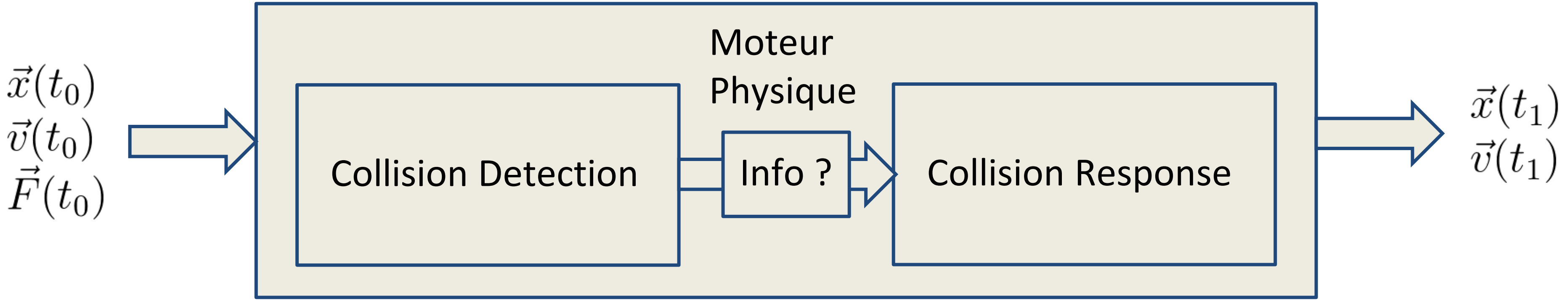
Cours Moteur Physique - Partie II

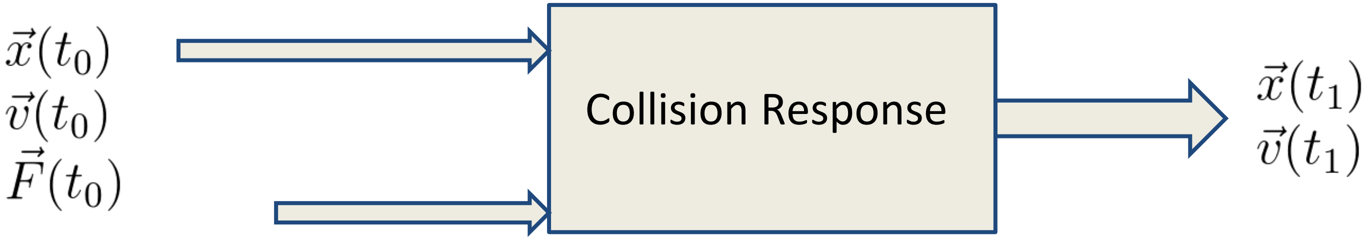
Maël Addoum

A chaque frame (tous les Rigidbody)

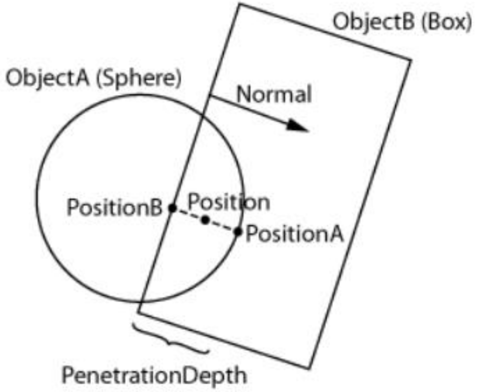



```
void PhysicUpdate(float deltaTime)
{
		CollisionDetection();
		CollisionResponse(deltaTime);
}
```

Réponse




```
struct CollisionPair
{
		Body *A, *B;
		Vec3		point;
		Vec3		normal;
		float		penetration;
}
```



Rôles de la réponse

1. Modifie les vitesses des solides lors d’un contact
2. Intègre les vitesses (2nde loi de Newton + Euler semi-implicite) en fonction des forces en présence (Gravité, friction, …)

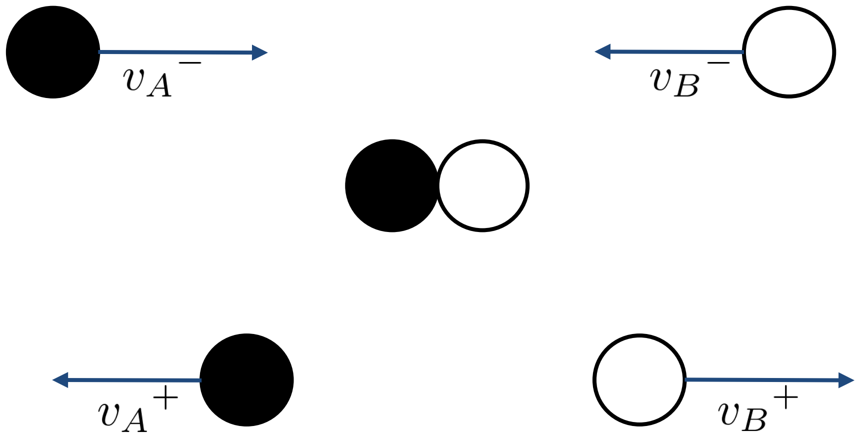
Réaction à une collision

1. Solides déplacés pour enlever inter-pénétration
2. Vitesses des solides mis à jour (rebond) pour pas qu’il y ait d’inter-pénétration la frame suivante

Réaction à une collision

1.Solides déplacés pour enlever inter-pénétration

2.Vitesses des solides mis à jour (rebond) pour pas qu'il y ait d'inter-pénétration la frame suivante



Comment alors calculer v_A^+ et v_B^+ ??

L’impulsion représente la quantité de mouvement transmise à un objet.

