Rapport de Projet
IGSD

Lubin

Longuépée

Aidan Barouk

2024

I. Organisation du travail :

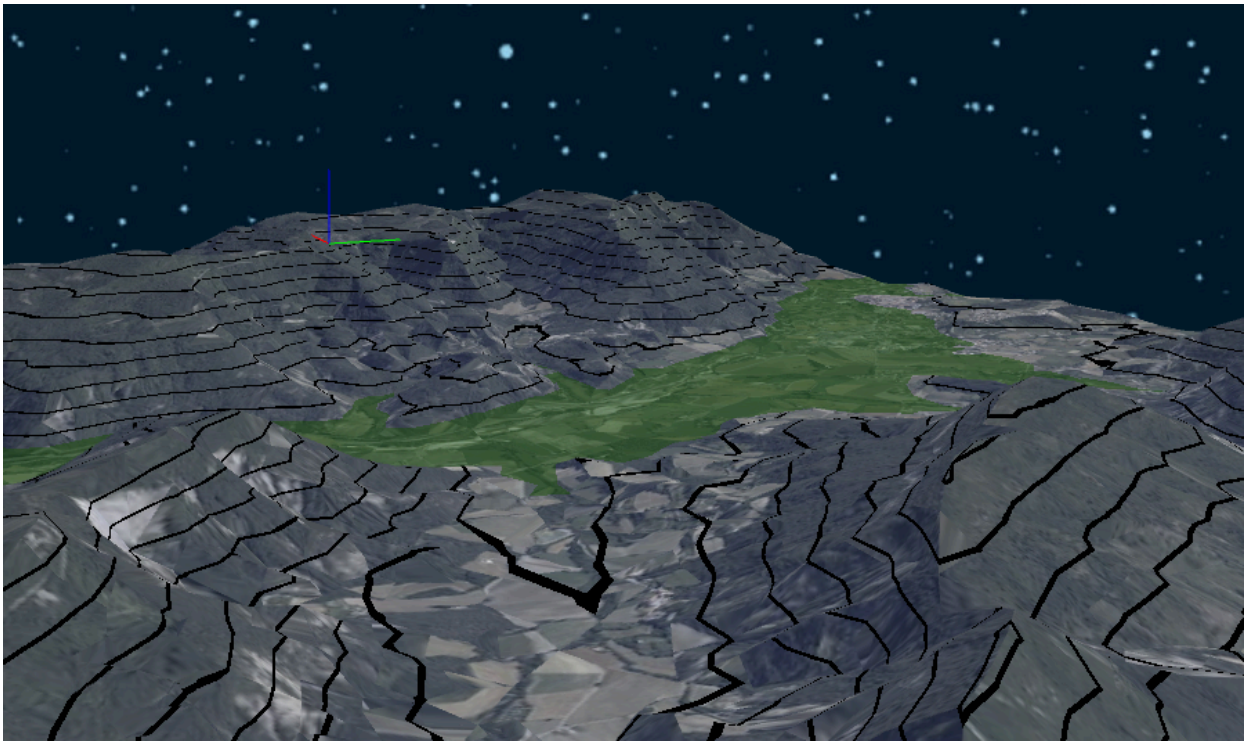
Nous avons décidé de faire la majorité du projet durant les séances de TP. Avoir des séances de TP de 4 heures a aidé à pouvoir finir en quasi-totalité le projet pendant les heures de cours. Nous avons fait une petite partie du travail de chez nous, en travaillant en simultané sur la même chose en s'appelant. Nous avons fait le choix de ne pas travailler séparément et donc de ne pas diviser le travail car nous estimions avoir besoin de tout faire à deux pour être efficace et bien comprendre le code que nous écrivons.

II. Choix d'Implémentation des Fonctions

1. Setup et Configuration Initiale

Nous avons dans la fonction setup initialisé nos PShape pour ne pas avoir à re-exécuter nos fonctions de dessin d'éléments à chaque appel de la fonction draw().

Les shaders sont appelés dans la fonction setup. Ceux-ci affichent des lignes horizontales sur tout le relief de la carte, et le bas de la carte est "peint" en vert, comme le montre l'image ci-dessous. Les shaders regardent les coordonnées Z interpolées, et si ces coordonnées sont en dessous de la coordonnée -199, la couleur est changée en vert. Et si les coordonnées sont comprises entre 0 et 0.15 mod 2, la couleur est changée en noir.



