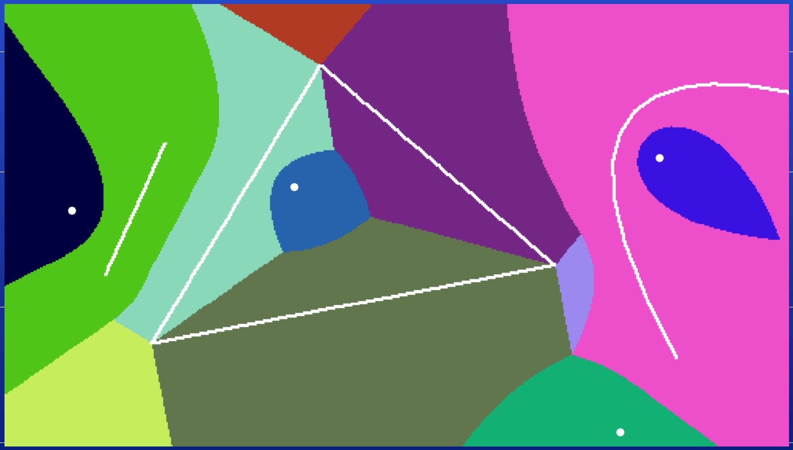
Fast Computation of Generalized Voronoi Diagrams Using Graphics Hardware

paper by Kennet E. Hoff et al.
(University of North Carolina at Chapel Hill)



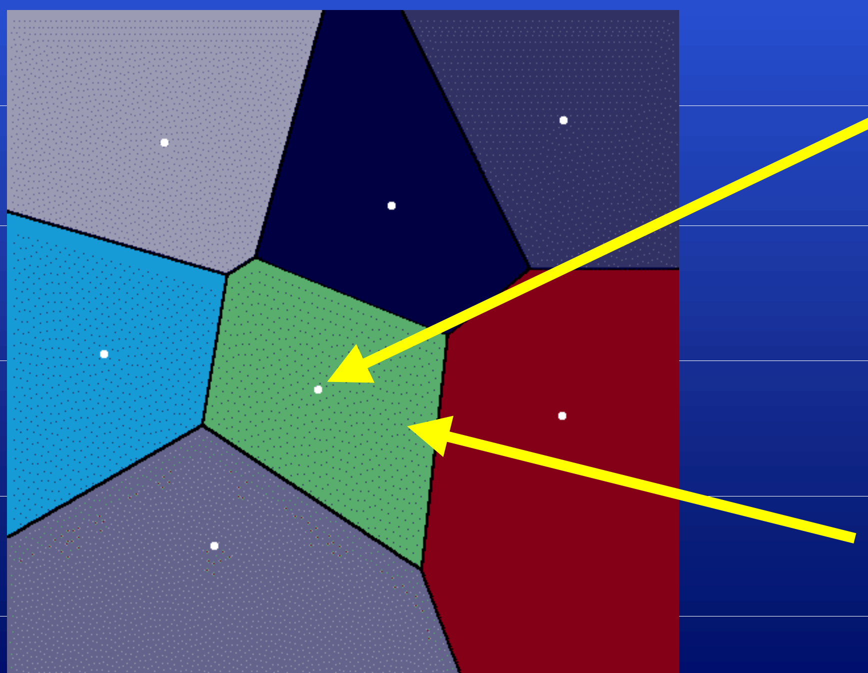
presented by Daniel Emmenegger
GDV-Seminar ETH Zürich, SS2000

Overview

- Introduction
- Motivation
- Basic Idea and Definitions
- The Distance Functions
- From 2D to 3D (basics)
- Error estimation
- Applications and Results
- Conclusions

What is a Voronoi Diagram?

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

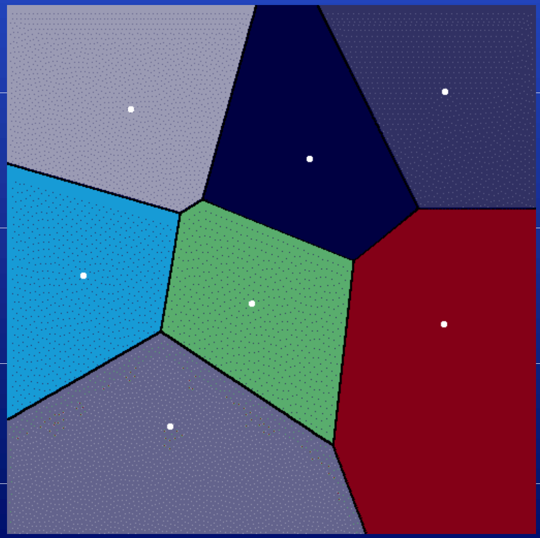


Primitive
(Voronoi site)

