Astral Simulation - Pré TPI

Rapport – Ethan Roth et Evan Comtesse, 3IND-3TPMa

Table des matières

[Introduction 3](#_Toc187065984)

[Choix du projet 3](#_Toc187065985)

[Objectif final de la simulation 4](#_Toc187065986)

[Stockage des données 4](#_Toc187065987)

[Aspects physiques du programme 4](#_Toc187065988)

[Valeurs utilisées et conversion 4](#_Toc187065989)

[Calculs et fonctions 5](#_Toc187065990)

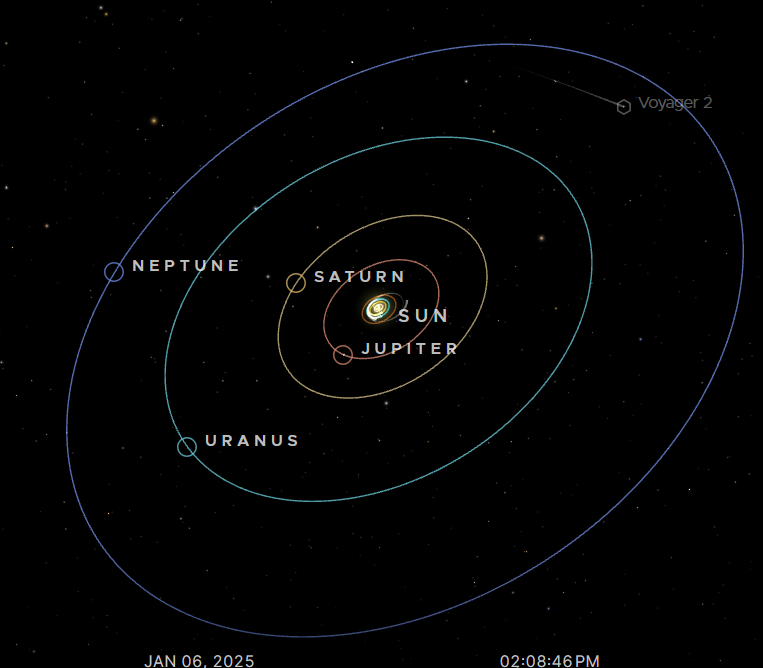
# Introduction

Durant cet atelier, nous avions pour travail de choisir un projet à faire à deux. Le sujet étant relativement libre, je me suis mis avec Evan et nous avons choisi de continuer sur la lancée de la première saison en faisant un projet C# en utilisant la librairie graphique Raylib sur Visual Studio. Nous disposions pour ce projet d’une centaine de périodes. Nous avions peu de contraintes au projet si ce n’est de proposer deux idées aux enseignants engagés avec nous dans cet atelier, M.Vadi ainsi que M.Schenk. Après discussion ils en ont choisi un que nous devions faire. Nous sommes quatre groupes dans la classe. Deux sont évalués par M.Schenk, dont notre groupe, et deux par M.Vadi.

# Choix du projet

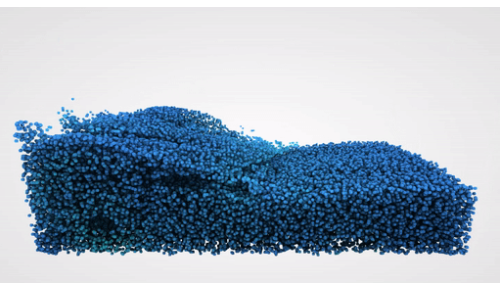
Nos projets proposés étaient les suivants :

1. Solar system simulation

Une simulation d’un système solaire  
(Simulation officielle de la NASA : <https://eyes.nasa.gov/apps/solar-system/#/home>)  


1. Fluid simulation

Une simulation de fluides

(Experiments with Google : <https://experiments.withgoogle.com/fluid-particles>)  


Bien entendu, sur ces deux idées proposées, une seule a été retenue. Celle de la simulation du système solaire. Nous sommes contraints d’avouer que c’est celle que nous préférons car nous sommes des fanatiques de l’espace et de physique, cette simulation est une excellente combinaison des deux.

# Objectif final de la simulation

Le but final de la simulation est d’avoir une caméra que nous puissions déplacer librement dans le système solaire afin d’observer les planètes, leur déplacement et leur rotation. Des shaders sur le Soleil seront également ajoutés afin d’avoir un meilleur éclairage et un meilleur rendu. Lors d’un clic sur une planète, une sidebar est affichée avec diverses informations sur celle-ci. Grâce au flèches directionnelles, nous pouvons nous déplacer rapidement entre les planètes, obtenir un zoom sur celles-ci ainsi qu’à nouveau la sidebar.

# Stockage des données

Les données des planètes (rayon, masse, etc.) sont stockées au format JSON.

