

# Moteurs de jeux

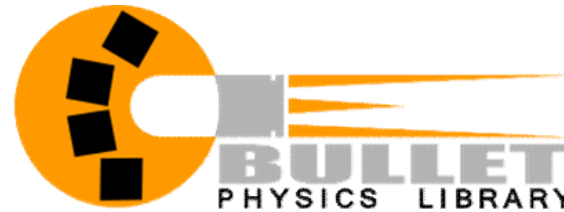
Moteur physique

Source cours d'Alexis Vaisse (Ubisoft) et Nicolas Pronost (Université d'Utrecht)

# Moteurs physiques existants

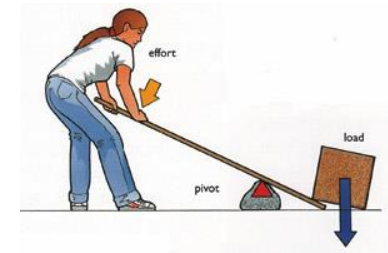
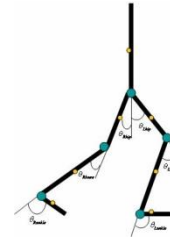
- Havok Physics
- nVidia PhysX
- Box2D
- Newton
- Tokamak
- ...

PhysX™



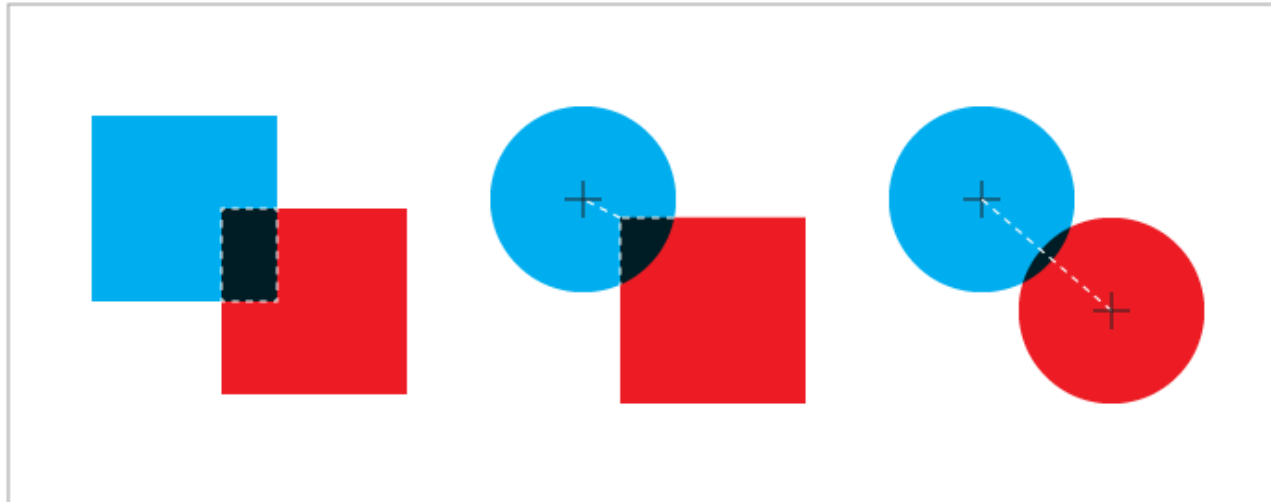
# Considérations principales

- Déplaçons les objets virtuels
  - Cinématique : description du mouvement
    - considère la position, la vitesse et l'accélération
  - Physique : l'effet des forces sur le mouvement
    - considère la masse, l'inertie et plus
- Avec la physique, les objets peuvent interagir par le biais de forces
  - Pas besoin de pré-calculer / scripter les mouvements / interactions
  - Réagit nativement selon les lois de physique (implémentées)
  - Bon pour les jeux !



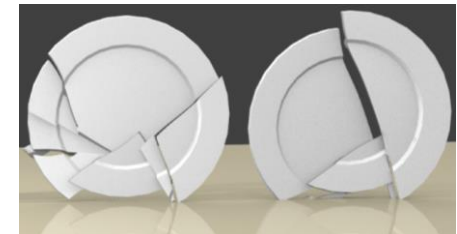
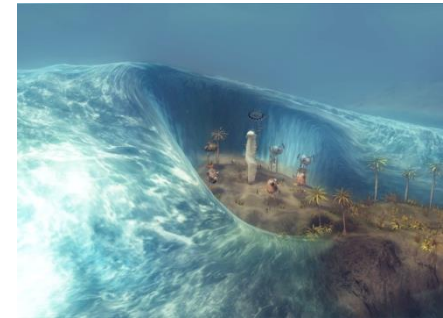
# Considérations principales

- Comment controller les forces pour réaliser un effet ?
- Les objets bougent, ils peuvent aussi entrer en collision
  - **Détection de collision** : est-ce que les objets sont entrés en collision et où ?
  - **Résolution de la collision resolution** : que fait t'on ensuite?

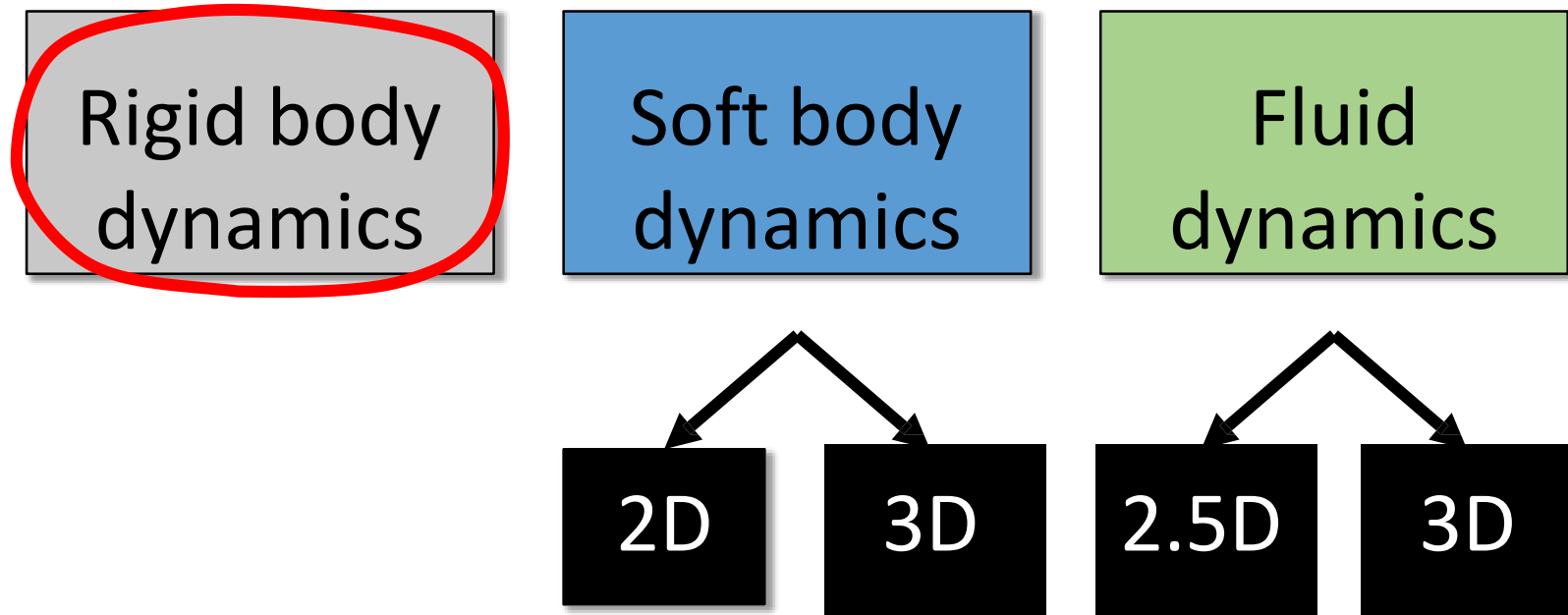


# Applications

- Nous pouvons utiliser la physique pour simuler
  - corps rigides
    - considérés mécaniquement comme des points
    - éventuellement connecté à d'autres organismes
  - corps mous
    - peut se déformer dans un continuum
    - éventuellement en interaction avec d'autres organismes
  - corps cassables
    - agissant comme des corps rigides simples jusqu'à ce que certains événements les divisent en plusieurs



# Que fait un moteur physique ?



# Modélisation d'objets

- En mécanique des corps solides, un objet est représenté par un point dans l'espace
  - **Centroïde** de l'objet
    - moyenne de toutes les positions décrivant l'objet
  - **Centre de masse** de l'objet
    - moyenne pondérée de la densité, identique au centroïde s'il est uniforme
- En mécanique des corps mous, un objet est représenté par un ensemble structuré de points dans l'espace
- La géométrie / limite d'un objet est utilisée dans la détection et la résolution des collisions, pas dans la mécanique

# Cinématique

- Pour déterminer la position d'un objet  $p_o$  à un temps  $t$
- Supposons la Vitesse constante  $v$ 
  - Alors  $p_o(t + \Delta t) = p_o(t) + v\Delta t$
  - Alternativement,  $\Delta p_o = p_o(t + \Delta t) - p_o(t) = v\Delta t$
  - Si  $p_o(0) = P$ , alors  $p_o(t) = P + vt$
  - Alors nous pouvons calculer la position des objets à chaque instant
- Mais  $v$  n'est surement pas constant
  - La Vitesse est une fonction du temps  $v(t)$  répondant aux forces extérieures, alors  $p_o(s) = P + \int_0^s v(t) dt$
  - La position doit être calculée régulièrement pour prendre en compte les changements de vitesse



# Cinématique

- La même propriété est valable pour l'accélération  $a$ 
  - Si l'accélération est constante, alors  $\Delta v = a\Delta t$
  - Si non, alors  $v(s) = V + \int_0^s a(t) dt$
- Doit-on vraiment intégrer?
  - Si on recalcule la position à chaque game loop appelée après  $\Delta t$
  - Et on suppose que la Vitesse et l'accélération sont constantes sur cet interval de temps (typiquement très court, quelques millisecondes ou moins)
  - On peut alors utiliser la formule

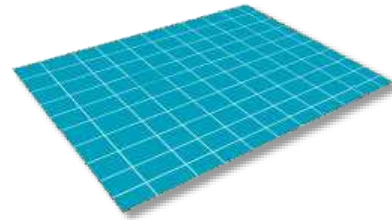
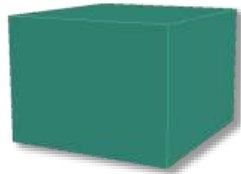
$$p_o(t + \Delta t) = p_o(t) + v\Delta t$$

et

$$v(t + \Delta t) = v(t) + a\Delta t$$

# La physique des corps rigides indéformables

**Primitives :**



**Meshes :**

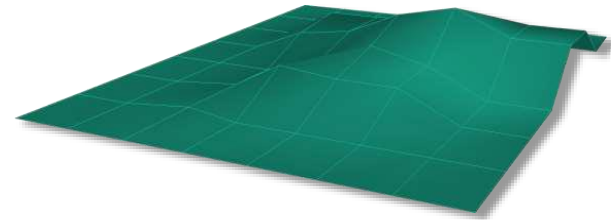


Convexe

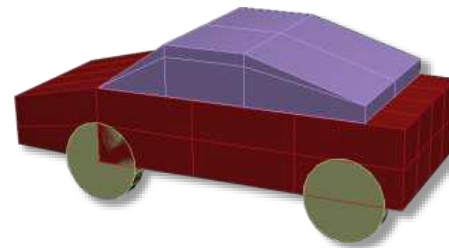


Quelconque

**Heightfield :**



**Composés :**



# La physique des corps rigides indéformables

Mission n° 1 :

Détecter les collisions

Mission n° 2 :

Résoudre les contraintes

Contraintes issues  
des collisions

Contraintes issues  
des joints

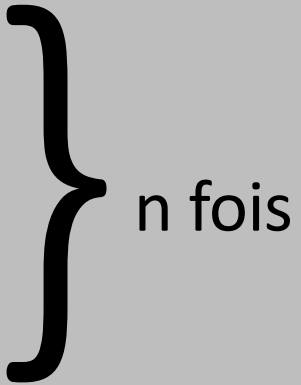
# La physique des corps rigides indéformables

- Partie 1 : La détection des collisions

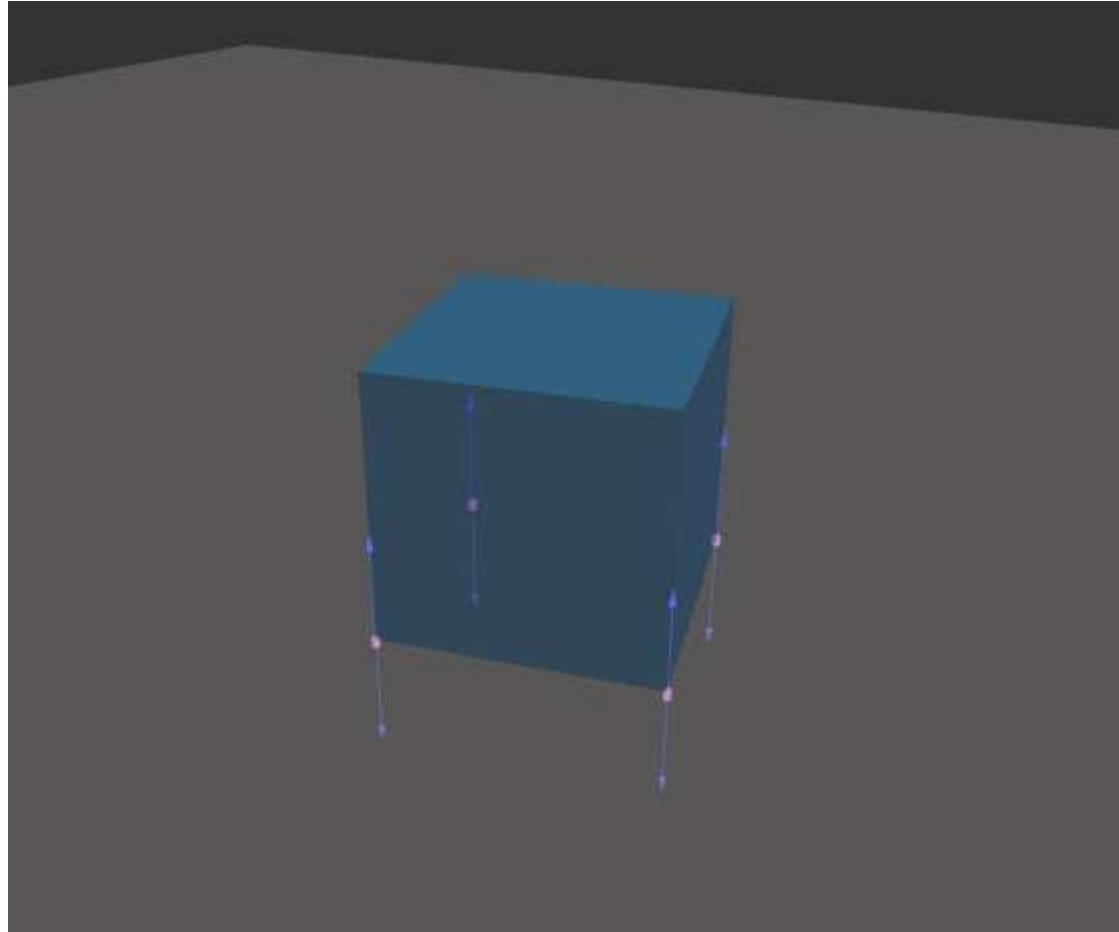
# La détection des collisions

## Objectifs :

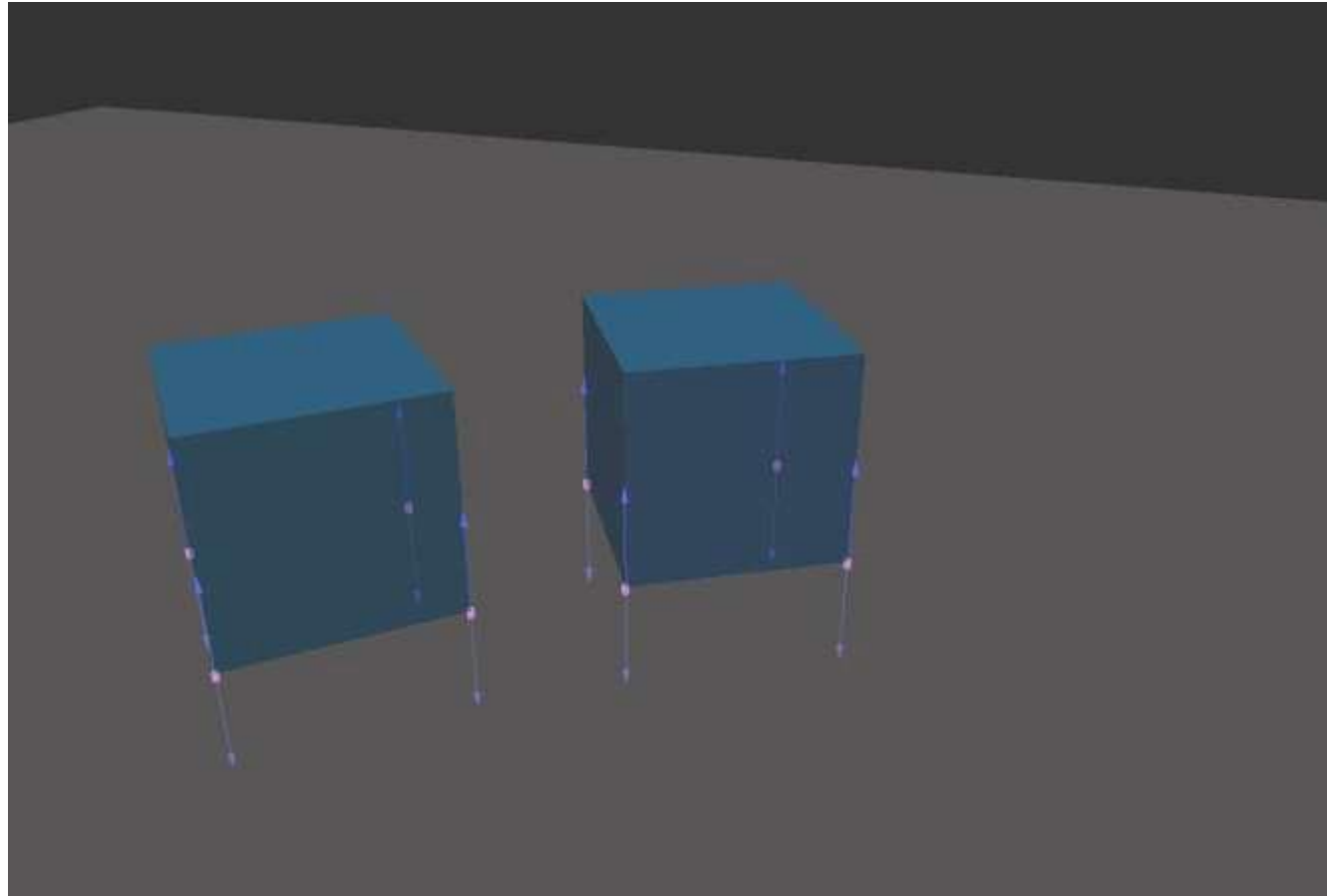
- Fournir la liste des paires de corps qui sont en intersection
- Pour chaque paire de corps :

- Identifiant du corps 1
  - Identifiant du corps 2
  - Point de contact sur le corps 1
  - Point de contact sur le corps 2
  - Normale
  - Distance de pénétration
- 
- n fois

# La détection des collisions



# La détection des collisions



# La détection des collisions

Phase n° 1 :

**Broad phase**

Détecter les paires de corps qui sont **potentiellement** en collision

Phase n° 2 :

**Narrow phase**

Déterminer si deux corps sont **réellement** en collision. Si oui, calculer les points de collision.



# La détection des collisions / Broad phase

Mission de la broad phase :

Détecter les paires de corps qui sont **potentiellement** en collision

On n'utilise pas la géométrie exacte des corps

- Pourquoi ? ➡ Pour des raisons de performance
- Qu'utilise-t-on à la place ? ➡ Des volumes englobants

# Broad phase / Volumes englobants

Volume englobant :

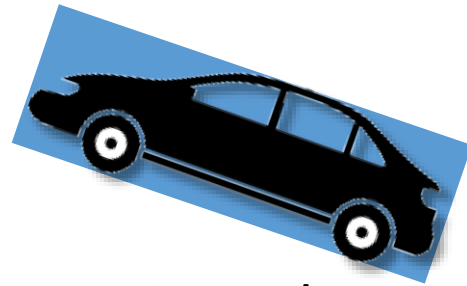
- Englobe entièrement le corps
  - Possède une géométrie plus simple qui permettra des calculs plus rapides
- 
- Si le volume englobant du corps A intersecte le volume englobant du corps B, alors les corps A et B sont **potentiellement** en collision.
  - Si le volume englobant du corps A n'intersecte pas le volume englobant du corps B, on a la garantie que les corps A et B **ne sont pas** en collision.

