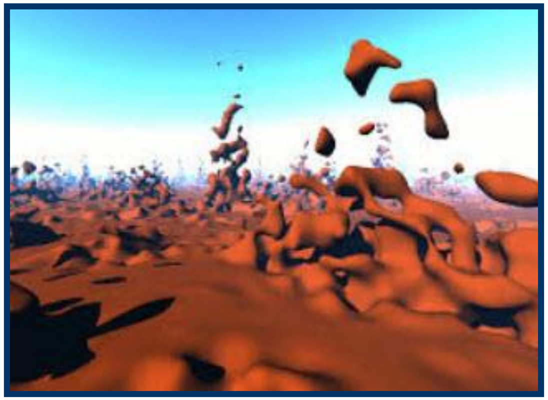
12. Special Models for Animation

Modeling techniques required in special effects in animation:

- implicit surfaces
- L-systems (for plants)
- subdivision surfaces

12.1 Implicit Surfaces

- animated implicit surfaces are useful for modeling liguids, clouds, and fanciful animal shapes



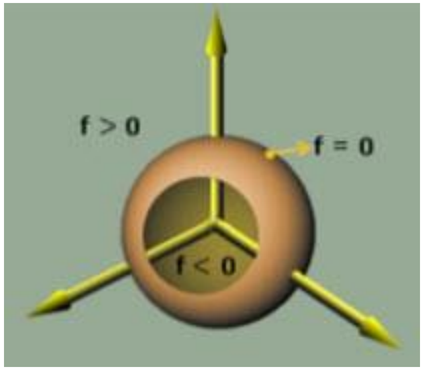
(J. Bloomenthal, ed., *Introduction to Implicit Surfaces, Morgan Kaufmann, San* Francisco)

Basic Implicit Surface Formulation

An implicit surface is defined by the set of points that satisfy an implicit function

$$f(P) = f(x, y, z) = 0$$

where $f(x, y, z)$ is a polynomial in the unknowns x, y, and z.



e.g.

$$x^2 + y^2 + z^2 - 4 = 0$$

$$y^2 - 2x = 0$$

Basic Implicit Surface Formulation

An implicit surface *f (x, y, z) = 0 is irreducible* if *f(x, y, z)* can not be factored as the product of two *nonconstant* polynomials

$$x^2 + y^2 + z^2 - 4 = 0$$ ← irreducible

$$x^2 - y^2 = 0$$ ← reducible

The *gradient* (or, *normal*) of the surface *f(x, y, z) = 0* is the vector (f_x , f_y, f_z) *where* f_x , f_y , *and* f_z are partial derivatives of *f* with respect to *x* , *y* , *and z, respectively.*

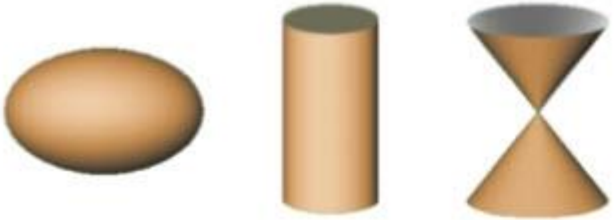
$$x^2 + y^2 + z^2 - 1 = 0 \quad at \quad (1, 0, 0)$$

Basic Implicit Surface Formulation

A point *(x_0, y_0, x_0)* on an irreducible surface is *regular* if the gradient at the point is not a zero vector. Otherwise, the point is *singular.*

$$y^2 - x^2 - x^3 = 0 \quad at \quad (0,0,0)$$

For every point of an implicit surface, there exists *tangent space* to the surface, consisting of all tangent lines to the surface at that point.



At a regular point, the tangent space is a plane, called the *tangent plane.* At a singular point, the *tangent space* is a cone.

Basic Implicit Surface Formulation

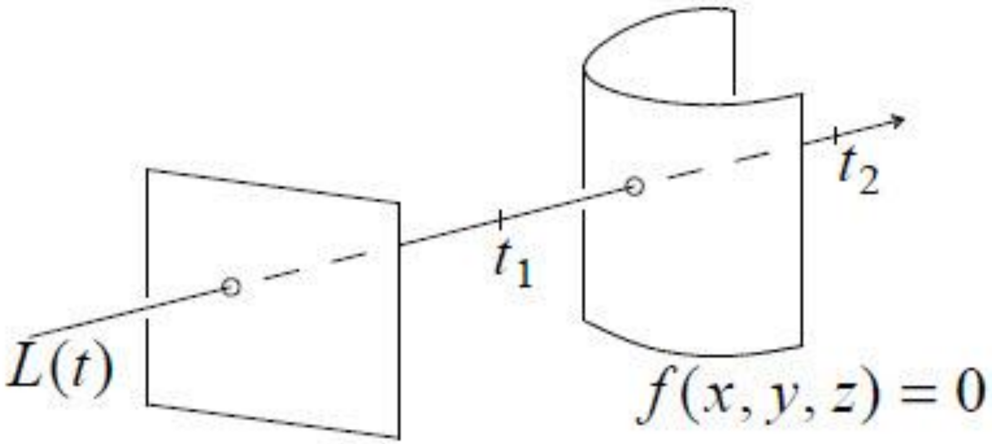
- If $f(x, y, z) = 0$ contains the origin, then at the origin, the equation of the tangent space is given by the terms of lowest degree in $f(x, y, z) = 0$.

