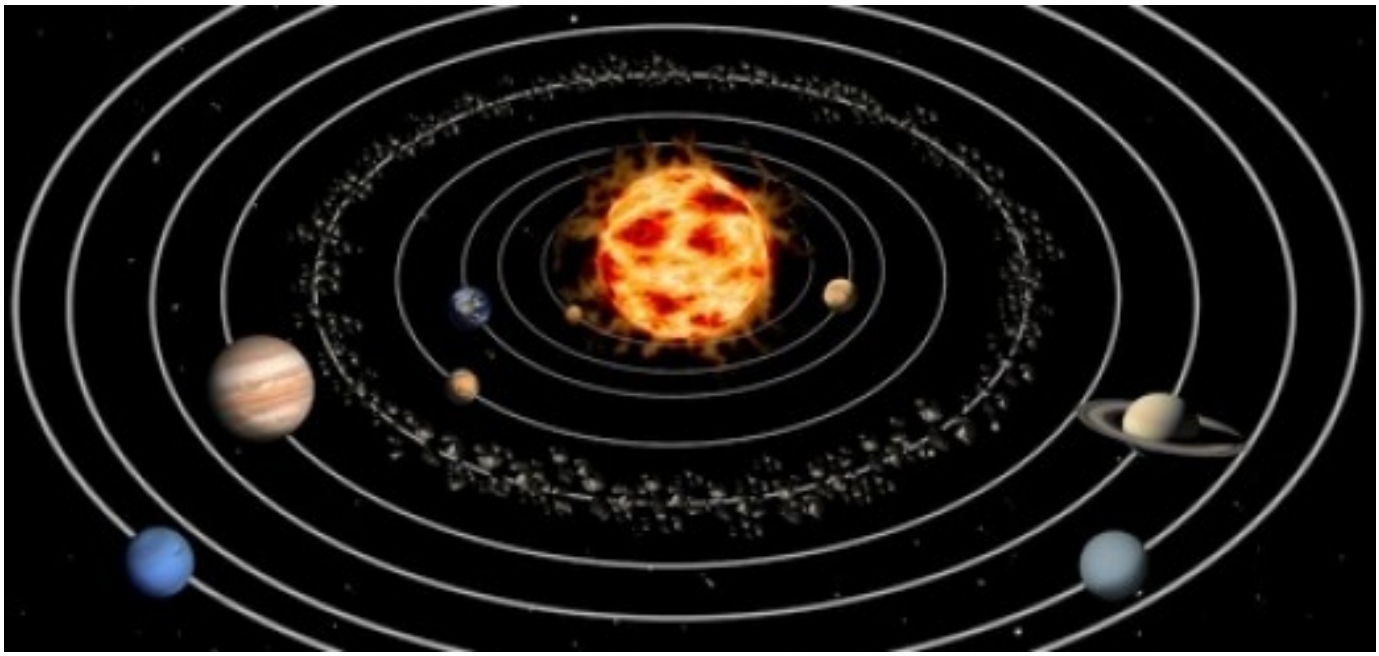
Projet OpenGL final

SpacIMAC

Contacts : biri@u-pem.fr/ maxime.maria@esiee.fr

1 Introduction

L'objectif de ce projet est de développer une simulation scientifique du système solaire. Voici un exemple de ce qui pourrait être attendu :



Vue 3D du système solaire.

La suite de ce document donne les spécifications nécessaires à cette application. **Ce projet est à réaliser en monôme ou binôme**.

2 L'application de visualisation

L'objectif du projet est de créer une application de visualisation scientifique du système solaire qui répond aux contraintes suivantes :

- Programmation en C++.
- Rendu avec OpenGL 3+ (utilisez les shaders, pas de pipeline fixe).
- Fonctionne au moins sous Linux : programme qui compile avec `g++` et exécutable sur les machines de la fac (typiquement celles de la salle 1B077).
- Gestion de la fenêtre et des évènements avec la SDL.
- Pas de fuite mémoire ! (utilisez Valgrind pour vérifier).
- Bonne architecture logicielle (par exemple, séparez bien le côté « Moteur de l'application » du côté « Moteur de rendu »).
- Code propre, commenté de façon pertinente.

— Toute bibliothèque supplémentaire (hormis GL, SDL, glimac, GLew, GLW) doit être validée par vos enseignants ou directement compilée par votre application.
— Réalisé grâce à un outil de gestion de version (type Git) pour partager votre code.