# Broad phase / Volumes englobants

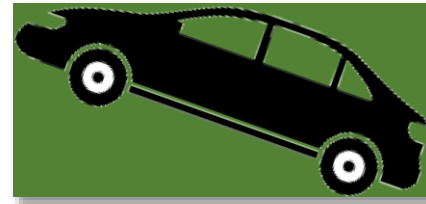
Quelques types de volumes englobants :



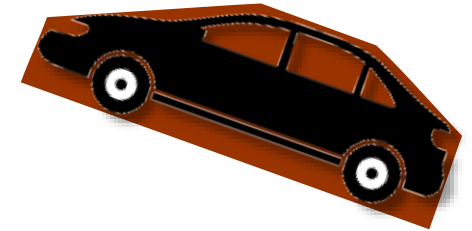
Bounding  
sphere



Oriented  
Bounding Box  
(OBB)


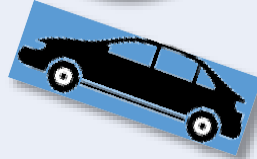

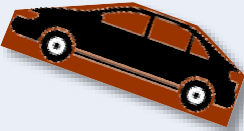


Axis-Aligned  
Bounding Box  
(AABB)


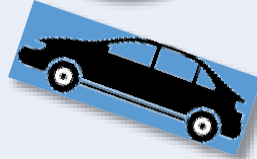
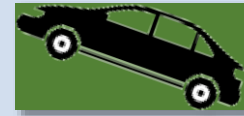
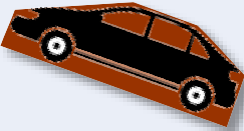


Convex


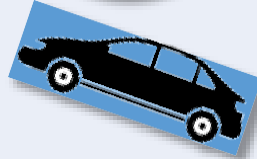
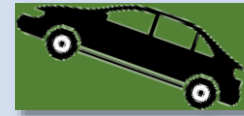
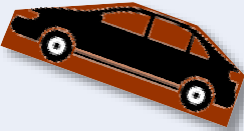
# Broad phase / Volume englobant

Type de volume englobant		
	Bounding sphere	
	Oriented Bounding Box	
	Axis-Aligned Bounding Box	
	Convex	


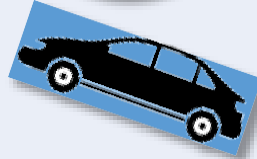
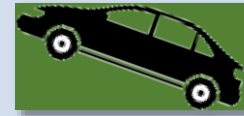
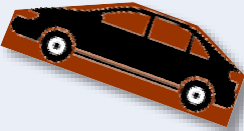
# Broad phase / Volume englobant

Type de volume englobant	Englobe le corps au plus juste	
 Bounding sphere		
 Oriented Bounding Box		
 Axis-Aligned Bounding Box		
 Convex		


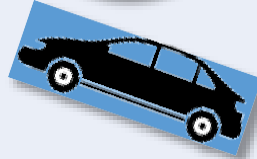

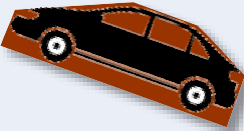
# Broad phase / Volume englobant

Type de volume englobant	Englobe le corps au plus juste	Rapidité des calculs d'intersection
 Bounding sphere		
 Oriented Bounding Box		
 Axis-Aligned Bounding Box		
 Convex		

# Broad phase / Volume englobant


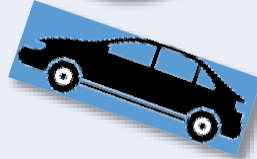
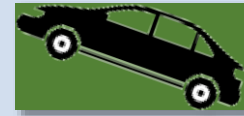
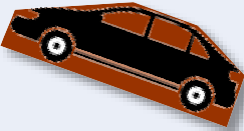
Type de volume englobant	Englobe le corps au plus juste	Rapidité des calculs d'intersection
 Bounding sphere	—	
 Oriented Bounding Box		
 Axis-Aligned Bounding Box		
 Convex		

# Broad phase / Volume englobant


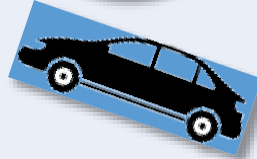

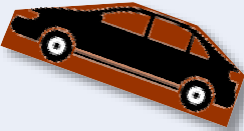
Type de volume englobant	Englobe le corps au plus juste	Rapidité des calculs d'intersection
 Bounding sphere	-	++
 Oriented Bounding Box		
 Axis-Aligned Bounding Box		
 Convex		




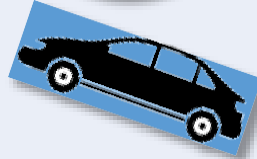

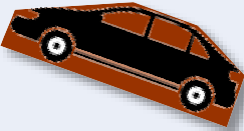
# Broad phase / Volume englobant

Type de volume englobant	Englobe le corps au plus juste	Rapidité des calculs d'intersection
 Bounding sphere	-	++
 Oriented Bounding Box	++	
 Axis-Aligned Bounding Box		
 Convex		


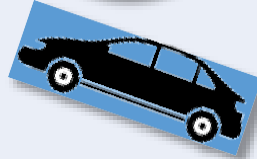

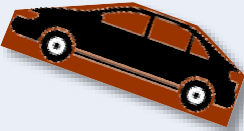
# Broad phase / Volume englobant

Type de volume englobant	Englobe le corps au plus juste	Rapidité des calculs d'intersection
 Bounding sphere	-	++
 Oriented Bounding Box	++	-
 Axis-Aligned Bounding Box		
 Convex		


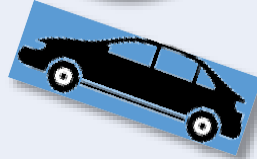

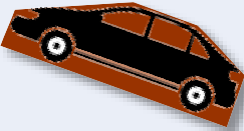
# Broad phase / Volume englobant

Type de volume englobant	Englobe le corps au plus juste	Rapidité des calculs d'intersection
 Bounding sphere	-	++
 Oriented Bounding Box	++	-
 Axis-Aligned Bounding Box	+	
 Convex		


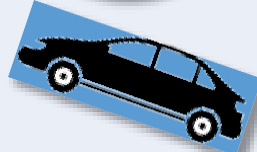
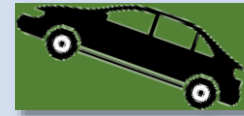
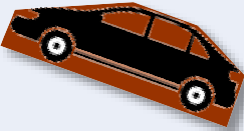
# Broad phase / Volume englobant

Type de volume englobant	Englobe le corps au plus juste	Rapidité des calculs d'intersection
 Bounding sphere	-	++
 Oriented Bounding Box	++	-
 Axis-Aligned Bounding Box	+	++
 Convex		

# Broad phase / Volume englobant

Type de volume englobant	Englobe le corps au plus juste	Rapidité des calculs d'intersection
 Bounding sphere	-	++
 Oriented Bounding Box	++	-
 Axis-Aligned Bounding Box	+	++
 Convex	+++	

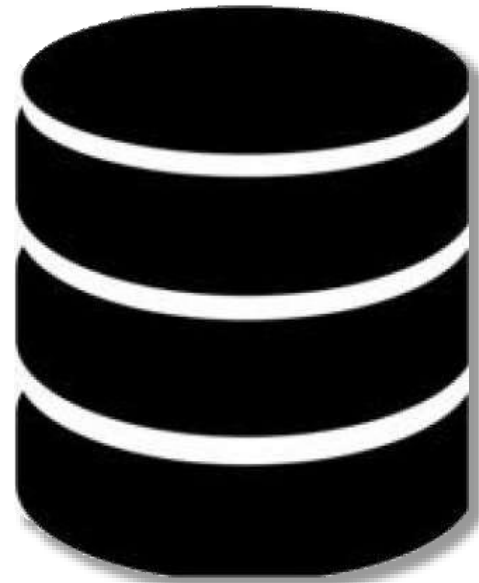
# Broad phase / Volume englobant

Type de volume englobant	Englobe le corps au plus juste	Rapidité des calculs d'intersection
 Bounding sphere	-	++
 Oriented Bounding Box	++	-
 Axis-Aligned Bounding Box	+	++
 Convex	+++	--

# Broad phase

Stockage des paires de corps  
potentiellement en collision

Broad phase



- Listes chaînées
- Map
- Hash table

# Broad phase

Quelques algorithmes de broad phase :

1. Algorithme « brute force »
2. Sweep & Prune
3. Les grilles
4. Les arbres



# Broad phase

Quelques algorithmes de broad phase :

1. Algorithme « brute force »
2. Sweep & Prune
3. Les grilles
4. Les arbres

# Broad phase

## Algorithme « brute force » :

- On teste chaque paire de volumes englobants
- Nombre de paires =  $\frac{n \cdot (n - 1)}{2}$
- Complexité =  $O(n^2)$
- Bien adapté pour  $0 \leq n \leq \sim 20$
- Inutilisable en pratique dès que  $n$  devient grand

# Broad phase

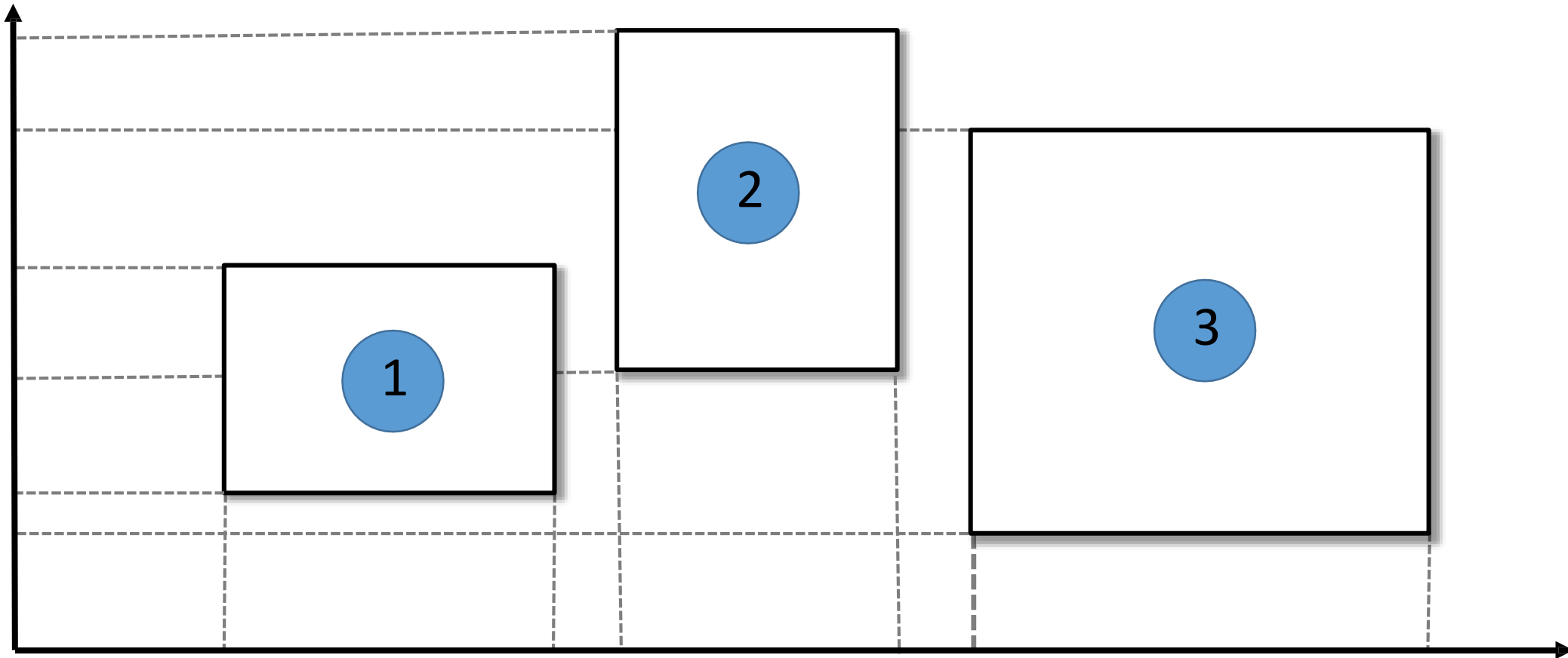
Quelques algorithmes de broad phase :

1. Algorithme « Brute force »
2. Sweep & Prune
3. Les grilles
4. Les arbres

# Broad phase / Sweep & Prune

Algorithme **Sweep & Prune** :

(1992)



# Broad phase / Sweep & Prune

```
struct Item
{
    unsigned int BodyId : 31;
    unsigned int Type   : 1;
    float        Value;
};
```

} 64 bits

```
enum Type
{
    Start = 0;
    End   = 1;
};
```

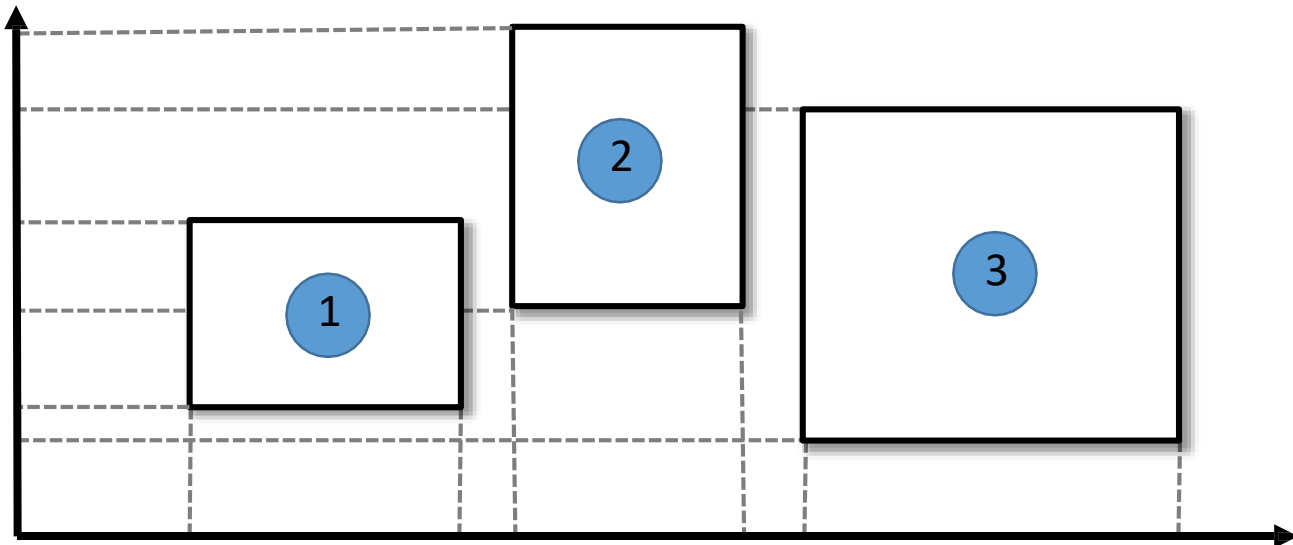
# Broad phase / Sweep & Prune

Axe x : 

1;Start;2.0	1;End;4.5	2;Start;5.0	2;End;7.0	3;Start;8.0	3;End;11.0
-------------	-----------	-------------	-----------	-------------	------------

Axe y : 

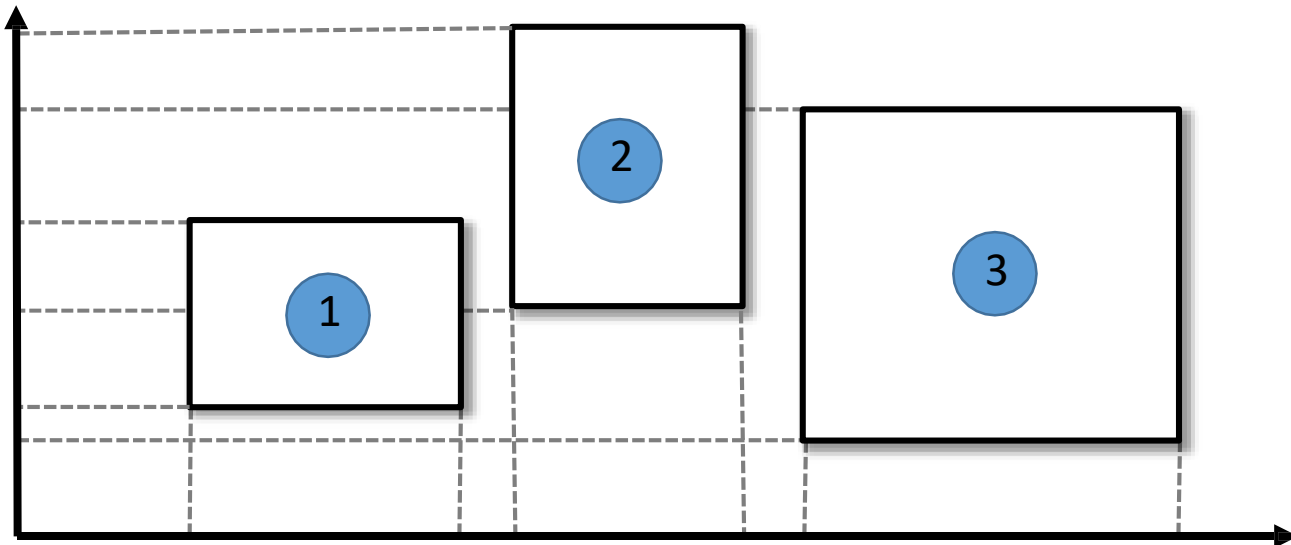
3;Start;1.0	1;Start;1.2	2;Start;2.0	1;End;3.0	3;End;4.0	2;End;4,5
-------------	-------------	-------------	-----------	-----------	-----------



# Broad phase / Sweep & Prune

Axe x : 1;Start;**3.0**    1;End;**5.5**    2;Start;5.0    2;End;7.0    3;Start;8.0    3;End;11.0

Axe y : 3;Start;1.0    1;Start;1.2    2;Start;2.0    1;End;3.0    3;End;4.0    2;End;4,5



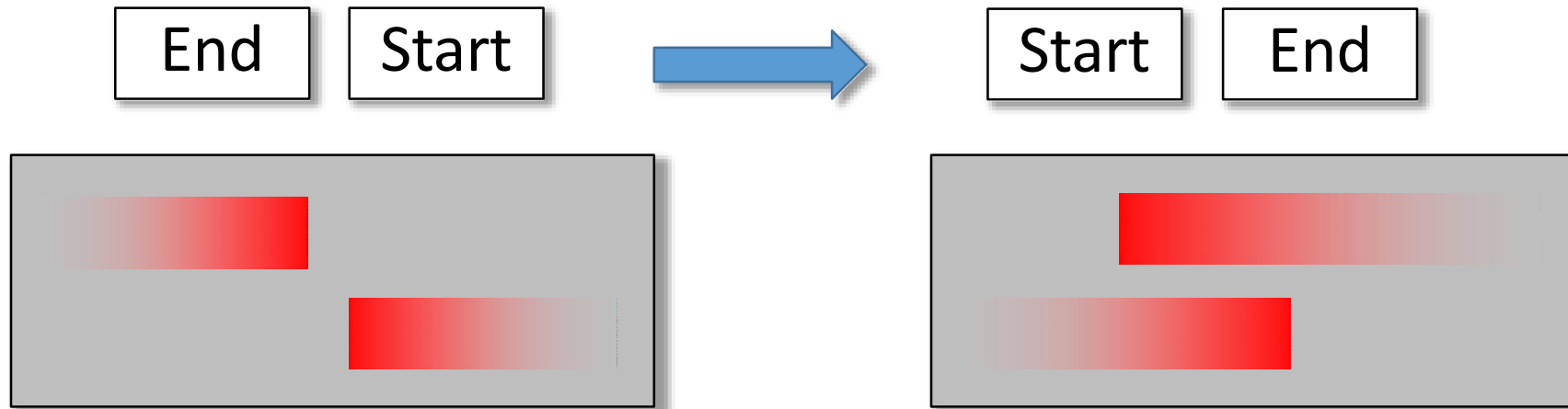
# Broad phase / Sweep & Prune

Pour chacun des axes, on parcourt l'ensemble des éléments :

1. On met à jour la valeur de l'élément
2. On compare la valeur de l'élément en cours à la valeur de l'élément précédent
3. Si cette valeur est inférieure, on échange les deux éléments

