}

} else {
    if ( movechoice == 5 ) {
        cout << my_messages[my_language][by] << endl;
       .isPlaying = false;
    } else {
        //On fait le mouvement
        my_model->orderMovement(movechoice);
        //Si le digger gagne un level
        if ( my_model->getLevel()->win() ) {
            my_model->getLevel()->resetWin();
            cout << my_messages[my_language][winlevel] << endl
                ;
        }

        } else {
            //Si la partie est fini
            if ( my_model->gameOver() ) {
                cout << my_messages[my_language][loosegame] <<
                    endl;
               .isPlaying = false;
            } else {
                //Si le digger perd un level
                if ( my_model->getLevel()->lose() ) {
                    my_model->getLevel()->resetLose();
                    cout << my_messages[my_language]
                        [looselevel] << endl;
                }
            }
        }
    }
}

//La partie est finie
cout << my_messages[my_language][name];
cin >> nom;
enterScore(nom);
showBestScore();
}

}

cout << " GOOD BYE " << endl;
```

```
#ifndef __PuruPuruDigger__PuruContext__
#define __PuruPuruDigger__PuruContext__

/***
 * \file PuruContext.h
 * \brief Notre classe PuruContext
 * \author CHARDAN Anaël
 * \author DAMEY Jérémy
 * \date 09/03/2014
 */

#include <iostream>
#include "Constantes.h"

/*! \class PuruContext
 * \brief Classe modélisant les setters et les getters pour faire fonctionner
 * le jeu
 */

class PuruContext {
private :
    bool my_isInBreak; /*!< Savoir si le jeu est en pause */
    bool my_isInPresentation; /*!< Savoir si on est dans le menu
                               principal */
    bool my_isChoosingOption; /*!< Savoir si on est dans le menu option
                               */
    bool my_isEnterABestScore; /*!< Savoir si on rentre un score */
    bool my_isPlaying; /*!< Savoir si on est en train de jouer */
    bool my_isInAnimation; /*!< Savoir s'il y a l'animation */
    bool my_isOver; /*!< Savoir si la partie est finie */
    bool my_isTimeOver; /*!< Savoir si le temps est écoulé */
    bool my_isViewingBestScore; /*!< savoir si on est dans le tableau des
                               scores */
    bool my_isEnableSound; /*!< Savoir si le son est activé ou pas */
    bool my_isEnableMusic; /*!< Savoir si la musique est activé ou pas */
    Language my_language; /*!< Langage du jeu */

public :
    /*!
     * \brief Constructeur
     *
     * Constructeur de la classe PuruContext
     */
    PuruContext();

    /*!
     * \brief Setteur du jeu en pause
     *
     * Affecte si oui ou non le jeu est en pausse
     *
     * \param[in] bool set
     */
    void setInBreak( bool set );

    /*!
     * \brief Setteur du menu principale
     *
     * Affecte si oui ou non on est dans le menu
     */
}
```

```
/*
 * \param[in] bool set
 */
void setInPresentation( bool set );

/*!
 * \brief Setteur du menu option
 *
 * Affecte si oui ou non on est dans le menu option
 *
 * \param[in] bool set
 */
void setChoosingOption( bool set );

/*!
 * \brief Setteur de la saisie des meilleurs score
 *
 * Affecte si oui ou non on rentre un score
 *
 * \param[in] bool set
 */
void setEnterABestScore( bool set );

/*!
 * \brief Setteur de play
 *
 * Affecte si oui ou non on est en train de jouer
 *
 * \param[in] bool set
 */
void setPlaying( bool set );

/*!
 * \brief Setteur de animation
 *
 * Affecte si oui ou non il y a des animations
 *
 * \param[in] bool set
 */
void setAnimation( bool set );

/*!
 * \brief Setteur de over
 *
 * Affecte si oui ou non la partie est fini
 *
 * \param[in] bool set
 */
void setOver( bool set );

/*!
 * \brief Setteur du temps
 *
 * Affecte si oui ou non le temps est écoulé
 *
 * \param[in] bool set
 */
void setTimeOver( bool set );

/*!
```

```
* \brief Setteur des meilleurs score
*
* Affecte si oui ou non on est train de saisir un score
*
* \param[in] bool set
*/
void setViewingBestScore( bool set );

/*!
* \brief Setteur du son
*
* Affecte si oui ou non il y a du son
*
* \param[in] bool set
*/
void setSound( bool set );

/*!
* \brief Setteur de musique
*
* Affecte si oui ou non il y a de la musique
*
* \param[in] bool set
*/
void setMusic( bool set );

/*!
* \brief Setteur de langage
*
* Affecte le langage utilisé
*
* \param[in] Language language
*/
void setLanguage( Language language );

/*!
* \brief Guetteur du menu option
*
* retourne l'option choisi
*/
bool isChoosingOption() const;

/*!
* \brief Guetteur du jeu en pause
*
* retourne vrai ou faux en fonction si le jeu est en pause ou non
*/
bool isInBreak() const;

/*!
* \brief Guetteur du menu principal
*
* retourne vrai ou faux en fonction si on est dans le menu principal
*/
bool isInPresentation() const;

/*!
* \brief Guetteur des meilleurs score
*
```

```
* retourne vrai ou faux en fonction si on est en train de saisir un
   score
*/
bool isEnterABestScore() const;

/*
 * \brief Guetteur de play
 *
 * retourne vrai ou faux en fonction si on est en train de jouer ou
   pas
*/
bool isPlaying() const;

/*
 * \brief Guetteur des animations
 *
 * retourne vrai ou faux en fonction si il y a des animations
 */
bool isInAnimation() const;

/*
 * \brief Guetteur de over
 *
 * retourne vrai ou faux en fonction si la prati est fini
 */
bool isOver() const;

/*
 * \brief Guetteur du temps
 *
 * retourne vrai ou faux en fonction si le temps est éoulé
 */
bool isTimeOver() const;

/*
 * \brief Guetteur du menu principales meilleurs score
 *
 * retourne vrai ou faux en fonction si on est dans le menu des
   meilleurs score
*/
bool isViewingBestScore() const;

/*
 * \brief Guetteur des sons
 *
 * retourne vrai ou faux en fonction si il y a un son
 */
bool isEnableSound() const;

/*
 * \brief Guetteur des musiques
 *
 * retourne vrai ou faux en fonction si il y a de la musique
 */
bool isEnableMusic() const;

/*
 * \brief Guetteur du langage
 *
 * retourne le lnagage
```

```
 */  
 Language getLanguage() const;  
};  
#endif /* defined(__PuruPuruDigger__PuruContext__) */
```

```
/**\n * \file PuruContext.h\n * \brief Notre classe PuruContext\n * \author CHARDAN Anaël\n * \author DAMEY Jérémy\n * \date 09/03/2014\n */\n\n#include "PuruContext.h"\n\nPuruContext::PuruContext() {\n    my_isInPresentation = true;\n    my_isInBreak = false;\n    my_isChoosingOption = false;\n    my_isPlaying = false;\n    my_isInAnimation = false;\n    my_isOver = false;\n    my_isTimeOver = false;\n    my_isEnterABestScore = false;\n    my_isViewingBestScore = false;\n    my_isEnableSound = true;\n    my_isEnableMusic = true;\n    my_language = français;\n}\n\n/* ======\nLes Setteurs\n=====*/\n\nvoid PuruContext::setInBreak( bool set ) { my_isInBreak = set; }\n\nvoid PuruContext::setInPresentation( bool set ) { my_isInPresentation = set; }\n\nvoid PuruContext::setChoosingOption( bool set ) { my_isChoosingOption = set; }\n\nvoid PuruContext::setEnterABestScore( bool set ) { my_isEnterABestScore = set; }\n\nvoid PuruContext::setPlaying( bool set ) { my_isPlaying = set; }\n\nvoid PuruContext::setAnimation( bool set ) { my_isInAnimation = set; }\n\nvoid PuruContext::setOver( bool set ) { my_isOver = set; }\n\nvoid PuruContext::setTimeOver( bool set ) { my_isTimeOver = set; }\n\nvoid PuruContext::setViewingBestScore( bool set ) { my_isViewingBestScore = set; }\n\nvoid PuruContext::setSound( bool set ) { my_isEnableSound = set; }\n\nvoid PuruContext::setMusic( bool set ) { my_isEnableMusic = set; }\n\nvoid PuruContext::setLanguage( Language language ) { my_language = language; }\n\n/* ======\nLes Guetteurs\n=====*/
```

```
bool PuruContext::isChoosingOption() const { return my_isChoosingOption; }

bool PuruContext::isInBreak() const { return my_isInBreak; }

bool PuruContext::isInPresentation() const { return my_isInPresentation; }

bool PuruContext::isEnterABestScore() const { return my_isEnterABestScore; }

bool PuruContext::isPlaying() const { return my_isPlaying; }

bool PuruContext::isInAnimation() const { return my_isInAnimation; }

bool PuruContext::isOver() const { return my_isOver; }

bool PuruContext::isTimeOver() const { return my_isTimeOver; }

bool PuruContext::isViewingBestScore() const { return my_isViewingBestScore; }

bool PuruContext::isEnableSound() const { return my_isEnableSound; }

bool PuruContext::isEnableMusic() const { return my_isEnableMusic; }

Language PuruContext::getLanguage() const { return my_language; }
```

```
#ifndef __purpurudigger_Level__
#define __purpurudigger_Level__

/***
 * \file Level.h
 * \brief Les méthodes liées à notre classe level
 * \author CHARDAN Anaël
 * \author DAMEY Jérémy
 * \date 09/03/2014
 */

