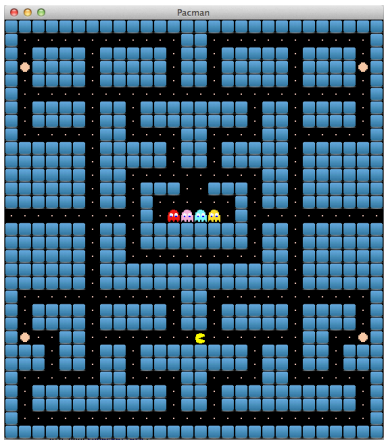
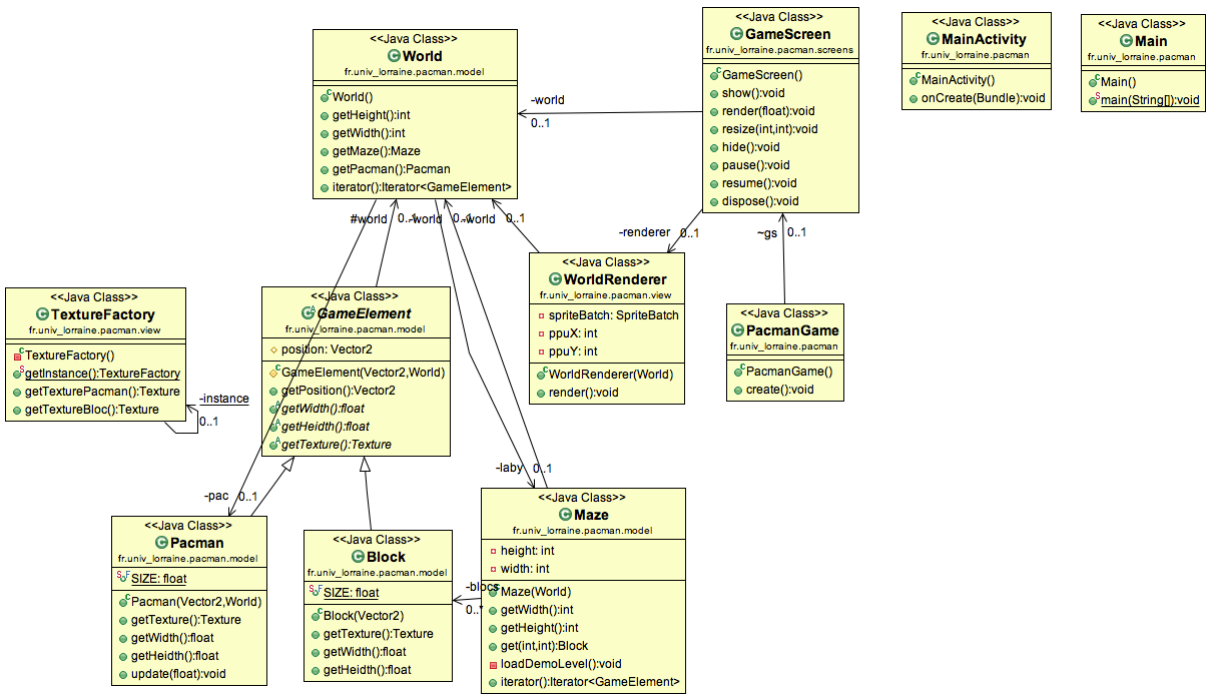
Projet LibGDX
2017-2018

Le but de ce projet est de réaliser le jeu du Pacman. A gauche le version originale, à droite la version minimale à développer.



Le diagramme UML ci-dessous indique une structure générale. Vous pouvez vous appuyer dessus comme base pour les classes mais pas sur leur contenu. Vous devez impérativement respecter au maximum un découpage MVC (séparation du modèle et de la vue).



Les classes Maze, GameElement et World représentent le modèle.

La classe Maze devra contenir le labyrinthe.

La classe World sera le regroupement des différents acteurs de ce jeu, acteurs statiques et dynamiques.

Les classes Maze et World doivent être itérables pour permettre l'utilisation de :

```
for (GameElement ge : Maze ou World) { … }
```

Cela permet de rajouter des GameElement dans Maze ou dans World sans pour autant modifier le code qui les parcourt. Cela peut s'implémenter de deux façons différentes.

La première se fait par l'intermédiaire d'une structure itérable. Exemple :

```
public class A implements Iterable<B>  {
      private B b1, b2, b3;
      …
      @Override
      public Iterator<B> iterator() {
            ArrayList<B> l = new ArrayList<B>();
            l.add(b1);
            l.add(b2);
            l.add(b3);
            return l.iterator();
      }
}
```

Cette solution n'est pas la meilleure. Il est également possible de définir son propre itérateur sans passer par une structure intermédiaire. Exemple :

```
public class A implements Iterable<B> {
      private B b1, b2, b3;
      …
      @Override
      public Iterator<B> iterator() {
            return new Iterator<B>() {
                  private int i = 0 ;

                  @Override
                  public boolean hasNext() {
                        return i < 2;
                  }

                  @Override
                  public GameElement next() {
                        if (!hasNext())
                              throw new NoSuchElementException("plus de B");

                        i++ ;
                        switch (i)
                        {
                        case 0  : return b1;
                        case 1  : return b2;
                        default : return b3;
                        }
                  }
            };
      }
}
```

La classe WorldRenderer servira de contrôleur afin de faire le lien entre le modèle et la vue.

