Amal ABDALLAH et Jade ANTONI

Enveloppe Convexe dans le plan

Projet de fin de semestre de Programmation C :

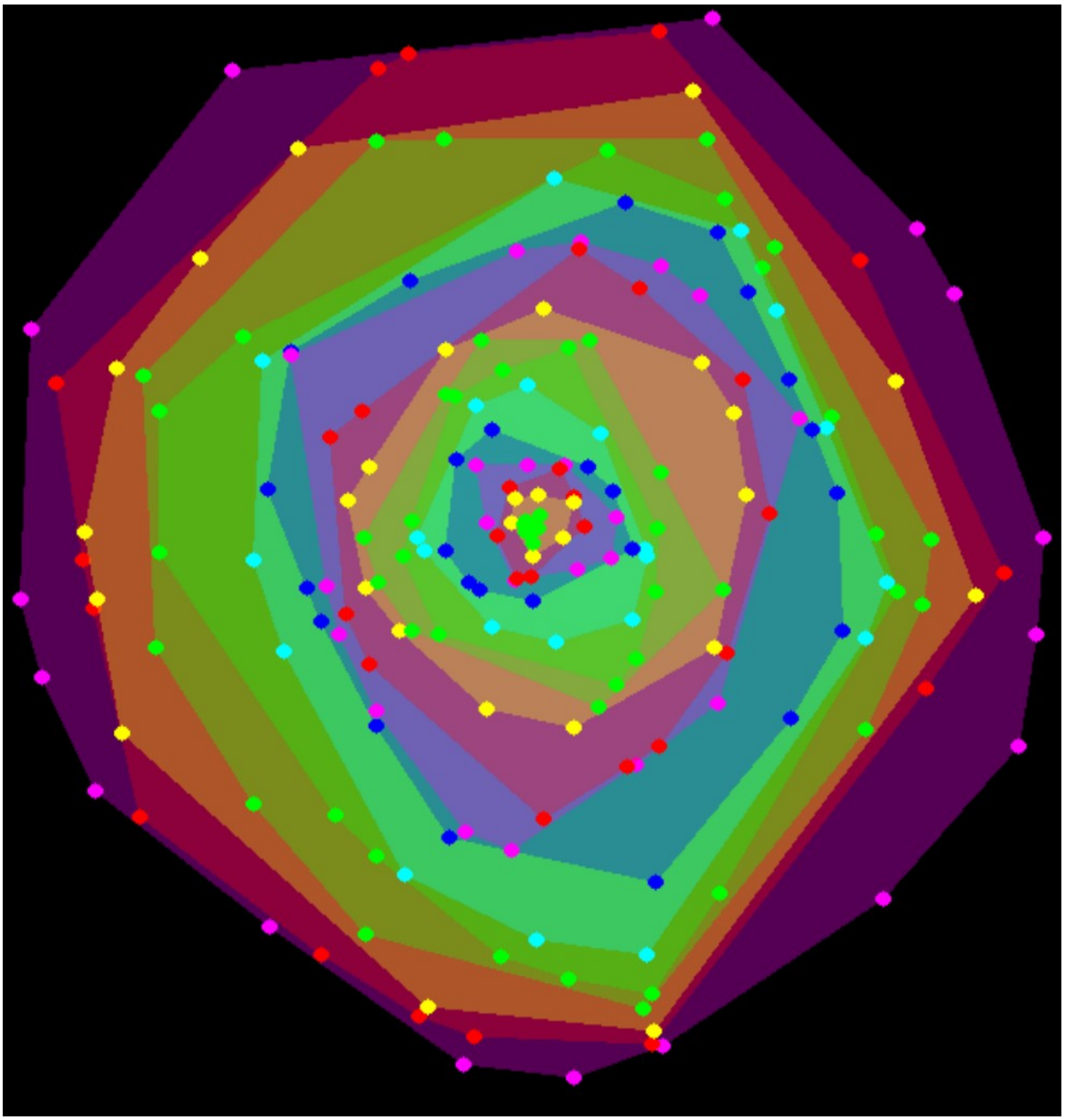
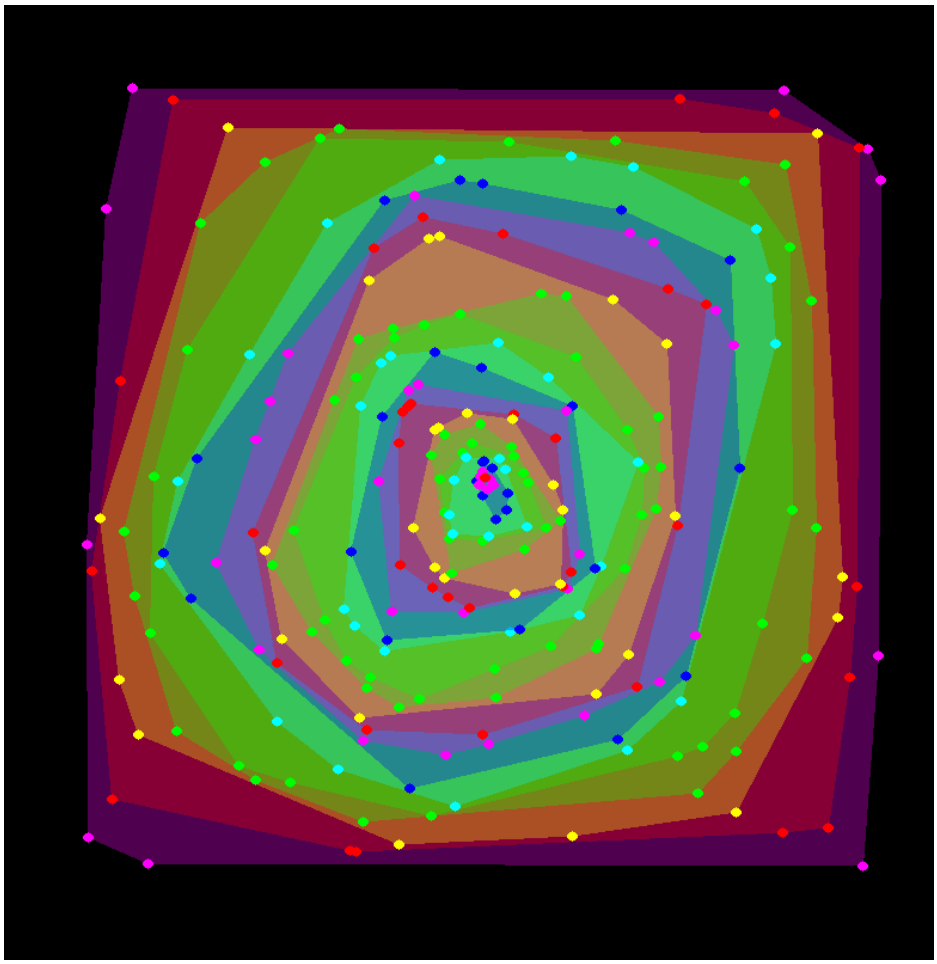


