TITRE

Merwan Achibet

Table des matières

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 affinnées 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 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 le cadre de ce type d'application, auquel on peut greffer 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 une balle de caoutchouc jetée sur un mur. 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 corps rigides 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 générée. 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 lachions au dessus d'un plan faiblement incliné, 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 lache 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 en apesanteur. En réalité, la gravité ne sera pas codée "en dur" mais appartiendra à une liste de forces environnementales qui contiendra toutes les influences que le monde extérieur applique aux objets qu'il contient. On pourra par exemple modéliser, entre autres, la résistance de l'air ou le roulis irrégulier du vent.

Cette observation 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. C'est la façon dont ces variables évolueront et s'influenceront entres elles qui déterminera la qualité du rendu du moteur physique. 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 incliné, 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 point des deux corps entrent en contact.

Une difficulté supplémentaire vient du fait que la simulation est mise à jour de façon discrète. 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 ordinateurs, 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. Puisque qu'elle repose sur un plan légerement incliné, elle roule quelques instants, de plus en plus lentement, jusqu'à parvenir à un arrêt total.

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 est 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 :

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

DESSIN
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, des micro-collisions lui permettent de rester à la surface du plan et d'équilibrer le système. On retrouvera ce phénomène dans le moteur physique.

2 Dynamique

2.1 La composante linéaire du mouvement

Nous allons pour l'instant 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 sont en deux dimensions pour des raisons de clarté mais le principe reste similaire lorsqu'il est étendu à la troisième dimension.

La quantité physique la plus perceptible visuellement pour un spectateur est la position $\vec{p}$ d'un corps. Pour notre affichage final, c'est cette valeur à tout instant t de la simulation que l'on veut déterminer. Le travail de la partie dynamique du moteur physique revient donc à déterminer la nouvelle position d'un objet à partir de la connaissance des forces qui lui sont appliquées. 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 dépendante de l'accélération.

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

Grâce à cette équation, 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 ces trois quantités sous forme vectorielle et l'une des tâches de base du moteur physique sera de traduire l'application de forces en un changement de position. On sait qu'elles entretiennent des relations de dérivation, il faudra donc procéder par intégration pour calculer la nouvelle position d'un objet à partir de son accélération.

2.2 Intégration

Dans la sous-section précédente, les trois quantités physiques entrant en jeu dans les mouvements linéaires ont été présentées. Étudions maintenant de façon plus concrète sur quels calculs se basera le travail 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 modèle du moteur physique sera donc discret 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}. 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 par rapport à l'état précédent.

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

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

$$\begin{aligned}
\vec{a}_{n+1} &= \frac{\sum \vec{F_i}}{m} \\
\vec{v}_{n+1} &= \vec{a}_n + \vec{a}_{n+1}\partial t \\
\vec{p}_{n+1} &= \vec{p}_n + \vec{v}_{n+1}\partial t
\end{aligned}$$

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 divise un pas de temps en plusieurs segments, ou bien telle que l'intégration de Verlet qui dispose d'une meilleure stabilité, mais Euler reste le choix le plus économique et fournit des résultats relativement corrects.

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 :

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

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

2.3 Modélisation d'un corps rigide

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 ?

On pourrait en premier lieu penser à représenter un tel corps par une liste de particules, chacune placée à un coin 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.

