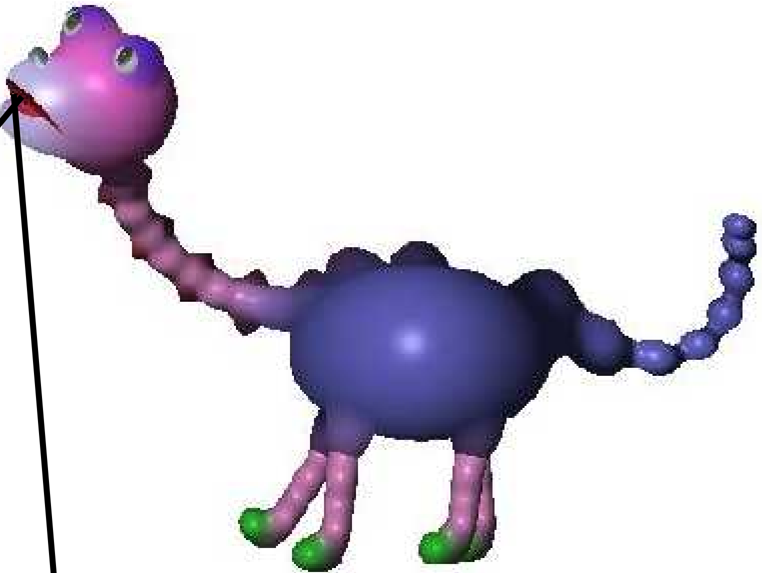
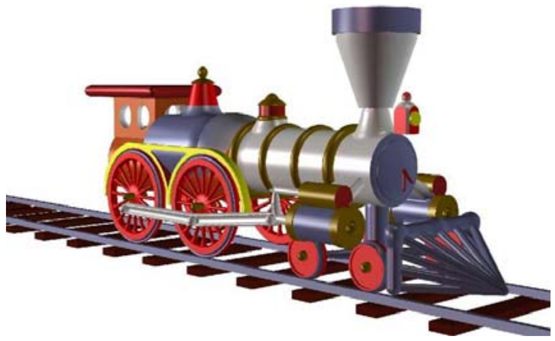
Come and hear

Introduction to Implicit Modelling

by Brian Wyvill



With a little help from his students



Overview

- Introduction to Implicit Surfaces
- Blending, Warping, CSG
- Some Problems
- The BlobTree
- Blending
- Texturing
- Animation
- Hierarchical Implicit Surfaces
- Building Models

Introduction to Implicit Surfaces

Implicit Definition

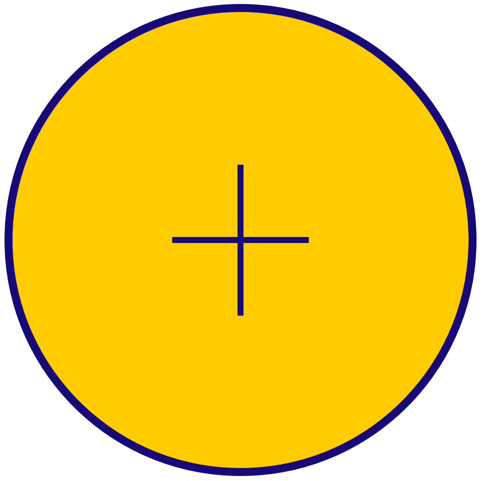
f(x,y) > x² , y² . r² > 1
e.g. r > !1
f)1-1*> !1 !, !1 !. 2 !⊨ !1 inside
f)1-1*> !2 !, !2 !. 2 !? !1 outside

implies search space to find
x,y to satisfy: f(x,y) > 1

iso-surface: f(x,y) - c > 1

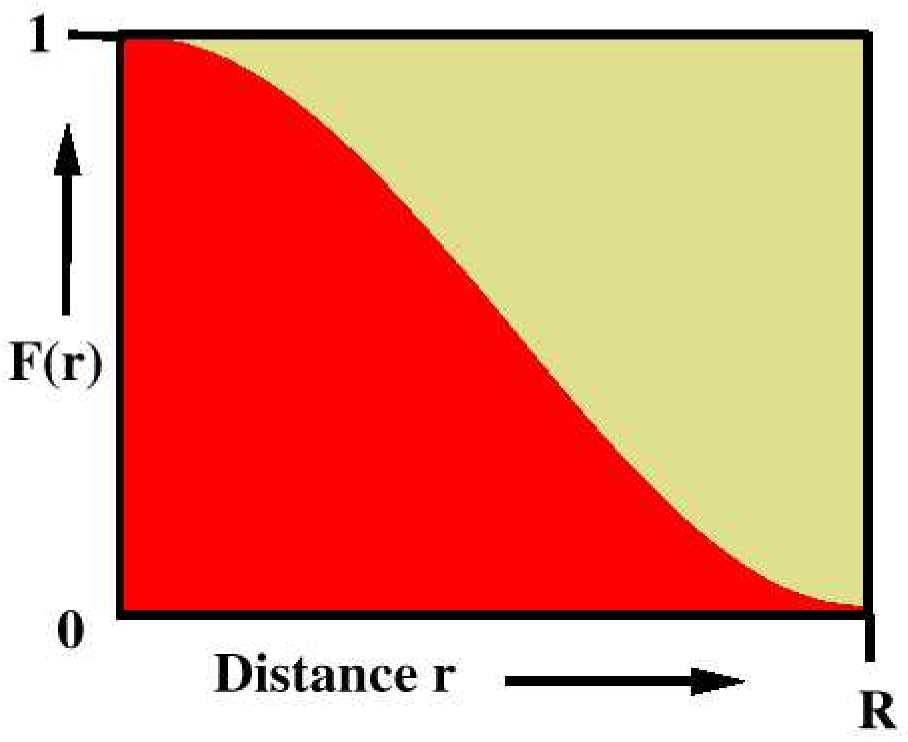
Parametric Definition

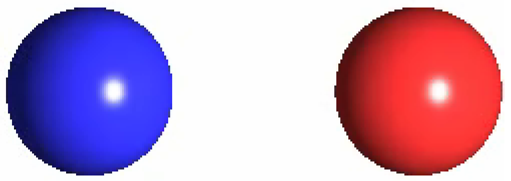
x > r sin(α)
y > r cos(α)
0 d α d !2π



The Geoff Function

Click Me

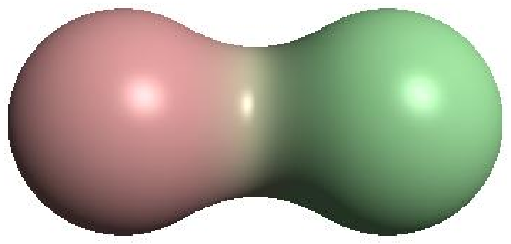




Proximity Blending:
Add contributions from generating skeletal elements in the neighbourhood

Field Function

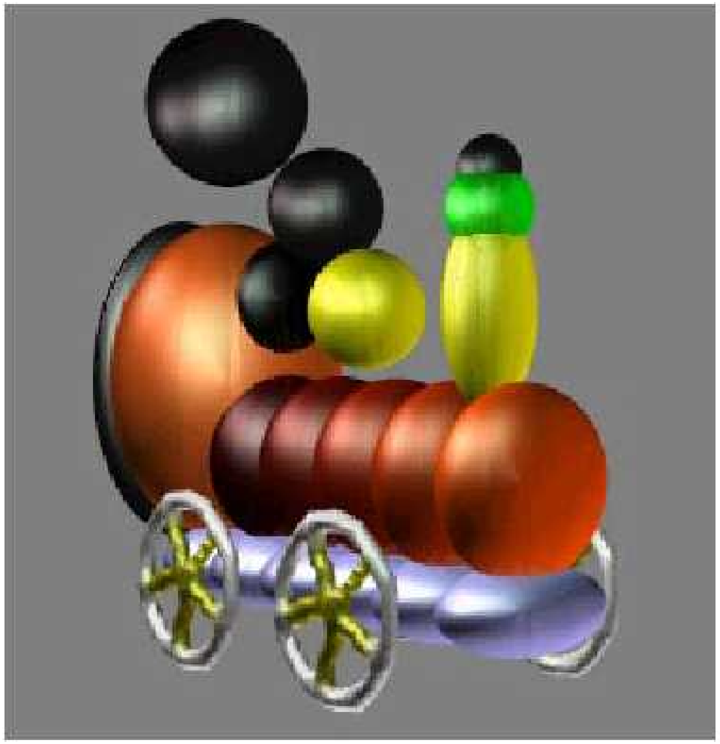
$$F(r) = 1 - (4/9)\frac{r^6}{R^6} + (17/9)\frac{r^4}{R^4} - (22/9)\frac{r^2}{R^2}$$

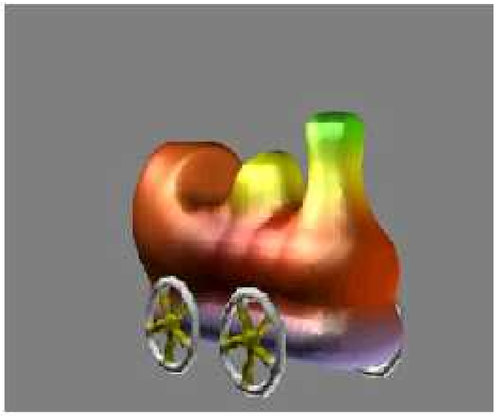


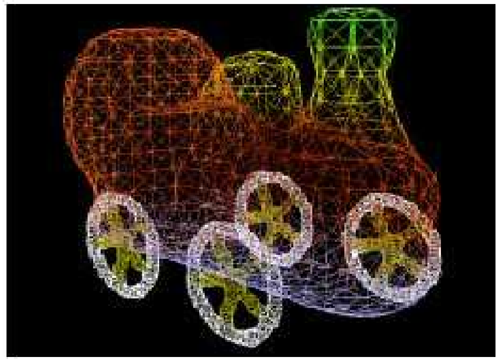
Blending and The Soft Train

1986

Polygonizer Info.

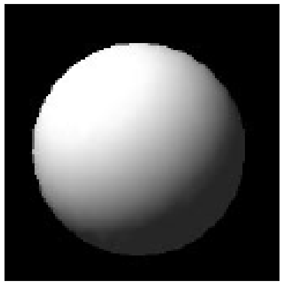


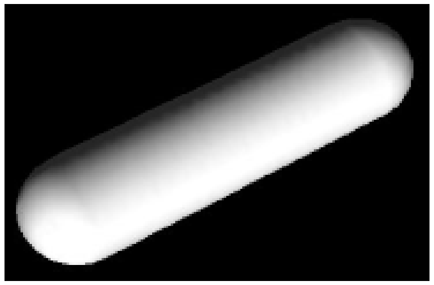


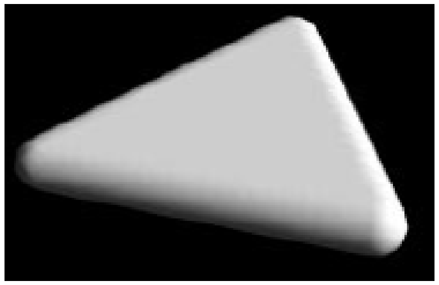


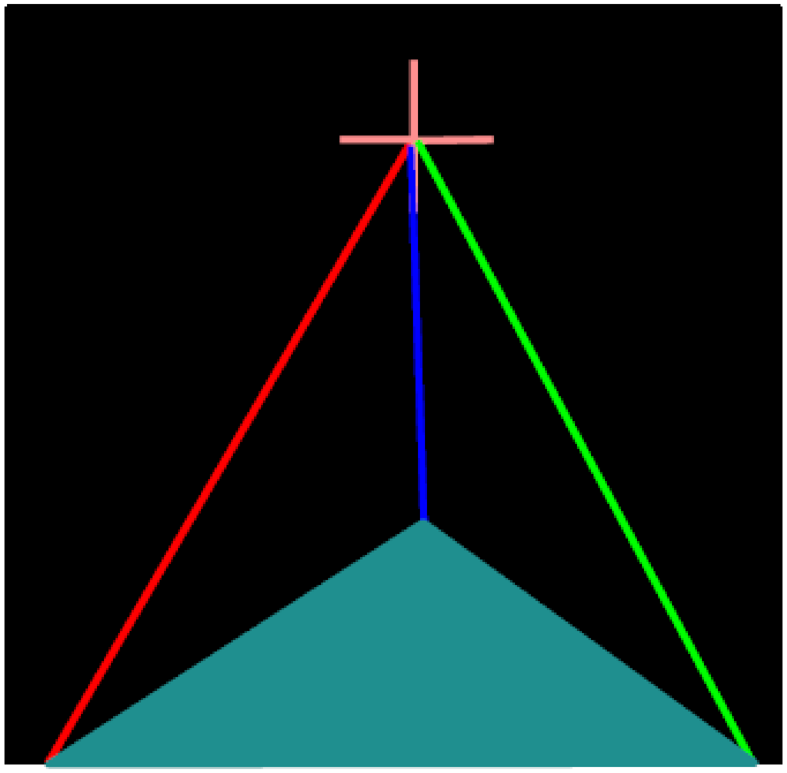
$$F_{total}(P)=\sum_{i=1}^{i=n} c_i F_i(r_i)$$

