IMPLICIT SURFACES AND MODELING

MODELING AND ANIMATION, LAB 4

Albert Cervin
albce943@student.liu.se

Wednesday 25th May, 2011

Abstract

In this paper I will present theory and results when implementing an implicit geometry and constructive solid geometry framework. This is the fourth lab in a lab series consisting of six labs in the course TNM079 - Modeling and Animation at Linköping University. Topics that will be covered include a general introduction to implicit surfaces and constructive solid geometry (hereafter CSG). This includes implementation of quadric implicit objects and boolean operations for CSG and furthermore also smoothing operations for the boolean operations. The results will then be presented showing that the boolean operations give good results and that implicits have nice geometrical properties and behave well. There are no holes, self intersection or similar artifacts. Implicit surfaces are very useful in computer graphics since they are well behaved geometrically.

1 Introduction

The course TNM079 - Modeling and Animation consists of six lab sessions to be performed. Three of these sessions are to be presented in a report presenting what has been done and how it has been done. This report describes my work in the fourth lab session covering implicit surfaces and constructive solid geometry.

1.1 Implicit surfaces

There are two different types of surfaces in computer graphics: *explicit* and *implicit*. An *explicit* surface is what is known most commonly as a mesh representation of a surface. This means that the surface is defined by explicit points in space placed in a mesh. An *implicit* surface on the other hand is defined as a mathematical function. A simple example of an implicit surface is a sphere that is defined as $f(x,y,z) = x^2 + y^2 + z^2$. Implicit surfaces describe volumes in a convenient way and are therefore often used in CSG modeling.

To be able to render an implicit surface in 3D, a specific set of the implicit function has to be chosen such that

$$S(C) \equiv x : f(x) = C \tag{1}$$

This is called an *iso-surface* or a *level-set*.

When working with implicit surfaces it is important to be able to classify if a point in space lies outside, inside or on the border of an object. This is defined as in equation 2.

$$\begin{array}{ll} f(x) < C & \text{(inside)} \\ f(x) > C & \text{(inside)} \\ f(x) = C & \text{(on surface)} \end{array} \tag{2}$$

One important property of the level-set is that it is of codimension 1, which means that it is always one dimension lower than the corresponding scalar function. For example a 2D scalar function as the circle will have a 1D level-set (a contour line) and a 3D scalar function will have a 2D level-set function (the bounding surface).

1.2 Quadrics

A quadric surface is a surface defined by a quadric implicit function. It is defined as follows:

$$\begin{aligned} f(x,y,z) &= Ax2 + 2Bxy + 2Cxz \\ &+ 2Dx + Ey2 + 2Fyz \\ &+ 2Gy + Hz2 + 2Iz \\ &+ J \end{aligned} \tag{3}$$

By writing equation 3 in matrix form (equation 4), using different coefficients (A-J), a total of 17 different surface types can be described [1] (see equation 6 - 11).

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

The gradient of $\nabla f(x,y,z)$ can be found when a quadric surface is known.

$$\nabla f(x,y,z) = 2 \begin{bmatrix} A & B & C & D \\ B & E & F & G \\ C & F & H & I \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{5}$$

1.3 Quadric surfaces

1.3.1 Planes

$$f(x,y,z) = ax + by + cz = 0$$

$$\mathbf{Q} = \begin{bmatrix} 0 & 0 & 0 & a/2 \\ 0 & 0 & 0 & b/2 \\ 0 & 0 & 0 & c/2 \\ a/2 & b/2 & c/2 & 0 \end{bmatrix} \tag{6}$$

1.3.2 Cylinders

$$f(x,y,z) = x^2 + y^2 - 1 = 0$$

$$\mathbf{Q} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix} \tag{7}$$

1.3.3 Spheres and ellipsoids

$$f(x,y,z) = x^2 + y^2 + z^2 - 1 = 0$$
$$f(x,y,z) = \frac{x^2}{a^2} + \frac{y^2}{b^2} + \frac{z^2}{c^2} - 1 = 0$$

$$\mathbf{Q} = \begin{bmatrix} 1/a^2 & 0 & 0 & 0 \\ 0 & 1/b^2 & 0 & 0 \\ 0 & 0 & 1/c^2 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix} \tag{8}$$

