UNIVERSITÉ DE NANTES

BILLAUD Maël
GATARD Maxime

Rapport final Programmation Orientée Objet

1.Fonctionnalités de l'application :

Notre interface graphique est composée de deux parties :

- une partie graphique contenant le dessin
- une partie "outils" avec tous les boutons destinés à changer les propriétés du dessin

Dans la partie outils, nous avons :

- Un bouton pour relancer le programme
- Des boutons radio pour choisir la forme
- Des boutons radio pour choisir le chemin
- Un bouton pour quitter le programme

Lorsque l'on lance le programme, un tracé de cercles suivant le chemin lemniscate est affiché.

2. Encapsulation :

L'encapsulation est respectée dans l'intégralité du code, sans quoi il ne pourrait pas fonctionner correctement. Nous faisons appel à beaucoup de getter et de setter afin de respecter l'intégrité et la cohérence de l'objet.

3.Relations d'héritage :

- Super classe : Drawable
- Sous classe : Form

La classe Form doit implémenter Drawable si l'on veut pouvoir dessiner les formes dans l'interface graphique.

- Super classe : Form
- Sous classes : Circle, Square

Nous avons créé deux sous classes de la classe Form car nous ne créons jamais d'objet de type Form, nous ne créons que des carrés et des cercles. Cependant, ces deux formes possèdent un attribut commun : le centre, qui est de type Point et qui nous permet de déplacer la forme le long d'un chemin (c'est ce point qui servira de référence pour le déplacement dans la méthode evolve de la classe Form). On crée donc un attribut centerForm qui constitue la liste des points que l'on va utiliser comme centres pour construire le chemin

- Super classe : Way
- Sous classes : Lemniscate, Spirale

Pour les mêmes raisons que la classe Form, on ne crée pas d'objets de type Way.

En revanche, nous avons deux chemins différents qui se construisent avec des formules mathématiques différentes : le Lemniscate et la Spirale. Ces deux formules se trouveront dans la méthode baseWay que l'on implémente dans les sous-classes

4.Polymorphisme

Dans ce projet, on met en oeuvre un polymorphisme dans 4 sous-classes :

- D'abord, dans nos sous-classes Circle et Square qui sont des classes filles de Form. Dans ces deux classes, nous allons implémenter les méthodes draw et contains de sorte à ce que l'on puisse dessiner un carré ou un cercle
- Il en va de même pour les deux sous-classes Lemniscate et Spirale qui implémentent la classe Way. En effet, ces deux sous-classes implémentent à leur manière la méthode baseWay qui permet de créer le chemin que va suivre la forme à partir de leur formule mathématiques.

L'utilisation du polymorphisme ici est capitale car on ne construit pas un carré comme on construit un cercle, ni un Lemniscate comme une Spirale. Ces classes ont des particularités de construction qui leur sont propres, il est donc nécessaire de les implémenter différemment.

5.Extensibilité

L'extensibilité de notre modèle se base sur les super-classe. En effet, nul besoin de modifier les super-classes si l'on veut, par exemple, rajouter une forme ou un chemin. Il suffit de créer une nouvelle classe fille qui implémente la super-classe et d'implémenter les méthodes correspondantes.

6.Interface graphique

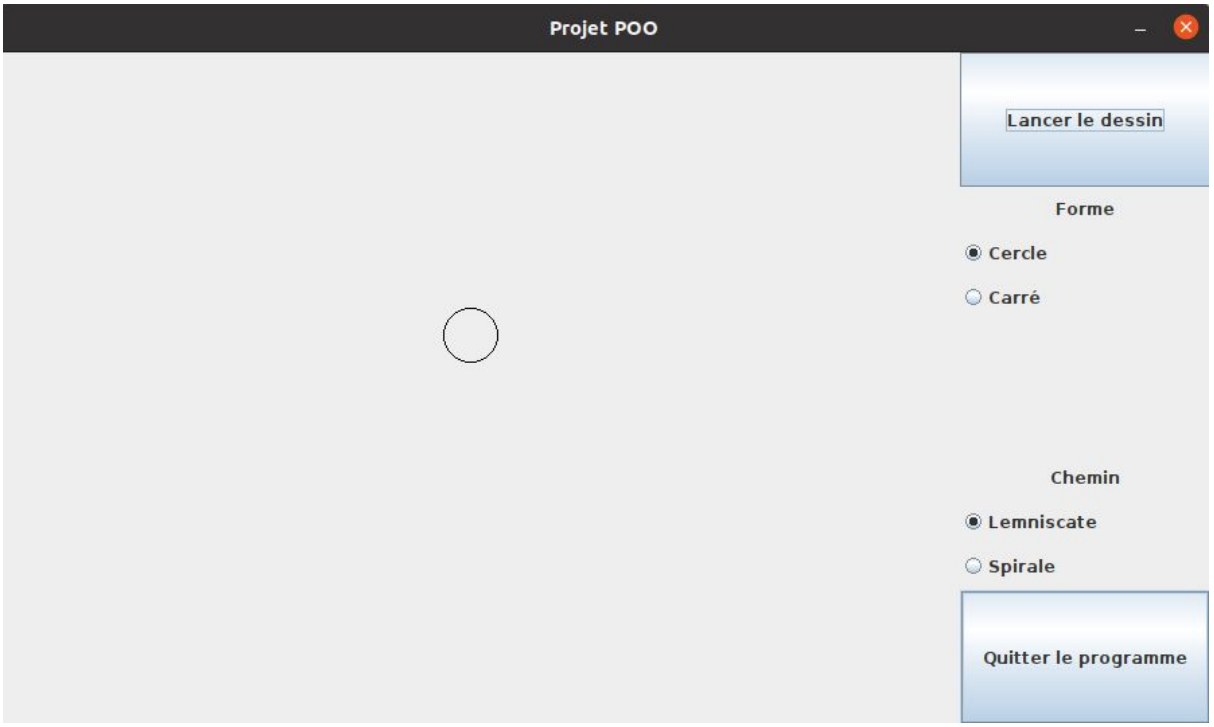
Notre interface graphique fonctionne selon le modèle MVC : la classe Form représente le modèle, notre classe view gère la vue et notre controller permet de gérer les interaction (ex : mouseclick)