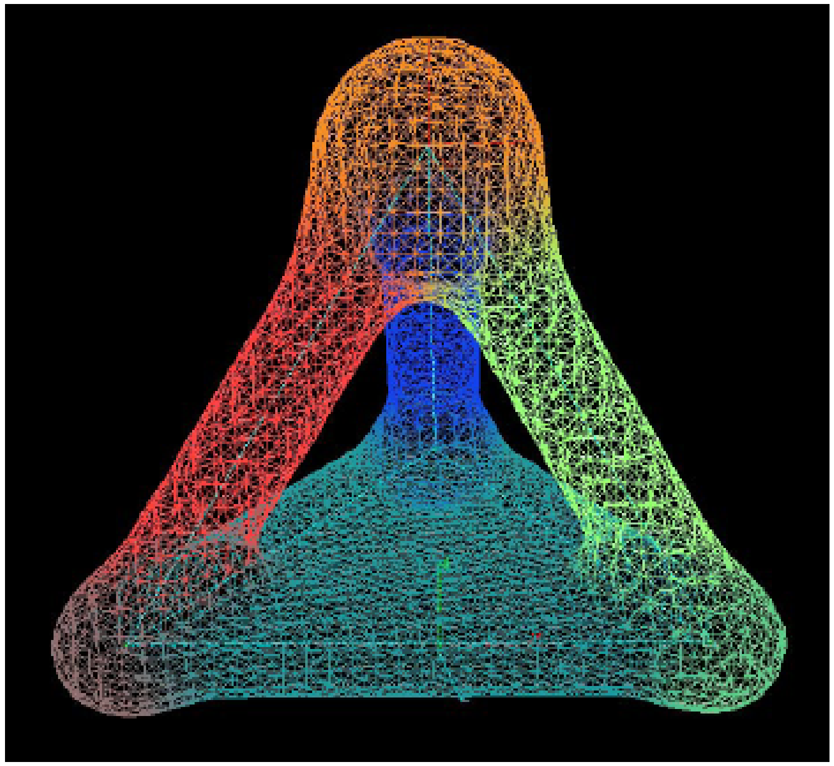
Skeletal Element Examples


Points

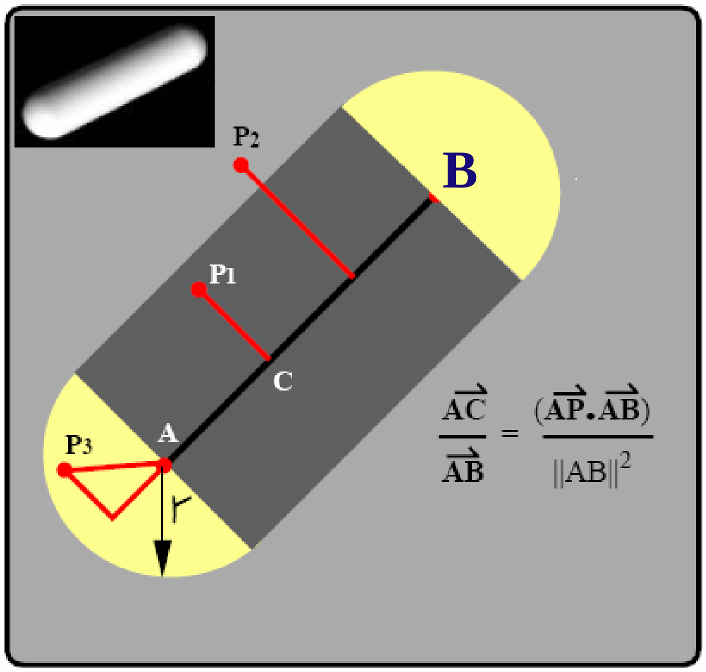

Lines


Polygons

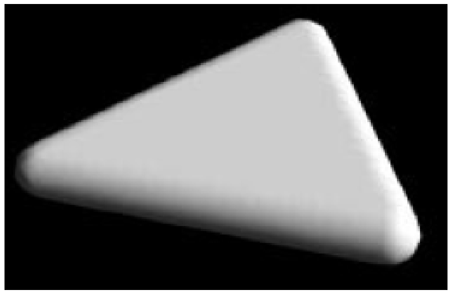




Skeletal Element – Line Skeleton

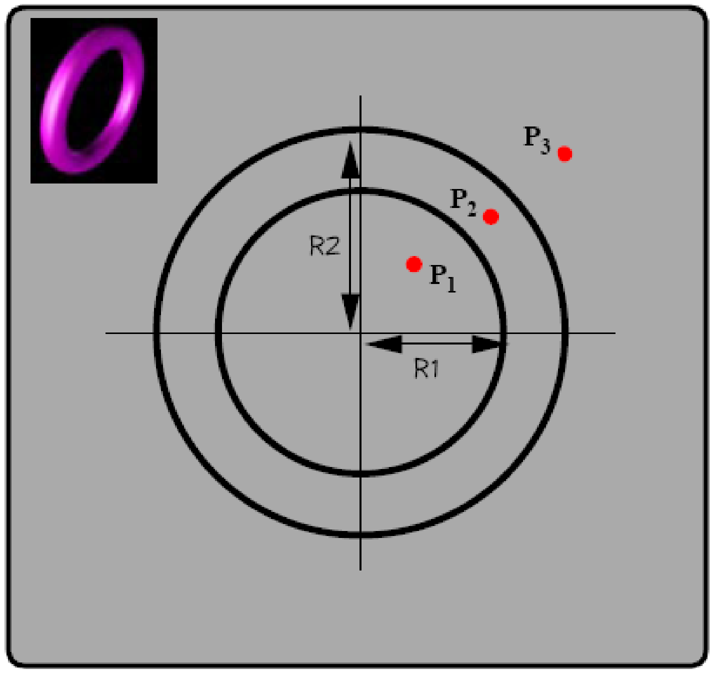


$$\frac{\overrightarrow{AC}}{\overrightarrow{AB}} = \frac{(\overrightarrow{AP}\cdot\overrightarrow{AB})}{\|AB\|^2}$$



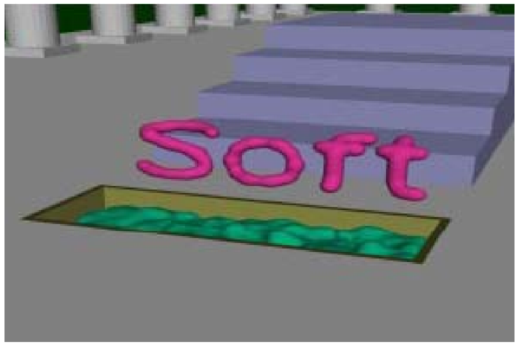
Polygon Offset Surface

Torus

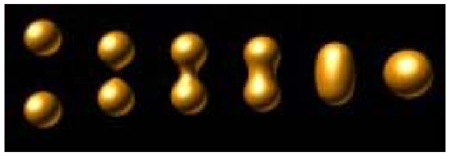


Calculating The Implicit Value

$$F_{total}(P)=\sum_{i=1}^{i=n} c_i F_i(r_i)$$

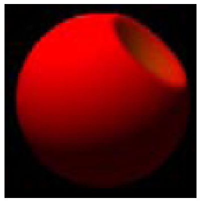


$F_{total}(P)$ is the value of the field at P

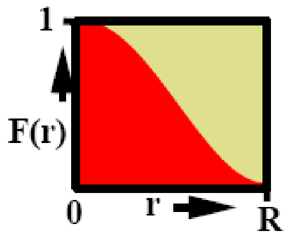


P is a point in space

n is the number of skeletal elements



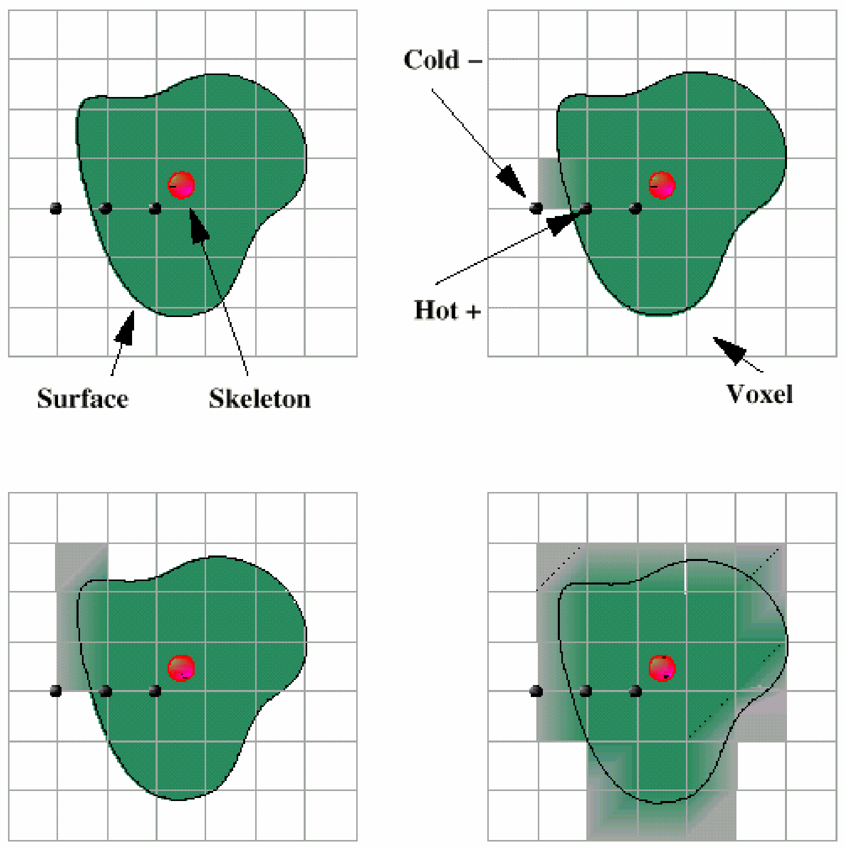
c_i is a scalar value (+/−)

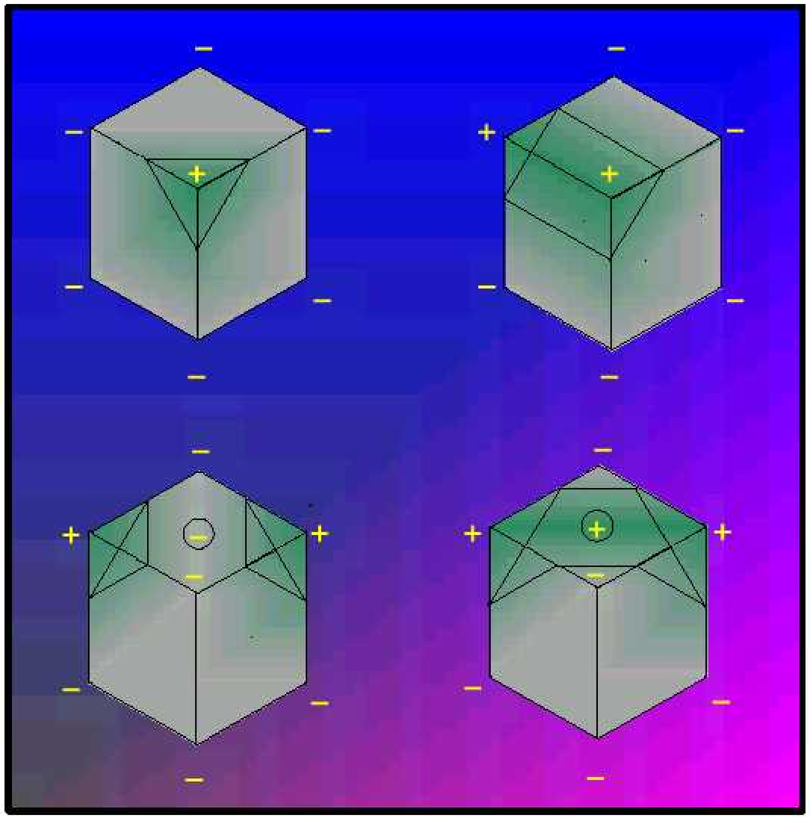


F_i is the blending function

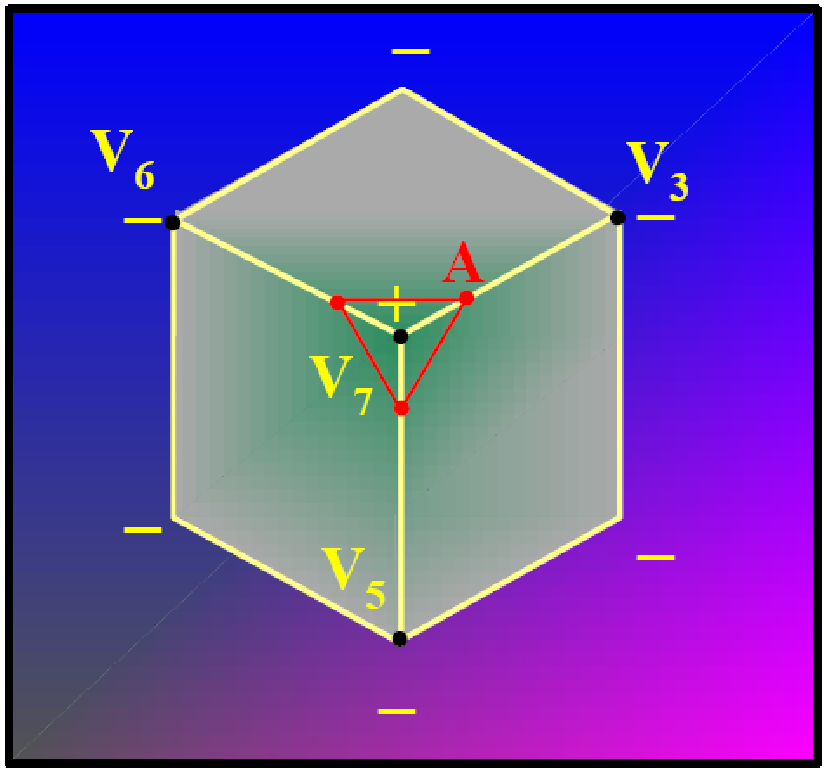
r_i is the distance from P to the nearest point Q_i on the i_{th} element

Polygonization Algorithm





Edge-Surface Intersections



Linear Interpolation
Quick and dirty (see GTR video)

$$\frac{f(A)-f(V_3)}{f(V_7)-f(V_3)} = \frac{A - V_3}{Side=1}$$

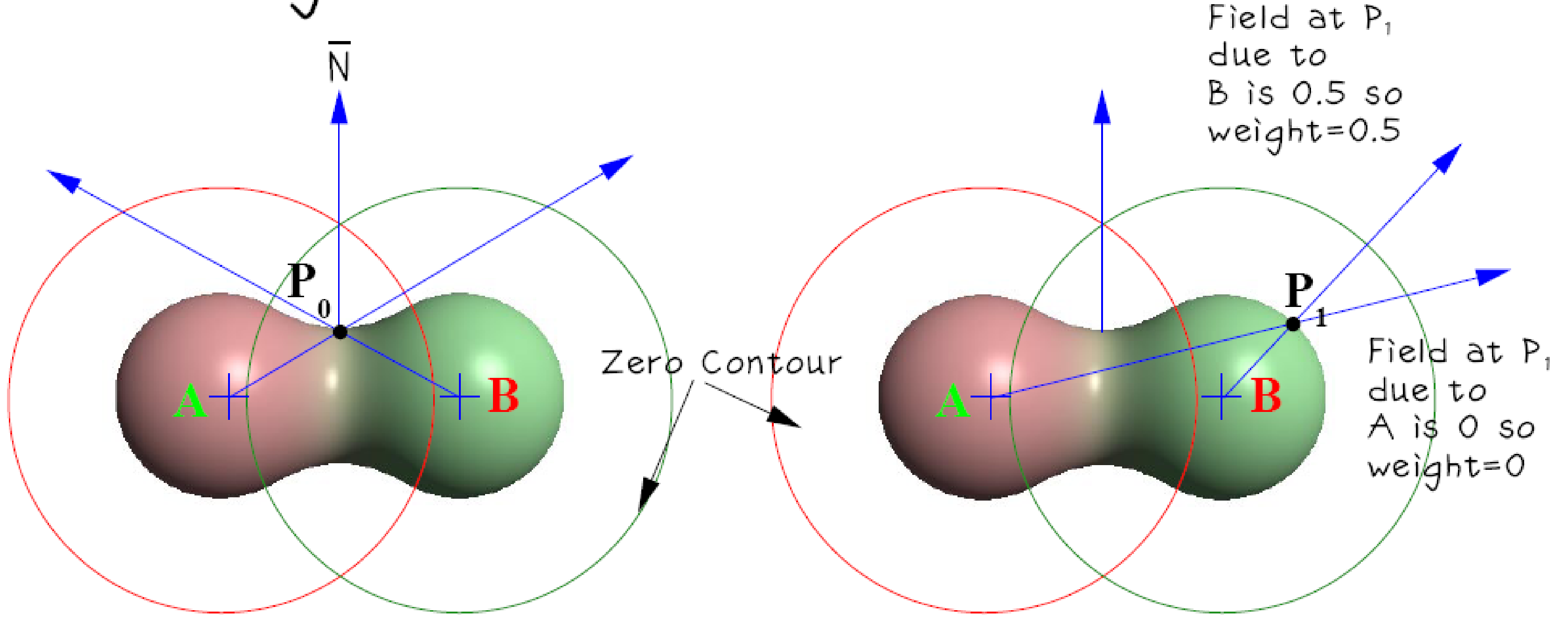
$f(A)$ = iso-value = 0.5

$$A = \frac{f(A)-f(V_3)}{f(V_7)-f(V_3)}$$

Binary Search - slower and more accurate (termination strategy)

For objects whose derivatives are known: Newtons Method (Regula Falsi)

Calculating Normals



$\bar{N}$
P_0
A
B
Zero Contour
Field at P_1 due to B is 0.5 so weight=0.5
P_1
Field at P_1 due to A is 0 so weight=0
A
B

From the gradient, the normals can be averaged weighted by field.

