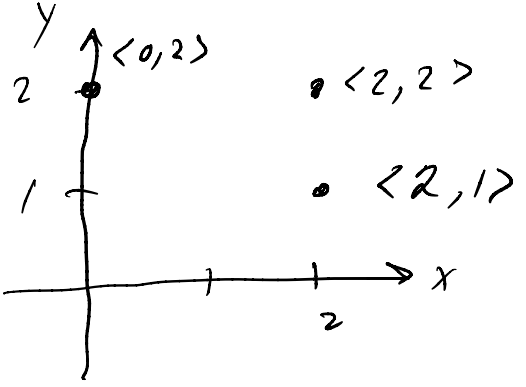
Drawing points, lines, triangles

Points - Display - consists of pixels.

Each pixel is a point / dot in the image.

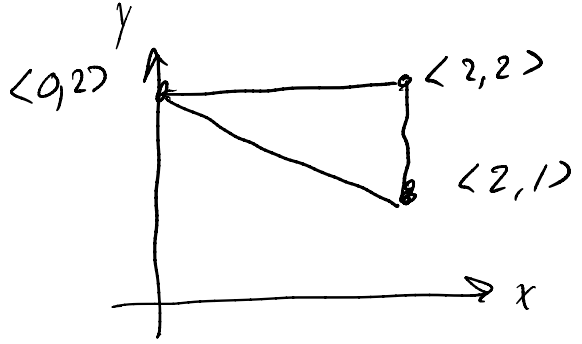


Convention

$\langle 2,1 \rangle = \begin{pmatrix} 2 \\ 1 \end{pmatrix}$

- a column vector.

Lines (Vector Graphics)



3 lines

In Javascript canvas:

penup / pendown / move

```
moveto (2,1);
lineto (2,2);
lineto (0,2);
lineto (2,1);
```

Early oscilloscopes

(Not really pixel-based)

In OpenGL:

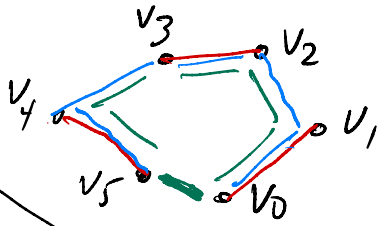
```
float verts[3][2] = { {2,1}, {2,2}, {0,2} };
glDrawArrays(GL_POINTS, 0, 3);
```

(0 ← starting point; 3 ↑ # of point)

In code:
(1) Definition of verts[2] array
(2) Load the verts array into the GPU buffers
VBO - Vertex Buffer Object, will hold a copy of verts
VAO - Vertex Array Object - holds information about what is in the VBO

---

glDrawArrays(GL_LINE_LOOP, 0, 3); ← Draws 3 lines
glDrawArrays(GL_LINE_STRIP, 0, 3) ← Draws 2 lines
" (GL_LINES, 0, ~~3~~ 2); ← Draws 1 line



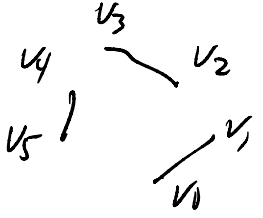
Suppose vertices are $V_0, V_1, V_2, V_3, V_4, V_5$.

GL_LINES

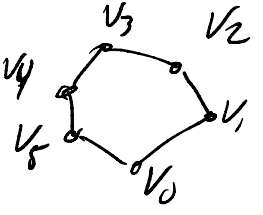
GL_LINE_STRIP

GL_LINE_LOOP

GL_LINES

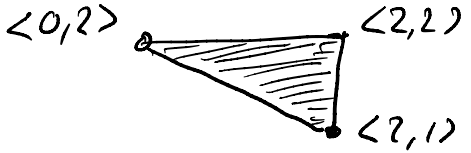


glDrawArrays(GL_LINE_LOOP, 0, 6)



glDrawArrays (GL_TRIANGLES, 0, 3);

(with verts[][] = { {2,1}, {2,2}, {0,2} })



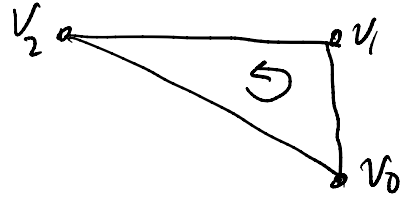
Each vertex: Processed by a vertex shader on the GPU

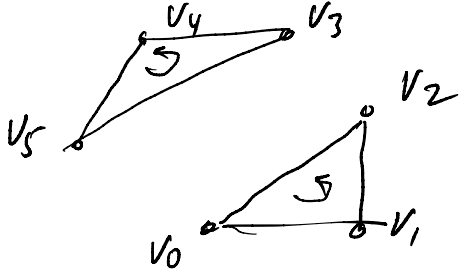
Each pixel in the triangle: - Processed by a fragment shader.

Triangles have Front Faces & Back Faces.

CCW rule (default)

"counterclockwise"



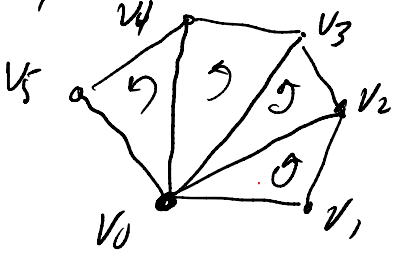


6 vertices
$v_0 \dots v_5$
in order

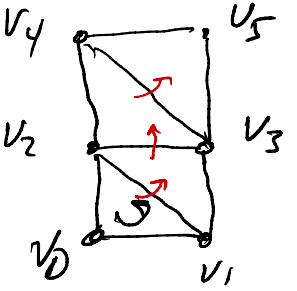
glDrawArrays (GL_TRIANGLES, 0, 6) – draw 2 triangles

---

glDrawArrays (GL_TRIANGLE_FAN, 0, 6)



glDrawArrays (GL_TRIANGLE_STRIP, 0, 6)



CCW rule applies to the 1ST one

(All four have front face pictured)

Last part of lecture was on the blackboard & should be visible on the podcast.