



## Display Listen in *OpenGL*

Manfred Brill

Oktober 2005

# 1 Display-Listen

Sie haben bisher OpenGL nur als *Immediate Mode*-API kennengelernt. Darunter versteht man die Tatsache, dass in Normalfall die *OpenGL*-Befehle sofort ausgeführt werden. Geben Sie Eckenkoordinaten oder eine Farbe an, dann werden diese Daten *sofort* zur Grafik-Hardware, der *Server-Seite* geschickt.

OpenGL bietet schon seit langer Zeit auch die Möglichkeit, im *retained Mode* zu arbeiten. Darunter versteht man die Eigenschaft, dass die angegebenen Befehle erst einmal im Speicherbereich der Anwendung, der *Client-Seite* abgelegt werden und erst zu einem vom System oder von der Anwendung definierten Zeitpunkt ausgeführt werden. Szenengraph-basierte APIs wie *Open Inventor* oder *Java3D* folgen meist diesem Paradigma.

*OpenGL* bietet mit den *Display Listen* die Möglichkeit, Kommandos beim Client abzulegen. Diese Technik war früher, bei Silicon Graphics, nicht sonderlich performant. Inzwischen gibt es insbesondere im PC-Bereich Treiber, die diese Listen effizient ablegen, so dass häufig mit *Display Listen* ein Performance-Gewinn erzielt werden kann. *OpenGL Display Listen* sind ein *Cache-Speicher*; eine einmal erzeugte Liste kann *nicht* verändert werden. Nach der Definition der Liste befindet sich diese auf der *Server-Seite*!

Sie erzeugen *Display Listen* mit der Funktion

```
GLuint glGenLists(GLsizei range);
```

Mit dem Parameter `range` geben Sie die Anzahl der zu erzeugenden Listen an. Der Rückgabewert der Funktion ist der Index der ersten erzeugten Liste. Wird 0 als Index zurückgegeben ist die „beantragte“ Anzahl von Listen zu groß; oder Sie haben einen nicht-positiven `range` angegeben. In Abbildung 1 sehen Sie das Ergebnis eines Programms, das mit Hilfe zweier *Display-Listen* arbeitet.

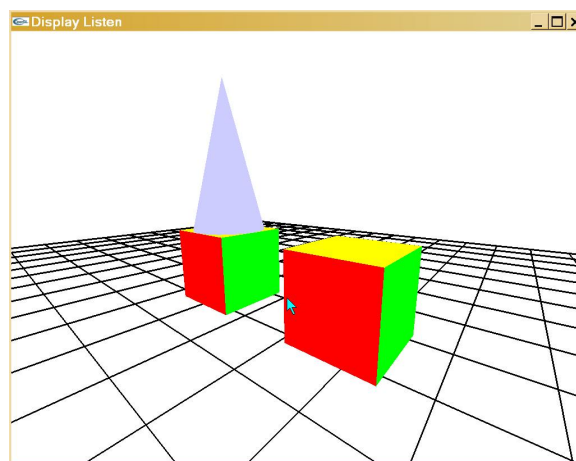


Abbildung 1: Zwei Display-Listen in OpenGL

Der Würfel wird als *Display-Liste* vereinbart und bei der Figur links wiederverwendet. Häufig werden Sie Listen in der *init*-Funktion erzeugen, wie im folgenden Quelltext:

```
COLORCUBE = glGenLists(2);  
FIGUR = COLORCUBE+1;
```

Die Definition der *Display Listen* erfolgt mit den Funktionen

```
glNewList(GLuint list, GLenum mode);  
    // Hier OpenGL-Kommandos einfügen, die in der Liste auftreten sollen  
glEndList();
```

Der erste Parameter `list` ist der Index einer geöffneten Liste. Mit `mode` wird gesteuert, ob die angegebenen Funktionen in der Liste nur gespeichert oder darüberhinaus sofort ausgeführt werden können. Mit `GL_COMPILE` werden die Funktionen nur abgespeichert; `GL_COMPILE_AND_EXECUTE` sorgt dafür, dass darüberhinaus die Funktionen *direkt* ausgeführt werden.