#include <iostream>
#include "Score.h"
#include "Cell/Digger.h"
#include "Cell/CellBase.h"
#include "Cell/ValueCell.h"
#include "Cell/GoldCell.h"
#include "Cell/Bomb.h"
#include "Constantes.h"
#include <ctime>

typedef std::vector<std::vector<CellBase*>> Grid; /*!< Typedef pour faciliter l'écriture */

/*! \class Level
 * \brief Classe modélisant un level
 */
class Level {

private :

    //La composition de notre Level
    Score* my_score;           /*!< Score de notre Level */
    Digger* my_digger;         /*!< Digger de notre Level */
    Grid my_grid;              /*!< Grille de notre Level */

    //Les objectifs de notre Level
    int my_goal;               /*!< Objectif en Point de notre Level */
    int my_currentMove;        /*!< Mouvement en cours */
    int my_bonus;               /*!< Bonus en Point de notre Level */

    //Les attributs temporels
    time_t my_depart;          /*!< Temps de commencement du Level */
    float timeGoal;             /*!< Objectif en temps du Level */

    //La connaissance de ce qu'il se passe dans notre Level.
    bool my_win;                /*!< Savoir si je viens de gagner un Level */
    bool my_lose;                /*!< Savoir si je viens de perdre un Level */

    ValueCell* ptr_valueCell;
    GoldCell* ptr_goldCell;

    /**
     * \brief Déplacement du Digger dans la grille
     *
     * \param[in] DeltaX La direction vertical de déplacement du digger
     * \param[in] DeltaY La direction horizontal de déplacement du digger
     */
}
```

```
void move( int deltaX, int deltaY );\n\n/*!\n * \brief Mélange d'un tableau en 2D\n */\nvoid shuffle();\n\n/*!\n * \brief Génération de la première grille\n */\nvoid initGrid();\n\n\npublic:\n/*!\n * \brief Nouvelle grille après avoir gagné\n */\nvoid winLevel();\n\n/*!\n * \brief Méthode pour regénérer une grille après avoir perdu ou gagné\n * sans perdre les attributs de notre Digger\n */\nvoid reset();\n/*!\n * \brief Constructeur\n *\n * Constructeur de la classe Level\n *\n * \param *score : score de notre partie (injection de dépendance )\n */\nLevel(Score* score);\n\n/*!\n * \brief Destructeur\n *\n * Destructeur de la classe Level\n */\n~Level();\n\n/*!\n * \brief Perdre le level\n *\n * Faire perdre une vie et régénération d'un niveau\n */\nvoid lostLevel();\n\n/*!\n * \brief remettre win a false\n *\n * remet l'attribut win false\n */\nvoid resetWin();\n\n/*!\n * \brief remettre lose a false\n *
```

```
* remet l'attribut lose false
*/
void resetLose();

/*! \brief Connaitre l'objectif en point du Level
 *
 * \return my_goal
 */
int getGoal() const;

/*! \brief Connaitre nos déplacements en cours
 *
 * \return my_currentMove
 */
int getCurrentMove() const;

/*! \brief Connaitre si le temps est écoulé
 *
 * \return true si le temps est écoulé
 */
bool timeIsUp() const;

/*! \brief Connaitre le temps restant
 *
 * \return le temps restant
 */
float leftTime() const;

/*! \brief Connaitre le temps actuelle à jour
 */
void resetTime();

/*! \brief Retourner notre digger pour avoir des informations sur lui
 *
 * \return un pointeur sur notre digger
 */
Digger* const getDigger() const ;

/*! \brief Connaitre si la case est franchissable ( de type numérique
 * et à côté du Digger
 *
 * param[in] click_x : la position verticale de notre click
 * param[in] click_y : la position horizontale de notre click
 *
 * \return true si la case est franchissable
 */
bool isCellClickable( int click_x, int click_y ) const;

/*! \brief Renvoyer notre grille ( affichage )
 *
 * \return my_grid
```

```
/*
const Grid& getGrid() const;

/*
 * \brief Connaitre si notre Digger est définitivement mort
 *
 * \return true si il est mort
 */
bool isDead() const;

/*
 * \brief Connaitre si l'on vient juste de gagner un niveau
 *       ( affichage )
 *
 * \return true si l'on vient de gagner un niveau
 */
bool win() const;

/*
 * \brief Connaitre si l'on vient juste de perdre un niveau
 *       ( affichage )
 *
 * \return true si l'on vient de perdre un niveau
 */
bool lose() const;

//Tous nos sucres de langages, il appeleront la fonction move avec
//notre digger et les bons deltas et le nombre de coup

/*
 * \brief Sucre pour se déplacer à gauche ( deltaX 0, deltaY -1, voir
 *        la methode move()
 * \see deplacement
 */
void moveWest();

/*
 * \brief Sucre pour se déplacer à droite ( deltaX 0, deltaY 1, voir
 *        la methode move()
 * \see deplacement
 */
void moveEast();

/*
 * \brief Sucre pour se déplacer en haut ( deltaX -1, deltaY 0, voir
 *        la methode move()
 * \see deplacement
 */
void moveNorth();

/*
 * \brief Sucre pour se déplacer en bas ( deltaX 1, deltaY 0, voir la
 *        methode move()
 * \see deplacement
 */
void moveSouth();

/*
 * \brief Sucre pour se déplacer en haut à droite ( deltaX -1, deltaY
```

```
1, voir la methode move()
 * \see deplacement
 */
void moveNorthEast();

/*
 * \brief Sucre pour se déplacer en haut à gauche ( deltaX -1, deltaY
 * -1, voir la methode move()
 * \see deplacement
 */
void moveNorthWest();

/*
 * \brief Sucre pour se déplacer en bas à gauche ( deltaX 1, deltaY
 * -1, voir la methode move()
 * \see deplacement
 */
void moveSouthWest();

/*
 * \brief Sucre pour se déplacer en bas à droite ( deltaX 1, deltaY 1,
 * voir la methode move()
 * \see deplacement
 */
void moveSouthEast();

};

#endif /* defined(__purpurudigger_Level__) */
```

```
/**\n * \file Level.cpp\n * \brief Les méthodes liées à notre classe level\n * \author CHARDAN Anaël\n * \author DAMEY Jérémy\n * \date 09/03/2014\n */\n\n#include "Level.h"\n#include "Cell/Bomb.h"\n#include "Cell/GoldCell.h"\n#include "Cell/EmptyCell.h"\n#include <vector>\n#include <algorithm>\n\nusing namespace std;\n/*=====*\nLe constructeur\n=====*/\n\nLevel::Level(Score* score): my_score(score) {\n    //On met nos mouvements courants à 0\n    my_currentMove = 0;\n\n    //Comme c'est le premier niveau, l'objectif est 10\n    my_goal = 10;\n\n    //Comme c'est le premier niveau, le bonus généré par le level est 500\n    my_bonus = 500;\n\n    //On donne l'instant présent à notre départ\n    time(&my_depart);\n    timeGoal = 60;\n\n    //On alloue le digger\n    my_digger = new Digger(0,0);\n\n    //On bloque la taille de notre vecteur\n    my_grid.resize( LIGNE );\n\n    for( int i = 0; i < LIGNE; i++ ) {\n        my_grid[i].resize( COLONNE );\n    }\n\n    my_win = false;\n    my_lose = false;\n\n    //On fait pointé notre case sur notre digger;\n    my_grid[0][0] = my_digger;\n\n    //On appelle la fonction n'initialisation\n    reset();\n}\n\n/*=====*\nLe destructeur\n=====*/\n\nLevel::~Level() {
```

```
for ( int i = 0; i < LIGNE; i++ ) {
    for ( int j = 0; j < COLONNE; j++ ) {
        delete my_grid[i][j];
    }
}

/*=====
Les méthodes privés
=====*/

void
Level::move( int DeltaX, int DeltaY ) {
    int nbStep = -1;
    int pointInGame = -1;

    if ( isCellClickable( ( my_digger->getX() + DeltaX ), ( my_digger->getY()
+ DeltaY ) ) ) {

        ///On transtype la case concerner, ce transtypage n'a pas besoin
        d'être vérifier
        ptr_valueCell = dynamic_cast<ValueCell*>(my_grid[ (my_digger->getX() +
DeltaX) ][ (my_digger->getY() + DeltaY) ]);

        ///On prend les nbStep à faire
        nbStep = ptr_valueCell->getValue();

        ///On regarde si c'est une goldCell
        if ( my_grid[ (my_digger->getX() + DeltaX) ][ (my_digger->getY() +
DeltaY) ]->getType() == "GoldCell" ) {
            ptr_goldCell = dynamic_cast<GoldCell*>(my_grid[ (my_digger->getX() +
DeltaX) ][ (my_digger->getY() + DeltaY) ]);
            pointInGame = ptr_goldCell->getPoints();
        } else {
            pointInGame = ptr_valueCell->getPoints();
        }
    }

    ///On met notre compteur à 0
    int cpt = 0;

    ///Tant que l'on à pas fait le bon nombre de coup et que la case d'a côté
    est bien une gold ou une value
    while ( cpt < nbStep && ( isCellClickable( ( my_digger->getX() + DeltaX ),
( my_digger->getY() + DeltaY ) ) ) ) {

        ///Si l'on rencontre un trésor pendant un déplacement
        if ( my_grid[ ( my_digger->getX() + DeltaX ) ][ ( my_digger->getY() +
DeltaY ) ]->getType() == "GoldCell" ) {
            ///On prend les points du bonus
            ptr_goldCell = dynamic_cast<GoldCell*>(my_grid[ ( my_digger->getX()
) + DeltaX ) ][ ( my_digger->getY() + DeltaY ) ]);

