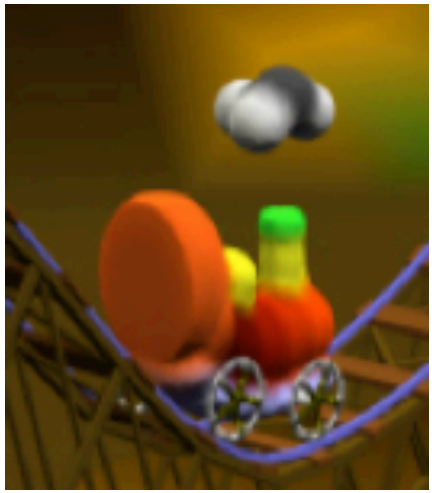
Bremen

CG VR

Advanced Computer Graphics
Modelling beyond Polygons (and Raytracing them ...)



G. Zachmann
University of Bremen, Germany
cgvr.informatik.uni-bremen.de

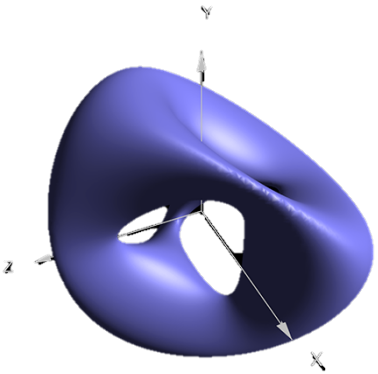
Implicit Surfaces

- An implicit surface is the set

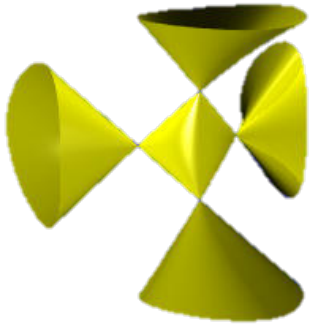
$$\{\mathbf{x} \mid F(\mathbf{x}) = 0\,,\ \mathbf{x} \in \mathbb{R}^3\}$$

  with some function F.

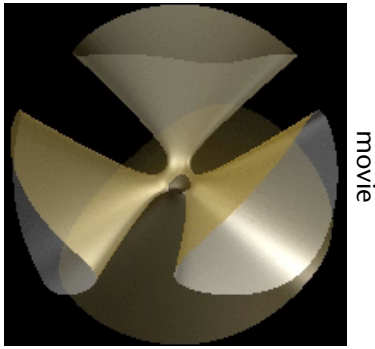
- Example: surface of sphere
- More & nicer examples:



$(x^2+y^2+z^2-ak^2)^2-b((z-k)^2-2x^2)\,((z+k)^2-2y^2)^2$



$8x^2-xy^2+xz^2+y^2+z^2-8$

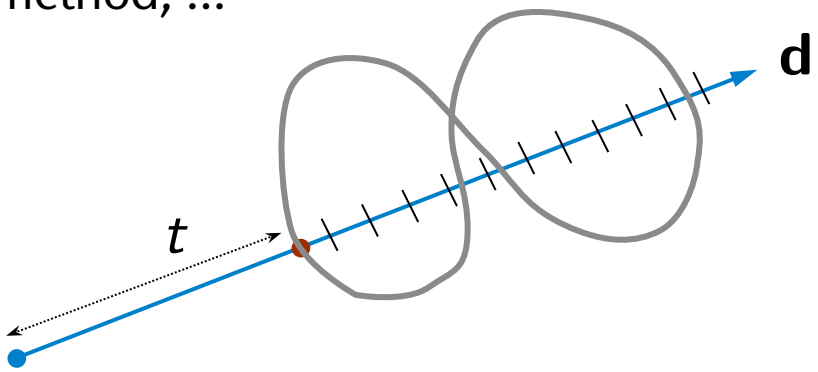


movie

$(x+y+z)^3=x^3+y^3+z^3+1$

Intersection Ray — Implicit Surface

- Ray: $P(t) = O + t \cdot \mathbf{d}$
- Inserting in implicit function $F(\mathbf{x}) = 0$
  yields polynomial
  $$F(\,P(t)\,) = 0$$
  in t of degree n
- Find the roots:
  - If degree < 5: solve for t analytically
  - Else: interval bisection, Newton's method, ...
  - Start values? ...



Root-Finding with Laguerre's Method

- Advantage: one of the very few "*sure-fire*" methods
- Limitations:
  - Works only for polynomials
  - Algorithm needs to perform calculations in complex numbers, even if all roots are real (and thus all coefficients)
  - Very little theory is known about its convergence behavior
    - If the root it converges to is a simple root, then the convergence order is (at least) 3
- Lots of empirical proof that the algorithm (almost) always converges towards a root; and it does so from (almost) any starting value!

Motivation for the Algorithm

- Given: the polynomial

$$P(x) = (x - x_1)(x - x_2)\dots(x - x_n) \qquad (0)$$

where the x_i are the — possibly — roots (which we don't know yet)

- From that, we can derive the following equations:

$$\ln|P(x)| = \ln|x - x_1| + \ln|x - x_2| + \cdots + \ln|x - x_n|$$

$$\frac{d}{dx}\ln|P(x)| = \frac{1}{x - x_1} + \cdots + \frac{1}{x - x_n} = \frac{P'(x)}{P(x)} =: G \qquad (1)$$

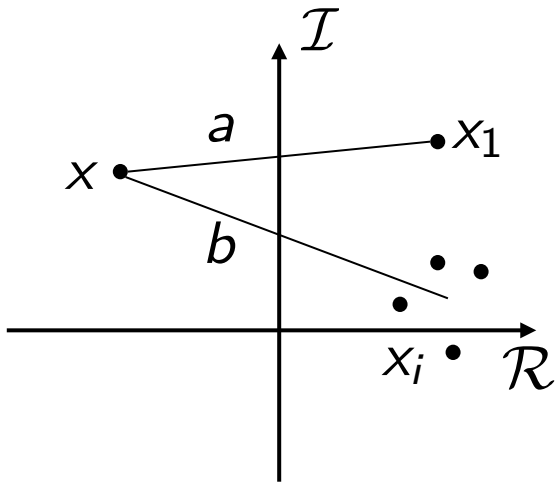
$$\frac{d^2}{dx^2}\ln|P(x)| = -\frac{1}{(x - x_1)^2} - \cdots - \frac{1}{(x - x_n)^2}$$

$$= \frac{P''(x)}{P(x)} - \left(\frac{P'(x)}{P(x)}\right)^2 =: -H \qquad (2)$$

- Let x be our current approximation of a root, w.l.o.g. root x_1

- Make a "drastic" assumption:
  - Denote distance $x - x_1 = a$
  - Assume, distance to all other roots is

$$x - x_i \approx b, \quad i = 2, 3, \ldots, n$$



- Then, we can write (1) & (2) like so:

$$G \approx \frac{1}{a} + \frac{n-1}{b} \qquad (3)$$

$$H \approx \frac{1}{a^2} + \frac{n-1}{b^2} \qquad (4)$$

- Plug (4) into (3) and solve for a :

$$a \approx \frac{n}{G \pm \sqrt{(n-1)\left(nH - G^2\right)}} \tag{5}$$

  - Choose sign in front of sqrt such that $|a|$ becomes minimal
  - Remark: discriminant under sqrt can become negative