$$\vec{F} = \frac{\vec{J}}{\Delta t}$$

La quantité de mouvement : *p* = *m V*

$\vec{J}$ représente l'impulsion transmise $(\mathrm{N} \cdot \mathrm{s})$

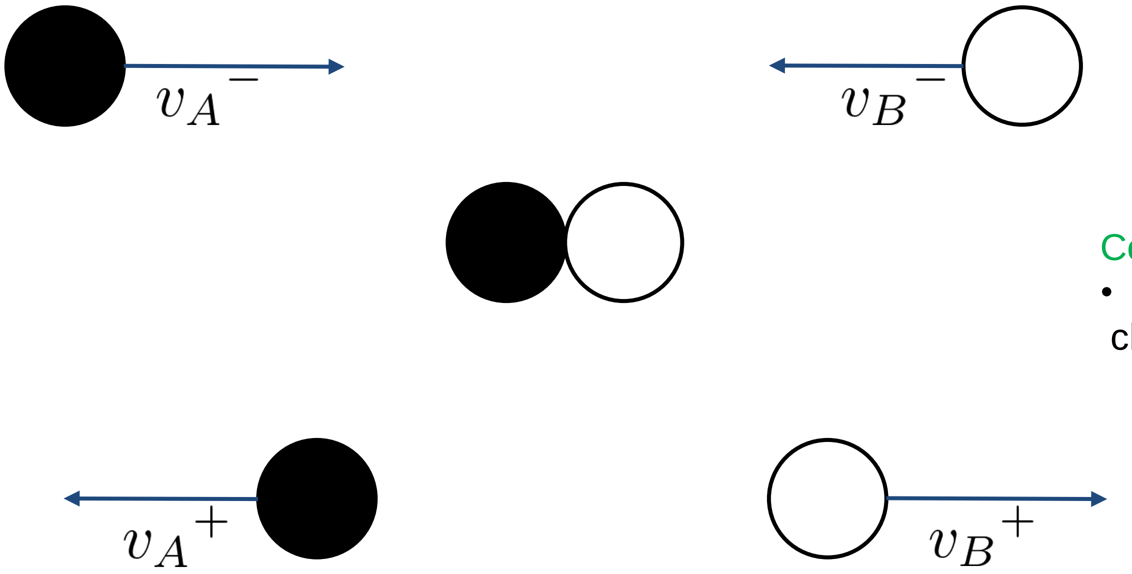
$\vec{F}$ représente la force (N)

$\triangle t$ représente le temps (s)

La vitesse après l’impact :

$$v^+ = v^- + \Delta v = v^- + \vec{a}\Delta t = v^- + \frac{\vec{F}}{m}\Delta t = v^- + \frac{\vec{J}}{m}$$

Collision élastique



Conservation de la quantité de mouvement

- Les particules ne se déforment pas ni changent de nature

Collision élastique

- Comment calculer $v_A{}^{+}, v_B{}^{+}$?

Collision élastique

- Comment calculer $v_A{}^+, v_B{}^+$?
- 3ème loi de Newton :

$$\text{Si A exerce une force totale } \vec{F} \text{ sur B, alors B exerce une force totale } -\vec{F} \text{ sur A}$$

Collision élastique

- Comment calculer $v_A{}^+, v_B{}^+$?
- 3ème loi de Newton :

Si A exerce une force totale $\vec{F}$ sur B, alors B exerce une force totale $-\vec{F}$ sur A

$$v_A^+ = v_A^- + \frac{J}{m_A}$$
$$v_B^+ = v_B^- - \frac{J}{m_B}$$

Type de Collisions

- Collision **100% inélastique** (énergie absorbée, pas de rebond, les objets restent collés)

$$v_A^+ - v_B^+ = 0$$

Type de Collisions

- Collision 100% inélastique (énergie absorbée, pas de rebond, les objets restent collés)

$$v_A^+ - v_B^+ = 0$$

- Collision 100% élastique

$$v_A^+ - v_B^+ = -(v_A^- - v_B^-)$$

Collision inélastique

- Cas général

$$v_A^+ - v_B^+ = -\epsilon(v_A^- - v_B^-)$$

- ϵ : Coefficient de restitution dans [0,1], bounciness (propriété du material physic)

Dû à une dissipation d'énergie cinétique au moment de l'impact.

une partie de l'énergie cinétique s'est convertie en énergie interne (échauffement et déformation).

Collision inélastique

- Impulsion en cas de collision

$$J = \frac{-(\epsilon + 1)(v_A^- - v_B^-)}{\frac{1}{m_A} + \frac{1}{m_B}}$$

$$v_A^+ = v_A^- + \frac{J}{m_A}$$

$$v_B^+ = v_B^- - \frac{J}{m_B}$$

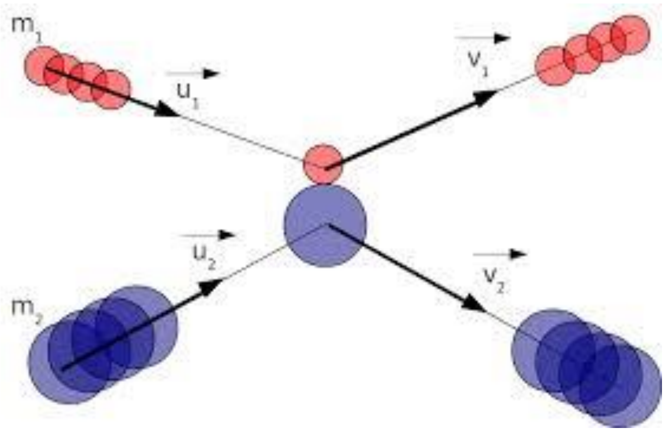
Collision inélastique

- Impulsion en cas de collision

$$J = \frac{-(\epsilon + 1)((\overrightarrow{v_A^-} - \overrightarrow{v_B^-})|\vec{n})}{\frac{1}{m_A} + \frac{1}{m_B}}$$

$$\overrightarrow{v_A^+} = \overrightarrow{v_A^-} + \frac{J}{m_A}\vec{n}$$

$$\overrightarrow{v_B^+} = \overrightarrow{v_B^-} - \frac{J}{m_B}\vec{n}$$



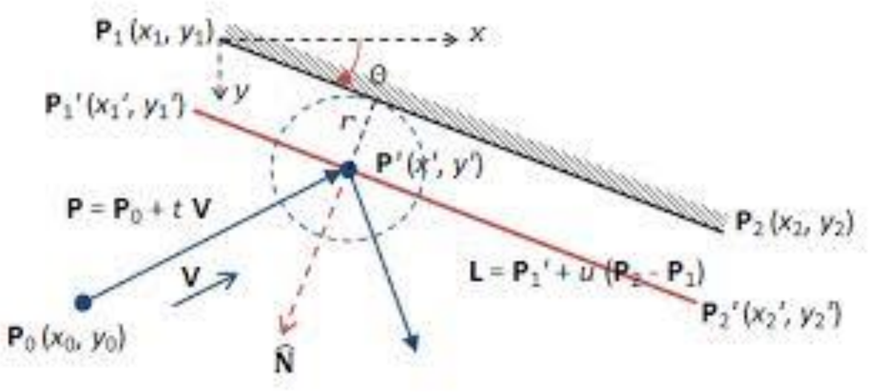
Collision élastique

- Cas particulier, B est un mur, restitution max : $\epsilon = 1, m_B = \infty, \frac{1}{m_B} = 0, \overrightarrow{v_B^-} = \vec{0}$

$$J = \frac{-2(\overrightarrow{v_A^-}|\vec{n})}{\frac{1}{m_A}} = -2(\overrightarrow{v_A^-}|\vec{n})m_A$$

$$\overrightarrow{v_A^+} = \overrightarrow{v_A^-} - 2(\overrightarrow{v_A^-}|\vec{n})$$

$$\overrightarrow{v_B^+} = \overrightarrow{v_B^-}$$



Collision élastique

- On ne peut pas représenter l’infini en flottant
- Donc on n’utilise pas la masse en programmation physique mais la masse inverse (1/m)
- Permet de gérer les objets statiques de façon transparente (invMass = 0), les calculs restent identiques

