Rapport de Projet

Des pylônes dans les collines

Par Antoine EL-KASSIS et Rayan LALAOUI

Brève Description :

L'objectif du projet était de modéliser sur une carte préexistante de st Aubin, une ligne électrique haute tension reliant la ville à un champ d'éolienne situé sur des collines situé non loin de celle-ci. Le projet contient 4 fichiers .pde **:** main, Eolienne , Lignes , pylon PShape **en plus des shaders.**

Fonctionnalité et éléments implémentées :

- Camera avec déplacement et changement d'orientation
- Pylônes éléctrique
- Lignes Haute tension
- Éoliennes
- Shader simple (coloration de la vallée et ligne de niveau)
- Shader avancé (cycle saisonnier)

Details du projet

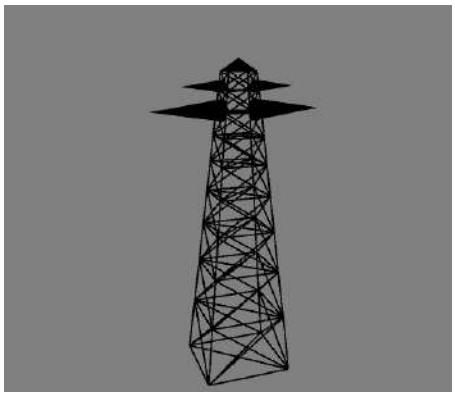
Camera

Pour modifier l'orientation de la camera il suffit d'effectuer un mouse drag (click + déplacement de la souris), cela permet de ne pas avoir un déplacement constant de la vue.Pour ce faire on utilise des coordonnées sphérique et on calcule à chaque mouse drag les variations de la longitude et de la latitude en fonction de la position de départ et d'arrivé de la souris relativement à la taille de l'écran.

Deux modes de déplacement sont possibles, l'un avec les flèches directionnelles permet de se déplacer en suivant le repère cartésien, l'autre avec W et S permet de se déplacer en suivant la direction de la camera vers l'avant ou l'arrière respectivement.

La camera est en mode perpective() **des zooms en été ajouté avec les touches L et K (augemente ou baisse le FOV).**

Pylône



Un pylône est représenté par un PShape global 'Pylone' **qui regroupe plusieurs PShape :**

- **les blocs de pylônes appelés "pylon_block".**
- **3 cônes.**

Nous avons codé le pylône de manière à pouvoir lui donner le nombre de blocs et la taille du pylône à l'avance en fixant les attributs "size" **et** "nb_Pylons" **qui seront initialisés dans** setup() **du fichier "**Main"**.**

La fonction void createPylonBlock(float size, PShape pylon_block, int nb_Pylons) **ajoute des segments a un PShape** "pylon_block" **avec la forme des segments de fers d'un vrai pylône.**

La fonction createPylonBlocss(float size, int nb_Pylons) **rends un array de** "pylon_block" **qui seront utilisés pour la création du pylône.**

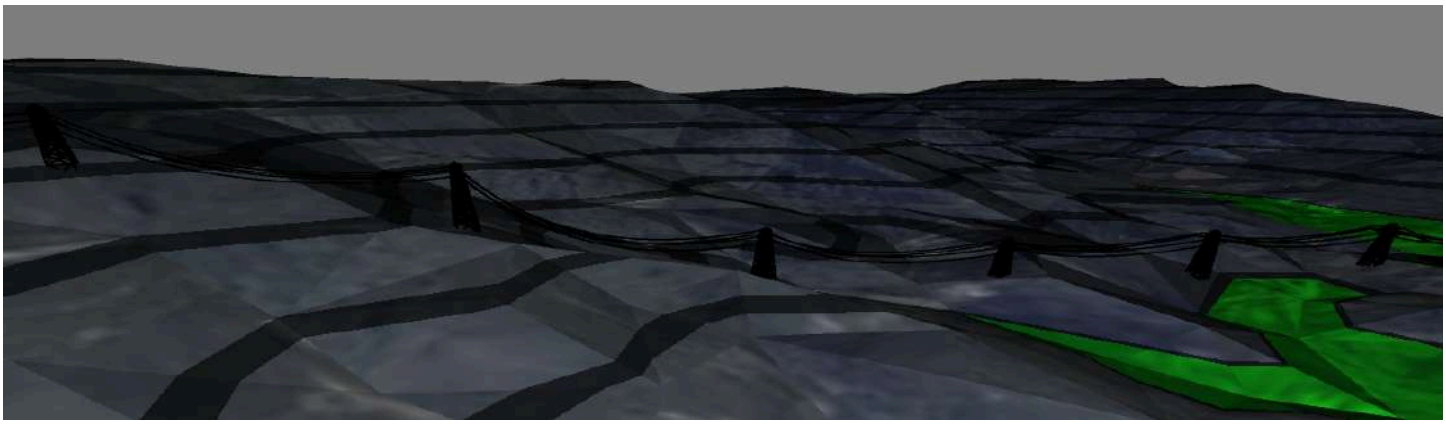
Les cônes qui représentent les bras et le chapeau du pylône sont ensuite ajoutés manuellement grâce à la fonction myCone(float sideSize, int nbSide, float x, float y, float z,float w, int dir) **(où w est un attribut fixant la largeur du cône par une relation de proportionalité avec** "sideSize" **et dir la direction vers laquelle le cône est orienté).**

