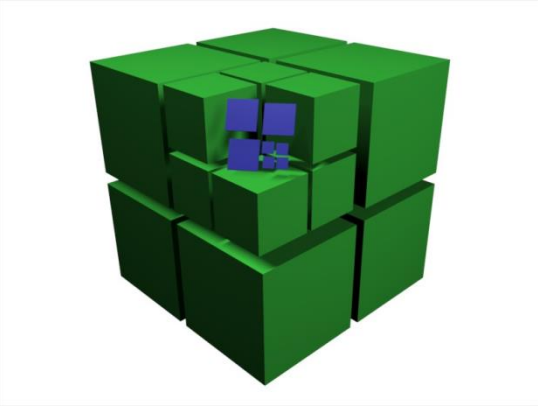
Informatique Graphique 3D & Réalité Virtuelle

Synthèse d'Images

Structures Spatiales

Tamy Boubekeur



Structurer l’espace 3D

- Grille Spatiale
- Arbres dimensionnels
  - Quadtrees
  - Octrees
- kD-Trees
- BSP-Trees
- Implémentation
- Hiérarchies de Volumes Englobants
- Commentaires

Données 3D

- Géométrie + Attributs
  - Apparence + Animation + Physique + …..
- Représentations dépendante de l'application

- Géométrie explicite nécessaires
  - A minima : P, un ensemble de points 3D
    - *Echantillon p de P = point {x,y,z}*

Livre : *Fundations of Multidimentional and Metric Data Structures*, par Hanan Samet

Données Géométriques 3D

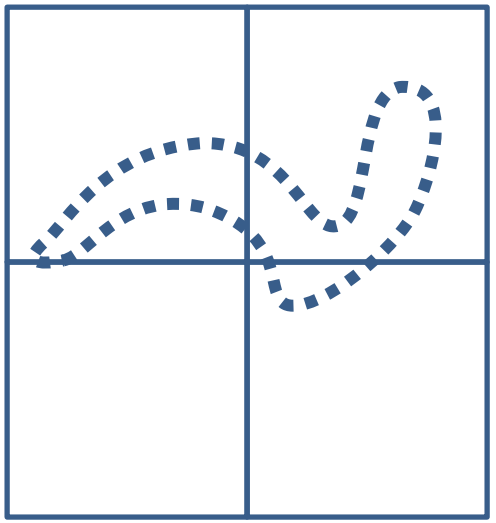
- Champs d'applications :
  - Modélisation
  - Synthèse d'Images
  - Animation
- Traitement local
  - Éviter d'inspecter l'ensemble du modèle 3D pour une requête spatialement défini

  → ***Partitionnement de l'objet***
- Découper l'objet en régions et les **indexer** spatialement
- Organiser les régions entre elles pour accélérer les requêtes de voisinage

Grille 3D

La solution la plus simple :

- Générer une grille 3D contenant l'objet
- Répartir les points du modèle dans les cellules les contenant
  - *Discrétisation 3D*
- Efficace, simple et parfois suffisant
- Non adaptatif
  - beaucoup de cellules vides
  - d'autres trop peuplées
- Résolution décidée à priori : accès à un élément en O(1)
  - Réduction des contantes, pas de l'ordre de complexité
- Solution : **les arbres !**



Structures de Subdivision Spatiale Hiérarchique

Omniprésentes en informatique graphique:

- Modélisation
  - *Partitionnement, analyse, simplification, reconstruction, génération de niveaux de détails, etc*
- Rendu
  - *Lancer de rayons, photon mapping, radiosité, approximation de l'illumination incidente, visibilité, etc*
- Animation
  - *Détection de Collision, simulation de foules, déformation basée physique*
- Réalité virtuelle
  - *Graphe de Scène, sélection de niveaux de détails, parallélisation, etc*

