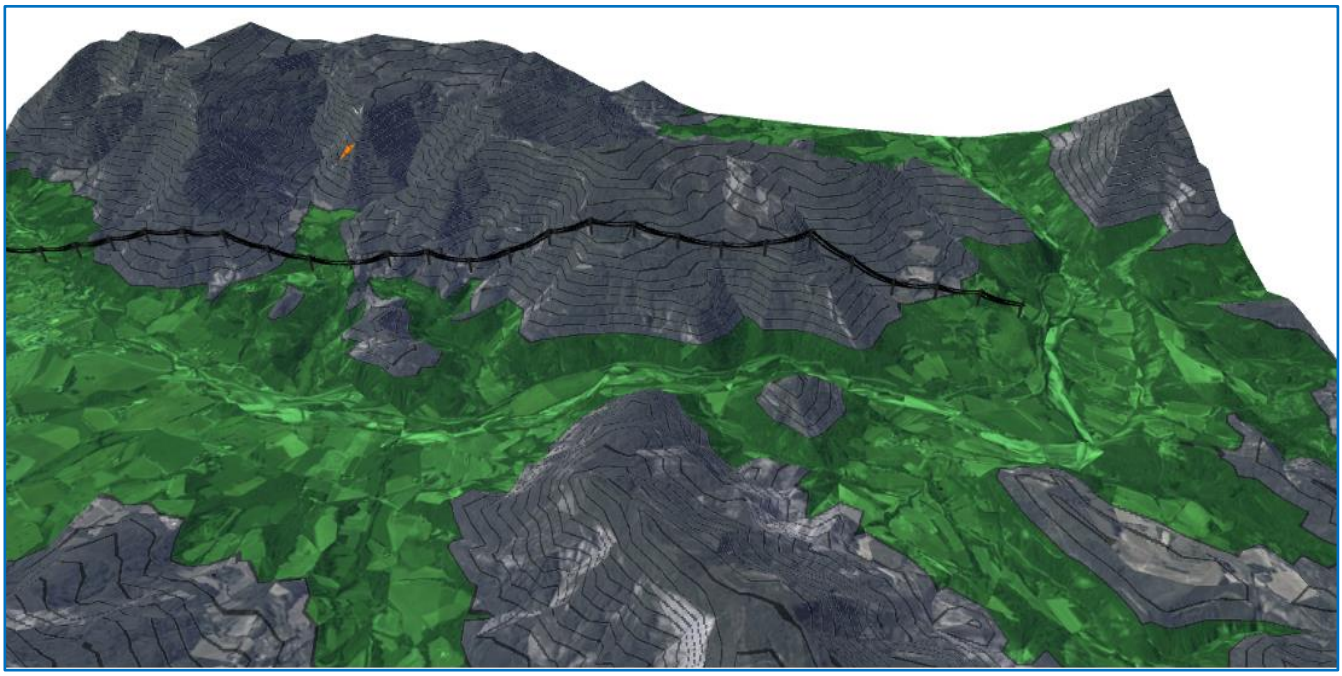
université
PARIS-SACLAY

Des pylônes dans les collines

Informatique graphique pour la science des données



Binôme :

- SAIL Ramy.
- CHABANE Oualid.

Encadrée par : Mme. Capucine Nighem.

2ème année licence informatique.

Année universitaire : 2023-2024

université
PARIS-SACLAY

Table des matières

université
PARIS-SACLAY

Introduction

Notre projet représente une application 3D, qui affiche un terrain texturé représentant une vallée entourée de collines et qui possède des pylônes liés par des lignes électrique et des éoliennes placées en haut des collines qui eux sont lié à la ville.

Composition code source

Pour faciliter la manipulation du code source il est décomposé en plusieurs modules :

- **Projet.pde** : module principale qui contient le setup et le draw avec une fonction qui dessine les axes du repère du terrain
- **eolienne.pde, pylônes.pde, ligne_electrique.pde** : permette de générer les PShape pour les éoliennes, pylônes et lines électrique, les PShape retournés sont en coordonnées locales (le (0,0,0) est relatif au formes).
- **creationEolienne.pde, creationLignesElectriques.pde, creationPylone.pde :** utilise les modules précédents pour générer le formes et font les transformations nécessaire pour ramener les PShape au coordonnées do monde (notre terrain).
- **creationModelePylonesLignes.pde :** créer et renvoi le modèle contenant les 25 pylônes et les lignes électriques qui les relies dans un PShape(GROUP)
- **calculeAltitude.pde :** contient les fonctions nécessaire pour calculer l'altitude d'un point (x,y) donné.
- **deplacement_Cam.pde :** contient l'affectation des touches au déplacement de la caméra sur le terrain, et aussi les touches pour afficher et cacher les modelés de pylônes et de lignes .

Partie 1 :

La première tâche réalisée était de charger le terrain dans un PShape et l'afficher, les informations sur les coordonnées entre lesquels le terrain est dessiné sont : Z entre

-202.09592 et -179.59933 donc une épaisseur de 22, X entre -135 et 127 ce qui fait 262 en largeur, en Y entre -158 et 159 ce qui fait 317 en hauteur.