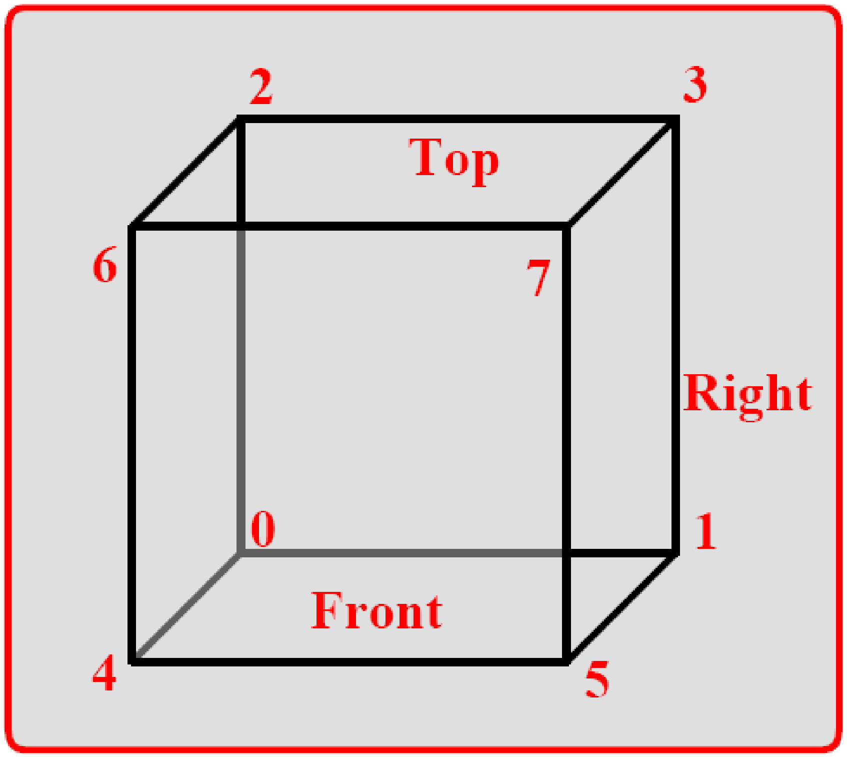
For black box functions use numerical technique:

Sample the field at P and at P+d

$$\bar{N} = \frac{f(x-\delta) - f(x+\delta)}{2\delta} \quad \frac{f(y-d) - f(y+d)}{2\delta} \quad \frac{f(z-d) - f(z+d)}{2\delta}$$

Voxel Numbering

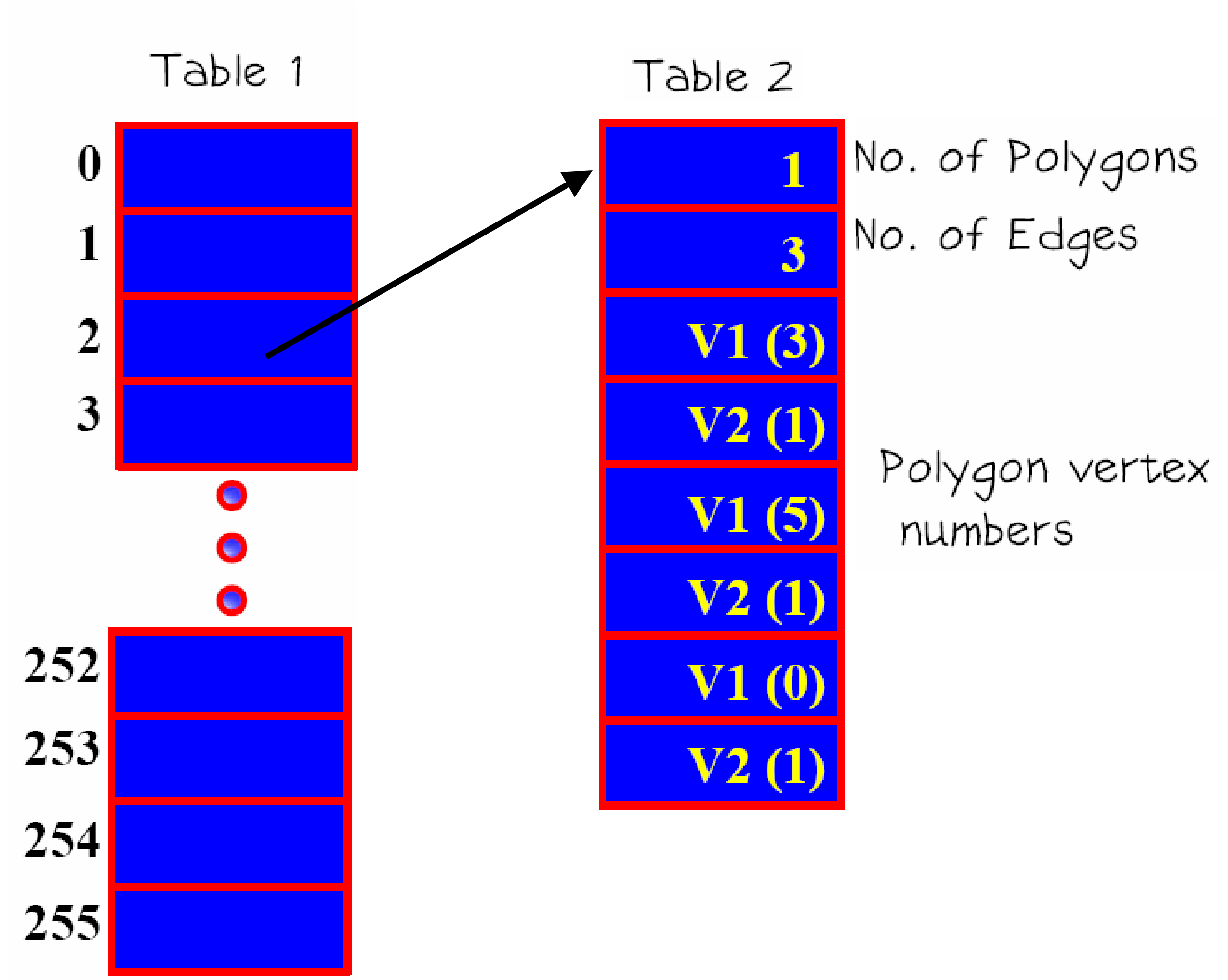
Right: vertices with bit 0 set
Top: vertices with bit 1 set
Front: vertices with bit 2 set



Vertex		If (+)
0	0	00000001
1	01	00000010
2	010	00000100
3	011	00001000
4	100	00010000
5	101	00100000
6	110	01000000
7	111	10000000

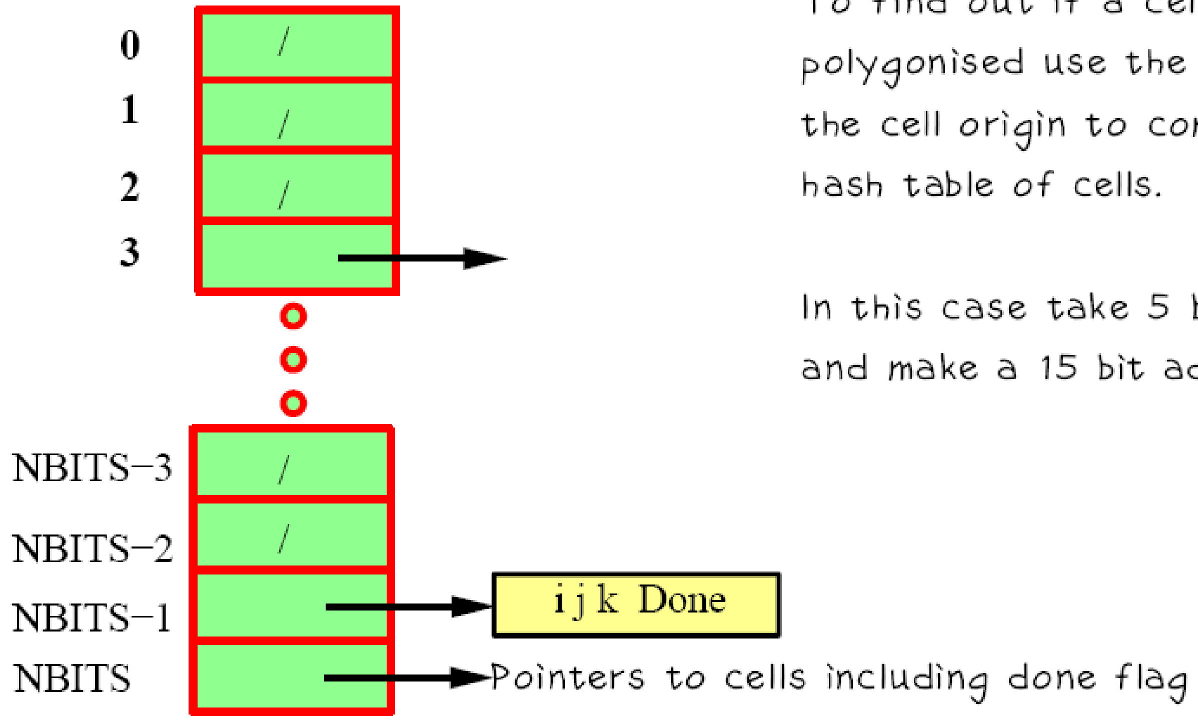
Address in table is 8 bits
taking one bit from each vertex

Polygon Tables



Hash Table

```
#define NBITS       5
#define BMASK       037
#define HASH(a,b,c)
(((a&BMASK)<<NBITS|b&BMASK)<<NBITS|c&BMASK)
#define HSIZE       1<<NBITS*3
```



To find out if a cell has already been polygonised use the integer coordinates of the cell origin to compute an address in the hash table of cells.

In this case take 5 bits out of each of x,y,z and make a 15 bit address.

The Queue (FIFO)

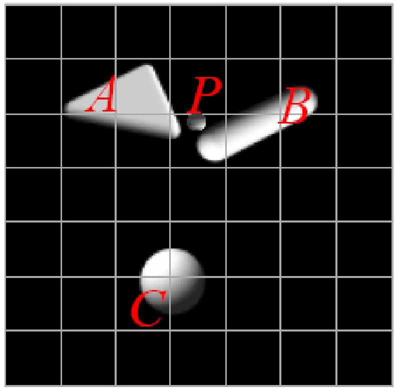
The Queue is used as temporary storage to identify the neighbors for processing (others have used a stack (LIFO list) although there is some evidence that the queue processes the cubes in a more memory efficient order). The algorithm begins with a seed cube that is marked as visited and placed on the queue. The first cube on the queue is dequeued and all its unvisited neighbors added to the queue. Each cube is processed and if it contains part of the surface output to the second phase of the algorithm. The queue is then processed until empty. The continuation algorithm proceeds as indicated in the pseudo code.

```
begin
   Set seed cube's done flag to true
   Add seed cube to the queue.
   while queue is not empty do
   begin
     remove one cube from the queue
     for each face of cube do
     begin if surface intesects face then
       begin select neighbour cube for that face
         if neighbours done flag is not true then
         begin set neighbours done flag to true
           add neighbour to queue
         end
       end
     end
     Pass cube to second stage
   end
end
```

Reducing Implicit Function Evaluations (IFE)

Measure of Efficiency:

- IFEPT (IFE per Triangle)
- IFEPT can be reduced by pre-sorting skeletal elements to voxels.
- In 2-space:

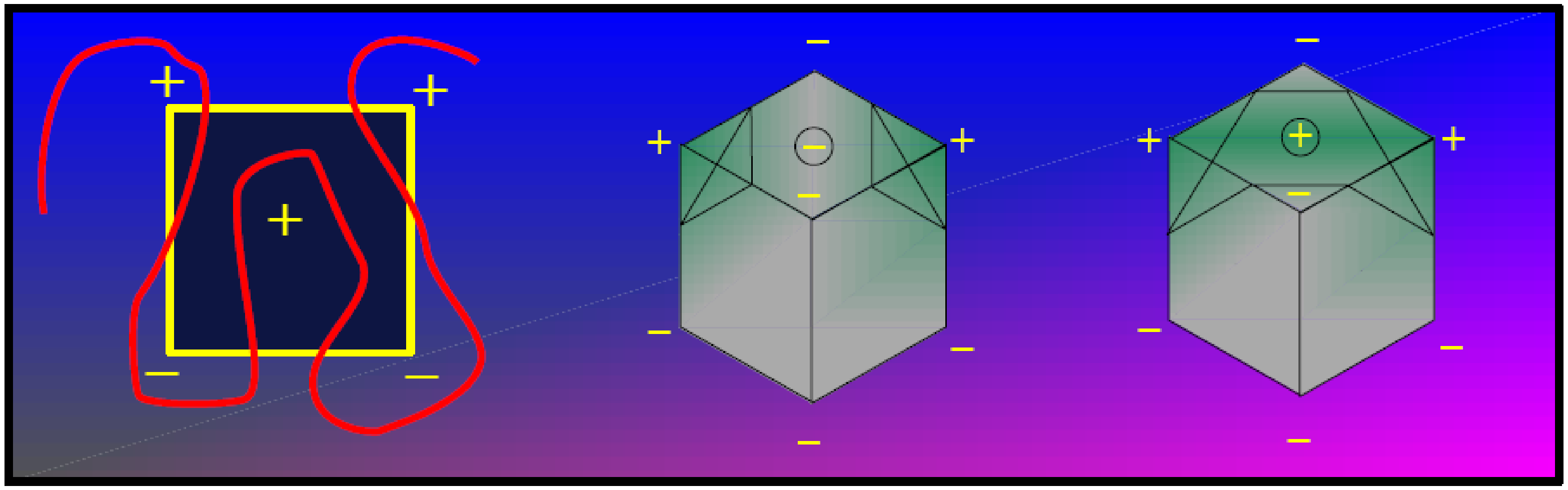


- For an arbitrary probe point (P) with skeletal elements polygon (A), line (B) and point (C).

$$F_{total}(P)=F_A(P)+F_B(P)$$

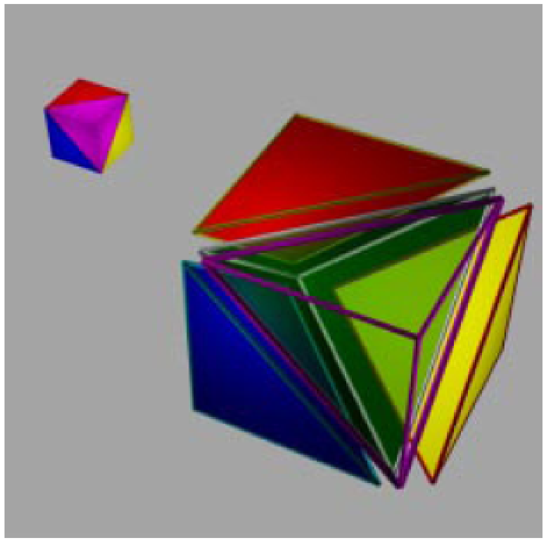
Sampling Problems

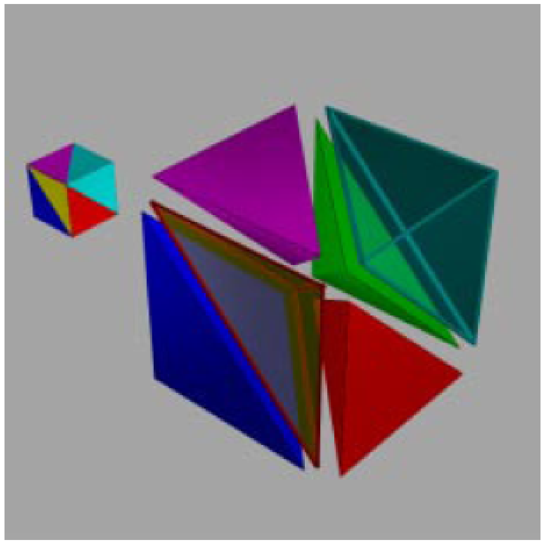
- Nothing is known about the surface between the sample points.
- Voxel Grid produces artifacts in animation



