Programmation orientée objet
TP5 : Space Invaders

Romain Marie & Hervé Gaudin

Octobre 2019

3iL INGENIEURS
</les sciences informatiques>

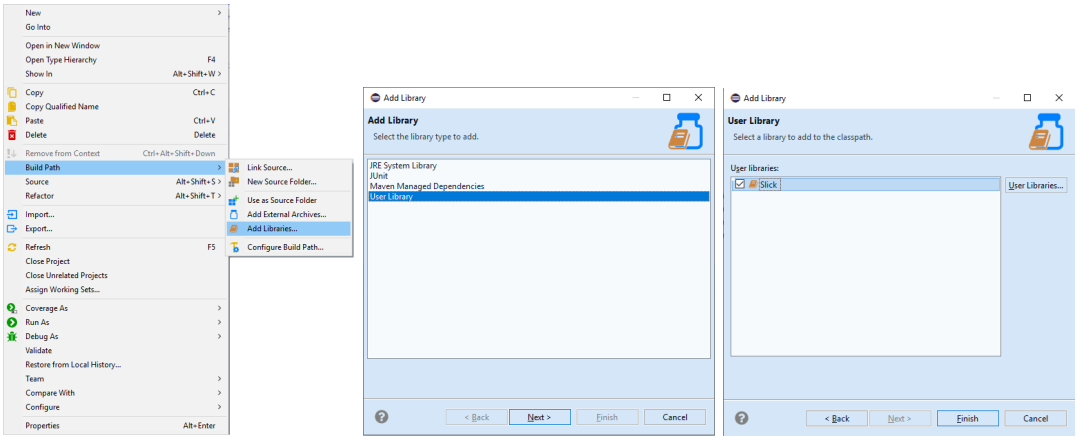
Dans ce TP, vous allez programmer un jeu de type space invaders. Les contraintes suivantes devront être respectées :

- La zone de jeu contient des ennemis immobiles (carrés pleins) qu'il faut détruire
- Le joueur déplace horizontalement un vaisseau (un oval) qui se trouve en bas de l'écran, grâce aux touches gauche et droite du clavier
- Grâce à la touche espace, il génère à la position du vaisseau, un projectile (disque de petite taille) qui se déplace verticalement à vitesse constante.
- En appuyant plusieurs fois sur la touche espace, un maximum de 5 projectiles pourront exister simultanément.
- Lorsqu'un projectile entre en collision avec un obstacle, les deux sont détruits
- Lorsqu'un projectile sort de l'écran, il doit être détruit
- Le jeu s'arrête lorsque tous les obstacles ont été détruits

1 Mise en place du projet

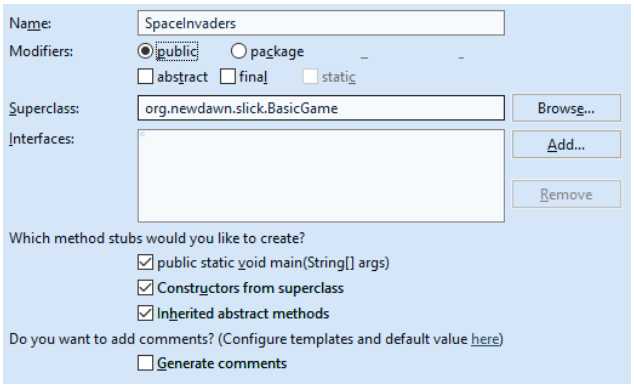
1.1 création et configuration du projet

1. Ouvrez Eclipse en prenant bien soin de choisir votre workspace qui a été configuré pour utiliser Slick2D.
2. Créez un nouveau projet que vous nommerez "Space Invaders".
3. Configurez ce projet pour qu'il utilise Slick2D. Un rappel illustré vous est présenté ci-dessous :



1.2 Classe principale et routine d'exécution

1. Ajoutez à votre projet une classe **SpaceInvaders** qui hérite de **BasicGame**. Cette méthode inclure une méthode *main()*, le constructeur faisant appel à la classe mère, et les méthodes abstraites. Pour vous aider, la configuration de la classe vous est présentée ci-dessous :



2. Ecrivez le contenu de la méthode *main()* pour qu'elle initialise et démarre l'application Slick. Il faut pour cela écrire le code source suivant :

```
public static void main(String[] args) throws SlickException {
    AppGameContainer app = new AppGameContainer(new SpaceInvaders("Space Invaders"));
    app.setShowFPS(false);
    app.start();
}
```

3. Exécutez votre projet, et vérifiez qu'une fenêtre vide sur fond noir s'ouvre. Si ce n'est pas le cas, appelez votre enseignant.

