Fast Computation of Generalized Voronoi Diagrams Using Graphics Hardware

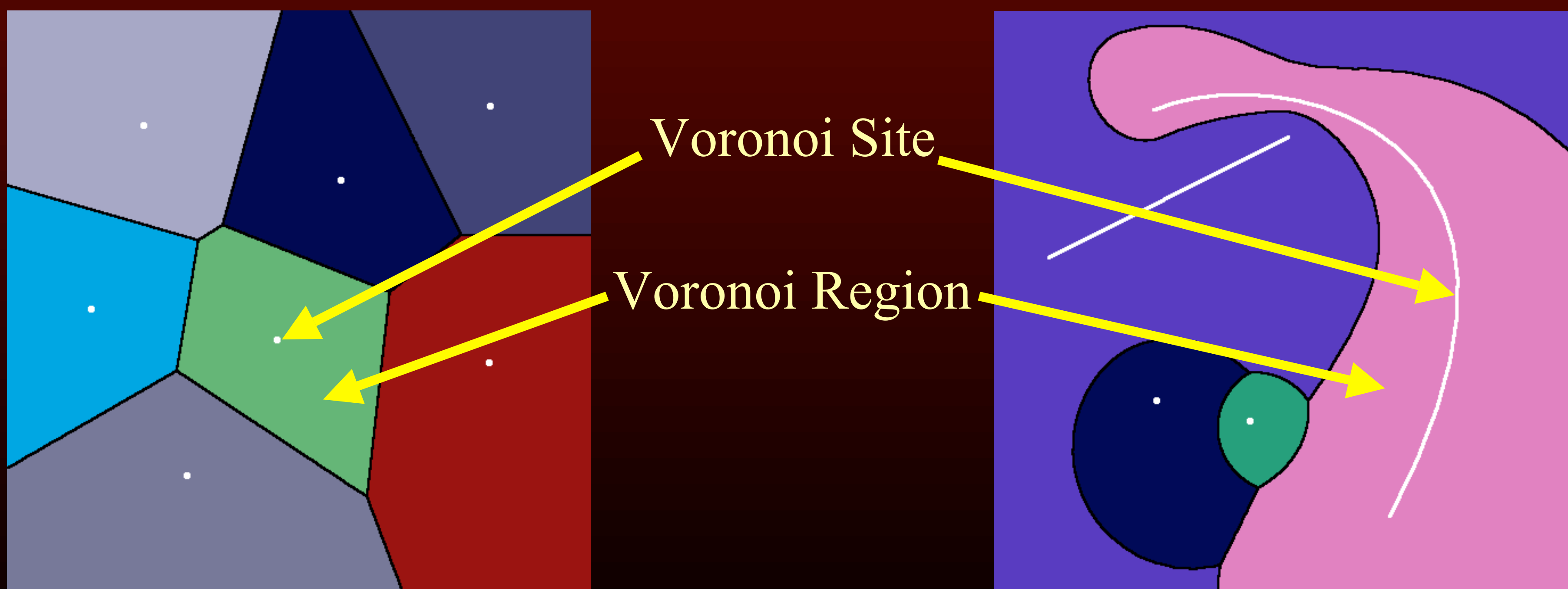
Kenneth E. Hoff III, Tim Culver, John Keyser, Ming Lin, and Dinesh Manocha

University of North Carolina at Chapel Hill

SIGGRAPH '99

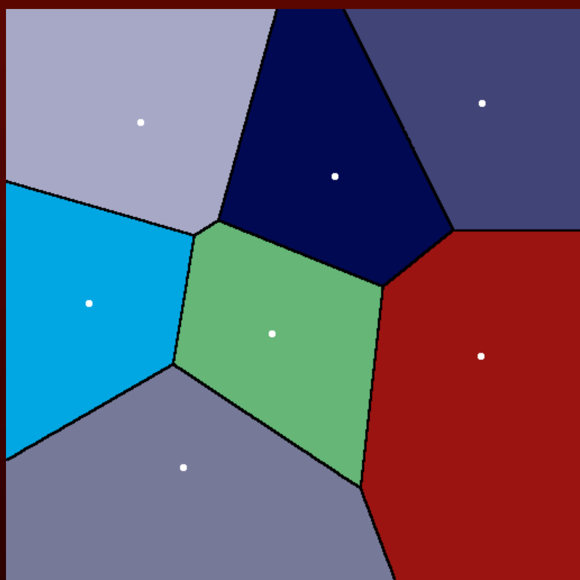
What is a Voronoi Diagram?

Given a collection of geometric primitives, it is a subdivision of space into cells such that all points in a cell are *closer* to one primitive than to any other



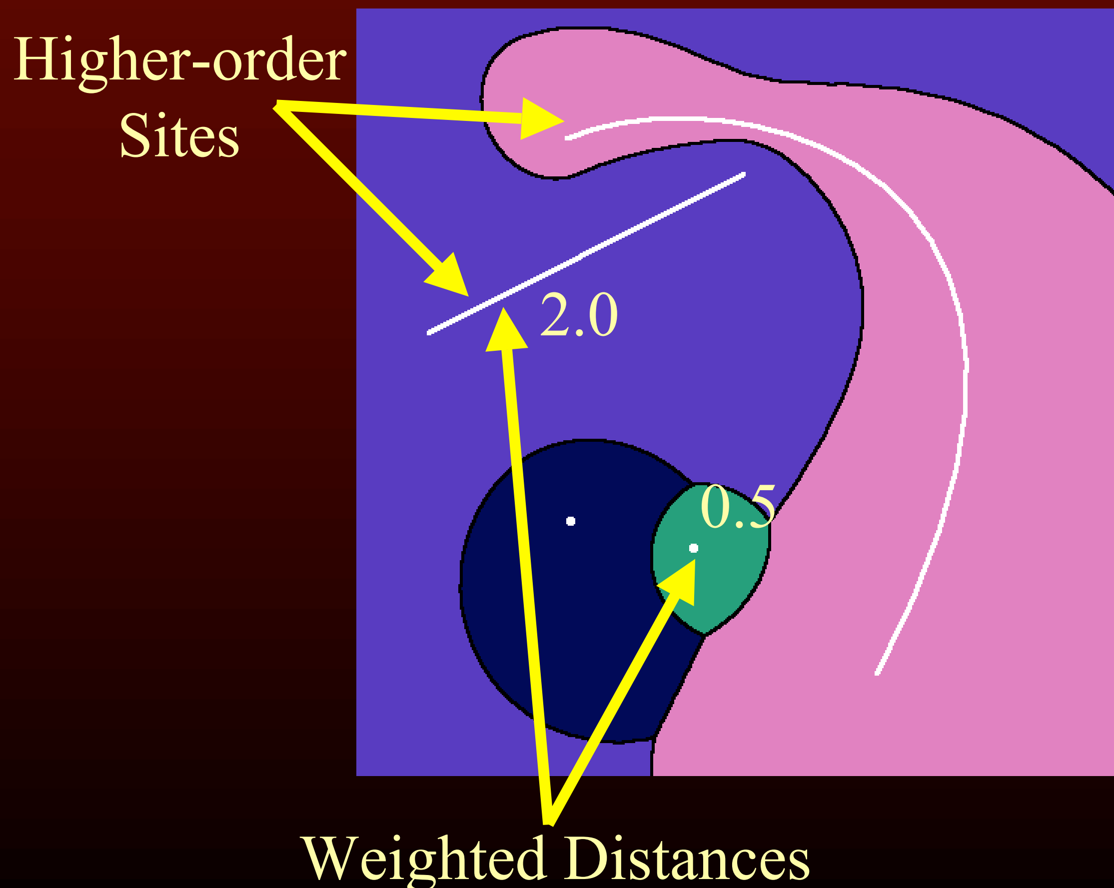
Ordinary

- Point sites
- Nearest Euclidean distance



Generalized

- Higher-order site geometry
- Varying distance metrics



Why Should We Compute Them?

