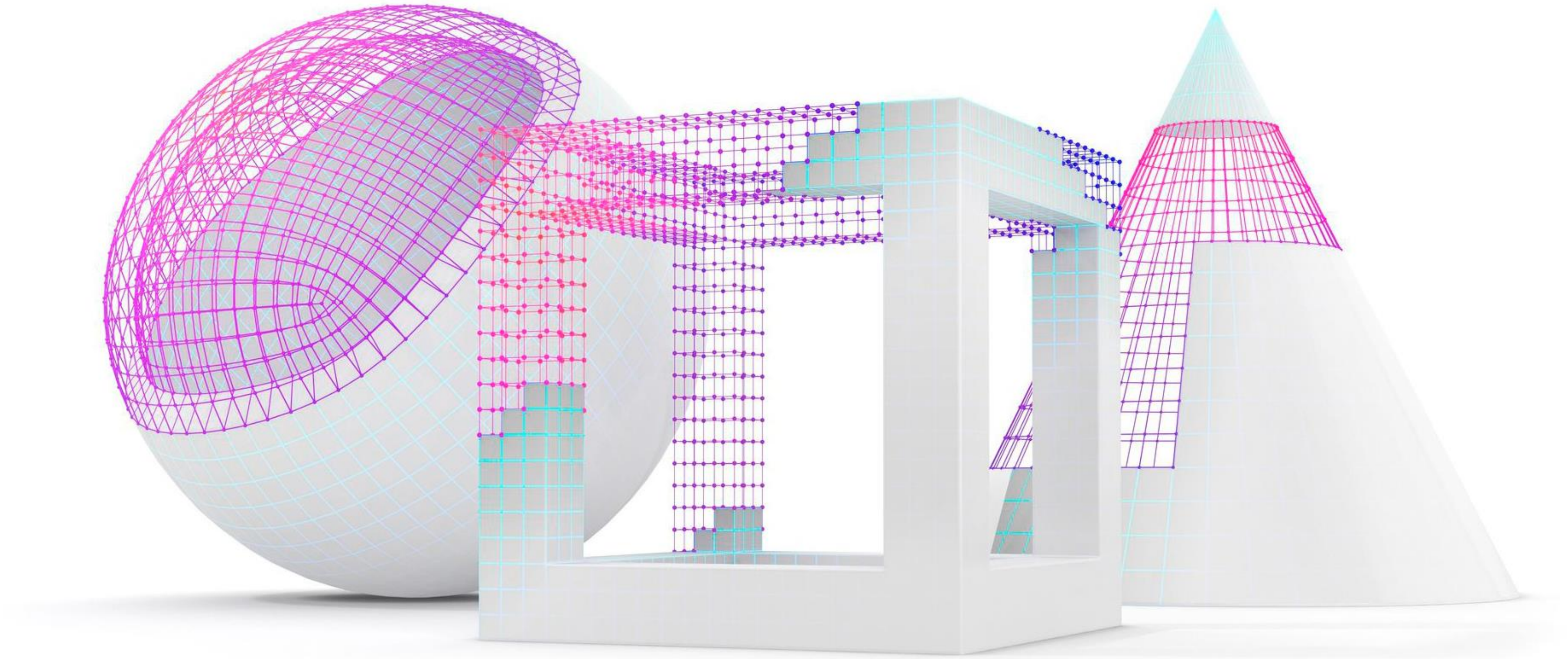


Computer Graphics

Graphics Pipeline

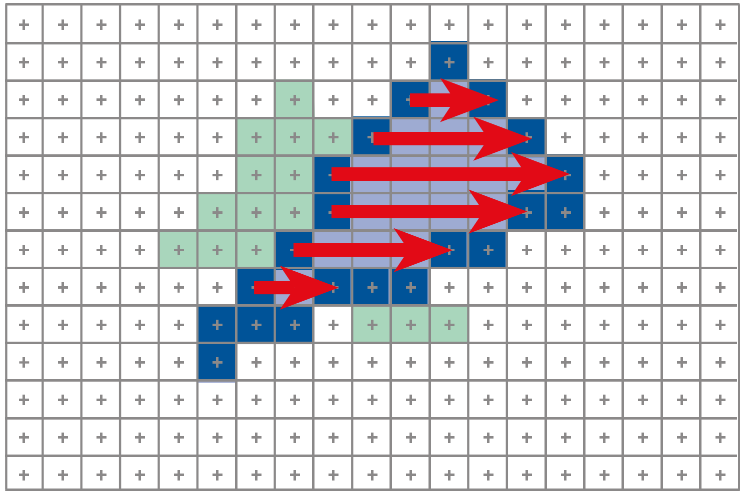
Andreas Aristidou
andarist@ucy.ac.cy
http://www.andreasaristidou.com

In general, there are two approaches for creating images

A. Rendering Pipeline

- For each triangle
  - For each projected pixel

Project scene to the pixels



B. Ray Casting

- For each pixel
  - For each object

Send pixels to the scene

Ray Casting

- Very simple method
- Very "realistic" images
- But: slow – definitely not "real-time"
-

Images from LightWork Design Ltd

But we are getting close…

https://youtu.be/1IliQZw_p_E

We can accelerate by simplifying

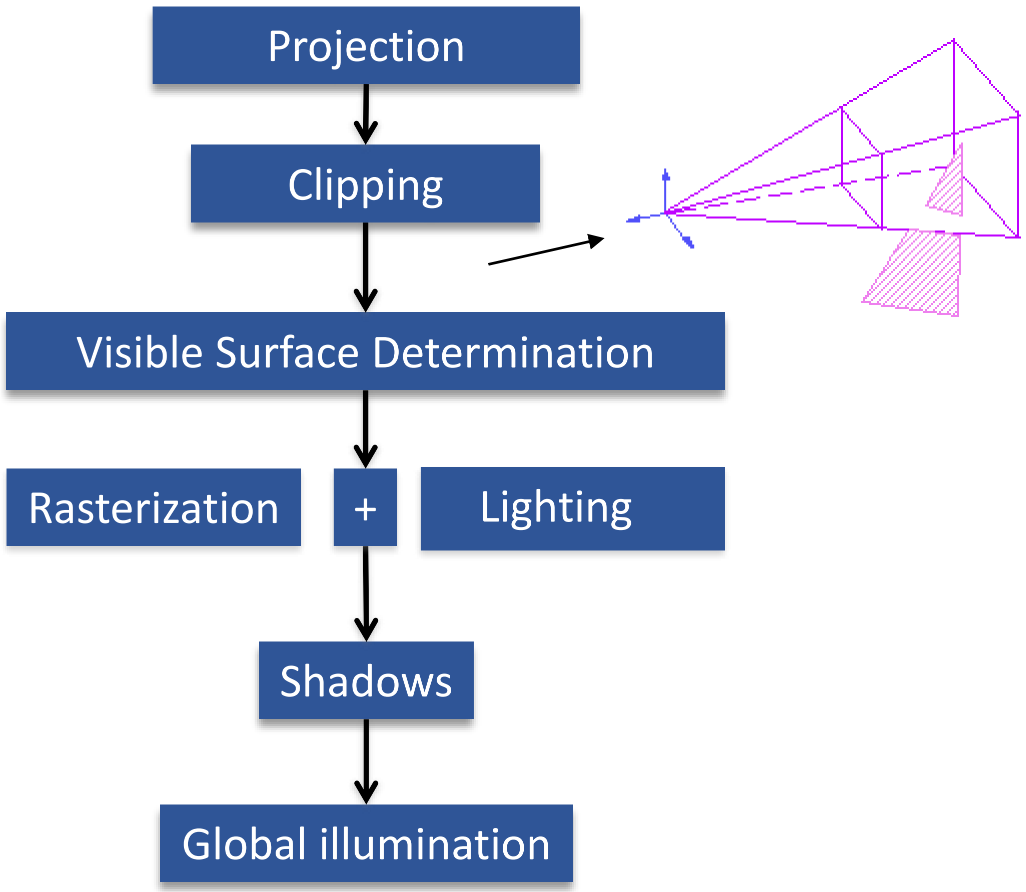
- Ray-tracing is a very slow process
    - E.g. for a cow we have 10,000 triangles, but to create an image of 1000 x 1000 pixels we need to shed 1,000,000 primary rays, each one having to intersect the 10,000 triangles (and then add reflections, shadows, etc.)
  - Reduce global lighting (i.e. no recursion)
  - Reduce lighting, left only the lighting from the environment
  - Assume that there are only polygons
  - Instead of detecting rays in each pixel, we simply locate them on the vertices and fill the interval in between
    - **More in the upcoming lectures**

Rendering Pipeline

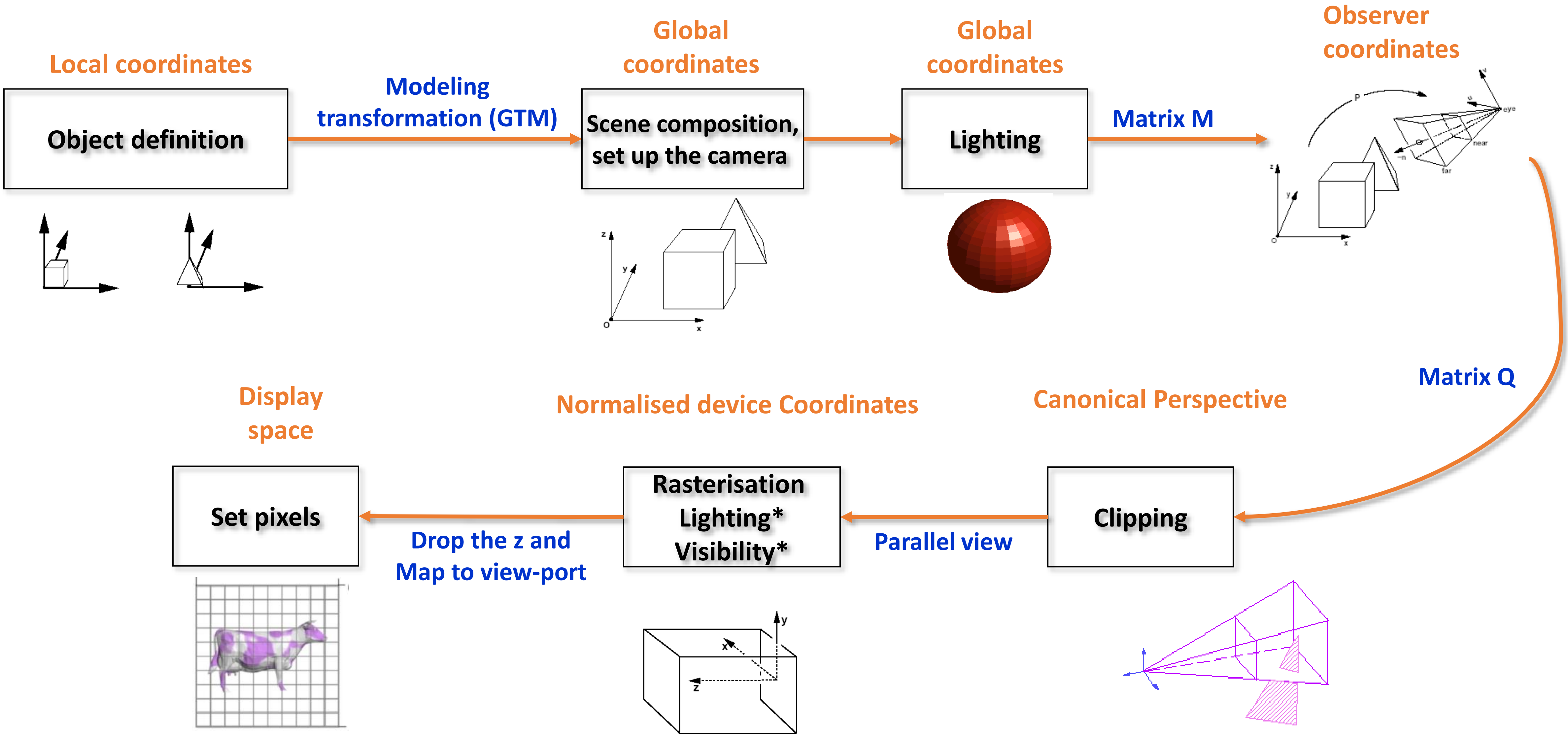
- In computer graphics, **computer graphics pipeline**, **rendering pipeline**, or just **graphics pipeline**, is a conceptual model that describes the steps that a graphics system must perform to render a 3d scene on a 2d screen.
- The 3D graphical pipeline usually refers to the most common 3D rendering format on a computer, and is usually called 3D polygon rendering
  - It differs from ray casting and ray tracing.
  - In **ray casting**, a ray is fired from where the camera is located, and if this ray hits a surface of an object, then the color and lighting of the point on the surface is calculated, and this is defined as the color of that pixel.
  - On the contrary, in the **3D polygon rendering**, the area that is in the viewing range of the camera is calculated, and then rays are created from each part of each surface, which are projected back to the camera.

