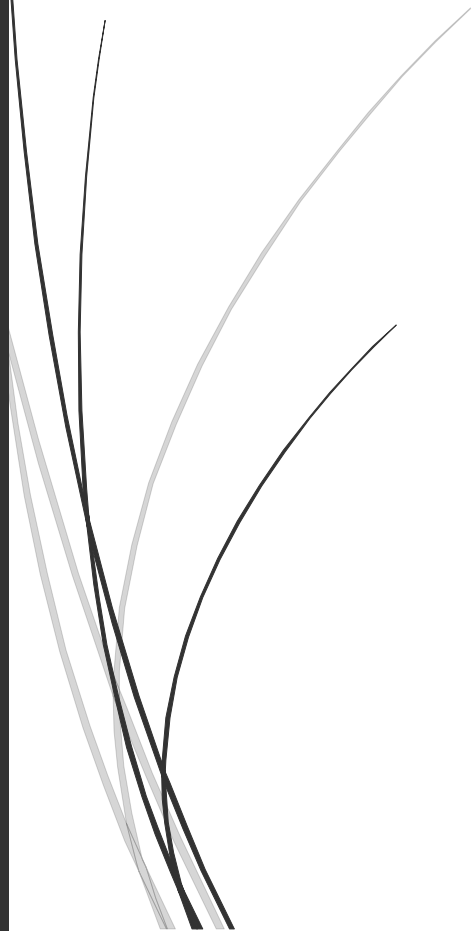
Année Universitaire 2020-2021

Rapport de stage

Développement d'une évolution d'un générateur de figures de Navon



Mathys Clerget
IUT INFORMATIQUE DE DIJON, LEAD

Remerciements

Je tiens à remercier Madame Poulin Charronnat Bénédicte, la Directrice du LEAD[1] pour m'avoir accepté au sein du laboratoire.

Je remercie tous les doctorants, Mme Pinson et Mme Martin du service Gestion avec lesquels j'ai passé mon stage, pour leur accueil, leur bonne humeur et pour m'avoir permis de m'intégrer facilement.

Je remercie Monsieur Bard, mon maître de stage, Ingénieur au laboratoire, pour sa confiance, pour m'avoir partagé ses connaissances et l'autonomie qu'il m'a offert pendant ce stage, me permettant d'affirmer mes compétences.

Je souhaite adresser mes remerciements au corps professoral et administratif de l'IUT Informatique de Dijon, pour la qualité de l'enseignement et plus particulièrement Madame Sologny, professeure à l'IUT, pour m'avoir suivi durant ce stage.

[1] Laboratoire d'Etude de l'Apprentissage et du Développement

Table des matières

Présentation du contexte

Présentation générale

Le LEAD est un laboratoire situé à Dijon effectuant des recherches, des expériences sur différents type de personnes, ceux-là peuvent aller du prénatale à la personne âgée. Les expériences sont très diverses, par exemple, le sujet peut avoir à :

- Écouter des sons, de la musique et réagir en fonction de ce qu'il a entendu.
- Faire du sport : il effectue une série de test après avoir exercé un effort physique.
- Compléter des tests de mémoire.
- Effectuer des tests visuels, repérer des éléments.
- Etc.

Les réponses sont récupérées par EEG[2] par Logiciel de traque des yeux ou parfois à l'écrit avec le sujet répondant directement sur l'ordinateur ou sur papier.

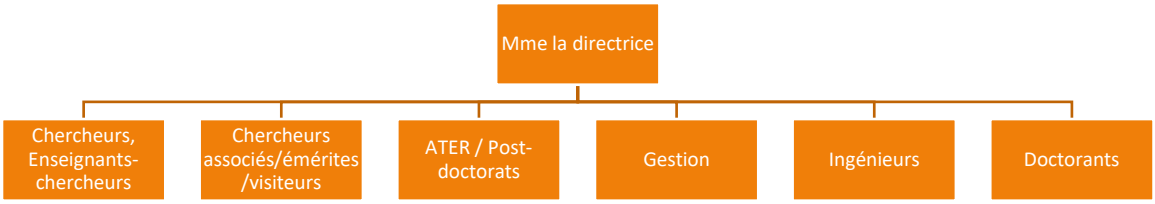


Image 1 : Organigramme du Laboratoire

Le Laboratoire est composé de plus de 50 personnes dont la directrice, Mme Poulin-Charronnat, 13 enseignants chercheurs dont M. Ambard enseignant à l'IUT. 2 personnes au service Gestion Mme Martin et Mme Pinson. 5 Ingénieurs et Techniciens, tel que mon tuteur de stage M. Bard. 12 autres chercheurs. 8 doctorants et les ATER [3]/ Post-doctorats, enseignant ou travaillant au laboratoire.

Présentation du service

Les ingénieurs et techniciens du laboratoire sont au nombre de 6 personnes. Ils permettent aux autres membres d'obtenir des programmes ou des systèmes embarqués améliorant, facilitant les recherches, ils permettent la maintenance du matériel informatique au sein du laboratoire.

[2] Électroencéphalographie
[3] Attachés Temporaires d'Enseignement et de Recherche (ATER)

Définition de la mission

La problématique

Avant de vous présenter la problématique je souhaiterais vous expliquer ce qu'est une « Figure de Navon » : C'est une figure constituée d'un élément de grande taille (une lettre par exemple) composé d'éléments de plus petite taille comme sur cette image (Image 2 : Exemple de figure de Navon) :

Image 2 : Exemple de figure de Navon

Par la suite j'utiliserai le terme de « *forme globale* [4]» afin de citer les différents ensembles de forme constituant la figure de Navon. Le terme « *éléments globaux* [5]» afin de citer les éléments tels que la ligne, le cercle et le demi-cercle, et « *éléments locaux* [6]» afin de citer les caractères ou les images remplaçants les éléments globaux créés à la création de la figure de Navon.

Au commencement, ce projet n'était qu'un sujet d'une discussion quelconque entre Monsieur Bard et Monsieur Witt (MCU[7]). Lorsque celui-ci faisait ses figures, il perdait beaucoup de temps à les former lettre par lettre. Une expérience ne se faisant pas qu'avec une seule de ces figures, la préparation était très longue. C'est à partir de ce moment-là que se développa l'idée d'un logiciel permettant de faciliter la tâche. N'ayant trouvé aucun logiciel, ils décidèrent de le créer, ce sujet fut donné auparavant en projet à des groupes d'étudiants d'ESIREM. Deux premières versions furent développées. Les deux étaient très différentes l'une de l'autre et leur fonctionnement, les résultats étaient très différents.

