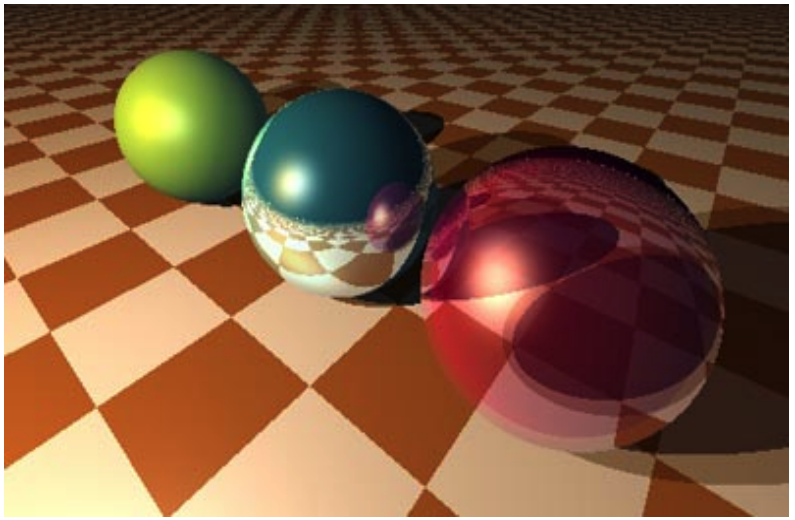
Lancer de Rayon

Objectif



Principaux intérêts

Haute qualité par défaut

- éclairage par pixel
- ombres portées
- réflexions
- réfractions

Principe

Génération des rayons

rayons primaires : rayons émis à partir d'un point de vue, au travers d'une image, vers la scène

Tracé de rayon

trouver la plus proche intersection avec la scène 3D

Calcul de l'apparence

éclairement direct à partir des sources de lumière (rayon d'ombre).

Difficultés

Génération des rayons

Tracé de rayon

- calculer l'intersection rayon / objet
- trouver l'intersection la plus proche **le plus rapidement possible** !

Calcul de l'apparence

- ombres, réfractions, etc.

Génération des rayons primaires

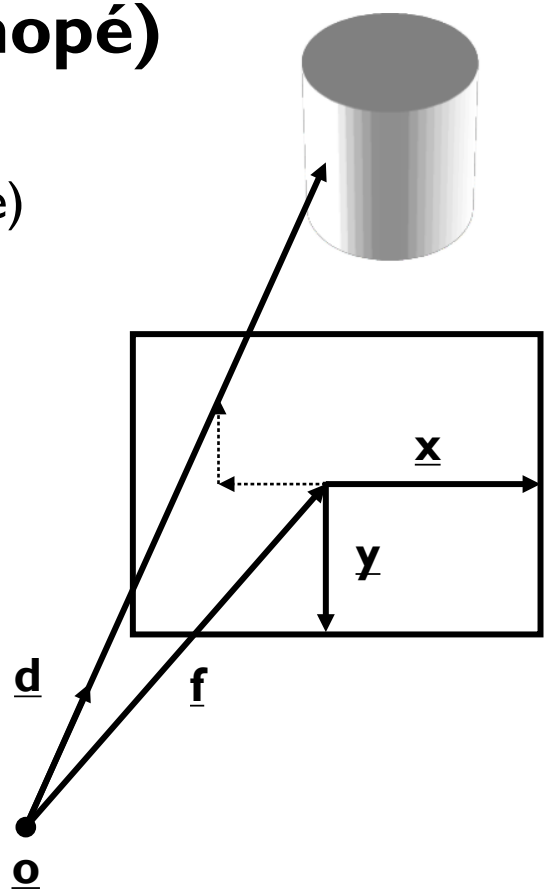
Un rayon : $\underline{r}(t) = \underline{o} + t\,\underline{d}$

- origine $\underline{o}=(o\underline{x}, o\underline{y}, o\underline{z})$, direction $\underline{d}=(d\underline{x}, d\underline{y}, d\underline{z})$ avec $||\underline{d}||=1$

Hypothèse : "Pinhole camera" (sténopé)

- $\underline{o}$: origine (point de vue)
- $\underline{f}$: axe optique (vecteur vers le centre de l'image)
- $\underline{x}$, $\underline{y}$: axes sur l'image
- width, height : résolution de l'image

```
for(i=0; i<width; i++)
  for(j=0; j<height; j++)
  {
    d = f + 2(i/width  - 0.5)x
          + 2(j/height - 0.5)y;
    d = d/|d|; // normalisation
    color = ray_tracing(o, d);
    write_pixel(i,j,color);
  }
```

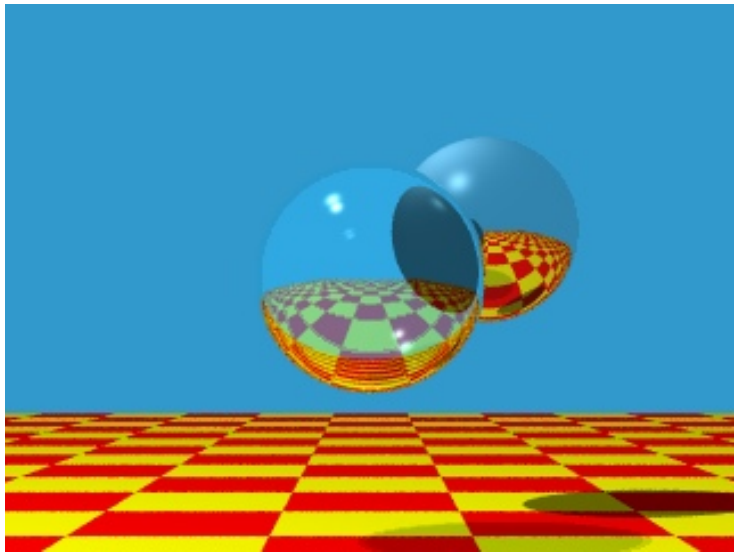


Extension du modèle [Turner Whitted 1980]

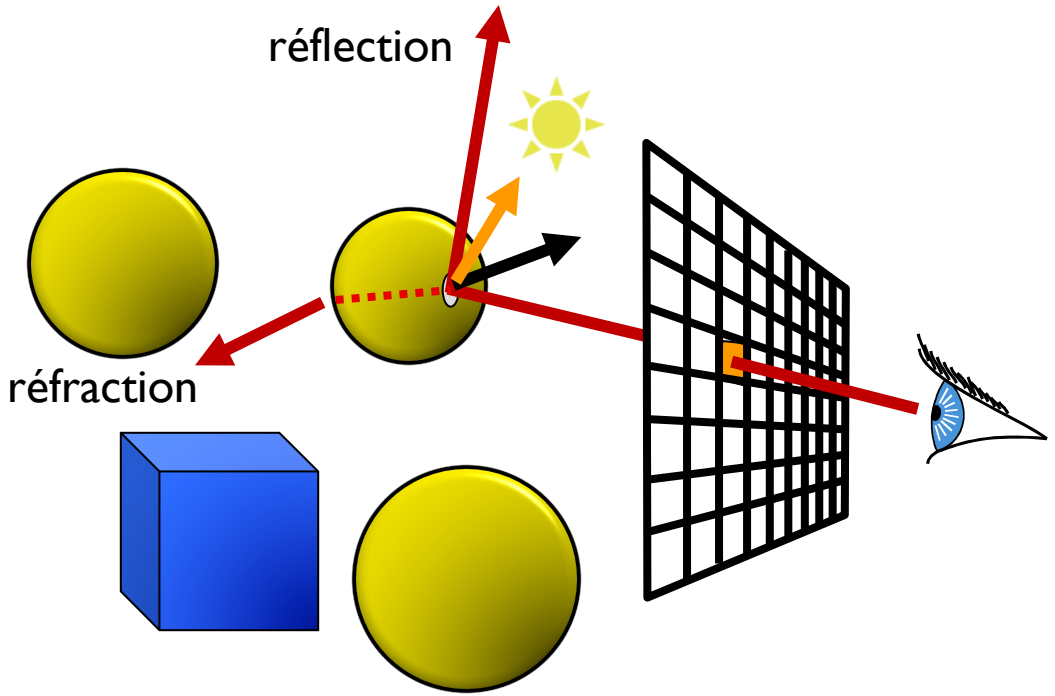
Trois nouveaux rayons sont générés :

- un rayon **réfracté***,
- un rayon **réfléchi***,
- un rayon **d'ombre**

$\Longrightarrow$ **lancer de rayon récursif**

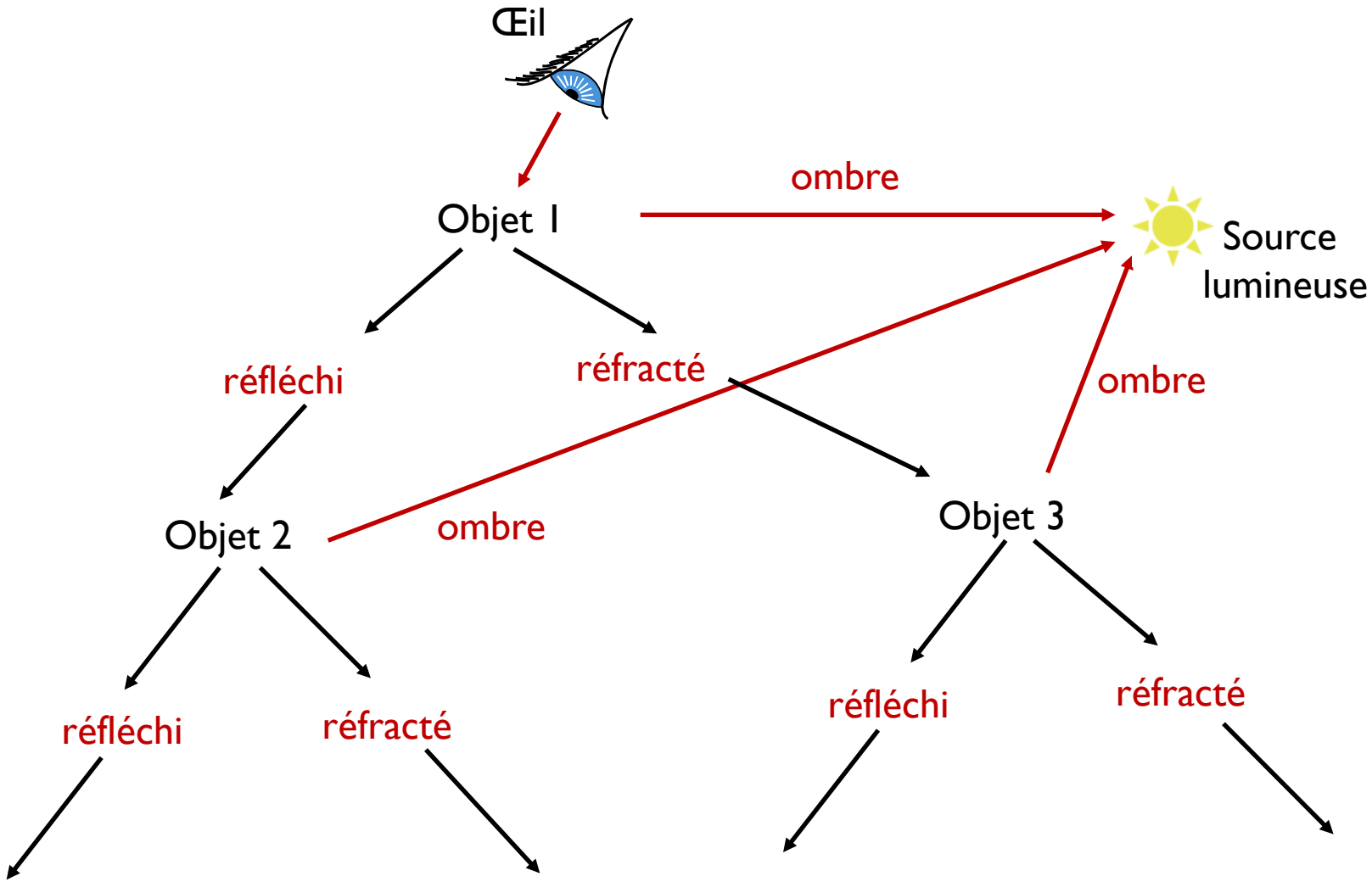


En 1980, 74 min de calcul.



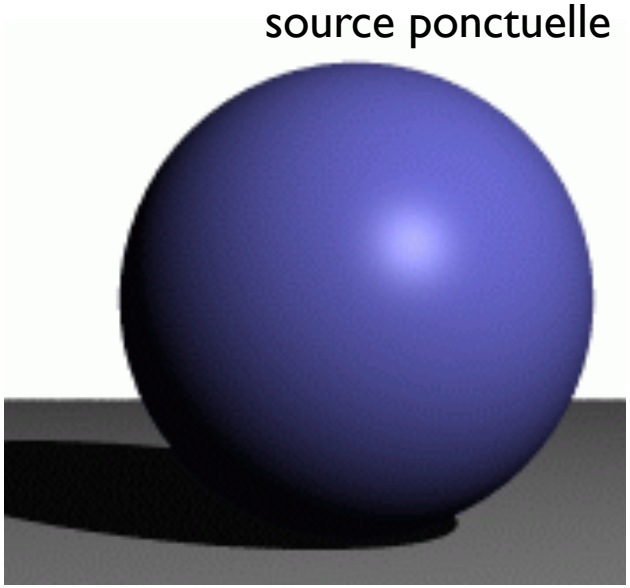
(*) formules réflexion et réfraction

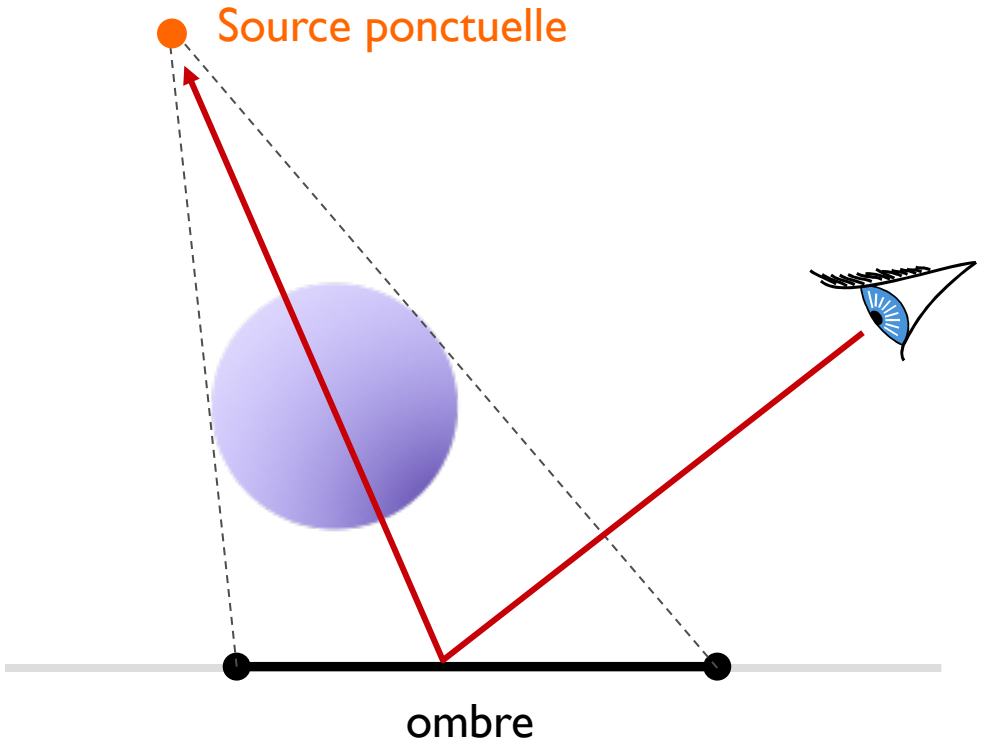
L’arbre des rayons



Ombres dures

Un rayon d'ombre par source de lumière **ponctuelle**





Intersection rayon-scène

Rayon : $\underline{r}\,(t) = \underline{o} + t\,\underline{d}$

$\underline{o} = (o_x, o_y, o_z)$, $\underline{d} = (d_x, d_y, d_z)$

$\Longrightarrow$ équation explicite

Sphère : $\|\underline{p} - \underline{c}\|^2 - r^2 = 0$

$\underline{c}$: centre de la sphère, r : rayon de la sphère

$\Longrightarrow$ équation implicite

Plan : $(\underline{p} - \underline{a}) \cdot \underline{n} = 0$

$\underline{n}$: normale à la surface, $\underline{a}$: un point sur le plan

$\Longrightarrow$ équation implicite

Triangle : partie d'un plan

Intersection rayon-sphère

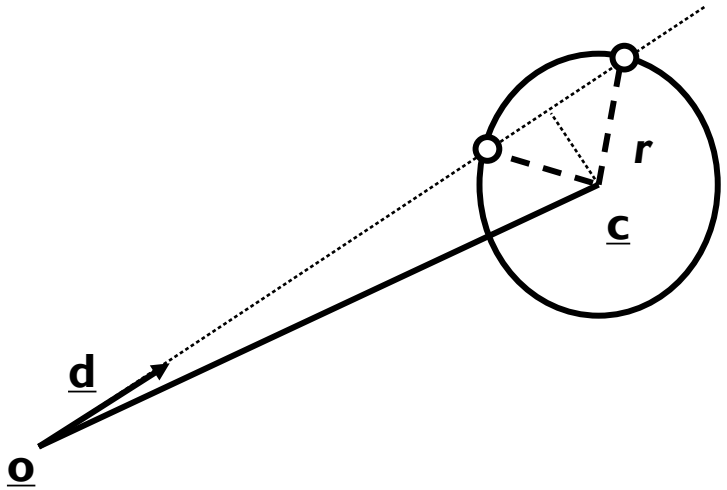
Rayon : $\underline{r}\ (t) = \underline{o} + t\ \underline{d}$

$\underline{o} = (o_x , o_y , o_z)$, $\underline{d} = (d_x , d_y , d_z)$

Sphère : $||\underline{p} - \underline{c}||^2 - r^2 = 0$

$\underline{c}$: centre de la sphère, r : rayon de la sphère

Point d'intersection ?



