Dynamique des Corps Rigides et Interaction

Merwan Achibet

Contents

1 Introduction

1.1 Les moteurs physiques

Invisibles, les phénomènes physiques qui régissent le fonctionnement de notre univers sont pourtant omniprésents et universels. Étudiées depuis des siècles, les lois les décrivant ont été à maintes reprises redéfinies et affinées et il est de nos jours indispensable de pouvoir les modéliser de façon fidèle ou tout du moins d'être capable de les approximer de façon plausible.

Une simulation industrielle visant par exemple à reproduire virtuellement les interactions entre les différentes pièces qui composent une automobile doit être capable de reproduire de façon réaliste la friction des pneumatiques sur le sol, l'influence de la gravité sur le véhicule, le comportement thermique et volumique des fluides qu'il contient ainsi que de nombreux autres aspects mécaniques de son fonctionnement. Le système informatique simulant tous ces facteurs est appellé moteur physique. Dans cet exemple, les enjeux de sécurité et de qualité sont grands et des résultats d'une précision extrême sont exigés. Les calculs à mettre en jeu pour les obtenir peuvent donc se permettre d'être très coûteux et de se baser sur des modélisations mathématiques complexes. Il n'est pas rare que ces processus de simulation soient répartis sur plusieurs machines et s'étalent sur une durée de calcul de plusieurs heures.

A contrario, les moteurs physiques d'autres types de système complexe ne peuvent se permettre une telle latence et doivent fonctionner en temps réel. La contrainte est encore plus forte lorsque le moteur physique doit cohabiter avec d'autres modules gérant différents aspects de l'application. On pense notamment aux jeux vidéo, qui partagent le temps de calcul alloué entre le moteur graphique, le moteur d'intelligence artificielle, la gestion du son, la gestion du réseau, et bien sûr le moteur physique. Dans ce cadre, auquel on peut gr-

effer la réalité virtuelle et ses applications dérivées, la contrainte la plus importante est le temps d'éxecution et non la précision des résultats. On ne cherche plus à obtenir des données exactes mais une représentation plausible du réel. Ainsi, certains raccourcis pourront être tolérés et une part de réalisme est sacrifiée au profit de la vitesse.

La physique est un champ très vaste dont les disciplines vont de l'acoustique à l'électronique. Néanmoins, le spectre d'action des moteurs physiques se limite à la mécanique classique, dite mécanique newtonienne. Ce sous-domaine répond à des questions telles que : Comment réagira cette balle si on la lance sur un mur ? Quelle est l'influence d'une planète sur un objet spatial donné ? Pourquoi un liquide visqueux dispose-t-il d'une faible vitesse d'écoulement ? Quelles déformations engendrera un choc entre deux voitures ?

La mécanique newtonienne est elle-même une vaste discipline et la précision d'une modélisation ne suffit pas à la différenciation entre tous les moteurs physiques. Souvent, un moteur physique sera spécialisé. Certains simuleront les interactions entre corps rigides comme une boîte en plastique tombant sur le sol. Certains simuleront le comportement d'objets déformables comme deux véhicules se percutant. D'autres se concentreront sur les réactions entre liquides ou entre gaz.

1.2 Travail à accomplir

L'objectif de ce projet est de concevoir un moteur physique de base permettant de gérer les interactions entre des corps rigides et convexes. On parle de corps rigides dans la mesure où les objets mis en jeu seront indéformables et incassables. Dans un tel contexte, une tasse en porcelaine surmontée d'un bloc de granite d'une tonne ne serait aucunement endommagée. On parle de corps convexes car se limiter dans un premier temps à ce type de structure autorise certaines facilités dans les calculs. Une piste pour étendre la simulation aux solides concaves sera présentée dans la dernière partie.

La contrainte principale de notre application sera le temps. On veut concevoir une simulation éxecutable en temps réel de telle façon que si l'utilisateur modifie l'environnement de la simulation à un instant quelconque, une réaction à cette interaction soit immédiatement perceptible. Certains raccourcis dans les calculs ainsi que plusieurs approximations seront acceptés. Le fonctionnement du moteur se base néanmoins sur des lois bien connues de la mécanique de Newton et ses résultats ne devront pas s'éloigner de façon demesurée de ceux que l'on retrouverait dans une situation réelle.

1.3 Étude de cas

L'activité physique d'un corps se divise en plusieurs phases. Afin de les détailler, analysons une situation concrète. Si l'on tenait une balle dans notre main et que nous la lâchions au dessus d'un plan, quelles seraient les étapes que cet objet traverserait avant d'arriver à un état de repos ?

1.3.1 La chute

La main s'ouvre et laisse s'échapper la balle. Notre appréhension du monde qui nous entoure nous permet de prévoir que l'objet tombera et accélérera vers le bas. Ce phénomène est quantifié par la seconde loi de Newton.

$$\vec{a} = \frac{1}{m} \sum \vec{F_i}$$

Cette règle décrit l'accélération $\vec{a}$ d'un corps comme étant le produit de l'inverse de sa masse m et de la somme des forces $\vec{F_i}$ qui lui sont appliquées. Dans notre exemple, on lâche la balle sans lui donner d'élan initial et la seule influence qu'elle subit au cours de sa chute est celle de la gravité. La gravité terrestre est une force de $9.81N$ dirigée vers le noyau de la planète mais dans la simulation, on peut la réduire à une force dirigée vers le "bas" (dans la direction négative de l'axe y). Afin de bénéficier d'un moteur physique versatile, la puissance de la gravité pourra être modifiée, pour simuler une situation lunaire par exemple, ou être totalement annulée, pour simuler une situation d'apesanteur. En réalité, la gravité ne sera même pas codée "en dur" mais appartiendra à une liste de forces environnementales qui contiendra toutes les influences que le monde de la simulation fait subir aux corps qu'il contient. On pourra

par exemple modéliser, entre autres, la résistance de l'air ou le roulis irrégulier du vent.

Cette observation du comportement de la balle nous oriente sur la façon dont l'on pourra modéliser les déplacements d'objets soumis à des forces extérieures. À chaque corps seront associées des quantités physiques telles que la position, la vitesse et l'accélération. Le rôle principal du moteur physique sera de faire évoluer ces variables de façon à ce que le résultat d'une simulation s'approche le plus possible de ce qui serait observable dans le monde réel. Dans la première partie de ce compte-rendu, on précisera donc les méthodes employées pour simuler la dynamique des corps.

1.3.2 Le rebond

Alors que la balle s'approche du plan, on s'attend naturellement à ce qu'elle entre en contact avec ce dernier et qu'une réaction proportionnelle à la puissance du choc soit produite. Cette réaction dépend de nombreux facteurs, notamment des masses des objets et de leur vitesse respective.

