Lecture 9: Point Location & Trapezoidal Maps
Outline for Today

- Homework 3 Questions?
- Last Time: kD Trees & Range Trees
- Motivating Application: Point Location
- Motivating Application: 2D/3D Mouse “Picking” for Graphics
- Brute Force Point Location
- Point Location by Vertical Slab
- Trapezoidal Map & Adjacency Structure
- Trapezoidal Map Analysis & Construction
- Think-Outside-of-the-Box Graphics Picking Algorithm
- Next Time:
Homework 3 - CGAL Programming Task

- Compute triangulation of input polygon
  & triangulation of “pockets” outside
  input polygon but inside convex hull
- Compute areas
- Compute changes to
  boundary edges
- Leverage CGAL libraries for
  convex hull & triangulation

*Computational Geometry Algorithms and Applications, de Berg, Cheong, van Kreveld and Overmars, Chapter 3*
Outline for Today

● Homework 3 Questions?
● Last Time: kD Trees & Range Trees
● Motivating Application: Point Location
● Motivating Application: 2D/3D Mouse “Picking” for Graphics
● Brute Force Point Location
● Point Location by Vertical Slab
● Trapezoidal Map & Adjacency Structure
● Trapezoidal Map Analysis & Construction
● Think-Outside-of-the-Box Graphics Picking Algorithm
● Next Time:
Higher Dimensional Database Queries

- Return all data points with
  - $x$ value in range $[x_0, x_1]$
  - $y$ value in range $[y_0, y_1]$
  - $z$ value in range $[z_0, z_1]$
  - and …

Find all values in an axis parallel box:
  - a "rectangular range query"
  - a.k.a. "orthogonal range query"

*Computational Geometry Algorithms and Applications, de Berg, Cheong, van Kreveld and Overmars, Chapter 5*
Using Photon Map for Rendering

- Find the tightest sphere capturing $k$ photons
- Divide the energy from those photons by the surface area covered by that sphere
- What is the best data structure to store millions of photons?
2D kd Tree Query Algorithm

- At each split point
- Determine if the query box overlaps the split line
- Recurse down one or both branches
- If a subtree lies complete inside the box, return all items in that subtree
- Perform filtering in the leaves as necessary

*Computational Geometry Algorithms and Applications, de Berg, Cheong, van Kreveld and Overmars, Chapter 5*
2D kd Tree Query Analysis

- 1 item is stored per leaf node
- For a query that will collect \( k \) items
- **Best/Average(?) Case:**
  - An approximately square query (equal width & height)
  - touches/overlaps \( O(k) \) leaves
  - gathering leaves \( O(\log n + k) \)
  - \( \text{Overall} \rightarrow O(\log n + k) \)
- **Worst Case Query:**
  - For a skinny / lopsided query box
  - touches/overlaps \( - \sqrt{n} + k \) leaves
  - gathering leaves \( O(\sqrt{n} + k) \)
  - \( \text{Overall} \rightarrow O(\sqrt{n} + k) \)
Is Query Time $= O(\sqrt{n} + k)$ a problem?

- $O(1) \ < \ O(\log n) \ < \ O(\log^2 n) \ < \ O(\sqrt{n}) \ < \ O(n)$
2D Range Tree (and higher dimension!)

How much memory does it use?

- Each point $p$ is stored once in the level 1 (organized by $x$) tree
- And many times in level 2 (organized by $y$) trees
- How many level 2 trees? And how big are they?
  - 1 tree with $n$ values
  - 2 trees with $n/2$ values
  - 4 trees with $n/4$ values
  - ...
  - $n$ trees with 1 values

$\rightarrow O(n \log n)$ memory
Summary Comparison

- For $n$ points, dimension $d$, with query to collect $k$ items
- $kd$ Tree
  - Construction time: $O(n \log n)$
  - Memory: $O(n)$
  - Query time
    - Square(ish) box: $O(\log n + k)$
    - Worst case (long, skinny box): $O(n^{(1-1/d)} + k)$
- Range Tree
  - Construction time $\rightarrow O(n \log^{d-1} n)$
  - Memory $\rightarrow O(n \log^{d-1} n)$
  - Query time $\rightarrow O(\log^d n + k)$

