Rapport final

TACHE 1:
Pour cette tache nous avons simplement suivis l'énoncé et écrit le corps des fonctions manquantes. Nous avons utilisé la structure de données Image (déjà fournit).
Aucune réel difficultés.

Ce dossier est formé par plusieurs dossier :
- TEST : Ce dossier contient tout les fichier .c destiner a être des executable et leurs executable sont aussi dans ce dossier apres compilation
- SRC : Ce dossier contient tous les .c dont ont besoin les fichier tests pour complier
- LIBRARY : Ce dossier contient/contiendra (après compilation) la/les librairies statiques dont ont besoins les fichier tests pour compiler
- includes : Ce dossier contiens tous les fichier d'interfaces necessaire à la compilation des SRC et des TEST
- binaries : Ce dossier contient simplement tous les .o cree lors de la compilation
- IMAGE_TEST(_NEG) : Ce sont juste des dossier de test local

COMPLATION :

Lors de la première compilation le make va cree une librairie statique « libimage.a" et va la placer dans le dossier LIBRARY.
Il va ensuite cree les executables test_image et test_lecture_fichier.

EXECUTABLE :

- test_image : Prend en ligne de commande une image .pbm et va crée son négatif qui se nommera pareil que l'image suffixé par _negatif.pbm

- test_lecture_fichier : prend une image en ligne de commande puis vas l'écrire dans le terminal avec des '.' Pour blanc et # pour noir.

TACHE 2:
Pour cette tache nous avons suivis l'annonce et crée un module geom2d avec un .h contentant 2 structures (Point et Vecteur) et un .c content différents fonctions sur des points et des vecteurs
Nous avons créé toutes les fonctions du cour (somme, produit par un scalaire, norme, …).
Nous avons cree deux fichier tests aussi réalisant des tests unitaire.
Aucune réel difficultés.

Voici l'organisation de cette tache.
Ce dossier est formé par plusieurs dossier :
- TEST : Ce dossier contient tout les fichier .c destiner a être des executable et leurs executable sont aussi dans ce dossier apres compilation
- SRC : Ce dossier contient tous les .c dont ont besoin les fichier tests pour complier
- LIB : Ce dossier contient/contiendra (après compilation) la/les librairies statiques dont ont besoins les fichier tests pour compiler
- INCLUDES : Ce dossier contiens tous les fichier d'interfaces necessaire à la compilation des SRC et des TEST
- BIN : Ce dossier contient simplement tous les .o cree lors de la

compilation
- IMAGE_TEST(_NEG) : Ce sont juste des dossier de test local

COMPLATION :

Lors de la première compilation le make va cree une librairie statique
"libimage.a", "libgeom2d.a" et va la placer dans le dossier LIB.
Il va ensuite cree les executables test_geom2D et test_alea_geom2d.

EXECUTABLE :

- test_geom2d : Fais des tests unitaires sur les fonction du module geom2d

- test_alea_geom2d : Fais es tests unitaires mais cette fois de facon aléatoire
  sur le même module

(Il y a un fichier « README.txt dans le dossier correspondant à cette tache).

TACHE 3:
Pour cette tache nous avons cree un module « contour » avec un .h contenant une structure enum (Nord, Sud, Est, Ouest) et une structure robot avec un champ orientation et un champ point (position du robot), et un .c qui contient toutes les fonction nécessaire au calcul du contour d'une image. Ce module se sert du module geom2d.
Et création du module « struct_liste ». Ce module contient trois structure : une liste chainé de point (appelé Liste_Point), associé avec une structure de cellule de la liste chainé (appelé Cellule_Liste_point) et une structure de tableau de Point (appelé Tableau_Point).
Ces structure serviront par la suite a stocké les contours.
Le .c de ce module contient toutes les fonction qui nous serviront par la suite.
Aucune réel difficultés.

Voici l'organisation de cette tache.
ORGANISATION :

Ce dossier est formé par plusieurs dossier :
- TEST : Ce dossier contient tout les fichier .c destiner a être des
  executable et leurs executable sont aussi dans ce dossier apres
  compilation
- SRC : Ce dossier contient tous les .c dont ont besoin les fichier tests
  pour complier
- LIB : Ce dossier contient/contiendra (après compilation) la
  librairies statiques dont ont besoins les fichier tests pour compiler
- INCLUDES : Ce dossier contiens tous les fichier d'interfaces necessaire à
  la compilation des SRC et des TEST
- BIN : Ce dossier contient simplement tous les .o cree lors de la
  compilation
- IMAGE_TEST(_NEG) : Ce sont juste des dossier de test local
- IMAGE_TEST(_NEG)_CONTOUR : Ce sont les contour associé a IMAGE_TEST(_NEG)

COMPLATION :

Lors de la première compilation le make va cree une librairie statique
"libimage.a", "libgeom2d.a", "libstruct_liste.a" et libcontour.a"
et va la placer dans le dossier LIB.
Il va ensuite cree les executables test_contour(_neg) et
test_contour_a_la_volee et test_struct_liste.

