CSCI 480 Computer Graphics
Lecture 7

Polygon Meshes
and Implicit Surfaces

Polygon Meshes
Implicit Surfaces
Constructive Solid Geometry
[Angel Ch. 12.1-12.3]

February 6, 2013
Jernej Barbic
University of Southern California

http://www-bcf.usc.edu/~jbarbic/cs480-s13/

1

Modeling Complex Shapes

Source: Wikipedia

• An equation for a sphere is possible, but how about an equation for a telephone, or a face?

• Complexity is achieved using simple pieces
  – polygons, parametric surfaces, or implicit surfaces

• Goals
  – Model *anything* with arbitrary precision (in principle)
  – Easy to build and modify
  – Efficient computations (for rendering, collisions, etc.)
  – Easy to implement (a minor consideration...)

2

What do we need from shapes in Computer Graphics?

• Local control of shape for modeling
• Ability to model what we need
• Smoothness and continuity
• Ability to evaluate derivatives
• Ability to do collision detection
• Ease of rendering

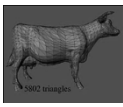No single technique solves all problems!
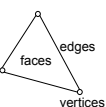
3

Shape Representations

Polygon Meshes
Parametric Surfaces
Implicit Surfaces

4

Polygon Meshes

5802 triangles

• Any shape can be modeled out of polygons
  – if you use enough of them…
• Polygons with how many sides?
  – Can use triangles, quadrilaterals, pentagons, … n-gons
  – Triangles are most common.
  – When > 3 sides are used, ambiguity about what to do when polygon nonplanar, or concave, or self-intersecting.
• Polygon meshes are built out of
  – *vertices* (points)
  – *edges* (line segments between vertices)
  – *faces* (polygons bounded by edges)

edges
faces
vertices

5

Polygon Models in OpenGL

• for faceted shading
```
glNormal3fv(n);
glBegin(GL_POLYGONS);
glVertex3fv(vert1);
glVertex3fv(vert2);
glVertex3fv(vert3);
glEnd();
```

• for smooth shading
```
glBegin(GL_POLYGONS);
glNormal3fv(normal1);
glVertex3fv(vert1);
glNormal3fv(normal2);
glVertex3fv(vert2);
glNormal3fv(normal3);
glVertex3fv(vert3);
glEnd();
```

6

Normals

Triangle defines unique plane
· can easily compute normal

$$\mathbf{n} = \frac{\mathbf{a} \times \mathbf{b}}{|\mathbf{a} \times \mathbf{b}|}$$

$$\mathbf{a} = \mathbf{v}_2 - \mathbf{v}_1$$
$$\mathbf{b} = \mathbf{v}_3 - \mathbf{v}_1$$

· depends on vertex orientation!
· clockwise order gives

$$\mathbf{n}' = -\mathbf{n}$$

Vertex normals less well defined
· can average face normals
· works for smooth surfaces
· but not at sharp corners
  – think of a cube

7

Where Meshes Come From

• Specify manually
  – Write out all polygons
  – Write some code to generate them
  – Interactive editing: move vertices in space

• Acquisition from real objects
  – Laser scanners, vision systems
  – Generate set of points on the surface
  – Need to convert to polygons

8

Data Structures for Polygon Meshes

• Simplest (but dumb)
  – float triangle[n][3][3]; (each triangle stores 3 (x,y,z) points)
  – redundant: each vertex stored multiple times

• Vertex List, Face List
  – List of vertices, each vertex consists of (x,y,z) geometric (shape) info only
  – List of triangles, each a triple of vertex id's (or pointers) topological (connectivity, adjacency) info only

  Fine for many purposes, but finding the faces adjacent to a vertex takes O(F) time for a model with F faces. Such queries are important for topological editing.

• Fancier schemes:
  Store more topological info so adjacency queries can be answered in O(1) time.

  Winged-edge data structure – edge structures contain all topological info (pointers to adjacent vertices, edges, and faces).

9

A File Format for Polygon Models: OBJ

```
# OBJ file for a 2x2x2 cube
v  -1.0 1.0 1.0        - vertex 1
v  -1.0 -1.0 1.0       - vertex 2
v  1.0 -1.0 1.0        - vertex 3
v  1.0 1.0 1.0         -…
v  -1.0 1.0 -1.0
v  -1.0 -1.0 -1.0
v  1.0 -1.0 -1.0
v  1.0 1.0 -1.0
f  1 2 3 4
f  8 7 6 5
f  4 3 7 8
f  5 1 4 8
f  5 6 2 1
f  2 6 7 3
```


Syntax:

v x y z - a vertex at (x,y,z)

f v_1 v_2 … v_n - a face with vertices v_1, v_2, … v_n

*anything*   - comment

10

How Many Polygons to Use?

