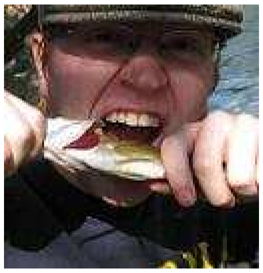
CS 3451

Jarek Rossignac

11/25/08

P05: Bulge

by

Derek Tatum	Michael Cook	Patrick Jahn
		

Our Bulge Project does LOTS of things. Firstly, we allow the user to draw a Polyloop using the same code as Project 4. Then, with the press of a key, the user can convert the 2D Polyloop to a 3D mesh, triangulate it, spray, re-triangulate based on the sprayed points, then smooth out the distribution of the new points with a Laplace smoothing algorithm, and finally compute the bulge. The bulged mesh can be viewed in a 3D mode with the press of another key, and various methods can be computed on it. Some methods that can be performed on the mesh include Isolation color rings, Shortest path coloring, and displaying vertex and triangle normals. Note that many of these methods, such as basic corner operations, can be computed in either 2D or 3D mode. Some methods, however, like the MAT and MAT spheres methods can only be displayed in 2D mode. All of the external source code we used came from the demos that Dr. Rossignac had posted on his website - specifically the Centers (for barycentre calculations), CT(for mesh class and methods), and Polymorph(for polyloops) applets.

The approach used in this assignment for actually performing the bulge is quite different from the bulge styles seen in PLUSHIE and TEDDY. The biggest difference between the three is that PLUSHIE, because of its design to have real-world application, maintains the "rest length" of each edge of the mesh (because it is intended to eventually be turned into cloth). Our bulge has no such constraints. With PLUSHIE, the user draws a 2D mesh for each piece of their plush doll. The system then mimics the physical act of stuffing the doll when it performs the bulge, by moving each edge out a small amount along its normal and then adjusting edge lengths to retain length. Thus, the bulge in PLUSHIE is not based off of a medial axis. With TEDDY, the original curve drawn by the user is triangulated using a constrained Delaunay triangulation. It then uses the triangulation to create a basic medial axis transform estimate. Then the algorithm starts on the terminal triangles of the mesh (those triangles with two or more edges on the border of the mesh), "pruning" away at the medial axis to create a chordal axis. The pruning process removes excess size from the medial axis for the next step. "Next, each vertex of the spine is elevated proportionally to the average distance between the vertex and the external vertices that are directly connected to the vertex. Each internal edge of each fan triangle, excluding spine edges, is converted to a quarter oval, and the system constructs an appropriate polygonal mesh by sewing together the neighboring elevated edges. The elevated mesh is copied to the other side to make the mesh closed and symmetric. Finally, the system applies mesh refinement algorithms to remove short edges and small triangles." [TEDDY Siggraph paper, 99] (Note: This section was quoted to better preserve the mechanics of the bulge in TEDDY.) So as a whole our algorithm is more similar to that in TEDDY but we do not include the iterative steps of pruning the medial axis so as to create a smoother shape once the 2D mesh is bulged.

What our bulge allows users to do:
*Draw, denoise, resample, refine, scale, translate, rotate, a polyloop
*Resampling: The resampling that our program uses redistributes the existing points more evenly, which is especially useful after individually editing vertex locations.
*Altering the number of control points for the polyloop is done by pressing the "<" and ">" symbols. It merely will roughly double or half the number of vertices in the polyloop by placing a new vertex at the midpoint between each vertex, or by removing every other vertex.
*Our triangulate method is based upon professor's Delaunay triangulation using a quartic algorithm. At the start of the method, we remove all existing triangles (in case we are re-triangulating). We also clean away all unused vertices and deleted triangles and "zero out" the z axis. Then we iterate through 3 "for" loops which represent ever-changing sets of 3 differing vertices which are triangulated based upon their distance to the circumcenter (of a circle) and their orientation (cw or ccw).
*Calculate barycentre points of triangles by calculating a weighted sum of the vertices of the triangle using code for the centers demo applet.
*Show an approximation of the MAT and show the corresponding circles by connecting adjacent barycentres. The radii for the spheres around each barycenter can be determined by calculating the distance between the barycentre and the min, max, or average of the three vertices of the associated triangle.
*Convert from a 2D polyloop/mesh into a 3D Mesh by elevating each interior vertex by the maximum corresponding height out each MAT sphere containing that vertex.
*We perform a shortest geodesic path along the 3D mesh between two triangles using the Professor's coloring code. The coloring reveals how many triangle unites there are to traverse between the two points.
*We perform an isolation algorithm which uses Professor's coloring code to color successive rings of triangles which are centered around a given point.
*Perform a Laplace smoothing on the polyloop by calculating a weighted average of the surrounding triangles for each interior vertex, then moving the given interior vertex a set scalar amount towards that point. (ShiftedPoint = OldPoint + WeightedSum/factor * Vec(OldPoint->WeightedSum)).
*We Save/Load polyloops to file
*We Show/hide polyloop vertices
*Spray random points into the triangulated 2D mesh using two different algorithms (based on random spray and the calculated barycenters of triangular subdivisions of each triangle and then repeats the Delaunay triangulation including the new spray points.
*Show the mouse projection onto the 3D mesh, the closest corner to the mouse projection, and the opposite of that corner

EXTRA CREDIT:
Our project is filled with methods that could be considered for extra credit. Here is a list:
*We have two spray modes (Sierpinski and random) which can be toggled with 'y'. The Sierpinski spray should be performed on a lower amount of initial vertices.
*We compute the geodesic distance between two triangles.
*We compute the path between two triangles
*We have 3 different radius modes (toggle with 'u','j', and 'i') for the computation of the bulge (radii of the MAT spheres).
*We compute both vertex normals and triangle normals.
*We use triangle operators in both 2D and 3D (we use o,l,r,n,p,s).