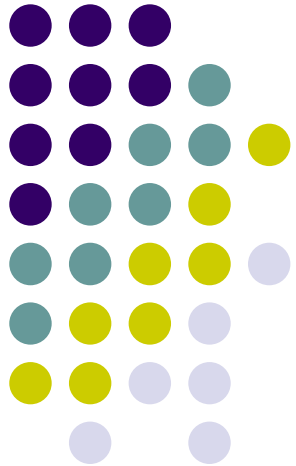
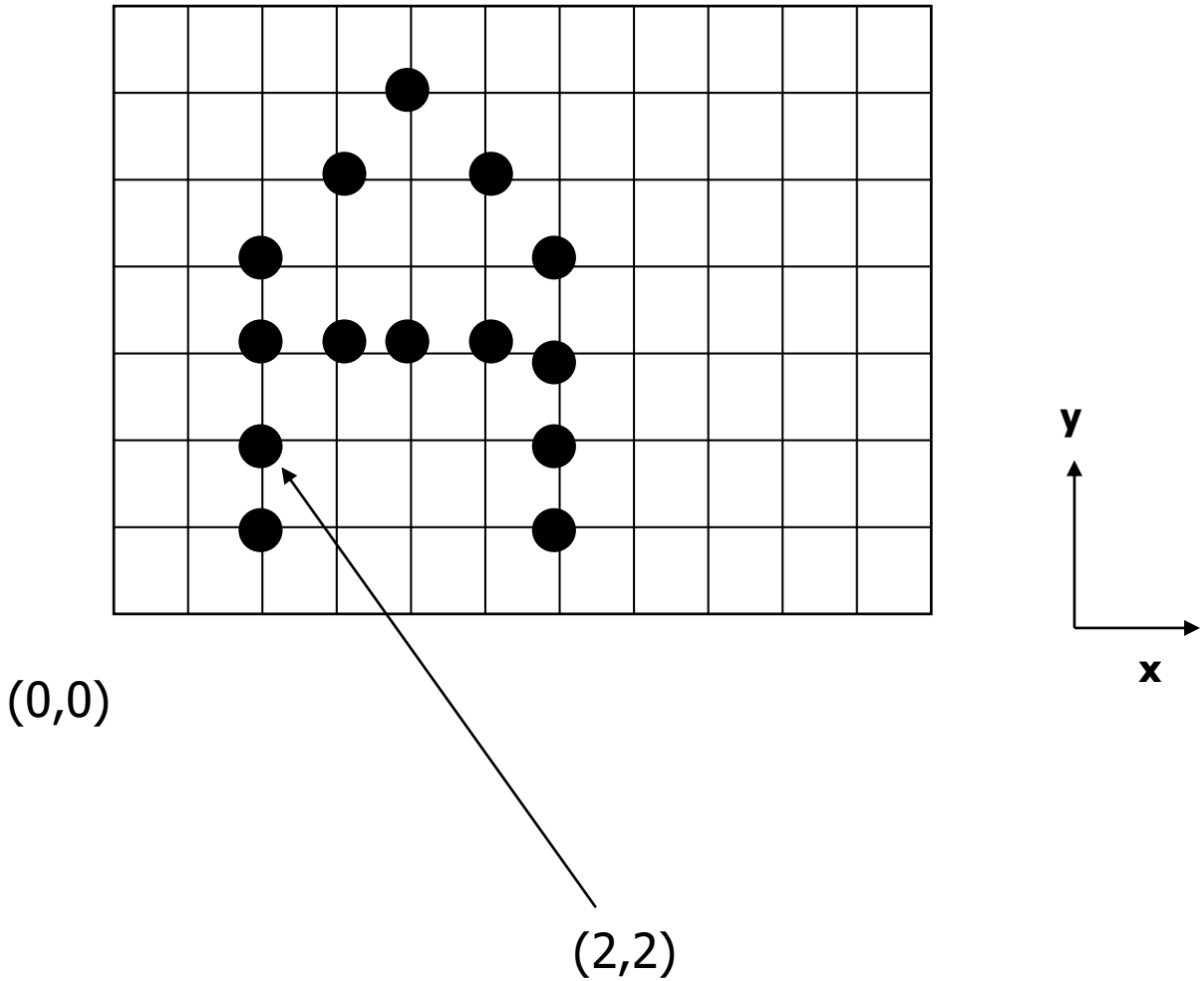
Computer Graphics (4731)
Viewports

