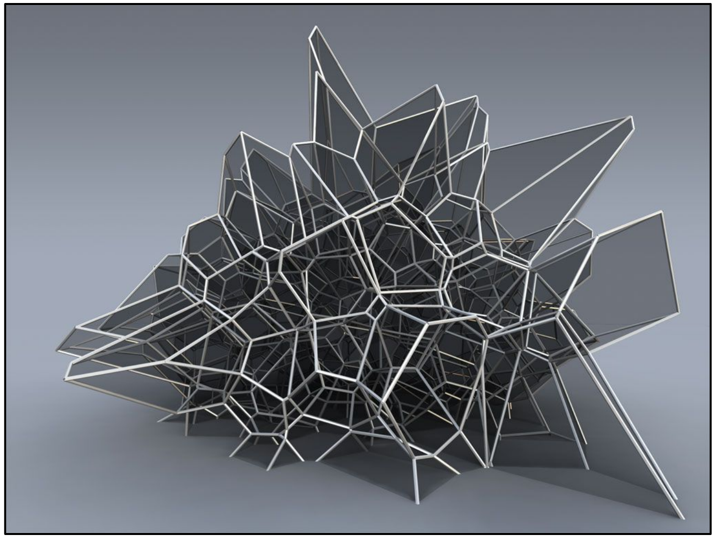
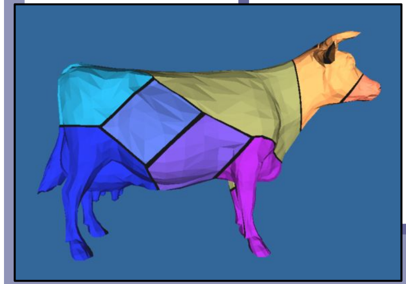
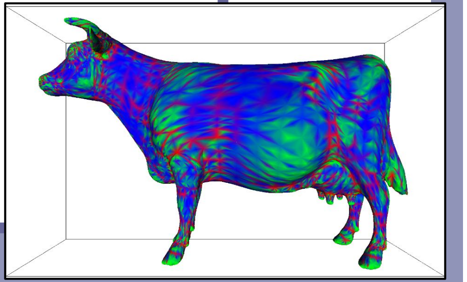
Further Graphics



A Brief Introduction to Computational Geometry



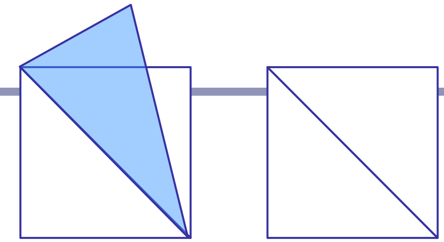


Alex Benton, University of Cambridge – alex@bentonian.com

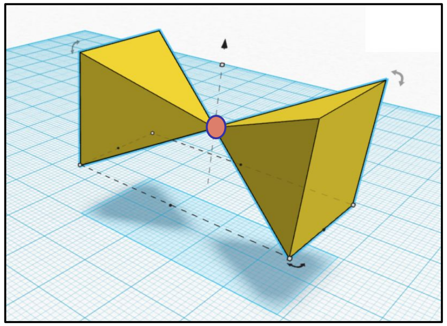
Supported in part by Google UK, Ltd

Terminology

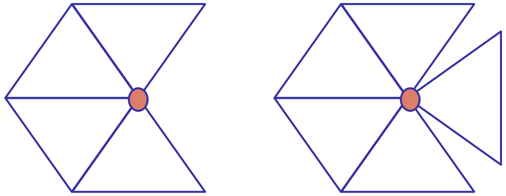
- We'll be focusing on *discrete* (as opposed to continuous) representation of geometry; i.e., polygon meshes
  - Many rendering systems limit themselves to triangle meshes
  - Many require that the mesh be *manifold*
- In a *closed manifold* polygon mesh:
  - Exactly two triangles meet at each edge
  - The faces meeting at each vertex belong to a single, connected loop of faces
- In a *manifold with boundary*:
  - At most two triangles meet at each edge
  - The faces meeting at each vertex belong to a single, connected strip of faces



Edge: Non-manifold vs manifold



Non-manifold vertex

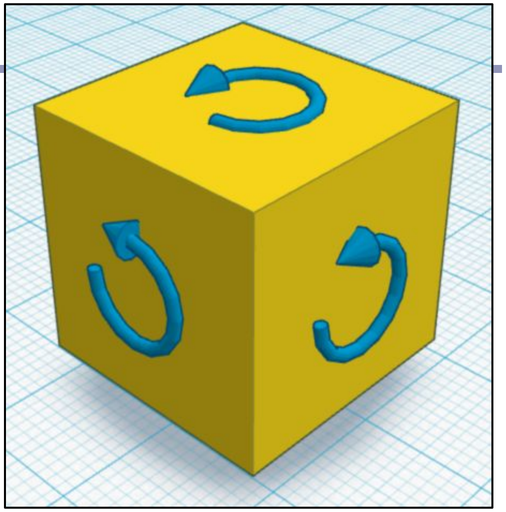


Vertex: Good boundary vs bad

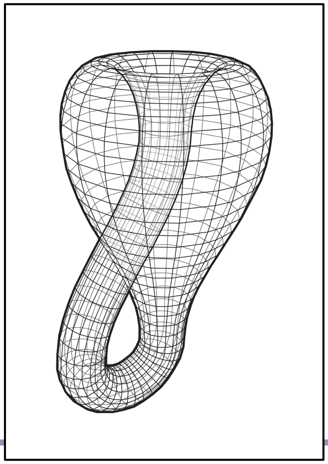
This slide draws much inspiration from Shirley and Marschner's *Fundamentals of Computer Graphics*, pp. 262-263

Terminology

- We say that a surface is *oriented* if:
  a. the vertices of every face are stored in a fixed order
  b. if vertices *i*, *j* appear in both faces *f1* and *f2*, then the vertices appear in order *i*, *j* in one and *j*, *i* in the other
- We say that a surface is *embedded* if, informally, "nothing pokes through":
  a. No vertex, edge or face shares any point in space with any other vertex, edge or face except where dictated by the data structure of the polygon mesh
- A closed, embedded surface must separate 3-space into two parts: a bounded *interior* and an unbounded *exterior*.



A cube with "anti-clockwise" oriented faces



Klein bottle: not an embedded surface.

