Ian Stewart
iastewar
10045484

Algorithms, data structures, and design decisions used

- Edge buffer functions implemented in import.cpp. The edge buffer is created whenever an object is loaded. Its size is the number of vertices by the maximum number of adjacent vertices to any vertex (this way the edge buffer will be as small as possible to save memory but will still maintain the speed of an array). When it is created, V, T1, and T2 are set for each edge.
- The crease (model) artist flag is also set upon creation for a default set of min/max values specified in the Model.cpp constructor. Crease flags are set based on the angle of the normal of T1 and T2 for each edge.
- The edge buffer is updated every frame before the model is rendered (display function in main.cpp). To update the edge buffer, the camera location is passed as parameters (calculated from the first 3 values of the last row of the inverse model view matrix) and used to calculate the eye vector by subtracting it from the center of every triangle. The normal of each triangle is calculated by getting the cross product of two edge vectors of the triangle. If the dot product of the normal and eye vectors is negative/positive update F, B, Fa, and Ba flags appropriately.
- The crease (user and model) artist flag is set whenever the edge buffer is updated. The user flag is set in the same way as the model, except that it is only set when both adjacent triangles are front facing (Fa=1 and Ba=0).
- The slope steepness artist flag is set whenever the edge buffer is updated. Is is set whenever the angle GA is between a min/max value as set by the user (default min=27.5 degrees, max=57.3 degrees). To calculate GA for an edge, a function has been added to import.cpp called setMVflags which is called from shapeMeasures. It also rotates the slope steepness direction.
- A function has been added to Model.cpp to render the edge buffer. glLineStipple is used to draw dotted and dashed lines. All lines are drawn according to boolean flags set by the GLUI user interface to specify colors, line styles, min/max crease values, and turn certain lines on or off.
- All triangle filling render methods have been modified to use glPolygonOffset so that lines are rendered properly.
- PA-1 (gooch shading), PA-2 (attribute based texture mapping), and PA-3 are also included with all bonuses.

References

- Code for shader.cpp and part of main.cpp taken directly from TA (GLSL_Lighting example). This code loads, compiles, and links the shaders.
- Code skeleton from instructor (basic OBJ loader) on course website
- Matrix inverse function from http://stackoverflow.com/questions/1148309/inverting-a-4x4-matrix. This was originally from the Mesa implementation of the GLU library.
- GLUI