Voronio Region

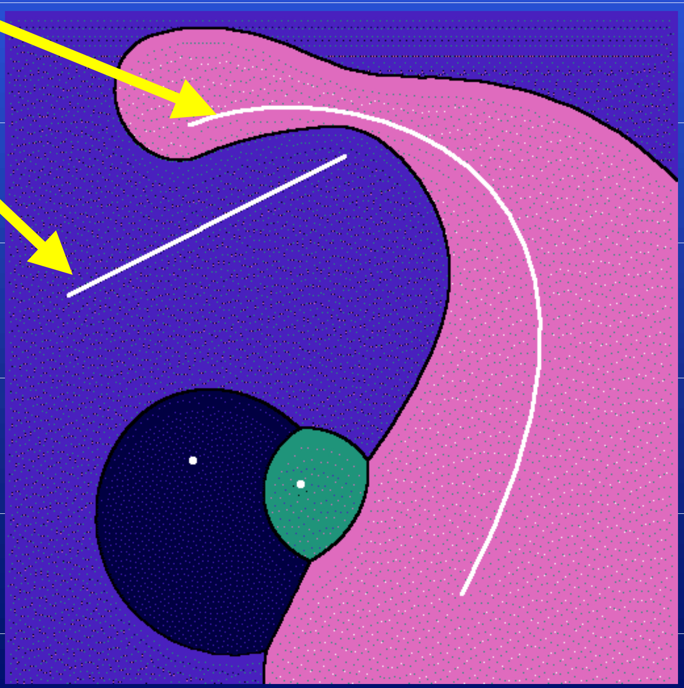
Ordinary vs. Generalized

- Primitives: Points
- Nearest Euclidean Distance



- Primitives: Points, Lines, Polygons, Curves, ...
- Varying distance metrics

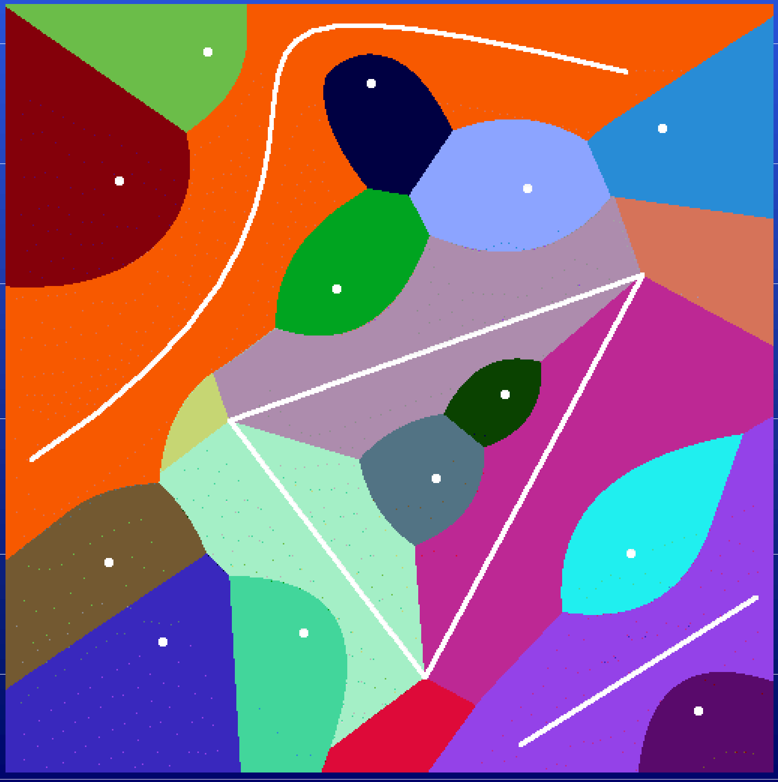
Higher-order Sites

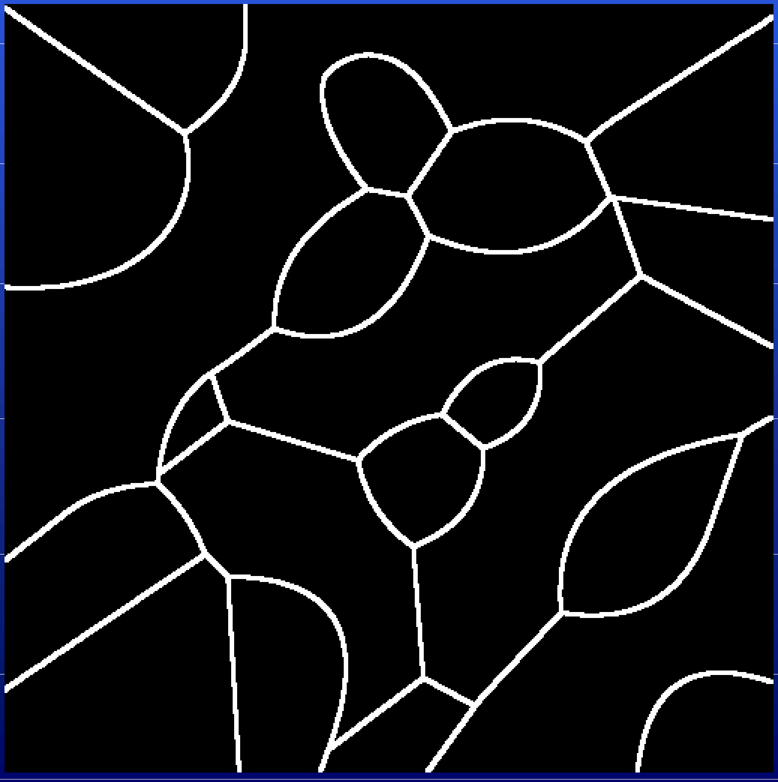


Voronoi Boundary

Curves forming the boundary between the various cells

frontier between two cells of different color

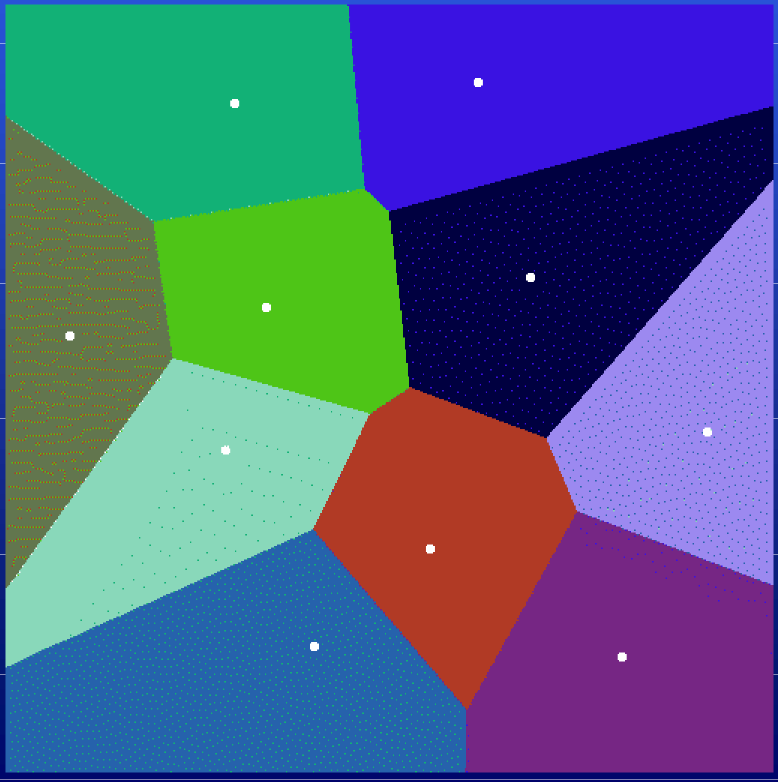


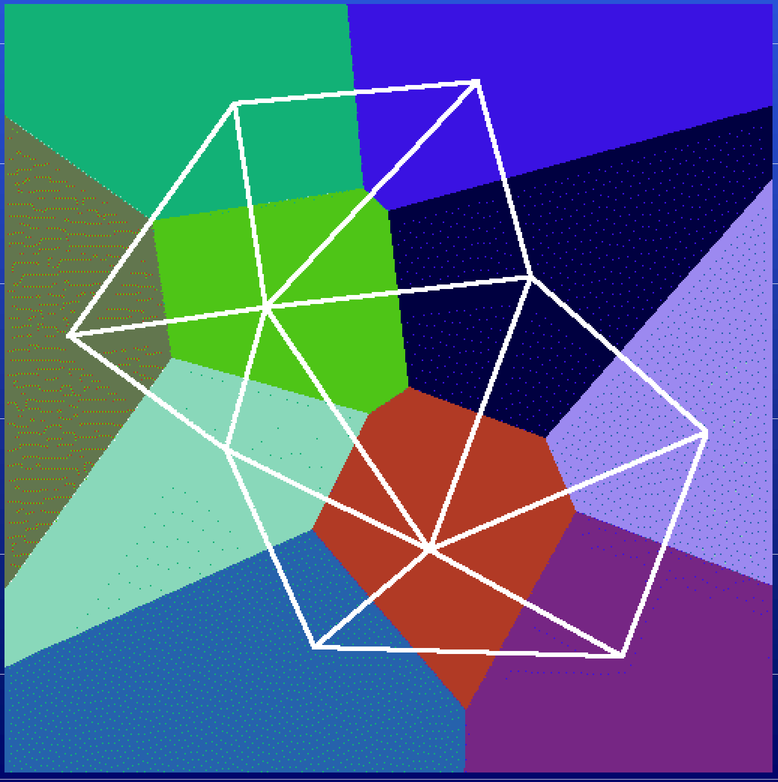


Delaunay-Triangulation

Duality structure to Ordinary Voronoi Diagrams:

Connect all primitives with their nearest neighbors



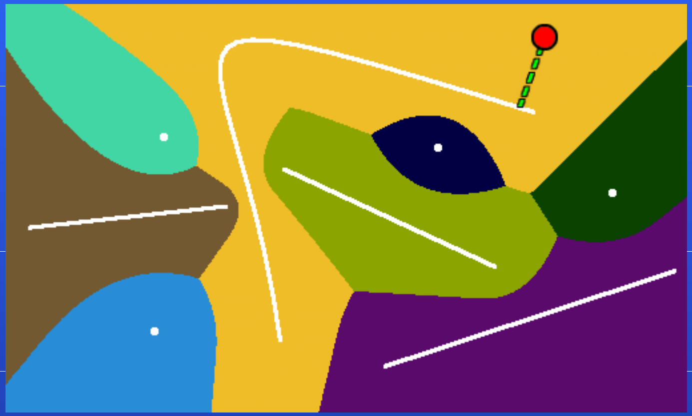


What can we do with this stuff?

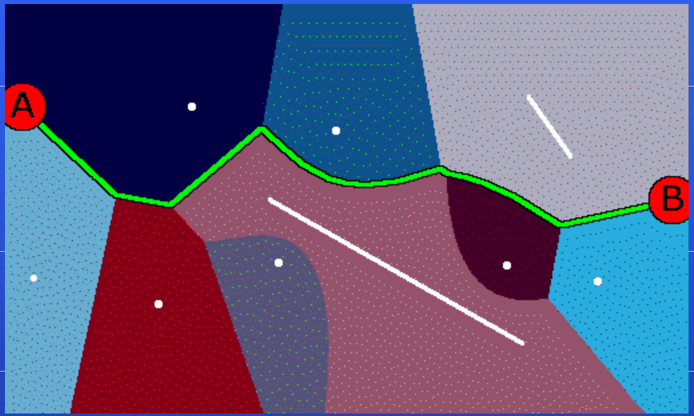
FUNDAMETAL CONCEPT:

1644	Descartes	Astronomy
1850	Dirichlet	Math
1908	Voronoi	Math
...		
1970	divers...	Computational geometry and related areas (first algorithms for computing Voronoi diagrams)
...		
1999	E. Hoff et al.	Fast Computation of Generalized Voronoi Diagrams Using Graphics Hardware

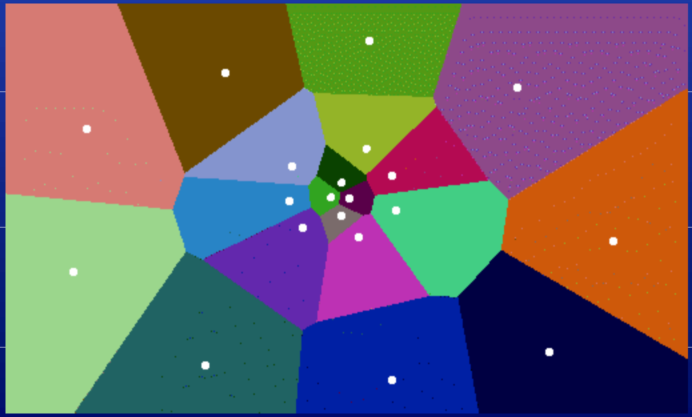
What can we do with this stuff?



