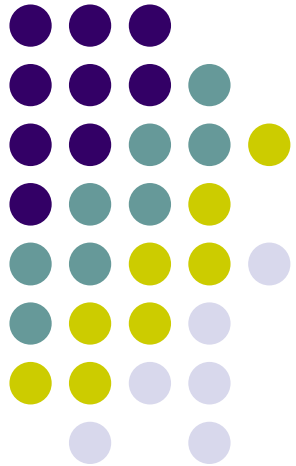
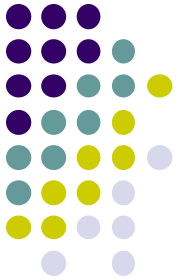
Computer Graphics (4731)
WebGL and More 2D Graphics Systems

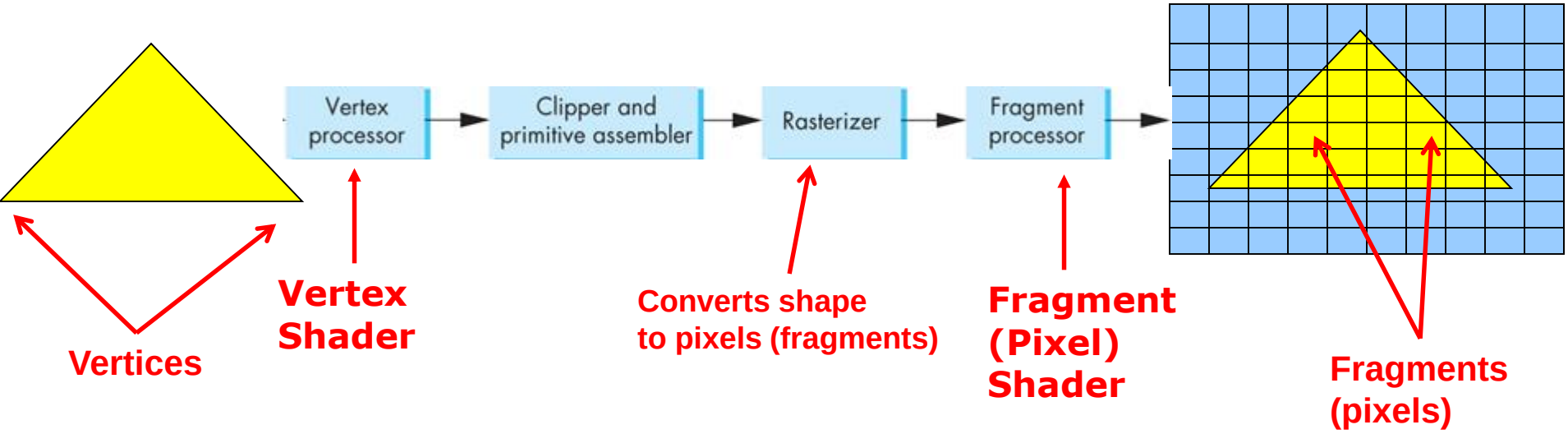
Joshua Cuneo



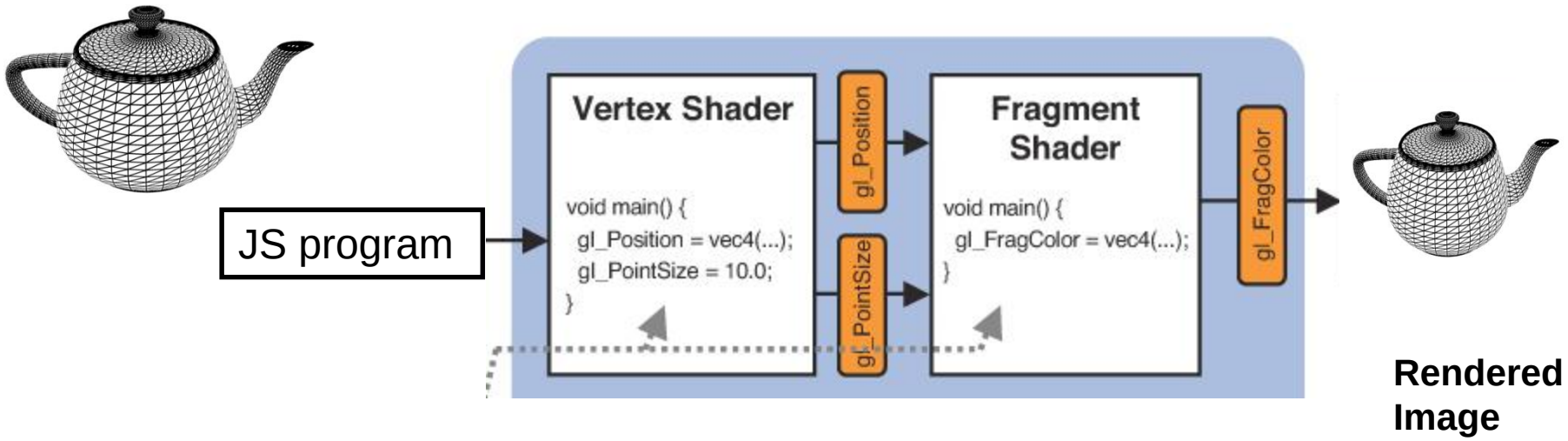
Graphics Pipeline

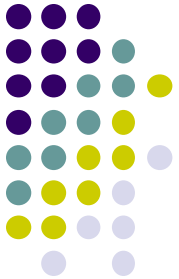


- **Vertex shader** code manipulates vertices of shapes
- **Fragment shader** code manipulates pixels



