RAPPORT PROJET LONG
PREMIERE ITERATION

LOUDYI Ayoub
LAANAIYA Mahmoud
LOTFI Mohamed Hamza
JAAFARI Amine
IKICH Mohamed
LAAOUINA Chouaib
JANAH Akram
Groupe IJ-5

Département Sciences du Numérique - Première année
2020-2021

1 Introduction

Ce document représente le rapport de la première itération du projet long réalisé par le groupe IJ-5. Il comprend un rappel du sujet abordé, des explications et des élaborations relatives aux différentes implantations et réalisations mises en oeuvre au cours du projet. Sans oublier bien sûr un petit aperçu sur l'organisation de l'équipe ainsi que les éléments élaborés suite aux séances gestion de projet.

2 Rappel

L'idée du projet est de créer un clone du jeu "Bomberman" en 2D. Ce jeu dans lequel un joueur se déplace et pose des bombes sur une ¨map¨ pour détruire des murs et vaincre des ennemis, qui à leurs tours vont ¨drop¨ des bonus pour le joueur. Les bombes peuvent également infliger des dégâts au joueur si il se trouve dans leur zone.
Le clone sera le plus fidèle possible au gameplay du vrai ¨Bomberman¨, avec une musique de fond (BGM), des effets sonores pour l'explosion des bombes et autres interactions (SFX), il sera doté d'un écran de démarrage doté d'un bouton de sélection de partie (Mode Solo, ou Multijoueur), un bouton ¨Réglages¨ pour ajuster le niveau de son, et réaffecter le commandes de jeu, et un bouton ¨Quitter¨ pour quitter l'interface.

3 Packages, Classes et Interfaces

Plusieurs packages ont été définis dans l'implantation :

3.1 Le package par défaut (default package)

Il contient les classes suivantes :

Launcher.java : La classe principale qui lance la partie, elle contient la méthode main et met toutes les classes suivantes en lien.

3.2 Controller

C'est un package qui contient des classes qui relèvent du traitement des touches saisies par l'utilisateur et leur mise en exécution :

Controller.java : Interface qui contient les méthodes relatives aux touches pressées par l'utilisateur.

Input.java : Implémente l'interface KeyListener des actions de l'utilisateur.

PlayerController.java : Classe qui renvoie les entrées clavier de l'utilisateur, selon sa saisie dans Input.java.

3.3 affichage

C'est le paquet regroupant les classes qui affichent le jeu sur l'écran :

Fenetre.java : La classe qui contient le canvas sur lequel on va dessiner.

Renderer.java : La classe qui dessine l'état du jeu à chaque mise à jour.

3.4 gameObjects

Ce paquet est un regroupement de tout ce que le joueur pourra intéragir avec, et il contient les classes :

GameObject.java : Classe abstraite des caractéristiques des objets du jeu

MovingObject.java : Hérite de GameObject, elle est dédiée aux objets pouvant se déplacer sur la carte

Player.java : Hérite de MovingObject, elle est dédiée au joueur principal manipulé par l'utilisateur.

Block.java : Interface spécialisant les blocs de la carte.

BreakableBlock.java : Implémente Block, dédiée aux blocs destructibles de la carte

UnbreakableBlock.java : Implémente Block, dédiée aux blocs non destructibles de la carte.

3.5 game

Ce paquet contient deux classes :

GameLoop.java : La classe responsable du déroulement de la partie, à chaque mise à jour.

Game.java : La classe qui configure le début de la partie. (Taille de la fenêtre, plus tard : le niveau du jeu, ...)

3.6 map

Ce paquet contient tout ce qui est en lien avec la carte du jeu :

GameMap.java : Classe de la carte du jeu.

Tile.java : Classe des cases/tuiles constituant la carte du jeu.

3.7 Menu

Ce paquet contient les classes relevant de l'interface utilisateur :

3.7.1 Menu de départ

Le menu de départ est traité dans **Launcher.java** et dans **JImagePanel.java**. Il y a le boutton Jouer, Options et Quitter. Une fois qu'on clique sur Jouer, la fenêtre du menu de départ se ferme, et à sa place s'ouvre la fenêtre du jeu.