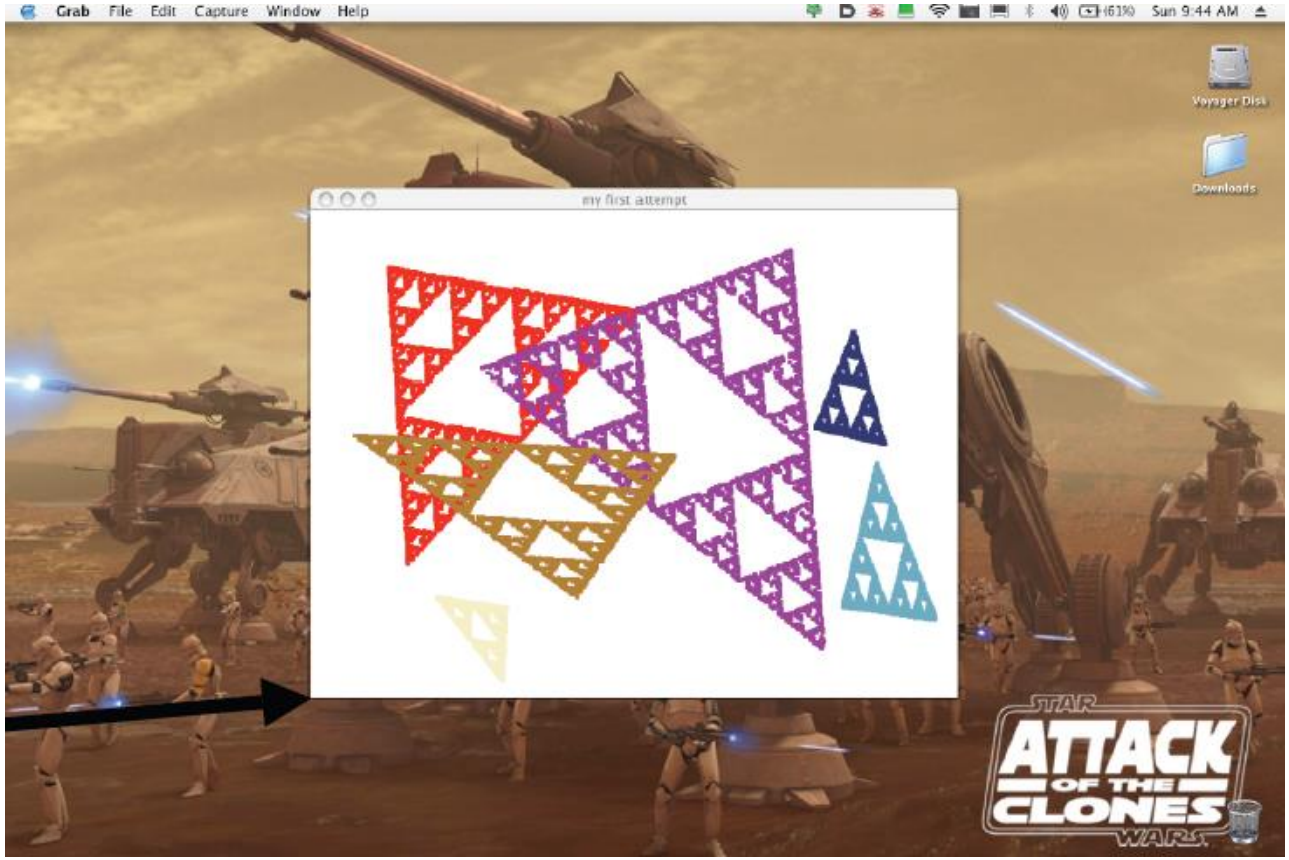
Joshua Cuneo



Screen Coordinate System



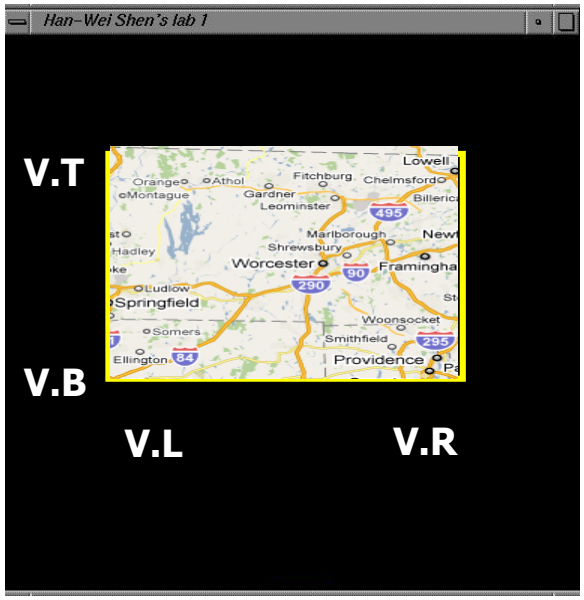
Screen Coordinate System

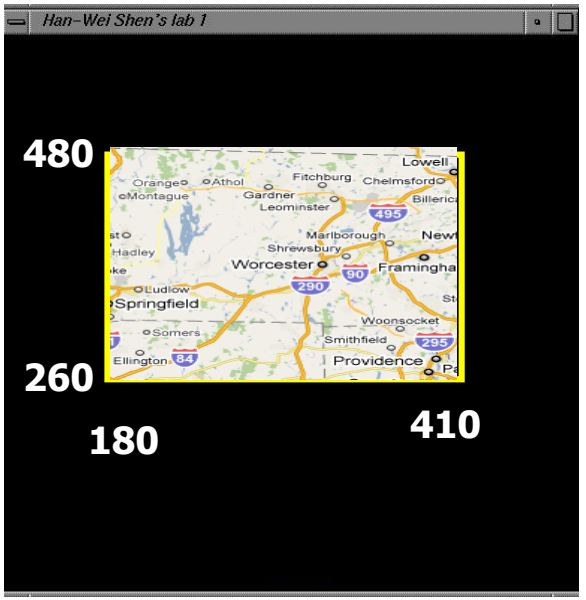


WebGL's (0,0)

Defining a Viewport

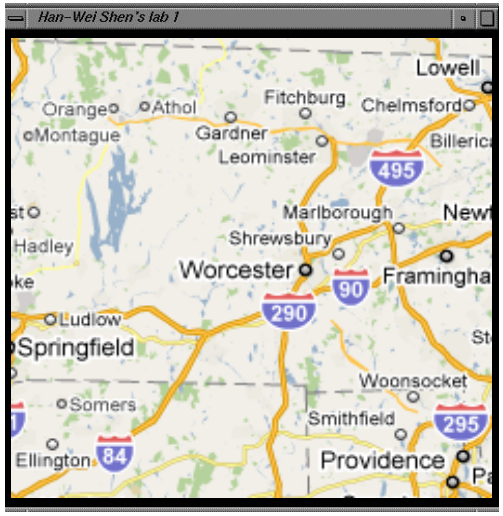
```
glViewport(left, bottom, width, height)
