Lancer de rayons en temps interactif

Gabriel Fournier

Doctorant au LIRIS

Contexte

- Article d'Ingo Wald présenté à Eurographics 2001: *Interactive rendering with coherent ray tracing.*
  - Lancer de rayons en temps interactif sur PC
- Méthode des vecteurs lumineux : Zaninetti, Serpaggi et Péroche
  - Méthode de Monte-Carlo pour le calcul de l'illumination globale
  - Temps de calcul très long
- Stage de DEA :
  - Appliquer les idées de Wald à la méthode des vecteurs lumineux.

Plan

- Interactive Rendering With Coherent Raytracing
- Comparaison Grille – KD-arbre
- Échantillonnage de sources étendues
- Échantillonnage d'un hémisphère
- Conclusion

Interactive Rendering With Coherent Raytracing

- Optimisations
  - Cohérence : des rayons proches (origines et directions) interceptent les mêmes triangles
    - Rayons primaires lancés par paquet de 4
      - 4 tests d'intersections simultanément avec les instructions SIMD (SSE)
      - Gain de temps au niveau des transferts de données entre CPU et mémoire
  - Gestion de la mémoire
    - Uniquement les données nécessaires aux calculs d'intersection
    - Pré-cachage des données : PREFETCH

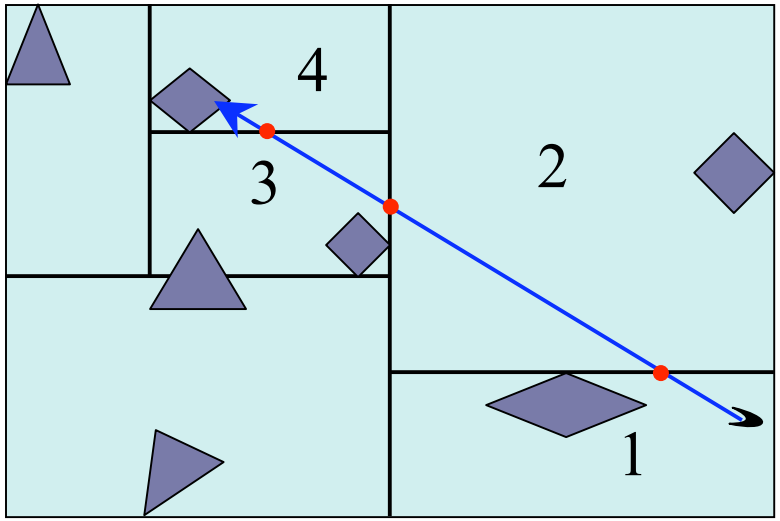
Interactive Rendering With Coherent Raytracing

- Limitations
  - Scène décrite par des triangles
  - Stockage dans un KD-arbre
  - Ombres et reflections limitées
  - Hardware : SSE uniquement à partir de Athlon XP et PIII
- Résultats
  - Tests d'intersections : x 3.5
  - Rayons d'ombres : x 2
  - 7,7 fps dans une scène de 970K triangles sans ombres et sans reflections
  - Lancer de rayons plus rapide que carte graphique à partir d'un certain nombre de triangles

Plan

- Interactive Rendering With Coherent Raytracing
- Comparaison Grille – KD-arbre
- Échantillonnage de sources étendues
- Échantillonnage d'un hémisphère
- Conclusion

Rappel sur le kD-arbre



- Arbre binaire
- Partition de l'espace
- Plans alignés avec les axes
- 3 ou 4 triangles dans chaque feuille
- Des triangles peuvent être dans plusieurs feuilles

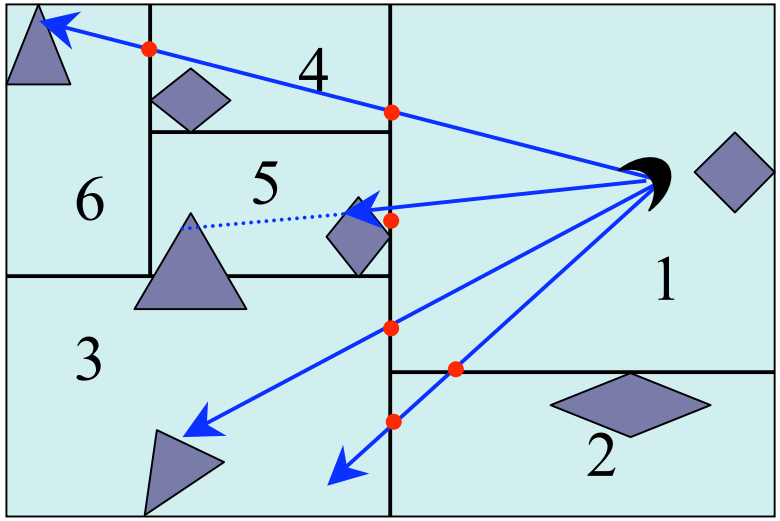
Algorithme (de Havran) de parcours par un rayon :

