CS 164 & CS 266: Computational Geometry

Lecture 7

Mesh generation, quadtrees, and continuous Dijkstra

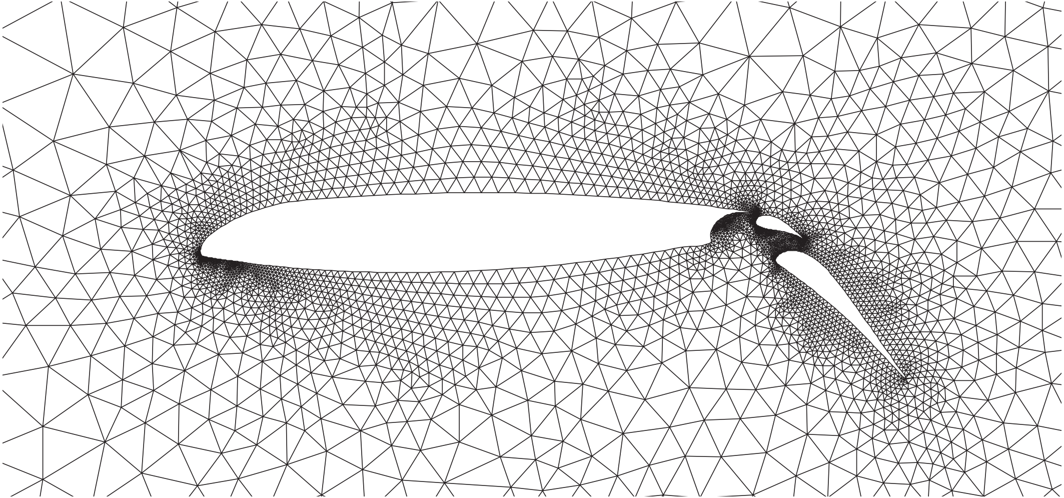
David Eppstein
University of California, Irvine

Fall Quarter, 2023

Main idea

What is a mesh?

Input: a 2d or 3d region in which we want to simulate airflow, heat, strain, or other physical properties



Mesh: subdivision into simple shapes such as triangles: “elements”

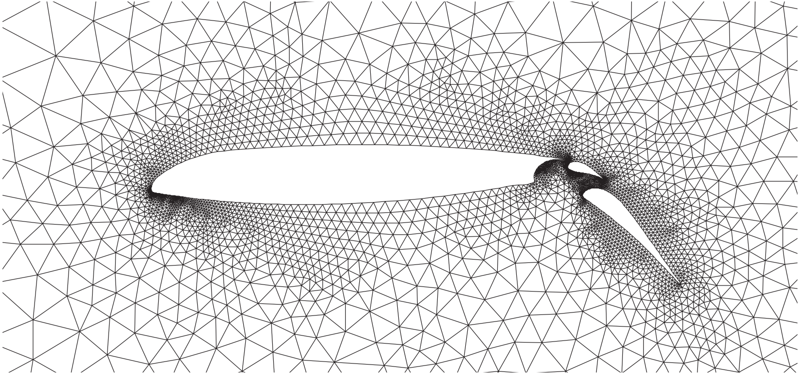
Finite element analysis

Once we have a mesh, solve a big system of linear equations:

Variables: air velocity and density within each triangle

Boundary conditions: constant velocity and density far from wing

Equations: Relate flows and densities between neighboring triangles (e.g. total air in must equal total air out)



Solution: Steady state flow

How does the mesh affect this analysis?

