Rapport - Simulation système solaire

Table des matières

1 Introduction

Pour réaliser une simulation d'un système planétaire, plusieurs facteurs sont à prendre en compte. Puisque dans notre univers, tous les corps sont soumis à des forces. Nous allons donc utiliser pour cela les trois lois de Newton :

1. Si un corps est immobile (ou en mouvement rectiligne uniforme), alors la somme des forces qu'il subit, appelée force résultante, est nulle. ($\vec{F} = 0$).
2. La force résultante subit par un corps est égale à la masse de ce dernier multipliée par son accélération. ($\vec{F} = m * a$).
3. Si un corps A subit une force de la part d'un corps B, alors le corps B subit une force de réaction de sens opposé et de même intensité ($\vec{F}_{b \to a} = -\vec{F}_{a \to b}$).

1.1 Réalisation de la simulation d'un système planétaire

1.1.1 Création d'une étoile(dans notre cas : le soleil)

Le soleil sera représenté comme une planète donc aura la structure d'une planète mais son comportement sera différentes des autres planètes. Pour ce faire, on a tout d'abord défini la structure d'une planète :

```
1  typedef struct _planet
2  {
3      double mass; //masse de la planète
4      int32_t color; //couleur sur la simulation
5      double radius; //Grandeur de la planète
6      double semi_major_axis; //Distance par rapport au soleil
7      double eccentricity; //excentricité
8      vec2 pos;       // x(t)
9      vec2 prec_pos; // x(t - dt)
10 } planet_t;
```

Comme convenu, pour la création de notre étoile nous avons donc eu besoin d'implémenter une fonction qui permet de créer les planètes. Cette fonction contient également un champ qui permettra de préciser si cette planète créée est une étoile(`champ: is_star`) :

```
1  planet_t create_planet(double mass, int color, double radius, double semi_major_axis, double ↩
       eccentricity, bool is_star) {
2      planet_t planet = { //attribution de tous les paramètres
3          .mass = mass,
4          .color = color,
5          .radius = radius,
6          .semi_major_axis = semi_major_axis,
7          .eccentricity = eccentricity,
8      };
9
10     if (is_star) { //Si c'est une étoile on le place au milieu donc vecteur 0
11         planet.prec_pos = planet.pos = vec2_create_zero();
12     } else { //Sinon on la place par rapport à sa distance au soleil(étoile) et son excentricité
13         planet.prec_pos = planet.pos = vec2_create(semi_major_axis * (1 - eccentricity), 0);
14     } //La planète est aligné horizontalement au soleil
15     return planet;
16 }
```

1.1.2 Ajout de planètes autour de étoile(Soleil

Ici on a déjà la base pour créer chaque planète, donc on désire en créer chaque planète autour de l'étoile. Pour cela nous avons décidé de créer les planètes à partir d'un fichier *.csv* pour ne pas avoir à modifier tout le code à chaque création de planètes. Notre fichier *.csv* est composé comme ci-dessous :

Name	Mass	Semi major axis	Eccentricity	Radius	Color
Sun	1.989e30	0	0	8	0x00FFFF00
Mercury	3.285e23	57.909e9	0.2056	0.5	0x00E98B41
Venus	4.867e24	108.209e9	0.0067	2	0x001AA89A
Earth	5.972e24	149.596e9	0.0167	1	0x00669EBF
Mars	6.417e23	227.923e9	0.0935	1.5	0x00E26642
Jupiter	1.989e30	378.570e9	0.0489	2.5	0x00F5F5DC
Saturn	568.32e27	1434e9	0.0565	2.25	0x00FFF2CC

Table 1 – Données

Pour créer chaque planète à partir de ce tableau, on a créer une fonction qui permettra de récupérer chaque élément du tableau et l’attribuer dans le bon champs de la planète.