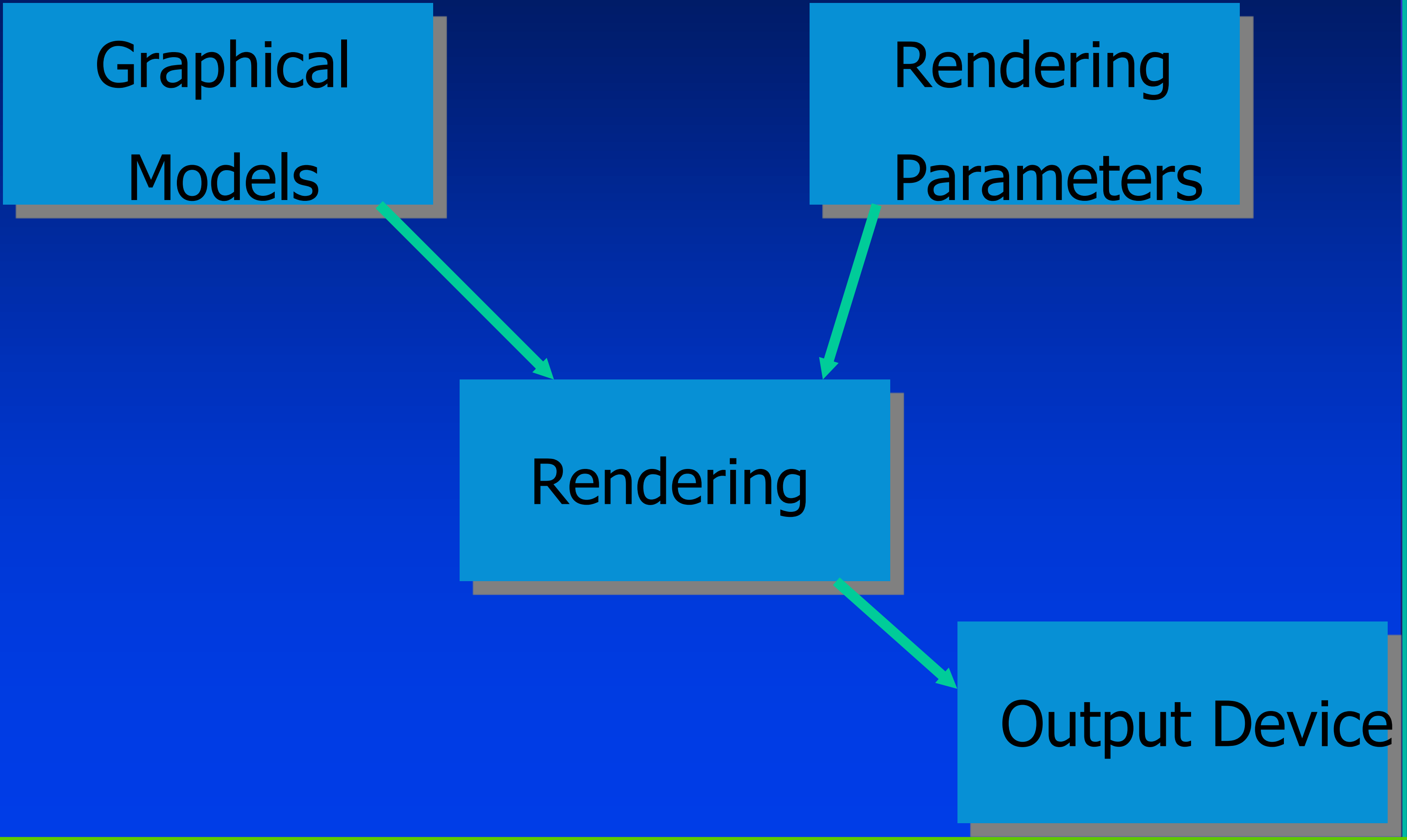
Computer Graphics

- (Realistic) pictorial synthesis of real and/or imaginary objects from their computer-based models (datasets)
- It typically includes modeling, rendering (graphics pipeline), and human-computer interaction
- So, we are focusing on computer graphics hardware, software, and mathematical foundations
- Computer Graphics is computation
  - A new method of visual computing
- Why is Computer Graphics useful and important?
- Course challenges: more mathematics oriented, programming requirements, application-driven, inter-disciplinary in nature, etc.

Computer Graphics Systems



Output Devices

- Vector Devices
  - Lasers (for example)
- Raster Devices
  - CRT, LCD, bitmaps, etc.
  - Most output devices are 2D
  - Can you name any 3D output devices?

