h e p i a

Haute école du paysage, d'ingénierie
et d'architecture de Genève

Simulation du système solaire

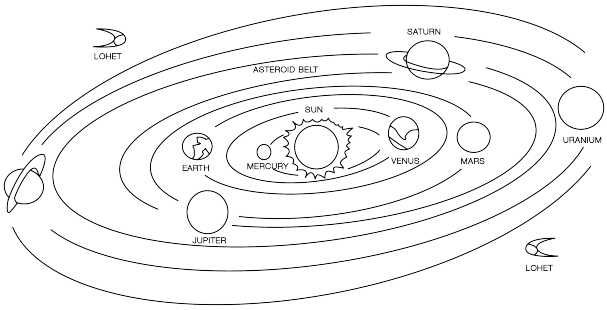
Orestis Malaspinas, Jonathan Lemus

2026

0.1 Introduction

La Physique est une science qui étudie par l'expérimentation et l'élaboration de concepts les propriétés fondamentales de la matière et de l'espace-temps. Ce n'est donc pas qu'une discipline permettant de construire des accélérateurs de particules, des bombes nucléaires ou juste de briller en société. Les concepts de la mécanique, de l'optique ou encore de l'électromagnétisme fournissent les briques de base afin de simuler numériquement des phénomènes souvent inaccessibles en laboratoire. Sans la Physique, pas de *Rocket League*, *Dragon Ball FighterZ* ou encore *Fortnite*. En effet, ces jeux vidéos tournent sous le moteur Unreal Engine, principal concurrent de Unity ou encore CryEngine. Ces moteurs de jeu vidéo, aussi puissants soient-ils, permettent une simulation de plus en plus fidèle à la réalité, grâce à l'intégration de principes physiques. A titre d'exemple, la mécanique permet de gérer les notions de mouvements alors que l'optique géométrique se concentrera sur les effets de *ray tracing*.

Sans prétention de créer un véritable moteur de jeu, l'objectif des travaux qui seront effectués durant l'année est d'appréhender des concepts physiques simples en les implémentant en C.



0.2 Simulation des orbitales planétaires

Dans notre univers, tous les corps sont soumis à des forces. La force de gravitation est une force qui apparaît entre tous les objets (ou corps) ayant une masse. Elle régit le mouvements des objets massifs (planètes, étoiles, trous noirs, galaxies...). Le force gravitationnelle est l'une des quatre forces fondamentales et elle est donnée par la relation

$$\vec{F}_G = \frac{GmM}{||\vec{r}||^3}\vec{r}$$

où m et M sont les masses respectives des deux objets en interaction, r la distance qui les sépare et G la constante gravitationnelle valant $G = 6.67.10^{-11}$ $\mathrm{m}^3.\mathrm{kg}^{-1}.\mathrm{s}^{-2}$.

Notre système solaire est composé d'une étoile (le Soleil) autour duquel gravitent huit planètes ainsi que divers objets (lunes, astéroïdes, comètes...). L'orbite d'une planète n'est pas un cercle parfait, il s'agit en réalité d'une ellipse. Cette orbite elliptique est définie par trois paramètres :

- Le demi-grand axe (a en mètres, *à ne pas confondre avec l'accélération* $\vec{a}_p$),
- Le demi-petit axe (b en mètres),
- L'excentricité (e sans unités).

On note aussi le périhélie comme étant la position orbitale pour laquelle l'objet est au plus proche du Soleil. La figure 1 illustre de façon exagérée l'orbite de la Terre autour du Soleil en fonction des différents paramètres.

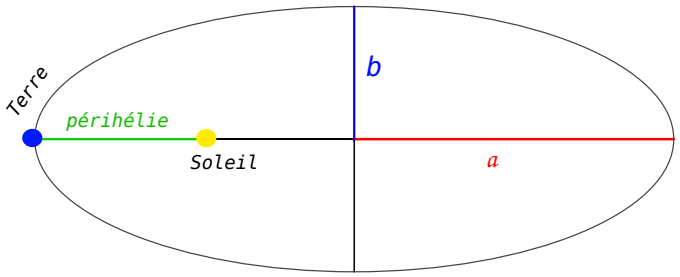


Figure 1: Exemple de l'orbite de la terre autour du soleil. Source: Alexis Durgnat (Bureau A403)

Par ailleurs, l'excentricité définit la forme plus ou moins arrondie de l'orbite comme le montre la figure 2.

