3D Object Representation

Michael Kazhdan
(601.457/657)
Suppose you would like to define a polynomial describing a circle with radius 2, centered about the point (1,2) in 2D:

\[ P(x, y) = (x - 1)^2 + (y - 2)^2 - 4 \]
\[ = x^2 + y^2 - 2x - 4y + 1 \]
Util::Polynomial< Dim, Degree >

\[ P(x, y) = x^2 + y^2 - 2x - 4y + 1 \]

Defining the polynomial:

```cpp
// Declare a polynomial in two variables of degree 2
Util::Polynomial2D< 2 > poly2D;

// Set the (non-zero) coefficients
poly2D.coefficient(2u,0u) = 1;  // the \( x^2y^0 \) term
poly2D.coefficient(0u,2u) = 1;  // the \( x^0y^2 \) term
poly2D.coefficient(1u,0u) = -2; // the \( x^1y^0 \) term
poly2D.coefficient(0u,1u) = -4; // the \( x^0y^1 \) term
poly2D.coefficient(0u,0u) = 1;  // the \( x^0y^0 \) term
```
Util::Polynomial< Dim , Degree >

\[ P(x, y) = x^2 + y^2 - 2x - 4y + 1 \]

Evaluating the polynomial:

// The x and y coordinates at which to evaluate P
double x , y;
...

// The point at which you want to evaluate P
Util::Point2D p(x,y);
...

// Evaluate P at the prescribed point
double value1 = poly2D( p );
double value2 = poly2D( x , y );
Util::Polynomial< Dim, Degree >

\[ P(x, y) = x^2 + y^2 - 2x - 4y + 1 \]

Restricting the polynomial:

// The ray to which you want to restrict P
Util::Ray2D ray;

... 

// The restriction of P to the ray is a polynomial
// of degree 2 in one variable
Util::Polynomial1D< 2 > poly1D = poly2D( ray );
\textbf{Util::Polynomial< Dim , Degree >}

\[ P(x, y) = x^2 + y^2 - 2x - 4y + 1 \]

Restricting the polynomial:

// The ray to which you want to restrict P
Util::Ray2D ray;

...

// The restriction of P to the ray is a polynomial
// of degree 2 in one variable
Util::Polynomial1D< 2 > poly1D = poly2D( ray );
$P(x, y) = x^2 + y^2 - 2x - 4y + 1$

Find the roots (of a 1D polynomial):

```cpp
Util::Polynomial1D< 2 > poly1D;
...
double roots[2];
unsigned int rootNum = p.roots( roots );
```

**Note:**
The number of roots may be less than the degree of the polynomial, so check the value of `rootNum`. 
Util::Polynomial< Dim, Degree >

\[ P(x, y) = x^2 + y^2 - 2x - 4y + 1 \]

Differentiating the polynomial:

// The partial derivative with respect to x
Util::Polynomial2D< 1 > dx = poly2D.d( 0 );

// The partial derivative with respect to y
Util::Polynomial2D< 1 > dy = poly2D.d( 1 );
3D Objects

How can this object be represented in a computer?
3D Objects

This one?

H&B Figure 10.46
3D Objects

This one?  

H&B Figure 9.9
3D Objects

This one?
3D Object Representations

• Raw data
  ◦ Point cloud
  ◦ Range image
  ◦ Polygon soup

• Surfaces
  ◦ Mesh
  ◦ Subdivision
  ◦ Parametric
  ◦ Sweep

• Solids
  ◦ Implicit
  ◦ Voxels
  ◦ BSP tree
  ◦ CSG

• High-level structures
  ◦ Scene graph
  ◦ Skeleton
  ◦ Application specific
Point Clouds

• Unstructured set of 3D point samples
  ◦ Acquired from random sampling, particle system implementations, etc.
Range Image

- An image storing depth instead of / as well as color
  - Acquired from 3D scanners

Range Image  |  Tessellation  |  Range Surface
Polygon Soups

• Unstructured set of polygons
  ○ Created with interactive modeling systems, combining range images, etc.
3D Object Representations

• Raw data
  ◦ Point cloud
  ◦ Range image
  ◦ Polygon soup

• Surfaces
  ◦ Mesh
  ◦ Subdivision
  ◦ Parametric
  ◦ Sweep

• Solids
  ◦ Implicit
  ◦ Voxels
  ◦ BSP tree
  ◦ CSG

• High-level structures
  ◦ Scene graph
  ◦ Skeleton
  ◦ Application specific
(Manifold) Meshes

