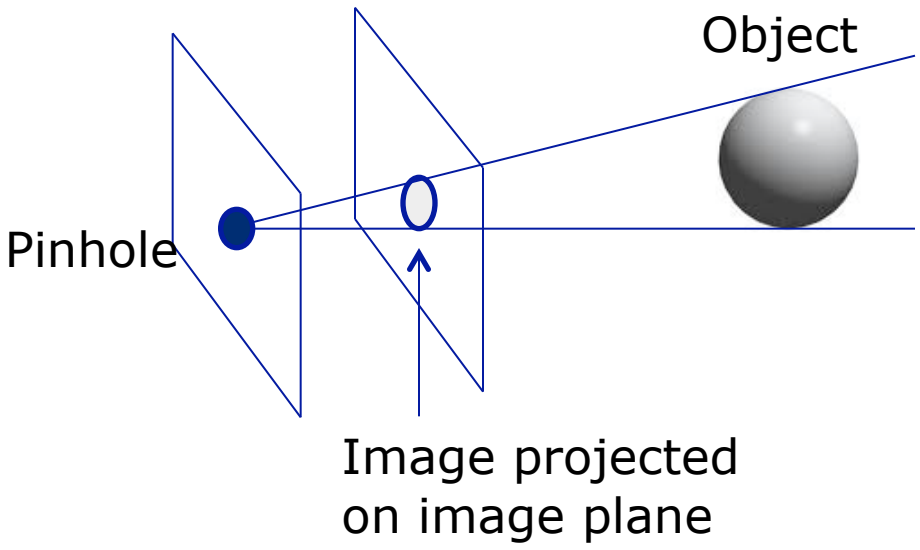
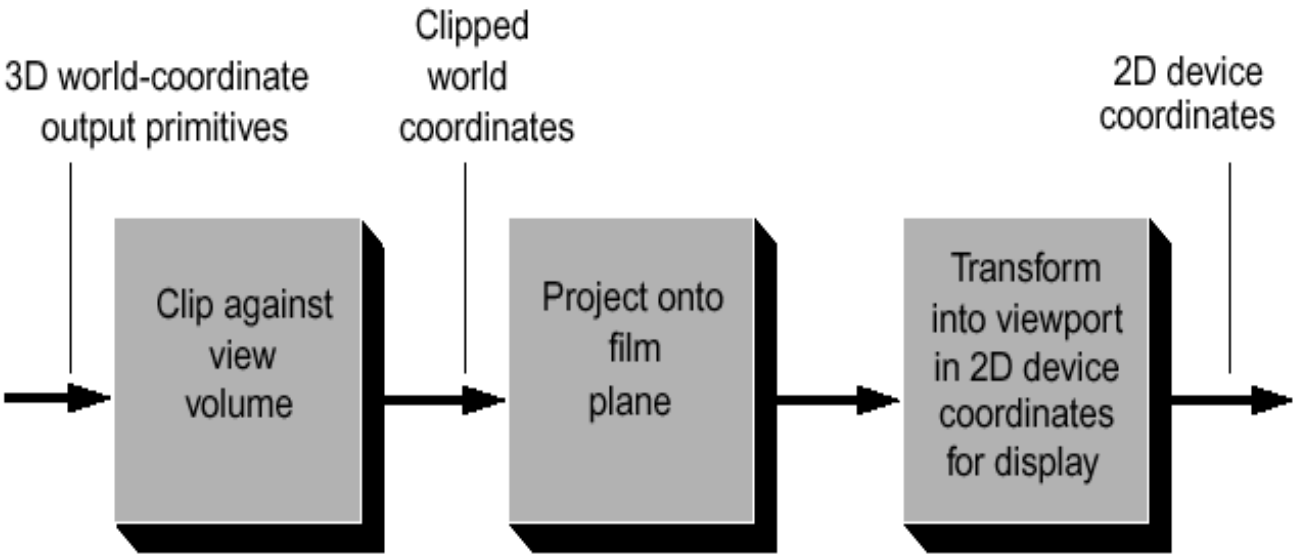
Virtual Camera: View Volumes



3D Viewing: Virtual Camera

- The virtual camera is the model (an abstraction) we use to specify 3D view projection parameters to the computer
- Generic virtual camera: each package has its own but they are all (nearly) equivalent:
  - position of camera
  - orientation
  - field of view (wide angle, normal…)
  - depth of field (near distance, far distance)
  - focal distance (blurring)
  - tilt of view/film plane (if not normal to view direction, produces oblique projections)
  - perspective or parallel projection-type (camera near objects or an infinite distance away)

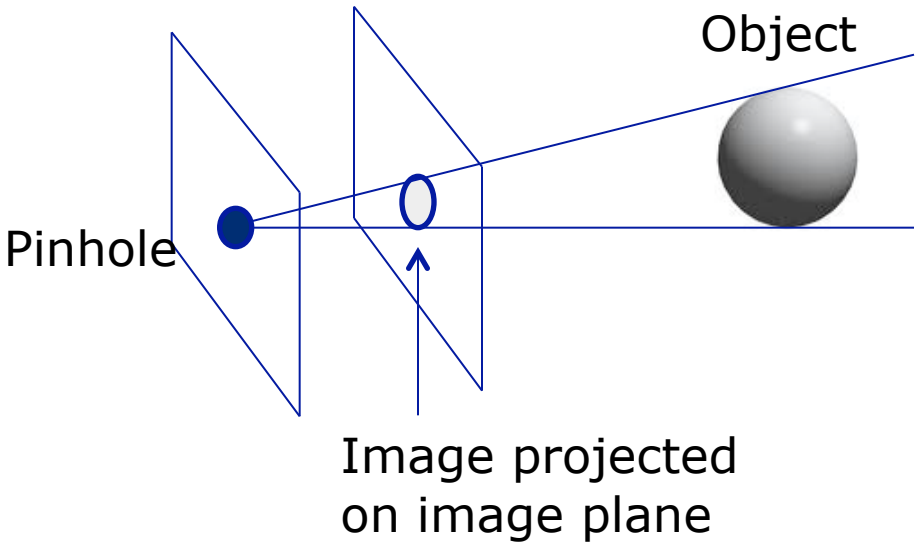
Cameras in Rendering Process



- Will detail coordinate systems for camera, i.e., view volume specification, projecting onto film plane, and transforming into viewport
- Let's first define viewport and view volume

