Prof. Dr. Hans-Peter Seidel
Dr. Michael Wand
Nils Hasler, Jens Kerber, Thomas Schultz,
Carsten Stoll, Martin Sunkel

Geometric Modeling

Assignment sheet 10 (Blobbies, due July 11th 2008)

Some of the material used in this exercise was not covered in the lecture yet. There is yet again a new software package you may want to use. It adds per-vertex-lighting and a marching cubes module.

(1) Blobby Objects [1+2+4 points]

Blobbies are implicitly defined objects. Traditionally, the function

$$C(r) = -\tfrac{4}{9} r^6 / R^6 + \tfrac{17}{9} r^4 / R^4 - \tfrac{22}{9} r^2 / R^2 + 1$$

is used to describe metaballs/blobbies. In contrast to exponential decay functions this function has the property that it reaches 0 at R and has the value 1/2 at R/2. If several of these functions are placed in space with overlapping tails and an implicit surface is drawn, organically looking shapes arise.

a. Blobbies are defined by a centre and a radius. Allow adding points to a 3D-world that act as centres for Blobbies.

b. Add a second point to every Blobby that acts as a handle for resizing the Blobby.

c. Compute the implicit Blobby function on a 3D-grid for several Blobbies and use the marching cubes module to triangulate the implicit surface. The user should be able to set the surface threshold and the grid resolution. Recompute when radius or centre changes.

(2) Normals [5+1 points]

a. Compute surface normals for the surface vertices and display them as small lines sticking out of the surface.

b. Set the normals of the displayed vertices to achieve smoother shading.

(3) Curvature [3+2+2 points]

Add buttons to compute the

a. Principal,

b. Secondary,

c. Mean curvature of the surface.

Use an appropriately scaled color gradient to display the different values.

Software Package Remarks:

The new package allows Per-Vertex-Lighting and provides a marching cubes implementation!

A user can swap between face-lighting and vertex-lighting by the button "**Light F/V"**

A **Point3D** has a normal now as well, this is required for correct light computation.

If it is not set then everything will be black or white.

Please inspect the **ExampleExperimentMarchingCubes**.

The first function shows a cube with different colors for every cube.

Play around with the **Handles** and **Light F/V** button.

The normals do not automatically recompute, so if you pull a corner then the lighting will NOT change accordingly.

The second function shows you how a cube is triangulated using the marching cubes function.

The vertices are filled with a random value in [0..1], so for every new function call you get a new triangulation.