            ///On fait un aléatoire pour voir si l'on va gagner du temps, de
            la vie, ou des points )
            int aleat = randomNumber(0,2);

            ///On fait les vérifications et on fait les modifictions en
            conséquences
            switch ( aleat ) {
```

```
    ///On gagne des points
    case 0 :
        my_score->addPoints( ptr_goldCell->getPoints() );
        break;

        ///On remet le temps à 0;
    case 1 :
        resetTime();
        break;

        ///On gagne une vie
    case 2 :
        my_digger->addLife();
        break;
    }

    my_score->addPoints( ptr_goldCell->getPoints() );

}

///On delete la case suivante
delete my_grid[ ( my_digger->getX() + DeltaX ) ][ ( my_digger->getY()
+ DeltaY ) ];

///On y place notre digger
my_grid[ ( my_digger->getX() + DeltaX ) ][ ( my_digger->getY() +
DeltaY ) ] = my_digger;

///On remplace notre ancienne case du digger par une case Vide avec un
autoSet
my_grid[ my_digger->getX() ][ my_digger->getY() ] = new EmptyCell(
my_digger->getX(), my_digger->getY() );

///On set les case de notre digger
my_digger->setX( my_digger->getX() + DeltaX );
my_digger->setY( my_digger->getY() + DeltaY );

///On passe au coup suivant
cpt++;
}

//Si le déplacement s'est mal passé ( donc cpt a bougé et est différent de
0 )
//Les autres renvoient -1
if ( cpt != 0 && cpt < nbStep ) {
    lostLevel();
} else if ( nbStep != -1 && cpt!=0 ){
    my_currentMove += nbStep;
    my_score->addPoints(pointInGame);
    //Si on a atteint l'objectif
    if ( my_currentMove >= my_goal ) {
        winLevel();
    }
}

void
Level::shuffle() {

    vector<CellBase*> tmp;
```

```
tmp.resize( LIGNE * COLONNE );
int z = 0;

//2D to 1D
while ( z < ( LIGNE * COLONNE ) ) {
    for ( int i = 0 ; i < LIGNE ; i++ ){
        for ( int j = 0; j < COLONNE ; j++ ){
            tmp[z] = my_grid[i][j];
            z++;
        }
    }
}

random_shuffle( tmp.begin(), tmp.end() );

//1D to 2D
z=0;
while ( z < LIGNE * COLONNE ) {
    //Parcours en hauteur
    for ( int i = 0 ; i < LIGNE ; i++ ) {
        //Parcours en inteur
        for ( int j = 0; j < COLONNE ; j++ ) {
            my_grid[i][j] = tmp[z];
            //On peut maintenant set chaque case avec les bon x et les bon
            //y dont le digger
            my_grid[i][j]->setX(i);
            my_grid[i][j]->setY(j);
            z++;
        }
    }
}
}

void
Level::initGrid() {
    //Calcul du nombre de bombe
    int nbrB = randomNumber(MINOBJ, MAXOBJ );

    //Remplissage du tableau avec des bombe
    for ( int i = 1 ; i <= nbrB; i++ ) {
        my_grid[0][i] = new Bomb();
    }

    //On place les trésors en fonction du nombre de bomb
    for ( int i = nbrB +1 ; i < COLONNE ; i++ ) {
        my_grid[0][i] = new GoldCell();
    }

    //On rempli tout le reste avec des numéros
    for ( int i = 1; i < LIGNE; i++ ) {
        for ( int j = 0; j < COLONNE; j++ ) {
            my_grid[i][j] = new ValueCell();
        }
    }

    //On mélange tout cela
    shuffle();
}

//Gagner un level
```

```
void
Level::winLevel() {
    my_win = true;
    //On additionne le bonus au score courant
    my_score->addPoints( my_bonus );
    //On dit que l'on fait un nouveau niveau
    my_score->addSuccess();
    //On augmente le bonus du level
    my_bonus += 500;
    //On augmente la difficulté
    my_goal +=10;
    //On reset le level
    reset();
}

void
Level::reset() {
    //On remet nos mouvements à 0
    my_currentMove = 0;

    Digger* digger_temp = new Digger( *my_digger ) ;

    //On delete tout
    for ( int i = 0; i < LIGNE; i++ ) {
        for ( int j = 0; j < COLONNE; j++ ) {
            delete my_grid[i][j];
        }
    }

    my_digger = new Digger( *digger_temp );

    my_grid[0][0] = my_digger;
    delete digger_temp;
    resetTime();
    initGrid();

}

/*=====
 Les méthodes publics
 =====*/
void
Level::resetWin() {
    my_win = false;
}

void Level::resetLose() {
    my_lose = false;
}

void
Level::lostLevel() {
    my_lose = true;
    //On fait perdre une vie au digger
    my_digger->lostLife();
    //On reset le score actuel
    my_score->resetScore();
    //On reset le level
    reset();
}
```

```
int
Level::getGoal() const {
    return my_goal;
}

int
Level::getCurrentMove() const {
    return my_currentMove;
}

bool
Level::timeIsUp() const {
    time_t dateActuelle;
    time(&dateActuelle);
    if ( difftime( dateActuelle, my_depart ) > timeGoal )
        return true;
    else
        return false;
}

float
Level::leftTime() const {
    time_t dateActuelle;
    time(&dateActuelle);
    return ( timeGoal - difftime(dateActuelle, my_depart) );
}

void
Level::resetTime() {
    time(&my_depart);
}

Digger* const
Level::getDigger() const {
    return my_digger;
}

bool
Level::isCellClickable( int click_x, int click_y ) const {

    /// Il faut vérifier si l'on ne sort pas du tableau
    if ( click_x >= 0 && click_x < LIGNE && click_y >= 0 && click_y < COLONNE
        ) {

        ///Il faut d'abord vérifier que la case est juste à côté de notre
        ///digger
        if ( ( ( click_x <= my_digger->getX() - 1 ) || ( click_x <= my_digger-
            >getX() + 1 ) ) && ( ( click_y <= my_digger->getY() - 1 ) || (
            click_y <= my_digger->getY() + 1 ) ) {

            ///On vérifie son type
            if ( my_grid[click_x][click_y]->getType() == "GoldCell" || my_grid
                [click_x][click_y]->getType() == "ValueCell" ) {
                return true;
            }
        }
    }
    return false;
}
```

```
const Grid&
Level::getGrid() const {
    return my_grid;
}

bool
Level::isDead() const {
    if ( my_digger->getLife() < 0 )
        return true;
    else
        return false;
}

bool
Level::win() const {
    return my_win;
}

bool
Level::lose() const {
    return my_lose;
}

/*=====
Les sucres
=====*/
void
Level::moveWest() {
    move( 0, -1 );
}

void
Level::moveEast() {
    move( 0, 1 );
}

void
Level::moveNorth() {
    move( -1, 0 );
}

void
Level::moveSouth() {
    move( 1, 0 );
}

void
Level::moveNorthEast() {
    move( -1, 1 );
}

void
Level::moveNorthWest() {
    move( -1, -1 );
}

void
```

```
Level::moveSouthWest() {
    move( 1, -1 );
}

void
Level::moveSouthEast() {
    move( 1, 1 );
}
```

```
#ifndef __purpurudigger__Score__
#define __purpurudigger__Score__

/***
 * \file Score.h
 * \brief Notre classe Score
 * \author CHARDAN Anaël
 * \author DAMEY Jérémie
 * \date 09/03/2014
 */

#include <iostream>
#include <vector>

/*! \class Score
 * \brief Classe modélisant un score
 */

class Score {
private :
    std::vector<int> my_success; /*!< Notre tableau de succès */
public :
    /*!
     * \brief Constructeur
     *
     * Constructeur de la classe Score
     */
    Score();

    /*!
     * \brief Connaitre le score courant
     *
     * \return la valeur de la dernière case de notre tableau de score
     */
    int getCurrent() const;

    /*!
     * \brief Connaitre le niveau
     *
     * \return la taille de notre tableau de score
     */
    int getCurrentStep() const;

    /*!
     * \brief Connaitre le score totale
     *
     * \return la somme du contenu de notre tableau de score
     */
    int getGlobale() const ;

    /*!
     * \brief Ajoute une case à notre tableau
     */
    void addSuccess();

    /*!
     * \brief Remet à 0 la valeur de la dernière case de notre tableau
     */
    void resetScore();
}
```

```
/*!
 * \brief Ajoute des points à notre case courante
 *
 * \param[in] la valeur de ce que l'on doit ajouter à notre score
 * courant
 */
void addPoints(const int &i);

#endif
```

```
/**\n * \file Score.cpp\n * \brief Méthodes liées à notre classe score\n * \author CHARDAN Anaël\n * \author DAMEY Jérémy\n * \date 09/03/2014\n */\n\n#include "Score.h"\nusing namespace std;\n\nScore::Score() {\n    my_success.resize(1);\n}\n\nint\nScore::getCurrent() const {\n    return my_success[ my_success.size() - 1 ];\n}\n\nint\nScore::getCurrentStep() const {\n    return ( static_cast<int>( my_success.size() ) );\n}\n\nint\nScore::getGlobale() const {\n    int sum = 0;\n    //On prends pas la case en cours.\n    for ( int i = 0; i < ( static_cast<int>( my_success.size() ) - 1 ) ; ++i )\n        sum += my_success[i];\n    return sum;\n}\n\nvoid\nScore::addSuccess() {\n    my_success.resize( my_success.size() + 1 );\n}\n\nvoid\nScore::addPoints( const int &i ) {\n    my_success[ my_success.size() - 1 ] += i;\n}\n\nvoid\nScore::resetScore() {\n    my_success[ my_success.size() - 1 ] = 0;\n}
```

