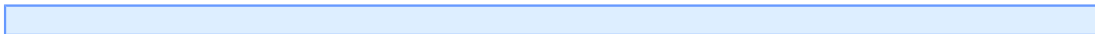


Conception d'un moteur physique

par [Gregory Corgie](#)

Date de publication : 15 septembre 2008

Dernière mise à jour :



Mon travail.....	3
Introduction.....	3
Algorithme général.....	3
Leçon 1 : Calculs des nouvelles positions.....	4
Équations de la dynamique.....	5
Équation de la cinématique.....	5
Leçon 2 : Gestion d'une sphère.....	6
Collision Sphère / Plan.....	6
Détection de la collision.....	6
Gestion de la collision.....	6
Recalage au point de contact exact.....	7
Collision Sphère / Sphère.....	8
Détection de la collision.....	8
Gestion de la collision.....	9
Recalage au point de contact.....	10
Leçon 3 : Gestion d'une box - Partie 1/2.....	11
La collision Boîte / Plan.....	11
La détection de la collision.....	11
Gestion de la collision.....	12
Théorie de l'impulsion.....	13
Classification du mouvement.....	14
Méthode de Poisson.....	15
Leçon 4 : Gestion d'une box - Partie 2/2.....	16
Application à la collision Box / Plan.....	16
Recalage au point de collision.....	16
Mise en équation du problème.....	16
Méthodes de résolution.....	17
La collision Box / Sphère.....	18
Détection de la collision.....	18
Gestion de la collision.....	19
Collision Boîte / Boîte.....	19
Détection de la collision Boîte / Boîte.....	19
Gestion de la collision Boîte / Boîte.....	20
Architecture.....	22
Conclusion.....	24
Bibliographie.....	24

Mon travail

Dans le cadre d'un projet d'étude mené lors de ma dernière de cycle ingénieur, j'ai développé avec l'aide d'un ami, **Julien Peyre**, un début de moteur physique. Nous avons géré les collisions pour des primitives de base.

- Ce projet a été développé sous Visual C++ Express
- Pour faire le rendu de nos calculs, nous avons utilisé OpenGL
- Quelques vidéos d'exemples sont disponibles  [ici](#)
- Les sources sont  [là](#)
- Le plus intéressant à mon sens est le  [rapport écrit](#), dans lequel les principes physiques et mathématiques permettant une implémentation.

Introduction

Avant de devenir une science à part entière, la mécanique a longtemps été une section des mathématiques. Elle en était un domaine applicatif totalement naturel.

La mécanique statique a été le premier domaine étudié avec les notions fondamentales comme l'équilibre ou les forces. Ensuite, ces notions ont laissé place à la dynamique, basée sur les phénomènes qui régissent les mouvements des solides. Deux scientifiques ont apporté une immense contribution à ce domaine : Galilée, avec sa théorie sur la chute des corps, et Newton, avec ces célèbres lois du mouvement.

La puissance des processeurs actuels permet la réalisation de ces calculs en temps réels. Ainsi il devient possible de modéliser et de visualiser le comportement des solides. Dès lors on s'aperçoit que les lois, établies depuis le XVIème siècle, fournissent des résultats très satisfaisants. Elles sont d'ailleurs communément utilisées dans les jeux vidéo.

Quel est le lien entre les jeux vidéos et Newton nous direz-vous ? Suite à l'essor des jeux vidéo en tant que loisirs de masse, les développeurs sont à la recherche d'univers toujours plus immersifs. Pour cela, ils souhaitent modéliser le plus fidèlement possible les phénomènes naturels. Des efforts importants ont été faits ces dernières années dans deux domaines de recherche : la qualité graphique et la dynamique.

Un nouveau type d'application est alors apparue : les moteurs physiques. Un moteur physique est une bibliothèque intégrable à un programme qui permet la résolution des équations physiques. Typiquement, il permet de détecter les collisions entre plusieurs solides et de calculer leurs réponses.

Ce rapport a pour but de présenter un projet qui pourrait constituer un début de moteur physique. Reposant sur des formes géométriques de base, nous avons souhaité implémenter les lois de la dynamique.

Algorithme général

Les objets de la scène sont donc décrits dans un fichier texte qui va être lu par le programme. Ce fichier va notamment initialiser les positions des objets. Leur position initiale est ainsi fixée dans ce fichier. Ensuite, à chaque frame, le moteur va réaliser ce même processus :

- Calcul des nouvelles positions grâce à une intégration dans le temps des équations de la dynamique sans tenir compte des collisions.
- Détection des collisions grâce à des algorithmes spécifiques à chaque type de collisions.
- Gestion éventuelle de ces collisions pour relancer le système dans un état valide : calcul des vitesses réfléchies.
- Mise à jour de l'état des solides.



Leçon 1 : Calculs des nouvelles positions

Pour simuler le mouvement des objets nous avons basé notre implémentation sur la mécanique des solides indéformables. Pour décrire le mouvement d'un objet cinq paramètres indispensables doivent être définis :

- Sa position,

$\vec{P}$

- Sa masse,

M

- Son repère locale,

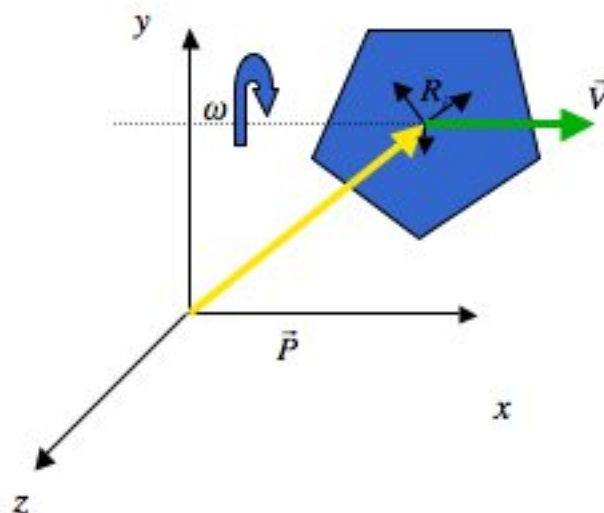
R_i

- Sa vitesse linéaire (vitesse du centre de masse),

$\vec{V}$

- Sa vitesse angulaire,

ω



Représentation d'un solide

Cependant il est à noter que dans l'implémentation de notre moteur nous n'avons pas tenu compte de la vitesse angulaire des sphères. En effet, nous avons décidé de mettre de côté cet aspect puisque si l'on associe une texture uniforme à une sphère, sa rotation n'est pas perceptible. En outre, ceci peut être considéré comme amélioration future.

Équations de la dynamique

Nous utilisons dans ce projet les deux équations fondamentales suivantes :

$$\begin{aligned} m\vec{a} &= \sum_i \vec{F}_i \\ I \frac{d\omega}{dt} &= \sum_i M_i \end{aligned}$$

m : Masse de l'objet.

$\vec{F}_i$: Force i appliquée sur l'objet.

$\vec{a}$: Accélération de l'objet.

I : Matrice d'inertie de l'objet

M_i : Moment i appliqué sur l'objet

A chaque frame, nous réalisons un bilan des forces exercées sur l'objet pour calculer la nouvelle accélération ainsi que le bilan des moments pour en déduire la nouvelle vitesse angulaire.