• Connected set of polygons (usually triangles)
Subdivision Surfaces

• Coarse mesh & subdivision rule
  ○ Define a smooth surface as limit of a hierarchical sequence of refinements
Parametric Surfaces

• Tensor product spline patches
  ○ Careful use of constraints to maintain continuity

FvDFH Figure 11.44
Parametric Surfaces: Sweep Surfaces

- Surface swept by curve along trajectory

Stephen Chenney
U Wisconsin
3D Object Representations

• Raw data
  ○ Point cloud
  ○ Range image
  ○ Polygon soup

• Surfaces
  ○ Mesh
  ○ Subdivision
  ○ Parametric
  ○ Sweep

• Solids
  ○ Implicit
  ○ Voxels
  ○ BSP tree
  ○ CSG

• High-level structures
  ○ Scene graph
  ○ Skeleton
  ○ Application specific
Implicit Surfaces

- Points satisfying: $F(x, y, z) = 0$
Voxels

- Uniform grid of volumetric samples
  - Acquired from CT, MRI, etc.
BSP Trees

- Binary space partition with solid cells labeled
  - Constructed from polygonal representations
Constructive Solid Geometry (CSG)

- Hierarchy of boolean set operations (union, difference, intersect) applied to simple shapes

FvDFH Figure 12.27
H&B Figure 9.9
3D Object Representations

• Raw data
  ○ Point cloud
  ○ Range image
  ○ Polygon soup

• Surfaces
  ○ Mesh
  ○ Subdivision
  ○ Parametric
  ○ Sweep

• Solids
  ○ Implicit
  ○ Voxels
  ○ BSP tree
  ○ CSG

• High-level structures
  ○ Scene graph
  ○ Skeleton
  ○ Application specific
Scene Graphs

- Union of objects at leaf nodes
Skeletons

- Graph of curves with geometry associated to individual curve positions

Stanford Graphics Laboratory
Application Specific

Apo A-1
(Theoretical Biophysics Group, University of Illinois at Urbana-Champaign)

Architectural Floorplan
Surfaces

• What makes a good surface representation?
  ◦ Concise
  ◦ Local support
  ◦ Affine invariant
  ◦ Arbitrary topology
  ◦ Guaranteed smoothness
  ◦ Natural parameterization
  ◦ Efficient display
  ◦ Efficient intersections
Surfaces

• What makes a good surface representation?
  ○ Concise
  ○ Local support
  ○ Affine invariant
  ○ Arbitrary topology
  ○ Guaranteed smoothness
  ○ Natural parameterization
  ○ Efficient display
  ○ Efficient intersections

smooth ≠ complex

H&B Figure 10.46
Surfaces

• What makes a good surface representation?
  ○ Concise (smooth $\neq$ complex)
  ○ Local support
  ○ Affine invariant
  ○ Arbitrary topology
  ○ Guaranteed smoothness
  ○ Natural parameterization
  ○ Efficient display
  ○ Efficient intersections

Not Local Support

edits are localized
Surfaces

• What makes a good surface representation?
  ○ Concise
  ○ Local support
  ○ Affine invariant
  ○ Arbitrary topology
  ○ Guaranteed smoothness
  ○ Natural parameterization
  ○ Efficient display
  ○ Efficient intersections

applying an affine transformation to the surface does not fundamentally change its representation.
Surfaces

• What makes a good surface representation?
  ◦ Concise
  ◦ Local support
  ◦ Affine invariant
  ◦ Arbitrary topology
  ◦ Guaranteed smoothness
  ◦ Natural parameterization
  ◦ Efficient display
  ◦ Efficient intersections

can represent surfaces without constraints on topology

Topological Genus Equivalences
Surfaces

• What makes a good surface representation?
  ○ Concise
  ○ Local support
  ○ Affine invariant
  ○ Arbitrary topology
  ○ Guaranteed smoothness
  ○ Natural parameterization
  ○ Efficient display
  ○ Efficient intersections

positions/normal vary continuously/smoothly over the surface
Surfaces

• What makes a good surface representation?
  ◦ Concise
  ◦ Local support
  ◦ Affine invariant
  ◦ Arbitrary topology
  ◦ Guaranteed smoothness
  ◦ Natural parameterization
  ◦ Efficient display
  ◦ Efficient intersections

A Parameterization (not necessarily natural) supports texture mapping
Surfaces

• What makes a good surface representation?
  ○ Concise
  ○ Local support
  ○ Affine invariant
  ○ Arbitrary topology
  ○ Guaranteed smoothness
  ○ Natural parameterization
  ○ Efficient display
  ○ Efficient intersections

supports efficient ray-tracing / real-time rendering