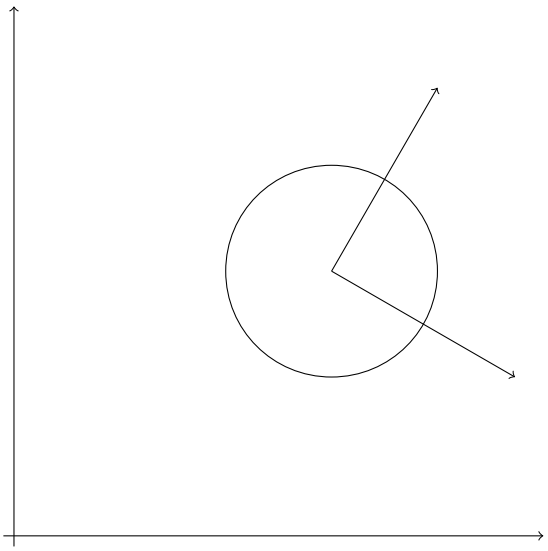


FIGURE 1 – Le repère absolu et le repère local d'un cercle

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 la simulation. Un repère local est un référentiel qui est unique à chaque corps et dont le centre se situe à l'intérieur même de cet objet.

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

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

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 :

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

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

2.4 La composante angulaire du mouvement

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 corps rigiques 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.

2.4.1 L'orientation

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, on utilisera une matrice de dimension 3 au sein de laquelle chaque vecteur colonne correspondra à un des axes du repère local [Witkit and Baraff, 1997]. Pour illustrer ces propos, analysons la matrice d'orientation égale à la matrice identité d'ordre 3.

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

Si l'on isole les vecteurs colonnes de cette matrice identité, on remarque que chacun correspond à un des axes du repère global. Le premier vecteur colonne d'une matrice d'orientation correspond donc 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.

2.4.2 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 très important 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 accompagné d'une rotation autour du centre de masse. On quantifie cette effet rotatif par le couple τ, qui dépend de la position $\vec{C}$ du centre de masse et de la position $\vec{x}$ du point d'application de la force.

$$\tau = (\vec{x} - \vec{C}) \times \vec{F} \tag{1}$$

2.5 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.

Concrètement, un tenseur d'inertie est une matrice de dimension 3 dont les coefficients servent de facteurs lors du calcul de l'orientation à partir de l'élan angulaire. Il dépend directement de la forme du corps considérés et affecte les axes de rotation principaux de ce dernier. Pour le calculer on doit effectuer des calculs d'intégration par rapport au volume du corps considéré. Les tenseurs d'inertie des volumes usuels sont souvent donnés dans les manuel de physique.

Le tenseur d'inertie instantanée est le pendant local du torseur d'inertie. blabla

$$I(t) = R(t) I R(t)^T$$

2.6 Intégration

$$\begin{aligned} I(t) &= R(t) I R(t)^T \\ \vec{w}_{n+1} &= \frac{\vec{A}}{I(t)} \\ R &= \vec{w} R \partial t \end{aligned}$$

2.6.1 Correction de l'intégration

Notre processus d'intégration est désormais complet et nous pouvons simuler les réactions de corps soumis à des forces externes. Il reste pourtant un problème à résoudre : à cause de l'utilisation de nombres à virgule flottante, de légères erreurs de précision apparaîssent à chaque intégration du système et s'accumulent. L'erreur

la plus flagrante touche l'orientation des objets (l'intégration linéaire ne passant que par des opérations de base, peu d'erreurs se font ressentir). Le problème principal vient du fait que la matrice d'orientation R, qui détermine la direction des axes du repère local, perd progressivement de son orthonormalité : les axes ne sont plus orthogonaux entre eux et leur longueur n'est plus égale à 1. Lors du rendu graphique de la simulation, on utilise la liste des points qui forment un corps ainsi que sa matrice de rotation pour le dessiner, et à cause de cette erreur de précision les solides déssinés sont déformés.

Afin de contrer cet effet indésirable, on ajoutera deux étapes supplémentaires à la fin de chaque intégration. Premièrement, il nous faudra normaliser les axes du repère local. Chaque axe correspondand à un vecteur colonne de la matrice d'orientation, il faudra 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 l'algorithme de Gram-Schmidt. blabla

3 Gestion des collisions

Maintenant que la première section nous a éclairé sur la façon dont les corps de la simulation évoluent indépendamment 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 contre-balancer les décalages dûs aux erreurs numériques et 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 d'une collision

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 fait ses preuves ait fait ses preuves dans le domaine des simulations physique, il faut garder à l'esprit que ce test sera éxécuté à chaque intégration de la simulation et ce, pour chaque paire de corps. Dans de telles circonstances, l'usage d'un algorithme a priori rapide peut se révéler désastreux au niveau des performances.

