CSE 167: Introduction to Computer Graphics Lecture #3: Projection

Jürgen P. Schulze, Ph.D.
University of California, San Diego
Fall Quarter 2014

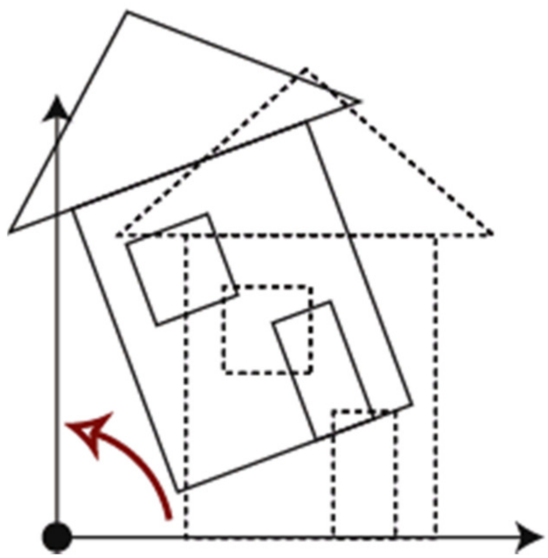
Announcements

- Project 1 due Friday, 10/17
- Discussion section on Mondays is now in a new location at a different time:
  - Building CSB (Cognitive Science Building), room 001
  - 5:00-5:50pm
- We <u>will</u> have a discussion this coming Monday
- Piazza to replace Ted discussion board
- TA office hours (Dylan) in lab today: 5-7pm
- Independent study positions available (CSE 199):
  - Raspberry Pi scripting with Python
  - Android app with web backend

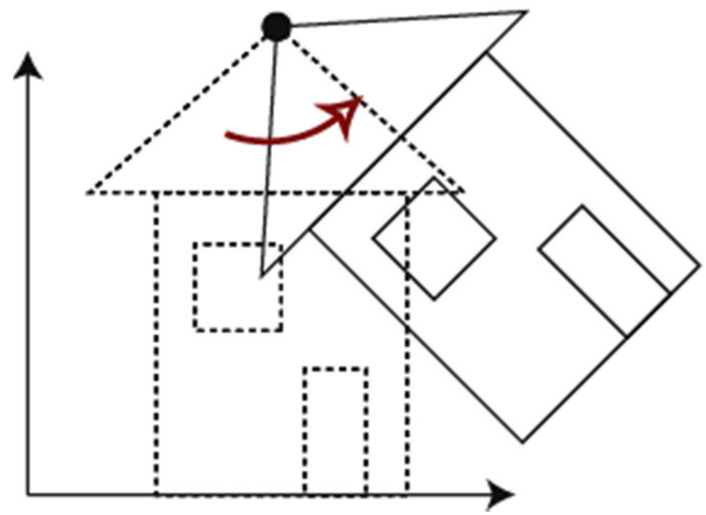
Lecture Overview

- Concatenating Transformations
- Coordinate Transformation
- Typical Coordinate Systems
- Projection

How to rotate around a Pivot Point?

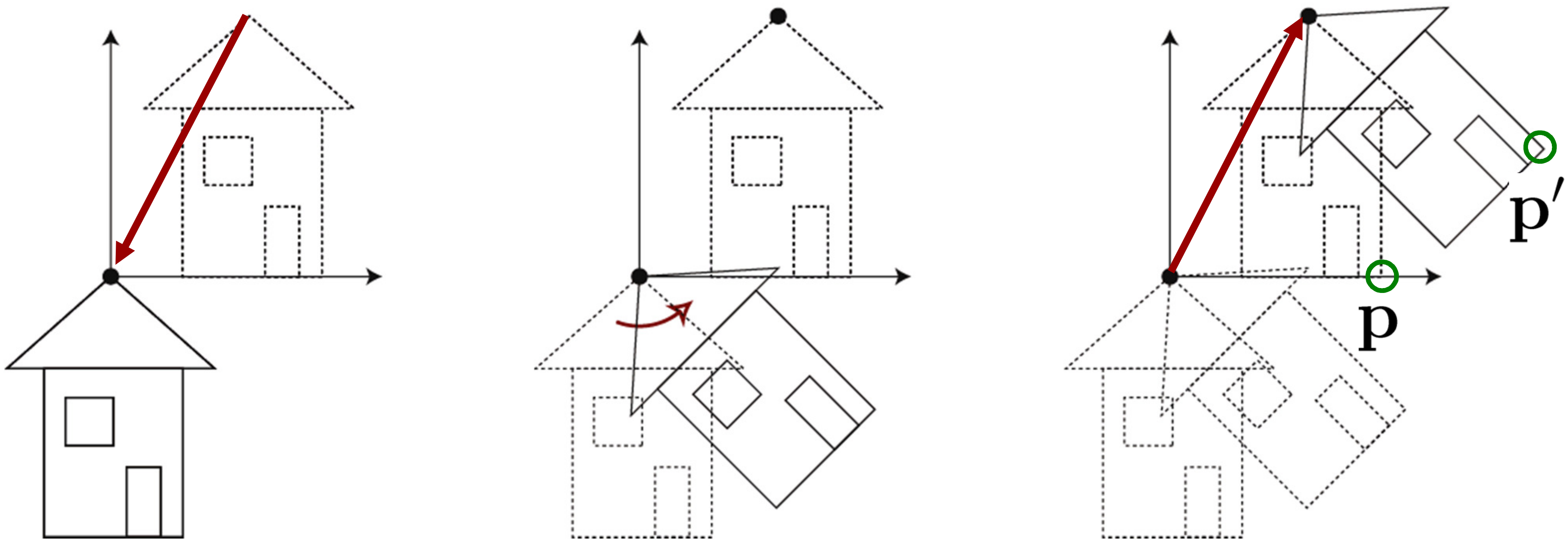


Rotation around origin:
p’ = R p



Rotation around pivot point:
p’ = ?

Rotating point p around a pivot point



1. Translation T 2. Rotation R 3. Translation T^{-1}

$$p' = T^{-1} R T p$$

Concatenating transformations

- Given a sequence of transformations $\mathbf{M}_3\mathbf{M}_2\mathbf{M}_1$

$$\mathbf{p}' = \mathrm{M}_3\mathrm{M}_2\mathrm{M}_1\mathbf{p}$$

$$\mathrm{M}_{total} = \mathrm{M}_3\mathrm{M}_2\mathrm{M}_1$$

$$\mathbf{p}' = \mathrm{M}_{total}\mathbf{p}$$

- Note: associativity applies:

