Announcement

- Homework #5 Due June 14 (Midnight)
  - Late submission allowed only upto *June 21*.

today

	2 Sampling	Chap 16
	7 Samplling/Reconstruction	Chap 16/17/18
	8 Open Lab	
	9 Geometirc modeling	Chap 22
	14 Animation	Chap 23
	21 Final Exam	

- Final Exam
  - Tuesday June 21 4PM E3-1, #1501
  - Reading: all chapters covered - omit 19 (Color)

Contest

<optional but highly-recommended!>

- Best image of a virtual floppy cube
  - Created by you
  - 512 x 512 (any standard image format is fine)
  - High quality rendered output
  - Title & short description will be nice to have
  - Enter by June 15 Midnight to KLMS
  - **Winner(s)** will be announced at the final on June 21 and receive a prize!

Geometric Modeling
(Basic Intro)

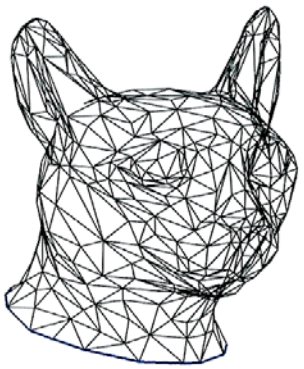
Chapter 22

Modeling Approach

- Modeling
  - The spatial description and placement of imaginary 3D objects, environments and scene with a computer system.
- Models are abstractions of the world
- In computer graphics, we model our worlds with *geometric objects*.
  - Which primitive to use in our models?
  - How to show relationships among them?
- Modeling approaches
  - Constructing from primitives (Hierarchical approach)
  - Procedural modeling (Language-based)
  - Analytic description (Curves and Surfaces)
  - Scanning and Reconstruction (Volumetric data sets)
  - Interactive modeling (Sketch-based, Modeling tools)

Triangle soup

- Triangle soup is the most popular.
- Quad soup (four-sided polygons) and polygon soup (general-sided polygons) are also used for representations.
  - For hardware rendering such polygons first need to be diced up into general triangles.

