CS559: Computer Graphics

Lecture 12: OpenGL

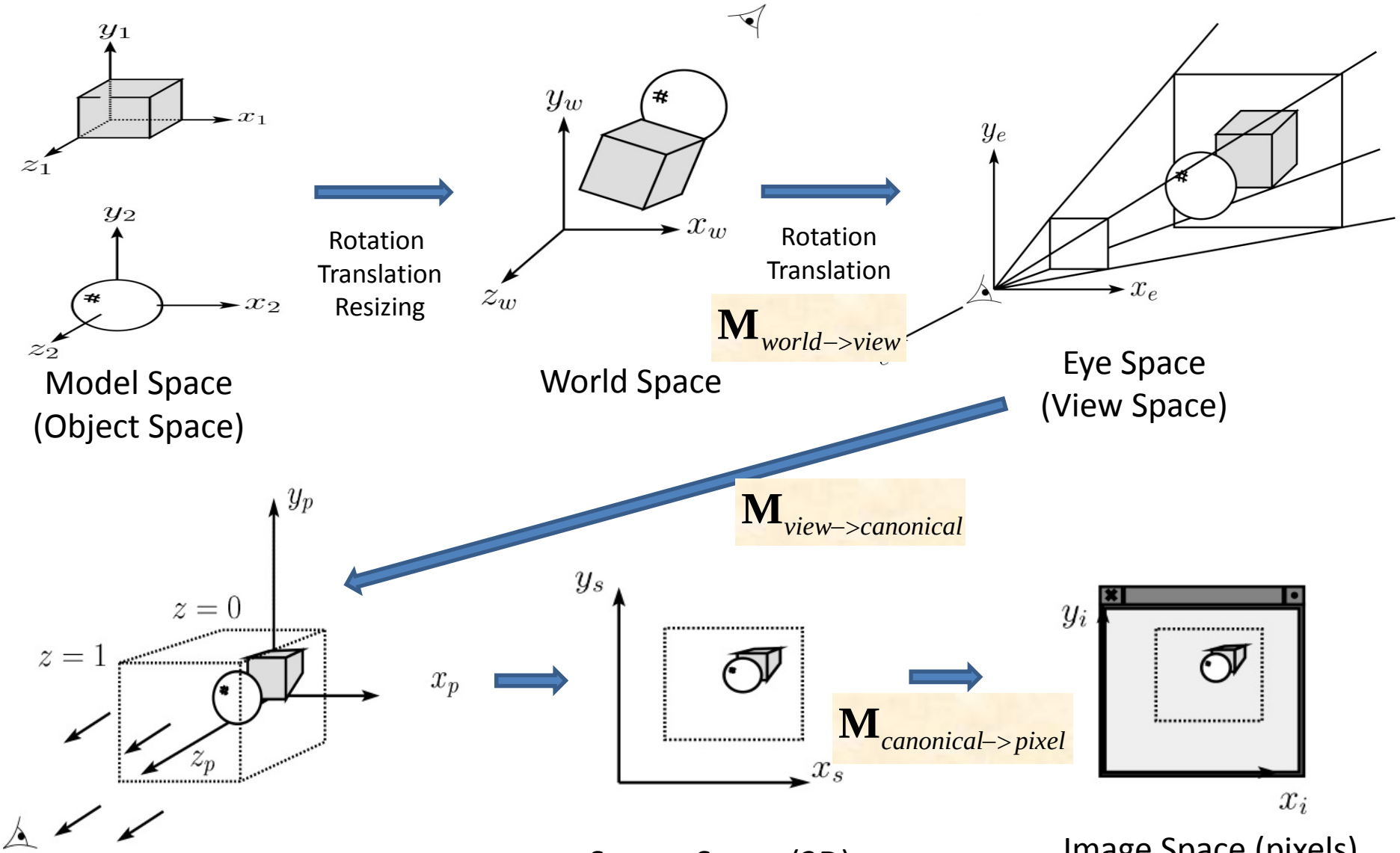
Li Zhang

Spring 2008

Reading

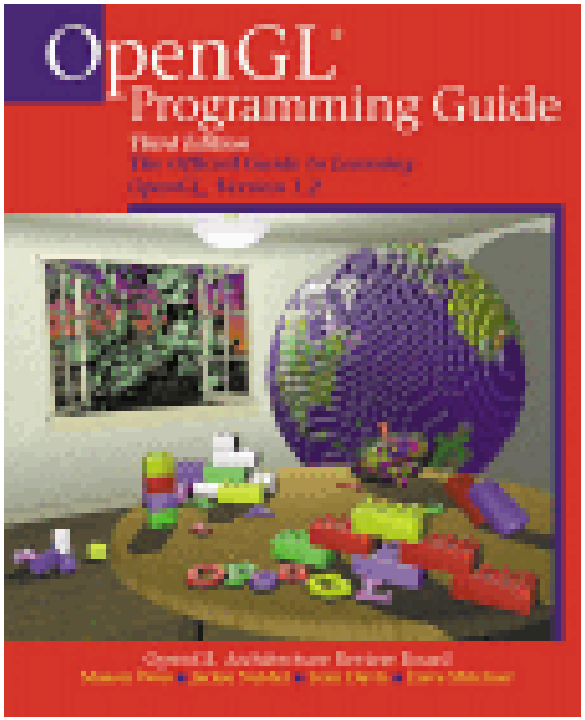
- Redbook
  - Ch 1 & 2

So far: 3D Geometry Pipeline

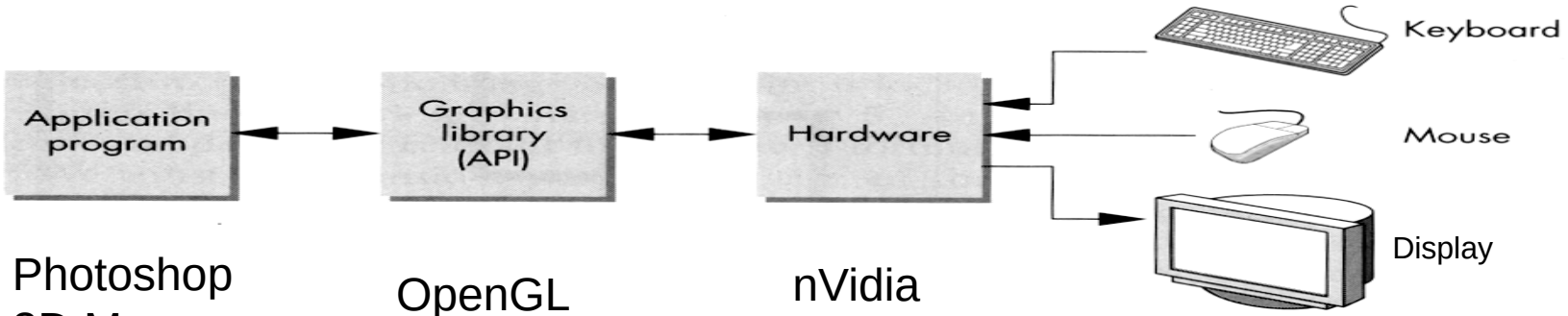


OpenGL

- We have been focused on math description
- We'll move on practical graphics programming for a week



Modern graphics systems



Photoshop
3D Max
Softimage
Maya
and ?

