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

- Objects (in world coordinate system)
- World Window
- Screen Window

Diagram: World Window and Screen Window relation, with objects in the world coordinate system.
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)

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

```c
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?

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

\[
gluOrtho2D(-1, 1, -1, 1); \quad \text{glViewport(50, 50, 350, 250);}\]
The problem

• Input:
  – 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

- \( W_{T} - W_{B} \)
- \( W_{R} - W_{L} \)
- \( V_{T} - V_{B} \)
- \( V_{R} - V_{L} \)
Keep the Same Ratio

\[
\frac{x - W_L}{W_R - W_L} = \frac{sx - V_L}{V_R - V_L}
\]

\[
sx = (x - W_L)(V_R - V_L) / (W_R - W_L) + V_L
\]
Keep the Same Ratio

\[
\frac{y - W_B}{W_T - W_B} = \frac{sy - V_B}{V_T - V_B}
\]

\[
sy = \frac{(y - W_B)(V_T - V_B)}{(W_T - W_B)} + V_B
\]
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
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

\[ r = \frac{\text{width}}{\text{height}} \]
Distortion happens when aspect ratios are not consistent

```c
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

\[ (-0.5, -0.5), (0.5, -0.5), (-0.5, 0.5), (0.5, 0.5) \]
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

\[ \text{center}_x, \text{center}_y \]

\[ \text{center}_x - \text{world\_width}/2 \]
\[ \text{center}_x + \text{world\_width}/2 \]
\[ \text{center}_y + \text{world\_height}/2 \]
\[ \text{center}_y + \text{world\_height}/2 \]