OpenGL and Splines

OpenGL (and our friend JOGL) supports NURBS and Bezier curves and surfaces. Though NURBS are more interesting and more useful, they are also more complex to use and we don't have a lot of time, so we will just work with Bezier splines. These are relatively straightforward to use. You need control points -- 4 for a line segment, 16 for a patch. If you want a curve made out of N connected line segments you need 3N+1 points.

Before we get to OpenGL code, there is an issue with buffers. You probably want to specify arrays of control points, with each point being an array of three floats. Ultimately, the graphics card wants a byte stream. There is a Java method

FloatBuffer.wrap(float [] coords, int index, int size)

that takes a flat array of floats and produces the appropriate stream. *index* is the starting point in the *coords* array and *size* is the number of floats you need (4 for a curve segment, 48 for a patch).

Before you start drawing anything you must

a) Put your control points in an array of points.
b) Copy the points into the flat array of coordinates.

For curves this is easy. For N segments you have 3N+1 control points, which you probably define in one array. Your coordinates array will have 9N+3 floats; it should be an easy matter to copy the 3 coordinates of the ith control point into the 3i, 3i+1 and 3i+2 entries of the coordinates array.

For patches you need the 16 control points for one patch to be consecutive in the coordinates array. The TestPatch.java demo has just 2 patches, with the control points in a 7x4 grid (7 rows, 4 columns), so it is easy to write these one row at a time into a float buffer.

```
buffer = new float[7*4*3];
for (int i = 0; i < 7; i++)
    for (int j = 0; j < 4; j++)
        for (int k = 0; k < 3; k++)
            buffer[i*4*3+j*3+k] = data[i][j][k];
```

For anything more complex than this it is probably better to build the coordinates buffer on the fly as you are drawing the spline.

To actually display a spline you need 3 steps:

- Build an evaluator (essentially a spline function). You do this with glMap1 for curves and glMap2 for patches.
- Build a mesh of t-values (for curves) or (s, t) pairs (for patches). You do this with glMapGrid1 for curves and glMapGrid2 for patches.
- Evaluate the spline on the mesh to get actual geometry that can be displayed. You do this with glEvalMesh1 for curves and glEvalMesh2 for patche.

For curves there are 6 arguments to glMap1:

- A constant describing the control points; for us that is always GL2.GL_MAP1_VERTEX_3
- float min; the minimum value of t. For us this is always 0.
- float max; the maximum value of t. For us this is always 1.
- How many floats come between one control point and the next; for us this is 3.
- The order (degree+1) of the spline; for us this is 4.
- The FloatBuffer.

A typical call to glMap1 is

gl.glMap1f(GL2.*GL_MAP1_VERTEX_3, 0, 1, 3, 4, FloatBuffer.wrap(buffer, 0, 12));*

The 10 args to glMap2 are similar

- Constant GL2.GL_MAP2_VERTEX_3
- minimum and maximum values of t: 0 and 1
- How many floats come between one control point and the next in its row; usually 3.
- The order (degree+1) of the t parameter: 4
- minimum and maximum values of s: 0 and 1
- How many floats between one control point and the one the row below it: this is often 12 (4 control points).
- The order of the s parameter: 4
- The float buffer

Here is a typical call to glMap2:

gl.glMap2f(GL2.*GL_MAP2_VERTEX_3, 0,1, 3, 4, 0, 1, 12, 4, FloatBuffer.wrap(buffer, 0, 48));*

The other functions are easier.

To create a mesh for a curve:

gl.glMapGrid1(numDivisions, startT, endT);

as in

gl.glMapGrid1(10, 0, 1);

To evaluate and draw a spline curve:

gl.glEvalMesh1(<style>, firstIndex, lastIndex);

as in

gl.glEvalMesh1(GL2.GL_LINE, 0, 10);

For a patch:

```
gl.glMapGrid2f(numTvalues, startT, endT,
                        numSvalues, startS, endS);
gl.glEvalMesh2(<style>, firstTindex, lastTindex,
                                firstSindex, lastSindex);
```

as in

```
gl.glMapGrid2f(10, 0, 1, 10, 0, 1);
gl.glEvalMesh2(GL2.GL_FILL 0, 10, 0, 10);
```

Altogether, here is all that it takes to draw one segment of a spline curve, after the control point coordinates have been put into the buffer array:

```
gl.glMap1f(GL2.GL_MAP1_VERTEX_3, 0, 1, 3, 4,
                              FloatBuffer.wrap(buffer, 0, 12));
 gl.glMapGrid1f(10, 0, 1);
 gl.glEvalMesh1(GL2.GL_LINE, 0, 10);
```

Here is the corresponding code for a patch:

```
gl.glMap2f(GL2.GL_MAP2_VERTEX_3, 0f,1f, 3, 4, 0f, 1f, 12, 4,
                                        FloatBuffer.wrap(buffer, 0, 48));
gl.glMapGrid2f(10, 0f, 1f, 10, 0f, 1f);
gl.glEvalMesh2(GL2.GL_FILL, 0, 10, 0, 10);
```