    $\longrightarrow a$ can become complex

- New approximation of root x_1 is

$$x_1 = x - a$$

The Algorithm

choose 0-th approximation $x^{(0)}$

repeat

compute $G = \dfrac{P'(x^{(k)})}{P(x^{(k)})}$

$$H = G^2 - \frac{P''(x^{(k)})}{P(x^{(k)})}$$

compute $a = \dfrac{n}{G \pm \sqrt{(n-1)\,(nH - G^2)}}$

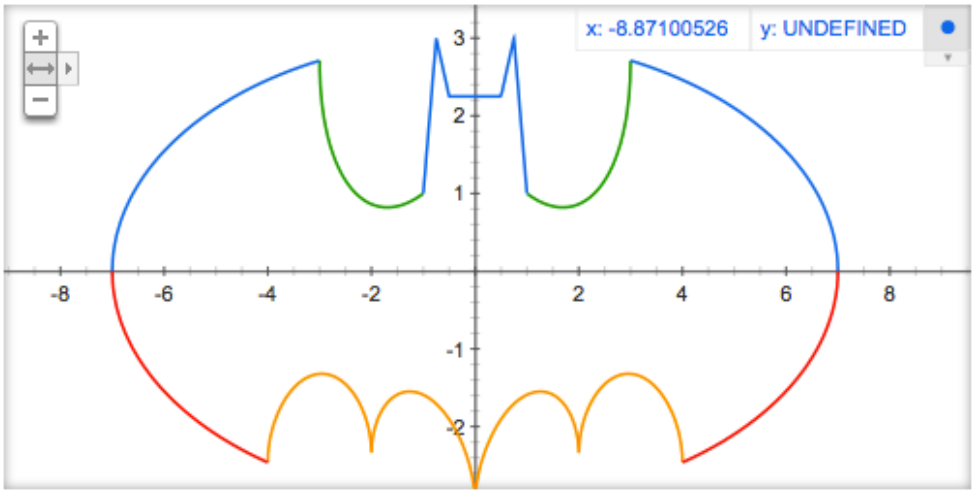
let $x^{(k+1)} = x^{(k)} - a$

until a **"small enough" or** $k \geq$ **max**

- Warning: try to use code from *Numerical Recipes*
- For ray-tracing: have to compute all roots!
  - When first root is found, factor it out of polynomial
  - Find next root of smaller polynomial, repeat Laguerre n times

The "Batman Equation"

- With a few tricks, one can even create complex objects 😀 :

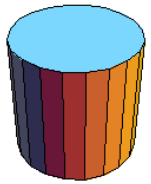


$$\left(\left(\frac{x}{7}\right)^2\sqrt{\frac{||x|-3|}{|x|-3}}+\left(\frac{x}{3}\right)^2\sqrt{\frac{|y+\frac{3\sqrt{33}}{7}|}{y+\frac{3\sqrt{33}}{7}}}-1\right)\cdot$$
$$\left(|\frac{x}{2}|-\left(\frac{3\sqrt{33}-7}{112}\right)x^2-3+\sqrt{1-(||x|-2|-1)^2}-y\right)\cdot$$
$$\left(9\sqrt{\frac{|(|x|-1)(|x|-.75)|}{(1-|x|)(|x|-.75)}}\right)\cdot\left(3|x|+.75\sqrt{\frac{|(|x|-.75)(|x|-.5)|}{(.75-|x|)(|x|-.5)}}-y\right)\cdot\left(2.25\sqrt{\frac{|(|x|-1)(|x|-.75)}{(1-|x|)(|x|-.75}}\right.$$
$$\left.\left(\frac{6\sqrt{10}}{7}+(1.5-.5|x|)\sqrt{\frac{||x|-1|}{|x|-1}}-\frac{6\sqrt{10}}{14}\sqrt{4-(|x|-1)^2}-y\right)=0\right.$$

Quadrics (Sphere, Cylinder, Paraboloid, Hyperboloid)

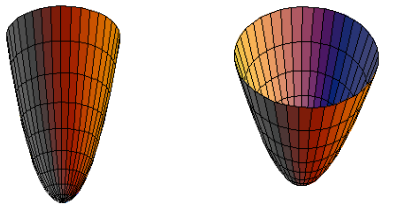
- Infinite cylinder:

$$x^2 + y^2 = 1$$



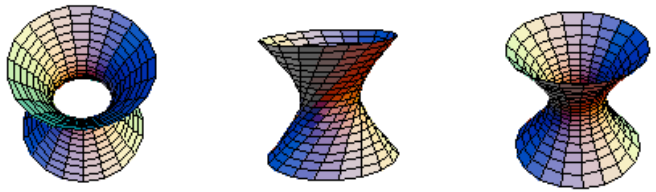
- Parabolloid:

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



- Hyperboloid (*one sheet*):

$$x^2 + y^2 - z^2 = 1$$

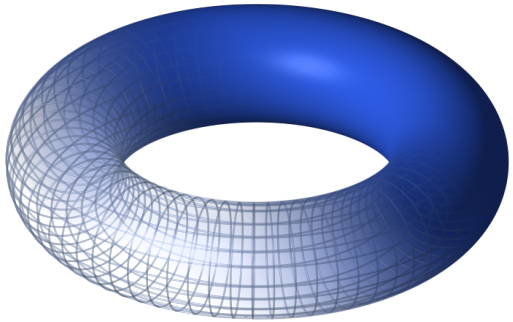


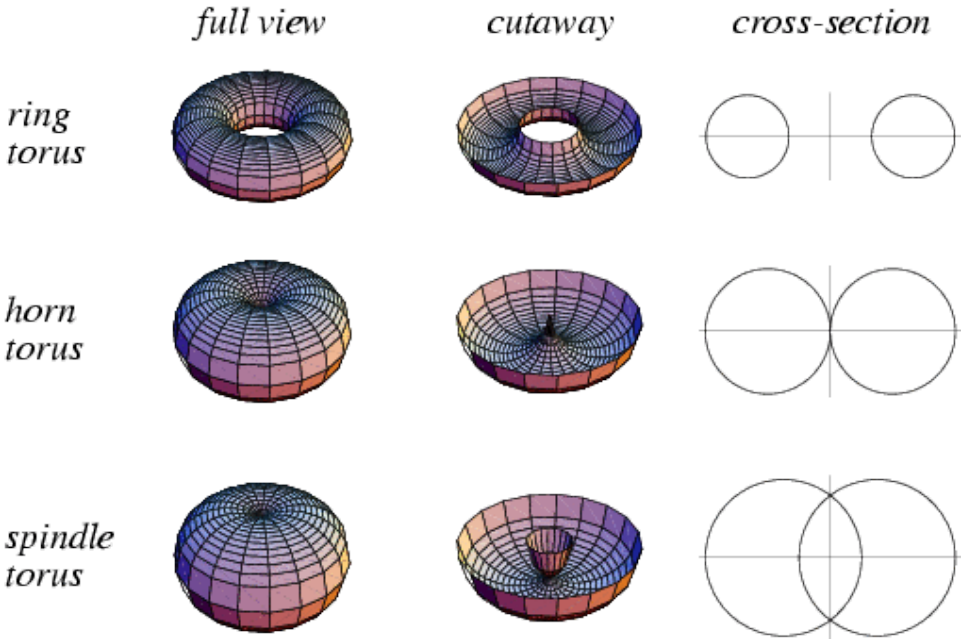
- Can also be written as a quadratic form:

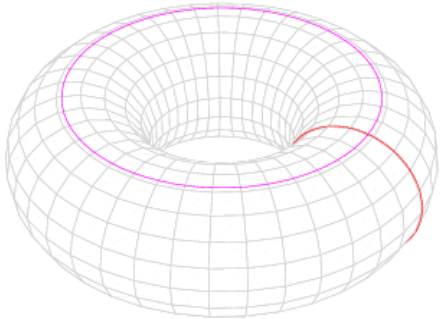
$$\mathbf{x}^T M \mathbf{x} = 0, \quad \mathbf{x} \in \mathbb{R}^3,\ M \in \mathbb{R}^{3\times 3}$$

- Torus (is not really a quadric!):

$$\left(c - \sqrt{x^2 + y^2}\right)^2 + z^2 = a^2$$



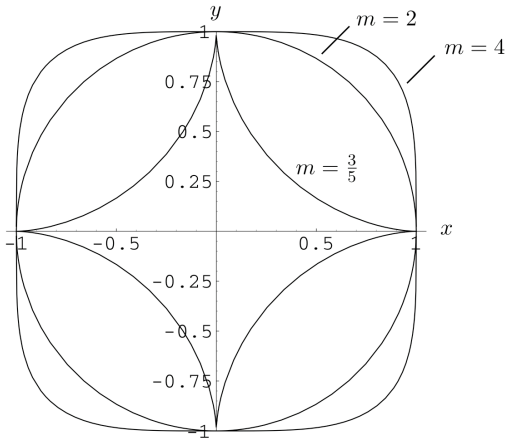




Superquadrics

- Generalization of quadrics
- Super-ellipsoid:

$$\left(\frac{x}{a}\right)^p + \left(\frac{y}{b}\right)^q + \left(\frac{z}{c}\right)^r = 1$$



