Lecture 12:

Rendering for Virtual Reality

Interactive Computer Graphics
Stanford CS248, Spring 2021

Virtual reality (VR) vs augmented reality (AR)

VR = virtual reality

User is completely immersed in virtual world (sees only light emitted by display

AR = augmented reality

Display is an overlay that augments user's normal view of the real world (e.g., terminator)

Image credit: Terminator 2 (naturally)

VR headsets

Oculus Quest 2

HTC Vive

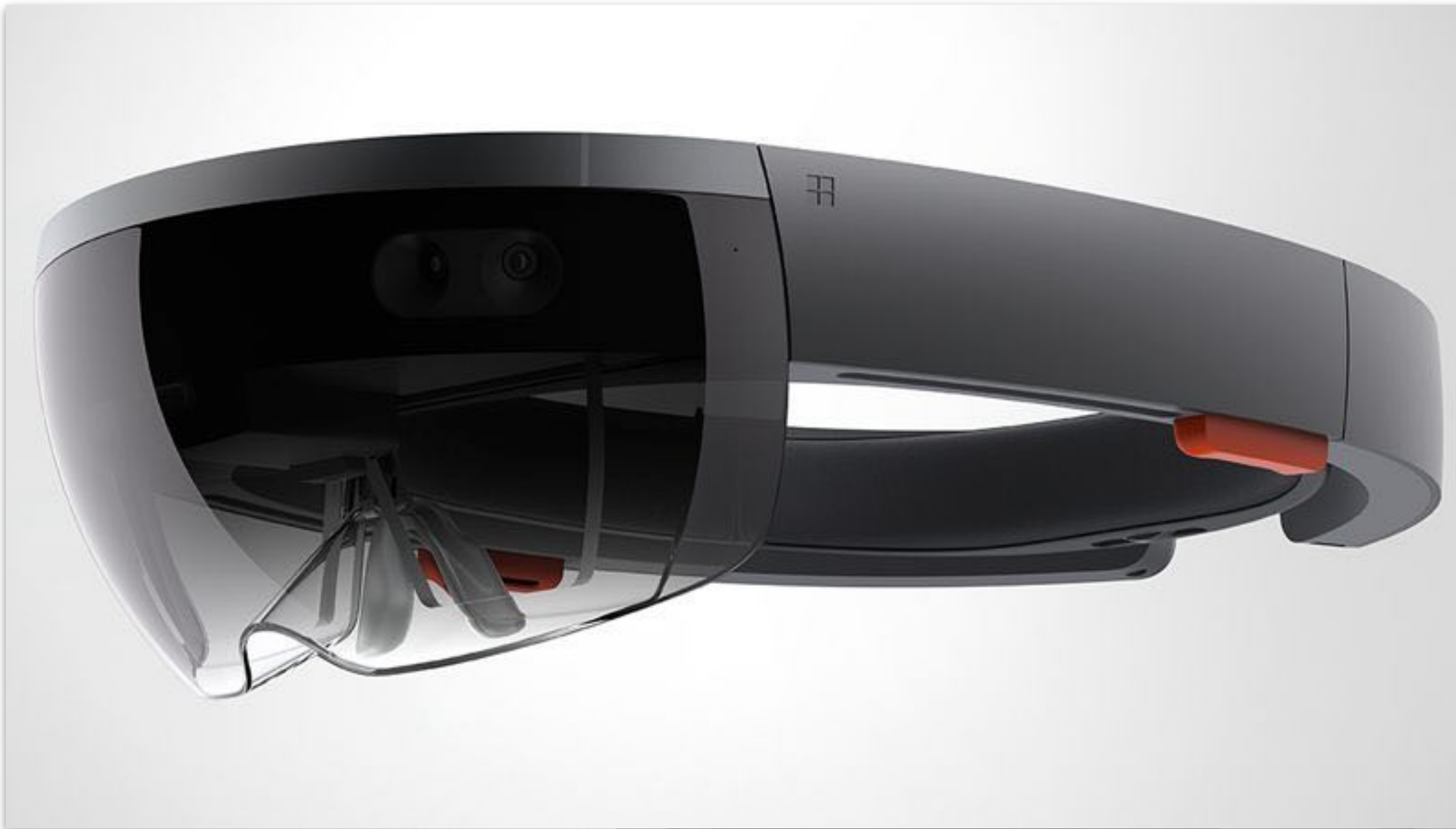
Sony Morpheus

Valve Index

Google Daydream

Google Cardboard

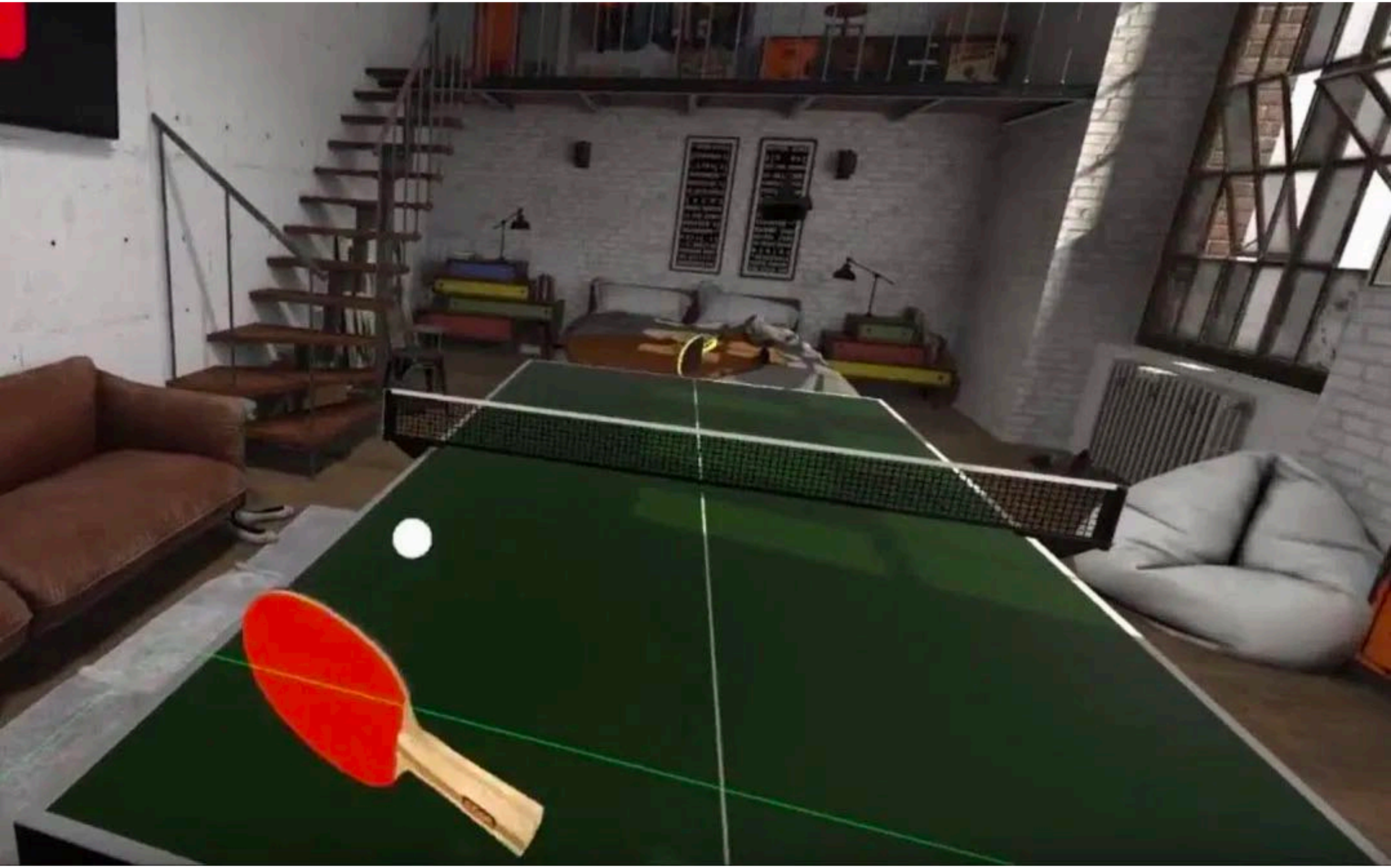
AR headset: Microsoft Hololens



AR on a mobile device