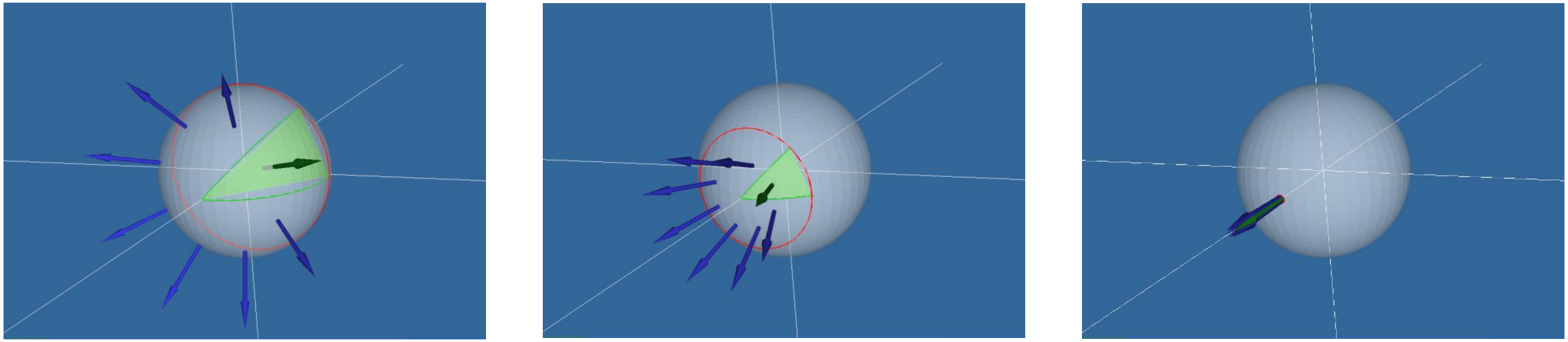
Also, terrible for holding drinks.

This slide draws much inspiration from Hughes and Van Dam's *Computer Graphics: Principles and Practice*, pp. 637-642

Normal at a vertex

Expressed as a limit,

The *normal of surface S at point P* is the limit of the cross-product between two (non-collinear) vectors from *P* to the set of points in *S* at a distance *r* from *P* as *r* goes to zero. [Excluding orientation.]



Normal at a vertex

Using the limit definition, is the ‘normal’ to a discrete surface necessarily a vector?

- The normal to the surface at any point on a face is a constant vector.
- The ‘normal’ to the surface at any edge is an arc swept out on a unit sphere between the two normals of the two faces.
- The ‘normal’ to the surface at a vertex is a space swept out on the unit sphere between the normals of all of the adjacent faces.

Finding the normal at a vertex

Take the weighted average of the normals of surrounding polygons, weighted by each polygon's *face angle* at the vertex

Face angle: the angle α formed at the vertex *v* by the vectors to the next and previous vertices in the face *F*

$$\alpha(F, v_i) = cos^{-1}\left(\frac{v_{i+1} - v_i}{|v_{i+1} - v_i|} \bullet \frac{v_{i-1} - v_i}{|v_{i-1} - v_i|}\right)$$

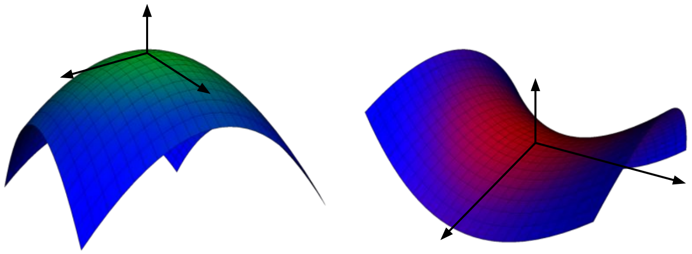
$$N(v) = \frac{\sum_F \alpha(F, v)\, N_F}{|\sum_F \alpha(F, v)|}$$

Note: In this equation, *arccos* implies a convex polygon. Why?

Gaussian curvature on smooth surfaces

Informally speaking, the *curvature* of a surface expresses "how flat the surface isn't".

- One can measure the directions in which the surface is curving *most*; these are the directions of *principal curvature*, k_1 and k_2.
- The product of k_1 and k_2 is the scalar *Gaussian curvature*.



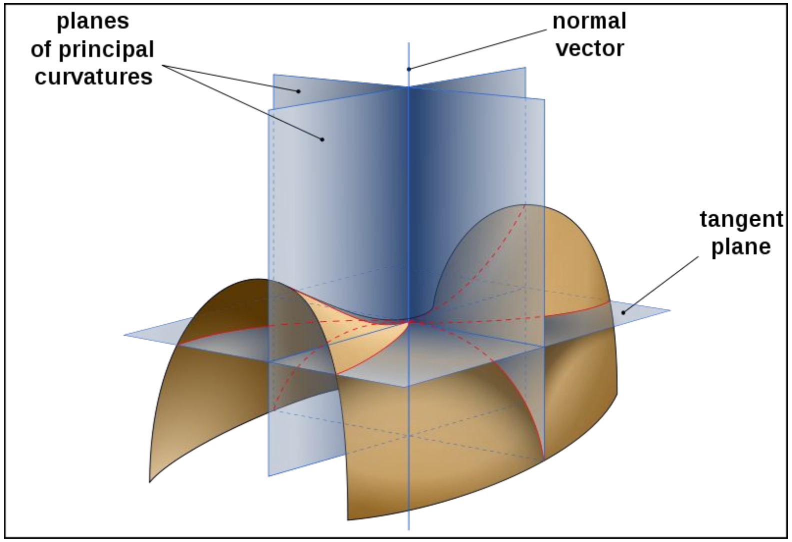
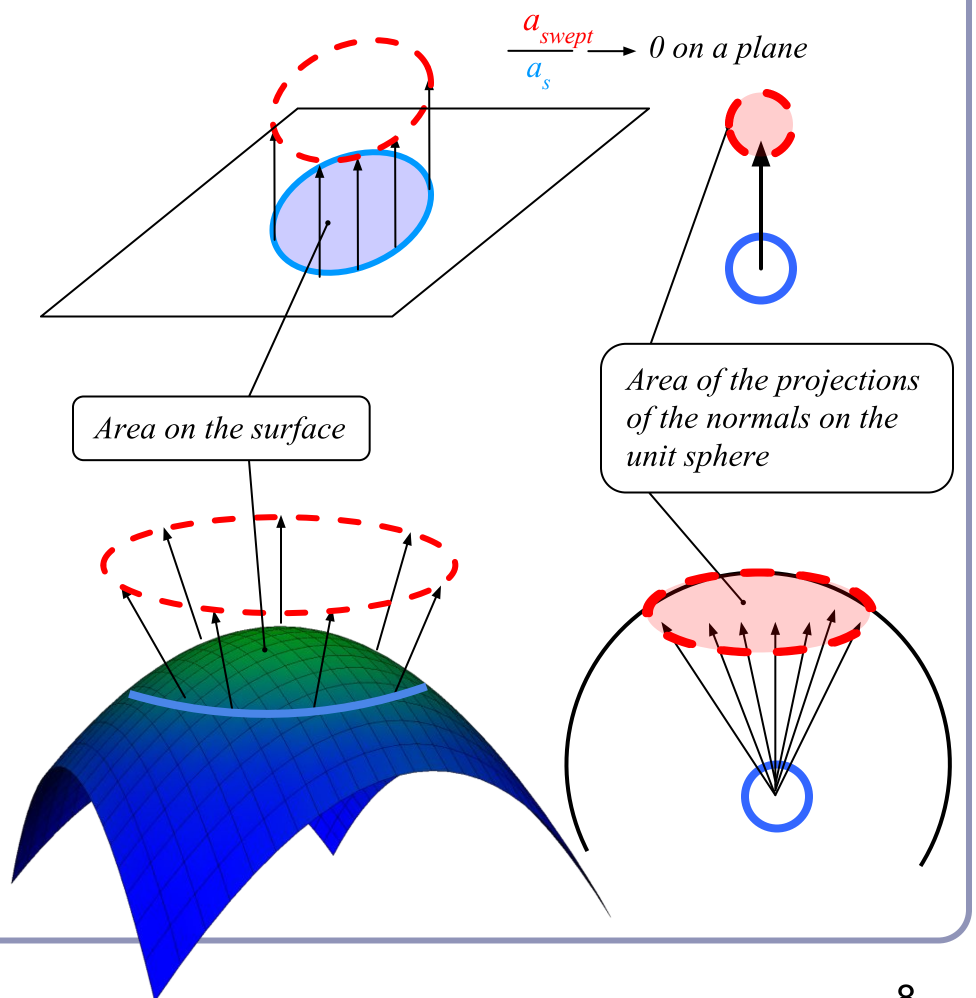