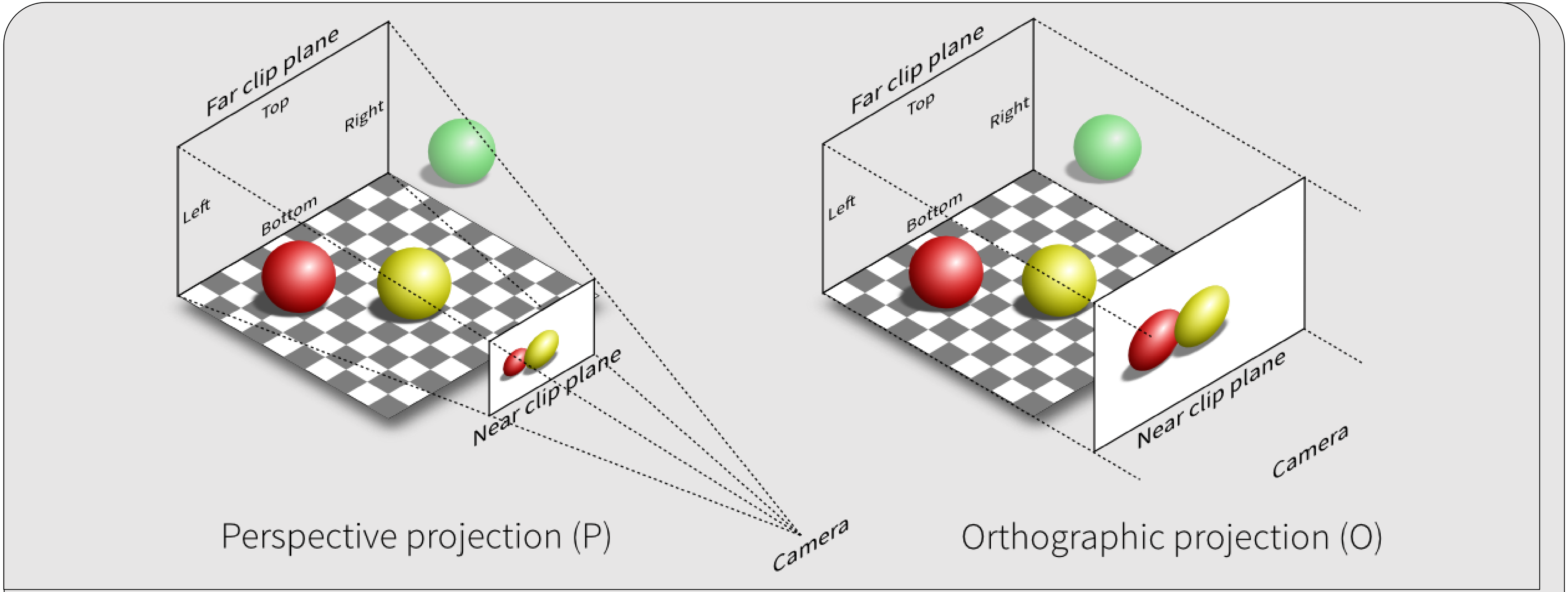
Graphics Pipeline

- Although this approach gives much higher speeds than ray shotting, **it creates several problems** that need to be solved:
  - Projecting the vertices
  - Clipping to the view volume
  - Visible surface determination
  - Rendering a polygon in 2D
  - Lighting
  - Shadows
  - Global illumination (Radiosity)
- Today’s lecture
  - Viewing



The Graphics Pipeline





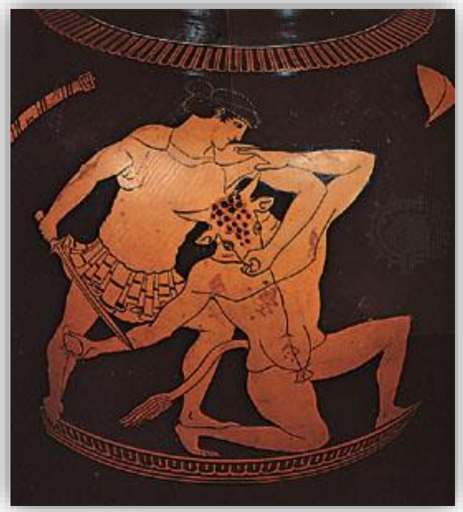
Computer Graphics

Viewing

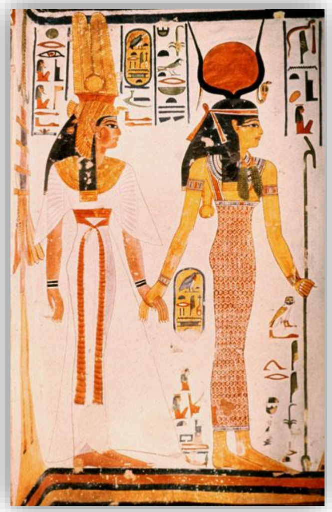
Andreas Aristidou
andarist@ucy.ac.cy
http://www.andreasaristidou.com

History and Overview of Projections

- Signs of efforts to deliver projection
  - *Greek vase:* note relative sizes of thighs and lower legs of minotaur
  - *Ancient Egyptian Art:* Note how depiction of body implies a front view but the feet and head imply side view (early cubism!)
- The painting was based on a mythical story
- A **projection** is the presentation of an image of a 3D scene on a surface and was achieve by adding shadows
  - Detail from:***The Invention of Drawing* (1830)** – Karl Friedrich Schinkle



Greek vase from the late 6th century BC



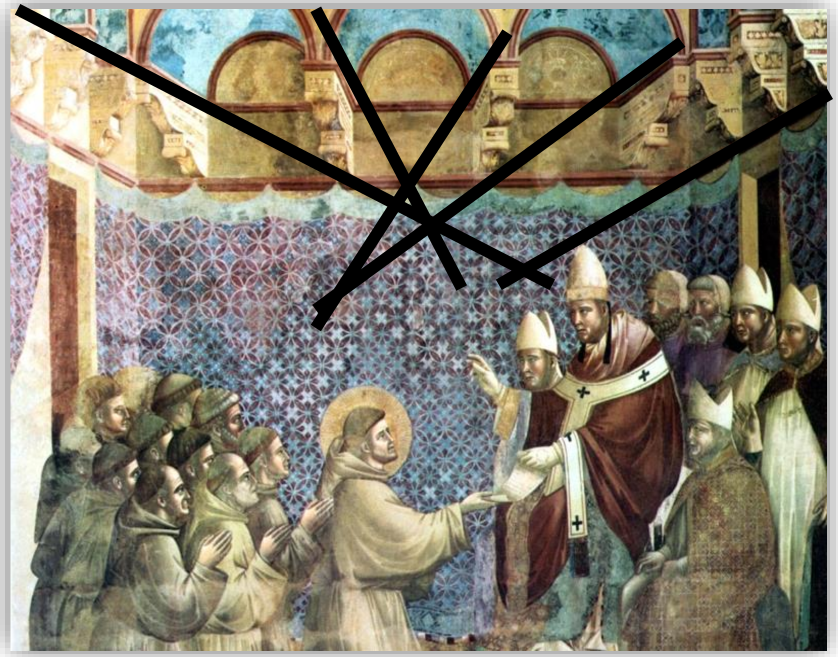
Tomb of Nefertari, Thebes (19th Dyn, ~1270 BC), *Queen Led by Isis.*

William J. Mitchell, *The Reconfigured Eye*, Fig 1.1

History and Overview of Projections

- Starting in the 13th century (AD): New emphasis on importance of individual viewpoint, power of observation (particularly of nature: astronomy, anatomy, etc.)
  - *Masaccio, Donatello, DaVinci, Newton*
- Attempts to represent 3D space more realistically. Earlier works invoke a sense of 3D space but not systematically
  - Parallel lines converge, but no single vanishing point (where all parallel lines converge)

William J. Mitchell, *The Reconfigured Eye*, Fig 1.1



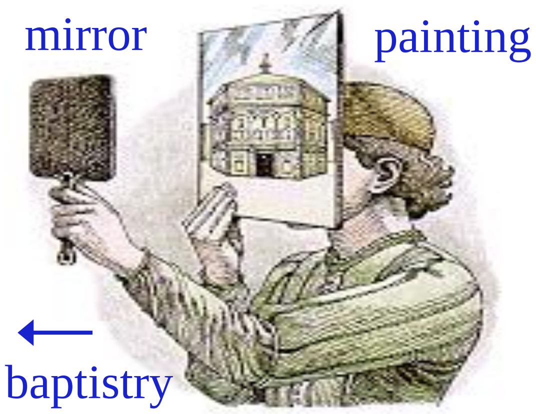
Giotto, *Franciscan Rule Approved,* Assisi, Upper Basilica, c.1288-1292

History and Overview of Projections

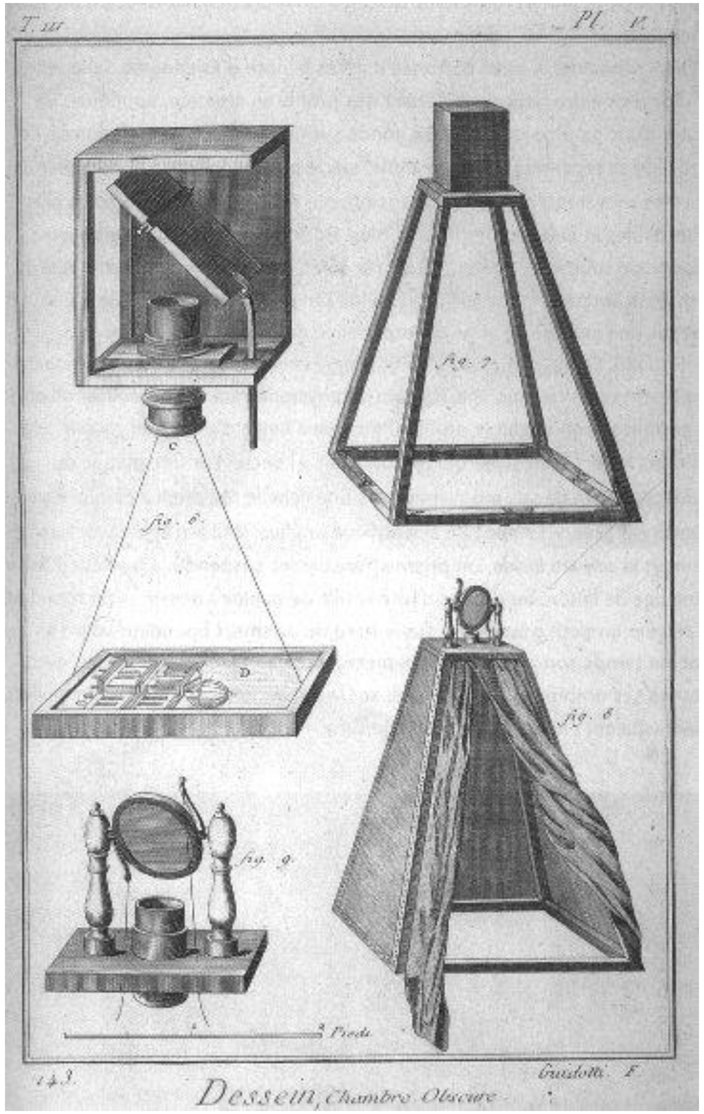
- Vermeer and others created *perspective boxes* where a picture, when viewed through viewing hole, had correct perspective.
- Brunelleschi, architect and inventor, invented systematic method of determining correct perspective projections (early 1400s). (Unfortunately, not documented).
  - The perspective is accurate only from one POV (point of view)
  - *Reported* to have determined the accuracy of his paintings by making a hole in the vanishing point, examining the reflection of his painting in a mirror and comparing the line convergence to the real model
    - His illusion inspired other artists to explore linear perspective

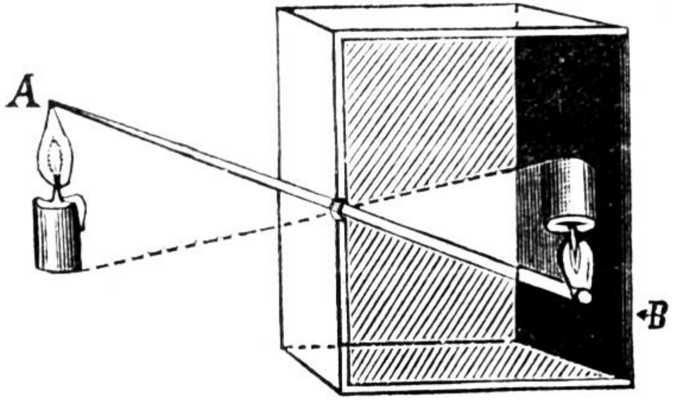
Perspective Box
Samuel van Hoogstraten
National Gallery, London

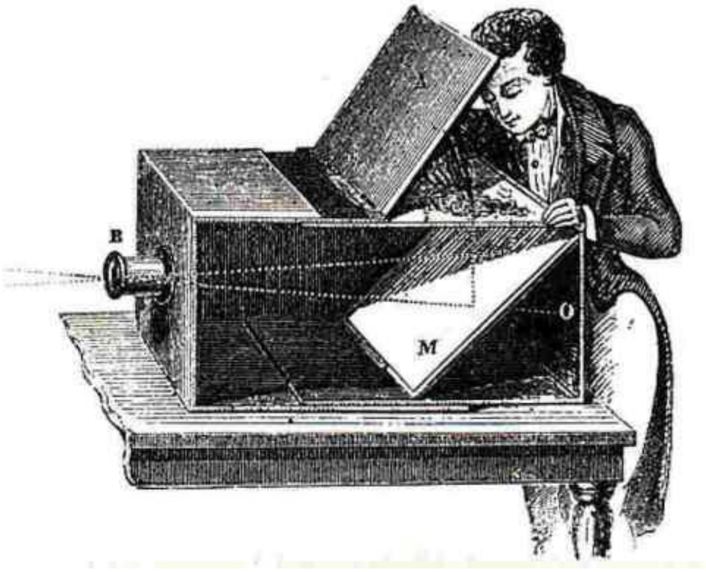
Perspective Box of a Dutch Interior
Samuel van Hoogstraten
National Gallery, London

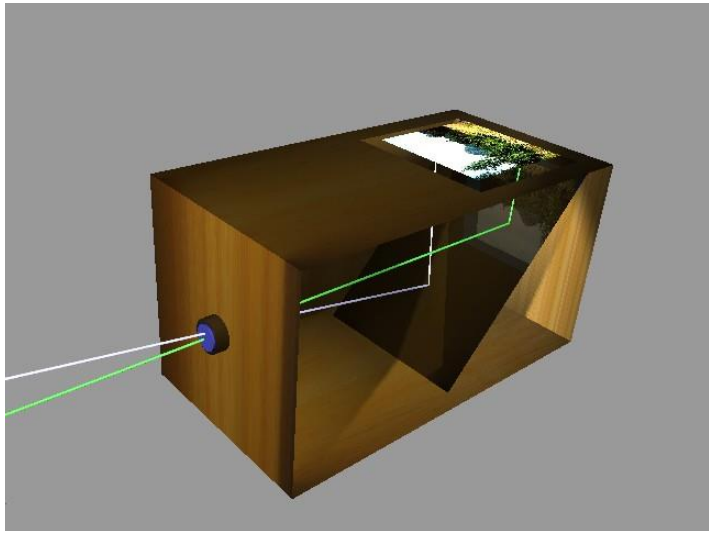


History and Overview of Projections



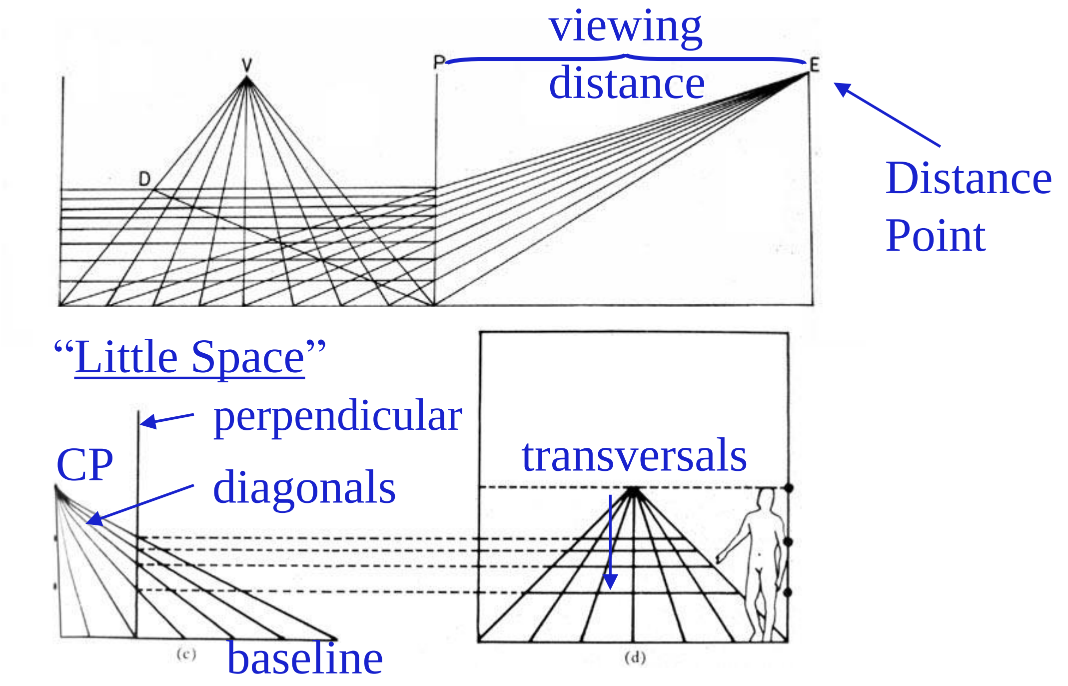






Linear Perspective (Vanishing Points)