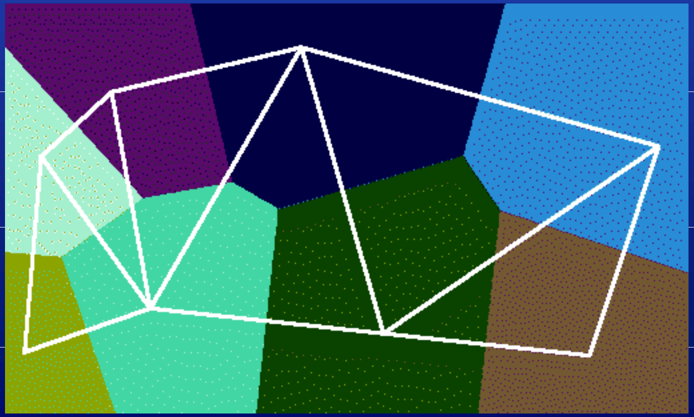
Nearest Site



Maximally Clear Path



Density Estimation



Nearest Neighbors

Motivation

- *Previous Work*: Exact Algorithms
  no error but ...
  - Boundaries composed of high-degree curves and surfaces and their intersections
  - Complex and difficult to implement
  - Robustness and accuracy problems

- *Previous Work*: Approximate Algorithms
  provide a practical solution but...
  - Difficult to error-bound
  - Restricted to static geometry
  - Relativly slow

Goals

Approximate generalized Voronoi Diagram computation with the following features:

- Easily generalized
- Efficient and practical
- Has tight bounds of accuracy
- Simple to understand and implement

Formal Definition

Set of input sites (primitives) $A_1, A_2, \ldots, A_k$

dist(p, Ai): distance from the point p to the site Ai

The dominance region of A_i over Aj is defined by

$$Dom(A_i, A_j) = \{ p \mid dist(p, A_i) \leq dist(p, A_j) \}$$

For a site A_i, the Voronoi region for A_i is defined by

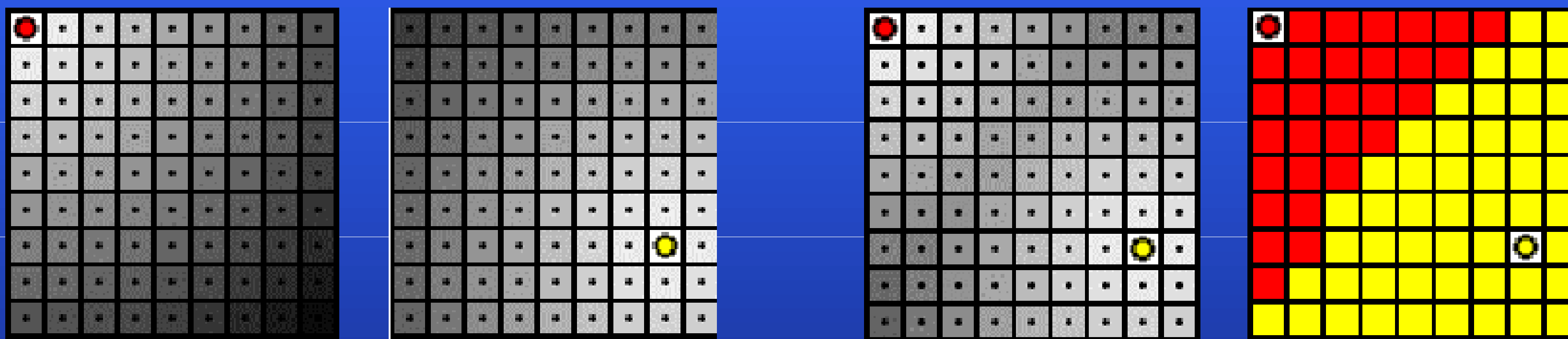
$$V(A_i) = \bigcap_{i \neq j} Dom(A_i, A_j)$$

Partition of space into $V(A_1)$, $V(A_2)$,...,$V(A_k)$:

Generalized Voronoi Diagrams

Discrete Voronoi Diagrams

Uniformliy point-sample the space containing Voronoi sites
For each sample find closest site and its distance

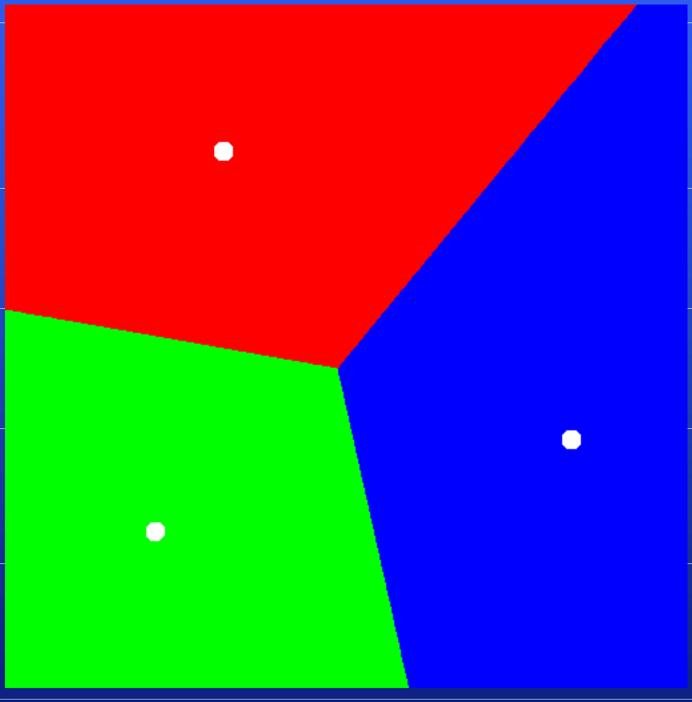


Brute-force-Algorithm:

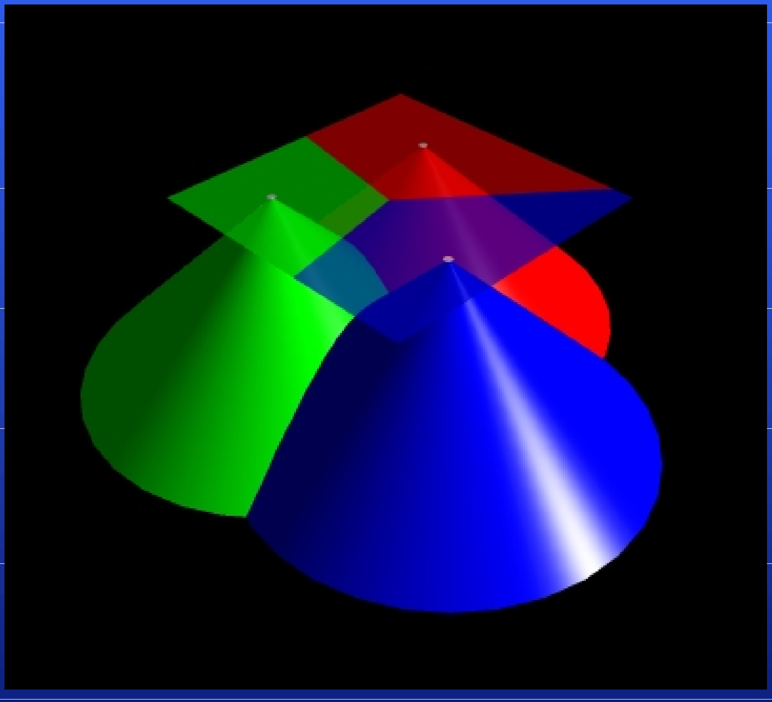
- iterate through all samplepoints (cells)
- iterate through all primitives => **HARDWARE**

Basic Idea : Cones

To visualize Voronoi Diagrams for points ...



topview, parallel

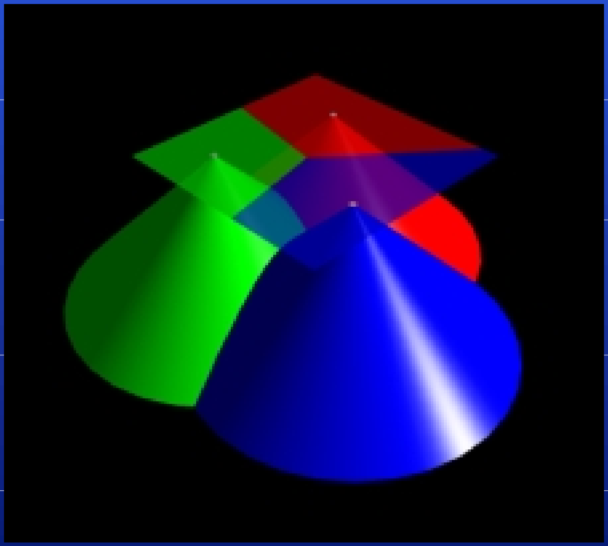


perspective view

Graphics Hardware Acceleration

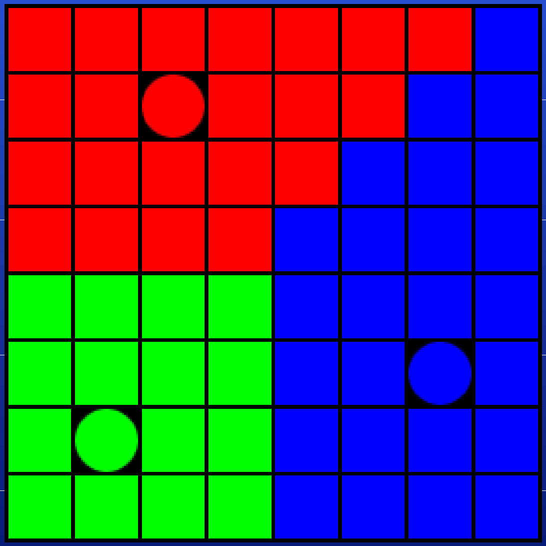
Simply rasterize the cones using graphics hardware