Intersection rayon-sphère

Étant donnée l'équation de la sphère : $\|\underline{x}-\underline{c}\|^2 - r^2 = 0$

$\underline{c}$: centre de la sphère, r : rayon de la sphère

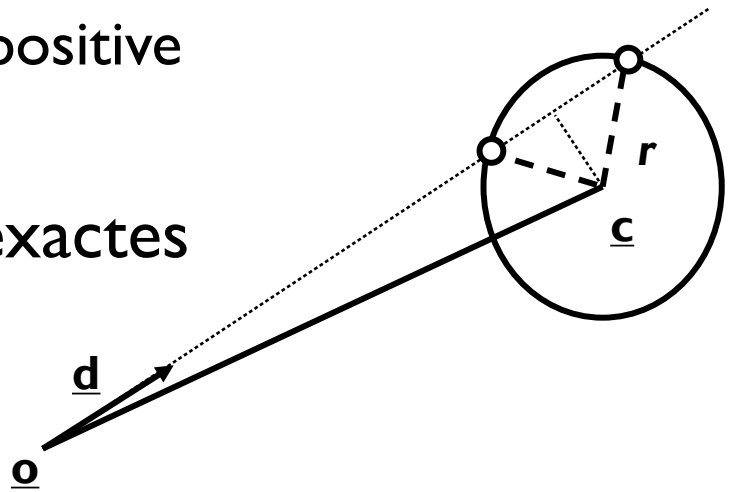
Remplacer $\underline{x}$ **par l'équation du rayon :**

$$t^2\, \underline{d}\cdot\underline{d} + 2t\, \underline{d}\cdot(\underline{o}-\underline{c}) + \|\underline{o}-\underline{c}\|^2 - r^2 = 0$$

équation du second degré en t

- discriminant négatif = pas d'intersection
- 2 racines : garder la plus proche positive (généralement t_-)

$\Longrightarrow$ Permet d'obtenir des sphères exactes



Intersection rayon-plan

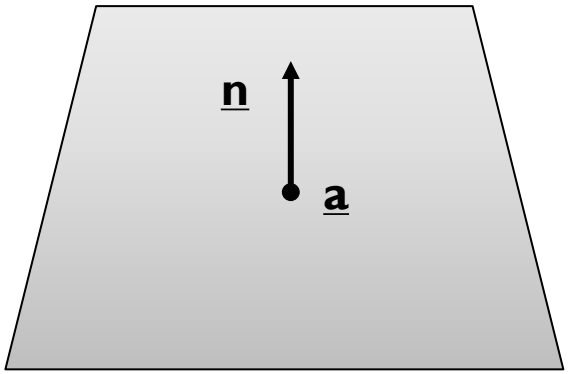
Rayon : $\underline{r}\,(t) = \underline{o} + t\,\underline{d}$

$\underline{o} = (o_x, o_y, o_z)$, $\underline{d} = (d_x, d_y, d_z)$

Plan : $(\underline{p} - \underline{a}) \cdot \underline{n} = 0$, $||\underline{n}|| = 1$

$\underline{\mathbf{n}}$: normale à la surface, $\underline{\mathbf{a}}$: un point sur le plan

Point d'intersection ?



Intersection rayon-plan

Équation du plan : $\underline{\mathbf{x}} \cdot \underline{\mathbf{n}} - D = 0$, $|\underline{\mathbf{n}}| = 1$

- Normale : $\underline{\mathbf{n}}$
- Distance du plan au centre (0, 0, 0) : D = $\underline{\mathbf{a}} \cdot \underline{\mathbf{n}}$

Remplacer $\underline{\mathbf{x}}$ **par l'équation du rayon :**

$$(\underline{o} + t\underline{d}) \cdot \underline{n} - D = 0$$

La solution devient :

$$t = \frac{D - \underline{o} \cdot \underline{n}}{\underline{d} \cdot \underline{n}}$$

4 cas:

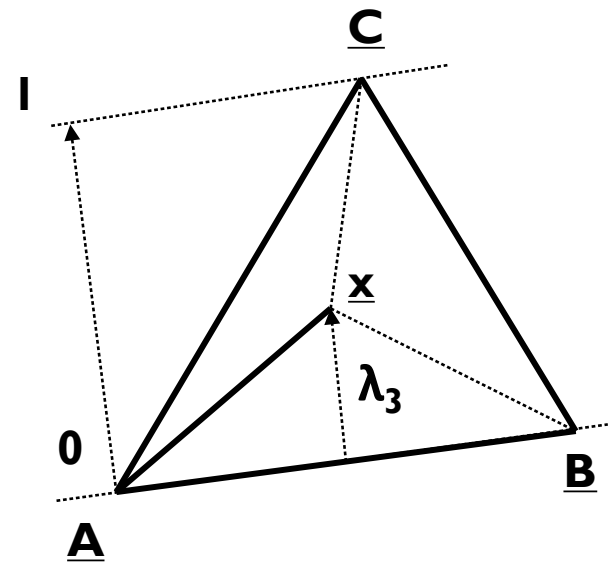
- t infini $\Longrightarrow$ rayon parallèle et distinct du plan
- t non-défini $\Longrightarrow$ rayon confondu avec le plan
- $t < 0 \Longrightarrow$ intersection derrière la caméra
- $t > 0 \Longrightarrow$ intersection devant la caméra

Intersection rayon-triangle (I)

Coordonnées barycentriques

- Pour un triangle non-dégénéré ABC
- $\underline{x} = \lambda_1 \underline{A} + \lambda_2 \underline{B} + \lambda_3 \underline{C}$
- $\lambda_1 + \lambda_2 + \lambda_3 = 1$
- λ_3 = Aire(ΔAxB) / Aire(ΔACB)

  $\Longrightarrow$ Aire relative signée

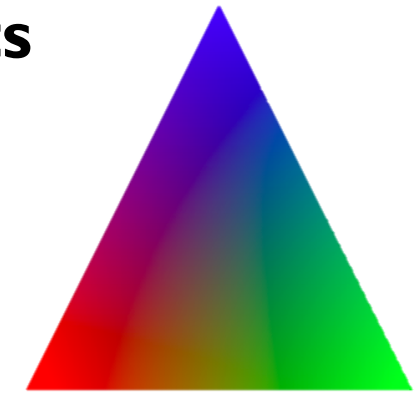


Test d'appartenance au polygone

- à l'intérieur si tous les λ_i sont plus grand ou égal à zéro

Interpolation des attributs aux sommets

- normales, couleurs, coordonnées de texture, etc.



Intersection rayon-triangle (1)

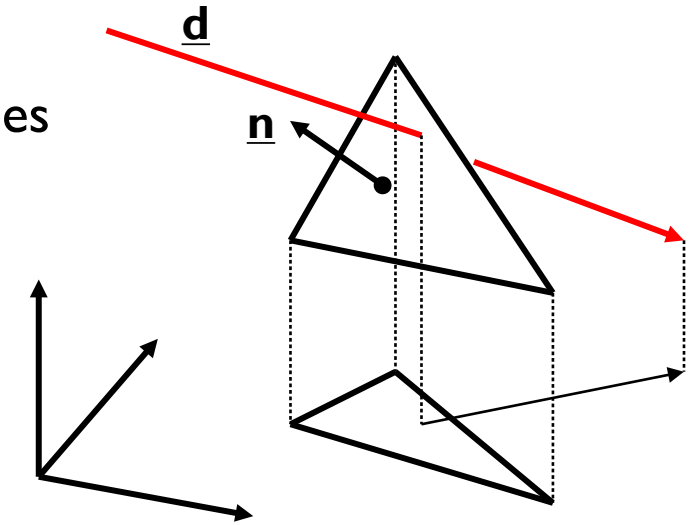
Calculer l'intersection avec le plan (infini)

- Attention aux cas tangents !

Test d'appartenance au polygone

Possibilité de le faire en 2D

- projection sur un des plans du **repère global** (choix de l'axe en fonction de la normale) = ignorer une coordonnée
- test avec les coordonnées barycentriques



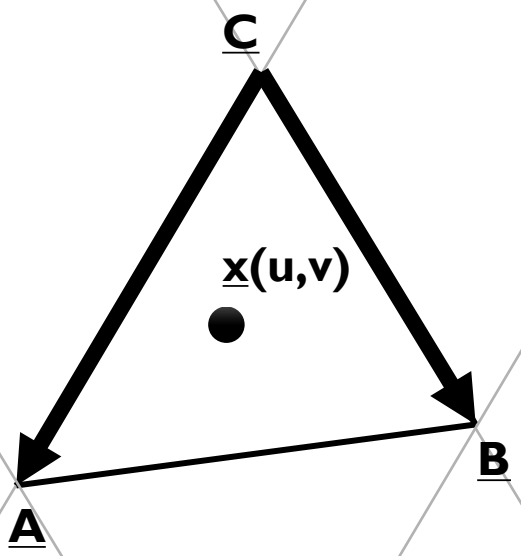
Intersection rayon-triangle (2)

$\lambda_1 + \lambda_2 + \lambda_3 = 1 \Leftrightarrow \lambda_3 = 1 - \lambda_1 - \lambda_2$

Réécriture avec $u = \lambda_1$ et $v = \lambda_2$:

$\underline{x} = u\,\underline{A} + v\,\underline{B} + (1 - u - v)\,\underline{C}$

$= u\,(\underline{A} - \underline{C}) + v\,(\underline{B} - \underline{C}) + \underline{C}$



Système de coordonnées du plan non-orthogonal

Intersection rayon-triangle (2)

Test d'intersection rapide

- ***t*** : distance à l'origine
- ***u***,***v*** : coordonnées dans le triangle (***u***,***v*** dans [0,1])

$$\underline{o} + t\underline{d} = u(\underline{A} - \underline{C}) + v(\underline{B} - \underline{C}) + \underline{C}$$

- Système de 3 équations à 3 inconnues, forme matricielle :

$$\underbrace{[-\underline{d},\ \underline{A} - \underline{C},\ \underline{B} - \underline{C}\ ]}_{M : \text{matrice } 3\text{x}3}[t, u, v]^T = \underline{o} - \underline{C}$$

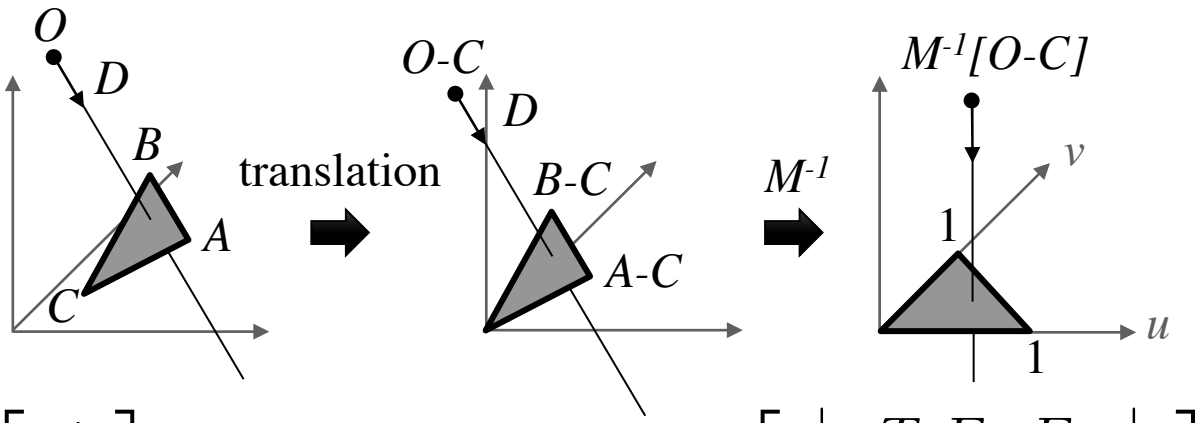
- Optimisation : stopper dès que ***u*** ou ***v*** est négatif

Méthode de référence !

« Fast, minimum storage ray-triangle intersection »
Tomas Möller and Ben Trumbore, Journal of Graphics Tools, 1997

Intersection rayon-triangle (2)

Interprétation géométrique :



Cramer's rule :

$$E_1 = A - C$$
$$\text{avec } E_2 = B - C$$
$$T = O - C$$

$$\begin{bmatrix} t \\ u \\ v \end{bmatrix} = \frac{1}{\begin{bmatrix} | & -D & E_1 & E_2 & | \end{bmatrix}} \begin{bmatrix} | & T & E_1 & E_2 & | \\ | & -D & T & E_2 & | \\ | & -D & E_1 & T & | \end{bmatrix}$$

où déterminant : $|A\ B\ C| = -(A \times C) \cdot B = -(C \times B) \cdot A$

D'où :

$$\begin{bmatrix} t \\ u \\ v \end{bmatrix} = \frac{1}{(D \times E_2) \cdot E_1} \begin{bmatrix} (T \times E_1) \cdot E_2 \\ (D \times E_2) \cdot T \\ (T \times E_1) \cdot D \end{bmatrix} = \frac{1}{P \cdot E_1} \begin{bmatrix} Q \cdot E_2 \\ P \cdot T \\ Q \cdot D \end{bmatrix}$$

avec $P = (D \times E_2)$ et $Q = (T \times E_1)$

Intersection rayon-triangle (2)

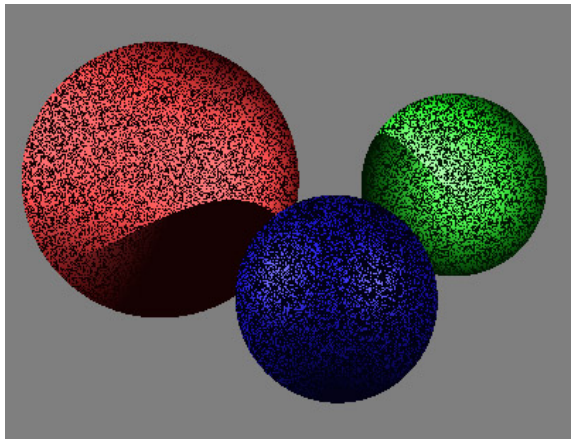
Avantages Möller-Trumbore :

- Rapide
  - produits scalaires et vectoriels (1 ÷, 27 ×, 17 ±)
- Faible coût mémoire
  - équation du plan pas stockée
  - normale pas stockée
- Coordonnées barycentriques en bonus !

Intersection rayon-scène

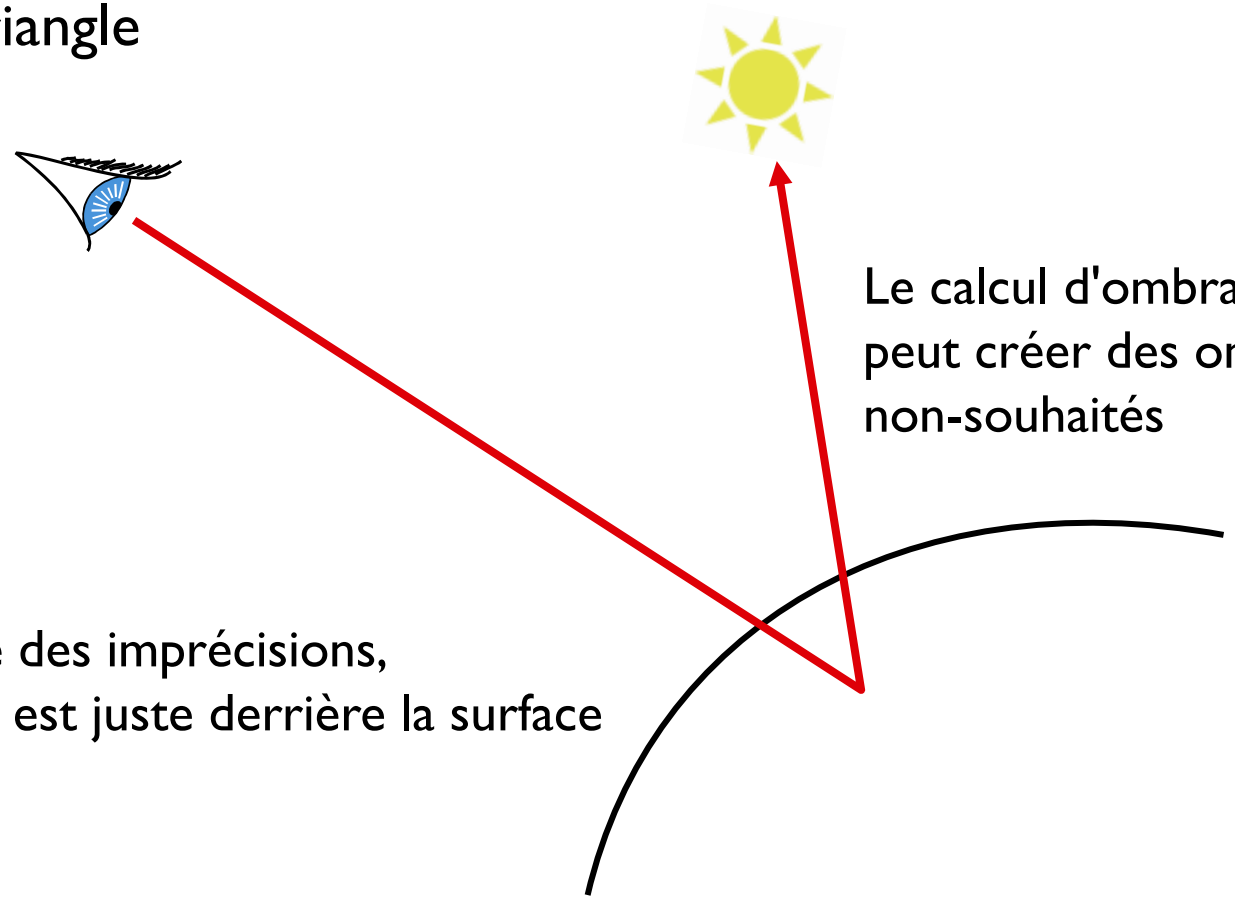
- Polygones : [Appel '68]
- Quadriques, CSG : [Goldstein & Nagel '71]
- Torres : [Roth '82]
- Patches bi-cubiques : [Whitted '80, Kajiya '82, Benthin '04]
- Surfaces algébriques : [Hanrahan '82]
- Swept surfaces : [Kajiya '83, van Wijk '84]
- Fractales : [Kajiya '83]
- NURBS : [Stürzlinger '98]
- Surfaces de subdivision : [Kobbelt et al. '98, Benthin '04]
- Points : [Schaufler et al. '00, Wald '05]

Problème de précision



Un point n'est jamais **exactement**

- sur le plan ou la sphère
- dans le triangle



Le calcul d'ombrage peut créer des ombres non-souhaités

À cause des imprécisions,
le point est juste derrière la surface

Lancer de rayon : avantages

Pas de calculs supplémentaires pour

- l'élimination des parties cachées
- les ombres
- la transparence
- le plaquage de textures (y compris procédurales)

Inter-réflexions spéculaires entre objets

Primitives graphiques quelconques

- pas seulement pour les polygones !