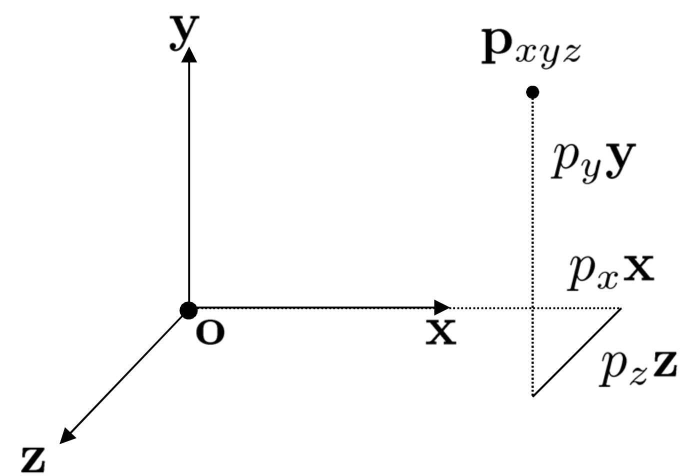
$$\mathrm{M}_{total} = (\mathrm{M}_3\mathrm{M}_2)\mathrm{M}_1 = \mathrm{M}_3(\mathrm{M}_2\mathrm{M}_1)$$

Lecture Overview

- Concatenating Transformations
- Coordinate Transformation
- Typical Coordinate Systems
- Projection

Coordinate System

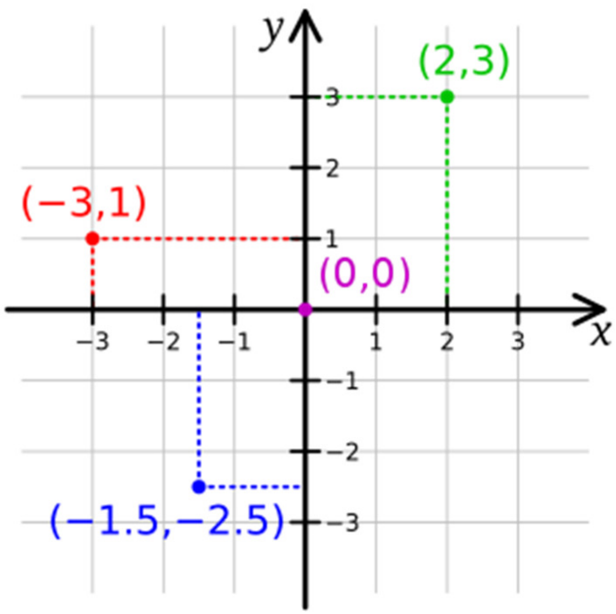
- Given point **p** in homogeneous coordinates: $\begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix}$
- Coordinates describe the point's 3D position in a coordinate system with basis vectors **x**, **y**, **z** and origin **o**:



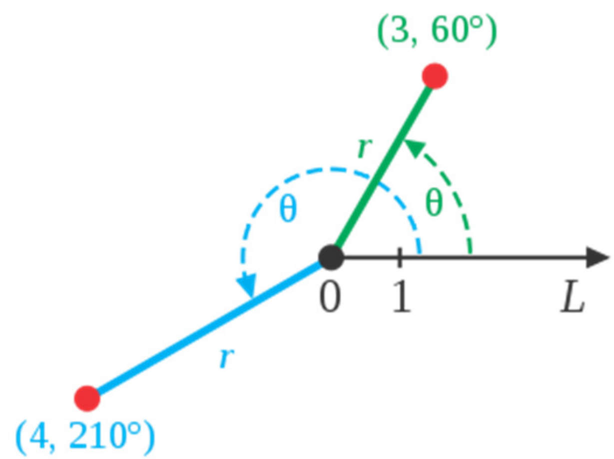
$$\mathbf{p}_{xyz} = p_x\mathbf{x} + p_y\mathbf{y} + p_z\mathbf{z} + \mathbf{o}$$

Rectangular and Polar Coordinates

Rectangular Coordinates



Polar Coordinates

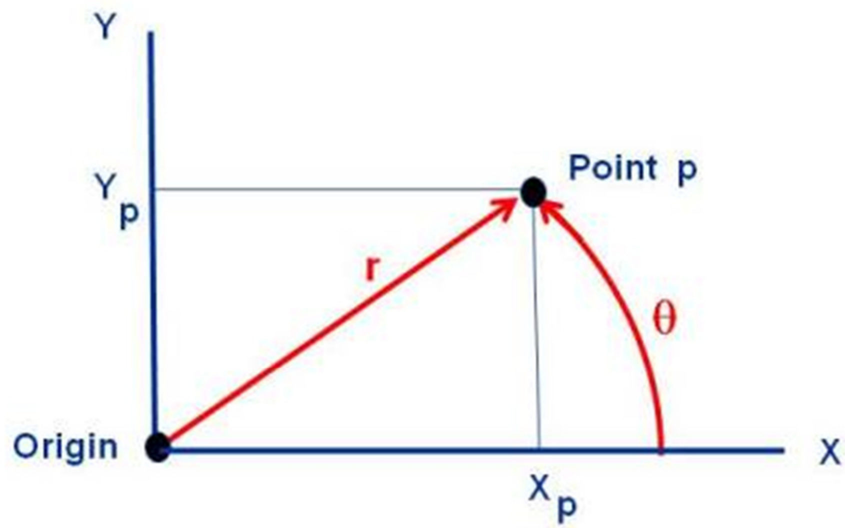


Images: Wikipedia

Rectangular and Polar Coordinates

National Aeronautics and Space Administration

Rectangular and Polar Coordinates



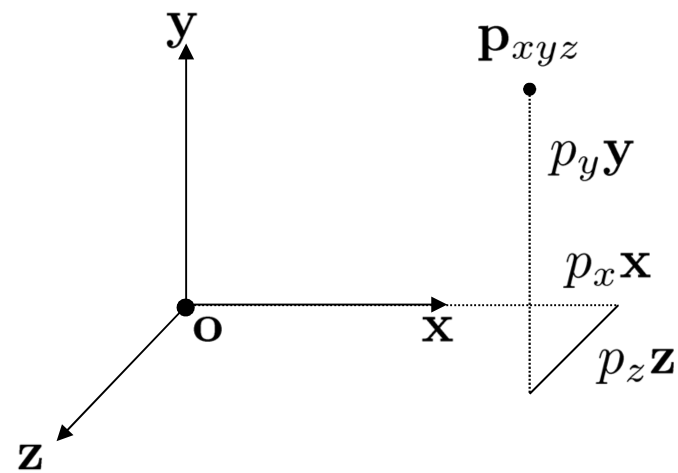
Point p can be located relative to the origin by Rectangular Coordinates (X_p, Y_p) or by Polar Coordinates (r, θ)

