II. Mission

Cette semaine l'objectif principal pour moi était l'optimisation du code. Pour le moment on peut bien obtenir le résultat souhaité (Les géodésiques de 5 cyclides imbriqués), cependant cela nécessite énormément de ressources de calcul, et le résultat obtenu est très difficile à manipuler, à cause du nombre trop élevé d'objets à traiter.

Nous avons d'abord envisagé de relier les segments de géodésiques entre eux afin de n'obtenir qu'un seul objet pour x segments reliés, au lieu d'obtenir x objets pour x segments tracés.
Cela fonctionnait mais le résultat ne convenait pas à Mr Jouk car les géodésiques obtenues ne sont pas conformes à la représentation des fibres du ventricule droit, on doit bien distinguer des « batônnets », non pas des morceaux de segments reliés entre eux.

Cette solution nous avait pris plusieurs jours à développer mais elle n'a pas été retenue, et je n'ai malheureusement pas pensé à conserver des traces des résultats.

Par la suite j'ai envisagé d'inspecter le script plus en détails, et je me suis rendu compte que la méthode utilisée pour choisir les sommets P1 P2 P3 génère parfois des combinaisons de sommets identiques. Cela peut se produire lorsque plusieurs paires de sommets génèrent des cercles de rayon minimal.
J'ai donc créé une clé unique pour chaque combinaison de sommets. La clé est obtenue en triant les indices des sommets et en les regroupant dans un tuple. Ainsi, même si les sommets sont choisis dans un ordre différent, la clé sera la même.

Puis, chaque fois que je créée une géodésique, je vérifie d'abord si la clé correspondante existe déjà dans l'ensemble :

```
generated_geodesics = set()  # Set pour stocker les
géodésiques générées
```

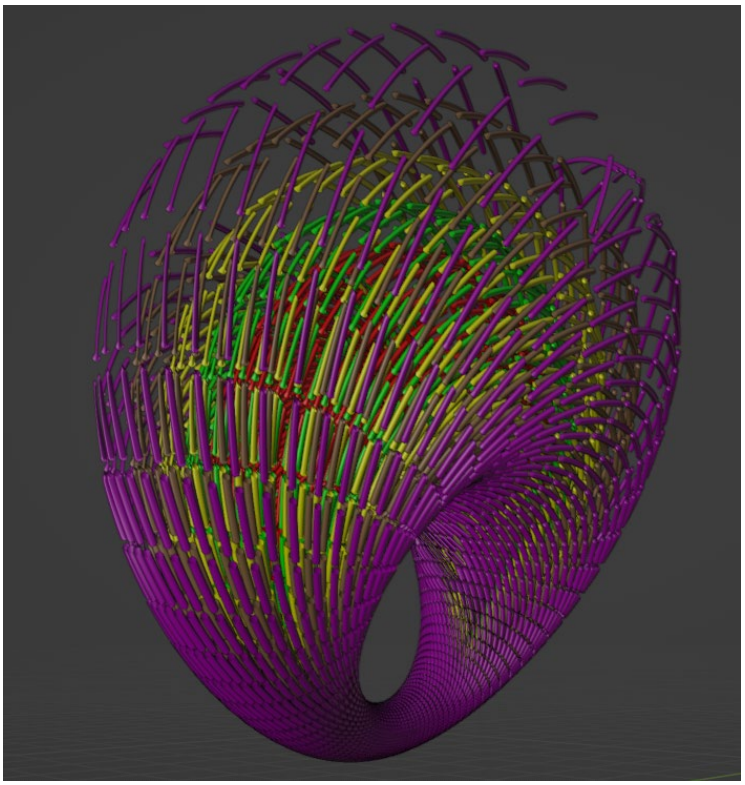
Voici mes boucles permettant de tracer les géodésiques, j'ai ajouté une condition if à la fin pour vérifier si la clé existe déjà dans l'ensemble.

```
for i in range(nb_vertices):
        M = 11111111110
        P1 = verts[0]
        P2 = verts[0]
        P3 = verts[0]

        for t in range(1, nb_vertices):
            if (i % nb_cercles + t < nb_cercles and i % nb_cercles - t >
0):
                Point1 = verts[(i - nb_cercles + t) % nb_vertices]
                Point2 = verts[i]
                Point3 = verts[(i + nb_cercles - t) % nb_vertices]
                R = calcul_parametres_cercle(Point1.co, Point2.co,
Point3.co)
                if R < M:
                    M = R
                    P1 = Point1
                    P2 = Point2
                    P3 = Point3

        # Générer une clé unique pour chaque géodésique
        key = tuple(sorted([P1.index, P2.index, P3.index]))

        if key not in generated_geodesics:
            # Créer la géodésique uniquement si elle n'a pas déjà été
générée
            create_geodesic_spline(P1, P2, P3, 1, 0.005, liste_couleurs[c])
            generated_geodesics.add(key)
```

