M2-Images

Accélérations

J.C. Iehl

November 12, 2009

Bilan

pour une image $L \times H$:

calculer l'intersection de chaque rayon avec chaque objet.

pour n objets, $L \times H \times n$ calculs d'intersections...
c'est un peu long !

Comment accélérer ?

réduire L, H ou n :

- facile de calculer une image plus petite,
- mais comment réduire le nombre d'objets ?

changer d'algorithmes ?

- réduire le temps d'intersection ?
- réduire le nombre d'intersections calculées ?

réduire le nombre d'intersections calculées ?

Structures accélératrices

comment ?

trouver l'intersection la plus proche revient à chercher le minimum d'un ensemble d'intersections.

idées :

- ne calculer que ce qui est nécessaire,
- trouver efficacement le minimum des résultats.

Structures accélératrices

organiser :

- mêmes idées que pour la recherche dichotomique,
- trier les données pour faire une recherche efficace.

exploiter l'organisation :

- pour ne calculer que ce qui est necessaire,
- pour traiter les résultats du min au max (et donc trouver facilement le min)

Intuition

dichotomie en 1D :

- rechercher une valeur dans un intervalle $[x_{min}\ x_{max}]$,
- on connait : l'intervalle de l'ensemble (l'englobant),
- l'ordre des intervalles dans l'ensemble (du min au max).

extension à la 3D :

- rechercher un point dans un cube aligné sur les axes $[x_{min}\ x_{max}] \times [y_{min}\ y_{max}] \times [z_{min}\ z_{max}]$,
- on connait : le cube de l'ensemble (l'englobant),
- quelle notion d'ordre en 3D ?

Intuition

ordre en 3D :

- ?
- utiliser la direction du rayon pour ordonner les cubes,
- parcourir les cubes en s'éloignant de l'origine du rayon,
- cf. *trouver le min efficacement*.

Structures classiques

des arbres :

- octree : récursion sur le volume englobant de la scène, les 3 axes simultanément,
- kd-tree : récursion sur le volume englobant de la scène, un axe à la fois,
- BVH : récursion sur les objets, construit a posteriori les volumes englobants, les 3 axes simultanément,
- BIH : récursion sur les objets, construit a posteriori les intervalles englobants, un axe à la fois

des grilles, des grilles à plusieurs niveaux, des tables de hachage, etc.

Structures classiques

arbre :

- les noeuds internes sont associés à un volume englobant (cube aligné, en général),
- les feuilles référencent les objets de la scène.

remarque :

un noeud englobe ses fils : le volume englobant d'un noeud est l'union des englobants de ses fils.

Parcours

rayon : origine + direction

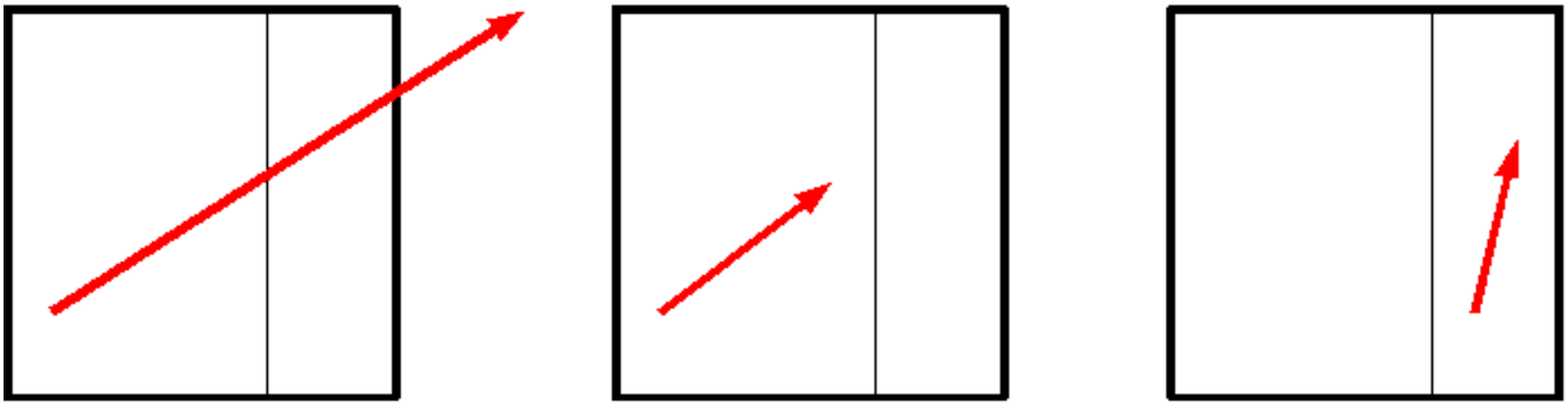
- déterminer si un fils peut intersecter le rayon,
- respecter l'ordre : s'éloigner de l'origine.

