Simulation De Systèmes De Particules

ZAKI Akram, BOURROUS Hani Anouar,

May 20, 2023

Simulation De Systèmes De Particules

Préambule

Dans ce compte rendu, nous allons explorer la simulation de systèmes de particules et répondre à plusieurs questions clés pour comprendre et analyser ce type de simulation. Nous allons examiner en détail les différentes classes utilisées dans ce projet, telles que la classe ***Univers*** qui représente l'environnement de la simulation et la classe ***Particle*** qui modélise les particules individuelles. Nous étudierons également les méthodes de simulation utilisées pour initialiser les particules, calculer les forces entre elles et mettre à jour leur position et leur vitesse.
En résumé, ce compte rendu fournira une vue d'ensemble détaillée des différentes classes utilisées, des méthodes de simulation employées, ainsi que des conclusions essentielles qui peuvent être tirées de ces simulations.

Lab 2 : Particules :

Mise en place des structures de données :

La classe ***Particle*** représente une particule dans un système physique avec des attributs tels que la position, la vitesse, la masse, l'identifiant, la catégorie et la force. Elle offre des méthodes pour calculer la distance entre les particules, calculer les forces exercées, mettre à jour la position et la vitesse, et gérer les forces.

• **Performances des collections de particules dans une simulation :**

Dans cette étude, nous avons évalué les performances de différentes collections (***vector***, ***deque***, ***list et forward list***) pour stocker un grand nombre de particules dans une simulation. Les tests ont été effectués en insérant 2000 particules pour différentes tailles de collection. Les résultats ont montré que les différences de performances deviennent significatives à partir d'une certaine taille de collection. Pour mesurer les performances, nous avons utilisé la bibliothèque chrono pour enregistrer les temps d'exécution. Cette analyse nous permet de choisir la collection la plus appropriée en fonction du nombre de particules.

Dans notre implémentation on a opté pour le choix de la classe ***vector*** pour certaines raisons à savoir Utilisation plus efficace de la mémoire et la manipulation simplifiée des éléments.

Méthode de Störmer-Verlet :

La classe ***Particle*** intègre une fonction ***StromerVerlet*** qui implémente l'algorithme de ***Störmer-Verlet***. Cette fonction permet l'évolution du système de particules en prenant en compte les forces d'interaction et en mettant à jour les positions des particules. Les résultats présentent les trajectoires des quatre astres (Terre, Jupiter, Halley et le Soleil) au cours du temps. Le graphe ci-dessous révèle le résultat final de la simulation:

Dans l'illustration ci-dessous, on peut observer les trajectoires de quatre astres : la Terre, Jupiter, Halley et le Soleil. Les trajectoires elliptiques des astres sont clairement discernables, avec le Soleil fixé en position centrale, reflétant ainsi de manière réaliste la configuration du système solaire. Il est intéressant de noter que les trajectoires dans cette figure respectent bien les lois de **Kepler**, mettant en évidence la cohérence des résultats obtenus par notre algorithme avec les trajectoires réelles des astres. En outre, il est évident que les orbites de Jupiter et de Halley présentent une excentricité plus marquée que celle de la Terre, ce qui signifie que ces planètes suivent des trajectoires plus allongées par rapport au Soleil.