Lancer de rayon : limitations

Arbre limité à une certaine profondeur

- Les objets complexes peuvent avoir un problème (diamant, cristal…)

Limité à Snell-Descartes

- Tous les objets réfléchissant sont métalliques
- Pas d'inter-réflexion entre objets diffus

Lent

- 95 % du temps est utilisé pour les intersections

Structures d'accélération

Trouver l'intersection la plus proche

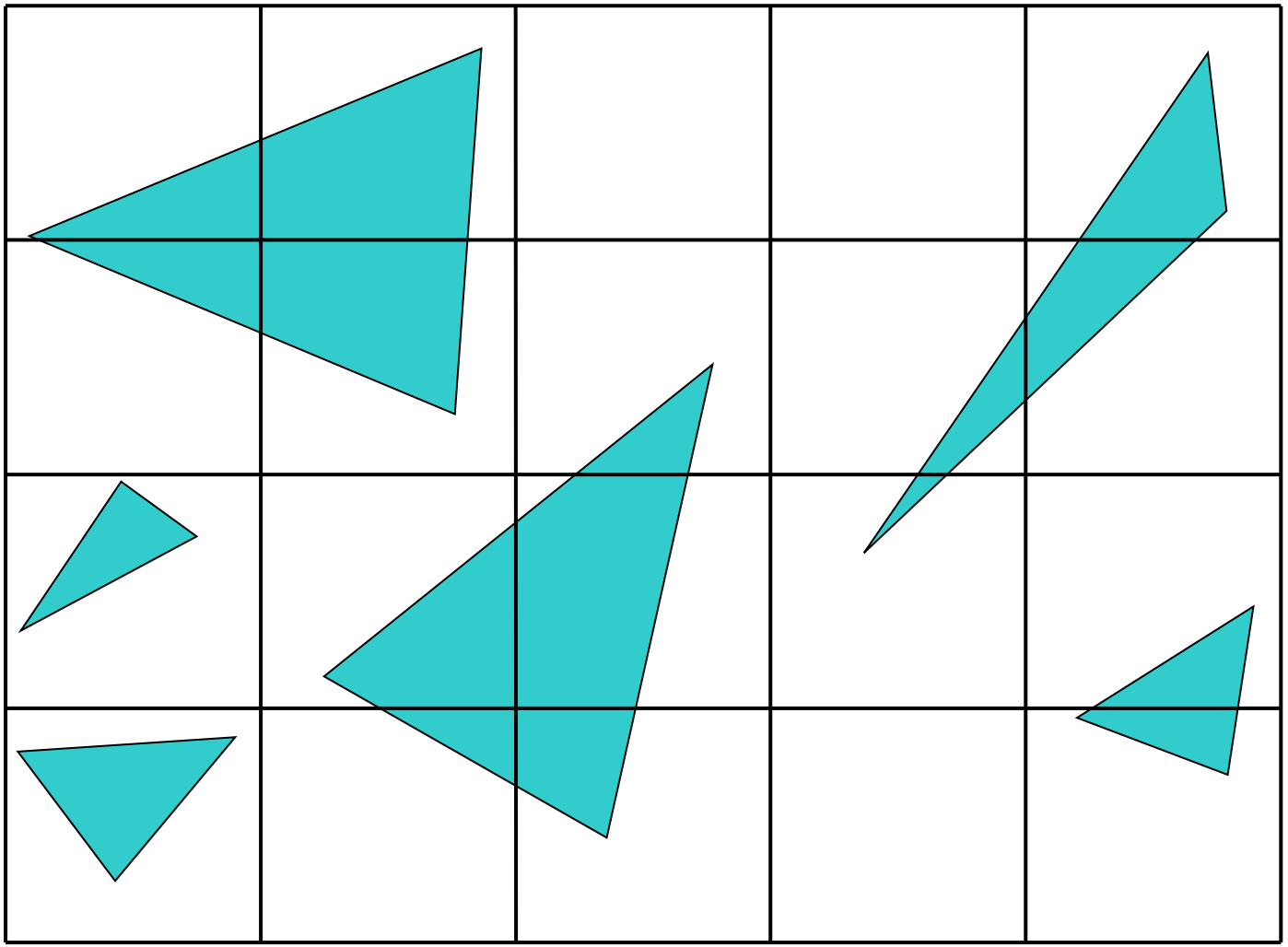
- Tester l'intersection du rayon avec **tous** les objets
- N objets, M rayons $\Longrightarrow$ O(NM)
- Trop coûteux !

Objectif

- Faire en sorte que la 1ère intersection calculée soit la bonne
- En pratique : compromis

Solution : partitionnement de l'espace
(souvent hiérarchique)

Grille régulière



Grille régulière

Construction

- Subdivision de la boîte englobante
- Résolution : souvent $\sim \sqrt[3]{n}$
- Une cellule : liste des objets l'intersectant

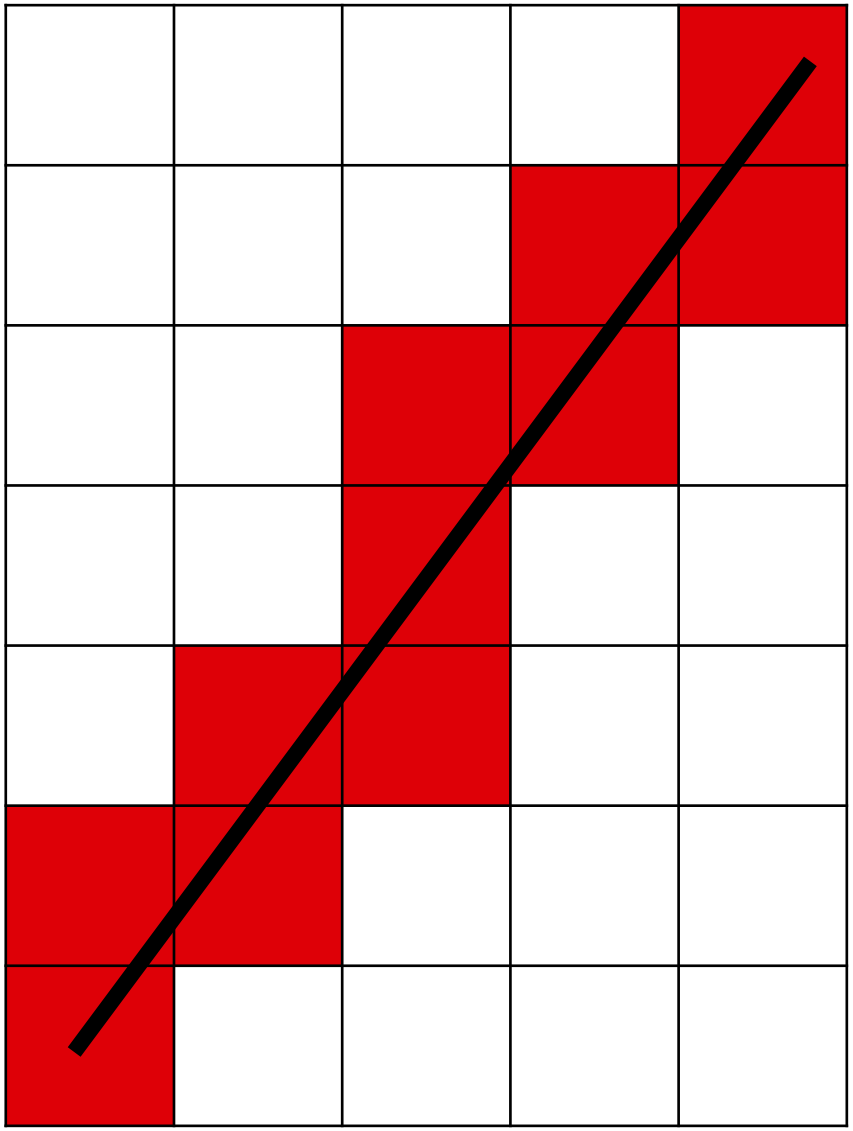
Parcours

- De proche en proche
- De l'origine vers l'arrière
- Arrêt si une intersection est trouvée

Parcours dans la grille

3DDDA

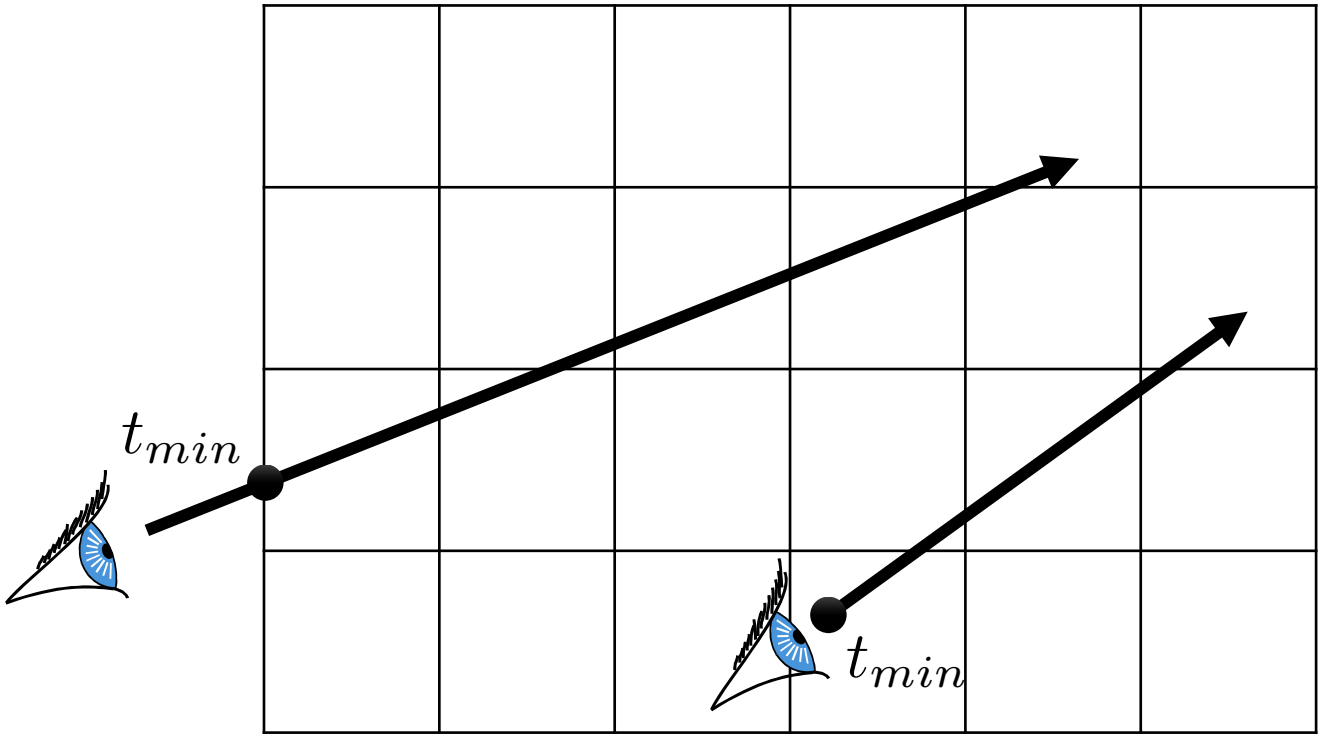
Three Dimensional Digital Difference Analyzer



Initialisation

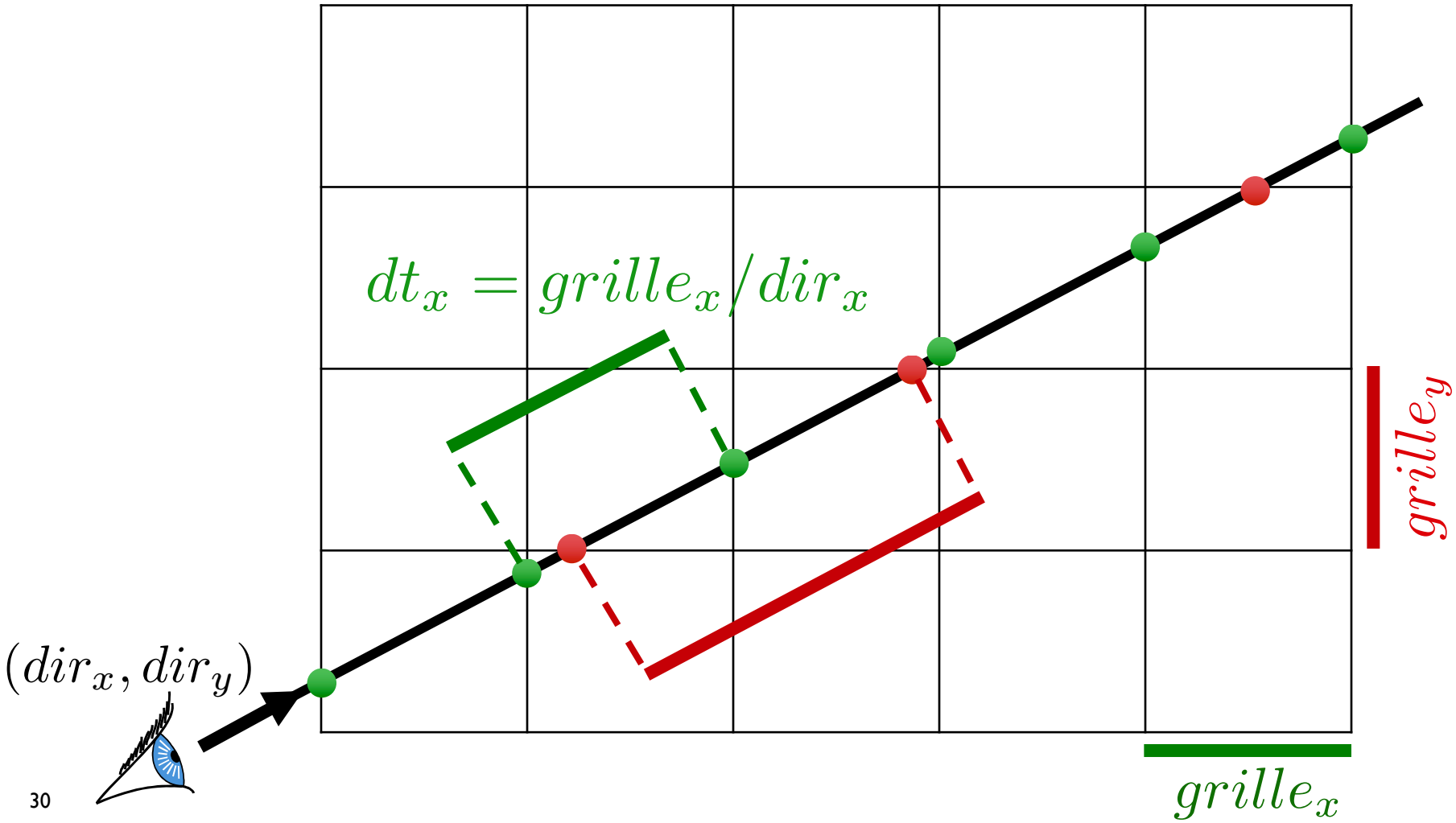
Calculer l'intersection avec la boite englobante t_{min}

(Attention, l'origine du rayon peut être dans la boîte)



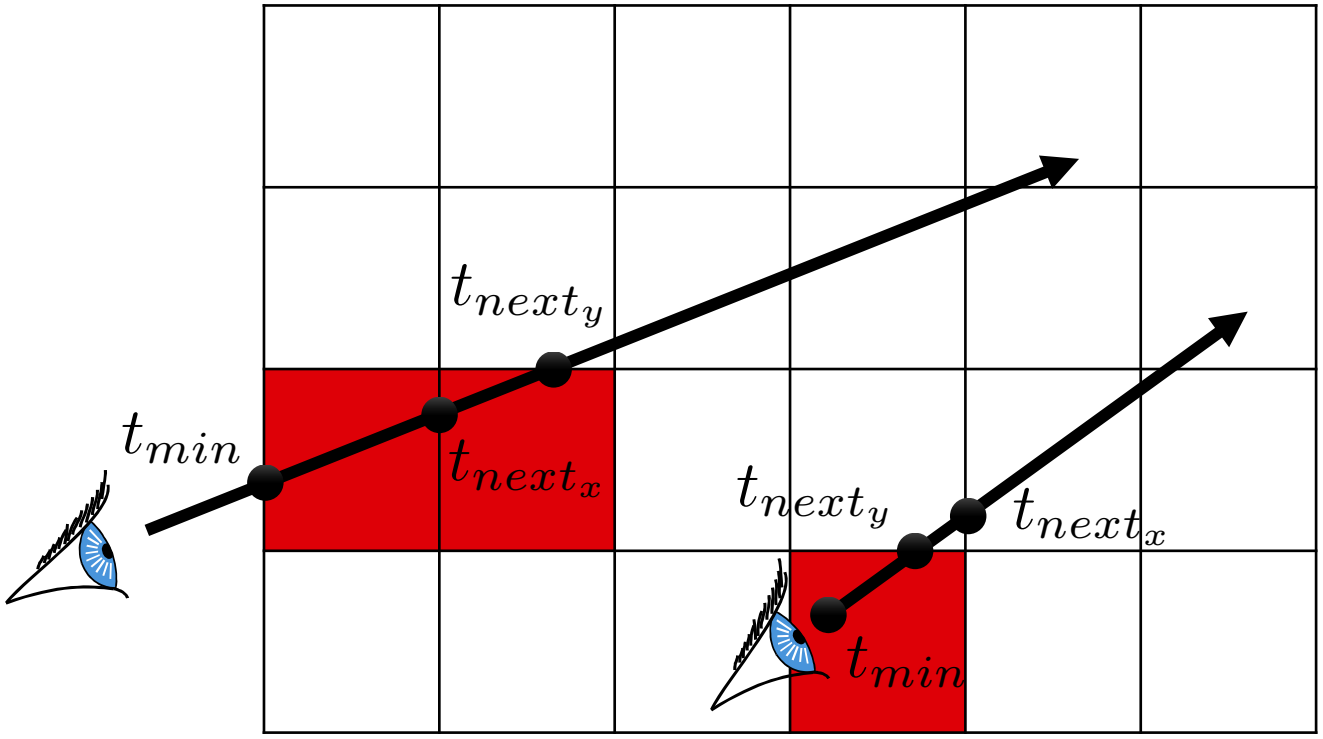
L'intersection est répétitive

Les intersections sur les axes sont équidistantes



Choisir la cellule suivante

Calculer les 2 intersections suivantes avec les axes $\begin{cases} t_{next_x} \\ t_{next_y} \end{cases}$



Choisir la cellule suivante

if $t_{next_x} < t_{next_y}$ **then**

$i \leftarrow i + sign_x$

$t_{min} = t_{next_x}$

$t_{next_x} \leftarrow t_{next_x} + dt_x$

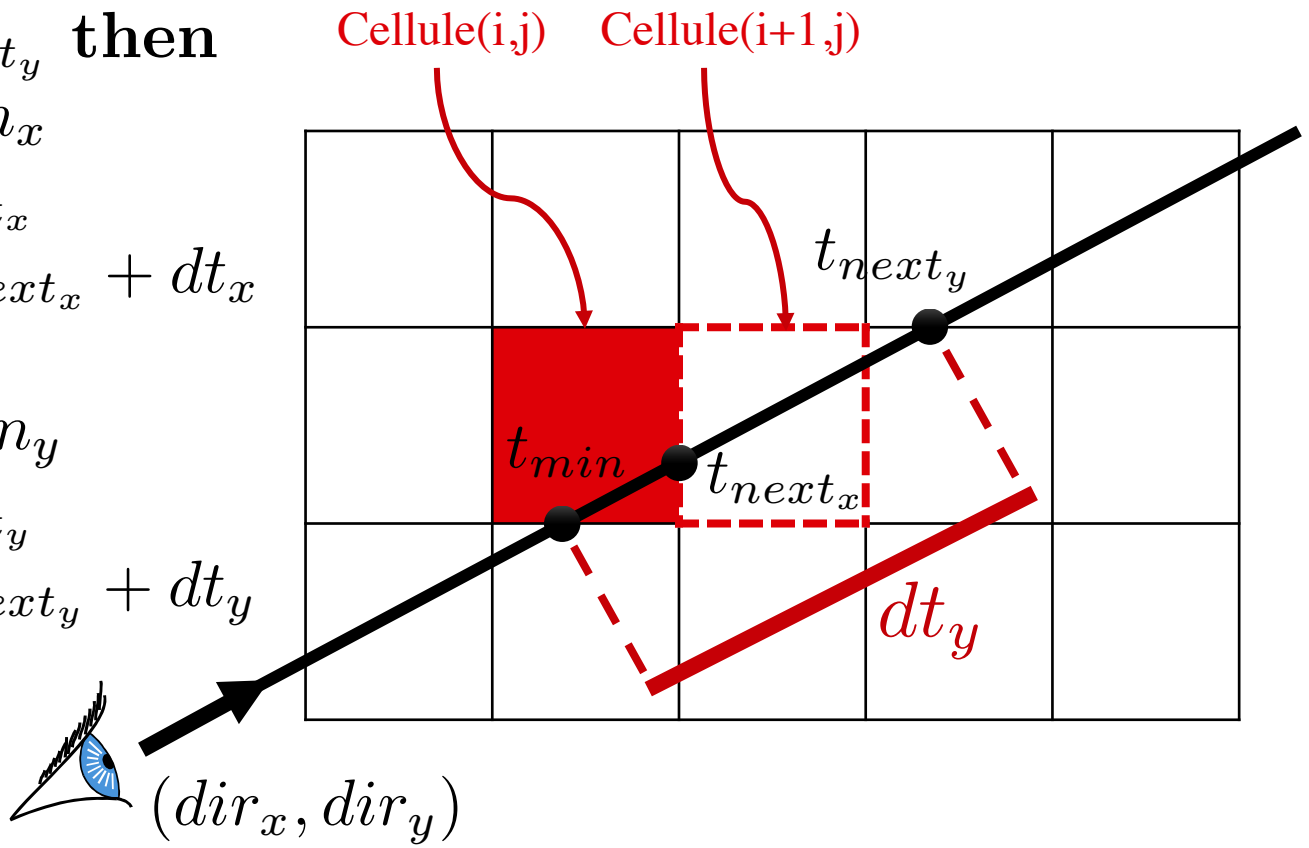
else

$j \leftarrow j + sign_y$

$t_{min} = t_{next_y}$

$t_{next_y} \leftarrow t_{next_y} + dt_y$

end if



if $dir_n > 0$ **then** $sign_n \leftarrow 1$ **else** $sign_n \leftarrow -1$ **end if**

