Table des matières

[1. Présentation 3](#_Toc154058879)

[1.1. Description générale 3](#_Toc154058880)

[1.2. Règles du jeu 3](#_Toc154058881)

[1.3. Cinématique des écrans 3](#_Toc154058882)

[2. Conception – Diagramme de classe 4](#_Toc154058883)

[2.1. Présentation générale 4](#_Toc154058884)

[2.2. Présentation détaillée MainWindow 4](#_Toc154058885)

[2.3. Présentation détaillée autre classe 5](#_Toc154058886)

[3. Partie Algorithmie 5](#_Toc154058887)

[3.1. Détection des collisions tirs joueur sur ennemi 5](#_Toc154058888)

[4. Conception graphique 6](#_Toc154058889)

[5. Cahier de recettes 7](#_Toc154058890)

[5.1 Tests de validation 7](#_Toc154058891)

[5.2 Tests de performance 7](#_Toc154058892)

# Présentation

## Description générale

## 

« Boss Rush » est un jeu du genre des Boss rush, ou l’on doit affronter 5 boss représentant les 5 éléments (Terre, Eau, Feu, Vent, Plante), qui une fois vaincu permet de récupérer chacun un pétale d’une fleur. Chaque pétale obtenu débloquerait une compétence en rapport avec le boss, Ces boss lance des projectiles qui accélère plus on avance dans les niveaux.

**Boss 1 –** (Élément Terre)

**Thème** : Un golem de pierre.

**Compétences après victoire** : Le joueur obtient une compétence lui permettant de renforcer sa prochaine attaque

**Boss 2 –** (Élément Eau)

**Thème** : Une créature aquatique.

**Compétences après victoire** : Le joueur débloque la capacité de créer des boucliers d’eau pour se protéger.

**Boss 3 –** (Élément Feu)

**Thème** : Un dragon de feu.

**Compétences après victoire** : Le joueur acquiert la capacité de lancer des boules de feu.

**Boss 4 –** (Élément Air)

**Thème** : Un monstre volant.

**Compétences après victoire** : Le joueur reçoit la capacité d’avoir des doubles sauts.

**Boss 5 –** (Élément Plante)

**Thème** : Un esprit ancien de la forêt.

**Compétences après victoire** : Le joueur double ses points de vie et régénère sa santé entièrement pour le boss final.

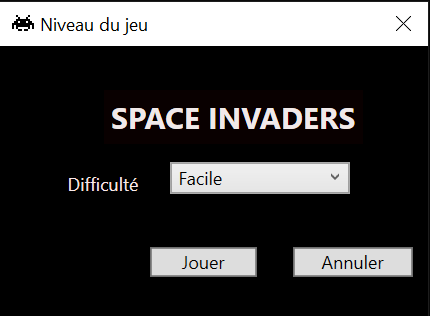
**Le Boss Final – La Fleur de l’Ultime Métamorphose :**

Une fois tous les pétales collectés, la fleur a atteint son plein pouvoir. Le boss final serait une fleur corrompue, maitre des éléments, qui proposera un véritable challenge pour les plus déterminés.

## Règles du jeu

Règle du jeu détaillée ainsi que la description des touches ou autre nécessaire pour jouer (code triche par exemple …)