Le moteur physique devra être capable de générer une réaction réaliste dont la détermination passe par une formule présentée dans la seconde partie. Néanmoins, le travail le plus complexe n'est pas de calculer une réponse mais de détecter une collision. Plusieurs processus géométriques devront être mis en place afin de vérifier si une collision a lieu et si tel est le cas, afin de mesurer précisément quels points des deux corps entrent en contact.

Une difficulté supplémentaire vient du fait que la simulation est mise à jour de façon discrète, par pas de temps fixe. Lorsqu'une collision sera détectée entre deux corps, il est presque impossible de se retrouver dans une situation de contact parfait. On aura plutôt des contacts pénétrants au sein desquels l'intégrité physique des corps est corrompue et les objets rentrent l'un dans l'autre. Une des tâches du moteur physique sera de pallier ce problème en recalant les corps dans la position de contact parfait qu'ils auraient dû atteindre.

Cette phase de la vie d'un corps rigide est la plus courte, puisqu'instantanée, mais demandera paradoxalement le plus de travail. Les considérations géométriques qui entrent en jeu, ainsi que les limites à contourner, imposées par l'arithmétique des nombres à virgule flottante, seront détaillées dans la seconde section.

1.3.3 Le repos

Les rebonds sur le plan sont de plus en plus faibles, jusqu'à ce que la balle n'ait plus assez d'énergie cinétique pour s'élever à nouveau.

Son état de repos laisse penser que plus aucune force ne s'applique sur elle. Contrairement aux apparences, dans le monde réel ce n'est pas parce qu'un corps est fixe qu'il est libre de toute influence. La gravité n'a pas disparu par magie et pourtant l'accélération de la balle est nulle, puisqu'elle reste immobile. La troisième loi de Newton est là pour démystifier cette situation :

> *Tout corps A exerçant une force sur un corps B subit une force d'intensité égale, de même direction mais de sens opposé, exercée par le corps B.*

Autrement dit :

$$\vec{F}_{A/B} = -\vec{F}_{B/A}$$

$$\vec{F}_{A/B} + \vec{F}_{B/A} = 0$$

Ici, la balle subit toujours la gravité mais le plan produit une force inverse qui permet d'annuler tout mouvement. Même si un spectateur aura l'impression que la balle est inactive, son état collisionel constant lui permet de rester à la surface du plan et d'équilibrer le système. Ce phénomène sera reproductible dans le moteur physique et sera en réalité basé sur la même méthode que pour les collisions classiques.

Puisque l'une des préoccupations majeurs de ce projet est la vitesse d'exécution, d'autres considérations plus techniques entreront elles aussi en jeu. Une fois qu'un objet est immobile, par exemple, peut-on le fixer artificiellement et ne plus faire évoluer son état pour économiser des ressources ? Si tel est le cas, quand devra-t-on le réactiver ?

2 Dynamique

2.1 La composante linéaire

2.1.1 Variables d'état

Commençons par nous concentrer sur l'aspect cinématique d'un corps rigide, c'est à dire son mouvement lorsqu'il n'est soumis à aucune force extérieure. Dans un premier temps, seule la composante linéaire du mouvement sera étudiée et les objets dont nous allons simuler le comportement seront réduits à de simples particules. Les figures présentées tout au long de ce rapport sont en deux dimensions pour des raisons de clarté mais le principe reste similaire lorsqu'étendu à la troisième dimension.

La quantité physique la plus perceptible visuellement pour un spectateur est la position $\vec{p}$ d'un corps. Pour mettre en place notre affichage final, c'est cette valeur à tout instant t de la simulation que l'on veut déterminer. Un corps possède aussi une vitesse $\vec{v}$, qui correspond à la variation de sa position pour une unité de temps. On note cette relation sous la forme dérivée :

$$\vec{v} = \frac{\partial \vec{p}}{\partial t}$$

Pareillement, l'accéleration $\vec{a}$, qui apparaissaît plus tôt dans la seconde loi de Newton, correspond à la variation de la vitesse par rapport à une unité de temps.

$$\vec{a} = \frac{\partial \vec{v}}{\partial t}$$

Par transitivité, on confirme que la position est den relation directe avec l'accélération.

$$\vec{a} = \frac{\partial \vec{v}}{\partial t} = \frac{\partial^2 \vec{p}}{\partial t}$$

Le travail de la partie dynamique du moteur physique revient à déterminer la nouvelle position d'un objet à partir de la connaissance de ses autres variables d'état. Grâce à l'équation précédente, on peut calculer l'accélération d'un objet à partir de sa variation de position. Dans le moteur physique, ce sera en fait l'opération inverse qui devra être effectuée : on connaîtra les forces appliquées, on en déduira par la seconde loi de Newton l'accélération induite puis on calculera le changement de position. Il faut donc exploiter le fait que ces trois quantités partagent aussi des relations de primitive.

$$\vec{p} = \int \vec{v}\, \partial t = \int \vec{a}\, \partial t^2$$

Récapitulons. Chacun des corps rigides dont l'on veut simuler l'évolution possèdera trois quantités sous forme vectorielle : position $\vec{p}$, vitesse $\vec{v}$ et accélération $\vec{a}$. L'une des tâches de base du moteur physique sera de traduire l'application de forces sur un corps en un changement de sa position. On sait que les quantités énoncées entretiennent des relations de dérivation, il faudra donc procéder par intégration pour calculer la nouvelle position d'un objet à partir de son accélération.

2.1.2 Intégration

Maintenant que les trois quantités physiques entrant en jeu dans les mouvements linéaires sont présentées, nous pouvons étudier de façon plus concrète sur quels calculs se basera l'exercice le plus élémentaire du moteur physique.

Les phénomènes mécaniques du monde réel évoluent de façon continue mais notre simulation ne peut pas s'autoriser ce luxe. Le moteur physique sera donc basé sur une simulation discrète et avancera par pas de temps fixe ∂t. À chaque mise à jour de la simulation, des intégrations doivent être réalisées pour déterminer le changement d'état d'un corps d'un instant t_n à un instant $t_{n+1} = t_n + \partial t$. Toujours pour des raisons d'efficacité, nous ne pouvons pas nous permettre d'allouer un temps de calcul trop important à cette phase de la mise à jour et nous devons trouver un moyen d'approximer ces intégrales.

