IMPLICIT SURFACES AND MODELING

MODELING AND ANIMATION

Dan Englesson

Wednesday 11th May, 2011
09:07

Abstract

In this lab which was done in the course TNM079, Modeling and animation in spring 2011 at Linkopings university, implicit surfaces in form of Implicit quadric surfaces and modeling with boolean operators such as union, intersection and difference were implemented and the operators were later improved by using super-elliptic blending with density functions which made the intersection edge much smoother and removed the C^0 continuity problem at these points. These three boolean operators are powerful tools when it comes to constructive solid geometry CSG, and worked very well when implemented. Implementations of the normal and curvature of an implicit surface was also done by using the gradient. The gradient was approximated by using a central difference approximation and a small value of ϵ gave very nice similarities between the generated normal from the marching-cube faces and the calculated normal from the normalized gradient. While a larger value of ϵ gave not so similar results between the generated normal and the normal calculated from the gradient, especially around sharp edges and "noisy" surfaces.

1 Introduction

The lab was done in the course TNM079, Modeling and animation, in spring 2011 and was performed to get a better understanding of implicit surfaces and implicit modeling. Implicit quadric surfaces where implemented, for example ellipsoids, cones and planes. Three different boolean operations where implemented and they were union, intersection and difference. These boolean operations where improved by introducing super-elliptic blending with density functions which removes the C^0 continuity problem at intersection edges. Implementation of the gradient and curvature for an implicit surface is also implemented.

2 Method

2.1 Implement CSG operators

In this assignment three different boolean operators were implemented. The boolean operators were union, intersection and difference. By taking the union of two implicit objects [1] A and B, one takes the minimum value of A and B, see equation (1). For the intersection the max value is instead taken and for the difference the maximum value of A and a negated implicit surface B is taken, see equation (1).

$$\textbf{Union}(A,B) = A \cup B = \min(A,B) \tag{1a}$$
$$\textbf{Intersection}(A,B) = A \cap B = \max(A,B) \tag{1b}$$
$$\textbf{Difference}(A,B) = A - B = \max(A,-B) \tag{1c}$$

Equation (1) were implemented in the CSG.h file in the GetValue-function for each of the three boolean operator classes. These operations creates a new implicit surface that can be rotated and translated in the world and therefor the comparison of the min/max values needs to be in object-coordinates to be able to rotate and translate the new implicit object.

2.2 Implement the quadric surface

An implicit quadratic function[2] can be described on matrix form as in the following equation:

$$\mathbf{p}^T\mathbf{Q}\mathbf{p} = \begin{bmatrix} x & y & z & 1 \end{bmatrix} \begin{bmatrix} A & B & C & D \\ B & E & F & G \\ C & F & H & I \\ D & G & I & J \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{2}$$

$$\begin{aligned} f(x,y,z) = Ax^2 + 2Bxy + 2Cxz \\ + \quad 2Dx + Ey^2 + 2Fyz \\ + \quad 2Gy + Hz^2 + 2Iz \\ + \quad J \end{aligned} \tag{3}$$

When multiplying the transposed position p^T to the matrix Q and position p, as in equation (2), one gets a quadratic equation $f(x,y,z)$, see equation (3). By changing this Q matrix one can get very interesting shapes such as ellipsoids, cones, hyperboloids and paraboloids for example. Here follow some examples of shapes that can be derived by changing the coefficients in the Q matrix according to the function $f(x,y,z)$.

- Planes

$$\begin{aligned} f(x,y,z) &= ax + by + cz = 0 \\ \mathbf{Q} &= \begin{bmatrix} 0 & 0 & 0 & a/2 \\ 0 & 0 & 0 & b/2 \\ 0 & 0 & 0 & c/2 \\ a/2 & b/2 & c/2 & 0 \end{bmatrix} \end{aligned} \tag{4}$$

- Cylinders

$$\begin{aligned} f(x,y,z) &= x^2 + y^2 - 1 = 0 \\ \mathbf{Q} &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix} \end{aligned} \tag{5}$$