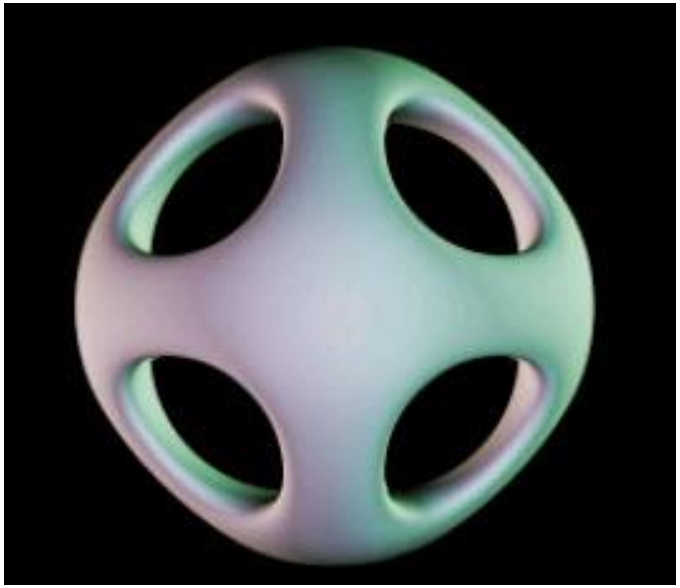
e.g.,

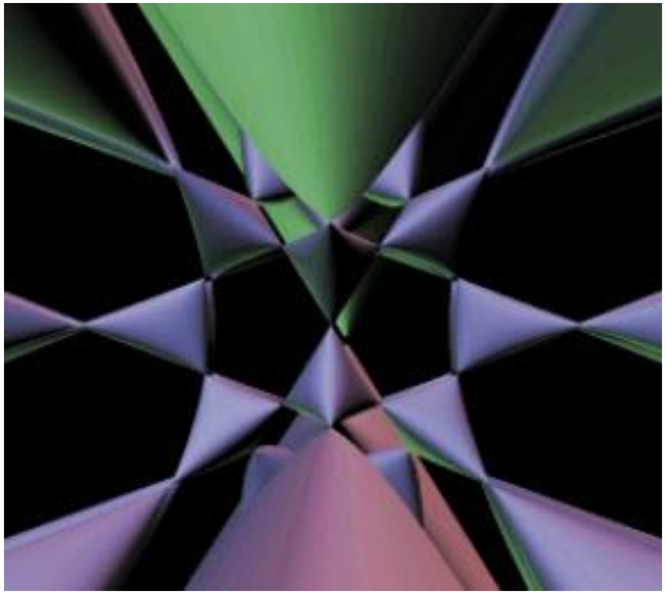
$$x^2 + y^2 + z^2 - 2x = 0$$

The terms of the lowest degree are called the *initial form* of f .

How to ray-trace an implicit surface?

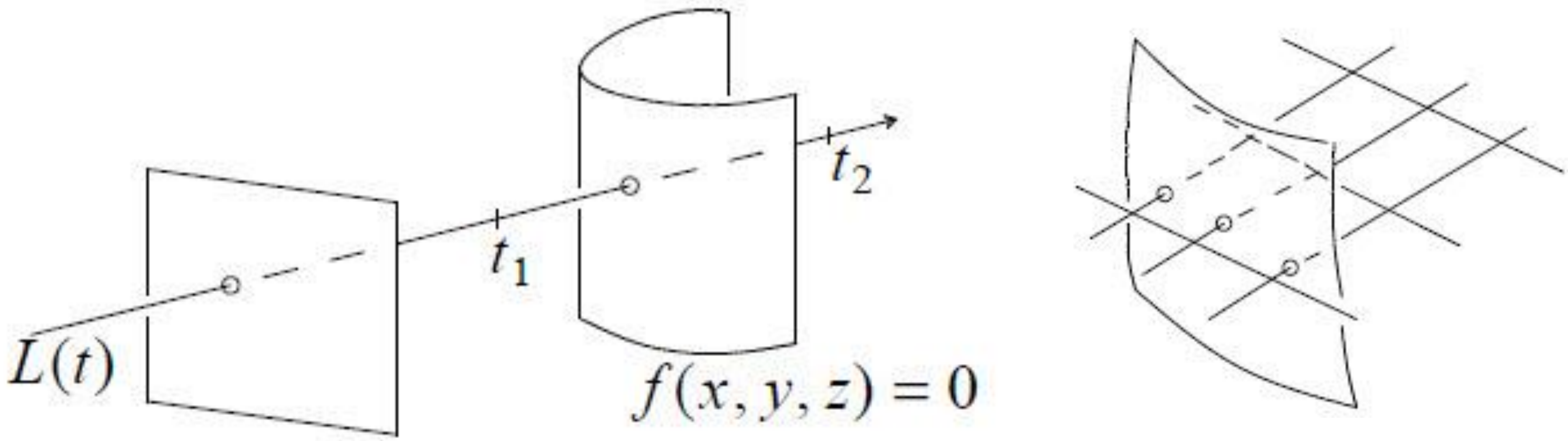






(Jorge Diaz 2008)

How to ray-trace an implicit surface?



How to construct an implicit polygonal representation of an implicit surface?

1. Sampling the implicit function at the vertices of a 3D mesh.
2. The implicit function values at mesh vertices are then interpolated along mesh edges to estimate the location of points that lie on the surface.

Polygonal Surface Simplification

- Usually we use a relatively small step size for the 3D mesh to ensure enough points are generated for regions of the implicit surface with large curvature.

- But this could generate too many points for regions already flat enough or relatively flat.