Tetrahedral Decomposition

Decomposing a cube
into 6 tetrahedra



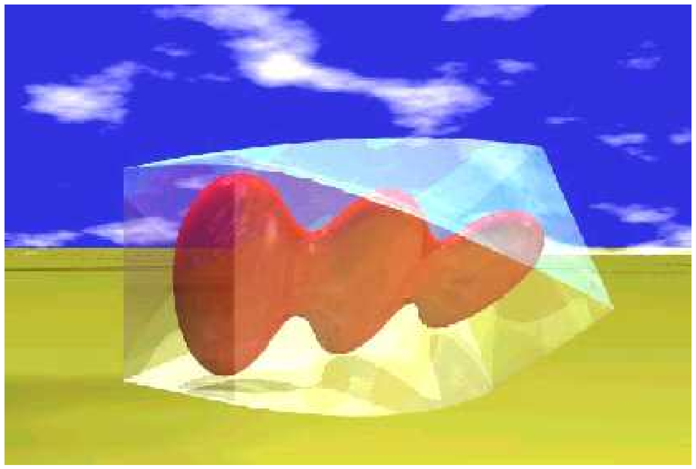


Decomposing a cube
into 5 tetrahedra

Tetrahedra avoid the ambiguity and produces correct meshes. The table is only 16 entries (4 vertices), however many more polygons result. These decompositions introduce diagonals on the cube faces, thus determining the resulting face contours. Consider two faces, although their polarity configurations are the same, the orientation of the diagonal affects the connectivity of the surface vertices. Because this orientation is arbitrarily determined by the decomposition, topological correctness is not provided. In order to maintain topological consistency, the orientation of the five-tetrahedral decomposition must alternate between face-adjacent cubes. This insures that the diagonal introduced on a cube face agrees with that of its neighbor.

Warping

$$F_{total}(P) = \Sigma\, c_i F_i(|P - Q_i|)$$



Warp function w:

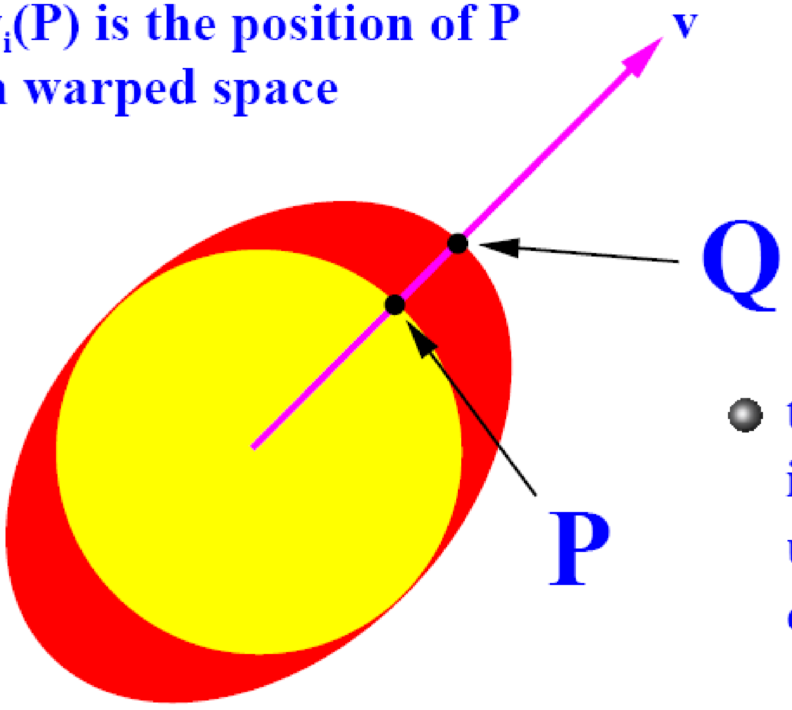
$$F_{total}(P) = \Sigma\, c_i F_i(|w(P) - Q_i|)$$

E.g. The Vector Warp

displace the evaluation point

$r_i = F_i(P) = \text{dist}(w_i(P), Q_i)$

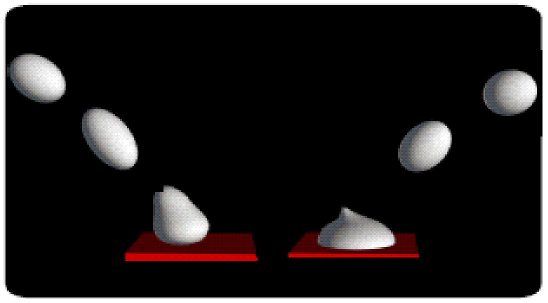
- $w_i(P) = P - \frac{v}{||v||}(v \cdot P)$
- $w_i(P)$ is the position of P in warped space

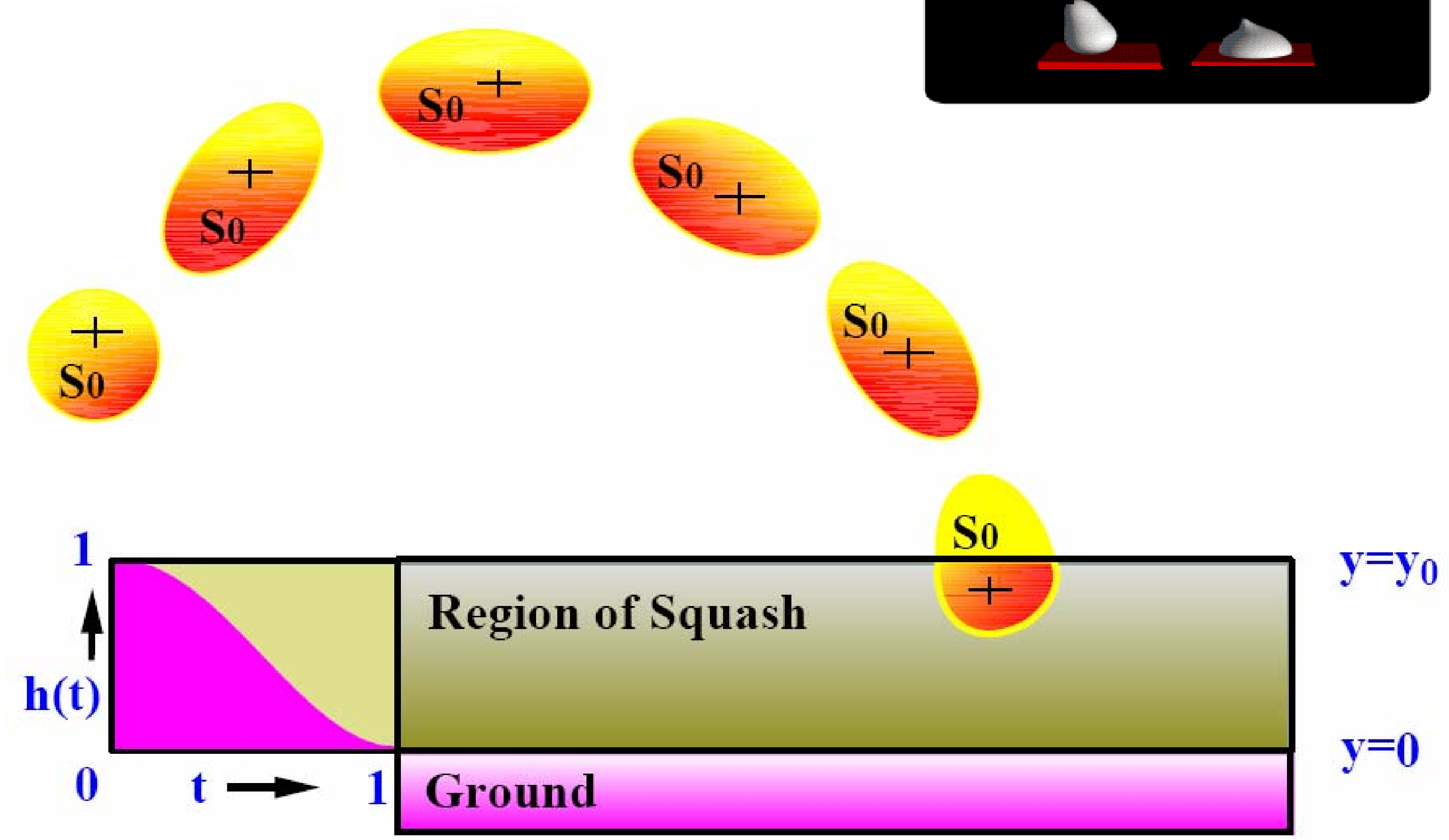


- the value returned for Q is the value of P in unwarped space (in this case the contour value)

Warping

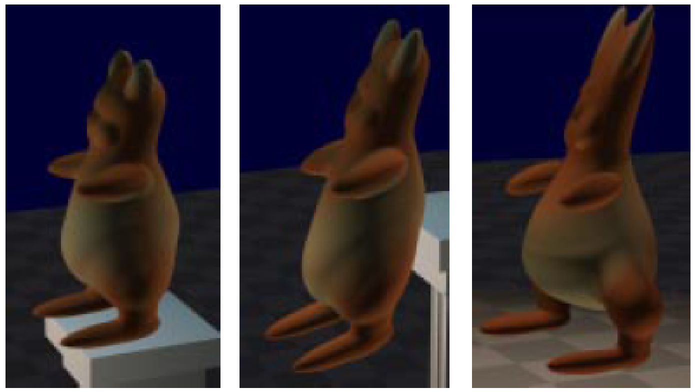
The bouncing ball example:



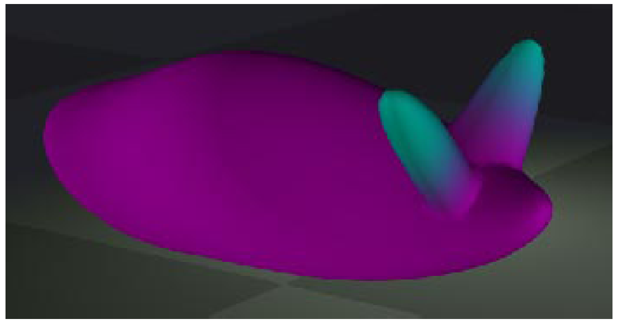


Warping

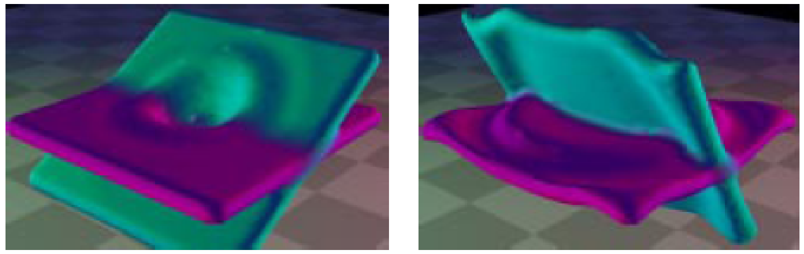
Applying the squash warp to a bear:



Non−linear periodic warp:

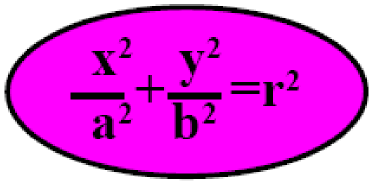


Wave simulation as a warp applied over time:

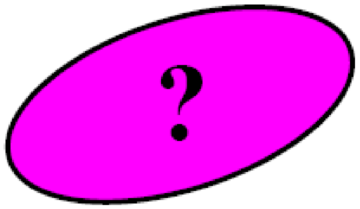


Affine Transformations as Warps

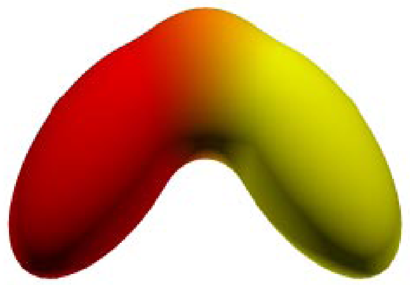
E.g.



$$\frac{x^2}{a^2}+\frac{y^2}{b^2}=r^2$$



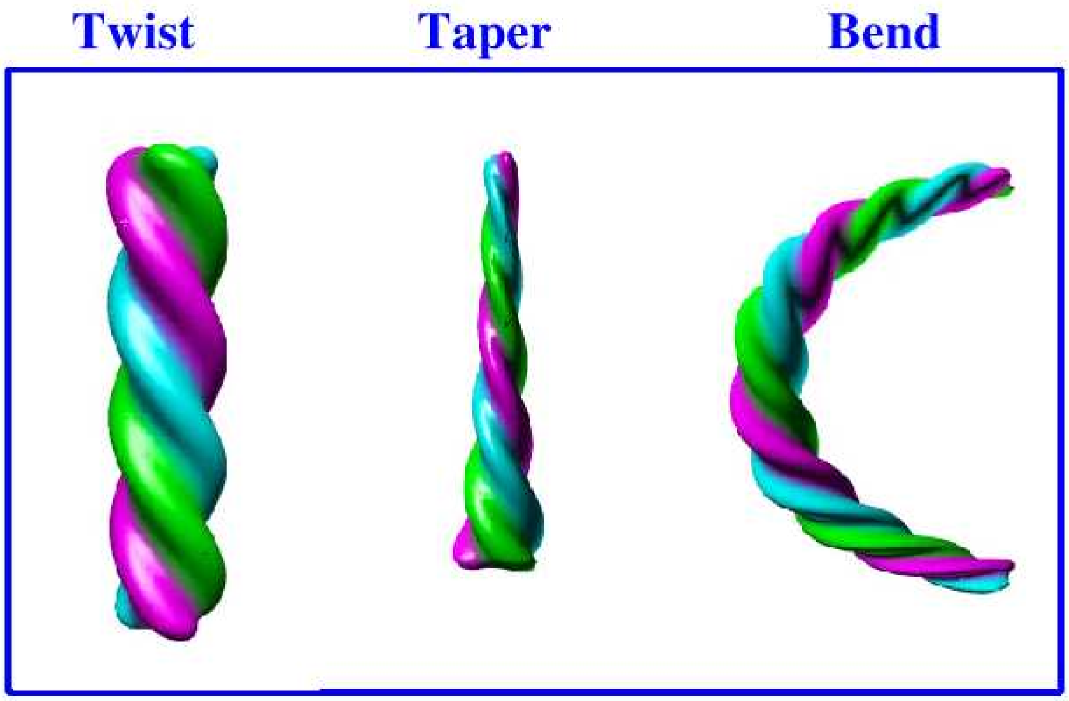
$$F_{\text{total}}(P)=\Sigma\ c_i F_i(|\text{rotate}(P)-P_i|)$$



Ellipsoids defined in the canonical position and then rotated by warping. For efficiency these can be concatanated in the normal way.

