



# Moteur Physique pour les jeux vidéo

## 5<sup>ème</sup> GamiX

### Chapitre 4 : Détection des collisions

---

2025 – 2026

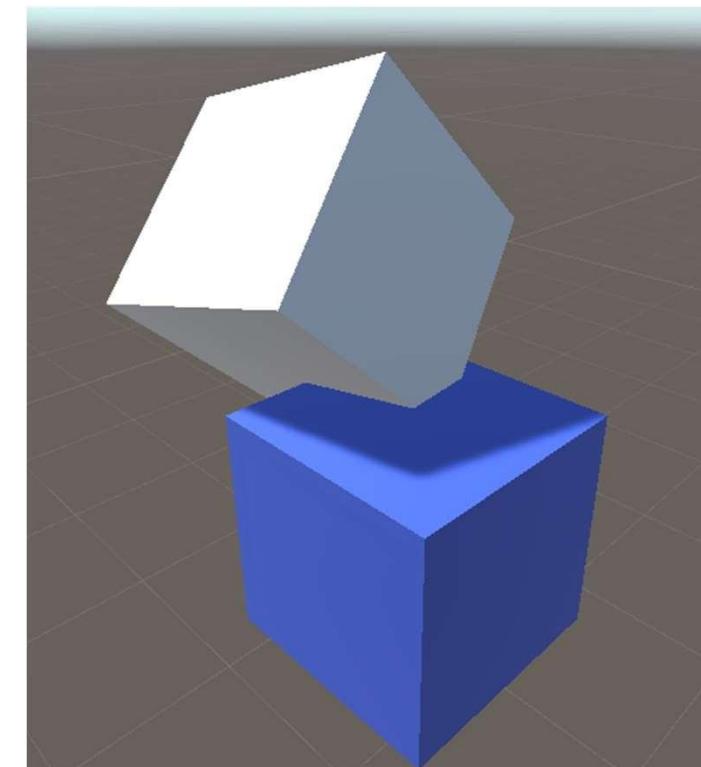
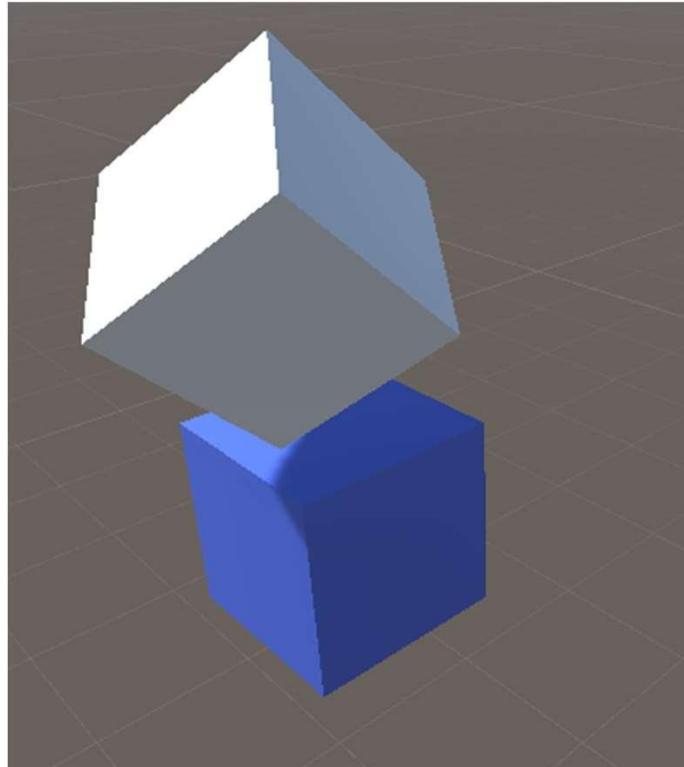
AHMED AMMAR

## Objectif

« La détection de collision a pour but de déterminer Si, Où et Quand deux objets sont en collision »

# Problème des contraintes non-pénétrables

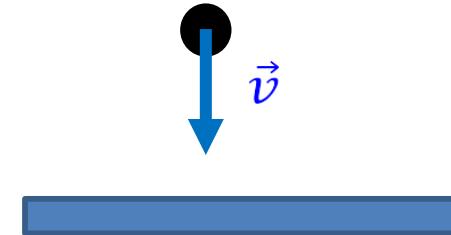
Le problème qui consiste à éviter que les différents objets s'interpénètrent lors de leurs mouvement dans un environnement donné.



# Problème des contraintes non-pénétrables

Supposons, dans un premier temps, que nous simulons une masse ponctuelle qui tombe sur un plancher fixe.

- Lorsque la particule va entrer en contact avec le sol, puisqu'il n'y a pas d'interpénétration, il faut que la particule **change soudainement de vecteur vitesse** (la collision s'opère instantanément.).
- Le **vecteur d'état** qui décrit la position et la vitesse de la particule considérée, va présenter une **discontinuité du vecteur vitesse ce qui pose un problème !**



Les routines numériques qui ont pour rôle la résolution des équations différentielles (**ODE**) sont **supposent que  $Y(t)$  varie de façon "continue"** ce qui n'est pas respecté lors de la collision.

# La solution ?

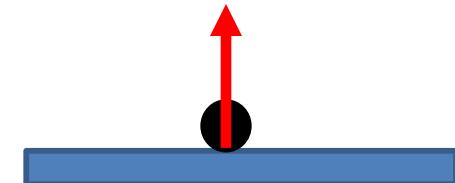
➤ Pour **contourner ce problème**, nous allons procéder comme suit:

- Si une collision se produit **au temps  $tc$** , nous disons au solveur de l'ODE de **s'Arrêter**.
- Nous prenons alors l'état à ce moment,  $Y(tc)$ , et calculons **comment les vitesses** des objets impliqués dans la collision **doivent changer**.
- Nous appelons le **nouvel état** qui tient compte de ces changements  $Y(tc)_{+}$   
*( $Y(tc)$  et  $Y(tc)_{+}$  présentent les mêmes valeurs pour toutes les variables spatiales (position et orientation), mais seront différents pour les variables de vitesse des objets impliqués dans la collision au temps  $tc$ )*
- On **redémarre** alors le solveur numérique, avec le **nouvel état  $Y(tc)_{+}$** , et on lui demander de simuler à partir de ce temps  $tc$ .

## Autre Problème

➤ Supposons maintenant le cas où un corps repose sur l'autre en un point p (la particule au repos sur le sol par exemple) tout en étant au repos.

- Dans ce cas, nous calculons une force qui empêche la particule d'accélérer vers le bas
- Nous appelons la force entre la particule et le sol la force de contact.
- Cet état de contact au repos ne nous oblige pas à arrêter et redémarrer le solveur ODE à chaque instant.



Du point de vue du solveur ODE, les forces de contact sont juste une partie de la force retournée par la routine qui calcule les forces extérieures appliquées à la particule .

# Problème des contraintes non-pénétrables

Dans ce qui suivra, nous devrons faire face à **deux problèmes**:

- le **calcul** de **changements de vitesse** pour une collision avec contact
- le **calcul** des **forces de contact** qui empêchent l'inter-pénétration.

!

Mais avant que nous puissions résoudre ces problèmes nous avons à traiter de la question géométrique de **déetecter les contacts entre les différents objets**

# La particule qui chute sur le sol.....

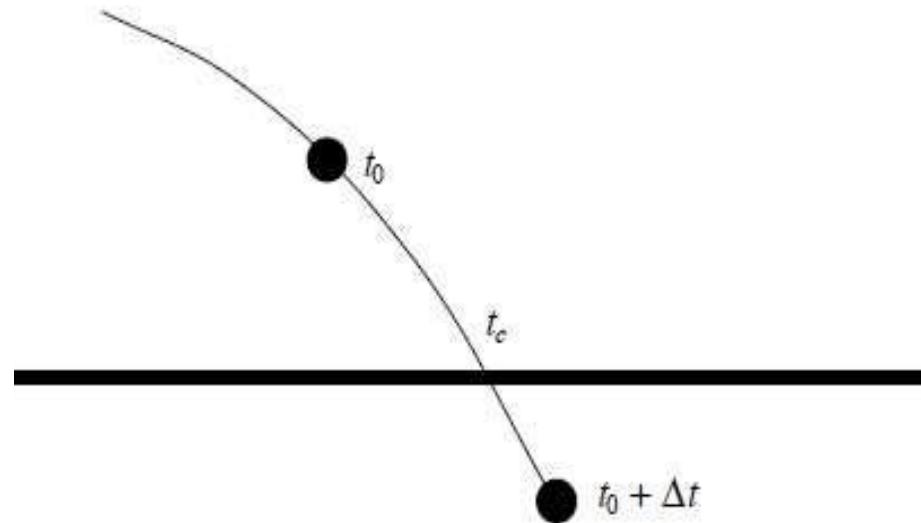
Lors des calculs, nous calculons la position de la particule lors de sa chute à des **instants spécifiques**.

$t_0, t_0+Dt, t_0+2Dt, \dots$

$$t_0 < t_c < t_0+Dt$$



l'instant qui correspond au  
contact particule-sol



Si nous devons stopper puis relancer le simulateur à l'instant  $t_c$  (comme expliqué précédemment), **nous devons tout d'abord être capable de calculer l'instant  $t_c$**  et pour l'instant la seule information que nous avons est :

$$t_0 < t_c < t_0+Dt$$

# Application

- Reprendre le code de la chute libre avec frottements ([avec RK4](#)).
- Le sol est en  $Y=0$
- Déetecter la collision.
- Traiter la collision: On prendra  $\overrightarrow{V_{apres}} = -e * \overrightarrow{V_{avant}}$  avec  $e=1$

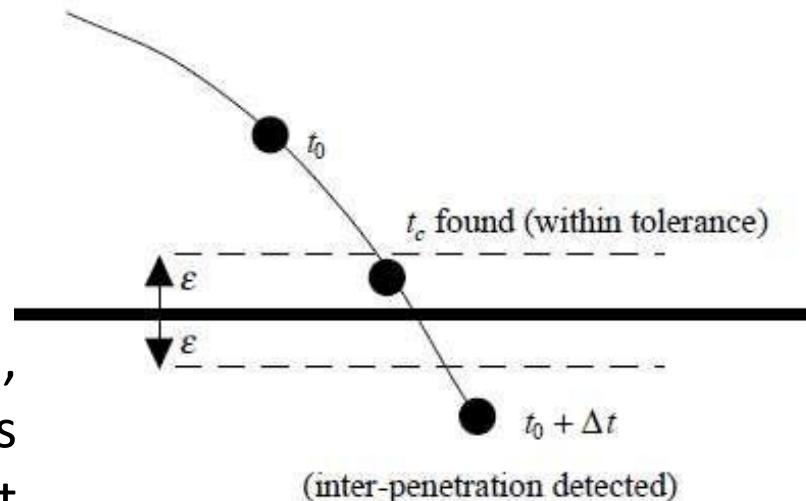


# La particule qui chute sur le sol.....

Nous résolvons pour  $t_c$  **numériquement**, avec une certaine **tolérance**.

