Travaux pratiques de mathématiques appliquées

Graphisme 2D et en 3D

Durée prévue : 3 H

1. Introduction

1.1. Objectifs

Les objectifs principaux de ce laboratoire sont de :

- comprendre les notions de coordonnées cartésiennes à deux et à trois dimensions ;
- transformer des coordonnées cartésiennes en coordonnées à l'écran ;
- afficher à l'écran une figure en deux dimensions ;
- représenter une figure 3D en perspective sur un écran 2D ;
- dessiner une figure récursive : « La courbe du dragon ».

Les notions théoriques utilisées dans les différents exercices sont expliquées en détail ci-après. Elles seront également introduites par votre responsable de laboratoire.

1.2. Evaluation

Les séances de travaux pratiques ont un caractère essentiellement formatif. Les exercices proposés dans cet énoncé ne seront pas notés. Néanmoins, il est fondamental que vous tentiez d'en résoudre un maximum par vous-même, ceci afin d'acquérir une bonne compréhension et une bonne maîtrise de la matière du cours.

Si vous éprouvez des difficultés, n'hésitez pas à solliciter l'aide de votre responsable de laboratoire. Vous pouvez également collaborer avec vos condisciples pour élaborer les solutions. Assurez-vous cependant que vous avez compris les notions mises en pratique. Ne vous contentez pas de recopier une solution sans la comprendre : vous devez être capable de la recréer par vous-même.

Les notions abordées dans ce laboratoire seront évaluées lors d'un examen pratique.

1.3. Consignes générales

Les exercices proposés dans cet énoncé doivent être résolus au moyen du langage Java et de l'environnement de développement Eclipse, en complétant le projet de départ « 2425_B1MATH_GPH.zip » mis à votre disposition sur HELMo Learn.

Le projet de départ contient toutes les classes citées dans cet énoncé avec, dans celles-ci, les prototypes des méthodes obligatoires que vous devrez compléter pour la réalisation des différentes tâches. Si nécessaire, vous pouvez enrichir le projet avec vos propres packages, classes et méthodes.

Chaque méthode est précédée d'une Javadoc. Lisez attentivement les informations qui y sont données en complément de l'énoncé.

2. Systèmes de coordonnées

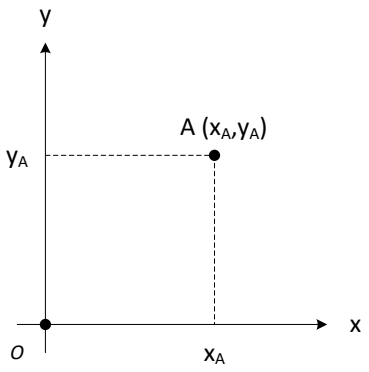
2.1. Coordonnées cartésiennes à deux et à trois dimensions

Nous souhaitons pouvoir afficher à l'écran une figure composée d'un ensemble de lignes dont les coordonnées des points extrémités sont exprimées dans un *système de coordonnées cartésiennes*. Pour obtenir une définition détaillée de ce système de coordonnées, nous vous renvoyons par exemple à : http://fr.wikipedia.org/wiki/Coordonnées_cartésiennes.

En pratique, nous pouvons dire que dans un système de coordonnées cartésiennes à 2 dimensions, la position d'un point A dans un plan est définie par un couple de nombres (X_A, Y_A) :

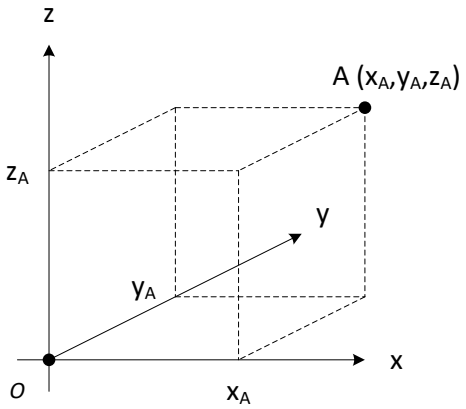
- la coordonnée X_A est appelée *« abscisse »*,
- la coordonnée Y_A est appelée *« ordonnée »*.

Chaque coordonnée exprime une distance sur l'axe correspondant, par rapport à un repère cartésien *O* appelé *origine* dont les coordonnées valent (0, 0). La valeur de cette distance dépend de la graduation de l'axe (échelle, unité).



Dans un système de coordonnées cartésiennes à 3 dimensions, la position d'un point A dans l'espace est définie par un triplet de nombres (X_A, Y_A, Z_A) :

- la coordonnée supplémentaire Z_A est appelée *« hauteur »* ou « cote ».