Graphical Models

- 2D and 3D objects
  - Triangles, quadrilaterals, polygons
  - Spheres, cones, boxes
- Surface characteristics
  - Color, reaction to light
  - Texture, material properties
- Composite objects
  - Other objects and their relationships to each other
- Lighting, fog, etc.
- Much, much more…

Rendering

- Conversion of 3D model to 2D image
  - Determine where the surfaces “project” to
  - Determine what every screen pixel might see
  - Determine the color of each surface

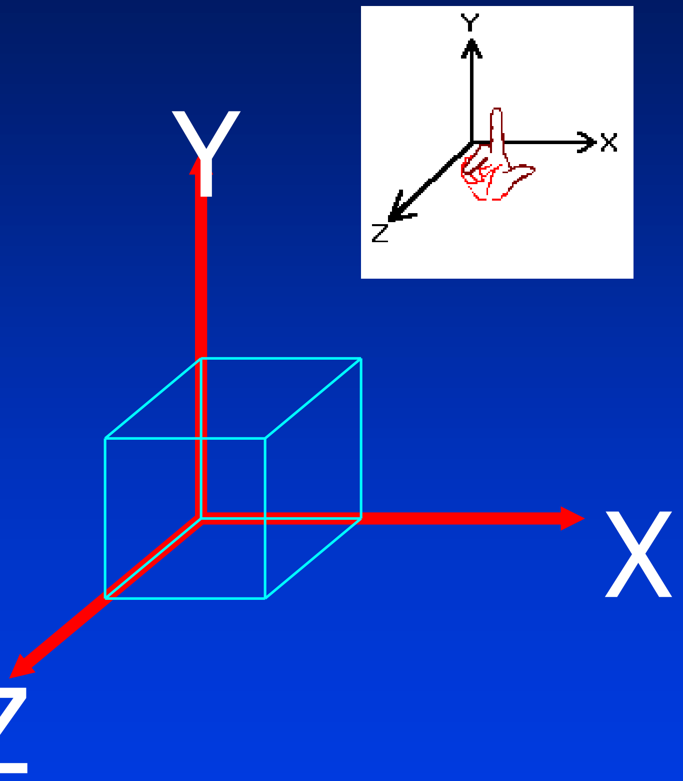
Rendering Parameters

- Camera parameters
  - Location
  - Orientation
  - Focal length

2D Graphics vs. 3D Graphics

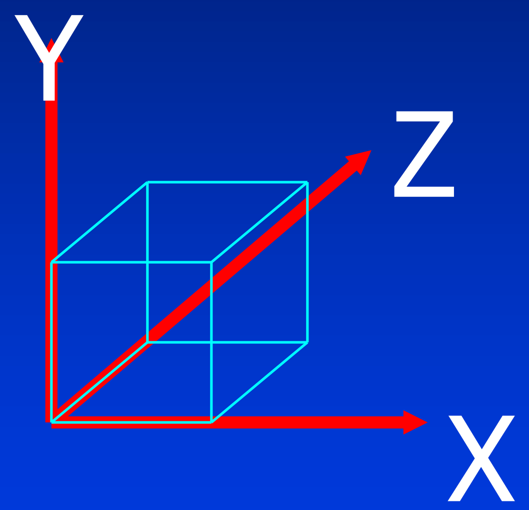
- 2D
  - X, Y - 2 dimensions only
  - We won’t spend time on 2D graphics in this course
- 3D
  - X, Y, and Z
  - Space
- Rendering is typically the conversion of 3D to 2D

3D Coordinate Systems



Right-Hand Coordinate System

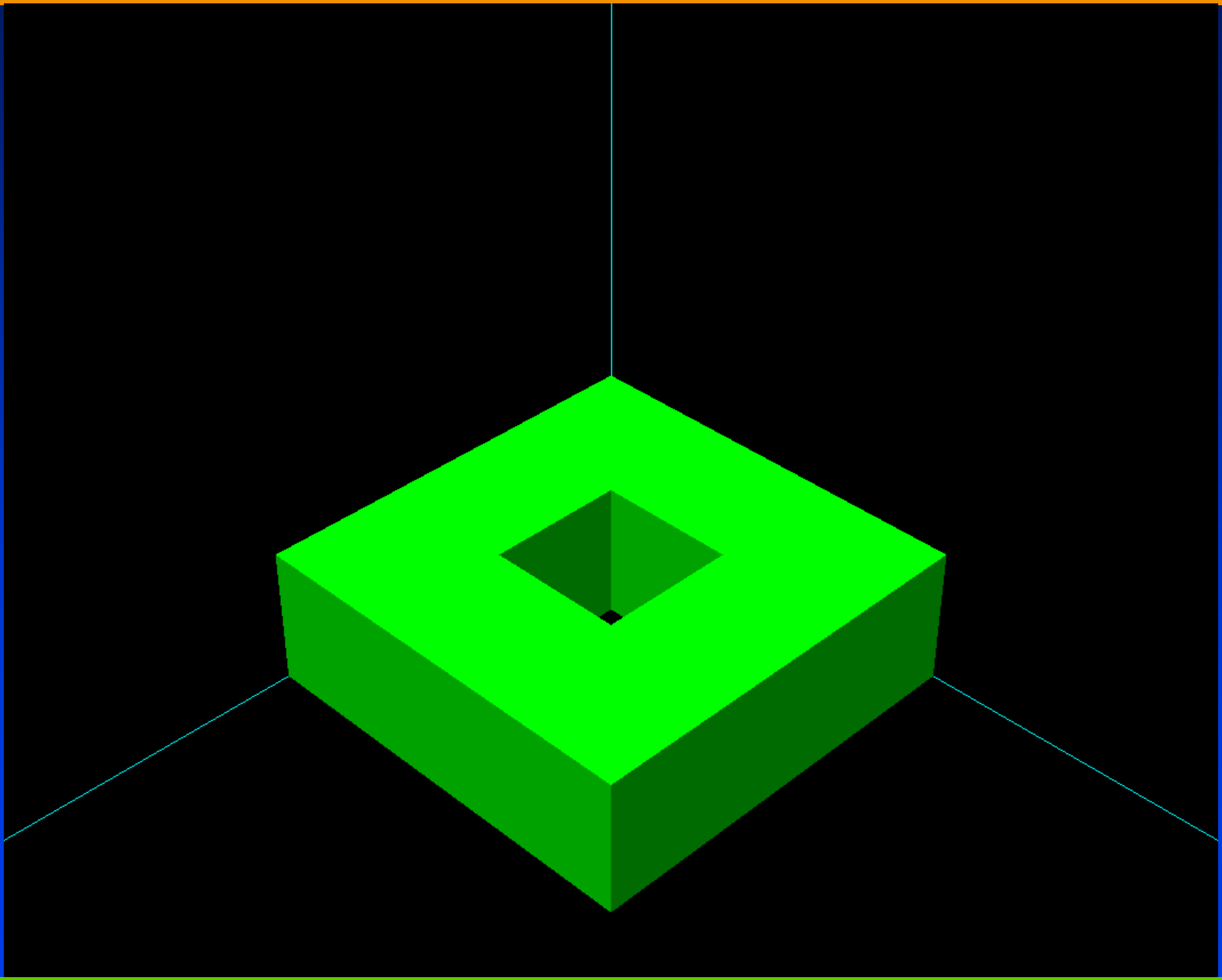
OpenGL uses this!



