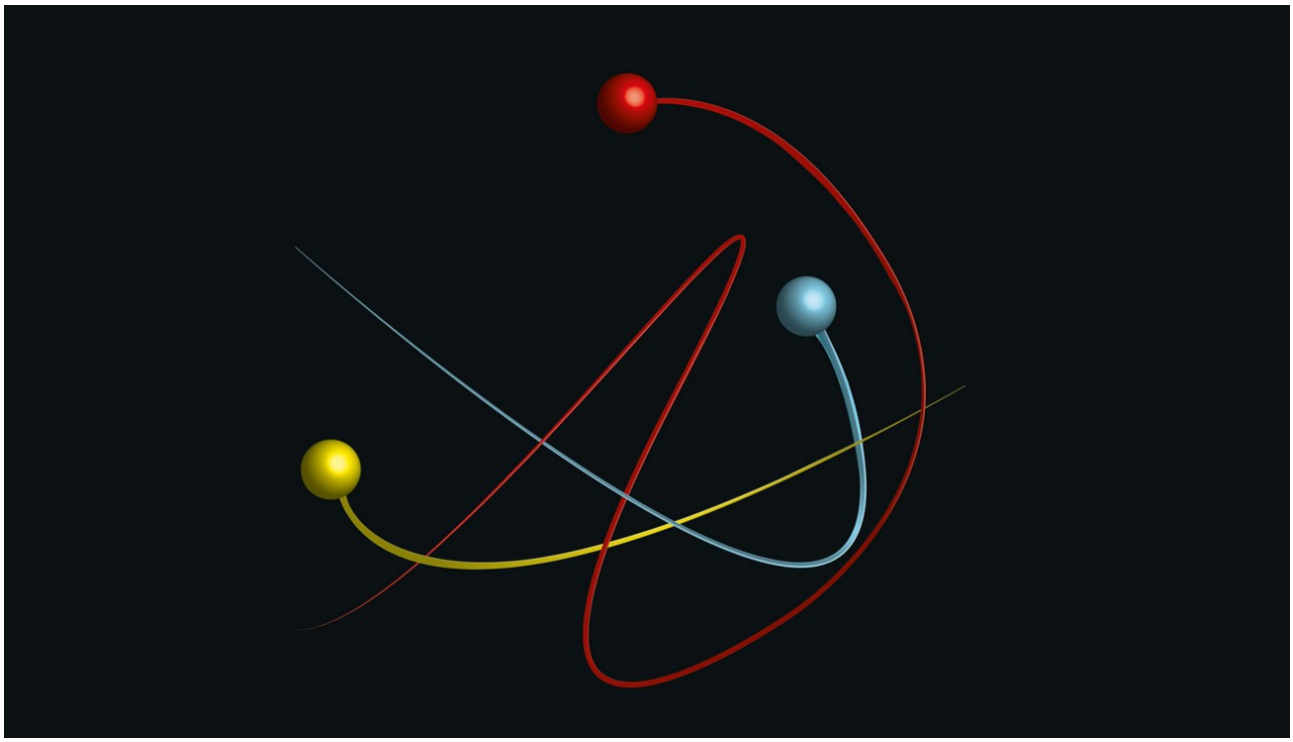
Simulation N-corps



Kessal Yacine, Neveu Gary, Langlois Emilien, Lombardo Arthur

Février 2023

Table des matières

1 Introduction

1.1 Description

Le problème à N corps est un problème de mécanique newtonienne où plusieurs corps se déplacent dans l'espace en étant soumis à leur propre inertie et l'attraction des autres corps. Le but de ce projet était dans un premier de simuler un espace newtonien en 2D où N corps interagissent et de visualiser cette simulation. Il s'agissait ensuite d'implémenter des algorithmes (Barnes-Hut et Fast Multipole Method) et de faire une étude comparative sur la simulation en fonction de l'utilisation ou non de ces algorithmes.

1.2 Problématique

Comment simuler une quantité élevée de corps dans un espace soumis à une mécanique newtonienne ?