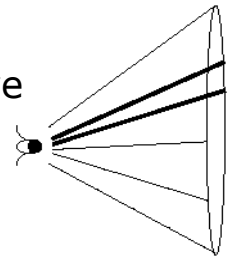
The Pinhole Model

- Think of camera as a pinhole, aka *camera position*, aka *Center of Projection COP*.
- As we look through pinhole we see a certain volume of space (rays of light reflect off objects and converge to pinhole to let us see the scene), aka our *view volume*
- *Projectors* intersect a *plane*, usually in between scene and pinhole, to project the object onto
- Lastly, projection is mapped to some form of viewing medium (2D screen, not shown here)



View Volumes

- A view volume contains everything the camera/ pinhole sees

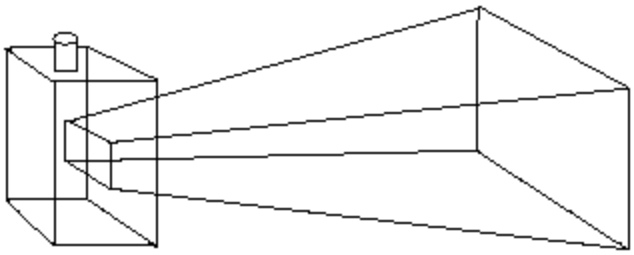


conical perspective view volume

- Conical view volume:
  - approximates what eye sees
  - expensive math (simultaneous quadratics) when clipping objects against cone's surface

- Can approximate with rectangular cone instead (called a *frustum*)
  - works well with a rectangular viewing window
  - simultaneous linear equations for easy clipping of objects against sides

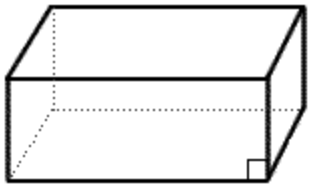
virtual camera



frustum approximation, view volume

- Also parallel view volumes, e.g., for orthographic projections.

These don't simulate eye or camera

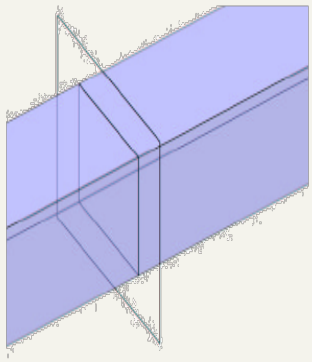


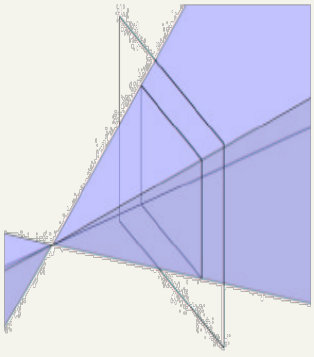
View volume

(Parallel view)

View Volumes and Projectors

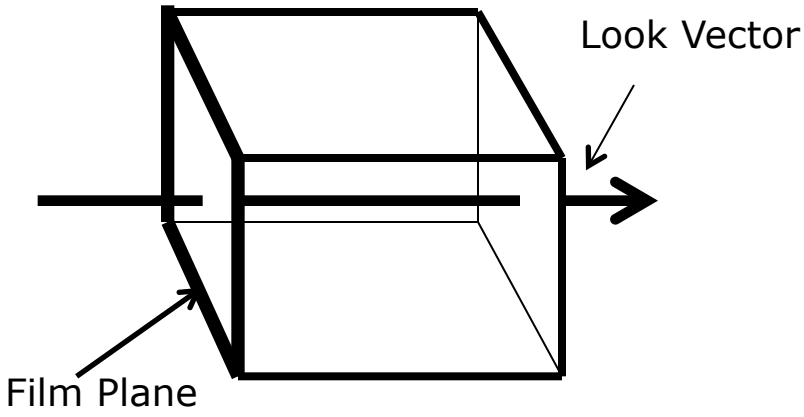
- Given our view volume need to start thinking about how to project scene contained in volume to film plane
- Projectors: Lines that essentially map points in scene to points on film plane
- **Parallel Volumes:** Parallel Projectors -- no matter how far away an object is, as long as it is in the view volume it will appear as same size
- **Perspective Volumes**: Projectors converge on eye point = center of projection, like rays of light converging to your eye



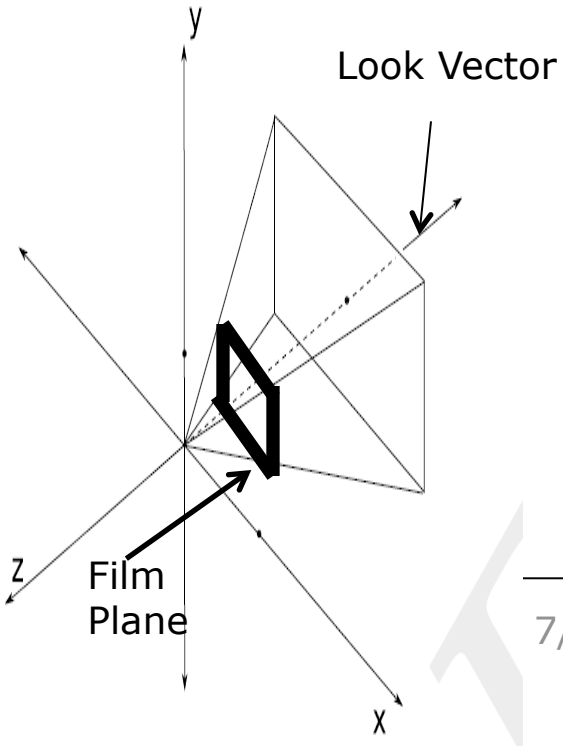


The Film Plane

- Film plane is in world space – 3D scene is projected onto a rectangle (film) on that plane using some projection and from there to the screen
- Film for our camera model will be perpendicular to and centered around the camera's look vector, and will match dimensions of our view volume