5802 triangles   800 triangles

300 triangles   100 triangles

11

Why Level of Detail?

• Different models for near and far objects
• Different models for rendering and collision detection
• Compression of data recorded from the real world

We need automatic algorithms for reducing the polygon count without
  • losing key features
  • getting artifacts in the silhouette
  • popping

12

Problems with Triangular Meshes?

- Need a lot of polygons to represent smooth shapes
- Need a lot of polygons to represent detailed shapes

- Hard to edit
- Need to move individual vertices
- Intersection test? Inside/outside test?
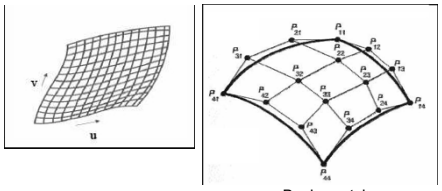
13

Shape Representations

Polygon Meshes
Parametric Surfaces
Implicit Surfaces

14

Parametric Surfaces

p(u,v) = [x(u,v), y(u,v), z(u,v)]
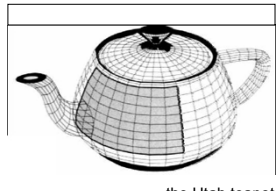
– e.g. plane, cylinder, bicubic surface, swept surface

Bezier patch

15

Parametric Surfaces

p(u,v) = [x(u,v), y(u,v), z(u,v)]

– e.g. plane, cylinder, bicubic surface, swept surface

the Utah teapot

16

Parametric Surfaces

Why better than polygon meshes?

– Much more compact
– More convenient to control --- just edit control points
– Easy to construct from control points

What are the problems?

– Work well for smooth surfaces
– Must still split surfaces into discrete number of patches
– Rendering times are higher than for polygons
– Intersection test? Inside/outside test?

17
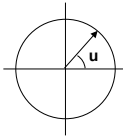
Shape Representations

Polygon Meshes
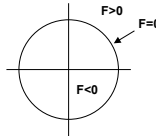**Parametric Surfaces**
Implicit Surfaces

18

Two Ways to Define a Circle

Parametric

Implicit

F>0

F=0

F<0

u

$x = f(u) = r \cos (u)$
$y = g(u) = r \sin (u)$

$F(x,y) = x^2 + y^2 - r^2$

19

Implicit Surfaces

outside:
$F>0$

inside:
$F<0$
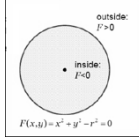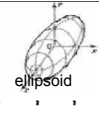
$F(x,y) = x^2 + y^2 - r^2 = 0$

• well defined inside/outside
• polygons and parametric surfaces
  do not have this information

• Computing is hard:
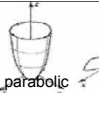   implicit functions for a cube?
      telephone?

• Implicit surface: F(x,y,z) = 0
  – e.g. plane, sphere, cylinder, quadric, torus, blobby models
    sphere with radius r : $F(x,y,z) = x^2+y^2+z^2-r^2 = 0$

  – terrible for iterating over the surface
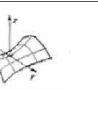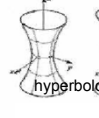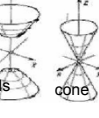 – great for intersections, inside/outside test

20

Quadric Surfaces

$F(x,y,z) = ax^2+by^2+cz^2+2fyz+2gzx+2hxy+2px+2qy+2rz+d = 0$

ellipsoid   parabolic
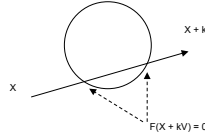
hyperboloids   cone   cylinder

21

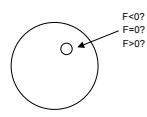## What Implicit Functions are Good For

X + kV

X

F(X + kV) = 0

F<0?
F=0?
F>0?

Ray - Surface Intersection Test   Inside/Outside Test

22

Surfaces from Implicit Functions

• Constant Value Surfaces are called
  (depending on whom you ask):
  – constant value surfaces
  – level sets
  – isosurfaces

• Nice Feature: you can add them! (and other tricks)
  – this merges the shapes
  – When you use this with spherical exponential potentials, it's
    called *Blobs*, *Metaballs,* or *Soft Objects*. Great for modeling animals.