1.3 Objectifs

Voici les différents objectif initiaux du projet :

- Implémenter une simulation naïve 2D
- Implémenter une visualisation de la simulation
- Implémenter la simulation avec des quadtrees (algorithme de Barnes-Hut) puis avec la méthode FMM
- Question scientifique : comment évolue les temps de calcul et l'erreur d'approximation en fonction des algorithmes et du nombre de corps ?
- Variante : implémenter des version 3D, paralléliser les algorithmes

1.4 Outils de travail

- Langage : Python
- Visuels : Pygame (Python) et Numpy
- Répartition et organisation des tâches : Trello et Git

2 Organisation du projet

2.1 Objectifs et fonctionnalités

En partant des objectifs donnés nous avons réalisé différentes fonctionnalités :

- Moteur physique :
  - Simuler un corps
  - Simuler plusieurs corps
- Moteur graphique :
  - Représenter grahiquement la simulation de façon basique

  - Ajouter un menu
  - Ajouter un menu de configuration
  - Ajouter une visualisation de statistiques

- Algorithmes :
  - Implémenter Barnes-hut
  - implémenter FMM

- Autres :
  - Système de fichier excel pour la configuration d'une simulation
  - Système de sauvegarde des données et statistiques des simulations

2.2 Gestion de projet

Dans une démarche de gestion de projet, nous avons mis en place une routine de travail afin d'avoir un rythme régulier sur l'envoie du travail de façon a garder nos séance de Projet2 avec Monsieur Wone pour corriger nos erreurs ou avoir des aides ou informations sur les algorithmes. Les 2 outils très utiles ont étés Trello et Git.

2.2.1 Trello

Trello nous a permis de nous organiser tout au long du projet. Nous avons utilisés le système de sprint, chaque sprint représente une semaine. Ainsi à chaque sprint, nous nous attribuons chacun un ou plusieurs objectifs à réaliser parmis ceux fixé ou non résolu la semaine passé. Au fur et à mesure des semaines, nous avons pu observer une net évolution de notre projet prenant forme petit à petit.

2.2.2 Git

Nous avons choisis d'utiliser Git pour plusieurs raison, on pourrait citer le fait que Git est la norme du travail sur projet, pour la gestion de plusieurs branches de travails afin de tous rassembler et surtout que nous sommes plus familier avec celui-ci et qu'il nous permet de nous améliorer aussi. Nous fonctionnons avec plusieurs branches. Tout d'abord la branche master, elle est la branche principale celle sur laquelle ne figure que le code fonctionnel et surtout elle est la plus stable. Nous avons aussi une branche dev qui à l'inverse de master ne sert qu'à mettre le code le plus avancée. Dès que la branche semble stable nous fusionnons dev sur master. Lorsqu'une personne commence un nouvel objectif il se crée une branche portant le nom de ce qu'il fait avec une petit description une fois fini, il push en ligne et nous faisions des tests dessus de façon a être sur que la branche ne va pas crée d'énorme problème lorsqu'elle fusionnera sur dev.

2.3 Répartition du projet

2.3.1 Répartition en temps

Nous avons séparés les tâches en deux temps : avant et après l'évaluation de mi parcours comme suit :

1. Avant :

- Réaliser le premier moteur physique
- Réaliser un moteur graphique basique et fonctionnel

2. Après :

   - Développer de nouvelles fonctionnalités facilitant l'utilisation de l'application ou son visuel
   - Implémenter les algorithmes
   - Expérimenter grâce aux algorithmes et un système de visualisation des statistiques.
   - Intégrer des systèmes de sauvegardes json et d'import Excel

2.3.2 Répartition dans le groupe

Notre groupe est composé de 4 membres. La distinction des rôles de chacun dans le projet c'est fait assez naturellement, Arthur et Yacine se sont intéréssés à la partie physique de l'application et Gary et Emilien à la partie visuelle. Nous nous sommes donc divisés en 2 groupes :

- Yacine et Arthur pour la réalisation du moteur physique et l'implémentation des algorithmes.
- Emilien et Gary pour la réalisation du moteur graphique, des fonctionnalités annexes et le développement autour des statistiques.