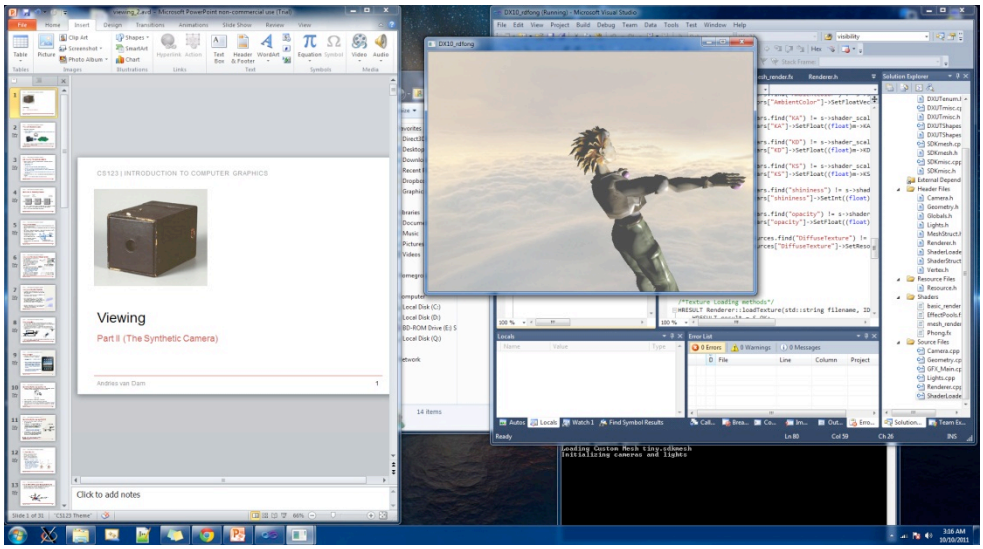


- Actual location of film plane along look vector doesn't matter as long as it is between eye/COP and scene



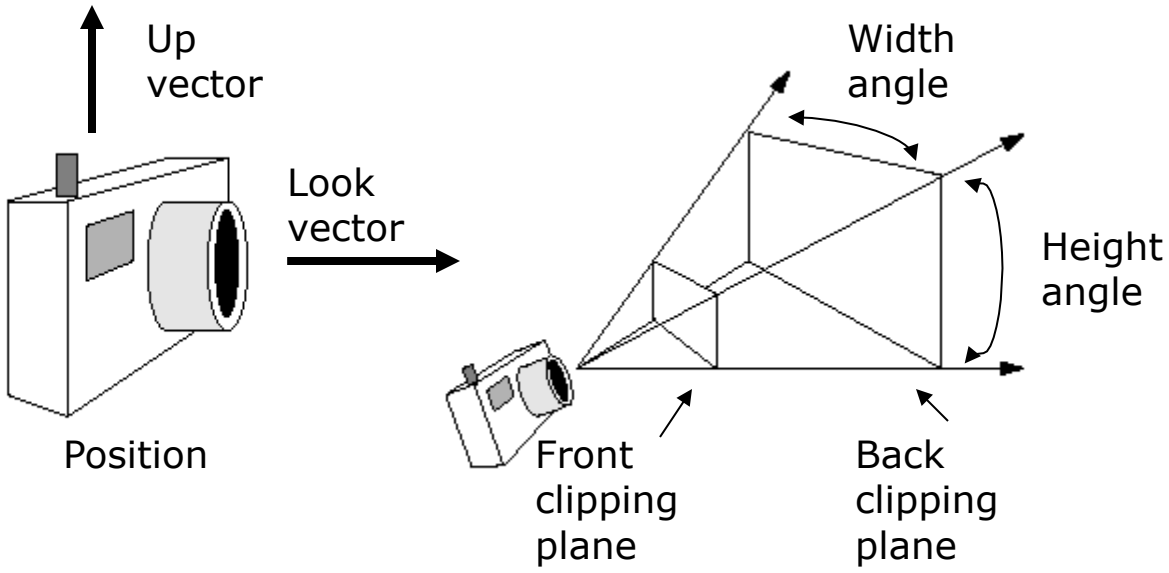
Viewport

- Viewport is rectangular area of the screen where a scene is rendered
  - this may or may not fill Window Manager's window
  - note: *window* in computer graphics used to mean a 2D clip rectangle on a 2D world coordinate drawing, and *viewport* denoted the 2D integer coordinate region of screen space to which the clipped window contents are mapped
- Viewport and 2D cross-section of 3D view volume may have different aspect ratios
  - viewport mapping (from film plane window to viewport's 2D device coordinates) specifies what to do if aspect ratios differ



Constructing the View Volume

- We need to know six things about our virtual camera model in order to take a picture



1) *Position* of the camera (from where it's looking)

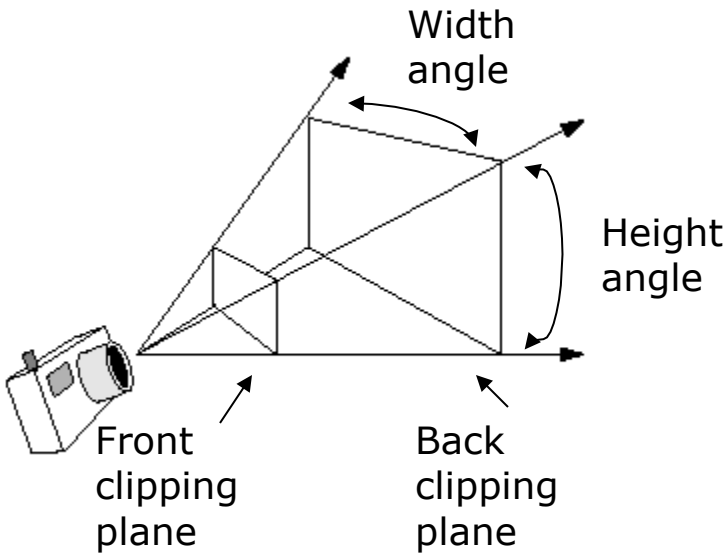
2) and 3) The *Look vector* specifies in what direction the camera is pointing. The camera's ***Orientation*** is determined by the *Look vector* and the angle through which the camera is rotated about that vector, i.e., the direction of the *Up vector*

Constructing the View Volume (2/2)

