LAB REPORT: LAB 4

TNM079, MODELING AND ANIMATION

Johan Beck-Norén
johbe559@student.liu.se

Sunday 28th July, 2013

Abstract

This report covers the fourth lab in a series of six labs for the course TNM079 Modeling and Animation held at Linköping University. This lab covers the subject of implicit surfaces. It is assumed that the reader is familiar with the basic concepts and theory since this report will mostly focus on the assignments completed for this lab. Completed assignments include implementation of CSG (Constructive Solid Geometry) operators, quadric surfaces, discrete gradient- and curvature operators for implicits, and super-elliptic blending using a density function. The results show that implicit surfaces are very useful in computer graphics. They have desirable geometrical properties since they cannot self-intersect, and have no holes.

1 Introduction

The purpose of this lab is to gain a theoretical understanding of implicit surfaces and a few boolean operators applicable to these surfaces. This report will describe how these operators were implemented, as well as quadric surfaces, discrete gradient- and curvature operators, and super-elliptic blending using a density function.

2 Assignments

2.1 Implementing CSG operators

CSG (Constructive Solid Geometry) operators are a collection of boolean operators. Since the implicit surfaces used are defined by negative-inside sign convention, we use this to implement these boolean operators. The operators implemented are $Union$, $Intersection$, and $Difference$. With the sign convention used, it is easy to determine the inside and outside of an implicit surface. This enables us to implement the boolean operators as follows:

$$\textbf{Union:}(A,B) = A \cup B = min(A,B) \tag{1}$$
$$\textbf{Intersection:}(A,B) = A \cap B = max(A,B) \tag{2}$$
$$\textbf{Difference:}(A,B) = A - B = max(A,-B) \tag{3}$$

In equation (1) through (3) we see that thanks to the sign convention used these CSG operators are reduced to simple operations from set theory. In equation (1), taking the minimum value

at each grid point will ensure that a negative value for any of the implicit surfaces will remain negative, while positive values might be overwritten if any of the implicit surfaces has a negative value at that grid point. Since we use a negative-inside sign convention, this will effectively join the two implicit surfaces together, thus making it a *Union* operator. The *Intersection* operator, equation (2), is performed in a similar fashion. Taking the maximum value for each grid point will ensure that a positive value in any of the two implicit surfaces will remain positive, and only grid points where both implicit surfaces have negative values will remain negative, making it an *Intersection* operator. The last operator has a bit more to it, but basically follows the same routine. By inverting the signs of the grid points for one implicit surface and then taking the maximum value at each grid point, every point inside the negated implicit surface will be marked as "outside" (positive value), and the only remaining negative values will be those grid points inside the unaffected implicit surface not shared with the negated surface. This procedure makes it an *Difference* operator, equation (3).
The implementations of the CSG operators can be found in the classes *Union*, *Intersection* and *Difference* in the file *CSG.h*.

2.2 Implementing quadric surfaces

A quadric surfaces is defined by a quadric implicit function. In three dimensions it can be described on matrix form for a point $\mathbf{p}$ as follows:

$$\mathbf{p}^T\mathbf{Q}\mathbf{p} = \begin{bmatrix} x & y & z & 1 \end{bmatrix} \begin{bmatrix} A & B & C & D \\ B & E & F & G \\ C & F & H & I \\ D & G & I & J \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{4}$$

This general quadric function can describe a total of 17 different standard surfaces including planes, cylinders, cones, paraboloids and more. Examples of quadric functions for a plane and a paraboloid are described in equations (5) and (6) respectively.

$$f(x,y,z) = ax + by + cz = 0 \tag{5}$$

$$f(x,y,z) = x^2 \pm y^2 - z = 0 \tag{6}$$

The normal for a quadric surfaces is defined as in equation (7) and the gradient is defined as in equation (8). We can apply these differentiations directly with the coefficient matrix $\mathbf{Q}$ since the surface is known analytically.

$$\nabla f = \vec{n}(\vec{n} \cdot \nabla f) \rightarrow \vec{n} = \pm\frac{\nabla f}{|\nabla f|} \tag{7}$$

$$\nabla f(x,y,z) = 2\begin{bmatrix} A & B & C & D \\ B & E & F & G \\ C & F & H & I \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = 2\mathbf{Q}_{sub}\mathbf{p} \tag{8}$$

Implemented quadric surfaces include planes, cylinders, spheres and ellipsoids, cones, paraboloids and hyperboloids. The code for the implementations of the normal and gradient calculations can be found in the file *Quadric.cpp*, and the implementations of the different quadric surfaces can be found in the file *FrameMain.cpp*.