Barr Operators

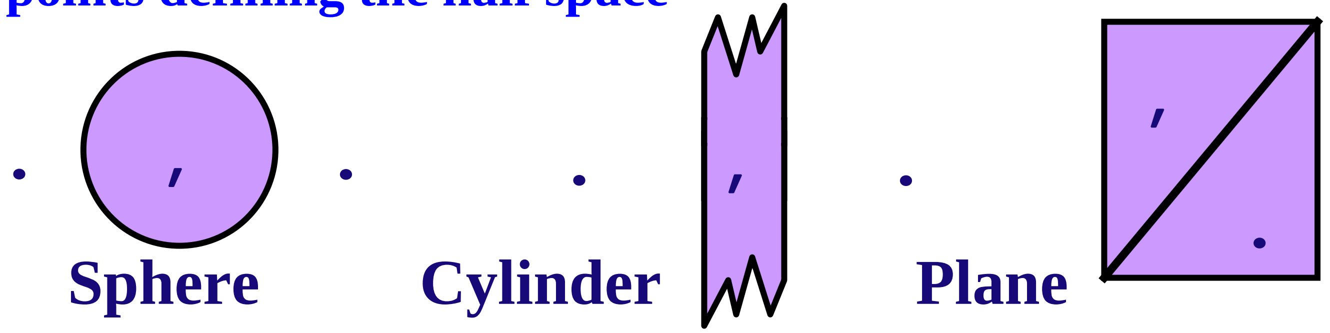
The Barr operators:



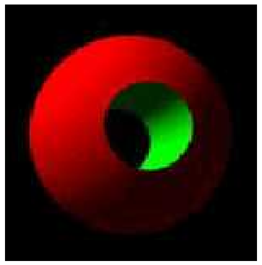
Constructive Solid Geometry (CSG)

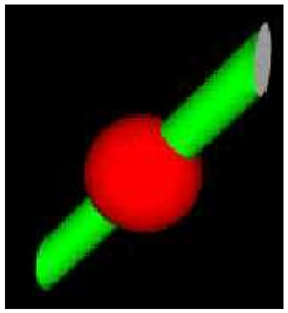
Primitives are combined using boolean set operations:
Union, Intersection, Difference. Each primitive represents a half space, ie the set of points defining the half space

E.g.



Boolean expression (u= union, d= difference, i= intersection)
d(sphere, cylinder)   u(sphere, i(i(cylinder, plane1), plane2))



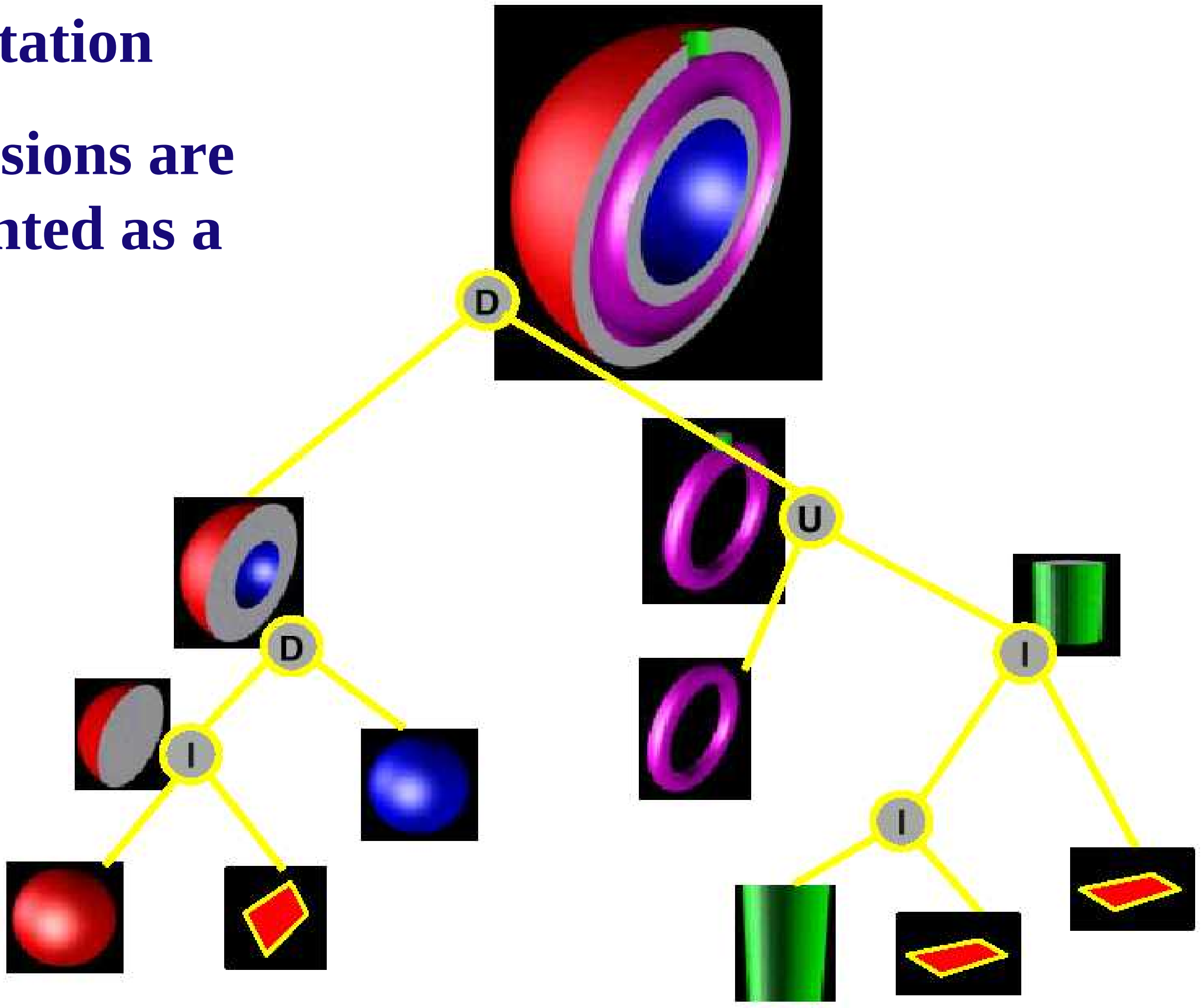


The cylinder is infinite in extent it is first intersected with two half space planes.

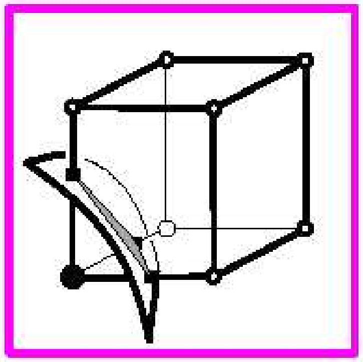
CSG Tree

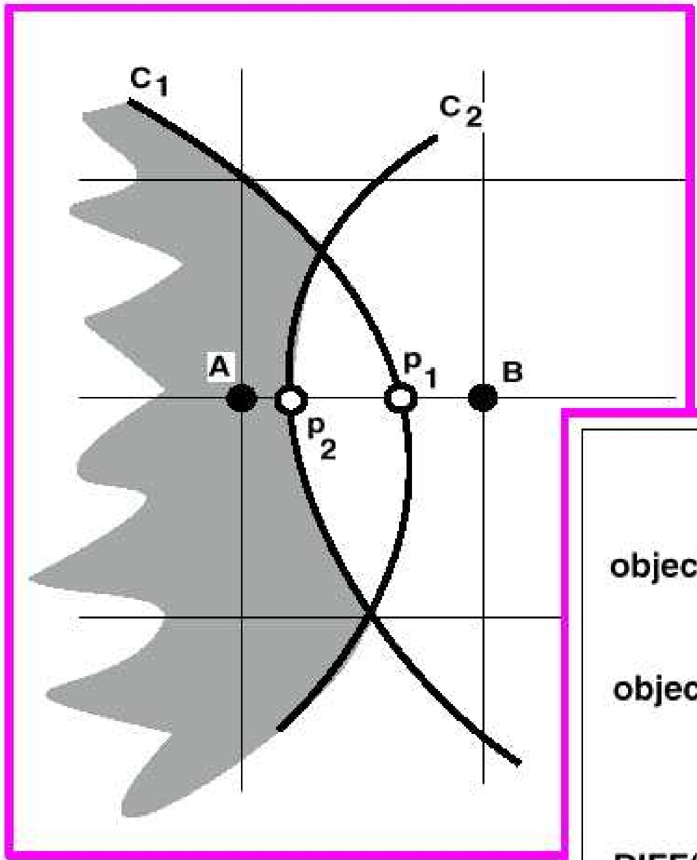
CSG Implementation

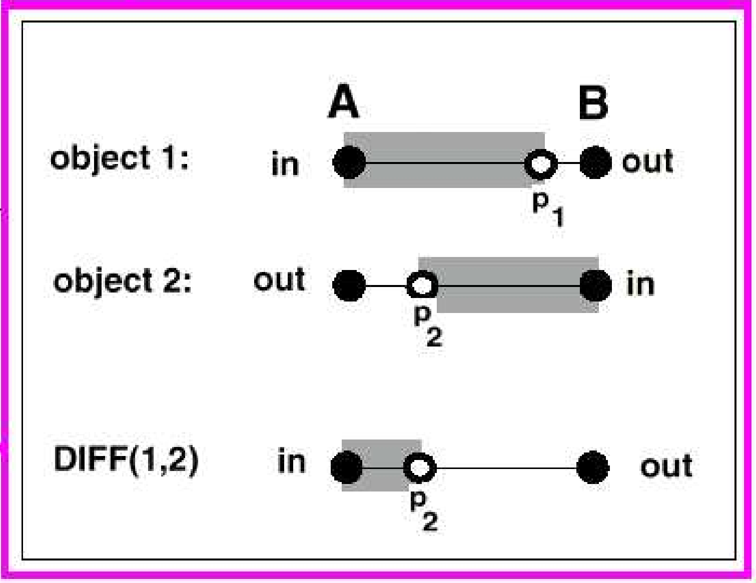
Boolean Expressions are usually represented as a binary tree.



CSG Intersections with Voxels







CSG Intersection Value

Boolean Operations

Union and intersection of primitives, A and B may be respectively defined as a composition of the field values, F_A , F_B

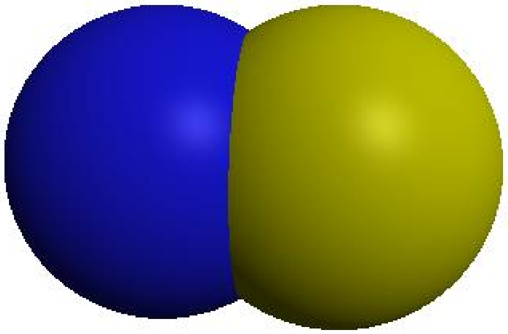
$F_A \times F_B = \max(F_A, F_B)$

$F_A \text{Ø} F_B = \min(F_A, F_B)$

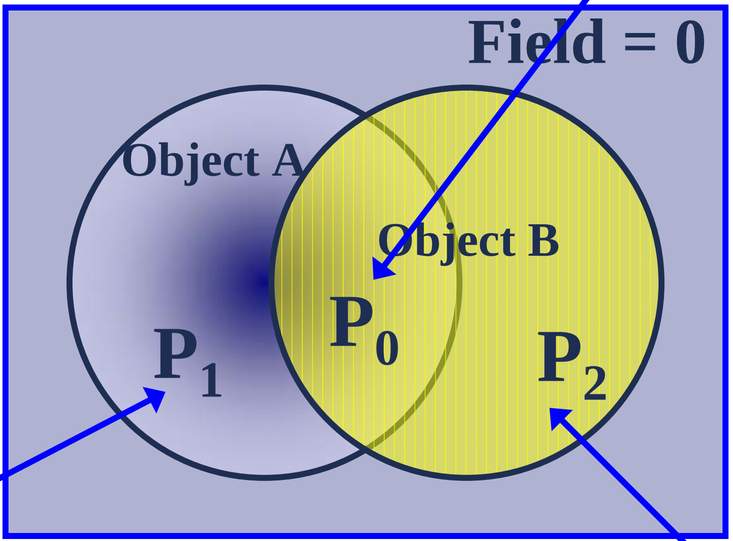
Difference use . min (F_A, F_B)
(. in this case inverts inside and outside)

CSG - Min and Max

Union



$f_{A|B}(p_0)$=Max($f_A(p_0)$, $f_B(p_0)$)
Depending on position of p_0

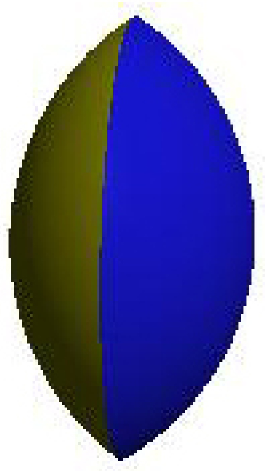


$f_A(p_1)$ =Max($f_A(p_1)$, $f_B(p_1)$)

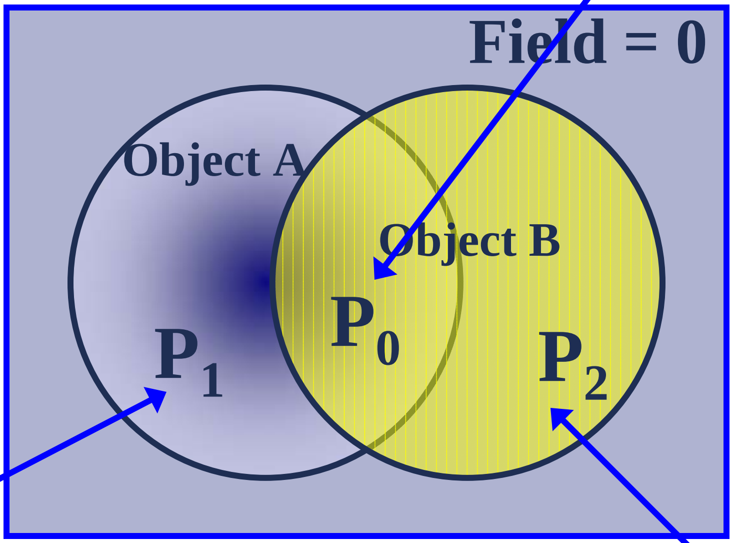
$f_B(p_2)$=Max($f_A(p_2)$, $f_B(p_2)$)

CSG - Min

Intersection



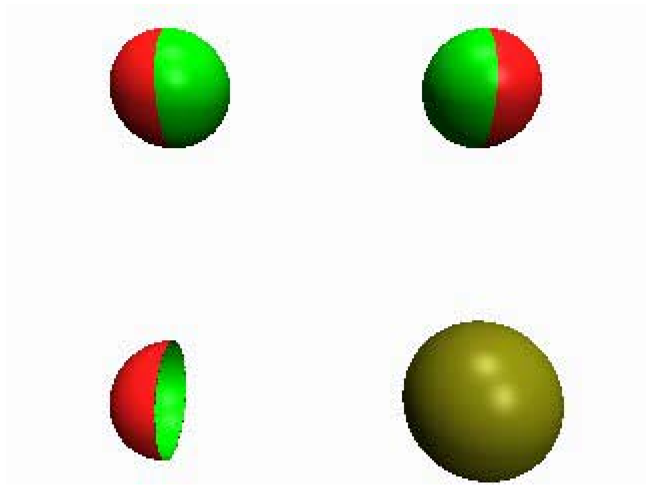
$f_{A|B}(p_0)$=Min($f_A(p_0)$, $f_B(p_0)$)
Depending on position of p_0