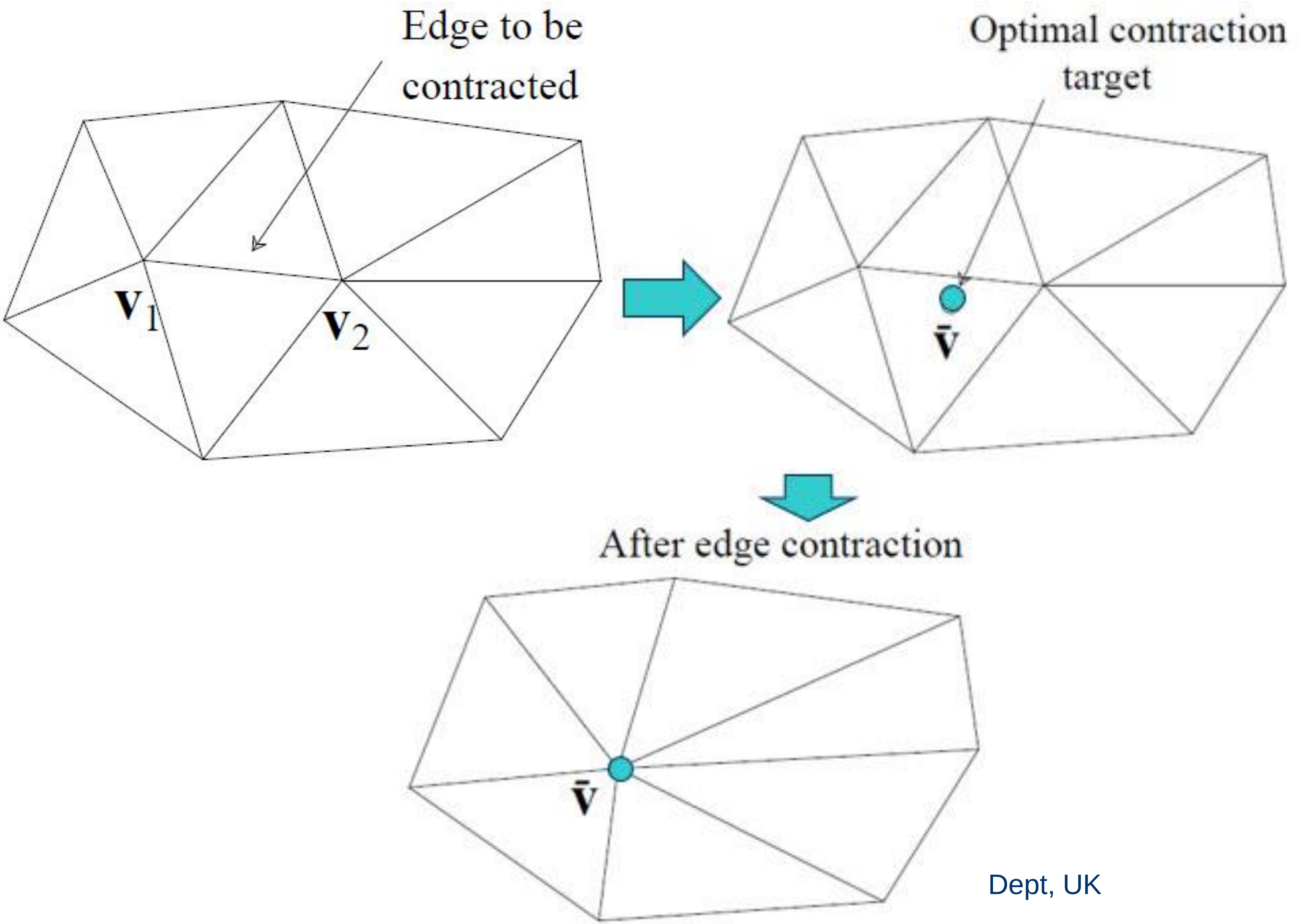
- Need to perform shape-preserving simplification

- We will use a technique proposed by M. Garland and P. Heckbert (Surface Simplification Using Quadric Error Metrics, SIGGRAPH 1997)

Polygonal Surface Simplification

- The input is a polygonal surface $\mathbf{M}_0$ with per-face normals (normalized)
- The idea is to iteratively contracting edges (or, vertex pairs) according to some cost function until a targeted vertex number is reached.

Basic idea of edge contraction:



How is $\bar{v}$ computed?

- $\bar{v}$ is the point with minimum distance (error) to all the adjacent triangles of **v_1** and **v_2**

- The **distance** of a point v = (*x, y, z)* to a plane is:

$$\left|pv^T\right|$$

where **p** = [a, b, c, d] represents the plane with $a^2 + b^2 + c^2 = 1$.

- The **squared distance** of a point v = (x, y, z) to a plane is:

$$\left(pv^T\right)^T\left(pv^T\right) = vp^T pv^T = vQ_p v^T$$

How is $\bar{v}$ computed? (conti)

- The distance (error) of a point v = (*x, y, z)* to the adjacent triangles of a vertex **v1** is:

$$\Delta(\mathbf{v}) = \sum_{\mathbf{p} \in planes(\mathbf{v}_1)} \mathbf{v}\ \mathbf{Q}_\mathbf{p} \mathbf{v}^T$$

$$= \mathbf{v} \left(\sum_{\mathbf{p} \in planes(\mathbf{v}_1)} \mathbf{Q}_\mathbf{p} \right) \mathbf{v}^T = \mathbf{v}\ \mathbf{Q}_1\ \mathbf{v}^T$$

- The **distance** (error) of a point v = (*x, y, z)* to the adjacent triangles of vertex **v1** and **v2** is:

$$\Delta(\mathbf{v}) = \mathbf{v}\ \mathbf{Q}_1\ \mathbf{v}^T + \mathbf{v}\ \mathbf{Q}_2\ \mathbf{v}^T$$

$$= \mathbf{v}\ (\mathbf{Q}_1 + \mathbf{Q}_2)\ \mathbf{v}^T$$

How is $\bar{\mathrm{v}}$ computed? (conti)

- How to minimize $\Delta(\mathrm{v})$?

$$\frac{\partial \Delta}{\partial x} = \frac{\partial \Delta}{\partial y} = \frac{\partial \Delta}{\partial z} = 0$$