- La méthode la plus facile est celle de la **bisection (dichotomie)**.

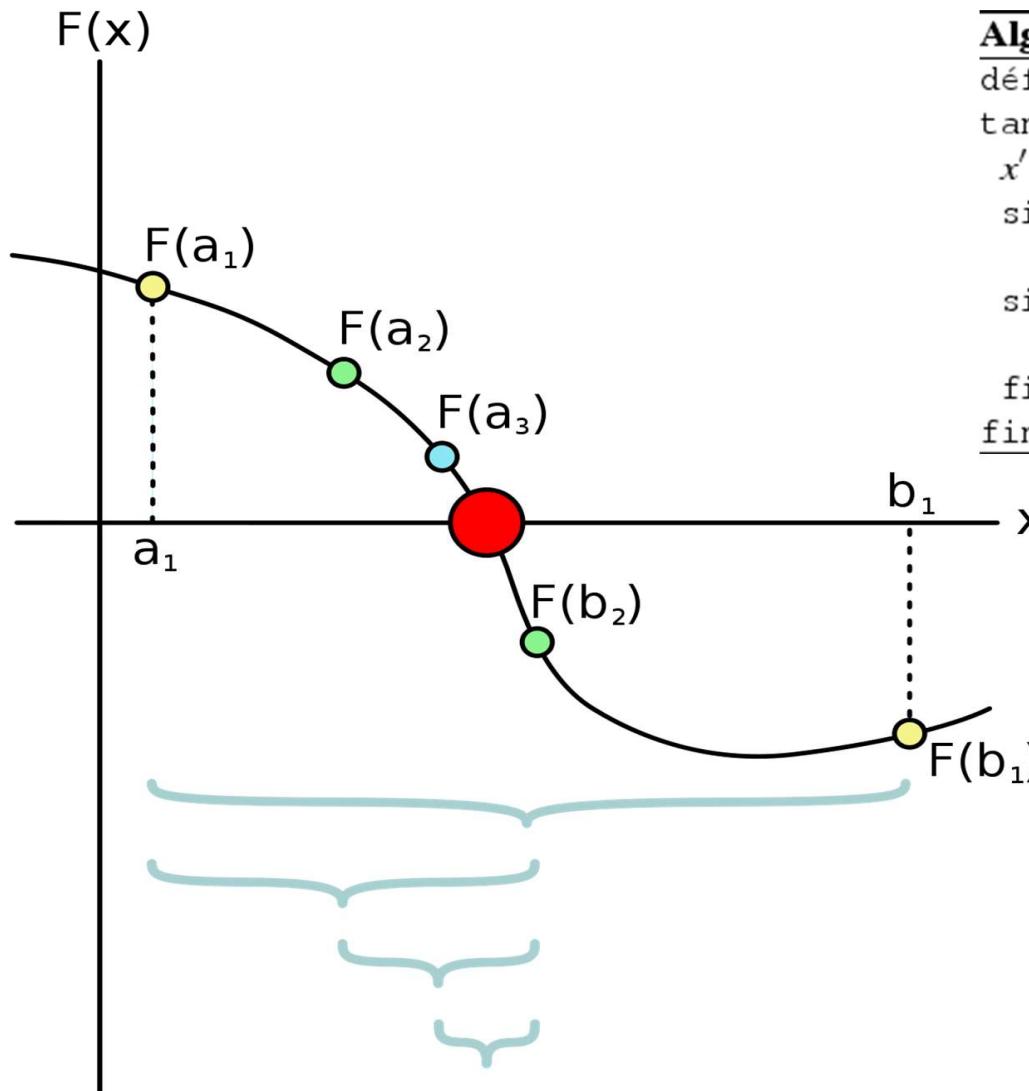
Si à  $t_0+Dt$  nous détectons inter-pénétration, nous informons le solveur ODE que nous souhaitons revenir au temps  $t_0$  et redémarrer, et simuler vers l'avant au temps  $t_0+Dt/2$



**Si** le simulateur atteint  $t_0+Dt/2$  sans rencontrer d'inter-pénétration, nous savons que le temps de collision  $t_c$  se situe entre  $t_0+Dt/2$  et  $t_0+Dt$ .

**Sinon**,  $t_c$  est inférieure à  $t_0+Dt/2$ , et nous essayons à nouveau avec  $t_0+Dt/4$ .

# Méthode de la Dichotomie (Bisection)



---

## Algorithme 3 bissection

---

```
définir a,b réels { avec  $f(a) * f(b) < 0$  }
tantque  $|a - b| > \varepsilon$ 
     $x' \leftarrow (a + b)/2$ 
    si  $f(x') * f(a) < 0$ 
         $b \leftarrow x'$ 
    sinon
         $a \leftarrow x'$ 
    finsi
fintantque
```

---



# La particule qui chute sur le sol.....

## Application

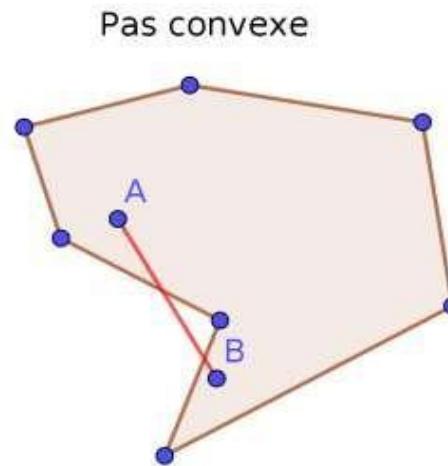
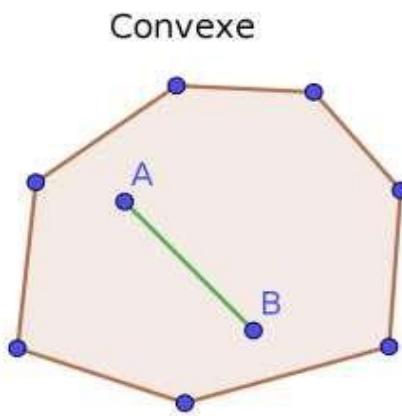
- Initialisez **un cube** en chute libre à  $Y > 0$  avec vitesse nulle et masse définie.
- Utilisez la méthode d'intégration **RK4** pour mettre à jour la position et la vitesse à chaque frame.
- **Déetectez la collision** avec le sol en vérifiant si  $Y \leq 0$  et calculez le temps de collision **tc** avec une **méthode de bisection**.
- **Arrêtez le cube** en fixant sa position à  $Y = 0$  et sa vitesse à zéro lorsque la collision se produit.



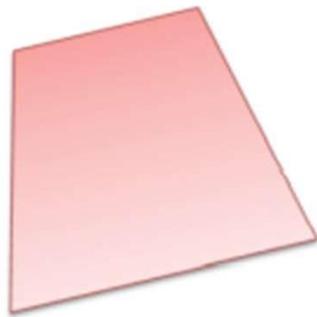
# Corps rigides

Nous allons d'abord décrire comment vérifier de manière efficace l'existence d'**inter-pénétration** ou de **points de contact** entre les **corps rigides** définis comme étant des **polyèdres convexes**.

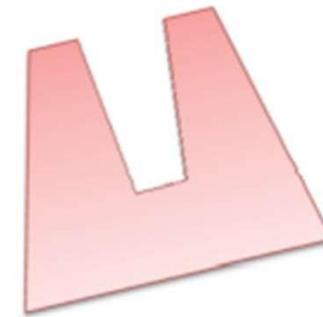
# Objet convexe



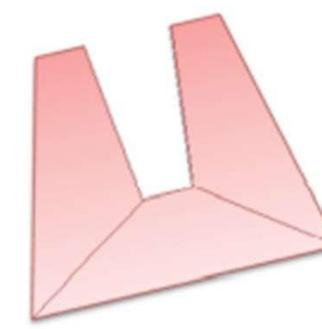
Pour déterminer si une forme est convexe ou concave, on regarde **le nombre de fois qu'une ligne coupe cette forme** : **Une forme convexe n'aura que deux points d'intersection.**



Objet Convexe



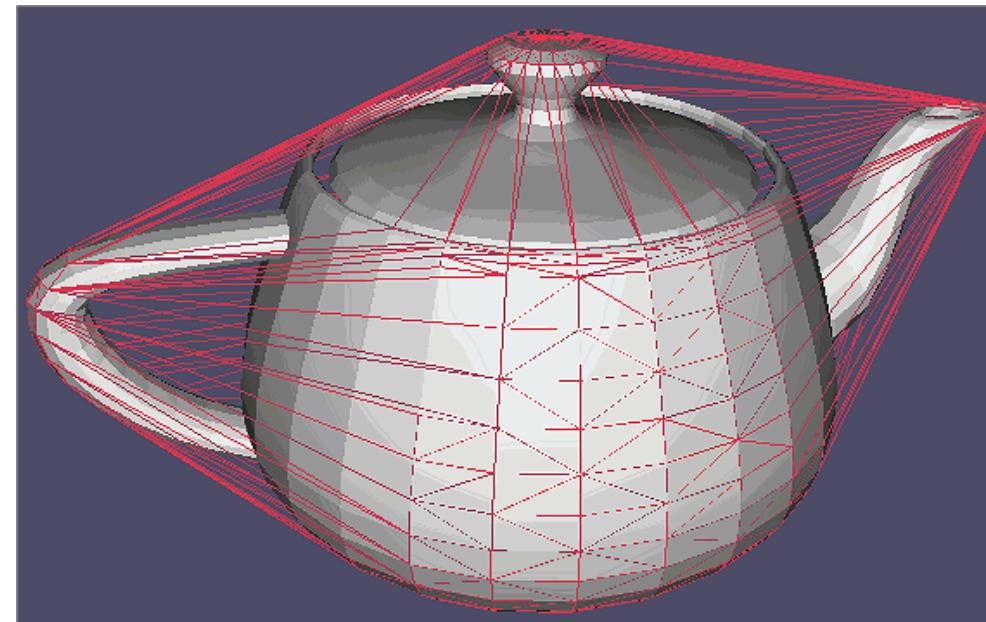
Objet non-convexe



Décomposition en objets convexes

# Enveloppe Convexe

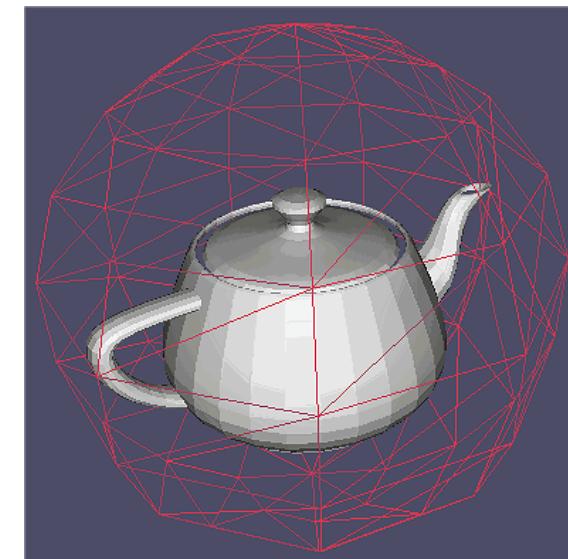
- ❑ Créez la plus petite **surface/volume** convexe **englobant** l'objet.
  - **Bonne représentation** de tous les objets convexes.
  - Crée **de fausses collisions positives** pour les objets concaves.
  - Peut encore être **très complexe**, donc détection coûteuse.