It is a fundamental concept

Descartes	Astronomy	1644	"Heavens"
Dirichlet	Math	1850	Dirichlet tesselation
Voronoi	Math	1908	Voronoi diagram
Boldyrev	Geology	1909	area of influence polygons
Thiessen	Meteorology	1911	Theissen polygons
Niggli	Crystallography	1927	domains of action
Wigner & Seitz	Physics	1933	Wigner-Seitz regions
Frank & Casper	Physics	1958	atom domains
Brown	Ecology	1965	areas potentially available
Mead	Ecology	1966	plant polygons
Hoofd et al.	Anatomy	1985	capillary domains
Icke	Astronomy	1987	Voronoi diagram

Why Should We Compute Them?

Useful in a wide variety of applications

Collision Detection

Surface Reconstruction

Robot Motion Planning

Non-Photorealistic Rendering

Surface Simplification

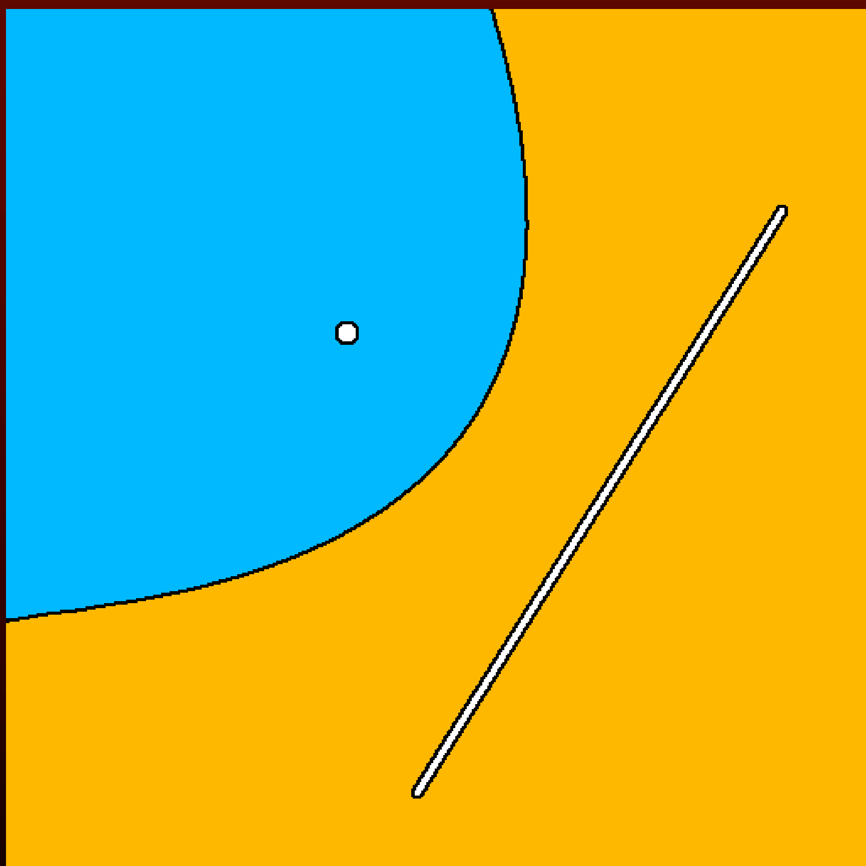
Mesh Generation

Shape Analysis

Outline

- Generalized Voronoi Diagram Computation
  - Exact and Approximate Algorithms
  - Previous Work
  - Our Goal
- Basic Idea
- Our Approach
- Basic Queries
- Applications
- Conclusion

Generalized Voronoi Diagram Computation
“Exact” Algorithms



Computes Analytic Boundary

Previous work

- Lee82
- Chiang92
- Okabe92
- Dutta93
- Milenkovic93
- Hoffmann94
- Sherbrooke95
- Held97
- Culver99

Previous Work: “Exact” Algorithms

- Compute analytic boundaries

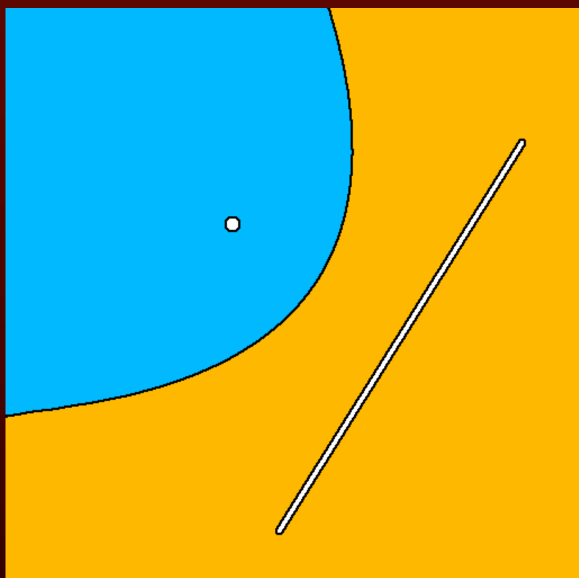
but...

- Boundaries composed of high-degree curves and surfaces and their intersections
- Complex and difficult to implement
- Robustness and accuracy problems

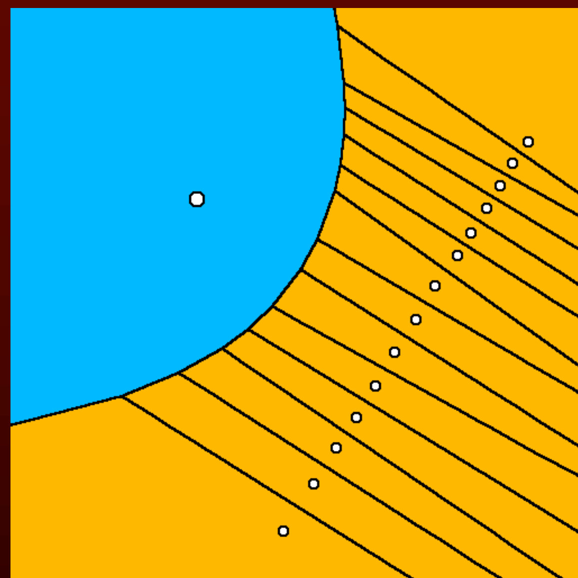
Generalized Voronoi Diagram Computation

Exact Algorithm

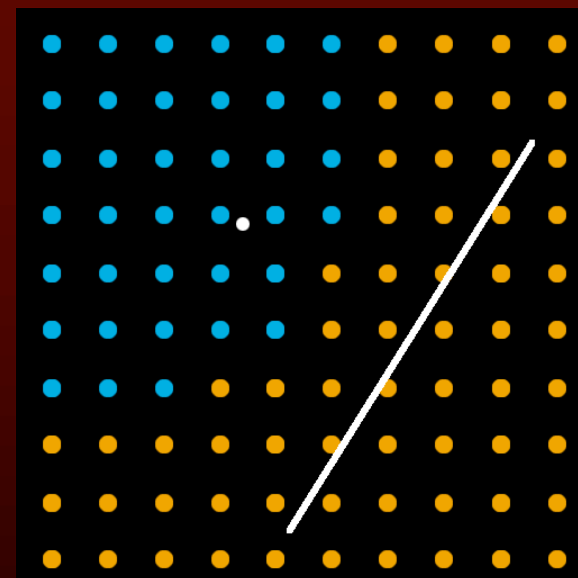
Approximate Algorithms



Analytic Boundary



Discretize Sites



Discretize Space

Previous work

Lavender92, Sheehy95, Vleugels 95 & 96, Teichmann97

Previous Work: Approximate Algorithms

- Provide practical solutions

but...