- Super-hyperboloid:

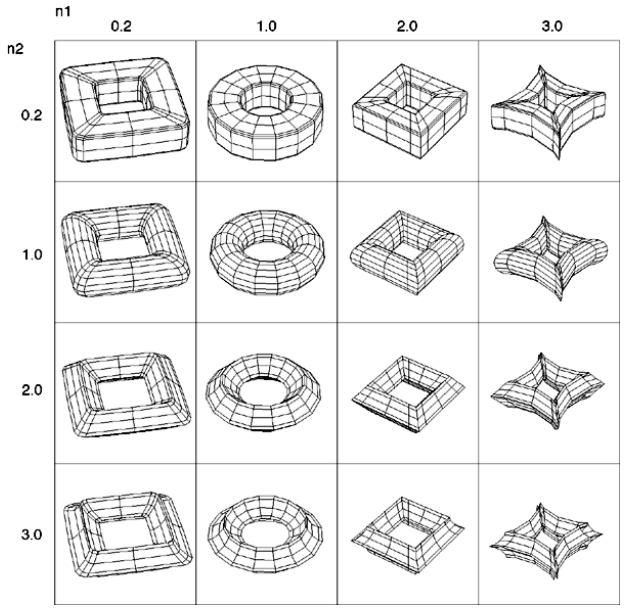
$$\left(\frac{x}{a}\right)^p + \left(\frac{y}{b}\right)^q - \left(\frac{z}{c}\right)^r = 1$$

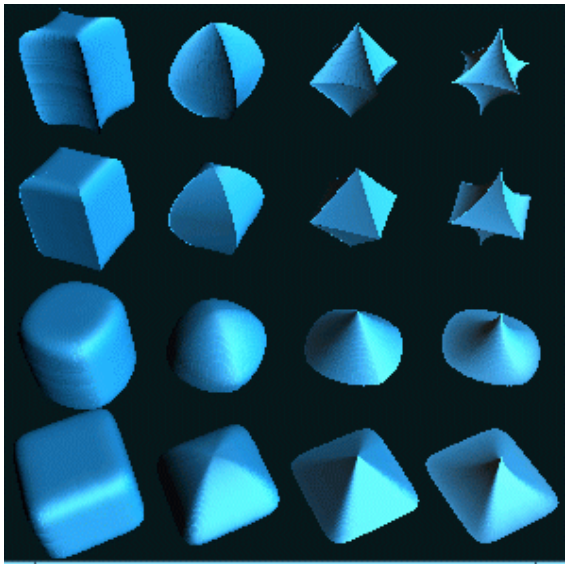
- Super-toroid:

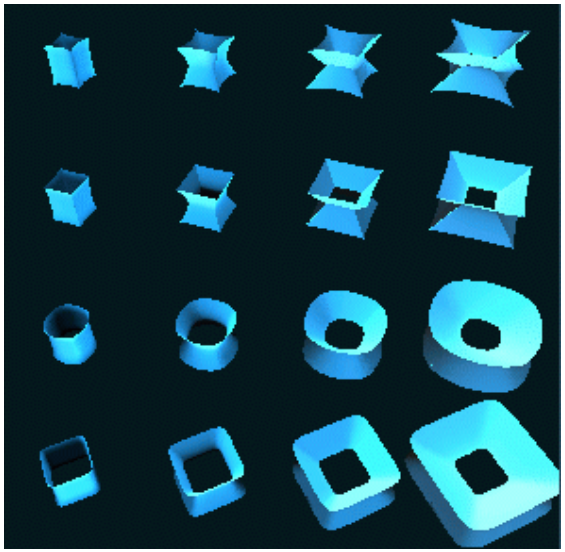
$$\left(d - \left(\left(\frac{x}{a}\right)^m + \left(\frac{y}{b}\right)^n\right)^q\right)^r + \left(\frac{z}{c}\right)^p = e^2$$

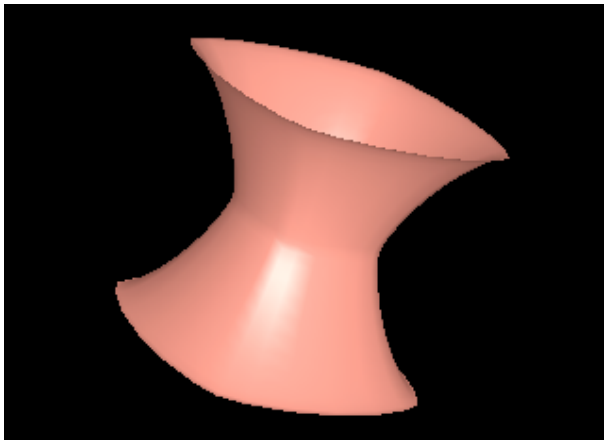
- Warning: in above equtions, we always mean $|x|^p$!

Examples of Super-Quadrics









XScreenSaver demo "SuperQuadrics"
(www.jwz.org/xscreensaver)

Ratioquadrics

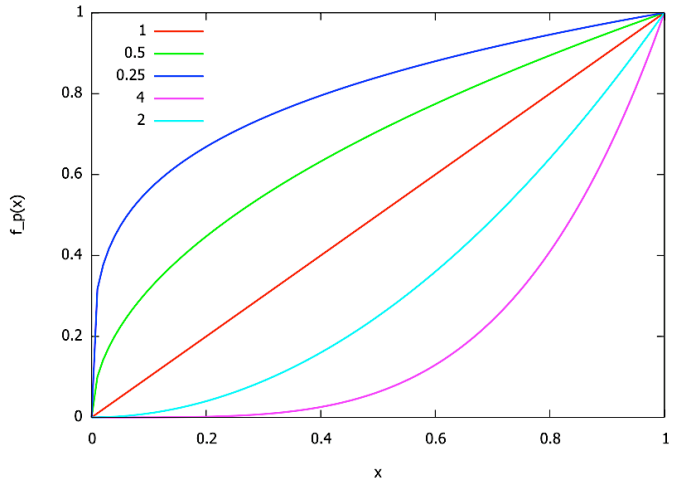
[Blanc & Schlick, 1996]

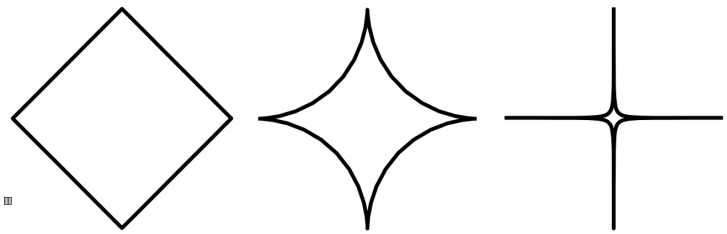
- Variant of Superquadrics with somewhat better properties
- Idea of superauadrics can be rewritten like so:

$$F(x,y,z) = f_p\left(\frac{x}{a}\right) + f_q\left(\frac{y}{b}\right) + f_r\left(\frac{z}{c}\right) - 1$$

$$f_p(x) = |x|^p$$

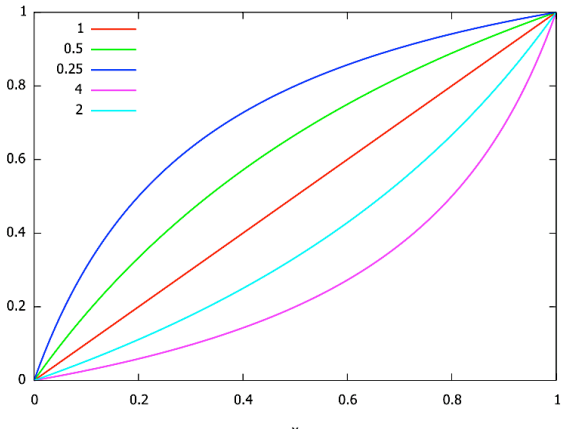
- Problem:
  - $f_p(x)$ is not differentiable at x=0 for $p \leq 1$
  - Therefore, we get cusps, which might be unwanted
  - Besides, $f_p(x)$ is fairly expensive to evaluate





