2 - Enveloppes Convexes

1) du modèle à l'écran

On considère un ensemble de points dans leurs coordonnées d'origine traditionnellement dans un repère "mathématique" direct(O,i,j). On considère que les coordonnées des points de l'ensemble sont comprises en x dans [Xmin,Xmax] et en y par [Ymin, Ymax]. On souhaite afficher ces points à l'écran dans une fenêtre ou dans une zone d'une fenêtre, appelée viewport, ayant son propre système de coordonnées et son propre repère. Le viewport constitue une zone d'affichage traditionnellement défini par les coordonnées du coin supérieur gauche (Ox,Oy), les dimensions de la zone (W,H), et un repère indirect.

On note (x,y) les coordonnées d'un point quelconque dans le modèle d'origine et (xs,ys) les coordonnées de ce même point dans le viewport.

1. Quelle transformation permet de calculer (xs,ys) en fonction (x,y)?
2. Implémenter cette transformation dans une classe `SpaceToViewport` paramétrée par les caractéristiques de l'espace des points d'origine (Xmin, Xmax, Ymin, Ymax) et du viewport (Ox,Oy,W,H).
3. On envisage de représenter l'ensemble des points d'origine dans une seule zone de dessin et de mettre en oeuvre quelques interactions. Envisager les structures de données et mécanismes logiciels permettant de conserver pour chaque point, ses coordonnées dans le modèle et ses coordonnées à l'écran et de limiter le recalcul des coordonnées écran aux cas où le modèle change.
4. Avec votre implémentation (1) afficher un ensemble de points donnés dans leurs coordonnées (repère direct et coordonnées bornées par [Xmin, Xmax], [Ymin, Ymax]) et (2) modifier interactivement les points en les sélectionnant et les déplaçant à l'écran à la souris.

   Tester votre implémentation en utilisant ces jeux de données : spaceToViewport.zip
5. Mettre en oeuvre les interactions permettant les opérations d'édition suivantes: (1) ajout d'un point, (2) suppression d'un point existant et (3) déplacement d'un point.

2) Tris géométriques d'un ensemble fini de points du plan

Pour cet exercice, vous pouvez implémenter les différents ordres à envisager dans des classes de comparateurs qui héritent de `Comparator` pour ensuite utiliser la méthode statique `sort` de la classe `Collections` pour obtenir le tri optimisé d'une liste de points selon vos comparateurs.

- Les relations définies ci-dessous, sont-elles des relations d'ordre?

  $R1 = \{(P, M) \mid P.x < M.x \text{ ou } (P.x = M.x \text{ et } P.y \leq M.y)\}$

  $R2 = \{(P, M)_O \mid P \leq_O M\}$ avec la relation de précédence notée $\leq_O$, définie par $P \leq_O M \Leftrightarrow \{0 < \alpha < \Pi \text{ ou } ((\alpha = 0 \text{ ou } \Pi) \text{ et } OP \leq OM)\}$ avec α = mesure de l'angle orienté $(\overrightarrow{OP}, \overrightarrow{OM})$
- Pour un ensemble fini de points noté V, effectuer les tris suivants:
  1. tri x,y, appelé tri lexicographique. Ce tri utilise la relation définie à la question précédente: il ordonne les points selon leur abscisse (x) croissante et en cas d'égalité sur l'abscisse, la comparaison se fait sur ordonnée (y) croissante.
  2. le tri y,x est une variante de ce tri qui s'obtient en inversant les rôles respectifs de x et y.
  3. tri θ,ρ, appelé tri polaire de centre O, de direction $\overrightarrow{u}$, ordonne les points selon leurs coordonnées polaires pour $(O, \overrightarrow{u})$. Dans cette première version l'ordre est le même pour θ,ρ. Ainsi, $P \leq M \Leftrightarrow P.\theta < M.\theta \text{ ou } (P.\theta = M.\theta \text{ et } P.\rho \leq M.\rho)\}$.
  4. le tri θ,ρ, dans cette deuxième version est similaire à la version précédente mais en inversant l'ordre pour ρ. Ainsi, $P \leq M \Leftrightarrow P.\theta < M.\theta \text{ ou } (P.\theta = M.\theta \text{ et } P.\rho \geq M.\rho)\}$.
- A quelle(s) condition(s), la relation R2 peut-elle devenir une relation d'ordre pour V?
  1. En déduire le calcul d'un ordre polaire de V n'utilisant ni les fonctions trigonométriques, ni leurs inverses.

Tester vos implémentations en utilisant ces jeux de données

3) Génération aléatoire des points et structures des données

Télécharger le code source pour commencer. Modifier la méthode d'initialisation des points de la classe `Vue` pour générer des points sur un disque de centre le centre de la fenêtre et de rayon `r`. La méthode d'initialisation prend alors pour arguments:

- `n` : nb points,
- `width`: largeur et hauteur de la fenêtre, et
- `r`: rayon du disque compris entre 0 et width.

NB: pour les tests initiaux, vous pouvez prendre `n = 20, width = 800, r = 250`.

4) Algorithme des demi-plans

En partant de l'algorithme des demi-plans, écrire les méthodes utiles au calcul de l'enveloppe convexe d'un ensemble de points et afficher les points et l'enveloppe convexe des points ainsi calculée.

Tester l'algorithme sur les points donnés dans des ensembles de points simples.

Tester l'algorithme sur les ensembles de points générés aléatoirement.

5) Algorithme de Jarvis (aussi appelé algorithme de la ficelle ou de l'emballage cadeau)

Ecrire la méthode `jarvis` qui part d'un ensemble de points du plan et qui construit la liste `envConv` contenant la liste cyclique des points formants les sommets de l'enveloppe convexe en utilisant l'algorithme de Jarvis. La liste cyclique des points de l'enveloppe convexe est la liste des points de l'enveloppe dans l'ordre de calcul de l'algorithme de Jarvis avec la répétition du dernier point qui est aussi le premier.

Tester l'algorithme sur les points donnés dans des ensembles de points simples.

Tester l'algorithme sur les ensembles de points générés aléatoirement.

6) Mise à jour dynamique de l'enveloppe convexe

Dans cet exercice, on suppose qu'un ensemble de points V est donné avec son enveloppe convexe EC et l'objectif est d'envisager l'ajout et la suppression de points de l'ensemble V après la construction initiale de l'EC.

Donner un algorithme qui permet d'ajouter un point à l'ensemble initial de points et à mettre à jour l'enveloppe convexe associée en O(k), où k représente le nombre de points dans EC.

Ecrire une méthode qui permet de supprimer un point de EC et de mettre à jour EC en O(n) où n représente le nombre de points de V.

7) Algorithme de Graham

Cet exercice consiste à implémenter l'algorithme de Graham vu en cours.

8) mesures empiriques

Ecrire les méthodes utiles à la mesure du temps mis pour calculer l'enveloppe convexe par chaque implémentation.

Faire la mesure du temps pour des ensembles de points de 10 à 10 000 pour tester empiriquement les implémentations des trois algorithmes et les comparer aux complexités théoriques.

9) union et intersection d'enveloppes convexes

Dans cet exercice, on considère plusieurs ensembles de points pour lesquels les enveloppes convexes sont données.

1. Proposer une méthode qui calcule l'intersection des enveloppes convexes de ces ensembles de points.
2. Proposer une méthode qui calcule l'union des enveloppes convexes de ces ensembles de points.
3. Déduire de la question précédente un algorithme permettant de calculer l'enveloppe convexe d'un ensemble de points en procédant par dichotomie.