Mon stage était initialement nommé : « Développement d'une évolution d'un générateur de figures de Navon » : Le but était de reprendre le code des précédents projets et de l'améliorer afin qu'il atteigne toutes les attentes voulues. Hélas je n'ai pas fait l'évolution des précédents projets : Les codes étaient complétés par aucun commentaire, ils possédaient des erreurs non gérées. Ce qui avait été fait était intéressant, mais je ne pouvais pas reprendre ces travaux. J'avais donc décidé de repartir sur une base nouvelle.

[4] Ensemble d'éléments globaux constituant une figure de Navon
[5] Une ligne, un demi-cercle ou un cercle dessiner par l'utilisateur
[6] Un caractère ou une image utilisée afin de reproduire
[7] Maître de Conférences Universitaire

La tâche à effectuer

L'idée de ce projet est que l'utilisateur ait la possibilité de créer des figures de Navon facilement et rapidement telles que des lettres de l'alphabet et la possibilité de dessiner à sa guise une figure de Navon. Et pour les deux façons, l'utilisateur possède à sa disposition plusieurs options telles que :

- Changer l'élément local ;
- Changer la taille et la font de l'élément local.
- Changer l'élément local en une « image locale » : Que la figure dessinée soit remplie de petites images ;
- Changer la densité : le nombre l'éléments locaux (caractères ou images) utilisés afin de faire la figure ;
- Avoir à disposition une zone de dessin et des outils afin de dessiner ;

D'autres options pourraient être intégrées par la suite dans le logiciel, tel que la possibilité de passer par un script et une Invite de commandes afin de créer très rapidement des figures de Navon. Pouvoir déplacer des éléments globaux créés, enregistrées les options et les formes créées afin de reprendre le dessin ultérieurement.

Méthode retenue

Les différentes solutions envisagées

Les solutions envisagées étaient celles recherchées et testées par les précédents groupes :

- La création d'une figure de Navon par géométrie analytique : On créer une figure de Navon à partir des coordonnées des éléments utilisés.
  Par exemple : Un « A » majuscule est composé de trois segments.

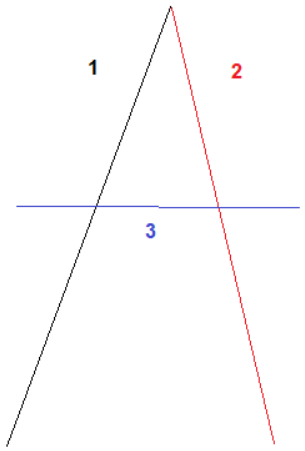


Image 3 : La lettre A dessinée par 3 segments

- Une grille avec des cases cliquables correspondantes aux emplacements où des lettres doivent apparaître.

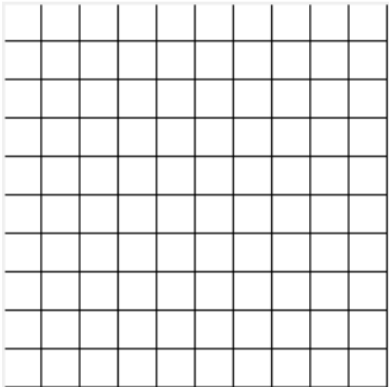


Image 4 : Grille de dessin

Il fallait dans un premier temps, tester ces possibilités, trouver comment celles-ci étaient améliorables, s'il y avait moyen d'implémenter le point le plus important : la densité.

La densité est dans ce logiciel, le nombre d'élément locaux utilisé afin de reconstruire les éléments globaux et la densité était, pour ces essais et les précédemment projets, le problème le plus important.

Les matériels et logiciels utilisés

Afin de réaliser ce projet, voici ce que j'ai utilisé :

Python 3.9

Pour ses bibliothèques diverses et son implémentation simple, j'ai décidé de faire mon projet en Python. La version utilisée est la 3.9, c'est à l'heure actuelle la version la plus stable qu'il existe du langage. Le Python possède plusieurs bibliothèques qui m'ont été très utile dans la réalisation de ce projet, des bibliothèques puissantes et assez simple d'utilisation :

Numpy : une bibliothèque permettant la création de tableaux et matrice à N dimensions, propose diverses fonctions mathématiques applicables directement aux tableaux créés et ce, sans réduire les performances du programme.

Image 5 : Numpy

SciPy : une bibliothèque à usage scientifique, utilisant d'autres bibliothèques telles que Numpy citée précédemment. Cette bibliothèque a été utilisé pour sa fonction d'interpolation.

Image 6 : SciPy

Pillow : une bibliothèque de gestion d'image « Python Image Library » me permettant de créer les images sur lesquelles sont dessinés les figures de Navon.

Image 7 : Pillow

Tkinter : une bibliothèque de graphique, m'ayant permis de créer l'interface homme-machine du programme.

Image 8 : Tkinter

Environnements de développements

Le premier EDI utilisé, principalement lors de la phase de test est Pycharm Community Edition :

Pycharm est un EDI utilisé pour programmer en Python, un EDI développé permettant d'analyser son code, d'utiliser un débugueur graphique et d'effectuer des tests entre autres. Il m'a permis, dans les premières semaines de développement, de test et de recherche, d'effectuer beaucoup de test, me permettant de découvrir les différentes bibliothèques citées plus haut.

Image 9 : Pycharm

Le second outil, celui que j'ai utilisé la majorité du temps est Sublime Text afin de développer ce programme. Sublime Text est un éditeur de texte, simple d'utilisation qui permet une navigation et une visualisation simple des fichiers utilisés.

Image 10 : Sublime Text

Conception

Visual Paradigm : Logiciel me permettant de faire les diagrammes UML[8]. Nous avions déjà utilisé ce logiciel en cours afin de créer la conception de projet.

Image 11 : Visual Paradigm

GitMind : La licence de Visual Paradigm ayant expirée avant la fin de mon stage, j'ai repris toute la conception sur GitMind, un site permettant de faire différents diagrammes UML.

Image 12 : GitMind

Suivi de l'avancée

Afin de suivre la progression globale du projet, j'ai utilisé plusieurs logiciels à ma disposition :

GitHub : Afin de faire des sauvegardes régulières de mon code et de faire un pas en arrière si besoin, mon avancée de la programmation du projet est enregistrée dans un dossier que seul moi peut accéder. Ces enregistrements permettent en cas de pépins (comme cela a pu m'arriver) de reprendre à un point précédent sans tout recommencer.