```
/**\n * \file main.cpp\n * \brief Notre main terminal\n * \author CHARDAN Anaël\n * \author DAMEY Jérémy\n * \date 09/03/2014\n */\n\n#include <iostream>\n#include "GameModel.h"\n#include "GameView.h"\n#include <cstdlib>\n\nusing namespace std;\n\n\nint main(int argc, const char * argv[])\n{\n    std::srand ( unsigned ( std::time(0) ) );\n\n    GameModel* model = new GameModel;\n    GameView * view = new GameView;\n\n    //Injection de dépendance\n    view->setModel(model);\n\n    view->treatGame();\n\n    delete view;\n    delete model;\n\n    return 0;\n}
```

```
#ifndef __PuruPuruDigger__IntDecFunctor__
#define __PuruPuruDigger__IntDecFunctor__

/***
 * \file IntDecFunctor.h
 * \brief Notre foncteur pour trier une map à l'envers
 * \author CHARDAN Anaël
 * \author DAMEY Jérémy
 * \date 09/03/2014
 */
#include <iostream>

/*! \class DecFunctor
 * \brief Foncteur pour trier une map à l'envers
 */
class DecFunctor {
public :
    template<typename T, typename S> /*!< Notre template */

    /**
     * \brief Constructeur
     *
     * \param[in] T n1
     * \param[in] S n2
     * \return booléen pour le sens
     */
    bool operator()( T n1, S n2 ) { return n2 < n1; }
};

#endif /* defined(__PuruPuruDigger__IntDecFunctor__) */
```

```
#ifndef __PuruPuruDigger__LanguageMessage__
#define __PuruPuruDigger__LanguageMessage__

/***
 * \file LanguageMessage.h
 * \brief Notre classe LanguageMessage
 * \author CHARDAN Anaël
 * \author DAMEY Jérémy
 * \date 09/03/2014
 */

#include <iostream>
#include <map>
#include <string>
#include "Constantes.h"

/*! \class LanguageMessage
 * \brief Classe pour enregistrer notre multilangue */
class LanguageMessage {
public :
    std::map< Language, std::map< Message, std::string> > my_languages; /
        *!< Notre map */

    /*!
     * \brief Constructeur
     *
     * Constructeur de la classe LanguageMessage
     */
    LanguageMessage();

    /*!
     * \brief Surcharge de l'opérateur crochet
     *
     */
    std::map<Message, std::string>& operator[]( Language language );
};

#endif /* defined(__PuruPuruDigger__LanguageMessage__) */
```

```
/**  
 * \file LanguageMessage.cpp  
 * \brief Notre classe LanguageMessage  
 * \author CHARDAN Anaël  
 * \author DAMEY Jérémy  
 * \date 09/03/2014  
 */  
  
#include "LanguageMessage.h"  
  
using namespace std;  
  
LanguageMessage::LanguageMessage() {  
    //Les messages français  
    my_languages[francais][choice] = " Choix ";  
    my_languages[francais][move] = " Déplacement ";  
    my_languages[francais][north] = " Nord ";  
    my_languages[francais][south] = " Sud ";  
    my_languages[francais][west] = " Ouest ";  
    my_languages[francais][east] = " Est ";  
    my_languages[francais][nwest] = " Nord Ouest ";  
    my_languages[francais][neast] = " Nord Est ";  
    my_languages[francais][swest] = " Sud Ouest ";  
    my_languages[francais][seast] = " Sud Est ";  
    my_languages[francais][stop] = " Quitter ";  
    my_languages[francais][score] = " Score ";  
    my_languages[francais][level] = " Niveau ";  
    my_languages[francais][global] = " Score Global ";  
    my_languages[francais][current] = " Score Courant ";  
    my_languages[francais][goal] = " Objectif ";  
    my_languages[francais][step] = " En cours ";  
    my_languages[francais][life] = " Vie ";  
    my_languages[francais][position] = " Position ";  
    my_languages[francais][winlevel] = " Vous gagnez un niveau " ;  
    my_languages[francais][looselevel] = " Vous perdez une vie, recommencez un  
        niveau ";  
    my_languages[francais][loosegame] = " Vous avez perdu la partie ";  
    my_languages[francais][name] = " Entrez votre nom ";  
    my_languages[francais][by] = " Vous arretez la partie " ;  
    my_languages[francais][ltime] = " Temps restant ";  
    my_languages[francais][timeup] = " Vous avez mis trop de temps, vous  
        perdez le niveau ";  
    my_languages[francais][best] = " Scores ";  
    my_languages[francais][play] = " Jouer ";  
    my_languages[francais][setting] = " Options ";  
    my_languages[francais][language] = " Langues ";  
    my_languages[francais][actual] = " En cours ";  
    my_languages[francais][theme] = " Thèmes ";  
    my_languages[francais][cheater] = " Tricheur ";  
  
    //les messages anglais  
    my_languages[english][choice] = " Choice ";  
    my_languages[english][move] = " Move ";  
    my_languages[english][north] = " North ";  
    my_languages[english][south] = " South ";  
    my_languages[english][west] = " West ";  
    my_languages[english][east] = " East ";  
    my_languages[english][nwest] = " North West ";
```

```
my_languages[english][neast] = " North East ";
my_languages[english][swest] = " South West ";
my_languages[english][seast] = " South East ";
my_languages[english][stop] = " Stop ";
my_languages[english][score] = " Score ";
my_languages[english][level] = " Level ";
my_languages[english][global] = " Global Score ";
my_languages[english][current] = " Current Score ";
my_languages[english][goal] = " Goal ";
my_languages[english][step] = " Step ";
my_languages[english][life] = " Life ";
my_languages[english][position] = " Position ";
my_languages[english][winlevel] = " You win a level ";
my_languages[english][looselevel] = " You lost a life, try again ";
my_languages[english][loosegame] = " You have lose a game ";
my_languages[english][name] = " Enter your name ";
my_languages[english][by] = " You stop the game ";
my_languages[english][ltime] = " Left Time ";
my_languages[english][timeup] = " You during too time, you left a level ";
my_languages[english][best] = " Score ";
my_languages[english][play] = " Play ";
my_languages[english][setting] = " Settings ";
my_languages[english][language] = " Languages ";
my_languages[english][actual] = " Ongoing ";
my_languages[english][theme] = " Theme ";
my_languages[english][cheater] = " Cheater ";
```

```
//les messages espagnol
my_languages[espanol][choice] = " Eleccion ";
my_languages[espanol][move] = " Desplazamiento ";
my_languages[espanol][north] = " Norte ";
my_languages[espanol][south] = " Sur ";
my_languages[espanol][west] = " Oeste ";
my_languages[espanol][east] = " Este ";
my_languages[espanol][nwest] = " Norte Oeste ";
my_languages[espanol][neast] = " Norte Este ";
my_languages[espanol][swest] = " Sur Oeste ";
my_languages[espanol][seast] = " Sur Este ";
my_languages[espanol][stop] = " Dejar ";
my_languages[espanol][score] = " Puntuacion ";
my_languages[espanol][level] = " Nivel ";
my_languages[espanol][global] = " Puntuacion Global ";
my_languages[espanol][current] = " Calification actual ";
my_languages[espanol][goal] = " Objetivo ";
my_languages[espanol][step] = " En marcha ";
my_languages[espanol][life] = " Vida ";
my_languages[espanol][position] = " Posicion ";
my_languages[espanol][winlevel] = " Que obtuvo un nivel ";
my_languages[espanol][looselevel] = " Usted gana un juego, iniciar un
    nivel ";
my_languages[espanol][loosegame] = " Usted perdio el juego ";
my_languages[espanol][name] = " Escriba su nombre ";
my_languages[espanol][by] = " Dejas de parte ";
my_languages[espanol][ltime] = " Tiempo restante ";
my_languages[espanol][timeup] = " Tomó demasiado tiempo, perder un nivel "
    ;
my_languages[espanol][best] = " Puntaje ";
my_languages[espanol][play] = " Jugar ";
```

```
my_languages[espanol][setting] = " Opcion ";
my_languages[espanol][language] = " Idiomas ";
my_languages[espanol][actual] = " En marcha ";
my_languages[espanol][themel] = " Tema ";
my_languages[espanol][cheater] = " Tramposo ";

//les messages italien
my_languages[italiano][choice] = " Scelta ";
my_languages[italiano][move] = " Spostamento ";
my_languages[italiano][north] = " Nord ";
my_languages[italiano][south] = " Sud ";
my_languages[italiano][west] = " Ovest ";
my_languages[italiano][east] = " Oriente ";
my_languages[italiano][nwest] = " Nord Ovest ";
my_languages[italiano][neast] = " Nord Oriente ";
my_languages[italiano][swest] = " Sud Ovest ";
my_languages[italiano][seast] = " Sud Oriente ";
my_languages[italiano][stop] = " Lasciare ";
my_languages[italiano][score] = " Punteggio ";
my_languages[italiano][level] = " Livello ";
my_languages[italiano][global] = " Punteggio totale ";
my_languages[italiano][current] = " Punteggio Courrant ";
my_languages[italiano][goal] = " Obiettivo ";
my_languages[italiano][step] = " In corso ";
my_languages[italiano][life] = " Vita ";
my_languages[italiano][position] = " Posizione ";
my_languages[italiano][winlevel] = " Di livello ";
my_languages[italiano][looselevel] = " Si perde una vita, avviare un
    livello ";
my_languages[italiano][loosegame] = " Hai perso la partita ";
my_languages[italiano][name] = " Inserisci il tuo nome ";
my_languages[italiano][by] = " Si smette di parte ";
my_languages[italiano][ltime] = " Tempo rimanente ";
my_languages[italiano][timeup] = " hai preso troppo a lungo, perdere un
    livello ";
my_languages[italiano][best] = " Punteggio ";
my_languages[italiano][play] = " Giocare ";
my_languages[italiano][setting] = " Opzione ";
my_languages[italiano][language] = " Lingue ";
my_languages[italiano][actual] = " In corso ";
my_languages[italiano][theme] = " Tema ";
my_languages[italiano][cheater] = " Baro ";

