CS468 - Geometry Processing Algorithms

Exercise 1 - Getting Started with OpenMesh

Handout date: 9/22/2010

Submission deadline: 10/06/2010, Midnight

Note

Copying of code (either from other students or from external sources) is strictly prohibited! Any violation of this rule will lead to expulsion from the class.

What to hand in

A .zip compressed file renamed to "Exercise*n*-YourStudentId.zip" where *n* is the number of the current exercise sheet. It should contain:

- A Microsoft Visual Studio 2008 file solution with all source files and project files. In order to avoid sending build files, close your Visual Studio solution and run the file "cleanSolution.bat" located in the top directory of the framework before you submit.
- A "readme.txt" file containing a description on how you solved each exercise (use the same numbers and titles) and the encountered problems.
- Send your solutions to ***mirela@stanford.edu*** before the submission deadline.
  No late submissions will be accepted.

This homework is 10% of your final grade.

Understanding the framework

The aim of the first exercise sheet is to familiarize yourself with the provided mesh processing framework and to implement a simple vertex valence visualization of a mesh using OpenMesh.

The framework is a simple C++ mesh tool which is provided as a MS Visual Studio 2008 project. The built binary can be launched from command line (via "CMD.EXE") and a mesh can be passed as a parameter. Once the application is started, the mesh will be processed and an OpenGL/GLUT based graphical user interface shows the resulting mesh.

The simple GUI allows you to navigate the loaded mesh with the following mouse controls:

- Left-click: rotate view
- Middle-click: mode view in *x* and *y*-axis
- Left-Middle-click: Zoom in and out

Right-clicking the GLUT window will give you access to a couple of rendering options as shown in Figure 1.

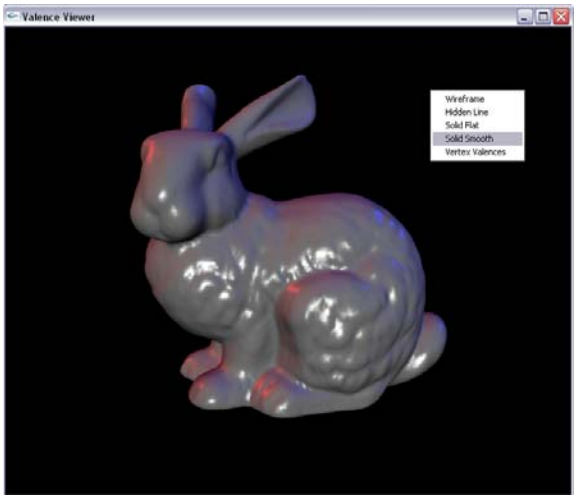

Figure 1: Screenshot of the mesh viewer. The "Solid Smooth" rendering mode is selected.

In this exercise, you will have access to the "Vertex Valences" menu item.

The following libraries are already provided with the project:

- OpenMesh v. 1.0
- OpenGL (Windows binaries)
- GLUT (Windows binaries)

Installing GLUT

The file “glut32.dll” is needed by the framework. You have two options:

1. Copy the file to the same location as the executable.
   Or
2. Change the PATH environment variable:
   - Right click on “Computer/My Computer”
   - Select “Properties”
   - Select “Advanced system settings” on the left (skip this step if running Windows XP)
   - Make sure you are in the “Advanced” tab
   - Click “Environment Variables”

- In the “System variables” section (on the bottom), click on “Path” and click “Edit”. ***Add*** the directory containing glut32.dll to the list. Make sure to separate all Path values with a semicolon.

Sample Software

See *01-ValenceViewer.exe* for an example of how your program should run.

1.1 Installation and getting started

- The solution file is found in "MSVC/MeshCourse.sln". Build the project in debug and release mode and load a mesh. The application is located in "MSVC/release/" and "MSVC/debug/".
- Familiarize yourself with the project file organization, the "GLUTViewer" class and its inherited classes "MeshViewer" and "ValenceViewer".
- Learn how to use the OpenMesh library (read the slides from the exercise session and the online tutorial).

1.2 Vertex valence of a triangle mesh

The valence $v(\mathbf{x}_i)$ of a vertex $\mathbf{x}_i$ in a triangle mesh is the number of vertices in the 1-ring neighborhood $\mathcal{N}(\mathbf{x}_i)$ of $\mathbf{x}_i$. In particular, the vertices of $\mathcal{N}(\mathbf{x}_i)$ are connected with an edge to $\mathbf{x}_i$.

The application entry point (main function) is located in "valenceview.cc". First, a ValenceViewer object is instantiated, then a mesh is loaded via the "open_mesh" function, and immediately the functions "calc_valences()" and "color_coding()" are called before the program enters the GLUT main loop to show the GUI window.

- Implement the "calc_valences()" function in "ValenceViewer.cc". It should compute the valences of all vertices of the mesh "mesh_" and store them in a custom attribute which you have to define for each vertex. "mesh_" is defined as a private member variable of the superclass "MeshViewer".

Feel free to modify the ValenceViewer class, but describe it accordingly in the "readme.txt" file.

1.3 Color visualization

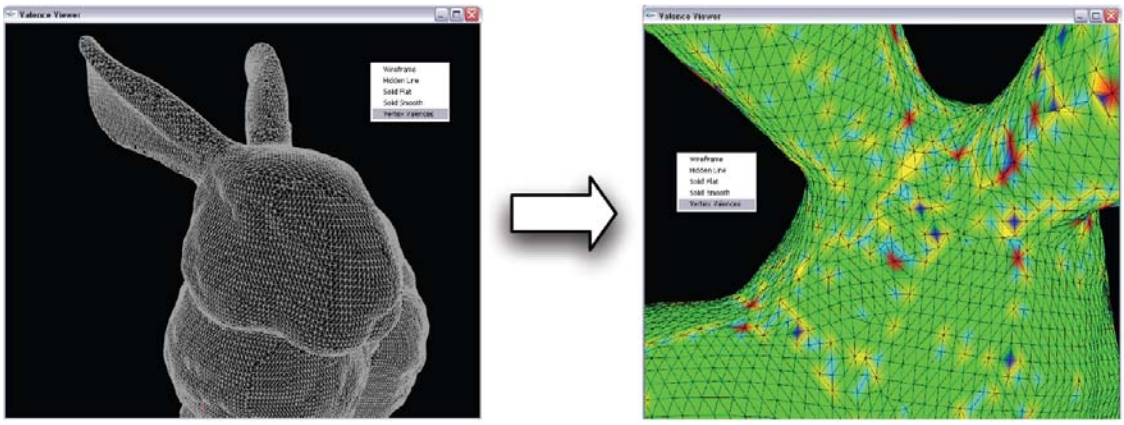

Figure 2: Left: No colors are set and only the wireframe of the mesh is visible. Right: Color-coded visualization of the vertex valence.

In order to evaluate the valences of the mesh, you will set a *meaningful* color of your choice for each vertex that represents the vertex valence. The color rendering is handled by the framework and can be accessed by right-clicking the GUI window and pushing the "Vertex Valences" menu item.

- Define a meaningful valence color-coding function $c : \mathbb{N} \rightarrow [0,255]^3$ and describe it in the "readme.txt" file
- Implement your colorization function in "color_coding()".

Vertex coloring should be done using the *predefined* custom attributes from OpenMesh in order to be rendered.