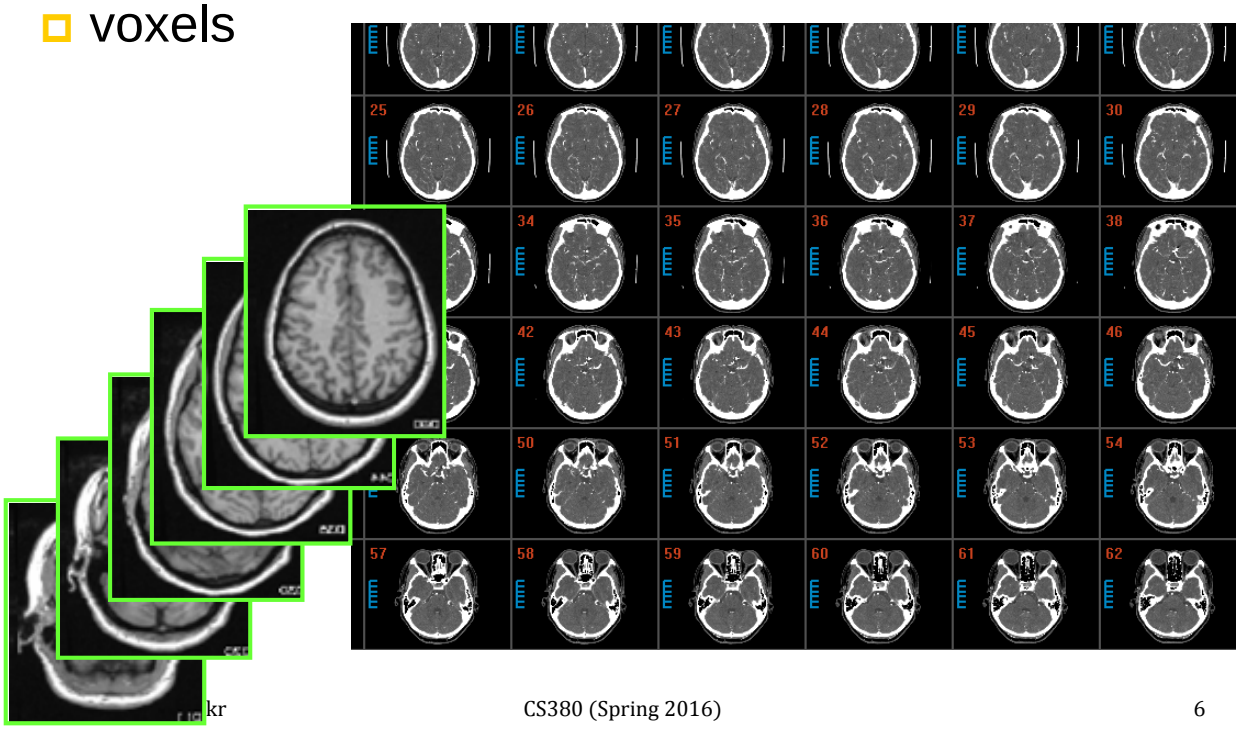


Meshes

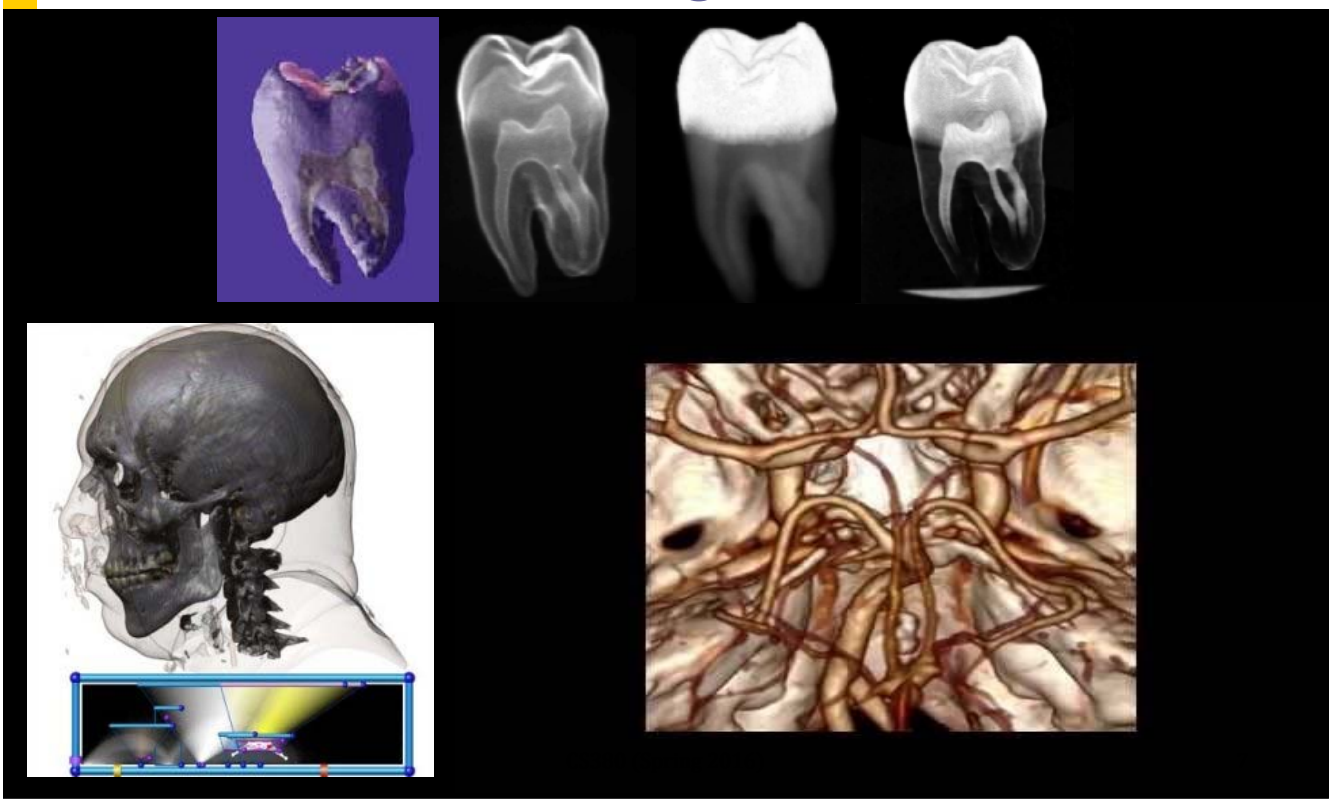
- Mesh data
  - Organizes the vertex, edge, and face data

Volumetric data

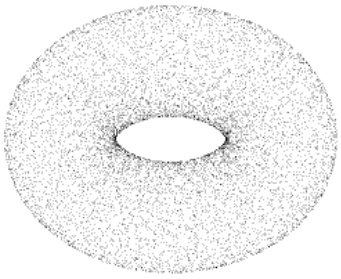
- voxels



Volume rendering

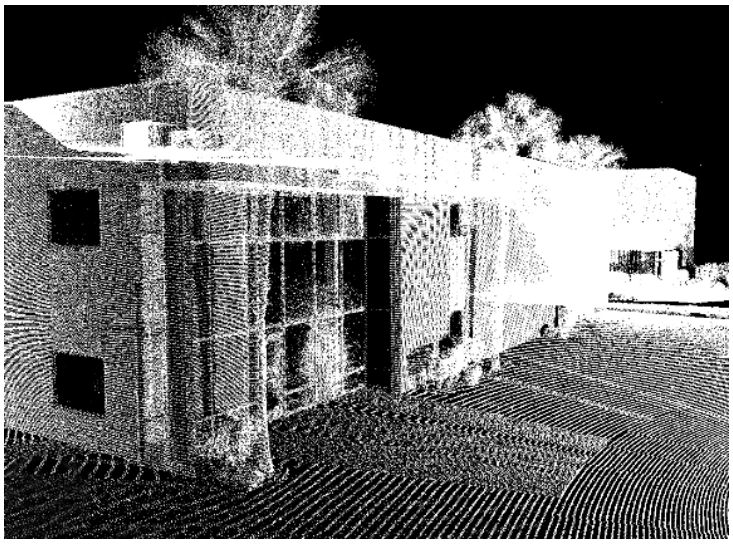


Particles / point cloud



https://en.wikipedia.org/wiki/Point_cloud

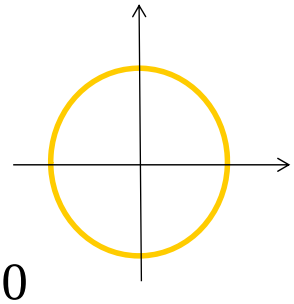
http://www.revittunes.com/gallerybg.php?img=revit-Leica_Point_Cloud_Scanning.jpg&t=0&publici=18



