



Conception et Programmation Objet

Projet long : Bomber7



À l'attention de : M. Xavier Crégut, Mme. Aurélie Hurault, Étudiants de FISA L3 SN

Team 4 : Thomas SILVESTRE - Nikita ZIUZIN - Stephane LOPPINET - Yoann FRANCOIS - Corentin PRADIER - Pierre CHAVEROUX

Sommaire

Partie 1 : Démonstration

Partie 2 : Présentation technique

Partie 3 : Gestion de projet

Partie 4 : Les perles



Partie 1 : Démonstration



Player 1

2 ❤️ 1 🌍 1

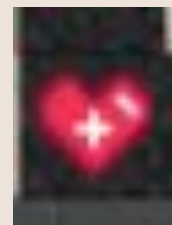
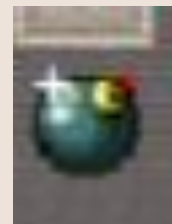
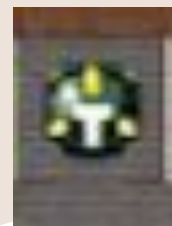
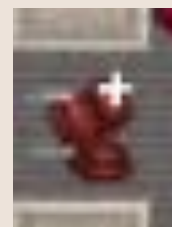
Player 2

6 ❤️ 1 🌍 1

Eleve 1

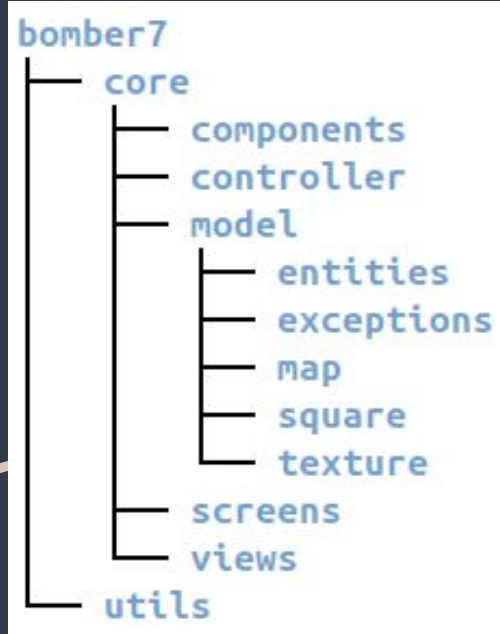
Prof

Demo 2



Partie 2 : Présentation Technique

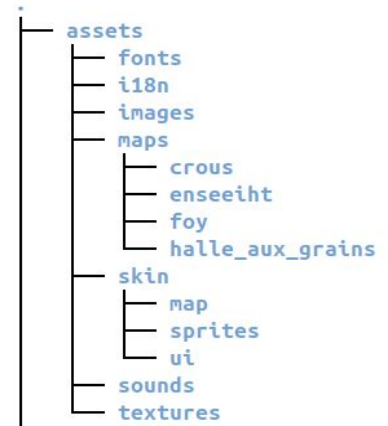
A. Packages



Les différents paquetages de notre application :

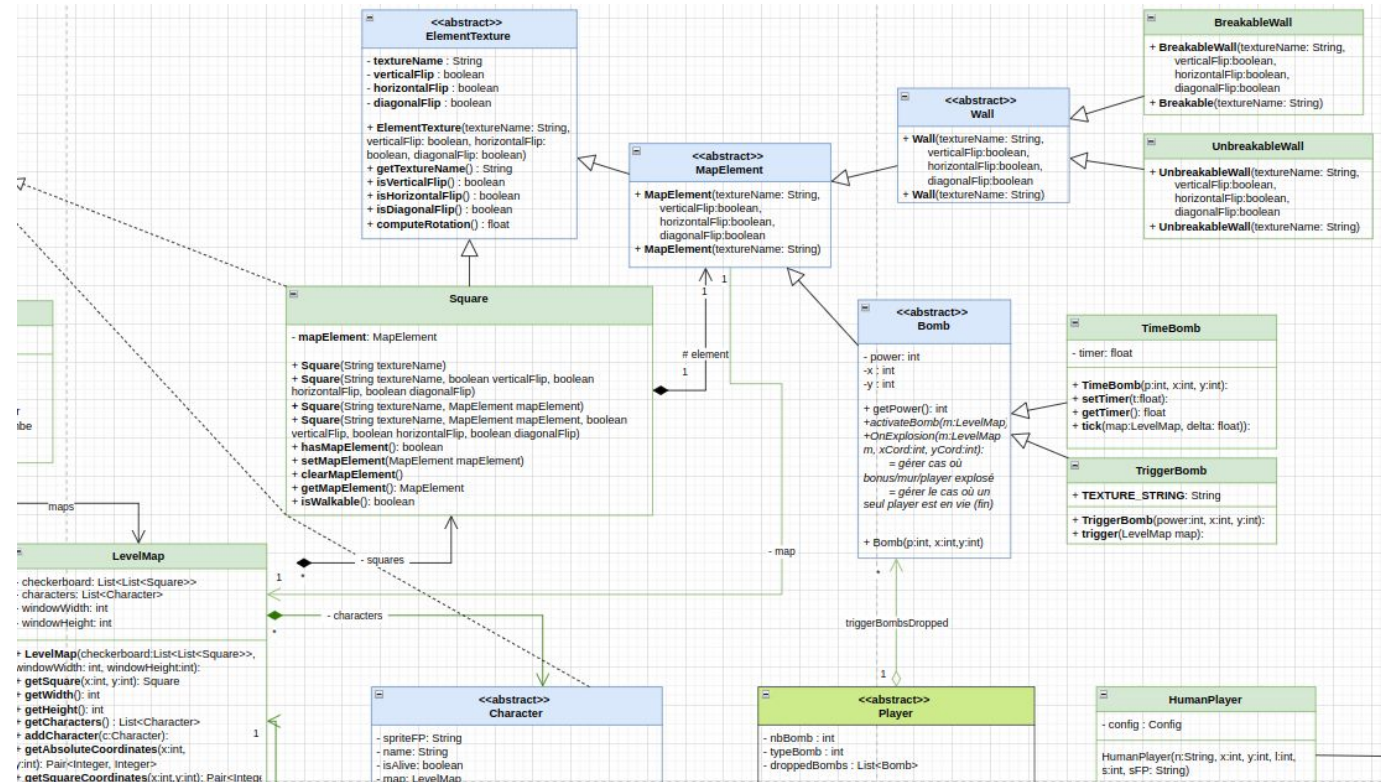
- components : éléments graphiques personnalisés
- controller : configuration des touches du joueur
- model : la logique de notre application
 - entities : les entités du jeu → Joueurs, IA, monstres, etc.
 - exceptions : règles sur les setters (vie, vitesse, ...)
 - map : création des cartes
 - square : éléments relatifs aux cases de la map
 - texture : manipulation des textures
- screens : écrans du jeu, visibles à l'utilisateur
- views : sous-mécanismes d'affichages

