OpenGL - Common Usage

CPU

C++ programs

Specify vertices in \( \mathbb{R}^3 \)

+ vertex attributes
  - uploaded to GPU

Give matrices that map point from \( \mathbb{R}^3 \)

- Screen coordinates
  (4x4 matrices!)

Issue draw commands

Specify how vertices from triangle
  (or lines/points)

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
Each pixel has a depth value or $z$-value. A measure of distance from the viewer.

Averaging or shading has to take perspective into account.

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

Triangle on a screen (rectangular array of pixels)
Hidden surface algorithms - based on depth values

Each pixel stores a depth value.
When a pixel is about to be overwritten, keep the pixel value with depth minimized (far triangle closest to the viewer).

**Painter's Algorithm for hidden surfaces**

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

- Geometric analysis

- Culling of back faces - helps with hidden surface.
Depth Buffer

Painters Algorithm

Geometric Analysis

Advantages
- Elegant
- Fast, fits OpenGL frameworks
- Very parallelizable
- Render in any order
- Elegant
- Good for transparency

Disadvantages
- Render some unseen objects
- Extra memory per pixel 32 bits \( \leq 2^{32} - 1 \)
- Problems with precision or round off errors
- There are minimum and maximum data values
- Render unseen things
- Surprisingly ahead
- Gather all triangles ahead of time
- No consistent order

Only seen (visible) things are rendered

Need sophisticated algorithms (spacial data structures)
**Linear Transformations**

Let's work in $\mathbb{R}^2$. (2-space)

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

**Definition** A mapping $A : \mathbb{R}^2 \to \mathbb{R}^2$ is **linear** 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 $\lambda \in \mathbb{R}$, $A(\lambda \vec{x}) = \lambda A(\vec{x})$.

**Example**

![Diagram](image)

$A(\vec{v}) = \langle 2, 2 \rangle$
Represent by a matrix formula:

\[ A(<x,y>) = <x+y, 2y> \]

or by a matrix

\[ A(<x,y>) = A((x, y)) = M(x) = (\begin{pmatrix} 1 & 0 \\ 0 & 2 \end{pmatrix}) \begin{pmatrix} x \\ y \end{pmatrix} = (x+y, 2y) \]

where \( M = (\begin{pmatrix} 1 & 1 \\ 0 & 2 \end{pmatrix}) \).
\[ \mathbf{\hat{u}} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad \mathbf{\hat{j}} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \]

Let \( \mathbf{\hat{u}} = A(\mathbf{\hat{u}}) = A(\begin{pmatrix} 1 \\ 0 \end{pmatrix}) = \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} \in \mathbb{R} \)

Let \( \mathbf{\hat{v}} = A(\mathbf{\hat{v}}) = A(\begin{pmatrix} 0 \\ 1 \end{pmatrix}) = \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} \)

Let \( M = \begin{pmatrix} u_1 & v_1 \\ u_2 & v_2 \end{pmatrix} = (\mathbf{\hat{u}} \cdot \mathbf{\hat{v}}) \)

Then \( M \begin{pmatrix} x \\ y \end{pmatrix} = A(\begin{pmatrix} x \\ y \end{pmatrix}) \) for all \( \begin{pmatrix} x \\ y \end{pmatrix} \in \mathbb{R}^2 \).

\textbf{pf} \quad A(\begin{pmatrix} x \\ y \end{pmatrix}) = A(x \mathbf{\hat{u}} + y \mathbf{\hat{v}}) = xA(\mathbf{\hat{u}}) + yA(\mathbf{\hat{v}}) \) by linearity

\[ = x \mathbf{\hat{u}} + y \mathbf{\hat{v}} = x \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} + y \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} \]

\[ = \begin{pmatrix} xu_1 + yv_1 \\ xu_2 + yv_2 \end{pmatrix} = \begin{pmatrix} xu_1 + yv_1 \\ xu_2 + yv_2 \end{pmatrix} \]

\text{q.e.d.}