$X_p = r \cos(\theta)$ $\qquad$ $r = \mathrm{sqrt}(X_p^2 + Y_p^2)$

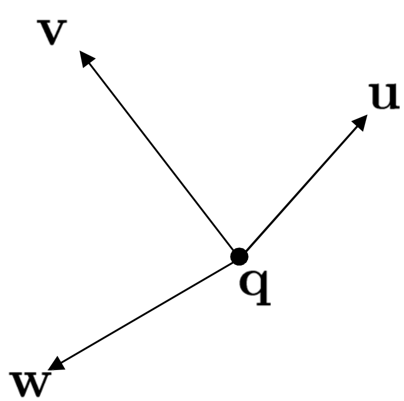
$Y_p = r \sin(\theta)$ $\qquad$ $\theta = \tan^{-1}(Y_p / X_p)$

www.nasa.gov

Coordinate Transformation



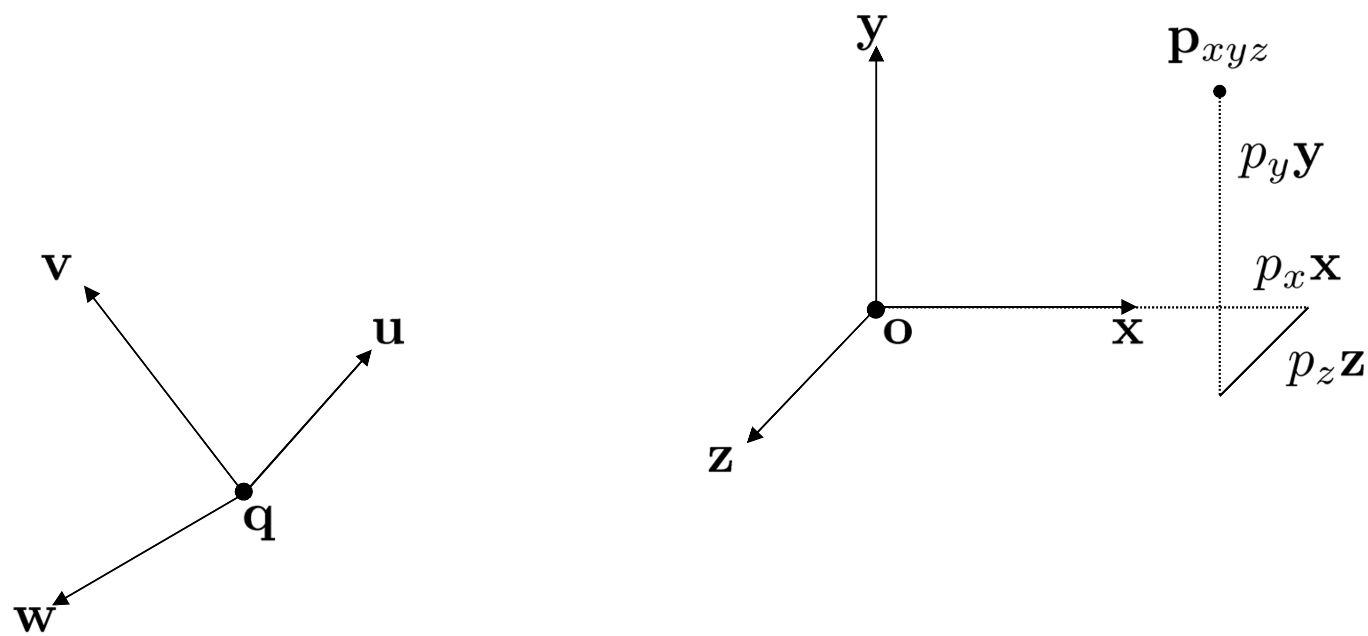
Original **xyzo** coordinate system



New **uvwq** coordinate system

Goal: Find coordinates of $\mathbf{p}_{xyz}$ in new **uvwq** coordinate system

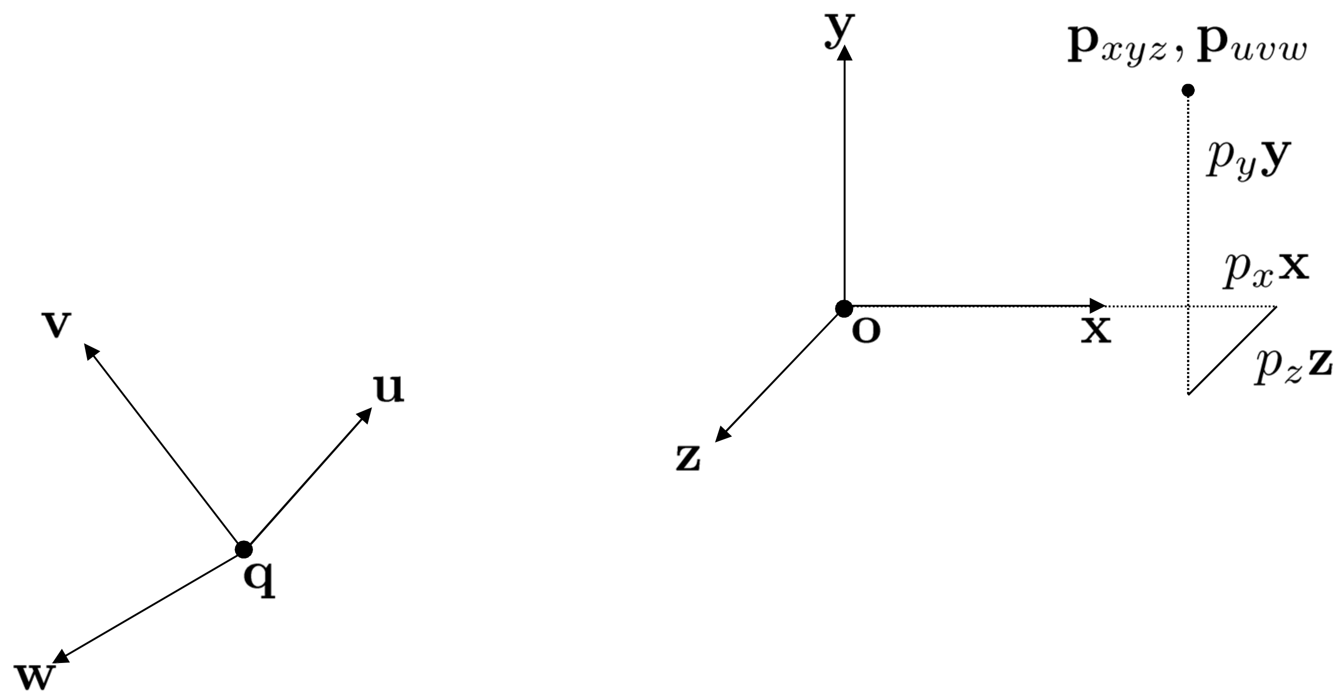
Coordinate Transformation



Express coordinates of $\mathbf{xyzo}$ reference frame with respect to $\mathbf{uvwq}$ reference frame:

$$\mathbf{x} = \begin{bmatrix} x_u \\ x_v \\ x_w \\ 0 \end{bmatrix} \qquad \mathbf{y} = \begin{bmatrix} y_u \\ y_v \\ y_w \\ 0 \end{bmatrix} \qquad \mathbf{z} = \begin{bmatrix} z_u \\ z_v \\ z_w \\ 0 \end{bmatrix} \qquad \mathbf{o} = \begin{bmatrix} o_u \\ o_v \\ o_w \\ 1 \end{bmatrix}$$