Les ressources statiques du jeu :



Partie 2 : Présentation Technique

B. Architecture UML : Modèle

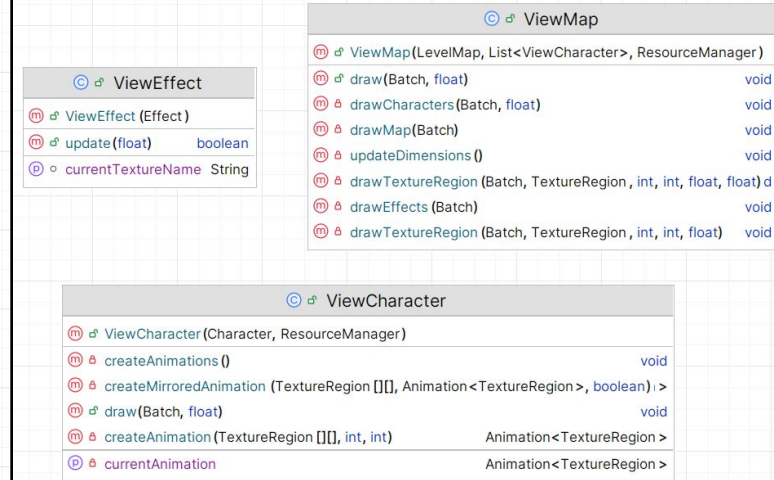
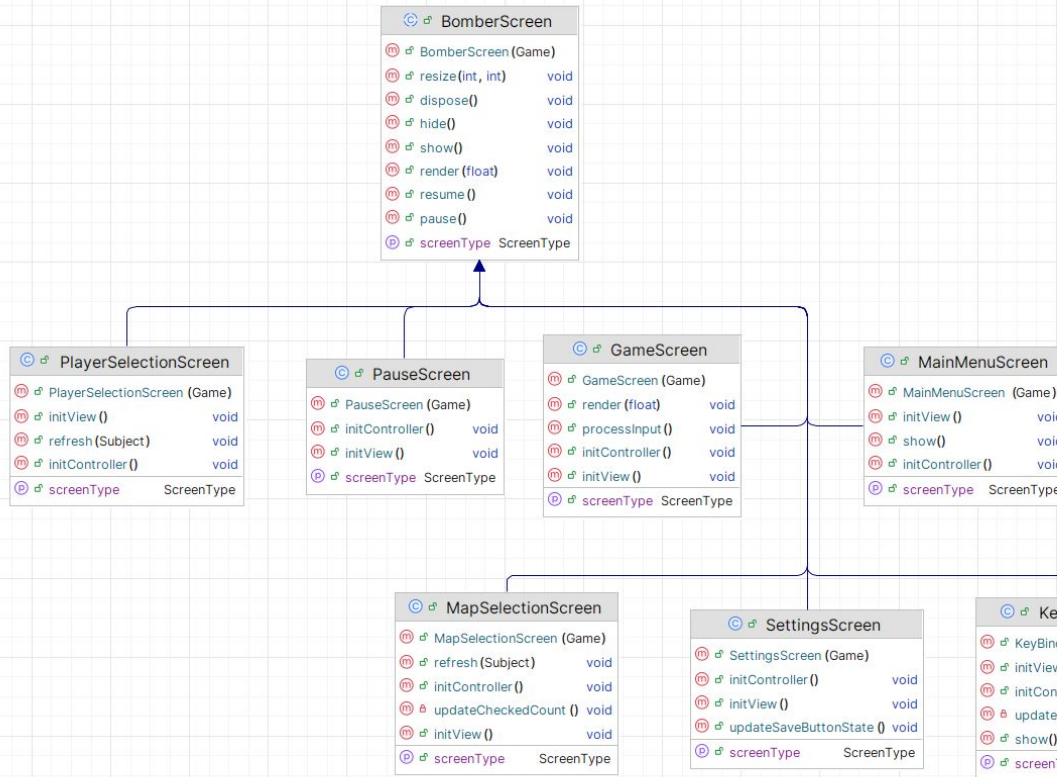