**Tip:**

Enthält eine Liste *OpenGL Arrays*, dann wirken sich Veränderungen der Array-Werte *nicht* auf die *Display Liste* aus!

Nicht alle *OpenGL*-Funktionen werden in einer *Display Liste* gespeichert. Beispielsweise werden `glEnable`, `glEnableClientState` oder Abfragen mit `glGet*` direkt ausgeführt und *nicht* gespeichert.

Ausgeführt wird eine *Display Liste* durch

```
void glCallList(GLuint list);
```

Hier das Beispiel für die beiden Display-Listen in Abbildung 1. Sie sehen, dass Display-Listen auf Listen zurückgreifen können. Diese Listen müssen noch nicht existieren!

```
glNewList(COLORCUBE, GL_COMPILE);
glBegin(GL_QUADS);
    // Unterseite
    glColor3f(1.0, 0.0, 1.0); // Magenta
    glNormal3f(0.0, -1.0, 0.0);
    glVertex3f(-1.0, -1.0, 1.0);
    glVertex3f(-1.0, -1.0, -1.0);
    glVertex3f(1.0, -1.0, -1.0);
    glVertex3f(1.0, -1.0, 1.0);
    // Oberseite
    glColor3f(1.0, 1.0, 0.0); // Gelb
    glNormal3f(0.0, 1.0, 0.0);
    glVertex3f(-1.0, 1.0, 1.0);
    glVertex3f(1.0, 1.0, 1.0);
    glVertex3f(1.0, 1.0, -1.0);
    glVertex3f(-1.0, 1.0, -1.0);
    // Vorderseite
    glColor3f(1.0, 0.0, 0.0); // Rot
    glNormal3f(0.0, 0.0, 1.0);
    glVertex3f(-1.0, 1.0, 1.0);
    glVertex3f(-1.0, -1.0, 1.0);
    glVertex3f(1.0, -1.0, 1.0);
    glVertex3f(1.0, 1.0, 1.0);
    // Linke Seite
    glColor3f(0.0, 1.0, 1.0); // Cyan
    glNormal3f(-1.0, 0.0, 0.0);
    glVertex3f(-1.0, 1.0, 1.0);
    glVertex3f(-1.0, 1.0, -1.0);
    glVertex3f(-1.0, -1.0, -1.0);
    glVertex3f(-1.0, -1.0, 1.0);
    // Rechte Seite
    glColor3f(0.0, 1.0, 0.0); // Grün
    glNormal3f(1.0, 0.0, 0.0);
    glVertex3f(1.0, 1.0, 1.0);
    glVertex3f(1.0, -1.0, 1.0);
    glVertex3f(1.0, -1.0, -1.0);
    glVertex3f(1.0, 1.0, -1.0);
    // Rückseite
```

```

        glColor3f(0.0, 0.0, 1.0); // Blau
        glNormal3f(0.0, 0.0, -1.0);
        glVertex3f(-1.0, 1.0, -1.0);
        glVertex3f(1.0, 1.0, -1.0);
        glVertex3f(1.0, -1.0, -1.0);
        glVertex3f(-1.0, -1.0, -1.0);
    glEnd();
glEndList();

// Zweite Display-Liste, verwendet die Liste COLORCUBE
glNewList(FIGUR, GL_COMPILE);
    glPushMatrix();
        glCallList(COLORCUBE);
        glTranslatef(0.0, 1.0, 0.0);
        glRotatef(-90.0, 1.0, 0.0, 0.0);
        glColor3f(0.8, 0.8, 1.0);
        glutSolidCone(1, 4, 20, 20);
    glPopMatrix();
glEndList();

```

Die Spezifikation sieht vor, dass die maximale Tiefe bei der hierarchischen Verwendung von Listen mindestens 64 beträgt. Hier nochmal ein Beispiel für die hierarchische Verwendung von Listen:

```

glNewList(index, GL_COMPILE);
    glCallList(l1);
    glCallList(l2);
    glTranslatef(1.0, 0.0, 0.0);
    glCallList(l1);
glEndListI();

```

Ob ein Index eine Liste repräsentiert können Sie mit

```
GLboolean glIsLlist(GLuint list);
```

abfragen; der Rückgabewert ist `GL_TRUE`, falls Sie einen gültigen Listenindex übergeben haben. Mit

```
void glDeleteLists(GLuint list, GLsize range);
```

können Sie eine Menge von Listen löschen; beginnend beim Index `list`.

