**I - Description Projet Terminale**

*C’est juste un Paint*

Le projet a pour but d’être un logiciel de création graphique, il sera réalisé grâce à la librairie pygame de python. Idée directrice au lancement du projet : réaliser une version du logiciel possèdent les fonctionnalités les plus importantes de *Paint*.

Liste de fonctionnalités : Dessiner, rogner, colorier, remplissage (paraît complexe à réaliser à première vue), zoomer, etc ….

Information pratique : utilisation de la librairie PIL image (vu en snt en second), utilisation de la POO pour créer une feuille qui servira de support au dessin (implémentation de l’objet *feuille*)

Plan

I - Les initialisation des variables

A - Les images en PIL

B - Les images en pygame surface

C - Les Variables, liste, tuples, …

D - La classe Feuille

II - Les fonctions

A - Les fonctions points

B - Les fonction relie points

C - Remplissage et Échange

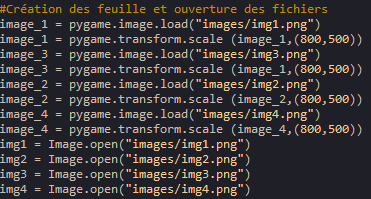
III - La boucle while

A - Les affichages

B - Les boutons

C - Get\_pressed sur la feuille

**I - Les initialisations**

****

**A - Les images en PIL**

* Les images 1, 2, 3, et 4 sont d’abord ouvertes en tant que *pygame surface* redimensionner pour contrer les éventuels problèmes de re dimension engendré par python, elles sont cependant de même définition que ce qui leur est appliqué par cest commande et permet donc de faire correspondre les dimension physique de l’image avec ça représentation sur pygame soit (800,500). Elles sont ensuite ouvertes avec PIL image pour permettre leur utilisation avec get\_pixel, put\_pixel et save .

**B - Les images en pygame** **surface**

* Création de variables correspondant à chaque autre image utilisé dans le programme qui permet ainsi de les redimensionner avec transforme.scale unifiant ainsi les taille des icônes des outils tel que le crayon, la gomme, etc. On implémente aussi les images correspondant aux différents menus.

Ce qui permettra ainsi de les placer au bon endroit et dans l’ordre voulu au moment de les afficher avec la commande blit.

**C - Les variables, listes et tuples**

* Ces variables servent comme précisé à côté à réaliser différentes tâches, notamment de comptage ou de mémorisation de données.

Couleur : est un tuple de trois valeur correspond à une couleur en rvd (rouge, vert, bleu), utilisé pour savoir de quelle couleur va écrire l'outil utilisé, initialisé par défaut sur (0,0,0), soit du noir.

menu\_ouvert : Permet de garder en mémoire si le menu paramètres est ouvert ou non.

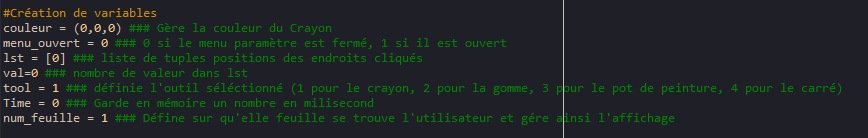
lst : Permet de garder en mémoire chaque clic effectué pendant l’exécution du programme, initialisé avec la valeur 0 pour ne pas créer de décalage entre le nombre de clique et l’indice de l’élément de la liste auquel il correspond.

val : garde en mémoire un nombre de clique effectué permet avec la commande time d’être réinitialisé afin de ne pas relier deux points alors que le temps écoulé entre les deux est trop important.

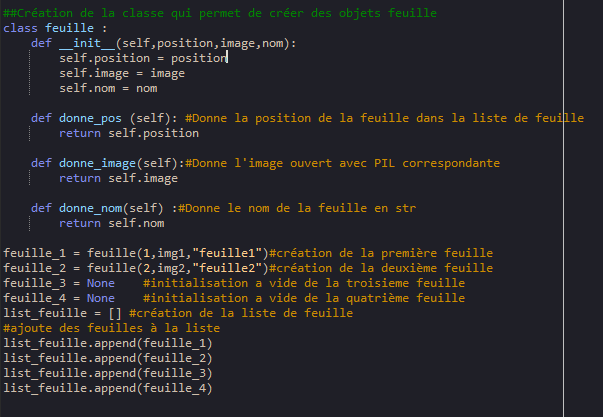
tool : Garde en mémoire quel outils à été sélectionné en dernier pour exécuter la bonne fonction en fonction dans la boucle while, initialisé par défaut à 1, soit le crayon.

