

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

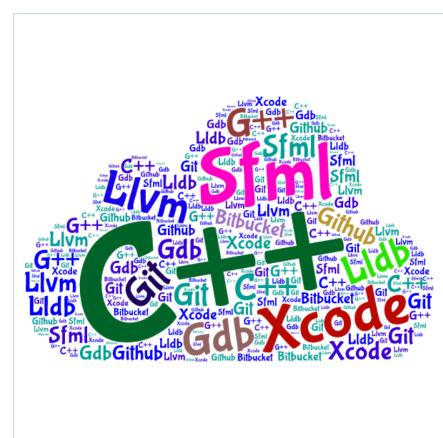
Tests Unitaires : Boost

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

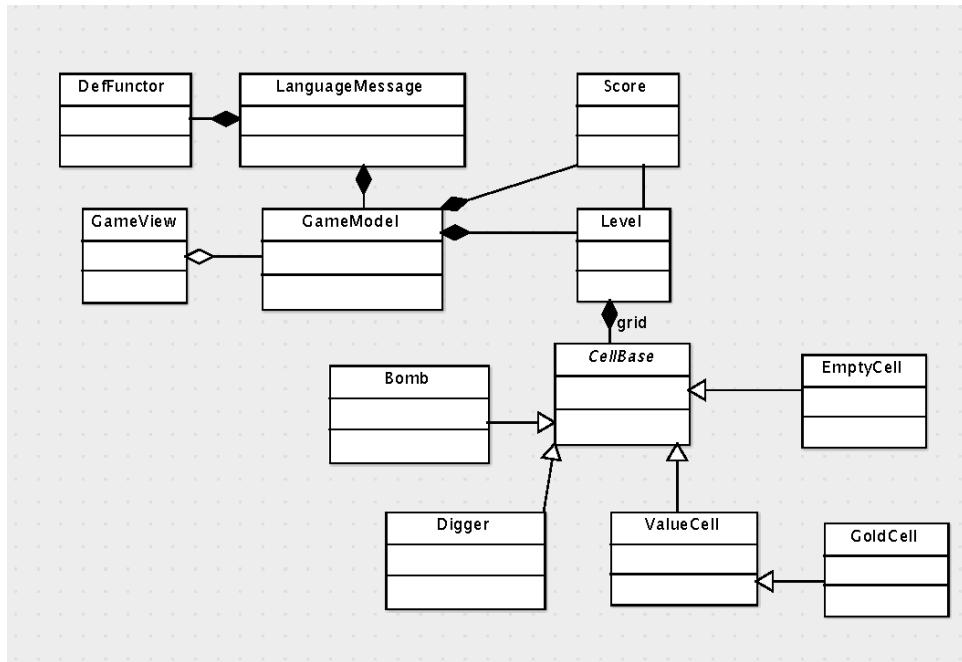
- Gestion du temps (l'objectif du niveau doit être réalisé en un temps donné)
- Animation des déplacement
- Ajout de bonus dans les cases numérotées
- Gestion des meilleurs scores avec un affichage d'un écran des meilleurs scores depuis le menu.
- Ajout de son

Objectifs bonus

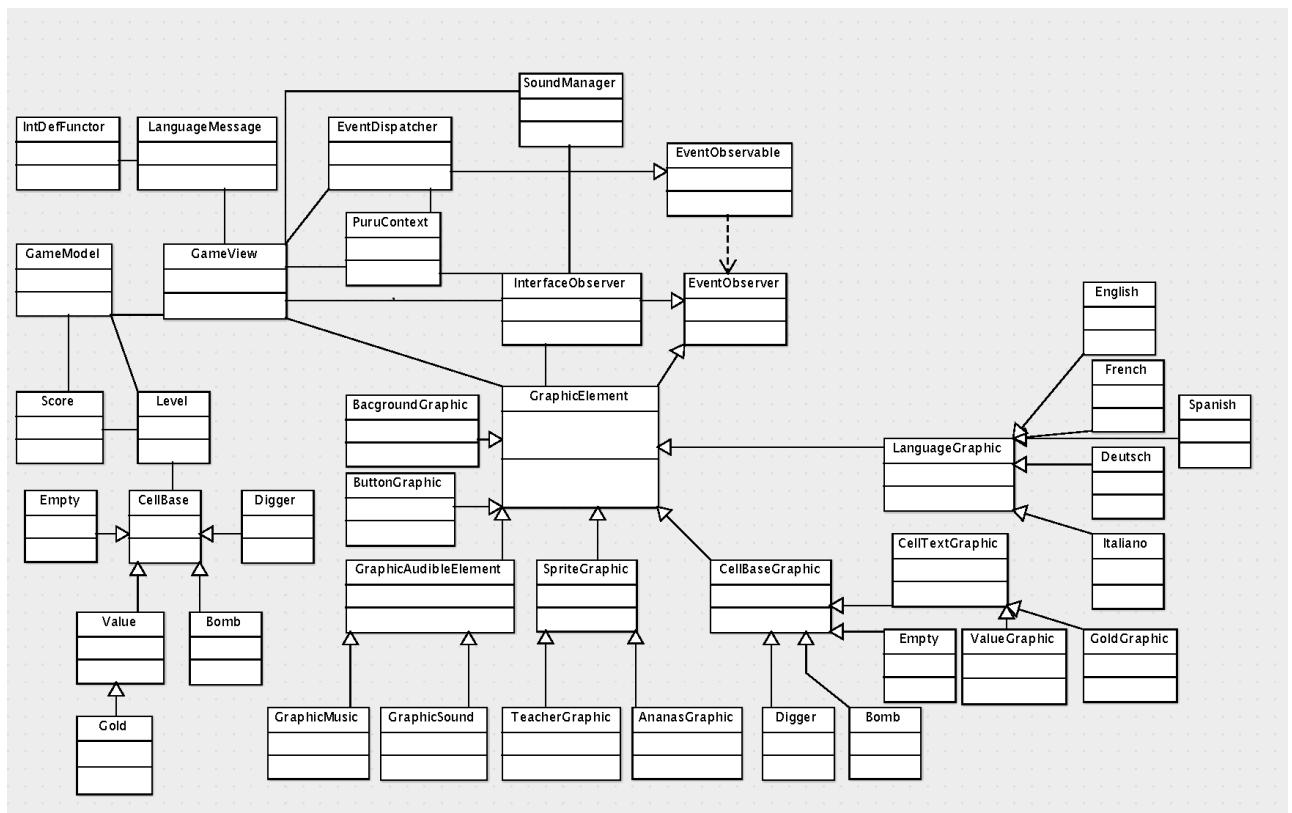
- Générateur aléatoire de niveau
- Nombre indéfini de niveau
- Changement de la charte graphique
- Support multi langues
- Éditeur de niveau
- Vérificateur de faisabilité de niveau