La matrice d'inertie permet de traduire la répartition de la masse du solide et ainsi de favoriser certains couples. Elle joue grossièrement le même rôle pour la vitesse angulaire que la masse pour la vitesse linéaire. La matrice d'inertie d'un solide a la forme suivante :

$$I = \iiint \rho(x,y,z) \begin{bmatrix} (y^2 + z^2) & xy & xz \\ xy & (x^2 + z^2) & yz \\ xz & yz & (x^2 + y^2) \end{bmatrix} dx dy dz$$

où $\rho(x,y,z)$ est la densité volumique du solide.

Matrice d'inertie des solides utilisés :

$$\begin{bmatrix} \frac{2mR^2}{5} & 0 & 0 \\ 0 & \frac{2mR^2}{5} & 0 \\ 0 & 0 & \frac{2mR^2}{5} \end{bmatrix} \quad \begin{bmatrix} m \frac{b^2 + c^2}{12} & 0 & 0 \\ 0 & m \frac{a^2 + c^2}{12} & 0 \\ 0 & 0 & m \frac{a^2 + b^2}{12} \end{bmatrix}$$

Pour une sphère / Pour un cube

Équation de la cinématique

La cinématique du solide peut être représentée mathématiquement comme l'intégration sur le temps des équations différentielles liant position, vitesse et accélération.

$$\vec{P} = \frac{d\vec{V}}{dt}$$

$$\vec{v} = \frac{d\vec{a}}{dt}$$

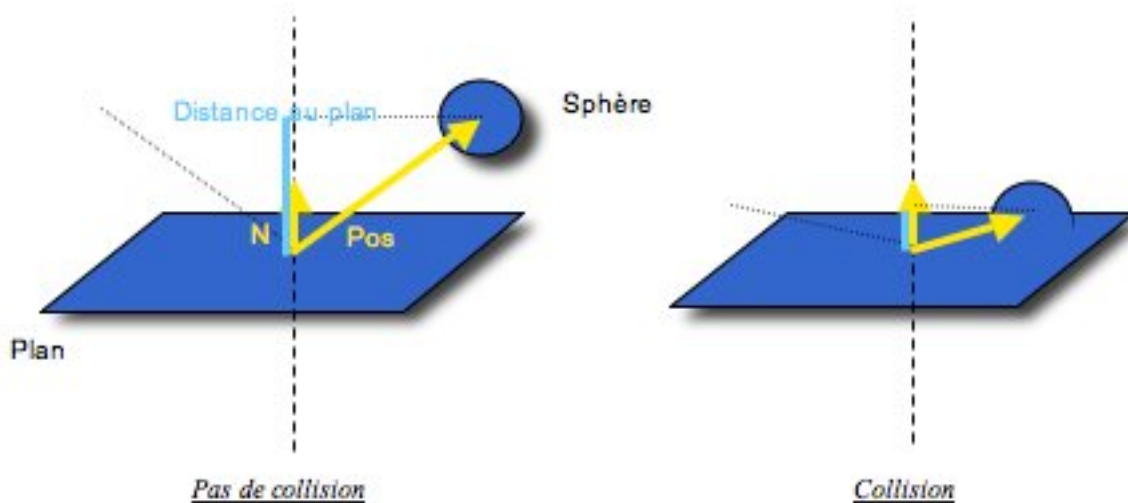
Il est également possible de définir la vitesse d'un point quelconque, P_1 , du solide : $\vec{V}_{P_1} = \vec{V}_G + \vec{GP_1} \wedge \vec{\omega}$ où $\vec{GP_1}$ est le vecteur liant P_1 au centre de masse.

Leçon 2 : Gestion d'une sphère

Collision Sphère / Plan

Détection de la collision

Une fois le calcul des nouvelles positions des objets effectué, il est nécessaire de détecter s'il collisionne avec un autre objet. Dans le cas de la collision Sphère / Plan cette détection se fait aisément en comparant la distance du centre de la sphère au plan avec le rayon de la sphère :



La condition de collision est $|\vec{P} \cdot \vec{N}| \leq R$, avec R rayon de la sphère.

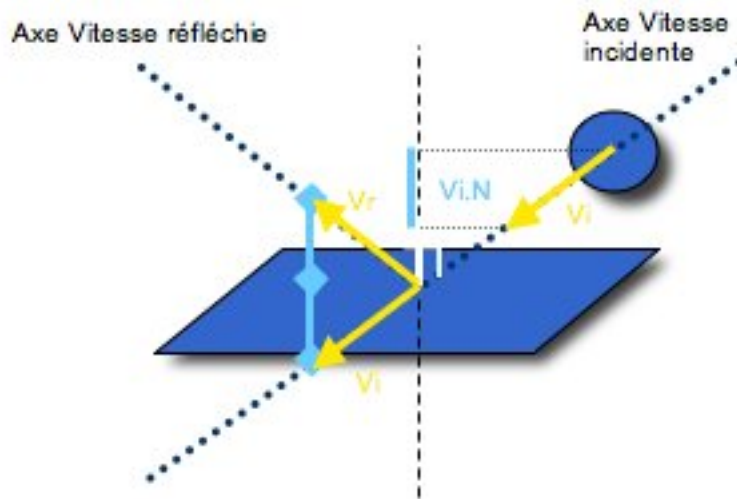
Gestion de la collision

Une première idée pour gérer la collision entre deux corps rigides est d'appliquer une force de réaction aux objets. Cependant cette force ne va pas arrêter les objets immédiatement et ne va donc pas empêcher leur interpénétration, car elle ne pourra pas changer instantanément le sens des vitesses.

Une force va mettre trop de temps à inverser les vitesses or les corps sont déjà en contact, il faut donc agir immédiatement sur leur vitesse. C'est pourquoi nous allons calculer la vitesse réfléchiée par rapport à la normale de la collision (la normale du plan dans ce cas) . Celle-ci remplacera ensuite l'ancienne vitesse.

La vitesse réfléchiée peut être obtenue grâce à la relation suivante:

$$\vec{V}_r = \vec{V}_i - 2 \times (\vec{V}_i \cdot \vec{N}) \times \vec{N}$$

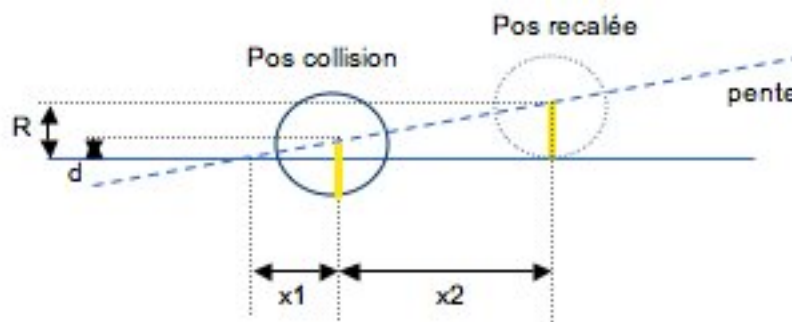


Construction du vecteur vitesse réfléchi

Recalage au point de contact exact

Du fait que le calcul des positions ne peut se réaliser de manière continue (il s'effectue à chaque frame donc tous les Δt secondes), au moment où l'on détecte la collision, les objets vont s'être déjà interpénétrés.