Les classes importantes :

- Character : Player, ...
- Wall : (un)breakable
- Bomb : Time, Trigger
- Bonus : Life, Speed, ..

Partie 2 : Présentation Technique

B. Architecture UML : Vue

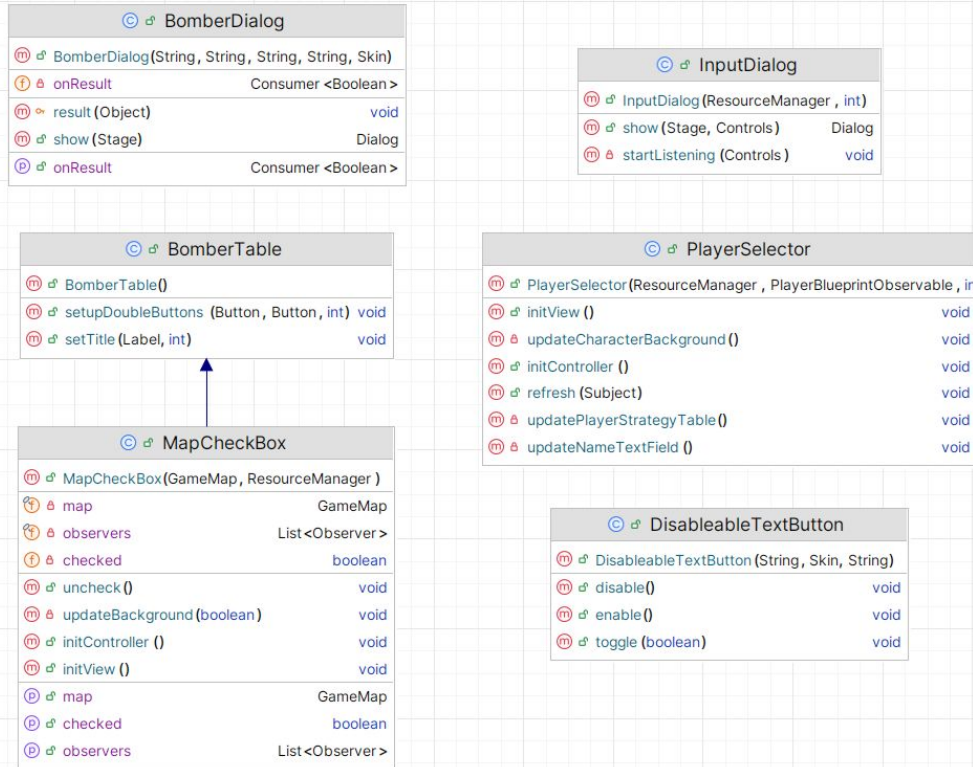


Les classes importantes :