1.3.4 Cones

$$f(x,y,z) = x^2 + y^2 - z^2 = 0$$

$$\mathbf{Q} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{9}$$

1.3.5 Paraboloids

$$f(x,y,z) = x^2 \pm y^2 - z = 0$$

$$\mathbf{Q} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \pm 1 & 0 & 0 \\ 0 & 0 & 0 & -1/2 \\ 0 & 0 & -1/2 & 0 \end{bmatrix} \tag{10}$$

1.3.6 Hyperboloids

$$f(x,y,z) = x^2 + y^2 - z^2 \pm 1 = 0$$

$$\mathbf{Q} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & \pm 1 \end{bmatrix} \tag{11}$$

1.4 Constructive solid geometry (CSG)

CSG is a technique used to model implicit surfaces in computer graphics. The simplest geometries (*primitives*, see section 1.3) are used as building blocks. Boolean operations are then used to combine surfaces.

$$\textbf{Union}(A,B) = A \cup B = \min(A,B) \tag{12a}$$
$$\textbf{Intersection}(A,B) = A \cap B = \max(A,B) \tag{12b}$$
$$\textbf{Difference}(A,B) = A - B = \max(A,-B) \tag{12c}$$

The operations in equation 12 can be used to model complex shapes from simple primitives.

Examples of the boolean operations can be seen in figure 1.

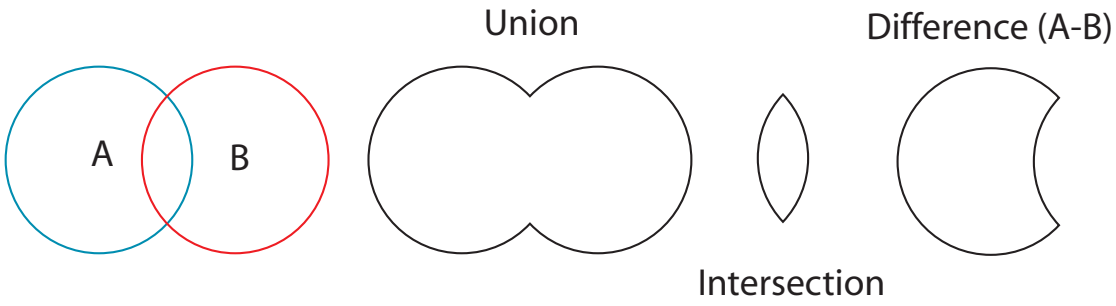


Figure 1: Boolean operations in two dimensions.

1.4.1 Blending operations

The problem with the simple boolean operators in equation 12 is that they only produce C^0 continuity along the surface intersection. This means that there will be unwanted sharp edges in the newly created objects. To address this problem, one way is to approximate the boolean operations to achieve smooth transitions around the intersections.

This can be done with density functions. Density functions describe the "density" of an implicit surface as

$$D_A(\mathbf{x}) = \begin{cases} > 1 & \text{if } \mathbf{x} \text{ is inside the surface} \\ = 1 & \text{if } \mathbf{x} \text{ is on the surface} \\ \in [0,1) & \text{if } \mathbf{x} \text{ is outside the surface} \end{cases} \tag{13}$$

To go from an implicit surface description to a density function equation 14 is used.

$$D_A(\mathbf{x}) = e^{-A(\mathbf{x})} \tag{14}$$

Super-elliptic blending is then performed as

$$D_{A\cup B} = \left(D_A^p + D_B^p\right)^{1/p} \tag{15a}$$

$$D_{A\cap B} = \left(D_A^{-p} + D_B^{-p}\right)^{-1/p} \tag{15b}$$

In equation 15 p is a form of sensitivity factor that represents the amount of smoothing to be done and as $p \to \infty$ the same result as with the pure boolean operations is obtained.