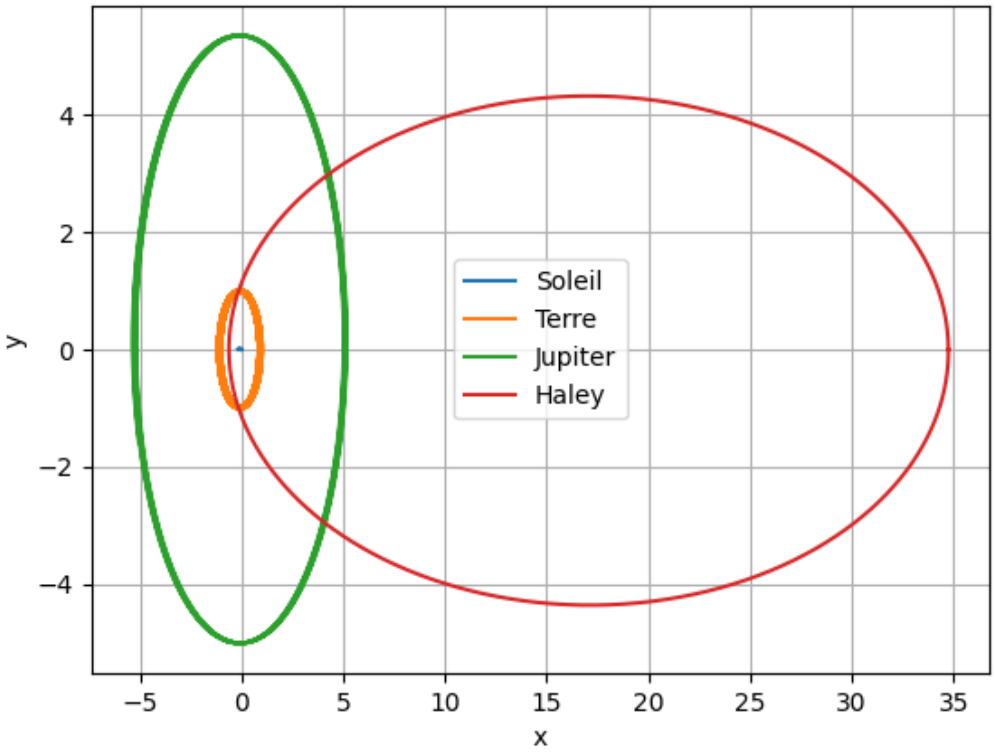

Figure 1: La trajectoire des quatres astres

Moralité

Dans ce Lab, nous étudions le comportement d'un système composé de plusieurs particules. Nous avons créé une classe appelée ***Particle*** qui représente un objet tridimensionnel caractérisé par sa position, sa vitesse, sa masse, son identifiant, sa catégorie et sa force. La simulation implique de stocker les particules dans une collection, et nous avons comparé différentes collections telles que ***vector***, ***deque***, ***list et forward list*** en termes de performances pour différents nombres de particules. Les forces entre les particules sont explicitement définies, et nous avons mis en œuvre l'algorithme de ***Störmer-Verlet*** pour calculer l'évolution du système de particules. De plus, nous avons réalisé une simulation d'un système gravitationnel composé du Soleil, de la Terre, de Jupiter et de la comète de Halley. Les positions des particules sont mises à jour à l'aide de l'algorithme, et les résultats sont enregistrés dans un fichier. La trajectoire des quatre corps célestes est ensuite analysée et visualisée.

Lab 3 : Utilisation des opérateurs :

Enrichissement des structures de données :

Création d'une classe vecteur et modification de la classe particule:

Dans le cadre du Lab 2, nous avons renforcé nos structures de données en introduisant un algorithme itératif pour déplacer des particules soumises à l'interaction gravitationnelle. Pour améliorer notre approche, nous avons créé une nouvelle classe appelée ***Vecteur***, capable de manipuler des vecteurs tridimensionnels.

Cette classe est autonome et comprend des constructeurs et des opérateurs essentiels, tels que l'addition, la soustraction, la multiplication par un scalaire, l'accès par indice [], le calcul de la norme et l'affichage via l'opérateur $<<$. Ensuite, nous avons modifié la classe ***Particle*** utilisée dans la séance précédente pour intégrer la nouvelle classe ***Vecteur***. Dans cette version, nous avons remplacé les structures de données précédentes par les attributs appropriés de la classe ***Vecteur*** pour représenter la vitesse, les forces et la position de la particule. De plus, nous avons intégré tous les calculs requis dans l'algorithme de Stromer-Verlet à la classe ***Particle***. Cette modification nous a permis de bénéficier des fonctionnalités de la classe ***Vecteur*** pour effectuer les opérations mathématiques nécessaires.

Univers des particules :

Nous avons développé une classe nommée ***Univers*** qui permet de créer et manipuler un univers de particules dans des dimensions spécifiées (1D, 2D ou 3D). Cette classe offre diverses fonctionnalités, notamment la capacité de faire progresser les particules, de calculer les forces d'interaction, de modifier les vitesses et d'afficher l'état de l'univers, à savoir les positions des particules à chaque instant. Parmi les méthodes implémentées, on trouve une méthode d'évolution appelée ***StormerVerlet*** qui fait avancer les particules dans le temps en utilisant l'algorithme de **Stromer-Verlet**. Cette méthode effectue des calculs de forces, met à jour les positions des particules, réinitialise les forces précédentes et met à jour les vitesses.