Left-Hand Coordinate System

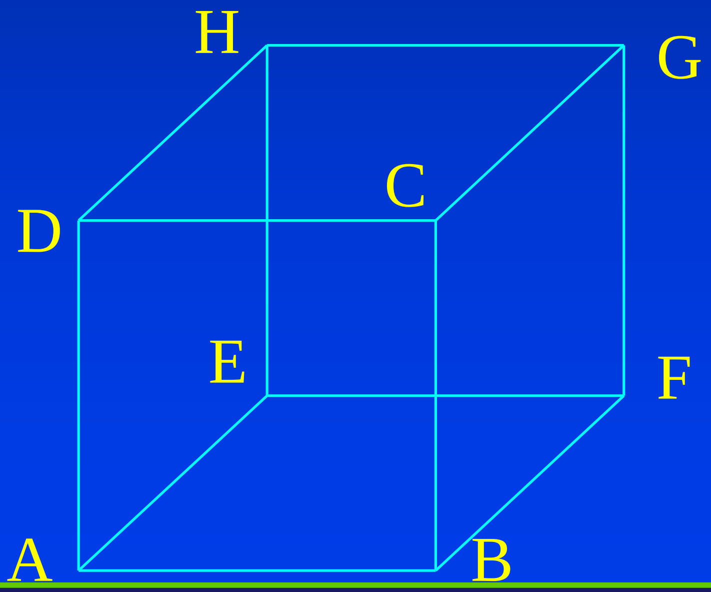
Direct3D uses this!

How to Model/Render This?