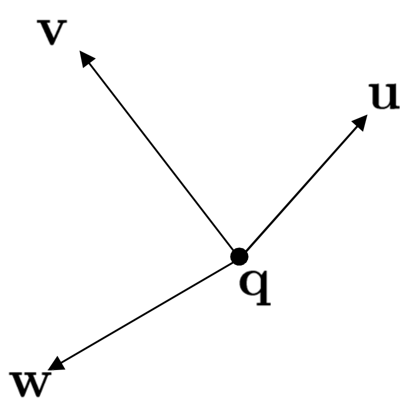
Coordinate Transformation

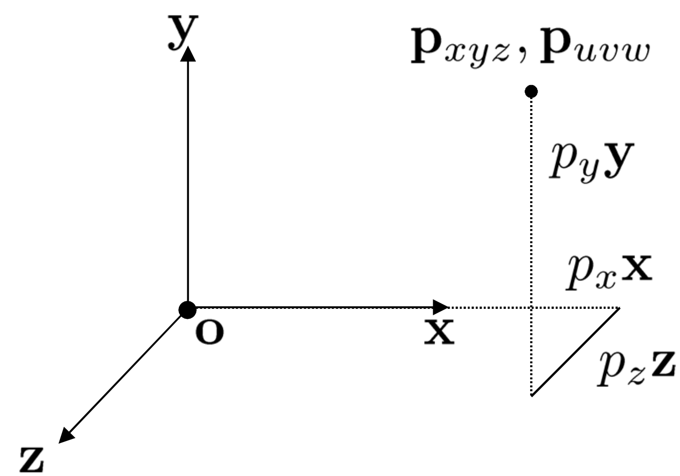


Point $\mathbf{p}$ expressed in new $\mathbf{uvwq}$ reference frame:

$$\mathbf{p}_{uvw} = p_x \begin{bmatrix} x_u \\ x_v \\ x_w \\ 0 \end{bmatrix} + p_y \begin{bmatrix} y_u \\ y_v \\ y_w \\ 0 \end{bmatrix} + p_z \begin{bmatrix} z_u \\ z_v \\ z_w \\ 0 \end{bmatrix} + \begin{bmatrix} o_u \\ o_v \\ o_w \\ 1 \end{bmatrix}$$

Coordinate Transformation





$$\mathbf{p}_{uvw} = \begin{bmatrix} x_u & y_u & z_u & o_u \\ x_v & y_v & z_v & o_v \\ x_w & y_w & z_w & o_w \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{x} & \mathbf{y} & \mathbf{z} & \mathbf{o} \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix}$$

Coordinate Transformation

Inverse transformation

- Given point $\mathbf{P}_{uvw}$ w.r.t. reference frame **uvwq:**
  - Coordinates $\mathbf{P}_{xyz}$ w.r.t. reference frame **xyzo** are calculated as**:**

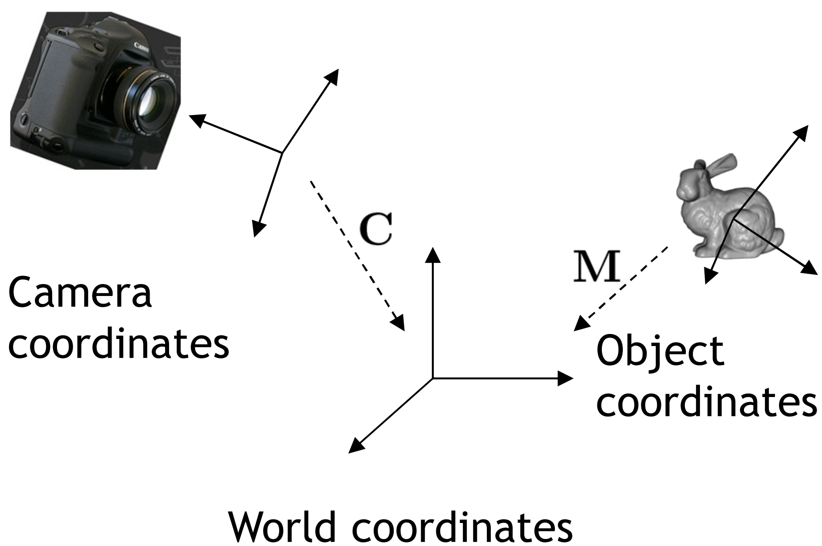
$$\mathbf{p}_{xyz} = \begin{bmatrix} x_u & y_u & z_u & o_u \\ x_v & y_v & z_v & o_v \\ x_w & y_w & z_w & o_w \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} p_u \\ p_v \\ p_w \\ 1 \end{bmatrix}$$

Lecture Overview

- Concatenating Transformations
- Coordinate Transformation
- Typical Coordinate Systems
- Projection

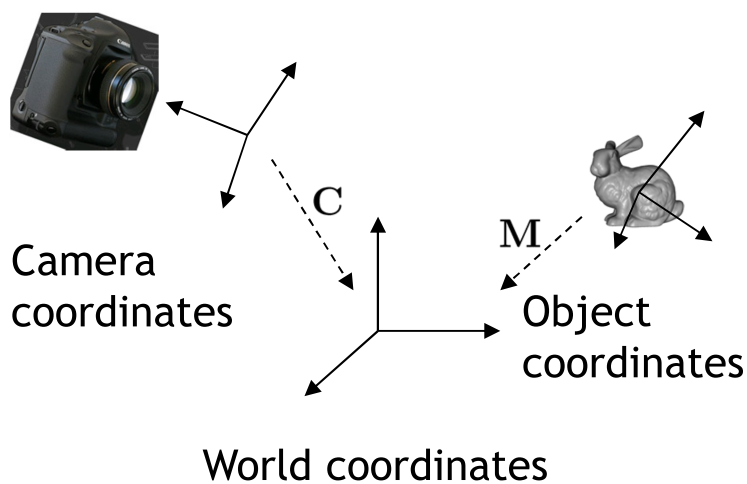
Typical Coordinate Systems

- In computer graphics, we typically use at least three coordinate systems:
  - World coordinate system
  - Camera coordinate system
  - Object coordinate system



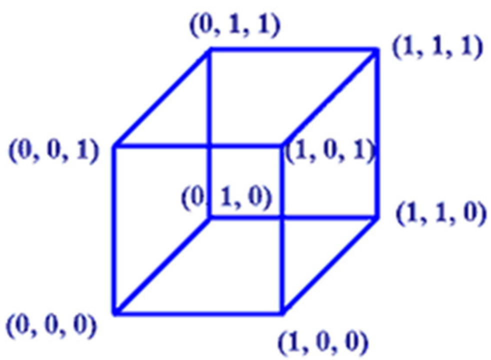
World Coordinates

- Common reference frame for all objects in the scene
- No standard for coordinate system orientation
  - If there is a ground plane, usually x/y is horizontal and z points up (height)
  - Otherwise, x/y is often screen plane, z points out of the screen

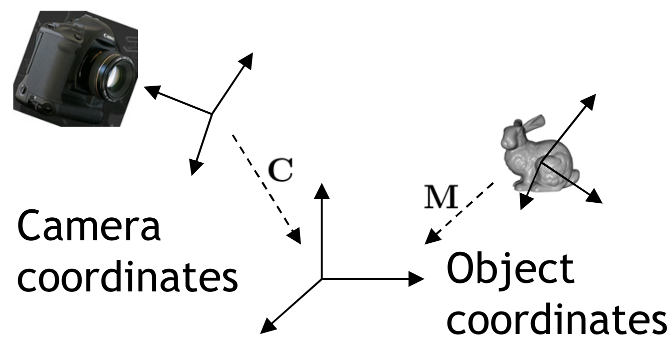


Object Coordinates

- Local coordinates in which points and other object geometry are given
- Often origin is in geometric center, on the base, or in a corner of the object
  - Depends on how object is generated or used.



