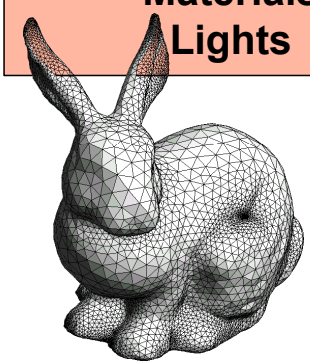
Rendering

Scene Description

3D objects
Coordinate Frame
Camera(s)
Materials
Lights

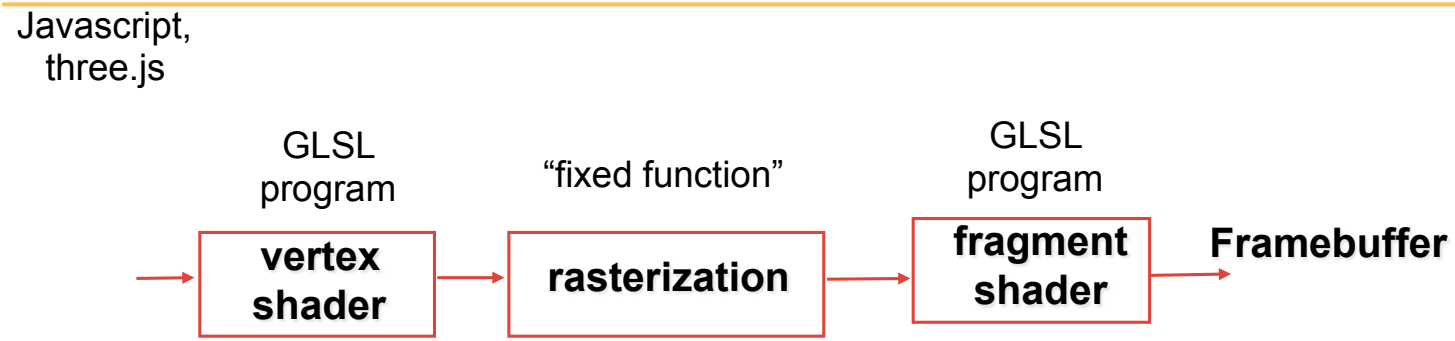


?

2D Image

OpenGL Rendering Pipeline

(with some details abstracted away)



Thus far…

- triangles
- vertices: $v=[x\ y\ z]^T$ local coords, on GPU
- vertex shader: $v' = M v$ to image coords
- fragment shader: colour = (1,0,0)
  colour = N•L
- instancing: redraw with $M_1, M_2, M_3, \ldots$
- many coordinate frames:
  e.g., wheel → car → world → camera → image

multiple instances in parallel
multiple shaders
"material": vertex + fragment shader.

Algorithm: "Projective Rendering"

Send all vertex geometry to GPU

```
for each frame
    clear screen
    for each object instance
        for each triangle j      // project onto image:
            transform vertices   // vertex shader
            for each pixel in j  // rasterization
                compute colour   // fragment shader
```

Linear Algebra Review

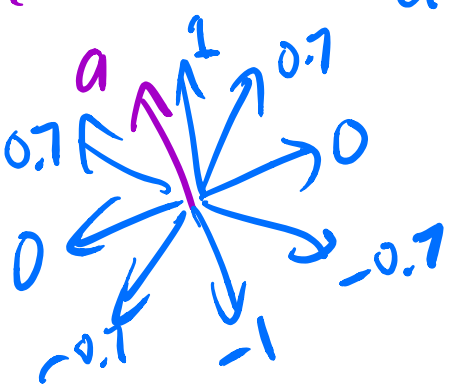
vectors

$\begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$ Column vectors by default.

$a^T = [a_1\ a_2\ a_3]$

dot product

$|a| = |b| = 1$



$$a \cdot b = a^T b = [a_1\ a_2\ a_3] \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

$$= a_1 b_1 + a_2 b_2 + a_3 b_3$$

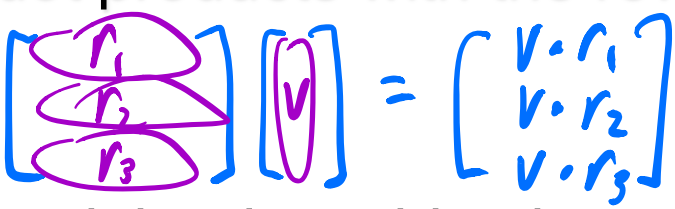
$$= |a||b| \cos(\theta)$$

$\Rightarrow$ maximized when aligned; 0 for $a \perp b$

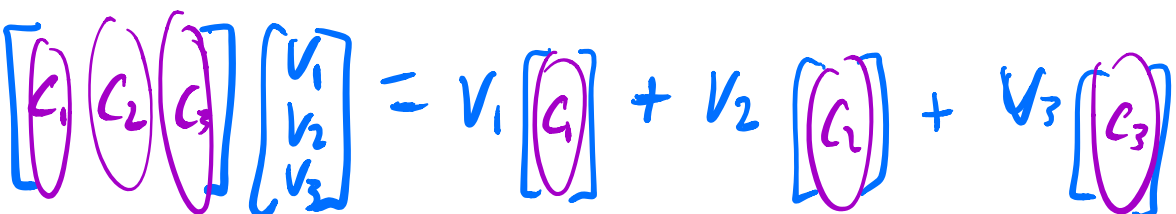
Math Review

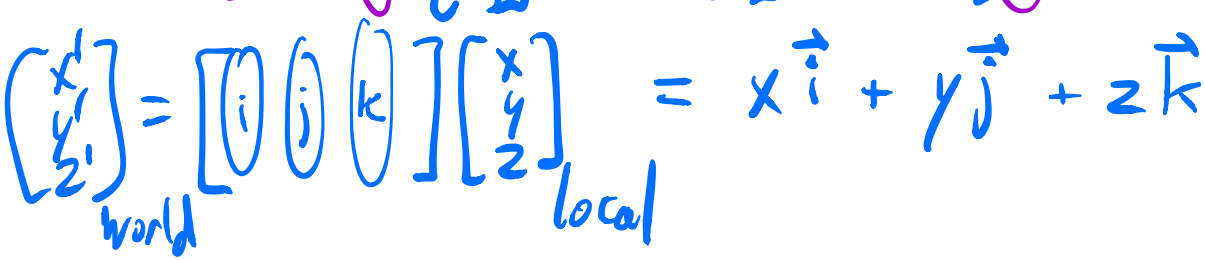
matrix-vector multiplication $v' = Mv$

(a) as dot products with the rows



(b) as weighted combinations of the columns

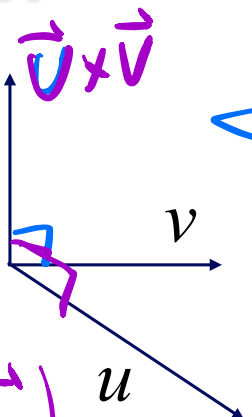




Math Review

Cross Product

LHCS
RHCS



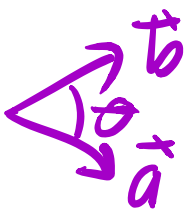
Right Handed Coordinate System

(curl fingers from u to v; thumb points to u x v)

$\vec{b}\times\vec{a} = -(\vec{a}\times\vec{b})$

$|a\times b| = |a||b|\sin\theta$

$\vec{a}\times\vec{a} = \vec{0}$

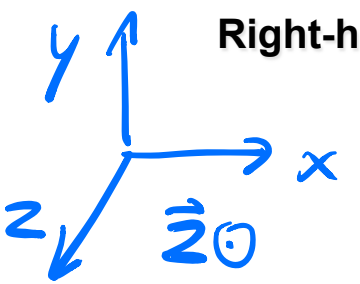


$$\vec{a}\times\vec{b} = \det\begin{bmatrix} i & j & k \\ a_x & a_y & a_z \\ b_x & b_y & b_z \end{bmatrix}$$

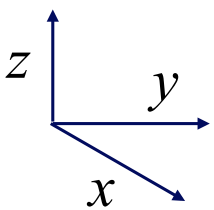
$$= \begin{bmatrix} a_y b_z - a_z b_y \\ a_z b_x - a_x b_z \\ a_x b_y - a_y b_x \end{bmatrix}$$