2.3 Implementing the discreet gradient operator for implicits

Since computers use finite schemes as opposed to infinite descriptions, we need to use finite differences when evaluating the gradient for an implicit surface. We reside in three dimensional space. We must therefore perform the partial finite difference three times, once for each "direction". Equation (9) results in the derivative for the function f in the x direction. The equations for the y and the z directions are the same, substituting x in equation (9) with the respective dimension sought. As seen in the equation, we use values from both sides of the grid point being solved for. This type of approximation is called a central difference approximation. Δx refers to a small value used to approximate the definition of the differential operator, since floating point precision in computers inhibits us from allowing a number to become arbitrarily small.

$$D_x \approx \frac{f(x + \Delta x) - f(x - \Delta x)}{2\Delta x} \tag{9}$$

The code for the discrete gradient implementation can be found in the file *Implicit.cpp*.

2.4 Implementing the discrete curvature operator for implicits

As discussed in the previous section, we need to use a discrete scheme for calculating differentials. When calculating curvature we need the second derivative of the implicit function. There are many ways of approximating the mean curvature of an implicit surface. One example can be seen in equation (10), and its discrete counterpart for the x direction in equation (11) where, again, Δx refers to some small value.

$$\kappa \approx \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} + \frac{\partial^2 f}{\partial z^2} \tag{10}$$

$$\frac{\partial^2 f}{\partial x^2} = D_{xx} \approx \frac{f(x + \Delta x) - 2f(x) + f(x - \Delta x)}{(\Delta x)^2} \tag{11}$$

The code implementing the discrete curvature can be found in the file *Implicit.cpp*.

2.5 Implementing super-elliptic blending

When using the CSG operators described in the first assignment in this report, we notice that the operators sometimes result in jagged creases in our implicit surfaces. This is due to the fact that the interface will have undefined normals and only be C^0 continuous along the intersection between the surfaces the operator was applied to. One way to remedy this and achieve a smooth transition between implicit surfaces is to use density functions. Instead of defining an implicit surface with negative meaning inside, positive outside, and a zero-level surface (assuming a negative-inside convention), we describe the density D_A of an implicit geometry A:

$$D_A(\mathbf{x}) = \begin{cases} > 1, & \text{if x is inside the surface} \\ = 1, & \text{if x is on the surface} \\ [0,1), & \text{if x is outside the surface} \end{cases} \tag{12}$$

We transform the implicit surface A into the density function D_A with equation (13), and perform for example union with equation (14).

$$D_A(\mathbf{x}) = e^{-A(\mathbf{x})} \tag{13}$$

$$D_{A \cup B} = \left(D_A^p + D_B^p \right)^{\frac{1}{p}} \tag{14}$$

An intersection is performed by negating the value p in equation (14), and the difference is obtained by negating either A or B and then performing an intersection. To transform the results back to an implicit surface we have to inverse equation (13). We do this by using the natural logarithm:

$$D_A(\mathbf{x}) = e^{-A(\mathbf{x})} \rightarrow \ln(D_A(\mathbf{x})) = \ln(e^{-A(\mathbf{x})}) \rightarrow A(\mathbf{x}) = -\ln(D_A(\mathbf{x})) \tag{15}$$

The code implementing the density function can be found in the classes *BlendedUnion*, *BlendedIntersection* and *BlendedDifference* in the file *CSG.h*.

3 Results

3.1 CSG operators

The implemented operators work as expected. Notice the jagged creases in figure 1 due to the lack of blending, especially for the difference operator (right) along the intersections between the spheres.

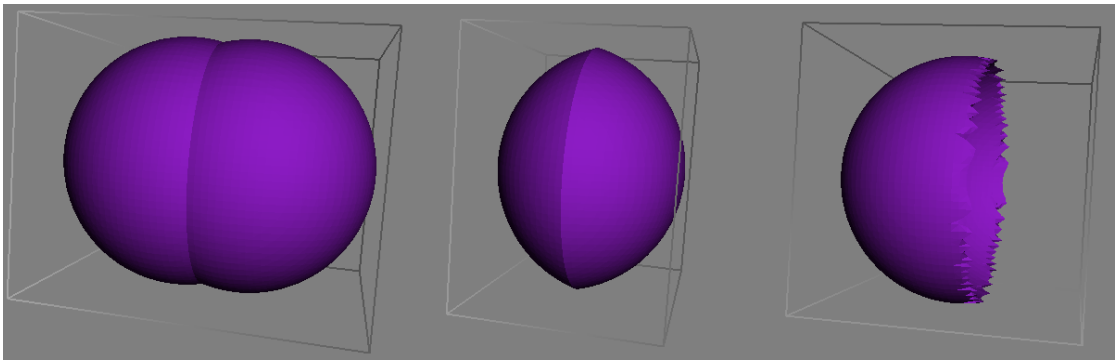

Figure 1: CSG operators demonstrated on two implicit spheres. From left to right; Union, intersection, and difference.