To then go back to an implicit surface from this it is necessary to apply the inverse transform of equation 14.

$$A(\mathbf{x}) = -\log D_a(\mathbf{x}) \tag{16}$$

1.5 The discrete gradient operator

By approximating the differential operator using the central difference approximation (shown in equation 17), the discrete gradient operator is approximated. This gives the discrete differential along the *x-axis*. To get the total gradient $\nabla f(x, y, z)$ for any given point i 3D space, the same equation is applied along the *y-* and *z-axis*.

$$D_x \approx \frac{f(x_0 + \epsilon) - f(x_0 - \epsilon)}{2\epsilon} \tag{17}$$

This can also be generalized to estimate higher order derivatives. The second order partial derivatives can be calculated as

$$\frac{\partial^2 f}{\partial x^2} = D_{xx} \approx \frac{f(x_0 + \epsilon) - 2f(x_0) + f(x_0 - \epsilon)}{\epsilon^2} \tag{18}$$

The approximations in 18 can then be used to approximate the mean curvature κ according to equation 19.

$$\kappa \approx \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} + \frac{\partial^2 f}{\partial z^2} \tag{19}$$

1.5.1 The normal

The gradient $\nabla f(x,y,z)$ is perpendicular to the iso-surface of f and points in the direction of increasing f. The gradient of a point on the interface is a vector which points in the same direction as the local normal.

$$\vec{n} = \frac{\nabla f(x,y,z)}{|\nabla f(x,y,z)|} \tag{20}$$

Equation 20 shows that the normal is equal to the normalized gradient.

2 Method

2.1 Implementation of CSG operators

The CSG operators in equation 12 was implemented in the file `CSG.h` in the `GetValue`-functions in the classes `Union`, `Intersection` and `Difference`. When implementing equation 12 it is important to note that the input to the `GetValue`-functions is in world coordinates and before evaluating, the coordinates has to be transformed to object space. This was done using the function `Implicit::TransformW2O`.

2.2 Implementation of quadric surfaces

To implement quadric surfaces, the matrices in equation 6 to 11 was implemented in the file `FrameMain.h`.

The evaluation of an implicit surface for an arbitrary point in space was implemented in `Quadric::GetValue`. The implementation is a matter of transforming the world coordinates to object space and then multiply it with the quadric matrix according to equation 4.

Calculation of the gradient for a quadric surface can be done according to equation 5. This was implemented in `Quadric::GetGradient`. Once again, the coordinates supplied to the function is expressed in world coordinates and has to be converted into object space before evaluation.

2.3 Implementation of the discrete gradient operator

Calculation of the gradient according to equation 5 only works for quadric surfaces and to calculate the gradient for arbitrary implicit surfaces, partial derivatives has to be used. These partial derivatives can be approximated with a central difference approximation as seen in equation 17. This is evaluated along each axis to construct the gradient as

$$\nabla f(x,y,z) = (D_x, D_y, D_z) \tag{21}$$

This was implemented in the function `GetGradient` in `Implicit.cpp`.

2.4 Computing the curvature

The mean curvature of an implicit surface is described in equation 19. To approximate the second order partial derivatives, equation 18 is used. This was implemented in the function `GetCurvature` in `Implicit.cpp`. Both this evaluation and the gradient evaluation are performed with world coordinates.