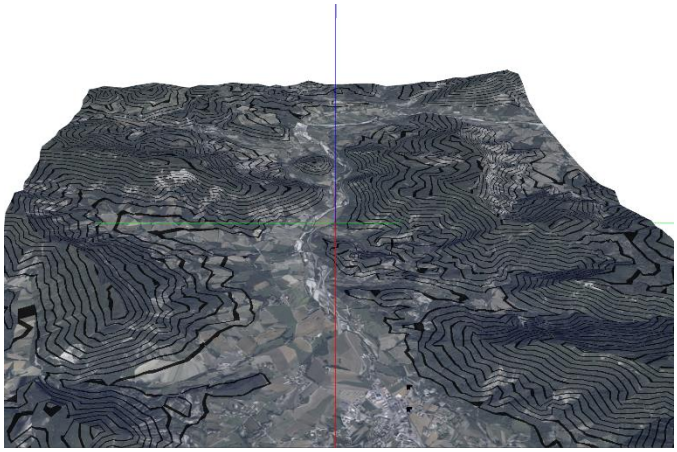
On positionne la caméra sur le terrain :

camera(xPosCam, yPosCam, zposCam, 0, -1, zDirCam, 0, 0, -1);

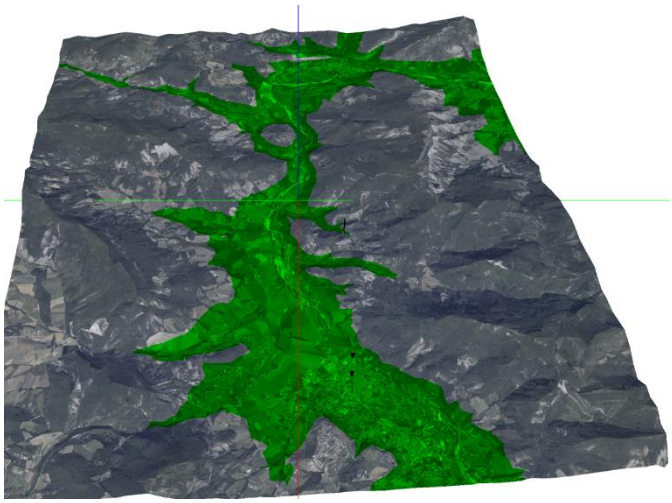
On inclut aussi les déplacements de la caméra pour faciliter le débugge et se déplacer sur le terrain pour vérifier chaque nouvelle tâche réalisée, avec les touches : flèche du haut pour le déplacement dans le Z négatif (Zoomer), flèche du bas pour le déplacement dans le Z positif (dézoomer), flèche droite pour le y positif, flèche gauche pour le y négatif, touche a pour le x négatif et touche z pour le x positif.

Partie 2 : Shader

- **Lignes de niveau** : on a décidé de dessiner des lignes de niveau à chaque fois qu'on monte de 1.1 sur le z, on a choisi cette valeur parce que ça semblait bien de représenter l'épaisseur (Différence en z) par à peu près 20 lignes de niveaux (22/1.1 = 20), les lignes de niveau sont dessinées avec un intervalle entre [0,0.2], ce qui donne des lignes pas trop épaisses mais bien visible.



- **Vallée** : pour la vallée qui est coloriée en vert on a considéré tous les points entre le point le plus bas (-202.09592) et -198.



Le code source pour la réalisation des lignes de niveaux et coloriage de la vallée est dans le fragment shader, qui traite la valeur du z renvoyé par le vertex shader pour tous les vertex du terrain

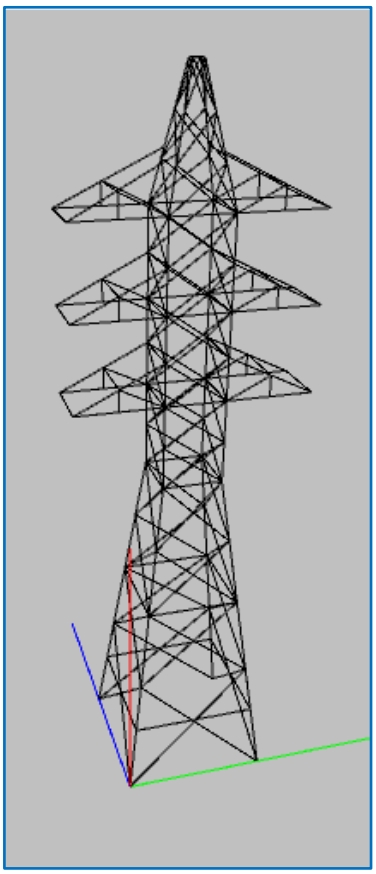
Partie 3 Pylônes

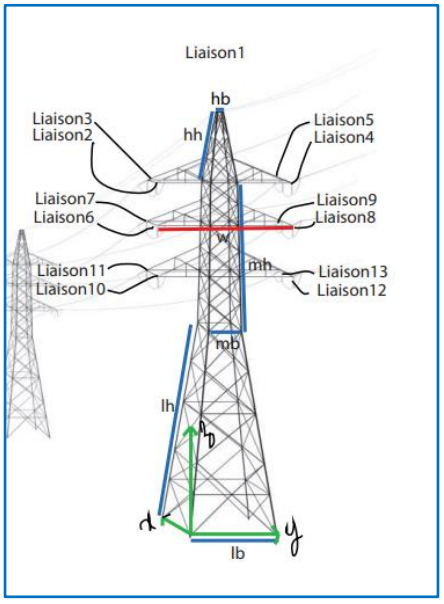
Création des pylônes :