Representations

- Explicit functions: y=f(x)
  - Curve in 3D: y=f(x), z=g(x)
  - Surface in 3D: z=f(x,y)

$$y = \pm\sqrt{1-x^2}$$

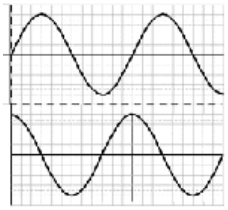


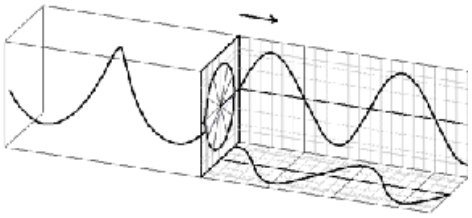
- Implicit equations: f(x,y)=0
  - Surface in 3D: f(x,y,z) = 0

$$x^2 + y^2 - 1 = 0$$

- Parametric equations: x=f(t), y=g(t)

$$x(t) = r\cos(t)$$
$$y(t) = r\sin(t)$$





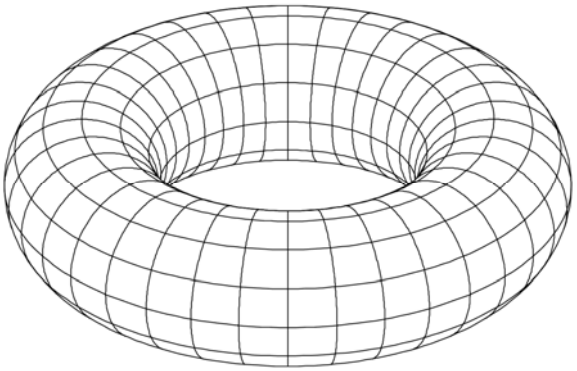
WIKIPEDIA
The Free Encyclopedia

Representations

- Explicit functions
  - Only one value of y for each x
  - Difficult to represent a slope of infinity
- Implicit functions
  - Need constraints to model just one part of a curve
  - Joining curves together smoothly is difficult
- Parametric equations
  - Slopes represented as parametric tangent vector (d/dt)
  - Easy to join curve segments smoothly

Implicit Surfaces

- A plane: $x + 2y - 3z + 1 = 0$
- A sphere: $x^2 + y^2 + z^2 - 4 = 0$
- A torus: $(x^2 + y^2 + z^2 - R^2 - a^2)^2 - 4R^2(x^2 + y^2) = 0$

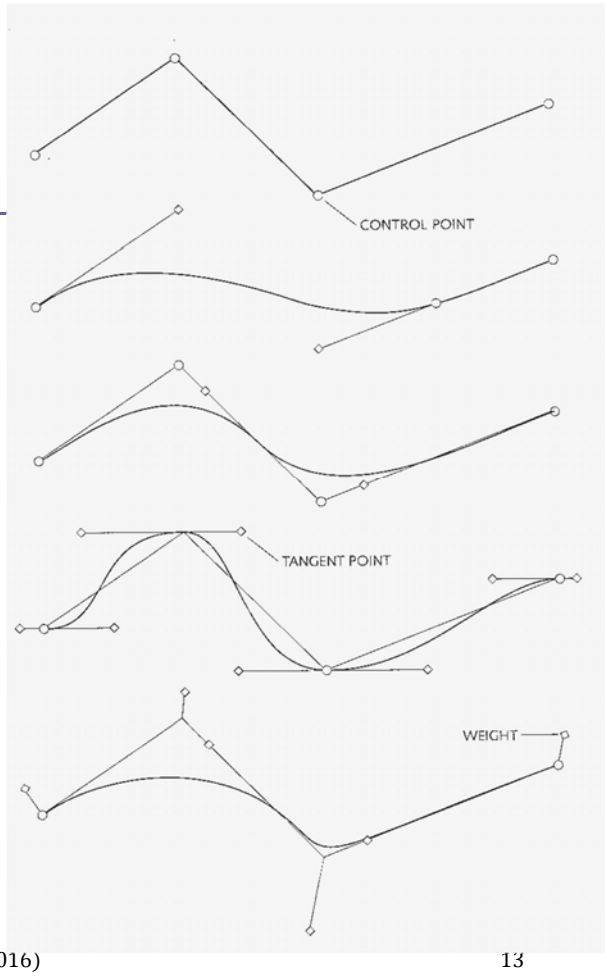


Parameterization

- Parameterizations are in general not unique.
  - $L(p_0,p_1) = p_0 + u\,(p_1-p_0)$ u=[0…1]
  - $L(p_0,p_1) = v(p_0-p_1)/2 + (p_1+p_0)/2$ v=[-1…1]
- Parameterizations can be changed to lie between desired bounds.
  - Reparameterize from u=[a…b] to w=[0…1]
    - Use w=(u-a)/(b-a), which gives u=w(b-a)+a.
    - Thus, we have p(u), u=[a…b]
      p(w(b-a)+a), w=[0…1]

Spline curves

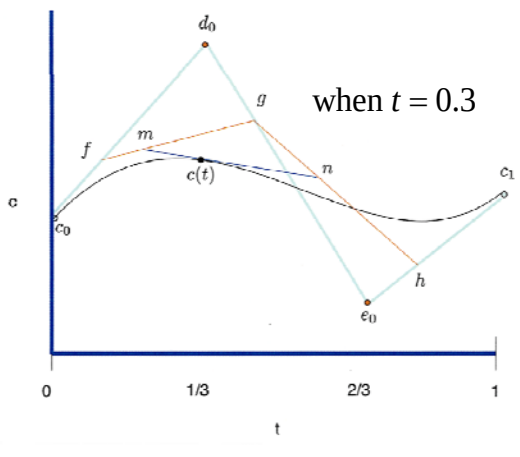
- Cardinal spline
  (a cubic Hermite spline)
  - Looks like a curve that passes through all of its control points
- …
- B-spline
  - Looks like a curved line that rarely passes through the control points
- Bezier curve
  - Passes through all of its control points
- NURBS
  - Non-uniform rational b-spline
  - Does not pass through its control points