Complexité d'insertion :

Lors de l'évaluation des performances de notre implémentation, nous avons testé l'insertion des particules dans l'univers. La première approche d'insertion choisie devient coûteuse à mesure que la valeur de k augmente. Afin d'optimiser le temps de calcul des interactions, nous avons proposé une modification simple. En exploitant la symétrie de la force gravitationnelle, qui s'applique dans les deux sens entre deux particules, nous avons constaté qu'il n'est pas nécessaire de parcourir toutes les paires possibles de particules. Au lieu de cela, nous nous sommes limités aux paires distinctes (i, j) où $i < j$. Cette modification réduit considérablement la complexité temporelle, divisant ainsi par deux le temps de calcul requis.

Il y a d'autres pistes d'amélioration. En effe, dans le lab 4, nous avons rencontré des difficultés d'exécution liées à la complexité des calculs effectués. Pour améliorer les performances, plusieurs pistes d'amélioration peuvent être explorées. Une solution possible est l'utilisation de threads pour paralléliser les calculs et exploiter efficacement les ressources du système. Les threads permettent d'exécuter des parties du code en parallèle, ce qui peut réduire le temps d'exécution global.

Moralité

Dans le cadre du Lab 2, nous avons renforcé nos structures de données en introduisant un algorithme itératif pour déplacer des particules soumises à l'interaction gravitationnelle. Pour cela, nous avons créé une nouvelle classe Vecteur capable de manipuler des vecteurs tridimensionnels, et nous l'avons intégrée à la classe Particle. Nous avons également développé la classe Univers, qui permet de créer et manipuler un univers de particules dans des dimensions spécifiées. L'univers offre des fonctionnalités telles que le calcul des forces d'interaction, la progression des particules dans le temps grâce à l'algorithme de Stormer-Verlet, et l'affichage de l'état de l'univers. Lors de l'évaluation des performances, nous avons observé que l'approche d'insertion des particules devenait coûteuse avec une valeur de k élevée. Pour optimiser les calculs, nous avons proposé une modification en exploitant la symétrie de la force gravitationnelle, ce qui nous a permis de réduire considérablement la complexité temporelle.

Lab 4 : Découpage de l'espace :

Pour répondre à cette question : Représenter le potentiel de Lennard-Jones pour 1 système à deux particules en fonction de la distance r. Nous comparons les résultats obtenus par notre algorithme de simulation avec le résultat réel attendu, les deux figures ci-dessous illustrent la représentation du potentiel de Lennard-Jones pour un système à deux particules en fonction de la distance r.

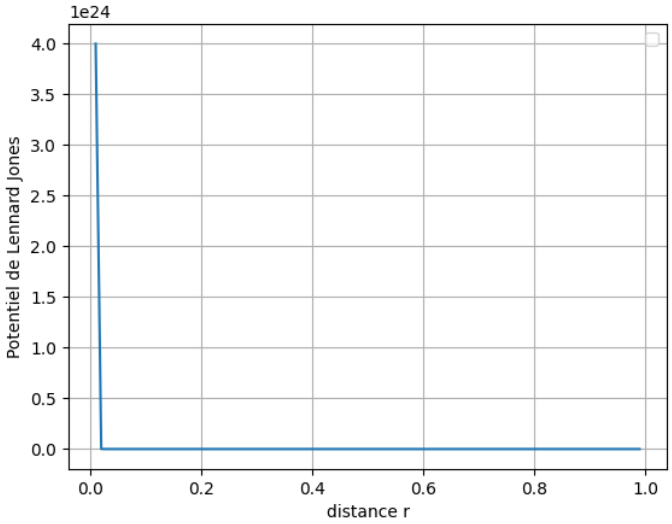

Figure 2: Potentiel de Lennard-Jones

Création d'un Univers :

Nous avons créé la classe ***Univers*** pour faciliter la gestion des particules dans l'espace en utilisant une approche de maillage. Pour cela, nous avons introduit une grille tensorielle de cubes dans laquelle les particules sont regroupées. Le nombre de cubes dans chaque direction de la grille dépend de la longueur caractéristique de la direction et de la taille de la cellule. Dans la classe ***Univers***, nous avons ajouté une liste de cellules qui contiennent les particules et qui ont des cellules voisines (3 en 1D, 9 en 2D et 27 en 3D). De plus, nous avons représenté cette grille en considérant que toutes les particules sont contiguës et en les stockant dans un

vecteur. Cette représentation nous permet de gérer efficacement les interactions entre les particules et d'accéder rapidement aux cellules voisines lors des calculs. De plus, nous avons utilisé un ***set*** pour modéliser chaque cellule, car nous effectuons de nombreuses insertions et suppressions de particules. Cependant, en raison du temps d'exécution élevé, nous avons été limités dans notre capacité à créer une image plus évoluée en temps.

