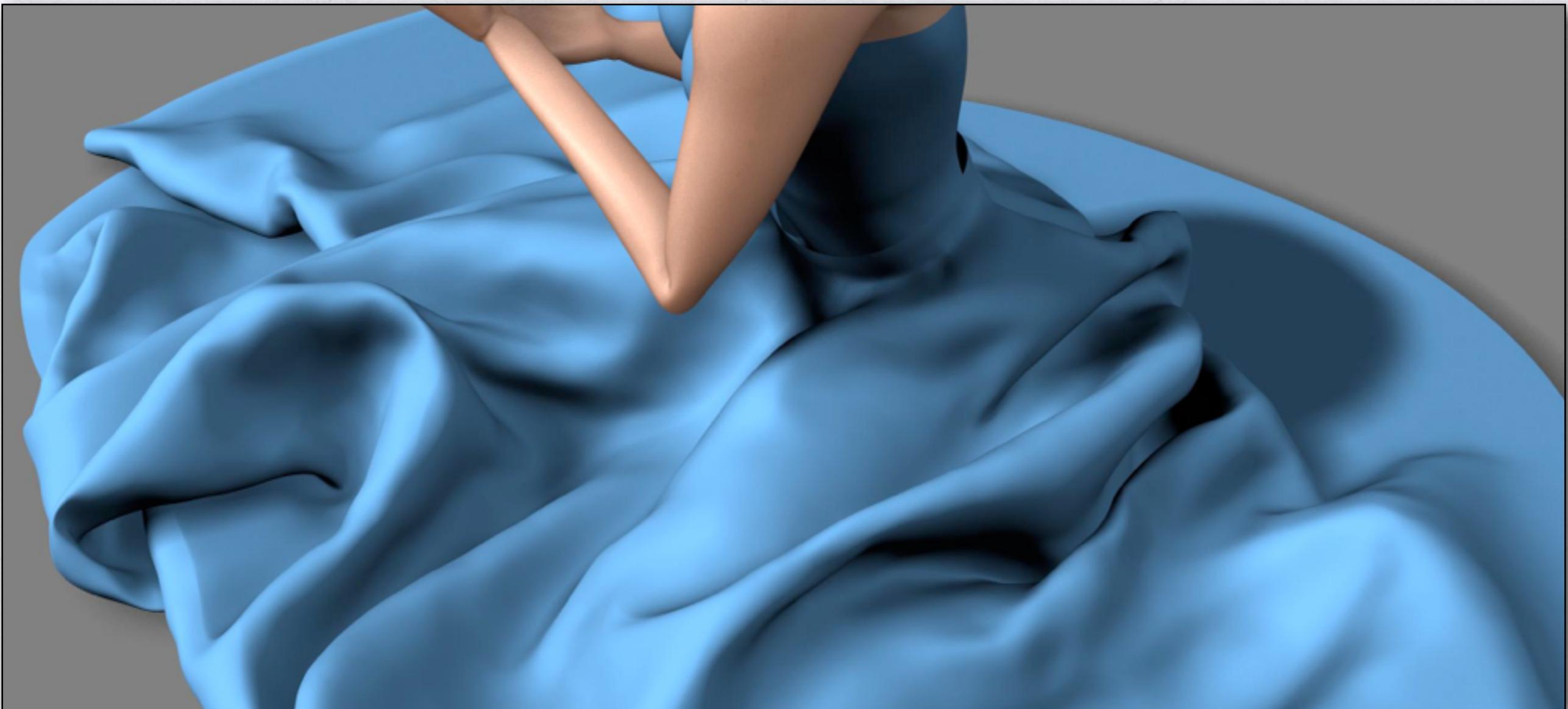


# DÉTECTION DE COLLISIONS

Jérémie Dequidt / [jeremie.dequidt@inria.fr](mailto:jeremie.dequidt@inria.fr) / <http://dequidt.plil.net>



# Plan

- \* Introduction (Rappels, Définition)
- \* Détection Spatiale
- \* Détection Spatio-Temporelle
- \* Primitives (Polyèdres, Surfaces Implicites, Paramétriques)
- \* Accélération (Pipeline, Broad Phase, Narrow Phase)
- \* Auto-Collisions

---

---

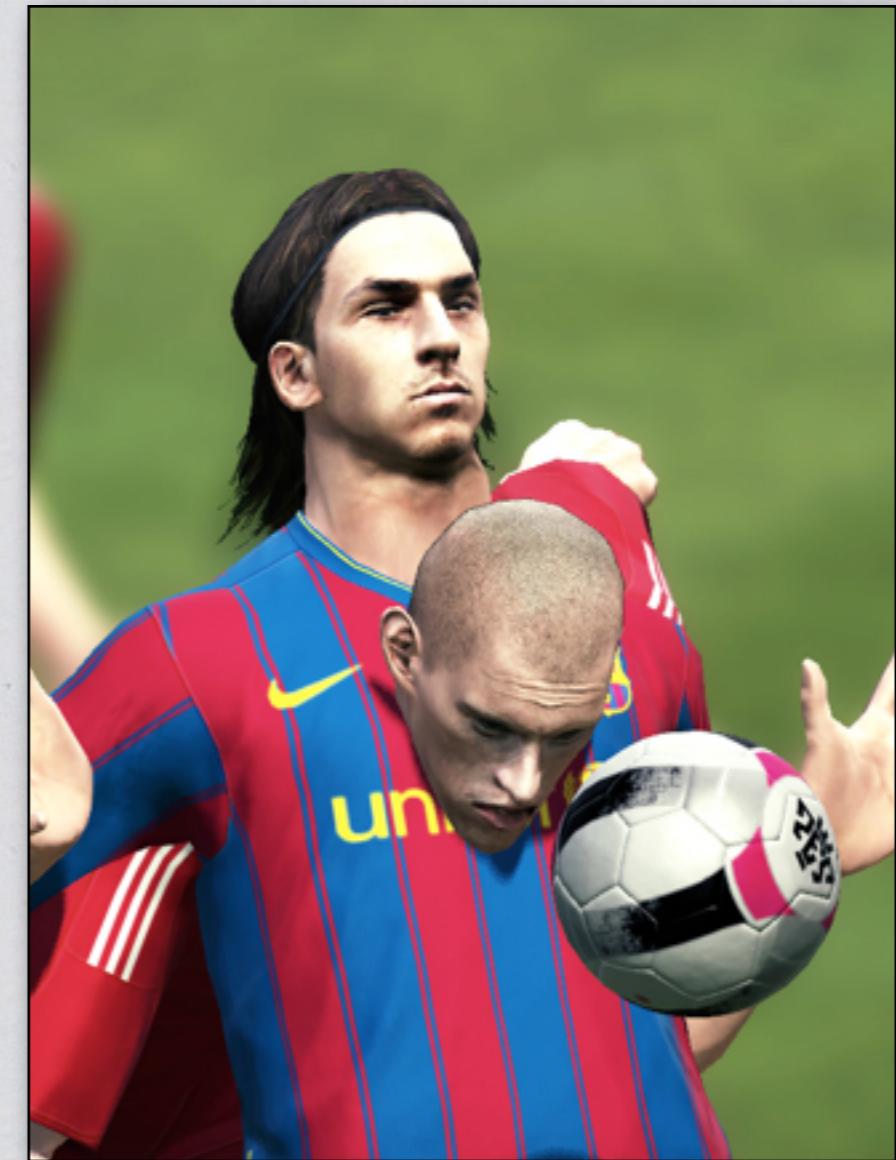
# 1. INTRODUCTION

Préambule, Objectifs, Rappels



# Préambule

- \* Objectif principal: donner de la consistance / tangibilité aux objets 3D
- \* Etudié en Mathématiques, Animation, Simulation Dynamique, Robotique, Interface Homme-Machine...



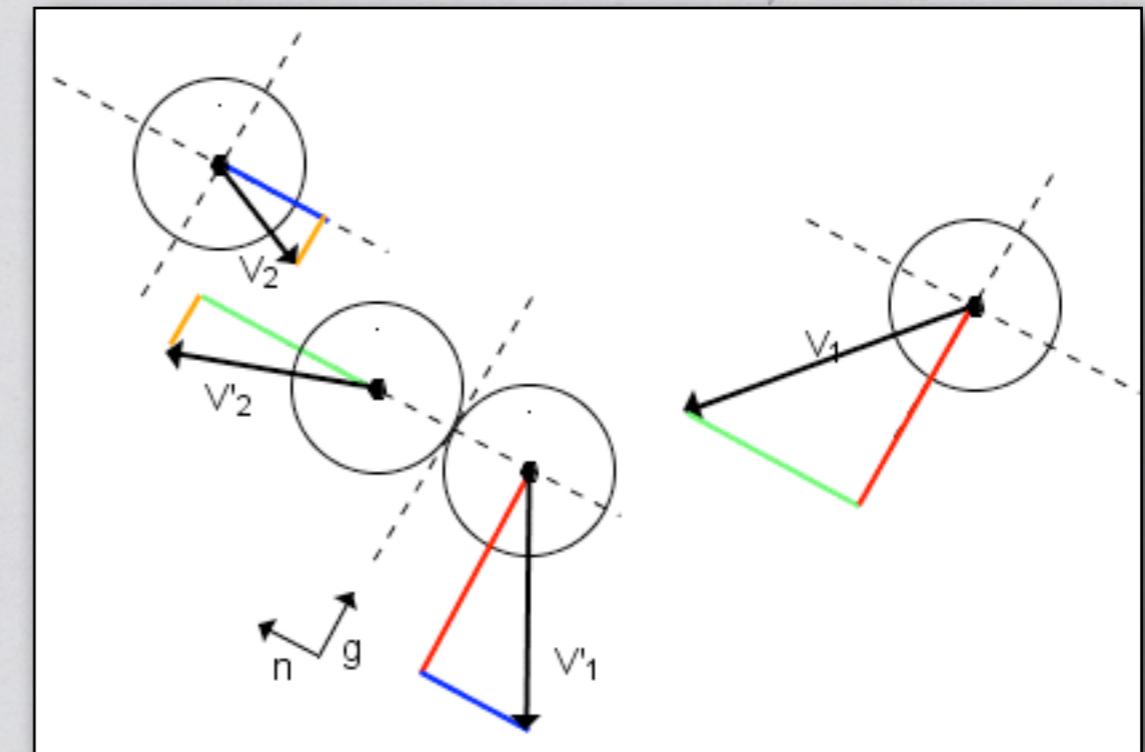
# Objectifs du cours



- \* Décrire les problèmes liés à la détection de collisions
- \* Description “*géométrique*” du problème ...
- \* ... pour la réponse mécanique à fournir lors d'une collision, voir le cours de Christian Duriez

# Rappels

- \* Théorie des chocs  
(élastique / inélastique):
- \* Actions réciproques  
*(action-réaction)*
- \* Nécessité de connaître  
l'instant de contact
- \* Nécessité de déterminer  
le plan de collision



# Rappels

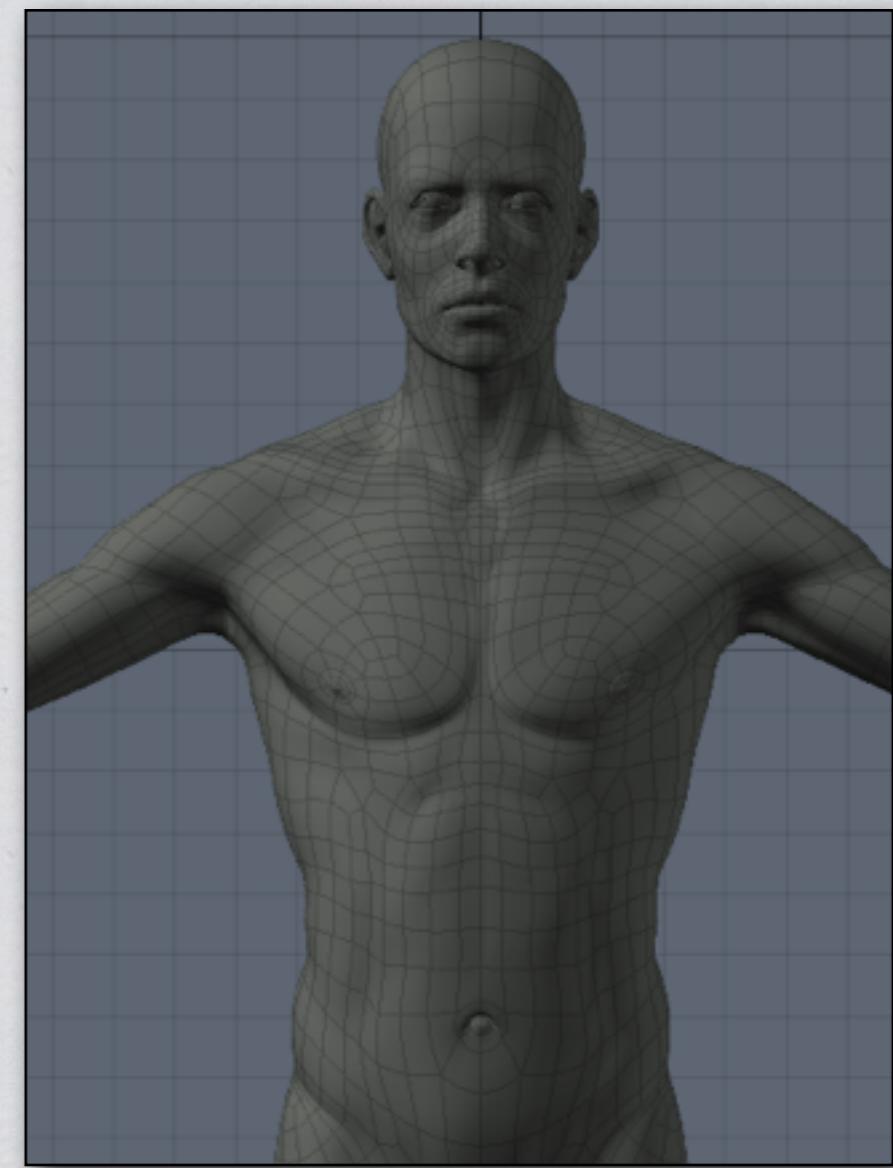
- \* Simulation discrète
- \* Positions des objets ne sont connus qu'à chaque pas de temps ( $k \times dt$ )
- \* Conséquence: difficulté de trouver l'instant de contact



**t = 0**

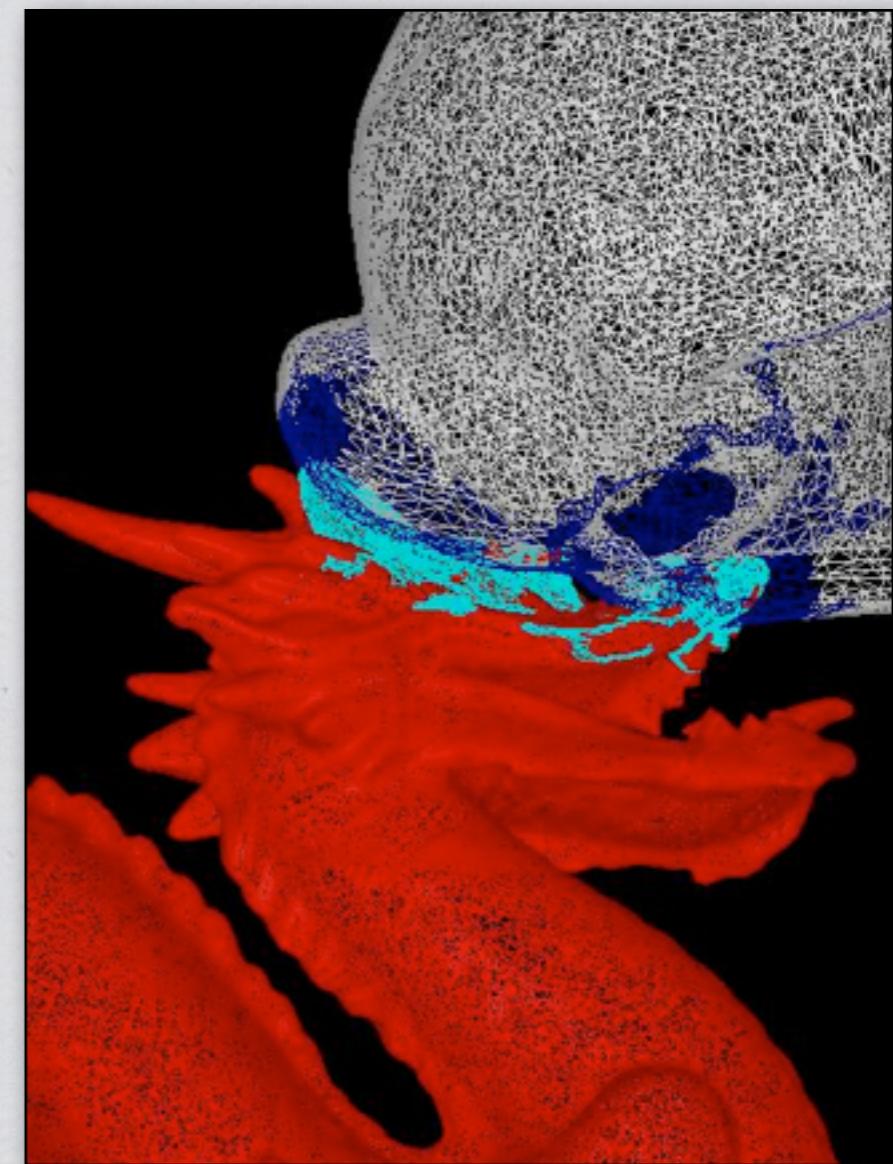
# Rappels

- \* Les modèles actuels sont composés d'un grand nombre de polygones (10k triangles au minimum)
- \* Géométrie complexe
- \* Conséquence: difficulté de déterminer points et plan de collision



# Définition

- \* Collision: existence d'un instant  $t$  où l'intersection de deux objets est non nulle.



# Deux “Stratégies”

- \* Détection Spatiale  
(détection à chaque  $dt$  de la simulation)
- \* Détection Spatio-Temporelle (prise en compte du temps dans la détection)



## 2. Détection Spatiale



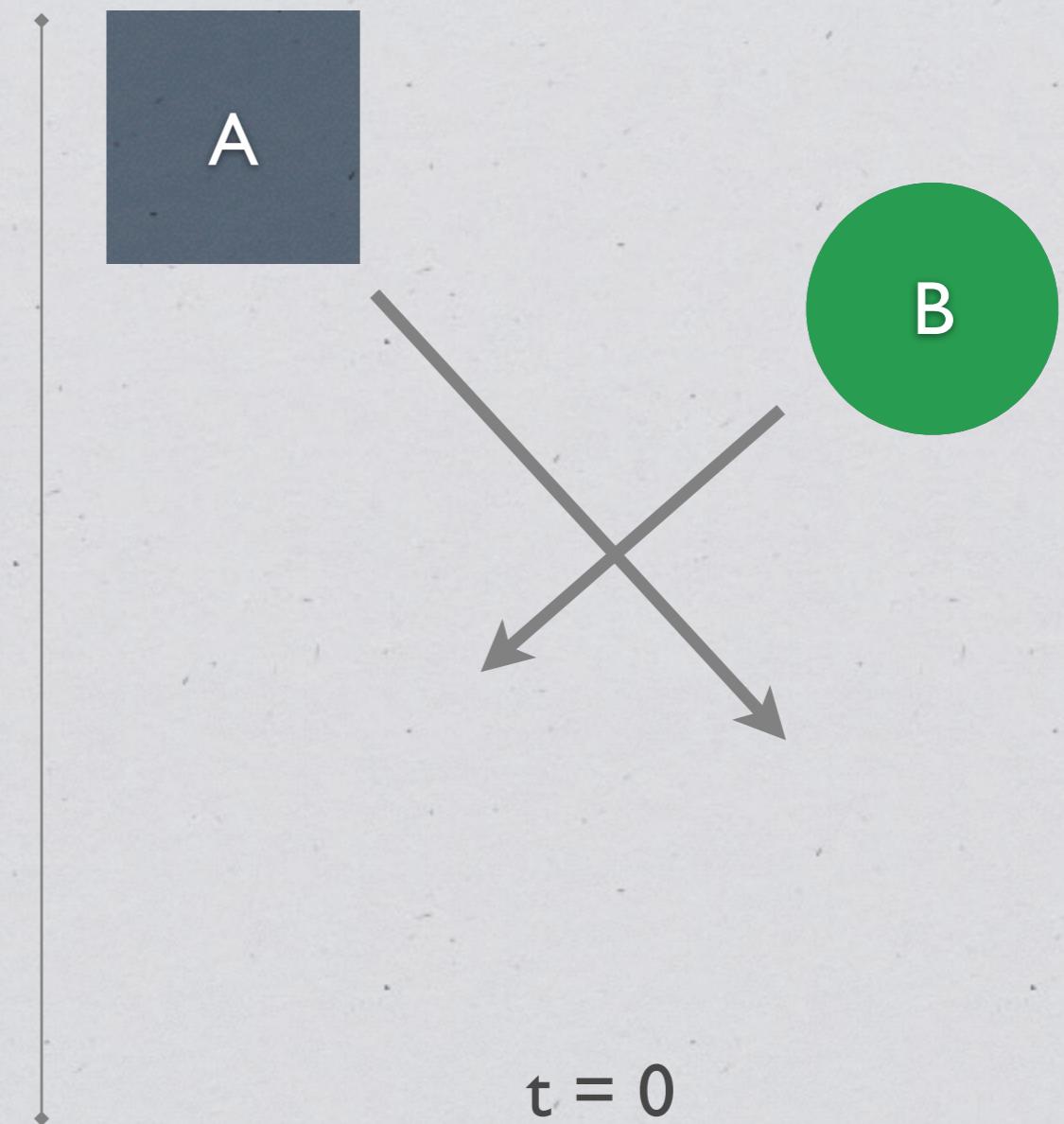
# Principe



- \* On observe le système à des instants donnés (*i.e.* tous les pas de temps de la simulation)
- \* On détecte les intersections non-vides entre objets
- \* Basé sur la géométrie algorithmique

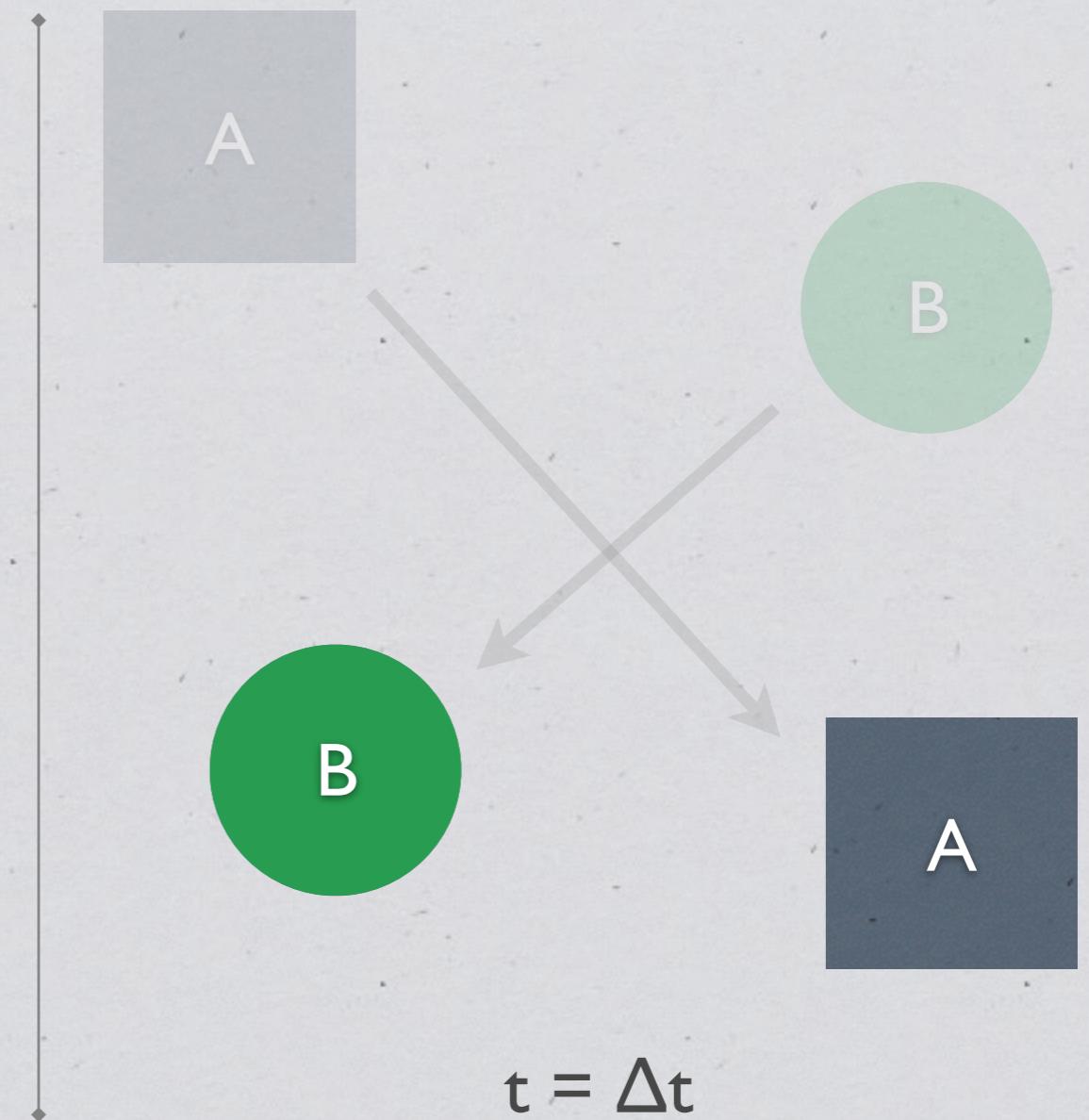
# Limite

\* Possibilité de “*manquer*”  
des collisions



# Limite

- \* Possibilité de “*manquer*” des collisions ... si vitesse trop importante



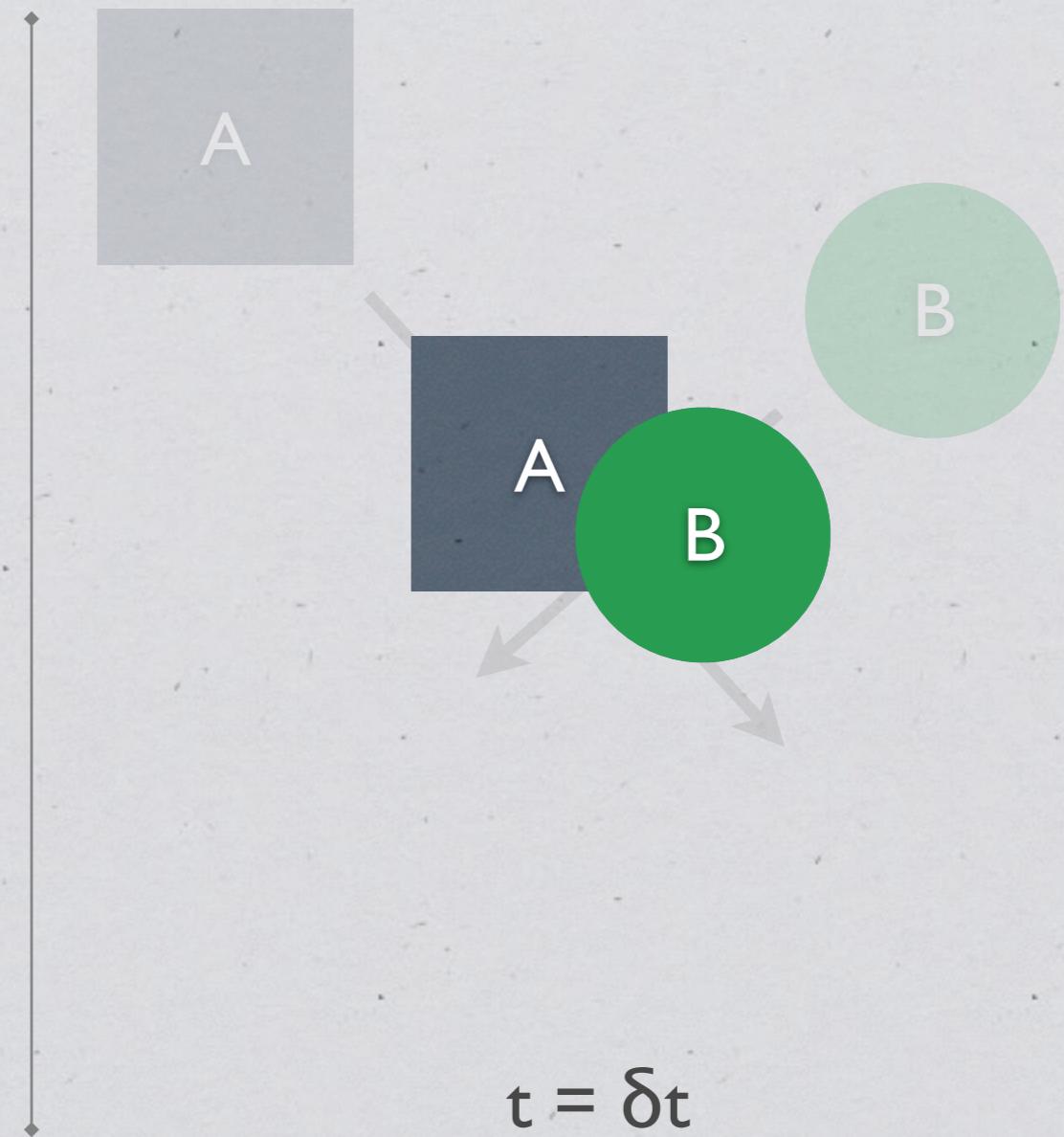
# Limite

- \* Possibilité de “*manquer*” des collisions ... si vitesse trop importante

- \* Notion de longueur critique:

$$l_c = v_{\max} \times \Delta t$$

- \* Les objets pourront passer l'un au travers de l'autre si leurs dimensions sont plus petites que  $l_c$