23

Blobby Models

by Brian Wyvill, http://www.cpsc.ucalgary.ca/~blob/

24

How to draw implicit surfaces?

- It's easy to ray trace implicit surfaces
  - because of that easy intersection test
- Volume Rendering can display them
- Convert to polygons: the Marching Cubes algorithm
  - Divide space into cubes
  - Evaluate implicit function at each cube vertex
  - Do root finding or linear interpolation along each edge
  - Polygonize on a cube-by-cube basis

25

Constructive Solid Geometry (CSG)

- Generate complex shapes with basic building blocks
- Machine an object - saw parts off, drill holes, glue pieces together
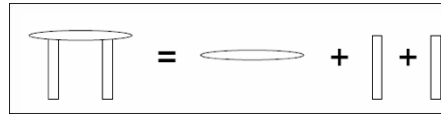
26

## Constructive Solid Geometry (CSG)

union   difference   intersection

Source: Wikipedia

the merger of two objects into one   the subtraction of one object from another   the portion common to both objects
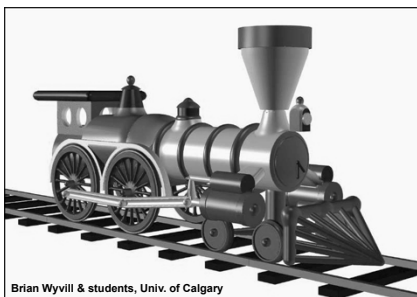
27

Constructive Solid Geometry (CSG)

- Generate complex shapes with basic building blocks
- Machine an object - saw parts off, drill holes, glue pieces together

- This is sensible for objects that are actually made that way (human-made, particularly machined objects)
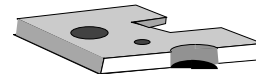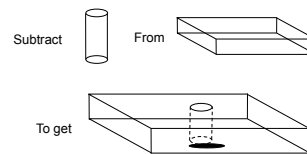

28

A CSG Train

Brian Wyvill & students, Univ. of Calgary

29

Negative Objects

Use point-by-point boolean functions
  - remove a volume by using a negative object
  - e.g. drill a hole by subtracting a cylinder

Subtract   From

To get

Inside(BLOCK-CYL) = Inside(BLOCK) And Not(Inside(CYL))

30

Set Operations

- UNION:　　　　　Inside(A) || Inside(B)
  - ➢ Join A and B

- INTERSECTION:　　Inside(A) && Inside(B)
  - ➢ Chop off any part of A that sticks out of B

- SUBTRACTION:　•　Inside(A) && (! Inside(B))
  - ➢ Use B to Cut A

Examples:
  - Use cylinders to drill holes
  - Use rectangular blocks to cut slots
  - Use half-spaces to cut planar faces
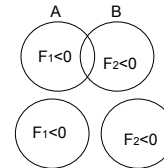 - Use surfaces swept from curves as jigsaws, etc.

31

Implicit Functions for Booleans

- Recall the implicit function for a solid: $F(x,y,z)<0$
- Boolean operations are replaced by arithmetic:
  - MAX　　　　replaces AND (intersection)
  - MIN　　　　replaces OR (union)
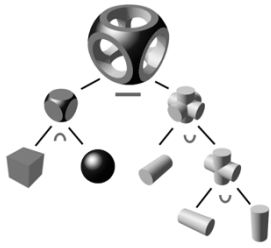 - MINUS　　　replaces NOT(unary subtraction)
- Thus
  - $F(Intersect(A,B)) = MAX(F(A),F(B))$
  - $F(Union(A,B))　= MIN(F(A),F(B))$
  - $F(Subtract(A,B)) = MAX(F(A), -F(B))$

A　　B

$F_1<0$　$F_2<0$

$F_1<0$　$F_2<0$

32

CSG Trees

- Set operations yield tree-based representation

33

Implicit Surfaces

  - Good for smoothly blending multiple components
  - Clearly defined solid along with its boundary
  - Intersection test and Inside/outside test are easy

  - Need to polygonize to render --- expensive
  - Interactive control is not easy
  - Fitting to real world data is not easy
  - Always smooth

34

Summary

  - Polygonal Meshes

  - Parametric Surfaces

  - Implicit Surfaces

  - Constructive Solid Geometry

35