- Spheres and ellipsoids

$$
\begin{aligned}
f(x,y,z) &= x^2 + y^2 + z^2 - 1 = 0 \\
f(x,y,z) &= \tfrac{x^2}{a^2} + \tfrac{y^2}{b^2} + \tfrac{z^2}{c^2} - 1 = 0 \\
\mathbf{Q} &= \begin{bmatrix} 1/a^2 & 0 & 0 & 0 \\ 0 & 1/b^2 & 0 & 0 \\ 0 & 0 & 1/c^2 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}
\end{aligned} \tag{6}
$$

With the Q matrix however each and every point needs to be transformed into world space, which is very costly. Therefor the Q matrix is transformed into world space once and instead evaluates the function for transformed points in global space, p'. To get the transformed point p' an affine matrix M can be applied to the point p, see equation (7).

$$
\mathbf{p}' = \mathbf{M}\mathbf{p} \quad \Leftrightarrow \quad \mathbf{p} = \mathbf{M}^{-1}\mathbf{p}' \tag{7}
$$

Where if inputed into the quadric matrix form in equation (2) gives the following:

$$
\begin{aligned}
f(\mathbf{p}) &= f(\mathbf{M}^{-1}\mathbf{p}') \\
&= \left(\mathbf{M}^{-1}\mathbf{p}'\right)^T \mathbf{Q}(\mathbf{M}^{-1}\mathbf{p}') \\
&= \left(\mathbf{p}'\right)^T \left(\mathbf{M}^{-1}\right)^T \mathbf{Q}(\mathbf{M}^{-1})\mathbf{p}'
\end{aligned}
$$

Gives the conclusion:

$$
\begin{aligned}
f(\mathbf{p}) &= (\mathbf{p}'t)^T \mathbf{Q}'\mathbf{p}', \text{where} \\
\mathbf{Q}' &= (\mathbf{M}^{-1})^T \mathbf{Q}\mathbf{M}^{-1}
\end{aligned} \tag{8}
$$

Q' was calculated *once* and was implemented in the Quadric constructor in Quadric.cpp, and later used in equation (8) which was implemented in the getValue-function.

$$
\nabla f(x,y,z) = 2 \begin{bmatrix} A & B & C & D \\ B & E & F & G \\ C & F & H & I \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = 2\mathbf{Q}_{sub}\mathbf{p} \tag{9}
$$

The gradient was calculated according to equation (9), in Quadric.cpp in the getGradient-function. The gradient is used in order to calculate the normal for an implicit surface according to equation (10).

$$
\begin{aligned}
\tfrac{\partial f}{\partial \vec{e}_1} = \tfrac{\partial f}{\partial \vec{e}_2} &= 0 \\
&\Rightarrow \quad \nabla f = \vec{n}(\vec{n} \cdot \nabla f) \\
&\Rightarrow \quad \vec{n} = \pm \tfrac{\nabla f}{|\nabla f|}
\end{aligned} \tag{10}
$$

2.3 Implement the discrete gradient operator for implicits

$$\frac{\partial f}{\partial x} \equiv \lim_{h \to 0} \frac{f(x_0 + h) - f(x_0)}{h} \tag{11}$$

In order to calculate normals, see equation (10), calculation of differentals on the implicit surface is needed, which is in theory done with equation (11). However equation (11) needs to be approximated and the reason is that h cannot be arbitrary small without any numerical errors occurring, and therefor a small value ϵ is chosen instead of h , see equation (12).

$$D_x \equiv \lim_{h \to 0} \frac{f(x_0 + h) - f(x_0)}{h} \approx \frac{f(x_0 + \epsilon) - f(x_0)}{\epsilon} \tag{12}$$

However the central difference approximation, equation (13),was used instead of the normal discrete difference operator in equation (12), because of the fact that equation (12) is asymmetric and therefor more weight will be put on the positive x values. By using equation (13) the asymmetry is gone and is therefor a better approximation.

$$\begin{aligned} D_x \equiv \lim_{h \to 0} \frac{f(x_0 + h) - f(x_0)}{h} \approx \\ \approx \frac{f(x_0 + \epsilon) - f(x_0 - \epsilon)}{2\epsilon} \end{aligned} \tag{13}$$

Equation (13) was applied three times, in Implicit.cpp in the getGradient-function, one for each dimension (x, y, z).