Parmi les techniques classiques d'intégration approximative, on trouve l'intégration d'Euler. Cette méthode part du principe que l'on dispose de la valeur initiale x_0 de la quantité que l'on souhaite faire évoluer ainsi que de son taux de changement x' et de la variation de temps ∂t par rapport à l'état précédent.

$$x_{n+1} = x_n + x'\partial t$$

Si l'on adapte cette méthode à notre problème, on obtient la succession de calculs suivante :

$$\vec{a}_{n+1} = \frac{1}{m}\sum \vec{F}_i$$

$$\vec{v}_{n+1} = \vec{v}_n + \vec{a}_{n+1}\partial t$$

$$\vec{p}_{n+1} = \vec{p}_n + \vec{v}_{n+1}\partial t$$

On calcule l'accélération à un instant t puis on intègre en fonction du temps jusqu'à obtenir la nouvelle position du corps. Ce processus doit être repété à chaque mise à jour du système et ce, pour chaque corps.

On aurait pu choisir une autre méthode d'intégration, telle que l'intégration Runge-Kutta d'ordre 4 qui calcule les pentes des sous-segments d'un pas de temps pour obtenir des résultats plus précis [3], ou bien telle que l'intégration de Verlet qui dispose d'une meilleure stabilité [1], mais Euler reste le choix le plus économique et fournit des résultats relativement satisfaisants.

Afin de réduire la complexité de la structure informatique qui représentera un corps rigide, nous allons introduire la notion d'élan linéaire. Pour un corps rigide, l'élan linéaire $\vec{L}$ est le produit de la masse et de la vitesse. Cette nouvelle quantité a pour avantage majeur de posséder comme primitive la variation de force exercée sur le corps.

$$\sum \vec{F}_i = \frac{\partial \vec{L}}{\partial t} = \frac{\partial (m\,\vec{v})}{\partial t}$$

Ce qui signifie que l'on peut réduire l'intégration de l'état d'un corps à :

$$\vec{L}_{n+1} = \vec{L}_n + \sum \vec{F}_i$$

$$\vec{p}_{n+1} = \vec{p}_n + \frac{\vec{L}_{n+1}}{m}\partial t$$

L'élan linéaire remplace l'accélération dans la définition d'un corps et permet de calculer sa vitesse si besoin en est. Cette organisation permet à première vue une intégration plus courte et une occupation moindre de la mémoire mais les avantages majeurs de ce choix prendront tous leurs sens lorsque le mouvement angulaire sera introduit.

2.1.3 Modélisation d'un corps

Nous avons décrit dans la partie précédente les quantités régissant le mouvement linéaire d'une particule ainsi que la façon dont elles évoluent mais le moteur physique que l'on conçoit a pour visée de simuler les interactions entre des corps rigides à volume convexe. Comment peut-on étendre les principes énoncés pour des particules à ce modèle plus complexe ?

On pourrait en premier lieu penser à représenter un tel corps par une liste de particules, chacune placée à un sommet de l'objet. Chaque particule évoluerait indépendement et des contraintes de cohésion entre particules voisines seraient appliquées pour empêcher toute déformation du corps. Cette méthode est envisageable mais elle présente plusieurs désavantages. Premièrement, les règles de cohésion à mettre en place nécessiteraient des traitements supplémentaires, et donc un temps de calcul plus élevé. Deuxièment, un corps devrait passer par autant d'intégrations qu'il a de points à chaque mise à jour. Il existe une solution plus simple et plus élégante qui permet de réduire les mouvements linéaires d'un corps rigide à ceux d'une unique particule judicieusement placée.

Introduisons en premier lieu la notion de repères absolu et local. Le repère absolu est le référentiel orthogonal dont l'origine sert de centre à l'environnement de

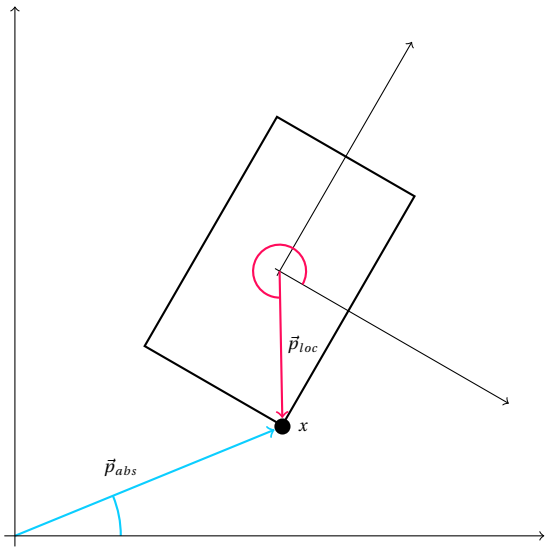

Figure 1: Les repères absolu et local d'un rectangle

la simulation. Un repère local est un référentiel qui est unique à chaque corps et dont le l'origine se situe à l'intérieur même de cet objet. La figure 1 nous montre la position q'un pointt x peut prendre par rapport aux deux repères.

La position exacte de l'origine du repère local dépend de la position des sommets qui forment l'objet mais il ne s'agit pas d'un simple centre géométrique puisque la masse de chaque sommet est aussi un facteur déterminant. Cette position se nomme le centre de masse, ou barycentre, et sera calculée par la formule suivante, avec M la masse totale des points du corps, m_i et $\vec{p}_i$ respectivement la masse et la position du sommet i dans le repère absolu et $\vec{C}$ la position du centre de masse dans le repère absolu.

$$\vec{C} = \frac{1}{M} \sum m_i \vec{p}_i$$

Une fois le centre de masse défini, la position locale $\vec{r}_i$ d'une particule i peut être calculée en fonction de sa position absolue $\vec{p}_i$ par :

$$\vec{r}_i = \vec{p}_i - \vec{C}$$

Grâce au centre de masse, on a une seule position à faire évoluer, quelle que soit la complexité de la structure du corps concerné.

2.2 La composante angulaire

2.2.1 Variables d'état

Le modèle que nous avons défini est encore incomplet puisqu'il ne prend pas en compte la composante rotationnelle des mouvements dont l'on peut être témoin dans un environnement réel. Les particules étant de simples points flottants dans l'espace, cela ne posait pas de problème précédemment mais le moteur physique que l'on conçoit doit gérer des volumes plus complexes. Imaginons une boîte cubique que l'on lancerait en l'air, si aucune rotation n'apparaît (si la base de la boîte reste parallèle au sol), l'imitation du réel que l'on souhaite reproduire perd toute crédibilité.

Les quantités physiques entrant en jeu dans la décomposition d'un déplacement angulaire sont analogues à celles présentées dans la partie traitant de la dynamique linéaire : à la position et à l'élan linéaire correspondent l'orientation et l'élan angulaire.

