

MOTEURS PHYSIQUES

Alexis Vaisse

20 novembre 2015



UBISOFT

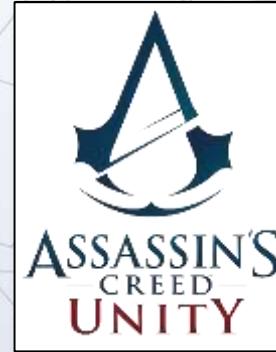
Introduction / Alexis Vaisse

- Lead programmer & software architect – Ubisoft Montpellier
- Dans les jeux vidéo depuis 20 ans

- Motion Physics



- Motion Cloth



Introduction / Moteurs physiques existants

- Havok Physics (Microsoft)
- nVidia PhysX
- Box2D
- Bullet
- ODE (Open Dynamics Engine)
- Newton
- Tokamak
- ...



Introduction / Que fait un moteur physique ?

Rigid body
dynamics



UBISOFT®



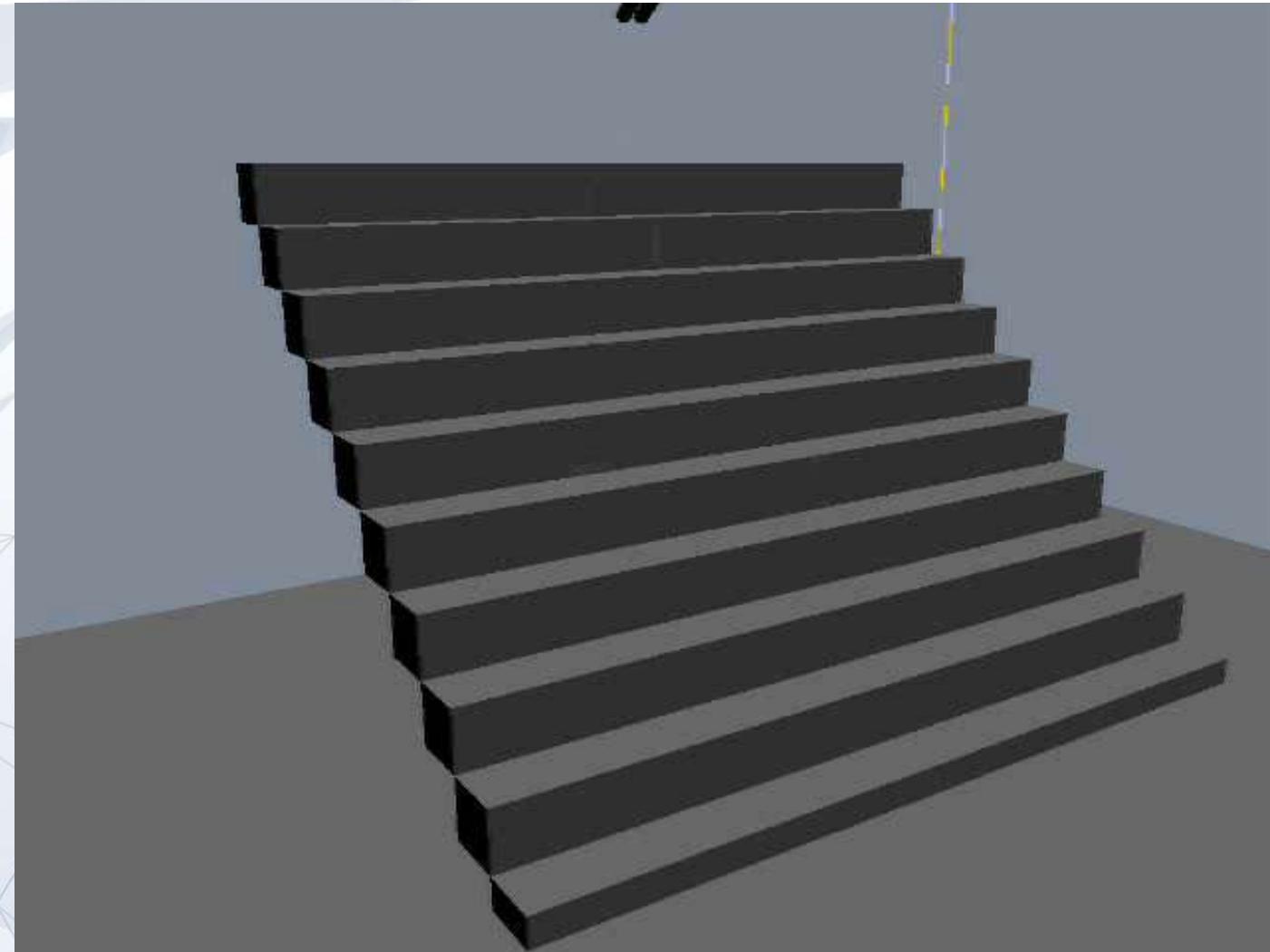
Introduction / Que fait un moteur physique ?



UBISOFT



Introduction / Que fait un moteur physique ?



UBISOFT

▶ Introduction / Que fait un moteur physique ?



UBISOFT



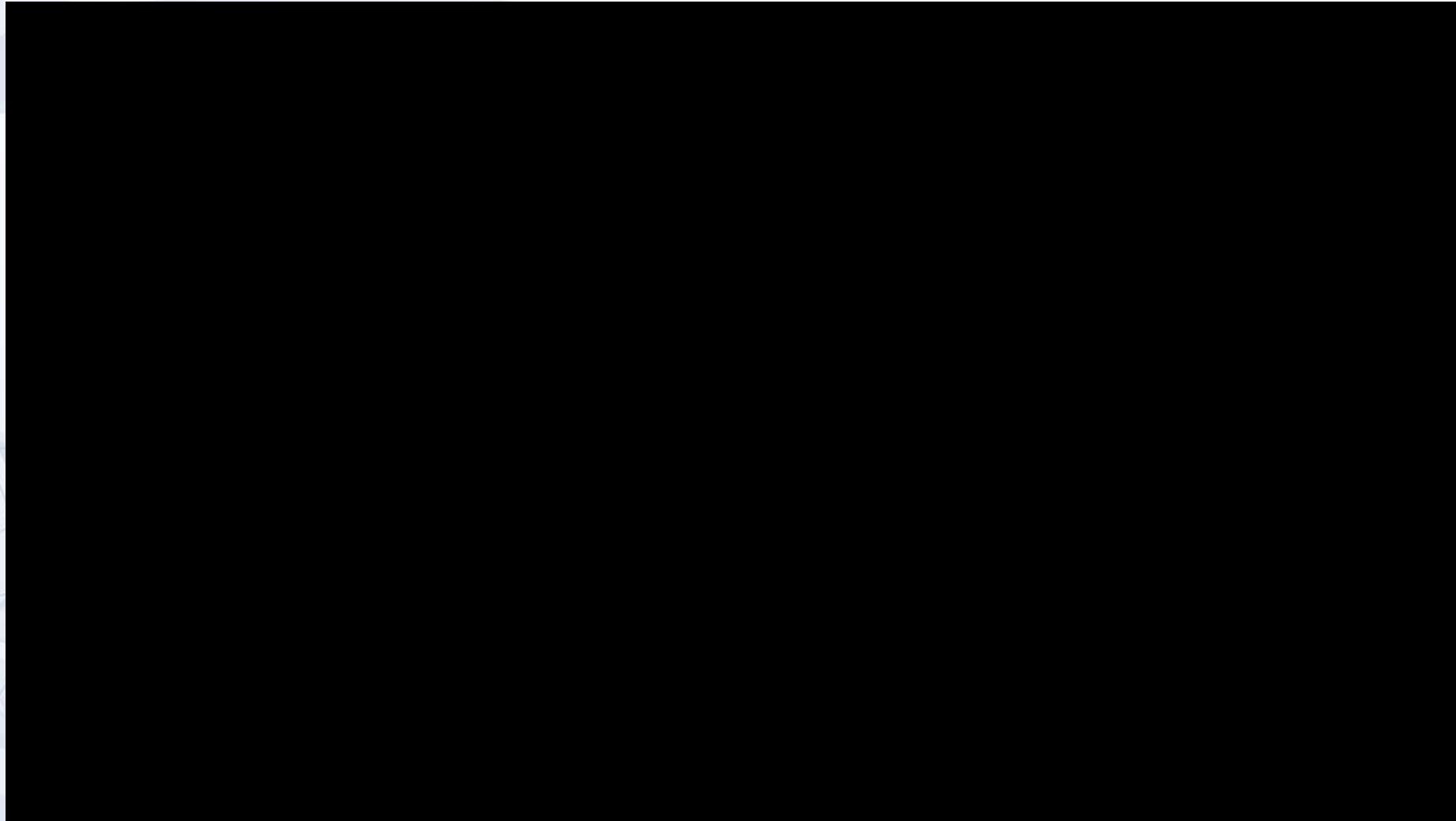
Introduction / Que fait un moteur physique ?



UBISOFT



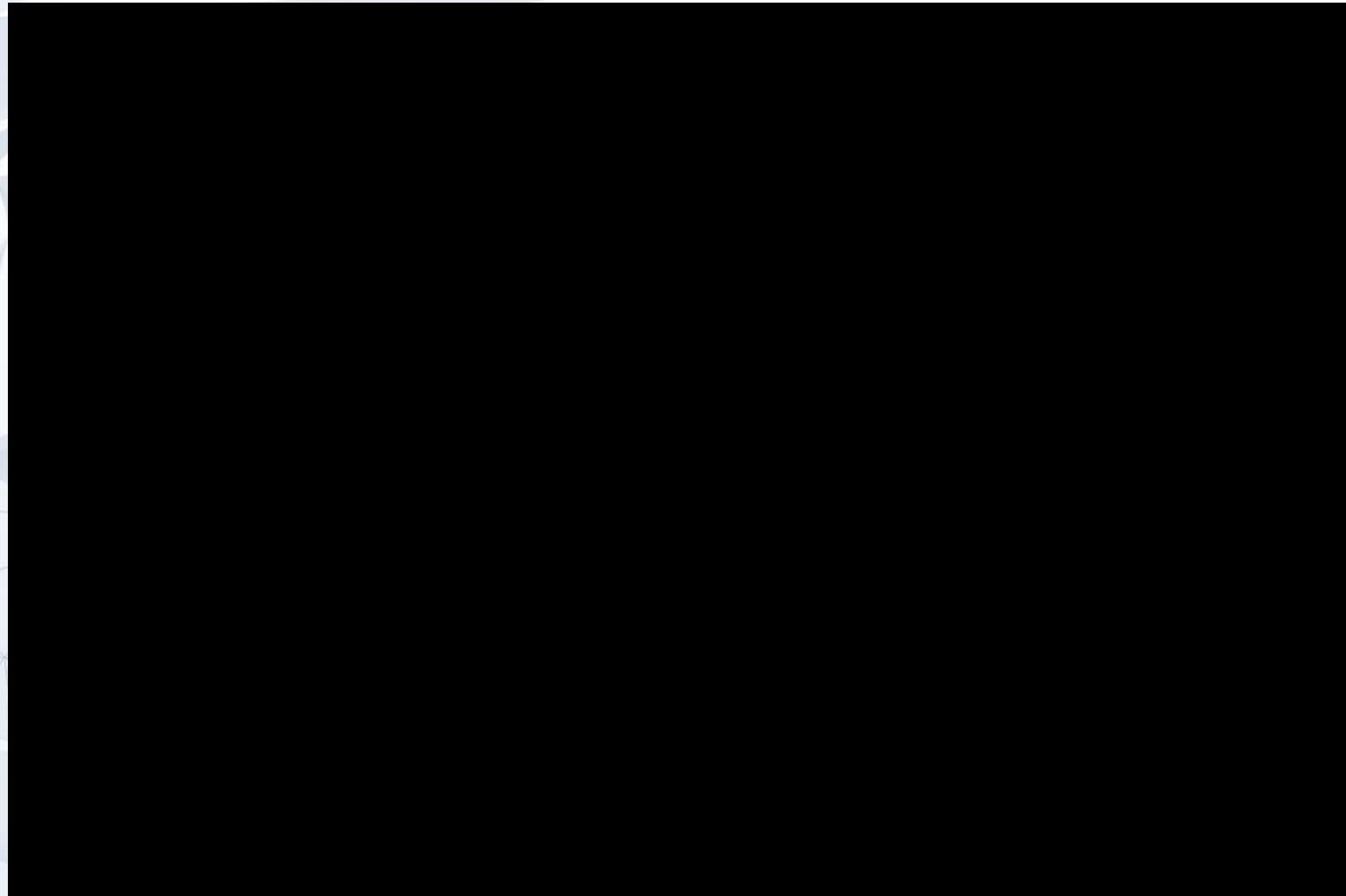
Introduction / Que fait un moteur physique ?



UBISOFT®



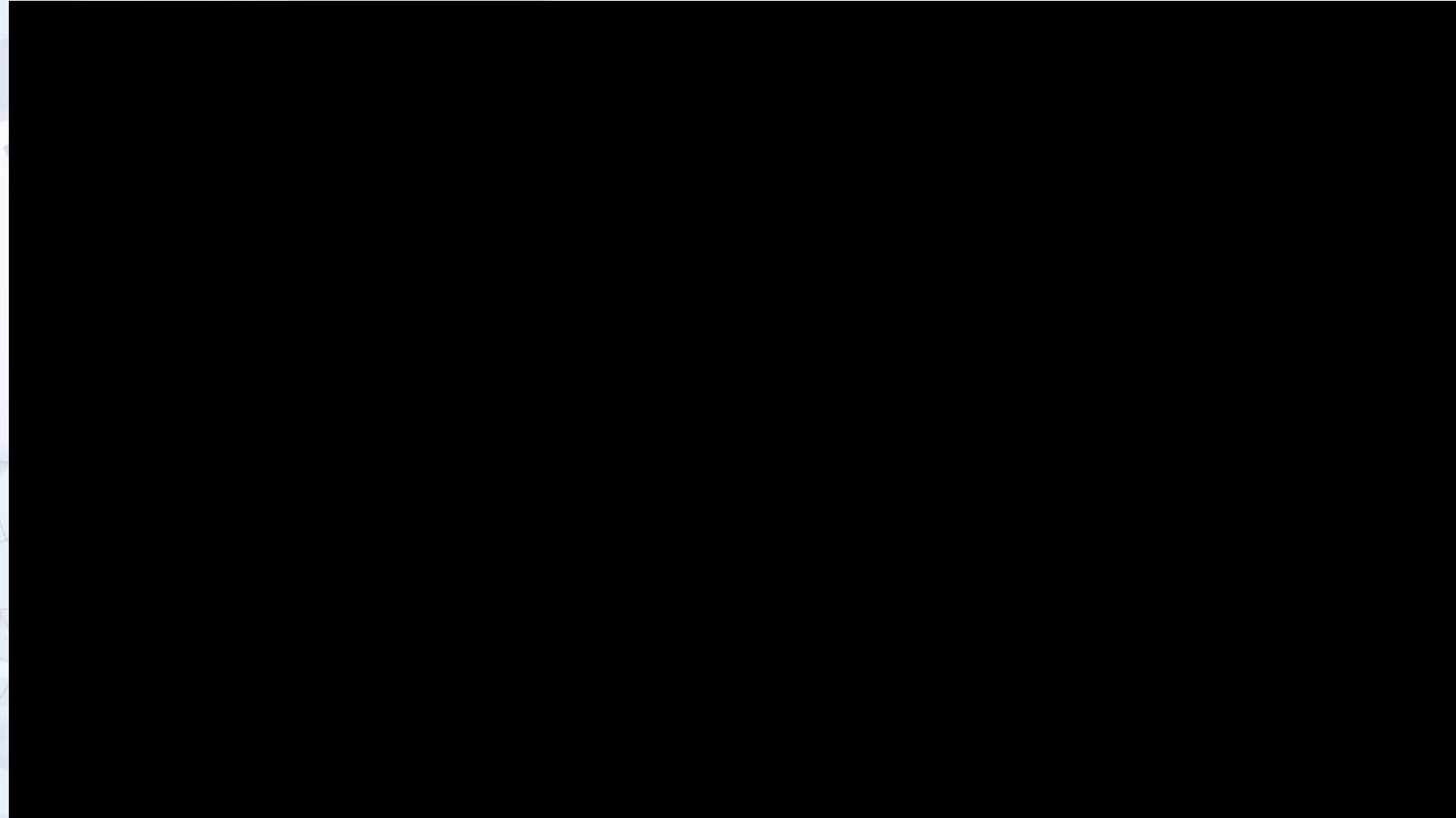
Introduction / Que fait un moteur physique ?



UBISOFT®



Introduction / Que fait un moteur physique ?



UBISOFT®



Introduction / Que fait un moteur physique ?

Rigid body
dynamics

Soft body
dynamics

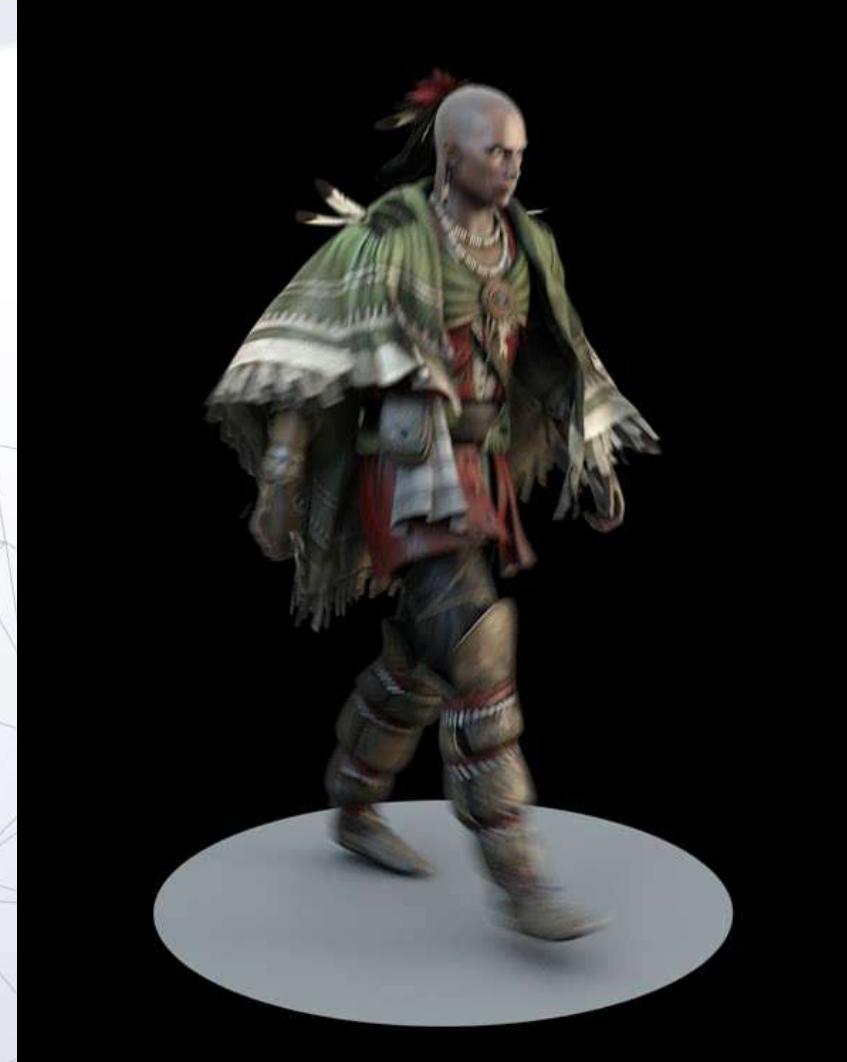
2D



UBISOFT®



Introduction / Que fait un moteur physique ?



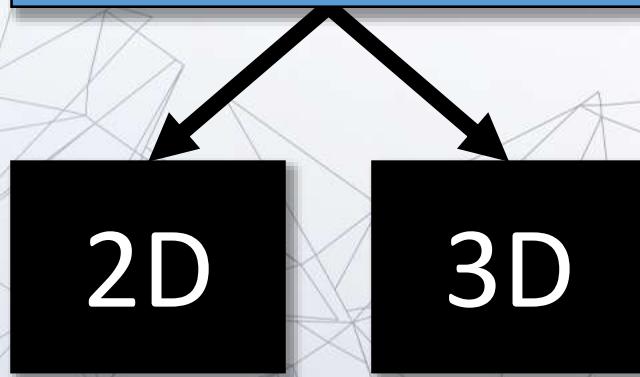
UBISOFT®



Introduction / Que fait un moteur physique ?

Rigid body
dynamics

Soft body
dynamics

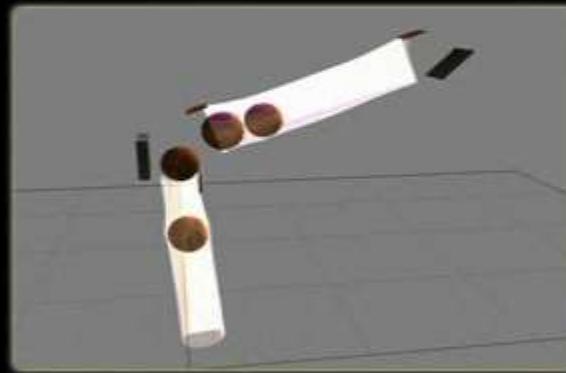
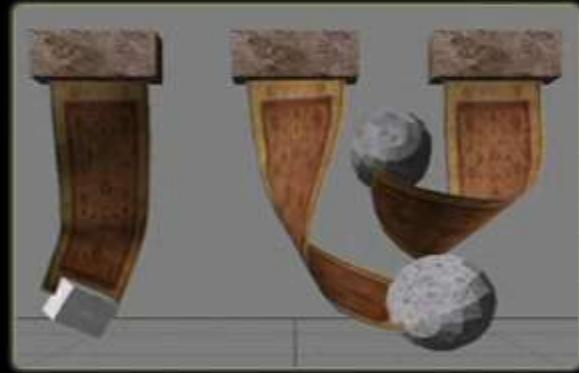


UBISOFT



Introduction / Que fait un moteur physique ?

Two Way Interaction with Rigid Bodies



UBISOFT®



Introduction / Que fait un moteur physique ?



UBISOFT



Introduction / Que fait un moteur physique ?



UBISOFT

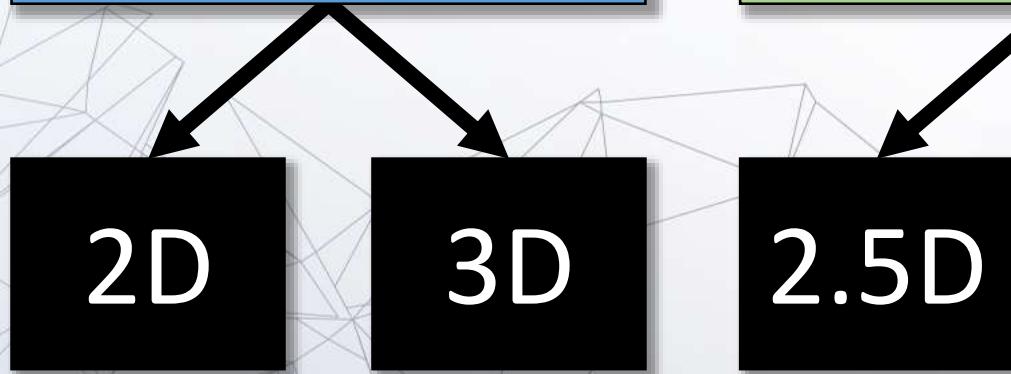


Introduction / Que fait un moteur physique ?

Rigid body
dynamics

Soft body
dynamics

Fluid
dynamics



UBISOFT®



Introduction / Que fait un moteur physique ?



UBISOFT®



Introduction / Que fait un moteur physique ?

Rigid body
dynamics

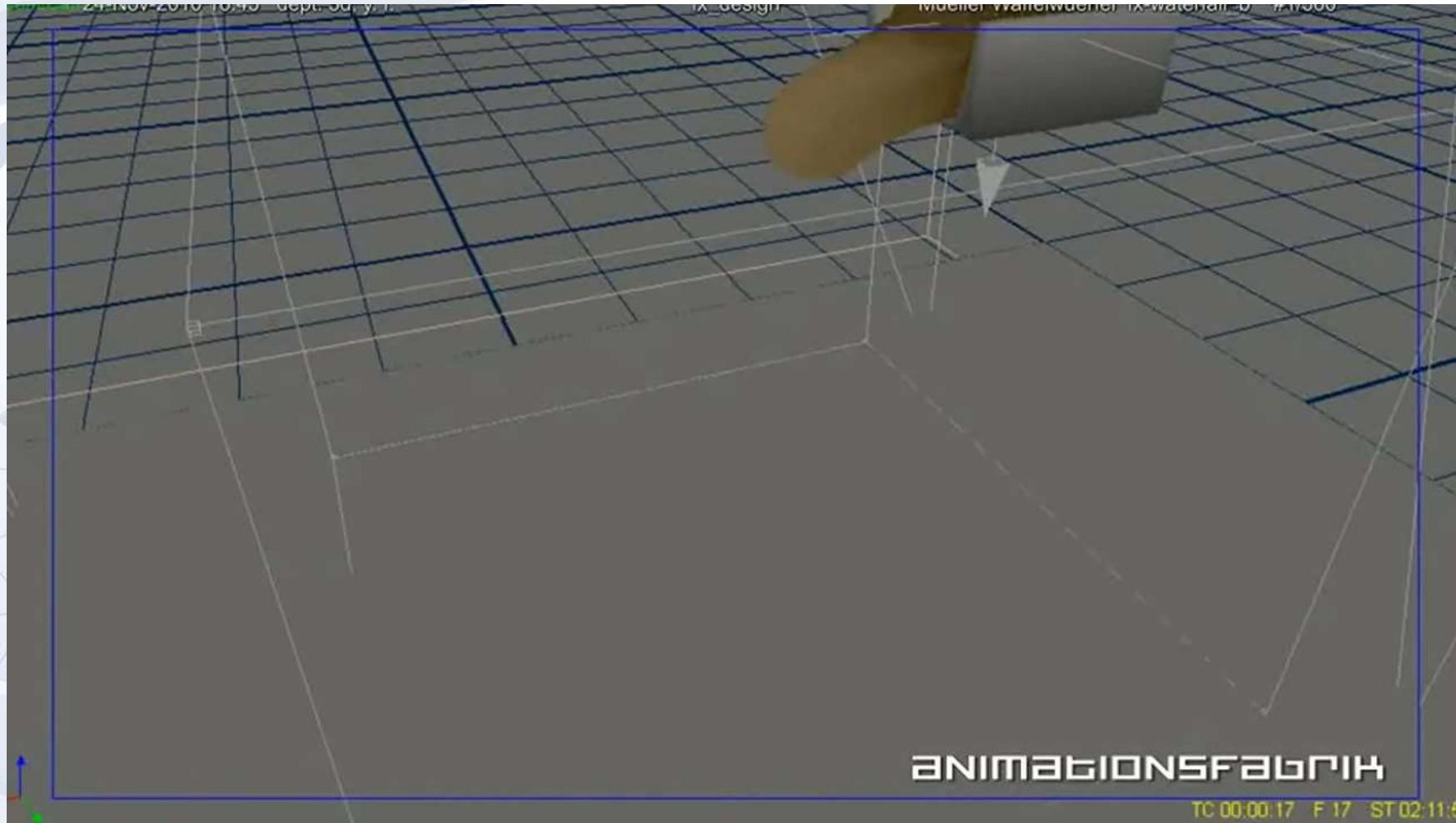
Soft body
dynamics

Fluid
dynamics





Introduction / Que fait un moteur physique ?



UBISOFT

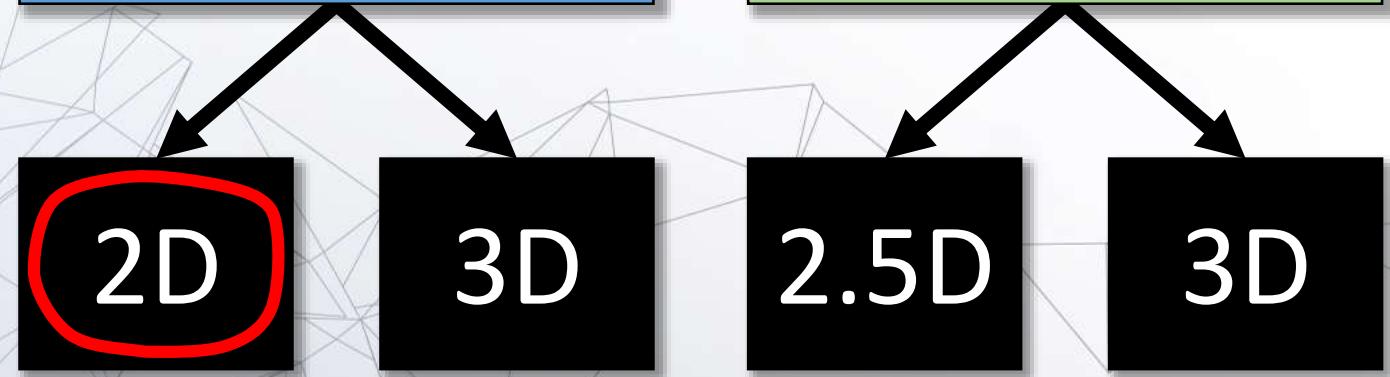


Introduction / Que fait un moteur physique ?

