Algorithme de Barnes-Hut pour le problème des N-Body

Gargne Fabien

1 Pourquoi utiliser cette méthode ?

Tout d'abord l'algorithme de Barnes-Hut permet de calculer les interactions et forces entre les planètes présentes. Son grand interêt est d'accélérer le traitement et le calcul des forces présentes. En effet la méthode brute qui consiste à prendre à part chaque planète et calculer pour chaque autre planète la force qu'elle exerce sur la première possède une complexité en $O(n^2)$. Barnes-Hut nous permet de passer cette complexité à $O(n * \log(n))$.

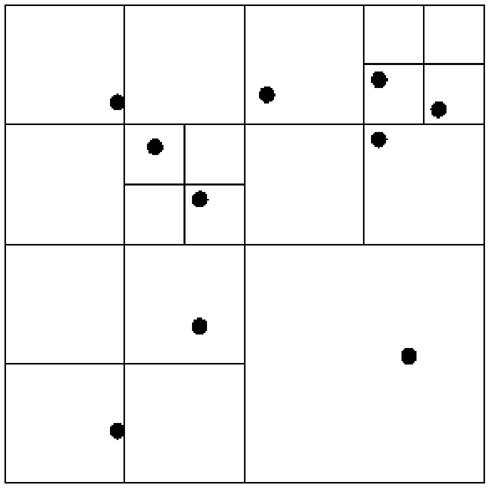
2 Quelques formules utiles

1. Distance entre 2 particules a et b :
$$d = \sqrt{(x_b - x_a)^2 + (y_b - y_a)^2 + (z_b - z_a)^2}$$
2. Force :
$$F = \frac{G * m_a * m_b}{d^2}$$
Avec G la constante gravitationnelle et d la distance entre les particules a et b.
3. Force en (x, y, z) :
$$F = G * m_a * m_b * (\frac{x_b - x_a}{d^3}, \frac{y_b - y_a}{d^3}, \frac{z_b - z_a}{d^3})$$
4. Accélération :
$$a = \frac{F}{m}$$
5. Vitesse :
$$v' = v + \frac{Fnet * \delta t}{m}$$
Avec v l'ancienne vitesse, $Fnet$ la somme de toutes les forces appliquées à la particule et δt l'intervalle de temps.
6. Position :
$$x' = x + v * \delta t$$
Avec x la position de la particule.

3 Comment ça marche ?

Barnes-Hut permet d'avoir de meilleures performances car il se base sur l'approximation. Pourquoi prendre à part chaque particule d'une galaxie trés lointaine au lieu de la considérer comme une seule particule de masse égale à la somme de toutes les particules présentes et une position égale à son centre de masse ? En effet en faisant ceci nous effectuons seulement un unique calcul et non pas n calculs, avec n le nombre de particules. Pour ce faire, nous allons découper notre espace (en 3 dimensions) en 8 parties égales. Chacune de ces 8 parties vont elles mêmes être divisées en 8 parties jusqu'à ce que dans chaque partie d'univers il n'existe au plus une particule.

Par exemple en 2 dimensions le déroulement est le suivant : on découpe à chaque itération notre carré en quatre partie égale jusqu'à ce que chaque carré contienne 0 ou 1 particule, comme suit.



Avec ces parties on crée alors un arbre, un OctTree (QuadTree pour l'ancienne version) qui signifie que chaque noeud possède 0 ou 8 fils, les 8 parties du cube divisé. Partant de là, on place à chaque noeud de notre arbre, deux valeurs essentielles : la masse totale et le centre de masse de cet amas de particule, qui comprend toutes les particules filles de ce noeud. En plus concret chaque feuille de notre arbre va correspondre à une particule et chaque noeud à un amas de particule (pour les approximations). Le noeud racine de l'arbre correspond donc à toutes les particules de l'univers. Enfin lorsque l'on va vouloir calculer la force exercée sur une particule, on va parcourir l'arbre et voir sur chaque noeud si on peut effectuer l'approximation sinon on descend dans les fils et on fait la même chose.

Il nous faut donc en résumé : premièrement créer l'octTree en divisant l'espace puis le remplir en partant des feuilles afin de mettre à chaque noeud sa masse totale ainsi que son centre de masse. Bien sûr il nous faudra le mettre à jour avec les mouvements incessants des particules.