2.4 Implement the discrete curvature operator for implicits

The curvature was approximated by adding three second partial derivatives, as in equation (14), by approximating the second partial derivative with equation (15), where ϵ is a sufficiently small value.

$$\kappa \approx \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} + \frac{\partial^2 f}{\partial z^2} \tag{14}$$

$$\frac{\partial^2 f}{\partial x^2} = D_{xx} \approx \frac{f(x_0 + \epsilon) - 2f(x_0) + f(x_0 - \epsilon)}{\epsilon^2} \tag{15}$$

Equation (14) and equation (15) were implemented in Implicit.cpp in the getCurvature-function. To be able to see the curvature estimate the ”Curvature” visualization mode and a known color-map such as HSV or Jet was enabled. Better control over the visualization is achieved by manually setting the ranges of the color-map so one can relate the color to the convexity and concavity.

2.5 Implement super-elliptic blending

A problem with simple boolean operators, see equation (1) is that the normal is not defined at the intersection points because it is C^0 continuous at the intersection point. The result is sharp and some times ill-deformed edges. To get rid of these sharp edges and achieve higher than C^0 continuity, *densityfunctions* is used. The density functions describes the the implicit geometry $A's$ ”density” D_A, see equation (16).

$$D_A(\mathbf{x}) = \begin{cases} > 1 & \text{if } \mathbf{x} \text{ is inside the surface} \\ = 1 & \text{if } \mathbf{x} \text{ is on the surface} \\ \in [0, 1) & \text{if } \mathbf{x} \text{ is outside the surface} \end{cases} \tag{16}$$

An implicit surface A is transformed into a density function as in equation (17), assuming that it is negative inside the implicit surface.

$$D_A(\mathbf{x}) = e^{-A(\mathbf{x})} \tag{17}$$

Super-elliptic blending on different boolean operators can then be performed according to the following equations:

$$D_{A\cup B} = \left(D_A^p + D_B^p\right)^{1/p} \tag{18a}$$

$$D_{A\cap B} = \left(D_A^{-p} + D_B^{-p}\right)^{-1/p} \tag{18b}$$

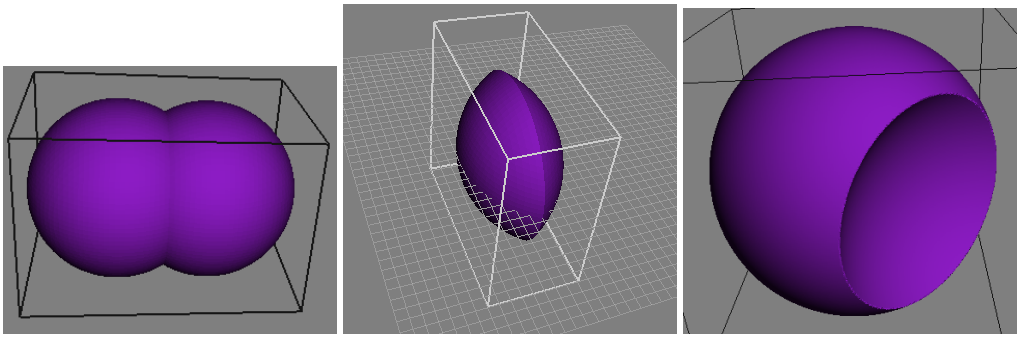
$$D_{A-B} = \left(D_A^{-p} + D_B^{p}\right)^{-1/p} \tag{18c}$$

The lower the p is the more blending is performed and when $p \to \infty$ the same result is achieved as in equation (1).The three different equations, equation (18a), equation (18b) and equation (18c) was implemented in the classes *blendedUnion*, *blendedIntersection* and *blendedDifference* in CSG.h. However to visualize it, the distance function needs to be transformed back to an implicit function. This is easily done by taking the inverse of equation (17), see equation (19) and put in the boolean distance function.

$$A(\mathbf{x}) = -ln(D_A(\mathbf{x})) \tag{19}$$

3 Results

3.1 CSG operators and sampling distance



(a) Union (b) Intersection (c) Difference

Figure 1: Different boolean operators. Note the sharp edges around the intersection

Figure 1 shows three types of boolean operators, Union, Intersection and Difference, where at the intersection there are only C^0 continuity which gives a very sharp edge and there are no normals defined for these points.