- BomberScreen, Screen
- ViewCharacter, ViewMap, ViewEffect,

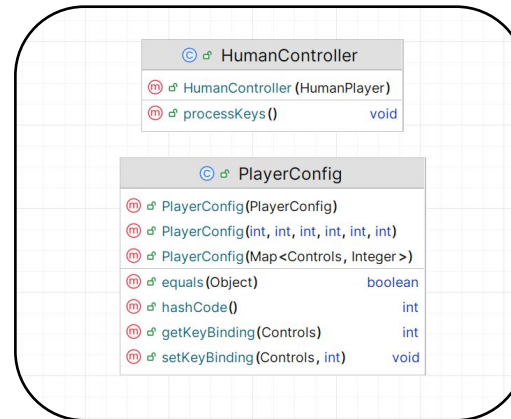
Partie 2 : Présentation Technique

B. Architecture UML : Contrôleurs



Deux types de contrôleurs

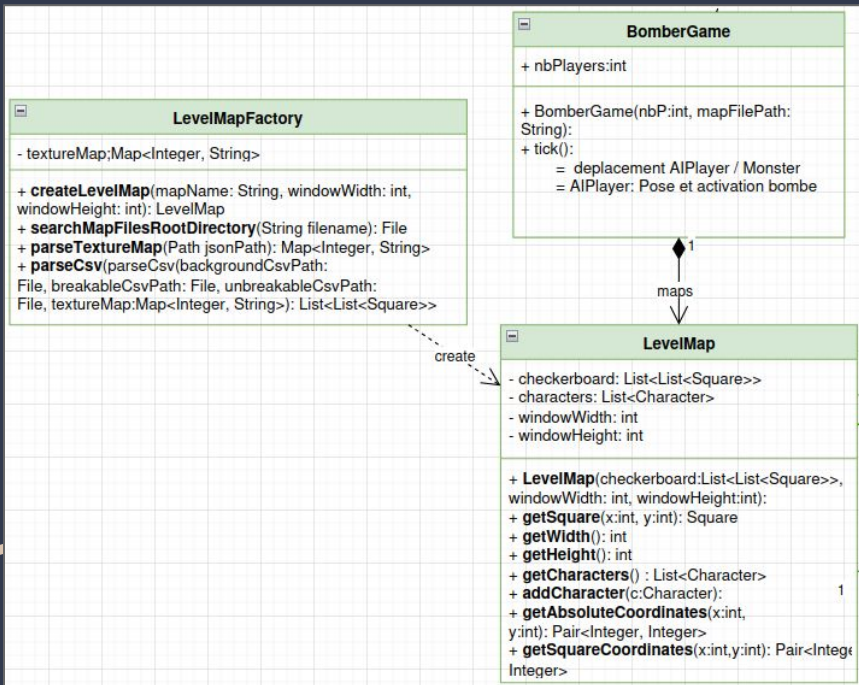
- Contrôleurs pour les “bindings” de touches
- Contrôleurs pour les éléments graphiques



Partie 2 :

Présentation Technique

B. Choix de conception



Les patrons de conception :

- **MVC** : HumanPlayer, ViewCharacter, HumanContr.
- Modèle-Vue-Contrôleur
- **Observateur** : Components uniques (PlayerSelector - GC)
- Vues interrogent le modèle sur quel élément afficher
- **Singleton** : Ressources, SoundManager, ...
- Une seule instance et fournit un point d'accès global
- **Fabrique** : LevelMapFactory
- Interface pour créer des objets : ici les levelMap
- **Stratégie** : AIDropBomb → AIDumb, AISmart
- Méthode commune, comportements différents

Partie 2 : Présentation Technique

C. Choix techniques



Les outils utilisés :

- **LibGDX** : Framework Open Source Java

→ Facilitation du rendu 2D, de la gestion de l'horloge, de la musique et des effets graphiques (sprites)

- **Gradle** : Gestion des dépendances

→ Téléchargement automatique des bibliothèques Java depuis des dépôts reconnus : Maven Central.

→ Permet d'exécuter les tests et vérifier les erreurs de checkstyle.

- **GitHub Pipeline CI/CD** :

→ WorkFlows avec vérification Checkstyle

→ Run des tests pour le maintien de la qualité du code

→ Protection de la main (Merge Request Reviews, ...)

Partie 3 : Gestion de projet

A. Méthodes agiles



The Scrum Guide

Ken Schwaber & Jeff Sutherland

The Definitive Guide to Scrum: The Rules of the Game

The Definitive Guide to Scrum: The Rules of the Game

Outils :

- GitHub Projects :

→ Gestion des sprints, des ressources, des missions

→ Graphiques utiles pour le post-mortem du projet

- 4 sprints aux missions dédiées :

→ Sprint 1 : Architecture applicative - UML - Pipeline

→ Sprint 2 : Classes principales du modèle + tests

→ Sprint 3 : Vues (map, joueur, bombe) et contrôleurs

→ Sprint 4 : Merge en main et interfaçage final

- Palliatif aux cours de GDP :

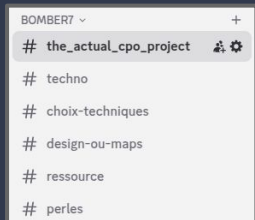
"The Scrum Guide", Ken Schwaber & Jeff Sutherland

→ Edition 2020,

→ <https://www.scrum.org/resources/scrum-guide>

Partie 3 : Gestion de projet

B. Outils GDP



Médias de communication :

- Présentiel :

→ Travail en classe sur les créneaux en autonomie

→ Mise en commun des connaissances sur tableau

- Distanciel :

→ Discord : partages des ressources, idées ...

Outils de gestion de projet :

- GitHub Projects :

→ Visualisation des sprints, répartition des tâches

→ Suivi du chemin critique du projet

→ Mise en commun des avancés sur le code : PR reviews validées après minimum trois validations

→ <https://github.com/users/Stephane-Lpt/projects/5>

Partie 4 : Les perles



Modify
the checkstyle
to accept
"case 3 :"



"case
2 + 1 :"



imgflip.com

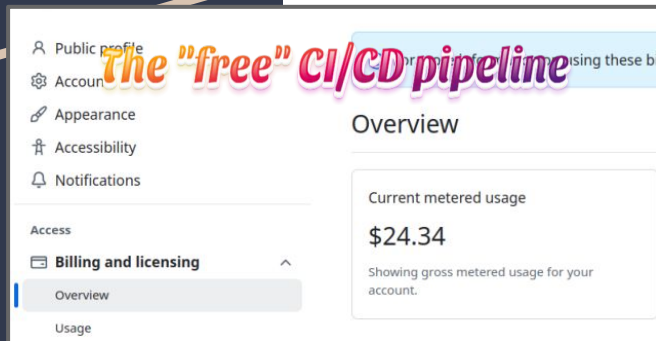
JAKE-CLARK.TUMBLR

**PIPELINE THE
CHECKSTYLE IN
GITHUB SO YOU
DOUBLE-CHECK IT**

**USE GRADLE
TO TO TEST THE
CHECKSTYLE LOCALLY**

**CHECKSTYLE
IS INSTINCT.
JUST USE MEANINGFUL
VARIABLE NAME**

**INT
TRENTETROIS = 33;
//MAGICNUMBER**





Merci pour votre écoute !
Place aux questions



Sources :