//les messages allemands
my_languages[deutsch][choice] = " Wahl ";
my_languages[deutsch][move] = " Verdrangung ";
my_languages[deutsch][north] = " Norden ";
my_languages[deutsch][south] = " Suden ";
my_languages[deutsch][west] = " Westen ";
my_languages[deutsch][east] = " Osten ";
my_languages[deutsch][nwest] = " Norden Westen ";
my_languages[deutsch][neast] = " Norden Osten ";
my_languages[deutsch][swest] = " Suden Westen ";
my_languages[deutsch][seast] = " Suden Osten ";
my_languages[deutsch][stop] = " Verlassen ";
my_languages[deutsch][score] = " Partitur ";
my_languages[deutsch][level] = " Ebene ";
my_languages[deutsch][global] = " Gesamtnote ";
```

```
my_languages[deutsch][current] = " Aktuelle punktzahl ";
my_languages[deutsch][goal] = " Ziel ";
my_languages[deutsch][step] = " Laufend ";
my_languages[deutsch][life] = " Leben ";
my_languages[deutsch][position] = " Position ";
my_languages[deutsch][winlevel] = " Sie hat eine ebene gewonnen ";
my_languages[deutsch][looselevel] = " Sie verlieren ein leben, starten sie
    ine ebene ";
my_languages[deutsch][loosegame] = " Sie verloren das spiel";
my_languages[deutsch][name] = " Geben sie ihren namen ";
my_languages[deutsch][by] = " Sie teil stoppen ";
my_languages[deutsch][ltime] = " Restzeit ";
my_languages[deutsch][timeup] = " Sie zu lange dauerte, verlieren eine
    Ebene ";
my_languages[deutsch][best] = " Ergebnis";
my_languages[deutsch][play] = " Spieler ";
my_languages[deutsch][setting] = " Optionen ";
my_languages[deutsch][language] = " Sprachen ";
my_languages[deutsch][actual] = " Laufend ";
my_languages[deutsch][theme] = " Thema ";
my_languages[deutsch][cheater] = " Betruger ";

}

std::map<Message, std::string>&
LanguageMessage::operator[](Language langue) {
    return my_languages[langue];
}
```

```
#ifndef __PuruPuruDigger__GameViewSFML__
#define __PuruPuruDigger__GameViewSFML__

/***
 * \file GameViewSFML.h
 * \brief Notre classe GameViewSFML
 * \author CHARDAN Anaël
 * \author DAMEY Jérémy
 * \date 09/03/2014
 */

#include <iostream>
#include <SFML/Window.hpp>

#include "GameModel.h"
#include "Graphics/ButtonGraphic.h"
#include "Observers/EventDispatcher.h"
#include "Graphics/GraphicMusic.h"
#include "Graphics/GraphicSound.h"
#include "PuruContext.h"
#include "Graphics/LanguageGraphic.h"
#include "Graphics/AnanasSprite.h"
#include "Graphics/TeacherSprite.h"
#include "Manager/SoundManager.h"
#include "Graphics/BackgroundGraphic.h"

/*! \class GameViewSFML
 * \brief Classe qui gère tout l'affichage du jeu
 */

class GameView {
private :
    sf::RenderWindow* my_window; /*!< La fenêtre de notre jeu */
    GameModel * my_model; /*!< La modèle de notre vue */
    EventDispatcher* my_eventDispatcher;

    ButtonGraphic* my_playButton; /*!< Le bouton play */
    ButtonGraphic* my_settingButton; /*!< Le bouton option */
    ButtonGraphic* my_bestButton; /*!< Le bouton meilleur score */
    ButtonGraphic* my_quitButton; /*!< Le bouton quitter */

    GraphicMusic *my_musicIcon; /*!< Le bouton pour la musique */
    GraphicSound *my_soundIcon; /*!< Le bouton pour le son */

    AnanasSprite *my_ananasSprite; /*!< Le mode ananas */
    TeacherSprite *my_teacherSprite; /*!< Le mode du professeur */

    BackgroundGraphic *my_background; /*!< Le fond de notre jeu */

    std::map< Language, LanguageGraphic*>* my_languageToSprite; /*!< Le
        choix des langages */

    PuruContext* my_context; /*!< Le context */
    SoundManager *my_soundManager; /*!< Le choix du son */

    /**
     * \brief Permet d'accéder au menu
     */
    void goToPresentation();
}
```

```
/*
 * \brief Permet d'accéder au menu option
 */
void goToSettings();

/*
 * \brief Permet de jouer
 */
void goToPlay();

/*
 * \brief Permet d'accéder au score
 */
void goToScore();

/*
 * \brief Permet de rentrer un score
 */
void goToEnterScore();

/*
 * \brief Initialise le jeu
 */
void initView();

/*
 * \brief Initialise le menu
 */
void initPresentation();

/*
 * \brief Initialise le menu option
 */
void initSettings();

/*
 * \brief Initialise le meilleur score
 */
void initBestScore();

/*
 * \brief Initialise le fait de saisir un score
 */
void initEnterScore();

/*
 * \brief Initialise le jeu
 */
void initPlay();

public :
    /*
     * \brief Constructeur
     *
     * Constructeur de la classe GameViewSFML
     */
    GameView();
```

```
/*!
 * \brief Destructeur
 *
 * Destructeur de la classe GameViewSFML
 */
~GameView();
/*! 
 * \brief Injection du modèle à la vue
 *
 * \param[in] model : le modèle à interpréter */
void setModel(GameModel * model);

/*! 
 * \brief La boucle de jeu
 */
void treatGame();

};

#endif /* defined(__PuruPuruDigger__GameViewSFML__) */
```

```
/**\n * \file GameViewSFML.cpp\n * \brief Notre classe GameViewSFML\n * \author CHARDAN Anaël\n * \author DAMEY Jérémy\n * \date 09/03/2014\n */\n\n#include "GameViewSFML.h"\n\n#include "Observers/InterfaceObserver.h"\n#include "Graphics/EnglishGraphic.h"\n#include "Graphics/FrenchGraphic.h"\n#include "Graphics/SpanishGraphic.h"\n#include "Graphics/DeutschGraphic.h"\n#include "Graphics/ItalianoGraphic.h"\n#include "Manager/SoundManager.h"\n\nusing namespace sf;\n\n//Constructeur\nGameView::GameView() {\n    //Le style de la fenêtres\n    my_window = new RenderWindow( VideoMode(WINDOWWIDTH, WINDOWHEIGHT, BPP),\n        "PuruPuruDigger", Style::Close);\n\n    //On bloque le rafraîchissement à 60 par seconde\n    my_window->SetFramerateLimit(60);\n\n    // buttons\n    my_playButton = new ButtonGraphic();\n    my_settingButton = new ButtonGraphic();\n    my_bestButton = new ButtonGraphic();\n    my_quitButton = new ButtonGraphic();\n\n    my_soundIcon = new GraphicSound();\n    my_musicIcon = new GraphicMusic();\n\n    my_ananasSprite = new AnanasSprite();\n    my_teacherSprite = new TeacherSprite();\n\n    my_background = new BackgroundGraphic();\n\n    my_languageToSprite = new std::map<Language, LanguageGraphic*>();\n\n    my_languageToSprite->operator[](english) = new EnglishGraphic();\n    my_languageToSprite->operator[](francais) = new FrenchGraphic();\n    my_languageToSprite->operator[](italiano) = new ItalianoGraphic();\n    my_languageToSprite->operator[](espanol) = new SpanishGraphic();\n    my_languageToSprite->operator[](deutsch) = new DeutschGraphic();\n\n    // dispatch d'event en tout genre\n    my_context = new PuruContext();\n    SoundManager::getInstance()->setContext( my_context );\n    my_eventDispatcher = new EventDispatcher( my_context );\n}\n\n//Destructeur
```

```
GameView::~GameView() {
    delete my_window;
    delete my_context;
    delete my_playButton;
    delete my_settingButton;
    delete my_bestButton;
    delete my_quitButton;
    delete my_soundIcon;
    delete my_ananasSprite;
    delete my_teacherSprite;
    delete my_background;
    delete my_eventDispatcher;

    for ( std::map<Language, LanguageGraphic*>::const_iterator it =
        my_languageToSprite->begin() ; it!=my_languageToSprite->end(); ++it) {
        delete (*my_languageToSprite)[ it->first ];
    }
}

//Injection de dépendance model
void GameView::setModel(GameModel *model) {
    my_model = model;
}

void GameView::goToPresentation() {
    my_context->setChoosingOption( false );
    my_context->setViewingBestScore( false );
    my_context->setInPresentation( true );
}

void GameView::goToSettings() {
    my_context->setInPresentation( false );
    my_context->setChoosingOption( true );
}

void GameView::goToPlay() {
    my_context->setInPresentation( false );
    my_context->setPlaying( true );

    if ( my_context->isEnableMusic() ) {
        SoundManager::getInstance()->playMusic();
    }
    my_model->reset();
}

void GameView::goToScore() {
    my_context->setInPresentation( false );
    my_context->setViewingBestScore( true );
}

void GameView::goToEnterScore() {
    my_context->setPlaying( false );
    my_context->setEnterABestScore( true );
    my_context->setAnimation( false );
}
```