Rigid body
dynamics

Soft body
dynamics

Fluid
dynamics





Sommaire

2h30 – 3h00

1. La physique des corps rigides indéformables
2. La simulation de vêtements 0h30
3. Le travail d'un programmeur physique 0h30



UBISOFT®

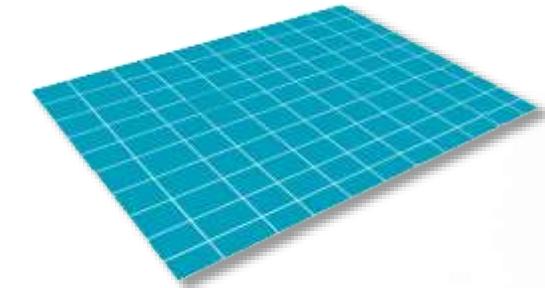
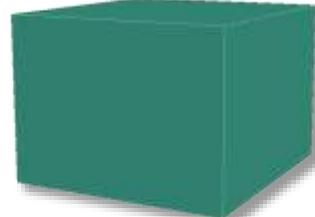
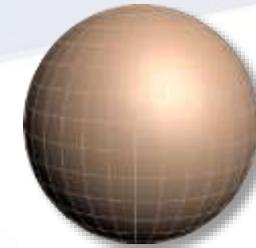
La physique des corps rigides indéformables



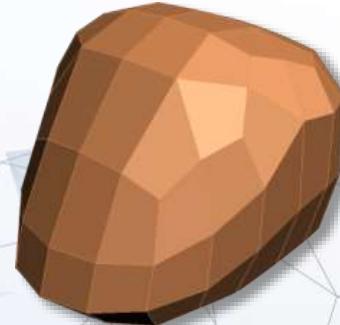
UBISOFT®

La physique des corps rigides indéformables

Primitives :



Meshes :

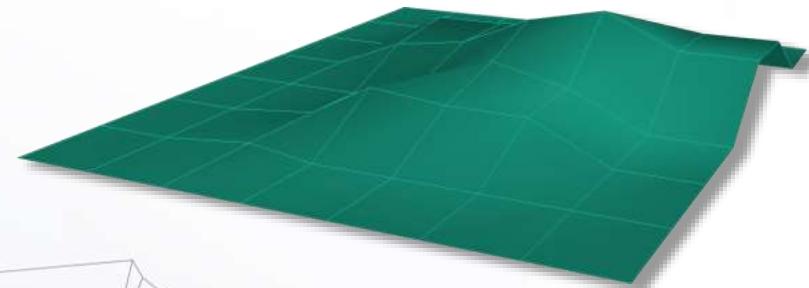


Convexe

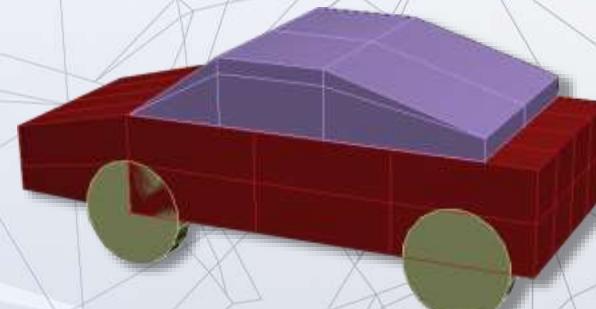


Quelconque

Heightfield :



Composés :



UBISOFT



La physique des corps rigides indéformables

Mission n° 1 :

Déetecter les collisions

Mission n° 2 :

Résoudre les contraintes

Contraintes issues
des collisions

Contraintes issues
des joints



UBISOFT®

La physique des corps rigides indéformables

Partie 1 : La détection des collisions



UBISOFT

La détection des collisions

Objectifs :

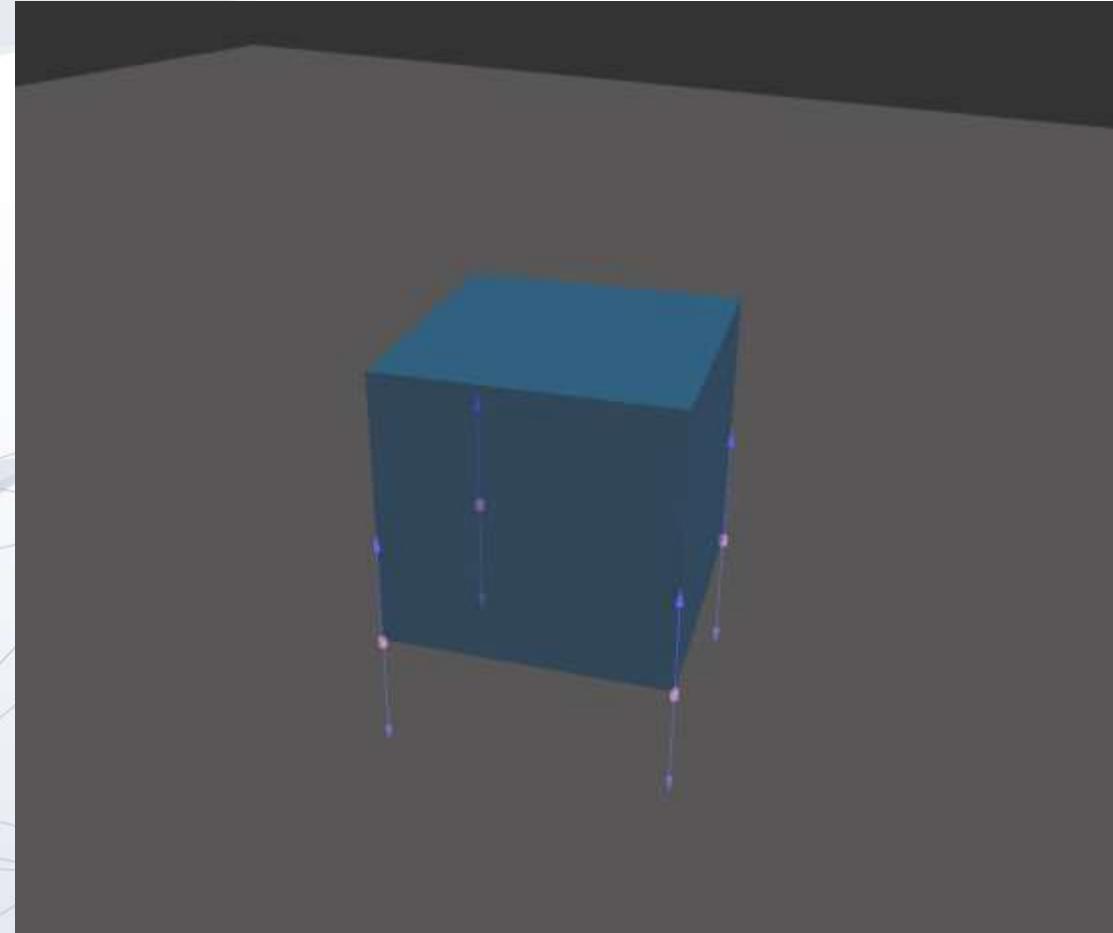
- Fournir la liste des paires de corps qui sont en intersection
- Pour chaque paire de corps :

- 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



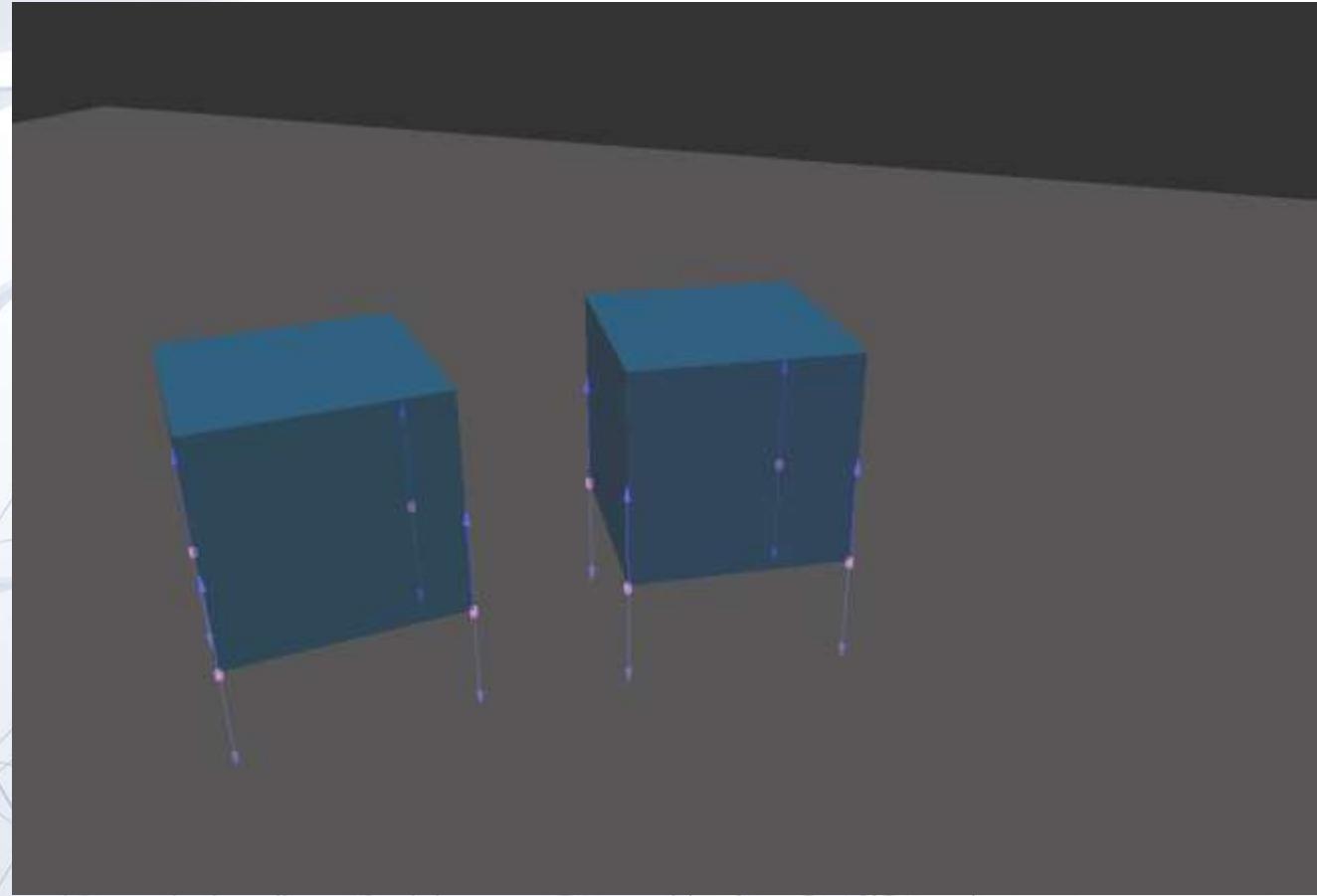
UBISOFT

La détection des collisions



UBISOFT

La détection des collisions



UBISOFT

La détection des collisions

Phase n° 1 :

Broad phase

Déetecter 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.



UBISOFT

La physique des corps rigides indéformables

Partie 1 : La détection des collisions

1.1 La broad phase



UBISOFT

La détection des collisions / Broad phase

Mission de la broad phase :

Déetecter 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



La détection des collisions / Broad phase / Volume englobant

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.



UBISOFT®

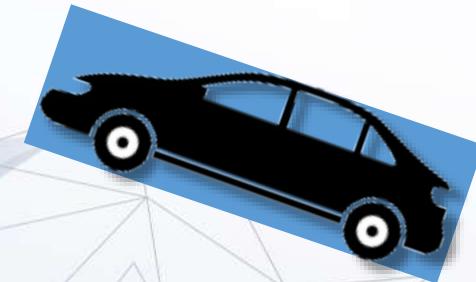


La détection des collisions / Broad phase / Volume englobant

Quelques types de volumes englobants :



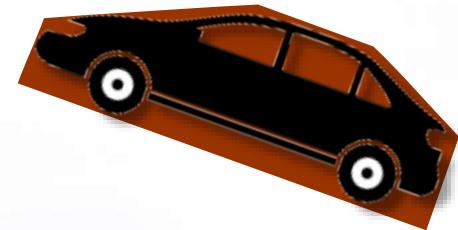
Bounding
sphere



Oriented
Bounding Box
(OBB)



Axis-Aligned
Bounding Box
(AABB)

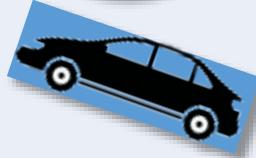
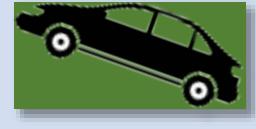
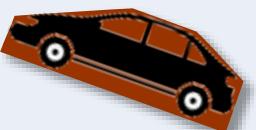


Convex



UBISOFT®

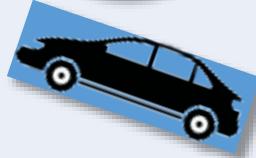
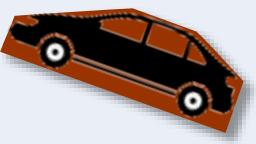
La détection des collisions / Broad phase / Volume englobant

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



UBISOFT

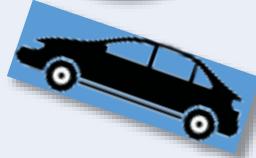
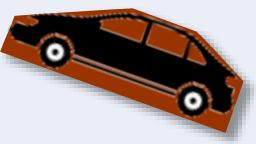
La détection des collisions / Broad phase / Volume englobant

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



UBISOFT®

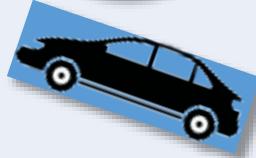
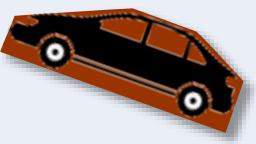
La détection des collisions / Broad phase / Volume englobant

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	



UBISOFT

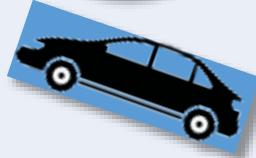
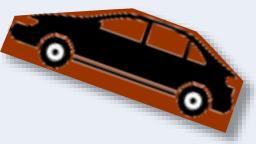
La détection des collisions / Broad phase / Volume englobant

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	



UBISOFT

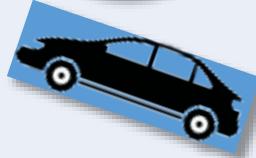
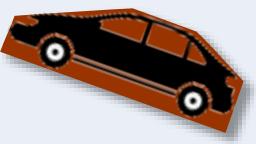
La détection des collisions / Broad phase / Volume englobant

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	



UBISOFT

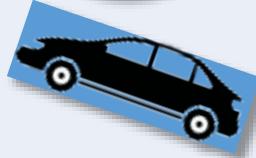
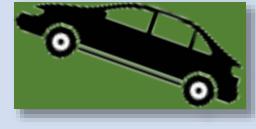
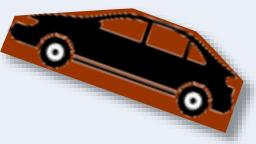
La détection des collisions / Broad phase / Volume englobant

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		



UBISOFT

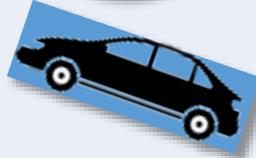
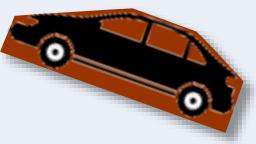
La détection des collisions / Broad phase / Volume englobant

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		



UBISOFT

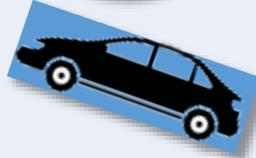
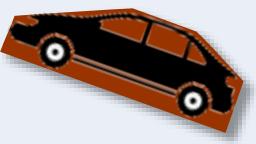
La détection des collisions / Broad phase / Volume englobant

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		



UBISOFT

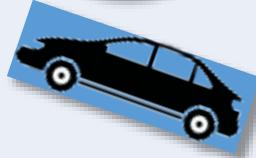
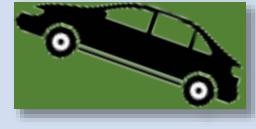
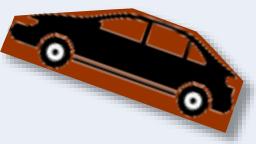
La détection des collisions / Broad phase / Volume englobant

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		



UBISOFT

La détection des collisions / Broad phase / Volume englobant

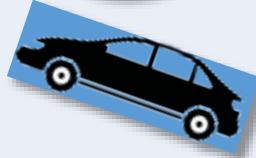
Type de volume englobant	Englobe le corps au plus juste	Rapidité des calculs d'intersection
	Bounding sphere	- ++
	Oriented Bounding Box	++ -
	Axis-Aligned Bounding Box	+
	Convex	+++



UBISOFT



La détection des collisions / 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	+++ --



UBISOFT

La détection des collisions / Broad phase

Broad phase

Stockage des paires de corps potentiellement en collision



- Listes chaînées
- Map
- Hash table



UBISOFT

La détection des collisions / Broad phase

Quelques algorithmes de broad phase :

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

La détection des collisions / Broad phase

Quelques algorithmes de broad phase :

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



UBISOFT

La détection des collisions / Broad phase / Brute force

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

La détection des collisions / Broad phase

Quelques algorithmes de broad phase :

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

La détection des collisions / Broad phase

Quelques algorithmes de broad phase :

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

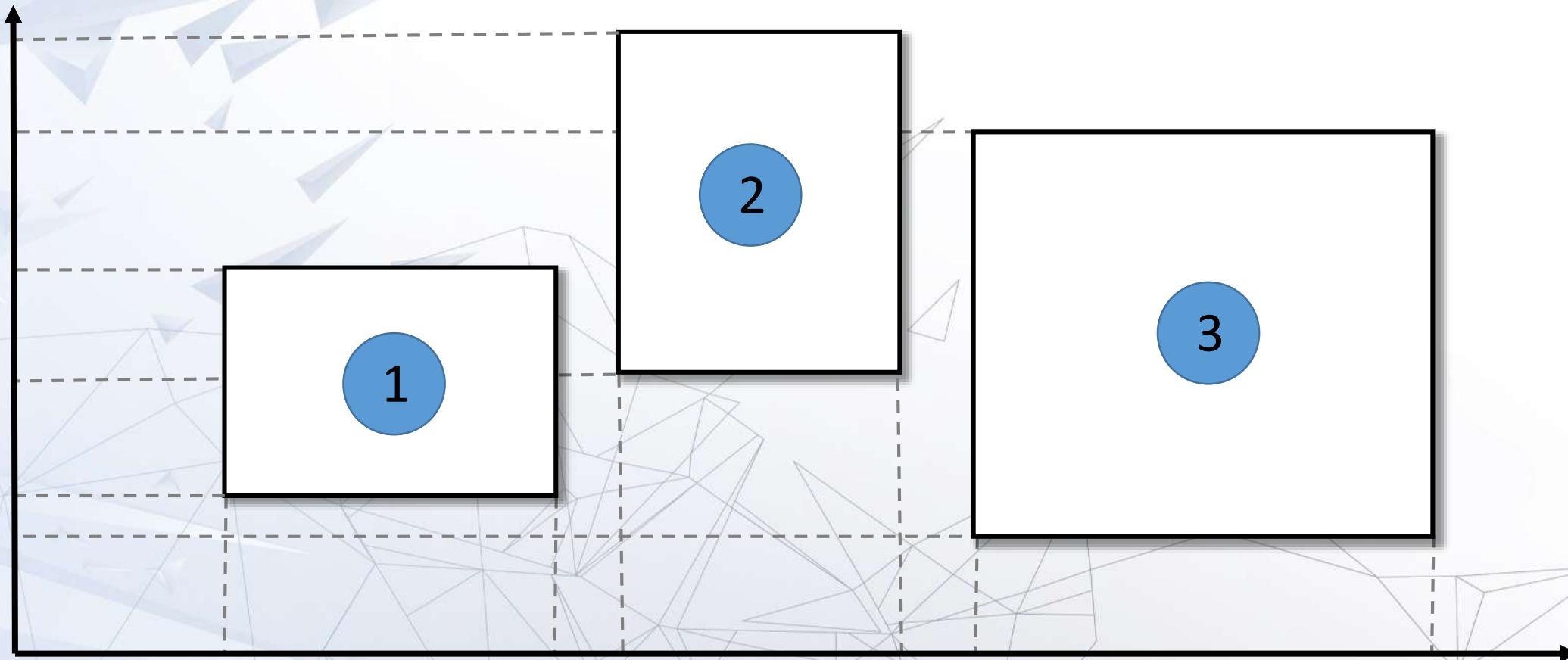


UBISOFT

La détection des collisions / Broad phase / Sweep & Prune

Algorithme Sweep & Prune :

(1992)



La détection des collisions / Broad phase / Sweep & Prune

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

}

64 bits

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



UBISOFT



La détection des collisions / 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

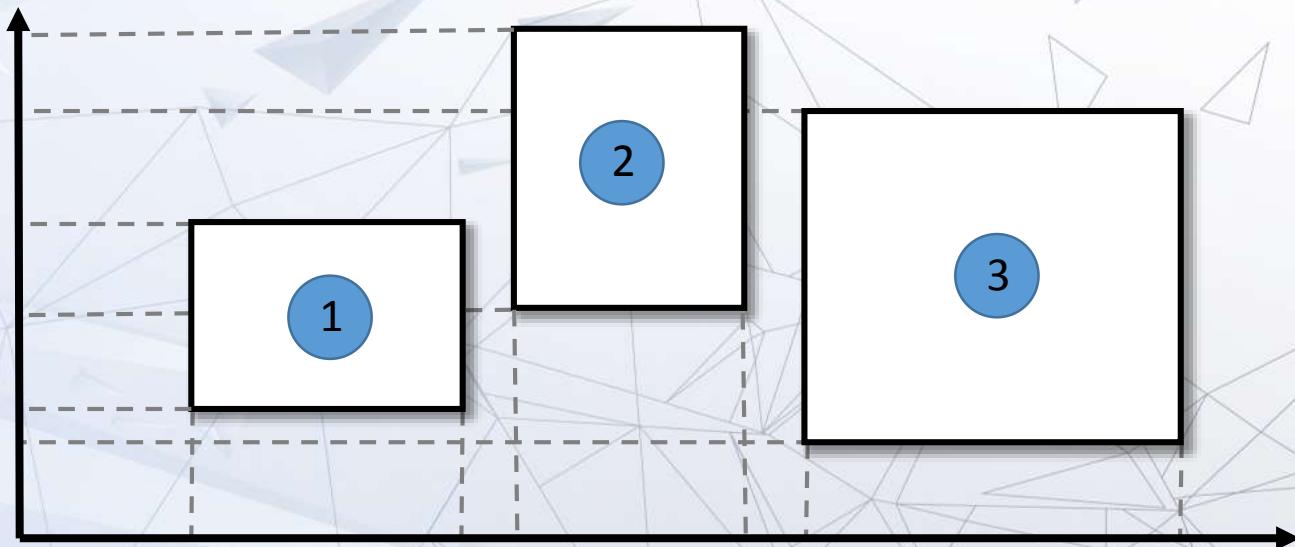
1;Start;1.2

2;Start;2.0

1;End;3.0

3;End;4.0

2;End;4.5



La détection des collisions / 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

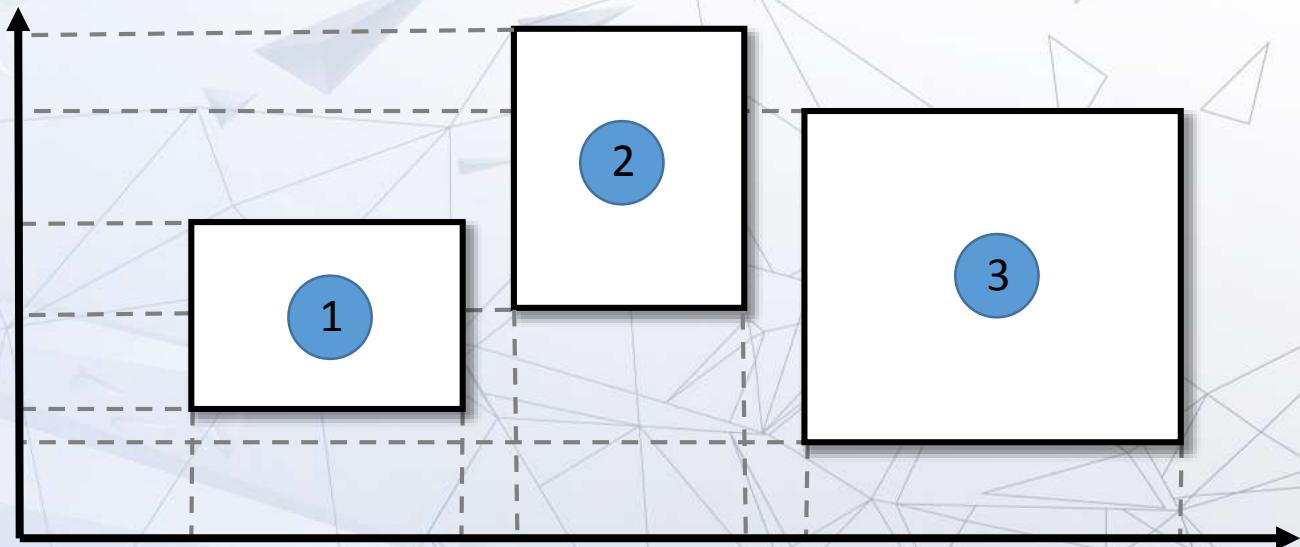
1;Start;1.2

2;Start;2.0

1;End;3.0

3;End;4.0

2;End;4,5





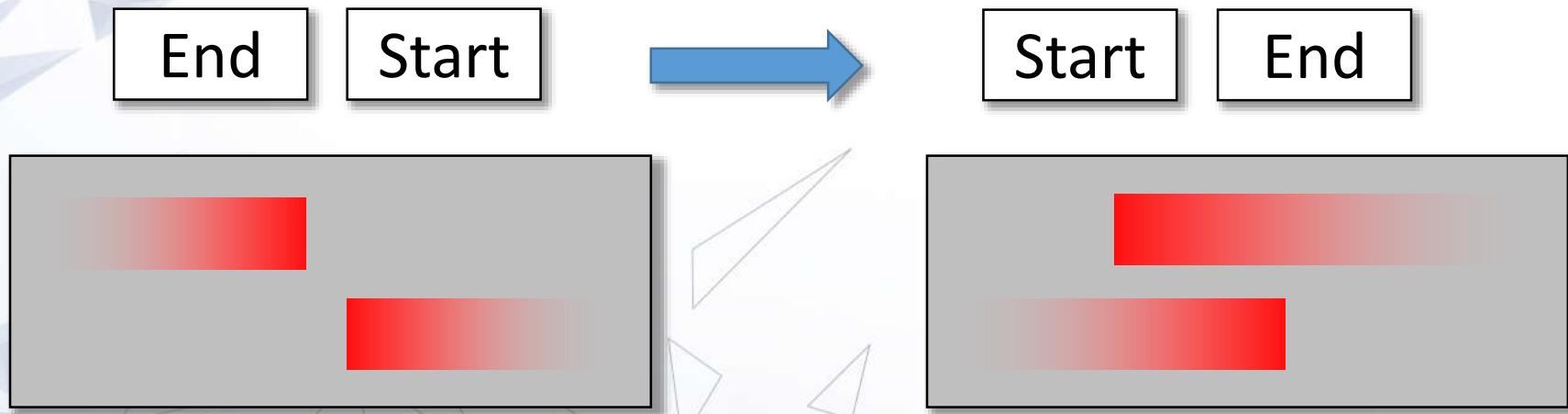
La détection des collisions / Broad phase / Sweep & Prune

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

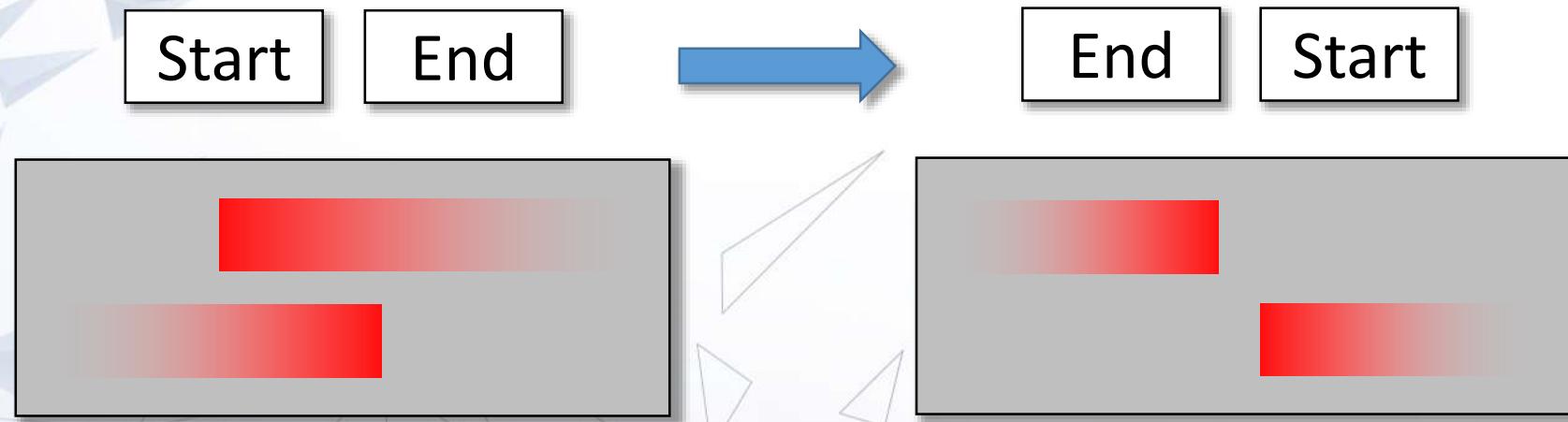


La détection des collisions / Broad phase / Sweep & Prune



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

La détection des collisions / Broad phase / Sweep & Prune



Si intersection sur les autres axes :
On enlève la paire



UBISOFT

La détection des collisions / Broad phase / Sweep & Prune

Complexité de la mise à jour à chaque trame :

- Pire des cas : **O(n²)** (pas de cohérence d'une trame à l'autre)
- En pratique : **~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

La détection des collisions / Broad phase

Quelques algorithmes de broad phase :