OpenGL
Direct3D
…

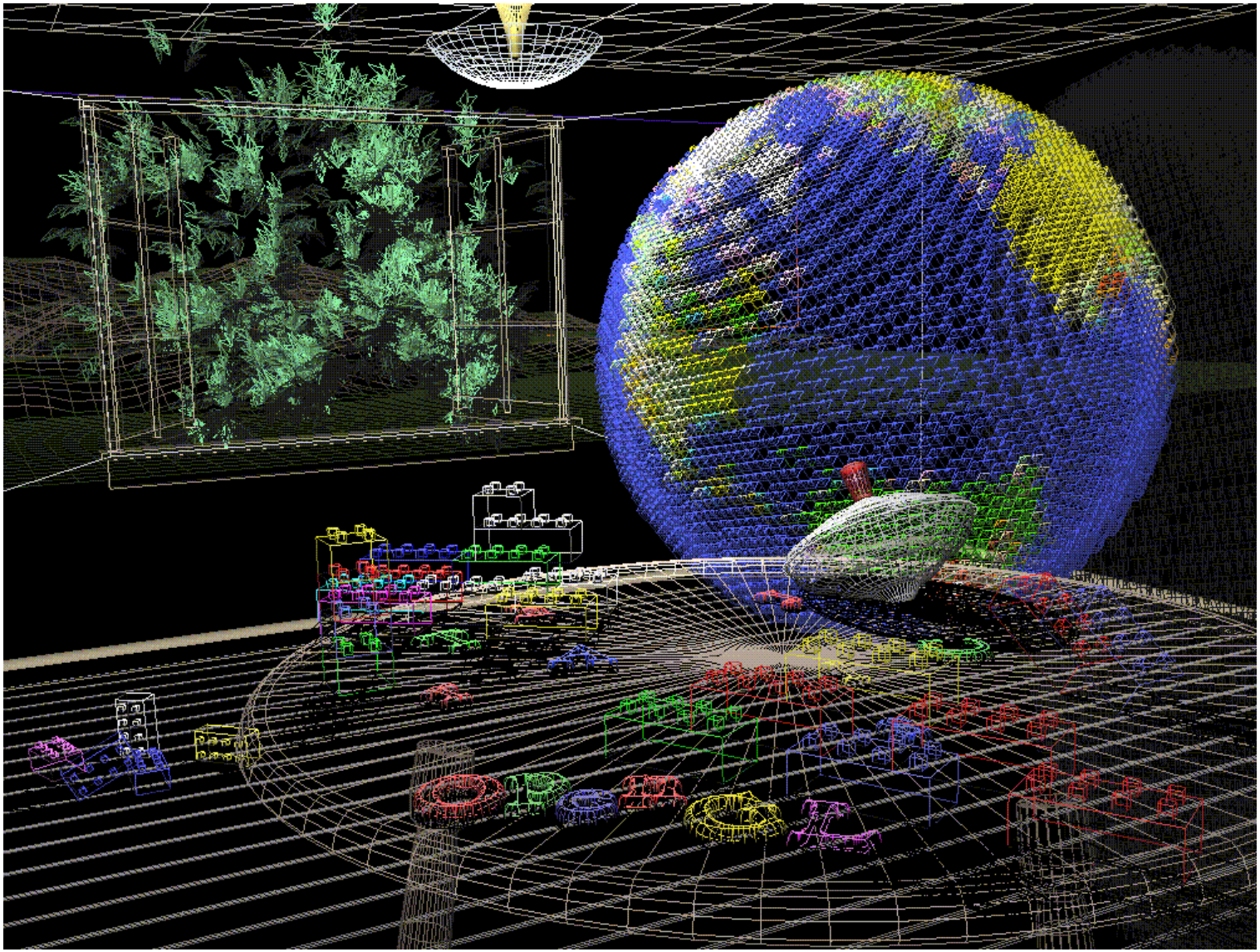
nVidia
ATI
…

Your homework

OpenGL

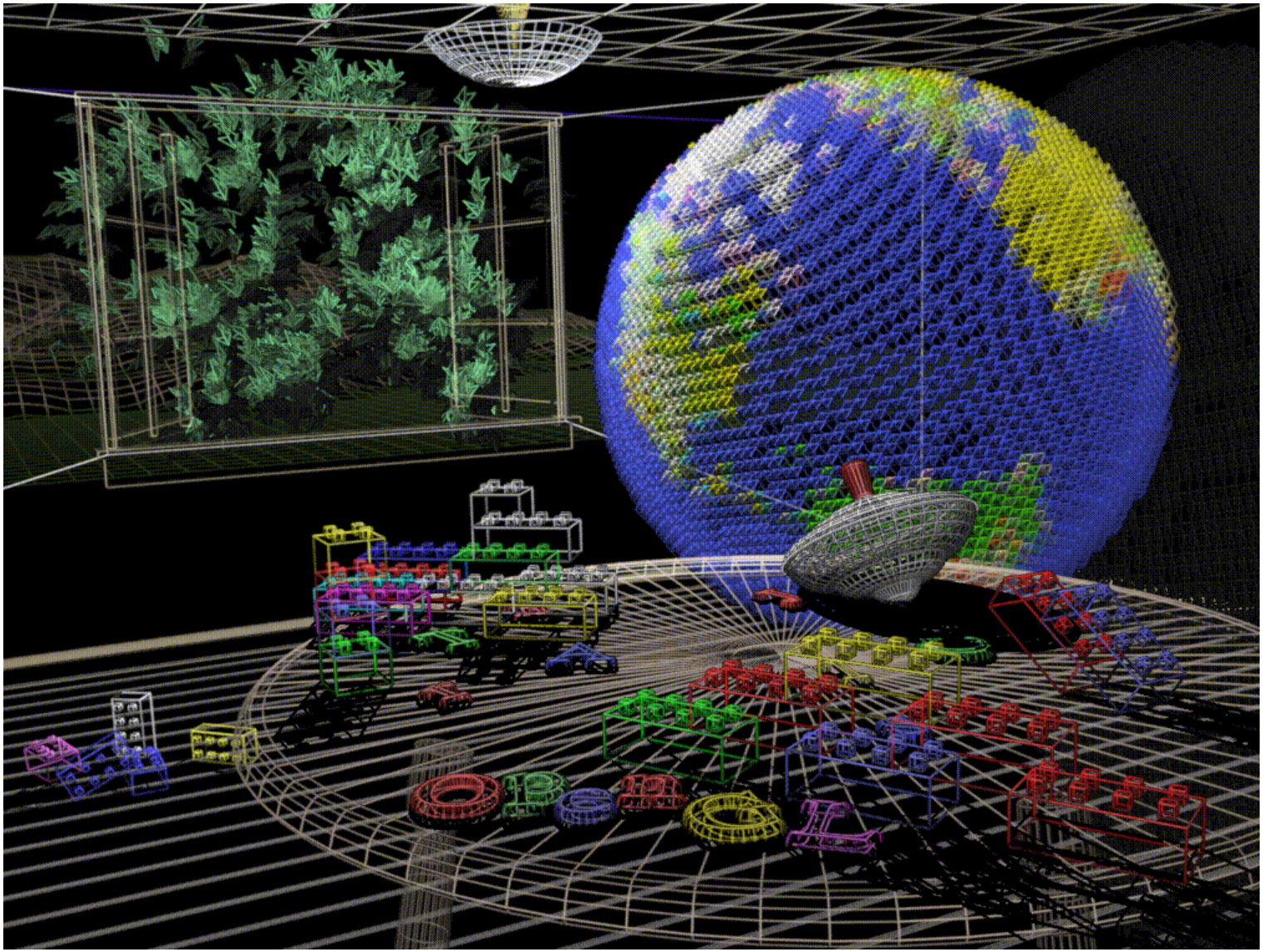
- A software interface to graphics hardware.
  - 700 distinct commands
    - 650 in the core OpenGL
    - 50 in the utility library
  - Specify objects, viewing, lighting, surface material
- Hardware independent interface
  - No commands for windowing tasks
  - No high level object models
    - You need to specify geometric primitives
      - Points, lines, polygons.

What can OpenGL do?