2.5 Implementation of super-elliptic blending

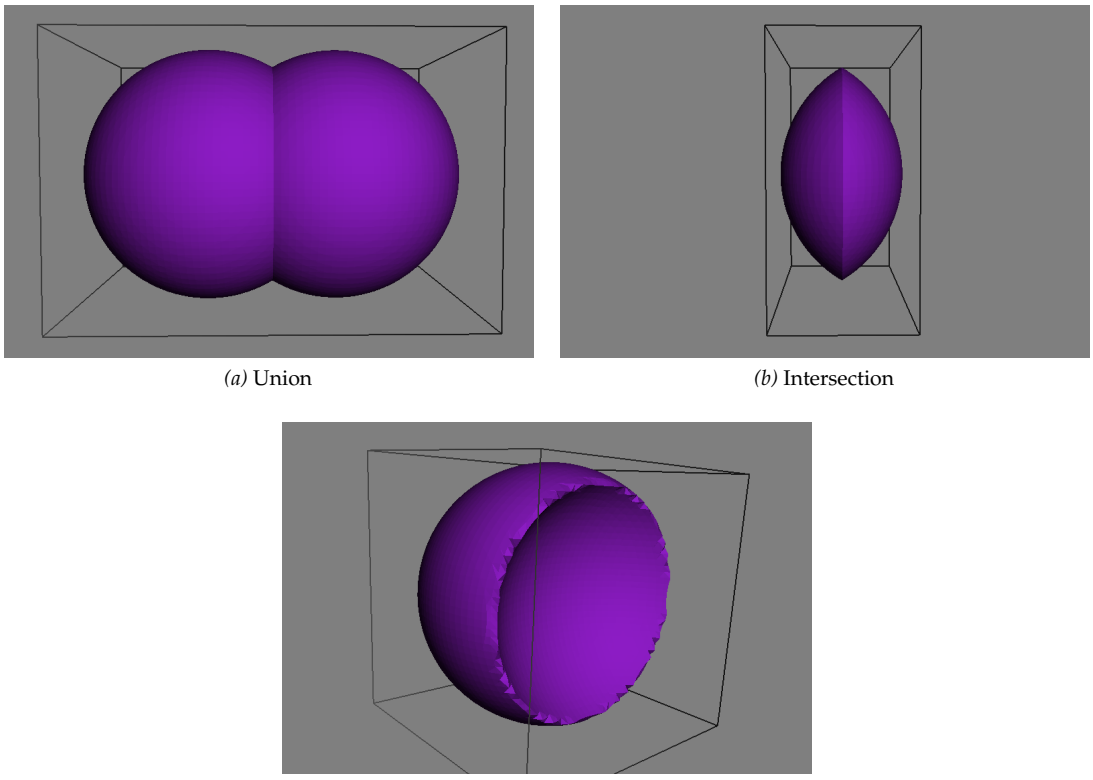
As described in section 1.4.1, the simple boolean operations only give C^0 continuity. To improve this, super-elliptic blending was implemented in the classes `BlendedUnion`, `BlendedIntersection` and `BlendedDifference` in `CSG.h`. When the blending operation is complete, the density function has to be transformed back to a surface. This is done by applying equation 16.

It is furthermore important to remember to transform the coordinates supplied as in-parameters from world-space to object-space.

3 Results

3.1 Implementation of CSG operators

The implementation of the operators work well. In figure 2 three examples can be seen where boolean operations have been carried out on two implicit spheres. It is possible to note the sharp edges that the raw boolean operations produce when no blending is involved.