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 qui informe de la possibilité d'une collision, si le résultat est positif alors on exécute la détection fine (mais plus coûteuse) 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 possible que le résultat soit positif sans qu'il y ait de collision réelle. 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 et contenant tous les points d'un corps. Pour savoir si deux corps rentrent possiblement en collision, on vérifie si leurs boîtes englobantes entrent en collision.

La construction d'une boîte englobante revient à enregistrer les deux points les plus extrêmes d'un corps (dans le sens négatif et dans le sens positif) et ce pour tous les axes de l'environnement de la simulation.

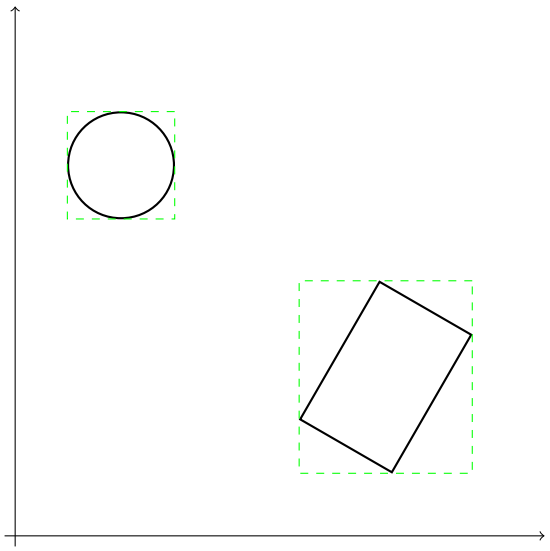
```
construire_boite_englobante :
  boite.min = [ ]
  boite.max = [ ]

  Pour tous les points du corps
    Pour tous les axes
      Si point[axe] < boite.min[axe]
        boite.min[axe] = point[axe]
      Si point[axe] > boite.max[axe]
        boite.max[axe] = point[axe]
```

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 et qu'on peut pour vérifier que deux AABB son en état d'interpénétration il suffit de vérifier qu'elles ne sont pas séparées sur tous les axes [Ericson, 2005].

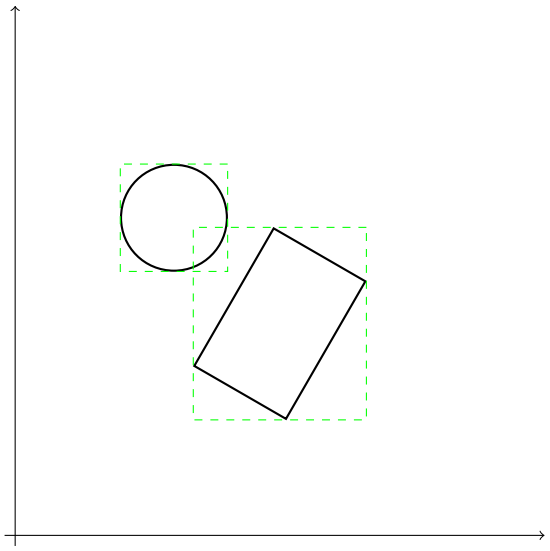
```
detection_collision_grossiere :
  Si b1.min[x] > b2.max[x] ou
     b1.max[x] < b2.min[x] ou
     b1.min[y] > b2.max[x] ou
     b1.max[y] < b2.min[x] ou
     b1.min[z] > b2.max[z] ou
     b1.max[z] < b2.min[z]
       retourner faux

  retourner vrai
```

