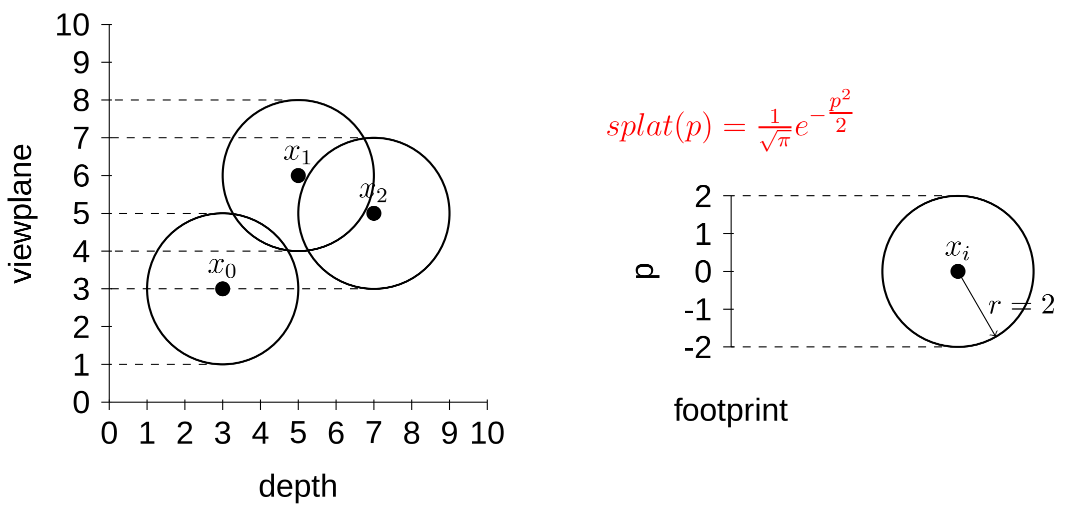
Dr. Filip Sadlo *Stuttgart, 14. 6. 2013*

Visualization (Assignment 8)

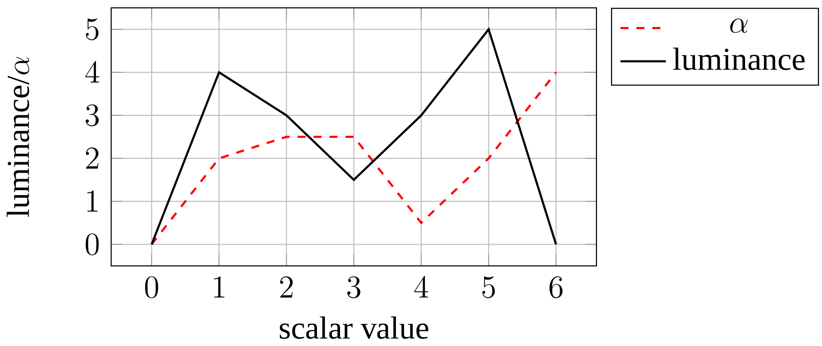
Exercise 8. 1 [8 Points] Splatting

The concept of volume splatting can be considered as throwing balls filled with ink onto a canvas. Instead of accumulating color/density values of the volume data along a ray for each pixel, we project the data in terms of interpolation kernels directly on the viewplane and accumulate the color/opacity for each pixel. The interpolation kernel $w(x, y, z)$ is projected from a 3D to a 2D representation resulting in a "footprint": $splat(x, y) = \int w(x, y, z)\mathrm{d}z$. Each footprint is weighted by the scalar value of the respective sample point and its contribution to color/opacity is accumulated on the viewplane. For simplicity the radial kernels are illustrated as circles and the viewplane is the y-axis with orthographic projection in this exercise.



1. Evaluate the values for the 4 discrete pixels of the footprint by using the given projected Gaussian splat function $splat(p) = 1/\sqrt{\pi}\ e^{-p^2/2}$ at the center of each pixel $(splat(-1.5), splat(-0.5), \ldots)$ as shown in the right figure.

2. Weight the pixels of the footprint for all volume sample points in the left figure using their corresponding values: $f(x_0) = 3, f(x_1) = 6, f(x_2) = 1$. Evaluate the α and luminance values using the following transfer function:



The solid line defines the transfer function for the luminance, the dashed line the transfer function for the opacity. All line segment end point coordinates are integer multiples of half a unit.

3. Accumulate the luminance values on the viewplane in back-to-front order using the calculated α-values and the back-to-front compositing formula from the lecture.

Exercise 8. 2 [4 Points] Function Color Mapping with OpenGL

Mapping values of a given function to a specific color can be achieved by using a fragment shader. Therefore, you can download the *OGL4Core.zip* package from the homepage. In the folder *OGL4Core/Plugins/ScalarFieldColoring* you can find a complete Visual Studio 2012 project with all the code necessary to run an application with OpenGL shaders. Your task is to edit the fragment shader *OGL4Core/Plugins/ScalarFieldColoring/resources/simpleShaderfragment.glsl*. Implement the following functionalities:

- Mapping of 2D texture coordinates to a scalar value [0,1], using the function: $f(\alpha,\beta,\gamma,\phi_1,\phi_2) = abs((\sin(\alpha)+1.0)*\sin(\beta*\phi_1+\cos(\alpha))*\cos(\gamma*\phi_2+\sin(\alpha)))$ with:
  α = value of the timer variable
  β = x value of the resolution
  γ = y value of the resolution
  ϕ_1 = x value of the texture coordinate
  ϕ_2 = y value of the texture coordinate

- Mapping of the scalar value f to a color (perform a linear interpolation in between):
  $f = 0$: blue
  $f = 0.5$: white
  $f = 1.0$: red

A single frame of the resulting animated texture is depicted in Figure 1. The zip file also contains a *readme.txt* that describes how to create a new plugin.

Submission: 21.06.2013, 10:00

please hand in your submission in the eClaus system.

.

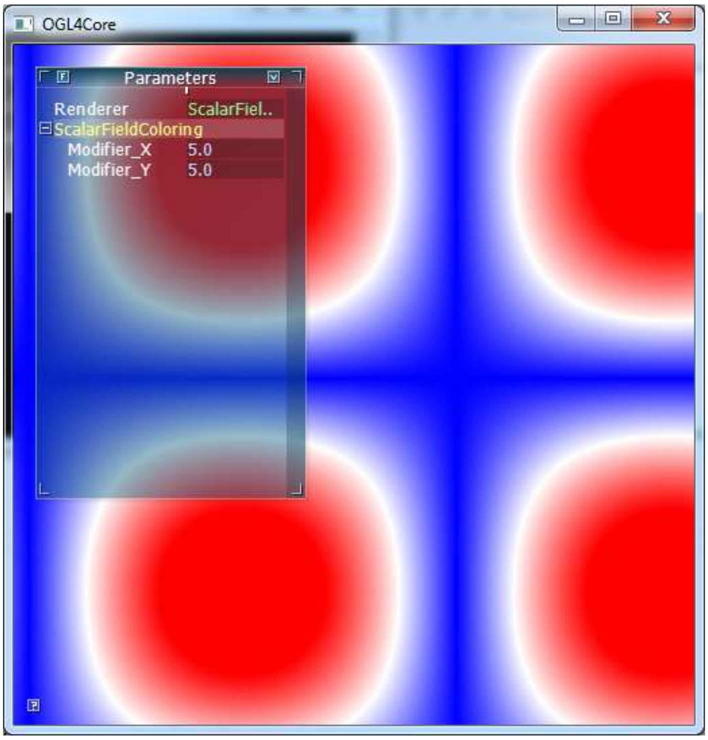


Figure 1: Function color mapping.