Image by Eric Gaba, from Wikipedia

Gaussian curvature on smooth surfaces

Formally, the *Gaussian curvature of a region* on a surface is the ratio between the area of the surface of the unit sphere swept out by the normals of that region and the area of the region itself.

The Gaussian curvature of a point is the limit of this ratio as the region tends to zero area.



$$\frac{a_{swept}}{a_s} \to r^{-2} \text{ on a sphere of radius } r$$

(please pretend that this is a sphere)

Gaussian curvature on discrete surfaces

On a discrete surface, normals do not vary smoothly: the normal to a face is constant on the face, and at edges and vertices the normal is—strictly speaking—undefined.

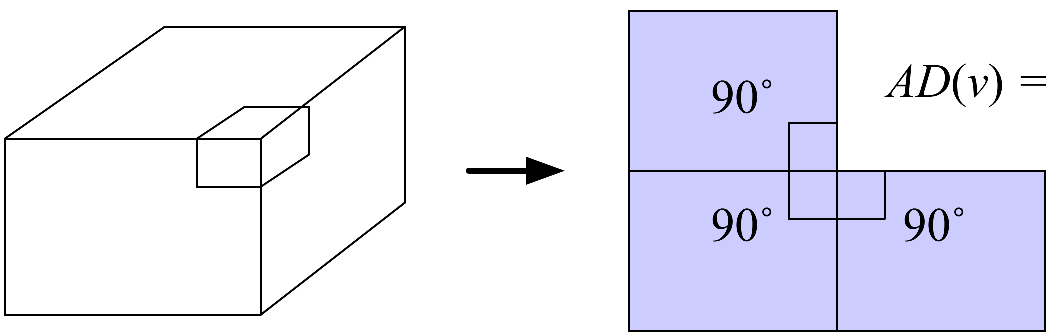
- Normals change instantaneously (as one's point of view travels across an edge from one face to another) or not at all (as one's point of view travels within a face.)

The Gaussian curvature of the surface of any polyhedral mesh is **zero** everywhere except at the vertices, where it is **infinite**.

Angle deficit – a better solution for measuring discrete curvature

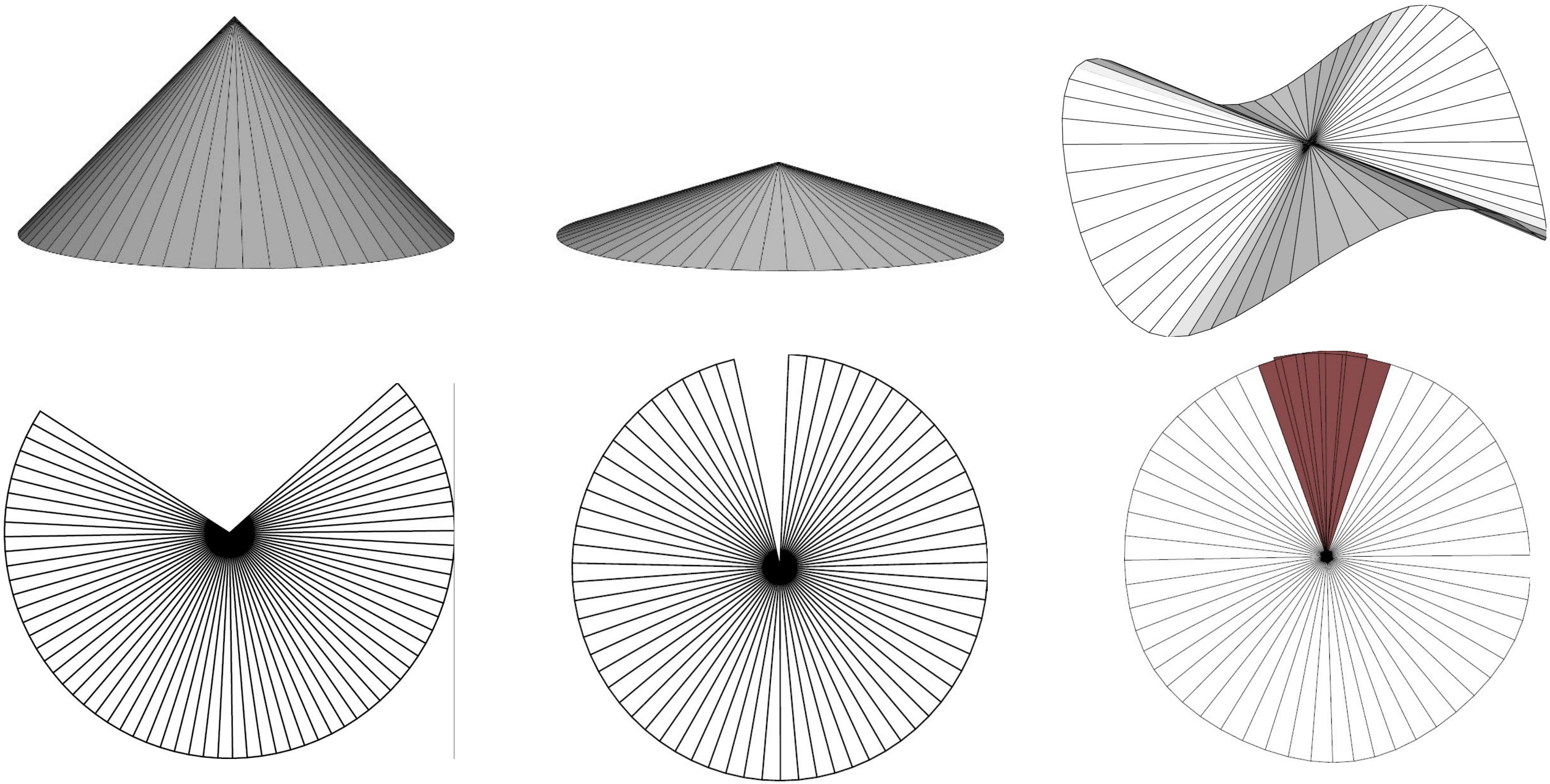
The *angle deficit AD(v)* of a vertex *v* is defined to be two π minus the sum of the face angles of the adjacent faces.