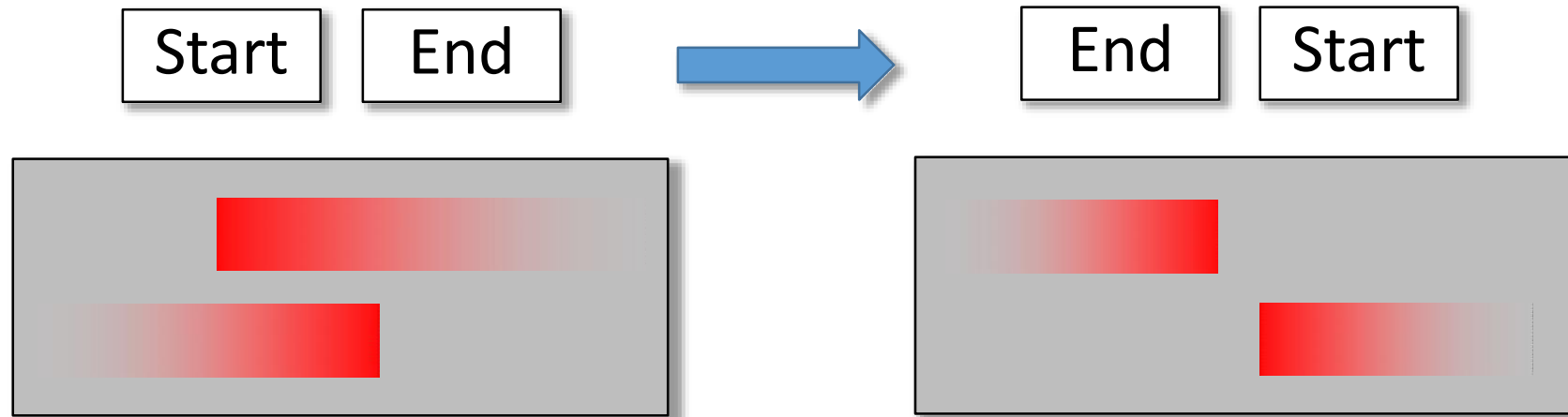


# Broad phase / Sweep & Prune



➡ Si intersection sur les autres axes :  
On ajoute la nouvelle paire


# Broad phase / Sweep & Prune




➡ Si intersection sur les autres axes :  
On enlève la paire

# Broad phase / Sweep & Prune

Complexité de la mise à jour à chaque trame :

- Pire des cas :  $O(n^2)$  (pas de cohérence d'une trame à l'autre)
- En pratique :  $\sim O(n)$  (grâce à la cohérence temporelle) 

Complexité de l'ajout d'un élément :  $O(n)$  



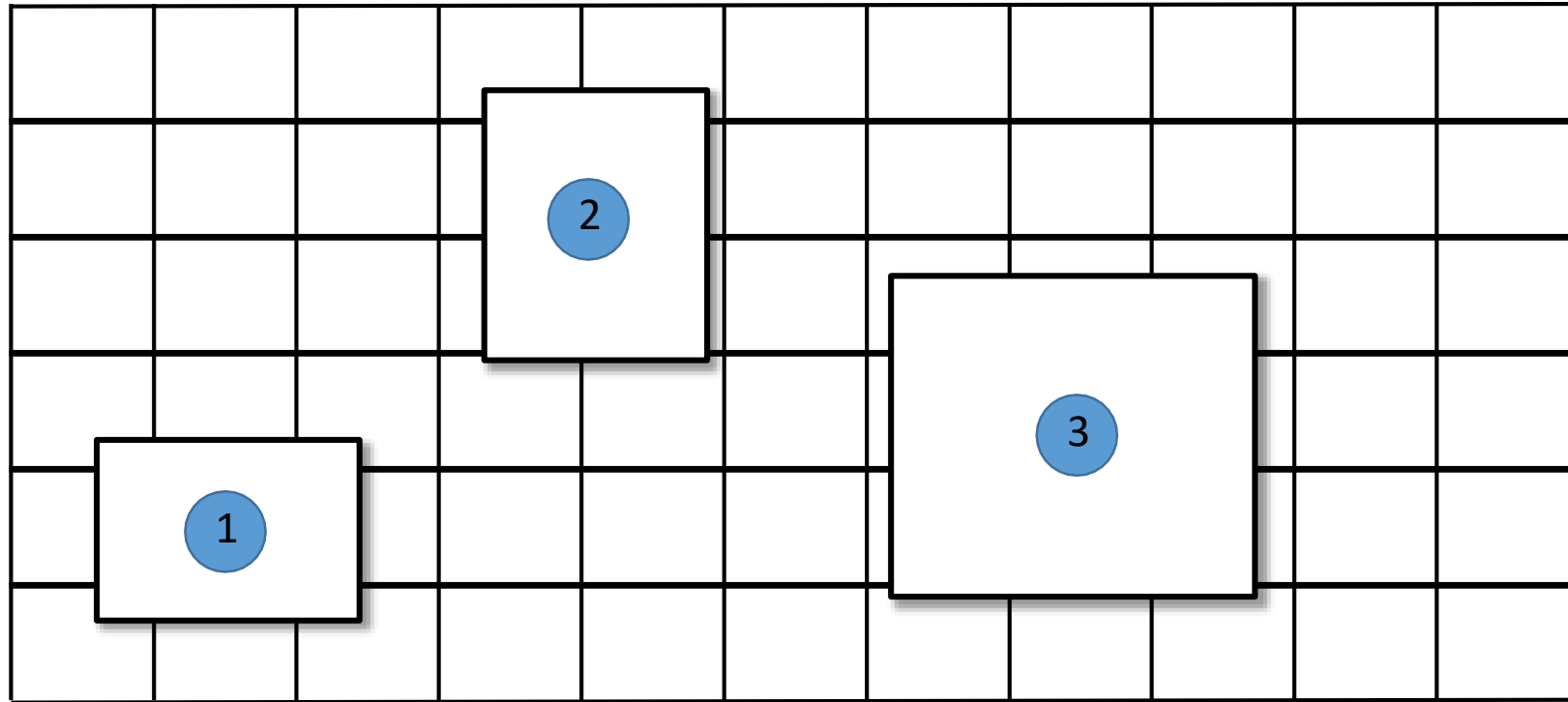
Pour améliorer, on va ajouter les éléments par paquets

# Broad phase

Quelques algorithmes de broad phase :

1. Algorithme « Brute force »
2. Sweep & Prune
3. Les grilles
4. Les arbres

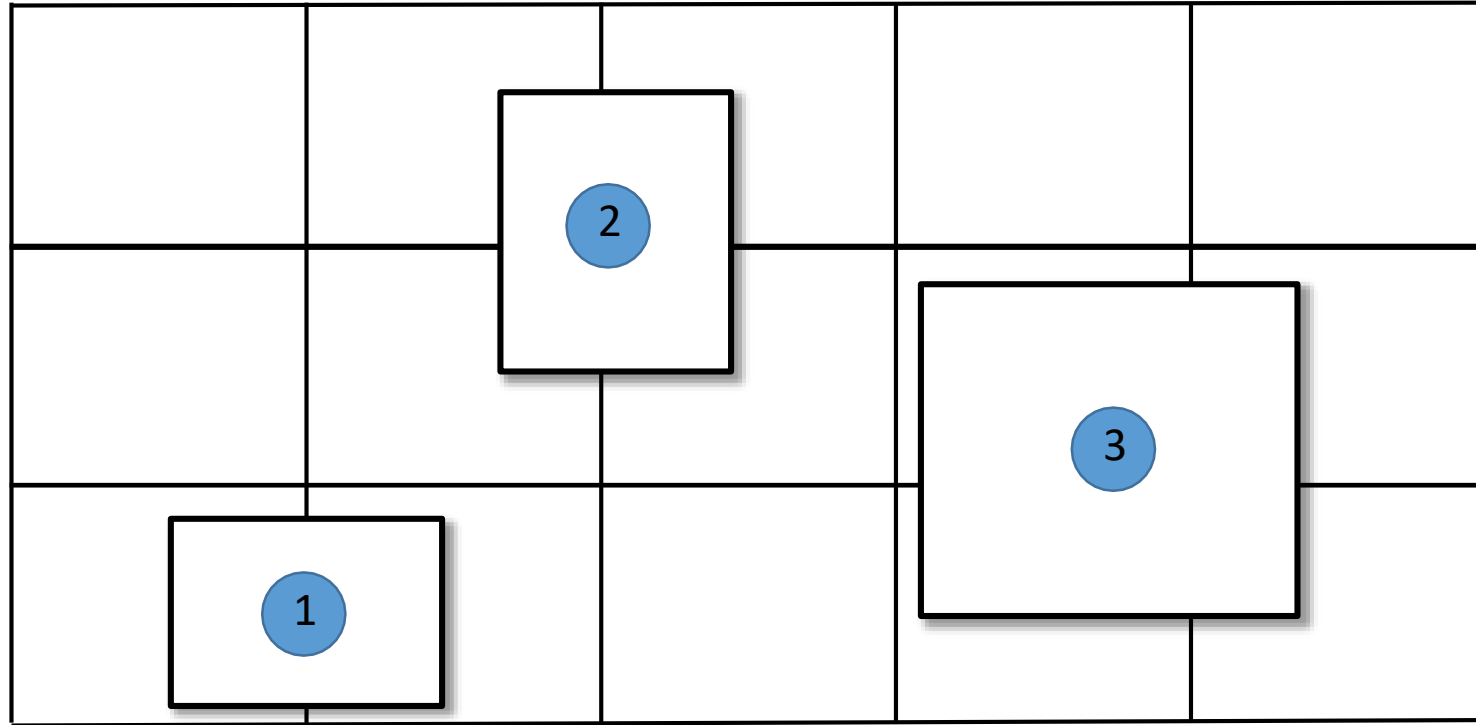
# Broad phase / Grille



# Broad phase / Grille

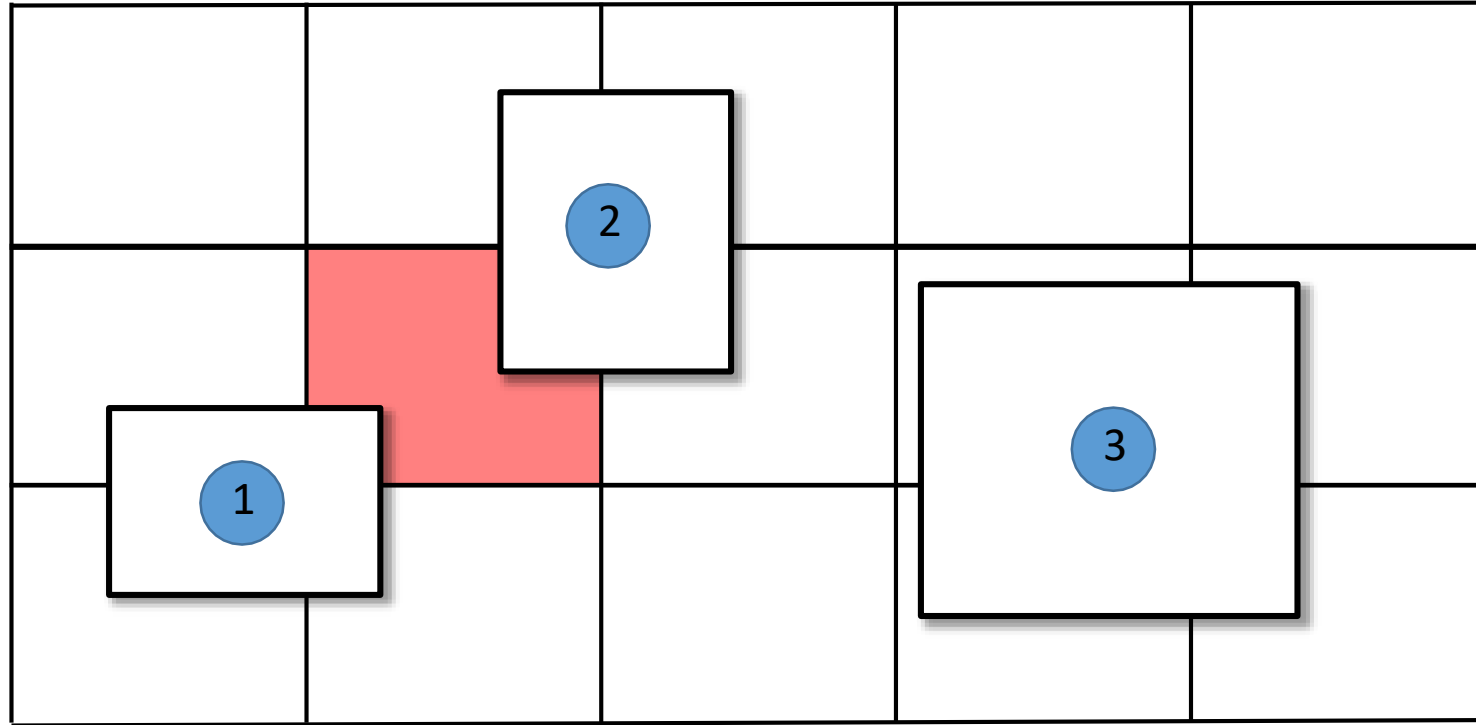
- Grille 2D ou 3D
- Chaque cellule a la liste des corps dont le volume englobant est inclus dans le volume de la cellule (partiellement ou entièrement)
- Un corps peut être dans plusieurs cellules
- On ne considère que les paires de corps qui appartiennent à la même cellule

# Broad phase / Grille



0 calcul d'intersection

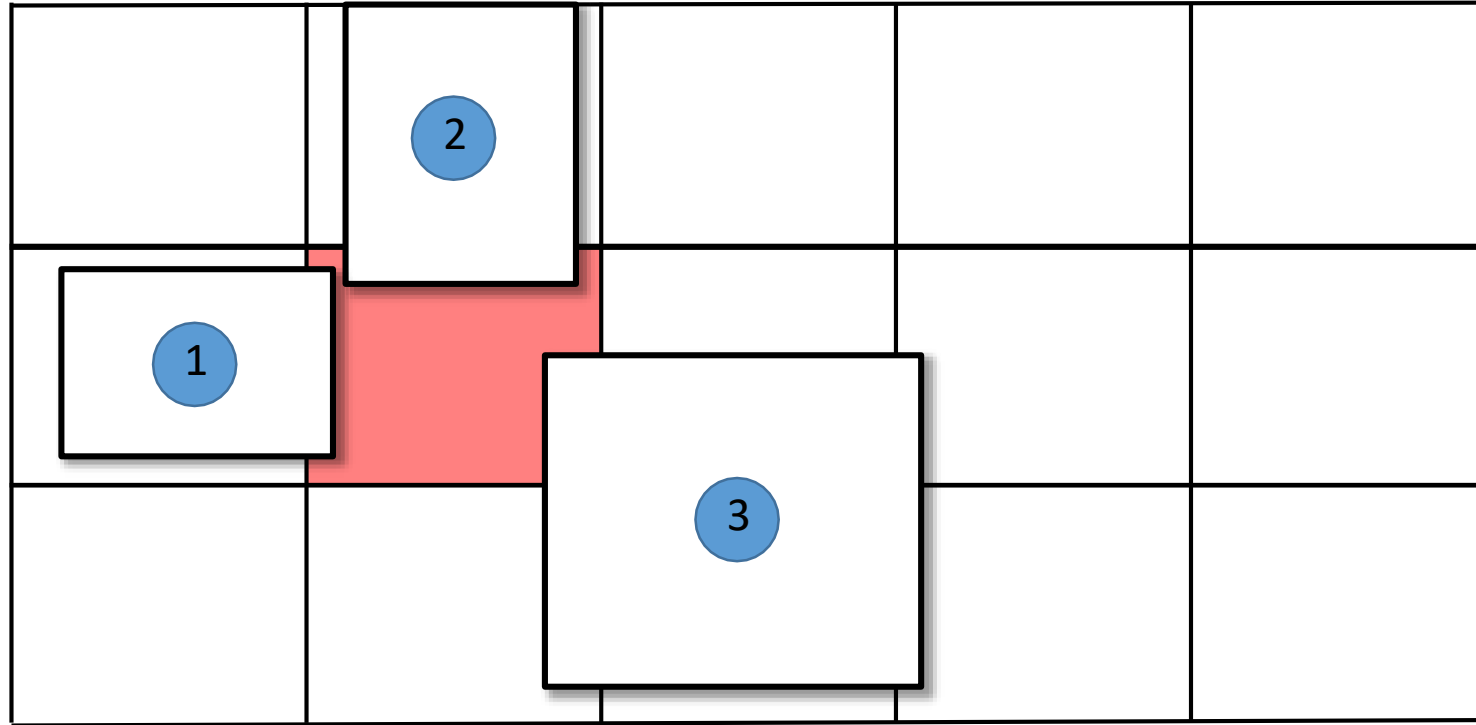
# Broad phase / Grille



1 calcul d'intersection

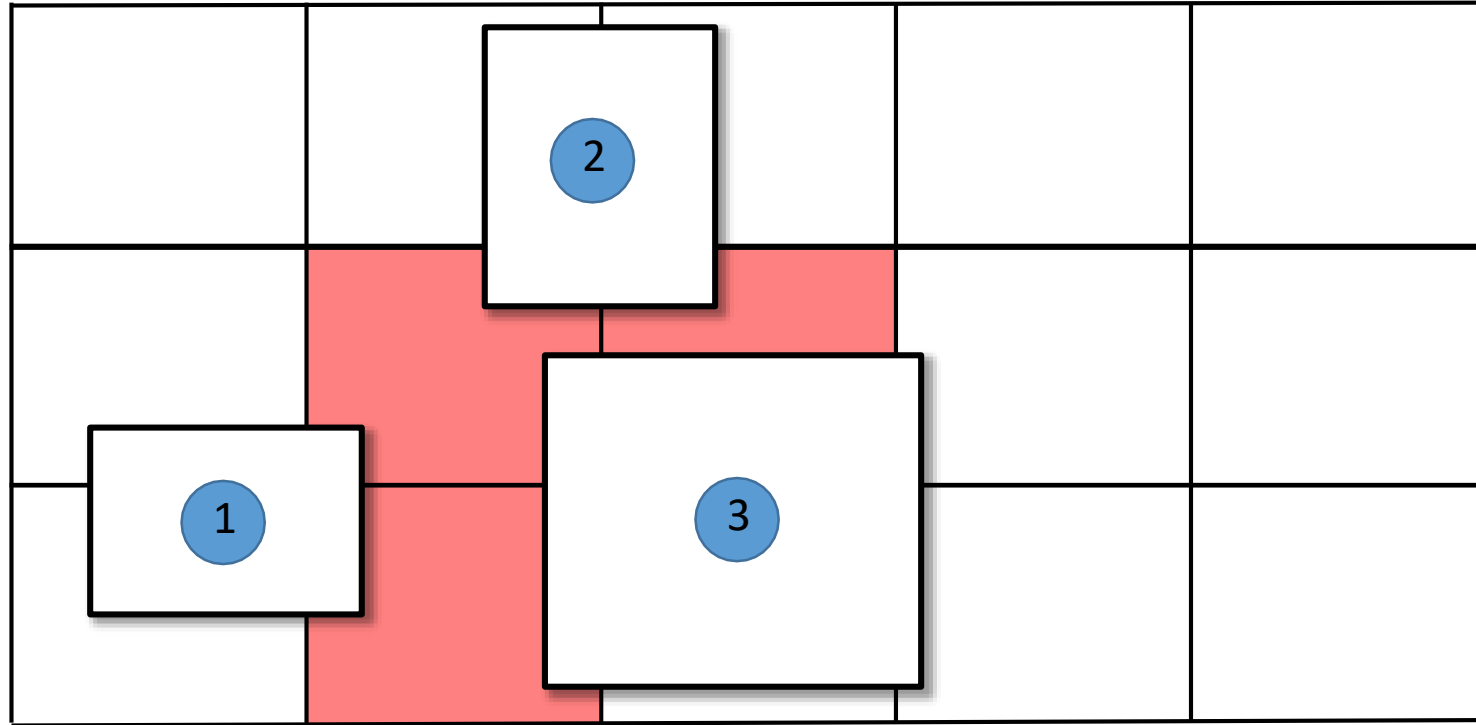


# Broad phase / Grille



3 calculs d'intersection


# Broad phase / Grille








5 calculs d'intersection

# Broad phase / Grille

## Avantages :

- Bonnes performances dans les meilleurs cas 
- Simple à implémenter 

## Inconvénients :

- Oblige à avoir une taille maximum pour le monde, ou bien à implémenter une grille dynamique 
- La taille idéale de la grille dépend beaucoup des données :
  - . Cellules trop grosses ➡ Trop d'objets par cellule ➡ Mauvaises performances 
  - . Cellules trop petites ➡ Trop de cellules ➡ Consommation mémoire   
➡ Mauvaises performances 
- Difficile de gérer des objets qui ont des tailles très différentes 

# La détection des collisions / Broad phase

Quelques algorithmes de broad phase :