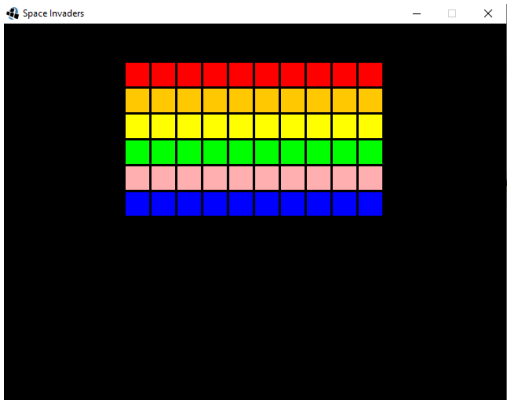
2 Création des ennemis

2.1 La classe Ennemi

1. Ajoutez à votre projet une classe **Ennemi**.
2. Ajoutez à cette classe les propriétés suivantes (dont vous déterminerez le type) :
   (a) **x,y** : Les coordonnées de son coin supérieur gauche
   (b) **c** : La couleur de l'ennemi
3. Ajoutez un constructeur avec paramètres qui permet d'entièrement spécifier les propriétés d'un ennemi.
4. Ajoutez une méthode *void dessiner(Graphics g)* qui dessine l'ennemi sous la forme d'un carré plein dont le côté vaut 30 pixels. Bien sûr, la position et la couleur devront correspondre aux propriétés.

2.2 Ajout et affichage des obstacles

1. Dans votre classe **SpaceInvaders**, ajoutez une **ArrayList** contenant des **Ennemi**, et que vous nommerez **e**. Si besoin, vous utiliserez internet pour la syntaxe.
2. Dans la méthode *init()*, utilisez deux boucles imbriquées pour remplir l'ArrayList avec 60 **Ennemi**. Comme illustré sur la première page, ils seront répartis en 6 lignes de 10 ennemis, chaque ligne ayant sa propre couleur. Le coin supérieur gauche du premier ennemi sera aux coordonnées (155,50), et l'espace entre chaque ennemi sera de 3 pixel (le deuxième sera donc en (188,50)).
3. Dans la méthode *render()*, parcourez l'ArrayList pour que chaque Ennemi soit dessiné à chaque itération.
4. Exécutez votre application. Vous devriez obtenir le résultat ci-dessous :



3 Gestion du vaisseau

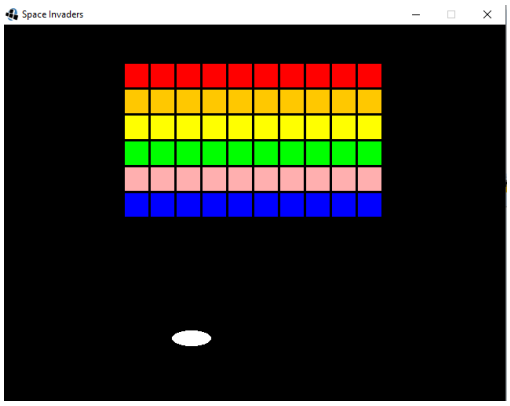
3.1 La classe Vaisseau

1. Ajoutez à votre projet une classe **Vaisseau**
2. Ajoutez à cette classe les propriétés suivantes (dont vous déterminerez le type) :
   (a) **x,y** : les coordonnées du centre de l'oval
3. Ajoutez un constructeur par défaut qui initialise le vaisseau aux coordonnées (340,400)
4. Ajoutez une méthode *void dessiner(Graphics g)* qui dessine le vaisseau sous la forme d'un oval blanc de rayons r1 = 25 pixels et r2 = 10 pixels. Vous ferez attention : les coordonnées x,y du vaisseau correspondent à son centre, pas au coin supérieur gauche de son rectangle englobant.

5. Ajoutez une méthode *void gauche()* et une méthode *void droite()* qui déplacent respectivement le vaisseau de 10 pixels vers la gauche et vers la droite.