EXECUTABLE :

- test_contour(_neg) : Cet executable prend en ligne de commande une image .pbm, il va ensuite la lire, et cree un fichier du même nom mais suffixé par .contour contenant le contour de l'image passé en paramètre au format du sujet (test_contour_neg fais la même chose mais sur les images négative)

- test_contour_a_la_volee : Fais la même chose que test_contour sauf que ce fichier affiche dans le terminal le contour au lieu de la mettre dans un fichier

- test_struct_liste : fais des test unitaires sur le module associés

.SH :
il y a aussi un fichier test_contour(_neg-.sh a la racine du dossier. Ce fichier prend comme argument un dossier contenant des .pbm. Il appelle l'exectable test_contour(_neg) sur toutes les images du dossier passé en paramètre (donc crée un contour par images contenues dans le dossier) et puis les place dans un dossier (qu'il créé si besoin) appllé comme le dossier passé en argument suffixé par _CONTOUR.
Ex : Si on se trouve dans TACHE3 et que l'on fais la commande suivante :
./test_contour.sh ../IMAGE_TESTS
Le programme vas cree tous les contour de toutes les images dans ce dossier et les placé dans ../IMAGE_TESTS_CONTOUR.

Il y a aussi un fichier test_contour_resultat.
Ce fichier calcul le contour de toutes les images de ../IMAGES_TESTS et de ../IMAGES_TESTS_NEG
(cree les négatif si besoin)
et utilise la commande diff pour voir si les contour sont différents
(si ils sont différents --> il y a une erreur car une image et son negatif ont le même contour)
ATTENTION !!! Il faut être dans le dossier TACHE3 pour executer ce script.

(Il y a un fichier « README.txt dans le dossier correspondant à cette tache).

TACHE 4:
Dans cette tache nous avons crée un module « eps ». Ce module n'a pas de structure particulière. Il sert juste a écrire une liste de point sous format eps comme dans le cours.
Aucune réel difficultés.

Voici l'organisation de cette tache.
ORGANISATION :

Ce dossier est formé par plusieurs dossier :
- TEST : Ce dossier contient tout les fichier .c destiner a être des executable et leurs executable sont aussi dans ce dossier apres compilation
- MODULE : Ce dossier contient des sous dossier. Il y a un sous dossier par module et chaque sous dossier contient le .c et le .h associé au module et aussi le .o associé après la compilation.
- LIB : Ce dossier contient/contiendra (après compilation) la/les librairies statiques dont ont besoins les fichier tests pour compiler
- INCLUDES : Ce dossier contiens tous les fichier d'interfaces necessaire à la compilation des TEST (ces fichier sont des liens symbolique vers les fichier .h contenu dans les sous dossier de MODULE).
- BIN : Ce dossier contient simplement tous les .o cree lors de la compilation
- IMAGE_TEST(_FILL(STROKE)_EPS) : Ce sont juste des dossier de test local

COMPLATION :

Lors de la première compilation le make va cree une librairie statique
"libimage.a", "libgeom2d.a" "libstruct_liste", libcontour" et "libeps"
et va la placer dans le dossier LIB.
Il va ensuite cree les executables test_eps_stroke et test_eps_fill.

EXECUTABLE :

- test_eps_stroke : prend en ligne de commande un fichier .pbm et ecrit dans un fichier le contour de l'image associé en mode "stroke"

- test_eps_fill : fais la même chose mais le fais en mode "fill"

.SH :
il y a aussi un fichier test_eps_fill(_stroke).sh a la racine du dossier. Ce fichier prend comme argument un dossier contenant des .pbm. Il appelle l'executable test_eps_fill(_stroke) sur toutes les images du dossier passé en paramètre (donc crée une image .eps par images contenues dans le dossier) et puis les place dans un dossier
(qu'il créé si besoin) appllé comme le dossier passé en argument suffixé par _FILL(_STROKE)_EPS.
Ex : Si on se trouve dans TACHE4 et que l'on fais la commande suivante :
./test_eps_fill.sh ../IMAGE_TESTS
Le programme vas cree toutes les images .eps de toutes les images .pbm dans ce dossier et les placé dans ../IMAGE_TESTS_FILL_EPS.