Image 13 : GitHub

OneNote : Le programme de prise de note de Microsoft, me permet de faire un suivi quotidien de mon avancée dans mes recherches.

Image 14 : OneNote

GanttProject : Un logiciel me permettant de modéliser le diagramme de Gantt de ce projet.

Image 15 : GanttProject

[8] Unified Modeling Language, Langage de modélisation unifié : langage de modélisation graphique à base de pictogrammes

Application de la méthode et résultats

Les différentes phases de la réalisation

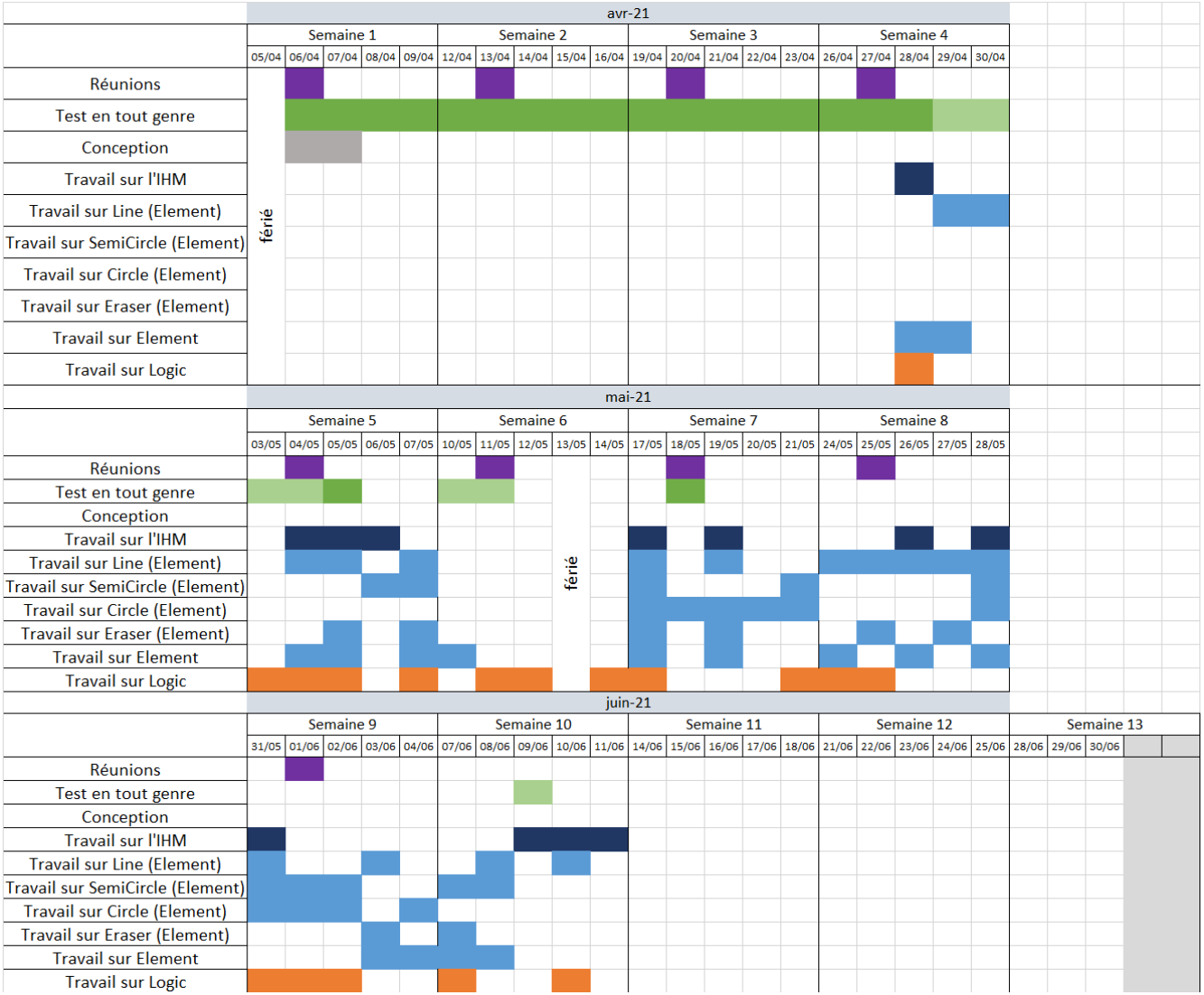


Image 16 : Diagramme Gantt de mon stage

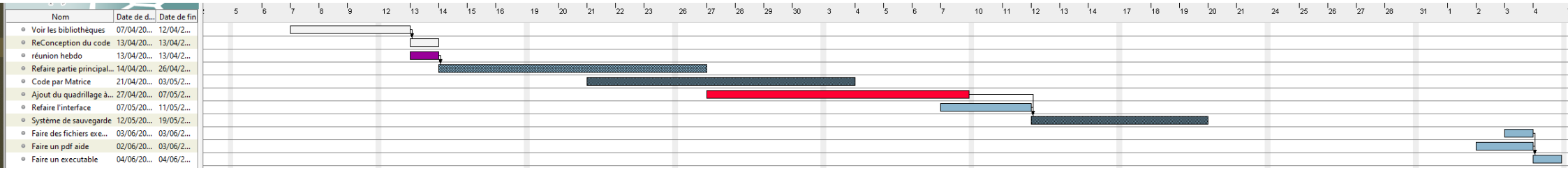


Image 17 : Diagramme de Gantt Provisionnel

Ce diagramme prévisionnel a été réalisé par moi-même au tout début du stage, c'était, ce que je pensais être, un bon diagramme, j'étais partie sur l'idée d'utiliser la grille de dessin afin de réaliser le projet, les tâches se superposent car je ne savais pas réellement comment celles-ci se passerait, et une espace vers la supposée fin de mon stage le 5 juin. Mais finalement les tâches ont tout de suite été différentes et plus tard mon stage a été prolongé d'un mois environ, jusqu'au 30 juin. Me laissant plus de temps afin de compléter ce projet.

Comme nous pouvons le voir sur ce diagramme de Gantt (Image 16 : Diagramme Gantt de mon stage) : mon travail s'est décomposé en deux grandes phases :

1. Une première phase d'une durée d'environ un mois : une phase de documentation, d'essais ;
2. La seconde phase, la programmation du projet, s'étant de la fin du mois d'Avril à aujourd'hui.

