Travail pratique #2

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

INF5071 - Infographie

Automne 2022

Table des matières

Exercice 1 : Télescope James-Webb

Après 30 ans d'opération, le télescope spatial Hubble a été remplacé le 25 décembre 2021 par le télescope spatial James-Webb. Ce nouveau satellite, développé par la NASA, l'ESA et l'agence spatiale canadienne, est en orbite autour du point de Lagrange L2 du système Terre-Soleil. Les points de Lagrange sont des positions dans l'espace où un petit corps céleste de masse négligeable (ex.: astéroïde) et soumis à l'attraction de deux plus gros corps célestes (ex.: la Terre et le Soleil) est en équilibre. Il y a 5 points de Lagrange pour le système Terre-Soleil, et leur position est fixe par rapport à l'orbite de la Terre. **L'objectif du TP2 est de créer une représentation schématique de l'orbite du télescope James-Webb à l'aide des notions d'infographie 3D vues jusqu'à maintenant dans le cours.**

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 ou Visual Studio Code avec l'extension Live Server (URL) pour exécuter le code dans un serveur local virtuel.

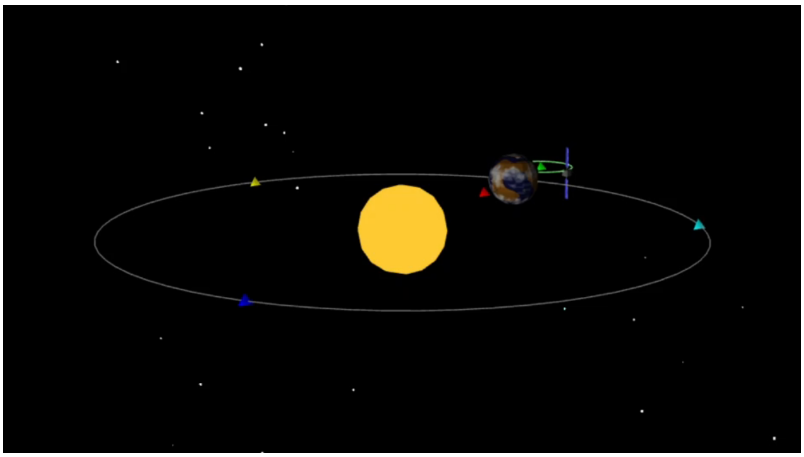

FIGURE 1 – Résultat attendu. Version animée : https://youtu.be/ncpW44AA-n4

a) Création de la scène 3D (5 pts)

Partie réalisée avec OpenGL1.1/GLSIM.js

Vous devez d'abord modéliser une scène 3D statique représentant le système Terre-Soleil, l'orbite de la Terre, les cinq (5) points de Lagrange, l'orbite du télescope James-Webb autour du point de Lagrange L2, ainsi que quelques étoiles dispersées aléatoirement dans la scène.

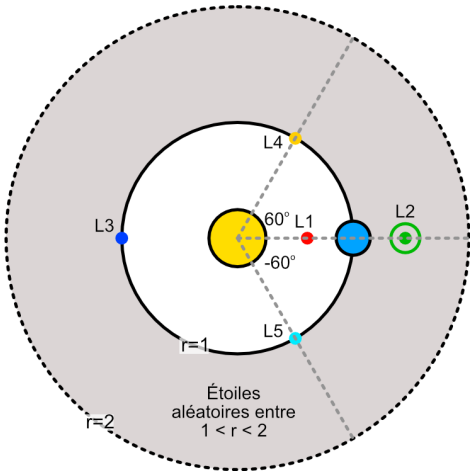

FIGURE 2 – Schéma de la scène 3D à modéliser

Voici les contraintes à respecter pour cette partie.

- Les orbites doivent être visibles même si elles ne sont pas illuminées.
- L'orbite de la Terre doit être un cercle de rayon = 1.
- Le Soleil et la Terre sont représentés à l'aide de la sphère décrite dans le fichier `data/tp2_sphere.obj`, et que vous pouvez importer dans votre application grâce à la fonction `loadOBJFile` fournie.
- Le Soleil doit être d'une couleur jaune/orange et doit être visible même s'il n'est pas illuminé.

- La Terre doit avoir une couleur de base blanche (diffuse, ambiante et spéculaire) et doit posséder une faible brillance.
- Pour dessiner le Soleil et la Terre, modifiez les fonctions `draw_sun` et `draw_earth` respectivement. Vous devez utiliser `glDrawElements` et les fonctions associées pour dire à OpenGL où se trouvent les données de sommets, de normales, et de coordonnées de texture.
- Les cinq (5) points de Lagrange doivent être représentés à l'aide de petites pyramides à base triangulaire (tétraèdre), dont les 4 sommets sont :
  - $v_1 = \left(\sqrt{\frac{8}{9}}, 0, -\frac{1}{3}\right)$
  - $v_2 = \left(-\sqrt{\frac{2}{9}}, \sqrt{\frac{2}{3}}, -\frac{1}{3}\right)$
  - $v_3 = \left(-\sqrt{\frac{2}{9}}, -\sqrt{\frac{2}{3}}, -\frac{1}{3}\right)$
  - $v_4 = (0, 0, 1)$
- Décrivez dans le fichier `tp2_rapport.pdf` comment vous avez défini votre structure de données IFS de la pyramide et comment vous avez calculé les vecteurs normaux associés à chaque face.
- Les pyramides représentant les points de Lagrange doivent être modélisées grâce à la structure de données IFS, et elles doivent être dessinées avec la fonction `glDrawElements`, et ils doivent posséder une faible couleur d'émission en plus de leur couleur de base.
- Les positions angulaires par rapport à la Terre, les rayons et couleurs de chaque point de Langrage doivent être :
  - L1 : $\theta = 0$, $r = 0.7$, couleur = rouge
  - L2 : $\theta = 0$, $r = 1.3$, couleur = vert
  - L3 : $\theta = 180$, $r = 1.0$, couleur = bleu
  - L4 : $\theta = 60$, $r = 1.0$, couleur=jaune
  - L5 : $\theta = -60$, $r = 1.0$, couleur=cyan