De la même façon que la position représentait visuellement l'état d'un corps au sein de la composante linéaire, un corps doit posséder une orientation. En deux dimensions, un nombre flottant représentant l'angle du corps par rapport à un axe fixe suffirait à décrire l'orientation d'un objet mais pas dans notre environnement en trois dimensions. Le repère local d'un corps a été introduit dans la partie précédente et se résumait à un centre de masse faisant office d'origine, mais un repère possède aussi des axes et ceux du repère local ne sont pas nécessairement alignés avec ceux du repère absolu. Pour représenter la direction des axes du repère local, et donc l'orientation du corps à qui il appartient, on utilise une matrice de dimension 3 dans laquelle chaque vecteur colonne correspondra à un des axes du repère local [6]. Pour illustrer ces propos, analysons la matrice identité d'ordre 3, qui correspond à la matrice d'orientation d'un corps parfaitement aligné avec les axes du repère absolu et n'ayant encore subit aucune rotation.

Figure 2

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \rightarrow \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

Si l'on isole les vecteurs colonnes de cette matrice identité, on remarque que chacun correspond à un des axes du repère absolu. Le premier vecteur colonne d'une matrice d'orientation correspondant à l'axe x du repère local. Il en est de même pour la seconde colonne avec l'axe y et la troisième colonne avec l'axe z.

La matrice d'orientation contient les directions des axes du repère local, or on calcule la position des sommets d'un corps grâce à la position de son centre de masse et à son orientation. Il est donc primordial d'attacher un soin particulier à la validité de cette matrice si l'on veut éviter toute déformation du solide. L'arithmétique des nombres à virgule flottante impose ici son premier effet négatif. En effet, à chaque intégration de l'état d'un corps, de légères erreurs de calcul apparaissent, principalement sur la matrice d'orientation. L'effet est infime mais ces erreurs s'accumulent à chaque itération, jusqu'à devenir visuellement perceptibles. Le problème vient du fait qu'avec chaque mise à jour de la simulation, les axes du repère local (donc les vecteurs colonne de la matrice) perdent progressivement de leur normalité et de leur orthogonalité. En conséquence, l'orientation n'est pas aussi réaliste qu'on le souhaiterait, pire, l'objet est gravement déformé. Cette situation est visible sur la figure 2.

IMAGE 2

Afin de contrer cet effet indésirable, on ajoutera deux étapes de corrections à la fin de chaque intégration. Premièrement, il nous faut normaliser les axes du repère local. Chaque axe correspondand à un vecteur colonne de la matrice d'orientation, il faut donc les extraire un par un et recalculer leur magnitude pour s'assurer de leur normalité.

En second lieu, il sera nécessaire de réorthogonaliser le repère local, autrement dit s'assurer que ses axes restent orthognaux entre eux. Pour cela, on orthogonalisera la matrice de rotation par le processus de Gram-Schmidt [4], une méthode d'orthogonalisation fonctionnant par projection itérative des axes les uns sur les autres.

Maintenant que l'orientation d'un corps a été définie, penchons nous sur l'élan angulaire. On pourrait utiliser vitesse et accélération angulaires en tant que variables d'état, mais comme pour la dynamique linéaire on choisit de remplacer ces deux valeurs par un unique élan angulaire. L'élan angulaire $\vec{A}$ possède comme primitive la variation de force exercée sur un corps. Mais attention, sa définition diffère de l'élan linéaire dans la mesure où il n'est en relation qu'avec la composante angulaire d'une force. En effet, il est primordial de faire la distinction entre l'influence linéaire et l'influence angulaire qu'une force exerce sur un corps. Quel que soit le point d'un objet sur lequel une force est exercée, la quantité d'élan linéaire ajoutée est la même. Par contre, la quantité d'élan angulaire transmis par une force dépend de son point d'application; plus précisément de son excentricité par rapport au centre de masse. Imaginons une boîte cubique flottant en état d'apesanteur. Si l'on exerce une légère poussée sur le milieu d'une de ses faces alors la boîte subira une translation. Si l'on applique maintenant une pression toujours dans la même direction mais cette fois sur l'un des coins de la boîte, la même translation sera accompagnée d'une rotation autour du centre de masse. On formule la composante angulaire d'une force par le couple $\vec{\tau}$, qui dépend de la position $\vec{C}$ du centre de masse et de la position $\vec{x}$ (dans le repère absolu) du point d'application de la force $\vec{F}$.

$$\vec{\tau} = (\vec{x} - \vec{C}) \times \vec{F}$$

Lorsqu'une force est appliquée à un objet, elle est décomposée en sa composante linéaire et en sa composante angulaire. Ses deux quantités sont ensuite stockées dans des accumulateurs de force et de couple, qui seront remis à zéro après chaque intégration. Ici, $\sum \vec{\tau}_i$ correspond aux forces accumulées pendant une itération. Il est de même pour le $\sum \vec{F}_i$ de l'intégration de la composante linéaire.

$$\vec{A}_{n+1} = \vec{A}_n + \sum \vec{\tau}_i$$

$$I_{abs} = RIR^T$$

2.2.2 Quantités auxiliaires

On sait désormais qu'un corps possède une orientation et un élan angulaire, on sait aussi comment passer de l'application d'une force à une variation d'élan angulaire. Néanmoins, l'intégration de la composante rotative d'un mouvement n'est pas aussi directe que pour sa version linéaire. Nous avons encore besoin de faire appel à plusieurs quantités auxiliaires, telles que le tenseur d'inertie et le tenseur d'inertie instantanée, avant de mesurer l'étendue de la variation d'orientation induite par l'application de forces.

Concrètement, un tenseur d'inertie est une matrice de dimension 3 dont les coefficients servent de facteurs lors du calcul de la variation d'orientation d'un corps à partir de son élan angulaire et représentent la répartition de la densité d'un corps. Il dépend directement de la forme du corps considéré et affecte les axes de rotation principaux de ce dernier. Pour le calculer précisément, on doit effectuer des intégrations par rapport au volume du solides considéré. Ces opérations sont un travail complexe en elles-mêmes mais pour notre plus grande satisfaction, la plupart des manuels de mécanique listent en annexe les tenseurs d'inertie usuel.

$$\begin{pmatrix} \frac{m}{12}(d_y^2 + d_z^2) & 0 & 0 \\ 0 & \frac{m}{12}(d_x^2 + d_z^2) & 0 \\ 0 & 0 & \frac{m}{12}(d_x^2 + d_y^2) \end{pmatrix}$$

Le tenseur d'inertie d'un corps sera attribué lors de la phase de préparation des solides et ne changera pas pendant la simulation. Le tenseur d'inertie instantanée est une autre quantité auxiliaure qui est le pendant absolu du tenseur d'inertie présenté précedemment, qui était lui plus local. Il doit être recalculé avant chaque intégration si le corps considéré à changé d'orientation depuis la dernière mise à jour.