ATTENTION !!! Il faut être dans le dossier TACHE4 pour executer ce script.

(Il y a un fichier « README.txt dans le dossier correspondant à cette tache).

TACHE 5:
Dans cette tache nous n'avons pas créé de nouveau module ou de nouvelle structure, nous avons seulement ajouter de nouvelles fonction aux modules déjà existant (cf. (voir) suivis de projet). Aucune réel difficultés.

Voici l'organisation de cette tache.
ORGANISATION :

ORGANISATION :

Ce dossier est formé par plusieurs dossier :
- TEST : Ce dossier contient tout les fichier .c destiner a être des executable et leurs executable sont aussi dans ce dossier apres compilation
- MODULE : Ce dossier contient des sous dossier. Il y a un sous dossier par module et chaque sous dossier contient le .c et le .h associé au module et aussi le .o associé après la compilation.
- LIB : Ce dossier contient/contiendra (après compilation) la/les librairies statiques dont ont besoins les fichier tests pour compiler
- INCLUDES : Ce dossier contiens tous les fichier d'interfaces necessaire à la compilation des TEST (ces fichier sont des liens symbolique vers les fichier .h contenu dans les sous dossier de MODULE).
- BIN : Ce dossier contient simplement tous les .o cree lors de la compilation
- IMAGE_TEST : Ce sont juste des dossier de test local
- IMAGE_TEST_CONTOUR : Les contours des images de IMAGE_TEST

- IMAGE_TEST_FILL_EPS : les même images que IMAGE_TEST mais sous format .eps et en mode fill
- MES_IMAGES : Ce dossier contiens les deux images que j'i crée
(il y image toutes blanche pour tester si le fais de ne pas avoir de contour remontait bien l'erreur)

COMPLATION :

Lors de la première compilation le make va cree une librairie statique
"libimage.a", "libgeom2d.a" "libstruct_liste", libcontour" et "libeps"
et va la placer dans le dossier LIB.
Il va ensuite cree les executables test_contour_masque et test_eps_multiple.

EXECUTABLE :

- test_contour_masque : Cet executable prend en ligne de commande une image .pbm, il va ensuite la lire, et cree un fichier du même nom mais suffixé par .contour.txt contenant le contour de l'image passé en paramètre au format du sujet

- test_eps_multiple : prend en ligne de commande un fichier .pbm et son fichier contour associé et écrit dans un fichier le contour de l'image associé en mode "fill"

.SH :
il y a aussi un fichier contour.sh a la racine du dossier. Ce fichier prend
comme argument un dossier contenant des .pbm. Il appelle l'exectable
test_contour_masque sur toutes les images du dossier passé en paramètre
(donc crée un contour par images contenues dans le dossier) et puis les place dans un dossier
(qu'il créé si besoin) appllé comme le dossier passé en argument suffixé par
_CONTOUR.
Ex : Si on se trouve dans TACHE5 et que l'on fais la commande suivante :
./contour.sh IMAGE_TESTS
Le programme vas cree tous les contour de toutes les images dans ce dossier et
les placé dans IMAGE_TESTS_CONTOUR.

il y a aussi un fichier test_eps_fill.sh a la racine du dossier. Ce fichier prend
comme argument un dossier contenant des .pbm. Il appelle l'executable
test_eps_multiple sur toutes les images du dossier passé en paramètre
(donc crée une image .eps par images contenues dans le dossier) et puis les place dans un dossier
(qu'il créé si besoin) appllé comme le dossier passé en argument suffixé par
_FILL(_STROKE)_EPS.
Ex : Si on se trouve dans TACHE4 et que l'on fais la commande suivante :
./test_eps_fill.sh IMAGE_TESTS
Le programme vas cree toutes les images .eps de toutes les images .pbm dans ce dossier et
les placé dans ../IMAGE_TESTS_FILL_EPS.

(Il y a un fichier « README.txt dans le dossier correspondant à cette tache).

TACHE 6:
Pour cette tache nous avons complété les modules déjà existant en rajoutant des fonction (voir suivis de projet) et nous avons cree un module appelé « simplification ». Ce module ne contient aucune nouvelle structure. Il y a justes les fonctions nécessaire à la simplification de contours par segments comme dans le cours (cf. Suivis de projets).
Le choix de calcul a la volée à étais fais dans cette tache et dans tous le projets.
Calcul a la volé signifie que pour chaque contour de l’image ont fais la simplification puis on écrit le résultat dans le fichier résultat et enfin on passe au contour suivant.