- Both Da Vinci and Alberti created accurate geometric ways of incorporating linear perspective into a drawing using the concept of vanishing points

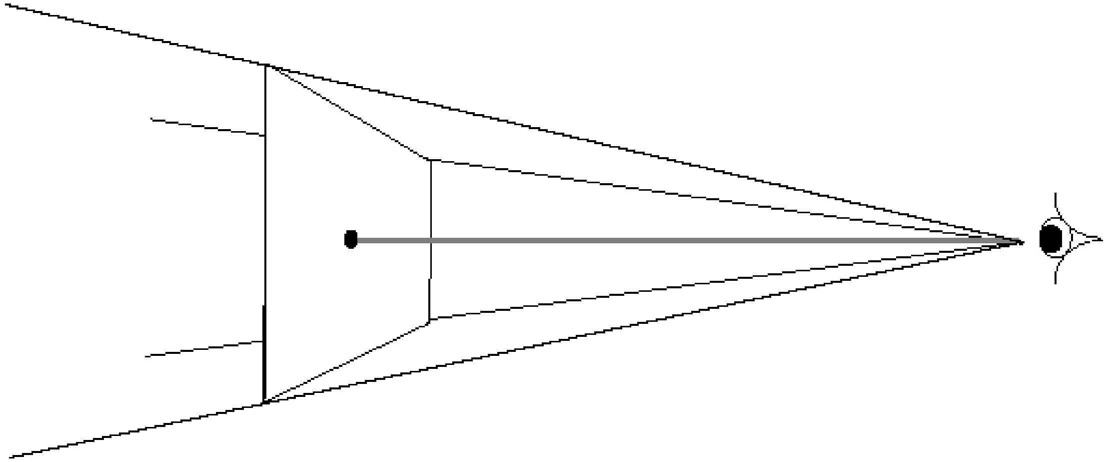


Da Vinci's Method

Alberti's Method

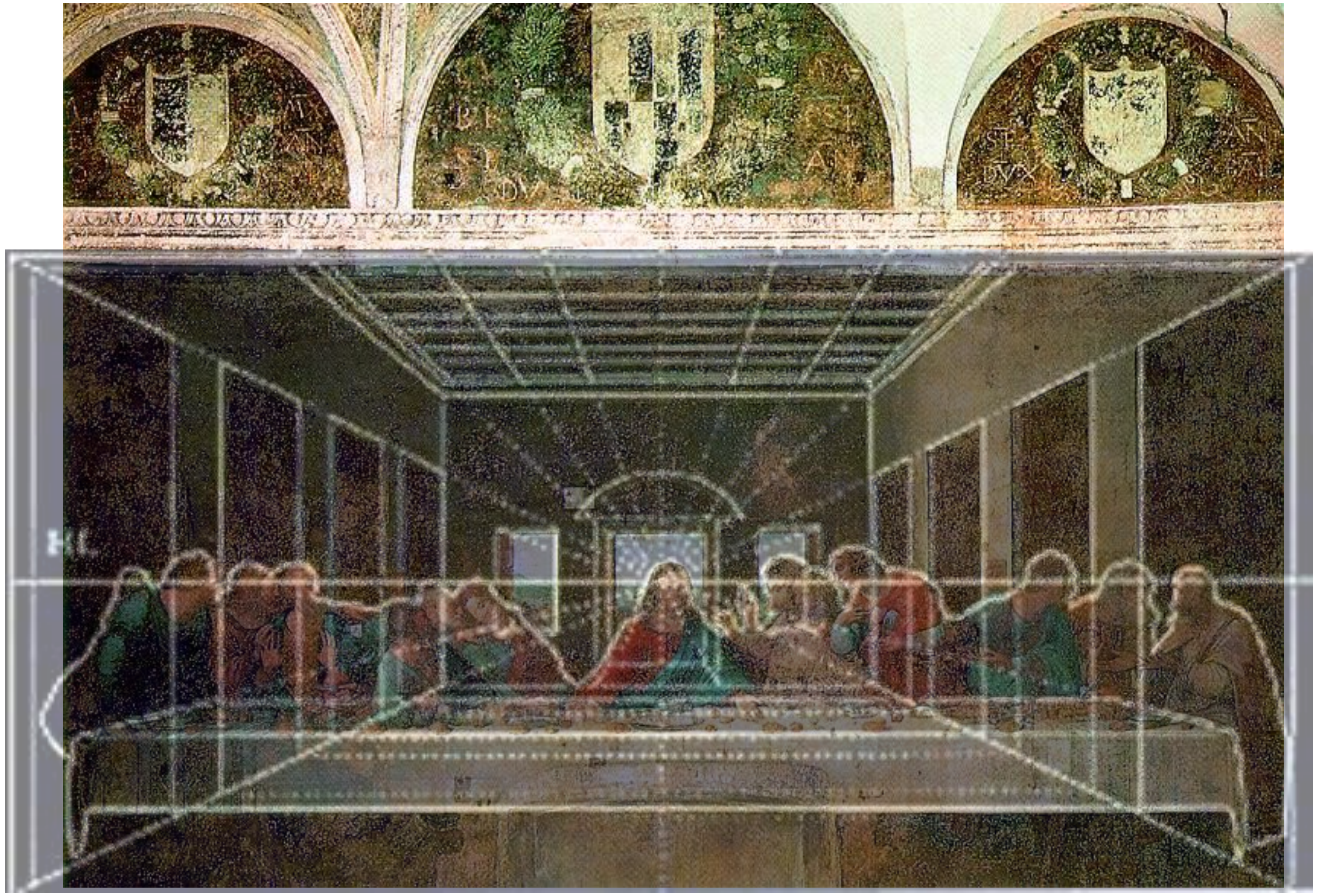
Alberti on Linear Perspective (View Points)

- “A painting **[the projection plane]** is the intersection of a visual pyramid **[view volume]** at a given distance, with a fixed center **[center of projection]** and a defined position of light, represented by art with lines and colors on a given surface **[the rendering]**.”
  - (Leono Battista Alberti (1404-1472), *On Painting*, pp. 32-33)
- A different way of perceiving perspective from a vanishing point



History and Overview of Projections

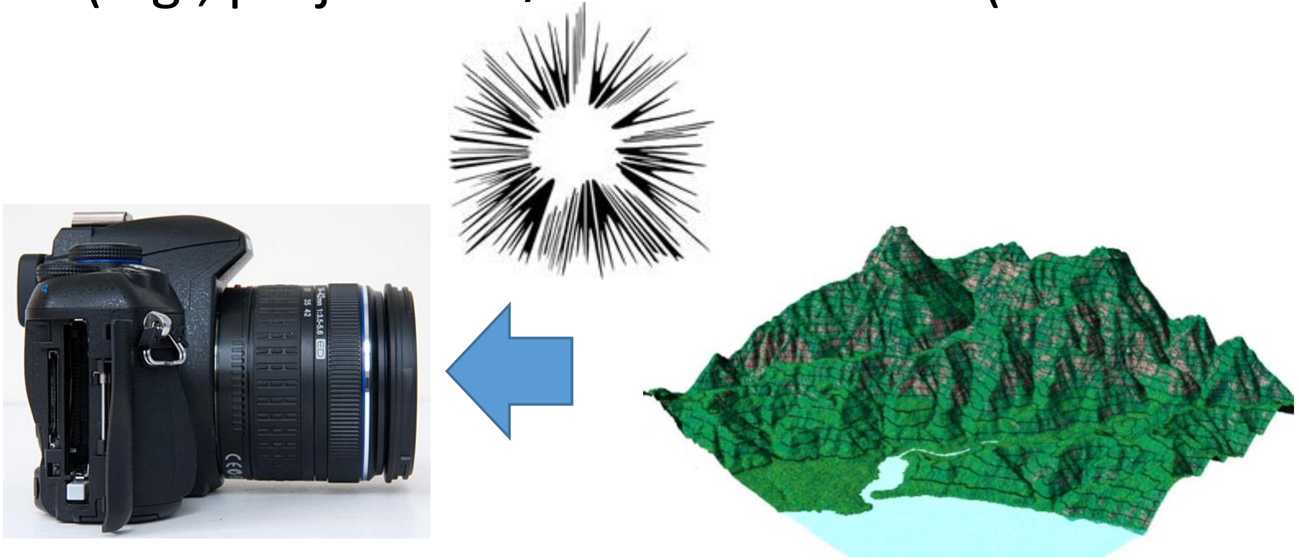
Duccio c. 1308



Leonardo da Vinci, *The Last Supper* (1495)

The camera and the scene

- What does a camera do?
- Operates in a three-dimensional world
  - He takes a scene (e.g., projectes it) to a 2D medium (on film or a screen)



- The **synthetic camera** is a programmer’s model for specifying how a 3D scene is projected onto the screen

3D projection: the synthetic camera

- Synthetic camera
  - Camera position
  - Orientation
  - Field of view (angle of view, wide, narrow, normal...)
  - Depth of field (field of view: near distance, far distance)
  - Focal distance
  - Tilt of view/ film plane
  - Perspective or parallel projection (camera in scene with objects or infinite distance away, resp.)

Transform from 3D to 2D

- Viewing is only part of the process of converting from 3D to a 2D image

An allegory with painting

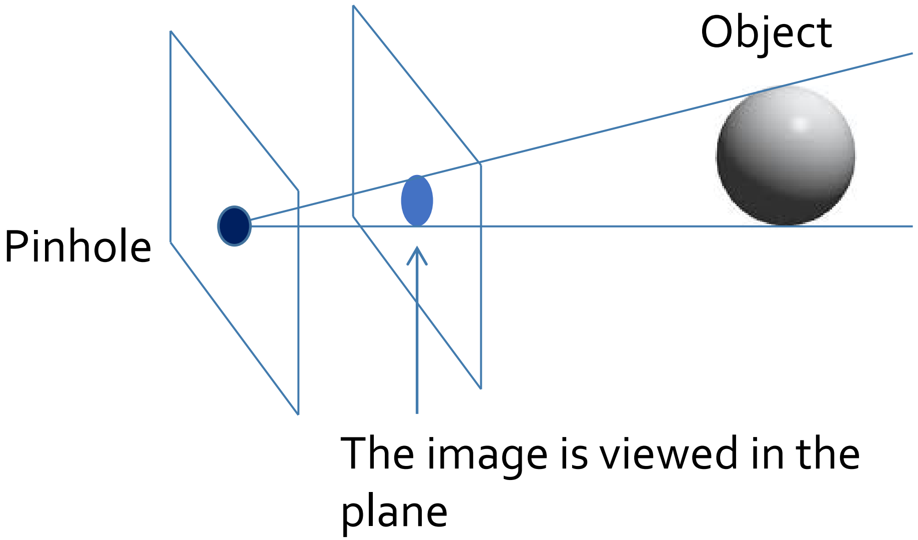
Albrecht Durer 1525

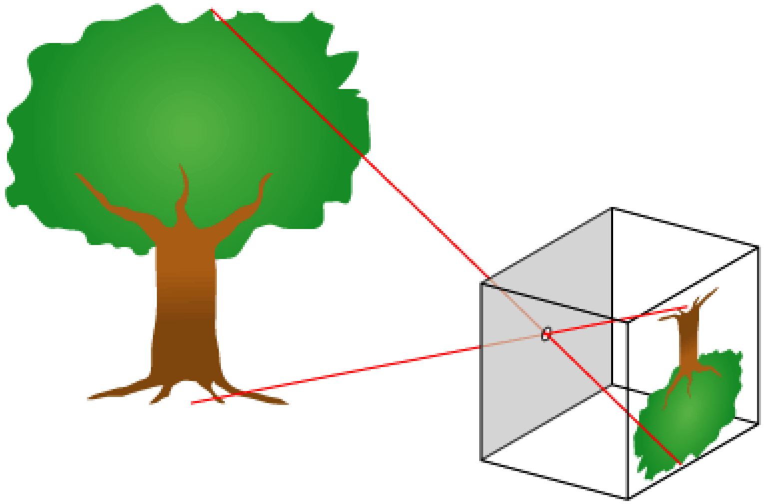
Image from Hitachi's Viewseum at http://www.viewseum.com/

- We have an artist who looks through a frame and carries what he sees on the canvas
- The frame consists of a grid with squares, and the canvas itself
  - equal in number but not necessarily in size
  - (*scene pixels, screen pixels*)
- Conditions/problems
  - Quality, movement of the head, process is instantaneous

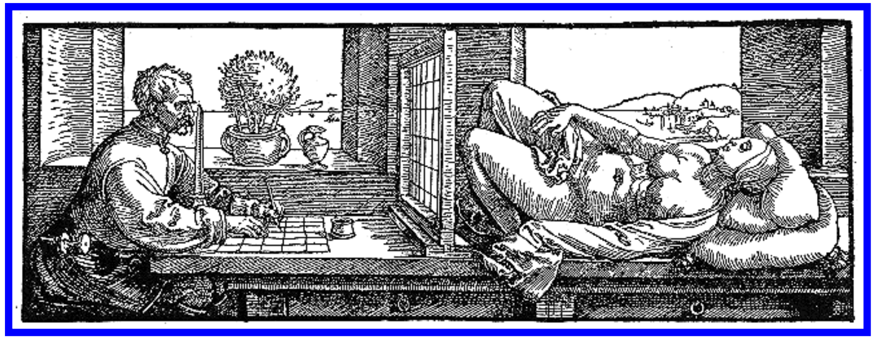
Pinhole Model for Perspective Projection

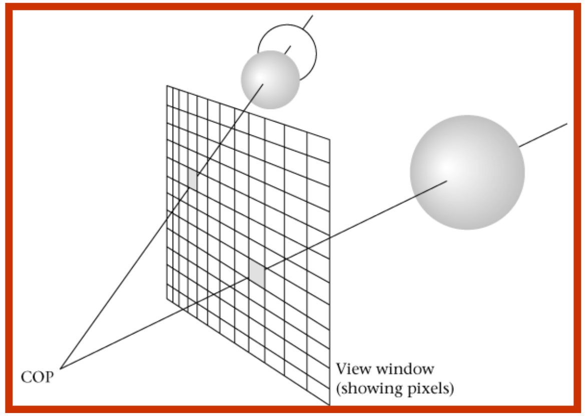
- Pinhole camera projects a volume of space (the scene) on a film plane or wall behind the pinhole. Rays of light reflect off objects and rays that are inside the view volume (thus visible from camera's POV) converge to the pinhole. The scene will be inverted!
- Pinhole is our camera **position** (“center of projection”, apex of the frustum), and **view volume** is what we see, i.e., “is in view”
- In our camera, camera’s projectors intersect a plane, usually in between scene and pinhole, projecting the scene onto that plane (no inversion)
- Lastly, in our camera, projection is mapped to some viewing medium (e.g., screen)
- For practical rendering, add front and back clipping planes





The 3 Elements of the allegory





A. The scene

the girl — the spheres

B. The view

the eye & the tilar — a simple camera

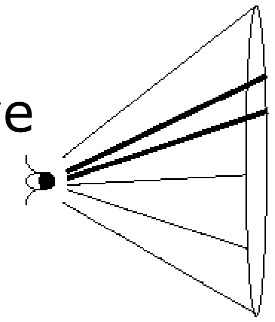
C. The rendering process

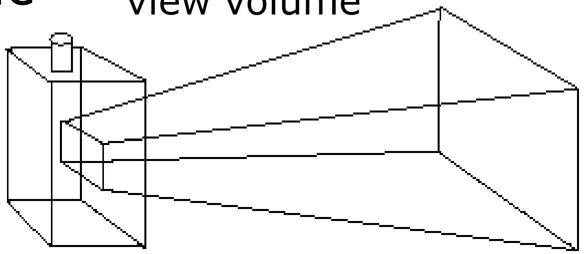
painting — ray casting

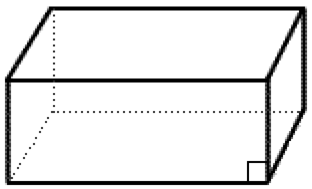
i. determination of the pixel's color
ii. painting the pixel on the screen

View Volumes

- A **view volume** contains everything the camera sees
- In the 3D view, a rectangular view window is used.
- **Conical** – Approximates what eyes see, expensive math when clipping objects against cone's surface
- You can approach it though using a rectangle.
- There are also parallel views. These do not, however, simulate the eye or the camera.

