Space Partitions
BSP & Quadtree

ZHENG Yufei
Jan. 24, 2017
Motivation

- **Hidden Surface Removal** — determine for each pixel on the screen the object that is visible at that pixel

- **z-buffer algorithm**
  - maintains 2 buffers:
    - *frame buffer* stores for each pixel the *intensity* of the currently visible object
    - *z-buffer* stores the *z-coordinate* of the point on the object that is visible at the pixel
  - select a pixel
    - If *z-coordinate* of the object at that pixel < the *z-coordinate* in *z-buffer*,
      - *frame buffer* ← intensity of the new object
      - *z-buffer* ← *z-coordinate*
Motivation — z-buffer Alg. vs. Painter’s Alg.

- **Disadvantage of z-buffer Algorithm** —
  - Extra storage needed for the z-buffer
  - Extra test on z-coordinate required for every pixel covered by the object

- **Painter’s algorithm**
  - Sorting the objects according to their distance to the view point (Avoid extra costs)
  - Objects are scan-converted in **depth-order**
Motivation – Problems with Painter’s Alg.

- Sort the objects quickly
- Depth order may not always exist
- Cyclic overlap

Solutions – split one or more of the objects till depth order exists for pieces.

Binary space partition tree (BSP tree)
Binary Space Partition (BSP)

- **BSP** is obtained by recursively splitting the plane with a line.
- Splitting lines partition the plane and cut objects into fragments.
- Splitting stops when there is only one fragment in each region.
**BSP for a set $S$ in $\mathbb{R}^d$ — Definition**

- Hyperplane $h$: $a_1 x_1 + a_2 x_2 + \cdots + a_d x_d + a_{d+1} = 0$
  
  $$h^+ := \{(x_1, x_2, \ldots, x_d): a_1 x_1 + a_2 x_2 + \cdots + a_d x_d + a_{d+1} > 0\}$$
  
  $$h^- := \{(x_1, x_2, \ldots, x_d): a_1 x_1 + a_2 x_2 + \cdots + a_d x_d + a_{d+1} < 0\}$$

- BSP tree is defined as a binary tree $T$ with the following properties:
  
  1. If $|S| \leq 1$, $T$ is a leaf. The object fragment in $S$ (if exists) is stored at this leaf.
  2. If $|S| \leq 1$, root $v$ of $T$ stores a hyperplane $h_v$. Left child of $v$ is $T^-$ for the set $S^- := \{h_v \cap s: s \in S\}$, right child of $v$ is $T^+$ for the set $S^+ := \{h_v \cap s: s \in S\}$. 
BSP

- A node in BSP and its corresponding convex region
BSP for line segments in $\mathbb{R}^2$ - Construction

- $S = \{s_1, \ldots, s_n\}$ is a set of $n$ non-intersecting line segments in the plane
- Only consider lines containing one of the segments in $S$ as candidate splitting lines (auto-partitions)

Algorithm 2DBSP($S$)

- If $|S| \leq 1$
  
  create $T$ with a single leaf node where $S$ is stored

- Else

  $S^- := \{s \cap l(s_1)^- : s \in S\}$, $T^- \leftarrow 2DBSP(S^-)$
  
  $S^+ := \{s \cap l(s_1)^+ : s \in S\}$, $T^+ \leftarrow 2DBSP(S^+)$
  
  create $T$ with root node $v$, left subtree $T^-$, right subtree $T^+$, and $S(v) = \{s \in S : s \subseteq l(s_1)\}$

  Return $T$
BSP Construction
- Difficult choice \(\Rightarrow\) random choice

**Algorithm** 2DRandomBSP\((S)\)
- Generate a random permutation \(S' = s_1, \ldots, s_n\) of set \(S\)
- \(T \leftarrow 2DBSP(S')\)
- Return \(T\)

**Lemma:** the expected number of fragments generated by the algorithm 2DRandomBSP is \(O(n \log n)\).

**Proof:**
- Let \(s_i\) be a fixed segment in \(S\)
- Analyze the expected number of other segments that are cut when \(l(s_i)\) is added
Proof Continued

- Define the distance of a segment w.r.t. the fixed $s_i$

\[
\text{dist}_{s_i}(s_j) = \begin{cases} 
\text{the number of segments intersecting } & \text{if } \ell(s_i) \text{ intersects } s_j \\
\ell(s_i) \text{ in between } s_i \text{ and } s_j & +\infty \text{ otherwise}
\end{cases}
\]

- Bound the probability that $l(s_i)$ cuts $s_j$

\[
\Pr[\ell(s_i) \text{ cuts } s_j] \leq \frac{1}{\text{dist}_{s_i}(s_j) + 2}
\]

- Bound the expected total number of cuts generated by $s_i$

\[
E[\text{number of cuts generated by } s_i] \leq \sum_{j \neq i} \frac{1}{\text{dist}_{s_i}(s_j) + 2} \leq 2 \sum_{k=0}^{n-2} \frac{1}{k + 2} \leq 2 \ln n.
\]
Proof Continued

- By linearity of expectation, conclude that the expected total number of cuts generated by all segments is at most $2n \ln n$.

- Expected total number of fragments is bounded by $n + 2n \ln n$

**Theorem:** BSP of size $O(n \log n)$ can be computed in expected time $O(n^2 \log n)$
BSP for triangles in $\mathbb{R}^3$ - Construction

- $S = \{t_1, \ldots, t_n\}$ is a set of $n$ non-intersecting triangles in $\mathbb{R}^3$.
- Only use partition planes containing a triangle of $S$ (auto-partitions).

**Algorithm 3DBSP($S$)**

- **If** $|S| \leq 1$
  - create $T$ with a single leaf node where $S$ is stored.
- **Else**
  
  $S^- := \{t \cap h(t_1)^- : t \in S\}$, $T^- \leftarrow 3DBSP(S^-)$
  
  $S^+ := \{t \cap h(t_1)^+ : t \in S\}$, $T^+ \leftarrow 3DBSP(S^+)$

  create $T$ with root node $v$, left subtree $T^-$, right subtree $T^+$, and $S(v) = \{t \in S : t \subseteq h(t_1)\}$

  Return $T$
Quadtree

**Definition** – a tree data structure in which each internal node has exactly four children.

Used to **divide** a 2D region into more manageable parts.

**Nodes** – axis-aligned squares
Quadtrees

- Starts as a single node
- Splits into 4 subnodes when more objects are added
- object that cannot fully fit inside a node’s boundary will be placed in the parent node
- Continue subdividing till the number of objects in each cell is $O(1)$
Depth of Quadtrees of Point Sets

Lemma: the depth of a quadtree of point set $S$ with minimal distance $c$ and bounding box of side length $s$ is at most $\log\left(\frac{s}{c}\right) + \frac{3}{2}$.

Proof:
- Side length of a square at depth $i$ is $\frac{s}{2^i}$
- Maximum distance between 2 points inside a square is the length of the diagonal, $\frac{\sqrt{2}s}{2^i}$
- An internal node at the ‘second last level’ has at least 2 points, denote its depth $d$
Proof - Continue

- Internal node at depth $i$ must satisfy:
  \[ \frac{\sqrt{2s}}{2^d} \geq c \Rightarrow d \leq \log_2 \left( \frac{\sqrt{2s}}{c} \right) = \log_2 \left( \frac{s}{c} \right) + \frac{1}{2} \]

- Depth of leaf is at most $d + 1$.

◎ If $s \gg c$, the tree is far from being balanced