Ce choix permet de ne pas créer de structure supplémentaire qui serai des liste de liste chainé et qui rendrai le programme beaucoup plus complexe.
Aucune réel difficultés.

Voici l'organisation de cette tache.
ORGANISATION :

Ce dossier est formé par plusieurs dossier :
- TEST : Ce dossier contient tout les fichier .c destiner a être des executable et leurs executable sont aussi dans ce dossier apres compilation
- MODULE : Ce dossier contient des sous dossier. Il y a un sous dossier par module et chaque sous dossier contient le .c et le .h associé au module et aussi le .o associé après la compilation.
- LIB : Ce dossier contient/contiendra (après compilation) la/les librairies statiques dont ont besoins les fichier tests pour compiler
- INCLUDES : Ce dossier contiens tous les fichier d'interfaces necessaire à la compilation des TEST (ces fichier sont des liens symbolique vers les fichier .h contenu dans les sous dossier de MODULE).
- BIN : Ce dossier contient simplement tous les .o cree lors de la compilation
- IMAGE_TEST : Ce sont juste des dossier de test local
- IMAGE_TEST_CONTOUR_COMPLET : Les contours des images de IMAGE_TEST avant simplifaction par segments.
- IMAGE_TEST_CONTOUR_SIMPLE : Les contours des images de IMAGE_TEST après simplifaction par segments.
- IMAGE_TEST_FILL_EPS_COMPLET : les même images que IMAGE_TEST mais sous format .eps et en mode fill avant simplifacation par segments
- IMAGE_TEST_FILL_EPS_SIMPLE : les même images que IMAGE_TEST mais sous format .eps et en mode fill apres simplifacation par segments

COMPLATION :

Lors de la première compilation le make va cree une librairie statique "libimage.a", "libgeom2d.a" "libstruct_liste", libcontour", "libeps" et "libsimplifaction" et va la placer dans le dossier LIB.
Il va ensuite cree les executables test_simplifaction_segment et test_distance

EXECUTABLE :

- test_distance : Ce fichier test NE PREND PAS d'argument en ligne de commande mais demande a l'utilisateur de saisir 3 points (P puis A et en fin B) et calcul et affiche la distance entre P et le segement [AB]

- test_simplifactaion_segment : Ce fichier prend en argument une image .pbm et un float qui sera "d"
  Il va ensuite calculer sont contour comme dans la tache 6 et le mettre dans le fichier du meme nom suffixé par .contour.txt
  Puis il va simplifier ce contour et mettre le nouveau contour dans un autre fichier (cette fois suffixé par .contour_simple.txt)
  Et enfin il va calculer a partir du contour et du contour simplifier lews images eps associés

.SH :
- simplifactaion_segemnt.sh :
  Ce fichier prend en argument un dossier contenant au moins un fichier .pbm
  Il calcul ensuite ses contours les simplifié et ensuite cree une image .eps de l'image (une a partir du contour originale et une a partir du contour simplifié
  Les différents fichier sont ensuite placé dans les dossier correspondant
  - IMAGES_TEST_CONTOUR_COMPLET : les contours complet des images
  - IMAGES_TEST_CONTOUR_SIMPLE : les contour simplifié des images

- IMAGES_TEST_FILL_EPS_COMPLET : les images eps cree a partir des contour complet
- IMAGES_TEST_FILL_EPS_SIMPLE : les images eps cree a partir des contour simple

(Il y a un fichier « README.txt dans le dossier correspondant à cette tache).

TACHE 7:
Dans cette tache nous avons de nouveau complété certes modules déjà existant (cf. Suivis de projet). Nous avons cree un nouveau module nommé « bezier » qui contient deux structure : Bezier2 et Bezier3 qui se servent de la structure Point pour avoir des type représentant respectivement une courbe de bezier de degrés 2 et une courbe de bezier de degrés 3.
Le .c contient toutes le fonctions nécessaire aux calculs des courbes de Béziers.
Aucune réel difficultés.