Première phase : documentation et essais

La première phase est une phase de recherche. Une observation du travail fait précédemment. Une découverte des bibliothèques : Numpy, Matplotlib, SciPy sont des bibliothèques très utiles pour les calculs scientifiques mais que je n'ai jamais utilisé auparavant, mes connaissances en Python étaient assez pauvres au début du stage. J'ai donc commencé par des tutoriels, des vidéos, des démonstrations de ce qu'il était possible de faire, afin de prendre en mains ces nouveaux outils.

Dès nos premières réunions, nous avons parlé du principal problème de ce projet : la gestion d'une densité : le nombre d'élément locaux qui seront utilisés afin de donner forme à la forme globale voulu (Image 18 : Démonstration densité).

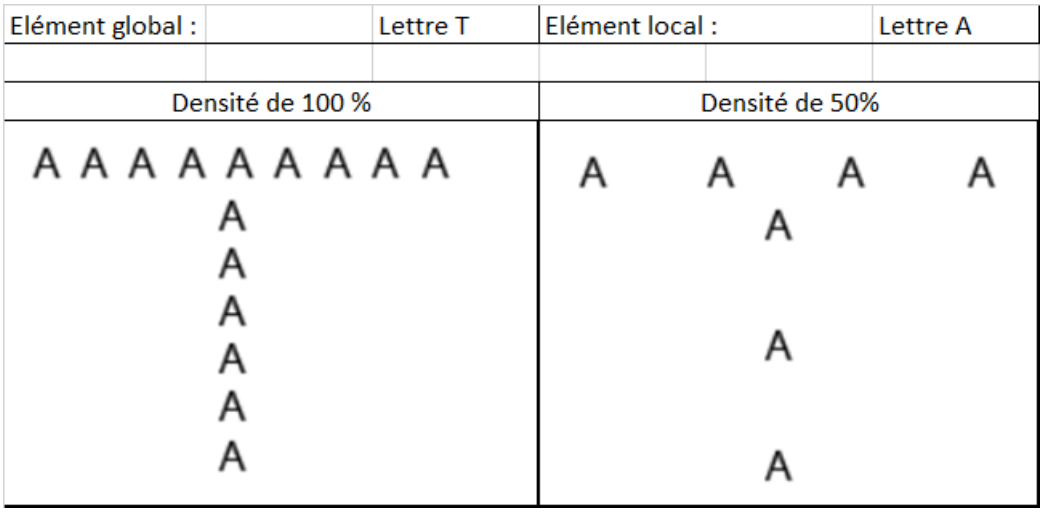


Image 18 : Démonstration densité

La première idée dont nous avons discuté était d'avoir une grille sur laquelle on pouvait cliquer afin de décider des emplacements où faire apparaître des éléments locaux lors de « l'impression » de la figure de Navon (Image 19 : Exemple grille).

Cette idée avait déjà étudié et programmé auparavant, en C++, l'interface n'était pas simple d'utilisation : Il fallait dessiner sur la grille la figure voulue, enregistrer un fichier au format .txt puis téléverser ce nouveau document sur le programme afin qu'il créer une figure de Navon à partir de celui-ci… Ce n'était pas possible de conserver un tel fonctionnement.

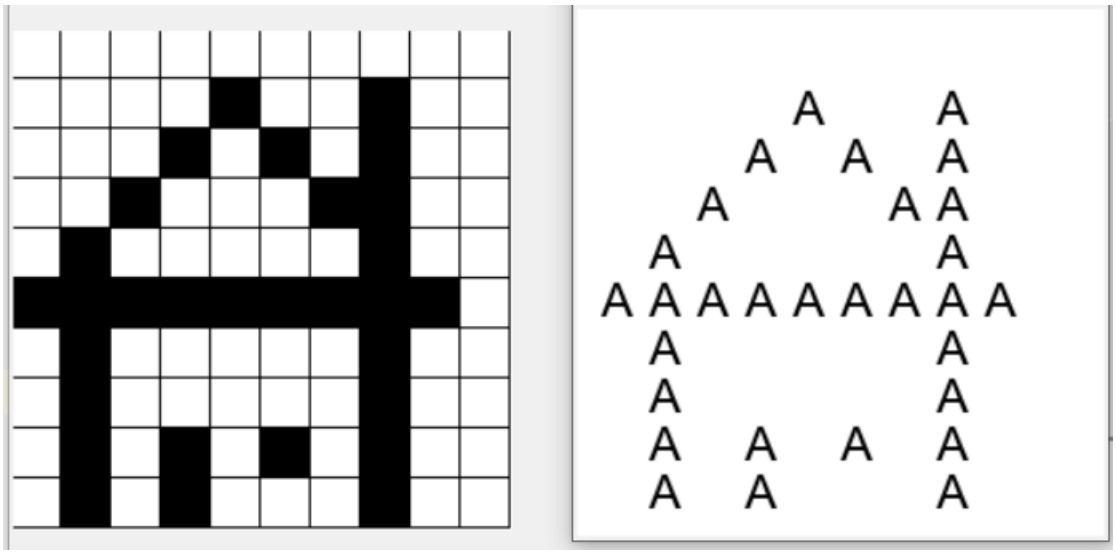


Image 19 : Exemple grille

Nous avons recherché plusieurs solutions possibles concernant la gestion de la densité avec cette méthode de grille de dessin et des cases cliquables, plusieurs techniques ont été testé :

1. La première consistait à créer un « masque de densité » : ce masque serait une grille de la même taille que la grille de dessin, et elle serait composée de 1 et de 0 en fonction du pourcentage de densité donné. Et avant de créer la figure de Navon, on « superpose » les deux grilles et on ne garde de la grille de dessin que les cases noircies (Image 20 : Exemple densité (50%) par masque).

   Comme on peut le voir sur cet exemple, le masque peut finir par faire perdre tout son sens à la figure de Navon voulue.

Image 20 : Exemple densité (50%) par masque

2. La seconde technique consistait à créer des espaces entre chaque lignes et colonnes de la grille de dessin. On reprend la grille de dessin et entre chaque ligne, on ajoute entre chaque ligne et sous chaque ligne un nombre d'espace choisit (Image 21 : Densité par 2 espaces).

