

---

CSE 472  
OpenGL Basics

---

CSE 472 Spring 2009

# About OpenGL

---

Do not:

- Deify OpenGL
- Think you don't need to know underlying fundamentals
- Think we will only do OpenGL
- Remember: 10 years from now OpenGL may be history...

# Reading and Lab

---

I suggest:

- Chapters 1 and 2

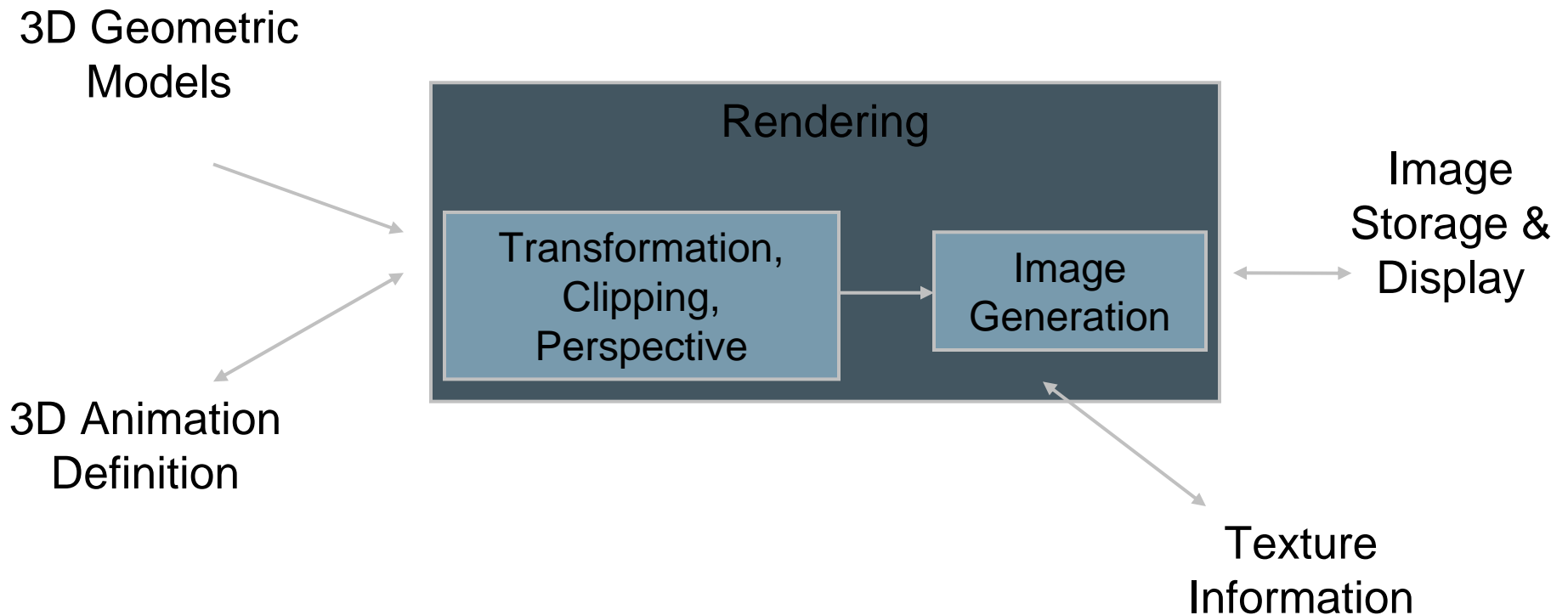
## Step 1

- Introduction to Visual Studio/OpenGL
- Using OpenGLWnd superclass
- Basic OpenGL
- Due next Wednesday (get going)

# Rendering

---

Turning everything into pictures



# Output Devices

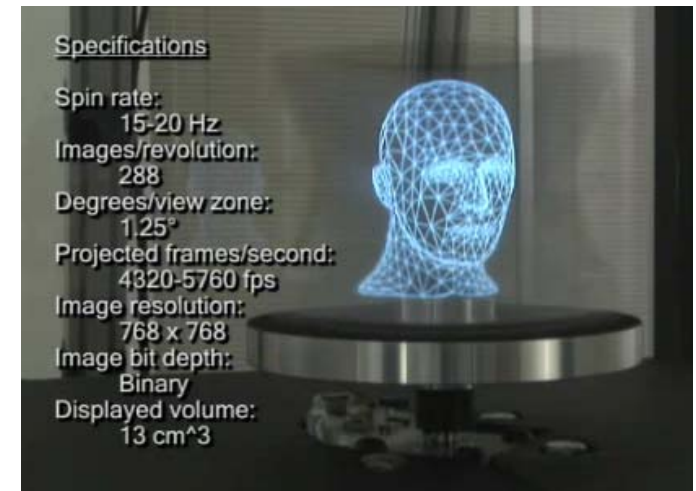
---

## Vector Devices

- Lasers for example

## Raster Devices

- CRT, LCD, bitmaps, etc.



[USC ICT]

- Most output devices are 2D
- Can you name any 3D output devices?

# Graphical Models

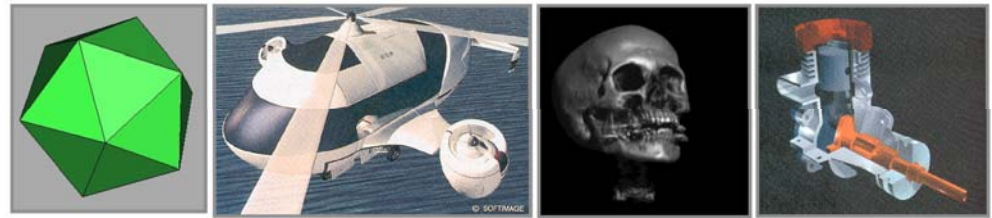
---

## 2D and 3D objects

- Triangles, quadrilaterals, polygons
- Spheres, cones, boxes

## Surface characteristics

- Color
- Texture



## Composite objects

- Other objects and their relationships to each other

Lighting, fog, etc.

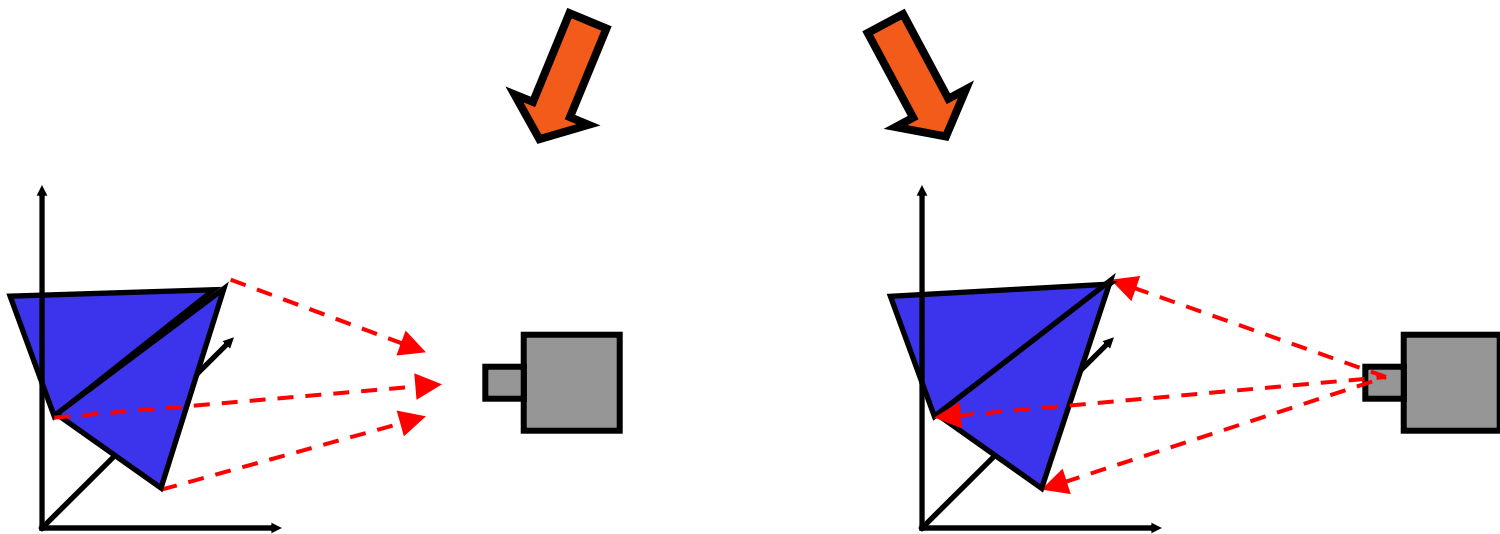
Much, much more...

---

# Basic Rendering Model

---

Models for objects and cameras?



## Rasterization:

Project geometry forward

## Ray Tracing:

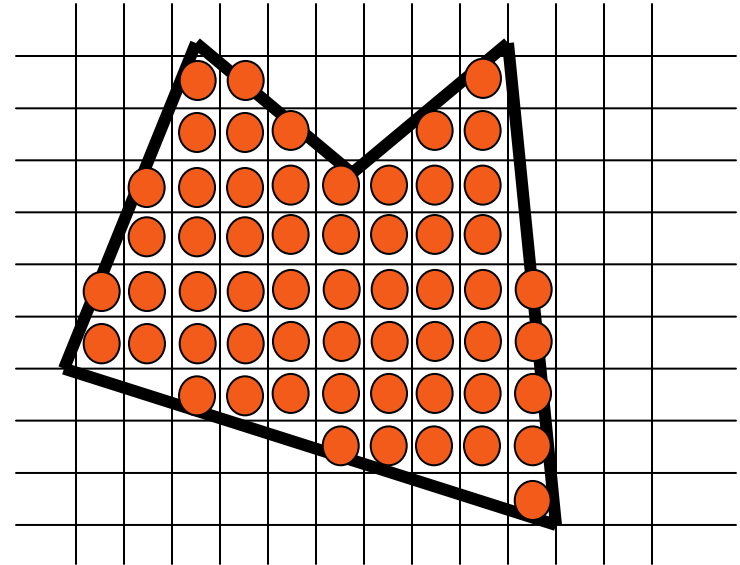
Project image samples backwards

# Rendering

---

## Conversion of 3D model to 2D image

- project
- determine pixel
- determine color

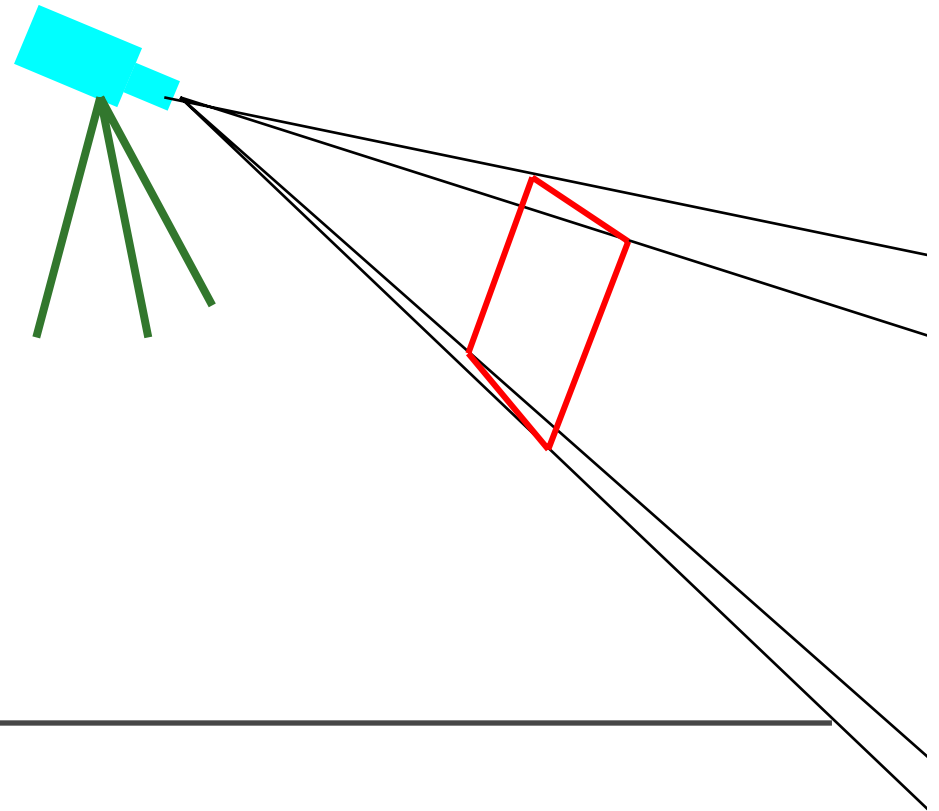


# Rendering Parameters

---

## Camera parameters

- Location
- Orientation
- Focal length



# 3D vs. 2D

---

## 2D

- X, Y - 2 dimensions only
- We'll not spend much time on this

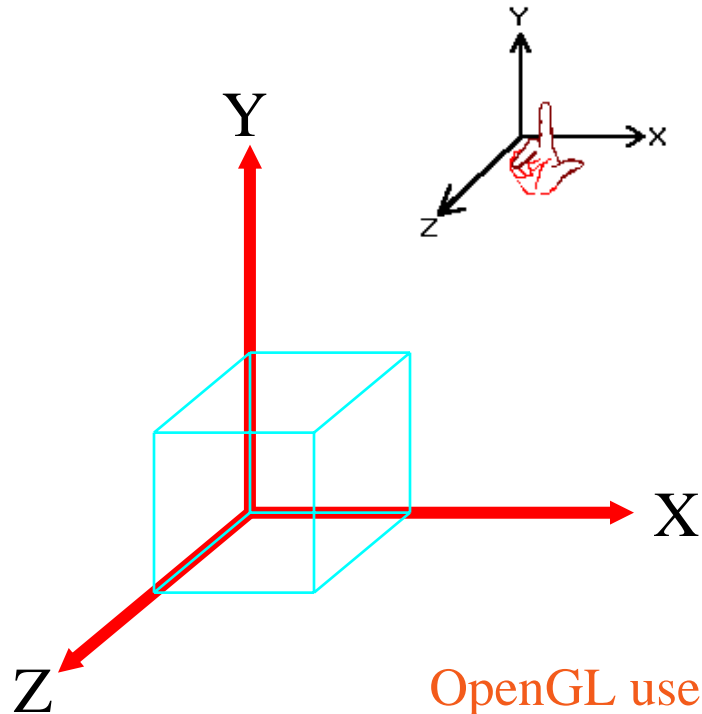
## 3D

- X, Y, and Z
- Space

Rendering is typically the conversion of  
3D to 2D

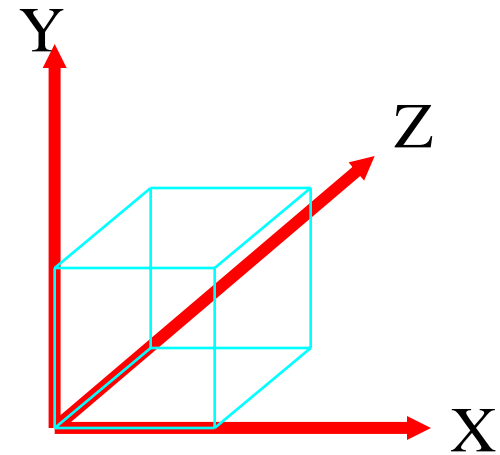
---

# 3D coordinate systems



OpenGL uses this!

Right-Hand  
Coordinate System

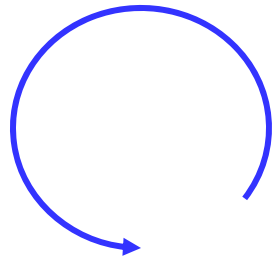


Direct3D uses this!

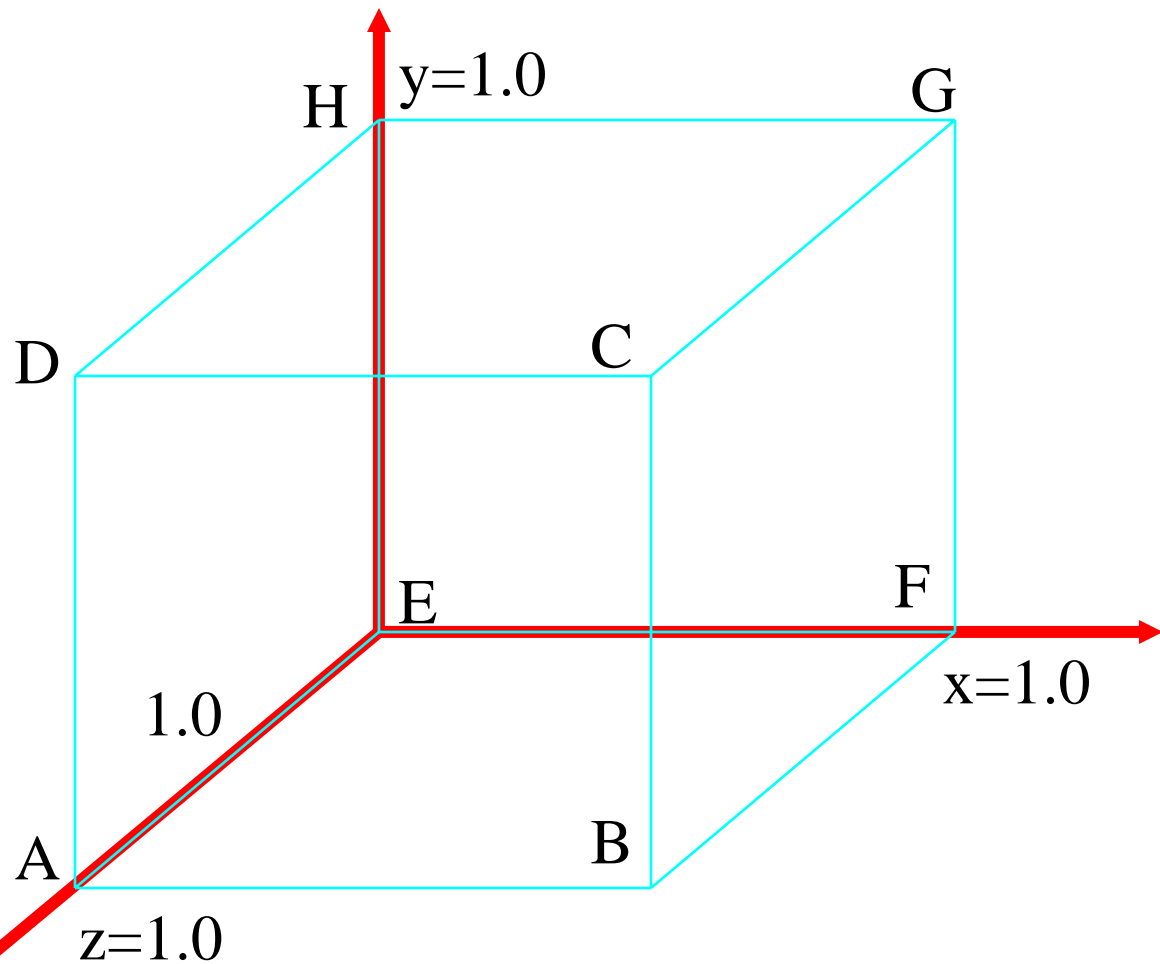
Left-Hand  
Coordinate System

# Visualizing in 3D

---



Counter-clockwise

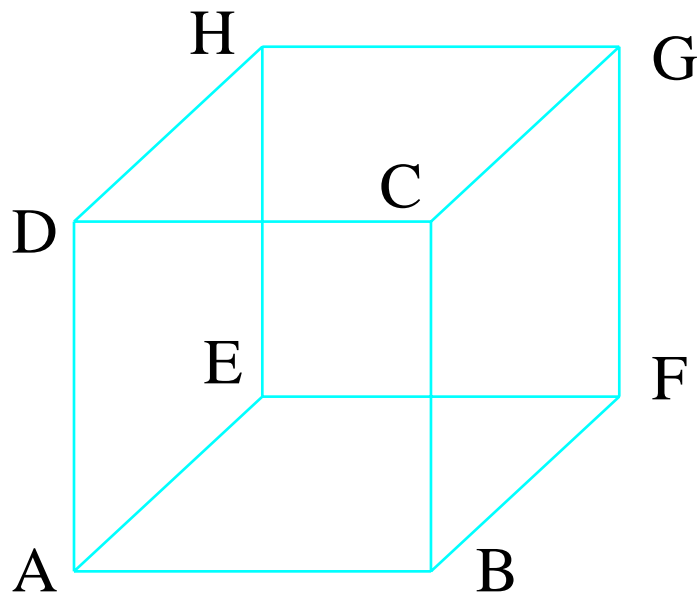


# Rendering a Box in OpenGL

---

We render the 6 faces as **polygons**

- Polygon specified as a list of vertices
- CCW order looking at the surface



# 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 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 Types

---

## Basic numeric types

- GLdouble = double
- GLfloat = float
- GLint = int
- GLshort = short

Mostly you'll use GLdouble and GLfloat

---

# Function suffixes

---

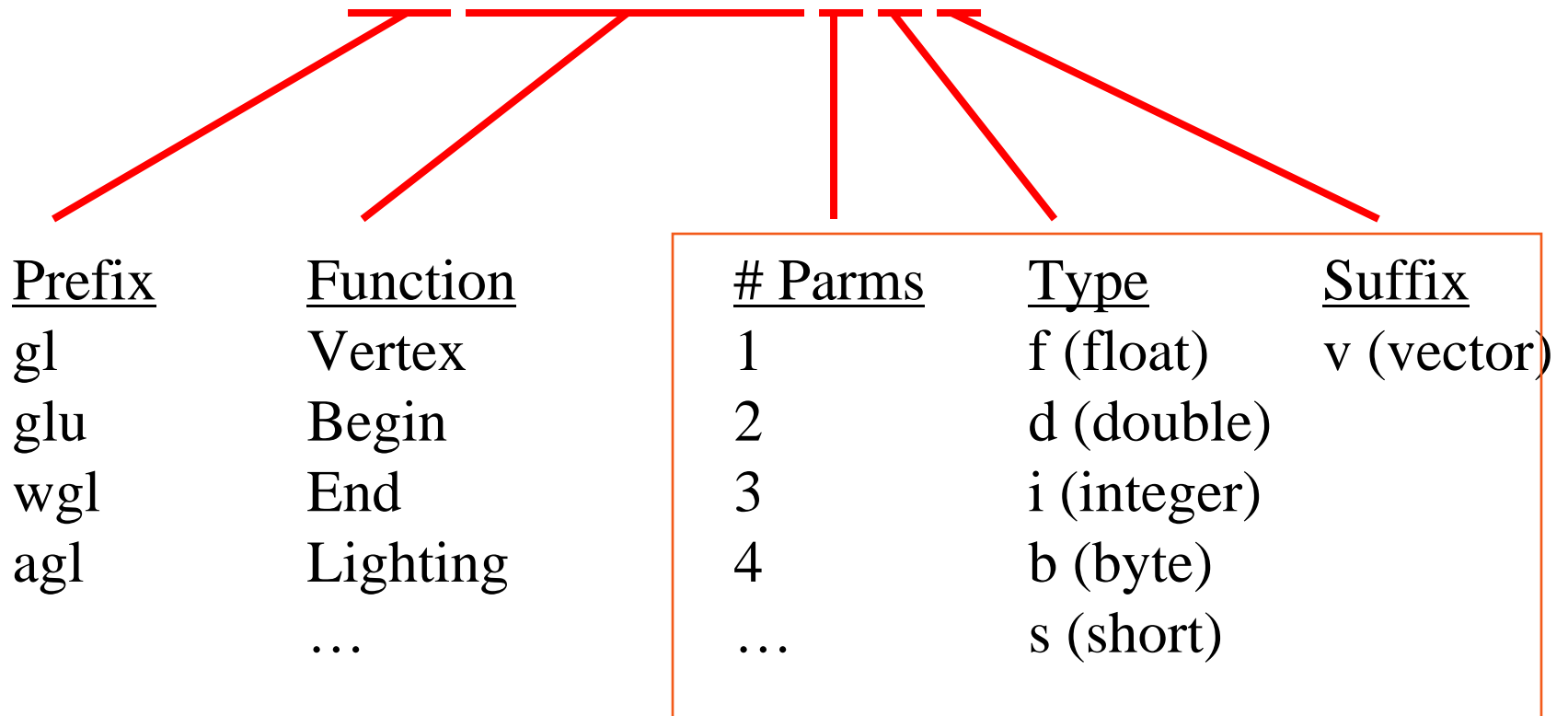
Many functions have alternatives

- 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);`

# Defined

---

## glVertex3fv



All of dem...

---

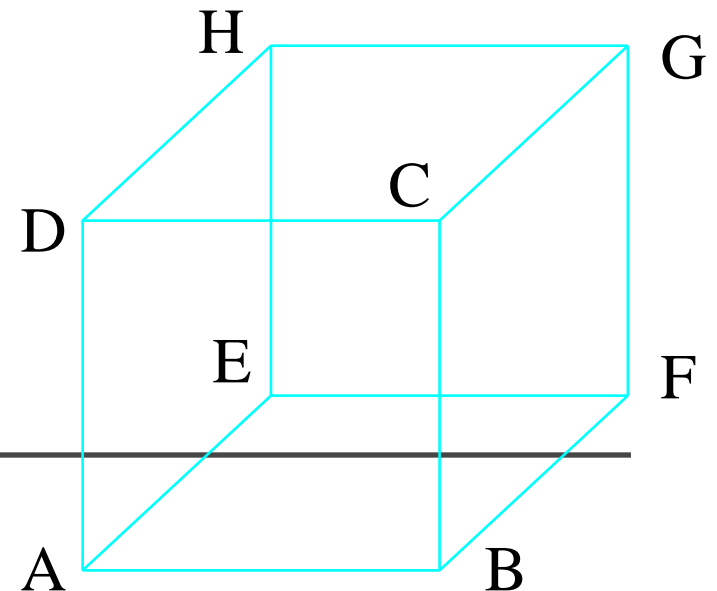
**glVertex2d, glVertex2f, glVertex2i,  
glVertex2s, glVertex3d, glVertex3f,  
glVertex3i, glVertex3s, glVertex4d,  
glVertex4f, glVertex4i, glVertex4s,  
glVertex2dv, glVertex2fv, glVertex2iv,  
glVertex2sv, glVertex3dv, glVertex3fv,  
glVertex3iv, glVertex3sv, glVertex4dv,  
glVertex4fv, glVertex4iv,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



Specifying 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

# 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 counterclockwise order

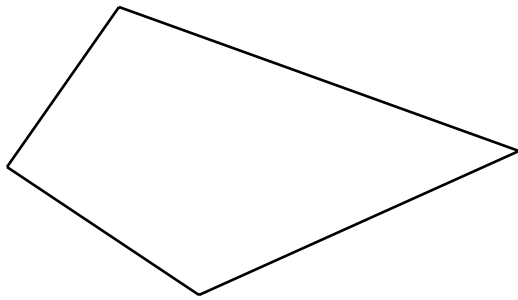
- From the visible side
-

# Convex

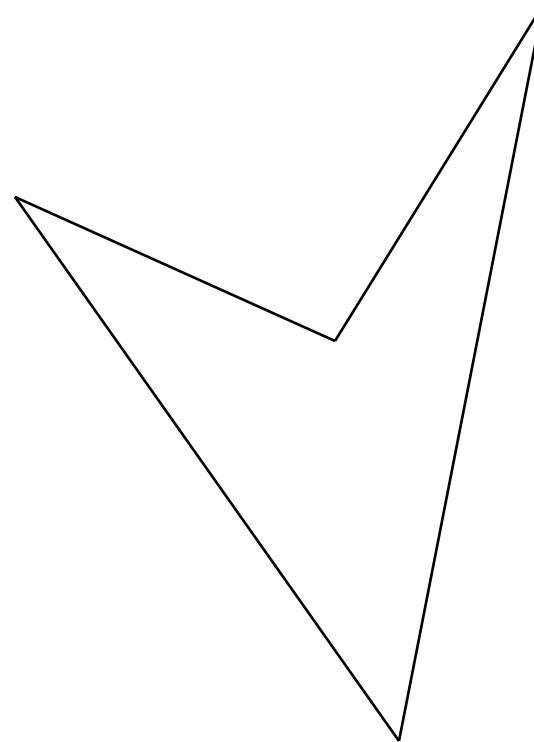
---

A polygon is convex if...

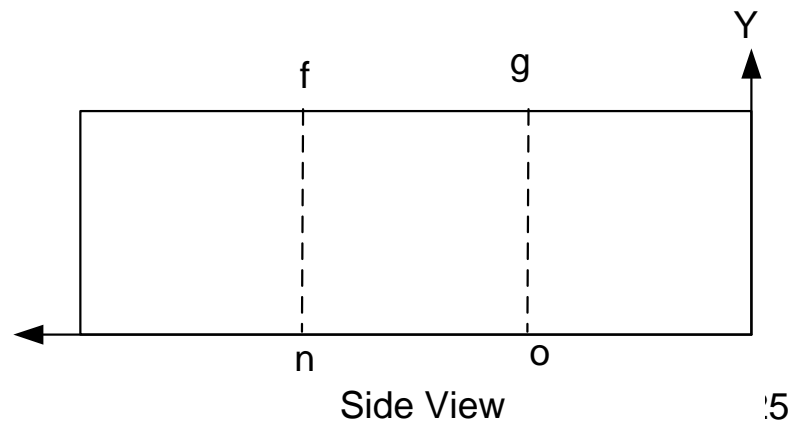
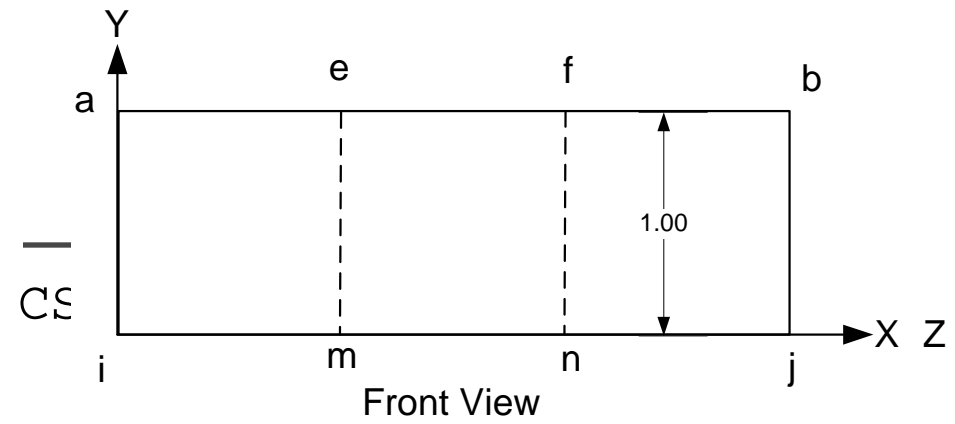
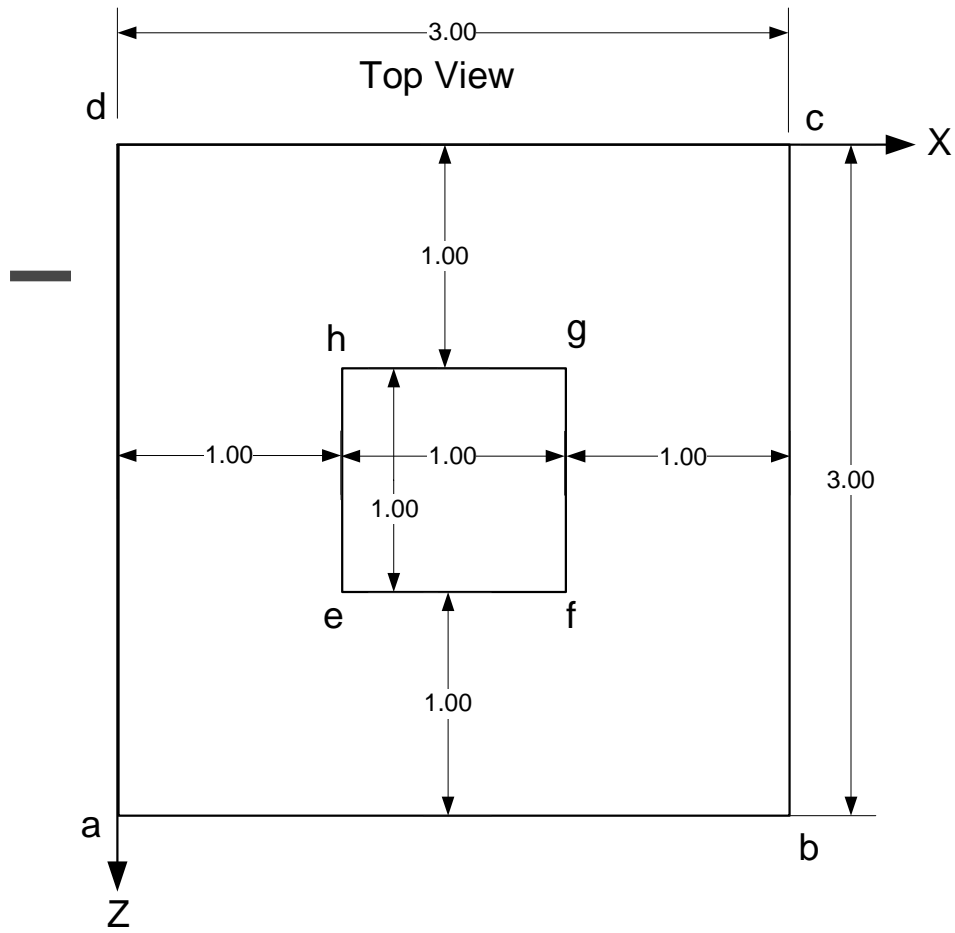
- A line segment connecting any two points on the polygon is contained in the polygon.
- If you can wrap a rubber band around the polygon and touch all of the sides, the polygon is convex



Convex

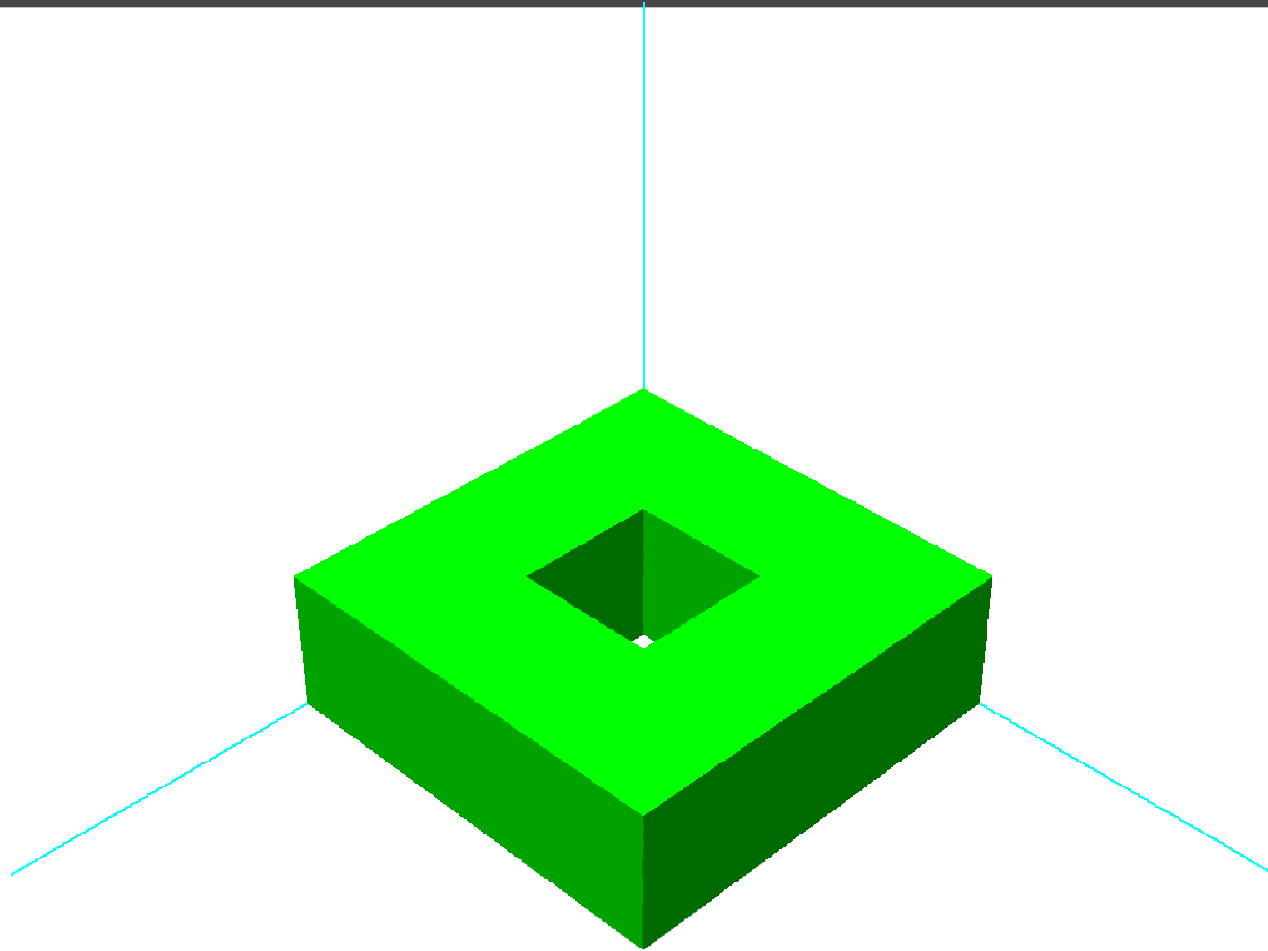


Not Convex



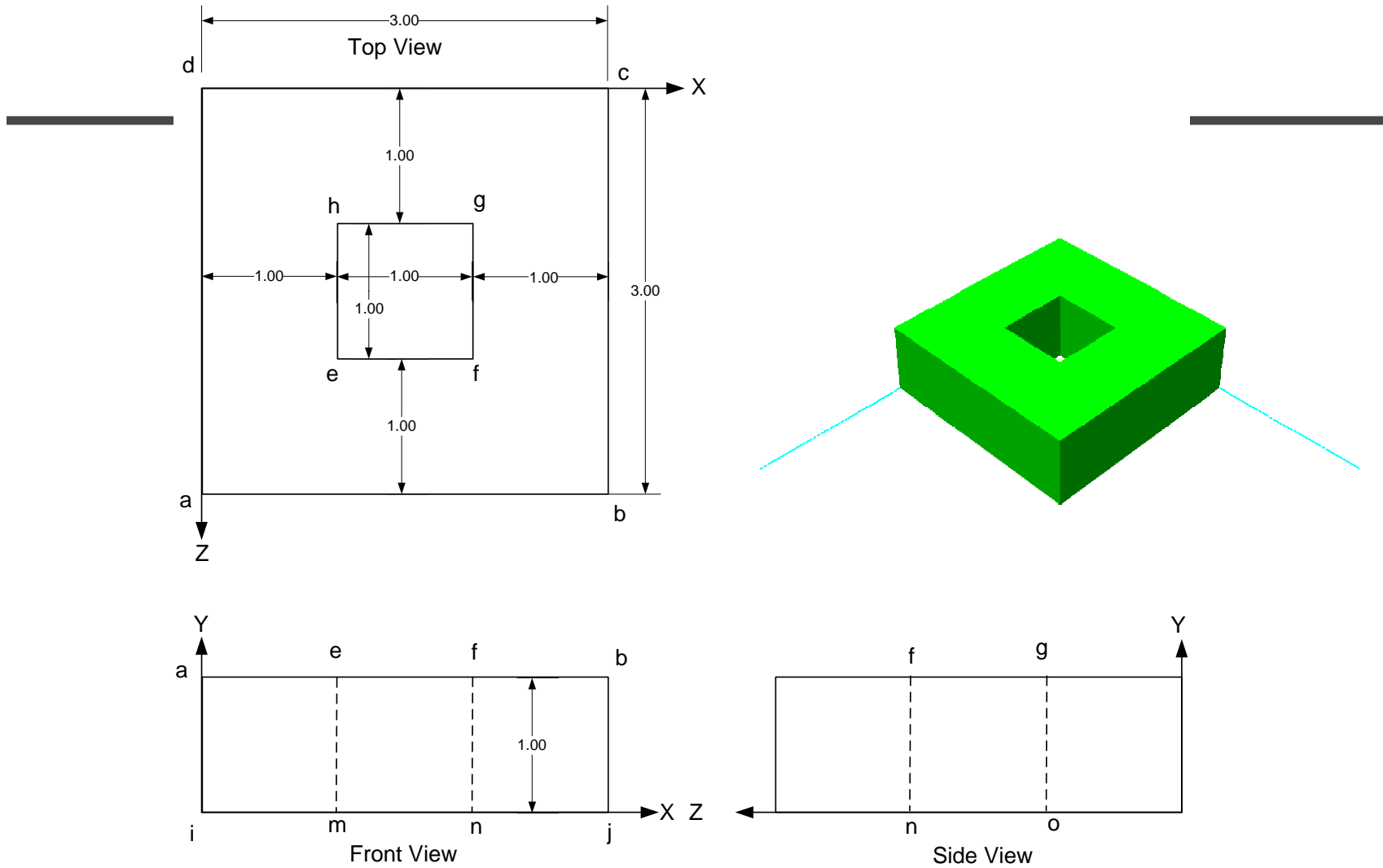
# How to model this?