3 Architecture du projet

Notre modèle d'architecture s'appuie sur le modèle MVC :

- Un dossier Controller contenant les différentes classes utilitaires pour effectuer diverses opérations.
- Un dossier Model contenant les classes associées à la physique et simulation des corps.
- Un dossier Vues contenant les différentes vues.

Nous avons aussi d'autres dossier contenant différents fichiers et objets nécessaire au fonctionnement du projets :

- Un dossier Assets contenant nos différentes images et objets graphiques ou polices d'écritures.
- Un dossier Data contenant nos fichiers de sauvegarde en Json ou nos exemples de fichier Excel.
- Un dossier Vendor contenant l'ensembles des fichiers secondaire mais utile dites librairies.

4 Simulation des corps

4.1 Un corps

La première chose à faire fut de créer un corps, c'est-à-dire de créer une classe représentant un objet et de lui affecter des attributs. Ainsi un corps possède :

- Une position en X et Y.
- Une vitesse.
- Une accélération.
- Une masse.

4.2 Deux corps

La deuxième chose à faire pour développer le moteur graphique était de tester la physique avec deux corps. Nous avons utilisé un système similaire au moteur de jeu vidéo Unity : A chaque image, on calcule la force d'un corps par rapport à l'autre avec la formule de la loi de gravitation universelle puis on met l'accélération des corps à F/m (F la force et m la masse du corps), on ajoute l'accélération à la vitesse, et on ajoute la vitesse à la position. C'est le fondement de tout ce que nous ferons par la suite.

4.3 Plus de corps

Lorsque la simulation avec 2 corps était fonctionnelle nous nous sommes attelés à essayer de voir ce qu'il se passerait si on avait plus de deux corps à la fois. Lorsqu'un corps est attiré par plusieurs autres en même temps, la force appliquée sur ce corps est simplement la somme des forces d'attraction par rapport à chacun des autres corps. Il suffisait donc à chaque image pour chacun des corps de réinitialiser l'accélération puis d'y ajouter la force par rapport à chacun des autres corps. C'était donc plutôt simple de passer de deux à trois corps, puis de passer à une quantité variable de corps. La classe MoreBodiesSimulation a donc été créée pour ça et c'était la première classe que nous avons essayé d'afficher avec le moteur graphique. Nous avons donc pu nous rendre compte des problèmes de manière plus précise qu'avant et avons passé quelques heures à corriger les petits problèmes/oublis qui faisait que les simulations ne se passaient pas comme supposées. Le processus était un peu long car c'est assez difficile d'estimer ce qui est censé arriver sans passer des heures à faire des calculs de physique.

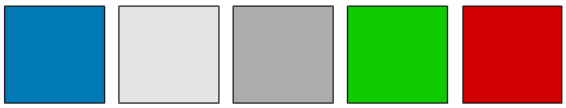
5 Visuels avec Pygame

5.1 Général

L'application est constituée "d'écran" c'est à dire des fonctions représentant chacune les pages/écrans de notre application. Chaque écran peut être lancé ou stoppé par une variable booléenne qui lui est attribuée. Le code de chaque écran est assez semblable : déclarer des variables et objets graphiques, une boucle pour afficher les objets et gérer les évènements.

5.2 Charte graphique

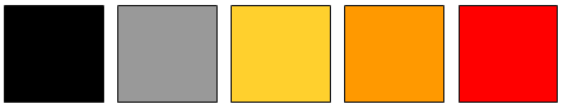
Nous avons décidé d'utiliser pour le fond, les couleurs de police et les objets graphiques majoritairement le bleu et les deux nuances de gris ci-dessous. Puis pour la gestion d'erreur et de succès la couleur rouge et verte ci-dessous :



Concernant la police d'écriture nous avons choisi la "Poppins" qui à été importé depuis "Google Font" :

Texte de Démonstration

Pour les couleurs des corps et la représentation de leurs masses nous avons choisis ces 5 couleurs ci-dessous :



5.3 Boutons et objets graphiques

La plupart des objets graphiques comme les boutons ou les icones sont tirés du site Flaticon. Nous les avons modifiés par la suite pour les décliner sous différents aspects notamment pour simuler des animations.