Dossier de conception

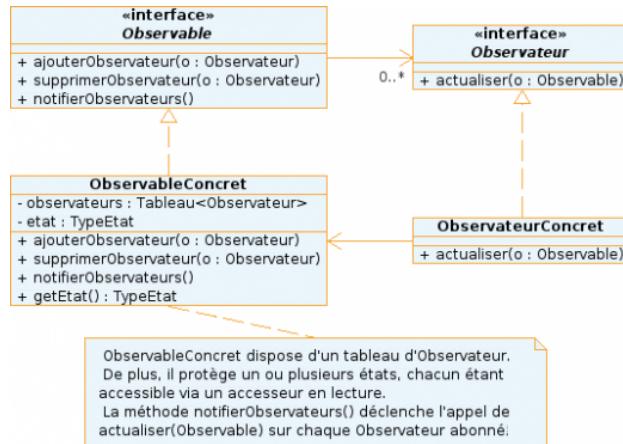
Tout d'abord, nous devions faire un rendu en mode Console, dont voici notre modèle de conception.



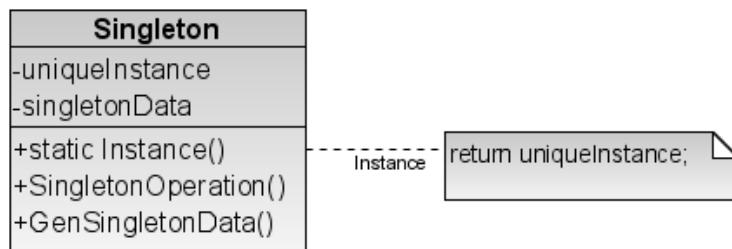
Mais après mûre réflexion, lorsque nous sommes en mode SFML nous avons ajouté des choses à notre conception que voici ci dessous.



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.

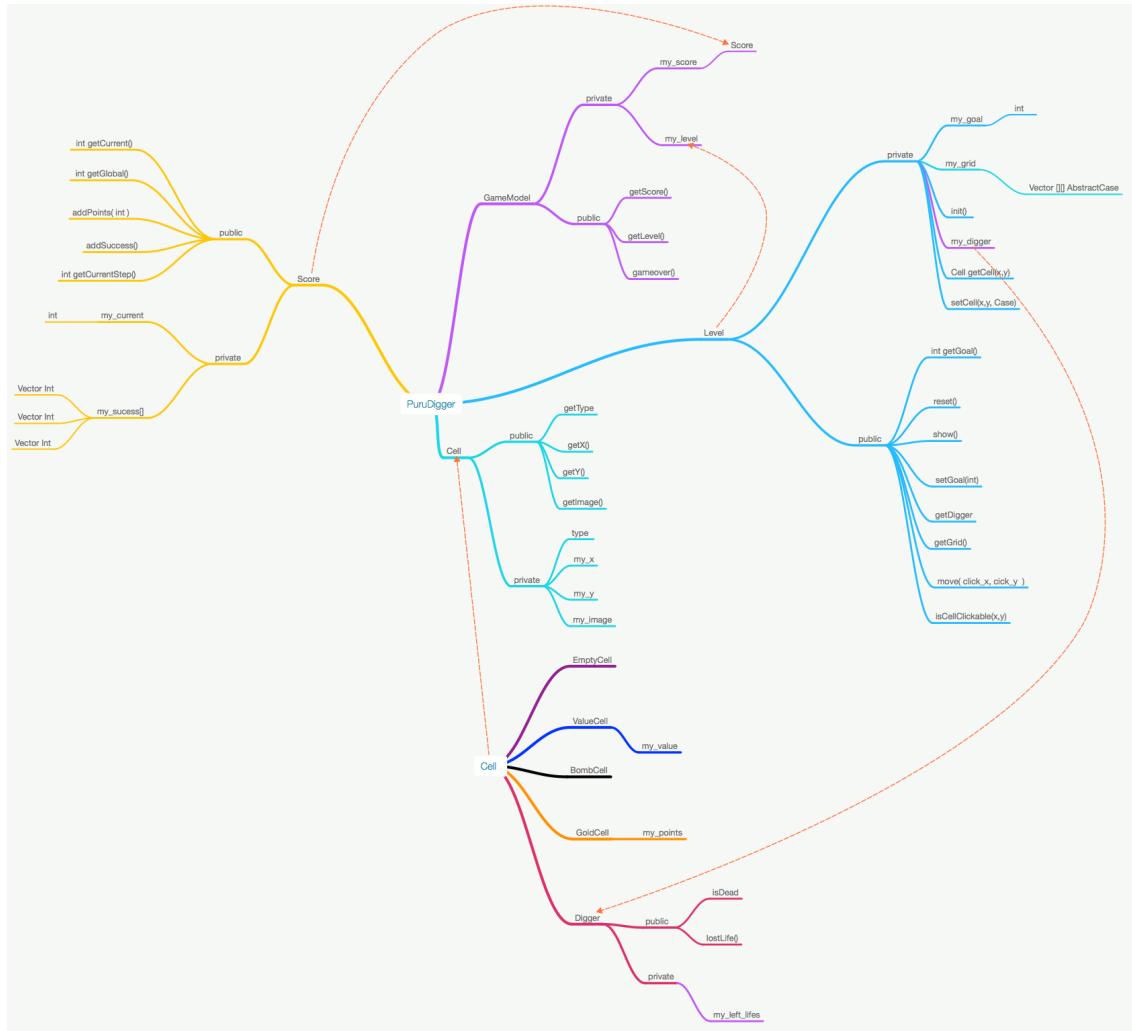


Ensuite nous avons décidé d'implémenter un Pattern Singleton afin de gerer une unique instance de la classe. Nous nous servons donc de ce modèle de conception pour faire un Manager de son afin de pouvoir jouer les sons beaucoup plus facilement et n'importe où dans le code. On sait alors où sont regroupes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

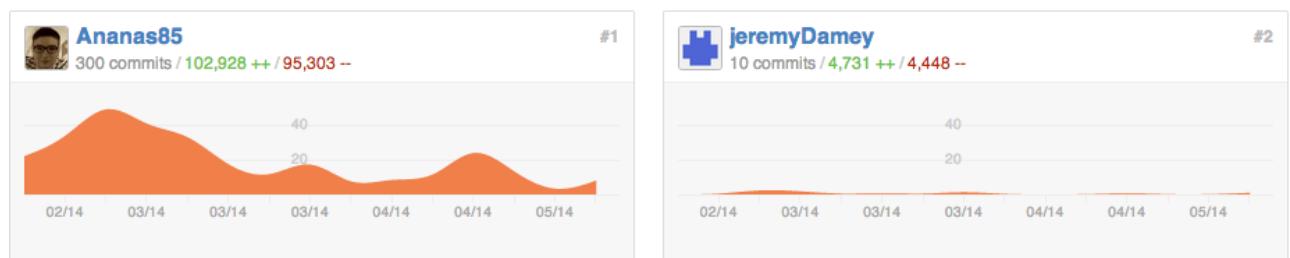
Ensuite la documentation fut fournie par Jérémy sous forme de commentaire Doxygen puis générée par Anaël et hébergée sur son site Web : <http://ananascorp.plopix.net/projet/untitled/html/index.html>.

