Midterm CS3451, Summer 06 NAME: ___Solution______ Grade:_________.

Individual test. Do not look at other people's work. Please write legibly on the blue book. Make sure to put your name on its cover and to write the question numbers on the top-right corner of each page.

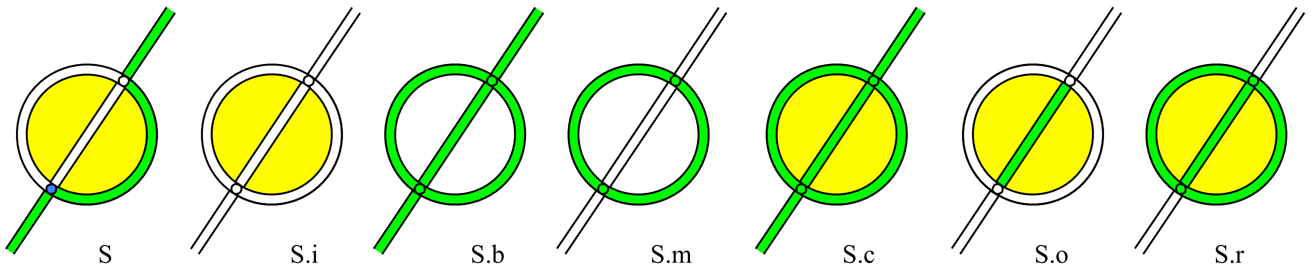
__/20 1) EDGE-TRIANGLE HIT: The points A, B and C are not collinear. P is the plane through them. T is the triangle with vertices A, B, and C. Points Q and R are exclusive. E is the edge joining them. E does not include its endpoints. (a) Provide the pseudo-code for testing whether P and E interfere. (b) Then, assuming that they interfere, provide the construction of their intersection point, I. (c) Finally, provide a test for checking whether I lies in T. (Use point and vector operators, not coordinates, in your solutions. Provide English explanations in addition to the geometric constructions.)

(a) N=AB×AC. P and E interfere when AQ•N and AR•N are both zero or have opposite signs.

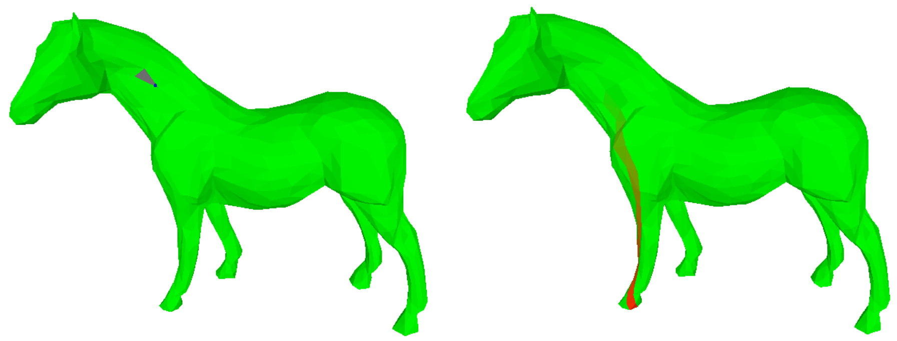
(b) s=QA•N / QR•N; I=Q+sQR

(c) QA•(QB×QC), QI•(QB×QC), QA•(QI×QC), QA•(QB×QI) have the same sign.

__/20 2) TOPOLOGY OPERATORS: Let S be the shaded set on the left. It contains two regions, one vertex, and three edges (two are semi-infinite). Fill-in (on this sheet) the cells in the sets (interior of S, boundary, membrane, closure, open-regularization, and closed-regularization) as marked below. Provide (in the blue-book) intuitive English definitions for each one of these sets. In these definitions, you may use the concepts of hair, cut, wound, and skin of S.



__/20 3) PATH ON A MESH: Assume that you have a Corner Table representation of a triangle mesh that is water-tight, manifold, connected and consistently oriented. You are given two corners s and e on different triangles. Provide an algorithm for computing the shortest (or nearly shortest) sequence of operators (.n, .p, .o, .l, .r) that would take s into e. (a) Provide a short English description of your approach, including the auxiliary data structures you would use. (b) Provide the pseudo-code for its implementation. State on what your loops or recursion operate (vertices, corners, triangles) and make sure that there is no ambiguity or hidden difficulty in your approach. (c) Draw a non-trivial example and show the output that your algorithm would produce on it.