# Sphère Englobante

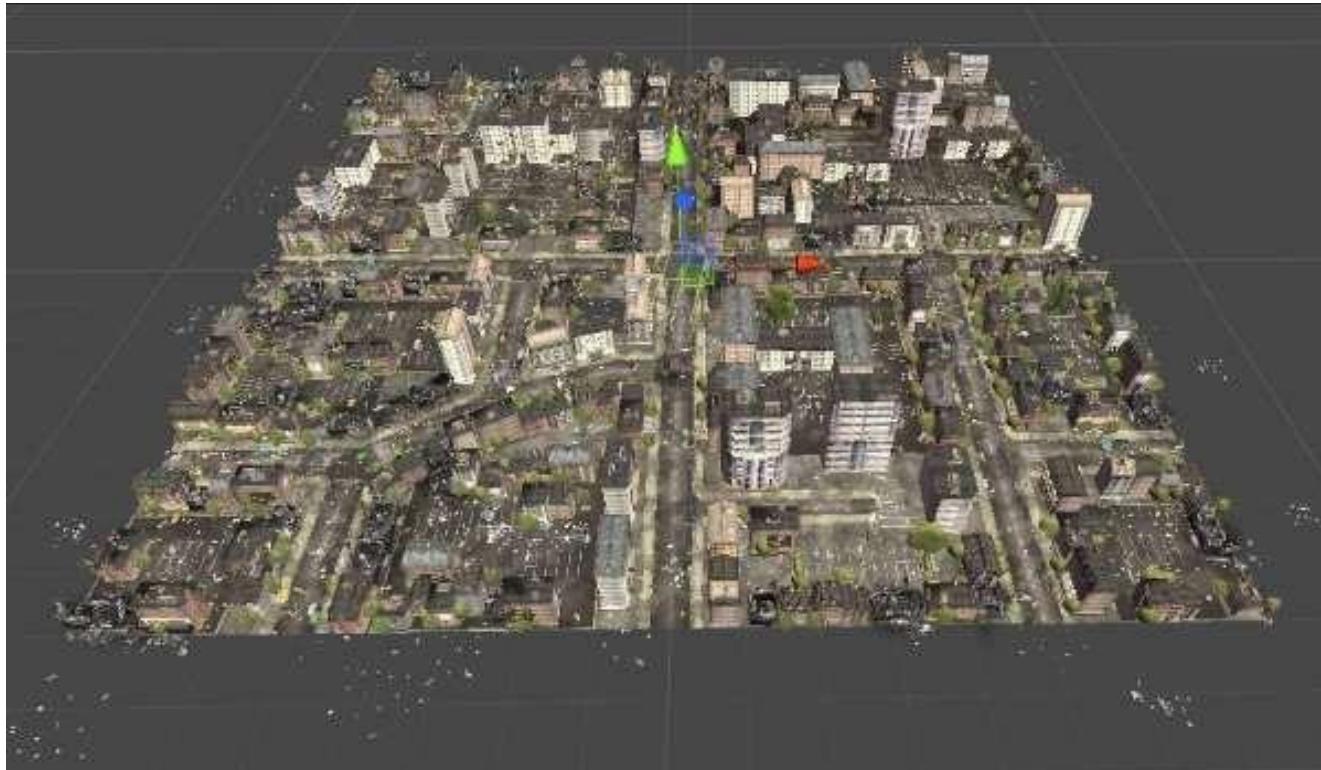
□ Créez la **sphère minimale englobant** l'objet.

- Généralement, un ajustement médiocre de l'objet (par exemple, un tuyau), entraînant de nombreuses fausses collisions positives.
- Stockée en seulement 4 scalaires, la détection de collisions entre sphères est très rapide (11 opérations primitives).
- Trivial à mettre à jour lors de la rotation...



## Problématique

- Il existe des algorithmes permettant de détecter les **collisions entre deux objets**.

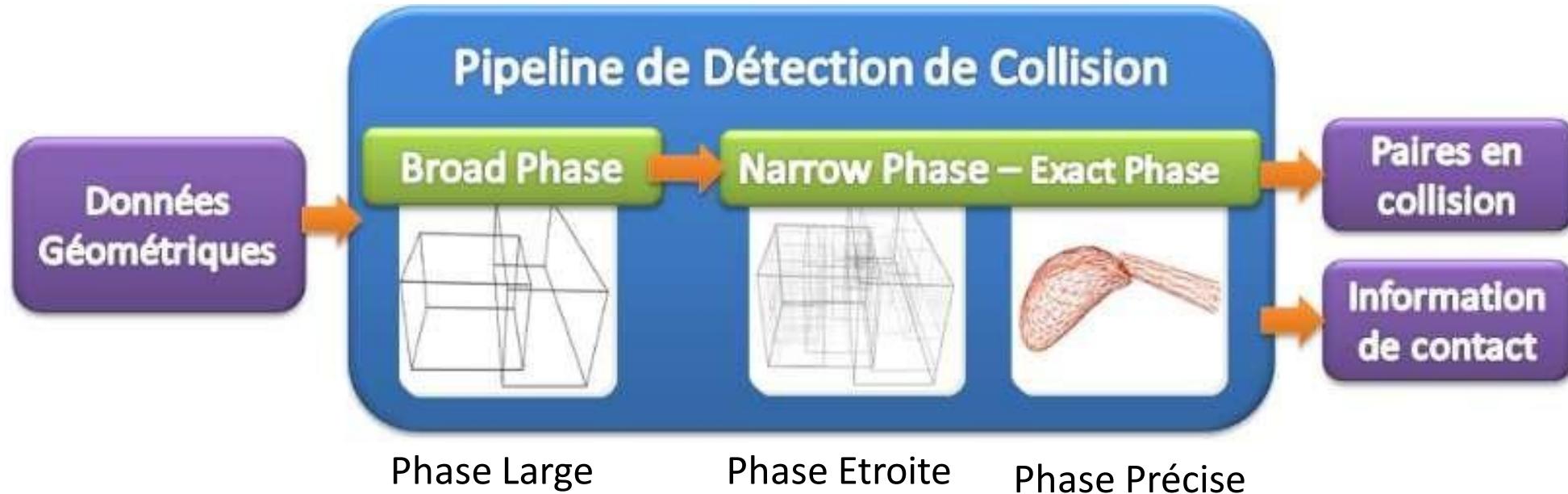


Plusieurs milliers  
d'objets ?

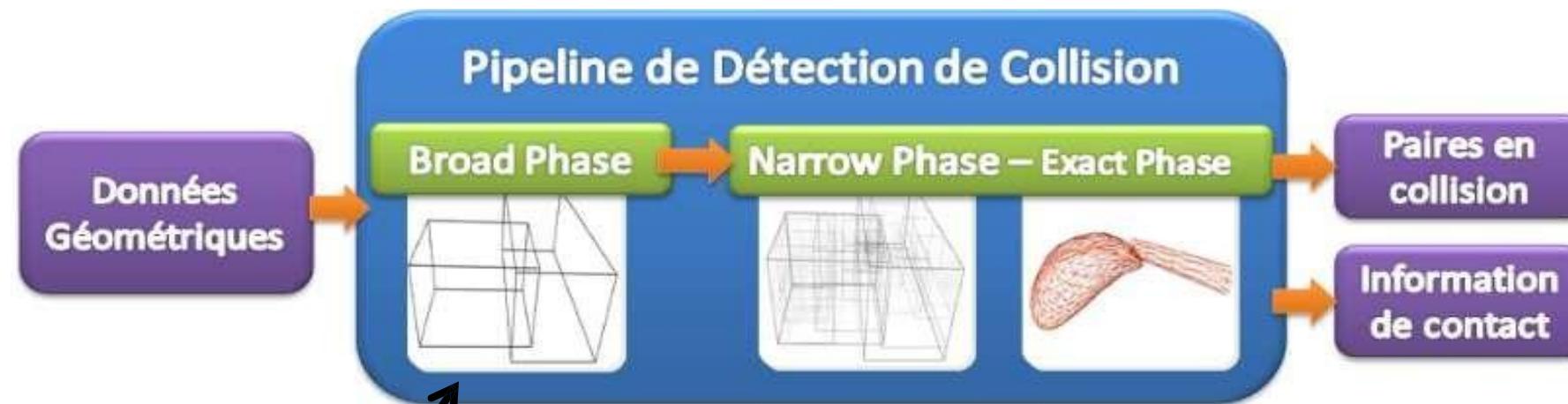


Se munir de **techniques d'accélération** !

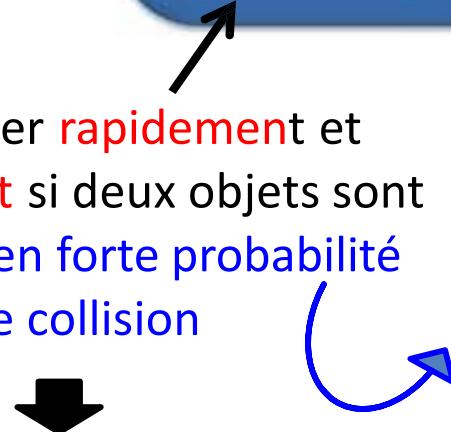
# Pipeline de détection de collisions



# Pipeline de détection de collisions



Déterminer **rapidement** et **efficacement** si deux objets sont ou non en **en forte probabilité de collision**



Exécute un test **plus précis** afin de déterminer les zones des objets en interaction.

Pour connaître l' **instant** et le **lieu précis** (vertex, cotes, faces) de contacte

Diminuer le plus largement possible le nombre d'objets à tester lors des étapes suivantes.

## Phase Large (Broad phase)

Trois principales familles d'algorithmes offrant la possibilité de **connaitre rapidement** si deux objets sont ou non en collision :

- La méthode de **force brute** (recherche exhaustive)
- La méthode dite du "**Sweep and Prune**" (SaP)
- La méthode utilisant le **découpage spatial**.
- La méthode **cinématique**.