7.Interactions

En ce qui concerne les interactions, l'utilisateur peut choisir la forme et le chemin qu'il veut représenter dans l'interface graphique. Il peut aussi changer la couleur de la forme (et de la trace qu'elle laisse) en cliquant au centre de la dernière forme créée. Le programme affecte un couleur aléatoire à chaque fois, néanmoins différente de la précédente. Lorsque l'utilisateur clique sur le bouton "Quitter le programme", le programme est quitté. Lorsqu'il appuie sur le bouton "Lancer le dessin", la fenêtre actuelle est fermée et une nouvelle est ouverte en tenant compte des différents JRadioButton sélectionnés précédemment. Malheureusement, ce dernier ne fonctionne pas correctement. Au lieu d'afficher, comme au lancement du programme, le tracé de la forme sélectionnée suivant le chemin sélectionné, un fenêtre blanche est affichée et le dessin se fait un fond. Puis une fois le dessin terminé, nous avons l'affichage de ce dernier avec la barre d'outil.

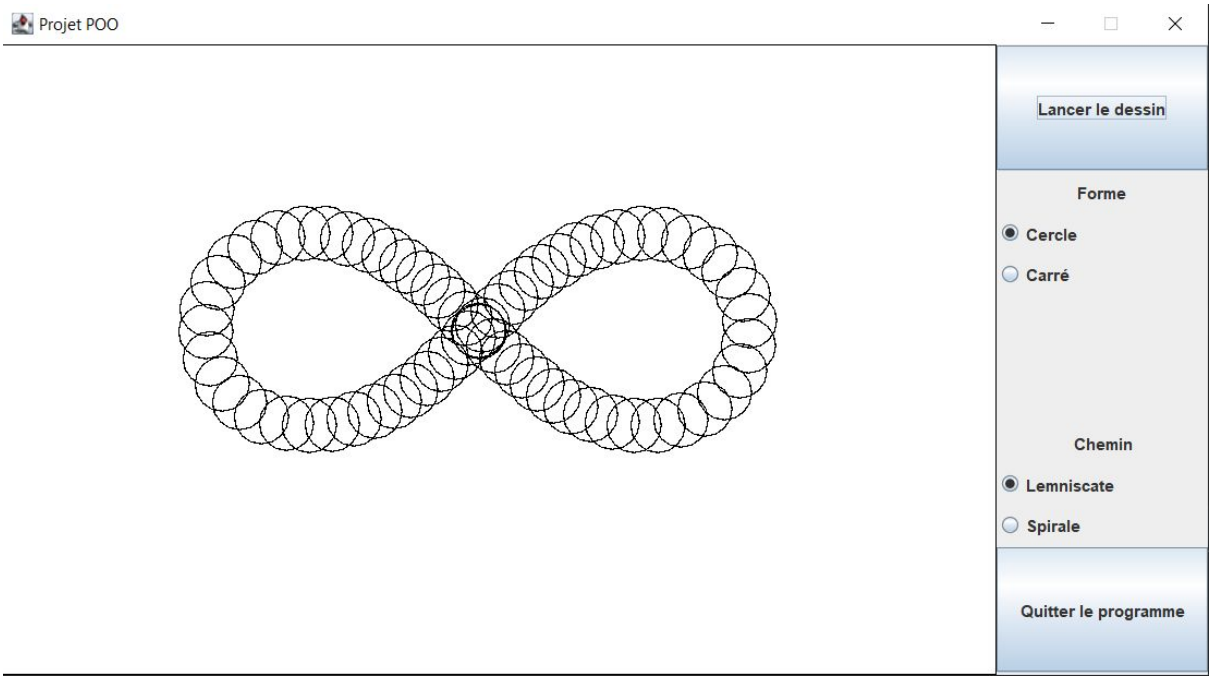
8.Expérimentations

Lors de notre initiation à l'interface graphique, nous avons commencé par faire bouger un cercle le long d'une droite :