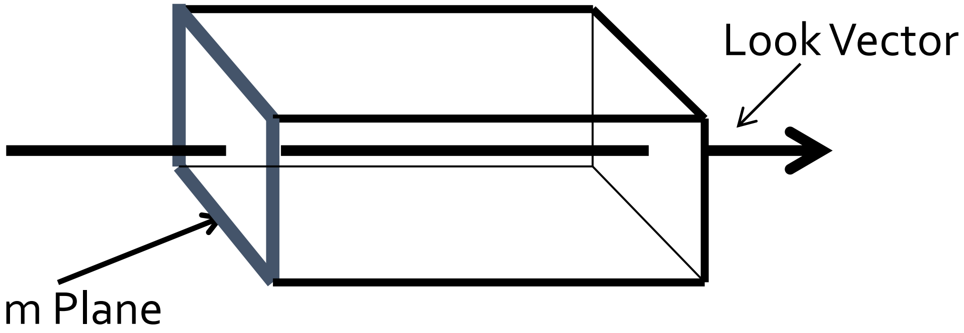


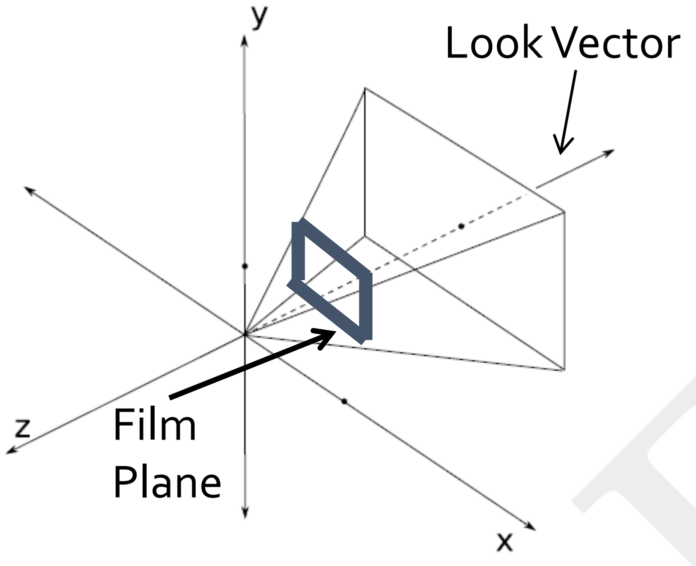




The View Plane (aka Projection Plane, Film Plane)

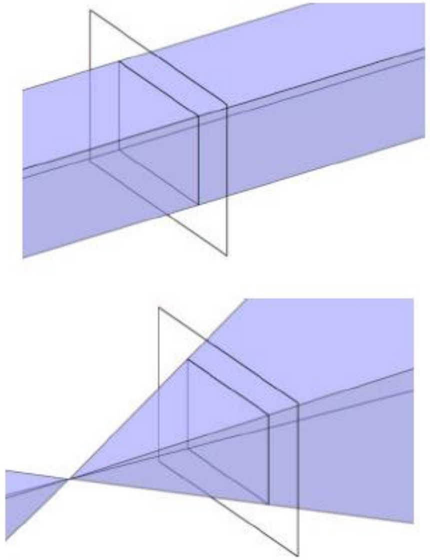
- View plane is in *world space* – 3D scene is projected onto a rectangle (view plane) on that plane via some projection transformation and from there onto the viewport on screen via window



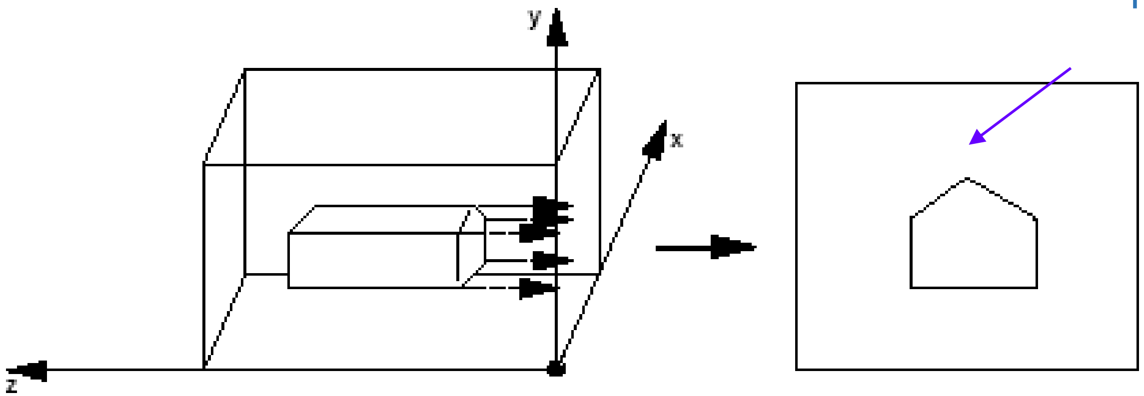


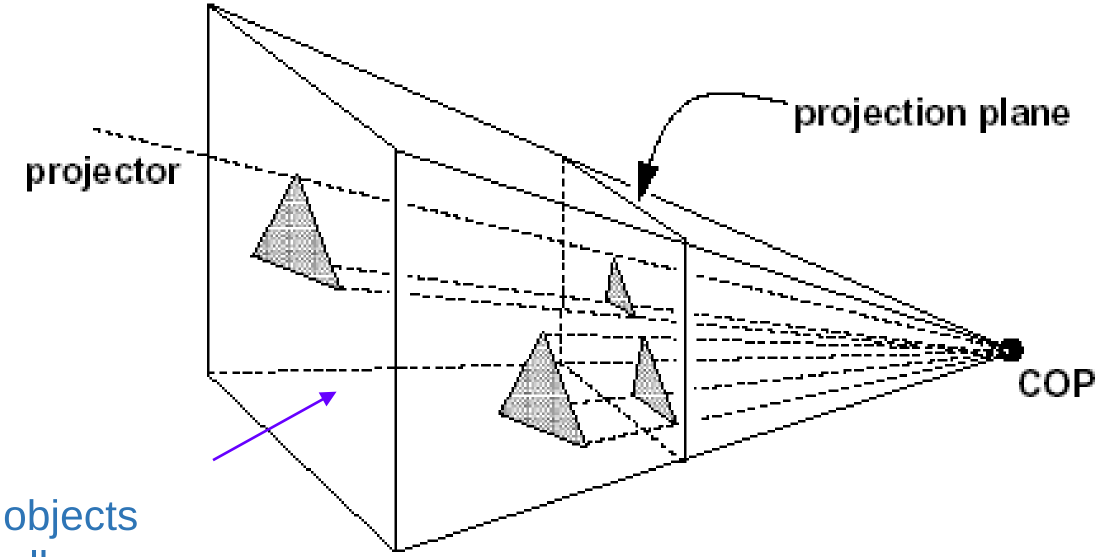
View Volumes and Projectors

- **Projectors**: Lines that map points in scene to points on film plane
  - **Parallel Volumes:** Projectors are parallel to the look vector (the direction in which the camera is looking). No matter how far away an object is, it will appear as the same size as long as it is in the view volume.
  - **Perspective Volumes**: Projectors emanate from eye point (center of projection), the inverse of rays of light converging to your eye