Calcul des potentiels :

Nous avons modifié notre algorithme et son implémentation afin de prendre en compte le maillage construit pour limiter le parcours des particules lors du calcul des potentiels. Au lieu de calculer la distance entre deux particules, nous calculons la distance entre une particule et le centre d'une cellule. Si la cellule se trouve dans le voisinage, nous examinons les particules de cette cellule et calculons le potentiel si le rayon de coupure est respecté. Sinon, nous ignorons la cellule. Cette approche nécessite de positionner correctement les particules dans les cellules, ce qui est effectué à la fin de chaque itération temporelle en mettant à jour leur position logique. Afin de vérifier les résultats, nous conservons la force gravitationnelle des sessions précédentes et nous nous assurons que les orbites sont correctes.

Application : collision de deux objets :

Les graphes ci-dessous illustrent les différentes étapes de simulation de la collision entre deux objets, permettant ainsi de tester l'efficacité de notre algorithme. Les paramètres utilisés sont les suivants :
σ =1, L_1 = 250, L_2 = 40, ϵ = 5, m= 1, v = (0, 10), N_1 = 1600, N_2 = 6400, r_{cut} = 2.5σ, et δt = 0.00005.
Le carré contient 40 × 40 particules équidistribuées, tandis que le rectangle contient 160 × 40 particules équidistribuées, avec une distance entre les particules de $\frac{2^{\frac{1}{6}}}{\sigma}$. Nous avons réalisé la simulation jusqu'à t = 19.5, et les résultats affichent l'état initial des particules ainsi que l'état final après la collision. Cela nous permet d'évaluer l'efficacité de notre algorithme et d'étudier la collision entre les deux objets.

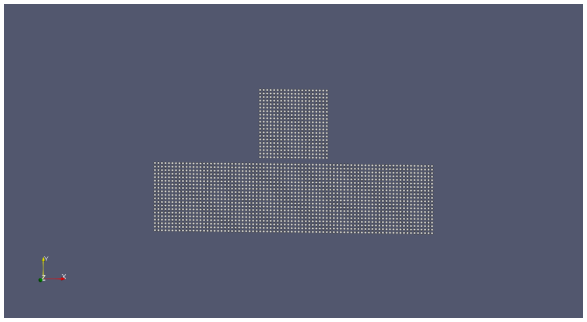

Figure 3: Avant la simulation

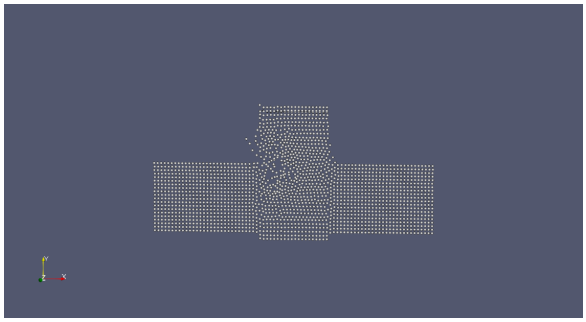

Figure 4: En cours de la simulation

Il faut noter que le temps d'exécution élevé a été un obstacle majeur qui a limité la possibilité de créer une image plus évoluée en temps.

Moralité

Pour répondre à la question de la représentation du potentiel de Lennard-Jones pour un système à deux particules en fonction de la distance, nous avons comparé les résultats obtenus par notre algorithme de simulation avec le résultat réel attendu. Dans le cadre de cette simulation, nous avons créé la classe Univers pour faciliter la gestion des particules dans l'espace en utilisant une approche de maillage. Cette classe comprend une grille tensorielle de cubes où les particules sont regroupées, permettant une gestion efficace des interactions et un accès rapide aux cellules voisines lors des calculs. De plus, nous avons modifié notre algorithme et son implémentation pour prendre en compte ce maillage, limitant ainsi le parcours des particules lors du calcul des potentiels. Nous avons également réalisé la simulation de la collision entre deux objets en utilisant les paramètres spécifiés, et les résultats affichent l'état initial des particules ainsi que l'état final après la collision. Cette approche nous permet d'évaluer l'efficacité de notre algorithme et d'étudier la collision entre les deux objets.