$$AD(v) = 2\pi - \sum_F \alpha(F, v)$$



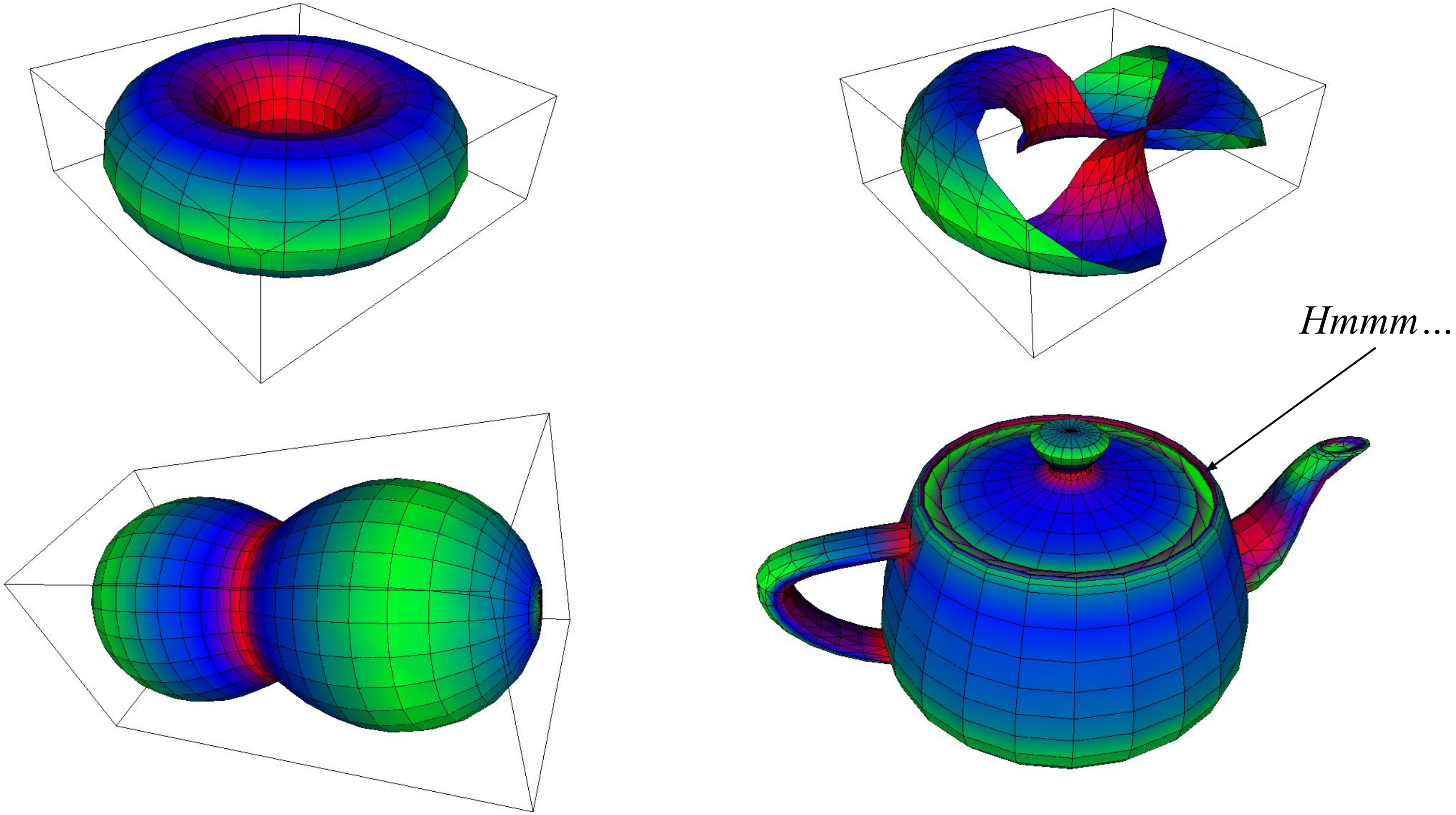
$AD(v) = 360° – 270° = 90°$

Angle deficit



High angle deficit | Low angle deficit | Negative angle deficit

Angle deficit



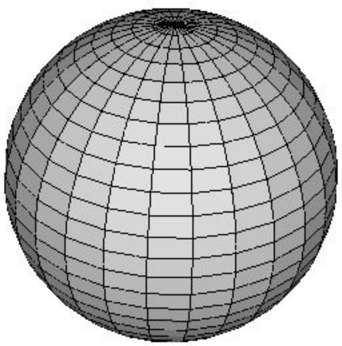
Genus, Poincaré and the Euler Characteristic

- Formally, the *genus g* of a closed surface is

  …"a topologically invariant property of a surface defined as the largest number of nonintersecting simple closed curves that can be drawn on the surface without separating it."

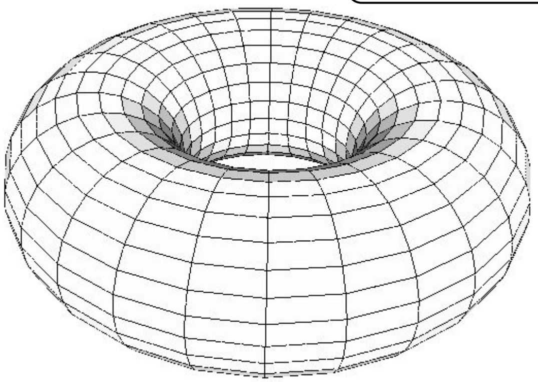
  --mathworld.com

- Informally, it's the number of coffee cup handles in the surface.