5.4 Textes

Pour l'affichage des textes nous avons définis une police d'écriture. Pour simplifier l'affichage de texte nous avons utilisé une fonction "display_text"

5.5 Ecran

Notre applications se compose de 5 écrans/pages :

- Le menu
- La notice
- Le menu de configuration
- La simulation
- Les statistiques

5.6 Corps

Nos corps sont représentés par des cercles pleins. Ils sont de différentes tailles et couleurs pour représenter graphiquement leur masses plus ou moins importantes.

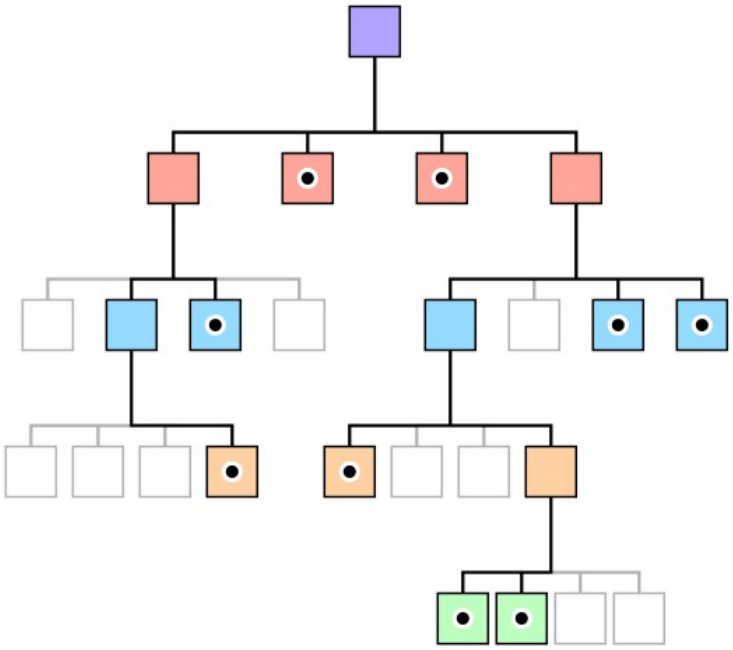
6 Algorithmes

6.1 Introduction

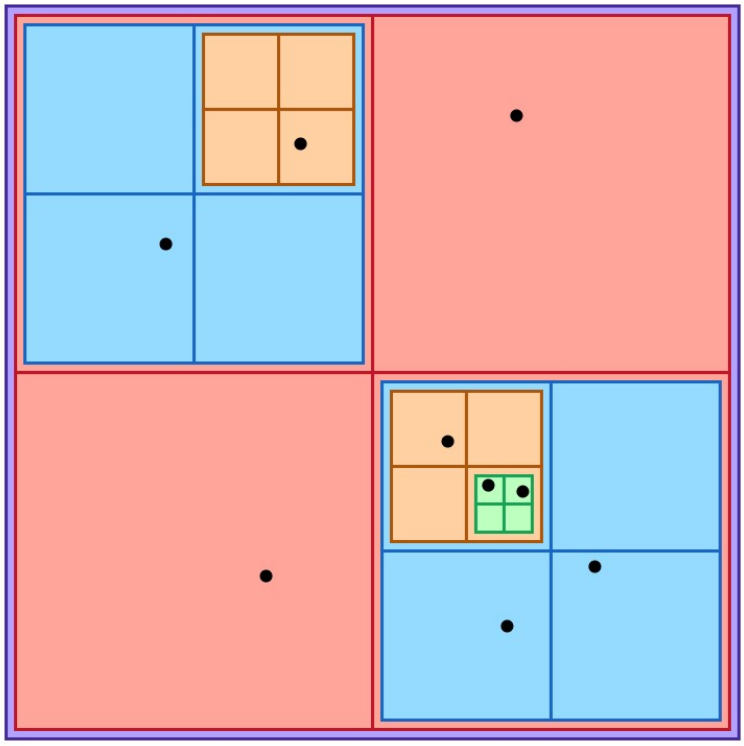
Lorsque la classe MoreBodiesSimulation était fonctionnelle, Arthur et Yacine se sont intéressés à l'implémentation d'algorithmes plus rapides (cf partie 7). Le pdf contenant la liste et la description des différents projets contenait des liens vers des explications sur les différents algorithmes de simulation de n-corps BarnesHut et FMM ainsi que sur la structure de donnée qu'ils utilisent tous les deux : les QuadTree. Nous allons donc d'abord présenter les QuadTree avant de parler plus en détail de comment ils sont utilisés dans chacun des algorithmes.