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.

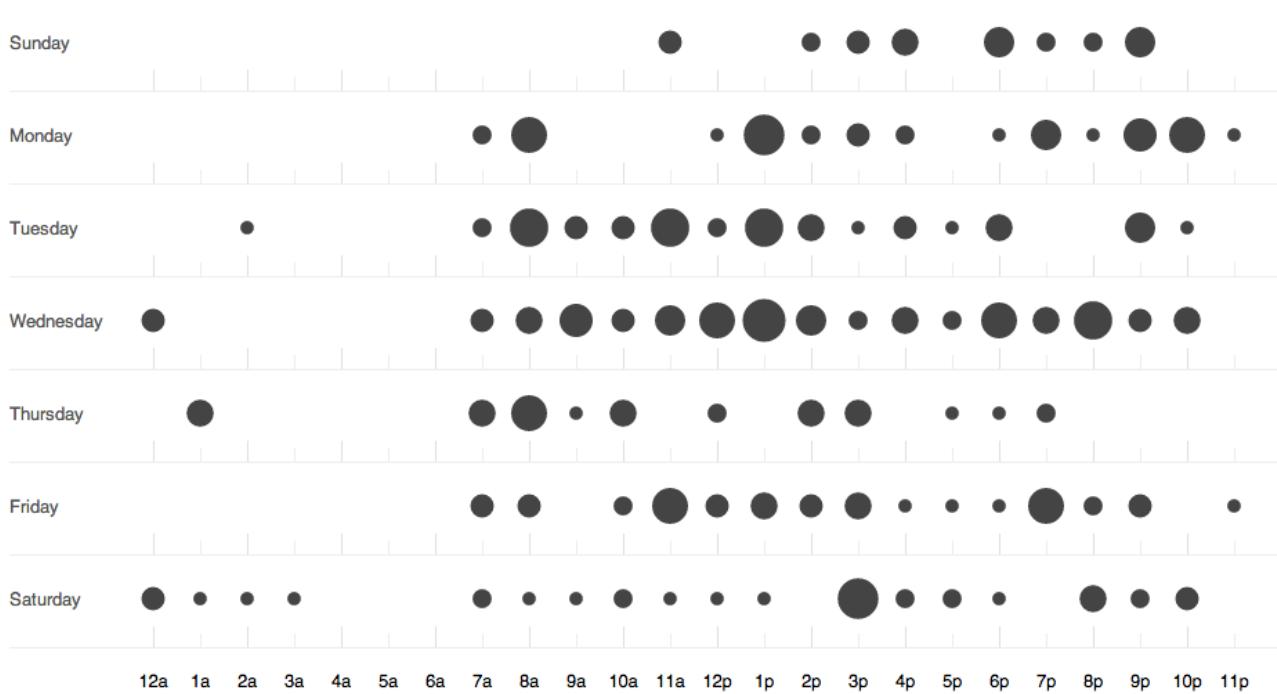
Vous pouvez voir ci dessous la courbe de travail, nous avons commencé très vite et très fort afin de ne pas être enseveli sous un charge de travail trop importante.



Contributeurs



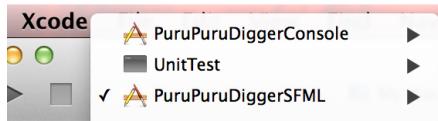
Punch Cards



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.



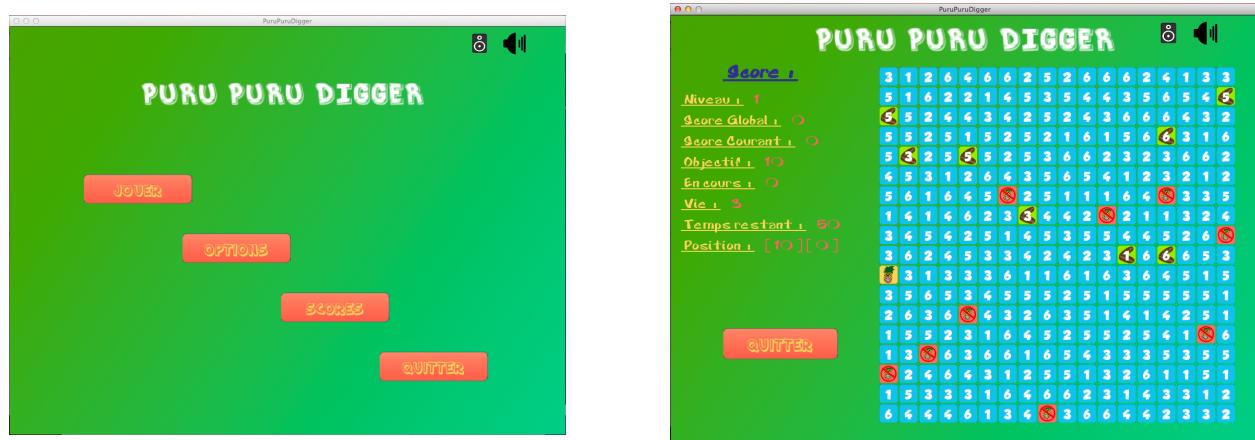
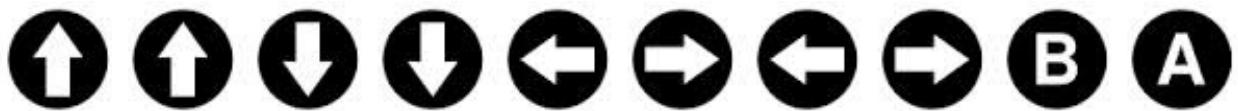
Si pendant le déplacement vous touchez une bombe, une case vide ou sortez du plateau vous perdez une vie et êtes téléporté dans un nouveau niveau de même objectif.

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.