3.7.2 Menu à l'interieur du jeu

Une fois le jeu lancé, on voit trois menus/boutons apparaître en haut de la fenêtre : Partie, Options, Aide. Ils sont traités dans quatres classes, et chaque classe contient une sous classe nommée **MenuActionListener** :
Partie.java

Options.java
Aide.java
Menu.java

3.8 physique

C'est le paquet responsable de tout ce qui est mouvement :

Size.java : Classe définissant la taille d'un objet

VectorDE.java : Classe du vecteur déplacement élémentaire d'un objet

Position.java : Classe de la position cartésienne d'un objet

Mouvement.java : Classe du mouvement de l'objet. Toutes les entrées de l'utilisateur concernant le mouvement sont traitées ici.

4 Permiers résultats et avancement

NB : Tout ceci n'est qu'une première représentation des éléments élaborés. De meilleures versions seront implantées lors des prochaines itérations.

5 Organisation de l'équipe

Pour la répartition des tâches, notre équipe a été divisée en trois groupes :

Un groupe responsable du Moteur de Jeu LOTFI Mohamed Hamza, IKICH Mohamed, LOUDYI Ayoub.

Un groupe responsable des Menus : UI LAANAIYA Mahmoud, JAAFARI Amine.

Un groupe responsable du coté audio-visuel : VFX LAAOUINA Chouaib, JANAH Akram.

6 Difficultés rencontrées

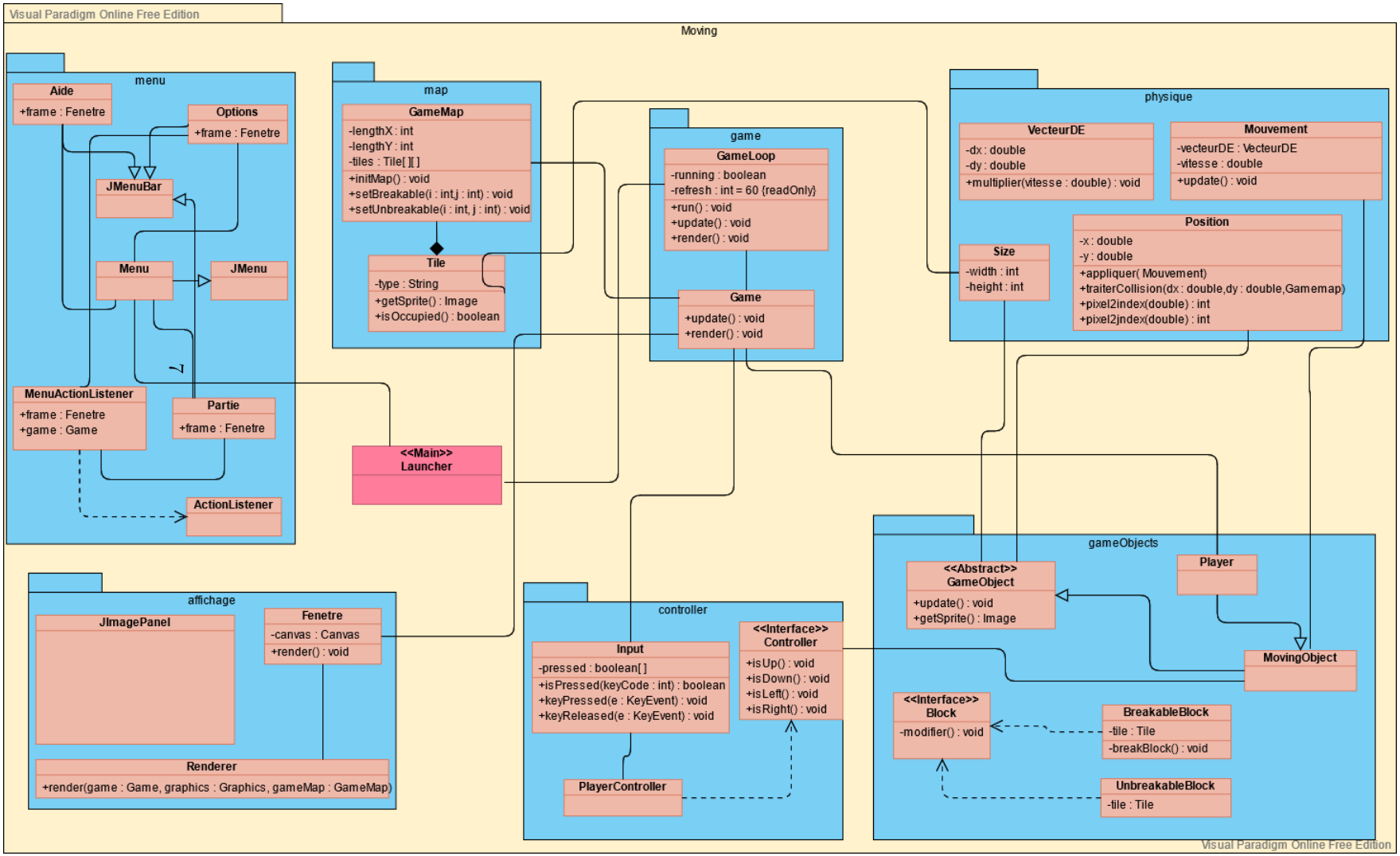
La première difficulté classique qu'on avait rencontrée a été bien sûr la conception des classes et des interfaces à utiliser. Mais nous avions réussi finalement après plusieurs essais à implanter un diagramme UML (première version). Une deuxième difficulté résidait dans le déplacement et les collisions entre le joueur et les blocs. En effet la position du joueur était repérée en pixels, mais il fallait toujours faire attention à sa position relativement à l'indice (i,j) en entier des tuiles sur la carte. Plus précisement, au niveau des collisions nous avons cru au début que la position en pixels correspondait auc centre du carré représentant le Joueur, mais en réalité elle représentait le sommet en haut à gauche du carré. Prenant ceci en considération nous avions orienté les collisions vers un détecteur de dépassement des quatre sommets du carré, ainsi dès que l'un des sommets après un déplacement chevauche une structure à laquelle ils n'ont pas accès le déplacement est annulé. Les autres difficultés relevaient essentiellement des Menus. Dans le menu de départ, on avait des difficultés pour afficher une image dans le background.
Enfin, dans le menu à l'interieur du jeu, on avait une difficulté pour faire fonctionner les boutons puisqu'on avait les classes Fenetre.java et Game.java séparées, et il fallait toujours les lier dans chaque traitement.