6.2 QuadTree

Un QuadTree est une structure de donnée qui peut être vue de différentes manière. La plus simple à expliquer est simplement celle d'un arbre avec chaque noeud possédant 4 fils :



Cependant, il est plus simple de comprendre comment la structure fonctionne et comment elle est créée et utilisée avec l'explication suivante : Un QuadTree et une manière de diviser un espace en 2D en 4 de manière récursive. C'est à dire qu'on va tout d'abord diviser l'espace en 4 une première fois, puis rediviser chaque subdivision en 4 au besoin et ainsi de suite :



Cela permet dans notre de regrouper des corps sous une même subdivision afin de faciliter les calculs. Plus de détail dans l'explication de chaque algorithme.

Une première version de la classe QuadTree à été réalisée que nous avons ensuite pu décliner en QuadTreeBarnesHut et QuadTreeFMM en fonction de l'algorithme l'utilisant. Cette classe contenait la liste des corps dans l'ensemble des noeuds ainsi que le noeud initial (la racine de l'arbre). La classe Noeud du même fichier est la vraie structure de donnée et ses attributs sont : sa position, sa taille, les corps qu'elle contient si c'est une feuille et ses enfants sinon. Il a également créé les méthodes "division" qui permet de diviser le noeud pour lui créer ses 4 enfants et de répartir les corps correctement dans chacun des enfants, et la méthode "insert" qui sert à insérer un corps en choisissant le bon enfant dans lequel l'insérer ou s'ajoute à soi-même le corps si c'est une feuille. Il y avait donc tout ce dont on avait besoin pour créer le QuadTree.

Pour les algorithmes suivants il a fallu modifier un peu ces bases car ils n'utilisent pas tous les deux le QuadTree de la même manière.

Barnes Hut requiert que chaque feuille n'ait qu'un seul enfant. La classe QuadTreeBarnesHut a donc été modifiée de manière à ce que la création d'un QuadTree ne s'occupe que de la création de la racine, et chaque ajout de corps va occasionner une subdivision si la feuille dans lequel il est censé aller contenait déjà un corps. Une fois tous les corps ajoutés, Barnes Hut demande à ce que toutes les feuilles ne contenant aucun corps soient supprimées, il a donc fallu créer la méthode "removeEmptyLeaves" pour ça.

FMM quant à lui nécessite de créer tout d'abord un QuadTree de profondeur (c'est à dire le nombre de division) fixe, puis d'y ajouter tous les corps peu importe combien il y en a dans une feuille.

6.3 Barnes Hut

6.3.1 Fonctionnement

Barnes Hut est le premier algorithme que nous avons essayé d'implémenter. Son fonctionnement n'est pas très complexe : A chaque image, il faut créer un QuadTree tel expliqué juste au dessus. Ensuite, l'objectif est de calculer le centre de masse et la masse de chaque noeud (feuille ou non), ce qui se fait plutôt bien de manière récursive.