# Aspects physiques du programme

## Valeurs utilisées et conversion

Pour rester le plus réaliste possible, nous avons choisi que chacune des valeurs utilisées, que ce soit la masse, le volume, le rayon, la distance par rapport au Soleil d’une planète, soient les vraies valeurs fournies ou par le site Wikipédia, ou par le site officiel de la NASA.

Par des soucis de taille, les valeurs de distances telles que la distance par rapport au Soleil ou encore le rayon de la planète sont divisées par 15'000'000 puis multipliées par 20.

Les valeurs telles que le poids, sont également réduites.

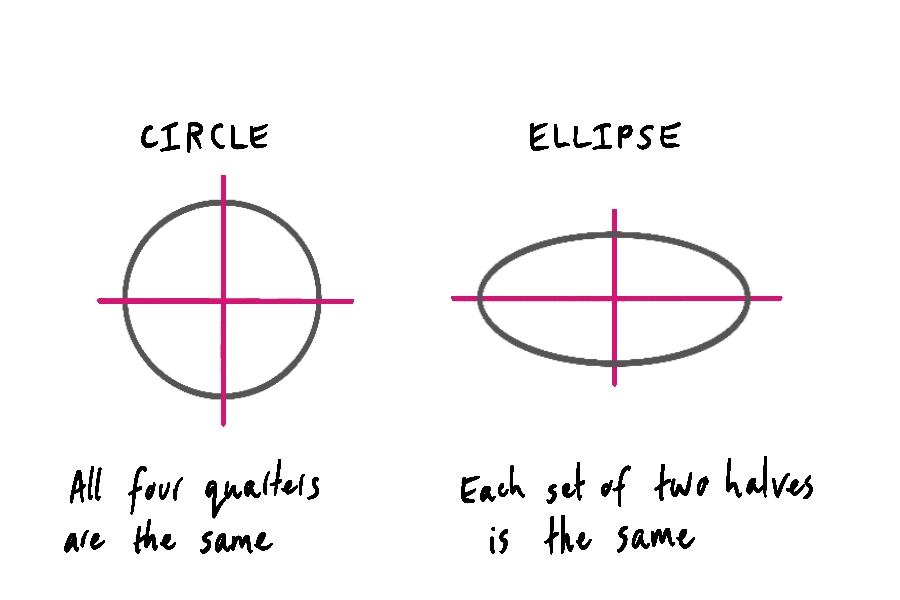
Puis pour l’affichage des données, les valeurs sont toutes reformatées par rapport à la réalité et notées en notation scientifique si trop grandes (exemple ci-dessous pour Neptune) :



## Calculs et fonctions

Au niveau des calculs de physique, il fallait calculer la trajectoire des planètes, calculer leur rotation, leur vitesse, la distance par rapport au Soleil ainsi que la période de rotation (le temps que met une planète à faire un tour sur elle-même).

En réalité une planète ne tourne pas en rond autour du Soleil, elle tourne de manière elliptique.

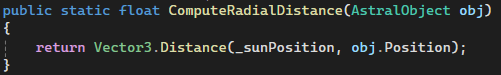


Au début, j’avais (Ethan) pensé et voulu faire le déplacement réel de chaque planète. Toutes les informations se trouvent relativement aisément sur le site de la NASA mais les mettre en place est une toute autre histoire. Ce sont des aspects de physique que je n’ai jamais traité et les implémenter dans le code pour que cela fonctionne m’a malheureusement été infaisable malgré l’aide de l’IA, d’internet et le temps passé dessus. J’ai donc pris la décision de faire tourner dans un premier temps les planètes en rond, puis s’il reste du temps, éventuellement repartir sur les ellipses.

Faire se déplacer les planètes en rond est quelques chose d’extrêmement aisé. Un simple MCU (Mouvement Circulaire Uniforme) et c’était terminé. Il suffit d’avoir une période de révolution (p.ex pour la Terre c’est ~23h et 56min) et un rayon (distance du centre du Soleil au centre de la planète).

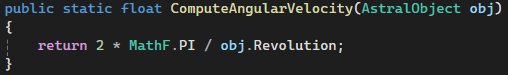
### Fonctions de calcul du rayon

Celle-ci calcule la distance entre les deux vecteurs de position des deux astres (\_sunPosition étant la position du Soleil et obj étant la planète).



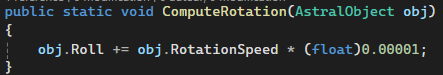
### Fonction de calcul de la vitesse angulaire

Celle-ci calcule la vitesse angulaire de la planète grâce à cette équation où est la période de révolution :



### Fonction de calcul de la rotation sur l’axe Z (roll)

Celle-ci met à jour la position de la planète sur l’axe Z. La multiplication par 0.0001 est là pour réduire la vitesse de la planète. Sans ça elle tourne beaucoup trop vite.



### Sauvegarde d’anciennes fonctions

Toutes les fonctions de calcul de trajectoires des planètes en ellipse sont gardées en commentaire dans le cas où le temps restant est suffisant pour reprendre ce travail-là.