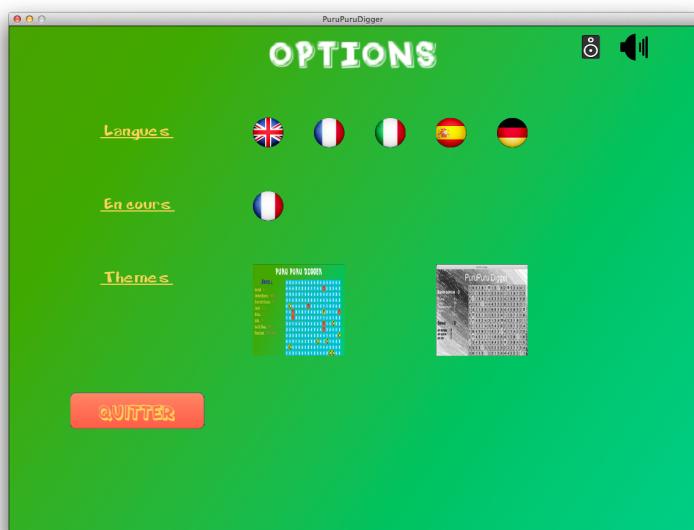
Manuel de l'interface en mode SFML :

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

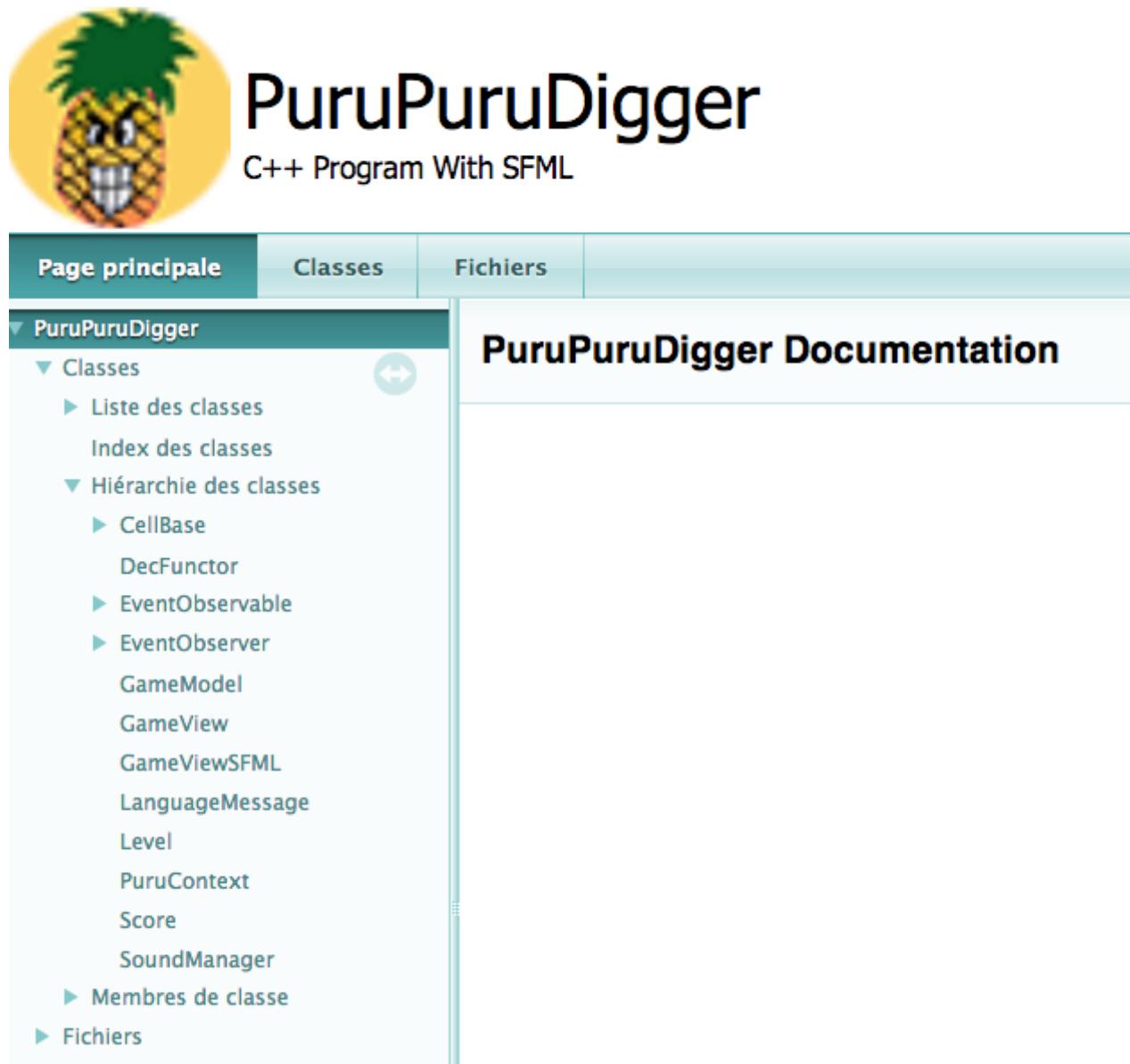
Du point de vue de notre cahier des charges, toutes les contraintes sont respectées. Tous les objectifs principaux ont été atteints comme nous pouvons le voir dans le cahier des charges. Nous disposons également de toutes les fonctionnalités supplémentaires ainsi que quelques fonctionnalités bonus.

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 lists various documentation sections: "PuruPuruDigger" (expanded), "Classes" (expanded), "Liste des classes", "Index des classes", "Hiérarchie des classes" (expanded), "CellBase", "DecFunctor", "EventObservable", "EventObserver", "GameModel", "GameView", "GameViewSFML", "LanguageMessage", "Level", "PuruContext", "Score", "SoundManager", "Membres de classe" (expanded), and "Fichiers".