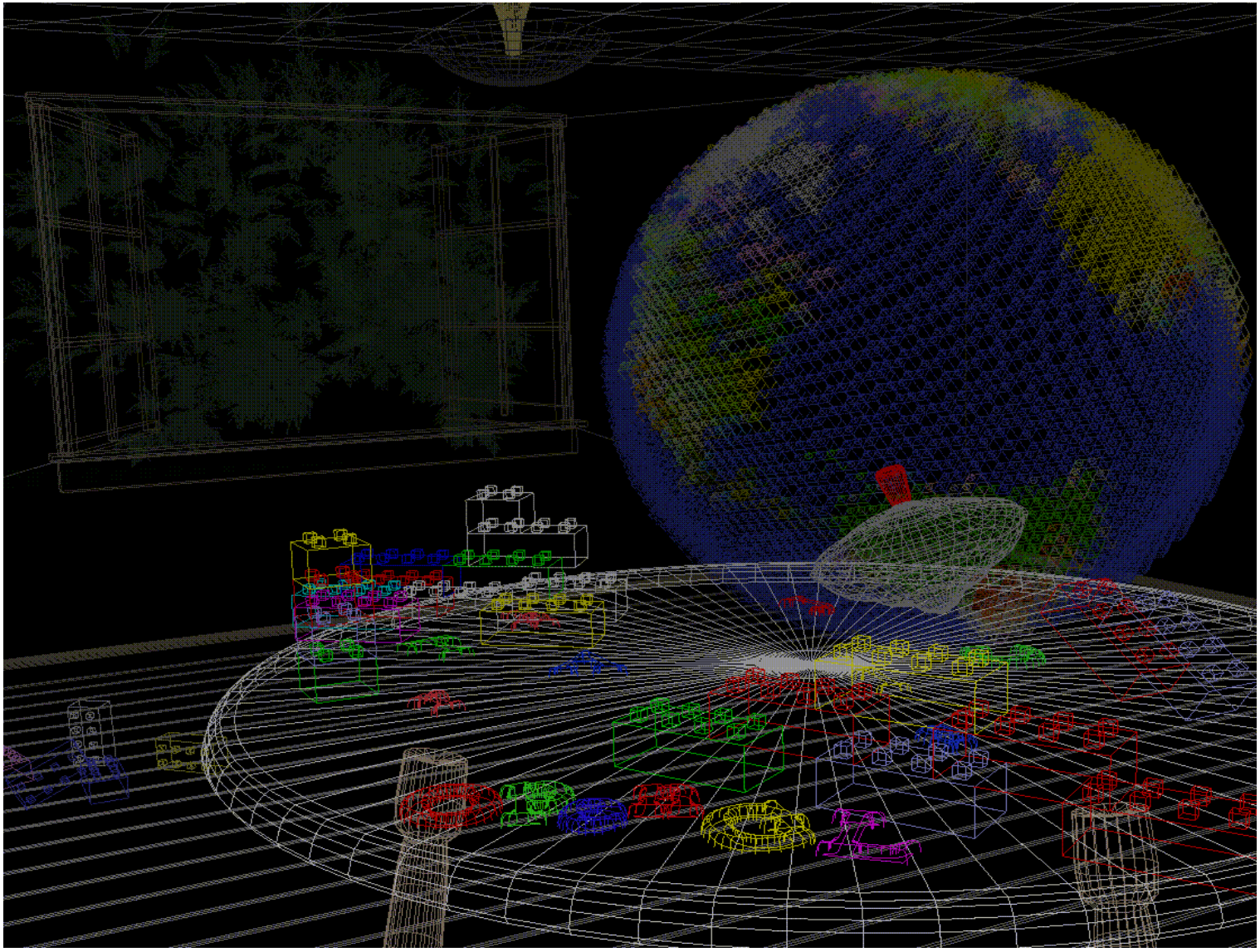
wireframe

What can OpenGL do?



Antialised lines

What can OpenGL do?

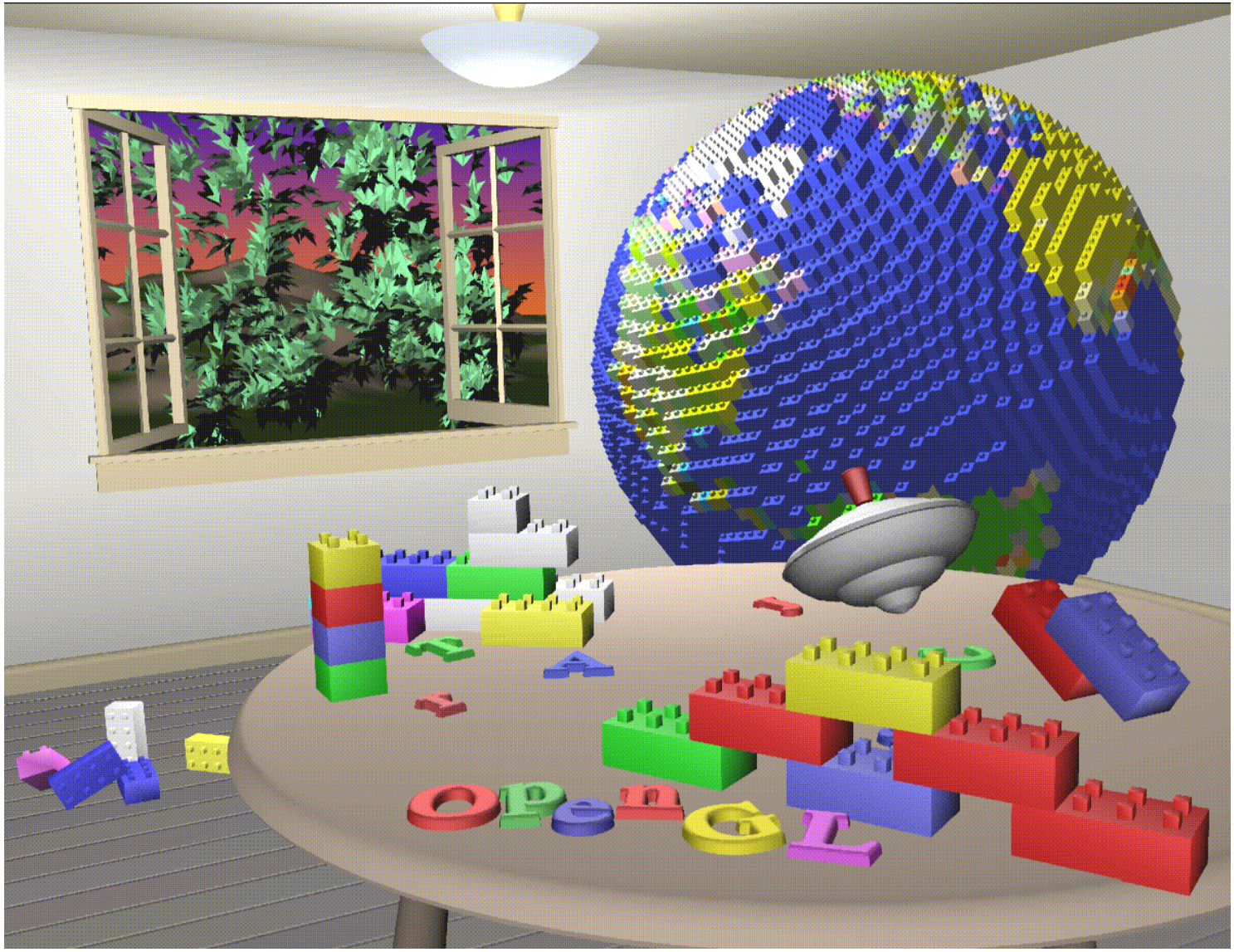


Depth cue using fog

What can OpenGL do?

Flat-shaded polygons

What can OpenGL do?