Choisir la cellule suivante

if $t_{next_x} < t_{next_y}$ **then**

$i \leftarrow i + sign_x$

$t_{min} = t_{next_x}$

$t_{next_x} \leftarrow t_{next_x} + dt_x$

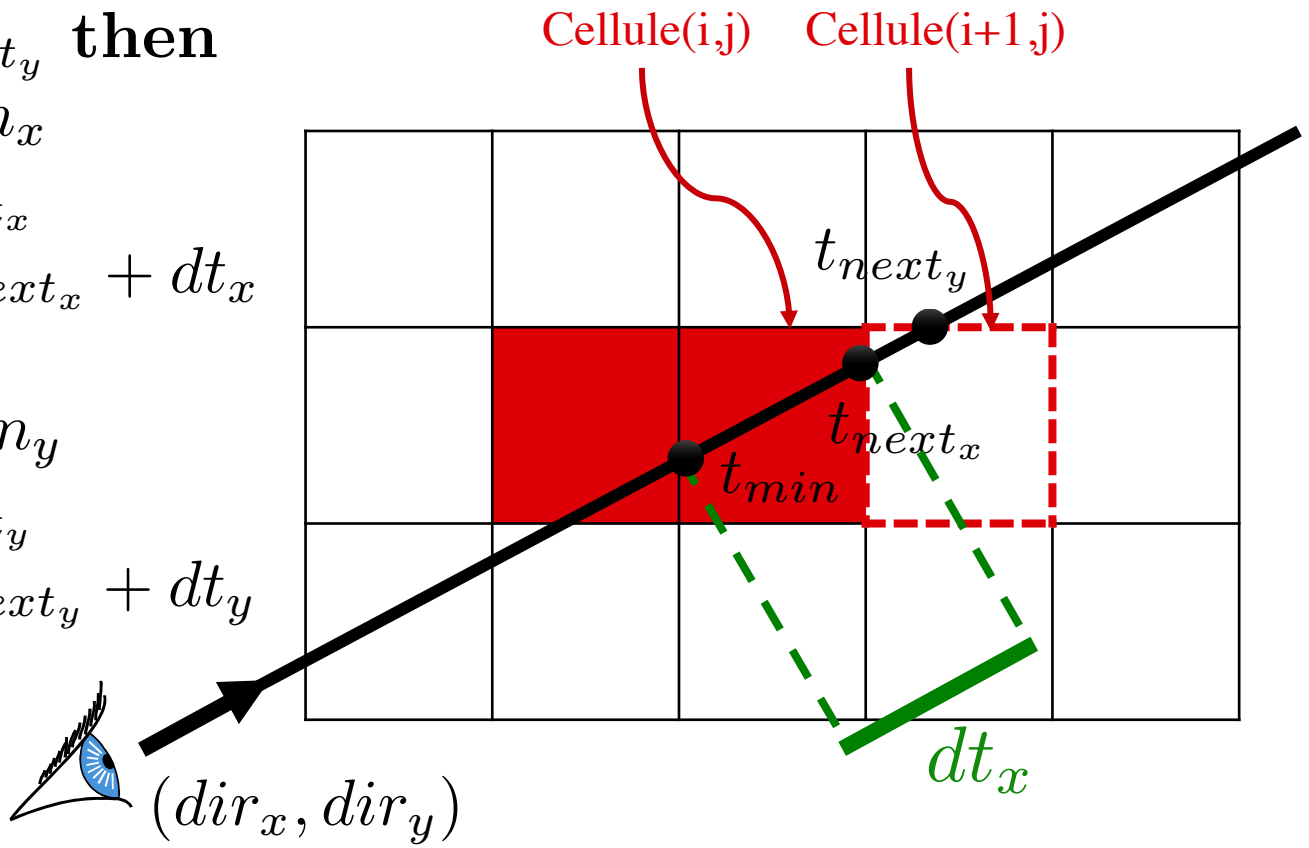
else

$j \leftarrow j + sign_y$

$t_{min} = t_{next_y}$

$t_{next_y} \leftarrow t_{next_y} + dt_y$

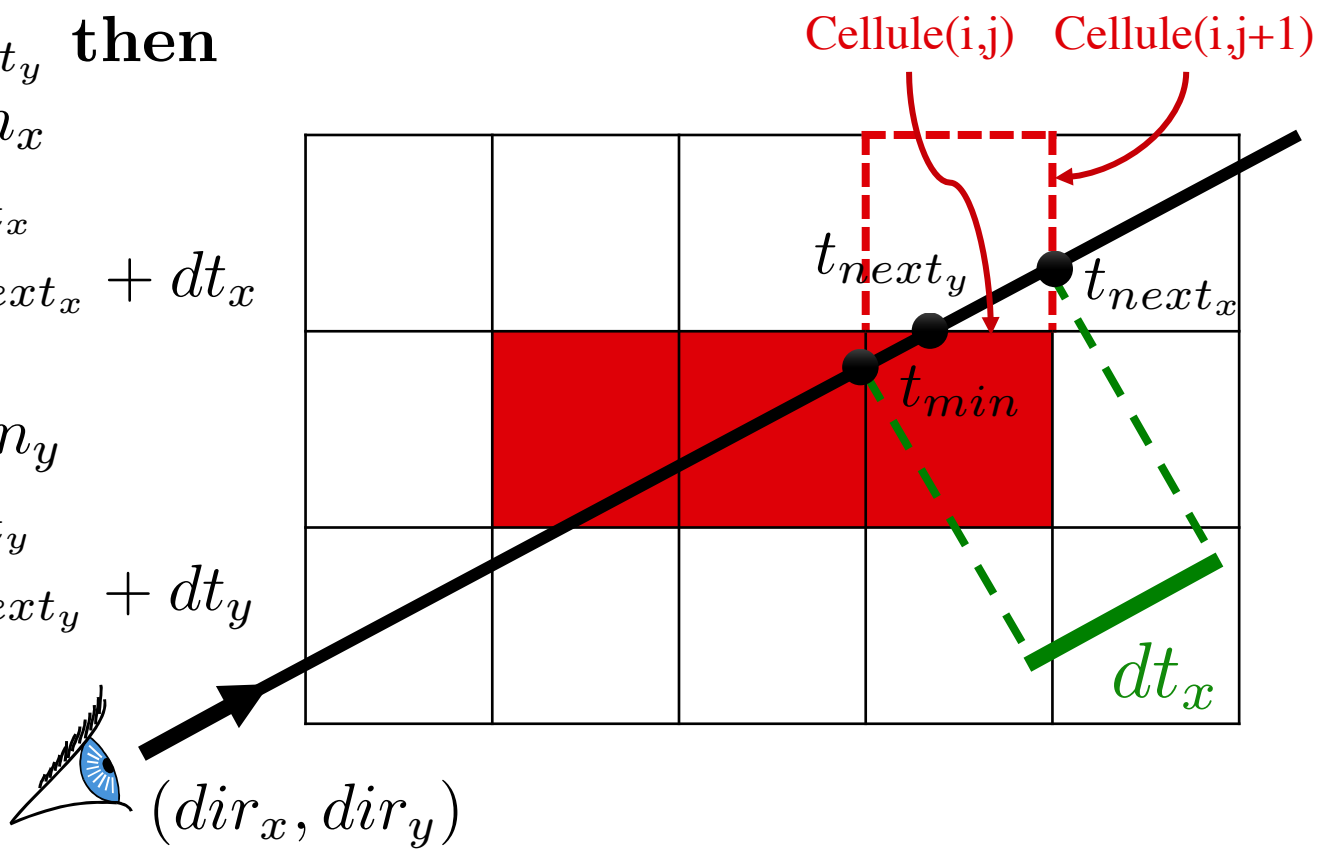
end if



if $dir_n > 0$ **then** $sign_n \leftarrow 1$ **else** $sign_n \leftarrow -1$ **end if**

Choisir la cellule suivante

if $t_{next_x} < t_{next_y}$ **then**
$i \leftarrow i + sign_x$
$t_{min} = t_{next_x}$
$t_{next_x} \leftarrow t_{next_x} + dt_x$
else
$j \leftarrow j + sign_y$
$t_{min} = t_{next_y}$
$t_{next_y} \leftarrow t_{next_y} + dt_y$
end if



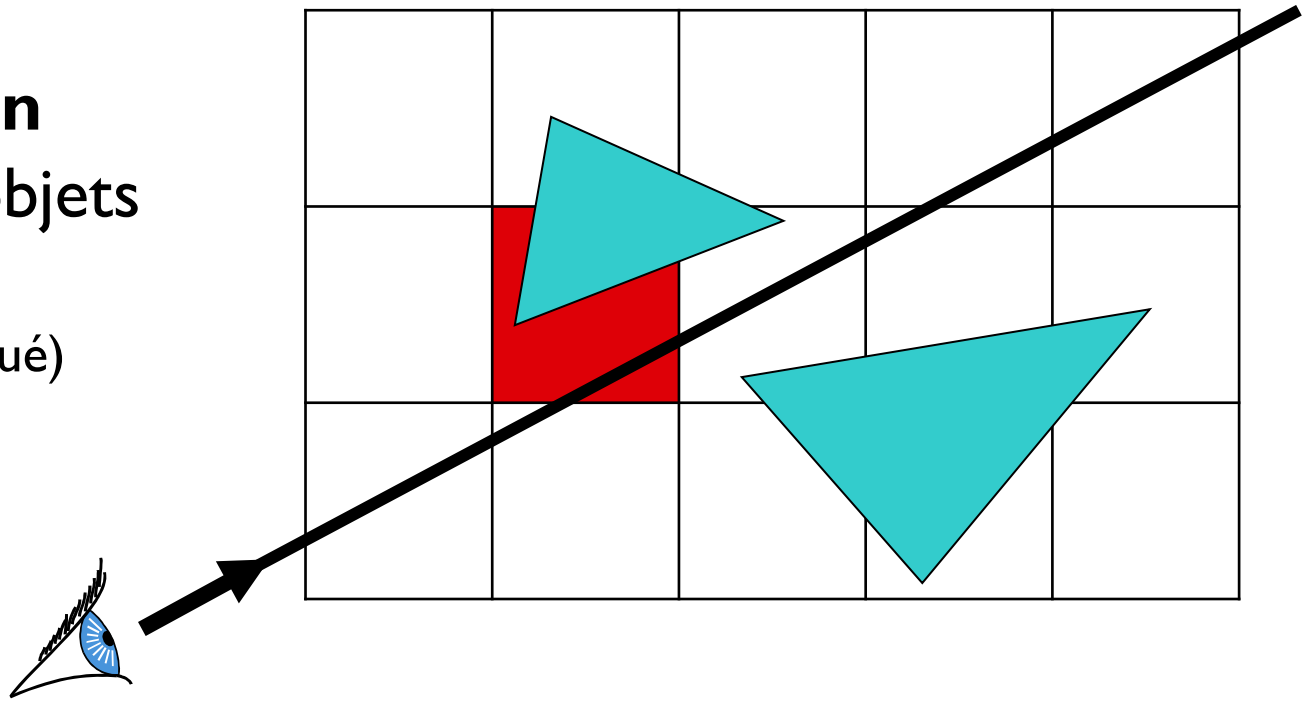
if $dir_n > 0$ **then** $sign_n \leftarrow 1$ **else** $sign_n \leftarrow -1$ **end if**

Test à faire sur chaque cellule

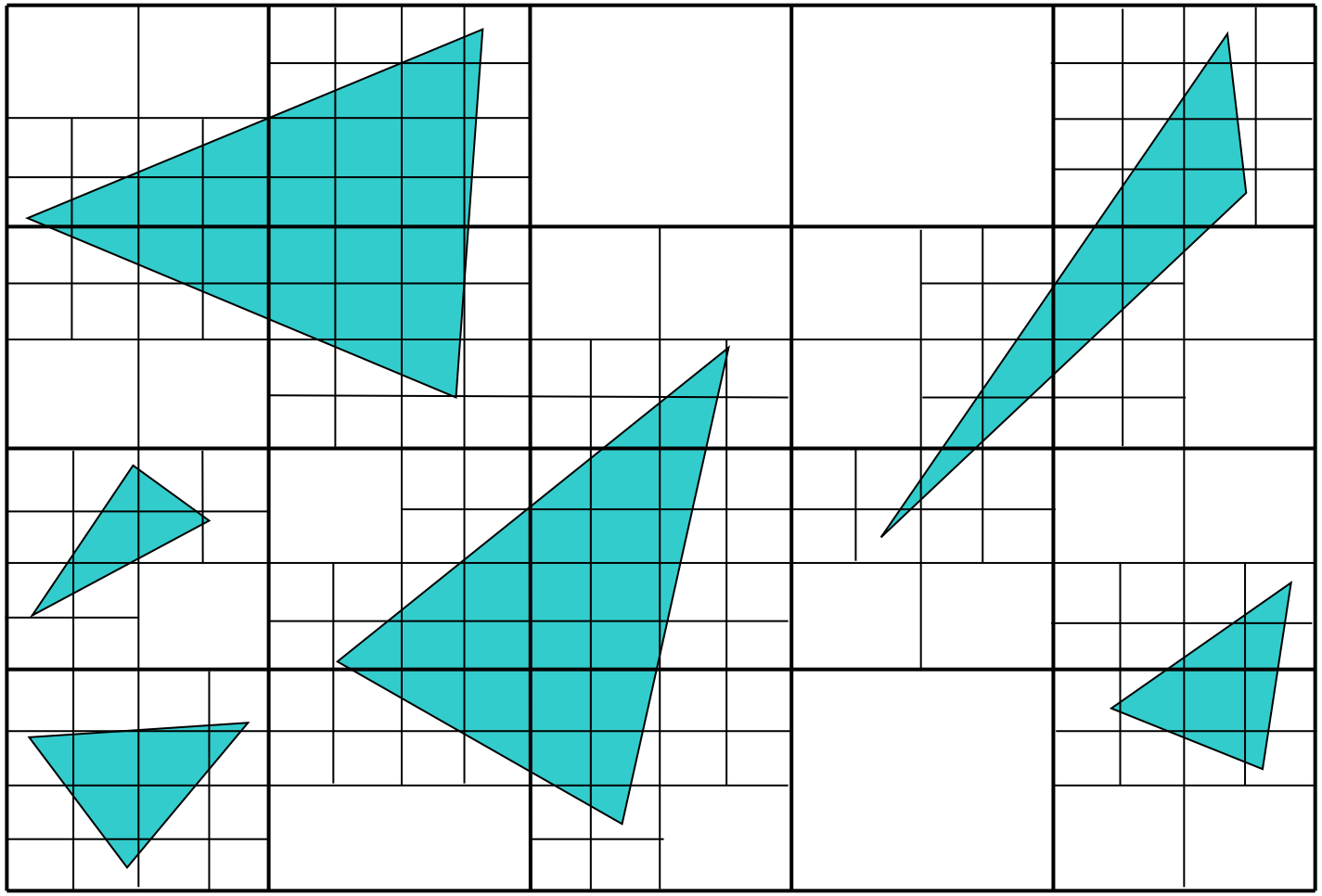
Intersection(s) dans la cellule ?