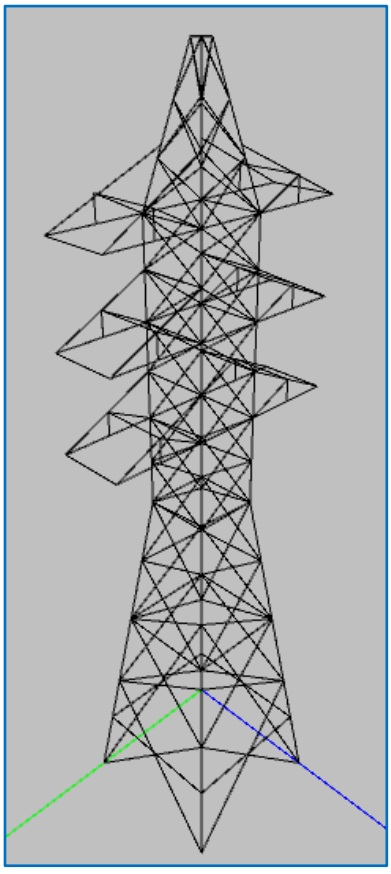
Les pylônes électriques sont inspirés par des vrais, on a utilisé l'outil adobe Illustrator pour prendre les bonnes mesures sur les dimensions des pylônes, dans la classe Pylône on trouve plusieurs attributs qui représentes la dimension des différentes parties qui caractérise le pylône, par exemple, l'attribut **lb (lower base)** représente la

```
private final float reduce=200;
private final float lb=192/reduce, mb=77/reduce, hb=13/reduce, mh=326/reduce, lh=451/reduce, hh=155/reduce, w=297/reduce;
private final float bratio=0.21;
private PShape shape;
private float xtemp, htemp, xtemp2, htemp2;
private PVector c1;
```

Longueur de la base du pylône, puisqu'il y en a plusieurs attributs caractérisent un pylône, on a pris les plus important, puis on a déduit les autres en utilisant les bases de la trigonométrie comme le théorème de Pythagore et celui de Thalès.







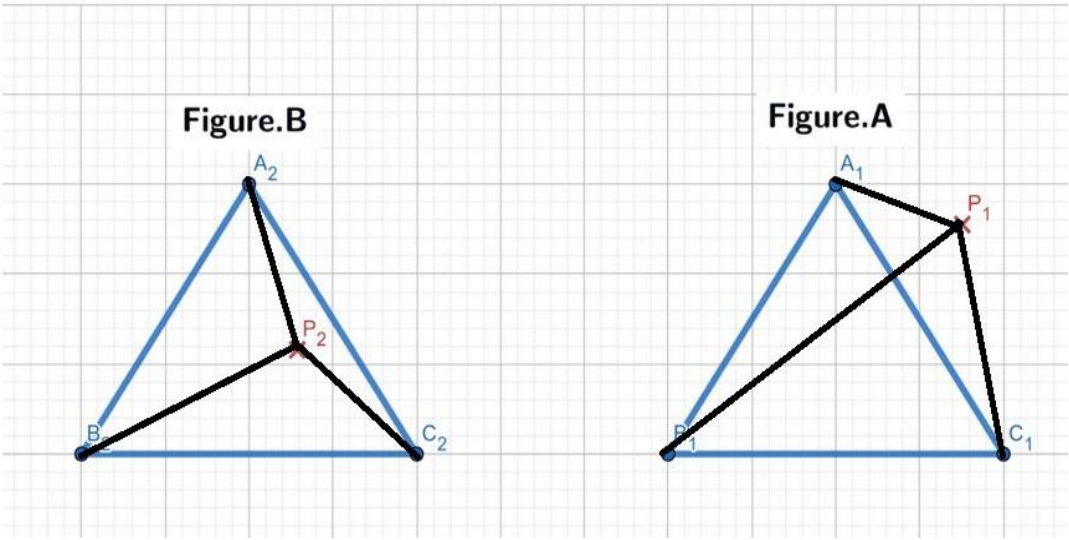
Les pylônes sont caractérisés par 11 points d'attache, chaque un est représenté par un PVector dans la classe Pylône, l'image en ci-dessus visualise les attributs et les liaisons.

Calcule de l'altitude à partir de (x, y) :

On a cherché à optimiser la recherche du point le plus proche de x,y qui sont données comme entré, au lieu de parcourir tous les vertex du terrain, notre recherche s'arrête dès que les coordonnées appropriées ont était trouvées.