4) *Aspect ratio* of the electronic "film:" ratio of width to height

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

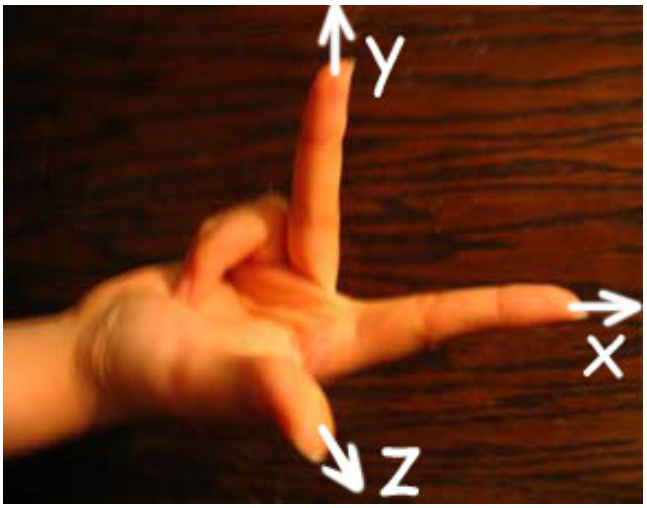
6) *Front* and *back clipping planes:* limit extent of camera's view by rendering (parts of) objects lying between them and throwing away everything outside of them



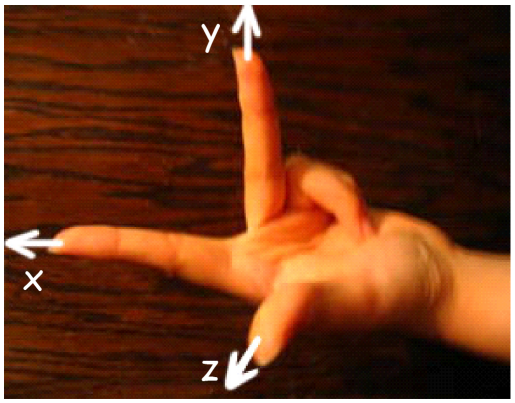
- Optional parameter — *Focal length*: often used for photorealistic rendering; objects at distance *Focal length* from camera are rendered in sharp focus, objects closer or farther away get blurred.

1) Position

- Where is the camera located with respect to the origin?
- Location given in the *right-handed x*, *y*, *z* coordinate system in 3-space



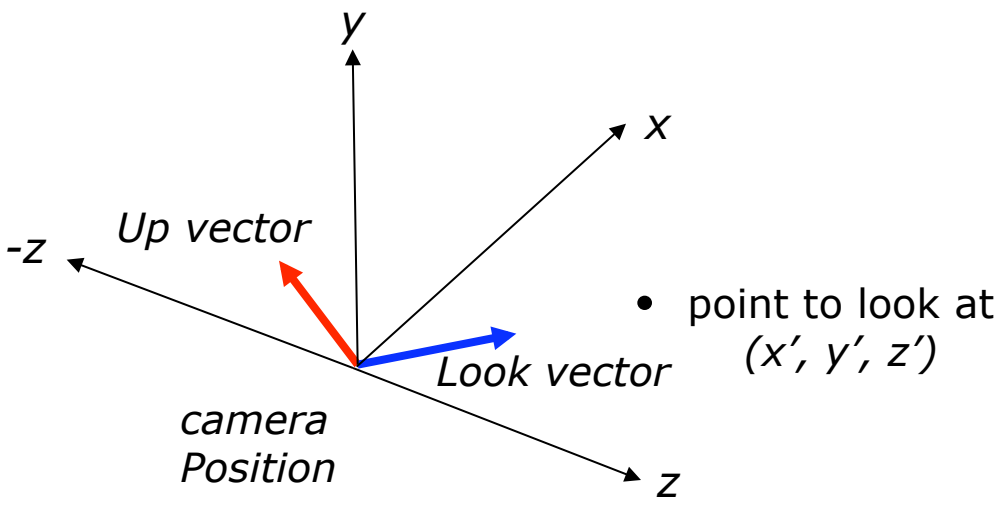
A right-handed coordinate system.



This is a left-handed coordinate system. Not used in 1566.

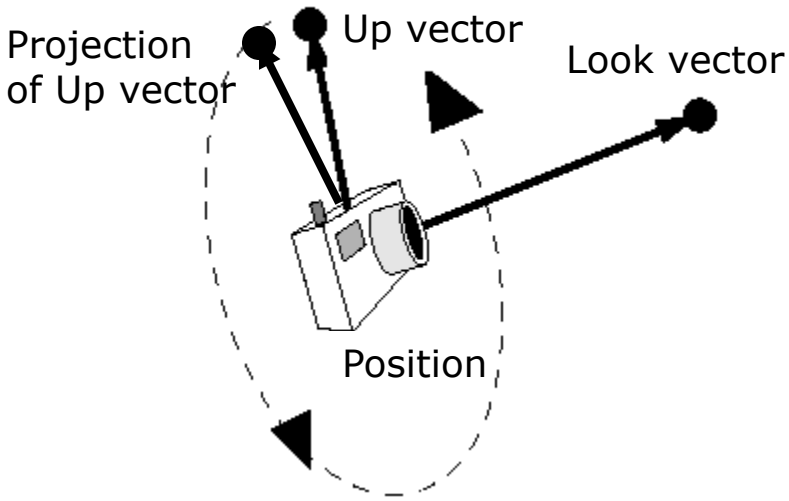
2) and 3) Orientation

- Orientation is specified by a point in 3D space to look at (or a direction to look in) and an angle of rotation about this direction
- Default (canonical) orientation is looking down the negative *z*-axis and up direction pointing straight up the *y*-axis
- In general the camera is located at the origin and is looking at an arbitrary point with an **arbitrary** up direction