Lab 5 : Test et Visualisation :

Mise en place de tests :

Nous avons utilisé Google Test pour mettre en place une infrastructure de tests unitaires pour les différentes classes de notre projet, y compris les classes Vecteur, Particule, Univers et Cellule. Ces tests ont été conçus pour valider notre implémentation en vérifiant le comportement attendu des différentes méthodes et fonctionnalités. Pour les tests fonctionnels, nous avons utilisé les deux simulations des laboratoires précédents, à savoir le système solaire et la collision entre deux objets. Pour valider les résultats, nous avons visualisé l'évolution du système au fil du temps à l'aide de Python ou Paraview. Cette approche nous a permis de vérifier la conformité de notre implémentation et d'assurer la fiabilité de notre projet. En suivant les critères recommandés, nos tests sont indépendants, bien organisés, portables, fournissent des informations détaillées en cas d'échec et s'exécutent rapidement.

Visualisation :

Pour visualiser les résultats de notre projet, nous avons intégré une méthode appelée printVtk dans la classe ***Univers*** qui permet de sauvegarder les données des particules dans un fichier au format VTK. Cette fonction prend en paramètre un vecteur de particules et un flux de sortie ***ostream*** où les données seront enregistrées.

Le fichier généré contient les informations nécessaires pour la visualisation, telles que le nombre de points ***particules*** dans le système, les coordonnées de chaque particule, la vitesse de chaque particule et la masse de chaque particule. Les données sont stockées de manière continue dans les champs correspondants du fichier VTK.

Pour utiliser cette fonction, il suffit d'appeler ***printVtk*** en lui passant le vecteur de particules et le flux de sortie approprié.

Une fois le fichier VTK généré, il peut être ouvert et visualisé à l'aide de logiciels de visualisation tels que Paraview. Cela permet de visualiser l'état de l'univers à l'instant où la fonction ***printVtk*** a été appelée, en affichant les particules avec leurs positions, vitesses et masses correspondantes.

En intégrant cette méthode dans notre projet, nous avons fourni une solution pour enregistrer les données des particules et faciliter leur visualisation ultérieure.

ACVL :

Dans cette section, nous représenterons les différents diagrammes de l'analyse a posteriori des développements réalisés. À savoir les diagrammes des cas d'utilisations, le diagramme de séquence, le diagramme de transitions et le diagramme de classes d'analyse.