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



UBISOFT

La détection des collisions / Broad phase

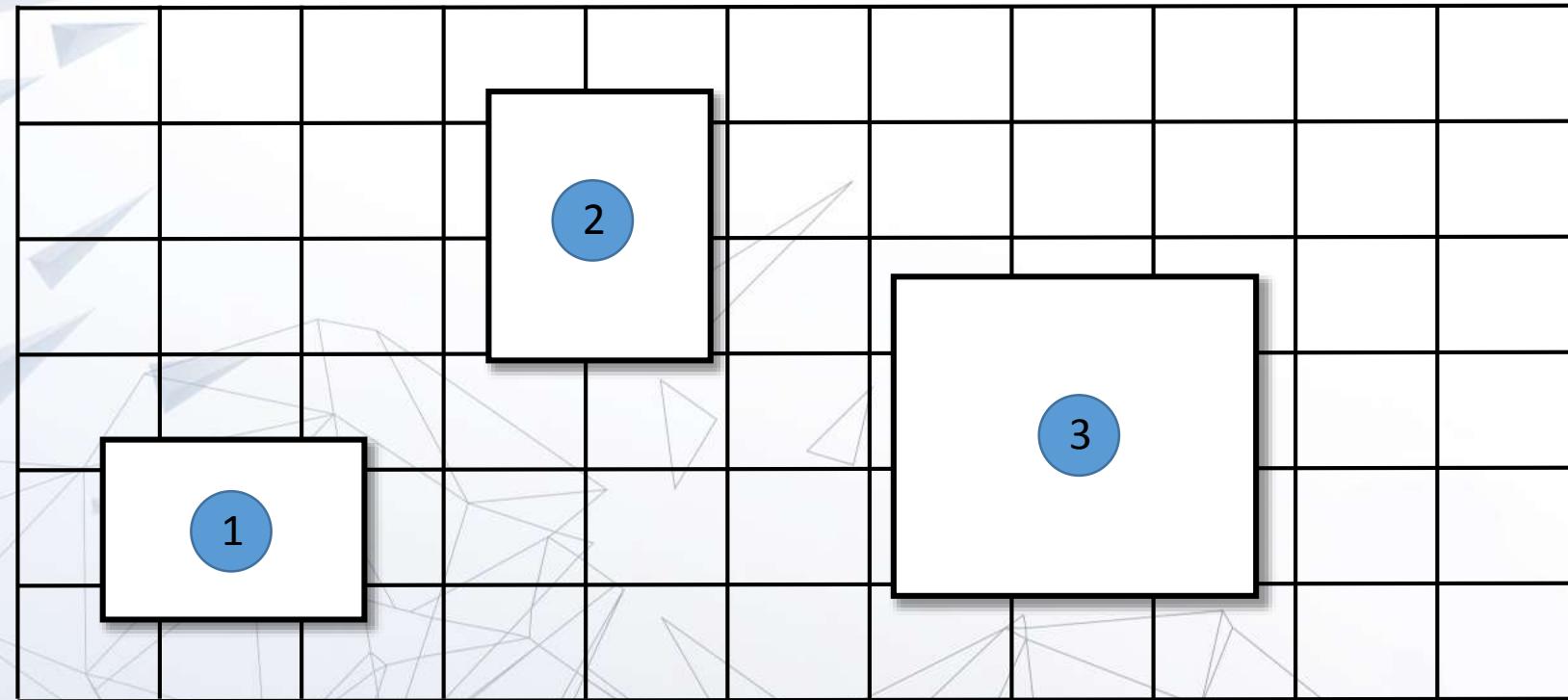
Quelques algorithmes de broad phase :

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



UBISOFT

La détection des collisions / Broad phase / Grille



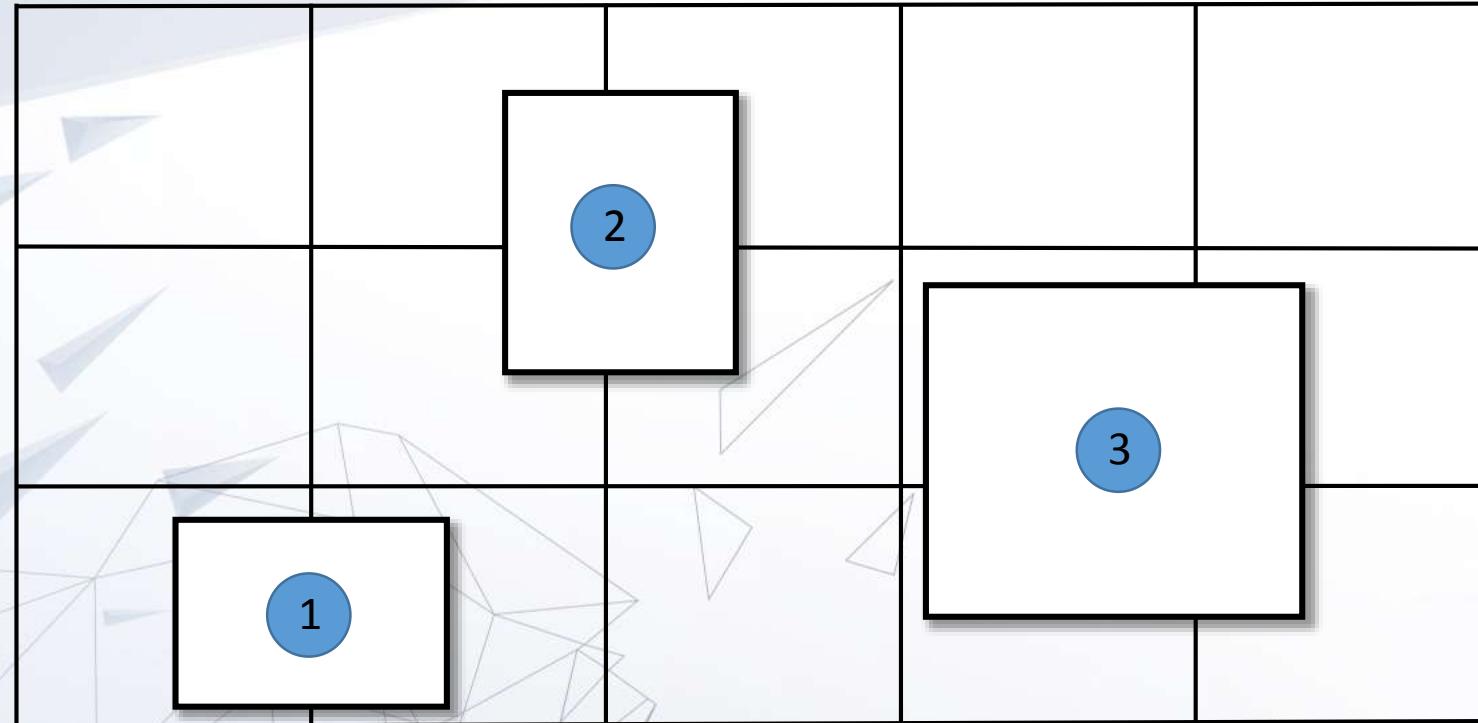
UBISOFT

La détection des collisions / 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



La détection des collisions / Broad phase / Grille

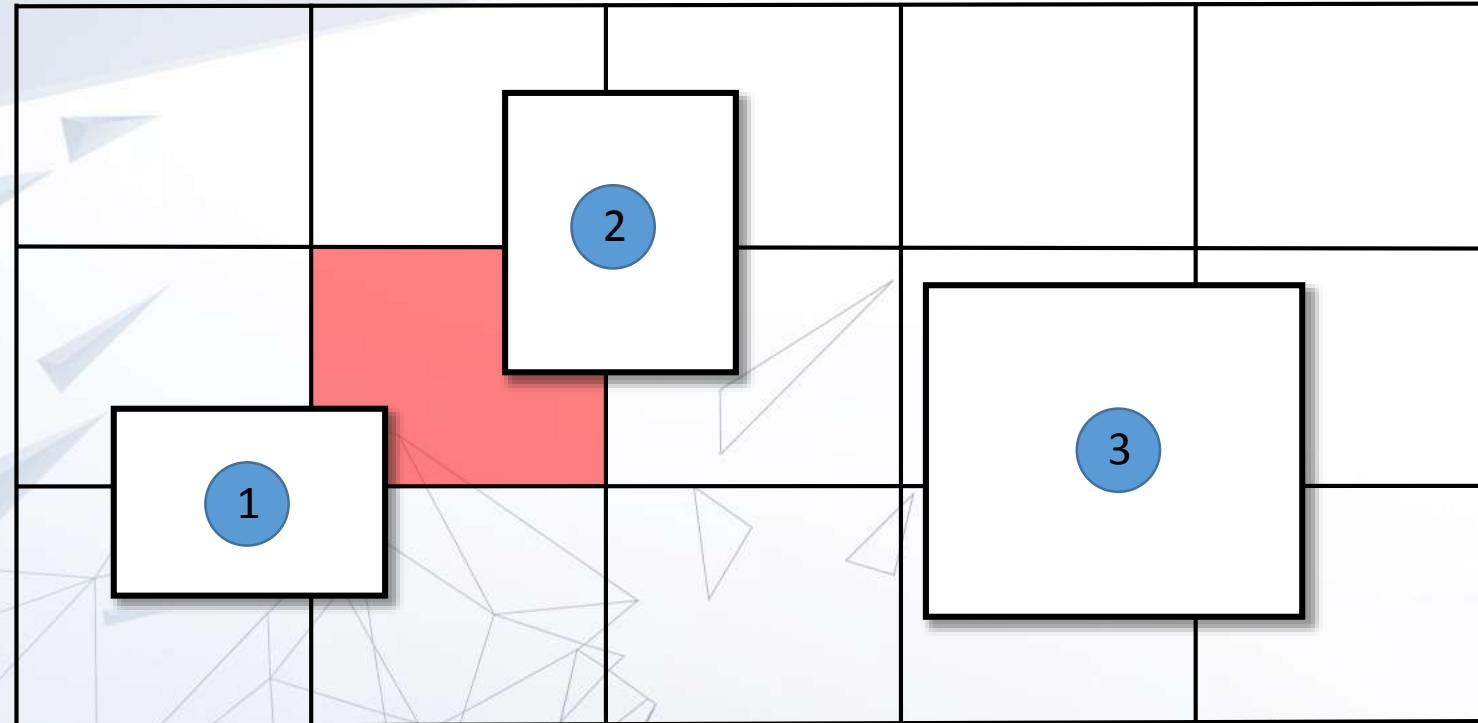


0 calcul d'intersection



UBISOFT

La détection des collisions / Broad phase / Grille

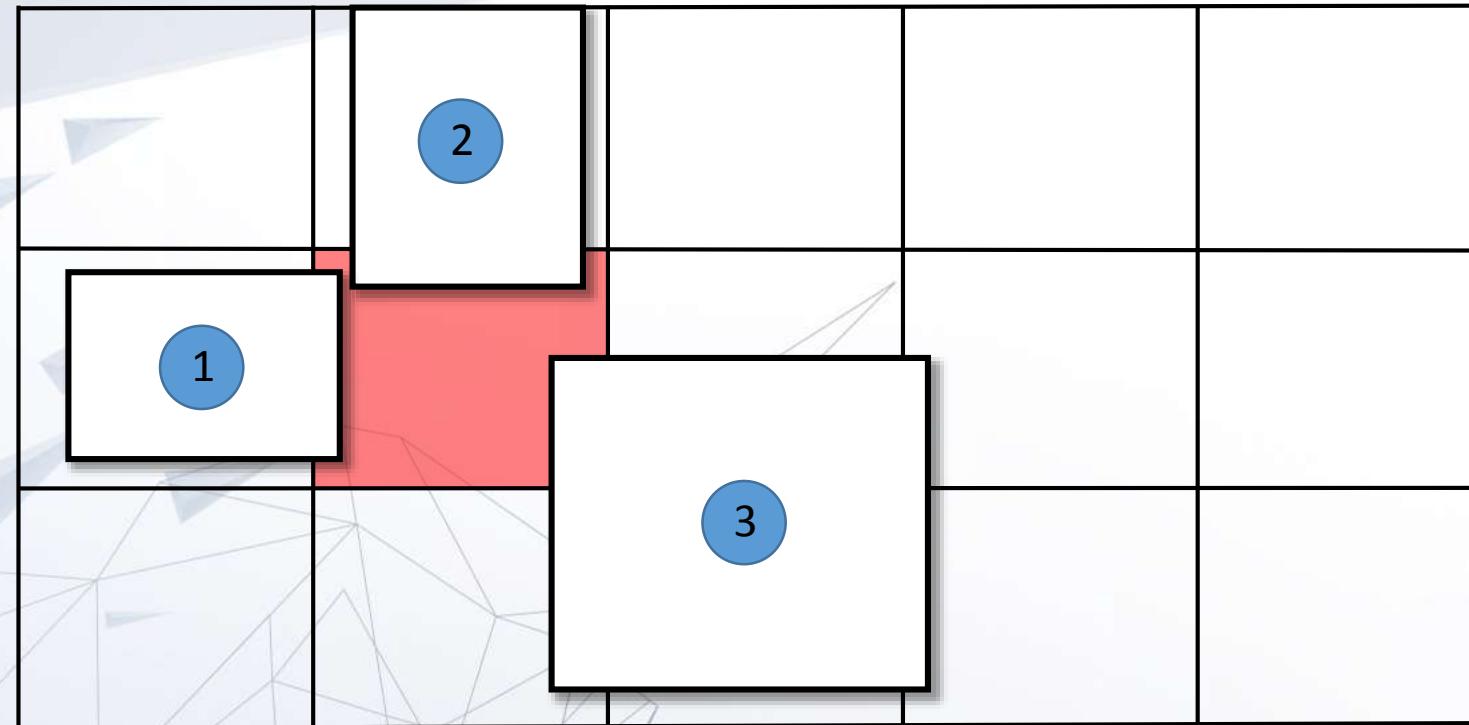


1 calcul d'intersection



UBISOFT®

La détection des collisions / Broad phase / Grille

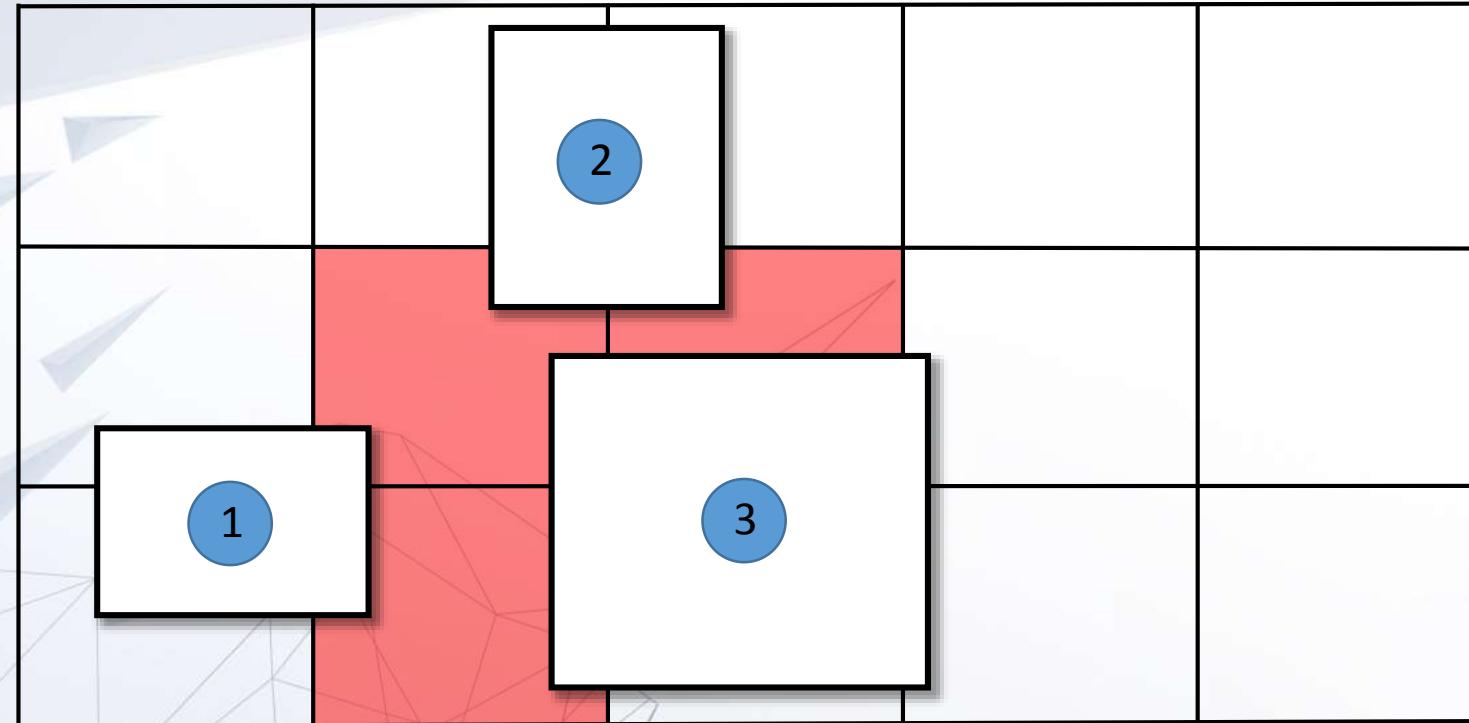


3 calculs d'intersection



UBISOFT

La détection des collisions / Broad phase / Grille



5 calculs d'intersection



UBISOFT®

La détection des collisions / 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

La détection des collisions / Broad phase

Quelques algorithmes de broad phase :

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



UBISOFT

La détection des collisions / 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



La détection des collisions / Broad phase / Les arbres

Quelques exemples d'arbres :

- Quadtree
- Octree
- k-d tree



UBISOFT®



La détection des collisions / Broad phase / Les arbres

Quelques exemples d'arbres :

- Quadtree
- Octree
- k-d tree

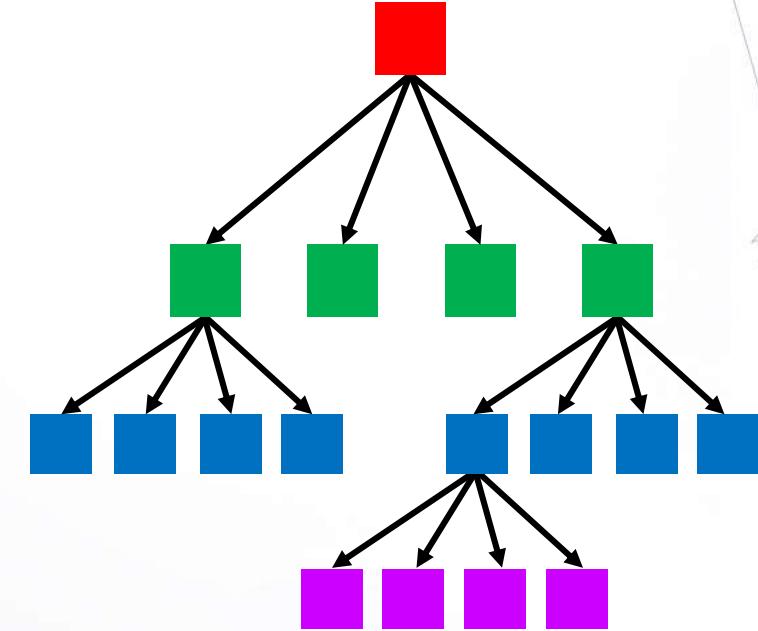
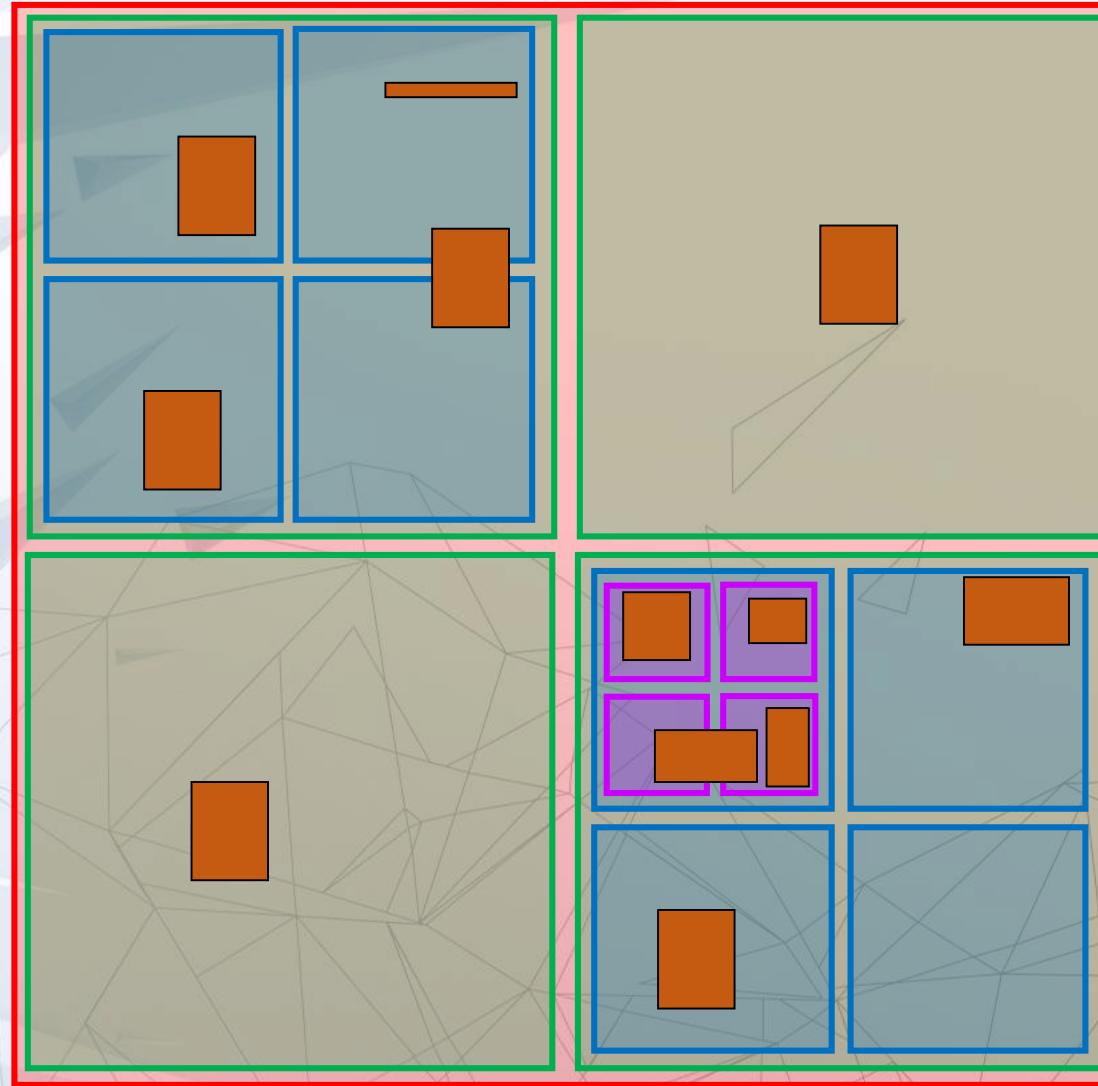


UBISOFT®



La détection des collisions / Broad phase / Les arbres

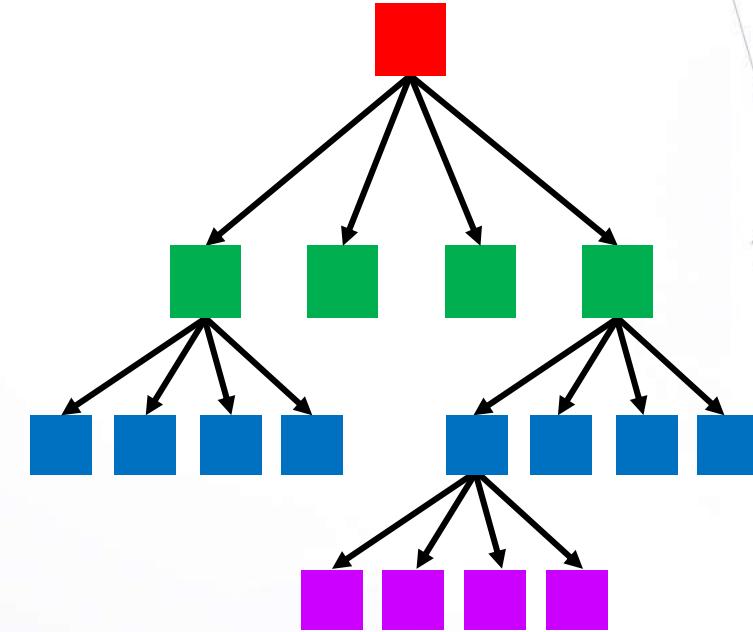
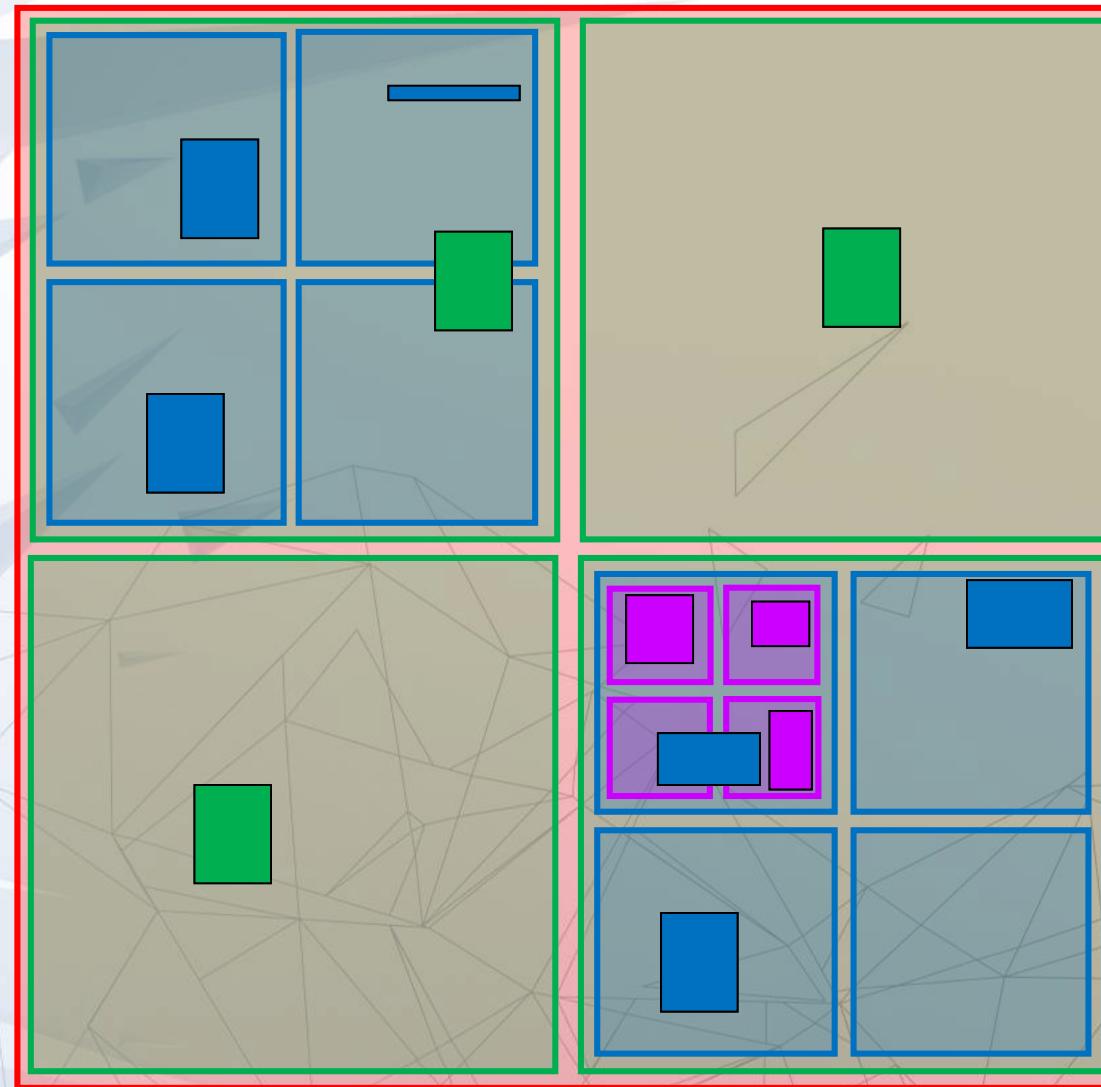
Quadtree :



UBISOFT®

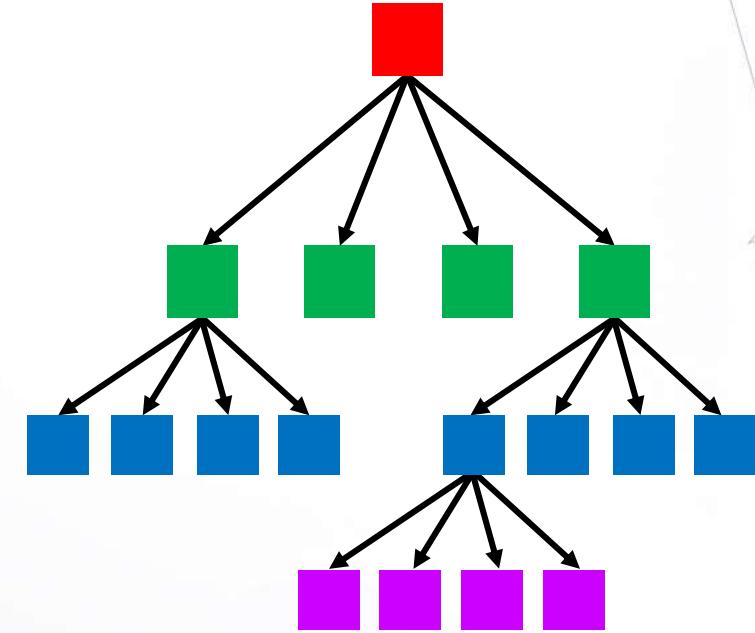
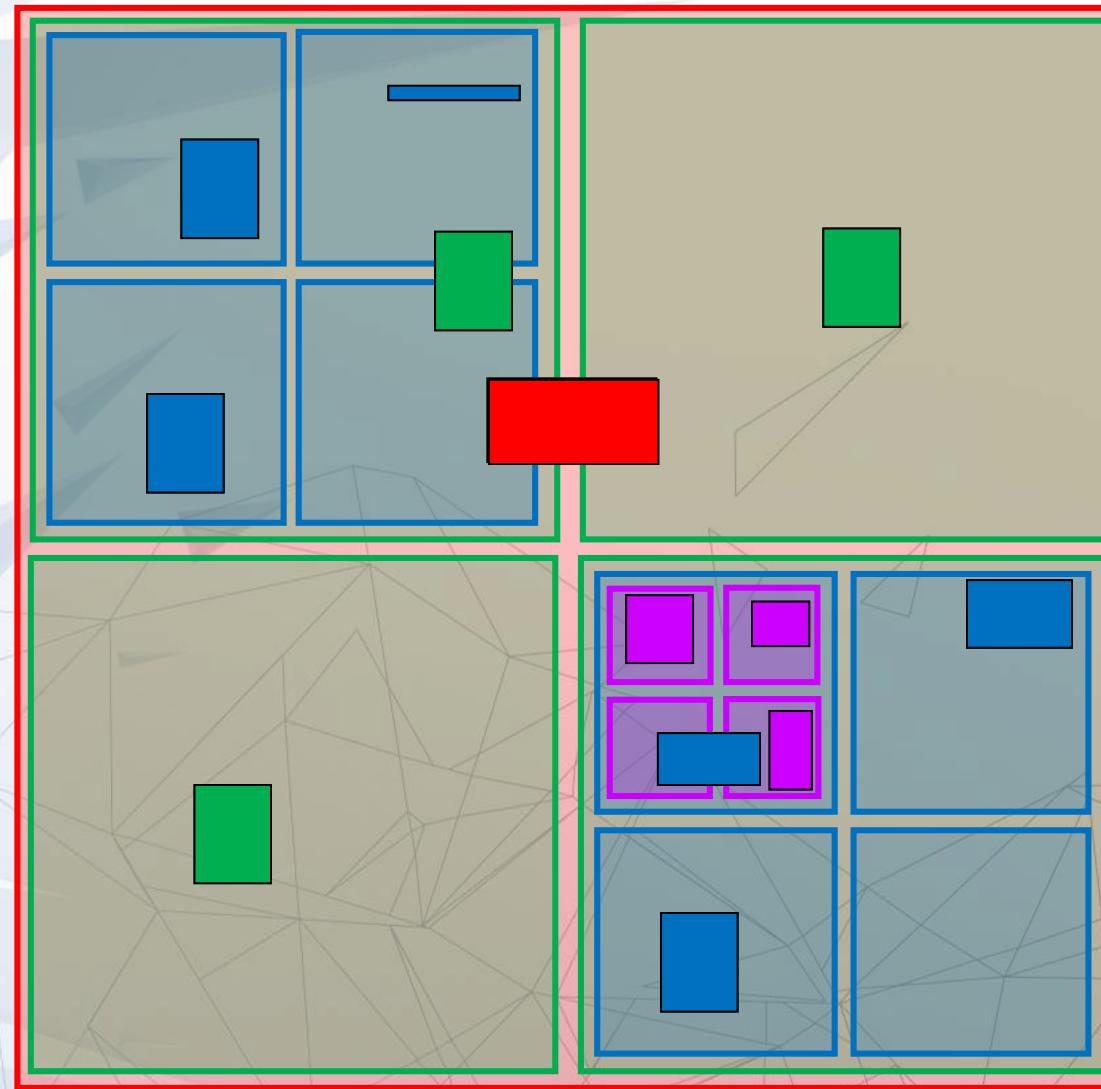
La détection des collisions / Broad phase / Les arbres

