ABHAY Annie
COLLARD Martin
HASENFRATZ Louis
PABOT Justin

Le dessin par séries de Fourier

A/ Cahier des charges

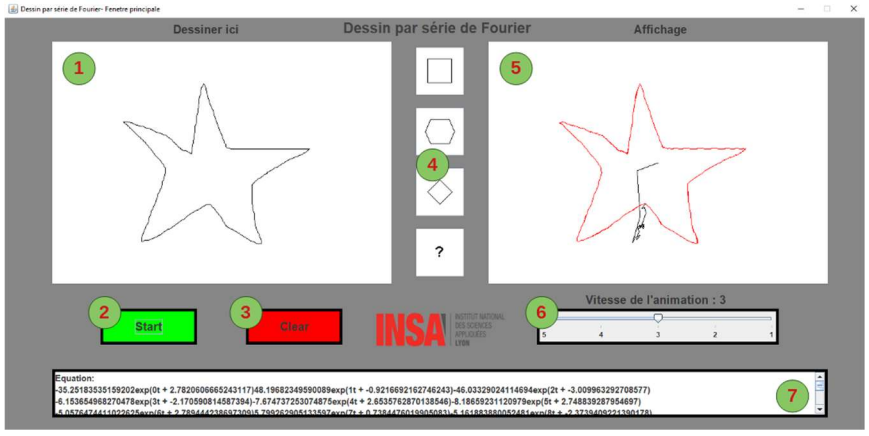
1- Présentation :

Les séries de Fourier permettent de décomposer n'importe quelle fonction en une somme de fonctions sinus. Nous allons utiliser ces séries afin de décomposer un dessin en une série de vecteurs permettant de recomposer celui-ci. Ce procédé permet notamment de compresser une image, ou une musique, avec une perte minime d'information.

2- Cahier des Charges :

On souhaite permettre à l'utilisateur de tracer un dessin (composé d'une seule ligne continue) puis, après appui sur un bouton, de donner la décomposition en série de Fourier de celui-ci avec une précision réglable et de d'afficher le résultat produit par cette décomposition dans un autre cadre afin de comparer les deux dessins. De plus, on souhaite que l'affichage du dessin formé se fasse avec une animation montrant le principe de la décomposition (vecteurs mis bouts à bouts qui tournent). L'utilisateur a aussi accès à des formes prédéfinies qui permettent d'obtenir la décomposition en série de Fourier de polygones communs.

3- Présentation de l'interface :



• Le dessin est acquis dans le cadre (1) : l'utilisateur dessine ici avec la souris. Le dessin doit être effectué en une seule fois : la figure dessinée auparavant s'efface si l'on clique dessus.

• Cliquer sur le bouton «Start» (2) pour démarrer l'acquisition.

• Le dessin est reproduit dans le cadre adjacent (5) ; la reproduction est animée et la vitesse de l'animation se contrôle avec le JSlider (6).

• L'expression en série de Fourier s'affiche sous dans le cadre (JScroll 7).

• Cliquer sur le bouton «Clear» (3) pour arrêter l'animation de droite et effacer ce cadre.

• L'utilisateur peut aussi visualiser des formes prédéfinies en appuyant sur les boutons (4).

B/ Le code

1- Principe de l'algorithme et résolution des problèmes

L'utilisateur dessine dans le panel gauche issu de la classe PanneauEntree (JPanel chargé de l'acquisition) qui enregistre la figure sous forme d'un tableau de complexes. Chaque point est représenté par un nombre complexe, et l'évènement qui permet de les capturer est MouseDragged. Puisque l'algorithme permettant de calculer la transformée de Fourier nécessite un nombre de points qui soit une puissance de 2, on rajoute des points en (0,0) ce qui correspond à la technique dite du « zero padding ». Cela permet d'atteindre une puissance de deux sans impacter la transformée.

Ensuite, dans la classe Window, on décompose ce tableau en utilisant les séries de Fourier : la classe FFT résout ce problème grâce à l'algorithme de Cooley-Turkey. Cet algorithme permet de calculer la transformée de Fourier d'une liste de points en divisant de nombreuses fois la liste de points par 2 et en réalisant une transformée de Fourier classique sur ces plus petits ensembles.

