# Travail pratique #2

Graphiques en 3D avec Blender, OpenGL 1.1, et glsim

INF5071 - Infographie

Automne 2020

## Table des matières



## Exercice 1 : La planète X

Depuis longtemps, les astronomes suspectent qu'une neuvième planète se trouve dans notre système solaire au-delà l'orbite de Pluton [1]. L'effet gravitationnel de ce corps céleste, nommé **planète X**, semble être la cause des perturbations orbitales des objets dans la ceinture de Kuiper et de celle de la planète naine Sedna. Les calculs des astrophysiciens attribuent à la planète X une masse entre 5 à 10 fois celle de la Terre, et une orbite elliptique autour du Soleil dont la durée serait entre 10000 et 20000 ans. En attendant que les astronomes découvrent si cette 9e planète existe vraiment et qu'ils l'observent avec un télescope, vous décidez de créer une vue d'artiste 3D de ce corps céleste. **L'objectif du TP2 est donc de créer une représentation fictive de la planète X à l'aide des notions d'infographie 3D vues jusqu'à maintenant dans le cours.**

1. https://www.futura-sciences.com/sciences/actualites/astronomie-mysterieuse-planete-9-existe-t-elle-vraiment-61311/

**Note** : Vous devez utiliser l'implémentation JavaScript d'OpenGL 1.1 pour ce travail pratique (`glsim`). Le gabarit fourni avec cet énoncé importe déjà le script `glsim` depuis le site web du manuel du cours. Je vous recommande aussi d'utiliser l'IDE WebStorm pour exécuter le code dans un serveur local virtuel.

## a) Géométrie d'une sphère (2 pts)

La première étape est la modélisation de la géométrie d'une sphère pour représenter la planète en 3D. Pour cet exercice, vous devez compléter la fonction `uvSphere(rayon, nLongitude, nLatitude)` du fichier `tp2.js`.

- La fonction doit recevoir les paramètres d'entrée suivants.
  - `rayon` : Un nombre réel indiquant le rayon de la sphère.
  - `nLongitude` : Le nombre de côtés du maillage polygonal dans la direction des longitudes (*est-ouest*)
  - `nLatitude` : Le nombre de côtés du maillage polygonal dans la direction des latitudes (*nord-sud*)
- La fonction doit retourner un objet contenant les attributs suivants
  - `vertexPositions` : Un tableau (*array*) contenant les positions 3D des sommets pour tous les sommets à la surface de la sphère
  - `vertexNormals` : Un tableau (*array*) contenant les composantes de chaque vecteur normal 3D associé aux sommets à la surface de la sphère
  - `vertexTextureCoords` : Un tableau (*array*) contenant les coordonnées de texture 2D associées aux sommets à la surface de la sphère
  - `indices` : Un tableau (*array*) contenant la liste des indices des sommets pour chaque face formant le maillage polygonal de la sphère.
- Cette structure de données est de type *indexed face arrays*.
- Voici l'équation pour la surface d'une sphère.

$$x = r\cos(\theta)\cos(\phi) \tag{1}$$
$$y = r\sin(\theta)\cos(\phi) \tag{2}$$
$$z = r\sin(\phi) \tag{3}$$

où $\theta \in [0, 2\pi]$ est l'angle associé aux longitudes (*Est-Ouest*), $\phi \in [-\pi/2, \pi/2]$ est l'angle associé aux latitudes (*Sud-Nord*) et $r$ est le rayon de la sphère.

- **Rappel** : Les vecteurs normaux doivent pointer vers l'extérieur d'un maillage polygonal fermé.
- **Rappel** : La convention est qu'un vecteur normal pointe dans la même direction que le vecteur obtenu par la règle de la main droite lorsque les doigts suivent les sommets de la face avant dans le sens antihoraire. Le pouce pointe alors dans la direction du vecteur normal à la surface.
- **Précisions** : Pour faire correspondre l'image de texture 2D et la surface 3D de la sphère, nous utiliserons une projection équirectangulaire. Pour cette méthode, l'axe horizontal $s$

de la texture est parallèle aux latitudes, et l'axe vertical $t$ de la texture est parallèle aux longitudes.