- Difficult to error-bound
- Restricted to static geometry
- Relatively slow

Our Goal

Approximate generalized Voronoi diagram computation that is:

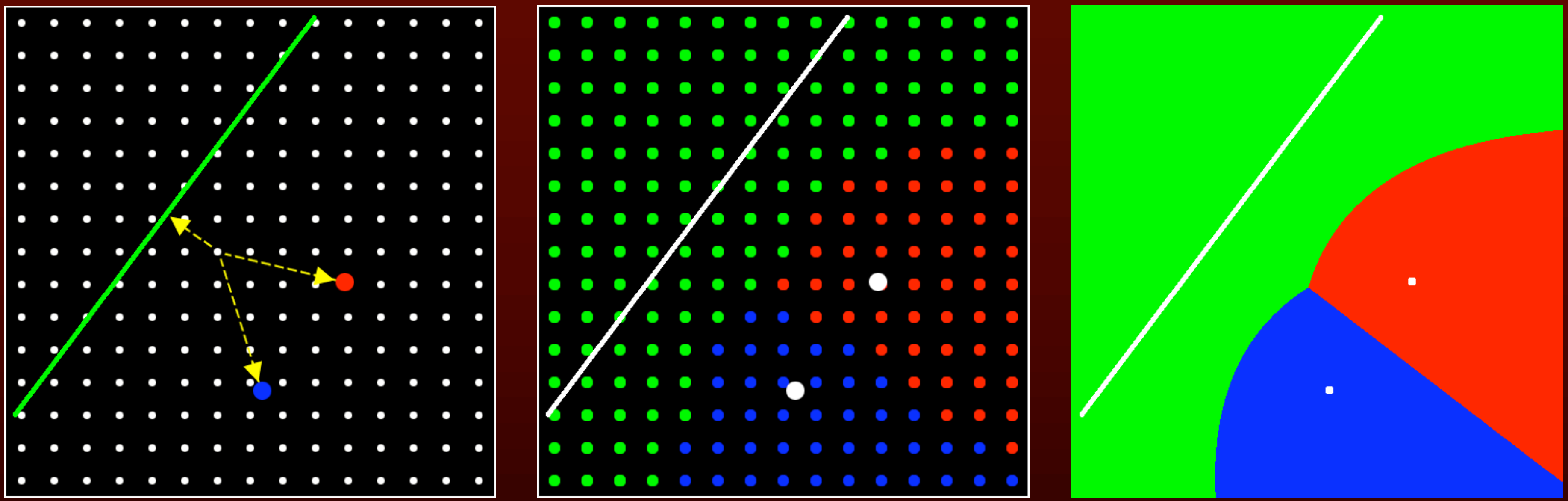
- Simple to understand and implement
- Easily generalized
- Efficient and practical

with all sources of error fully enumerated

Outline

- Generalized Voronoi Diagram Computation
- Basic Idea
  - Brute-force Algorithm
  - Cone Drawing
  - Graphics Hardware Acceleration
- Our Approach
- Basic Queries
- Applications
- Conclusion

Brute-force Algorithm



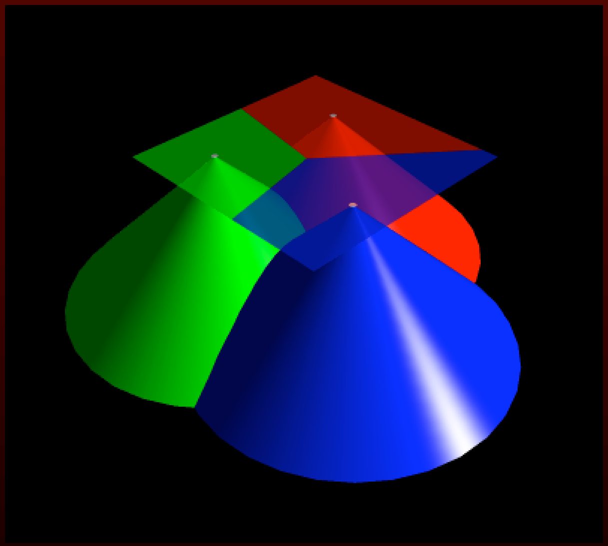
Record ID of the closest site to each sample point

Coarse point-sampling result

Finer point-sampling result

Graphics Hardware Acceleration

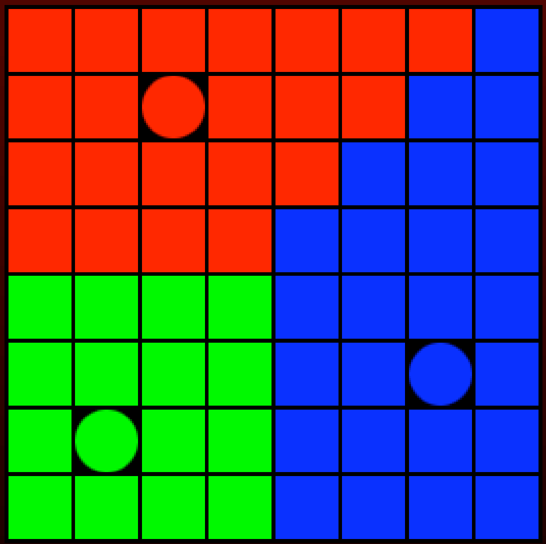
Simply rasterize the cones using graphics hardware



Haeberli90, Woo97

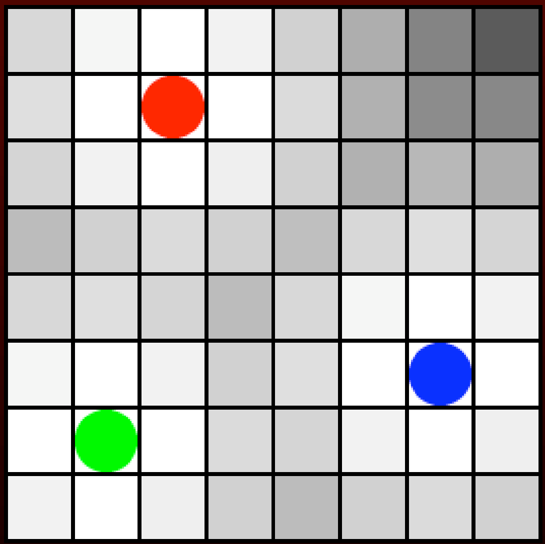
Our 2-part discrete Voronoi diagram representation