Genus 0

Genus 1



Genus, Poincaré and the Euler Characteristic

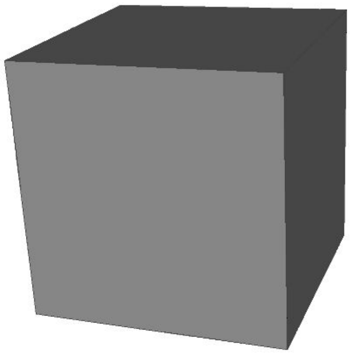
Given a polyhedral surface S without border where:

- V = the number of vertices of S,
- E = the number of edges between those vertices,
- F = the number of faces between those edges,
- χ is the *Euler Characteristic* of the surface,

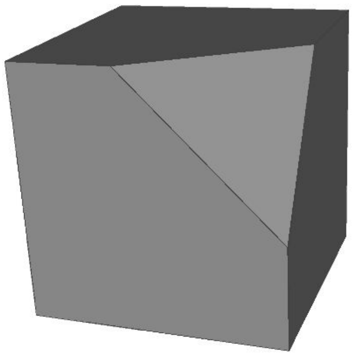
the Poincaré Formula states that:

$$V - E + F = 2 - 2g = \chi$$

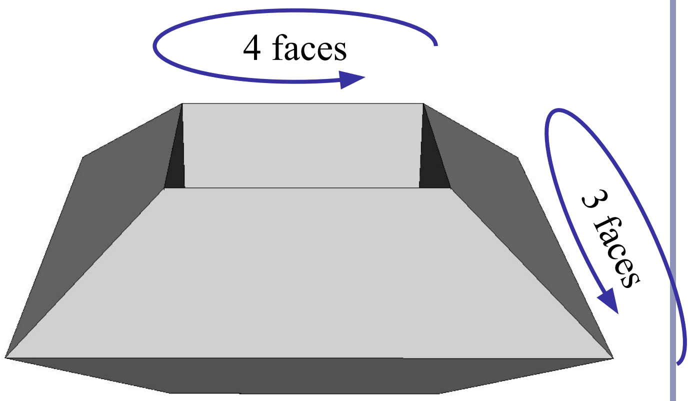
Genus, Poincaré and the Euler Characteristic



$g = 0$
$E = 12$
$F = 6$
$V = 8$
$\underline{V\text{-}E+F = 2\text{-}2g = 2}$



$g = 0$
$E = 15$
$F = 7$
$V = 10$
$\underline{V\text{-}E+F = 2\text{-}2g = 2}$



$g = 1$
$E = 24$
$F = 12$
$V = 12$
$\underline{V\text{-}E+F = 2\text{-}2g = 0}$

The Euler Characteristic and angle deficit

Descartes' *Theorem of Total Angle Deficit* states that on a surface S with Euler characteristic χ, the sum of the angle deficits of the vertices is $2\pi\chi$:

$$\sum_S AD(v) = 2\pi\chi$$

Cube:

- $\chi = 2-2g = 2$
- $AD(v) = \pi/2$
- $8(\pi/2) = 4\pi = 2\pi\chi$

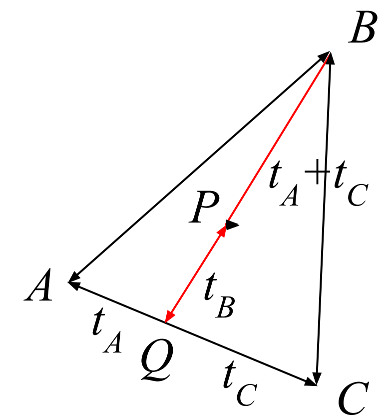
Tetrahedron:

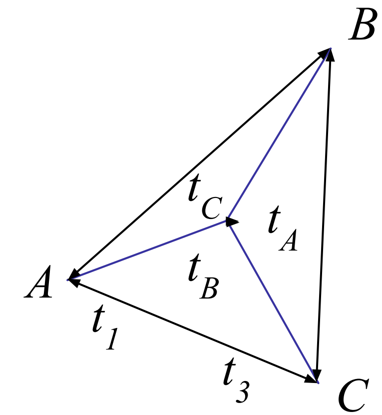
- $\chi = 2-2g = 2$
- $AD(v) = \pi$
- $4(\pi) = 4\pi = 2\pi\chi$

Barycentric coordinates

Barycentric coordinates (t_A,t_B,t_C) are a coordinate system for describing the location of a point P inside a triangle (A,B,C).

- You can think of (t_A,t_B,t_C) as 'masses' placed at (A,B,C) respectively so that the center of gravity of the triangle lies at P.
- (t_A,t_B,t_C) are proportional to the subtriangle areas of the three vertices.
  - The area of a triangle is ½ the length of the cross product of two of its sides.





Barycentric coordinates

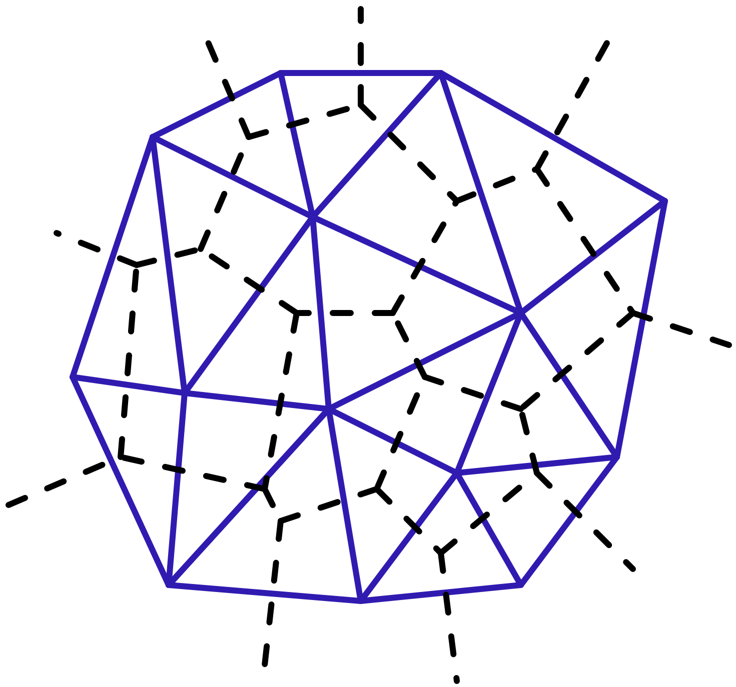
```
// Compute barycentric coordinates (u, v, w) for
// point p with respect to triangle (a, b, c)
vec3 barycentric(vec3 p, vec3 a, vec3 b, vec3 c) {
  vec3 v0 = b - a, v1 = c - a, v2 = p - a;
  float d00 = dot(v0, v0);
  float d01 = dot(v0, v1);
  float d11 = dot(v1, v1);
  float d20 = dot(v2, v0);
  float d21 = dot(v2, v1);
  float denom = d00 * d11 - d01 * d01;
  float v = (d11 * d20 - d01 * d21) / denom;
  float w = (d00 * d21 - d01 * d20) / denom;
  float u = 1.0 - v - w;
  return vec3(u, v, w);
}
```

Code credit: Christer Ericson, *Real-Time Collision Detection* (2004)
(adapted to GLSL for this lecture)

Voronoi diagrams

The *Voronoi diagram*[2] of a set of points P_i divides space into 'cells', where each cell C_i contains the points in space closer to P_i than any other P_j.

The *Delaunay triangulation* is the dual of the Voronoi diagram: a graph in which an edge connects every P_i which share a common edge in the Voronoi diagram.



A Voronoi diagram (dotted lines) and its dual Delaunay triangulation (solid).