La création d'un pylône dépend d'un angle de rotation appelé "angle_mort"**, qui est calculé à l'avance pour orienter correctement les pylônes.**

En effet, nous avons construit nos pylônes et nos lignes de manière à ce qu'ils s'affichent correctement, indépendamment du point de départ et du point d'arrivée.

Enfin, on dessine les pylônes grâce à un appel à la fonction shape()**.**

Ligne Electrique



Une ligne électrique est représentée par un PShape qui regroupe une succession de PShape de segments très petits connectés. Il y a trois fonctions qui créent trois types de lignes :

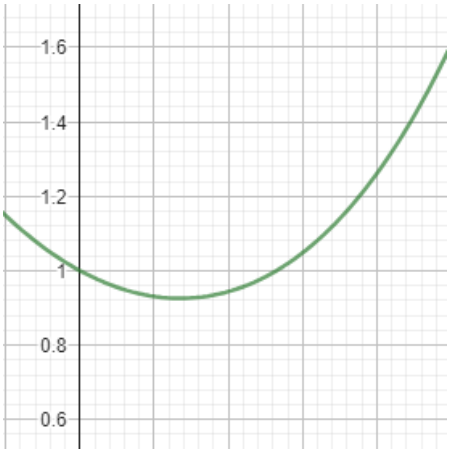
- create_ligne(LinkedList<PVector> coords, float angle_rotation, float size, int nb_Pylons): **cette fonction crée la ligne qui passe par les bras des pylônes dessinés. Elle prend comme arguments la liste des coordonnées des pylônes, l'angle de rotation des pylônes, la taille d'un bloc de pylône et le nombre de blocs de pylônes par pylône. Elle renvoie la ligne qui est supposée passer par le bras bas droit des pylônes. Les trois autres lignes sont dessinées en effectuant les translations nécessaires.**
- create_ligne_box(float angle_rotation, PVector coords, boolean down, boolean right, PVector c, float size, int nb_Pylons): **cette fonction crée une ligne d'un pylône vers un bloc qui représenterait le transformateur électrique (d'où le nom de ses coordonnées optimus_Prime). Les booléens down et right indiquent de quel bras du pylône commencer la ligne.**
- create_ground_ligne(PVector src, Eolienne tgt): **cette fonction crée des lignes qui passent de** src **à** tgt **par terre. Elles représentent les lignes électriques entre les éoliennes et le transformateur.**

La ligne entre les pylônes est stockée dans un PShape appelé "Ligne"**, tandis que les autres sont stockées dans une liste de PShapes appelée** "lignes_box".

La fonction utilisée pour donner un effet de gravité aux lignes fines est :

f(a,x,b) = cosh(a.x)+x(1-cosh(beta))-1;

- **Cela ferait varier la raideur de la fonction, calculée en fonction de la distance entre deux pylônes et de la différence en Z** « difZ ».
- **Cela représenterait à quel point la ligne électrique serait tendue. On peut la tirer beaucoup ou peu, selon ce qui se trouve en dessous de nous (si le câble touche un rocher, on essaie de le tirer, bien sûr, avec une limite).**



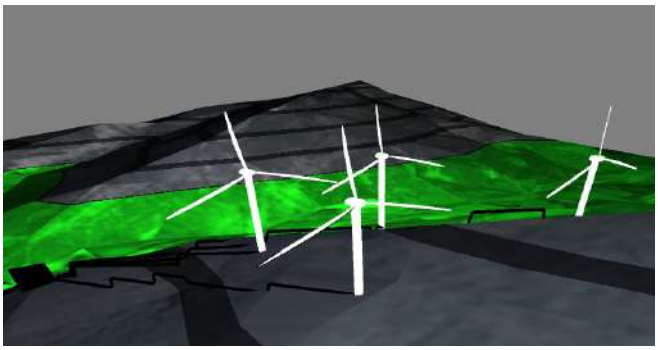
Notez bien que la fonction ici représente f(a,x,b) + 1 et non f(a,x,b), juste pour la représentation dans ce cas.