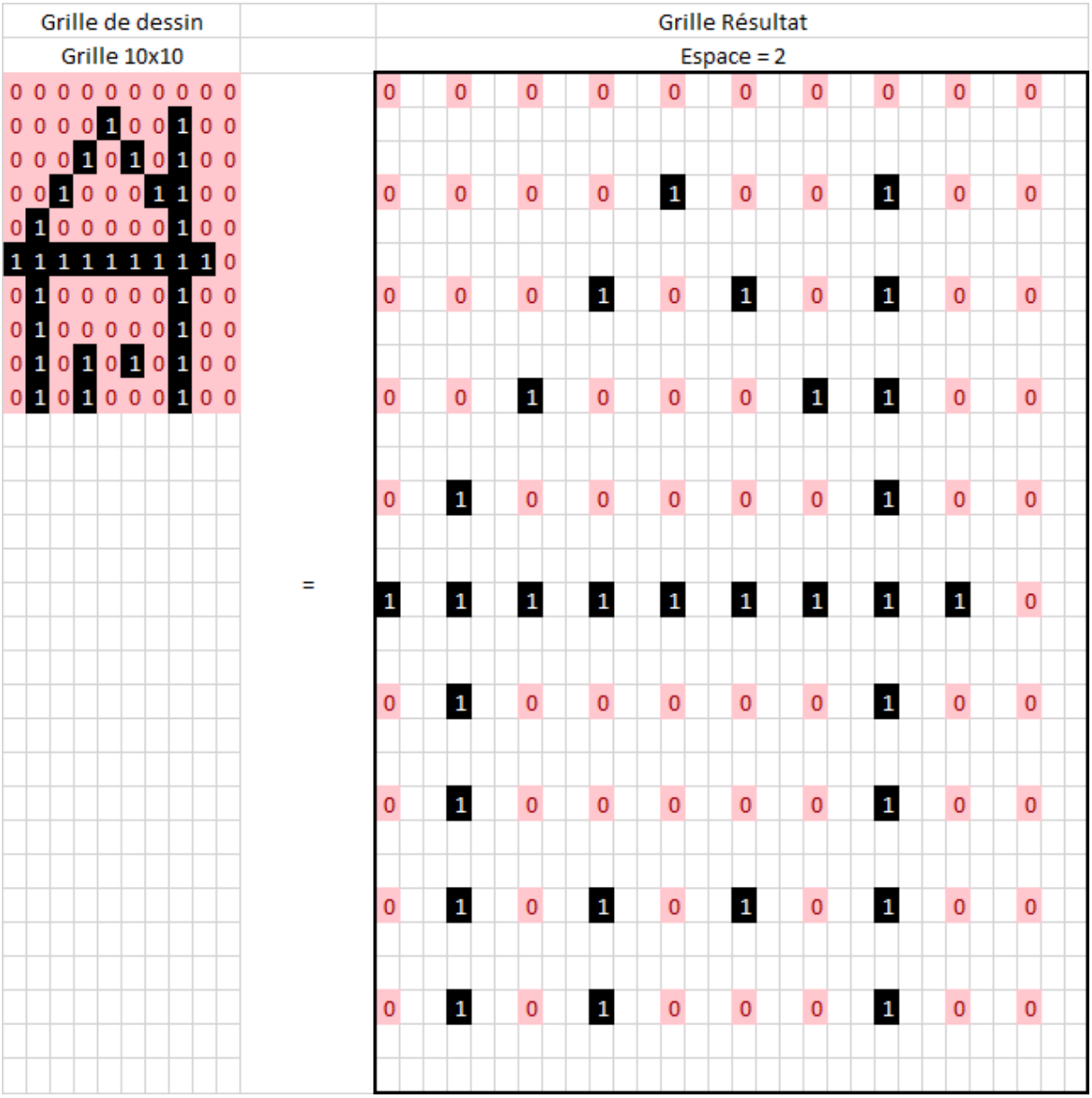


Image 21 : Densité par 2 espaces

Une figure de Navon créée avec 2 espaces comme ici augmente considérablement la taille de la figure de Navon et le temps d'exécution, de création de la figure de Navon augmente considérablement lui aussi.

3. La troisième technique consistait à ajouter, au lieu d'espace comme la version précédente, d'autres éléments locaux du même type : on multipliait l'état de la case un nombre de fois choisit par l'utilisateur (Image 22 : Densité par multiplication des cases) :