Structures de Subdivision Spatiale Hiérarchique

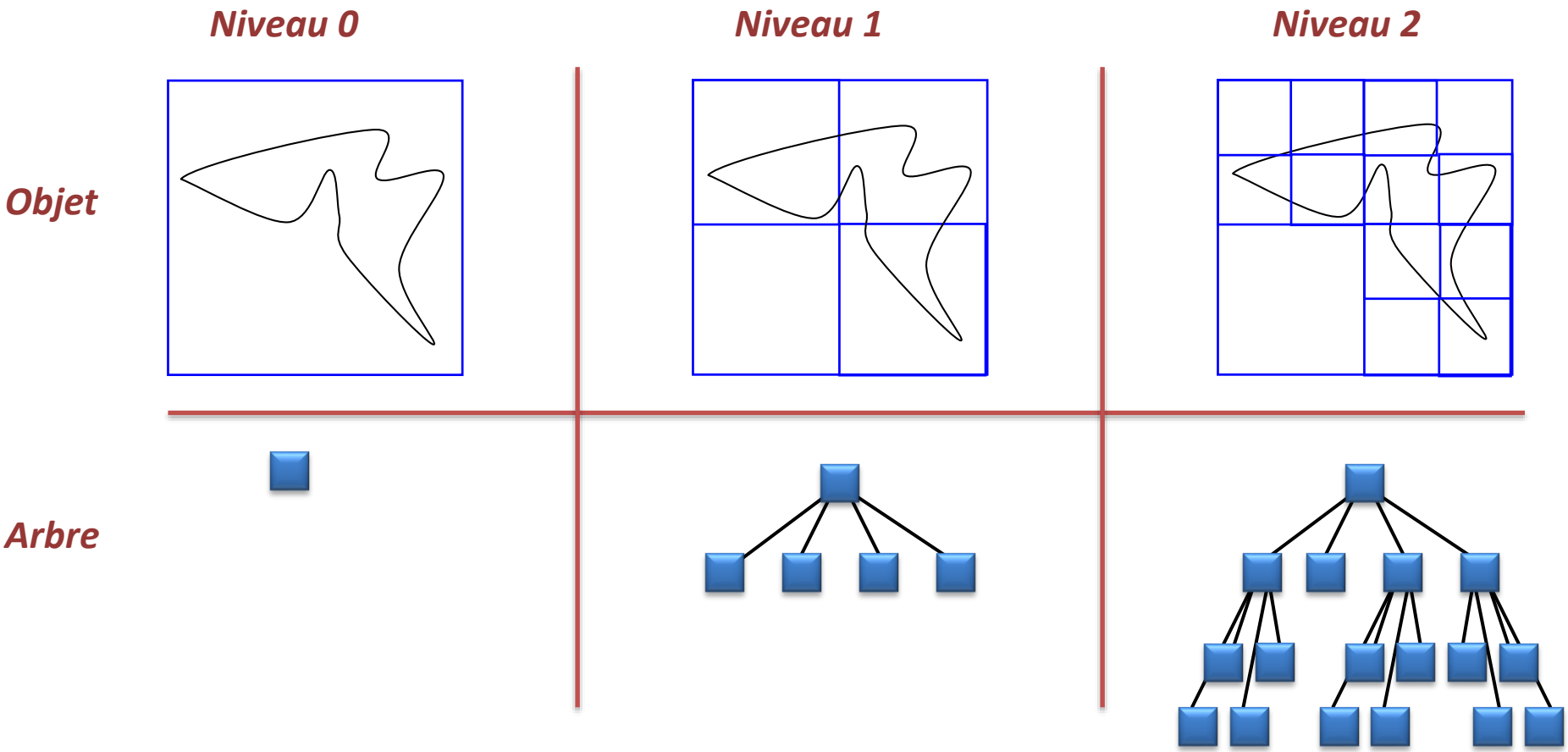
- Principales structures:
  - **kD-Tree** [Bentley 1975] **:** organisation orthogonale d'un ensemble d'échantillons
  - **BSP-Tree** [Fuchs et al. 1980] **:** Subdivision binaire et récursive de l'espace par des hyperplans
  - **Quadtree/Octree** [Jackins & Tanimoto 1980] **:** dimension de l'espace dans la structure (subdivision 1-pour-4 en 2D, 1-pour-8 en 3D)
  - **BVH** : Hiérarchie de Volumes Englobant
- **De très nombreuses utilisation et combinaisons dans la littérature.**

Structures de Subdivision Spatiale Hiérarchique

- Principaux opérateurs:
  - **Intersection**
    - **Point:** quelle partition contient un point 3D donné ?
    - **Rayon:** comment ordonner les partitions le long d'un rayon ?
  - **Requête de voisinage:** soit un point p, quels sont
    - Parmi les éléments de **P**, quels sont les k plus proches de p ?
    - Parmi les éléments de **P**, quels sont ceux situés à un distance maximale donnée de p ?
  - **Parcours des partitions**