Notre altitude est une altitude interpolée entre les sommets des triangles qui composent le terrain, en effet pour le hyperSimplify on a un nombre de child pour le terrain qui est de 9999, et en regardant le nombre de vertex pour chacun d'eux on trouve qu'il est égal à 3. Donc pour avoir notre altitude on par trois étapes représentées par des fonctions :

- **pointInChild (PVector p , PShape triangle) :** pour un point donné p et un triangle retourne true si ce point appartient au triangle et pour ça, on calcule les vecteurs normaux des plans PBC, PCA, PAB, (A, B, C étant les sommet du triangle), si le point appartient au triangle alors tous les vecteurs normaux sont dans la même direction, on le vérifie avec le produit scalaire.

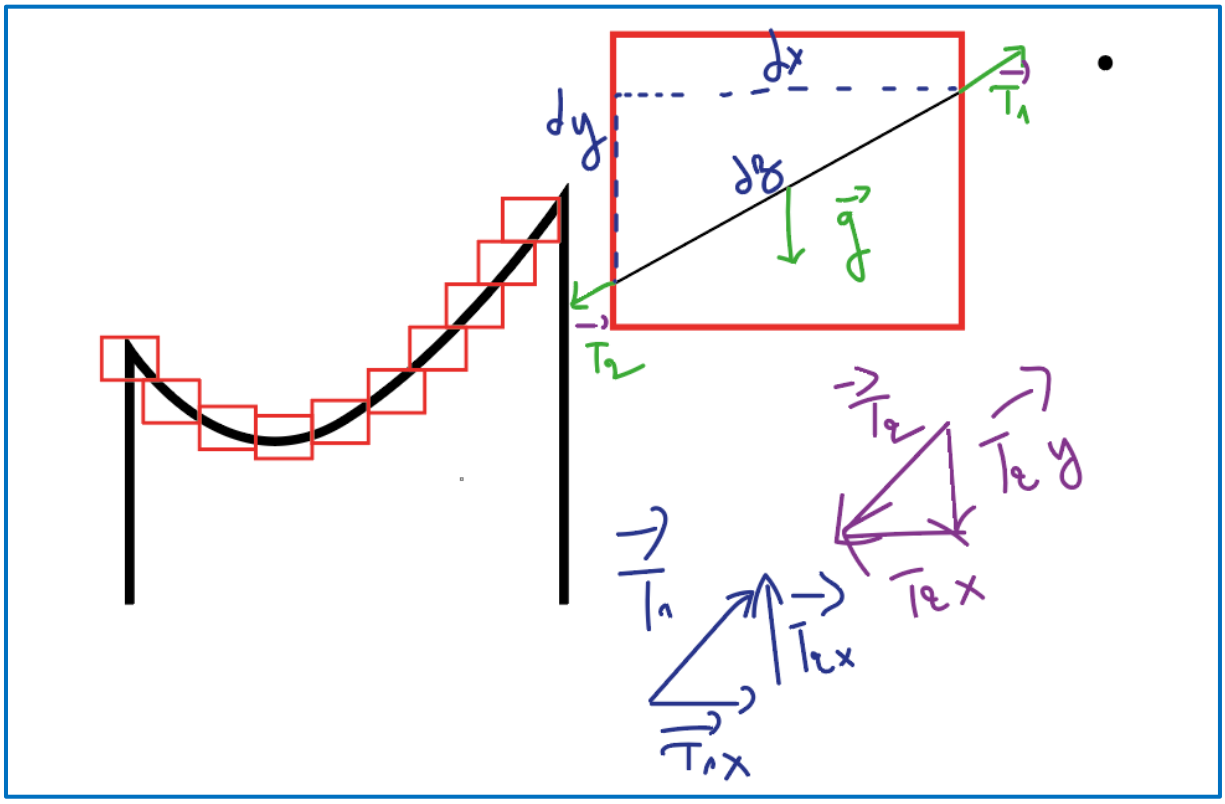


- **getTriangle (float x, float y)** : trouve à quel triangle appartient le point (x, y) dont on est en train de recherché l'altitude et le retourne, elle utilise la fonction précédente et parcoure tous les child du terrain (triangles) jusqu'à trouver le triangle et le retourne.
- **getZ (float x, float y) :** elle retourne la moyenne des z des sommets du triangle auquel appartient le point représenté par x et y, elle utilise les deux fonctions précédente.

Partie 4 : Lignes électriques

On a voulu simuler l'inclinaison réelle des lignes électriques en utilisant les bases de la physique, après des recherches approfondies, on a trouvé que c'est un problème standard de la mécanique : **catenary physics**, en fait, ce problème est basé sur Minimal Energy Principal, qui dit qu'un système fermé cherche toujours à atteindre le niveau minimal d'énergie, ce qui rend ce problème difficile est le fait que les hauteurs des deux pylônes peuvent être différents (une monté ou descente).