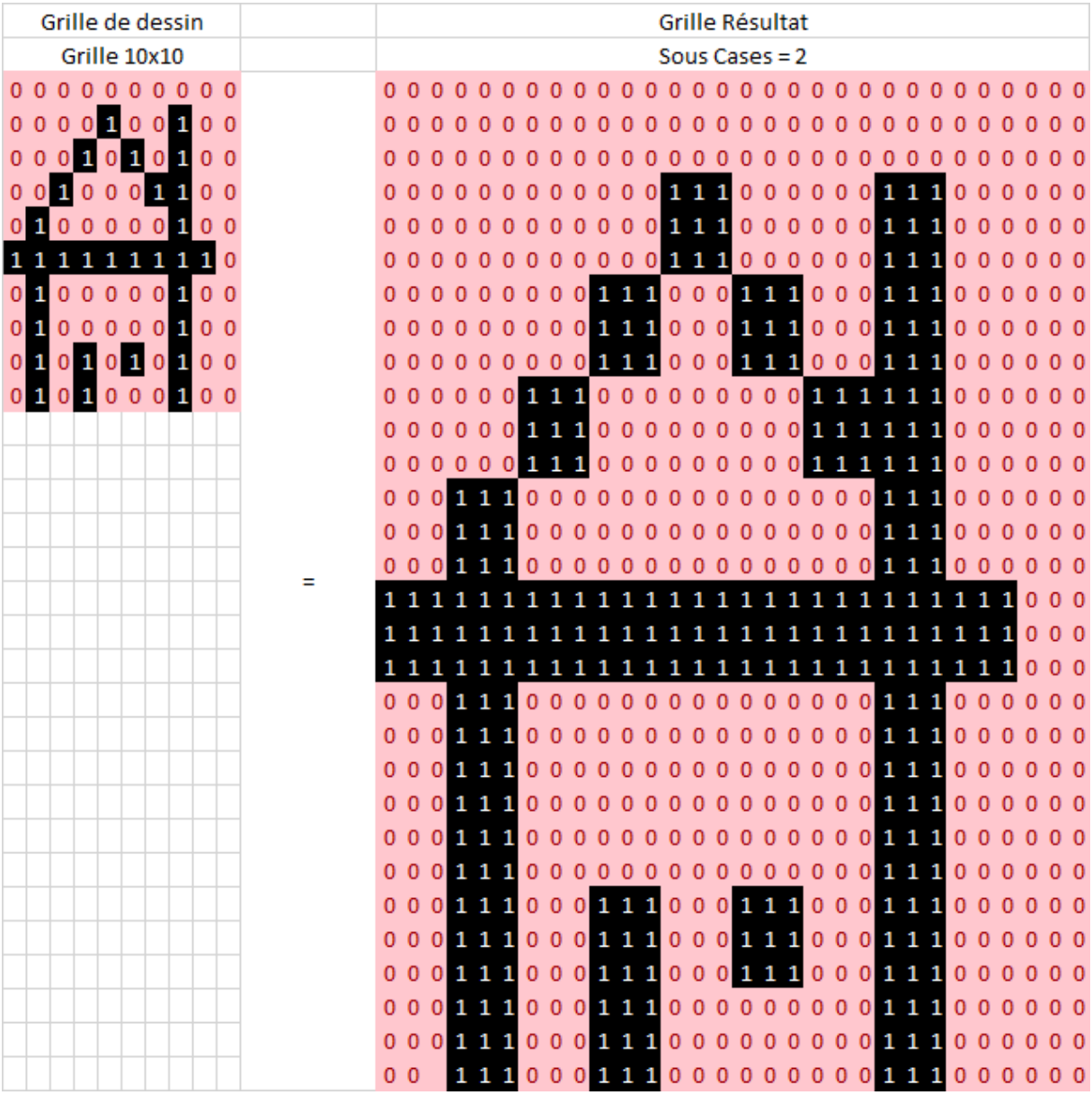


Image 22 : Densité par multiplication des cases

Comme avec les espaces, le temps d'exécution et la taille de la figure augmentent considérablement.

4. Il y eu d'autres expérimentations tels que le mélange de la technique « d'espacement » et de « multiplication des cases » mais sans réels résultats convenables.

Ces derniers tests n'étant pas concluant, il fallait trouver une meilleure solution… Les grilles sont simples à utiliser, mais le long temps de création de la figure de Navon n'est pas envisageable pour le programme final et les grilles présentent un autre problème : Nous ne pouvons pas créer de figures complexes composées de cercles ou de demi-cercles. Ces tests nous ont prouvé que passer par une grille de dessin n'est pas une solution adaptée pour ce que l'on cherche à faire.

Nous partons donc sur une variante de la seconde idée programmée auparavant, nous allons utiliser une surface de dessin, les coordonnées des éléments dessinés afin de création la figure de Navon.

Seconde phase : programmation du projet

La programmation de ce projet ne s'est pas passé tel un simple projet fait durant les cours à l'IUT : Chaque avancée ne fut que de petites avancées et il m'a fallu souvent revenir sur mon précédent code afin de le développer, de le modifier voire de le supprimer.

Sachant que ce rapport vous a été donné avant la fin de mon stage, il se peut que beaucoup de chose change à nouveau. Mais l'idée générale sera toujours expliquée :

Comme on peut le retrouver sur le diagramme de Gantt (Image 16) : J'ai séparé la programmation en trois grandes parties :

1. La partie IHM : (en bleu foncé) comprend les différentes parties visibles du projet : la fenêtre principale est composée d'une partie permettant à l'utilisateur de créer une figure de Navon de A à Z et seconde partie permettant d'obtenir facilement une figure à partir de plusieurs paramètres.
   Une troisième partie, ayant débuté juste avant le rendu des rapports, comprend un aperçu direct de la figure de Navon en cours de création.

2. Les éléments globaux : (en bleu clair) correspond à tous les éléments dessinables que comprend le programme : l'utilisateur peut dessiner un trait, un demi-cercle et un cercle et le programme « traduit » ces éléments en une suite de lettres ou d'images locales pour la figure de Navon. L'utilisateur a, à sa disposition, une gomme (Eraser) afin de supprimer un élément.

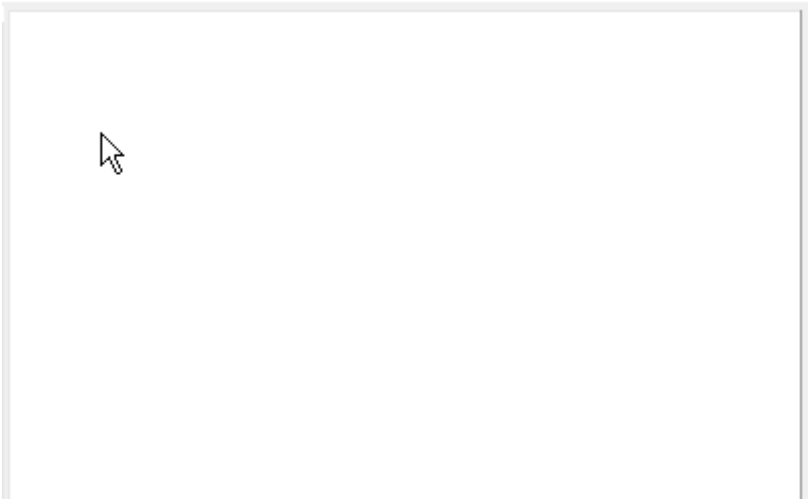
3. La partie Logic : (en orange) comprend toutes les fonctions de créations de la figure de Navon. Logic comprend aussi une partie avec des paramètres nécessaires à la création de l'application.

A travers ces trois parties nous pouvons nous poser une multitude de questions : Comment dessiner ses différents éléments ? Comment la densité a-t-elle finalement été gérée ? De quelle façon sont placés les lettres ou les images locales pour la figure de Navon finale ?

Comment dessiner ses différents éléments ?

L'utilisateur, comme toute autre personne dépend de sa souris afin d'utiliser l'application, j'ai donc utilisé ses « dépendances » afin de créer ses éléments globaux sur la surface de dessin, par exemple : la création d'un trait débute par un **Clic** sur la surface de dessin, puis l'utilisateur **Déplace** la souris jusqu'à obtenir l'élément sous sa forme voulue et fini par **Relâcher** le bouton de la souris afin de confirmer la création de cet élément.

Ses trois **dépendances** sont les mêmes pour chaque élément que l'utilisateur peut créer. A l'exception du Eraser ayant une fonction différente de ces autres éléments. Le **Clic** indique la création d'un trait, d'un cercle. Chaque **déplacement** de la souris -avec le clic enfoncé- change la forme de l'élément, puis le **relâchement** du clic de la souris permet son enregistrement : Ses informations, telles que ses coordonnées, son type etc… sont enregistrées afin d'être réutilisées ultérieurement (Gif 1 : Création d'une ligne).