Cubic Bezier Splines in X, Y, Z

- To evaluate the function $c(t)$ at any value of t, we perform the following sequence of linear interpolations:

$$f = (1-t)c_0 + td_0$$
$$g = (1-t)d_0 + te_0$$
$$h = (1-t)e_0 + tc_0$$
$$m = (1-t)f + tg$$
$$n = (1-t)g + th$$
$$c(t) = (1-t)m + tn$$



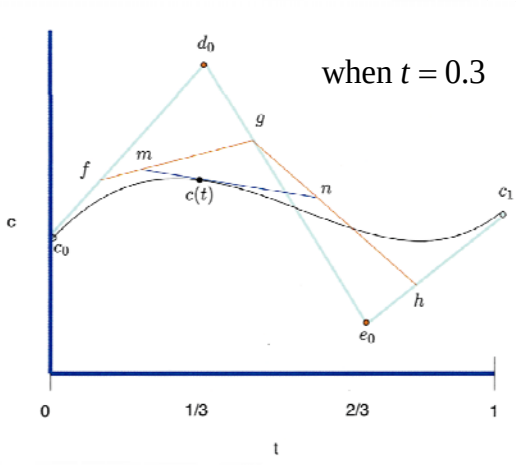
Cubic Bezier Splines in X, Y, Z

- By unwrapping the evaluation steps before, we can verify $c(t)$ has the form:

$$c(t) = c_0(1-t)^3 + 3d_0t(1-t)^2 + 3e_0t^2(1-t) + c_1t^3$$

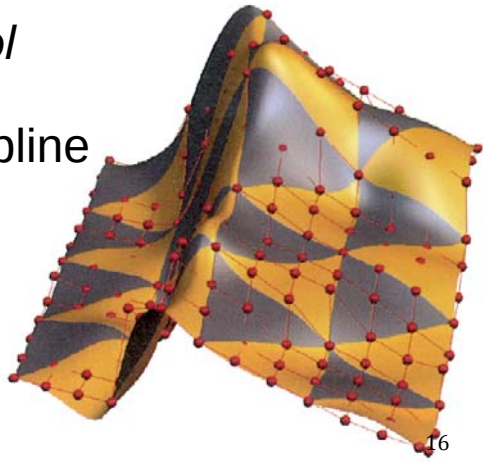
- It is a cubic function
- The c_i are interpolated:

$$c(0) = c_0 \text{ and } c(1) = c_1$$



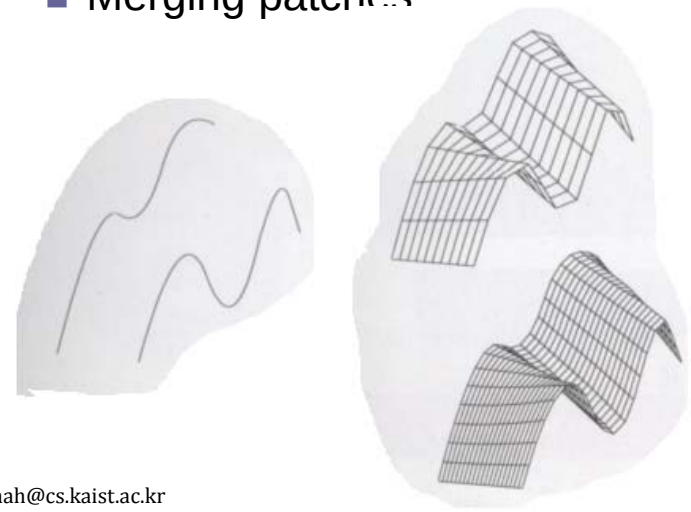
Tensor-product spline surface

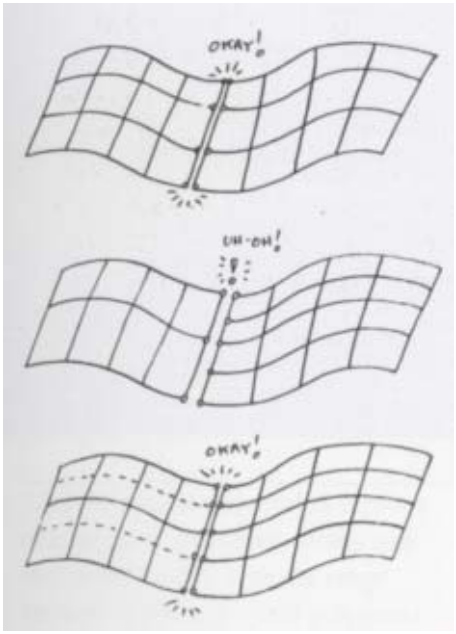
- Parametric patch uses three coordinate functions *x(s,t), y(s,t) and z(s,t).* These functions are defined over some square or triangular portions of the *(s,t)* plane.
- This construction is upgraded from curves to surfaces
- The control polygon is replaced by an *m* and *n* rectilinear *control mesh*, with vertices in 3D.
- By appropriately applying the spline definitions in both the *s* and *t* variables, we end up with a parametric patch.



Patches

- Curved patches
  - A small curved area created from 2 or 4 curves
  - Merging patches





Anjyu (Namco Limited)
Siggraph 2001

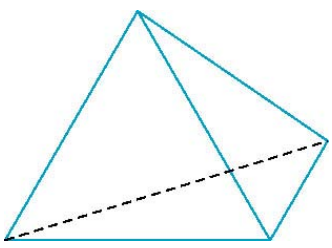
"Anjyu" refers to its composited layers of calmness.