0.2.1 Votre mission

Vous rejoignez l'équipe technique d'un studio qui développe des animations pédagogiques sur l'astronomie pour le grand public. Votre cahier des charges vous impose d'écrire en C une application permettant de visualiser les orbites planétaires.

Question directrice:

Le mandat se décompose en cinq étapes, à traiter dans l'ordre:

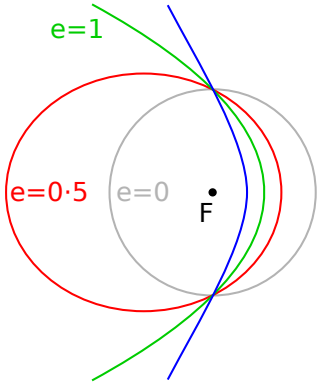


Figure 2: Différentes orbites en fonction de l'excentricité. Source: Wikipédia, https://bit.ly/3x53F4A.

1. poser les hypothèses physiques et le modèle géométrique du problème;
2. définir les forces agissant sur chaque planète;
3. calculer la position de chaque planète à chaque pas de temps de calcul;
4. implémenter et animer les orbites en C;
5. valider les orbites obtenues avec des données réelles.

0.2.2 Livrable et critères de réussite

À la fin de ce projet, vous devez remettre un programme C (basé sur le squelette fourni) qui, une fois compilé avec le `Makefile` donné:

- ouvre une fenêtre *gfx* et y dessine, avec l'API *gfx*, le Soleil, les planètes et les orbites qu'elles dessinent en temps réel;
- fait avancer les planètes à chaque pas de temps Δt selon la formule de Verlet;
- reste lisible: constantes nommées, code commenté, structure proche du squelette fourni.

0.2.3 Étape 1: Contexte, hypothèses et modèle géométrique

Ce système sera inspiré de notre système solaire où toutes les planètes sont sur le même plan (ce qui est une assez bonne approximation de ce qui se passe dans notre système solaire). Au centre nous aurons une étoile (fixe, encore une approximation) et un certain nombre de planètes qui orbitent autour de cette dernière.

Avant de continuer, posez-vous les questions suivantes (elles conditionnent tout le reste du projet):

- quelles forces agissent réellement sur chaque planète?
- est-ce raisonnable de simuler le système en deux dimensions et pourquoi ?
- si l'on s'inspire de notre système solaire, peut-on afficher toutes les planètes ?
- comment gérer l'affichage en fonction des échelles réelles ?

0.2.4 Étape 2: Mettre le mouvement des planètes en équations

Concentrons nous tout d'abord sur l'évolution de la simulation (nous verrons les conditions initiales dans un second temps). Soit une planète p. Pour déterminer les informations nécessaires, on

commence par calculer la force résultante sur celle-ci. En effet, la planète subit l'attraction du Soleil ainsi que des autres planètes:

$$\vec{F}_p = \sum_{\{q \in C | q \neq p\}} \vec{F}_{qp}$$

où C est l'ensemble des corps célestes (les planètes et l'étoile) de votre simulation.

Ensuite, on calcule la nouvelle position à partir de sa position actuelle et de sa position précédente. Pour ce faire, on reprend l'algorithme de Verlet:

$$\vec{x}_p(t + \Delta t) = 2\vec{x}_p(t) - \vec{x}_p(t - \Delta t) + (\Delta t)^2 \vec{a}_p(t)$$

Avec t et Δt en s, $x_p(t)$ la position de la planète p en m et $a_p(t)$ l'accélération de la planète p en $\mathrm{m.s}^{-2}$.

On remarque donc qu'il n'est pas nécessaire de retenir l'évolution de v_p pour calculer le prochain état de notre simulation. Nous n'avons besoin que des deux dernières positions $(x_p(t), x_p(t - \Delta t))$ et de l'accélération $(a_p(t))$. Nous calculerons donc les x_p itérativement et le calcul de $a_p(t)$ se fera à l'aide d'une équation bien connue qui porte le nom d'un grand physicien.