- Simple idea: use different power functions
- A new pseudo-power function [Blanc & Schlick]:

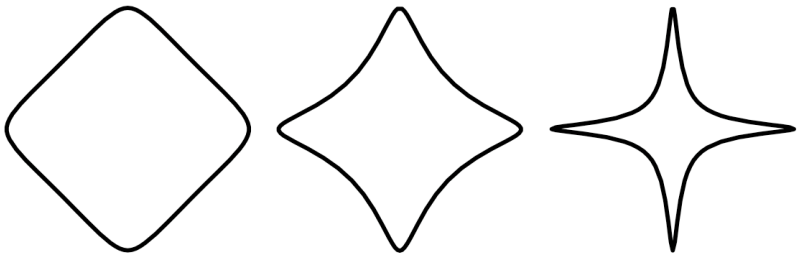
$$g_p(x) = \frac{x}{p + (1-p)x}$$



- With that, the ratioquadric for a "ratio-ellipsoid" is

$$F(x,y,z) = g_p\left(\tfrac{x}{a}\right) + g_q\left(\tfrac{y}{b}\right) + g_r\left(\tfrac{z}{c}\right) - 1$$

- Result:



Metaballs

- Inspired by molecules
- Idea: consider the surface of a sphere as the set of points that have the same "potential", where the maximum is reached at the center of the sphere → Isosurface
- A potential field is described by a potential field function, e.g.

$$p(r) = \frac{1}{r^2}$$

where

$$r = r_1(\mathbf{x}) = \|\mathbf{x} - \mathbf{x}_1\|$$

- The sphere's surface is thus

$$K = \{\mathbf{x} \mid p(\mathbf{x}) = \tau\}$$

  - τ is called threshod or Isovalue

- More complex objects can be created by blending (superposition) of several potential fields
  - Simplest blending is (weighted) addition of the potential fields:

$$P(\mathbf{x}) = \sum_{i=1}^{n} a_i \frac{1}{r_i^2(\mathbf{x})}, \qquad r_i(\mathbf{x}) = \|\mathbf{x} - \mathbf{x}_i\|$$

  - The set of points x_i is called the *skeleton*,
    P is the total potential, the a_i determine the influence (= "field's force")
  - Negative influence can "carve out" material (e.g., for making holes)
- Overall, metaballs are comprised of

  distance function + potential function

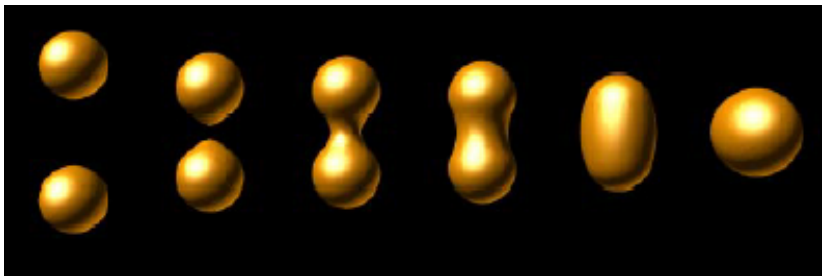
- Many names for this kind of modeling methodology: "metaballs", "soft objects", "blobs", "blobby modeling", "implicit modeling" ...

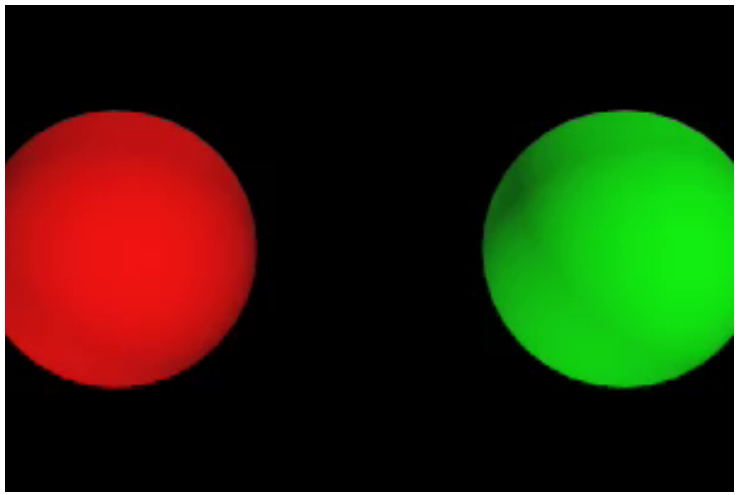
- In general, a metaballs object is defined as the isosurface

$$\mathcal{F} = \{\ P(\mathbf{x}) = \tau \mid \mathbf{x} \in \mathbb{R}^3\ ,\ P(\mathbf{x}) = \sum a_i p(d_i(\mathbf{x}))\ \}$$

with p = potential function,
d_i = distance function to i-th skeletal point

- Examples for 2 skeleton points:



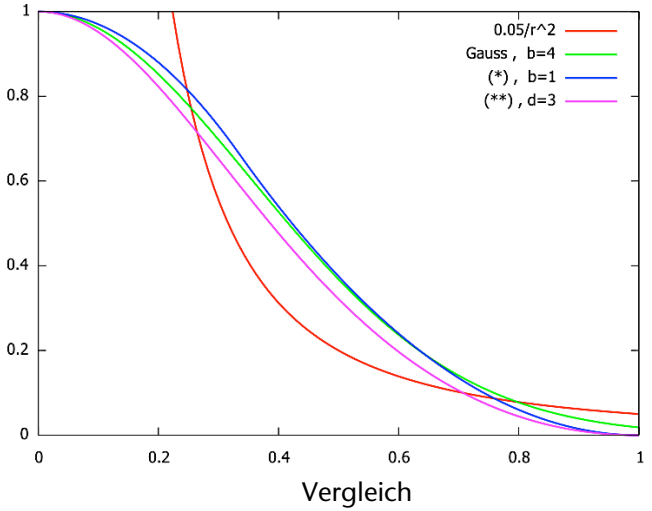


- Other potential functions:

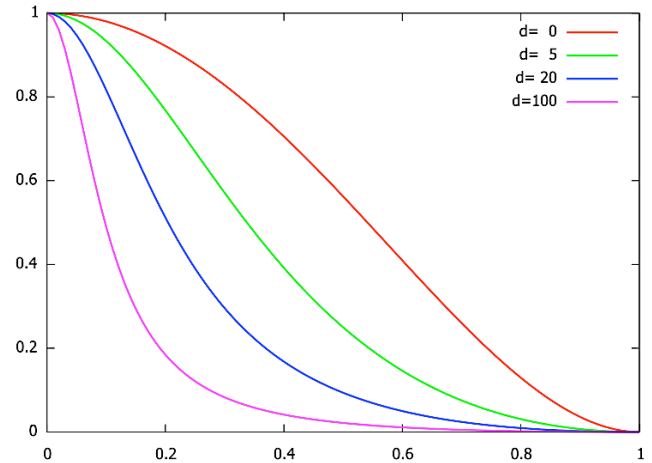
$$p_i(r) = e^{-br^2}$$

$$p(r) = \begin{cases} 1 - 3\frac{r^2}{b^2} & , \; r \leq \frac{1}{3}b \\ \frac{3}{2}(1 - \frac{r}{b})^2 & , \; \frac{1}{3}b \leq r \leq b \\ 0 & , \; r > b \end{cases} \quad (*)$$

$$p(r) = \begin{cases} \frac{r^4 - 2r^2 + 1}{1 + dr^2} & , \; r \leq 1 \\ 0 & , \; r > 1 \end{cases} \quad (**)$$