- Note that

$$\frac{\partial \Delta}{\partial x} = [1, 0, 0, 0]\,(\,\mathbf{Q}_1 + \mathbf{Q}_2\,)\,\mathbf{v}^{\mathrm{T}} + \mathbf{v}\;\;(\,\mathbf{Q}_1 + \mathbf{Q}_2\,)\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

- equivalent to solving

$$\begin{bmatrix} q_{11} & q_{12} & q_{13} & q_{14} \\ q_{12} & q_{22} & q_{23} & q_{24} \\ q_{13} & q_{23} & q_{33} & q_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} \bar{\mathbf{v}} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

The Algorithm:

- Compute the **Q** matrices for all the initial vertices
- Compute the optimal contraction target $\bar{v}$ for each edge (**v**$_1$, **v**$_2$). The error $\bar{v}(Q_1 + Q_2)\bar{v}^T$ of this target vertex becomes the cost of contracting that edge
- Place all the edges in a heap keyed on cost with the minimum cost edge at the top
- Iteratively remove the edge (**v**$_1$, **v**$_2$) of least cost from the heap, contract this edge, and update the costs of all edges involving **v**$_1$ and **v**$_2$

How to use implicit surfaces to define objects useful for animation?:

- Construct the implicit function as a summation of implicitly defined primitive functions

(i) $F(p) = \sum w_i \, f_i(p) - T$

e.g.,

$$\begin{cases} f_1(p) = (x-1)^2 + (y-0)^2 + (z-0)^2 - 4 \\ f_2(p) = (x+1)^2 + (y-0)^2 + (z-0)^2 - 4 \end{cases}$$

How to use implicit surfaces to define objects useful for animation?:

$$\begin{cases} f_1(p) = (x-1)^2 + (y-0)^2 + (z-0)^2 - 4 \\ f_2(p) = (x+1)^2 + (y-0)^2 + (z-0)^2 - 4 \end{cases}$$

then

$$F(p) = f_1(p) + f_2(p) - 12$$

or

$$F(p) = f_1(p) - f_2(p) - 12$$

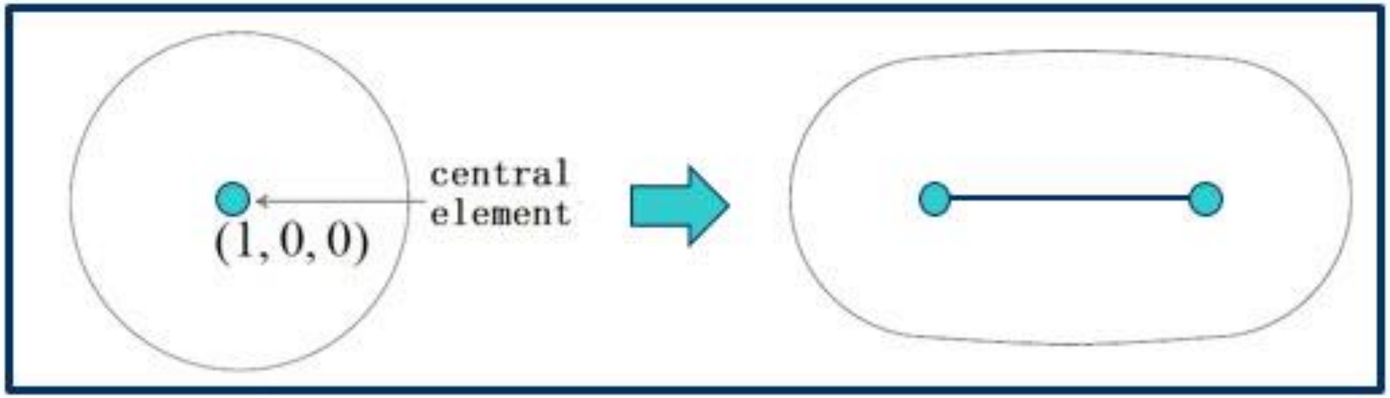
or

$$F(p) = f_1(p) - \frac{1}{2} f_2(p) - 12$$

How to use implicit surfaces to define objects useful for animation?:

(ii) Use wider range of central elements in the definition of a primitive

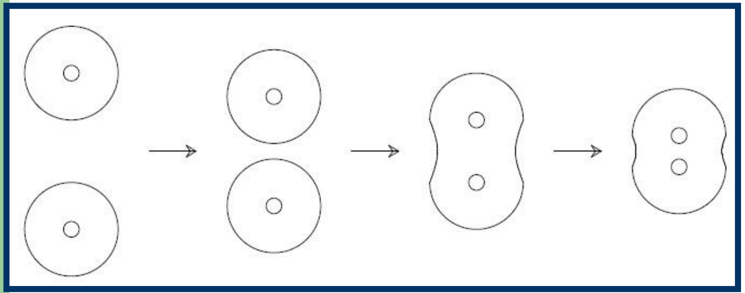
e.g., $f(p) = (x-1)^2 + (y-0)^2 + (z-0)^2 - 4 = 0$