Tradeoff:
Use more memory
Faster runtime
Outline for Today

- Homework 3 Questions?
- Last Time: kD Trees & Range Trees
- Motivating Application: Point Location
- Motivating Application: 2D/3D Mouse “Picking” for Graphics
- Brute Force Point Location
- Point Location by Vertical Slab
- Trapezoidal Map & Adjacency Structure
- Trapezoidal Map Analysis & Construction
- Think-Outside-of-the-Box Graphics Picking Algorithm
- Next Time:
Motivation Application: GPS Point Localization

- Given a 2D coordinate, e.g., a latitude & longitude
- What region of the ocean contains this point?
  - Access currents, weather, etc.

NASA Scientific Visualization Studio
https://svs.gsfc.nasa.gov/
Graphics/VR Application: What is “Picking”? 

- Get the (3D) world coordinates of a (2D) mouse click 
- Identify which object was selected and the point on the object closest to the click 
- Do we as users take this for granted?? 
  - What are the performance bottlenecks? 
  - What are the usability concerns? 

https://www.csit.carleton.ca/~rteather/pdfs/GI_2018_EZCursorVR.pdf
Graphics Application: 3D Painting

http://www-ui.is.s.u-tokyo.ac.jp/~takeo/gallery/chameleon.png
Outline for Today

- Homework 3 Questions?
- Last Time: kD Trees & Range Trees
- Motivating Application: Point Location
- Motivating Application: 2D/3D Mouse “Picking” for Graphics
- **Brute Force Point Location**
- Point Location by Vertical Slab
- Trapezoidal Map & Adjacency Structure
- Trapezoidal Map Analysis & Construction
- Think-Outside-of-the-Box Graphics Picking Algorithm
- Next Time:
“Picking” by Ray Tracing

- Construct a ray from the eye through the image plane into the scene
- Intersect with all objects in the scene
- Keep the closest

Concerns:
- Cost of intersection
- How often are you asking?
  - on click
  - continuously
- Position imprecision/noise
Brute Force Picking Algorithm

- Given a planar subdivision
  - E.g., a collection of non-overlapping triangles (or polygons) that cover the plane
- And a query point $Q$
- Which triangle/polygon is $Q$ inside of?
  - E.g., $T_7$
Is Query Point *inside* a specific Triangle?

- Compare the point to each line segment
- Are you on the “right side” of all three line segments?
- Are you on the “wrong side” of one or two segments?

- Use cross product! (more on this later…)
Is Query Point *inside* a specific Triangle?

- Does the half edge adjacency data structure accelerate this query?
Is Query Point *inside* a specific Triangle?

- Does the half edge adjacency data structure accelerate this query?
  - *Unfortunately* … NO!
  - *While we can navigate to the adjacent neighbors, we can NOT do better than a* $O(n)$ *linear floodfill to find the correct triangle.*
Outline for Today

- Homework 3 Questions?
- Last Time: kD Trees & Range Trees
- Motivating Application: Point Location
- Motivating Application: 2D/3D Mouse “Picking” for Graphics
- Brute Force Point Location
- **Point Location by Vertical Slab**
- Trapezoidal Map & Adjacency Structure
- Trapezoidal Map Analysis & Construction
- Think-Outside-of-the-Box Graphics Picking Algorithm
- Next Time:
Point Location in Planar Subdivision

- Given $v$ vertices, $n$ edges, and $f$ polygonal faces
- Which polygonal region contains the query point $Q$?
Point Location in Planar Subdivision

- Given \( v \) vertices, \( n \) edges, and \( f \) polygonal faces
- Which polygonal region contains the query point \( Q \)?
- Let’s slice the plane into vertical “slabs”
- Draw a vertical line through every point

*Computational Geometry Algorithms and Applications, de Berg, Cheong, van Kreveld and Overmars, Chapter 6*
Point Location in Planar Subdivision

Let’s assume “General Position”:

- No two points have same x coordinate
- There will be no vertical segments!
- The query point will not be on a vertical segment or on a vertex.
- \textit{Workaround is to have a tie breaker, rotate/shear the diagram a tiny amount}
Point Location in a Vertical Slab?