Number of triangles $\Rightarrow$ size of system of equations

Fewer triangles: faster

Size of elements compared to size of features of input shape and solution flow $\Rightarrow$ accuracy of simulation

Smaller triangles: more accurate

Shape of elements $\Rightarrow$ "stiffness" of system of equations
$\Rightarrow$ speed and accuracy of iterative numerical methods

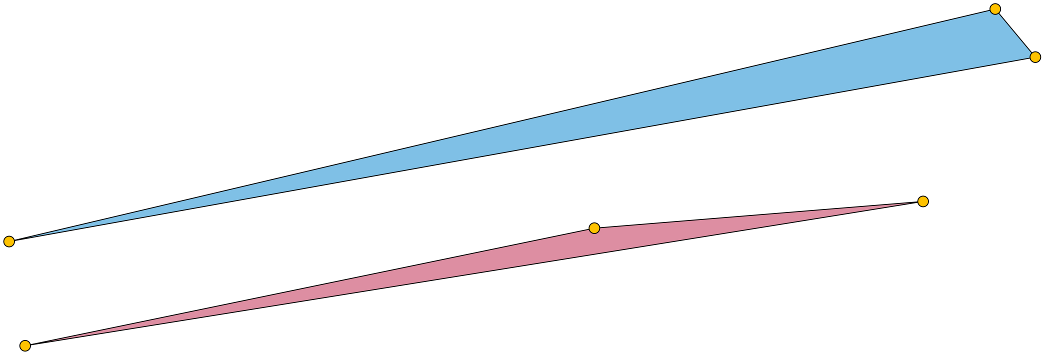
Less-sharp triangles: easier to solve

Precise relation of shape to stiffness not well understood

What shapes should we aim for?

Possibility 1: Avoid sharp (near-zero) angles

Possibility 2: Sharp ok, but avoid wide (near-π) angles

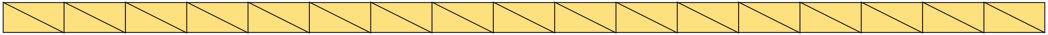


Possibility 3: Whether a shape is good or bad depends on the solution values near it, and not on the shape itself

Lower bounds on numbers of triangles

Simple shapes may require many triangles

If we forbid sharp angles ($< \varepsilon$ for some $\varepsilon > 0$)
then $1 \times x$ rectangle requires $\Omega(x)$ triangles
even though $n = 4 = O(1)$



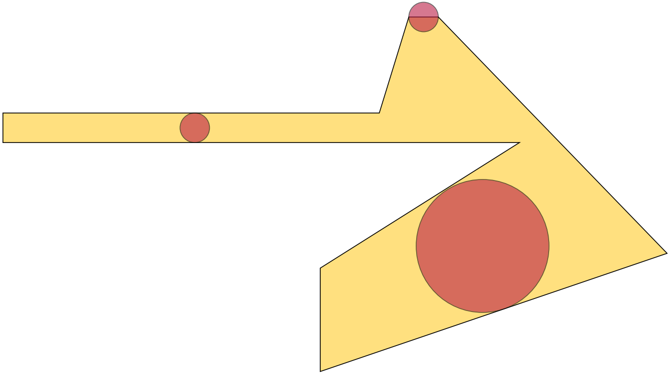
Corollary: Number of triangles cannot be a function only of n

and we cannot get a time bound depending only on n

[Bern et al. 1990]

Local feature size

Radius of smallest circle, centered at a given point, that intersects two non-touching polygon edges

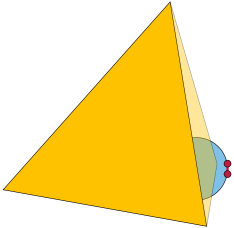


Area vs local feature size

Claim: In a triangle mesh with no sharp angles (all angles $> \varepsilon$), each point in a triangle of area A has local feature size $\geq$ constant $\cdot \sqrt{A}$

Ideas of proof:

- No sharp angles $\Rightarrow$ all sides of triangle have length proportional to $\sqrt{A}$
- If any point in the triangle is near two disjoint features $\Rightarrow$ closest boundary point of the triangle is near the same two features
- The next triangle across that edge must stay inside the polygon, forcing it to have a sharp angle



Lower bound on number of triangles

For a domain (polygon) D, In a triangle mesh of D with no sharp angles, let N be the number of triangles in the mesh, and let $a(p)$ be the area of the triangle containing any point p.

