

Pour réaliser l'application avec l'IA générative CLAUDE, voici les prompts que nous avons utilisés. Voici la liste de toutes les demandes utilisées pour pouvoir avoir l'application complète.

1. Liste des prompts utilisés

1)Créer une application Streamlit qui génère un Diagramme de Voronoï coloré à partir d'un fichier source (JSON ou TXT), sans utiliser de bibliothèques de haut niveau pour le calcul géométrique (interdiction d'utiliser `scipy.spatial.Voronoi` ou `Delaunay`). Algorithme : Tu dois implémenter manuellement la Triangulation de Delaunay via l'algorithme de Bowyer-Watson.

2)Image fournie] C'est pas normal, ya des points qui n'ont pas de zone.

3)Fait un fichier de tests unitaires.

4)[Erreur] `FileNotFoundException` : le fichier de test ne trouvait pas `voronoi_app.py` car il s'exécutait depuis le dossier `tests/` et non depuis la racine du projet.

5)[Erreur] `ValueError` — palette 'earth' : not enough values to unpack (expected 3, got 0). La palette utilisait `mcolors.to_rgb()` de `matplotlib`, qui était mocké dans les tests.

6)Je veux que tu fasses du clean code et qu'il y ait une architecture.

7)[Erreur] `ImportError` : cannot import name 'bowyer_watson' from 'algorithms' (unknown location). Streamlit ne trouvait pas le package local car il n'exécutait pas depuis la racine du projet.

8)Il n'y a pas de fichiers from `algorithms` import `bowyer_watson`, `compute_voronoi`.

9)Donne un fichier JSON et un fichier TXT à mettre dans un dossier data.

2. Bilan de la session

Pour réaliser l'application avec CLAUDE, il aura fallu environ 40 à 50 minutes pour obtenir un résultat satisfaisant et architecturé. Le plus difficile était la correction du bug des cellules de Voronoï manquantes sur les points de l'enveloppe convexe. Pour que CLAUDE comprenne précisément le problème, il a fallu lui fournir directement une capture d'écran du diagramme défectueux montrant les points sans zone colorée.

Points marquants de la session :

- **Implémentation correcte dès le 1er prompt** — CLAUDE a immédiatement produit Bowyer-Watson sans reformulation.
- **Bug détecté via image** — Les points sur l'enveloppe convexe n'avaient pas de cellule. CLAUDE a identifié la cause (cellules non-bornées) et l'a corrigée en ajoutant des rayons infinis clippés par Sutherland-Hodgman.
- **Clean code sur demande** — L'architecture en 4 packages (`geometry`, `algorithms`, `loaders`, `visualization`) n'a été produite qu'après une demande explicite.
- **Tests unitaires complets** — 80 tests couvrant tous les modules, avec gestion correcte des mocks Streamlit/Matplotlib.

- **Corrections de configuration** — Plusieurs erreurs liées à l'environnement (sys.path, conflit de nom io, palette mcolors mockée) ont nécessité des échanges supplémentaires.

3. Conclusion

Le développement humain et le développement assisté par IA sont différents. Par exemple, lors de la compréhension d'un besoin, un humain essaie de comprendre le contexte et se pose des questions sur la façon de faire, tandis que l'IA ne fait que produire du code sans réellement comprendre la logique ce qui mène à des code avec des erreurs.

Un humain va écrire un code propre et bien architecturé qu'alors l'IA ne le fait que quand on lui demande. Codé avec l'IA nous donne un résultat rapide qu'alors un code humain sera plus qualitatif car le contexte est compris, la propreté du code va limiter les bugs ou les trouver plus rapidement. Pour un débutant, générer du code avec IA sans rien comprendre va engendrer beaucoup de problèmes et prendre plus de temps car à un moment l'IA sera limitée(manque de contexte) et on ne pourra pas comprendre ce que l'IA a mal fait.