- Oui : retourner la plus proche
- Non : continuer

Optimisation
marquer les objets
déjà testés
(mais plus compliqué)



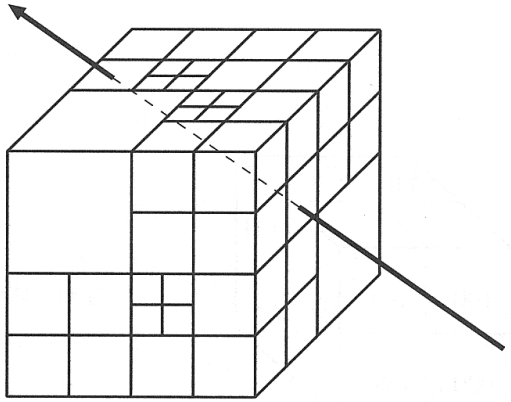
Grille adaptative : Octree [Meagher '80]

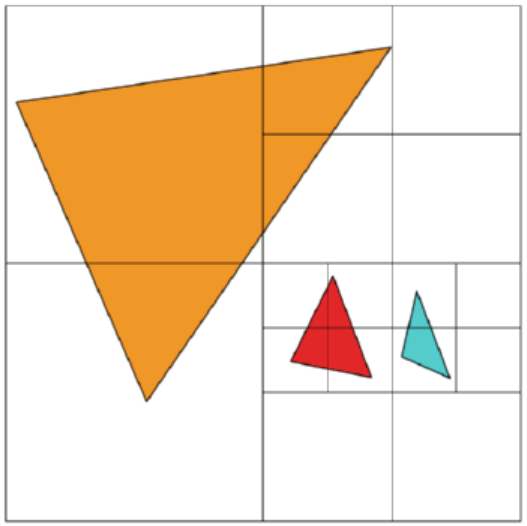


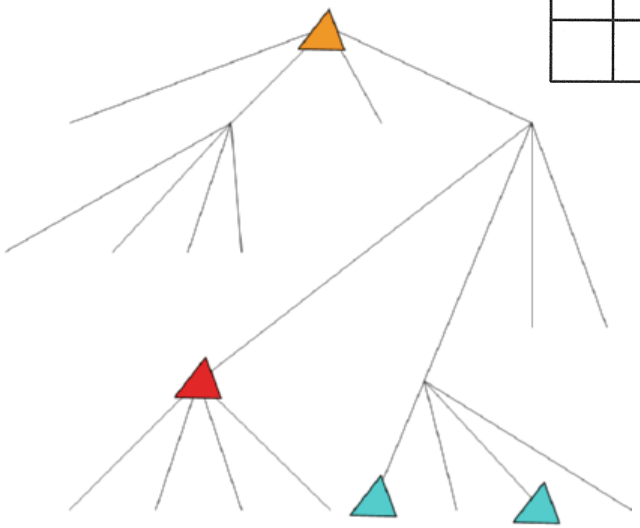
Grille adaptative : Octree

Partitionnement hiérarchique de l'espace

- Subdivise adaptivement chaque voxel en **8** sous-voxels de manière récursive
- Différents critères possibles
  - nombre de primitives par cellules
  - ratio de « vide »







Question – 3 mn

Comparer grille régulière et Octree

Comparaison

Grille régulière

- ✓ construction facile et rapide
- ✓ parcours simple
- ✗ nombreuses cellules vides
- ✗ cellules avec beaucoup d'objets
- ✗ choix de la résolution

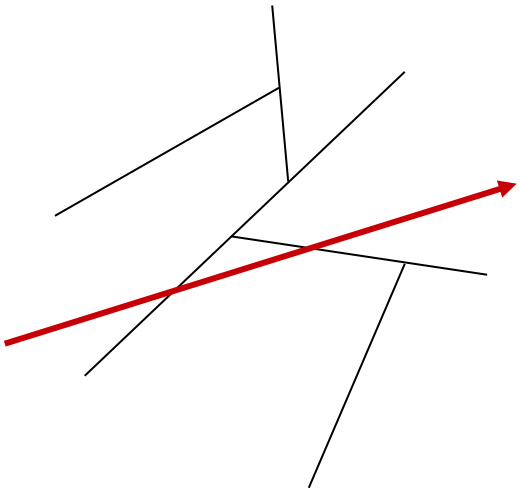
Octree

- ✓ initialisation rapide
- ✓ peu de cellules vides
- ✗ parcours récursif couteux
- ✗ convergence lente pour les zones complexes

Grille adaptative : BSP- et Kd-Trees

Arbres binaires

- Nœud = plan de subdivision de l'espace

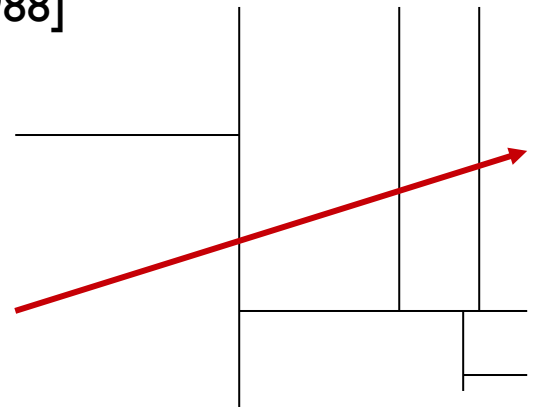


Binary Space Partition [Fuchs et al. 1980]

- Plans quelconques « judicieusement » placés
- Couteux à construire, stocker et utiliser
-

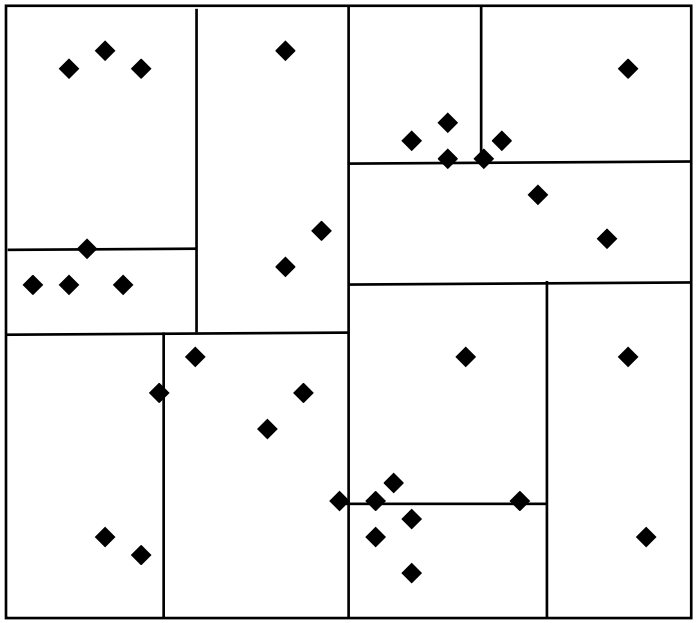
Kd-Tree [Bentley 1975, Fussell and Subramanian 1988]

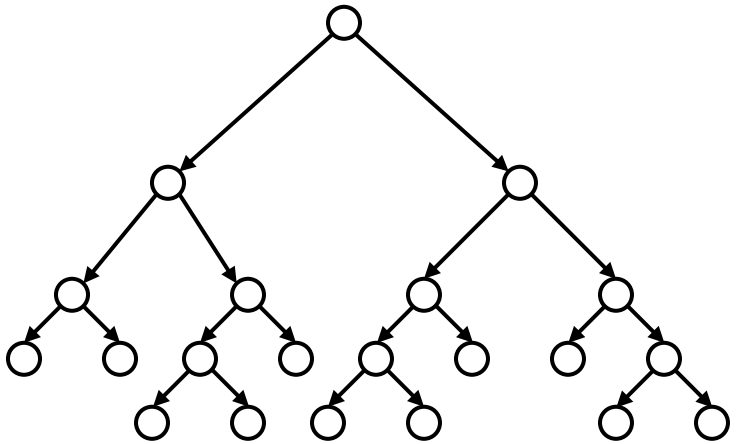
- Plans alignés sur les axes
- Simple, léger et très efficace



KD-tree : définition

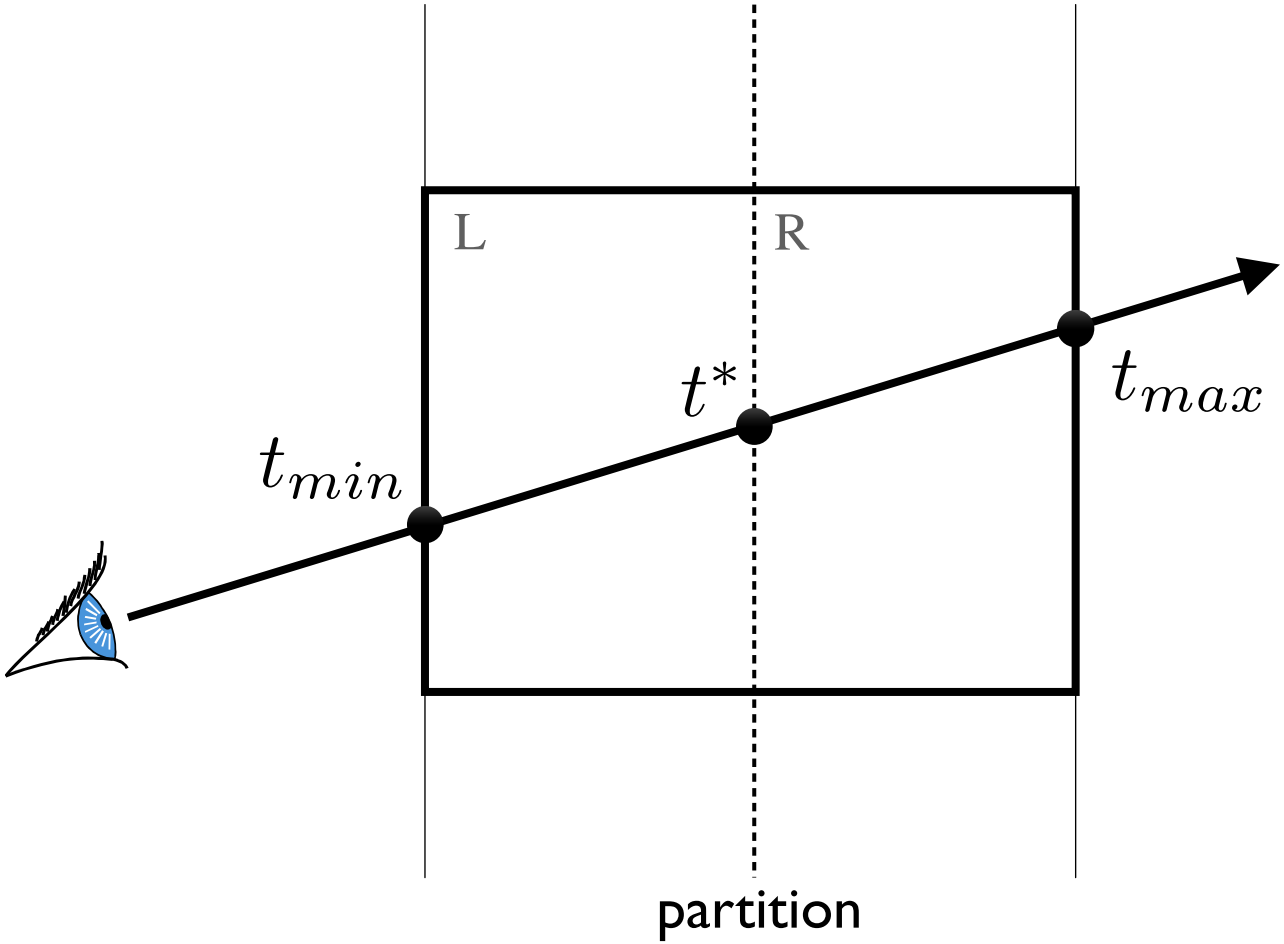
Subdivision récursive





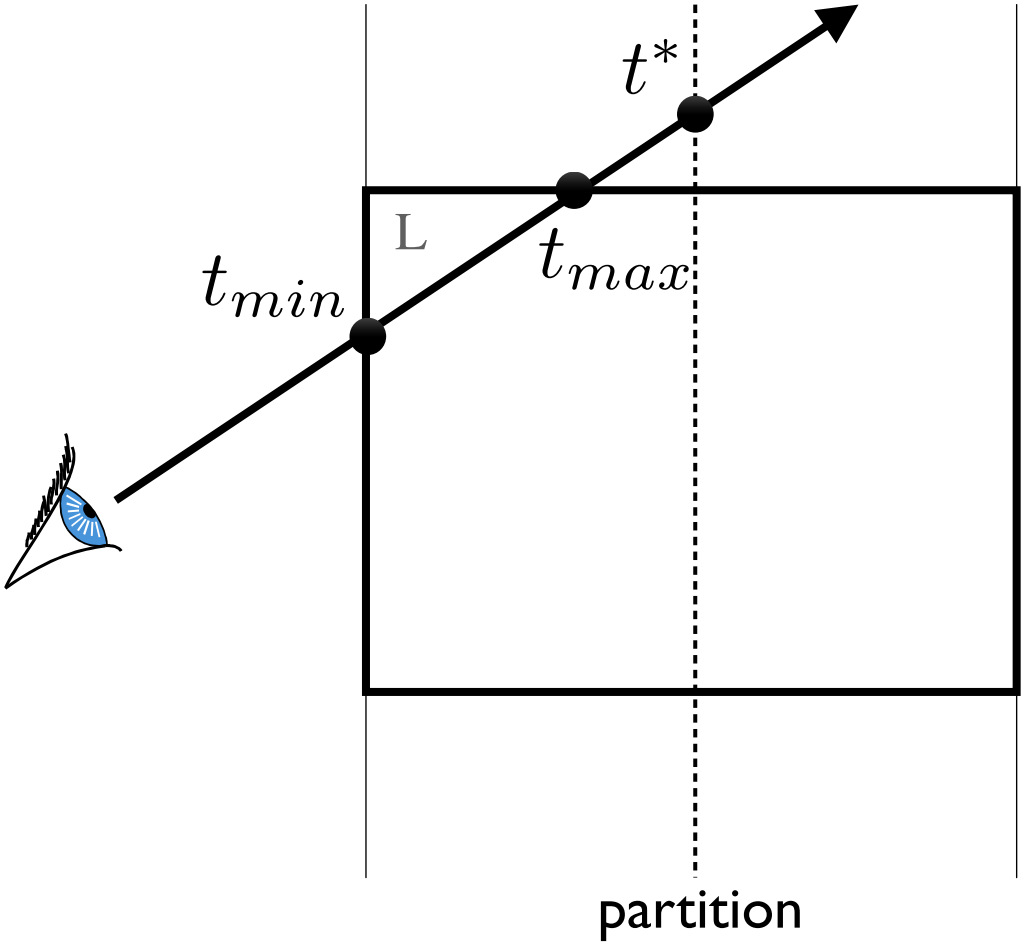
KD-tree : parcours récursif

$t_{min} < t_* < t_{max} \Rightarrow \text{Intersect}(\text{L}, t_{min}, t^*)$ et $\text{Intersect}(\text{R}, t^*, t_{max})$



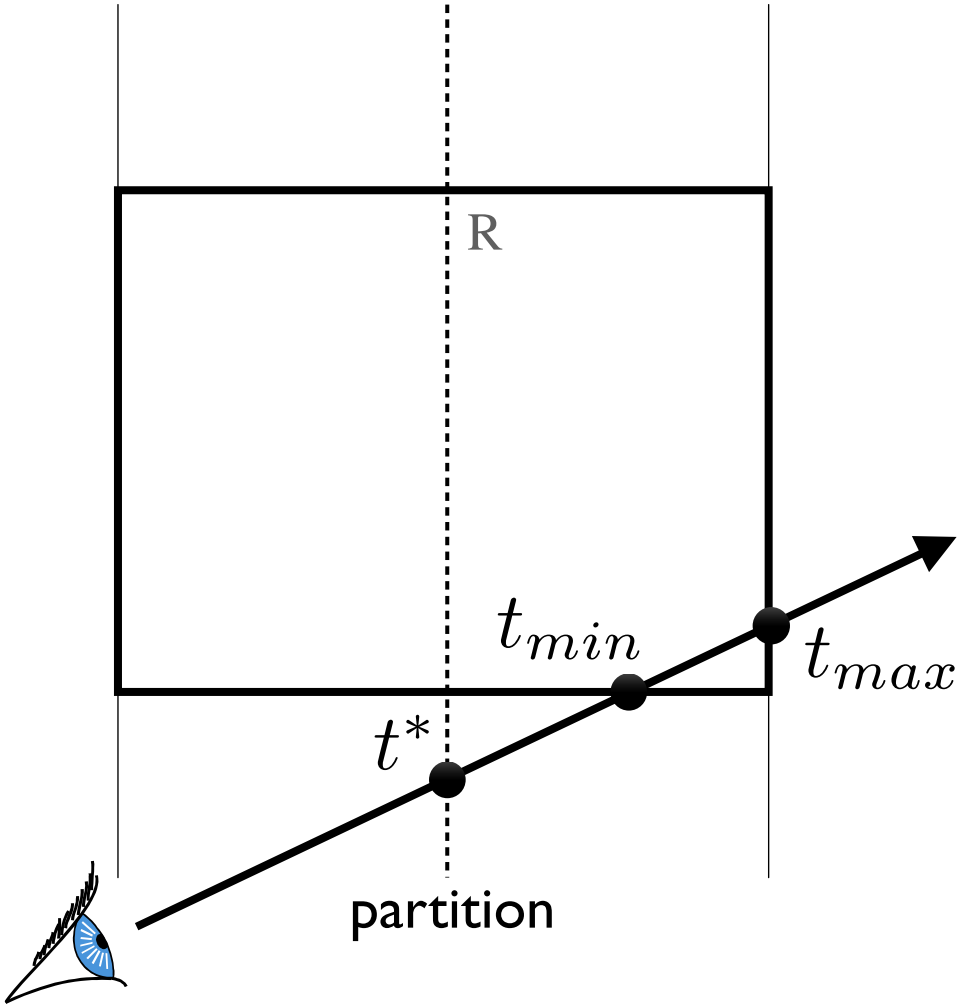
KD-tree : parcours récursif

$t_{max} < t^* \Rightarrow \text{Intersect}(\text{L}, t_{min}, t_{max})$



KD-tree : parcours récursif

$t_* < t^{min} \Rightarrow \text{Intersect}(\text{R}, t_{min}, t_{max})$



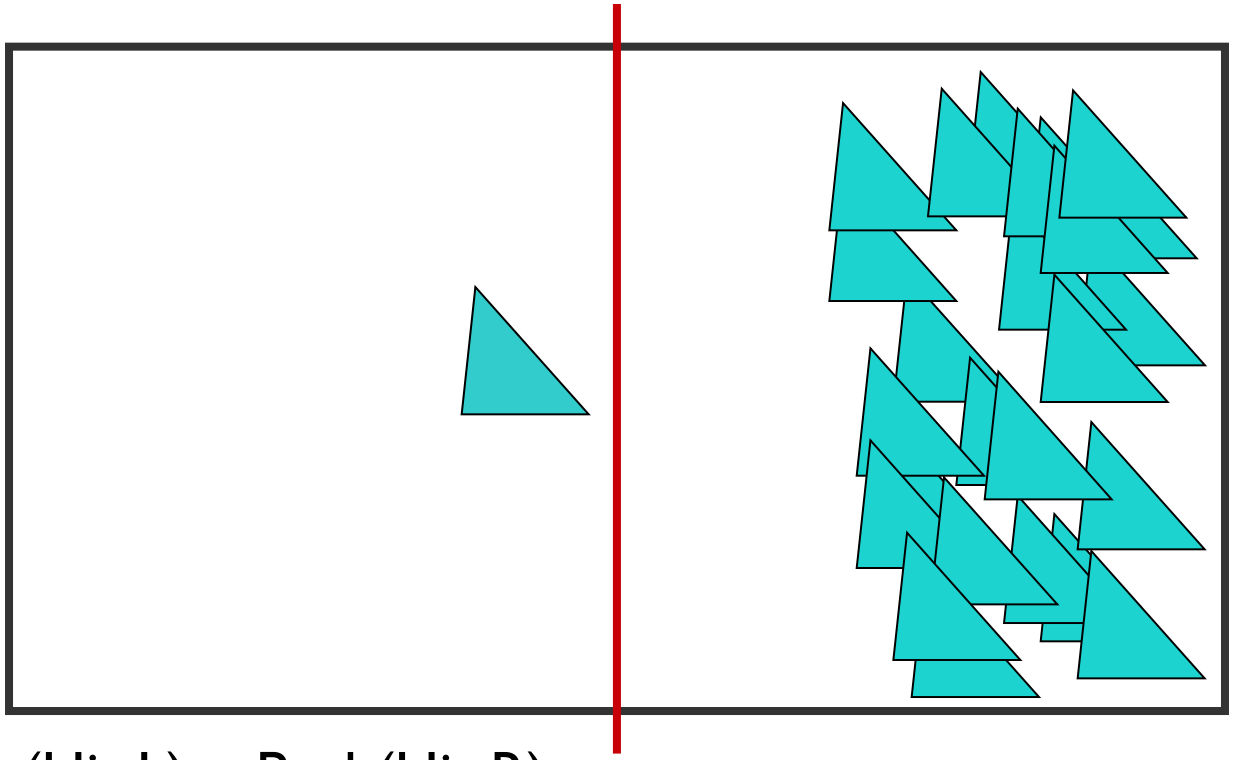
KD-tree : construction

Algo « naïf »

- Axe de coupe : le long de la plus grande dimension
- Position de coupe : au centre ou médian de la géométries (arbre équilibré)
- Critère d'arrêt : nombre de primitives, profondeur max.