or glViewport(V.L, V.B, V.R – V.L, V.T – V.B)
e.g. glViewport(180, 260, (410 – 180), (480 – 260) )
```





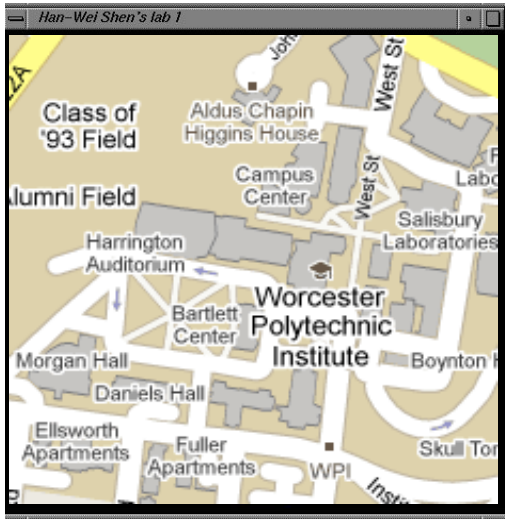
Note: Set desired viewport, then draw

World Coordinate System



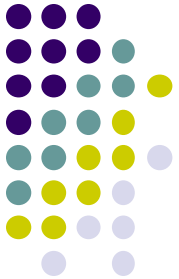
100 pixels = 30 miles

Change
World window
(mapping)