2.2.3 Intégration

[ajouter détails]

$$\vec{A}_{n+1} = \vec{A}_n + \sum \vec{\tau}$$

$$I_{abs} = RIR^T$$

$$R_{n+1} = \frac{\vec{A}_{n+1}}{I_{abs}} R_n \partial t$$

3 Gestion des collisions

Maintenant que la première section nous a éclairé sur la façon dont les corps de la simulation évoluent indépendament les uns des autres, il est temps de les faire interagir entre eux. Cette phase se divise en trois étapes. Premièrement, le moteur physique doit être capable de déterminer si une collision a lieu entre deux corps. Ensuite, il doit pouvoir corriger leur état afin de contrebalancer les décalages dûs à l'intégration discrète de l'environnement. Finalement, une force de séparation doit être calculée et appliquée aux deux objets afin de simuler le rebond ou le repos.

3.1 Détection

Une collision fait entrer en jeu deux corps rigides dont l'intégrité physique a été corrompue, autrement dit : les corps rentrent l'un dans l'autre. Nous allons définir un traitement à appliquer à deux objets pour savoir si tel est le cas et de façon certaine. Néanmoins, bien que l'algorithme que nous allons présenter ait fait ses preuves dans le domaine des simulations physiques, il faut garder à l'esprit que ce test sera éxécuté à chaque intégration de la simulation et ce, pour chaque paire de

solides. Dans de telles circonstances, l'usage d'un algorithme a priori rapide peut se révéler désastreux au niveau pour les performances du moteur.

Nous faisons donc le choix de séparer la détection de collision en deux étapes : une détection grossière et une détection fine. La détection grossière fera usage d'un algorithme approximatif mais économique qui informe de la possibilité d'une collision, si le résultat est positif alors on exécute la détection fine pour confirmer ou infirmer le contact de façon certaine.

3.1.1 Détection grossière

Le but de la phase de détection grossière (*broad-phase collision*) est de renseigner sur la possibilité d'une collision et ce, à moindre coût. Si une collision a réellement lieu, le résultat sera toujours positif, néanmoins il est aussi possible que le résultat soit positif sans qu'aucune collision ne prenne vraiment place. Dans ce dernier cas, la détection fine invalidera la collision.

On se base sur l'utilisation de boîtes englobantes, ou *AABB* (*axis-aligned bounding boxes*), pour détecter l'éventualité d'un contact. Les boîtes englobantes sont des volumes alignés avec les axes du repère global qui contiennent tous les sommets d'un corps. Puisque tous les sommets de l'objet sont contenus dans l'AABB, il est évident que tous les points du corps le seront aussi. Pour savoir si deux corps rentrent possiblement en collision, on vérifie si leur boîte englobante entrent en collision.

Le processus de construction d'une boîte englobante est simple : on enregistre pour chaque axe le sommet du corps le plus éloigné dans la direction négative (`min` dans l'algorithme) et le sommet le plus éloigné dans la direction positive (`max`).

```
entrée : le corps C
sortie : une boîte englobante B

boite_englobante(C) :
  Pour tous les axes A
    Pour tous les sommets S de C
      Si S.A < B.A.min
        B.A.min = S.A
      Si S.A > B.A.max
        B.A.max = S.A
  retourner boite
```

Le test vérifiant la collision entre deux boîtes englobantes est très rapide puisque qu'il tire parti du fait que les boîtes sont alignées avec le repère absolu. Pour vérifier que deux AABB sont en état d'interpénétration, on utilise le théorème des axes de séparation selon lequel deux boîtes alignées sur les axes du repère absolu sont séparées si leur projection sur au moins un axe ne se superposent pas. En deux dimension, on peut résumer ce théorème par le fait que si une ligne parallèle à un des axes du repère global sépare deux corps, alors ils ne peuvent pas être en état de collision. En trois dimension le principe est le même, à la différence que l'on parle de plan de séparation.

```
entrée : deux boîtes englobantes B1 et B2
sortie : booléen

detection_collision_grossiere :
  Si B1.x.min > B2.x.max ou
     B1.x.max < B2.x.min ou
     B1.y.min > B2.y.max ou
     B1.y.max < B2.y.min ou
     B1.z.min > B2.z.max ou
     B1.z.max < B2.z.min
       retourner faux

  retourner vrai
```

La figure 3 illustre les trois situations possibles impliquant deux corps; un cercle et un rectangle. Sur la sous-figure *a*, les boîtes englobantes ne rentrent pas en collision car leur projection sur l'axe des abscisses ne se superposent pas. Il est donc impossible que les corps qu'elles contiennent soient eux-mêmes en état de collision, il est inutile d'aller plus loin et de lancer la détection fine.

La sous-figure *b* montre qu'il est possible dans certaines configurations que les boîtes englobantes entrent en collision sans que ce soit nécessairement le cas pour les corps qu'elles contiennent. Les boîtes sont en état d'interpénetration puisque leurs projections ne se séparent sur aucun axe. Ici, la détection grossière renvoit un résultat positif et c'est l'algorithme de détection fine qui réfutera la collision entre les deux solides.

La sous-figure *c* illustre quant à elle un troisième cas de figure dans lequel les boîtes entrent en collision et les corps aussi. La détection fine sera appellée et confirmera la collision.

3.1.2 Détection fine

La phase de détection fine (*narrow-phase collision*) est plus coûteuse mais détermine de façon certaine si deux corps sont en collision.

Introduisons en premier lieu la somme de Minkowski, une opération mathématique notée $A \oplus B = \{a + b \mid a \in A, b \in B\}$ avec A et B deux corps. On peut résumer la somme de Minkowski en un balayage de chaque corps par l'autre. Nous utiliserons une variante de cette opération : la différence de Minkowski, notée $A \ominus B = A \oplus (-B)$. La propriété de cette opération qui nous intéresse le plus est le fait que la plus petite distance entre les points qui la forment et l'origine du repère absolu est égale à la plus petite distance entre les corps A et B. La figure 4 illustre cette caractéristique. Nous pouvons exploiter cette singularité pour déterminer si deux corps entrent en collision. En effet, si la distance minimum entre les deux corps est supérieure à zéro, alors aucune collision ne peut possiblement exister. Si au contraire, la distance minimum est nulle, alors les deux objets sont en état d'interpénétration.

L'algorithme de détection fine consistera donc à se baser sur la différence de Minkowski M entre deux corps pour en calculer la plus petite distance la séparant de l'origine du repère absolu. Dans cette optique, on utilise l'algorithme GJK, pour *Gilbert-Johnson-Keerthi algorithm*. De façon générale, cet algorithme est capable de calculer efficacement la plus petite distance entre la coque formée par un nuage de points et un autre point quelconque de l'espace. Dans le cadre de notre problématique, le nuage de points correspond aux sommets de M et le point cible correspond à l'origine du repère absolu.

