hepia

Plants vs Zombies

POO Java

Dos Reis Cédric
14/01/2018

Table des matières

Rapport de développement

Introduction

Plants vs Zombie est un jeu. Le but est de protégé sa maison d'une invasion de zombies qui veulent manger votre cervelle. Pour cela, vous devez planter des plantes dans votre jardin afin qu'elles puissent défendre votre maison de l'attaque des zombies. Chaque plante à son propre but : attaquer, ralentir ou supporter.

Analyse fonctionnelle

Le joueur doit pouvoir :

- Choisir la plante qu'il veut planter
- Choisir ou planter sa plante
- Déterrer une plante

Le programme s'occupe de :

- Afficher tous les objets sur le terrain de jeu
- Générer différents zombies
- Déplacer les zombies en direction de la maison

Analyse Organique

Positionnement des plantes

Le jardin fonctionne comme une grille composée de 5 lignes et 5 colonnes. Le joueur doit cliquer avec la souris sur l'emplacement où il souhaite planter la plante. À partir de là, le programme se charge de trouver la case de la grille la plus proche de l'emplacement du clique.

Génération de zombies

La génération des zombies est gérée par un *Thread*. Les zombies commencent à apparaitre en dehors du jardin dès que la première plante a été plantée. Il y a trois différents zombies dont chacun à sa propre apparence et ses propres points de vie. Les zombies sont générés en fonction d'un temps définit. C'est-à-dire qu'il y a un temps qui doit s'écouler entre la génération de chaque zombie. Ils apparaissent à une hauteur définie choisi aléatoirement. Lorsqu'un zombie est généré, il commence à se déplacer vers la maison.

Gestion de l'affichage

Les plantes, leurs projectiles ainsi que les zombies sont tous dérivé de la mémé classe : *Element*. Cet objet contient la méthode *paintComponent(Graphics g)* qui sert à dessiner l'image de l'objet dans le fenêtre. Cette méthode peut être redéfinie pour afficher plus de choses.

Mise à jour, Déplacement des objets

Les plantes, leurs projectiles ainsi que les zombies sont tous dérivé de la mémé classe : *Element*. Cet objet contient la méthode *Update(long elapsedTime)* qui sert à mettre à jour l'emplacement de ceux-ci en fonction de leur vitesse et du temps passer depuis le dernier appel de la cette méthode. Elle peut être redéfinie pour mettre à jour d'autre valeur.

Système de jeu

Le système du jeu se distingue par deux actions différentes : la mise à jour des valeurs, et l'affichage des objets. C'est la classe principale *Game* qui se charge d'exécuter ces actions grâce au fait qu'elle soit un *Thread.* Lorsque celui-ci est exécuté, il parcourt tous les objets pour les mettre à jour et les afficher.

Détection des collisions

La détection des collisions n'a pas été implémenté.

Sélection de la plante à planter

L'utilisateur choisi la plante à l'aide boutons dans la fenêtre. Il peut choisi entre *Pea, SnowPea, SunFlower, Nut & Shovel*. Malheureusement, les buttons ne sont pas visible dans la fenêtre, l'utilisateur ne peut donc pas choisir de plantes / pelle. La pelle doit permettre de déterre une plante.

Diagramme de classe UML

Description des classes

Box : Représente une case de la grille du jardin. Sert de « pot » pour planter les plantes.

Grid : C'est la Grille du jardin. Elle est composée de cases. Ces cases sont position en fonction des la position du jardin dans la fenêtre du jeu.

Game : C'est la classe principale du programme, le gestionnaire du jeu. Elle hérite de *Thread.*

GameUi : C'est une interface utilisateur représenté par un *JPanel*, elle détecte les actions de la souris (surtout les cliques).

Window : C'est la fenêtre du jeu, elle hérite de Frame. Elle contient les boutons pour la sélection des plantes ainsi que les actions à exécuter pour chaque bouton.