Conditions initiales

Si l'on regarde l'équation précedente, on remarque que pour calculer $x_p(t + \Delta t)$ en t=0 (c'est à dire $x_p(\Delta t)$), il nous faudrait la position $x_p(-\Delta t)$. Comme nous ne considérons pas de temps négatif dans notre simulation, **nous allons fixer la valeur en t=0** avec un développement de Taylor en 0 à l'ordre 2. Ce qui nous donne :

$$\vec{x}_p(\Delta t) = \vec{x}_p(0) + \Delta t \vec{v}_p(0) + \frac{(\Delta t)^2}{2} \vec{a}_p(0)$$

Cette formule est valable uniquement pour la première itération de votre simulation

Trois données que l'on peut aisément trouver en ligne sont:

- la distance au périhélie $\vec{x}_p(0)$,
- l'excentricité orbitale e,
- le demi-grand axe a de l'orbite.

En effet, on rappelle que l'orbite d'une planète est elliptique. Connaissant ces trois valeurs, on peut donc déduire la vitesse au périhélie par la relation:

$$\vec{v}_p(0) = \sqrt{\frac{GM_\odot(1 + e)}{a(1 - e)}} \frac{\vec{r}_\perp}{||\vec{r}_\perp||}$$

où $M_\odot$ est la masse de l'étoile en kg. Le vecteur $\vec{r}_\perp$ est simplement une rotation de 90° du vecteur entre la planète à son périhélie et l'étoile.

0.2.5 Étape 3: Implémenter en C

Votre travail maintenant consiste à utiliser ces connaissances en mécanique pour développer une petite application en C permettant de simuler la rotation des planètes.

Notation vectorielle

Avant de commencer la mise en orbite, un outil fondamental doit être mis en place. Remarquez que les différentes formules données dans cet énoncé sont sous forme vectorielle. Même si nous allons rester en deux dimensions (sauf pour les plus courageux), sachez que les opérations vectorielles facilitent grandement l'implémentation du problème. En deux dimensions, un vecteur possède deux composantes selon les deux axes du repère dans lequel on travaille. Dans les TP précédents, nous avions par exemple v_x et v_y pour désigner respectivement la vitesse horizontale et verticale. Ces deux variables sont les composantes d'un vecteur $\vec{v}$ dans le repère cartésien que l'on note:

$$\vec{v} = \begin{pmatrix} v_x \\ v_y \end{pmatrix}$$

Par exemple: Si l'on reprend la position du boulet de canon dans le TP n°1, elle peut être donnée par le vecteur $\vec{p}(t)$ tel que

$$\vec{p}(t) = \begin{pmatrix} x(t) \\ y(t) \end{pmatrix}$$

avec bien sûr

$$y(t) = -\frac{1}{2}gt^2 + \sin(\alpha)v_0 t + h$$
$$x(t) = \cos(\alpha)v_0 t$$

Pour utiliser cet outil mathématique en C, nous allons utiliser la librairie SDL2 (se trouvant dans le paquet `libsdl2-dev` sur Ubuntu).

Une fois que vous avez installé le paquet requis, parcourez les différentes fonctions implémentées dans le dossier *vec2* du projet. Il est fortment conseillé d'essayer ces fonctions afin d'en étudier leur comportement (comme par exemple la fonction permettant d'effectuer des rotations de vecteur...).

Mise en place de la simulation

Vous avez à votre disposition un squelette de code nommé `tp_planets.tar.gz`. Ce squelette est là dans le but de vous aider à démarrer rapidement. Vous devez impérativement utiliser la méthode et les équations présentées dans la partie théorique de cet énoncé.

Prenez connaissance des fichiers `tp_planets/celestial_body/celestial_body.h` et `tp_planets/main.c`. Ces fichiers contiennent des commentaires pour vous guider.

Pour dessiner vos planètes et votre étoile, vous pouvez utiliser la fonction `draw_full_circle` présente dans la librairie `tp_planets/gfx`.

Conseil: Si vous souhaitez simuler le système solaire, il est recommandé de n'afficher que les quatres premières planètes (Mercure, Vénus, Terre, Mars) pour des raisons évidentes que vous expliquerez.