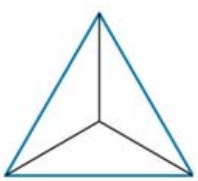
Procedural Methods

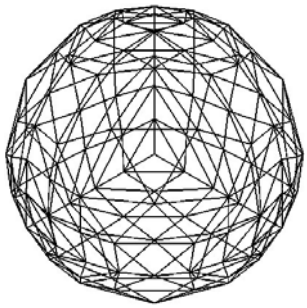
- Describe objects in an algorithmic manner and produce polygons only when needed as part of the rendering process
- Analogy
  - Irrational number (square roots, sines, cosines, etc.) representation in a computer
    - $x^2 = 2$ abstract
    - $\sqrt{2} = 1.414 \ldots$ numerically
    - Within a computer, it might be the result of executing an algorithm such as Newton's method
      $x_{k+1} = x_k/2 + 1/x_k$ starting with $x_0 = 1$

Procedural Methods

- Analogy
  - Irrational number $\sqrt{2}$
  - Sphere $x^2+y^2+z^2 = r^2$
    - Subdivision process for generating a sphere from a tetrahedron

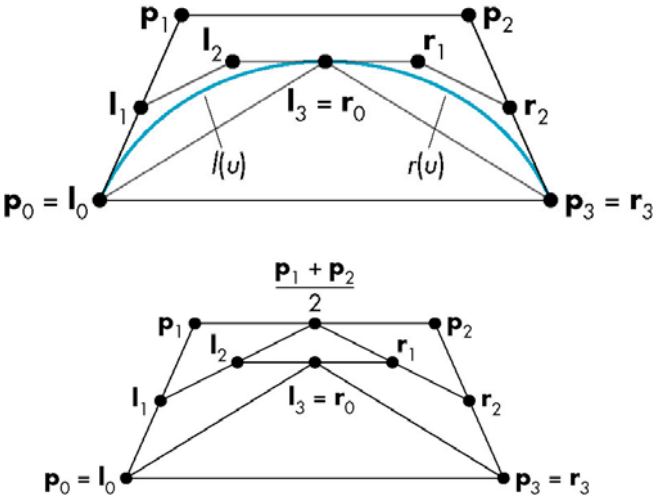


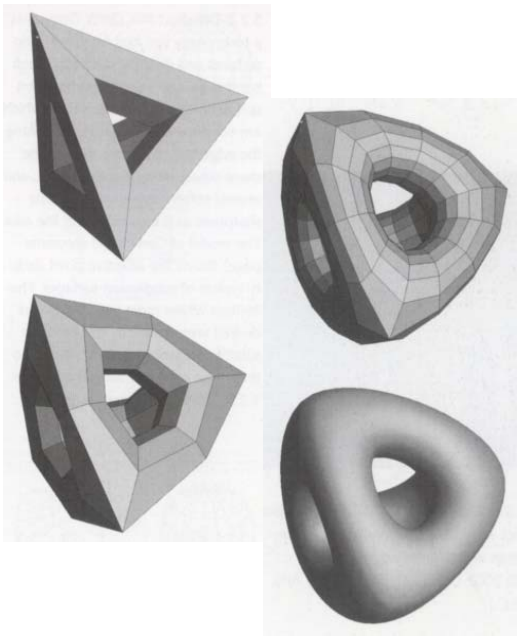


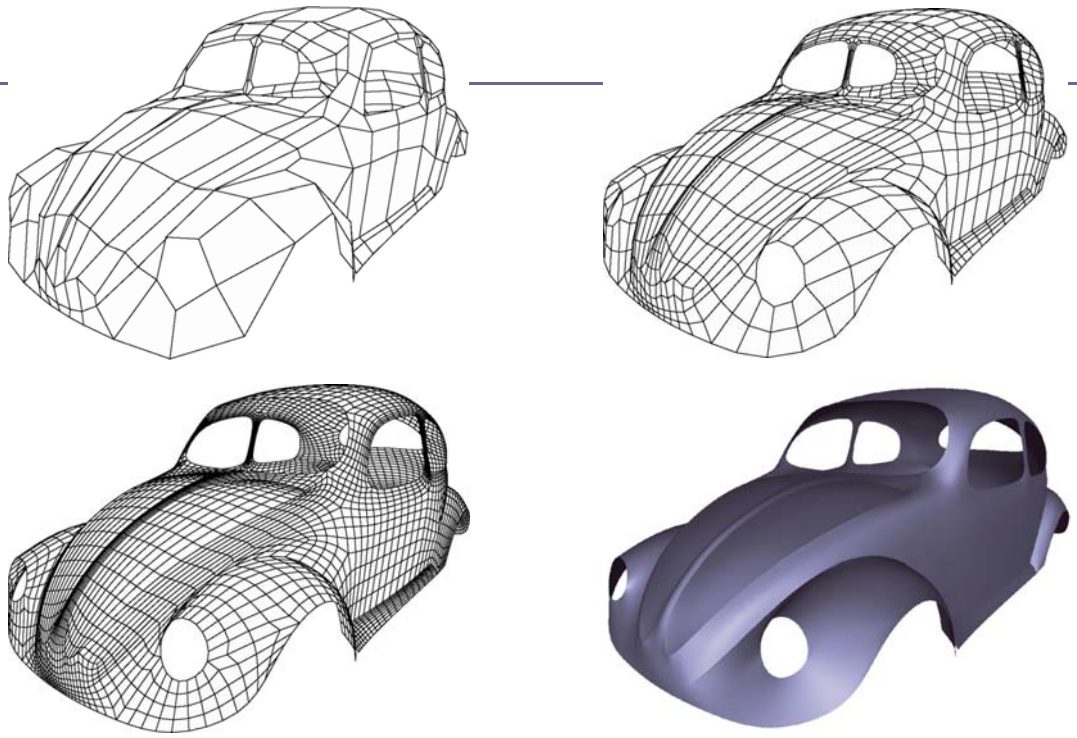


Subdivision

- Recursive subdivision of Bezier polynomials



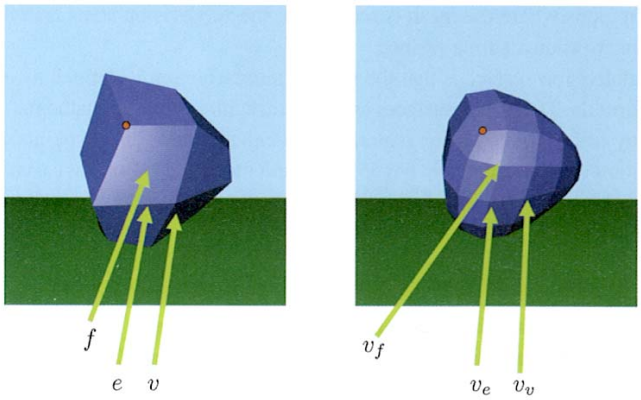




Subdivision surfaces

- Theory was generalized from tensor product B-spline surfaces.
- Allows for use of a single control mesh to describe entire geometry no patching.
- Start with a mesh
- Use rules to refine the mesh.

Original

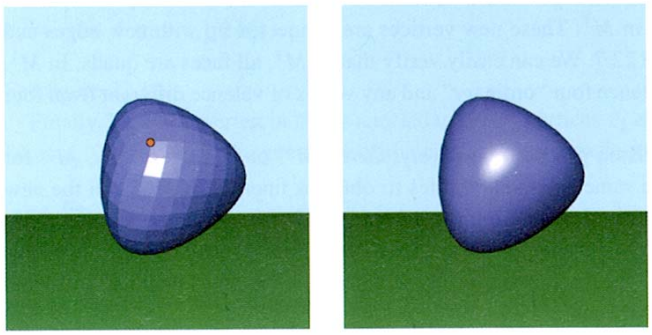


One level subdivision

Subdivision surfaces

- Surface is the infinite limit of this process, but typically only apply it a small number of times.