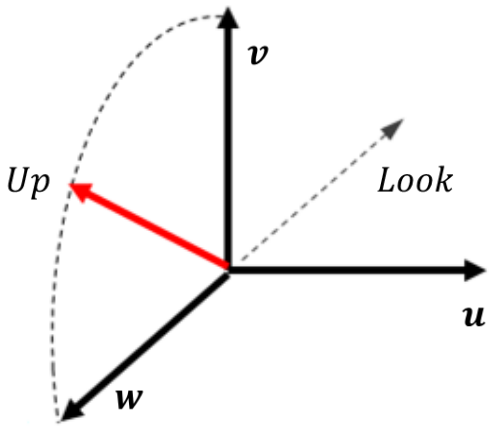
Look and Up Vectors

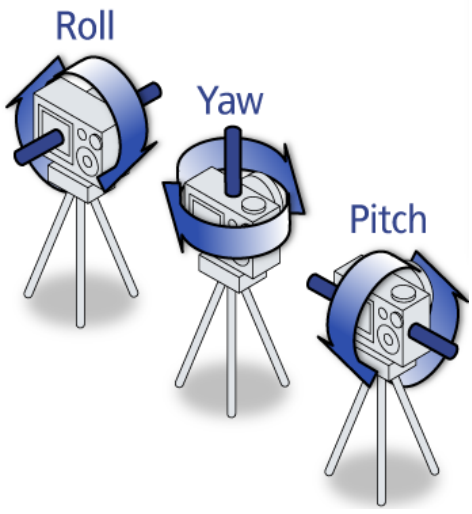
- More concrete way to say the same thing as orientation
- *Look Vector*
  - the direction the camera is pointing
  - three degrees of freedom; can be any vector in 3-space
- *Up Vector*
  - determines how the camera is rotated around the *Look vector*
  - for example, whether you're holding the camera horizontally or vertically (or in between)
  - Projection of *Up vector* must be in the plane perpendicular to the look vector (this allows *Up vector* to be specified at an arbitrary angle to its *Look vector*)



The Camera Coordinate Space

- The equivalent of ***x, y*** and ***z*** axes in camera space are unit vectors ***u, v*** and ***w*** (not to be confused with homogenous coordinate, **w**)
  - **Also a right handed coordinate system**
  - ***w*** is a unit vector in the opposite direction of the look vector
  - ***v*** is the part of the up vector perpendicular to the look vector, normalized to unit length
  - ***u*** is the unit vector perpendicular to both **v** and **w**
- Camera can rotate w.r.t. its (u,v,w) axes



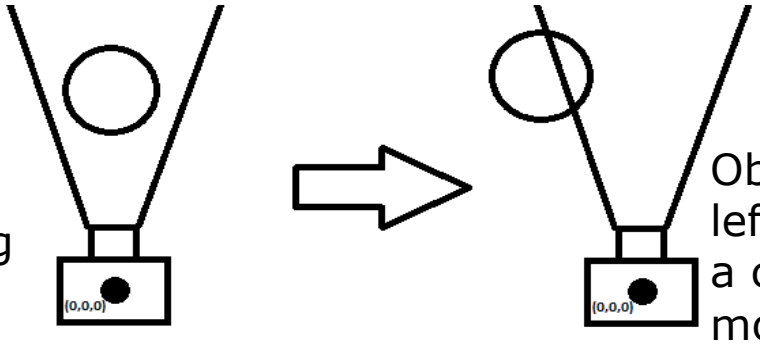


Note: The Camera As a Model

- There are different ways we can model a camera
- In the generalized model we have a camera and a scene where both the camera and objects in the scene are free to be transformed independently
- In a more restricted model we have a camera that remains fixed in one position and orientation
  - To transform the camera we actually apply inverse transformation to objects in scene
- This is the model OpenGL uses; note however that GLU abstracts this concept away from the programmer (gluLookAt())

Translate "camera" to the right

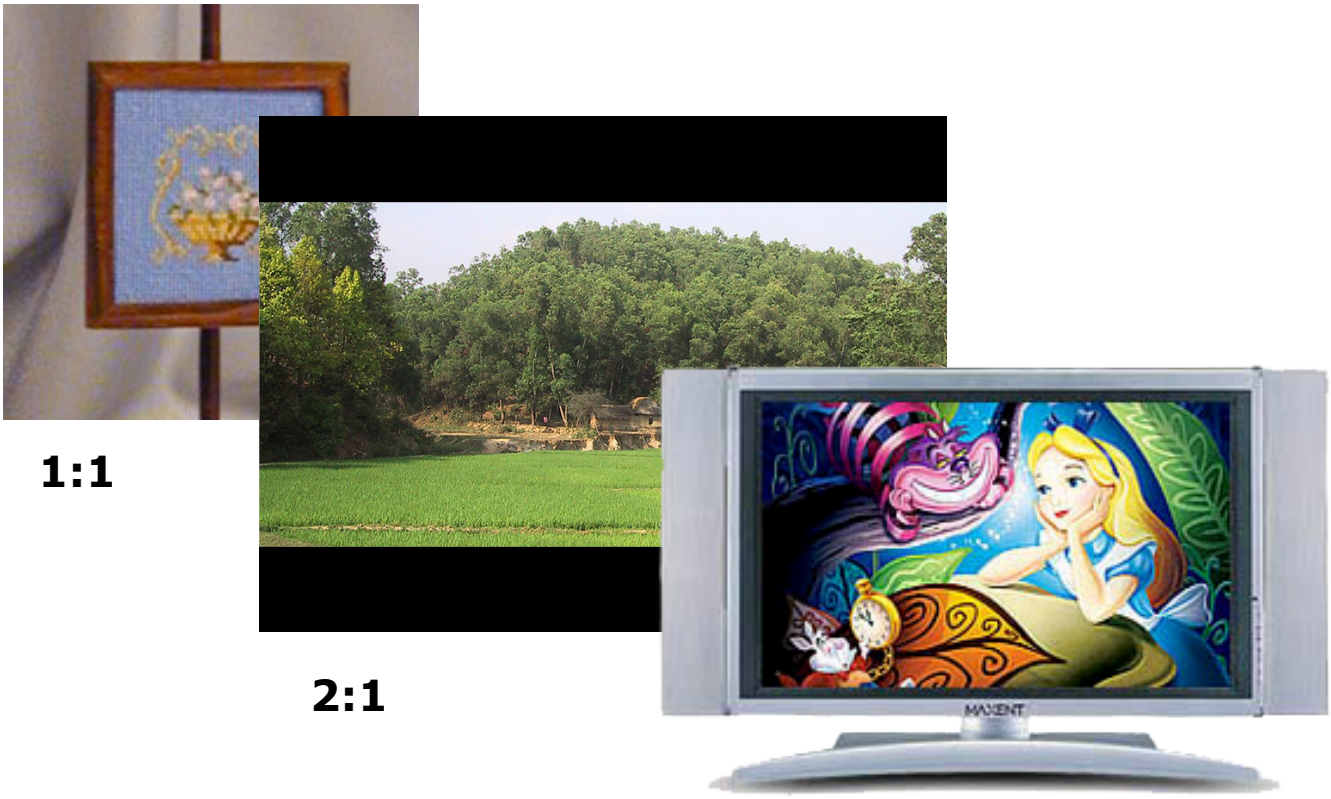
Field of view in OpenGL can be thought of as view of camera at origin looking down –z axis