Quadtree :



La détection des collisions / Broad phase / Les arbres

Quadtree :





La détection des collisions / 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



La détection des collisions / 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



UBISOFT



La détection des collisions / Broad phase / Les arbres

Quelques exemples d'arbres :

- Quadtree
- Octree
- k-d tree



UBISOFT®

La détection des collisions / Broad phase / Les arbres

Quelques exemples d'arbres :

- Quadtree
- Octree
- k-d tree

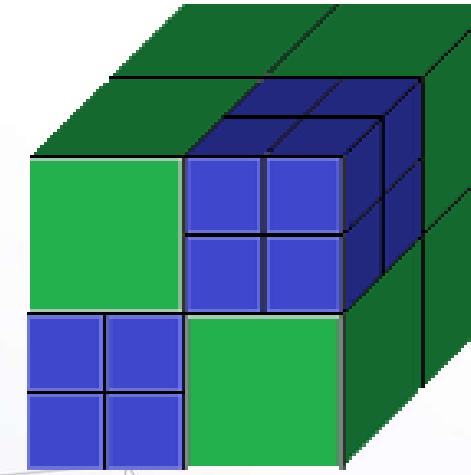
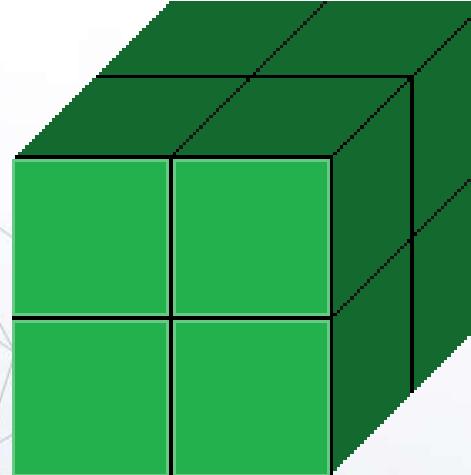
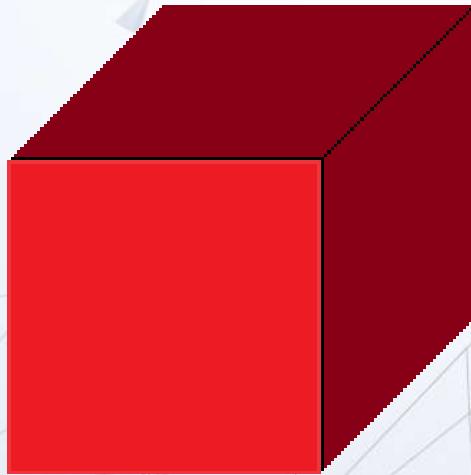


UBISOFT®



La détection des collisions / Broad phase / Les arbres

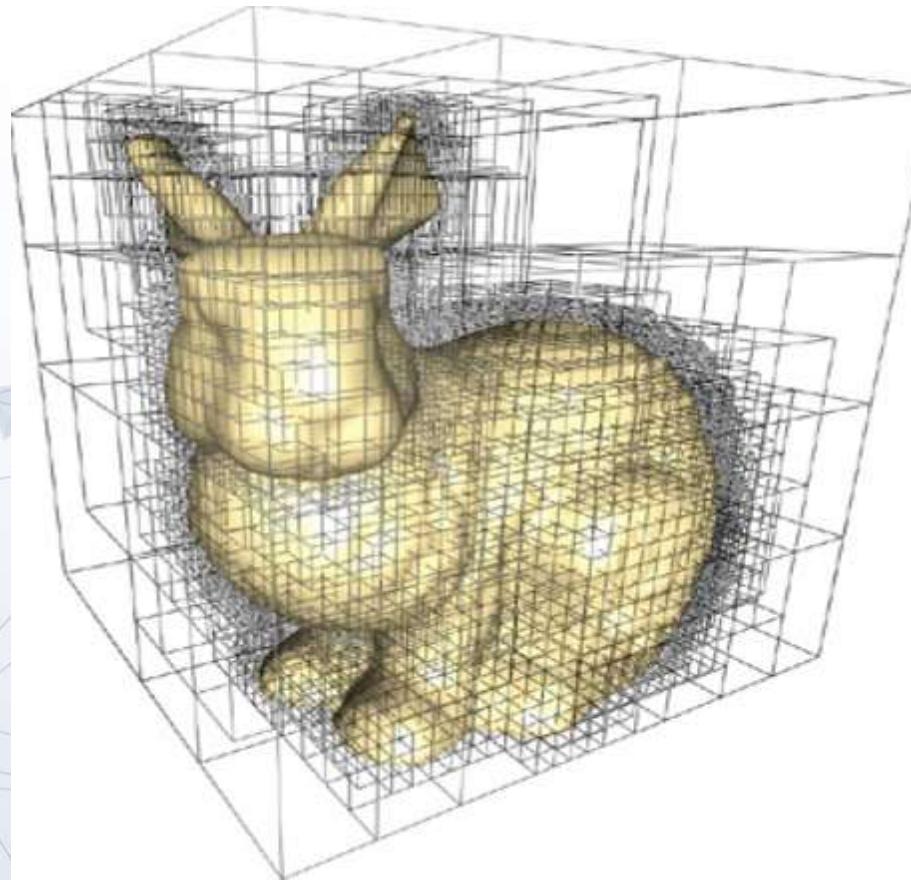
Octree :



UBISOFT

▶ La détection des collisions / Broad phase / Les arbres

Octree :



UBISOFT

La détection des collisions / 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~~



La détection des collisions / Broad phase / Les arbres

Quelques exemples d'arbres :

- Quadtree
- Octree
- k-d tree



UBISOFT®

La détection des collisions / Broad phase / Les arbres

Quelques exemples d'arbres :

- Quadtree
- Octree
- k-d tree

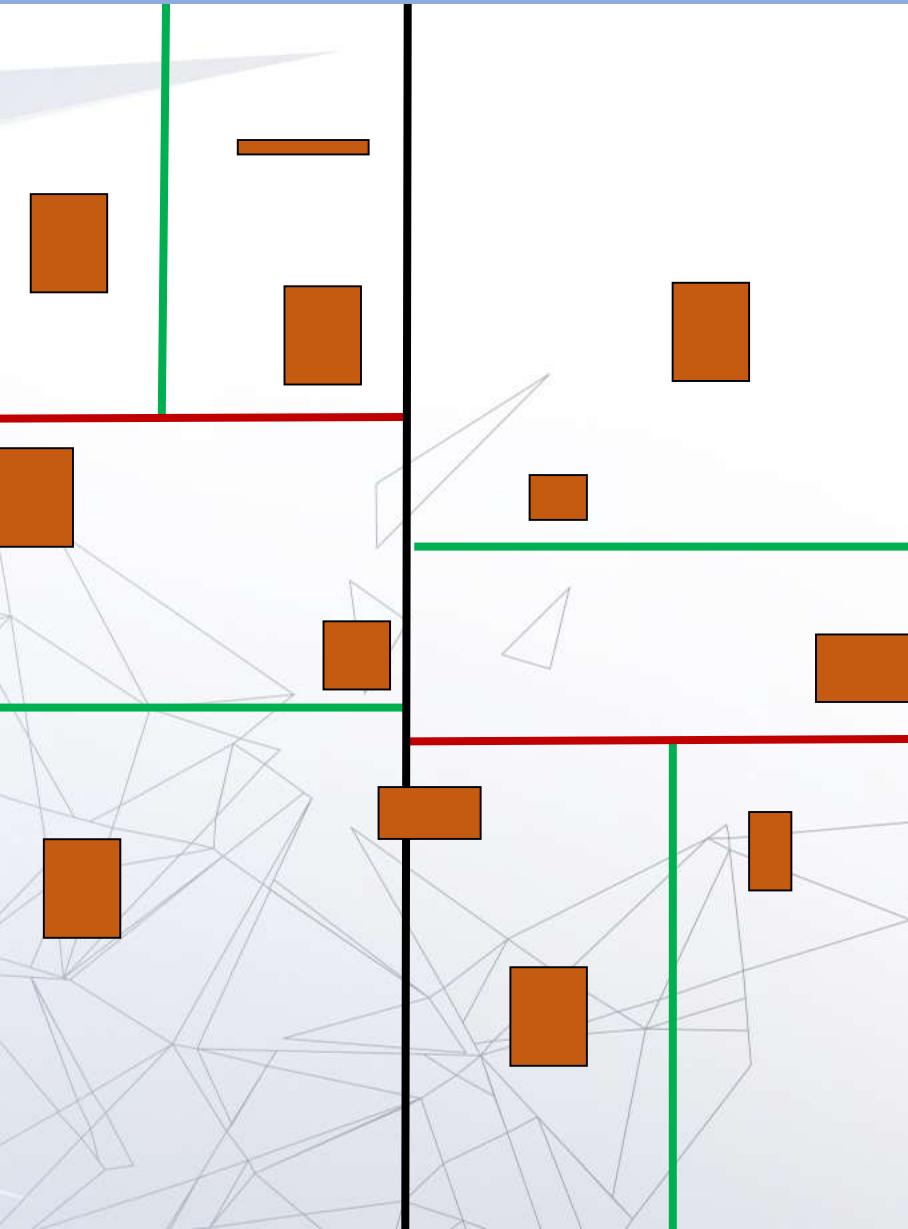


UBISOFT®



La détection des collisions / Broad phase / Les arbres

k-d tree :



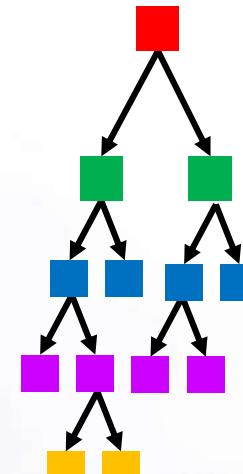
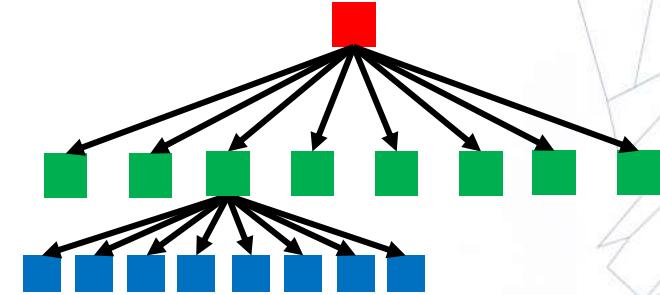
UBISOFT



La détection des collisions / 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

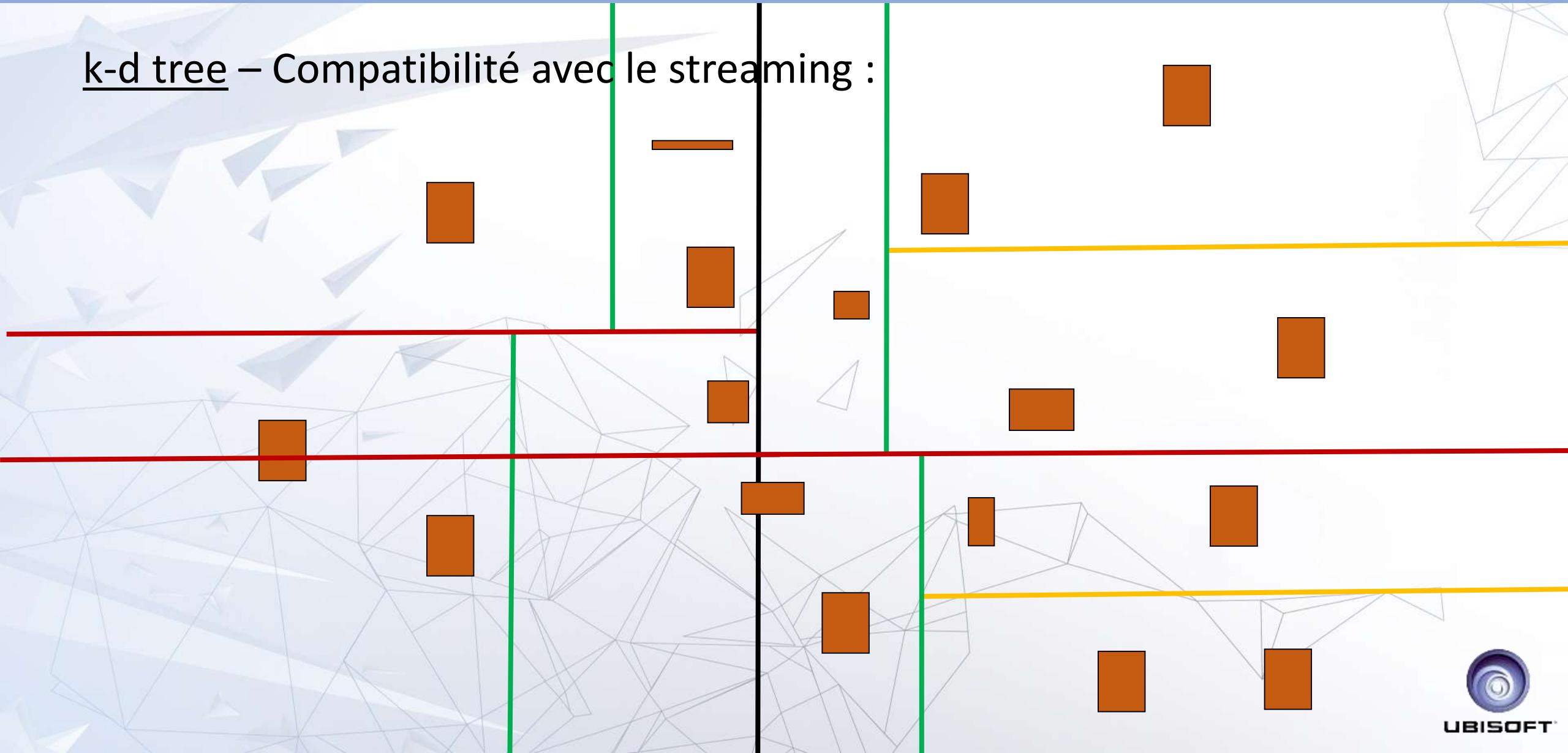


UBISOFT®



La détection des collisions / Broad phase / Les arbres

k-d tree – Compatibilité avec le streaming :



UBISOFT

La détection des collisions / 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~~



UBISOFT

La physique des corps rigides indéformables

Partie 1 : La détection des collisions

1.1 La broad phase

Le filtrage des collisions



UBISOFT®

▶ La détection des collisions / 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

▶ La détection des collisions / Broad phase / Filtrage

Le filtrage des collisions :

Comment ?

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



UBISOFT®

Contenu du cours

Introduction

Alexis Vaisse

Moteurs physiques existants

Que fait un moteur physique ?

La physique des corps rigides indéformables

1. La détection des collisions

1.1 La broad phase

- 1.1.1 Les volumes englobants
- 1.1.2 Algorithme « brute force »
- 1.1.3 L'algorithme Sweep & Prune
- 1.1.4 Les grilles
- 1.1.4 Les arbres
 - Quadtree
 - Octree
 - k-d tree
- 1.1.5 Le filtrage des collisions



UBISOFT

La physique des corps rigides indéformables

Partie 1 : La détection des collisions

1.1 La broad phase



UBISOFT

La physique des corps rigides indéformables

Partie 1 : La détection des collisions

1.1 La broad phase

1.2 La narrow phase



UBISOFT

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

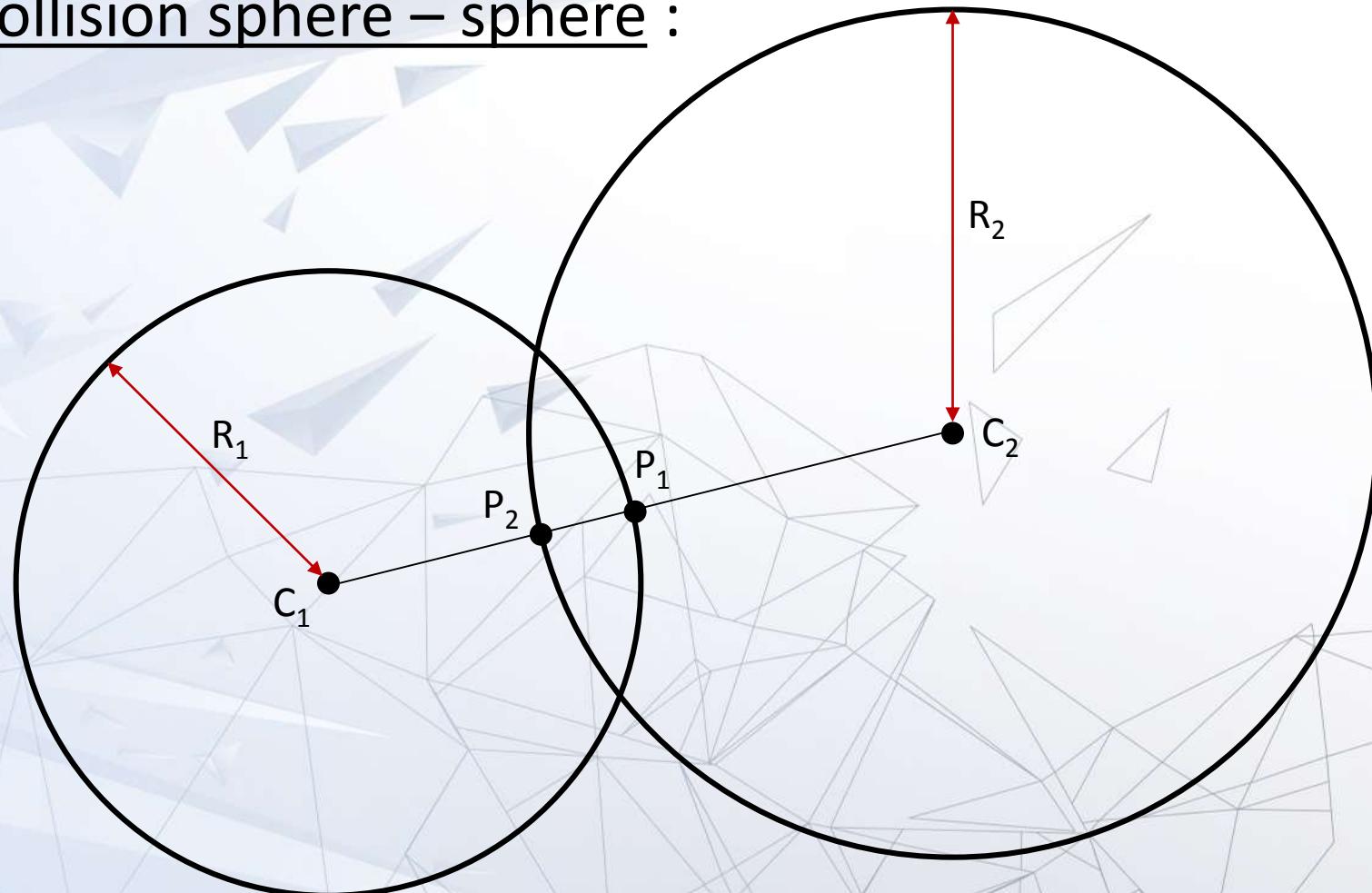


UBISOFT



La détection des collisions / Narrow phase / Sphère

Collision sphère – sphère :



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

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

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

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

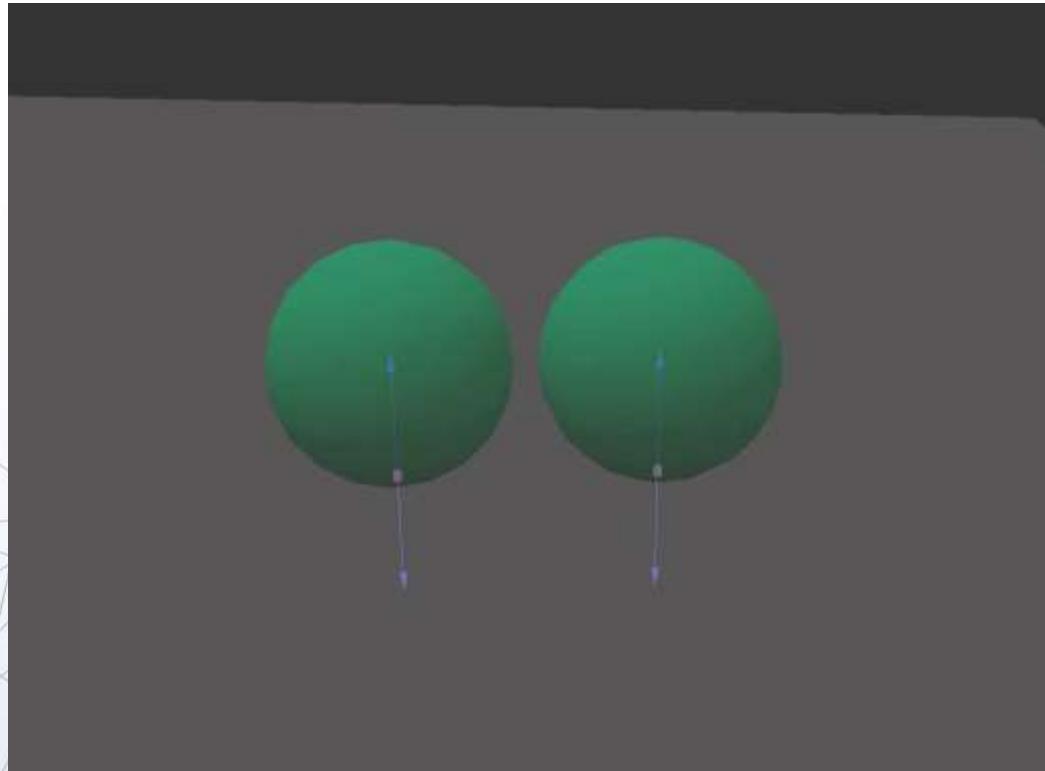
$$d = -P_1P_2$$



UBISOFT

▶ La détection des collisions / Narrow phase / Sphère

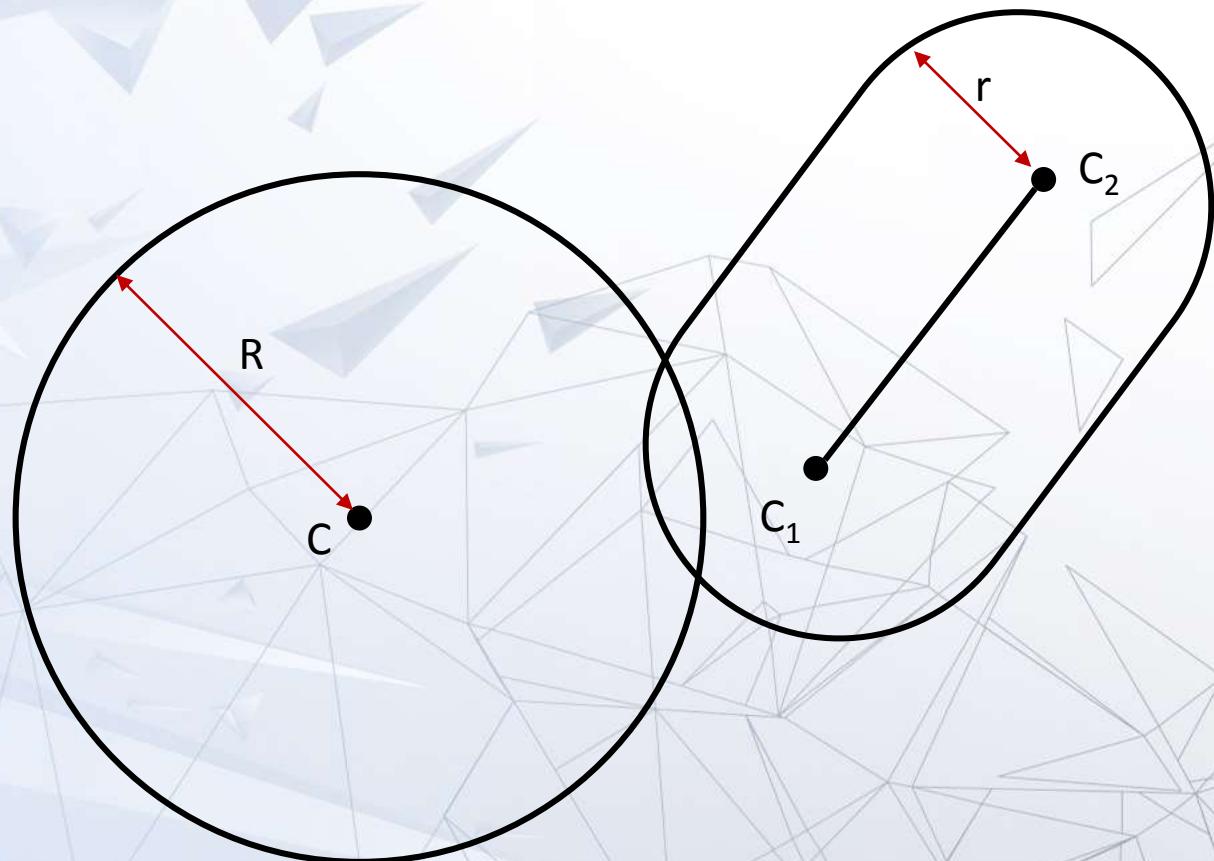
Collision sphère – sphère :



UBISOFT

La détection des collisions / Narrow phase / Capsule

Collision sphère – capsule :



Collision si

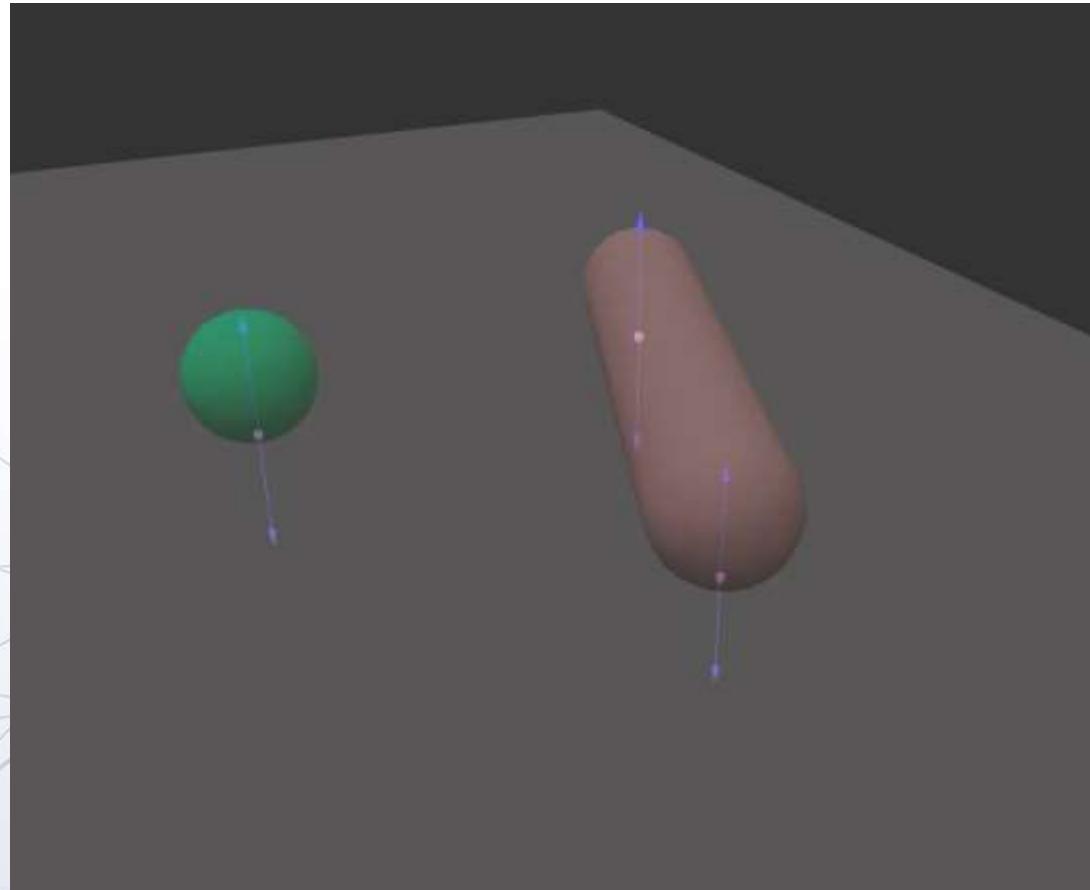
$$d(C, C_1C_2)^2 \leq (R+r)^2$$



UBISOFT

▶ La détection des collisions / Narrow phase / Capsule

Collision sphère – capsule :