The implicit object are converted into polygonal objects by the marching-cube algorithm and can therefor be sampled with different sampling rate. Figure 1 (c) is sampled with a sampling distance of 0.01 and figure 2 shows a sampling distance of 0.05 instead which gives a poorer result, especially around sharp edges.

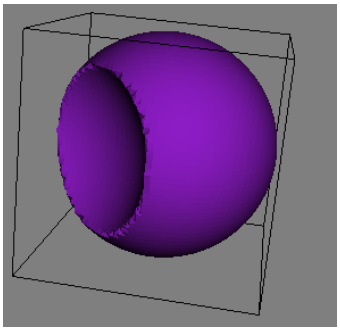


Figure 2: Difference, with a sampling distance of 0.05.

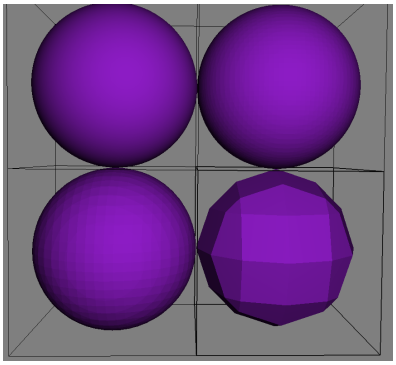
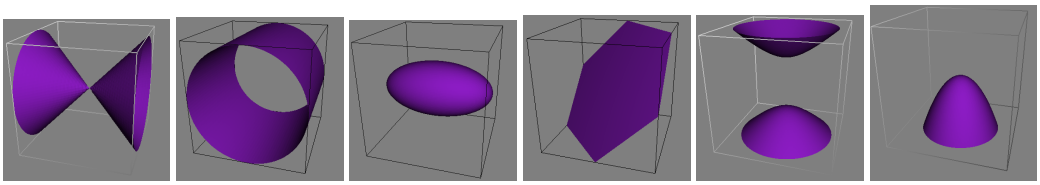


Figure 3: Implicit sphere representation with different sampling distances. Top-left 0.01,Top-right 0.05, Lower-left 0.5, Lower-right 0.1

In figure 3 different sampling distances for the marching-cube algorithm are applied to an implicit sphere in order to visualize it. The marching-cube algorithm was already implemented and is not part of this lab and will therefor not be further discussed. One clearly sees that the top two spheres in figure 3 approximates the implicit sphere quite good, and the bottom two does not approximate it as good, especially the lower-right with a sampling distance of 0.1. Any boolean operation would be poorly visualized with this sampling distance.

3.2 Implement the quadric surface

By changing the Q matrices according to a function $f(x, y, z)$ as shown in section 2.2 many different shapes can be derived, see figure 4.



(a) Cone (b) Cylinder (c) Ellipsoid (d) Plane (e) Hyperboloid (f) Paraboloid

Figure 4: Quadric implicit surfaces

For their corresponding Q matrix, see *AppendixA*.

The calculated gradient for an implicit quadric surface is shown in figure 5 which is parallel to the normals calculated from the faces created by the marching-cubes algorithm. The gradient is shown as blue lines and the normal is shown as red lines in figure 5.

