Interactive Computer Graphics
Coursework: Rendering

Important

- The Computer Graphics coursework MUST be submitted electronically via CATE. For the deadline of the coursework see CATE. The files you need to submit are described later in this document.
- Before starting the assignment please make sure you have read the description of the environment and data formats below.

Make sure that you give yourself enough time to do the coursework by starting it well in advance of the deadline. If you have questions about the coursework or need any clarifications then you should come to the tutorials!

Coursework

This section describes the main aim of this coursework task. *Before you start* the task, however, make sure you have read and understood the sections on the environment and data formats at the end of this document. This coursework exercise is a practical programming exercise which should be done using OpenGL. The goal of the coursework is to produce a realistic looking 3D rendering of a face.

You are provided with the following:

- This coursework description.
- A skeleton program in the form of a pair of source files (.cpp and .h).
- A Makefile for building the program in a Linux environment.
- A data file containing the surface model of the face, i.e. the geometric and texture coordinates and the polygon information needed to render it. This file is in VTK format (see below).
- A data file containing the texture information that can be used to produce a realistic rendering of the face. This file is an image in PPM format (see below).

You will need to add *your own code* in order to read and parse the input data files and then to generate the renderings required. You can add new classes as part of an object oriented approach if you like. You will need to amend the Makefile if you decide to add your own additional header and/or source files. For further details about the data files for the model and texture, see below.

When you have completed your code, you will need to submit the following items:

1. A Gouraud rendering of the face that is similar to the rendering shown on the left in Figure 1. It just needs to be similar to the Figure, not exactly the same, i.e. the camera should be in roughly the same place and pointing roughly in the same direction. Capture a screenshot of the rendering in `PNG` format and make sure this file is named `gouraud-1.png` when uploading it on CATE.

2. A Gouraud rendering of the face with a different viewpoint from the previous one. Make sure this file is named `gouraud-2.png` when uploading it on CATE.

3. A Gouraud rendering with a different viewpoint from the previous two - you may also like to adjust the lighting or material properties. Make sure this file is named `gouraud-3.png` when uploading it on CATE.

4. A texture mapped rendering using the same viewpoint used in 3. Make sure this file is named `texture.png` when uploading it on CATE.

5. The source code that you used to produce the renderings. Make sure this file is named `code.tar.gz` or `code.zip` when uploading it on CATE.

Ensure that your source code submission is a *genuine* zip or tar file, do not use a different archiving tool (e.g. rar) and simply change the suffix. Please do not simply archive your whole working directory as this will include binaries and system files that are not required.

It is very important that you carefully follow the instructions on organising and naming the files and folders submitted. Failure to do so may mean that the work cannot be marked.