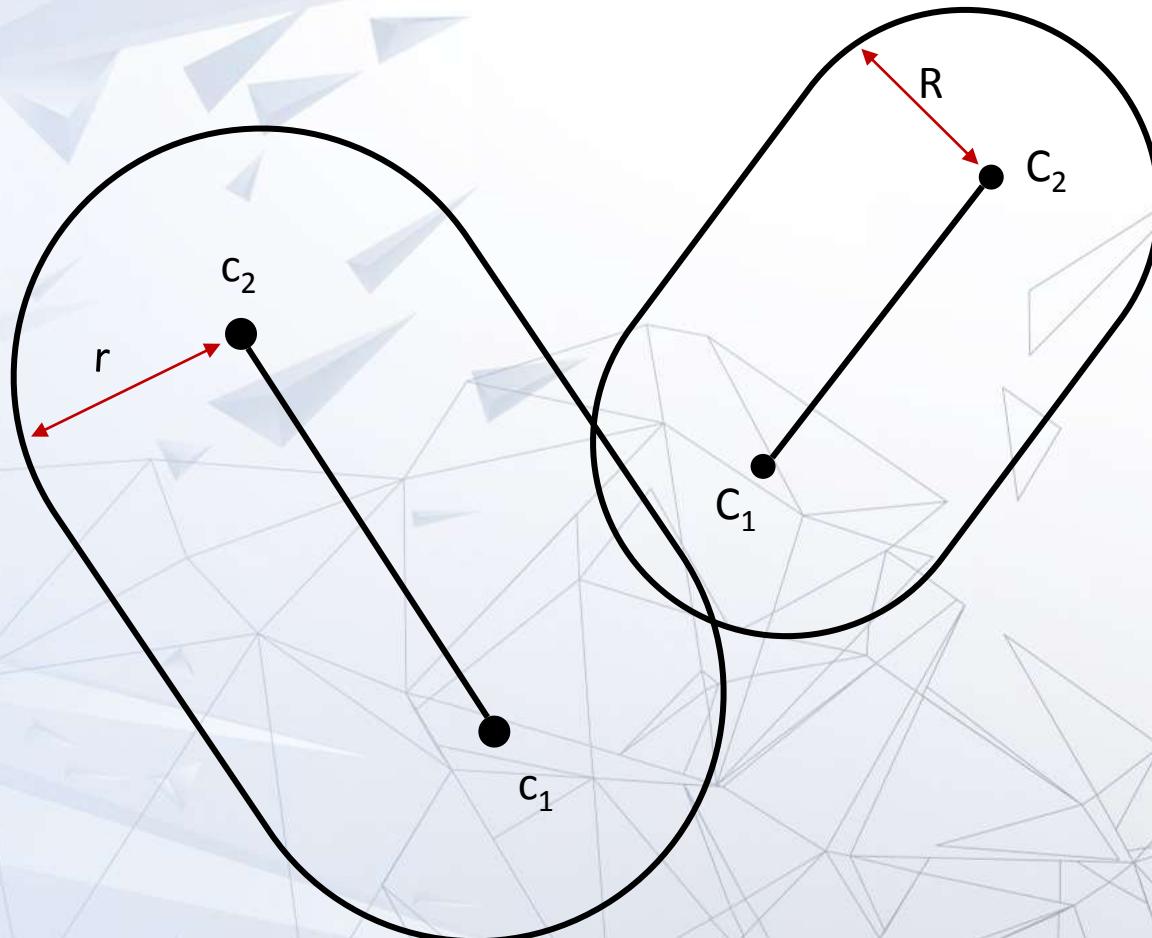


UBISOFT



La détection des collisions / Narrow phase / Capsule

Collision capsule – capsule :



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



UBISOFT



La détection des collisions / Narrow phase / Boite

Collision boite – 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

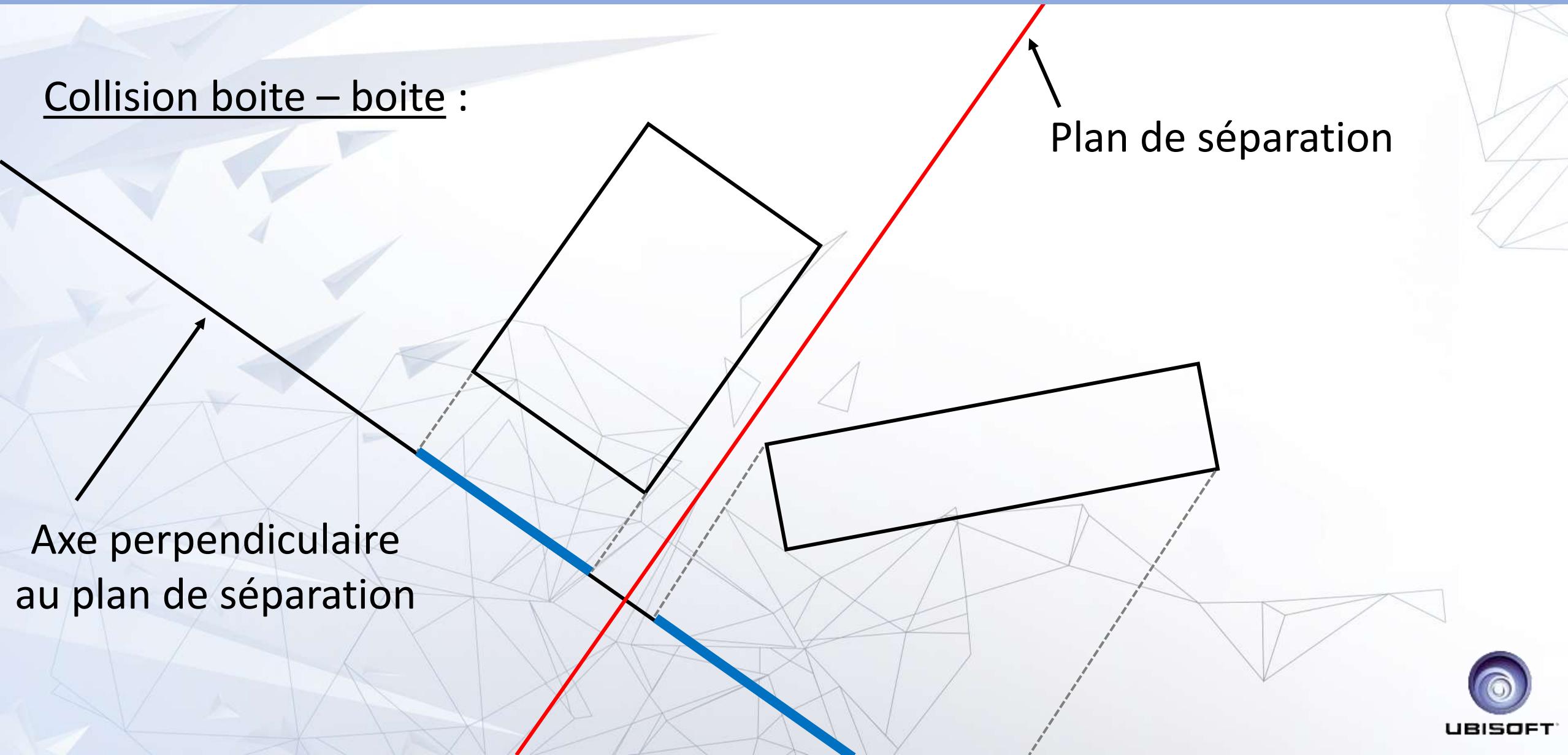


UBISOFT



La détection des collisions / Narrow phase / Boîte

Collision boîte – boîte :



UBISOFT™

La détection des collisions / Narrow phase / Boîte



Si les deux boîtes ne sont pas en collision :

Comment choisir le plan de séparation ?

Si les deux boîtes sont en collision :

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



UBISOFT



La détection des collisions / Narrow phase / 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

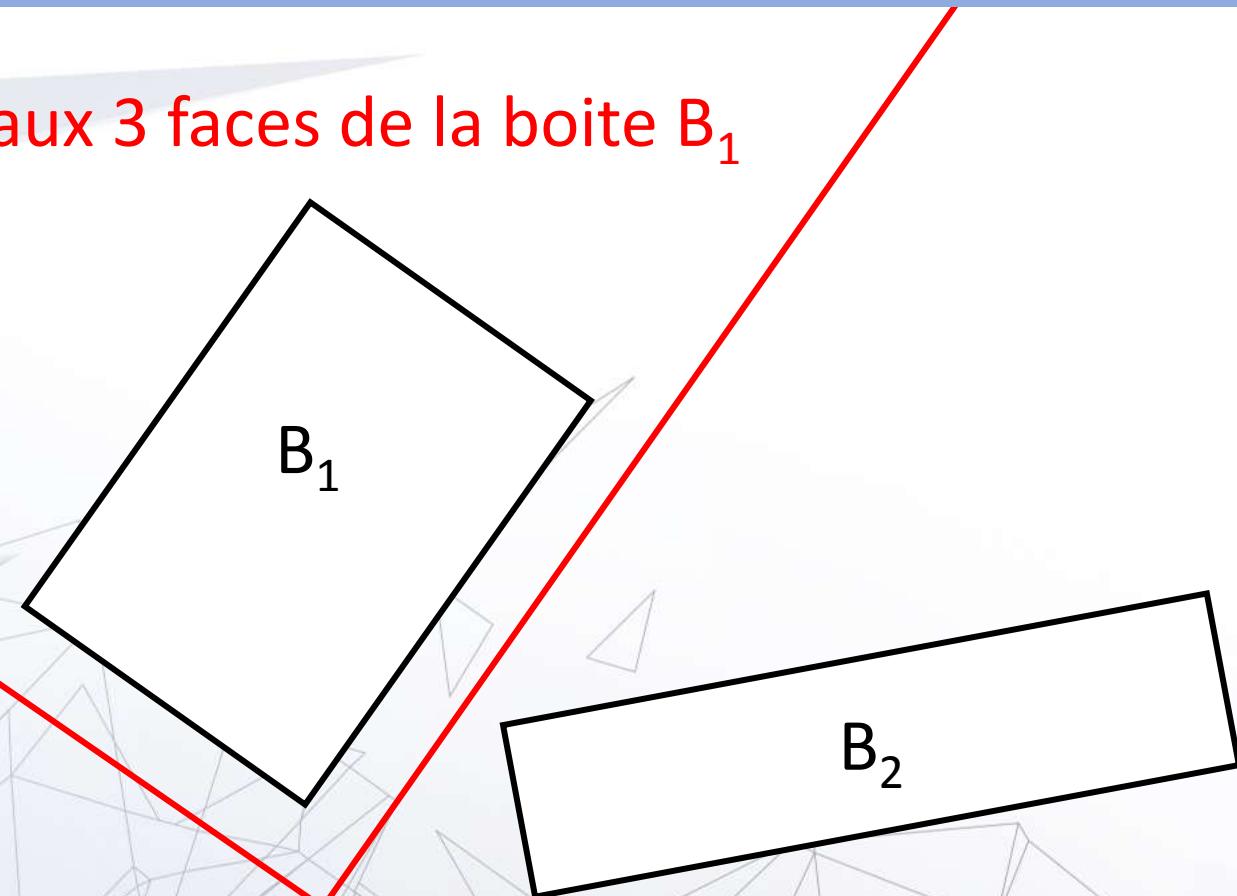
Théorème :

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



La détection des collisions / Narrow phase / Boite

Les 3 plans parallèles aux 3 faces de la boite B_1

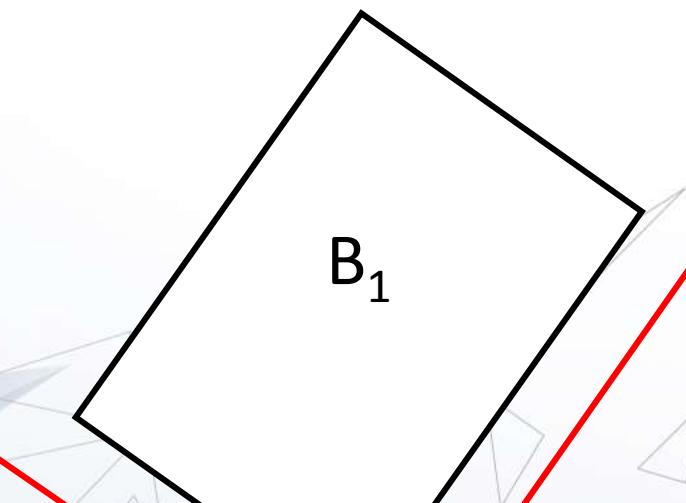


UBISOFT

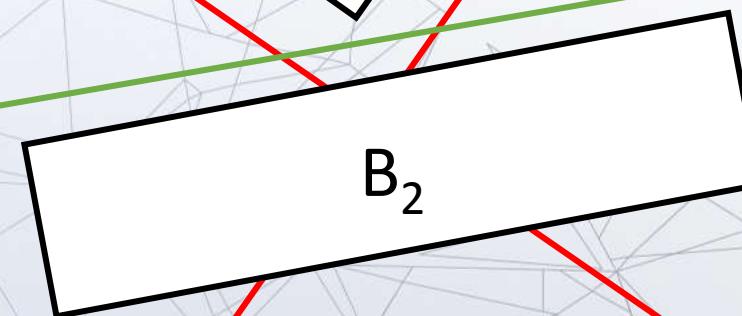


La détection des collisions / Narrow phase / Boîte

Les 3 plans parallèles aux 3 faces de la boite B_1



Les 3 plans parallèles aux 3 faces de la boite B_2



UBISOFT



La détection des collisions / Narrow phase / Boîte

En 2D : 4 droites sont suffisantes

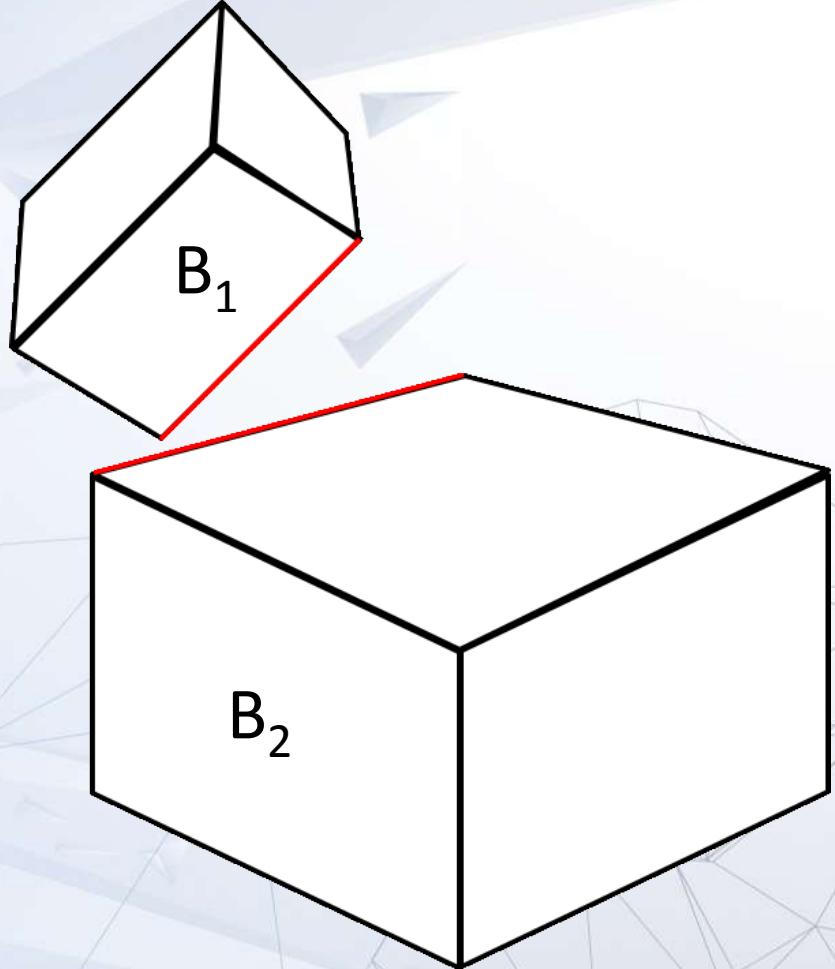
En 3D : Est-ce que 6 plans sont suffisants ?



UBISOFT



La détection des collisions / Narrow phase / Boîte



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 boîtes sont en intersection





La détection des collisions / Narrow phase / Convexe

Collision convexe – convexe :

Théorème de séparation :

convexes

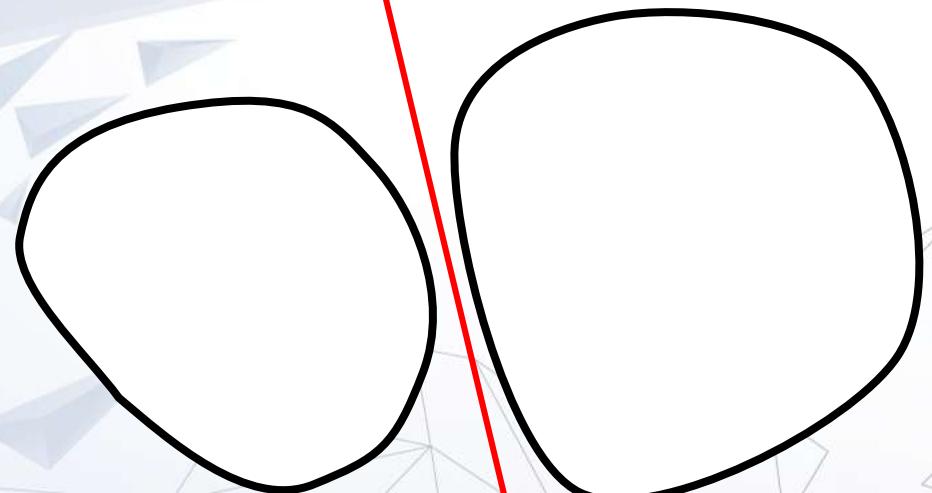
Si deux ~~boites~~ ne sont pas en intersection, il est possible de trouver un plan qui sépare l'espace en deux demi-espaces de sorte que chacun contienne entièrement l'un ~~X~~ des ~~boites~~ convexes



UBISOFT



La détection des collisions / Narrow phase / Convexe



?



UBISOFT

La détection des collisions / 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 :

$$f_1 + f_2 + \cancel{a_1 \times a_2}$$

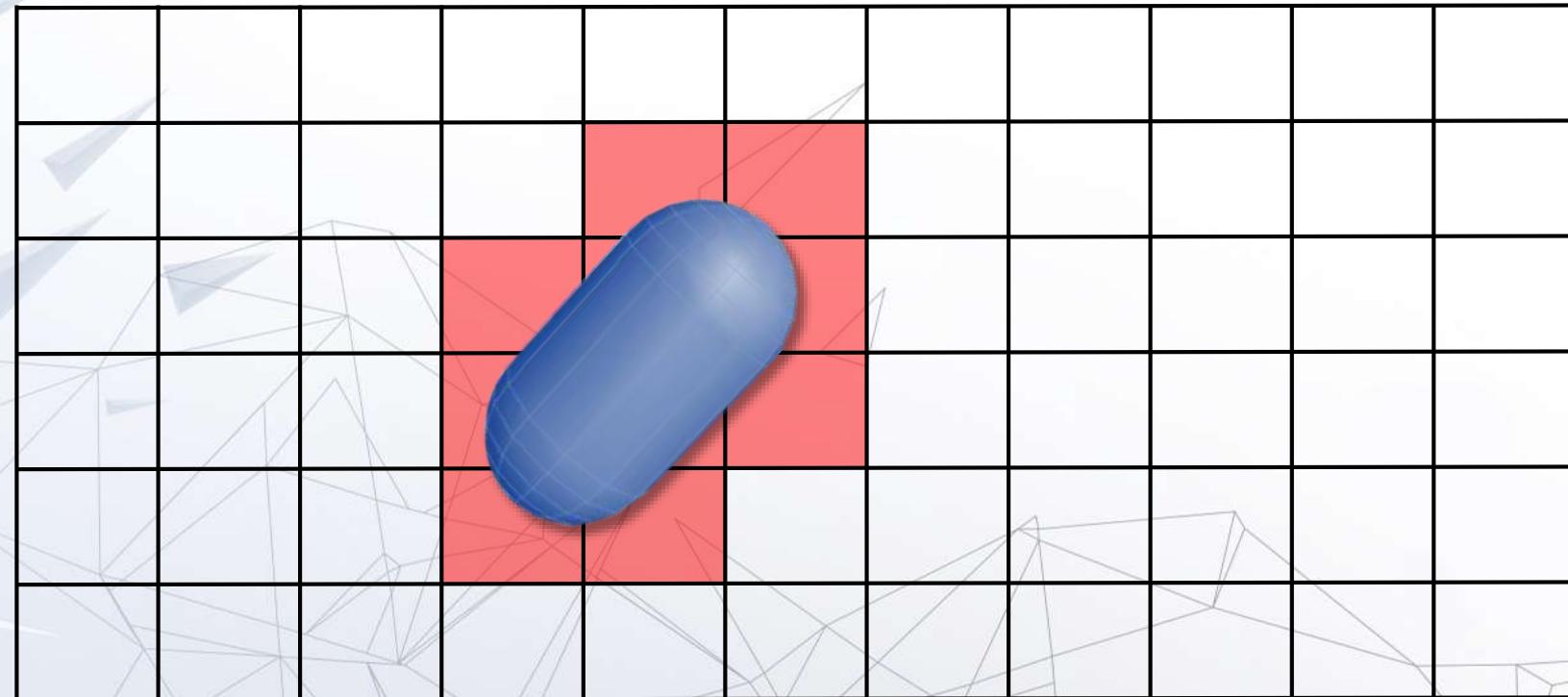
$\underbrace{}_{O(n)} \quad \underbrace{\phantom{\cancel{a_1 \times a_2}}}_{O(n^2)}$



UBISOFT

La détection des collisions / Narrow phase / Heightfield

Collision avec un heightfield :



UBISOFT

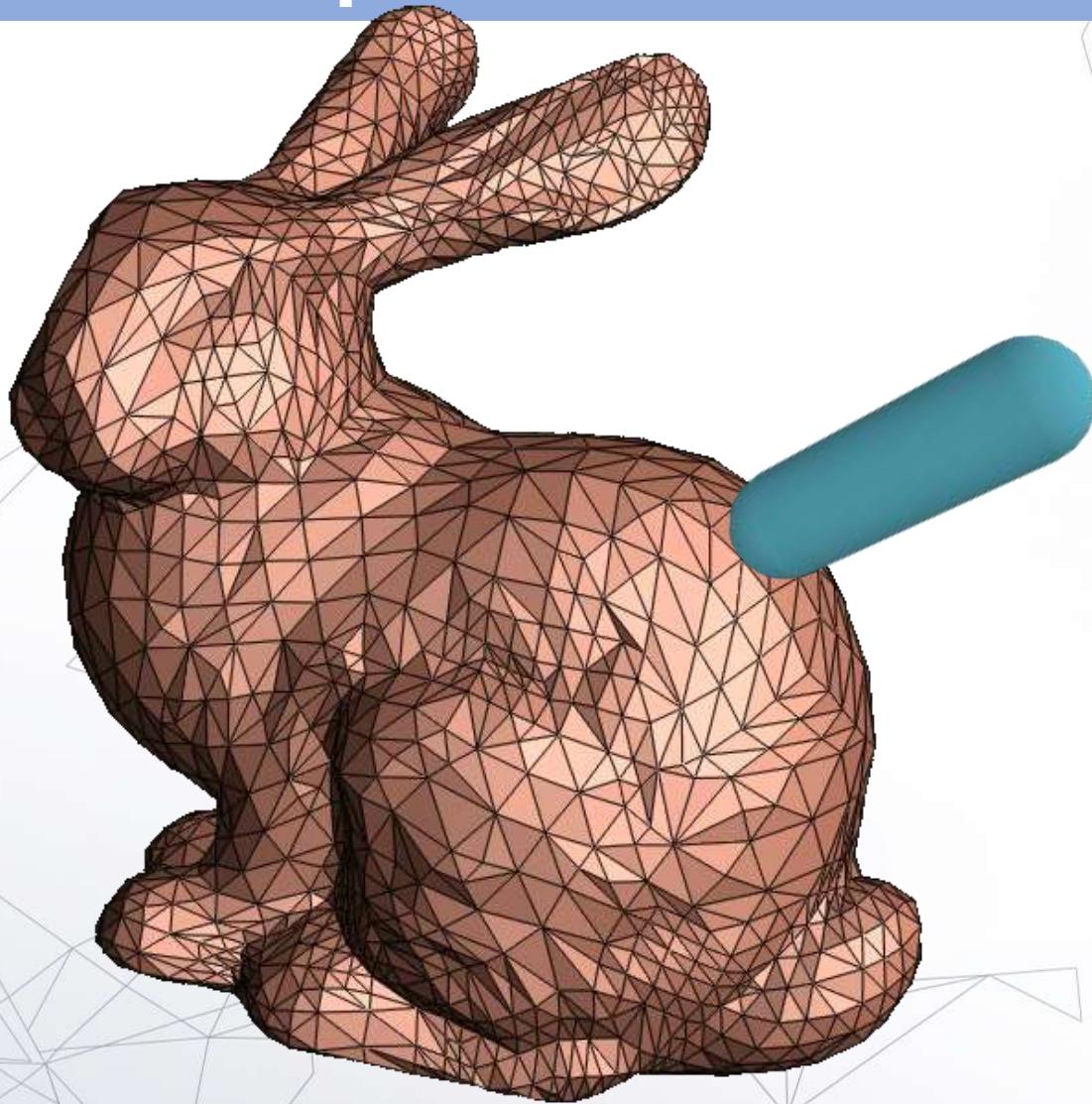


La détection des collisions / Narrow phase / Mesh

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

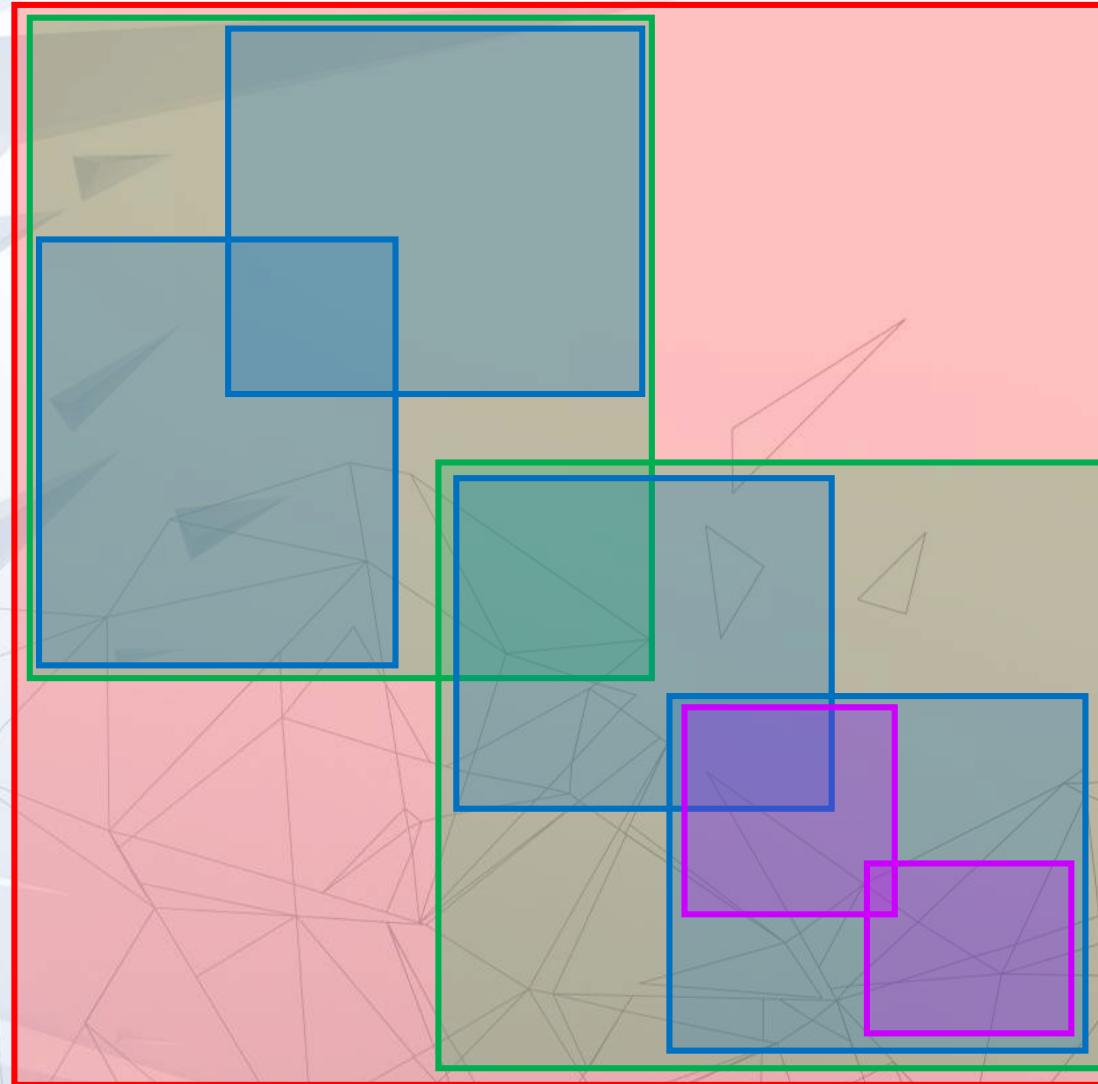


UBISOFT



La détection des collisions / Narrow phase / Mesh

AABB tree :



Avantages :

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

Contenu du cours

Introduction

Alexis Vaisse

Moteurs physiques existants

Que fait un moteur physique ?

La physique des corps rigides indéformables

1. La détection des collisions

1.1 La broad phase

- 1.1.1 Les volumes englobants
- 1.1.2 Algorithme « brute force »
- 1.1.3 L'algorithme Sweep & Prune
- 1.1.4 Les grilles
- 1.1.4 Les arbres
 - Quadtree
 - Octree
 - k-d tree
- 1.1.5 Le filtrage des collisions

1.2 La narrow phase

- 1.2.1 Collision sphère – sphère
- 1.2.2 Collision sphère – capsule
- 1.2.3 Collision capsule – capsule
- 1.2.4 Collision boite – boite
- 1.2.5 Collision convexe – convexe
- 1.2.6 Collision avec un heightfield
- 1.2.7 Collision avec un mesh

La physique des corps rigides indéformables

Partie 1 : La détection des collisions

1.1 La broad phase

1.2 La narrow phase



UBISOFT®

La physique des corps rigides indéformables

Partie 1 : La détection des collisions

1.1 La broad phase

1.2 La narrow phase

Partie 2 : La résolution des contraintes



UBISOFT®

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



UBISOFT®

La physique des corps rigides indéformables

Partie 2 : La résolution des contraintes

2.1 Les joints

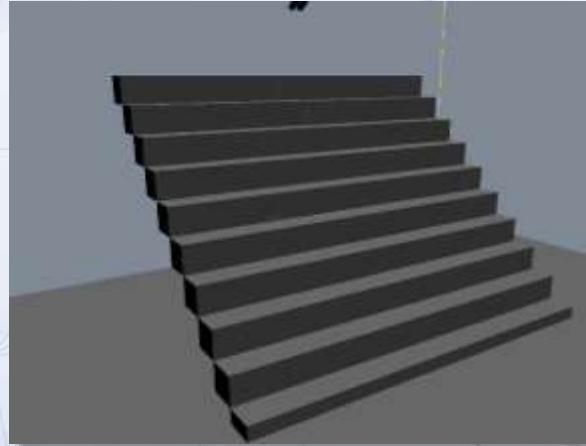


UBISOFT

La résolution des contraintes / Les joints

Que peut-on faire avec des joints ?