Il est alors nécessaire de replacer les objets au point de contact exact pour être certain qu'à la frame suivante, les deux objets se seront bien séparés. En effet si l'on ne réalise pas ce recalage, il est possible que les objets n'aient pas eu le temps de compenser l'interpénétration à la frame suivante (notamment si le Δt suivant est faible). Dès lors, le moteur va détecter de nouveau une collision etc.



Recalage de la sphère sur le plan

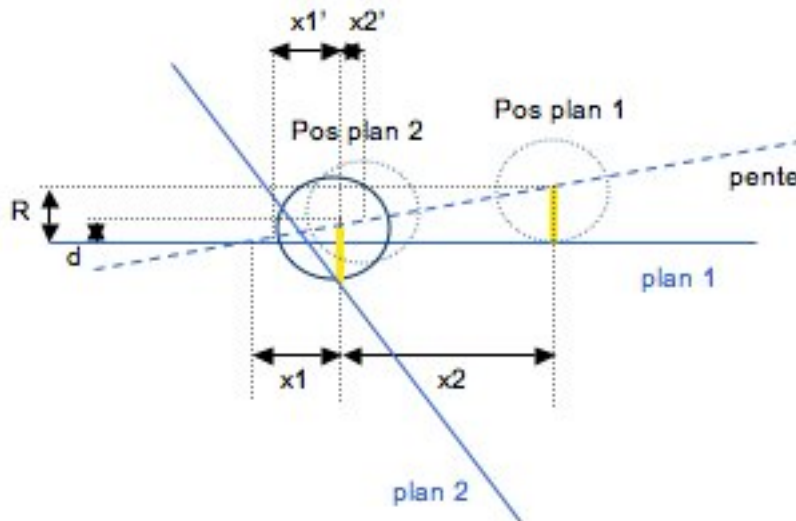
On recalc l'objet dans la direction de sa vitesse au moment où l'on détecte la collision. On cherche en fait à « remonter dans le temps » afin de trouver le point de collision exact. On calcule dans un premier temps la pente de la vitesse (représentée par des pointillés bleus sur le schéma ci-dessus).

On a alors $x_1 = d / \text{Pente}$ or, $\frac{x_1}{x_1 + x_2} = \frac{d}{R}$ d'où, $\frac{d / \text{Pente}}{d / \text{Pente} + x_2} = \frac{d}{R}$ et donc finalement,

$$x_2 = \frac{R - d}{\text{Pente}}$$

Connaissant x_2 on peut alors aisément replacer la sphère à la bonne position.

Dans le cas où la sphère entre en collision avec plusieurs objets (par exemple avec deux plans), on calcule, pour chacune des collisions, la distance x_2 nécessaire au recalage de la sphère. Finalement, on garde le plus grand x_2 pour recaler physiquement la sphère et ainsi éviter toute interpénétration avec les autres objets :



Recalage de la sphère lors d'une collision avec 2 plans

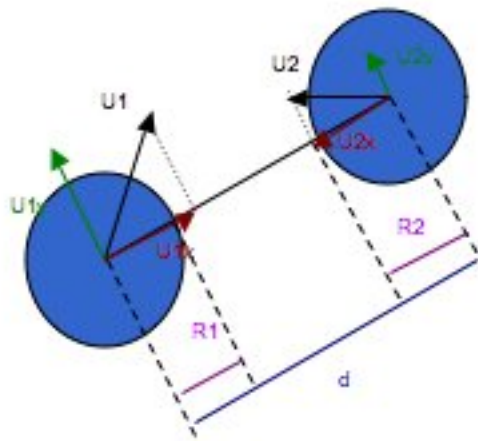
Collision Sphère / Sphère

Détection de la collision

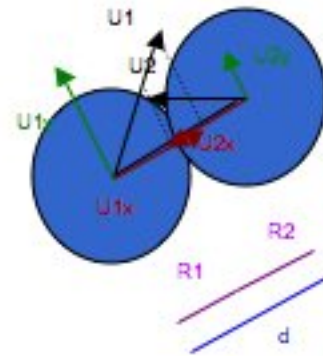
La détection de la collision Sphère / Sphère est relativement simple. Soient deux sphères :

- de centre S_1
et de rayon C_1
- de centre S_2
et de rayon C_2

On forme le vecteur $\overrightarrow{C_1 C_2}$ et l'on pose $d = |\overrightarrow{C_1 C_2}|$. Ensuite, il suffit de vérifier que $d \leq R_1 + R_2$ pour s'assurer de la collision.



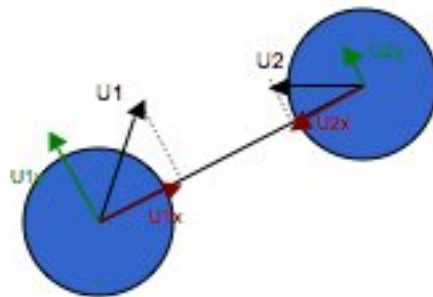
Pas de collision car $d > R_1 + R_2$



Collision car $d = R_1 + R_2$

Gestion de la collision

Notre gestion de la collision va prendre en compte l'inertie des deux objets. En effet, une boule de bowling lancée à grande vitesse ne sera pas influencée par le contact avec une balle de ping-pong.



Gestion de la collision Sphère / Sphère

Tout d'abord, on se place dans le repère local défini par l'axe de direction $\overrightarrow{C_1 C_2}$ et l'axe perpendiculaire. Seule la composante des vitesses suivant l'axe $\overrightarrow{C_1 C_2}$ sera modifiée. On donnera ici la solution sans démonstration. On pose donc :

$$\begin{cases} V_{1x} = \frac{(U_{1x} \times M_1) + (U_{2x} \times M_2) - (U_{1x} - U_{2x}) \times M_2}{M_1 + M_2} \\ V_{1y} = U_{1y} \end{cases}$$

et l'on a $\vec{V}_1 = \vec{V}_{1x} + \vec{V}_{1y}$.

De même,

$$\begin{cases} V_{2x} = \frac{(U_{1x} \times M_1) + (U_{2x} \times M_2) - (U_{2x} - U_{1x}) \times M_2}{M_1 + M_2} \\ V_{2y} = U_{2y} \end{cases}$$

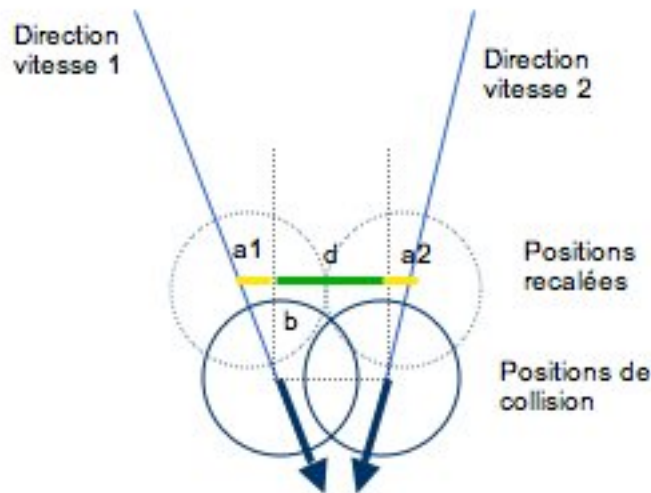
et l'on a $\vec{V}_2 = \vec{V}_{2x} + \vec{V}_{2y}$.