Image tirée du mode de génération aléatoire d’enveloppes convexes emboîtées sous forme de cercle.

Table des matières

Image tirée du mode de génération aléatoire d’enveloppes convexes emboîtées sous forme de carré.



Objectif de ce projet :

Le but de ce projet était, grâce à nos connaissances en mathématiques et en C, d'implémenter un algorithme de détermination d'enveloppe convexe dans un plan.

Pour cela, nous devions utiliser une structure de type liste double chaînée et bouclée, afin de pouvoir parcourir et ajouter, ou bien enlever, des vertex à notre guise.

Manuel d'utilisateur :

- Lancement du programme

Nous avons créé un Makefile, permettant, d'une part, de compiler les différents fichiers séparés, et d'autre part de nettoyer le dossier objet temporaire pour la création de l'exécutable. Ainsi qu'un fichier « run.sh » permettant de combiner « make all », « make clean » et « ./main » en une seule commande.
Il suffira donc de lancer « ./run.sh » pour lancer notre programme.

- Menu

Une fois le programme lancé, le terminal affichera un menu, permettant de choisir le mode d'affichage exact que l'on souhaite.

```
(base) SeifB@SeifPC:/media/sf_Fichier_partag_VM/Jade/L2/Prog_C/Projet-convexe-master_amal$ ./run.sh
rm -f obj/*
clang -Wall -Wfatal-errors -std=c17 -g -o obj/etape1.o -c src/etape1.c
clang -Wall -Wfatal-errors -std=c17 -g -o obj/listepoints.o -c src/listepoints.c
clang -Wall -Wfatal-errors -std=c17 -g -o obj/enveloppeconvex.o -c src/enveloppeconvex.c
clang -Wall -Wfatal-errors -std=c17 -g -o obj/convexemboite.o -c src/convexemboite.c
clang -Wall -Wfatal-errors -std=c17 -g -lMLV -lm -o etape1 obj/etape1.o obj/listepoints.o obj/enveloppeconvex.o obj/convexemboite.o

============================================================
 Bienvenue dans le programme de dessin de l'enveloppe convexe.
============================================================

Veuillez choisir votre mode de génération :
1 : Entrée manuelle
2 : Génération aléatoire
```

1. **Le choix du mode de génération :**

Vous aurez d'abord à choisir entre une génération manuelle, où vous saisirez les points à la main ; ou bien une génération automatique, qui affichera dynamiquement, un par un, chaque point créé aléatoirement

2. **Le choix du type d'enveloppe convexe :**

Vous pourrez alors choisir entre une enveloppe convexe simple ou des enveloppes convexes emboîtées

3. **Le choix de la forme de l'enveloppe :**

Enfin, dans le cas d'une génération automatique, vous aurez à choisir la forme que prendra l'enveloppe, soit en carré, soit en cercle.

Dans le cas d'une génération automatique, une fois la fenêtre ouverte, votre premier clic définira le point de départ de la figure (et donc son centre), et votre deuxième clic définira son rayon, c'est à dire la distance jusqu'à laquelle la figure grandira.

Fonctionnalités du code :

Notre code a été divisé en de nombreux fichiers .c et .h pour améliorer la clarté et l'optimisation du programme.

Les fichiers « enveloppeconvex » regroupent les fonctions essentielles à la partie 1 du programme. Tandis que les fichiers « convexemboite » regroupent les fonctions essentielles à la partie 2 du programme.
Ils contiennent, tout deux, les fonctions principales tel que l'initialisation d'une enveloppe convexe, les fonctions d'allocation de mémoire, d'ajout d'un vertex à l'enveloppe, de parcours, d'insertion de point et de nettoyage, qui permet de supprimer de l'enveloppe, les poins n'en faisant plus partie.

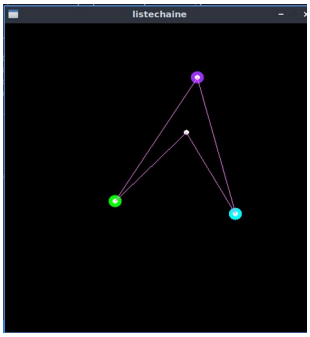
En plus de cela, ils contiennent une fonction essentielle, qui est celle permettant de tester si un point est dans l'enveloppe ou non.