Our 2-part discrete Voronoi diagram representation



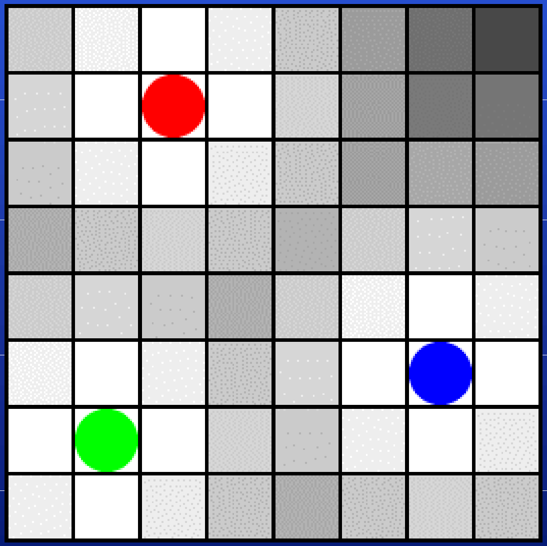
Origin: Woo97

Color Buffer



Site IDs

Depth Buffer



Distance

Basic Idea: Distance Function

Render a polygonal mesh approximation to each site's distance function.

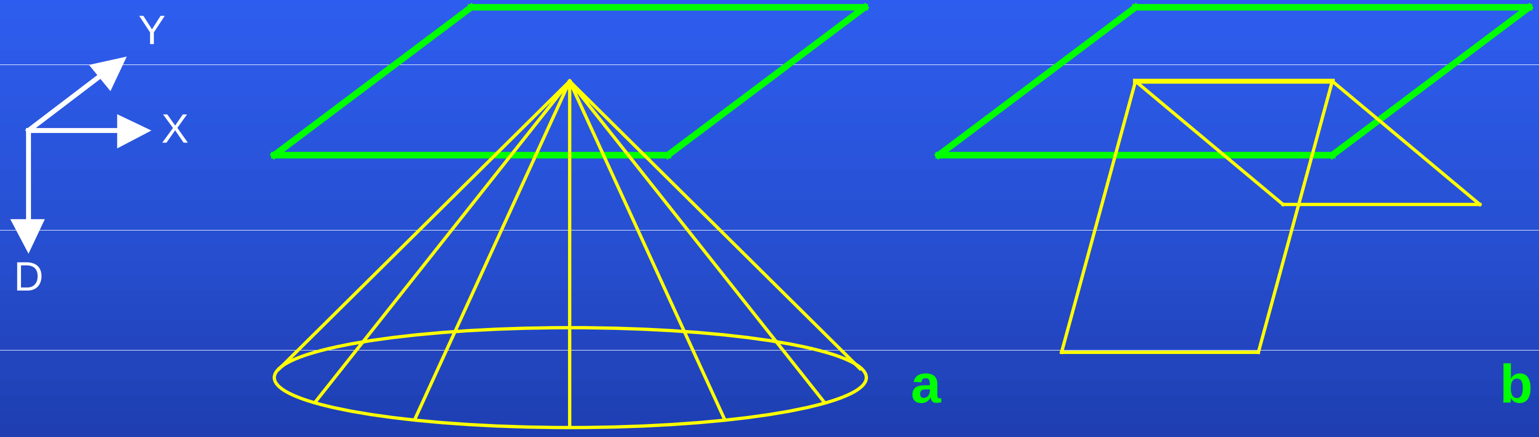
Each site has:

- unique color ID assigned
- corresponding distance mesh rendered in this color using parallel projection

We make use of:

- linear interpolation across polygons
- Z-Buffer depth comparison operation

The Distance Function



Cone

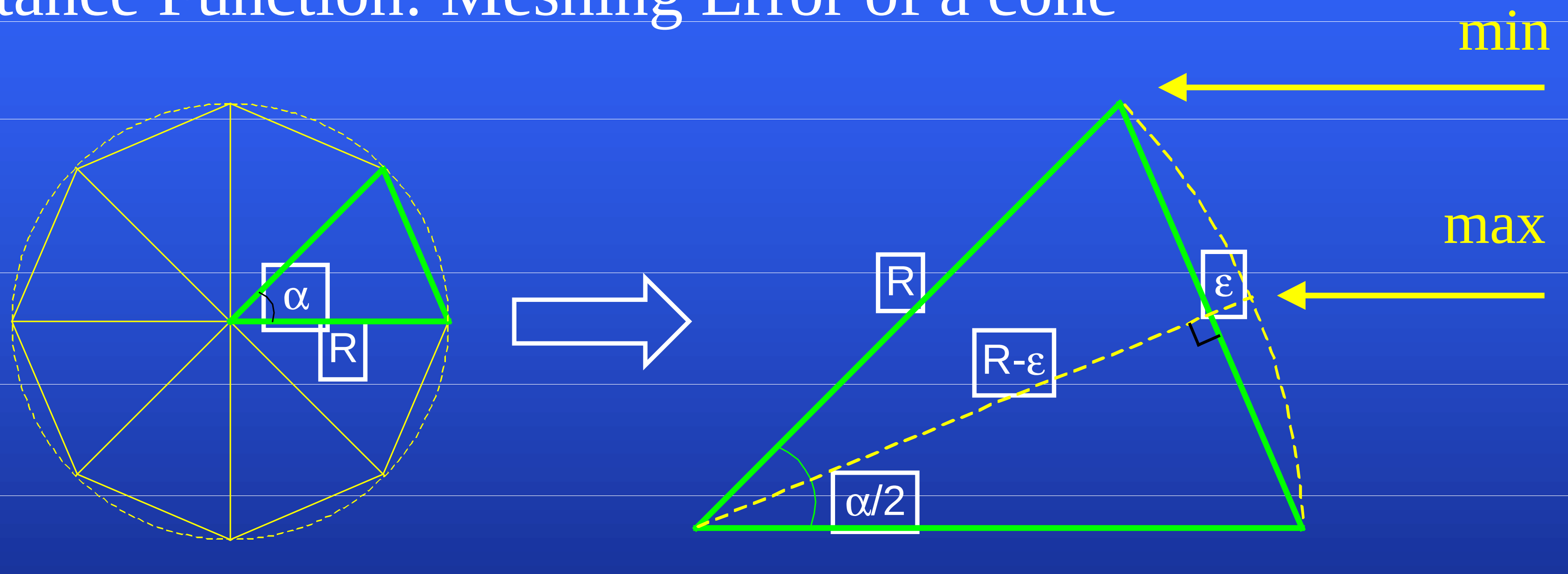
„Tent“

And Polygons, Bezièr-Curves, ... What is their distance function?

Compose them of points and lines!

Approximation Error

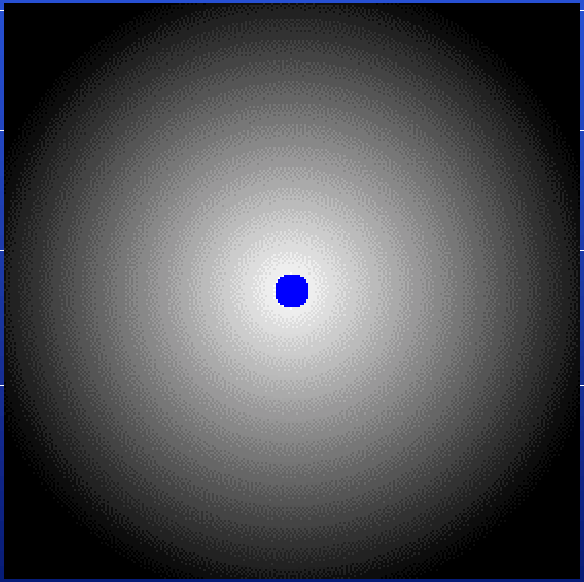
Distance Function: Meshing Error of a cone



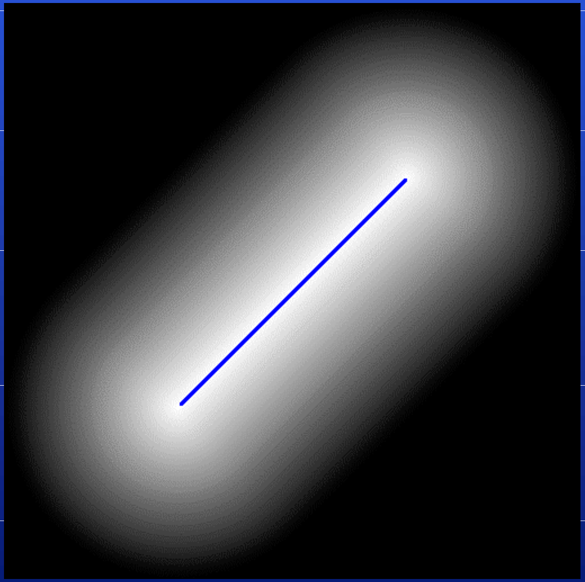
$$\cos\left(\frac{\alpha}{2}\right) = \frac{R-\varepsilon}{R} \rightarrow \alpha = 2\cos^{-1}\left(\frac{R-\varepsilon}{R}\right)$$