Then the integral of the constant function $1/\text{area}$ over a single triangle is one, and combining with the inequality of area versus local feature size gives:

$$N = \int_D \frac{1}{a(p)} dx\, dy \geq \text{constant} \cdot \int_D \frac{1}{\text{lfs}(p)^2} dx\, dy.$$

Idea: If we can make every triangle have area $\geq$ (local feature size)^2, the reverse of the inequality, both integrals will be within a constant factor of each author $\Rightarrow$ number of triangles will be near-optimal

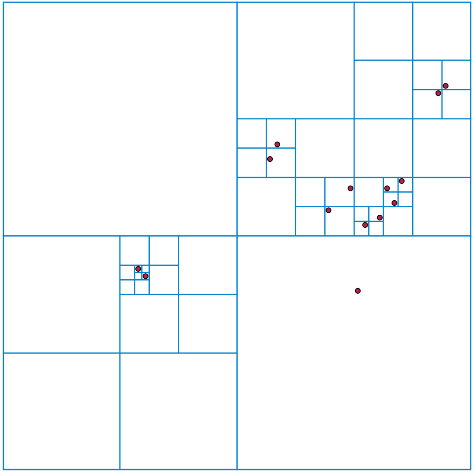
[Ruppert 1993]

Quadtree-based meshing

Quadtree

(More precisely, "point quadtree"; there are other kinds)

Recursively divide squares into four smaller squares



Simplifying assumptions:

- Point coordinates are integers in range $0 \ldots 2^b - 1$ for some b
- Squares have side lengths 2^k for $0 \le k \le b$
- Coordinates of square sides are integer $+ \frac{1}{2}$ so points avoid square sides

Representation and construction

Each square stores:

- Its square
- Whether it is empty, has one point, or has multiple points
- If one point, what is that point?
- If multiple points, four child squares

Start with a big power-of-two-size square containing all of the points, and a list of all its points

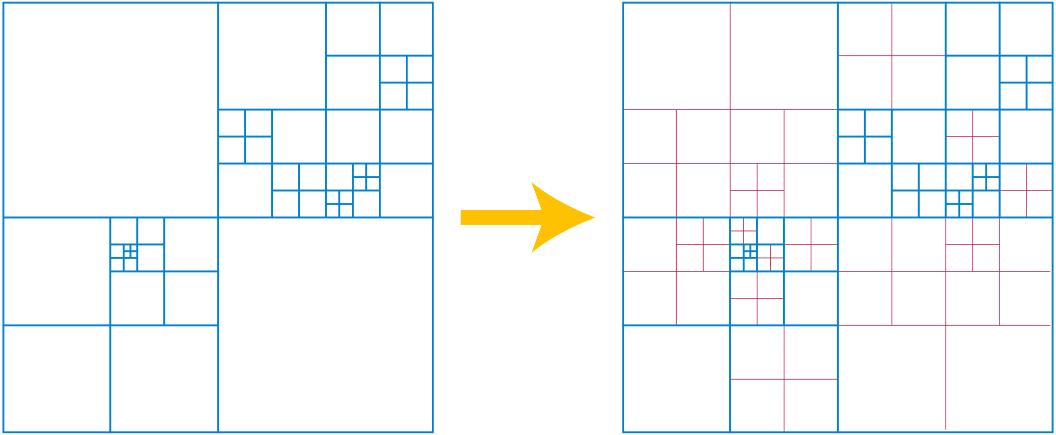
Test whether list is empty, one point, or more than one

If more than one, partition points into four quadrants and recursively construct four child squares