Voici l'organisation de cette tache.
ORGANISATION :

Ce dossier est formé par plusieurs dossier :
- TEST : Ce dossier contient tout les fichier .c destiner a être des executable et leurs executable sont aussi dans ce dossier apres compilation
- MODULE : Ce dossier contient des sous dossier. Il y a un sous dossier par module et chaque sous dossier contient le .c et le .h associé au module et aussi le .o associé après la compilation.
- LIB : Ce dossier contient/contiendra (après compilation) la/les librairies statiques dont ont besoins les fichier tests pour compiler
- INCLUDES : Ce dossier contiens tous les fichier d'interfaces necessaire à la compilation des TEST (ces fichier sont des liens symbolique vers les fichier .h contenu dans les sous dossier de MODULE).
- BIN : Ce dossier contient simplement tous les .o cree lors de la compilation
- IMAGE_TEST : Ce sont juste des dossier de test local
- IMAGE_TEST_CONTOUR_COMPLET : Les contours des images de IMAGE_TEST avant simplifaction
- IMAGE_TEST_CONTOUR_SIMPLE : Les contours des images de IMAGE_TEST après simplifaction par courbe de bezier de degres 2
- IMAGE_TEST_CONTOUR_SIMPLE_3 : Les contours des images de IMAGE_TEST après simplifaction par courbe de bezier de degres 3
- IMAGE_TEST_FILL_EPS_COMPLET : les même images que IMAGE_TEST mais sous format .eps et en mode fill avant simplifacation par segments
- IMAGE_TEST_FILL_EPS_SIMPLE : les même images que IMAGE_TEST mais sous format .eps et en mode fill apres simplifacation par courbe de bezier de dregre 2
- IMAGE_TEST_FILL_EPS_SIMPLE_3 : les même images que IMAGE_TEST mais sous format .eps et en mode fill apres simplifacation par courbe de bezier de dregre 3

COMPLATION :

Lors de la première compilation le make va cree une librairie statique
"libimage.a", "libgeom2d.a" "libstruct_liste", libcontour",
"libeps","libsimplifaction" et "libbezier"
et va la placer dans le dossier LIB.
Il va ensuite cree les executables test_bezier test_bezier3, test_simplifiation_bezier et test_simplifaction_bezier3

EXECUTABLE :

- test_bezier et test_bezier3 : Ces deux fichier fonct des test unitaire sur

les focntions du module bezier pour les courbes de bezier de degres 2 et 3
respectivement

- test_simplifactaion_bezier(/3) :
Ces fichier prend en argument une image .pbm et un float qui sera "d"
Il va ensuite calculer sont contour comme dans la tache 6 et le mettre dans
le fichier du meme nom suffixé par .contour.txt
Puis il va simplifier ce contour grace au courbe de bezier de degres 2(/3)
fichier (cette fois suffixé par .contour_bezier_simple.txt)
Et enfin il va calculer a partir du contour et du contour simplifier les images eps associés

.SH :
- simplifactaion_bezier2.sh et simplification_bezier3.sh :
Ce fichier prend en argument un dossier contenant au moins un fichier .pbm
Il calcul ensuite ses contours les simplifié et ensuite cree une image .eps
de l'image (une a partir du contour originale et une a partir du contour simplifié
Les différents fichier sont ensuite placé dans les dossier correspondant
- IMAGES_TEST_CONTOUR_COMPLET : les contours complet des images
- IMAGES_TEST_CONTOUR_SIMPLE : les contour simplifié des images
- IMAGES_TEST_FILL_EPS_COMPLET : les images eps cree a partir des contour complet
- IMAGES_TEST_FILL_EPS_SIMPLE : les images eps cree a partir des contour simple

(Il y a un fichier « README.txt dans le dossier correspondant à cette tache).

TACHE 8:
Pour cette tache nous n'avons pas cree de dossier. Nous avons juste utilisé le code de la tache 7 et ajouter la commande « time » pour avoir le temps d 'éxecution.
Donc l'organisation est la même que la tache 7.

		s3	s4	s5	s6	s7	s8	s9	s10	s11	s12	s13	s14	s15	s16	s17	s18	s19
Tâche 1	Manipulation d'image																	
Tâche 2	Outils de calcul géométrique en 2D																	
Tâche 3	Extraction du contour externe																	
Tâche 4	Sortie PostScript																	
Tâche 5	Extraction de plusieurs contours																	
Tâche 6	Simplification de contour par segments																	
Tâche 7	Simplification de contour par courbe de Bézier																	
Tâche 8	Tests & comparatif des simplifications																	
	Tests et intégration																	
	Préparation démo																	
	Soutenance finale							CANCES HIV		NE DES PA						ICES PRINT		