```
void GameView::initPresentation() {
    my_eventDispatcher->addObserver( my_playButton );
    my_eventDispatcher->addObserver( my_bestButton );
    my_eventDispatcher->addObserver( my_settingButton );
    my_eventDispatcher->addObserver( my_quitButton );

    for ( std::map<Language, LanguageGraphic*>::const_iterator it =
        my_languageToSprite->begin() ; it!=my_languageToSprite->end(); ++it ) {
        my_eventDispatcher->removeObserver( (*my_languageToSprite)[ it->first ] );
    }

    my_eventDispatcher->removeObserver(my_ananasSprite);
    my_eventDispatcher->removeObserver(my_teacherSprite);
}

void GameView::initSettings() {
    for ( std::map<Language, LanguageGraphic*>::const_iterator it =
        my_languageToSprite->begin() ; it!=my_languageToSprite->end(); ++it ) {
        my_eventDispatcher->addObserver( (*my_languageToSprite)[ it->first ] );
    }

    my_eventDispatcher->addObserver(my_ananasSprite);
    my_eventDispatcher->addObserver(my_teacherSprite);

    my_eventDispatcher->removeObserver( my_playButton );
    my_eventDispatcher->removeObserver( my_bestButton );
    my_eventDispatcher->removeObserver( my_settingButton );

}

void GameView::initBestScore() {
    my_eventDispatcher->addObserver( my_quitButton );
    my_eventDispatcher->removeObserver( my_playButton );
    my_eventDispatcher->removeObserver( my_bestButton );
    my_eventDispatcher->removeObserver( my_settingButton );
}

void GameView::initEnterScore() {
    my_eventDispatcher->removeObserver( my_quitButton );
}

void GameView::initPlay() {
    my_eventDispatcher->removeObserver( my_playButton );
    my_eventDispatcher->removeObserver( my_bestButton );
    my_eventDispatcher->removeObserver( my_settingButton );
};

void GameView::initView() {
    if ( my_context->isInPresentation() ) {
        initPresentation();
    } else if ( my_context->isChoosingOption() ) {
        initSettings();
    } else if ( my_context->isViewingBestScore() ) {
        initBestScore();
    } else if ( my_context->isPlaying() ) {
```

```
    initPlay();
} else if ( my_context->isEnterABestScore() ) {
    initEnterScore();
}
}

//Boucle d'événement
void GameView::treatGame( ) {

Interface0bserver* interface0bserver = new Interface0bserver( my_window,
    my_model, my_playButton, my_settingButton, my_bestButton, my_quitButton
    , my_musicIcon, my_soundIcon, my_languageToSprite, my_ananasSprite,
    my_teacherSprite, my_background );

my_eventDispatcher->add0bserver( interface0bserver );
my_eventDispatcher->add0bserver( my_soundIcon );
my_eventDispatcher->add0bserver( my_musicIcon );
my_eventDispatcher->add0bserver( my_background );

// On abonne quand meme pour le change theme
// cela sera desabommer dans le init
for ( std::map<Language, LanguageGraphic*>::const_iterator it =
    my_languageToSprite->begin() ; it!=my_languageToSprite->end(); ++it) {
    my_eventDispatcher->add0bserver( (*my_languageToSprite)[ it->first ] )
        ;
}
my_eventDispatcher->add0bserver(my_ananasSprite);
my_eventDispatcher->add0bserver(my_teacherSprite);
my_eventDispatcher->add0bserver( my_playButton );
my_eventDispatcher->add0bserver( my_bestButton );
my_eventDispatcher->add0bserver( my_settingButton );
my_eventDispatcher->add0bserver( my_quitButton );

// theme par defaut
my_eventDispatcher->changeTheme("ananas");

initPresentation();

while ( my_window->IsOpened( ) ) {
    Event event;
    while ( my_window->GetEvent( event ) ) {
        if ( event.Type == Event::Closed ) {
            my_window->Close();
        } else {
            initView();

            switch (event.Type) {
                case Event::MouseButtonPressed:
                    if ( my_context->isInPresentation() ) {
                        if ( my_playButton->isInZone(event.MouseButton.X,
                            event.MouseButton.Y) ) {
                            goToPlay();
                        } else if ( my_quitButton->isInZone(event.
                            MouseButton.X, event.MouseButton.Y) ) {
                            my_window->Close();
                        } else if ( my_settingButton->isInZone(event.
                            MouseButton.X, event.MouseButton.Y) ) {
                            goToSettings();
                        } else if ( my_bestButton->isInZone(event.
                            MouseButton.X, event.MouseButton.Y) ) {
                            goToBest();
                        }
                    }
            }
        }
    }
}
```

```
        MouseButton.X, event.MouseButton.Y ) ) {  
            goToScore();  
        }  
    } else if ( my_context->isChoosingOption() ) {  
        for ( std::map<Language, LanguageGraphic*>::  
            const_iterator it = my_languageToSprite->begin()  
            ; it!=my_languageToSprite->end(); ++it ) {  
            if ( (*my_languageToSprite)[ it->first ]->  
                isInZone ( event.MouseButton.X, event.  
                MouseButton.Y ) ) {  
                my_context->setLanguage( it->first );  
                break;  
            }  
        }  
  
        if ( my_ananasSprite->isInZone( event.MouseButton.  
            X, event.MouseButton.Y ) ) {  
            my_eventDispatcher->changeTheme("ananas");  
        } else if ( my_teacherSprite->isInZone( event.  
            MouseButton.X, event.MouseButton.Y ) ) {  
            my_eventDispatcher->changeTheme("teacher");  
        } else if ( my_quitButton->isInZone(event.  
            MouseButton.X, event.MouseButton.Y) ) {  
            goToPresentation();  
        }  
    } else if ( my_context->isViewingBestScore() ) {  
        if ( my_quitButton->isInZone(event.MouseButton.X,  
            event.MouseButton.Y ) ) {  
            goToPresentation();  
        }  
    } else if ( my_context->isPlaying() ) {  
        int valueY = convertYPixel( event.MouseButton.Y );  
        int valueX = convertXPixel( event.MouseButton.X );  
  
        if ( ( valueY != -1 ) && ( valueX != -1 ) && ( !  
            my_context->isInAnimation() ) ) {  
            SoundManager::getInstance()->clickCell();  
            my_model->orderMovement( valueY, valueX );  
            my_context->setAnimation( true );  
        }  
        if ( my_quitButton->isInZone(event.MouseButton.X,  
            event.MouseButton.Y) ) {  
            goToEnterScore();  
        }  
    }  
    break;  
default:  
    break;  
  
}  
my_eventDispatcher->notify( event );  
}  
}  
  
my_eventDispatcher->preDisplay();  
my_window->Display();  
my_eventDispatcher->postDisplay();  
}
```

```
    delete interface0bserver;  
}
```

```
/**\n * \file mainSFML.cpp\n * \brief Notre main SFML\n * \author CHARDAN Anaël\n * \author DAMEY Jérémy\n * \date 09/03/2014\n */\n\n#include <iostream>\n#include "GameModel.h"\n#include "GameViewSFML.h"\n#include <cstdlib>\n\nusing namespace std;\n\n\nint main(int argc, const char * argv[])\n{\n    std::srand ( unsigned ( std::time(0) ) );\n\n    GameModel* model = new GameModel;\n    GameView* view = new GameView;\n\n    //Injection de dépendance\n    view->setModel(model);\n\n    view->treatGame();\n\n    delete view;\n    delete model;\n\n    return 0;\n}
```

```
#ifndef __PuruPuruDigger_Utils__
#define __PuruPuruDigger_Utils__


/***
 * \file Utils.h
 * \author CHARDAN Anaël
 * \author DAMEY Jérémy
 * \date 09/03/2014
 */

#include <string>

/*! \brief Header pour toute les fonctions qui se repete souvent
 */

/*! \brief Change les texte de couleur
 *
 * \param[in] const char* out
 * \param[in] int color
 */
std::string colorMessage( const char* out , int color );

/*! \brief Convertit un int en string
 *
 * \param[in] int i
 */
std::string intToString( int i );

/*! \brief Tire un nombre au hasard entre min et max
 *
 * \param[in] int min
 * \param[in] int max
 */
int randomNumber( int min, int max );

/*! \brief Convertit l'indice i en pixel
 *
 * \param[in] int i
 */
int convertIndiceXToPixel( int i );

/*! \brief Convertit l'indice j en pixel
 *
 * \param[in] int j
 */
int convertIndiceYToPixel( int j );

/*! \brief Convertit un pixel en indice
 *
 * \param[in] int xpixel
 */
int convertXPixel( int xpixel );
```

```
/*!
 * \brief Convertit un pixel en indice
 *
 * \param[in] int ypixel
 */
int convertYPixel( int ypixel );