- utilise une pile
- 1 calcul d'intersection
- 3 comparaisons

simple, rapide

4 rayons dans un kD-arbre

Lorsqu'on envoie 4 rayons dans l'arbre des problèmes se posent :



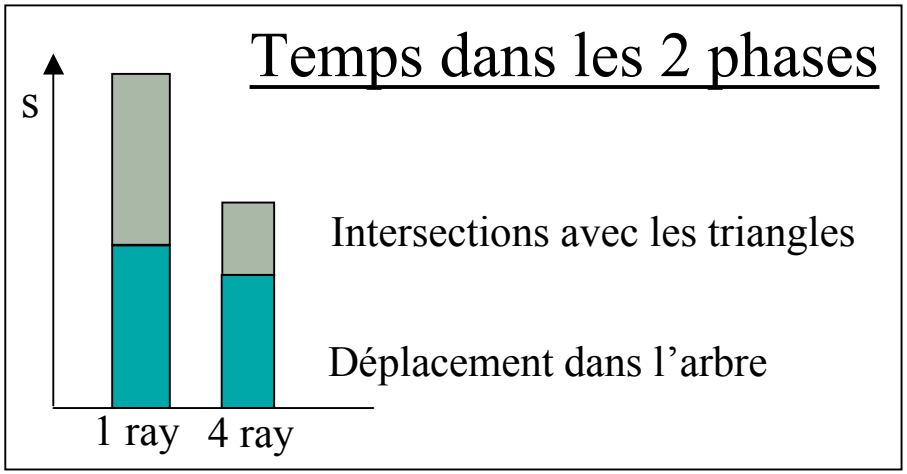
- arrêt +/- rapide
- chemins différents
- triangles dans plusieurs cellules

Algorithme de parcours plus compliqué:

- on masque les calculs lorsqu'un rayon ne passe pas dans un nœud testé
- plusieurs tests pour déterminer l'ordre dans lequel parcourir les nœuds