Pour commencer on découpe la file vers une infinité de sous parties, au point ou une découpe devient comme une ligne droite, comme le montre l'image en dessous :



Sur la droite, on a zoomé sur une partie, puis, on a représenté les forces qui agit sur un bout de file, donc on a :

$$\vec{g}: force\ de\ gravitation$$
$$\overrightarrow{T_1}: force\ de\ tire\ d'un\ pylone\ sur\ le\ file$$
$$\overrightarrow{T_2}: force\ de\ tire\ du\ file$$

On applique les projections sur les deux axes $\vec{x}$, $\vec{y}$, on obtient d'autres forces qui sont représenté sur l'image, $\overrightarrow{T_{x1}}, \overrightarrow{T_{x2}}, \overrightarrow{T_{y1}}, \overrightarrow{T_{y2}}, \overrightarrow{g_x} = \vec{0}, \overrightarrow{g_y}$
étant donné que le file est stable, donc son accélération est nulle alors par la 2 -ème loi du newton, on en déduit que :

$\overrightarrow{T_{x1}} + \overrightarrow{T_{x2}} = \vec{0}$ $et\ x2 = x1 + dx$ $(car\ on\ travail\ par\ rapport$ à $un\ plan\ x, y)$

$$\Rightarrow T_x(x1) - T_x(x1 + dx) = 0$$

$$\Rightarrow T_x(x1) = T_x(x1 + dx)$$

$$\Rightarrow T_x\ est\ constante$$

Par le théorème de Pythagore on a :

$$dz^2 = dx^2 + dy^2$$

$$\Rightarrow dz = dx.\sqrt{1 + \frac{dy^2}{dx}}$$

$$\Rightarrow dz = dx.\sqrt{1 + y'(x)^2}$$

$(le\ file\ devient\ une\ ligne\ droite, car\ on\ l'a\ découpé$ à $une\ infinité\ de\ sous\ parties)$

Et on a aussi :

$$\overrightarrow{T_{y1}(x)} + \overrightarrow{T_{y2}(x)} + \overrightarrow{g(x)} = \vec{0}$$

$$dans\ la\ suite\ on\ remplace\ y1\ par\ y\ pour\ faciliter\ les\ lecture.$$

$$\Rightarrow T_y(x) - T_y(x + dx) = g.dm,$$

$$dm: la\ masse\ de\ la\ sous\ partie\ droite\ du\ file$$

On suppose que le file est uniforme, en d'autres termes, il a une distribution uniforme de la masse sur sa longueur, donc :

$$dm = dz.\frac{M}{L}\ ou\ M\ est\ la\ masse\ du\ file, L\ est\ sa\ longueur\ et\ dz\ est\ celle$$

$$qu'on\ a\ calculé\ précédement, qui\ représente\ la\ taille\ de\ la\ sous\ partie\ du\ file.$$

On remplace dm dans l'équation précédente par celui qu'on vient de calculer :

$$T_y(x) - T_y(x + dx) = g.\frac{M}{L}.dx.\sqrt{1 + y'(x)^2}$$

On divise les deux coté de l'équation par dx :

$$\frac{T_y(x) - T_y(x + dx)}{dx} = g.\frac{M}{L}.\sqrt{1 + y'(x)^2}$$

$$\Rightarrow T_y'(x) = g.\frac{M}{L}.\sqrt{1 + y'(x)^2}$$

Maintenant on doit éliminer la force T dans l'équation, on sait que le vecteur $\vec{T}$ point dans la même direction que le file $\overrightarrow{(dx, dy)}$, donc on en déduit que :

$$\frac{T_y}{T_x} = \frac{dy}{dx}$$

$$\Rightarrow \frac{T_y}{T_x} = y'(x)$$

$$Or\ T_x\ est\ une\ constante\ donc$$

$$T_y = C.y'(x)$$
$$\Rightarrow T_y'(x) = C.y''(x)$$

En passant par les équations déduit précédemment, on obtient une équation différentielle qui est la suivante :

$$y''(x) = g.\frac{M}{L.C}.\sqrt{1+y'(x)^2}$$

On remplace $y''(x)$ par $u(x)$ pour simplifier les calculs, donc on obtient

$$\frac{du(x)}{dx} = g.\frac{M}{L.C}.\sqrt{1+u(x)^2}$$

On integre les deux cotés de l'équation, mais avant on multiplie par $\frac{dx}{\sqrt{1+u(x)^2}}$

$$\int \frac{du(x)}{\sqrt{1+u(x)^2}} = \int g.\frac{M}{L.C}.dx\,,$$

ou l représente la distance entre les deux pylons.

Le premier intégral est un intégral connu, la primitive est la fonction $sh(x)$, on peut arriver au résultat en appliquant le changement de variable suivant :

$$u(x) = i\,.\sin(x)$$

Par ça, on obtient un cos dans le dénominateur, et on simplifie :

$$\int \frac{i.\cos(x)\,.dx}{\sqrt{1-\sin(x)^2}} = g.\frac{M}{L.C}.x + A'$$
$$\Rightarrow \int \frac{i.\cos(x)\,.dx}{|\cos(x)|} = g.\frac{M}{L.C}.x + A'$$

étant donné que x est au voisinage de 0, parce que on travaille sur les sous parties des pylônes donc on obtient :

$$\Rightarrow \int i.dx = g.\frac{M}{L.C}.x + A'$$
$$\Rightarrow i.x = g.\frac{M}{L.C}.x + A$$
$$\Rightarrow \sin(x) = \sin\left(\frac{g.\frac{M}{L.C}.x + A}{i}\right)$$
$$\Rightarrow \mathrm{u(x)} = i.\sin\left(\frac{g.\frac{M}{L.C}.x + A}{i}\right)$$