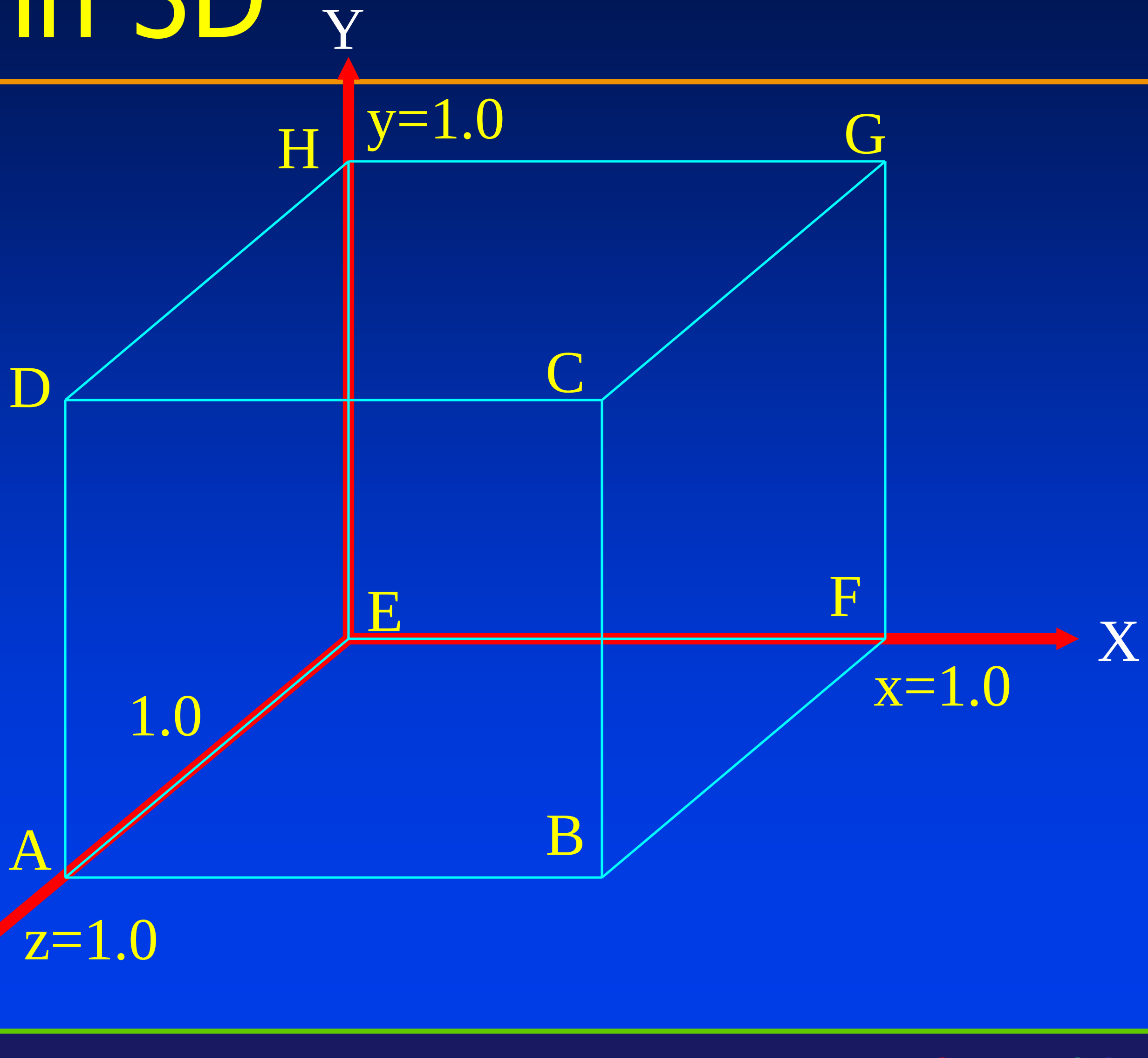
Render/Display a Box in OpenGL

- We render the 6 faces as polygons
  - Polygons are specified as a list of vertices
  - Vertices are specified in counter-clockwise order looking at the surface of the face!



Visualizing in 3D

Counter-clockwise



OpenGL Conventions

- C library
  - – All function names start with gl
- OpenGL is a retained mode graphics system
  - – It has a state
  - – glBegin(GL_POLYGON) puts us into a polygon rendering state

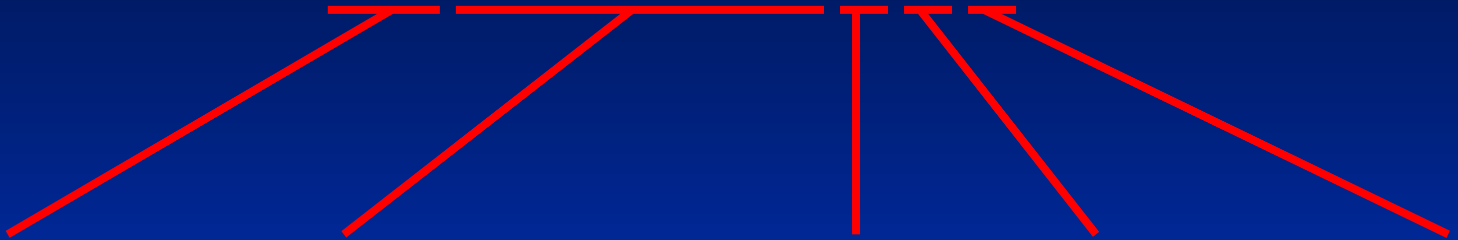
OpenGL Polygon Rendering

```
GLdouble size = 1.0;
glBegin(GL_POLYGON);      // front face
  glVertex3d(0.0,  0.0,  size);
  glVertex3d(size, 0.0,  size);
  glVertex3d(size, size, size);
  glVertex3d(0.0,  size, size);
glEnd();
```

OpenGL Types

- Basic numeric types
  - GLdouble = double
  - GLfloat = float
  - GLint = int
  - GLshort = short
- Mostly, you’ll use GLdouble and GLfloat

Defined glVertex3fv



Prefix	Function	# Parms	Type	Suffix
gl	Vertex	1	f (float)	v (vector)
glu	Begin	2	d (double)	
wgl	End	3	i (integer)	
agl	Lighting	4	b (byte)	
	…	…	s (short)	

Only if varying arguments

Function Suffixes

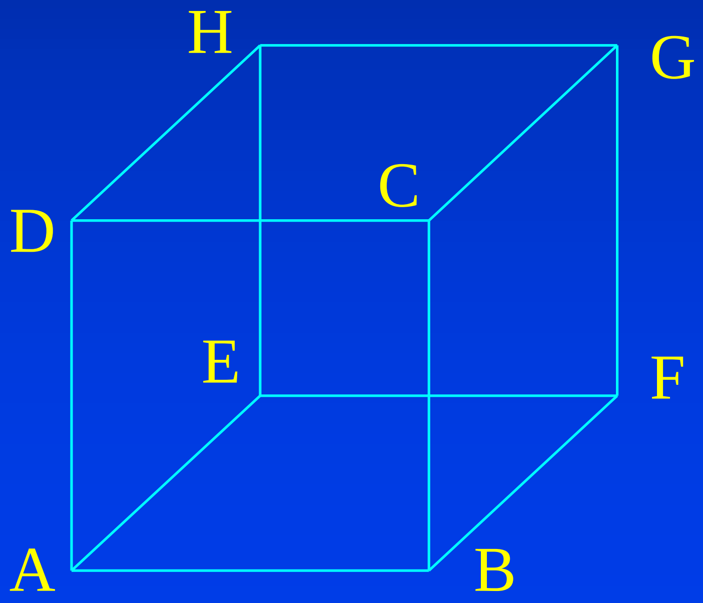
- Many functions have alternatives
  - Alternatives are specified by the suffix
  - **glVertex2d**
    - **2 double parameters**
    - **void glVertex2d(GLdouble x, GLdouble y);**
  - **glVertex3f**
    - **3 float parameters**
    - **void glVertex3f(GLfloat x, GLfloat y, GLfloat z);**
  - **glVertex3fv**
    - **void glVertex3fv(const GLfloat *v);**