Object moves to left to simulate a camera moving to right

4) Aspect Ratio

- Analogous to the size of film used in a camera
- Determines proportion of width to height of image displayed on screen
- Square viewing window has aspect ratio of 1:1
- Movie theater "letterbox" format has aspect ratio of 2:1
- NTSC television has an aspect ratio of 4:3, and HDTV is 16:9



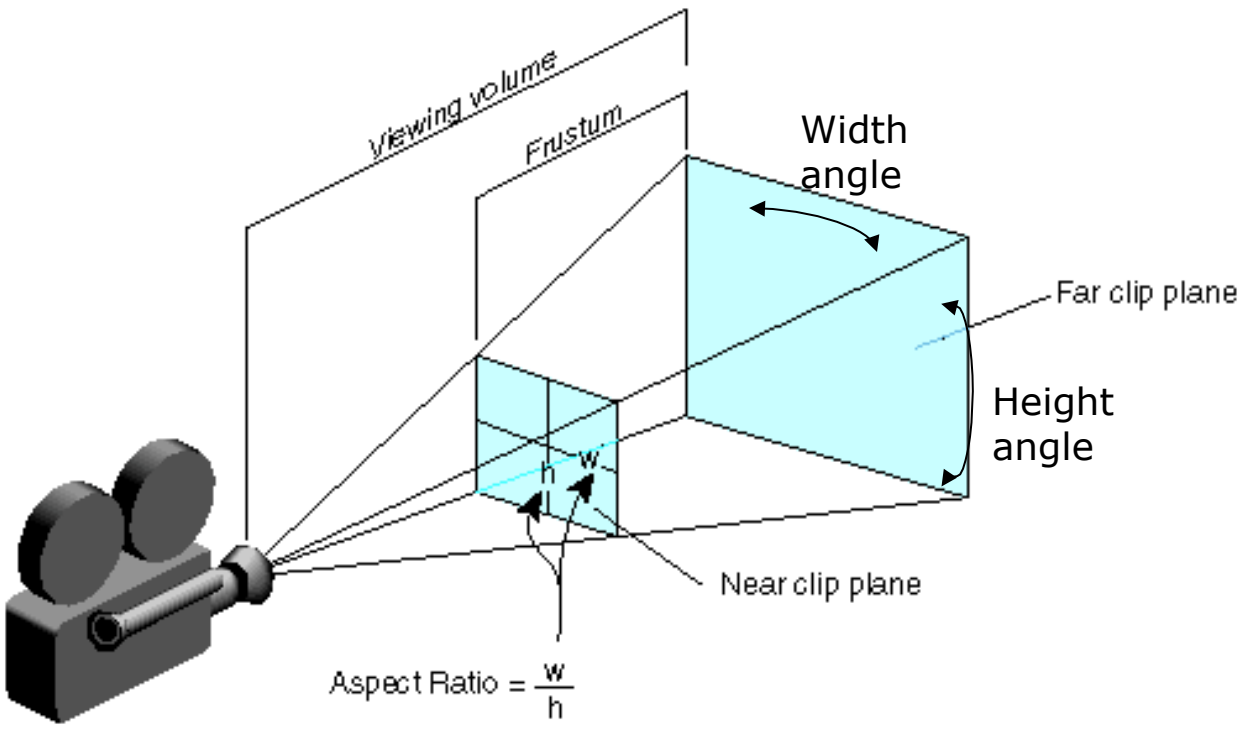
1:1

2:1

16:9

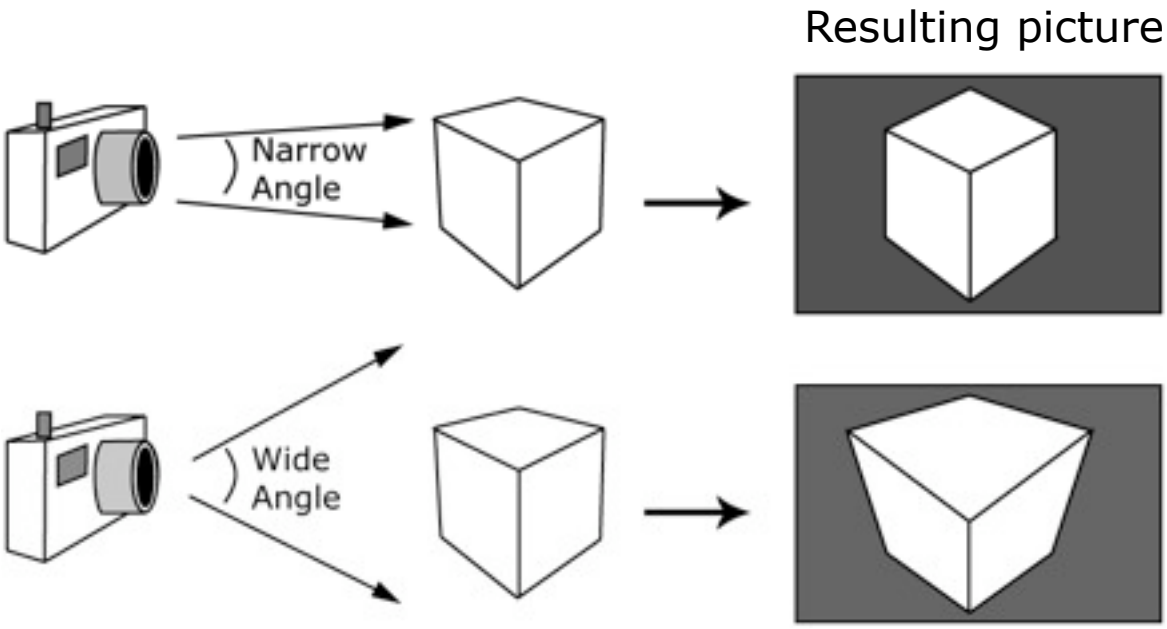
5) View Angle (1/2)

- 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
- Choosing *View angle* analogous to photographer choosing a specific type of lens (e.g., a wide-angle or telephoto lens)