#endif /* defined(__PuruPuruDigger_Utils__) */
```

```
/**\n * \file Utils.h\n * \brief Notre utilitaire\n * \author CHARDAN Anaël\n * \author DAMEY Jérémy\n * \date 09/03/2014\n */\n\n#include "Utils.h"\n#include <iostream>\n#include <sstream>\n#include <cstdlib>\n#include <string>\n#include "Constantes.h"\n#include <cmath>\n\nusing namespace std;\n\nstring intToString( int i ) {\n    ostringstream oss;\n    oss << i;\n    return oss.str();\n}\n\nstd::string\ncolorMessage( const char* out , int color ) {\n    ostringstream o;\n#ifndef __linux__\n    o << "\x1B[" << color << ";1m" << out << "\x1B[m";\n    return o.str();\n#else\n    o << out;\n    return o.str();\n#endif\n}\n\nint convertIndiceXToPixel( int i ) {\n    return CASEWITDH * i + MARGINLEFT + PADDINGRIGHT * i ;\n}\n\nint convertIndiceYToPixel ( int j ) {\n    return CASEHEIGHT * j + MARGINTOP + PADDINGBOTTOM * j ;\n}\n\nint convertXPixel( int xpixel ) {\n    if ( xpixel >= MARGINLEFT && xpixel <= ( MARGINLEFT + ( COLONNE *\n        CASEWITDH ) + ( ( COLONNE - 1 ) * PADDINGRIGHT ) ) ) {\n        return ceil( ( xpixel - MARGINLEFT ) / ( CASEWITDH + PADDINGRIGHT ) )\n    ;\n    }\n    return -1;\n}\n\nint convertYPixel( int ypixel ) {\n    if ( ypixel >= MARGINTOP && ypixel <= ( MARGINTOP + ( LIGNE * CASEHEIGHT )\n        + ( ( LIGNE - 1 ) * PADDINGBOTTOM ) ) ) {\n        return ceil( ( ypixel - MARGINTOP ) / ( CASEHEIGHT + PADDINGBOTTOM )\n    );\n}
```

```
    return -1;
}

int randomNumber( int min , int max ) {
    return min + ( rand() % ( max + 1 - min ) );
}
```

```
#define BOOST_TEST_DYN_LINK
#define BOOST_TEST_MODULE PuruPuruTest

/***
 * \file test.cpp
 * \brief Nos tests unitaires
 * \author CHARDAN Anaël
 * \author DAMEY Jérémie
 * \date 09/03/2014
 */

#include<boost/test/unit_test.hpp>
#include "../Utils.h"
#include "../Constantes.h"
#include "../Cell/Digger.h"
#include "../Cell/GoldCell.h"
#include "../Cell/EmptyCell.h"
#include "../Cell/Bomb.h"
#include "../Score.h"
#include "../Level.h"
#include "../GameModel.h"

using namespace std;

/*=====
    UTILS
=====*/

//Test de la méthode colorMessage
BOOST_AUTO_TEST_CASE ( testcolorMessage ) {
    std::string plop = colorMessage( "Plop", RED );
#ifdef __linux__
    BOOST_CHECK( plop == "\E[31;1mPlop\E[m" );
#else
    BOOST_CHECK( plop == "Plop" );
#endif
}

//Test de la méthode randomNumber
BOOST_AUTO_TEST_CASE ( testRandomNumber ) {
    int number = randomNumber( 5, 6 );
    BOOST_CHECK( number == 5 || number == 6 );
}

//Test de la méthode intToString
BOOST_AUTO_TEST_CASE ( testintToString ) {
    string cinq = intToString(5);
    BOOST_CHECK( cinq == "5" );
}

//Test de la méthode convertIndiceXToPixel
BOOST_AUTO_TEST_CASE ( testconvertIndiceXToPixel ) {
    int convertIndice5ToPixel = convertIndiceXToPixel(5);
    BOOST_CHECK( convertIndice5ToPixel == 590 );
}

//Test de la méthode convertIndiceYToPixel
BOOST_AUTO_TEST_CASE ( testconvertIndiceYToPixel ) {
    int convertIndice5ToPixel = convertIndiceYToPixel(5);
    BOOST_CHECK( convertIndice5ToPixel == 290 );
```

```
}

//Test de la méthode convertXPixel
BOOST_AUTO_TEST_CASE ( testconvertXPixel ) {
    int convert5Pixel = convertXPixel(590);
    BOOST_CHECK( convert5Pixel == 5 );
}

//Test de la méthode convertYPixel
BOOST_AUTO_TEST_CASE ( testconvertYPixel ) {
    int convert5Pixel = convertYPixel(290);
    BOOST_CHECK( convert5Pixel == 5 );
}

//Test de la méthode convertXPixel
BOOST_AUTO_TEST_CASE ( testconvertXPixeltooShort ) {
    int convert5Pixel = convertXPixel(5);
    BOOST_CHECK( convert5Pixel == -1 );
}

//Test de la méthode convertYPixel
BOOST_AUTO_TEST_CASE ( testconvertYPixeltooShort ) {
    int convert5Pixel = convertYPixel(5);
    BOOST_CHECK( convert5Pixel == -1 );
}

/*=====
BOMB
=====
*/
//Test constructeur par défaut
BOOST_AUTO_TEST_CASE ( testDefaultConstructorBomb ) {
    Bomb b;
    BOOST_CHECK( b.getType() == "Bomb" && b.getX() == 0 && b.getY() == 0 );
}

//Test constructeur paramétré
BOOST_AUTO_TEST_CASE ( testparamConstructorBomb ) {
    Bomb b(5,5);
    BOOST_CHECK( b.getType() == "Bomb" && b.getX() == 5 && b.getY() == 5 );
}

//Test du constructeur par copie
BOOST_AUTO_TEST_CASE ( testcopyConstructorBomb ) {
    Bomb b(5,5);
    Bomb c(b);
    BOOST_CHECK( b.getX() == c.getX() && b.getY() == c.getY() && b.getType() ==
                 c.getType() );
}

//Test de l'opérateur d'affectation
BOOST_AUTO_TEST_CASE ( testoperatoregalBomb ) {
    Bomb b(5,5);
    Bomb c = b;
    BOOST_CHECK( b.getX() == c.getX() && b.getY() == c.getY() && b.getType() ==
                 c.getType() );
}

//Test du get et du set X
BOOST_AUTO_TEST_CASE ( testGetAndSetXBomb ) {
    Bomb b;
```

```
b.setX(5);
BOOST_CHECK( b.getX() == 5 );
}

//Test du get et du set Y
BOOST_AUTO_TEST_CASE ( testGetAndSetYBomb ) {
    Bomb b;
    b.setY(5);
    BOOST_CHECK( b.getY() == 5 );
}

//Test du getType
//Test constructeur paramétré
BOOST_AUTO_TEST_CASE ( testGetTypeBomb ) {
    Bomb b;
    BOOST_CHECK( b.getType() == "Bomb" );
}

/*=====
EmptyCell
=====
*/
//Test constructeur par défaut
BOOST_AUTO_TEST_CASE ( testDefaultConstructorEmptyCell ) {
    EmptyCell b;
    BOOST_CHECK( b.getType() == "EmptyCell" && b.getX() == 0 && b.getY() == 0
);
}

//Test constructeur paramétré
BOOST_AUTO_TEST_CASE ( testparamConstructorEmptyCell ) {
    EmptyCell b(5,5);
    BOOST_CHECK( b.getType() == "EmptyCell" && b.getX() == 5 && b.getY() == 5
);
}

//Test du constructeur par copie
BOOST_AUTO_TEST_CASE ( testcopyConstructorEmptyCell ) {
    EmptyCell b(5,5);
    EmptyCell c(b);
    BOOST_CHECK( b.getX() == c.getX() && b.getY() == c.getY() && b.getType() ==
c.getType() );
}

//Test de l'opérateur d'affectation
BOOST_AUTO_TEST_CASE ( testoperatoregalEmptyCell ) {
    EmptyCell b(5,5);
    EmptyCell c = b;
    BOOST_CHECK( b.getX() == c.getX() && b.getY() == c.getY() && b.getType() ==
c.getType() );
}

//Test du get et du set X
BOOST_AUTO_TEST_CASE ( testGetAndSetXEmptyCell ) {
    EmptyCell b;
    b.setX(5);
    BOOST_CHECK( b.getX() == 5 );
}

//Test du get et du set Y
BOOST_AUTO_TEST_CASE ( testGetAndSetYEmptyCell ) {
```

```
EmptyCell b;
b.setY(5);
BOOST_CHECK( b.getY() == 5 );
}

//Test du getType
//Test constructeur paramétré
BOOST_AUTO_TEST_CASE ( testGetTypeEmptyCell ) {
    EmptyCell b;
    BOOST_CHECK( b.getType() == "EmptyCell" );
}

/*=====
Digger
=====
*/
//Test constructeur par défaut
BOOST_AUTO_TEST_CASE ( testDefaultConstructorDigger ) {
    Digger b;
    BOOST_CHECK( b.getType() == "Digger" && b.getX() == 0 && b.getY() == 0 &&
                 b.getLife() == 3 );
}

//Test constructeur paramétré
BOOST_AUTO_TEST_CASE ( testparamConstructorDigger ) {
    Digger b(5,5);
    BOOST_CHECK( b.getType() == "Digger" && b.getX() == 5 && b.getY() == 5 &&
                 b.getLife() == 3 );
}

//Test du constructeur par copie
BOOST_AUTO_TEST_CASE ( testcopyConstructorDigger ) {
    Digger b(5,5);
    Digger c(b);
    BOOST_CHECK( b.getX() == c.getX() && b.getY() == c.getY() && b.getType() ==
                 c.getType() && b.getLife() == c.getLife() );
}

//Test de l'opérateur d'affectation
BOOST_AUTO_TEST_CASE ( testoperatoregalDigger ) {
    Digger b(5,5);
    Digger c = b;
    BOOST_CHECK( b.getX() == c.getX() && b.getY() == c.getY() && b.getType() ==
                 c.getType() && b.getLife() == c.getLife() );
}

//Test du get et du set X
BOOST_AUTO_TEST_CASE ( testGetAndSetXDigger ) {
    Digger b;
    b.setX(5);
    BOOST_CHECK( b.getX() == 5 );
}