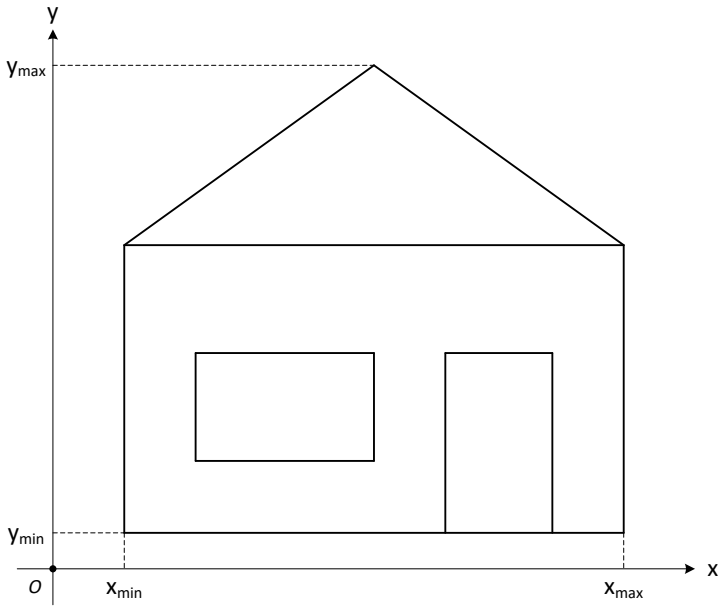
L'*orientation des axes* peut varier d'un système de coordonnées à l'autre. Nous utiliserons les orientations illustrées sur les figures précédentes, à savoir :

- l'axe X est horizontal et orienté vers la droite tant en 2D qu'en 3D ;
- l'axe Y est orienté vers le haut en 2D ou dans le plan horizontal vers l'arrière en 3D ;
- l'axe Z (3D) est orienté vers le haut.

2.2. Affichage à l'écran d'une figure en deux dimensions

L'affichage d'une figure quelconque dans une fenêtre graphique pose un certain nombre de problèmes que nous allons devoir résoudre.

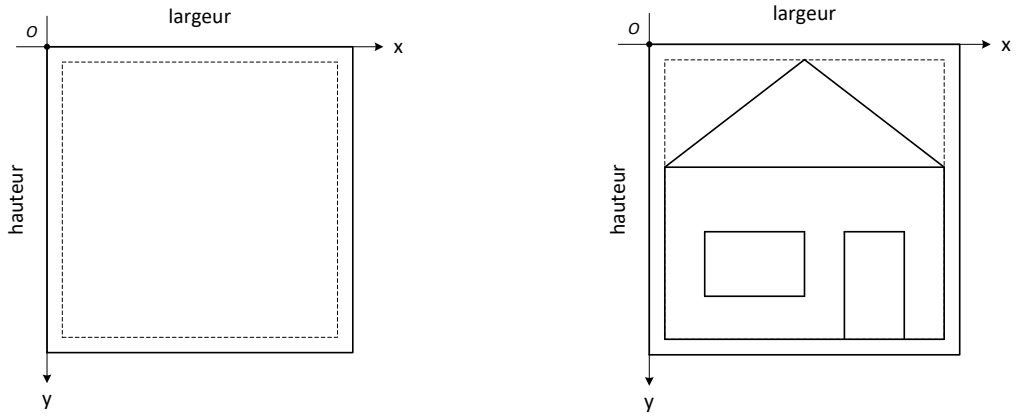
Supposons que la figure à dessiner soit la suivante :



Cette figure possède les caractéristiques suivantes :

- elle occupe une zone qui s'étend des coordonnées X_{min} à X_{max} selon l'axe des X et de Y_{min} à Y_{max} selon l'axe Y ;
- la valeur des coordonnées des différents points dépend de l'unité utilisée (exemple : X_{max} pourrait valoir 10 si on s'exprime en mètres, ou bien 10.000 si on s'exprime en millimètres) ;
- le coin inférieur gauche de notre figure ne correspond pas nécessairement à l'origine du système d'axes : X_{min} et Y_{min} peuvent avoir des valeurs quelconques, positives ou négatives.

Notre fenêtre d'affichage graphique possède quant à elle des caractéristiques bien définies :



- la *largeur* et la *hauteur* de la zone de dessin sont fixes et exprimées en nombre de points ;
- l'axe X est horizontal et orienté vers la droite ;
- l'axe Y est vertical et orienté vers le bas ;
- l'origine du système de coordonnées (0,0) est située en haut à gauche ;
- le point de coordonnées (*largeur*, *hauteur*) correspond au coin inférieur droit.