[2] AKA "Voronoi tesselation", "Dirichelet domain", "Thiessen polygons", "plesiohedra", "fundamental areas", "domain of action"…

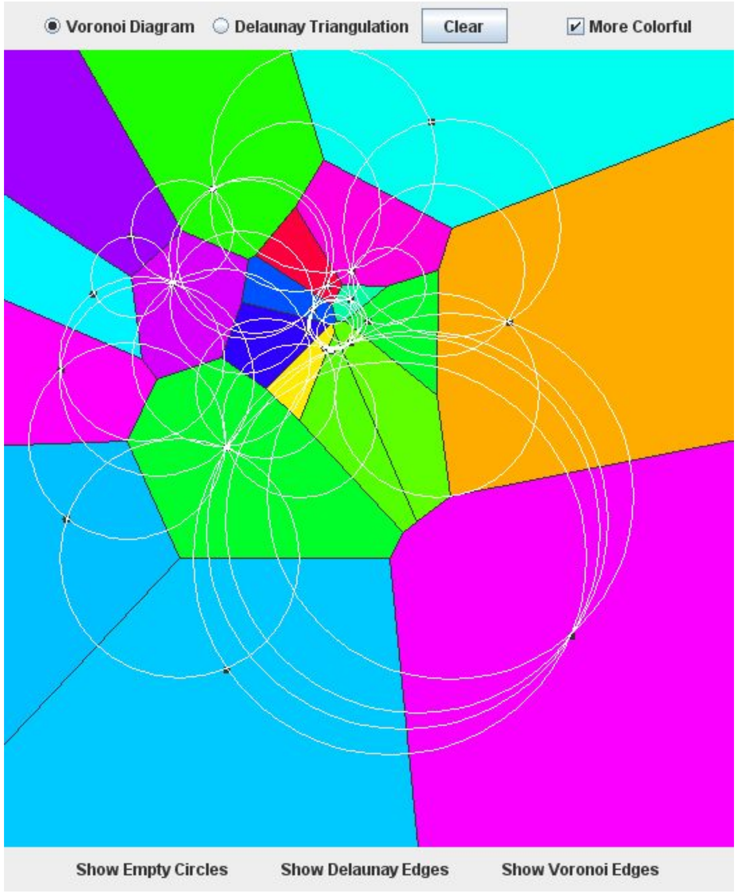
Voronoi diagrams

Given a set $S=\{p_1,p_2,\ldots,p_n\}$, the formal definition of a Voronoi cell $C(S,p_i)$ is

$$C(S,p_i)=\{p \in R^d \mid |p-p_i|<|p-p_j|,\ i\neq j\}$$

The p_i are called the *generating points* of the diagram.

Where three or more boundary edges meet is a *Voronoi point*. Each Voronoi point is at the center of a circle (or sphere, or hypersphere…) which passes through the associated generating points and which is guaranteed to be empty of all other generating points.



Delaunay triangulation applet by Paul Chew ©1997—2007
http://www.cs.cornell.edu/home/chew/Delaunay.html

Delaunay triangulations and *equi-angularity*

The *equiangularity* of any triangulation of a set of points S is a sorted list of the angles $(\alpha_1 \ldots \alpha_{3t})$ of the triangles.

- A triangulation is said to be *equiangular* if it possesses lexicographically largest equiangularity amongst all possible triangulations of S.
- The Delaunay triangulation is equiangular.

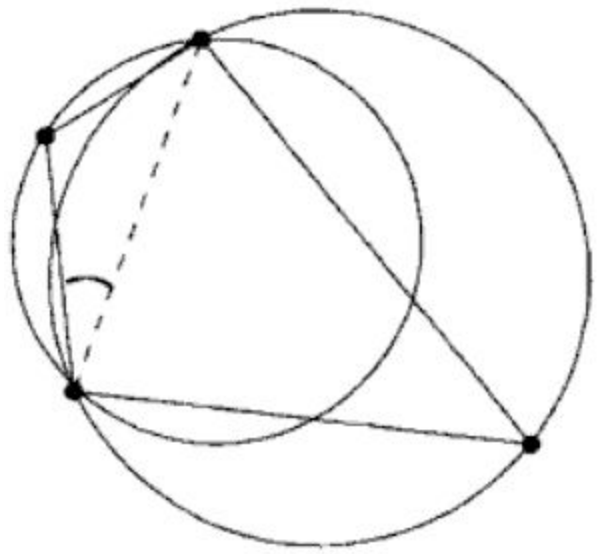


Image from *Handbook of Computational Geometry* (2000) Jörg-Rüdiger Sack and Jorge Urrutia, p. 227

Delaunay triangulations and *empty circles*

Voronoi triangulations have the *empty circle* property: in any Voronoi triangulation of *S*, no point of *S* will lie inside the circle circumscribing any three points sharing a triangle in the Voronoi diagram.

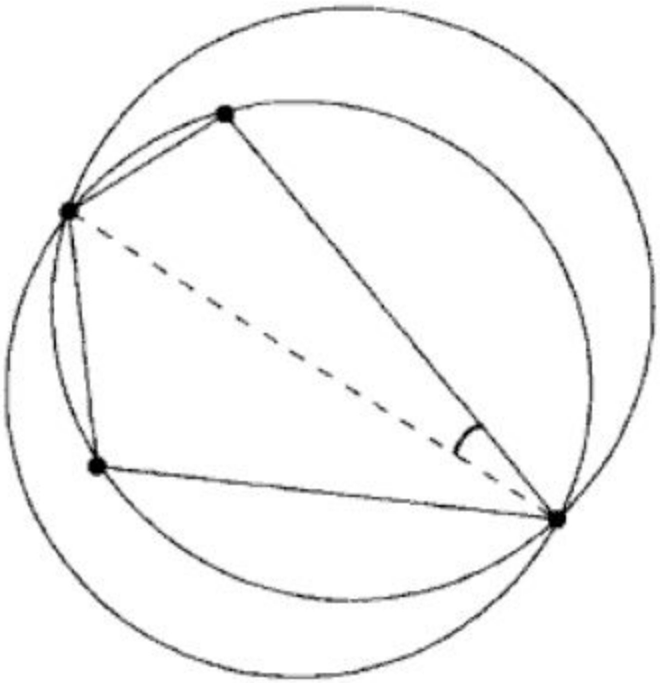


Image from *Handbook of Computational Geometry* (2000) Jörg-Rüdiger Sack and Jorge Urrutia, p. 227