4 rayons dans un kD-arbre

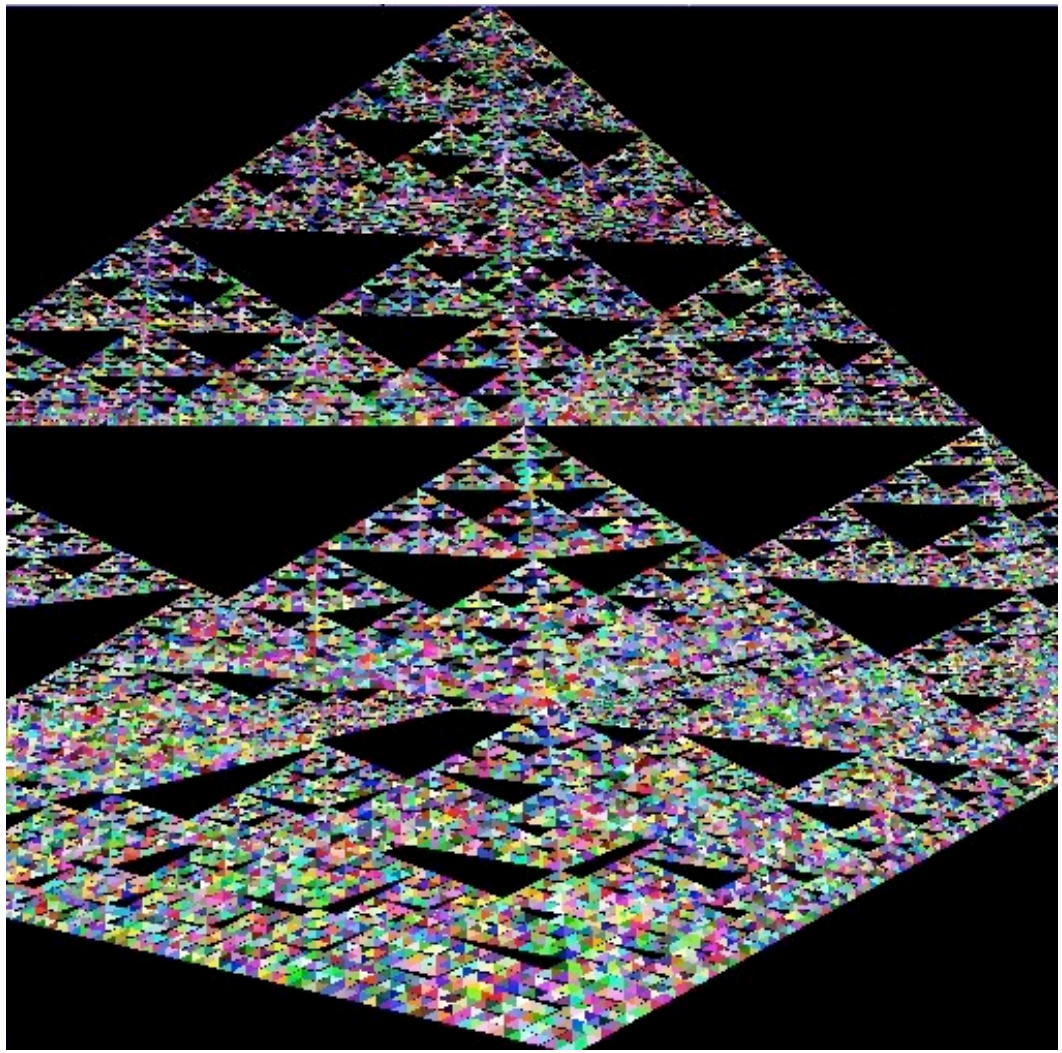
Tests de l'algorithme sur plusieurs scènes dont tétraèdre de Sierpinsky



- Gain moyen : x 1.5
- 4 rayons : très efficace pour l'intersection
- Mais peu efficace pour le déplacement dans l'arbre
- Inefficacité totale du pré-cachage