Enfin, la dernière partie correspond au calcul des forces. Pour un corps donné, au lieu de calculer sa force par rapport à chaque autre corps, on va traverser le QuadTree en partant du premier noeud. Voici la partie la plus importante : A chaque noeud, si le noeud est assez loin du corps dont on essaie de calculer la force, on peut approximer l'attraction de ce corps par rapport aux corps à l'intérieur du noeud à celle de l'attraction du corps par rapport au centre de masse du noeud. Cela permet de ne pas perdre de temps à calculer précisément la force par rapport à chacun des corps et donc de gagner du temps. Le seul désavantage est que l'utilisation d'approximations mène à des erreurs de calcul.

6.3.2 Précision

La simulation ne se déroulera donc pas parfaitement selon les lois de la physique mais plutôt "approximativement" selon celles-ci. Plus la précision est importante, et moins les approximations seront utilisées donc plus le temps de calcul sera grand. A l'inverse, moins la précision est nécessaire, et plus les approximations seront faites pour réduire le temps de calcul. Il y a donc un paramètre de précision pour la classe BarnesHutSimulation qui va être pris en compte lors du calcul des forces de chaque corps.

6.3.3 Ajouts

Une fois la première version de l'algorithme terminée, il restait encore quelques cas dans lesquels l'algorithme ne fonctionnait pas. Le plus embêtant était lorsque les corps sortait de

l'écran car on ne pouvait alors plus calculer la force par rapport à eux. En effet, le QuadTree créé dans BarnesHutSimulation possédait une taille fixe définie à la taille de l'écran. Pour pallier à cela il a fallu deux choses : Une recherche dynamique du corps de plus éloigné dans chacune des 4 directions du plan afin d'avoir la taille du QuadTree, mais également un décalage des corps de manière à ce qu'aucun d'entre eux n'aient des coordonnées négatives. Sans cette deuxième étape, le QuadTree ne pouvait pas répartir correctement les différents corps dans l'arbre sans des modifications majeures.

6.4 FMM

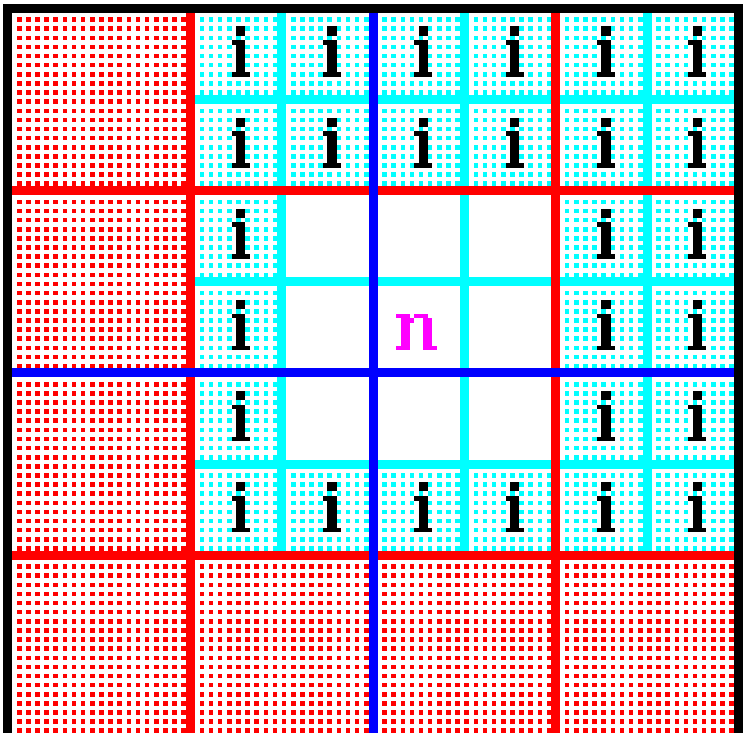
L'algorithme FMM (Fast Multipole Method) et à été bien plus compliqué à développer et implémenter en comparaison avec Barnes-Hut. Premièrement, les explications présentes dans le lien fourni étaient moins complètes que celles pour Barnes-Hut. De plus FMM demande beaucoup plus de calculs complexes et moins intuitifs que son homologue, et enfin les formules pour le calcul de certaines valeurs n'étaient pas fournis et/ou ne sont pas très clairs.