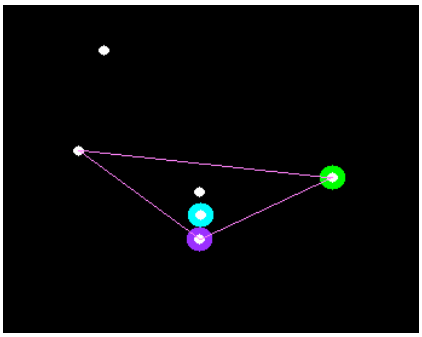
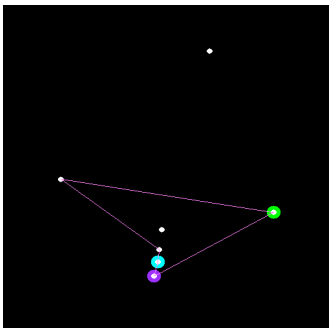
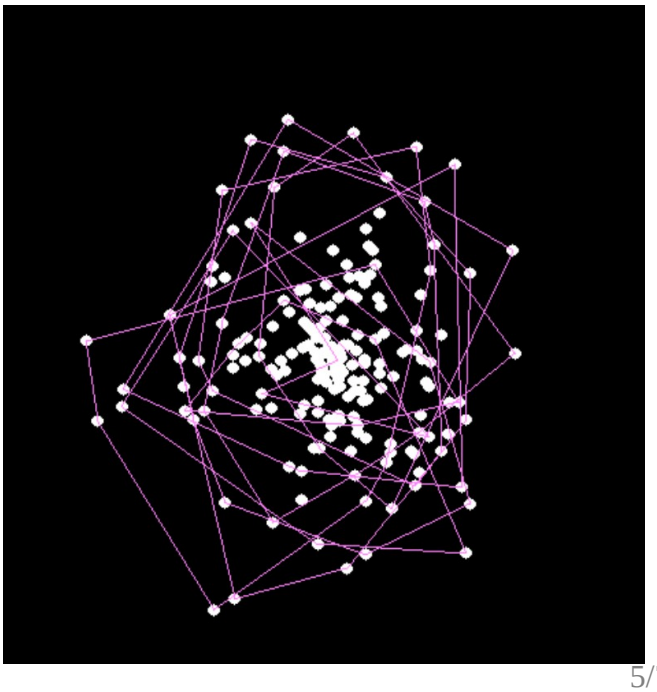
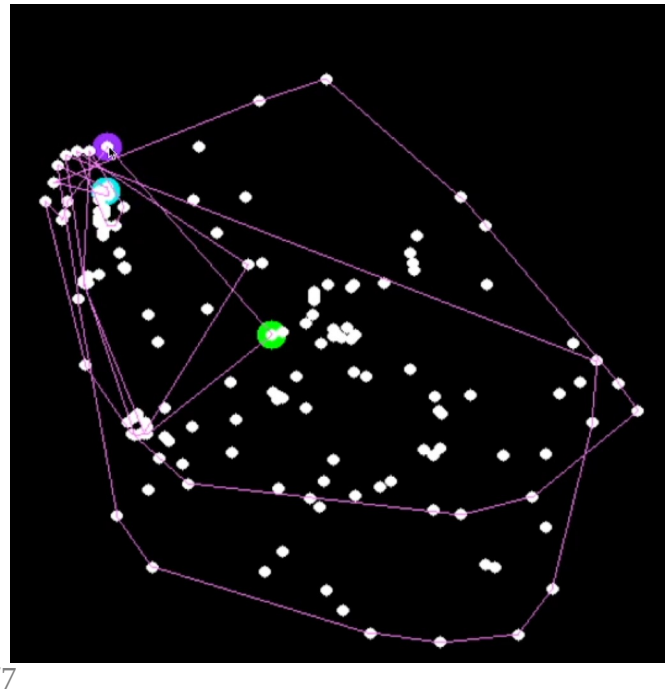
Ensuite, nous avons les fichiers « listepoints » qui contiennent les structures de « point » et de « liste », ainsi que leurs fonctions de base.

Et enfin, nous retrouvons les fichiers « main », qui s'occupent de l'affichage des enveloppes convexes. C'est dans ces fichiers que nous retrouverons les fonctions permettant de dessiner les points, d'implémenter les couleurs, de dessiner la figure selon sa forme ou même d'avoir un menu dans le terminal de lancement.