Time : garde en mémoire un durée de temps, qui si elle excède une durée défini se réinitialise à 0.

num\_feuille : Garde en mémoire le numéro de la feuille sur laquelle on se trouve permet ainsi de dessiner sur la bonne feuille.



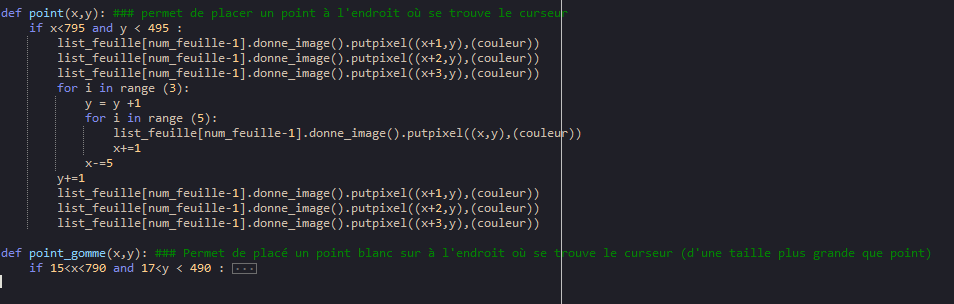
**D - La classe feuille**

* Permet de créer des objets de la classe feuille qui auront comme attribut trois valeurs pré renseignés lors de leur création : position, une valeur int permettant de récupérer le numéro de la feuille, image : permet de récupérer la variable à laquelle correspond l’image ouverte avec PIL, et son nom en chaine de caractère pour pouvoir l’afficher plus facilement.

On y trouve aussi trois méthodes permettant d’accéder aux trois valeurs énoncées précédemment : donne\_pos, donne\_image et donne\_nom.

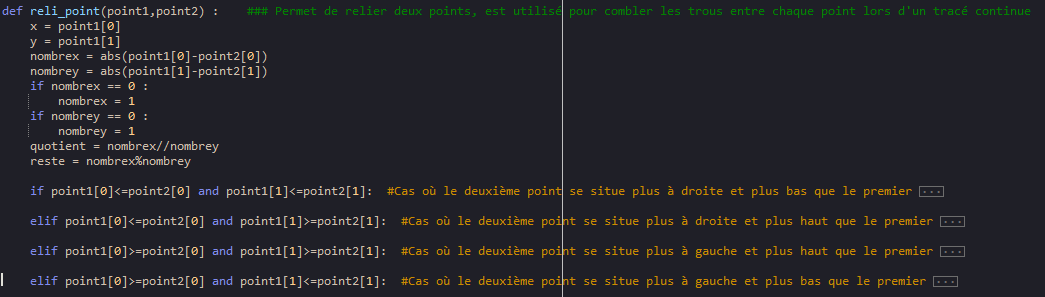
On crée ensuite 4 éléments de la classe feuille, deux étant pour l'instant initialisés à None (ne sont donc pas encore des éléments de la classe), on les ajoute ensuite dans une liste de feuilles qui permettrait de les parcourir plus facilement au moment de les afficher.

