Projet d'Algorithmique

Simulation d'une galaxie

1 Introduction

L'objectif de ce projet est d'implanter une simulation des mouvements des corps dans l'espace (en 2 dimensions) et leurs interactions par la gravitation. Le temps de la simulation avance par des pas discrets et après chaque avancement, on met à jour les vélocités et les positions des corps. Ensuite, on visualise les corps, par exemple par la libraire MLV.

Un corp peut être modéliser par la structure suivante :

```
typedef struct _body {
    double px;   /* x position */
    double py;   /* y position */
    double vx;   /* x velocity */
    double vy;   /* y velocity */
    double fx;   /* x force */
    double fy;   /* y force */
    double mass; /* mass */
} body;
```

Vous pouvez y ajouter d'autres champs dont vous avez besoin, par exemple la couleur du corps.

Un petit exemple `twobodies.c` avec deux corps se trouve sur le site du cours et peut être compilé par

```
gcc twobodies.c -o twobodies -lMLV -lm -lSDL
```

Le but du projet est de concevoir un programme qui permet de faire évoluer N corps (où N peut être des milliers) en temps à peu près réel. Vous devez faire deux versions (en ordre) :

- Dans un premier temps, on implante une version "brute force" qui calcule pour chaque corps B, la somme des forces qui agissent sur B par tous les autres corps. Cela revient à une complexité de $O(N^2)$ et devient rapidement trop lent.
- Dans un deuxième temps, on utilise une structure arborescente (un "quadtree") qui permet de calculer les forces plus efficacement et donc d'augmenter davantage le nombre de corps. La complexité de cette méthode est de $O(N \log_2 N)$ (sous certaines conditions).

2 Version A – brute force

Cette version est une simple généralisation de la version à deux corps. Le comportement du programme est le suivant.

pour l'heure `t` allant de 0 à l'infini par pas de `dt`
- pour tout corps B_i, $i = 1, \ldots, N$
  - initialiser la force `(fx, fy)` sur B_i à `(0, 0)`.
  - pour tout corps B_j, $j = 1, \ldots, N$, $j \neq i$
    - ajouter à `(fx, fy)` de B_i la force gravitationnelle exercée par B_j sur B_i (voir le code `twobodies.c`)
- pour tout corps B_i, $i = 1, \ldots, N$
  - mettre à jour la vélocité `(vx, vy)` de B_i
  - mettre à jour la position `(px, py)` de B_i
- visualiser les corps

Noter qu'on ne calcule pas la force gravitationnelle exercée par un corps sur lui-même.

Conseils

Sur le site du cours, vous trouverez des fichiers de données pour tester votre programme. Le format est le suivant :

```
N
WidthOfRegion
px1 py1 vx1 vy1 mass1
px2 py2 vx2 vy2 mass2
...
pxN pyN vxN vyN massN
```

Vous pouvez visualiser les corps à l'intérieur de la région spécifiée par les sommets opposés $(-w/2, -w/2)$ et $(w/2, w/2)$, où w est `WidthOfRegion`.

Vous pouvez également essayer d'initialiser vos propres galaxies mais il faut faire attention à la physique ! L'ensemble de corps devient facilement instable.

Il y a un effet dû à la discrétisation qui fait que des corps qui s'approchent trop récupèrent de l'énergie imaginaire et sont lancés rapidement vers la borne de l'univers. Pour éviter (au mieux) cet effet, vous pouvez remplacer l'expression `(dist*dist)` dans le calcul de la force gravitationnelle par `(dist*dist+C*C)` où `C` est une constante (assez grande, essayer avec `C = 1e4` par exemple).

3 Version B – quadtree

La version brute force devient rapidement très lente pour des grandes valeurs de N ; l'algorithme a une complexité de $O(N^2)$. Ici, on améliore la complexité de la phase de calcul de la force gravitationnelle sur un corps `B`.