## Phase Large (Broad phase)

La méthode de **force brute** (recherche exhaustive) :

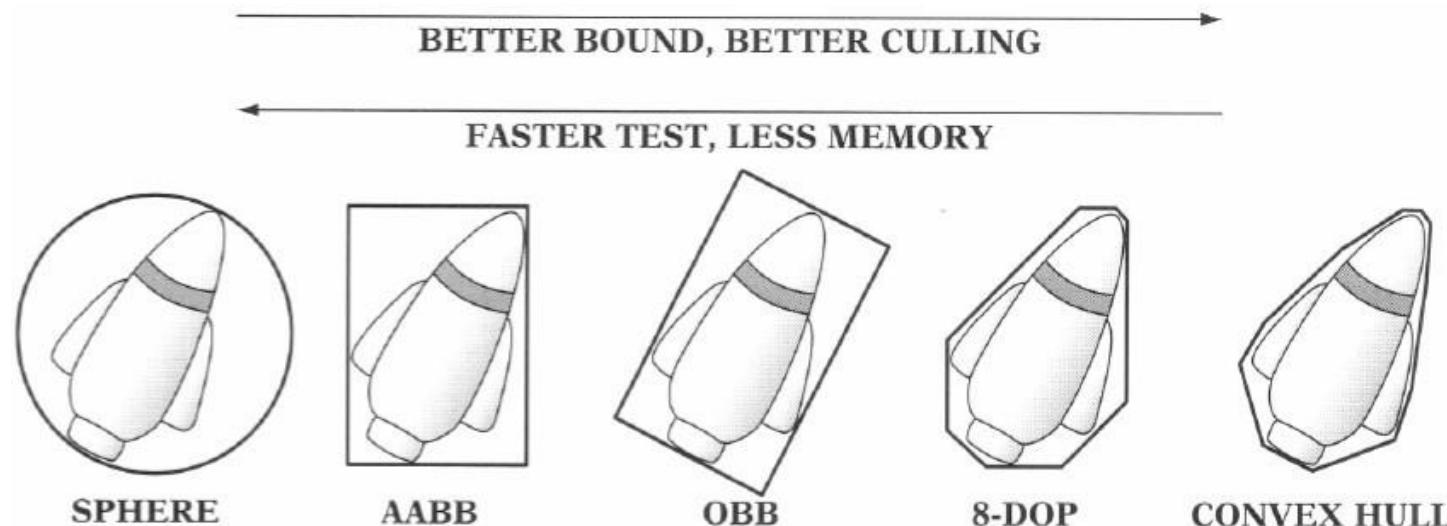
- Se base sur la **comparaison** des **volumes englobants** (**bounding boxes**) des paires d'objets entre eux afin de déterminer s'ils sont ou non en collision.
- L'approche **la plus simple** et **la plus coûteuse**.  
→ Dans le cas d'une simulation avec  $n$  objets rigides, **le nombre de tests à effectuer** est de  $\frac{(n^2 - n)}{2}$

Pas d'auto-test

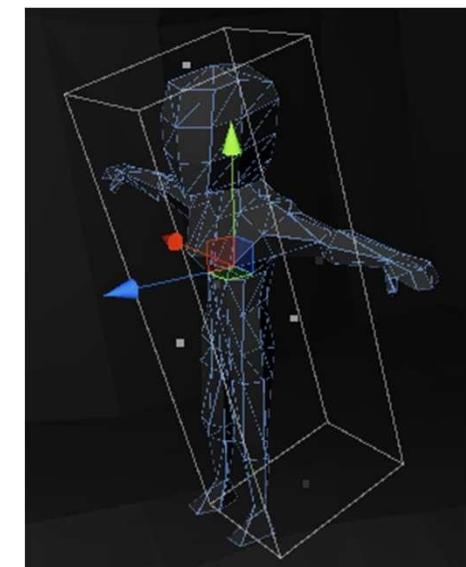
/

Pas de test répétés (A,B) et (B,A)

## Phase Large (Broad phase) : Volumes englobants (Bounding volumes)

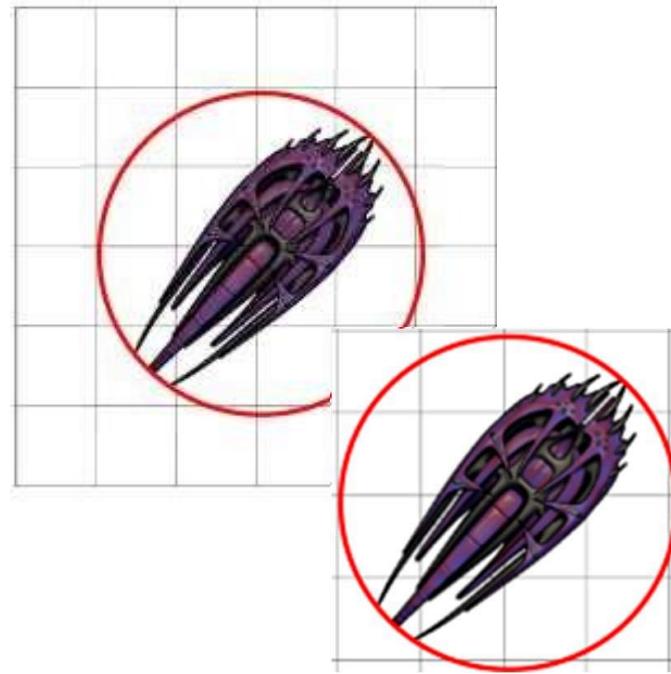


- sphere,
- Axis-Aligned Bounding Box (AABB),
- Oriented Bounding Box (OBB),
- eight-direction Discrete Oriented Polytope (8-DOP)
- convex hull



## Phase Large (Broad phase) : Volumes englobants (Bounding volumes)

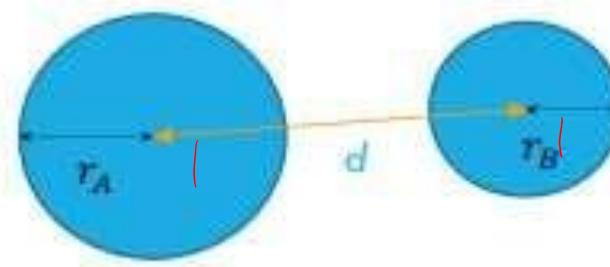
### □ Bounding sphere



•Inconvénient: n'est pas un bon volume englobant pour les objets allongés.

- Nécessite 2 valeurs: centre et rayon

- Tester :



Collision ssi:  $(d < (r_A + r_B))$



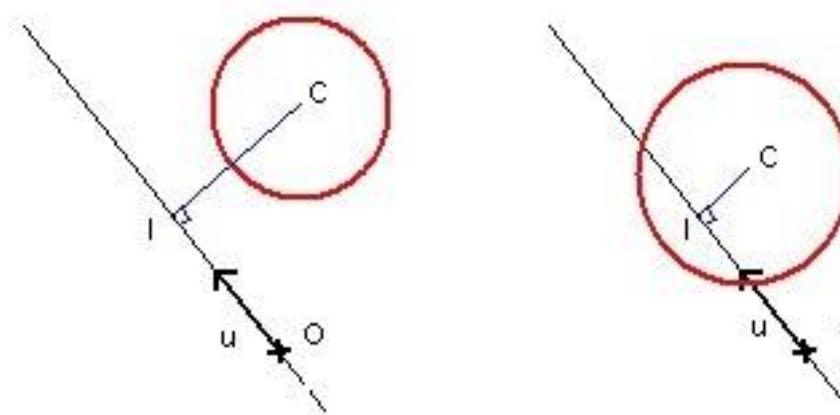
# Phase Large (Broad phase) : Volumes englobants (Bounding volumes)

## □ Bounding sphere

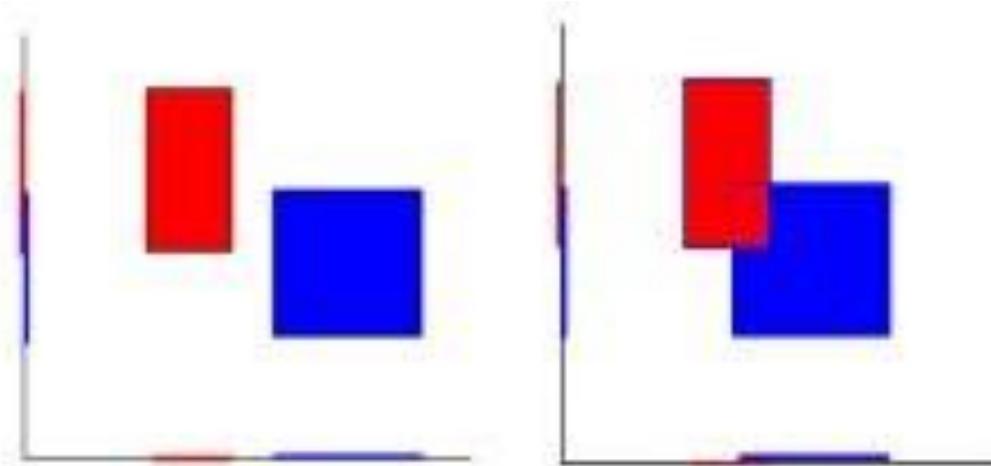
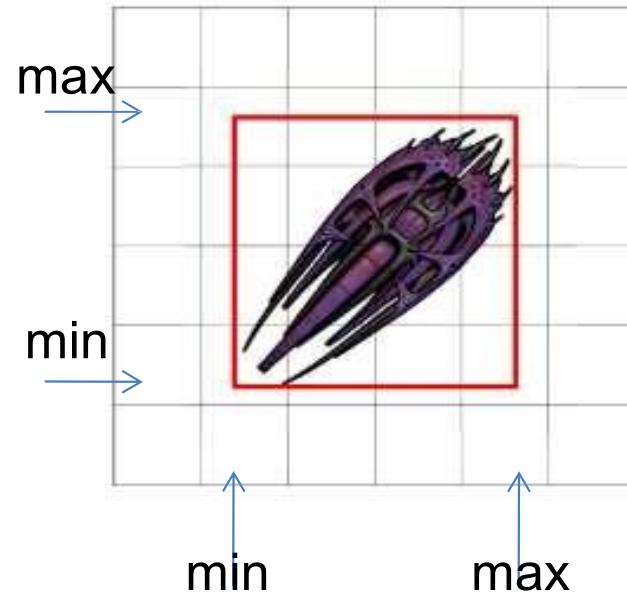


une balle qui touche une raquette  
"horizontale"

Collision entre un cercle et un segment quelconque



## Phase Large (Broad phase) : Volumes englobants (Bounding volumes)



- Nécessite 6 valeurs: min et max de chaque dimension
- Tester le recouvrement des projections
- Inconvénient: n'est pas invariant par rotation

## Phase Large (Broad phase) : Volumes englobants (Bounding volumes)



**sol plat**

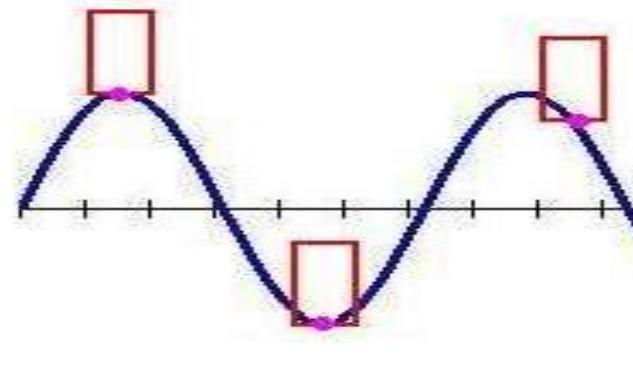
Le sol plat n'a qu'un seul paramètre : son altitude « *a* ».