Remarque: Les calculs que vous effectuerez se font dans les unités du système international (mètre, seconde, kilogramme...). Pour afficher vos résultats dans la fenêtre *gfx*, il faudra procéder à une conversion permettant de passer des mètres aux positions de pixels.

0.2.6 Étape 5: Validation

Malgré les hypothèses "grossières" mises en place pour réaliser cette simulation, il est toutefois possible de vérifier la cohérence des résultats. Pour cela, on peut par exemple mesurer la période de révolution des planètes autour du Soleil et comparer avec les valeurs réelles. On peut aussi comparer avec des lois physiques telles que les lois de Kepler (voir annexe).

Tout résultat de simulation éloigné des valeurs théoriques attendues doit alerter soit sur une éventuelle erreur de programmation, soit sur une hypothèse trop agressive. Quoiqu'il arrive, vous devez être capable de l'expliquer.

Remarque: A titre d'exemple, des étudiants d'une promotion précédente ont eu l'idée de remplacer le Soleil par un trou noir supermassif (que l'on trouve généralement au centre d'une galaxie). Malgré la caractère "stylé" de la démarche, ces étudiants ont conclu sur le succès de la simulation car "les planètes tournent bien autour du trou noir". Cette "validation" qualitative est évidemment insuffisante car la vitesse des planètes en question dépassait la vitesse de la lumière dans le vide. Ceci illustre deux choses:

- la nécessité d'une validation quantitative comme expliqué plus haut
- les limites physiques de nos hypothèses car elles ne prennent pas en compte les effets relativistes aux abord d'un objet aussi massif.

0.2.7 Pour aller plus loin (bonus)

Il est possible d'utiliser le contexte de cette simulation pour étudier la mise en orbite d'un objet autour d'une planète pendant la révolution de cette dernière autour du Soleil.

Vous êtes libres "d'éjecter" tout objet d'une planète avec la bonne vitesse et le bon angle afin que cet objet reste en orbite stable.

0.3 Annexe: les lois de Kepler

Les lois de Kepler sont des lois phénoménologiques permettant de décrire certains comportement des planètes autour du Soleil. Elles sont au nombre de trois que nous énonçons comme suit.

0.3.1 Première loi: Loi des orbites

Les planètes du système solaire suivent des trajectoire en forme d'ellipses dont le Soleil occupe l'un des foyer.

0.3.2 Deuxième loi de Kepler: Loi des aires

Sur la trajectoire elliptique, des aires égales sont balayées en des durées égales. Sur la figure 3 les aires S_1 et S_2 sont balayées pendant le même temps Δt.

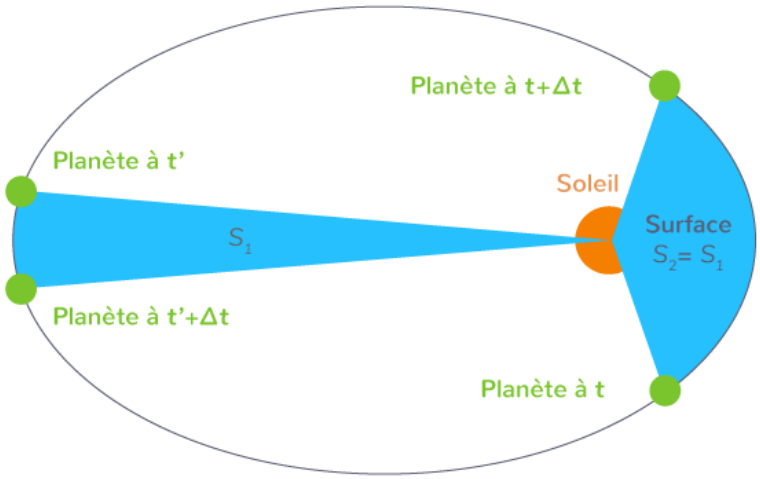


Figure 3: Illustration de la seconde loi de Kepler

0.3.3 Troisième loi: Loi des périodes

Il y a proportionnalité entre le carré de la période sidérale T de la planète et le cube du demi-grand-axe a de l'orbite elliptique. En résumé on a

$$T^2 = ka^3$$

où k est un coefficient de proportionnalité. (Pour rappel, la période sidérale est le temps mis pour réaliser une révolution totale)