OpenGL

Yihsiang Liow (December 2, 2011)

Contents

1 Fedora and g++

First install free glut:

```
yum -y install freeglut-devel
```

Now for the test. Create this file `a.cpp`:

```
#include <GL/glut.h>

void display()
{
    glClear(GL_COLOR_BUFFER_BIT);
    glFlush();
}

int main(int argc, char ** argv)
{
    glutInit(&argc, argv);
    glutInitWindowPosition(300, 100);
    glutInitWindowSize(400, 400);
    glutInitDisplayMode(GLUT_DEPTH | GLUT_SINGLE | GLUT_RGBA);
    glutCreateWindow("opengl!!!");

    glutDisplayFunc(display);
    glutMainLoop();

    return 0;
}
```

Compile:

```
gcc a.cpp -lGL -lGLU -lglut
```

or

```
g++ a.cpp -lGL -lGLU -lglut
```

and run:

```
./a.out
```

Minimal makefile:

```
exe: a.cpp
    g++ a.cpp -lGL -lGLU -lglut

run:
    ./a.out

clean:
    rm a.out
```

2 Clear Color

```
1  #include <GL/glut.h>
2
3  void display()
4  {
5      glClearColor(1.0f, 1.0f, 0.0f, 0.0f); // RGBA
6      glClear(GL_COLOR_BUFFER_BIT);
7      glFlush();
8  }
9
10 int main(int argc, char ** argv)
11 {
12     glutInit(&argc, argv);
13     glutInitWindowPosition(100, 100);
14     glutInitWindowSize(200, 200);
15     glutInitDisplayMode(GLUT_DEPTH | GLUT_SINGLE | GLUT_RGBA);
16     glutCreateWindow("test");
17
18     glutDisplayFunc(display);
19     glutMainLoop();
20
21     return 0;
22 }
```

3 Points

```
#include <ctime>
#include <cstdlib>
#include <GL/glut.h>

void init()
{
    glClearColor(1.0f, 1.0f, 1.0f, 0.0f);
    glMatrixMode(GL_PROJECTION);
    gluOrtho2D(0.0f, 200.0f, 0.0f, 200.0f);
}

void display()
{
    glClear(GL_COLOR_BUFFER_BIT);

    glColor3f(0.0f, 0.0f, 0.0f);
    glBegin(GL_POINTS);
    for (int i = 0; i < 1000; i++)
    {
        glVertex2i(rand() % 100, rand() % 100);
    }
    glEnd();

    glFlush();
}

int main(int argc, char ** argv)
{
    srand((unsigned int) time(NULL));

    glutInit(&argc, argv);
    glutInitWindowPosition(100, 100);
    glutInitWindowSize(200, 200);
    glutInitDisplayMode(GLUT_DEPTH | GLUT_SINGLE | GLUT_RGBA);
    glutCreateWindow("test");

    glutDisplayFunc(display);
    init();
    glutMainLoop();
```

```
40
41      return 0;
42  }
```

What if you move

```
glColor3f(0.0f, 0.0f, 0.0f);
```

into the `init()` function?

4 Points: Size

```
1  #include <ctime>
2  #include <cstdlib>
3  #include <GL/glut.h>
4
5  void init()
6  {
7      glClearColor(1.0f, 1.0f, 1.0f, 0.0f);
8      glMatrixMode(GL_PROJECTION);
9      gluOrtho2D(0.0f, 400.0f, 0.0f, 400.0f);
10 }
11
12 void display()
13 {
14     glClear(GL_COLOR_BUFFER_BIT);
15
16     glColor3f(0.0f, 0.0f, 0.0f);
17     glPointSize(5.0f);
18     glBegin(GL_POINTS);
19     for (int i = 0; i < 1000; i++)
20     {
21         glVertex2i(rand() % 400, rand() % 400);
22     }
23     glEnd();
24
25
26     glFlush();
27 }
28
29 int main(int argc, char ** argv)
30 {
31     srand((unsigned int) time(NULL));
32
33     glutInit(&argc, argv);
34     glutInitWindowPosition(100, 100);
35     glutInitWindowSize(400, 400);
36     glutInitDisplayMode(GLUT_DEPTH | GLUT_SINGLE | GLUT_RGBA);
37     glutCreateWindow("test");
38
39     glutDisplayFunc(display);
```

```
40      init();
41      glutMainLoop();
42
43      return 0;
44  }
```

5 Points: Antialiasing

```
#include <ctime>
#include <cstdlib>
#include <GL/glut.h>

void init()
{
    glClearColor(1.0f, 1.0f, 1.0f, 0.0f);
    glMatrixMode(GL_PROJECTION);
    gluOrtho2D(0.0f, 400.0f, 0.0f, 400.0f);
    glColor3f(0.0f, 0.0f, 0.0f);
    glPointSize(5.0f);
    glEnable(GL_POINT_SMOOTH);
}

void display()
{
    glClear(GL_COLOR_BUFFER_BIT);

    glBegin(GL_POINTS);
    for (int i = 0; i < 1000; i++)
    {
        glVertex2i(rand() % 400, rand() % 400);
    }
    glEnd();

    glFlush();
}

int main(int argc, char ** argv)
{
    srand((unsigned int) time(NULL));

    glutInit(&argc, argv);
    glutInitWindowPosition(100, 100);
    glutInitWindowSize(400, 400);
    glutInitDisplayMode(GLUT_DEPTH | GLUT_SINGLE | GLUT_RGBA);
    glutCreateWindow("test");

    glutDisplayFunc(display);
```

```
40      init();
41      glutMainLoop();
42
43      return 0;
44  }
```

6 Lines

```
#include <ctime>
#include <cstdlib>
#include <GL/glut.h>

void init()
{
    glClearColor(1.0f, 1.0f, 1.0f, 0.0f);
    glMatrixMode(GL_PROJECTION);
    gluOrtho2D(0.0f, 400.0f, 0.0f, 400.0f);
    glColor3f(0.0f, 0.0f, 1.0f);
}

void display()
{
    glClear(GL_COLOR_BUFFER_BIT);

    glBegin(GL_LINES);
    glVertex2i(0, 0);
    glVertex2i(100, 100);
    glVertex2i(200, 0);
    glVertex2i(200, 300);
    glEnd();

    glFlush();
}

int main(int argc, char ** argv)
{
    srand((unsigned int) time(NULL));

    glutInit(&argc, argv);
    glutInitWindowPosition(100, 100);
    glutInitWindowSize(400, 400);
    glutInitDisplayMode(GLUT_DEPTH | GLUT_SINGLE | GLUT_RGBA);
    glutCreateWindow("test");

    glutDisplayFunc(display);
    init();
    glutMainLoop();
```

```
40
41      return 0;
42  }
```

7 Lines: Width

```
#include <ctime>
#include <cstdlib>
#include <GL/glut.h>

void init()
{
    glClearColor(1.0f, 1.0f, 1.0f, 0.0f);
    glMatrixMode(GL_PROJECTION);
    gluOrtho2D(0.0f, 400.0f, 0.0f, 400.0f);
    glColor3f(0.0f, 0.0f, 1.0f);
    glLineWidth(10.0f);
}

void display()
{
    glClear(GL_COLOR_BUFFER_BIT);

    glBegin(GL_LINES);
    glVertex2i(0, 0);
    glVertex2i(100, 200);
    glVertex2i(200, 0);
    glVertex2i(200, 300);
    glEnd();

    glFlush();
}

int main(int argc, char ** argv)
{
    srand((unsigned int) time(NULL));

    glutInit(&argc, argv);
    glutInitWindowPosition(100, 100);
    glutInitWindowSize(400, 400);
    glutInitDisplayMode(GLUT_DEPTH | GLUT_SINGLE | GLUT_RGBA);
    glutCreateWindow("test");

    glutDisplayFunc(display);
    init();
```

```
40      glutMainLoop();
41
42      return 0;
43  }
```

8 Lines: Antialiasing

```
#include <ctime>
#include <cstdlib>
#include <GL/glut.h>

void init()
{
    glClearColor(1.0f, 1.0f, 1.0f, 0.0f);
    glMatrixMode(GL_PROJECTION);
    gluOrtho2D(0.0f, 400.0f, 0.0f, 400.0f);
    glColor3f(0.0f, 0.0f, 1.0f);
    glEnable(GL_LINE_SMOOTH);
}

void display()
{
    glClear(GL_COLOR_BUFFER_BIT);

    glBegin(GL_LINES);
    glVertex2i(0, 0);
    glVertex2i(100, 100);
    glVertex2i(200, 0);
    glVertex2i(200, 300);
    glEnd();

    glFlush();
}

int main(int argc, char ** argv)
{
    srand((unsigned int) time(NULL));

    glutInit(&argc, argv);
    glutInitWindowPosition(100, 100);
    glutInitWindowSize(400, 400);
    glutInitDisplayMode(GLUT_DEPTH | GLUT_SINGLE | GLUT_RGBA);
    glutCreateWindow("test");

    glutDisplayFunc(display);
    init();
```

```
40      glutMainLoop();
41
42      return 0;
43  }
```

9 Line Strips

```
1  #include <ctime>
2  #include <cstdlib>
3  #include <GL/glut.h>
4
5  void init()
6  {
7      glClearColor(1.0f, 1.0f, 1.0f, 0.0f);
8      glMatrixMode(GL_PROJECTION);
9      gluOrtho2D(0.0f, 400.0f, 0.0f, 400.0f);
10     glColor3f(0.0f, 0.0f, 1.0f);
11 }
12
13 void display()
14 {
15     glClear(GL_COLOR_BUFFER_BIT);
16
17     glBegin(GL_LINE_STRIP);
18     glVertex2i(0, 0);
19     glVertex2i(100, 50);
20     glColor3f(1.0f, 0.0f, 1.0f);
21     glVertex2i(200, 0);
22     glColor3f(0.0f, 1.0f, 1.0f);
23     glVertex2i(300, 380);
24     glEnd();
25
26     glFlush();
27 }
28
29 int main(int argc, char ** argv)
30 {
31     srand((unsigned int) time(NULL));
32
33     glutInit(&argc, argv);
34     glutInitWindowPosition(100, 100);
35     glutInitWindowSize(400, 400);
36     glutInitDisplayMode(GLUT_DEPTH | GLUT_SINGLE | GLUT_RGBA);
37     glutCreateWindow("test");
38
39     glutDisplayFunc(display);
```

```
40      init();
41      glutMainLoop();
42
43      return 0;
44  }
```

10 Line Loops

```
1  #include <ctime>
2  #include <cstdlib>
3  #include <GL/glut.h>
4
5  void init()
6  {
7      glClearColor(1.0f, 1.0f, 1.0f, 0.0f);
8      glMatrixMode(GL_PROJECTION);
9      gluOrtho2D(0.0f, 400.0f, 0.0f, 400.0f);
10     glColor3f(0.0f, 0.0f, 1.0f);
11 }
12
13 void display()
14 {
15     glViewport(0, 0, 100, 100);
16
17     glClear(GL_COLOR_BUFFER_BIT);
18
19     glBegin(GL_LINE_LOOP);
20     glVertex2i(0, 0);
21     glVertex2i(100, 50);
22     glVertex2i(200, 0);
23     glVertex2i(300, 380);
24     glEnd();
25
26     glFlush();
27 }
28
29 int main(int argc, char ** argv)
30 {
31     srand((unsigned int) time(NULL));
32
33     glutInit(&argc, argv);
34     glutInitWindowPosition(100, 100);
35     glutInitWindowSize(400, 400);
36     glutInitDisplayMode(GLUT_DEPTH | GLUT_SINGLE | GLUT_RGBA);
37     glutCreateWindow("test");
38
39     glutDisplayFunc(display);
```

```
40      init();
41      glutMainLoop();
42
43      return 0;
44  }
```

11 Triangles

```
#include <ctime>
#include <cstdlib>
#include <GL/glut.h>

void init()
{
    glClearColor(1.0f, 1.0f, 1.0f, 0.0f);
    glMatrixMode(GL_PROJECTION);
    gluOrtho2D(0.0f, 400.0f, 0.0f, 400.0f);
    glColor3f(0.0f, 1.0f, 0.0f);
}

void display()
{
    glClear(GL_COLOR_BUFFER_BIT);
    glBegin(GL_TRIANGLES);

    glColor3f(1.0f, 0.0f, 0.0f);
    glVertex2f(0.0f, 0.0f);
    glVertex2f(100.5f, 100.25f);
    // Second run: uncomment the next line
    glColor3f(0.0f, 1.0f, 0.0f);
    glVertex2f(200.0f, 100.0f);

    glColor3f(0.0f, 0.0f, 1.0f);
    glVertex2f(300.0f, 300.0f);
    glVertex2f(350.5f, 300.25f);
    glVertex2f(300.0f, 380.0f);

    glEnd();
    glFlush();
}

int main(int argc, char ** argv)
{
    srand((unsigned int) time(NULL));

    glutInit(&argc, argv);
    glutInitWindowPosition(100, 100);
```

```
40      glutInitWindowSize(400, 400);
41      glutInitDisplayMode(GLUT_DEPTH | GLUT_SINGLE | GLUT_RGBA);
42      glutCreateWindow("test");
43
44      glutDisplayFunc(display);
45      init();
46      glutMainLoop();
47
48      return 0;
49  }
```

12 Triangle Fans

```
#include <ctime>
#include <cstdlib>
#include <GL/glut.h>

void init()
{
    glClearColor(1.0f, 1.0f, 1.0f, 0.0f);
    glMatrixMode(GL_PROJECTION);
    gluOrtho2D(0.0f, 400.0f, 0.0f, 400.0f);
    glColor3f(0.0f, 1.0f, 0.0f);
}

void display()
{
    glClear(GL_COLOR_BUFFER_BIT);
    glBegin(GL_TRIANGLE_FAN);

    glColor3f(1.0f, 0.0f, 0.0f);

    glVertex2f(0.0f, 0.0f);

    glVertex2f(300.5f, 100.25f);
    glVertex2f(300.0f, 150.0f);

    // Second run: uncomment the next two lines
    //glColor3f(0.0f, 1.0f, 0.0f);
    //glVertex2f(200.0f, 300.0f);

    // Third run: uncomment the next two lines
    //glColor3f(0.0f, 0.0f, 1.0f);
    //glVertex2f(100.5f, 350.25f);

    glEnd();
    glFlush();
}

int main(int argc, char ** argv)
{
    srand((unsigned int) time(NULL));
```

```
40
41      glutInit(&argc, argv);
42      glutInitWindowPosition(100, 100);
43      glutInitWindowSize(400, 400);
44      glutInitDisplayMode(GLUT_DEPTH | GLUT_SINGLE | GLUT_RGBA);
45      glutCreateWindow("test");
46
47      glutDisplayFunc(display);
48      init();
49      glutMainLoop();
50
51      return 0;
52  }
```

