Structure Algébrique pour l'informatique
Logiciel de dessin

— Le projet s'effectue en groupe de 4, puisqu'il contient 4 parties. Chaque partie est idéalement gérée par un des membres du groupe.
— Les groupes se forment de préférence dans un même groupe de TD, mais vous pouvez déroger à cette règle afin de former un groupe de 4.
— Vous devez **impérativement** former un groupe de 4 avant le 1er février. L'un de vous m'enverra alors un mail à l'adresse `julien.david@unicaen.fr` avec le nom, le prénom et le numéro d'étudiant des 4 membres. Passé ce délai, je formerai les groupes moi-même. N'hésitez pas à me contactez avant si vous avez du mal à former un groupe complet.
— **tout soupçon de fraude sera sanctionné par un** 0 **ou un conseil de discipline pour toutes les parties**. Faites donc attention à ne pas partager votre code et à le protéger des curieux. Les justifications du type "On en a discuté ensemble" ou "On a travaillé ensemble" ne sont pas valables hors de votre groupe de travail.
— Un programme qui ne compile pas n'est pas corrigé et vaut automatiquement 0. Il vaut mieux faire "moins de choses", tant que cela fonctionne.
— Vous serez noté sur la partie que vous avez faite, il est donc possible que vous n'ayez pas la même note que vos autres binômes.
— La lisibilité du code comptera dans la notation.
— Je dois pouvoir tester rapidement les fonctionnalités de votre programme. Par les tests unitaires et par le programme principal. Si la partie interface graphique ne fonctionne pas, rien ne vous empêche de me faire un dessin à l'exécution du programme principal et d'utiliser les fonctions de sauvegarde et de chargement de fichier.
— Au début de chaque fichier, vous devez écrire le nom de la personne (ou des personnes) ayant travaillé dessus. **Il est interdit de rendre un devoir et de dire "on a tout fait ensemble"**. Vous devez répartir les tâches.

1 Les figures

1.1 Niveau 1

Utilisez l'exemple donné dans le code source du projet pour coder les structures suivantes :

— Segment
— Rectangle
— Triangle
— Cercle

Attention, il faudra penser à ajouter les nouveaux fichiers dans le répertoire `src/figures` et à modifier la variable `FIGURES` dans le fichier `Makefile`

1.2 Niveau 2

— Utilisez les algorithmes de Bresenham vu en cours pour reprogrammer les fonctions d'affichage du segment et du cercle, qui n'utilisent que la fonction `SDL_RenderDrawPoint`.
— créez la structure polygone.

2 Les contrats et les tests unitaires

Attention, pour chaque programme contenant des tests unitaires, vous devrez modifier le fichier `Makefile` afin d'ajouter la compilation de ces nouveaux programmes de tests.

2.1 Niveau 0

Rédigez les contrats des fonctions liées à la structure `liste_figure` et programmez un ensemble de tests unitaires pour l'ensemble de ces fonctions dans un unique programme (vous pouvez vous inspirer de ce qui a été fait pour `couleur_t` et `point_t`).

2.2 Niveau 1

Rédigez les contrats et les tests unitaires de toutes les structures de type `figure`.

2.3 Niveau 2

Rédigez les contrats et les tests unitaires des fonctions de manipulations d'image. Pour le niveau 1 sur la manipulation d'image, on créera **par exemple** des bitmaps entièrement rouge, entièrement bleu et entièrement vert. Pour le niveau 2 on vérifiera par exemple que le fichier créé lors d'une sauvegarde contient bien le bon nombre de lignes, ou que la liste des figures chargées contient bien le bon nombre de figures.

3 Les images

Les fonctions permettant de manipuler des images devront se situer dans le fichier `fichier_image.c`. Vous n'avez pas le droit de modifier la déclaration des fonctions se trouvant dans le fichier `fichier_image.h`.

3.1 Niveau 1

— Écrire une fonction permettant de sauvegarder une liste de figures dans un fichier. Pour réaliser cette fonction, vous êtes autorisés à modifier la définition des structures, afin d'y ajouter les informations (ou des pointeurs de fonctions) que vous jugez nécessaires. Vous êtes libres d'écrire le format de fichier que vous souhaitez pour cette fonction.
— Écrire une fonction qui charge une liste de figures à partir d'un fichier (dans le format que vous avez créé à la question précédente).