All of Them…

- **glVertex2d, glVertex2f, glVertex2i, glVertex2s, glVertex3d, glVertex3f, glVertex3i, glVertex3s, glVertex4d, glVertex4f, glVertex4i, glVertex4s, glVertex2dv, glVertex2fv, glVertex2iv, glVertex2sv, glVertex3dv, glVertex3fv, glVertex3iv, glVertex3sv, glVertex4dv, glVertex4fv, glVertex4iv, glVertex4sv**

Vector Parameters

```
GLdouble a[ ] = {0, 0, 1};

GLdouble b[ ] = {1, 0, 1};

GLdouble c[ ] = {1, 1, 1};
GLdouble d[ ] = {0, 1, 1};


glBegin(GL_POLYGON);              // front face
   glVertex3dv(a);
   glVertex3dv(b);
   glVertex3dv(c);
   glVertex3dv(d);
glEnd();
```



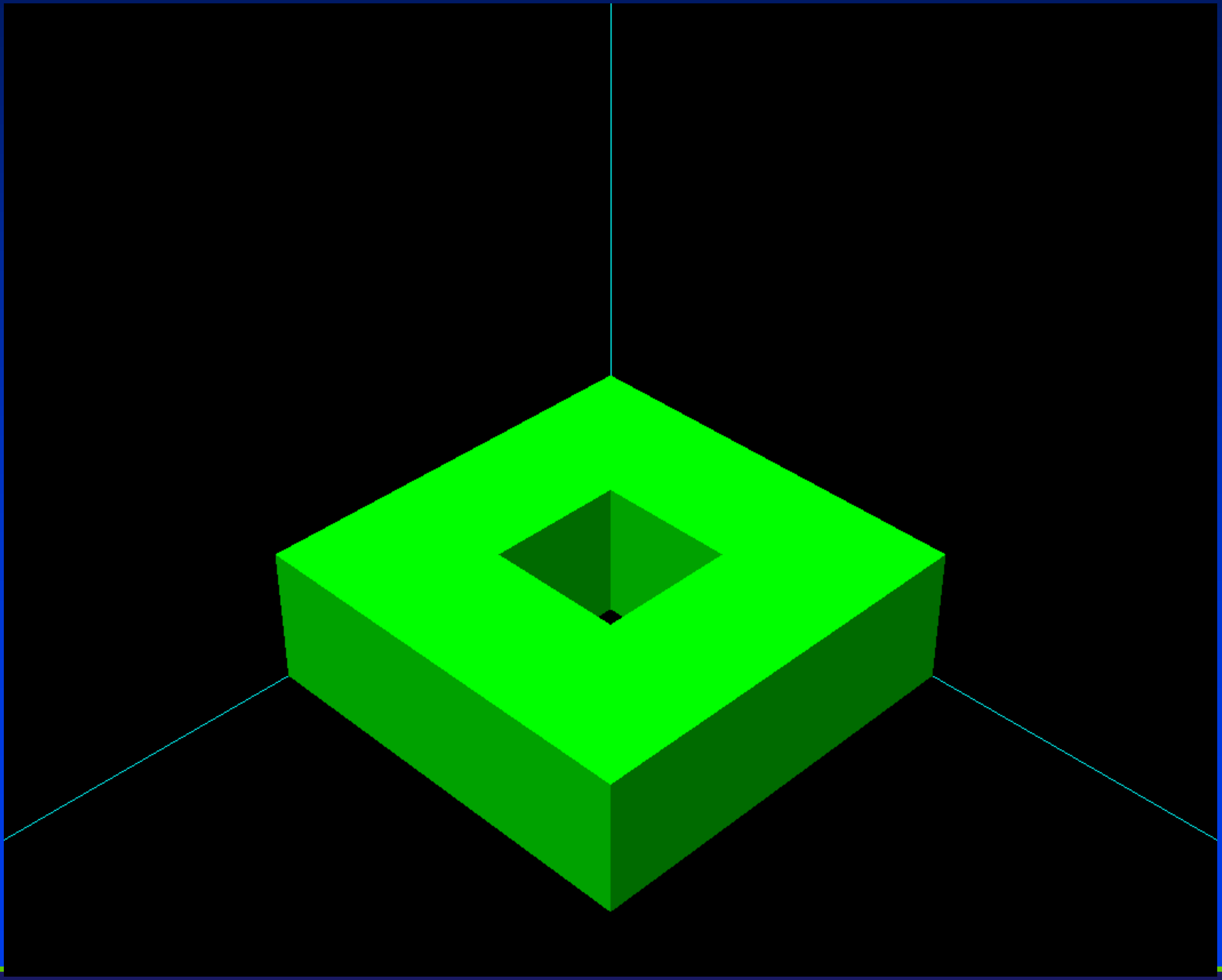
Specify a Color (No Lighting)