KD-tree : construction

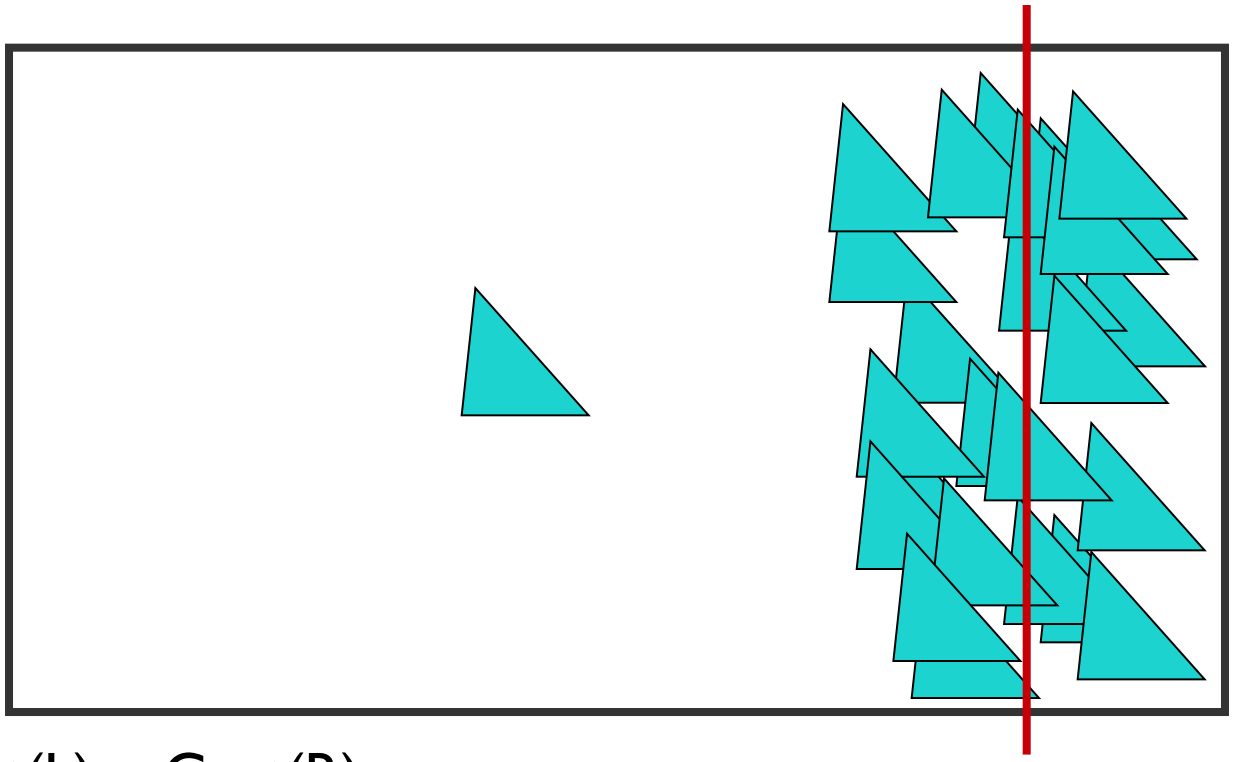
Couper au milieu



- Prob(Hit L) = Prob(Hit R)
- Ne prend pas en compte le coût de L & R

KD-tree : construction

Couper à la médiane



- Cost(L) = Cost(R)
- Ne prend pas en compte les probabilités d'entrer dans L et R

KD-tree : construction

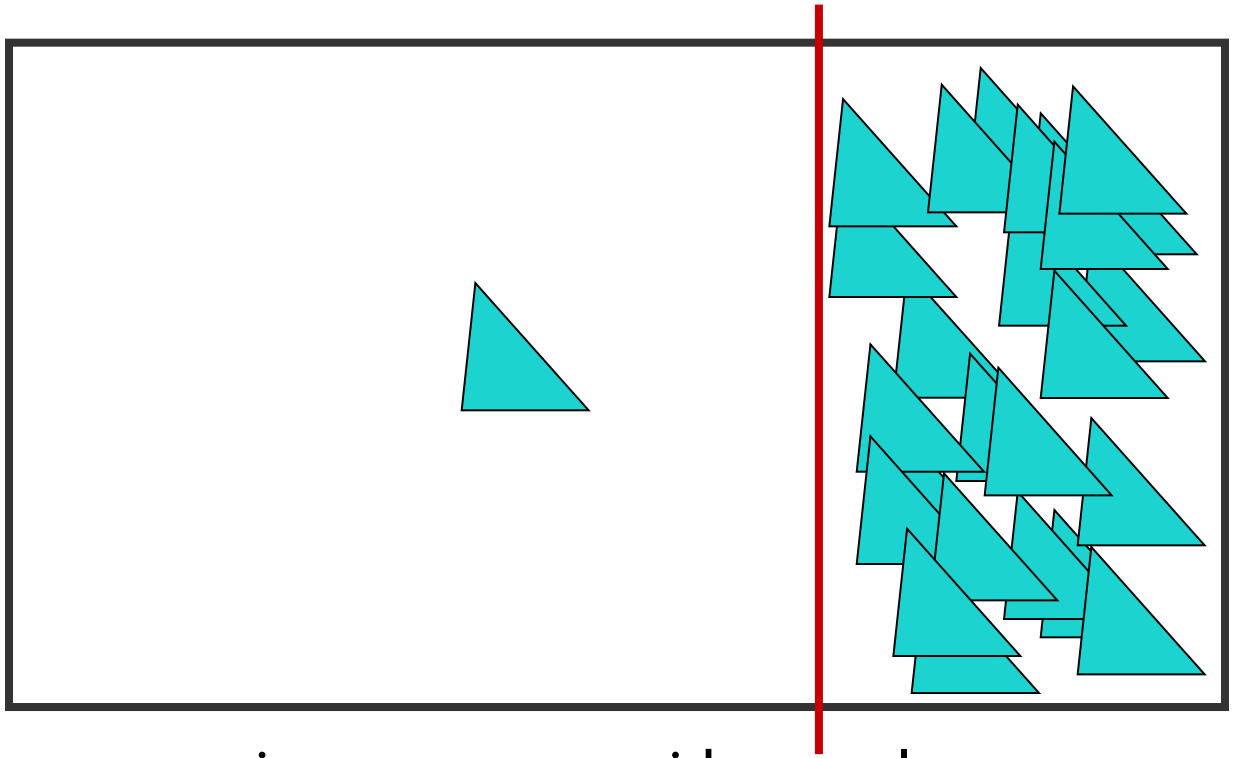
Algo « intelligent »

- Objectif : choisir le plan de coupe qui rend le tracé de rayon le moins couteux possible
- Définir un modèle de coût et le minimiser
- Quel est le coût de tracer un rayon au travers d'une cellule ?

Cost(cell) = C_trav + Prob(hit L) x Cost(L) + Prob(hit R) x Cost(R)

KD-tree : construction

Optimisation de la fonction de coût



- Isole automatiquement et rapidement les zones complexes
- Génère de grands espaces vides / concentre les primitives dans de petits nœuds

KD-tree : construction [MacDonald and Booth 1990]

Probabilité de rentrer dans une cellule

$\Longrightarrow$ proportionnel à l'aire de la surface de la cellule (SA)

Coût de parcours d'une cellule

$\Longrightarrow$ nombre de triangles (TriCount)

Cost(cell) = C_trav + Prob(hit L) x Cost(L) + Prob(hit R) x Cost(R)

= C_trav + SA(L) x TriCount(L) + SA(R) x TriCount(R)

Critère d'arrêt

$\Longrightarrow$ quand subdiviser ne réduit plus le modèle de coût (seuillage)

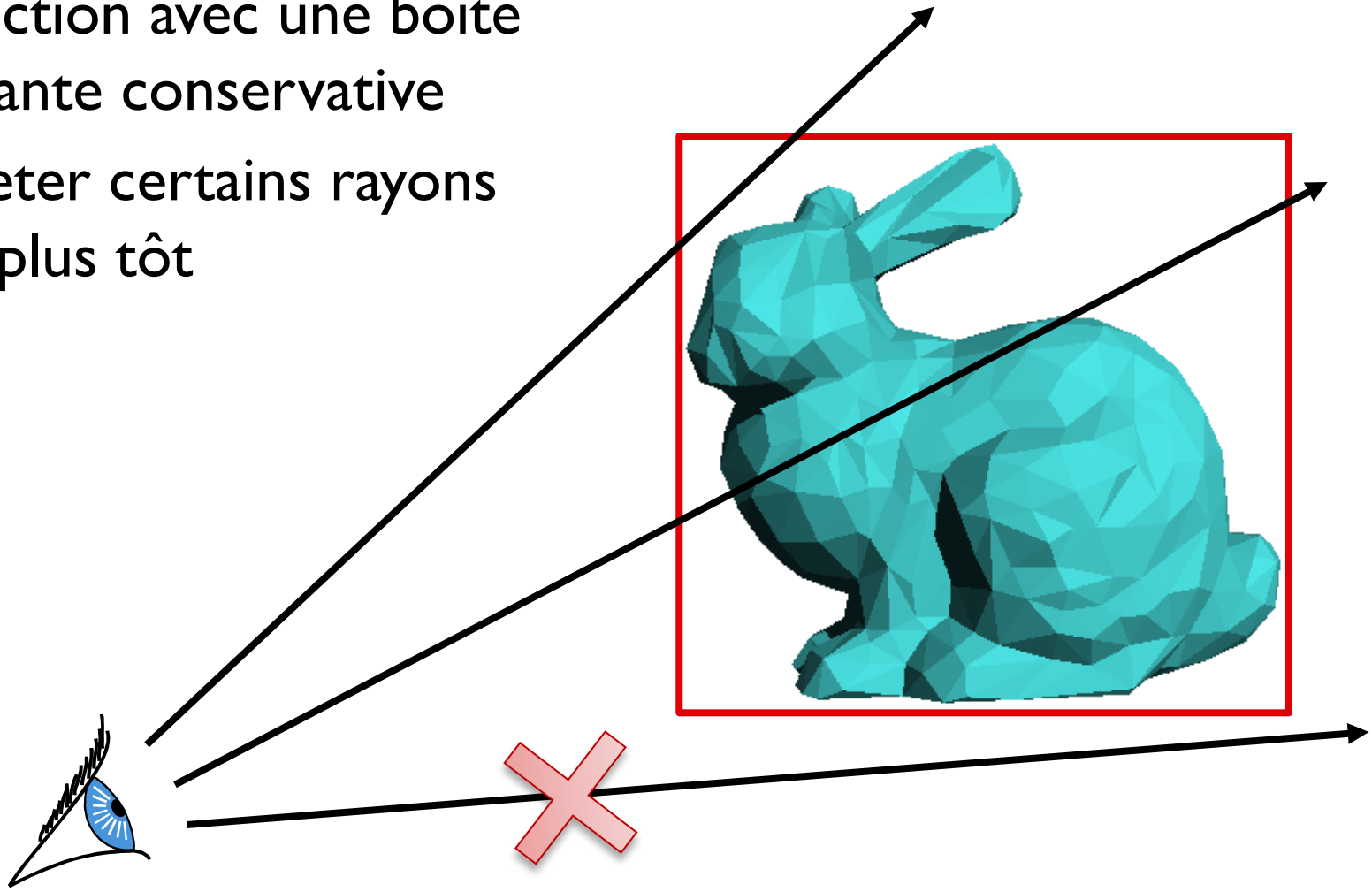
Résultats

$\Longrightarrow$ un « bon » KD-tree est de 2 à 5 fois plus rapide

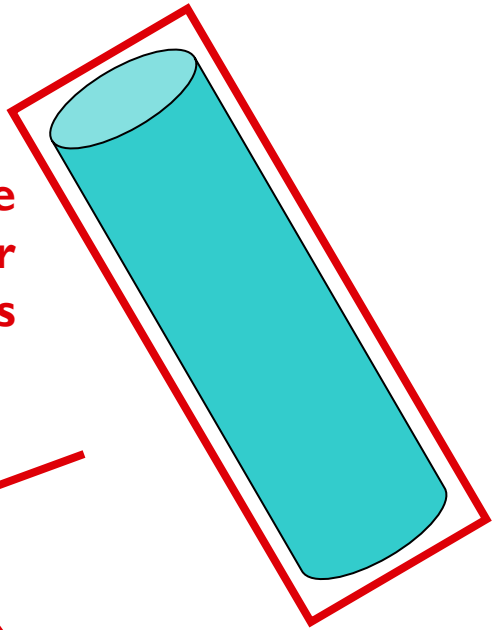
Boites englobantes

Intersection avec une boite englobante conservative

$\Longrightarrow$ rejeter certains rayons au plus tôt

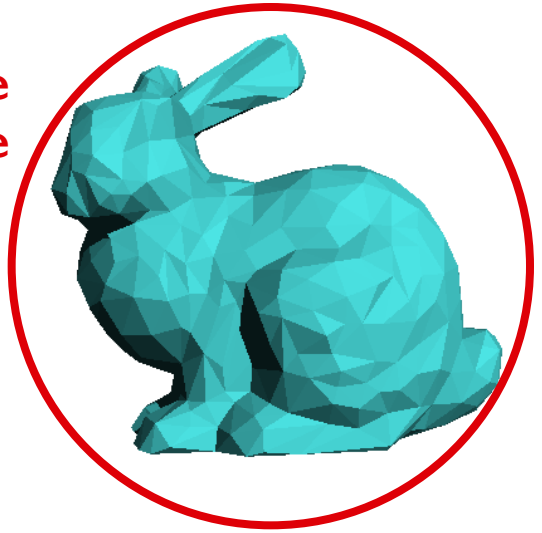


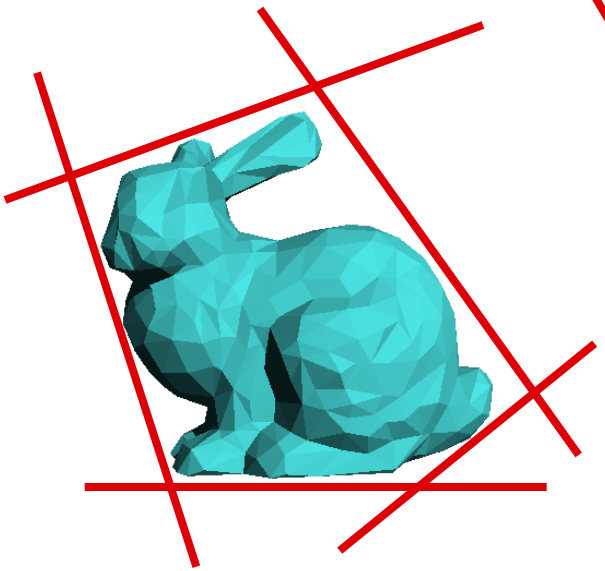
Boites englobantes



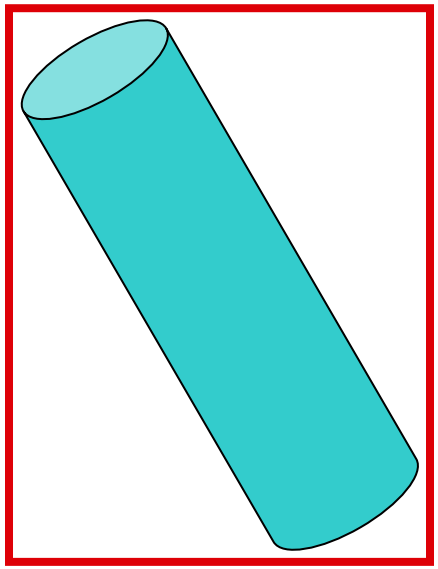
Boite englobante non-alignée sur les axes

