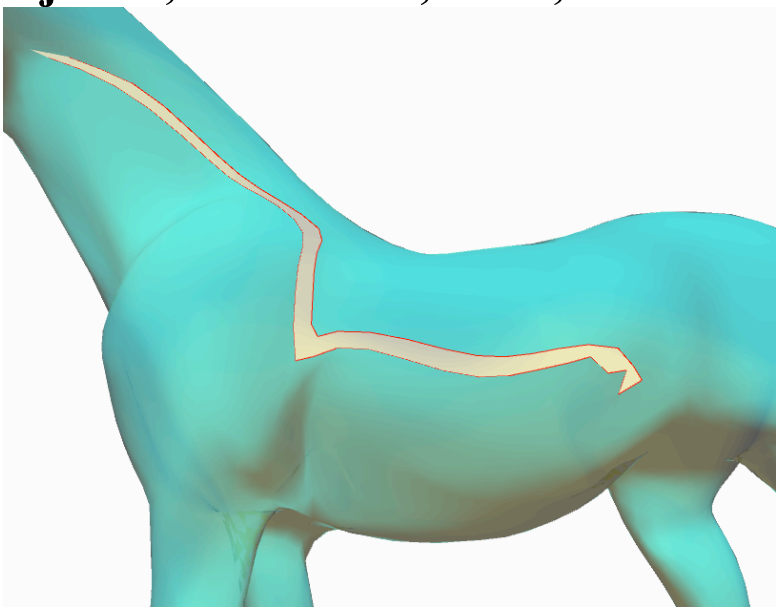
CS3451: Project 4, due Nov 1, 2011, teams of one or two

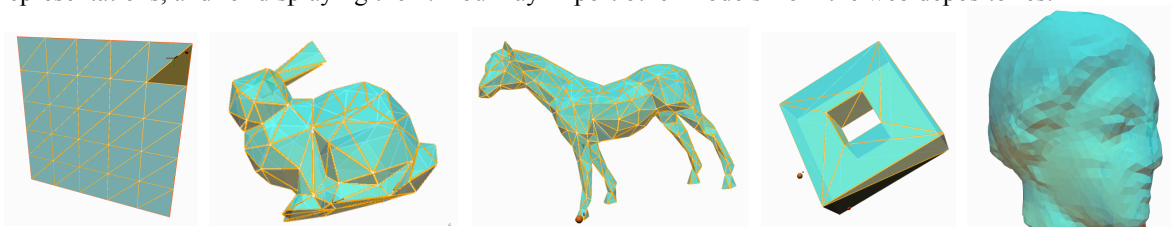


1 Objective

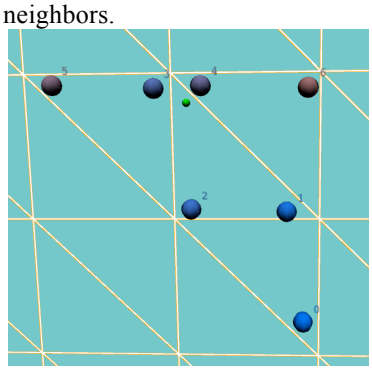
Learn how to program algorithms that construct, traverse, modify, and display a triangle mesh.
Apply them to the simulation of crack propagation.

2 Provided code

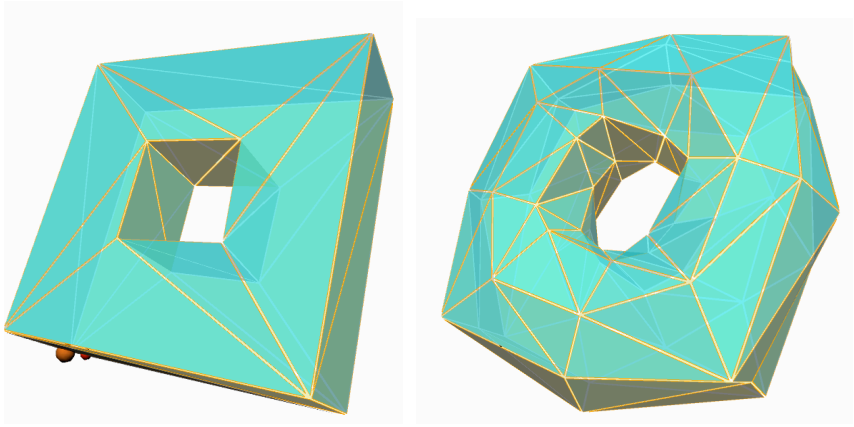
The provided *Viewer* base code offers tools for reading various models from files ('G'), for building their Corner Table representations, and for displaying them. You may import other models from the web depositories.