Gif 1 : Création d'une ligne

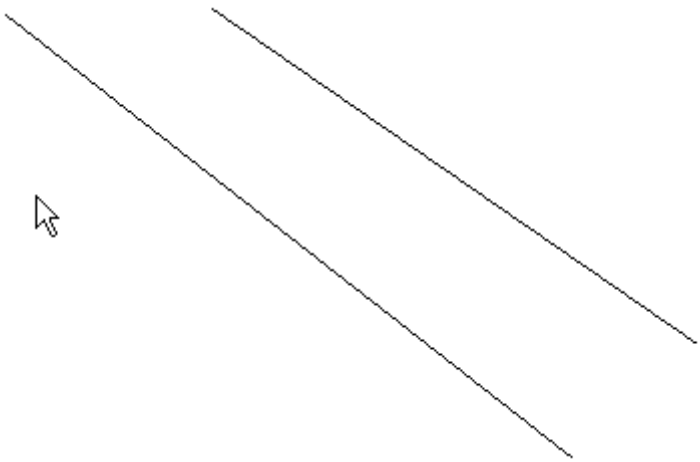
Voici un exemple visuel pour représenter le principe : **Clic, mouvement, Déclic.** Ce fonctionnement est similaire à n'importe quel logiciel de dessin tel que Paint. Il reste, tout de même, quelques différences :

Les cercles et demi-cercles se créent depuis leur centre : l'emplacement du **clic** indique le centre et l'emplacement du **déclic** correspond à un point de l'élément : voici un exemple pour chacun de ces deux éléments :

Gif 2 : Création d'un cercle

Gif 3 : Création d'un demi-cercle

Le dernier élément programmé pour l'instant, ne fonctionnant pas comme les trois précédents est le *Eraser* : il ne possède pas de fonction liée à son **clic**, ni liée à son **mouvement**, il en possède une à son **déclic**, permettant en ayant la souris sur un autre élément, de le supprimer. Cette fonction de suppression a été placé sur le **déclic** par choix, celui-ci permet à l'utilisateur d'avoir « une deuxième chance » qui lui permet lorsqu'il se rend compte qu'il ne veut pas supprimer un élément après avoir cliqué, d'effectuer le **déclic** dans une zone ne comportant aucun élément.



Gif 4 : Utilisation du Eraser

Même après ces informations sur les éléments, tout n'est pas expliqué : laisser libre court à l'utilisateur lui permet de créer n'importe quel type de figure de Navon mais cela amène un problème : la superposition des éléments. Nous reviendrons là-dessus par la suite.

Comment la densité a-t-elle finalement été gérée ?

La densité, comme indiqué précédemment, est le point important de ce projet.

Après plusieurs recherches, monsieur Bard avait conclu que la fonction gérant de la densité se calculait avec ces paramètres :

- Le pourcentage de densité choisi par l'utilisateur : (d)
- La somme des longueurs de tous les éléments sauvegardés : (sumL)
- La taille de la police du caractère local, ou la longueur de la diagonale de l'image locale (size)
- La longueur de l'élément actuel (l)

La fonction calcule le pourcentage, d, multiplié par la somme, sumL, divisé par la taille de la police, size, le tout divisé par la division de la taille, l, par la somme, sumL.

$$d * sumL/size * l/sumL$$

Ce calcul renvoie un nombre N, correspondant au nombre d'éléments locaux contenus dans toute la longueur de l'élément global. Et ce calcul est utile dans la dernière partie du programme, la création de la figure de Navon, lorsqu'il faut transformer tous les éléments de la zone de dessin en une suite de lettres ou d'images.

De quelle façon sont placés les lettres ou les images locales pour la figure de Navon finale ?

Au départ, la figure de Navon n'est qu'une image blanche puis, pour chaque élément global, il faut calculer N comme indiqué ci-dessus, calculer l'interpolation à l'aide des informations enregistrées (coordonnées, rayon etc.) afin d'obtenir sa forme et à partir de ces résultats, on ajoute sur une l'image, aux coordonnées calculées, les éléments locaux.

Gif 5 : Création d'une Figure de Navon

Maintenant que nous sommes capables de créer un des éléments globaux puis de les convertir en éléments locaux sur la figure de Navon, le programme est dans sa globalité : réalisé. Il reste cependant plusieurs problèmes à gérer…

Les difficultés rencontrées

Le principal problème rencontré dans ce projet sans l'expliquer à nouveau était la gestion de la densité. Mais d'autres moins importants sont apparus, Ceux-là étaient souvent dû un manque de connaissance spécifique sur certains sujets de ma part ou une mauvaise conception de mon code.

La superposition d'éléments globaux

L'utilisateur veut, par exemple, créer un X, telle une croix comme sur une carte aux trésors : pour réaliser cela : il fait simplement croiser deux lignes de tailles à peu près égales, mais ce X est constitué donc d'une intersection (Image 24 : Croix dessinée), et cette intersection provoque, lors de la création de la figure de Navon, une superposition d'éléments locaux (Image 24 : Figure de Navon de la croix).

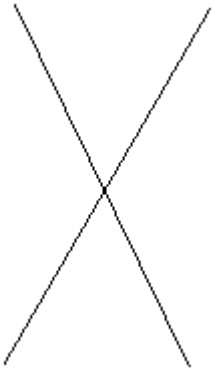


Image 24 : Croix dessinée

Image 24 : Figure de Navon de la croix

Cette superposition, comme on peut le voir, créé une collision entre différents éléments locaux et ceux-là sont décalés les uns des autres : Cela pose un problème dans l'utilisation du logiciel, voici comme j'ai procédé pour y remédier :

Ce genre de problème doivent obligatoirement se régler sous deux aspects : du côté utilisateur et du côté exécution du code

1. Il fallait, premièrement, indiquer à l'utilisateur qu'il allait créer une collision entre deux éléments : lorsque l'utilisateur effectue un **mouvement** avec un élément, il peut à un moment ou un autre rencontrer un autre élément, que ce soit sur le long d'une ligne, sur le périmètre d'un cercle ou d'un demi-cercle. Dès que deux éléments se rencontre, un point rouge est créé à l'écran indiquant, l'emplacement du point d'intersection des deux éléments. Indiquant ainsi visuellement qu'il y a collision (Image 20).