On remarquera dans ces formules que si les sphères sont de masses égales et si les vitesses au point de rencontre sont les mêmes alors on a $\vec{V}_{1x} = -2U_{1x}$. On retrouve alors le calcul classique d'un vecteur réfléchi par rapport à une normale (suivant $\vec{C}_2\vec{C}_1$) soit $\vec{V}_r = \vec{V}_i - 2 \times (\vec{V}_i \cdot \vec{N}) \times \vec{N}$.

Recalage au point de contact

Pour les mêmes raisons que précédemment (voir recalage Sphère / Plan) on va chercher à recaler les deux sphères.

Dans le cas de cette collision on va reculer les deux sphères suivant la direction de leur vitesse respective jusqu'au moment où elles ne possèdent plus qu'un seul point d'intersection, leur point de contact.



Calcul du recalage Sphère / Sphère

On a $\begin{cases} b/a_1 = p_1 \\ b/a_2 = p_2 \end{cases}$ avec p_1 (resp. p_2), pente de la vitesse de l'objet 1 (resp. 2). et de plus il faut que $a_1 + a_2 + d = R_1 + R_2$ pour être au point exact de contact. Donc,

$$\begin{cases} a_1 \times p_1 = a_2 \times p_2 \\ a_1 + a_2 + d = R_1 + R_2 \end{cases}$$

On a donc un système à deux équations et deux inconnus que l'on peut résoudre aisément par les méthodes classiques. On arrive alors à

$$a_1 = \frac{p_2 \times (R_1 + R_2 - d)}{p_1 + p_2}$$

$$a_2 = \frac{p_1 \times (R_1 + R_2 - d)}{p_1 + p_2}$$

$$b = \frac{p_1 p_2 \times (R_1 + R_2 - d)}{p_1 + p_2}$$

il faudra donc reculer la sphère 1 de $\sqrt{a_1^2 + b^2}$ et la sphère 2 de $\sqrt{a_2^2 + b^2}$.

 *Rem : Suite à la présentation orale du projet, une question nous a permis de voir une limite dans cette méthode. En effet, la variable « b », posée ainsi, suppose que les vitesses sont coplanaires, ce qui n'est pas forcément vrai.*

Il faudrait donc trouver la démonstration dans le cas général, en distinguant les variables b_1 et b_2 .

Leçon 3 : Gestion d'une box - Partie 1/2

Dans un second temps, une fois l'architecture du moteur établi et la gestion des sphères implémentée, nous avons décidé de nous intéresser à la gestion d'un objet qui paraît simple mais dont la dynamique est assez complexe : le cube.

La collision Boîte / Plan

La détection de la collision

Soient une boîte B_1 , de centre C_1 , de dimensions (a_1, a_2, a_3) suivant les axes de son repère local $(\vec{R}_1, \vec{R}_2, \vec{R}_3)$ et un plan, de centre C_2 et de normale $\vec{N}$.

On calcule tout d'abord la distance du centre de la boîte au plan. En projetant C_1 sur la normale, on a $d = \overrightarrow{C_2 C_1} \cdot \vec{N}$.

Ensuite, on va chercher la contribution de chaque côté du cube suivant la normale. Pour cela, on projette chaque axe du repère local sur la normale que l'on multiplie par les dimensions de la boîte.

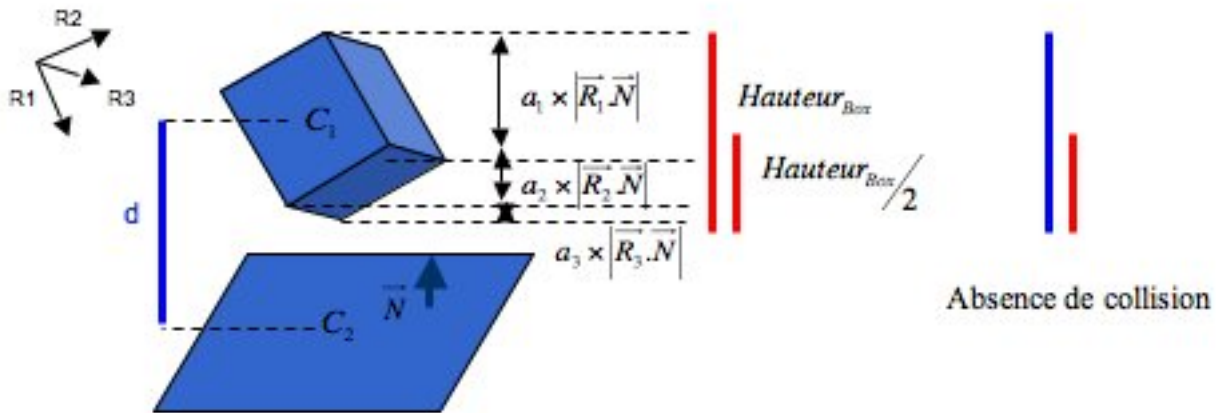
$$Hauteur_{Box} = a_1 \times |\vec{R}_1 \cdot \vec{N}| + a_2 \times |\vec{R}_2 \cdot \vec{N}| + a_3 \times |\vec{R}_3 \cdot \vec{N}|$$

ou encore,

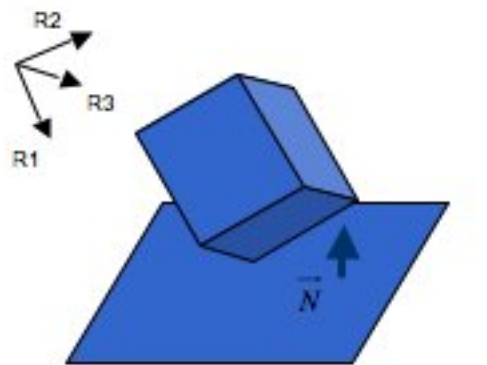
$$Hauteur_{Box} = \sum_{i=1}^3 a_i |\vec{R}_i \cdot \vec{N}|$$

Il n'y a plus qu'à calculer la différence entre la distance séparant les deux centres et la demi-hauteur du cube.

On vérifie donc le signe de l'expression, $d = \frac{\text{Hauteur}_{\text{Boite}}}{2}$.



Détection de la collision boîte / plan



Limite de collision boîte / plan

Gestion de la collision

Les choses commencent à se compliquer ici. En effet, pour la gestion des collisions des sphères, seule la vitesse linéaire était à gérer. En faisant intervenir les masses, on calculait la vitesse réfléchi.

Avec les boîtes, un nouveau paramètre rentre en compte: la vitesse angulaire. Dès lors, en plus de la vitesse linéaire, il faut calculer la réponse agissant sur la vitesse angulaire. Nous avons utilisé la théorie de l'impulsion qui est applicable à tout type de collision.