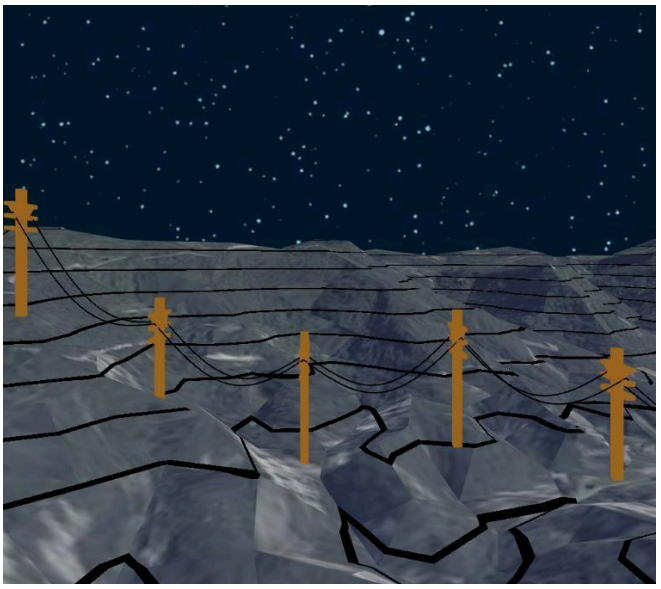
2. Contrôle de la Caméra et Interaction

Les fonctions keyPressed() et keyReleased(). Ces fonctions gèrent les entrées du clavier pour déplacer la caméra dans la scène, permettant à l'utilisateur de naviguer de manière intuitive (avancer, reculer, gauche, droite). Quand on appuie sur Z, on avance, quand on appuie sur S, on recule etc.… Nous avons fait en sorte de pouvoir se déplacer en restant appuyé sur la touche, pour ne pas avoir à appuyer constamment sur les touches de déplacement. Quand on appuie sur espace, la caméra monte. A l'inverse, quand on appuie sur la touche shift (Majuscule), la caméra descend.

Les fonctions mouseWheel() et mouseDragged(). Ajustent la distance de la caméra par rapport au terrain et la direction de la caméra, respectivement. Ces interactions renforcent l'immersion de l'utilisateur en permettant un contrôle fluide de la vue. Quand on tourne la molette vers le haut, la caméra monte, tandis que quand on la tourne vers le bas, la caméra descend. Quand on appuie sur un des boutons de la sourie, et qu'en parallèle on tourne la souris horizontalement, cela permet de déplacer la caméra en suivant les déplacements de la souris

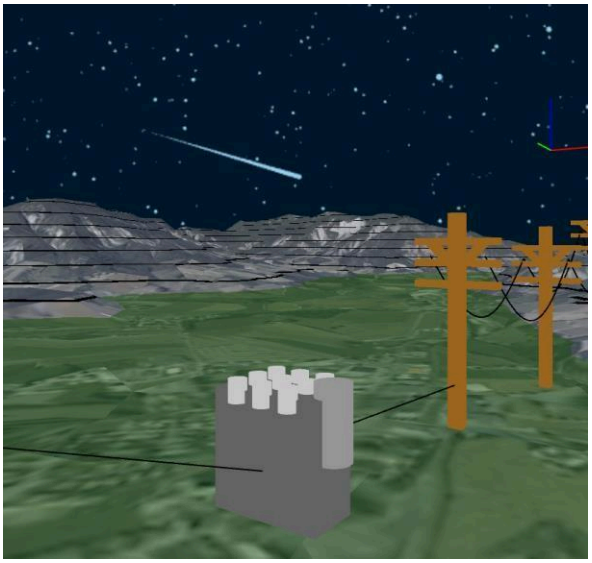
3. Rendu Graphique

La fonction draw(). Dans la fonction draw, nous affichons tous nos éléments de décor. Nous dessinons principalement les fils électriques entre les poteaux et les éoliennes et transformateurs. Pour cela, nous avons fait le choix de diviser notre ligne en 50 petites lignes que nous incurvons pour donner un effet de courbure (due à la gravité) à nos lignes électriques. Voir image ci-dessous