exploiter les propriétés de la structure :

- structures récursives sur le volume :
  les fils ne s'intersectent pas,
  la première intersection trouvée est la bonne !
- structures récursives sur les objets :
  les fils peuvent s'intersecter, vérifications supplémentaires.

identifier l'ensemble des fils intersectés par le rayon.

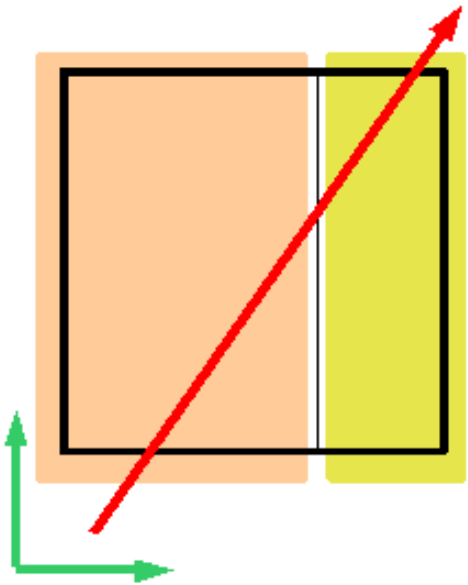
Parcours (volume)

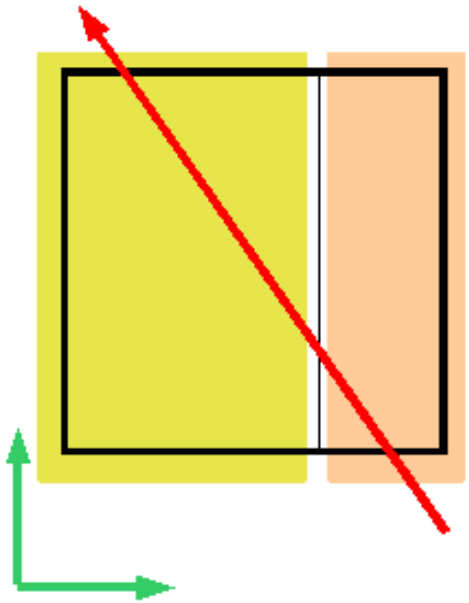


déterminer quels fils parcourir :

- un seul,
- les deux . . .
- ou aucuns.

Parcours (volume)