Note importante

- On applique une impulsion uniquement si les objets vont l'un vers l'autre
- Sinon, on risque d'appliquer l'impulsion plusieurs frames d'affilées
- On regarde le signe de la vitesse relative

```
void OnCollision(Body* A, Body* B, Vec3 normal, float penetration, Vec3 point)
{
		float vRel = Vec3.Dot(A->speed - B->speed, normal);
		if (vRel < 0)
		{
				// Objects getting closer => bounce
				float J = (-(1 + restitution)*vRel)/(A->invMass + B->invMass);
				A->speed += J * A->invMass * normal;
				B->speed -= J * B->invMass * normal;
		}
}
```

Friction statique

- Friction : force de frottement qui s'oppose au mouvement
- Friction statique : empêche mouvement du solide selon la tangente (force la vitesse selon la tangente à être nulle)
- Il suffit de reprendre l'impulsion précédente, selon la tangente, epsilon = 0

Friction statique

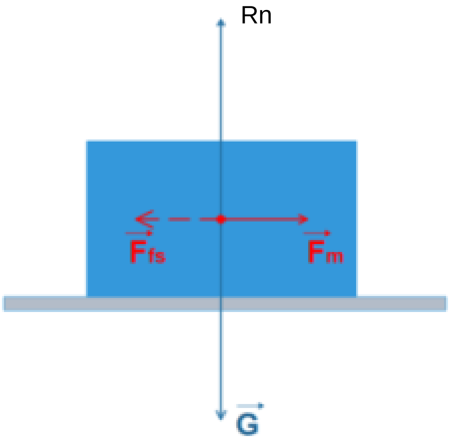
- Friction : force de frottement qui s’oppose au mouvement
- Friction statique : empêche mouvement du solide selon la tangente (force la vitesse selon la tangente à être nulle)
- Il suffit de reprendre l’impulsion précédente, selon la tangente, epsilon = 0

$$\vec{J} = \frac{-(\overrightarrow{v_A^-} - \overrightarrow{v_B^-})|\vec{t}}{\frac{1}{m_A} + \frac{1}{m_B}} \vec{t}$$

Friction statique

- Force de friction statique est limitée évidemment…
- Elle ne s’applique que si la force normale est suffisamment grande

$$f_{s\{max\}} = \mu R_n$$



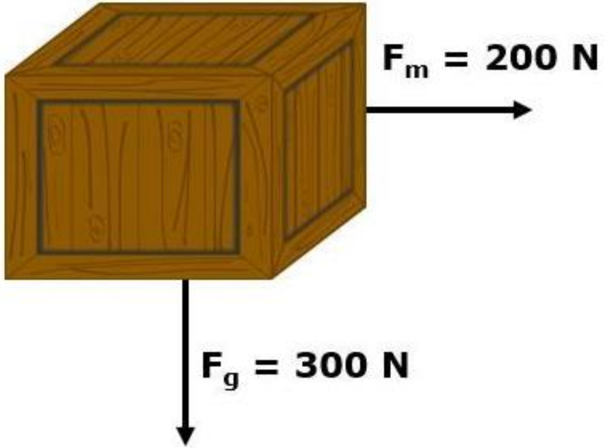
- μ : coefficient de friction statique
- R_n : force de contact, Force normale appliquée par le support sur l’objet

Friction statique

On essaie de mettre en mouvement une caisse de bois ayant un poids de 300N sur un plancher de bois franc, et ce avec une force de 200 N

Est-ce que la caisse se mettra-t-elle en mouvement ?

μ Entre bois et bois = 0,58



$$\begin{aligned} F_{f_s} &= \mu_s \cdot F_N \\ &= 0,58 \cdot 300\ N \\ &= 174\ N \end{aligned}$$

Friction statique

```
void ApplyFriction(Body* A, Body* B, Vec3 normal, float colImpulse)
{
		Vec3 tangent = GetTangent(A->speed - B->speed, normal);
		float vTangent = Vec3.Dot(A->speed - B->speed, tangent);

		float J = -vTangent/(A->invMass + B->invMass);
		J = clamp(J, -Abs(colImpulse) * friction, Abs(colImpulse) * friction);

		A->speed += J * A->invMass * tangent;
		B->speed -= J * B->invMass * tangent;
}
```

Position correction

- Baumgarte : on ajoute une impulsion en fonction de la pénétration
- Autre méthode simple : simplement déplacer bodys selon pénétration

```
void OnCollision(Body* A, Body* B, Vec3 normal, float penetration, Vec3 point)
{
		// Position Correction
		float damping = 0.2f;
		float correction = (penetration * damping)/(A->invMass + B->invMass);
		A->position += A->invMass * correction * normal;
		B->position -= B->invMass * correction * normal;

		// Velocity impulse ...
}
```

- On peut répéter plusieurs fois (diviser correction par nombre d’itérations)

Rotation

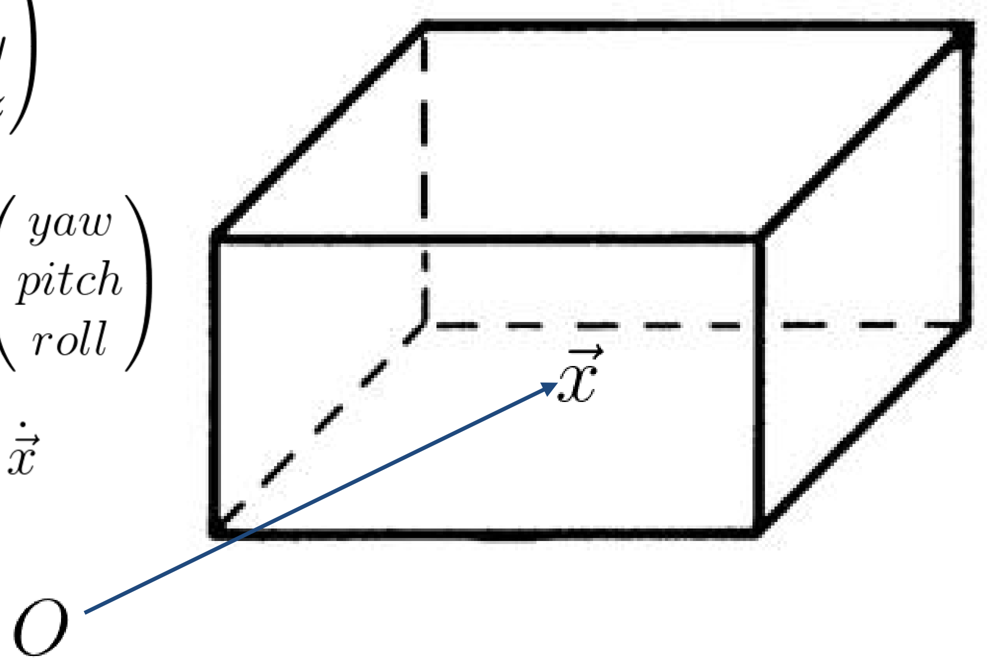
- Analogue en tout point à la position…
- ...en (un peu) plus compliqué