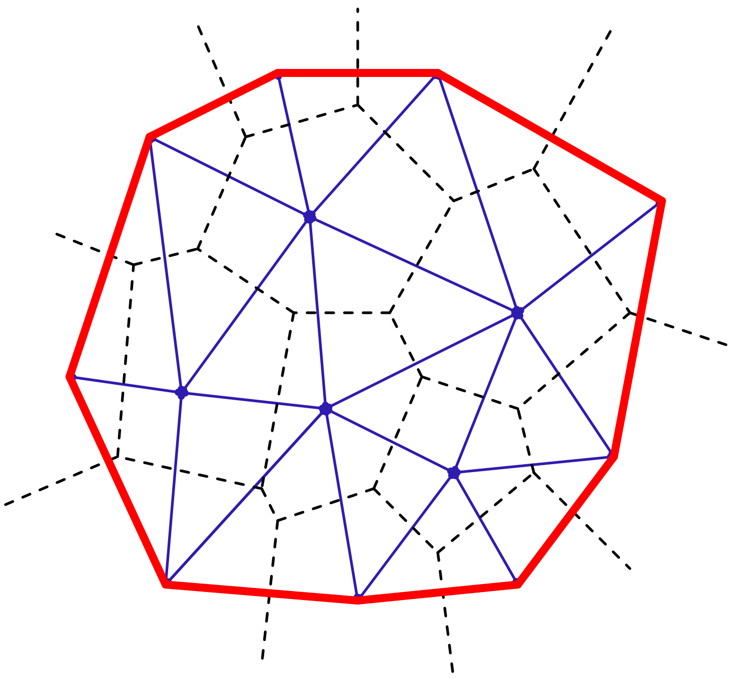
Delaunay triangulations and convex hulls

The border of the Delaunay triangulation of a set of points is always convex.

- This is true in 2D, 3D, 4D…

The Delaunay triangulation of a set of points in R^n is the planar projection of a convex hull in R^{n+1}.

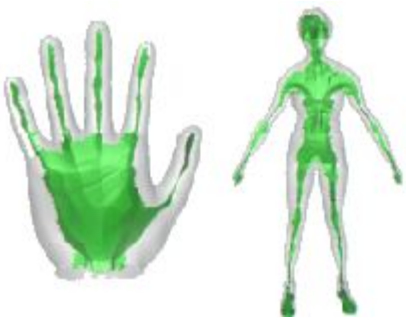
- Ex: from 2D ($P_i=\{x,y\}_i$), loft the points upwards, onto a parabola in 3D ($P'_i=\{x,y,x^2+y^2\}_i$). The resulting polyhedral mesh will still be convex in 3D.



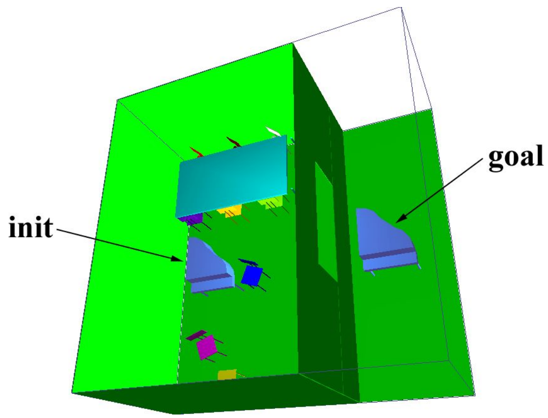
Voronoi diagrams and the *medial axis*

The *medial axis* of a surface is the set of all points within the surface equidistant to the two or more nearest points on the surface.

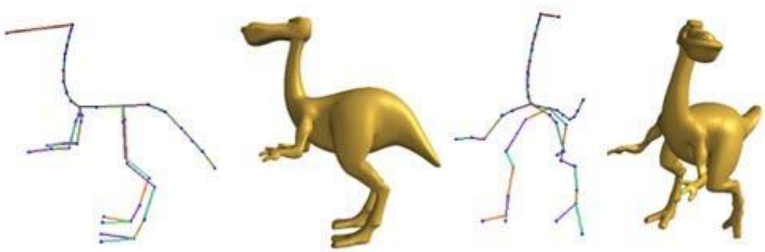
- This can be used to extract a skeleton of the surface, for (for example) path-planning solutions, surface deformation, and animation.



Approximating the Medial Axis from the Voronoi Diagram with a Convergence Guarantee
Tamal K. Dey, Wulue Zhao



A Voronoi-Based Hybrid Motion Planner for Rigid Bodies
M Foskey, M Garber, M Lin, DManocha

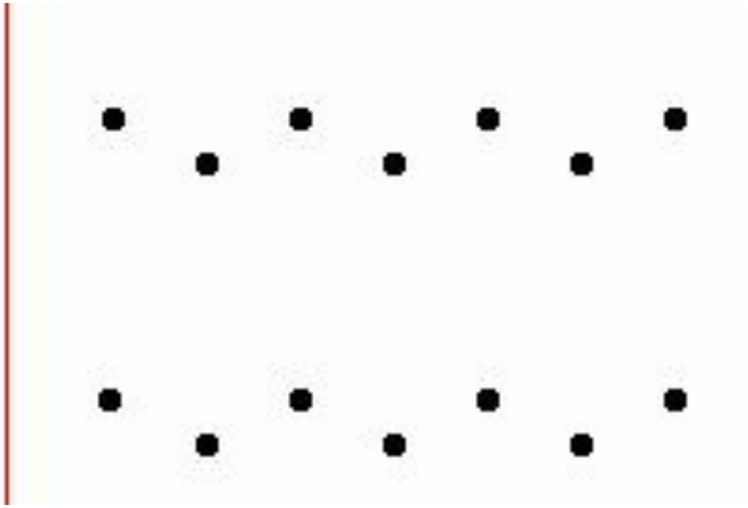


Shape Deformation using a Skeleton to Drive Simplex Transformations
IEEE Transaction on Visualization and Computer Graphics, Vol. 14, No. 3, May/June 2008, Page 693-706
Han-Bing Yan, Shi-Min Hu, Ralph R Martin, and Yong-Liang Yang

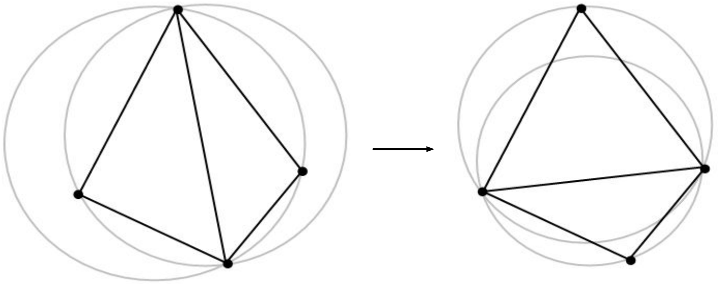
Finding the Voronoi diagram

There are four general classes of algorithm for computing the Delaunay triangulation:

- Divide-and-conquer
- Sweep plane
  - Ex: Fortune's algorithm →
- Incremental insertion
- "Flipping": repairing an existing triangulation until it becomes Delaunay



Fortune's Algorithm for the plane-sweep construction of the Voronoi diagram (Steve Fortune, 1986.)