3.1.2 Détection fine

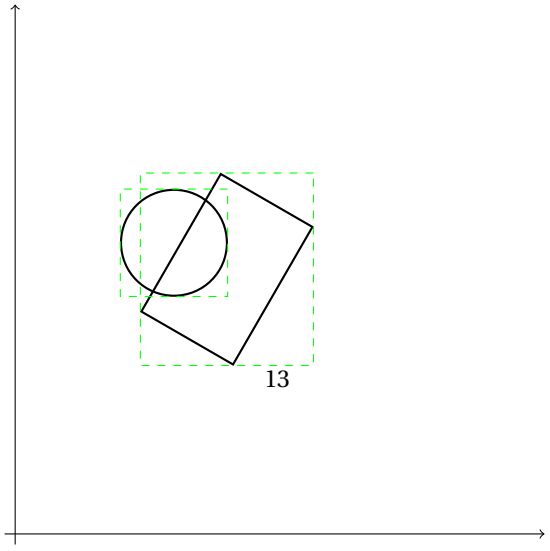
La phase de détection fine (*narrow-phase collision*) est plus coûteuse mais détermine de façon certaine si deux corps sont en collision.



(a)



(b)



(c)

FIGURE 2 – Somme de Minkowski entre un rectangle et un cercle

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 impliqué dans cette opération par l'autre.

Nous utilisons 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. Nous pouvons exploiter cette caractéristique 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 inférieure à zéro, alors les deux objets sont état d'interpénétration.

Le processus de détection fine passera donc par le calcul de la différence de Minkowski M entre deux corps. Une fois M calculée, il reste encore à calculer la distance séparant M et l'origine du repère absolu. Dans cette optique, on utilise l'algorithme GJK, ou *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 point précis de l'espace. Dans le cadre de notre problématique, le nuage de points correspond aux vertex de M et le point cible correspond à l'origine du repère absolu.

GJK se base sur l'utilisation de simplex. 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 point, un simplex de dimension 1 est une ligne, une simplex de dimension 2 est un triangle et un simplex de dimension 3 est un tétrahèdre. Un corps de dimension d peut contenir un simplex de dimension d au maximum. Le principe de GJK est de partir d'un simplex de base choisi alétoirement (souvent un point, donc un simplex de dimension 0) et d'augmenter sa dimension pour se rapprocher progressivement du point de la figure le plus proche du point cible. À la fin de cette procédure, on a determiné, le point de M le plus proche de l'origine, et une simple soustraction vectorielle permet de calculer leur distance.

La figure 3 illustre la succession d'étapes qui amène à la détermination du point le plus proche de l'origine O

- Le point A est choisi aléatoirement pour faire office de simplex de base. Le point actuellement le plus proche de l'origine est naturellement A.
- On cherche le point de la figure le plus extrême dans la direction du point actuellement le plus proche (A) à l'origine. C'est le point C. On l'ajoute au simplex.
- On détermine le point du simplex le plus proche de l'origine (p_1) et on calcule quel est le point de la figure le plus extrême dans la direction de ce point à l'origine. C'est le point B.
- Le simplex a atteint une dimension de 2 mais le point A n'est plus utile à la