4 Algorithme

L'algorithme se compose donc de trois parties :

1. construire l'OctTree.
2. pour chaque sous-ensemble dans l'OctTree, calculer le centre de masse et la masse totale de toutes les particules qu'il contient.
3. pour chaque particule, descendre dans l'arbre afin de calculer la force exercée sur elle.

4.1 Création de l'OctTree

procedure Creer-OctTree
  octTree ← vide
  pour i de 1 à n **faire**
    inserer-particule-dans-l-arbre(i, octTree)
  finpour
  supprimer-feuilles-vides(octTree)
finprocedure

On essaie d'inserer la particule i au noeud n dans l'octTree.
Par construction, à la fin chaque feuille contiendra soit 1 soit 0 particule. C'est pour cela que l'on doit

lancer la procédure *supprimer-feuilles-vides* pour supprimer les feuilles vides inutiles apres l'insertion de toutes les particules.

procedure inserer-particule-dans-l-arbre(i, octTree)
  si le nombre de particules dans le sous-arbre de racine n > 1 **alors**
    c ← déterminer dans quel fils du noeud n est la particule i
    inserer-particule-dans-l-arbre(i, c)
  sinonsi le sous-arbre de racine n contient une seule particule (n est une feuille) **alors**
    ajouter 8 fils au noeud n dans l'octTree
    mettre la particule déjà dans n dans le fils correspondant
    c ← fils dans lequel la particule i se situe
    inserer-particule-dans-l-arbre(i, c)
  sinonsi le sous-arbre de racine n est vide (n est une feuille) **alors**
    ranger la particule i dans le noeud n
  finsi
finprocedure

Aprés la construction, chaque noeud possède 0 (c'est une feuille) ou 8 fils.

procedure supprimer-feuilles-vides(octTree)
  pour i de 1 à 8 **faire**
  c ← fils N° i
    si c est vide **alors**
      supprimer le fils c
    sinon faire
      supprimer-feuilles-vides(c)
    finsi
  finpour
finprocedure

4.2 Remplissage de l'octTree

fonction calculer-masse(octTree) **retourne** (masse,centre-masse)
  si n contient une seule particule **alors**
    masse ← masse de la particule
    centre-masse ← position de la particule
    stocker (masse, centre-masse) au noeud n
    retourner (masse, centre-masse)
  sinon faire
    pour tous les fils i de n (i = 1,2,3,4,5,6,7,8) **faire**
      (masse(i), centre-masse(i)) = calculer-masse(fils(i))
    finpour
    $masse \leftarrow masse(1) + masse(2) + \cdots + masse(8)$
    $centre - masse \leftarrow \frac{masse(1)*centre-masse(1)+\cdots+masse(8)*centre-masse(8)}{masse}$
    stocker (masse, centre-masse) au noeud n
    retourner (masse, centre-masse)
finfonction

4.3 Force appliquée sur une particule

On parcours l'arbre et on regarde pour chaque noeud (donc chaque cube) si le rapport

$$\frac{D}{r} = \frac{longueurd'unearetducube}{distanceentrelaparticuleetlecentredemasseducube}$$

est assez petit, si c'est le cas alors on peut approximer la force par ce noeud. Sinon on additionne les forces exercées par tous les fils du noeud.

procedure Calculer-forces-exercées
    pour i de 1 à n **faire**
        calculer-force-sur-particule(i, octTree)
    finpour
finprocedure

fonction calculer-force-sur-particule(i, noeud)
    f = 0
    si $noeud$ contient une seule particule **alors**
        f = calculer avec formule N° 3
    sinon faire
        r = distance entre i et le centre de masse du noeud $noeud$
        D = longueur d'une arête du cube correspondant à $noeud$
        si $\frac{D}{r} < \theta$ **alors**
            f = calculer avec formule N° 3 en utilisant la masse et le centre de masse du noeud
        sinon faire
            pour tous les fils c de noeud
                f += calculer-force-sur-particule(i, c)
            finpour
        finsi
    finsi
finfonction

5 To Be Continued ...