OpenGL - Common Usage

CPU

C^{++} programs

Specify vertices in $\mathbb{R}^3$
+ vertex attributes
- uploaded to GPU

Give matrices that map points from $\mathbb{R}^3$ to screen coordinates
(4x4 matrices!)

Issue draw commands

Specify how vertices from triangles
(or lines/points)

$\Longrightarrow$

GPU Graphics Processing Unit

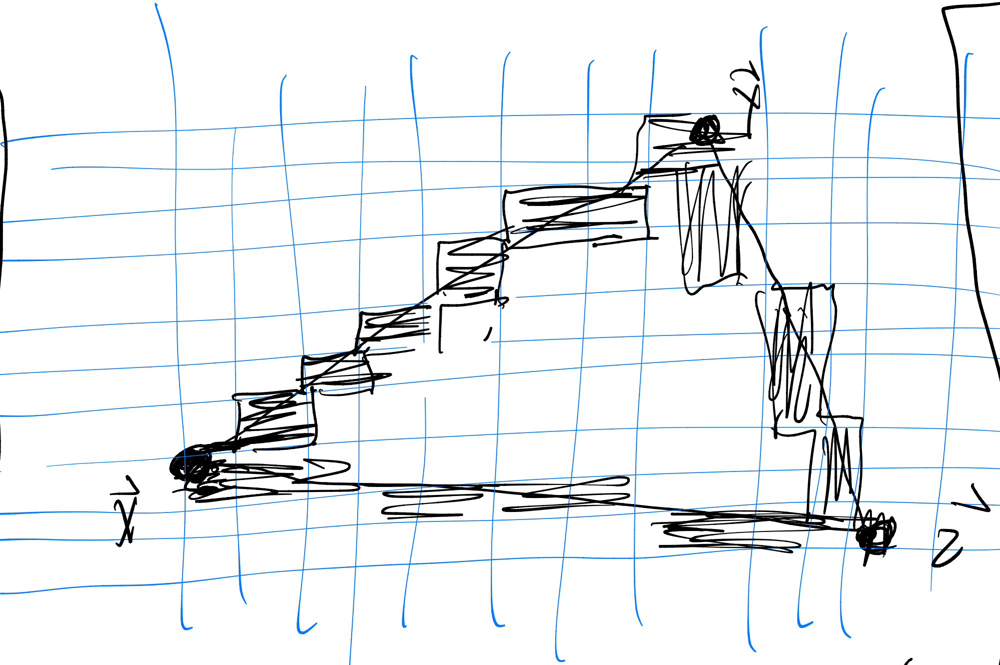
Vertex Shader

Small program - operates on each vertex.
- Compute screen coordinates
- Send on (or update) vertex attributes

Fragment Shader

- Small program that operates on each pixel.
- Using the vertex attributes from the vertex shader
- usually averaged (smoothed or shaded) across the triangle.
- Also accesses uniform values.

Averaging a Shading has to take perspective into account



Each pixel has a depth value or z-value.

A measure of distance from the viewer.

Triangle on a screen (rectangular array of pixels)

Aliasing - Any problem or effect caused by conversion between analogue & digital or between different levels of precision.

Hidden surface algorithms — based on depth values

Each pixel stores a depth value.

When a pixel is about to overwritten

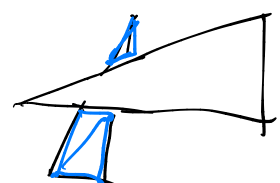
keep the pixel value with depth minimized

(for triangle closest to the viewer.

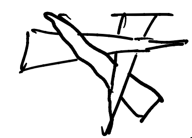
- Painter's Algorithm for hidden surfaces

  - Render more distant triangles first, overwriting previous triangles as we go.

- Geometric analysis



- Culling of back faces - helps with hidden surface.

	Advantages	Disadvantages
Depth Buffer	• Elegant • Fast, fits OpenGL frameworks, Very parallizable • Render in any order	• Render some unseen objects • Extra memory per pixel 32 bits $z \in [0, 2^{32}-1]$ • Problems with precision or round off errors • There are minimum + maximum distances
Painter's Algorithm	• Elegant • Good for transparency.	• Render unseen things • Sorting ahead Gather all triangles ahead of time. • No consistent order 
Geometric Analysis	• Only seen (visible) things are rendered.	• Need sophisticated algorithms (spacial data structures)

Linear Transformations

Lets work in $\mathbb{R}^2$. (2-space)

