

Ribbon modeling

Due date: Monday, November 17, 12:00 pm

The goal of this assignment is to build a simple interactive modeler, that would allow you to create ribbon-like shapes.

Ribbons. A ribbon is built out of a curve $p(u) = (x(u), y(u), z(u))$ in 3D. Intuitively, the ribbon can be thought of as a long rectangle of width 2 which is bent and twisted. At each point of the curve $p(u)$ we specify a twist angle $\alpha(u)$. The ribbon is a surface given by the formula

$$q(u, v) = (x(u) + v \cos(\alpha(u)), y(u) + v \sin(\alpha(u)), z(u))$$

The parameter u varies from 0 to $n - 1$, where n is the number of control points that we specify, and v varies in the range -1 to 1. The original curve is a line along the center of the ribbon. If the twist angle is zero for all u , then the ribbon is parallel to the X-Z plane.

The curve $p(t)$ will be called the spine of the ribbon.

Implementing ribbons. We will specify the spine using two orthogonal curve views, one looking at the XY plane, the other looking at the XZ plane. The twist angle will be specified using one view, showing a circle on which the angle can be chosen.

The ribbon can be specified using cubic B-splines. The user specifies n control points $p_0, \dots, p_{n-1}$, and for each point the twist angle α_i , $i = 0 \dots n - 1$ (the user interface for this is described below). Thus, a ribbon is represented by two arrays: an array of control points for the spine and an array of twist angles of the same size.

To render the ribbon we compute an approximation with quads in the following way. First, we subdivide the spine curve using the algorithms described in class. The result, after k subdivision steps, is an array of $2^k(n - 1) + 1$ points, which we denote $p_i^k = (x_i^k, y_i^k, z_i^k)$, $i = 0 \dots (n - 1)2^k$. We apply the same subdivision rules to compute the array of twist angles, α_i^k , $i = 0 \dots (n - 1)2^k$. Once both arrays are computed, we pick a number M , which is the number of polygons across the ribbon. A reasonable value would be 10, unless the twist angles are big, and control points are close together.

Finally we render the quads approximating the ribbon. We draw a total of $(N - 1)M$ quads, where $N = 2^k(n - 1)$. The vertices of quad (i, j) are

$$\begin{aligned} & (x_i^k + (2j/M - 1) \cos \alpha_i^k, y_i^k + (2j/M - 1) \sin \alpha_i^k, z_i^k) \\ & (x_i^k + (2(j + 1)/M - 1) \cos \alpha_i^k, y_i^k + (2(j + 1)/M - 1) \sin \alpha_i^k, z_i^k) \\ & (x_{i+1}^k + (2(j + 1)/M - 1) \cos \alpha_{i+1}^k, y_{i+1}^k + (2(j + 1)/M - 1) \sin \alpha_{i+1}^k, z_{i+1}^k) \\ & (x_{i+1}^k + (2j/M - 1) \cos \alpha_{i+1}^k, y_{i+1}^k + (2j/M - 1) \sin \alpha_{i+1}^k, z_{i+1}^k) \end{aligned}$$

where i is in the range $0 \dots N - 2$, and j is in the range $0 \dots M - 1$.

1 What your program should do

Your program should have four windows: two for defining the spine, one for defining the twist, and one for showing the results. A sample program with two windows will be provided. The two windows for specifying the curve are the orthogonal views of the XY and XZ plane. In each case, use a fixed field of view, extending, say, from -10 to 10 in each dimension. Draw only the spine curve in these windows, do not draw complete ribbons. In the third window, just show a circle, and a line segment indicating the angle.

The spine curve windows. These two windows work in a similar way. Each time the user clicks on one of these windows holding the shift key, a control point is added to the curve, with XY (or XZ) coordinates determined from the position of the click, and the third (Z or Y) coordinate set to zero. If the user clicks on one of the existing control points without holding the shift key, this point is moved in the XY plane (XZ plane in XZ window) until the button is released. If there is no control point under the mouse when the click happens, no action is necessary. Finally, pressing “E” key removes the last control point.

Maintain the current control point for your operations. Each time a new point is added, it becomes current. Each time an old control point is selected by clicking on it, it becomes current. Initially, there are no control points, so there is no current one.

Each time a control point is added or a position of control point is changed, recompute the ribbon and update all windows, using `glutSetWindow` and `glutPostRedisplay` function calls.

Twist angle window. The twist angle window provides a simple interface for setting the twist angle. Whenever a control point p_i becomes current, the corresponding twist angle α_i is displayed in the twist window. To display a twist angle α , draw a unit circle, the coordinate axes and a unit line segment, such that the angle between the segment and the X axis is α . To change the angle, the user should click on the twist angle window. Compute the direction from the center of coordinates to the selected point and set the angle to the angle between this direction and the X axis. Do not forget to update the display. Note that this window has to be updated each time a change is made in one of the spiner curve windows.

View window. Select a viewpoint, say, somewhere in the direction (1,1,1), to show the ribbon in 3D. Make sure the viewing angle is large enough. Add lights to the scene as described in the previous homework or in any way you find interesting. Note that this is the only window where the whole ribbon is visible. No interaction is required in this window. However, for extra credit you may want to add a way to change the position of the camera (e.g. move around the ribbon, so that it is visible from different sides, or move the camera back and forth).

What to hand in. Just your working program.