Rendering

Scene Description

3D objects
Coordinate Frame
Camera(s)
Materials
Lights

?  2D Image
Algorithm: “Projective Rendering”

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
OpenGL Rendering Pipeline (with some details abstracted away)

Javascript, three.js

vertex shader

GLSL program

"fixed function"

rasterization

GLSL program

fragment shader

Framebuffer
Concepts

- geometry:
- vertices:
- vertex shader:
- fragment shader:
- instancing:
- many coordinate frames:
Linear Algebra Review

vectors

dot product
Math Review

*matrix-vector multiplication*

(a) as dot products with the rows

(b) as weighted combinations of the columns
Math Review

Cross Product

Right Handed Coordinate System

(curl fingers from $u$ to $v$; thumb points to $u \times v$)
Math Review

Coordinate Systems

Right-handed Coordinate System

Left-handed Coordinate System
Math Review

**Points and Vectors**

*vector space*
- vectors are invariant under translation

*affine space*:
- allows vector-to-point addition
Math Review

Coordinate System vs Frame

coordinate system:
frame:
Math Review

Working with Frames

\[ P = O + x\vec{i} + y\vec{j} \]
Many Coordinate Frames in a Scene
(and using transformation matrices to move between them)