Le code source

mai 22, 14 20:37

Constantes.h

Page 1/2

```

#ifndef purpurudigger_Constantes_h
#define purpurudigger_Constantes_h

/***
 * \file Constantes.h
 * \brief Les constantes
 * \author CHARDAN Anaël
 * \author DAMEY Jérôme
 * \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 WINDOWWIDTH = 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 CASEWIDTH = 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;

```

mai 22, 14 20:37

Constantes.h

Page 2/2

```

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 BUTTONNORMSX = 2;
const int BUTTONNORMEX = 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 = WINDOWWIDTH - ( 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
#ifdef __linux__
const std::string FILEBESTSCORE = "Ressources/bestScores.txt";
#else
const std::string FILEBESTSCORE = "bestScores.txt";
#endif

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

/*
 * \enum Message
 * \brief Voici l'enumeration 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
```

mai 22, 14 20:37

GameModel.h

Page 1/1

```

#ifndef __purpurudigger__Game__
#define __purpurudigger__Game__

/***
 * \file GameModel.h
 * \brief Ce que reprÃ©sente une partie
 * \author CHARDAN AnaÃ«l
 * \author DAMEY JÃ©rÃ©my
 * \date 09/03/2014
 */

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

/*! \class GameModel
 * \brief Classe modÃ©lisant une partie
 */

class GameModel {
private :
    Level* my_level; /*!< Nos Levels ( en vÃ©ritÃ© 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 Ã  notre grille
     *
     * \param[in] le mouvement
     */
    void orderMovement( int depl );

    /*!
     * \brief Ordonner un mouvement Ã  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 terminÃ©e
     *
     * \return true si la partie est finie
     */
    bool gameOver() const ;

    void reset();
};

#endif /* defined(__purpurudigger__Game__) */

```

mai 22, 14 20:37

GameView.h

Page 1/1

```

#ifndef __PuruPuruDigger__GameView__
#define __PuruPuruDigger__GameView__
/***
 * \file GameView.h
 * \brief Affichage de notre partie en mode terminal
 * \author CHARDAN AnaÃ«l
 * \author DAMEY JÃ©rÃ©my
 * \date 09/03/2014
 */

#include <iostream>

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

/*! \class GameView
 * \brief Classe modÃ©lisant ce qu'est une vue
 */

class GameView {
private :
    Language my_language; /*!< La langue de notre partie */
    LanguageMessage my_messages; /*!< La bibliothÃ¨que 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 modÃ©le 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 dÃ©placement
     */
    void showInstruction() ;

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

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

public:
    GameView();
    /**
     * \brief Injection du modÃ©le Ã  la vue
     *
     * \param[in] model : le modÃ©le Ã  interprÃ©ter
     */
    void setModel(GameModel * model);

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

#endif /* defined(__PuruPuruDigger__GameView__) */

```

mai 22, 14 20:37

GameViewSFML.h

Page 1/2

```

#ifndef __PuruPuruDigger__GameViewSFML__
#define __PuruPuruDigger__GameViewSFML__

/***
 * \file GameViewSFML.h
 * \brief Notre classe GameViewSFML
 * \author CHARDAN Anaël
 * \author DAMEY JÃ©rÃ©my
 * \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 les meilleures scores
     */
    void initBestScore();
}

```

mai 22, 14 20:37

GameViewSFML.h

Page 2/2

```
/*!
 * \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__) */
```

mai 22, 14 20:37

IntDecFunctor.h

Page 1/1

```
#ifndef __PuruPuruDigger__IntDecFunctor__
#define __PuruPuruDigger__IntDecFunctor__

/***
 * \file IntDecFunctor.h
 * \brief Notre foncteur pour trier une map à l'envers
 * \author CHARDAN Anaïl
 * \author DAMEY Jérôme
 * \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 bool@en pour le sens
     */
    bool operator()( T n1, S n2 ) { return n2 < n1; }
};

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

mai 22, 14 20:37

LanguageMessage.h

Page 1/1

```
#ifndef __PuruPuruDigger__LanguageMessage__
#define __PuruPuruDigger__LanguageMessage__

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

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

/*! \class LanguageMessage
 * \brief Classe pour enregistrer notre multilingue */
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__) */
```

mai 22, 14 20:37

Level.h

Page 1/3

```

#ifndef __purpurudigger_Level__
#define __purpurudigger_Level__

/***
 * \file Level.h
 * \brief Les mÃ©thodes liÃ©es Ã notre classe level
 * \author CHARDAN AnaÃ«l
 * \author DAMEY JÃ©rÃ©my
 * \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'Ã©criture */

/*!
 * \class Level
 * \brief Classe modÃ©lisant 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 DÃ©placement du Digger dans la grille
     *
     * \param[in] DeltaX La direction vertical de dÃ©placement du digger
     * \param[in] DeltaY La direction horizontal de dÃ©placement du digger
     */
    void move( int deltaX, int deltaY );

    /*!
     * \brief MÃ©lange d'un tableau en 2D
     */
    void shuffle();

    /*!
     * \brief GÃ©nÃ©ration de la premiere grille
     */
    void initGrid();


public:
    /*!
     * \brief Nouvelle grille aprÃ¨s avoir gagnÃ©
     */
    void winLevel();

    /*!
     * \brief MÃ©thode pour regÃ©nerer une grille aprÃ¨s avoir perdu ou gagnÃ© sans perdre les attributs de notre Digger
     */
    void reset();
    /*!
     * \brief Constructeur
     *
     * Constructeur de la classe Level
     *
     * \param *score : score de notre partie (injection de dÃ©pendance )
     */
    Level(Score* score);

    /*!
     * \brief Destructeur
     *
     * Destructeur de la classe Level
     */
}

```

mai 22, 14 20:37

Level.h

Page 2/3

```

/*
~Level();

/*!
 * \brief Perdre le level
 *
 * Faire perdre une vie et regeneration d'un niveau
 */
void lostLevel();

/*!
 * \brief remettre win a false
 *
 * remet l'attribut win false
 */
void resetWin();

/*!
 * \brief remettre lose a false
 *
 * remet l'attribut lose false
 */
void resetLose();

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

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

/*!
 * \brief Connaître si le temps est à coul
 *
 * \return true si le temps est à coul
 */
bool timeIsUp() const;

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

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

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

/*!
 * \brief Connaître si l'on vient juste de perdre un niveau ( affichage )
 *
 */

```

mai 22, 14 20:37

Level.h

Page 3/3

```

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

//Tous nos sucres de languages, 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 méthode move()
 * \see déplacement
 */
void moveWest();

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

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

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

/*
 * \brief Sucre pour se déplacer en haut à droite ( deltaX -1, deltaY 1, voir la méthode move()
 * \see déplacement
 */
void moveNorthEast();

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

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

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

};

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

```

#ifndef __PuruPuruDigger__PuruContext__
#define __PuruPuruDigger__PuruContext__

/***
 * \file PuruContext.h
 * \brief Notre classe PuruContext
 * \author CHARDAN Anaïl
 * \author DAMEY Jérôme
 * \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 épuisé */
    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 activée 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
     */

```

mai 22, 14 20:37

PuruContext.h

Page 2/3

```

/*
 * Affecte si oui ou non la partie est fini
 */
void setOver( bool set );

/*
 * \brief Setteur du temps
 *
 * Affecte si oui ou non le temps est coulÃ©
 */
void setTimeOver( bool set );

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

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

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

/*
 * \brief Setteur de langage
 *
 * Affecte le langage utilisÃ©
 */
void setLanguage( Language language );

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

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

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

/*
 * \brief Getteur des meilleurs score
 *
 * retourne vrai ou faux en fonction si on est en train de saisir un score
 */
bool isEnterABestScore() const;

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

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

/*
 * \brief Getteur de over
 *
 * retourne vrai ou faux en fonction si la parti est fini
*/