Color Buffer



Site IDs

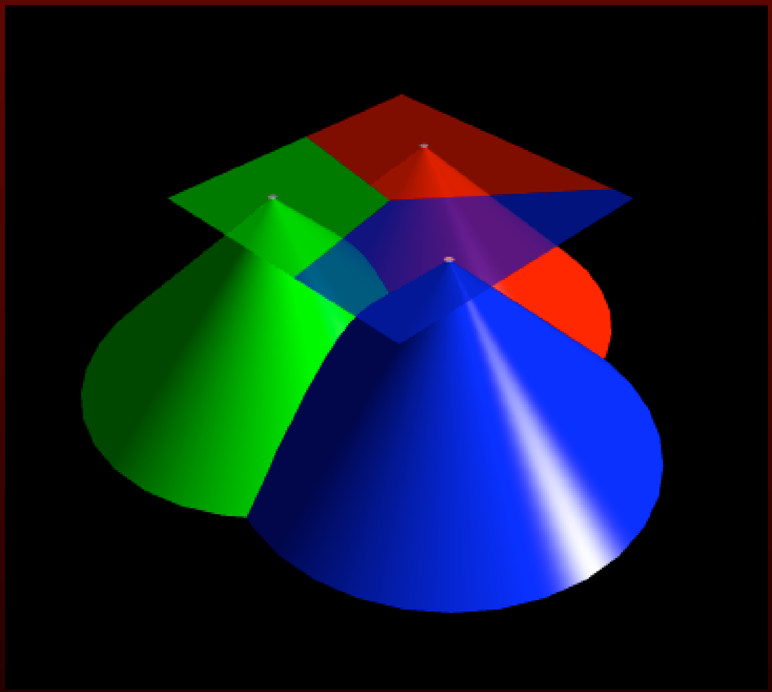
Depth Buffer



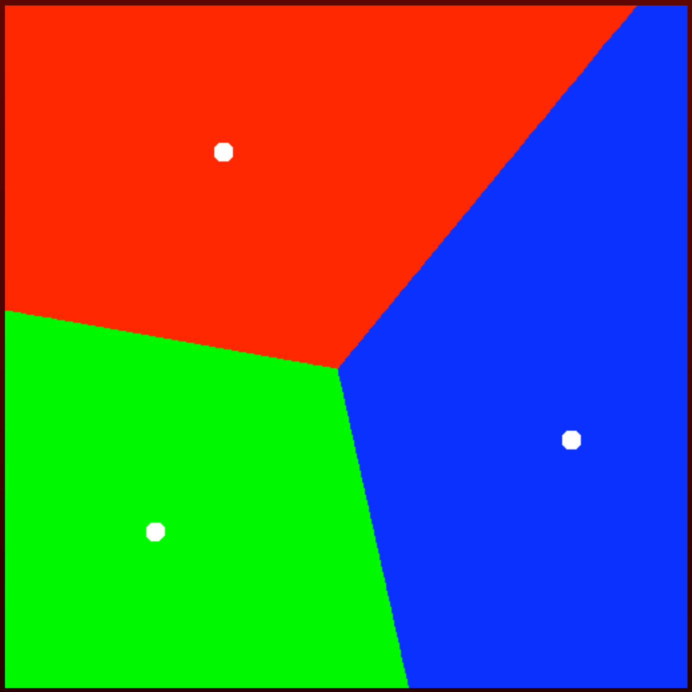
Distance

Cone Drawing

To visualize Voronoi diagram for points in 2D…



Perspective, 3/4 view



Parallel, top view

Dirichlet 1850 & Voronoi 1908

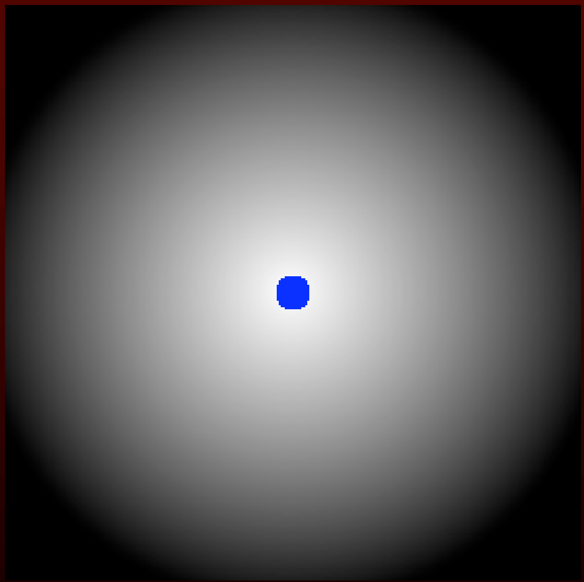
Outline

- Generalized Voronoi Diagram Computation
- Basic Idea
- Our Approach
  - Meshing Distance Function
  - Generalizations
  - 3D
  - Sources of Error
- Basic Queries
- Applications
- Conclusion