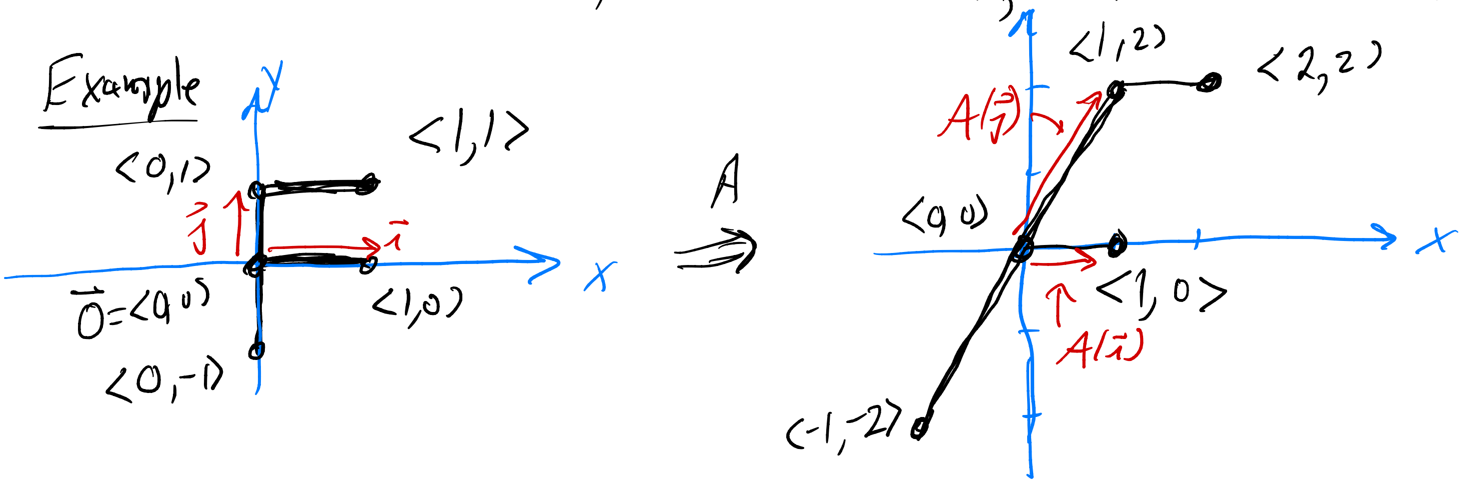
Points $\vec{x} = \langle x_1, x_2 \rangle = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$

<u>Def'n</u> A mapping $A : \mathbb{R}^2 \to \mathbb{R}^2$ is <u>linear</u> if

① For all $\vec{x}, \vec{y} \in \mathbb{R}^2$, $A(\vec{x}+\vec{y}) = A(\vec{x}) + A(\vec{y})$

② For all $\vec{x} \in \mathbb{R}^2$, all scalars $\alpha \in \mathbb{R}$, $A(\alpha \vec{x}) = \alpha A(\vec{x})$.

<u>Example</u>



Represent by a ~~matrix~~ formula:

$$A(\langle x, y\rangle) = \langle x+y, 2y\rangle$$

or by a matrix

$$A(\langle x, y\rangle) = A\left(\begin{pmatrix} x \\ y \end{pmatrix}\right) = M\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 0 & 2 \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x+y \\ 2y \end{pmatrix}$$

where $M = \begin{pmatrix} 1 & 1 \\ 0 & 2 \end{pmatrix}$.

$$\boxed{M = \begin{array}{cc} & A(\vec{j}) \\ \multicolumn{2}{c}{\begin{pmatrix} 1 & 1 \\ 0 & 2 \end{pmatrix}} \\ A(\vec{i}) & \end{array}}$$

$$\vec{i} = \langle 1, 0 \rangle \qquad \vec{j} = \langle 0, 1 \rangle$$

Let $\vec{u} = A(\vec{i}) = A(\langle 1, 0 \rangle) = \langle u_1, u_2 \rangle \in \mathbb{R}$

Let $\vec{v} = A(\vec{j}) = A(\langle 0, 1 \rangle) = \langle v_1, v_2 \rangle$

Let $M = \begin{pmatrix} u_1 & v_1 \\ u_2 & v_2 \end{pmatrix} = \begin{pmatrix} \vec{u} & \vec{v} \end{pmatrix}$

<u>Then</u> $M\begin{pmatrix} x \\ y \end{pmatrix} = A(\langle x, y \rangle)$ for all $\langle x, y \rangle \in \mathbb{R}^2$.

<u>Pf</u> $A(\langle x, y \rangle) = A(x\vec{i} + y\vec{j}) = xA(\vec{i}) + y \cdot A(\vec{j})$ by linearity

$= x\vec{u} + y\vec{v} = x \cdot \langle u_1, u_2 \rangle + y \cdot \langle v_1, v_2 \rangle$

$= \langle xu_1 + yv_1, xu_2 + yv_2 \rangle = \begin{pmatrix} xu_1 + yv_1 \\ xu_2 + yv_2 \end{pmatrix} = \begin{pmatrix} u_1 & v_1 \\ u_2 & v_2 \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix}$

qed.