GJK se base sur l'utilisation de deux éléments : le simplex et le point de support. Un simplex d'un corps est une structure géométrique entièrement contenue dans ce dernier et étant caractérisée par une dimension. Un simplex de dimension 0 est un sommet, un simplex de dimension 1 est une arête, un simplex de dimension 2 est un triangle et un simplex de dimension 3 est un té-

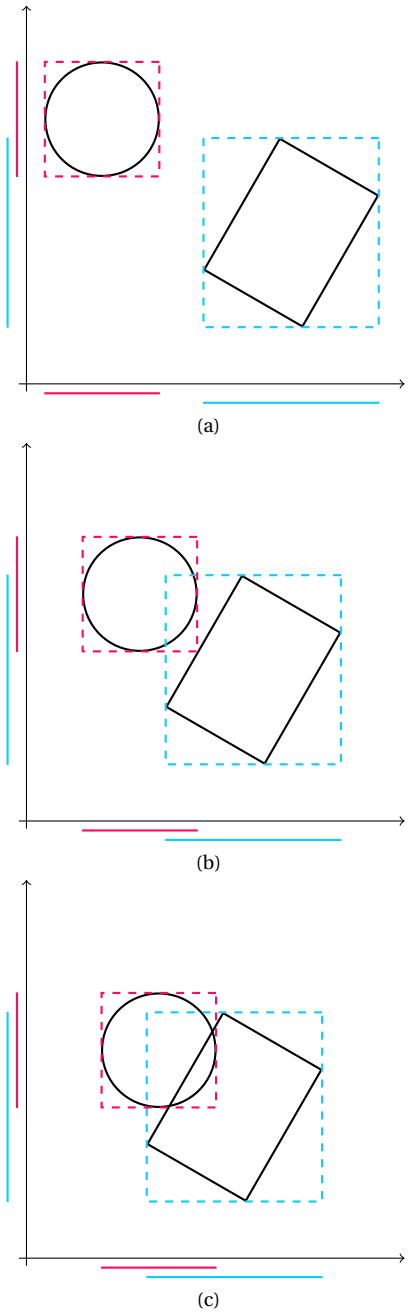

Figure 3: blabla

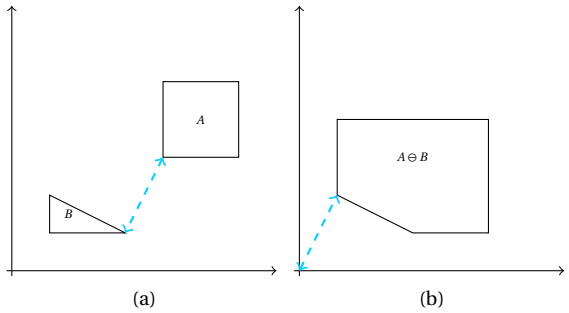

Figure 4: Différence de Minkowski entre un rectangle et un triangle

trahèdre. Un corps de dimension d peut contenir un simplex de dimension d au maximum. Au départ de l'algorithme on part d'un simplex de base choisi aléatoirement (souvent un sommet, donc un simplex de dimension 0). Ensuite, et à chaque étape, on augmente sa dimension pour l'agrandir jusqu'à ce que le point de M le plus proche de l'origine en fasse partie. À la fin de chaque étape, on détermine le point du simplex actuellement le plus proche de l'origine. Afin d'alléger la charge de calcul, il est aussi nécessaire de réduire la dimension du simplex dés que des sommets ne sont plus nécessaires à la définition du point actuellement le plus proche. En effet plus la dimension du simplex est faible et plus les calculs géométriques associés à la détermination du point du simplex actuellement le plus proche sont rapides.

Un point de support est un sommet de M fourni par une fonction de support $S(\vec{d})$, avec $\vec{d}$ la direction du point du simplex actuellement le plus proche de l'origine à l'origine. La fonction de support est employée dans l'algorithme principal de GJK pour obtenir le sommet de M le plus extrême dans la direction $\vec{d}$. Il est à noter que grâce à la fonction de support il n'est pas nécessaire de calculer explicitement la différence de Minkowski dans son intégralité [2], car $S_{A\ominus B}(\vec{d}) = S_A(\vec{d}) - S_B(-\vec{d})$.

La figure 5 illustre la succession d'étapes qui amène à la détermination du point le plus proche de l'origine O

- Le vertex A est choisi aléatoirement pour faire office de simplex de base. Le point du simplex actuellement le plus proche de l'origine ne peut que être A / S_0. On cherche un point de support dans la direction $\vec{S_0O}$.

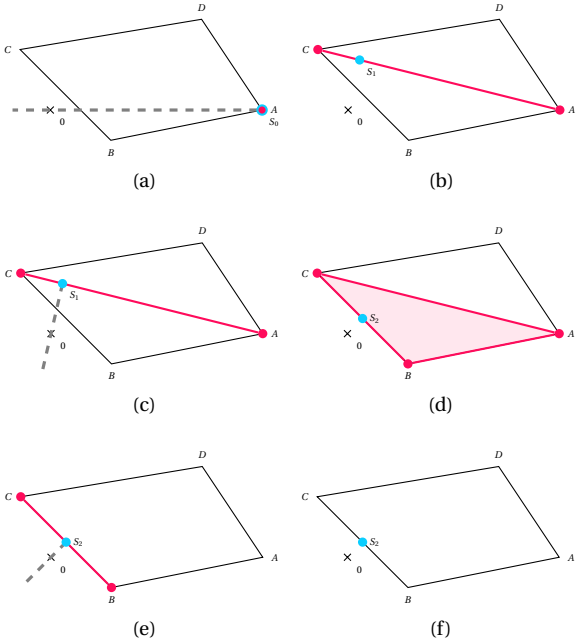

Figure 5: Étapes de l'algorithme GJK

- Le sommet le plus extrême dans la direction $\vec{AO}$ est C, on l'ajoute au simplex. Le point du simplex actuellement le plus proche de l'origine est S_1.
- On cherche le vertex le plus extrême dans la direction $\vec{S_1O}$.
- Le point de support retourné est B, on l'ajoute au simplex. Le point du simplex le plus proche de l'origine est $\vec{S_2}$.
- Le vertex A n'est plus utile à la définition de $\vec{S_2}$, on le retire du simplex. On cherche le vertex le plus extrême dans la direction $\vec{S_2O}$.
- Aucun vertex n'est plus extrême que $\vec{S_2}$ dans la direction $\vec{S_2O}$, c'est donc le point de M le plus proche de l'origine.