Pour la collision de deux corps rigides, on suppose que le processus de déformation microscopique consiste en deux phases : la phase de compression et la période de restitution. La phase de compression s'étend de l'instant du contact jusqu'à l'instant où la vitesse devient nulle. La restitution commence alors et prend fin lors de la séparation des corps. La durée de contact étant supposée très courte et les forces très grandes, on considère que :

- Le processus de collision est instantané et le changement de vitesse est discontinu
- Il n'y a pas de déplacement pendant l'impulsion
- Les forces qui interagissent sont des impulsions et les toutes les autres forces finies sont négligeables.

Théorie de l'impulsion

Considérons deux objets, O_1 et O_2 , qui viennent tout juste de collisionner en un point P. Lorsque les deux corps entrent en collision, une impulsion normale P_y et une impulsion tangentielle P_x sont générées au point de contact.

Nous allons distinguer deux repères correspondant aux repères locaux de chaque objet. On considère que la probabilité d'avoir un contact de type sommet/sommet est nulle. On pose plusieurs variables :

- Soit

$$\vec{V}_0 \begin{pmatrix} \dot{x}_{i_0} \\ \dot{y}_{i_0} \end{pmatrix}$$

la vitesse linéaire initiale de l'objet

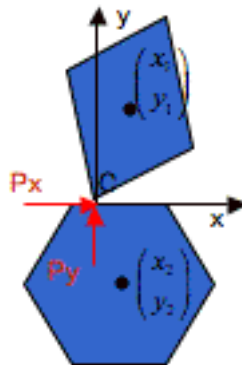
O_1

- Soit

$$\omega_{i_0}$$

la vitesse angulaire initiale de l'objet

O_1



Pour l'objet O_1 , on a les relations suivantes :

- L'impulsion correspond à la différence de quantité de mouvement. En considérant chaque composante, on a

$$\begin{cases} m_1 \times (\dot{x}_1 - \dot{x}_{i_0}) = P_x \\ m_1 \times (\dot{y}_1 - \dot{y}_{i_0}) = P_y \end{cases}$$

- La seconde relation fait appel au théorème des moments d'inertie,

$$I \times \frac{d\omega}{dt} = \sum m_i$$

. En appliquant cette relation à l'objet

O_1

, on a :

$$I_1 \times (\omega_1 - \omega_{i_0}) = I_{x_1}^P \cdot P_y - I_{y_1}^P \cdot P_x$$

- En appliquant cette relation à l'objet

$$O_1$$

, on a :

$$\vec{V}_{C_1} = \vec{V}_{O_1} + \vec{\omega}_1 \times \vec{r}_{C_1/O_1}$$

- Similairement, la vitesse du point de collision C suivant l'objet

$$O_2$$

est

$$\vec{V}_{C_2} = \begin{cases} \dot{x}_{C_2} = \dot{x}_2 + \omega_2 \cdot y_2 \\ \dot{y}_{C_2} = \dot{y}_2 - \omega_2 \cdot x_2 \end{cases}$$

On définit alors deux grandeurs :

- La vitesse de glissement: elle correspond à la composante tangentielle de la vitesse relative des points de contact.

$$S = \dot{x}_{C_1} - \dot{x}_{C_2}$$

- La vitesse de compression: Elle correspond à la composante normale de la vitesse relative des points de contact.

$$C = \dot{y}_{C_1} - \dot{y}_{C_2}$$

- En combinant les équations précédentes, on obtient le système

$$\begin{cases} S = S_0 + B_1 P_x - B_3 P_y \\ C = C_0 - B_3 P_x + B_2 P_y \end{cases}$$

$$B_1 = \frac{1}{m_1} + \frac{1}{m_2} + \frac{y_1^2}{I_1} + \frac{y_2^2}{I_2} \quad B_3 = \frac{x_1 y_1}{I_1} + \frac{x_2 y_2}{I_2}$$

$$B_2 = \frac{1}{m_1} + \frac{1}{m_2} + \frac{x_1^2}{I_1} + \frac{x_2^2}{I_2} \quad S_0 = \dot{x}_{C_{10}} - \dot{x}_{C_{20}}$$

$$\text{En posant } C_0 = \dot{y}_{C_{10}} - \dot{y}_{C_{20}} \quad \text{et}$$

Classification du mouvement

Une fois tous ces coefficients calculés, il faut classer le mouvement pour définir le bon type d'impulsion.

La résolution de ce système se fera par la technique graphique de Routh. Cette technique permet de déterminer des modes de contacts, ainsi que les formules à utiliser pour chacun des modes.

On pose les coefficients suivants :

$$\begin{aligned} \mu_s &= -\frac{B_3}{B_1} \\ P_d &= (B_2 + s \mu B_3) \times s S_0 \\ P_q &= (\mu B_1 + s B_3) \times (-C_0) \end{aligned} \quad \text{avec } s = \begin{cases} \frac{S_0}{|S_0|} & \text{Si } S_0 \neq 0 \\ 1 & \text{sinon} \end{cases}$$

et μ étant le coefficient frottement du matériau

En fonction de ces valeurs, nous allons définir 5 cas

	$\mu > \mu_s $	$\mu \leq \mu_s $
$P_d > (1+e)P_q$	Glissement	Glissement
$P_q < P_d < (1+e)P_q$	R-adhérence	R-glissement vers l'arrière
$P_d < P_q$	C-adhérence	C-glissement vers l'arrière

Méthode de Poisson

Nous donnons ici directement les résultats.

- Glissement

$$\begin{aligned} P_x &= -s\mu P_y \\ P_y &= -(1+e) \times \frac{C_0}{B_2 + s\mu B_3} \end{aligned}$$

- C-adhérence

$$\begin{aligned} P_x &= \frac{B_3 P_y - S_0}{B_1} \\ P_y &= -(1+e) \times \frac{B_1 C_0 + B_3 S_0}{B_1 B_2 - B_3^2} \end{aligned}$$

- R-adhérence

$$\begin{aligned} P_x &= \frac{B_3 P_y - S_0}{B_1} \\ P_y &= -(1+e) \times \frac{C_0}{B_2 + s\mu B_3} \end{aligned}$$

- C-glissement vers l'arrière

$$\begin{aligned} P_x &= s\mu \times \left[P_y - \frac{2S_0}{B_3 + s\mu B_1} \right] \\ P_y &= -\frac{1+e}{B_2 - s\mu B_3} \times \left[C_0 + \frac{2s\mu B_3 S_0}{B_3 + s\mu B_1} \right] \end{aligned}$$

- R-glissement vers l'arrière

$$\begin{aligned} P_x &= s\mu \times \left[P_y - \frac{2S_0}{B_3 + s\mu B_1} \right] \\ P_y &= -(1+e) \times \frac{C_0}{B_2 + s\mu B_3} \end{aligned}$$