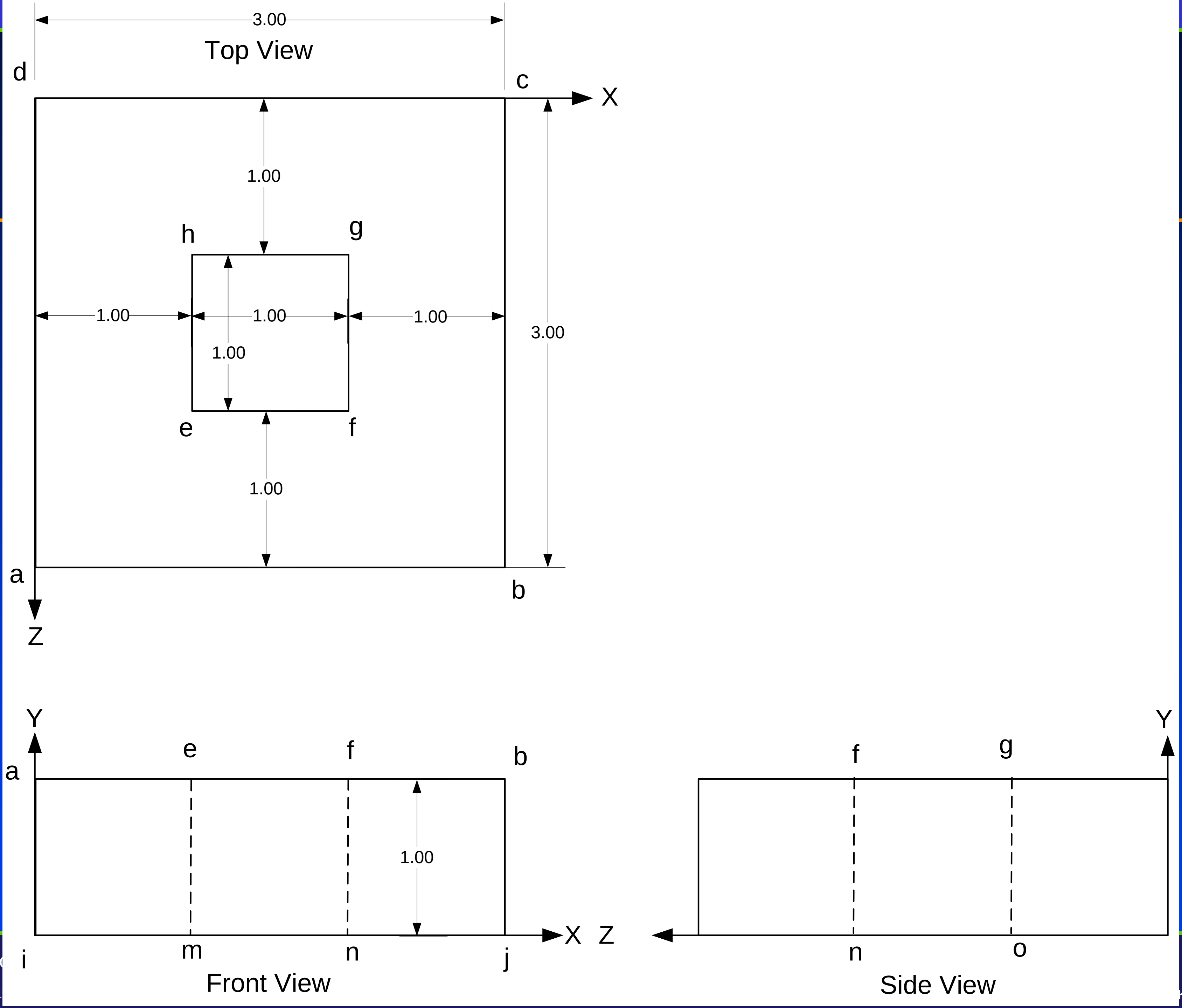
- glColor3f(red, green, blue);
- Most of the same suffixes apply…