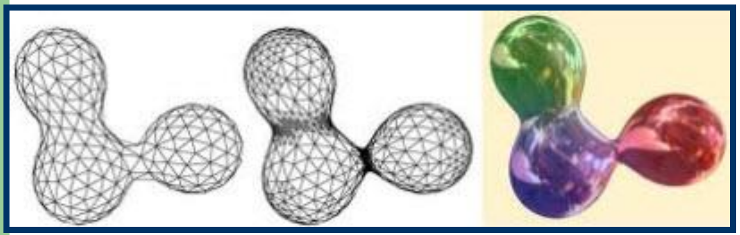


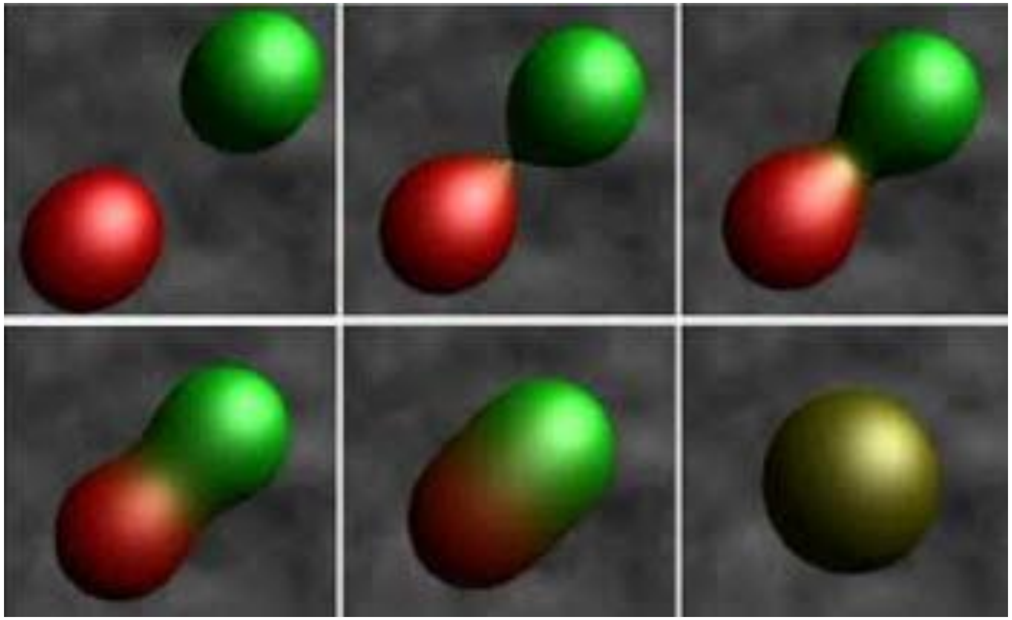
replace (1, 0, 0) with a line segment, or a triangle, a convex polyhedron, or even a concave polyhedron

Animation using implicitly defined surfaces

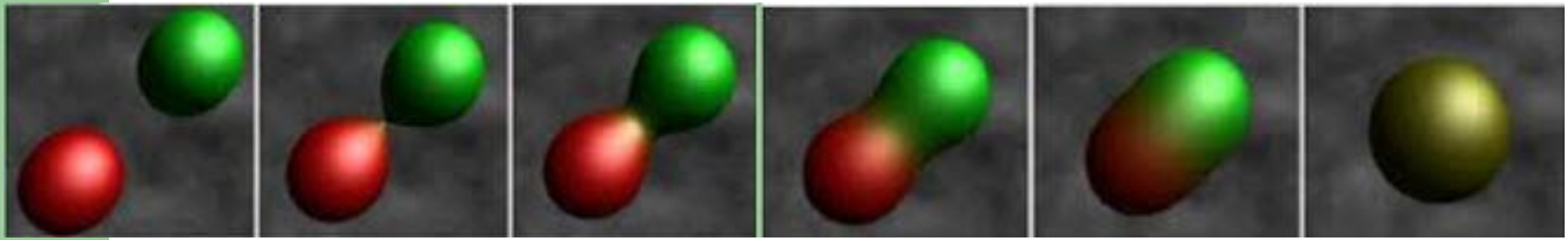
- modifying the shape of implicit surfaces by controlling the movement of the underlying control elements







Animation using implicitly defined surfaces



Define

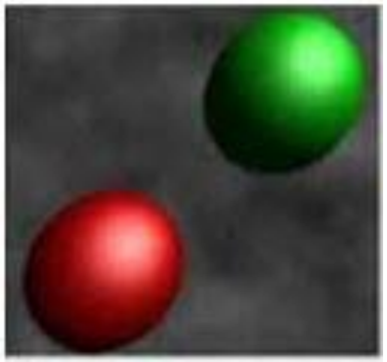
$$F(x,y,z) = \sum_1^2 \frac{1}{(x-x_i)^2+(y-y_i)^2+(z-z_i)^2} - 1$$

where $P_i = (x_i, y_i, z_i)$ is the center point of drop i.

Initially, set $P_1 = (0, 2, 0)$ and $P_2 = (0, -2, 0)$

Then move P_1 and P_2 toward each other.

Animation using implicitly defined surfaces



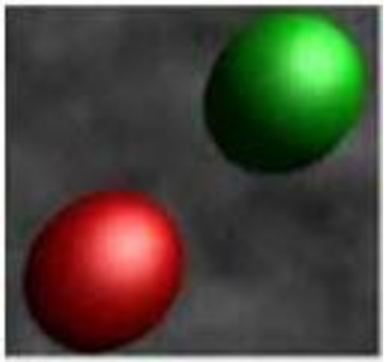
Why would we have two disjoint components?

$$F(x, y, z) = \sum_{1}^{2} \frac{1}{(x - x_i)^2 + (y - y_i)^2 + (z - z_i)^2} - 1$$

$$P_1 = (0, 2, 0)\ , \quad P_2 = (0, -2, 0)$$

Note that $F(0, y, 0) = \frac{1}{(y-2)^2} + \frac{1}{(y+2)^2} - 1$

Animation using implicitly defined surfaces



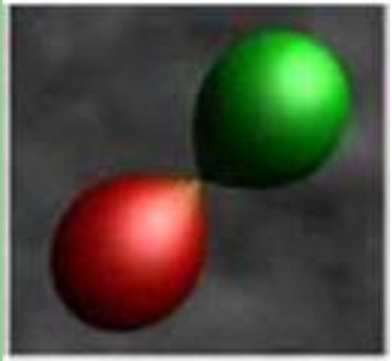
Why would we have two disjoint components?