1. Algorithme « Brute force »
2. Sweep & Prune
3. Les grilles
4. Les arbres

# Broad phase / Les arbres

## Idée :

- Conserver les avantages des grilles :

Bonnes performances quand il n'y a pas beaucoup d'objets par cellules et pas trop de cellules

- En éliminant les cas problématiques :

Monde limité ou grille dynamique

Cellules trop grosses

Cellules trop petites

 Utiliser des cellules de taille variable

# Broad phase / Les arbres

Quelques exemples d'arbres :

- Quadtree
- Octree
- k-d tree

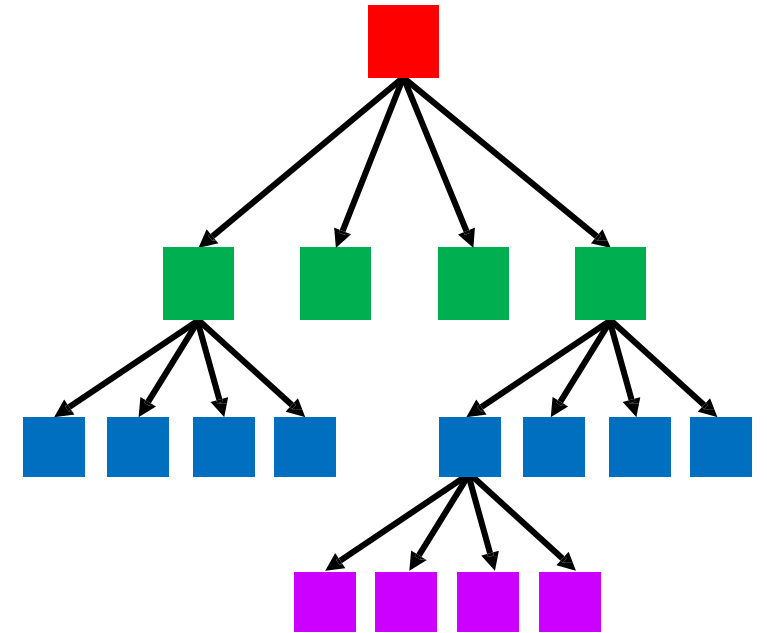
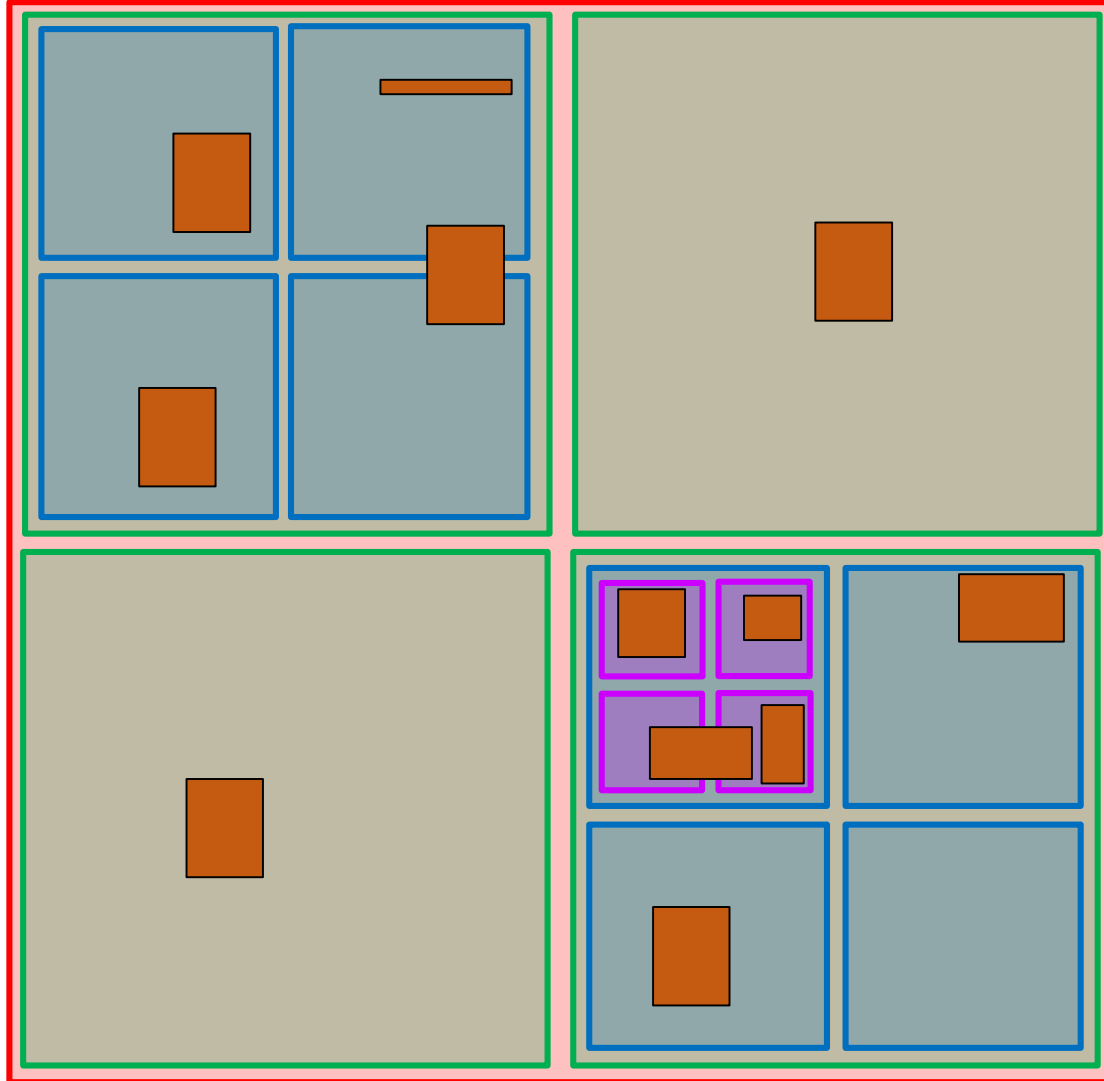
# Broad phase / Les arbres

Quelques exemples d'arbres :

- Quadtree
- Octree
- k-d tree

# Broad phase / Les arbres

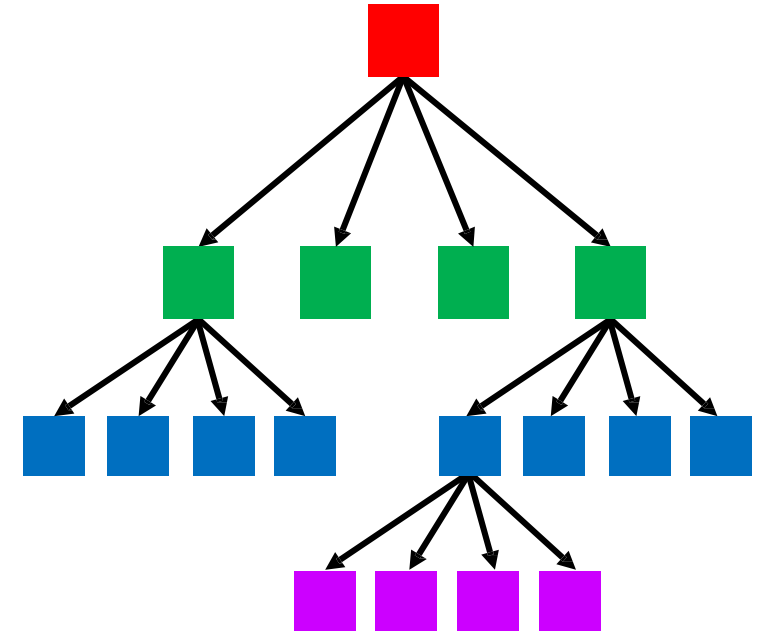
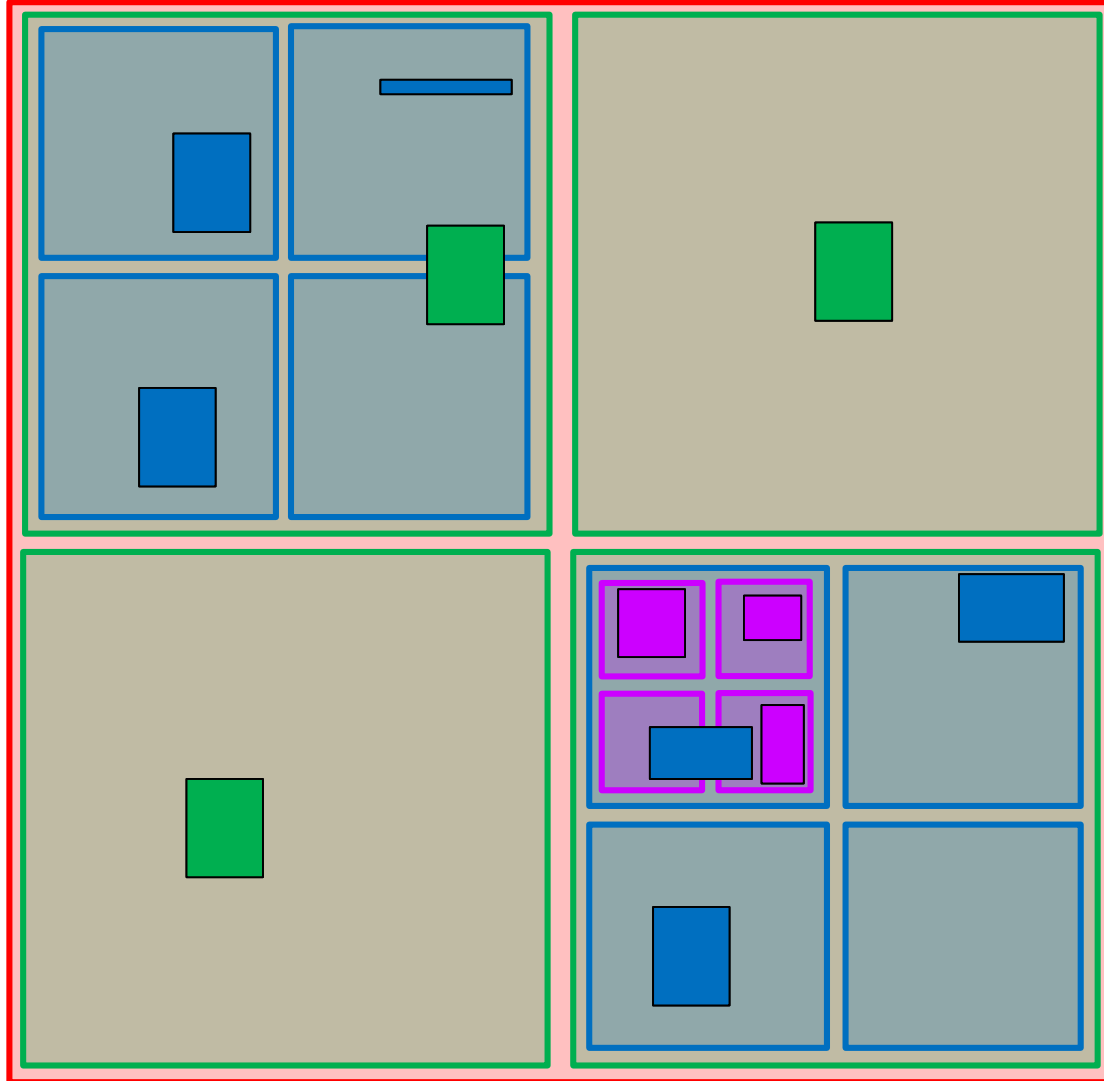
Quadtree :





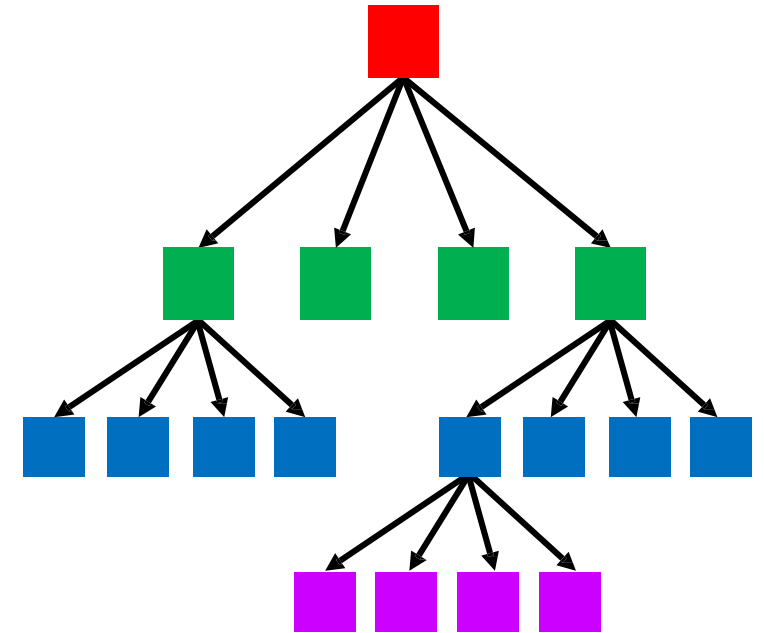
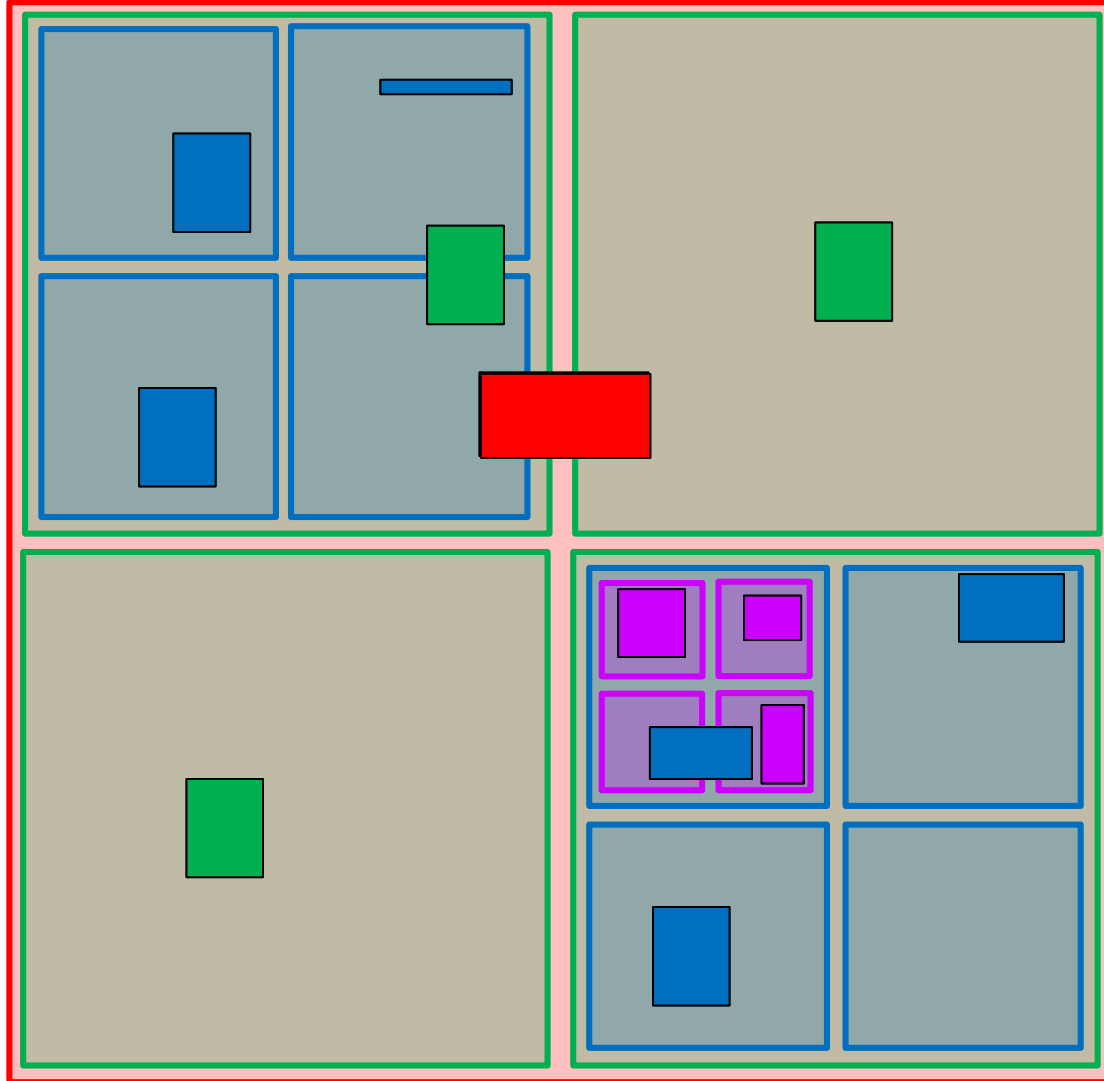
# Broad phase / Les arbres

Quadtree :






# Broad phase / Les arbres

Quadtree :



# Broad phase / Les arbres

## Quadtree – Mise à jour à chaque trame :

- Pour chaque objet, regarder s'il a changé de cellule (algorithme up-down)
- Calculer les collisions entre chaque objet et :
  - Les objets de sa cellule
  - Les objets des cellules parentes
- Si une cellule possède trop d'objets  On la subdivise
- Si des cellules feuilles de même parent ont peu d'objets  
 On les supprime  Les objets se retrouvent dans la cellule parente

# Broad phase / Les arbres

## Quadtree – Stockage en mémoire :

Pour chaque cellule :

- Liste des objets dans cette cellule
- Référence vers les cellules filles

 **Stockage très efficace en mémoire**

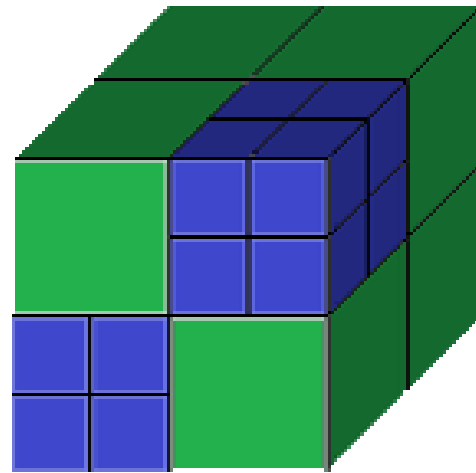
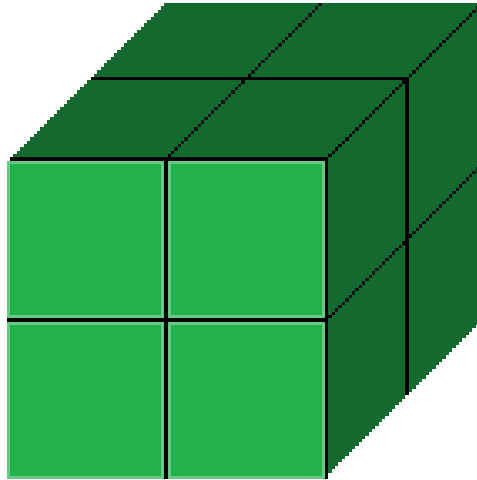
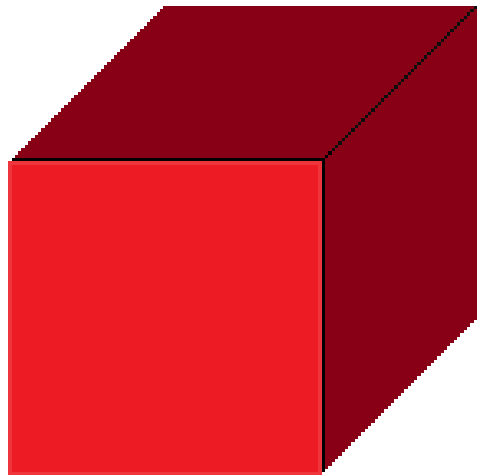
# Broad phase / Les arbres

Quelques exemples d'arbres :

