M2-Images

Intersections

J.C. Iehl

January 9, 2014

Résumé des épisodes précédents

résumé :

- déterminer la visibilité de 2 points,
- permet de calculer les transferts d'énergie,
- ombre, pénombre, reflet, transparence, ...

trop d'intersections calculées ...

c'est un peu long ? non ?

calculer les intersections :

- pour chaque rayon :
- calculer l'intersection avec les objets,
- ne garder que la plus proche de l'origine de chaque rayon...

- beaucoup de tests d'intersection inutiles...

plus vite ?

approche classique :

- si le probleme est trop "gros" / complexe,
- le découper en 2,
- jusqu'à ce que la solution soit directe,
- et regrouper les résultats intermédiaires.

Application au lancer de rayons

N rayons $\times T$ triangles :

- si $N \times T > k$ découper...
- sinon calculer toutes les intersections.

comment découper ?

- en 2 moitiés égales (cf complexité algo) :
- facile de partager l'ensemble d'objets en 2,
- facile de partager l'ensemble de rayons en 2,

peu de chances de découper $N \times T$ en 2...

Application au lancer de rayons

partager les objets :

- $+$ trouver les rayons qui peuvent intercepter ces objets,

ou partager les rayons :

- $+$ trouver les objets qui peuvent intercepter ces rayons,

peu de chances de découper $N \times T$ en 2...
et en temps lineaire...

Modèle de coût

idée :

- si on pouvait estimer a priori le nombre de rayons passant dans une région de la scène ?
- (l'englobant d'un ensemble d'objets, par exemple...)
- il suffirait de trouver quel sous ensemble de triangles permet de couper $N \times T$ en 2 !!

Modèle de coût

idée algo :

- il suffit de tester plusieurs découpages :
- pour chaque découpage αT :
- 1. construire un englobant des αT triangles,
- 2. estimer les βN rayons interceptant a priori l'englobant,
- 3. calculer $\alpha T \times \beta N$
- garder le meilleur découpage, le plus proche de $\frac{N \times T}{2}$

Modèle de coût

géométrie et probabilités :

- probabilité qu'un rayon passe dans une région convexe A :
- (sachant qu'il passe deja dans l'englobant B des T triangles)
- $\beta = \frac{Aire(A)}{Aire(B)}$
- pour N rayons, il y en a βN qui passent par $A \in B$...
- si on construit A avec αT triangles...

"Some Integral Geometry Tools to Estimate the Complexity of 3D Scenes"
F. Cazals, M. Sbert, 1997

Modèle de coût

ca devrait marcher...

- à vérifier en pratique !

à lire sur le sujet :

- "Efficient Divide-And-Conquer Ray Tracing using Ray Sampling"
  K. Nabata, K. Iwasaki, Y. Dobashi, T. Nishita, 2013
- "Naive Ray-Tracing: A Divide-And-Conquer Approach"
  B. Mora, 2011

Modèle de coût

ca marche, mais :

- "trier" les objets pour chaque ensemble de rayons :
- rayons primaires (depuis la camera),
- rayons d'ombres (vers les sources),
- rayons indirects (vers les autres objets)
- coûteux de "retrier" les objets à chaque fois...

Modèle de coût

utiliser les mêmes idées pour construire un arbre :

- stocker l'arbre construit implicitement par l'algo d'intersection,
- chaque noeud de l'arbre correspond à une récursion de l'algo d'intersection,
- parcourir l'arbre pour trouver les triangles pouvant a priori intercepter le rayon.

Hiérarchie (de groupes) d'objets

comment construire la hiérarchie ?

- s'inspirer des arbres binaires de recherche équilibrés ?
- d'une recherche dichotomique ?
- (utiliser le modèle de coût ?)

arbre "binaire" en dimension 3 ? combien de fils ?

Hiérarchie (de groupes) d'objets

arbres :

- arbre en dimension 3 : octree, "arbre octal", 8 fils,
- rester en dimension 1 ? découper un axe à la fois, récursivement ?

combien de "découpages binaires" pour représenter un octree ?