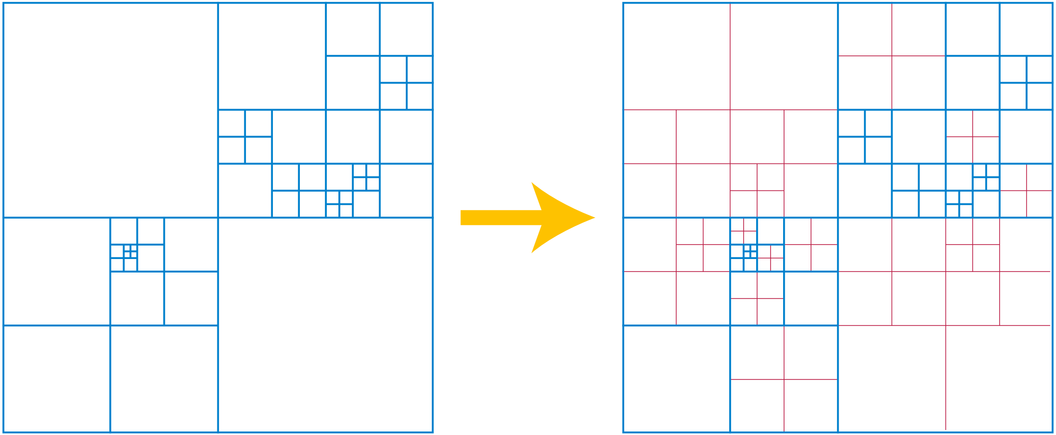
Balanced quadtree

Construct a quadtree normally (recursively split overfull squares)

Then, while any square has neighboring squares $< \frac{1}{2}$ its size, split it



Balanced quadtree



A split square of side s has distance $\leq 2s$ to a smaller square of the original quadtree

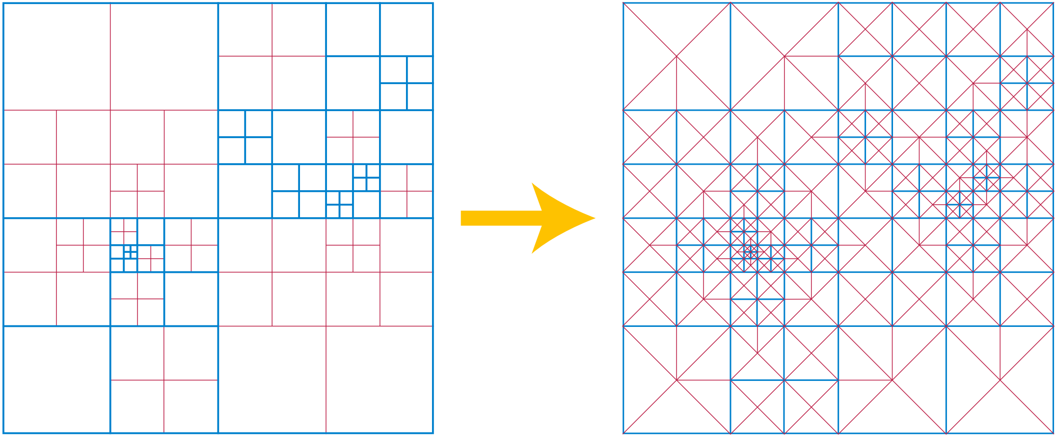
Each square of unbalanced quadtree $\Rightarrow$ $O(1)$ squares of balanced quadtree

Each square has side length proportional to local feature size

Triangulating a balanced quadtree

Add a vertex at the center of each square

Connect it to the vertices on the boundary of the square

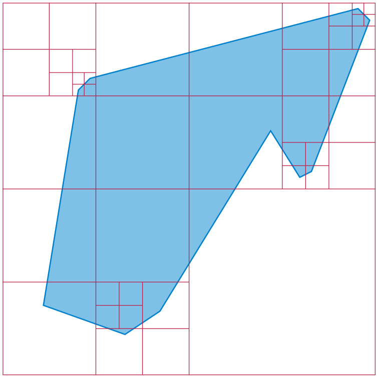


All triangles are isosceles right triangles, angles 45° and 90°

Quadtree-based meshing

- Surround whole polygon in a bounding square
- Recursively subdivide squares crossed by non-touching edges

- More subdivision + messy case analysis for squares crossed by boundary
- Balance
- Triangulate empty squares



