Ray Tracing Implicit Surfaces

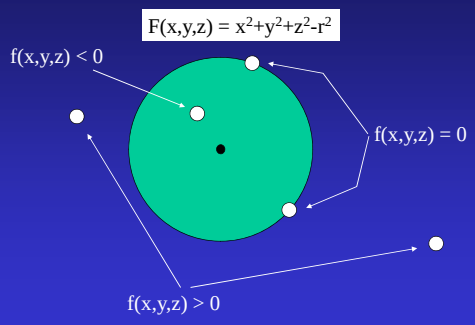
Overview

- Similar to **CSG**
  - Combine primitive objects to form complex object
- Primitives are "**density fields**"
- Combine by **summing** densities
- The surface is all points at which the density equals a user-defined **threshold**

Implicit Surface

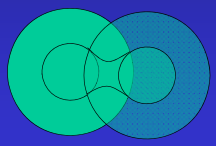
- A surface not explicitly represented
- The surface consists of all points which satisfy a function
  - $F(x,y,z) = 0$
- Usually, the implicit function is defined so that
  - $F(x,y,z) < 0$ => inside the object
  - $F(x,y,z) > 0$ => outside the surface
  - Sometimes $F(x,y,z)$ is based on a distance-to-a-central-element
- The surface points have to be searched for!

For example: single metaball



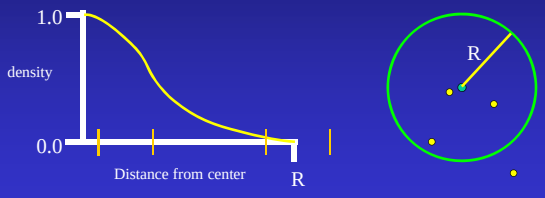
Multiple Implicits

- Define each primitive as positive density field
- Sum densities
- Surface is defined at threshold
- Usually have finite radius of influence



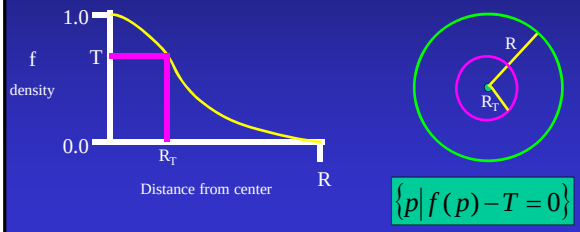
Organic shapes

Density Function



Threshold

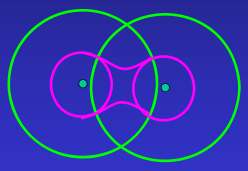
- Define threshold that defines density of surface
- R_T is the radius of the isosurface (blob) in isolation



$$\{p \mid f(p) - T = 0\}$$

Blended Blobs

- Define surface as sum of densities



$$\{p | \sum f_i(p) - T = 0\}$$

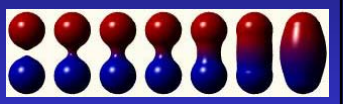
$$\{p | \sum w_i f_i(p) - T = 0\}$$

Weighted Density Functions

- Define surface as weighted sum of densities

$F(p) = \Sigma\ w_i\ f_i\ (p) - T = 0$

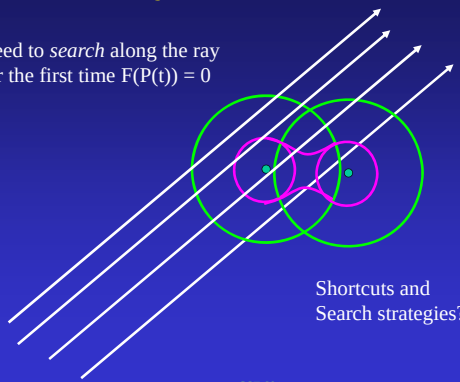
To keep the same radius, but increase blending, change weight, w_i, and the threshold, T, simultaneously.



Weights can be negative, too!

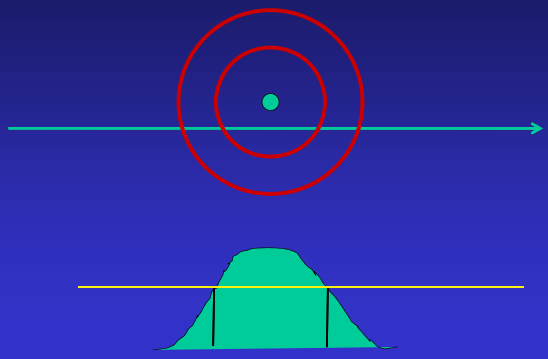
Ray Intersection

Need to *search* along the ray for the first time F(P(t)) = 0



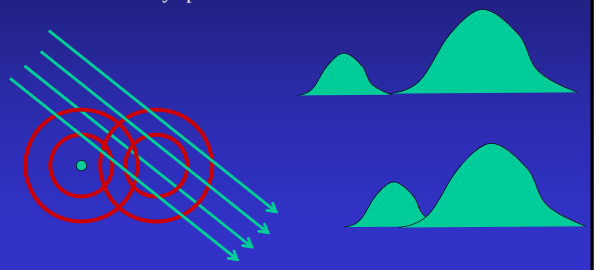
Shortcuts and Search strategies?