C'est une méthode d'approche qui s'appelle une **simulation de Barnes-Hut** :

`https://en.wikipedia.org/wiki/Barnes-Hut_simulation`

Si un groupe de corps est loin de `B`, alors on peut approcher la force gravitationnelle exercée par les corps du groupe par celle d'une masse égale à la masse totale du groupe positionné au centre de masse du groupe (voir la figure 1). Cela permet de faire moins de calculs sur les interactions des corps. Si les corps sont raisonnablement distribués dans l'espace, alors le nombre de calculs avec cette méthodes sera $O(N \log_2 N)$.

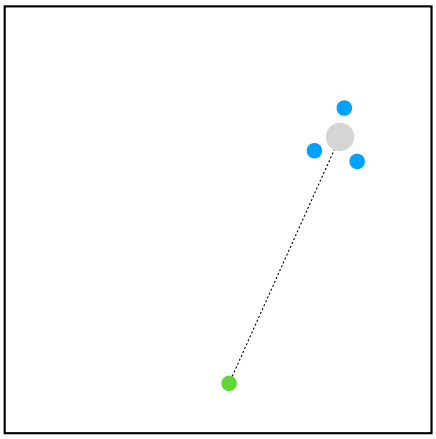

FIGURE 1 – Le cercle gris représente le centre de masse des cercles bleus. Il suffit de faire un seul calcul de la force gravitationnelle.

Vous allez implanter cette idée à l'aide d'une structure de donnée arborescente appelée *quadtree* qui organise les corps. Un quadtree est un arbre enraciné où chaque nœud possède au plus 4 fils. Dans un tel arbre, un nœud représente une région de l'espace : la racine représente l'univers entier et les 4 fils représentent les 4 *quadrants*, ici appelés NW (nord-ouest), NE (nord-est), SE (sud-est) et SW (sud-ouest). Les grand-fils représentent à leur tour le partionnement des quadrants en sous-quadrants.

Les corps sont insérés dans l'arbre afin que
- une feuille contient 0 ou 1 corps ;
- un nœud interne contient la masse et le centre de masse du groupe de corps qui se trouvent dans la région associée au nœud. Les corps de ce groupe sont les descendants du nœud.

Ci-dessous sont décrits le comportement du programme principal et les deux algorithmes principaux : le calcul de la force gravitationnelle à l'aide du quadtree et la construction du quadtree.

3.1 Le programme principal

Le comportement principal de la simulation de cette version devient alors :

pour `t` allant de 0 à l'infini par pas de `dt`
- créer un quadtree avec un seul nœud et 0 corps
- **insérer** tout corps `B` dans le quadtree
- pour tout corps B_i
  - initialiser la force `(fx, fy)` sur B_i à `(0, 0)`.
  - **calculer la force gravitationnelle** sur B_i à l'aide du quadtree
- pour tout corps B_i
  - mettre à jour la vélocité `(vx, vy)` de B_i
  - mettre à jour la position `(px, py)` de B_i
- visualiser les corps
- libérer le quadtree

3.2 Le calcul de la force gravitationnelle

On suppose pour l'instant que le quadtree a été construit (l'algorithme pour le faire est détaillé ci-dessous). Pour calculer la force gravitationnelle qui agit sur le corps `B`, on initialise d'abord les champs `(fx, fy)` de `B` à `(0,0)`. Ensuite, on parcourt le quadtree. Si on rencontre une feuille contenant un corps `C`, alors on ajoute à `(fx, fy)` la force gravitationnelle exercée par `C` sur `B`. Si on rencontre un nœud interne, alors on vérifie si la distance entre `B` et le centre de masse de ce nœud est grande par rapport à la longueur du côté de la région représentée par le nœud. Si oui, alors on ajoute à `(fx, fy)` la force gravitationnelle exercée par le centre de masse du nœud sur `B` et termine le parcours de ce sous-arbre. Sinon, on continue récursivement le parcours dans les 4 fils du nœud.

