OpenGL 3D Graphics in Liberty BASIC

Lesson Five: Complex Surfaces

by Robert McAllister

OpenGL makes it easy to create complex surfaces such as line strips, triangle strips and quad strips. A strip uses the last set of coordinates of one object as the first set for the next object. By far the easiest to understand is the Line Strip. The comments after each vertex explain it best.

```
' a line strip
CALL
 ClearView eyeX , eyeY , eyeZ , centerX , centerY , centerZ , upX , up
Y , upZ
CALL glColor4fv 1 , 0 , 0 , 1
CALL glBegin GL.LINE.STRIP
  CALL glVertex 0 , 0 , 0 'start of first line
  CALL glVertex 1 , 1 , 0 'end of first line, start of second line
  CALL glVertex .5 , 1.5 , 0 'end of second line, start of third line
  CALL glVertex -1 , .5 , 0 'end of third line, start of fourth line
  CALL glVertex -1.5 , 1.5 , 0 'end of fourth line
CALL glEnd
CALL RefreshView
WAIT
```

For a triangle strip the first three vertices describe the first triangle. Then the fourth vertex along with the previous two describe the next triangle, and so on. Since I couldn't make up my mind which color to use for the first triangle, this snippet also demonstrates how to convert normal RGB values for use with OpenGL.

```
' a triangle strip
CALL
 ClearView eyeX , eyeY , eyeZ , centerX , centerY , centerZ , upX , up
Y , upZ
COLORDIALOG "", chosen$
red = val(word$(chosen$,1))/255
green = val(word$(chosen$,2))/255
blue = val(word$(chosen$,3))/255
CALL glColor4fv red , green , blue , 1
CALL glBegin GL.TRIANGLE.STRIP
  CALL glVertex -1 , -1 , 0 'first set triangle coordinates
  CALL glVertex -1 , 1 , 0 'second set triangle coordinates
  CALL glVertex 1 , 1 , 0 'third set triangle coordinates
```

```
    CALL glColor4fv 1 , 1 , 0 , 1
'change color to make it easier to tell triangles apart
  CALL glVertex .5 , 1.5 , 0
'coordinate for second triangle, along with the previous two
    CALL glColor4fv 0 , 0 , 0 , 1
  CALL glVertex 2 , 1.25 , 0
'coordinate for third triangle, along with the previous two
CALL glEnd
CALL RefreshView
WAIT
```

The same principle applies for quad strips.

```
' a quad strip
CALL
 ClearView eyeX , eyeY , eyeZ , centerX , centerY , centerZ , upX , up
Y , upZ
CALL glColor4fv 1 , 0 , 1 , 1
CALL glBegin GL.QUAD.STRIP
  'coordinates for first quad
  CALL glVertex -.5 , -.5 , 0
  CALL glVertex 0 , -.5 , 0
  CALL glVertex -.25 , 0 , 0
  CALL glVertex .25 , 0 , 0
  CALL glColor4fv 0 , 0 , 0 , 1

  'coordinates for second quad, along with the previous two
  CALL glVertex .25 , .5 , 0
  CALL glVertex .75 , .5 , 0
    CALL glColor4fv 0 , 1 , 0 , 1

  'coordinates for third quad, along with the previous two
  CALL glVertex .25 , 1 , 0
  CALL glVertex .75 , 1 , 0
CALL glEnd
CALL RefreshView
WAIT
```

OpenGL also has the ability to create polygons, shapes with 3 or more sides. There is no practical limit to the number of sides in a polygon but there shouldn't be any concave areas (indentations).

```
' polygon
GL.POLYGON = 9
CALL
```

```
 ClearView eyeX , eyeY , eyeZ , centerX , centerY , centerZ , upX , up
Y , upZ
CALL glColor4fv 0 , 0 , 1 , 1
CALL glBegin GL.POLYGON
  CALL glVertex -1.5 , 0 , 0
  CALL glVertex -.3 , 1.1 , 0
  CALL glVertex 1.6 , .3 , 0
  CALL glVertex 1.5 , -1.3 , 0
  CALL glVertex -.7 , -1.4 , 0
CALL glEnd
CALL RefreshView
WAIT
```

In the next lesson we will start "Creating shapes"