AR “passthrough” on VR headset

VR gaming

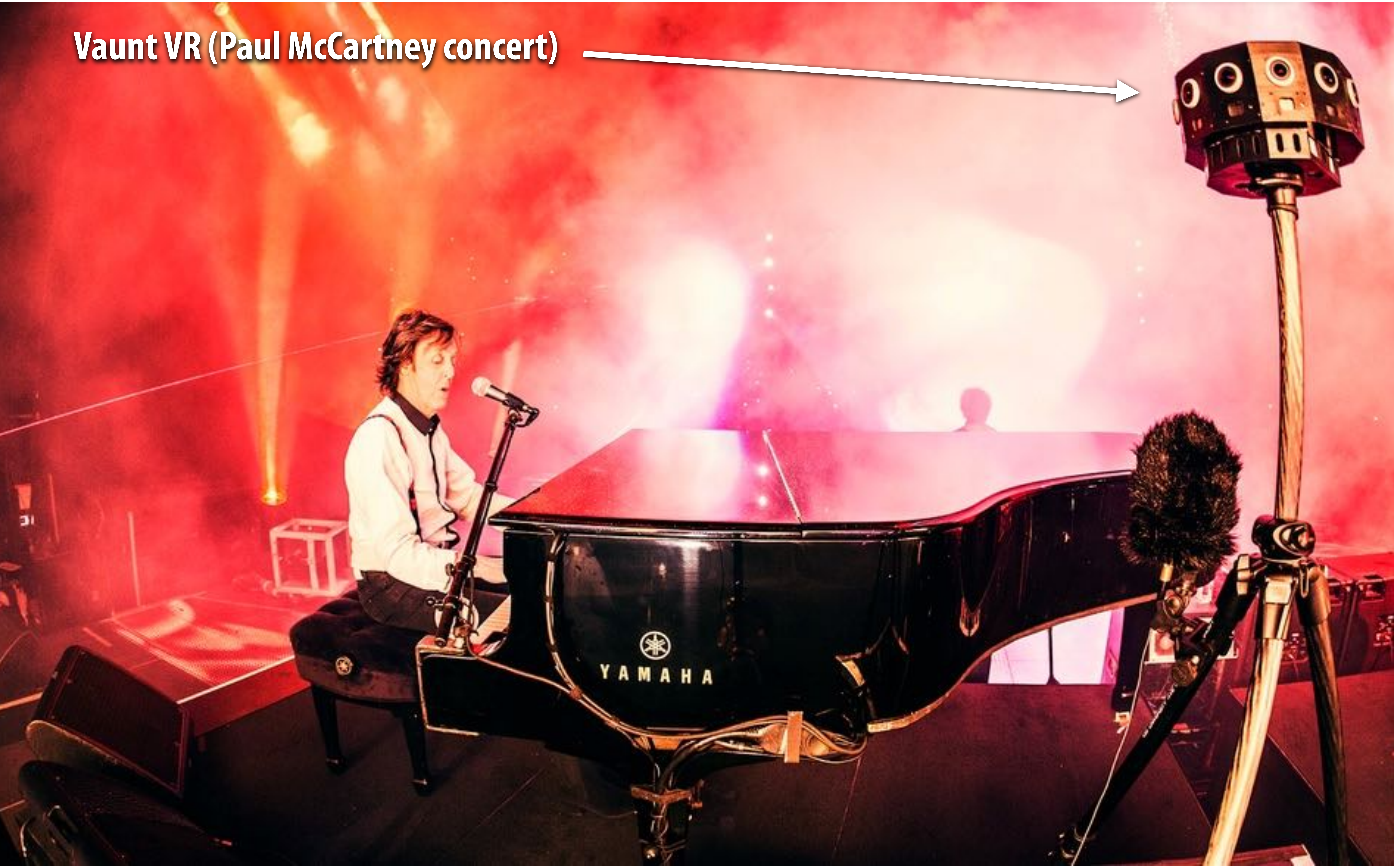


Eleven: Table Tennis (Fun Labs)

VR teleconference / video chat

Image credit: Spatial

VR video



VR video

Today: rendering challenges of VR

- Today we will talk about the unique challenges of rendering for modern VR headsets

