Victor Defrance
Chloé Balmes

Rapport IN104 : N-Body Problem

Description du problème :

On va chercher à modéliser les interactions et à résoudre les équations du mouvement de N corps quelconques dans un champ gravitationnel de dimension 2. Commençons par poser les notations :

- $b_i \; pour \; i \in [\![1, N]\!]$ correspondant au corps numéro i avec m sa masse,
- $p_i(t) = \begin{pmatrix} x(t) \\ y(t) \end{pmatrix}, v_i(t) = \begin{pmatrix} v_x(t) \\ v_y(t) \end{pmatrix} \; et \; a_i(t) = \begin{pmatrix} a_x(t) \\ a_y(t) \end{pmatrix}$ la position, la vitesse et l'accélération du corps i à l'instant $t \in R$.
- Pour deux corps i et j, on note $r_{i,j}(t) = \|p_j(t) - p_i(t)\|$ la distance euclidienne, et $u_{i,j} = \frac{p_j(t) - p_i(t)}{r_{i,j}}$ le vecteur unitaire allant de $p_i(t)$ à $p_j(t)$.
- On note la force gravitationnelle qu'applique un corps j à un corps i de la manière suivante :

$$F_{i,j}(t) = G * \frac{m_i * m_j}{r_{i,j}(t)} * u_{i,j}(t)$$

Ce qui nous donne le Principe Fondamental de la Dynamique suivant :

$$m_i * a_i(t) = \sum_{j \in [\![0,N]\!], j \neq i} F_{i,j}(t)$$

Notre implémentation :

Nous avons travaillé sur 3 grandes parties : la caméra (*camera.py*), la résolution d'équations différentielles (*solver.py)* et l'application à notre problème gravitationnel (*engine.py*) correspondant respectivement aux classes *Camera, Solver* et *Engine*.

Nous ne redéfinissons pas entièrement les classes à chaque fois : chaque classe écrite hérite d'une classe de base dans laquelle nous redéfinissons les fonctions à changer. Cela nous permet notamment de réduire les doublons de code.

Caméra

Le but de cette classe est de préparer l'affichage à l'écran.

Ses fonctions gèrent le changement de repère depuis les coordonnées utilisées pour les calculs à celles que prend en compte pour l'affichage à l'écran. Nous avons eu du mal à définir les deux fonctions *to_screen_coords* et *from_screen_coords* en fonction de la position de la caméra. Au départ, nous n'avions mis toujours le même point au milieu de l'interface quel que soit l'emplacement initial des planètes ou la « position de la caméra ». Nous avons ensuite changé nos formules dans ces deux fonctions pour prendre en compte la position de la caméra. Ainsi les tests *CameraTestCase* ont été vérifiés. C'est à ce moment-là que nous avons décidé d'ajouter des tests pour vérifier aussi notre fonction *from_screen_coords* qui n'étaient pas testée.

Nous allons essayer d'améliorer notre programme pour l'interface graphique en ajoutant la possibilité de se déplacer avec les flèches puis de suivre un astre.

Solver

Cette partie correspond à l'intégration de l'équation de la forme $y' = f(y, t)$.

Nous avions commencé par résoudre tous les calculs dans une seule fonction. Nous faisions une boucle sur tous les corps i, puis sur tous les autres éléments exerçant une force sur i. On appliquait le Principe Fondamental de la Dynamique avec chacun de ses corps en sommant toutes les forces. Puis on modifiait les vitesses et les positions.

Mais nous nous sommes rendus compte qu'il fallait en fait diviser les calculs entre *Solver* et *Engine*. En plus nous n'avions pris en compte que le pas de temps égal à 1. Nous avons donc changé notre fonction *Integrate* avec une recherche du pas optimal de la bonne longueur. On a choisi de calculer la longueur du pas pour que toutes les intégrations se fassent sur le même temps. Nous utilisons la fonction f pour obtenir y'. A l'aide de la formule d'Euler (à l'ordre 1) nous intégrons ainsi la vitesse et la position. Dans cette fonction *Integrate* nous utilisons la classe *Vector* pour alléger les écritures.

Engine

La classe *Engine* correspond au moteur physique de la simulation.

Elle ressence les fonctions *gravitational_force, derivatives(applique le PFD pour trouver y'), make_solver_state (état initial)* qui permettent de faire la résolution du problème.

Nous avons donc repris ce que nous avions précédemment écrit dans le fichier *Solver* puis l'avons retranscrit dans *derivatives* et *make_solver_state*. La fonction *gravitational_force* correspond seulement à l'implémentation de la formule écrite en introduction de la force gravitationnelle.

Pour aller plus loin :

Nous avons réfléchi à quelques améliorations :

- Ajout de tests : *Camera* tester *from_screen_coords*
- Gestion des collisions
- Faire Euler à l'ordre 2
- Déplacement de la caméra avec les flèches du clavier
- Déplacement de la caméra avec sélection d'astres à la souris
- Déplacement de la caméra vers le centre de masse
- Implémentation de la 3ème dimension

Distribution du travail :

Nous avons décidé de travailler et de compléter les programmes ensemble. Nous nous sommes cependant laissé du temps à chacun pour réfléchir plus particulièrement à certaines parties avant de commencer à coder. Chloé s'est chargé de l'interface graphique, avec la caméra, des tests ainsi que du rapport. Victor a plutôt travaillé sur *Engine* et *Solver*.

Lien du GitHub commun :

Inoxagen/IN104_Chloe_BALMES_et_Victor_DEFRANCE: N-Body problem: projet IN104 (github.com)