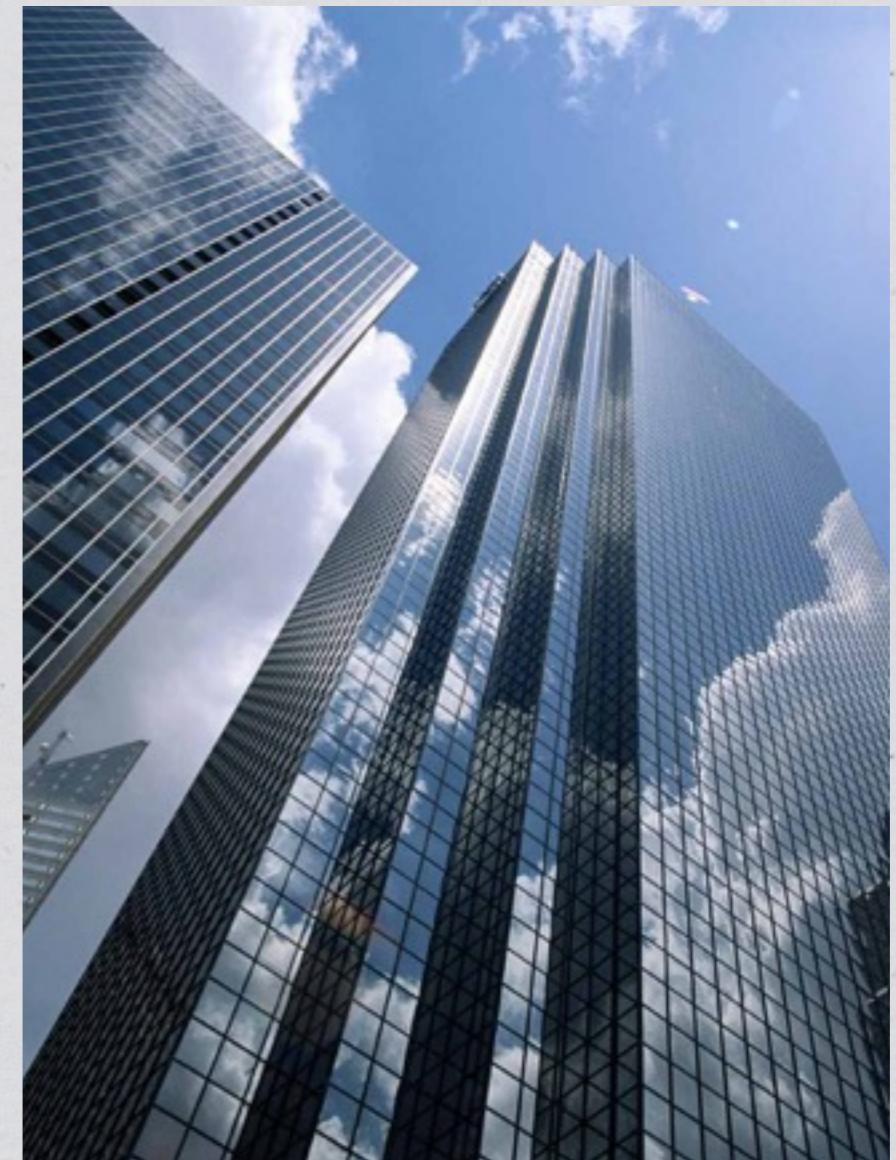


# Classes d'algorithmes

- \* Différents types d'informations peuvent caractériser une collision
- \* De manière générale, plus l'information est complète, plus l'algorithme pour la fournir est complexe
- \* Classification des méthodes par complexité croissante

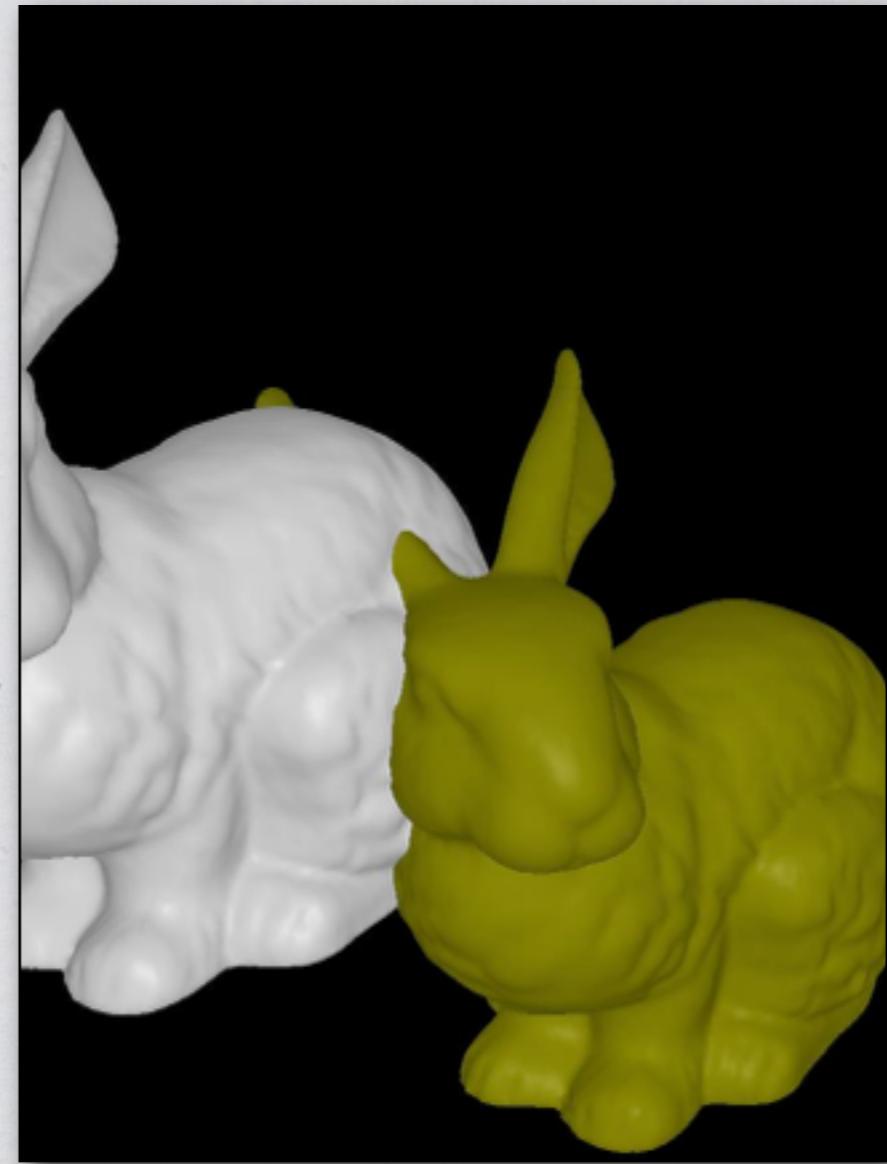
# Classes d'algorithmes

- \* Détection d'intersection non-vide
- \* réponse booléenne



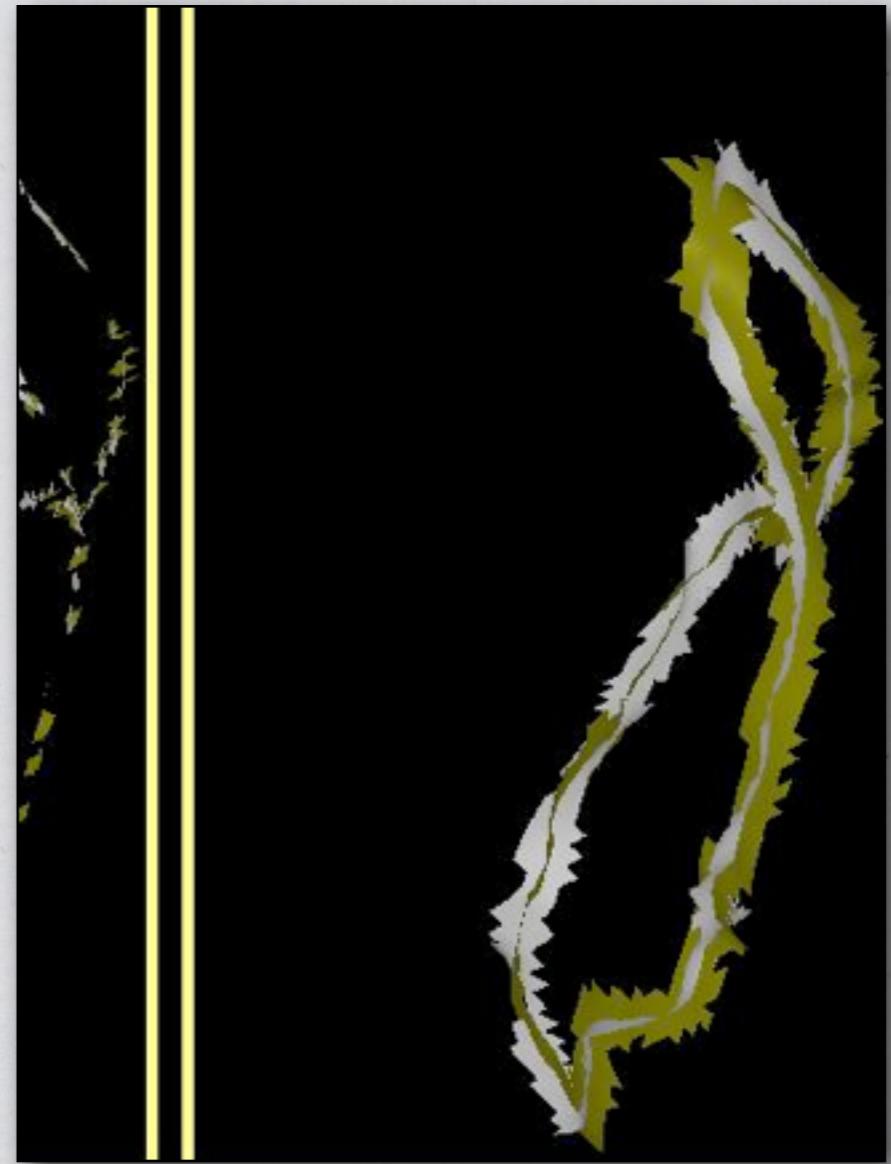
# Classes d'algorithmes

- \* Intersection des surfaces des objets:
- \* recherche des primitives d'un objet qui intersectent un autre objet



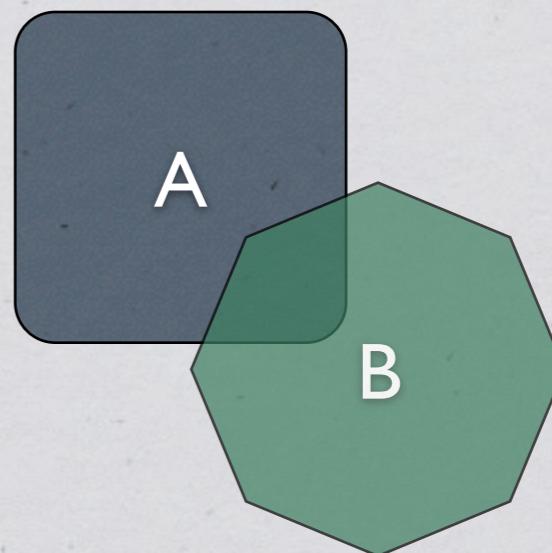
# Classes d'algorithmes

- \* Intersection des surfaces des objets:
- \* recherche des primitives d'un objet qui intersectent un autre objet



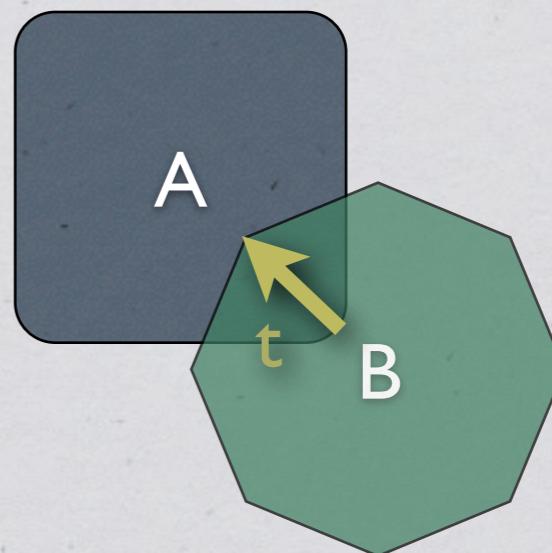
# Classes d'algorithmes

- \* Estimation de l'amplitude de l'intersection
- \* volume de l'intersection
- \* distance d'interpénétration (plus petite translation qui permet de séparer les objets)



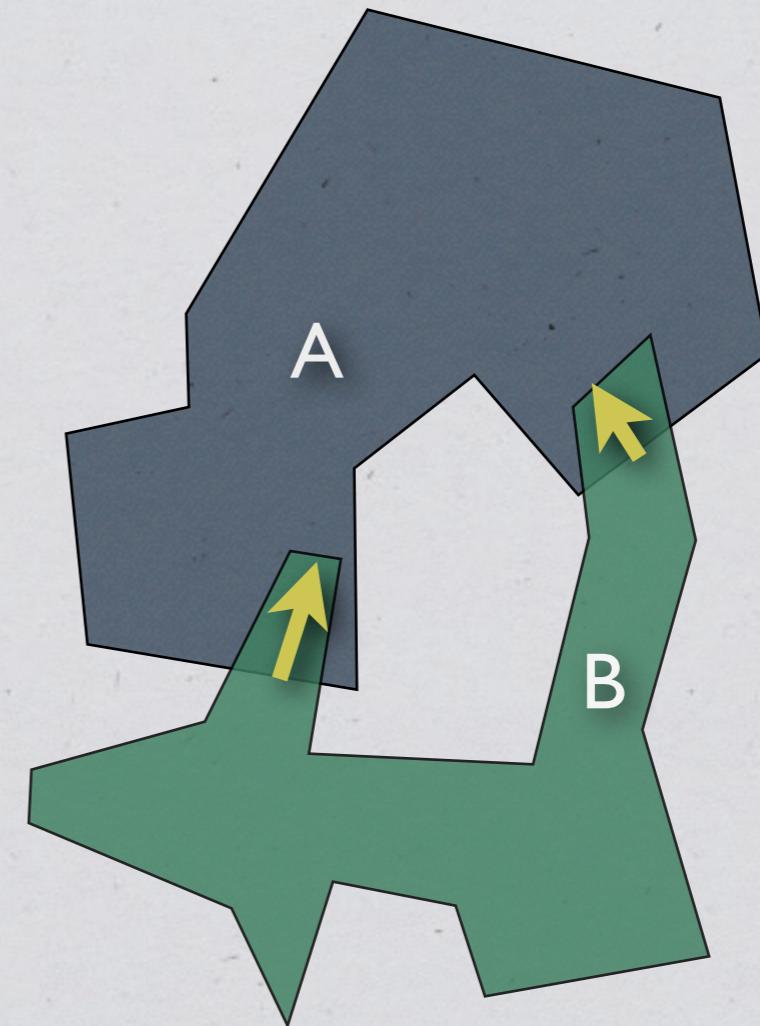
# Classes d'algorithmes

- \* Estimation de l'amplitude de l'intersection
- \* distance d'interpénétration (plus petite translation qui permet de séparer les objets)
- \* volume de l'intersection



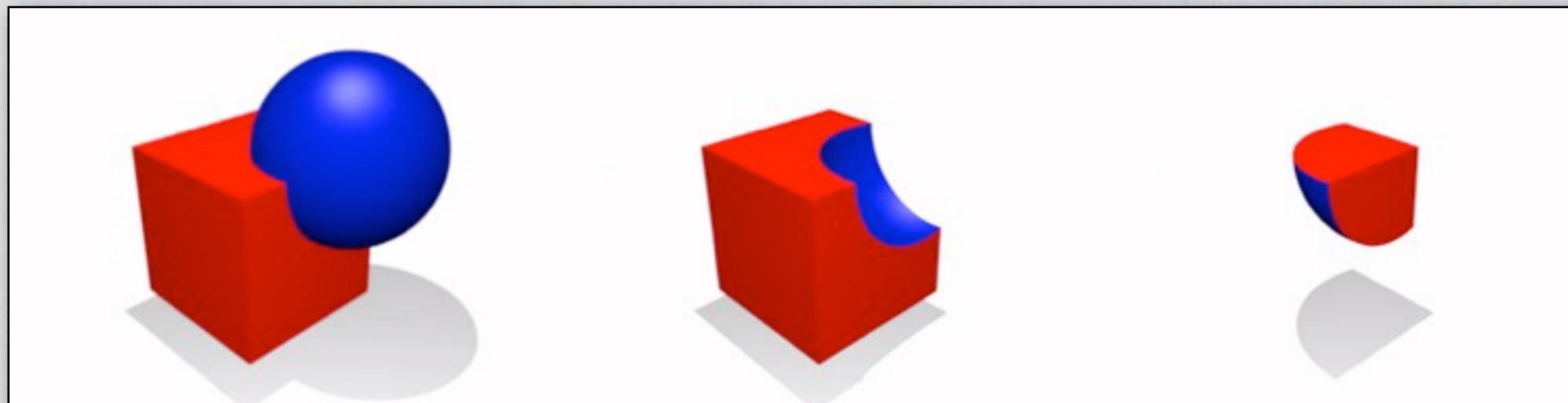
# Classes d'algorithmes

- \* Aparté sur la distance d'interpénétration ...
- \* ... cas d'un volume d'intersection
- \* convexe
- \* non-convexe



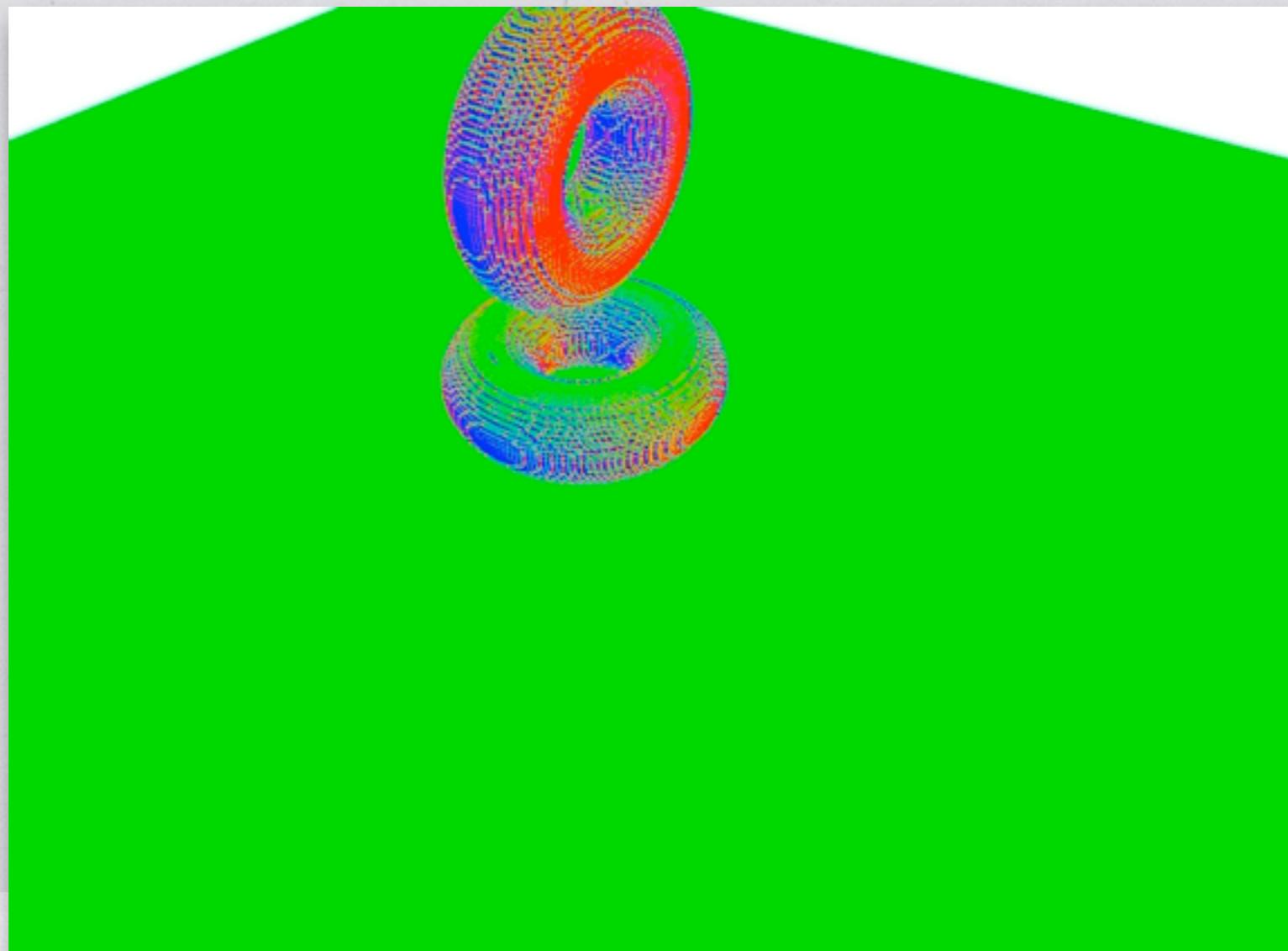
# Classes d'algorithmes

- \* Calcul de la forme du volume d'intersection (calcul formel très difficile, possibilité d'approximation numérique)