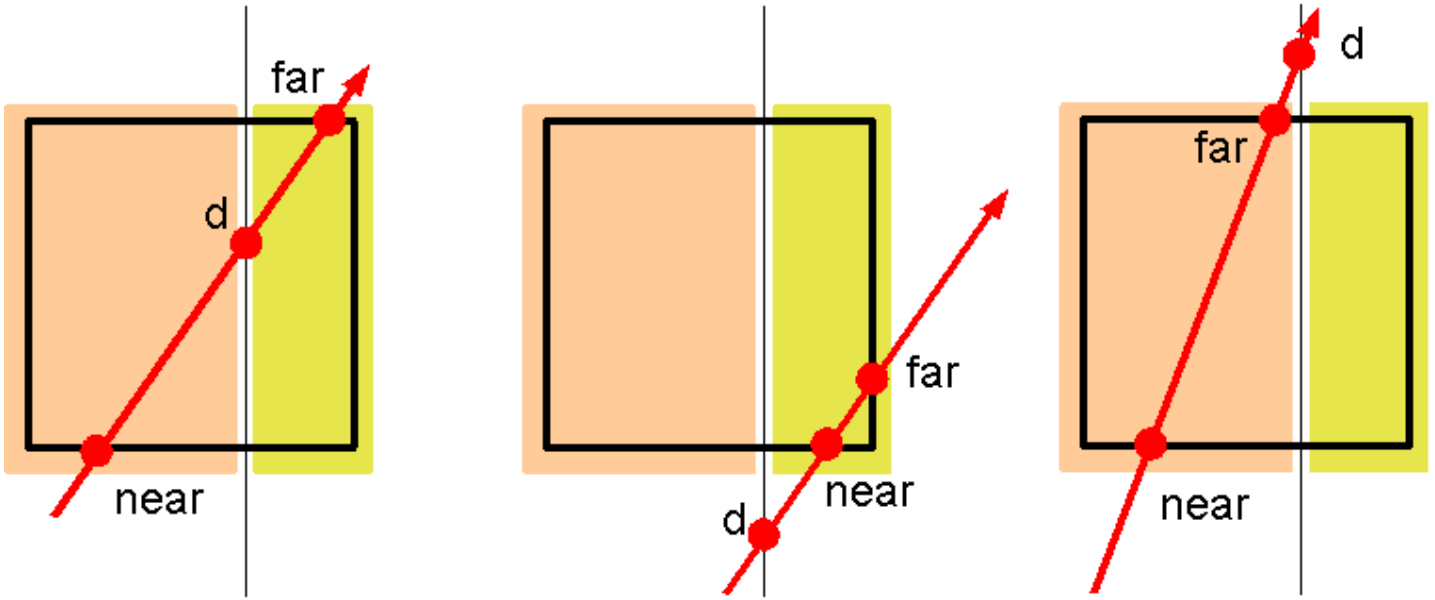




dans quel ordre ?

le signe de la direction du rayon suffit . . .

Parcours (volume)

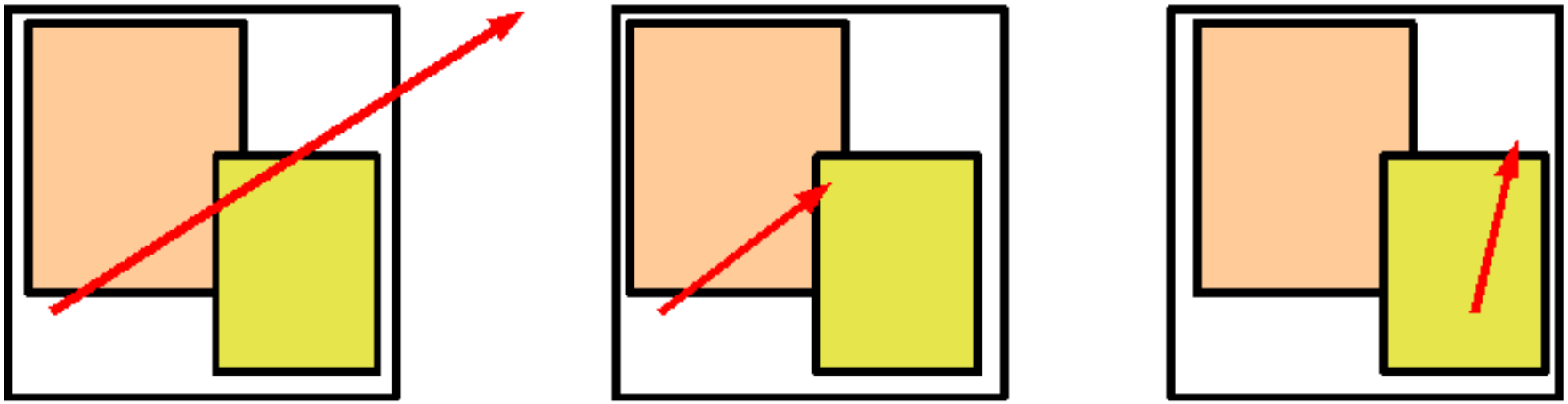


quels fils ?

dépend de la position de d par rapport à far et near :

- d $>$ far, visiter fils gauche,
- d $<$ near, visiter fils droit,
- near $\leq$ d $\leq$ far, visiter fils gauche et droit.