(a) Union

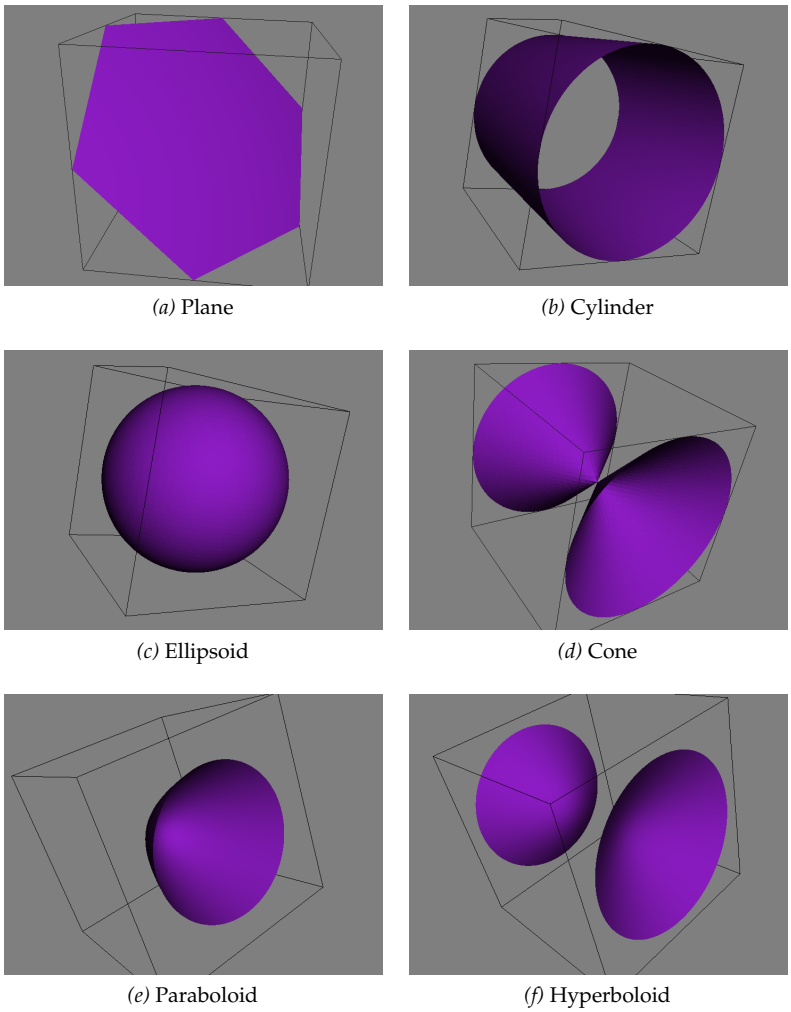
(b) Intersection

(c) Difference

Figure 2: Boolean operations on two implicit spheres.

3.2 Implementation of quadric surfaces

Quadric surfaces was successfully implemented and the six surfaces implemented are shown in figure 3.



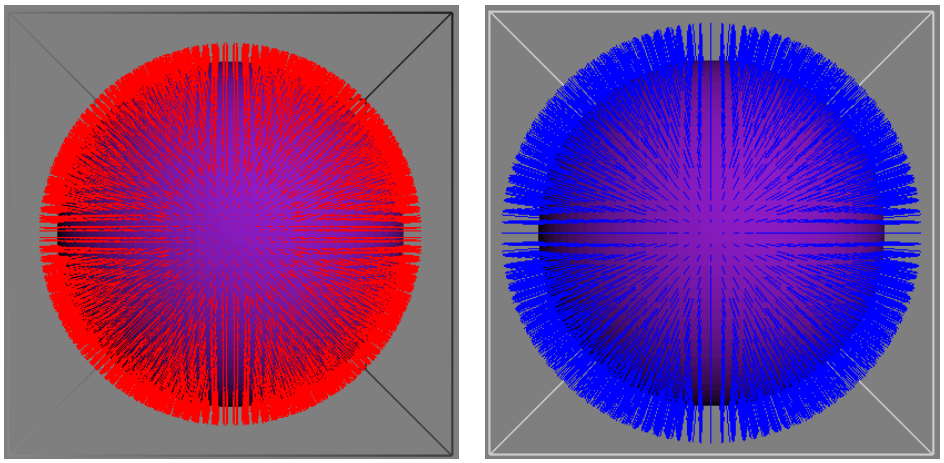
(a) Plane *(b)* Cylinder

(c) Ellipsoid *(d)* Cone

(e) Paraboloid *(f)* Hyperboloid

Figure 3: The six implemented type of quadric surfaces.

The gradient calculations for quadrics was also implemented in `GetGradient()` in the file `Quadric.cpp` and the results are shown in figure 4

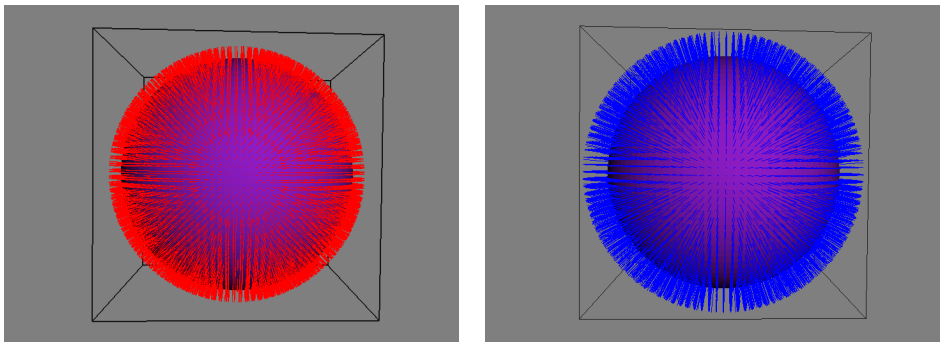


(a) Normals *(b)* Gradients

Figure 4: Normals (left) and Gradients (right) for the quadric sphere. Note that the direction is visually identical which is expected.