Distance Function

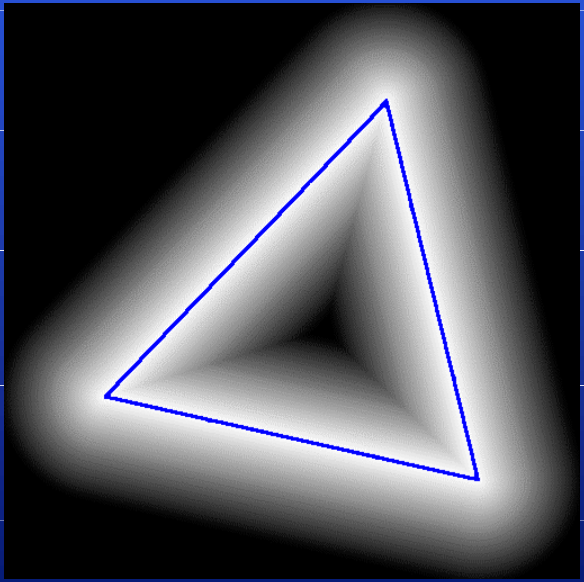
- Evaluate distance at each pixel for all sites
- Accelerate using graphics hardware



Point



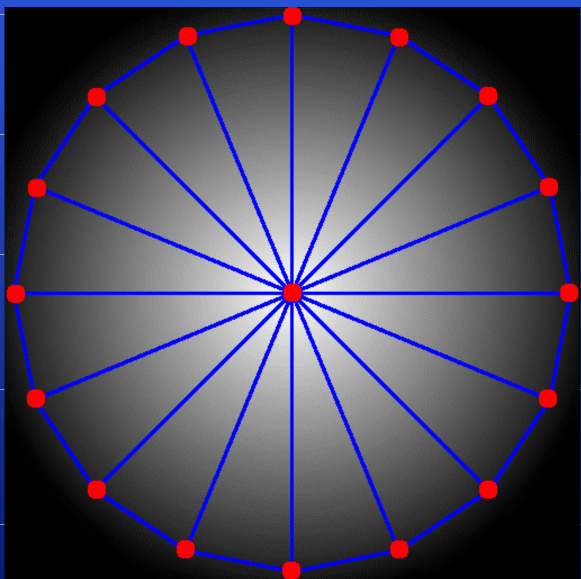
Line



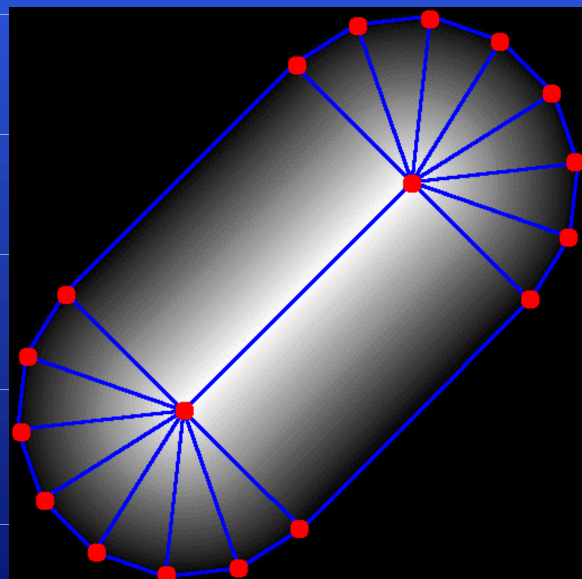
Triangle

Approximation of the Distance Function

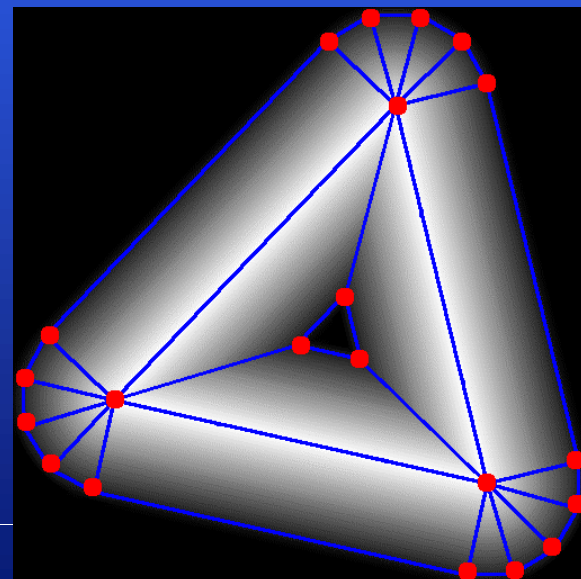
- Avoid per-pixel distance evaluation
- Point-sample the distance function
- Reconstruct by rendering polygonal mesh