- Quadtree
- Octree
- k-d tree

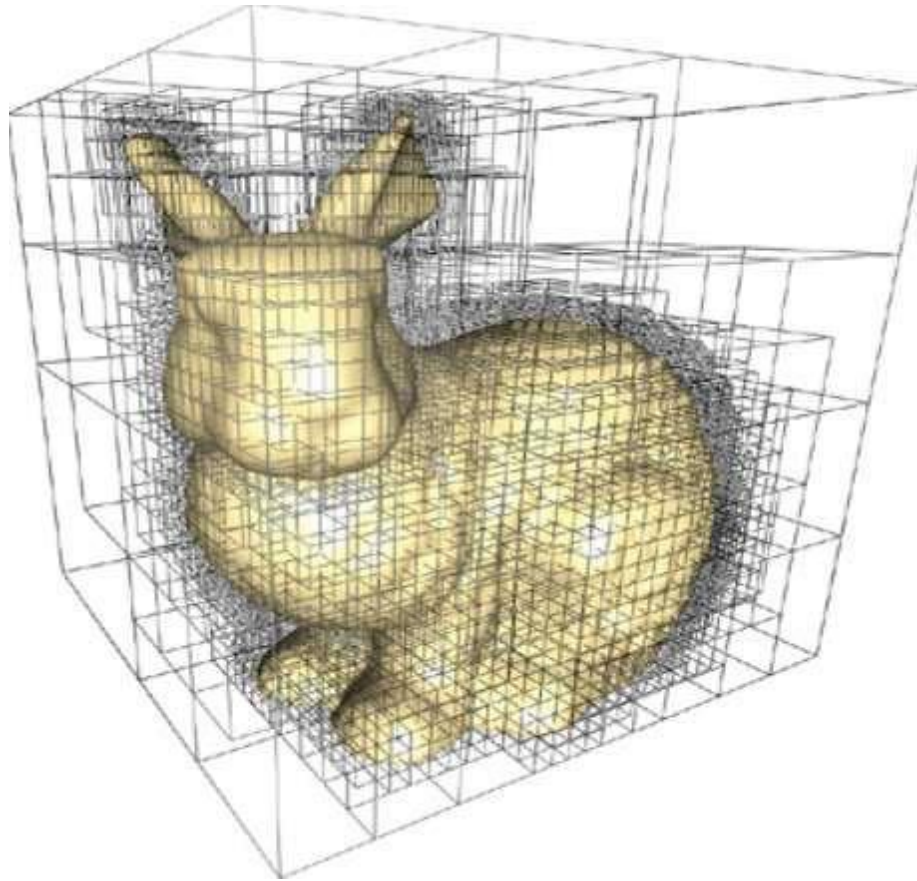
# Broad phase / Les arbres

Octree :



# Broad phase / Les arbres

Octree :



# Broad phase / Les arbres

Idée :

- Conserver les avantages des grilles :

Bonnes performances quand il n'y a pas beaucoup d'objets par cellules et pas trop de cellules



- En éliminant les cas problématiques :

Monde limité



~~Cellules trop grosses~~

~~Cellules trop petites~~





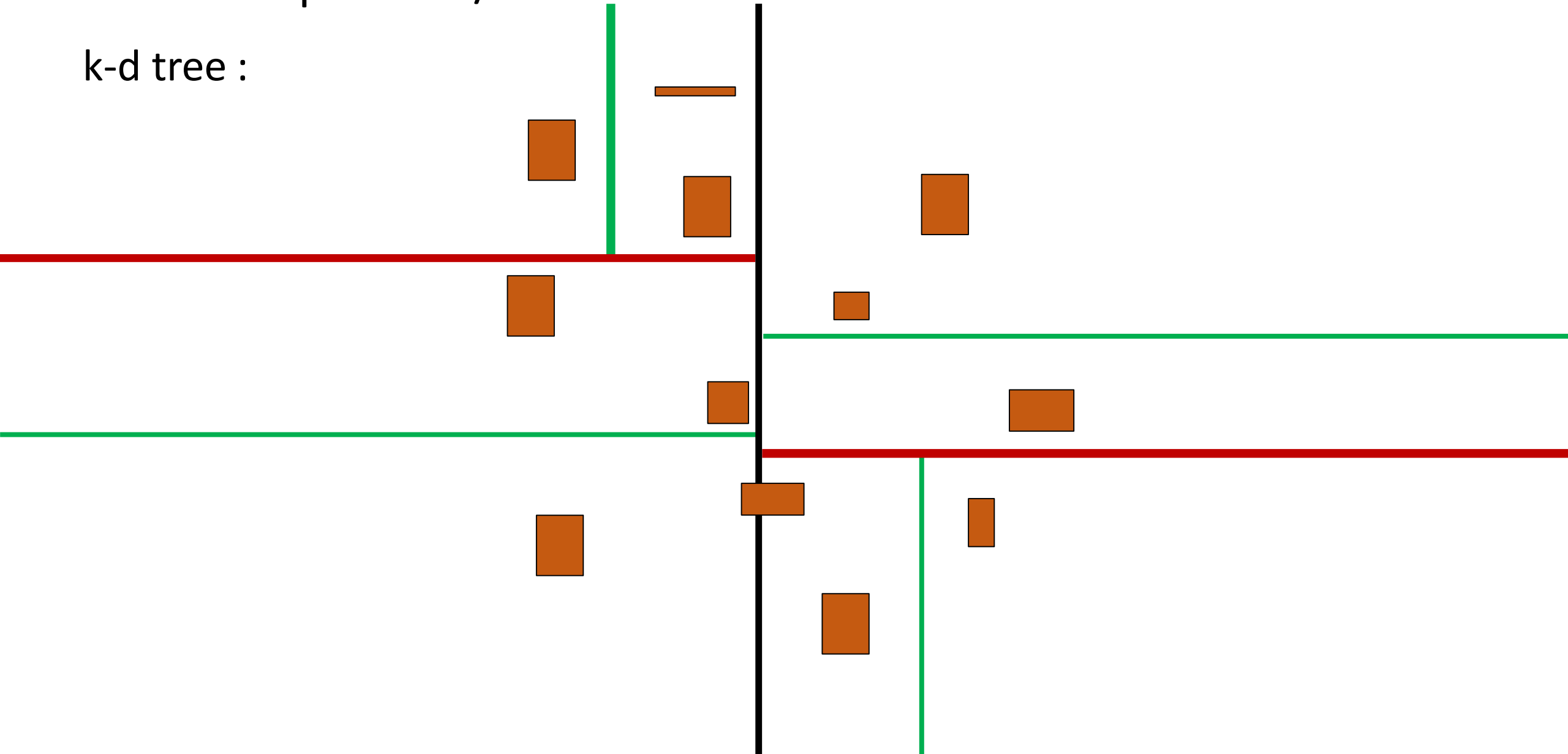
# Broad phase / Les arbres

Quelques exemples d'arbres :

- Quadtree
- Octree
- k-d tree

# Broad phase / Les arbres

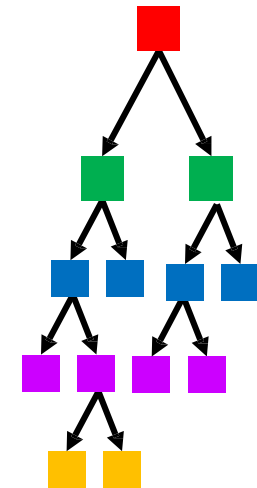
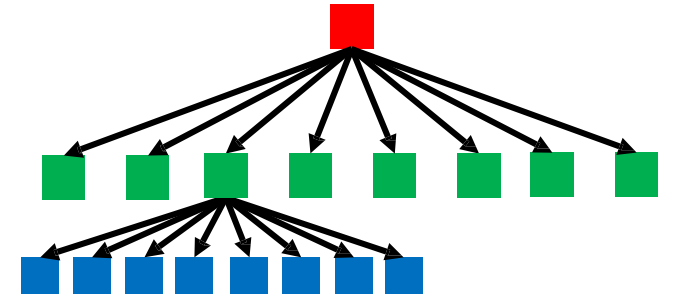
k-d tree :



# Broad phase / Les arbres

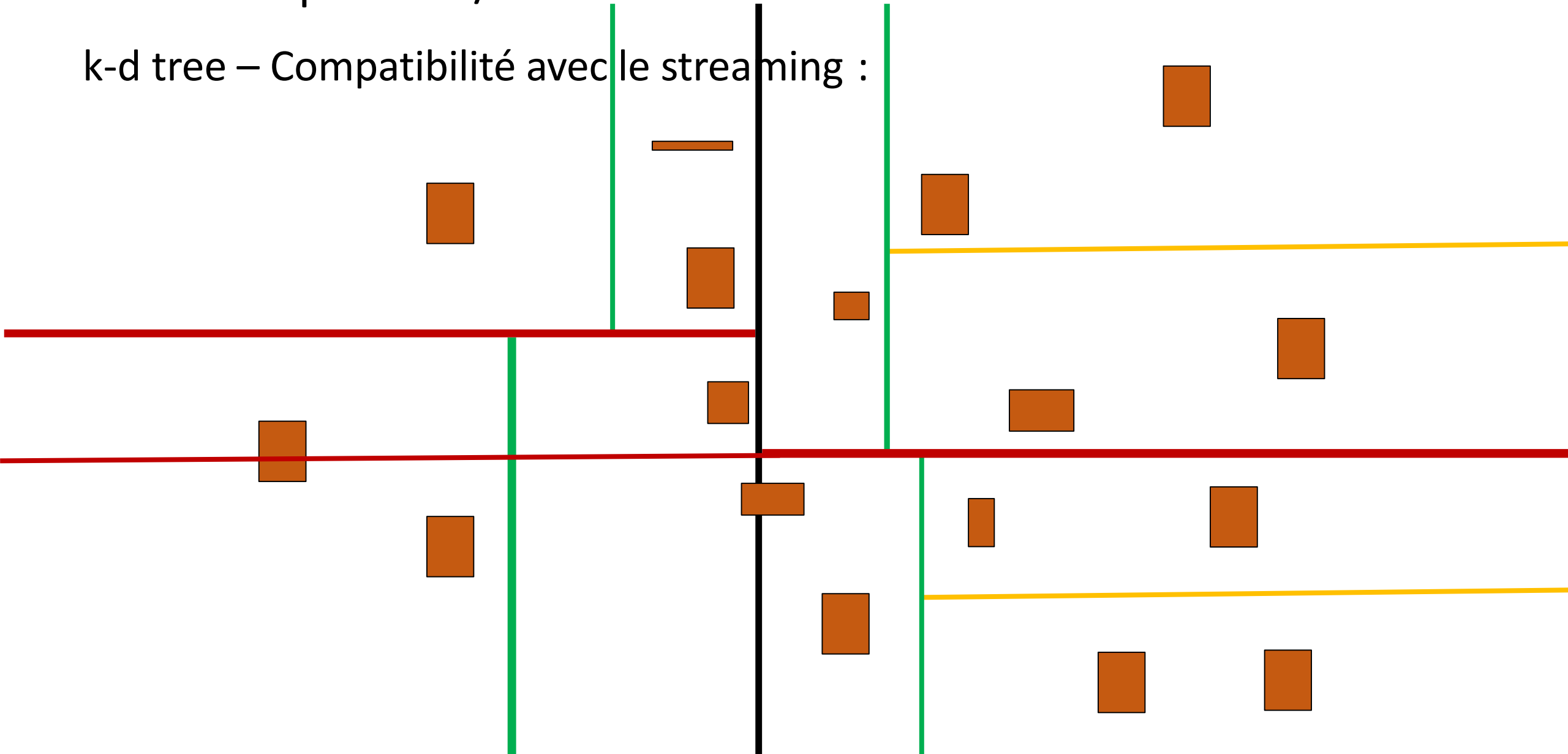
K-d tree – Différences par rapport à un octree :

- On stocke les coordonnées du plan de séparation
- Chaque nœud a 2 fils



# Broad phase / Les arbres

k-d tree – Compatibilité avec le streaming :



# Broad phase / Les arbres

Idée :

- Conserver les avantages des grilles :

Bonnes performances quand il n'y a pas beaucoup d'objets par cellules et pas trop de cellules



- En éliminant les cas problématiques :

~~Monde limité~~



~~Cellules trop grosses~~

~~Cellules trop petites~~



# Broad phase / Filtrage

Le filtrage des collisions :

Qu'est-ce que c'est ?

- Certaines paires de corps en collision sont ignorées

Pourquoi ?

- Pour des raisons liées au gameplay
- Pour des questions de performance

# Broad phase / Filtrage

Le filtrage des collisions :

Comment ?

- Flags (champs de bits)
- Catégories
- Liste d'exclusion
- Callback

# La détection des collisions / Narrow phase

Mission de la narrow phase :

Pour chacune des paires potentielles trouvées par la broad phase :

- Déterminer si les deux corps sont **réellement** en collision.
- Si oui, calculer :
  - . Les points de contact
  - . La normale
  - . La distance de pénétration

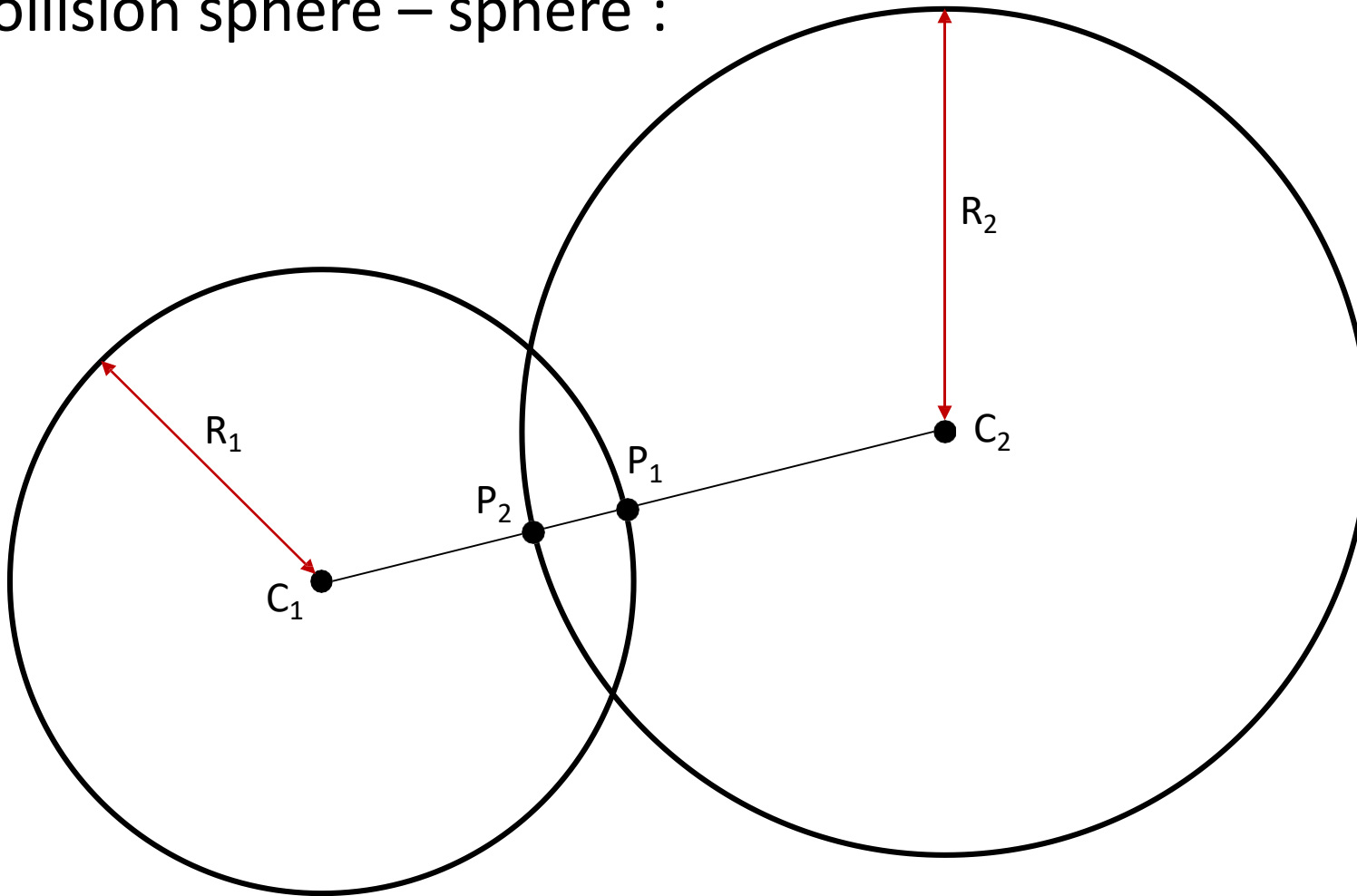
**Collisions :**

<http://www.realtimerendering.com/intersections.html>



# La détection des collisions / Narrow phase

Collision sphère – sphère :



Collision si  $C_1C_2^2 \leq (R_1+R_2)^2$

$$P_1 = C_1 + R_1 \cdot \frac{C_1C_2}{C_1C_2}$$

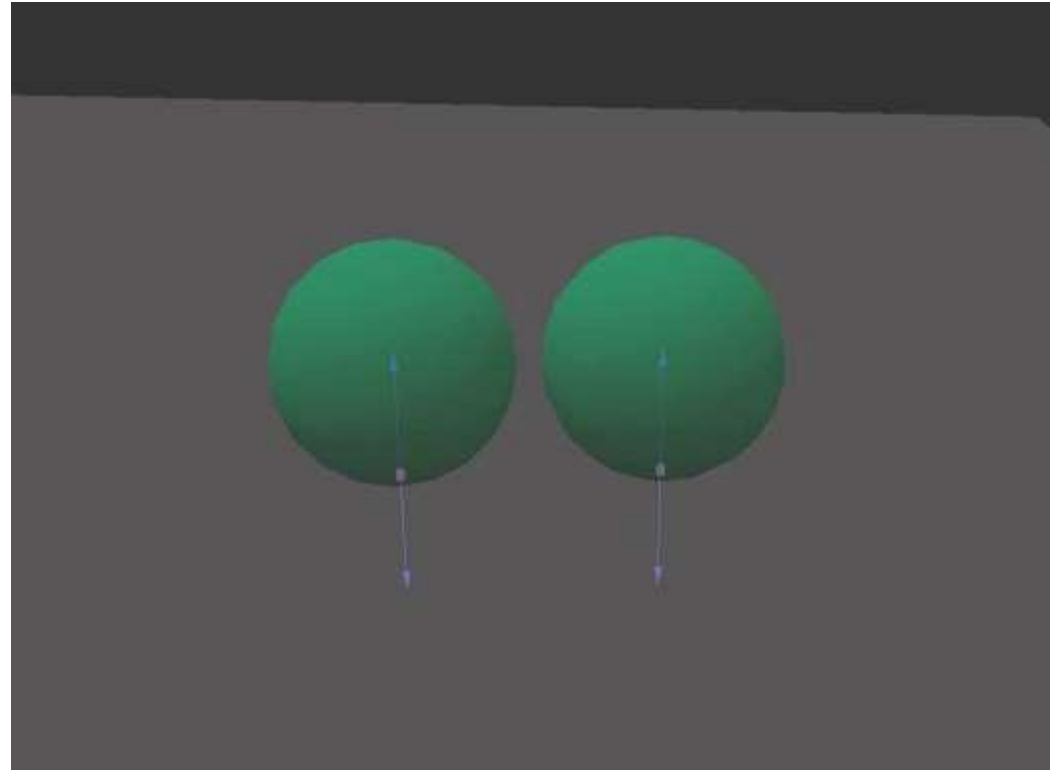
$$P_2 = C_2 + R_2 \cdot \frac{C_2C_1}{C_2C_1}$$

$$n = \frac{P_1P_2}{P_1P_2}$$

$$d = -P_1P_2$$

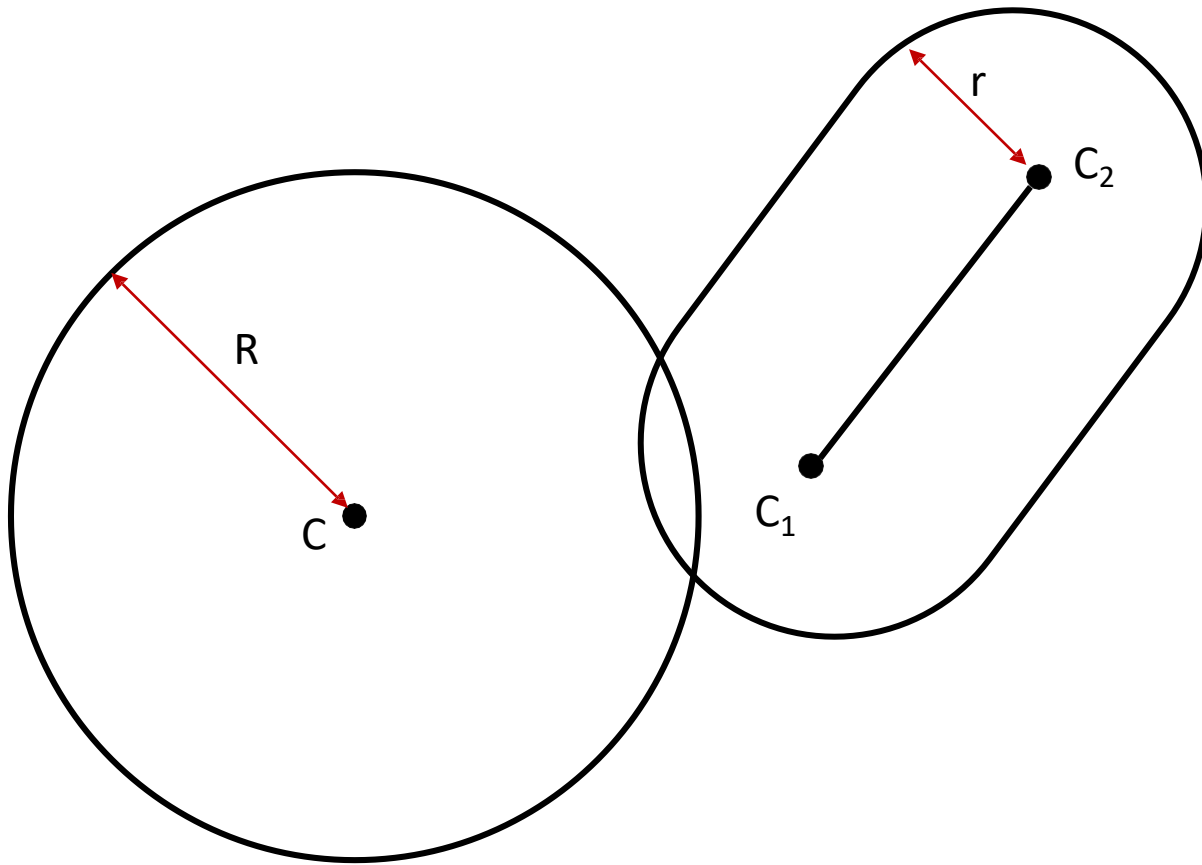
# La détection des collisions / Narrow phase

