Polyloop Manipulation

Created by Richard Shemaka, Alex Scarlata, Nick Poirier

November 25, 2008

Polyloop Manipulation

Created by Richard Shemaka, Alex Scarlata, Nick Poirier

Overview

Polyloop Manipulation is a 2D/3D curve and mesh editor built in Processing. It has the capacity to produce:

- A master polyloop with a variable number of control vertices.
  - Closed-loop curve-editing.
- A smoothed cloud of random points inside the loop.
  - Uses primitive Laplacian smoothing.
- A constrained Delaunay triangulation of the full figure.
  - Stored in corner tables for geometry(G), vertices index(V), opposite index(O).
  - Capable of full mesh traversal from corner to corner in 2D.
- A dense sampling of an approximate Medial Axis Transform.
  - Based on circumcenters of triangulated control vertices from the polyloop.
- A 3D bulge and/or shape.
  - Based on Medial Axis and mesh from 2D.
  - Capable of rotating shape manually or automatically.

Polyloop Manipulation was built upon Jarek Rossignac's 2D Geometry Sandbox [4].

Controls

The controls are as follows:

- 2D Functions
  - Hold ‘d’ and Draw – Enter new polyloop
  - Click and Drag – Manually alter the polyloop
  - ‘r’ – Refine the polyloop
  - ‘t’ – Resample the polyloop
  - ‘z’ – Translate the polyloop
  - ‘x’ – Scale the polyloop
  - ‘R’ – Reset
  - ‘S’ – Save
  - ‘L’ – Load
  - ‘w’ – Display/Hide the mesh, axis, and traversal
  - ‘q’ – Respray the interior of the polyloop
  - ‘e’ – Reset mesh index array, opposite table, medial axis, and 3D capacity
  - ‘m’ – Switch to full 3D and/or back to 2D
  - ‘b’ – Swith to bulged 3D and/or back to 2D
- Traversal Functions
  - ‘n’ and ‘p’ - Next and Previous in mesh index array
  - ‘o’ - The corresponding value in the opposite table
  - ‘s’ - Swing through a mesh
- 3D Functions
  - ‘a’ – Toggle automatic mesh animation

2D Methods

Spray

The spray operates by randomly selecting a point in the viewing space then tests to see if it lies within the polyloop. If it does, it keeps it; otherwise, it selects a new point and retests. It then runs a primitive Laplacian smoothing to make the mesh more ideal for a 3D bulge. The smoothing tests every point in the spray against every point in the cloud; if it's too close to another point, it moves both points away from each other. If the points get too close to the border, the smoothing lightly pushes them back. We did not implement any further 3D smoothing. Because of this the sampling becomes sparse in the 3D figure towards the stitching. Further 3D smoothing is risky because it also tends to shrink the volume of the shape.

Triangulation

The triangulation method has a triple for() loop that checks every set of three points in the cloud. It draws a circle that has the three points on its circumference and checks to see if there are any other points inside the temporary circle. If there are the circle is discarded and the for() loops move on. After a valid circle is found, the N point (weighted sum) of the triangle is used to verify that the triangle lies in the polyloop. If the triangle passes all the tests, it is then passed into the vertices index table (V) with order based upon the orientation of the points.

Medial Axis Transform

The medial axis sampling is created in much the same way as the triangulation. This time, the algorithm takes in only the polyloop control vertices. The temporary circles are drawn as before and their centers are tested to see if they lie in the polyloop. If a center passes, it is added to the medial axis array and its radius is recorded in a corresponding array.

Opposite Corner Table

The opposite corner index (O table) is formulated by comparing the next and previous values of every corner in the mesh with every other corner. If a corner is found to be an exact inverse with another point (V[p(i)]==V[n(j)] and V[p(j)]==V[n(i)]) the algorithm sets the O table values equal to the opposite corner. If no matches are found for a point the O table sets the index to itself.

Mesh Traversal

Traversing the mesh is done by picking a random point on the mesh and running it through a series of functions based on the corner tables. Next and previous (n() and p()) are accomplished by incrementing through V and looping back when necessary. Opposite is simply done by referencing the O table. Left and right (l() and r()) are combinations of O, n(), and p() while swing is simply left followed by previous.

3D Methods

Mesh Bulge

The z-axis values for the mesh are computed using the medial axis sampling. The bulge function loops every point in the mesh through every medial axis sample which contains that point. Based on their distance and the medial axis radius, it computes its corresponding z value via Pythagorean Theorem and selects the maximum.

This is very similar to the way in which TEDDY [1] bulges its 2D shapes. TEDDY uses a more refined version of the medial axis transform, which it calls the chordial axis. It computes the z value of the points along the chordial axis based on their distance from the polyloop. It then projects those points onto their respective medial axis spheres in the same manner we do. TEDDY, however, goes on to refine the polygon more by smoothing the triangle mesh. PLUSHIE [2] operates in a similar manner towards how the 3D mesh is generated.

Isolation Measure

We did not include the tools to determine isolation measure in our code, though we will explain how one would go about implementing it. To compute the isolation measure of a given triangle, one must compute that triangle's average distance to every other triangle in the mesh. We begin by selecting every triangle on the mesh and running a simple algorithm to take neighboring triangles and 'tag' them with a value. As the algorithm completes, the entire mesh becomes covered in temporary values relative to that point, which we then save. We then take the data and produce an aggregate value for each triangle dependent on values from the spread and therefore its relative isolation.

Mesh Mirroring and Stitching

Mirroring and stitching the mesh is done in one function with various steps. It begins by moving all the control vertices from the polyloop to the z=0 plane to ensure proper stitching. Then it proceeds to double the size of the vertices index table (V table) and the vertex geometry table to accommodate the mirror. The points are copied from the back to the front only now the z value for the geometry is multiplied by -1. Finally the old V table is copied into the new array twice, the second time with all of the V table references shifted by the length of the old V table in order to allow the second half of the new V table to point to the mirrored half of the mesh.

Sources

[1] Igarashi, Takeo “Teddy: A Sketching Interface for 3D Freeform Design”

< http://www-ui.is.s.u-tokyo.ac.jp/~takeo/papers/siggraph99.pdf>

[2] Mori, Yuki and Takeo Igarashi “Plushie: An Interactive Design System for Plush Toys”

<http://www.den.rcast.u-tokyo.ac.jp/~yuki/papers/plushie_siggraph07.pdf>

[3] Rossignac, Jarek “Bulge” < http://www.gvu.gatech.edu/~jarek/demos/bulge/>

[4] Rossignac, Jarek “Morphing Curves”

<http://www.gvu.gatech.edu/~jarek/demos/polymorph/>

[5] Rossignac, Jarek “Naïve Delaunay”

< http://www.gvu.gatech.edu/~jarek/demos/Delaunay/>

[6] Rossignac, Jarek “Triangle Centers” <http://www.gvu.gatech.edu/~jarek/demos/centers/>