Logo Signal : [https://en.wikipedia.org/wiki/Signal %28software%29](https://en.wikipedia.org/wiki/Signal_%28software%29)

Logo Discord : <https://www.logiciels.pro/logiciel-saas/discord/>

The Scrum Guide : <https://www.scrum.org/resources/scrum-guide>

Mooc GDP : <https://gestiondeprojet.pm/>

MemeGenerator : <https://imgflip.com/>

LibGDX : <https://seeklogo.com/vector-logo/402393/libgdx>

GitHub Pipeline Logo : <https://medium.com/@blogs4devs/>

Gradle Logo : <https://cdnlogo.com/logo/gradle 43471.html>



BACKUP

Présentation Technique

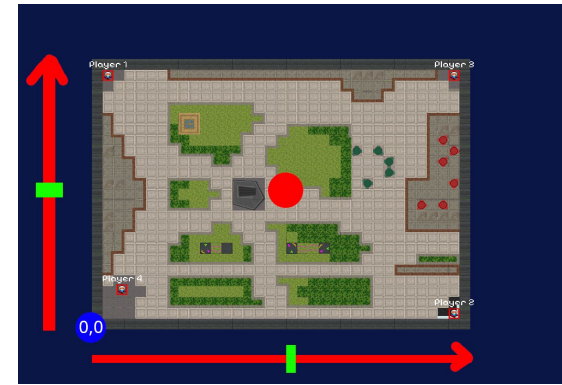
Map

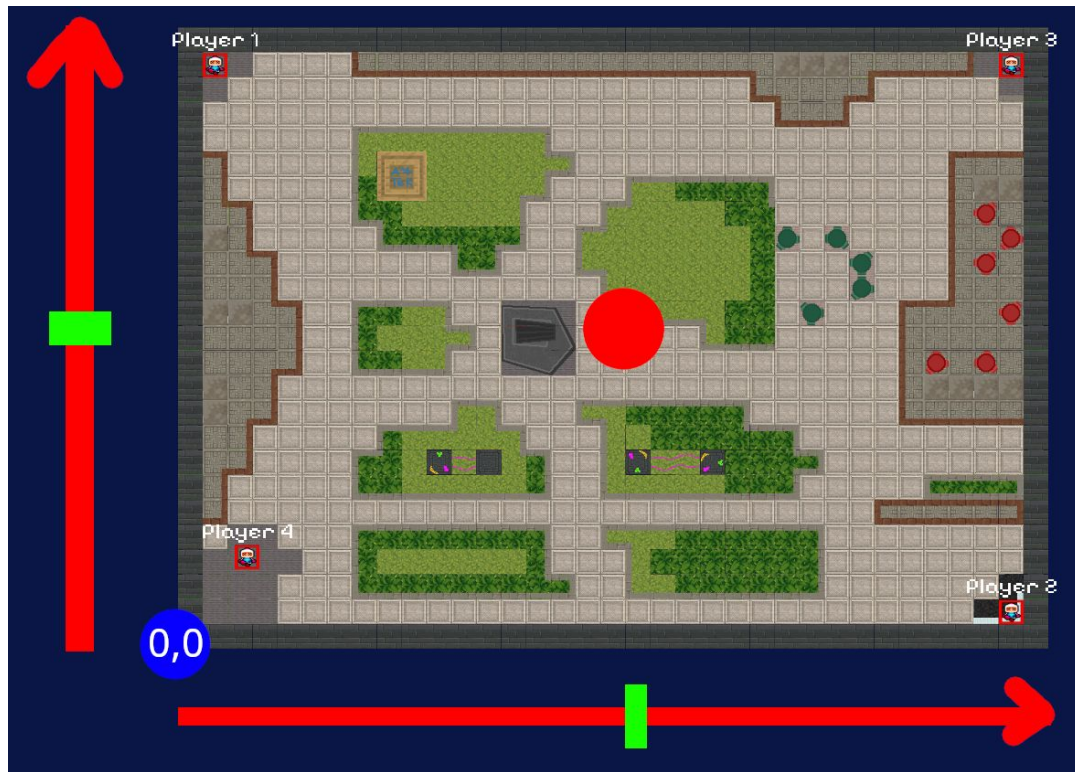
Dessiner les éléments :

- Gérer logique affichage
- Problématique : Vitesse en pixels
 - Sur quelle case suis-je ?