13 Quadrilaterals

```
#include <ctime>
#include <cstdlib>
#include <GL/glut.h>

void init()
{
    glClearColor(1.0f, 1.0f, 1.0f, 0.0f);
    glMatrixMode(GL_PROJECTION);
    gluOrtho2D(0.0f, 400.0f, 0.0f, 400.0f);
    glColor3f(0.0f, 1.0f, 0.0f);
}

void display()
{
    glClear(GL_COLOR_BUFFER_BIT);
    glBegin(GL_QUADS);

    glColor3f(0.0f, 1.0f, 0.0f);
    glVertex2f(0.0f, 0.0f);
    glVertex2f(300.0f, 0.0f);
    glVertex2f(300.5f, 100.25f);

    glColor3f(1.0f, 0.0f, 0.0f);
    glVertex2f(400.0f, 200.0f);
    glVertex2f(300.0f, 300.0f);
    glVertex2f(250.0f, 400.0f);


    glColor3f(0.0f, 0.0f, 1.0f);
    glVertex2f(200.0f, 300.0f);
    glVertex2f(100.0f, 200.0f);

    glEnd();
    glFlush();
}

int main(int argc, char ** argv)
{
    srand((unsigned int) time(NULL));
```

```
40
41      glutInit(&argc, argv);
42      glutInitWindowPosition(100, 100);
43      glutInitWindowSize(400, 400);
44      glutInitDisplayMode(GLUT_DEPTH | GLUT_SINGLE | GLUT_RGBA);
45      glutCreateWindow("test");
46
47      glutDisplayFunc(display);
48      init();
49      glutMainLoop();
50
51      return 0;
52  }
```

14 Quadrilateral Strips

```
#include <ctime>
#include <cstdlib>
#include <GL/glut.h>

void init()
{
    glClearColor(1.0f, 1.0f, 1.0f, 0.0f);
    glMatrixMode(GL_PROJECTION);
    gluOrtho2D(0.0f, 400.0f, 0.0f, 400.0f);
    glColor3f(0.0f, 1.0f, 0.0f);
}

void display()
{
    glClear(GL_COLOR_BUFFER_BIT);
    glBegin(GL_QUAD_STRIP);

    glColor3f(0.0f, 1.0f, 0.0f);
    glVertex2f(0.0f, 0.0f);
    glVertex2f(300.5f, 0.25f);
    glVertex2f(300.0f, 100.0f);
    glVertex2f(100.0f, 400.0f);

    // Second run: uncomment next line
    //glColor3f(1.0f, 0.0f, 0.0f);
    //glVertex2f(150.0f, 350.0f);
    //glVertex2f(400.0f, 350.0f);
    //glVertex2f(400.0f, 400.0f);
    //glVertex2f(350.0f, 400.0f);

    glEnd();
    glFlush();
}

int main(int argc, char ** argv)
{
    srand((unsigned int) time(NULL));

    glutInit(&argc, argv);
```

```
40      glutInitWindowPosition(100, 100);
41      glutInitWindowSize(400, 400);
42      glutInitDisplayMode(GLUT_DEPTH | GLUT_SINGLE | GLUT_RGBA);
43      glutCreateWindow("test");
44
45      glutDisplayFunc(display);
46      init();
47      glutMainLoop();
48
49      return 0;
50  }
```

15 Polygons

```
#include <ctime>
#include <cstdlib>
#include <GL/glut.h>

void init()
{
    glClearColor(1.0f, 1.0f, 1.0f, 0.0f);
    glMatrixMode(GL_PROJECTION);
    gluOrtho2D(0.0f, 400.0f, 0.0f, 400.0f);
    glEnable(GL_POLYGON_SMOOTH);
}

void display()
{
    glClear(GL_COLOR_BUFFER_BIT);
    glBegin(GL_POLYGON);

    glColor3f(0.0f, 1.0f, 0.0f);
    glVertex2f(0.0f, 0.0f);
    glVertex2f(300.5f, 0.25f);
    glVertex2f(300.0f, 100.0f);

    // Second run: uncomment next line
    glColor3f(1.0f, 0.0f, 0.0f);
    glVertex2f(100.0f, 400.0f);

    // Third run: uncomment next line
    glColor3f(0.0f, 0.0f, 1.0f);
    glVertex2f(50.0f, 400.0f);
    glVertex2f(10.0f, 200.0f);

    glEnd();
    glFlush();
}

int main(int argc, char ** argv)
{
    srand((unsigned int) time(NULL));

```

```
40      glutInit(&argc, argv);
41      glutInitWindowPosition(100, 100);
42      glutInitWindowSize(400, 400);
43      glutInitDisplayMode(GLUT_DEPTH | GLUT_SINGLE | GLUT_RGBA);
44      glutCreateWindow("test");
45
46      glutDisplayFunc(display);
47      init();
48      glutMainLoop();
49
50      return 0;
51  }
```

16 Viewport

```
1  #include <GL/glut.h>
2
3  void init()
4  {
5      glClearColor(1.0f, 1.0f, 1.0f, 0.0f);
6      glMatrixMode(GL_PROJECTION);
7      gluOrtho2D(0.0f, 400.0f, 0.0f, 400.0f);
8      glColor3f(0.0f, 0.0f, 1.0f);
9  }
10
11 void display()
12 {
13     // Second run: uncomment next line
14     //glViewport(0, 0, 100, 100);
15     // Third run: Change the above to glViewport(0, 0, 100, 300);
16     glClear(GL_COLOR_BUFFER_BIT);
17
18     glBegin(GL_LINE_LOOP);
19     glVertex2i(0, 0);
20     glVertex2i(100, 50);
21     glVertex2i(200, 0);
22     glVertex2i(300, 380);
23     glEnd();
24
25     glFlush();
26 }
27
28 int main(int argc, char ** argv)
29 {
30     glutInit(&argc, argv);
31     glutInitWindowPosition(100, 100);
32     glutInitWindowSize(400, 400);
33     glutInitDisplayMode(GLUT_DEPTH | GLUT_SINGLE | GLUT_RGBA);
34     glutCreateWindow("test");
35
36     glutDisplayFunc(display);
37     init();
38     glutMainLoop();
39
```

```
40      return 0;
41  }
```

17 Line Stipple Pattern

```
#include <GL/glut.h>

void init()
{
    glClearColor(1.0f, 1.0f, 1.0f, 0.0f);
    glMatrixMode(GL_PROJECTION);
    gluOrtho2D(0.0f, 400.0f, 0.0f, 400.0f);
    glEnable(GL_LINE_STIPPLE);
    GLushort stipplepattern = 0xFAFA;
    glLineStipple(2, stipplepattern);

    // 0xFAFA = 1111 1010 1111 1010

    // Second run: Use 0xAAAA for pattern
}

void display()
{

    glClear(GL_COLOR_BUFFER_BIT);
    glBegin(GL_LINE);

    glColor3f(1.0f, 0.0f, 0.0f);
    glVertex2f(0.0f, 0.0f);
    glVertex2f(100.0f, 100.0f);
    glVertex2f(300.0f, 300.0f);
    glVertex2f(400.0f, 400.0f);

    glEnd();
    glFlush();
}

int main(int argc, char ** argv)
{
    glutInit(&argc, argv);
    glutInitWindowPosition(100, 100);
    glutInitWindowSize(400, 400);
    glutInitDisplayMode(GLUT_DEPTH | GLUT_SINGLE | GLUT_RGBA);
    glutCreateWindow("test");
```

```
40
41      glutDisplayFunc(display);
42      init();
43      glutMainLoop();
44
45      return 0;
46  }
```

18 Double Buffer

```
1  // Double buffering means we work with two buffers.
2  // The front buffer is used by the hardware for displaying.
3  // The back buffer is used by the software for writing.
4  //
5  // Once a buffer is completely drawn, calling a buffer swap function
6  // will then tell the system to use the drawn buffer (the back buffer)
7  // for hardware display, i.e., it becomes the front buffer while the
8  // front buffer that was used for hardware display becomes the back
9  // buffer for writing.
10 //
11 // This prevents partially drawn buffer from being displayed and is
12 // needed for smooth animation.
13
14 #include <GL/glut.h>
15
16 void init()
17 {
18     glClearColor(1.0f, 1.0f, 1.0f, 0.0f);
19     glMatrixMode(GL_PROJECTION);
20     gluOrtho2D(0.0f, 200.0f, 0.0f, 200.0f);
21 }
22
23 void display()
24 {
25     glClear(GL_COLOR_BUFFER_BIT);
26
27     glColor3f(0.0f, 0.0f, 0.0f);
28     glBegin(GL_POINTS);
29     for (int i = 0; i < 1000; i++)
30     {
31         glVertex2i(rand() % 100, rand() % 100);
32     }
33     glEnd();
34
35     // Use glutSwapBuffers() instead of glFlush()
36     glutSwapBuffers();
37 }
38
39 int main(int argc, char ** argv)
```

```
40 {
41     glutInit(&argc, argv);
42     glutInitWindowPosition(100, 100);
43     glutInitWindowSize(200, 200);
44 
45     // Use GLUT_DOUBLE instead of GLUT_SINGLE
46     glutInitDisplayMode(GLUT_DEPTH | GLUT_DOUBLE | GLUT_RGBA);
47     glutCreateWindow("test");
48 
49     glutDisplayFunc(display);
50     init();
51     glutMainLoop();
52 
53     return 0;
54 }
```

19 Text

```
90-=lk\][
\][#include <cstring>
#include <GL/glut.h>

void init()
{
    glClearColor(1.0f, 1.0f, 1.0f, 0.0f);
    glMatrixMode(GL_PROJECTION);
    gluOrtho2D(0.0f, 400.0f, 0.0f, 400.0f);
    glViewport(0, 0, 100, 100);
}

void display()
{
    glClear(GL_COLOR_BUFFER_BIT);
    glColor3f(0.0f, 0.0f, 1.0f);

    glRasterPos2i(100, 100);
    char s[] = "hello world";
    for (int i = 0; i < strlen(s); i++)
    {
        glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24, s[i]);
    }

    glRasterPos2i(100, 150);
    for (int i = 0; i < strlen(s); i++)
    {
        glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_10, s[i]);
    }

    glRasterPos2i(100, 200);
    for (int i = 0; i < strlen(s); i++)
    {
        glutBitmapCharacter(GLUT_BITMAP_HELVETICA_10, s[i]);
    }

    glRasterPos2i(100, 250);
    for (int i = 0; i < strlen(s); i++)
    {
```

```
40          glutBitmapCharacter(GLUT_BITMAP_HELVETICA_12, s[i]);
41      }
42
43      glRasterPos2i(100, 300);
44      for (int i = 0; i < strlen(s); i++)
45      {
46          glutBitmapCharacter(GLUT_BITMAP_HELVETICA_18, s[i]);
47      }
48
49      glutSwapBuffers();
50  }
51
52  int main(int argc, char ** argv)
53  {
54      glutInit(&argc, argv);
55      glutInitWindowPosition(100, 100);
56      glutInitWindowSize(400, 400);
57      glutInitDisplayMode(GLUT_DEPTH | GLUT_DOUBLE | GLUT_RGBA);
58      glutCreateWindow("test");
59      glutDisplayFunc(display);
60      init();
61      glutMainLoop();
62
63      return 0;
64  }
```

20 Keyboard

```
#include <ctime>
#include <cstdlib>
#include <GL/glut.h>

void init()
{
    glClearColor(1.0f, 1.0f, 1.0f, 0.0f);
    glMatrixMode(GL_PROJECTION);
    gluOrtho2D(0.0f, 200.0f, 0.0f, 200.0f);
    glLineWidth(5.0f);
    glEnable(GL_LINE_SMOOTH);
}

const int X0=50, Y0=50;
const int X1=150, Y1=150;

int x0=X0, y0=Y0;
int x1=X1,y1=Y1;
float r0=1.0f, g0=0.0f, b0=0.0f;
float r1=0.0f, g1=0.0f, b1=1.0f;

void display()
{
    glClear(GL_COLOR_BUFFER_BIT);

    glBegin(GL_LINES);
    glColor3f(r0, g0, b0);
    glVertex2i(x0, y0);
    glColor3f(r1, g1, b1);
    glVertex2i(x1, y1);
    glEnd();

    glFlush();
}

// This function is executed when a key such as 'a', 'A', etc.
// is pressed.
void keyboard(unsigned char key, int x, int y)
{
```

```
40      switch (key)
41      {
42          case 'a':
43          case 'A': x0--; break;
44
45          case 'd':
46          case 'D': x0++; break;
47
48          case 's':
49          case 'S': y0--; break;
50
51          case 'w':
52          case 'W': y0++; break;
53      }
54      display();
55  }
56
57  // This function is executed when a special key (example: F1, arrow
58  // key, Ctrl, ...) is pressed.
59  void specialkeyboard(int key, int x, int y)
60  {
61      switch (key)
62      {
63          case GLUT_KEY_UP   : y1++; break;
64          case GLUT_KEY_DOWN : y1--; break;
65          case GLUT_KEY_LEFT : x1--; break;
66          case GLUT_KEY_RIGHT: x1++; break;
67
68          case GLUT_KEY_F1:    x0 = X0; y0 = Y0; x1 = X1; y1 = Y1; break;
69      }
70      display();
71  }
72
73  int main(int argc, char ** argv)
74  {
75      srand((unsigned int) time(NULL));
76
77      glutInit(&argc, argv);
78      glutInitWindowPosition(100, 100);
79      glutInitWindowSize(200, 200);
80      glutInitDisplayMode(GLUT_DEPTH | GLUT_SINGLE | GLUT_RGBA);
81      glutCreateWindow("test");
```

```
82
83      glutDisplayFunc(display);
84
85      // Set up keyboard callbacks
86      glutKeyboardFunc(keyboard);
87      glutSpecialFunc(specialkeyboard);
88
89      init();
90      glutMainLoop();
91
92      return 0;
93  }
94
95  // As as exercise, setup other keys to change the color of the
96  // end points.
```

21 Mouse Button

```
1  // Run this program and
2  // 1. Move the mouse pointer to different places and then click and release
3  //    the mouse button. Do this a couple of times.
4  // 2. Next, hold the mouse button down, move the mouse pointer to a
5  //    different part of the time, and then release the mouse button. DO this
6  //    a couple of time.
7
8  #include <ctime>
9  #include <cstdlib>
10 #include <GL/glut.h>
11
12 void init()
13 {
14     glClearColor(1.0f, 1.0f, 1.0f, 0.0f);
15     glMatrixMode(GL_PROJECTION);
16     gluOrtho2D(0.0f, 200.0f, 0.0f, 200.0f);
17     glLineWidth(5.0f);
18 }
19
20 int x = 50;
21 int y = 50;
22
23 bool mouse_down = false;
24
25 void display()
26 {
27     glClear(GL_COLOR_BUFFER_BIT);
28
29     glBegin(GL_LINES);
30     if (mouse_down)
31     {
32         glColor3f(1.0f, 0.0f, 0.0f);
33     }
34     else
35     {
36         glColor3f(0.0f, 0.0f, 1.0f);
37     }
38     glVertex2i(0, 0);
39     glVertex2i(x, y);
```

```
40      glEnd();
41
42      glFlush();
43  }
44
45
46  // This function will be executed when a mouse button is pressed
47  // or released.
48  void mouse(int button, int state, int x, int y)
49  {
50      // Set global x to the local x
51      ::x = x;
52      // Note that the local variable y (the  y position of the mouse
53      // pointer) is measured from the top to bottom whereas the global
54      // y (used by opengl for drawing) is measured from bottom to top.
55      ::y = 200 - y;
56
57      mouse_down = (state == GLUT_DOWN);
58
59      display();
60  }
61
62  int main(int argc, char ** argv)
63  {
64      srand((unsigned int) time(NULL));
65
66      glutInit(&argc, argv);
67      glutInitWindowPosition(100, 100);
68      glutInitWindowSize(200, 200);
69      glutInitDisplayMode(GLUT_DEPTH | GLUT_SINGLE | GLUT_RGBA);
70      glutCreateWindow("test");
71
72      glutDisplayFunc(display);
73      glutMouseFunc(mouse);
74
75      init();
76      glutMainLoop();
77
78      return 0;
79  }
```

