

CSCI 550 (Fall 2011)

Assignment #2

Handout: Tuesday, Sept. 27, 2011

Due: 11:49pm, Tuesday, Oct. 25, 2011

All assignments will be submitted through oncourse.

The non-programming part can be in any format such as Microsoft Word, PDF, plain text or scanned hand-writing.

The programming part should include: (1) all source code files; (2) the executable file (.exe); and (3) a README file containing necessary instructions, known bugs, and any other notes you would like me to read.

Part 1: non-programming problems (75 pts)

1. The projection of a point $P = (x, y, z)$ to a plane in 3D space is a point on the plane that is closest to P . If the plane is defined by a point $P_0 = (x_0, y_0, z_0)$ and a normal vector $\mathbf{n} = (x_1, y_1, z_1)$, compute the projection of P on this plane.
2. Given four 3D points P_0, P_1, P_2, P_3 , design a robust procedure to determine whether the four points are on the same plane.
3. Prove that a rotation (about any coordinate axis) and a uniform scaling are commutative. Are they still commutative if the scaling is non-uniform? Why?
4. (a) Write a 2D rotation matrix that rotates by 90 degree clockwise.
(b) Write the above matrix as a product of 3 shearing matrices.
5. Derive (describe your derivation process) the transformation matrix for a reflection transformation about a plane with equation $x + y + z = 1$.

Part 2: Programming problems (100 pts)

I. 2D Bazier Curve Drawing.

Bezier curve is an important modeling tool in curve/surface design. It is defined by a sequence of connected line segments called control polyline, as shown in Figure 1(a). To draw a Bezier curve, a subdivision algorithm can be used to recursively refine the control polyline to generate progressive linear approximations to the curve.

Let P_i^0 ($i = 0 \cdots n - 1$) be the original vertices of the control polyline. The following procedure generates one subdivision:

1. Define P_i^1 ($i = 1 \cdots n - 1$) as the mid-points of all line segments in the control polyline, i.e. $P_i^1 = (P_{i-1}^0 + P_i^0)/2$, ($i = 1 \cdots n - 1$).
2. Similarly, define P_i^2 ($i = 2 \cdots n - 1$) as the mid-points of all new line segments formed by P_i^1 , i.e. $P_i^2 = (P_{i-1}^1 + P_i^1)/2$, ($i = 2 \cdots n - 1$).
3. Continue doing the above for each newly formed polyline, i.e. $P_i^k = (P_{i-1}^{k-1} + P_i^{k-1})/2$, where $k = 1 \cdots n - 1$, and $i = k \cdots n - 1$. When $k = n - 1$, there is only one point left, the process is then complete.

After this subdivision, the original control polyline is divided into two separate control polylines: P_i^i ($i = 0 \cdots n - 1$), and P_{n-1}^i ($i = n - 1 \cdots 0$), as shown in Figure 1(b). Each of these two polylines represents half of the curve. But they are now closer to the curve than the original polyline. We call the original polyline the level-0 approximation, and the two refined polylines the level-1 approximation. Naturally, the above procedure can be applied to each of the level-1 polylines to generate four new (more refined) polylines for level-2 approximation to the curve. The process can continue for the level-2 polylines and all subsequent higher level polylines to generate level-3, level-4, ... polylines.

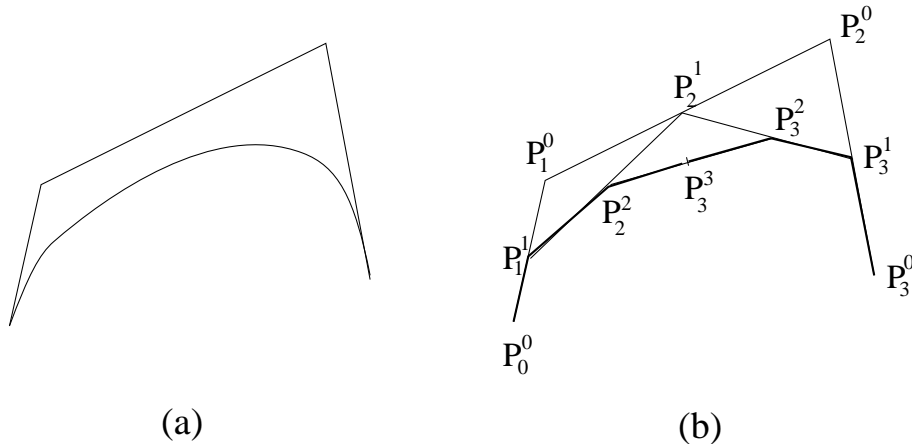


Figure 1: The Bezier curve subdivision algorithm

You will write an interactive program for the definition and drawing of Bezier curve. Your program should be able to:

1. Interactively define the initial control polyline as an OpenGL line strip. This can be done by a slight modification of your project 1 program.
2. **a number** i , ($i = 1 \dots 9$): draw the level- i polylines generated from the procedure described above.
3. **“+” or “-”**: increase or decrease the current level of approximation by 1, and draw the new polylines.
4. All functionalities of project 1 should be maintained, including polyline editing.

II. Surface Generation by Bezier Curve Extrusion.

For each Bezier curve you created on the X-Y plane, you need to build a 3D surface object by extruding the curve toward the z direction. The surface will be represented using a polygon mesh data structure, and viewed as a wireframe surface.

A keyboard or menu event will be used to switch between the 2D curve mode and the 3D surface mode. Once in the 3D mode, the surface object will be rotated continuously (animated) in all 3 directions, as in the rotating-cube example. Only wireframe rendering is required, i.e. no lighting and shading is necessary.