# Classes d'algorithmes

\* Exemple: approximation numérique sur GPU



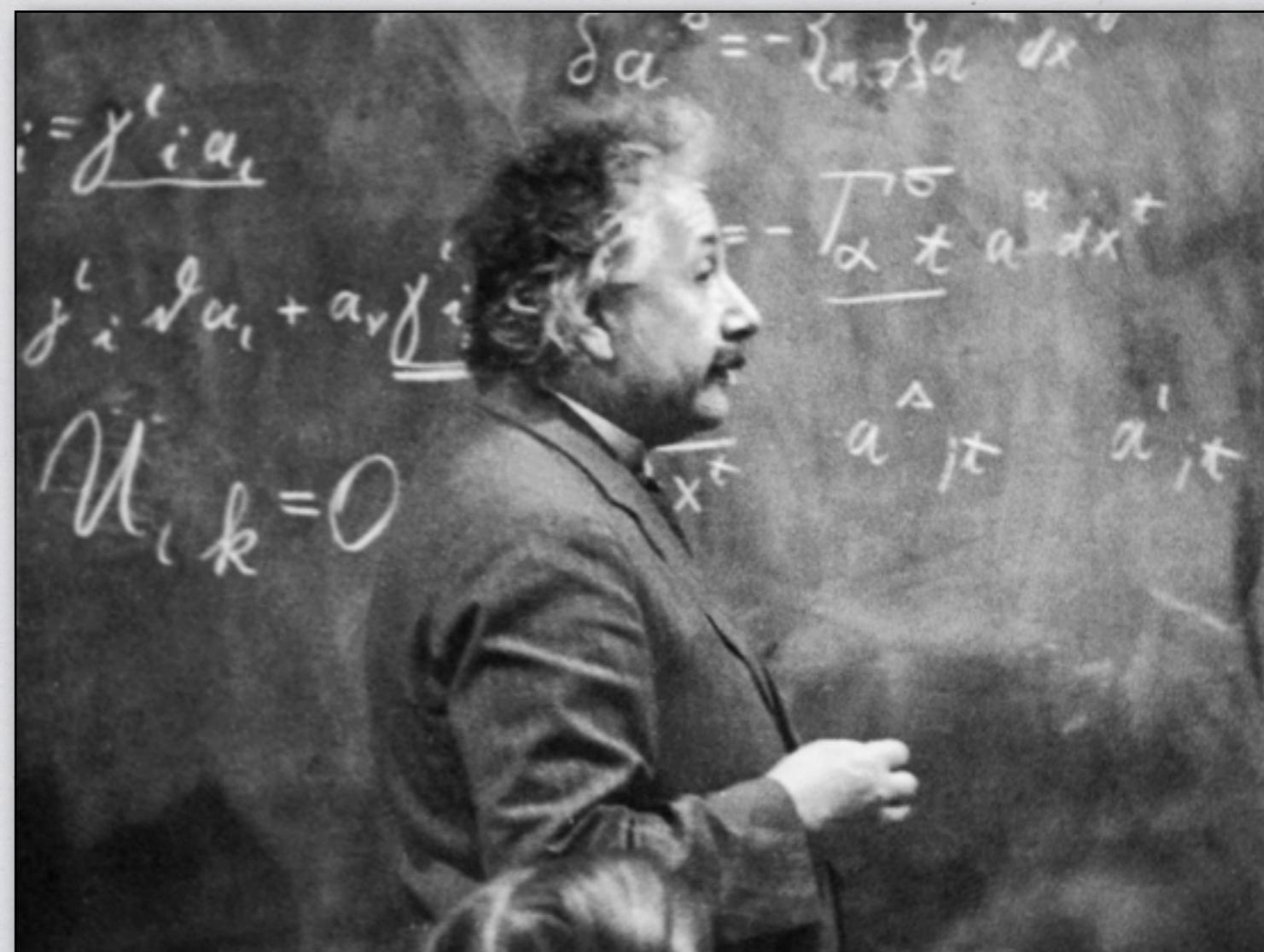
# Classes d'algorithmes

\* Exemple: utilisation du volume d'interpénétration

Aggregate Constraints for Virtual  
Manipulation with Soft Fingers

Anthony Talvas, Maud Marchal, Christian Duriez  
and Miguel A. Otaduy

### 3. Détection Spatio-Temporelle



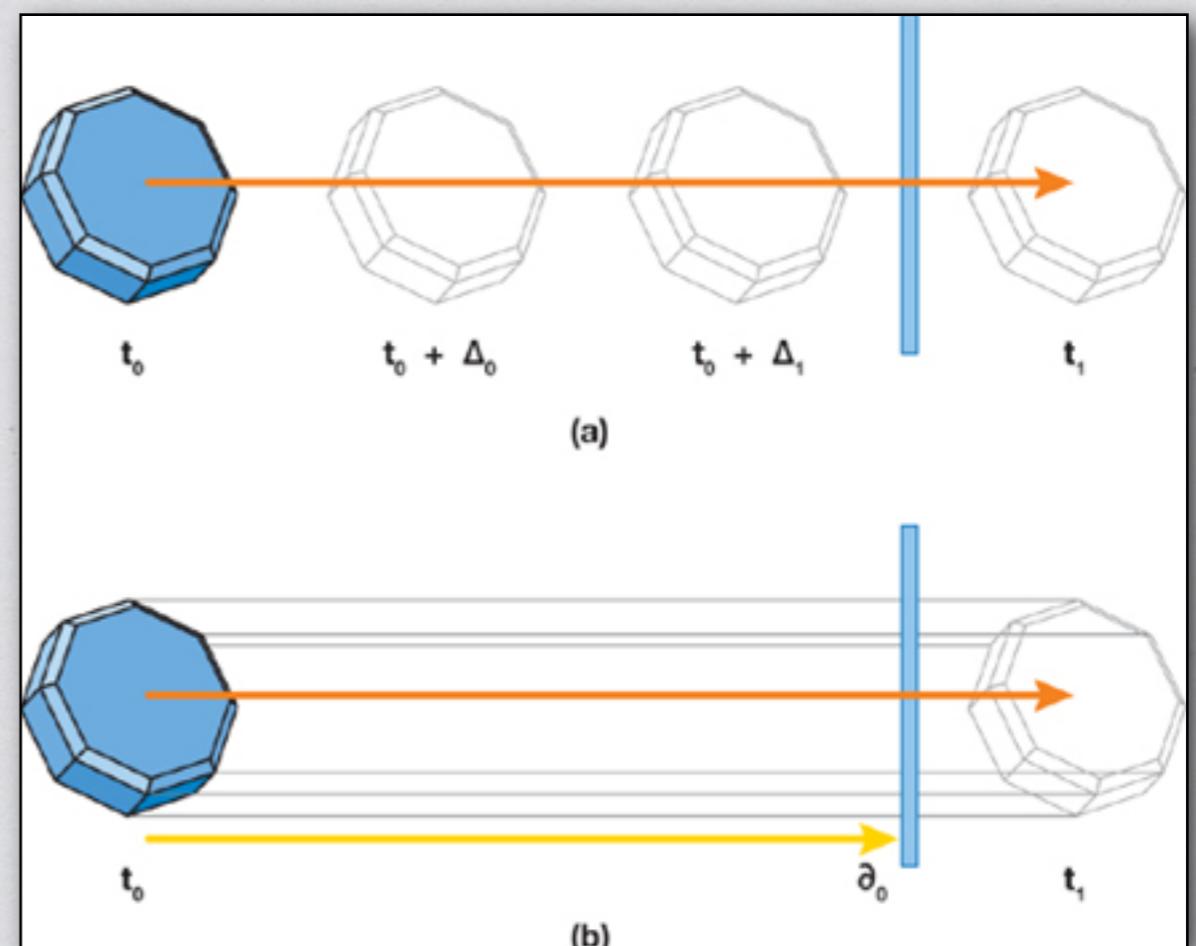
# Principe

\* Pour cette stratégie,  
l'inconnue à trouver est le  
temps (instant de contact)

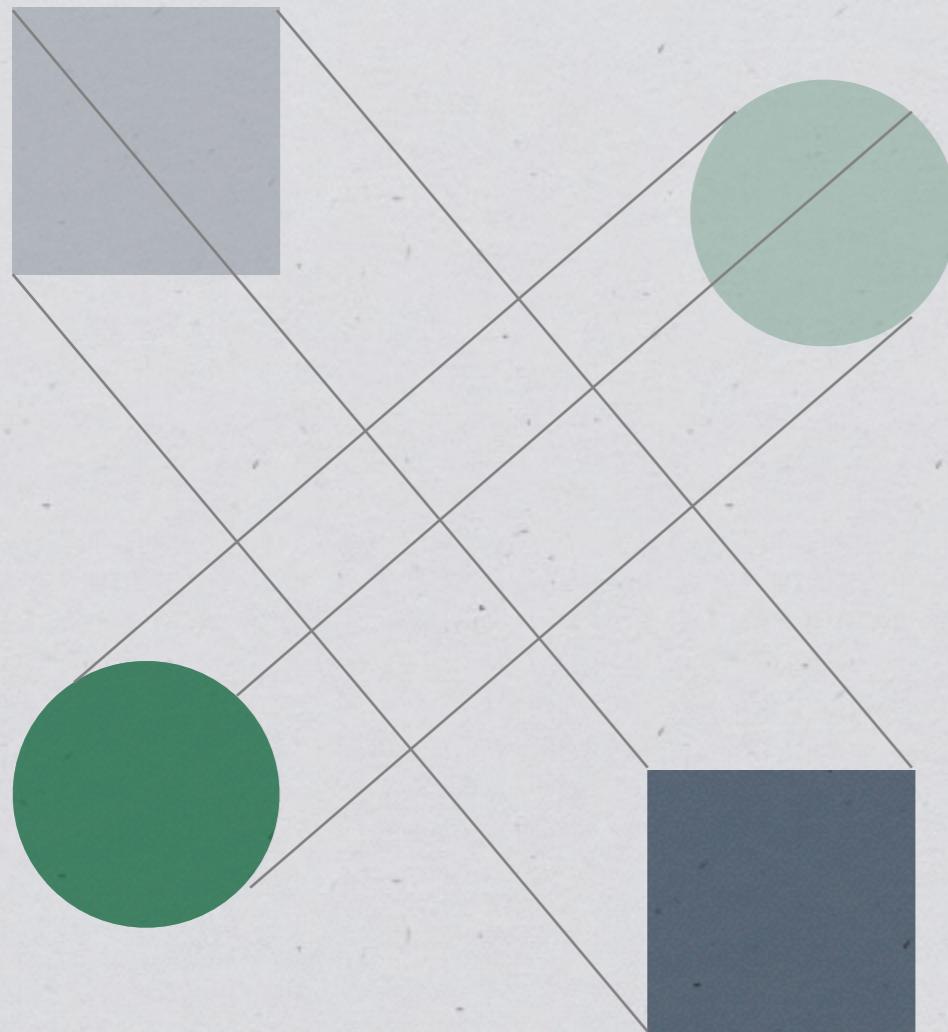


# Classes d'algorithmes

- \* A partir d'intersections 3D
- \* Détection 3D
- \* En cas de collisions,  
dichotomie sur  
l'intervalle  $\Delta t$  pour  
estimer l'instant de  
contact



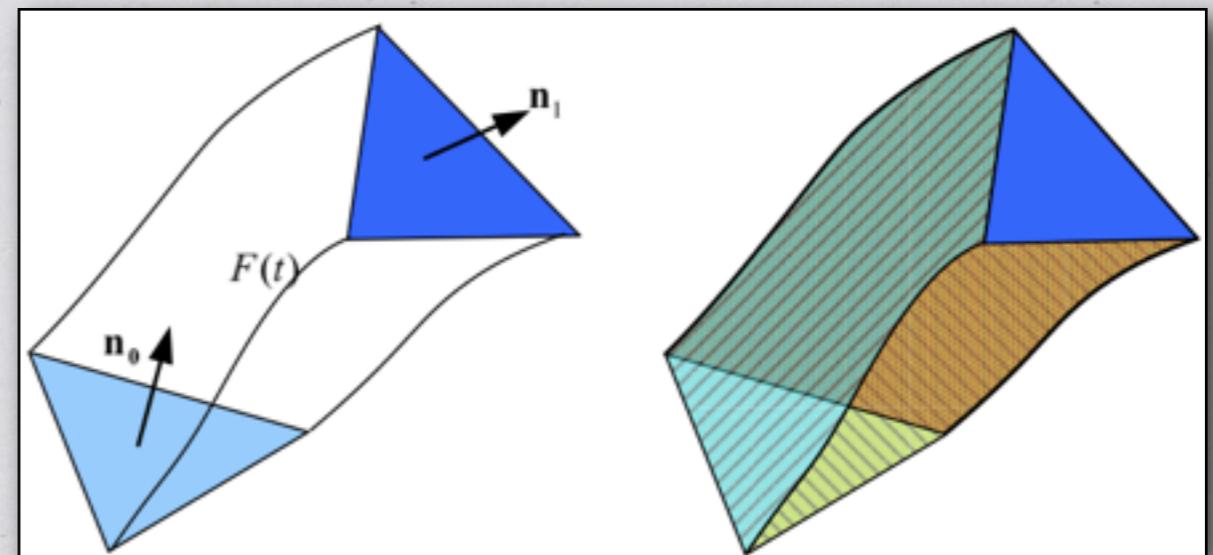
# Classes d'algorithmes



- \* Intersection de volumes balayés:
- \* définition d'un volume à partir des positions d'un objet à  $t$  et  $t + \Delta t$  ... puis intersection 3D
- \* Attention, s'il y a intersection des volumes balayés, il n'y a pas obligatoirement de collision

# Classes d'algorithmes

- \* Intersection de volumes extrudés selon le temps:
- \* volumes “4D”
- \* collision = intersection des projections des volumes sur un hyperplan 3D



# Classes d'algorithmes

- \* Méthodes analytiques:
- \* résolution d'un système d'équations dont le temps est la seule inconnue.
- \* Faisable pour des cas simples comme les primitives mais plus complexe sur un objet (surface implicite, etc..)

The notes show the derivation of a primitive function for the expression  $\frac{1}{\sin^2 x}$ . The steps include:

- Using trigonometric identities:
$$\sin^2 x = (1 - \cos^2 x) \sin x$$
$$\cos^2 x = (1 - \sin^2 x) \cos x$$
$$1 + \tan^2 x = \sec^2 x$$
- Integrating  $\frac{1}{\sin^2 x}$ :
$$\int \frac{1}{\sin^2 x} dx = \int \frac{1}{(1 - \cos^2 x) \sin x} dx$$
- Substituting  $u = \cos x$ ,  $du = -\sin x dx$ :
$$\int \frac{1}{(1 - u^2) u} du$$
- Simplifying the integral:
$$\int \frac{1}{u^2 - 1} du$$
- Using partial fraction decomposition:
$$\frac{1}{u^2 - 1} = \frac{1}{(u-1)(u+1)} = \frac{A}{u-1} + \frac{B}{u+1}$$
- Solving for  $A$  and  $B$ :
$$1 = A(u+1) + B(u-1)$$
$$1 = (A+B)u + (A-B)$$
$$A+B=0 \quad A-B=1$$
$$A=\frac{1}{2}, B=-\frac{1}{2}$$
- Integrating each term:
$$\int \frac{1}{u-1} du - \int \frac{1}{u+1} du$$
- Using substitution  $u = \cos x$ :
$$\int \frac{1}{\cos x - 1} dx - \int \frac{1}{\cos x + 1} dx$$
- Using the identity  $\cos x = \frac{1 - e^{i\omega t}}{2}$  and  $\cos x = \frac{e^{i\omega t} + 1}{2}$ :
- Final results:
$$\int \frac{1}{\sin^2 x} dx = \frac{1}{2} \ln \left| \frac{\cos x - 1}{\cos x + 1} \right| + C$$
$$\int \frac{1}{\sin^2 x} dx = \frac{1}{2} \ln \left| \frac{1 - e^{i\omega t}}{e^{i\omega t} + 1} \right| + C$$
$$\int \frac{1}{\sin^2 x} dx = \frac{1}{2} \ln \left| \frac{e^{-i\omega t} - 1}{e^{i\omega t} + 1} \right| + C$$

---

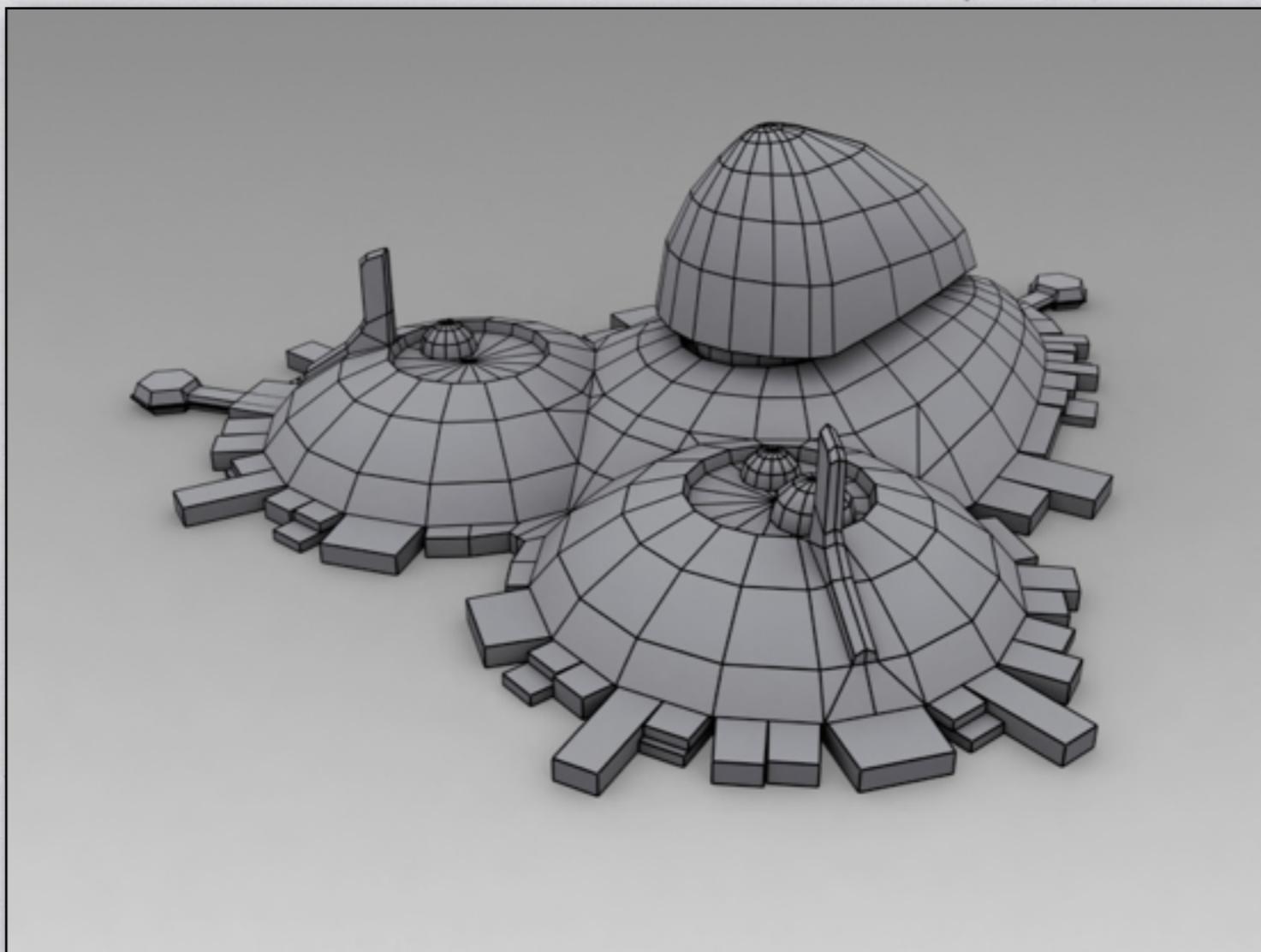
---

# 4. LES PRIMITIVES

Polyèdres, Surfaces Implicites / Paramétriques, Champs de Distances



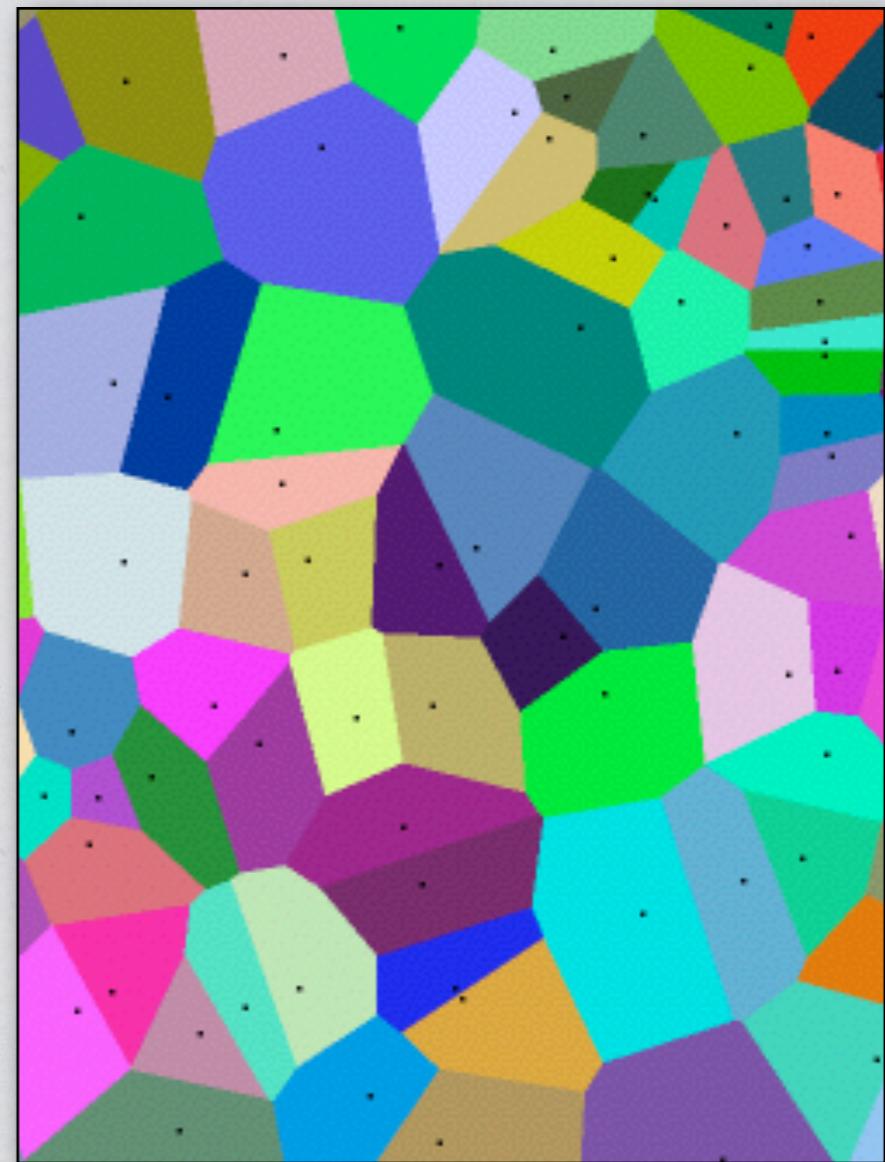
# Détection entre Polyèdres



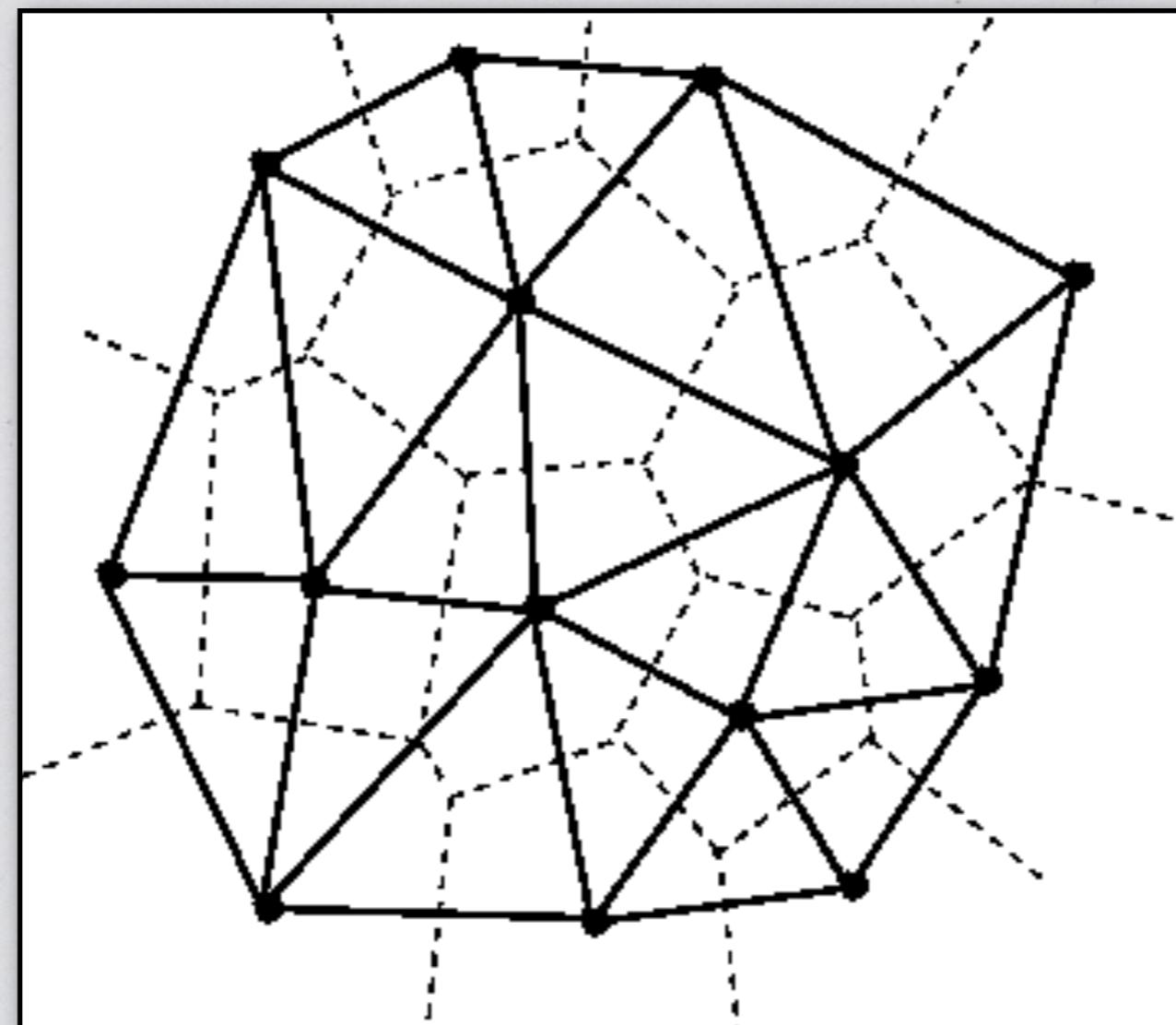
# A. Calcul de Distance

- \* 1. Voronoi Marching
- \* Rappels: diagramme de Voronoi
- \* Soient  $E$  un espace euclidien,  $S$  un ensemble fini de primitives de  $E$ , on définit  $V$  comme étant:

$$V_s(p) = \{x \in E / \forall q \in S, d(x, p) \leq d(x, q)\}$$



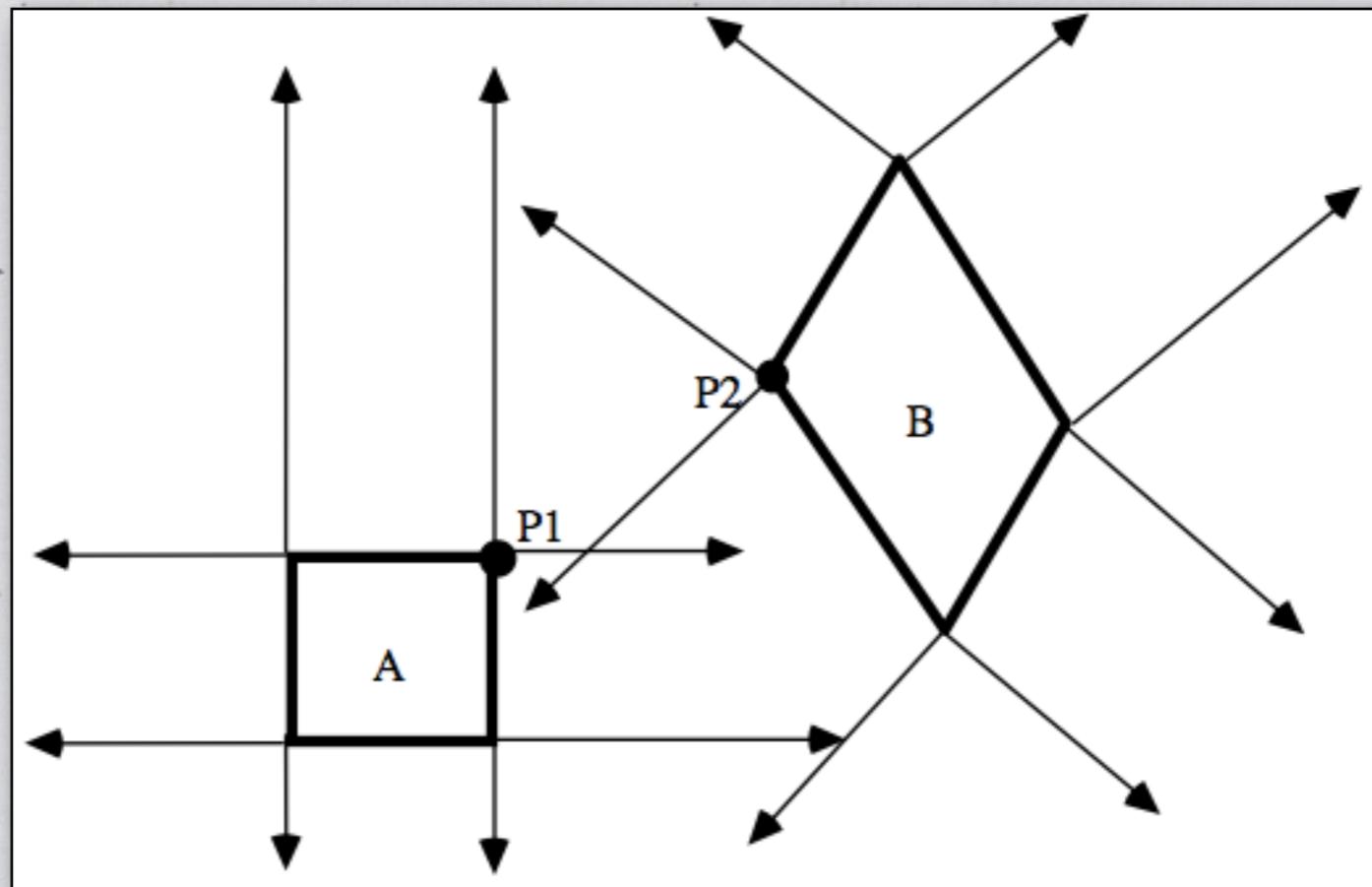
# A. Calcul de Distance



# A. Calcul de Distance

\* 1. Voronoi Marching: Lin & Canny 1991

\* Principe:



# A. Calcul de Distance

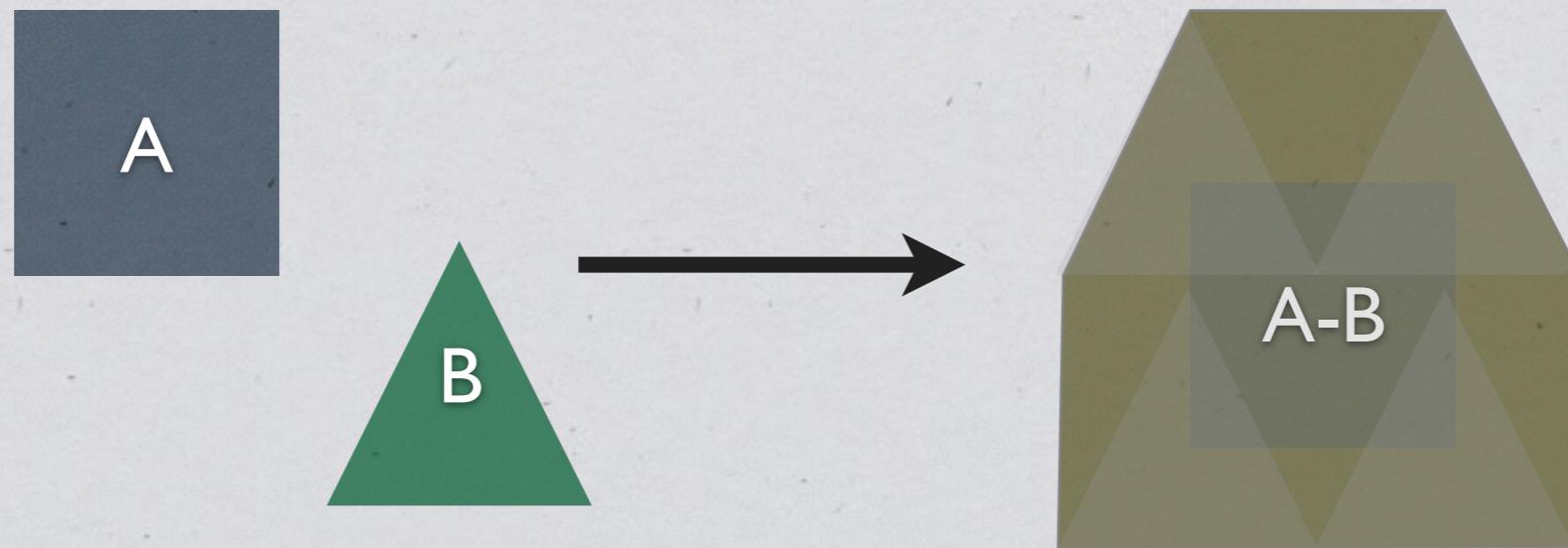
- \* 1. Voronoi Marching: Lin & Canny 1991
- \* Propriétés:
  - \* Localité géométrique
  - \* Cohérence temporelle (possibilité de temps constant)
  - \* Rapide sur du rigide (moins sur du déformable)

# A. Calcul de Distance

\* 2. Algorithme GJK

\* Utilisation de la somme de Minkowski (en fait la différence  $\odot$ )

$$A - B = \{(a - b) / a \in A, b \in B\}$$

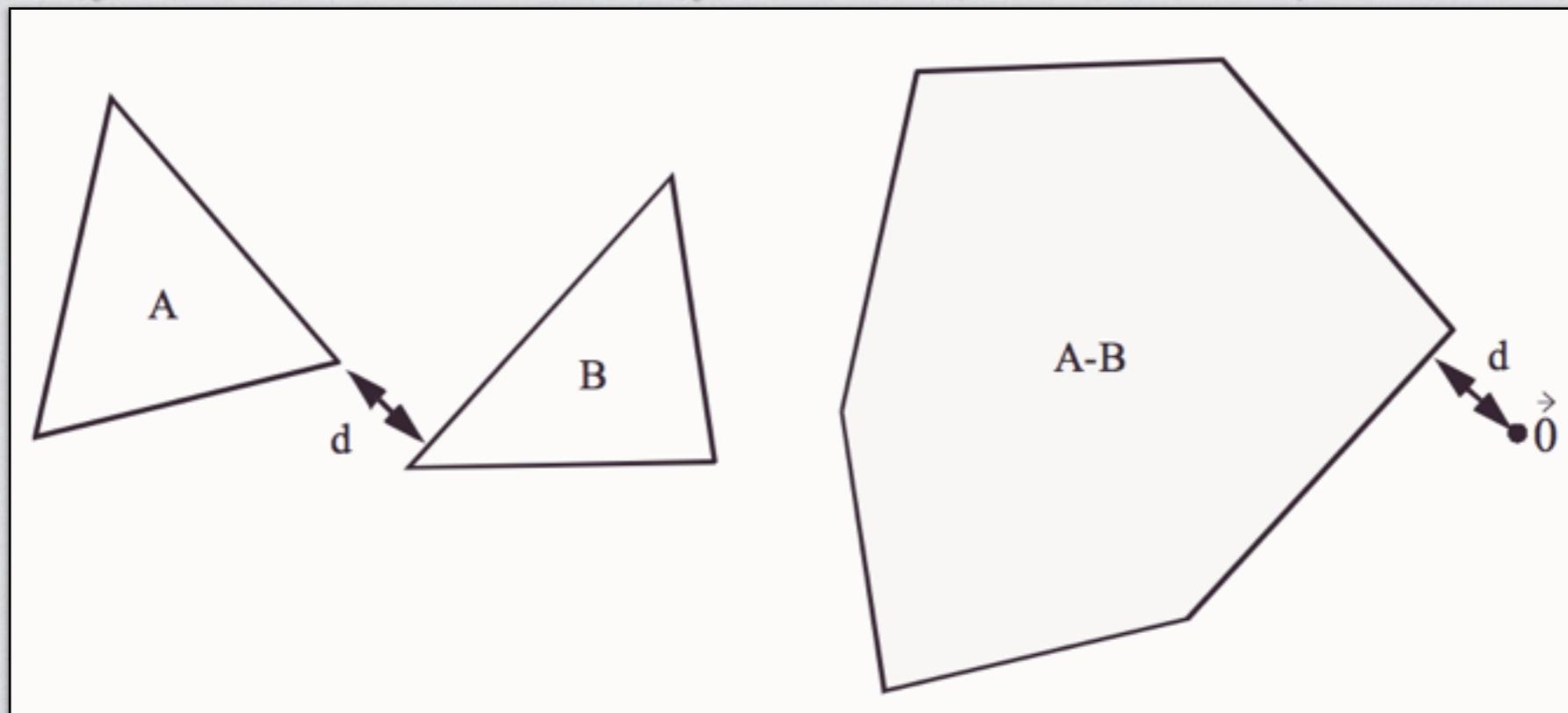


# A. Calcul de Distance

- \* 2. Algorithme GJK
- \* Une somme de Minkowski est convexe ssi A et B sont convexes
- \* Intérêt de la somme de Minkowski: passer d'un calcul de distance entre deux polyèdres à un calcul entre un polyèdre et un point

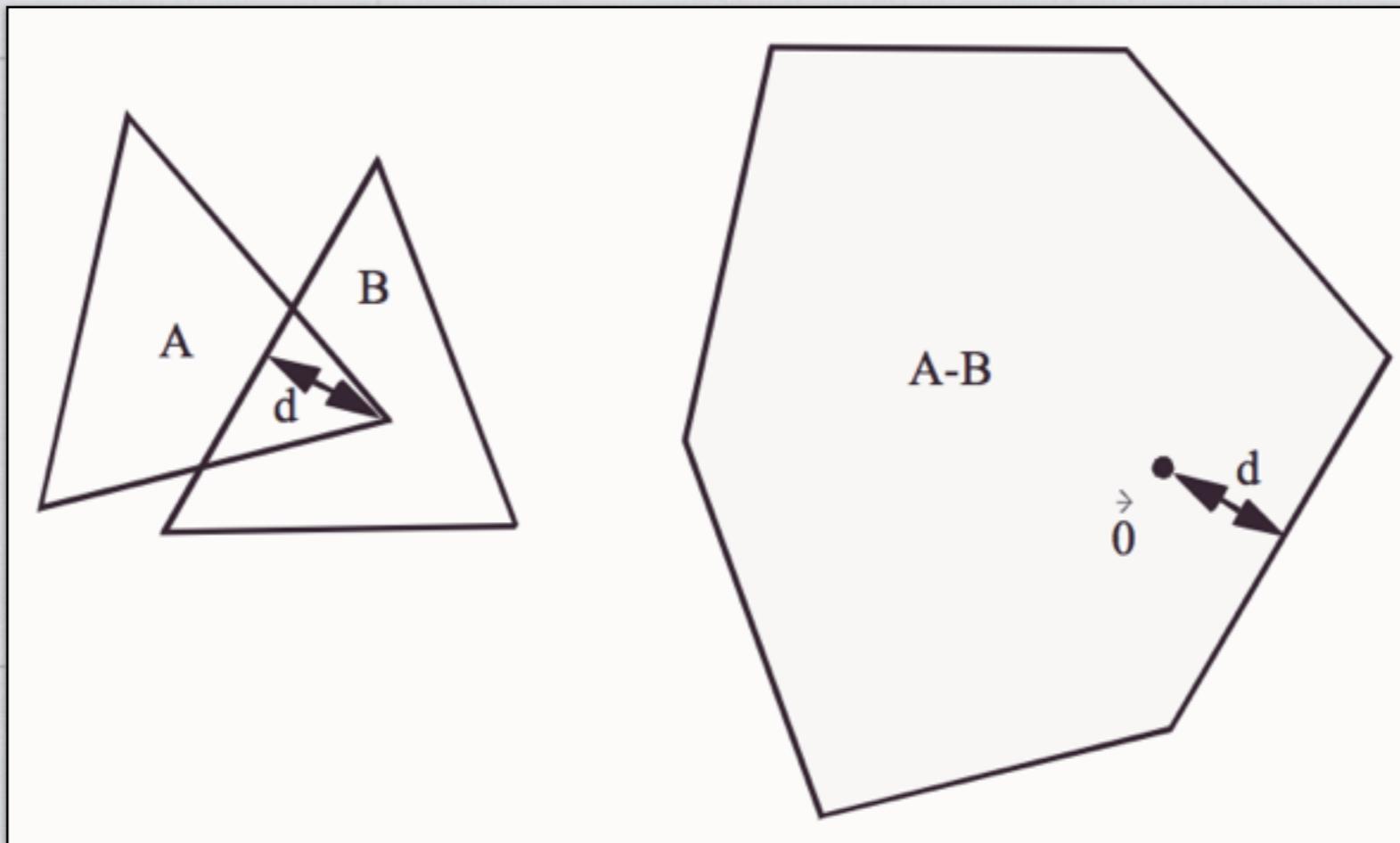
# A. Calcul de Distance

## \* 2. Algorithme GJK



# A. Calcul de Distance

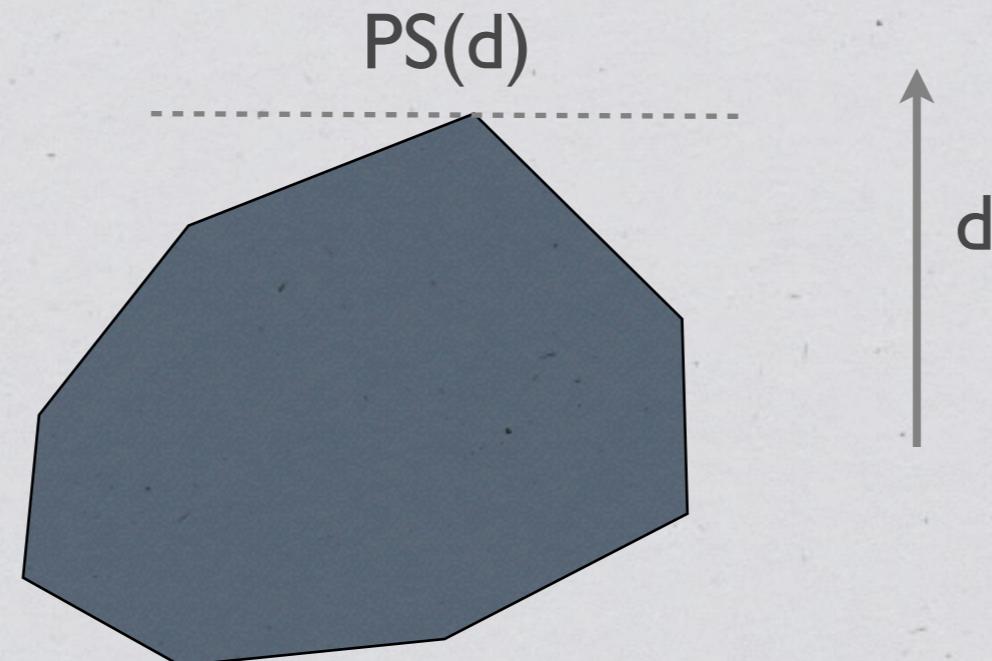
## \* 2. Algorithme GJK



# A. Calcul de Distance

\* 2. Algorithme GJK

\* Un peu de mathématiques: point de support

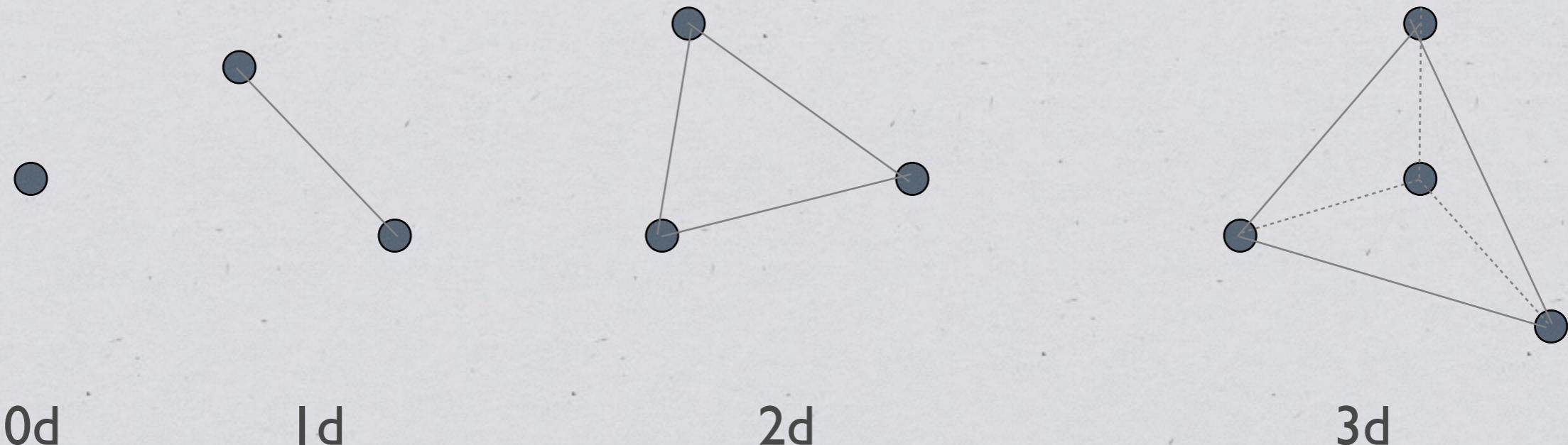


$$PS(d) = x / \forall p \in P, p \cdot d \leq x \cdot d$$

# A. Calcul de Distance

\* 2. Algorithme GJK

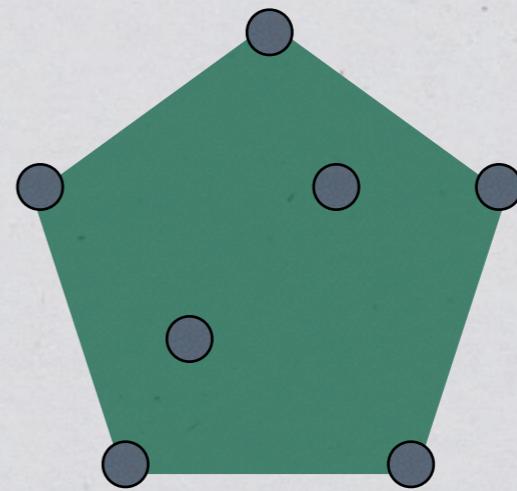
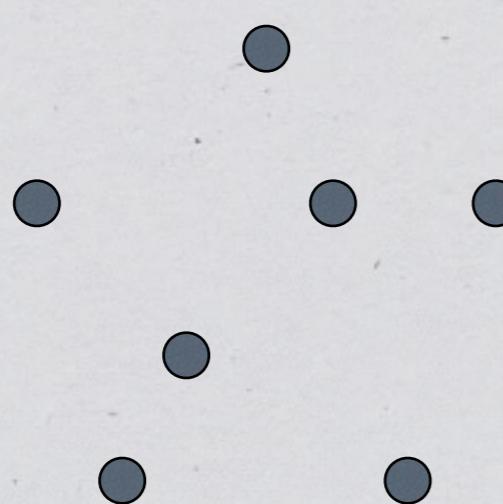
\* Un peu de mathématiques: simplexe (ensemble de  $n+1$  points pour former un espace affine)



# A. Calcul de Distance

\* 2. Algorithme GJK

\* Un peu de mathématiques: enveloppe convexe



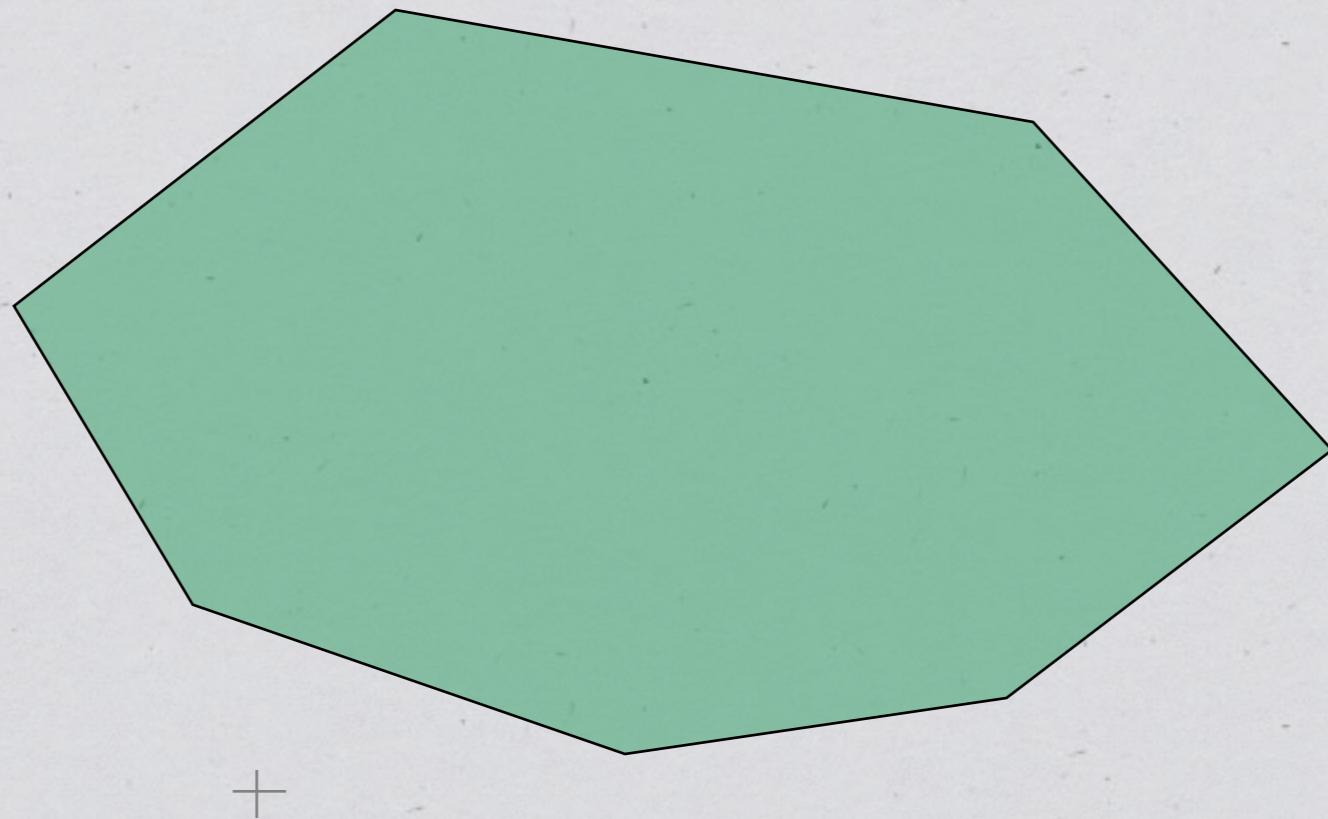
# A. Calcul de Distance

## \* 2. Algorithme GJK

1. Initialise un simplexe Q
2. Calculer le point P de norme minimale dans EC(Q)
3. Si P = Origine, retourner 0
4. Réduire Q au plus petit sous-ensemble Q' de Q tel que P appartient à EC(Q')
5. Calcul V, comme étant  $PS(-P)$
6. Si V n'est pas tel que  $V \cdot (-P) \geq P \cdot (-P)$  retourner  $\|P\|$
7. Ajouter V à Q et reboucler sur l'étape 2