3.3 Implementation of the discrete gradient operator

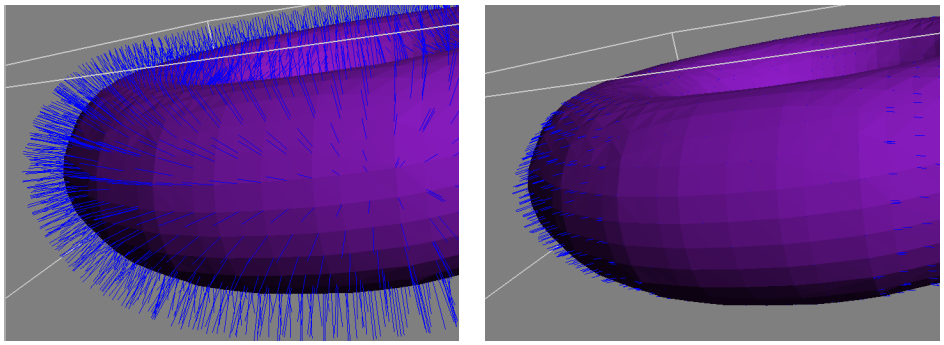
The discrete gradient operator works well and the results are shown in figure 5, where the normals of the mesh are also shown for comparison. It can be seen that the normals and the gradients are at least visually equal in terms of direction.



(a) Normals *(b)* Gradients

Figure 5: Normals (left) and Gradients (right) for the implicit sphere. Note that the direction is visually identical which is expected.

It is also possible to vary the differential scale (ϵ in equation 17) and it can be seen in figure 6 that when ϵ is increased, the gradients decrease in magnitude. This is since the central difference approximation in equation 17 is divided by an increasingly large number. The result can also be due to the fact that when ϵ increases, the evaluation of the implicit surface "misses" the surface and therefore the gradient goes towards zero.



(a) Gradients with a small differential scale

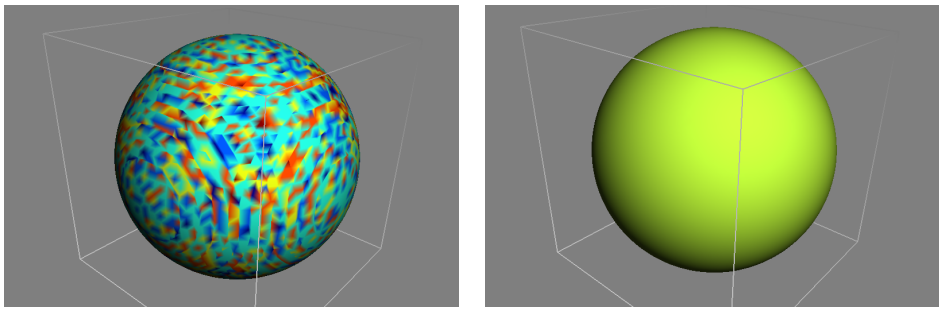
(b) Gradients with a large differential scale

Figure 6: Gradients for an implicit torus. Note that when increasing the differential scale, the magnitude of the approximation gets smaller and thus also the magnitude of the gradients.

3.4 Computing the curvature

The curvature operation is a rather crude approximation of mean curvature which can be seen when numerically evaluating the curvature values for the unit sphere. A unit sphere has a

constant curvature of 1. For this approximation the sphere get a curvature of approx. 6. It is also important to manually set the range of color mapping when using colormaps on objects with constant curvature. This is since the small fluctuations in curvature will be mapped over the entire color map with seemingly strange results as shown in figure 7.



(a) Automatic color mapping of curvature

(b) Manual mapping of curvature.

Figure 7: Automatic (left) versus manual (right $[-50, 50]$) mapping of curvature color. It can be seen that when there is small differences in curvature, the range is mapped over the whole color map which gives results as in the left image.

