

UNIVERSITÄT DES SAARLANDES
DR.-ING. HENDRIK P.A. LENSCH
MAX PLANCK INSTITUT INFORMATIK
ART TEVS (TEVS@MPI-INF.MPG.DE)
BORIS AJDIN (BAJDIN@MPI-INF.MPG.DE)
MATTHIAS HULLIN (HULLIN@MPI-INF.MPG.DE)



10. JANUARY 2008

COMPUTER GRAPHICS I ASSIGNMENT 9

Submission deadline for the exercises: Thursday, 17th January 2008

Rule: Solutions have to be submitted in the lecture room before the lecture.

9.1 Clipping (40 Points)

Given an axis aligned clipping window spanned by the two points $(0, 0)$ and $(12, 6)$

$$C_1 = (0, 0), C_2 = (0, 6), C_3 = (12, 6), C_4 = (12, 0)$$

and a polygon with the lines $P_1 - P_2 - P_3 - P_4 - P_1$ with

$$P_1 = (9, 8), P_2 = (-6, 3), P_3 = (14, 20), P_4 = (15, -4).$$

Use the Sutherland-Hodgeman algorithm to perform the first step of the algorithm by clipping the polygon against the top edge of the clipping window only. Describe each step of the algorithm. You do not have to compute the intersection point of the lines with the clipping window, give them names in a figure and use that names instead.

9.2 Rasterization (10 Points)

Why does mipmapping speed up texturing in rasterization hardware? Look at the memory access patterns and especially the cache.

9.3 Triangle Order for Rasterization (10 + 10 Points)

Assume you are rasterizing some triangles in different orders.

- a) Is the depth buffer the same for each order of the triangles after the computation (depth test is enabled)?
- b) Is the color buffer the same for each order of the triangles after the computation (with and without depth buffer)?

9.4 Vertex and Pixel shading (15 + 15 + 20 Bonuspoints for c)

In this exercise you will write an OpenGL application using CG vertex- and pixel shaders. To start, download the new framework from <http://www.mpi-inf.mpg.de/departments/d4/teaching/ws2007-2008/cg/skeleton.tgz>. Unzip it and read through the code to understand what is roughly happening. The result of this exercise should be a waving water surface (animated with a vertex program) with a reflection of an environment map.

a) In this exercise we will implement a simple water wave simulation using vertex-programs. Vertex-programs transform the vertices of the primitives prior to rasterization. All you have to do here is to change the world z-coordinate of the incoming vertex position. To do so edit `water_vert.fx` and add the appropriate parameter bindings to `water_sim.cpp`. A simple way to simulate water behaviour is to combine *sin* functions.

b) Once your waves are moving, you have to calculate the proper normals. To calculate tangents to a surface you can use the numerical derivative approximation given by $f'(x) \approx \frac{f(x+d)-f(x)}{|d|}$, x and $d \in \mathbb{R}^2$. This formula gives you the derivative of f in direction d . The length of d determines the approximation error, so smaller lengths give better approximations, but be aware of numerical errors.

Hint: You can visualize the calculated normals by setting the fragment color to the normal value in the fragment program.

c) Now, you have to implement the water-surface reflection with the environment map. To do so, edit the fragment program `water_frag.fx` and add the proper bindings to `water_sim.cpp`. Calculate the reflection vector from the eyepoint to the shaded point. Use the reflected vector to look up the environment map.

In order to calculate the world eyepoint coordinates you need to multiply the point $(0, 0, 0, 1)$ with the inverse model-view transformation. (see *OpenGL documentation for glGet()*). You can pass the calculated point to the fragment shader as a uniform parameter. You also have to bind the cube-environment map texture (look in the method `drawSky()`).

Hint: Disable the wave generation (both height and normal) and get the reflections working for the flat surface first. The normal should be $(0, 1, 0)$ here.

To make things work you will need a graphics-board that supports CG-shaders like everything above the nVidia 5200 FX. The CIP pool (105) has been tested with our solution and everything works fine.