Display Lists

Mike Bailey

mjb@cs.oregonstate.edu

Oregon State University

Drawing a Sphere –
Notice a lot of Trig Function Calls and Pointer Arithmetic!

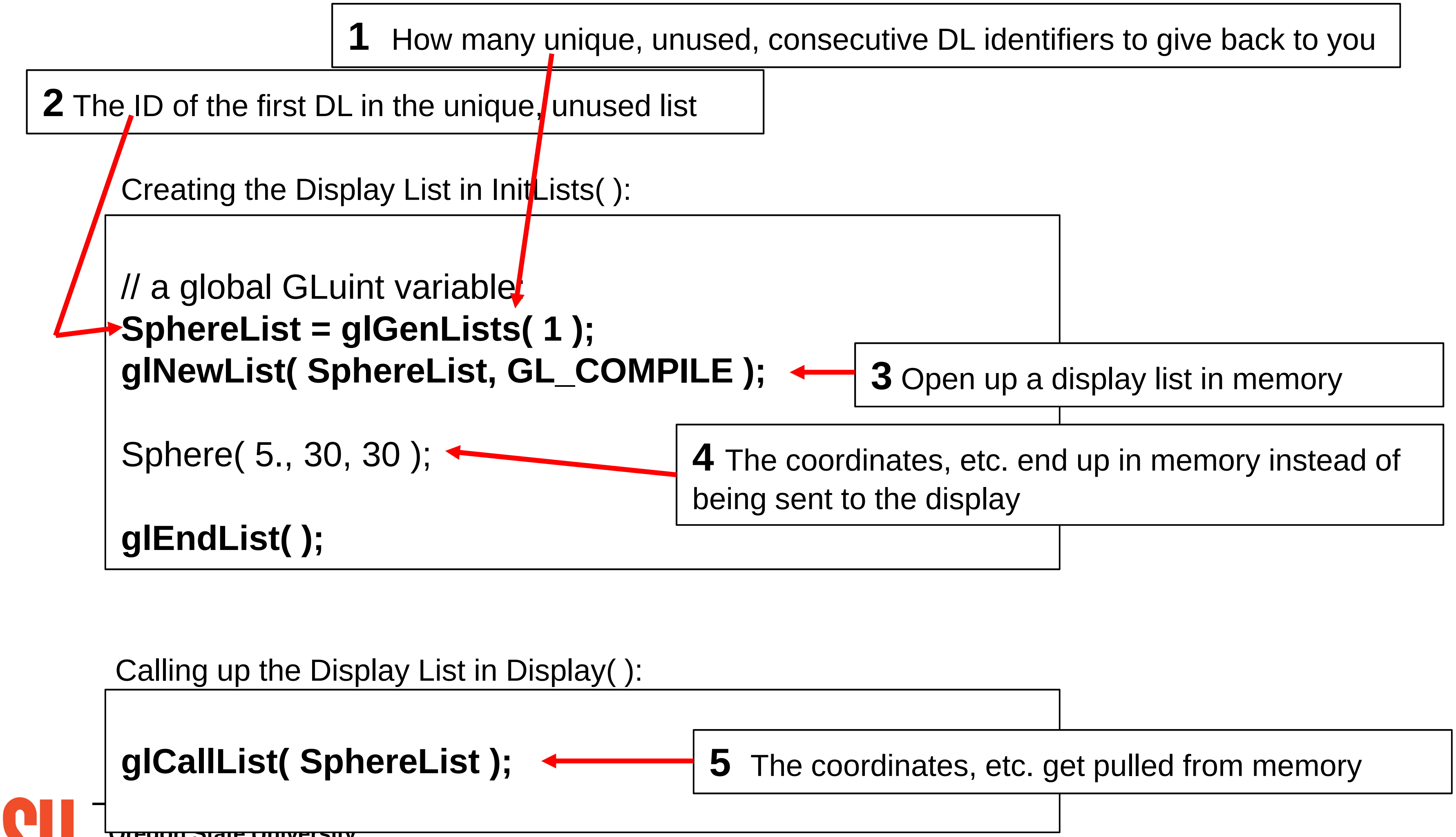
```
void
Sphere( float radius, int slices, int stacks )
{
	struct point top, bot;		// top, bottom points
	struct point *p;
	NumLngs = slices;
	NumLats = stacks;

	Pts = new struct point[ NumLngs * NumLats ];
	for( int ilat = 0; ilat < NumLats; ilat++ )
	{
		float lat = -M_PI/2.  +  M_PI * (float)ilat / (float)(NumLats-1);
		float xz = cos( lat );
		float y  = sin( lat );
		for( int ilng = 0; ilng < NumLngs; ilng++ )
		{
			float lng = -M_PI  +  2. * M_PI * (float)ilng / (float)(NumLngs-1);
			float x =  xz * cos( lng );
			float z = -xz * sin( lng );
			p = PtsPointer( ilat, ilng );
			p->x  = radius * x;
			p->y  = radius * y;
			p->z  = radius * z;
			p->nx = x;
			p->ny = y;
			p->nz = z;
			p->s = ( lng + M_PI    ) / ( 2.*M_PI );
			p->t = ( lat + M_PI/2. ) / M_PI;
		}
	}
```

```
top.x   =  0.;          top.y   = radius;       top.z   = 0.;
top.nx = 0.;            top.ny = 1.;            top.nz = 0.;
top.s   = 0.;           top.t    = 1.;
bot.x   =  0.;           bot.y   = -radius;       bot.z = 0.;
bot.nx = 0.;             bot.ny = -1.;             bot.nz = 0.;
bot.s   = 0.;            bot.t    =  0.;

glBegin( GL_QUADS );
for( int ilng = 0; ilng < NumLngs-1; ilng++ )
{
        p = PtsPointer( NumLats-1, ilng );
        DrawPoint( p );
        p = PtsPointer( NumLats-2, ilng );
        DrawPoint( p );
        p = PtsPointer( NumLats-2, ilng+1 );
        DrawPoint( p );
        p = PtsPointer( NumLats-1, ilng+1 );
        DrawPoint( p );
}
glEnd( );