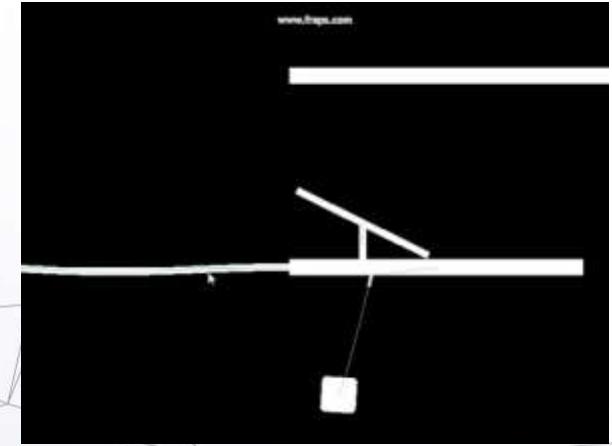
Ragdoll



Amortisseur



Corde



La résolution des contraintes / Les joints

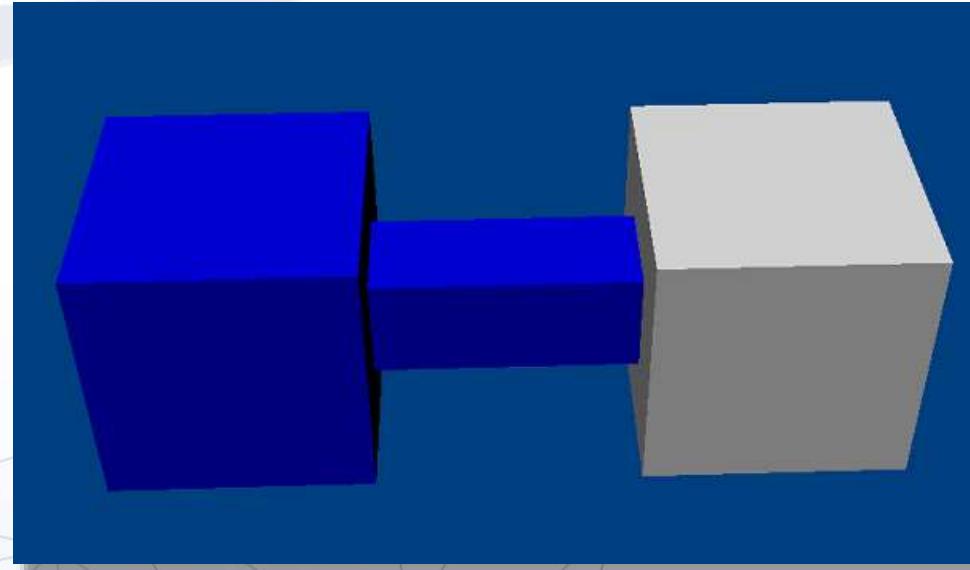
- Un joint est une liaison entre 2 corps rigides indéformables
- Il existe de nombreux types de joints différents



UBISOFT

▶ La résolution des contraintes / Les joints / Joint fixe

Joint fixe :



Paramètres :

- Identifiant du corps A
- Identifiant du corps B
- Position et orientation relative

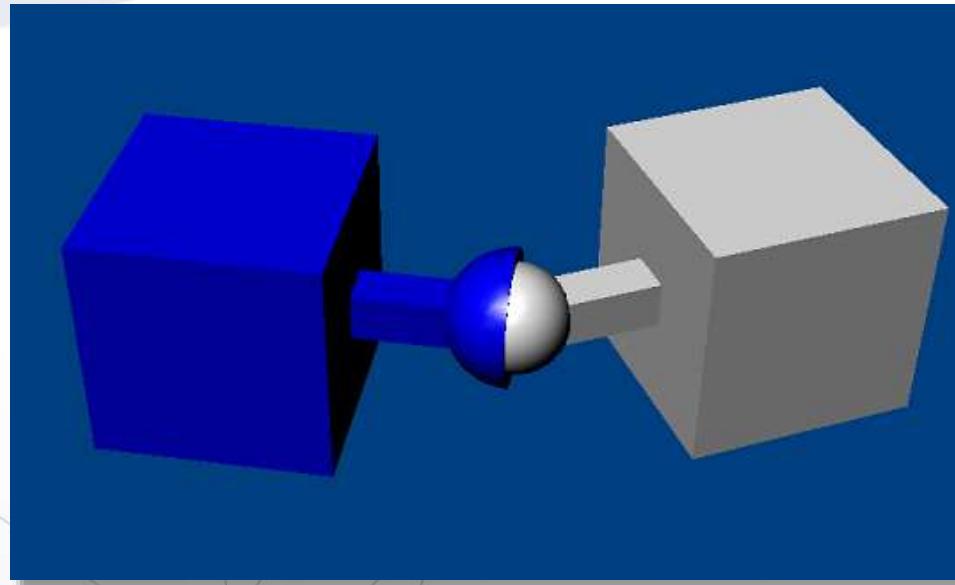
Intérêts ?

- Joint cassable
- Connaître la force exercée sur le joint

La résolution des contraintes / Les joints / Joint sphérique

Joint sphérique:

Joint point-to-point :



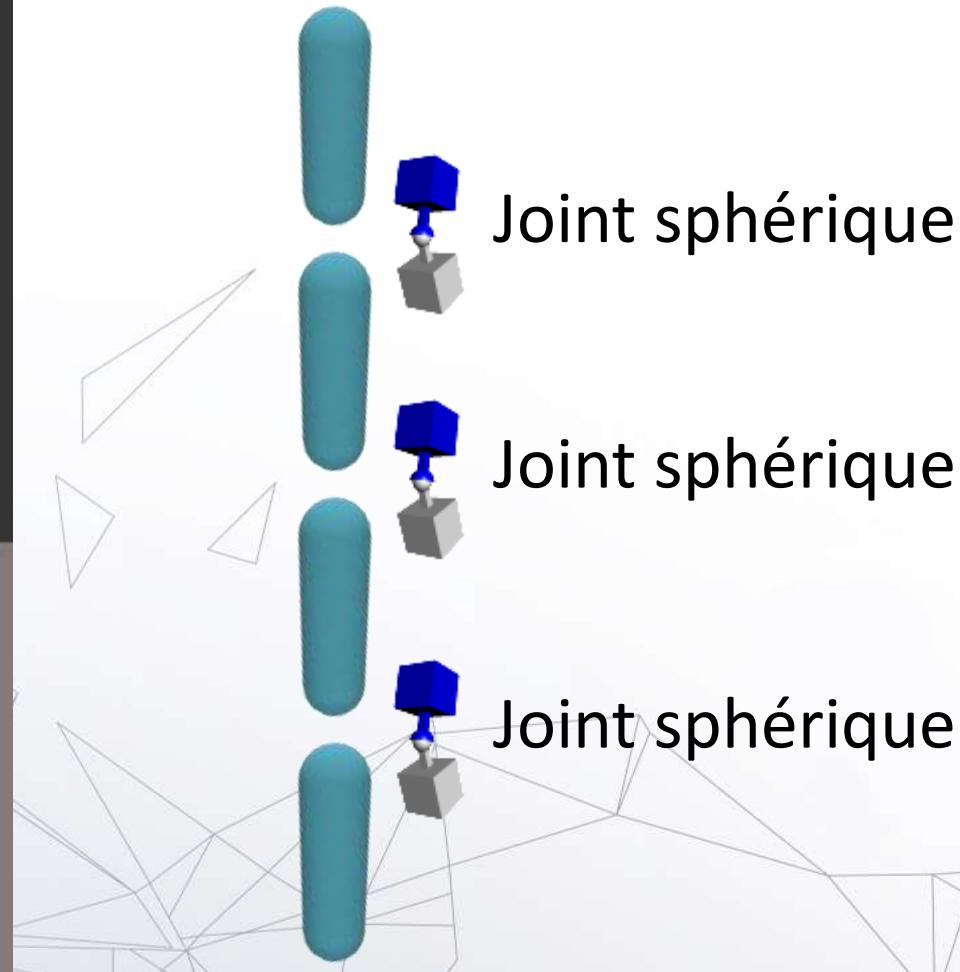
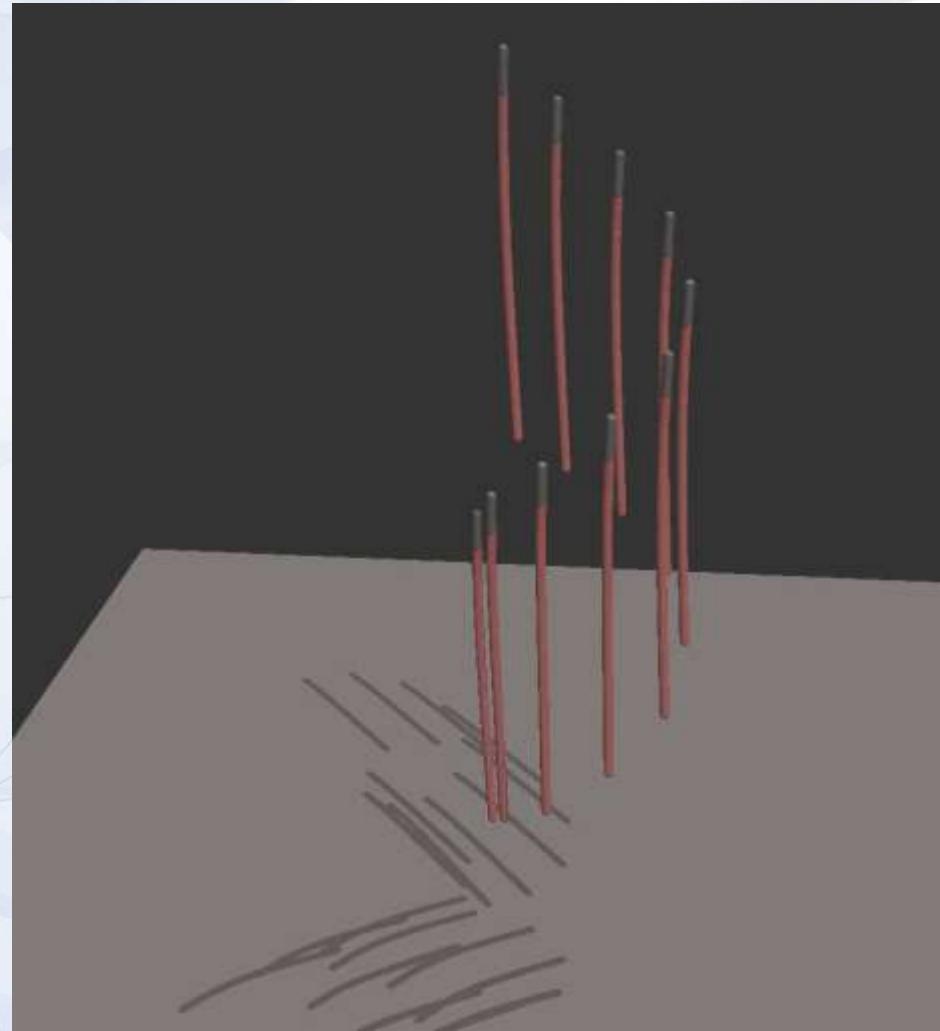
Paramètres :

- Identifiant du corps A
- Identifiant du corps B
- Point dans le repère de A
- Point dans le repère de B



UBISOFT

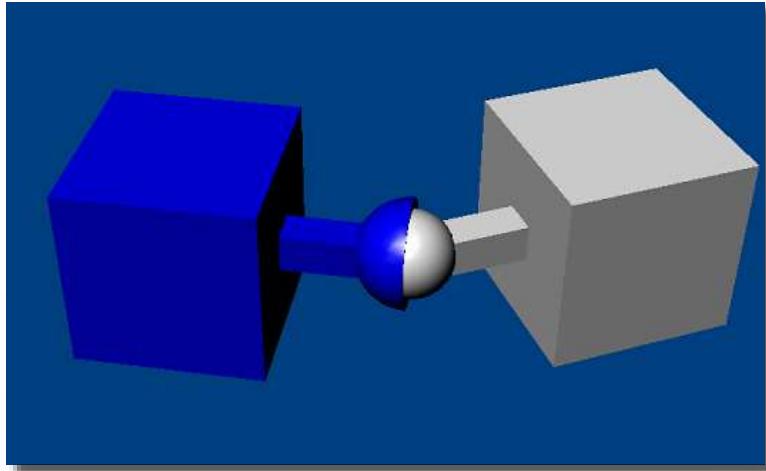
▶ La résolution des contraintes / Les joints / Joint sphérique



La résolution des contraintes / Les joints / Joint sphérique

Joint sphérique:

Joint point-to-point :



Cône à base circulaire

Paramètres :

- Identifiant du corps A
- Identifiant du corps B
- Point dans le repère de A
- Point dans le repère de B
- Axe
- Demi-angle du cône



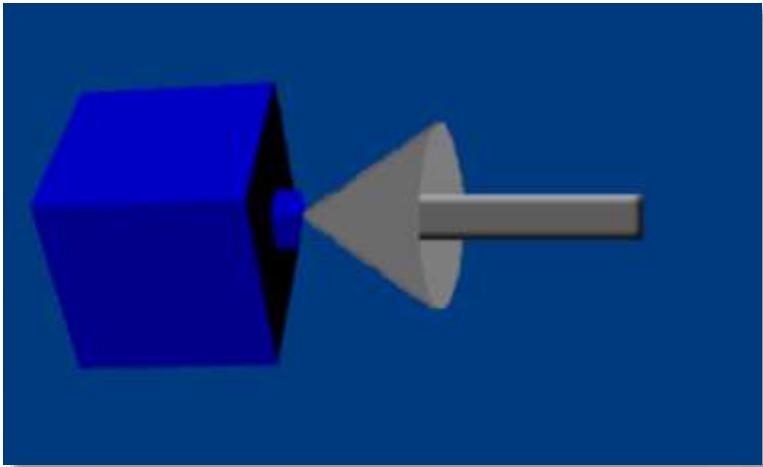
UBISOFT



La résolution des contraintes / Les joints / Joint sphérique

Joint sphérique:

Joint point-to-point :



Cône à base circulaire ellipsoïdale

Paramètres :

- Identifiant du corps A
- Identifiant du corps B
- Point dans le repère de A
- Point dans le repère de B
- 2 axes
- 1^{er} demi-angle du cône
- 2^{ème} demi-angle du cône

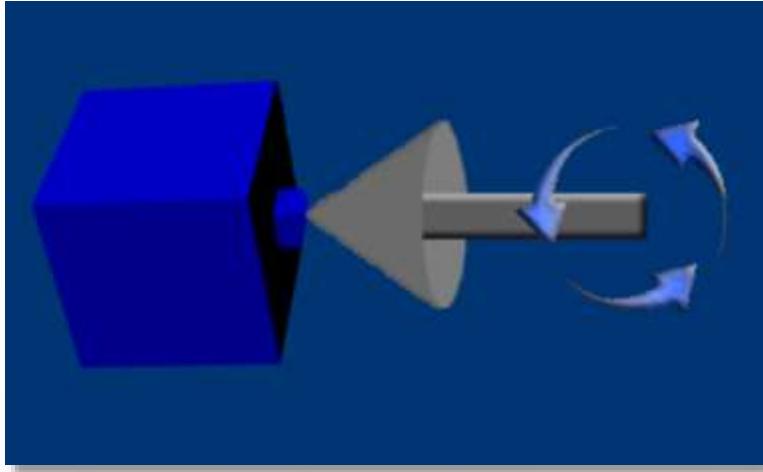


UBISOFT

▶ La résolution des contraintes / Les joints / Joint ragdoll

Joint sphérique:

Joint ragdoll :



Paramètres :

- Identifiant du corps A
- Identifiant du corps B
- Point dans le repère de A
- Point dans le repère de B

- 2 axes
- 1^{er} demi-angle du cône
- 2^{ème} demi-angle du cône
- Angle de twist maximum

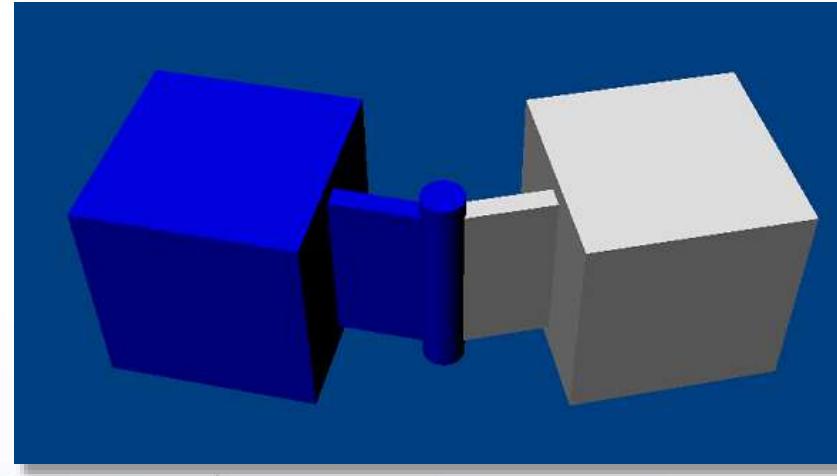


UBISOFT

La résolution des contraintes / Les joints / Joint charnière

Joint de révolution :

Joint charnière :



Utilisation ?

- Porte
- Coude
- Genou



▶ La résolution des contraintes / Les joints / Joint prismatique

Joint prismatique :



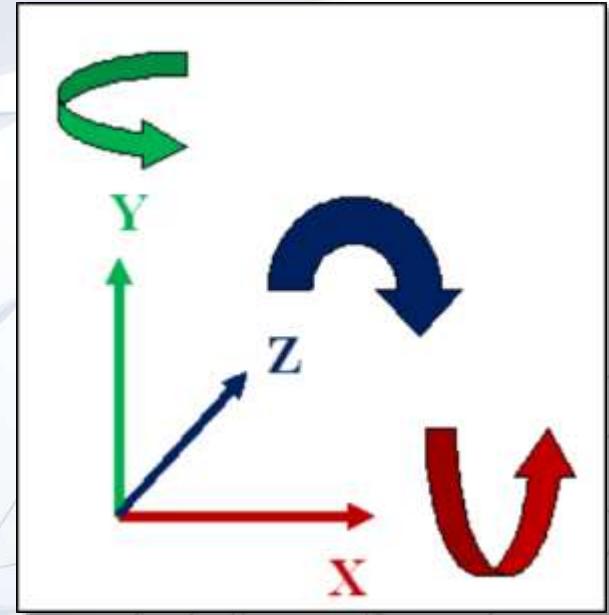
Utilisation ? • Amortisseur



UBISOFT®

▶ La résolution des contraintes / Les joints / Joint 6 dof

Joint 6 dof :



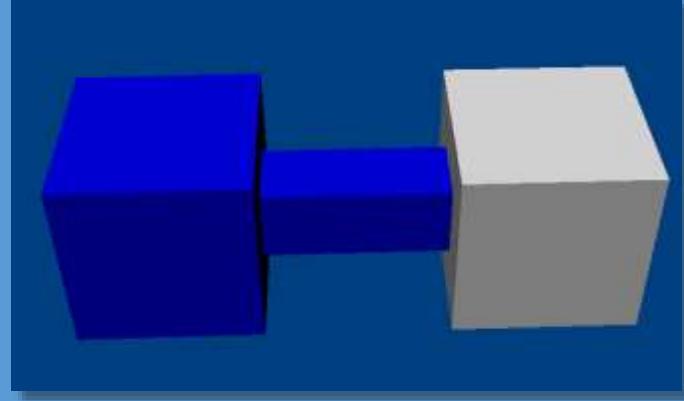
Chaque degré de liberté peut être:

- Libre
- Bloqué
- Limité

- Réduit ou supprime certains degrés de liberté (entre 1 et 6)
- Permet de "tout" faire

La résolution des contraintes / Les joints / Joint 6 dof

Joint fixe



Translation

Bloqué
Bloqué
Bloqué

Rotation

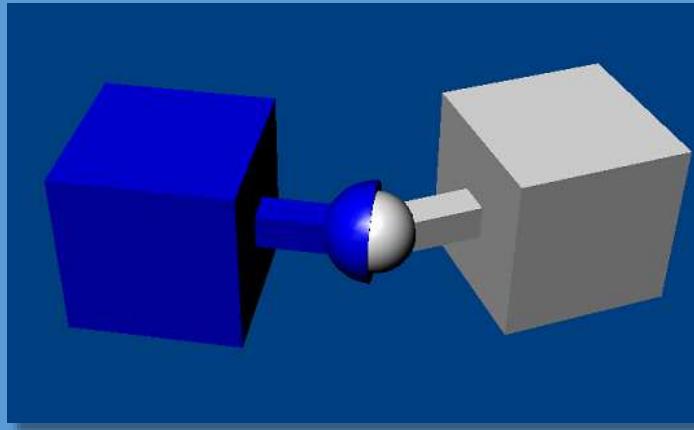
Bloqué
Bloqué
Bloqué



UBISOFT

La résolution des contraintes / Les joints / Joint 6 dof

Joint sphérique



Translation

Bloqué
Bloqué
Bloqué

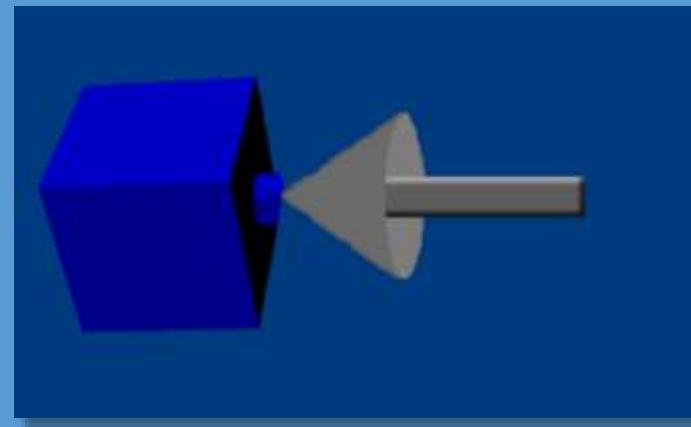
Rotation

Libre
Libre
Libre



La résolution des contraintes / Les joints / Joint 6 dof

Joint sphérique



Translation

Bloqué
Bloqué
Bloqué

Rotation

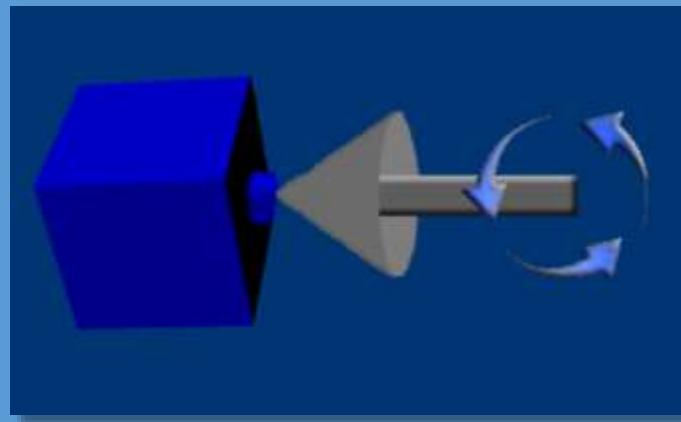
Libre
~~Libre~~ Limité
~~Libre~~ Limité



UBISOFT

La résolution des contraintes / Les joints / Joint 6 dof

Joint ragdoll



Translation

Bloqué
Bloqué
Bloqué

Rotation

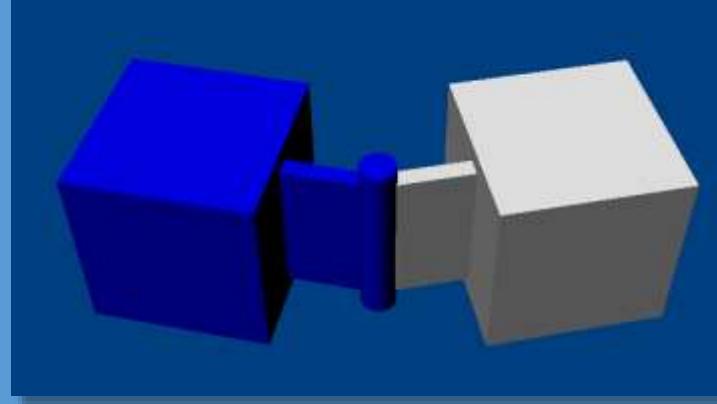
~~Libre~~ Limité
~~Libre~~ Limité
~~Libre~~ Limité



UBISOFT™

La résolution des contraintes / Les joints / Joint 6 dof

Joint charnière



Translation

Bloqué
Bloqué
Bloqué

Rotation

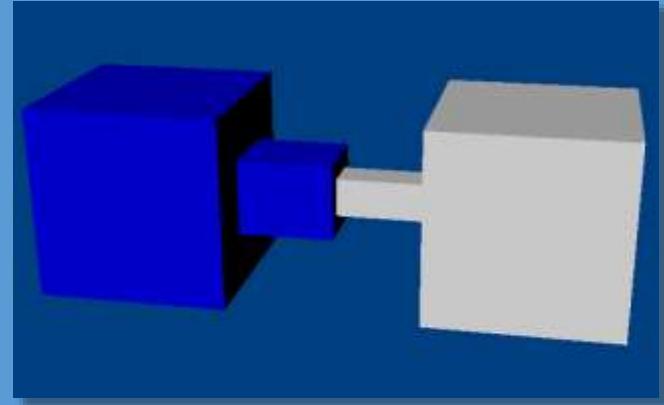
Libre ou limité
Bloqué
Bloqué



UBISOFT

La résolution des contraintes / Les joints / Joint 6 dof

Joint prismatique



Translation

Libre ou limité
Bloqué
Bloqué

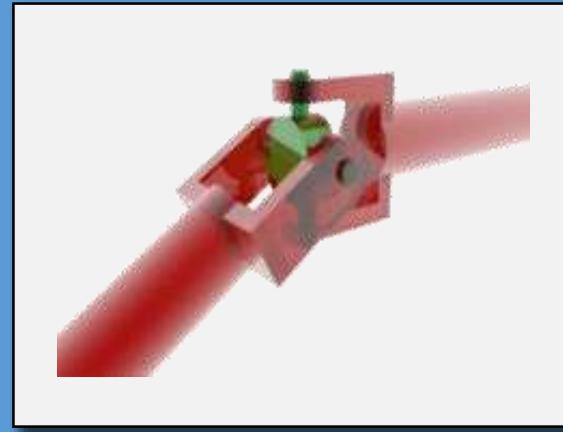
Rotation

Bloqué
Bloqué
Bloqué



La résolution des contraintes / Les joints / Joint 6 dof

Joint universel



Translation

Bloqué
Bloqué
Bloqué

Rotation

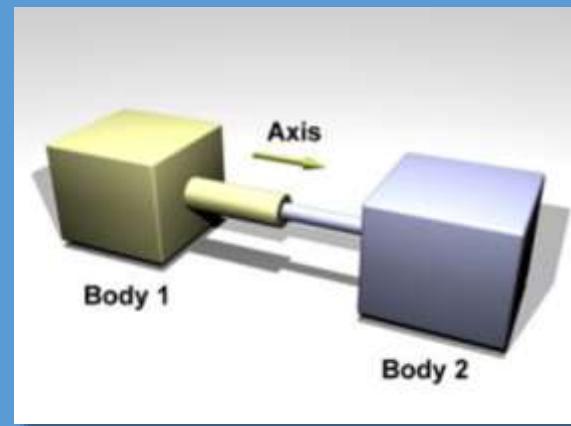
Bloqué
Libre ou limité
Libre ou limité



UBISOFT

La résolution des contraintes / Les joints / Joint 6 dof

Joint cylindrique



Translation

Libre ou limité
Bloqué
Bloqué

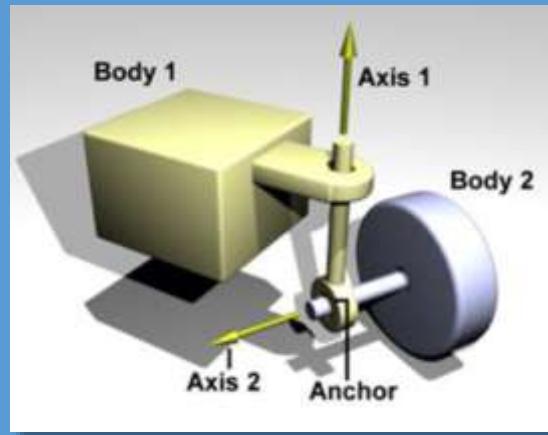
Rotation

Libre ou limité
Bloqué
Bloqué



La résolution des contraintes / Les joints / Joint 6 dof

Joint prismatique universel



Translation

Bloqué
Libre ou limité
Libre ou limité

Rotation

Bloqué
Libre ou limité
Libre ou limité



UBISOFT™

La résolution des contraintes / Les joints / Joint 6 dof

Déplacement contraint dans un plan
?

Translation

Libre
Libre
Bloqué

Rotation

Libre
Libre
Libre



UBISOFT

La résolution des contraintes / Les joints / Joint 6 dof

Déplacement contraint dans un rectangle
?