Réf.	Date	Problème/Information	Action/Décision	Date de réalisation prévue	Date de réalisation réelle	Etat
	22/12/23	**TACHE 1**				
1	22/12/23	Importation de toutes les images tests, du contenu de TACHE1 dans TACHE1 Et creation de tous dossier TACHE1, 2...		22/12/23	22/12/23	terminé
2	23/12/23	écritures des fonction ecrire_image et négatif_images partie (A).		23/12/23	23/12/23	terminé
	23/12/23	**TEST test_lecture_fichier**				
3	23/12/23	Ecriture d'un fichier test_lecture_fichier qui test si un fichier a le bon format Sert a faire les tests de robustesse de la fonction lire_fichier_image Et création d'un dossier IMAGE_TESTS_ERR Qui contient des fichier images contenant des erreurs	Si un fichier contient P1 au début et ne reviens pas a la ligne pour les dimension alors il n'y a pas d'erreur mais les dimension ne sont pas bonnes. (Voir exemple ligne 2) Donc modification de la fonction entete_fichier_pbm pour vérifier que après la P1 il y a un \n (il faut pas qu'il y ai des espace après P1) ---> modif ligne 145	23/12/23	fin des tests 25/12/2023	terminé
4	23/12/23	Suite des tests du points 3	si il n'y a pas de séparateur entre les dimensions alors pas d'erreur mais pas les bonnes dimensions voir exemple ligne 3	23/12/23	23/12/23	terminé
5	23/12/23	REF 4	ajout de la fonction « verif_sep » qui vérifie si il y a au moins un séparateur (' ' ou '\n') (renvoie 0 si il n'y en a pas et 1 si il y en a au moins un). Ajout de la vérification du fais que y < H et si c'est le cas alors pas assez de Pixel.	23/12/23	23/12/23	terminé
6	23/12/23	REF 4	Ajout dans lire_fichier_image de la vérification que il n'y a pas plus de pixel qu'annoncé par les dimension	24/12/23	25/12/23	terminé
7	25/12/23	REF 3 (idée) Si on a un pixel different de 0 ou 1 ou ' ' ou '\n' soit on stop le procesus soit on met un pixel blanc	C'est une idée ---> question à poser	?	?	en attente
	25/12/23	**TEST ecrire_image et negatif_image**				
8	25/12/23	Création d'une fonction ecrire_fichier_image qui ecrit une image I en fichier .pbm		25/12/2023	25/12/2023	terminé
9	25/12/23	Création d'une fonction oracle verif_neg qui prend une image I et son négatif et verifie qui le negatif et bien le négatif de I	c'est une idee ---> pas sur que se soit vraiment utile	?		en attente
10	25/12/23	Modification test_image.c	Pour ecrire dans un fichier le negatif de l'image passer en ligne de commande	25/12/2023	25/12/2023	terminé
11	25/12/2023	Creation de test_image.sh	Ce fichier prend en ligne de commande un dossier contenant des .pbm et fais leur négatif et les met dans un dossier du meme nom suffixe par _NEG (ce dossier se trouve dans le meme sous_dossier que le dossier passer en argument)	25/12/2023	25/12/2023	terminé
12	25/12/23	modification de types_macros	Pour ajouer de la couleur au printf	25/12/23	25/12/23	terminé
13	25/12/23	creation image "carre_tout_blanc.pbm"	tests concluant	25/12/23	25/12/23	terminé
14	25/12/23	CONCLUSION TACHE 1	Tous les tests sont concluant que se soit les tests fonctionnels ou de robustesse	25/12/23	25/12/23	terminé
		TACHE 2				
15	25/12/2023	Création du module geom2d, copie et modification du makefile et copier coller de type_macros.h avec modification en ajoutant le type coordonnee	Ecriture des fonction du cours	25/12/23	25/12/23	terminé
16	25/12/23	idee --> implémentation des fonctionnalité translation, homothétie, rotation	A implémenté en fonction du temps disponible	?		en attente
	25/12/23	**TEST du module geom2d**				
17	25/12/23	test grace a de aléatoire et aussi des tests unitaires	tester une fois chaque fonction grâce aux fonction rand() et srand(time(NULL) (fichier : test_alea_geom2d) et faires des tests unitaires grâce a assert (fichier : test_geom2d)	25/12/23	25/12/23	terminé
		TACHE3				
18	3/1/24	Importation de tout les fichier contenus dans le dossier de l'UE				
19	3/1/24	creation du module struct_liste et de test_struct_liste.c	le module struct liste est essentiellment le meme code que celui dans exemple.c a Le fichier test_struct_liste.c sert à faire les test sur le module associé	3/1/12	5/1/12	terminé
20	5/1/24	importation du module image de la tache 1 du module geom2d de la tahce 2 et de types_macros				terminé