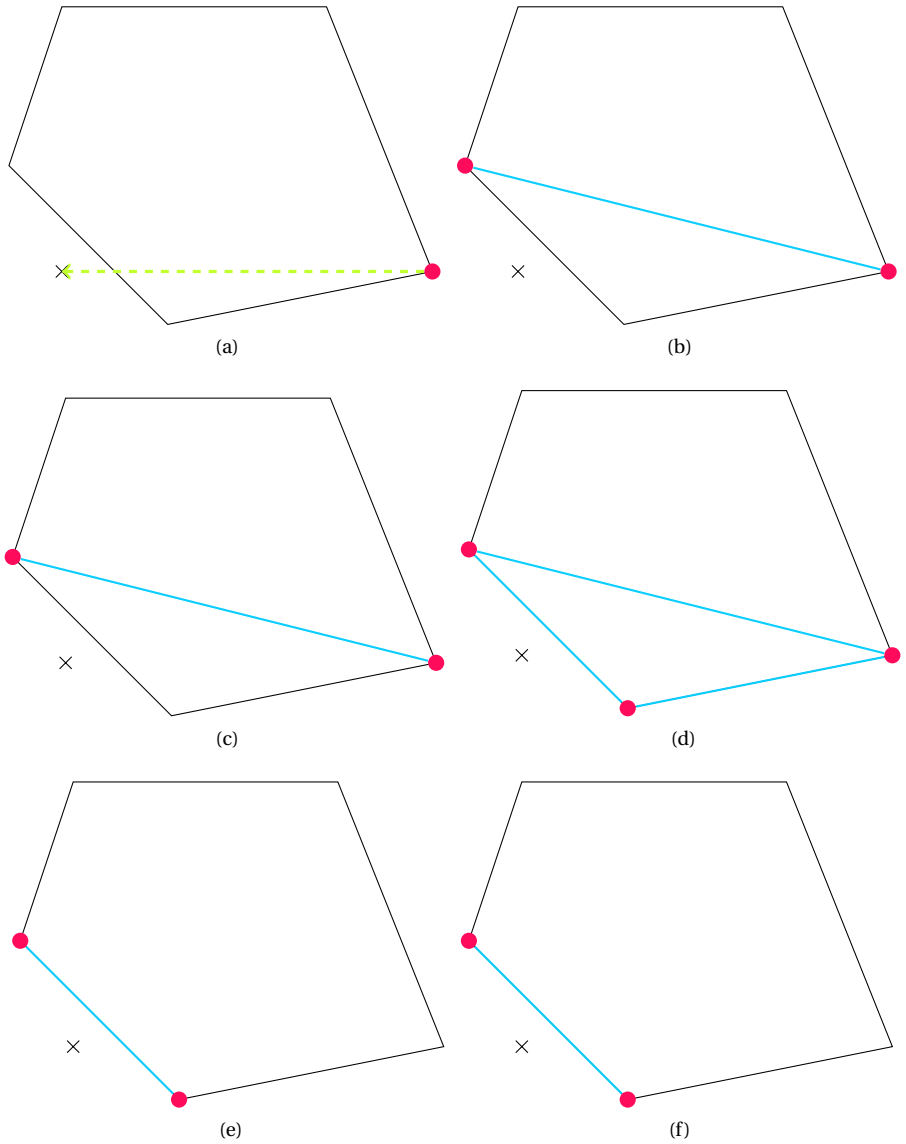


FIGURE 3 – Étapes de l'algorithme GJK

définition du point actuellement le plus proche. On le retire du simplex.
– On cherche le point du simplex le plus proche de l'origine et on détermine quel est le point de la figure le plus extrême dans sa direction vers l'origine. Aucun n'est plus extrême que le point actuellement le plus proche, on peut donc arrêter la recherche.

3.2 Correction d'une collision

À chaque itération de la simulation, les corps présents dans l'environnement du moteur physique sont susceptibles de se déplacer et donc d'entrer en contact. Le moteur simule ces évolutions de manière discrète et il est très improbable qu'une collision soit détectée au moment précis où elle se produit. On est donc régulièrement témoin de situations d'inter-pénétration au sein desquelles des corps rentrent l'un dans l'autre.

DESSIN : inter-pénétration

On pourrait se satisfaire de cette imprécision et calculer les forces de rebond appropriées mais plusieurs situations critiques risquent d'apparaître. Pour trouver le point de contact réel entre deux objets, 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 par interpolation en fonction du temps de la dernière intégration. Il est évident que des objets sujets à des déplacements non linéaires (variation de la vitesse pendant un pas de temps, trajectoire elliptique) seront mal replacés.

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 en sens inverse. Grâce à cette caractéristique, dés qu'une inter-pénétration est détectée, on peut revenir en arrière jusqu'à retrouver le temps de contact exact. Le terme "exact" est à prendre avec des pincettes puisque notre simulation est limitée par l'arithmétique des nombres à virgules flottantes et que l'on devra se contenter d'un résultat quasi-exact, validé par un seuil de tolérance adapté.

