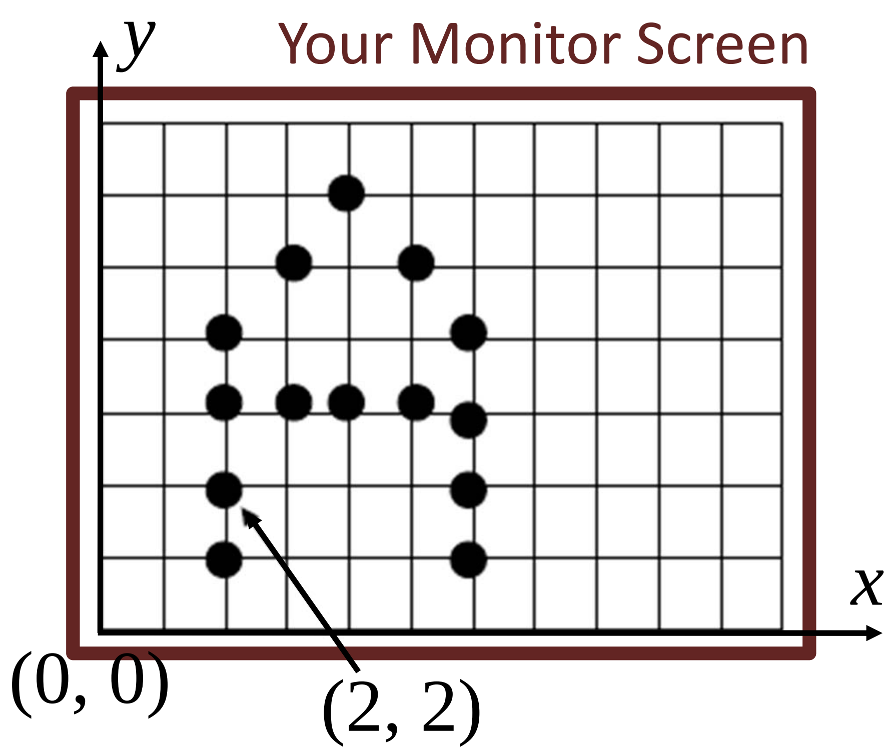
2D Coordinate Systems and Drawing

Coordinate Systems

- Screen coordinate system
- World coordinate system
- World window
- Viewport
- Window to viewport mapping

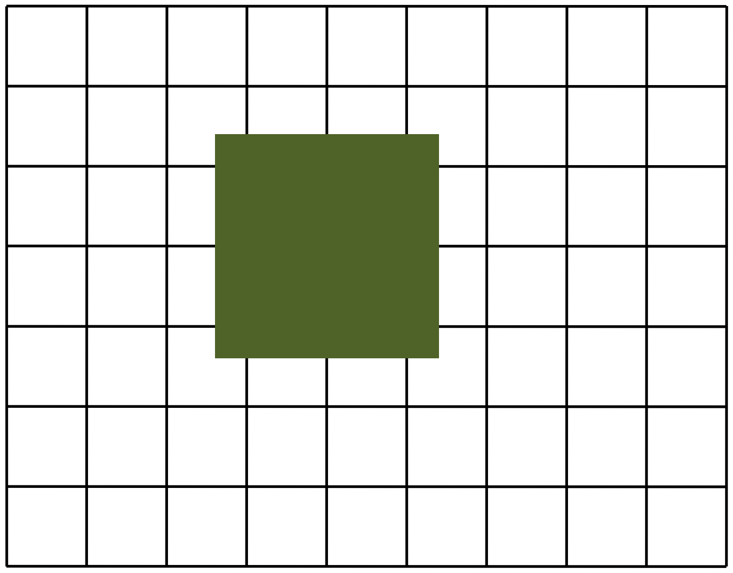
Screen Coordinate System

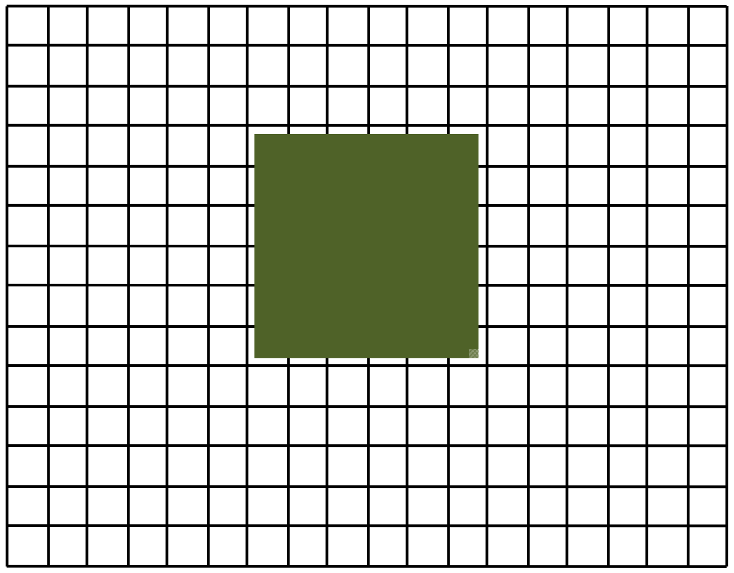
- 2D regular Cartesian grid
- Origin (0, 0) at the lower left
  (OpenGL convention)
- Pixels are defined at intersections
- Defined relatively to the display window



Screen Coordinate System

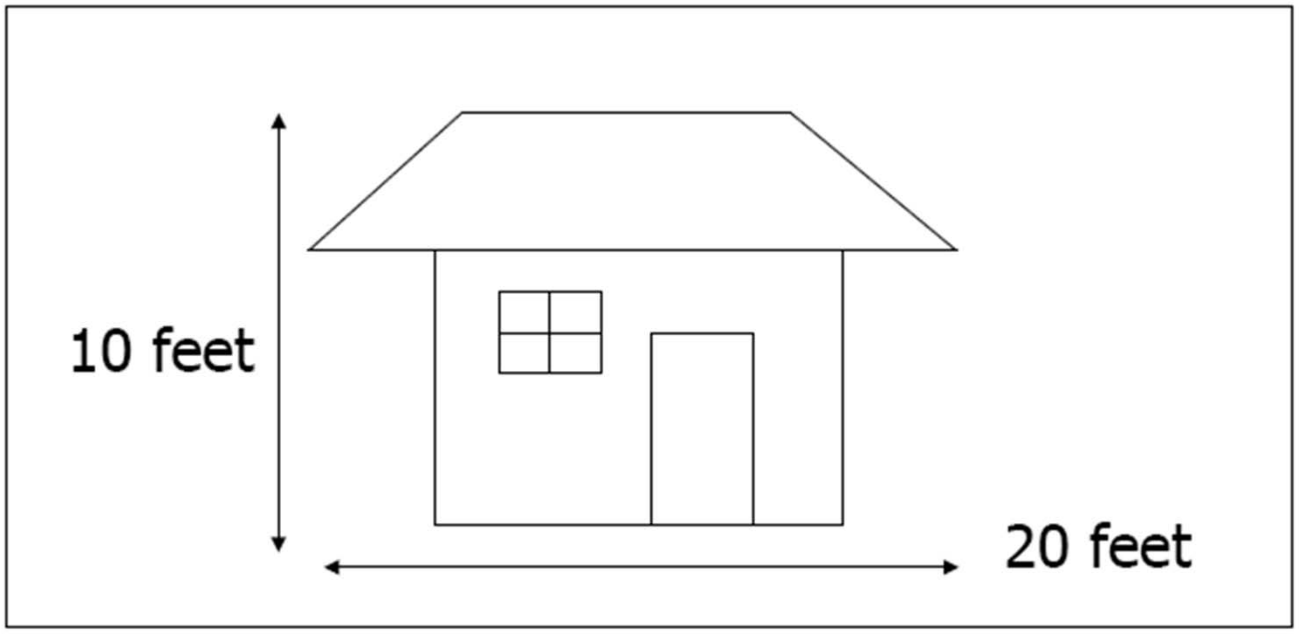
- Not easy to use in practice
  - Window size can vary