Leçon 4 : Gestion d'une box - Partie 2/2

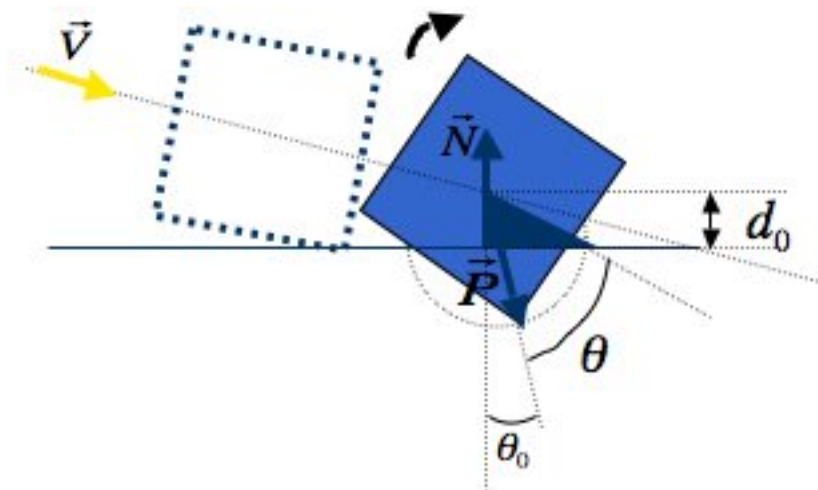
Application à la collision Box / Plan

Une fois le point d'intersection trouvé, il suffit d'appliquer ces formules pour calculer l'impulsion, en considérant que le plan n'a ni vitesse linéaire, ni vitesse angulaire. De plus on considère sa masse comme étant infinie et de même pour son moment d'inertie.

Les résultats obtenus avec ce calcul de l'impulsion sont très satisfaisants. Nous avons commencé cette implémentation pour la collision Box/Plan mais comme ce calcul est totalement générique, nous l'avons appliqué pour tous les autres types de collision. Cependant, à cause d'un manque de temps, nous n'avons pas modifiés les méthodes de gestion d'intersection que nous avons déjà implémentées.

Recalage au point de collision.

Pour le cas de la collision Boite/Plan, nous allons devoir « remonter dans le temps » pour retrouver le moment où la boîte avait exactement un seul point de contact avec le plan. Une difficulté supplémentaire apparaît alors : non seulement la boîte s'est translattée, mais elle a également tourné en même temps.



Recalage d'une boîte après collision avec un plan

Mise en équation du problème

On appelle « d » la distance du centre de la boîte au plan. d_0 est la distance initiale.

- $\vec{P}$ représente les coordonnées du point de contact dans le repère local de la boîte.
- θ_0 représente l'angle initial entre le point de contact et la normale et θ mesure la variation de cet angle au cours du temps.

$$\cos(\theta_0) = \frac{d_0}{|\vec{P}|}$$

On a dans le triangle colorié en bleu :

$$\cos(\theta_0 + \theta) = \frac{d}{|\vec{P}|}$$

Plus généralement lorsqu'on « remonte dans le temps », on a :

Or, $\vec{d} = d_0 - (\vec{V} \vec{N}) \times \Delta t$ et $\vec{P} = \vec{P}_0 + \vec{\omega} \times \vec{r}$ avec ω vitesse angulaire de la boîte. Donc,

$$\cos(\theta_0 + \omega \times \Delta t) = \frac{d_0 - (\vec{V} \vec{N}) \times \Delta t}{|\vec{P}|}$$

On cherche alors à trouver le Δt .

Méthodes de résolution

Notre équation nous ramène donc à chercher le point d'intersection entre un cosinus et une droite, en effet notre équation peut se mettre sous la forme $\cos(\theta_0 + \omega \Delta t) = a\Delta t + b$ avec a, b, ω et θ_0 positifs.

Pour résoudre cette équation nous avons pensé à deux possibilités:

- Développement limité du cosinus

En réalisant un développement limité du cosinus nous nous ramèrions à une équation du second degré et nous pourrions ainsi en tirer une expression analytique pour Δt :

$$\Delta t = \frac{\sqrt{a^2 + 2a\omega\theta_0 - 2(b-1)\omega^2} + a + \omega\theta_0}{\omega^2}$$

Cependant le développement limité n'est valable qu'au voisinage de 0, on risque alors d'avoir une imprécision sur Δt dès que l'angle formé par $\theta_0 + \omega \Delta t$ devient grand.

- Dichotomie

C'est la méthode que nous avons choisi d'implémenter. Nous avons remarqué que l'angle $\theta_0 + \omega \Delta t$ est compris entre 0 et $\pi/2$. On a donc une première indication sur un intervalle dans lequel nous pourrions trouver Δt : $0 < \Delta t < (\pi/2 - \theta_0)/\omega$. Ceci constituera notre intervalle de départ pour la dichotomie. De plus, on peut donc en déduire que le cosinus de $\theta_0 + \omega \Delta t$ sera décroissant sur l'intervalle considéré et sera compris entre 0 et $\cos(\theta_0)$.

Or a et b sont positif, donc la droite elle est croissante. De plus, en $\Delta t=0$ le membre de gauche vaut $\cos(\theta_0)$ et le membre de droite vaut b .

Or $\cos(\theta_0) = \frac{|\vec{P} \vec{N}|}{|\vec{P}|}$ et $b = \frac{d_0}{|\vec{P}|}$ avec $|\vec{P} \vec{N}| > d_0$ donc $\cos(\theta_0) > b$.

Il y a donc bien une solution et une seule à notre problème.

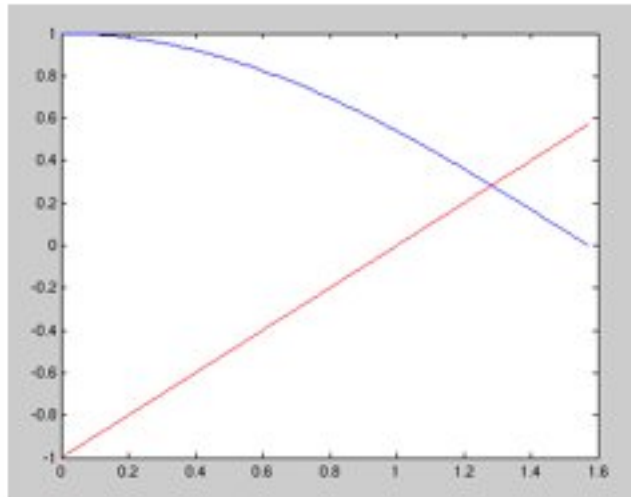


Illustration du problème : recherche du point d'intersection cosinus-droite

On va alors pouvoir procéder à la recherche de Δt par dichotomie en évaluant, $\cos(\omega \Delta t + \theta_0) - (a \Delta t + b)$ et en rétrécissant l'intervalle par le bas si la quantité est positive et par le haut si elle est négative. On a estimé qu'en 10 itérations nous avons une bonne estimation de Δt .

Connaissant Δt , par l'une ou l'autre des méthodes, on peut alors recalculer notre boîte sans difficulté.

La collision Box / Sphère

Détection de la collision

La détection est faite en utilisant tout ce qui a été vu précédemment. On calcule les contributions des axes du cube par projection sur l'axe liant les deux centres de gravité des objets.