Hiérarchie (de groupes) d'objets

algorithme de construction :

- problème (et solution) classique :
- insérer les objets dans l'ordre mais l'arbre ne sera pas équilibré,
- ou "trier" les objets pour équilibrer l'arbre.

Hiérarchie (de groupes) d'objets

algorithme de construction équilibrée classique :

- construction de la racine vers les feuilles :
- trier les objets,
- affecter la première moitiée au fils gauche,
- le reste au fils droit,
- recommencer sur chaque sous ensemble.

trier des objets 3d ?

- trier sur X, partitionner sur X == 2 sous ensembles,
- puis, trier sur Y, partitionner sur Y == 4 sous ensembles,
- puis, trier sur Z, partitionner sur Z == 8 sous ensembles.

Hiérarchie (de groupes) d'objets

construire les noeuds :

- un sous ensemble d'objets est associé à chaque noeud interne :
- déterminer un englobant de ces objets,
- les noeuds internes ne stockent pas les objets, uniquement l'englobant et les fils,
- les feuilles référencent le sous ensemble d'objets.

Hiérarchie (de groupes) d'objets

calculs d'intersection avec un arbre :

- intersection(noeud, rayon) :
- calculer l'intersection du rayon et de l'englobant du noeud,
- si pas d'intersection avec l'englobant, terminer.
- sinon recommencer pour chaque fils,
- intersection(feuille, rayon) :
- calculer l'intersection de chaque objet avec le rayon,
- renvoyer l'intersection valide la plus proche de l'origine du rayon.

Hiérarchie (de groupes) d'objets

améliorations :

- une seule intersection est nécessaire, la plus proche de l'origine du rayon,
- éviter de tester les noeuds "trop loins" ?
- éviter de tester les noeuds après avoir trouvé la bonne intersection ?

intuition :

- choisir dans quel ordre parcourir les fils ? en s'éloignant de l'origine du rayon,
- ne pas parcourir les noeuds qui ne peuvent plus intercepter le rayon.

Hiérarchies

plusieurs types d'arbres :

- sur les objets, BVH (Bounding Volume Hierarchy), répartition des objets,
- sur les positions des objets, octree, découpage de l'espace, les 3 axes à la fois,
- sur les positions des objets, kD-tree, découpage de l'espace, un axe à la fois.

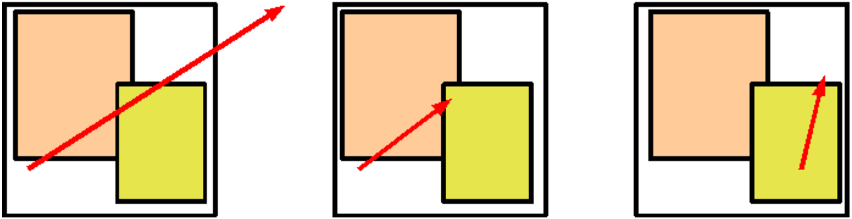
Hiérarchies

propriétés différentes selon le type d'arbre :

- récursion sur le volume :
  les fils ne s'intersectent pas,
  la première intersection trouvée est la bonne !
- récursion sur les (groupes d') objets :
  les fils peuvent s'intersecter, vérifications supplémentaires.

détails de construction et du parcours exploitant ces propriétés.

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)

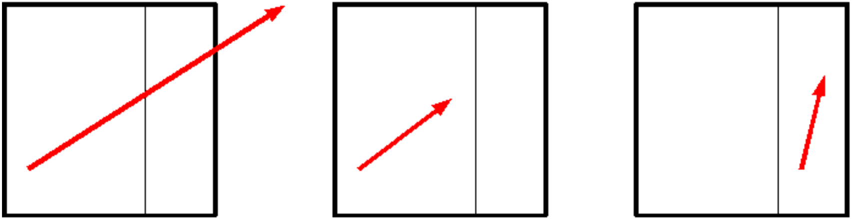
dans quel ordre :

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.

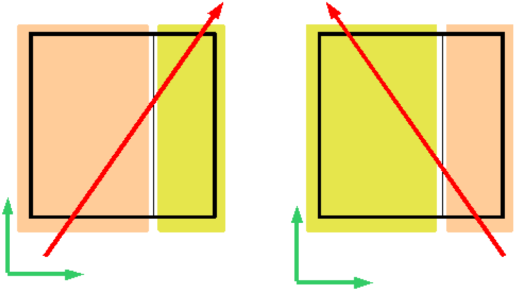
Parcours (volume)



déterminer quels fils parcourir :

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

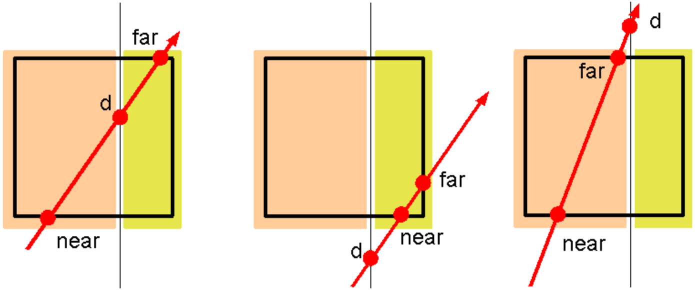
Parcours (volume)



dans quel ordre ?

le signe de la direction du rayon suffit ...

Parcours (volume)

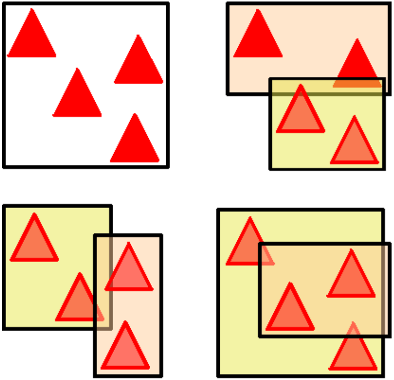


quels fils ?

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

- d > far, visiter fils gauche,
- d < near, visiter fils droit,
- near ≤ d ≤ far, visiter fils gauche et droit.

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) :

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

estimations :

- nombre de noeuds internes intersectés : $\sum P(noeud_i|racine)$,
- nombre de feuilles intersectées : $\sum P(feuille_i|racine)$,
- nombre d'objets intersectés : $\sum P(feuille_i|racine) \times N_i$

Construction guidée par une heuristique

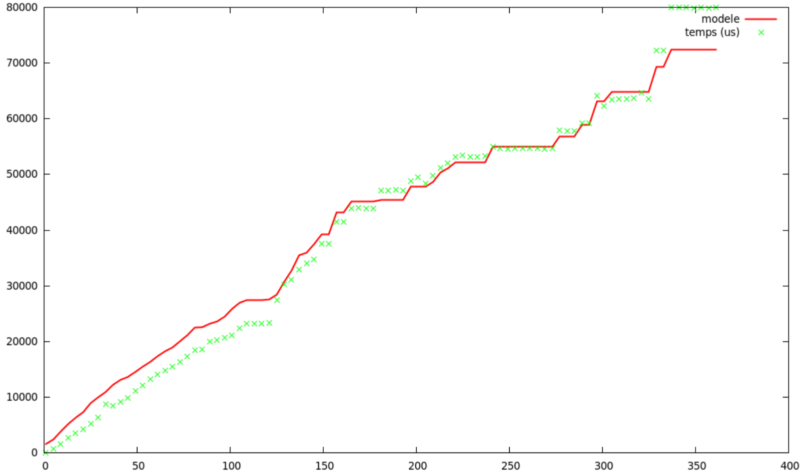
et pour tout l'arbre ?

$$T = \sum P(noeud_i | racine) T_{noeud} + \sum P(feuille_i | racine) \times N_i \times T_{objet}$$

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

"Heuristics for ray tracing using space subdivison"
J. D. MacDonald, K.S. Booth, 1990

et ça marche ?


modele
temps (us)
0
10000
20000
30000
40000
50000
60000
70000
80000
0
50
100
150
200
250
300
350
400

et ça marche ?

pas trop mal !?

- à lire :
- "On Quality Metrics of Bounding Volume Hierarchies"
  T. Aila, T. Karras, S. Laine, HPG 2013

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.

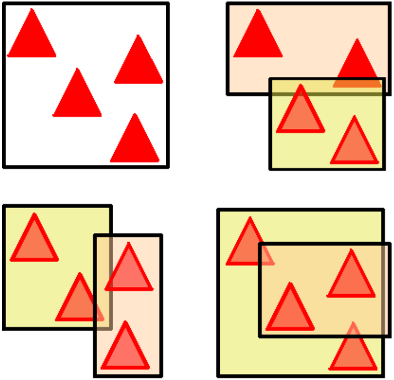
parcours ordonné :

pas obligatoire, mais beaucoup plus efficace.

Construction

trouver la meilleure répartition des objets pour chaque noeud :

choisir un plan candidat et répartir les objets.



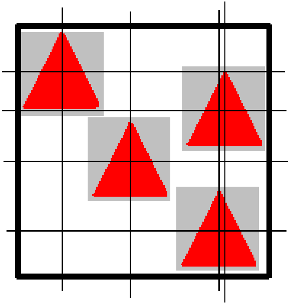
Construction SAH d'un BVH

minimiser T : trouver le meilleur candidat

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

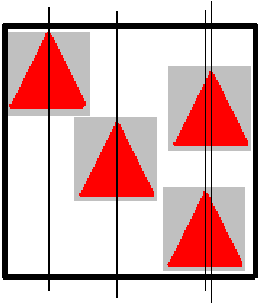
Construction : algorithme

- 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,
- 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"
I. Wald, S. Boulos, P. Shirley, 2007

construction rapide :
"On fast construction of SAH based bounding volume hierarchies"
I. Wald, 2007

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"
H. Dammertz, J. Hanika, A. Keller, 2008

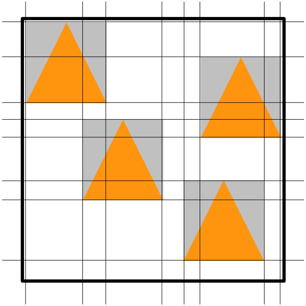
kD-tree : les détails

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.

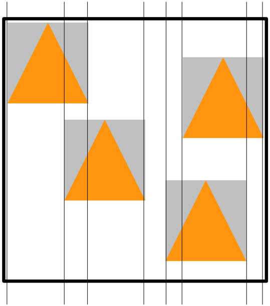
kD-tree : les détails

- 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)"
I. Wald, V. Havran, 2006

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"

I. Wald, S. Boulos, P. Shirley, 2007

pénible sur les kD-tree :

...

Encore plus vite ?

avec une carte graphique :

- "Understanding the efficiency of ray traversal on GPUs"
  T. Aila, S. Laine, 2009
- "Architecture considerations for tracing incoherent rays"
  T. Aila, T. Karras. 2010
- "Fast Ray Sorting and Breadth-First Packet Traversal for GPU Ray Tracing"
  K. Garanzha, C. Loop, 2010
- "Active Thread Compaction on the GPU"
  I. Wald, 2011

Encore plus vite ?

avec une hiérarchie de meilleure qualité :

- "Fast Insertion-Based Optimization of Bounding Volume Hierarchies"
  J. Bittner, M. Hapala, V. Havran, CGF 2013
- "Object Partitioning Considered Harmful: Space Subdivision for BVHs"
  S. Popov, I. Georgiev, R. Dimov, P. Slusallek, 2009
- "Spatial Splits in Bounding Volume Hierarchies"
  M. Stich, H. Friedrich, A. Dietrich, 2009

Encore plus vite ?

avec un parcours plus efficace :

- certaines "opérations" sont inutiles dans le parcours ?
- *erreur* de parcours: le noeud / la feuille testé n'a pas d'intersection...
- fréquent "en bas" de l'arbre, mais peu couteux (feuilles),
- moins fréquent "en haut" (noeuds internes) de l'arbre, mais très couteux ?