Son fonctionnement reste cependant quelque peu similaire à celui de Barnes-Hut, sauf que contrairement à ce dernier on calcule pas la force mais l'expression du potentiel à chaque point, On calcule le potentiel des corps uniquement s'ils sont proches et sinon on utilise une approximation. La différence réside dans la manière de calculer cette approximation et aussi dans le fait que FMM utilise beaucoup plus d'informations pour le calcul.

Il faut donc tout d'abord créer le QuadTree comme expliqué plus haut. Il y ensuite 2 étapes de calcul à faire sur chaque noeud : le moment multipolaire extérieur(outer) et le moment multipolaire intérieur(inner). On va donc parcourir l'arbre depuis les feuilles afin de calculer ce moment multipolaire extérieur puis faire un parcours en profondeur pour calculer le moment multipolaire intérieur

Le moment multipolaire extérieur d'un noeud correspond au potentiel que les particules situées dans ce noeud exercent sur une particule située loin de ce noeud. Il est calculé pour une feuille comme étant une liste de : la masse totale des particules à l'intérieur du noeud, le centre du noeud, et des coefficients alpha que l'on obtient grâce à un calcul du développement de Taylor du potentiel des points à l'intérieur du noeud. Lorsque nous avons calculé le moment multipolaire des feuilles, nous devons le transférer aux parents grâce à un décalage de moment multipolaire extérieur. Pour faire court, on va récupérer le moment multipolaire extérieur de chacun des enfants et on va en changer le centre pour qu'il corresponde au centre de son parent puis l'ajouter au total. C'est ce total qui sera le moment multipolaire extérieur du parent.

Une fois ces calculs effectués, il est temps de calculer le moment multipolaire intérieur de chacun des noeuds. Ce moment correspond au potentiel des particules à l'intérieur du noeud par rapport aux particules à l'extérieur (suffisamment loin). On met tout d'abord le moment multipolaire intérieur de la racine à 0 puis pour chaque noeud récursivement : on fait un décalage du moment multipolaire intérieur du parent de ce noeud vers le noeud puis on ajoute à cela un décalage du moment multipolaire extérieur des particules de son set d'interaction vers un moment multipolaire intérieur du noeud. Ce set d'interaction définit les noeuds dont le parent est voisin du noeud en train d'être calculé mais qui ne sont pas directement son voisin (cf image ci-dessous).



Les interactions avec les noeuds plus loins (en rouge sur l'image) auront en fait déjà été calculé dans le moment multipolaire intérieur du parent et celles avec les noeuds plus proches (en blanc) se feront plus tard car ils sont trop proches pour se satisfaire d'une approximation.

La dernière étape est donc de calculer les interactions avec ces particules restantes, ainsi que celles à l'intérieur de chaque feuille. On va donc pour tous les corps ajouter la force corps par corps (comme au début) par rapport aux corps présent dans la même feuille et dans les feuilles voisines.

La précision de cet algorithme dépend de la profondeur de l'arbre choisie au début, et c'est donc le paramètre de création du QuadTree.

7 Calculs et simulation

7.1 Protocoles

Les tests ci-dessous ont été effectués sur une machine. Pour chaque algorithme nous avons effectués 10 mesures avec respectivement : 25, 50, 75 et 100 corps. Pour chaque mesure nous avons récupérés le nombre de corps, le nombre d'appel de fonctions et le temps XXXXX

7.2 Résultats

7.3 Analyse des résultats

- Quadratique :
  - 25 corps : Temps = 6.2376, Frames = 160.3346.
  - 50 corps :
  - 75 corps :
  - 100 corps :
- Barnes-Hut :