  - Give us a smooth surface that approximate the control mesh.
  - Special rules can be added in to get creases where desired.

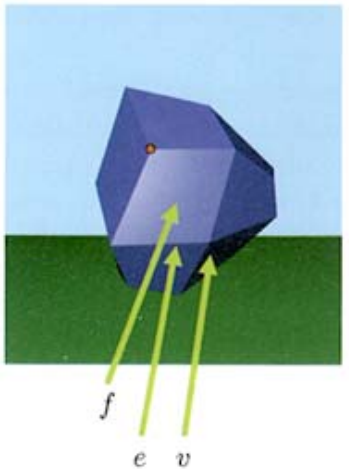
Two level subdivision

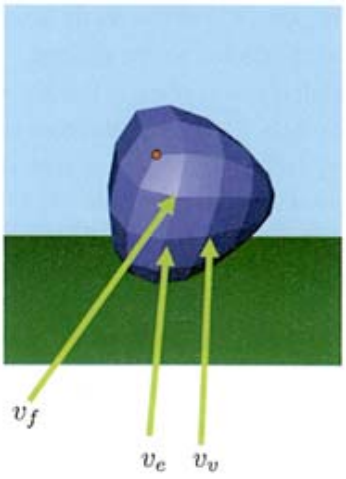


Same mesh with smooth shaded

Catmull-Clark

- Start with a mesh M^0
- Apply a set of connectivity update to get a new refined mesh M^1, with its own connectivity and geometry.
- Connectivity of M^1:
  - For each vertex v in M^0, we associate a new 'vertex-vertex' v_v in M^1.
  - For each edge e in M^0, we associate a new "edge-vertex" v_e in M^1.
  - For each face f in M^0, we associate a new "face-vertex" v_f in M^1.
  - Connected up as shown.
- In M^1, all faces are quads.
  - "ordinary" vs "extraordinary" vertices

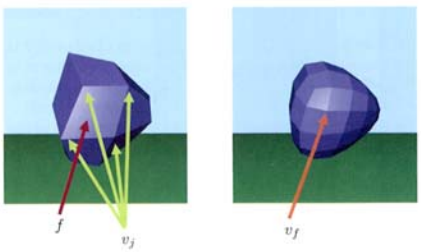




Geometry rules

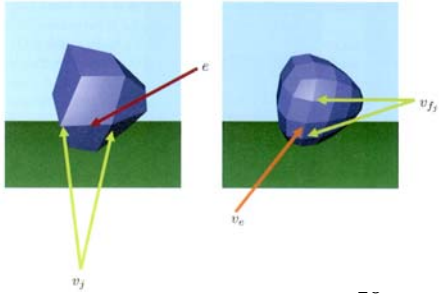
- First, let f be a face in M^i surrounded by the vertices v_j (and let m_f be the number of such vertices).
- We set the geometry of each new face-vertex v_f in M^{i+1} to be

$$(m_f = 4) \qquad v_f = \frac{1}{m_f}\sum_j v_j$$



- The centroid of the vertices in M^i defining that face.
- Edge rules: Next, let e be an edge in M^i connecting the vertices v_1 and v_2, and separating the faces f_1 and f_2. We set the geometry of the new edge-vertex in M^{i+1} *to be:*