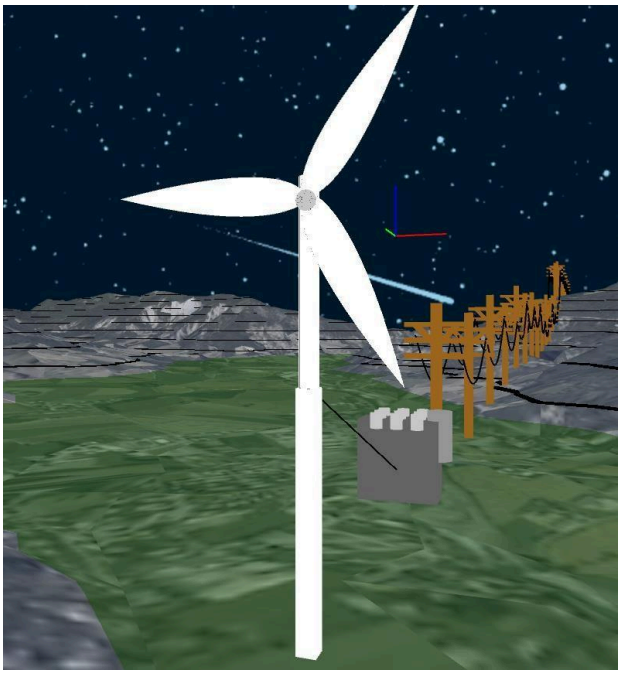
La création des poteaux électriques à l'aide de la fonction createPoteaux. La fonction renvoie un PShape contenant un pylône électrique. Nous les avons représentés comme des poteaux en bois sur lesquels nous ajoutons des barres horizontales pour ressembler à de vrais poteaux. Nous nous sommes inspirés de l'image ci-dessous pour construire notre poteau électrique. Nous dessinons ensuite le PShape grâce à la fonction drawPoteauElectrique(). Nous avons en fait créé une fonction PlacePoteauxElectriques qui dessine un nombre N de poteaux entre les coordonnées données en paramètre grâce à la fonction drawPoteauElectrique.

La création des transformateur électrique à l'aide de la fonction dessinTransformateur(). De la même manière, nous avons créé une fonction qui renvoie un PShape contenant un transformateur électrique. Nous avons fait en sorte que celui-ci ressemble à l'image ci-dessous. Nous avons ensuite affiché ces transformateurs grâce à la fonction drawTransfo().



La fonction getTerrainAltitude() permet de calculer l'altitude du terrain à une position (x, y) donnée. Cela est utile pour positionner tous nos éléments de décor.

Et enfin la création des éoliennes grâce à la fonction drawEolienne. Cette fonction dessine nos éoliennes aux coordonnées et à la taille donnée en paramètre. Cette fonction crée ainsi la base de l'éolienne, puis un cylindre au- dessus de cette base, puis les pales de l'éolienne au sommet de ce cylindre. Nous avons aussi rajouté un cône entre les 3 pales pour ressembler à une vraie éolienne, comme le montre l'image ci-dessous.



III. Comment on a ressenti le projet ?

Nous avons eu du mal à démarrer et comprendre comment aborder les enjeux du sujet, mais une fois que nous avons commencé la partie des poteaux électriques, nous avons eu un déblocage et nous avons commencé à être à l'aise avec le sujet. Dans l'ensemble, le projet nous a appris des choses et nous a rendu plus à l'aise avec Processing.

IV. Les difficultés rencontrées

Nous avons eu du mal sur la partie avec les shaders car nous ne comprenions pas comment les utiliser. Cependant, à l'aide de la documentation trouvée sur internet et l'aide de certains camarades, nous avons appris à les maitriser et nous avons finalement réussi à avoir l'effet attendu.

De plus, nous avons eu comme première idée pour la réalisation des fils électriques de créer des courbes de Bézier entre chaque poteau pour donner l'impression d'un fil courbé. Mais nous nous sommes rendu compte que trouver les bons points de contrôles pour chaque courbe était très difficile à faire et nous avons finalement envisagé une autre solution. Nous avons décidé de réaliser 50 lignes droites à la suite, mais en leur donnant des coordonnées différentes grâce à la fonction sinus, qui permet de donner un effet arrondi à nos courbes.

V. Ce que nous aurions pu rajouter

On aurait aimé faire des poteaux électriques plus réalistes et plus détaillés mais nous n'avons pas eu le temps de le faire. Nous aurions aussi aimé créer des habitations comme des maisons ou des immeubles dans la vallée, mais malheureusement, nous avons également manqué de temps.