Parcours (objets)



déterminer quels fils parcourir :

calculer l'intersection du cube englobant et du rayon pour les deux fils : $[near\ far]_{gauche}$, $[near\ far]_{droit}$

Parcours (objets)

dans quel ordre :

ordre sur les intervalles $[near\ far]_{gauche}$, $[near\ far]_{droit}$

précalculer :

- les fils ne changent pas de place pendant le calcul d'une image,
- inutile de calculer plusieurs fois leur disposition (cf. algo de construction), déterminer une relation avec la direction du rayon.

Construction

découper l'espace en cubes disjoints :

partition sur les volumes, octree, kd-tree,

ou, grouper les objets et déterminer l'espace occupé :

partition sur les objets, BVH, BIH.

comment grouper les objets ou découper l'espace ?

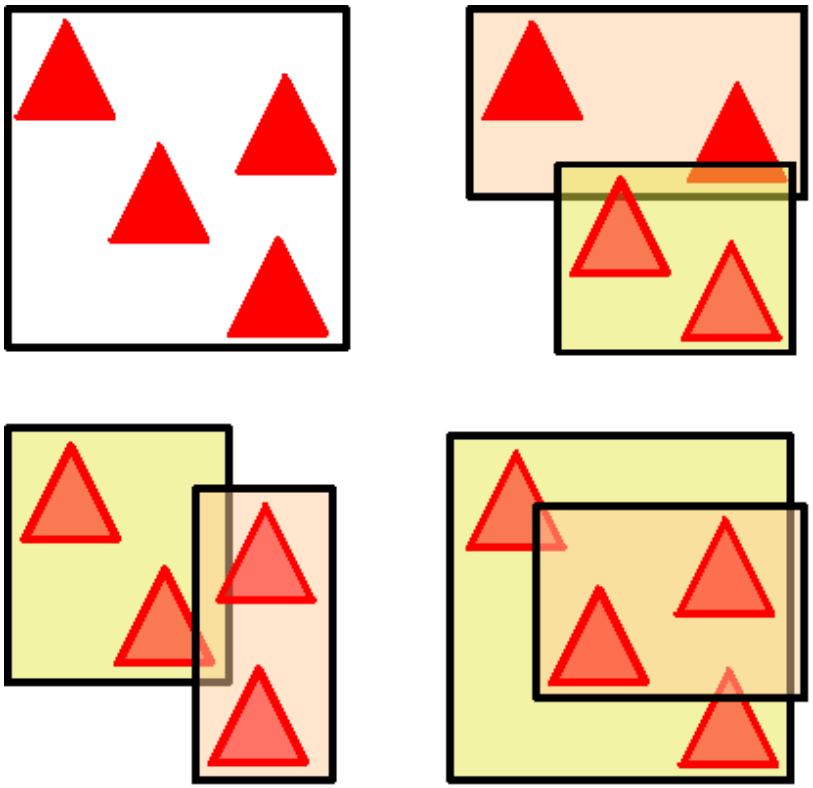
Construction (objets)

solution classique :

- construction récursive d'un arbre équilibré :
- trier les objets sur un axe,
- affecter la première moitiée des objets au fils gauche,
- affecter le reste au fils droit,
- recommencer.

c'est la pire solution ! (ne tient pas compte des rayons)

Construction (objets)



comment répartir les objets en les séparant le plus possible ?
comment limiter le nombre d'intersections calculées ?

Construction guidée par une heuristique

comment évaluer la qualité d'un arbre ?

minimiser le nombre d'intersections calculées pour chaque rayon.

comment ?

probabilité d'intersection d'un fils sachant que le père est lui-même intersecté (par un rayon) :

$$P(fils|noeud) \propto \frac{A(fils)}{A(noeud)}$$

Construction guidée par une heuristique

et pour tout l'arbre ?

$$T = \sum_{n \in noeuds} \frac{A(n)}{A(racine)} \cdot 2\, T_{englobant} + \sum_{f \in feuilles} \frac{A(f)}{A(racine)} \cdot N_{objet}\, T_{objet}$$

trouver l'arbre qui minimise T !
trop de solutions, utiliser une heuristique simple (SAH).