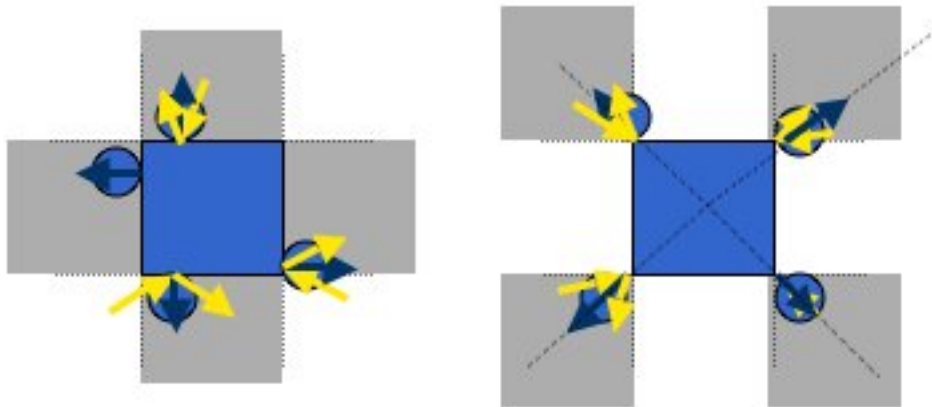
Soient une boîte B_1 , de centre C_1 , de dimensions (a_1, a_2, a_3) suivant les axes de son repère local $(\vec{R}_1, \vec{R}_2, \vec{R}_3)$ et une sphère S_1 de centre C_2 et de rayon R_1 .

Une étape supplémentaire est nécessaire ici pour se servir correctement du calcul de l'impulsion. Il s'agit de calculer la normale au point de collision. En effet, deux cas de figures sont possibles :

- La collision se produit sur la face du cube. Dans ce cas, la normale de collision est égale à la normale de la face du cube.
- La collision a lieu sur un sommet du cube. Dans ce cas, la normale de collision est égale au vecteur

$$\vec{C_1 C_2}$$

normalisé.



Représentation de la normale dans la collision Boîte / Sphère

On pose $d = |\overline{C_1 C_2}|$. Ensuite on a toujours $Hauteur_{Box} = \sum_{i=1}^3 a_i |\overline{R_i} \cdot \overline{N}|$. La condition de la collision est alors :

$$d \leq \frac{Hauteur_{Box}}{2} + R_i$$

Gestion de la collision

Nous calculons l'impulsion en considérant que la sphère n'a pas de vitesse angulaire et que le moment d'inertie est infini.

Collision Boîte / Boîte

Détection de la collision Boîte / Boîte

Pour détecter une collision entre deux boîtes, on s'appuie sur le théorème de l'axe séparateur. Si deux boîtes ne sont pas en contact alors il existe un axe séparateur $\overline{L} = \overline{M} \wedge \overline{N}$, où $\overline{M}$ et $\overline{N}$ sont des vecteurs disjoints choisis parmi les 6 axes des boîtes.

Un axe est dit séparateur s'il existe un plan orthogonal à celui-ci qui sépare les deux boîtes et que cet axe soit orthogonal à la face d'une des deux boîtes ou orthogonal à deux cotés de chaque boîte.

Ceci conduit à tester les 15 axes suivants :

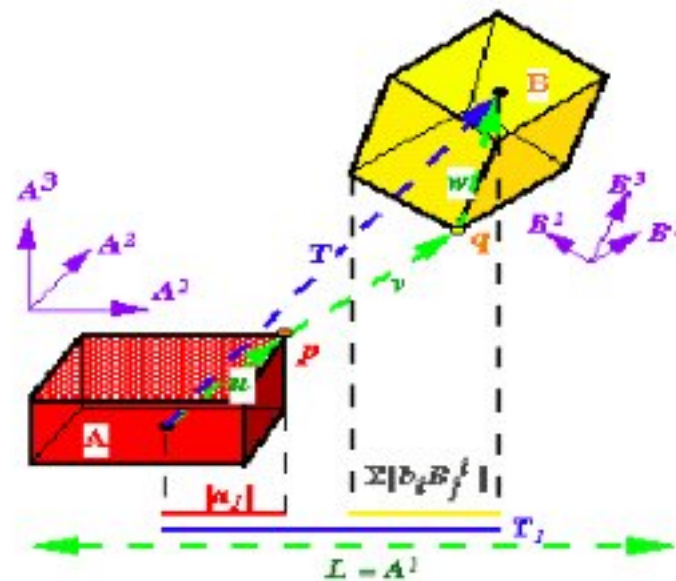
Collision face avec un sommet

- Normale face1 objet 1
- Normale face2 objet 1
- Normale face3 objet 1
- Normale face1 objet 2
- Normale face2 objet 2
- Normale face3 objet 2

Collision arrête avec une autre arrête

- Normale face1 objet1 ^ Normale face1 objet2
- Normale face1 objet1 ^ Normale face2 objet2
- Normale face1 objet1 ^ Normale face3 objet2
- Normale face2 objet1 ^ Normale face1 objet2
- Normale face2 objet1 ^ Normale face2 objet2
- Normale face2 objet1 ^ Normale face3 objet2
- Normale face3 objet1 ^ Normale face1 objet2
- Normale face3 objet1 ^ Normale face2 objet2
- Normale face3 objet1 ^ Normale face3 objet2

Le test s'effectue simplement par projection des boîtes sur les axes:



Représentation de la détection Boîte / Boîte

Ainsi on peut montrer que L est un axe séparateur s'il satisfait la condition suivante:

$$|\vec{T} \cdot \vec{L}| > \sum_i |a_i \vec{A}_i \cdot \vec{L}| + \sum_i |b_i \vec{B}_i \cdot \vec{L}|$$

avec a_i et b_i les dimensions des cubes.

L'algorithme s'arrête dès la détection du premier axe, séparateur il n'y a alors pas de collision. Si pour aucun des 15 axes cette condition n'est vérifiée alors il y a collision.

Gestion de la collision Boîte / Boîte

Nous utilisons la technique décrite dans le chapitre Gestion de la collision Boîte / Plan, généralisée à deux objets.

La normale de la collision est alors définie comme étant l'axe séparateur trouvé précédemment.

Pour trouver les coordonnées du point de contact nous séparons les cas :

- Collision entre un sommet et une face
- Collision entre deux arrêtes

Collision sommet/face:

On cherche le sommet de la boîte en collision, pour cela on se base sur le produit scalaire entre les axes de cette boîte et la normale:

Implémentation collision sommet / face
$signe = \frac{\vec{A_1} \cdot \vec{N}}{ \vec{A_1} \cdot \vec{N} }$
avec
$\vec{N}$
orienté vers la boîte dont le sommet est en collision.
$PointContact = -dim_1 \times signe \times \vec{A_1}$
$signe = \frac{\vec{A_2} \cdot \vec{N}}{ \vec{A_2} \cdot \vec{N} }$
$PointContact += -dim_2 \times signe \times \vec{A_2}$
$signe = \frac{\vec{A_3} \cdot \vec{N}}{ \vec{A_3} \cdot \vec{N} }$
$PointContact += -dim_3 \times signe \times \vec{A_3}$
PointContact = PointContact + PositionBoite //Conversion des coordonnées locales en coordonnées globales

Collision arrête/arrête:

De la même manière on va chercher le point de contact dans le repère locale d'une des boîtes puis le convertir en coordonnées globales. Par définition la normale est orthogonale à un des axes de la boîte. Pour les deux autres axes on procédera de la même manière que précédemment.