Collision sphère – sphère :



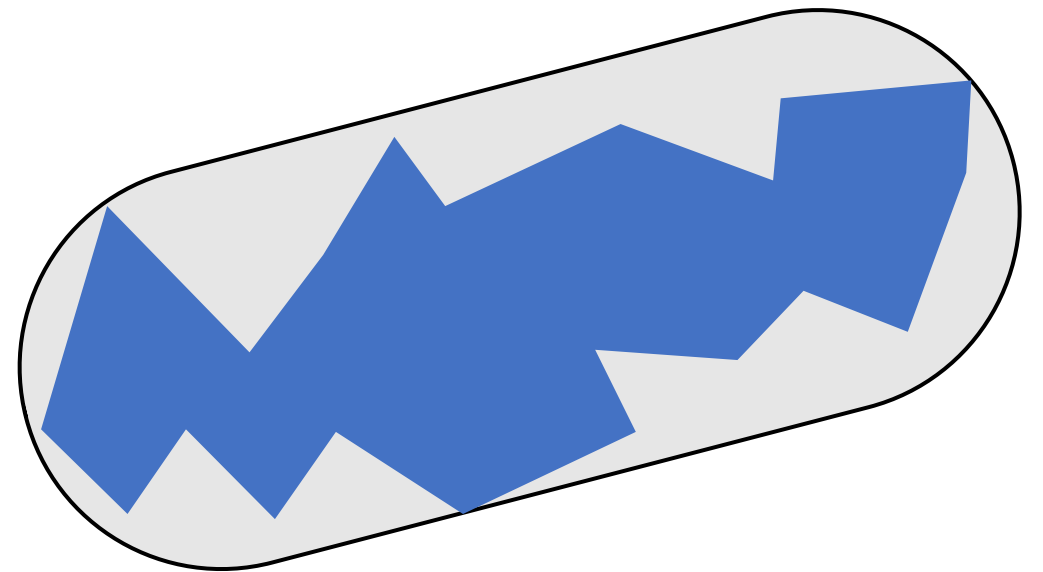
# Narrow phase / capsule

Collision sphere – capsule :



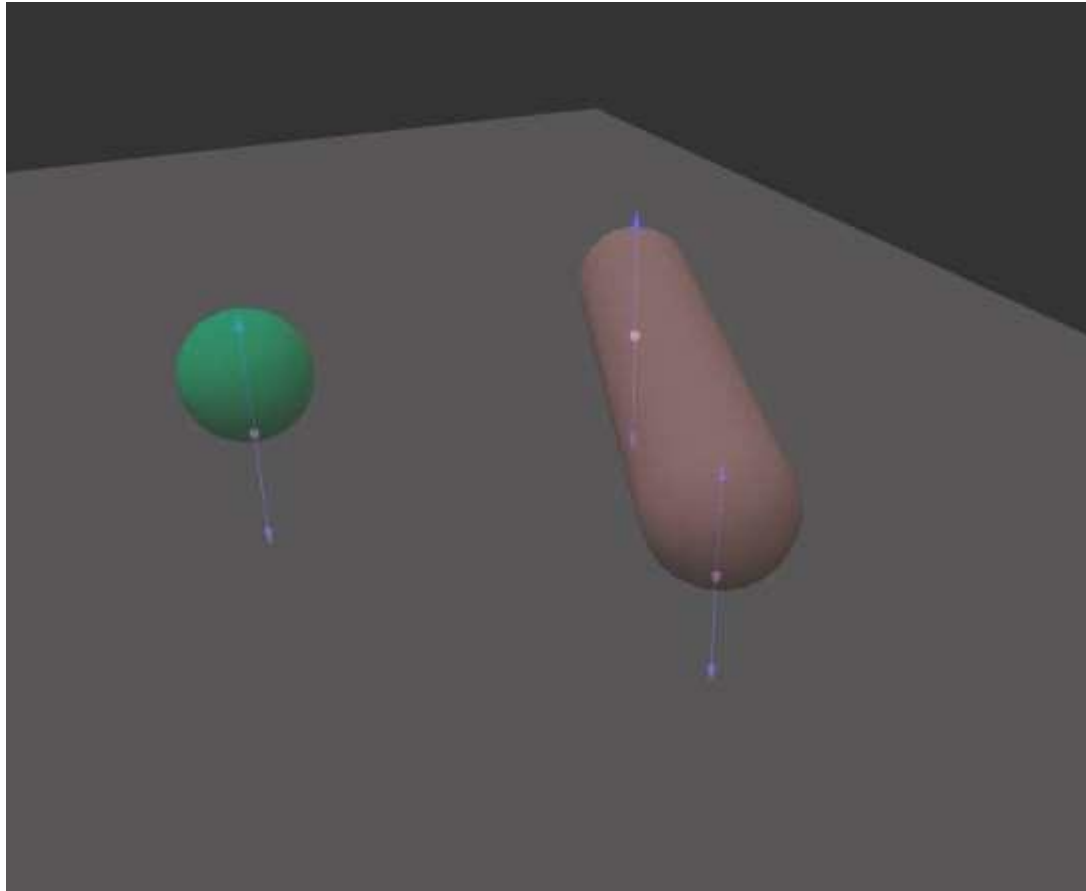
Collision si

$$d(C, C_1C_2)^2 \leq (R+r)^2$$



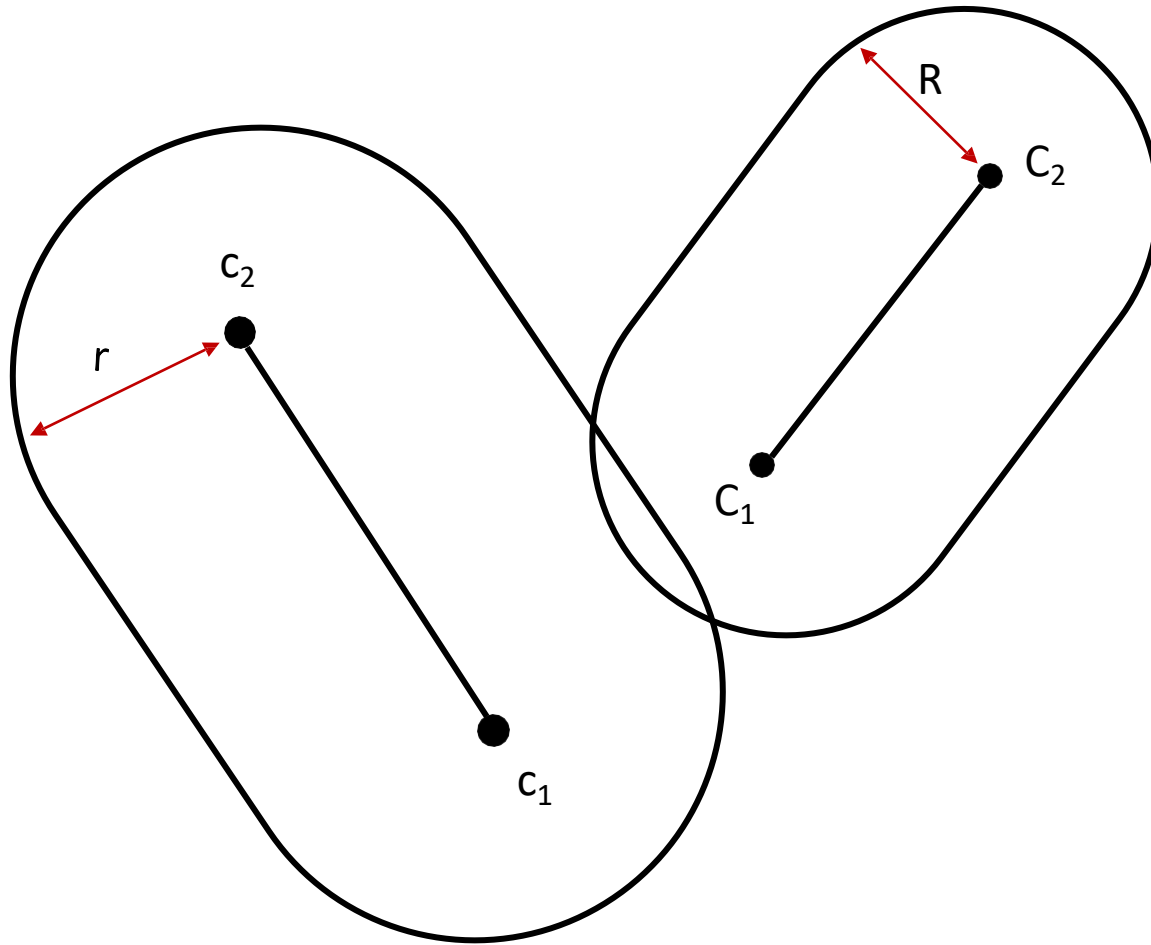
# Narrow phase / capsule

Collision sphere – capsule :



# Narrow phase / capsules

Collision capsule – capsule :

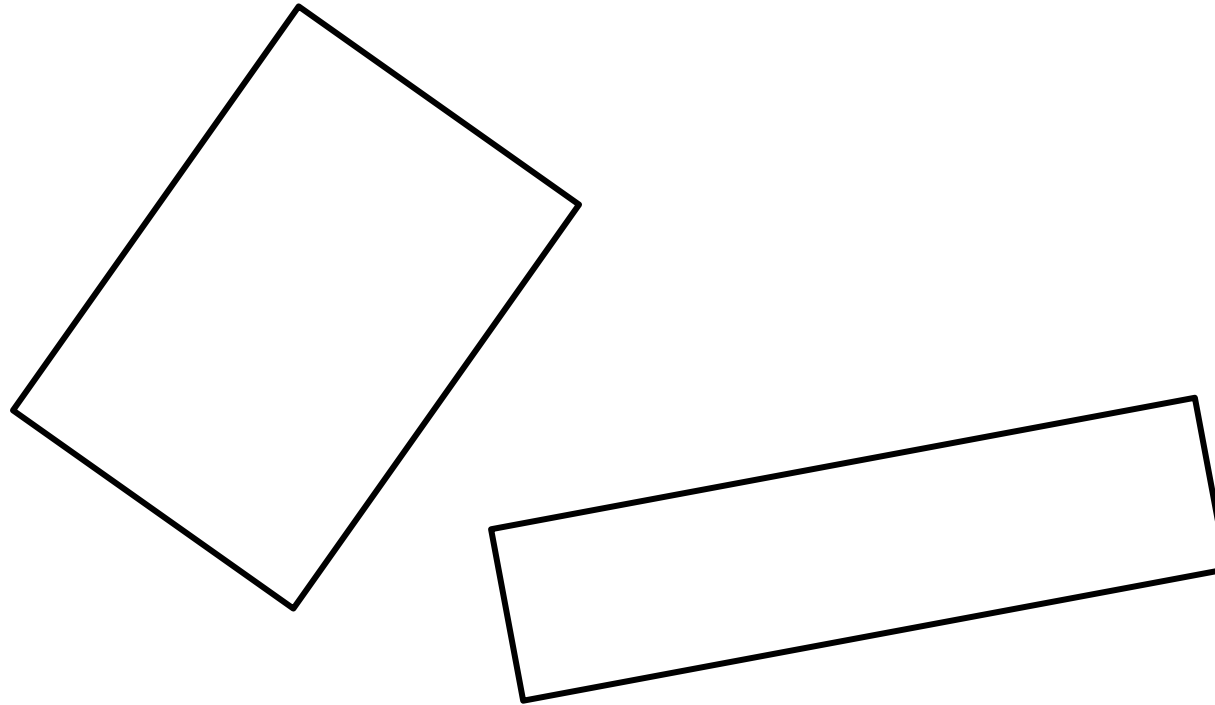


Collision si

$$d(c_1c_2, C_1C_2)^2 \leq (R+r)^2$$

# Narrow phase / boîte

Collision boîte – boîte :



# Narrow phase / boîte

Collision boîte – boîte :

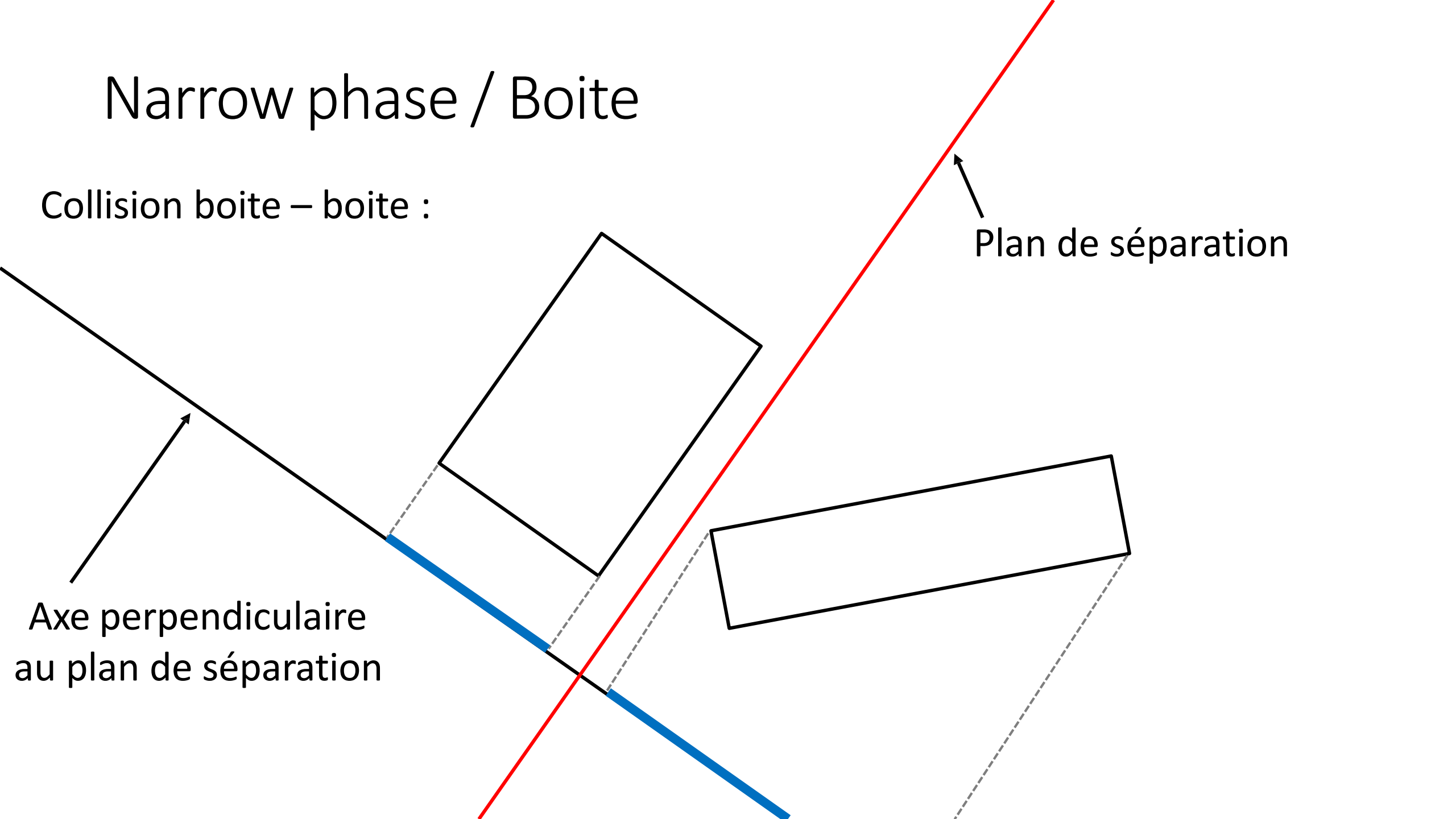


Théorème de séparation :

Si deux boîtes ne sont pas en intersection, il est possible de trouver un plan **qui sépare l'espace en deux demi-espaces** de sorte que chacun contienne entièrement l'une des boîtes

# Narrow phase / Boite

Collision boite – boite :



Plan de séparation

Axe perpendiculaire  
au plan de séparation



# Narrow phase / Boite



Si les deux boites ne sont pas en collision :

Comment choisir le plan de séparation ?

Si les deux boites sont en collision :

S'il faut tester une infinité de plans,  
l'algorithme n'est pas utilisable en pratique

# Narrow phase / Boite

## Théorème de séparation :

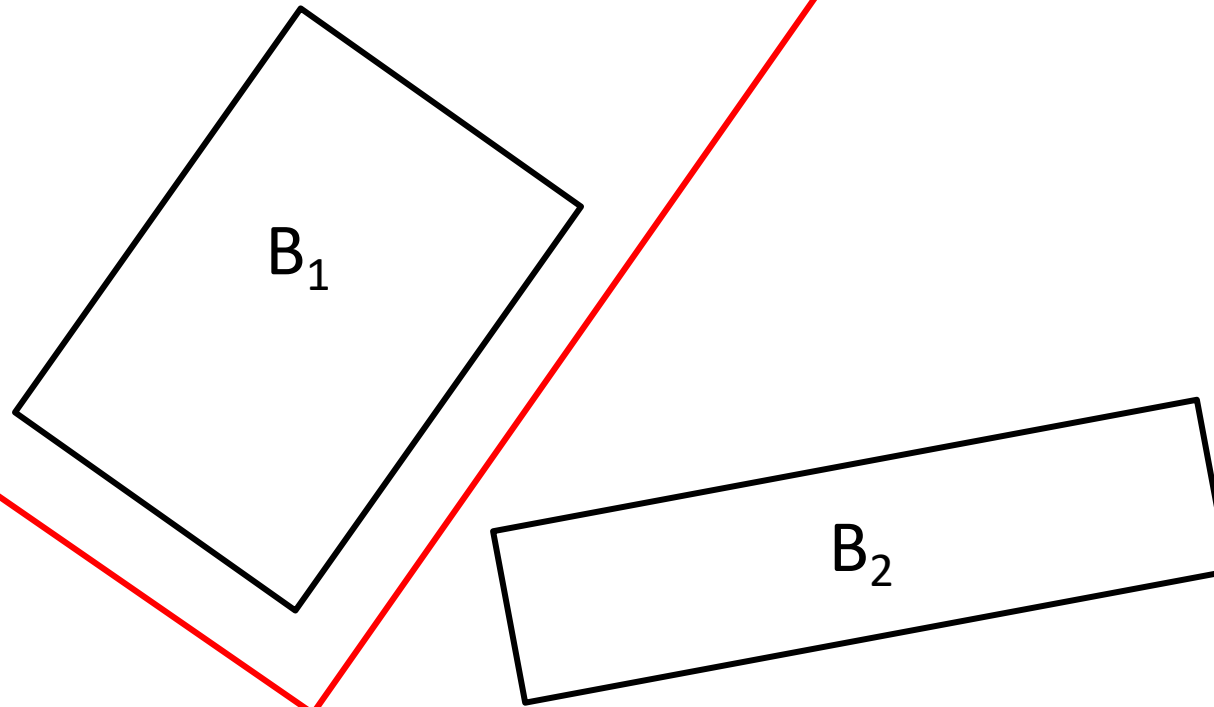
Si deux boites ne sont pas en intersection, il est possible de trouver un plan qui sépare l'espace en deux demi-espaces de sorte que chacun contienne entièrement l'une des boites

## Théorème :

Il existe un ensemble fini de plans qu'il est suffisant de tester pour savoir si deux boites sont en intersection

