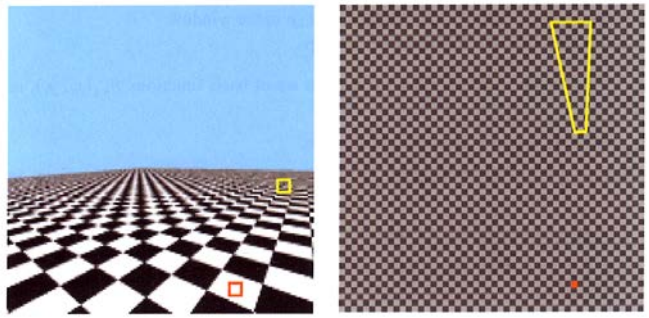
CS380: Introduction to Computer Graphics
Geometric Modeling
Chapter 22

Min H. Kim
KAIST School of Computing

Reconstruction & Resampling

SUMMARY

Resampling



Resampling equation

- In summary, when *F* is a box filter

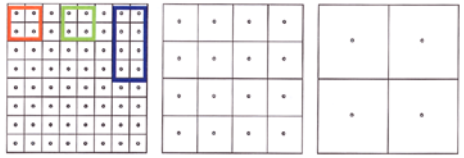
$$\mathrm{I[i][j]} \leftarrow \iint_{\Omega} \left(\sum_{k,l} B_{k,l}[M(x_w, y_w)]\mathrm{T[k][l]} \right) dx_w\, dy_w$$

$$= \iint_{M(\Omega_{i,j})} \left|\det(D_N)\right| \left(\sum_{k,l} B_{k,l}\mathrm{T[k][l]} \right) dx_t dy_t$$

- where $B_{k,l}(p)$ is how much texel *k,l* blends to obtain the value at continuous texture location *p,* from reconstruction step.

Mip mapping

- Mip mapping with trilinear interpolation is specified with the call glTexParameteri(GL TEXTURE 2D, GL TEXTURE MIN FILTER,GL LINEAR MIPMAP LINEAR)
- Trilinear interpolation requires OpenGL to fetch 8 texture pixels and blend them appropriately for every requested texture access.



Chapter 22

GEOMETRIC MODELING

Triangle soup

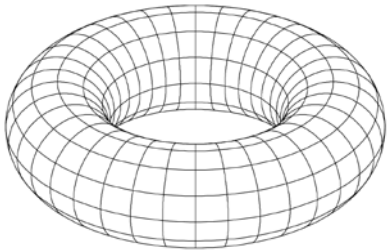
- Triangle soup is the most popular.
- Quad soup (four-sided polygons) and polygon soup (general-sided polygons) are also used for representations.
  - For hardware rendering such polygons first need to be diced up into general triangles.

Implicit Surfaces

- An implicit surface is a surface in Euclidean space, defined by an equation, e.g. $f(x,y,z)=0$
- Represent a smooth surface is as the set of points that evaluate to zero under some given trivariate function $f(x,y,z)$.
- The graph of a function is described by an equation $z=f(x,y)$ and is called an *explicit* representation.
- It can be described as a parametric representation: $(x(s,t),y(s,t),z(s,t))$, where the xyz coordinates are represented by three functions $x(s,t),y(s,t),z(s,t)$ depending on common parameters, s,t

Implicit Surfaces

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



Implicit Surfaces

- The change of representations:
  explicit representation: $z=f(x,y)$
  implicit representation: $z-f(x,y)=0$ *or* $f(x,y,z)=0$
  parametric representation: (x,y,z) *or* $(s,t,f(s,t))$
- Mesh data structure is a representation that organizes the vertex, edge and face data.
- Implicit surfaces
  - We can define an implicit function as a sum of simpler elemental implicit functions:
    $f(x,y,z)=\sum_i f_i(x,y,z)$

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:

$f=(1-t)c_0+td_0$

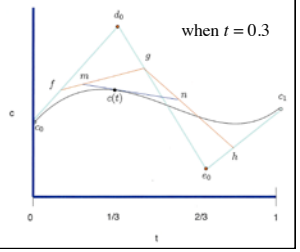
$g=(1-t)d_0+te_0$

$h=(1-t)e_0+tc_0$

$m=(1-t)f+tg$

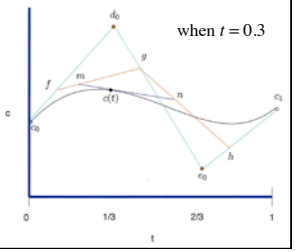
$n=(1-t)g+th$

$c(t)=(1-t)m+tn$



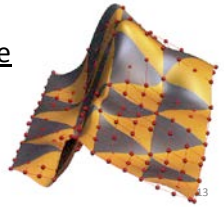
Cubic Bezier Splines in X, Y, Z

- 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:
  $c(0)=c_0$ and $c(1)=c_1$



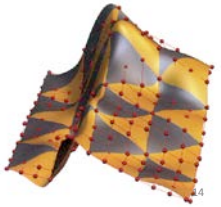
Tensor-product spline surface

- 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.



Tensor-product spline surface

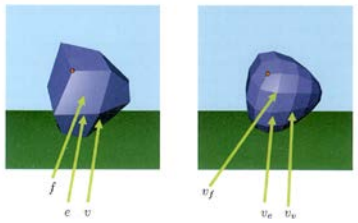
- We upgrade from a cubic spline curve construction, each square piece is defined by a bicubic polynomial in *(s,t)* domain (i.e., have terms up to the highest power: s^3t^3).
- Spline surfaces are very popular in the computer-aided design systems because of the explicit parametric polynomial representation.



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

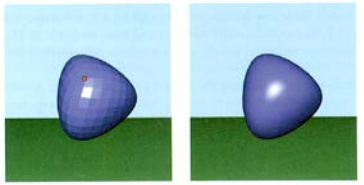


One level subdivision

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.

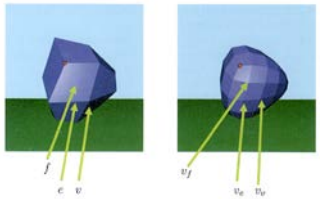
Two level subdivision



Same mesh with smooth shaded

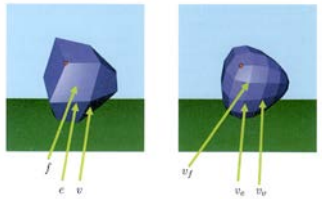
Catmull-Clark

- Start with watertight 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.



Catmull-Clark

  - For each face f in M0, we associate a new "face-vertex" v_f in M^1.
  - Connected up as shown.
- In M^1, all faces are quads.
- In M^1, we call any vertex of valence four "ordinary" and any vertex of valence different from four "extraordinary".

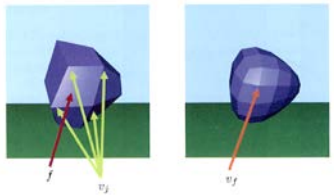


Geometry rules

- 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 $(m_f = 4)$

$$v_f = \frac{1}{m_f}\sum_j v_j$$

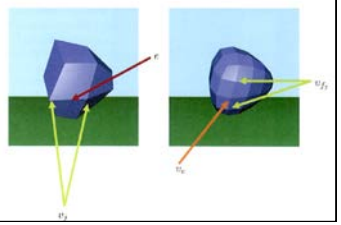
- The centroid of the vertices in M^i defining that face.



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:*

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



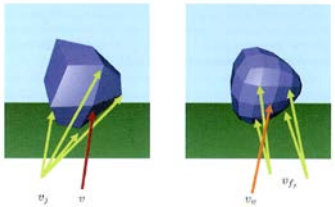
Vertex rules

- Finally let v be a vertex in M^i connected to the n_v vertcies 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, with valence $n_v=4$, 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.
- Verify that 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.
  - This is nice for the theory

Vertex rules

- Theory
  - It has been proven that this converges
  - It converges to a surface with continuous first derivative

Software tool for mesh

- Meshlab (http://meshlab.sourceforge.net)