ZombieGenerator : Cette classe hérite de l'objet *Thread.* Elle s'occupe de générer les zombies en fonction du temps passé. *PlanType :* Cette classe est un énumérable qui sert à stocker le choix de la plante que l'utilisateur veux planter.

Entity : Cette interface permet de regrouper les différents objets du jeu et de les stocker dans le même champ.

Element : C'est la classe parent pour les objet qui seront sur le terrain. Elle contient les éléments de base pour afficher et déplacer une image sur le terrain.

Zombie : Cette classe hérite de *Element*, représente un zombie et implémente l'interface *Entity*.

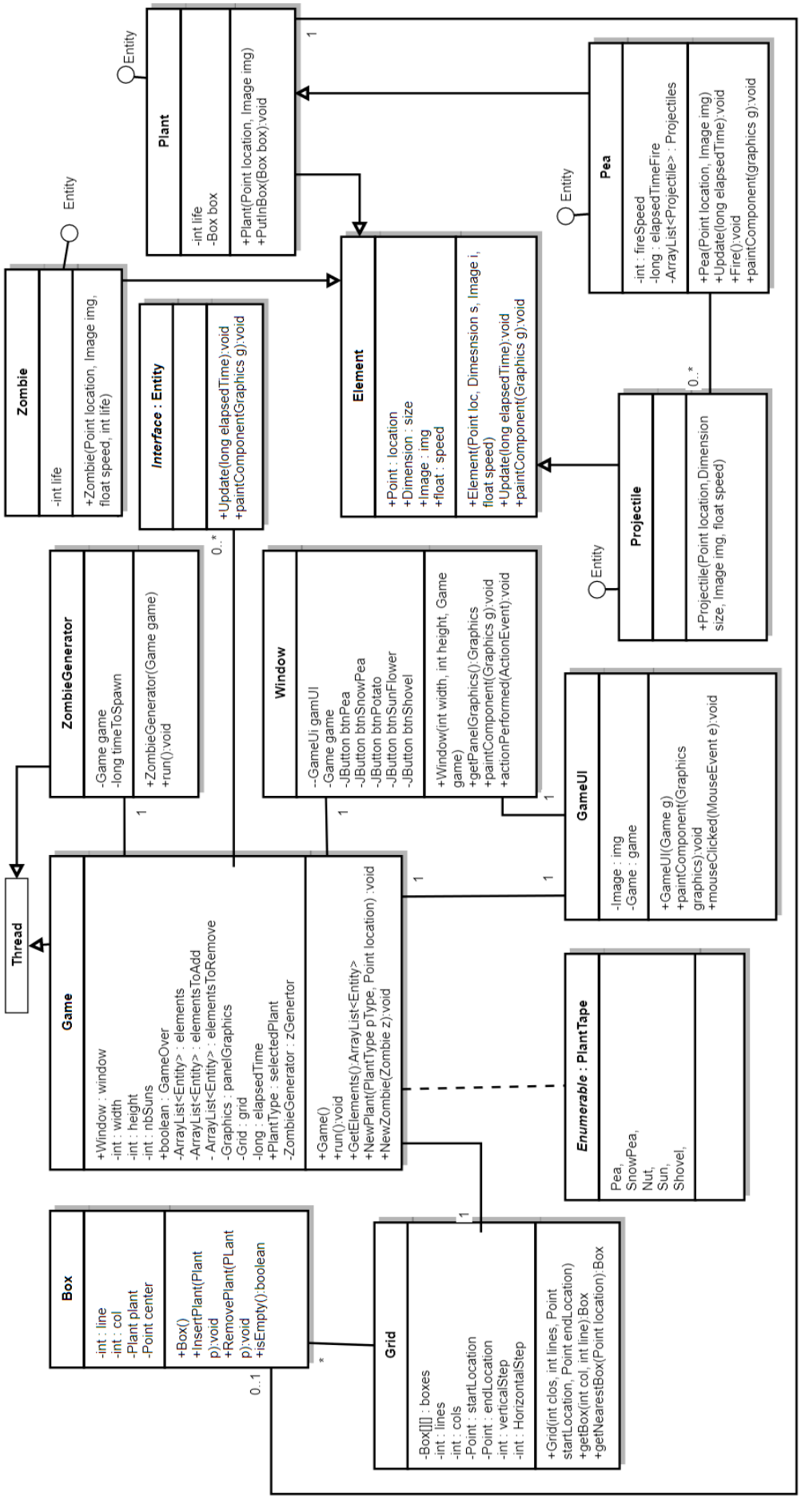
Plant : Cette classe hérite de *Element* et représente une plante et implémente l'interface *Entity*.