Lighting and smooth-shaded polygons

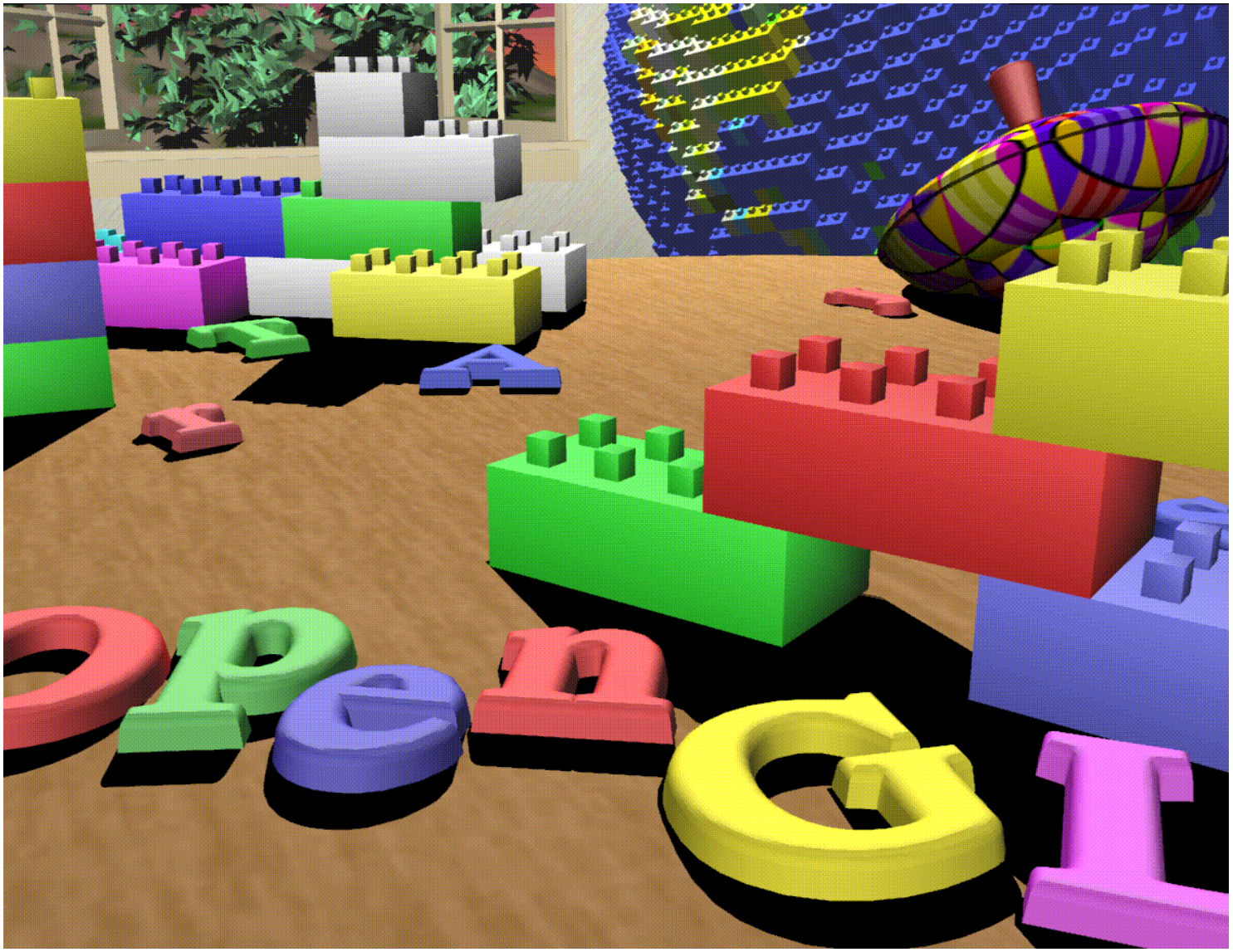
What can OpenGL do?

Texturemap and shadow

What can OpenGL do?

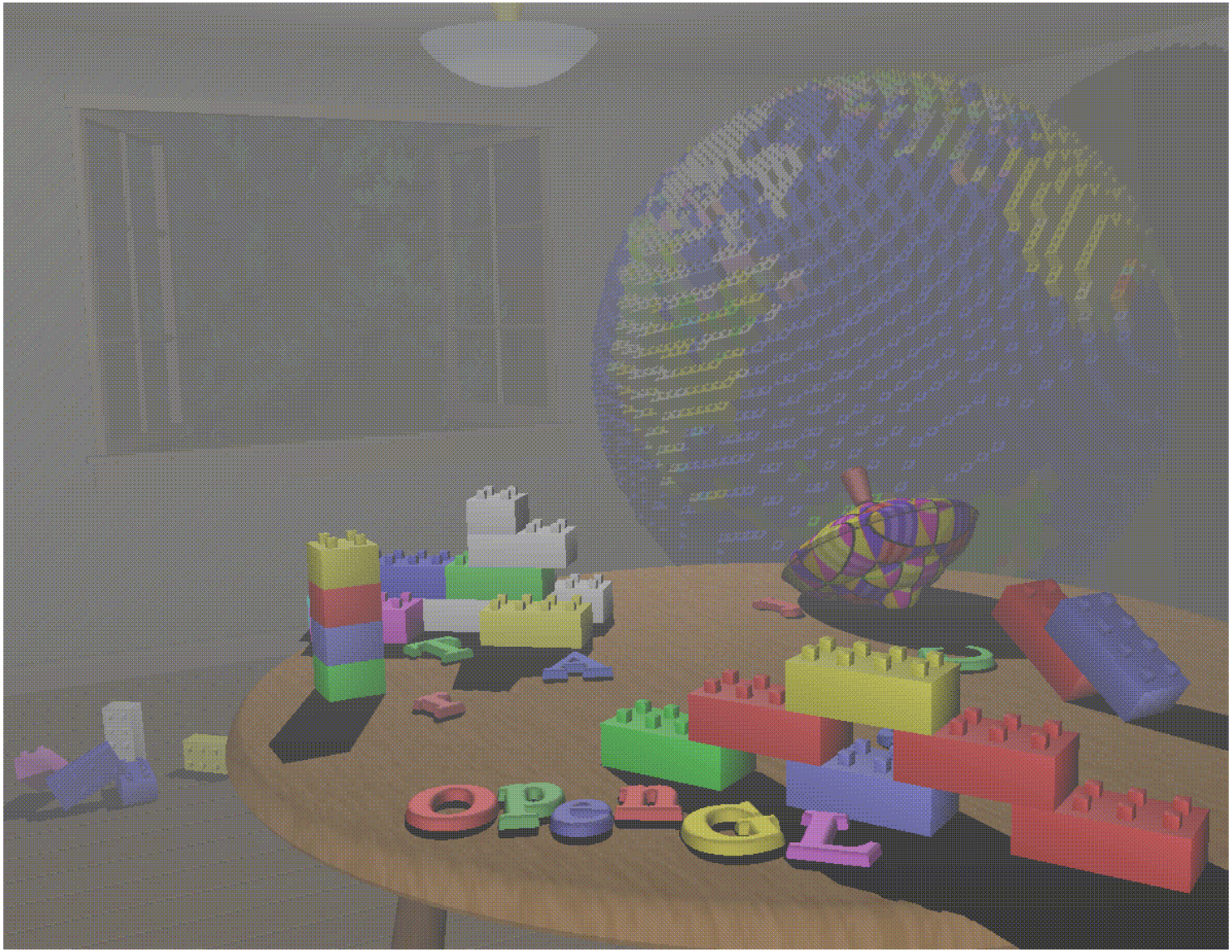
Motion blur

What can OpenGL do?



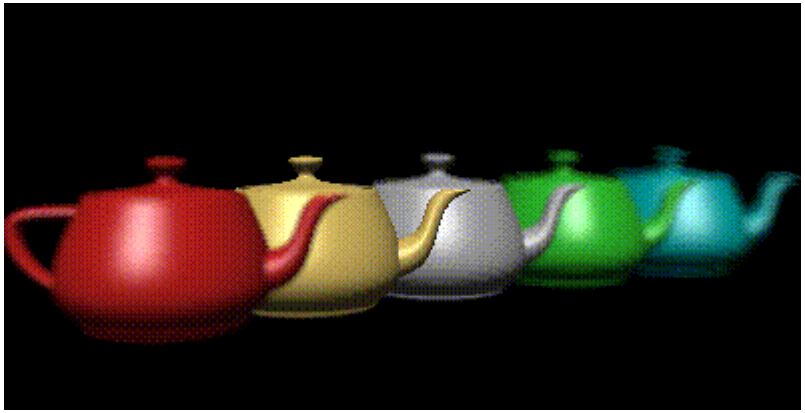
View point change

What can OpenGL do?



Smoke

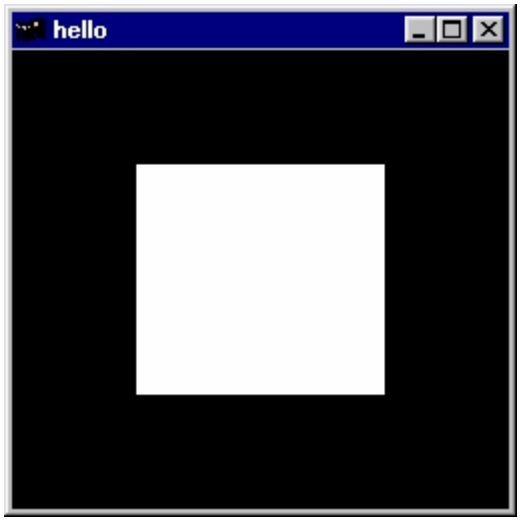
What can OpenGL do?