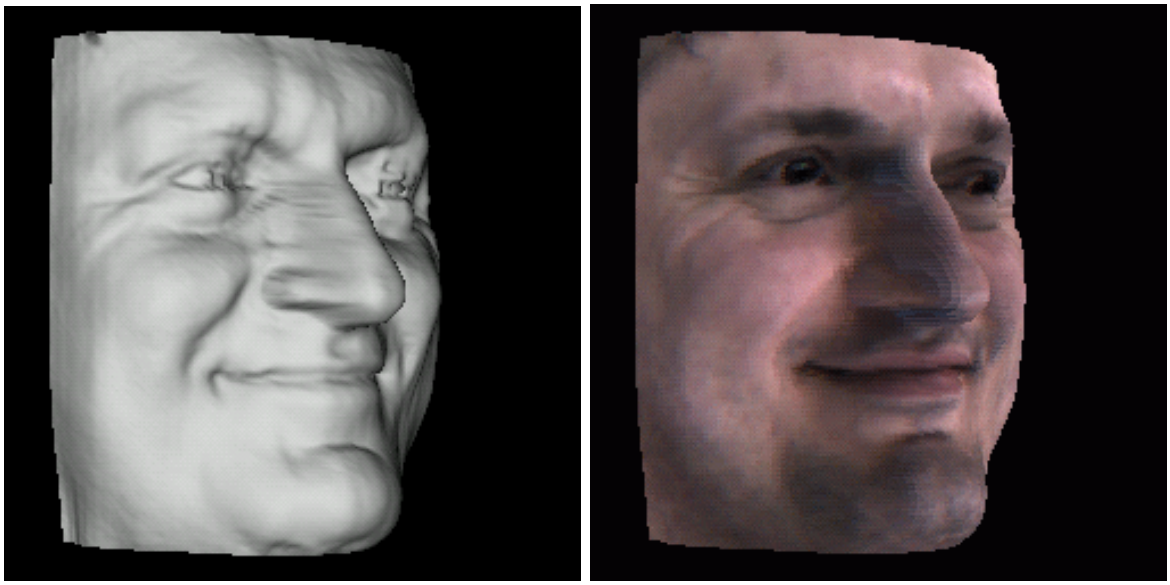

Figure 1: Example renderings of the data used in this coursework exercise. Left: Gouraud shading. Right: Texture mapping.

Things to look out for

Make sure that:

- you calculate average normals per vertex;
- your surface model 'well placed' in the scene;
- your surface model is well lit, e.g. not too dark or bright;
- specular reflections are well controlled;
- you have chosen a good colour scheme;
- your viewpoints show genuine variety;

- you have altered material properties, lighting and camera position;
- your code is well structured and efficient;
- your code is commented.

Environment

The programming environment for the exercises is OpenGL on the Linux/X11 machines of the department. OpenGL should be installed on all Linux/X11 machines. Apple Macs may also provide a suitable Linux environment. You may use a Windows machine but we recommend that you use the Linux environment. You can program either in C or C++.

In order to view ppm format image files on the Lab Linux machines 'Eye Of Gnome' can be used, it can be run by typing `eog` at the command line. Taking screenshots can be done in a variety of ways. For example the Gnu Image Manipulation Program (`gimp`) can be used. After starting gimp, go to File → Create (or Acquire) to obtain a screenshot.

The provided skeleton program

The skeleton OpenGL program that is provided for you is contained in the following files:

- cgRender.h
- cgRender.cpp
- A Linux Makefile

To set the program up, create a new directory and copy all three files into it. To compile the program, go to the new directory and type `make` at the command line. This should compile the sample program for you.

To run the sample program, type `./cgRender` at the command line. The skeleton OpenGL program should create an empty window for you. Take a good look at the skeleton program. It contains a number of hints on how to use OpenGL to start solving the coursework.

Data

The data provided consists of two files, one contains the surface model and the other contains the texture data.

Surface model file

The surface model of the face is stored in the VTK format[1] file `face.vtk` provided. This surface model consists of three components: Vertex coordinates, polygon data and texture coordinates.

You can assume that the surface model contains only simple and planar polygons. In addition, you can assume that the surface model is consistent, i.e. the ordering of vertices of the polygons is always the same.

The surface model file consists of the following four parts:

[1] www.vtk.org

1. Header

The header contains the following lines:

```
# vtk DataFile Version 3.0
Somebody's face
ASCII
DATASET POLYDATA
```

2.Vertex data

The first line contains the key word POINTS to indicate the start of the vertex data. The next number equals the number of vertices of the surface model, followed by the type of each vertex (you can assume that this always corresponds to float). Each of the following lines contains the x, y and z world-coordinates for each vertex as floating point numbers.

3. Polygon data

The first line contains the key word POLYGONS to indicate the start of the polygon data, followed by the number of polygons and the size of the cell list for all polygons. Each of the following lines contains the number of vertices forming the polygon, followed by the indices of the vertices which form the polygon. The indices of the vertices correspond to the order in which they are listed in the vertex data (see above).
You may want to remind yourself of how vertex and polygon data are stored by looking at the simple example given in Lecture 1.

4. Texture data

This is indicated by the keyword POINT_DATA, followed by the number of points with texture coordinates. The next line contains the key word TEXTURE_COORDINATES to indicate the start of the texture data and some miscellaneous information. The following lines contain the x and y texture coordinates for each vertex as floating point numbers. The order of the texture coordinates is the same as the order of the vertices

Texture file

The provided ppm format image file `face.ppm` contains the 512×512 texture map of the face. The texture map is represented as a PPM format file with a short ASCII header followed by the R (red), G (green) and B (blue) components for each pixel as a separate unsigned byte. To find out more about the typical header of a PPM file, type `man ppm` at the command line or see `http://netpbm.sourceforge.net/doc/ppm.html`.