Collision :

- `+ getAbsoluteCoordinates(int x, int y):`
`Pair<Integer, Integer>`
- `+ getSquareCoordinates(int x, int y):`
`Pair<Integer, Integer>`





Partie 2 : Présentation Technique

D. UMLv1 VS UMLv2

V1:

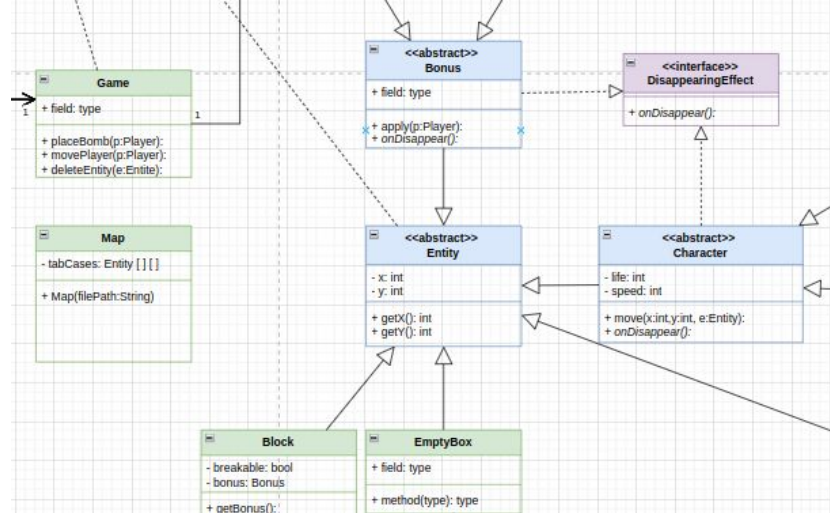
1. Tout hérite de Entity.
2. Map contient une liste d'entités

Problème :

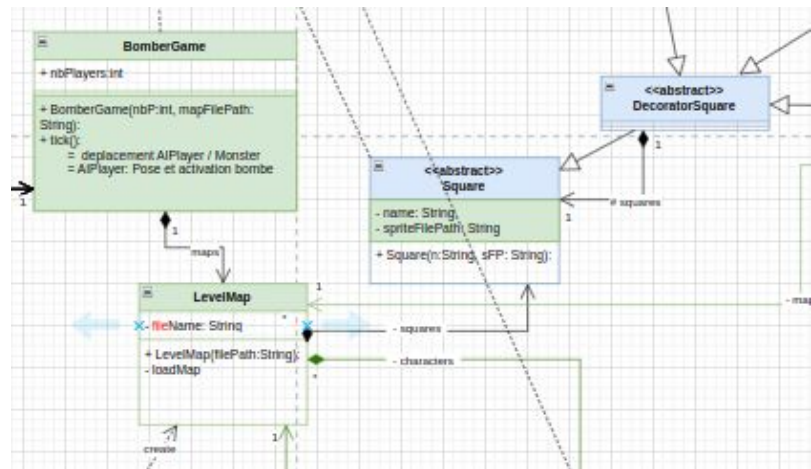
1. Manque de hiérarchie.
2. Déplacement du personnage = Le rechercher dans tout le tableau

Solution : Distinguer les acteurs principaux du reste

V1



V2



Partie 2 : Présentation Technique

D. UMLv2 VS UMLv3

V2:

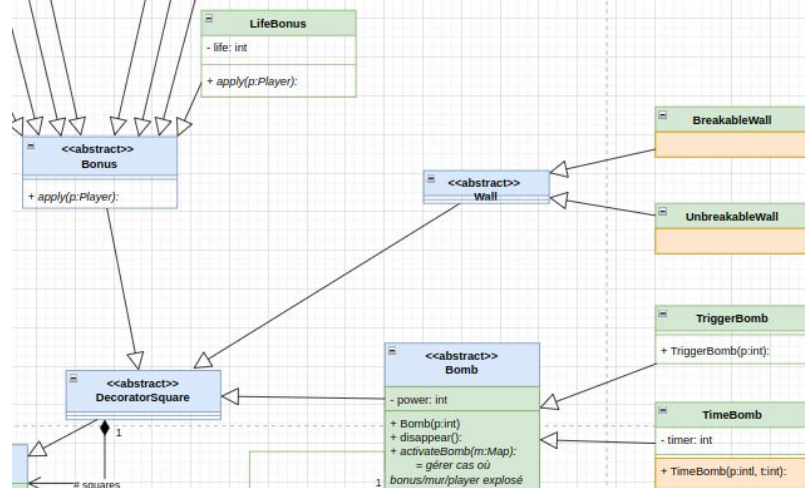
1. Décorateur = plusieurs éléments qui peuvent s'accumuler sur une case

Problème :

