Outils d’ingénierie

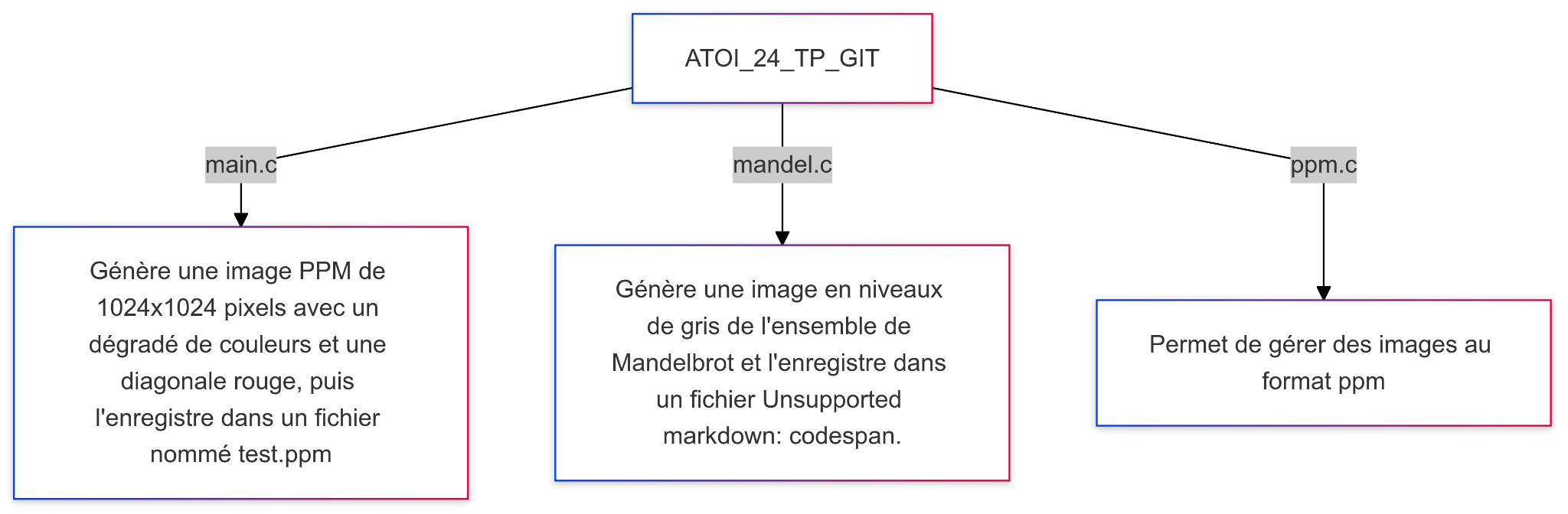
Compte-rendu TP2

**Exercice 1**

1. Configurons les commits à l’aide des commandes ci-dessous :

git config --global user.name "Your Name"

git config --global user.email "[youremail@yourdomain.com](mailto:youremail@yourdomain.com)"

1. Voici un diagramme pour expliquer le rôle de chaque fichier .c : 
2. Nous avons rajouté des commentaires sur le fichier ppm.h et avons commit sur la branche master.

**Exercice 2**

1. Nous avons corrigé mandel.c et le Makefile puis commit les modifications sur GitHub.
2. Le programme commence par définir les paramètres nécessaires pour créer une image représentant l'ensemble de Mandelbrot, notamment la taille de l'image, un seuil pour déterminer si un point diverge et un maximum d'itérations pour les calculs. Il initialise ensuite une image PPM qui servira à stocker les résultats. Chaque pixel de l'image est associé à un point sur une grille, avec des coordonnées calculées par des fonctions spécifiques.

Pour chaque pixel, le programme effectue des calculs en boucle pour déterminer si le point associé reste stable ou s'éloigne au-delà d'une certaine limite. La couleur du pixel est ensuite ajustée en fonction du résultat, avec des nuances de gris qui varient selon le comportement du point.

À la fin, l'image générée est enregistrée dans un fichier nommé m.ppm, et les ressources allouées pour la construction de l'image sont libérées.

**Exercice 3**

1. Pour ramener la branche color\_support au dessus de la master, on fait un merge.
2. Il est nécessaire de revérifier chaque fichier manuellement pour gérer les conflits et corriger le code
3. On balaye chaque fichier ligne par ligne. Ce qui est entre <<<<<<<HEAD et ======= appartient au fichier courant, ce qui est entre ======= et >>>>>>>color\_support appartient à ce qui était dans la branche color\_support.   
   Ensuite on choisit les parties du code qui nous intéressent, en prenant soin d’enlever les <<<<<<<HEAD et ======= pour que le fichier compile.

Une fois le tout fait, on valide la résolution des conflits avec un add <fichier>.

1. Enfin, on commit et on push le tout sur le dépôt distant.

**Exercice 4**

1. Nous avons créé un repo GitHub dans lequel nous avons enregistré les modifications apportées aux fichiers
2. Pour envoyer le travail dans le dépôt distant, on utilise les commandes : git add <fichier>, git commit -m “<message>” et git push.
3. Lien du repo : https://github.com/WolfoS92/Outils\_inge\_TP2