## Cinématique des écrans



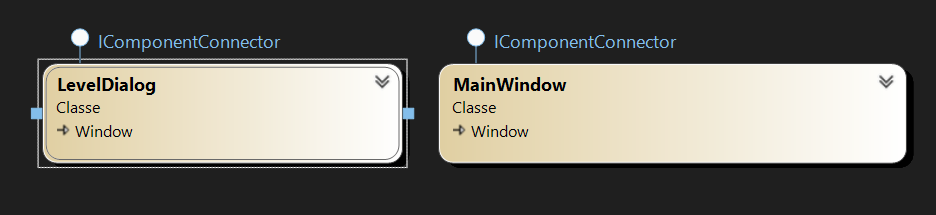


# Conception – Diagramme de classe

Attention : cette partie doit justifier et expliquer la note de l’autoévaluation

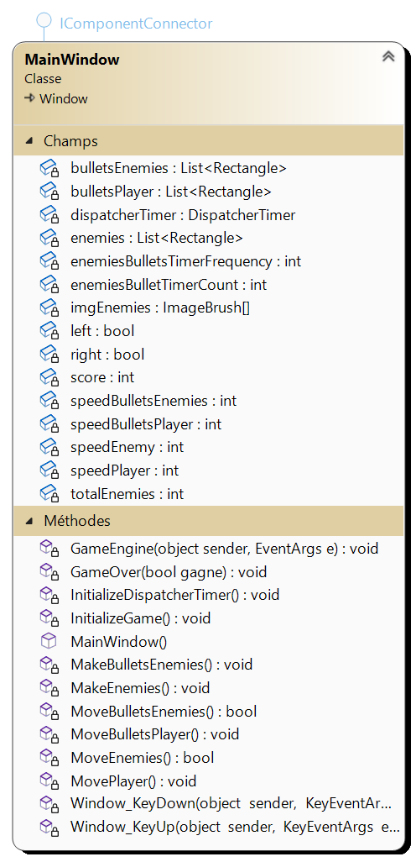
## Présentation générale

Faites une impression écran générale (sans le détail des classes ) accompagnée d’explications synthétiques pour chaque classe. N’hésitez pas à regrouper et découper si schéma trop grand.



Aide : sur votre projet, « Ajoutez un nouvel élément / Diagramme de classes » puis faites glisser vos classes depuis l’explorateur de solution dans la fenêtre.

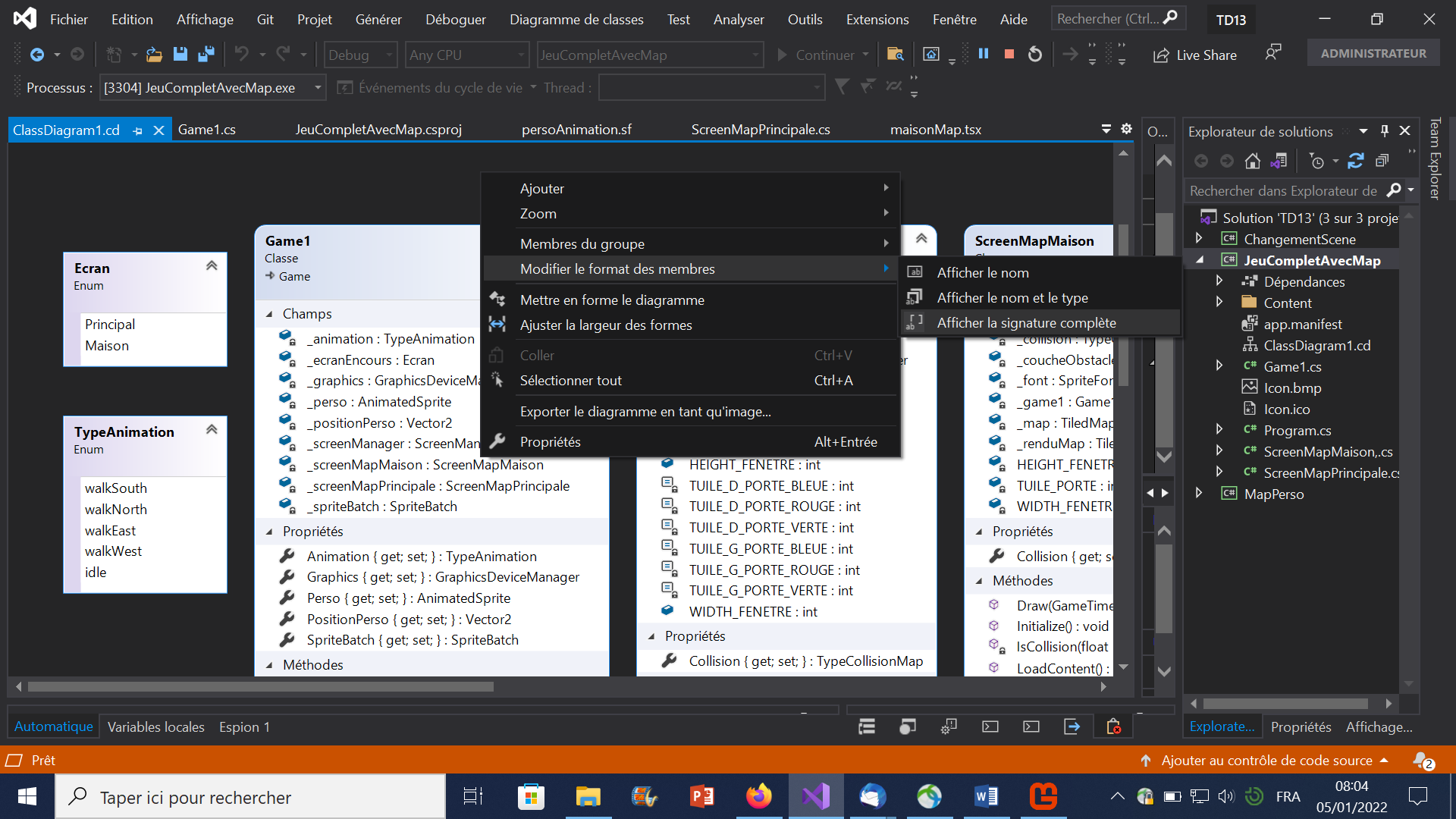
Justifiez vos choix de conception : expliquez et mettez en évidence si vous avez fait des classes pour améliorer, factoriser votre code