(a) Grow edge-connected rings around e.t until you reach s.t. Then, backtrack from s to e.

(b) Pseudocode

```
void computePath() {
  for(int i=0; i<nt; i++) {Mt[i]=0;}; Mt[0]=1;
  for(int i=0; i<nc; i++) {P[i]=false;};
  int r=1;
  boolean searching=true;
  while (searching) {
     for(int i=0; i<nc; i++) {
       if (searching&&(Mt[t(i)]==0)&&(o(i)!=-1)) {
         if(Mt[t(o(i))]==r) {
           Mt[t(i)]=r+1;
           P[i]=true;
           if(t(i)==t(c)){searching=false;}; };  };     };
     r++;  };
  for(int i=0; i<nt; i++) {Mt[i]=0;};
  rings=1;
  while (t(c)!=0) {rings++;
    if (P[c]) {c=o(c); print(".o");} else {if (P[p(c)]) {c=r(c);print(".r");} else {c=l(c);print(".l");};};}; Mt[t(c)]=rings;};}
```

__/20 4) SIMPLIFICATION: (a) Provide the pseudo-code for collapseEdge(c), where c is a corner, that collapses vertex c.p.v to c.n.v. Your code must restore the V and O tables. In addition to the pseudocode, provide a drawing and comments to explain your code. (b) Explain precisely how you would evaluate the cost of (i.e error associated with) such a collapse. (You should use a practical algorithm that is efficient and simple to implement.) Make sure to provide the details of the geometric construction used for evaluating this error. (c) List the advantages and then the drawbacks of simplifications that are based on edge-collapse operations versus the vertex-clustering approach.

```
(a) void collapse(int c) { int b=c.p, oc=c.o, vnc=c.n.v;
    for (int a=b; a!=oc.n; a=a.r.p) {V[a]=vnc;}; V[c.p]=vnc; V[oc.n]=vnc;
    O[c.l]=c.r; O[c.r]=c.l;      O[oc.l]=oc.r; O[oc.r]=oc.l;  }
(b) D=c.n.v.g-c.p.v.g; N=vertexNormal(c.v); Error=D•N;
```

(c) Edge collapse simplification will remove vertices in large and nearly flat regions that would not be removed by vertex clustering. It preserves the connectivity of the mesh. Vertex clustering will work on any simplicial complex (not only manifold meshes) and can simplify meshes with many handles and components, which would not be simplified by edge-collapse operations.

__/20 5) SILHOUETTES: Assume that you have a connected, manifold, water-tight triangle mesh represented by a corner table. Assume that the triangles are oriented consistently so that they appear clockwise when seen from the outside. Provide the pseudo-code and detailed constructions for identifying all corners c such that the edge (c.p.v,c.n.v) is convex and is a silhouette when seen from the viewpoint E. (a) Explain precisely how to test whether edge (c.p.v,c.n.v) is convex and provide the pseudocode for that test using vector and corner-table operators assuming that the only thing you are given is c. (HINT: Note that when the edge is not-convex, triangle c.o.t appears clockwise to a viewer at c.v.g). (b) Then provide the details of the implementation of the test for checking whether edge (c.p.v,c.n.v) is silhouette. (HINT: A silhouette edge bounds a clockwise and a counterclockwise face as seen from E. Note that a silhouette edge may, but needs not, be visible.) (c) Provide the structure of the overall algorithm that uses these two tests.

(a) c.v.g should see triangle c.o.t as clockwise

```
(b) boolean s(A,B,C,D) {return((AB×AC)•AD); };}
convex[c] = s(c.v.g, c.o.v.g, c.o.n.v.g, c.o.p.v.g);
silhouette[c] = s(E,c.v.g,c,c.n.v.g,c.p.v.g)!=s(E,c.0.v.g,c,c.0.n.v.g,c.0.p.v.g);
(c) ∀c, good[c]=convex[c]&&silhouette[c];
Note that good[c]==good[c.o]; hence, we will draw edge (c.p.v.g,c.n.v.g) only when c<c.0;
```