Sphère englobante



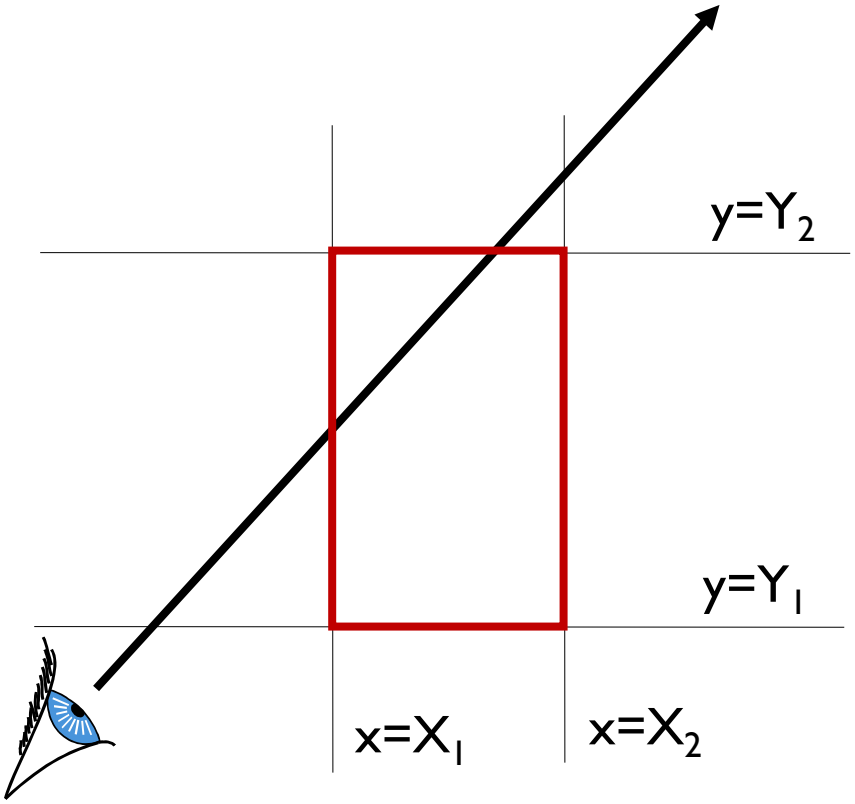


Région convexe arbitraire

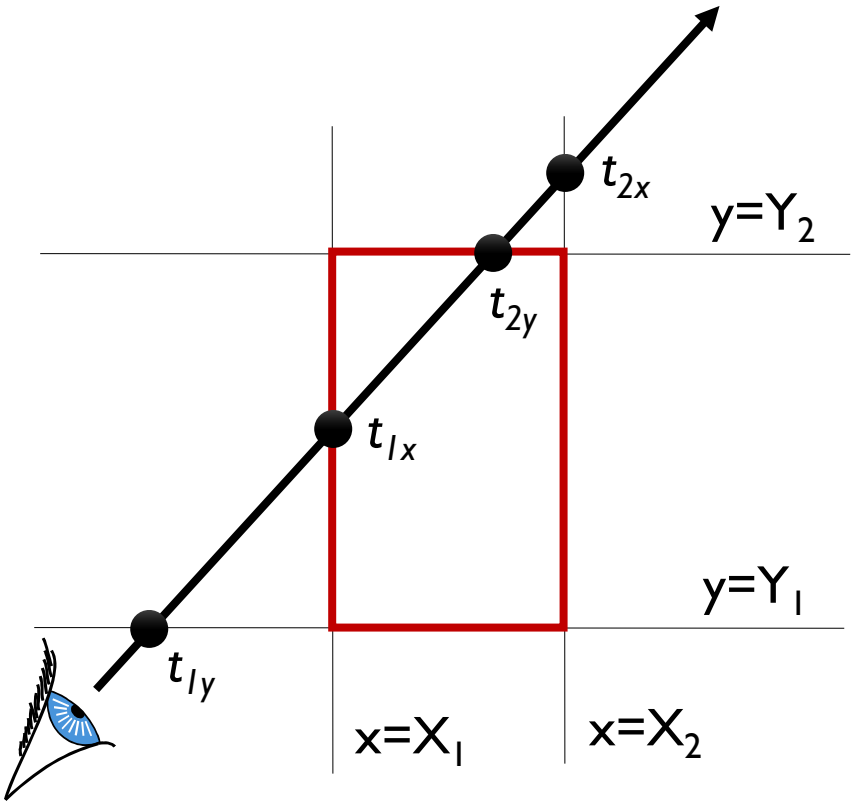


Boite englobante alignée sur les axes

Boîte englobante alignée sur les axes

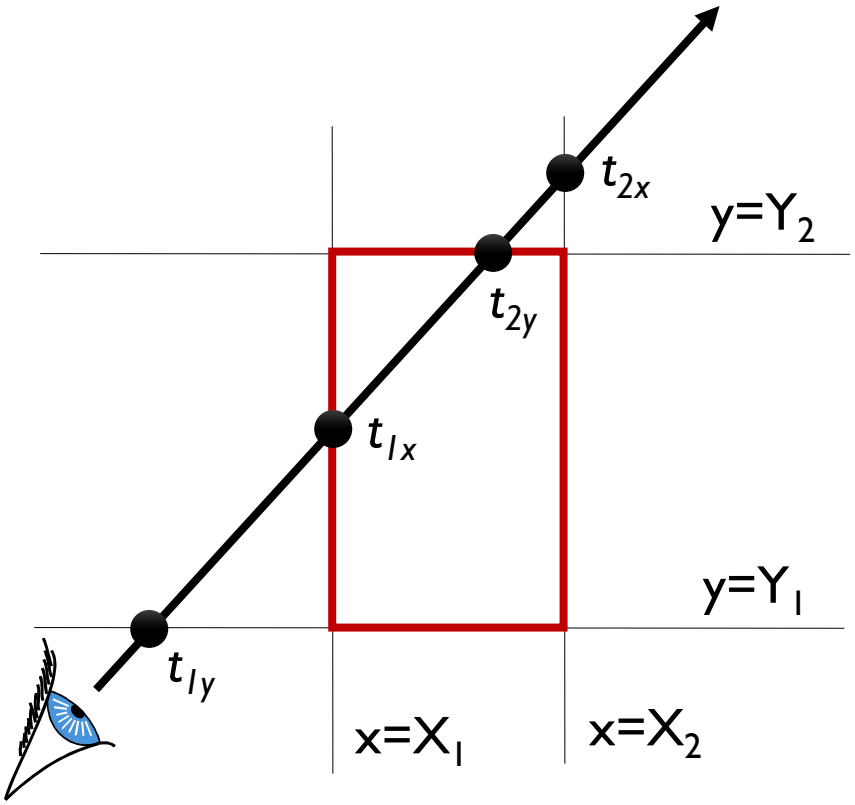


Boîte englobante alignée sur les axes



- Boîte = 6 plans
- Calculer toutes les intersections
- Conserver la plus proche **à l'intérieur** de la boîte

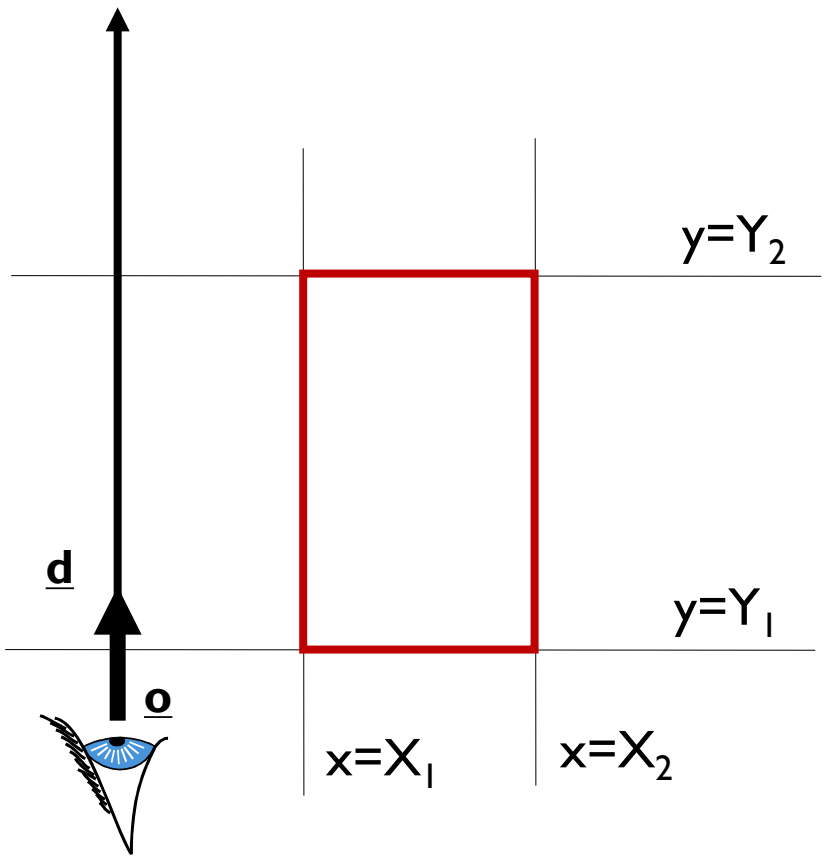
Simplifier les calculs



- Chaque paire de plan a la **mêmes normale**
- Une **seule** composante de la normale est **non-nulle**

$\Longrightarrow$ considérer une dimension à la fois

Tester si le rayon est parallèle



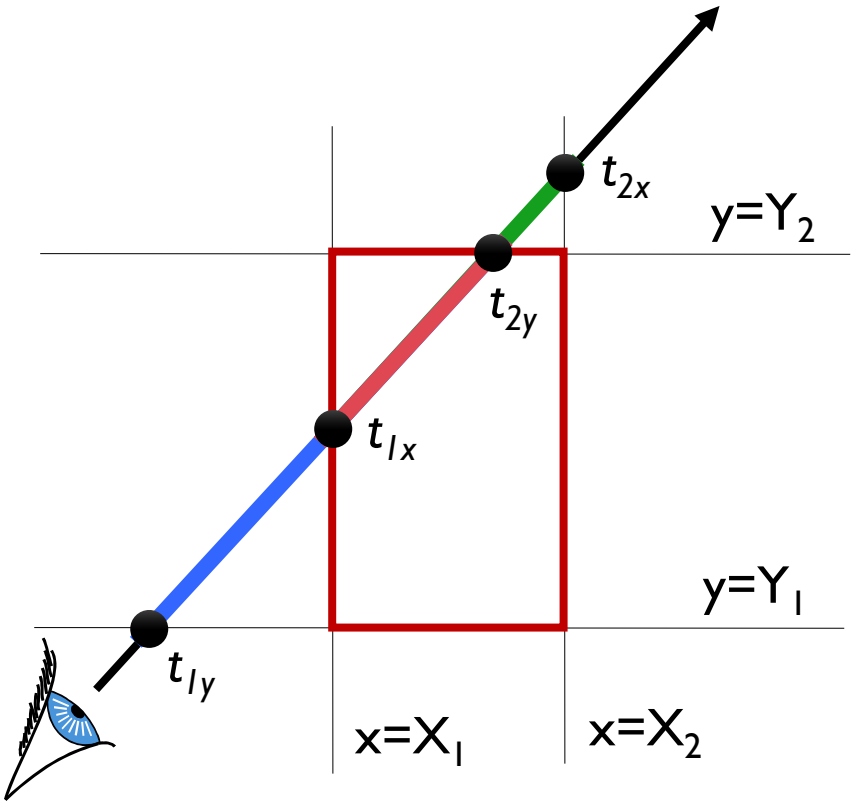
$\underline{d}_x = 0$

…et s'il n'y a pas d'intersection :

$\underline{o}_x < X_1$ ou $\underline{o}_x > X_2$

(Idem en Y et Z, bien-sûr)

Trouver les intersections par axe



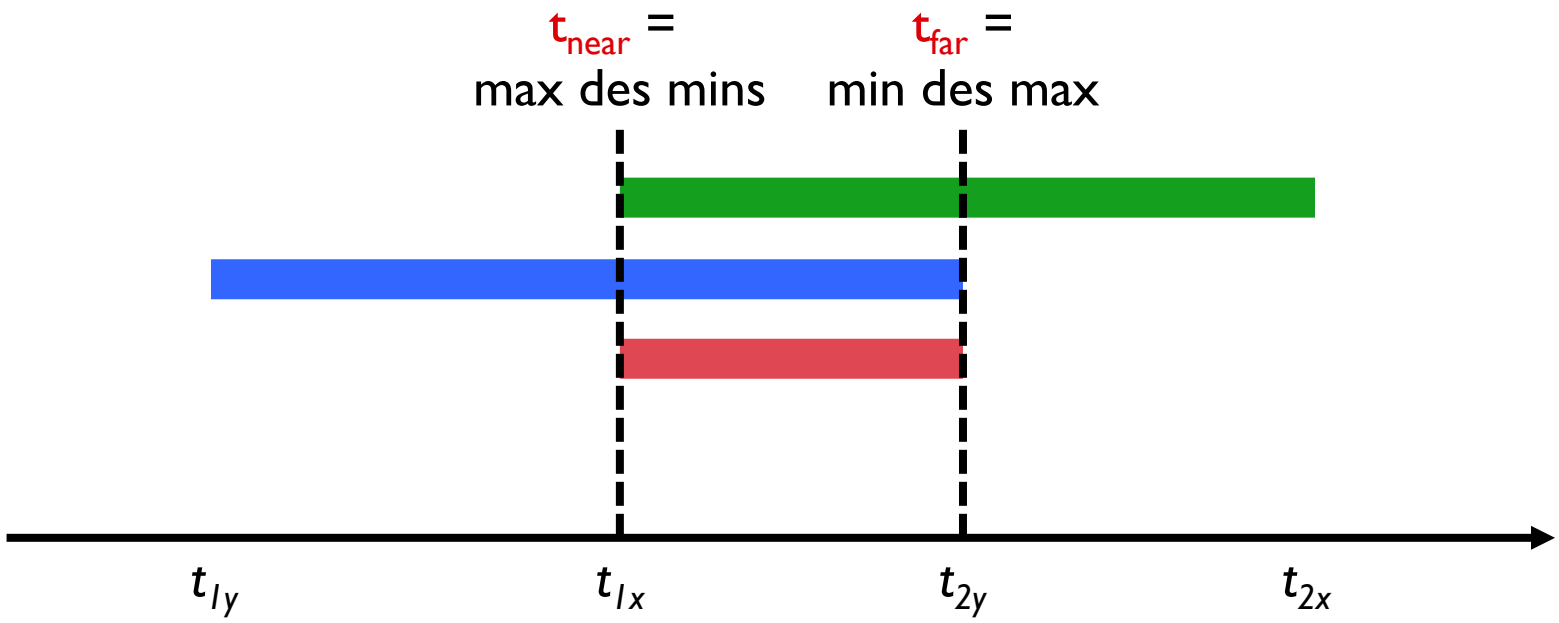
- Déterminer un intervalle par dimension
- Calculer l'intersection de ces intervalles 1D

Intervalle entre X_1 et X_2

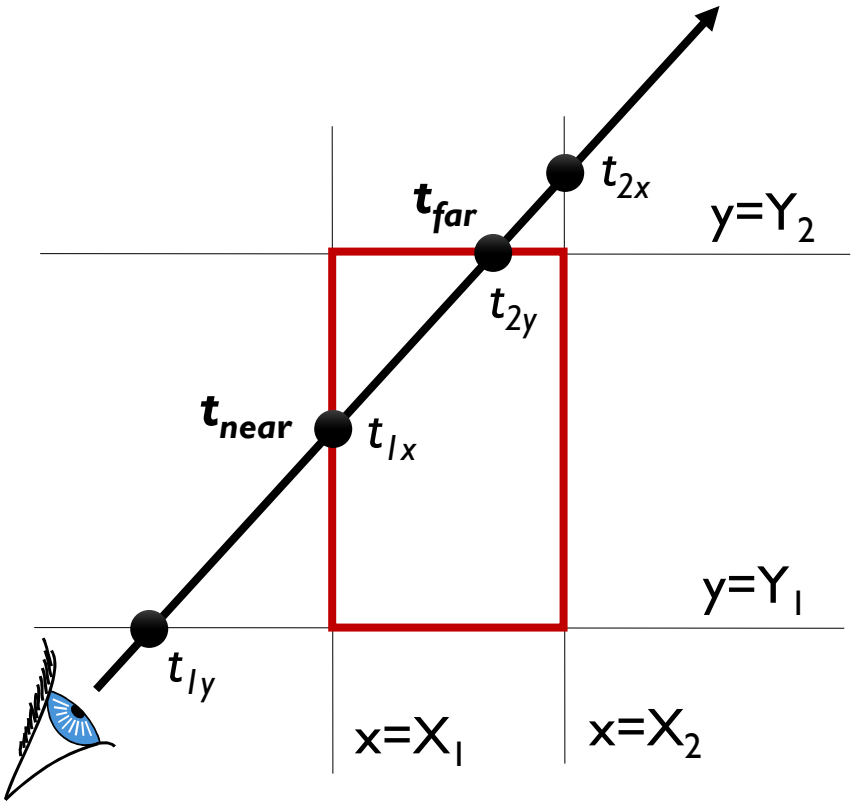
Intervalle entre Y_1 et Y_2

Intersection

Intersection d'intervalles 1D



Boîte englobante alignée sur les axes



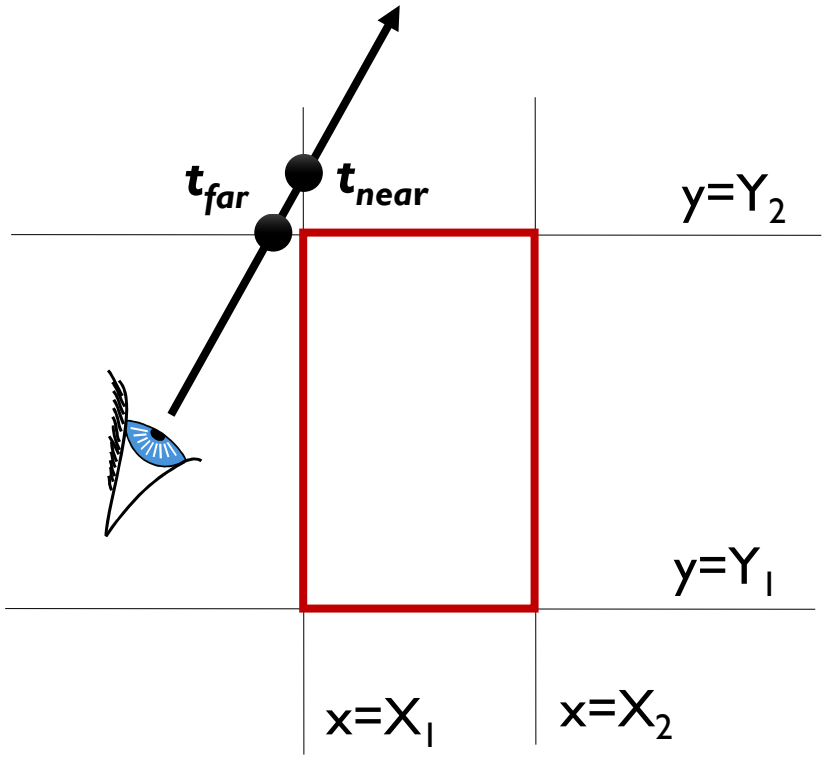
Calculer les distances d'intersection t_1 et t_2 pour chaque axe :

- $t_{1x} = (X_1 - o_x) / \mathbf{d_x}$
- $t_{2x} = (X_2 - o_x) / \mathbf{d_x}$

puis :

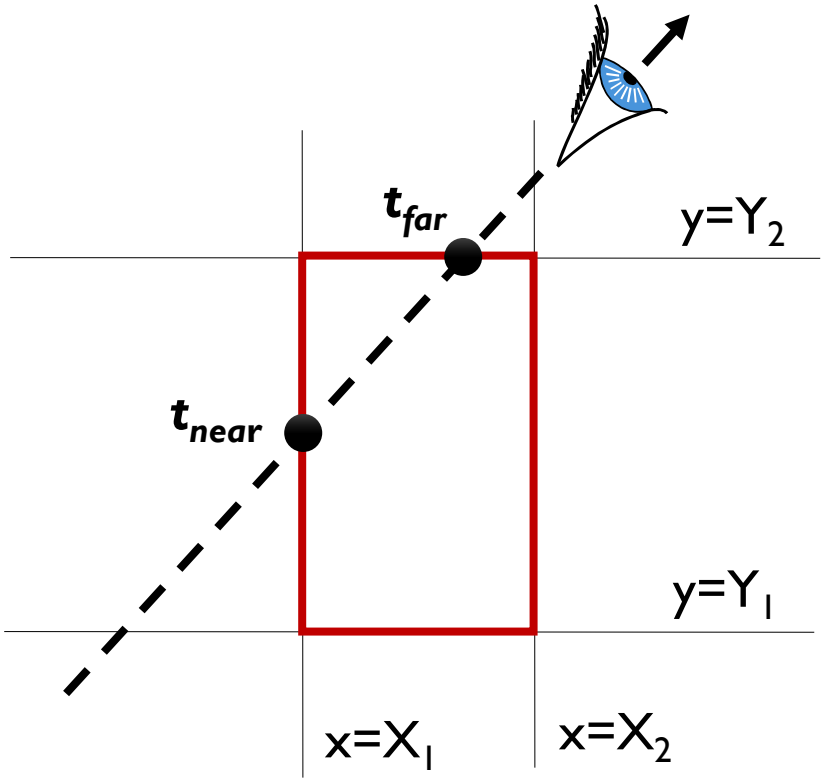
- $t_{near} = \max(t_{1x}, t_{1y})$
- $t_{far} = \min(t_{2x}, t_{2y})$

Boîte englobante alignée sur les axes



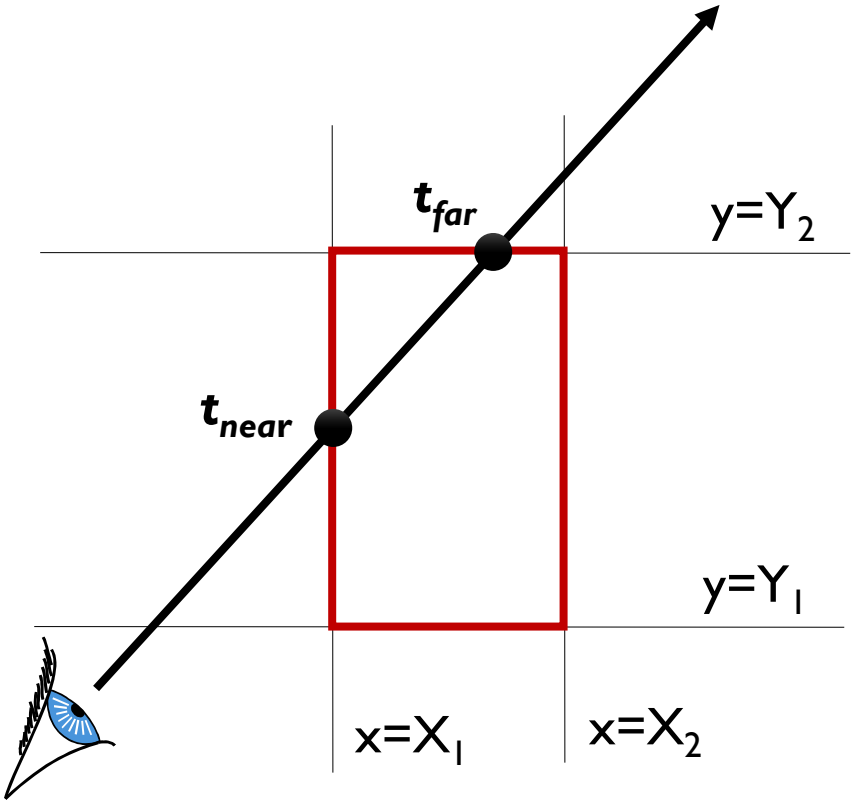
Si $t_{near} > t_{far}$, pas d'intersection