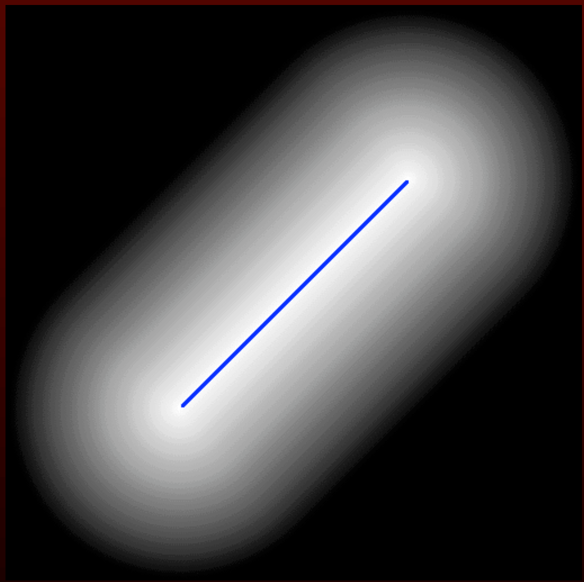
The Distance Function

Evaluate distance at each pixel for all sites

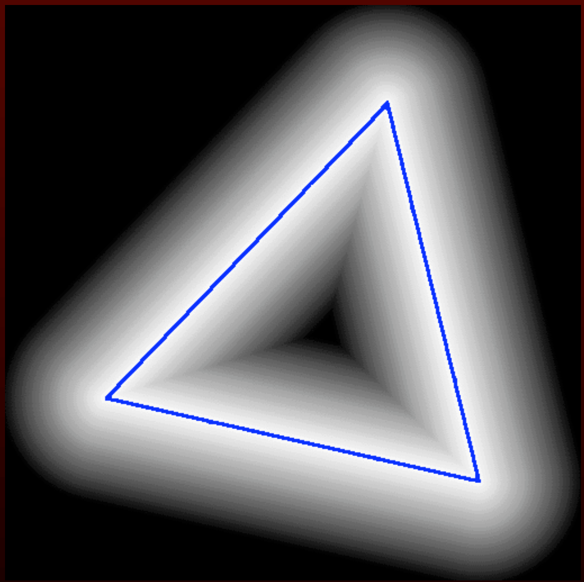
Accelerate using graphics hardware



Point



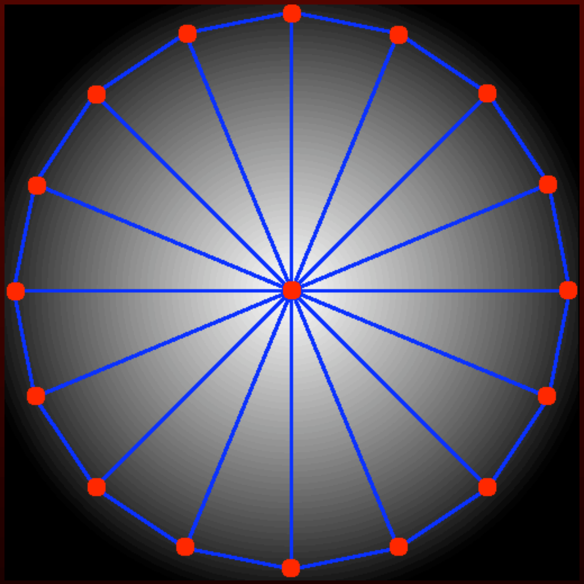
Line



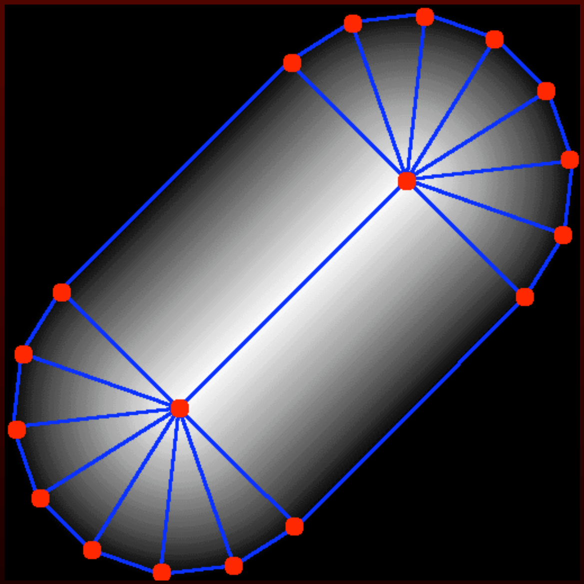
Triangle

Approximating the Distance Function

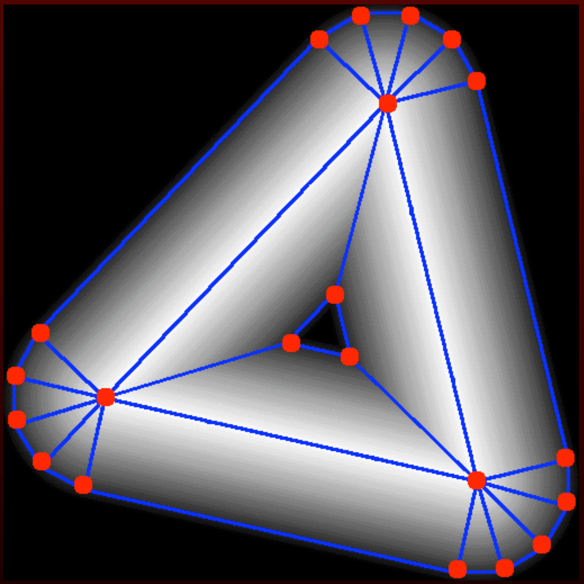
- Avoid per-pixel distance evaluation
- Point-sample the distance function
- Reconstruct by rendering polygonal mesh