Graphics Pipeline



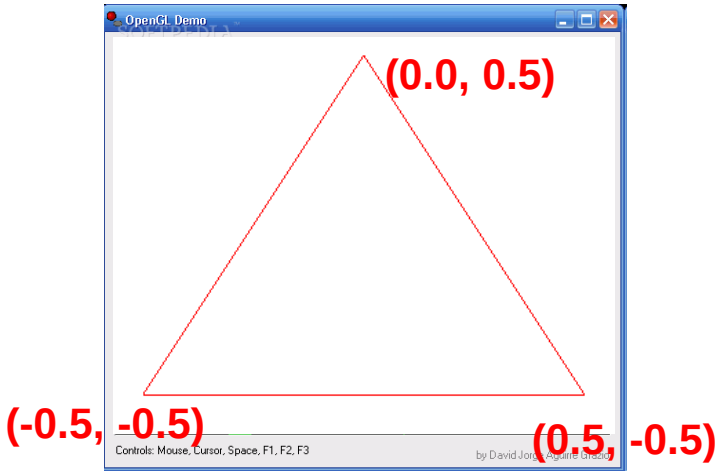


1. Generate triangle corners (3 vertices)

```
// declare array
var points = [];

// generate 3 triangle vertices + store in points array
points.push(vec2(-0.5, -0.5 ));
points.push(vec2( 0.0, 0.5 ));
points.push(vec2( 0.5, -0.5 ));
```

x *y*

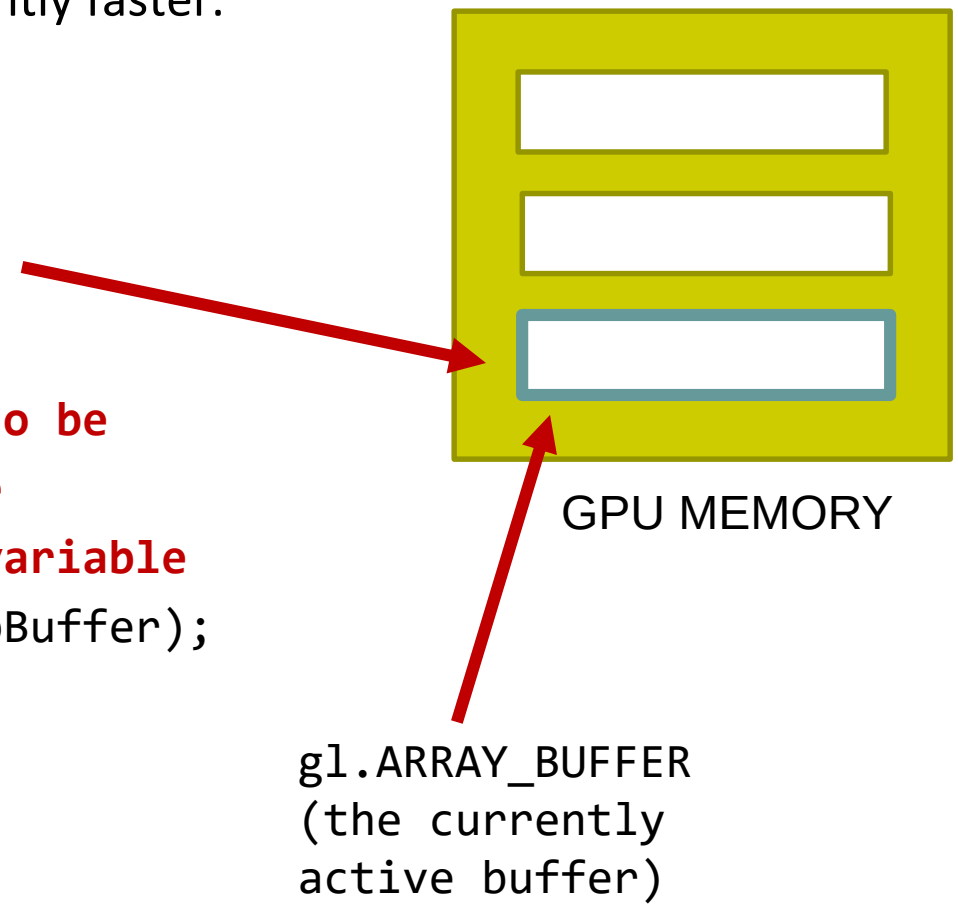


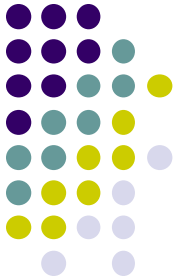
2. Create GPU Buffer for Vertices

Rendering from GPU memory significantly faster.
Move data there.

```
//Create a buffer in the GPU
var pBuffer = gl.createBuffer();

//Set the buffer as the buffer to be
//worked on, i.e. bind it to the
//ARRAY_BUFFER global internal variable
gl.bindBuffer(gl.ARRAY_BUFFER, pBuffer);
```