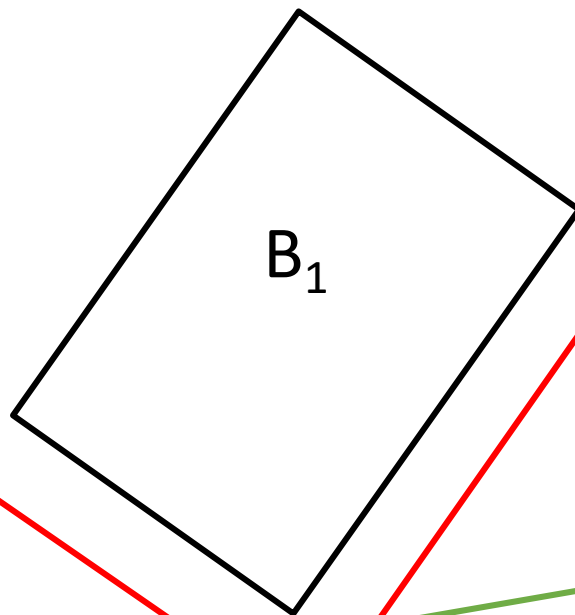
# Narrow phase / Boite

Les 3 plans parallèles aux 3 faces de la boîte  $B_1$

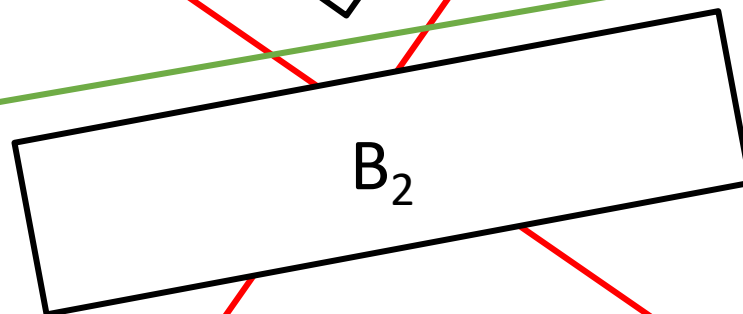


# Narrow phase / Boite

Les 3 plans parallèles aux 3 faces de la boîte  $B_1$



Les 3 plans parallèles  
aux 3 faces de la boîte  $B_2$



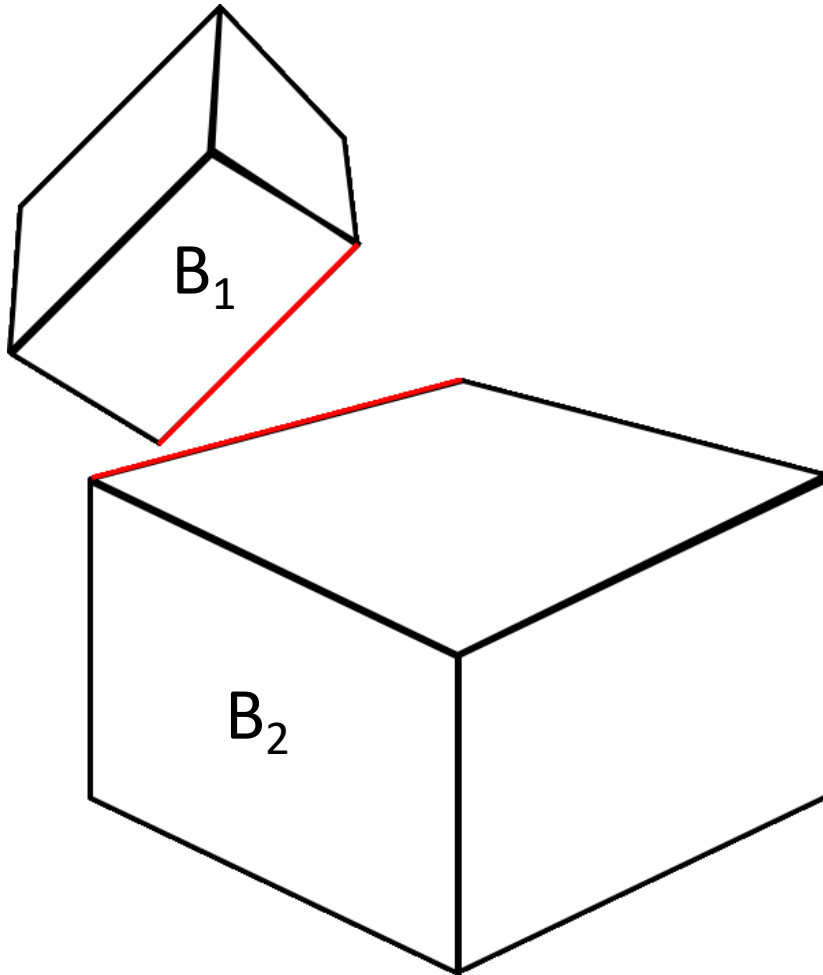
# Narrow phase / Boite

En 2D : 4 droites sont suffisantes

En 3D : Est-ce que 6 plans sont suffisants ?



# Narrow phase / Boite



Il est nécessaire de considérer également les plans dont la normale est le produit vectoriel d'une arête de B1 et d'une arête de B2

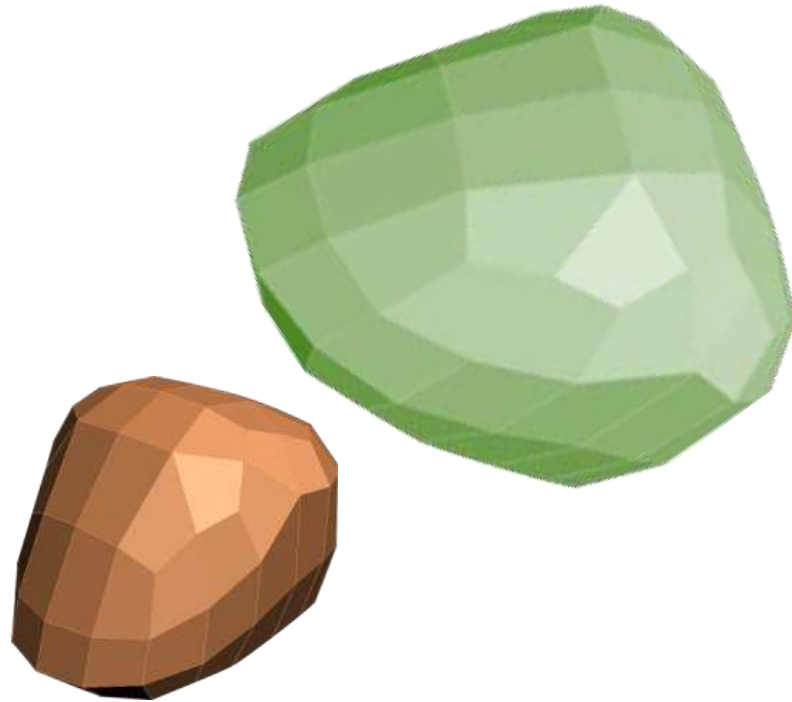
Nombres de plans =  $3 + 3 + 3 \times 3 = 15$

Théorème :

Il est nécessaire et suffisant de tester 15 plans pour déterminer si deux boites sont en intersection

# Narrow phase / Convexe

Collision convexe – convexe :



# Narrow phase / Convexe

Collision convexe – convexe :

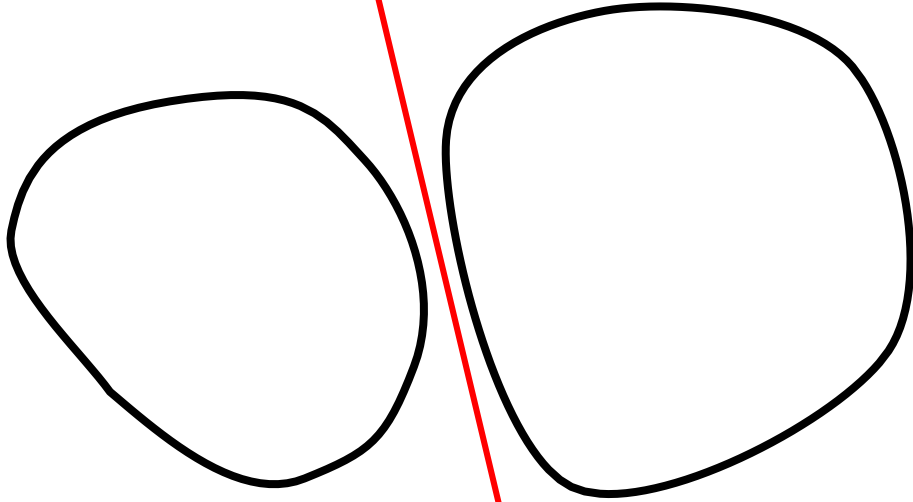
Théorème de séparation :

convexes

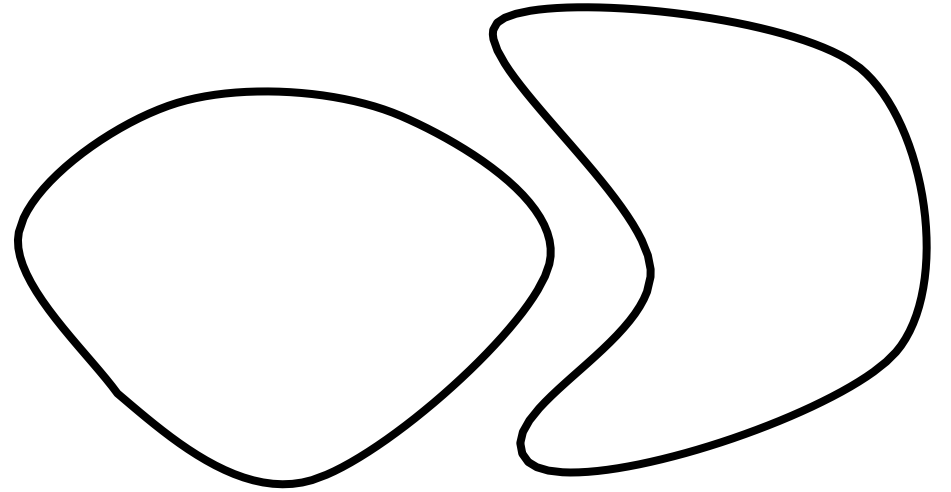
Si deux ~~boites~~ ne sont pas en intersection, il est possible de trouver un plan qui sépare l'espace en deux demi-espaces de sorte que chacun contienne entièrement l'une ~~des~~ ~~boites~~ convexes



Narrow phase / Convexe



?



# Narrow phase / Convexe

Collision convexe – convexe :



Convexe  $C_1$  :  $f_1$  faces et  $a_1$  arrêtes



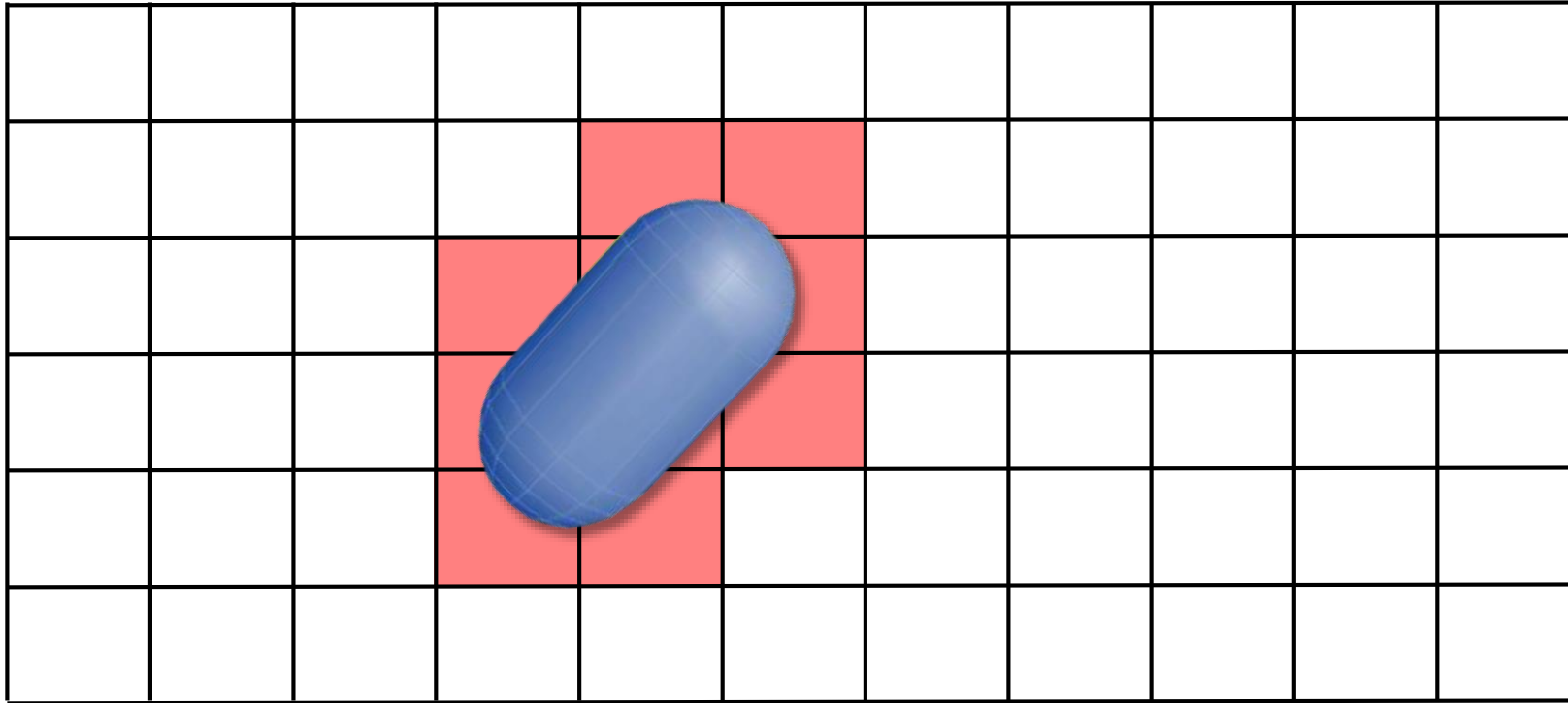
Convexe  $C_2$  :  $f_2$  faces et  $a_2$  arrêtes

Nombre de plans à tester :  $f_1 + f_2 + a_1 \times a_2$

$\underbrace{\hspace{1.5cm}}_{O(n)} \quad \underbrace{\hspace{1.5cm}}_{O(n^2)}$

# Narrow phase / Heightfield

Collision avec un heightfield :

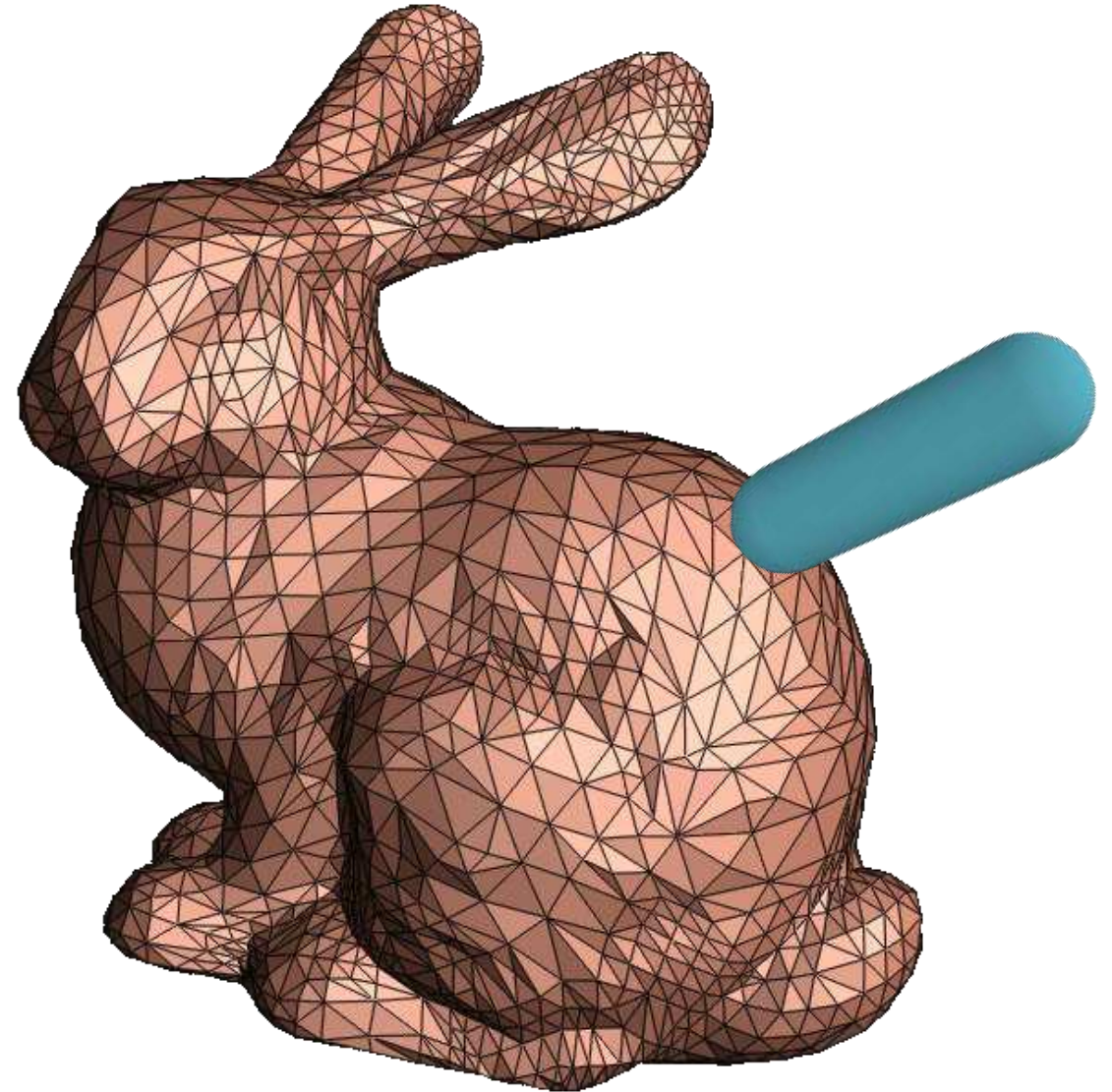


# Narrow phase / Mesh

Collision avec un mesh :

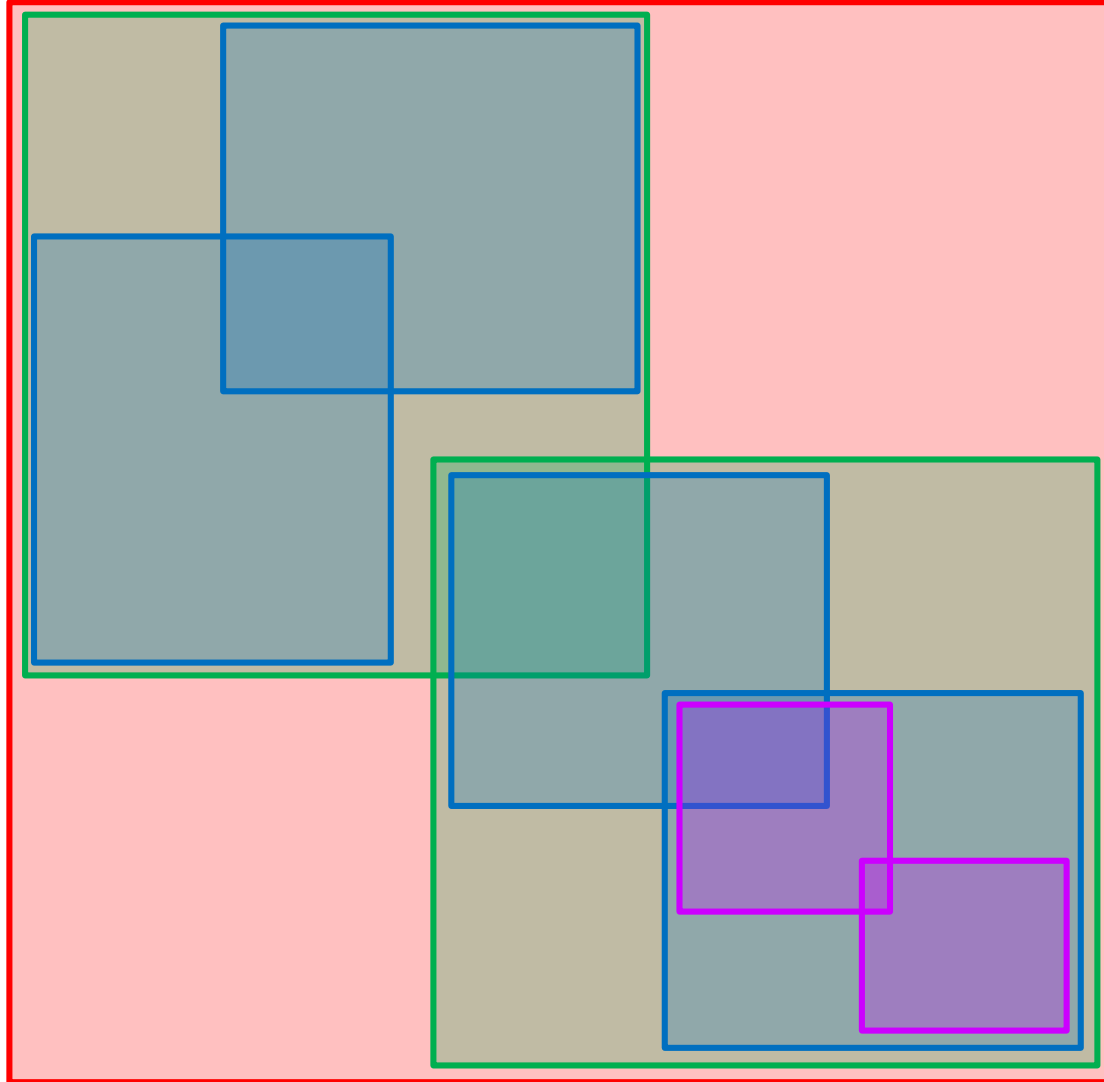
Comment déterminer les triangles du mesh en collision avec la capsule sans tester tous les triangles ?