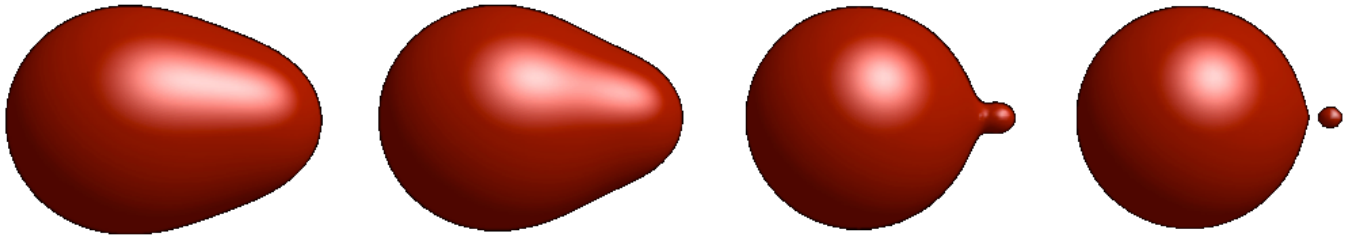


Vergleich



(**) mit verschiedenen Parametern

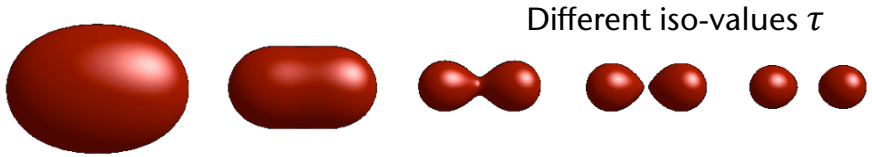
- Effect of the variation of the parameter *d* :

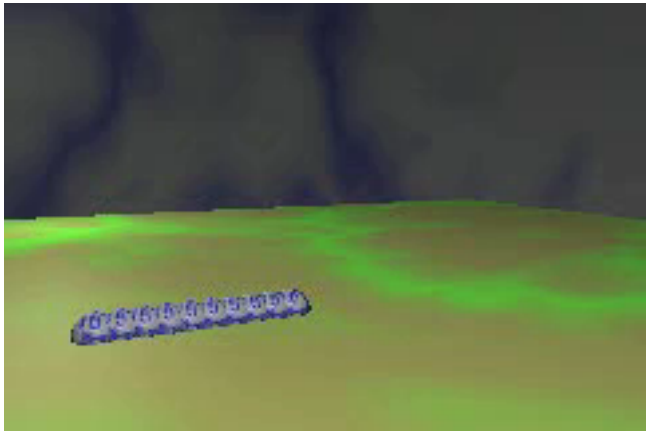


Potential fct is (**), ***d*** is fixed for the left skeleton point, ***d*** = 10 ... 2000 for the right skeleton point

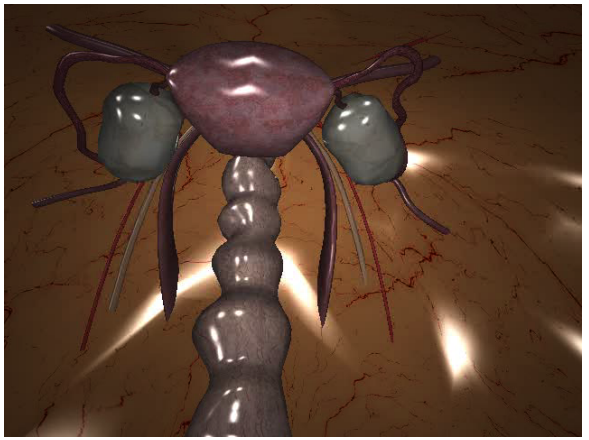
Deformable Models

- With implicit modeling (metaballs), it is easy to create and animate deformable "blob-like" objects:
  - Animate (move) the skeleton points
  - Modify parameters a_i, d, ...
  - Modify the iso-value τ

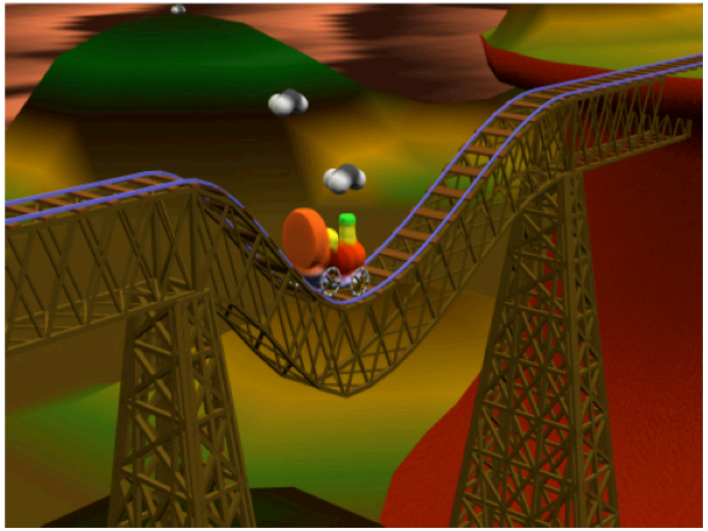




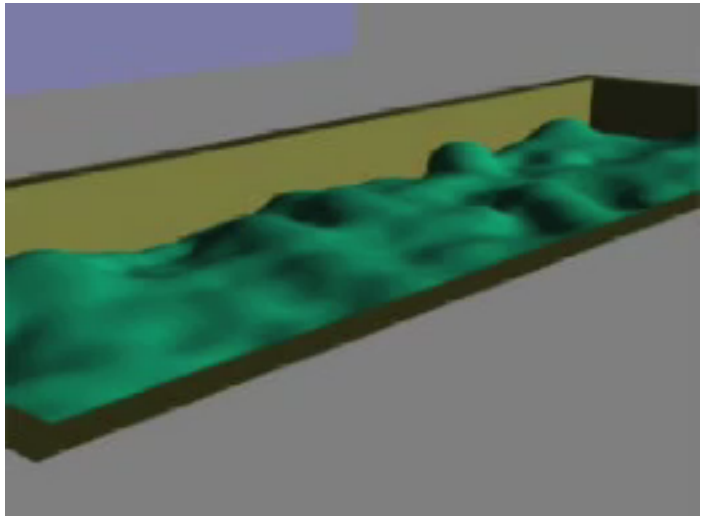
Brian Wyvill
http://pages.cpsc.ucalgary.ca/~blob/animations.html



Frédéric Triquet
http://www2.lifl.fr/~triquet/implicit/video/



"The Great Train Rubbery" — Siggraph 1986



"Soft"

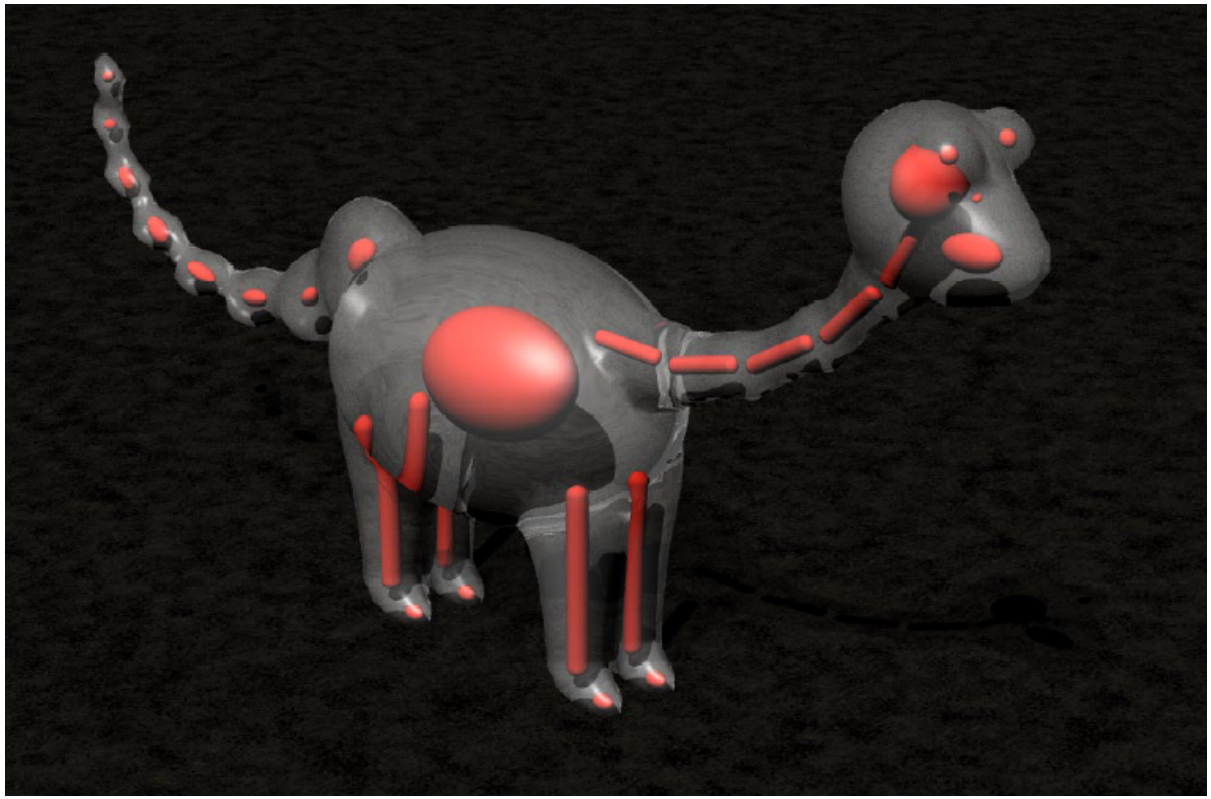
"The Wyvill Brothers"