The colormaps *jet* and *HSV* are shown in figure 8.

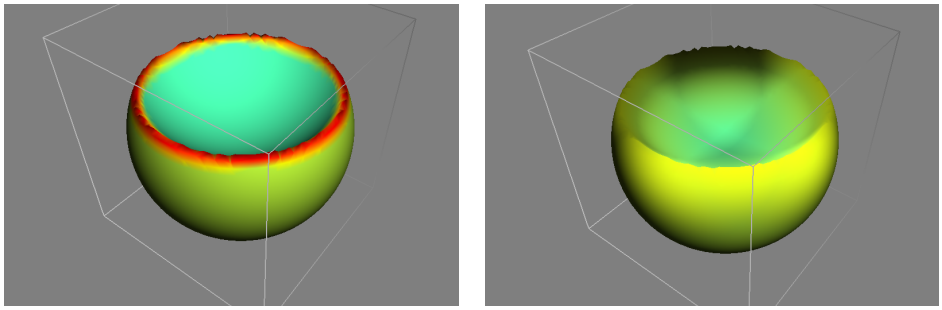
(a) Jet colormap

(b) HSV colormap

Figure 8: Jet and HSV colormap.

A more interesting example of the mean curvature is shown in figure 9, where a difference between two implicit spheres is evaluated in terms of curvature. It can be seen that there are convex values on the inside of the sphere and concave values on the outer shell.

If the value of ϵ becomes higher, the mean curvature becomes less sensitive to high frequency changes in the mesh. This is due to the fact that the mean curvature approximation is done on points that are separated in space by a distance depending on ϵ. This can be seen in the right part of figure 9, where the value of ϵ is increased, noting that the red border disappears.



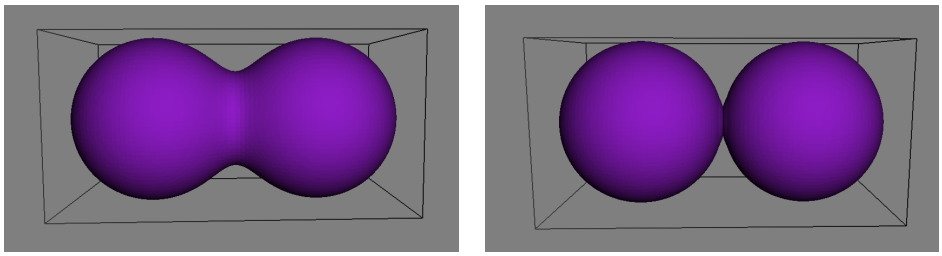
(a) Curvature with a small differential scale

(b) Gradients with a large differential scale

Figure 9: Curvature of the implicit sphere differenced with another sphere. Note that when increasing the differential scale, the high frequency changes in curvature (red in left figure) disappears.

3.5 Implementation of super-elliptic blending

Super-elliptic blending works well and results of blending between two implicit spheres can be seen in figure 10. It can be seen that a lower p-value gives a smoother surface intersection between the two spheres.



(a) Blended union with $p = 2$ *(b)* Blended union with $p = 10$

Figure 10: Super-elliptic blending union between two implicit spheres for two different values of p. It can be seen that as p increases, the result gets closer to the raw boolean operation (which would result in no common surface between the two spheres).

4 Conclusion

Implicit meshes can be less intuitive than explicit ones to render. Separate calculations (Ray-casting, Marching cubes etc.) for discretizing the surface has to be done to be able to render an implicit surface.

Albeit this, implicit surfaces are very interesting since easy and intuitive boolean operations can create complex shapes. The robustness of the implicit surfaces make them well suited for animations and simulations of different kinds.

It is also impressive to get that many different primitives out of one quadric matrix and that demonstrates the power and simplicity of implicit surfaces.

Lab partner and grade

I did this lab together with Nathalie Ek (natek725). All tasks marked with * was completed, as well as all tasks marked with **, which should qualify me for grade 5.

References

[1] Gunnar Läthén, Ola Nilsson, and Andreas Söderström. Implicit surfaces and modeling. Technical report, ITN, 2011.