22 Mouse Motion

```
1  // Run this program and move your mouse around the window, into and out of
2  // the opengl window, without or without a mouse button pressed.
3
4  #include <ctime>
5  #include <cstdlib>
6  #include <GL/glut.h>
7
8  void init()
9  {
10     glClearColor(1.0f, 1.0f, 1.0f, 0.0f);
11     glMatrixMode(GL_PROJECTION);
12     gluOrtho2D(0.0f, 200.0f, 0.0f, 200.0f);
13     glLineWidth(5.0f);
14 }
15
16 int x = 100, y = 100;
17 float  r = 0.0f, g = 0.0f, b = 0.0f;
18
19 void display()
20 {
21     glClear(GL_COLOR_BUFFER_BIT);
22
23     glColor3f(r, g, b);
24     glBegin(GL_LINES);
25     glVertex2i(0, 0);
26     glVertex2i(x, y);
27     glEnd();
28
29     glFlush();
30 }
31
32
33 // This function will be executed when the mouse is moved with the
34 // mouse button pressed
35 void motion(int x, int y)
36 {
37     ::x = x;
38     ::y = 200 - y;
39     r = 1.0f;
```

```
40      g = 0.0f;
41      b = 0.0f;
42      display();
43  }
44
45  // This function will be executed when the mouse is moved without the
46  // mouse button being pressed
47  void passive_motion(int x, int y)
48  {
49      ::x = x;
50      ::y = 200 - y;
51      r = 0.0f;
52      g = 0.0f;
53      b = 1.0f;
54      display();
55  }
56
57
58  // This function will be execute when the mouse pointer enters or leaves
59  // the opengl window
60  void entry(int state)
61  {
62      ::x = x;
63      ::y = 200 - y;
64      r = 0.0f;
65      g = 1.0f;
66      b = 0.0f;
67      display();
68  }
69
70
71  int main(int argc, char ** argv)
72  {
73      srand((unsigned int) time(NULL));
74
75      glutInit(&argc, argv);
76      glutInitWindowPosition(100, 100);
77      glutInitWindowSize(200, 200);
78      glutInitDisplayMode(GLUT_DEPTH | GLUT_SINGLE | GLUT_RGBA);
79      glutCreateWindow("test");
80
81      glutDisplayFunc(display);
```

```
82      glutMotionFunc(motion);
83      glutPassiveMotionFunc(passive_motion);
84      glutEntryFunc(entry);
85
86      init();
87      glutMainLoop();
88
89      return 0;
90  }
```

23 Idle

```
#include <ctime>
#include <cstdlib>
#include <GL/glut.h>

void init()
{
    glClearColor(1.0f, 1.0f, 1.0f, 0.0f);
    glMatrixMode(GL_PROJECTION);
    gluOrtho2D(0.0f, 200.0f, 0.0f, 200.0f);
    glLineWidth(2.0f);
    glEnable(GL_LINE_SMOOTH);
}

float x0=50, y0=70, dx0=0.3, dy0=-0.1;
float x1=10,y1=20, dx1=-0.1,dy1=0.2;

void display()
{
    glClear(GL_COLOR_BUFFER_BIT);

    glBegin(GL_LINES);
    glColor3f(1.0f, 0.0f, 0.0f);
    glVertex2f(x0, y0);
    glColor3f(0.0f, 0.0f, 1.0f);
    glVertex2f(x1, y1);
    glEnd();

    glutSwapBuffers();
}

void mod(float & a, float & da)
{
    a += da;
    if (a < 0)
    {
        a = 0; da = -da;
    }
    else if (a > 200)
    {
```

```
40          a = 200; da = -da;
41      }
42  }
43
44  // This function will be executed when the event queue is empty.
45  void idle()
46  {
47      mod(x0, dx0);
48      mod(y0, dy0);
49      mod(x1, dx1);
50      mod(y1, dy1);
51      display();
52  }
53
54  int main(int argc, char ** argv)
55  {
56      srand((unsigned int) time(NULL));
57
58      glutInit(&argc, argv);
59      glutInitWindowPosition(100, 100);
60      glutInitWindowSize(200, 200);
61      glutInitDisplayMode(GLUT_DEPTH | GLUT_DOUBLE | GLUT_RGBA);
62      glutCreateWindow("test");
63
64      glutDisplayFunc(display);
65      glutIdleFunc(idle);
66
67      init();
68      glutMainLoop();
69
70      return 0;
71  }
72
```

24 3D Wired Cube

```
1  #include <GL/glut.h>
2
3  float eyex;
4  float eyey;
5  float eyez;
6
7  void init()
8  {
9      // Second run: set eyex to 0.5f
10     // Third run: set eyex to 1.0f
11     eyex = 0.0f;
12     eyey = 0.0f;
13     eyez = 1.0f;
14
15     // Set view volume for orthographic (parallel) projection
16     // (Note: lengths of line segment stays the same, i.e.,
17     // lines further away do not appear shorted, i.e., no
18     // sense of perspective. This is the case for architectural
19     // drawing, engineering CAD drawing, 2D games, isometric
20     // games, etc.
21     //
22     // The 6 values passed to glOrtho() is the clipping volumn,
23     // i.e., only objects in this volume are seen.
24     glMatrixMode(GL_PROJECTION);
25     glLoadIdentity();
26     glOrtho(-4.0, 4.0,  // -4 <= x <= 4
27             -4.0, 4.0,  // -4 <= y <= 4
28             -4.0, 4.0); // -4 <= z <= 4
29
30     // set camera
31     glMatrixMode(GL_MODELVIEW);
32     glLoadIdentity();
33     gluLookAt(eyex, eyey, eyez, // eye position x,y,z
34               0.0, 0.0, 3.0, // eye direction x,y,z
35               0.0, 1.0, 0.0);// eye up direction x,y,z
36
37     glViewport(0, 0, 400, 400);
38
39     glClearColor(1.0f, 1.0f, 1.0f, 0.0f);
```

```
40      glColor3f(0.0f, 0.0f, 1.0f);
41  }
42
43  void display()
44  {
45      glMatrixMode(GL_MODELVIEW);
46      glLoadIdentity();
47      gluLookAt(eyex, eyey, eyez, // eye position x,y,z
48                0.0, 0.0, 0.0, // eye direction x,y,z
49                0.0, 1.0, 0.0);// eye up direction x,y,z
50
51      glClear(GL_COLOR_BUFFER_BIT);
52
53      glColor3f(0.0f, 0.0f, 1.0f);
54      glutWireCube(1.0f);
55
56      glutSwapBuffers();
57  }
58
59  void keyboard(unsigned char key, int x, int y)
60  {
61      switch(key)
62      {
63          case 'a': eyex -= 0.1; break;
64          case 'd': eyex += 0.1; break;
65          case 'w': eyey += 0.1; break;
66          case 's': eyey -= 0.1; break;
67
68          // Notice that this changing eyez does not change the
69          // the image (because we're using parallel projection)
70          case 'o': eyez -= 0.1; break;
71          case 'l': eyez += 0.1; break;
72      }
73      display();
74  }
75
76  int main(int argc, char ** argv)
77  {
78    glutInit(&argc, argv);
79    glutInitWindowPosition(100, 100);
80    glutInitWindowSize(400, 400);
81    glutInitDisplayMode(GLUT_DEPTH | GLUT_DOUBLE | GLUT_RGBA);
```

```
82    glutCreateWindow("test");
83
84    glutDisplayFunc(display);
85    glutKeyboardFunc(keyboard);
86    init();
87    glutMainLoop();
88
89    return 0;
90  }
```

25 3D Wired Torus

```
#include <GL/glut.h>

float eyex;
float eyey;
float eyez;

void init()
{
    eyex = 1.0f;
    eyey = 1.0f;
    eyez = 0.5f;

    // set view volume
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glOrtho(-4.0, 4.0,  // -4 <= x <= 4
            -4.0, 4.0,  // -4 <= y <= 4
            -4.0, 4.0); // -4 <= z <= 4

    // set camera
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    gluLookAt(eyex, eyey, eyez, // eye position x,y,z
              0.0, 0.0, 0.0, // eye direction x,y,z
              0.0, 1.0, 0.0);// eye up direction x,y,z

    glViewport(0, 0, 400, 400);

    glClearColor(1.0f, 1.0f, 1.0f, 0.0f);
    glColor3f(0.0f, 0.0f, 1.0f);
}

void display()
{
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    gluLookAt(eyex, eyey, eyez, // eye position x,y,z
              0.0, 0.0, 0.0, // eye direction x,y,z
              0.0, 1.0, 0.0);// eye up direction x,y,z
```

```
40
41      glClear(GL_COLOR_BUFFER_BIT);
42
43      glColor3f(0.0f, 0.0f, 1.0f);
44      glutWireTorus(0.5, 1.7, 10, 20);
45
46      glutSwapBuffers();
47  }
48
49  void keyboard(unsigned char key, int x, int y)
50  {
51      switch(key)
52      {
53          case 'a': eyex -= 0.1; break;
54          case 'd': eyex += 0.1; break;
55          case 'w': eyey += 0.1; break;
56          case 's': eyey -= 0.1; break;
57      }
58      display();
59  }
60
61  int main(int argc, char ** argv)
62  {
63    glutInit(&argc, argv);
64    glutInitWindowPosition(100, 100);
65    glutInitWindowSize(400, 400);
66    glutInitDisplayMode(GLUT_DEPTH | GLUT_DOUBLE | GLUT_RGBA);
67    glutCreateWindow("test");
68
69    glutDisplayFunc(display);
70    glutKeyboardFunc(keyboard);
71    init();
72    glutMainLoop();
73
74    return 0;
75  }
```

26 3D Wired Teapot

```
#include <GL/glut.h>

float eyex;
float eyey;
float eyez;

void init()
{
    eyex = 0.0f;
    eyey = 0.0f;
    eyez = 1.0f;

    // set view volume
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glOrtho(-4.0, 4.0,  // -4 <= x <= 4
            -4.0, 4.0,  // -4 <= y <= 4
            -4.0, 4.0); // -4 <= z <= 4

    // set camera
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    gluLookAt(eyex, eyey, eyez, // eye position x,y,z
              0.0, 0.0, 0.0, // eye direction x,y,z
              0.0, 1.0, 0.0);// eye up direction x,y,z

    glViewport(0, 0, 400, 400);

    glClearColor(0.0f, 0.0f, 0.0f, 0.0f);
    glColor3f(0.0f, 1.0f, 0.0f);
}

void display()
{
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    gluLookAt(eyex, eyey, eyez, // eye position x,y,z
              0.0, 0.0, 0.0, // eye direction x,y,z
              0.0, 1.0, 0.0);// eye up direction x,y,z
```

```
40
41      glClear(GL_COLOR_BUFFER_BIT);
42      glutWireTeapot(1.0);
43
44      glutSwapBuffers();
45  }
46
47  void keyboard(unsigned char key, int x, int y)
48  {
49      switch(key)
50      {
51          case 'a': eyex -= 0.1; break;
52          case 'd': eyex += 0.1; break;
53          case 'w': eyey += 0.1; break;
54          case 's': eyey -= 0.1; break;
55      }
56      display();
57  }
58
59  int main(int argc, char ** argv)
60  {
61    glutInit(&argc, argv);
62    glutInitWindowPosition(100, 100);
63    glutInitWindowSize(400, 400);
64    glutInitDisplayMode(GLUT_DEPTH | GLUT_DOUBLE | GLUT_RGBA);
65    glutCreateWindow("test");
66
67    glutDisplayFunc(display);
68    glutKeyboardFunc(keyboard);
69    init();
70    glutMainLoop();
71
72    return 0;
73  }
```

27 3D Wired Sphere

```
#include <GL/glut.h>

float eyex;
float eyey;
float eyez;

void init()
{
    eyex = 1.0f;
    eyey = 1.0f;
    eyez = 0.5f;

    // set view volume
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glOrtho(-4.0, 4.0,  // -4 <= x <= 4
            -4.0, 4.0,  // -4 <= y <= 4
            -4.0, 4.0); // -4 <= z <= 4

    // set camera
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    gluLookAt(eyex, eyey, eyez, // eye position x,y,z
              0.0, 0.0, 0.0, // eye direction x,y,z
              0.0, 1.0, 0.0);// eye up direction x,y,z

    glViewport(0, 0, 400, 400);

    glClearColor(1.0f, 1.0f, 1.0f, 0.0f);
    glColor3f(0.0f, 0.0f, 1.0f);
}

void display()
{
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    gluLookAt(eyex, eyey, eyez, // eye position x,y,z
              0.0, 0.0, 0.0, // eye direction x,y,z
              0.0, 1.0, 0.0);// eye up direction x,y,z
```

```
40
41      glClear(GL_COLOR_BUFFER_BIT);
42
43      glColor3f(0.0f, 0.0f, 1.0f);
44      glutWireSphere(1.0, 10, 10);
45
46      glutSwapBuffers();
47  }
48
49  void keyboard(unsigned char key, int x, int y)
50  {
51      switch(key)
52      {
53          case 'a': eyex -= 0.1; break;
54          case 'd': eyex += 0.1; break;
55          case 'w': eyey += 0.1; break;
56          case 's': eyey -= 0.1; break;
57      }
58      display();
59  }
60
61  int main(int argc, char ** argv)
62  {
63    glutInit(&argc, argv);
64    glutInitWindowPosition(100, 100);
65    glutInitWindowSize(400, 400);
66    glutInitDisplayMode(GLUT_DEPTH | GLUT_DOUBLE | GLUT_RGBA);
67    glutCreateWindow("test");
68
69    glutDisplayFunc(display);
70    glutKeyboardFunc(keyboard);
71    init();
72    glutMainLoop();
73
74    return 0;
75  }
```