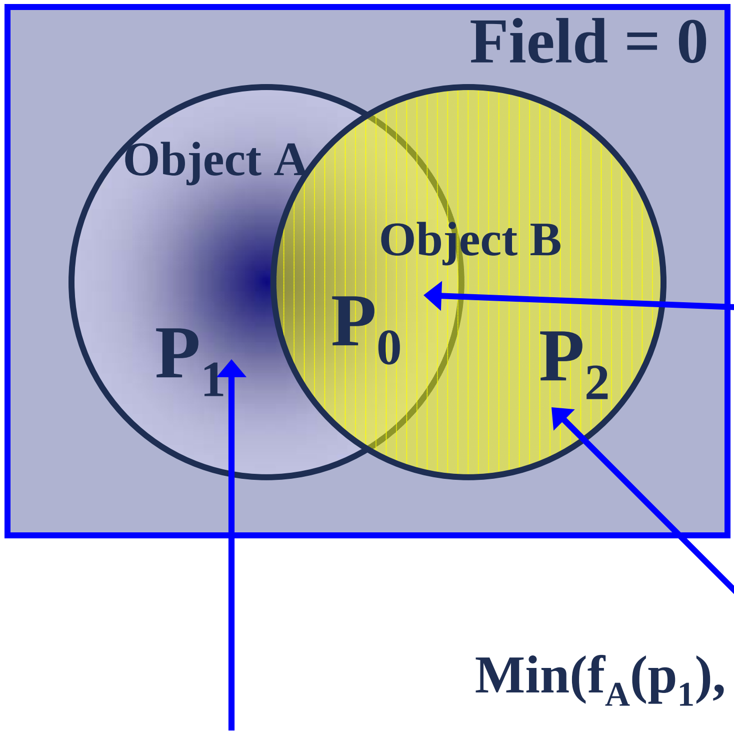
$f_B(p_1)$ =Min($f_A(p_1)$, $f_B(p_1)$)=0

$f_A(p_2)$=Min($f_A(p_2)$, $f_B(p_2)$)=0

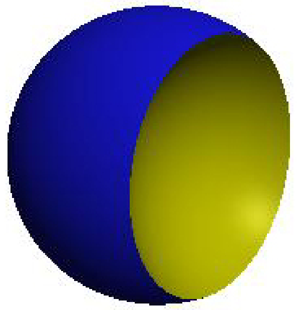
CSG - Min



Difference



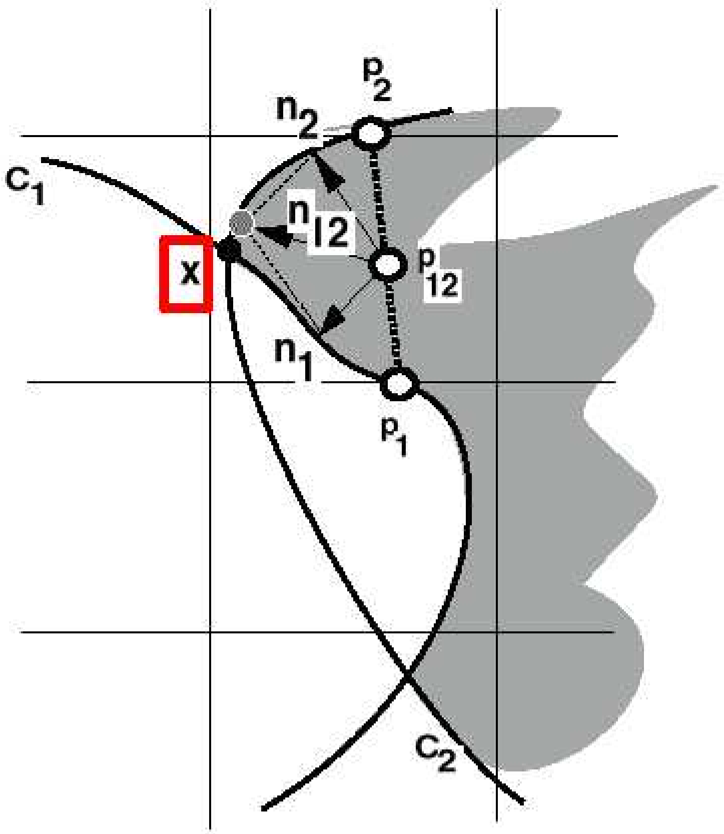
$\mathrm{Min}(f_A(p_0), f_B(p_0)) = 1 - f_{A|B}(p_0)$
Depending on position of p_0



$\mathrm{Min}(f_A(p_1), 1 - f_B(p_1))=0$

$\mathrm{Min}(f_A(p_1), 1 - f_B(p_1))=f_A(p_1)$

Polygonization Problems



X is the true intersection point for C_1 and C_2

Segment $P_1 P_2$ is far from x.

Estimate for x s. t. f_1)x*> f_2)x*> !1

We can apply a first order Taylor expansion to the difference : $n_{12} = x - P_{12}$

1 > f $_1$)x*= f $_1$)P$_{12}$, n$_{12}$*> f$_1$)P$_{12}$*!, !)n$_{12}$ -!◆ f$_1$) P$_{12}$ **
1 > f$_2$)x*> !f$_2$)P$_{12}$, n$_{12}$*> f$_2$)P$_{12}$*!, !)n$_{12}$-!◆ f$_2$) P$_{12}$ **

Iterating to the Surface

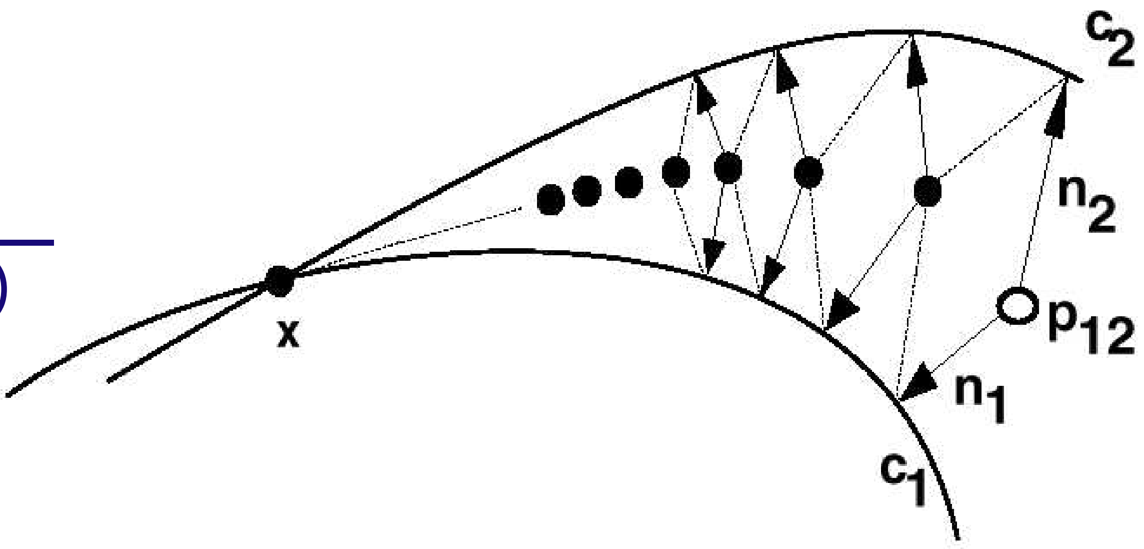
$$\lambda_1 = \frac{-f_1}{(\blacklozenge f_1, \blacklozenge f_1)}$$

so

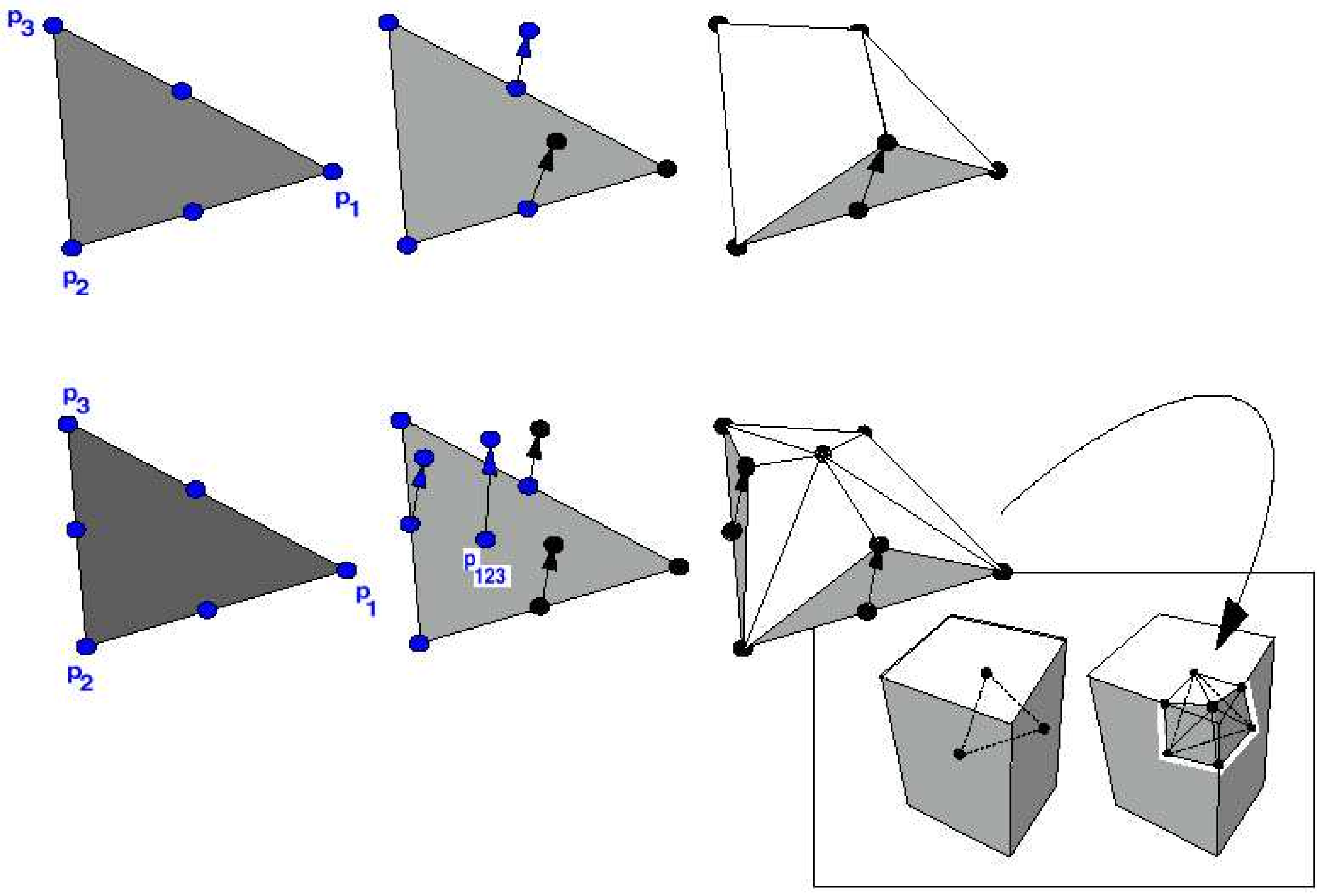
$$n_1 = \frac{-f_1 \blacklozenge f_1}{(\blacklozenge f_1, \blacklozenge f_1)}$$

and similarly

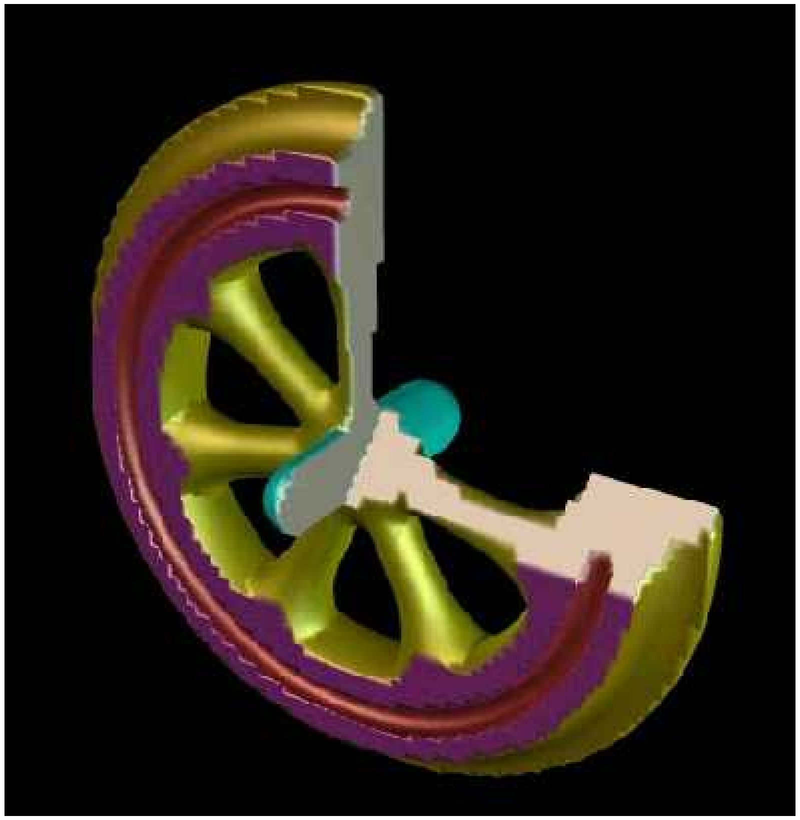
$$n_2 = \frac{-f_2 \blacklozenge f_2}{(\blacklozenge f_2, \blacklozenge f_2)}$$

