Modeling

A Volkswagan Beetle becomes the subject of a 1970 simulation project. Ivan Sutherland (left) and assistants plot coordinates for digitizing the car.

Modeling the Everyday World

Three broad areas:

- **Modeling (Geometric)** = Shape
- **Animation** = Motion/Behavior
- **Rendering** = Appearance
Geometric Modeling

1. 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

1. How to represent 3d shapes
   - Smooth surfaces
     - Bicubic spline surfaces
     - Subdivision surfaces

Caltech Head

Utah Teapot
**Geometric Modeling**

1. How to represent 3D shapes
2. How to create 3D shapes
   1. CAD tools
   2. Scanners
   3. Procedurally
3. How to manipulate 3D shapes
   1. Deform/skin/morph/animate
   2. Smooth/compress
   3. Set operations, ...

**OpenGL Primitives**
**Primitive API**

```cpp
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();
```

**Polygons**

```cpp
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();
```
Topology

#f - #e + #v = 2

Points/Polygons

typedef float Point[3];

Point verts[6] = {
  (-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[6] = {
(-1.,-1.,-1.),
( 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 a, int b, int c, int d) {
    glBegin(GL_POLYGON);
    glVertex3fv(verts[a]);
    glVertex3fv(verts[b]);
    glVertex3fv(verts[c]);
    glVertex3fv(verts[d]);
    glEnd();
}

cube() {
    for( int i = 0; i < n; i++ )
        face(polys[i][0], polys[i][1], polys[i][2], polys[i][3]);
}

Representations

Polygons
+ Simple
- Redundant information

Points/Polygons
+ Share vertices (compress/consistency)

Additional topological information
+ Constant time access to neighbors
    More advanced algorithms such as
    surface normal calculation, subdivision ...
- Additional storage for topology
- More complicated data structures
Triangle Adjacency

struct Vert {
    Point pt;
    Face *f;
}

struct Face {
    Vert *v[3];
    Face *f[3];
}
Chaiken’s Algorithm

\[ P^1_0 = (1-t)P_0 + tP_1 \]
Chaiken’s Algorithm

\[ P_0^1 = (1-t)P_0 + tP_1 \]
\[ P_1^1 = (1-t)P_1 + tP_2 \]

\[ P_0^2 = (1-t)P_0^1 + tP_1^1 \]
Chaiken’s Algorithm

\[ P_0^1 = (1-t)P_0 + tP_1 \]
\[ P_1^1 = (1-t)P_1 + tP_2 \]
\[ P_0^2 = (1-t)P_0^1 + tP_1^1 \]
\[ P(t) = P_0^2 \]

Beziers Curves

Generalize last algorithm to 4 points
**Bezier Curve = Bernstein Polynomials**

\[
P(t) = \sum_{i=0}^{3} P_i B^3_i(t)
\]

\[
B^3_i(t) = \binom{3}{i} t^i (1-t)^{3-i}
\]

---

**Bezier Curves – Midpoint Subdivision**

Recursively divide into two curves

Left side

\[Q_0 = P_0\]
\[Q_1 = P_0^1\]
\[Q_2 = P_0^2\]
\[Q_3 = P_0^3\]
Beziers Curves – Midpoint Subdivision

Recursively divide into two curves

Left side

\[ Q_0 = P_0 \]
\[ Q_1 = P_0^1 = \frac{1}{2} P_0 + \frac{1}{2} P_1 \]
\[ Q_2 = P_0^2 = \frac{1}{4} P_0 + \frac{1}{2} P_1 + \frac{1}{4} P_2 \]
\[ Q_3 = P_0^3 = \frac{1}{8} P_0 + \frac{3}{8} P_1 + \frac{3}{8} P_2 + \frac{1}{8} P_3 \]

Right side

\[ R_0 = P_0^3 \]
\[ R_1 = P_1^2 \]
\[ R_2 = P_2^1 \]
\[ R_3 = P_3 \]
Beziers Curves – Midpoint Subdivision

Recursively divide into two curves

\[ R_0 = P_0^3 = \frac{1}{8} P_0 + \frac{3}{8} P_1 + \frac{3}{8} P_2 + \frac{1}{8} P_3 \]

\[ R_1 = P_1^2 = \frac{1}{4} P_1 + \frac{1}{2} P_2 + \frac{1}{4} P_3 \]

\[ R_2 = P_2^1 = \frac{1}{2} P_2 + \frac{1}{2} P_3 \]

\[ R_3 = P_3 \]

Subdivision Surfaces
Triangle Mesh

Triangle Mesh – Topological Subdivide
Loop Algorithm - Edge

Loop Algorithm - Vert
Semi-Regular Meshes

Extraordinary Points

Loop Subdivision – Extraordinary Vertex

\[ \beta = \frac{1}{k} \left[ \frac{5}{8} - \left( \frac{3}{8} + \cos \left( \frac{2\pi}{k} \right) \right)^2 \right] \]

\[ k \text{ neighbors} \ldots \]
Fractal Subdivision

\[ \Delta x \]
Fractal Subdivision

\[ \Delta y = \text{random()} \cdot \Delta x \]

Fractal Subdivision: Height Field
Summary

Digital geometry processing ala signal processing

Three common representations
- Dense polygon meshes
- Bicubic surfaces
- Subdivision surfaces

Common operations
- Instancing
- Transformation: linear and non-linear (bend)
- Compressing and simplifying
- ...

CS148 Lecture 18  Pat Hanrahan, Winter 2007