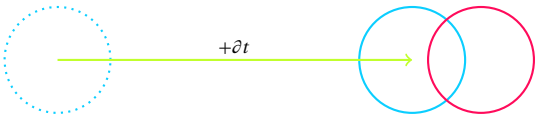

Figure 6: Situation d'interpénétration

3.2 Correction

À chaque mise à jour de la simulation, des corps sont susceptibles de se déplacer et donc d'entrer en contact les uns avec les autres. Le moteur simule ces évolutions par pas de temps fixe et il est donc improbable qu'une collision soit détectée au moment exact où elle se produit (c'est à dire à l'instant précis où les deux corps sont posés l'un contre l'autre). On est donc en permanence témoin de situations d'interpénétration au sein desquelles deux objets rentrent l'un dans l'autre. On pourrait ignorer cette imprécision et malgré tout calculer les forces de rebond appropriées à ce contact mais plusieurs situations critiques risquent d'apparaître. On pense notamment au fait que la force de séparation calculée soit sensible à cette erreur et ne déplace pas assez les corps à l'itération suivante pour qu'ils soient séparés.

Pour trouver le point de contact réel entre deux objets en collision, il existe plusieurs approches. Les systèmes préférant favoriser le temps d'éxécution au détriment du réalisme se contentent de repositionner les corps à un point de contact calculé en fonction de la profondeur de l'interpénétration. Mais même si la position de contact est corrigée par cette technique, il n'en sera pas de même pour les autres quantités physiques. Le corps recalé aura une position correcte mais son élan linéaire, par exemple, restera le même qu'à la fin de l'intégration. Même si la vitesse d'éxecution est un facteur important dans nos choix de conception, nous allons prendre un chemin différent et sélectionner une méthode plus précise qui fonctionnera par retour en arrière.

La fonction d'intégration du moteur prend comme seul argument le pas de temps duquel faire avancer la simulation et pour l'instant on utilisait un pas constant. Or, une caractéristique intéressante de cette fonction et qu'elle peut prendre en argument un pas de temps négatif et simuler l'évolution d'un système en sens inverse. Cette possibilité nous permet notamment de revenir en arrière dans la simulation dés qu'une interpénétration est détectée et ce jusqu'à retrouver le point de contact exact. Le terme "exact" est à prendre avec des pincettes puisque le moteur physique est limitée par l'arithmétique des nombres à virgules flottantes et que l'on doit se contenter d'un résultat validé par un seuil de tolérance adapté. Le pas de temps choisi devra correspondre à une fraction du pas de temps originel puisque le but de cette manœuvre est de déterminer à quel moment du pas de temps les corps sont réellement entrés en contact.

Usuellement, ce retour en arrière s'effectue en intégrant plusieurs fois la simulation pas un pas de temps négatif et fixe, par exemple $-\frac{\partial t}{10}$. Néanmoins cette méthode s'adapte mal aux simulations contenant des objets évoluant à hautes vitesses puisqu'en utilisant cette même fraction du pas de temps, un corps s'étant déplacé de dix mètres pendant l'intégration sera au moins recalé un mètre avant tout contact réel, même si la profondeur de pénétration n'était que de quelques centimètres.

Afin d'accélérer la phase de recherche on fait le choix de procéder par dichotomie. À chaque étape i de cette recherche, on recule de $\frac{\partial t}{2i}$ unité de temps si les corps s'interpénétrent, sinon on avance de cette même durée. On considère qu'une position de contact valide a été déterminée lorsqu'il n'y a plus d'interpénétration et que la distance entre les objets est inférieure à un seuil de tolérance choisi au préalable. Le processus est visible sur la figure 7.

```
entrée: deux corps A et B, le pas de temps dt
sortie: deux corps A et B

correction:
  d = distance_GJK(A,B)

  si 0 <= d <= seuil
    retourner A et B
  sinon d = 0
    A.integrer(-dt/2)
    B.integrer(-dt/2)
  sinon
    A.integrer(dt/2)
    B.integrer(dt/2)
```

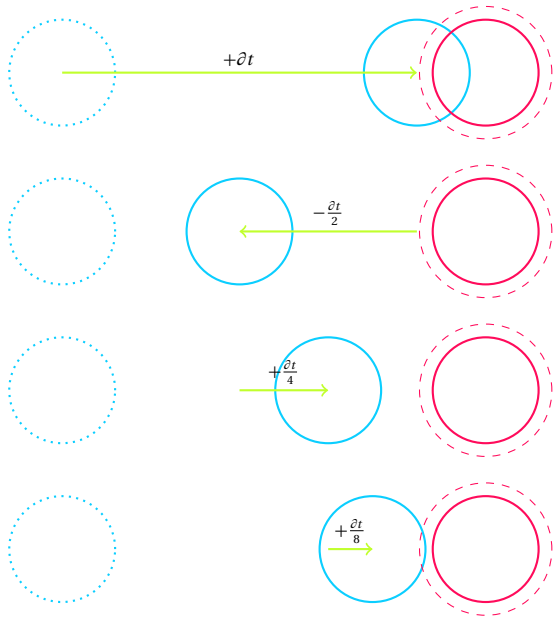

Figure 7: Correction dichotomique de la configuration de contact

Une fois ce processus achevé, la configuration de contact est retrouvée et les deux corps ne sont plus en situation de pénétration mutuelle. On remarque que grâce à cette technique, la position n'est pas la seule valeur à avoir été recalée : orientation, élans angulaire et linéaires sont eux aussi revenus aux valeurs qu'ils auraient dû atteindre à l'instant du contact.

3.3 Réponse

[Points de contacts]

Un coefficient de restitution ε, avec $0 \leq \varepsilon \leq 1$, détermine le taux de rebond d'un corps. Si $\varepsilon = 1$, la balle rebondira avec autant d'énergie qu'elle est arrivée (une situation concrètement impossible à retrouver dans le monde réel). Si $\varepsilon = 0$, la balle repartira avec une énergie nulle; elle restera collée au plan. Le coefficient de restitution dépend habituellement de la matière que l'on cherche à simuler : une balle en caoutchouc aura un ε elevé tandis qu'un bloc d'argile aura un ε proche de zéro.

[Friction]
[5]

4 Le moteur physique

4.1 Ordre des tâches

4.2 Démonstrations

4.3 Limitations et perspectives d'évolution

4.3.1 Tunneling

Le moteur physique que nous concevons fonctionne par intégration discrète et si aucun contrôle n'est effectué pour contrecarrer les effets négatifs de cette caractéristique, on risque d'obtenir des résultats imprécis ou pire, complètement éloignés de ce que l'on retrouverait dans la réalité. On a par exemple vu plus tôt qu'une phase de recherche dichotomique est nécessaire pour recaler les corps à leur position réelle de contact lorsqu'une collision est détectée.

