28/09

* Création d’un cube : possibilité de la déplacer et de changer sa taille en indiquant des valeurs dans une textbox
* Création d’un cube avec un gradient de couleur sur une face
* Création d’un menu

05/10

* Création d’un cube avec possibilité de le bouger en cliquant sur des flèches
* Créer des objets de base à partir du menu
* Changer l’angle de la caméra en cliquant sur un bouton à partir du menu
* Continuer de travailler sur le gradient de couleur : ajouter un gradient à différents sommets du cube
* Se documenter sur GMSH (calcul du maillage)

bug à voir :

* création d’un cube superposé à un autre cube
* les buttons décalent l’endroit du clic

Application 14/10:

L’application est basée sur three.js. Elle permet de créer des objets 3D et de les modifier de différentes manières dans différents modes accessibles via différentes touches du clavier:

* m : mouvements
* c : contraintes
* s : taille
* r : rotation

Le mode mouvement permet de créer des objets (cube, cylindre, sphère) et de les déplacer. On sélectionne une forme en cliquant dessus, on déplace en utilisant les flèches qui sont apparues sur la forme sélectionnée.

Dans le mode contrainte, une petite sphère apparaît sur chaque sommet. Cliquer sur la sphère permet de sélectionner le sommet puis de lui appliquer une contrainte en utilisant le menu (actuellement une couleur).

ToDo :

* Chaque face est sélectionnable, permettant de sélectionner tous les sommets de la face.
* Maintenir la touche Ctrl permet de sélectionner plusieurs sommets ou faces.

Le mode taille permet de changer la taille de la forme sélectionnée dans le menu.

ToDo :

* La taille peut être changer graphiquement.

Le mode rotation permet de faire pivoter la forme sélectionnée dans le menu.

ToDo :

* La rotation peut être faite graphiquement

Chaque mode possède une fonction start et une fonction stop qui s’exécutent respectivement quand on active le mode et quand on le quitte.

Le menu utilise la bibliothèque dat.gui. Il permet de :

* changer la position de la camera
* créer un objet du type souhaité (cube, cylindre, sphere)
* sélectionner un objet
* changer la taille
* changer la position
* pivoter un objet
* changer la couleur
* appliquer une contrainte (actuellement une couleur) au sommet sélectionné

ToDo :

* La forme du menu doit être dépendante du mode
* supprimer un objet

A l’aide du logiciel GMSH, nous réalisons des maillages sur nos figures 3D. Pour cela nous récupérons notre figure au format .stl. Cependant, pour réaliser un maillage avec GMSH, nous avons besoin de traduire ce .stl en un fichier .geo. Pour cela nous avons crée un script python.

ToDo :

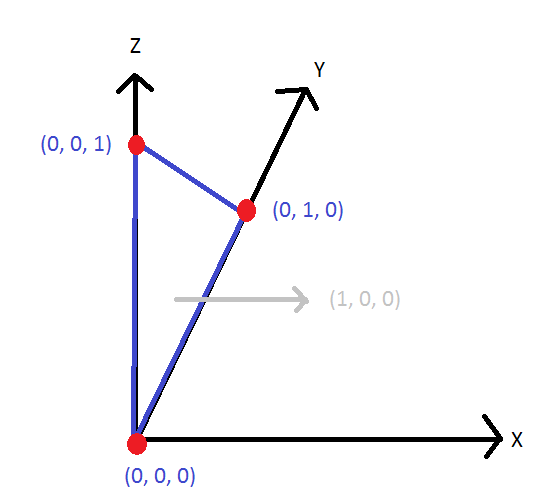
* utiliser la library gmsh (en c++) pour automatiser le maillage sur l’application
* créer un serveur à l’aide de flask (library python) afin d’envoyer le maillage crée par gmsh à l’application web

**Convertion d’un .stl en .geo**

Les fichiers stl et geo correspondent à des objets 3D.

I] Le format STL

Le format STL décrit un objet 3D uniquement par sa surface externe. Il s’agit d’une surface fermée et définie par une série de triangles. On défini ces triangles par leur coordonnées cartésiennes (x, y, z) et par un vecteur normal orienté vers l’extérieur de la forme.