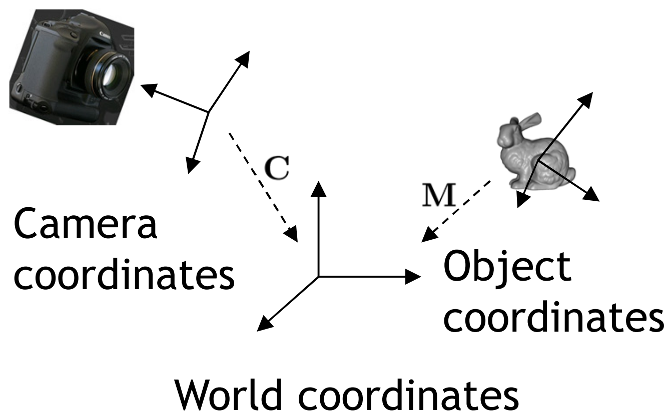
Source: http://motivate.maths.org



World coordinates

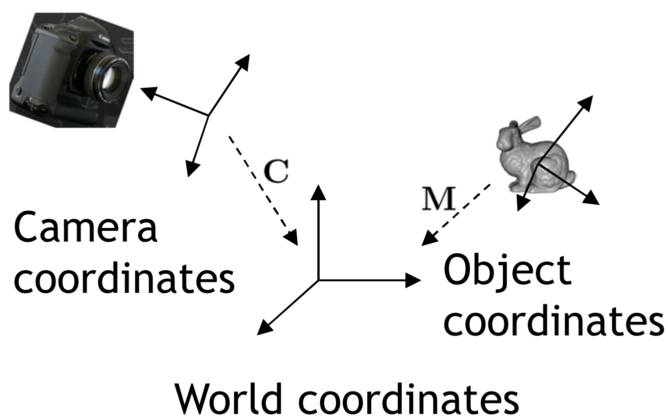
Object Transformation

- The transformation from object to world coordinates is different for each object.
- Defines placement of object in scene.
- Given by "model matrix" (model-to-world transformation) **M**.



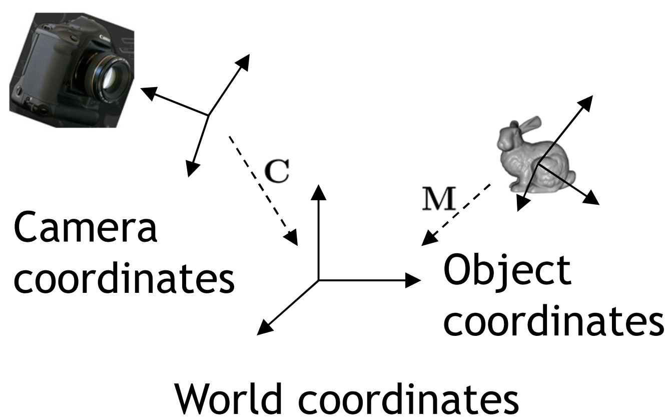
Camera Coordinate System

- Origin defines center of projection of camera
- x-y plane is parallel to image plane
- z-axis is perpendicular to image plane



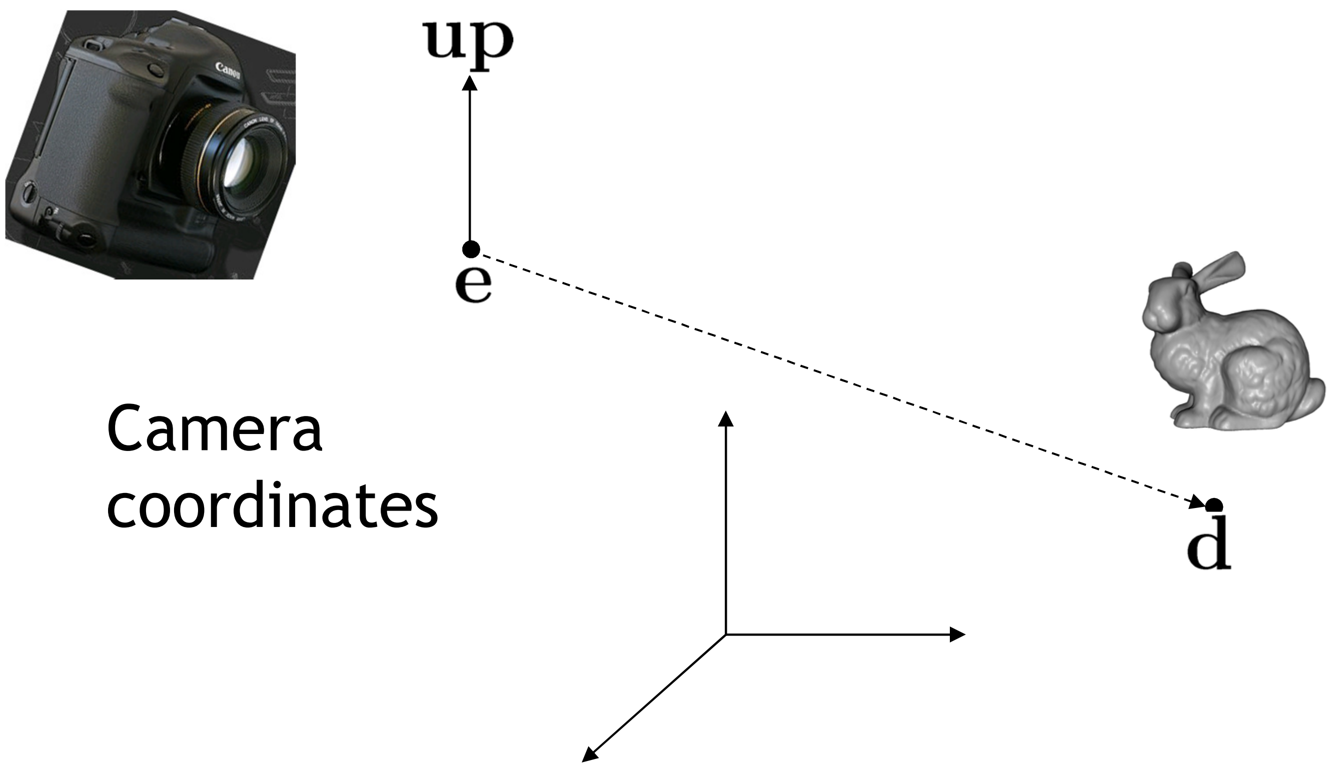
Camera Coordinate System

- The Camera Matrix defines the transformation from camera to world coordinates
  - Placement of camera in world