Point

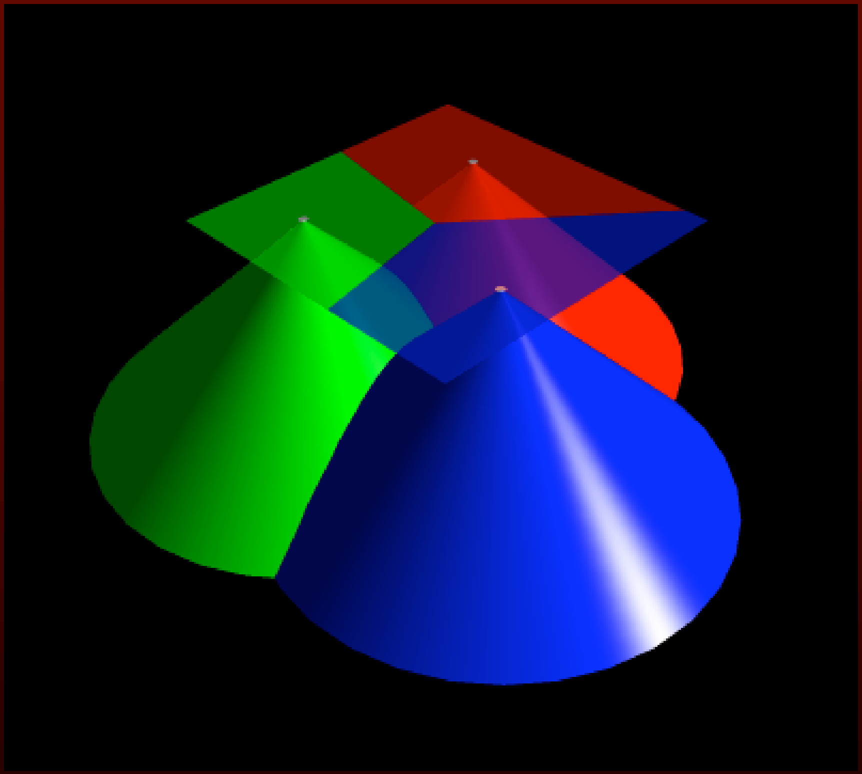


Line

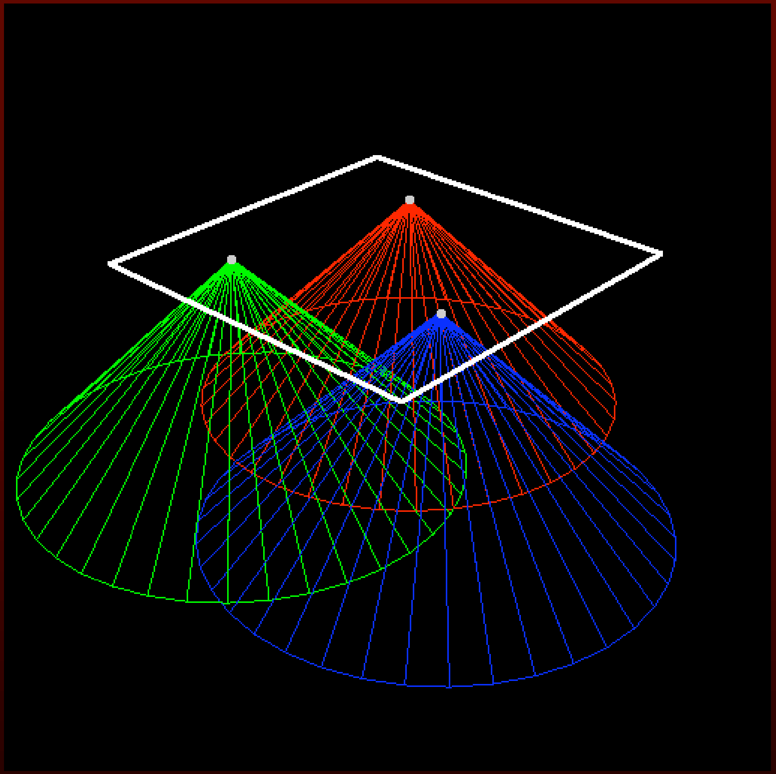


Triangle

Meshing the Distance Function

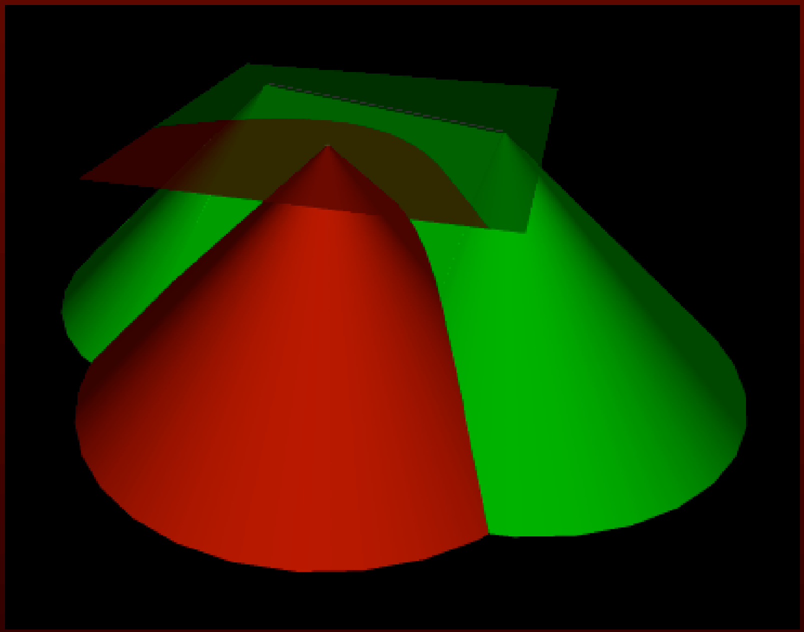


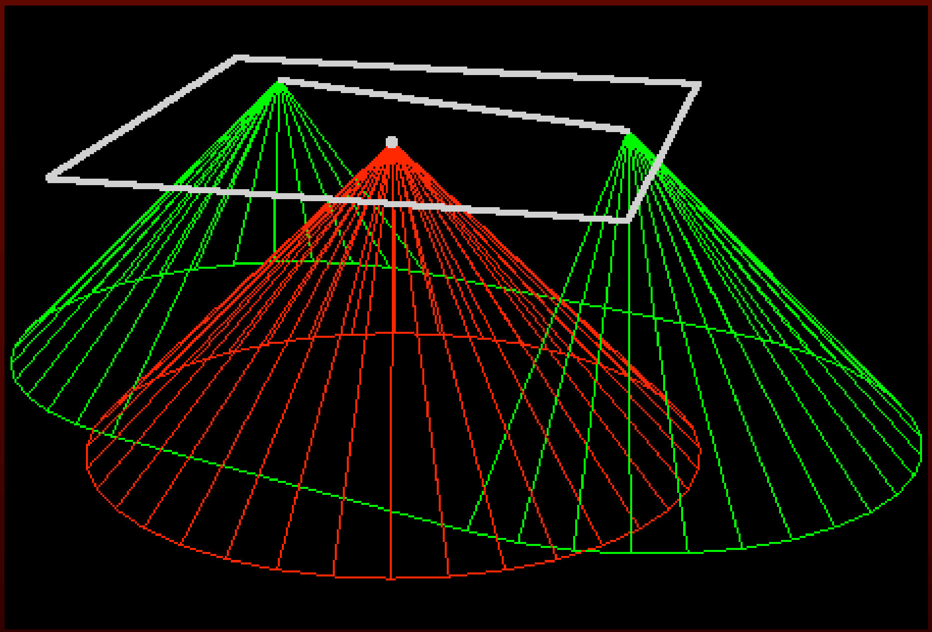
Shape of distance function for a 2D point is a cone



Need a bounded-error tessellation of the cone

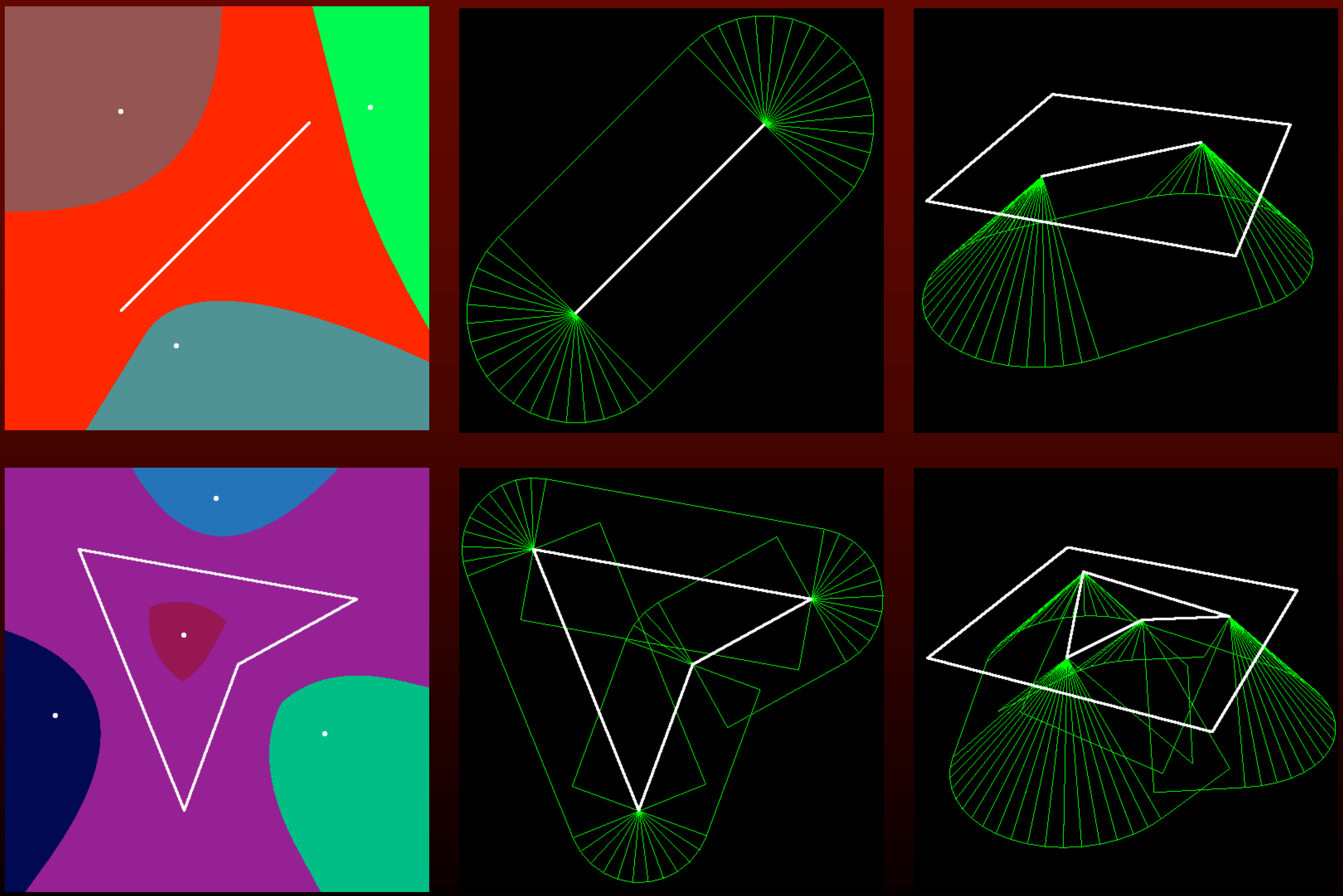
Shape of Distance Functions



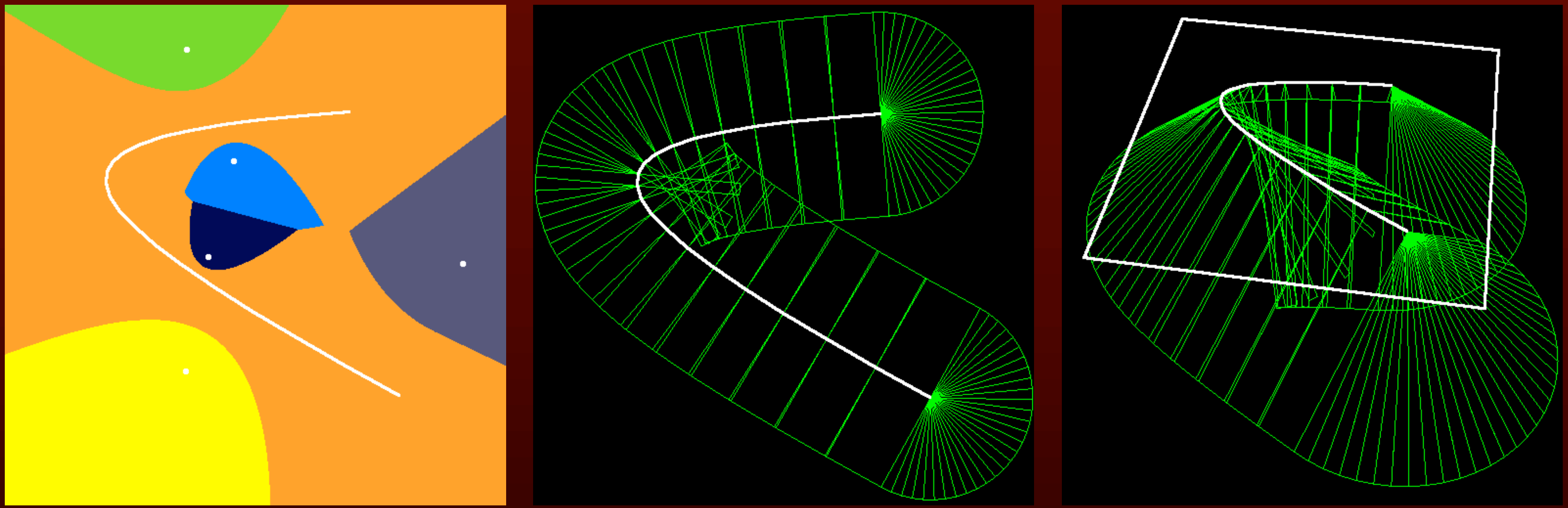


Sweep apex of cone along higher-order site to obtain the shape of the distance function