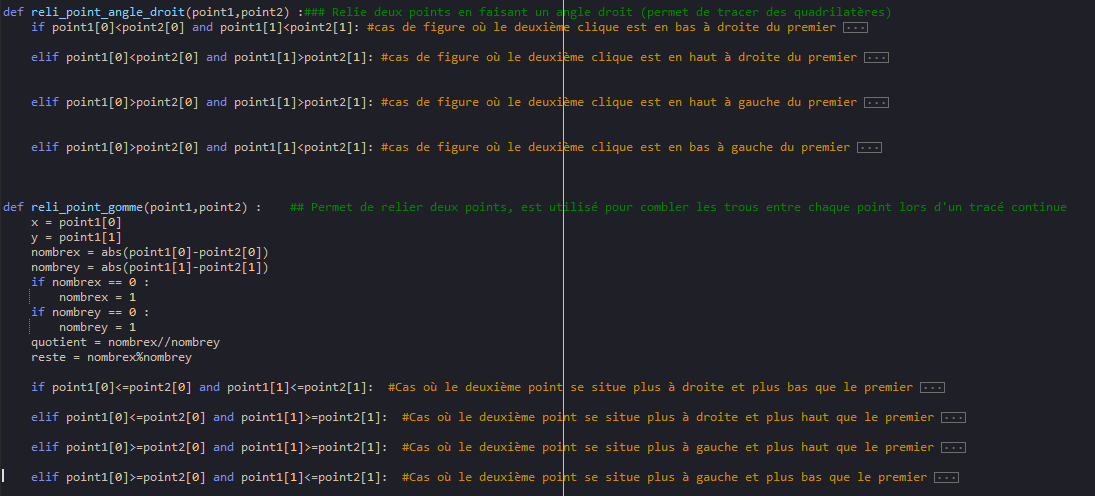
**II - Les fonctions**

**A - Les fonctions point** 

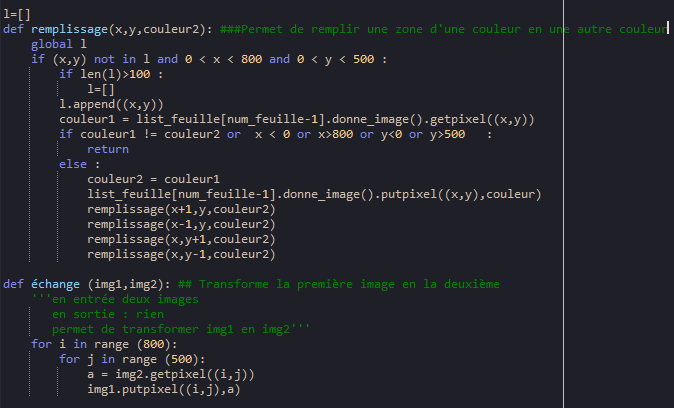
* Point : Permet de changer de couleur les pixels autour d’un pixel choisie, il créra alors une forme comparable à la représentation si-contre.
* Point gomme : Permet elle aussi de changer le pixels autour du point cliqué mais change beaucoup plus de pixels que la fonction point, le tracé de la forme produite étant beaucoup plus irrégulier que le précédent son code n’a pas encore été optimisé.

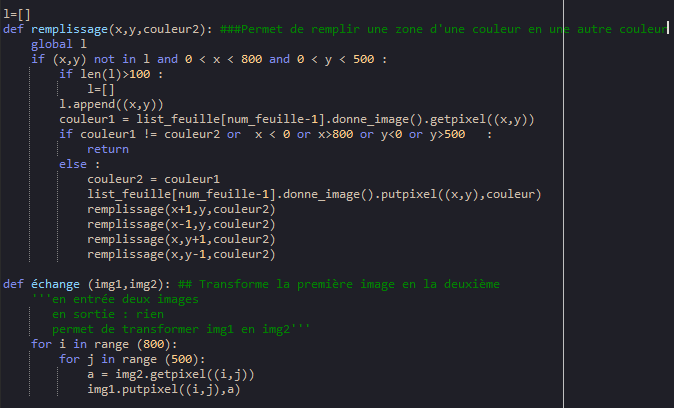
**B - Les fonction relie points**

* reli point : Permet de relier les deux points placés en arguments grâce à leur coordonnées, il se sert alors du nombre de points séparant les deux abscisses des points ainsi que leurs ordonnées pour déterminer le nombre de pixel les séparant et trouvé la ratio entre ces deux valeurs. Ensuite en fonction de la position du premier point par rapport au deuxième, il exécute d’abord deux pixel dans le coordonnées le plus important et un pixel dans l’autre en décrémentant à chaque exécution le reste de 1, il peut ensuite finir en faisant un pixel en abscisse et un pixel en ordonné jusqu’à arriver au deuxième point (chaque exécution est exprimé par l'exécution de la fonction point).
* reli point gomme : Exécute le même programme que reli point en remplaçant l’exécution de la fonction point par la fonction reli point gomme.
* reli point angle : Permet de relier deux points mais cette fois en en ne tenant compte que de coordonné qui sépare les points en abscisse et en ordonné, on trouve alors de nouveau quatre cas correspondant au position d’un point par rapport à l’autre. Il relit d’abord le premier point au second en commençant par tous les points en abscisse puis tous les points en ordonnée et recommence dans l’ordre inverse.