glBegin( GL_QUADS );
for( int ilng = 0; ilng < NumLngs-1; ilng++ )
{
        p = PtsPointer( 0, ilng );
        DrawPoint( p );
        p = PtsPointer( 0, ilng+1 );
        DrawPoint( p );
        p = PtsPointer( 1, ilng+1 );
        DrawPoint( p );
        p = PtsPointer( 1, ilng );
        DrawPoint( p );
}
glEnd( );
```

```
glBegin( GL_QUADS );
for( int ilat = 2; ilat < NumLats-1; ilat++ )
{
        for( int ilng = 0; ilng < NumLngs-1; ilng++ )
        {
                p = PtsPointer( ilat-1, ilng );
                DrawPoint( p );
                p = PtsPointer( ilat-1, ilng+1 );
                DrawPoint( p );
                p = PtsPointer( ilat, ilng+1 );
                DrawPoint( p );
                p = PtsPointer( ilat, ilng );
                DrawPoint( p );
        }
}
glEnd( );
```

You don’t want to execute all that code every time you want to redraw the scene, so draw it once and call it back up

The solution is to incur the sphere-creation overhead *once*, and whenever the sphere needs to be re-drawn, just draw the saved coordinates, not the equations. This is a **Display List**.



1 How many unique, unused, consecutive DL identifiers to give back to you

2 The ID of the first DL in the unique, unused list

Creating the Display List in InitLists( ):

```
// a global GLuint variable
SphereList = glGenLists( 1 );
glNewList( SphereList, GL_COMPILE );

Sphere( 5., 30, 30 );

glEndList( );
```

3 Open up a display list in memory

4 The coordinates, etc. end up in memory instead of being sent to the display

Calling up the Display List in Display( ):

```
glCallList( SphereList );
```

5 The coordinates, etc. get pulled from memory