La classe TextureFactory représente la vue, c'est elle et exclusivement elle qui contiendra les textures. Il faut implémenter la texture factory avec la méthode introduite dans les fichiers « Aide 1 et Aide 2.pptx » (iTexturable, etc.).

Le déplacement de Pacman se fait sans saccade. Il se fait en continue, Pacman continue dans la direction qui lui est affectée jusqu'à une intersection ou un mur le bloquant. Sa bouche s'ouvre et se ferme sans jamais s'arrêter même quand Pacman est bloqué par un mur. Lorsque l'utilisateur indique une nouvelle direction, le changement est appliqué à la prochaine intersection où le changement de direction est possible.

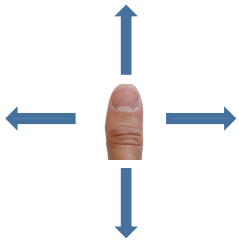
Vous devez implémenter DEUX techniques de changement de direction afin de contrôler Pacman. Le choix devra pouvoir se faire au début de la partie. Ces deux techniques sont à choisir parmi les quatre suivantes :

1)

- Par les touches de déplacement du curseur
  - Une touche appuyée/relâchée = un déplacement dans la direction demandée

2)

- Par un geste (le doigt glisse sur l'écran)
  - Un geste = un déplacement dans la direction du geste

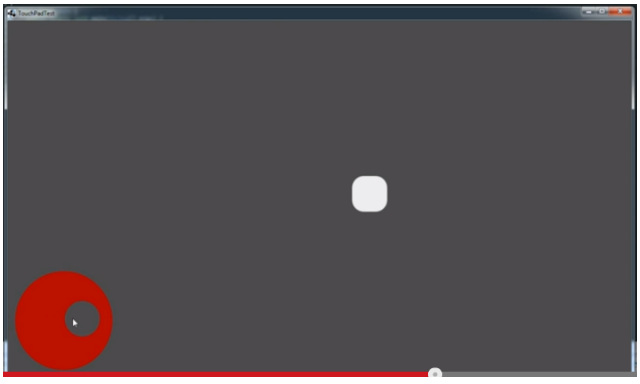


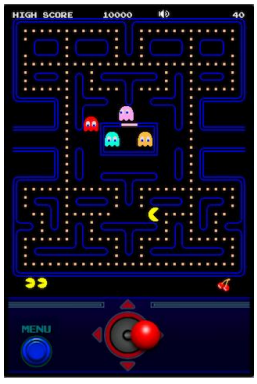
3)

- Par un appui sur l'écran
  - L'écran est séparé en 4 zones
  - Une zone = une direction
    - Haut, Bas, Gauche, Droit
  - Un appui/relâché = un déplacement dans la direction demandée

4)

- Par un joystick virtuel
  - http://www.bigerstaff.com/libgdx-touchpad-example/
  - Une impulsion dans une direction = un déplacement dans la direction demandée





Le labyrinthe doit comporter des Pac-gommes sur toutes les cases libres à l'exception de la maison des fantômes. Elles doivent s'afficher avec une texture particulière

Lorsque Pacman se déplace il mange les Pac-gommes dans les cases qu'il parcourt. La Pac-gomme mangée doit disparaître. Un score doit s'afficher, ce score doit tenir compte du nombre de Pac-gommes mangées et du temps. Lorsque toutes les Pac-gommes sont mangées, un écran de fin s'affiche avec le score visible.

Les quatre fantômes sont contrôlés par l'ordinateur à une vitesse paramétrable dans le code. Ils sont toujours en mouvement et regardent dans la direction de leur déplacement. Chaque fantôme a une stratégie différente :

- un fantôme se déplace aléatoirement. Sa direction change à chaque intersection.
- un fantôme cherche à rejoindre Pacman en minimisant à chaque intersection la distance en X et la distance en Y.
- un fantôme choisit soit la stratégie du premier fantôme ou du second fantôme à chaque intersection.
- un fantôme utilise un plus court chemin. Je propose d'utiliser un parcours en largeur (inondation à partir de Pacman ou inondation à partir du fantôme qui utilise cette technique).

Si Pacman rencontre un fantôme, Pacman meurt.
Si Pacman mange une super pac-gomme, Pacman peut manger les fantômes pendant un certain laps de temps :

- Les fantômes apparaissent d'une manière particulière,
- les doivent clignoter pour indiquer le temps restant à Pacman pour les manger,
- les fantômes mangés par Pacman se transforment en yeux et doivent rejoindre leur maison en parcourant le labyrinthe. Puis ils en ressortent après un certain délai.

Au lancement du jeu un écran doit en faire la publicité. Pour les écrans, il faudra utiliser des Screen. Un exemple est donné dans les fichiers « Aide 1 et Aide 2.pptx ».