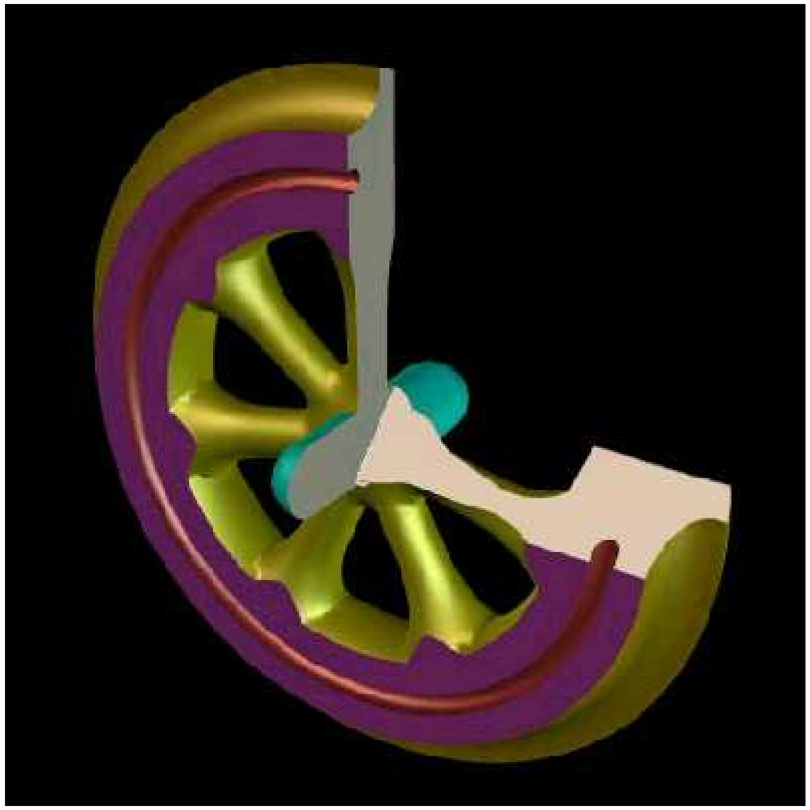


Adaptive Polygonisation



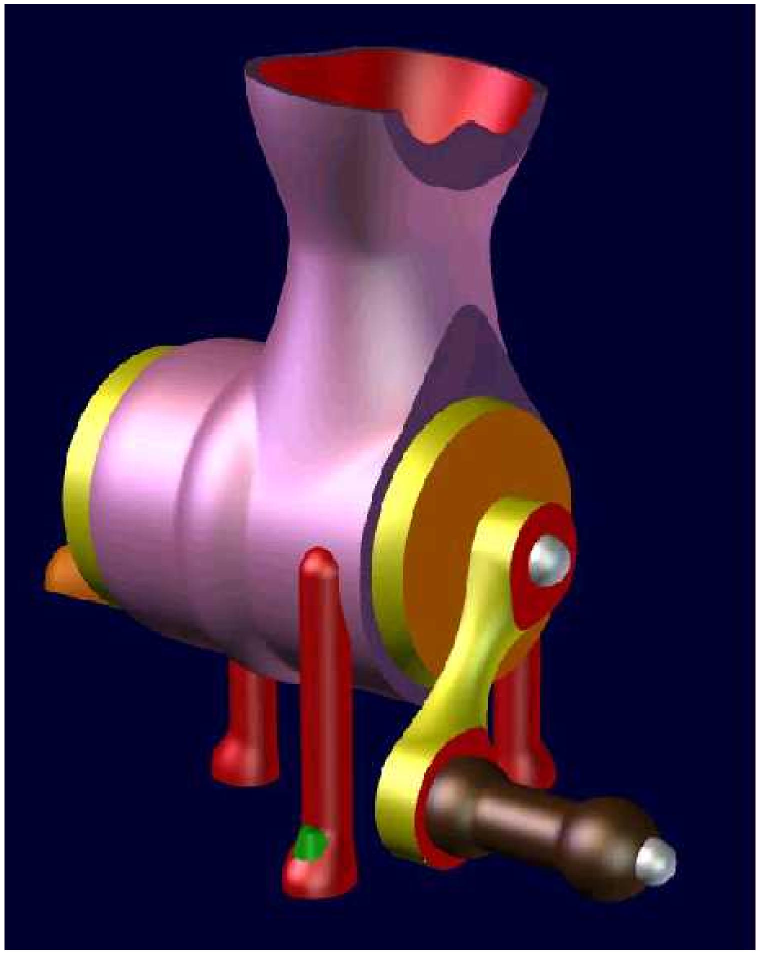
CSoftG Wheels

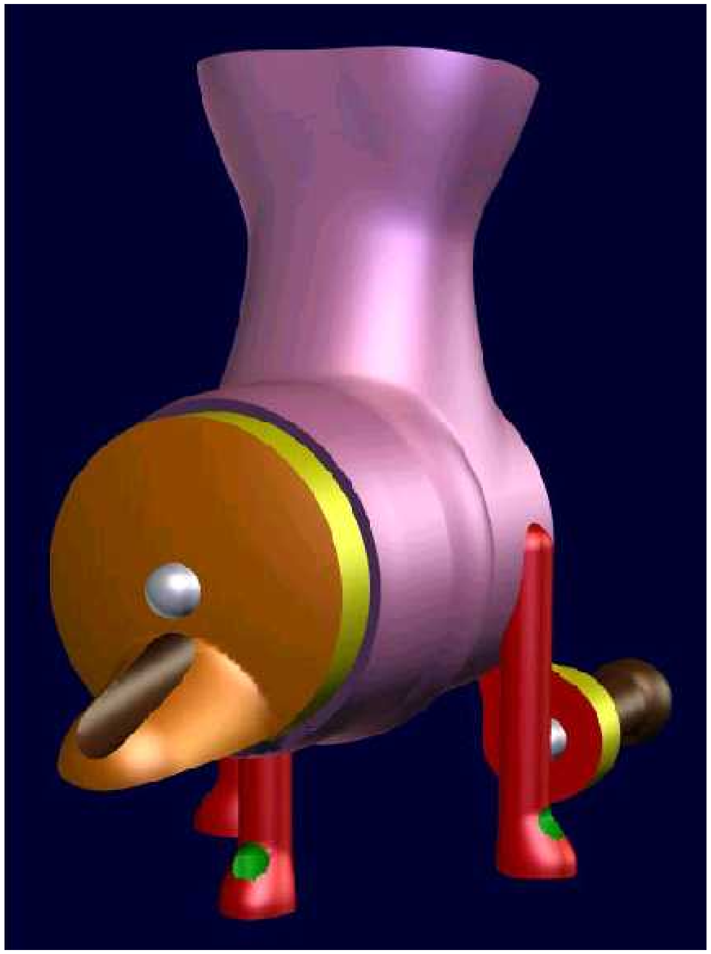




Csoft Wheel before and after removal of artifacts

Canmore Coffee Grinder

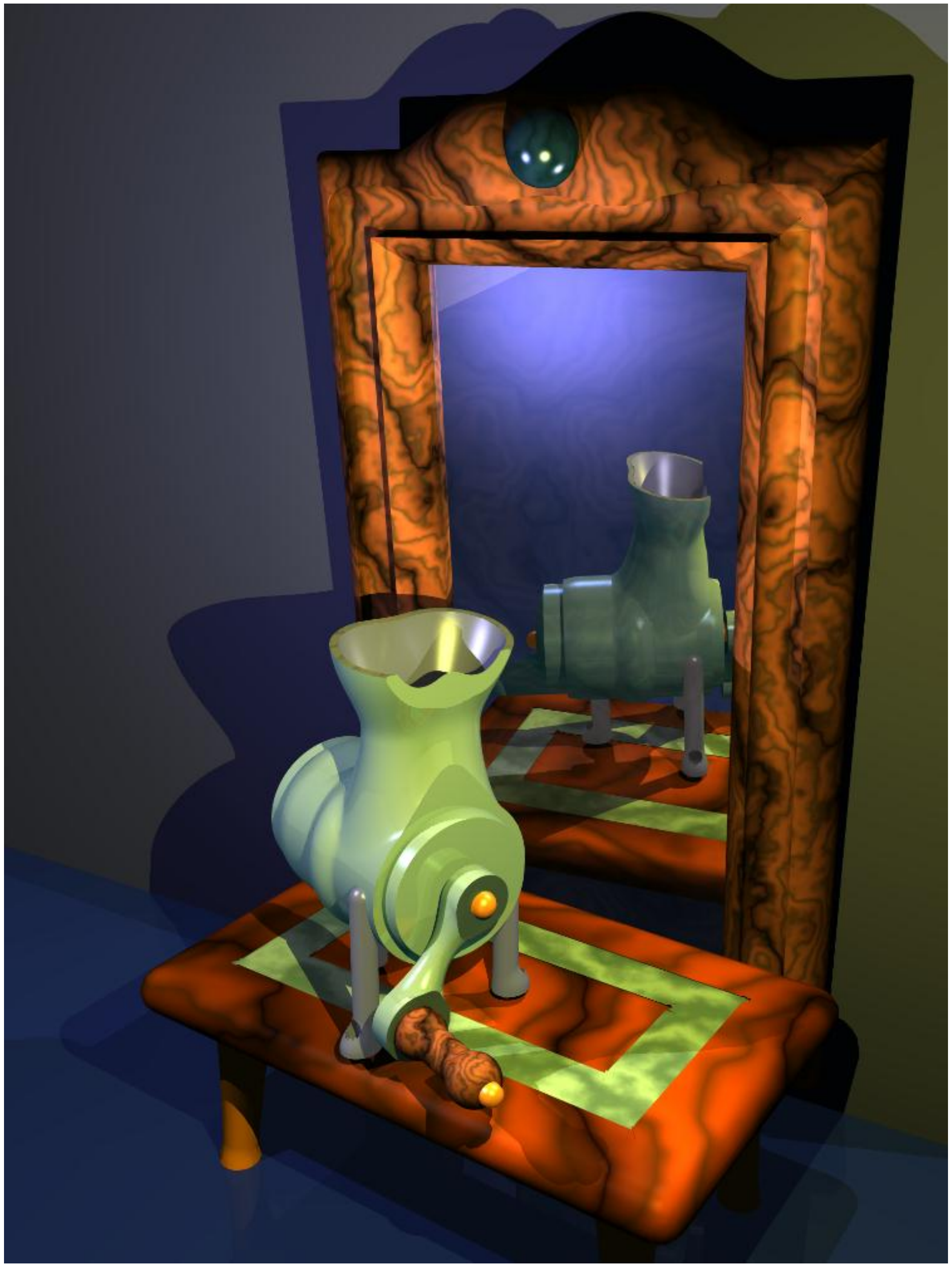




Ray Traced Canmore Coffee Grinder

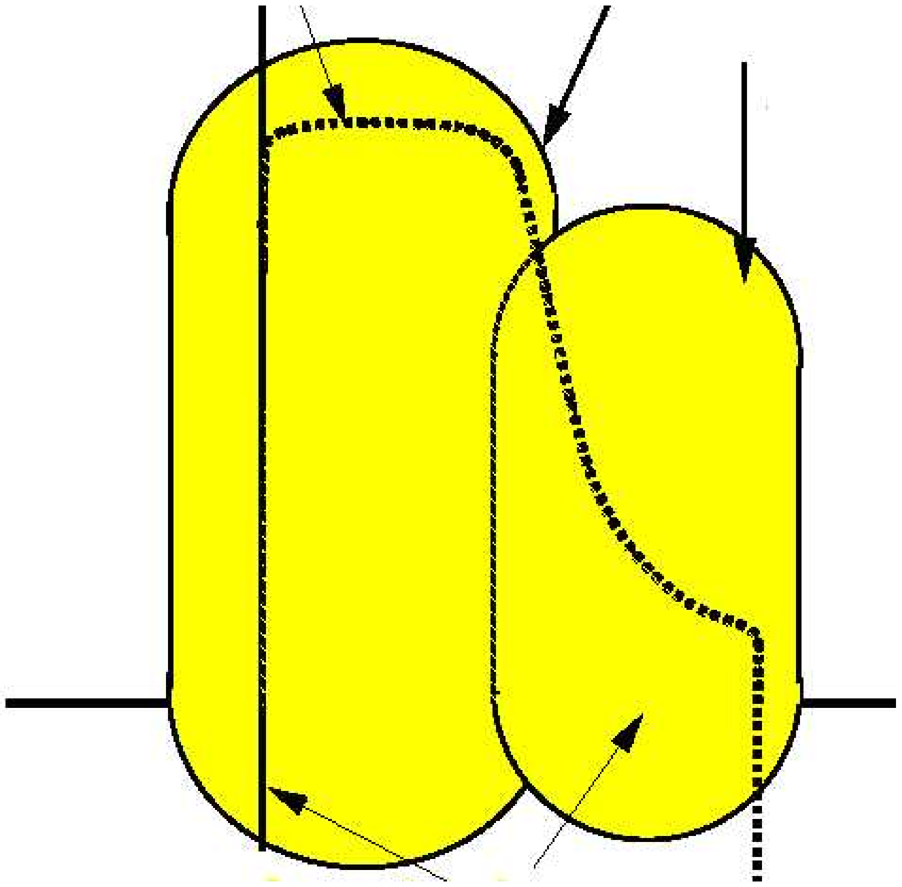
by

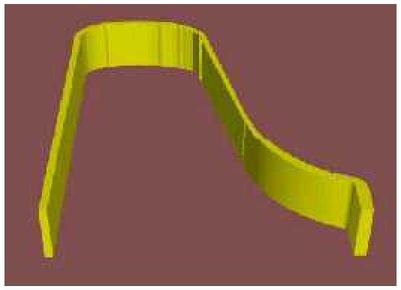
Kees van Overveld
and
Brian Wyvill



Building the Piano

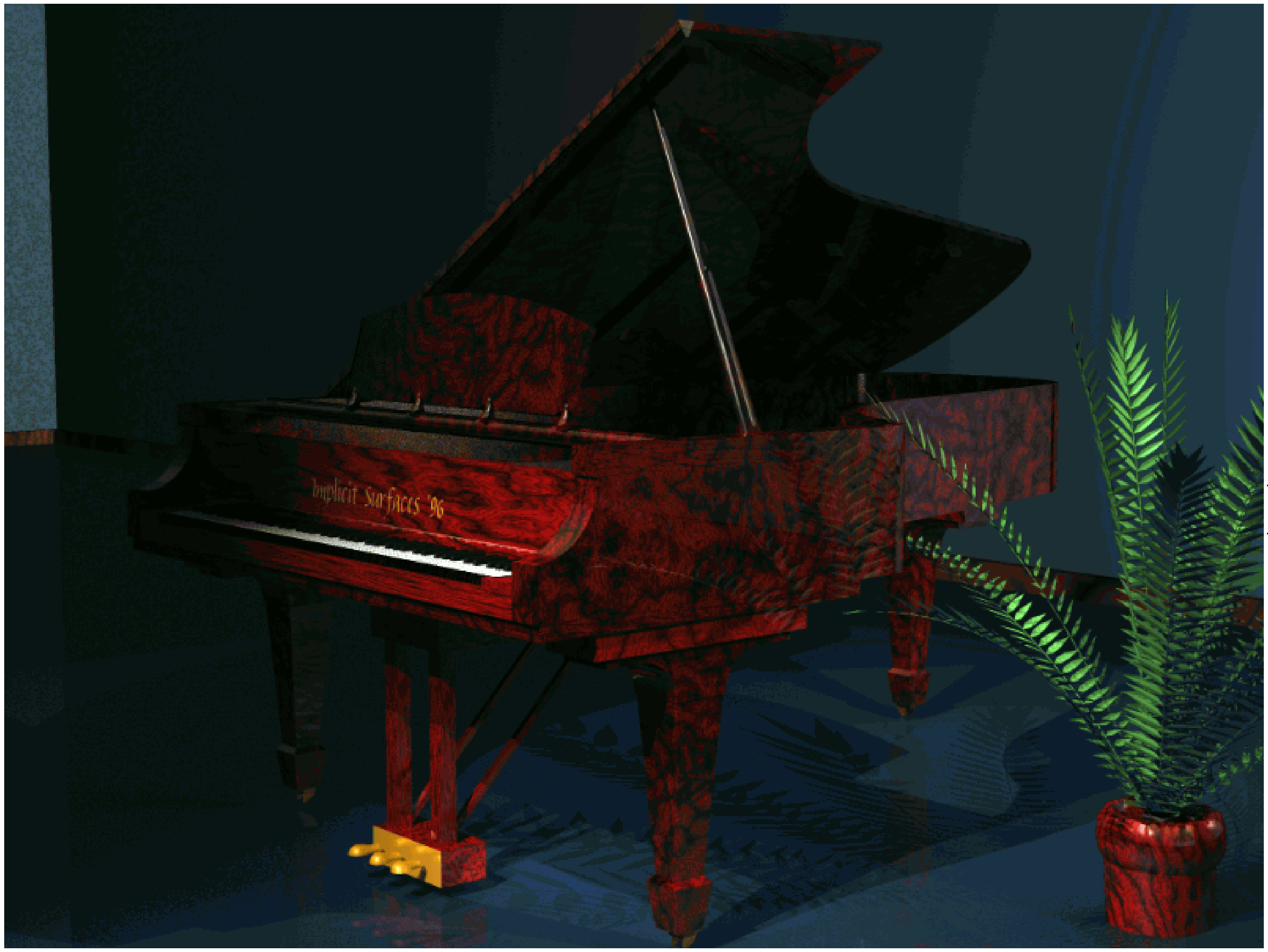
Parametric Bounding Curve





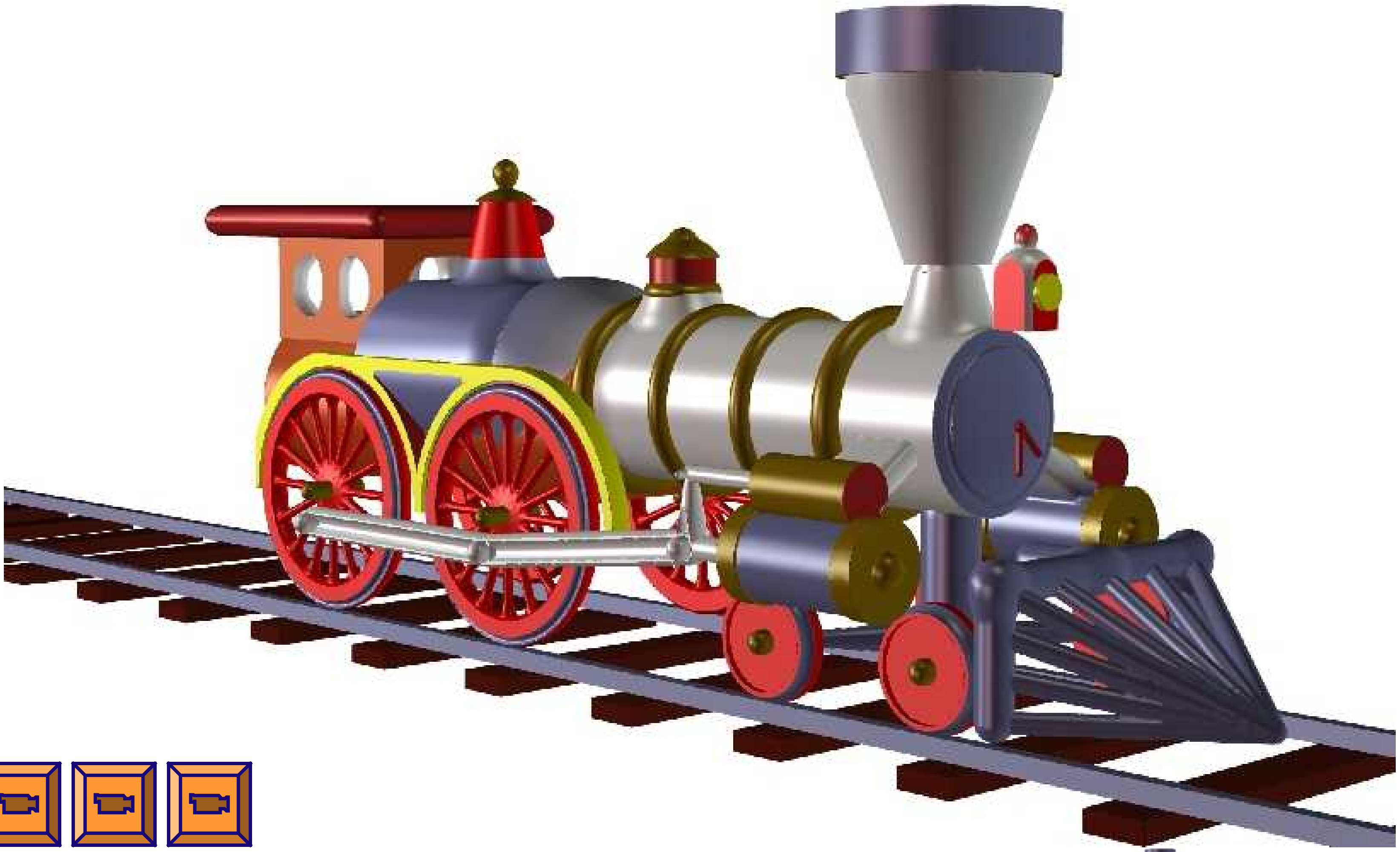
Cylinders intersected with bounding plane and parametric curve

Model of 9ft. Steinway Concert Grand

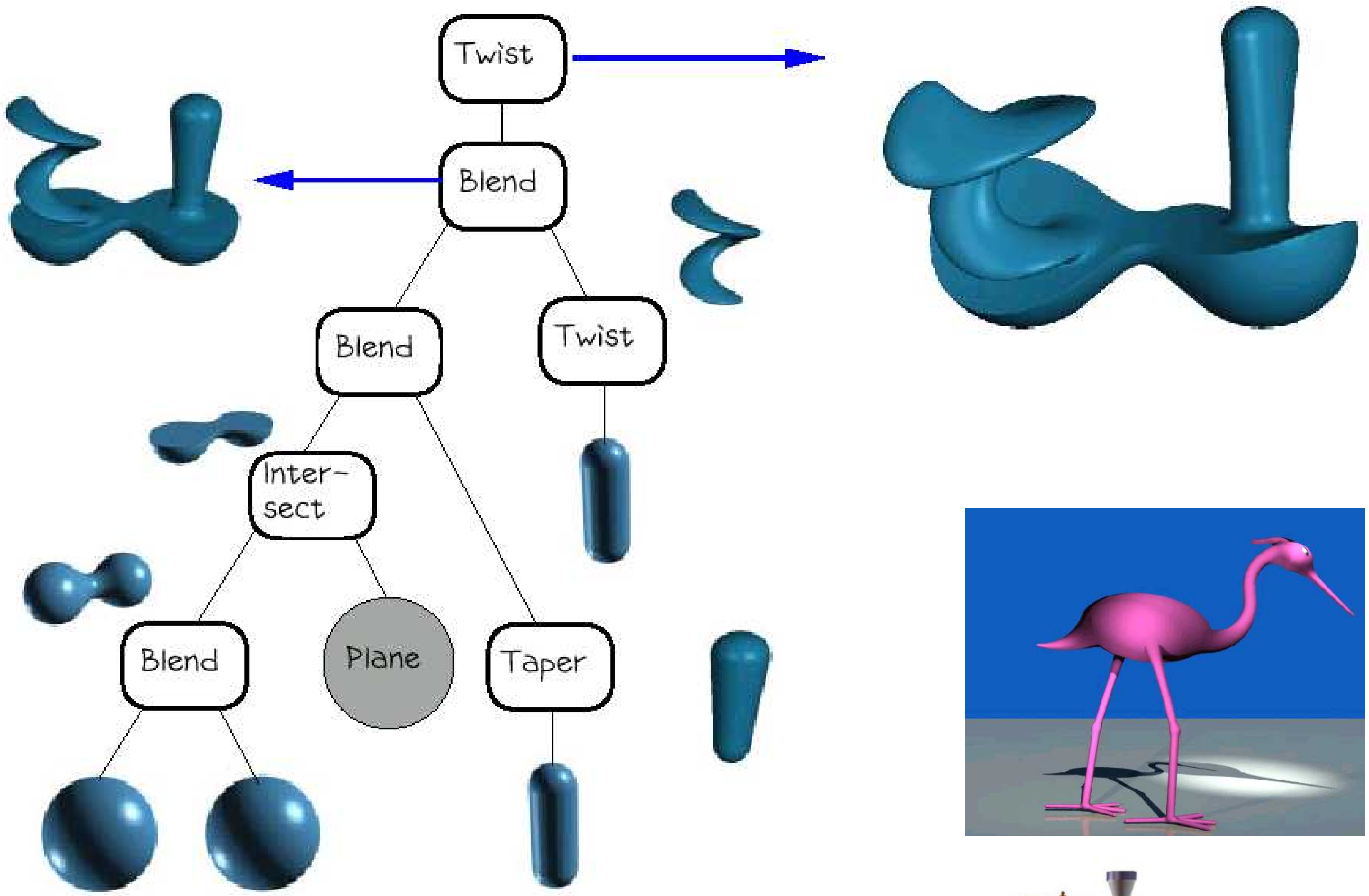