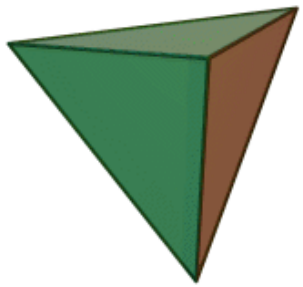

FIGURE 3 – Exemple de tétraèdre (Wikipedia)

- L'orbite du satellite James-Webb doit être représentée par un cercle vert centré sur le point de Lagrange L2, et son rayon doit être 1/8 du rayon de l'orbite de la Terre autour du soleil. (Vous devez réutiliser la fonction `draw_orbit` pour cette partie et appliquer les transformations dans la fonction `draw()`).

— Vous devez placer une source de lumière directionnelle qui illumine le côté de la Terre faisant face au soleil.
— Finalement, vous devez afficher quelques étoiles positionnées aléatoirement autour de votre scène à une distance du soleil entre 1 et 2. Les étoiles doivent posséder une couleur d'émission et elles sont dessinées à l'aide d'une primitive `GL_POINTS`.

Note : Les fonctions à modifier pour cette partie sont indiquées dans le fichier `tp2.js` à l'aide d'un commentaire `//TODO: a)`. Réalisez votre dessin principal dans la fonction `draw()`

b) Créer, afficher et animer une texture (3 pts)

Partie réalisée avec Krita et OpenGL1.1/GLSIM.js

— 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.
— Modifiez également la fonction `draw_earth` afin d'utiliser la texture sur la planète.
— **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).
— **Note** : Pour la texture, vous n'avez pas besoin de reproduire la Terre, vous pouvez dessiner une planète fictive.

c) Modélisation du satellite (4 pts)

Utilisez le logiciel Blender et OpenGL1.1 pour cette partie

Modélisation Blender

Modélisez un satellite fictif pour représenter le télescope James-Webb dans votre application. Modélisez ce 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`)

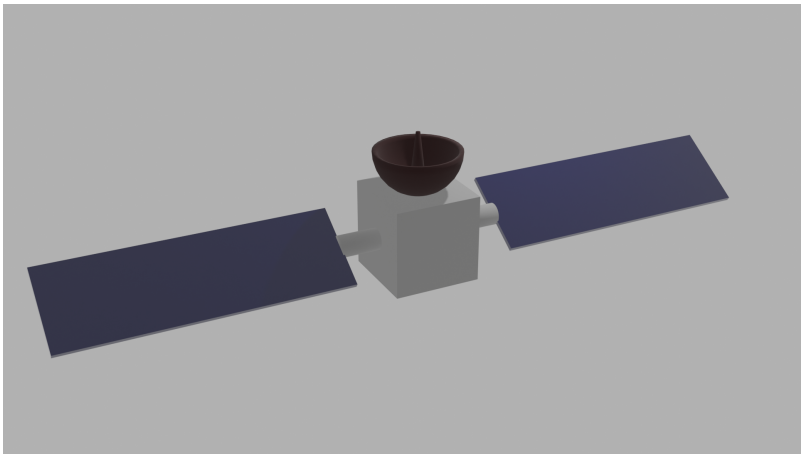

FIGURE 4 – Exemple d'un satellite à modéliser avec Blender

Importation dans l'application

— Dans l'initialisation de la scène 3D, utilisez ensuite 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.
— Placez le satellite sur l'orbite verte centrée sur le point de Lagrange L2. Les panneaux solaires doivent être orientés perpendiculairement au plan orbital.

d) Animation de la scène (3 pts)

Pour cette dernière partie, modifiez la fonction `update()` pour animer la scène. Pour ce faire, utilisez le concept de modélisation hiérarchique, ce qui vous permet de mettre à jour que 3 valeurs dans la fonction `update` pour réaliser votre animation, soit

— Translation `tx` des coordonnées de texture pour simuler la rotation de la Terre autour de son axe.
— Angle de rotation θ_1 pour simuler l'orbite du satellite autour du point de Lagrange L2
— Angle de rotation θ_2 pour simuler l'orbite de la Terre, des points de Lagrange et du satellite autour du soleil.

Note : Le soleil et les étoiles doivent être immobiles.

Instructions générales

- Le travail pratique doit être réalisé en équipe de 2 maximum.
- Vous devez soumettre un seul fichier `zip` qui contient tous les fichiers nécessaires pour tester votre travail, et le nom de ce fichier `zip` doit inclure le nom de famille de chaque membre de l'équipe.
- 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`
  - `tp2_sphere.obj`
  - `tp2_rapport.pdf`
- Veuillez décrire brièvement vos démarches et votre approche de programmation dans le fichier `tp2_rapport.pdf`
- 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. Les images doivent s'afficher.
- Indiquez dans votre rapport quel fureteur web vous avez utilisé pour réaliser le TP. Veuillez utiliser Firefox, Chrome ou Edge.
- 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.

Références

- WebStorm
- Documentation GLSIM
- Documentation OpenGL 1.1
- Futura-sciences : Télescope Spatial James-Webb