Mais on sait que $\sin(x) = \frac{e^{i.x} - e^{-i.x}}{2.i}$ donc en remplaçant dans l'expression de u(x) on obtient :

$$\mathrm{u(x)} = \frac{e^{g.\frac{M}{L.C}.x+A} - e^{-g.\frac{M}{L.C}.x-A}}{2}$$

Cette expression en fait, ce n'est rien que :

$$\mathrm{u(x)} = \sinh\left(g.\frac{M}{L.C}.x + A\right)$$

Pour déduire la valeur de y, il suffit d'intégrer u(x) car y'(x)=u(x) donc :

$$\mathrm{y(x)} = \frac{1}{g.\frac{M}{L.C}}\cosh\left(g.\frac{M}{L.C}.x + A\right) + B$$

On remarque qu'avec l'intégration, on a rajouté 3 variables à l'expression de y, qui sont A, B et C, donc il nous manque 3 équations pour déduire es valeurs de ces constantes :

On sait que le file passe par deux points qui sont $\left(\frac{-l}{2}, h1\right), (\frac{l}{2}, h1)$ et on sait que la longueur de la file est L donc :

$$y\left(\frac{-l}{2}\right) = h1;\ y\left(\frac{l}{2}\right) = h2;\ \int_{\frac{-l}{2}}^{\frac{l}{2}} dz = L$$

$(dz\ est\ la\ longueur\ du\ petit\ bout\ du\ file, donc\ pour\ trouver\ la\ longueur\ totale, on\ doit\ passer\ par\ l'intégrale).$

$$\int_{\frac{-l}{2}}^{\frac{l}{2}} dz = L \Rightarrow \int_{\frac{-l}{2}}^{\frac{l}{2}} dx.\sqrt{1 + y'(x)^2} = L\ (on\ a\ déja\ résolu\ cette\ intégrale),$$

$ça\ nous\ donne$

$$\frac{1}{g.\frac{M}{L.C}}.\sinh\left(g.\frac{M}{L.C}.x + A\right)\Big|_{\frac{-l}{2}}^{\frac{l}{2}} = L$$

On obtient 3 contraintes à la fin :

$$\frac{1}{g.\frac{M}{L.C}}.\sinh\left(g.\frac{M}{L.C}.x + A\right)\Big|_{\frac{-l}{2}}^{\frac{l}{2}} = L\ équivalente\ à\ A = \mathrm{atanh}\left(\frac{h2 - h1}{L}\right)$$

$$\frac{1}{g.\frac{M}{L.C}}\cosh\left(-g.\frac{M.l}{2.L.C}.+A\right) + B = h1$$

$$\frac{1}{g.\frac{M}{L.C}}\cosh\left(g.\frac{M.l}{2.L.C}.+A\right) + B = h2$$

Pour A c'est facile de le trouver, mais pour les valeurs de B et C, c'est un peu délicat, car on n'a pas une manière directe pour résoudre l'équation donc on est obligé de trouver une méthode analytique pour les résoudre.

Recherche des valeurs de B et C par une méthode analytique :

Pour la résolution analytique on a utilisé la méthode de newton pour résoudre ces équations, donc on a l'a rendu à un problème de minimisation, la méthode de newton consiste à trouver les racines d'une fonction d'une façon très optimale,

```
// Function to find the root
double newtonRaphson(double x)
{

    double h = funcAlpha(x) / derivFuncAlpha(x);
    while (Math.abs(h) >= eps)
    {
        h = funcAlpha(x)/derivFuncAlpha(x);

        // x(i+1) = x(i) - f(x) / f'(x)
        x = x - h;
    }

    return x;
}
```

Dans la démonstration précédente on est passé par un point ou on a déduit l'équation suivante :

$$\frac{1}{g.\frac{M}{L.C}}.\sinh\left(g.\frac{M}{L.C}.x + A\right)\Big|_{\frac{-l}{2}}^{\frac{l}{2}} = L$$

On peut la simplifier plus, en appliquant les règles de simplification cosh et sinh à :

$$\frac{2}{g.\frac{M}{L.C}}.\sinh\left(g.\frac{M.l}{L.C}\right).\cosh\left(\mathrm{A}\right) = L$$

Et c'est cette dernière qu'on a utilisé dans l'algorithme Newton Raphson pour trouver la valeur du C, puis on remplace dans une des équations précédentes pour trouver la valeur de B.