```

mai 22, 14 20:37

PuruContext.h

Page 3/3

```
/*
bool isOver() const;

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

<*/
* \brief Guetteur du menu principales meilleures score
* retourne vrai ou faux en fonction si on est dans le menu des meilleures 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 langage
*/
Language getLanguage() const;
};

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

mai 22, 14 20:37

Score.h

Page 1/1

```

#ifndef __purpurudigger__Score__
#define __purpurudigger__Score__

/***
 * \file Score.h
 * \brief Notre classe Score
 * \author CHARDAN Anaïl
 * \author DAMEY Jérôme
 * \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

```

mai 22, 14 20:37

Utils.h

Page 1/1

```

#ifndef __PuruPuruDigger_Utils__
#define __PuruPuruDigger_Utils__

/***
 * \file Utils.h
 * \author CHARDAN Anaïl
 * \author DAMEY Jérôme
 * \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__) */

```

mai 22, 14 20:37

GameModel.cpp

Page 1/2

```
/*
 * \file GameModel.cpp
 * \brief Ce que reprÃ©sente une partie
 * \author CHARDAN AnaÃ«l
 * \author DAMEY JÃ©rÃ©my
 * \date 09/03/2014
 */

#include "GameModel.h"
using namespace std;

GameModel::GameModel(){
    my_score = new Score();
    my_level = new Level( my_score );
}

GameModel::~GameModel() {
    delete my_level;
    delete my_score;
}

void
GameModel::orderMovement( int depl ) {
    switch ( depl ) {
        case 1 : my_level->moveSouthWest();
            break;
        case 2 : my_level->moveSouth();
            break;
        case 3 : my_level->moveSouthEast();
            break;
        case 4 : my_level->moveWest();
            break;
        case 6 : my_level->moveEast();
            break;
        case 7 : my_level->moveNorthWest();
            break;
        case 8 : my_level->moveNorth();
            break;
        case 9 : my_level->moveNorthEast();
            break;
    }
}

void
GameModel::orderMovement( int xclick, int yclick ) {
    if ( my_level->isCellClickable( xclick, yclick ) ) {
        if ( xclick - my_level->getDigger()->getX() == 1 && yclick - my_level->getDigger()->getY() == -1 ) {
            my_level->moveSouthWest();
            my_movement = SWest;
        } else if ( xclick - my_level->getDigger()->getX() == 1 && yclick - my_level->getDigger()->getY() == 0 ) {
            my_level->moveSouth();
            my_movement = South;
        } else if ( xclick - my_level->getDigger()->getX() == 1 && yclick - my_level->getDigger()->getY() == 1 ) {
            my_level->moveSouthEast();
            my_movement = SEast;
        } else if ( xclick - my_level->getDigger()->getX() == 0 && yclick - my_level->getDigger()->getY() == 1 ) {
            my_level->moveEast();
            my_movement = East;
        } else if ( xclick - my_level->getDigger()->getX() == -1 && yclick - my_level->getDigger()->getY() == 1 ) {
            my_level->moveNorthEast();
            my_movement = NEast;
        } else if ( xclick - my_level->getDigger()->getX() == -1 && yclick - my_level->getDigger()->getY() == 0 ) {
            my_level->moveNorth();
            my_movement = North;
        } else if ( xclick - my_level->getDigger()->getX() == -1 && yclick - my_level->getDigger()->getY() == -1 ) {
            my_level->moveNorthWest();
            my_movement = Nwest;
        } else if ( xclick - my_level->getDigger()->getX() == 0 && yclick - my_level->getDigger()->getY() == -1 ) {
            my_level->moveWest();
            my_movement = West;
        }
    }
}

bool
GameModel::gameOver() const {
    if ( my_level->isDead() )
        return true;
    else
        return false;
}

Score* const
GameModel::getScore() {
    return my_score;
}

Level* const
GameModel::getLevel() {
    return my_level;
}

void
GameModel::reset() {
    delete my_score;
    delete my_level;
    my_score = new Score();
    my_level = new Level( my_score );
}
```

mai 22, 14 20:37

GameModel.cpp

Page 2/2

```
}
```

```
Movement
```

```
GameModel::getMovement() const {
```

```
    return my_movement;
```

```
}
```

mai 22, 14 20:37

GameView.cpp

Page 1/4

mai 22, 14 20:37

GameView.cpp

Page 2/4

```

        }

        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 << col
orMessage(":", 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] << "      " << 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 opening 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();
        }
    }
}

```

mai 22, 14 20:37

GameView.cpp

Page 3/4

```

} 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 VM-^Nifie le temps, s'il est M-^Ncoulem-^N
            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 VM-^Nifie 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;
}
```

mai 22, 14 20:37

GameViewSFML.cpp

Page 1/3

```
/*
 * \file GameViewSFML.cpp
 * \brief Notre classe GameViewSFML
 * \author CHARDAN Ana«l
 * \author DAMEY JÃ©rÃ©my
 * \date 09/03/2014
 */

#include "GameViewSFML.h"

#include "Observers/InterfaceObserver.h"
#include "Graphics/EnglishGraphic.h"
#include "Graphics/FrenchGraphic.h"
#include "Graphics/SpanishGraphic.h"
#include "Graphics/DeutschGraphic.h"
#include "Graphics/ItalianoGraphic.h"
#include "Manager/SoundManager.h"

using namespace sf;

//Constructeur
GameView::GameView() {
    //Le style de la fenÃªtre
    my_window = new RenderWindow( VideoMode(WINDOWWIDTH, WINDOWHEIGHT, BPP), "PuruPuruDigger", Style::Close);

    //On bloque le rafraichissement Ã  60 par seconde
    my_window->SetFramerateLimit(60);

    // buttons
    my_playButton = new ButtonGraphic();
    my_settingButton = new ButtonGraphic();
    my_bestButton = new ButtonGraphic();
    my_quitButton = new ButtonGraphic();

    my_soundIcon = new GraphicSound();
    my_musicIcon = new GraphicMusic();

    my_ananasSprite = new AnanasSprite();
    my_teacherSprite = new TeacherSprite();

    my_background = new BackgroundGraphic();

    my_languageToSprite = new std::map<Language, LanguageGraphic*>();

    my_languageToSprite->operator[](english) = new EnglishGraphic();
    my_languageToSprite->operator[](francais) = new FrenchGraphic();
    my_languageToSprite->operator[](italiano) = new ItalianoGraphic();
    my_languageToSprite->operator[](espanol) = new SpanishGraphic();
    my_languageToSprite->operator[](deutsch) = new DeutschGraphic();