⇒Peu satisfaisant

Images de test: tétraèdre de Sierpinsky



65 536 triangles
512x512
kD-arbre
4 rayons : 0.8s
1 rayon:1s
Sur Athlon XP 1800+

Grille

- La grille

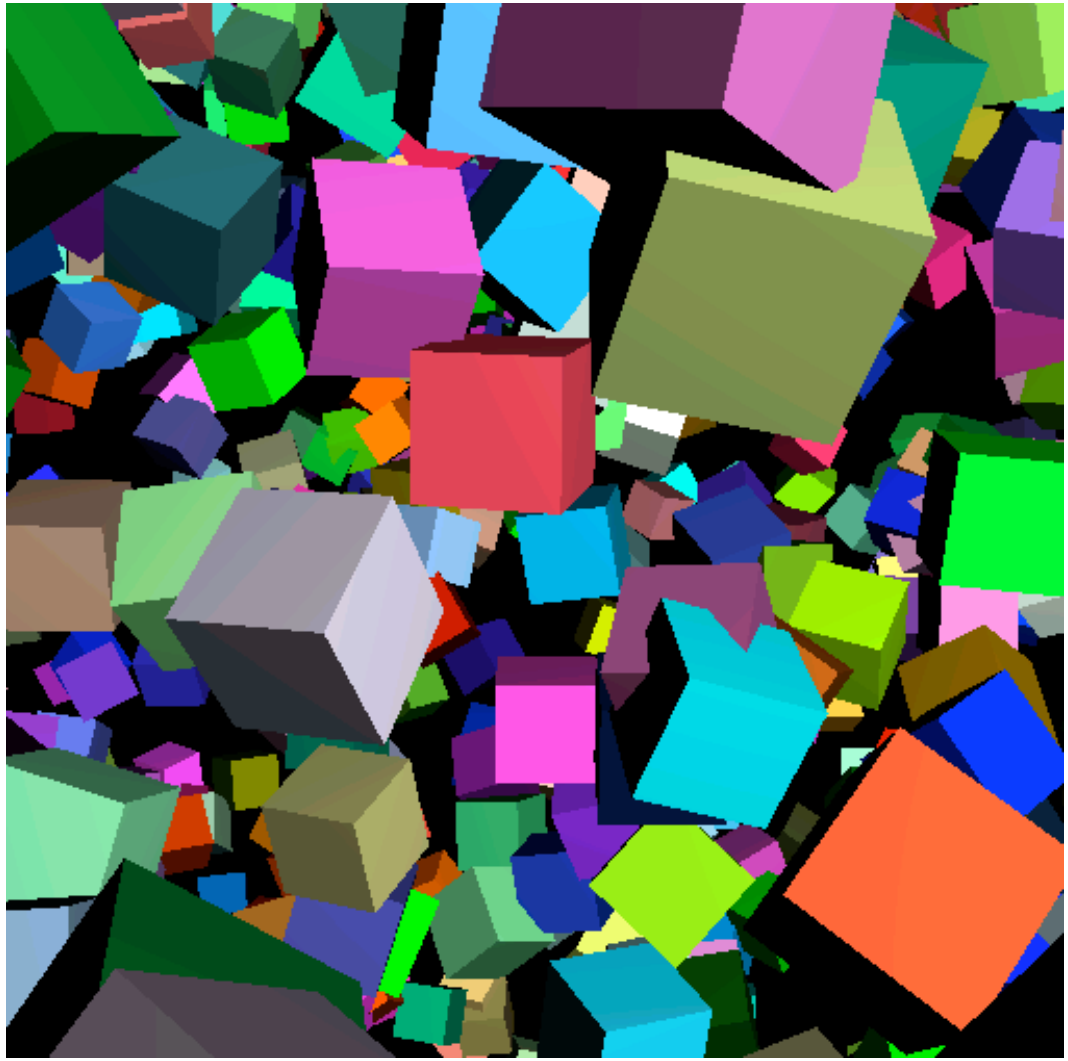
  Développement d'un algo pour traiter 4 rayons (très simple, à partir de l'algo d'Amanatides)

  - On avance sur chaque rayon indépendamment
  - A chaque itération on regarde dans quelles cellules (1 à 4) se trouvent les rayons
  - On teste les 4 rayons contre les triangles de chacune des cellules
  - On ne suit plus les rayons déjà interceptés

  Gain x 1.5 à 2 selon la scène pour le parcours comme le calcul des intersections

  Progression le long des rayons: beaucoup d'arithmétique entière meilleure optimisation avec SSE 2 (sur Pentium 4 uniquement)