- VR presents many other difficult technical challenges
  - display technologies
  - accurate tracking of face, head, and body position
  - haptics (simulation of touch)
  - sound synthesis
  - user interface challenges (inability of user to walk around environment, how to manipulate objects in virtual world)
  - content creation challenges
  - and on and on…

Oculus Rift CV1

Oculus Rift CV1 headset

Image credit: ifixit.com

Oculus Rift CV1 headset

Image credit: ifixit.com

Oculus Rift CV1 headset

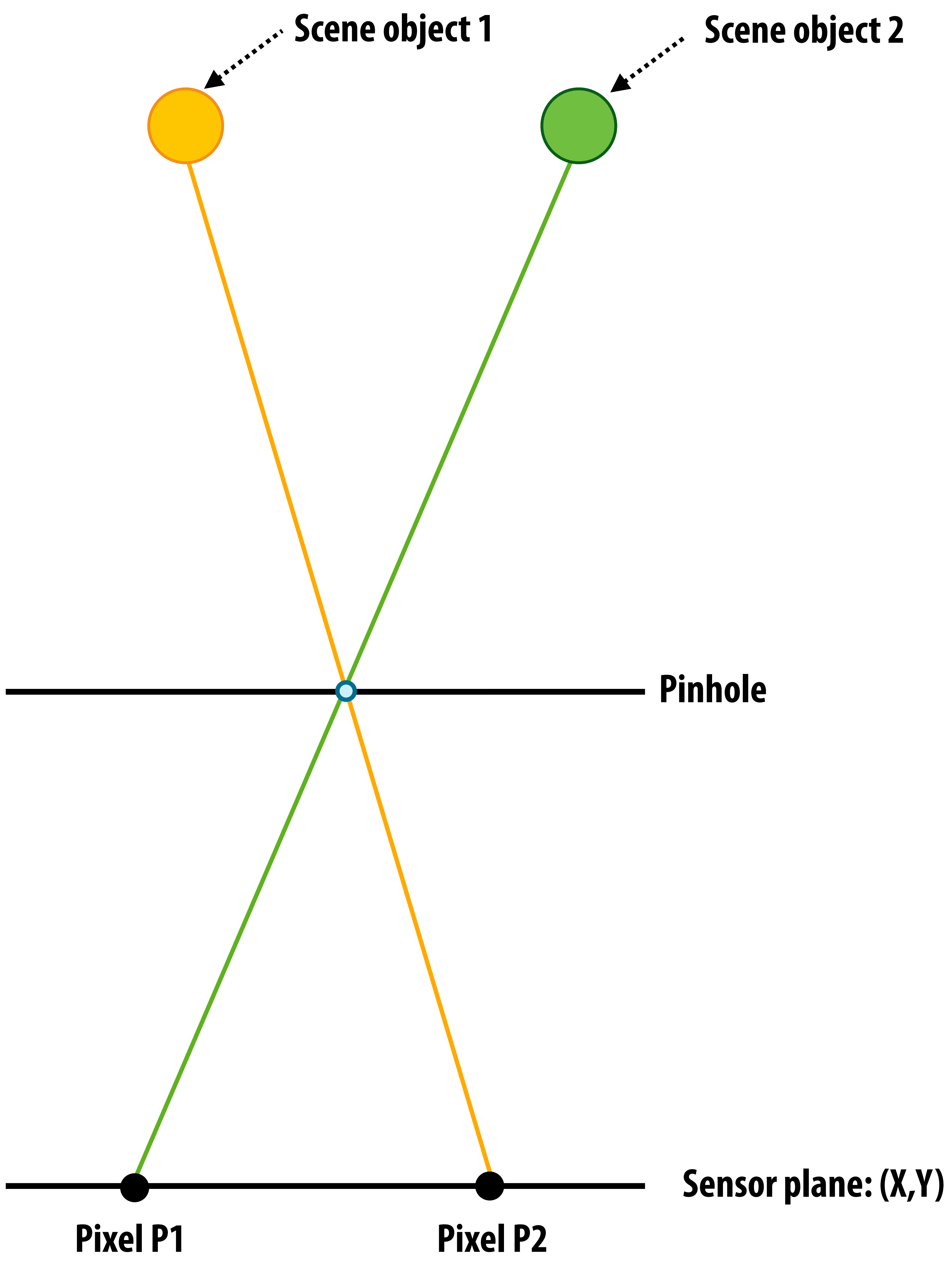
1080x1200 OLED display per eye
(2160 x 1200 total pixels)
90 Hz refresh rate
110° field of view

Image credit: ifixit.com

Aside: what does a lens do?