➤ si l'ordonnée du point du bas de la bounding box est supérieure à '*a*', alors on a pas de collision

## Phase Large (Broad phase) : Volumes englobants (Bounding volumes)



Sol courbe



une fonction cartésienne  $f(x) = y$

Pour savoir si le perso touche ou pas la fonction, nous n'allons considérer qu'un seul point  $x,y$  : celui **en bas au milieu de la AABB**.

Comment savoir si le joueur est en dessus ou en dessous de la courbe ?

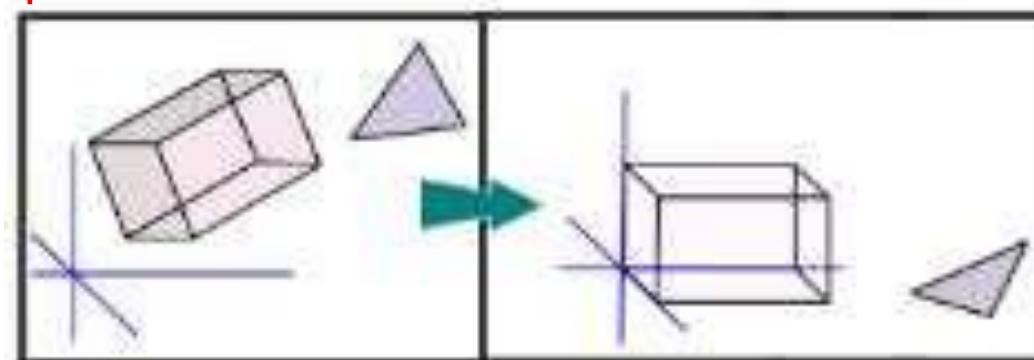
→ Il suffit de voir si  $f(x) > y$  ou non.

# Phase Large (Broad phase) : Volumes englobants (Bounding volumes)

## □ Oriented bounding box OBB



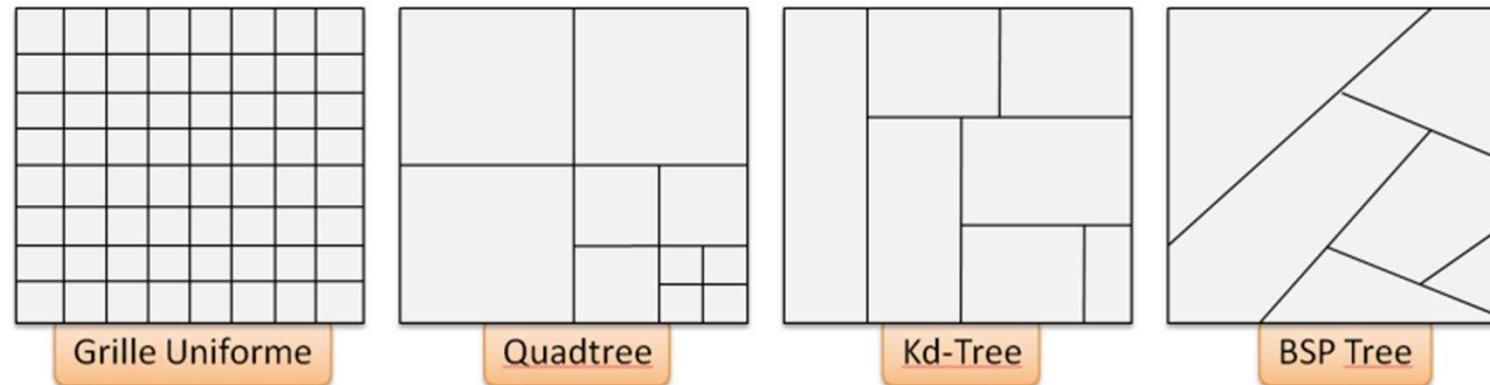
- Nécessite 15 valeurs: position (3), orientation (9), étendue (3)
- Tester changement d'axes (prendre avec les axes de la boîte) ou bien « Théorème de l'axe séparateur »



- Inconvénient: couteux en calcul

- **Subdivision (découpage) spatiale (Spatial subdivision)**

- Il s'agit, en général, de subdivisionnaire l'espace en cellules puis d'utiliser les emplacements de ces dernières.



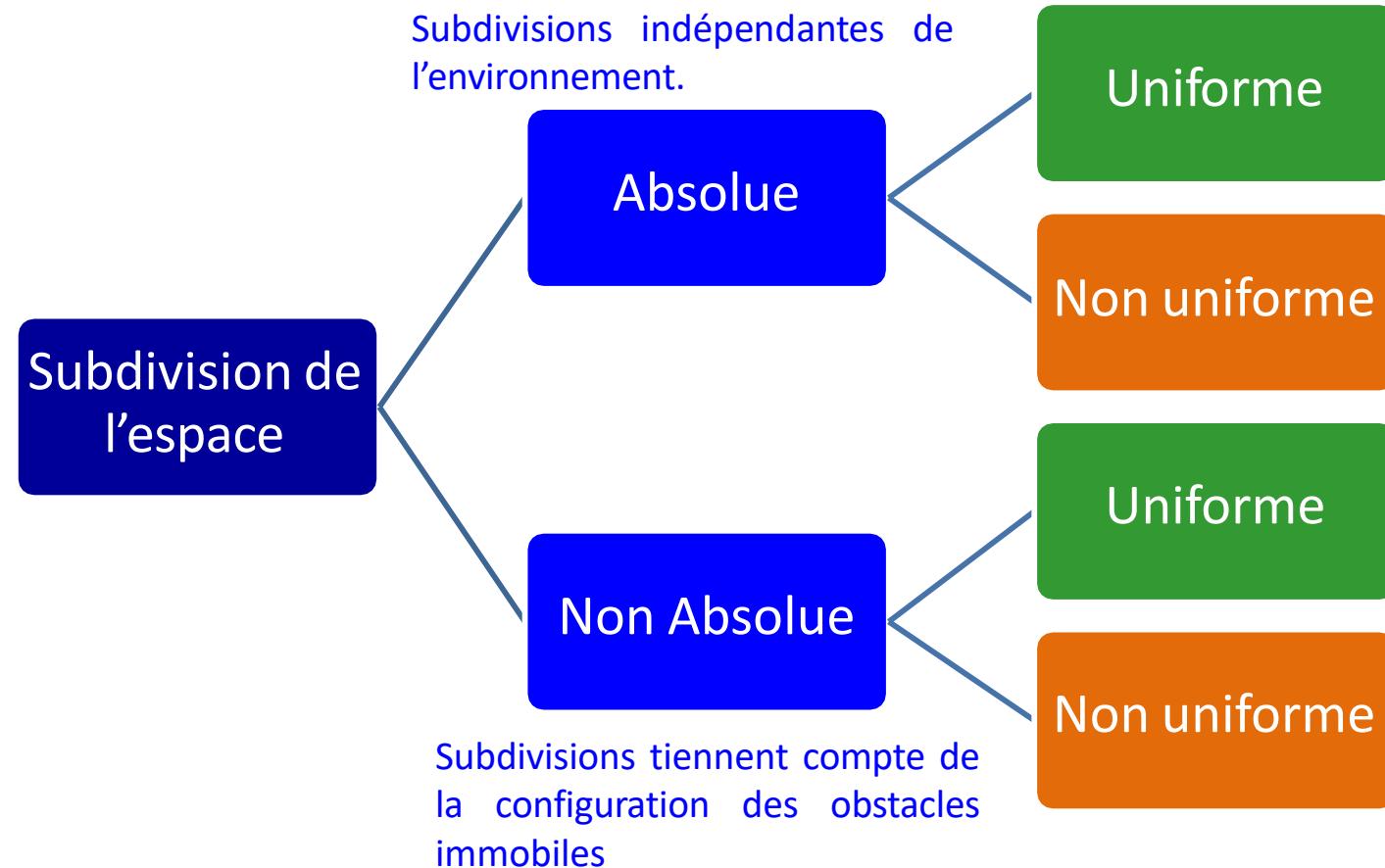
- **La logique** : deux objets étant éloignés l'un de l'autre ne peuvent vraisemblablement pas être en collision.

