ENSSAT

LANNION

Rapport de Projet
Modélisation d'un système solaire en utilisant WebGL

LEVERGER Colin
NANTEL Maëlig
IMR2

1) Introduction

Dans le cadre de la formation Informatique, Multimédia et Réseaux dispensé à l'ENSSAT de Lannion, nous avons étudié la synthèse d'image en seconde année. Cette matière, qui s'inscrit dans la composante « multimédia » de notre cursus, nous a permis de comprendre les concepts clés de la création d'images et de scène 3D par programmation. Nous avons notamment travaillé à l'utilisation de WebGL & JavaScript pour créer des formes animées.

Pour pratiquer et appréhender au mieux les sujets évoqués en CM, nous avons effectué un projet noté. Il s'agissait d'écrire avec WebGL un programme de modélisation de système solaire. Il nous était demandé de développer une simulation réaliste et en 3 dimensions, en utilisant le JavaScript.

Le code associé à ce compte rendu pourra être trouvé dans l'archive jointe. Très peu de code sera directement exposé dans le rapport, mais les structures complexes et les choix d'implémentation des algorithmes y seront clairement explicités.

2) Développement

a) Analyse de l'existant et constats

Pour le développement de notre système solaire, nous sommes partis du code fourni. Sans modifications, le système solaire était composé de formes qui tournaient autour d'une terre. Le code était basé sur les tutoriels en ligne de WebGL suivant : http://learningwebgl.com/blog/?page_id=1217/.

Après un état des lieux du code fourni, nous avons pu tirer les conclusions suivantes :

- Un certain nombre de formes n'allaient pas être utiles pour modéliser notre système solaire (carré, triangle) ;
- Le programme ne chargeait initialement qu'une seule et unique texture, celle de la terre ;
- La rotation autour du soleil pouvait être gérée facilement, car les liens de parenté entre objets étaient déjà établis ;
- Pas de lumière initialisée ;
- Un certain nombre de variables et/ou de buffers n'ont pas étés initialisés, notamment pour les normales des sphères représentant les planètes.

b) Fonctionnalités obligatoires

Un certain nombre de fonctionnalités obligatoires étaient demandées. Nous avons donc commencé par ces premières.

Lors du développement, nous avons suivi les étapes suivantes :

- Chargement des textures multiples. Dans un premier temps, les textures seront chargées dans des variables indépendantes.

- Application des textures sur 4 planètes :
    - Terre
    - Soleil
    - Mars
    - Lune
- Gestion des rotations parents/enfants (pratique pour la lune)
    - Vitesses différentes/direction différentes.
- Application d'une lumière diffuse (qui semble venir de partout à la fois)
    - Pour ce faire, besoin de calculer les normales des planètes (sans normales, pas de lumière possible)
    - Modification des shaders pour ajouter la lumière.

c) Fonctionnalités additionnelles

Une fois ces étapes effectuées, nous nous sommes vite rendu compte du manque de réalisme de la scène. Afin d'y remédier, nous avons ajouté les fonctionnalités suivantes (dans l'ordre de développement).

- Skybox : ajout des étoiles et d'un fond dynamique pour notre scène
- Lumière venant du soleil : pour ajouter du réalisme, la lumière devrait naturellement venir du soleil !
    - Modification des shaders.
    - Ajout d'une variable booléenne « `isLightsource` » si l'objet est une source de lumière.
    - Inversion des normales du soleil pour qu'il rayonne.
- Toggle des lumières avec un bouton.
- Caméra multiple (les caméras annexes étant statiques) et gestion de la caméra améliorée (dans un objet).
    - Gestion d'une caméra dirigée avec la souris.
- Stopper la rotation/augmenter la vitesse de rotation des planètes
- Affichage ou non de la terre.
- Refactoring du code.
    - Passe qualité du code.
    - Optimisations
- Anneaux de saturne : manque de temps pour terminer cette fonctionnalité. L'objectif était de tracer deux cercles ayant pour centre le noyau de Saturne, et d'appliquer une texture sur la partie intérieure se trouvant entre les deux cercles.
- Astéroïdes : fonctionnalité essayée mais non menée à terme. Il nous est apparu plus difficile que prévu de gérer des déformations de sphères pour obtenir une forme ressemblante à un astéroïde.

3) Problèmes rencontrés

Les problèmes rencontrés lors du développement sont les suivants :

- Problème avec la lumière ponctuelle (venant du soleil) et les déplacements : quand on se déplaçait dans la scène avec les flèches du clavier, nous avions la source de lumière qui se déplaçait aussi ! Ce fut une de nos plus grandes difficultés, que nous avons résolues en observant avec précision la mvMatrix.
- Les déplacements et directions de déplacement nous ont aussi posé problème :
    - Nous n'avons pas trouvé comment nous diriger dans la direction du visage (il est en effet possible de tourner l'orientation de la tête vers la droite/gauche/haut/bas). Pour pallier à ce problème, nous avons décidé de réinitialiser l'orientation de la caméra lors des avancements vers l'avant ou l'arrière...
    - Ce dernier problème posera aussi soucis lors du changement de caméra : si la seconde caméra est située derrière la scène orientée dans le sens contraire que celle de départ, un appui sur le bouton avancer va simplement faire reculer le personnage. Pour résoudre ce problème, les caméras additionnelles seront statiques.

4) Répartition des tâches entre binômes

Colin LEVERGER	Maelig NANTEL
- Gestion des textures - Lumière ponctuelle & inversion des normales - Caméras multiples - Boutons + interactions avec l'utilisateur - Affichage ou non de la terre	- Lumière diffuse - Placement de toutes les planètes - Vitesses de rotation + amélioration de l'object planète - Skybox - Refactoring global & qualité du code - Anneaux de saturne et astéroïdes

5) Conclusion

Ce projet était un bon moyen d'asseoir nos connaissances en développement WebGL. Nous avons pu manipuler les shaders, créer des lumières, et utiliser des fonctionnalités poussées de WebGL. C'était aussi un bon moyen de coder en utilisant le langage JavaScript.

Nous avons pris plaisir à développer notre système solaire, car ce travail était assez visuel et ludique : le résultat de nos modifications était rapidement visible, et travailler avec un navigateur web permet de déployer facilement le résultat de nos expérimentations. Le

projet est par ailleurs directement visible à l'URL suivante : http://colinleverger.github.io/solar-system-webgl/.

S'il fallait recommencer ce projet, nous profiterions du cours de M. Nerzic pour avoir une utilisation des shaders plus poussées que celle que nous avons eue. Nous manquions en effet de connaissances pour coder des fonctionnalités plus poussées (utilisation des normales pour des objets singuliers tels que des astéroïdes par exemple) au début de ce projet.