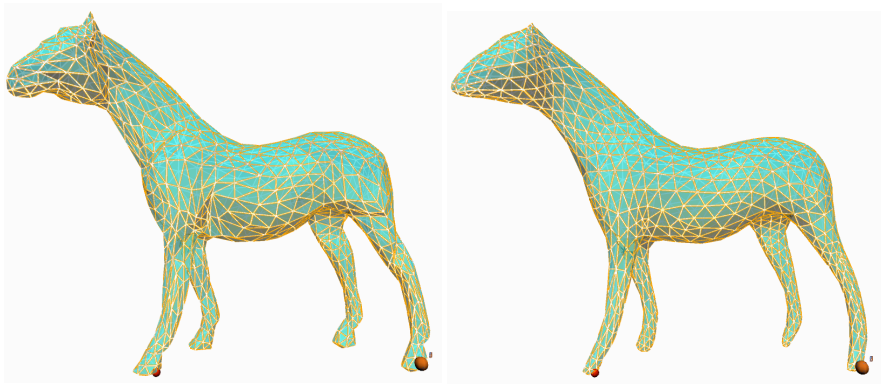
Navigation is performed by keeping SPACE or 'z' pressed and moving the mouse (without pressing the mouse button).
Viewer lets you walk from a corner c to its opposite 'O', next 'N', previous 'P', swing 'S', unswing 'U', left 'L', and right 'R' neighbors.



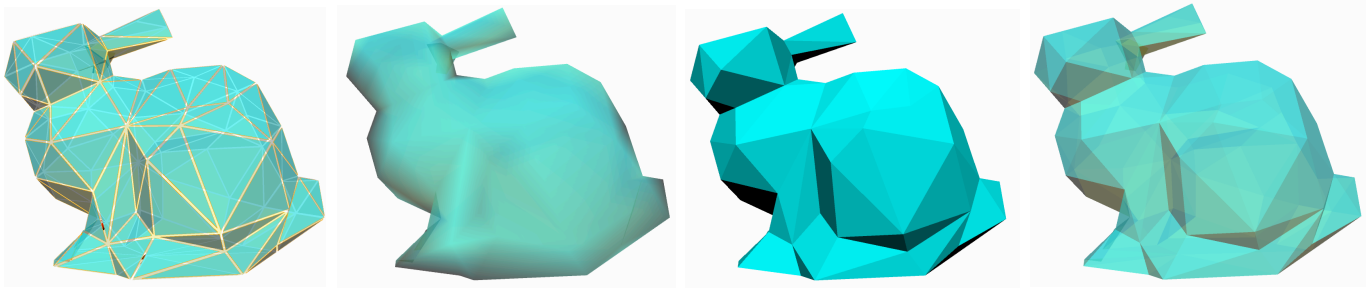
Viewer supports uniform subdivision 'Y'



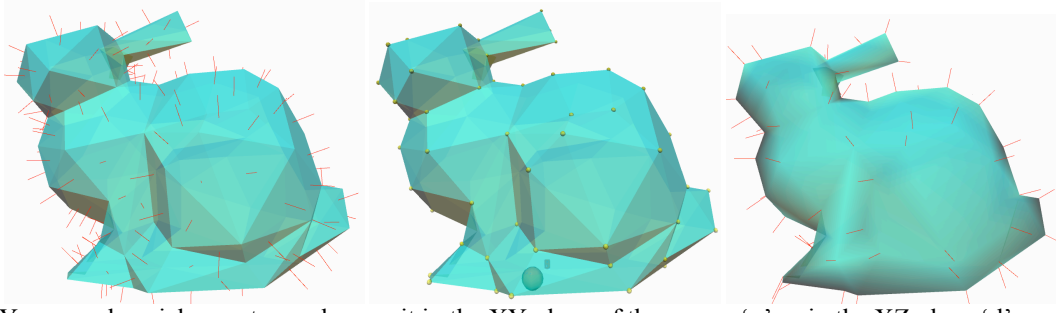
and smoothing ‘B’