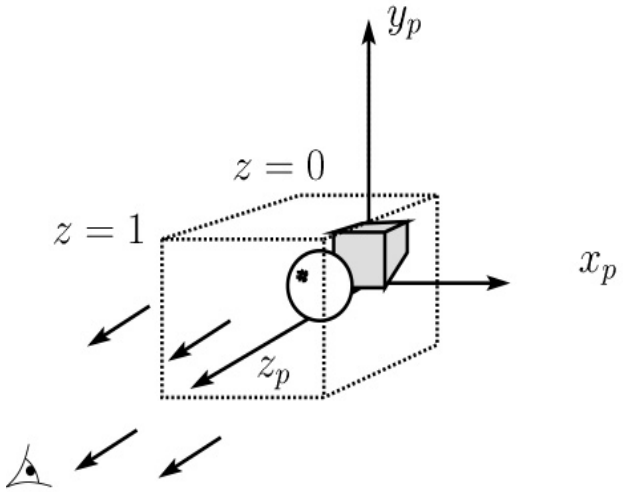


Depth of field

Hello, world

```
#include <whateverYouNeed.h>

main() {

   OpenAWindowPlease();

   glClearColor(0.0, 0.0, 0.0, 0.0);
   glClear(GL_COLOR_BUFFER_BIT);
   glColor3f(1.0, 1.0, 1.0);
   glOrtho(-1.0, 1.0, -1.0, 1.0, -1.0, 1.0);
   glBegin(GL_POLYGON);
      glVertex2f(-0.5, -0.5);
      glVertex2f(-0.5, 0.5);
      glVertex2f(0.5, 0.5);
      glVertex2f(0.5, -0.5);
   glEnd();
   glFlush();

   KeepTheWindowOnTheScreenForAWhile();
}
```





OpenGL syntax

gl prefix for all commands

GL_ for constants

```
glColor3f(1.0, 1.0, 1.0);
```

```
glColor3d(1.0, 1.0, 1.0); glColor3s(1, 1, 1); glColor3i(1, 1, 1); ……
```

Suffix	Data Type	Typical Corresponding C-Language Type	OpenGL Type Definition
b	8-bit integer	signed char	GLbyte
s	16-bit integer	short	GLshort
i	32-bit integer	long	GLint, GLsizei
f	32-bit floating-point	float	GLfloat, GLclampf
d	64-bit floating-point	double	GLdouble, GLclampd
ub	8-bit unsigned integer	unsigned char	GLubyte, GLboolean
us	16-bit unsigned integer	unsigned short	GLushort
ui	32-bit unsigned integer	unsigned long	GLuint, GLenum, GLbitfield

OpenGL syntax

```
glVertex2i(1, 1);
```

```
glVertex2f(1.0, 1.0);
```

```
glColor3f(1.0, 0.0, 0.0);
```

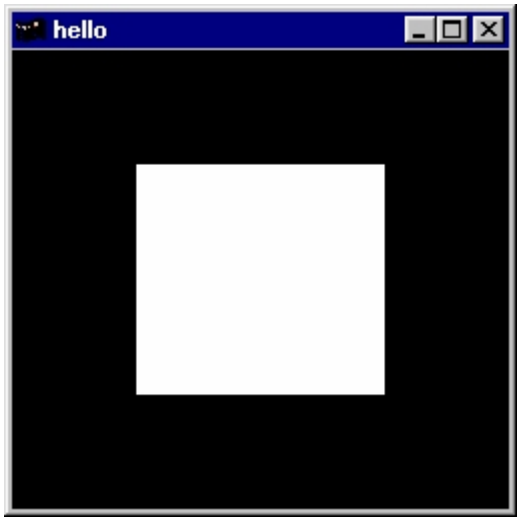
```
glColor3ub(255, 0, 0);
```

```
glColor3f(1.0, 0.0, 0.0);
```

```
float color_array[] = {1.0, 0.0, 0.0};
glColor3fv(color_array);
```

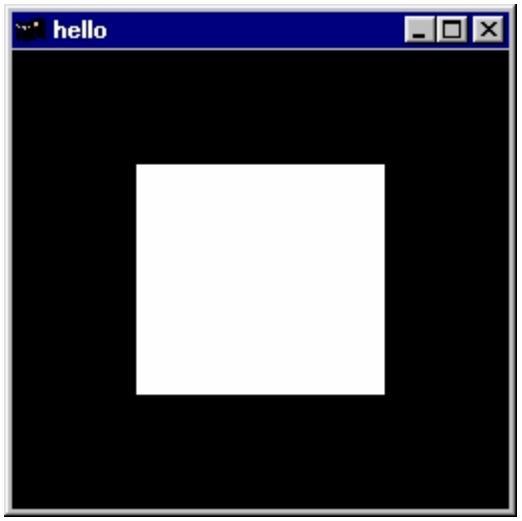
Windows management GLUT lib

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

int main(int argc, char** argv){
  glutInit(&argc, argv);
  glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
  glutInitWindowSize (250, 250);
  glutInitWindowPosition (100, 100);
  glutCreateWindow ("hello");
  init ();



}
```



Windows management GLUT lib

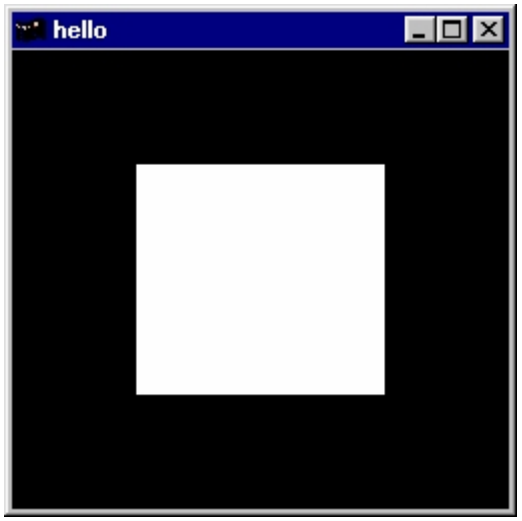
```
#include <GL/gl.h>
#include <GL/glut.h>

int main(int argc, char** argv){
  glutInit(&argc, argv);
  glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
  glutInitWindowSize (250, 250);
  glutInitWindowPosition (100, 100);
  glutCreateWindow ("hello");
  init ();
  glutDisplayFunc(display);


}
```




```
void init (void) {
  glClearColor (0.0, 0.0, 0.0, 0.0);
  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();
  glOrtho(0.0, 1.0, 0.0, 1.0, -1.0, 1.0);
}
```

Windows management GLUT lib

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

int main(int argc, char** argv){
  glutInit(&argc, argv);
  glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
  glutInitWindowSize (250, 250);
  glutInitWindowPosition (100, 100);
  glutCreateWindow ("hello");
  init ();
  glutDisplayFunc(display);
  glutMainLoop();
  return 0;
}
```



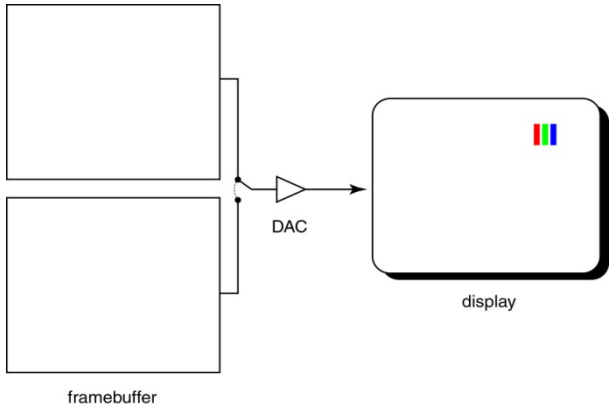

```
void display(void){
     glClear (GL_COLOR_BUFFER_BIT);
     glColor3f (1.0, 1.0, 1.0);
     glBegin(GL_POLYGON);
     glVertex2f(-0.5, -0.5);
     glVertex2f(-0.5, 0.5);
     glVertex2f(0.5, 0.5);
     glVertex2f(0.5, -0.5);
     glEnd();
     glFlush ();
}
```

Animation

```
open_window();
for (i = 0; i < 1000000; i++) {
    clear_the_window();
    draw_frame(i);
    wait_until_a_24th_of_a_second_is_over();
}
```

Q: What happens when you write to the framebuffer while it is being displayed on the monitor?