Rigidbody

$$\vec{x}\begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

$$\vec{rot}\begin{pmatrix} yaw \\ pitch \\ roll \end{pmatrix}$$

$$\vec{v} = \dot{\vec{x}}$$



- Solide indéformable
- Position : centre de gravité du solide

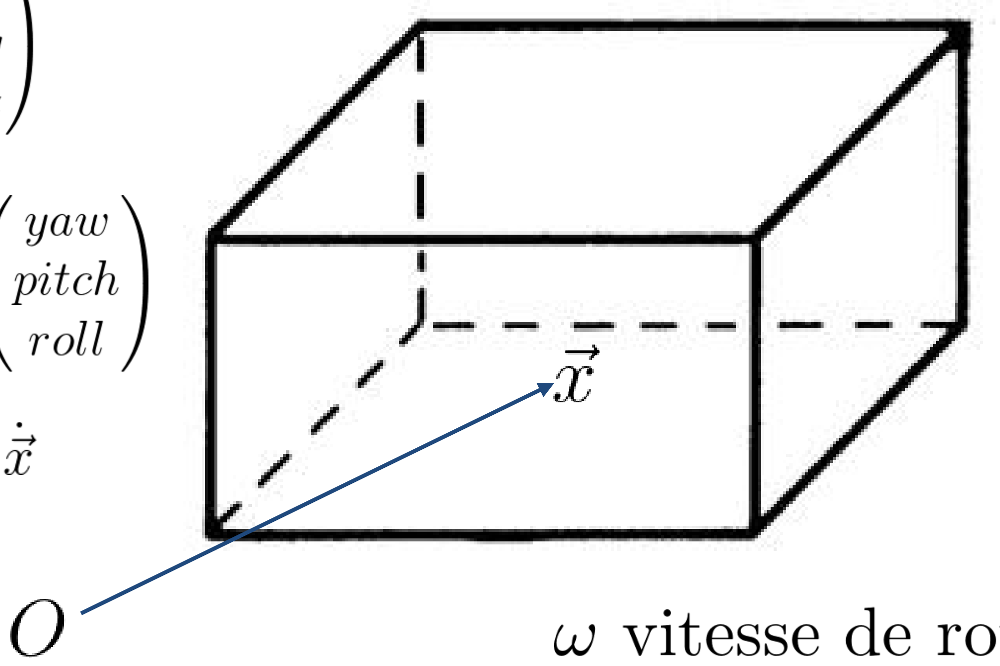
Réponse

Rigidbody

$$\vec{x}\begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

$$\vec{rot}\begin{pmatrix} yaw \\ pitch \\ roll \end{pmatrix}$$

$$\vec{v} = \dot{\vec{x}}$$



- Vitesse angulaire :

$$\vec{\omega} = \omega\,\vec{u}$$

$\vec{u}$ axe de rotation

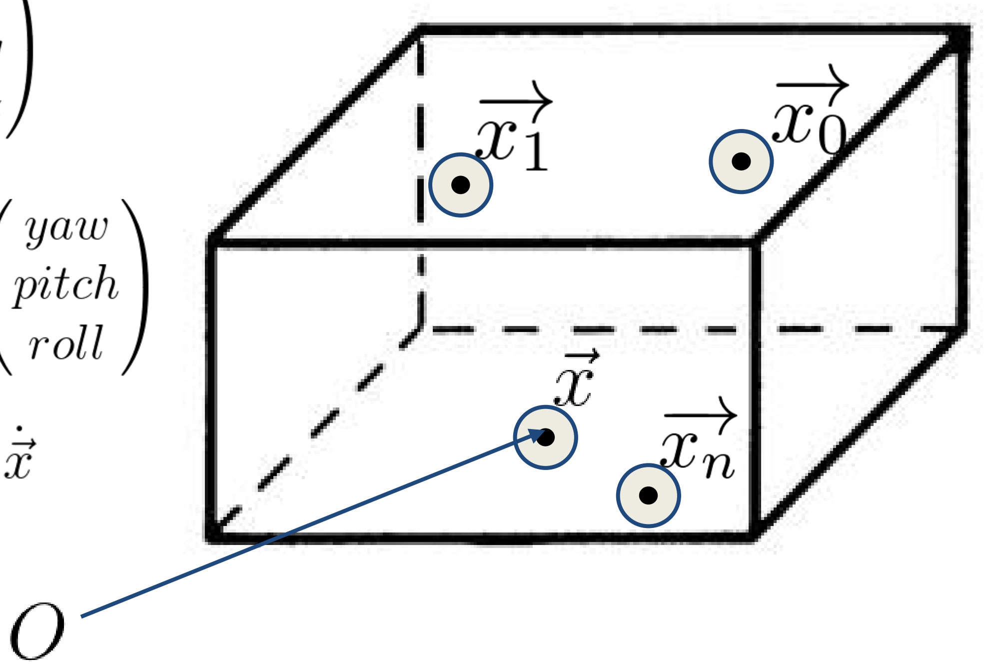
ω vitesse de rotation en radians/seconde