3. Move points into GPU memory

```
//Copy our data into the buffer
gl.bufferData(gl.ARRAY_BUFFER, flatten(points), gl.STATIC_DRAW);
```

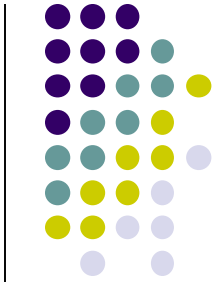
Data to be transferred to GPU memory (generated earlier)

GPU MEMORY

gl.ARRAY_BUFFER

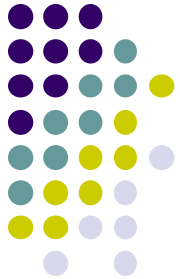
(0.0, 1.0, 0.5, 1.0, 0.0, …)

4. Draw points (from VBO)

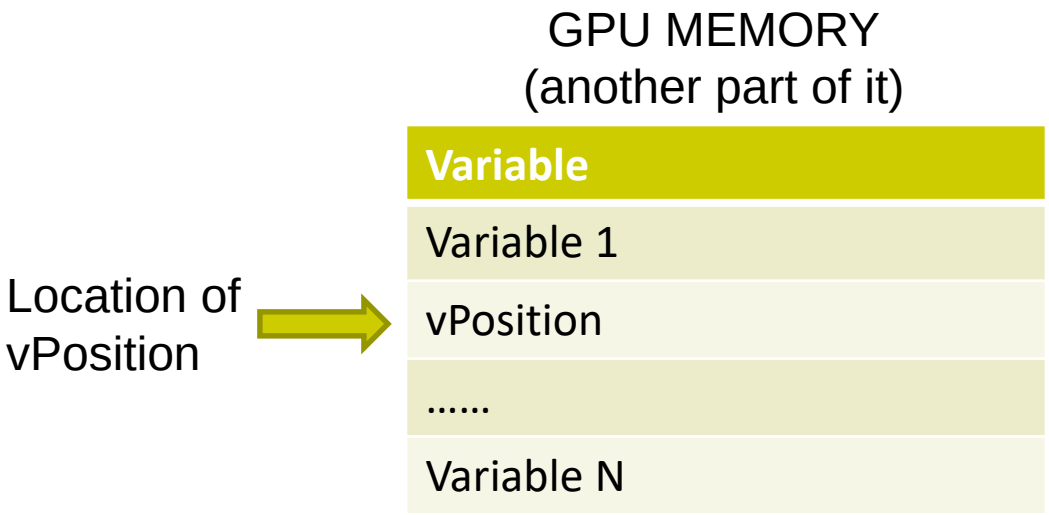


```
//Enable vertex array attribute at location
//of vPosition (essentially turns the attribute on)
gl.enableVertexAttribArray(vPosition);
```

4. Draw points (from VBO)



```
//Ask WebGL for the memory location of the
//"vPosition" attribute
var vPosition = gl.getAttribLocation(program,  "vPosition");
```



4. Draw points (from buffer)

```
//Get the data from the buffer that is currently bound
//to ARRAY_BUFFER
gl.vertexAttribPointer(vPosition, 4, gl.FLOAT, false, 0, 0);
```

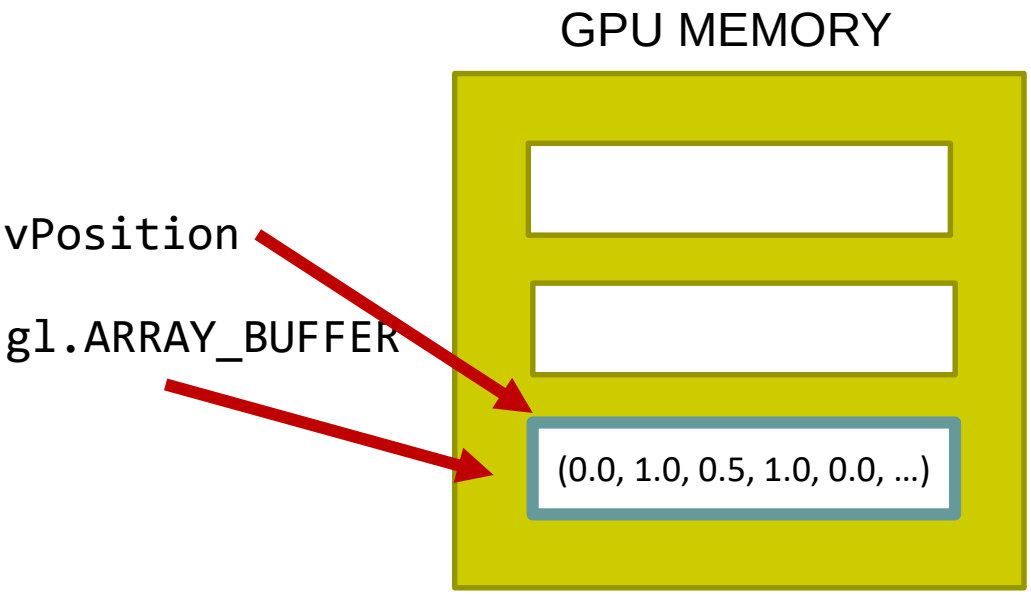
Location of **vPosition** in table of variables