## Phase Large (Broad phase) : Subdivision (decoupage) spatiale

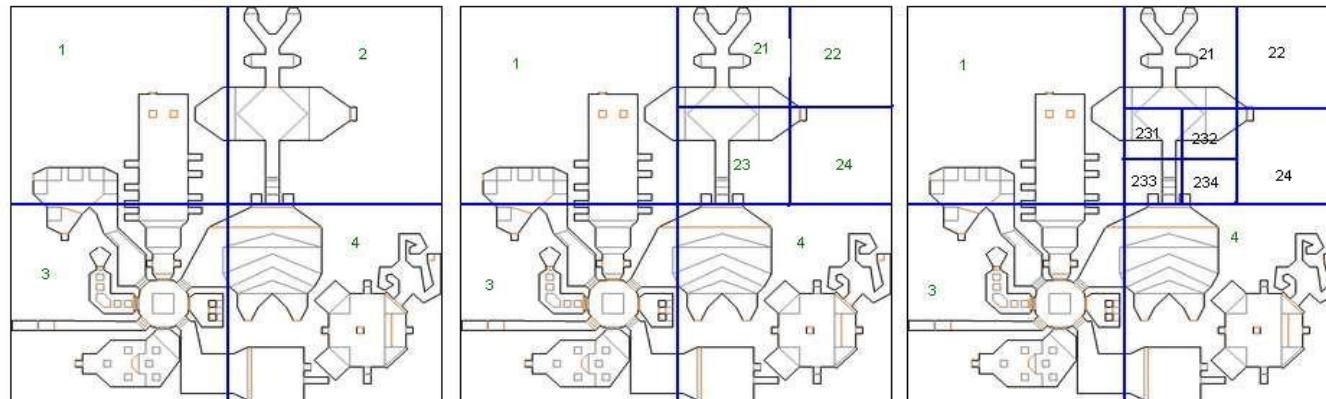


# Phase Large (Broad phase) : Subdivision (decoupage) spatiale

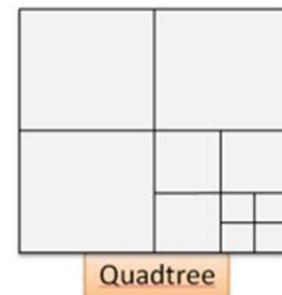
Le découpage de l'espace peut s'opérer de **deux façons** :



# Phase Large (Broad phase) : Subdivision (decoupage) spatiale

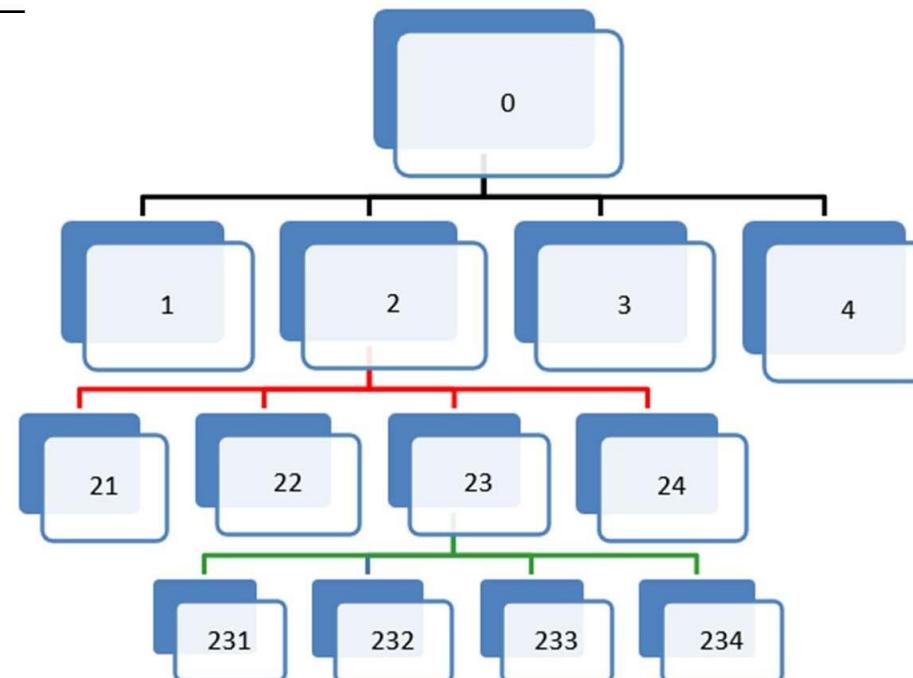


- Absolue
- Non uniforme



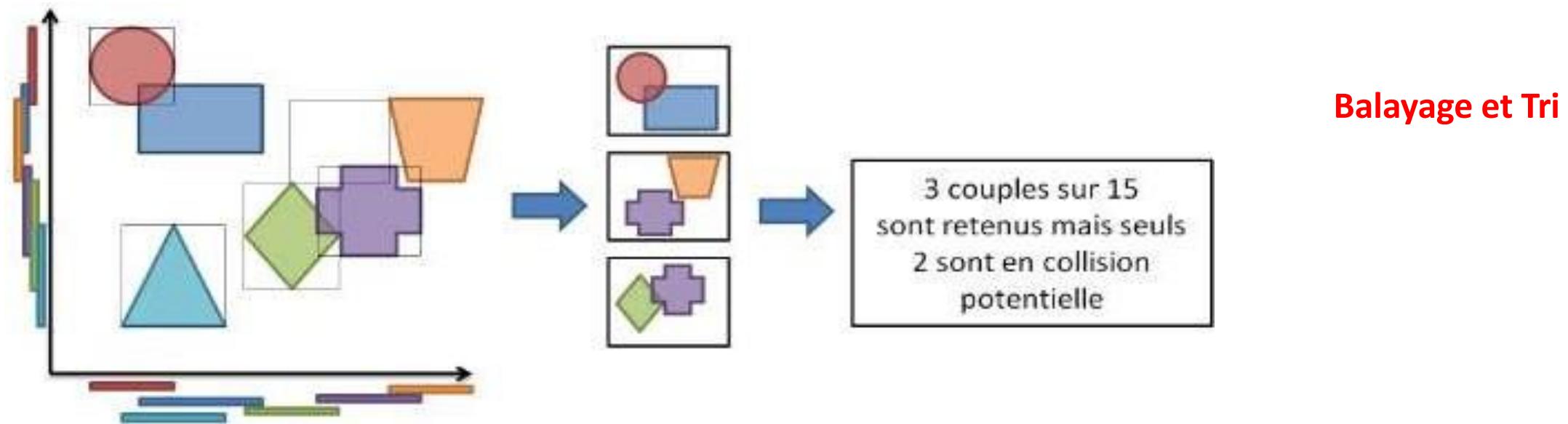
Une **hiérarchie** !

Un « Arbre »



## Phase Large (Broad phase) : Sweep and Prune (SAP)

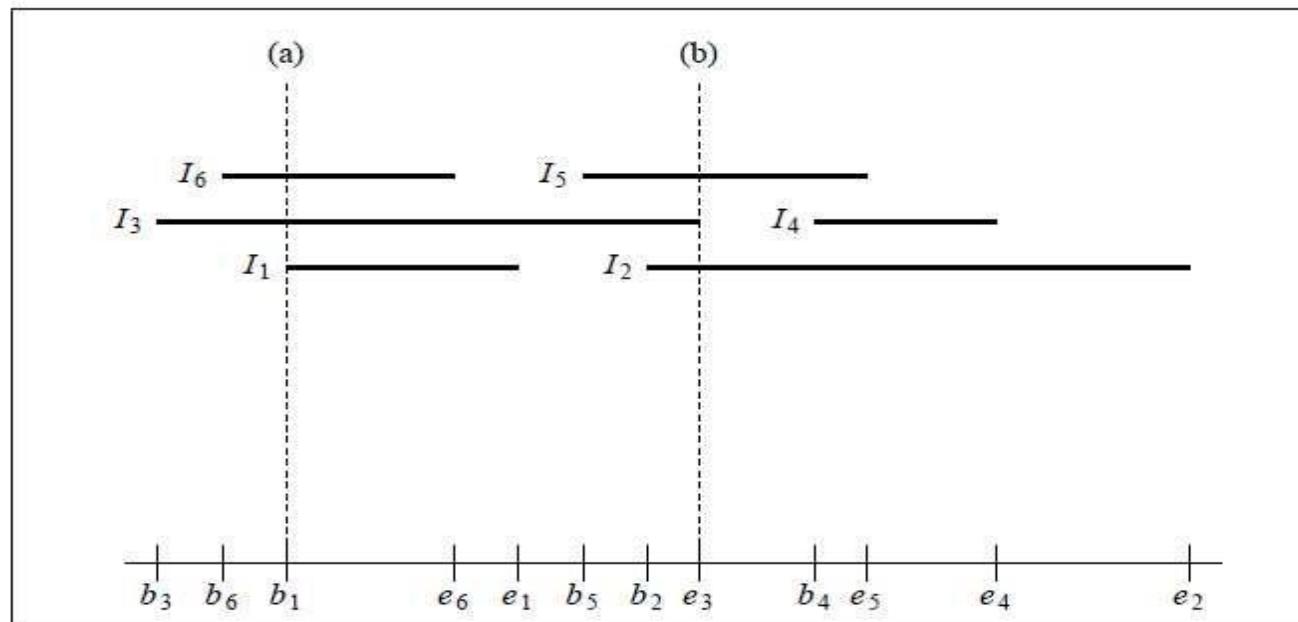
- Fournit une élimination efficace et rapide des paires d'objets et ne dépend pas de la complexité de ces derniers.
- Prend en entrée tous les objets de l'environnement et donne en sortie une liste des paires d'objets en collision potentielle.



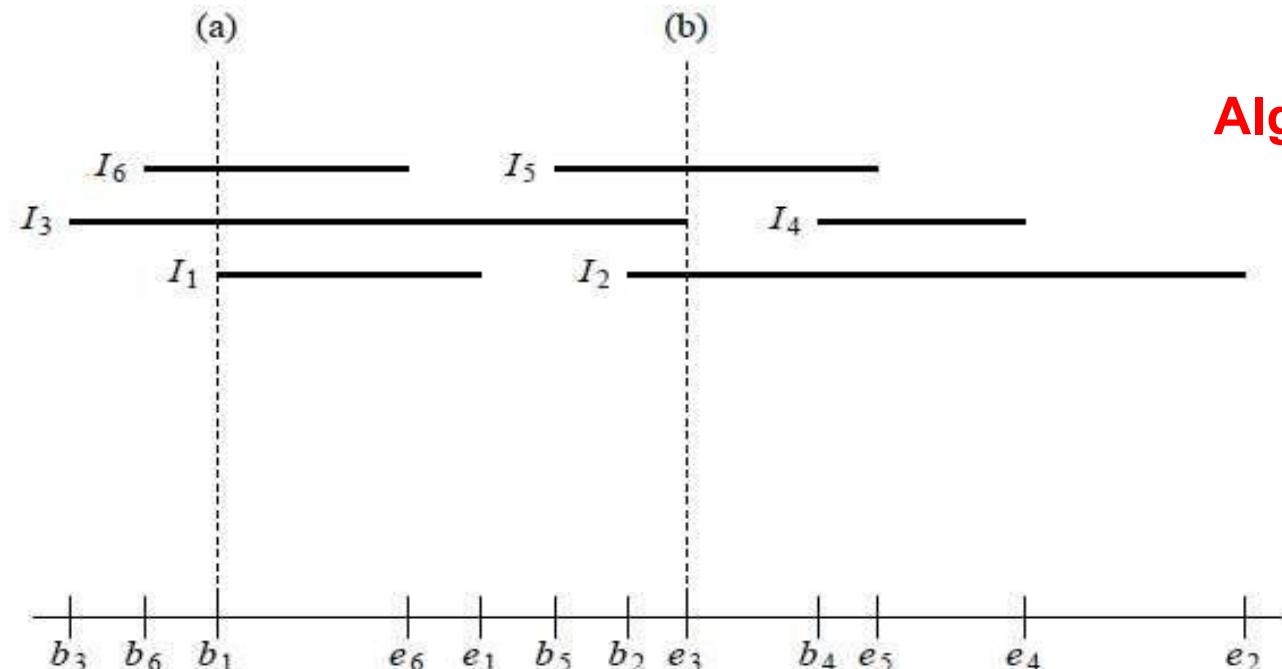
# Phase Large (Broad phase) : Sweep and Prune (SAP)