3.2 Quadric surfaces

The quadric surfaces work well and look correct (figure 2). They were defined by their respective quadric matrices as described in section 2.2.

3.3 Discrete gradient operator for implicits

The discrete gradient operator works well as seen in figure 3. One thing to take note of is how the gradient is affected when changing Δx in equation (9). Increasing Δx results in gradients with the same direction, but with smaller magnitude, while decreasing Δx results in a greater magnitude. This can be explained by the fact that Δx is the denominator of equation (9).

3.4 Discrete curvature operator for implicits

The discrete curvature operator works reasonably well. A rather simple approximation is used to calculate the mean curvature. Imperfections in this approximation can skew the results when visually examining the curvature, The results are better when visualizing the curvature using an HSV colormap with manually set ranges. In figure 4 the difference operator has been applied to two implicit spheres and the curvature is visualized using an HSV colormap with a range of $[-50, 50]$.

3.5 Super-elliptic blending

Super-elliptic blending using density functions works well and results in a smooth transition between implicit surfaces. An example can be seen in figure 5 where the union operator has been applied to two implicit spheres, with $p = 2$ in equation (14).

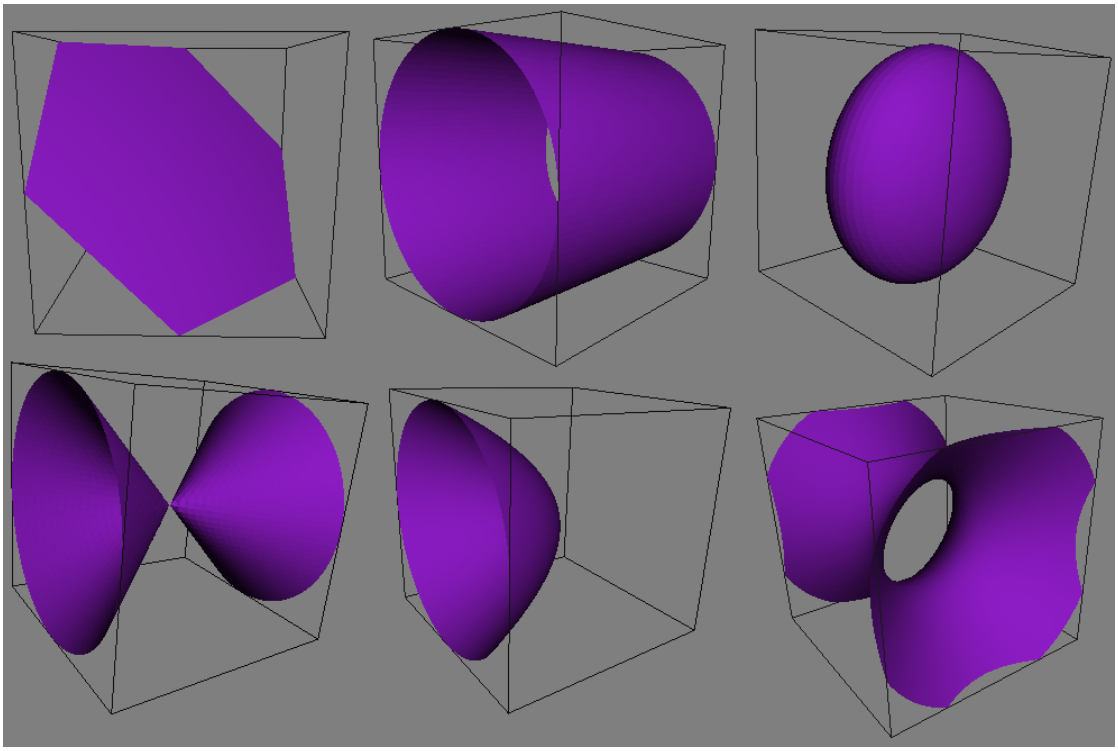

Figure 2: Quadric surfaces defined by quadric implicit functions. Left to right, top to bottom; Plane, cylinder, ellipsoid, cone, paraboloid, and hyperboloid.

4 Conclusion

Although a little abstract at first, implicit surfaces turns out to be a powerful method to create complex shapes using simple boolean operators. Regarding quadric surfaces, being able to create such a plethora of primitive shapes using the same quadric matrix can really come in handy.