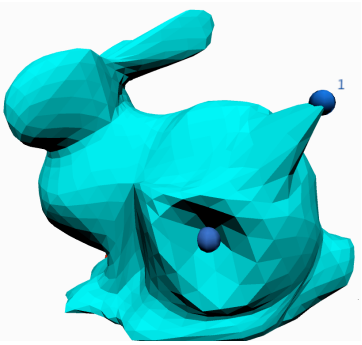
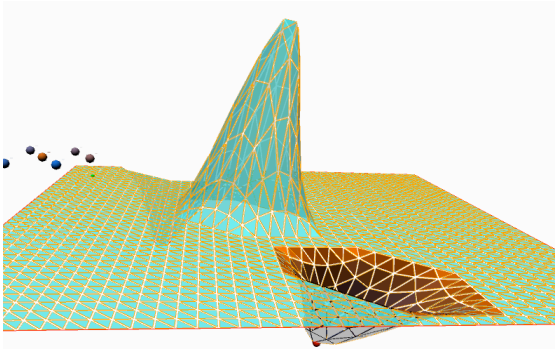
and different viewing modes (‘-’ showing edges, ‘_’ flat shading, ‘=’ transparency),



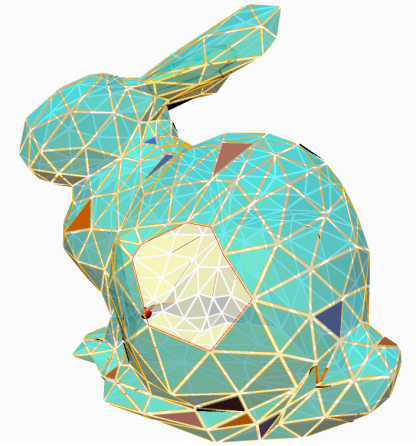
It can show triangle normals ‘|’, and also vertices ‘^’ and vertex normals (when in smooth shading mode).



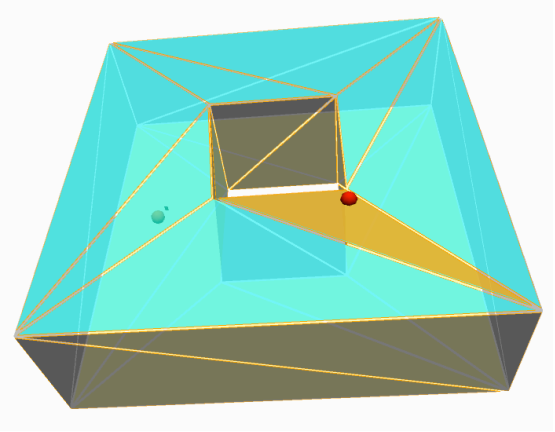
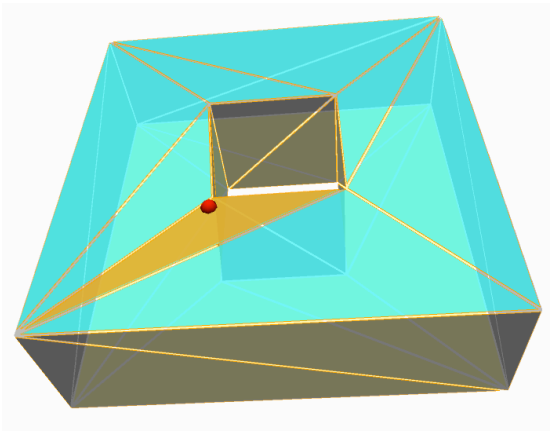
You can also pick a vertex and move it in the XY plane of the screen ‘v’ or in the XZ plane ‘d’.

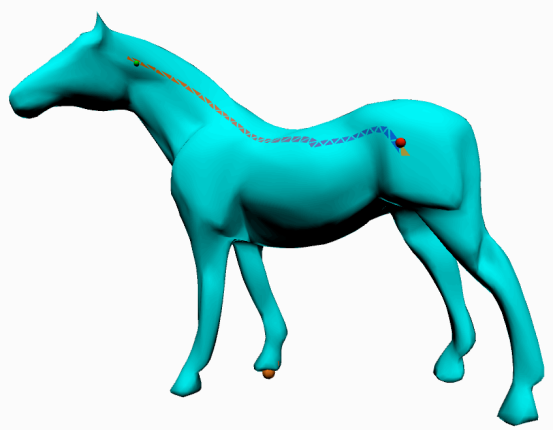
You can also delete selected triangles ‘x’ and recomputed the Corner Table ‘`’. Border edges are drawn in red.