## Exemple 1D:

- Considérons  $n$  intervalles où le  $i^{\text{ème}}$  intervalle est noté  $[b_i, e_i]$ .
- Le problème consiste alors à déterminer les paires  $i, j$  tels que les intervalles  $[b_i, e_i]$  et  $[b_j, e_j]$  présentent une intersection



# Phase Large (Broad phase) : Sweep and Prune (SAP)



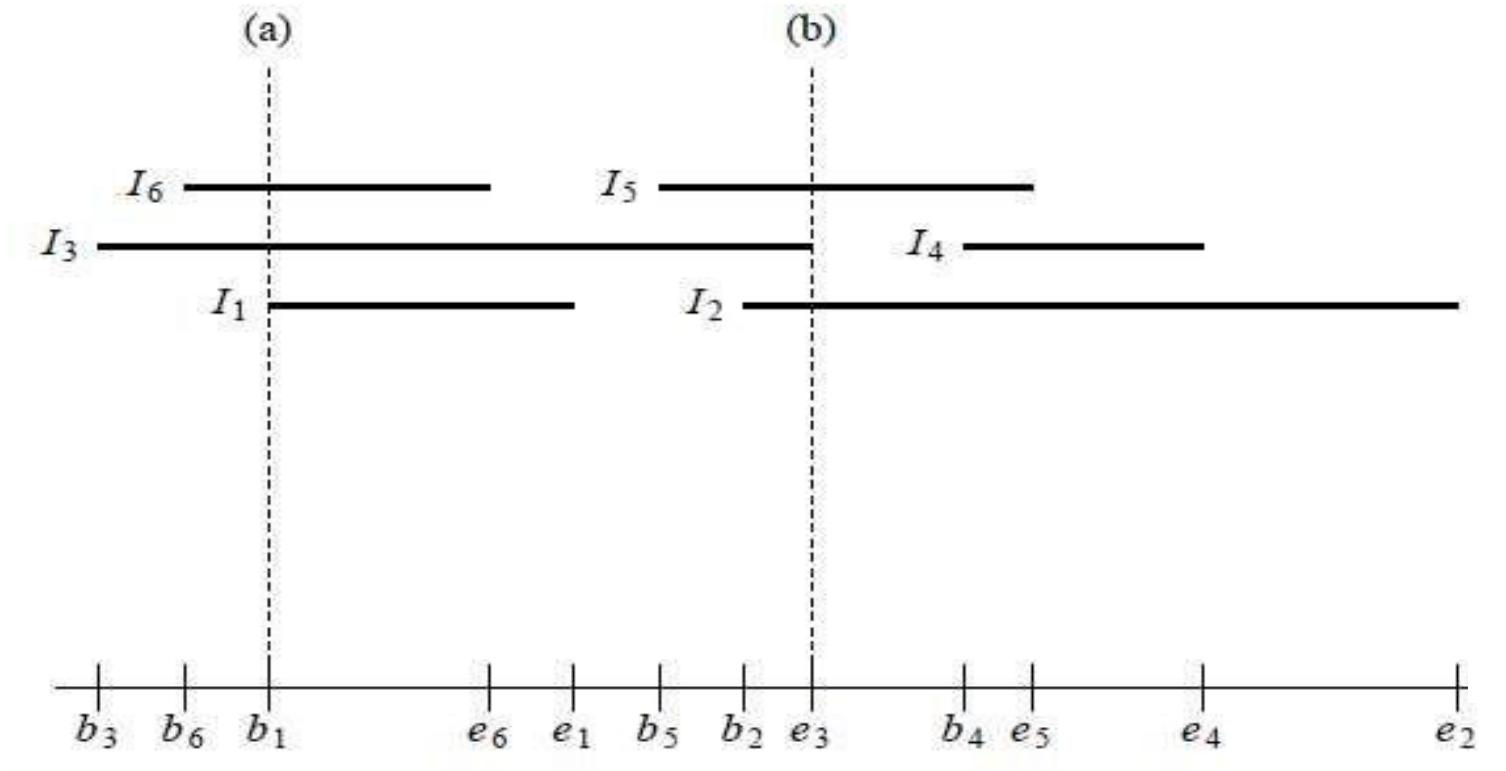
## Algorithmes de balayage et de tri

- Une liste de tous les  $b_i$  et de tous les  $e_i$  est tout d'abord créée en **les classant de la plus petite valeur vers la plus grande**.
- La **liste** est ensuite **balayée**(dans le sens croissant) et une liste, initialement vide, d'**intervalles actifs**, est maintenue.
- Quand un  $b_i$  est rencontré, l'intervalle  $i$  entre dans cette liste et si l'on rencontre le  $e_i$  correspondant, l'intervalle sort de cette liste.
- **Deux intervalles qui appartiennent à la liste active se chevauchent.**

## Phase Large (Broad phase) : Sweep and Prune (SAP)

- Application: Implémenter l'algorithme SAP et tester la structure suivante:

b3=1  
b6=2  
b1=3  
e6=6  
e1=7  
b5=7,5  
b2=7,7  
e3=8  
b4=8,9  
e5=11  
e4=11,2  
e2=18



## Phase Large (Broad phase) : Sweep and Prune (SAP)

Dans c'est beaucoup plus compliqué et nécessite des algorithmes plus performant que celui de balayage et de tri utilisé au cas 1D.

→ le cas 3D, on aura trois intervalles

$$[b_i^x, e_i^x] \quad [b_i^y, e_i^y] \quad [b_i^z, e_i^z]$$

qui correspondent aux intervalles le long de chacune des trois directions des axes de coordonnées.

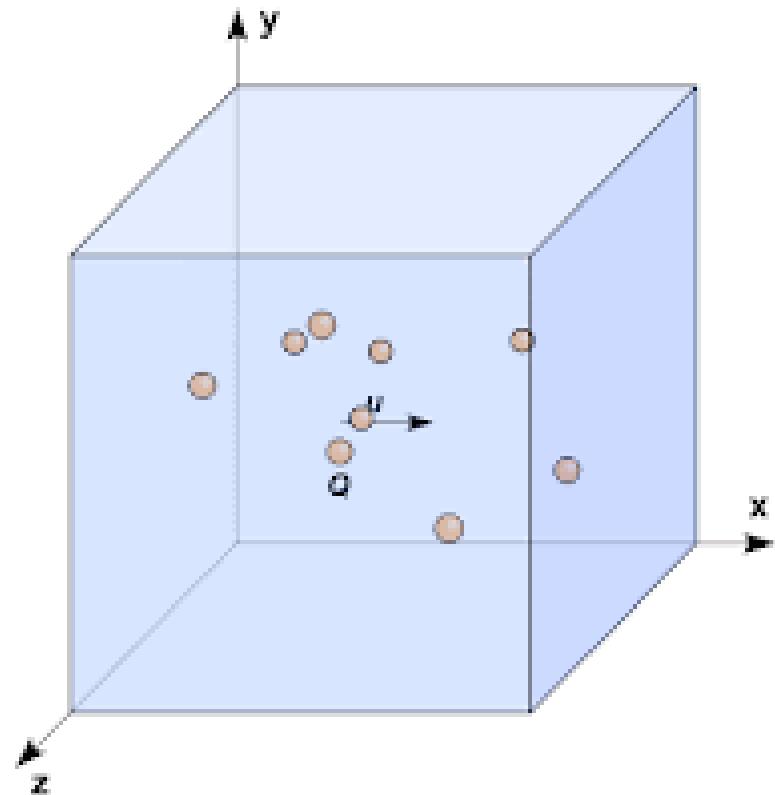
## Phase Large (Broad phase) : Méthode cinématique

- Prend en compte le mouvement des objets (si les objets s'éloignent ils ne peuvent pas entrer en collision)
- Afin de réduire le nombre de tests à exécuter, les normales des faces d'un objet sont comparées au vecteur vitesse, **les faces en sens opposé sont évincées du calcul.**

# Phase Large (Broad phase) : Application comportement des particules dans un volume confiné

Pour simuler **un gaz parfait** avec des sphères dans une boîte, nous allons modéliser le mouvement et les forces agissant sur les sphères selon les lois de la physique.

Cette simulation repose principalement sur les concepts de la **dynamique des particules**, en supposant que les interactions entre les sphères sont des **collisions élastiques (sans perte d'énergie)**.



# Phase Large (Broad phase) : Application comportement des particules dans un volume confiné

## 1. Modélisation Mathématique et Physique des Forces et des Collisions

### A. Modélisation de la Dynamique des Sphères

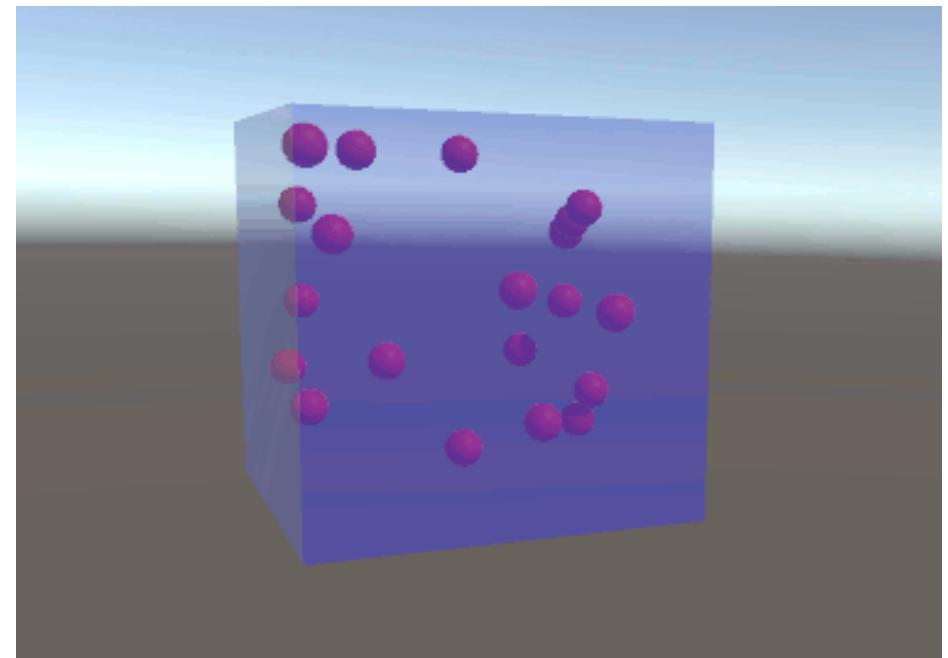
Chaque sphère de rayon  $r$ , de masse  $m$ , et de vitesse initiale  $\vec{v}$  se déplace librement à l'intérieur de la boîte. Les forces principales dans cette simulation sont :

#### 1. Force de Collision avec les Parois :

- Lorsqu'une sphère atteint une paroi, elle subit **une force normale** qui la repousse en inversant sa vitesse **perpendiculaire à la paroi**.
- Matériellement, cette force n'est pas directement simulée par un calcul, mais nous modifions la vitesse de la sphère pour simuler un rebond.