3.2 Affichages et déplacements du vaisseau

1. Dans votre classe **SpaceInvaders**, ajoutez un **Vaisseau**, que vous initialiserez dans la méthode *init()*.
2. Mettez à jour la méthode *render()* pour que le vaisseau soit dessiné à chaque itération.
3. Ajoutez une méthode *void keyPressed(int key,char c)*. En vous inspirant du projet Tetris et/ou du support de cours Slick, définissez son contenu pour que les flèches gauche et droite du clavier entrainent un déplacement du vaisseau.
4. Exécutez votre application. Vous devriez pouvoir déplacer le vaisseau, et obtenir le résultat visuel suivant



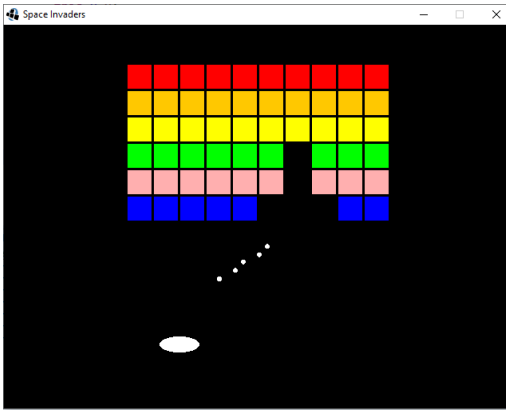
4 Projectiles et collisions

4.1 La classe Projectile

1. Ajoutez à votre projet une classe **Projectile**
2. Ajoutez à cette classe les propriétés suivantes (dont vous déterminerez le type) :
   (a) **x,y** : les coordonnées du centre du projectile
   (b) **vy** : la vitesse de déplacement du projectile
3. Ajoutez un constructeur qui initialise entièrement un projectile.
4. Ajoutez une méthode *void dessiner(Graphics g)* qui dessine le projectile sous la forme d'un disque blanc de rayon 3.
5. Ajoutez une méthode *void deplacer(int delta)* qui déplace le projectile en fonction de sa vitesse **vy** et du temps écoulé **delta**. Vous ferez attention, le projectile remonte l'image, ce qui veut dire que sa coordonnée y diminue.
6. Ajoutez une méthode *boolean horsEcran()* qui renvoie un booléen indiquant si la projectile est sorti de l'écran.
7. Ajoutez une méthode *boolean testCollision(Ennemi e)* qui renvoie un booléen indiquant si le projectile est en contact avec l'ennemi **e**.

4.2 Affichage, déplacements, et gestion des collisions

1. Dans votre classe **SpaceInvaders**, ajoutez une **ArrayList** contenant des **Projectile**, et que vous nommerez **p**.
2. Mettez à jour la méthode *render()* pour que chaque projectile existant soit dessiné à chaque itération.
3. Dans la méthode *update()*, faites en sorte que chaque projectile existant se déplace à chaque itération.
4. Toujours dans la méthode *update()*, testez pour chaque projectile s'il est sorti de l'écran. Si c'est le cas, supprimer le !
5. Toujours dans la méthode *update()*, testez pour chaque couple projectile/ennemi s'il y a collision. Si c'est le cas, retirez l'ennemi concerné de l'ArrayList **e**, et le projectile concerné de l'ArrayList **p**. **Attention : vous aurez probablement besoin d'ajouter des méthodes à la classe Ennemi**.
6. Mettez à jour la méthode *keyPressed()* pour que la touche Espace entraine la création d'un nouveau projectile au centre du vaisseau. La vitesse de chaque projectile sera de 50 pixels par seconde. **Attention, il ne peut pas y avoir plus de 5 projectiles existants au même instant**.
7. Exécutez votre programme. Normalement, votre jeu devrait être complet !



5 Pour aller plus loin

Pour les plus rapides d'entre vous, voici quelques idées d'amélioration :

- Améliorer l'aspect visuel : vous pouvez utiliser des images pour donner un peu plus de style aux ennemis et au vaisseau
- Mieux gérer les projectiles : vous pouvez ajouter un délai minimum entre deux tirs.
- Rendre les obstacles mobiles : vous pouvez donner à vos ennemis la capacité de se déplacer.
- ... : laissez parler votre imagination !