BVH : les détails

cas simple : 2 fils, volumes englobants : cubes alignés sur les axes.

construction :

en fonction du volume occupé par les fils, et du temps de calcul de l'intersection du rayon avec les objets associés aux fils.

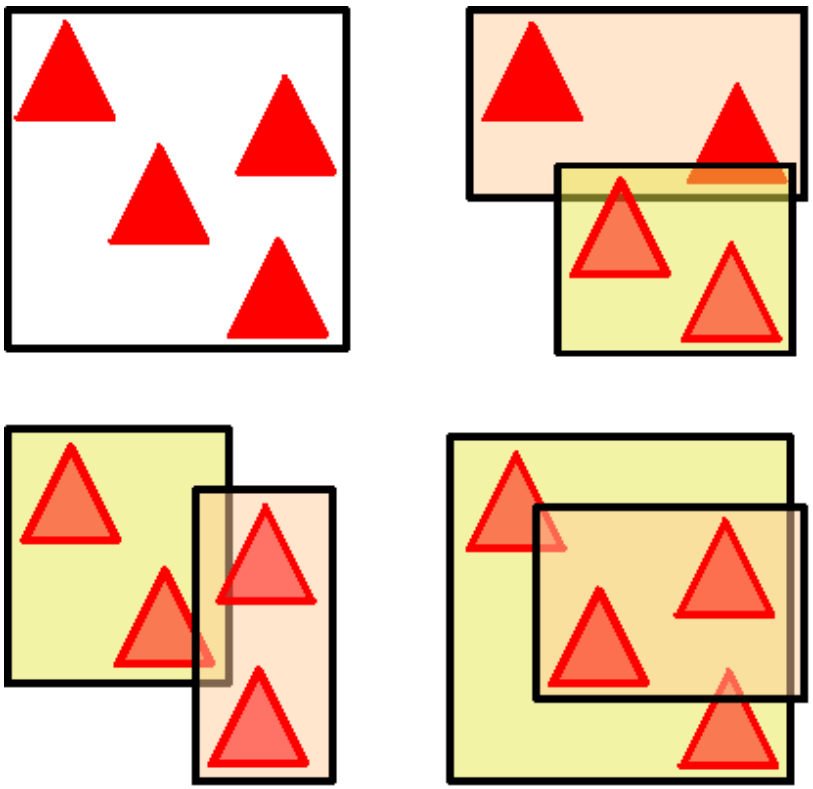
parcours ordonné :

pas obligatoire, mais beaucoup plus efficace.

Construction

trouver la meilleure répartition des objets pour chaque noeud :

choisir un plan candidat et répartir les objets.



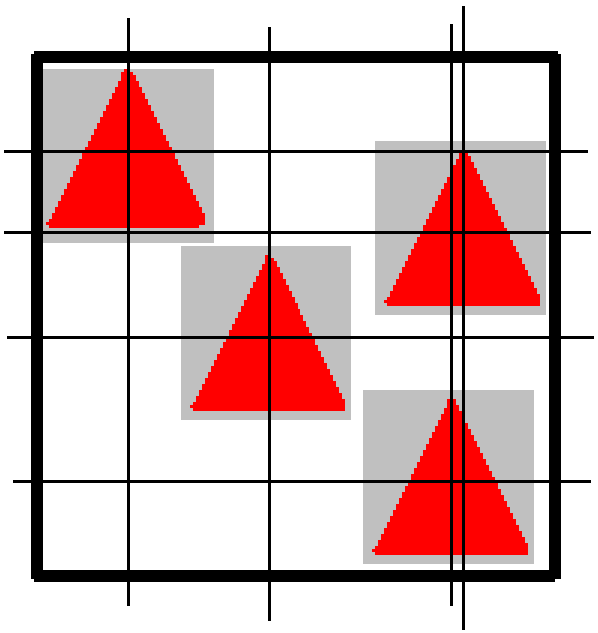
Construction SAH d'un BVH

minimiser T : trouver le meilleur candidat

$$
\begin{aligned}
T &= T_{fils\_gauche} + T_{fils\_droit} \\
T_{fils} &= T_{cube} + \frac{A(fils)}{A(noeud)} N(fils) \times T_{objet}
\end{aligned}
$$