Es gibt auch die Möglichkeit, *mehrere* Listen auszuführen. Dazu gibt es die Funktion `glListBase` und `glCallLists`. Mehr dazu und zum ganzen Thema *Display Listen* finden Sie in [WND00] und [WS04].

## 2 Ein Performance-Vergleich

Um zu testen, ob die Verwendung von *Display Listen* einen Vorteil in der Performance bringen kann das folgende Programm verwendet werden. Klar ist, dass man eine möglichst große Anzahl von Objekten verwenden sollte, um wirklich einen Vergleich anstellen zu können.

In der `init`-Funktion werden eine größere Menge von Körpern in *Display Listen* gelegt:

```

sphere = glGenLists(10);
cube = sphere+1;
torus = cube+1;
ico = torus+1;
octa = ico+1;
tetra = octa+1;
dode = tetra+1;

```

```
cone = dode+1;
teapot = cone+1;
SZENE = teapot+1;

glNewList(sphere, GL_COMPILE);
    drawSphere();
glEndList();

glNewList(cube, GL_COMPILE);
    drawCube();
glEndList();

glNewList(torus, GL_COMPILE);
    drawTorus();
glEndList();

glNewList(ico, GL_COMPILE);
    drawIcosahedron();
glEndList();

glNewList(octa, GL_COMPILE);
    drawOctahedron();
glEndList();

glNewList(tetra, GL_COMPILE);
    drawTetrahedron();
glEndList();

glNewList(dode, GL_COMPILE);
    drawDodecahedron();
glEndList();

glNewList(cone, GL_COMPILE);
    drawCone();
glEndList();

glNewList(teapot, GL_COMPILE);
    drawTeapot();
glEndList();

glNewList(SZENE, GL_COMPILE);
    /* Die Listen ausgeben          */
    glPushMatrix();
    glTranslatef(2.0, 0.0, 0.0);
    glCallList(sphere);
    glPopMatrix();

    glPushMatrix();
    glTranslatef(2.0, 0.0, 4.0);
    glCallList(cube);
    glPopMatrix();

    glPushMatrix();
    glTranslatef(2.0, 0.0, -4.0);
    glCallList(torus);
    glPopMatrix();

    glPushMatrix();
```

```
    glTranslatef(-2.0, 0.0, -4.0);
    glCallList(ico);
    glPopMatrix();

    glPushMatrix();
    glTranslatef(-2.0, 0.0, 4.0);
    glCallList(octa);
    glPopMatrix();

    glPushMatrix();
    glTranslatef(-2.0, 0.0, 8.0);
    glCallList(tetra);
    glPopMatrix();

    glPushMatrix();
    glTranslatef(2.0, 0.0, 8.0);
    glCallList(dode);
    glPopMatrix();

    glPushMatrix();
    glTranslatef(-2.0, 0.0, 0.0);
    glCallList(cone);
    glPopMatrix();

    glPushMatrix();
    glTranslatef(0.0, 0.0, 12.0);
    glCallList(teapot);
    glPopMatrix();

    glPushMatrix();
    glTranslatef(0.0, 0.0, -8.0);
    glCallList(teapot);
    glPopMatrix();
glEndList();
}
```

In der `display`-Funktion wird dann, abhängig von der globalen Variable `LISTS` die Szene mit `glCallList(SZENE)` aufgerufen; oder mit einem Quelltext wie bei der Definition der Listen die Ausgabe *ohne* Listen durchgeführt. Wenn Sie den Shortcut F2 für die Ausgabe der Frame-Rate auf dem Fenstertitel verwenden und einen Shortcut, beispielsweise `l`, definieren, um `LISTS` umzuschalten, erhält man einen Eindruck, ob Listen auf dem verwendeten Rechner einen Vorteil bieten. Um einen Eindruck zu gewinnen können Sie mit `p` eine Rotation aktivieren; man fliegt durch die eine ständige Veränderung des Azimuth-Winkels um die Szene.