The discrete schemes for calculating gradient and curvature for implicit surfaces are interesting. They rely heavily on the size chosen for Δx (as well as Δy and Δz respectively). A large value will result in ”low resolution” if you will, and may miss high frequency elements if the surface, such as sharp features of creases. This was especially noticeable when visualizing the curvature in figure 4, where a large value would miss the increase in curvature around the crease.

5 Lab partner and grade

The assignments in this lab were completed together with Hans-Christian Helltegen (hanhe945). I have completed all assignments marked with (*) and (**), therefore I should qualify for grade 5.

References

[1] Gunnar Läthén, Ola Nilsson, Andreas Söderström, Stefan Lindholm *Mesh Data Structures*. TNM079 Modeling and Animation, Linköping University, 2013.

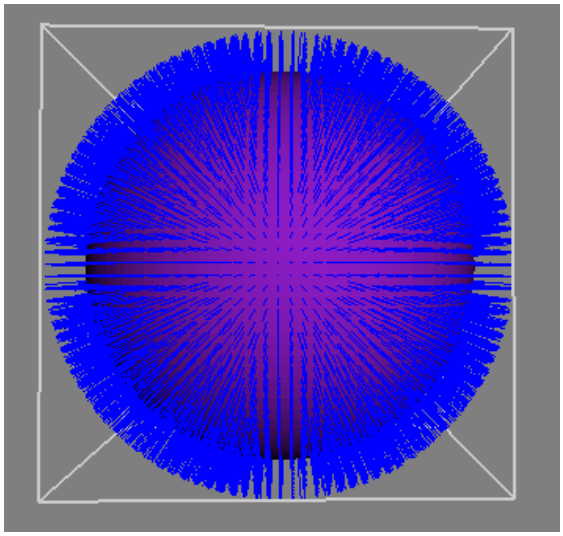

Figure 3: Gradients for an implicit sphere shown in blue.

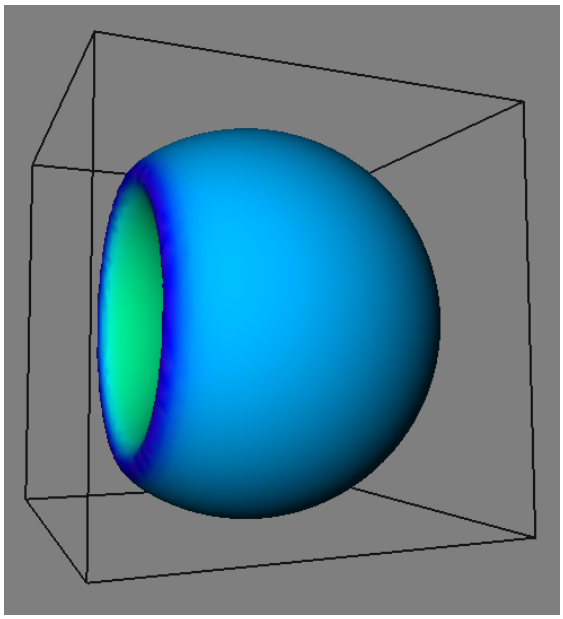

Figure 4: Mean curvature visualized using an HSV colormap with a range of $[-50, 50]$.

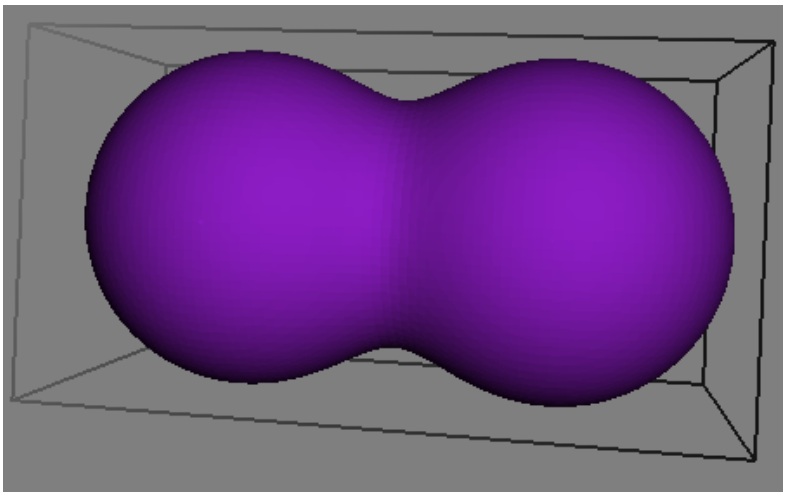

Figure 5: Super-elliptic blending. Union operator applied to two implicit spheres with $p = 2$ in the density function used.