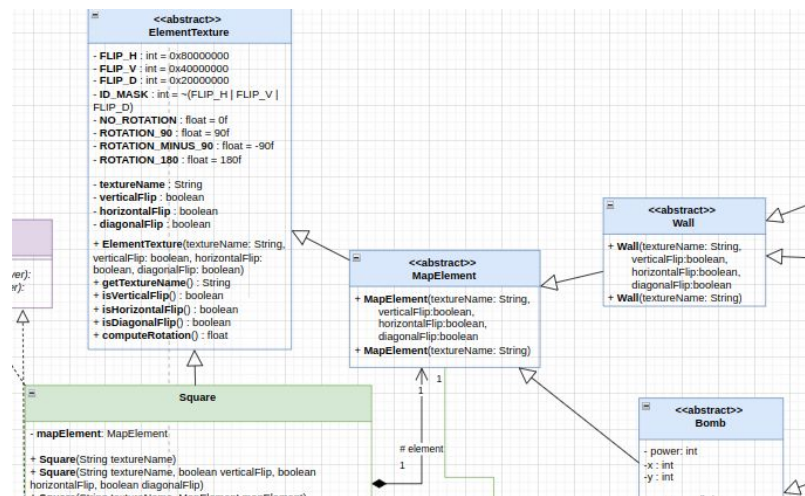
1. Il n'y aura jamais deux éléments au même endroit (soit un mur, soit un bonus, ...).

Solution : Classe abstraite MapElement

V2



V3



Partie 3 : Gestion de projet

A. Méthodes agiles

May 2025

14 15 16 17 18 19 20 21 +2 23 24 25 26 27 28 29

11 ✓ Implement getAbsoluteCoordinates a... #77

12 ✓ Controller classes development #11

13 ✓ Map class development #7

14 ✓ Character vue class development #47

15 ✓ Character class development #6

16 ✓ Bill of specifications #16

17 ✓ Interfaces design #18

18 ✓ Player classes development #43

Controller classes development #11 🤖

Character class development #6 🤖

Bill of specifications #16 🤖

Interfaces design #18 🤖

Player classes development #43

Completed Sprint 1
1 week May 22 - May 28

Completed Sprint 2
1 week May 29 - Jun 04

Completed Sprint 3
1 week Jun 05 - Jun 11

Current Sprint 4
1 week Jun 12 - Jun 18

1 Active 3 Completed

Project Management (Sprints)

Sprint Next iteration Prioritized backlog Roadmap In review My items + New view

Filter by keyword or by field

In progress 0
This is actively being worked on

bomber7 #98
UserManual Redaction

bomber7 #99
Rapport redaction

bomber7 #101
Diaporama for oral presentation

+ Add item

Ready 0
This is ready to be picked up

bomber7 #72
ScoreBoard development

+ Add item

In review 0
This item is in review

bomber7 #23
Bonus classes development

bomber7 #100
Fixing AI Player implementation

bomber7 #13
Find sprites for each character, wall...

bomber7 #97
Final merge from all branch to main

+ Add item

Done 0
This has been completed

bomber7 #2
Class diagram V1 creation : complete

bomber7 #32
Pipeline Junit tests

bomber7 #12
Design game maps

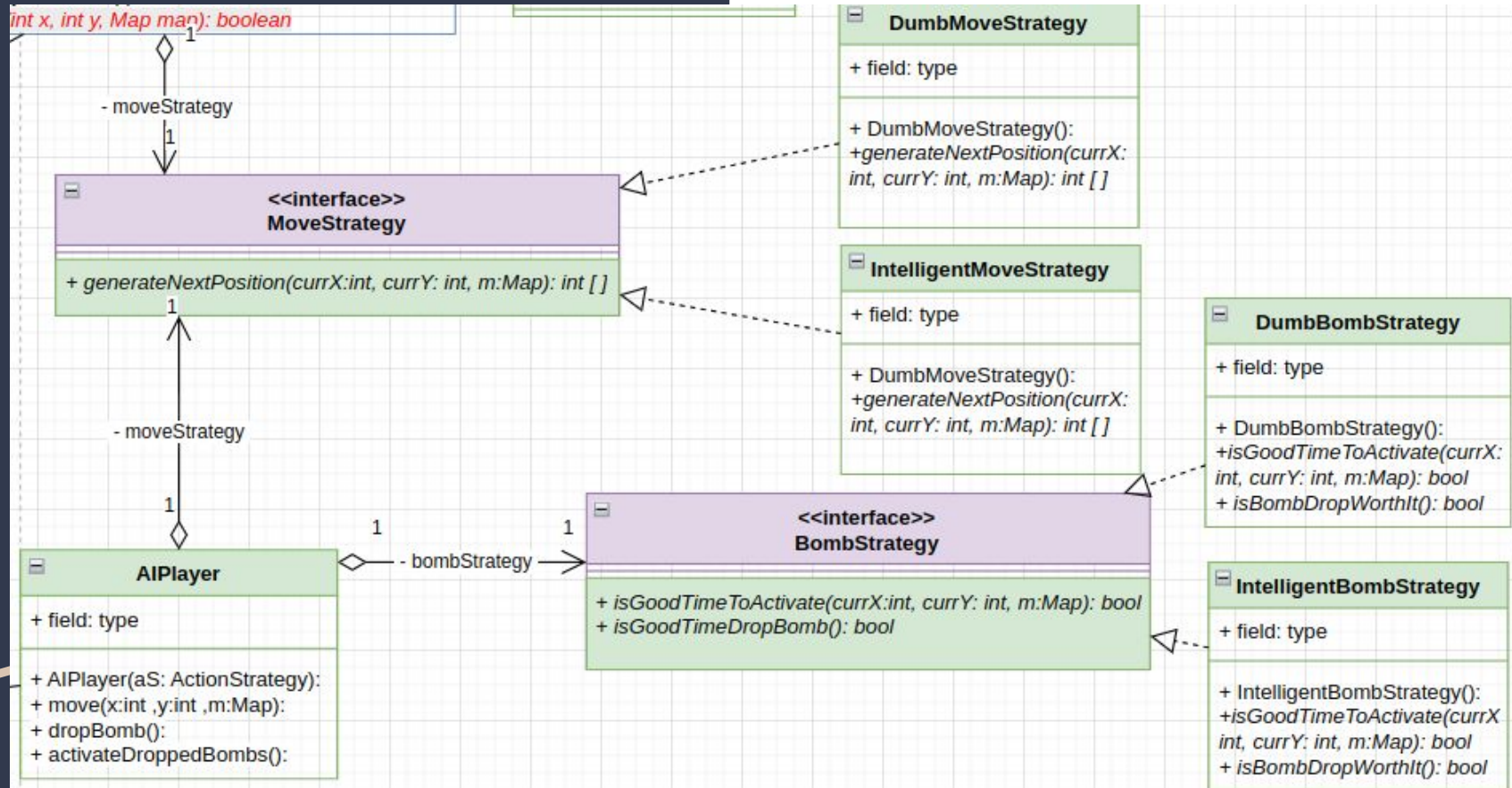
bomber7 #74
Complete DroppedBombMethod in Character