# A. Calcul de Distance

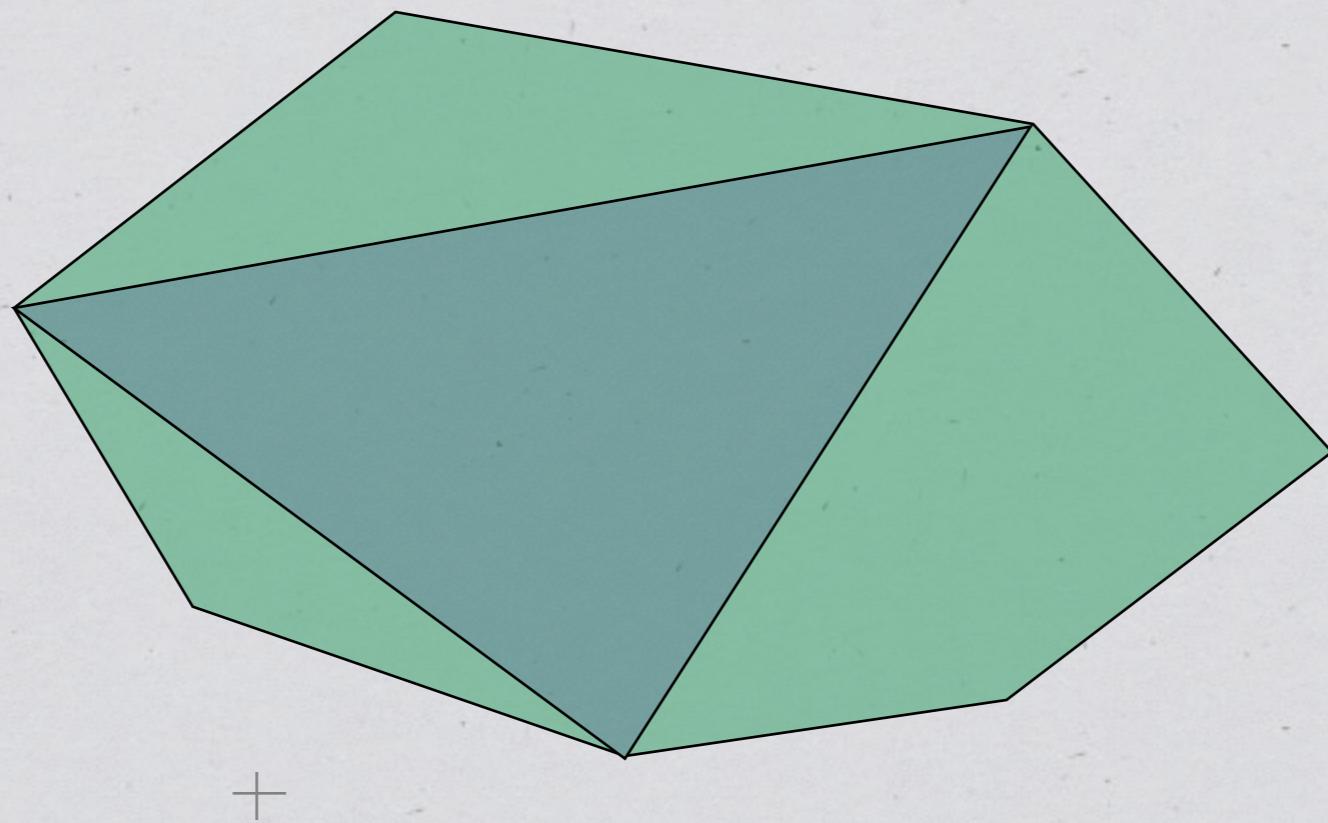
\* 2. Algorithme GJK: exemple



# A. Calcul de Distance

\* 2. Algorithme GJK: exemple

$$Q=\{0, 2, 5\}$$

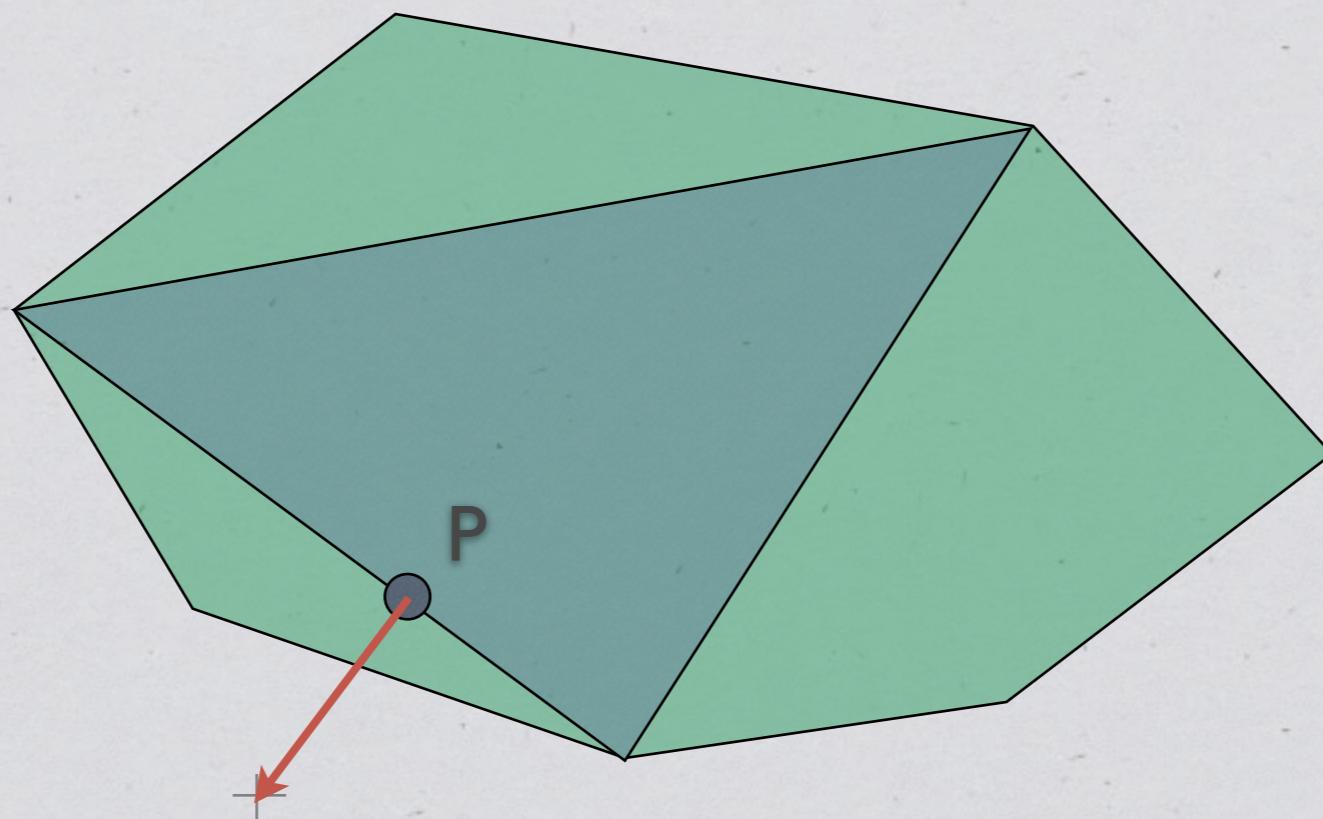


1. Initialise un simplexe Q

# A. Calcul de Distance

## \* 2. Algorithme GJK: exemple

$$Q = \{0, 2, 5\}$$

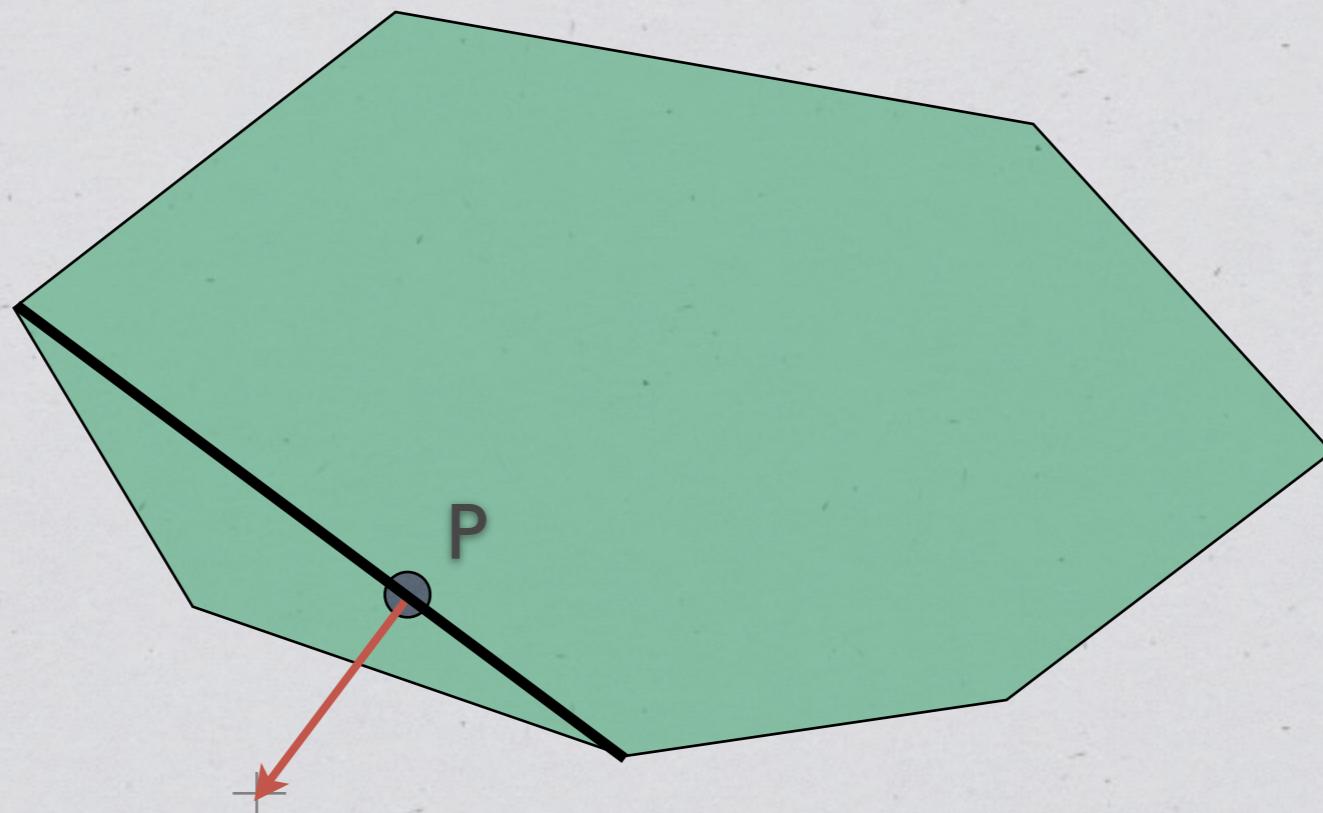


2. Calculer le point P de norme minimale dans  $EC(Q)$

# A. Calcul de Distance

## \* 2. Algorithme GJK: exemple

$Q = \{0, 5\}$

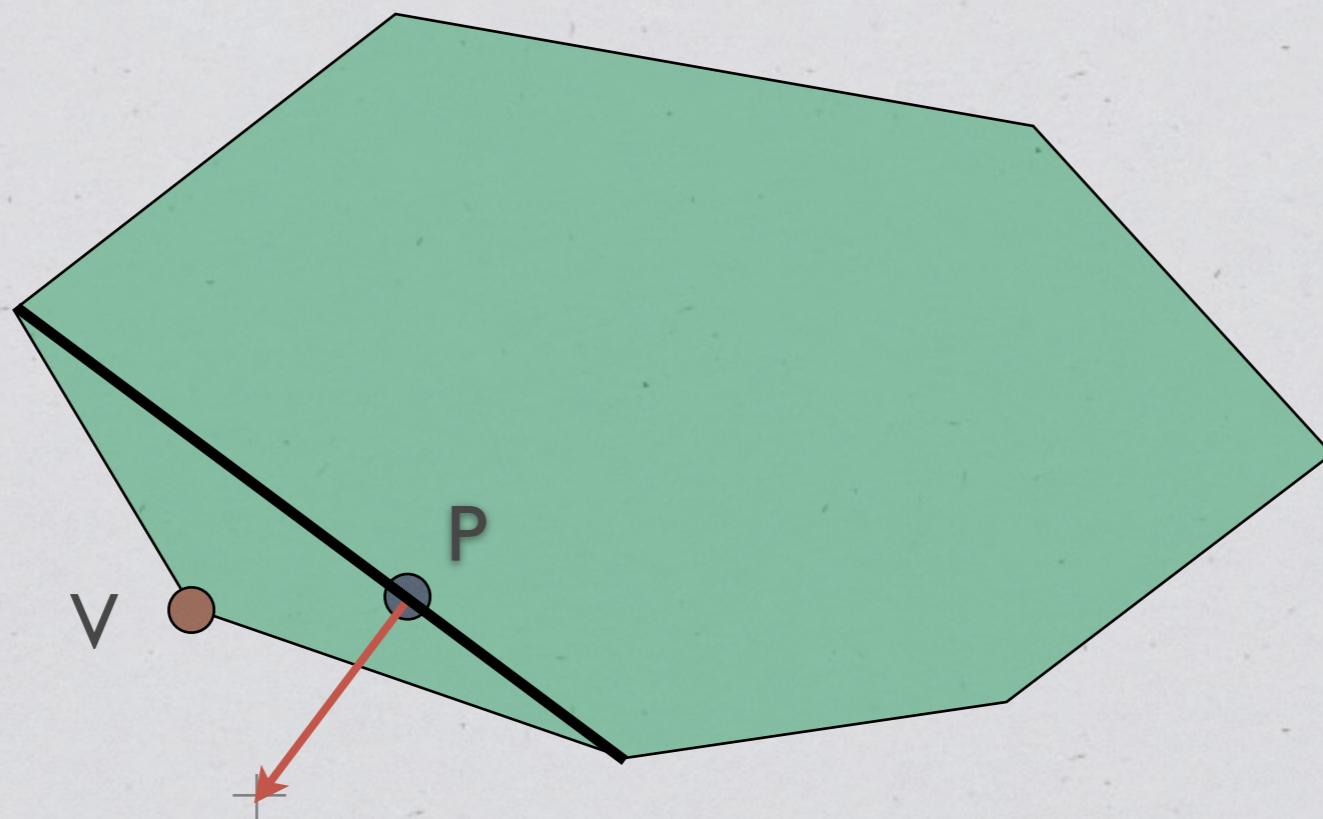


3. Si  $P = \text{Origine}$ , retourner 0
4. Réduire  $Q$  au plus petit sous-ensemble  $Q'$  de  $Q$  tel que  $P$  appartient à  $\text{EC}(Q')$

# A. Calcul de Distance

\* 2. Algorithme GJK: exemple

$$Q=\{0, 5\}$$

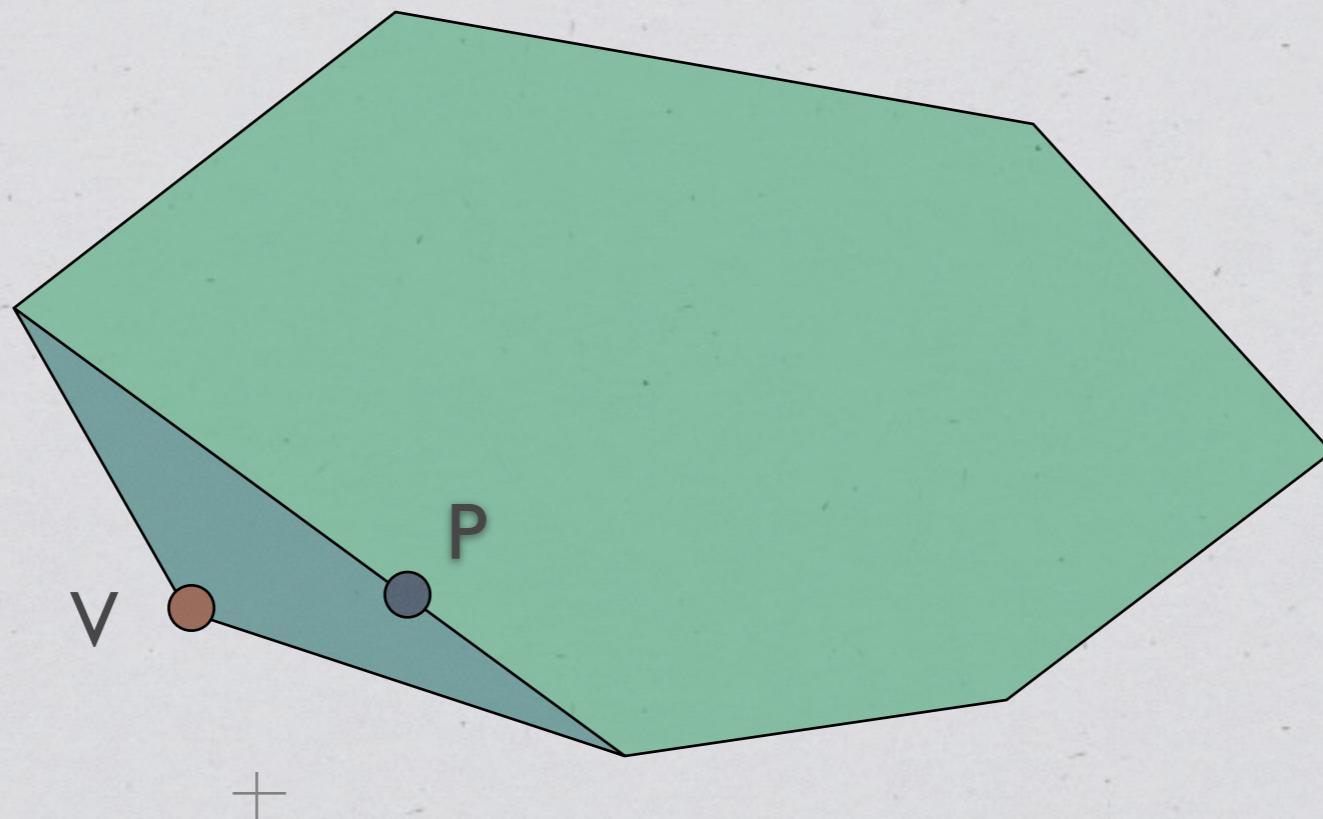


5. Calcul  $V$ , comme étant  $PS(-P)$

# A. Calcul de Distance

## \* 2. Algorithme GJK: exemple

$Q = \{0, 5, V=6\}$

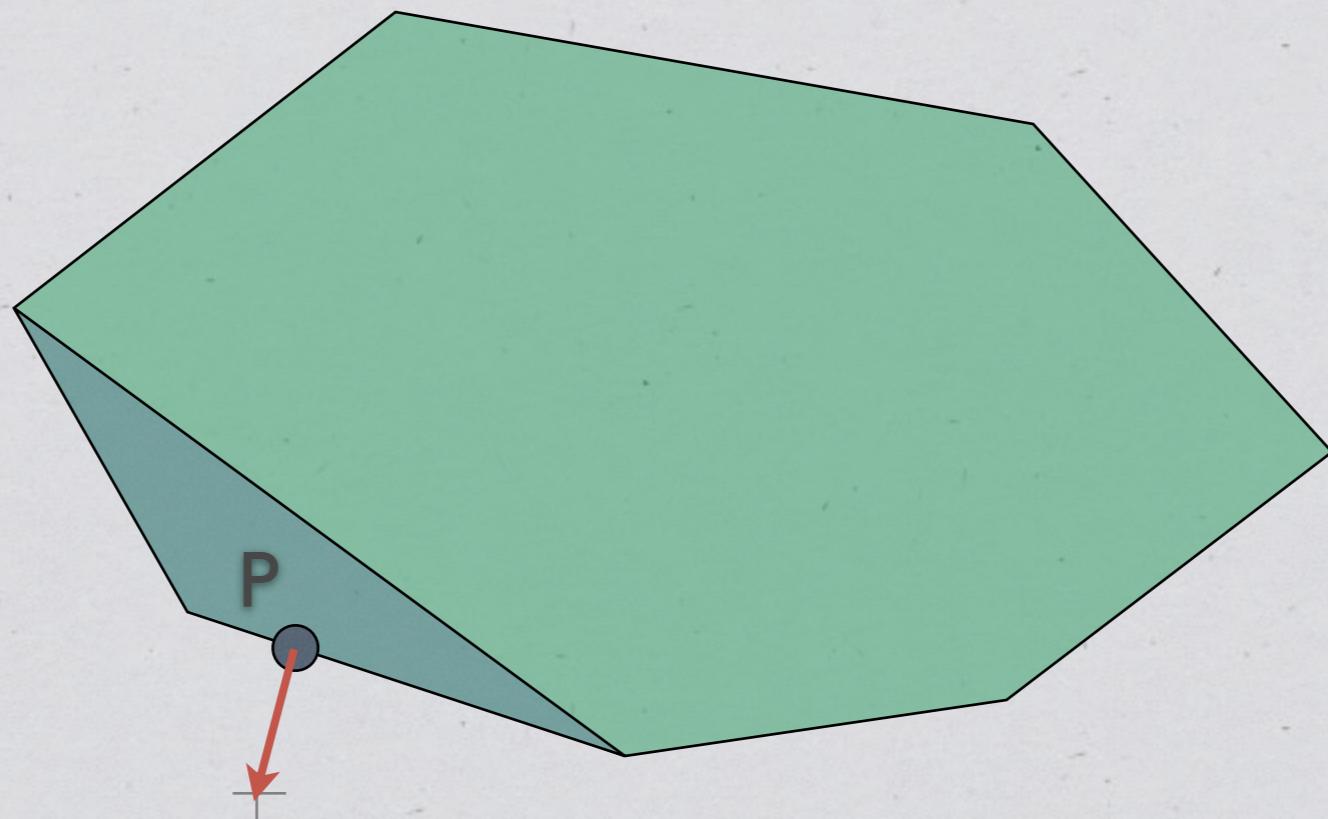


6. Si  $V$  n'est pas tel que  $V \cdot (-P) \geq P \cdot (-P)$  retourner  $\|P\|$
7. Ajouter  $V$  à  $Q$  et reboucler sur l'étape 2

# A. Calcul de Distance

\* 2. Algorithme GJK: exemple

$$Q = \{0, 5, 6\}$$

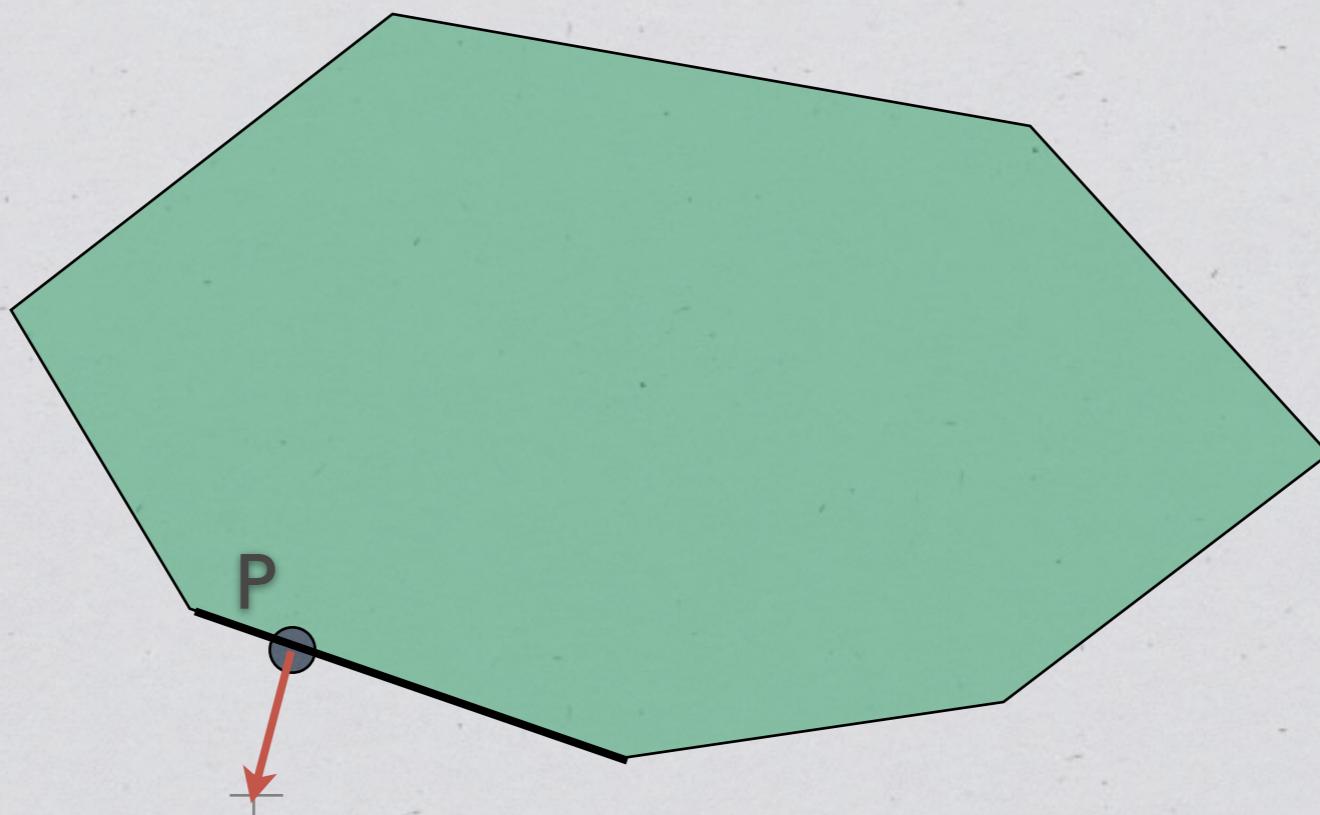


2. Calculer le point P de norme minimale dans  $EC(Q)$

# A. Calcul de Distance

## \* 2. Algorithme GJK: exemple

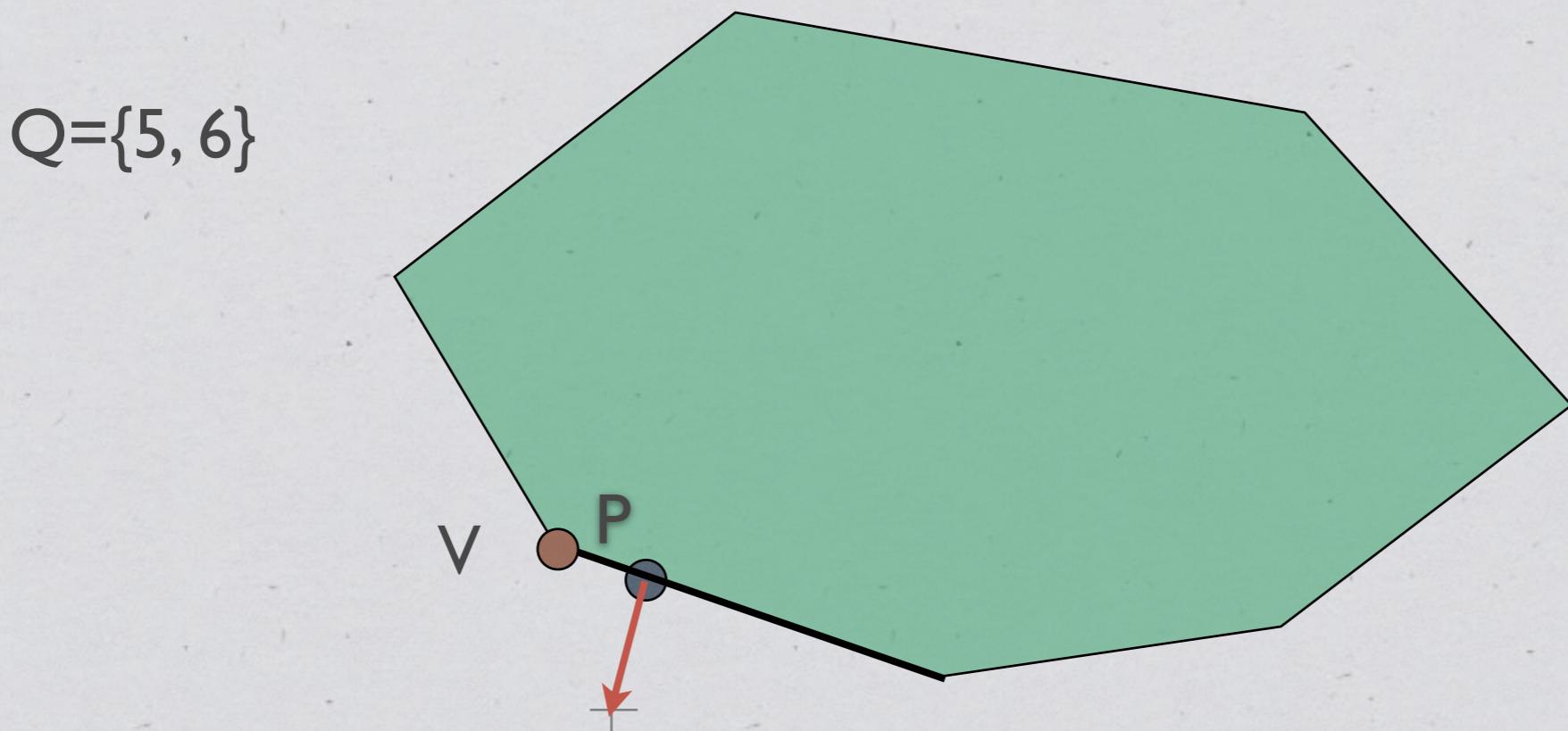
$Q = \{5, 6\}$



3. Si  $P = \text{Origine}$ , retourner 0
4. Réduire  $Q$  au plus petit sous-ensemble  $Q'$  de  $Q$  tel que  $P$  appartient à  $\text{EC}(Q')$