Translation

Limité
Limité
Bloqué

Rotation

Libre
Libre
Libre



UBISOFT

La résolution des contraintes / Les joints / Joint 6 dof

Permet d'avoir une capsule
qui reste droite



Translation

Libre
Libre
Libre

Rotation

Bloqué
Bloqué
Libre

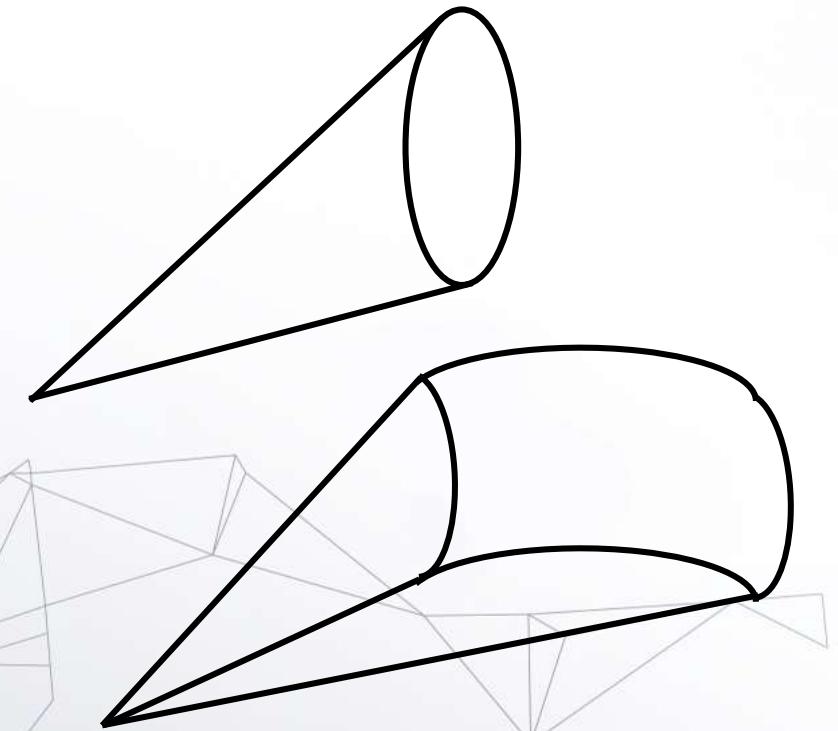


UBISOFT

La résolution des contraintes / Les joints / Joint 6 dof

Pourquoi ne pas proposer que des joints 6 dof ?

- Facilité d'utilisation
- Performance
- Stabilité
- Pas exactement les mêmes limites



UBISOFT

La physique des corps rigides indéformables

Partie 2 : La résolution des contraintes

2.1 Les joints

Les joints cassables



UBISOFT



La résolution des contraintes / Les joints cassables

- Un joint cassable est défini par une force et un couple maximum
- Quand la force ou le couple exercés sur le joint dépassent ces limites, le joint casse
- Le code utilisateur est généralement prévenu par un appel de callback



UBISOFT®

▶ La résolution des contraintes / Les joints cassables



UBISOFT®

La physique des corps rigides indéformables

Partie 2 : La résolution des contraintes

2.1 Les joints

Les joints motorisés



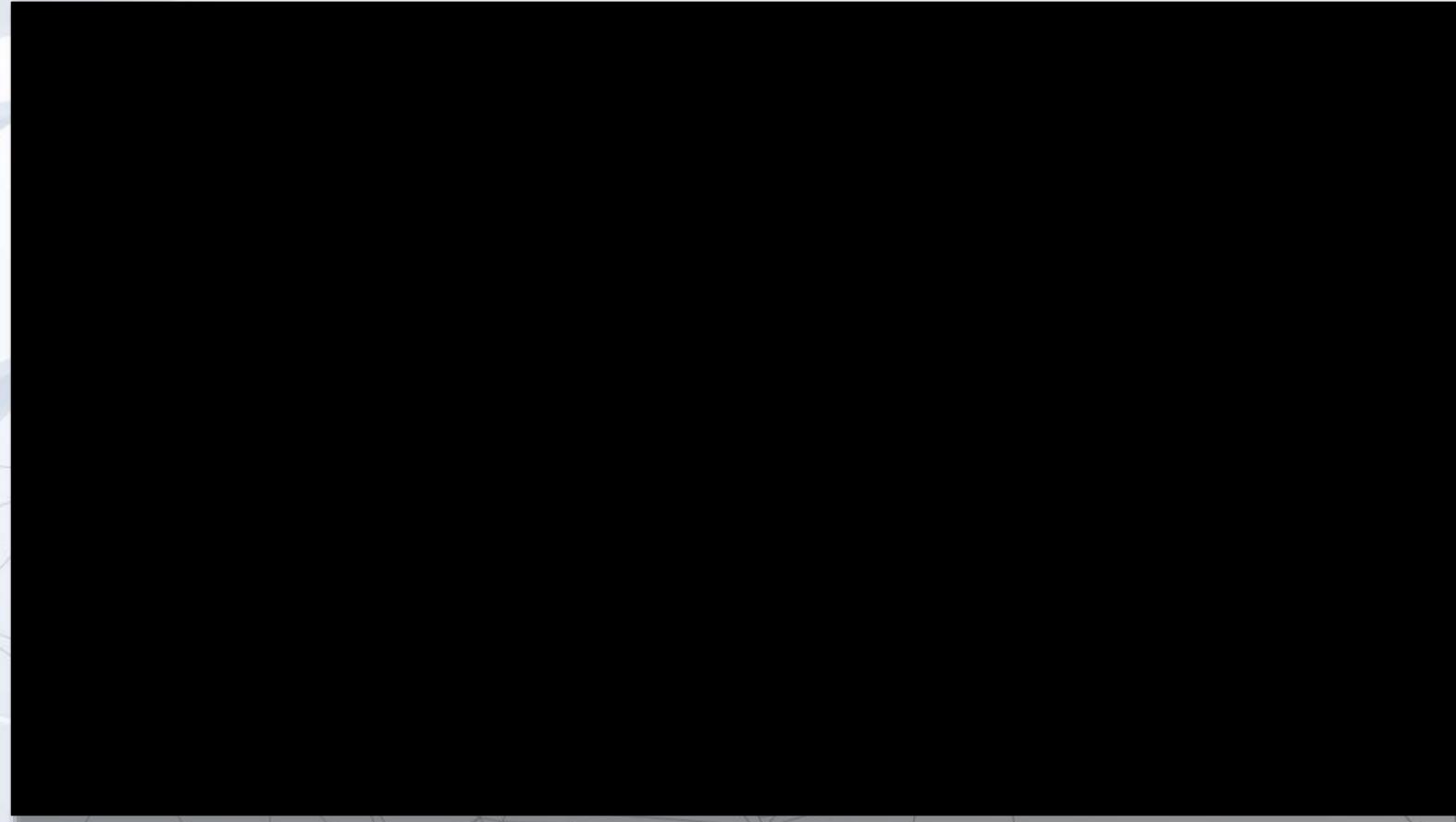
UBISOFT

La résolution des contraintes / Les joints motorisés

Principe des joints motorisés :

- On imagine un moteur au niveau du joint
- A chaque trame, on donne un objectif au moteur
(orientation : matrice 3x3 ou quaternion)
- Le moteur possède | une vitesse angulaire maximale
| un couple maximal

▶ La résolution des contraintes / Les joints motorisés



UBISOFT®

▶ La résolution des contraintes / Les joints motorisés

Joints motorisés – Exemples d'utilisation :

- Passer en mode « ragdoll » mais arriver à une pose prédéterminée
- Jouer une animation par l'intermédiaire d'un moteur physique



UBISOFT®

La physique des corps rigides indéformables

Partie 2 : La résolution des contraintes

2.1 Les joints



UBISOFT

La physique des corps rigides indéformables

Partie 2 : La résolution des contraintes

2.1 Les joints

2.2 Le solveur



UBISOFT

La résolution des contraintes / Le solveur

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



UBISOFT®

La résolution des contraintes / Le solveur

- Le « cerveau » d'un moteur physique
- La partie la plus difficile et la plus complexe techniquement



UBISOFT®

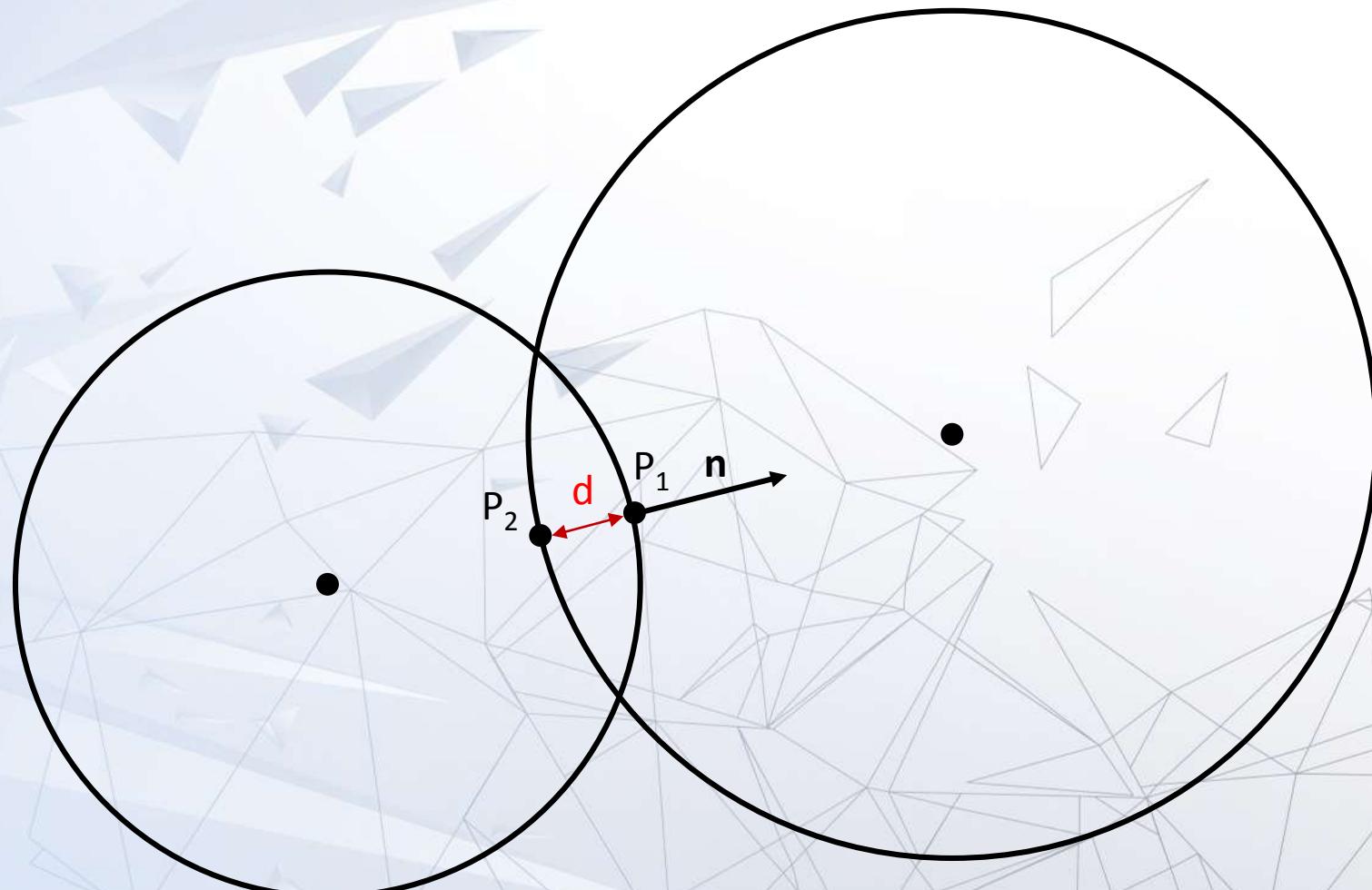
La résolution des contraintes / Le solveur

Chaque contrainte se traduit par une ou plusieurs équations ou inéquations :

- 1 degré de liberté bloqué  1 équation
- 1 degré de liberté limité  2 inéquations
- 1 point de contact  1 inéquation

La résolution des contraintes / Le solveur

Informations fournies par la narrow phase :



- Points de contact : P_1 et P_2
- Normale : n
- Distance de pénétration : d (< 0)

Inéquation pour le solveur :

$$d \geq 0$$

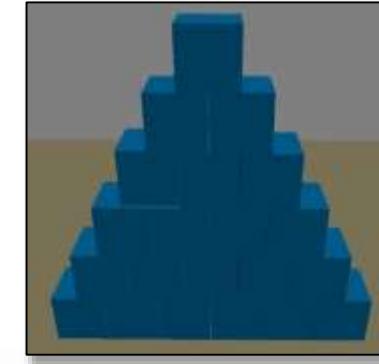
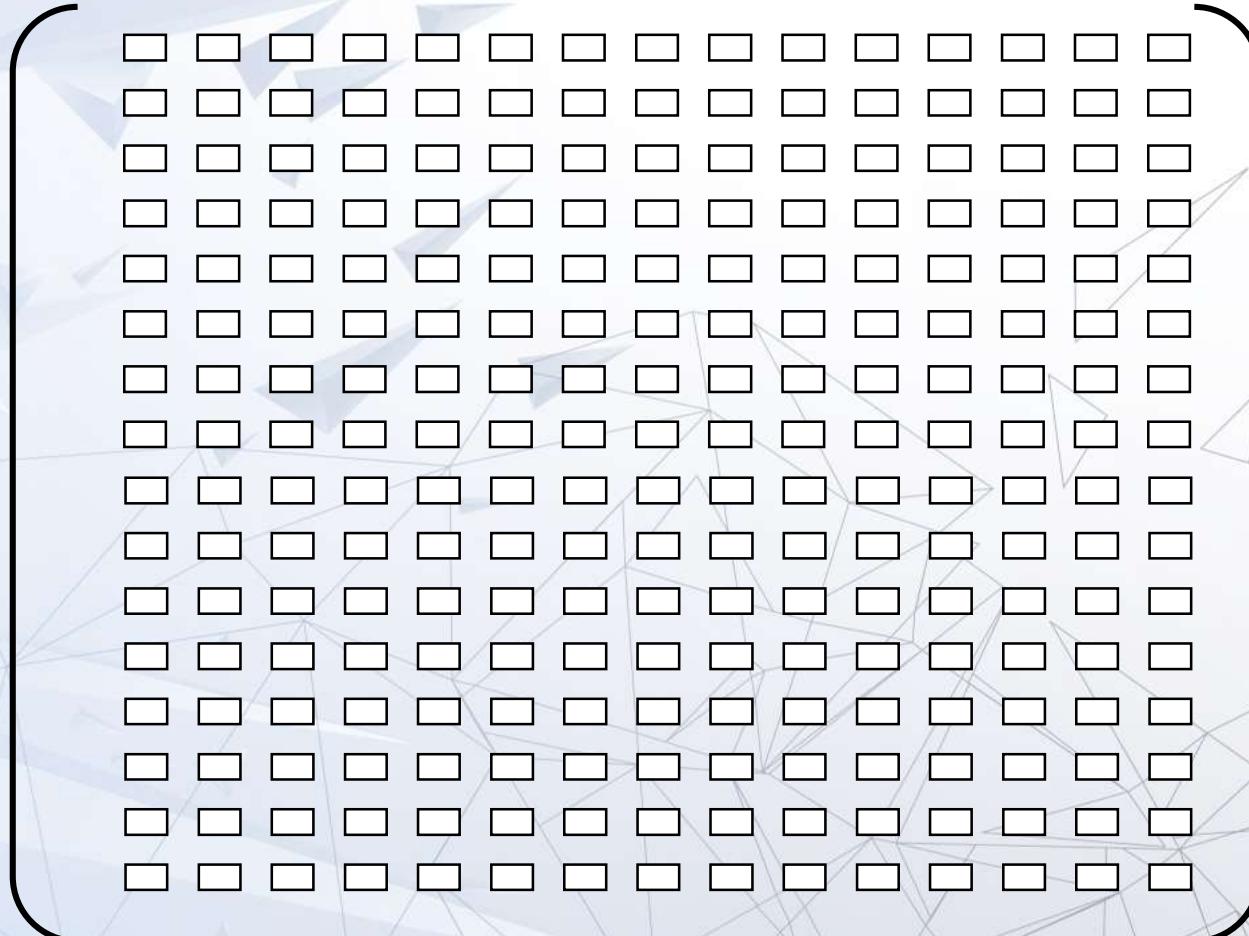
La résolution des contraintes / Le solveur

Chaque contrainte se traduit par une ou plusieurs équations ou inéquations :

- 1 degré de liberté bloqué → 1 équation
- 1 degré de liberté limité → 2 inéquations
- 1 point de contact → 1 inéquation
- Gestion de la friction → 1 ou plusieurs inéquations supplémentaires

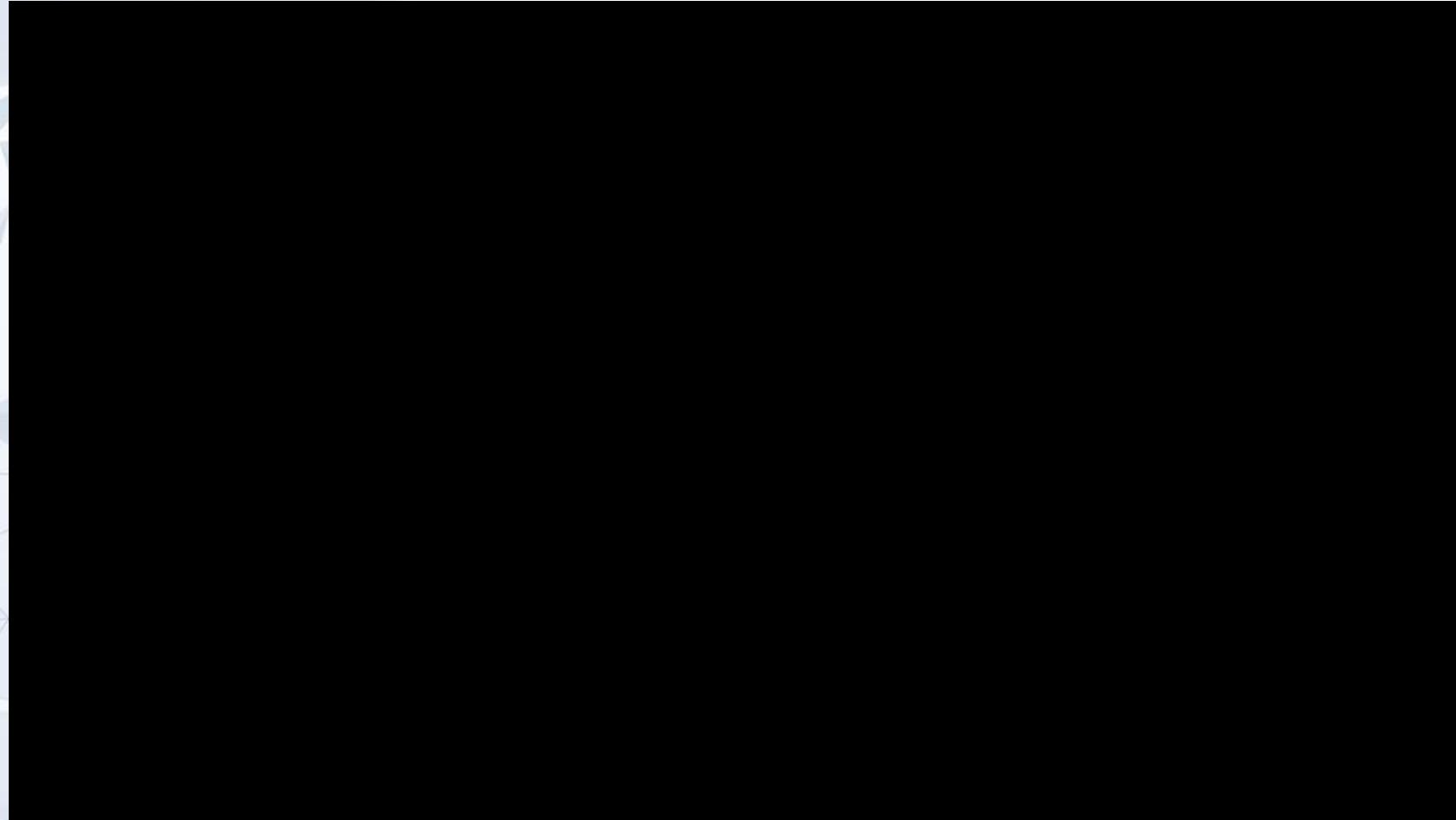
La résolution des contraintes / Le solveur

LCP (Linear Complementary Problem)



- 21 cubes
- 51 paires de corps
- 7 inéquations par paires
(contact boite-boite + friction)
- 357 inéquations

► La résolution des contraintes / Le solveur



UBISOFT®

La résolution des contraintes / Le solveur

Résolution dans le cas général : $O(n^3)$ → Inutilisable

On utilise des méthodes itératives qui ont une complexité de $O(n)$:

- Méthode de Gauss-Seidel :
 - . Converge vite 
 - . Le résultat dépend de l'ordre de résolution des contraintes 
- Méthode de Jacobi :
 - . Le résultat est indépendant de l'ordre de résolution 
 - . La convergence est plus lente 

Contenu du cours

Introduction

Alexis Vaisse

Moteurs physiques existants

Que fait un moteur physique ?

La physique des corps rigides indéformables

1. La détection des collisions

1.1 La broad phase

1.2 La narrow phase

2. La résolution des contraintes

2.1 Les différents types de joints

2.1.1 Joint fixe

2.1.2 Joint sphérique, joint ragdoll

2.1.3 Joint charnière

2.1.4 Joint prismatique

2.1.5 Joint 6 dof

2.1.7 Les joints cassables

2.1.8 Les joints motorisés

2.2 Le solveur



UBISOFT

La physique des corps déformables

—

Exemple de la simulation de vêtements



UBISOFT®

La simulation de vêtements / Introduction

- Algorithme basé sur une représentation la plus réaliste possible
- Exemple : ~~Simulation par Eléments Finis~~
- Les propriétés du matériaux sont définies par le tenseur d'élasticité à chaque point



Performances pas suffisantes pour une utilisation dans les jeux vidéo

La simulation de vêtements / Introduction

Dans les jeux vidéo :

- Algorithmes non basés sur une simulation réaliste
- Beaucoup plus rapides
- On ajuste les paramètres pour que la simulation ait l'air aussi réaliste que possible

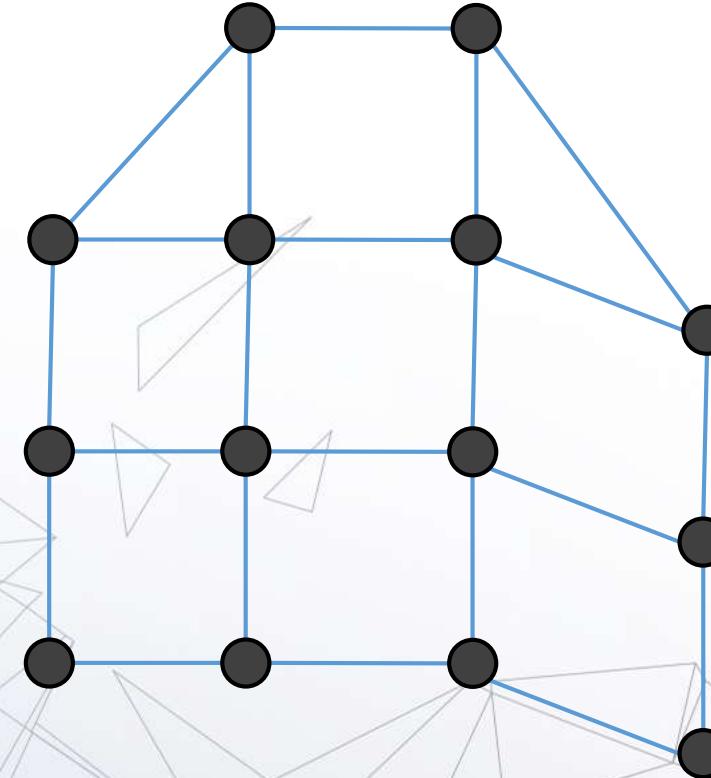


UBISOFT

La simulation de vêtements / Les contraintes

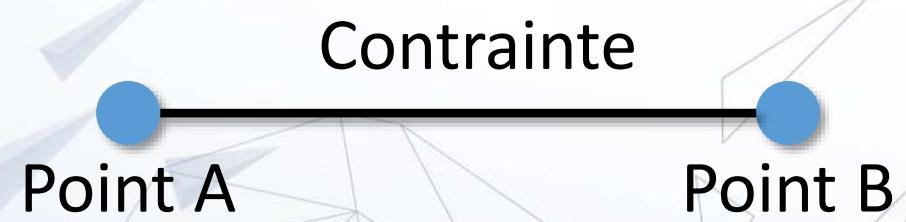


Contrainte



UBISOFT

► La simulation de vêtements / Les contraintes



► La simulation de vêtements / Les contraintes



UBISOFT®

► La simulation de vêtements / Les contraintes



UBISOFT®

► La simulation de vêtements / Les contraintes

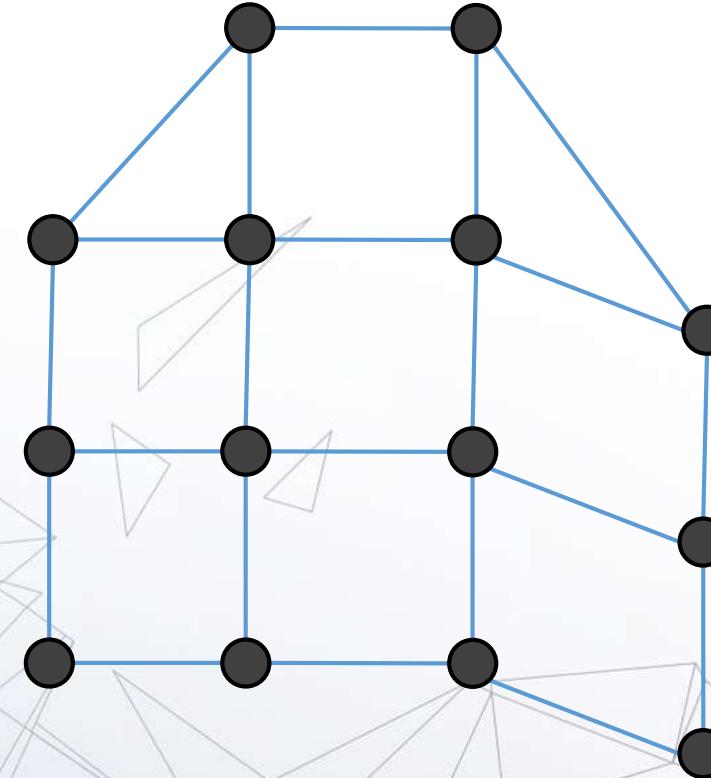
```
void ApplyConstraint
    (Vector &A,
     Vector &B,
     float   RestingLength)
{
    float Distance = Distance(A, B);
    Vector CorrectionVector =
        (Distance - RestingLength) * (B - A) / Distance;

    A += CorrectionVector / 2;
    B -= CorrectionVector / 2;
}
```



La simulation de vêtements / Les contraintes de structure

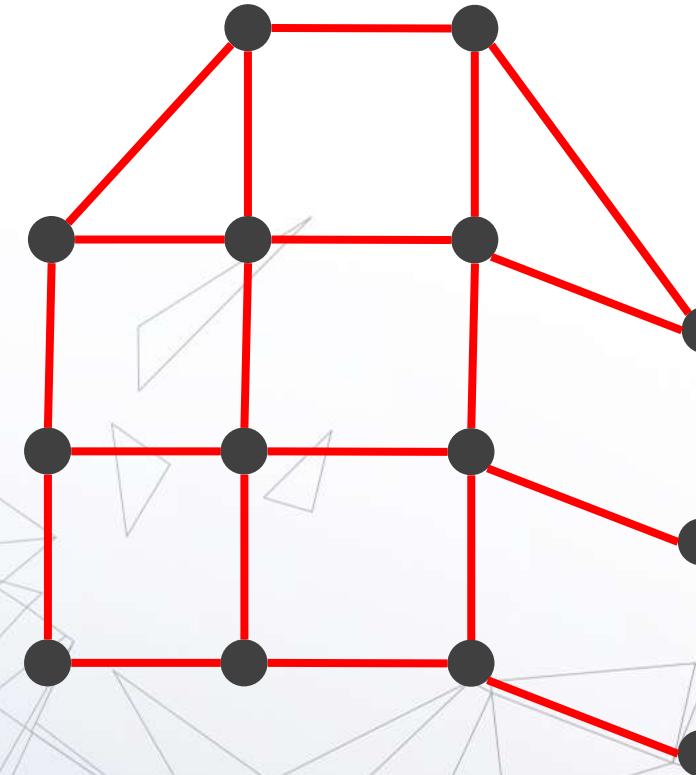
Contraintes de structure :



UBISOFT®

La simulation de vêtements / Les contraintes de structure

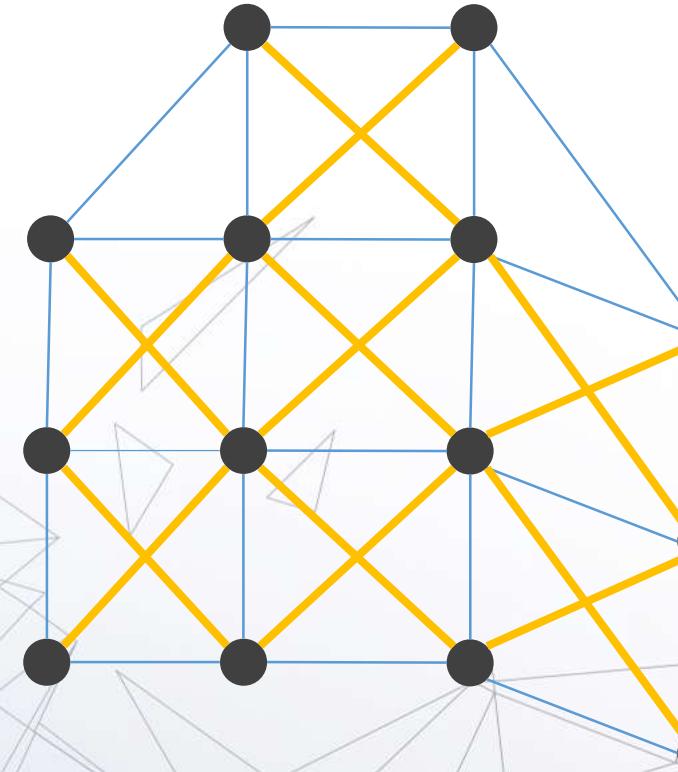
Contraintes de structure :