Camera Matrix

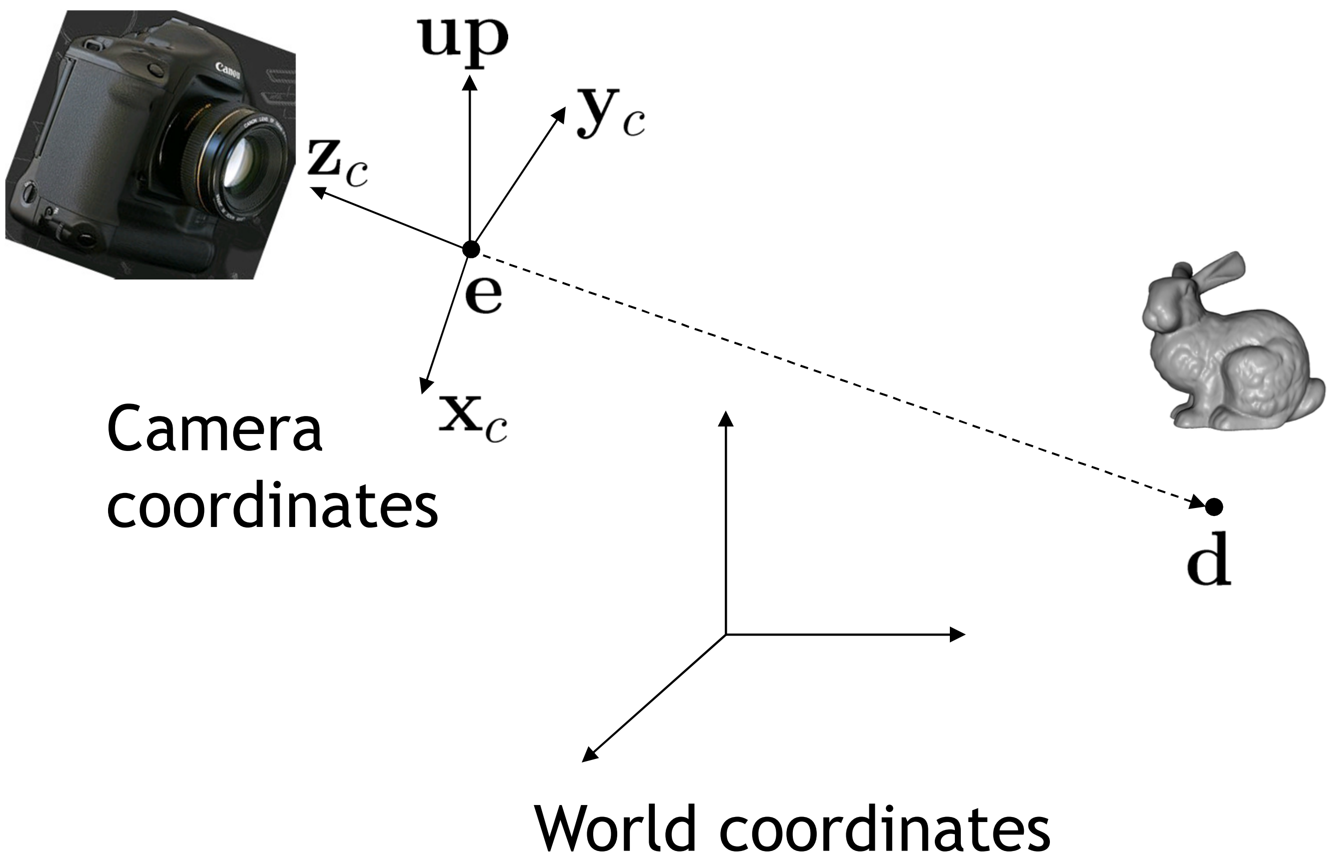
- Construct from center of projection **e**, look at **d**, up-vector **up:**



World coordinates

Camera Matrix

- Construct from center of projection **e**, look at **d**, up-vector **up** (up in camera coordinate system):



Camera Matrix

- z-axis

$$\boldsymbol{z}_C = \frac{\boldsymbol{e}-\boldsymbol{d}}{\|\boldsymbol{e}-\boldsymbol{d}\|}$$

- x-axis

$$\boldsymbol{x}_C = \frac{\boldsymbol{up} \times \boldsymbol{z}_C}{\|\boldsymbol{up} \times \boldsymbol{z}_C\|}$$

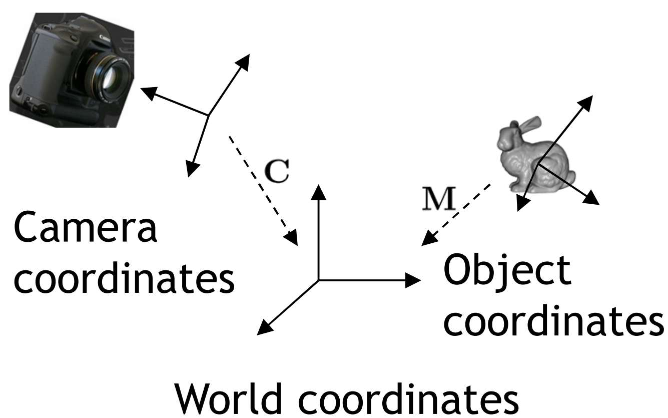
- y-axis

$$\boldsymbol{y}_C = \boldsymbol{z}_C \times \boldsymbol{x}_C = \frac{\boldsymbol{up}}{\|\boldsymbol{up}\|}$$

$$\boldsymbol{C} = \begin{bmatrix} \boldsymbol{x}_C & \boldsymbol{y}_C & \boldsymbol{z}_C & \boldsymbol{e} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Transforming Object to Camera Coordinates

- Object to world coordinates: **M**
- Camera to world coordinates: **C**
- Point to transform: **p**
- Resulting transformation equation: $\mathbf{p}' = \mathbf{C}^{-1} \mathbf{M} \mathbf{p}$



Tips for Notation

- Indicate coordinate systems with every point or matrix
  - Point: $\mathbf{p}_{\text{object}}$
  - Matrix: $\mathbf{M}_{\text{object}\rightarrow\text{world}}$
- Resulting transformation equation:

$$\mathbf{p}_{\text{camera}} = (\mathbf{C}_{\text{camera}\rightarrow\text{world}})^{-1}\ \mathbf{M}_{\text{object}\rightarrow\text{world}}\ \mathbf{p}_{\text{object}}$$

- Helpful hint: in source code use consistent names
  - Point: `p_object` or `p_obj` or `p_o`
  - Matrix: `object2world` or `obj2wld` or `o2w`
- Resulting transformation equation:

```
wld2cam = inverse(cam2wld);
p_cam = p_obj * obj2wld * wld2cam;
```

