Geometric Modeling
Assignment sheet #12
"Marching Cubes/Squares"
(due July 25th/July 27th 2012 during the interviews)

Silke Jansen, Ruxandra Lasowski,
Art Tevs, Michael Wand

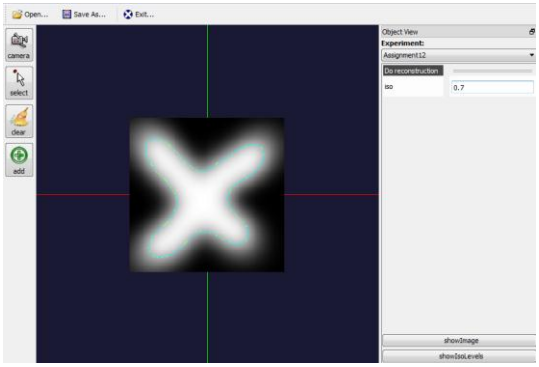
---

(1) 2D Isosurface reconstruction [5 + 4 + (2 Bonus) points]

In this exercise you will write a routine to reconstruct the isosurface of a given implicit function.

a. Create a new experiment and put a button to load an image *blob.bmp* from the disk. You can interpret the color channel 0 of the loaded points as implicit function value. Add a button which reconstruct an isosurface for a given threshold value τ using 2D version of marching cubes algorithm, also known as marching squares. The number of used cells should be equal to the number of pixels in the image. Work with squared images only. *Hint: It is ok to split every edge of the cell in the middle.*

b. Same as (a) however perform now a correct linear interpolation for the computed contours within the cell. You should split the cell's edge not in the middle but depending on the implicit function values of cell's vertices and τ.



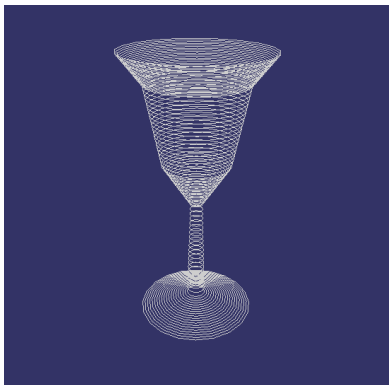
Extracted isoline for τ=0.7

c. Same as (a), however let user specify the reconstruction resolution. Hence the user should be able to set the amount of cells in x and y directions used for marching squares approach. Use linear interpolation when sampling the implicit function in each cell.

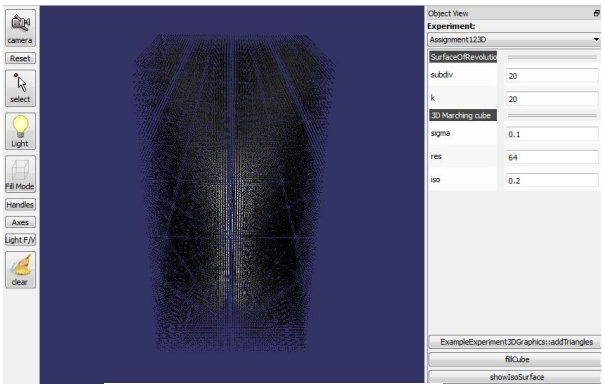
(2) 3D Isosurface reconstruction [5 + 4 Points]

In this exercise we will write a method to reconstruct the isosurface of a 3D implicit function, i.e. volume. As a source for our 3D volume data, we will use the surface of revolution from the last practical assignment sheet. You can use the *sor.cpp* file on the webpage which provides a method to generate fixed surface of revolution in three dimensions. A three dimensional volume containing the surface of revolution can be created by splatting a Gaussian function centralized on each point. This will create some kind of a blurry volumetric representation of the object.

a. Create a new 3D experiment. Add a button which splats a Gaussian function into the 3D volume for every point of the surface of revolution. Let user specify the size of the volume ***d*** and the ***σ*** of the Gaussian function. Assume that resolution and sigma are equal in each dimension.
Hint: You can debug the volume by rendering a small point/line in each volume cell colored by the Gaussian distribution.
Hint: The performance of the splatting is in worst case cubic, so be patient.

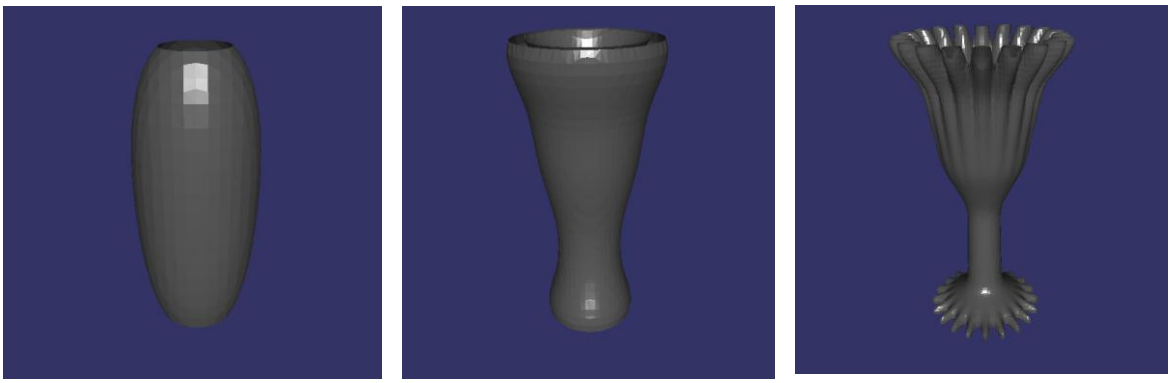


Surface of revolution found in sor.cpp



Visualized volume of the surface

b. Add a button which performs the isosurface reconstruction. For this use the method `MarchingCubes3D::triangulate(…)`, found in `math/MarchingCubes3D.h`. Add reconstructed triangles to the viewer. Experiment with different sigma and resolution levels.
Hint: Take a look into `ExampleExperimentMarchingCubes` *to see how the method is used.*



(left) σ =0.5, d=16, iso=0.7 (center) σ =0.1, d=32, iso=0.15, (right) σ =0.035, d=64,