Quelles que soient les dimensions et la position de la figure de départ, notre objectif est de l'afficher dans la fenêtre graphique de telle manière qu'elle soit orientée correctement et occupe tout l'espace disponible à l'exception d'une *marge* de taille prédéfinie située sur le pourtour du dessin.

Pour parvenir au résultat final, il est nécessaire de réaliser une série d'opérations :

1. *Translation* des coordonnées pour faire coïncider le coin inférieur gauche du dessin avec l'origine (0, 0) du système de coordonnées.

$$X' = X - X_{min} \qquad Y' = Y - Y_{min}$$

2. *Mise à l'échelle* en X et en Y pour adapter à la taille de la figure à celle de la zone de dessin.
   - Déterminer les coordonnées $X_{min}, X_{max}, Y_{min}, Y_{max}$ pour l'ensemble des points de la figure afin de déterminer la largeur Δ_X et la hauteur Δ_Y de la zone occupée :

$$\Delta_X = (X_{max} - X_{min}) \qquad \Delta_Y = (Y_{max} - Y_{min})$$

   - Déterminer la largeur L_U et la hauteur H_U <u>utiles</u> de la fenêtre d'affichage graphique en tenant compte de la marge :

$$L_U = largeur - 2 \times marge \qquad H_U = hauteur - 2 \times marge$$

   - Le facteur d'échelle s'obtient en calculant le rapport entre la taille disponible à l'écran et la taille de la figure :

$$E_X = L_U/\Delta_X{}^{(*)} \qquad E_Y = H_U/\Delta_Y{}^{(*)}$$

   - Pour obtenir une coordonnée exprimée en nombre de « points écran », il suffit alors de multiplier la coordonnée réelle par le facteur d'échelle :

$$X_E = X' \cdot E_X \qquad Y_E = Y' \cdot E_Y$$

3. *Calcul des coordonnées finales à l'écran.*
   - Décaler X_E et Y_E d'un nombre de points qui correspondant à la marge autour du dessin.
   - Inverser l'axe Y.

$$X_{final} = X_E + marge \qquad Y_{final} = hauteur - (Y_E + marge)$$

(*) En procédant de cette façon, la mise à l'échelle est calculée indépendamment pour chaque axe, de manière à utiliser tout l'espace d'affichage disponible. La figure affichée sera dès lors déformée si la fenêtre d'affichage n'a pas les mêmes proportions (largeur/hauteur) que celle de la figure de départ.
Pour conserver les proportions, il faut choisir la même valeur pour E_X et pour E_Y.

Exercice 1 – Dessin fil de fer en 2D

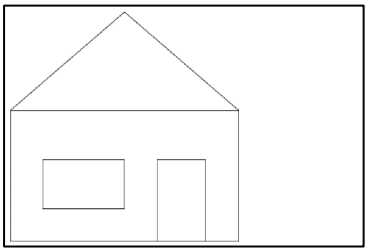
- La classe « `Dessin2D` » du package « `dessin` » permet d'afficher des dessins de type « fil de fer » en 2D.
- Complétez les méthodes suivantes pour que l'affichage final soit correct :

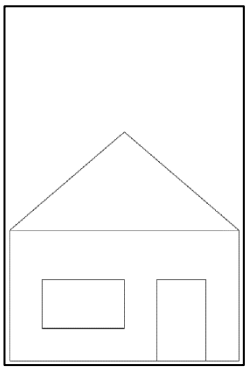
```
public void miseAJourBornesMinMax(Point2D point)
public void miseAJourEchelleX()
public void miseAJourEchelleY()
public int xAffichage(double x)
public int yAffichage(double y)
```

- Consultez la documentation du code pour de plus amples informations sur l'utilité des différentes méthodes.
- Note : dans cet exercice, vous ne devez pas conserver les proportions du dessin.
- Des exemples de dessins 2D sont préconfigurés dans la classe « DessinFilDefer ». Vérifiez que vos méthodes fonctionnent correctement en exécutant la méthode `main()` de cette classe.

Exercice 1bis – Mise à l'échelle proportionnelle

- Modifiez la classe « `Dessin2D` » de manière prendre en considération le paramètre « `proportions` » du constructeur.
- Si la valeur du paramètre est « `true` » vous devez utiliser la zone d'affichage disponible au maximum, sans altérer les proportions de l'image de départ.
- Les deux exemples de test illustrés ci-dessous sont disponibles dans la classe « DessinFilDefer ». Supprimez les symboles de commentaires pour les exécuter.