Screen Coordinate System

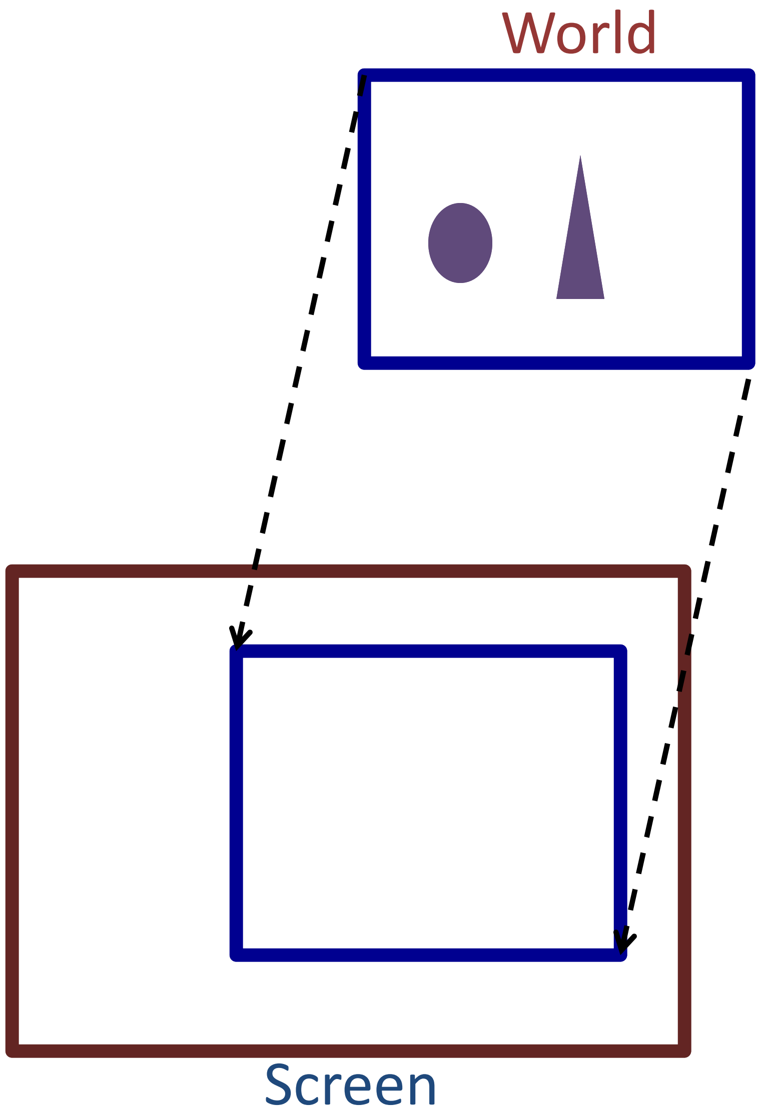
- Not easy to use in practice
  - Window size can vary
  - People prefer to specify objects in their actual sizes



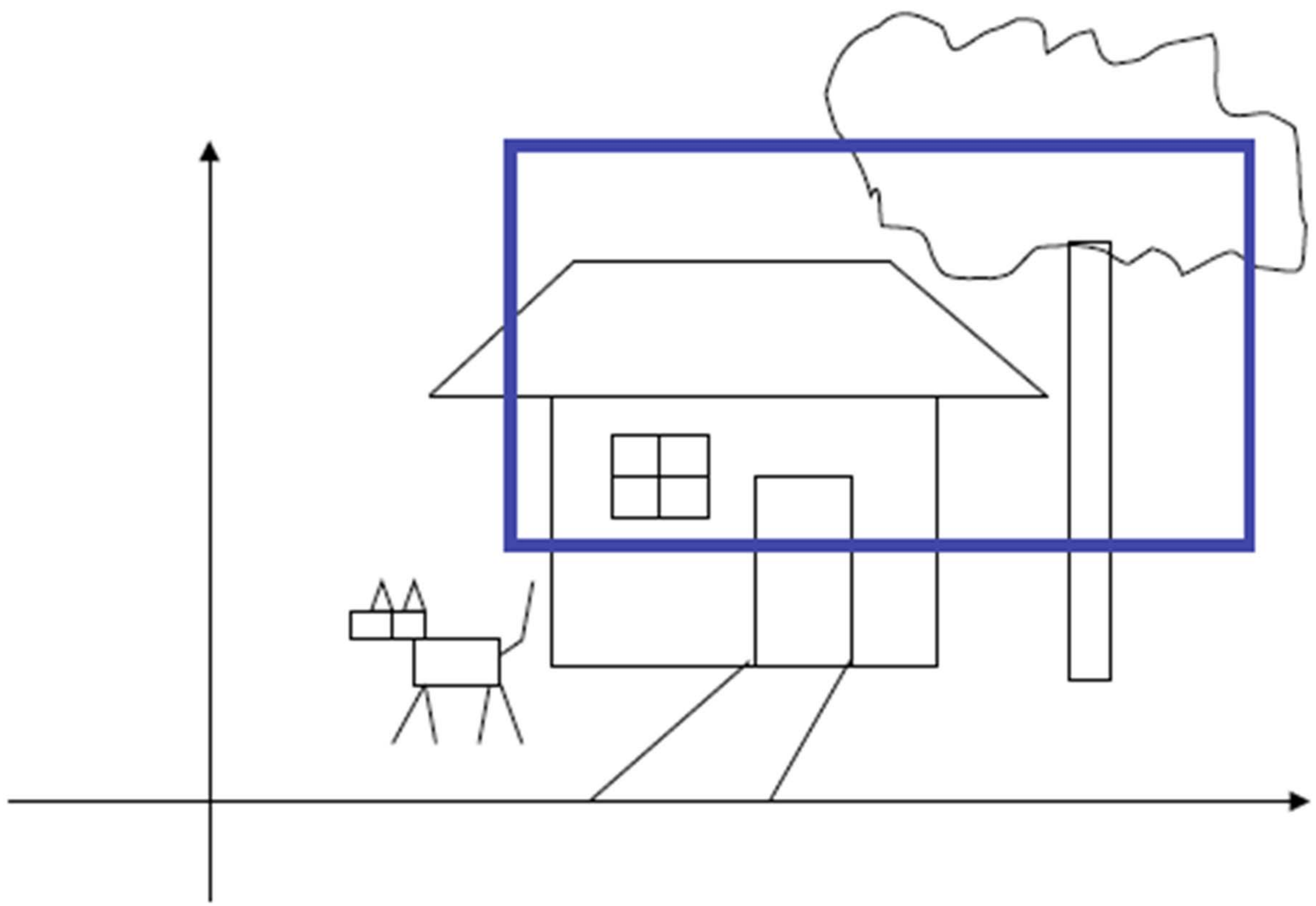
Objects should be specified independent of the screen coordinate system.

2D Drawing

$$\begin{tikzcd}
\text{Objects (in world coordinate system)} \arrow[d] \\
\text{World Window} \arrow[d] \\
\text{Screen Window}
\end{tikzcd}$$

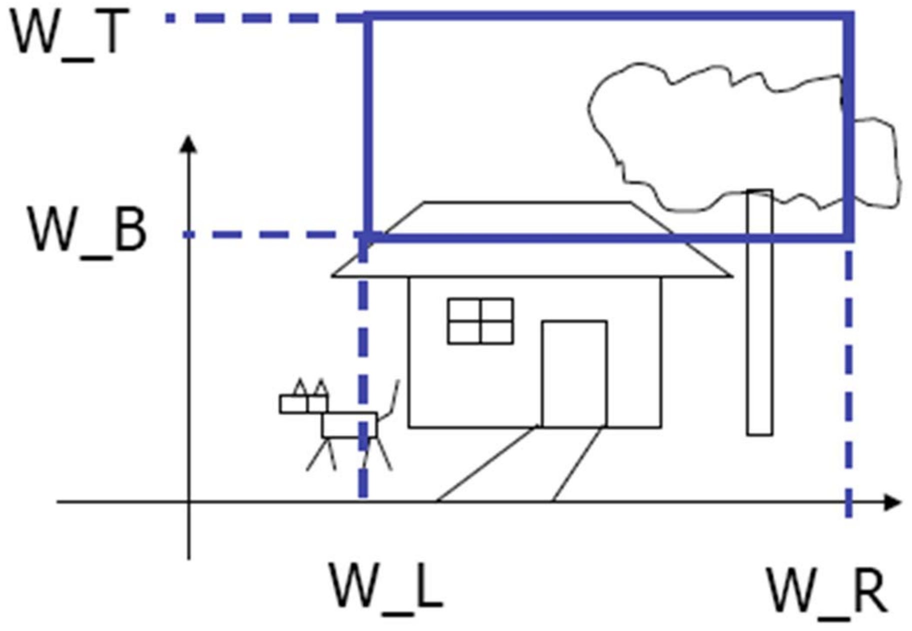


Define a world window



Define a world window

- A rectangular region in the world that is to be displayed (in world coordinate system)



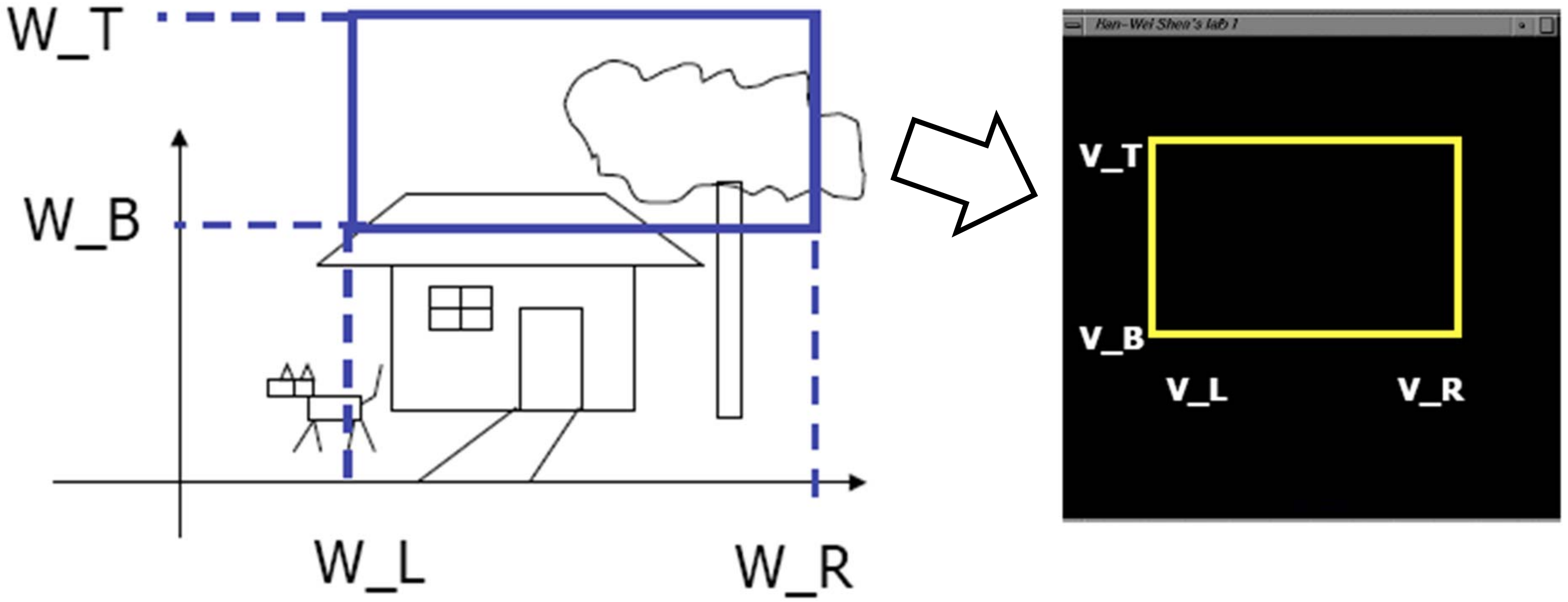
```
gluOrtho2D(W_L, W_R, W_B, W_T)
```

OpenGL function:
2D orthogonal projection

Viewport

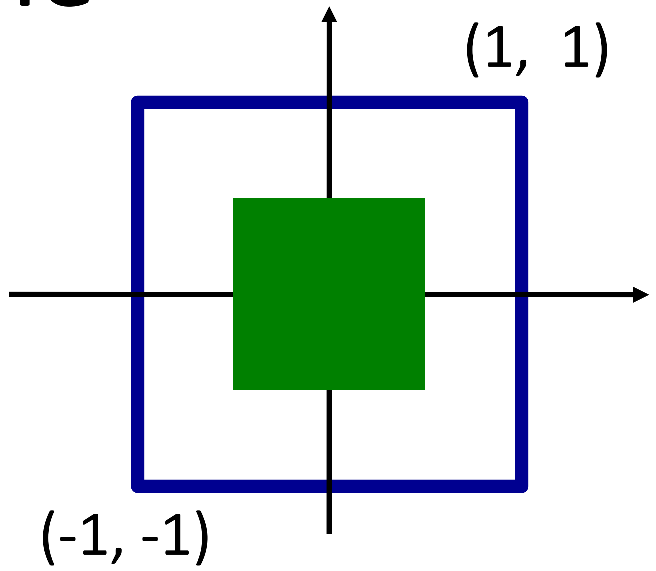
- A rectangular region in the screen for display (in screen coordinate system)

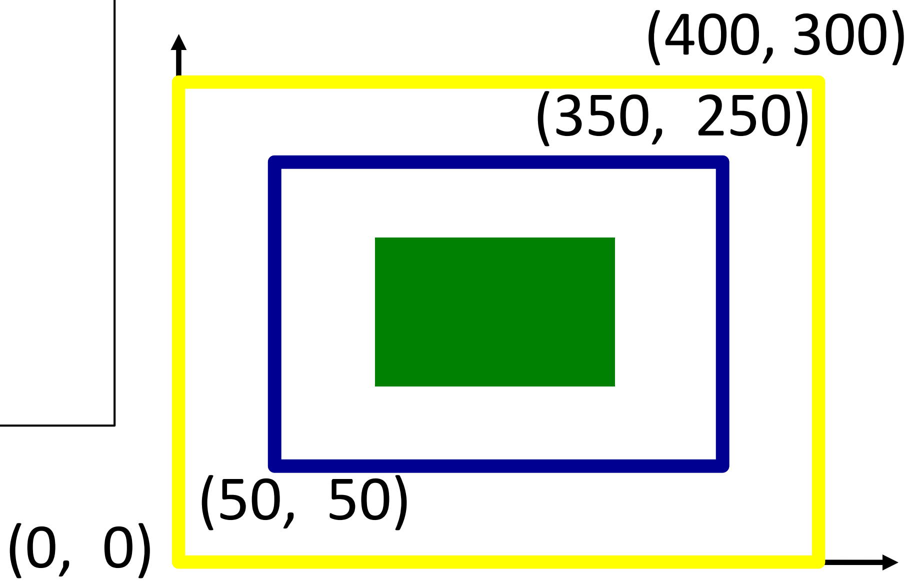
```
glViewport(V_L, V_R, V_B, _T)
```



An Example

```
void DrawQuad()
{
    glViewport(50, 50, 350, 250);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(-1, 1, -1, 1);
    glBegin(GL_QUADS);
    glVertex2f(-0.5, -0.5);
    glVertex2f( 0.5, -0.5);
    glVertex2f( 0.5,  0.5);
    glVertex2f(-0.5,  0.5);
    glEnd();
}
```



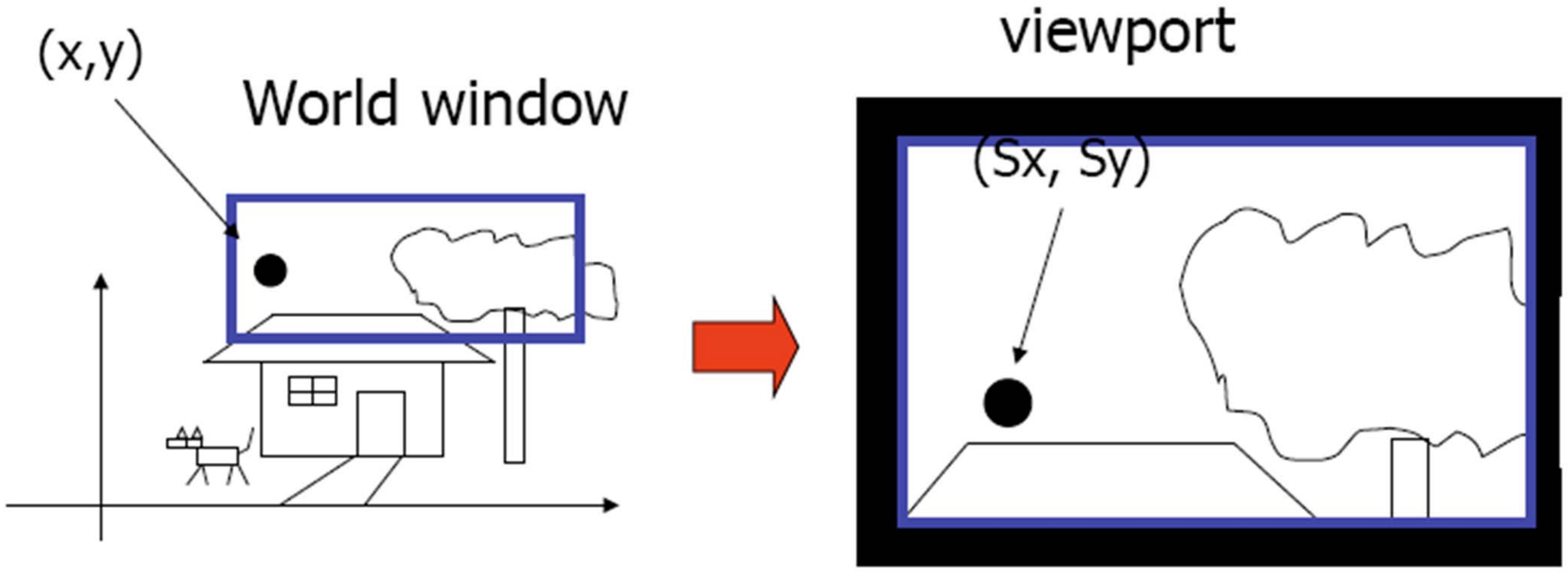


Remember to…

- Remember to specify the matrix type:

```
void DrawQuad()
{
    glViewport(50, 50, 350, 250);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(-1, 1, -1, 1);
    glBegin(GL_QUADS);
    glVertex2f(-0.5, -0.5);
    glVertex2f( 0.5, -0.5);
    glVertex2f( 0.5,  0.5);
    glVertex2f(-0.5,  0.5);
    glEnd();
}
```

How to achieve this mapping?



```
gluOrtho2D(-1, 1, -1, 1);
```

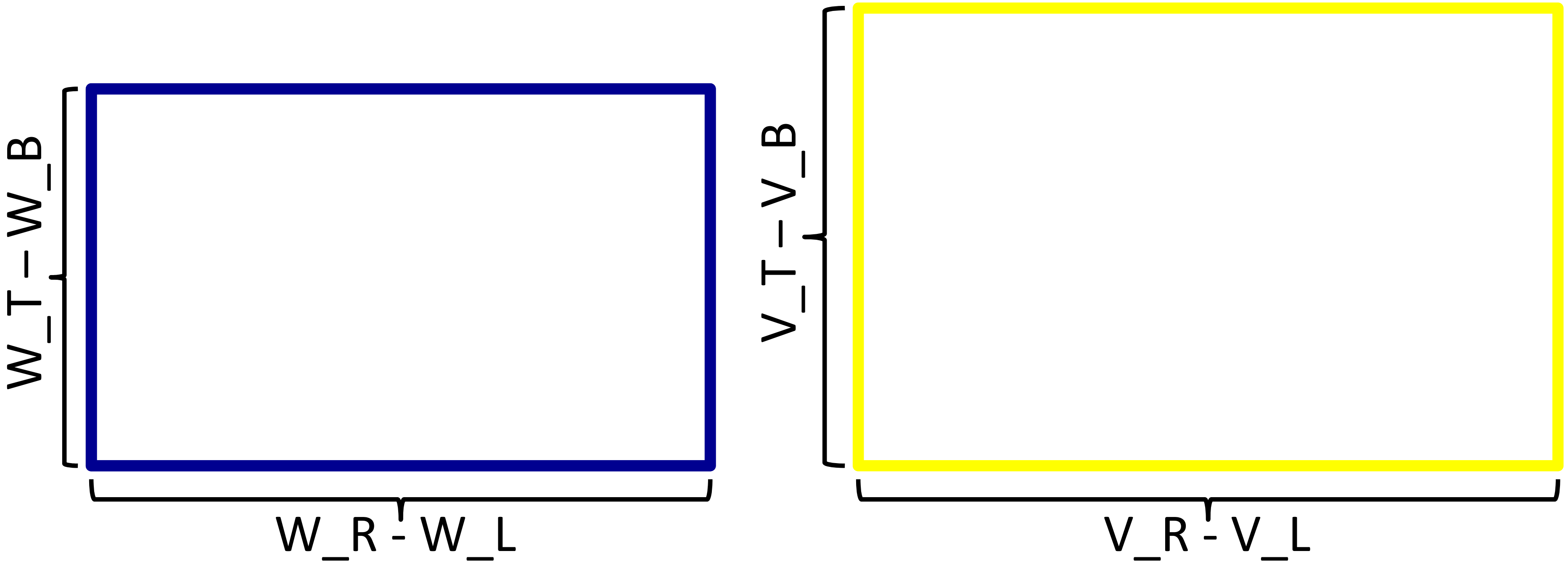
```
glViewport(50, 50, 350, 250);
```

No need to do mapping by yourself, just call those two functions!

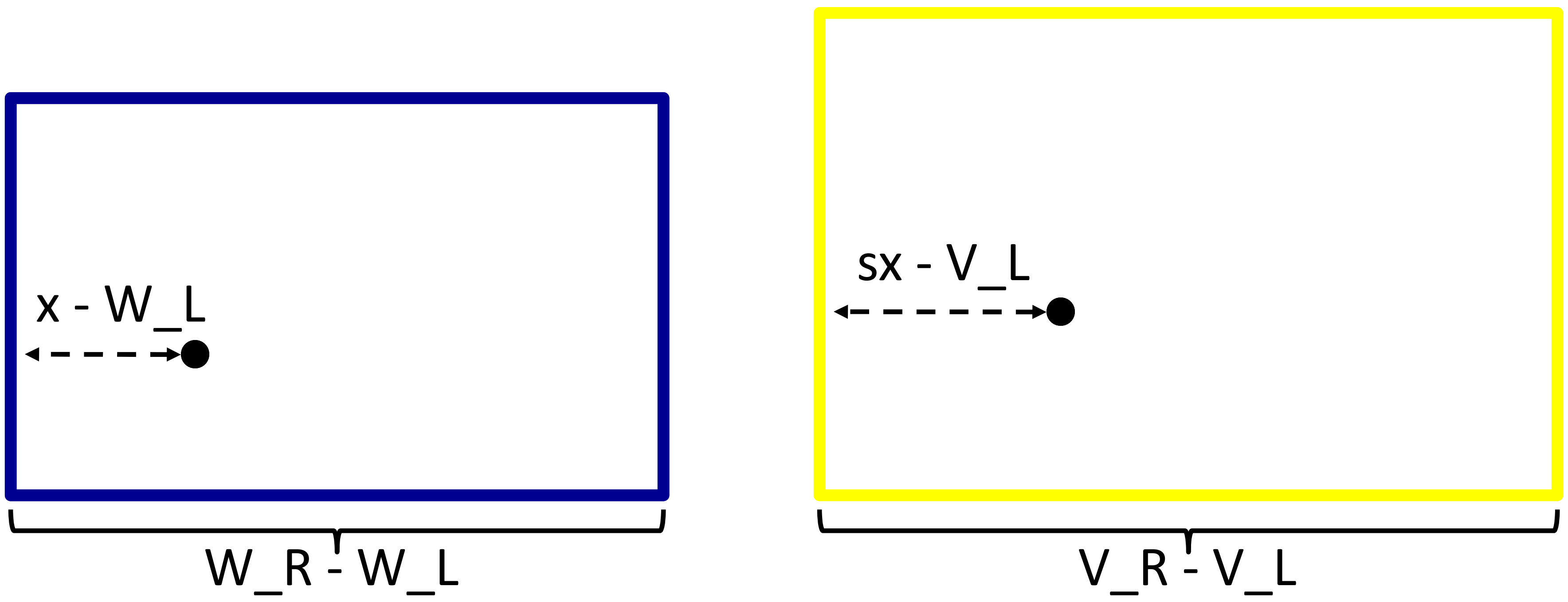
The problem

- Input:
  - World window: W_L, W_R, W_B, W_T
  - Viewport: V_L, V_R, V_B, V_T
  - Some point (x, y) in the world coordinate system
- Output:
  - (sx, sy) on the screen

Basic Information



Keep the Same Ratio

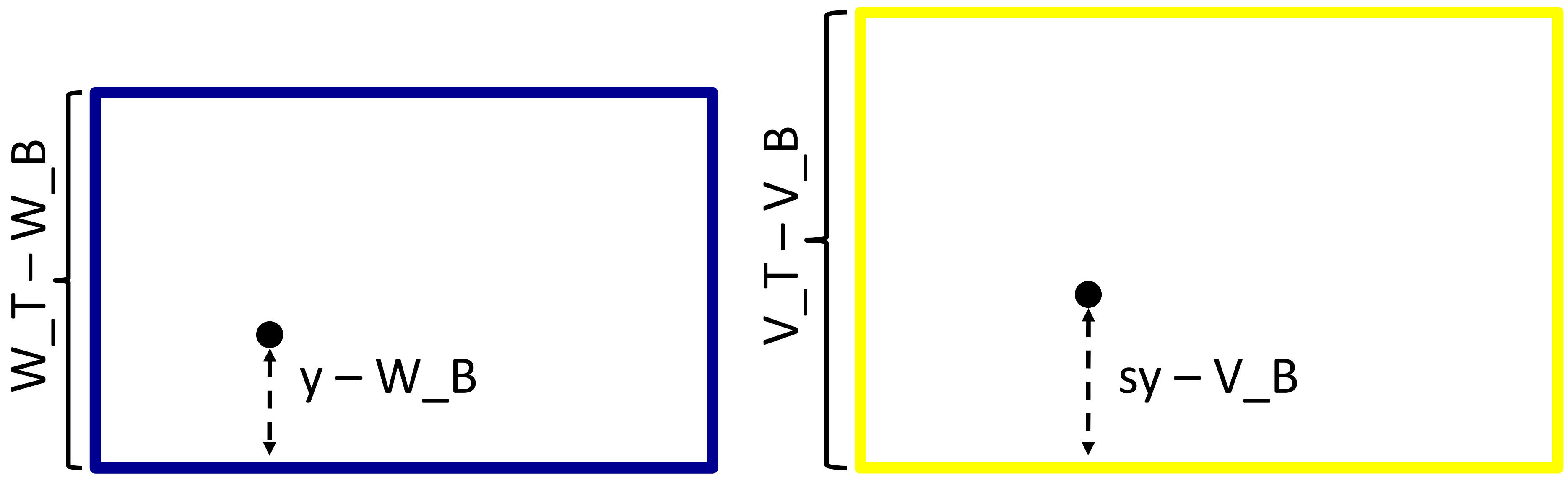


$$\boxed{(x - W\_L) / (W\_R - W\_L) = (sx - V\_L) / (V\_R - V\_L)}$$

$$\Downarrow$$

$$\boxed{sx = (x - W\_L)(V\_R - V\_L) / (W\_R - W\_L) + V\_L}$$

Keep the Same Ratio



$$\boxed{(y - W\_B) / (W\_T - W\_B) = (sy - V\_B) / (V\_T - V\_B)}$$

$$\Downarrow$$

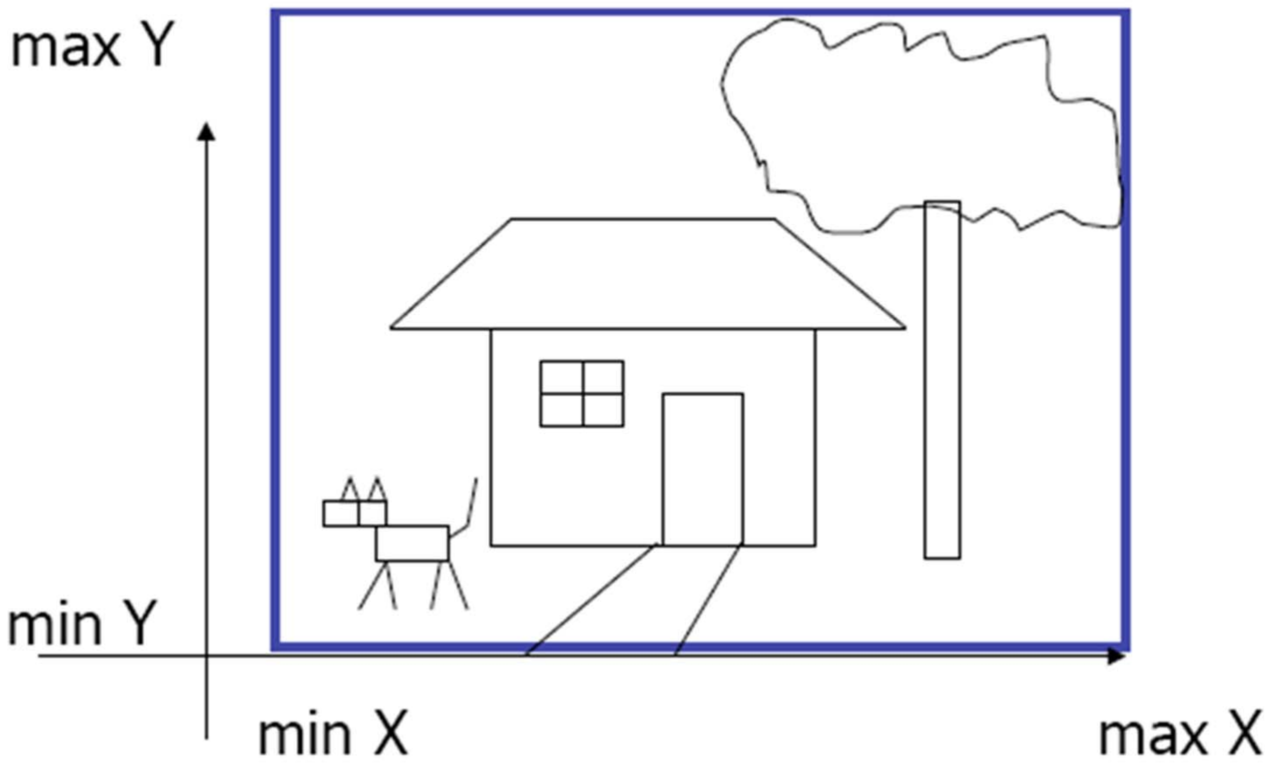
$$\boxed{sy = (y - W\_B)(V\_T - V\_B) / (W\_T - W\_B) + V\_B}$$

Practical Questions

- How to initialize
  - The world window
  - The viewport
- How to transform
  - Translation
  - Zoom in, zoom out…

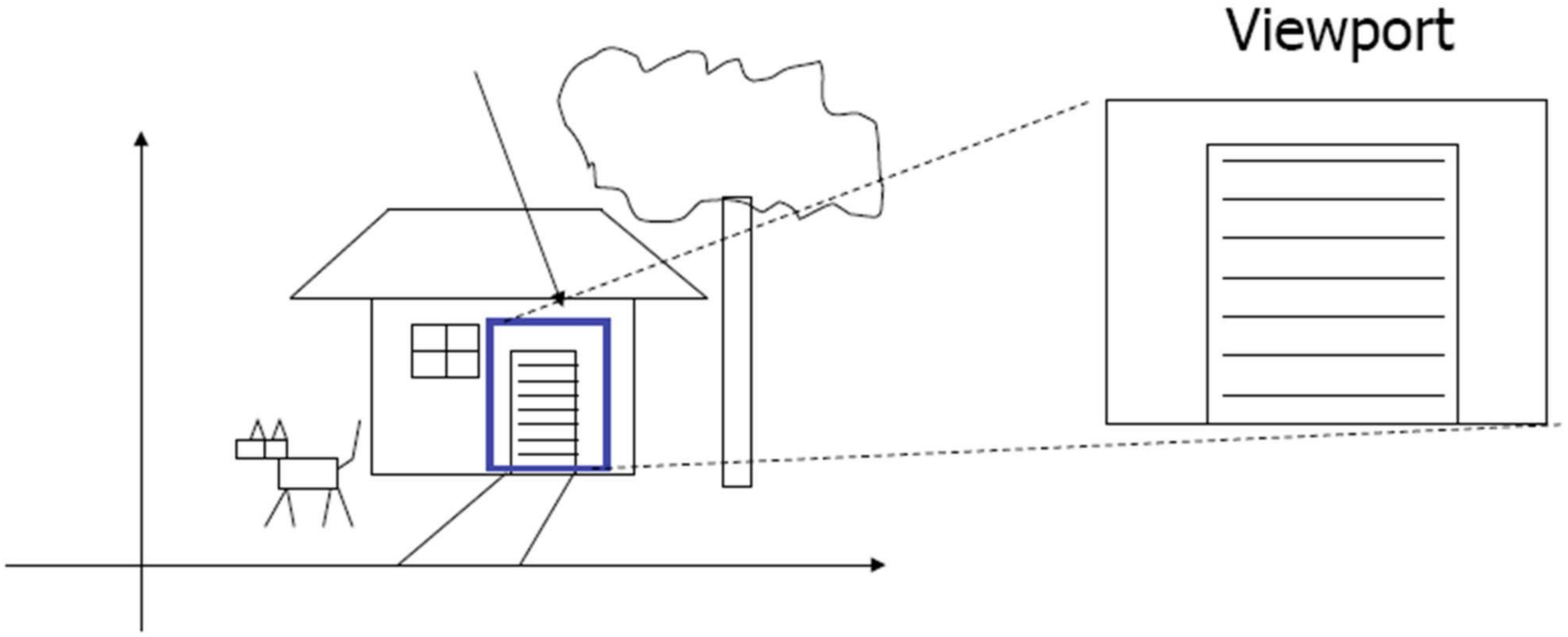
A simple way to initialize the world window

- Cover everything



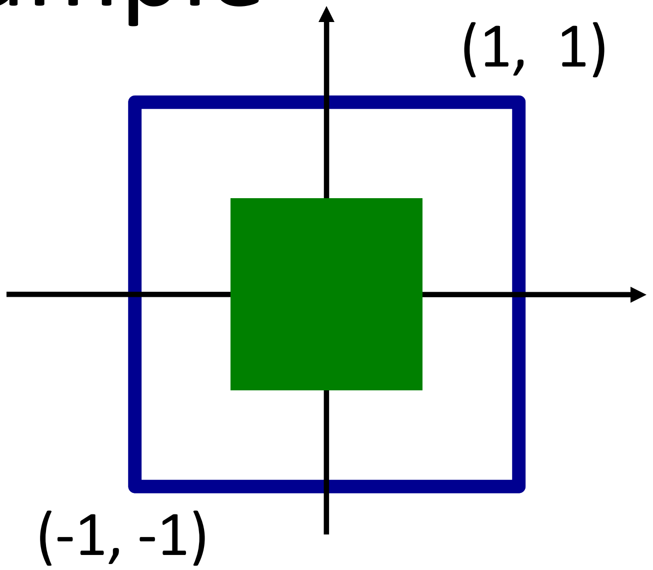
Zoom In/Out

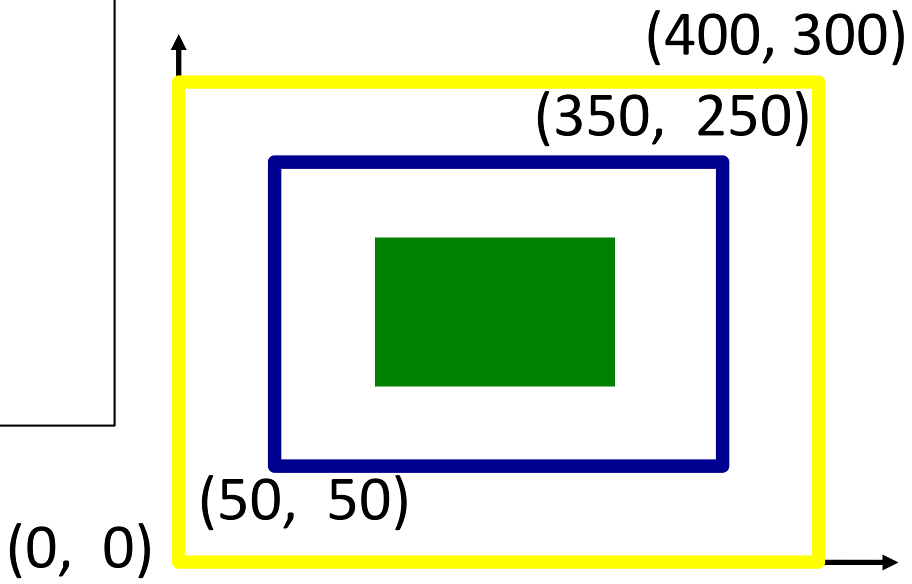
- Call gluOrtho2D() with new ranges



Distortion Example

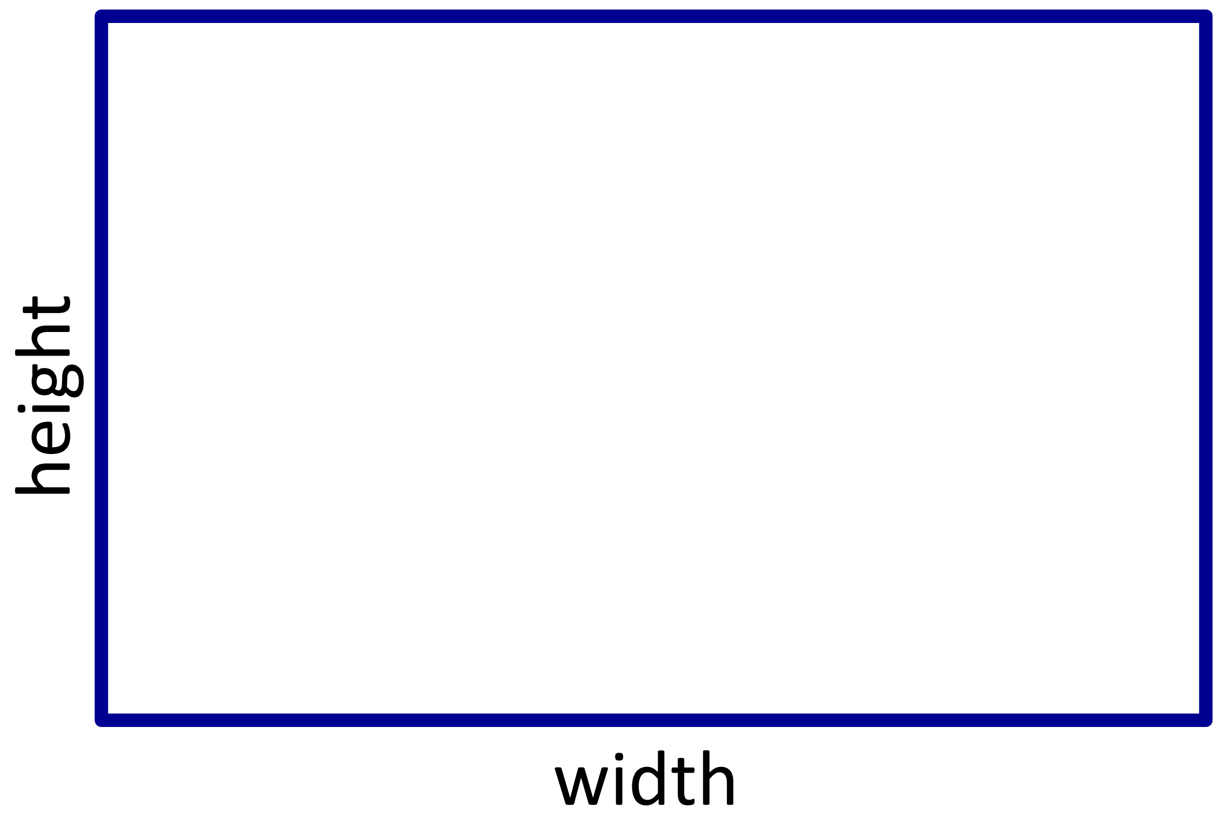
```
void DrawQuad()
{
    glViewport(50, 50, 350, 250);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(-1, 1, -1, 1);
    glBegin(GL_QUADS);
    glVertex2f(-0.5, -0.5);
    glVertex2f( 0.5, -0.5);
    glVertex2f( 0.5,  0.5);
    glVertex2f(-0.5,  0.5);
    glEnd();
}
```





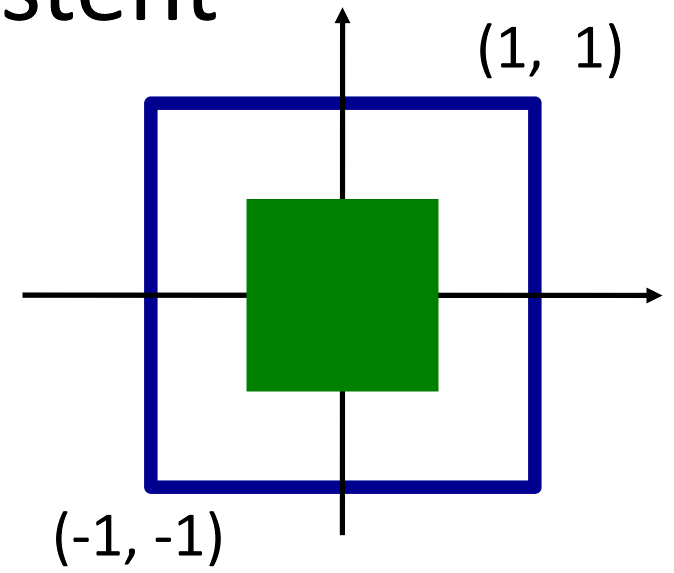
Aspect Ratio

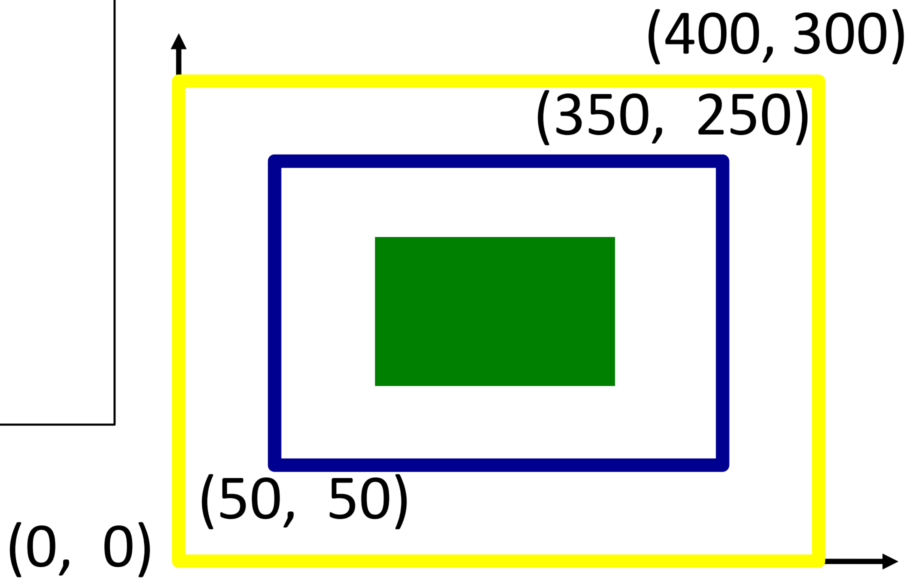
r= width/height



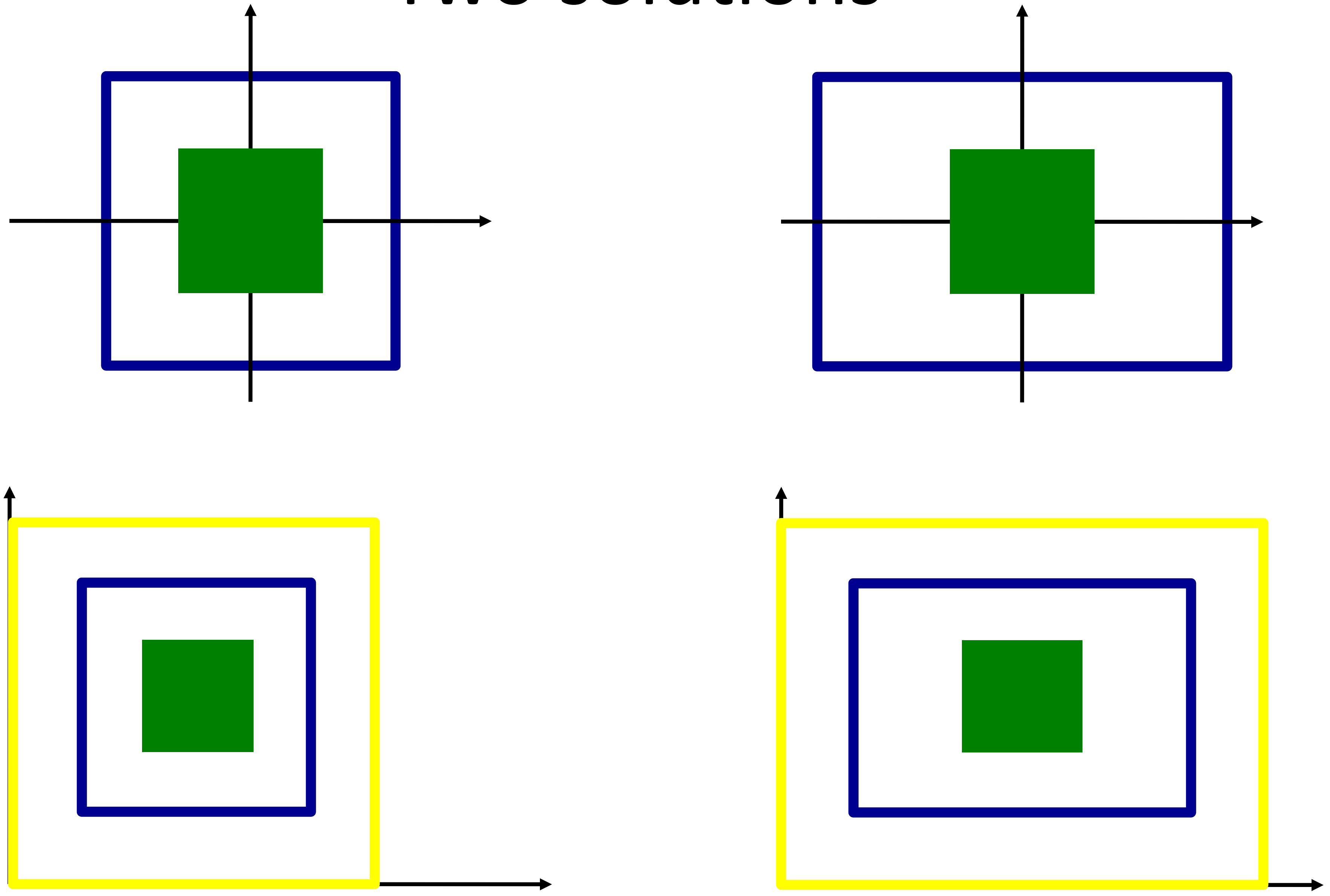
Distortion happens when aspect ratios are not consistent

```
void DrawQuad()
{
    glViewport(50, 50, 350, 250);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(-1, 1, -1, 1);
    glBegin(GL_QUADS);
    glVertex2f(-0.5, -0.5);
    glVertex2f( 0.5, -0.5);
    glVertex2f( 0.5,  0.5);
    glVertex2f(-0.5,  0.5);
    glEnd();
}
```





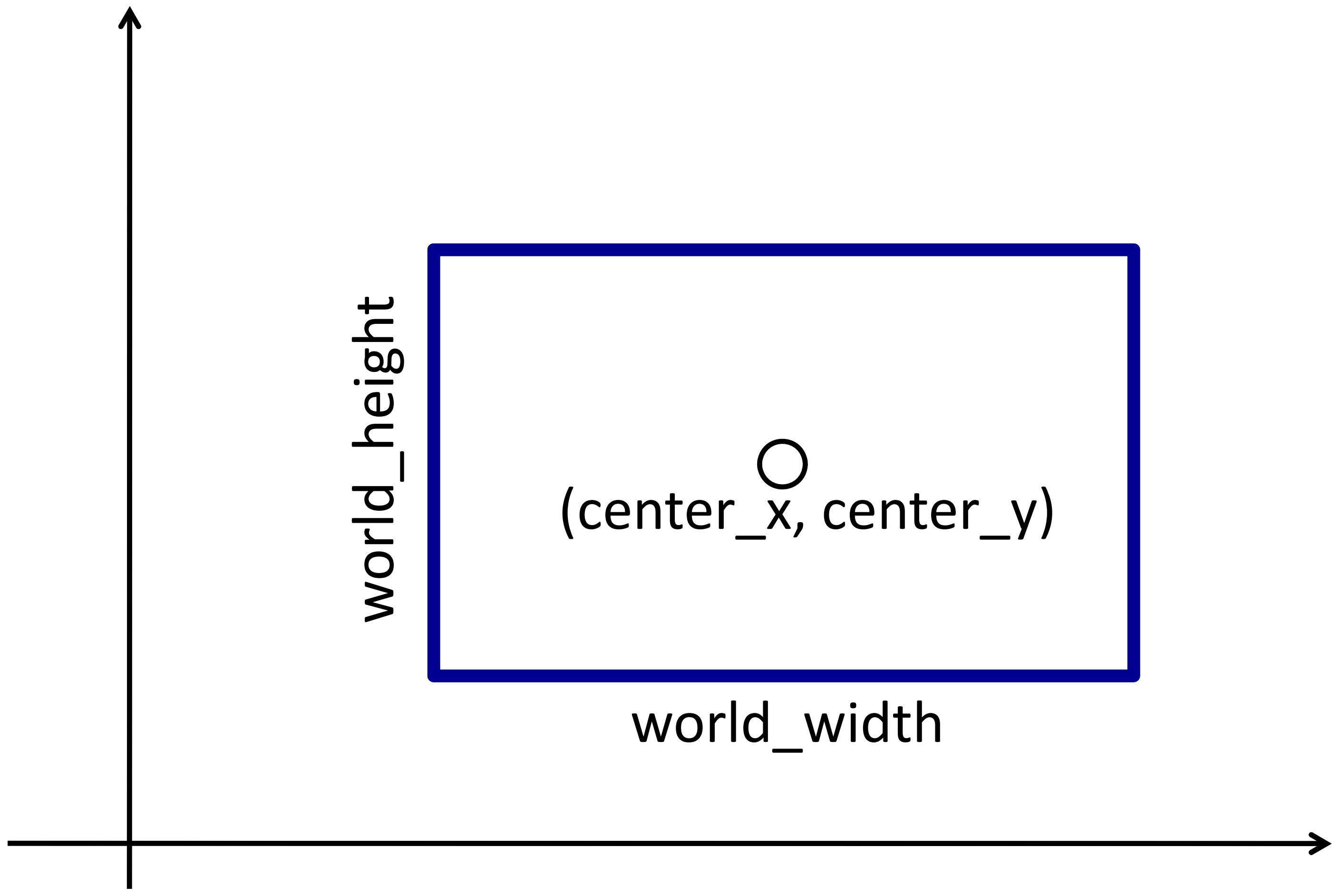
Two solutions



Where to define viewport?

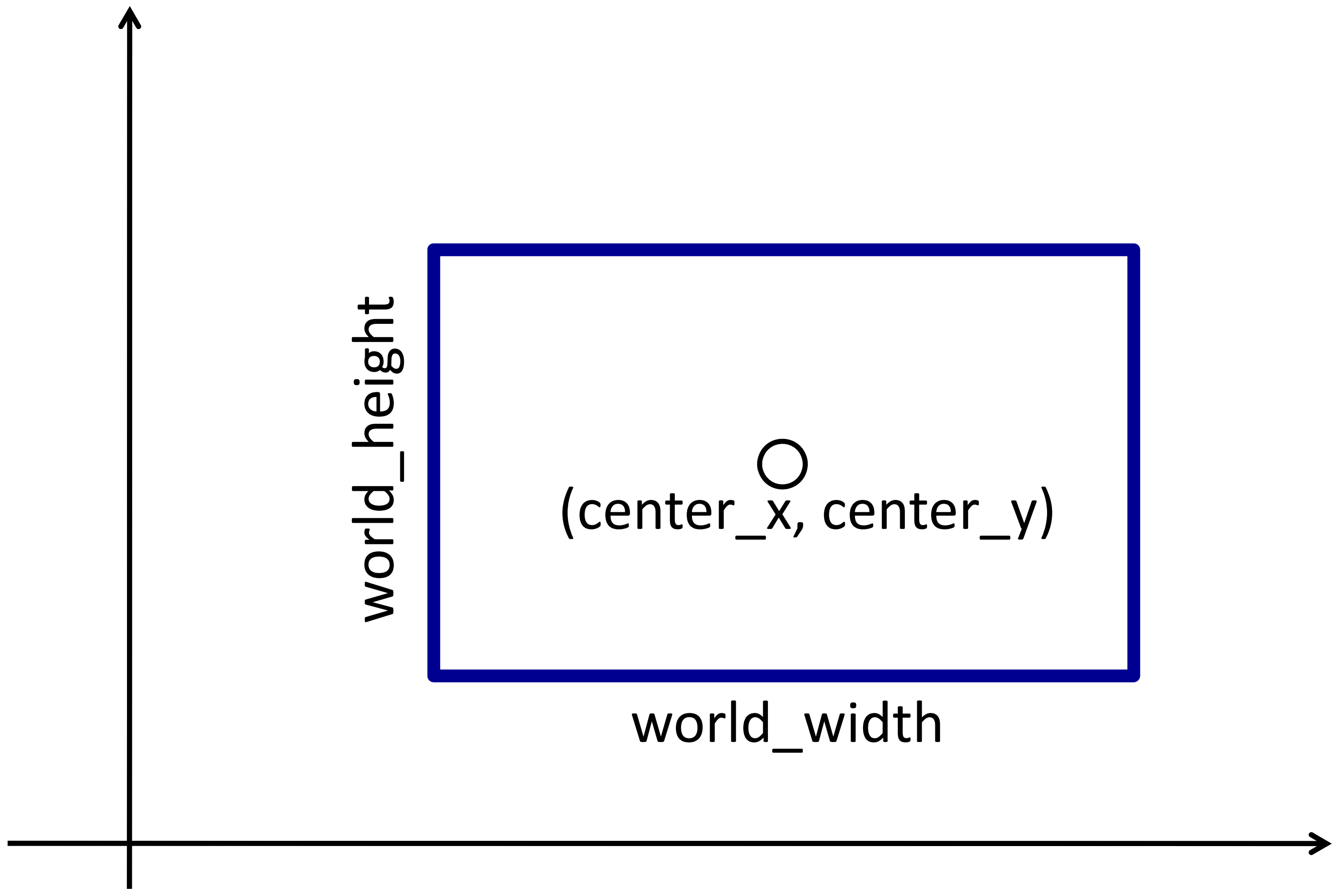
- Two places
  - Initialization: the same size as the whole window
  - Every time the user resizes the window
    - Call Viewport in your resize callback function

Example



world_height=world_width*view_port_height/view_port_width

Example



W_L=center_x-world_width/2 W_B=center_y+world_height/2

W_R=center_x+world_width/2 W_T=center_y+world_height/2