|  |  |
| --- | --- |
| dOSSIER D’ANALYSE  Projet 2, Application C++ | Marc FURLAN  ADAI 2017/2018 |

Table des matières

[1. Introduction 2](#_Toc508364608)

[2. Tableau des fonctionnalités 2](#_Toc508364609)

[3. Schéma Objet (MCD) et schéma relationnel de base de données (MLD) 3](#_Toc508364610)

[4. Dictionnaire des données 3](#_Toc508364611)

[5. Schéma UML cas d’utilisation 4](#_Toc508364612)

[6. Diagramme de classes 5](#_Toc508364613)

# Introduction

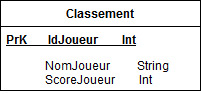
L’application Galactic Clash est un jeu de tir (Shoot ‘em Up, ou Shmup comme on l’abrège aujourd’hui) inspiré des classiques de l’arcade des années 80 et 90 (tels que Galaga, R-Type, voire même le plus ancien Space Invaders). Le design général est également inspiré des jeux de cette époque (pour des raisons esthétiques et de simplification de la mise en œuvre) et le gameplay se veut très simple et dynamique. Le joueur contrôle un vaisseau spatial au travers de plusieurs niveaux très courts et à la difficulté croissante. Le jeu s’achève sur un combat contre un ennemi très puissant (le fameux boss final).

Le code est réalisé en C++ sur l’environnement de développement Microsoft Visual Studio. La librairie graphique SFML est utilisée, ainsi que la base de données SQLite.

# Tableau des fonctionnalités

|  |  |  |
| --- | --- | --- |
| FONTIONNALITES | OBLIGATOIRE | OPTIONNELLE |
| JOUEUR |  |  |
| Déplacement du vaisseau | X |  |
| Tir principal | X |  |
| Tir secondaire plus puissant en nombre limité |  | X |
| Le joueur dispose de plusieurs vies | X |  |
| Déplacement des ennemis (aléatoire ou déterminé) | X |  |
| Tir des ennemis | X |  |
| Plusieurs niveaux | X |  |
| Plusieurs types d’ennemis | X |  |
| Boss final très puissant | X |  |
| Ecran titre | X |  |
| Effets sonores et musique | X |  |
| Gestion du score | X |  |
| Sauvegarde du score en base de données | X |  |
| Les meilleurs score s’affichent à chaque partie | X |  |
| Réglage du niveau de difficulté |  | X |
| Possibilité d’utiliser une manette de jeu ou un joystick |  | X |
| Amélioration du vaisseau/tir du joueur |  | X |
| Intelligence artificielle simple pour les ennemis |  | X |
| Défilement du décor des niveaux (scrolling) |  | X |

# Schéma Objet (MCD) et schéma relationnel de base de données (MLD)



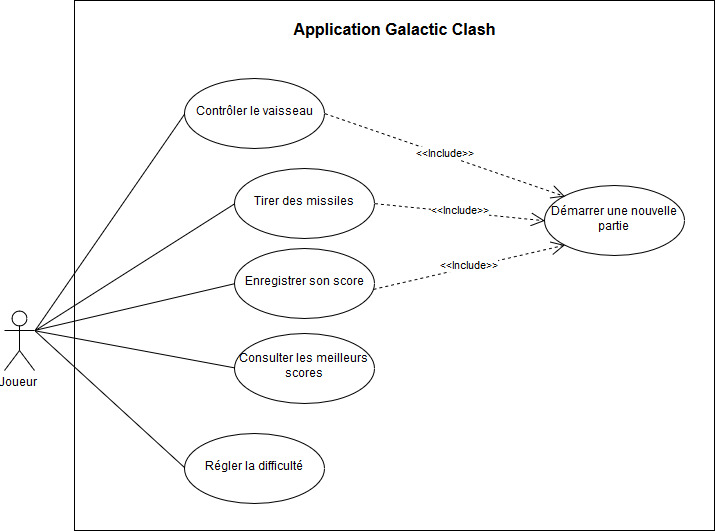
Classement ( **IdJoueur**, NomJoueur, ScoreJoueur)

Pour ce projet, la base de données est volontairement très simple : pour respecter l’esprit des jeux influençant le fonctionnement de mon application, seul compte le score ! Le joueur peut entrer son nom et son score est sauvegardé en base de données. Le classement des 10 meilleurs scores est affiché dans l’application.

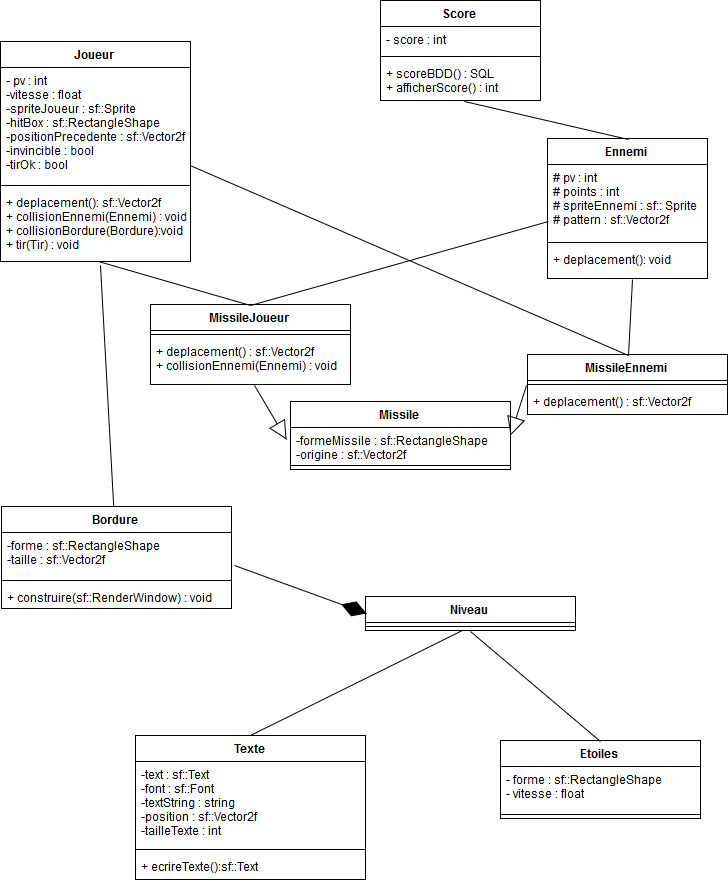
# Dictionnaire des données

|  |  |  |
| --- | --- | --- |
| Libellé | Nom | Type (BDD SQLite) |
| Identifiant du joueur | IdJoueur | Int (auto-incrémenté) |
| Nom du joueur | NomJoueur | Text |
| Score du joueur | ScoreJoueur | Int |

# Schéma UML cas d’utilisation



# Diagramme de classes



Tous les ennemis sont créés à partir de la classe Ennemi.