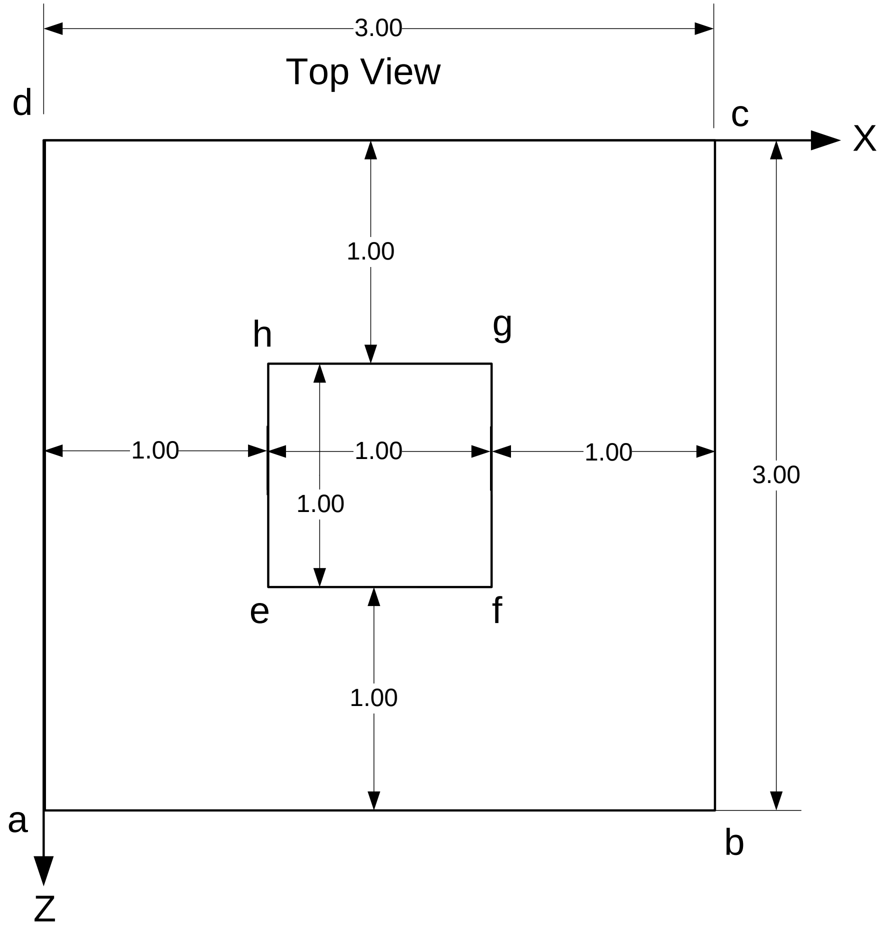
```
GLdouble size = 1.0;

glColor3d(1.0, 0.0, 0.0);                // red
glBegin(GL_POLYGON);  // front face
    glVertex3d(0.0,   0.0,   size);
    glVertex3d(size,  0.0,   size);
    glVertex3d(size,  size, size);
    glVertex3d(size,  0.0,   size);
 glEnd();
```

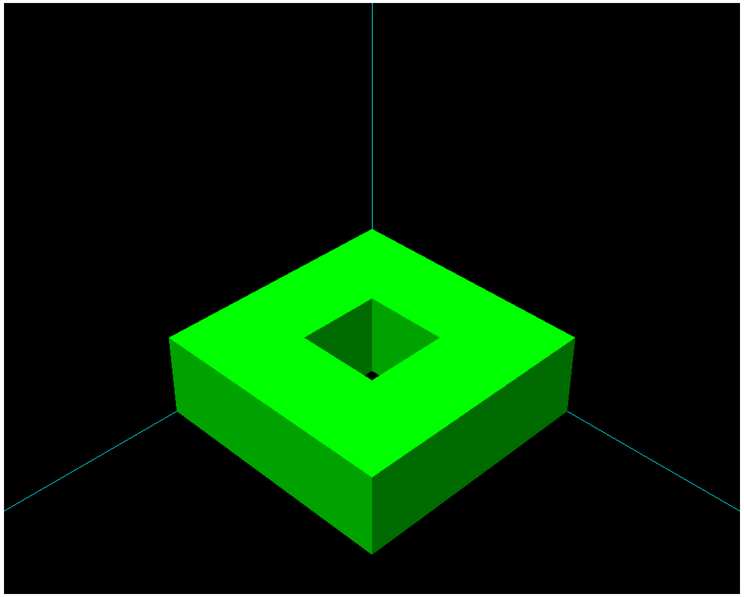
Colors range from 0 to 1

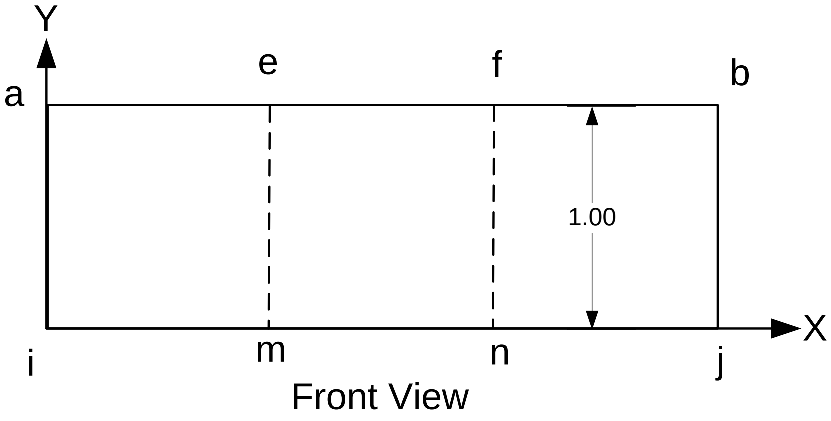
How to Model/Render This?

