OpenGL 3D Graphics in Liberty BASIC

Lesson Nine: OpenGL Calls and Argument Types

by Robert McAllister

OpenGL function names follow a standard naming system. All of them begin with “gl”, for “graphics library”. They are then followed with each word of the function capitalized, such as “glClearColor”.

You may notice that some function names are followed with a number and/or a few letters, such as “glRotatef”. In this case the function expects the individual supplied values to be floats. Then there are calls such as “glColor4iv”, which uses 4 integer (long) values in an array. When there’s a number it refers to the number of values the call needs.

Many of the OpenGL functions have several versions of the function name, but they all do the same thing. “glColor”, for example, has 32 versions. I can only assume this is due to backward compatibility or because OpenGL is system independent, meaning that it will work on any operating systems, Linux, Mac, Windows, or?

Command suffixes and data types:

Suffix	Data Type	C Type	OpenGL Type	LB Type
b	8-bit Integer	signed char	GLbyte	word or char[1] [1]
s	16-bit Integer	short	GLshort	short
i	32-bit Integer	long	GLint, GLsizei	long
f	32-bit Floating-Point	float	GLfloat, GLclampf	converted ulong [2]
d	64-bit Floating-Point	double	GLdouble, GLclampd	double
ub	8-bit Unsigned Integer	unsigned char	GLubyte, GLboolean	word or char[1] [3]
us	16-bit Unsigned	unsigned short	GLushort	ushort

	Integer			
ui	32-bit Unsigned Integer	unsigned long	GLuint, GLenum, GLbitfield	ulong

1. ^ For CallDll combine Byte+Byte=Word, for structs use char[1], based on Converting C types to LB types
2. ^ Using R4() function from ULong may be used as a "float" device
3. ^ See footnote 1. above

Of all the possible versions for a call, I have found that the float or double versions are usually the most dependable. Since a “Float” value is not an optional argument type with LB, the R4() function is used to convert the values to floats and then pass them to the call as a Ulong. (Many thanks to Brent Thorn for putting this function on his forum)

Below is how the description of an OpenGL function will typically look on MSDN. If you Google a function name, usually one of the first few results will give you the description for it. Results from "www.opengl.org/documentation/specs/..." will be in a plain text format so they load faster than MSDN.

```
void glRotatef(
    GLfloat angle,
    GLfloat x,
    GLfloat y,
    GLfloat z
);
```

And this is how it would be translated for LB. The four values are first converted with the R4() function.

```
    amt = R4( amt )
    x = R4( x )
    y = R4( y )
    z = R4( z )
    CALLDLL #gl , "glRotatef" ,_
      amt AS ulong ,_
      x AS ulong ,_
      y AS ulong ,_
      z AS ulong ,_
      glRotatef AS void
```

For calls that use an array of values, there will be a "v" at the end of the name and a struct should be used. The definition for “glVertex3dv” (3 doubles in an array) will normally be given as:

```
void  glVertex3dv(
    const GLdouble *v
);
```

Which would be translated in LB to:

```
Struct glVertex3dv,_
  X as double,_
  Y as double,_
  Z as double

glVertex3dv.X.struct=X
glVertex3dv.Y.struct=Y
glVertex3dv.Z.struct=Z

CALLDLL #gl , " glVertex3dv" ,_
  glVertex3dv as struct,
  ret as void
```

In the definition for some calls you may see the word “clamp”. This means that the values need to be between 0 and 1. You will usually see this with values that represent a percentage such as with glClearColor.

```
void glClearColor(
  GLclampf red,
  GLclampf green,
  GLclampf blue,
  GLclampf, alpha
```

In the next lesson we will take a look at OpenGL's companion, the “OpenGL Utility Library”