Note that $F(0,y,0) = \frac{1}{(y-2)^2} + \frac{1}{(y+2)^2} - 1$

$\frac{1}{(y-2)^2} + \frac{1}{(y+2)^2} - 1 = 0$ has four solutions

$y = \pm\sqrt{5+\sqrt{17}}$ $\qquad$ $y = \pm\sqrt{5-\sqrt{17}}$

Animation using implicitly defined surfaces



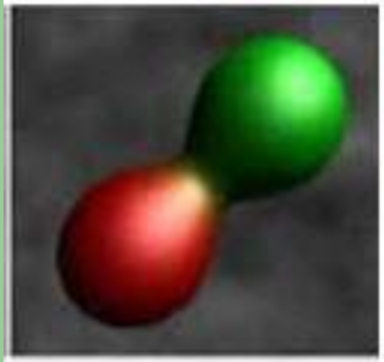
What happened?

When $P_1 = \left(0, \sqrt{2}, 0\right)$ and $P_2 = (0, -\sqrt{2}, 0)$

$F(0, y, 0) = 1$ has three solutions

$y = \pm\sqrt{6}$ $\quad$ $y = 0$

Animation using implicitly defined surfaces



What happened?

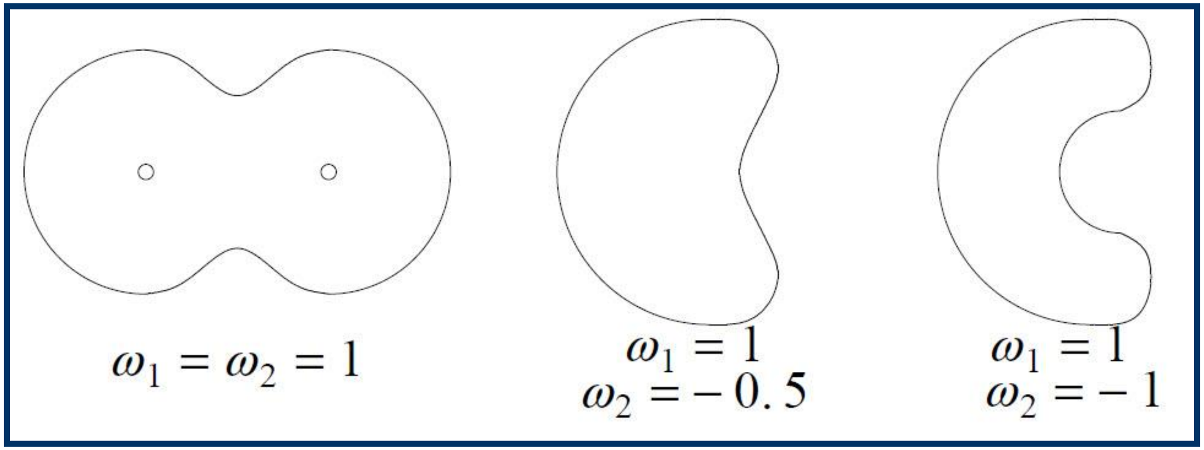
When $P_1 = (0, \alpha, 0)$ and $P_2 = (0, -\alpha, 0)$ with $0 \leq \alpha < \sqrt{2}$