Difficultés rencontrées :

Nous sommes restés bloqués plusieurs jours sur un problème à l'étape 1.
En effet, nous avions mal géré la polarité de l'enveloppe convexe, et pour cette raison, certains vertex restaient connectés à l'enveloppe alors qu'ils devaient être supprimés, et inversement, certains étaient supprimés alors qu'ils n'auraient pas dû.
Il nous a fallu du temps pour comprendre que cela venait du fait qu'un triangle, censé être indirect, était parfois calculé comme direct, à cause de cette mauvaise polarisation.
Cela donnait donc des figures avec des formes sans aucun sens et des points qui se rajoutaient lorsqu'il ne fallait pas :



Nous avons donc perdu beaucoup de temps sur la première étape, à essayer de régler un problème que nous faisions que déplacer au début.

De plus, nous nous étions assuré qu'aucun point ne sorte de la fenêtre, malheureusement, nous avons découverts, un peu par hasard, que dans le cas des enveloppes convexes emboîtée générées automatiquement, si nous décidions de commencer la figure coller à un bord, nous obtenions une erreur de segmentation dû au nombre de points.
Nous avons donc simplement fini par enlever cette option pour ce type de figures. Néanmoins, elle reste présente pour les enveloppes convexes simples.

Améliorations possibles :

Le menu a prit plus de temps que prévu, au début nous étions parti sur l'idée d'un menu graphique, avec des boutons. Néanmoins, on a vite été dépassé par la gestion de bouton avec la bibliothèque MLV.

En effet, avec notre menu actuel, cela demandait de créer deux rectangles à chaque « étape » du menu, faisant office de bouton. Puis d'insérer du texte à l'intérieur, puis enfin gérer les clics souris de l'utilisateur pour qu'ils soient correctement interprétés.
A chaque étape, il fallait effacer les boutons précédents, redessiner deux nouveaux boutons et recommencer.
Malheureusement, tout ne s'actualisait pas toujours correctement, et on a fini par opter pour un menu intégré au terminal, pour que tout soit clair et fonctionnel.
Un menu graphique pourrait donc tenté d'être implanté.

De plus, certaines de nos fonctions sont longues et complexes et pourraient, très certainement, être raccourcies, divisées en plusieurs autres fonctions ou optimisées.

Il manque également quelques options proposées par l'énoncé, comme les informations statistiques de l'affichage, l'affichage terminal ou encore une option permettant de relancer automatiquement.

Répartition du travail :

- Partie 1 du sujet :

1. **Pour la première étape :**

Jade a conçu les structures de données et les fonctions d'initialisation, d'allocation, de parcours, de test si l'enveloppe est vide, de suppression et de nettoyages. Elle a également documenter chaque fonction et a aidé à régler les problèmes techniques.

Amal a conçu les fonctions d'insertion d'un vertex, d'insertion d'adresse, de test du triangle indirect et de test si le point est dans l'enveloppe convexe.
Il a également identifié le problème de polarité, et l'a résolu.

Les fonctions : dessinePoint(), dessineLstPoint(), dessineConvexe(), choixcouleur(), dessineLstConvex(), test() du main ont été faites à deux.

Les fonctions polyAleaCarre() et polyAleaCercle(), trouverayon() ont été réalisées par Amal.
Les fonctions Choixfigure() et menu() ont été réalisées par Jade.

2. Pour la deuxième étape :

L'affichage dynamique a été réalisé par Amal, l'optimisation et le réglage de bugs par Jade.

- Partie 2 du sujet :

Pour cette partie, nous avons choisi l'option 1 : Enveloppes Convexes Emboîtées.

Elle nous semblait plus intéressante à faire, et l'option de complexité ne nous tentait pas.

Les fonctions ont majoritairement été faites en même temps, à deux.
Amal a testé, trouvé, corriger et résolu certains bugs.
Jade a corrigé les erreurs dans le code et a simplifier des fonctions.

- Autres :

Amal a conçu le Makefile et le run,sh afin que le code soit le plus rapide possible à lancer, il a également réorganiser le code pour qu'il soit le mieux rangé et agréable possible.

Jade a créé les fichiers .h, écrit le rapport et le ReadMe.