**DISPLAY LISTS IN OPENGL**

**Introduction**

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

- Most OpenGL commands can be either stored in a display list or issued in **immediate mode**.
For example, suppose you want to draw a circle with 100 line segments

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();
}
This method is terribly inefficient because the trigonometry has to be performed each time the circle is rendered. Save the coordinates in a table:

```c
drawCircle()
{
    GLint i;
    GLfloat cosine, sine;
    static GLfloat circoords[100][2];
    static GLInt inited=0;
    if(inited==0){
        inited=1;
        for(i=0;i<100;i++){
            circoords[i][0]=cos(i*2*PI/100.0);
            circoords[i][1]=sin(i*2*PI/100.0);
        }
    }
    glBegin(GL_POLYGON);
    for(i=0;i<100;i++)
        glVertex2fv(&circoords[i][0]);
    glEnd();
}
```
• Draw the circle once and have OpenGL remember how to draw it for later use.

```c
#define MY_CIRCLE_LIST 1

buildCircle() {
    GLint i;
    GLfloat cosine, sine;
    glEndList (MY_CIRCLE_LIST, GL_COMPILE);
    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();
    glEndList();
}

MY_CIRCLE_LIST is an integer index that uniquely identifies this display list.
You can execute the display list later with this glCallList() command:  glCallList(MY_CIRCLE_LIST);
```
• A display list contains only OpenGL calls.

• The coordinates and other variables are evaluated and copied into the display list when the list is compiled.

• You can delete a display list and create a new one, but you can’t edit an existing display list.

• Display lists reside with the server and network traffic is minimized. Matrix computations, lighting models, textures, etc.

• Display List **disadvantages**: large storage; immutability of the contents of a display list.
If `glLoadMatrix(M); glCallList(1);` are called all the time, it is better to store the matrix in the display list:

```c

glNewList(1, GL_COMPILE);
    glLoadMatrix(M); draw_some_geometry();
glEndList();

glCallList(1);
```

Use a Display List: `list.c`

```c

glNewList (listName, GL_COMPILE);
    glColor3f(1.0, 0.0, 0.0);
    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);
    glEndList ();
    glShadeModel (GL_FLAT);
```
void display(void)
{
    GLuint i;
    glClear (GL_COLOR_BUFFER_BIT);
    glColor3f(0.0, 1.0, 0.0);
    for (i = 0; i < 10; i++)
        glCallList (listName);
    drawLine (); /* color red; affected by the 10 translate */
    glFlush ();
}

Constants are stored and won’t change

GLfloat color_vector[3]={0.0,0.0,0.0};
glNewList(1,GL_COMPILE);
    glColor3fv(color_vector);
    glEndList();
    color_vector[0]=1.0; // color will be black if you use the display list
Use `glPushMatrix()` to save the current transformation matrix and `glPopMatrix()` to restore.

```c
void display(void)
{
    GLint i;
    glClear (GL_COLOR_BUFFER_BIT); glColor3f(0.0, 1.0, 0.0);
    for (i = 0; i < 10; i++)  glCallList (listIndex);
    drawLine ();     glFlush ();
}
```

The code below would draw a green, untranslated line.

```c
void display(void)
{
    GLint i;
    glClear (GL_COLOR_BUFFER_BIT); glColor3f(0.0, 1.0, 0.0);
    for (i = 0; i < 10; i++)  glCallList (listIndex);
    drawLine ();     glFlush ();
}
```
Hierarchical Display Lists

• You can create a hierarchical display list, a display list that executes another display list.

• Useful for an object that’s made of components which are used more than once.

```c
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();
```
Editable Display Lists

• Example editable display list: To render the polygon, call display list number 4. To edit a vertex, you need only recreate the single display list corresponding to that vertex.

```c
    glNewList(1, GL_COMPILE);
    glVertex3f(v1);
    glEndList();
    glNewList(2, GL_COMPILE);
    glVertex3f(v2);
    glEndList();
    glNewList(3, GL_COMPILE);
    glVertex3f(v3);
    glEndList();

    glNewList(4, GL_COMPILE);
    glBegin(GL_POLYGON);
    glCallList(1); glCallList(2); glCallList(3);
    glEnd();
    glEndList();
```
Managing Display List Indices

List Indices can be automatically generated:

```c
listIndex=glGenLists(1);
if(listIndex!=0) {
    glNewList(listIndex,GL_COMPILE);
    ...
    glEndList();
}
```
Fonts and Display Lists

Fonts are array of characters. Different font types are saved in the same bitmap arrays with different offset that is added to the indices

• glGenLists(GLsizei range); Allocates range number of continuous list indices (for different characters.) It returns the index that marks the beginning of the list indices

• glListBase(GLuint base); the offset added to the display list indices in glCallLists below; it has no effect on glCallList. It is used as an offset to different font types

• glCallLists(GLsizei n, GLenum type, const GLvoid *lists); to call all display lists that draw individual characters.

    n is the # of characters to be drawn, type is GL_BYTE for the font, and lists is an array of character codes;