# A. Calcul de Distance

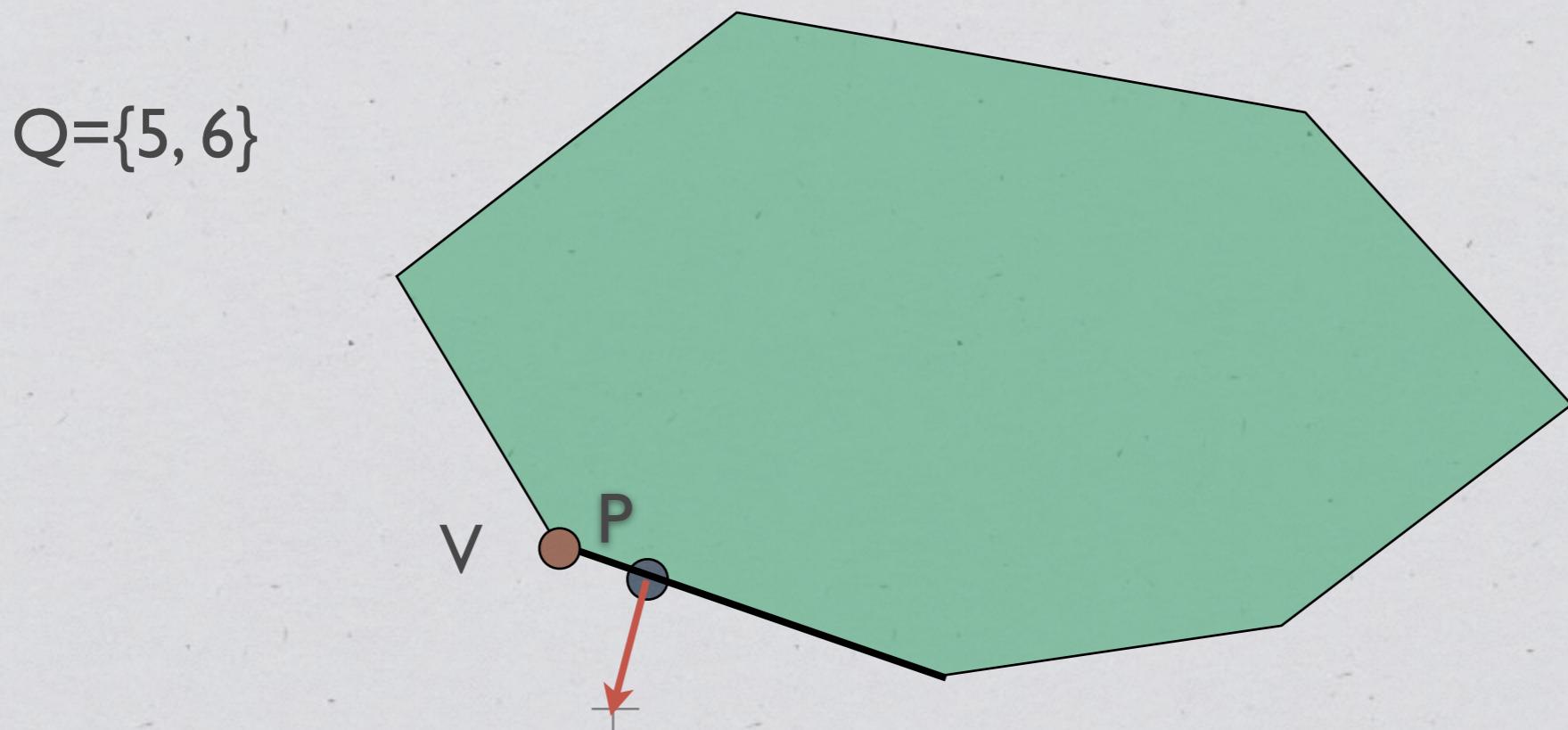
\* 2. Algorithme GJK: exemple



5. Calcul  $V$ , comme étant  $PS(-P)$

# A. Calcul de Distance

\* 2. Algorithme GJK: exemple



6. Si  $V$  n'est pas tel que  $V \cdot (-P) \geq P \cdot (-P)$  retourner  $\|P\|$

# A. Calcul de Distance

\* 2. Algorithme GJK

\* Propriétés:

- \* complexité linéaire (en fonction du nombre de sommets)
- \* utilisation de la cohérence temporelle (pour calcul en temps constant)
- \* point de support facilement calculable pour les “polyèdres classiques”

# B. Recherche d'un plan séparateur

\* Définition:



\* Pour la somme de Minkowski, il suffit de déterminer si l'origine appartient à  $A-B$  (ce qui peut se faire avec le GJK)

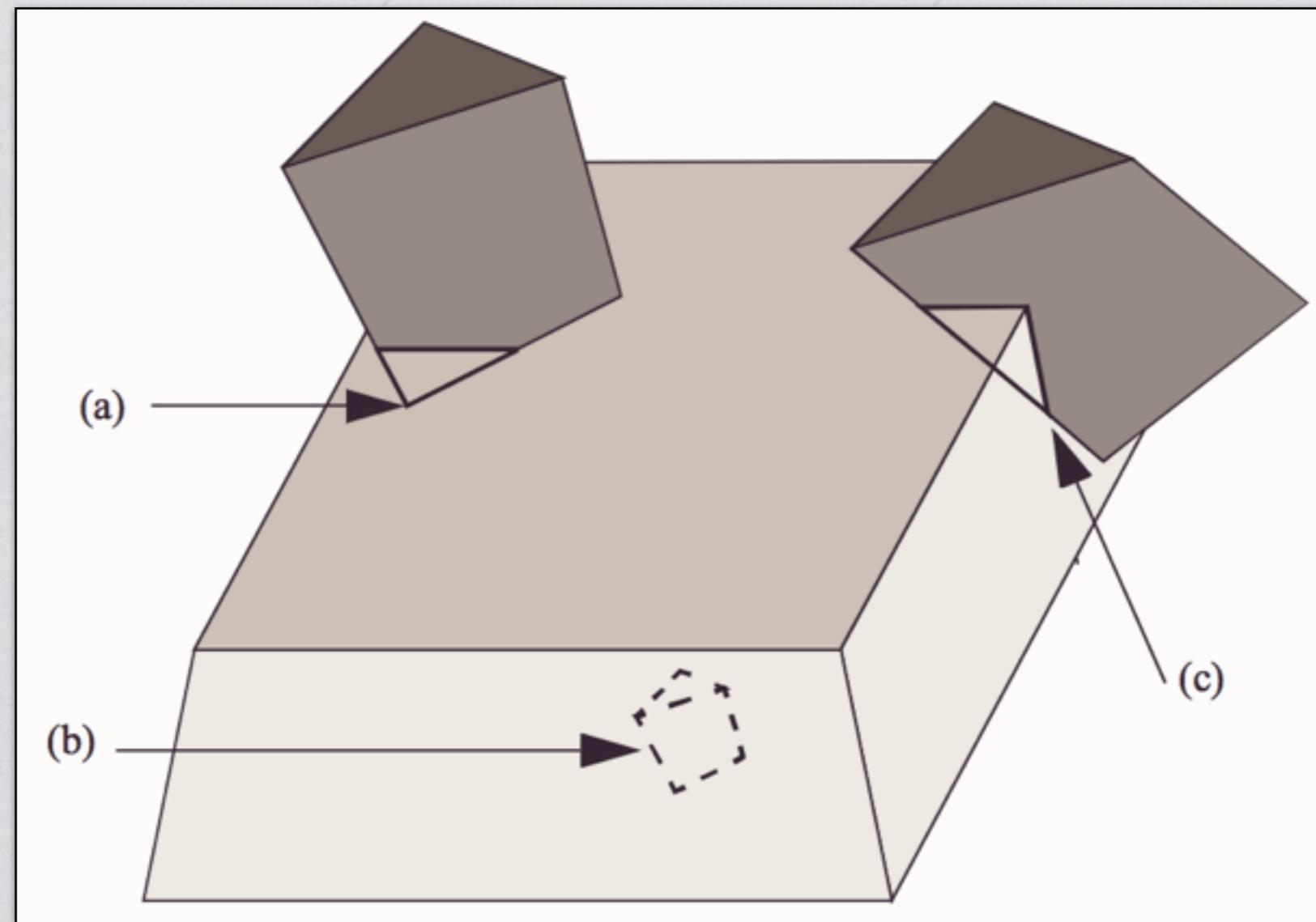
# Premier Bilan

- \* Complexité linéaire ( $\cong$  au nombre de sommets)
- \* Utilisation de la cohérence temporelle pour converger plus rapidement
- \* Cependant:
  - \* Incompatible avec des objets déformables
  - \* objets convexes → sinon décomposition en parties convexes  
(pb NP-dur)

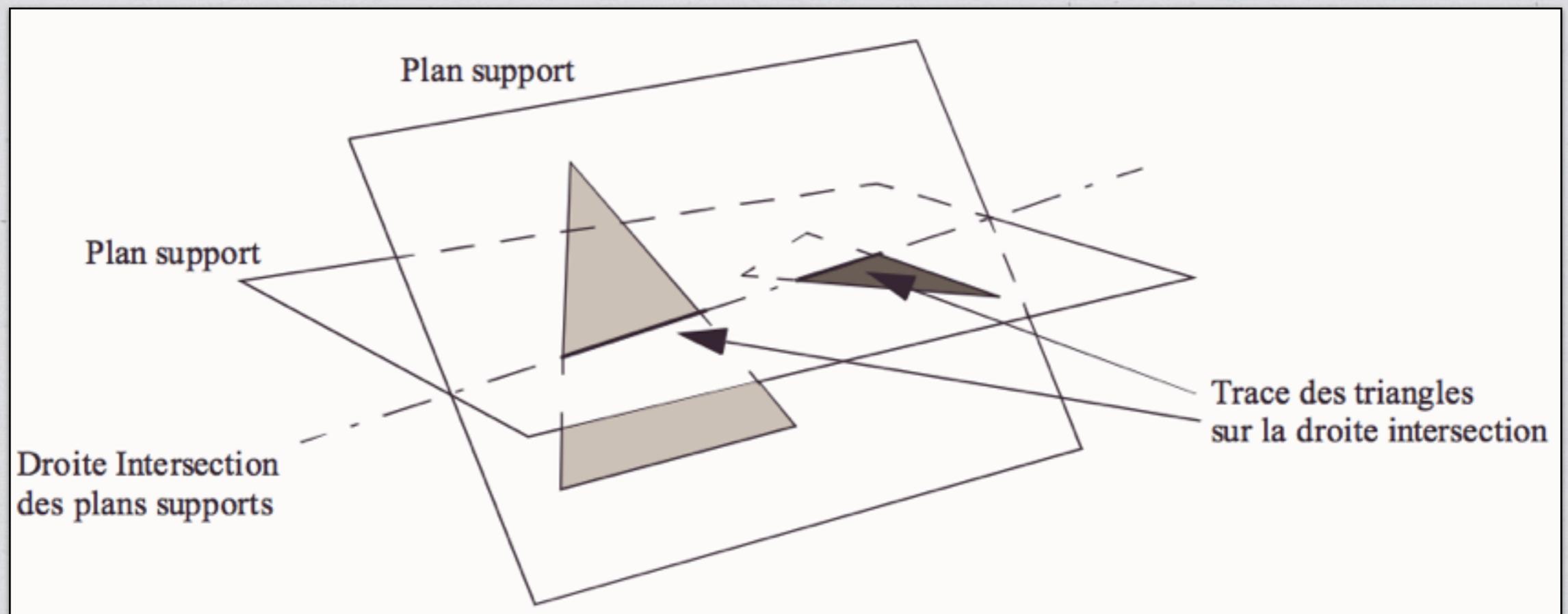
# C. Détection des intersections

- \* L'intersection de deux polyèdres est non vide ssi une de ces deux conditions est vraie:
  - \* un sommet de l'un appartient à l'autre (VF test)
  - \* une arête de l'un intersecte une face de l'autre (EE test)
- \* Complexité Quadratique

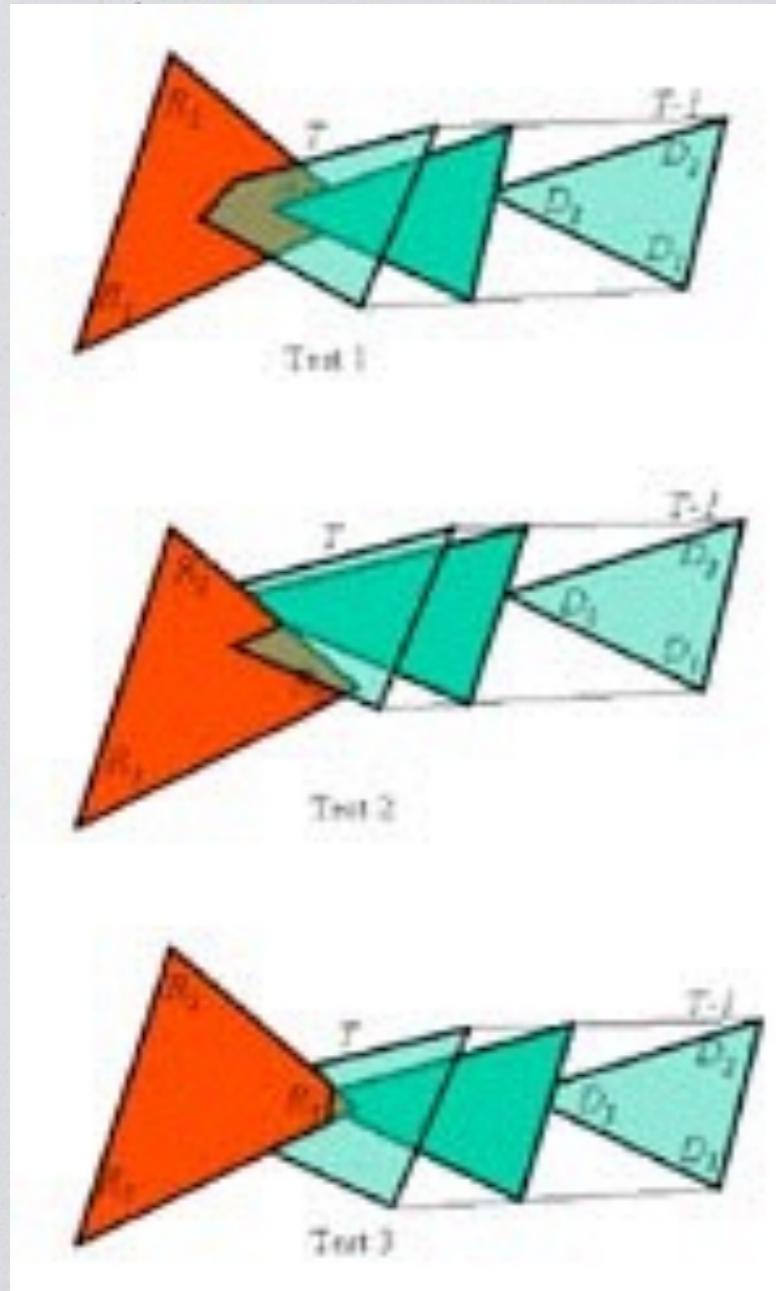
# C. Détection des intersections



# C. Détection des intersections

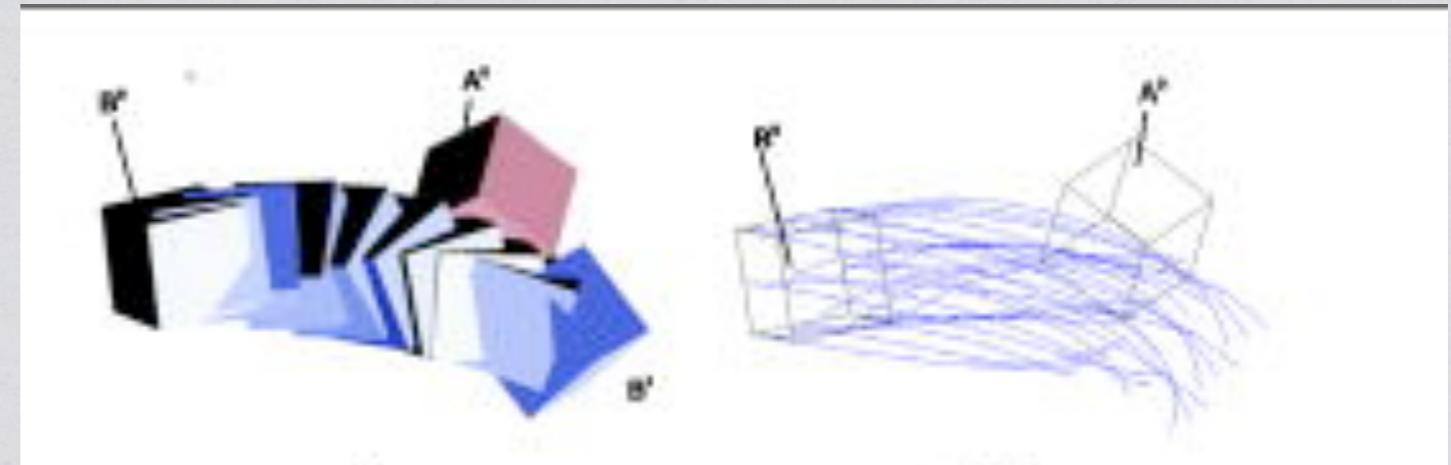
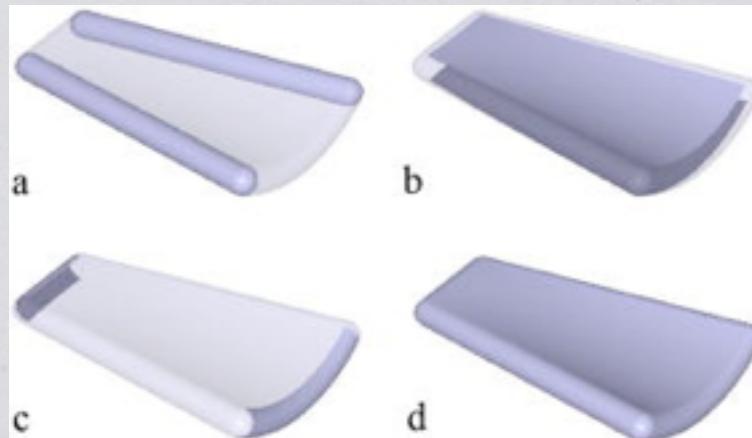


# F. Calcul analytique des contacts



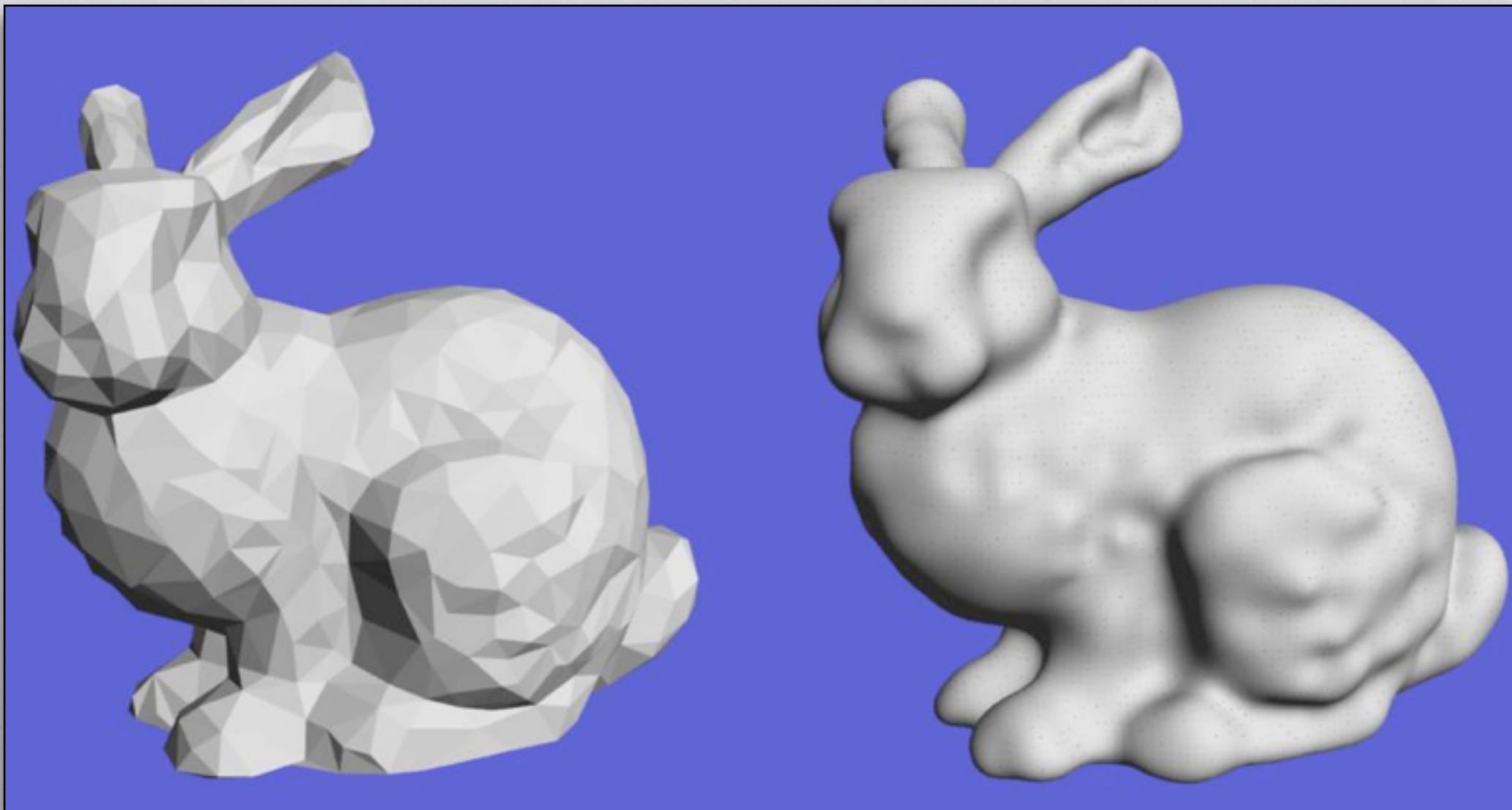
- \* Détection “continue”: mouvement des primitives approximé de manière linéaire
- \* Résolution de polynôme d’ordre 3 pour retrouver l’instant du contact

# F. Calcul analytique des contacts



- \* Détection “continue”: mouvement des objets (rigides) est approximé par un vissage (translation + rotation)
- \* Utilisation d’arithmétique par intervalles (voir Redon 2002) sur un intervalle temporel