//Test du get et du set Y
BOOST_AUTO_TEST_CASE ( testGetAndSetYDigger ) {
    Digger b;
    b.setY(5);
    BOOST_CHECK( b.getY() == 5 );
}
```

```
//Test du getType
//Test constructeur paramétré
BOOST_AUTO_TEST_CASE ( testGetTypeDigger ) {
    Digger b;
    BOOST_CHECK( b.getType() == "Digger" );
}

//Test lostlife
BOOST_AUTO_TEST_CASE ( testlostLifestart ) {
    Digger b;
    b.lostLife();
    BOOST_CHECK( b.getLife() == 2 );
}

//Test addlife dès le début
BOOST_AUTO_TEST_CASE ( testaddLifestartDigger ) {
    Digger b;
    b.addLife();
    BOOST_CHECK( b.getLife() == 3 );
}

//Test addlife après une perte
BOOST_AUTO_TEST_CASE ( testaddLifeDigger ) {
    Digger b;
    b.lostLife();
    b.addLife();
    BOOST_CHECK( b.getLife() == 3 );
}

//Test addlife après une perte
BOOST_AUTO_TEST_CASE ( testresetLifeDigger ) {
    Digger b;
    b.lostLife();
    b.resetLife();
    BOOST_CHECK( b.getLife() == 3 );
}

/*=====
ValueCell
=====*/
//Test constructeur par défaut
BOOST_AUTO_TEST_CASE ( testDefaultConstructorValueCell ) {
    ValueCell b;
    BOOST_CHECK( b.getType() == "ValueCell" && b.getX() == 0 && b.getY() == 0
        && ( b.getValue() >= 1 && b.getValue() <= 6 ) && b.getPoints() == ( b.
        getValue() * 10 ) );
}

//Test constructeur paramétré
BOOST_AUTO_TEST_CASE ( testparamConstructorValueCell ) {
    ValueCell b(5,5);
    BOOST_CHECK( b.getType() == "ValueCell" && b.getX() == 5 && b.getY() == 5
        && ( b.getValue() >= 1 && b.getValue() <= 6 ) && b.getPoints() == ( b.
        getValue() * 10 ) );
}

//Test du constructeur par copie
BOOST_AUTO_TEST_CASE ( testcopyConstructorValueCell ) {
    ValueCell b(5,5);
    ValueCell c(b);
```

```
BOOST_CHECK( b.getX() == c.getX() && b.getY() == c.getY() && b.getType() ==
             = c.getType() && b.getValue() == c.getValue() && b.getPoints() == c.
             getPoints());
}

//Test de l'opérateur d'affectation
BOOST_AUTO_TEST_CASE ( testoperatoregalValueCell ) {
    ValueCell b(5,5);
    ValueCell c = b;
    BOOST_CHECK( b.getX() == c.getX() && b.getY() == c.getY() && b.getType() ==
                 = c.getType() && b.getValue() == c.getValue() && b.getPoints() == c.
                 getPoints());
}

//Test du get et du set X
BOOST_AUTO_TEST_CASE ( testGetAndSetXValueCell ) {
    ValueCell b;
    b.setX(5);
    BOOST_CHECK( b.getX() == 5 );
}

//Test du get et du set Y
BOOST_AUTO_TEST_CASE ( testGetAndSetYValueCell) {
    ValueCell b;
    b.setY(5);
    BOOST_CHECK( b.getY() == 5 );
}

//Test du getType
BOOST_AUTO_TEST_CASE ( testgetTypeValueCell ) {
    ValueCell b;
    BOOST_CHECK( b.getType() == "ValueCell" );
}

//Test getValue
BOOST_AUTO_TEST_CASE ( testgetValueValueCell ) {
    ValueCell b;
    BOOST_CHECK( b.getValue() >= 1 && b.getValue() <= 6 );
}

//Test getPoints
BOOST_AUTO_TEST_CASE ( testgetPointsValueCell ) {
    ValueCell b;
    BOOST_CHECK( b.getPoints() == ( b.getValue() * 10 ) );
}

/*=====
GoldCell
=====*/
//Test constructeur par défaut
BOOST_AUTO_TEST_CASE ( testDefaultConstructorGoldCell ) {
    GoldCell b;
    BOOST_CHECK( b.getType() == "GoldCell" && b.getX() == 0 && b.getY() == 0 &
                & ( b.getValue() >= 1 && b.getValue() <= 6 ) && b.getPoints() >= ( (b.
                getValue() * 10 ) + 10 ) && b.getPoints() <= ( (b.getValue() * 10 ) +
                100 ) );
}

//Test constructeur paramétré
```

```
BOOST_AUTO_TEST_CASE ( testparamConstructorGoldCell ) {
    GoldCell b(5,5);
    BOOST_CHECK( b.getType() == "GoldCell" && b.getX() == 5 && b.getY() == 5 &
        & ( b.getValue() >= 1 && b.getValue() <= 6 ) && b.getPoints() >= ( (b.
        getValue() * 10 ) + 10 ) && b.getPoints() <= ( (b.getValue() * 10 ) +
        100 ) );
}

//Test du constructeur par copie
BOOST_AUTO_TEST_CASE ( testcopyConstructorGoldCell ) {
    GoldCell b(5,5);
    GoldCell c(b);
    BOOST_CHECK( b.getX() == c.getX() && b.getY() == c.getY() && b.getType() ==
        = c.getType() && b.getValue() == c.getValue() && b.getPoints() == c.
        getPoints());
}

//Test de l'opérateur d'affectation
BOOST_AUTO_TEST_CASE ( testoperatoregalGoldCell ) {
    GoldCell b(5,5);
    GoldCell c = b;
    BOOST_CHECK( b.getX() == c.getX() && b.getY() == c.getY() && b.getType() ==
        = c.getType() && b.getValue() == c.getValue() && b.getPoints() == c.
        getPoints());
}

//Test du get et du set X
BOOST_AUTO_TEST_CASE ( testGetAndSetXGoldCell ) {
    GoldCell b;
    b.setX(5);
    BOOST_CHECK( b.getX() == 5 );
}

//Test du get et du set Y
BOOST_AUTO_TEST_CASE ( testGetAndSetYGoldCell ) {
    GoldCell b;
    b.setY(5);
    BOOST_CHECK( b.getY() == 5 );
}

//Test du getType
BOOST_AUTO_TEST_CASE ( testgetTypeGoldCell ) {
    GoldCell b;
    BOOST_CHECK( b.getType() == "GoldCell" );
}

//Test getValue
BOOST_AUTO_TEST_CASE ( testgetValueGoldCell ) {
    GoldCell b;
    BOOST_CHECK( b.getValue() >= 1 && b.getValue() <= 6 );
}

//Test getPoints
BOOST_AUTO_TEST_CASE ( testgetPointsGoldCell ) {
    GoldCell b;
    BOOST_CHECK( b.getPoints() >= ( b.getValue() * 10 + 10 ) && b.getPoints()
        <= ( b.getValue() * 10 + 100 ) );
}

/*=====
```

```
Score
=====*/
BOOST_AUTO_TEST_CASE( testaddPointsandGetCurrent ) {
    Score s;
    s.addPoints(10);
    BOOST_CHECK( s.getCurrent() == 10 );
}

BOOST_AUTO_TEST_CASE( testresetPoints ) {
    Score s;
    s.addPoints(10);
    s.resetScore();
    BOOST_CHECK( s.getCurrent() == 0 );
}

BOOST_AUTO_TEST_CASE( testaddSuccessandGetCurrentStep ) {
    Score s;
    s.addSuccess();
    BOOST_CHECK( s.getCurrentStep() == 2 );
}

BOOST_AUTO_TEST_CASE(testgetGlobale) {
    Score s;
    s.addPoints(10);
    BOOST_CHECK( s.getGlobale() == 0 );
    s.addSuccess();
    BOOST_CHECK( s.getGlobale() == 10 );
}

/*=====
Level
=====*/
BOOST_AUTO_TEST_CASE( testgetGoal ) {
    Score *s;
    Level l(s);
    BOOST_CHECK( l.getGoal() == 10 );
}

BOOST_AUTO_TEST_CASE( testgetCurrentMove ) {
    Score *s;
    Level l(s);
    BOOST_CHECK( l.getCurrentMove() == 0 );
}

BOOST_AUTO_TEST_CASE( testtimeIsUp ) {
    Score *s;
    Level l(s);
    BOOST_CHECK( l.timeIsUp() == 0 );
}

BOOST_AUTO_TEST_CASE( testleftTime) {
    Score *s;
    Level l(s);
    BOOST_CHECK( l.leftTime() == 60 );
}

BOOST_AUTO_TEST_CASE( testisDead ) {
```

```
Score *s;
Level l(s);
BOOST_CHECK( l.isDead() == false );
}

BOOST_AUTO_TEST_CASE( testwin ) {
    Score *s;
    Level l(s);
    BOOST_CHECK( l.win() == false );
}

BOOST_AUTO_TEST_CASE( testlose ) {
    Score *s;
    Level l(s);
    BOOST_CHECK( l.lose() == false );
}

/*=====
GameModel
=====*/
BOOST_AUTO_TEST_CASE( testgameOver ) {
    GameModel g;
    BOOST_CHECK( g.gameOver() == false );
}

/*BOOST_AUTO_TEST_CASE( testgetScore ) {
    GameModel g;
    BOOST_CHECK( g.getScore() == 0 );
    cout << g.getScore() << endl;
}

BOOST_AUTO_TEST_CASE( testgetLevel ) {
    GameModel g;
    BOOST_CHECK( g.getLevel() == 0 );
    cout << g.getLevel() << endl;
}*/
```