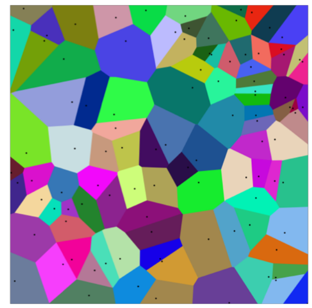
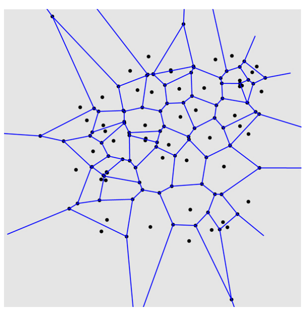
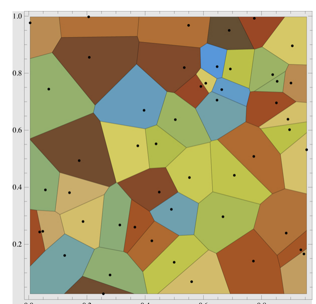
Computational Geometry

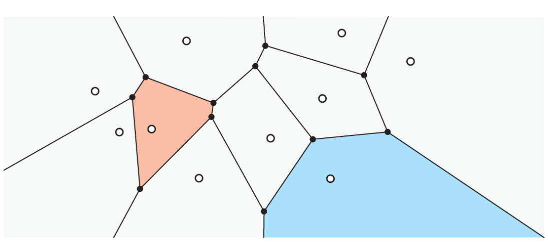
Voronoi Diagrams







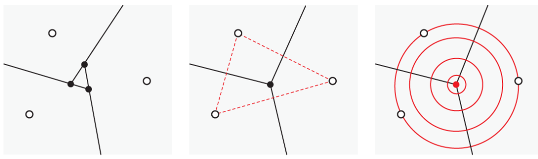
Voronoi Diagram of 10 Sites



Definition

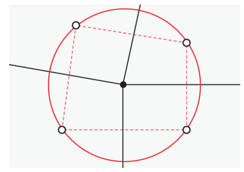
- The Voronoi region *Vor*(*p*) of a site *p* in *S* is $Vor(p) = \{x \in R^2 \mid |x\text{-}p| \le |x\text{-}q|, \forall q \in S\}$, where $|x\text{-}y|$ denotes the distance between two points *x* and *y*.
- The Voronoi region of *p* consists of all points that are at least as close to *p* as any other sites in *S*.
- Vor(p) is the intersection of all halfplanes H(p, q), where q is any other sites in S.

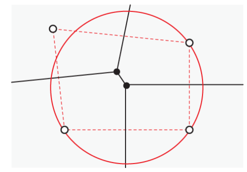
3 Sites



(a) (b) (c)

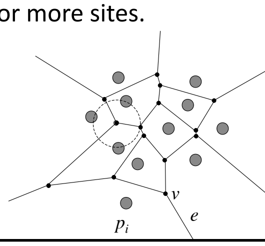
4 or More Sites





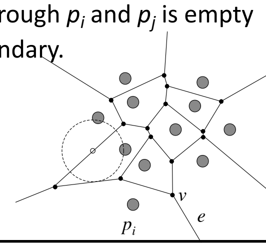
Voronoi Vertex

- Let S be a point set with Voronoi diagram $Vor(S)$. A point v is a Voronoi vertex of $Vor(S)$ if and only if there exists an empty circumcircle centered at v of three or more sites.



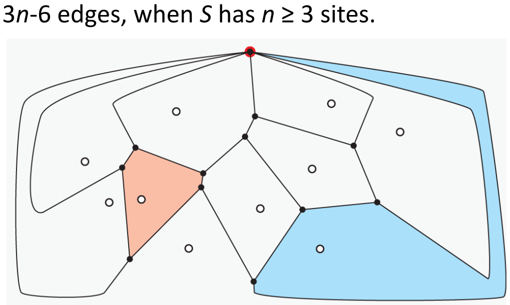
Voronoi Edge

- Let e be a connected subset of the bisector between sites p_i and p_j of S. e is a Voronoi edge of $Vor(S)$ iff for each point x in e, the circle centered at x through p_i and p_j is empty in its interior and boundary.



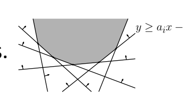
Combinatorics of Voronoi

- $Vor(S)$ has at most $2n$-5 Voronoi vertices and $3n$-6 edges, when S has $n \geq 3$ sites.

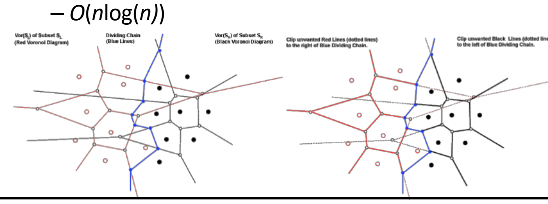


Voronoi Algorithms

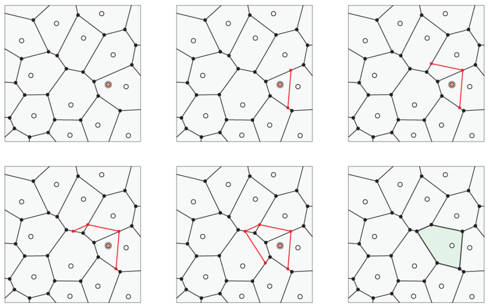
- Naïve Voronoi
  - Intersection of n-1 half planes.
  - Complexity?