➡ On utilise une structure d'accélération : grille ou arbre



# Narrow phase / Mesh

## AABB tree :

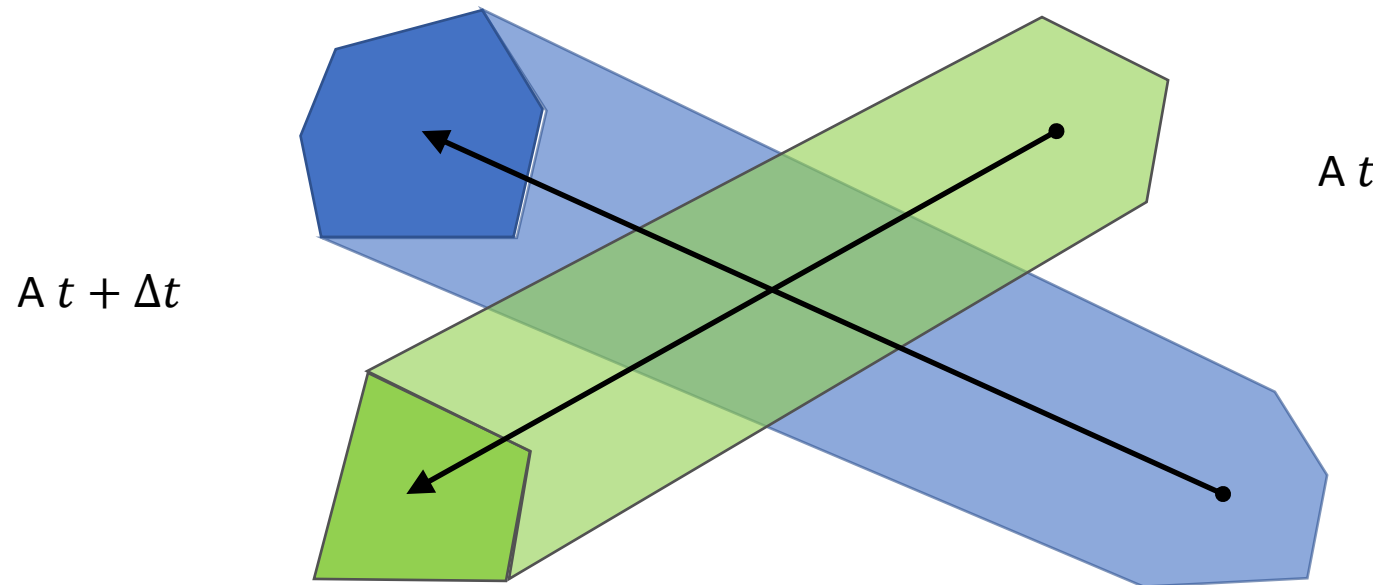


## Avantages :

- Meilleurs volumes englobants
- Tous les triangles sont dans les nœuds feuilles

# Le problème du temps

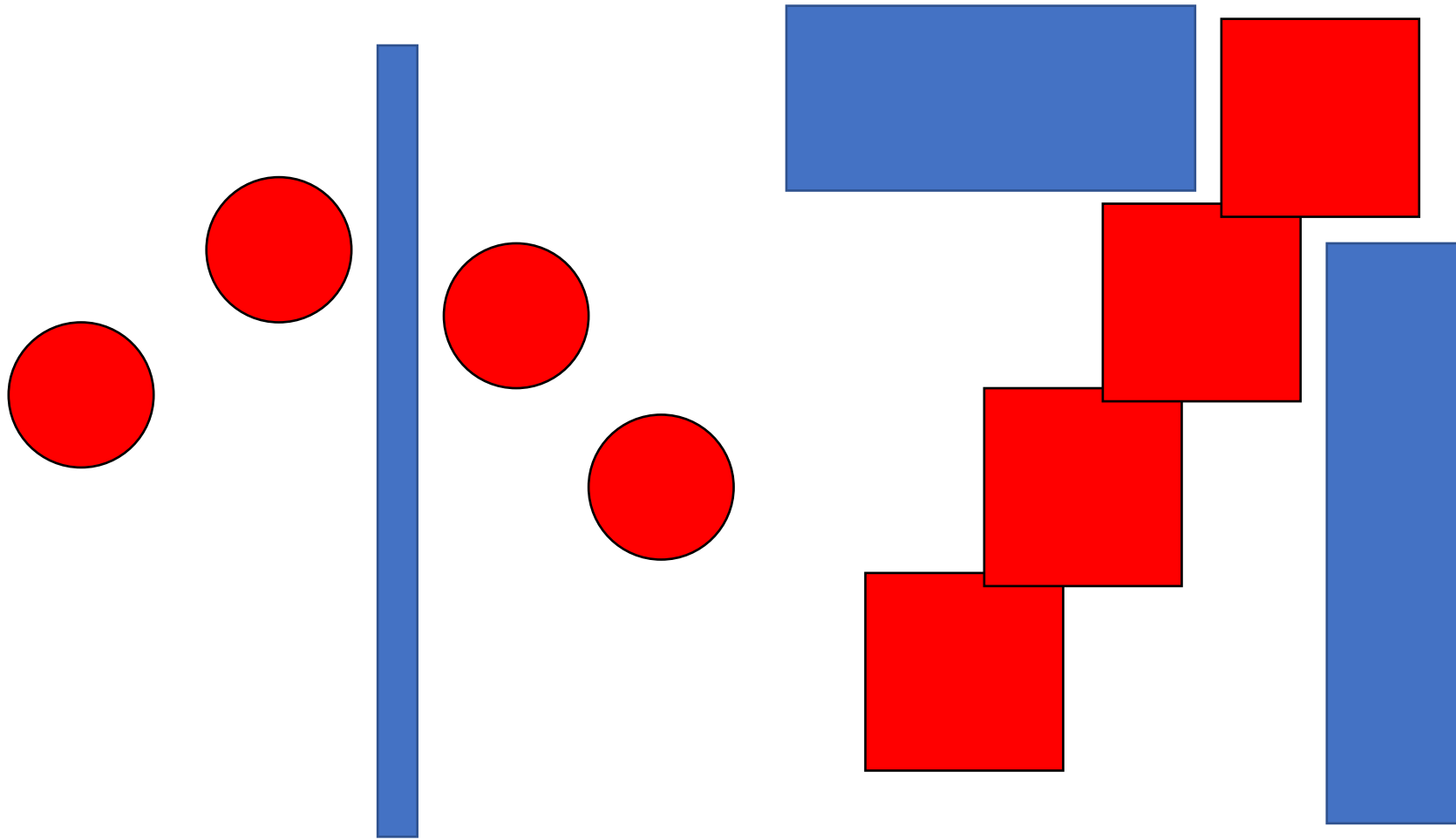
- Regarder des sequences de positions décorrélées n'est pas suffisant
- Les objets sont en mouvement et nous devons trouver le moment et point d'intersection
  - Pour réagir à la collisions *e.g.* rebondir



# Tunneling

- Collision entre deux pas peut créer un effet de **tunneling**
  - Les objets passent au travers
    - Pas de collision en  $t$  ni en  $t + \Delta t$
    - Mais entre les 2
  - faux positifs
- Tunneling est un problème sérieux pour le gameplay
  - Joueurs arrivant à des endroits imprévus
  - Projectiles passant à travers les personnages et les murs
  - Impossibilité pour le joueur de déclencher des actions à l'évènement de contact

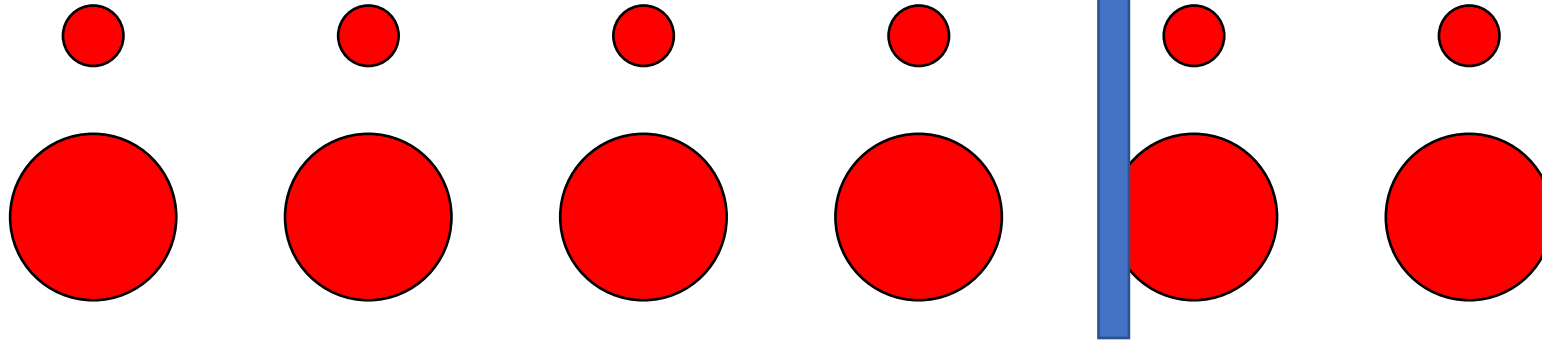
# Tunneling



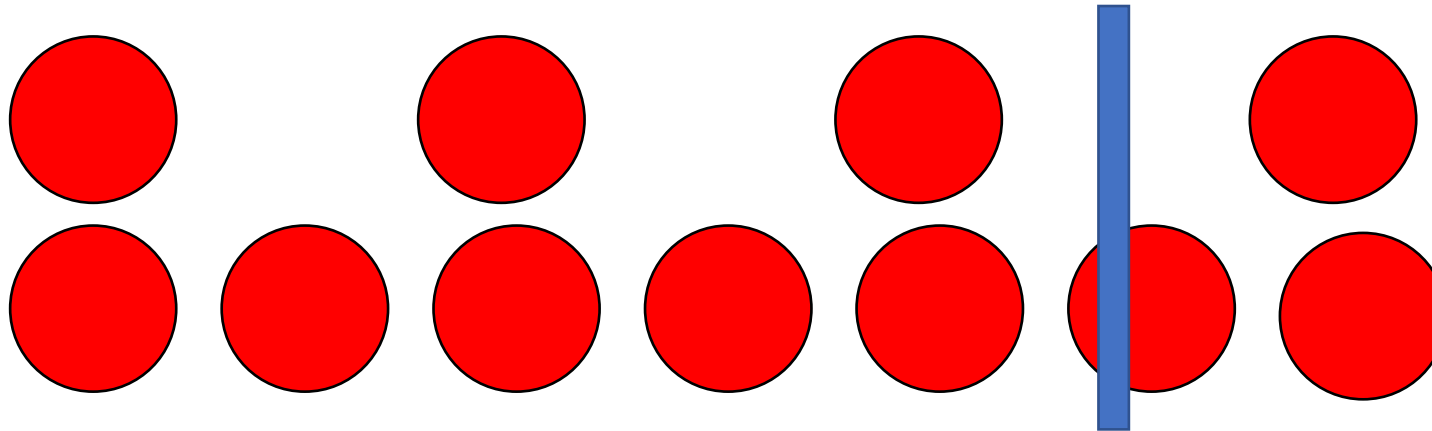


# Tunneling

- Plus frequents pour les petits objets



- Plus frequent pour les objets rapides



# Tunneling

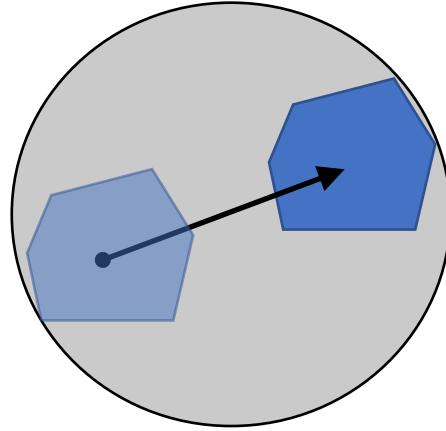
- Solutions possibles
  - Taille minimum d'objet ?
    - Les objet rapides *tunnel* toujours
  - Vitesse maximum?
    - Objets petits et rapides non autorisés (*e.g.* balles...)
  - Petit pas de temps ?
    - Meme problème que la limite de Vitesse
- Nous avons besoin d'une autre approche

# Borner le mouvement

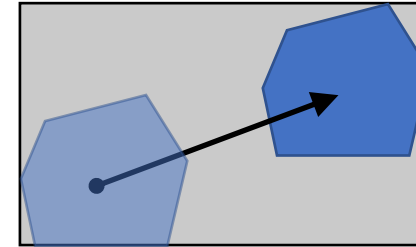
- Bornes incluant le mouvement de la forme
  - Dans le  $\Delta t$  minimal, le mouvement linéaire de la forme est inclus
  - Des bornes convexes sont utilisées, alors les mouvements sont des primitives de formes

# Borner le mouvement

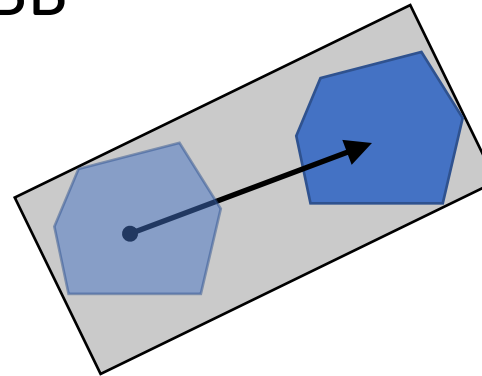
- Sphere



- AABB

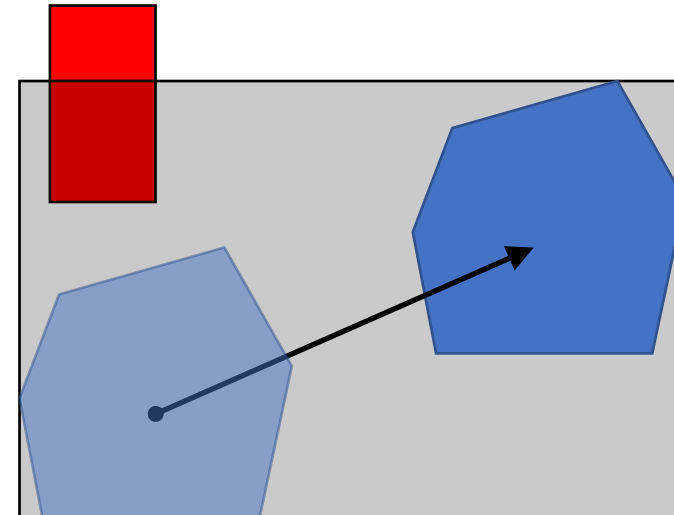
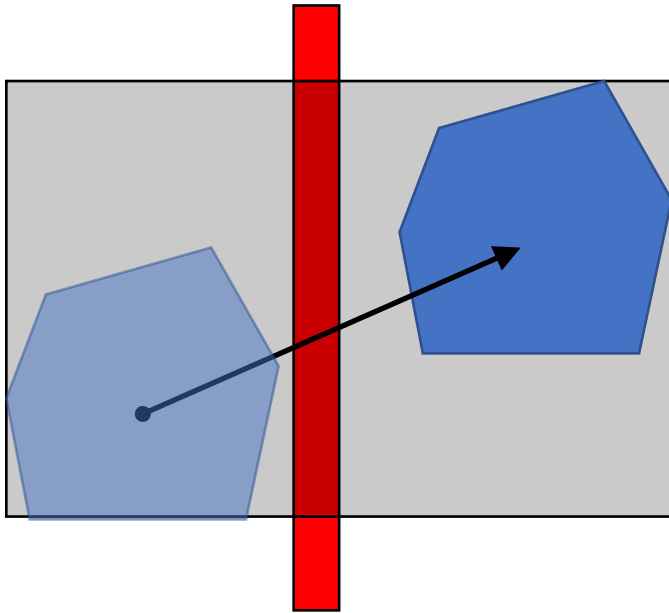


- OBB



# Borner le mouvement

- Si les bornes du mouvement ne s'intersectent pas, il n'y a pas de collision
- Si oui, il y a potentiellement une collision



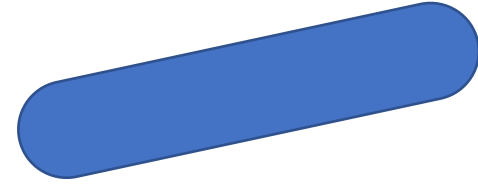
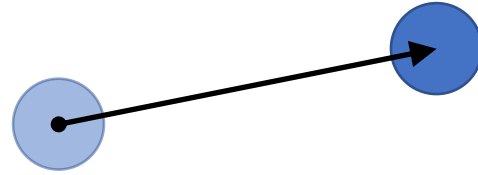
# Swept bornes

- Comme les limites de mouvement basées sur les primitives n'ont pas vraiment de bon ajustement, nous pouvons utiliser des limites balayées
  - Plus précis mais plus coûteux
- Une **swept bound** (or swept shape) ou borne balayée est construite de l'union de toutes les surfaces (volumes) des formes subissant une transformation
  - Nous utilisons la transformation affine de  $t$  à  $t + \Delta t$

# Swept bornes

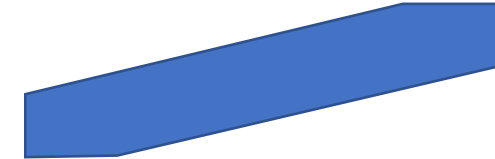
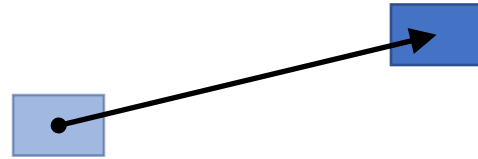
- Swept sphere

➤ capsule



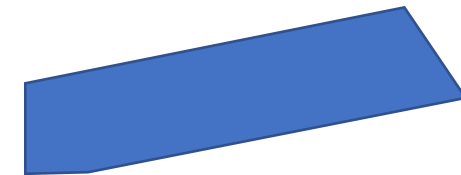
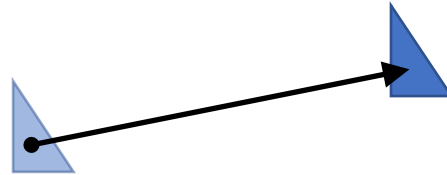
- Swept AABB

➤ convex poly



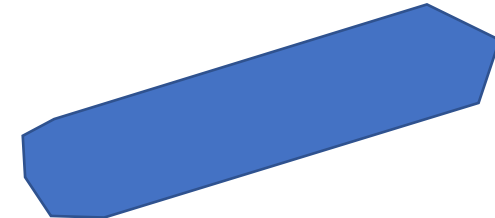
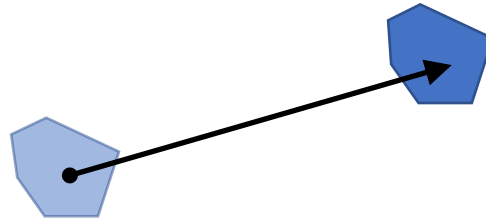
- Swept triangle

➤ convex poly



- Swept convex poly

➤ convex poly



# La résolution des contraintes

Mission d'un solveur de physique :

Mettre à jour la position et les vitesses linéaires et angulaires de tous les corps de manière à satisfaire au mieux toutes les contraintes :

- Les contraintes issues des joints créés par l'utilisateur
- Les contraintes issues des collisions

En conservant au mieux :

- L'énergie
- La quantité de mouvement