$$v_e = \frac{1}{4}(v_1 + v_2 + v_{f_1} + v_{f_2})$$



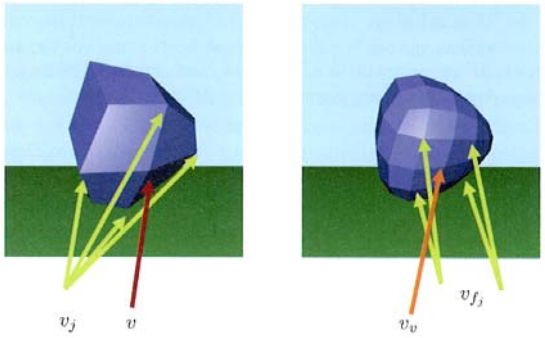
Vertex rules

- Finally let v be a vertex in M^i connected to the n_v vertices v_j and surrounded by the n_v faces f_j.
- Then, we set the geometry of the new vertex-vertex in M^{i+1} to be:

$$v_v = \frac{n_v - 2}{n_v} v + \frac{1}{n_v^2} \sum_j v_j + \frac{1}{n_v^2} \sum_j v_{f_j}$$

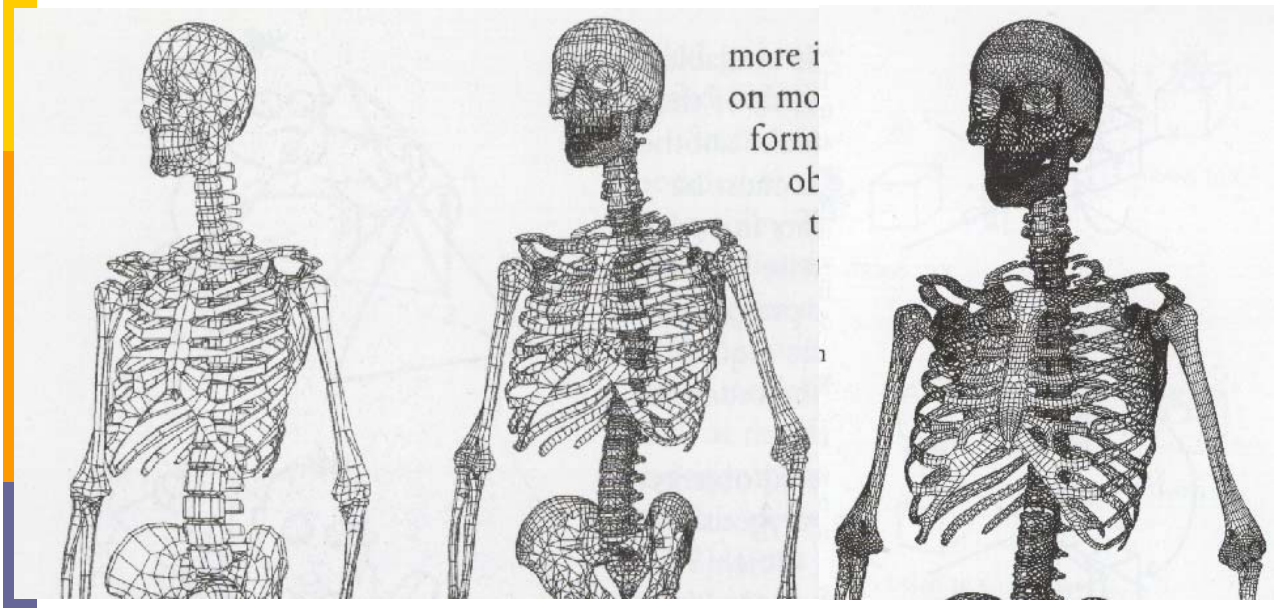
- For an ordinary vertex (i.e., $n_v=4$), this becomes:

$$v_v = \frac{1}{2} v + \frac{1}{16} \sum_j v_j + \frac{1}{16} \sum_j v_{f_j}$$



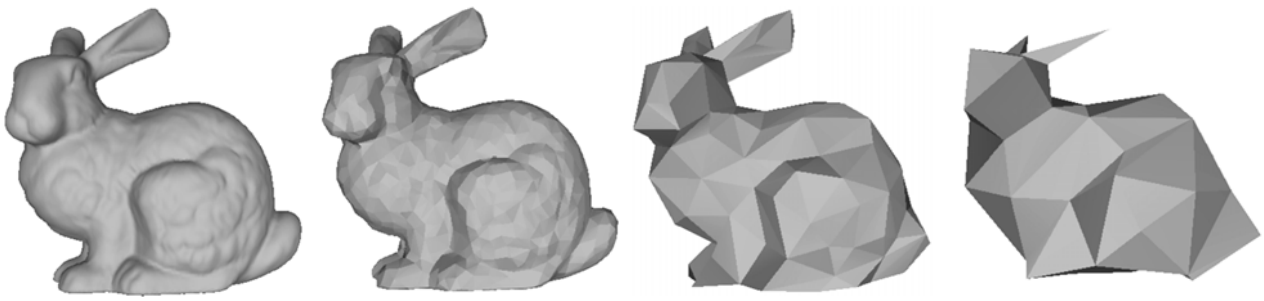
Recursive subdivision

- We can apply this subdivision process recursively.
- Given M^i, for any $i \geq 1$, we apply the same subdivision rules to obtain a finer mesh M^{i+1}
- In the new mesh, we have roughly 4 times the number of vertices.
  - The number of extraordinary vertices stays fixed.
- Thus, during subdivision, more and more of the mesh looks locally rectilinear, with a fixed number of isolated extraordinary points in between.
- Theory
  - It has been proven that this converges
  - It converges to a surface with continuous first derivative



