OpenGL Notes [a]

Stu Pomerantz

smp@psc.edu

http://www.psc.edu/∼ smp

September 7, 2004

[a]All material is adapted from: OpenGL ARB, et. al, "The OpenGL Programming Guide", Third Ed., Reading: Addison-Wesley, 1999

OpenGL Boolean State Changes

Some OpenGL states have many values.

- `glColor3ub()` ; Has about 16 million possible states.

Other OpenGL states are either on or off. These capabilities are turned on using:

- `glEnable(`*constant*`)` ;

They are turned off using:

- `glDisable(`*constant*`)` ;

OpenGL Boolean State Changes

For example, to turn on *back face culling* of polygons:

- `glEnable(GL_CULL_FACE) ;`

and to turn this state off:

- `glDisable(GL_CULL_FACE) ;`

OpenGL Display Lists

- Valid only for the OpenGL context in which they were created *unless you do some extra work.*
- Download geometry and *some* OpenGL state changes to GPU memory for execution later.
- Display lists are *retained mode* (e.g. structures remain on the card).
- Contrast this with *immediate mode* where geometry is downloaded to GPU memory each time it is displayed.
- *Fast* performance.
- Of course, this is a limited resource.

OpenGL Display Lists

Display lists are built like this:

`glNewList(`*GLuint id, GLenum mode*`) ;`

your code in the order it should be executed

`glEndList() ;`

- *id* is an unsigned integer used to identify the display list to GL.
- *mode* may be
  - `GL_COMPILE` Send information to GPU memory for later execution.
  - `GL_COMPILE_AND_EXECUTE` Commands are executed as they are placed into the display list.

OpenGL Display Lists

Once a display list is created, it can be executed by:

`glCallList(`*GLuint id*`) ;`

For example, at *initialization* time:

```
glNewList(1, GL_COMPILE) ;
  glColor3f(0,0,1) ;
  glBegin(GL_LINES) ;
    glVertex2i(0,0) ;
    glVertex2i(1,1) ;
  glEnd() ;
glEndList() ;
```

And at *display* time:

```
glCallList(1) ;
```

Raster Position

- The raster position is the position in *world space* to use for pixel and bitmap write operations.
- Set with `glRasterPos*() ;`
- For example: `glRasterPos2f(2.4, 3.5) ;`
- So, how to write pixels to the screen ?

glDrawPixels

`glDrawPixels(`*GLsizei width, GLsizei height, GLenum format, GLenum type, const GLvoid *pixels*`) ;`

- *width & height* specify the dimensions of the pixel rectangle to be written into the frame buffer.
- *format* specifies the format of the pixel data. For example: `GL_RGB` or `GL_GREEN`.
- *type* specifies the data type of the pixel data. For example `GL_UNSIGNED_BYTE` or `GL_FLOAT`. It is permisible to have floating point pixels.
- *pixels* specifies a pointer to the pixel data.

glDrawPixels

For example, suppose this was the array of pixels:

```
GLfloat pixels[4][4] = {
       {1,1,1,1},                          //brightest
       {0.75, 0.75, 0.75, 0.75 },
       {0.5, 0.5, 0.5, 0.5 },
       {0.25, 0.25, 0.25, 0.25 }    //dimmest
   } ;
```

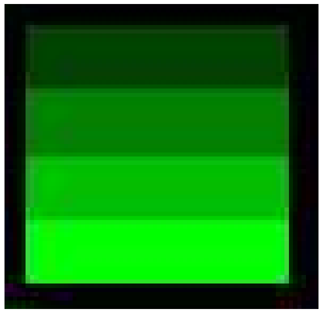
This would be the call

```
glDrawPixels(4,4,GL_GREEN,GL_FLOAT,pixels) ;
```

to draw the contents of this array on the screen as green pixels.

glDrawPixels

The result of the **`glDrawPixels()`** is this:



- Notice that the brightest pixels, which have the value 1 are at the *bottom* of the picture. See `glPixelZoom()` to change this.
- Note that `glPixelStore()` may need to be adjusted if the pixels are not divisible by 4.

GLUT Callback Functions

How to set the most important GLUT callback functions:

- `glutKeyboardFunc()` set the keyboard callback function.
- `glutMouseFunc()` set the mouse callback function.
- `glutReshapeFunc()` set the reshape callback function.
- `glutDisplayFunc()` set the display callback function.
- `glutIdleFunc()` set the idle callback function.
- `glutTimerFunc()` set the timer callback function.

Each of these functions takes as its argument a pointer to a function. The kind of function is different for each callback.

GLUT Callback Functions

`glutKeyboardFunc()` set the keyboard callback function. This function is called when a key is pressed.

Example:

```
void keyboard( unsigned char key, int x, int y ) {
  switch( key ) {
    case 'q':
      fprintf(stderr,"the 'q' key has been pressed\n") ;
    break ;
  }
}
```

The *x & y* variables contain the position of the mouse in window coordinates when the key was pressed.

GLUT Callback Functions

`glutMouseFunc()` set the mouse callback function. This function is called when the mouse button is pressed or released.

Example:

```
void mouse(int button, int state, int x, int y) {
  switch( mouse_button ) {
    case GLUT_LEFT_BUTTON:
      if( state == GLUT_DOWN ) {
        fprintf(stderr,"left mouse button down.\n") ;
      }
      break ;
  }
}
```

GLUT Callback Functions

`glutReshapeFunc()` set the reshape callback function. This function is called when the window is resized.

Example:

```
void reshape(int w, int h) {
  glViewport(0,0,w,h) ;
}
```

The x *&* y variables contain new width and height of the window.

GLUT Callback Functions

`glutDisplayFunc()` set the display callback function. This function is called when the window contents need to be redrawn.

Example:

```
void display(void) {
  glBegin(GL_POINTS) ;
    glVertex2i(0,0) ;
  glEnd() ;
}
```

GLUT Callback Functions

`glutIdleFunc()` set the idle callback function. This function is continuously called when events are not being recieved.

Example:

```
void idle(void) {
  time++ ;
  glutPostRedisplay() ;
}
```

Increment the variable `time` and tell glut to call the `glutDisplayFunc` at its next opportunity.

GLUT Callback Functions

`glutTimerFunc()` set the timer callback function. This function is triggers a one-time call back after a specified number of milliseconds.

Example:

```
glutTimerFunc( 150, animate, 3) ;
...
void animate(int type) {
  if(type == 3)
       fprintf(stderr,"animate object type 3\n") ;
}
```

The `animate()` function will be called with the argument `3` 150 millisections after the `glutTimerFunc()` is executed.