4 (x,y) floats per vertex

Data not normalized to 0-1 range

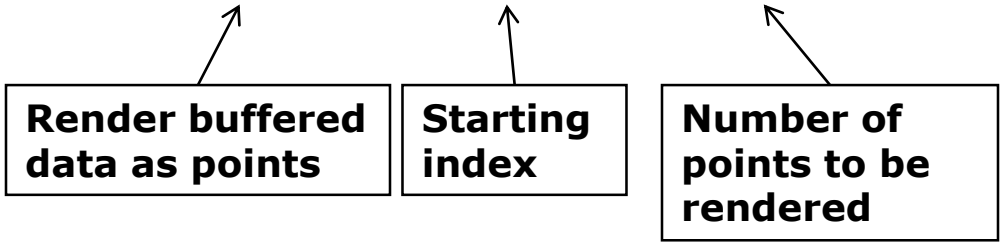
Padding between Consecutive vertices

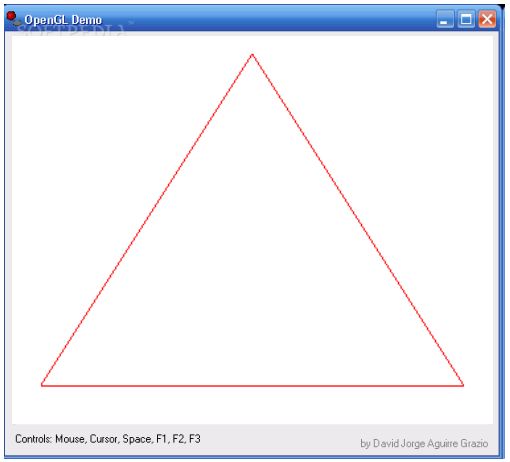
Data starts at offset from start of array



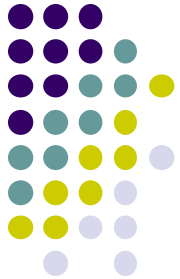
4. Draw points (from buffer)

```
//Draw the points
gl.drawArrays(gl.POINTS, 0, points.length);
```

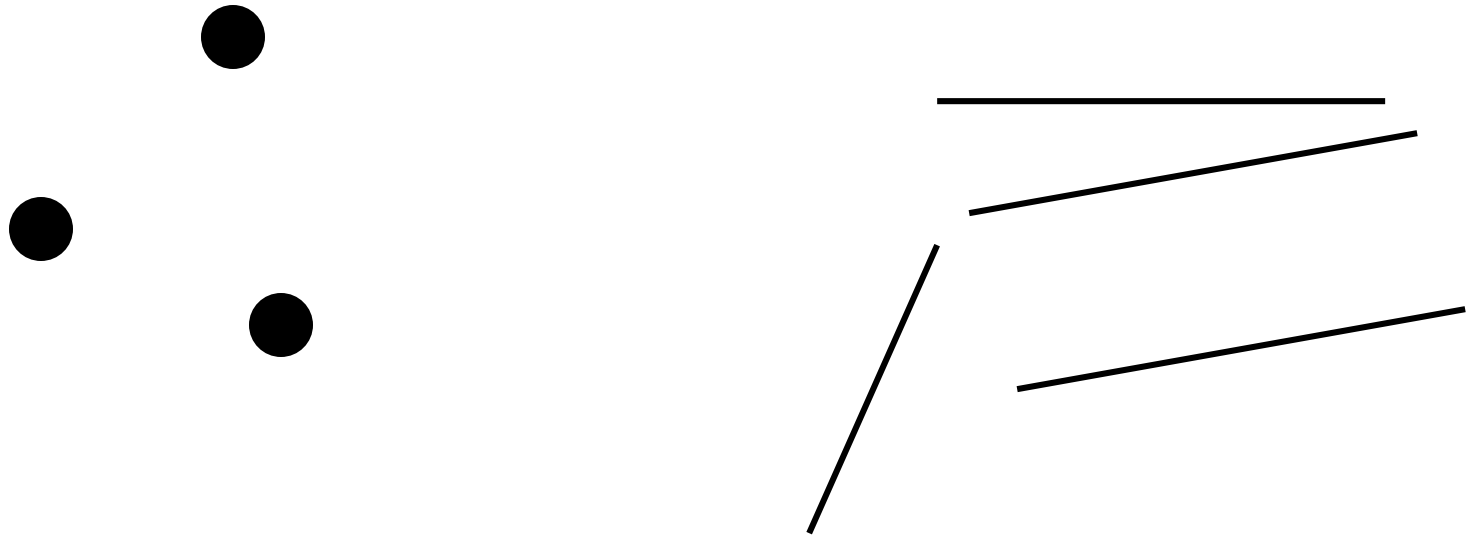




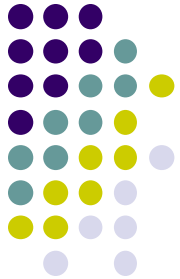
Other possible arguments to `glDrawArrays` instead of `GL_LINE_LOOP`?