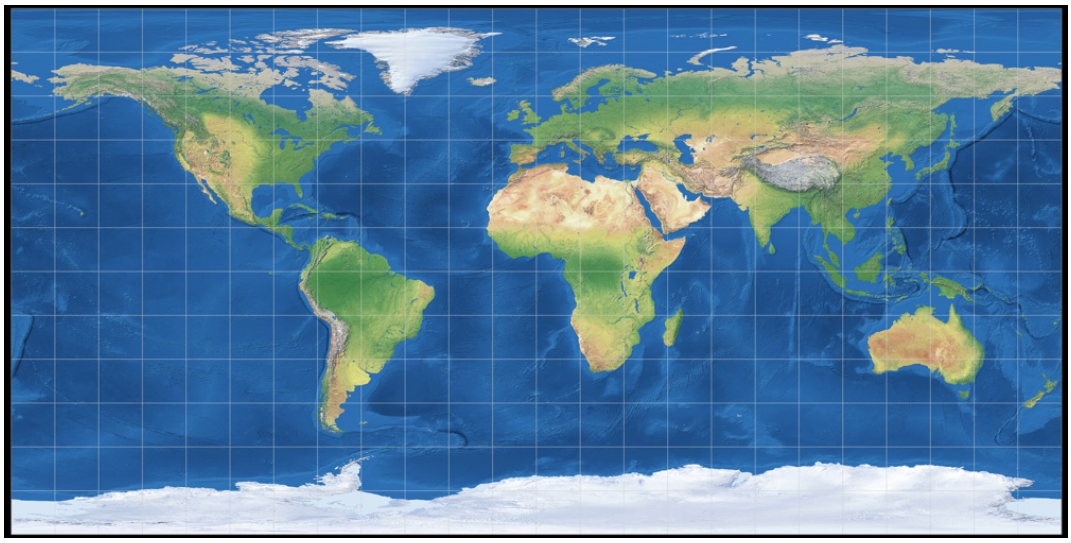

FIGURE 1 – Projection équirectangulaire (Source : map-projections.net)

## b) Afficher une sphère avec la méthode IFS (2 pts)

Modifiez la fonction `draw_planet` pour dessiner la planète. Vous devez définir un matériau blanc uni ayant une très faible composante spéculaire et une faible brillance. Vous devez aussi utiliser `glDrawElements` et les fonctions associées pour dire à OpenGL où se trouvent les données de sommets, de normales, et de coordonnées de texture.

## c) Créer une texture géographique (2 pts)

Utilisez le logiciel de dessin Krita pour dessiner une image de texture de taille 512x256 représentant une vue satellite de la planète. Assurez-vous d'utiliser le mode enveloppant (comme au laboratoire 1) pour que la texture se répète horizontalement. Exportez votre image sous le nom `tp2_texture_planete.jpg` et enregistrez l'image dans le même dossier que les autres fichiers du TP.

Ensuite, modifiez la fonction `init()` pour importer l'image de texture et l'envoyer au GPU. Utilisez une méthode d'interpolation linéaire pour le filtre de minification.

**Note** : Pour des raisons de sécurité, les images de texture doivent être dans le même domaine que le script OpenGL. Une façon d'expérimenter avec des textures locales est d'utiliser un serveur local (par exemple avec WebStorm).

## d) Lumière et environnement (2 pts)

Pour cette partie, nous allons créer l'environnement et les sources de lumière pour notre scène 3D. - Modifiez la couleur d'arrière-fond pour du noir. - Ajoutez une lumière directionnelle qui est fixe par

rapport aux coordonnées du monde. Choisissez une lumière blanche, ayant la même composante spéculaire et diffuse, et ayant une faible composante ambiante. Faites en sorte que cette lumière pointe dans la direction de X - Modifiez la fonction `generate_randomStars()` pour générer une liste de positions 3D (x,y,z), de taille et de couleur aléatoires. Assurez-vous que les étoiles sont au moins à une distance de 0.75 unité du centre de la planète. Générez des tailles d'étoiles variant de 1 à 4 unités. - Modifiez la fonction `draw_stars()` pour dessiner les étoiles à l'aide de primitives `GL_POINTS`. Modifiez la taille des étoiles avec la fonction `glPointSize`, et utilisez la couleur de l'étoile pour modifier sa couleur d'émission.

## e) Animation de la texture (2 pts)

— Pour simuler la rotation de la planète, on utilisera une transformation de texture. Modifiez la fonction `update()` pour simuler la rotation de la planète par une transformation de sa texture. Quelle transformation devez-vous effectuer ?

## f) Modeliser un satellite (2 pts)

Vous décidez d'ajouter un satellite artificiel dans votre scène 3D. Modélisez un satellite 3D à l'aide des primitives suivantes. - Cube : Corps du satellite - Cylindres : Joints reliant le corps aux panneaux solaires - Cubes : Panneaux solaires - Sphère modifiée : Coupole - Cylindre modifié : Antenne de la coupole.

Donnez des noms à chacun des objets de votre satellite. Ces noms seront réutilisés lors de l'importation de l'objet dans la scène 3D OpenGL. Utilisez également un *smooth shading* pour la coupole de l'antenne et les joints cylindriques.