# Détection entre Surfaces Implicites



# Intérêt ?

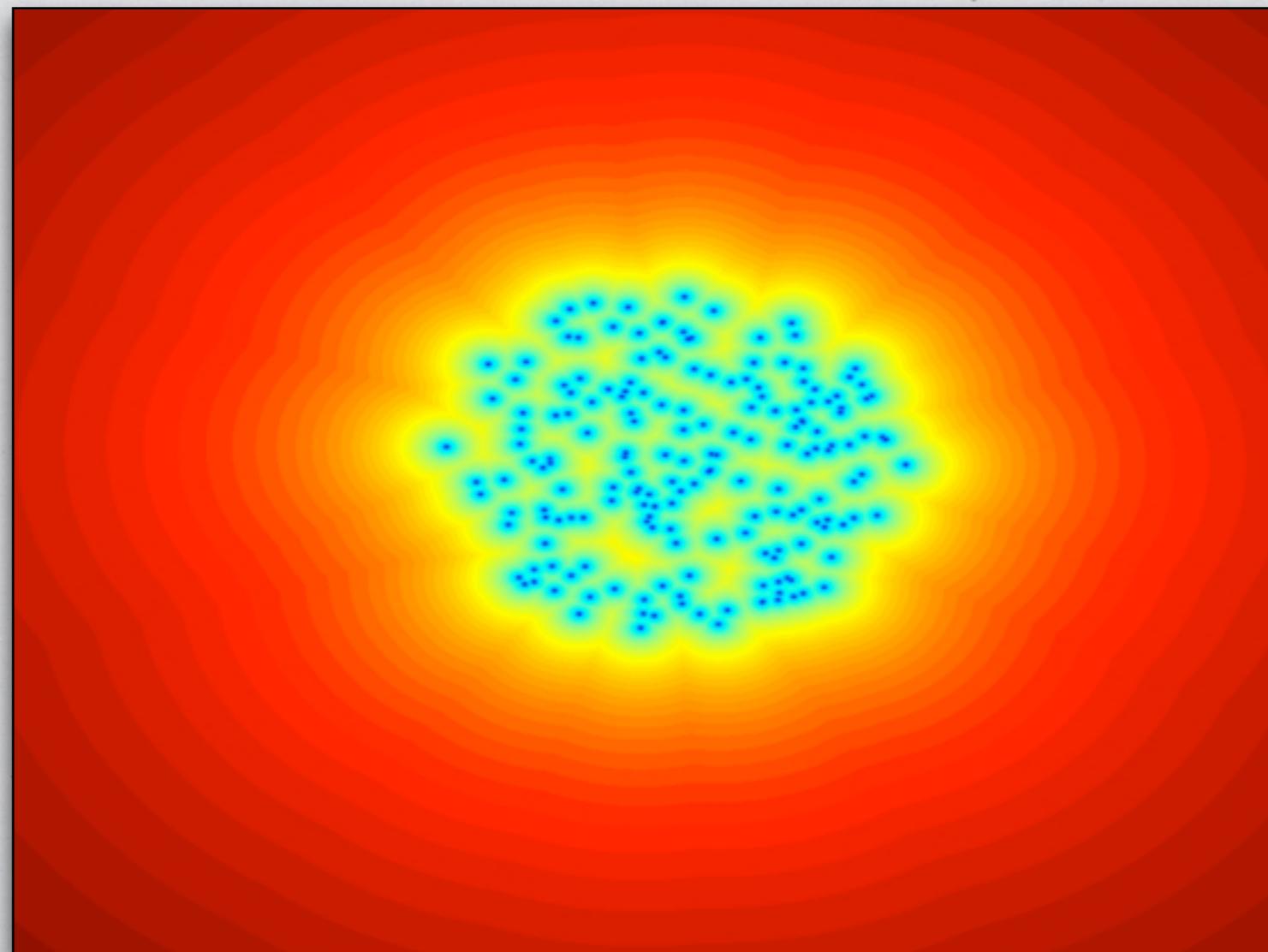
- \* Pour un point donné, très facile de savoir s'il est en dehors / dedans / en contact d'une surface implicite donnée (cf évaluation du signe du potentiel)
- \* Détection d'intersection vide pour deux surfaces  $f, g$ , recherche du maximum de la fonction  $h$ :
$$h(M) = f(M) + g(M) - \sqrt{f^2(M) + g^2(M)}$$
- \* si  $h(M)$  est positif, cela signifie que  $f(M)$  et  $g(M)$  le sont aussi

# Calcul de Distance

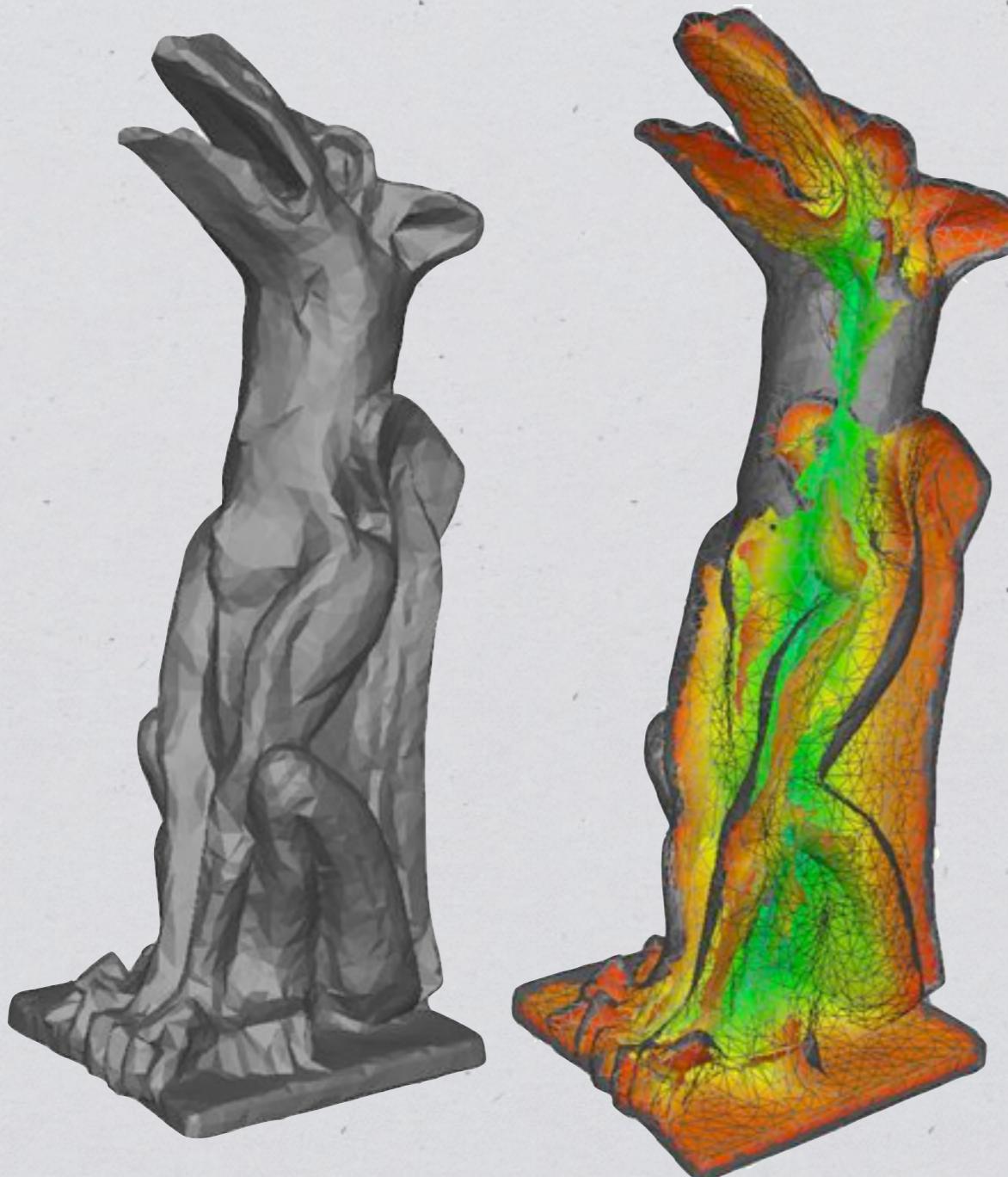
\* Système d'équations à résoudre (en général de manière itérative)

$$\left\{ \begin{array}{l} f(A) \\ g(B) \\ \nabla f(a) = \lambda \times \nabla g(b) \\ \overrightarrow{AB} = \mu \times \nabla g(b) \end{array} \right\}$$

# Autres types de surfaces

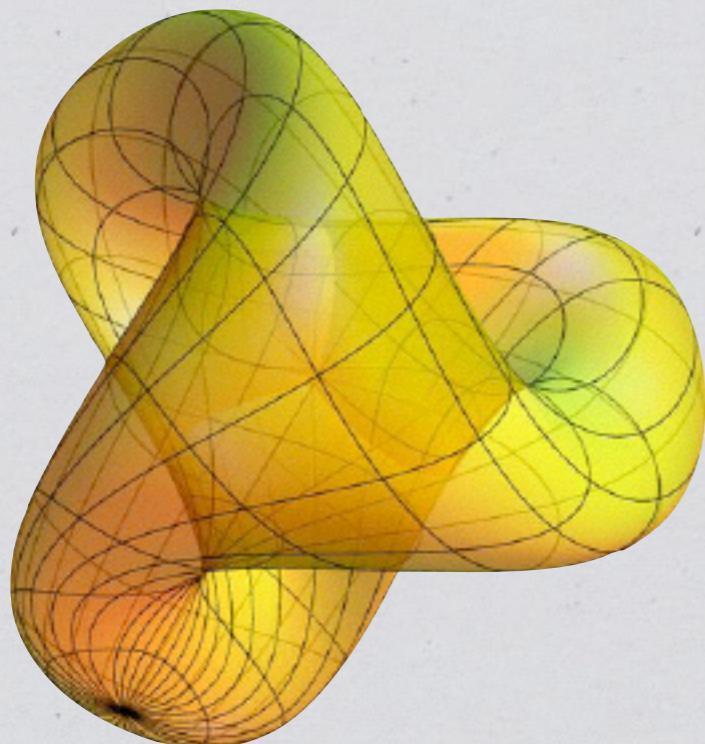


# Cartes de distance



- \* Équivalent à une surface implicite discrète (grille régulière où pour chacun des sommets on calcule la distance à la surface)
- \* Méthode de détection équivalente aux surfaces implicites

# Surfaces Paramétriques



\* Résolution formelle très complexe (proche des surf. implicites):

$$\vec{F}(s_0, t_0) - \vec{G}(u_0, v_0) = \lambda \left( \frac{\partial \vec{G}(u_0, v_0)}{\partial u} \wedge \frac{\partial \vec{G}(u_0, v_0)}{\partial v} \right)$$

$$\frac{\partial \vec{F}(u_0, v_0)}{\partial s} \wedge \frac{\partial \vec{F}(u_0, v_0)}{\partial t} = \mu \left( \frac{\partial \vec{G}(u_0, v_0)}{\partial u} \wedge \frac{\partial \vec{G}(u_0, v_0)}{\partial v} \right)$$

\* En général, approximation locale et polygonale des surfaces

# Bilan

- \* Processus coûteux: chaque test implique des dizaines d'instructions et il y a  $(n.m)$  tests à faire avec  $n$  et  $m$  qui sont ( $> 10'000$ )...
- \* ... et ce pour un couple d'objets. Dans une simulation, il a  $k$ .  
 $(k-1)$  couples d'objets à tester...
- \* ... nécessité d'accélérer le calcul des collisions
- \* La réponse à la collision conditionne le choix de la détection

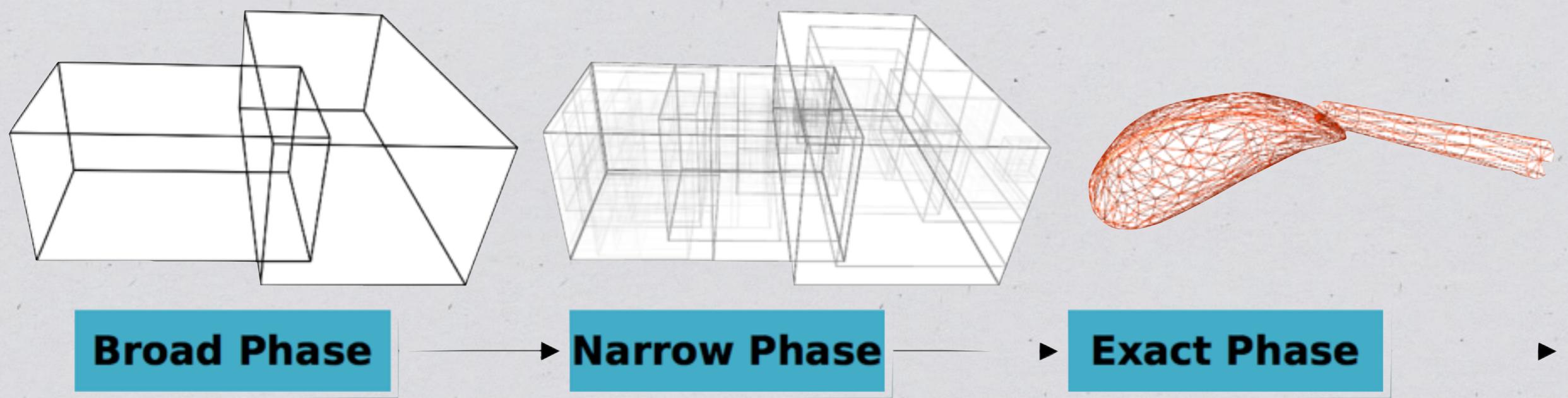
# 5. Accélérations



# Terminologie

- \* Réduction du nombre de couples à inspecter → **Broad Phase**
- \* Pour un couple d'objets, réduction du nombre de “tests” à effectuer → **Narrow Phase**
- \* Assemblage: Broad Phase, Narrow Phase, Détection exacte = **Pipeline de collisions**

# Illustration

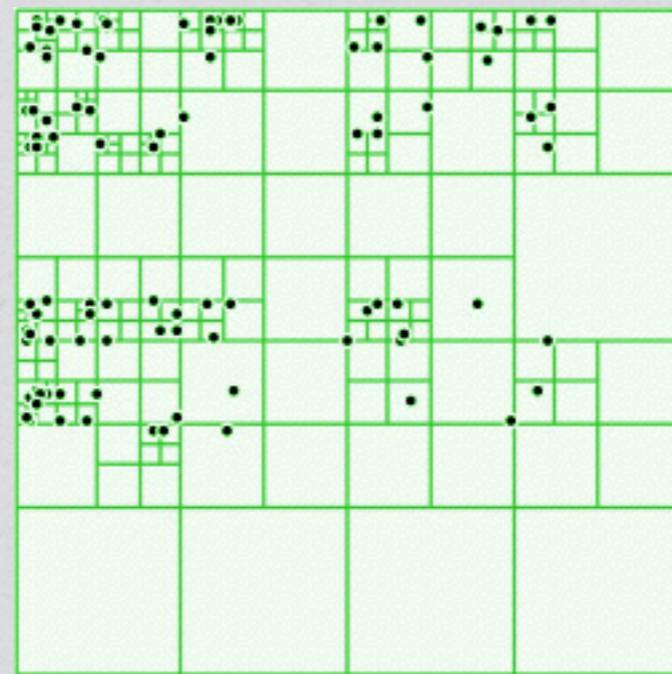


# Broad Phase

- \* 1. Découpage spatiaux absolus
- \* Grille de voxels = grille régulière dans laquelle on plonge les différents objets. Sont testés les couples qui sont présents dans une même cellule de la grille
- \* Choix de la taille d'une cellule ?

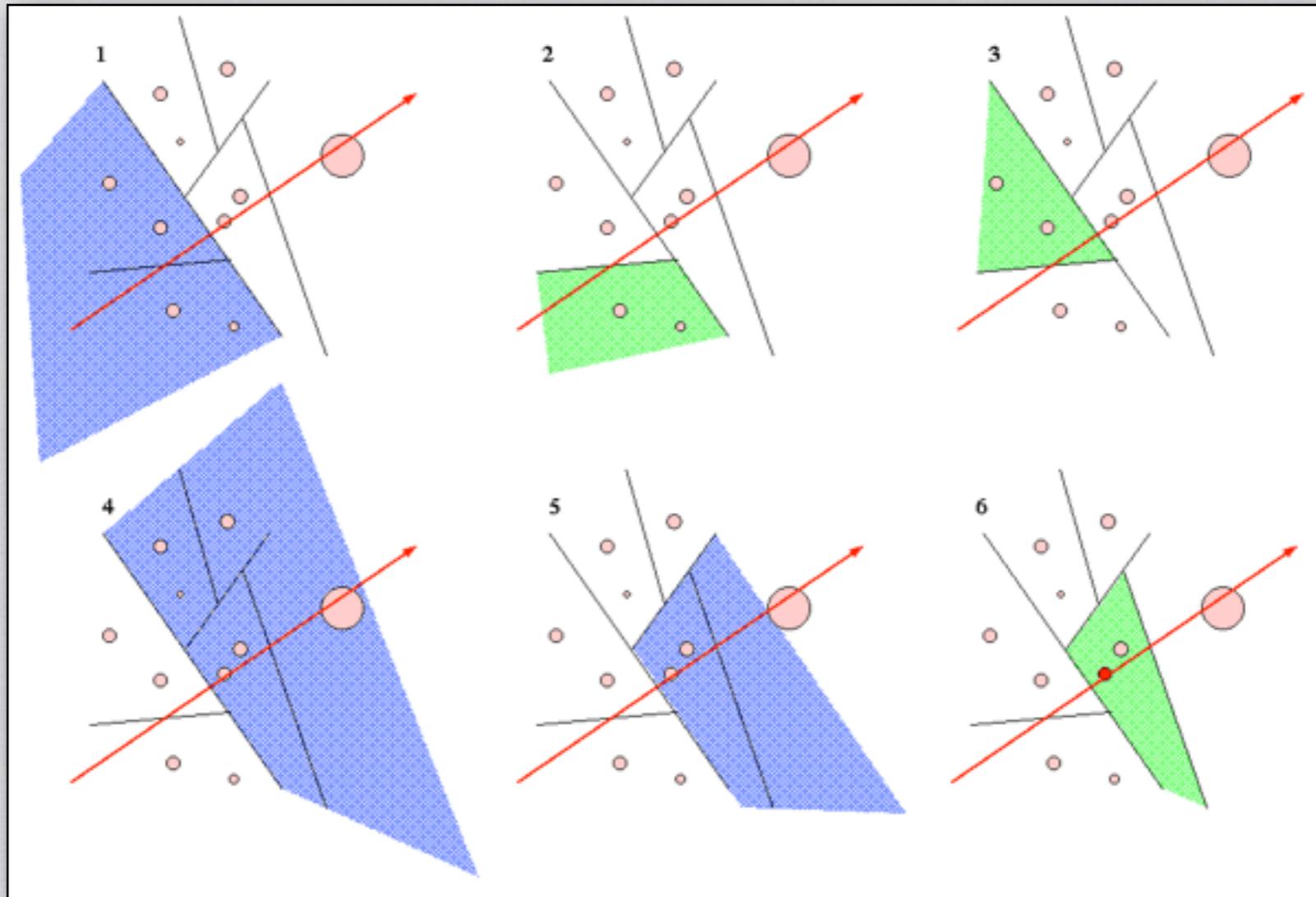
# Broad Phase

- \* 1. Découpage spatiaux absolus
- \* Quad-trees, Octrees



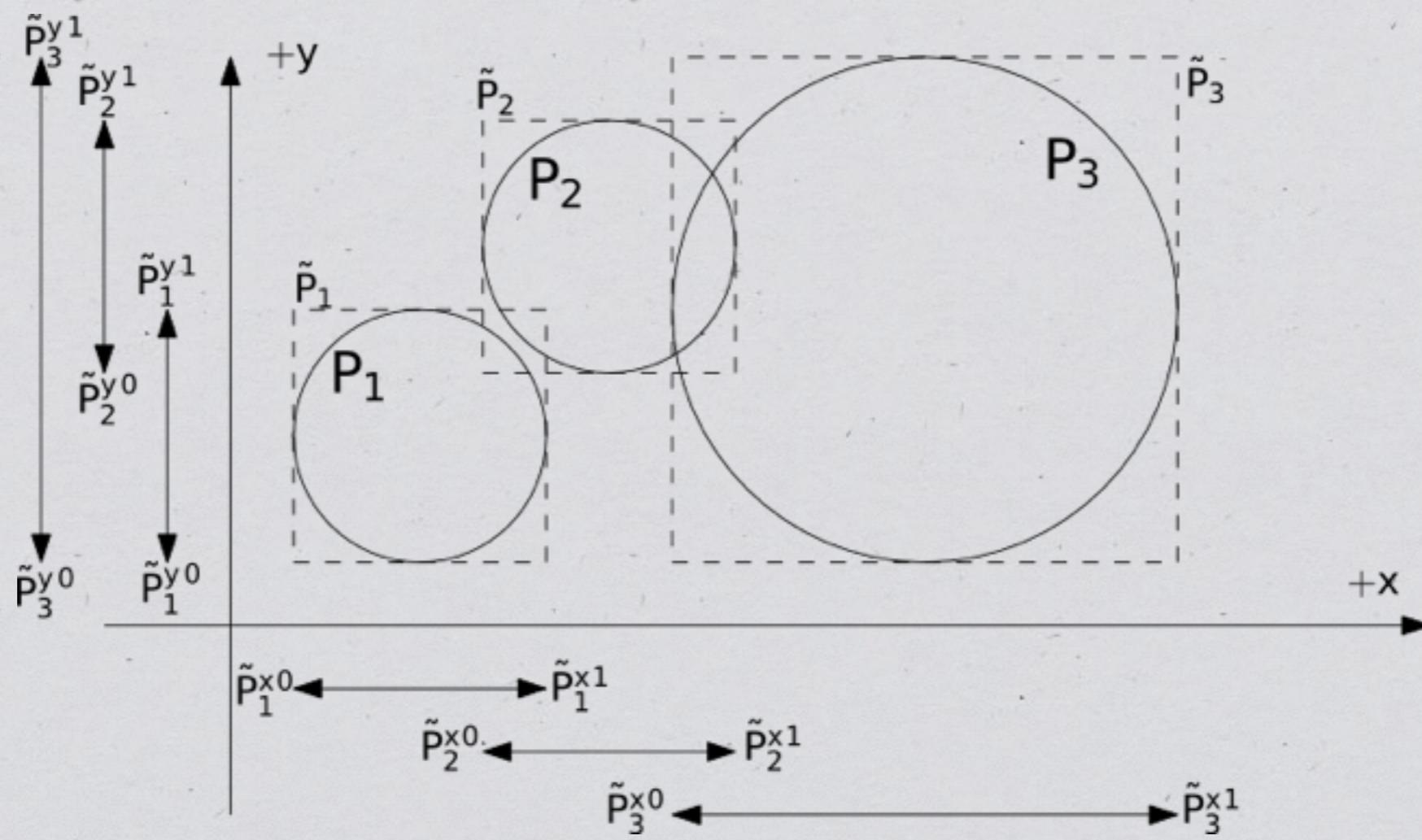
# Broad Phase

- \* 1. Découpage spatiaux relatifs: BSP Tree (Binary Space Partitions)



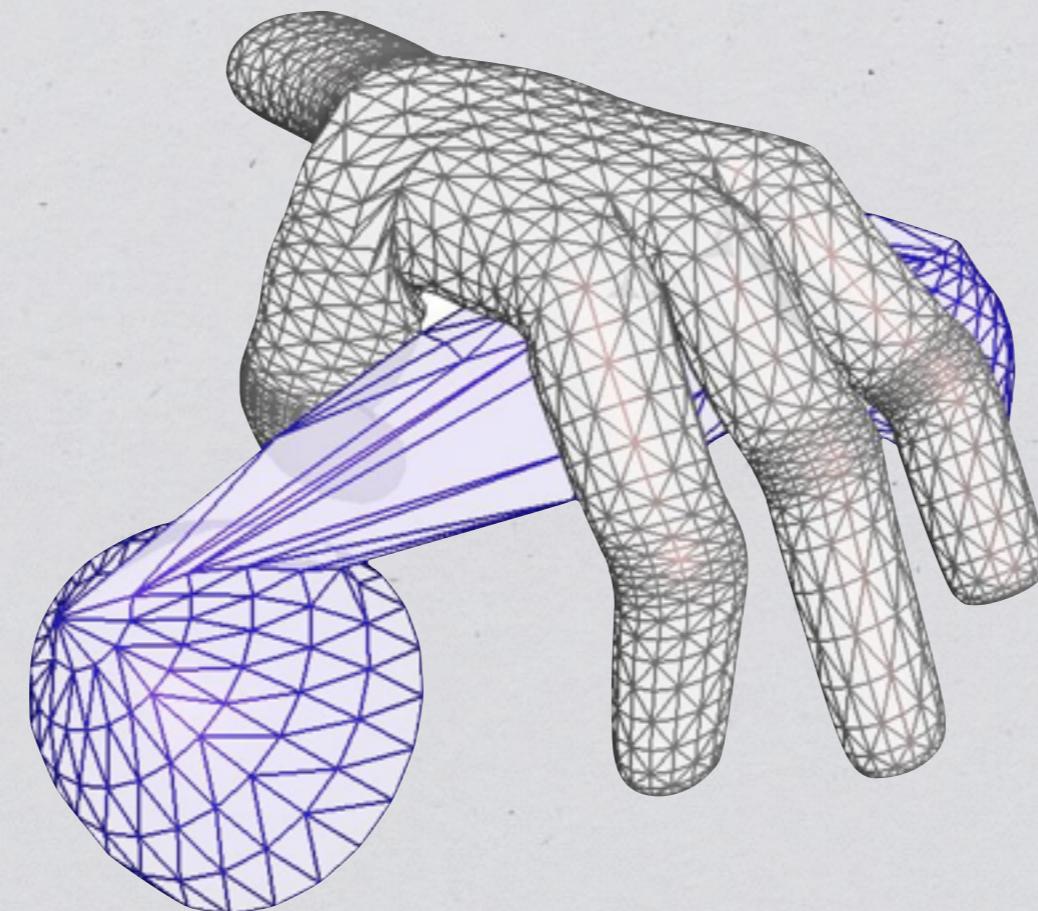
# Broad Phase

\* 2. Approche par classement: (temps constant, tri incrémental)



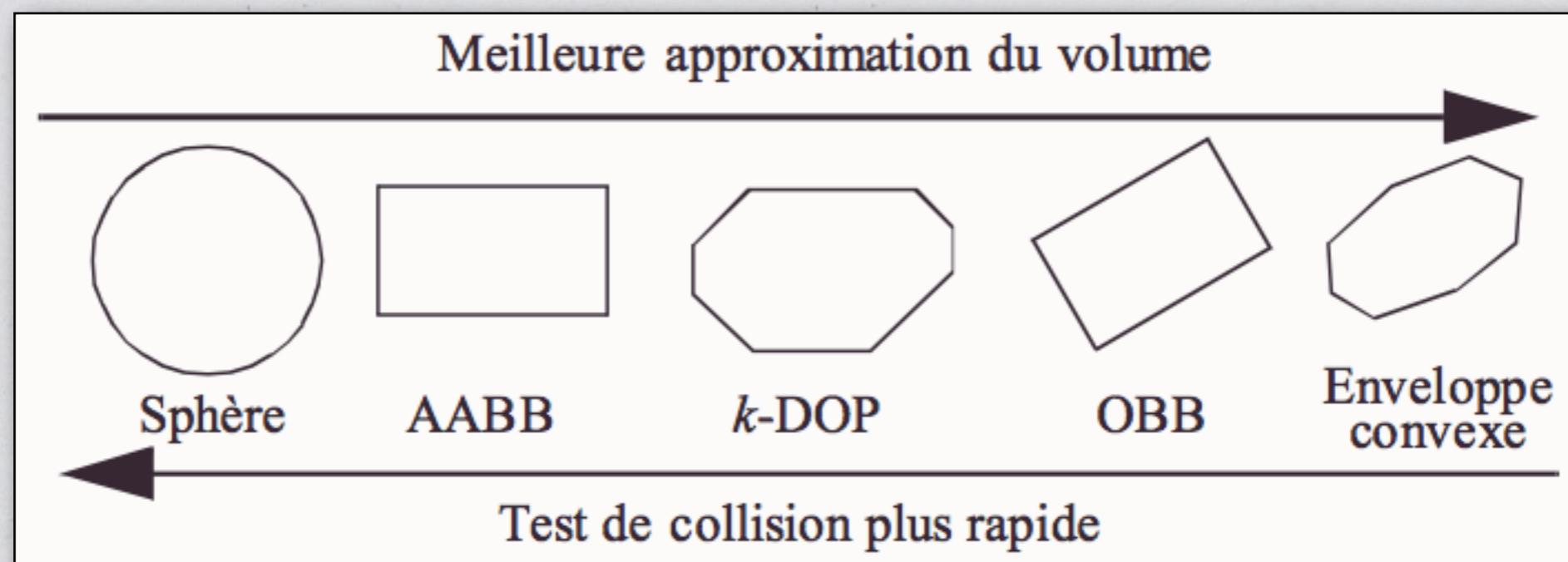
# Broad Phase

- \* 3. Approche cinématique: distance inter-objets + vitesse relative. Si les objets s'éloignent, pas de test de collision.



