Simulating the Everyday World

Three broad areas:
- Modeling (Geometric) = Shape
- Animation = Motion/Behavior
- Rendering = Appearance
Geometric Modeling

How to represent 3d shapes

- Polygonal meshes

Stanford Bunny
69451 triangles

David, Digital Michelangelo Project
28,184,526 vertices, 56,230,343 triangles

Geometric Modeling

How to represent 3d shapes

- Smooth surfaces
  - Bicubic spline surfaces
  - Subdivision surfaces

Caltech Head
Utah Teapot
Geometric Modeling

How to represent 3D shapes
How to create 3D shapes
  ■ CAD tools
  ■ Scanners
  ■ Procedurally
How to manipulate 3D shapes
  ■ Deform/skin/morph/animate
  ■ Smooth/compress
  ■ Set operations: union, difference

OpenGL Primitives: Coordinates

glBegin(GL_POLYGON);
  glVertex3f(-1.0,-1.0,0.0);
  glVertex3f(1.0,-1.0,0.0);
  glVertex3f(1.0,1.0,0.0);
  glVertex3f(-1.0,1.0,0.0);
glEnd();
Coordinates Stored in Arrays

```c
float v1[3] = {-1.0,-1.0,0.0};
float v2[3] = { 1.0,-1.0,0.0};
float v3[3] = { 1.0, 1.0,0.0};
float v4[3] = {-1.0, 1.0,0.0};

glBegin(GL_POLYGON);
    glVertex3fv(v1);
    glVertex3fv(v2);
    glVertex3fv(v3);
    glVertex3fv(v4);
glEnd();
```

Points/Polygons

```c
typedef float Point[3];

Point verts[8] = {
    {-1.,-1.,-1.},
    { 1.,-1.,-1.},
    { 1., 1.,-1.},
    {-1., 1.,-1.},
    {-1.,-1., 1.},
    { 1.,-1., 1.},
    { 1., 1., 1.},
    {-1., 1., 1.},
};

face(int a, int b, int c, int d) {
    glBegin(GL_POLYGON);
        glVertex3fv(verts[a]);
        glVertex3fv(verts[b]);
        glVertex3fv(verts[c]);
        glVertex3fv(verts[d]);
    glEnd();
}
// Note consistent ccw orientation!
cube() {
    face(0,3,2,1);
    face(2,3,7,6);
    face(0,4,7,3);
    face(1,2,6,5);
    face(4,5,6,7);
    face(0,1,5,4);
}
Points/Polygons

typedef float Point[3];
Point verts[8] = {
  {-1.,-1.,-1.},
  { 1.,-1.,-1.},
  { 1., 1.,-1.},
  {-1., 1.,-1.},
  {-1.,-1., 1.},
  { 1.,-1., 1.},
  { 1., 1., 1.},
  {-1., 1., 1.}};
int polys[6][4] = {
  {0,3,2,1},
  {2,3,7,6},
  {0,4,7,3},
  {1,2,6,5},
  {4,5,6,7},
  {0,1,5,4}};

face(int poly[4]) {
  glBegin(GL_POLYGON);
  glVertex3fv(poly[0]);
  glVertex3fv(poly[1]);
  glVertex3fv(poly[2]);
  glVertex3fv(poly[3]);
  glEnd();
  }

cube() {
  for( int i = 0; i < n; i++ )
    face(polys[i]);
}

Comparison

Polygons
+ Simple
- Redundant information

Points/Polygons
+ Sharing vertices reduces memory usage
+ Ensure integrity of the mesh (moving a vertex causes that vertex in all the polygons to be moved)
Calculating Normals at Vertices

```python
for f in mesh.faces():
    N = 0
    for v1, v2, v3 in f.consecutivevertices():
        N += cross(v2-v1,v3-v1)
    f.N = normalize(N)

for v in mesh.verts():
    N = 0
    for f in v.faces():
        N += f.N
    v.N = normalize(N)
```

Additional Topological Information

Applications:
- Constant time access to neighbors
e.g. surface normal calculation, subdivision
- Editing the geometry
e.g. adding new vertices, new faces, etc.
- Maintain topological consistency

Answer: Topological data structures
Topological Properties of “Manifolds”

Def. A 2D manifold is a surface that when cut with a small sphere, always yields a disk.

Mesh manifolds have the following properties:
- An edge connects exactly two faces
- An edge connects exactly two vertices
- A face consists of a ring of edges and vertices
- A vertex consists of a ring of edges and faces
- Euler’s formula \( #f - #e + #v = 2 \) (for a surface topologically equivalent to a sphere)
  (Check for a cube: \( 6 - 12 + 8 = 2 \))
Triangle

struct Vert {
    Point pt;
    Face *f;
}

struct Face {
    Vert *v[3];
    Face *f[3];
}

Finding Next Face

Find the next face clockwise around a vertex \( v \) from a face \( f \)

Face *fcwvf(Vert *v, Face *f)
{
    if ( v == f->v[0] )
        return f[1];
    if ( v == f->v[1] )
        return f[2];
    if ( v == f->v[2] )
        return f[0];
}

Pat Hanrahan, Fall 2011
Winged-Edge Representation

http://www.baumgart.org/winged-edge/winged-edge.html

CS148 Lecture 18
Pat Hanrahan, Fall 2011

Winged-Edge Representation