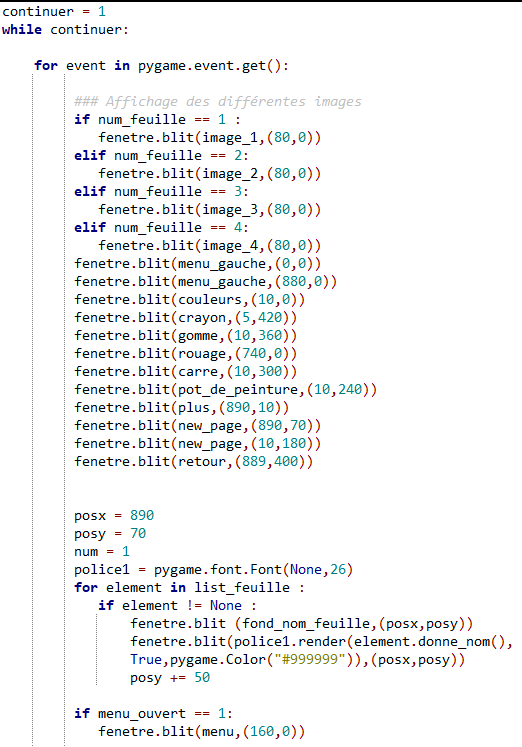


**C- Les fonctions Remplissage et Échange**

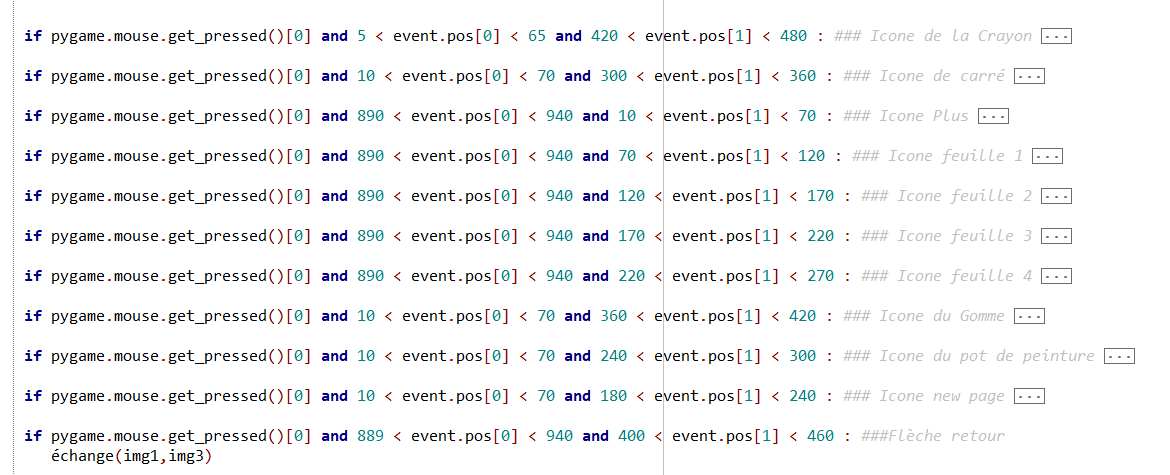
* Remplissage : Permet de remplir récursivement tout un zone de l’image d’une couleur en une autre. Chaque pixel est changé de couleur point les quatre pixels qui l’entoure aussi de façon récursive.
* Échange : Permet de transformer l’image 1 en l’image 2 en parcourant tous les pixels des images et en plaçant la couleur du pixel de l'image 2 au même endroit de l’image