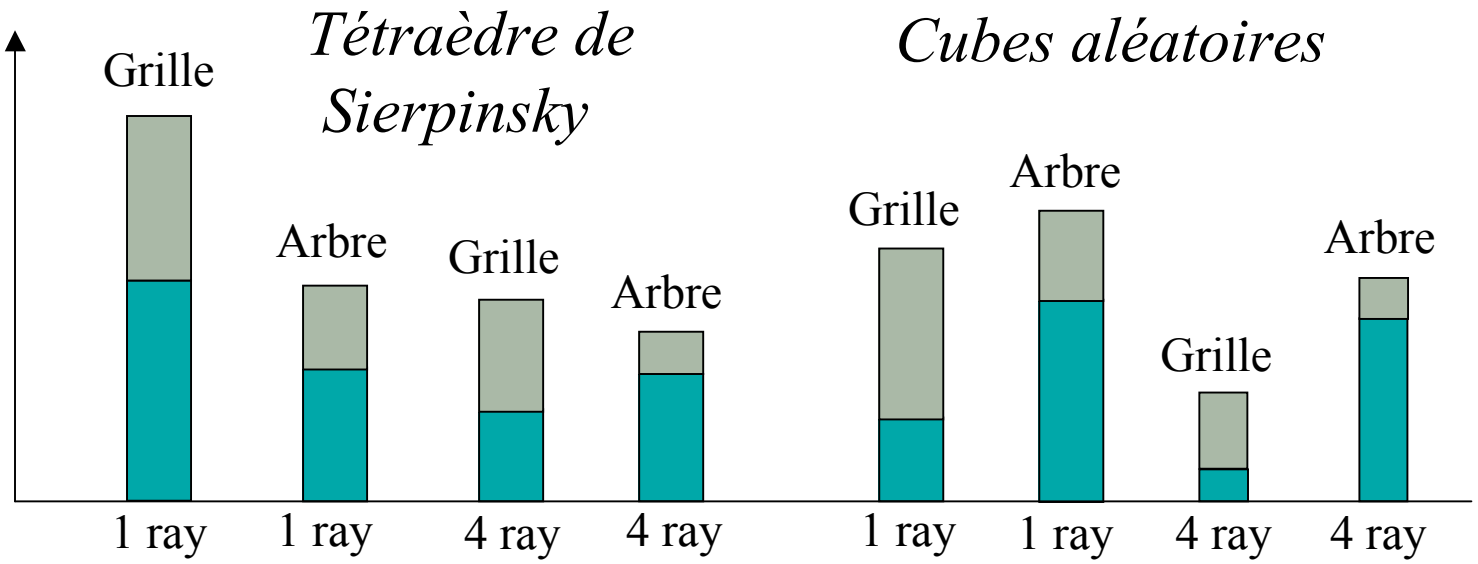
Images de tests: Cubes aléatoires



240 000 triangles
512x512
Grille
4 rayons: 0.6s
1 rayon: 1.3s
Sur Athlon XP 1800+

Grille ou kD-arbre ?

Comparaison grille et kD-arbre dans 2 scènes:



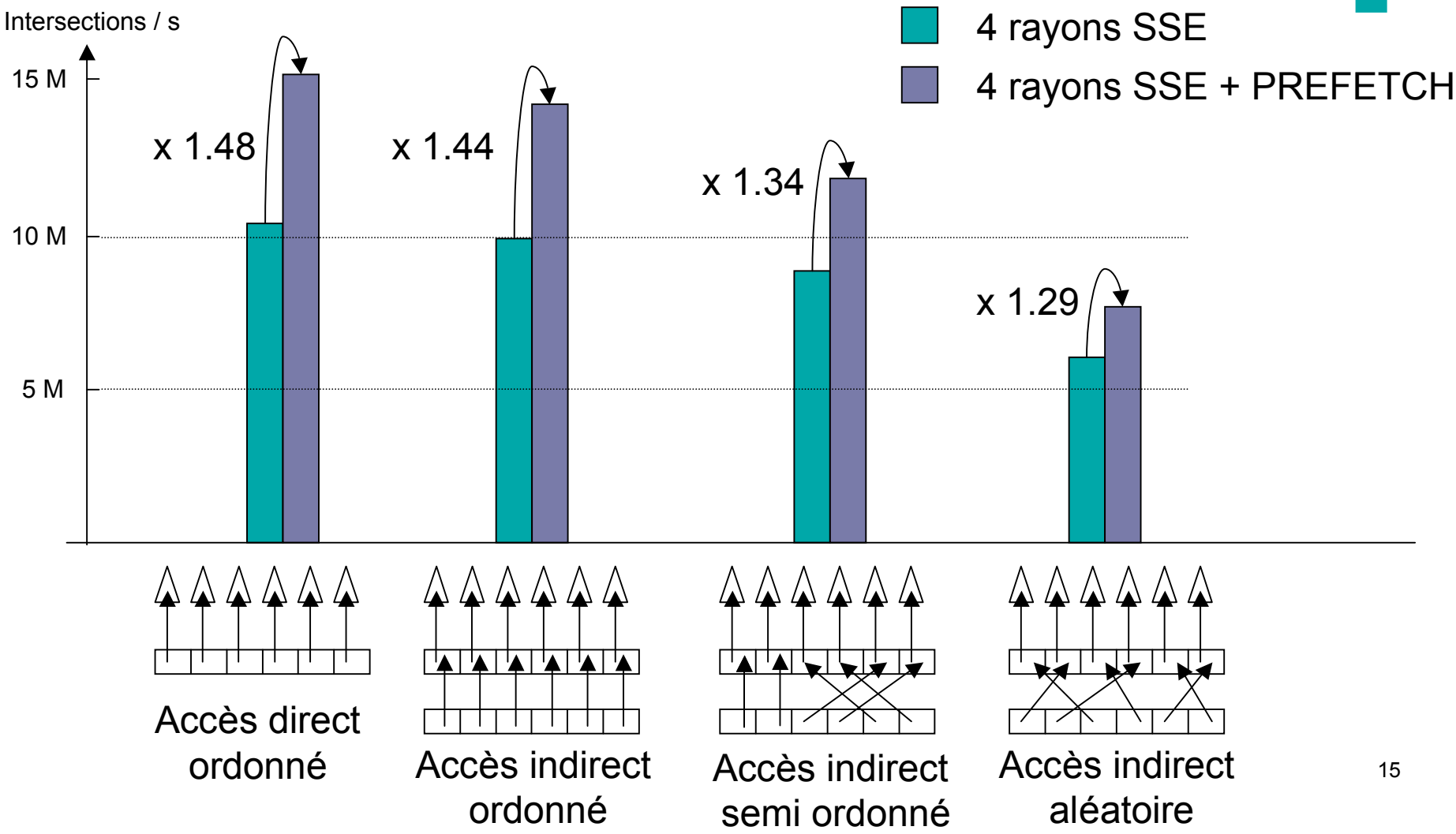
- Parcours beaucoup plus rapide dans la grille
- Intersection plus lente car plus d'objets et moins de cohérence