7 Travail qui reste à faire :

1. Implémentation de la bombe et ses interactions.
2. Simplification de l'accès à GameMap et des autres niveaux.
3. Implémentation des autres fonctionnalités de L'UI (pause, nouvelle partie, ...).
4. Mise en place des animations de sprites et musique de fond sans latence.

8 Annexe :

8.1 Diagramme UML :


Visual Paradigm Online Free Edition
Moving
menu
Aide
+frame : Fenetre
Options
+frame : Fenetre
JMenuBar
Menu
JMenu
MenuActionListener
+frame : Fenetre
+game : Game
Partie
+frame : Fenetre
ActionListener
map
GameMap
-lengthX : int
-lengthY : int
-tiles : Tile[ ][ ]
+initMap() : void
+setBreakable(i : int,j : int) : void
+setUnbreakable(i : int, j : int) : void
Tile
-type : String
+getSprite() : Image
+isOccupied() : boolean
game
GameLoop
-running : boolean
-refresh : int = 60 {readOnly}
+run() : void
+update() : void
+render() : void
Game
+update() : void
+render() : void
physique
VecteurDE
-dx : double
-dy : double
+multiplier(vitesse : double) : void
Mouvement
-vecteurDE : VecteurDE
-vitesse : double
+update() : void
Position
-x : double
-y : double
+appliquer(Mouvement)
+traiterCollision(dx : double,dy : double,Gamemap)
+pixel2index(double) : int
+pixel2jndex(double) : int
Size
-width : int
-height : int
<<Main>>
Launcher
affichage
JImagePanel
Fenetre
-canvas : Canvas
+render() : void
Renderer
+render(game : Game, graphics : Graphics, gameMap : GameMap)
controller
Input
-pressed : boolean[ ]
+isPressed(keyCode : int) : boolean
+keyPressed(e : KeyEvent) : void
+keyReleased(e : KeyEvent) : void
<<Interface>>
Controller
+isUp() : void
+isDown() : void
+isLeft() : void
+isRight() : void
PlayerController
gameObjects
<<Abstract>>
GameObject
+update() : void
+getSprite() : Image
Player
MovingObject
<<Interface>>
Block
-modifier() : void
BreakableBlock
-tile : Tile
-breakBlock() : void
UnbreakableBlock
-tile : Tile
Visual Paradigm Online Free Edition