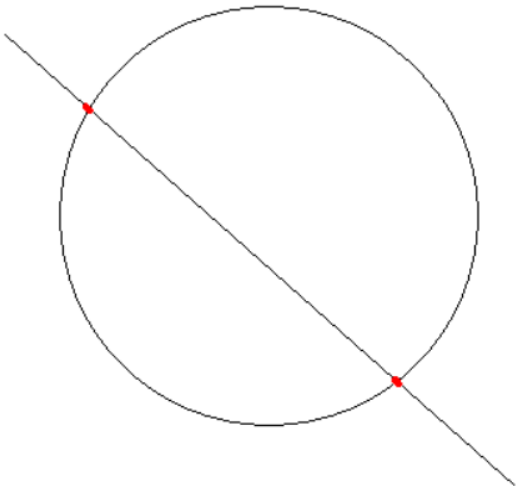


Image 25 : Exemple collision

2. Maintenant que l'on sait quels éléments se rencontrent et l'emplacement de cette rencontre, il faut utiliser ses informations pour que lors de la création de la figure de Navon, ne plus avoir cette superposition : Il y eu pour cela deux versions de réaliser :

    a. La première version consistait à supprimer lors de la création des éléments locaux sur la figure de Navon (De quelle façon sont placés les lettres ou les images locales pour la figure de Navon finale ?). Dans l'exemple du X, Une ligne serait composée de tous ses éléments locaux et il manquerait un élément au lieu de la superposition à l'autre ligne.

Image 26 : Image avec superposition gérée

Comme on peut le voir, certes il n'y a plus d'éléments locaux se superposant mais l'image n'est convenable tout de même : il n'y a pas, entre chaque élément local, la même distance les séparant et cela gêne énormément la silhouette créée par la forme globale de la figure de Navon.

b. La deuxième version, qui est toujours un travail en cours, est un peu plus complexe, il consiste à utiliser le point de superposition comme « point de séparation » et donc séparer les éléments se superposant en deux éléments chacun commençants à ce point-là :

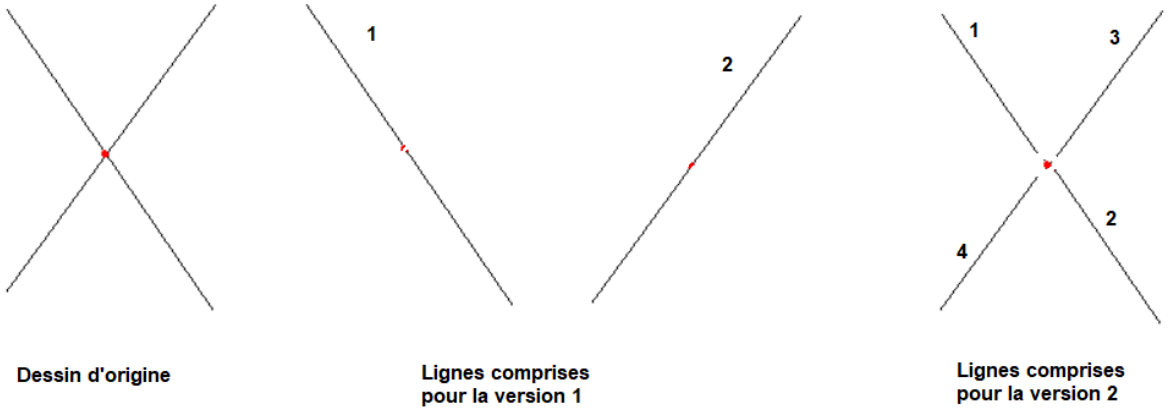


Image 27 : Superposition gérée par séparation des éléments

Les éléments locaux seront tous à la même distance et ne posent plus de problème de superposition.

Conclusion

Les leçons tirées de ce travail

Le stage de fin d'étude permet à tout étudiant de valider ses compétences, ses capacités acquises au cours des deux années de DUT Informatique. C'est aussi une première expérience professionnelle pour beaucoup d'entre-nous. Ce stage collait parfaitement avec mes recherches : « découvrir le monde informatique dans un environnement qui n'est pas basé, fixé sur l'informatique ». J'ai pu découvrir ce monde différent du cadre de travail entre étudiant en informatique et encore plus d'une entreprise basée sur l'informatique.

J'ai pu améliorer mes compétences informatiques en découvrant ces nouveaux outils Python. J'ai pu confirmer, grâce à ce stage, mes capacités en matière de programmation, de conception et de gestion de projet. Ce stage m'a permis de prendre plus ample confiance en moi pour la suite de mes études.

Lexique

ATER : Attachés Temporaires d'Enseignement et de Recherche

EEG : Électroencéphalographie

Élément global : Une ligne, un demi-cercle ou un cercle dessiner par l'utilisateur

Élément local : Un caractère ou une image utilisée afin de reproduire

Forme globale : Ensemble d'éléments globaux constituant une figure de Navon

LEAD : Laboratoire d'Etude de l'Apprentissage et du Développement

MCU : Maître de Conférences Universitaire

UML : Unified Modeling Language, Langage de modélisation unifié : langage de modélisation graphique à base de pictogrammes

Bibliographie

Documentation Numpy : https://numpy.org/doc/stable/user/whatisnumpy.html

- https://youtu.be/QUT1VHiLmmI?list=PLWKc8GOkH2IecwiTMDFGTa-XuKK89Uy6P
-

Documentation Tkinter : https://anzeljg.github.io/rin2/book2/2405/docs/tkinter/index.html

- https://docs.python.org/3/library/tk.html
- https://stackoverflow.com/questions/32289175/list-of-all-tkinter-events

Documentation Pillow : https://pillow.readthedocs.io/en/stable/reference/ImageDraw.html

La majorité de ma documentation provient de forums tels que StackOverflow , GeekForGeek et principalement de la documentation officielle des bibliothèques Python utilisées.

Résumé et mots-clés :

Mon stage consiste à développer un générateur de Figure de Navon : l'utilisateur peut faire un dessin à partir de trait, de cercle et demi-cercle ou générer une figure préenregistrée et utiliser toutes les options à sa disposition afin de modifier la figure comme bon lui semble.

Mots-clés : Python, Développement logiciel, Mathématiques, Laboratoire, Autonomie

My internship consists in developing a Navon's Figure generator : The user can make a drawing from line, circle and semi-circle, or he can generate a pre-saved Figure and use all the options available in order to modify the Navon's figure.

Keywords : Python, Software Development, Mathematics, Laboratory, Autonomous.