`gl.drawArrays`(gl.POINTS, ….) **`gl.drawArrays`**((gl.LINES, …)

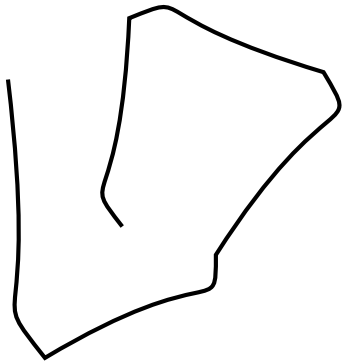


`glDrawArrays( )` Parameters



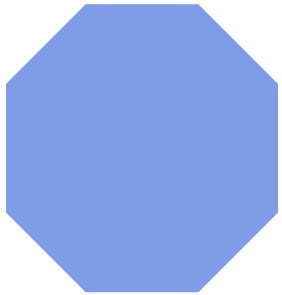
`gl.drawArrays`(gl.LINE_STRIP,..)

– polylines



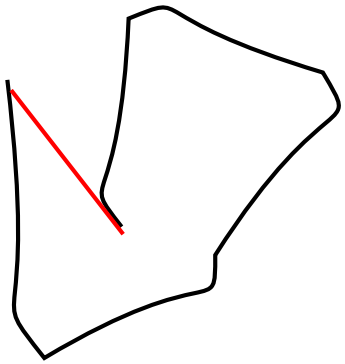
`gl.drawArrays`(gl.POLYGON,..)

– convex filled polygon



`gl.drawArrays`(gl.LINE_LOOP)

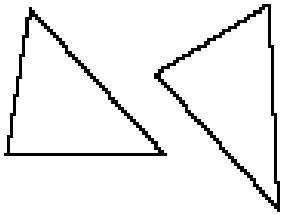
– Close loop of polylines
(Like gl.LINE_STRIP but closed)