#### 2. Force de Collision entre les Sphères :

- Deux sphères entrent en collision lorsqu'elles sont à une distance de  $r_1 + r_2$ , où  $r_1$  et  $r_2$  sont leurs rayons.
- La collision est modélisée comme **élastique** : il n'y a pas de perte d'énergie, et les sphères rebondissent avec un échange de quantité de mouvement le long de la direction de collision.



# Phase Large (Broad phase) : Application comportement des particules dans un volume confiné

## 1. Physique des Forces et des Collisions

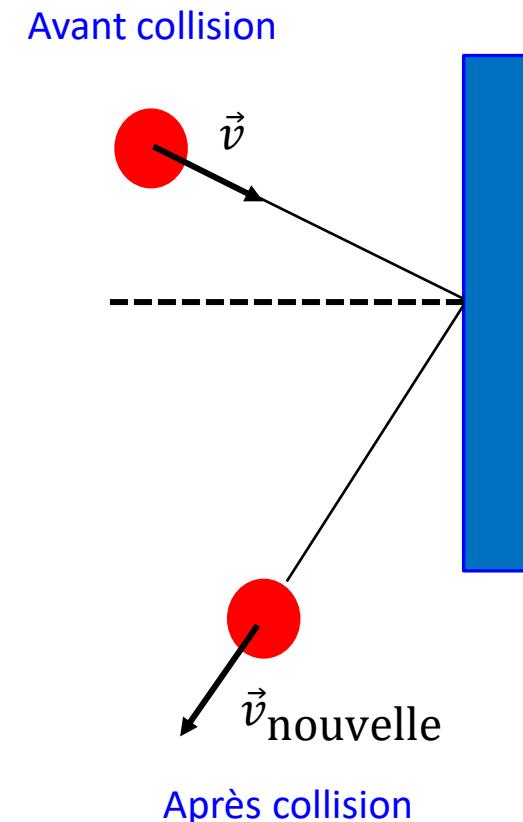
### B. Modèle Mathématique des Collisions

#### Collision avec les Parois :

- Si une sphère de position  $\vec{p}$  atteint une paroi de la boîte, la composante de sa vitesse normale à cette paroi s'inverse :

$$\vec{v}_{\text{nouvelle}} = \vec{v} - 2(\vec{v} \cdot \vec{n})\vec{n}$$

où  $\vec{n}$  est le vecteur normal à la paroi concernée.



# Phase Large (Broad phase) : Application

## comportement des particules dans un volume confiné

### Collision entre Deux Sphères :

- Si deux sphères de positions  $\vec{p}_1$  et  $\vec{p}_2$  se rapprochent à une distance  $d = r_1 + r_2$ , elles subissent une collision.

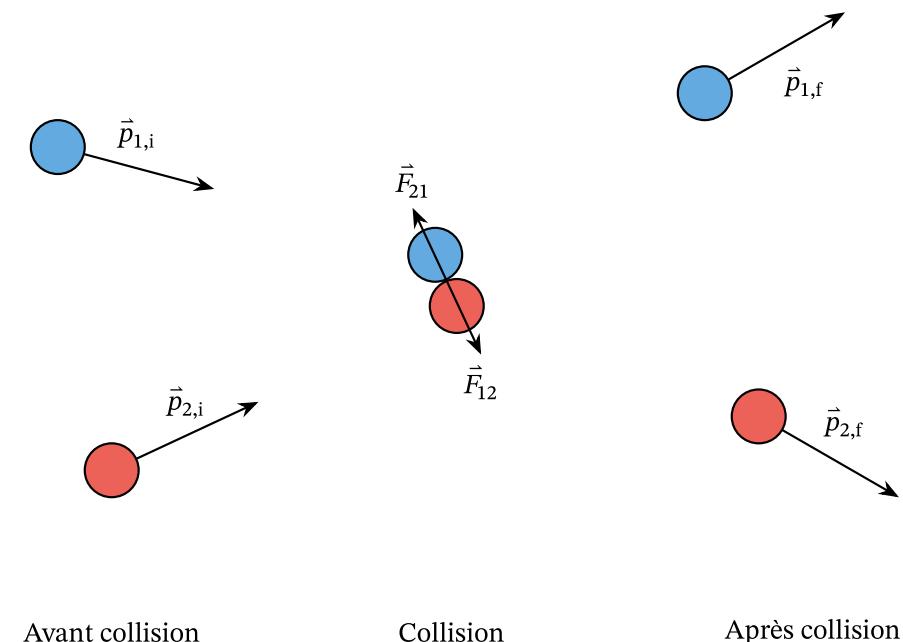
- La réaction de collision élastique est déterminée en échangeant la quantité de mouvement dans la direction de la collision. Si  $\vec{n}$  est la direction unitaire de la collision :

$$\vec{n} = \frac{\vec{p}_2 - \vec{p}_1}{|\vec{p}_2 - \vec{p}_1|}$$

- La composante de la vitesse de chaque sphère dans la direction de la collision est inversée en appliquant une quantité de mouvement égale et opposée :

$$\vec{v}_{1,\text{nouvelle}} = \vec{v}_1 - \left( 2 \frac{m_2}{m_1 + m_2} (\vec{v}_1 - \vec{v}_2) \cdot \vec{n} \right) \vec{n}$$

$$\vec{v}_{2,\text{nouvelle}} = \vec{v}_2 + \left( 2 \frac{m_1}{m_1 + m_2} (\vec{v}_1 - \vec{v}_2) \cdot \vec{n} \right) \vec{n}$$



Cela garantit que la quantité de mouvement et l'énergie cinétique sont conservées dans le système.

# Phase Large (Broad phase) : Application comportement des particules dans un volume confiné

## 2. Modèle Mathématique

La simulation repose sur trois concepts principaux : le mouvement des sphères, les collisions avec les parois, et les collisions entre sphères.

### A. Équations de Mouvement

Chaque sphère se déplace selon les équations de mouvement de la dynamique classique, où l'on néglige la gravité :

$$\vec{p}(t + \Delta t) = \vec{p}(t) + \vec{v}\Delta t$$
$$\vec{v}(t + \Delta t) = \vec{v}(t)$$

- $\vec{p}$  : Position de la sphère.
- $\vec{v}$  : Vitesse de la sphère (constante en l'absence de forces externes).

$\Delta t$  : Pas de temps pour la mise à jour de la position.

### B. Collisions avec les Parois

Pour une boîte de taille  $L = 10$ , chaque paroi est située à  $x = \pm L/2$ ,  $y = \pm L/2$ , et  $z = \pm L/2$ . Lorsqu'une sphère touche une paroi, elle inverse la composante de vitesse correspondant à cette direction.

Si  $r$  est le rayon de la sphère, et  $p_x, p_y, p_z$  sa position actuelle, les conditions de collision sont :

- Paroi X :  $p_x - r \leq -L/2$  ou  $p_x + r \geq L/2$
- Paroi Y :  $p_y - r \leq -L/2$  ou  $p_y + r \geq L/2$
- Paroi Z :  $p_z - r \leq -L/2$  ou  $p_z + r \geq L/2$

Lorsque l'une de ces conditions est vérifiée, la vitesse de la sphère est inversée dans cette direction :

$$\vec{v}_{\text{nouvelle}} = -\vec{v}_{\text{ancienne}}$$

# Phase Large (Broad phase) : Application comportement des particules dans un volume confiné

## 2. Modèle Mathématique

### C. Collisions entre Sphères

Lorsqu'une sphère  $A$  de position  $\vec{p}_A$  et une sphère  $B$  de position  $\vec{p}_B$  se rapprochent suffisamment pour que :

$$|\vec{p}_A - \vec{p}_B| \leq r_A + r_B$$

une collision est détectée, et leurs vitesses sont ajustées pour simuler une collision élastique. Les nouvelles vitesses après collision sont calculées en utilisant les équations de conservation de l'énergie cinétique et de la quantité de mouvement.

Les nouvelles vitesses  $\vec{v}_A'$  et  $\vec{v}_B'$  sont données par :

$$\begin{aligned}\vec{v}_A' &= \vec{v}_A - \frac{2m_B}{m_A + m_B} \frac{\langle \vec{v}_A - \vec{v}_B, \vec{p}_A - \vec{p}_B \rangle}{|\vec{p}_A - \vec{p}_B|^2} (\vec{p}_A - \vec{p}_B) \\ \vec{v}_B' &= \vec{v}_B - \frac{2m_A}{m_A + m_B} \frac{\langle \vec{v}_B - \vec{v}_A, \vec{p}_B - \vec{p}_A \rangle}{|\vec{p}_B - \vec{p}_A|^2} (\vec{p}_B - \vec{p}_A)\end{aligned}$$

où : -  $\langle \vec{u}, \vec{v} \rangle$  représente le produit scalaire des vecteurs  $\vec{u}$  et  $\vec{v}$ . -  $m_A$  et  $m_B$  sont les masses des sphères (supposées identiques ici, donc la formule se simplifie).

# Phase Large (Broad phase) : Application comportement des particules dans un volume confiné

## 3. Algorithme de Simulation

Le cycle de simulation pour chaque mise à jour de frame suit ces étapes :

1. **Mise à jour des positions** : Appliquez les équations de mouvement pour toutes les sphères.
2. **Détection des collisions** :
  - Pour les parois : vérifiez si chaque sphère dépasse les limites de la boîte et, si c'est le cas, ajustez sa vitesse.
  - Pour les sphères entre elles : utilisez un algorithme de balayage pour détecter rapidement les collisions (par exemple, Sweep and Prune).
3. **Réponse aux collisions** : Corrigez les positions et ajustez les vitesses des sphères pour les collisions détectées.

## 4. Implémentation Algorithme : Sweep and Prune (SAP)

Le SAP optimise la détection de collision en triant les sphères sur chaque axe pour éviter de tester toutes les paires possibles :

1. **Projet de chaque sphère sur chaque axe** :
  - On prend les valeurs minimales et maximales de chaque sphère selon les axes  $x$ ,  $y$ , et  $z$ .
2. **Tri et détection de recouvrement** :
  - Sur chaque axe, triez les sphères par leurs coordonnées de début et de fin. Si deux sphères se chevauchent sur un axe, vérifiez si elles se chevauchent également sur les deux autres axes. Si c'est le cas, elles sont en collision.
3. **Collision Handling** :
  - Une fois les collisions détectées, appliquez les équations de collision élastique pour les vitesses des sphères impliquées.