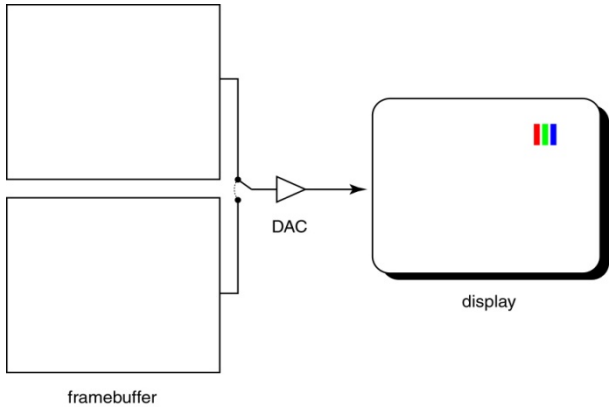
Animation




```
open_window();
for (i = 0; i < 1000000; i++) {
    clear_the_window();
    draw_frame(i);
    wait_until_a_24th_of_a_second_is_over();
}
```

Q: What happens when you write to the framebuffer while it is being displayed on the monitor?

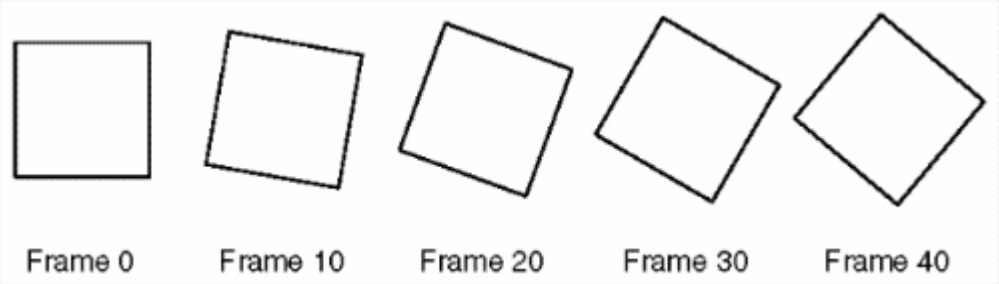
Animation




```
open_window();
for (i = 0; i < 1000000; i++) {
    clear_the_window();
    draw_frame(i);
    wait_until_a_24th_of_a_second_is_over();
    swap_the_buffers();
}
```

Q: What happens when you write to the framebuffer while it is being displayed on the monitor?

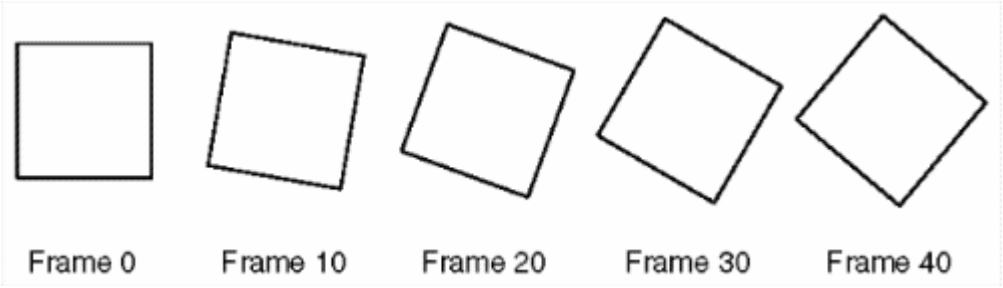
Animation Example




```
int main(int argc, char** argv){
     glutInit(&argc, argv);
     glutInitDisplayMode (GLUT_DOUBLE | GLUT_RGB);
     glutInitWindowSize (250, 250);
     glutInitWindowPosition (100, 100);
     glutCreateWindow (argv[0]);
     init ();
     glutDisplayFunc(display);



}
```

```
void init(void) {
     glClearColor (0.0, 0.0, 0.0, 0.0);
     glShadeModel (GL_FLAT);
}
```

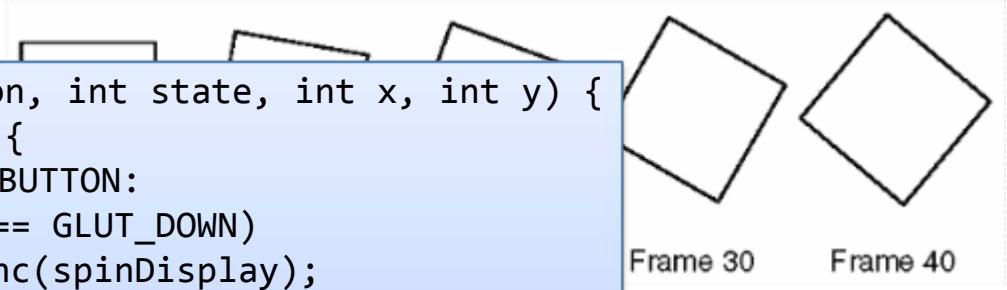
Animation Example




```
static GLfloat spin = 0.0;

void display(void){
     glClear(GL_COLOR_BUFFER_BIT);
     glPushMatrix();
     glRotatef(spin, 0.0, 0.0, 1.0);
     glColor3f(1.0, 1.0, 1.0);
     glRectf(-25.0, -25.0, 25.0, 25.0);
     glPopMatrix();
     glutSwapBuffers();
}
```

```
int main(int argc, char** argv){
     glutInit(&argc, argv);
     glutInitDisplayMode (GLUT_DOUBLE | GLUT_RGB);
     glutInitWindowSize (250, 250);
     glutInitWindowPosition (100, 100);
     glutCreateWindow (argv[0]);
     init ();
     glutDisplayFunc(display);
     glutMouseFunc(mouse);
```

Animation Example

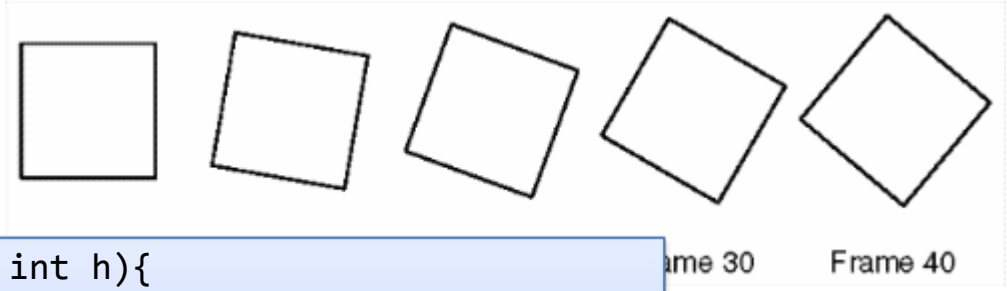



```
void mouse(int button, int state, int x, int y) {
    switch (button) {
     case GLUT_LEFT_BUTTON:
          if (state == GLUT_DOWN)
          glutIdleFunc(spinDisplay);
          break;
     case GLUT_MIDDLE_BUTTON:
          if (state == GLUT_DOWN)
          glutIdleFunc(NULL);
          break;
     default:
          break;
     }
}
```

```
rgc, char** argv){
&argc, argv);
isplayMode (GLUT_DOUBLE | GLUT_RGB);
indowSize (250, 250);
indowPosition (100, 100);
eWindow (argv[0]);

ayFunc(display);
glutMouseFunc(mouse);
glutReshapeFunc(reshape);
```

```
spinDisplay(void){
     spin = spin + 2.0;
     if (spin > 360.0) spin -= 360.0;
     glutPostRedisplay();
}
```