```c
struct Edge {
    Face *pf, *nf;
    Vert *pv, *nv;
    Edge *pccw, *pcw;
    Edge *nccw, *ncw;
}

struct Vert {
    Point pt;
    Edge *e;
}

struct Face {
    Plane pl;
    Edge *e;
}
```
Finding Next Edge

Find the next edge clockwise around a face f

Edge *ecwef(Edge *e, Face *f)
{
    if( f == e->pf )
        return e->pcw;
    if( f == e->nf )
        return e->ncw;
}

Finding Next Face

Find the next face clockwise from an edge e about a vertex v

Edge *fcwev(Edge *e, Vert *v)
{
    if( v == e->pv )
        return e->nf;
    if( v == e->nv )
        return e->pf;
}
Subdivision Surfaces

Recall: Subvision Algorithm

\[ P_0 \to P_1 \to P_2 \to P_3 \]
Compute Midpoints

\[ P_1^1 = \frac{1}{2}(P_1 + P_2) \]

\[ P_0^1 = \frac{1}{2}(P_0 + P_1) \]

\[ P_2^1 = \frac{1}{2}(P_2 + P_3) \]

Again

\[ P_0^2 = \frac{1}{2}(P_0^1 + P_1^1) \]

\[ P_1^2 = \frac{1}{2}(P_1^1 + P_2^1) \]
And Again

\[ P_0^3 = \frac{1}{2}(P_0^2 + P_1^2) \]

Split the Curve at the Midpoint

\[ P_0^3 = P\left(\frac{1}{2}\right) \]

\[ P_0 = P(0) \]

\[ P_3 = P(1) \]
Each Half is a Bezier Curve

Left Bezier Curve

Right Bezier Curve

Can We Generalize Subdivision to Surfaces?
Loop Subdivision

Loop Subdivision Surface

Refine a triangular mesh
Triangle Mesh

Subdivide Each Tri. into 4 Triangles
Subdivide Each Tri. into 4 Triangles

Nomenclature: Two Types of Vertices

Old – Vertices that existed in previous mesh
New – Vertices created in new mesh
Loop Algorithm – New Vertices

Weights of the surrounding vertices

N. B. weights sum to 1

Loop Algorithm – Old Vertices

For degree 6 vertices
Semi-Regular Meshes

Most of the mesh has vertices with degree 6
If the mesh is topologically equivalent to a sphere,
then not all the vertices can have degree 6
Must have a few extraordinary points
(degree $\neq 6$)

Extraordinary point

---

Proof: Always an Extraordinary Vert

Mesh has $V$ vertices, $E$ edges, and $T$ triangles

$2E = 3T$

- There are 3 edges per triangle
- Each edge is part of 2 triangles

$T = 2V-4$

- $T - E + V = 2 \Rightarrow V = \frac{3}{2}T - T + 2 = T/2 + 2$

If all vertices had 6 triangles, $6V = 3T \Rightarrow T = 2V$

- There are 3 edges per triangle
- There are 6 edges per vertex

$T$ cannot equal $2V-4$ and $2V$, a contradiction

Therefore, a (finite) surface cannot always have 6 triangles per vertex
Weights at an Extraordinary Point

Challenge: find weights that generate a smooth surface (tangent plane continuous)

Want the surface normal to be continuous

This is a hard math problem!

Warren weights

\[
\beta = \begin{cases} 
3 & n > 3 \\
\frac{3}{8} n & n = 3 \\
\frac{3}{16} & n = 3 
\end{cases}
\]

Loop Subdivision Surfaces
Catmull-Clark Subdivision

Quad Face Subdivision – Insert Vertex

\[
\begin{array}{c}
\frac{1}{4} \\
\frac{1}{4} \\
\frac{1}{4} \\
\frac{1}{4}
\end{array}
\]

---

CS148 Lecture 18
Pat Hanrahan, Fall 2011
Edge Subdivision – Insert Vertex

Original vertex

New face vertex

Update Vertex using Computed Vertices

New edge vertex

\[ \beta = \frac{2}{k} \]

New face vertex

\[ \gamma = \frac{1}{k} \]

Old vert vertex

\[ 1 - \beta - \gamma \]
Can Generalize to Any Mesh

1. Each new face vertex is the average of the vertices of the face

2. Each new edge vertex is the average of the two original vertices and the two new face vertices (same as quadrilateral meshes)

3. Update the old vertex using the formula on previous slide

Things to Remember

Dense polygon mesh data structures
- Polygons
- Points/Polygon

Subdivision surfaces
- Loop subdivision algorithm
- Catmull-Clark subdivision algorithm