    // dispatch d'event en tout genre
    my_context = new PuruContext();
    SoundManager::getInstance()->setContext( my_context );
    my_eventDispatcher = new EventDispatcher( my_context );
}

//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 dÃ©pendance 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 );
}
```

mai 22, 14 20:37

GameViewSFML.cpp

Page 2/3

```

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'@v@nement
void GameView::treatGame( ) {

    InterfaceObserver* interfaceObserver = new InterfaceObserver( 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->addObserver( interfaceObserver );
    my_eventDispatcher->addObserver( my_soundIcon );
    my_eventDispatcher->addObserver( my_musicIcon );
    my_eventDispatcher->addObserver( my_background );

    // On abonne quand mème pour le change theme
    // cela sera désabonner dans le init
}

```

mai 22, 14 20:37

GameViewSFML.cpp

Page 3/3

```

for ( std::map<Language, LanguageGraphic*>::const_iterator it = my_languageToSprite->begin() ; it!=my_languageToSprite
te->end(); ++it) {
    my_eventDispatcher->addObserver( (*my_languageToSprite)[ it->first ] );
}
my_eventDispatcher->addObserver(my_ananasSprite);
my_eventDispatcher->addObserver(my_teacherSprite);
my_eventDispatcher->addObserver( my_playButton );
my_eventDispatcher->addObserver( my_bestButton );
my_eventDispatcher->addObserver( my_settingButton );
my_eventDispatcher->addObserver( 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) ) {
                            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.MouseBu
tton.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 interfaceObserver;
}

```

mai 22, 14 20:37

IntDecFunctor.cpp

Page 1/1

```
//  
// IntDecFunctor.cpp  
// PuruPuruDigger  
// Created by Ananas-Mac on 04/03/2014.  
//  
#include "IntDecFunctor.h"
```

mai 22, 14 20:37

LanguageMessage.cpp

Page 1/3

```
/***
 * \file LanguageMessage.cpp
 * \brief Notre classe LanguageMessage
 * \author CHARDAN Anaïl
 * \author DAMEY JÃ©rÃ©my
 * \date 09/03/2014
 */
#include "LanguageMessage.h"

using namespace std;

LanguageMessage::LanguageMessage() {
    //Les messages franÃ§ais
    my_languages[francais][choice] = "Choix";
    my_languages[francais][move] = "DÃ©placement";
    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";
}
```

mai 22, 14 20:37

LanguageMessage.cpp

Page 2/3

```

my_languages[espanol][score] = " Puntuacion ";
my_languages[espanol][level] = " Nivel ";
my_languages[espanol][global] = " Puntuacion Global ";
my_languages[espanol][current] = " Clasificacion 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 perdió 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][theme] = " 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>&

```

mai 22, 14 20:37

LanguageMessage.cpp

Page 3/3

```
LanguageMessage::operator[](Language langue) {
    return my_languages[langue];
}
```

mai 22, 14 20:37

Level.cpp

Page 1/5

```
/*
 * \file Level.cpp
 * \brief Les mÃ©thodes liÃ©es Ã notre classe level
 * \author CHARDAN AnaÃ«l
 * \author DAMEY JÃ©rÃ©my
 * \date 09/03/2014
 */

#include "Level.h"
#include "Cell/Bomb.h"
#include "Cell/GoldCell.h"
#include "Cell/EmptyCell.h"
#include <vector>
#include <algorithm>

using namespace std;
/*=====
Le constructeur
=====*/
Level::Level(Score* score): my_score(score) {
    //On met nos mouvements courants Ã 0
    my_currentMove = 0;

    //Comme c'est le premier niveau, l'objectif est 10
    my_goal = 10;

    //Comme c'est le premier niveau, le bonus gÃ©nÃ©rÃ© par le level est 500
    my_bonus = 500;

    //On donne l'instant prÃ©sent Ã  notre dÃ©part
    time(&my_depart);
    timeGoal = 60;

    //On alloue le digger
    my_digger = new Digger(0,0);

    //On bloque la taille de notre vecteur
    my_grid.resize( LIGNE );

    for( int i = 0; i < LIGNE; i++ ) {
        my_grid[i].resize( COLONNE );
    }

    my_win = false;
    my_lose = false;

    //On fait pointÃ© notre case sur notre digger;
    my_grid[0][0] = my_digger;

    //On appelle la fonction n'initialisation
    reset();
}

/*=====
Le destructeur
=====*/
Level::~Level() {
    for ( int i = 0; i < LIGNE; i++ ) {
        for ( int j = 0; j < COLONNE; j++ ) {
            delete my_grid[i][j];
        }
    }
}

/*=====
Les mÃ©thodes privÃ©es
=====*/
void
Level::move( int DeltaX, int DeltaY ) {
    int nbStep = -1;
    int pointInGame = -1;

    if ( isCellClickable( ( my_digger->getX() + DeltaX ), ( my_digger->getY() + DeltaY ) ) ) {
        //On trans-type la case concerner, ce trans-type n'a pas besoin d'Ãªtre vÃ©rifier
        ptr_valueCell = dynamic_cast<ValueCell*>(my_grid[ (my_digger->getX() + DeltaX) ][ (my_digger->getY() + DeltaY) ]);

        //On prend les nbStep Ã 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 Ã 0
    int cpt = 0;

    //Tant que l'on Ã  pas fait le bon nombre de coup et que la case d'a cÃ´tÃ© est bien une gold ou une value
}
```

mai 22, 14 20:37

Level.cpp

Page 2/5

```

while ( cpt < nbStep && ( isCellClickable( ( my_digger->getX() + DeltaX ), ( my_digger->getY() + DeltaY ) ) ) ) {
    ///Si l'on rencontre un trÃ©sor pendant un dÃ©placement
    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 alÃ©atoire pour voir si l'on va gagner du temps, de la vie, ou des points
        int aleat = randomNumber(0,2);
        ///On fait les vÃ©rifications et on fait les modifications en consÃ©quences
        switch ( aleat ) {
            ///On gagne des points
            case 0 :
                my_score->addPoints( ptr_goldCell->getPoints() );
                break;
            ///On remet le temps Ã 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 dÃ©placement s'est mal passÃ© ( donc cpt a bougÃ© et est diffÃ©rent 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++;
            }
        }
    }
}

```

mai 22, 14 20:37

Level.cpp

Page 3/5