Rigidbody

$$\vec{x}\begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

$$\vec{rot}\begin{pmatrix} yaw \\ pitch \\ roll \end{pmatrix}$$

$$\vec{v} = \dot{\vec{x}}$$



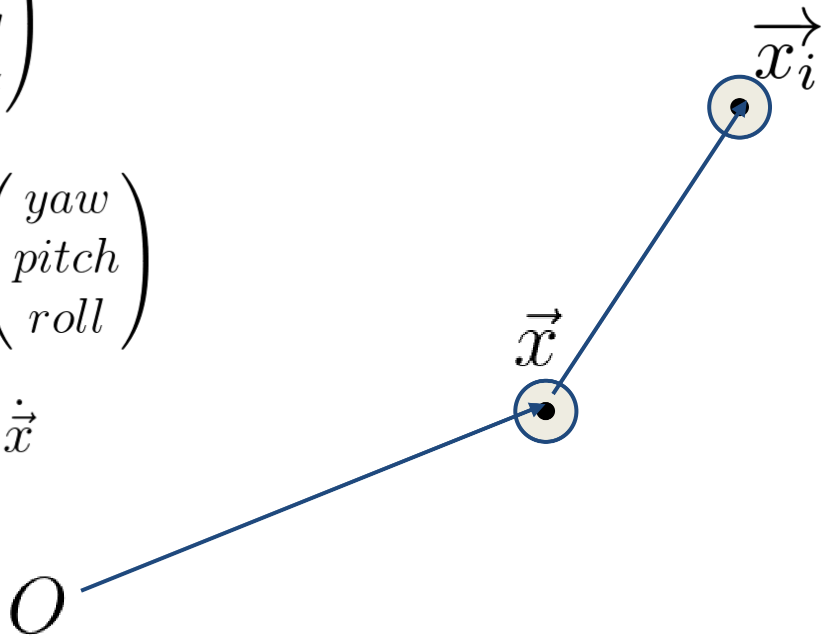
- Ensemble de particules xi de masse mi
- La distance entre 2 particule est constante (indéformable)
- Rotation solide = translation particules

Rigidbody

$$\vec{x}\begin{pmatrix}x\\y\\z\end{pmatrix}$$

$$\vec{rot}\begin{pmatrix}yaw\\pitch\\roll\end{pmatrix}$$

$$\vec{v}=\dot{\vec{x}}$$



$$\vec{r_i}=\vec{x_i}-\vec{x}$$

$$\dot{\vec{r_i}}=\vec{v_i}-\vec{v}$$

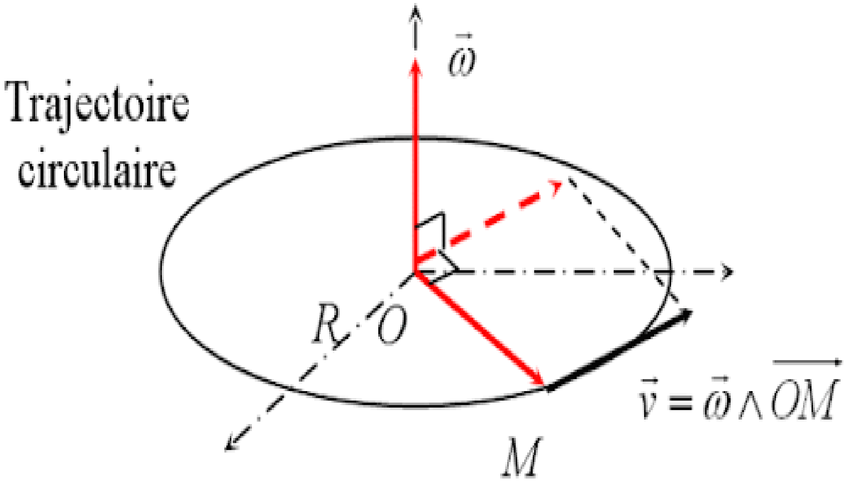
$$\dot{\vec{r_i}}=f(\vec{r_i},\vec{\omega})\ ?$$

Rigidbody

$\vec{x}\begin{pmatrix} x \\ y \\ z \end{pmatrix}$

$\vec{rot}\begin{pmatrix} yaw \\ pitch \\ roll \end{pmatrix}$

$\vec{v} = \dot{\vec{x}}$



O

$$\vec{r_i} = \vec{x_i} - \vec{x}$$

$$\dot{\vec{r_i}} = \vec{v_i} - \vec{v}$$

$$\dot{\vec{r_i}} = \vec{\omega} \wedge \vec{r_i}$$

$$\vec{v_i} = \vec{v} + \vec{\omega} \wedge \vec{r_i}$$

Analogie position/rotation

$$\vec{p} = m\vec{v} \text{ quantité de mouvement}$$

Analogie position/rotation

$$\vec{p} = m\vec{v} \text{ quantité de mouvement}$$

$$\vec{L} = \sum_i \vec{r_i} \wedge \vec{p_i} \ : \ \text{moment cinétique}$$

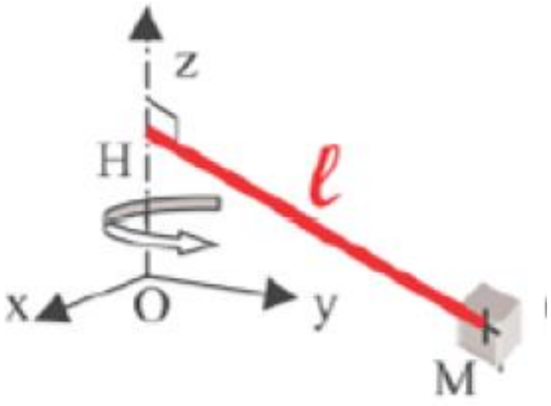
$$\text{quantité de rotation}$$

Analogie position/rotation

$$\vec{L} = \sum_i \vec{r_i} \wedge \vec{p_i} = \sum_i \vec{r_i} \wedge m_i \vec{v_i} = \sum_i \vec{r_i} \wedge m_i (\vec{v} + \vec{\omega} \wedge \vec{r_i})$$

RAPPEL

Par définition le moment d'inertie J_{Oz}, par rapport à un axe Oz, d'un point matériel M de masse m située à une distance *l* de Oz est :



$$\boxed{I_{Oz} = m.l^2}$$

en $Kg.m^2$

- Si le solide est 2 fois plus lourd, il sera 2 fois plus difficile à entrainer en rotation.
- Si le solide est 2 fois plus éloigné de l’axe, il sera 4 fois plus difficile à entrainer en rotation.