- 25 corps :
- 50 corps :
- 75 corps :
- 100 corps : 52.42836478582197

100 corps : Temps :56.58749961853027, Corps :100, frames par secondes :17.671747413143127 Temps :57.47449707984924, Corps :100, frames par secondes :17.399021319154848 Temps :54.411575794219 Corps :100, frames par secondes :18.37844218630823 Temps :50.94770860671997, Corps :100, frames par secondes :19.627968113723973 Temps :53.0928099155426, Corps :100, frames par secondes :18.83494208708769 Temps :52.0560986995697, Corps :100, frames par secondes :19.21004502798566 75 corps : Temps :32.22746157646179, Corps :75, frames par secondes :31.029437351974916 Temps :32.786499977111816, Corps :75, frames par secondes :30.50035840050319 Temps :34.51292324066162 Corps :75, frames par secondes :28.974653726863785 Temps :36.702881813049316, Corps :75, frames par secondes :27.245816965916305 Temps :31.772499561309814, Corps :75, frames par secondes :31.473759188204557 50 corps : Temps :17.765212059020996, Corps :50, frames par secondes :56.28978684170618 Temps :18.101902723312378, Corps :50, frames par secondes :55.24281150357518 Temps :19.539405584335327, Corps :50, frames par secondes :51.17862954857217 Temps :18.83494329452514 Corps :50, frames par secondes :53.09280651196202 Temps :17.829243183135986, Corps :50, frames par secondes :56.08763028965035 Temps :18.066161155700684, Corps :50, frames par secondes :55.35210227461384 25 corps : Temps :7.093503475189209, Corps :25, frames par secondes :140.9740621820625 Temps :6.24264669418335, Corps :25, frames par secondes :160.1884663650372 Temps :6.179133176803589, Corps :25, frames par secondes :161.83499714069785 Temps :6.853435039520264 Corps :25, frames par secondes :145.91223149172788 Temps :6.562495470046997, Corps :25, frames par secondes :152.38105756632828

8 Difficultés rencontrés

9 Conclusion

Nous avons pu réaliser les fonctionnalités principales et importantes du projet. Donc un moteur physique et graphique. Un algorithme. Ainsi que des fonctionnalités annexes comme la gestion de fichiers et de statistiques.

10 Manuel d'utilisation

10.1 Menu de configuration

Le menu de configuration permet à l'utilisateur de renseigner différentes caractéristiques pour le lancement de la simulation. L'utilisateur peut renseigner ces informations de 2 manières différentes :

- Par le biais des champs de saisies de texte : Il faut renseigner le nombres de corps voulus ainsi que les masses minimales et maximales définissant l'intervalles des masses des corps.
- Par l'import d'un fichier excel où sont contenus les informations : Un fichier excel vide avec le format compatible pour la simulation est mis à disposition par le biais d'un telechargement. Une fois ce fichier remplis il suffit de l'importer dans le logiciel et de lancer la simulation.

La première approche est plus rapide à mettre en place pour l'utilisateur mais utilise beaucoup d'aléatoire. La deuxième nécessite de configurer un fichier excel et donc un temps de préparation plus long mais permet d'avoir des données figées et exactes.

Il peut aussi définir quel algorithme sera utilisé en cochant les cases respectives.

10.2 Simulation

La simulation ne fait qu'afficher nos différents corps, aucune interactions n'est possible. Cependant elle donne accès à un éran de statistiques après un certain moment. Pour y accéder il suffit de cliquer sur la flèche en bas à droite de l'écran.

10.3 Notice

La notice indique à l'utilisateur le fonctionnement de l'application, notamment pour le menu de configuration.

10.4 Statistiques

L'écran de statistiques permet d'afficher différentes informations. A gauche un graphique qui regroupe nos differentes simulations et les algorithmes utilisés. A droite un graphique ne s'affichant qu'avec quelques conditions :

- Lancer avec un fichier et l'algorithme quadratique une première simulation.
- Lancer une deuxième simulation avec le même fichier mais cette fois-ci avec l'algorithme de Barnes-Hut.

Ces étapes réalisées un graphique dans la partie droite va s'afficher. Ce graphique montre la précision d'un algorithme par rapport à l'algorithme quadratique.
En dessous des graphiques est inscrit le nombre de corps et l'approximation.