QuadTree

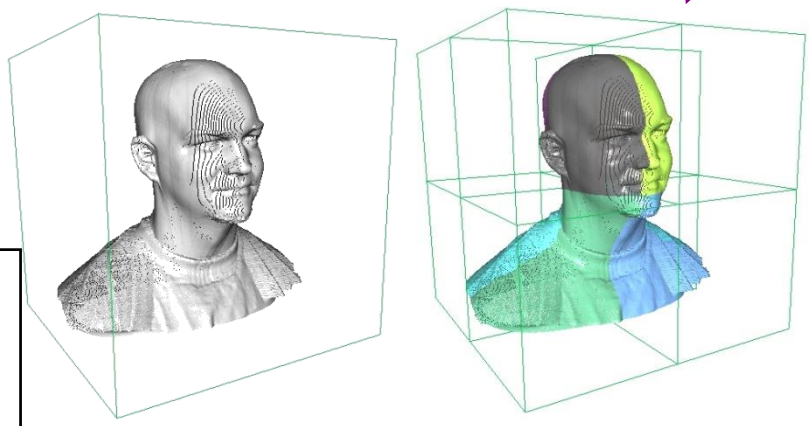
- Séparation récursive 1-4 d'un domaine planaire
- Organisation des partition dans un arbre quaternaire



Octree

- Généralisation 3D du Quadtree
- Hiérarchie de cube englobant

Partitionnement de la surface induit par l'octree



Niveau 0 ***Niveau 1***

Implémentation récursive simple

```
TYPE OctreeNode {
  OctreeNode children[8];
  Data data;  // Bounding Cube + misc. data (e.g. 3D points)
}
OctreeNode buildOctree (Data data) {
   OctreeNode node;
   if (stopCriteria (data))
      init (node, data); // fill children with NULL and affect data
   else
      Data childData[8];
      dataSpatialSplit (data, dataChild); // 8-split of the bounding
                                          // cube and partitioning of data
      for (int i = 0; i < 8; i++)
        node.children[i] = buildOctree (childData[i]);
        node.data = NULL;
   return node;
}
```

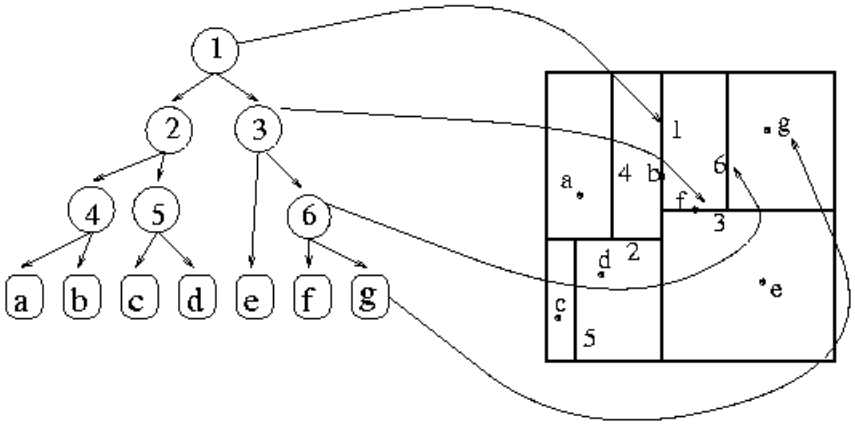
Peut également être implémenter

- dans une table
  - sans pointeur, arbre quasi-parfait)
- avec une table de hashage
  - code de Morton

kD-Tree

- Structure de partitionnement ***orthogonale*** d'échantillons
- Arbre binaire. A chaque niveau:
  - Calculer la boite englobante de **P**
  - Diviser **P** le long du plus grand axe (X, Y ou Z)
- Algorithme de construction:

```
KDNode buildKDTree (PointList P) {
        BBox B = computeBoundingBox (P);
        Point q = findMedianSample (B,P);
        Node n;
        Plane H = plane (q, maxAxis (B)
        n.data = <q,H>;
        PointList Pu = upperPartition (P, H);
        PointList Pl = lowerPartition (P, H);
        n.leftChild = buildKDTree (Pu);
        n.rightChild = buildKDTree (Pl);
        return n;
}
```