L'application devra permettre de visualiser :
— Le système solaire vu "de haut" (au 3/4 au dessus du plan de l'écliptique) et de profil avec le soleil et les "planète" suivante : Mercure, Venus, Terre, Mars, Jupiter, Saturne, Uranus, Neptune, Pluton.
— Une planète en particulier avec ses éventuels lunes et anneaux
— Ces visualisations doivent pouvoir s'animer en faisant défiler le temps

3 Ressources et données

L'objectif du projet étant de réaliser une application de visualisation scientifique, il sera important de prendre en compte les véritables unités physiques existantes. C'est l'objet de cette partie.

Les éléments à prendre en compte pour la visualisation des planètes sont :
— Son ellipse autour du soleil (**Aphelion / Perihelion**)
— Le diamètre relatif (voir ci-dessous) (**Diameter**) de chaque planète
— La période de rotation (**Orbital period**) par rapport au soleil
— La période de rotation propre de chaque planète (**Length of days** pour faire simple)
— L'inclinaison par rapport à l'écliptique (**Orbital inclination**)

Dans la liste ci-dessus, les termes entre parenthèse correspondent aux données que vous pourrez trouver sur le site de la NASA ici : `https://nssdc.gsfc.nasa.gov/planetary/factsheet/` Concernant le diamètre des planètes, il faut conserver les rapports mais pas la taille absolue sinon on ne verrait rien. Vous devez donc réaliser une homothétie identique pour chaque planète.

Vous considérerez que le soleil a une rotation propre de 25 jours.

Concernant les satellites, vous devrez prendre en compte les suivants :
— Terre : La lune
— Mars : Phobos et Demios
— Jupiter : Callisto, Ganymède, Europa, Io
— Saturne : Mimas, Enceladus, Tethys, Dione, Rhea, Titan, Hyperion, Iapetus ainsi que les anneaux bien sûr
— Uranus : Ariel, Umbriel, Titania, Obéron, Miranda et ses anneaux
— Neptune : Triton, Nereid
— Pluton : Charon

Les données relatives aux satellites peuvent se trouver ici : `https://nssdc.gsfc.nasa.gov/planetary/planetfact.html`

Vous prendrez en compte les caractéristiques suivantes pour les satellites : Diamètre, Ellipse (**Semi-major axis + Eccentricity**), Inclinaison (**Inclination**)

Tous les satellites et planètes seront représentés par des sphères parfaites (même si ce n'est pas le cas). Enfin, vous trouverez un ensemble de texture pour les planètes sur le site suivant : `http://planetpixelemporium.com/planets.html`.

Vous n'avez pas à prendre en compte le positionnement initial de ces planètes et de ces satellites. Seule l'évolution sera à représenter.

4 Spécifications

Interaction avec l'utilisateur

L'utilisateur devra pouvoir (à l'appui de certaines touches par exemple, ou en cliquant sur une planète) basculer entre plusieurs modes de vue :

— Un mode de vue au dessus de l'écliptique du système solaire (plan de l'orbite de la Terre autour du Soleil) à 90 degrés ou à 66 degrés selon ce qui vous semble le plus visible/explicite. Dans cette vue, le Soleil est au centre. L'utilisateur pourra zoomer ou dézoomer.

— Un mode de vue sur le plan de l'écliptique (donc de profil). Là encore le Soleil est au centre de la vue et on peut zoomer et dézoomer. On doit pouvoir également tourner autour du soleil.

— Un mode de vue planétaire qui "zoome" sur chaque système planétaire et ses satellites. La planète sera au centre de la vue et on pourra tourner autour ainsi que zoomer et dézoomer.

D'autre part, à l'appui sur une touche, dans chaque mode de vue, on doit pouvoir avancer (rapidement) dans le temps afin de voir les planètes et satellites évoluer sur leurs orbites en mode pas à pas ou de manière continue (on appui sur une touche et le temps défile jusqu'à ce que l'on réappuie sur la touche).

A l'appui d'une touche, on doit pouvoir sortir (proprement) de l'application.

Apparence de l'application

Sur les deux premiers modes de vues, vous devrez représenter le Soleil et les planètes, texturées, ainsi que les trajectoires de leur orbites.

Sur le troisième mode de vue, on veut voir la planète texturée, ses satellites et ses éventuels anneaux, ainsi que les trajectoires des satellites.

Le fond (l'espace) devra être représenté par une bounding box affichant un "ciel" étoilé.

Spécifications liées au C++

Cette partie concerne **uniquement** les étudiants devant réaliser leur rattrapage en C++. L'évaluation sera effectuée sur la partie concernant la gestion des planètes. Ces étudiants devront également suivre les consignes suivantes :

— Code propre, commenté pertinemment

— Pas de fuite mémoire

— Respectez le principe d'encapsulation

— Créez une hiérarchie pour représenter les planètes. Vous devez distinguer les planètes telluriques des planètes gazeuses.