100 pixels = 0.25 miles

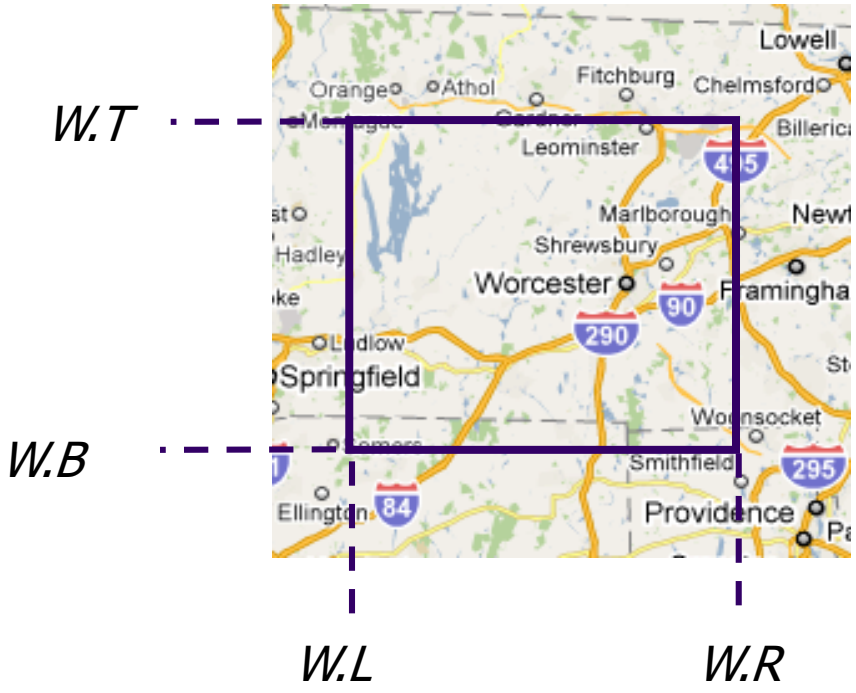
Using Window Coordinates



- Programming steps:
  1. Define world window (original drawing extents)
  2. Define viewport (drawing extents on screen)
  3. Map drawings within window to viewport

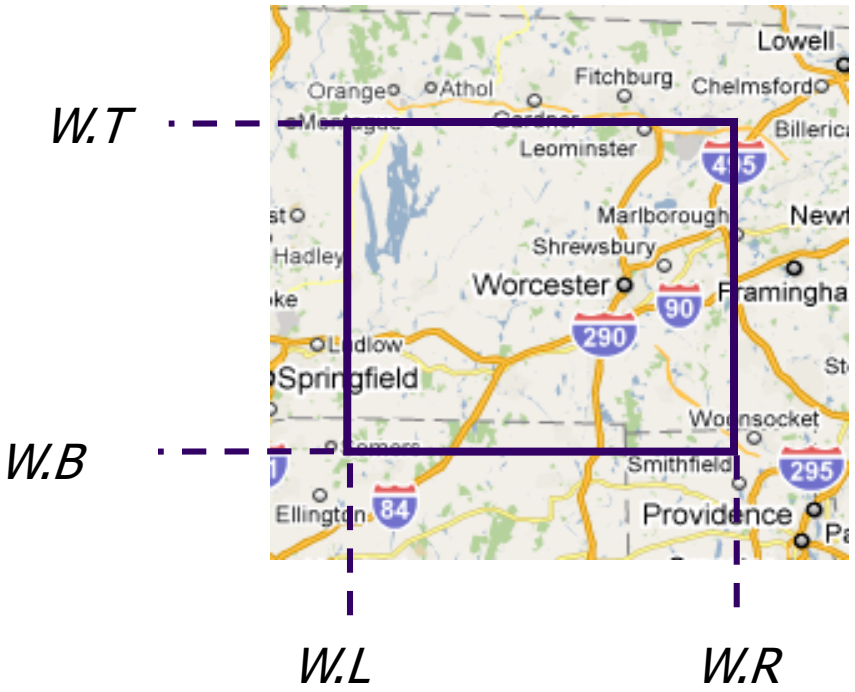
World Coordinate System

- **World Window:** region of **source** drawing to be rendered



Defining World Window

```
var projMatrix = ortho(left, right, bottom, top, -1.0, 1.0)
OR
var projMatrix = ortho(W.L, W.R, W.B, W.T, -1.0, 1.0)
```



generates 4x4 matrix that scales input drawing

Apply ortho( ) matrix in Vertex Shader

```
var projMatrix = ortho(W.L, W.R, W.B, W.T, -1.0, 1.0)
```

```
uniform mat4 Proj;
in vec4 vPosition;

void main( ){
    gl_Position = Proj * vPosition;
}
```

In vertex shader, multiply each vertex with **proj** matrix

Drawing Polyline Files