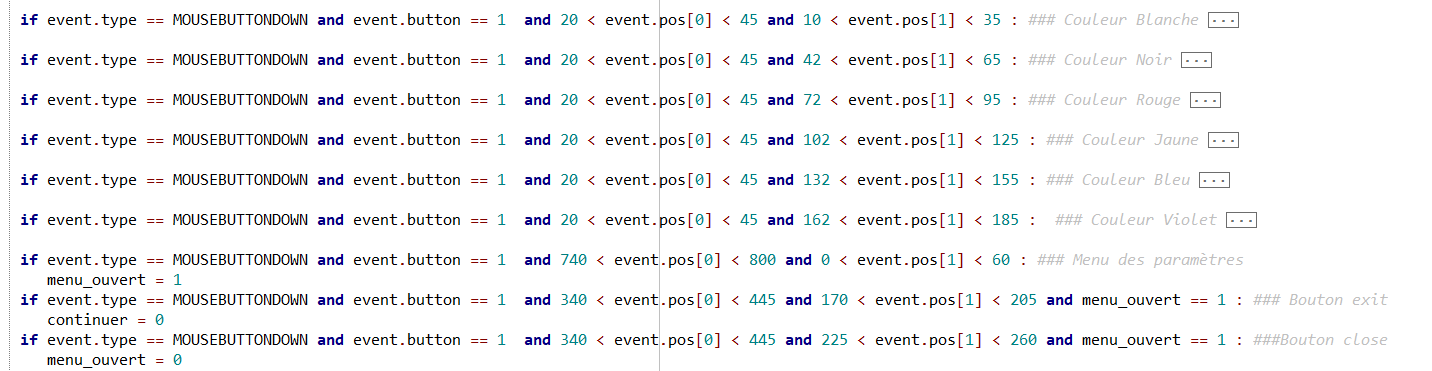
**III - La Boucle while**

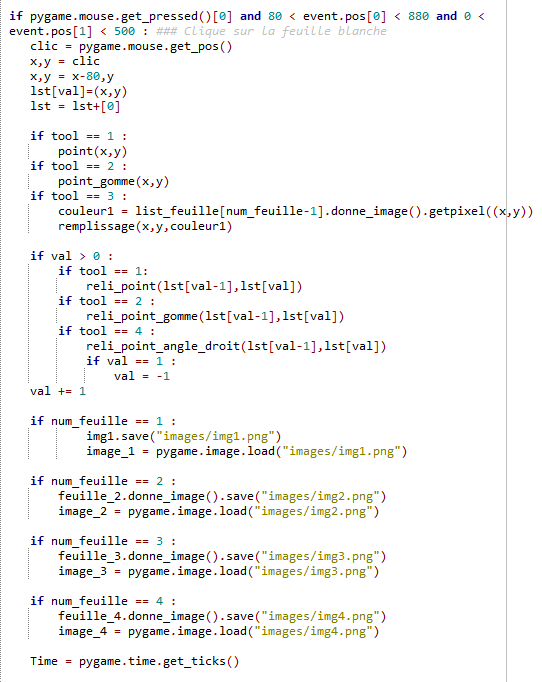
 **A - Les Affichages**

* Utilisation de la commande blit qui prend en paramètre un image en pygame surface et des coordonnées pour pouvoir afficher l’image au bon endroit du display. On retrouve alors les différentes feuilles conditionnées par la valeur de num feuille correspondante, les différentes images de menu et des icônes puis l’affichage des noms des icônes des feuilles avec leurs noms respectifs en fonction de si les objets de classe feuille existent bien et enfin l’affichage de du menu paramètres.

**B - Les Boutons**

* On retrouve d’abord tous les morceaux de code correspondant aux icônes, il utilise notamment les variables tool et num feuille pour changer les affiche et les actions liées aux clique sur la feuille.



* On retrouve aussi les différentes couleurs qui elles font varier la variable couleur et ce qui rend les menu des paramètres effectif.

**C - Get\_pressed sur la feuille**

* On y trouve en premier la récupération des coordonnées au moment du clique qui permettent après qu’ils ai été ajouté à la liste d'exécuter la fonction définie en fonction de la valeur de la variable tool et après si val est supérieur à 0 soit si la liste admet plus d’un élément on utilise les fonction qui relient les points pour relier le point actuel au précédent qui peut être récupérer car il avait été enregistrer dans la liste. Puis on écrase l'image correspondant à la valeur de num feuille qui se trouve dans le dossier par celle que l’on vient de créer et on recharge l’image en pygame surface pour que logiciel comprend le changement et qu'il puisse afficher correctement l’image.

**II – Suivi du projet (par semaine)**

*Le suivi de projet ne regroupe malheureusement pas toutes les étapes de la création du projet mais il reste très complet pour la durée qu’il à pris.*

* Semaine 1 :

*04/05/2022 :* Début du projet, réflexion sur le concept et la façon de le mettre en place + réappropriation des bases de pygame.

*06/02/2022 :* Réalisation d’une ébauche fonction permettant de dessiner ainsi que d’enregistrer le dessin cependant les trait réalisés sont discontinus plus la souris se déplace rapidement, hypothèse : il s'agit peut être d’une répétition trop lente de la boucle while.

* Semaine 2

*07/02/2022 :* Création des première images dont le menu gauche et le menu des paramètres et implémentation des différents boutons ainsi que des affichages de images.