- Within this slab, the line segments:
  - Do not cross
    (guaranteed by planar subdivision construction)
  - Do not start or stop
    (we’ve split at every vertex)
- We can sort the line segments vertically
  (by left endpoint’s y coordinate)
- Which trapezoid is Q located within?
  - Each trapezoid is mapped back
to the original polygonal face
Is Query Point above (or below) Line Segment?

- $P_1_x < Q_x < P_2_x$
- Is $0^\circ < \Theta < 180^\circ$
Cross Product

- If the $\Theta > 0^\circ$ & $\Theta < 180^\circ$, then $\mathbf{a} \times \mathbf{b}$ will be positive in the $z$ axis.
- If the $\Theta > 180^\circ$ & $\Theta < 360^\circ$, then $\mathbf{a} \times \mathbf{b}$ will be negative in the $z$ axis.
- If $\mathbf{a}$ is parallel to $\mathbf{b}$ ($\Theta = 0^\circ$ or $\Theta = 180^\circ$), then $\mathbf{a} \times \mathbf{b}$ will have zero magnitude.
- $| \mathbf{a} \times \mathbf{b} | = \sin \Theta$

\[
\mathbf{a} \times \mathbf{b} = \begin{vmatrix}
\mathbf{i} & \mathbf{j} & \mathbf{k} \\
 a_1 & a_2 & a_3 \\
b_1 & b_2 & b_3
\end{vmatrix}
= (a_2 b_3 - a_3 b_2)\mathbf{i} - (a_1 b_3 - a_3 b_1)\mathbf{j} + (a_1 b_2 - a_2 b_1)\mathbf{k}
\]
Analysis: Running Time

- Algorithm Preprocess

- Point Location Algorithm
Analysis: Running Time

- Algorithm Preprocess
  - Sort slabs left to right
  - Within each slab, sort trapezoids from top to bottom

- Point Location Algorithm
  - Binary search to locate the correct slab between two points
    - Left vertical \( x < Q_x < \) right vertical \( x \)
  - Binary search to locate correct trapezoid
    - Q is below the upper segment and above the lower segment
Analysis: Running Time

- Algorithm Preprocess
  - Sort slabs left to right → $O(n \log n)$
  - Within each slab, sort trapezoids from top to bottom → $O(n \log n)$
- Point Location Algorithm
  - Overall: → $O(\log n)$
    - Binary search to locate the correct slab between two points
      - Left vertical $x < Q_x < $ right vertical $x$ → $O(\log n)$
    - Binary search to locate correct trapezoid
      - Q is below the upper segment and above the lower segment → $O(\log n)$
Analysis: Memory Usage

- Unfortunately, this representation is very costly.
- It redundantly storing every many faces in many slabs.
- In the worst case:

\[ \frac{n}{4} \text{ slabs} \]

*Computational Geometry Algorithms and Applications*, de Berg, Cheong, van Kreveld and Overmars, Chapter 6
Analysis: Memory Usage

- Unfortunately, this representation is very costly.
- It redundantly storing every many faces in many slabs.
- In the worst case:
  - Every polygon appears in nearly every slab! → $O(n^2)$
- Even average/expected case is unacceptable: → $O(n \sqrt{n})$
Outline for Today