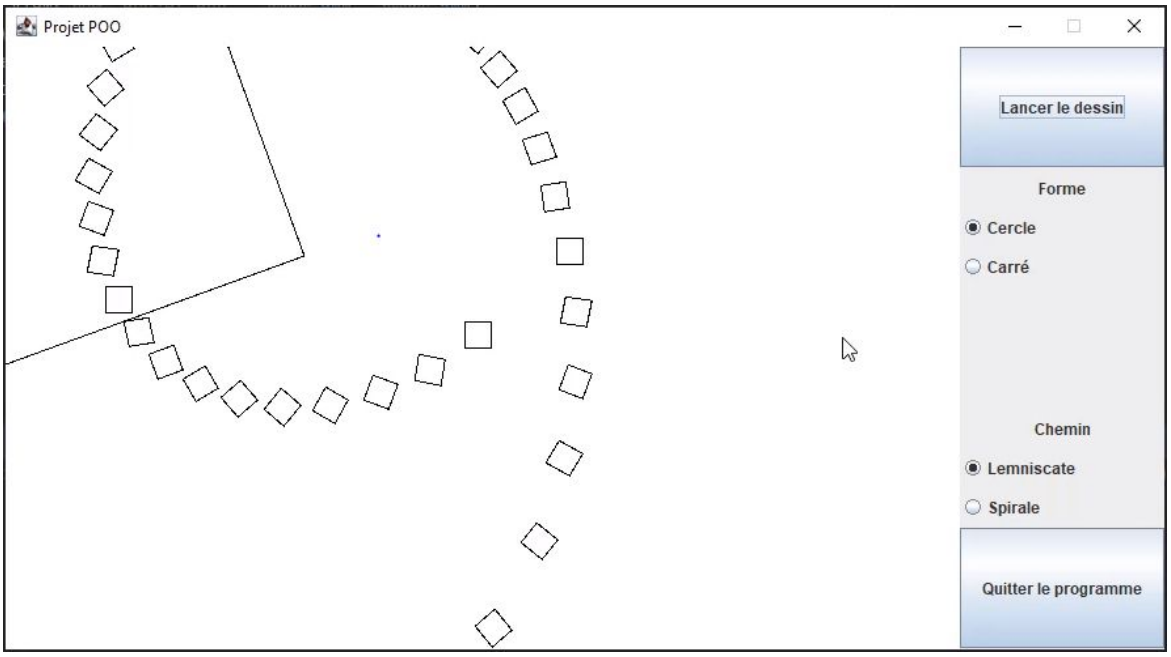


Ensuite, nous avons implémenter le Lemniscate ce qui nous a permis de faire bouger une forme le long de ce chemin (le cercle comme le carré)

Après quoi nous avons réussi à laisser la trace de la forme sur le chemin :



Ensuite nous avons fait plusieurs essais pour effectuer un mouvement de rotation avec le carré (ici avec le lemniscate) qui n'ont pas abouti :



9.Regard critique

Avec du recul, on peut s'apercevoir que notre projet aurait pu être amélioré sur certains points. D'abord, lorsque l'on relance le dessin depuis l'interface, la fenêtre s'affiche d'abord en blanc avant d'afficher le dessin déjà terminé. On ne voit donc pas la forme bouger.
De plus, nous n'avons pas réussi à régler le problème de rotation du carré, nous avons donc dû abandonner l'idée. Cependant nous avons peut être une idée pour le mettre en œuvre. Comme vous avez pu le voir sur notre dernière image, nous avons fait tourner tout le panel. Part ailleurs, faire une rotation du point autour du centre du carré ne servirait à rien car pour créer notre carré on se sert de g.drawRect qui le dessine avec les coordonnées d'un point (celui en haut à gauche du carré), donc si on bouge ce point, le carré ne subira pas de rotation.
Il faudrait donc créer à chaque fois des carrés dans des panel (qui se superposent sans cacher les panels qu'ils recouvrent) et faire tourner les panels. Cependant nous ne voyons pas comment mettre en œuvre cette idée.
De plus, les touches de clavier ne sont pas reconnues malgré l'implémentation de la classe KeyListener, et pour des raisons inconnues, les cliques souris dans la dernière forme ne sont pas toujours reconnues lorsqu'elle est en mouvement.

En revanche, ce projet nous a beaucoup appris, notamment sur la partie graphique et le MVC. En effet, nous n'avons pas eu de TD sur ces deux parties, et les transparents du cours ne nous permettaient pas d'implémenter un controller utilisable. Nous avons donc dû nous documenter par nous même et cela nous a permis de mieux nous imprégner du sujet et des notions.