

# Display Lists

## Display Lists in OpenGL

Display lists are a mechanism for improving performance of interactive OpenGL applications.

A *display list* is a group of OpenGL commands that have been stored for later execution.

In OpenGL display lists are not editable. Once they are created they may not be modified. To change a display list it must be deleted and recreated. Thus display lists are akin to a form of cache.

In immediate mode, the default, OpenGL commands are executed immediately.

When using display lists OpenGL commands are stored for later use, and is a different approach.

OpenGL is based on a client-server model. The computer on which an application (client) is running may be different to the computer which displays the graphics (server).

The use of display lists becomes particularly crucial when the client and server are on different computers.

---

## Display Lists

---

Display lists act like named caches of groups of OpenGL commands. They require storage space.



## Circle Example

Say we want to draw a circle by using a polygonal approximation. We could do the following:

```
drawCircle()
{
    GLint i;
    GLfloat cosine, sine;
    glBegin(GL_POLYGON);
        for(i=0;i<100;i++){
            cosine=cos(i*2*PI/100.0);
            sine=sin(i*2*PI/100.0);
            glVertex2f(cosine,sine);
        }
    glEnd();
}
```

However every time we draw a circle because all the computations are made afresh. The trigonometry functions in particular are expensive.

---

We could store the results in an array.

```
drawCircle()
{
    GLint i;
    GLfloat cosine, sine;
    static GLfloat circcoords[100][2];
    static GLint inited=0;

    if(inited==0){
        inited=1;
        for(i=0;i<100;i++){
            circcoords[i][0]=cos(i*2*PI/100.0);
            circcoords[i][1]=sin(i*2*PI/100.0);
        }
    }

    glBegin(GL_POLYGON);
        for(i=0;i<100;i++)
            glVertex2fv(&circcoords[i][0]);
    glEnd();
}
```

There still remains a slight overhead with the above.

Using a display list we do the following.

---

## Display Lists

---

```
/*
 * Copyright (c) 1993-1997, Silicon Graphics, Inc.
 * ALL RIGHTS RESERVED
 * Permission to use, copy, modify, and distribute this software for
 * any purpose and without fee is hereby granted, provided that the above
 * copyright notice appear in all copies and that both the copyright notice
 * and this permission notice appear in supporting documentation, and that
 * the name of Silicon Graphics, Inc. not be used in advertising
 * or publicity pertaining to distribution of the software without specific,
 * written prior permission.
 *
 * OpenGL(R) is a registered trademark of Silicon Graphics, Inc.
 */

/*
 * list.c
 * This program demonstrates how to make and execute a
 * display list. Note that attributes, such as current
 * color and matrix, are changed.
 */
#include <GL/glut.h>
#include <stdlib.h>

GLuint listName;

static void init (void)
{
    listName = glGenLists (1);
    glNewList (listName, GL_COMPILE);
        glColor3f (1.0, 0.0, 0.0); /* current color red */
        glBegin (GL_TRIANGLES);
            glVertex2f (0.0, 0.0);
            glVertex2f (1.0, 0.0);
            glVertex2f (0.0, 1.0);
        glEnd ();
        glTranslatef (1.5, 0.0, 0.0); /* move position */
    glEndList ();
    glShadeModel (GL_FLAT);
}
```

---

## Display Lists

---

```
static void drawLine (void)
{
    glBegin (GL_LINES);
    glVertex2f (0.0, 0.5);
    glVertex2f (15.0, 0.5);
    glEnd ();
}

void display(void)
{
    GLuint i;

    glClear (GL_COLOR_BUFFER_BIT);
    glColor3f (0.0, 1.0, 0.0); /* current color green */
    for (i = 0; i < 10; i++) /* draw 10 triangles */
        glCallList (listName);
    drawLine (); /* is this line green? NO! */
                /* where is the line drawn? */
    glFlush ();
}

void reshape(int w, int h)
{
    glViewport(0, 0, w, h);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    if (w <= h)
        gluOrtho2D (0.0, 2.0, -0.5 * (GLfloat) h/(GLfloat) w,
                    1.5 * (GLfloat) h/(GLfloat) w);
    else
        gluOrtho2D (0.0, 2.0 * (GLfloat) w/(GLfloat) h, -0.5, 1.5);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
}

void keyboard(unsigned char key, int x, int y)
{
    switch (key) {
        case 27:
            exit(0);
            break;
    }
}
```

---

## Display Lists

---

```
}

/* Main Loop
 * Open window with initial window size, title bar,
 * RGBA display mode, and handle input events.
 */
int main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(650, 50);
    glutCreateWindow(argv[0]);
    init ();
    glutReshapeFunc (reshape);
    glutDisplayFunc (display);
    glutKeyboardFunc (keyboard);
    glutMainLoop();
    return 0;
}
```

A display list contains only OpenGL commands. The function calls to `sin` and `cos` are not saved as part of the display list.