Plant by Dr. Prusinkiwicz

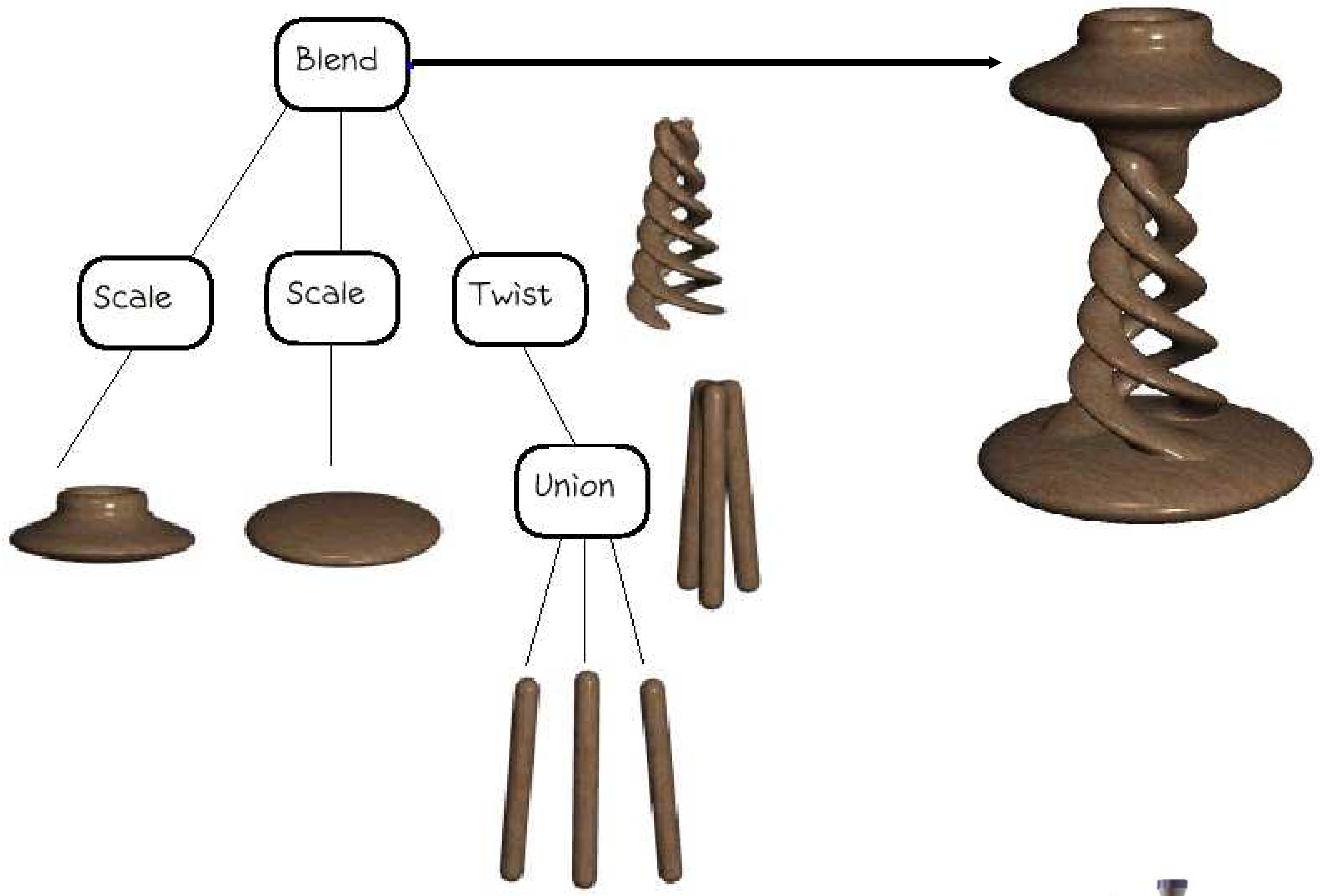
American Type 4-4-0



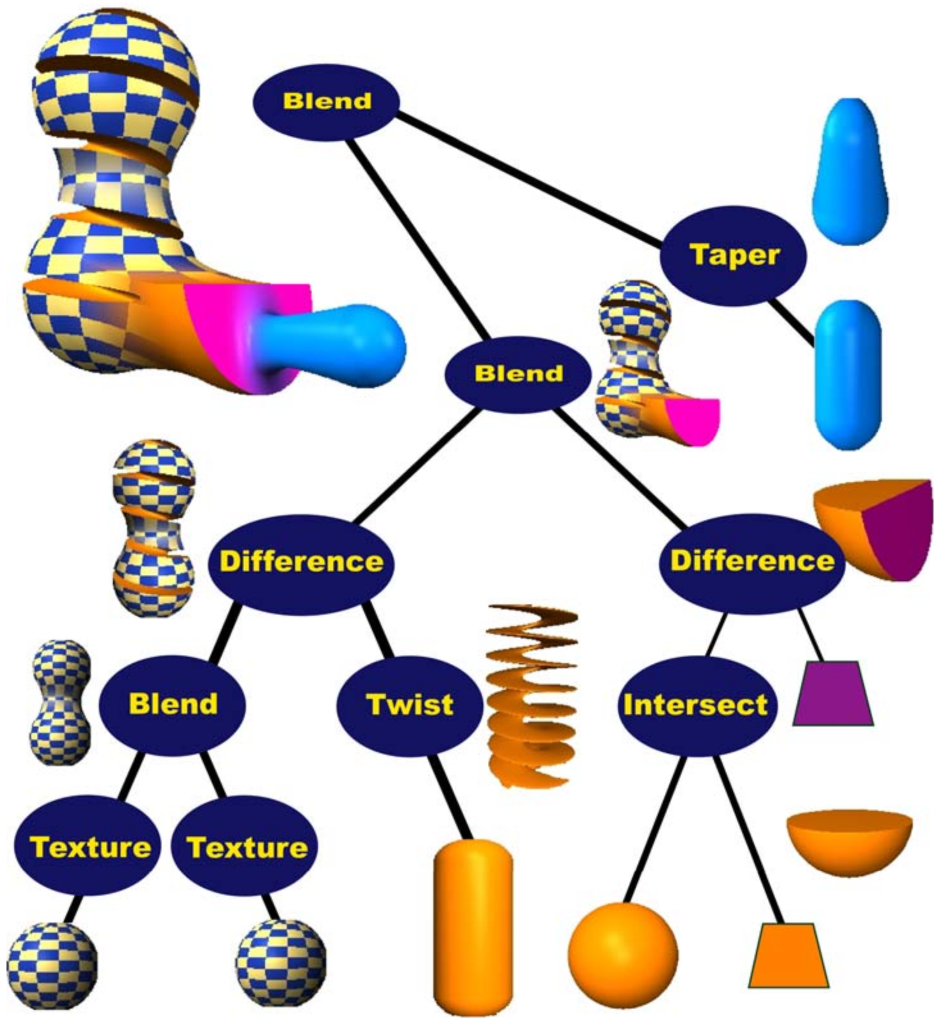
The BlobTree



A BlobTree Example



More BlobTree Nodes



Note the inclusion of the texture node.

Traversing The BlobTree

N - indicates a node in the BlobTree
L (N) - left child R (N) - right child
function F returns the field value for the node N at the point M

```
function F(N , M)
    1. Primitive: F( M)
    2. Warp: F( L (N ) , w( M)) (warp is a unary operator)
    3. Blend: F( L (N ) , M)+ F( R (N ) , M))
    4. Union: max( F( L (N ) , M), F( R (N ) , M))
    5. Intersection: min( F( L (N ) , M), F( R (N ) , M))
    6. Difference: min( F( L (N ) , M), -F( R (N ) , M))
end
```