bomber7 #3
Description of features

bomber7 #77

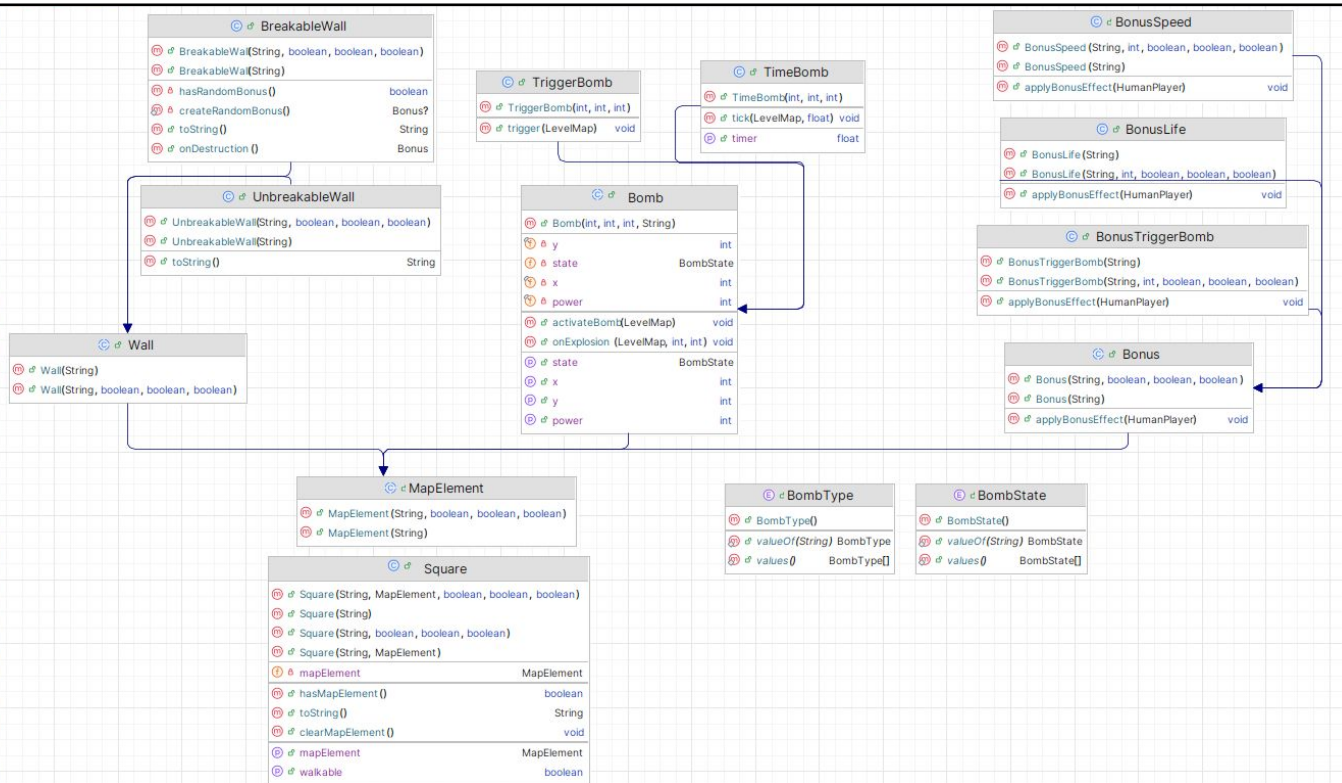
+ Add item

Patron Stratégie :



Partie 2 : Présentation Technique

B. Architecture UML : Modèle



Les classes importantes :

- Character : Player, ...
- Wall : (un)breakable
- Bomb : Time, Trigger
- Bonus : Life, Speed, ..