UBISOFT®

La simulation de vêtements / Les contraintes de cisaillement

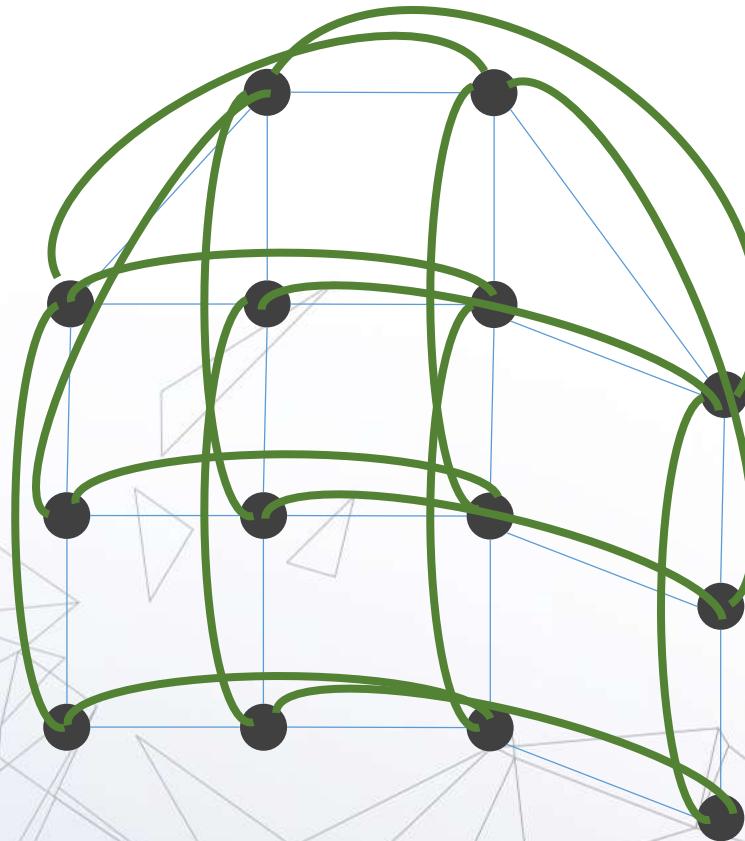
Contraintes de cisaillement (shearing) :



UBISOFT

La simulation de vêtements / Les contraintes de flexion

Contraintes de flexion
(bending) :



La simulation de vêtements / Les contraintes

Pour un mesh de n points :

- ~ $2n$ contraintes de structure
- ~ $2n$ contraintes de cisaillement
- ~ $2n$ contraintes de flexion



Pour une passe

Pour un mesh de 1 000 points :

6 000 contraintes \times 3 passes \times 20 opérations = 360 000 opérations



UBISOFT®

La simulation de vêtements / Les contraintes

Simulation de vêtements par intégration de Verlet :

Avantages :



Performances

Inconvénients :



Le comportement dépend de la résolution du mesh



UBISOFT

► La simulation de vêtements / Les contraintes



UBISOFT®

La simulation de vêtements / Les contraintes

Simulation de vêtements par intégration de Verlet :

Avantages :



Performances

Inconvénients :



Le comportement dépend de la résolution du mesh



La pose de base est « enregistrée » dans le vêtement



UBISOFT®

► La simulation de vêtements / Les contraintes



UBISOFT

La simulation de vêtements / Les contraintes

Simulation de vêtements par intégration de Verlet :

Avantages :



Performances

Inconvénients :



Le comportement dépend de la résolution du mesh



La pose de base est « enregistrée » dans le vêtement



Pas de plasticité

La simulation de vêtements

- Les contraintes – La simulation par intégration de Verlet



UBISOFT®

La simulation de vêtements

- Les contraintes – La simulation par intégration de Verlet
- La gestion des points attachés

► La simulation de vêtements / Les points attachés

Gestion des points attachés :



UBISOFT



La simulation de vêtements / Les points attachés

Gestion des points attachés :

```
void ApplyConstraint
    (Vector &A, bool AIsAttached,
     Vector &B, bool BIsAttached,
     float   RestingLength)
{
    assert(!AIsAttached || !BIsAttached);
    float Distance = Distance(A, B);
    Vector CorrectionVector = (Distance - RestingLength) * (B - A) / Distance;

    if (AIsAttached)          B -= CorrectionVector;
    else if (BIsAttached)    A += CorrectionVector;
    else {
        A += CorrectionVector / 2;
        B -= CorrectionVector / 2; }
}
```



► La simulation de vêtements / Les points attachés

Déroulement des calculs :

1. Calcul de l'animation → Position des os
2. Calcul du skinning → Position des points attachés
3. Calcul de la simulation de vêtement

► La simulation de vêtements / Les points attachés

Gestion des points attachés :



UBISOFT®

La simulation de vêtements

- Les contraintes – La simulation par intégration de Verlet
- La gestion des points attachés



UBISOFT®

La simulation de vêtements

- Les contraintes – La simulation par intégration de Verlet
- La gestion des points attachés
- Les collisions

La simulation de vêtements / Les collisions

Collision avec des primitives :

- Sphères
- Capsules
- Boites

On utilise des structures d'accélération :

- Exemple : AABB tree
- Mis à jour à chaque trame



La simulation de vêtements / Les collisions

Déroulement des calculs :

- Calcul de l'animation → Position des os
- Calcul de la position des primitives de collision (capsules...)
- Calcul du skinning → Position des points attachés
- Calcul de la simulation de vêtement

La simulation de vêtements / Les collisions

2 possibilités :

- Collision entre les points du vêtement et les primitives

- Rapide 



- Collision entre les triangles du vêtement et les primitives

- Beaucoup moins de problèmes de collision 



La simulation de vêtements / Les collisions

Pour minimiser les problèmes de collision :

- Augmenter la fréquence de calcul (exemple : 60 Hz au lieu de 30)
- Collisions continues
- Augmenter la résolution
 - Uniquement pour les collisions
 - ➡ Résolution différente pour la simulation et pour les collisions
 - ➡ Permet de ne pas modifier le comportement

La simulation de vêtements / Les collisions

Collisions vêtement – mesh et vêtement - vêtement :

- Possible mais très coûteux
- Collision entre les points du vêtement et les triangles du mesh
- AABB tree contenant les point du vêtement
- AABB tree contenant les triangles du mesh



UBISOFT®

► La simulation de vêtements / Les collisions

Collision vêtement - vêtement :



UBISOFT®

La simulation de vêtements

- Les contraintes – La simulation par intégration de Verlet
- La gestion des points attachés
- Les collisions

La simulation de vêtements

- Les contraintes – La simulation par intégration de Verlet
- La gestion des points attachés
- Les collisions
- Les déchirures

► La simulation de vêtements / Les déchirures



UBISOFT®

La simulation de vêtements / Les déchirures

Pour chaque contrainte :

- Si la contrainte est trop étirée, elle casse
- Exemple : cassure si $d_{courante} > k \cdot d_{repos}$
 - $k = 1,5$ → Vêtement très fragile
 - $k = 2$ → Vêtement moyennement solide
 - $k = 3$ → Vêtement très solide

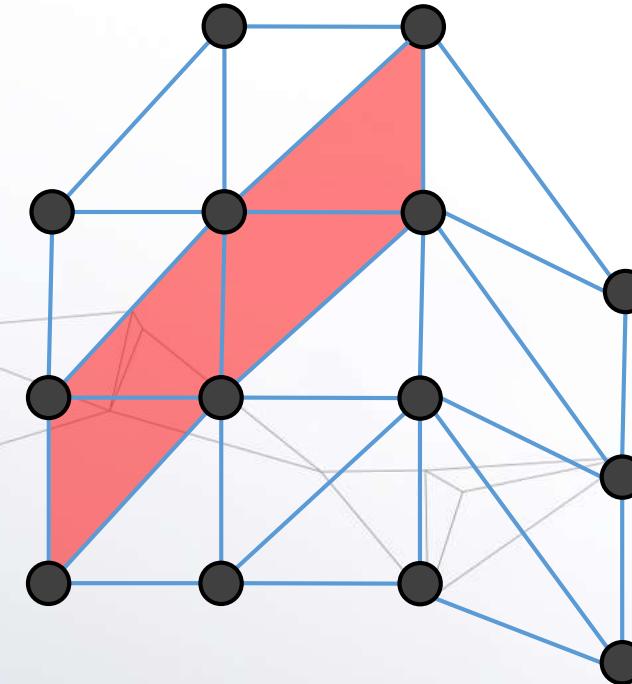
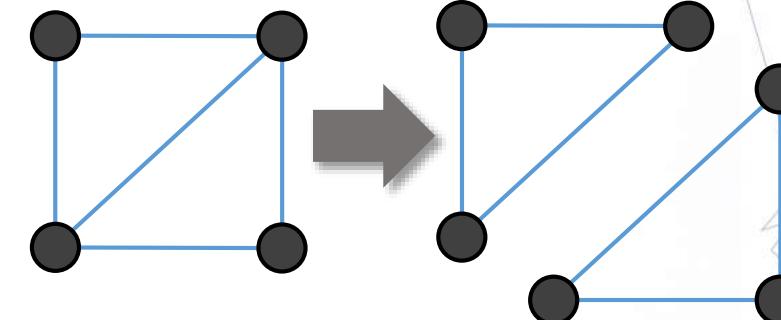


UBISOFT

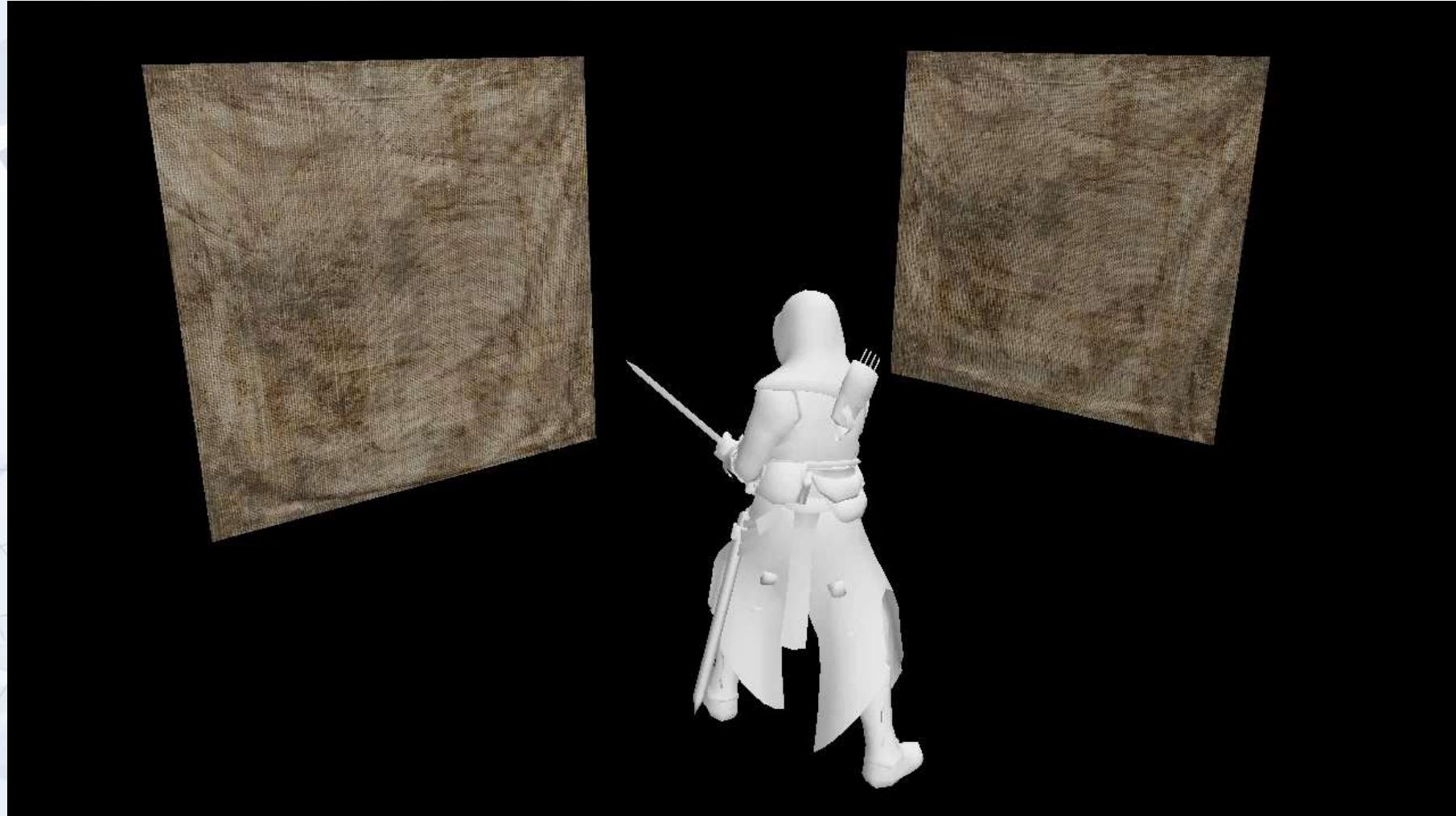
La simulation de vêtements / Les déchirures

2 techniques :

- Les contraintes sont désactivées
 - Nécessite de modifier le vertex buffer 
- Les contraintes sont désactivées et les triangles sont masqués
 - Pas de modification du vertex buffer 
 - Perte de matière 

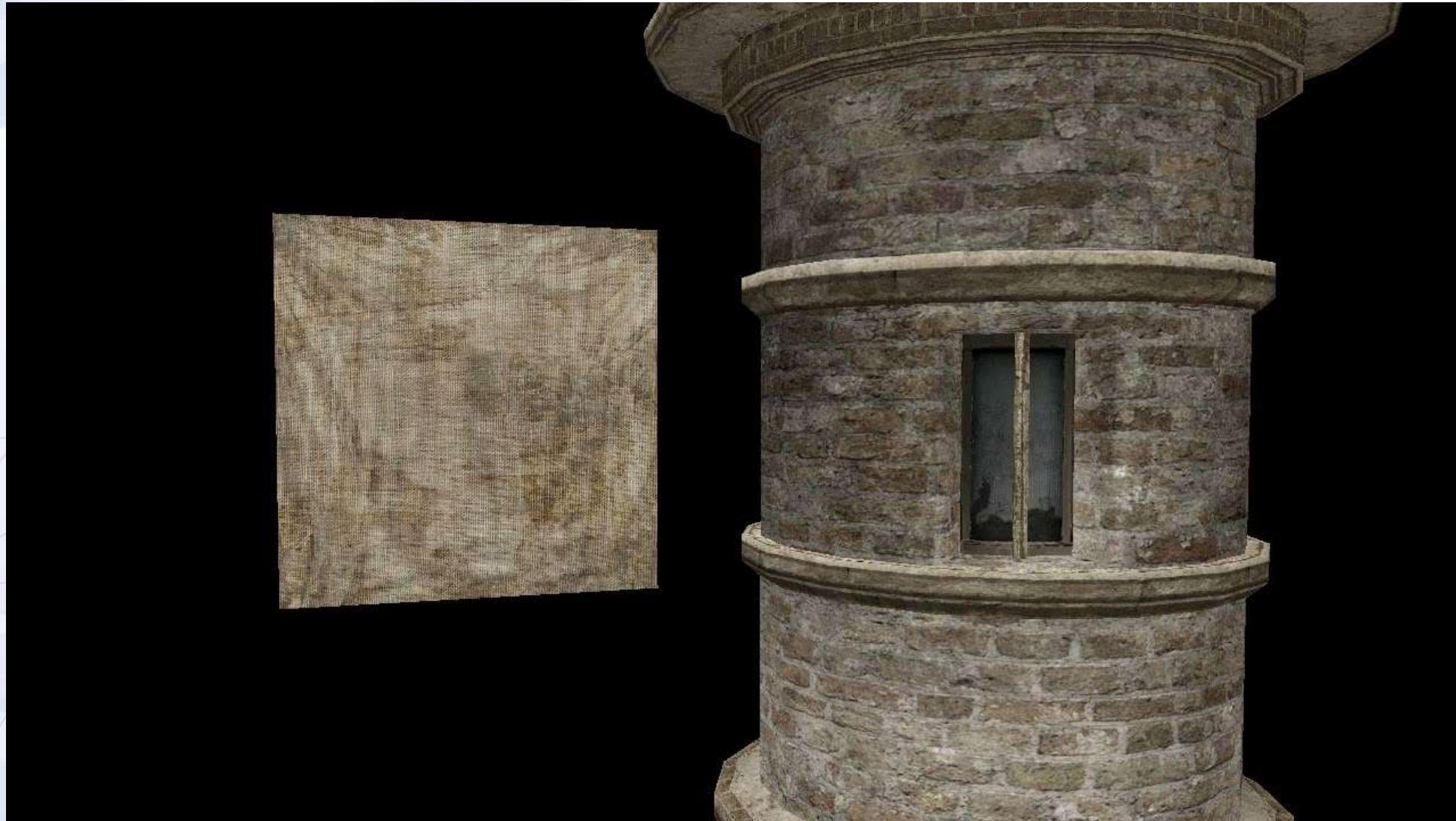


► La simulation de vêtements / Les déchirures



UBISOFT®

► La simulation de vêtements / Les déchirures



UBISOFT®

La simulation de vêtements

- Les contraintes – La simulation par intégration de Verlet
- La gestion des points attachés
- Les collisions
- Les déchirures

La simulation de vêtements

- Les contraintes – La simulation par intégration de Verlet
- La gestion des points attachés
- Les collisions
- Les déchirures
- Le contrôle des vêtements

► La simulation de vêtements / Le contrôle des vêtements

Beaucoup de paramètres
« non physiques »

- Parce que le solveur n'est pas parfait
- Parce que les situations de jeux ne sont pas réalistes

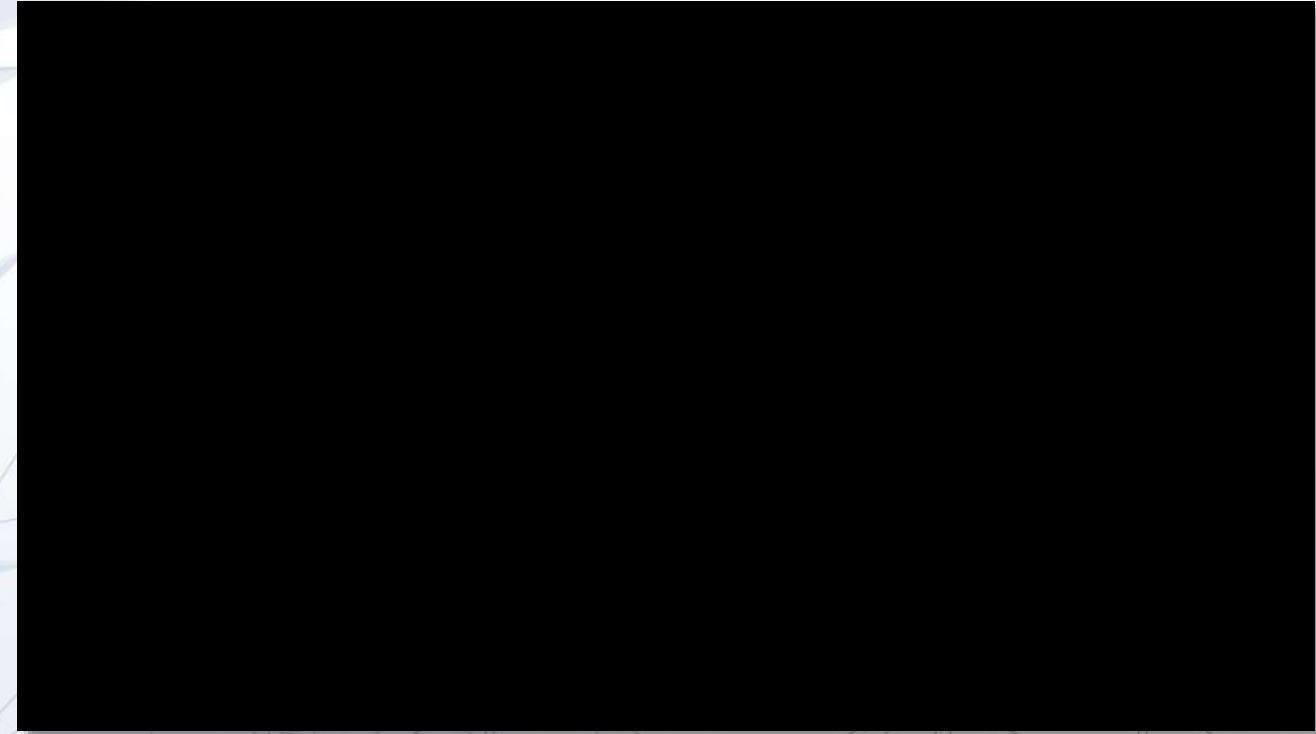


► La simulation de vêtements / Le contrôle des vêtements



UBISOFT®

► La simulation de vêtements / Le contrôle des vêtements



UBISOFT®

► La simulation de vêtements / Le contrôle des vêtements



UBISOFT

La simulation de vêtements

- Les contraintes – La simulation par intégration de Verlet
- La gestion des points attachés
- Les collisions
- Les déchirures
- Le contrôle des vêtements

Le travail d'un programmeur physique



UBISOFT®

Equipe
moteur



Programmeur
physique

Equipe
gameplay



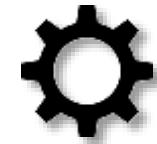
Gameplay
programmeur
physique



UBISOFT

Programmeur physique / Développement de technologies

Développement de technologies :



R&D

- Moteur physique
- Moteur de simulation de vêtements
- Moteur de destruction

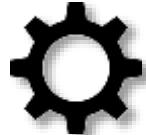


UBISOFT



Programmeur physique / Intégration de middlewares

Intégration de middlewares de physique :



- Met en place toute l'interface qui permettra aux gameplay programmeurs d'utiliser la physique :
 - Le code : abstraction du bas niveau / adaptation aux besoins
 - La gestion des données
- Tient compte des contraintes du moteur (exemple : multithreading)
- Les performances sont cruciales
- Joue le rôle d'expert physique auprès des autres programmeurs



Généralement un rôle de senior programmeur



UBISOFT

Programmeur physique / Character controller

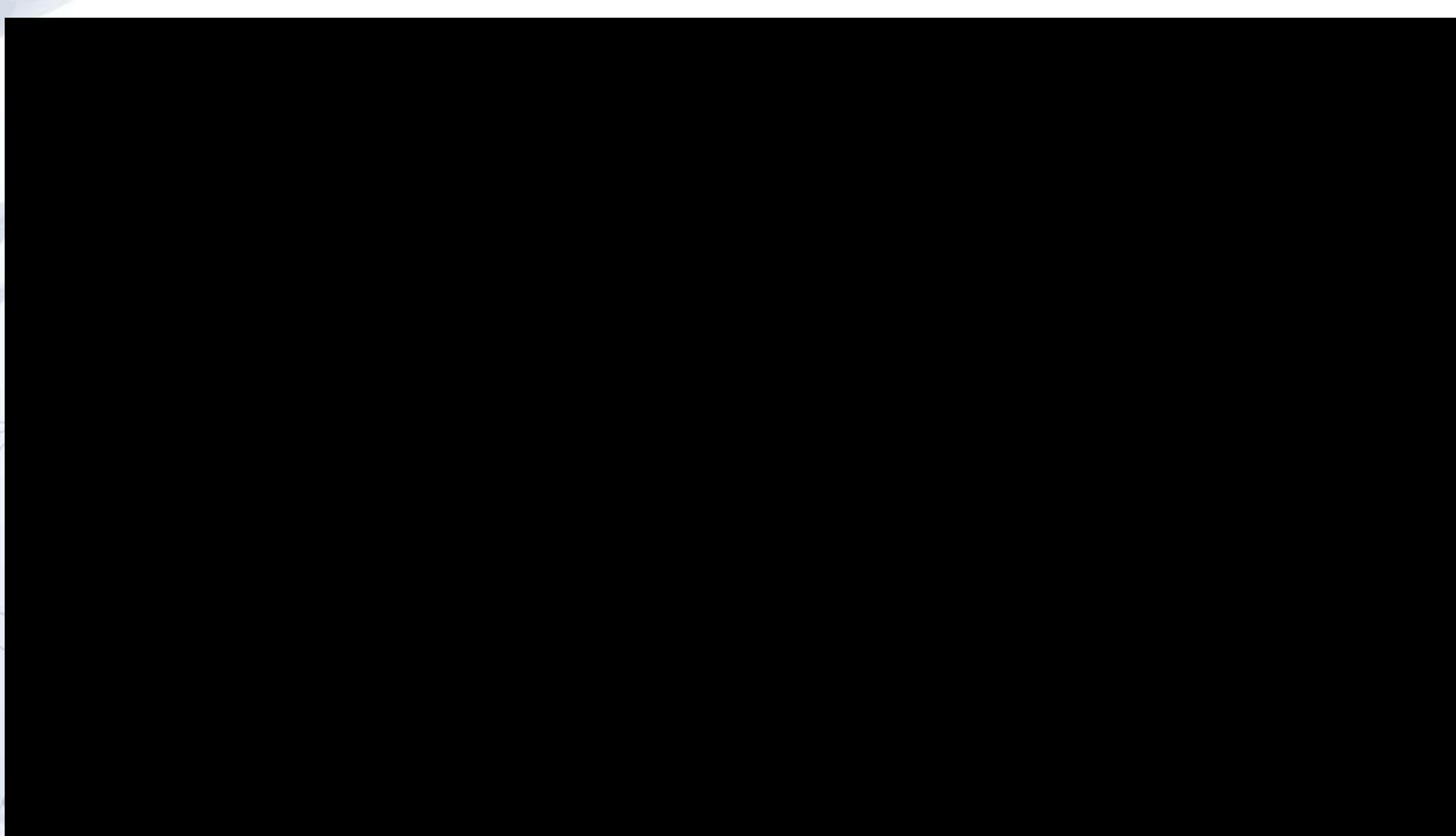
Character controller :



- Code qui contrôle les déplacements d'un personnage (PC et / ou NPC)
- 2 types de character controllers :
 - Non physique :
 - Basé sur des triggers, des lancers de rayons ou de sphères
 - Toutes les interactions (sol, mur obstacle, autres personnages) sont gérées explicitement
 - Physique :
 - 1 personnage = 1 capsule
 - Peut changer de forme (debout / accroupi / rampe)

► Programmeur physique / Character controller

Character controller :



UBISOFT®

Programmeur physique / Character controller

Character controller :



- Character controller non physique :

- Meilleur contrôle de tout ce qu'il se passe 
- Difficile d'obtenir un comportement 100 % crédible 
- Beaucoup de travail pour gérer tous les cas 
- Le comportement peut être répétitif 

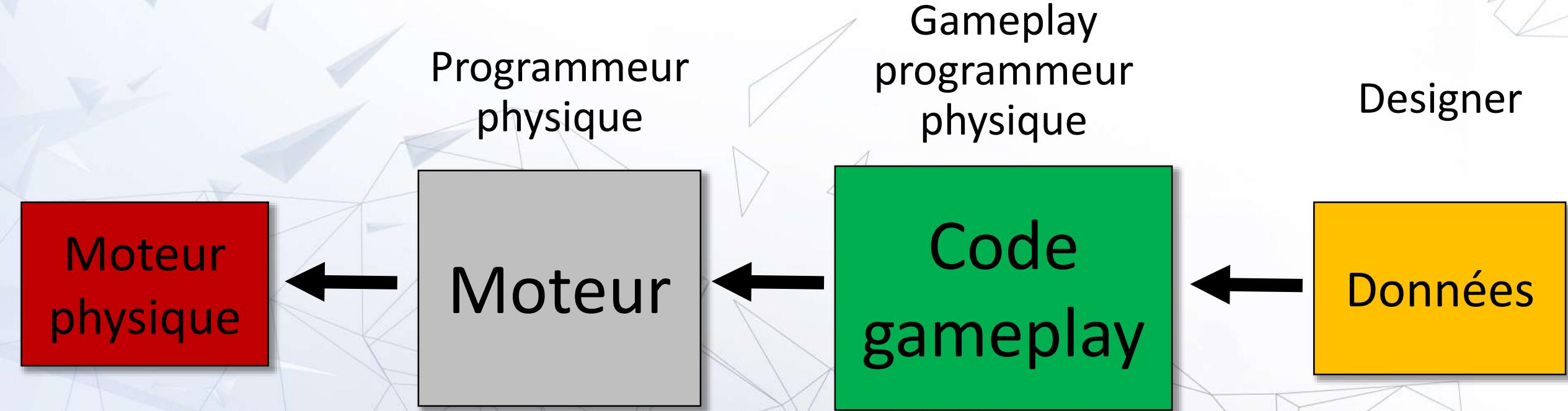
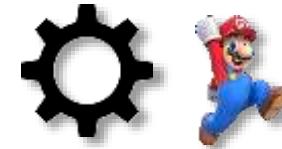
- Character controller physique :

- On a rapidement un premier comportement physique 
- Comportement plus réaliste 
- Des actions potentiellement différentes à chaque fois 
- On a des comportements non prévus 
- Beaucoup de travail pour que tout fonctionne correctement 

Programmeur physique / Vehicle controller

Vehicle controller :

- Réglage du comportement physique des véhicules



Programmeur physique / Ragdoll

Ragdoll passif :



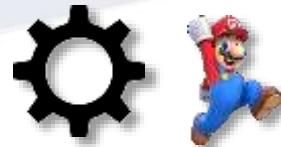
- Réglage de ragdoll
- Passage du mode « animation » au mode « ragdoll » ← Délicat
- Gérer tous les problèmes et les cas particuliers qui vont se poser



UBISOFT

Programmeur physique / Ragdoll

Ragdoll actif :



- Nécessite un réglage « parfait »
- Aller vers une pose précise → Déterminer les bonnes formules de blend
- Jouer une animation par l'intermédiaire d'un moteur physique
→ Beaucoup de cas à gérer

Programmeur physique / Comportements physiques

- Nage
- Ski
- Glissade
- Gestion de la friction
- Gestion des vêtements
- Gestion des mouvements dans un référentiel qui bouge
(exemple : personnages dans un train à grande vitesse)



Programmeur physique / Gameplay

- Tests de visibilité
- Cover
- Tirs :
 - Ligne droite
 - Parabole (exemple : grenade)
 - Avec ou sans rebond
- Connaissance de l'environnement



UBISOFT

Programmeur physique / Gameplay Elements

- Objets qui ont des collisions
 - Objets cassables
-
- Dans l'idéal, avoir un pipeline de production qui permet aux artistes d'être indépendants
 - En pratique, il y a souvent du code gameplay à faire



UBISOFT

Programmeur physique / Conclusion

Au niveau moteur :

- Souvent une personne spécialisée

Au niveau gameplay :

- De nombreux gameplay programmeurs sont amenés à utiliser de la physique, de manière plus ou moins intensive

Beaucoup de dialogue et de collaboration moteur – gameplay

Beaucoup de tâches variées à faire



UBISOFT

Le travail d'un programmeur physique

- Développement de technologies
- Intégration de middlewares de physique
- Character controller
- Vehicle controller
- Ragdoll passif / Ragdoll actif
- Comportements physiques
- Gameplay
- Gameplay elements



UBISOFT

Contenu du cours

La physique des corps rigides indéformables

1. La détection des collisions

1.1 La broad phase

1.2 La narrow phase

2. La résolution des contraintes

2.1 Les joints

2.2 Le solveur

Questions ?

La simulation de vêtements

Le travail d'un programmeur physique



UBISOFT®