- Grille plus adaptée aux scènes de densités uniformes
- On peut envisager la multigrille

Pré-cachage

- Instruction PREFETCH : amène données de la RAM vers le cache
  - Problème de la distance de pré-cachage (3 intersections en avance)
  - Nombre limité de PREFETCH ‘en cours’ : 6
- En fait peu de triangles pré-cachés car:
  - Dans le KD-arbre: meilleure performance avec 3 ou 4 triangles par feuille.
  - Dans grille : avec 6 ou 7 triangles par feuilles.
- Déplacement en avance sur intersection : aucun gain

Pré-cachage et mémoire

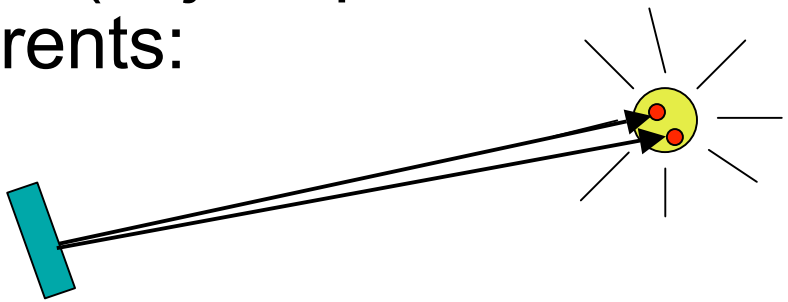


Plan

- Interactive Rendering With Coherent Raytracing
- Comparaison Grille – KD-arbre
- Échantillonnage de sources étendues
- Échantillonnage d'un hémisphère
- Conclusion

Échantillonnage d'une source sphérique

- Les rayons (objet points de la source) sont très cohérents:

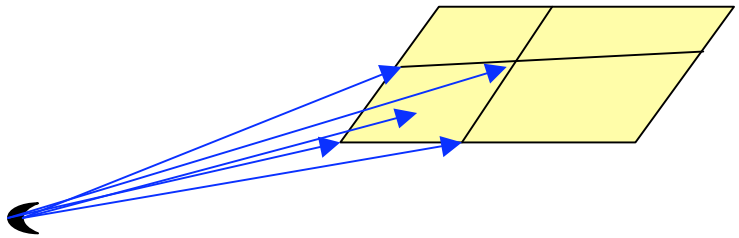


-Test sur le dernier objet intercepté
  on évite le passage par la structure de données

Gains (sur le calcul complet) lorsqu'on envoie les rayons par 4 : **x 2**

Échantillonnage d'une source surfacique

- Zanninetti propose une méthode d'échantillonnage adaptatif:
  - 5 rayons par cellule avant de les subdiviser si besoin



- Difficulté:
  - Combiner envoi des rayons par 4 et réutilisation des valeurs de rayons du niveau précédent:
  - Coût des calculs nécessaires au regroupement des rayons proches supérieur ou égal au gain apporté par l'envoi de rayons plus cohérents
- Gain de 1.5

Plan

- Interactive Rendering With Coherent Raytracing
- Comparaison Grille – KD-arbre
- Échantillonnage de sources étendues
- Échantillonnage d'un hémisphère
- Conclusion