- Homework 3 Questions?
- Last Time: kD Trees & Range Trees
- Motivating Application: Point Location
- Motivating Application: 2D/3D Mouse “Picking” for Graphics
- Brute Force Point Location
- Point Location by Vertical Slab
  - Trapezoidal Map & Adjacency Structure
  - Trapezoidal Map Analysis & Construction
  - Think-Outside-of-the-Box Graphics Picking Algorithm
- Next Time:
Idea: Reduce Redundant Storage

- Horizontally merge some of these cells
- Split vertically at every vertex
- But stop splitting when you reach the closest line segment above & below

*Computational Geometry Algorithms and Applications*, de Berg, Cheong, van Kreveld and Overmars, Chapter 6
This defines a planar subdivision of with full coverage of the plane by non-overlapping

- convex trapezoids
- degenerate trapezoids:
  - triangles
Can we connect these triangles and trapezoids with a classic half-edge adjacency data structure?
Can we connect these triangles and trapezoids with a classic half-edge adjacency data structure?

- **No!**

Many of the faces have one or more “T junctions” on their top and/or bottom edges.

- This is NOT ALLOWED with a traditional polygonal planar subdivision.

---

Adjacency Structure

*Computational Geometry Algorithms and Applications*, de Berg, Cheong, van Kreveld and Overmars, Chapter 6
Classic Half-Edge Adjacency Structure

- Each face points to a half edge
- Each vertex points to a half edge
- Each half edge points:
  - Its opposite edge – only 1!
  - Its next edge
  - Its face
  - Its vertex
- A hacked modification would require an array of unknown size to point at all “opposite” edges

*This would be inefficient and an implementation nightmare!*

Computational Geometry Algorithms and Applications, de Berg, Cheong, van Kreveld and Overmars, Chapter 6
Trapezoid Map Adjacency Structure

Instead… each trapezoid (or triangle) points to:

- line segment *top*, makes upper boundary
- line segment *bottom*, makes lower boundary
- vertex *leftp*, defines left vertical boundary
- vertex *rightp*, defines right vertical boundary

*Computational Geometry Algorithms and Applications*, de Berg, Cheong, van Kreveld and Overmars, Chapter 6
Trapezoid Map Adjacency Structure

Instead... each trapezoid (or triangle) points to:

- line segment **top**, makes upper boundary
- line segment **bottom**, makes lower boundary
- vertex **leftp**, defines left vertical boundary
- vertex **rightp**, defines right vertical boundary

Additionally... each trapezoid $\Delta$ may have up to 4 adjacent neighbors (or NULL if they do not exist)

- **upper left neighbor**, shares top and leftp
- **lower left neighbor**, shares bottom and leftp
- **upper right neighbor**, shares top and rightp
- **lower right neighbor**, shares bottom and rightp
Trapezoid Map Adjacency Structure

- Does this new adjacency structure allow us to navigate through the structure more efficiently, faster than a $O(n)$ floodfill for the classic polygon adjacency structure?
Trapezoid Map Adjacency Structure

- Does this new adjacency structure allow us to navigate through the structure more efficiently, faster than a $O(n)$ floodfill for the classic polygon adjacency structure?

- Unfortunately, no…

- But we can build a binary tree (actually a DAG) for this structure to perform these queries!
Outline for Today

- Homework 3 Questions?
- Last Time: kD Trees & Range Trees
- Motivating Application: Point Location
- Motivating Application: 2D/3D Mouse “Picking” for Graphics
- Brute Force Point Location
- Point Location by Vertical Slab
- Trapezoidal Map & Adjacency Structure
- Trapezoidal Map Analysis & Construction
- Think-Outside-of-the-Box Graphics Picking Algorithm
- Next Time:
Directed Acyclic Graph (DAG)

- Intermediate notes are vertices (vertical lines) and line segments
- The leaves are the trapezoidal regions (map back to original polygons)
Directed Acyclic Graph (DAG)

- Intermediate notes are vertices (vertical lines) and line segments
- The leaves are the trapezoidal regions (map back to original polygons)

Computational Geometry Algorithms and Applications, de Berg, Cheong, van Kreveld and Overmars, Chapter 6
Analysis: Directed Acyclic Graph (DAG)

Size of the DAG?

