Points - Display consists of pixels.

Each pixel is a point/dot in the image.

Convention:

\[ <2, 1> = \begin{pmatrix} 2 \\ 1 \end{pmatrix} \]

-a column vector.
Lines (Vector Graphics)

\begin{align*}
&\langle 0,2 \rangle \quad \langle 2,2 \rangle \\
&\langle 2,1 \rangle
\end{align*}

In Javascript canvas:

- `moveTo(2,1);`
- `lineTo(2,2);`
- `lineTo(0,2);`
- `lineTo(2,1);`

`penup / pendown / move`

Early oscilloscopes

(Not really pixel-based)
In Open GL:

```c
float verts[3][2] = { { 2.1 }, { 2.2 }, { 0.2 } },
                   { 1 };
```

- Starting point

```c
glDrawArrays (GL_POINTS, 0, 3);
```

# of point

In code:

1) Definition of verts[3][2] array

2) Load the verts array into the GPU buffers

```c
VBO - Vertex Buffer Object,
      will hold a copy of verts

VAO - Vertex Array Object - holds information about what we use
      the VBO
```

```c
// Draws 3

// Draws 2

// Draws 1 line
```
Suppose vertices are $V_0, V_1, V_2, V_3, V_4, V_5$.

```gl
DrawArrays(GL_LINE_LOOP, 0, 6)
```
Draw Arrays (GL-TRIANGLES, 0, 3):

\[
\text{with } \text{verts}[7] = \{ [2,1], [2,2], [0,2] \}
\]

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): V0 → V1 → V2

"counterclockwise"
6 vertices
V₀ ... V₅
in order

`glDrawArrays(GL_TRIANGLES, 0, 6)` — draw 2 triangles

`glDrawArrays(GL_TRIANGLE_FAN, 0, 6)`
DrawArrays (GL_TRIANGLES.Strip, 0, 6)

CCW rule applied to the 1st one

(All four have front face pictured)

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