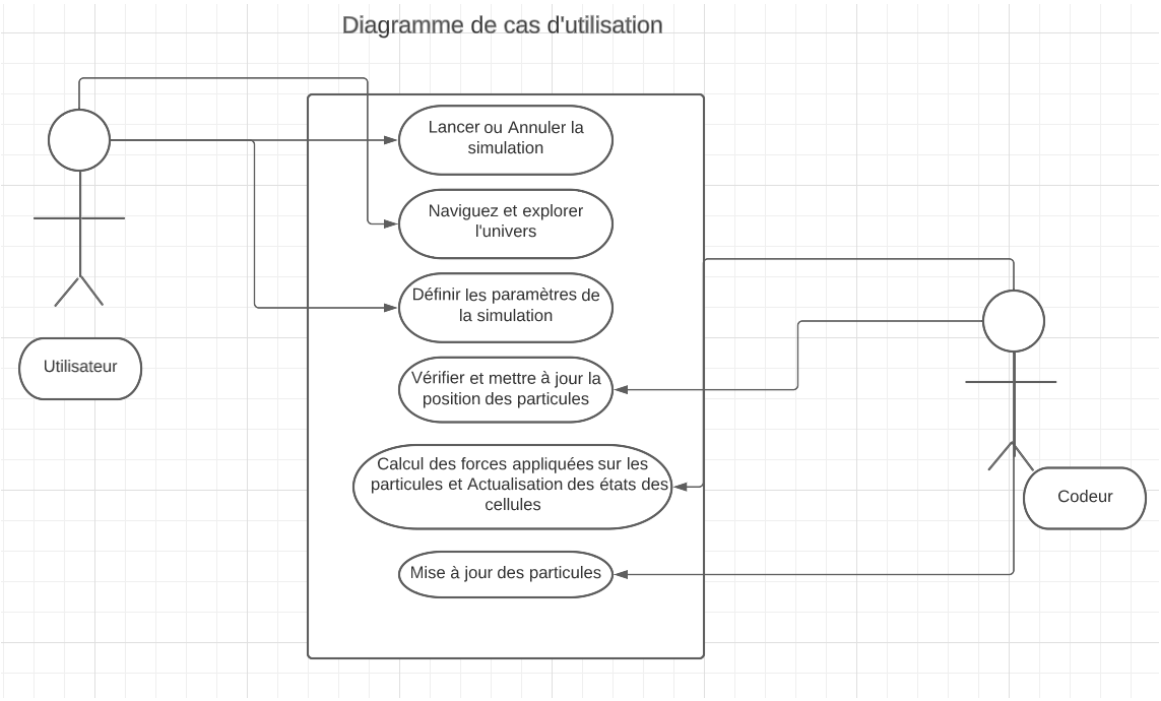

Figure 5: Diagramme de cas d'utilisation

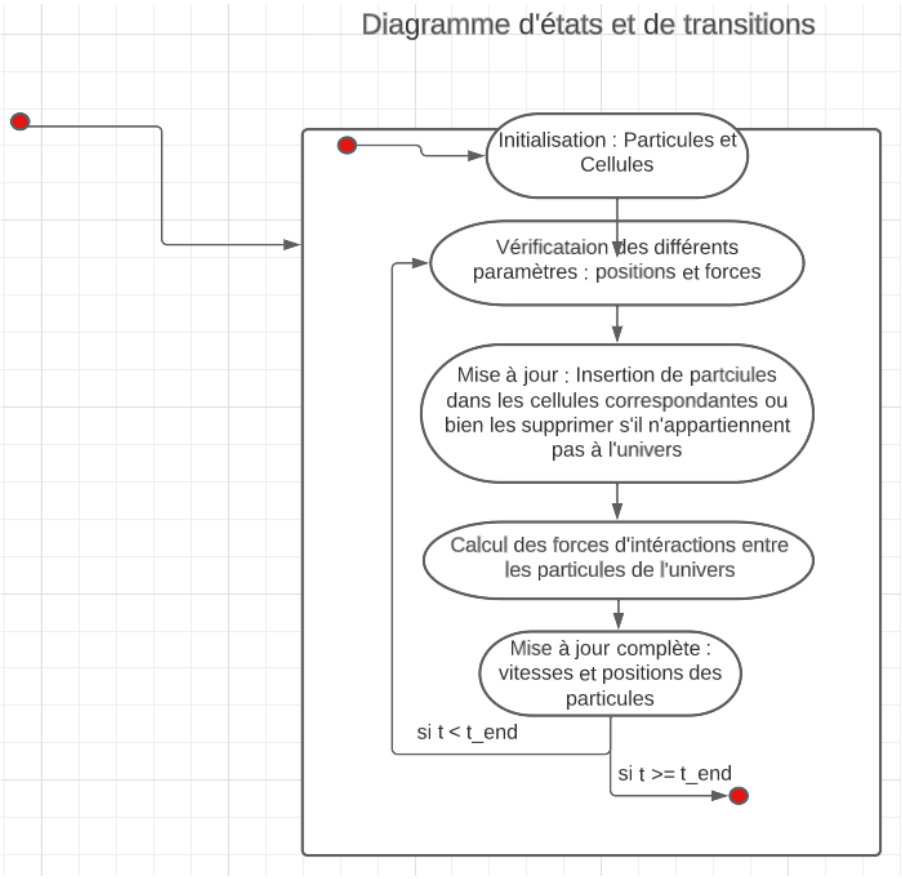

Figure 6: Diagramme d'états et de transitions

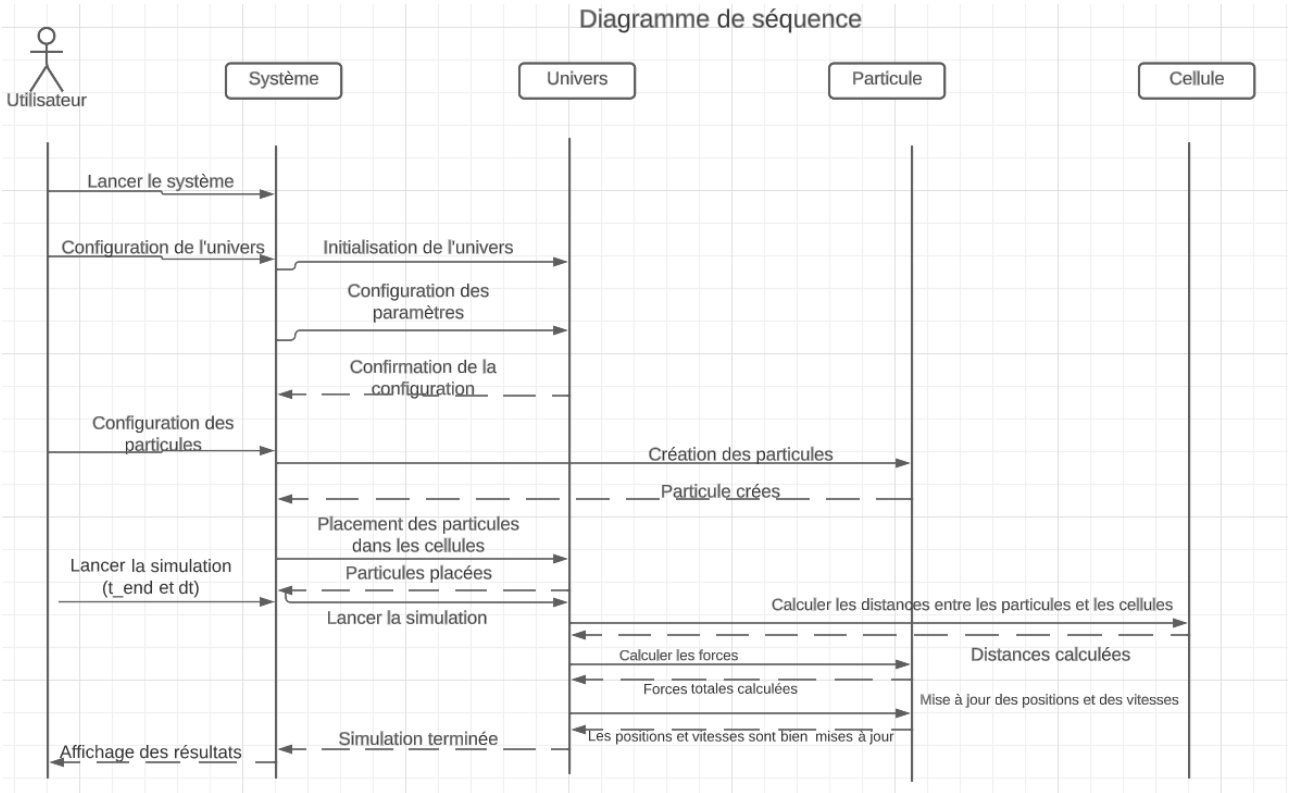

Figure 7: Diagramme de séquence

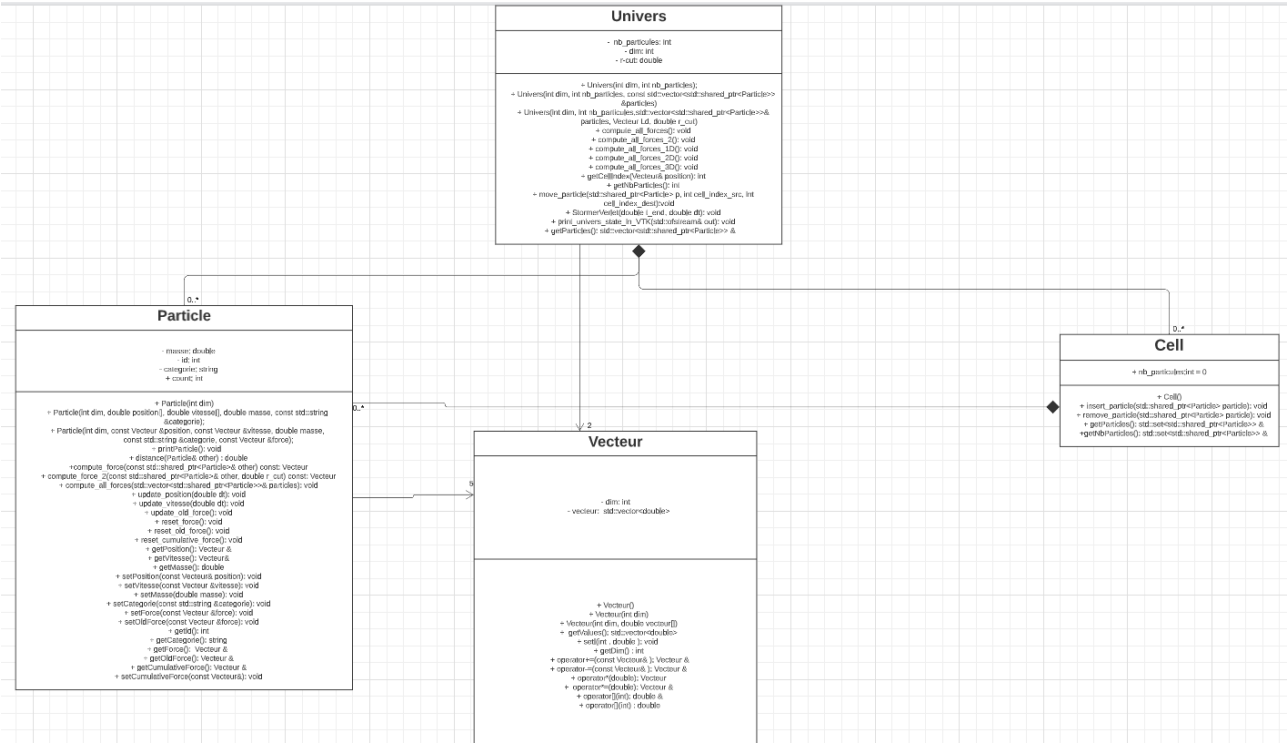

Figure 8: Diagramme de classe

Moralité

Dans ce Lab5, nous avons utilisé Google Test pour mettre en place une infrastructure de tests unitaires pour les différentes classes, tels que Vecteur, Particule, Univers et Cellule. Ces tests ont été conçus pour valider l'implémentation en vérifiant le comportement attendu des méthodes et fonctionnalités. Nous avons également effectué des tests fonctionnels en utilisant les simulations du système solaire et de la collision entre deux objets des laboratoires précédents. Pour la visualisation des résultats, nous avons intégré une méthode appelée printVtk dans la classe Univers qui permet de sauvegarder les données des particules dans un fichier au format VTK. Cette fonction facilite la