Implémentation collision arrête / arrête
If(
$\vec{A_1} \cdot \vec{N}$
== 0)
$signe = \frac{\vec{A_1} \cdot \vec{N}}{ \vec{A_1} \cdot \vec{N} }$
avec
$\vec{N}$
orienté vers la boîte dont le sommet est en collision.

```

                                 $PointContact = -dim_1 \times signe \times \vec{A_1}$ 
else
                                 $PointContact = \vec{P} \vec{A_1}$ 
//avec P vecteur liant le centre des deux boites
endif
// -----
If(
                                 $\vec{A_2} \vec{N}$ 
== 0 )
                                 $signe = \frac{\vec{A_2} \vec{N}}{|\vec{A_2} \vec{N}|}$ 
                                 $PointContact += -dim_2 \times signe \times \vec{A_2}$ 
else
                                 $PointContact += \vec{P} \vec{A_2}$ 
endif
// -----
If(
                                 $\vec{A_3} \vec{N}$ 
== 0 )
                                 $signe = \frac{\vec{A_3} \vec{N}}{|\vec{A_3} \vec{N}|}$ 
                                 $PointContact += -dim_3 \times signe \times \vec{A_3}$ 
else
                                 $PointContact += \vec{P} \vec{A_3}$ 
endif
// -----
PointContact = PointContact + PositionBoite //Conversion des coordonnées locales en
coordonnées globales

```

Architecture

Dans cette partie, nous allons rapidement détailler les classes et leurs relations. L'affichage des objets se fait via l'API OpenGL.

La classe Objet

La classe objet est la classe mère des toutes les primitives graphiques. Elle possède les paramètres communs à tous les objets :

- Masse
- Position, Vitesses, Accélération
- Matériau
- Nombre de collisions

Concernant les méthodes, cette classe possède principalement des accesseurs. Cependant, la méthode d'affichage est une méthode virtuelle pure, ce qui rend cette classe abstraite.

La classe Sphère

Cette classe hérite de la classe objet. Elle définit les paramètres propres à la sphère. Elle surcharge la méthode d'affichage, on soulignera ici la méthode pour texturer une sphère.

```
glPushMatrix();  
  
GLUQuadric* params = gluNewQuadric(); //création d'une quadrique  
  
gluQuadricTexture(params, GL_TRUE); //activation des coordonnées de texture  
  
glTranslatef( m_vPos.x, m_vPos.y, m_vPos.z);  
gluSphere( params, m_fRayon, 30, 30);  
  
glPopMatrix();  
  
gluDeleteQuadric(params); //je supprime la quadrique
```

Il ne faut en effet pas passer par la fonction GLUT, mais plutôt par la création d'une quadrique.

La classe Plan

A partir des 3 points du plan fournis au constructeur, on calcule le dernier point et la méthode d'affichage affiche un simple quad texturé.

La classe Box

Une box est une boîte qui possède son propre repère local. Ainsi, il suffit de modifier les axes pour changer l'orientation de la boîte. On peut alors se passer de toutes fonctions de transformation d'OpenGL.

La méthode d'affichage se résume à afficher de 6 quads texturés.

Le fichier PhysicEngine.cpp

Il s'agit du point d'entrée du programme. Les méthodes OpenGL y sont définies ainsi que la scène et la caméra.

La boucle principale permet de passer le temps écoulé depuis la dernière frame à la scène. De plus, le parseur de scène se trouve ici et est exécuté lors de l'initialisation d'OpenGL. Au niveau des textures, elles sont chargées à ce moment-là et l'index de la texture est stocké dans le matériau.

La classe Scène

Il s'agit du cœur du moteur. Elle contient trois objets du type list de la Standard Library :

- La première contient tous les objets
- La seconde tous les matériaux

- La dernière répertorie toutes les intersections de la frame courante

Dans cette classe sont définies les méthodes d'intersection (détection) et de calcul de la réponse (gestion). De plus, à chaque frame, la scène met à jour la dynamique des objets.

La classe Image

Cette classe associée au fichier tga.cpp permet de lire et de charger les textures de format *.tga via l'objet Image_Manager.

La classe Matériau

Elle conserve l'index de la texture chargé par OpenGL. Les coefficients de frottements et de glissement permettent de modifier la réponse de l'objet.

Conclusion

Voici les quelques améliorations que nous pourrions apporter au projet :

- Amélioration des techniques de recalage. En effet nous n'avons pas eu le temps nécessaire afin de trouver un algorithme du recalage lors d'une collision entre sphère et boîte et entre boîte et boîte. De plus l'implémentation du recalage lors d'un contact boîte / plan est à améliorer. En effet, quelques problèmes surviennent parfois, notamment lorsque la vitesse linéaire devient trop tangentielle au plan.
- Amélioration du traitement des collisions arrête / arrête entre deux boîtes. En effet nous avons remarqué que parfois le calcul du point de contact paraît erroné. De plus, on ne trouve pas le même point si l'on considère l'une ou l'autre boîte.
- Nous pourrions également modifier la classe sphère afin d'ajouter un mouvement de rotation aux sphères. En effet dans l'état les sphères ne font que se translater et ne tournent pas.
- Ajouter une force de réaction dans le cas de collision entre une boîte et une sphère ou entre deux boîtes lorsque les 2 objets sont à l'arrêt. En effet sans cela si l'on pose un objet sur un autre sans vitesse, l'impulsion calculée va être nulle et l'objet du dessus va « entrer » dans l'objet du dessous, il faudrait alors à ce moment appliquer une force de réaction.
- Enfin une optimisation du code source nous permettrait de gagner du temps et ainsi nous permettrait de pouvoir traiter des scènes plus importantes.

Bibliographie

-  <http://roso.epfl.ch/dm/pdf/CHAP3.PDF>
-  http://fr.wikipedia.org/wiki/Moment_%28m%C3%A9canique%29
-  <http://www.yov408.com/tutorials/gdmphys3.pdf>
-  http://www.jdotec.net/s3i/Mecanique/Cinematique/Cinematique_du_contact_ponctuel.php
-  http://www.jdotec.net/s3i/Mecanique/ActionsMeca/Lois_du_frottement.php
-  <http://www.sciences-indus-cpge.apinc.org/Matrice-d-inertie-des-solides>
-  <http://graphics.stanford.edu/courses/cs448-01-spring/papers/moore.pdf>
-  <http://www.euclideanspace.com/physics/dynamics/collision/index.htm>
-  <http://www-evasion.imag.fr/Membres/Francois.Faure/ascollisions/divers/dea-romain-rodriguez.pdf>
-  <http://mustapha.bismi.free.fr/articles/obb.pdf>
-  <http://www.librecours.org/cgi-bin/domain?callback=info&elt=78>
-  <http://www.sciences.univ-nantes.fr/physique/perso/blanquet/synophys/13mesol/13mesol.htm>

-  http://perso.orange.fr/physique.chimie/TS_Physique/Physique_12_CHUTE_VERTICALE_D_UN_SOLIDE.htm
-  <ftp://ftp.inria.fr/INRIA/publication/publi-pdf/RR/RR-5401.pdf>