Formellement, l'algorithme est décrit comme suit :

Calculer la force gravitationnelle
Entrées : la racine d'un quadtree `qt` et un corps `B`
- Si le nœud `qt` est une feuille qui contient un corps `C`, alors si `C` $\neq$ `B`, on ajoute à `(fx, fy)` la force gravitationnelle exercée par `C` sur `B` et on retourne.
- Si le nœud `qt` est un nœud interne, alors on calcule la distance d entre `B` et le centre de masse de `qt`. Soit s la longueur du côté de la région de `qt`.
- Si $s < d/2$, alors `B` est loin du centre par rapport à la taille de la région. Dans ce cas, on ajoute à `(fx, fy)` la force gravitationnelle exercée par le centre de masse de `qt` de `B`. Ensuite, on termine l'exploration de ce sous-arbre.
- Sinon, on appelle récursivement l'algorithme sur les 4 sous-arbres de `qt`.

3.3 La construction du quadtree

Pour construire le quadtree on crée d'abord un nœud qui représente la plus grande région ("l'univers") sans corps. Ensuite, on insère un par un les corps. Pour insérer un corps, on fait une recherche dans l'arbre par rapport à la position `(px, py)` du corps et on navigue toujours vers le quadrant qui contient cette position. Dans chaque nœud interne, on met également à jour le centre de masse et la masse totale de la région représentée par le nœud. Pour ce faire, on utilise la formule suivante pour calculer le centre de masse et la masse totale de deux corps :

$$\begin{aligned} m_{tot} &= m_1 + m_2 \\ x_{centre} &= (x_1 \times m_1 + x_2 \times m_2)/m_{tot} \\ y_{centre} &= (y_1 \times m_1 + y_2 \times m_2)/m_{tot} \end{aligned}$$

Finalement, quand on se retrouve dans une feuille : si la feuille est vide, on insère le corp `B`. Sinon, la feuille contient déjà un corps `C`. Dans ce cas, on partitionne la région représentée par la feuille en quadrants et on insère le corps `B` et réinsère le corps `C` dans les quadrants correspondants. Ici, c'est possible que `B` et `C` seront encore insérés dans le même quadrant, ce qui provoquera récursivement un autre partionnement en plus petits quadrants.

Formellement, l'algorithme est décrit comme suit :

Insérer un corps dans le quadtree
Entrées : la racine d'un quadtree `qt` et un corps `B`

- Si `qt` est un nœud interne :
  - mettre à jour le centre de masse et la masse totale du nœud `qt` pour prendre en compte le corps `B` (qu'on va insérer dans un des fils) :

```
new_mass = old_mass + B.mass
new_centre_x = (old_centre_x * old_mass + B.px * B.mass)/new_mass
new_centre_y = (old_centre_y * old_mass + B.py * B.mass)/new_mass
```

  - appeler récursivement l'algorithme sur le quadrant de `qt` qui contient `B`
- Sinon, `qt` est une feuille :
- Si `qt` ne contient pas de corps, alors on insère `B` dans cette feuille et termine
- Sinon, `qt` est une feuille qui contient déjà un autre corps `C`. Dans ce cas,
  - on crée les quatre fils de `qt` (`qt` devient un nœud interne)
  - on "extrait" `C` de `qt` et on initialise le centre de masse de `qt` à `(0,0)` et sa masse à `0`
  - on insère récursivement `B` et `C` dans les quadrants correspondants

Conseils

Implanter une structure `region` qui représente une région rectangulaire de l'espace (par exemple par deux sommets opposés de la région). Écrire des fonctions qui permettent de créer un quadrant

d'une région (par ex. `region quad_NW(region r)`). Vous pouvez également écrire des fonctions pour savoir si un corps est dans une région (par ex. `is_in_region(region r, body B)`.

Un nœud du quadtree contiendra une région et un pointeur sur un corps (`body`) ainsi que des pointeurs sur ses 4 fils. Si le nœud est une feuille, le corps représentera un vrai corps de la simulation. Sinon, il représentera le centre de masse du sous-arbre enraciné en ce nœud.

3.4 Un exemple

Les figures suivantes montrent l'insertion de 4 corps dans un quadtree initialement vide. (Noter qu'un quadtree vide contient toujours une racine représentant l'univers vide.)