visualisation en enregistrant les coordonnées, vitesses et masses des particules. De plus, nous avons créé des diagrammes de l'analyse a posteriori des développements réalisés pour représenter les cas d'utilisation, les séquences, les transitions et les classes d'analyse. En suivant ces approches, nous avons pu valider notre implémentation, assurer la fiabilité du projet et faciliter la visualisation des résultats.

Lab 6 : Raffinement du modèle :

Conditions aux limites :

Pour la réflexion, nous utilisons soit la vitesse soit le potentiel pour déterminer le nouveau mouvement des particules lorsqu'elles rencontrent la paroi. Ainsi, elles continuent leur trajectoire après la réflexion.

Pour l'absorption, les particules disparaissent simplement lorsqu'elles entrent en contact avec la paroi. Elles ne sont pas réfléchies ou réinsérées dans le domaine.

Quant aux conditions périodiques, les particules se réinsèrent de l'autre côté du domaine lorsqu'elles atteignent une frontière. Cela leur permet de continuer leur déplacement sans rencontrer de paroi physique.

Ces fonctionnalités sont implémentées dans les fonctions ***handlereflexion*** et ***handleperiodicmvt***.

Conclusion

En conclusion de ce projet de simulation de systèmes de particules, nous avons acquis de précieuses connaissances en implémentant les différents algorithmes en langage C++. Nous avons pu approfondir notre compréhension des concepts clés tels que la modélisation des particules, les interactions entre elles, la gestion des conditions aux limites et la visualisation des résultats. Cette expérience nous a également permis de renforcer nos compétences en programmation orientée objet, en résolution de problèmes et en collaboration au sein de l'équipe.

Nous souhaitons exprimer nos sincères remerciements au professeur Mr **Christophe Picard** pour son encadrement tout au long du projet. Ses conseils, son expertise et sa disponibilité ont été d'une grande aide dans notre apprentissage et notre avancement.

Enfin ce projet fut plaisant et de grande utilité.