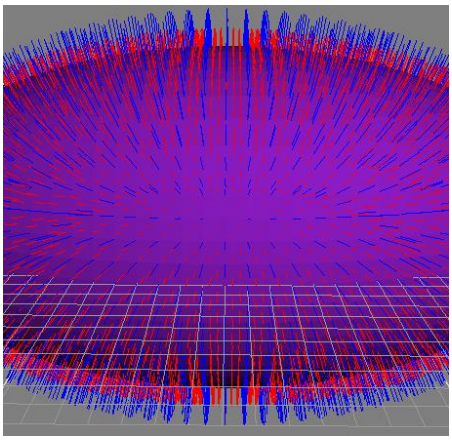
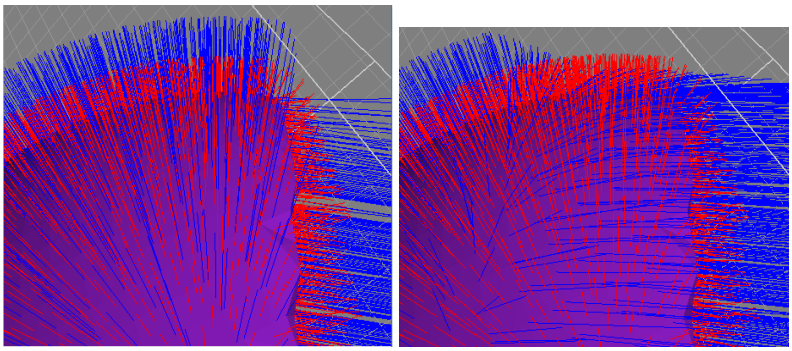


Figure 5: Gradient visualized as blue lines, and the red lines as the normal.

3.3 Implement the discrete gradient operator for implicits



(a) small ϵ (b) large ϵ

Figure 6: Gradient visualized as blue lines and the red lines as the normal.

As seen in figure 6 (a) the gradient is approximativly parallel to the normal because of the very small value of ϵ. By making ϵ larger, a larger distance away from the original point is taken into account which gives large errors around sharp edges, and surfaces that varying much, see figure 6 (b). As mentioned before, the gradient should be parallel to the face-normal, which is not the case when a arbitrary large value of ϵ is chosen.

3.4 Implement the discrete curvature operator for implicits

The discrete curvature was implemented for implicit objects as in section 2.4 and visualized using the curvature visualization mode and a chosen color-map, see figure 8 where the range for the color-map was $[-10, 10]$. The color-map used was the *Jet* color-map, see figure 7.

Figure 7: Jet color-map

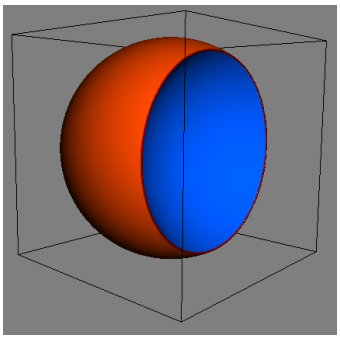
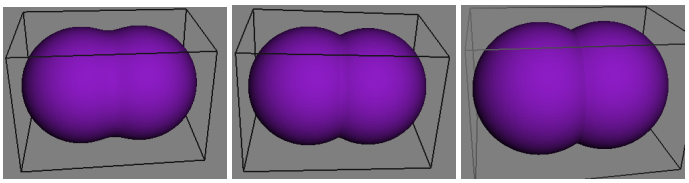


Figure 8: Curvature visualization with Jet color-map and range $[-10, 10]$. Orange and red, different level of convexity. Blue is concavity.

By comparing the colors with figure 7 one sees that the convex surface is represented in orange and red color according to the amount of convexity. The blue surface is according to figure 7 negative and therefore convex. Figure 8 is a good visualization of that the curvature works for implicit surfaces.

3.5 Implement super-elliptic blending

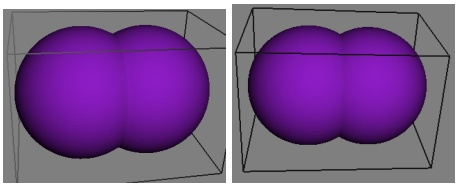
By using super-elliptic blending by using distance functions instead a smoother edge can be applied and there is a higher order than C^0 continuity at these intersection-points, see figure 9.