Néanmoins, une éventualité n'a pas été envisagée : que se passerait-il si un objet en traversait entièrement un autre pendant un pas de temps ? On peut imaginer un objet A lancé vers un autre objet B fixe. À l'instant t, A n'entre pas en collision avec B mais se dirige dans sa direction. À l'instant $t+1$, A a entièrement traversé B. À aucun de ces deux instants une collision n'a été détectée et pourtant A est impunément passé à travers B. Le pas de temps utilisé pour réguler l'intégration du système était donc trop faible pour s'assurer des collisions entre corps évoluant à haute vitesse. À l'heure de l'écriture de ce rapport, le moteur physique est sensible à ce genre de situation. Réellement, ce cas ne se produit pas, puisque les simulations développées représentent des situations terrestres qui font participer des forces d'opposition telles que le frottement de l'air et donc les corps n'atteignent jamais de vitesses démesurées. On souhaite tout de même disposer d'un moteur physique solide et versatile, évaluons les possibilités qui s'offrent à nous pour pallier ce problème.

On pourrait envisager diminuer la durée du pas de temps d'intégration afin de diminuer le risque de tunneling, mais même si un pas de temps faible améliore la qualité des résultats, il ne pourra pas être indéfini-

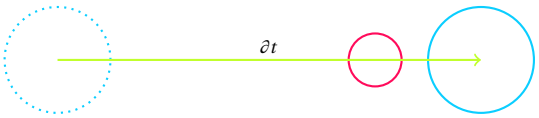

Figure 8

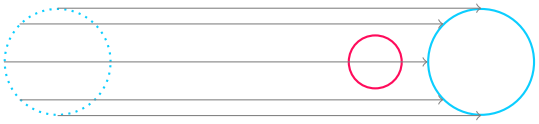

Figure 9

ment réduit. Il est principalement limité par le temps de calcul d'une mise à jour du système, puisqu'à partir du moment où ∂t est inférieur au temps moyen nécessaire à une machine pour calculer une itération de la simulation, le programme perdra son statut d'application temps réel.

Une technique usuellement admise est le lancer de rayons (*raycasting*) entre les points de la position de départ du solide et les points de sa position d'arrivée. Si l'un des rayons touche un autre corps, on sait qu'une collision aurait dûe être détectée et on peut revenir en arrière. Cette méthode présente pourtant une faiblesse de taille, car le résultat de cette recherche dépend directement de la concentration de rayons. Si on lance peu de rayons, on risque de ne pas détecter les corps de petite taille et donc de les traverser tandis qu'une concentration élevée de rayons pourrait se réveler regrettable en terme de coûts de calcul.

La technique du ??? permet de contrôler de façon certain tout phénomène de tunneling. L'idée est la suivante : soit deux corps A et B dont l'on veut détecter l'éventuelle collision. On construit deux volumes fantômes F_A et F_B englobant tous les points par lesquels passent respectivement A et B entre deux intégrations. Ces volumes n'interviendront pas dans les collisions des autres objets de la simulation, et on vérifie simplement s'ils rentrent tous deux en collision. Si tel est le cas, alors on sait qu'il est possible qu'un tunneling se soit produit. L'un des avantages majeurs de cette méthode, en plus du fait qu'elle ne rate aucun tunneling, est sa simplicité de mise en place puisque toutes les routines utilisées (détection de collision entre volumes, retour en arrière)

Figure 10

ont été écrites et entrent déjà en jeu dans le fonctionnement de base du moteur physique.

4.3.2 Partitionnement de l'espace

Comme mis en avant tout au long de ce rapport, on cherche à concevoir un moteur physique dont la rapidité d'éxécution est un facteur déterminant. Dans cette optique, il est primordial d'optimiser les routines géométriques entrant en jeu mais aussi d'éviter leur éxécution inutile dés que possible. La détection des collisions entre les corps qui peuplent notre système se déroule en deux étapes : une détection grossière visant à vérifier de façon économique si deux objets se touchent et une détection fine servant à confirmer ou à infirmer de façon sûre ce résultat. Bien que la détection grossière soit peu coûteuse en termes de calcul, elle est éxécutée ??? à chaque itération, où n est le nombre de corps actifs dans la simulation. Pourtant, il est très souvent inutile de détecter si deux objets sont en situation d'interpénétration, par exemple lorsqu'ils sont très éloignés l'un de l'autre. Il serait tout à notre avantage d'ajouter au moteur une couche supplémentaire tenant compte de ces disparités spatiales pour accélérer la détection de collisions.

Le principe général du partitionnement de l'espace (*Spatial partitioning*) est de subdiviser l'environnement de la simulation en sous-espaces. La répartition des corps dans ces sous-espaces dépend bien évidemment de leur position dans le repère global mais aussi du volume qu'ils occupent. On est certain que seules les entités appartenant à un même sous-espace peuvent interagir entres elles, on dispose donc d'une aide quant au choix des objets entre lesquels vérifier si collision il y a.

Sous sa forme la plus simple, le partitionnement de l'espace prend la forme d'une grille de taille fixe dans les cases de laquelle les corps de la simulation sont ré-

partis. L'aspect statique d'une telle structure présente plusieurs inconvénients majeurs, notamment le fait que le gain de performance variera fortement selon le nombre d'objets considérés, leur taille et la finesse de la grille. Par exemple, si une grille a des cases trop grandes, chacune contiendra de nombreux objets, et le gain de performance sera peu flagrant. Si au contraire les cases de la grille sont trop petites, les plus grands objets ne pourront pas rentrer dans leur intégralité à l'intérieur d'une unique case. Une solution à ce problème est de ranger un même objet dans plusieurs cases différentes de la grille, mais une fois encore, si des corps sont démultipliés de la sorte, le temps de maintenance d'une telle structure sera augmentée.

Une variante plus subtile de partitionnement de l'espace est le partionnement binaire blabla.

Quatree, octree, blabla

4.3.3 Extension aux corps concaves

References

[1] B. Bitterli. A verlet based approch for 2d game physics, 2009.

[2] C. Ericson. *Real-Time Collision detection.* 2005.

[3] G. Fiedler. Integration basics, 2006.

[4] G. B. A. Hans-Jurgen Weber. *Essential Mathematical Methods for Physicists.* 2003.

[5] I. Newton. *Philosophiæ Naturalis Principia Mathematica.* 1687.

[6] A. Witkit and D. Baraff. An introduction to physically based modeling. Technical report, Carnegie Mellon University, 1997.