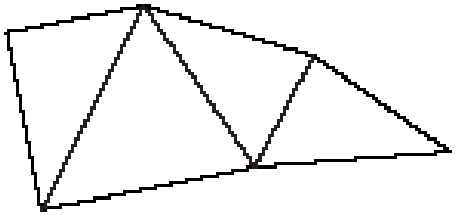
`glDrawArrays( )` Parameters

- Triangles: Connect 3 vertices

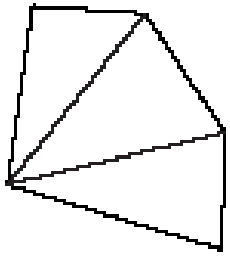
gl.TRIANGLES

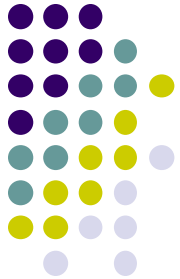


gl.TRIANGLE_STRIP



gl.TRIANGLE_FAN

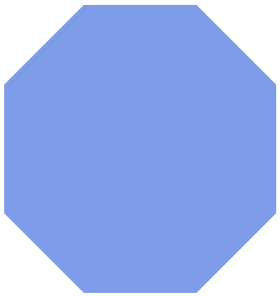


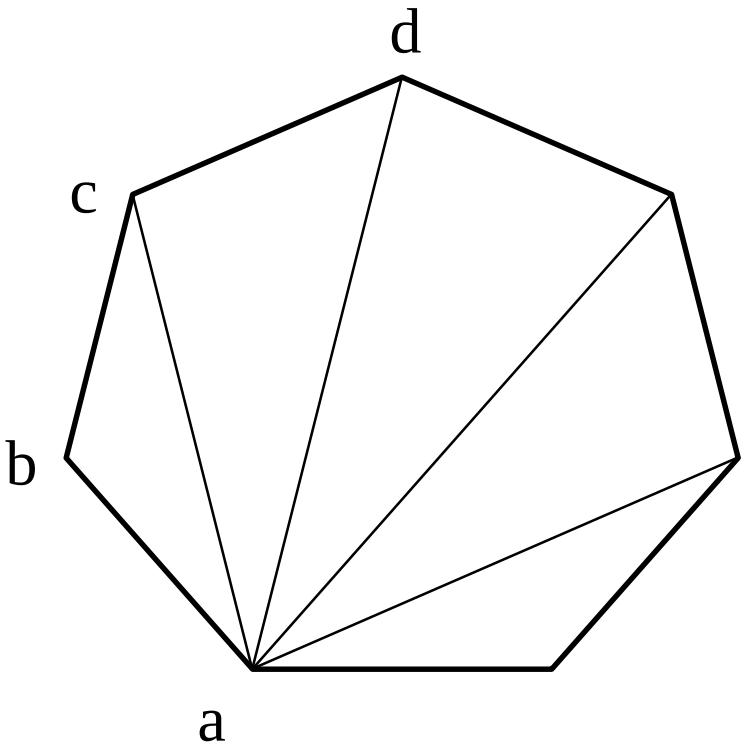


Triangulation

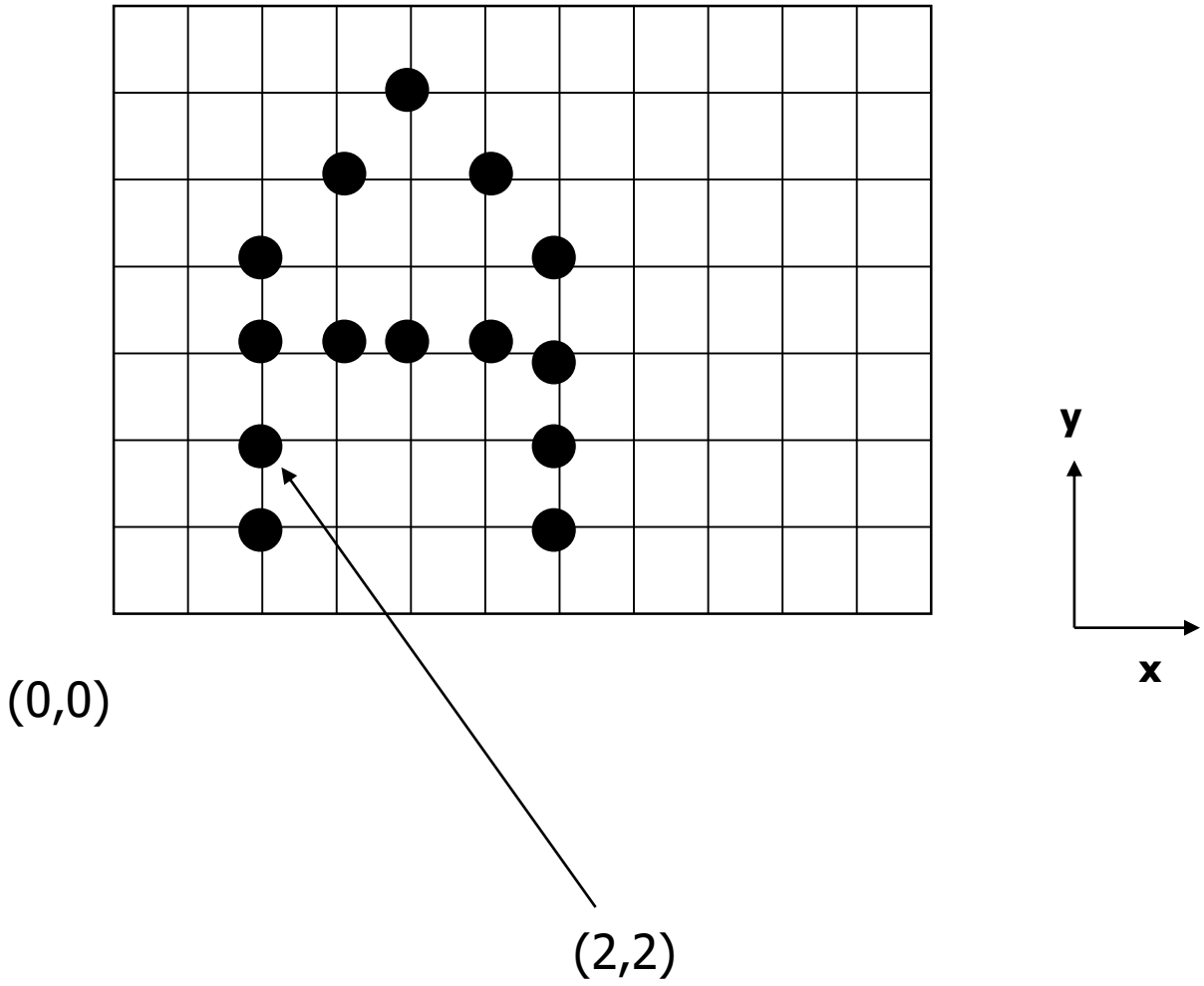
- Generally WebGL breaks polygons down into triangles which are then rendered.

`gl.drawArrays`(gl.POLYGON,..)