Quadtree mesh analysis

Mary Poppins:
"practically perfect in every way"

All angles bounded away from 0 and 180°

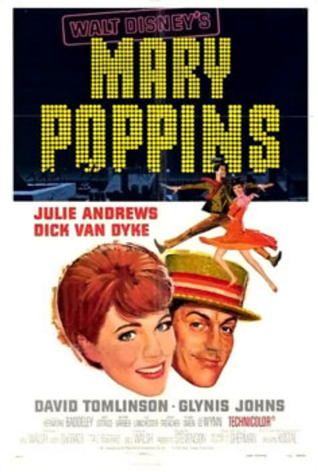
(except for sharp angles of input polygon)

All triangles have diameter proportional to local feature size ⇒ # triangles is within a constant factor of optimal (by the argument involving the integral of $1/\text{lfs}^2$)

Similar argument with the integral of $1/\text{lfs}$ shows total edge length is within a constant factor of optimal

Construction takes time linear in mesh size

[Bern et al. 1990]



Non-obtuse triangulation of polygons

Right angles are special

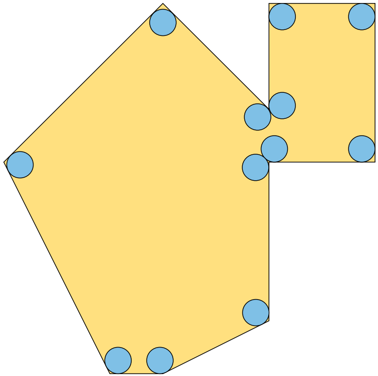
We already saw that if we want all angles $\geq \varepsilon$,
$\#$ triangles depends on geometry not just on n

But if we try to get all angles $\leq 90^\circ - \varepsilon$, we also get all angles $\geq 2\varepsilon$

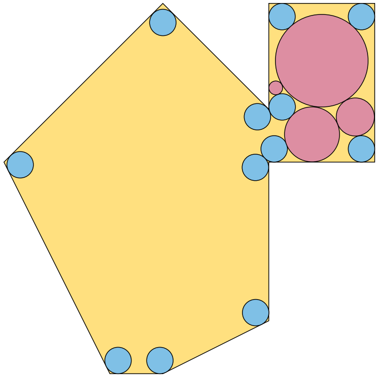
$\Rightarrow$ if we want $O(n)$ triangles, 90° is the best we can hope for

For any polygon, it is possible to find a mesh with $O(n)$ triangles, all angles $\leq 90^\circ$!
Main idea: Pack circles into the polygon and use them to guide the mesh

Packing circles into a polygon



Protect vertices



Split regions with > 4 sides

After doing this, we are left with $O(n)$ circles and $O(n)$ four-sided regions

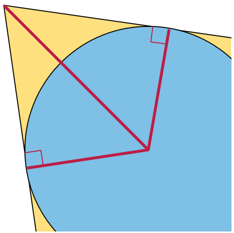
(Side of a region can be either part of a polygon edge, or an arc of a circle.)

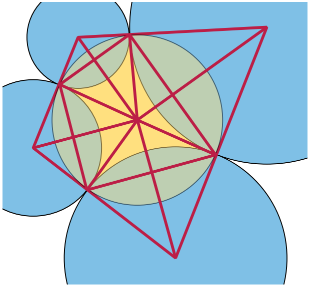
Non-obtuse triangulation

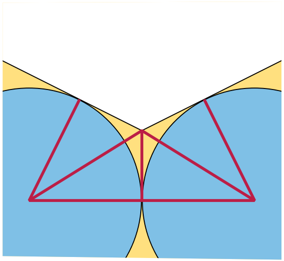
Add radii from centers to points of tangency

Messy case analysis: All $\leq$ 4-sided regions can be triangulated

(in such a way that the triangles from adjacent regions meet edge-to-edge)