---



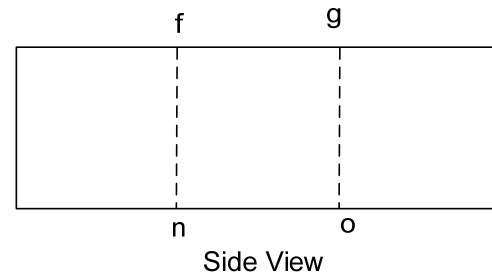
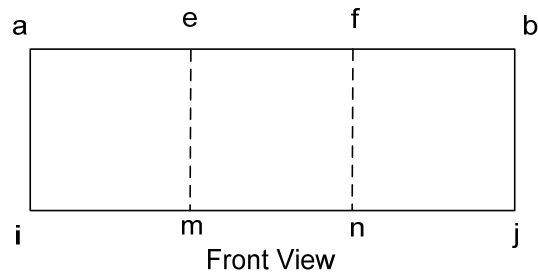
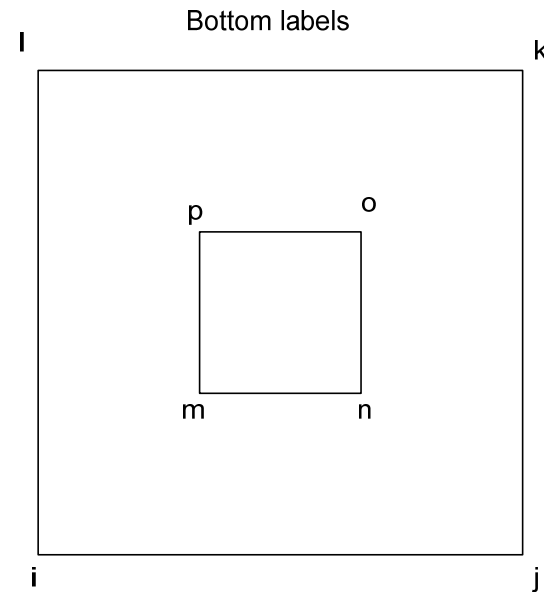
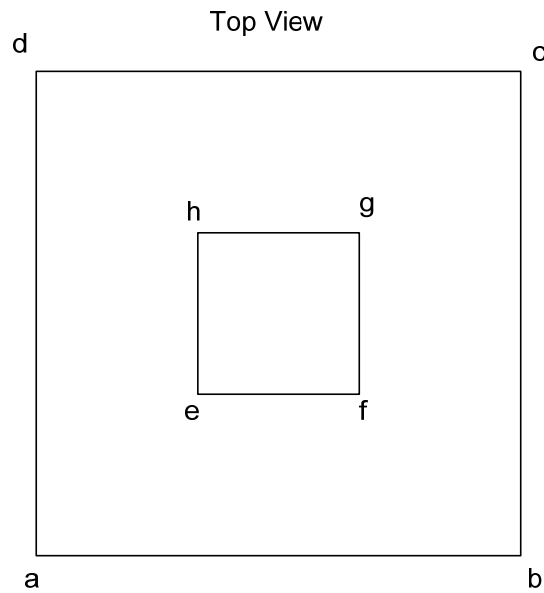
---

CSE 472 Spring 2009



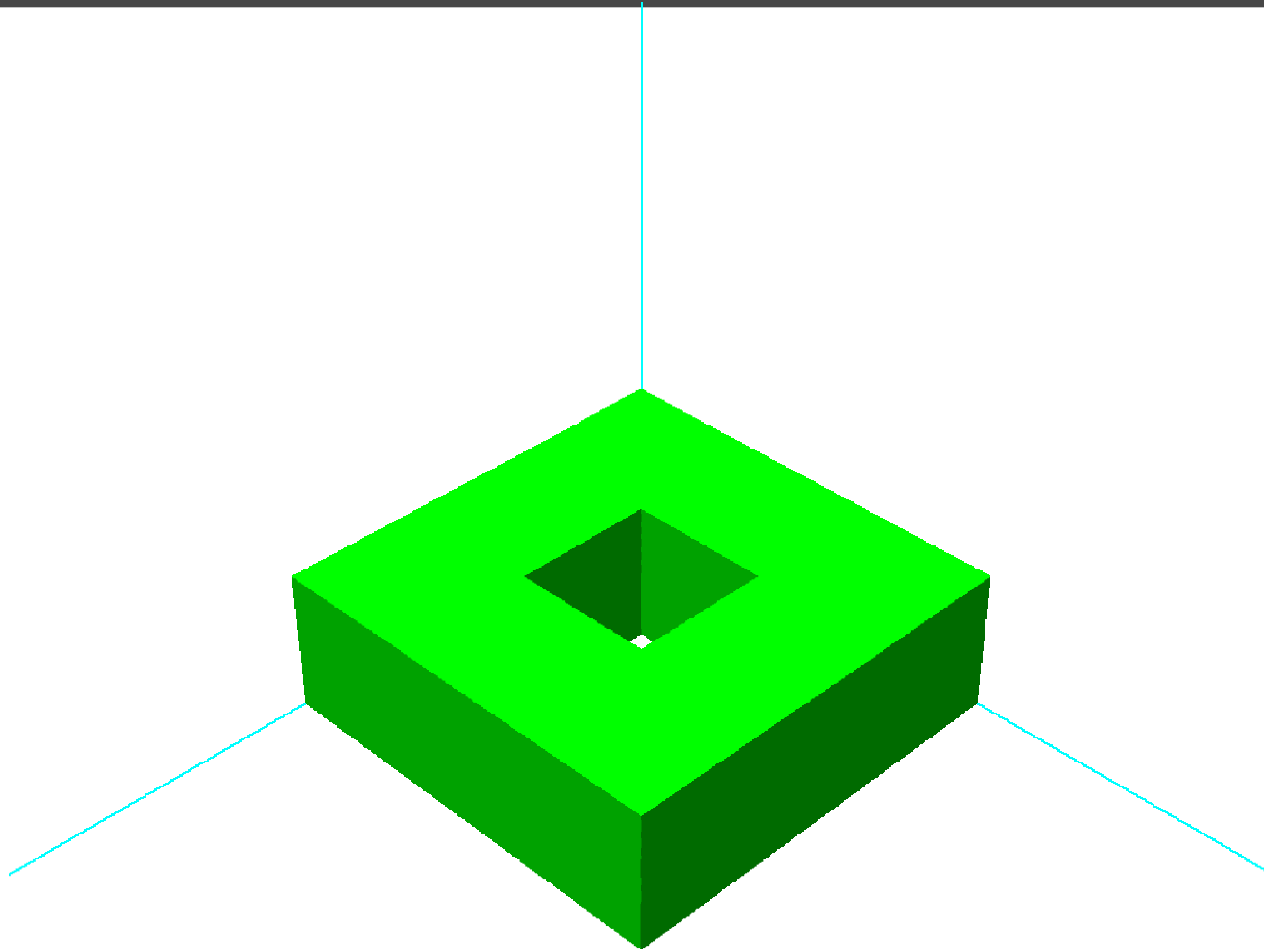
# Labels

---



# How to model this?

---



---

CSE 472 Spring 2009