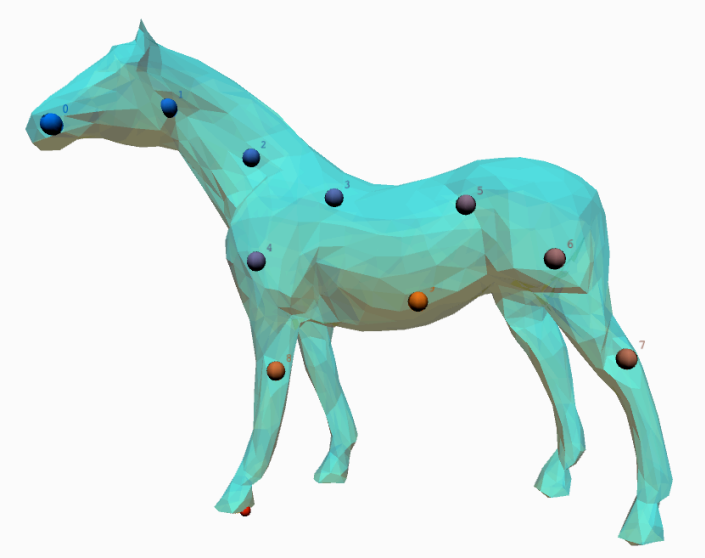
Viewer lets you flip the opposite edge

Viewer also offers tools for tracing a path ‘p’ between a seed triangle ‘s’ and the current triangle ‘c’.



Finally, viewer lets you place 10 markers on the mesh. Keep ‘0’, ‘1’… or ’9’ pressed and glide the mouse over the mesh, then release.



You need to become familiar with the use of Viewer and understand how these various functionalities have been implemented. IN particular, you need to know how to write the code to:

- flip an edge
- trace a path

To make sure that you understand these solutions, you should practice implementing your own version.
Understand how the Corner Table represents a triangle mesh and how it is constructed (at least the naïve construction of the O table).

3 Deliverables

3.1 Edge flip

Provide a short high-level description in English your version of the edge flipping code. Strive for clarity, simplicity, and conciseness. Include a figure if it helps.

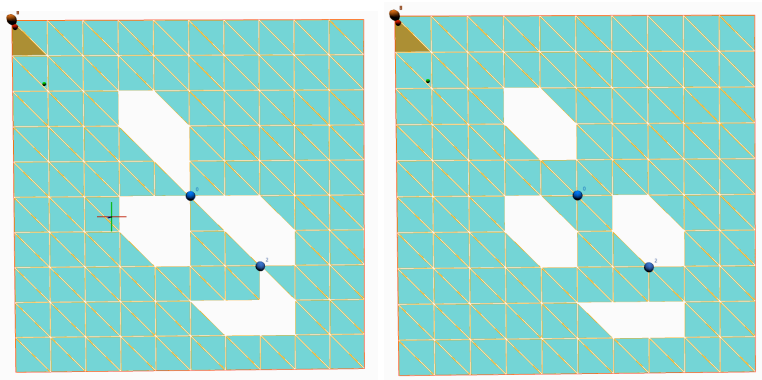
3.2 Path trace

Provide a short high-level description in English your version of the path tracing code. Strive for clarity, simplicity, and conciseness. Include a figure if it helps.

3.3 Non-manifold vertices

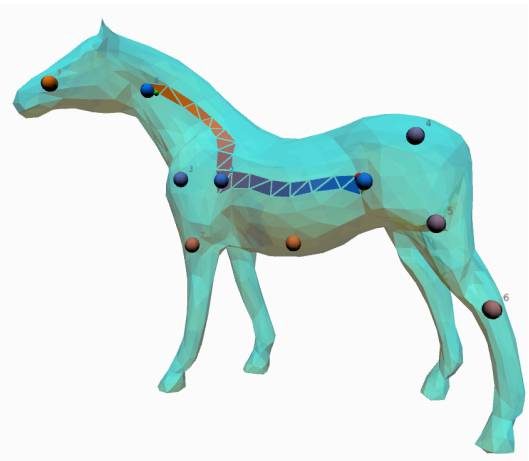
Produce concise code that correctly detects and shows all non-manifold border vertices (as large spheres).
In the write up, provide a brief outline of your approach and an elegant piece of code that marks these vertices.
Include a figure with an example output of your program showing that your program works on meshes with more than one non-manifold vertex. Explain how to activate this function. Provide a mesh file with several non-manifold vertices as shown below.
To do so, use viewer to delete some triangles ‘x’, to recompute the representation ‘`’, and to save it on file ‘W’, which creates a file names mesh.vts. You will need to add its name to the list of example files.

