Scientific Visualization

Project I

Due November 11, 2011

The purpose of the this project is to help you

(1) become familiar with typical data structures for mesh processing.
(2) review relevant mathematical concepts and their numerical computation algorithms.
(3) acquire experience and develop intuition with shape visualization.

Detailed tasks are listed below. Please submit everything necessary to compile and run your program, and a project report that contains results and discussions. **Note it is impossible to obtain a good grade without a well-written report. Figures are mostly welcome.** Feel free to explore and have fun!

Note: Use **JavaView** for the development. Be sure to provide a README file so that we know how to run your program.

1. **(Euler Characteristics)** Compute V-E+F for all the test models. Find all of the handles in these models and show them in pictures. A handle is also sometimes called a tunnel depending on its configuration relative to the rest of the shape. What relationship do you find? Can you provide some intuition about it?

2. **(Mesh coloring)** Add the following functionalities to **JavaView**:

   a. Assign a color to each polygon based on its normal as follows, $R = |N_x|,\ G = |N_y|,\ B = |N_z|$. (Normal Map).
   b. Assign a color to each vertex based on its 3D coordinates according to the following map $R = f(\lfloor V_x / L \rfloor),\ G = f(\lfloor V_y / L \rfloor),\ B = f(\lfloor V_z / L \rfloor)$ where L is a use-defined positive real number and $f(n) = \begin{cases} 0 & n \text{ is odd} \\ 1 & n \text{ is even} \end{cases}$. (3D checkerboard).
   c. Run your program using all two color schemes on the Bunny, the Happy Buddha, the Dragon, and the Feline. For the 3D checkerboard, try different L's. What problems do you encounter? Propose some solutions.
   d. Compute the 3D checkerboard on the Icosahedron. Do you see any new problem(s) that weren't obvious with the other models? What do you think is the cause of the problem? Suggest some solution(s).
   e. Assign a unique color to each polygon based on its polygon ID. Use any simple scheme. Run your program on the Bunny, the Happy Buddha, the Dragon, and the Feline. (Polygon ID map).

f. Compute the unweighted surface visibility measures as follows: Given a triangle T in the mesh M, its unweighted visibility measure is the percentage of viewpoints placed on an enclosing sphere S from where at least part of the triangle T is visible. To compute this measure efficiently for all triangles in M, you can discrete the enclosing sphere with a subdivided platonic solid. Make sure this subdivided mesh K has at least 200 nodes and that the nodes of K are spread out rather evenly, i.e., even coverage. Then for each viewpoint, compute the polygon ID map (as in item (e)), read the image buffer and increment the counter for each polygon whose ID is in the image buffer. Once all viewpoints have been processed, divide the counter value corresponding to each polygon by the number of nodes in K. What is the maximum of the unweighted visibility measure?

g. Compute the weighted surface visibility measure which differs from item (f) only in the following regard: The counter value corresponding to each polygon needs to be weighted by the dot product between the normal of the polygon and viewing direction. In other words, head-on views are favored over side views. Make sure to views themselves are also weighted so that the weighted surface visibility has the same maximum value as the unweighted visibility measure.

3. **(Bounding boxes)** Modify **JavaView** to compute the bounding boxes of a shape using two different methods. You will need to compute eigenvalues and eigenvectors of a 3x3 symmetric matrix. Note that **JavaView** already provides this capability.

   a. Compute the first and second moments of the input surface based on vertex locations. Construct a bounding box based on the first moment (center of gravity) and second moment (orientation). Compare this bounding box to the axis-aligned bounding box that is currently used in **JavaView**. Which one is more preferred, and why?

   b. The method described in (a) can be rather inaccurate if the vertices are unevenly distributed on the mesh (can you find such an example?). Ideally all the points on the mesh should contribute to the computation of the bounding box. Derive the formula for the second moment that considers all surface points. Implement this method and compare it with the bounding box in (a).

   c. Implement yet another way of constructing bounding boxes by using normal-based second moments to decide the orientation of a bounding box. Compare this algorithm to the one described in (b).

   d. Run your program on all test models that are provided. Draw the bounding boxes such that parallel edges are assigned the same color. Use red, green, and blue to indicate directions that correspond to major, middle, and minor eigenvectors of the second-moments.