→ représente la difficulté à faire tourner un objet

Analogie position/rotation

$$\vec{L} = \begin{pmatrix} \sum m_i(y_i^2 + z_i^2) & -\sum m_i x_i y_i & -\sum m_i x_i z_i \\ -\sum m_i y_i x_i & \sum m_i(z_i^2 + x_i^2) & -\sum m_i y_i z_i \\ -\sum m_i z_i x_i & -\sum m_i z_i y_i & \sum m_i(x_i^2 + y_i^2) \end{pmatrix} \vec{\omega}$$

Analogie position/rotation

$$\vec{L} = \begin{pmatrix} \sum m_i(y_i^2 + z_i^2) & -\sum m_i x_i y_i & -\sum m_i x_i z_i \\ -\sum m_i y_i x_i & \sum m_i(z_i^2 + x_i^2) & -\sum m_i y_i z_i \\ -\sum m_i z_i x_i & -\sum m_i z_i y_i & \sum m_i(x_i^2 + y_i^2) \end{pmatrix} \vec{\omega}$$

$$\vec{L} = Q \times I_{body} \times Q^T \times \vec{\omega}$$

Rotation

Tenseur inertie
(constante)

Rotation inverse

Analogie position/rotation

$$\vec{p} = m\vec{v},\ \dot{\vec{p}} = \vec{F}\ :\ \text{lien entre force et vitesse}$$

2eme loi de Newton (principe de la dynamique)

Analogie position/rotation

$$\vec{p} = m\vec{v},\ \dot{\vec{p}} = \vec{F}\ :\ \text{lien entre force et vitesse}$$

$$\vec{L} = QI_{body}Q^T \times \vec{\omega},\ \dot{\vec{L}}?$$

Analogie position/rotation

$$\vec{L} = \sum_i \vec{r_i} \wedge \vec{p_i}$$

$$\dot{\vec{L}} = \sum_i (\underbrace{\dot{\vec{r_i}} \wedge \vec{p_i}}_{\text{Car la vitesse et p sont Colinéaires}} + \vec{r_i} \wedge \dot{\vec{p_i}}) = \sum_i \vec{r_i} \wedge \dot{\vec{p_i}}$$

Analogie position/rotation

$$\vec{L} = \sum_i \vec{r_i} \wedge \vec{p_i}$$

$$\dot{\vec{L}} = \sum_i (\dot{\vec{r_i}} \wedge \vec{p_i} + \vec{r_i} \wedge \dot{\vec{p_i}}) = \sum_i \vec{r_i} \wedge \dot{\vec{p_i}}$$

$$\dot{\vec{L}} = \sum_i \vec{r_i} \wedge \vec{F_i}$$

Force totale appliquée sur particule i

D'après le PFD

Analogie position/rotation

$$\dot{\vec{L}} = \sum_i \vec{r_i} \wedge \vec{F_i}$$

Force totale appliquée sur particule i

Moment de force

Capacité d’une force à faire tourner un objet par rapport à un point

Analogie position/rotation

$$\vec{p} = m\vec{v}, \dot{\vec{p}} = \sum_i \vec{F_i}$$

Principe fondamental du dynamique

$$\vec{L} = \underbrace{QI_{body}Q^T}_{\text{Tenseur d'inertie}} \times \vec{\omega}, \dot{\vec{L}} = \sum_i \vec{r_i} \wedge \vec{F_i}$$

D'après le théorème du moment cinétique

$$\sum_i \vec{M_O}(\vec{F_i}) = \frac{dL_{Oz}}{dt}$$

Analogie position/rotation

$$\vec{p} = m\vec{v}, \dot{\vec{p}} = \sum_i \vec{F}_i$$

$$\vec{L} = QI_{body}Q^T \times \vec{\omega},\ \dot{\vec{L}} = \sum_i \vec{r_i} \wedge \vec{F}_i$$

- La masse *m* s'oppose au mouvement de translation
- Tenseur d'inertie s'oppose au mouvement de rotation
- Vitesses linéaires et angulaires intégrées avec Euler semi-implicite

Analogie position/rotation

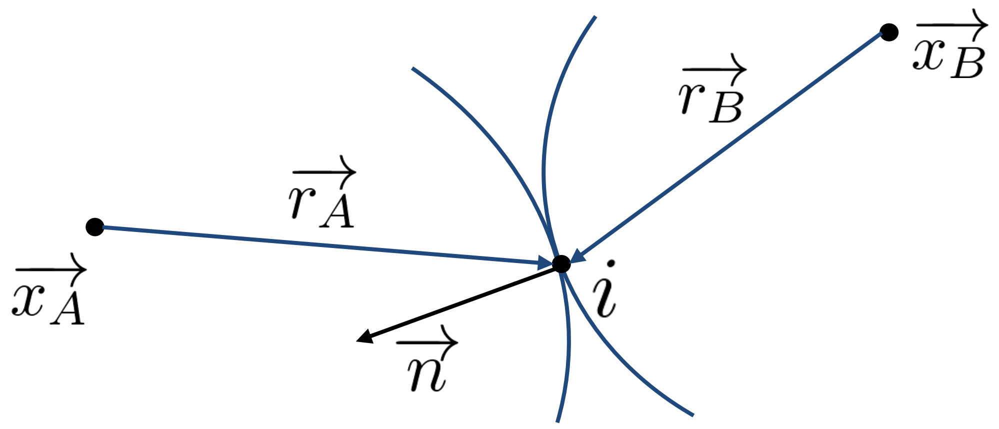
```
void ApplyForce(Body* A, Vec3 localPoint, Vec3 force)
{
        A->forces += force;
        A->torques += Vec3::Cross(localPoint, force); // la somme des moments == dL/dt
}


void UpdateSpeed(Body* A, float dt)
{
        A->speed += A->invMass * A->forces * dt; // vitesse de translation

        Mat3x3 invWorldTensor = A->rotation * A->invLocalTensor * A->rotation.Inverse();

        A->angSpeed += invWorldTensor * A->torques * dt;  // avec L = I w
}
```

Collision response



- A et B collisionnent en le point i
- Force (impulsion) appliquée en i sur A et B pour les séparer ?

Comment calculer les vitesses angulaires après collision

L'impulsion J

$$\vec{\omega} = \vec{\omega}_0 + \Delta\vec{\omega} = \vec{\omega}_0 + \int_{\Delta t_c} \vec{\alpha}(t)dt = \vec{\omega}_0 + \frac{1}{I_{CM}} \int_{\Delta t_c} \vec{\tau}(t)dt = \vec{\omega}_0 + \frac{1}{I_{CM}} \int_{\Delta t_c} \vec{r}(t) \wedge \vec{F}(t)dt$$

