

## Polygon Meshes and Implicit Surfaces

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

February 2, 2011  
Jernej Barbic

University of Southern California

<http://www-bcf.usc.edu/~jbarbic/cs480-s11/>

1

## Modeling Complex Shapes



Source: Wikipedia

- An equation for a sphere is possible, but how about an equation for a telephone, or a face?
  - polygons, parametric surfaces, or implicit surfaces
- 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

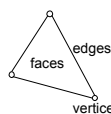
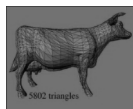
## Curve Representations

### Polygon Meshes Parametric Surfaces Implicit Surfaces

4

## Polygon Meshes

- 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)



5

## Polygon Models in OpenGL

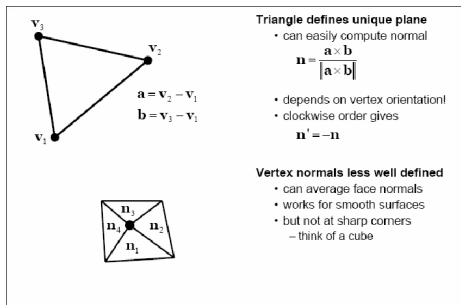
- for faceted shading

```
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



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

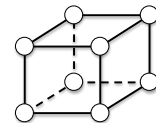
- Simplest (but dumb)
  - float triangle[n][3][3]; (each triangle stores 3 (x,y,z) points)
  - redundant: each vertex stored multiple times
- Vertex List, Face List
  - List of vertices, each vertex consists of (x,y,z) geometric (shape) info only
  - List of triangles, each a triple of vertex id's (or 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:

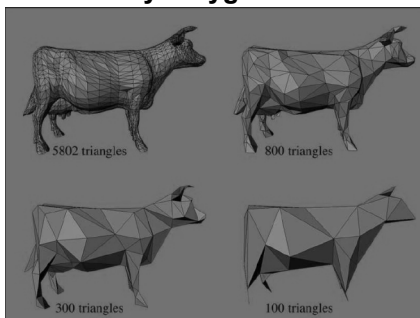
**v** x y z - a vertex at (x,y,z)

**f**  $V_1 V_2 \dots V_n$  - a face with vertices  $V_1, V_2, \dots V_n$

**#** anything - comment

10

## How Many Polygons to Use?



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

## Curve Representations

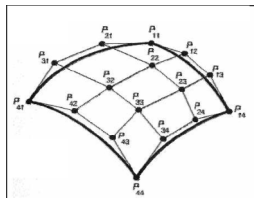
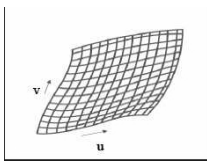
**Polygon Meshes**  
**Parametric Surfaces**  
**Implicit Surfaces**

14

## Parametric Surfaces

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

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



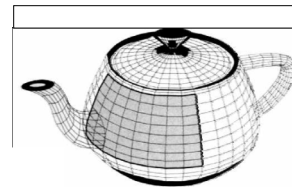
bezier patch

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

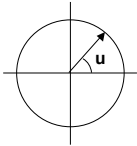
## Curve Representations

**Polygon Meshes**  
**Parametric Surfaces**  
**Implicit Surfaces**

18

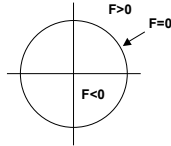
## Two Ways to Define a Circle

### Parametric



$$\begin{aligned} x &= f(u) = r \cos(u) \\ y &= g(u) = r \sin(u) \end{aligned}$$

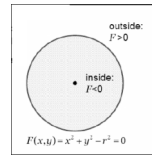
### Implicit



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

19

## Implicit Surfaces



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

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

### • Implicit surface: $F(x,y,z) = 0$

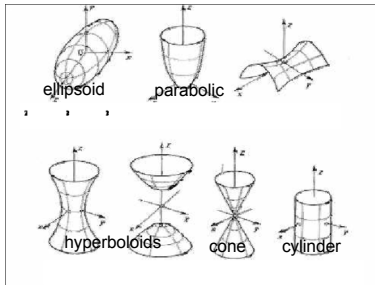