Geoff

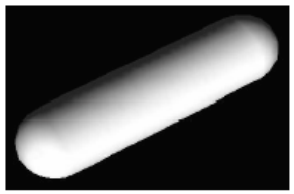
Brian

Generalizatoin / Variants

- Points are the simplest kind of primitive for metaballs skeletons; analogously, we can use lines, polygons, ellipsoids, etc.:



Examples of other primitives:

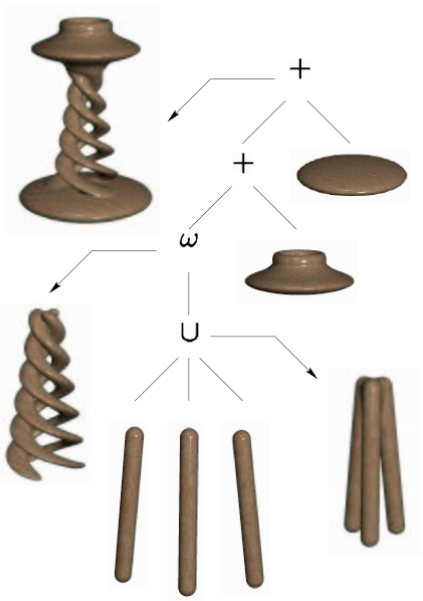


- Other blending functions:

$$P_{\cup}(\mathbf{x}) = \max\{p_1(\mathbf{x}), p_2(\mathbf{x})\}$$

$$P_{\cap}(\mathbf{x}) = \min\{p_1(\mathbf{x}), p_2(\mathbf{x})\}$$

- A tree of "blending" operations (similar to CSG) — the "BlobTree":



Remarks on Implicit Modeling

- One can achieve some nice effects very easily
- The technique did not get traction in the tool set of animation industries and CAD, because there is too much "black magic" involved in achieving a particular effect [says Geoff Wyvill, too]
- For special kinds of deformable objects, it can be very useful, e.g., for fluids

The Normal on Implicit Surfaces

- The normal in a point $\boldsymbol{x}$ on the implicit surface $f(\boldsymbol{x})$:

$$\mathbf{n}(\mathbf{x}) = \nabla f(\mathbf{x}) = \begin{pmatrix} \frac{\partial f}{\partial x}(\mathbf{x}) \\ \frac{\partial f}{\partial y}(\mathbf{x}) \\ \frac{\partial f}{\partial z}(\mathbf{x}) \end{pmatrix}$$

$$\approx \begin{pmatrix} f(x+\varepsilon, y, z) - f(\mathbf{x}) \\ f(x, y+\varepsilon, z) - f(\mathbf{x}) \\ f(x, y, z+\varepsilon) - f(\mathbf{x}) \end{pmatrix}$$

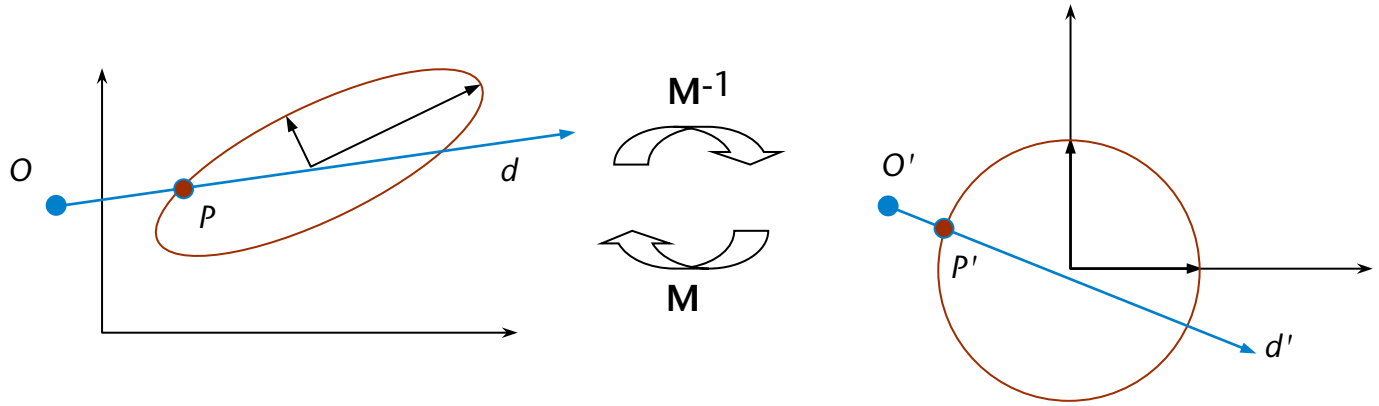
$$\approx \begin{pmatrix} f(x+\varepsilon, y, z) - f(x-\varepsilon, y, z) \\ f(x, y+\varepsilon, z) - f(x, y-\varepsilon, z) \\ f(x, y, z+\varepsilon) - f(x, y, z-\varepsilon) \end{pmatrix}$$

Instancing / Ray Transformation

- "Complex" (transformed) shapes can often be reduced to simpler & canonical shapes
- Idea:
  - Transform ray by inverse of shape's transformation
  - Compute intersection of ray and canonical shape
  - Transform intersection point (and normal) back

Example: Ray – Ellipsoid Intersection

- The back-and-forth transformations:



- The algorithm:

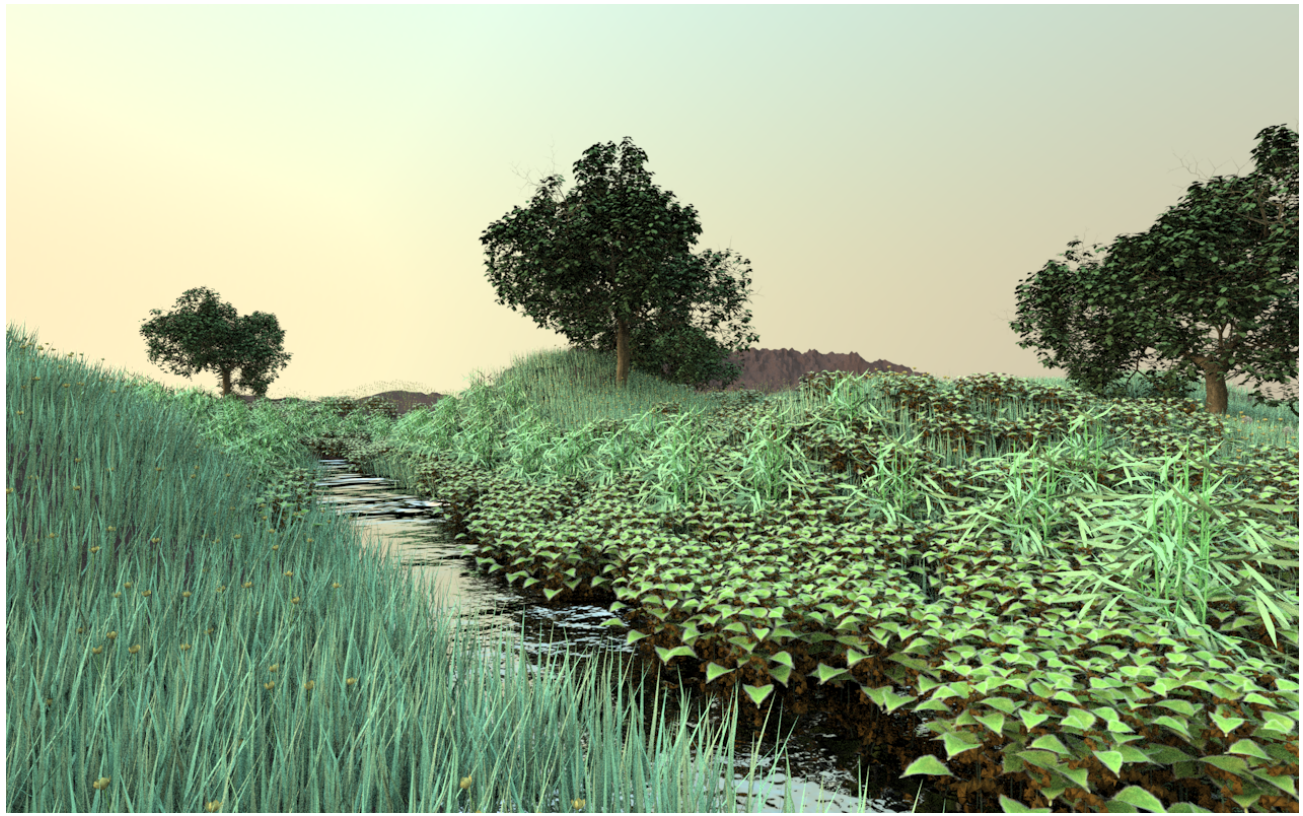
berechne $P'(t) = \mathbf{M}^{-1}O + t\mathbf{M}^{-1}d$

schneide $P'(t)$ mit Einheitskugel $\rightarrow P'$, $\mathbf{n}'$, t'

$P := \mathbf{M} \cdot P'$; $\mathbf{n} := (\mathbf{M}^{-1})^{\mathrm{T}} \cdot \mathbf{n}'$; $t :=?$

Another Reason for *Instancing*

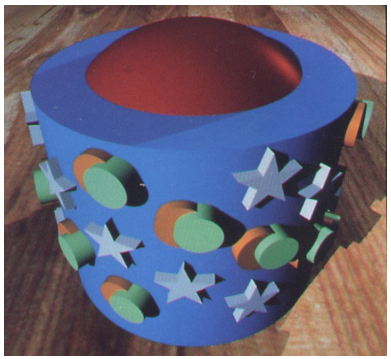
- Memory efficiency: only using instancing can you fit such huge scenes into main memory

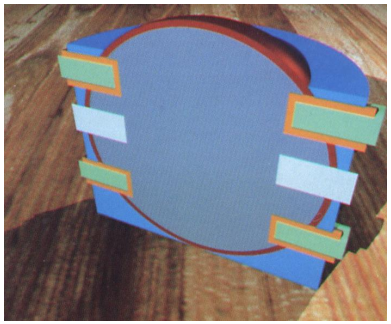


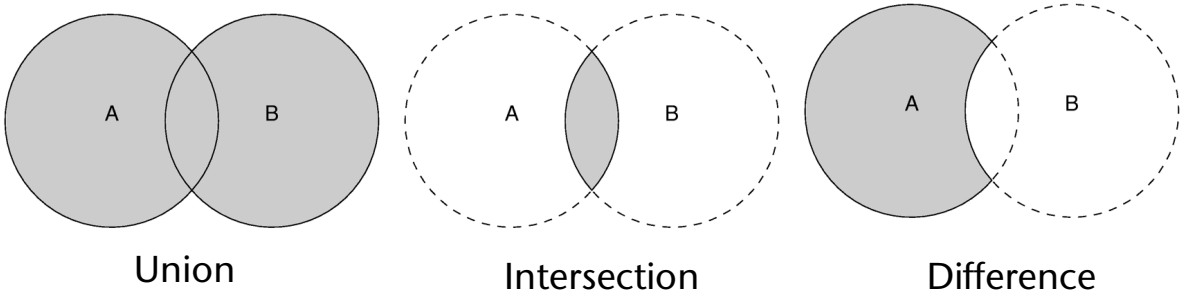
With instancing: 61 unique plant models, 1.1M unique triangles, 300MBytes
With explicit representation: 4000 instanced plants in the scene, 19.5M triangles

Constructive Solid Geometry (CSG)

- Very easy to render by ray-tracing
- Central idea: construct new object by set operations performed on simple, volumetric objects → *constructive solid geometry* (CSG)
- Simple primitives: all objects that can be described and ray-traced easily (e.g., sphere, box, ...)
- Set operations : union, intersection, difference



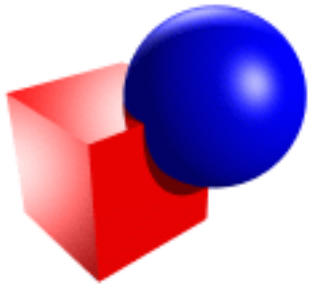




Union Intersection Difference

Examples

- The three set operations applied to sphere & box in 3D:



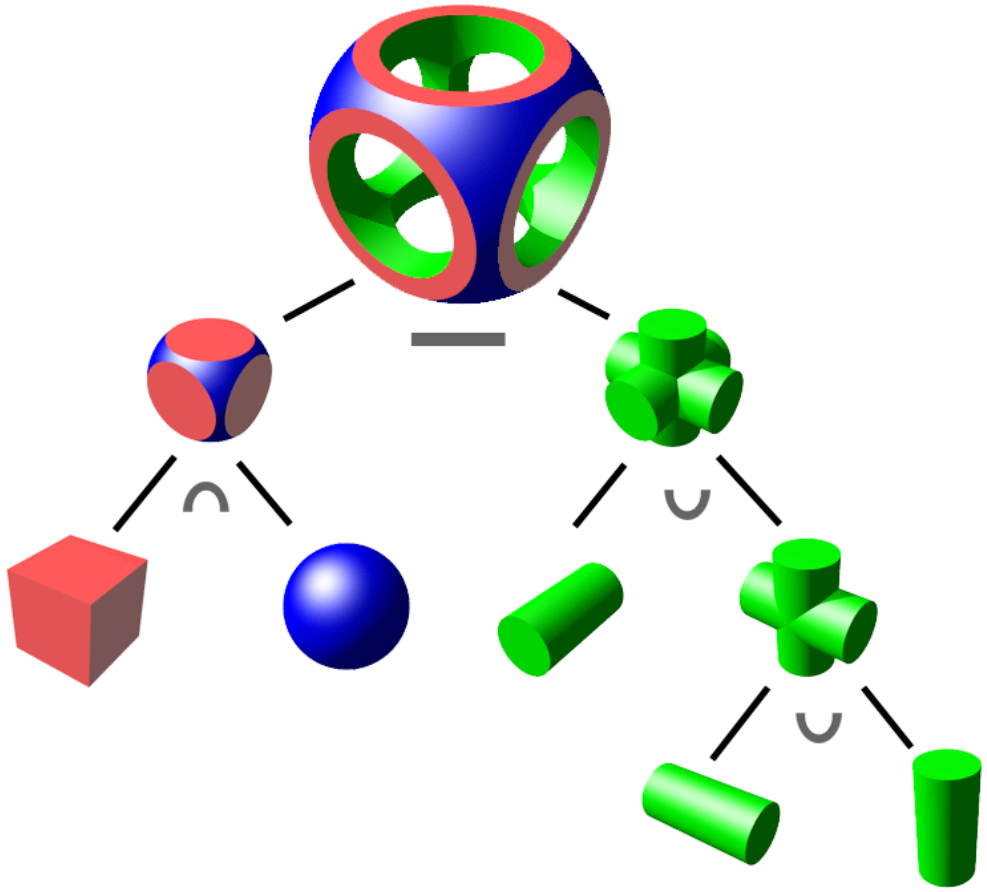
Union

Difference

Intersection

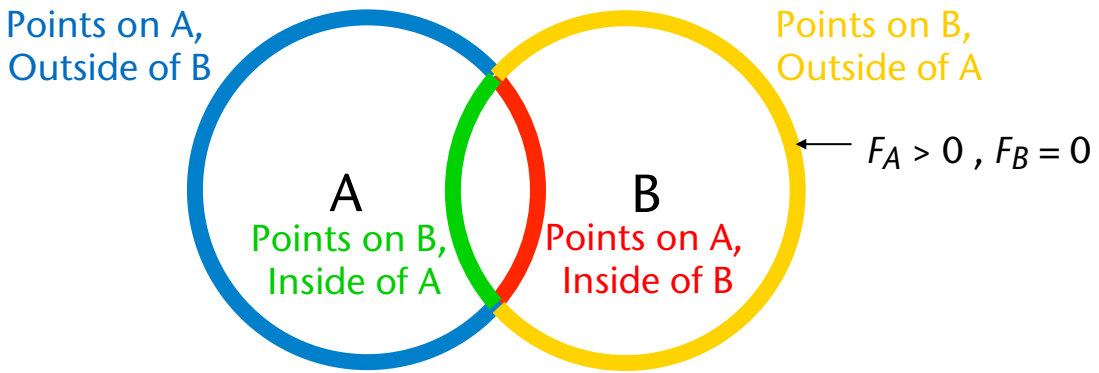
The CSG Tree by Way of an Example

- Recursive application of the set operations
  ⟶ CSG tree = one CSG object
- Evaluation of CSG trees works similar to evaluation of arithmetic expression trees

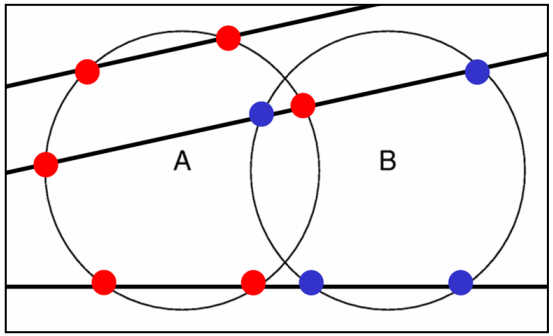


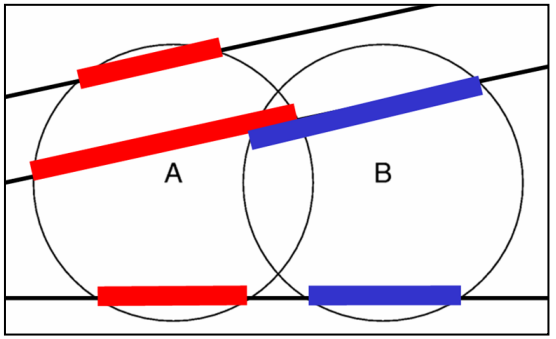
Raytracing of CSG Objects

- Use implicit or explicit representation of the primitives:

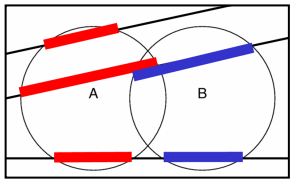


- Determine **all** intersection points of a ray with the 2 primitives
- If primitives are convex → 1 interval per primitive on the ray

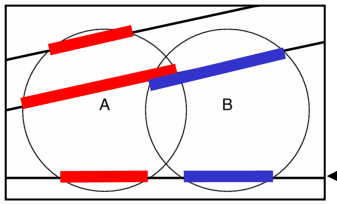




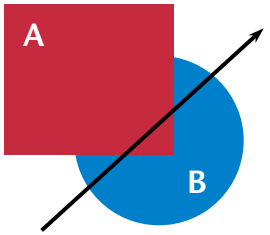
- Apple the CSG operation on the intervals
- Propagate intervals up through the CSG tree



- If interval is empty when reaching root → no intersection
- Else: choose closest interval and point closest to viewpoint
- Warning:
  - During CSG operations on intervals, the resulting interval can be non-contiguous (i.e., several intervals need to be maintained during tree traversal)!



With A∪B, we get "one" non-contiguous interval in case of this ray!



Dito in this case with B - A !

  - Also, pay attention to numerical robustness (e.g., kill too small intervals)

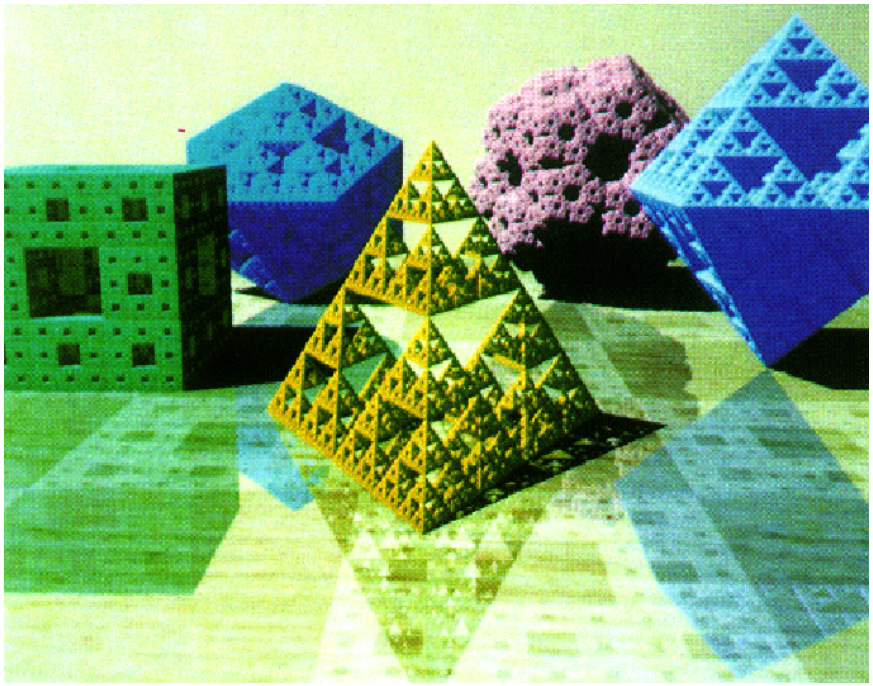
"Villarceau Circles" by Tor Olav Kristensen (2004)

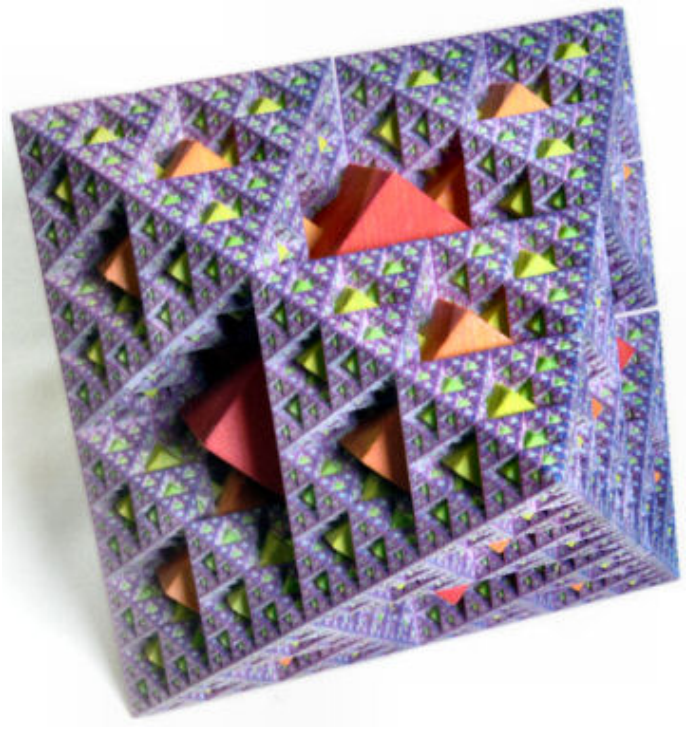
For every point on a torus one can draw four different circles through it that all lie on the surface of the torus. Two of these four circles are called Villarceau circles. The four narrow pairs of bands in this image follow such Villarceau circles.
All the shapes in this image are made with Constructive Solid Geometry operations with tori only (except for the ground plane of course).

Fractals

- Fractals can be ray-traced trivially
  - At least the simple fractals as shown below
- Just recurse "on demand" up to some predefined depth

→ Prozedural objects





Eine frühe Anwendung

- Fraktale eignen sich hervorragend, um Terrain prozedural zu modellieren:

Loren C. Carpenter: Vol Libre. 1980