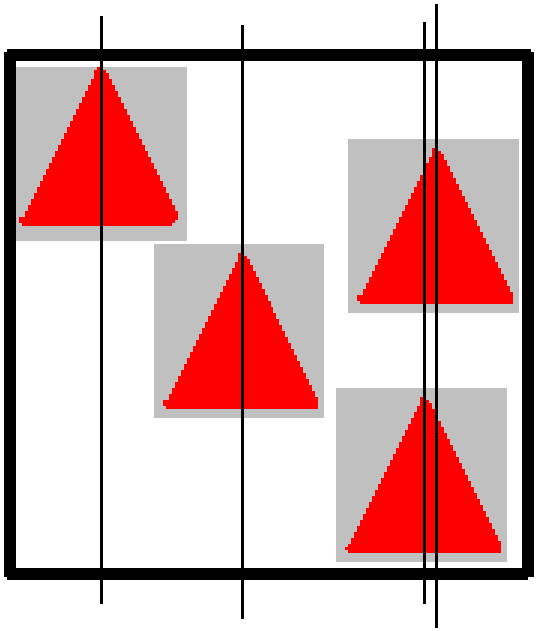
Construction : algorithme

- travaille sur les cubes englobants de chaque objet,
- les candidats sont les 3 plans passants par le centre des cubes englobants.



Construction : algorithme

- travaille sur un axe à la fois,
- teste tous les centres,
- évalue T à chaque fois et garde le meilleur (sur les 3 axes).



Construction : algorithme

- construction récursive :
- critère d'arret ? lorsqu'il n'est plus interessant de continuer.

Evaluer T : algorithme

$$T_{fils} = T_{cube} + \frac{A(fils)}{A(noeud)} N(fils) \times T_{objet}$$

déterminer les 2 sous ensembles d'objets + cubes englobants :

- naif : re-trier à chaque fois, cf. sujet TP,
- incrémental : trier une seule fois par axe, puis exploiter l'ordre pour contruire les cubes englobants,
- il est facile de calculer le min et le max d'un ensemble lorsqu'on insère un élement,
- mais pas le contraire (lorsqu'un supprime un élément) ?

BVH : les détails

construction, parcours et mise à jour :
Ray Tracing Deformable Scenes using Dynamic Bounding Volume Hierarchies

construction rapide :
On fast Construction of SAH based Bounding Volume Hierarchies

Evaluer T : algorithme

- parcourir de min vers max et construire la partie gauche et son cube englobant :
- $N(fils_{gauche})$, $A(fils_{gauche})$
- parcourir de max vers min pour la partie droite :
- $N(fils_{droit})$, $A(fils_{droit})$
- tous les termes de T sont connus pour tous les candidats,
- finir l'évaluation de T pour chaque candidat,
- garder le meilleur, répartir les objets en 2 sous-ensembles,
- recommencer

encore plus vite ?

quel autre type d'arbre permet de gagner facilement du temps lors du parcours ?

idée :

- quelle est la hauteur d'un arbre binaire ?
- quelle est la hauteur d'un arbre dont les noeuds internes ont k fils ?

