CS 523: Computer Graphics — Assignment 1: Mesh "Hello World"

Handout: Tuesday, January 25 — Due date: February 15, 2011

1 Implementation

This assignment is designed to get you acquainted with some basic mesh data structure programming, rendering a mesh, and basic UI programming. Note that subsequent assignments will rely on this foundation. Your compilable code must fulfill the following requirements

1. The application must be able to read a mesh from an arbitrary `*.off` file, and convert the indexed face set representation in the `*.off` file to a half-edge data structure (see Figure 1, left). Some `*.off` files will be available for download from the course website. You are strongly advised to make use of the OpenMesh library, which reads `*.off` files and stores them in a half-edge data structure. Essentially you are asked to familiarize yourself with OpenMesh via the online documentation and code examples (see `www.openmesh.org`).
2. Once the file is read and stored, your application must render the mesh to the viewport. The minimum requirements are that you render the mesh flat shaded (Figure 1, middle) and as a wireframe composite (Figure 1, right). In subsequent assignments you will visualize the results of various algorithms with per-vertex scalar and vectorial quantities. Keep this in mind when coding your mesh renderer.
3. For the user interface (UI) the minimum requirements are that the user must be able to (a) translate, rotate and zoom the model by holding the left mouse button over the viewport and dragging the mouse, and (b) manually select the rendering mode. Translation, rotation and zoom modes, as well as the rendering mode must be manually adjustable via e.g. radio-buttons or context menus (be creative).
4. Using the OpenMesh API, the application must implement a breadth-first search algorithm, which collects vertices that are within an edge distance d from an arbitrary source vertex (make d and the source vertex selectable via the UI). Faces adjacent to these vertices should be rendered, such that they are clearly discernable from the rest of the mesh.

For the UI and rendering we recommend using OpenGL, GLUT and GLUI (for GLUT and GLUI, see links on the course website). One common alternative is (Open Source) Qt, combined with libqglviewer. GLUI/GLUT is lightweight and simple to use, while Qt/libqglviewer is robust and comes with many useful auxiliary data structures and camera classes. The decision on which of these combinations works better is left to you. In both cases, you will be using the OpenGL API, for which a few tutorial links are provided on the course website.

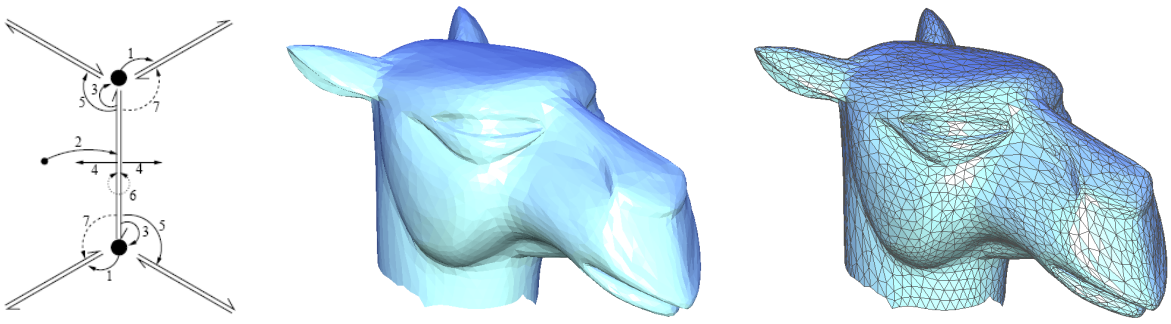

Figure 1: Mesh "Hello World"

2 Theory

1. Prove the Euler characteristic $V - E + F = 1$ for a triangular mesh homeomorphic to a disc by induction.
2. Sketch and describe the modifications that must be made to a half-edge data structure for a single *edge collapse* operation.
3. Given points in $\mathbb{R}^d$ with scalar values in the range $[-\varepsilon, +\varepsilon]$, describe the range of possible values for the (interpolating or approximating) MLS function when using constant ($0th$ order) polynomials $f(\mathbf{x}) = c$. Does the choice of weight function $\theta(d)$ have any influence on this range? Please describe your answer in sufficient detail.
4. Can the Marching Cubes algorithm cause aliasing artifacts? Discuss.