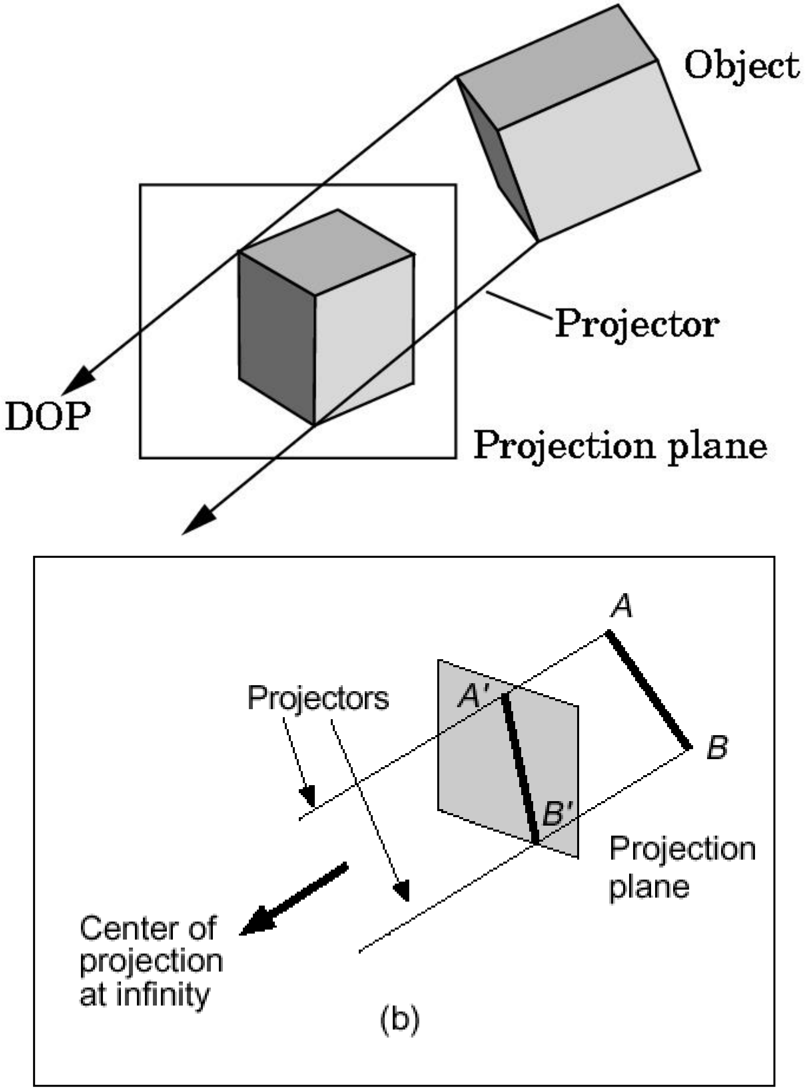
Parallel lines remain parallel





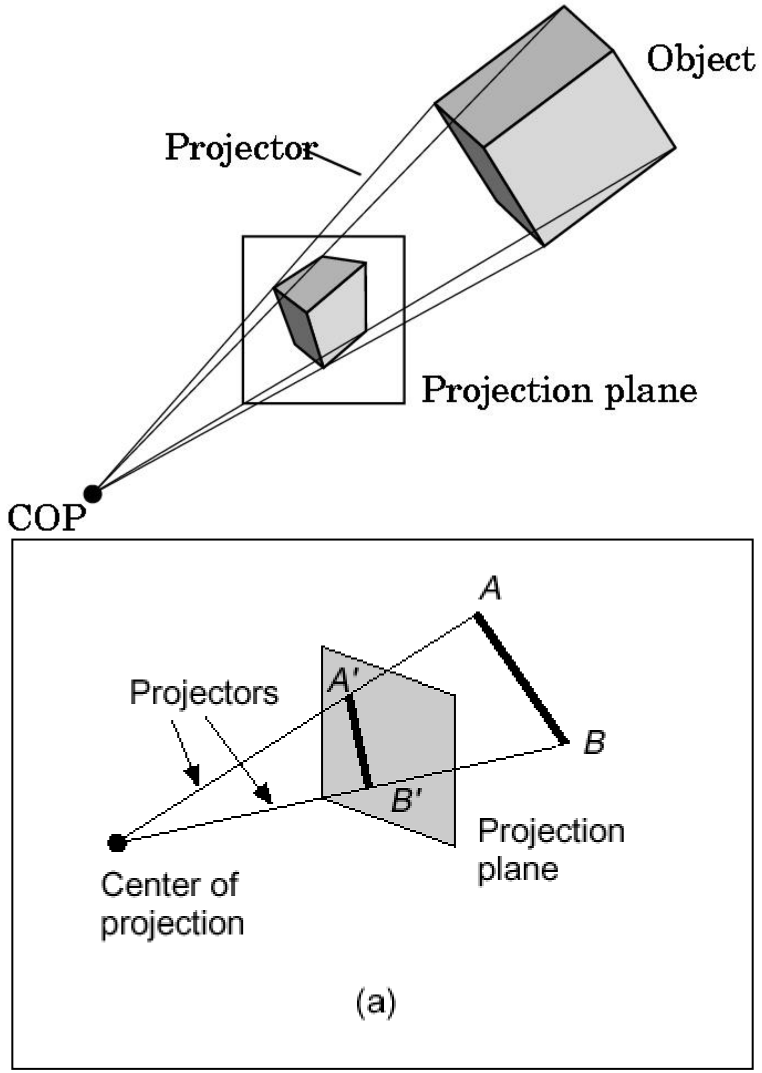
Distant objects look smaller

Projections



Parallel Volumes

Usually used for architectural and mechanical drawings



Perspective Volumes

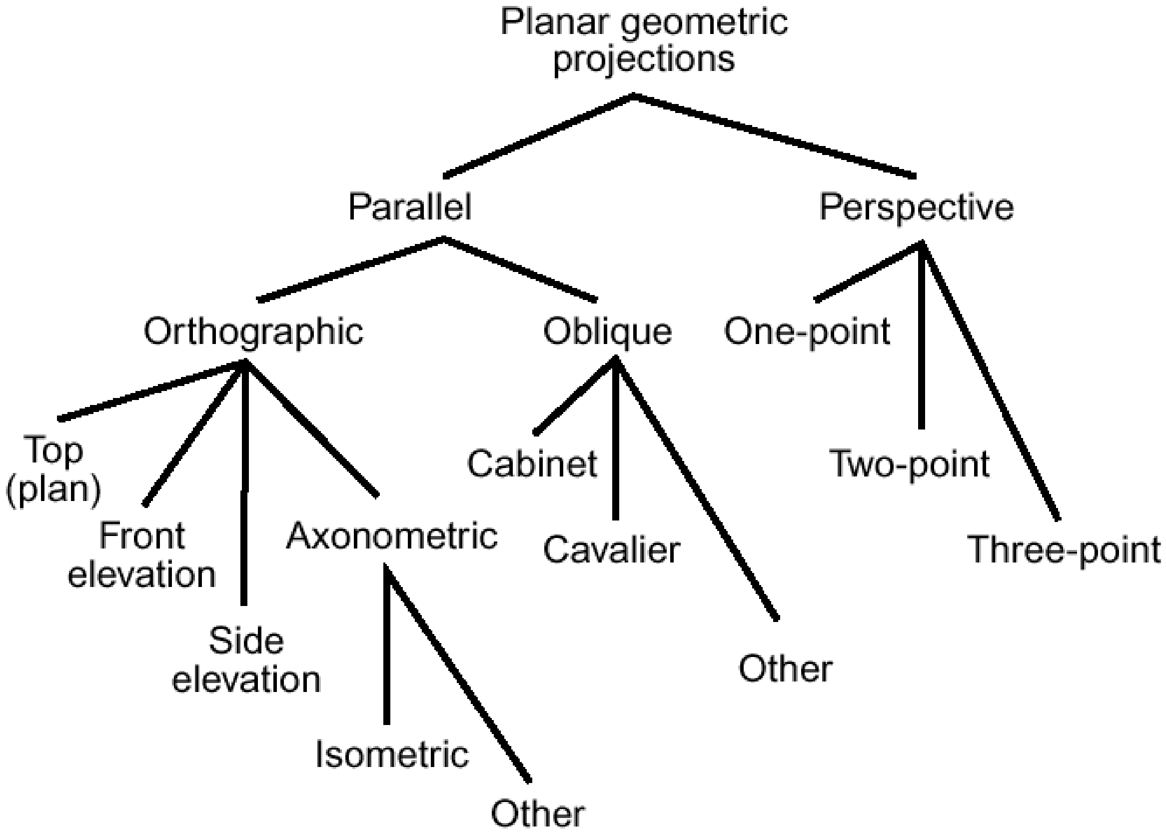
More realistic, usually used in graphics

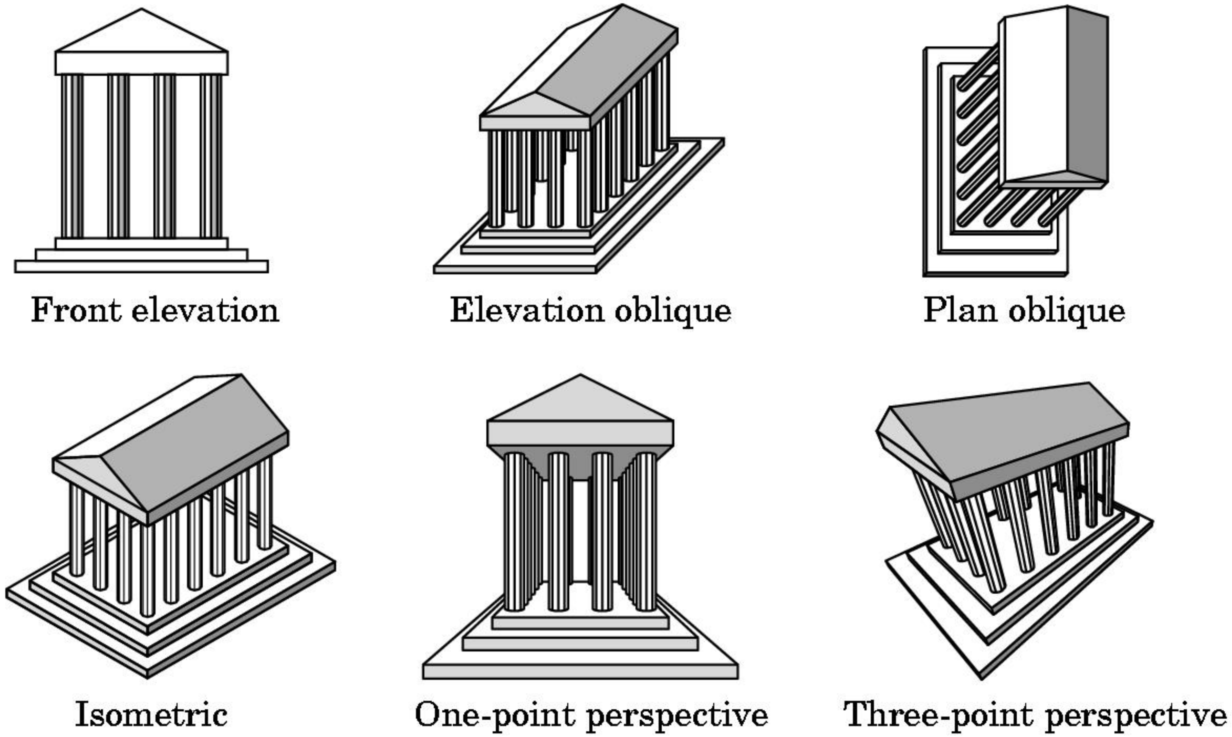
Parallel projection

Projection with perspective

Types of projections

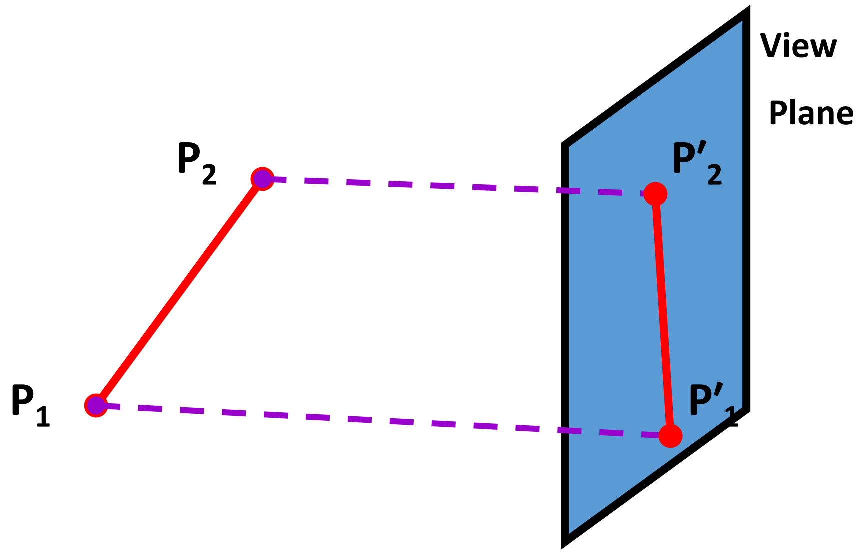
- For those who have study architecture





Parallel projection

Position coordinates are displayed on the plane using parallel lines.



Perpendicular projection

The projection is perpendicular to the viewing plane.

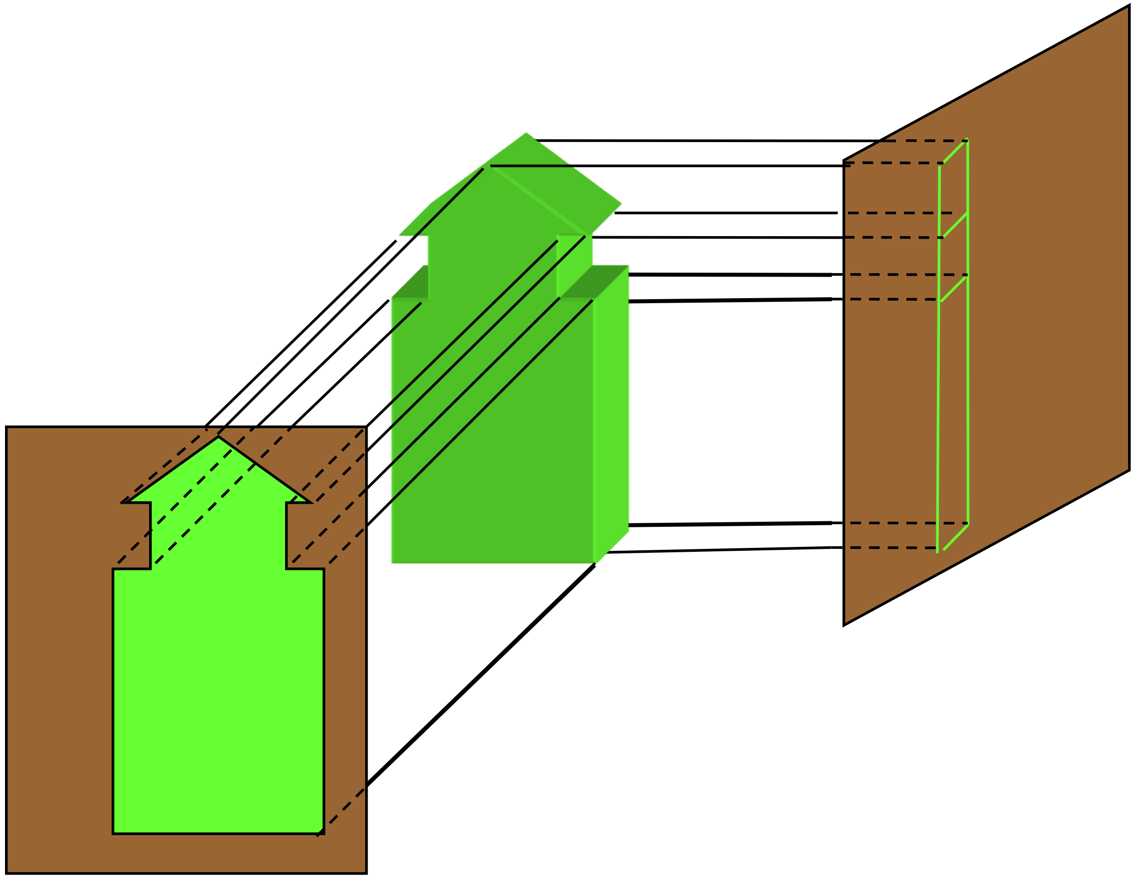
Not Perpendicular projection

The parallel projection is not perpendicular to the viewing plane.

Perpendicular projection

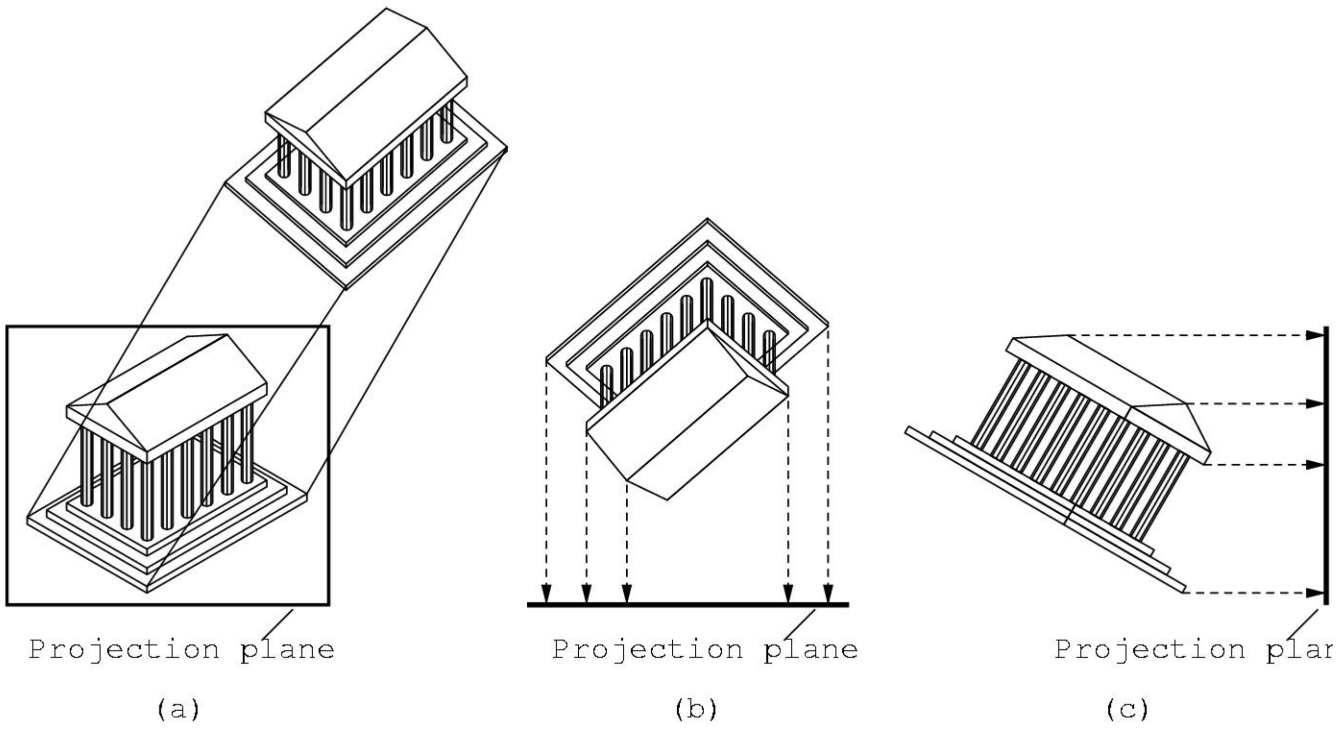
The transformation

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$



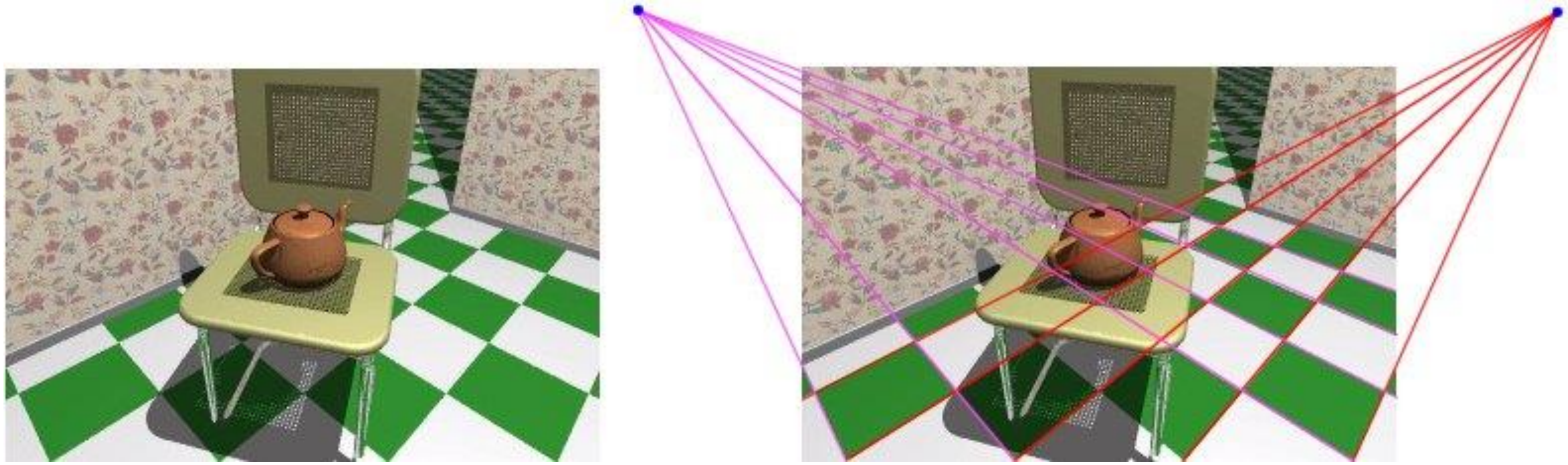
Not Perpendicular projection

- Projections are still perpendicular to the viewing plane
- But the projection plane may have different orientation relative to the object.
- It is used extensively in architectural and mechanical design.
-