## Présentation détaillée MainWindow

Impression écran détaillée de chaque classe avec leur signatures **accompagnées d’explications un peu plus détaillées.**

Aide : clic droit sur le fond de fenêtre vide de votre diagramme de classes puis Menu ci-dessous :



## Présentation détaillée autre classe

….

# Partie Algorithmie

Attention : cette partie doit justifier et expliquer la note de l’autoévaluation

Listez les points d’algo intéressants (leur but et leur logique en pseudo-code voir exemple si dessous) à mettre en valeur pour démontrer la qualité de votre travail ( avec emplacement dans votre projet : Fichier et numéro de ligne ) Mettez en évidence leur complexité et accompagnez les d’affichage de debug pour justifier la validité de vos choix. Démontrez aussi le cout. Une sous partie pour chaque point important.

Remarque : Il est possible d’utiliser des algorithmes existants (ex : A\*) dans ce cas la expliquer la façon dont vous les avez utilisés/codés.

## Détection des collisions tirs joueur sur ennemi

On a décidé de tester tous les tirs avec tous les ennemis :

Pour chaque tir du joueur

Deplacement du tir

S’il dépasse de la zone de jeu, on prévoit de le supprimer

sinon

Pour chaque ennemi

si collision ennemi/joueur

on prévoit de ennemi et joueur

on marque un point

Finsi

Fin Pour

Finsinon

Fin pour

Dans MainWindow.xaml.cs : méthode MoveBulletsPlayer (ligne 243)

On peut compter 30 tests (au maximum : 1 par ennemi) pour chaque tir, à raison de 10 tirs max à gérer en simultané, on peut estimer à 300 tests toutes les 16 ms, soit 18 600 tests par seconde.

# Conception graphique

Indiquez si vos images, animations, sons sont des sources existantes, dans ce cas donnez leur provenance. Expliquez les retouches ou création que vous avez peut-être réalisé.

# Cahier de recettes

## 5.1 Tests de validation

Faites un listing des fonctionnalités , le nom de l’étudiant qui a travaillé dessus et son état :

|  |  |  |
| --- | --- | --- |
| Nom | Fonctionnalités/ Dialogues/Classes | Etat |
| Gruson | LevelDialog | Non achevé/ Bug/ OK |
| Diard | MainWindow |  |
| Gruson | Déplacement et tir joueur |  |
| Diard | Déplacements et tirs ennemis |  |

## 5.2 Tests de performance

A l’aide des outils de diagnostiques : vous prendrez quelques mesures (captures écrans) à des moments clefs de votre jeu de l’utilisation de la mémoire et du processeur. Vous commenterez bien évidemment les moments choisis et les mesures.