28 3D Wired Dodecahedron

```
1  #include <GL/glut.h>
2
3  float eyex;
4  float eyey;
5  float eyez;
6
7  void init()
8  {
9      // Second run: set eyex to 0.5f
10     // Third run: set eyex to 1.0f
11     eyex = 0.0f;
12     eyey = 0.0f;
13     eyez = 1.0f;
14
15     // Set view volume for orthographic (parallel) projection
16     // (Note: lengths of line segment stays the same, i.e.,
17     // lines further away do not appear shorted, i.e., no
18     // sense of perspective. This is the case for architectural
19     // drawing, engineering CAD drawing, 2D games, isometric
20     // games, etc.
21     //
22     // The 6 values passed to glOrtho() is the clipping volumn,
23     // i.e., only objects in this volume are seen.
24     glMatrixMode(GL_PROJECTION);
25     glLoadIdentity();
26     glOrtho(-4.0, 4.0,  // -4 <= x <= 4
27             -4.0, 4.0,  // -4 <= y <= 4
28             -4.0, 4.0); // -4 <= z <= 4
29
30     // set camera
31     glMatrixMode(GL_MODELVIEW);
32     glLoadIdentity();
33     gluLookAt(eyex, eyey, eyez, // eye position x,y,z
34               0.0, 0.0, 3.0, // eye direction x,y,z
35               0.0, 1.0, 0.0);// eye up direction x,y,z
36
37     glViewport(0, 0, 400, 400);
38
39     glClearColor(1.0f, 1.0f, 1.0f, 0.0f);
```

```
40      glColor3f(0.0f, 0.0f, 1.0f);
41  }
42
43  void display()
44  {
45      glMatrixMode(GL_MODELVIEW);
46      glLoadIdentity();
47      gluLookAt(eyex, eyey, eyez, // eye position x,y,z
48                0.0, 0.0, 0.0, // eye direction x,y,z
49                0.0, 1.0, 0.0);// eye up direction x,y,z
50
51      glClear(GL_COLOR_BUFFER_BIT);
52
53      glColor3f(0.0f, 0.0f, 1.0f);
54      glutWireDodecahedron();
55
56      glutSwapBuffers();
57  }
58
59  void keyboard(unsigned char key, int x, int y)
60  {
61      switch(key)
62      {
63          case 'a': eyex -= 0.1; break;
64          case 'd': eyex += 0.1; break;
65          case 'w': eyey += 0.1; break;
66          case 's': eyey -= 0.1; break;
67
68          // Notice that this changing eyez does not change the
69          // the image (because we're using parallel projection)
70          case 'o': eyez -= 0.1; break;
71          case 'l': eyez += 0.1; break;
72      }
73      display();
74  }
75
76  int main(int argc, char ** argv)
77  {
78    glutInit(&argc, argv);
79    glutInitWindowPosition(100, 100);
80    glutInitWindowSize(400, 400);
81    glutInitDisplayMode(GLUT_DEPTH | GLUT_DOUBLE | GLUT_RGBA);
```

```
82    glutCreateWindow("test");
83
84    glutDisplayFunc(display);
85    glutKeyboardFunc(keyboard);
86    init();
87    glutMainLoop();
88
89    return 0;
90  }
```

29 3D Wired Octahedron

```
#include <GL/glut.h>

float eyex;
float eyey;
float eyez;

void init()
{
    // Second run: set eyex to 0.5f
    // Third run: set eyex to 1.0f
    eyex = 0.0f;
    eyey = 0.0f;
    eyez = 1.0f;

    // Set view volume for orthographic (parallel) projection
    // (Note: lengths of line segment stays the same, i.e.,
    // lines further away do not appear shorted, i.e., no
    // sense of perspective. This is the case for architectural
    // drawing, engineering CAD drawing, 2D games, isometric
    // games, etc.
    //
    // The 6 values passed to glOrtho() is the clipping volumn,
    // i.e., only objects in this volume are seen.
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glOrtho(-4.0, 4.0,  // -4 <= x <= 4
            -4.0, 4.0,  // -4 <= y <= 4
            -4.0, 4.0); // -4 <= z <= 4

    // set camera
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    gluLookAt(eyex, eyey, eyez, // eye position x,y,z
              0.0, 0.0, 3.0, // eye direction x,y,z
              0.0, 1.0, 0.0);// eye up direction x,y,z

    glViewport(0, 0, 400, 400);

    glClearColor(1.0f, 1.0f, 1.0f, 0.0f);
```

```
40      glColor3f(0.0f, 0.0f, 1.0f);
41  }
42
43  void display()
44  {
45      glMatrixMode(GL_MODELVIEW);
46      glLoadIdentity();
47      gluLookAt(eyex, eyey, eyez, // eye position x,y,z
48                0.0, 0.0, 0.0, // eye direction x,y,z
49                0.0, 1.0, 0.0);// eye up direction x,y,z
50
51      glClear(GL_COLOR_BUFFER_BIT);
52
53      glColor3f(0.0f, 0.0f, 1.0f);
54      glutWireDodecahedron();
55
56      glutSwapBuffers();
57  }
58
59  void keyboard(unsigned char key, int x, int y)
60  {
61      switch(key)
62      {
63          case 'a': eyex -= 0.1; break;
64          case 'd': eyex += 0.1; break;
65          case 'w': eyey += 0.1; break;
66          case 's': eyey -= 0.1; break;
67
68          // Notice that this changing eyez does not change the
69          // the image (because we're using parallel projection)
70          case 'o': eyez -= 0.1; break;
71          case 'l': eyez += 0.1; break;
72      }
73      display();
74  }
75
76  int main(int argc, char ** argv)
77  {
78    glutInit(&argc, argv);
79    glutInitWindowPosition(100, 100);
80    glutInitWindowSize(400, 400);
81    glutInitDisplayMode(GLUT_DEPTH | GLUT_DOUBLE | GLUT_RGBA);
```

```
82    glutCreateWindow("test");
83
84    glutDisplayFunc(display);
85    glutKeyboardFunc(keyboard);
86    init();
87    glutMainLoop();
88
89    return 0;
90  }
```

30 3D Wired Icosahedron

```
#include <GL/glut.h>

float eyex;
float eyey;
float eyez;

void init()
{
    eyex = 0.0f;
    eyey = 0.0f;
    eyez = 1.0f;

    // set view volume
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glOrtho(-4.0, 4.0,  // -4 <= x <= 4
            -4.0, 4.0,  // -4 <= y <= 4
            -4.0, 4.0); // -4 <= z <= 4

    // set camera
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    gluLookAt(eyex, eyey, eyez, // eye position x,y,z
              0.0, 0.0, 0.0, // eye direction x,y,z
              0.0, 1.0, 0.0);// eye up direction x,y,z

    glViewport(0, 0, 400, 400);

    glClearColor(0.0f, 0.0f, 0.0f, 0.0f);
    glColor3f(0.0f, 1.0f, 0.0f);
}

void display()
{
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    gluLookAt(eyex, eyey, eyez, // eye position x,y,z
              0.0, 0.0, 0.0, // eye direction x,y,z
              0.0, 1.0, 0.0);// eye up direction x,y,z
```

```
40
41      glClear(GL_COLOR_BUFFER_BIT);
42      glutWireIcosahedron();
43
44      glutSwapBuffers();
45  }
46
47  void keyboard(unsigned char key, int x, int y)
48  {
49      switch(key)
50      {
51          case 'a': eyex -= 0.1; break;
52          case 'd': eyex += 0.1; break;
53          case 'w': eyey += 0.1; break;
54          case 's': eyey -= 0.1; break;
55      }
56      display();
57  }
58
59  int main(int argc, char ** argv)
60  {
61    glutInit(&argc, argv);
62    glutInitWindowPosition(100, 100);
63    glutInitWindowSize(400, 400);
64    glutInitDisplayMode(GLUT_DEPTH | GLUT_DOUBLE | GLUT_RGBA);
65    glutCreateWindow("test");
66
67    glutDisplayFunc(display);
68    glutKeyboardFunc(keyboard);
69    init();
70    glutMainLoop();
71
72    return 0;
73  }
```

31 3D Wired Cylinder

```
#include <GL/glut.h>

float eyex;
float eyey;
float eyez;

void init()
{
    eyex = 0.0f;
    eyey = 0.0f;
    eyez = 1.0f;

    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glOrtho(-4.0, 4.0,  // -4 <= x <= 4
            -4.0, 4.0,  // -4 <= y <= 4
            -4.0, 4.0); // -4 <= z <= 4

    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    gluLookAt(eyex, eyey, eyez, // eye position x,y,z
              0.0, 0.0, 3.0, // eye direction x,y,z
              0.0, 1.0, 0.0);// eye up direction x,y,z

    glViewport(0, 0, 400, 400);

    glClearColor(1.0f, 1.0f, 1.0f, 0.0f);
    glColor3f(0.0f, 0.0f, 1.0f);
}

void display()
{
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    gluLookAt(eyex, eyey, eyez, // eye position x,y,z
              0.0, 0.0, 0.0, // eye direction x,y,z
              0.0, 1.0, 0.0);// eye up direction x,y,z

    glClear(GL_COLOR_BUFFER_BIT);
```

```
40
41      glColor3f(0.0f, 0.0f, 1.0f);
42
43      GLUquadricObj * p = gluNewQuadric();
44      //gluQuadricDrawStyle(p, GLU_LINE);
45      gluQuadricDrawStyle(p, GLU_FILL);
46
47      float base_radius = 1.0f;
48      float top_radius = 1.0f;
49      float height = 2.0f;
50      int slice_per_ring = 20;
51      int rings = 20;
52
53      gluCylinder(p, base_radius, top_radius, height, slice_per_ring, rings);
54      gluDeleteQuadric(p);
55
56      glutSwapBuffers();
57  }
58
59  void keyboard(unsigned char key, int x, int y)
60  {
61      switch(key)
62      {
63          case 'a': eyex -= 0.1; break;
64          case 'd': eyex += 0.1; break;
65          case 'w': eyey += 0.1; break;
66          case 's': eyey -= 0.1; break;
67
68          // Notice that this changing eyez does not change the
69          // the image (because we're using parallel projection)
70          case 'o': eyez -= 0.1; break;
71          case 'l': eyez += 0.1; break;
72      }
73      display();
74  }
75
76  int main(int argc, char ** argv)
77  {
78    glutInit(&argc, argv);
79    glutInitWindowPosition(100, 100);
80    glutInitWindowSize(400, 400);
81    glutInitDisplayMode(GLUT_DEPTH | GLUT_DOUBLE | GLUT_RGBA);
```

```
82    glutCreateWindow("test");
83
84    glutDisplayFunc(display);
85    glutKeyboardFunc(keyboard);
86    init();
87    glutMainLoop();
88
89    return 0;
90  }
```

32 3D Wired Disk

***** NEW *****

```
#include <GL/glut.h>

float eyex;
float eyey;
float eyez;

void init()
{
    eyex = 0.0f;
    eyey = 0.0f;
    eyez = 1.0f;

    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glOrtho(-4.0, 4.0,  // -4 <= x <= 4
            -4.0, 4.0,  // -4 <= y <= 4
            -4.0, 4.0); // -4 <= z <= 4

    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    gluLookAt(eyex, eyey, eyez, // eye position x,y,z
              0.0, 0.0, 3.0, // eye direction x,y,z
              0.0, 1.0, 0.0);// eye up direction x,y,z

    glViewport(0, 0, 400, 400);

    glClearColor(1.0f, 1.0f, 1.0f, 0.0f);
    glColor3f(0.0f, 0.0f, 1.0f);
}

void display()
{
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    gluLookAt(eyex, eyey, eyez, // eye position x,y,z
              0.0, 0.0, 0.0, // eye direction x,y,z
              0.0, 1.0, 0.0);// eye up direction x,y,z

```

```
39      glClear(GL_COLOR_BUFFER_BIT);
40
41      glColor3f(0.0f, 0.0f, 1.0f);
42
43      GLUquadricObj * p = gluNewQuadric();
44      gluQuadricDrawStyle(p, GLU_LINE);
45
46      float inner_radius = 0.0f;
47      float outer_radius = 2.0f;
48      int slice_per_ring = 10;
49      int rings = 40;
50
51      gluDisk(p, inner_radius, outer_radius, slice_per_ring, rings);
52      gluDeleteQuadric(p);
53
54      glutSwapBuffers();
55  }
56
57  void keyboard(unsigned char key, int x, int y)
58  {
59      switch(key)
60      {
61          case 'a': eyex -= 0.1; break;
62          case 'd': eyex += 0.1; break;
63          case 'w': eyey += 0.1; break;
64          case 's': eyey -= 0.1; break;
65
66          // Notice that this changing eyez does not change the
67          // the image (because we're using parallel projection)
68          case 'o': eyez -= 0.1; break;
69          case 'l': eyez += 0.1; break;
70      }
71      display();
72  }
73
74  int main(int argc, char ** argv)
75  {
76    glutInit(&argc, argv);
77    glutInitWindowPosition(100, 100);
78    glutInitWindowSize(400, 400);
79    glutInitDisplayMode(GLUT_DEPTH | GLUT_DOUBLE | GLUT_RGBA);
80    glutCreateWindow("test");
```

```
81
82    glutDisplayFunc(display);
83    glutKeyboardFunc(keyboard);
84    init();
85    glutMainLoop();
86
87    return 0;
88  }
```

33 Translation

```
#include <GL/glut.h>

float eyex;
float eyey;
float eyez;

GLUquadricObj * p = gluNewQuadric();

void init()
{
    eyex = 0.0f;
    eyey = 0.0f;
    eyez = 1.0f;

    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glOrtho(-4.0, 4.0,  // -4 <= x <= 4
            -4.0, 4.0,  // -4 <= y <= 4
            -4.0, 4.0); // -4 <= z <= 4

    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    gluLookAt(eyex, eyey, eyez, // eye position x,y,z
              0.0, 0.0, 3.0, // eye direction x,y,z
              0.0, 1.0, 0.0);// eye up direction x,y,z

    glViewport(0, 0, 400, 400);

    glClearColor(1.0f, 1.0f, 1.0f, 0.0f);
    glColor3f(0.0f, 0.0f, 1.0f);

    gluQuadricDrawStyle(p, GLU_LINE);
}

void display()
{
    glClear(GL_COLOR_BUFFER_BIT);

    glMatrixMode(GL_MODELVIEW);
```

```
40      glLoadIdentity();
41      gluLookAt(eyex, eyey, eyez, // eye position x,y,z
42                0.0, 0.0, 0.0, // eye direction x,y,z
43                0.0, 1.0, 0.0);// eye up direction x,y,z
44
45      float base_radius = 0.5f;
46      float top_radius = 0.25f;
47      float height = 2.0f;
48      int slice_per_ring = 20;
49      int rings = 20;
50      glColor3f(1.0f, 0.0f, 0.0f);
51      gluCylinder(p, base_radius, top_radius, height, slice_per_ring, rings);
52
53      // second cylinder
54      // second run: uncomment the next line
55      //glLoadIdentity();
56      glTranslatef(1.0f, 0.0f, 0.0f);
57      glColor3f(0.0f, 1.0f, 0.0f);
58      gluCylinder(p, 0.4f, 0.4f, 1.0f, 10, 10);
59
60      // third cylinder
61      // second run: uncomment the next line
62      //glLoadIdentity();
63      glTranslatef(1.0f, 0.0f, 0.0f);     // Note that the translation is
64                                          // accumulative
65      glColor3f(0.0f, 0.0f, 1.0f);
66      gluCylinder(p, 0.1f, 0.2f, 0.5f, 10, 10);
67
68      glutSwapBuffers();
69  }
70
71  void keyboard(unsigned char key, int x, int y)
72  {
73      switch(key)
74      {
75          case 'a': eyex -= 0.1; break;
76          case 'd': eyex += 0.1; break;
77          case 'w': eyey += 0.1; break;
78          case 's': eyey -= 0.1; break;
79
80          // Notice that this changing eyez does not change the
81          // the image (because we're using parallel projection)
```

```
82          case 'o': eyez -= 0.1; break;
83          case 'l': eyez += 0.1; break;
84      }
85      display();
86  }
87
88  int main(int argc, char ** argv)
89  {
90    glutInit(&argc, argv);
91    glutInitWindowPosition(100, 100);
92    glutInitWindowSize(400, 400);
93    glutInitDisplayMode(GLUT_DEPTH | GLUT_DOUBLE | GLUT_RGBA);
94    glutCreateWindow("test");
95
96    glutDisplayFunc(display);
97    glutKeyboardFunc(keyboard);
98    init();
99    glutMainLoop();
100
101   gluDeleteQuadric(p);
102
103   return 0;
104 }
```