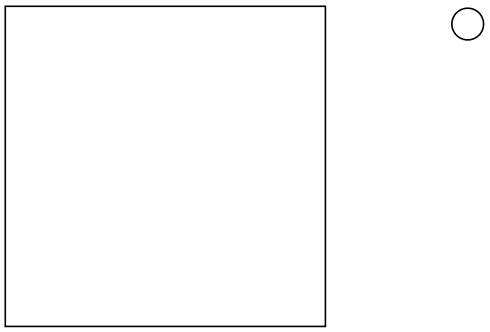

FIGURE 2 – Une région vide.

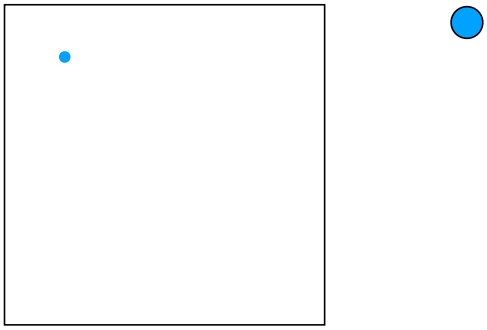

FIGURE 3 – Après l'insertion du corps bleu.

L'insertion d'un deuxième corps (turquoise) provoque le partitionnement de l'espace en 4 quadrants (figure 4).

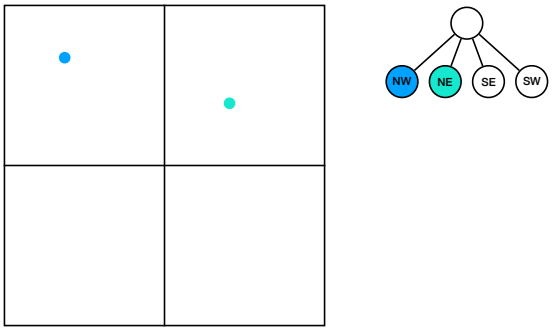

FIGURE 4 – Après l'insertion du corps turquoise.

L'insertion du troisième corps (rouge) provoque d'abord le partionnement du fils NE de la racine pour séparer les corps rouge et turquoise. Ensuite, comme les deux corps sont toujours dans le même quadrant après le premier partitionnement, un deuxième partionnement du quadrant sud-ouest de NE est effectué (figure 5).

L'insertion du quatrième corps se fait dans une feuille vide sous le nœud NE (figure 6).

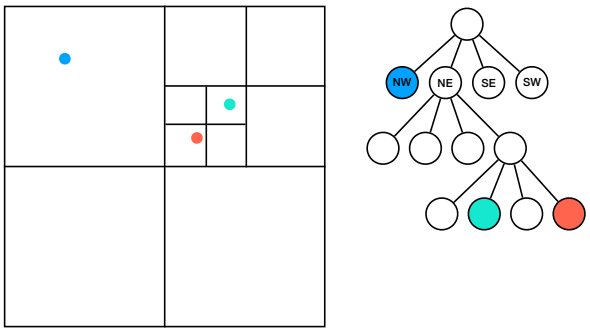

FIGURE 5 – Après l'insertion du corps rouge qui provoque deux partitionnements consécutifs pour séparer le corps turquoise et le corps rouge.

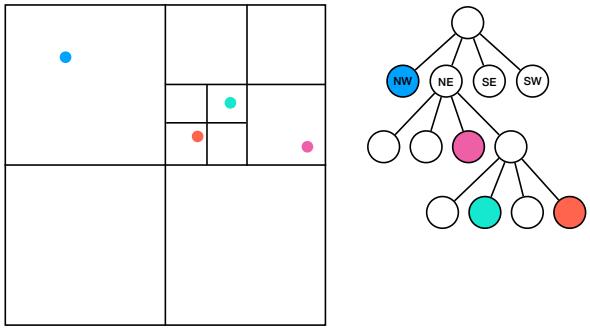

FIGURE 6 – Après l'insertion du corps violet.