Boîte englobante alignée sur les axes



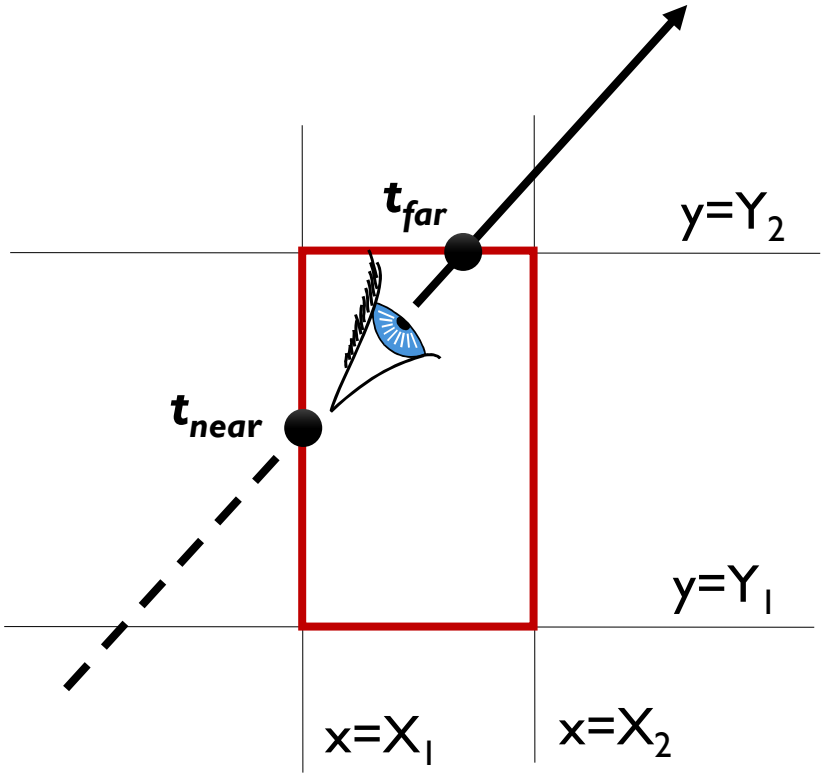
Si $t_{far} < t_{min}$, la boîte est derrière

Boîte englobante alignée sur les axes



Si $t_{near} > t_{min}$, l'intersection se fait à la distance t_{near}

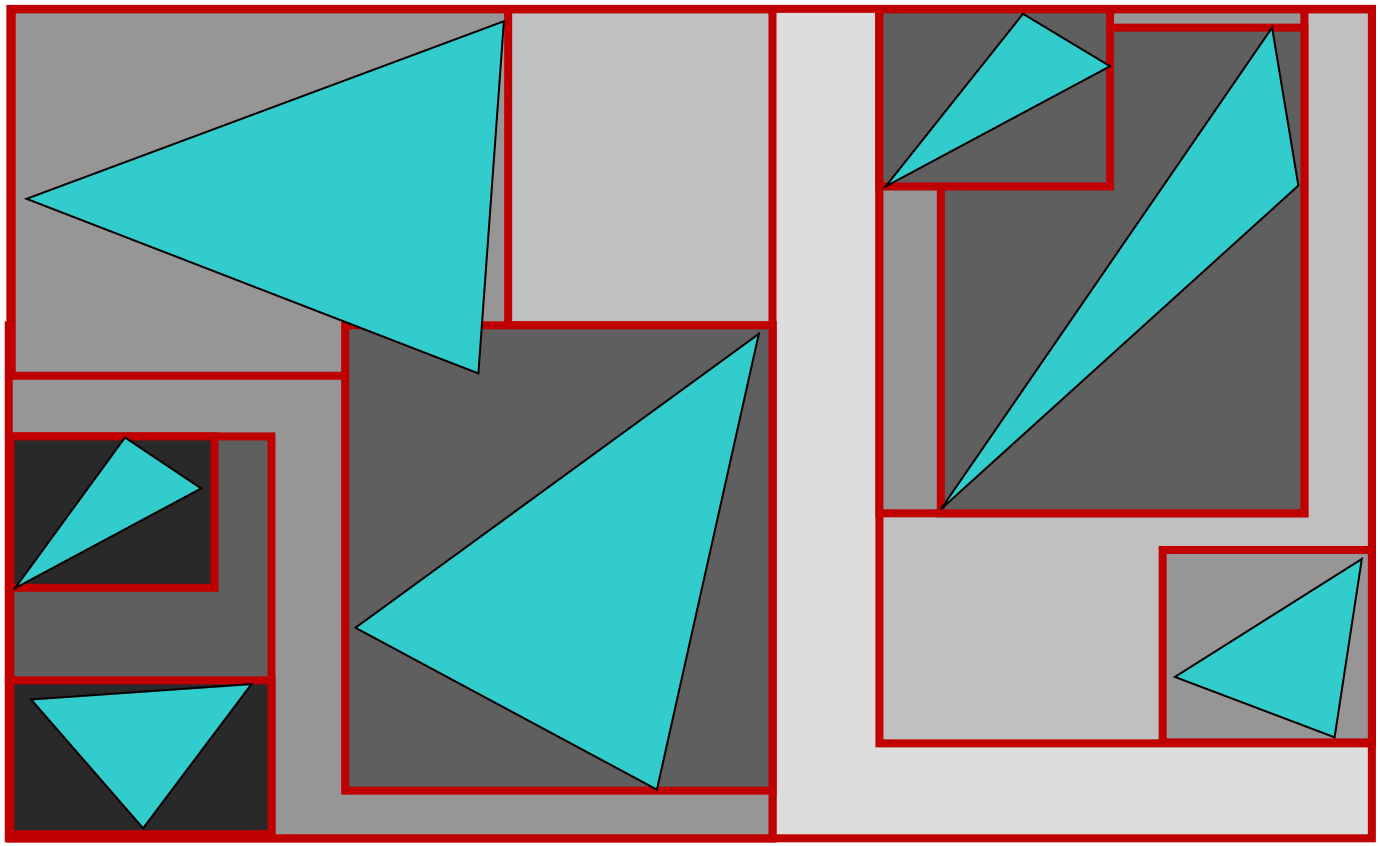
Boîte englobante alignée sur les axes



Sinon, elle se fait à t_{far}

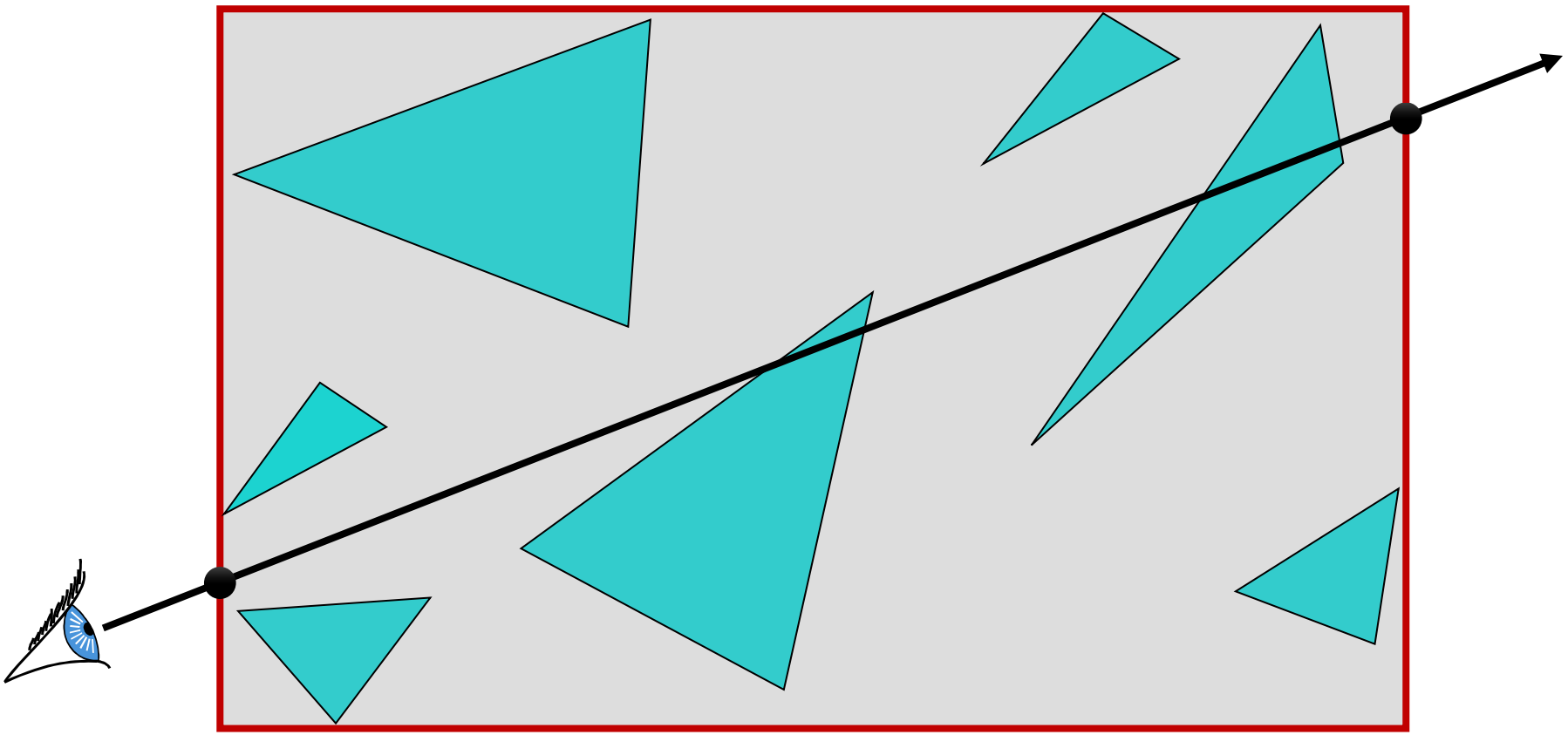
Hiérarchie de boites englobantes

BVH = KD-tree avec 1 boîte par nœud



BVH : intersection

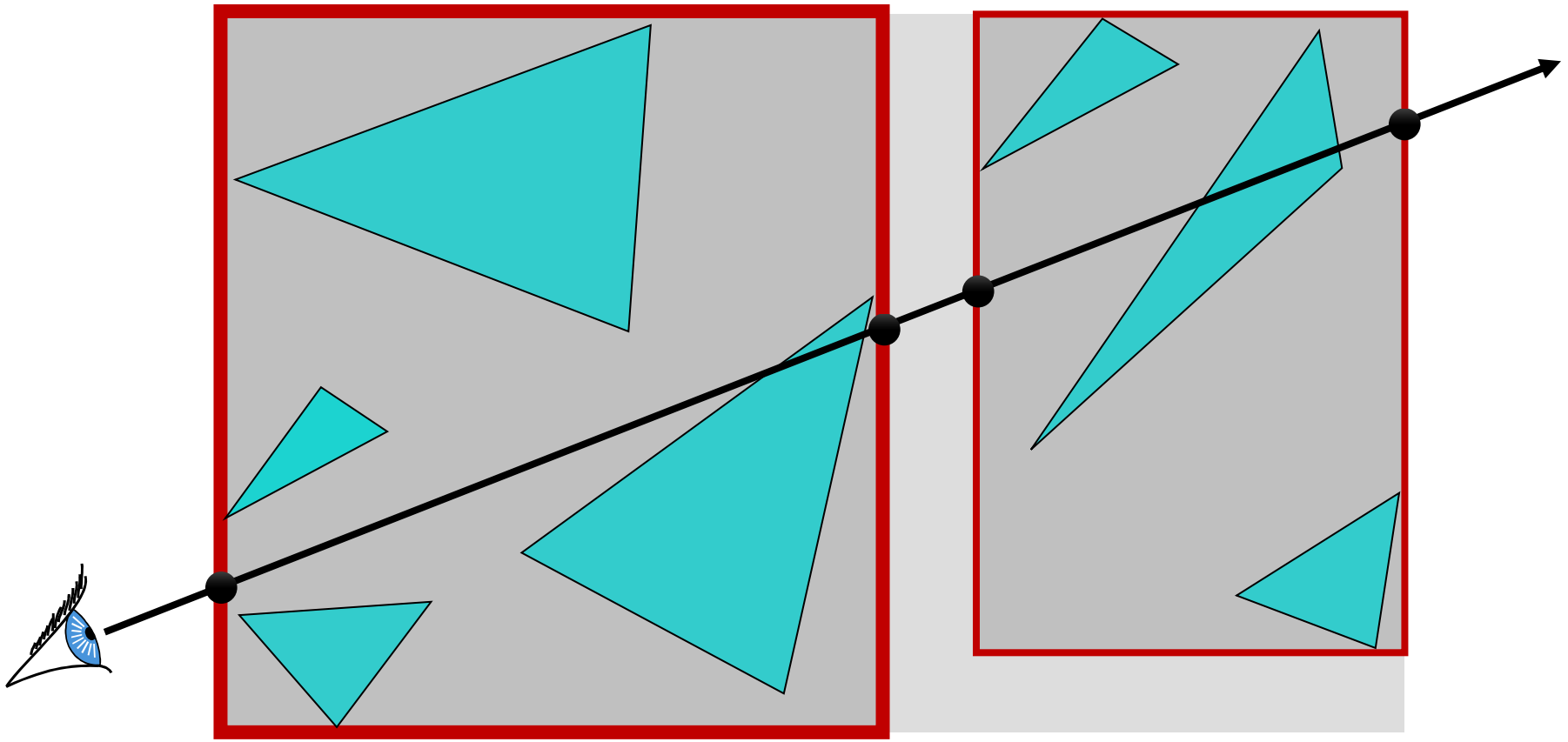
Tester la boîte parente



Hiérarchie de boites englobantes

Si Intersection, descendre sur les fils

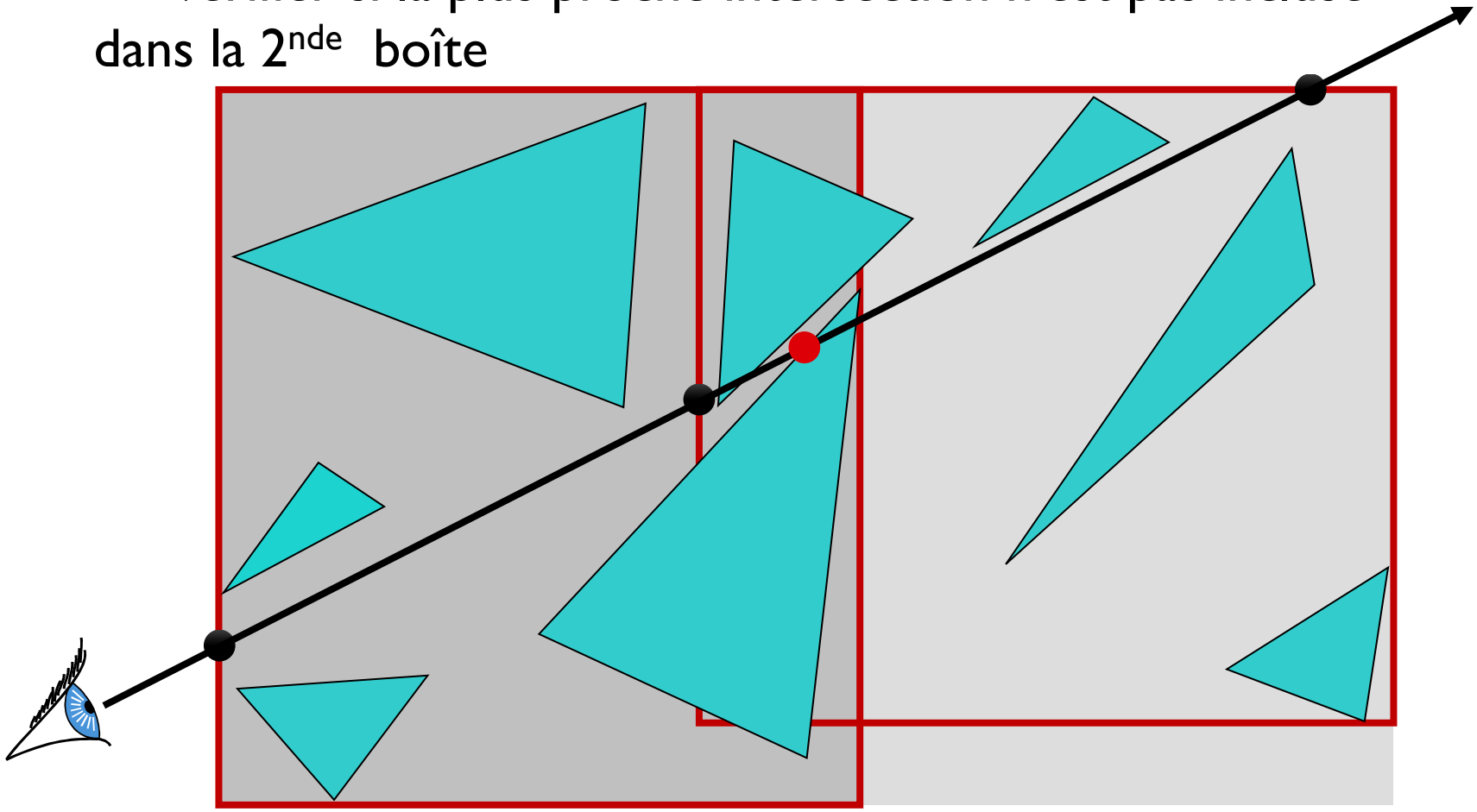
Tester la boîte avec l'intersection la plus proche



Hiérarchie de boites englobantes

Attention, les boîtes peuvent se superposer !

$\Longrightarrow$ vérifier si la plus proche intersection n'est pas incluse dans la 2nde boîte



Question – 3 mn

Comparer KD-tree et BVH

Comparaison

KD-tree

- léger en mémoire (si bien codé)
- parcours simple et rapide
- construction optimale plus facile

BVH

- arbre moins profond
- permet de décaler légèrement un objet sans avoir à reconstruire entièrement l'arbre

Conclusion

- Scènes statiques $\Longrightarrow$ KD-tree
- Scènes dynamiques $\Longrightarrow$ BVH
  (les préférences semblent converger vers le tout BVH)