34 Translation and Rotation

```
#include <GL/glut.h>

float eyex;
float eyey;
float eyez;

GLUquadricObj * p = gluNewQuadric();

void init()
{
    eyex = 0.0f;
    eyey = 0.0f;
    eyez = 1.0f;

    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glOrtho(-4.0, 4.0,  // -4 <= x <= 4
            -4.0, 4.0,  // -4 <= y <= 4
            -4.0, 4.0); // -4 <= z <= 4

    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    gluLookAt(eyex, eyey, eyez, // eye position x,y,z
              0.0, 0.0, 3.0, // eye direction x,y,z
              0.0, 1.0, 0.0);// eye up direction x,y,z

    glViewport(0, 0, 400, 400);

    glClearColor(1.0f, 1.0f, 1.0f, 0.0f);
    glColor3f(0.0f, 0.0f, 1.0f);

    gluQuadricDrawStyle(p, GLU_LINE);
}

void display()
{
    glClear(GL_COLOR_BUFFER_BIT);

    glMatrixMode(GL_MODELVIEW);
```

```
40      glLoadIdentity();
41      gluLookAt(eyex, eyey, eyez, // eye position x,y,z
42                0.0, 0.0, 0.0, // eye direction x,y,z
43                0.0, 1.0, 0.0);// eye up direction x,y,z
44
45      float base_radius = 0.5f;
46      float top_radius = 0.25f;
47      float height = 2.0f;
48      int slice_per_ring = 20;
49      int rings = 20;
50      glColor3f(1.0f, 0.0f, 0.0f);
51      gluCylinder(p, base_radius, top_radius, height, slice_per_ring, rings);
52
53      // second cylinder
54      glTranslatef(1.0f, 0.0f, 0.0f);
55      float t = 30.0f;
56      glRotatef(t, 1, 0, 0); // t degrees about x axis
57      glColor3f(0.0f, 1.0f, 0.0f);
58      gluCylinder(p, 0.4f, 0.4f, 1.0f, 10, 10);
59
60      // third cylinder
61      glTranslatef(1.0f, 0.0f, 0.0f);     // Note that the translation is
62                                          // accumulative
63      glRotatef(t, 1, 0, 0);              // Note that the rotation is
64                                          // accumulative
65      glColor3f(0.0f, 0.0f, 1.0f);
66      gluCylinder(p, 0.1f, 0.2f, 0.5f, 10, 10);
67
68      glutSwapBuffers();
69  }
70
71  void keyboard(unsigned char key, int x, int y)
72  {
73      switch(key)
74      {
75          case 'a': eyex -= 0.1; break;
76          case 'd': eyex += 0.1; break;
77          case 'w': eyey += 0.1; break;
78          case 's': eyey -= 0.1; break;
79
80          // Notice that this changing eyez does not change the
81          // the image (because we're using parallel projection)
```

```
82          case 'o': eyez -= 0.1; break;
83          case 'l': eyez += 0.1; break;
84      }
85      display();
86  }
87
88  int main(int argc, char ** argv)
89  {
90    glutInit(&argc, argv);
91    glutInitWindowPosition(100, 100);
92    glutInitWindowSize(400, 400);
93    glutInitDisplayMode(GLUT_DEPTH | GLUT_DOUBLE | GLUT_RGBA);
94    glutCreateWindow("test");
95
96    glutDisplayFunc(display);
97    glutKeyboardFunc(keyboard);
98    init();
99    glutMainLoop();
100
101   gluDeleteQuadric(p);
102
103   return 0;
104 }
```

35 Translation, Rotation, and Scaling

```
#include <GL/glut.h>

float eyex;
float eyey;
float eyez;

GLUquadricObj * p = gluNewQuadric();

void init()
{
    eyex = 0.0f;
    eyey = 0.0f;
    eyez = 1.0f;

    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glOrtho(-4.0, 4.0,  // -4 <= x <= 4
            -4.0, 4.0,  // -4 <= y <= 4
            -4.0, 4.0); // -4 <= z <= 4

    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    gluLookAt(eyex, eyey, eyez, // eye position x,y,z
              0.0, 0.0, 3.0, // eye direction x,y,z
              0.0, 1.0, 0.0);// eye up direction x,y,z

    glViewport(0, 0, 400, 400);

    glClearColor(1.0f, 1.0f, 1.0f, 0.0f);
    glColor3f(0.0f, 0.0f, 1.0f);

    gluQuadricDrawStyle(p, GLU_LINE);
}

void display()
{
    glClear(GL_COLOR_BUFFER_BIT);

    glMatrixMode(GL_MODELVIEW);
```

```
40      glLoadIdentity();
41      gluLookAt(eyex, eyey, eyez, // eye position x,y,z
42                0.0, 0.0, 0.0, // eye direction x,y,z
43                0.0, 1.0, 0.0);// eye up direction x,y,z
44
45      float base_radius = 0.5f;
46      float top_radius = 0.25f;
47      float height = 2.0f;
48      int slice_per_ring = 20;
49      int rings = 20;
50      glColor3f(1.0f, 0.0f, 0.0f);
51      gluCylinder(p, base_radius, top_radius, height, slice_per_ring, rings);
52
53      // second cylinder
54      glTranslatef(1.0f, 0.0f, 0.0f);
55      float t = 30.0f;
56      glRotatef(t, 1, 0, 0); // t degrees about x axis
57      glColor3f(0.0f, 1.0f, 0.0f);
58      gluCylinder(p, 0.4f, 0.4f, 1.0f, 10, 10);
59
60      // third cylinder
61      glTranslatef(1.0f, 0.0f, 0.0f);     // Note that the translation is
62                                          // accumulative
63      glRotatef(t, 1, 0, 0);              // Note that the rotation is
64                                          // accumulative
65      glScalef(6.0f, 7.0f, 8.0f);
66      glColor3f(0.0f, 0.0f, 1.0f);
67      gluCylinder(p, 0.1f, 0.2f, 0.5f, 10, 10);
68
69      // Note that the lines starts at (0,0,0) which corresponds to
70      // the original of this model's (0,0,0), not original world
71      // coordinate system
72      glBegin(GL_LINES);
73      glVertex3f(0, 0, 0);
74      glVertex3f(1, 1, 1);
75      glEnd();
76
77      glutSwapBuffers();
78  }
79
80  void keyboard(unsigned char key, int x, int y)
81  {
```

```
82      switch(key)
83      {
84          case 'a': eyex -= 0.1; break;
85          case 'd': eyex += 0.1; break;
86          case 'w': eyey += 0.1; break;
87          case 's': eyey -= 0.1; break;
88
89          // Notice that this changing eyez does not change the
90          // the image (because we're using parallel projection)
91          case 'o': eyez -= 0.1; break;
92          case 'l': eyez += 0.1; break;
93      }
94      display();
95  }
96
97  int main(int argc, char ** argv)
98  {
99    glutInit(&argc, argv);
100   glutInitWindowPosition(100, 100);
101   glutInitWindowSize(400, 400);
102   glutInitDisplayMode(GLUT_DEPTH | GLUT_DOUBLE | GLUT_RGBA);
103   glutCreateWindow("test");
104
105   glutDisplayFunc(display);
106   glutKeyboardFunc(keyboard);
107   init();
108   glutMainLoop();
109
110   gluDeleteQuadric(p);
111
112   return 0;
113 }
```

36 Perspective Viewing

```
#include <GL/glut.h>

float eyex;
float eyey;
float eyez;

void init()
{
    eyex = 0.0f;
    eyey = 1.0f;
    eyez = 5.0f;

    // set view volume
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();

    // Sides of viewport define an infinite pyramid
    // The pyramid is truncated by zNear,zFar to get a frustum
    float fovy = 90.0;  // fovy = field of view, the angle made, along y-axis
    float aspect = 1.0; // aspect ratio of viewport: width/height
    float zNear = 3.0;  // second run: use 0.0
    float zFar = 5.0;   // second run: use 100.0
    gluPerspective(fovy, aspect, zNear, zFar);

    // set camera
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    gluLookAt(eyex, eyey, eyez,
              0.0, 0.0, 0.0,
              0.0, 1.0, 0.0);

    glViewport(0, 0, 400, 400);

    glClearColor(0.0f, 0.0f, 0.0f, 0.0f);
    glColor3f(0.0f, 1.0f, 0.0f);
}

void display()
{
```

```
40      glMatrixMode(GL_MODELVIEW);
41      glLoadIdentity();
42      gluLookAt(eyex, eyey, eyez, // eye position x,y,z
43                0.0, 0.0, 0.0, // eye direction x,y,z
44                0.0, 1.0, 0.0);// eye up direction x,y,z
45
46      glClear(GL_COLOR_BUFFER_BIT);
47
48      glBegin(GL_LINES);
49      for (float x = -100.0f; x < 100.0f; x += 1.0f)
50      {
51          glVertex3f(x, 0, -200);
52          glVertex3f(x, 0, 100);
53      }
54      glEnd();
55
56      // Third run:
57      //glTranslatef(0.0f, 0.0f, 5.0f);
58      for (int i = 0; i < 10; i++)
59      {
60          glutWireTeapot(1.0);
61          glTranslatef(0.0f, 0.0f, -i * 2.0f);
62      }
63
64      glutSwapBuffers();
65  }
66
67  void keyboard(unsigned char key, int x, int y)
68  {
69      switch(key)
70      {
71          case 'a': eyex -= 0.1; break;
72          case 'd': eyex += 0.1; break;
73          case 'w': eyey += 0.1; break;
74          case 's': eyey -= 0.1; break;
75      }
76      display();
77  }
78
79  int main(int argc, char ** argv)
80  {
81    glutInit(&argc, argv);
```

```
82    glutInitWindowPosition(100, 100);
83    glutInitWindowSize(400, 400);
84    glutInitDisplayMode(GLUT_DEPTH | GLUT_DOUBLE | GLUT_RGBA);
85    glutCreateWindow("test");
86
87    glutDisplayFunc(display);
88    glutKeyboardFunc(keyboard);
89    init();
90    glutMainLoop();
91
92    return 0;
93  }
```

37 Modelview Stack

```
#include <GL/glut.h>

float eyex;
float eyey;
float eyez;

GLUquadricObj * p = gluNewQuadric();

// The modelview stack
// -------------------
// The transformations you have been using (glTranslatef, glRotatef,
// glScale) is applied to a matrix of transformation that OpenGL
// maintains. The effect is accumulative (you already know that).
// If the current transformation is matrix M, and you call one of the
// above transformations say with matrix N, the current transformation
// becomes M*N.
//
// This matrix is call the modelview matrix.
//
// When you want to work with this matrix you call
//
//     glMatrixMode(GL_MODELVIEW);
//
// Why? Because OpenGL works with many matrices and you need to tell
// OpenGL which matrix you want to work with. The above loads the
// matrix.
//
// The following will reset the modelview matrix to the identity matrix
// so that essentially there is no transformation on the object you're
// modeling:
//
//   glMatrixMode(GL_PROJECTION);
//   glLoadIdentity();
//
// Actually OpenGL can maintain a stack of matrices. The one that you
// work on is the top of the stack. (Therefore there must be at least
// one in the stack.)
//
// Suppose the stack looks like this [M].
```

```
40 // If you call a glRotatef and the corresponding matrix for this
41 // transformation is N, then the stack becomes [(M*N)]. There is
42 // still one modelview matrix in the stack.
43 //
44 // You can get OpenGL to push this stack. See the push function below.
45 // After pushing the modelview stack, it becomes
46 //   [(M*N), (M*N)]
47 // (The bottom of stack is on the left.) In other words OpenGL makes
48 // a copy of the top and push it onto the stack.
49 //
50 // If you now apply a glScalef transformation whose matrix is P, then
51 // the stack becomes
52 //   [(M*N), (M*N*P)]
53 // Remember that the current modelview matrix that OpenGL uses for
54 // modeling objects is the top of the stack. If you apply another
55 // transformation say with matrix Q, the top modelview stack becomes
56 //   [(M*N), (M*N*P*Q)]
57 //
58 // OpenGL also lets you pop this modelview stack. When you pop the
59 // stack it becomes
60 //   [(M*N)]
61 // At this point, the modelview matrix for modeling objects is M*N.
62 //
63 // Note that OpenGL works with many matrices and stacks of matrices.
64 // The modelview stack is only one of them.
65 void push(void)
66 {
67     glMatrixMode(GL_MODELVIEW); // just in case the modelview matrix
68                                 // was not loaded.
69     glPushMatrix();
70 }
71
72
73 void pop(void)
74 {
75     glMatrixMode(GL_MODELVIEW);
76     glPopMatrix();
77 }
78
79
80
81
```

```
82  void init()
83  {
84      eyex = 0.0f;
85      eyey = 0.0f;
86      eyez = 1.0f;
87
88      glMatrixMode(GL_PROJECTION);
89      glLoadIdentity();
90      glOrtho(-4.0, 4.0,  // -4 <= x <= 4
91              -4.0, 4.0,  // -4 <= y <= 4
92              -4.0, 4.0); // -4 <= z <= 4
93
94      glMatrixMode(GL_MODELVIEW);
95      glLoadIdentity();
96      gluLookAt(eyex, eyey, eyez, // eye position x,y,z
97                0.0, 0.0, 3.0, // eye direction x,y,z
98                0.0, 1.0, 0.0);// eye up direction x,y,z
99
100     glViewport(0, 0, 400, 400);
101
102     glClearColor(1.0f, 1.0f, 1.0f, 0.0f);
103     glColor3f(0.0f, 0.0f, 1.0f);
104
105     gluQuadricDrawStyle(p, GLU_LINE);
106 }
107
108 void display()
109 {
110     glClear(GL_COLOR_BUFFER_BIT);
111
112     glMatrixMode(GL_MODELVIEW);
113     glLoadIdentity();
114     gluLookAt(eyex, eyey, eyez, // eye position x,y,z
115               0.0, 0.0, 0.0, // eye direction x,y,z
116               0.0, 1.0, 0.0);// eye up direction x,y,z
117
118     // first cylinder
119     push();
120     float base_radius = 0.5f;
121     float top_radius = 0.25f;
122     float height = 2.0f;
123     int slice_per_ring = 20;
```

```
124     int rings = 20;
125     glColor3f(1.0f, 0.0f, 0.0f);
126     gluCylinder(p, base_radius, top_radius, height, slice_per_ring, rings);
127     pop();
128 
129     // second cylinder
130     push();
131     glTranslatef(1.0f, 0.0f, 0.0f);
132     float t = 30.0f;
133     glRotatef(t, 1, 0, 0);
134     glColor3f(0.0f, 1.0f, 0.0f);
135     gluCylinder(p, 0.4f, 0.4f, 1.0f, 10, 10);
136     pop();
137 
138     // third cylinder
139     push();
140     glTranslatef(1.0f, 0.0f, 0.0f);
141     glRotatef(t, 1, 0, 0);
142     glScalef(6.0f, 7.0f, 8.0f);
143     glColor3f(0.0f, 0.0f, 1.0f);
144     gluCylinder(p, 0.1f, 0.2f, 0.5f, 10, 10);
145     pop();
146 
147     // a line
148     // Of course you can use glPushMatrix, glPopMatrix without
149     // using the above push, pop functions
150     glPushMatrix();
151     glBegin(GL_LINES);
152     glVertex3f(0, 0, 0);
153     glVertex3f(1, 1, 1);
154     glEnd();
155     glPopMatrix();
156 
157     glutSwapBuffers();
158 }
159 
160 void keyboard(unsigned char key, int x, int y)
161 {
162     switch(key)
163     {
164         case 'a': eyex -= 0.1; break;
165         case 'd': eyex += 0.1; break;
```

```
166         case 'w': eyey += 0.1; break;
167         case 's': eyey -= 0.1; break;
168 
169         // Notice that this changing eyez does not change the
170         // the image (because we're using parallel projection)
171         case 'o': eyez -= 0.1; break;
172         case 'l': eyez += 0.1; break;
173     }
174     display();
175 }
176 
177 int main(int argc, char ** argv)
178 {
179   glutInit(&argc, argv);
180   glutInitWindowPosition(100, 100);
181   glutInitWindowSize(400, 400);
182   glutInitDisplayMode(GLUT_DEPTH | GLUT_DOUBLE | GLUT_RGBA);
183   glutCreateWindow("test");
184 
185   glutDisplayFunc(display);
186   glutKeyboardFunc(keyboard);
187   init();
188   glutMainLoop();
189 
190   gluDeleteQuadric(p);
191 
192   return 0;
193 }
```