Pea : Cette classe hérite de *Plant* et implémente l'interface *Entity*. C'est une plante spécifique qui peut lancer des projectiles sur les zombies.

Projectile : Cette classe hérite de *Element* et implémente l'interface *Entity.* Elle représente les projectiles lancé par certaines plantes.

Bilan

Je n'ai pas consacré assez de temps à la réalisation de ce projet. Je n'ai donc pas pu terminer ni commencer certaines fonctionnalités (détection des collisions, chois de la plante par l'utilisateur, gestion du score, implémentation des autre plantes, déterrage d'une plante).


Plant
-int life
-Box box
+Plant(Point location, Image img)
+PutInBox(Box box):void
Entity
Zombie
-int life
+Zombie(Point location, Image img, float speed, int life)
Entity
Interface : Entity
+Update(long elapsedTime):void
+paintComponentGraphics g):void
Element
+Point : location
+Dimension : size
+Image : img
+float : speed
+Element(Point loc, Dimesnsion s, Image i, float speed)
+Update(long elapsedTime):void
+paintComponent(Graphics g):void
Entity
Pea
-int : fireSpeed
-long : elapsedTimeFire
-ArrayList<Projectile> : Projectiles
+Pea(Point location, Image img)
+Update(long elapsedTime):void
+Fire():void
+paintComponent(graphics g):void
Entity
Projectile
+Projectile(Point location,Dimension size, Image img, float speed)
ZombieGenerator
-Game game
-long timeToSpawn
+ZombieGenerator(Game game)
+run():void
Window
--GameUi gamUI
-Game game
-JButton btnPea
-JButton btnSnowPea
-JButton btnPotato
-JButton btnSunFlower
-JButton btnShovel
+Window(int width, int height, Game game)
+getPanelGraphics():Graphics
+paintComponent(Graphics g):void
+actionPerformed(ActionEvent):void
GameUI
-Image : img
-Game : game
+GameUI(Game g)
+paintComponent(Graphics graphics):void
+mouseClicked(MouseEvent e):void
Thread
Game
+Window : window
-int : width
-int : height
-int : nbSuns
+boolean : GameOver
-ArrayList<Entity> : elements
-ArrayList<Entity> : elementsToAdd
- ArrayList<Entity> : elementsToRemove
-Graphics : panelGraphics
-Grid : grid
-long : elapsedTime
+PlantType : selectedPlant
-ZombieGenerator : zGenertor
+Game()
+run():void
+GetElements():ArrayList<Entity>
+NewPlant(PlantType pType, Point location) :void
+NewZombie(Zombie z):void
Enumerable : PlantTape
Pea,
SnowPea,
Nut,
Sun,
Shovel,
Box
-int : line
-int : col
-Plant plant
-Point center
+Box()
+InsertPlant(Plant p):void
+RemovePlant(PLant p):void
+isEmpty():boolean
Grid
-Box[][] : boxes
-int : lines
-int : cols
-Point : startLocation
-Point : endLocation
-int : verticalStep
-int : HorizontalStep
+Grid(int clos, int lines, Point startLocation, Point endLocation)
+getBox(int col, int line):Box
+getNearestBox(Point location):Box
1
0..*
1
1
1
1
1
1
1
*
0..1