In den Abbildung 2 und 3 sehen Sie das Ergebnis auf einem IBM ThinkPad T42p mit einer ATI FireGL.

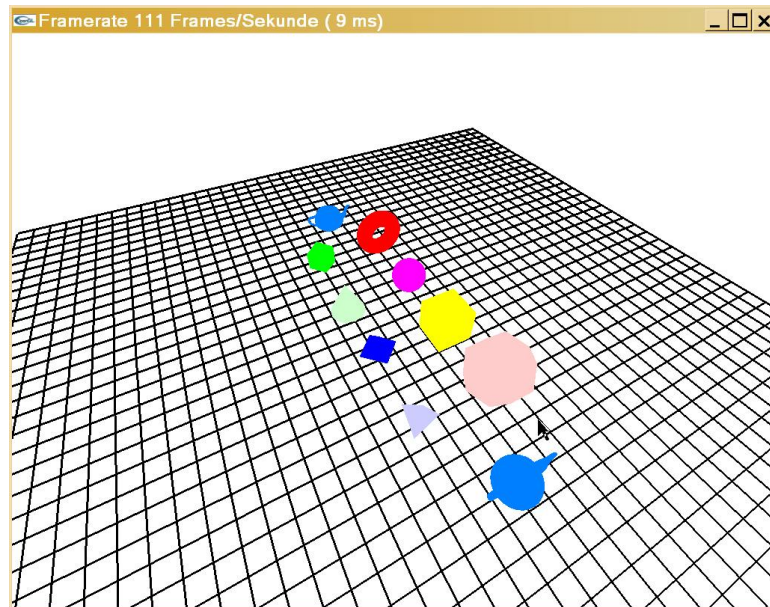


Abbildung 2: Das Beispielprogramm mit abgeschalteten *Display Listen*

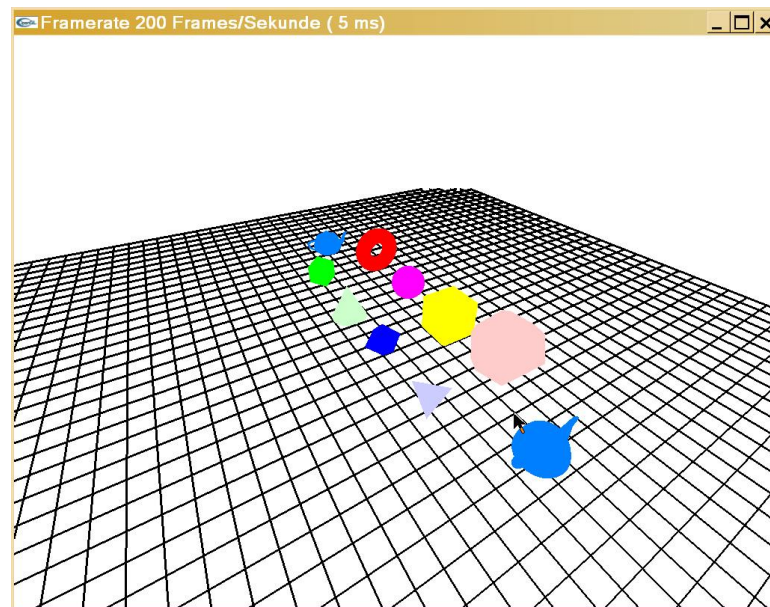


Abbildung 3: Das Beispielprogramm mit *Display Listen*

## Literatur

- [WND00] WOO, MASON, NEIDER, JACKIE und DAVIS, TOM: *OpenGL Programming Guide (2nd Ed.) – The Official Guide to Learning OpenGL*. Addison-Wesley, 2000.
- [WS04] WRIGHT, RICHARD und SWEET, MICHAEL: *OpenGL Superbible*. Sams, 2004.