Search for Intersection



Search for Intersection

Identify spans of interest: bounds on intersection



Density Functions

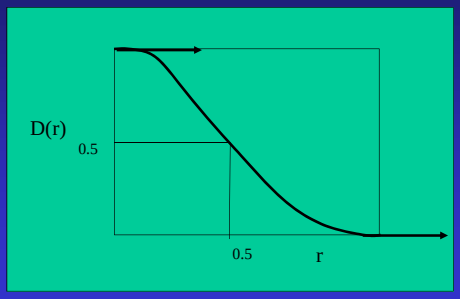
Define a density function that is:

- Easy to evaluate
- Blends smoothly
- Intuitive to use

Density functions proposed in the literature

- Exponential
- Piecewise cubic
- Cubic in distance squared

Density Functions



Distance-based Density Functions

$f_i(p) = D(|P-C_i|/R) = D(r)$

r is normalized distance

$D_1(r) = (1-r^2)^3$ $\quad 0 <= r < 1$

$D_2(r) = 1 - (4/9)r^6 + (17/9)r^4 - (22/9)r^2$ $\quad 0 <= r < 1$

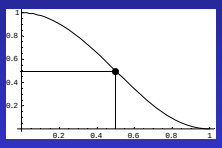
$D_3(r) = \exp(-ar^2)$

$D_4(r) = 1-3r^2$ $\quad 0 <= r < 1/3$

$(3/2)(1-r)^2$ $\quad 1/3 <= r < 1$

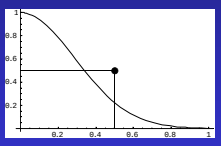
Distance-based Density Functions

$D_2(r) = 1 - (4/9)r^6 + (17/9)r^4 - (22/9)r^2 \qquad 0 <= r < 1$



Distance-based Density Functions

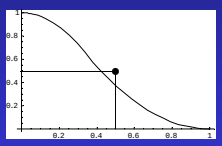
$D_3(r) = \exp(-ar^2)$



Distance-based Density Functions

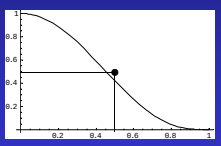
$D_4(r) = 1-3r^2 \qquad 0 <= r < 1/3$

$(3/2)(1-r)^2 \qquad 1/3 <= r < 1$



Distance-based Density Functions

$D_1(r) = (1-r^2)^3 \qquad 0 <= r < 1$



Distance-based Primitives

Point

Line

Polygon

Polygonal mesh

Polyline

Polyhedron

Anything you can efficiently compute the distance from

Distance-based Primitives

Point

Distance from point

Line

Distance from line or endpoints - partition by perpendiculars

Polyline

Distance from one of lines or points - partition by perpendiculars

Distance-based Primitives

Polygon

Partition space by planes perpendicular to plane through an edge

Polygonal mesh

Same, for each face - two planes per edge

Distance-based Primitives

Polyhedra

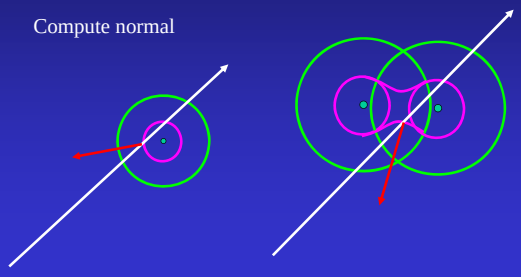
Convex?

Concave?

Display Considerations

Find point of intersection along ray: F(P(t)) = 0

Compute normal



Computing the Normal

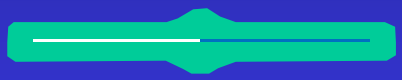
Form analytic expression of implicit function
And take partial derivatives
N = (δF/δx, δF/δy, δF/δz)

Take discrete approximation by sampling function
Compute gradient
N = (F(x+dx,y,z)-F(x,y,z), F(x,y +dy,z)-F(x,y,z), F(x,y,z +dz)-F(x,y,z))

Bulge problem

One long primitive

Two side-by-side primitives



CSG-approach to control blending

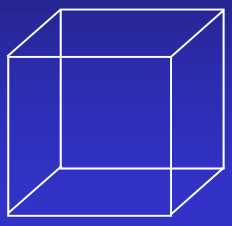
Use nodes to combine primitives by either summing or taking max of functions

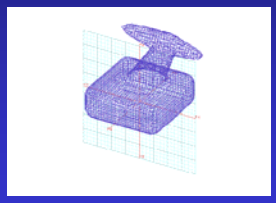
Complexity

- Bounding volumes
- Spatial subdivision - cellular bucket sort
- Hierarchical spatial subdivision – quadtree
- Binary spatial partitioning

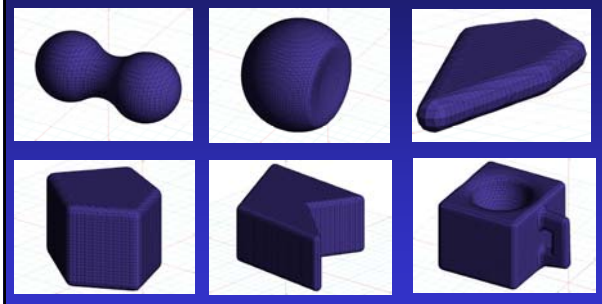
Display alternative

Marching cubes algorithm - construct surface fragments from isosurface intersections with grid cells

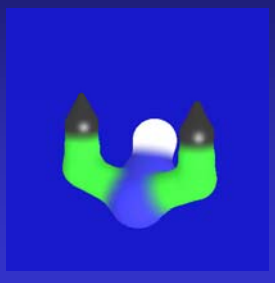




Distance-based Primitives



Examples



Examples


Examples