38 Solids and Lights

```
#include <GL/glut.h>

float eyex;
float eyey;
float eyez;

float lightx;
float lighty;
float lightz;

void init()
{
    eyex = 0.0f;
    eyey = 1.0f;
    eyez = 5.0f;

    lightx = 2.0f;
    lighty = 6.0f;
    lightz = 3.0f;

    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    float fovy = 90.0;
    float aspect = 1.0;
    float zNear = 3.0;
    float zFar = 100.0;
    gluPerspective(fovy, aspect, zNear, zFar);

    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    gluLookAt(eyex, eyey, eyez,
              0.0, 0.0, 0.0,
              0.0, 1.0, 0.0);

    glViewport(0, 0, 640, 480);
    glClearColor(0.0f, 0.0f, 0.0f, 0.0f);
}

void display()
```

```
40  {
41      // Set up light source properties for GL_LIGHT0.
42      // The light source can have 3 components:
43      // Ambient light:  light that's been scattered so much that the light
44      //                 seems to come from all directions. When ambient
45      //                 light  strikes a surface, it is scattered equally
46      //                 in all direction.
47      // Diffuse light:  Light coming from one direction. When it hits
48      //                 a surface, it is scattered equally in all direction.
49      // Specular light: Light coming from one direction. When it hits
50      //                 hits a surface, it bounces off in one direction.
51      GLfloat light_position[] = {lightx, lighty, lightz, 0.0f}; // 0.0 - at inf
52      GLfloat light_ambient[] = {0.1f, 0.1f, 0.1f, 1.0f};
53      GLfloat light_diffuse[] = {0.7f, 0.7f, 0.7f, 1.0f};  // max 1.0f
54      GLfloat light_specular[] = {1.0f, 1.0f, 1.0f, 1.0f}; // max 1.0f
55      glLightfv(GL_LIGHT0, GL_POSITION, light_position);
56      glLightfv(GL_LIGHT0, GL_AMBIENT, light_ambient);
57      glLightfv(GL_LIGHT0, GL_DIFFUSE, light_diffuse);
58      glLightfv(GL_LIGHT0, GL_SPECULAR, light_specular);
59
60      /* Second run
61      GLfloat light_position1[] = {-10, -1, 1, 0.0f};
62      glLightfv(GL_LIGHT1, GL_POSITION, light_position1);
63      glLightfv(GL_LIGHT1, GL_AMBIENT, light_ambient);
64      glLightfv(GL_LIGHT1, GL_DIFFUSE, light_diffuse);
65      glLightfv(GL_LIGHT1, GL_SPECULAR, light_specular);
66      */
67
68      glMatrixMode(GL_MODELVIEW);
69      glLoadIdentity();
70      gluLookAt(eyex, eyey, eyez,
71                0.0, 0.25, 0.0,
72                0.0, 1.0, 0.0);
73
74      // Begin drawing
75      glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
76
77      // Set up surface material
78      // Diffuse reflection: Most important in perceiving the color of object.
79      // Ambient reflection: Determines overall color of object.
80      // Specular reflection: Produces highlights of object.
81      // Shininess: Controls size and brightness of highlight
```

```
82      // Emission: Material that gives off light. (Example: simulate lamp)
83      GLfloat mat_ambient[] = {0.7f, 0.7f, 0.7f, 1.0f}; // gray
84      GLfloat mat_diffuse[] = {0.6f, 0.6f, 0.6f, 1.0f};
85      GLfloat mat_specular[] = {1.0f, 1.0f, 1.0f, 1.0f};
86      GLfloat mat_shininess[] = {50.0f};
87      glMaterialfv(GL_FRONT, GL_AMBIENT, mat_ambient);
88      glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_diffuse);
89      glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
90      glMaterialfv(GL_FRONT, GL_SHININESS, mat_shininess);
91
92      // Draw light source
93      glPushMatrix();
94      glTranslatef(lightx, lighty, lightz);
95      glutSolidSphere(0.1, 30, 30);
96      glPopMatrix();
97
98      glutSolidTeapot(1.0);
99
100     glTranslatef(0, 0, -3);
101
102     // Set up surface material
103     GLfloat mat_ambient2[] = {1.0f, 0.0f, 0.0f, 1.0f}; // red
104     GLfloat mat_diffuse2[] = {0.6f, 0.6f, 0.6f, 1.0f};
105     GLfloat mat_specular2[] = {0.7f, 0.7f, 0.7f, 1.0f};
106     GLfloat mat_shininess2[] = {50.0f};                // 0.0 to 128.0
107     glMaterialfv(GL_FRONT, GL_AMBIENT, mat_ambient2);
108     glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_diffuse2);
109     glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular2);
110     glMaterialfv(GL_FRONT, GL_SHININESS, mat_shininess2);
111     glutSolidTeapot(1.0);
112
113     glTranslatef(-3, 0, 3);
114     glutSolidSphere(1, 30, 30);
115
116     glTranslatef(0, 0, -3);
117     glScalef(1,3,1);
118     glutSolidSphere(1, 30, 30);
119
120     glTranslatef(6, 0, 3);
121     glutSolidCone(0.5, 2, 30, 30);
122
123     glutSwapBuffers();
```

```
124 }
125
126 void keyboard(unsigned char key, int x, int y)
127 {
128     switch(key)
129     {
130         case 'a': eyex -= 0.1; break;
131         case 'd': eyex += 0.1; break;
132         case 'w': eyey += 0.1; break;
133         case 's': eyey -= 0.1; break;
134         case '1': eyez += 0.1; break;
135         case '2': eyez -= 0.1; break;
136
137
138         case 'o': lightx -= 0.1; break;
139         case 'p': lightx += 0.1; break;
140         case 'k': lighty -= 0.1; break;
141         case 'l': lighty += 0.1; break;
142         case 'n': lightz -= 0.1; break;
143         case 'm': lightz += 0.1; break;
144     }
145     display();
146 }
147
148 int main(int argc, char ** argv)
149 {
150     glutInit(&argc, argv);
151     glutInitWindowPosition(100, 100);
152     glutInitWindowSize(640, 480);
153     glutInitDisplayMode(GLUT_DEPTH | GLUT_DOUBLE | GLUT_RGBA);
154     glutCreateWindow("test");
155
156     glEnable(GL_LIGHTING); // Enable lighting in general
157     glEnable(GL_LIGHT0);   // Enable light source GL_LIGHT0
158                            // (There are 8 light sources.)
159
160     /* Second run
161     glEnable(GL_LIGHT1);
162     */
163     glShadeModel(GL_SMOOTH); // Third run: GL_FLAT
164     glEnable(GL_DEPTH_TEST); // for hidden surface removal
165     glEnable(GL_NORMALIZE);  // normalize vectors for proper shading
```

```
166
167     glutDisplayFunc(display);
168     glutKeyboardFunc(keyboard);
169
170     init();
171     glutMainLoop();
172
173     return 0;
174 }
```

39 Solids and Local Light

```
#include <GL/glut.h>

float eyex;
float eyey;
float eyez;

float lightx;
float lighty;
float lightz;


void init()
{
    eyex = 2.0f;
    eyey = 4.0f;
    eyez = 4.0f;

    lightx = -2.0f;
    lighty = 4.0f;
    lightz = -2.0f;

    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    float fovy = 90.0;
    float aspect = 1.0;
    float zNear = 1.0;
    float zFar = 100.0;
    gluPerspective(fovy, aspect, zNear, zFar);

    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    gluLookAt(eyex, eyey, eyez,
              0.0, 5.0, 0.0,
              0.0, 1.0, 0.0);

    glViewport(0, 0, 640, 480);
    glClearColor(0.0f, 0.0f, 0.0f, 0.0f);
}

```

```
40
41
42  void table()
43  {
44      // table
45      // top
46      glPushMatrix();
47      glTranslatef(0, 3, 0);
48      glScalef(3, 0.1, 3);
49      glutSolidCube(1);
50      glPopMatrix();
51      // leg in pos x, pos z
52      glPushMatrix();
53      glTranslatef(1.4, 1.5, 1.4);
54      glScalef(0.1, 3, 0.1);
55      glutSolidCube(1);
56      glPopMatrix();
57      // leg in neg x, pos z
58      glPushMatrix();
59      glTranslatef(-1.4, 1.5, 1.4);
60      glScalef(0.1, 3, 0.1);
61      glutSolidCube(1);
62      glPopMatrix();
63      // leg in pos x, neg z
64      glPushMatrix();
65      glTranslatef(1.4, 1.5, -1.4);
66      glScalef(0.1, 3, 0.1);
67      glutSolidCube(1);
68      glPopMatrix();
69      // leg in neg x, neg z
70      glPushMatrix();
71      glTranslatef(-1.4, 1.5, -1.4);
72      glScalef(0.1, 3, 0.1);
73      glutSolidCube(1);
74      glPopMatrix();
75  }
76
77  void display()
78  {
79      glMatrixMode(GL_MODELVIEW);
80      glLoadIdentity();
81      gluLookAt(eyex, eyey, eyez,
```

```
82              0.0, 0.25, 0.0,
83              0.0, 1.0, 0.0);
84      // Begin drawing
85      glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
86
87      // Spotlight
88      // Note that that the last value of light_position is 1.0. This means
89      // that the light is a "local" light source. The other 3 values
90      // specifies the position of the light source.
91      // (Recall: If the last value is 0.0, then the light source is at
92      // infinity - i.e.infintely far away and the first 3 numbers represent
93      // the direction of all ligth rays from infinitely far away.
94      GLfloat light_position[] = {lightx, lighty, lightz, 1.0f}; // 1.0 - local
95      GLfloat dir[] = {1.0, -0.5, 1.0}; // direction of spolight
96      float cutoff_angle = 90.0;  // the maximum angle of light rays to
97                                  // the dir vector
98
99      // The local light source is at (lightx, lighty, lightz) and the
100     // light rays are restricted so that the angle made by the light rays
101     // to the direction vector <1.0, 0, 1.0> is at most the cutoff_angle.
102     //
103     // Second run: change the cutoff_angle to 45, 180, etc.
104     // Third run: change the direction of the spot light from {1.0, 0, 1.0}
105     //            to {1.0, 0, 0}, {-1.0, 0, 0}, etc.
106
107     float spot_exp = 5.0f;      // exponent of decrease of light intensity
108                                 // with distance. High means light rays die
109                                 // out faster with distance.
110     // Fourth run: Change the spot_exp to 1, 50, etc.
111
112     GLfloat light_ambient[] = {0.7f, 0.7f, 0.7f, 1.0f};
113     GLfloat light_diffuse[] = {1.0f, 1.0f, 1.0f, 1.0f};
114     GLfloat light_specular[] = {1.0f, 1.0f, 1.0f, 1.0f};
115
116     glLightf(GL_LIGHT0, GL_SPOT_CUTOFF, cutoff_angle);
117     glLightf(GL_LIGHT0, GL_SPOT_EXPONENT, spot_exp);
118     glLightfv(GL_LIGHT0, GL_SPOT_DIRECTION, dir);
119
120     glLightfv(GL_LIGHT0, GL_POSITION, light_position);
121     glLightfv(GL_LIGHT0, GL_AMBIENT, light_ambient);
122     glLightfv(GL_LIGHT0, GL_DIFFUSE, light_diffuse);
123     glLightfv(GL_LIGHT0, GL_SPECULAR, light_specular);
```

```
124
125
126     // Set up surface material
127     GLfloat mat_ambient[] = {0.7f, 0.7f, 0.7f, 1.0f}; // gray
128     GLfloat mat_diffuse[] = {1.0f, 1.0f, 1.0f, 1.0f};
129     GLfloat mat_specular[] = {1.0f, 1.0f, 1.0f, 1.0f};
130     GLfloat mat_shininess[] = {50.0f};
131     glMaterialfv(GL_FRONT, GL_AMBIENT, mat_ambient);
132     glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_diffuse);
133     glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
134     glMaterialfv(GL_FRONT, GL_SHININESS, mat_shininess);
135
136     // A sphere is drawn at the light source (so you can tell
137     // where the position of the light source is.)
138     glPushMatrix();
139     glTranslatef(lightx, lighty, lightz);
140     glutSolidSphere(0.5, 30, 30);
141     glPopMatrix();
142
143     // room
144     glPushMatrix();
145
146     // floor
147     glPushMatrix();
148     glScalef(10, 0.1, 10);
149     glutSolidCube(1);
150     glPopMatrix();
151
152     // walls
153     glPushMatrix();
154     glTranslatef(0, 5, 0);
155
156         // left
157         glPushMatrix();
158         glTranslatef(-5, 0, 0);
159         glScalef(0.1, 10, 10);
160         glutSolidCube(1);
161         glPopMatrix();
162
163         // right
164         glPushMatrix();
165         glTranslatef(5, 0, 0);
```

```
166         glScalef(0.1, 10, 10);
167         glutSolidCube(1);
168         glPopMatrix();
169 
170         // back
171         glPushMatrix();
172         glTranslatef(0, 0, -5);
173         glScalef(10, 10, 0.1);
174         glutSolidCube(1);
175         glPopMatrix();
176 
177     glPopMatrix(); // end walls
178 
179     // Fifth run: Make a ceiling for the room.
180 
181     glPopMatrix();
182 
183     // end room
184 
185     table();
186 
187     // teapot
188     glPushMatrix();
189     glTranslatef(0.5, 3.2, 0.5);
190     glScalef(0.25, 0.25, 0.25);
191     glutSolidTeapot(1.0);
192     glPopMatrix();
193 
194     // bagel
195     glPushMatrix();
196     glTranslatef(0.0, 3.2, 0.7);
197     glScalef(0.2, 0.2, 0.2);
198     glRotatef(90, 1, 0, 0); // t degrees about x axis
199     glutSolidTorus(0.2, 0.5, 10, 20);
200     glPopMatrix();
201 
202     glutSwapBuffers();
203 }
204 
205 
206 void keyboard(unsigned char key, int x, int y)
207 {
```

```
208
209     switch(key)
210     {
211         case 'a': eyex -= 0.1; break;
212         case 'd': eyex += 0.1; break;
213         case 'w': eyey += 0.1; break;
214         case 's': eyey -= 0.1; break;
215         case '1': eyez += 0.1; break;
216         case '2': eyez -= 0.1; break;
217
218         // Controls the position of light source
219         case 'o': lightx -= 0.1; break;
220         case 'p': lightx += 0.1; break;
221         case 'k': lighty -= 0.1; break;
222         case 'l': lighty += 0.1; break;
223         case 'n': lightz -= 0.1; break;
224         case 'm': lightz += 0.1; break;
225     }
226     display();
227 }
228
229 int main(int argc, char ** argv)
230 {
231     glutInit(&argc, argv);
232     glutInitWindowPosition(100, 100);
233     glutInitWindowSize(640, 480);
234     glutInitDisplayMode(GLUT_DEPTH | GLUT_DOUBLE | GLUT_RGBA);
235     glutCreateWindow("test");
236
237     glEnable(GL_LIGHTING); // Enable lighting in general
238     glEnable(GL_LIGHT0);   // Enable light source GL_LIGHT0
239                            // (There are 8 light sources.)
240     glShadeModel(GL_SMOOTH); // Third run: GL_FLAT
241     glEnable(GL_DEPTH_TEST); // for hidden surface removal
242     glEnable(GL_NORMALIZE);  // normalize vectors for proper shading
243
244     glutDisplayFunc(display);
245     glutKeyboardFunc(keyboard);
246
247
248     init();
249     glutMainLoop();
```

```
250
251     return 0;
252 }
```