21	5/1/24	creation d'un module contour	Ce module sert a calculer le contoure d'une image Il y a deux fonction calcul_contour : calcul_contour_a_la_vole où memoriser position consiste à écrire la position à l'écran. et calcul_contour qui renvoie une liste chainé contenant la liste des point du contour	5/1/24	5/1/24	terminé
22	5/12/24	creation de test_contour et test_contour_a_la_volee	Ces fichiers utilise respectivement les fonction calcul_contour et calcul_contour_a_la_volee sur un fichier .pbm passé en ligne de commande Le fichier calcul_contour cree un fichier suffixé par .contour qui contient le contour de l'image dans le format spécifié dans le cours	5/12/24	5/12/24	terminé
23	5/1/24	creation de fichiers .pbm	Le bu est de tester les "4 possibilité" (vue en cours de la fonction nouvelle_orientation de l'algo du cours) Et ceci sur les 4 direction possibles	5/1/24		
24	5/1/24	creation de contours.sh	Ce fichier prend en argument un dossier contenant des .pbm et calcul leurs contour Puis met les contours cree dans un dossier du même nom que le paramètre suffixé par _CONTOUR	5/1/24	5/12/24	terminé
25	5/12/24	creation de contour_neg	Cette fonction calcul aussi le contour d'une images mais cette fois ci l'image et sous sa forme négatives (il y a une nouvelle fonction car elle doit avoir un pixel blanc a sa droite et un pixel noir a sa gauche)	5/12/24	5/12/24	terminé
26	5/12/24	creation de contour_neg.sh	Fais la même chose que contour.sh (REF 24) mais sur les images négatives	5/12/24	5/12/24	terminé
27	5/12/24	creation de test_contour_resultat.sh	Ce fichier calcul le contour de toutest les images de IMAGES_TESTS et de IMAGES_TESTS_NEG (cree les négatif si besoin) et utilise la commande diff pour voir si les contour sont différents (si ils sont différents --> il y a une erreur car une image et son negatif ont le même contour)	5/12/24	5/12/24	terminé
		TACHE 4				
28	23/01/24	copie de tous les module de la tache3	Copie des module image, contour, geom2d struct_liste	23/01/24	23/01/24	terminé
29	23/01/24	creation du module eps	Ce module va servir a cree les fichier eps depuis un contour calculer comme dans la tache 3			
30	23/01/24	creation des executable	On fais un executable pour la commande stroke et un pour fill Le nom des executable : test_esp_fill et test_eps_stroke	23/1/24	23/1/24	terminé
31	23/01/24	creation des .sh associé	On cree test_eps_stroke.sh et test_eps_fill.sh qui prennent en ligne de commande un dossier avec au moins une image .pbm ensuite ils ont executé respectivement test_eps_stroke(fill) pour cree l'image .eps associé et enfin placer les images cree dans le dossier du même nom suffixé par _STROKE(/FILL)_EPS	23/1/24	23/01/24	terminé
32	23/01/24	Test	apres lancement des deux .sh sur IMAGE_TEST (de la tache, le dossier contenu dasn le dossier TACHE4) et visualisation et comparaison avec les .pbm associé on peut être plutôt sur que le code est bon.	23/01/24	23/01/24	terminé
		TACHE 5				
33	25/01/24	creation de le fonction calcul_contour_masque	Cette fonction est la même que calcul_contour de la tache 3 mais elle prend en plus deux argument : - la position initialisé car elle dépend de l'image masque - ET l'image masque pour pouvoir la modifier celon l'algorithme du cours	25/01/24	25/01/24	terminé
34	25/01/24	creation de test_contour_masque.c	Ce fichier est déstiné a etre l'executable pour pouvoir tester la tache. Il y a deux fonction en plus du main : - La première est ecrire_Contour_fichier qui ecrit un contour passé en paramètre dans un fichier passé en paramètre - la seconde est calcul_contour_multiple qui parcour l'image masque est fais un contour pour chaque pixel noir de cette image (et utile la premiere fonction pour afficher ces contour dans le fichier correspondant)	25/01/24	25/01/24	terminé
35	25/01/24	ajout de la fonction cree_image_eps_multiple	Cette fonction fais comme cree_image_eps mais avec une image contenant plusieurs contours Cette fonction doit prendre en argument le nom du fichier contour	25/01/24	25/01/24	terminé
