

Assignment #1

Group 1 presents solutions at 9/11

Group 4 prepares questions/discussion

In total: 20 minutes

Running Cg programs at Daimi

- Windows machines with ATI Radeon 9800 graphics cards
 - Zuse 127 – all
 - Visual studio .net 2003 on first row – more to follow...
 - Zuse 114 – nussbaum, hyltdgaard, dickow

Compiling Cg programs at Daimi

- Install Cg (<http://developer.nvidia.com/cg>)
- Install GLUT. Put in eg C:\GLUT\
(<http://www.xmission.com/~nate/glut.html>)
 - Put glut.h in a GL subdirectory
- Sample applications come with Visual Studio .net 2003 solutions. On machines with Visual Studio 6.0:
 - Locate a sample .dsp file from the Cg installation directory. Eg C:\Program Files\NVIDIA Corporation\Cg\examples\runtime_ogl_vertex_fragment\ogl_example.dsp. Copy directory and replace project .cpp and .cg files.

Setting up VC++ paths and Windows environment variables

- In Visual Studio .net 2003 under Tools->options->projects->VC++ Directories:
 - Add include paths to <...>\Cg\include and C:\GLUT\ to the VC++ include paths
 - Add lib paths to <...>\Cg\lib and C:\GLUT to the VC++ lib paths
- From the Windows control panel:
 - Open the system controls. Under the advanced tab, press the "environment variables" button and add <...>\Cg\bin and C:\GLUT to the PATH environment variable. Restart Visual Studio.

Part A

- **Compile and run the sample programs from the lectures**

- Available from the course home page
- Do you encounter any problems?**
 - If so, please report (and solve).

Part B

- **Specular lighting**

Add a specular lighting term to the sample applications

- Chapter 6 of the "Redbook" at <http://rush3d.com/reference/opengl-redbook-1.1/> defines the specular lighting component:

Specular lighting mathematics

Specular Term

The specular term also depends on whether light falls directly on the vertex. If $\mathbf{L} \cdot \mathbf{n}$ is less than or equal to zero, there is no specular component at the vertex. (If it's less than zero, the light is on the wrong side of the surface.) If there's a specular component, it depends on the following:

- The unit normal vector at the vertex (n_x, n_y, n_z).
- The sum of the two unit vectors that point between (1) the vertex and the light position (or light direction) and (2) the vertex and the viewpoint (assuming that `GL_LIGHT_MODEL_LOCAL_VIEWER` is true; if it's not true, the vector (0, 0, 1) is used as the second vector in the sum). This vector sum is normalized (by dividing each component by the magnitude of the vector) to yield $\mathbf{s} = (s_x, s_y, s_z)$.
- The specular exponent (`GL_SHININESS`).
- The specular color of the light (`GL_SPECULARlight`).
- The specular property of the material (`GL_SPECULARmaterial`).

Using these definitions, here's how OpenGL calculates the specular term:

$(\max\{\mathbf{s} \cdot \mathbf{n}, 0\})^{\text{shininess}} * \text{specularlight} * \text{specularmaterial}$

However, if $\mathbf{L} \cdot \mathbf{n} = 0$, the specular term is 0.

Part C

- The bump mapping sample application uses normal maps with normals pointing mainly along the positive z axis.
- The rendered quad hence lies in the x-y plane when lighting is calculated in eye-space.
- Alternatively, lighting could be calculated in *tangent-space* - <http://www.opengl.org/resources/tutorials/advanced/advanced97/notes/node73.html>
- **Implement bump mapping in tangent space by rotating the light into tangent space. Use steps 1-4 from link above. Use a single normal texture to map some 3D geometry of your own choice.**



Optional

- **Look at parallax mapping**

- Find inspiration at

- http://www.infiscape.com/doc/parallax_mapping.pdf