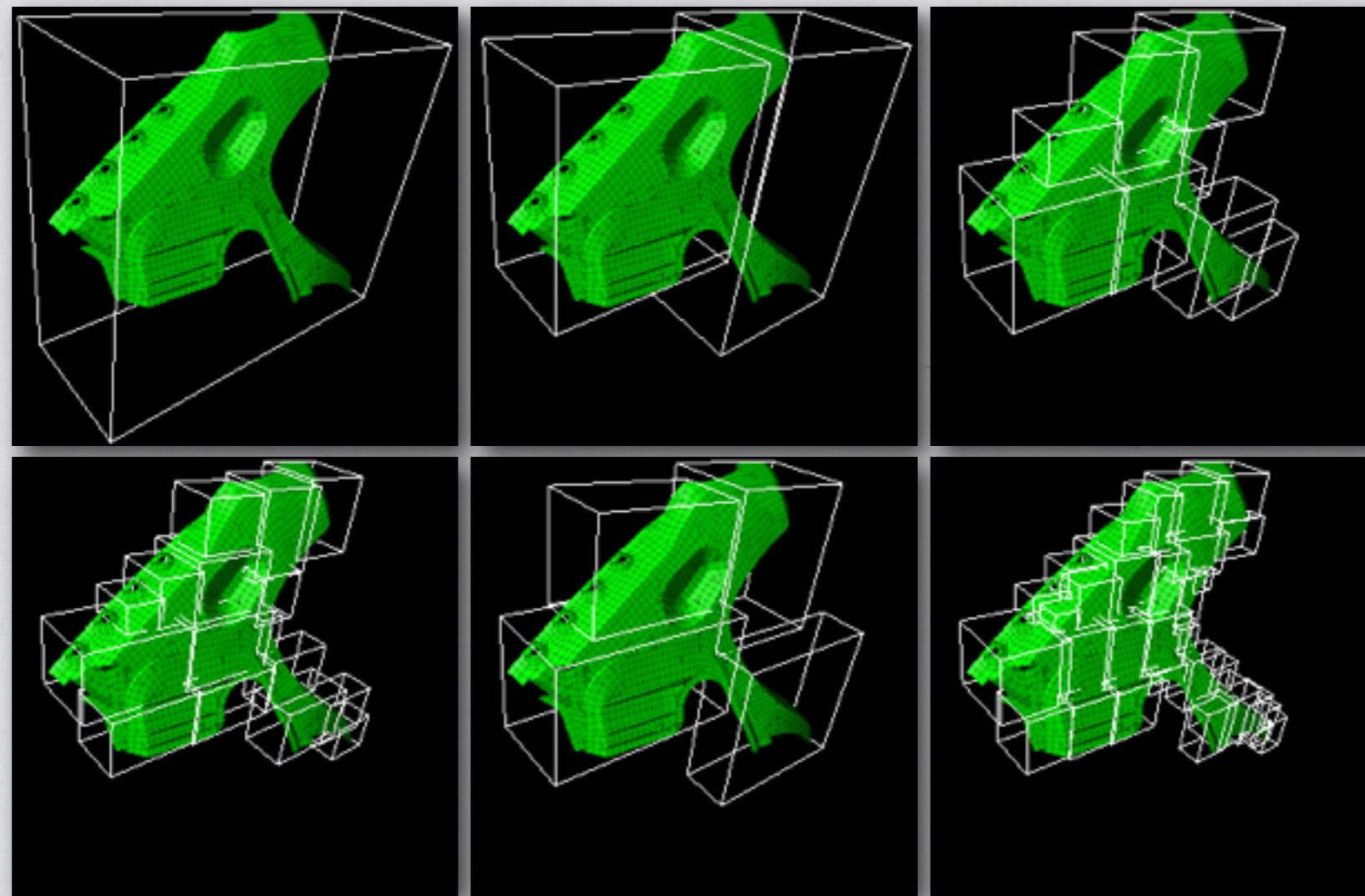
# Broad Phase

\* 4. Volume d'influence (*i.e.* volume englobant). Principe: réaliser sur des volumes simples qui approchent la géométrie des objets



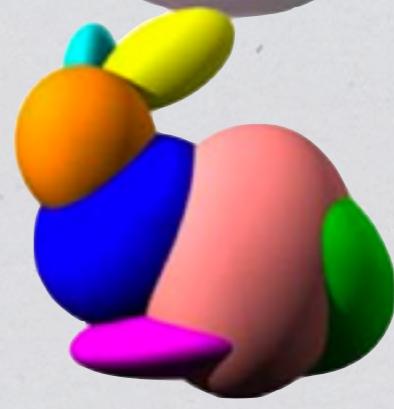
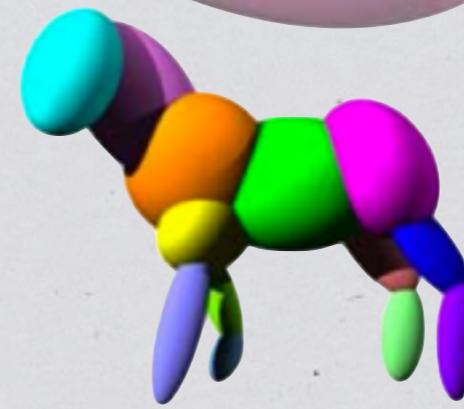
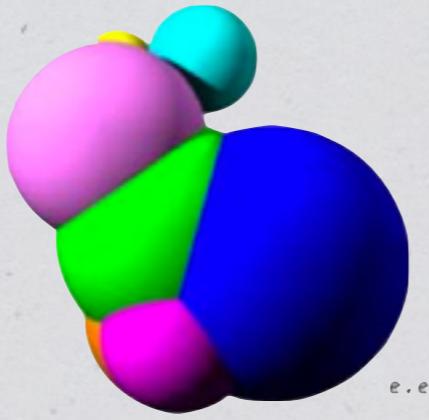
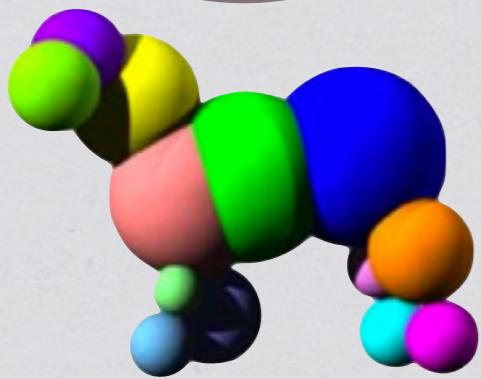
# Narrow Phase

- \* Hiérarchie de Volumes Englobants (Bounding Volume Hierarchy)

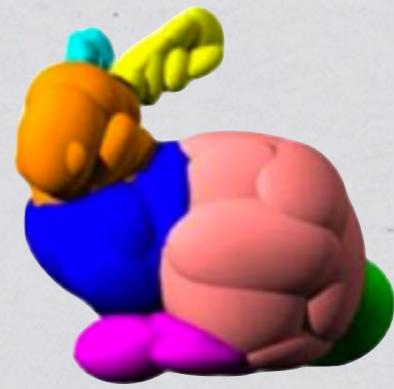
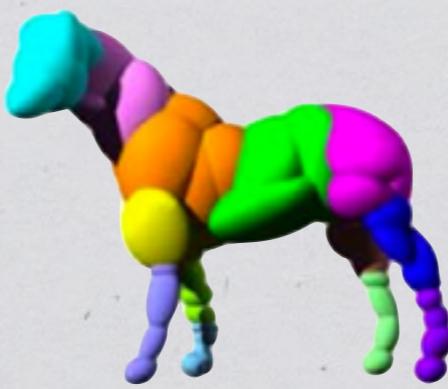
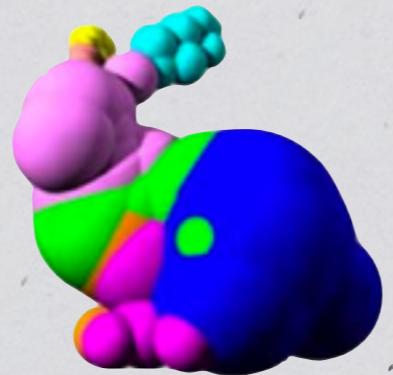
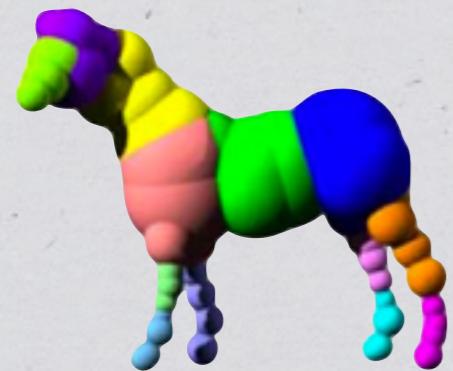




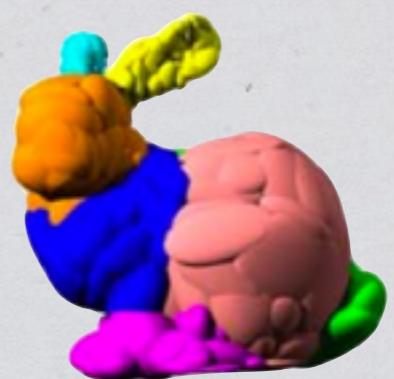
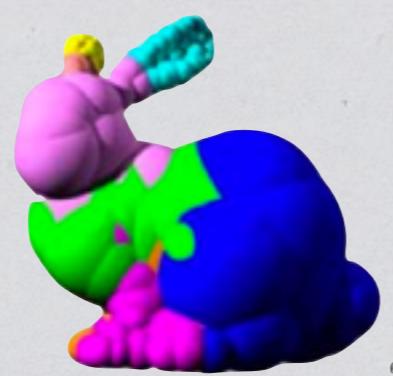
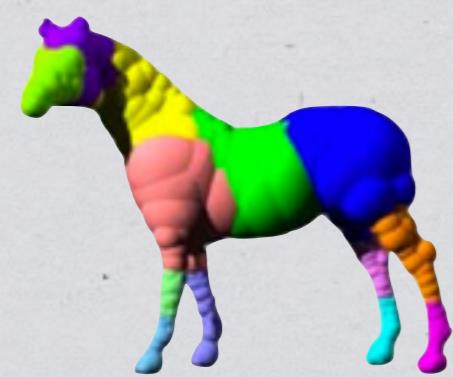
Level 0



Level 1



Level 2



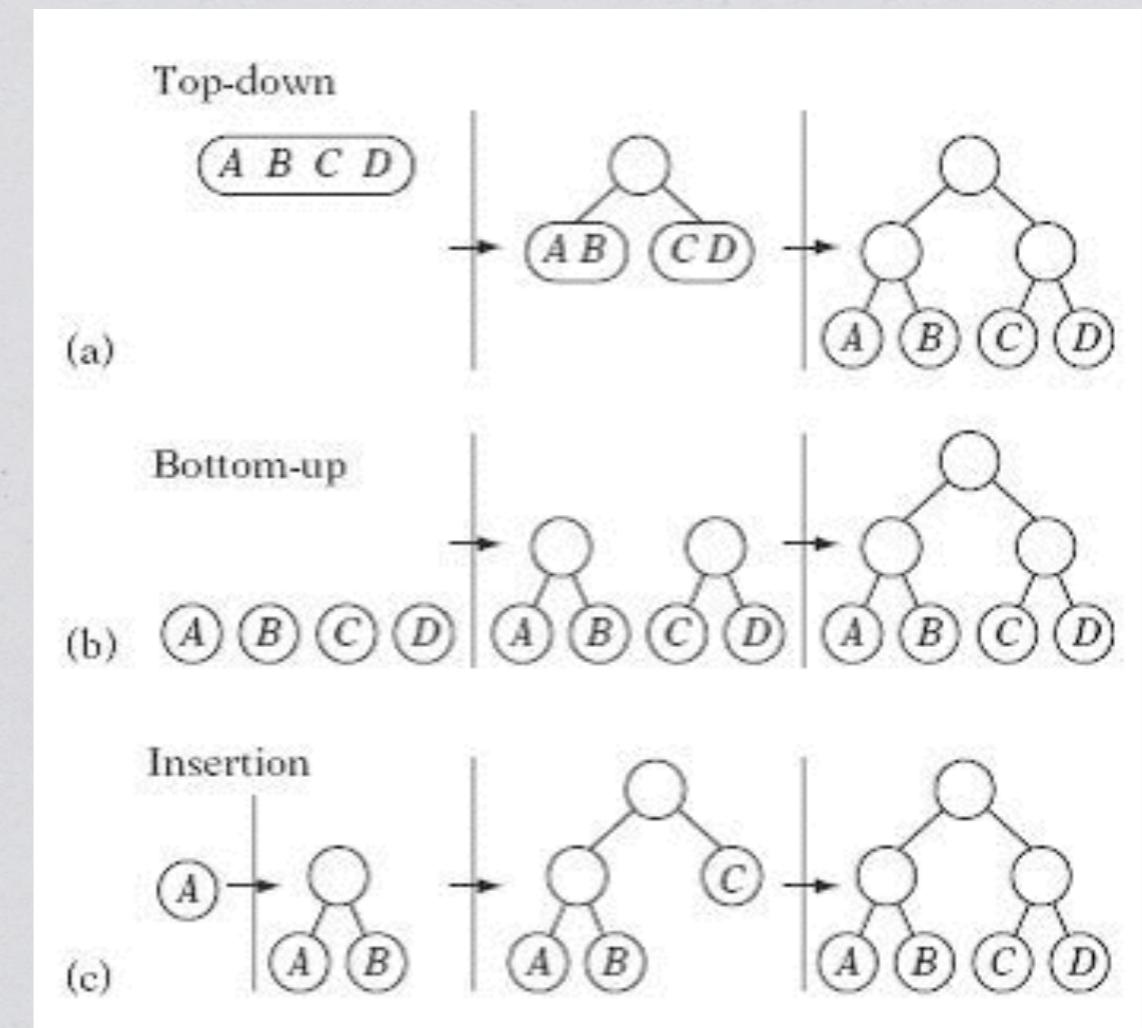
Level 3

# Narrow Phase

- \* BVH:

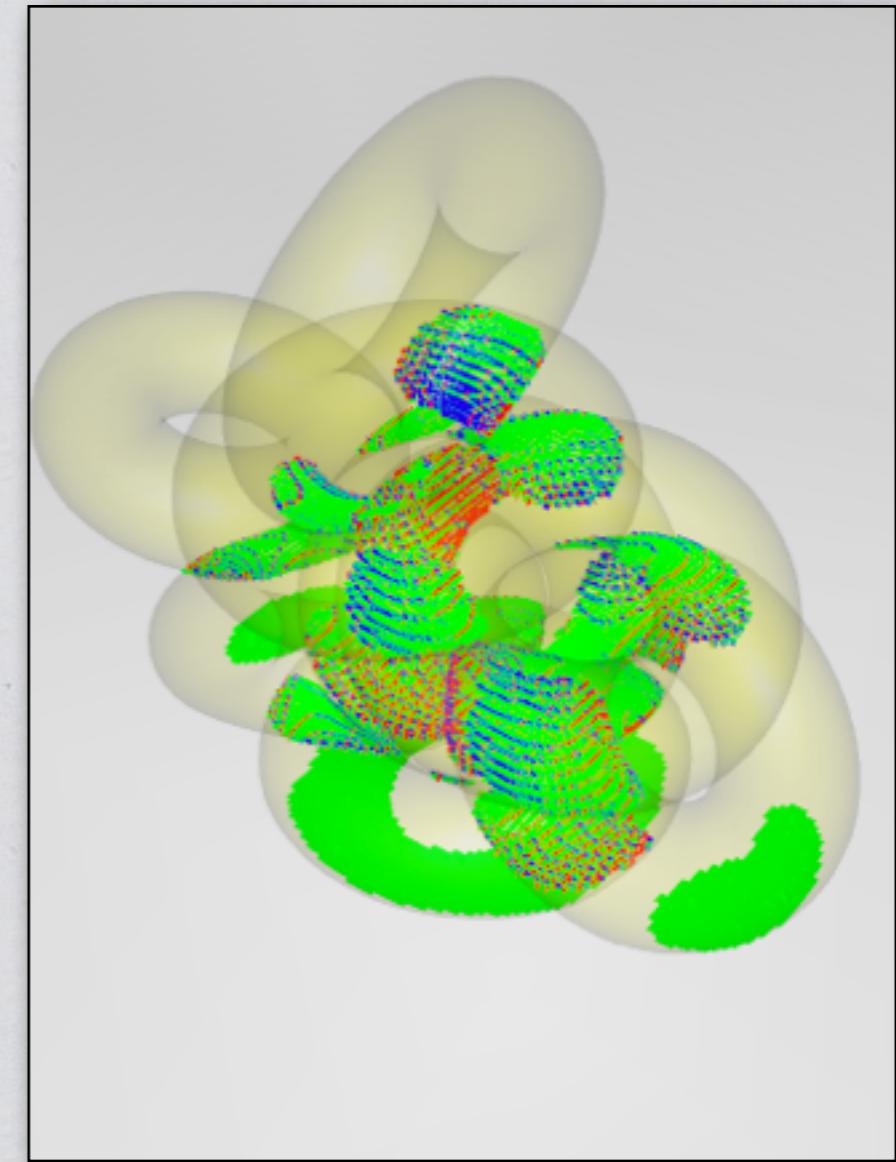
- \* Construction “Bottom-Up” vs “Top-Down”

- \* Coût de la mise à jour de l’arbre (objets déformables)

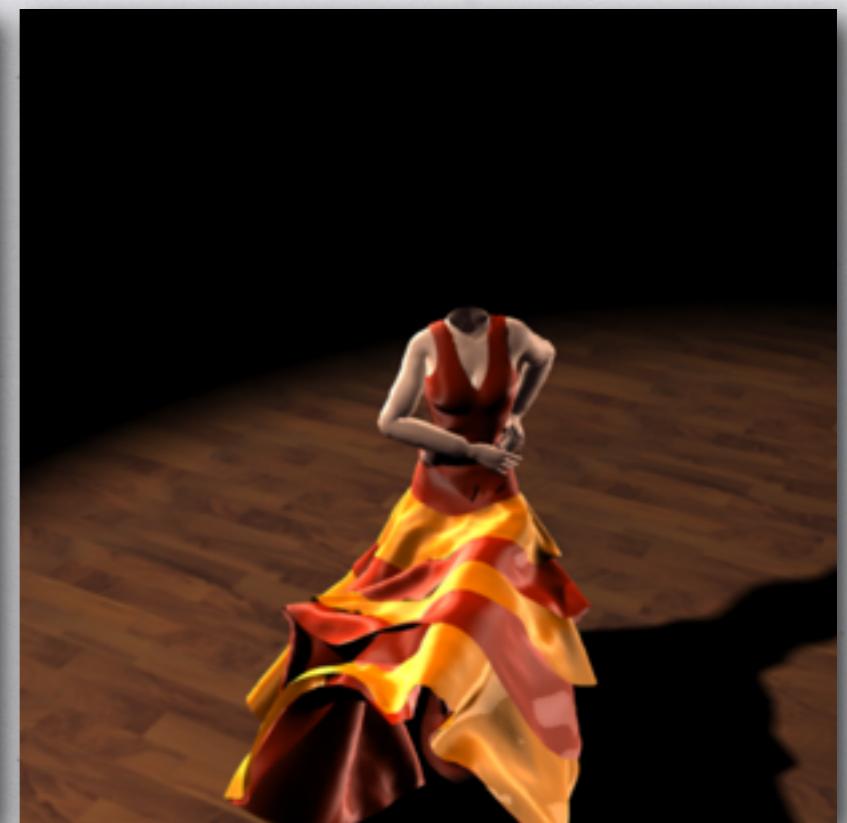
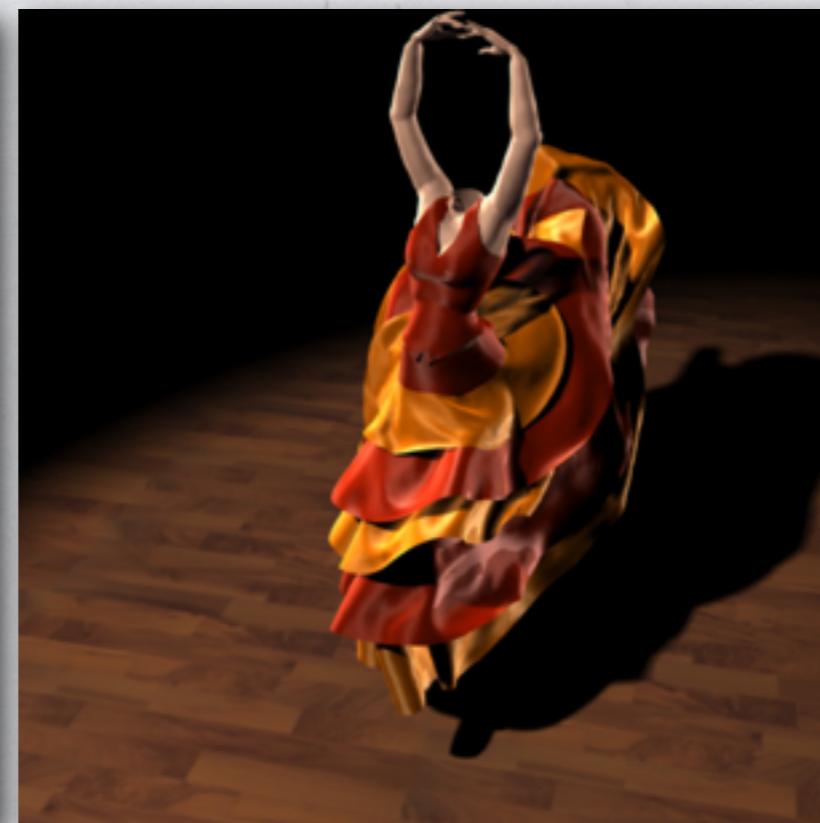


# Narrow Phase

- \* Utilisation du GPU  
(rasterizer).



# Auto-Collisions



# Auto-Collisions: Besoin de méthodes spécifiques

- \* Les accélérations proposées précédemment ne “fonctionnent” pas:
  - \* *broad phase*: inutile cf même objet
  - \* *narrow phase avec BVH*: énormément de collisions potentielles détectées (triangles adjacents)

---

# CONCLUSION

---

# Bilan

- \* Etape la plus coûteuse d'une boucle de simulation, primordial d'optimiser
- \* Pas de panacée, dépend fortement de l'application (modélisation, nature des objets, type de réponse souhaitée) néanmoins présence des deux étapes du pipeline
- \* Prise en compte d'objets déformables empêche / limite certaines optimisations.
- \* Conséquence: état de l'art très vaste, difficile de benchmarker toutes les méthodes (études parfois contradictoires)



**THE END**

SLIDES DISPONIBLES EN PDF:  
[HTTP://DEQUIDT.PLIL.NET](http://dequidt.plil.net)