Placement des pylones :

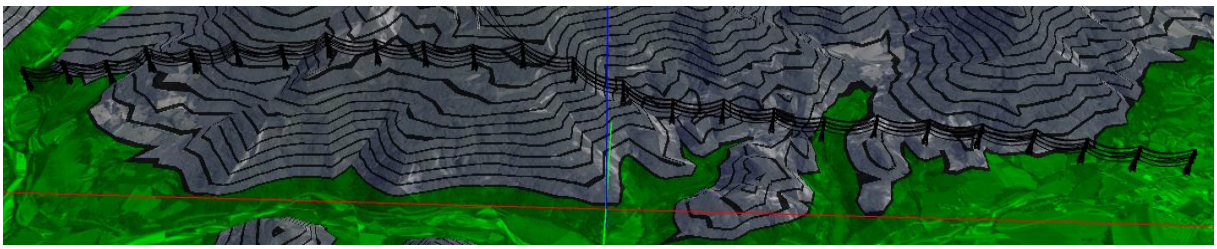
Les pylônes étant voulu entre les points 20,100 et 40.-115, pour les aligner on calcule l'équation de la droite qui relie ces deux point là on trouve qu'elle est de y = -10.75x + 315, et étant donné qu'on a 25 pylônes à placer sur cette lignes on va faire notre décalage uniforme sur les valeurs de x (pour aller de 20 à 40 en 25 pas on avance de 40-20/24 à chaque fois).

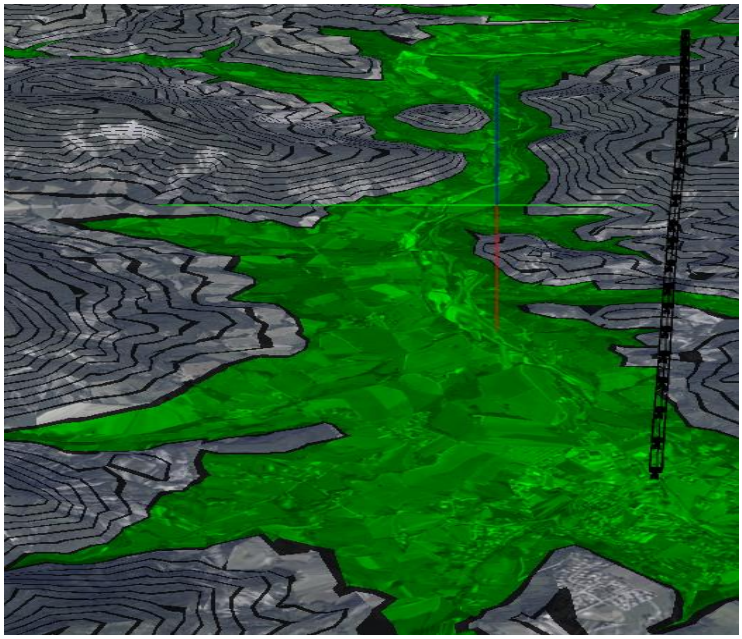
Avec la fonction getZ(x,y) on récupère l'altitude pour un x et y donné sur la droite et on fait la translation nécessaire et aussi la rotation en suivant le coefficient directeur -10.75.

Attache des lignes électriques :

Comme montrer dans schéma de construction de l'éolienne chaque point d'attache d'un pylône P1 et destiné à être relié à un point précis dans le pylône P2 par exemple liaison 4 avec 5, 2 avec 3 etc., ces informations sont récupérées dans une HashMap, les transformation après étant faite pour placé les lignes et les orienter selon le coefficient -10.75 de la droite des pylônes.

Pour afficher/cacher les pylônes on appui sur p, pour les lignes c'est l.