Échantillonnage de l'illumination indirecte

- Étude de la cohérence de 4 rayons lors de l'échantillonnage à travers un hémisphère

  Angle solide de 4 rayons:

  - Pour 1024 rayons/hémisphère : 6,16 10^{-3} sr
  - Les 4 rayons primaires :1,28 10^{-4} sr

  Gain:

  - Pour une scène simple (*60 triangles*):

    Pour 1024 rayons/hémisphère: **x 1.3**

    Pour 100 rayons/hémisphère: **x 1.18**

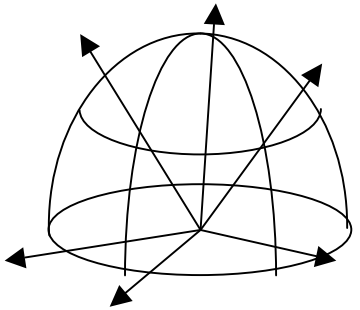
  - Pour une scène complexe (*65000 triangles*):

    Pour 1024 rayons/hémisphère: **x 1.2**

    Pour 100 rayons/hémisphère : **x 0.96** (pénalisant)

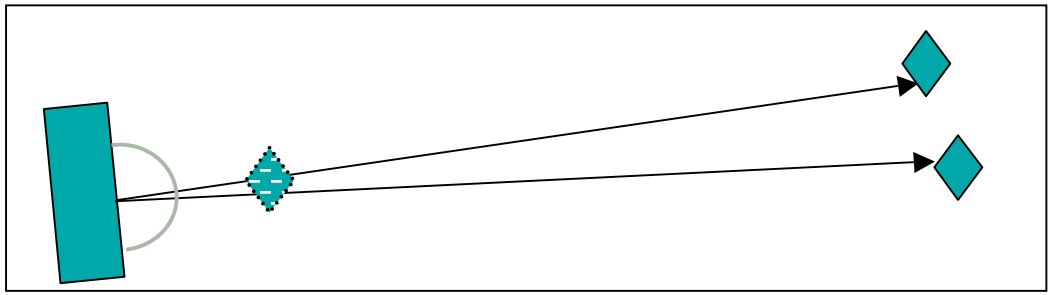
  - Pour une scène très complexe (*240 000 triangles*)

    X 0.60 à x0.70 : envoyer 4 rayons est très pénalisant



Échantillonnage de l'illumination indirecte

- On ne peut pas envoyer les rayons par 4 dans toutes les situations.
  - Quelle probabilité d'avoir un gain de temps ?
  - Forte probabilité si les rayons sont vite arrêtés



- Besoin d'un algorithme d'échantillonnage progressif
  - Subdivision progressive des cellules de l'hémisphère
  - A partir d'un seuil sur l'angle solide et la distance de la précédente intersection au sein de la cellule
    - Envoi des rayons par 4

Plan

- Interactive Rendering With Coherent Raytracing
- Comparaison Grille – KD-arbre
- Échantillonnage de sources étendues
- Échantillonnage d'un hémisphère
- Conclusion

Conclusion

- L'envoi des rayons par 4 apporte un gain de temps important:
  - X 3 sur l'intersection des triangles
  - Plus limité sur le parcours de la partition de l'espace
    - La grille semble plus adaptée que l'arbre
  - Difficulté d'implémentation du pré-cachage
- Adapté à l'échantillonnage de sources étendues
- L'échantillonnage d'un hémisphère nécessite plus de travail:
  - envoyer les rayons par 4 uniquement lorsqu'on en attend un gain réel. Envoyer les rayons par 4 peut être plus coûteux que les envoyer un par un.

Références

- I. Wald, C. Benthin, M. Wagner, P. Slusallek.
  Interactive rendering with coherent raytracing,
  Eurographics 2001
- J. Zaninetti, Xavier Serpaggi, Bernard Péroche.
  A vector approach for global illumination in ray tracing,
  Computer Graphics Forum 1998
- X. Serpaggi, B. Péroche.
  An adaptative methode for indirect illumination using light vectors,
  Computer Graphics Forum 2001
- V. Havran, T. Kopal, J. Bittner, J. Zara.
  Fast robust BSP tree traversal algorithm for ray tracing,
  Journal of Graphics Tools 1997
- J. Amanatides, A. Woo.
  A fast voxel traversal algorithm for ray tracing,
  Eurographics1987