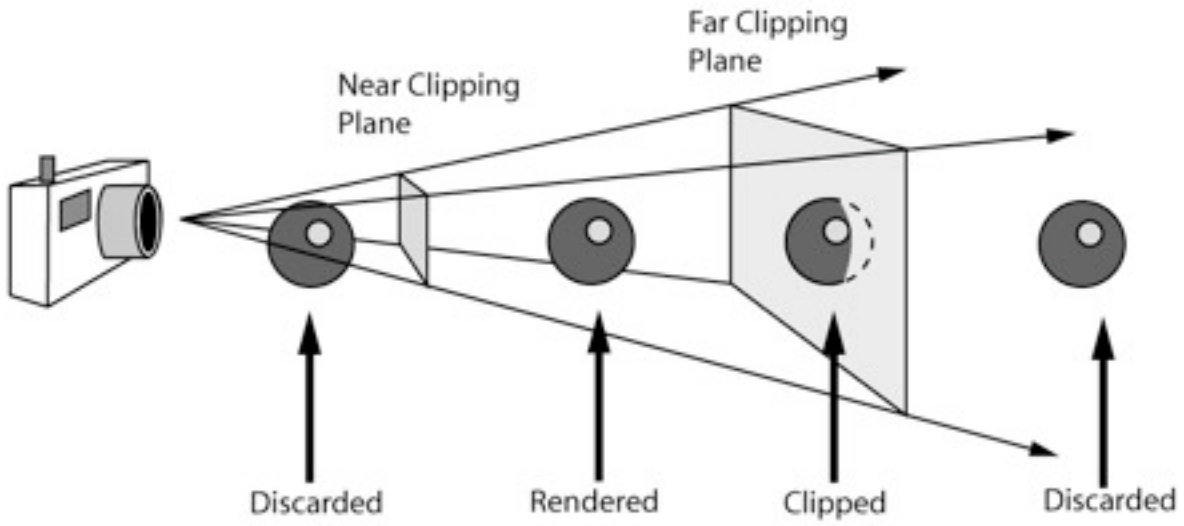
5) View Angle (2/2)

- Lenses made for distance shots often have a nearly parallel viewing angle and cause little perspective distortion, though they foreshorten depth
- Wide-angle lenses cause a lot of perspective distortion



6) Front and Back Clipping Planes (1/4)

- Volume of space between *Front* and *Back clipping planes* defines what camera can see
- Position of planes defined by distance along *Look vector*
- Objects appearing outside of view volume don't get drawn
- Objects intersecting view volume get clipped



6) Front and Back Clipping Planes (2/4)

- Reasons for *Front* (near) *clipping plane*:
  - Don't want to draw things too close to the camera
    - would block view of rest of scene
    - objects would be prone to distortion
  - Don't want to draw things behind camera
    - wouldn't expect to see things behind the camera
    - in the case of the perspective camera, if we were to draw things behind the 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
    - by discarding them we lose a small amount of detail but reclaim a lot of rendering time
    - alternately, the scene may be filled with many significant objects; for visual clarity, we may wish to declutter the scene by rendering those nearest the camera and discarding the rest

6) Front and Back Clipping Planes

(3/4)

- Have you ever played a video game and all of the sudden some object pops up in the background (e.g. a tree in a racing game)? That's the object coming inside the far clip plane.
- The old hack to keep you from noticing the pop-up is to add fog in the distance. A classic example of this is from *Turok: Dinosaur Hunter*

- Now all you notice is fog and how little you can actually see. This practically defeats the purpose of an outdoor environment! And you can *still* see pop-up from time to time.
- Thanks to fast hardware and level of detail algorithms, we can push the far plane back now and fog is much less prevalent

6) Front and Back Clipping Planes

(4/4)

- Putting the near clip plane as far away as possible helps Z precision. Sometimes in a game you can position the camera in the right spot so that the front of an object gets clipped letting you see inside of it.
- Modern video games uses various techniques to avoid this visual glitch. One technique is to have objects that are very close to the near clip plane fade out before they get cut off, as can be seen from these screenshots of *Okami*.

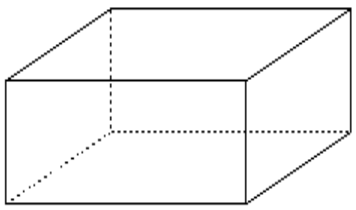
This technique gives a clean look while solving the near clipping problem (the wooden fence fades out as the camera follows the running wolf).

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 camera are rendered in focus; objects closer or farther away than *Focal length* get blurred
- *Focal length* used in conjunction with clipping planes
- Only objects within view volume are rendered, whether blurred or not. Objects outside of view volume still get discarded

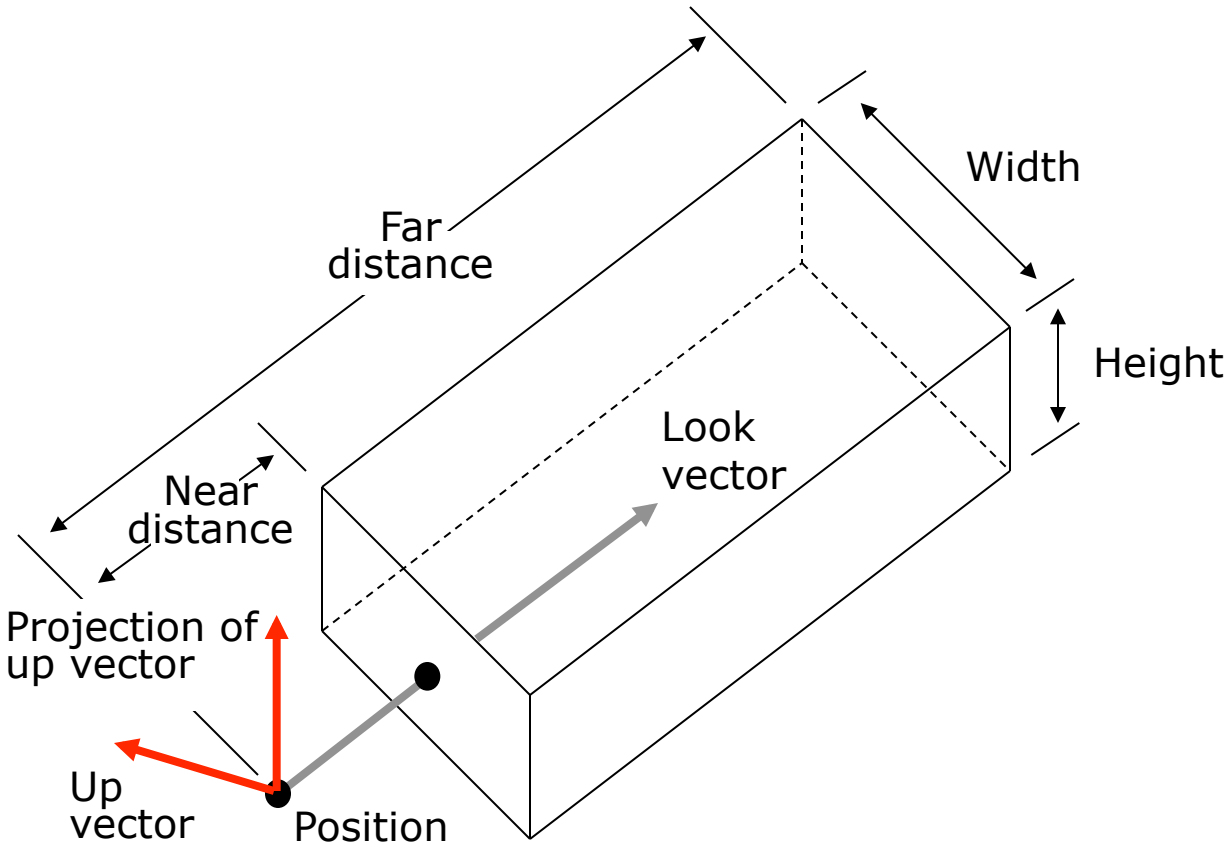
The Parallel View Volume (1/2)