Math Review

Coordinate Systems



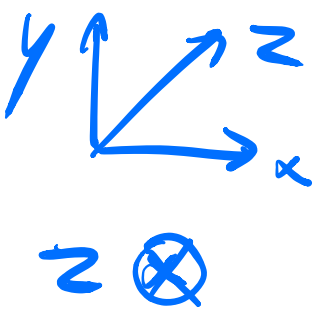
Right-handed Coordinate System

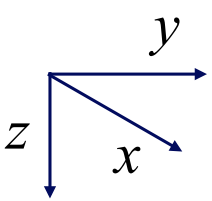


nearly always

using right-hand rule

Left-handed Coordinate System





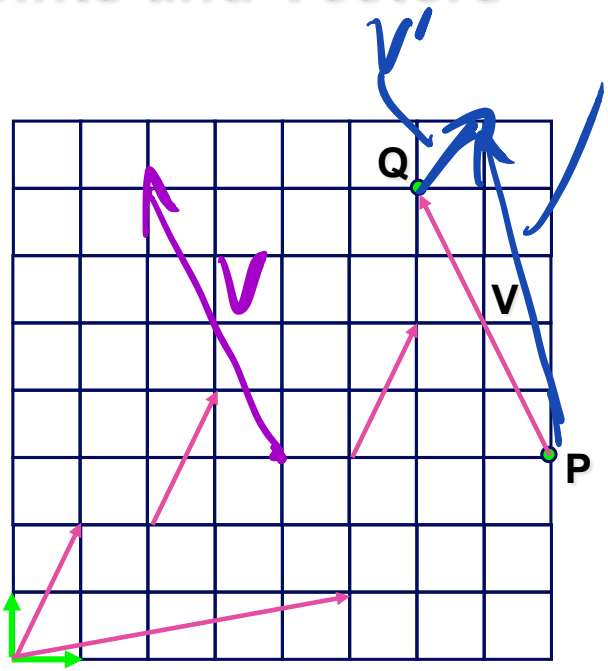
used for device coords (screen)

using left-hand rule

z will be the distance to objects

Math Review

Points and Vectors



vector space
vectors are invariant under translation

affine space:
allows vector-to-point addition

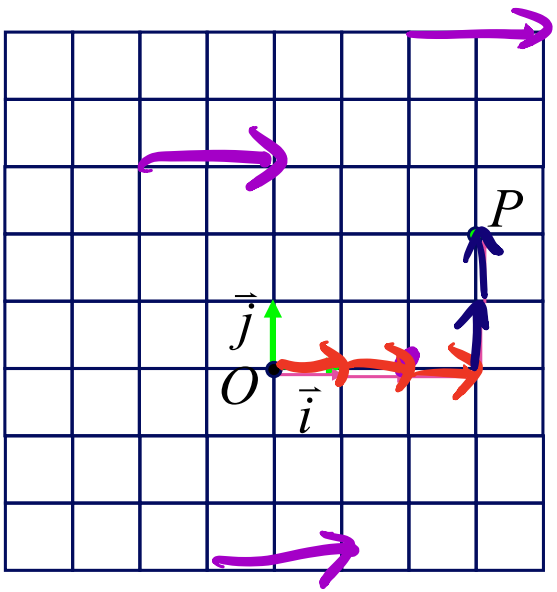
$Q - P = V$

$Q + P = ?$

$0.5Q + 0.3P = ?$

Math Review

Coordinate System vs Frame



coordinate system: basis vectors

frame: origin + basis vectors

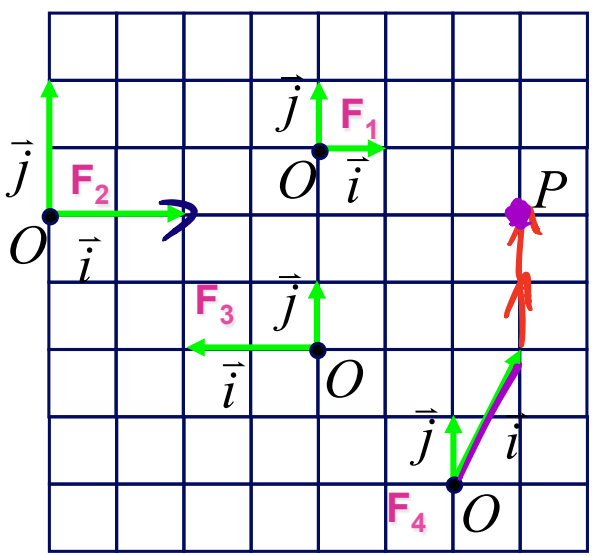
$$P = O + x\vec{i} + y\vec{j} + z\vec{k}$$

origin

3 2 0

Math Review

Working with Frames



start at the origin, then add x copies of $\vec{i}$, and add y copies of $\vec{j}$

$$P = O + x\vec{i} + y\vec{j}$$

F_1 $(3, -1)$

F_2 $(3.5, 0)$

F_3 $(-1.5, 2)$

F_4 $(1, 2)$

Many Coordinate Frames in a Scene

(and using transformation matrices to move between them)