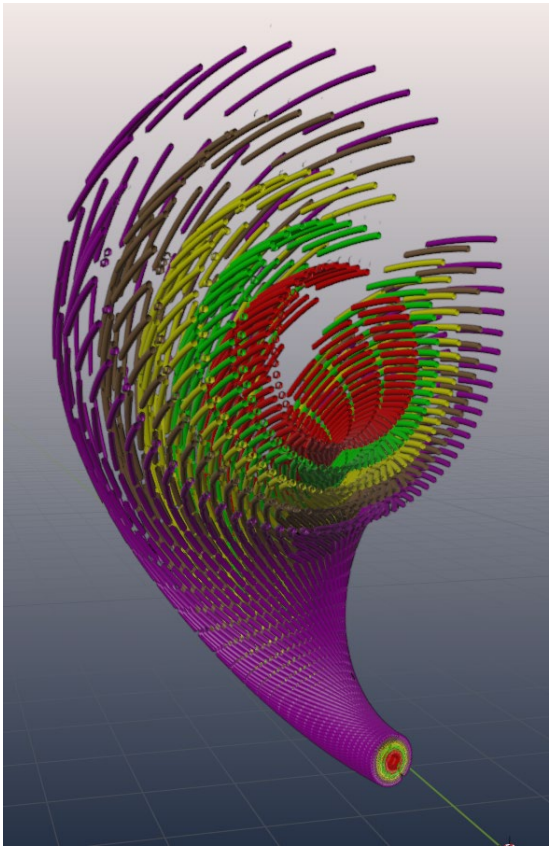
Si la clé existe, cela signifie que cette géodésique a déjà été générée et je l'ignore donc pour éviter les doublons. En utilisant cette approche j'ai donc pu m'assurer que chaque géodésique est générée une seule fois, même si les sommets sont choisis dans un ordre différent.

Grâce à cette amélioration, le nombre d'objets créés dans la collection est divisé par 2.
Ainsi, je peux modéliser des géodésiques sur 5 cyclides, et en définissant encore plus de sommets afin d'avoir un rendu plus précis, plus visuel. Auparavant je ne modélisais que des cyclides à partir de tores de taille 48*48 au maximum (c'était déjà beaucoup) maintenant je peux me permettre de modéliser des cyclides à partir de tores de taille 60*60, d'en imbriquer 5, et également d'augmenter l'épaisseur des géodésiques afin d'avoir un meilleur rendu visuel.

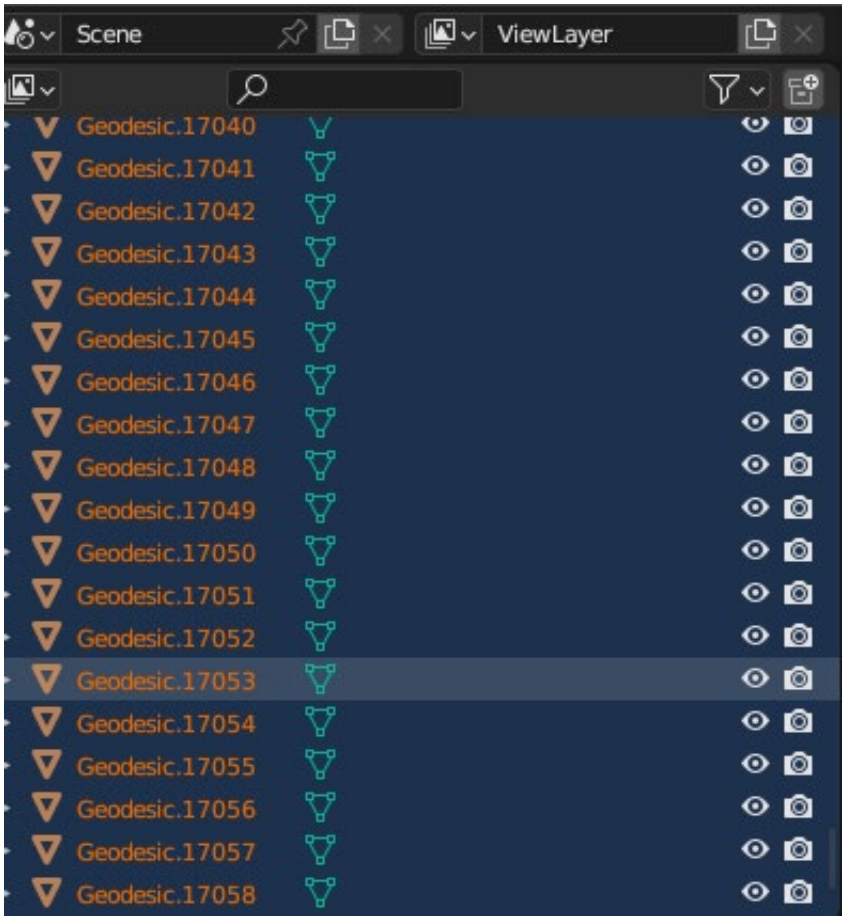
Voici quelques images du rendu :



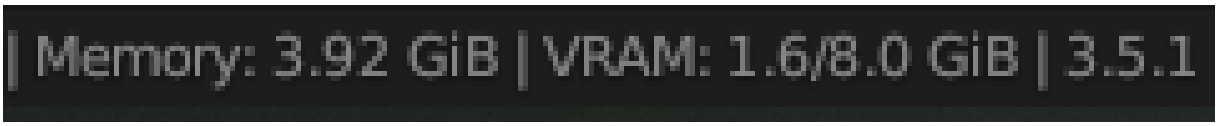
Une coupe dans l'axe vertical :



Le nombre de géodésiques générées au total est de 17058, on avait auparavant 13000 géodésiques environ pour seulement 3 cyclides imbriqués de taille 48*48.



La consommation en RAM et VRAM du modèle présenté au-dessus :



Le résultat est donc très satisfaisant, et nous permet à la fois de générer des modèles plus détaillés, mais également de les manipuler en ayant des temps de traitement plus courts.