Propriétés du kD-Tree

Générique

- Ordonnancement spatial en dimension arbitraire
- Robuste et constructible sur n'importe quel ensemble de points

Accélération de la recherche des plus proches voisins (NN)

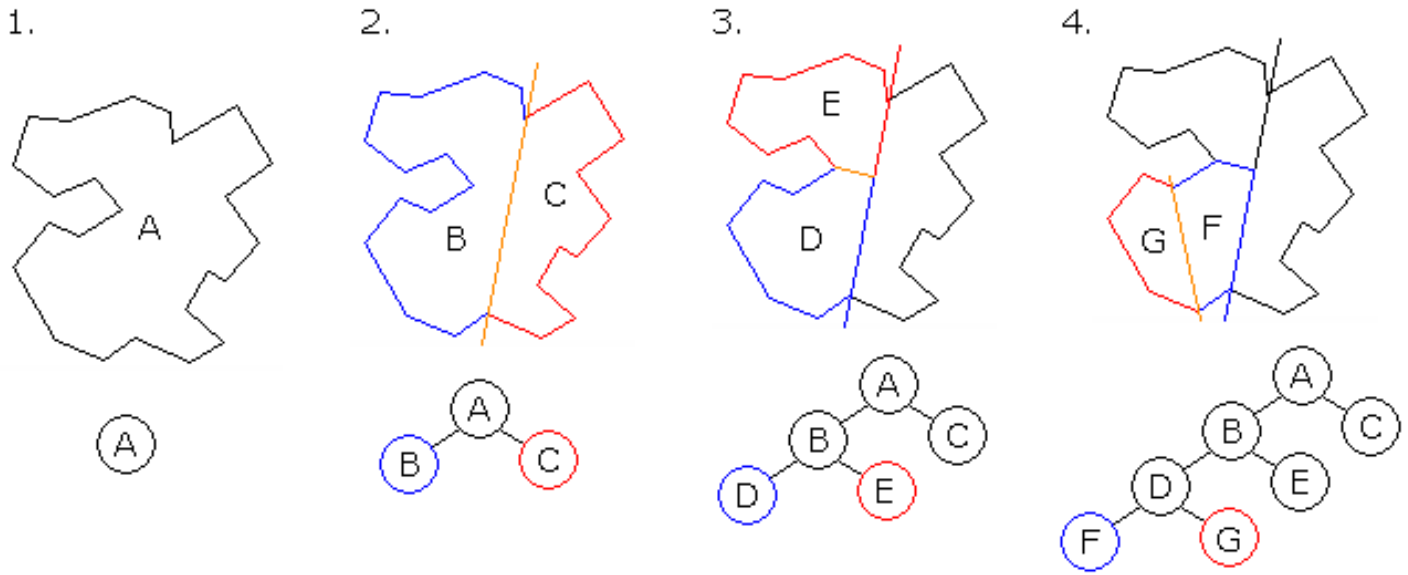
- Recherche par distance: tous les points à située dans une boule de rayon r centrée sur l'origine de la recherche
  - Basées sur le test d'intersection sphère/boite
- Recherche par cardinal: trouver les k plus proches voisins (kNN)
  - via une file à priorité de taille maximum pour ordonner les points (O(k log N) pour une arbre équilibré

Accélération des tests d'intersection

- Test récursif
- Traitement de « paquets de rayons »
- Possibilité d'ajouter un biais géométrique
  - e.g. *Surface Area Heuristic* ou SAH

BSP Tree

- Arbre binaire de partitionnement spatial
- Chaque nœud définit un hyperplan séparant son sous-espace associé