*08/02/2022 :* Tentative de débogage peut fructueuse (les améliorations apportées ne permettait pas de régler le problème). Début de solution permettant de contourner le problème en reliant chaque point entre eux cependant les traits sont pour l'instant rectiligne, forme des angles droits et ne fonctionne que de haut en bas de gauche à droite (fonction reli\_point\_angle\_droit).

*11/02/2022 :* Amélioration de la fonction reli\_point\_angle\_droit qui est à présent utilisable dans toutes les directions cependant elle est mise de côté car je ne vois pas de façon de la rendre plus efficace (elle sera ré utilisée plus tard pour tracer des quadrilatères). Création de la fonction reli\_point qui relient de façon plus directe les points, il faut quand même la peaufiner car elle n’est pas toujours appropriée (ne fonctionne aussi que de haut en bas, de gauche à droite). On note une amélioration évidente, le tracé est beaucoup plus fluide et continu.

* Semaine 3 :

*22/02/2022 :* Amélioration de la fonction reli\_point qui fonctionne à présent dans toutes les directions. + Implémentation des boutons “crayon” et “gomme”.

* Semaine 4 :

02/03/2022 : Correction d’un bug qui gardait la taille du curseur de la gomme alors que l’on avait cliqué sur le crayon qui doit avoir un trait plus fin.

*03/30/2022 :* Ajout d’une seconde feuille ainsi que de la possibilité d’en rajouter à nouveau grâce à la création de la classe “feuille”  ainsi que de l’affichage des différentes feuille à la droite de l’écran.  + Amélioration de la taille du curseur de la gomme qui était trop petit pour effacer efficacement le traits grâce à la fonction “point\_gomme” et “relie\_point\_gomme”.

* Semaine 5 :

08/03/2022 : Création de l’icone de pot de peinture associé qui a pour but de remplir une zone de couleur uniforme avec une autre couleur, problémes rencontrés :

* limite de réception atteint, solution -> augmentation de cette limite
* Après que le programme ai fonctionné avec succès sur des zones de petites taille, il se ferme tout seul lors de l’essaie avec des zones plus conséquente solution possible : utiliser la fonction point à l'intérieur de la fonction remplissage et répéter la récursion tous les 25 pixel (taille d’un point) plutôt que tout les pixel comme jusqu’à présent.

11/03/2022 : création de l’icône new page qui permet de remplacer la feuille sélectionnée par une feuille blanche. + amélioration de la fonction remplissage qui peut à présent remplir des formes qui ne sont pas de la même couleur que la couleur choisie pour remplir.

* Semaine 6 :

15/03/2022 : Programmation de l’outil carré grâce à la fonction préexistante reli point angle qui a été modifié. Il permet donc de créer un quadrilatère(carré ou pavé)  qui prend comme angle un premier clique et comme son angle opposé le second clique. Il est aussi possible de changer la couleur avant et pendant l’opération.   + Tentative de résoudre le problème de la fonction remplissage en limitant le nombre d'exécution récursive maximum sans succès (Corentin en est témoin).

* Semaine 7 :

22/03/2022 : Essaie de programmation d’un outil qui permettra le retour en arrière sur son action cependant les nombreuse erreur suite à divers tentative qui ont toutes conduit à des échecs incompréhensible on conduit à l’abandon du projet (juste cette fonction là).

* Semaine 8 :

05/04/2022 : Malgré l’échec cuisant précédent, une nouvelle idée m’est venue à l’esprit ce qui à conduit à la réouverture du dossier “retour en arrière”. Plutôt que d'enregistrer directement un image à l'emplacement d’une autre pour pouvoir les après coup la récupérer afin de revenir sur une action, j’ai choisi une solution surement plus coûteuse qui consiste à parcourir tout les pixels de l’image enregistrer et à changer un par un tout les pixel de la second. Donc création de la fonction échange qui implémente l’explication précédente ainsi que mise en fonction du bouton de retour déjà présent mais inutilisé. Après des résultats assez concluants on observe un nouveau problème : il faut recliquer à un endroit de l’image pour qu'elle s’actualise, la cause est pour l’instant inconnue. Il reste d'ailleurs aussi à l’étendre à toutes les feuilles (ne fonction que pour la feuille 1).

06/04/2022 : Implémentation du bouton Plus qui était déjà présent dans le projet qui permet d’ajouter des feuilles supplémentaires à la gauche de l’écran (max. 4).