3.2 Niveau 2

— Créez une structure de type `figure`, nommée `bitmap_t`, qui contient un tableau à deux dimensions de pixels. La fonction d'affichage de ces bitmaps utilisera uniquement la fonction d'affichage d'un pixel `SDL_RenderDrawPoint`.
— Écrire une fonction `creer_bitmap` qui prend en entrée une liste de figures et dessine ces figures dans une bitmap. Pour réaliser cette fonction, vous êtes autorisé à modifier la définition des structures, afin d'y ajouter les informations (ou des pointeurs de fonctions) que vous jugez nécessaires.
— Écrire une fonction qui permet de sauvegarder un `bitmap_t` au format de fichier `ppm`.
— Écrire une fonction qui permet de charger un `bitmap_t` à partir d'un fichier au format. `ppm`.

4 L'interface graphique

Attention, pour cette partie, il est très facile de faire très rapidement de très grands blocs de code très illisibles. Ainsi, on fera très attention à découper son code en très petites fonctions.

4.1 Niveau 1

— Lire le code et comprendre comment créer un raccourci clavier. Créez une fonction qui affiche la liste des raccourcis clavier existant, appelé au lancement du programme. Cette fonction doit être mise à jour à chaque raccourci que vous créez.
— Créer un raccourci "Point". Lorsque l'utilisateur l'utilise, il peut ensuite cliquer dans la fenêtre et y dessiner des points (on utilisera la fonction `liste_ajouter_fin` pour ajouter des points dans la liste des figures à dessiner).

— Créer un raccourci "Couleur". Lorsque l'utilisateur l'utilise, on lui demande, sur le terminal, de saisir 3 champs (rouge, vert, bleu). La fonction vérifiera que ces champs sont bien compris entre 0 et 255. On crée ensuite une couleur à partir de ces valeurs, qui devient la couleur dans laquelle les utilisateurs pourront désormais dessiner.

4.2 Niveau 2

— Créer des raccourcis permettant de sauvegarder le dessin actuellement à l'écran dans un fichier. La fonction demandera à l'utilisateur de saisir le nom du fichier en sortie sur le terminal.
— Créer des raccourcis permettant de dessiner des segments, des rectangles, des triangles, des cercles.
— Créer des raccourcis permettant de charger un dessin / une image dans un fichier. La fonction appellée supprime toutes la liste des figures actuellement affichées et la remplace par la ou les nouvelles figures.

5 Conseil sur les rôles

— **Les figures** : ce rôle est assez simple du point de vue programmation, mais nécessite une bonne compréhension des algorithmes vus en cours. De plus, ce rôle est central car il est nécessaire que les tâches soient effectuées pour que les autres puissent travailler. Il est **capital** que le niveau 1 soit réalisé **au plus vite**. **Si vous aimez l'algorithmique et SURTOUT si ne travaillez pas à la dernière minute, ce rôle est pour vous**.
— **Les contrats et les tests unitaires** : la personne qui écrit les tests unitaires et les contrats joue un rôle central car c'est elle qui vérifie que le code des autres personnes fonctionne correctement. Elle ne nécessite pas d'être le meilleur programmeur du groupe, mais d'être sérieux et de vérifier que les autres avancent dans le projet. Ne minimisez pas ce rôle, à chaque fois que le programme est modifié par l'un d'entre vous, les tests doivent être relancés avec la commande
make test afin de vérifier que les tests unitaires fonctionnent toujours. **Si les problèmatiques de gestion de projet et de management vous intéressent, ce rôle est pour vous**
— **Les images** : cette partie est à réserver au meilleur développeur du groupe. Cette personne doit être capable de créer ces propres pointeurs de fonctions, de créer son propre format de fichier, de comprendre les principes de représentation des données sur un ordinateur, de modifier le code de quelqu'un d'autre sans créer de nouvelles erreurs. **Si vous voulez vous perfectionner au maximum en programmation, ce rôle est pour vous**
— **L'interface graphique** : la personne qui travaille sur l'interface graphique travaille quasiment indépendamment des autres. Elle ne doit JAMAIS modifier le code des autres. Cette personne doit être capable de

lire de la documentation sur internet pour trouver comment résoudre un problème en `SDL`. C'est donc une personne **autonome** capable de s'autoformer. **Si vous aimez les aspects graphiques de la programmation, et que vous voulez apprendre à vous autoformer, ce rôle est pour vous**