Example Distance Meshes

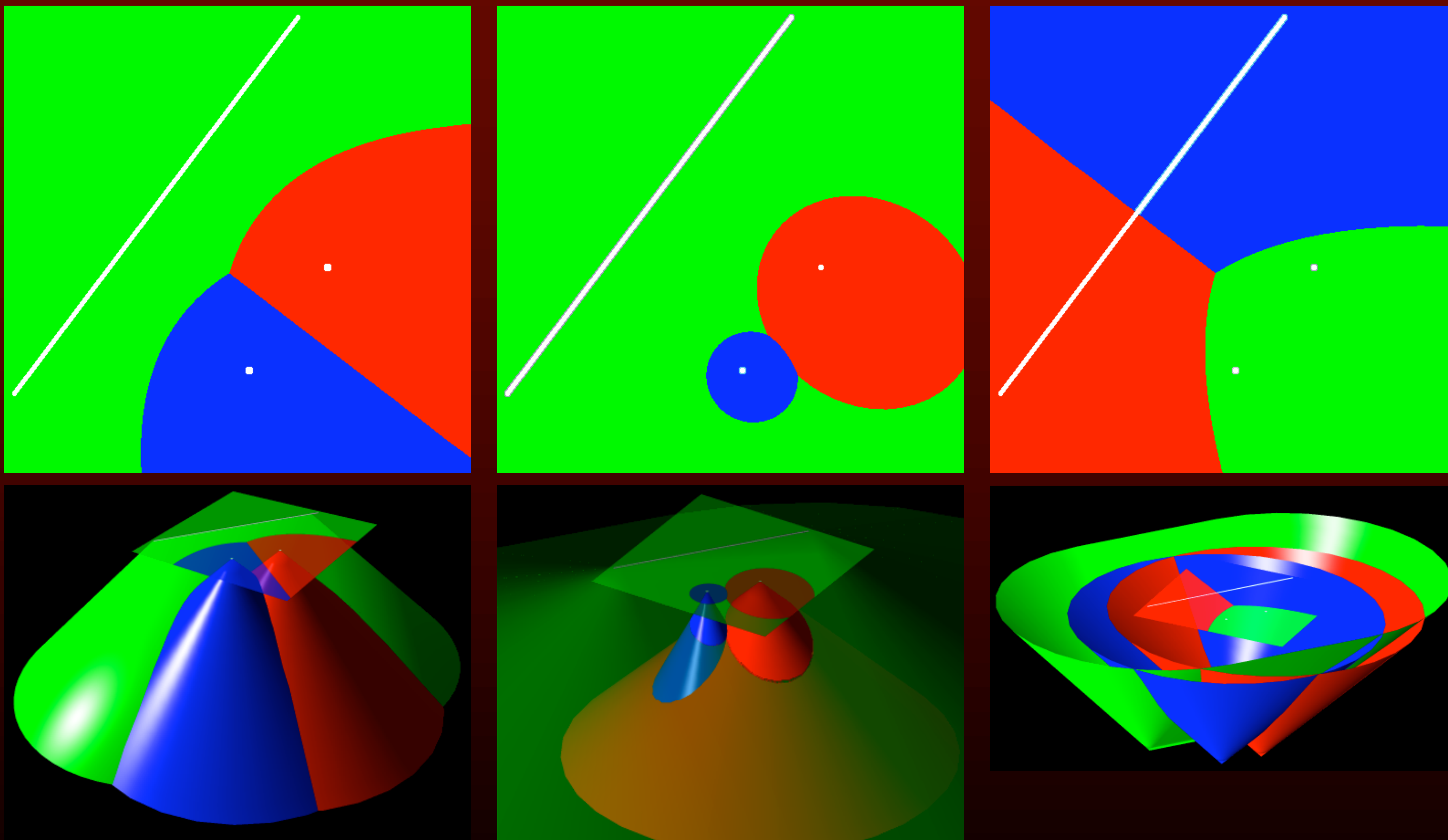


Curves



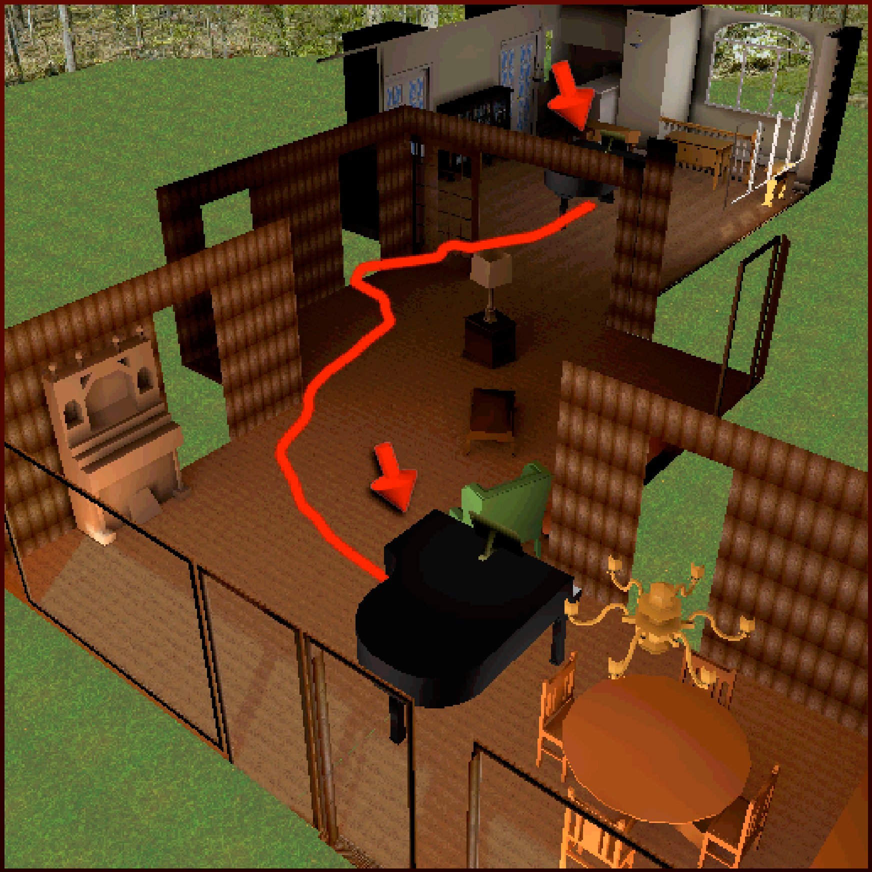
Tessellate curve into a polyline
Tessellation error is **<u>added</u>** to meshing error

