Computer Science 112
Art & Science of Computer Graphics
The College of Saint Rose
Fall 2015

Topic Notes: Bézier Curves

Our next topic involves the procedure for creating truly round objects and surfaces other than those that are derived from our built-in cylinder, cone, and sphere primitives.

`Prism`s

Before we get to that, we take a quick look at Ambrosia's `Prism` class. A `Prism` is quite simply a polygon in two dimensions which has been given a third dimension, or "stretched out" with a "thickness" of 100 units.

`Prism`s are similar to `Mesh` objects created with the `extrude` function, where the transformation is just a translation.

As with the `Spindle`, we begin with a polygon in the xy-plane – all of the z coordinates are going to be 0. Recall that we can specify the polygon directly or by using the `polygon` function to get a regular polygon with a given number of sides. If the polygon is defined in three dimensions, we can attach 0 z coordinates to a polygon with the `raisePoly` function.

We can use this to create an object with a square (but appearing diamond-shaped, as the corners are along the x- and y-axes).

On the Wiki: Prismdiamond

Our square prism is an object we can add and transform like our other objects. It has a profile of the polygon provided and stretches from -50 to +50 in the z dimension.

The `Prism` class can be used also to as yet another way to create our standard $100 \times 100 \times 100$ cube object...

On the Wiki: Prismcube

So far these are the same kinds of object we could have created with an `extrude`. Our next topic will make clear why we have interest in these `Prism` objects.

Bézier Curves

We'll begin with an example, then think about exactly what's happening.

On the Wiki: Simplebezier

The `Prism` and `Spindle` classes take an additional message: `bezier`, that instructs the class to treat the `profile` polygon as a set of control points for one or more *Bézier curve*.

Looking at the images generated by the above example, we can see that this results in objects with rounded instead of flat, angular surfaces.

In the example, we use a couple outline polygons to create objects with the `Spindle` and `Prism` constructs.

To understand what is going on here and how we can use this to create the curved surfaces we want, we will look in detail at Bézier curves.

We will consider *cubic* Bézier curves, at least to start.

These are defined by 4 *control points*; we'll call them A, B, C, and D. Intutitively, the curve defined by these points is the one traced out by an object "launched" from A in the direction of B and "arriving" at D from the direction of C.

All points on the Bézier curve always lies within the figure $ABCD$.

More formally and precisely, the points on the Bézier curve specified by points A, B, C, and D are the determined by the formula:

$$A(1-t)^3 + 3B(1-t)^2t + 3C(1-t)t^2 + Dt^3$$

where t ranges from 0 to 1.

If you think a bit about the formula, you'll see that at $t = 0$, it evaluates to A (as all other terms become 0) and at $t = 1$, it evaluates to D.

We can use a geometric construction to compute specific points on the Bézier curve. We will build an "A-frame scaffolding" to do this. To compute the point on the curve at, for example at $t = \frac{1}{3}$:

1. Draw straight lines from A to B, B to C, and C to D.
2. Find the points $\frac{1}{3}$ of the way from A to B (call it E), B to C (call it F), and C to D (call it G).
3. Draw straight lines E to F and F to G.
4. Find the points $\frac{1}{3}$ of the way from E to F (call it H), F to G (call it I).
5. Draw a straight line from H to I.
6. Find the point $\frac{1}{3}$ of the way from H to I (call it J). J is the point on the Bézier curve for $t = \frac{1}{3}$.

The Wikipedia article for Bézier curves has some excellent diagrams and animations that demonstrate this construction.

There are a number of web sites that you might find useful to specify Bézier curves. Links are on the lecture page.

Looking back at the `SimpleBezier` example, the polygon `outline`, when thought of as a Bézier curve, is really two sets of points, defining two cubic Bézier curves:

```
outline = raisePoly([(0, 0), (25, 100), (50, 25), (75, 0),
                     (75, 0), (50, -25), (25, -100), (0, 0)])
```

When a `Prism` or a `Spindle` is sent the `bezier` message, it will interpret the `profile` as sets of 4 Bézier control points.

We can define very complex and interesting curves with this method.

Some tips about stringing together Bézier curves:

- If you want a straight segment from A to D, place B at A and C at D.
- If you want two adjacent Bézier curves (defined by control points A, B, C, D and A', B', C', D') to join together smoothly, you need to make sure that D and A' are the same point, and that C, D/A', and B' are colinear. Otherwise, there will be a sharp corner at D/A'.

On the Wiki: Bell

The bell uses a straight segment, then two curved Bézier segments, but since we want it to be smooth, the last two points of the second segment and the first two points of the third are colinear.