40 3D Models with Primitives

Here's an example of a 3d shape made up of 3 triangles.

```
#include <GL/glut.h>

float eyex;
float eyey;
float eyez;

float lightx;
float lighty;
float lightz;

float yangle = 0;
float xangle = 0;
float zangle = 0;

void init()
{
    eyex = 0.0f;
    eyey = 4.0f;
    eyez = 10.0f;

    lightx = 0.0f;
    lighty = 4.0f;
    lightz = 0.0f;

    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    float fovy = 90.0;
    float aspect = 1.0;
    float zNear = 1.0;
    float zFar = 100.0;
    gluPerspective(fovy, aspect, zNear, zFar);

    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    gluLookAt(eyex, eyey, eyez,
              0.0, 4.0, -10.0,
              0.0, 1.0, 0.0);
```

```
38
39      glViewport(0, 0, 640, 480);
40      glClearColor(0.0f, 0.0f, 0.0f, 0.0f);
41  }
42
43
44
45  void spotlight()
46  {
47      GLfloat light_position[] = {lightx, lighty, lightz, 1.0f};
48      GLfloat dir[] = {1.0, -1, 1.0};
49      float cutoff_angle = 90.0;
50      float spot_exp = 5.0f;
51
52      GLfloat light_ambient[] = {0.7f, 0.7f, 0.7f, 1.0f};
53      GLfloat light_diffuse[] = {1.0f, 1.0f, 1.0f, 1.0f};
54      GLfloat light_specular[] = {1.0f, 1.0f, 1.0f, 1.0f};
55
56      glLightf(GL_LIGHT0, GL_SPOT_CUTOFF, cutoff_angle);
57      glLightf(GL_LIGHT0, GL_SPOT_EXPONENT, spot_exp);
58      glLightfv(GL_LIGHT0, GL_SPOT_DIRECTION, dir);
59
60      glLightfv(GL_LIGHT0, GL_POSITION, light_position);
61      glLightfv(GL_LIGHT0, GL_AMBIENT, light_ambient);
62      glLightfv(GL_LIGHT0, GL_DIFFUSE, light_diffuse);
63      glLightfv(GL_LIGHT0, GL_SPECULAR, light_specular);
64
65      glPushMatrix();
66      glTranslatef(lightx, lighty, lightz);
67      glutSolidSphere(0.5, 30, 30);
68      glPopMatrix();
69  }
70
71
72  void shape()
73  {
74      { // triangle with slanting surface
75          GLfloat mat_ambient[] = {0.7f, 0.0f, 0.0f, 1.0f};
76          GLfloat mat_diffuse[] = {1.0f, 1.0f, 1.0f, 1.0f};
77          GLfloat mat_specular[] = {1.0f, 1.0f, 1.0f, 1.0f};
78          GLfloat mat_shininess[] = {50.0f};
79          glMaterialfv(GL_FRONT, GL_AMBIENT, mat_ambient);
```

```
80          glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_diffuse);
81          glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
82          glMaterialfv(GL_FRONT, GL_SHININESS, mat_shininess);
83
84          glBegin(GL_TRIANGLES);
85          glNormal3f(1, 1, 1);
86          glVertex3f(0, 1, 0);
87          glVertex3f(0, 0, 1);
88          glVertex3f(1, 0, 0);
89          glEnd();
90      }
91      { // triangle in yz-plane
92          GLfloat mat_ambient[] = {0.0f, 1.0f, 0.0f, 1.0f};
93          GLfloat mat_diffuse[] = {1.0f, 1.0f, 1.0f, 1.0f};
94          GLfloat mat_specular[] = {1.0f, 1.0f, 1.0f, 1.0f};
95          GLfloat mat_shininess[] = {50.0f};
96          glMaterialfv(GL_FRONT, GL_AMBIENT, mat_ambient);
97          glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_diffuse);
98          glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
99          glMaterialfv(GL_FRONT, GL_SHININESS, mat_shininess);
100
101         glBegin(GL_TRIANGLES);
102         glNormal3f(-1, 0, 0);
103         glVertex3f(0, 1, 0);
104         glVertex3f(0, 0, 0);
105         glVertex3f(0, 0, 1);
106         glEnd();
107     }
108     { // triangle in xy-plane
109         GLfloat mat_ambient[] = {0.0f, 0.0f, 0.7f, 1.0f};
110         GLfloat mat_diffuse[] = {0.0f, 0.0f, 0.7f, 1.0f};
111         GLfloat mat_specular[] = {1.0f, 1.0f, 1.0f, 1.0f};
112         GLfloat mat_shininess[] = {50.0f};
113         glMaterialfv(GL_FRONT, GL_AMBIENT, mat_ambient);
114         glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_diffuse);
115         glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
116         glMaterialfv(GL_FRONT, GL_SHININESS, mat_shininess);
117
118         glBegin(GL_TRIANGLES);
119         glNormal3f(0, 0, 1);
120         glVertex3f(0, 1, 0);
121         glVertex3f(1, 0, 0);
```

```
122         glVertex3f(0, 0, 0);
123         glEnd();
124     }
125 }
126 
127 
128 
129 void display()
130 {
131     glMatrixMode(GL_MODELVIEW);
132     glLoadIdentity();
133     gluLookAt(eyex, eyey, eyez,
134               0.0, 0.25, 0.0,
135               0.0, 1.0, 0.0);
136 
137     // Begin drawing
138     glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
139 
140     // Set up surface material
141     GLfloat mat_ambient[] = {0.7f, 0.7f, 0.7f, 1.0f}; // gray
142     GLfloat mat_diffuse[] = {1.0f, 1.0f, 1.0f, 1.0f};
143     GLfloat mat_specular[] = {1.0f, 1.0f, 1.0f, 1.0f};
144     GLfloat mat_shininess[] = {50.0f};
145     glMaterialfv(GL_FRONT, GL_AMBIENT, mat_ambient);
146     glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_diffuse);
147     glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
148     glMaterialfv(GL_FRONT, GL_SHININESS, mat_shininess);
149 
150     glPushMatrix();
151 
152         // user view
153         glRotatef(xangle, 1, 0, 0);
154         glRotatef(yangle, 0, 1, 0);
155         glRotatef(zangle, 0, 0, 1);
156 
157         spotlight();
158 
159         glPushMatrix();
160         glScalef(3, 3, 3);
161         shape();
162         glPopMatrix();
163 
```

```
164     glPopMatrix();
165
166     glutSwapBuffers();
167 }
168
169
170 void keyboard(unsigned char key, int x, int y)
171 {
172     switch(key)
173     {
174         case 'a': eyex -= 0.1; break;
175         case 'd': eyex += 0.1; break;
176         case 'w': eyey += 0.1; break;
177         case 's': eyey -= 0.1; break;
178         case '1': eyez += 0.1; break;
179         case '2': eyez -= 0.1; break;
180
181         // Controls the position of light source
182         case 'o': lightx -= 0.1; break;
183         case 'p': lightx += 0.1; break;
184         case 'k': lighty -= 0.1; break;
185         case 'l': lighty += 0.1; break;
186         case 'n': lightz -= 0.1; break;
187         case 'm': lightz += 0.1; break;
188
189         // rotations
190         case 'r': xangle = (xangle == 359 ?   0 : xangle + 1); break;
191         case 't': xangle = (xangle ==   0 ? 359 : xangle - 1); break;
192         case 'f': yangle = (yangle == 359 ?   0 : yangle + 1); break;
193         case 'g': yangle = (yangle ==   0 ? 359 : yangle - 1); break;
194         case 'v': zangle = (zangle == 359 ?   0 : zangle + 1); break;
195         case 'b': zangle = (zangle ==   0 ? 359 : zangle - 1); break;
196     }
197     display();
198 }
199
200 int main(int argc, char ** argv)
201 {
202     glutInit(&argc, argv);
203     glutInitWindowPosition(100, 100);
204     glutInitWindowSize(640, 480);
205     glutInitDisplayMode(GLUT_DEPTH | GLUT_DOUBLE | GLUT_RGBA);
```

```
206     glutCreateWindow("test");
207
208     glEnable(GL_LIGHTING); // Enable lighting in general
209     glEnable(GL_LIGHT0);   // Enable light source GL_LIGHT0
210                            // (There are 8 light sources.)
211     glShadeModel(GL_SMOOTH); // Third run: GL_FLAT
212     glEnable(GL_DEPTH_TEST); // for hidden surface removal
213     glEnable(GL_NORMALIZE);  // normalize vectors for proper shading
214
215     // Second run: comment out the next 2 statements
216     glFrontFace(GL_CCW);    // counterclockwise polygons face out
217     glEnable(GL_CULL_FACE); // do not compute inside
218
219     glutDisplayFunc(display);
220     glutKeyboardFunc(keyboard);
221
222     init();
223     glutMainLoop();
224
225     return 0;
226 }
```

41 3D Models with Primitives: Jet Propulsion Engine

Here's an example of a 3d shape. It's basically a cylinder with thickness. We make use the various symmetries in the shape. After drawing the basic shape, we create the whole cylinder by performing various rotations to form the whole shape.

```
#include <cmath>
#include <GL/glut.h>

float eyex;
float eyey;
float eyez;

float lightx;
float lighty;
float lightz;

float yangle = 0;
float xangle = 0;
float zangle = 0;

void init()
{
    eyex = 0.0f;
    eyey = 4.0f;
    eyez = 10.0f;

    lightx = 0.0f;
    lighty = 4.0f;
    lightz = 0.0f;

    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    float fovy = 90.0;
    float aspect = 1.0;
    float zNear = 1.0;
    float zFar = 100.0;
    gluPerspective(fovy, aspect, zNear, zFar);

    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
```

```
36      gluLookAt(eyex, eyey, eyez,
37                0.0, 4.0, -10.0,
38                0.0, 1.0, 0.0);
39
40      glViewport(0, 0, 640, 480);
41      glClearColor(0.0f, 0.0f, 0.0f, 0.0f);
42  }
43
44
45  void axes()
46  {
47      GLUquadricObj * p = gluNewQuadric();
48      gluQuadricDrawStyle(p, GLU_FILL);
49      glPushMatrix();
50
51      for (int j = 0; j < 3; j++)
52      {
53          glPushMatrix();
54          if (j == 0) {}
55          else if (j == 1) { glRotatef(90, 0, 1, 0); }
56          else if (j == 2) { glRotatef(-90, 1, 0, 0); }
57          glBegin(GL_LINES);
58          glVertex3f(0, 0, 0);
59          glVertex3f(0, 0, 100);
60          glEnd();
61
62          for (int i = 0; i < 30; i++)
63          {
64              glPushMatrix();
65              glTranslatef(0, 0, 3*i);
66              gluCylinder(p, 0.2, 0, 1, 10, 10);
67              glPopMatrix();
68          }
69          glPopMatrix();
70      }
71
72      glPopMatrix();
73      gluDeleteQuadric(p);
74  }
75
76
77  void spotlight()
```

```
78  {
79      GLfloat light_position[] = {lightx, lighty, lightz, 1.0f};
80      GLfloat dir[] = {1.0, -1, 1.0};
81      float cutoff_angle = 90.0;
82      float spot_exp = 5.0f;
83
84      GLfloat light_ambient[] = {0.8f, 0.8f, 0.8f, 1.0f};
85      GLfloat light_diffuse[] = {1.0f, 1.0f, 1.0f, 1.0f};
86      GLfloat light_specular[] = {1.0f, 1.0f, 1.0f, 1.0f};
87
88      glLightf(GL_LIGHT0, GL_SPOT_CUTOFF, cutoff_angle);
89      glLightf(GL_LIGHT0, GL_SPOT_EXPONENT, spot_exp);
90      glLightfv(GL_LIGHT0, GL_SPOT_DIRECTION, dir);
91
92      glLightfv(GL_LIGHT0, GL_POSITION, light_position);
93      glLightfv(GL_LIGHT0, GL_AMBIENT, light_ambient);
94      glLightfv(GL_LIGHT0, GL_DIFFUSE, light_diffuse);
95      glLightfv(GL_LIGHT0, GL_SPECULAR, light_specular);
96
97      // A sphere is drawn at the light source (so you can tell
98      // where the position of the light source is.)
99      glPushMatrix();
100     glTranslatef(lightx, lighty, lightz);
101     glutSolidSphere(0.5, 30, 30);
102     glPopMatrix();
103 }
104
105
106 void booster()
107 {
108     GLfloat mat_ambient[] = {0.9f, 0.9f, 0.9f, 1.0f};
109     GLfloat mat_diffuse[] = {1.0f, 1.0f, 1.0f, 1.0f};
110     GLfloat mat_specular[] = {1.0f, 1.0f, 1.0f, 1.0f};
111     GLfloat mat_shininess[] = {50.0f};
112     glMaterialfv(GL_FRONT, GL_AMBIENT, mat_ambient);
113     glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_diffuse);
114     glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
115     glMaterialfv(GL_FRONT, GL_SHININESS, mat_shininess);
116
117     int numpieces = 100;
118     int angle = atan(0.1) * 180 / 3.14159265;
119     glPushMatrix();
```

```
120     for (int i = 0; i < 360/angle; i++)
121     {
122         glPushMatrix();
123         glTranslatef(0, 1, 0);
124         for (int j = 0; j < 181; j += 180)
125         {
126             glPushMatrix();
127             glRotatef(j, 0, 1, 0);
128 
129             glBegin(GL_QUADS);
130             glNormal3f( 0, 1, 0); glVertex3f( 0,   0.125, -0.1);
131             glNormal3f( 0, 1, 0); glVertex3f( 0,   0.125,  0.1);
132             glNormal3f( 0, 1, 0); glVertex3f( 2,   0.125,  0.1);
133             glNormal3f( 0, 1, 0); glVertex3f( 2,   0.125, -0.1);
134 
135             glVertex3f(  2,  0.125, -0.1);
136             glVertex3f(  2,  0.125,  0.1);
137             glVertex3f(2.05,  0.05,  0.1);
138             glVertex3f(2.05,  0.05, -0.1);
139 
140             glVertex3f(2.05,  0.05, -0.1);
141             glVertex3f(2.05,  0.05,  0.1);
142             glVertex3f(2.05, -0.05,  0.1);
143             glVertex3f(2.05, -0.05, -0.1);
144 
145             glVertex3f(2.05, -0.05, -0.1);
146             glVertex3f(2.05, -0.05,  0.1);
147             glVertex3f(2.0, -0.125,  0.1);
148             glVertex3f(2.0, -0.125, -0.1);
149 
150             glVertex3f(2.0, -0.125, -0.1);
151             glVertex3f(2.0, -0.125,  0.1);
152             glVertex3f(0.0, -0.125,  0.1);
153             glVertex3f(0.0, -0.125, -0.1);
154             glEnd();
155 
156             glPopMatrix();
157         }
158         glPopMatrix();
159 
160         glRotatef(angle, 1, 0, 0);
161     }
```

```
162     glPopMatrix();
163 }
164
165
166 void display()
167 {
168     glMatrixMode(GL_MODELVIEW);
169     glLoadIdentity();
170     gluLookAt(eyex, eyey, eyez,
171               0.0, 0.25, 0.0,
172               0.0, 1.0, 0.0);
173
174     // Begin drawing
175     glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
176
177     // Set up surface material
178     GLfloat mat_ambient[] = {0.7f, 0.7f, 0.7f, 1.0f}; // gray
179     GLfloat mat_diffuse[] = {1.0f, 1.0f, 1.0f, 1.0f};
180     GLfloat mat_specular[] = {1.0f, 1.0f, 1.0f, 1.0f};
181     GLfloat mat_shininess[] = {50.0f};
182     glMaterialfv(GL_FRONT, GL_AMBIENT, mat_ambient);
183     glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_diffuse);
184     glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
185     glMaterialfv(GL_FRONT, GL_SHININESS, mat_shininess);
186
187     glPushMatrix();
188         // use view
189         glRotatef(xangle, 1, 0, 0);
190         glRotatef(yangle, 0, 1, 0);
191         glRotatef(zangle, 0, 0, 1);
192
193         spotlight();
194         axes();
195
196         glPushMatrix();
197         glScalef(3, 3, 3);
198         booster();
199         glPopMatrix();
200
201     glPopMatrix();
202
203     glutSwapBuffers();
```

```
204 }
205
206
207 void keyboard(unsigned char key, int x, int y)
208 {
209     switch(key)
210     {
211         case 'a': eyex -= 0.1; break;
212         case 'd': eyex += 0.1; break;
213         case 'w': eyey += 0.1; break;
214         case 's': eyey -= 0.1; break;
215         case '1': eyez += 0.1; break;
216         case '2': eyez -= 0.1; break;
217
218         // Controls the position of light source
219         case 'o': lightx -= 0.1; break;
220         case 'p': lightx += 0.1; break;
221         case 'k': lighty -= 0.1; break;
222         case 'l': lighty += 0.1; break;
223         case 'n': lightz -= 0.1; break;
224         case 'm': lightz += 0.1; break;
225
226         // rotations
227         case 'r': xangle = (xangle == 359 ?   0 : xangle + 1); break;
228         case 't': xangle = (xangle ==   0 ? 359 : xangle - 1); break;
229         case 'f': yangle = (yangle == 359 ?   0 : yangle + 1); break;
230         case 'g': yangle = (yangle ==   0 ? 359 : yangle - 1); break;
231         case 'v': zangle = (zangle == 359 ?   0 : zangle + 1); break;
232         case 'b': zangle = (zangle ==   0 ? 359 : zangle - 1); break;
233     }
234     display();
235 }
236
237 int main(int argc, char ** argv)
238 {
239     glutInit(&argc, argv);
240     glutInitWindowPosition(100, 100);
241     glutInitWindowSize(640, 480);
242     glutInitDisplayMode(GLUT_DEPTH | GLUT_DOUBLE | GLUT_RGBA);
243     glutCreateWindow("test");
244
245     glEnable(GL_LIGHTING); // Enable lighting in general
```

```
246     glEnable(GL_LIGHT0);   // Enable light source GL_LIGHT0
247                            // (There are 8 light sources.)
248     glShadeModel(GL_SMOOTH); // Third run: GL_FLAT
249     glEnable(GL_DEPTH_TEST); // for hidden surface removal
250     glEnable(GL_NORMALIZE);  // normalize vectors for proper shading
251 
252     //glFrontFace(GL_CCW);    // counterclockwise polygons face out
253     //glEnable(GL_CULL_FACE); // do not compute inside
254 
255     glutDisplayFunc(display);
256     glutKeyboardFunc(keyboard);
257 
258     init();
259     glutMainLoop();
260 
261     return 0;
262 }
```