Remarquez que notre fonction suppose que pylone1 est en dessous de pylone2. Le premier pylône se situerait à l'abscisse 0 **et le deuxième pylône se situerait à l'abscisse dist(pylone1, pylone2).**

Dans le cas contraire, on remplace difZ **(qui serait plus petite que** 0**) par** -difZ **et on effectue les calculs de f en allant de** 0 **à** nb_points **au lieu de de nb_points à** 0**.**

Autrement dit, on suppose que l'abscisse de p1 serait en dist(pylone1, pylone2) et l'abscisse de p2 en 0**, et on effectue les calculs conformément.**

Éoliennes



Une éolienne est représentée dans la classe homonyme par 3 PShape principale : pales,support,poteau **. Tous sont contenus dans la PShape** fome**.**

Dans le même fichier se trouvent les fonctions:

- helice(float r, float x , float y,float z)**: créer une pale à l'aide de bezierVertex aux coordonnées indiqué.**
- poteau(float h , float r)**: Crée le poteau qui va supporter les pales.**
- createSupport(float cylinderRadius , float cylinderHeight)**: Crée un support d'attache pour les pales sur le poteau.**

Deux fonctions auxilières sont utilisées régulièrement pour la génération de ses PShape: drawCylinder(float cylinderRadius , float cylinderHeight) **et** drawCircle(float radius, float y, float z) **qui renvoie respectivement les cotés d'un cylindre et un cercle faisant office de couvercle pour le haut ou le bas.**

La méthode drawEolienne() **permet de dessiner l'éolienne (avec un appelle à** shape(forme)**) quand elle est appelée dans** draw()**, elle fait également marcher l'éolienne en faisant tourner les pales en fonction du** frameCount**.**

Les éoliennes peuvent être masquées en appuyant sur U. Elles sont également reliées à un poste de transformation qui lui est générer dans le fichier Lignes.pde.

Shaders

Pour le shader, nous somme parti sur un shader qui gère à la fois la lumière et les textures. Le vertex shader ne fait rien de particulier à ce niveau là sauf passez la composante sur l'axe z dans le varrying float z **qui sera plus tard utilisé dans le fragment shader pour calculer les lignes de niveaux. Mais ce la permet déjà d'améliorer l'ombrage par rapport à l'angle de la camera ce qui fait mieux ressortir les reliefs.**

Dans le fragment shader on initialise la hauteur de la vallée à zmin + (zmax-zmin)/5 et un vec4 color **qui servira à interpoler la couleur finale.**

Pour les lignes de niveau, on prend la composante z (qu'on mutiplie par 2 car la carte est relativement tassée en hauteur) qu'on convert en entier puis on calcule le reste modulo 5. Les couleurs sont alors affichées à 40% de leur intensité donnant les lignes sombres.

Quand la composante z du vertex est inferieur à la hauteur de la vallée les composantes r,b de la couleur sont réduites de ¾ alors que la composante g est multiplié par 2 donnant une vallée verdoyante.

Pour le shader des saisons, on passe la variable frameCount%360 **au** fragment shader**, puis on effectue un découpage suivant le jour de l'année.**

- **En hiver, la neige descend vers la vallée puis remonte vers la fin**
- **La vallée commence a verdir au avant la fin de l'hiver puis remonte le long du printemps**
- **En été, la couleur passe du vert au doré. Puis la vallée perd en hauteur annonçant l'hiver prochain.**

Tous ça et gérer pas des if **en cascade, les dégradés de couleurs sont linéaires et toujours calculer dans** color**. Malheureusement on a pas pu travailler suffisamment sur le rendu de celui-ci.**

Difficultés Rencontrées

- **Une des premières difficultés rencontrées est liée au shader, car certaines parties un peu trop escarpé de la carte dans la vallée deviennent noires, ceci est probablement lié à la gestion de la lumière à ces endroits là.**
- **La 2ème plus grosse difficulté aura été de trouver une fonction adéquate pour tracer les lignes électriques de manière crédible.**
- **Ce qui amena la difficulté suivante, être capable de renvoyer les coordonnées des points d'accroches avec précision car nos PShape pour les pylônes dans leur construction son créer en biais ce qui fait qu'avant de les dessiner ils doivent être tourner ce qui complique la récupération des points d'accroches telle que modéliser dans le** draw() **principale.**