Point

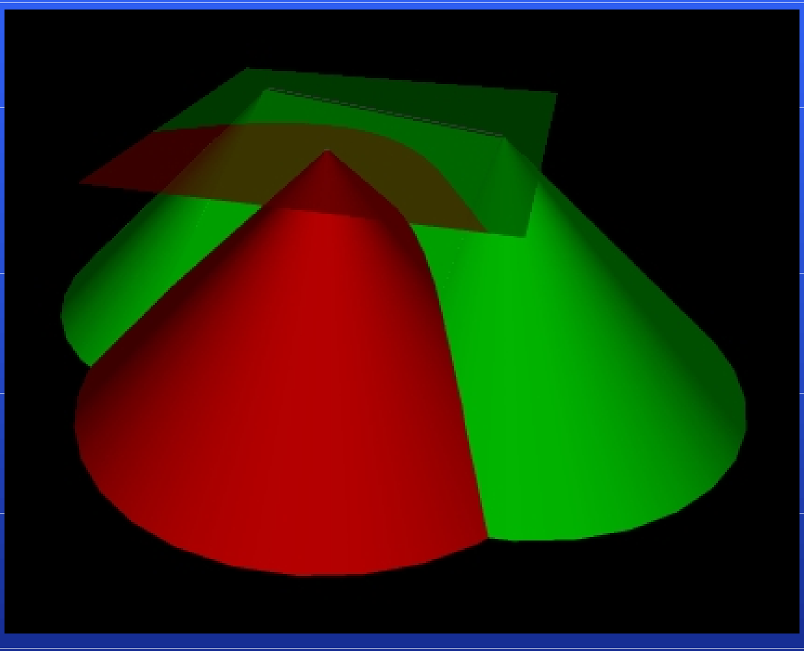


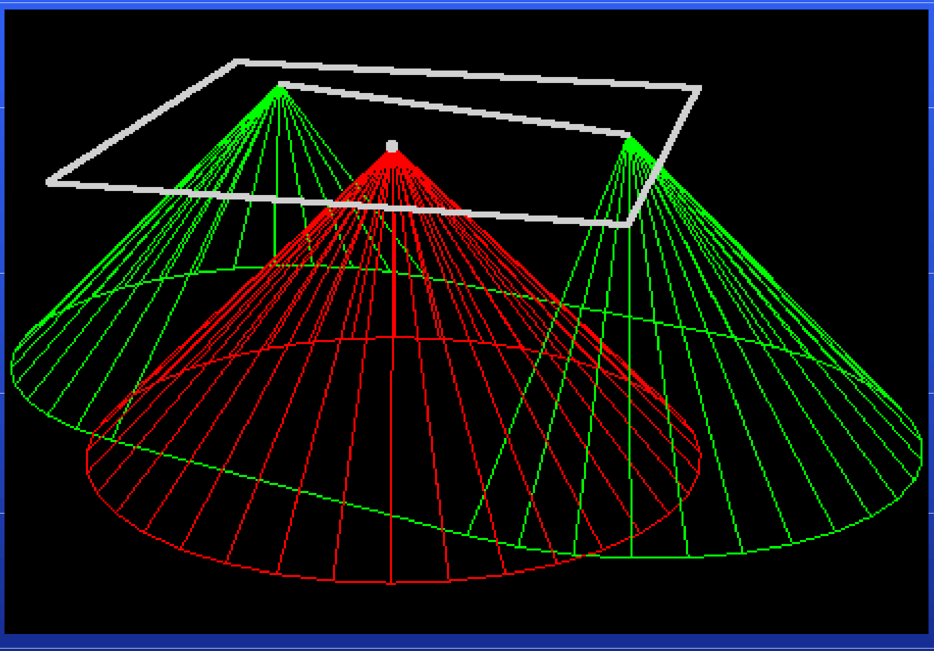
Line



Triangle

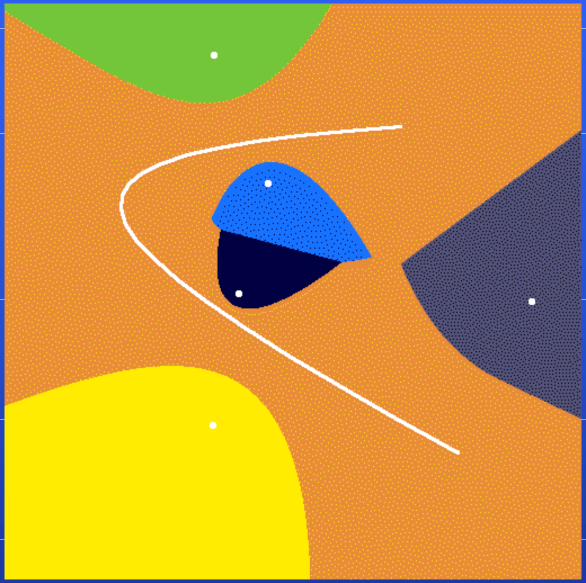
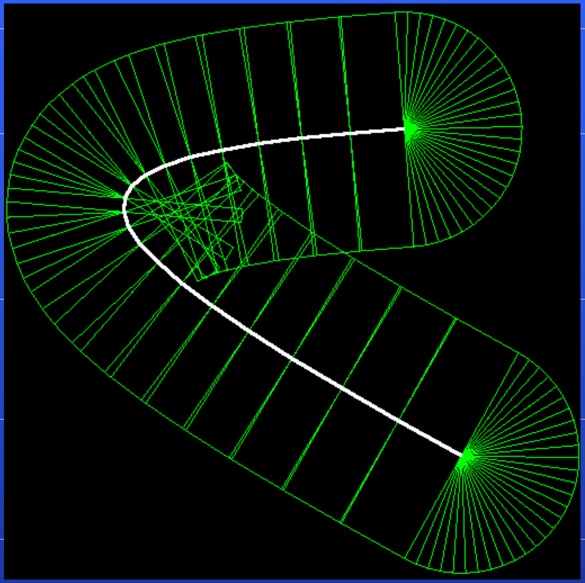
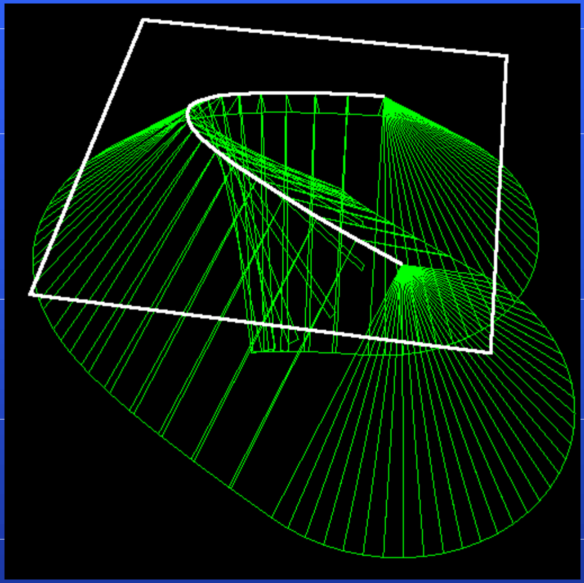
Shape of Distance Function





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

Curves

Tessellate curve into a polyline

Tessellation error is **added** to meshing error

Boundaries & ...

Algorithm A: (very simple, accelerated through image op. in ghw)
- examine each pairs of adjacent cells
- if color different, location between is marked as boundary-point

Algorithm B: *continuation method*
- choose seed (known point of boundary)
- walk along boundaries until all boundry points are found

... Neighbors

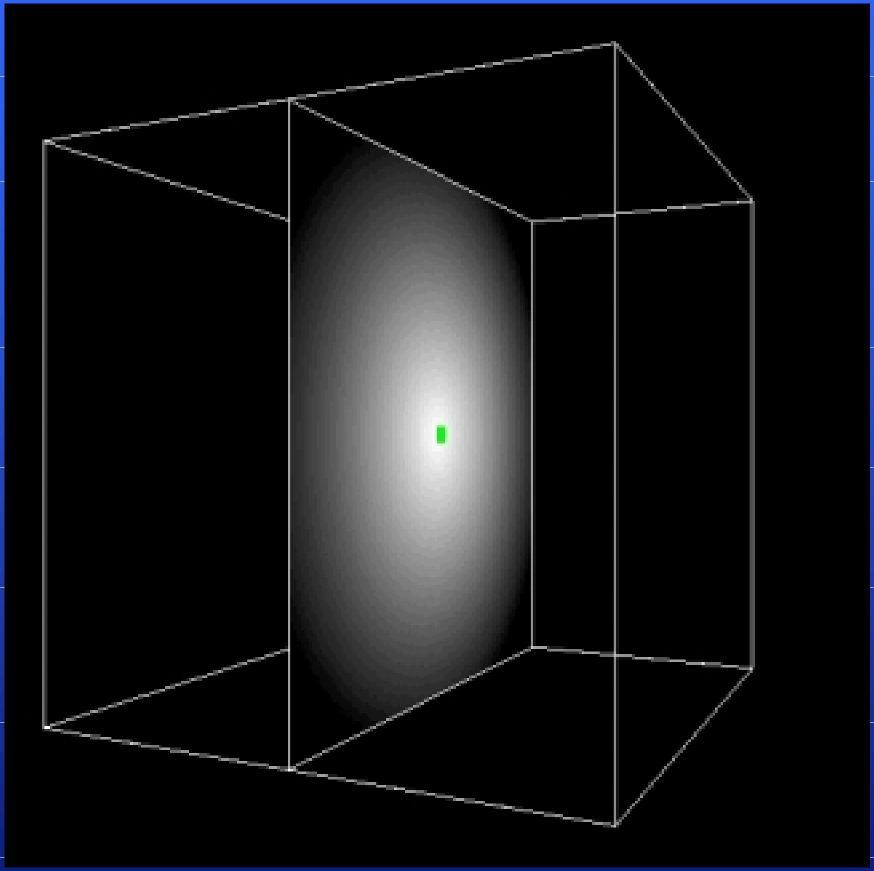
Main Question:

Which colors touch in the image?

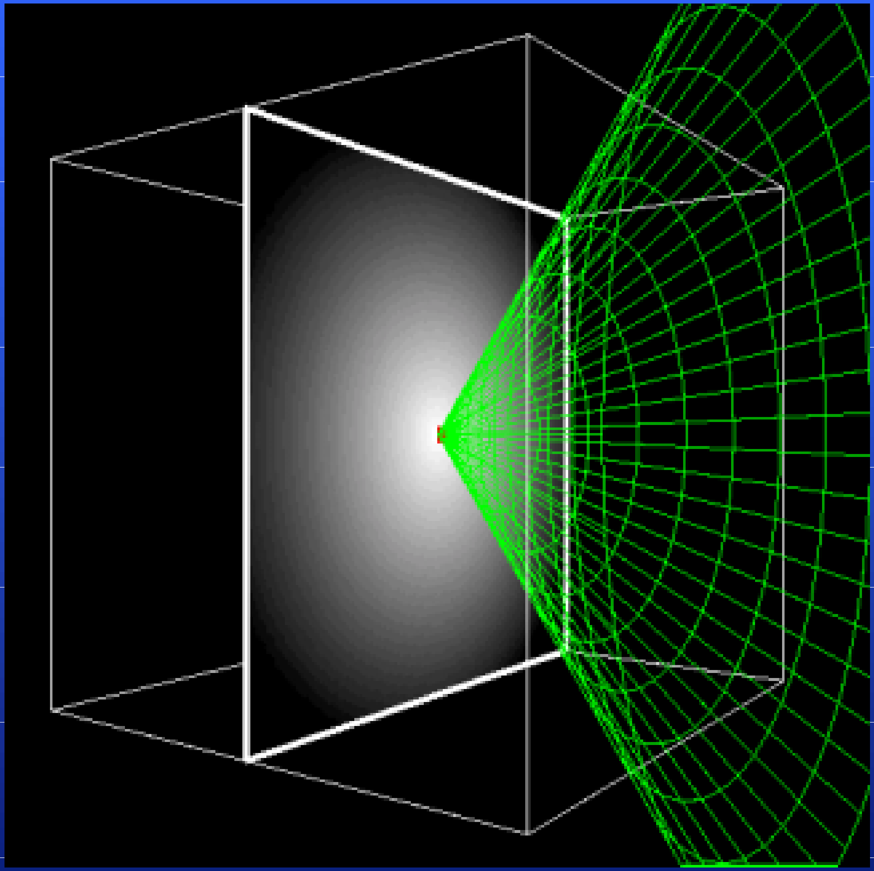
Answer, how to find them:

Same algorithm as used for finding boundaries

What about 3D?