42 Basic physics modeling of graphical object

```
#include <ctime>
#include <iostream>
#include <GL/glut.h>
#include <iomanip>
#include <cmath>

float eyex;
float eyey;
float eyez;

float lightx;
float lighty;
float lightz;
time_t mytime; // Time of last update to ball's physics


//---------------------------------------------------------------------------
// A minimalistic vector class for each physics computation
//---------------------------------------------------------------------------
class vec3f
{
public:
    vec3f(float x0=0.0f, float y0=0.0f, float z0=0.0f)
        : x(x0), y(y0), z(z0)
    {}
    vec3f & operator+=(const vec3f & v) { x += v.x; y += v.y; z += v.z; }
    vec3f   operator+(const vec3f & v) { return (vec3f(*this) += v); }
    vec3f & operator-=(const vec3f & v) { x -= v.x; y -= v.y; z -= v.z; }
    vec3f   operator-(const vec3f & v) { return (vec3f(*this) -= v); }
    vec3f & operator*=(float f) { x *= f; y *= f; z *= f; }
    vec3f   operator*(float f) { return (vec3f(*this) *= f); }
    vec3f   operator-() { return vec3f(-x, -y, -z); }
    float & operator[](int i) { return (i == 0 ? x :
                                        i == 1 ? y :
                                                 z); }
private:
    float x, y, z;
};
vec3f operator*(float f, const vec3f & v) { return vec3f(v) * f; }
```

```
40 std::ostream & operator<<(std::ostream & cout, const vec3f & v)
41 {
42     vec3f u(v);
43     cout << '<' << u[0] << ", " << u[1] << ", " << u[2] << '>';
44     return cout;
45 }
46
47 float len(const vec3f & v)
48 {
49     vec3f u(v);
50     return sqrt(u[0] * u[0] + u[1] * u[1] + u[2] * u[2]);
51 }
52
53
54
55 // Ball
56 vec3f p; // position vector
57 vec3f v; // velocity vector
58 vec3f a; // acceleration vector
59
60
61
62 void init()
63 {
64     // Initialize physics of the ball
65     p = vec3f(0, 3, 0);
66     v = vec3f(1, 0, 0.5);
67     a = vec3f(0, -32, 0);
68
69     eyex = 4.0f;
70     eyey = 1.0f;
71     eyez = 8.0f;
72
73     lightx = -2.0f;
74     lighty = 4.0f;
75     lightz = -2.0f;
76
77     glMatrixMode(GL_PROJECTION);
78     glLoadIdentity();
79     float fovy = 90.0;
80     float aspect = 1.0;
81     float zNear = 1.0;
```

```
82      float zFar = 100.0;
83      gluPerspective(fovy, aspect, zNear, zFar);
84
85      glMatrixMode(GL_MODELVIEW);
86      glLoadIdentity();
87      gluLookAt(eyex, eyey, eyez,
88                0.0, 5.0, 0.0,
89                0.0, 1.0, 0.0);
90
91      glViewport(0, 0, 640, 480);
92      glClearColor(0.0f, 0.0f, 0.0f, 0.0f);
93  }
94
95
96  void display()
97  {
98      glMatrixMode(GL_MODELVIEW);
99      glLoadIdentity();
100     gluLookAt(eyex, eyey, eyez,
101               0.0, 0.25, 0.0,
102               0.0, 1.0, 0.0);
103     glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
104
105     GLfloat light_position[] = {lightx, lighty, lightz, 1.0f};
106     GLfloat dir[] = {1.0, -0.5, 1.0};
107     float cutoff_angle = 90.0;
108
109     float spot_exp = 5.0f;
110
111     GLfloat light_ambient[] = {0.7f, 0.7f, 0.7f, 1.0f};
112     GLfloat light_diffuse[] = {1.0f, 1.0f, 1.0f, 1.0f};
113     GLfloat light_specular[] = {1.0f, 1.0f, 1.0f, 1.0f};
114
115     glLightf(GL_LIGHT0, GL_SPOT_CUTOFF, cutoff_angle);
116     glLightf(GL_LIGHT0, GL_SPOT_EXPONENT, spot_exp);
117     glLightfv(GL_LIGHT0, GL_SPOT_DIRECTION, dir);
118
119     glLightfv(GL_LIGHT0, GL_POSITION, light_position);
120     glLightfv(GL_LIGHT0, GL_AMBIENT, light_ambient);
121     glLightfv(GL_LIGHT0, GL_DIFFUSE, light_diffuse);
122     glLightfv(GL_LIGHT0, GL_SPECULAR, light_specular);
123
```

```
124
125     // Set up surface material
126     GLfloat mat_ambient[] = {0.7f, 0.7f, 0.7f, 1.0f}; // gray
127     GLfloat mat_diffuse[] = {1.0f, 1.0f, 1.0f, 1.0f};
128     GLfloat mat_specular[] = {1.0f, 1.0f, 1.0f, 1.0f};
129     GLfloat mat_shininess[] = {50.0f};
130     glMaterialfv(GL_FRONT, GL_AMBIENT, mat_ambient);
131     glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_diffuse);
132     glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
133     glMaterialfv(GL_FRONT, GL_SHININESS, mat_shininess);
134
135     // A sphere is drawn at the light source (so you can tell
136     // where the position of the light source is.)
137     glPushMatrix();
138     glTranslatef(lightx, lighty, lightz);
139     glutSolidSphere(0.5, 30, 30);
140     glPopMatrix();
141
142     // begin room
143     glPushMatrix();
144
145     // floor
146     glPushMatrix();
147     glScalef(10, 0.1, 10);
148     glutSolidCube(1);
149     glPopMatrix();
150
151     // walls
152     glPushMatrix();
153     glTranslatef(0, 5, 0);
154
155         // left
156         glPushMatrix();
157         glTranslatef(-5, 0, 0);
158         glScalef(0.1, 10, 10);
159         glutSolidCube(1);
160         glPopMatrix();
161
162         // right
163         glPushMatrix();
164         glTranslatef(5, 0, 0);
165         glScalef(0.1, 10, 10);
```

```
166         glutSolidCube(1);
167         glPopMatrix();
168 
169         // back
170         glPushMatrix();
171         glTranslatef(0, 0, -5);
172         glScalef(10, 10, 0.1);
173         glutSolidCube(1);
174         glPopMatrix();
175 
176     glPopMatrix(); // end walls
177 
178     glPopMatrix(); // end room
179 
180     {
181         // Draw the ball
182         GLfloat mat_ambient[] = {0.9f, 0.0f, 0.0f, 1.0f};
183         GLfloat mat_diffuse[] = {0.9f, 0.0f, 0.0f, 1.0f};
184         GLfloat mat_specular[] = {0.9f, 0.0f, 0.0f, 1.0f};
185         GLfloat mat_shininess[] = {50.0f};
186         glMaterialfv(GL_FRONT, GL_AMBIENT, mat_ambient);
187         glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_diffuse);
188         glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
189         glMaterialfv(GL_FRONT, GL_SHININESS, mat_shininess);
190         glPushMatrix();
191         glTranslatef(p[0], p[1], p[2]);
192         glutSolidSphere(0.2, 30, 30);
193         glPopMatrix();
194     }
195 
196     glutSwapBuffers();
197 }
198 
199 
200 void keyboard(unsigned char key, int x, int y)
201 {
202     switch(key)
203     {
204         case 'a': eyex -= 0.1; break;
205         case 'd': eyex += 0.1; break;
206         case 'w': eyey += 0.1; break;
207         case 's': eyey -= 0.1; break;
```

```
208         case '1': eyez += 0.1; break;
209         case '2': eyez -= 0.1; break;
210 
211         // Controls the position of light source
212         case 'o': lightx -= 0.1; break;
213         case 'p': lightx += 0.1; break;
214         case 'k': lighty -= 0.1; break;
215         case 'l': lighty += 0.1; break;
216         case 'n': lightz -= 0.1; break;
217         case 'm': lightz += 0.1; break;
218     }
219     display();
220 }
221 
222 
223 void idle()
224 {
225     double dt = difftime(time(NULL), mytime) / 1000; // time since last
226                                                      // physics update of
227                                                      // ball
228     std::cout << "idle ... " << dt << std::endl;
229     if (dt <= 0) return;
230     mytime += dt;
231 
232     // Compute new position and velocity vector. (Acceleration is assumed
233     // fixed and due to gravity. Otherwise you have to compute the sum
234     // of forces acting on the ball and divide by the mass of the ball
235     // to get the new acceleration vector.)
236     v += a * dt;
237     p += v * dt;
238 
239     // Collision detection against the 4 side walls and floor. (No
240     // collision detection for ceiling).
241     float a = 0.1/2 + 0.2; // half of thickness of wall and radius
242     if (p[1] < a)      { p[1] =  a    ; v[1] = -v[1] * 0.9; } // floor
243     if (p[0] < -5 + a) { p[0] = -5 + a; v[0] = -v[0] * 0.9; } // left wall
244     if (p[0] > 5 - a)  { p[0] =  5 - a; v[0] = -v[0] * 0.9; } // right wall
245     if (p[2] < -5 + a) { p[2] = -5 + a; v[2] = -v[2] * 0.9; } // back wall
246     if (p[2] > 5 - a)  { p[2] =  5 - a; v[2] = -v[2] * 0.9; } // front
247 
248     // A dampening factor of 0.9 is used for each collision.
249 
```

```
250     std::cout << dt << " ... "
251               << v << " ... " << len(v) << std::endl;
252     display();
253 }
254 
255 
256 int main(int argc, char ** argv)
257 {
258     glutInit(&argc, argv);
259     glutInitWindowPosition(100, 100);
260     glutInitWindowSize(640, 480);
261     glutInitDisplayMode(GLUT_DEPTH | GLUT_DOUBLE | GLUT_RGBA);
262     glutCreateWindow("test");
263 
264     glEnable(GL_LIGHTING); // Enable lighting in general
265     glEnable(GL_LIGHT0);   // Enable light source GL_LIGHT0
266                            // (There are 8 light sources.)
267     glShadeModel(GL_SMOOTH); // Third run: GL_FLAT
268     glEnable(GL_DEPTH_TEST); // for hidden surface removal
269     glEnable(GL_NORMALIZE);  // normalize vectors for proper shading
270 
271     glutDisplayFunc(display);
272     glutKeyboardFunc(keyboard);
273     glutIdleFunc(idle);
274 
275     init();
276     mytime = time(NULL);
277     glutMainLoop();
278 
279     return 0;
280 }
```

43 Windows and Microsoft Visual Studio 2010