Afin d'accélérer cette phase de recherche de la configuration de contact, le retour en arrière se fera par dichotomie. On recule en premier lieu de la moitié du pas de temps puis, si l'inter-pénétration est toujours présente, on recule d'un quart du pas de temps, sinon on avance d'un quart du pas de temps, et ainsi de suite jusqu'à obtenir une précision satisfaisante.

ALGORITHME

DESSIN : recherche de la vraie position 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 à leurs valeurs au moment du contact.

3.3 Réponse à une collision

Un coefficient de restitution $0 \leq \varepsilon \leq 1$ 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 ; autrement dit, elle restera collée au plan. Le coefficient de restitution dépend habituellement de la matière que l'on cherche à simuler : une balle en cahoutchouc aura un ε elevé tandis qu'un bloc d'argile aura un ε proche de zéro.

Le coefficient de friction influe sur la réponse rotationnelle qu'une collision est susceptible de provoquer. Si la balle avait été laché, à partir d'une vitesse nulle, sur un plan droit, le rebond aura été purement linéaire. Dans le cas d'un plan incliné, un frottement non perpendiculaire a lieu et la balle va entrer en rotation. Plus le coefficient de friction est élevé et plus cet effet est puissant. On peut simplifier cette notion en disant que le coefficient de friction détermine si un corps est recouvert d'une matière qui accroche ou au contraire qui glisse.

[Newton, 1687]

4 Le moteur physique

4.1 Ordre des tâches

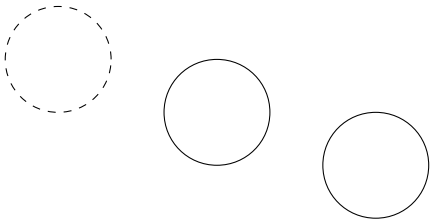
4.2 Démonstrations

4.3 Limitations et perspectives d'évolution

4.3.1 Tunneling

Le fait que ce moteur physique soit une simulation discrète nécessite une attention particulière afin d'éviter les situations imprécises menant à des résultats différents de ce à quoi on pourrait s'attendre. Comme vu plus tôt, une phase de recherche dichotomique est par exemple 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. Á aucun de ces deux instants une collision n'a été détectée et pourtant A est impunément passé à travers B sans qu'aucun recalage n'ai été exécuté. 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 jamais, puisque les simulations développées représentent des situations du terrestre 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 l'effet de tunneling, mais même si un pas de temps petit améliore la qualité des résultats, il ne pourra pas être indéfiniment réduit. Il est tout d'abord limité par le temps de calcul d'une mise à jour du système, puisqu'à partir du moment où ∂t est inférieur au temps nécessaire à une machine pour calculer une mise à jour, le programme perdra son statut d'application temps réel.

Une technique usuellement admise est le lancer des rayons (raycasting) entre les points de la position de départ du solide et les points de sa position d'arrivée. De cette façon si un rayon 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, le résultat de cette recherche dépend directement de la concentration de rayons. Peu de rayons risquent de ne pas toucher les corps de petite taille et donc de les laisser passer tandis qu'une concentration élevée de rayons serait regrettable en terme de coûts de calcul.

La technique du ??? permet de ne laisser passer aucun 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 englobant tous les points par lesquels respectivement A et B sont passés. 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 dffsdfdsqu'il est possible qu'un tunneling se soit produit et donc on revient en arrière pour vérifier si tel est le cas. 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) ont déjà été écrites et entrent en jeu dans le fonctionnement de base du moteur physique.

4.3.2 Partitionnement de l'espace

Références

[Ericson, 2005] Ericson, C. (2005). *Real-Time Collision detection.*

[Newton, 1687] Newton, I. (1687). *Philosophiæ Naturalis Principia Mathematica.*

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