Exemple de code stl :

*solid name*

*facet normal 1 0 0* Le vecteur normal du triangle

*outer loop*

*vertex 0 0 1* Les sommets (vertices) du triangle

*vertex 0 1 0*

*vertex 0 0 0*

*end loop*

*end facet*

*...*

*...*

*…*

*endsolid name*

II] Le format GEO

Contrairement au format STL, le format geo ne décrit pas seulement une surface mais aussi un volume. De plus, l’objet 3D n’est pas défini par des triangles mais par ses faces (surfaces fermées).

Le format geo est structuré de la manière suivante :

* on commence par définir tous les sommets de notre objet numérotés à partir de 1
* on défini ensuite toutes les arêtes numérotés à partir de 1 (on “relie deux sommets”)
* on défini les faces de notre objet (surfaces fermées et le numéro des lignes de chaque faces doivent suivre un cycle)
* pour finir, on ajoute un volume à l’objet

Exemple de code geo :

*Point(1) = {0, 0, 1, 0.3};* On défini les sommets de notre objet. Le 0.3 est une longueur

*Point(2) = {0, 1, 0, 0.3};* caractéristique (valeur du pas localement autour du point)

*Point(3) = {0, 0, 0, 0.3};*

*Line(1) = {1, 2};* On défini les arêtes de notre objet en reliant les sommets

*Line(2) = {2, 3};*

*Line(3) = {3, 1}*

*Line Loop(4) = {1, 2, 3};* On défini les faces de notre objet en spécifiant les côtés de

*Plane Surface(5) = {4};* chaque face

*//Surface Loop(6) = {5};* (Non utile dans cet exemple car il s’agit d’un surface et non

d’un objet 3D) On défini la figure finale par ses faces

*//Volume(7) = {6};* (Non utile dans cet exemple pour la même raison) On donne

un volume à notre objet

III] Problèmes rencontrés lors de la conversion

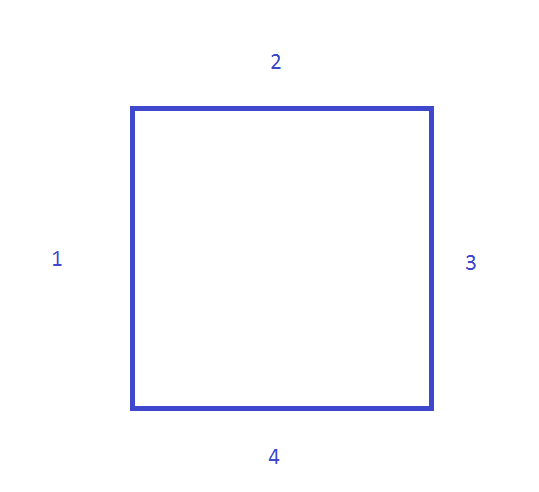
Nous avons décidé de créer un script python pour réaliser la conversion stl -> geo. Pour celà nous lisons le fichier stl pour le retranscrire dans un nouveau fichier geo.

Cependant, nous avons rencontrés plusieurs problèmes :

* Tout d’abord, les objets stl sont défini par des triangles, alors que les objets geo sont défini par des faces. Si on prend l’exemple du cube, une face du cube en stl sera formé par deux triangles, alors qu’une face du cube en geo sera un simple carré. En stl, ce cube aura donc des diagonales qu’il faudra supprimer lors de la conversion en geo (car on doit définir des surfaces fermées).

Pour cela, on s’aide du vecteur normal : si plusieurs triangles ont le même vecteur normal, c’est qu’ils forment une seule et même face. Et si ces triangles partagent une ligne en commun, c’est que cette ligne est à l’intérieur de la face et donc qu’elle doit être supprimée (on doit définir l’extérieur de la face).

* Ensuite, lorsqu’on défini les faces de notre objet en geo, il faut que les lignes qui composent une face forment un cycle.

exemple : 

On peut définir cette face de la manière suivante : {1, 2, 3, 4} car ces lignes se suivent (elles forment un cycle).

En revanche, on ne peut pas définir cette face de la manière suivante : {1, 3, 4, 2}.

* **(EN COURS)** Lorsqu’on défini les faces, il faut parfois ajouter des “-” avant le numéro de la ligne. Avec un “-” on parcourt la ligne dans le sens indirect, alors que sans on la parcourt dans le sens direct.
* **(EN COURS)** Trouver un moyen de définir la longueur caractéristique en fonction de la position du point par rapport à l’origine.