- # of leaves = # of trapezoids
- # of intermediate nodes = # of vertices + # of line segments
- Height of DAG

Computational Geometry Algorithms and Applications, de Berg, Cheong, van Kreveld and Overmars, Chapter 6
Analysis: Directed Acyclic Graph (DAG)

Size of the DAG?

- # of leaves = # of trapezoids
  \[ \rightarrow O(n) \]
- # of intermediate nodes
  = # of vertices + # of line segments
  \[ \rightarrow O(n) \]
- Height of DAG
  \[ \rightarrow O(\log n) \text{ best case} \]
  \[ \rightarrow O(n) \text{ worst case} \]
- Use Randomized Incremental Construction to achieve height
  \[ \rightarrow O(\log n) \text{ expected case!} \]
Randomized Incremental Construction

- Randomize the order of the line segments
- Inserting the segments one at a time
- Handle all of the cases

Book has lengthy description of the full algorithm & proof!

Computational Geometry Algorithms and Applications, de Berg, Cheong, van Kreveld and Overmars, Chapter 6
Analysis: Directed Acyclic Graph (DAG)

- Height of the DAG?
  $\rightarrow O(\log n)$ expected

- Query time to locate the trapezoid/polygon containing point Q?
  $\rightarrow O(\log n)$ expected

- Cost to construct?
  $\rightarrow O(n \log n)$ expected

Book has lengthy description of the full algorithm & proof!

Same runtime as vertical slabs!
Linear memory usage!
Outline for Today

● Homework 3 Questions?
● Last Time: kD Trees & Range Trees
● Motivating Application: Point Location
● Motivating Application: 2D/3D Mouse “Picking” for Graphics
● Brute Force Point Location
● Point Location by Vertical Slab
● Trapezoidal Map & Adjacency Structure
● Trapezoidal Map Analysis & Construction
● Think-Outside-of-the-Box Graphics Picking Algorithm
● Next Time:
"Picking" by the Framebuffer

- Graphics “Hack”
- Take advantage of fast GPU hardware rendering
- Color each object a different, unique color (no lighting/shading)
- Grab the color of the pixel from the framebuffer (object id)
- Grab the z-value (depth) from the depth buffer

"Capturing and Animating Occluded Cloth" White, Crane, & Forsyth, SIGGRAPH 2007
“Picking” by the Framebuffer

- Are there enough colors?
- Screen Resolution

"Capturing and Animating Occluded Cloth" White, Crane, & Forsyth, SIGGRAPH 2007
“Picking” by the Framebuffer

- Are there enough colors?
  - 3 colors (RGB)
    - w/ 8 bits each
  - \(2^8 \times 2^8 \times 2^8 = 2^{24} = 16\) million

- Screen Resolution
  - “4k” = 4096 x 2160
    - = 9 million pixels
  - “8k” = 7680 x 4320
    - = 33 million pixels

“Capturing and Animating Occluded Cloth”
White, Crane, & Forsyth, SIGGRAPH 2007
Painting by Picking a Picket Fence?

2D → 3D & Usability:

- You “click” on a picket to start painting
- Move up and down, you stay on the picket
- Move left or right, you fall between the pickets.
  - Does you hover in the air between pickets?
  - Does your mouse z coordinate change?
  - Do you start painting the ground?

https://www.fencenashville.net/
Outline for Today

- Homework 3 Questions?
- Last Time: kD Trees & Range Trees
- Motivating Application: Point Location
- Motivating Application: 2D/3D Mouse “Picking” for Graphics
- Brute Force Point Location
- Point Location by Vertical Slab
- Trapezoidal Map & Adjacency Structure
- Trapezoidal Map Analysis & Construction
- Think-Outside-of-the-Box Graphics Picking Algorithm
- Next Time: