# Advanced Modeling

Andreas H. König, Peter Rautek

March 9, 2011

## 1 Introduction

Nowadays (triangle) meshes are by far the most commonly used representation for modeling geometric objects in computer graphics. However, in many situations meshes are not well suited. For instance the creation of convincing models for fluids, smoke, fire, organic structures, repetitive structures, etc., is very tedious and sometimes impossible using meshes. There are several reasons why meshes are not well suited for these tasks:

**Fuzzy object boundaries:** Many real world phenomena do not have clearly defined object boundaries. Therefore it is impossible to model them with a surface representation approach (like meshes). The modeling of these phenomena requires volumetric representation approaches.

**Large deformations of the surface:** Modeling of objects that deform over time (such as wave fronts in a wave propagation simulation) are very hard to model using meshes. These objects are often modeled and rendered more conveniently using particle or grid based methods.

**Changing topology:** Meshes have only limited use for models that change their topology over time (i.e., holes are introduced or removed, they split and merge, etc.). Meshes are mainly not very well suited for this situations because they are tedious to model and many algorithms on meshes were developed assuming that the topology is constant. In some situations (often in combination with physical simulations) topology changes are better handled with alternative modeling techniques.

**High geometric complexity:** Certain phenomena exhibit huge geometric complexity. Although it is possible to model them with meshes, it is often more convenient to model them with an alternative method. In many

(a) Water (b) Sea (c) Smoke

Figure 1: Examples of complex phenomena that: exhibit large surface deformations (a), quickly changing topology (b) and fuzzy object boundaries (c). Images taken from [4, 5][1].

cases the resulting models can afterwards be converted to triangle meshes for rendering.

Figure 1 shows examples of complex phenomena that are typically not modeled with meshes.

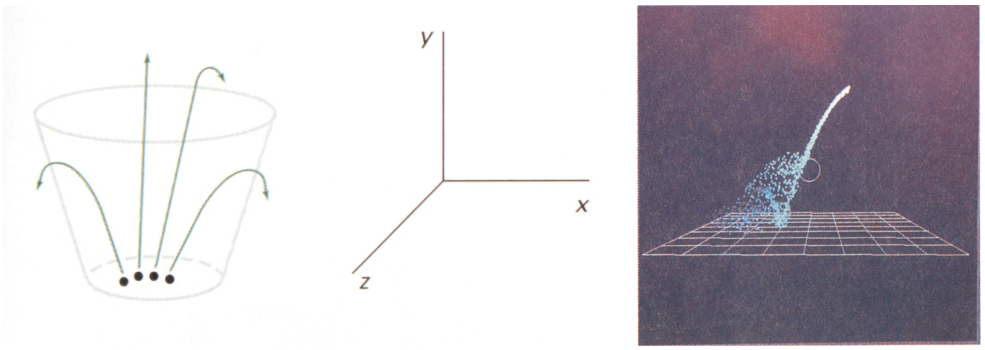
To overcome the limitations of meshes a wide variety of advanced modeling techniques were invented. In this Chapter we will review the basics of frequently used techniques. In Section 2 an introduction to particle systems is given. Particles are very useful for modelng fuzzy objects or phenomena such as smoke, and fire. In Section 3 we review implicit modeling techniques that are in computer graphics mainly used to render fluids. Apart from the applications in computer graphics the introduced methods do have a wide area of applications including image segmentation, fluid simulation, etc. In Section 4 we will review procedural methods that are mainly applicable for objects with very high geometric complexity. These objects include hair and fur, as well as organic structures like roots of plants, trees, grass, etc. Finally, Section 5 deals with structure deforming transformations, that are themselves not modeling techniques. However, they are applicable to many representations of geometric objects, yielding results that are sometimes hard to achieve with other modeling techniques.

[1]There is lots of additional information on Ronald Fedkiw's homepage at Stanford: `http://physbam.stanford.edu/~fedkiw/`

# 2 Particle Systems

Particle systems are a method for modeling natural objects, or other irregularly shaped objects, that exhibit fluid-like properties. This method is particularly good for describing objects that change over time by flowing, billowing, spattering, or expanding. Objects with these characteristics include clouds, smoke, fire, fireworks, waterfalls, water spray, and clumps of grass. For example, particle systems were used to model the planet explosion and expanding wall of fire due to the *genesis bomb* in the motion picture *Star Trek II: The Wrath of Khan*.

Random processes are used to generate objects within a defined region of space and to vary their parameters over time. After a random time, each object is deleted. During the lifetime of a particle, its path and surface characteristics may be color-coded and displayed. Particle shapes can be small spheres, ellipsoids, boxes, or other shapes. The size and shape of particles may vary randomly over time. Also, other properties such as particle transparency, color, and movement can vary randomly. In some applications, particle motion may be controlled by specified forces, such as a gravity field.




(a) Simulated growing grass (b) Simulated waterfall

Figure 2: Particle systems used for simulation of complex phenomena like grass or waterfalls. Images taken from [3]

As each particle moves, its path is plotted and displayed in a particular color. For example, a fireworks pattern can be displayed by randomly generating particles within a spherical region of space and allowing them to move radially, outward. The particle paths can be color-coded from red to yellow, for instance, to simulate the temperature of the exploding par-

Figure 3: An effect known as particle dispersion or disintegration[2].

ticles. Similarly, realistic displays of grass clumps have been modeled with trajectory-particles that are shot up from the ground and fall back to earth under gravity. In Figure 2(a) the particle paths originate within a tapered cylinder, and might be color-coded from green to yellow.

Figure 2(b) illustrates a particle-system simulation of a waterfall. The water particles fall from a fixed elevation, are deflected by an obstacle, and then lash up from the ground. Different colors are used to distinguish the particle paths at each stage.

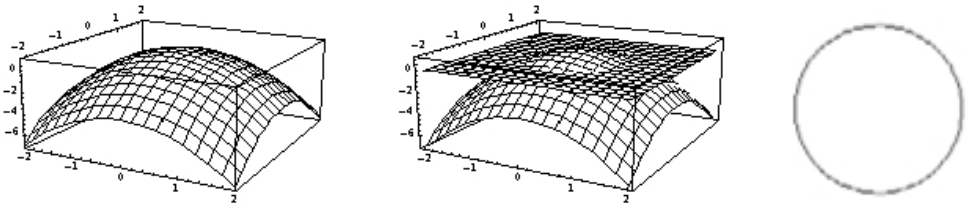
Special effects for motion pictures also frequently employ particle systems. A frame of an animation, simulating the disintegration of an object is shown in Figure 3 . The object disintegrates into the particle distribution from left to right.

# 3 Implicit Modeling

Implicit equations (in contrast to eplicit equations) do not express one variable in terms of another one. Equation 1 is an example of an implicit equation:

$$f(x, y) = -(x^2 + y^2) = T \tag{1}$$

[2]A tutorial about particle dispersion can be found at: `http://forcg.com/tutorials/particle-system/particle-dispersion-effect-using-3ds-max/`



(a) 2d equation (b) intersection with plane $T$ (c) result

Figure 4: (a) The equation $-(x^2 + y^2)$ is plotted, (b) The intersection with plane T is shown, (c) The resulting 2d model.

where $x$ and $y$ are variables, and $T$ is a constant. In contrast Equation 2 is an example of explicit equations (i.e., a function) that does express one variable (in this case $y$) with the help of another one (in this case $x$):

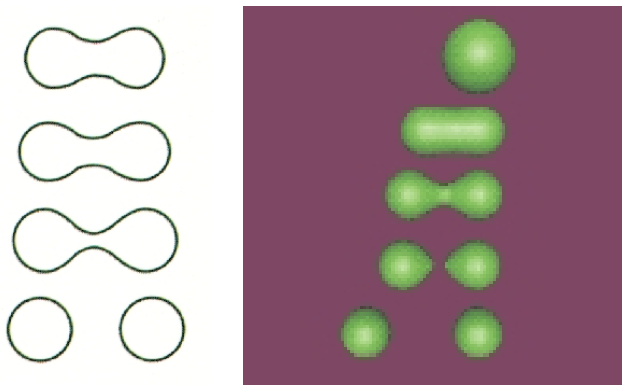
$$f(x) = y = kx + d \tag{2}$$

where $k$ and $d$ are constants.

Implicit modeling refers to a technique that makes use of implicit equations. The surface of an implicit model is defined as the set of points, that fullfill the given equation. Figure 4 shows a plot of Equation 1. $T$ is a threshold that can be interpreted as a plane intersecting the surface. $T$ can be changed to alter the radius of the resulting iso-circle. Implicit models are also called level curves, iso contours, or contour lines in 2D; level surfaces or iso surfaces in 3D and level hypersurfaces in nD. Specific implicit equations and the corresponding models have specific names such as blobby objects, soft objects, metaballs, superquadrics, etc.

## 3.1 Blobby Objects

Some objects do not maintain a fixed shape, but change their surface characteristics in certain motions or when in proximity to other objects. Examples in this class of objects include molecular structures, water droplets and other liquid effects, melting objects, and muscle shapes in the human body. These objects exhibit a certain *blobbiness* and are often simply referred to as blobby objects, since their shapes show a certain degree of liquidity. A molecular shape, for example, can be described as spherical in isolation but the shape changes when one molecule approaches another one. Distortion of the shape



(a) 2d model (b) 3d model

Figure 5: Molecular bonding. As two molecules move away from each other, the surface shapes stretch, snap, and finally contract into circles 5(a) or spheres 5(b).

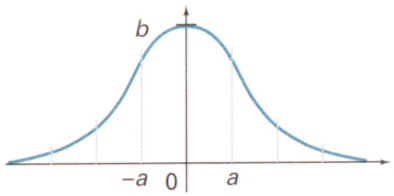



Figure 6: Gaussian bump centered at position 0, with height b and standard deviation a.

of the electron density cloud is due to the *bonding* that occurs between the two molecules.

Figure 5 illustrates the stretching, snapping, and contracting effects on molecular shapes when two molecules move apart. These characteristics cannot be adequately described with spheres or elliptical shapes. More information on modeling with Blobby objects can be found in the work of Wyvill et al. [6].

Several models have been developed for representing blobby objects as distribution functions over a region of space. One possibility is a combination of Gaussian density functions:

$$f(x, y, z) = be^{-ar^2} \tag{3}$$

is the Gaussian density function (see Figure 6 for a plot of the 1D version of the Gaussion density function). $r$ is the distance to the center point $(x_0, y_0, z_0)$ of the Gaussian function:

$$r = \sqrt{(x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2} \tag{4}$$

If $k$ *Gaussian bumps* are placed at the $k$ control points $(x_k, y_k, z_k)$ and are combined, the surface function of the combined model is defined as

$$f(x, y, z) = \sum_k b_k e^{-a_k r_k^2} - T = 0 \tag{5}$$

where

$$r_k^2 = (x - x_k)^2 + (y - y_k)^2 + (z - z_k)^2 \tag{6}$$

Parameter $T$ is a threshold, and parameters $a_k$ and $b_k$ are used to adjust the shape of the individual blobs.

Negative values for parameters $b_k$ can be used to produce dents instead of bumps. Figure 5 shows examples of 2D and 3D blobs that move closer to each other and finally merge.

Other methods for generating blobby objects use density functions that fall off to 0 in a finite interval, rather than exponentially. The *metaball* model describes composite objects as combinations of quadratic density functions of the form

$$f(r) = \begin{cases} a(1 - \frac{3r^2}{b^2}) & \text{if } 0 < r \leq b/3 \\ \frac{3a}{2}(1 - \frac{r}{b})^2 & \text{if } b/3 < r \leq b \\ 0 & \text{if } b < r \end{cases} \tag{7}$$

And the *soft-object* model uses the function

$$f(r) = \begin{cases} a(1 - \frac{4r^6}{9b^6} + \frac{17r^4}{9b^4} - \frac{22r^2}{9b^2}) & \text{if } 0 < r \leq b \\ 0 & \text{if } b < r \end{cases} \tag{8}$$

Figure 7 shows a plot of the above mentioned density functions.

## 3.2 Superquadrics

This class of objects is a generalization of quadric representations. Superquadrics are formed by incorporating additional parameters into the quadric

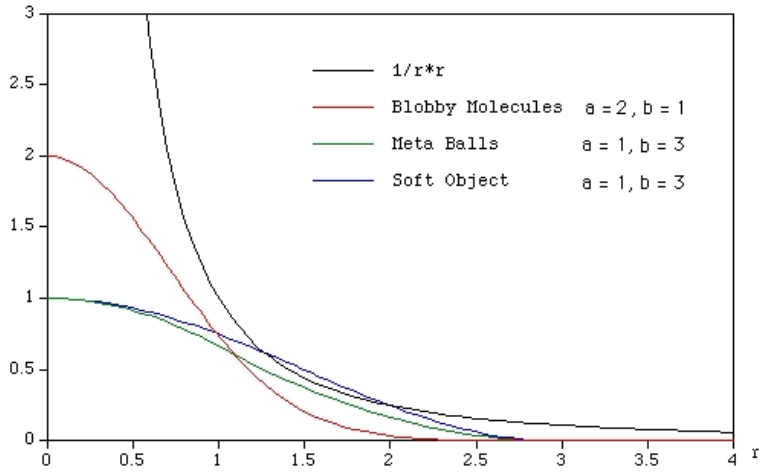



Figure 7: Comparison of different density functions [1].

equations to provide increased flexibility for adjusting object shapes. The number of additional parameters is equal to the dimension of the object: one parameter for curves and two parameters for surfaces.

### 3.2.1 Superellipse

The implicit equation of the superellipse is

$$f(x,y) = \left(\frac{x}{r_x}\right)^{\frac{2}{s}} + \left(\frac{y}{r_y}\right)^{\frac{2}{s}} = 1 \tag{9}$$

where $r_x$ and $r_y$ are the parameters of the ellipse and $s$ is the parameter of the superellipse. $s$ can be assigned any real positive value (e.g., for $s = 1$, we get an ordinary ellipse).

The corresponding parametric equations for the superellipse are expressed as

$$x(\kappa) = r_x \cos^s \kappa \tag{10}$$
$$y(\kappa) = r_y \sin^s \kappa \tag{11}$$

for $-\pi \leq \kappa \leq \pi$.

Figure 8 illustrates supercircle shapes that can be generated using various values for parameter $s$.

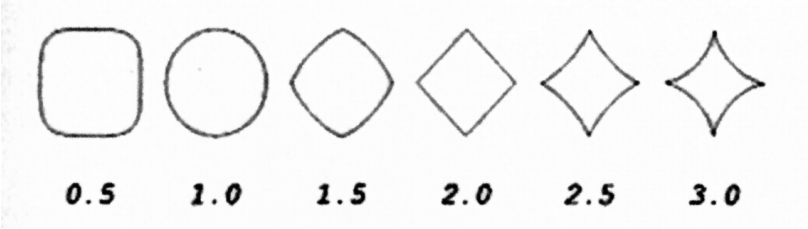



Figure 8: Superellipses plotted with different values for parameter $s$ and with $r_x = r_y$ (i.e., a supercircle).

### 3.2.2 Superellipsoid

The implicit representation of a superellipsoid is obtained from the equation of an ellipsoid by incorporating two parameters as exponents:

$$f(x,y,z) = \left[ \left( \frac{x}{r_x} \right)^{\frac{2}{s_2}} + \left( \frac{y}{r_y} \right)^{\frac{2}{s_2}} \right]^{\frac{s_2}{s_1}} + \left( \frac{z}{r_z} \right)^{\frac{2}{s_1}} = 1 \tag{12}$$

For $s_1 = s_2 = 1$, we have an ordinary ellipsoid. The corresponding parametric representation for the superellipsoid is

$$x(\kappa_1, \kappa_2) = r_x \cos^{s_1}(\kappa_1) \cos^{s_2}(\kappa_2) \tag{13}$$
$$y(\kappa_1, \kappa_2) = r_y \cos^{s_1}(\kappa_1) \sin^{s_2}(\kappa_2) \tag{14}$$
$$z(\kappa_1) = r_z \sin^{s_1}(\kappa_1) \tag{15}$$

where $-\frac{\pi}{2} \leq \kappa_1 \leq \frac{\pi}{2}$, and $-\pi \leq \kappa_2 \leq \pi$.

Figure 9 shows supersphere shapes that can be generated using various values for parameters $s_1$, and $s_2$. These and other superquadric shapes can be combined to create more complex structures.

# 4 Procedural Modeling

Procedural modeling is typically used when dealing with repetitive structures of high geometric complexity. The models are given as simple primitives and a program or a grammar that manipulates the shapes. Figure 10 shows an

[3]Sven Havemann's dissertation deals with the generative modeling language (GML). More information on GML can be found at `http://www.generative-modeling.org/`

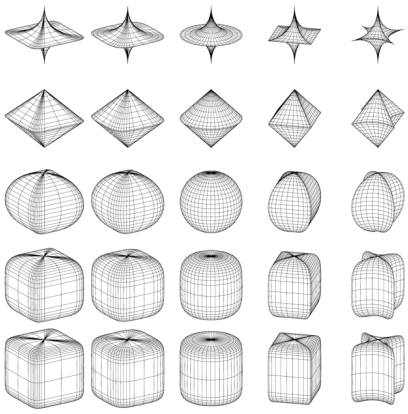


Figure 9: Shapes of the supersphere.

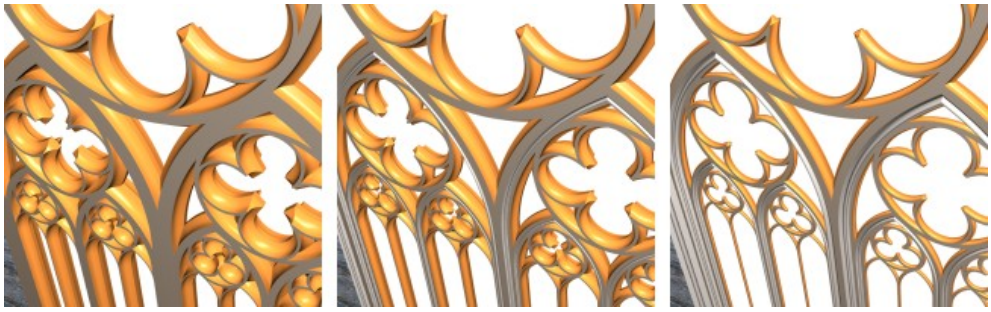


Figure 10: Window of a cathedral with increasing geometric detail. Images by Sven Havemann[3] [2].

example of a geometric object that is modeled using a procedural approach. The window of a cathedral is shown with different geometric complexity. The model with the highest geometric complexity consists of 7 million triangles. Stored as a procedural model it can be described with 126 KB of code. Procedural models often exhibit certain parameters that change the appearence of the object. Figure 11 shows an exampe of a procedural model with several parameters that define the final shape. Many objects can be modeled changing the parameters of a chair-like model.

Complex gemometric modeling tecniques employ their own languages to program the behaviour of the model generation (e.g., vegetation-, terrain simulation, etc.). A more simple example are sweeps that have fixed behaviour but have changing geometric primitives and parameters.

Figure 11: Model of a chair. Changing the parameters of the chair leads to different representations. Images by Sven Havemann[3] [2].

Solid-modeling packages usually provide a number of construction techniques, often including sweeps. Sweep representations are useful for constructing three-dimensional objects that possess translational, rotational, or other symmetries. We can represent such objects by specifying a two-dimensional shape and a curve along which the shape is moves through space. A set of two-dimensional primitives, such as circles and rectangles, are often provided as menu options. Other methods for obtaining two-dimensional figures include closed spline-curve constructions and cross-sectional slices of solid objects.

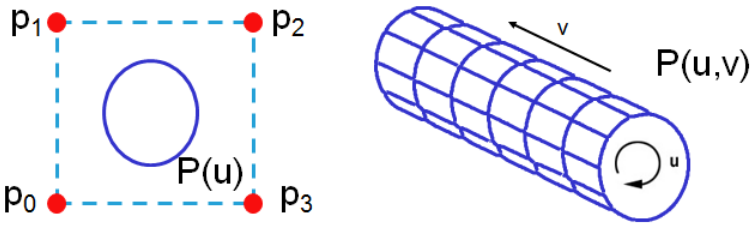



Figure 12: Translational sweep

Figure 12 illustrates a translational sweep. The periodic spline curve on the left of Figure 12 defines the object's cross section. A translational sweep is created by moving the control points $p_0 \ldots p_3$ along a straight path

perpendicular to the plane of the cross section. At given intervals along this path, the cross-sectional shape is replicated and connected with a set of lines.

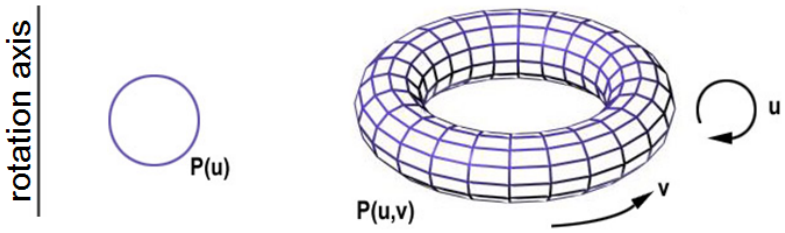



Figure 13: Rotational sweep

An example of a rotational sweep is given in Figure 13. The cross section $P(u)$ is rotated around an axis in the plane of the cross section. Any axis can be chosen for a rotational sweep. If we use a rotation axis perpendicular to the plane of the cross section in Figure 13, we generate a two-dimensional shape. A solid object (e.g., a sphere) could also be used as geometric primitive, that is used to generate another three-dimensional object. In general, a sweep is specified using a geometric primitive that is moved along an arbitrary path. For rotational sweeps, the object is moved along a circular path (from 0 to 360 degrees). For non-circular paths a curve of certain length is defined and the geometric primitive is moved along this curve. In addition the shape, size, and orientation of the geometric primitive can be varied along the sweep path.

# 5 Structure Deforming Transformations

Structure deforming transformations are used in combination with other modeling techniques to alter the shape of arbitrary objects. They mathematically define a deformation of space and the embedded objects.

**Tapering** is a non-linear scaling. The equation

$$\vec{x}' = \begin{pmatrix} f_x(\vec{x}) & 0 & 0 \\ 0 & f_y(\vec{x}) & 0 \\ 0 & 0 & f_z(\vec{x}) \end{pmatrix} \vec{x} \tag{16}$$

defines tapering by transforming each point $\vec{x} = (x, y, z) \in \Re^3$ to $\vec{x}'$ by applying scaling factors $f_x$, $f_y$, and $f_z$ that are themselves functions of $\vec{x}$.

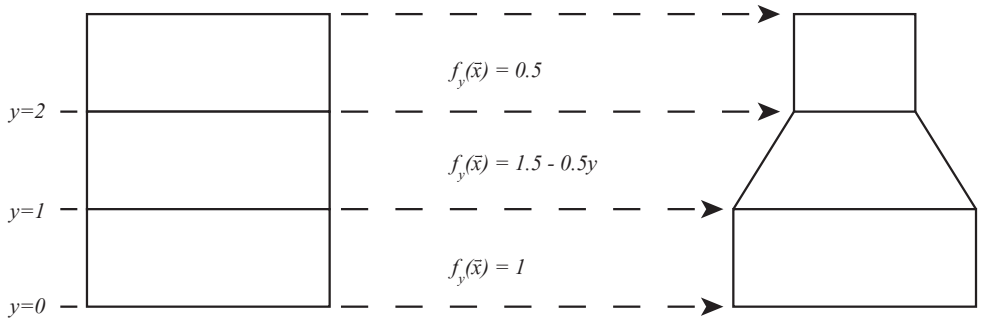



Figure 14: Tapering of a rectangular shape in 2D. The scaling parameter of the $y$-ordinate is defined as a function $f_y$ that depends on $y$ (i.e., the second ordinate of $\vec{x}$). In this example the scaling parameter for the $x$-ordinate is set to constant 1 (i.e., $f_x(\vec{x}) = 1$).

For example the function

$$f_y(\vec{x}) = \begin{cases} 1 & 0 \leq y < 1 \\ 1.5 - 0.5y & 1 \leq y < 2 \\ 0.5 & y \geq 2 \end{cases} \tag{17}$$

tapers the input rectangle as shown in Figure 14.

**Twisting** is a non-linear rotation where the rotation parameters depend on the coordinates of the input point.

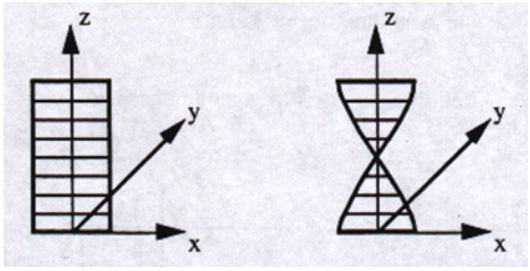



Figure 15: Twisting of a rectangular shape in 3D. The twisting parameter of the $z$-rotation is defined as a function $f$ that only depends on the ordinate $z$ in this example.

For instance twisting around the z-axis is given by the equation

$$\vec{x}' = \begin{pmatrix} cosf(\vec{x}) & -sinf(\vec{x}) & 0 \\ sinf(\vec{x}) & cosf(\vec{x}) & 0 \\ 0 & 0 & 1 \end{pmatrix} \vec{x} \tag{18}$$

where $f(\vec{x})$ is the position-dependent twisting parameter.

For instance

$$f(\vec{x}) = z \tag{19}$$

where $z$ is the third ordinate of $\vec{x}$, leads to a twist like the one shown in Figure 15.

**Bending** is also a non-linear rotation where the rotation parameters depend on the coordinates of the input point. The transformation around a line parallel to the x-axis is for example given by the coordinate-wise equations:

$$x' = x \tag{20}$$

$$y' = \begin{cases} -sin(\Theta)(z-r) + y_0 & y_{min} \leq y \leq y_{max} \\ -sin(\Theta)(z-r) + y_0 + cos(\Theta)(y - y_{min}) & y < y_{min} \\ -sin(\Theta)(z-r) + y_0 + cos(\Theta)(y - y_{max}) & y > y_{max} \end{cases} \tag{21}$$

$$z' = \begin{cases} cos(\Theta)(z-r) + y_0 + r & y_{min} \leq y \leq y_{max} \\ cos(\Theta)(z-r) + y_0 + r + sin(\Theta)(y - y_{min}) & y < y_{min} \\ cos(\Theta)(z-r) + y_0 + r + sin(\Theta)(y - y_{max}) & y > y_{max} \end{cases} \tag{22}$$

Figure 16 graphically defines the parameters of Equations 21 and 22 for a bend around the $x$ axis.

# References


[1] P. Bourke. Implicit surfaces. `http://paulbourke.net/miscellaneous/implicitsurf/`, 1997.

[2] S. Havemann. *Generative Mesh Modeling*. PhD thesis, TU Braunschweig, 2005.

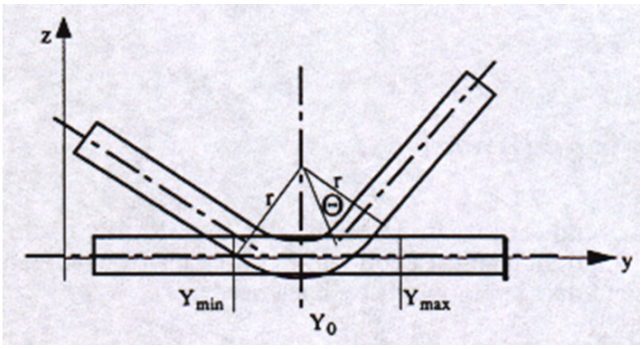



Figure 16: Bending of a rectangular shape. A bending around the $x$ axis transforms the coordinates $y$ and $z$. The bending depends on the radius $r$ the angle $\Theta$ and the parameters $y_{min}$ and $y_{max}$.

[3] D. Hearn and M. P. Baker. *Computer graphics (2nd ed.): C version.* Prentice-Hall, Inc., 1997.

[4] M. Lentine, W. Zheng, and R. Fedkiw. A novel algorithm for incompressible flow using only a coarse grid projection. *ACM Transactions on Graphics*, 29(4), 2010.

[5] F. Losasso, J. O. Talton, N. Kwatra, and R. Fedkiw. Two-way coupled sph and particle level set fluid simulation. *IEEE Transactions on Visualization and Computer Graphics*, 14(4):797–804, 2008.

[6] B. Wyvill, A. Guy, and E. Galin. The blobtree. warping, blending and boolean operations in an implicit surface modelling system. Technical report, 1999.