36	25/01/24	creation de test_eps_multiple.c	Ce fichier est destiné à être un executable qui prend en ligne de commande une image .pbm et son fichier contour multiple qui contion tous les contours de cette images.	25/01/24	25/01/24	terminé
37	25/01/24	creation de contour.sh et test_fill_eps.sh	C'est deux scripts font essentiellement la meme chose que dans la tache 3 et 4 (cf ref 24 et 31) mais pour des contour multiple	25/01/24	25/01/24	terminé
38	25/01/24	ajout du README.txt	A lire	25/01/24	25/01/24	terminé
		TACHE 6				
39	18/03/24	Modification de contour.c	On ajoute juste les focntion qui était dans test_contour_masque.c dans le fichier contour.c Cela permetra de calculer tous les contour de l'image et de les ecrire sans devoir copier es fonctions dans le fichier test	18/03/24	18/0324	terminé
40	18/03/24	Creation de la fonction calcul_distance_point_segment	Cette fonction prend trois point P, A et B et calcul la distance entre P et le segment [AB]	18/03/24	18/0324	terminé
41	18/03/24	Creation du fichier test_distance.c	Ce fichier est le fichier test qui demande a l'utilisateur 3 points (P, A et B) puis appelle la fonction de la ref 40 et affiche la distance entre le point P et le segment S = [AB]	18/03/24	18/03/24	terminé
42	18/03/24	Création du compte rendu de la partie 1 de la tache		18/03/24	18/03/24	terminé
43	21/03/24	Creation du module simplification	Ce module est destiné au fonction pour la simplification de contour	21/03/24	21/03/24	terminé
44	21/03/24	Creation des fonctions simplification_douglas_peucker et simplification_segment	La fonction simplification_douglas_peucker suit exactement l'algo donné en cours et a la meme fonction Et la fonction simplification_segment prend en argument : - le nom d'un fichier contour n1, le nom d'un fichier de sortie n2, un reel d Puis calcul grace a l'algorithme de Douglas Peucker la simplification de n1 et met le resultat dans n2. Cette fonction fais les calcul a la volé : c'est a dire pour chacun des contour de n1 elle va calculer sa simplification puis le mettre dans n2 avant de passer au contour suivant.	21/03/24	21/03/24	terminé
45	21/03/24	Creation de test_simplification_segment	Ce fichier prend en argument une image .pbm Il va ensuite calculer sont contour comme dans la tache 6 et le mettre dans le fichier du meme nom suffixé par .contour.txt Puis il va simplifier ce contour et mettre le nouveau contour dans un autre fichier (cette fois suffixé par .contour_simple.txt) Le fichier s'occupe de demandé a l'utilisateur la valeur de d qu'il souhaite. Et enfin il va calculer a partir du contour et du contour simplifier les images eps associés	21/03/24	21/03/24	terminé
46	21/03/24	Ecriture de simplification_segment.sh	Ce script bash prend en argument un dossier contenant des .pbm et va appeler test_simplification_segment sur toutes les images présentes Et il va ensuite placé les différents fichier créer dans les dossier correspondant	21/03/24	21/03/24	terminé
		TACHE 7				
47	25/03/24	Ecriture du module bezier	Contient dans le .h les structures bezier2 et bezier3 et dans le .c toutes les fonctions (approx2 et 3, distance2 et 3 etc...)	25/03/24	25/03/24	terminé
48	25/03/24	ecriture des test unitaire pour les deux partis	Ces test sont dans test_bezier2 et test_bezier_3 et sont tous bon (Ce sont les test du cours) .	25/03/24	25/03/24	terminé
49	25/03/24	ecriture des fonction cree_image_eps_bezier(2/3)	Dans le module eps et prennent le contour simplifier de courbe de bezier et ecrivent un fichier sous forme eps	25/03/24	25/03/24	terminé
50	25/03/24	ecriture des test_simplifcation_bezier pour degres 2 et 3	Tes pour la simplification (même méthode que dans la tache 6)	25/03/24	25/03/24	terminé
51	25/03/24	ecriture de simplification_bezier2(/3).sh	Pour calculer toutes les simplifcation pour toutes les images (Comme dans la tache 6)	25/03/24	25/03/24	terminé
52	25/03/24	creation des deux compte rendu		25/03/24	25/03/24	terminé
		TACHE 8				
53	26/03/24	il n'y a pas eut de modification de code pendant cette tache		26/03/24	26/03/24	terminé
54	26/03/24	Création des fichier texte de compte rendu pour les 2 partie et du pdf contenant les images de Asterix avec les différents types de simplifictaion et différentes distance seuil		26/03/24	26/03/24	terminé