Shallow Bounding Volume Hierarchies for Fast SIMD Ray Tracing of Incoherent Rays

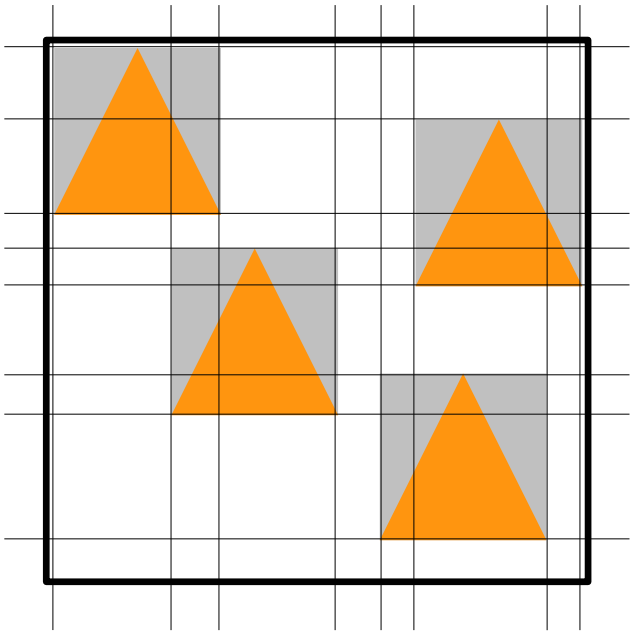
KD-Tree : les détails

même idée que pour les BVH :

- identifier les plans candidats,
- compter le nombre d'objets à gauche, à droite, sur le plan,
- évaluer l'heuristique (le coût),
- recommencer pour le fils gauche et pour le fils droit.

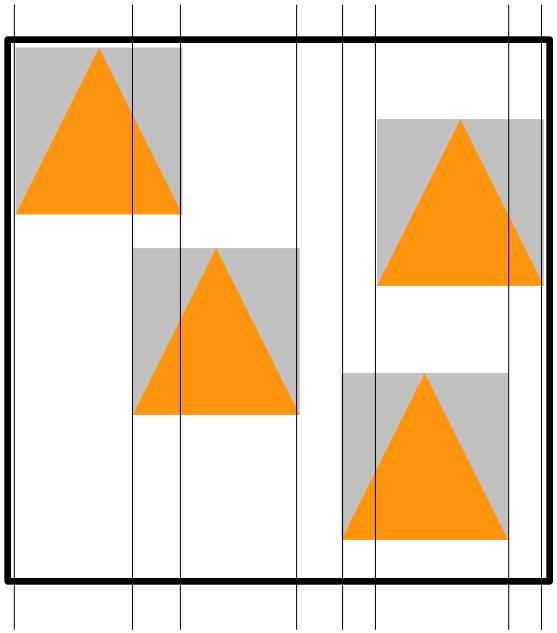
KD-Tree : les détails

- travaille sur les cubes englobants de chaque objet,
- les candidats sont les faces des cubes englobants.



KD-Tree : les détails

- travaille sur un axe à la fois, plan par plan,
- compter le nombre d'objets à gauche, à droite, et sur le plan candidat,
- évalue T et garde le meilleur (sur les 3 axes).



KD-Tree : les détails

construction :

On building fast kd-Trees for Ray Tracing, and on doing that in O(N log N)

Parcours cohérent

parcourir l'arbre encore plus vite ?

- ?
- que se passe-t-il pour des rayons "proches" ?

"paquet" de rayons :

les rayons associés à des pixels proches dans l'image.

Parcours cohérent

sur le haut de l'arbre :
les rayons "proches" parcourent les mêmes noeuds,

sur le bas de l'arbre :
les rayons se répartissent dans les différentes feuilles (perte de cohérence)

comment exploiter cette cohérence ?

Parcours cohérent

idée :

ne pas tester tous les rayons,

utiliser un rayon "représentatif" et supposer que les autres rayons se comportent de la même manière.

Parcours cohérent

algorithme :

- déterminer le rayon "représentatif",
- si aucun rayon n'intersecte le noeud, arrêter,
- déterminer l'ordre des fils en fonction du rayon "représentatif",
- parcourir le fils "proche",
- parcourir le fils "loin".

Rayon "représentatif"

l'arbre est une hiérarchie d'englobants :

si un rayon n'intersecte pas l'englobant du noeud actuel, il ne peut pas intersecter les fils du noeud.

trouver le premier rayon du "paquet" qui intersecte l'englobant du noeud actuel.

Parcours cohérent

facile pour les BVH :

Ray Tracing Deformable Scenes using Dynamic Bounding Volume Hierarchies

pénible sur les KD-Tree :

...