This triangulation fails the circumcircle definition; we flip its inner edge and it becomes Delaunay. *(Image from Wikipedia.)*

Fortune’s algorithm

1. The algorithm maintains a sweep line and a “beach line”, a set of parabolas advancing left-to-right from each point. The beach line is the union of these parabolas.
   a. The intersection of each pair of parabolas is an edge of the voronoi diagram
   b. All data to the left of the beach line is “known”; nothing to the right can change it
   c. The beach line is stored in a binary tree
2. Maintain a queue of two classes of event: the addition of, or removal of, a parabola
3. There are O(n) such events, so Fortune’s algorithm is O(n log n)

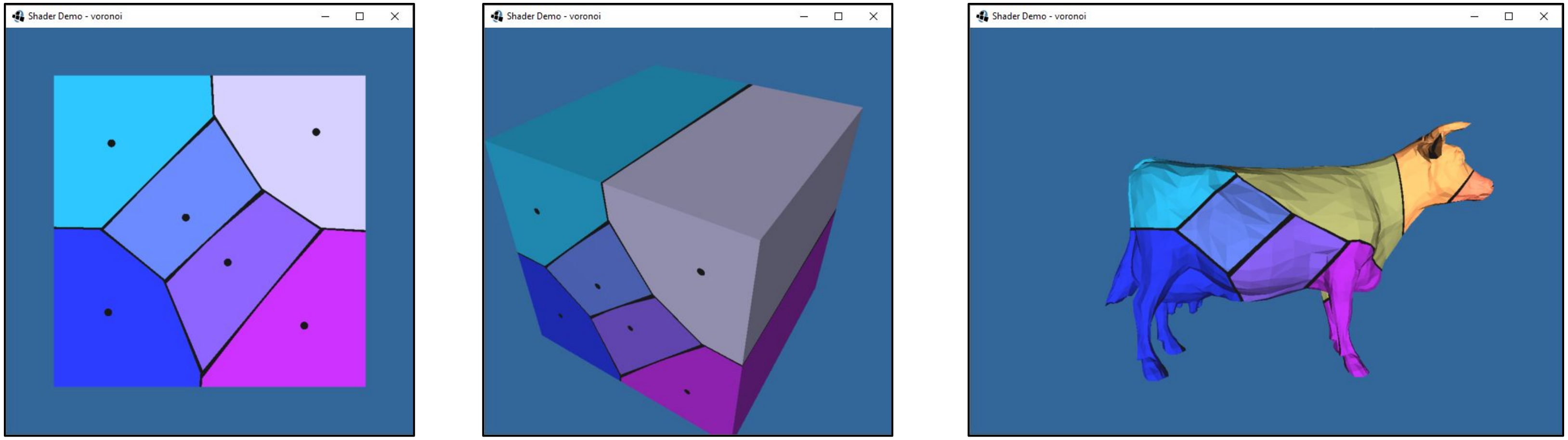
GPU-accelerated Voronoi Diagrams

Brute force:

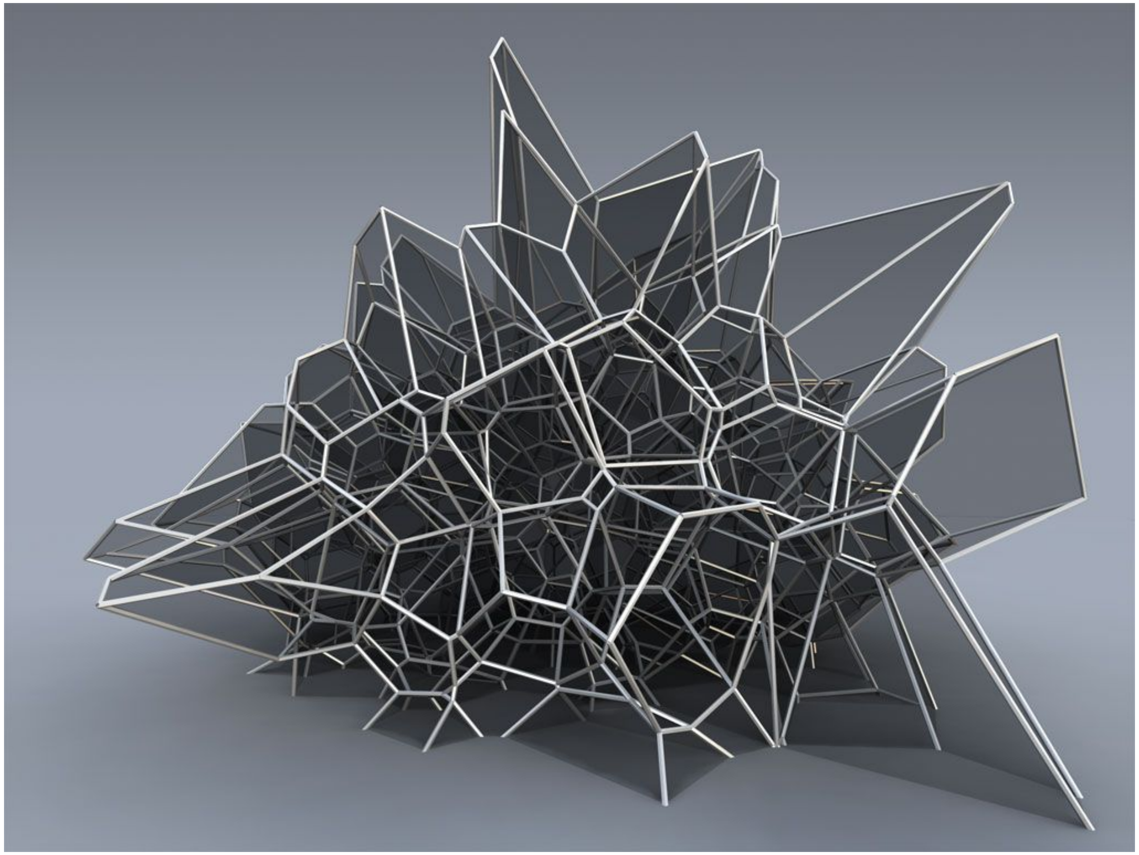
- For each pixel to be rendered on the GPU, search all points for the nearest point

Elegant (and 2D only):

- Render each point as a discrete 3D cone in isometric projection, let z-buffering sort it out



Voronoi cells in 3D



Silvan Oesterle, Michael Knauss

References

Gaussian Curvature:

http://en.wikipedia.org/wiki/Gaussian_curvature

http://mathworld.wolfram.com/GaussianCurvature.html

The Poincaré Formula:

http://mathworld.wolfram.com/PoincareFormula.html

Jordan curves:

R. Courant, H. Robbins, *What is Mathematics?*, Oxford University Press, 1941

http://cgm.cs.mcgill.ca/~godfried/teaching/cg-projects/97/Octavian/compgeom.html

Voronoi diagrams:

M. de Berg, O. Cheong, M. van Kreveld, M. Overmars, "*Computational Geometry: Algorithms and Applications*", Springer-Verlag, http://www.cs.uu.nl/geobook/

http://www.ics.uci.edu/~eppstein/junkyard/nn.html

http://www.iquilezles.org/www/articles/voronoilines/voronoilines.htm