Accélération angulaire

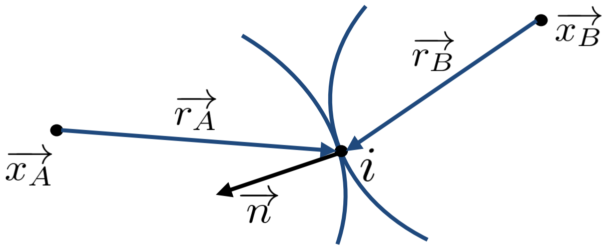
d'après le théorème du moment cinétique

$$\sum_i \vec{M}_O(\vec{F}_i) = I\frac{d\omega}{dt}$$

Accélération angulaire

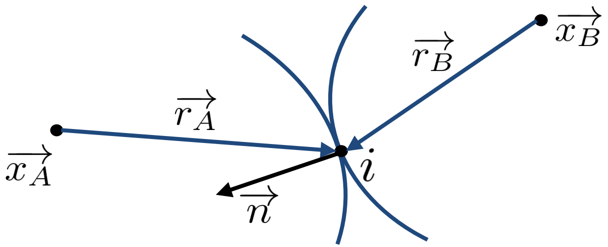
$$\overrightarrow{\omega_A^+} = \overrightarrow{\omega_A^-} + J(I_A^{-1} \times (\overrightarrow{r_{A_i}} \wedge \overrightarrow{n}))$$
$$\overrightarrow{\omega_B^+} = \overrightarrow{\omega_B^-} - J(I_B^{-1} \times (\overrightarrow{r_{B_i}} \wedge \overrightarrow{n}))$$

Collision response



$$\text{3ème loi de Newton :} \quad J\,\overrightarrow{n} = \overrightarrow{F_{A_i}} = -\overrightarrow{F_{B_i}}$$

Collision response

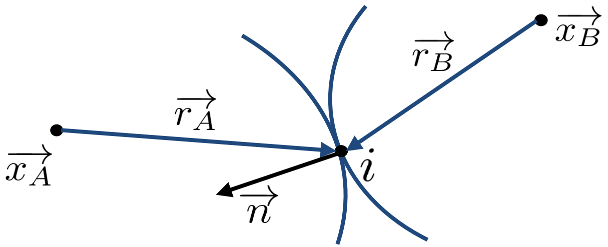


3ème loi de Newton : $\quad J\ \overrightarrow{n} = \overrightarrow{F_{A_i}} = -\overrightarrow{F_{B_i}}$

$$\overrightarrow{v_A^+} = \overrightarrow{v_A^-} + J\, m_A^{-1} \overrightarrow{n}$$

$$\overrightarrow{v_B^+} = \overrightarrow{v_B^-} - J\, m_B^{-1} \overrightarrow{n}$$

Collision response



$$\text{3ème loi de Newton :} \quad \text{J}\ \overrightarrow{n} = \overrightarrow{F_{A_i}} = -\overrightarrow{F_{B_i}}$$

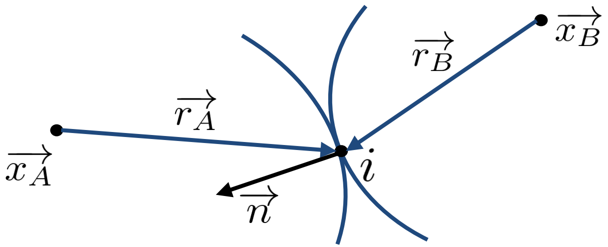
$$\overrightarrow{v_A^+} = \overrightarrow{v_A^-} + J\, m_A^{-1} \overrightarrow{n}$$

$$\overrightarrow{v_B^+} = \overrightarrow{v_B^-} - J\, m_B^{-1} \overrightarrow{n}$$

$$\overrightarrow{\omega_A^+} = \overrightarrow{\omega_A^-} + J(I_A^{-1} \times (\overrightarrow{r_{A_i}} \wedge \overrightarrow{n}))$$

$$\overrightarrow{\omega_B^+} = \overrightarrow{\omega_B^-} - J(I_B^{-1} \times (\overrightarrow{r_{B_i}} \wedge \overrightarrow{n}))$$

Collision response



3ème loi de Newton : $J\ \vec{n} = \overrightarrow{F_{A_i}} = -\overrightarrow{F_{B_i}}$

$$\overrightarrow{v_A^+} = \overrightarrow{v_A^-} + J\, m_A^{-1} \vec{n}$$

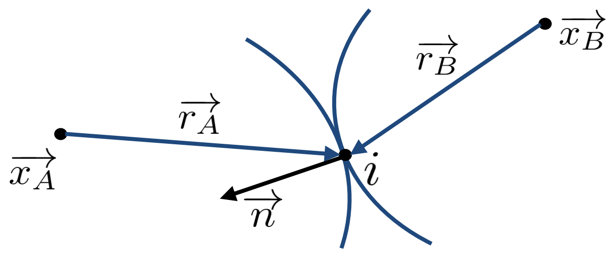
$$\overrightarrow{v_B^+} = \overrightarrow{v_B^-} - J\, m_B^{-1} \vec{n}$$

$$\overrightarrow{\omega_A^+} = \overrightarrow{\omega_A^-} + J(I_A^{-1} \times (\overrightarrow{r_{A_i}} \wedge \vec{n}))$$

$$\overrightarrow{\omega_B^+} = \overrightarrow{\omega_B^-} - J(I_B^{-1} \times (\overrightarrow{r_{B_i}} \wedge \vec{n}))$$

$$(\overrightarrow{v_{A_i}{}^+} - \overrightarrow{v_{B_i}{}^+})|\vec{n} = -\epsilon(\overrightarrow{v_{A_i}{}^-} - \overrightarrow{v_{B_i}{}^-})|\vec{n}$$

Collision response



$$\text{3ème loi de Newton : } \quad \mathrm{J}\ \overrightarrow{n} = \overrightarrow{F_{A_i}} = -\overrightarrow{F_{B_i}}$$

$$\overrightarrow{v_A^+} = \overrightarrow{v_A^-} + J\, m_A^{-1} \overrightarrow{n}$$

$$\overrightarrow{v_B^+} = \overrightarrow{v_B^-} - J\, m_B^{-1} \overrightarrow{n}$$

$$\overrightarrow{\omega_A^+} = \overrightarrow{\omega_A^-} + J(I_A^{-1} \times (\overrightarrow{r_{A_i}} \wedge \overrightarrow{n}))$$

$$\overrightarrow{\omega_B^+} = \overrightarrow{\omega_B^-} - J(I_B^{-1} \times (\overrightarrow{r_{B_i}} \wedge \overrightarrow{n}))$$

$$(\overrightarrow{v_{A_i}{}^+} - \overrightarrow{v_{B_i}{}^+})|\overrightarrow{n} = -\epsilon(\overrightarrow{v_{A_i}{}^-} - \overrightarrow{v_{B_i}{}^-})|\overrightarrow{n}$$

$$\overrightarrow{v_{A_i}} = \overrightarrow{v_A} + \overrightarrow{\omega_A} \wedge \overrightarrow{r_{A_i}}$$

$$\overrightarrow{v_{B_i}} = \overrightarrow{v_B} + \overrightarrow{\omega_B} \wedge \overrightarrow{r_{B_i}}$$