(a) Blended union p = 7 (b) Blended union p = 50 (c) Blended union p = 100

Figure 9: Different values of p

Figure 9 (a) shows a very nice smooth transition after union operation, and by looking at figure 9 (a),(b) and (c) one can conclude that the edge at the intersection gets sharper when increasing p. By looking at the figures in figure 10, one can back up the theory that super-elliptic blending converges to the ordinary boolean operations when $p \to \infty$.



(a) Blended union p = 100 (b) Union without blending

Figure 10: Comparision between a large value of p and union without any super-elliptic blending

4 Conclusion

This lab gave a good understanding of how to implement implicit quadric surfaces and implicit *constructive solid geometry* (CSG)[3] modeling with different boolean operator.

Here are some conclusion that can be drawn from this lab:

Simple boolean operations such as union, intersection and difference is three very powerful tools in CSG modeling, however there will always be a sharp intersection edge because of the C^0 continuity. This can be changed by instead convert the implicit functions to density functions and do the boolean operations on the density functions and then go back to implicit functions again in order to visualize it. By letting $p \to \infty$ equation (18a) will converge to an normal boolean operation as in equation (1).

The normal of an implicit surface can be approximated with the normalized gradient, where the gradient is approximated very well when a sufficiently small value of ϵ is chosen. With a larger value of ϵ , gradients at sharp edges or at ”noisy” surfaces will differ from the actual normal quite a lot.

By changing the constants in the Q matrix according to a function $f(x, y, z)$ many different shapes can be derived, such as Cones, Cylinders, Ellipsoids, Spheres and planes fore example, and by transforming the Q matrix into world space once each and every point p does not need to be transformed into world coordinates every time which is very costly.

Lab partner and grade

The lab was carried out by me, Dan Englesson, and my lab partner Emil Brissman. We implemented all mandatory assignments as well as all assignments needed for grade 4 and 5. Therefore me and my lab partner have full-filled the requirements for grade 5.

References

[1] Gunnar Läthén Ola Nilsson Andreas Söderström. Implicit surfaces and modeling. pages 1–2.

[2] Gunnar Läthén Ola Nilsson Andreas Söderström. Implicit surfaces and modeling. page 4.

[3] Gunnar Läthén Ola Nilsson Andreas Söderström. Implicit surfaces and modeling. page 7.

5 Appendix A

The corresponding Q-matrices.

- Cones

$$f(x,y,z) = x^2 + y^2 - z^2 = 0$$
$$\mathbf{Q} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{20}$$

- Cylinders

$$f(x,y,z) = x^2 + y^2 - 1 = 0$$
$$\mathbf{Q} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix} \tag{21}$$

- Spheres and ellipsoids

$$f(x,y,z) = x^2 + y^2 + z^2 - 1 = 0$$
$$f(x,y,z) = \tfrac{x^2}{a^2} + \tfrac{y^2}{b^2} + \tfrac{z^2}{c^2} - 1 = 0$$
$$\mathbf{Q} = \begin{bmatrix} 1/a^2 & 0 & 0 & 0 \\ 0 & 1/b^2 & 0 & 0 \\ 0 & 0 & 1/c^2 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix} \tag{22}$$

- Planes

$$f(x,y,z) = ax + by + cz = 0$$
$$\mathbf{Q} = \begin{bmatrix} 0 & 0 & 0 & a/2 \\ 0 & 0 & 0 & b/2 \\ 0 & 0 & 0 & c/2 \\ a/2 & b/2 & c/2 & 0 \end{bmatrix} \tag{23}$$

- Hyperboloids

$$f(x,y,z) = x^2 + y^2 - z^2 \pm 1 = 0$$
$$\mathbf{Q} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & \pm 1 \end{bmatrix} \tag{24}$$

- Paraboloids

$$f(x,y,z) = x^2 \pm y^2 - z = 0$$
$$\mathbf{Q} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \pm 1 & 0 & 0 \\ 0 & 0 & 0 & -1/2 \\ 0 & 0 & -1/2 & 0 \end{bmatrix} \tag{25}$$