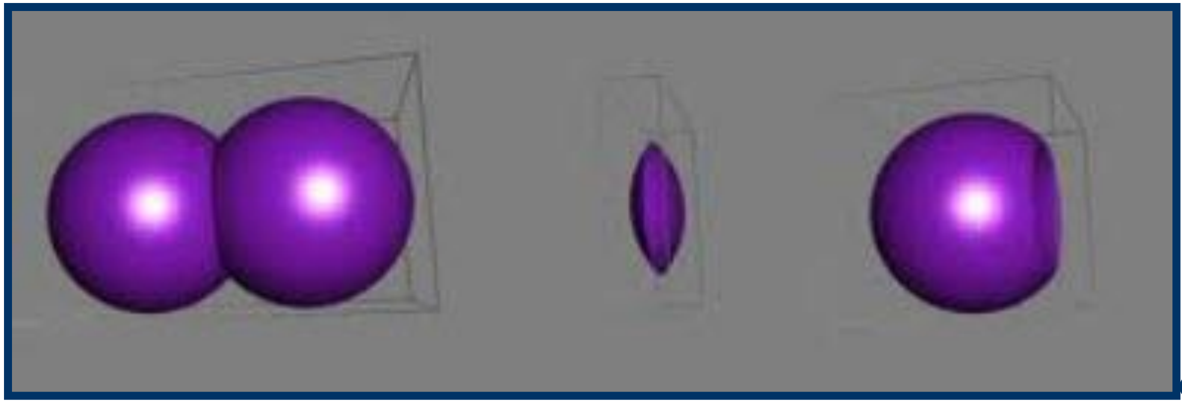
$F(0, y, 0) = 1$ has two solutions

$$y = \pm\sqrt{\alpha^2 + 1 + \sqrt{4\alpha^2 + 1}}$$

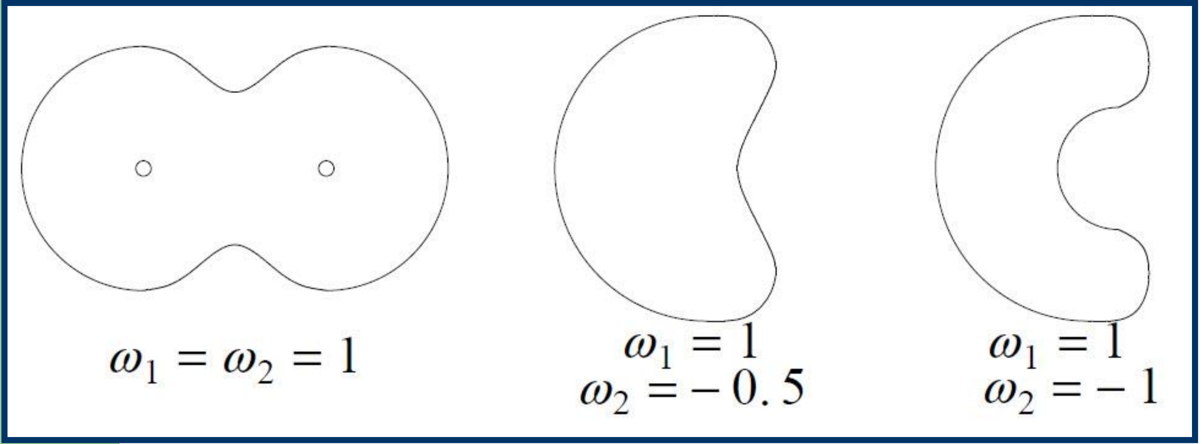
Animation using implicitly defined surfaces

- modifying the weights to effect bulging or the size of an implicit object





Animation using implicitly defined surfaces

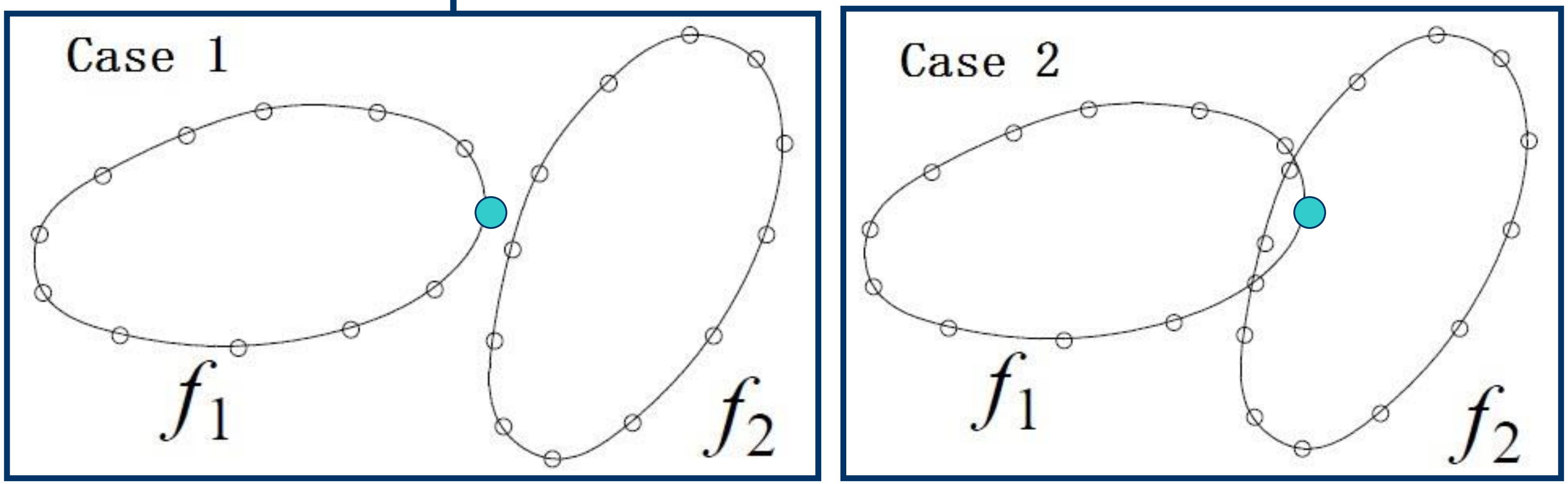


These three cases can be represented by

$$F(x,y) = \frac{\omega_1}{\sqrt{(x-1)^2 + y^2}} + \frac{\omega_2}{\sqrt{(x+1)^2 + y^2}} - 1.2$$