— Enregistrez ce fichier blender sous le nom : `tp2_satellite.blend`
— Générez une image de cette scène où on peut bien voir le satellite et enregistrez cette image sous le nom : `tp2_satelitte.png`
— **Exportation** : Assurez-vous que toutes les faces sont des triangles (URL), puis exportez le satellite sous format `.obj`. (nom de fichier : `tp2_satellite.obj`)

## g) Ajouter le satellite dans la scène 3D (2 pts)

— Dans l'initialisation de la scène 3D, utilisez la fonction `loadOBJFile` fournie avec ce TP pour importer le satellite sous format IFS. (La fonction `loadOBJFile` est définie dans le fichier `tp2_helper_functions.js` et elle est importée au début du fichier HTML)
— Modifiez ensuite la fonction `draw_satellite` pour dessiner le satellite dans la scène 3D. Ajustez la taille et la position du modèle à l'aide de transformations. Utilisez la fonction `glDrawElements` avec la primitive `GL_TRIANGLES`.
— Utilisez un matériau gris métallique, sauf pour les panneaux solaires qui devraient être bleus et brillants avec un reflet spéculaire blanc.

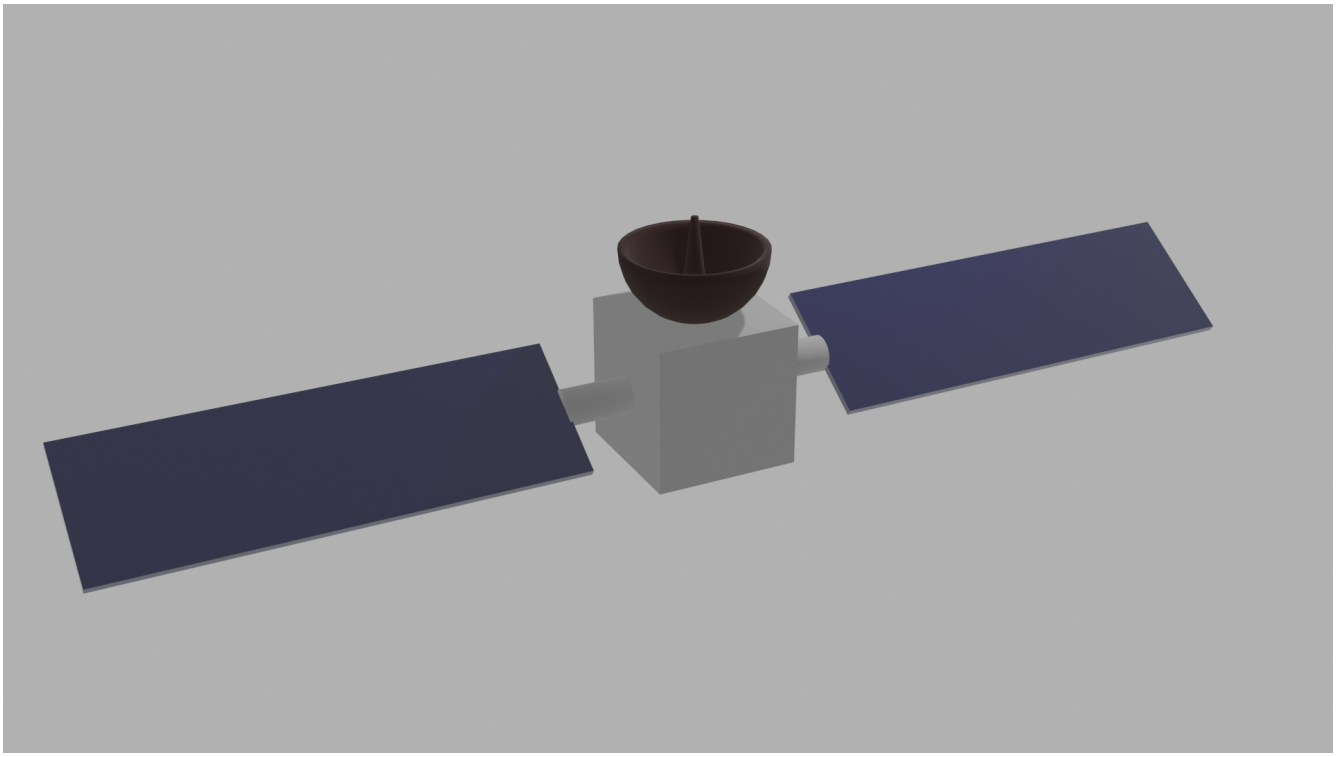

FIGURE 2 – Exemple d'un satellite à modéliser avec Blender

### h) Animation du satellite (1 pt)

Pour cette dernière partie, modifiez la fonction `update()` pour ajouter une animation de rotation au satellite. Utilisez les angles de rotation `satAngleX`, `satAngleY`, `satAngleZ`, et les vitesses de rotation `satSpeedX`, `satSpeedY` et `satSpeedZ` défini en en-tête du fichier.

### Résultat attendu

## Références

- WebStorm
- Documentation GLSIM
- Documentation OpenGL 1.1

## Instructions générales

- Le travail pratique doit être réalisé en équipe de 2 maximum.
- Vous devez impérativement être membre d'une équipe pour avoir accès à la remise du TP.
- **Si vous désirez changer de groupe pour ce TP, ou si vous êtes seul.e et vous désirez travailler avec quelqu'un d'autre pour ce travail, écrivez-moi. Je vais mettre en correspondance les gens seuls et désirant se trouver un.e partenaire.**

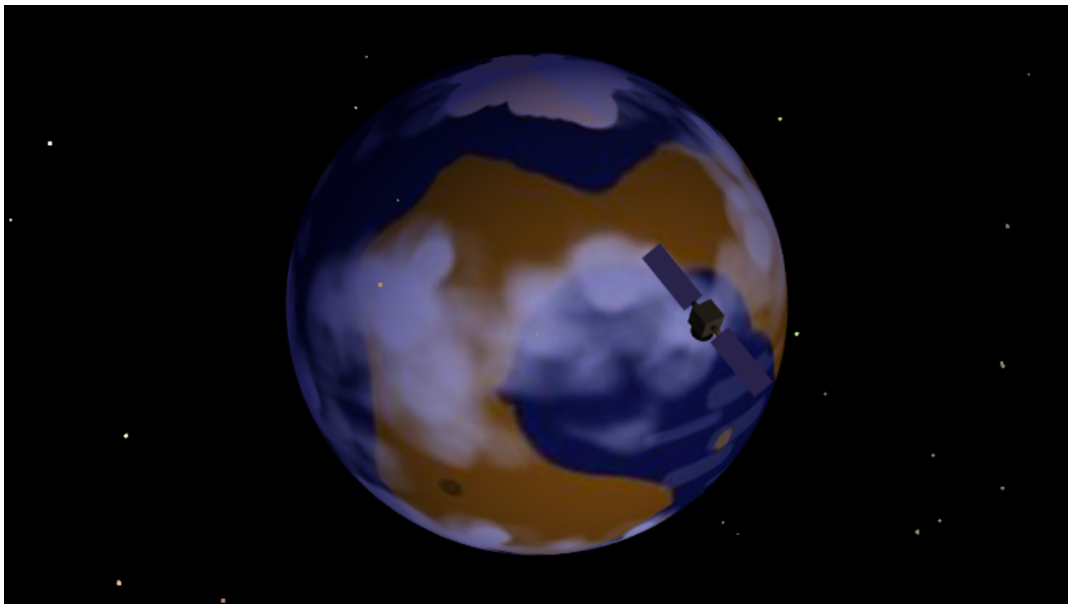

FIGURE 3 – Résultat attendu. Un enregistrement vidéo du résultat attendu est disponible dans le fichier zip associé au TP et sur Panopto

- Vous devez soumettre un seul fichier `zip` qui contient tous les fichiers nécessaires pour tester votre travail.
- Votre fichier zip doit être nommé comme suit :

`tp1_<code_MS_Etudiant1>_<code_MS_Etudiant2>.zip`

- Le fichier compressé doit contenir les fichiers suivants
  - `tp2.html`
  - `tp2.js`
  - `tp2_helper_functions.js`
  - `tp2_satellite.blend`
  - `tp2_satellite.obj`
  - `tp2_satellite.png`
  - `tp2_texture_planete.kra`
  - `tp2_texture_planete.jpg`
  - `notes.md` (optionnel)
- Veuillez utiliser le fichier `notes.md` pour écrire tous les autres commentaires, démarches, ou réponses aux questions qui ne peuvent pas être incluses dans les autres fichiers
- Veuillez utiliser les gabarits fournis avec le TP pour commencer les exercices.
- Vous devez soumettre votre travail via Moodle obligatoirement. Les soumissions par courriel ne seront pas corrigées.
- Le plagiat ne sera pas toléré, écrivez votre propre code. Les normes de plagiat de l'université seront appliquées en cas de plagiat (voir la Politique 18).
- Il est important qu'il n'y ait pas d'erreurs d'exécution pour la correction.
- Une partie des points pour chaque question est attribuée à la lisibilité du code. Mettre des commentaires pour expliquer votre démarche.

— La qualité artistique n'est pas un objectif du cours. Ne passez donc pas trop de temps sur l'esthétique de vos graphiques, puisque cela ne sera pas évalué.
— **Note :** ce travail compte pour 15% de la note finale.

## Pondération

| Élément | Pondération |
|---|---|
| a) Géométrie | 2 pts |
| b) Affichage de la planète | 2 pts |
| c) Texture | 2pts |
| d) Lumière et environnement | 2 pts |
| e) Animation de la texture | 2 pts |
| f) Modélisation satellite | 2 pts |
| g) Affichage du satellite | 2 pts |
| h) Animation du satellite | 1 pt |
| **Total** | 15 points |