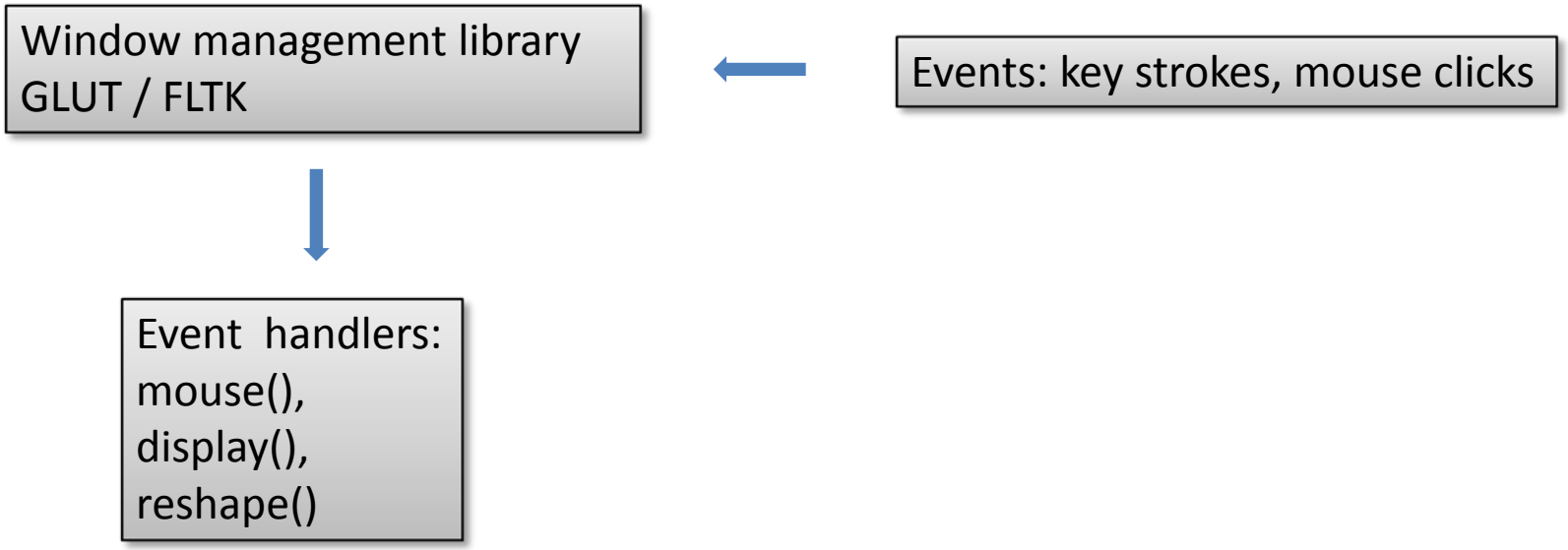
Animation Example




```
void reshape(int w, int h){
     glViewport (0, 0, (GLsizei) w, (GLsizei) h);
     glMatrixMode(GL_PROJECTION);
     glLoadIdentity();
     glOrtho(-50.0, 50.0, -50.0, 50.0, -1.0, 1.0);
     glMatrixMode(GL_MODELVIEW);
     glLoadIdentity();
}
```

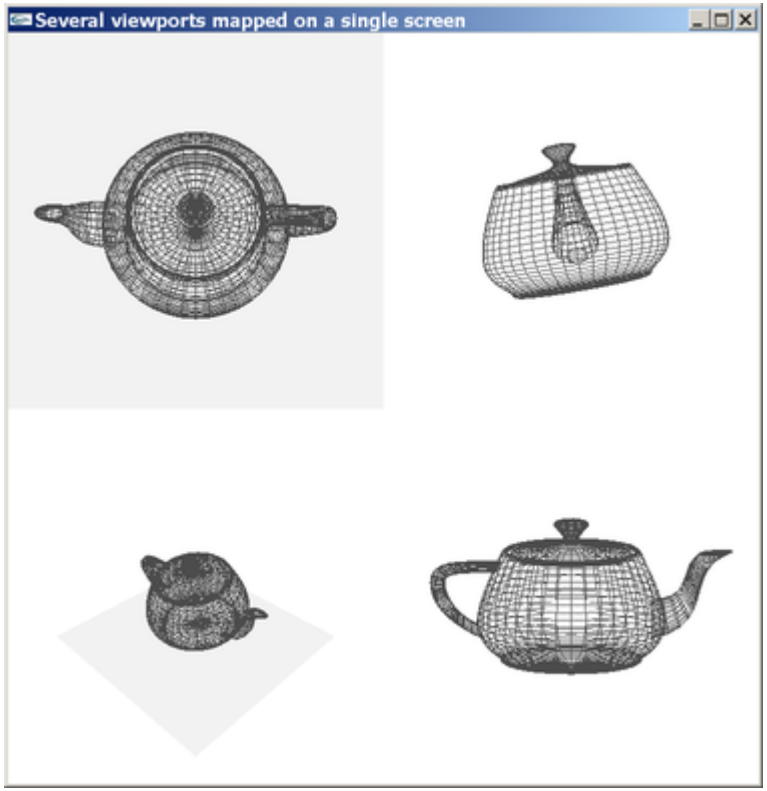
```
gc, char** argv){
rgc, argv);
playMode (GLUT_DOUBLE | GLUT_RGB);
dowSize (250, 250);
dowPosition (100, 100);
indow (argv[0]);
     init ();
     glutDisplayFunc(display);
     glutMouseFunc(mouse);
     glutReshapeFunc(reshape);
     glutMainLoop();    return 0;
}
```

Event-Driven Programming



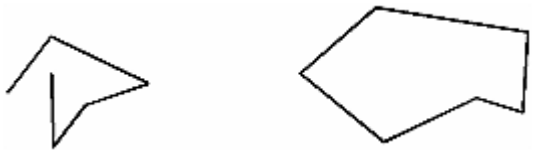
Viewport

```
void reshape (int w, int h) {
    glViewport (0, 0, w, h);
    glMatrixMode (GL_PROJECTION);
    glLoadIdentity ();
    gluOrtho2D (0.0, w, 0.0, h);
}
```

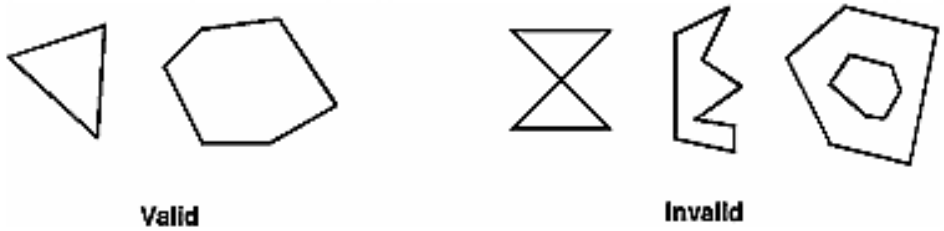


Points, lines, and polygons

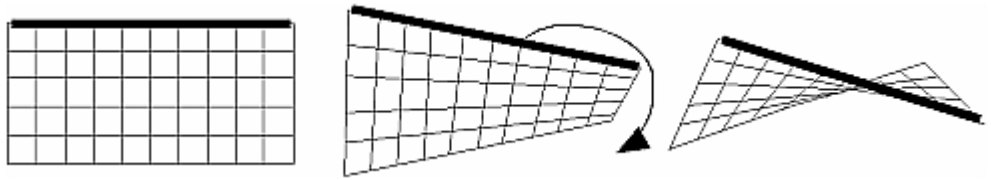
- Points
- Lines



- Polygons



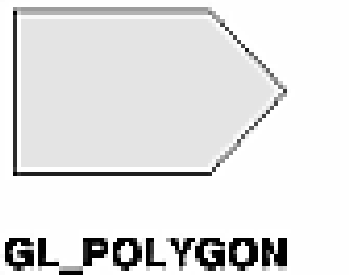
Simple Polygonss: Convex & planar



Nonplanar Polygon Transformed to Nonsimple Polygon

Drawing Primitives

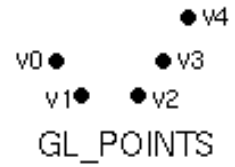
```
glBegin(GL_POLYGON);
glVertex2f(0.0, 0.0);
glVertex2f(0.0, 3.0);
glVertex2f(4.0, 3.0);
glVertex2f(6.0, 1.5);
glVertex2f(4.0, 0.0);
glEnd();
```