- Up until now we've been describing the specifications for a perspective view volume
- We also need to discuss the parallel view volume (as mentioned last time, used for orthogonal/metric views)
- What do we need to know this time?
  - Everything we wanted for a perspective view volume except for width and height angles, replaced by just a width and height (also the width and height of our film on our film plane)
  - A parallel view volume is a parallelepiped (all opposite edges parallel)



Rectangular Parallelpiped

The Parallel View Volume (2/2)

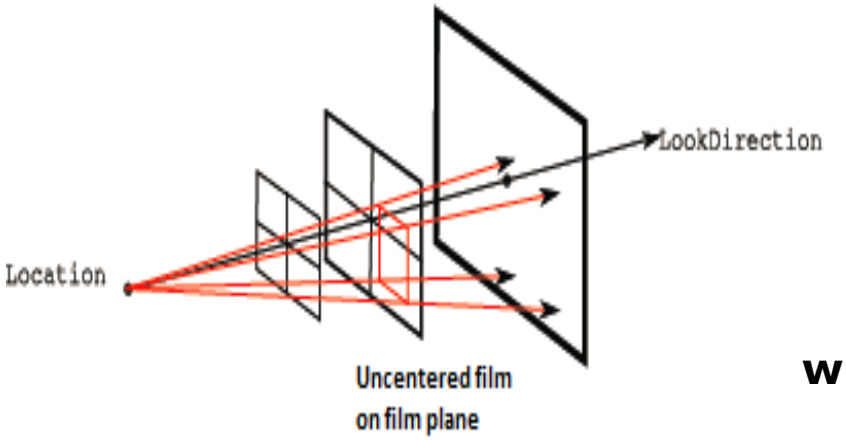


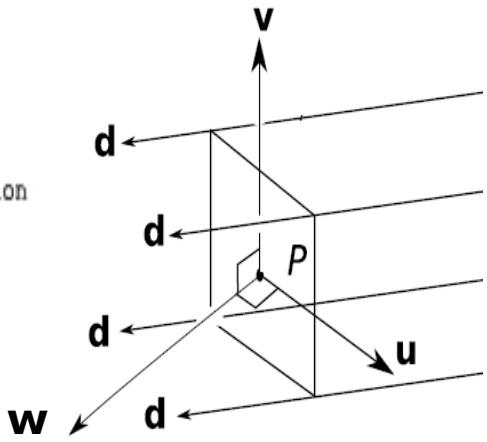
- Objects appear the same size no matter how far away they are since projectors are all parallel
- A benefit of the parallel view volume is that it's really easy to project a 3D scene to a 2D medium

Capabilities of a Generalized Camera (1/2)

- In a more generalized camera the viewing window doesn't have to be centered about the *Position* + *LookDirection*

- Nor does it have to be perpendicular to the LookDirection





- This allows us to use a more flexible view as well as enable the use of more view types

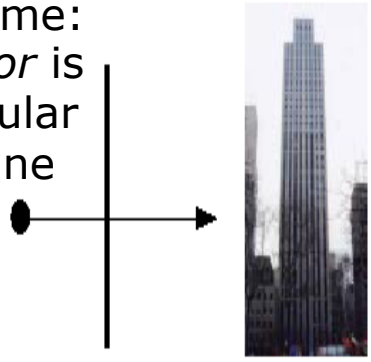
- Using an uncentered film we can essentially choose which part of our original perspective projection to view

Capabilities of a Generalized Camera
(2/2)

- Using a parallel view volume we can do oblique projections (cavalier, cabinet, perspective oblique) where the look vector/projectors and film plane aren't perpendicular:

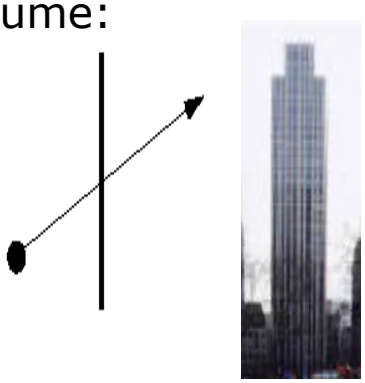
Non-oblique perspective view volume:

Look vector is perpendicular to film plane



Oblique perspective view volume:

Look vector is at an angle to the film plane



- Our model of a camera is not all encompassing
- There are some capabilities that we have omitted for the sake of simplicity
- Our film is centered around the camera position and always perpendicular to the look vector

Recap

- Virtual camera, aka 3D to 2D projection in CG
- Camera in rendering process
- Pinhole model
- View volumes, projectors, film plane, viewport
- Constructing the Perspective View Volume
  - position (eye or Point Of View)
  - direction (Look) and orientation (Up)
    - Look and Up vectors
    - camera coordinate space
    - camera motion as relative motion
  - aspect ratio (size of picture on screen)
  - field of view (wide or narrow angle; width and height angles)
  - depth of field (near distance and far distance)
  - focal length

- Parallel view volume
- Generalized cameras

PS: Get ready for some math!