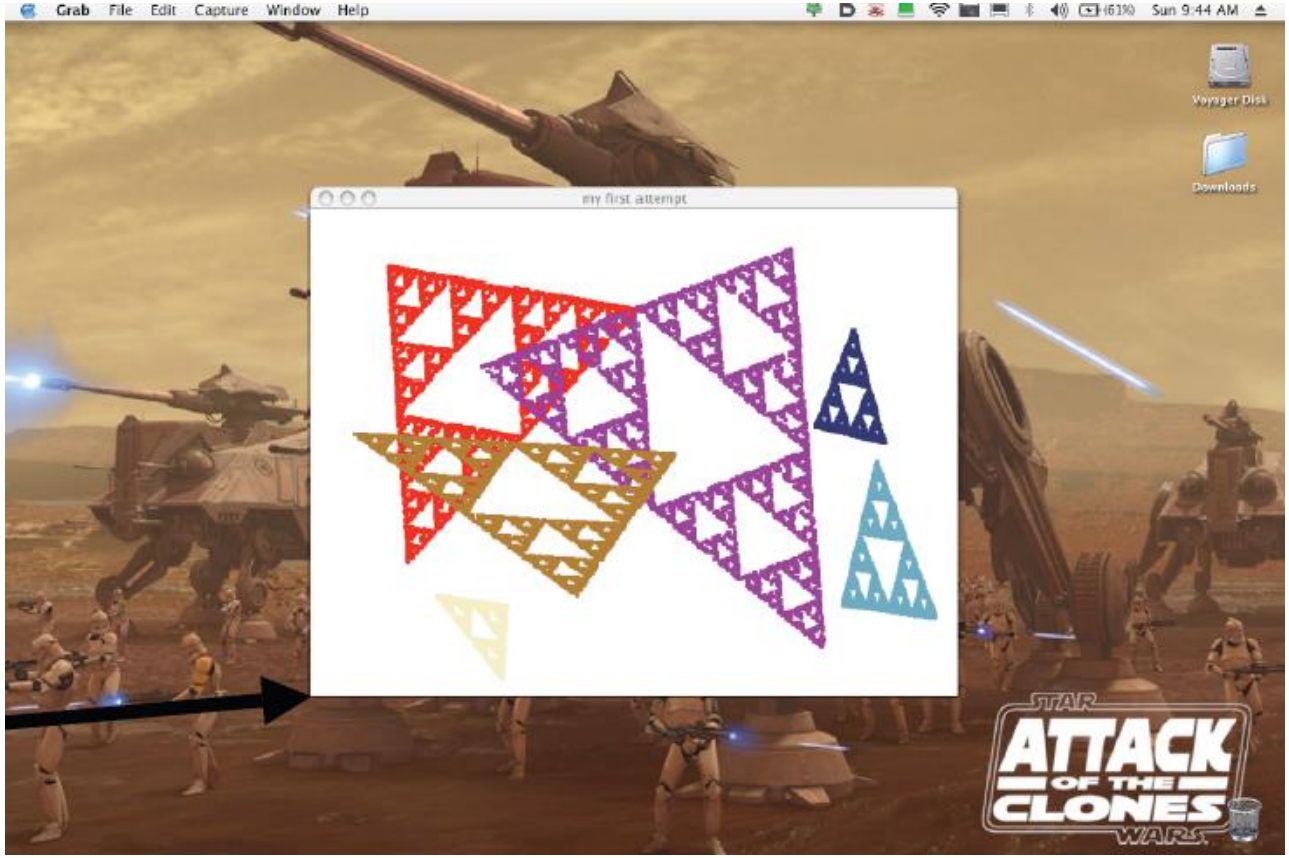




Screen Coordinate System



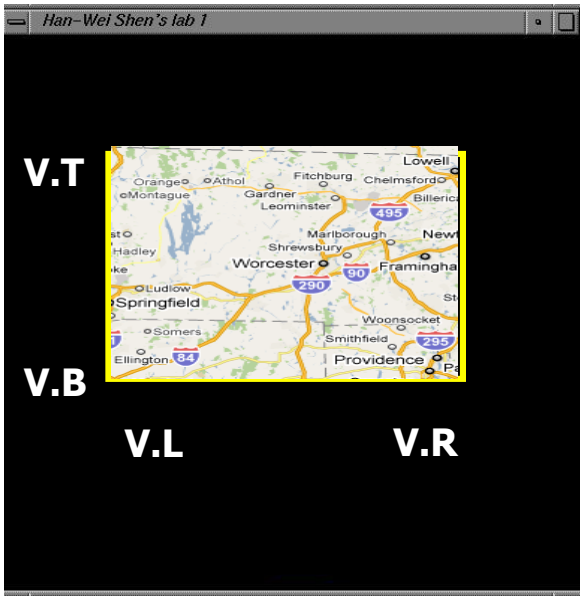
Screen Coordinate System

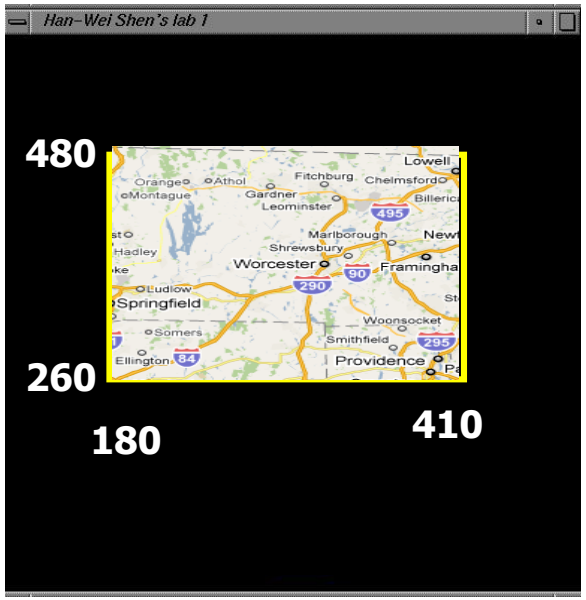


WebGL's (0,0)

Defining a Viewport

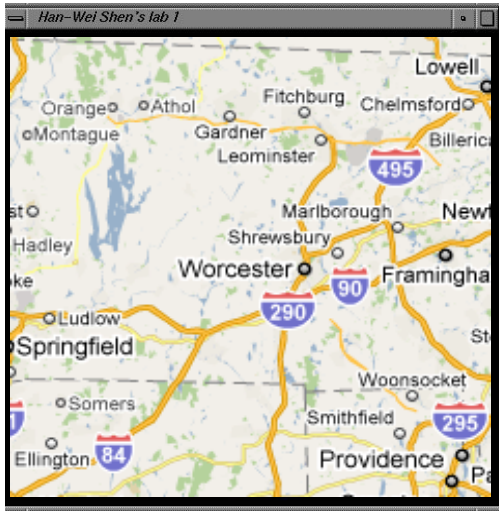
```
glViewport(left, bottom, width, height)
or glViewport(V.L, V.B, V.R – V.L, V.T – V.B)
e.g. glViewport(180, 260, (410 – 180), (480 – 260) )
```