LOD

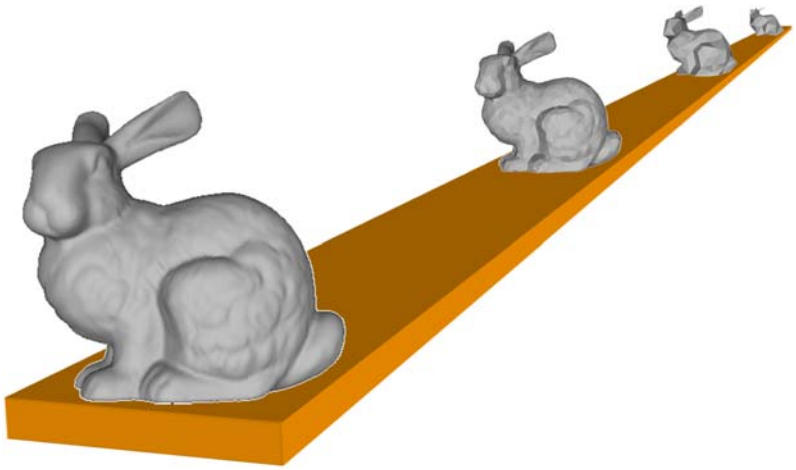
- Create *levels of detail* (*LODs*) of objects:
- Distant objects use coarser LODs:



69,451 polys 2,502 polys 251 polys 76 polys

LOD

- Create *levels of detail* (*LODs*) of objects:
- Distant objects use coarser LODs:



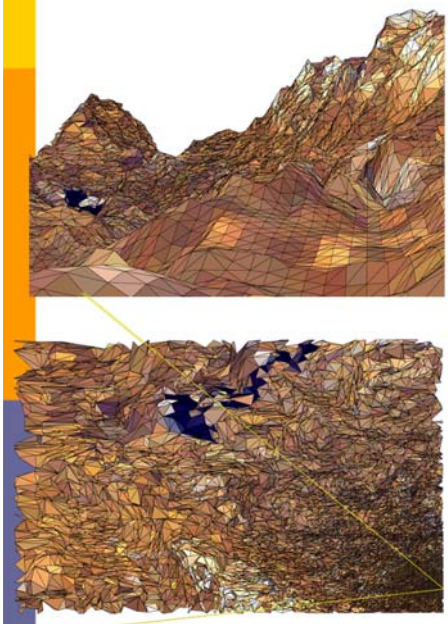
Some issues

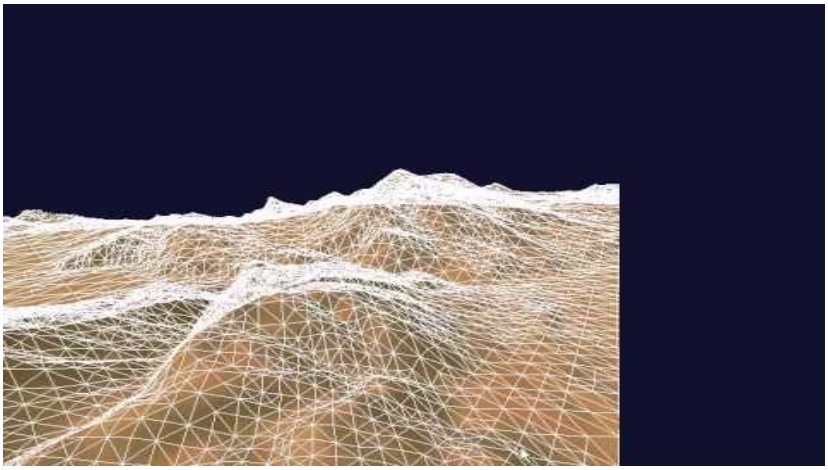
- *How to represent and generate simpler versions of a complex model?*
- *How to evaluate the fidelity of the simplified models?*
- *When to use which LOD of an object?*

- View dependent LOD
  - Show nearby portions of object at higher resolution than distant portions

View dependent LOD

View from eyepoint





ROAM

Sketching 3D Shapes

T.E.D.D.Y. https://youtu.be/e2H35SlLmUA?list=PLE223769830697780

https://www.youtube.com/watch?v=W0XGkS7zebo&list=PLE223769830697780&index=8

FiberMesh: Designing Freeform Surfaces with 3D Curves

Andrew Nealen, TU Berlin
Takeo Igarashi, The University of Tokyo / PRESTO JST
Olga Sorkine, TU Berlin
Marc Alexa, TU Berlin

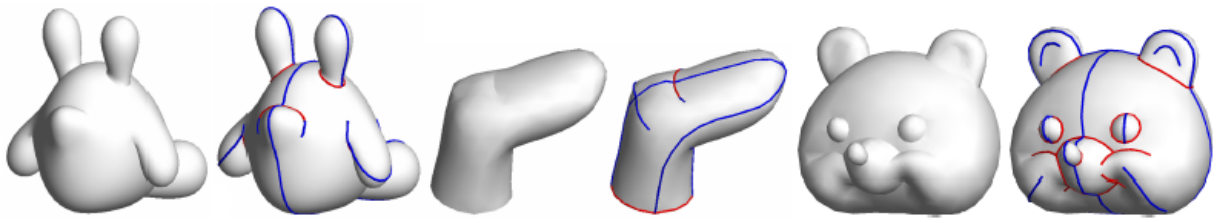


Figure 1: *Modeling results using* FIBERMESH. *The user interactively defines the control curves, combining sketching and direct manipulation, and the system continuously presents fair interpolative surfaces defined by these curves (blue = smooth curve, red = sharp curve).*