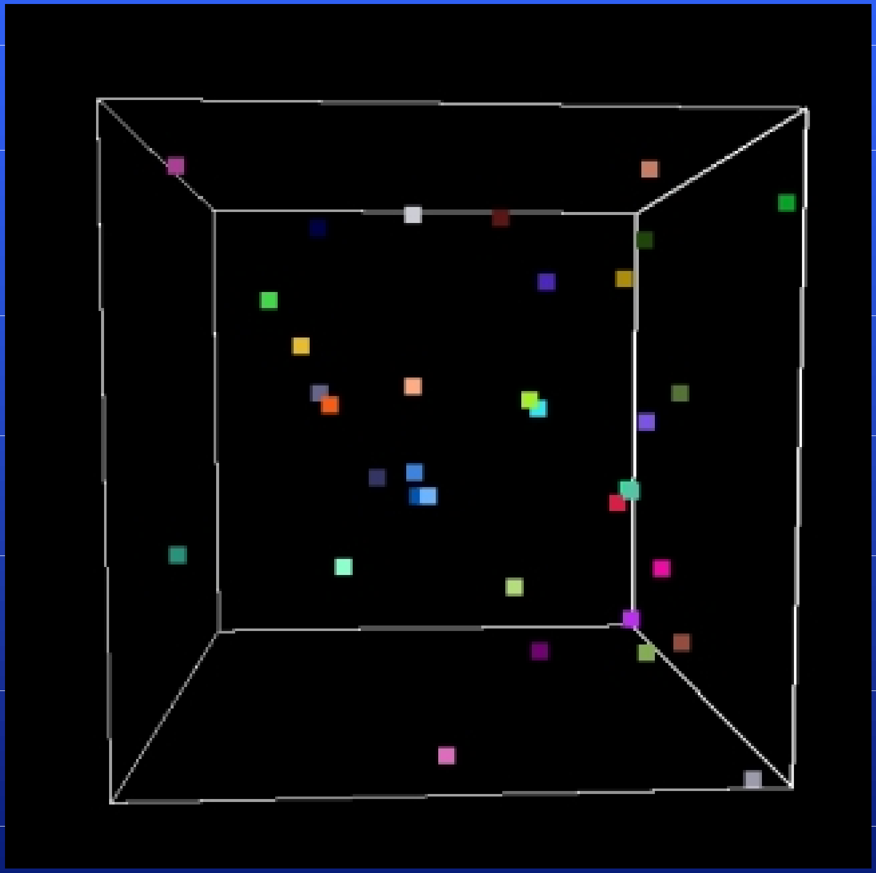


Slices of the distance function for a 3D point site



Distance meshes used to approximate slices

What about 3D?

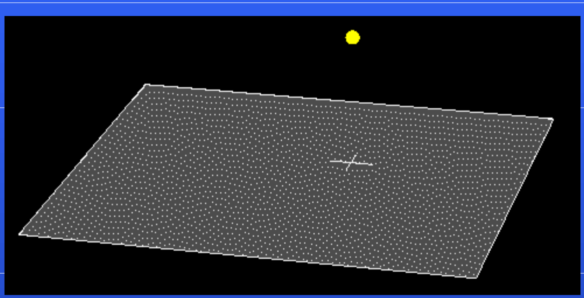


Point sites

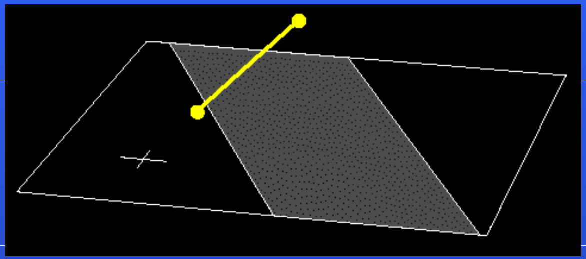
Graphics hardware can generate one 2D slice at a time

3D Distance Functions

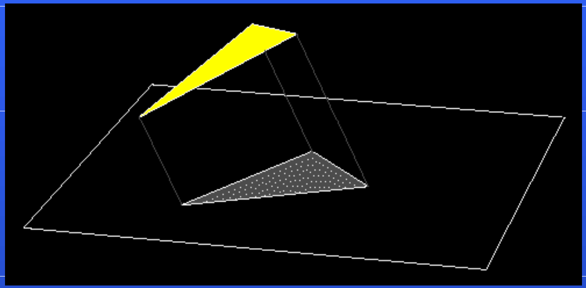
Point

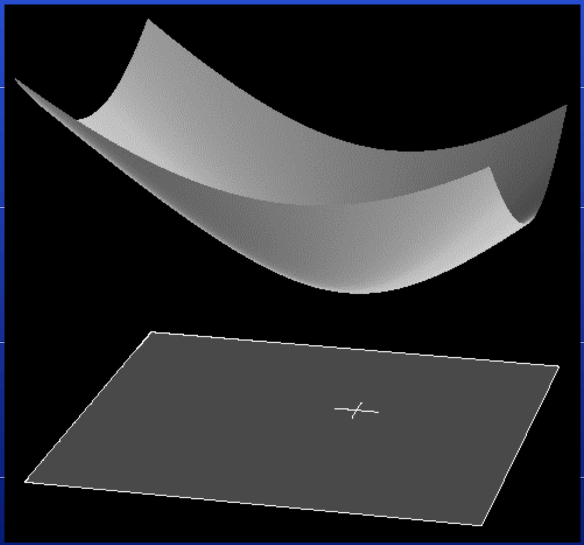


Line segment

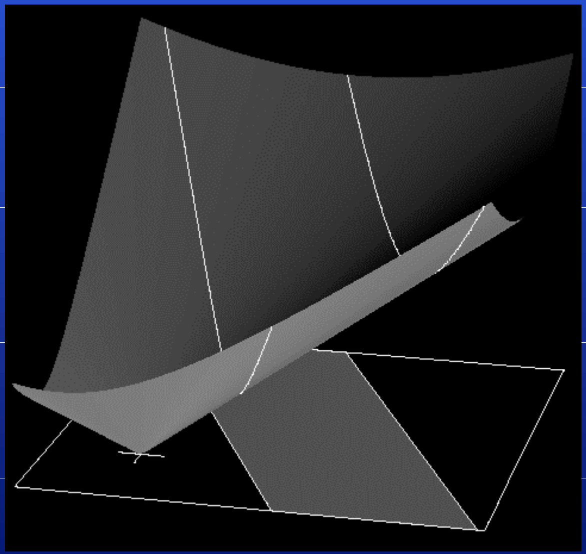


Triangle

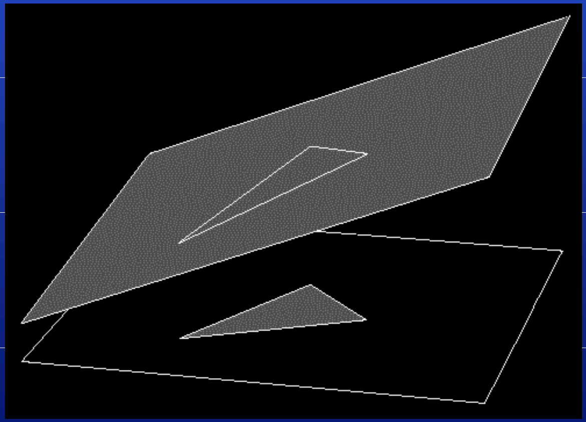




1 sheet of a hyperboloid



Elliptical cone



Plane

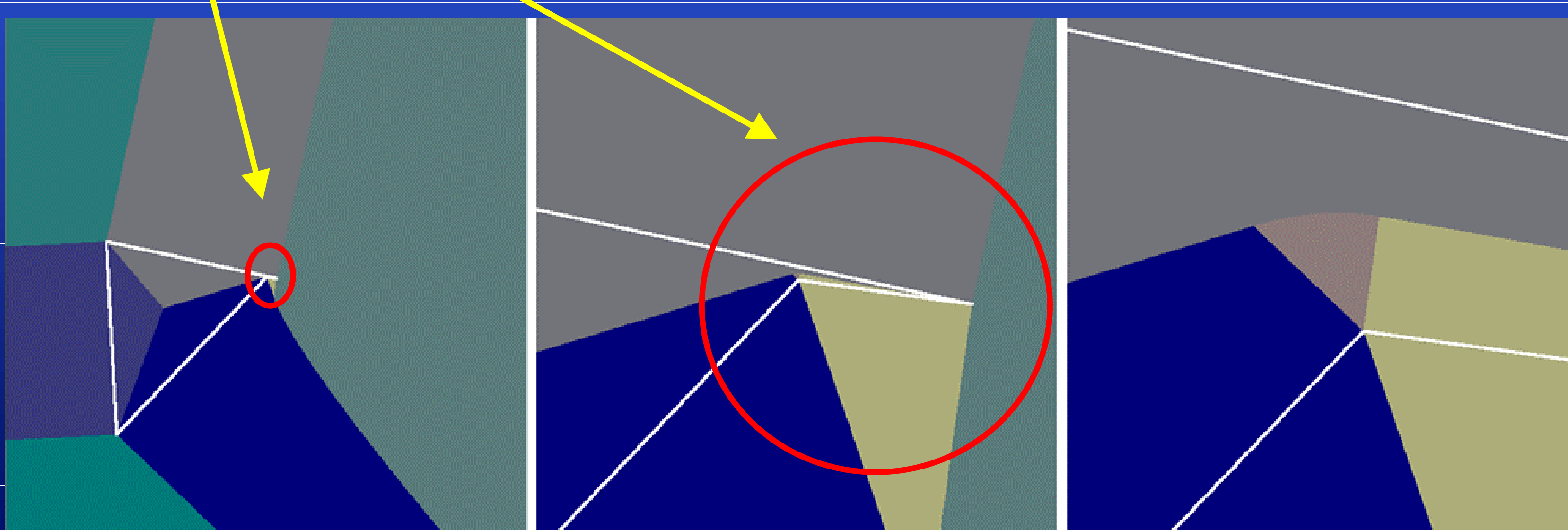
Sources of Error

- Distance Error
  - meshing
  - tesselation
  - hardware precicision
- Combinatorial Error
  - Z-Buffer precision
  - distance
  - pixel resolution