- e.g. plane, sphere, cylinder, quadric, torus, blobby models
- sphere with radius  $r$ :  $F(x,y,z) = x^2 + y^2 + z^2 - r^2 = 0$

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

20

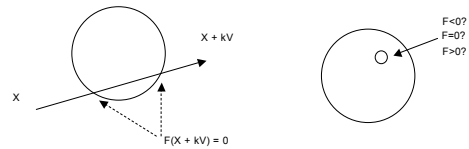
## Quadric Classes

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



21

## What Implicit Functions are Good For



Ray - Surface Intersection Test

Inside/Outside Test

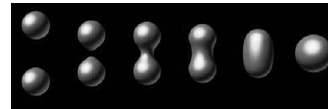
22

## 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



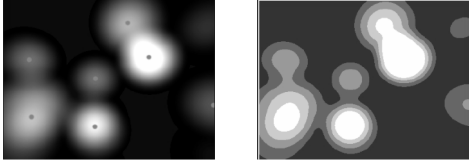
- Implicit function is the sum of Gaussians centered at several points in space, minus a threshold
- varying the standard deviations of the Gaussians makes each blob bigger
- varying the threshold makes blobs merge or separate

24

## Bloppy Models

$$f(x,y,z) = 1.0 / (x^2 + y^2 + z^2)$$

form blobs if close

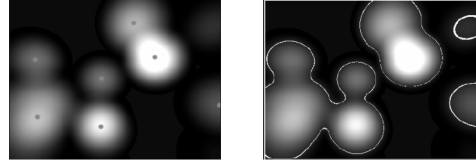


25

## Bloppy Models

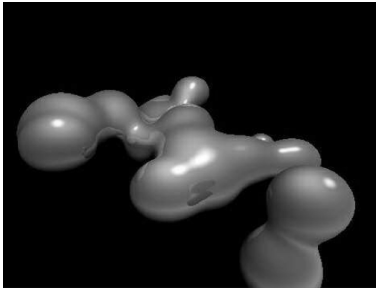
$$f(x,y,z) = 1.0 / (x^2 + y^2 + z^2)$$

form blobs if close



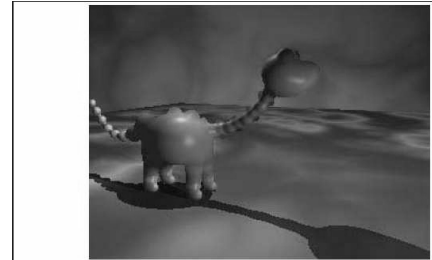
26

## Bloppy Models



27

## Bloppy Models



by Brian Wywill, <http://www.cpsc.ucalgary.ca/~bloby/>

28

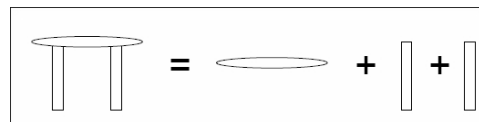
## How to draw implicit surfaces?

- 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

29

## Constructive Solid Geometry (CSG)

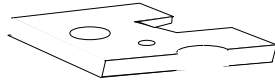
- Generate complex shapes with basic building blocks
- Machine an object - saw parts off, drill holes, glue pieces together



30

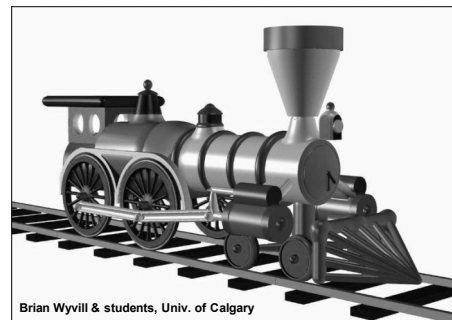
## Constructive Solid Geometry (CSG)

- 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)



31

## A CSG Train



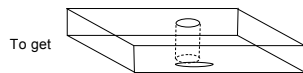
Brian Wyvill & students, Univ. of Calgary

32

## 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



$$\text{Inside}(\text{BLOCK}-\text{CYL}) = \text{Inside}(\text{BLOCK}) \text{ And Not}(\text{Inside}(\text{CYL}))$$

33

## Set Operations

- UNION:  $\text{Inside}(A) \parallel \text{Inside}(B)$   
  - Join A and B
- INTERSECTION:  $\text{Inside}(A) \ \&\& \ \text{Inside}(B)$   
  - Chop off any part of A that sticks out of B
- SUBTRACTION:  $\text{Inside}(A) \ \&\& \ (\text{! Inside}(B))$   
  - 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.

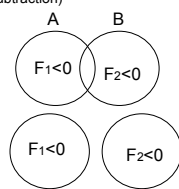
34

## 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

- $F(\text{Intersect}(A,B)) = \text{MAX}(F(A), F(B))$
- $F(\text{Union}(A,B)) = \text{MIN}(F(A), F(B))$
- $F(\text{Subtract}(A,B)) = \text{MAX}(F(A), -F(B))$



35

## 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

36

## Summary

- Polygonal Meshes
- Parametric Surfaces
- Implicit Surfaces
- Constructive Solid Geometry