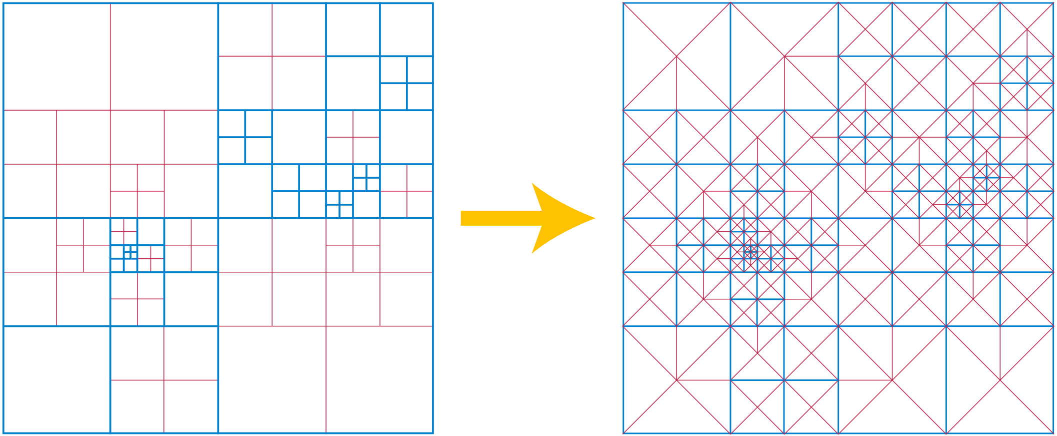




Result: non-obtuse triangulation, $O(n)$ triangles [Bern et al. 1995]

Compressed quadtrees and non-obtuse triangulation of point sets

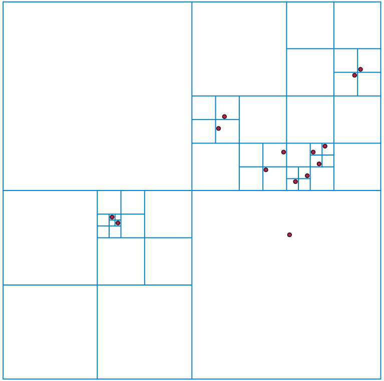
Triangulating a point quadtree



Already gives us a non-obtuse triangulation

(Not shown: case analysis for triangulating one point somewhere in the middle of a quadtree square.)

Problem: Quadtree can be much bigger than linear

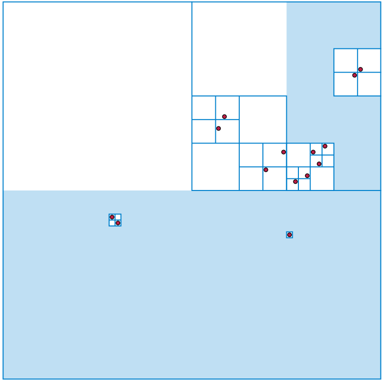


Construction can perform many levels of recursion without splitting anything

(like on lower left of example)

So # triangles is not linear!

Solution: Compressed quadtree



When constructing a quadtree node, shrink its square to smallest power-of-two square containing its points

Every non-leaf square has more than one non-empty child

Total # squares is $O(n)$

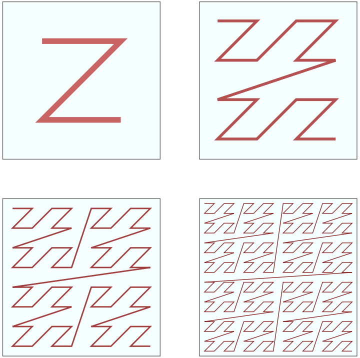
Fast construction of compressed quadtree

Sort by “Z-curve”: recursively traverse quadtree northwest-northeast-southwest-southeast

Same as shuffling the bits of the x and y coordinates into a single $2b$-bit binary number and sorting by that number

All compressed quadtree squares are minimum power-of-two squares around two consecutive points