— Évitez le plus possible la redondance de code (mettre les attributs/méthodes où il faut dans la hiérarchie)

— Utilisez le polymorphisme pour les planètes dans la boucle d'affichage

— Utilisez une `std::map` pour stocker les planètes (attention au polymorphisme)

— Utilisez l'algorithme de la STL `for::each` avec une fonction lambda pour afficher les différentes planètes

En bonus (et pour vous aider dans le développement de votre projet), n'hésitez pas à utiliser :

— Les exceptions pour gérer les éventuelles erreurs d'exécution

— Les asserts

5 Extensions possibles

Une fois l'ensemble des fonctionnalités présentées implémenté, testé et validé, vous pouvez améliorer votre programme en proposant par exemple (attention, certaines extensions sont beaucoup plus complexes que d'autres) :

- L'éclairage du soleil permettant de ne voir que les parties éclairées des planètes (voire des satellites)
- Un mode de vue "freefly" où on peut naviguer comme on le souhaite dans le système solaire.
- Prise en compte d'éléments physiques supplémentaires comme : l'inclinaison des planètes, l'albedo (en lien avec le modèle d'éclairage) ;
- La forme des planètes et des satellites (en général ce ne sont pas des sphères parfaites)
- Le positionnement initial correct des planètes (voire des satellites)
- La visualisation de la ceinture d'astéroïde
- Visualiser d'autres satellites que ceux demandés

Si vous avez d'autres idées, n'hésitez pas !

6 Aide à la gestion de projet

1. Avant de vous lancer tête baissée dans le code, vous devez réfléchir ensemble à l'architecture de votre programme. Cette étape est essentielle à la réussite du projet : une architecture mal réfléchie au départ entraînera des modifications par la suite et donc un retard possiblement conséquent dans l'avancement du projet. Néanmoins, ne passez pas trop de temps sur cette réflexion, l'objectif est d'avoir une architecture simple, répondant aux attentes des spécifications. Il n'est pas nécessaire que votre programme soit ultra-modulable.
2. Optez pour de petits cycles de développement type analyse/spécification/développement/tests/validation.
3. Découpez le projet en tâches courtes, intégrables et testables rapidement, pour pouvoir mieux jauger l'avancement du projet.
4. Définissez, en amont, le rôle de chacun en fonction de ses compétences et ses affinités.
5. Utilisez un outil de gestion de version type GIT, par exemple GitHub (`https://github.com/`) ou Bitbucket (`https://bitbucket.org/`).
6. Utilisez un outil de répartition et suivi de tâches, par exemple Trello (`https://trello.com/`) ou Azendoo (`https://app.azendoo.com`)
7. Codez une surcouche à OpenGL pour faciliter le développement. Elle pourrait contenir des classes simplifiant la gestion des ressources (VBO, textures, *etc.*) ou encapsulant des techniques utilisées régulièrement.

7 Livrables

La veille de la soutenance (*i.e.* fin mai 2018), vous devrez nous fournir par mail (biri@u-pem.fr/ maxime.maria@esiee.fr) :

- le code source : donnez nous accès à votre dépôt Git ;
- un rapport en PDF (environ 4 pages, hors page de garde, tables des matières, *etc.*), contenant au moins :
  - un mode d'emploi ;
  - une description simplifiée de l'architecture du programme (pas le détail de toutes les classes, concentrez vous sur l'essentiel) ;

— la justification de vos choix ;
— une présentation de la gestion de projet (qui a fait quoi, qu'est qui a fonctionné ou non, *etc.*) ;
— les résultats.

N.B. : vous l'aurez compris avec le projet Spacimac, mettre les informations demandées ci-dessus dans le rapport ne signifie pas faire une partie pour chaque... Organisez votre rapport ! Par exemple, une partie « Justification des choix » n'est pas très pertinente. Justifiez vos choix au fur et à mesure.

8 Soutenance

Vous devrez présenter votre projet lors d'une soutenance d'une quinzaine de minutes. Votre présentation devra comporter :

— une démonstration de votre application, en présentant les diverses fonctionnalités (spécialement les fonctionnalités supplémentaires).
— une présentation succincte de l'architecture de votre programme et des bibliothèques utilisées (pas OpenGL, ni SDL) ;
— un résumé de votre gestion de projet (qui a fait quoi, difficultés rencontrées, outils utilisés, si c'était à refaire vous changeriez quoi ?, *etc.*).

Nous attendons de vous une soutenance claire et fluide. Vous devez donc bien la travailler, et faire ou ou plusieurs répétitions avant le jour J (et pas avant la minute M).