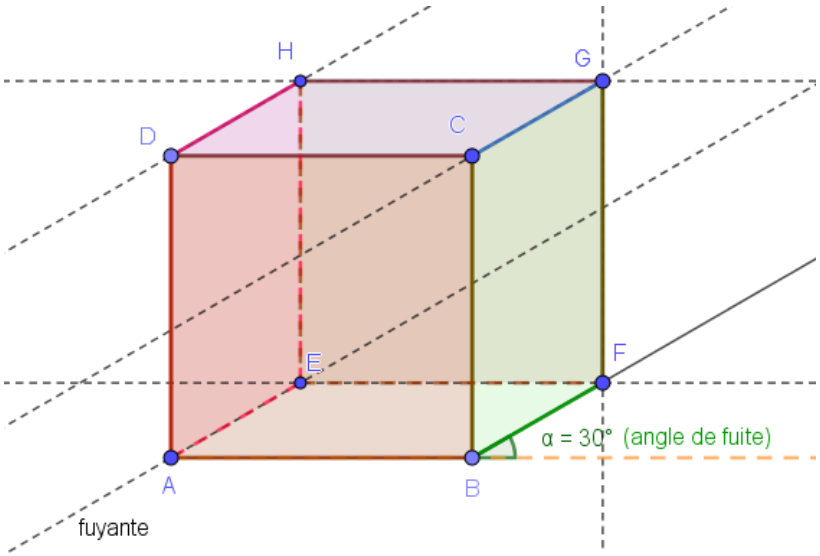




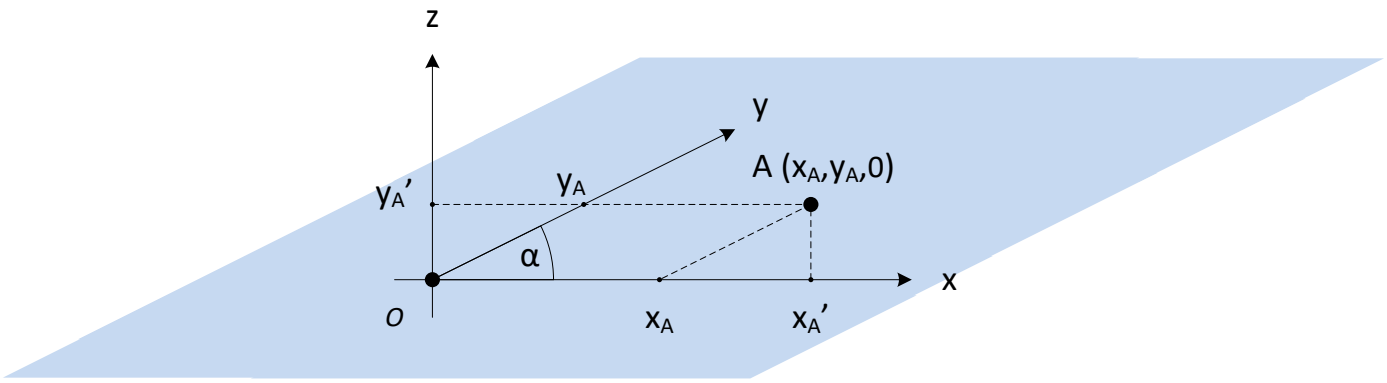
2.3. Affichage à l'écran d'une figure en trois dimensions

Pour afficher une figure en trois dimensions sur un écran d'ordinateur qui n'en comporte que deux, nous allons réaliser une *représentation en perspective cavalière*.

Exemple : représentation d'un cube en perspective cavalière :



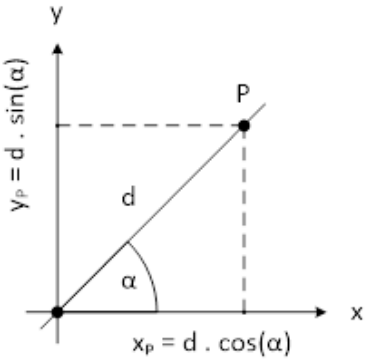
Considérons tout d'abord le cas d'un point A dont la hauteur est nulle, c'est-à-dire dont la coordonnée $Z_A = 0$. La situation est représentée à la figure suivante.



Les coordonnées du point A dans notre vue en perspective 2D sont notées (X'_A, Y'_A).

Si α est l'angle utilisé pour la représentation en perspective (*angle de fuite*), alors :

- $X'_A = X_A + Y_A \cdot cos(\alpha)$
- $Y'_A = Y_A \cdot sin(\alpha)$



Si la hauteur Z_A du point A n'est pas nulle, il suffit de l'ajouter à Y'_A.
On obtient finalement :

- $Y'_A = Y_A \cdot sin(\alpha) + Z_A$

Généralement, la perspective cavalière introduit aussi un *coefficient de réduction* K qui permet de faire varier l'effet de profondeur sur l'axe Y, ce qui donne :

- $X'_A = X_A + Y_A \cdot K \cdot cos(\alpha)$
- $Y'_A = Y_A \cdot K \cdot sin(\alpha) + Z_A$

Mise à l'échelle

En deux dimensions, la mise à l'échelle en X dépend de la *largeur* de la fenêtre et de l'intervalle des abscisses utilisées : $\Delta_X = X_{max} - X_{min}$.
De même, la mise à l'échelle en Y dépend de la *hauteur* la fenêtre et de l'intervalle des ordonnées utilisées : $\Delta_Y = Y_{max} - Y_{min}$.

Dans une vue en perspective, nous venons d'expliquer que l'abscisse finale X'_A dépend de X_A et de Y_A, et que l'ordonnée finale Y'_A dépend de Y_A et de Z_A ; il en va de même pour la mise à l'échelle.

- Un facteur d'échelle commun en X et en Y permet de conserver l'angle α choisi pour la perspective. Ce facteur (noté E_{XY}) est égal au rapport entre :
  - la *largeur utile* de la fenêtre (on tient compte de la marge), et
  - $\Delta_X + \Delta_Y \cdot K \cdot cos(\alpha)$
- Le facteur d'échelle en Z (noté E_Z) est quant à lui égal au rapport entre :
  - la moitié de la *hauteur* de la fenêtre, et
  - $\Delta_Z = z_{max} - z_{min}$

Coordonnées finales à l'écran

Les coordonnées (X_E, Y_E) en perspective et à l'échelle d'un point A de coordonnées initiales (X_A, Y_A, Z_A) s'obtiennent au moyen des formules suivantes :

- $X_E = (X_A + Y_A \cdot K \cdot cos(\alpha)) \cdot E_{XY}$
- $Y_E = Y_A \cdot K \cdot sin(\alpha) \cdot E_{XY} + Z_A \cdot E_Z$

Pour obtenir les coordonnées finales à l'écran, il reste à tenir compte comme dans le cas 2D de la translation des coordonnées $(X_A - X_{min}, Y_A - Y_{min}, Z_A - Z_{min})$, de la marge autour du dessin et de l'inversion de l'axe Y.

Exercice 2 – Dessin fil de fer en 3D

- La classe « `Dessin3D` » du package « `dessin` » permet d'afficher des dessins de type « fil de fer » en 3D. Son principe de fonctionnement est similaire à celui de la classe « `Dessin2D` ».
- Complétez les méthodes suivantes pour que l'affichage final soit correct :
  - `public void miseAJourBornesMinMax(Point3D point)`
  - `public void miseAJourEchelleXY()`
  - `public void miseAJourEchelleZ()`
  - `public int xAffichage(double x, double y)`
  - `public int yAffichage(double y, double z)`
- Note : dans cet exercice, vous ne devez pas conserver les proportions du dessin.
- Consultez la documentation du code pour de plus amples informations sur l'utilité des différentes méthodes. Pensez à utiliser les constantes **K**, **SIN_ALPHA** et **COS_ALPHA**.
- Des exemples de dessins 3D sont préconfigurés dans la classe « DessinFilDefer ». Vérifiez que vos méthodes fonctionnent correctement en exécutant la méthode `main()` de cette classe.

Exercice 2bis – Mise à l'échelle proportionnelle

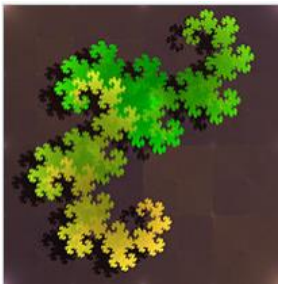
- Modifiez la classe « `Dessin3D` » de manière prendre en considération le paramètre « `proportions` » du constructeur.
- Deux exemples de test avec des cubes sont disponibles dans la classe « DessinFilDefer ». Supprimez les symboles de commentaires pour les exécuter.

3. Graphisme et récursivité

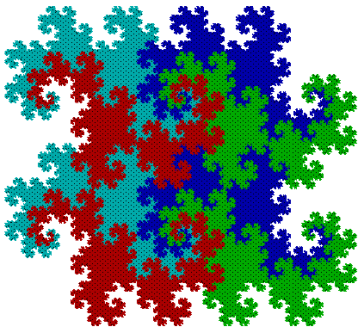
3.1. Introduction

*« La **courbe du dragon** (ou « **fractale du dragon** » ou « **courbe de Heighway** » ou « **dragon de Heighway** ») a été pour la première fois étudiée par les physiciens de la NASA John Heighway, Bruce Banks, et William Harter. Elle a été décrite par Martin Gardner dans sa chronique de jeux mathématiques du Scientific American en 1967. Nombre de ses propriétés ont été publiées par Chandler Davis et Donald Knuth. Elle est apparue dans le roman Jurassic Park de Michael Crichton»*

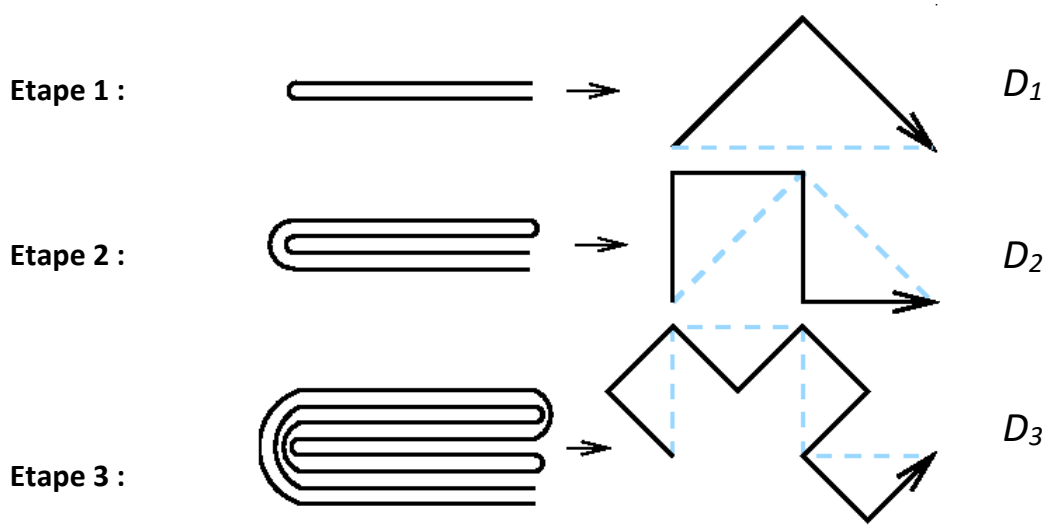
[Source : https://fr.wikipedia.org/wiki/Courbe_du_dragon]



Une des propriétés remarquables de la courbe du dragon est qu'elle permet de paver le plan :



On peut la définir très simplement à partir d'une petite manipulation : si l'on plie une feuille de papier n fois sur elle-même, toujours dans le même sens, et que l'on déplie la feuille en remplaçant les plis par des angles droits, le profil de la feuille donne la courbe du dragon à l'étape n.

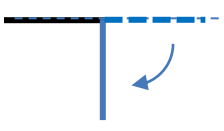


On peut en déduire une construction géométrique simple de la courbe du dragon d'ordre n :

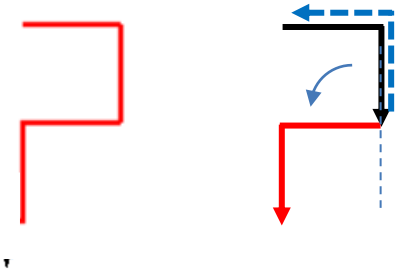
- on dessine une ligne D_0 (de la longueur que l'on veut) ;
- la ligne orientée D_n (« Dragon à l'étape n ») s'obtient à partir de D_{n-1} (« Dragon à l'étape $n-1$ ») en prolongeant cette dernière à partir de son extrémité par une deuxième occurrence de la figure D_{n-1} qui a au préalable subit une rotation autour de cette extrémité de 90°.

Ainsi :

$n = 0$

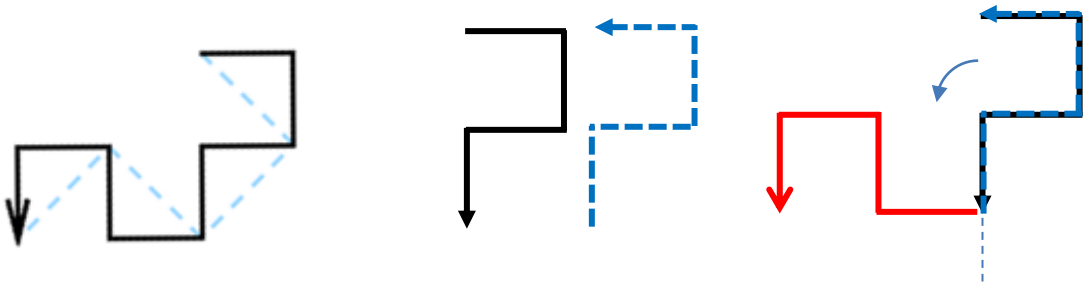
$n = 1$ 

$n = 2$ résultat attendu :



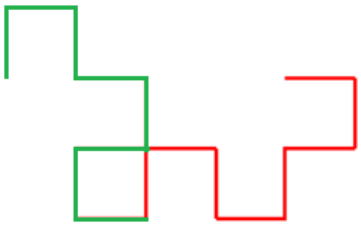
= tourner à droite de 90°,
puis dessiner le deuxième dragon
en miroir par rapport au premier

$n = 3$ résultat attendu :

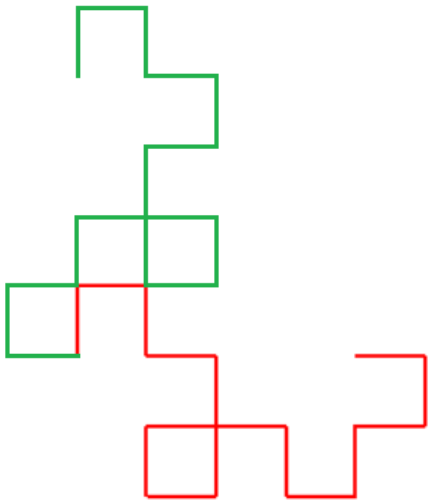


$n = 4$:

= tourner à droite de 90°,
puis dessiner le deuxième dragon
en miroir par rapport au premier



$n = 5$:



Ceci nous permet d'écrire la suite S_n des sens gauche (*G*) ou droite (*D*) des plis à l'étape n :

$n = 1 : S_1 =$ **D**

$n = 2 : S_2 =$ D**D**G

$n = 3 : S_3 =$ DDG**D**DGG

$n = 4 : S_4 =$ DDGDDGG**D**DDGGDGG

⇒ Le deuxième dragon doit être dessiné en miroir,
c'est-à-dire que tous les angles sont dessinés :
- dans l'ordre inverse
- en miroir

Et, de manière générale :

$$S_n = S_{n-1}\mathbf{D}\overline{S_{n-1}}$$

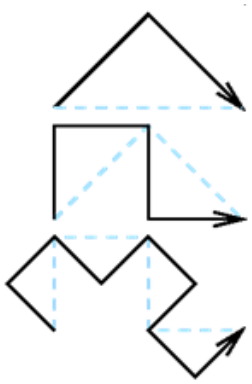
$\overline{S_{n-1}}$ indique que le mot est écrit à l'envers, en remplaçant la lettre D par la lettre G et inversement.

En résumé, pour dessiner la courbe du dragon à l'étape n :

- Si $n = 0$, il faut dessiner un segment de la longueur voulue (= **cas de base**)
- Sinon ($n > 0$), il faut dessiner le premier dragon de l'étape $n - 1$, ensuite tourner de 90°, et dessiner le deuxième dragon de l'étape $n - 1$ dans l'ordre inverse (= **cas de propagation**).

Le calcul des coordonnées du deuxième point définissant le segment à dessiner se base sur le même raisonnement que celui décrit au paragraphe 2.3.

Une autre implémentation (plus simple) peut être réalisée si on regarde différemment les premières figures obtenues.



On peut dessiner récursivement la courbe du dragon à l'étape n, en remplaçant chaque segment du dragon à l'étape $n-1$ par deux segments de même longueur formant entre eux un angle droit.

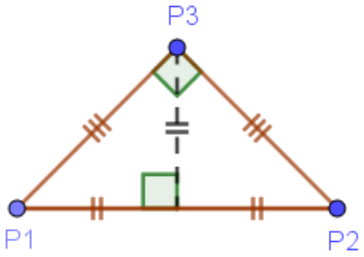
= construire récursivement "l'angle droit noir" sur chaque segment "pointillé bleu"

= remplacer la base d'un triangle rectangle isocèle par ses deux côtés.

Dans cette solution, la fonction récursive s'appuiera sur deux points, au lieu d'un point et d'une orientation.

Les coordonnées du troisième point (P3) intervenant dans cette construction sont celles du sommet du triangle rectangle isocèle dont la base est formée par le segment P1P2, à savoir :

$$x_3 = \frac{x_1 + x_2 - y_2 + y_1}{2}$$
$$y_3 = \frac{y_1 + y_2 + x_2 - x_1}{2}$$



Cette deuxième implémentation peut être résumée de la manière suivante :

pour dessiner la courbe du dragon à l'étape n :

- Si $n = 0$, il faut dessiner un segment de longueur voulue (= **cas de base**)
- Sinon ($n > 0$) , il faut :
  - calculer les coordonnées du point P3,
  - dessiner le premier dragon de l'étape $n-1$ en s'appuyant sur les points P1 et P3, et
  - dessiner le deuxième dragon de l'étape $n-1$ en s'appuyant sur les points P2 et P3, dans cet ordre (dragon inversé) (= **cas de propagation**).

Exercice 3 – Courbe du dragon

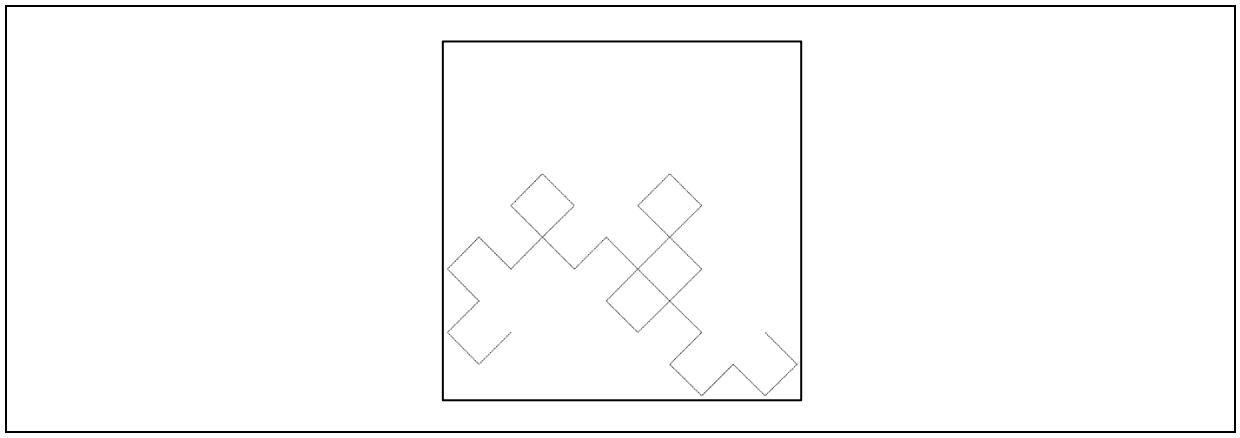
- Dans le fichier « Dragon » du package « `dessin` », complétez la méthode :

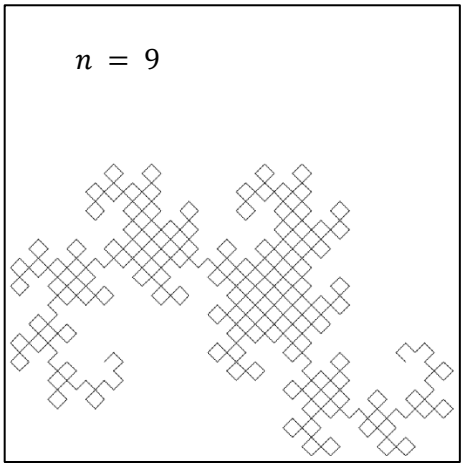
  `public static void` **`dragonRecursion`**`(Dessin2D dessinCourbe, double x1, double y1, double x2, double y2, int ordre)`

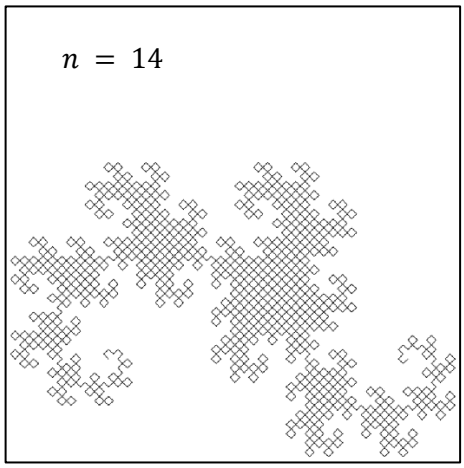
  → Dessine récursivement la courbe du dragon à l'étape n.

  [Suggestion : utilisez la méthode `ajouterLigne()` pour ajouter au dessin un segment défini par deux points. Inspirez-vous de la méthode `creerDessinFilDeFer2D()` de la classe « `DessinFilDeFer` ».]

- Consultez la documentation du code pour de plus amples informations sur les différentes méthodes et sur les classes « `Lignes2D` » et « `Point2D` ».
- Testez que votre méthode fonctionne correctement en exécutant la méthode « `main()` » de la classe.
  Le test de base est configuré avec $n = 5$; ce qui doit conduire au résultat suivant :







Bon travail !