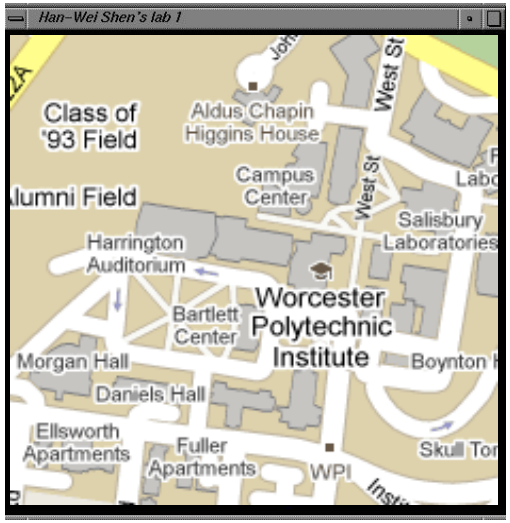
Note: Set desired viewport, then draw

World Coordinate System



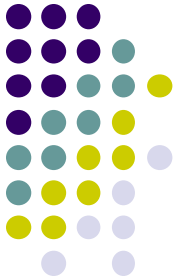
100 pixels = 30 miles

Change
World window
(mapping)



100 pixels = 0.25 miles

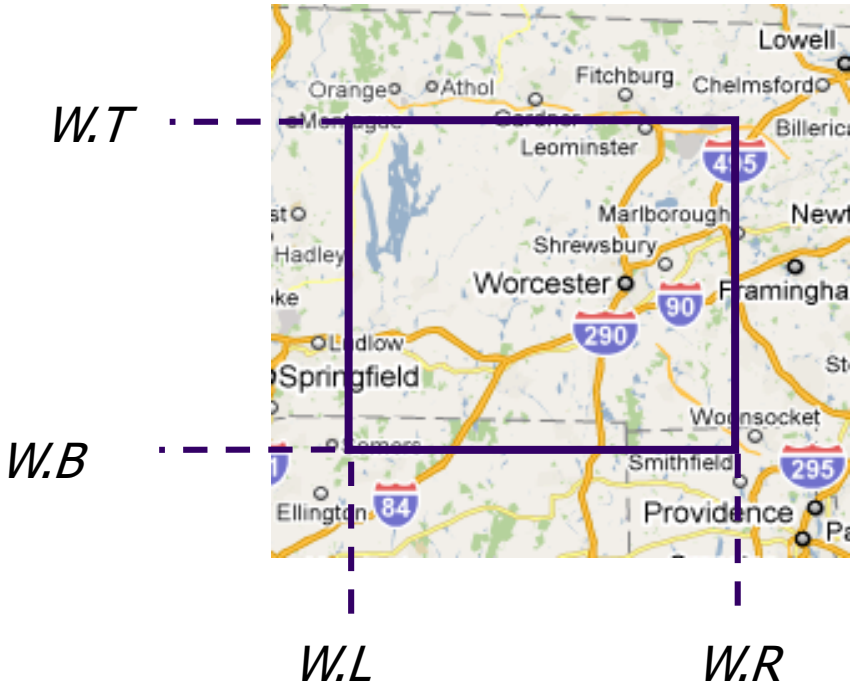
Using Window Coordinates



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

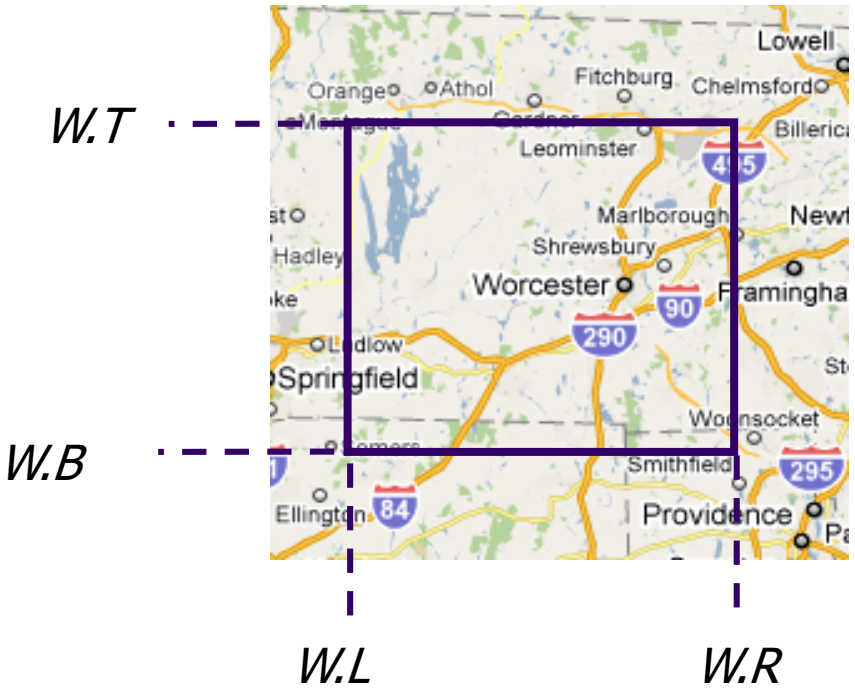
World Coordinate System

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



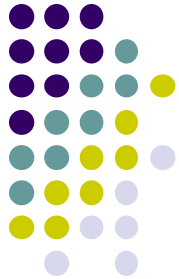
Defining World Window

```
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