BSP Tree

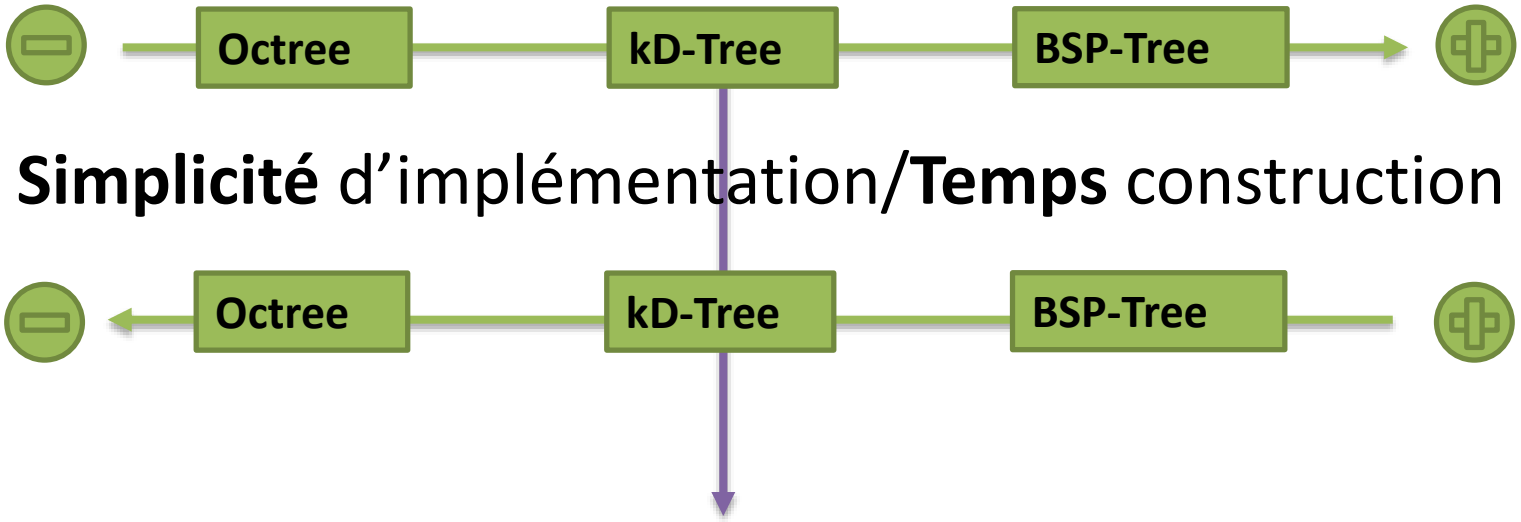
- Calcul des plans de coupe:
  - Alignés sur les axes
    - découpage similaire à un kD-Tree
  - Analyse en composante principale (*top-down*)
    - ACP sur P
    - Subdivision de P selon le plan définit par la barycentre de P et le vecteur propre associé à la plus grande valeur propre

- *Différence kD-Tree / BSP alignés sur les axes ?*
  - kD-Tree : structure d'organisation de P, partition spatiale de faible qualité
  - BSP : meilleurs partition spatiales, subdivision binaire générique (non limitée aux plans)

Notes : rien n'est fixé, et tout dépend de l'implémentation choisie et de l'application ciblée.

Comparaison

Adaptivité à une profondeur donnée



Simplicité d'implémentation/**Temps** construction

Bon Compromis

Heuristique efficaces pour (quasi) garantir un temps d'accès en O(log(N)) (test d'intersection)

Comparaison

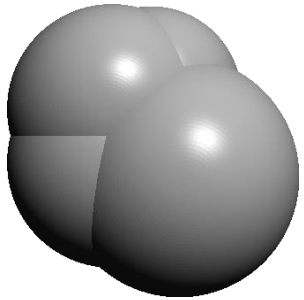
Grille

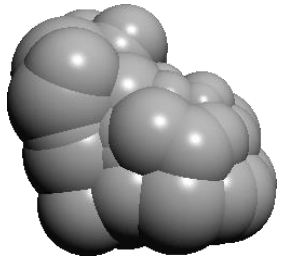
BSP

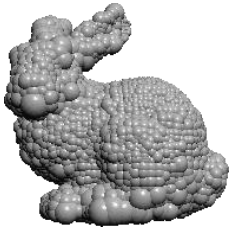
Octree

Hiérarchie de Volumes Englobants (BVH)

- BVH vs Octree/kd-Tree/BSP-Tree:
  - Noeud != somme de ses enfants
  - Intersection entre volumes englobants (BV)
  - Valence arbitraires des nœuds de l'arbre
  - Forme arbitraire des volumes de partitions
    - En général convexes pour des tests d'intersections plus simples