Collision response

$$J = \frac{-(\epsilon+1)(\overrightarrow{v_{A_i}^-} - \overrightarrow{v_{B_i}^-})|\vec{n}}{m_A^{-1} + m_B^{-1} + ((I_A^{-1}(\overrightarrow{r_{A_i}} \wedge \vec{n}) \wedge \overrightarrow{r_{A_i}}))|\vec{n} + ((I_B^{-1}(\overrightarrow{r_{B_i}} \wedge \vec{n}) \wedge \overrightarrow{r_{B_i}}))|\vec{n}}$$

$$\overrightarrow{v_A^+} = \overrightarrow{v_A^-} + J\, m_A^{-1}\overrightarrow{n}$$
$$\overrightarrow{v_B^+} = \overrightarrow{v_B^-} - J\, m_B^{-1}\overrightarrow{n}$$
$$\overrightarrow{\omega_A^+} = \overrightarrow{\omega_A^-} + J(I_A^{-1} \times (\overrightarrow{r_{A_i}} \wedge \overrightarrow{n}))$$
$$\overrightarrow{\omega_B^+} = \overrightarrow{\omega_B^-} - J(I_B^{-1} \times (\overrightarrow{r_{B_i}} \wedge \overrightarrow{n}))$$

Collision response

```
void OnCollision(Body* A, Body* B, Vec3 normal, float penetration, Vec3 point)
{
		Vec3 rA = point - A->position, rB = point - B->position;
		Vec3 vAi = A->speed + Vec3::Cross(A->angularSpeed, rA), vBi = ...;
		Vec3 momentumA = A->invTensorWorld * Vec3::Cross(rA, normal), momentumB = ...;
		float weightRotA = Vec3::Dot( Vec3::Cross(momentumA, rA), normal ), weightRotB = 
...;
	float vRel = Vec3.Dot(vAi - vBi, normal);
	if (vRel >= 0)
		return;

	float J = (-(1 + restitution)*vRel)/(A->invMass + B->invMass + weightRotA + 
weightRotB);

		A->speed += J * A->invMass * normal;
		B->speed -= J * B->invMass * normal;

	A->angularSpeed += J * momentumA;
	B->angularSpeed -= J * momentumB;
}
```

Stabilité

- La partie la plus difficile d’un moteur physique
- Quelques millisecondes en moins peuvent tout détruire
- L’instabilité/l’imprécision sont généralement causée par 2 raisons :

Stabilité

- La partie la plus difficile d’un moteur physique
- Quelques millisecondes en moins peuvent tout détruire
- L’instabilité/l’imprécision sont généralement causée par 2 raisons :

1. Génération de contacts insuffisante (la détection de collision doit générer plusieurs points de contacts quand 2 faces sont à peu près alignées)

Stabilité

- La partie la plus difficile d’un moteur physique
- Quelques millisecondes en moins peuvent tout détruire
- L’instabilité/l’imprécision sont généralement causée par 2 raisons :

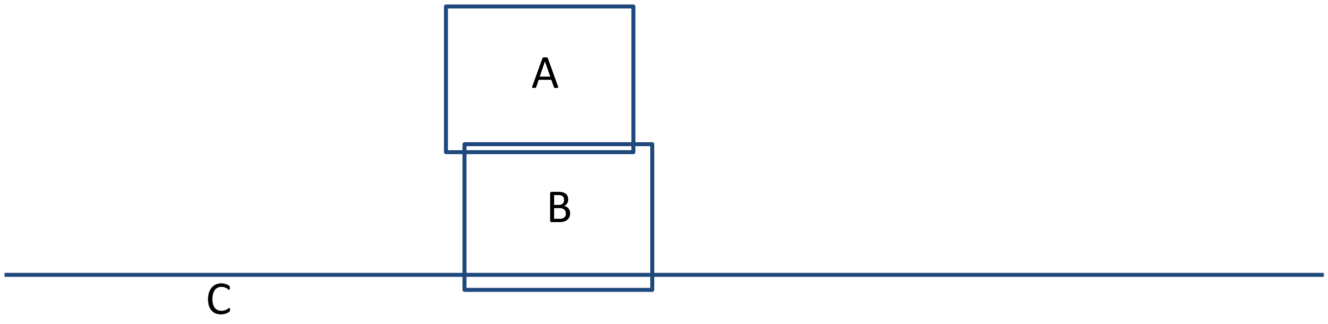
1. Génération de contacts insuffisante (la détection de collision doit générer plusieurs points de contacts quand 2 faces sont à peu près alignées)
2. Les impulses calculées sont incorrectes (imprécises)

Imprécision des impulses

- Collision entre A et B, on applique une impulsion, A et B se séparent (leurs vélocités sont mises à jours)

Imprécision des impulses

- Collision entre A et B, on applique une impulsion, A et B se séparent (leurs vélocités sont mises à jours)
- Collision entre B et C, on applique une impulsion, B et C se séparent…
- ...mais la nouvelle vélocité de B peut l’amener à collisionner avec A de nouveau



Imprécision des impulses

- Problème mathématique : systèmes de n contraintes à résoudre (pour n collisions)
- Résoudre une contrainte “défait” les autres

Imprécision des impulses

- Problème mathématique : systèmes de n contraintes à résoudre (pour n collisions)
- Résoudre une contrainte “défait” les autres
- Solution simple : on reparcourt toutes les contraintes une par une encore…
- “Sequential Impulses”

Sequential Impulses

- 2 choses importantes à vérifier à chaque impulse
  1. L’impulse totale ne doit pas être négative (une collision ne doit jamais attirer un solide vers l’autre)
  2. Soit vitesse au point de contact est exactement la vitesse de rebond, soit l’impulse est nulle (une impulse ne pousse que si nécessaire)

Autres astuces pour stabilités

- Cacher les impulses
- Sub-stepping
- Ilôts (optimisation, permet de gagner des perfs qu’on réinvestit en stabilité)

References

- Physics-Based Animation by Kenny Erleben et al.
- Real-Time Collision Detection by Christer Ericson.
- Collision Detection in Interactive 3D Environments by Gino van den Bergen.
- Fast Contact Reduction for Dynamics Simulation by Adam Moravanszky and Pierre Terdiman in Game Programming Gems 4.
- https://erikonarheim.com/posts/understanding-collision-constraint-solvers/
- https://box2d.org/files/ErinCatto_SequentialImpulses_GDC2006.pdf