---

## Display-List Design Philosophy

There are several situations where display lists may be advantageous:

1. Matrix operations.
2. Raster bitmaps and images.
3. Lights, material properties and lighting models.
4. Textures.
5. Polygon stipple patterns.

Display lists have small overhead. They can be used without concern that they are introducing a (major) source of time inefficiency. They must, however, be stored and may introduce a source of memory usage.

Different implementations of OpenGL may pay more attention to optimising one aspect of OpenGL than another - display lists included.

---

## Creating and Using Display Lists

The command `glNewList` specifies the start of a display list.

```
void glNewList (GLuint list, GLenum mode);
```

OpenGL routines that are called subsequently (until `glEndList()` is called to end the display list) are stored in a display list.

The list parameter is a unique positive integer that identifies the display list. The possible values for the mode parameter are `GL_COMPILE` and `GL_COMPILE_AND_EXECUTE`.

To mark the end of a display list use `void glEndList (void)`

---

## What's Stored in a Display List

Only the expression values for OpenGL commands are stored in a display list. Subsequent changes to variables, etc., do not change the values in the display list.

```
GLfloat color_vector[3]={0.0,0.0,0.0};
glNewList(1, GL_COMPILE);
    glColor3fv(color_vector);
glEndList();
color_vector[0]=1.0;
```

The change in the `color_vector` array is not made in the display list.

There are some OpenGL commands which cannot be stored and executed from within a display list: `glDeleteLists()`, `glIsEnabled()`, `glFeedbackBuffer()`, `glIsList()`, `glFinish()`, `glPixelStore()`, `glFlush()`, `glReadPixels()`, `glGenLists()`, `glRenderMode()`, `glGet*()`, `glSelectBuffer()`.

---

## Hierarchical Display Lists

Display lists can contain other display lists - in other words display lists can be hierarchical.

```
glNewList(listIndex, GL_COMPILE);  
    glCallList(handlebars);  
    glCallList(frame);  
    glTranslatef(1.0,0.0,0.0);  
    glCallList(wheel);  
    glTranslatef(3.0,0.0,0.0);  
    glCallList(wheel);  
glEndList();
```

The limit on the depth (to avoid infinite recursion) is at least 64.

---

## Executing a Display List

To execute a display list call `glCallList`.

Some commands, such as `glColor` change the value of OpenGL state variables.

If these commands are called inside a display list then their effect will persist after the display list has finished.

To isolate the effect of OpenGL commands which change the OpenGL state to just the display list use `glPushAttrib` and `glPopAttrib`.

```
glNewList(listIndex, GL_COMPILE);
    glPushMatrix();
    glPushAttrib(GL_CURRENT_BIT);
        glColor3f(1.0, 0.0, 0.0);
        glBegin(GL_POLYGON);
            glVertex2f(0.0,0.0);
            glVertex2f(1.0,0.0);
            glVertex2f(0.0,1.0);
        glEnd();
    glTranslatef(1.5,0.0,0.0);
    glPopAttrib();
```

---

Display Lists

---

```
    glPopMatrix();  
glEndList();
```

---

## Display List Indices

Display lists have positive integers as identifiers.

When using display lists there is a need to manage their indices. For instance, when new display lists are created only indices which are not currently in use should be used.

OpenGL provides two mechanisms for managing display list indices:

- Using `glGenLists` to request a range of contiguous, unallocated indices. The range may be just one.
  - Use `glIsList` to check if an index is currently in use.
-

```
listIndex=glGenLists(1);  
if(listIndex!=0) {  
    glNewList(listIndex, GL_COMPILE);  
    ...  
    glEndList();  
}
```

The command `glDeleteLists` allows deletion of a contiguous range of display lists.

---