Weighted and Farthest Distance

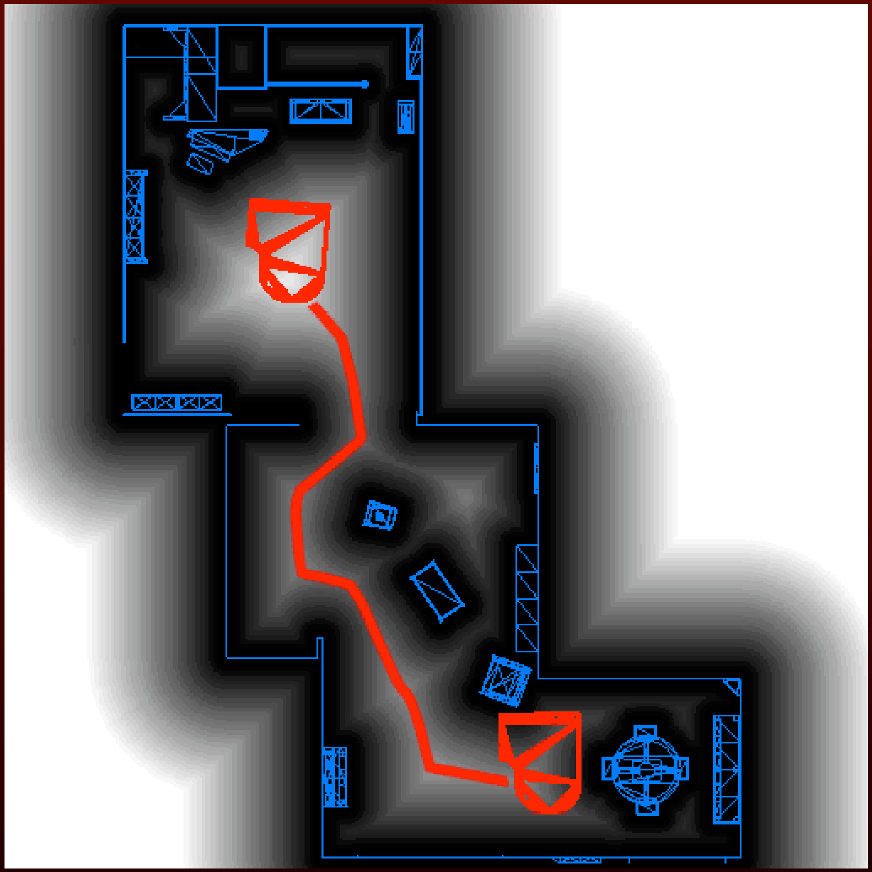


Nearest Weighted Farthest

Real-time Motion Planning : Static Scene

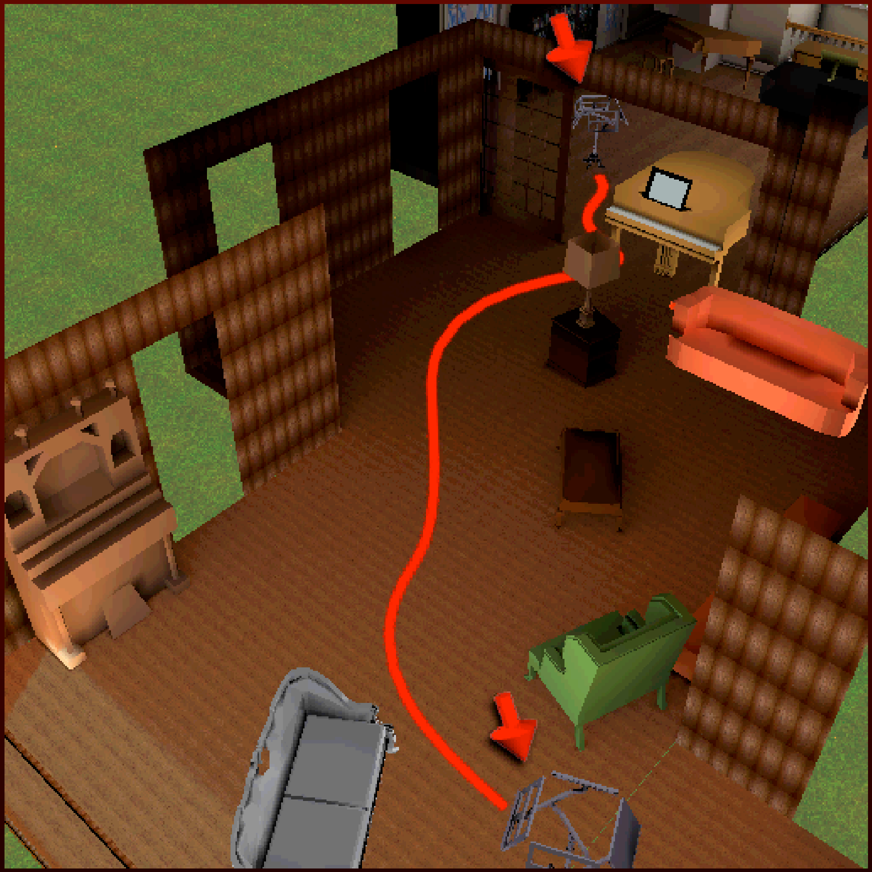


Plan motion of piano (arrow) through 100K triangle model

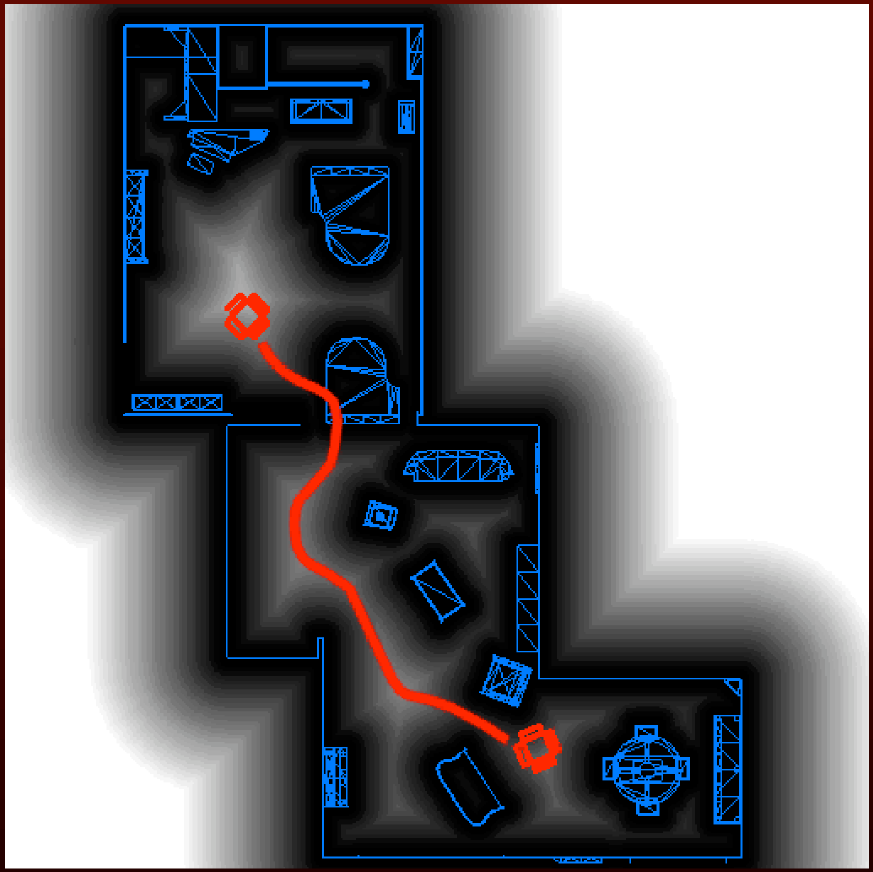


Distance buffer of floorplan used as potential field

Real-time Motion Planning : Dynamic Scene



Plan motion of music stand around moving furniture



Distance buffer of floor-plan used as potential field

Conclusion

Meshing Distance Functions

Graphics Hardware Acceleration

\+ Brute-force Approach

Fast and Simple, Approximate

Generalized Voronoi Diagrams

Bounded Error