Inverse of Camera Matrix

- How to calculate the inverse of the camera matrix $\mathbf{C}^{-1}$?
- Generic matrix inversion is complex and compute-intensive
- Affine transformation matrices can be inverted more easily
- Observation:
  - Camera matrix consists of translation and rotation: $\mathbf{T} \times \mathbf{R}$
- Inverse of rotation: $\mathbf{R}^{-1} = \mathbf{R}^T$
- Inverse of translation: $\mathbf{T}(t)^{-1} = \mathbf{T}(-t)$
- Inverse of camera matrix: $\mathbf{C}^{-1} = \mathbf{T}^{-1} \times \mathbf{R}^{-1}$

Objects in Camera Coordinates

- We have things lined up the way we like them on screen
  - **x** to the right
  - **y** up
  - **-z** into the screen
  - Objects to look at are in front of us, i.e. have negative z values
- But objects are still in 3D
- Next step: project scene to 2D plane

Lecture Overview

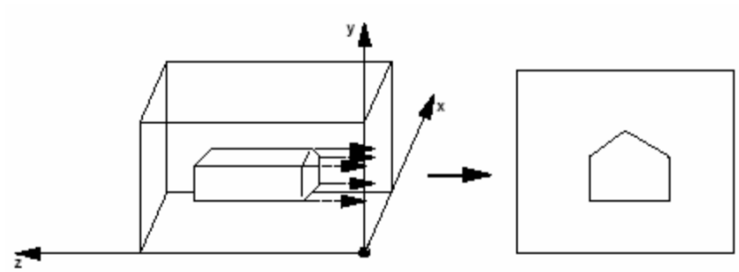
- Concatenating Transformations
- Coordinate Transformation
- Typical Coordinate Systems
- Projection

Projection

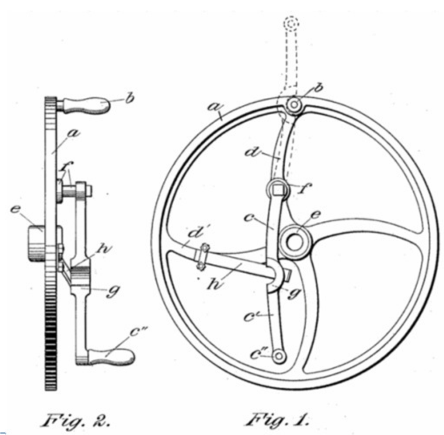
- Goal:
  Given 3D points (vertices) in camera coordinates, determine corresponding image coordinates
- Transforming 3D points into 2D is called Projection
- OpenGL supports two types of projection:
  - Orthographic Projection (=Parallel Projection)
  - Perspective Projection

Orthographic Projection

- Can be done by ignoring **z**-coordinate
  - Use camera space **xy** coordinates as image coordinates
- Project points to **x-y** plane along parallel lines

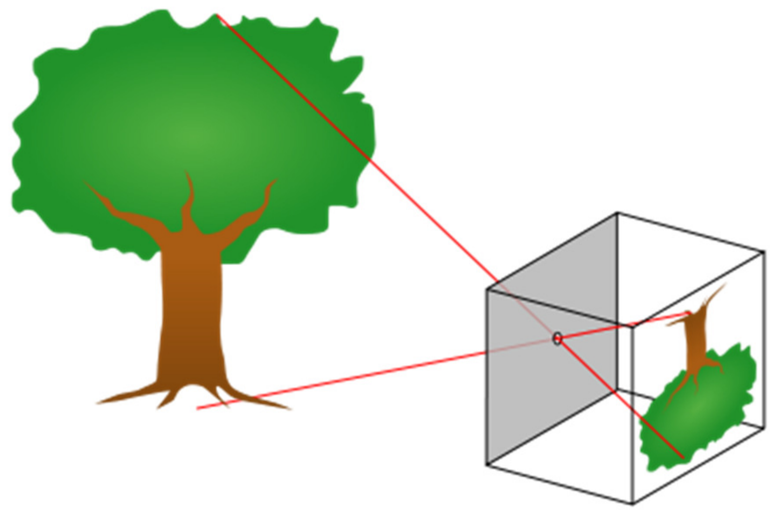


- Often used in graphical illustrations, architecture, 3D modeling



Perspective Projection

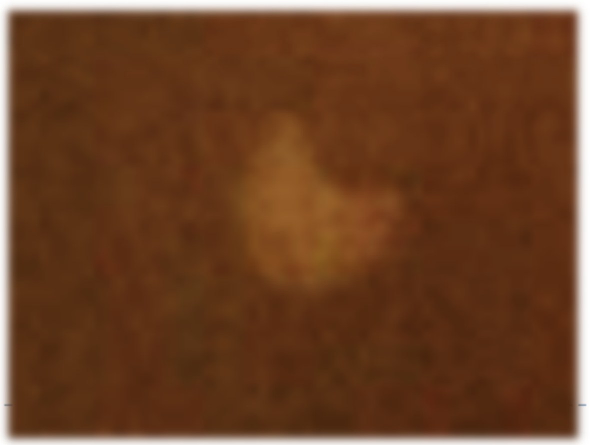
- Most common for computer graphics
- Simplified model of human eye, or camera lens (*pinhole camera*)



- Things farther away appear to be smaller
- Discovery attributed to Filippo Brunelleschi (Italian architect) in the early 1400's