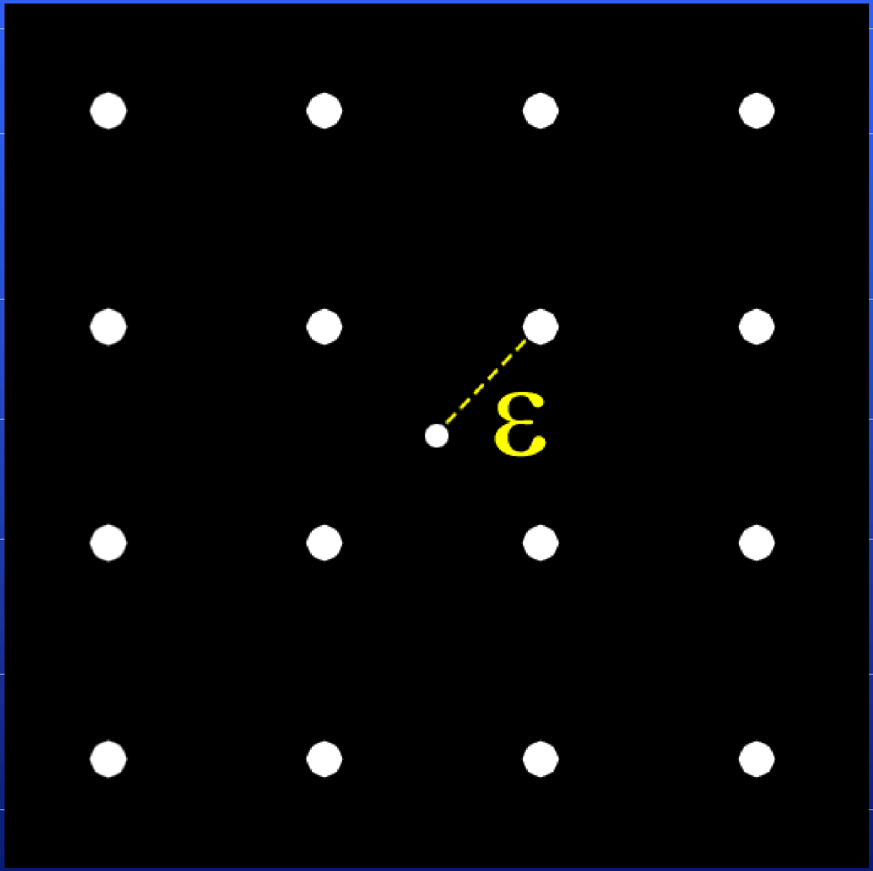
Resolution Error

Adaptive Resolution

zoom in to reduce resolution error



Error Bounds



Error bound is determined by the pixel resolution

$\varepsilon \leq$ farthest distance a point can be from a pixel sample point

Error Bounds

Assume: no Z-Buffer precision error

we can bound the maximum distance error by ε

for a pixel P colored with ID of site (primitive) A and with computed depth buffer of value D, we know:

$$D - \varepsilon \le dist(P, A) \le D + \varepsilon$$

further we know, for any other site B

$$D - \varepsilon \le dist(P, B)$$

With this information we easily determine that

$$dist(P, A) \le dist(P, B) + 2\varepsilon$$

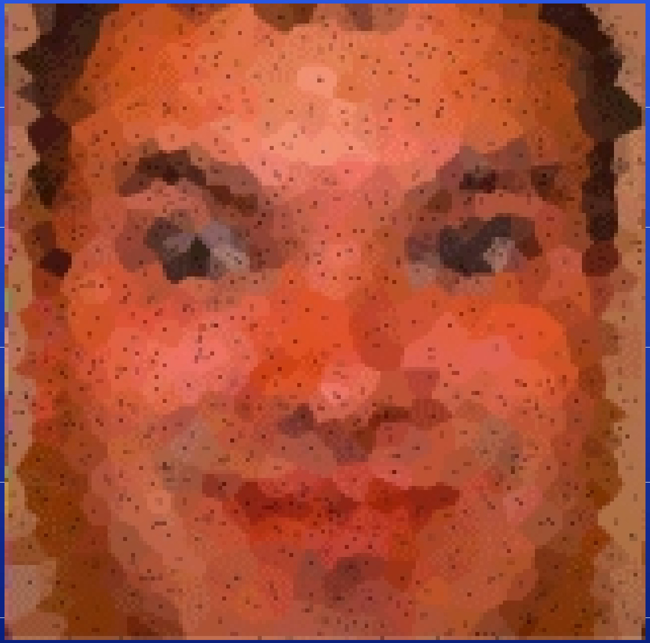
Implementation

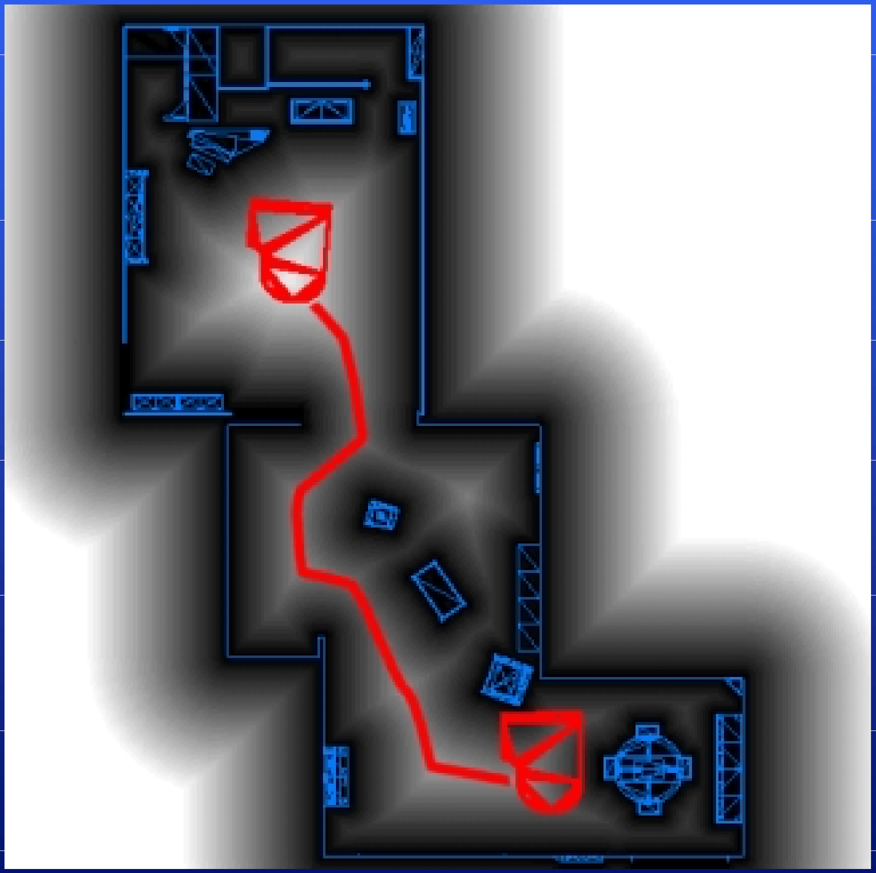
- complete interactive system in 2D
  - written in C++ using OpenGL and GLUT
  - a standard Z-buffered interpolation-based raster graphics system
- some first prototypes in 3D
- runs (without source modification) on:
  - MS-Windows-based PC
  - high-end SGI Onxy2
- several problem-based modifications to increase performace...

Applications

Mosaics

Realtime Motion Planning





Demo

VIDEO

Conclusions

- General:
  - Idea is originally not from E. Hoff or one of the other writters => Open GL Programming Guide, 2nd Edition M. Woo et al.

- My opinion:
  - Concept very easy to understand...
  - ...but the main idea is not immediatly obvious!
  - All ideas are implemented, so the reader can easily determine if everything (the notion of distance function etc.) really works

THE END

Questions?

http://www.cs.unc.edu/~hoff/