```

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 trÃ©sors en fonction du nombre de bomb
    for ( int i = nbrB +1 ; i < COLONNE ; i++ ) {
        my_grid[0][i] = new GoldCell();
    }

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

    //On mÃ©lange 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 difficultÃ©
    my_goal +=10;
    //On reset le level
    reset();
}

void
Level::reset() {
    //On remet nos mouvements Ã  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 mÃ©thodes 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;
}

```

mai 22, 14 20:37

Level.cpp

Page 4/5

```

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 vÃ©rifier 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 vÃ©rifier que la case est juste Ã  cÃ´tÃ© 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 vÃ©rifie 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 sures
=====*/
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 );
}

```

mai 22, 14 20:37

Level.cpp

Page 5/5

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

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

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

mai 22, 14 20:37

main.cpp

Page 1/1

```
/**\n * \file main.cpp\n * \brief Notre main terminal\n * \author CHARDAN Ana«l\n * \author DAMEY JÃ©rÃ©my\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\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 dÃ©pendance\n    view->setModel(model);\n\n    view->treatGame();\n\n    delete view;\n    delete model;\n\n    return 0;\n}
```

mai 22, 14 20:37

mainSFML.cpp

Page 1/1

```
/***
 * \file mainSFML.cpp
 * \brief Notre main SFML
 * \author CHARDAN Anaïl
 * \author DAMEY Jérôme
 * \date 09/03/2014
 */

#include <iostream>
#include "GameModel.h"
#include "GameViewSFML.h"
#include <cstdlib>

using namespace std;

int main(int argc, const char * argv[])
{
    std::srand ( unsigned ( std::time(0) ) );
    GameModel* model = new GameModel;
    GameView* view = new GameView;

    //Injection de dépendance
    view->setModel(model);

    view->treatGame();

    delete view;
    delete model;

    return 0;
}
```

mai 22, 14 20:37

PuruContext.cpp

Page 1/1

```
/*
 * \file PuruContext.h
 * \brief Notre classe PuruContext
 * \author CHARDAN Ana«l
 * \author DAMEY JÃ©rÃ©my
 * \date 09/03/2014
 */

#include "PuruContext.h"

PuruContext::PuruContext() {
    my_isInPresentation = true;
    my_isInBreak = false;
    my_isChoosingOption = false;
    my_isPlaying = false;
    my_isInAnimation = false;
    my_isOver = false;
    my_isTimeOver = false;
    my_isEnterABestScore = false;
    my_isViewingBestScore = false;
    my_isEnableSound = true;
    my_isEnableMusic = true;
    my_language = francais;
}

/* =====
Les Setteurs
=====*/
void PuruContext::setInBreak( bool set ) { my_isInBreak = set; }
void PuruContext::setInPresentation( bool set ) { my_isInPresentation = set; }
void PuruContext::setChoosingOption( bool set ) { my_isChoosingOption = set; }
void PuruContext::setEnterABestScore( bool set ) { my_isEnterABestScore = set; }
void PuruContext::setPlaying( bool set ) { my_isPlaying = set; }
void PuruContext::setAnimation( bool set ) { my_isInAnimation = set; }
void PuruContext::setOver( bool set ) { my_isOver = set; }
void PuruContext::setTimeOver( bool set ) { my_isTimeOver = set; }
void PuruContext::setViewingBestScore( bool set ) { my_isViewingBestScore = set; }
void PuruContext::setSound( bool set ) { my_isEnableSound = set; }
void PuruContext::setMusic( bool set ) { my_isEnableMusic = set; }
void PuruContext::setLanguage( Language language ) { my_language = language; }

/* =====
Les Guetteurs
=====*/
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; }
```

mai 22, 14 20:37

Score.cpp

Page 1/1

```
/***
 * \file Score.cpp
 * \brief MÃ©thodes liÃ©es Ã notre classe score
 * \author CHARDAN AnaÃ«l
 * \author DAMEY JÃ©rÃ©my
 * \date 09/03/2014
 */

#include "Score.h"
using namespace std;

Score::Score() {
    my_success.resize(1);
}

int
Score::getCurrent() const {
    return my_success[ my_success.size() - 1 ];
}

int
Score::getCurrentStep() const {
    return ( static_cast<int>( my_success.size() ) );
}

int
Score::getGlobale() const {
    int sum = 0;
    /*On prends pas la case en cours.
    for ( int i = 0; i < ( static_cast<int>( my_success.size() ) - 1 ) ; ++i )
        sum += my_success[i];
    return sum;
}

void
Score::addSuccess() {
    my_success.resize( my_success.size() + 1 );
}

void
Score::addPoints( const int &i ) {
    my_success[ my_success.size() - 1 ] += i;
}

void
Score::resetScore() {
    my_success[ my_success.size() - 1 ] = 0;
}
```

mai 22, 14 20:37

Utils.cpp

Page 1/1

```
/***
 * \file Utils.h
 * \brief Notre utilitaire
 * \author CHARDAN Anaël
 * \author DAMEY Jérôme
 * \date 09/03/2014
 */

#include "Utils.h"
#include <iostream>
#include <sstream>
#include <cstdlib>
#include <string>
#include "Constantes.h"
#include <cmath>

using namespace std;

string intToString( int i ) {
    ostringstream oss;
    oss << i;
    return oss.str();
}

std::string
colorMessage( const char* out , int color ) {
    ostringstream o;
#ifndef linux
    o << "\E[" << color << ";1m" << out << "\E[m";
    return o.str();
#else
    o << out;
    return o.str();
#endif
}

int convertIndiceXToPixel( int i ) {
    return CASEWITDH * i + MARGINLEFT + PADDINGRIGHT * i ;
}

int convertIndiceYToPixel ( int j ) {
    return CASEHEIGHT * j + MARGINTOP + PADDINGBOTTOM * j ;
}

int convertXPixel( int xpixel ) {
    if ( xpixel >= MARGINLEFT && xpixel <= ( MARGINLEFT + ( COLONNE * CASEWITDH ) + ( ( COLONNE - 1 ) * PADDINGRIGHT ) )
) {
        return ceil( ( xpixel - MARGINLEFT ) / ( CASEWITDH + PADDINGRIGHT ) );
    }
    return -1;
}

int convertYPixel( int ypixel ) {
    if ( ypixel >= MARGINTOP && ypixel <= ( MARGINTOP + ( LIGNE * CASEHEIGHT ) + ( ( LIGNE - 1 ) * PADDINGBOTTOM ) ) ) {
        return ceil( ( ypixel - MARGINTOP ) / ( CASEHEIGHT + PADDINGBOTTOM ) );
    }
    return -1;
}

int randomNumber( int min , int max ) {
    return min + ( rand() % ( max + 1 - min ) );
}
```