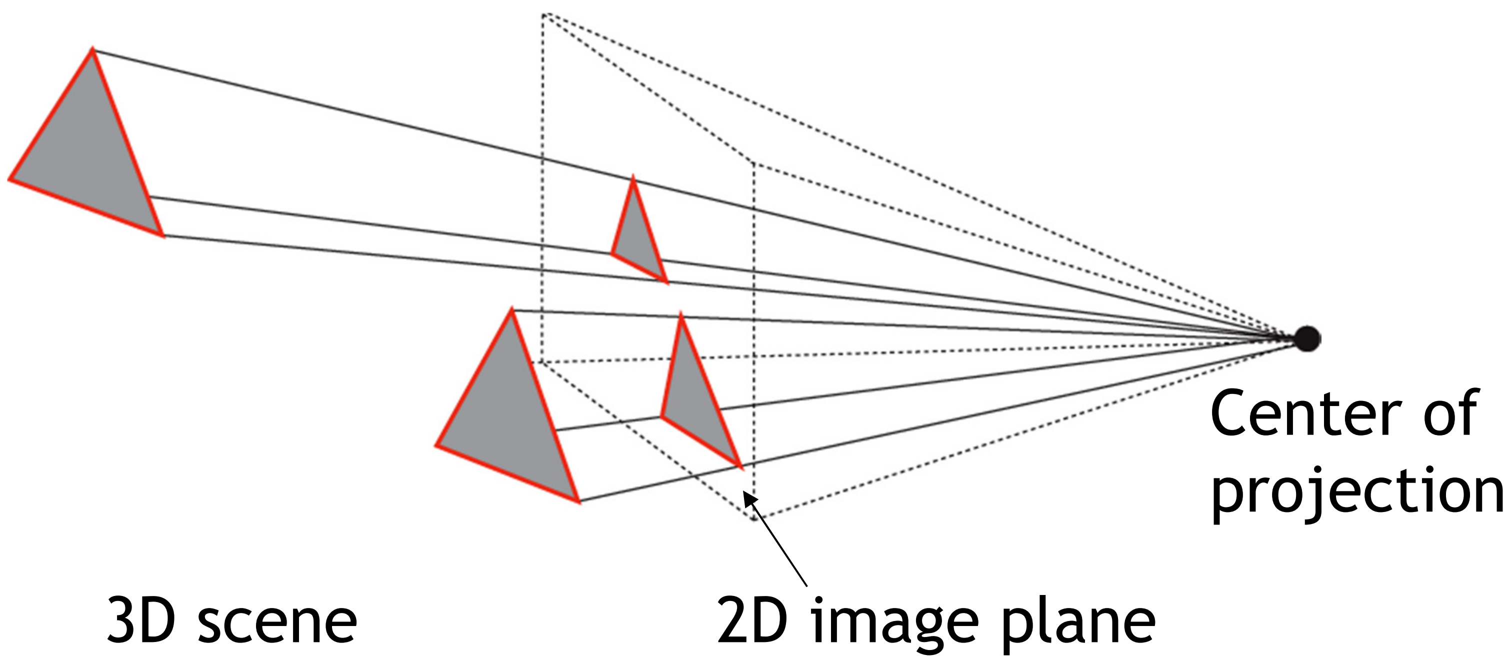
Pinhole Camera

- San Diego, May 20th, 2012

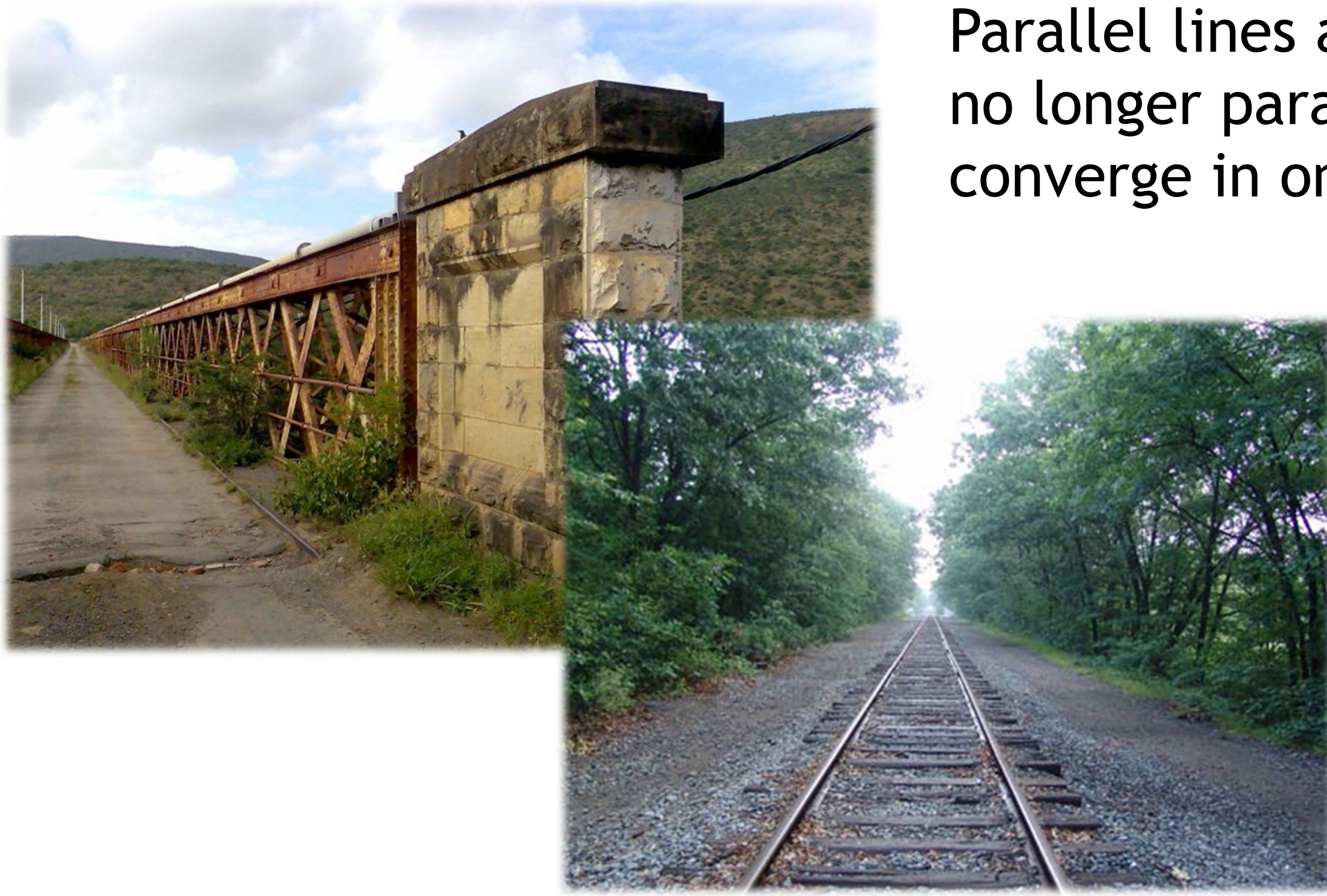


Perspective Projection

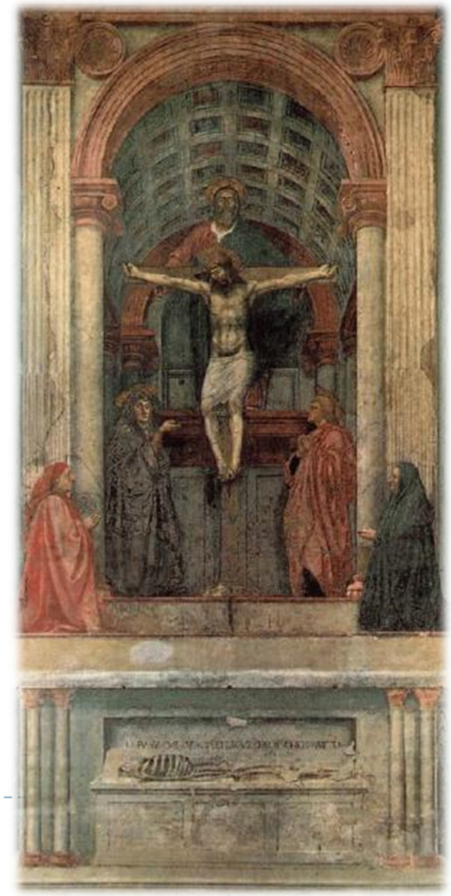
- Project along rays that converge in center of projection



Perspective Projection



Parallel lines are no longer parallel, converge in one point



Earliest example:
La Trinitá (1427) by Masaccio

Video

- Professor Ravi Ramamoorthi on Perspective Projection
  - http://www.youtube.com/watch?v=VpNJbvZhNCQ