Pour calculer la force gravitationnelle sur le corps bleu, on parcourt l'arbre (partiellement). D'abord, on descend de la racine à la feuille NW. Cette feuille contient le corps lui-même qui ne contribue pas à la force totale. Donc on continue le parcours dans le fils NE de la racine (figure 7).

Ici, la longueur du côté de la région, s, est trop grande par rapport à la distance d entre le corps bleu et le centre de masse des trois autres corps. Donc, on continue le parcours dans les quatre fils. Le fils "sud-est" contient le corps violet et on calcule la force gravitationnelle exercée par ce corps sur le corps bleu (figure 8).

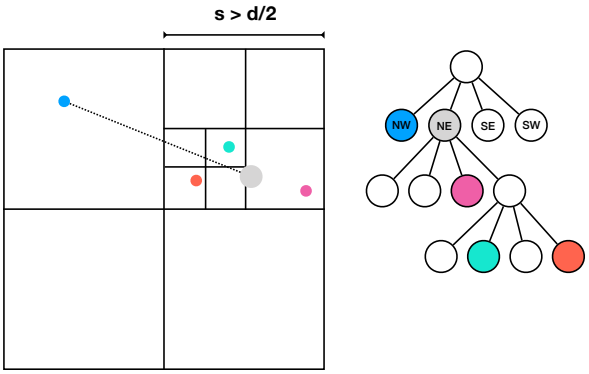

FIGURE 7 – La longueur du côté, s, est supérieure à $d/2$.

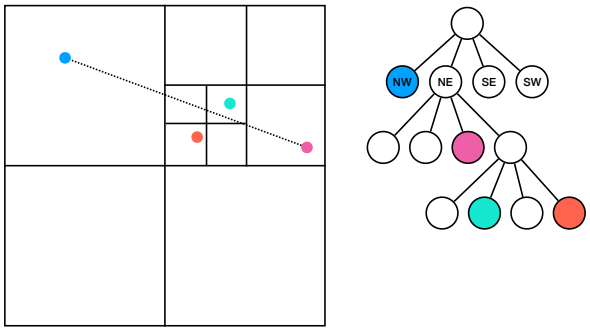

FIGURE 8 – On calcule la force gravitationnelle exercée par le corps violet sur le corps bleu.

Ensuite, on descend dans le fils "sud-ouest" du nœud NE. Ce nœud représente le centre de masse des corps turquoise et rouge. Ici, la longueur du côté est inférieure à la distance divisée par 2 (figure 9). Donc, on utilise le centre de masse pour calculer la force gravitationnelle exercée par les deux corps sur le corps bleu.

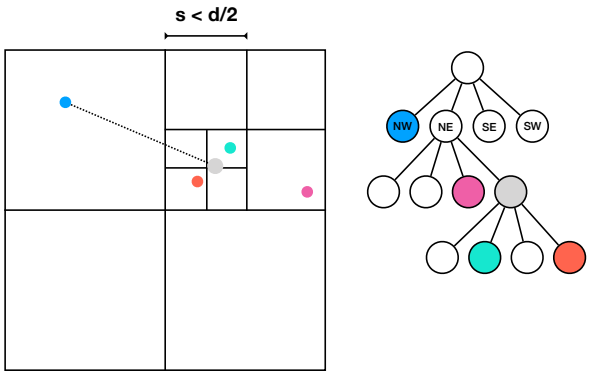

FIGURE 9 – La longueur du côté, s, est inférieure à $d/2$.

Le parcours est terminé : on a dû calculer deux contributions à la force gravitationnelle sur le corps bleu au lieu des trois nécessaires pour la version brute force.

4 Pour aller plus loin

Quelques suggestions :

- Permettez à un petit et un grand corps de fusionner s'ils sont proches.
- Visualisez le quadtree à l'aide des carrés superposés sur la galaxie.
- Créez une animation (pré-calculée, pas en temps réel) d'un grand nombre de corps (100 000 ou plus).
- Gérez l'allocation de mémoire de manière plus efficace (en particulier, évitez `malloc` et `free` dans la boucle principale). Comparez la performance.