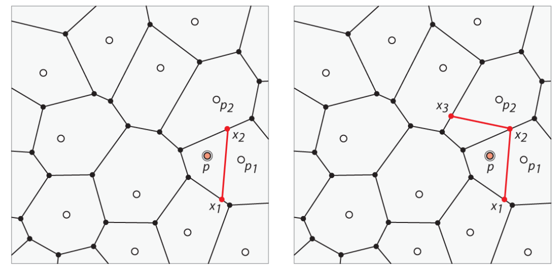
- Divide-and-Conquer (Shamos & Hoey 1975)
  - $O(n\log(n))$



Incremental



Incremental - Details



Incremental

- Youtube video
- Applet

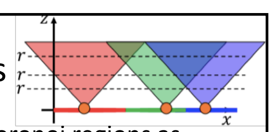
Dynamic Voronoi

- Emergent Voronoi
- Particles
- Swarm
- Personal space
- Voronoi and Delaunay

Fortune's

- desmos(HTML5)
- YouTube video

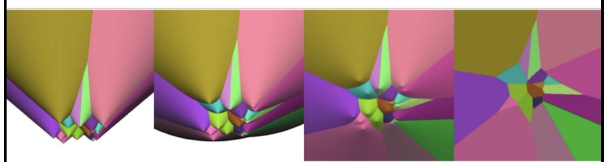
Voronoi and Cones



- We think of generating Voronoi regions as expanding circles centered on the sites
- When multiple circles overlap a point, track the one that is closer
- Visualize the Voronoi region by drawing cones along the positive Z
- Cones with radius r are the projections of the intersections of the plane $Z = r$ with the cones onto the xy-plane

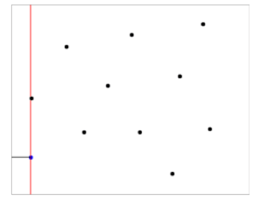
xy Projection

- To track the closer cone, we render the cones with an orthographic camera looking up the Z-axis



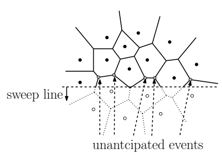
Fortune's

- 1986
- A plane sweep algorithm
- Run time complexity $n\log(n)$
- Beach line



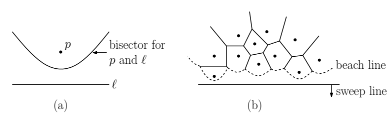
Difficulty

- Portion of Voronoi diagram behind sweep line should be complete
- Sites that lie ahead of the sweep line may generate Voronoi vertices behind the sweep line
- Unanticipated events
- Can finalize points closer to a site than the line



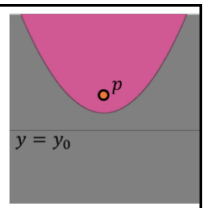
The Beach Line

- Maintain an additional x-monotone curve
- Lags behind the sweep line
- Bisector between a point and a line



- Guaranteed unaffected by unswept sites
- Portion of Voronoi above the beach line is safe

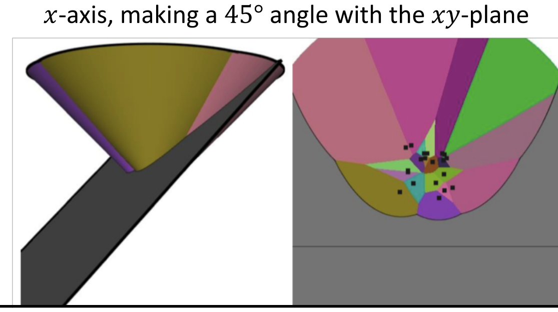
Geometric Realizations



- Given a site p and some line $y = y_0$, we can finalize the points satisfying:
$$\{(x,y) \mid \|(x,y) - p\|^2 < (y - y_0)^2\}$$
- points on the parabola satisfy:
$$\|(x,y) - p\|^2 = (y - y_0)^2$$
- $z = \|(x,y) - p\|$ - points on the cone, centered at p
- $z = y - y_0$ - points on an angled plane passing through the line $y = y_0$ at $z = 0$

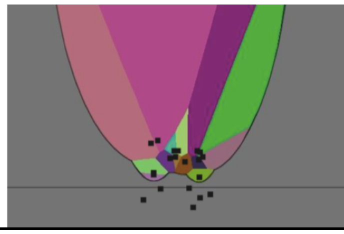
Fortune's

- Sweep the cones with a plane parallel to the x-axis, making a $45°$ angle with the xy-plane



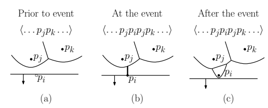
Fortune's

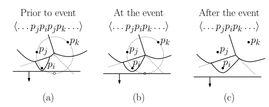
- As the plane π_y advances, the algorithm maintains a set of parabolic fronts (the projection of the intersection of π_y with the cones)



Sweep-line Events

- Site events
  - when the sweep line passes over a new site
  - new parabolic arc inserted
  - known ahead of time
- Voronoi vertex event
  - three sites whose arcs appear consecutively on the beach line
  - when the sweep line passes the circumcircle
  - a new Voronoi vertex





Algorithm Implementation

- Partial Voronoi
  - stored in any reasonable data structure for planar subdivisions (doubly linked edge list)
- Beach line
  - sorted sequence of sites whose arcs (not stored) form the beach line (balanced binary tree)
  - break points (not stored) can be computed as a function of p_i, p_j and the sweep line *y*
- Event queue (priority queue)