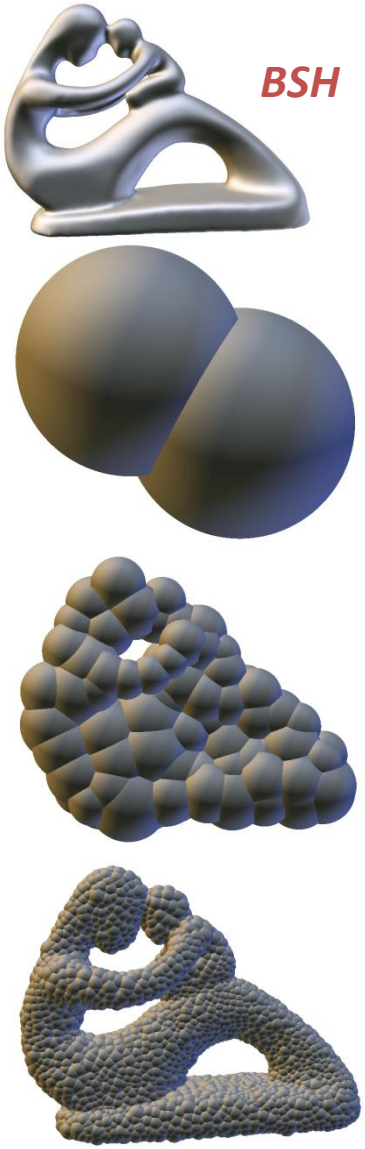




BSH

Hiérarchie de Volumes Englobants (BVH)

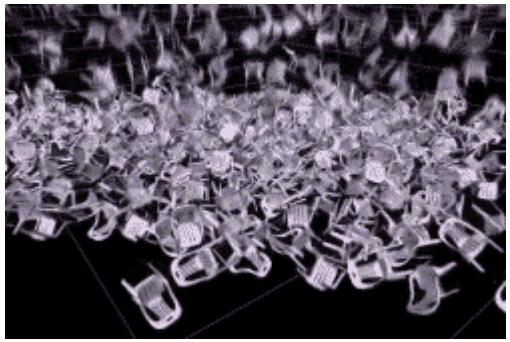
- Construction :
  1. **Trouver le plus petit BV** de P
  2. Subdiviser BV en n sous-BV
  3. Classifier P dans les n sous-BV et recommencer
- BV hanbituels :
  - Sphère englobante (Bounding Sphere ou BS)
  - Boîte englobante (Axis-Aligned Bounding Box ou AABB),
  - Boîte orientée (Oriented Bounding Box ou OBB)
  - Polyhèdre convexe à k faces (k-DOP)



Hiérarchie de Volumes Englobants

Bien adaptés aux modèles dynamique en animation :

- **BD-Trees** pour les collisions
- Lancer de rayon interactif de scènes dynamiques
- Plus simple à construire qu'un kD-tree
  - Pas de découpe de triangle



Structure de base d'un graphe de scène

Inventor, OpenSG, OpenSceneGraph, NVSG, VRML, X3D, COLLADA… la plupart des graphes de scènes sont des BVH enrichis

Critère de Partitionnement

- *Quelle condition d'arrêt pour la récursion ?*
  - Profondeur maximale : uniforme, prédiction difficile
  - Densité : m points par feuille au maximum
  - Métrique d'erreur : l'ensemble contenu dans une feuille ne doit pas violer un certain **prédicat :**
    - **Géométrique** (e.g., variance spatiale, courbure maximale)
    - Visuel (e.g., saillance, variance chromatique)
    - Etc

Implémentation Séquentielle

- Pointeurs: simple, facile à manipuler
- Décalage explicite: arbres complets
  - Chaque nœud interne a exactement **d** fils
- Table de hachage: définir une clé spatiale
  - e.g. code de Morton
  - Accès en temps quasi-constant
- Construction:
  - En largeur (BF) : à l'aide d'une file, permet d'atteindre un « budget » de nœud de manière équilibrée
  - En profondeur (DF) : à l'aide d'une pile (implicite dans une construction récursive)

Implémentation Parallèle

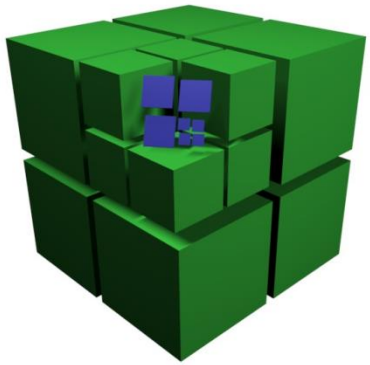
Construction sur GPU/CPU multicoeurs ?