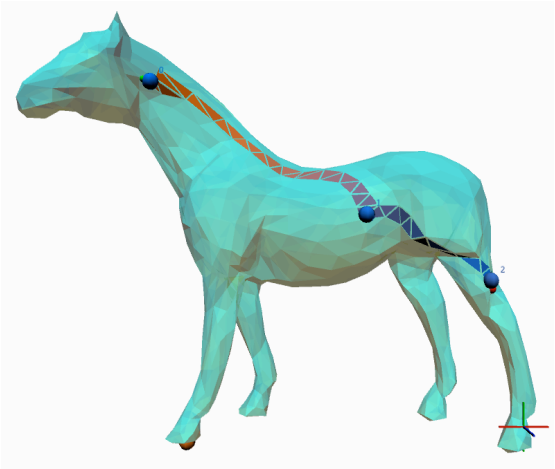
Suggest and implement a mesh modification that removes these non-manifold vertices (either by adding one triangle to each or by replicating the non-manifold vertex.



3.4 Spanning tree

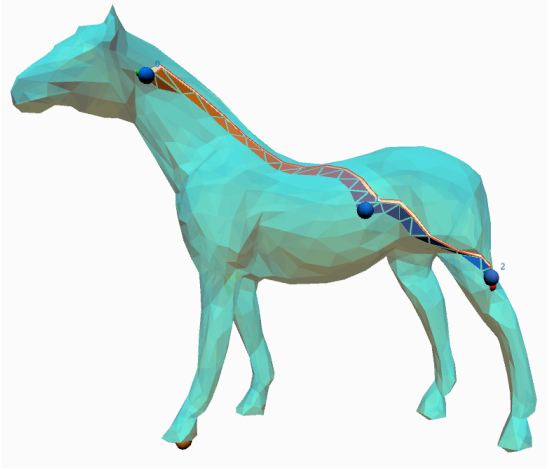
Identify the triangles associated with each marker and compute a minimum (or nearly minimum) spanning tree of edge-connected triangles that connect the marker triangles. Strive to minimize the number of triangles, or in the sum of their area, or in the length of the border of the spanning tree. Your algorithm should be interactive (less than 10 seconds for a model with 1000 triangles).
The figure below shows only a trivial portion of this spanning tree. Implement the algorithm for computing the complete spanning tree. In the write up, say what you optimize and explain how you do this (clearly but succinctly). Include references to printed or posted material that inspired your solution. Show several examples with complex configurations to show that your algorithm works and produces nearly optimal path trees.

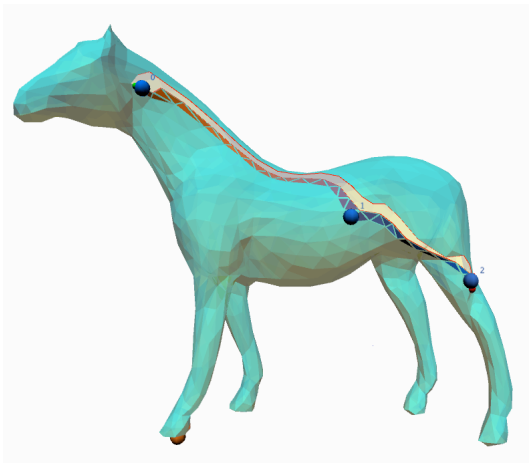




3.5 Cut

Implement an animation that starts at the triangle of marker 0 and progressively advances through the path tree opening the mesh simulation the propagation of a crack that advances and bifurcates and enlarges with time. A possible approach is to use the distance associated with the path and to "cut" the mesh along the left border between the spanning tree and the rest of the mesh.
By cutting, I mean that the triangles on opposite side of an edge of that border a no longer opposite of each other and that the vertices of the border are duplicated so that each side owns its own vertex. Then, you spread these vertices as in smoothing, by pulling them towards the centroid of their incident triangles. Synchronize that pull so that the cracks opens up progressively over time and that it also propagates along the spanning tree border, as a real crack would.
In the write up, provide a concise explanation of how you have implemented this part, include **only the most relevant** snippets of code (only if they are **short**, elegant, and easy to understand), and include a few images showing what your code does.





3.6 Extra credit

Some ideas:
Implement a physical action that simulates why the crack occurs (collision with a flying ball, bulge).
Have particles (fire?) fly out through the crack.
Use a texture map to simulate the burning of the mesh away from the crack.

3.7 Final animation

Produce an animation that will show a realistic simulation of the propagation of a crack that advances through the mesh, splits several times, and expands. Design the spanning tree so that it spells the GT logo on the horse. Produce a video of your final animation. The beginning of the video should include a title, a reference to the class (CS3451 - Fall 2011), and the names and photos of the team members. Post your write-up and video on T-square. Bring 2 print-outs to class on the due date.