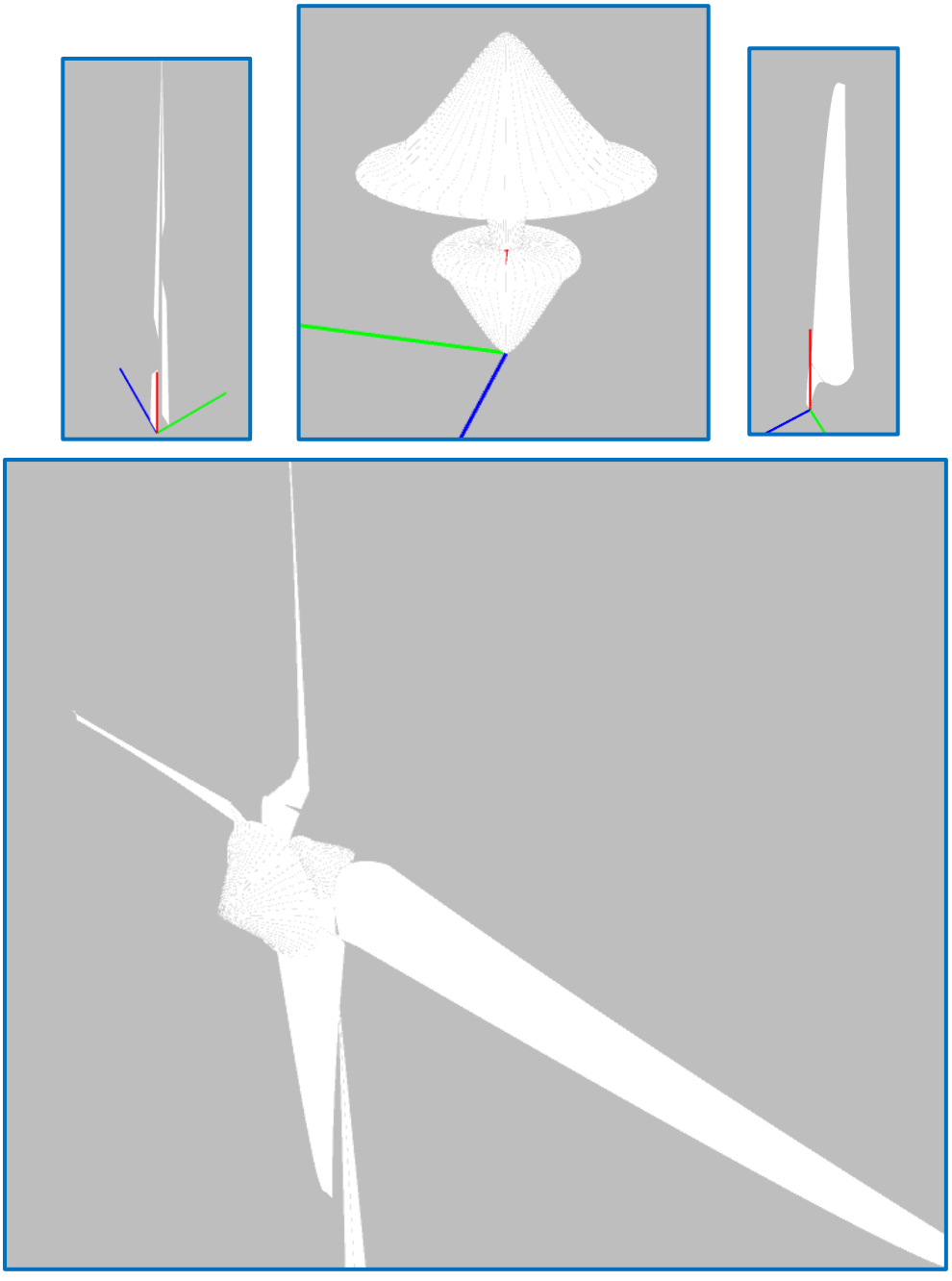




Partie 5 : Eolienne

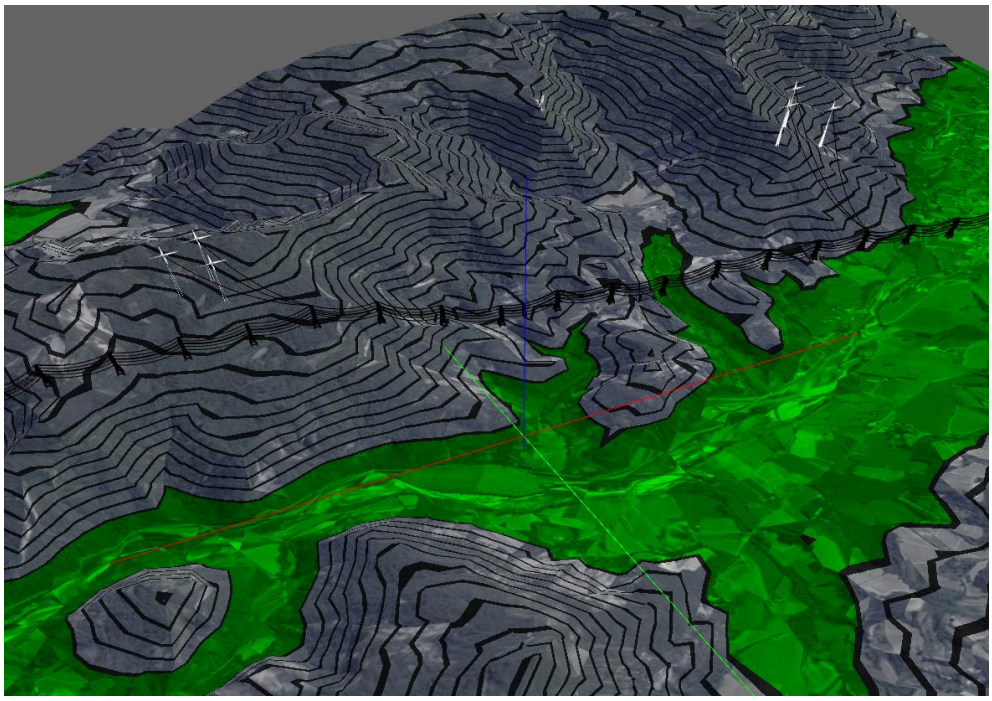
On a décomposé l'éolienne vers trois sous parties, chaque partie est créé en utilisant les courbes de Bézier, la première partie est le Holder qui est la partie bas de l'éolienne,

La deuxième partie est Head, qui est le moteur de l'éolienne et troisième partie est les ailes, chaque partie est traité individuellement, en utilisant le même principe que celui des pylônes.



Placement des éoliennes :

Les éoliennes sont placées sur les collines sur le côté X positif des pylônes pour pouvoir être liée au pylônes électrique qui vont transmettre l'énergie au village dans la vallée.



université
PARIS-SACLAY

Conclusion

Ce projet nous a permet d'apprendre pleine de points essentiels et d'appliquer les notions théoriques qu'on a appris comme la méthode d'Euler et les fonctions hyperboliques, aussi, il nous a permis de découvrir d'autres logiciels comme adobe Illustrator qu'on juge très importants pour notre carrière comme informaticiens.