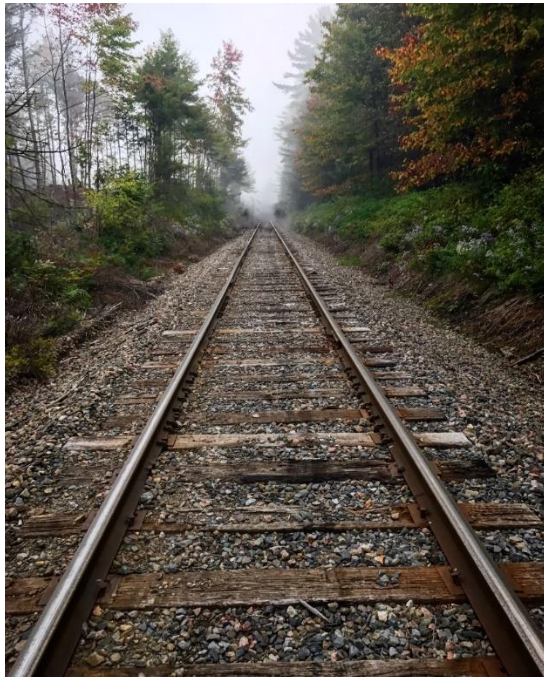
Projection with perspective

- It was discovered by Donatello, Brunelleschi, and da Vinci during the Renaissance period
- Objects that are closer to the viewer appear larger
- The parallel lines seem to converge on a single point
- It gives a realistic view and feeling about the three-dimensional form of the object
- Does not preserve the object's shape or size (except where the object intersects the viewing layer)
- Different from a parallel view because
  - Parallel lines become non-parallel based on the level of view they converge
  - the size of the object decreases with the distance
  - The perspective is not uniform

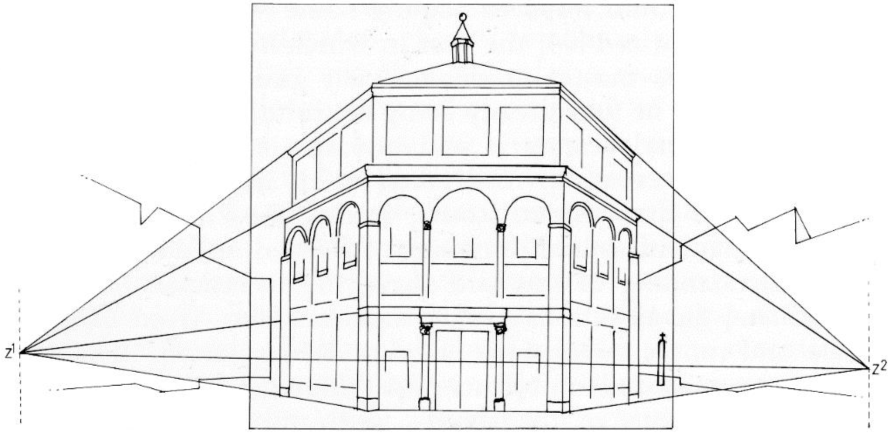


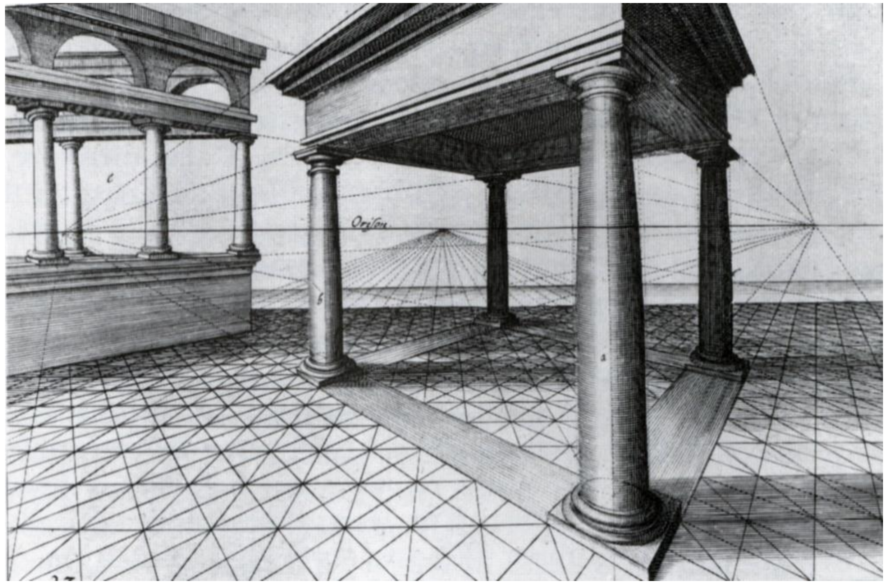
Parallel Projection Vs Projection with Perspective

- Projection with perspective creates images that are much more realistic than parallel projections
-



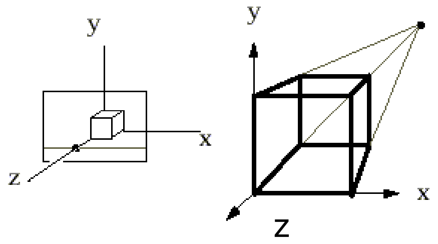
If we saw this scene using a parallel view, the rails would not converge



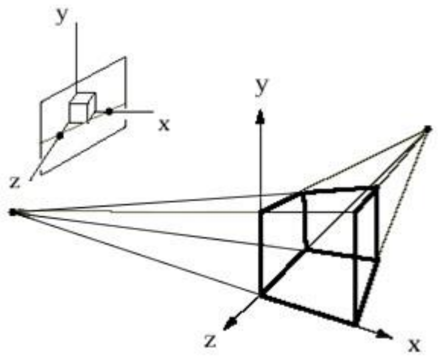


Projectionwith perspective

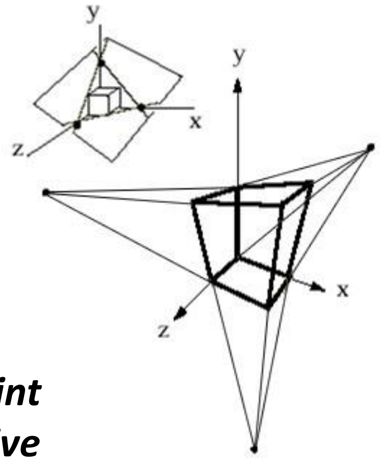
- Lines extending from edges converge to common vanishing point(s)
- For right-angled forms whose face normals line up with the *x, y, z* coordinate axes, number of vanishing points equals number of coordinate axes intersected by projection plane
  - There are different kinds of view perspectives



One Point Perspective
(*z*-axis vanishing point)



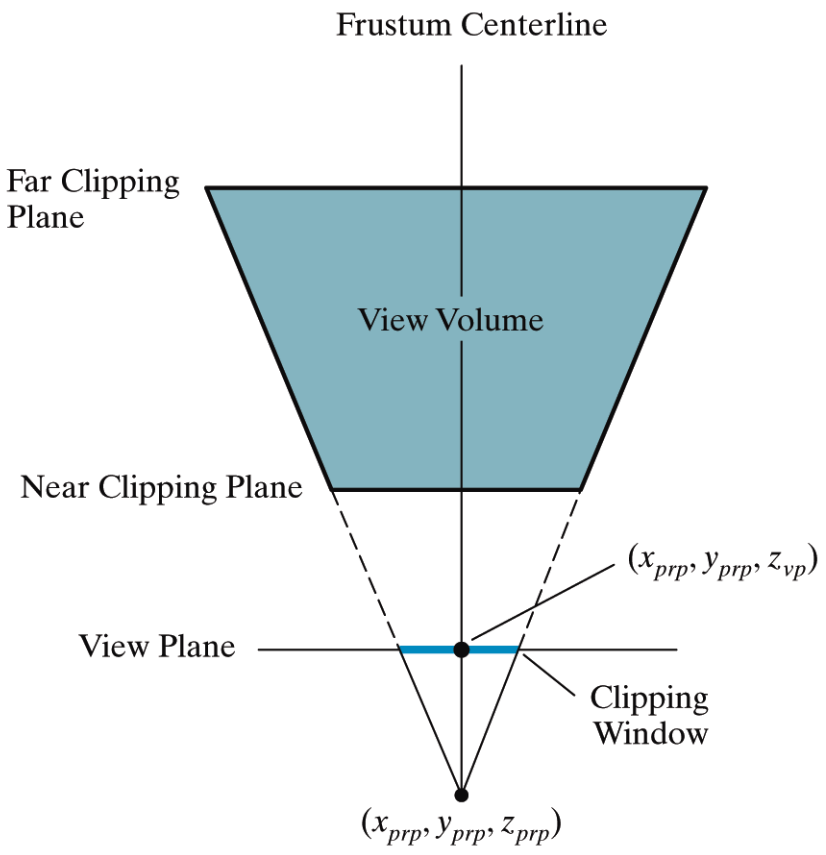
Two Point Perspective
(*z* and *x*-axis vanishing points)



Three Point Perspective
(*z*, *x*, and *y*-axis vanishing points)

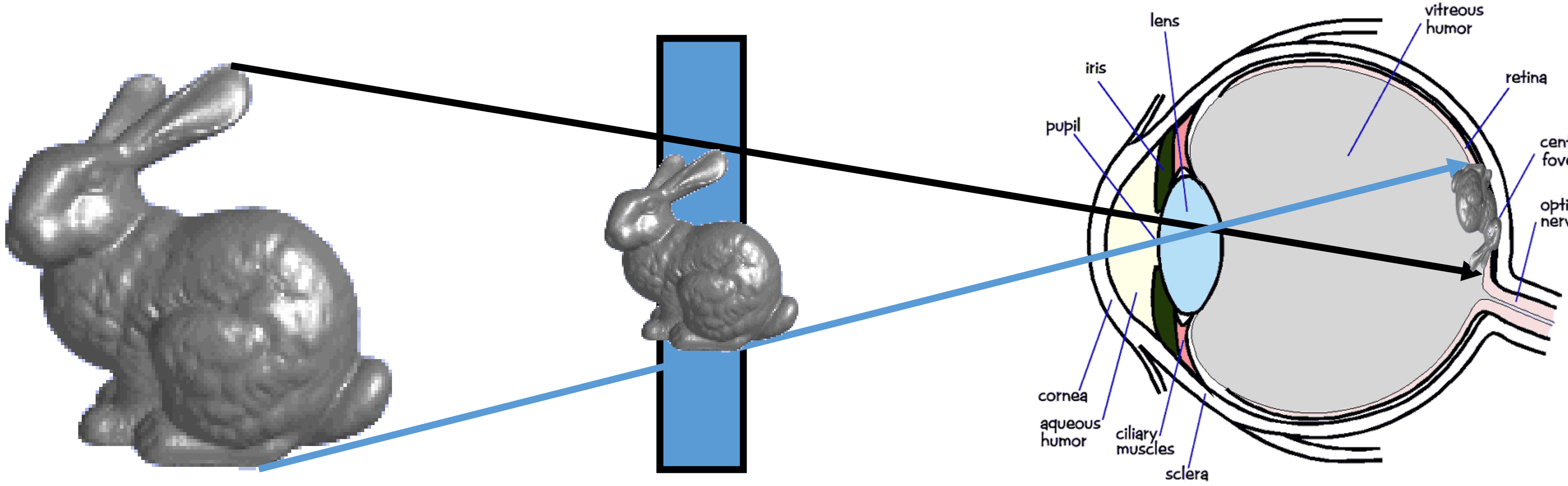
Projection with perspective

- A perspective projection can be set up by specifying the position and size of the view plane and the position of the projection reference point
- However, this can be kind of awkward