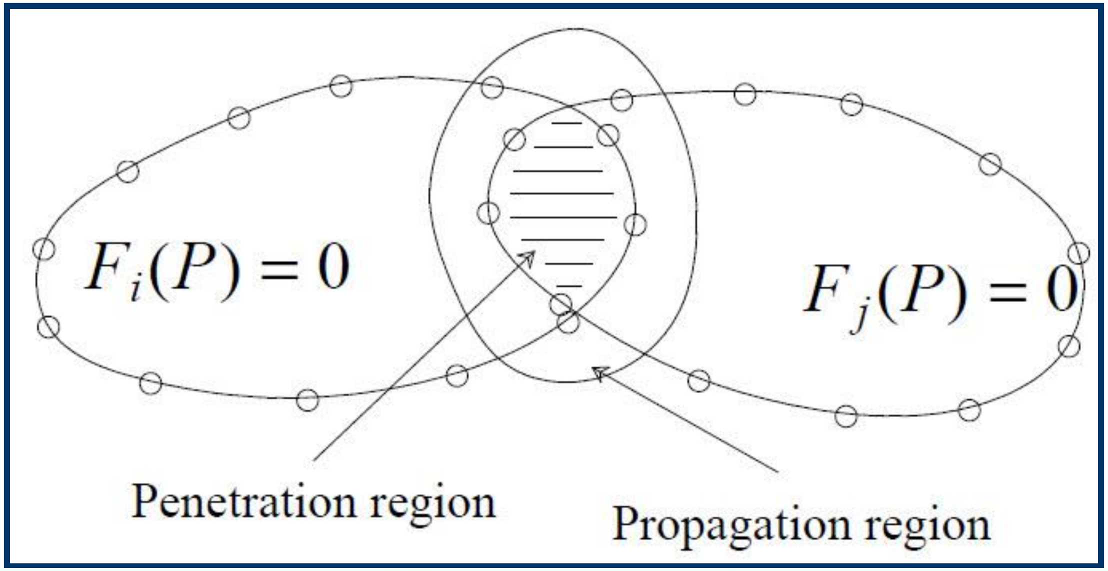
Collision Detection

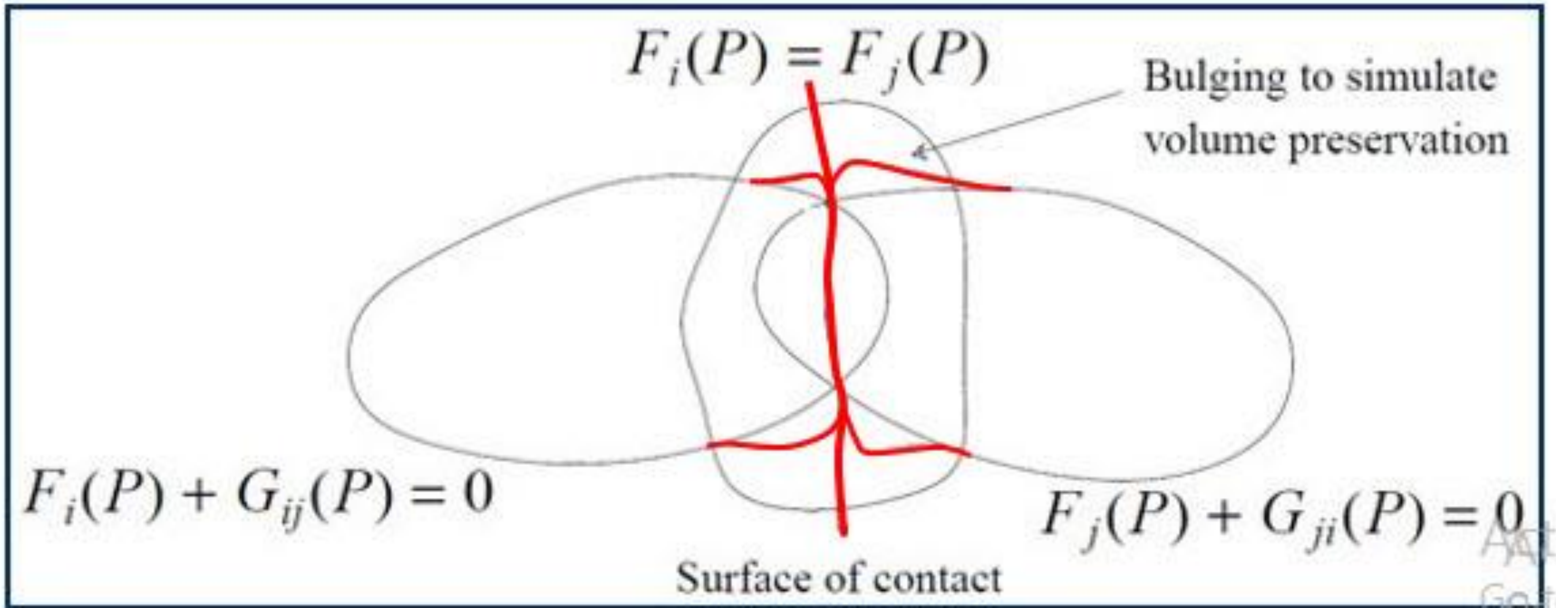
- easier for implicit surfaces



- evaluating the implicit function of one object as sample points of the other object and vice versa
- can also be used to test for collisions between polyhedral objects (if these objects can be approximated by some implicit surfaces)

Deforming the implicit surface as a result of collision





Deforming the implicit surface as a result of collision

- After the overlapping region (called *penetration region)* has been detected, objects i and j are deformed so that they coincide in the region of overlap
- Add a deformation term $G_{ij}(P)$ *to* $F_i(P)$

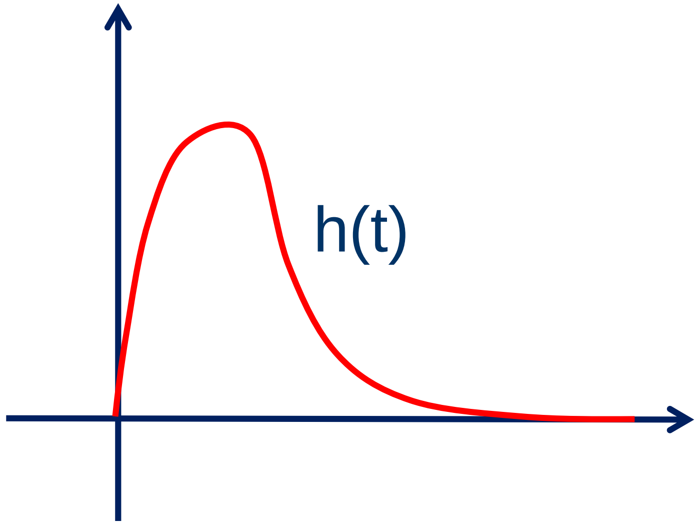
$$F_i(P) + G_{ij}(P) = 0$$

$$G_{ij}(P) = \begin{cases} -F_j(P), & \textit{overlapping region} \\ h(P), & \textit{propagation region} \\ 0, & \textit{outside propagation} \end{cases}$$

Deforming the implicit surface as a result of collision

h(P) is a function of the distance to the border of the penetration region

$h'(0)$ must be equal to the directional derivative of G_{ij} along the gradient at point *P,* to ensure *C1* continuity.



End of Special Models for Animation I