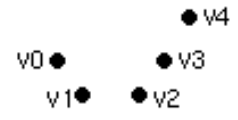
Drawing Primitives

```
glBegin(GL_POINTS);
glVertex2f(0.0, 0.0);
glVertex2f(0.0, 3.0);
glVertex2f(4.0, 3.0);
glVertex2f(6.0, 1.5);
glVertex2f(4.0, 0.0);
glEnd();
```

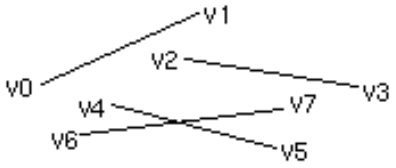
Drawing Primitives



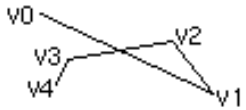
Drawing Primitives



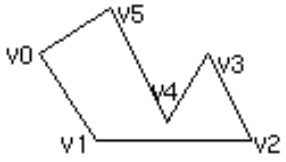
GL_POINTS



GL_LINES

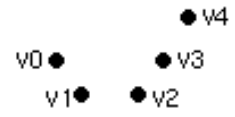


GL_LINE_STRIP

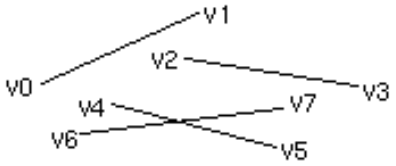


GL_LINE_LOOP

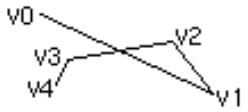
Drawing Primitives



GL_POINTS

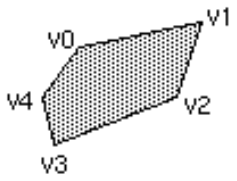


GL_LINES

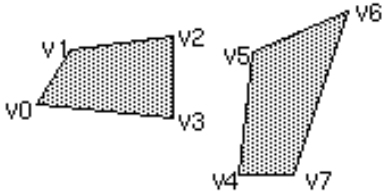


GL_LINE_STRIP

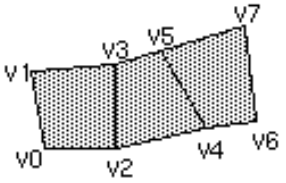
GL_LINE_LOOP



GL_POLYGON

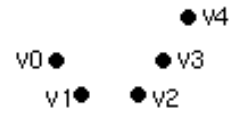


GL_QUADS

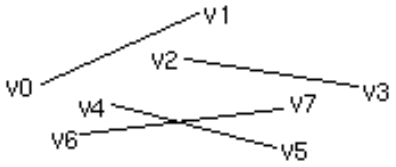


GL_QUAD_STRIP

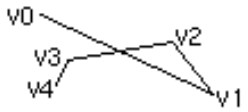
Drawing Primitives



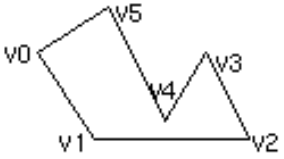
GL_POINTS



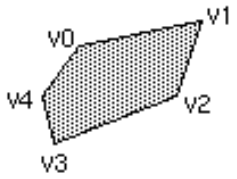
GL_LINES



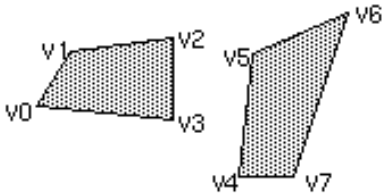
GL_LINE_STRIP



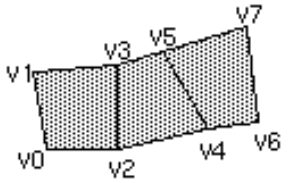
GL_LINE_LOOP



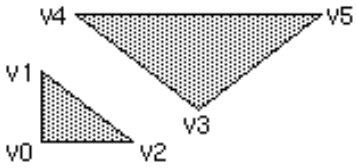
GL_POLYGON



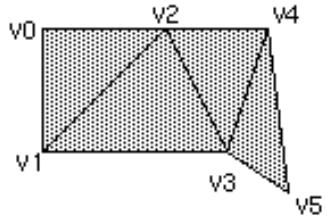
GL_QUADS



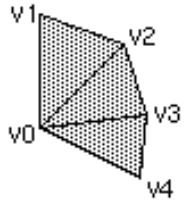
GL_QUAD_STRIP



GL_TRIANGLES



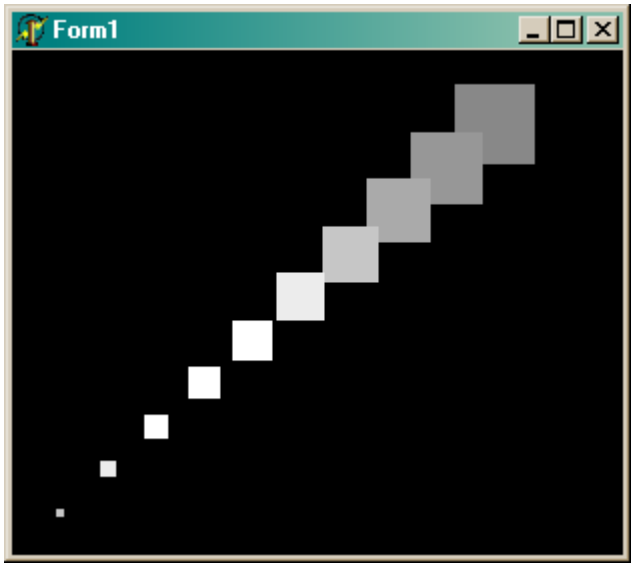
GL_TRIANGLE_STRIP

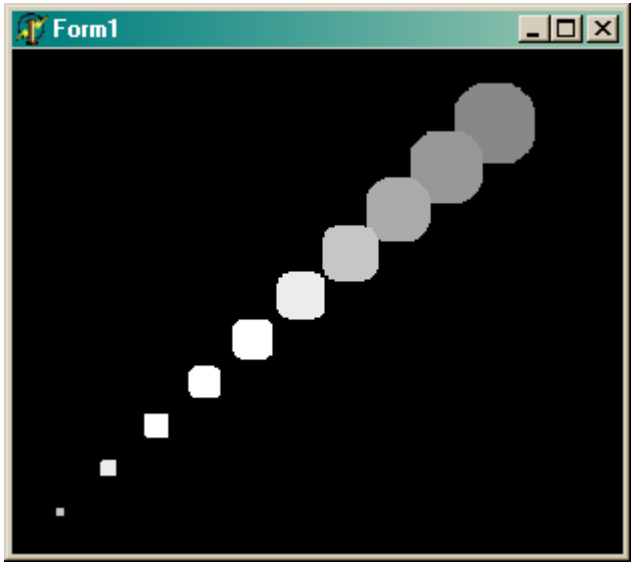


GL_TRIANGLE_FAN

Primitive Details

- glPointSize(GLfloat size)
  - Approximate the point by squares for anti-aliasing






```
glEnable(GL_POINT_SMOOTH);
```

Primitive Details

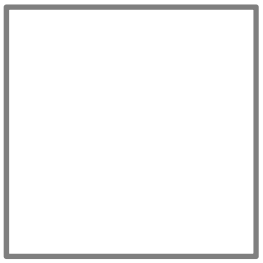
- glLineWidth(GLfloat width)
  - Approximate the line by a rectangle for anti-aliasing

```
glEnable (GL_LINE_SMOOTH);
glLineWidth (1.5);
```

Primitive Details

- glPolygonMode(*GLenum face, GLenum mode);*
  - face: GL_FRONT, GL_BACK
  - mode: GL_POINT, GL_LINE, GL_FILL

```
glPolygonMode(GL_FRONT, GL_FILL);
glRectf(0, 0, 100, 100);
```



```
glPolygonMode(GL_FRONT, GL_LINE);
glRectf(0, 0, 100, 100);
```