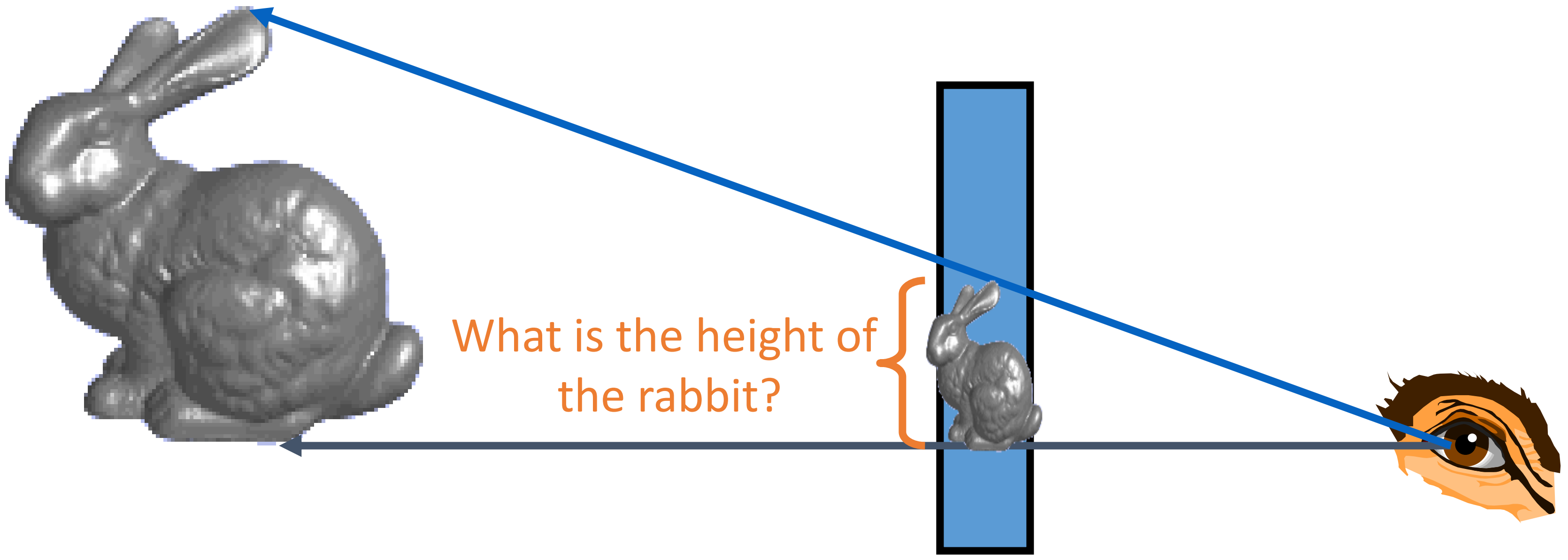
Projection with perspective

- Objects that are far away appear smaller



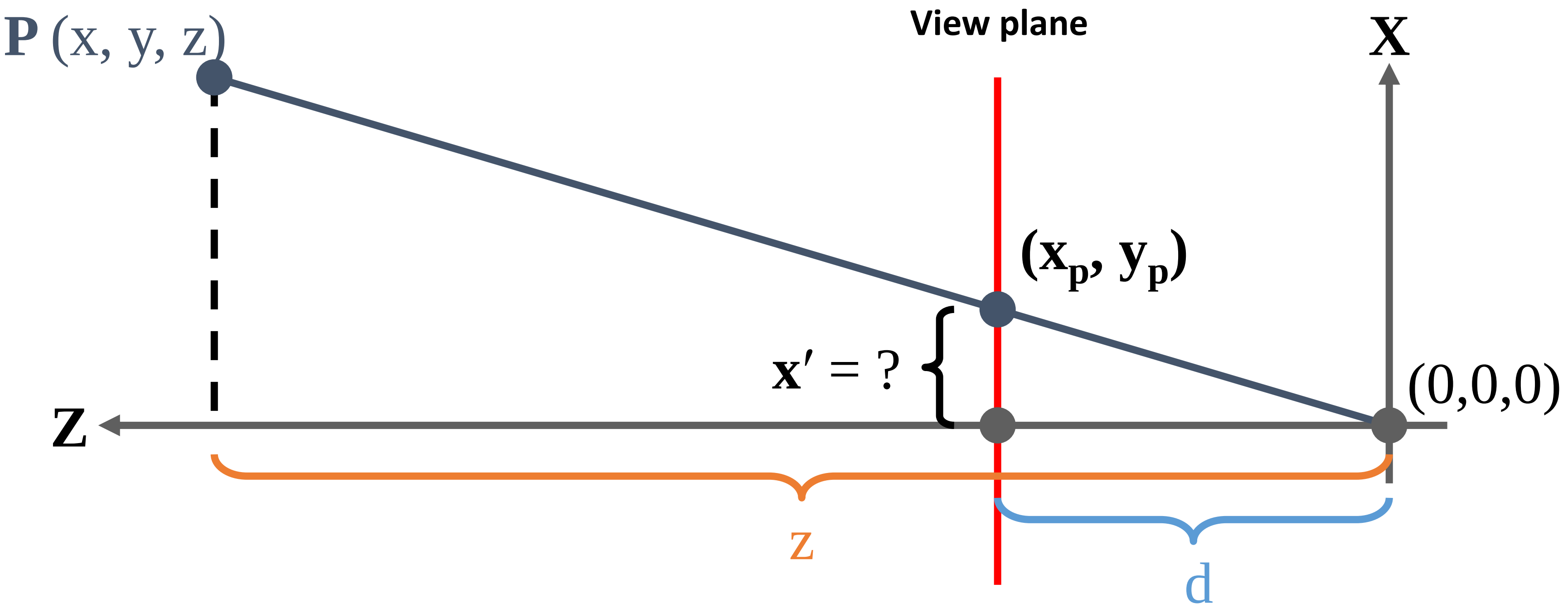
Projection with perspective

- In 3D graphics, we think of the screen as a 2D window on the 3D world



Projection with perspective

What is the height of the rabbit?



Projection with perspective

The result of the projection of $[x, y, z, 1]^T$ at the plane is:

$$\frac{x'}{d} = \frac{x}{z}, \quad \frac{y'}{d} = \frac{y}{z}$$

$$x' = \frac{d \cdot x}{z} = \frac{x}{z/d}, \quad y' = \frac{d \cdot y}{z} = \frac{y}{z/d}, \quad z' = d$$

Projection with perspective

$$\begin{bmatrix} x_h \\ y_h \\ z_h \\ h \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/d & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

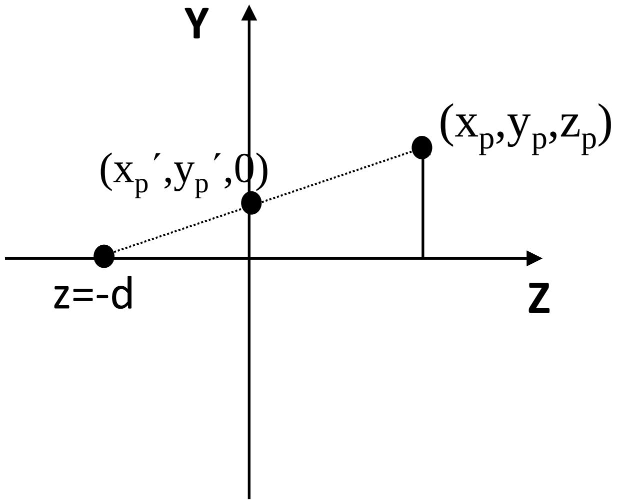
$$h = z / d$$

$$x_p = x_h / h, \qquad y_p = y_h / h$$

Projection with perspective

Case where the level of view is in the (0,0,0)

- To have a natural image we need **perspective**
- For a simple layout it is easy to find the view (using identical triangles)



$$\frac{y_p{}'}{d} = \frac{y_p}{z_p + d} \Rightarrow y_p{}' = y_p \frac{d}{z_p + d}$$

$$\frac{x_p{}'}{d} = \frac{x_p}{z_p + d} \Rightarrow x_p{}' = x_p \frac{d}{z_p + d}$$

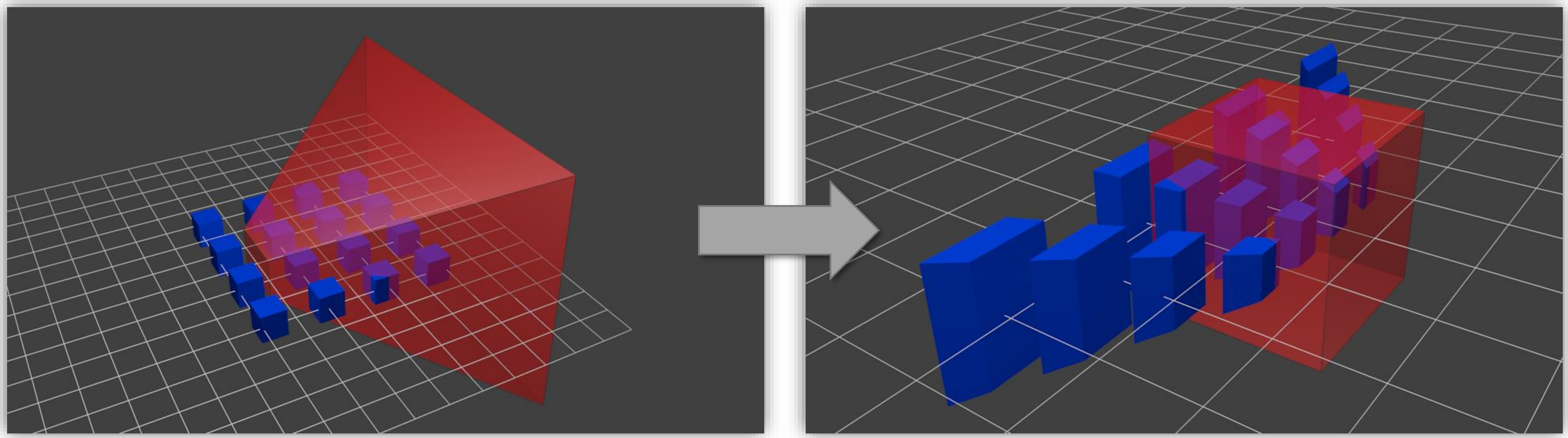
- Note that as the cop goes to infinity, the perspective becomes equivalent to a parallel projection

Camera tilted up: converging vertical lines (3 vanishing points)

[Philip Greenspun]

Lens shifted up: parallel vertical lines (3 vanishing points)

General Perspective to Canonical Parallel



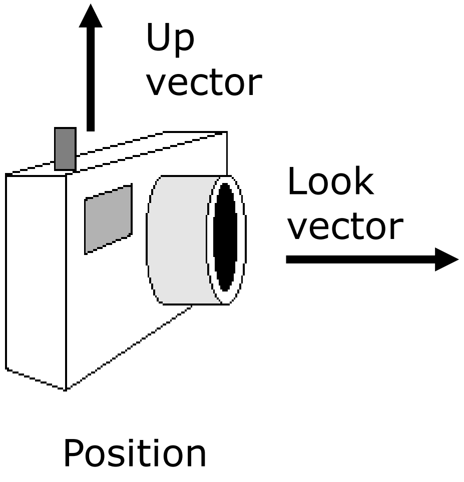
Viewport

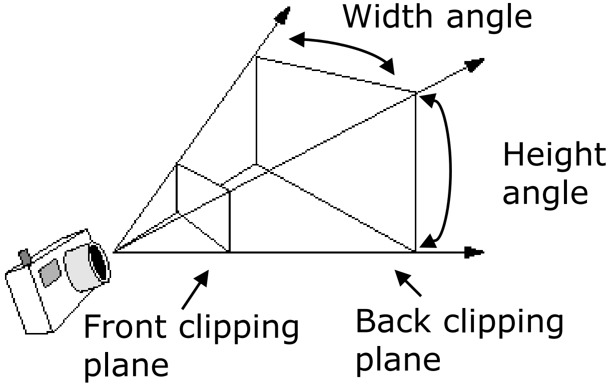
- **Viewport is a rectangular area of the screen where the scene is rendered**
  - pixel coordinates for viewing usually refer to the use of the coordinate system (u,v)

Setting the field of view

We need to know six parameters (think degrees of freedom) about our synthetic camera model in order to take a picture using our perspective view **frustum:**

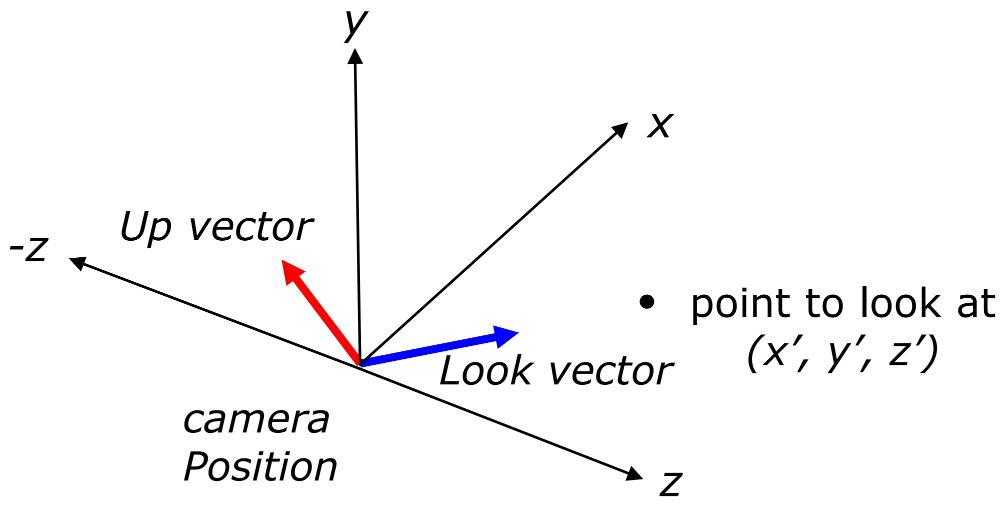
1) *Position of the camera*
2) *Look vector* specifying the direction of the camera
3) Camera's orientation determined by **look vector** and angle by which the camera is rotated about that vector, i.e., the direction of the Up vector in xyz World Coordinate System (WCS). It will be specified by an **Up vector** that needn't be perpendicular to Look, i.e., lie in the plane defined by Look as its normal
4) Aspect ratio
5) Height angle: determines how much of the scene we will fit into our view volume; larger height angles fit more of the scene into the view volume (width angle determined by height angle and aspect ratio)
   - the greater the angle, the greater the amount of perspective distortion
6) Front and back clipping planes: limit extent of camera's view by rendering (parts of) objects lying between them and ***clipping*** everything outside of them – avoids problem of having far-away details map onto same pixel, another kind of aliasing, i.e., **sampling error**





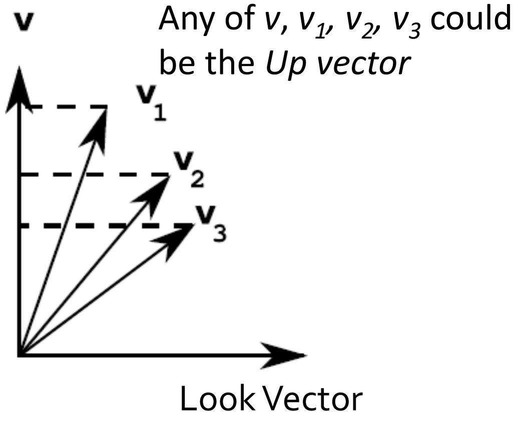
Orientation: *Look* and *Up* vectors

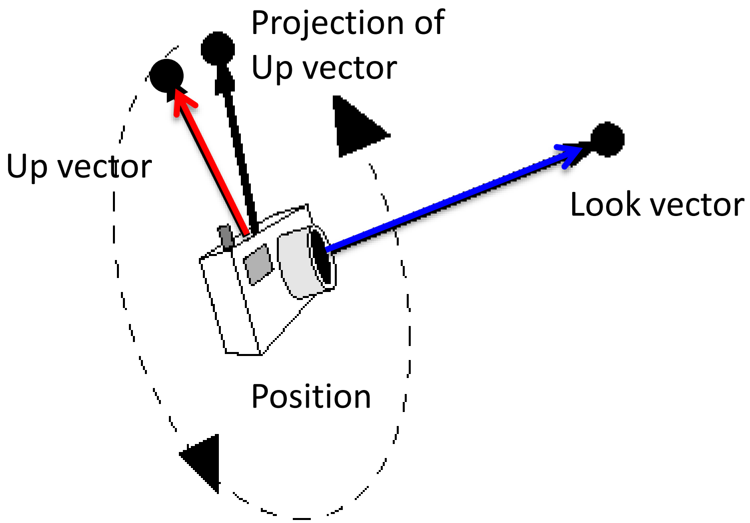
1. **Position:** Where is the camera located with respect to the origin, in the world coordinate system?
2. **Orientation** Orientation is specified by a direction to ***look*** in (equivalently, a point in 3D space to look at) and a non-collinear vector ***Up*** from which we derive the rotation of the camera about ***Look***



Orientation: Look and Up vectors

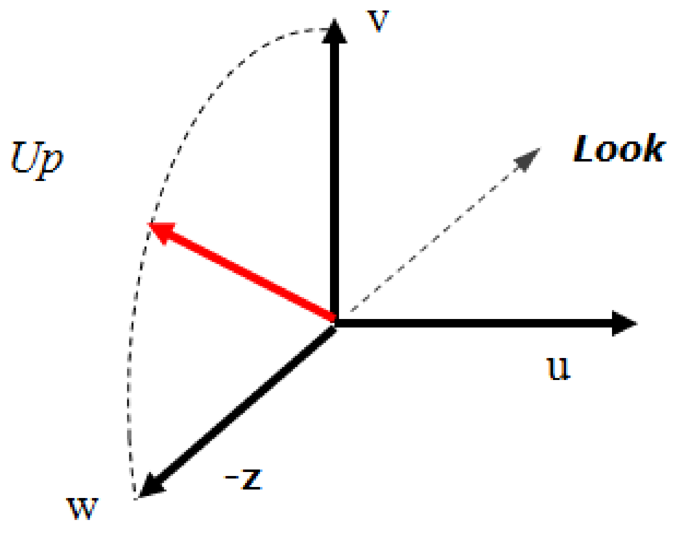
- ***Look Vector***
  - Direction the camera is pointing
  - Three degrees of freedom; can be any vector in 3D-space
- ***Up Vector***
  - Determines how camera is rotated about look
  - For example, holding camera in portrait or landscape mode
  - up must not be co-linear to look but it doesn't have to be perpendicular– actual orientation will be defined by the unit vector v perpendicular to look in the plane defined by look and up
    - easier to spec an arbitrary (non-collinear) vector than one perpendicular to look





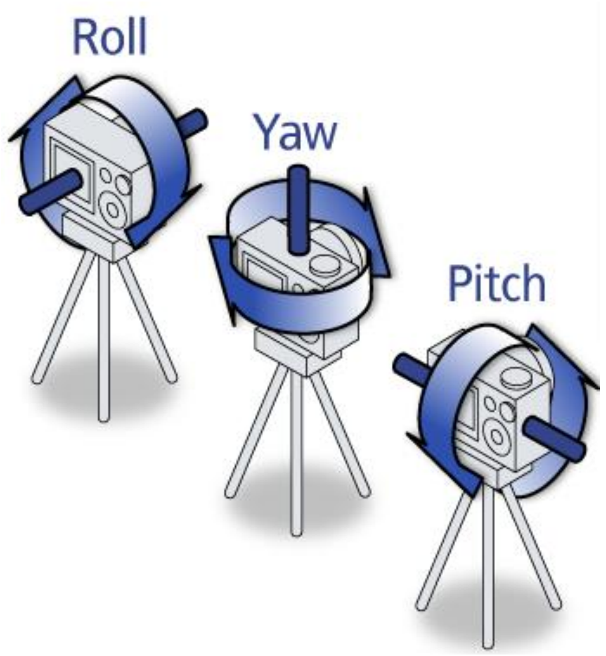
The Camera Coordinate Space

- The equivalent of x, y and z WCS axes in camera space are unit vectors u, v, w (u, v, not to be confused with texture axes, w is the homogenous coordinate)
  - Also a right handed coordinate system
  - w is a unit vector in the opposite direction of look (i.e. look lies along the –-w axis)
  - v is the component of the up vector perpendicular to look, normalized to unit length
  - u is the unit vector perpendicular to both v and w



Three common rotation transformations that use camera space axes, with camera at arbitrary position

- Roll: Rotating your camera around ***w***
- Yaw: Rotating your camera around ***v***
- Pitch: Rotating your camera around ***u***



Aspect Ratio

- Analogous to dimensions of film in camera
- Ratio of width to height of viewing window
- Viewport's aspect ratio usually defined by screen being used , or by common ratio's, e.g., 16:9
  - Square viewing window has a ratio of 1:1
  - NTSC TV is 4:3, HDTV is 16:9 or 16:10
- Aspect ratio of viewing window defines dimensions of the image that gets projected to view plane, after which it is mapped to viewport
  - Typically it's a good idea to have same aspect ratio for both viewing window and viewport, to avoid distortions/stretching
  - Note: the black strips on the 16:9 image is a technique called letter boxing. It preserves the aspect ratio of the image when the screen can't accommodate it. This is in contrast to simply stretching the image which distorts the images (most notably, faces)

1:1

Courtesy of http://www3.flickr.com/photos/zerogrizzly/4037144257/

4:3

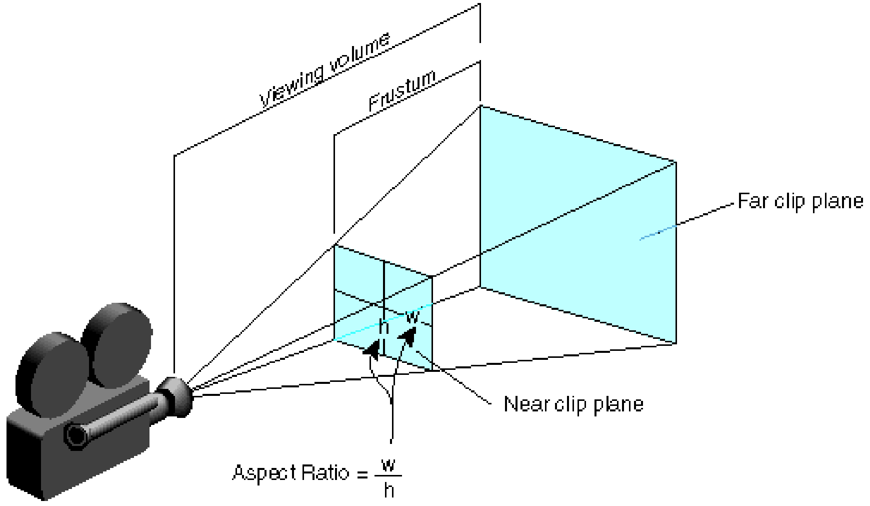
Courtesy of http://en.wikipedia.org/wiki/File:Aspect_ratio_4_3_example.jpg

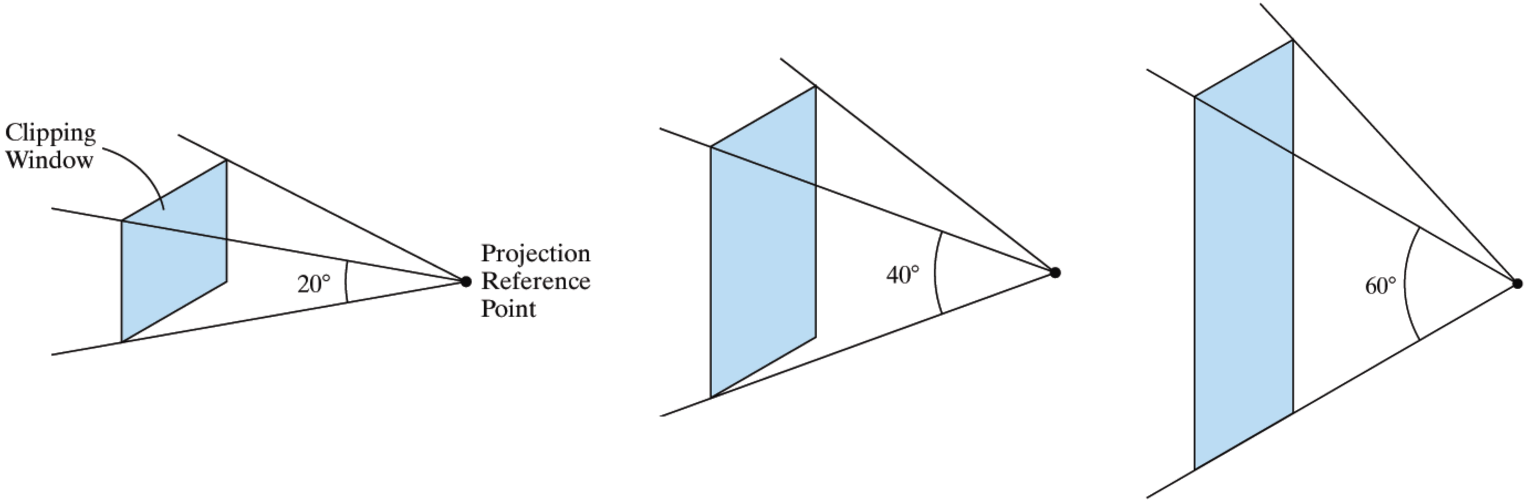
Courtesy of http://forum.videohelp.com/threads/236536-720-vs-704

16:9

Viewing Angle

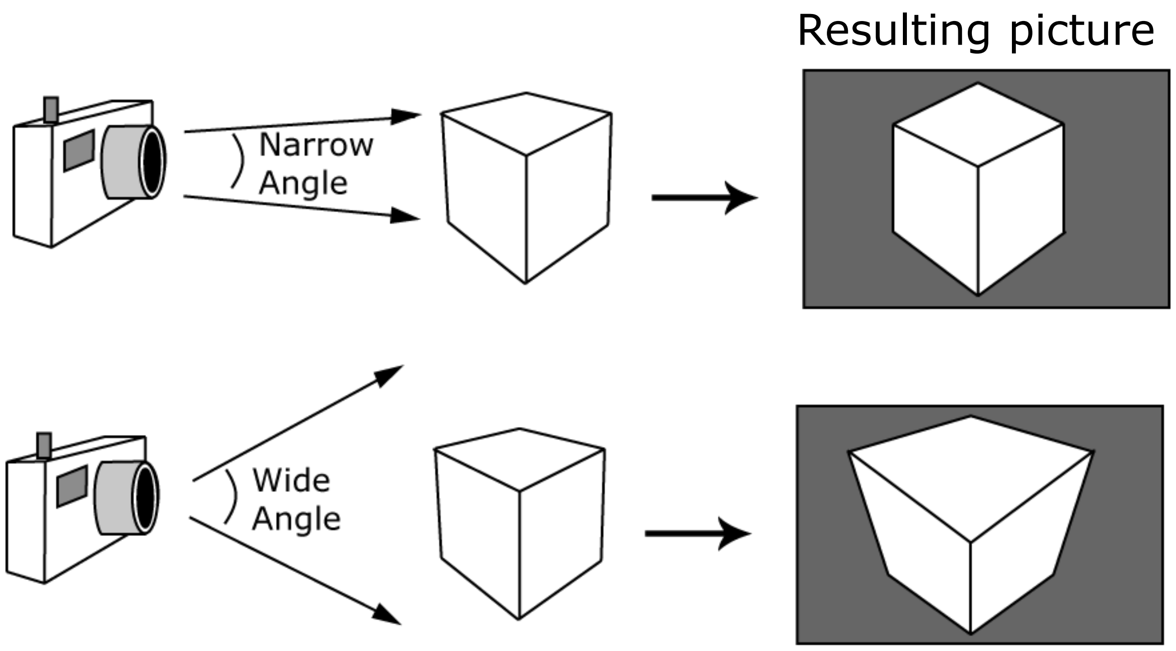
- Determines amount of perspective distortion in picture, from none (parallel projection) to a lot (wide-angle lens)
- In a frustum, two viewing angles: width and height angles
  - Usually width angle is specified using height angle and aspect ratio
- Choosing *view angle* is analogous to photographer choosing a specific type of lens (e.g., a wide-angle or telephoto lens)





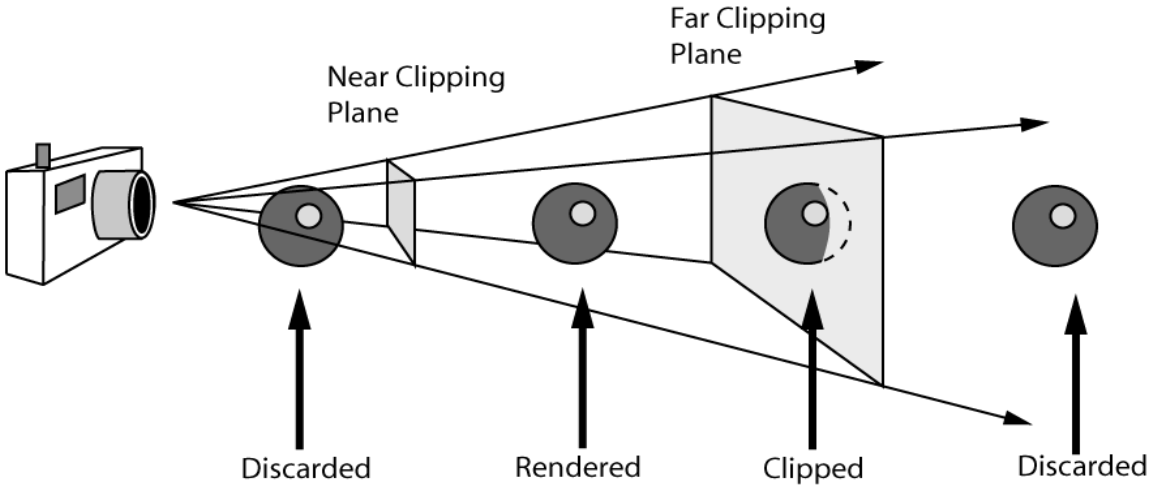
Viewing Angle

- Telephoto lenses made for distance shots often have a nearly parallel viewing angle and cause little perspective distortion, though they foreshorten depth (condense space)
- Wide-angle lenses cause a lot of perspective distortion (expand space)



Near and Far Clipping Planes

- With what we have so far we can define four rays extending to infinity. These define the edges of our current view volume.
- Now we need to bound front and back to make a finite volume – can do this using the *near* and *far* clipping planes, defined by distances along ***look*** (also note that ***look*** and clipping planes are perpendicular)



- This volume (the frustum) defines what we can see in the scene
- Objects outside are discarded
- Objects intersecting faces of the volume are “clipped”

Αποκοπή επιπέδων (Near and Far Clipping Planes)

- **Reasons for back (far) clipping plane::**
  - Usually don’t want to draw things too close to camera
    - Would block view of rest of scene
    - Objects would be quite distorted
  - Don’t want to draw things behind camera
    - Wouldn’t expect to see things behind camera
    - In the case of perspective camera, if we were to draw things behind camera, they would appear upside-down and inside-out because of perspective transformation
- **Reasons for back (far) clipping plane:**
  - Don’t want to draw objects too far away from camera
    - Distant objects may appear too small to be visually significant, but still take long time to render; different parts of an object may map onto same pixel (sampling error)
    - By discarding them we lose a small amount of detail but reclaim a lot of rendering time
    - Helps to declutter a scene
- These planes need to be properly placed, not too close to the camera, not too far
- More information and ways for clipping, in the next lesson!

Focal Length

- Some camera models take a *focal length*
- *Focal length* is a measure of ideal focusing range; approximates behavior of real camera lens
- Objects at distance equal to *focal length* from came farth
- *Focal* plane
    - Only outside of view volume still get discarded

In the simplified graphics that we will see in this course we will not deal at all with this effect.
We assume that the our idealized camera has no lens and its opening is the size of a mathematical point, therefore in the standard pipeline everything looks sharp.