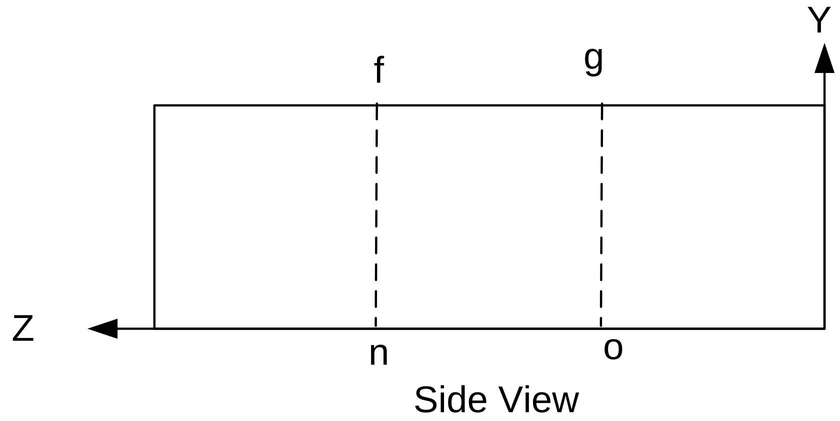



3.00
Top View
d
c
X
1.00
h
g
1.00
1.00
1.00
1.00
3.00
e
f
1.00
a
b
Z
Y
a
e
f
b
1.00
X
i
m
n
j
Front View
Y
f
g
Z
n
o
Side View

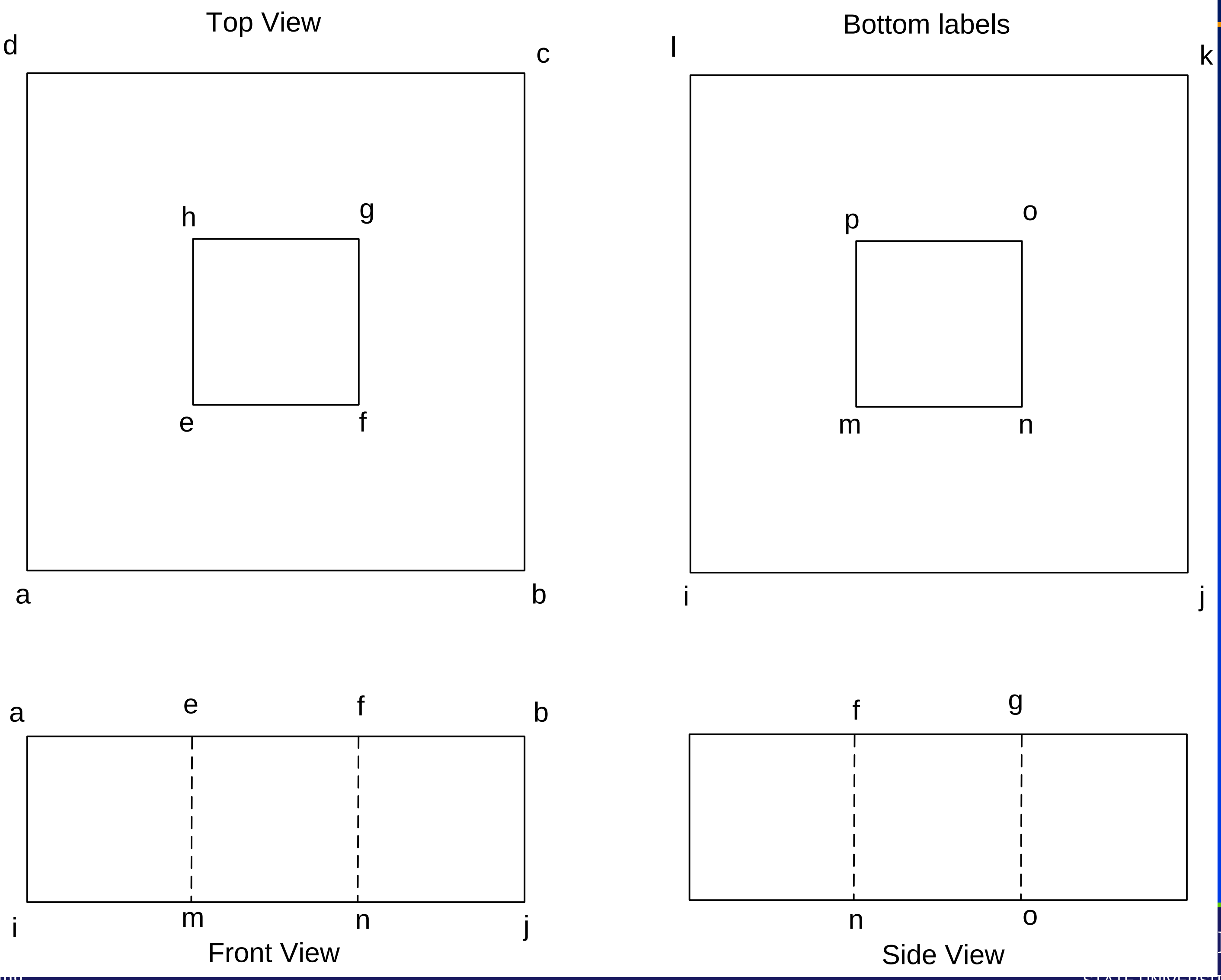

3.00
Top View
d
c
X
1.00
h
g
1.00
1.00
1.00
1.00
3.00
e
f
1.00
a
b
Z




Y
a
e
f
b
1.00
X
i
m
n
j
Front View


Y
f
g
Z
n
o
Side View

Labels


Top View
d
c
h
g
e
f
a
b
Bottom labels
l
k
p
o
m
n
i
j
a
e
f
b
i
m
n
j
Front View
f
g
n
o
Side View

The Basic Idea

- Describe an object using surfaces
- Surfaces are polygons
  - Triangles, quadrilaterals, whatever
  - Important thing is that they are flat
  - They must also be convex
- Provide points in counter-clockwise order
  - From the visible side