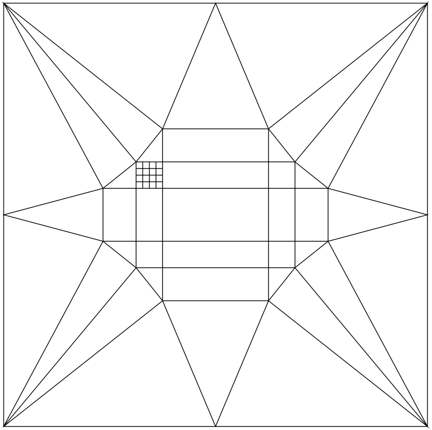
Total time $O(n \log n)$

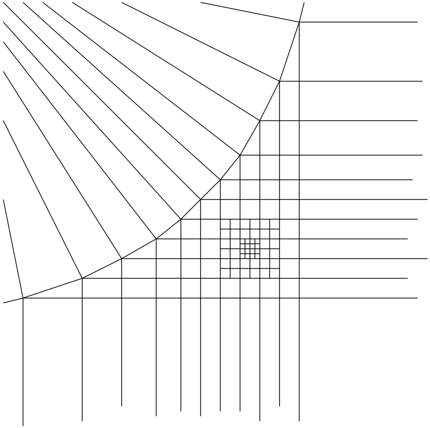


[Bern et al. 1999]

Meshing with compressed quadtrees

When a quadtree square has only one (much smaller) child),
carefully connect its outer boundary to the child





Can guarantee: $O(n)$ triangles, all triangle angles $\leq 90^\circ$

[Bern et al. 1990]

Shortest paths revisited

$O(n \log n)$ time shortest path algorithm

"Continuous Dijkstra": simulate a wave expanding from start point

Wavefront = sequence of circular arcs

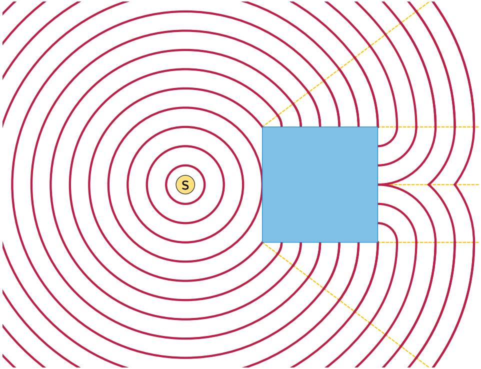
Use compressed quadtree squares as a mesh (don't triangulate it!)

Use an approximate version of the wavefront, accurate to mesh cell size, and fix up the approximation later

Key property: $O(1)$ mesh cells within distance $2 \times$ length of any mesh edge

Events: wavefront interacts w/ mesh

$O(n \log n)$ [Hershberger and Suri 1999]



References

M. Bern, D. Eppstein, and S.-H. Teng. Parallel construction of quadtrees and quality triangulations. *Int. J. Comput. Geom. Appl.*, 9(6):517–532, 1999. doi: 10.1142/S0218195999000303.

Marshall Bern, Scott Mitchell, and Jim Ruppert. Linear-size nonobtuse triangulation of polygons. *Discrete & Computational Geometry*, 14(4):411–428, 1995. doi: 10.1007/BF02570715.

Marshall W. Bern, David Eppstein, and John R. Gilbert. Provably good mesh generation. In *31st Annual Symposium on Foundations of Computer Science, St. Louis, Missouri, USA, October 22-24, 1990, Volume I*, pages 231–241. IEEE Computer Society, 1990. doi: 10.1109/FSCS.1990.89542.

John Hershberger and Subhash Suri. An optimal algorithm for Euclidean shortest paths in the plane. *SIAM Journal on Computing*, 28(6):2215–2256, 1999. doi: 10.1137/S0097539795289604.

Jim Ruppert. A new and simple algorithm for quality 2-dimensional mesh generation. In Vijaya Ramachandran, editor, *Proceedings of the Fourth Annual ACM/SIAM Symposium on Discrete Algorithms, 25-27 January 1993, Austin, Texas, USA*, pages 83–92, 1993.