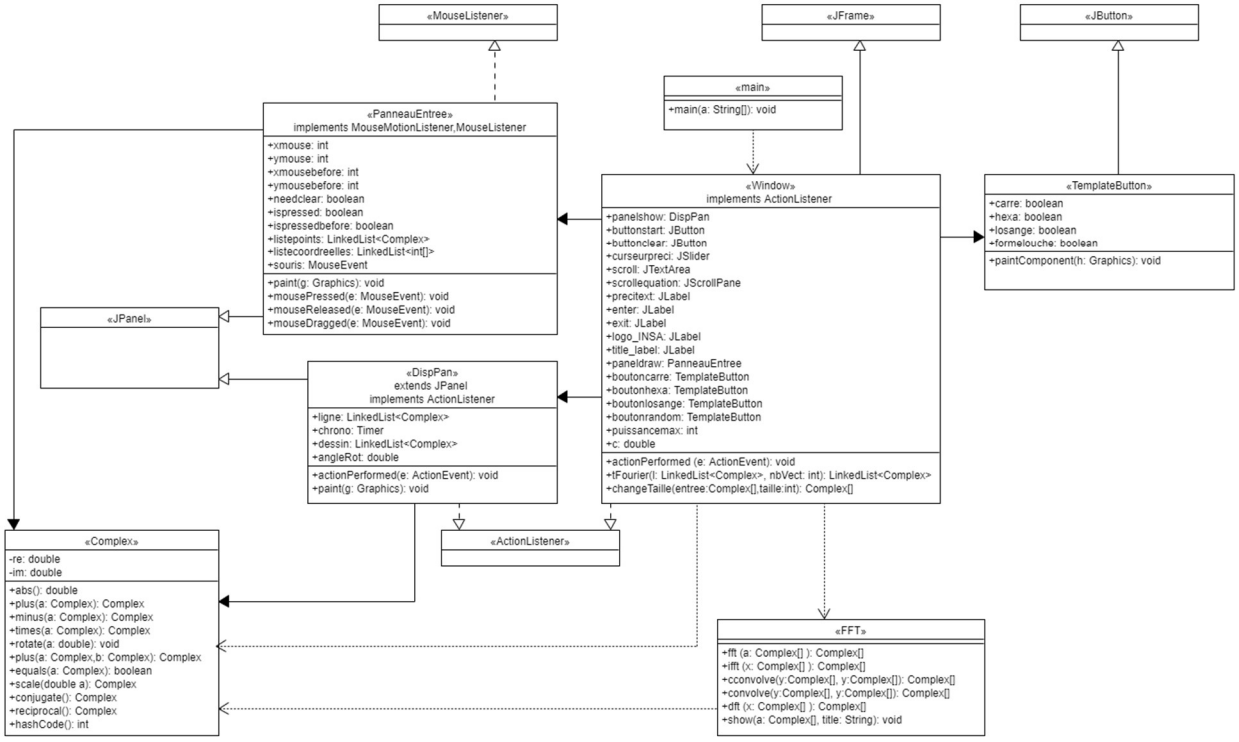
Finalement, le tableau contenant les vecteurs est affiché dans le JScroll et est envoyé au DispPan, le JPanel chargé de l'affichage. Celui-ci affiche l'animation dans le panneau de droite. Cette animation est réalisée grâce à un Timer ; à chaque itération du Timer, on trace les vecteurs de la série de Fourier bout à bout en partant du milieu du DispPan puis on ajoute le point à l'extrémité du vecteur au dessin. Enfin on affiche le dessin, ce qui à l'écran apparaîtra comme une suite de vecteurs qui tracent le dessin, chacun tournant à des vitesses différentes.

2- Suggestions d'amélioration de notre projet, bugs connus

Le principal problème de notre projet est que l'on ne peut malheureusement pas modifier la précision de la transformée. Ceci nous semblait facile, et nous avons réussi à implémenter cette fonctionnalité dans la première version de l'algorithme puisqu'il suffisait de limiter le nombre de vecteurs que l'on envoie au panneau d'affichage. Nous n'avons cependant pas réussi à implémenter de réglage de la précision avec l'algorithme de la librairie utilisée.

Un autre problème est que la précision de la décomposition en série de Fourier (nombre de vecteurs) ne peut dépasser le nombre de points en entrée, sinon le dessin devient aberrant.

3- La structuration des données



Pour l'acquisition des points du tracé, nous avons choisis d'utiliser une LinkedList de nombres complexes puisque l'on ne peut pas savoir à l'avance le nombre de points du tracé, et de nombreux ajouts à cette liste sont à prévoir tout au long du tracé.

Le même raisonnement s'applique pour le tracé dans le panneau d'affichage : l'utilisation de boucles 'for each' permet ainsi d'afficher rapidement tous les vecteurs du tracé. L'efficacité de

l'affichage est très importante car si l'affichage n'a pas le temps de se réaliser entre deux itérations du Timer, nous rencontrerions des problèmes d'affichage.

Cependant, l'algorithme permettant de calculer la transformée de Fourier du tracé prend un tableau de nombres complexes en entrée. Il faut donc transformer la LinkedList des points en entrée en un tableau dont la taille est une puissance de 2 (ce qui définit par ailleurs la précision de notre décomposition).

Nous avons opté pour une structure d'IHM classique ne nécessitant pas de Layout Manager avec une fenêtre principale contenant deux panneaux : un pour l'acquisition du tracé et l'autre pour son affichage. Cette structure nous permet de diviser le travail efficacement et de garder une structure simple et donc facilement compréhensible.

C/ Conduite du projet et sources

1- Carnet de route et échéancier

Semaine 0 : discussion autour du Projet :

- Sujet choisi : Série de Fourier
- Discussion, réalisation du premier cahier des charges

Semaine 1 : début du Projet (en classe) :

- Division des tâches
- Création de la classe graphique Window (IHM) avec des panels et les boutons qui ne déclenchent rien
- Création de la classe du Panel à dessiner dessus (PanneauEntree) qui renvoie un tableau de points (APoint)
- Création de la classe du Panel qui dessine le dessin final (sans l'animation) en testant avec des tableaux de points
- Début de la création de la méthode de décomposition en série de Fourier

Semaine 2 :

- Changement du type des points : Apoint → Complexe
- Finalisation de la classe PanneauEntree (pas de bug) : la classe renvoie un tableau de Complexe
- Continuation sur DispPan, l'animation marche mais il existe des problèmes avec les angles de rotations et le Timer

- Connection entre Window, DispPan et PanneauEntree, activation des boutons Start et clear : le programme donne un bon résultat même si la fonction de Fourier ne marche pas très bien encore.

Semaine 3 :

- Bug : repaint et Layout (Puisque qu'on a deux Panels, ils se superposent sur l'un et l'autre
- Bug : affiche un chaos lorsque le nombre de points du dessin est supérieur aux nombres de vecteurs
- Création des boutons de Templates et sa classe pour pouvoir tester avec des exemples

Semaine 4 :

- Amélioration visuel du Jframe (changement de couleurs, des emplacements, des polices…)
- Réparation des bugs de repaint et de Layout
- Classe Complex modifiée pour optimiser la mémoire (on enlève les attributs de rho et thêta)
- Sélection de la classe FFT sur internet : nous chercherons sûrement à l'employer car elle est plus optimisée que la méthode que nous avions écrite
- Mise en place d'un curseur (Jslider) qui modifie le nombre de vecteurs
- Bug : on ne peut pas dessiner dans le sens anti-horaire

Semaine 5 :

- Modification de la méthode tFourier() qui décompose le tableau de points en série de Fourier, pour utiliser un algorithme plus performant, cette implémentation nécessite de nombreux changements
- Réparation d'un bug où le Panel du PanneauEntree n'apparaît pas avant qu'on clique dessus

Semaine 6 :

- Réalisation du diagramme UML
- Amélioration des Templates
- Emploi de la FFT et réparation des bugs : mettre en place pour que la taille du tableau d'entrée est une puissance de 2 et que les angles de rotation soient en $2pi/2^2$: les dessins en sortie sont enfin d'une précision satisfaisante

Semaine 7 :

- Réalisation du rapport
- Rmélioration du visuel du Jframe : ajout de bords sur les boutons, le logo de L'INSA, la taille des Strings…
- Bug : la vitesse de l'animation à ralentir car le tracé semble instantané

Semaine 8 :

- Réparation de l'animation
- Modifications finales
- Commentaires sur le code

2- Pourcentage d'implication de chaque membre

Justin PABOT : 29%

Louis HASENFRATZ : 26%

Annie ABHAY : 23%

Martin COLLARD : 22%

3- Bibliographie

Robert Sedgewick and Kevin Wayne, FFT.java [en ligne] :
https://introcs.cs.princeton.edu/java/97data/FFT.java.html (vu le 07/04/2020)

Robert Sedgewick and Kevin Wayne, Complex.java [en ligne] :
https://introcs.cs.princeton.edu/java/97data/Complex.java.html (vu le 07/04/2020)

Oracle, how to use sliders [en ligne] :
https://docs.oracle.com/javase/tutorial/uiswing/components/slider.html (vu le 25/03/2020)

ThaiCreate.Com, Java Slider Swing Example [en ligne] :
https://www.thaicreate.com/java/java-gui-swing-jslider.html

RishabhPrabhu, How to convert LinkedList to Array in Java? [en ligne] :
https://www.geeksforgeeks.org/how-to-convert-linkedlist-to-array-in-java/ (vu le 15/04/2020)