- Toujours plus ou moins un problème ouvert
- Une stratégie simple :
  1. construction BF sur quelques niveaux afin de générer suffisamment de sous-arbres à construire en parallèle
  2. Construction parallèle des sous-arbres
- Sous-optimal lorsque la densité de P varie fortement dans l'espace
- Implémentation hybride possible : (1) sur CPU, (2) sur GPU

Optimisation

- Complétion partielle :
  - Rendre *parfaits* les M premiers niveaux de l'arbre
    - Accès *direct* jusqu'à la profondeur M (équivalent à une grille)
    - Réduit fortement le temps d'accès pour un faible coût mémoire
- *Lifting scheme*
  - « décalage local » pour l'implémentation par tableaux
    - Décalage **quantifié** sur peu de bits
- « Exploiter les nœuds internes »
  - Données additionnelle par nœud :
    - Description grossière de leur sous-arbre.

Arbres exotiques



De nombreux arbres sont spécialisés à certaines applications :

- **Volume-Surface Tree**: combine un octree et une forêt de quadtree locaux pour la partitionnement de surface, adapté au traitement géométrique des surfaces
- **Tile-Trees**: arbres de « faces de cubes » pour la plaquage de textures
- **Bounding Interval Hierarchy,** pour le lancer de rayons
- **Tetrahedra Hierarchy**: hiérarchie de simplexes pour les volumes

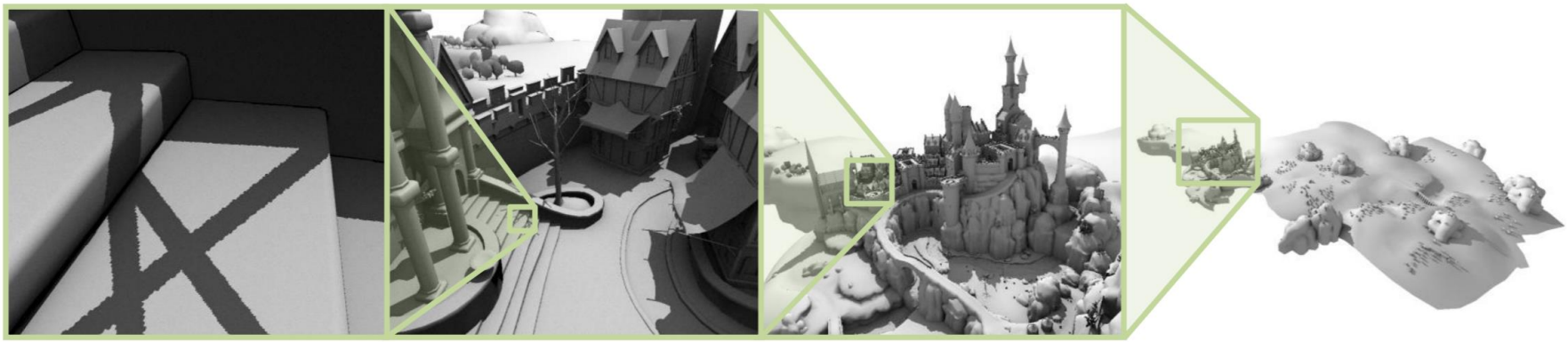
Structures hybrides

Grilles récursives

- Hiérarchie de grilles
- Généralisation des octrees
- Duales ou primales

- En pratique : seuls quelques niveaux nécessaires
- Très efficace lorsque le contrôle précis de la mémoire consommée n’est pas critique

Structures parcimonieuses



- Pour modéliser spatialement la pellicule de volume englobant une surface, à très haute résolution
- Typiquement employé pour décrire l'occupation de l'espace, et résoudre les questions de visibilité
- **Sparse-Voxel Octree (SVO)** : Octree codant en chaque nœud uniquement la présence où des fils
- **Sparse-Voxel Directed Acyclic Graphs (SVDAGs)** : SVO factorisé
- **Voxel Hashing** : table de hachâge spatiale