Recall: pinhole camera (no lens)

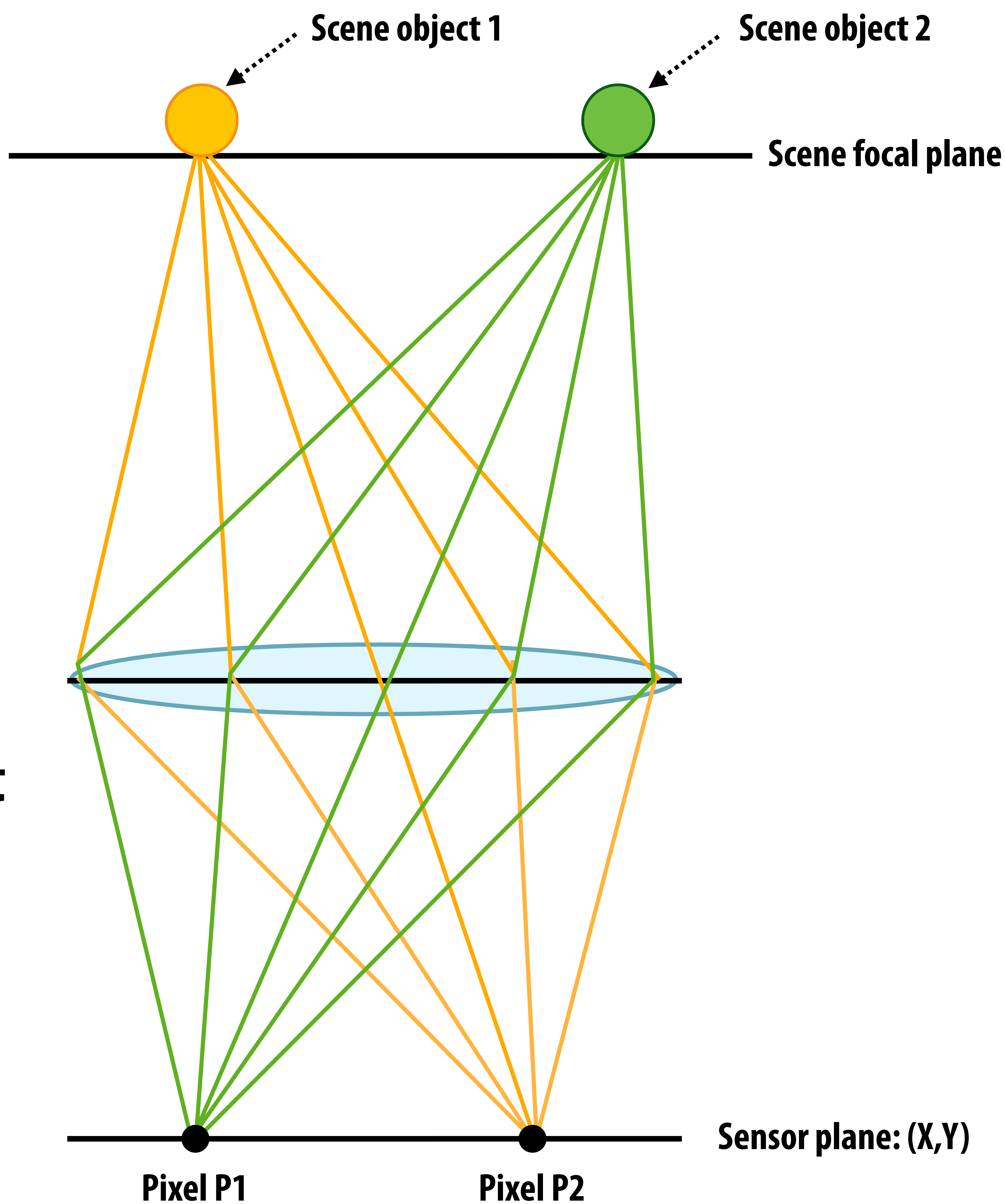
(every pixel measures light intensity along ray of light passing through pinhole and arriving at pixel)



What does a lens do?

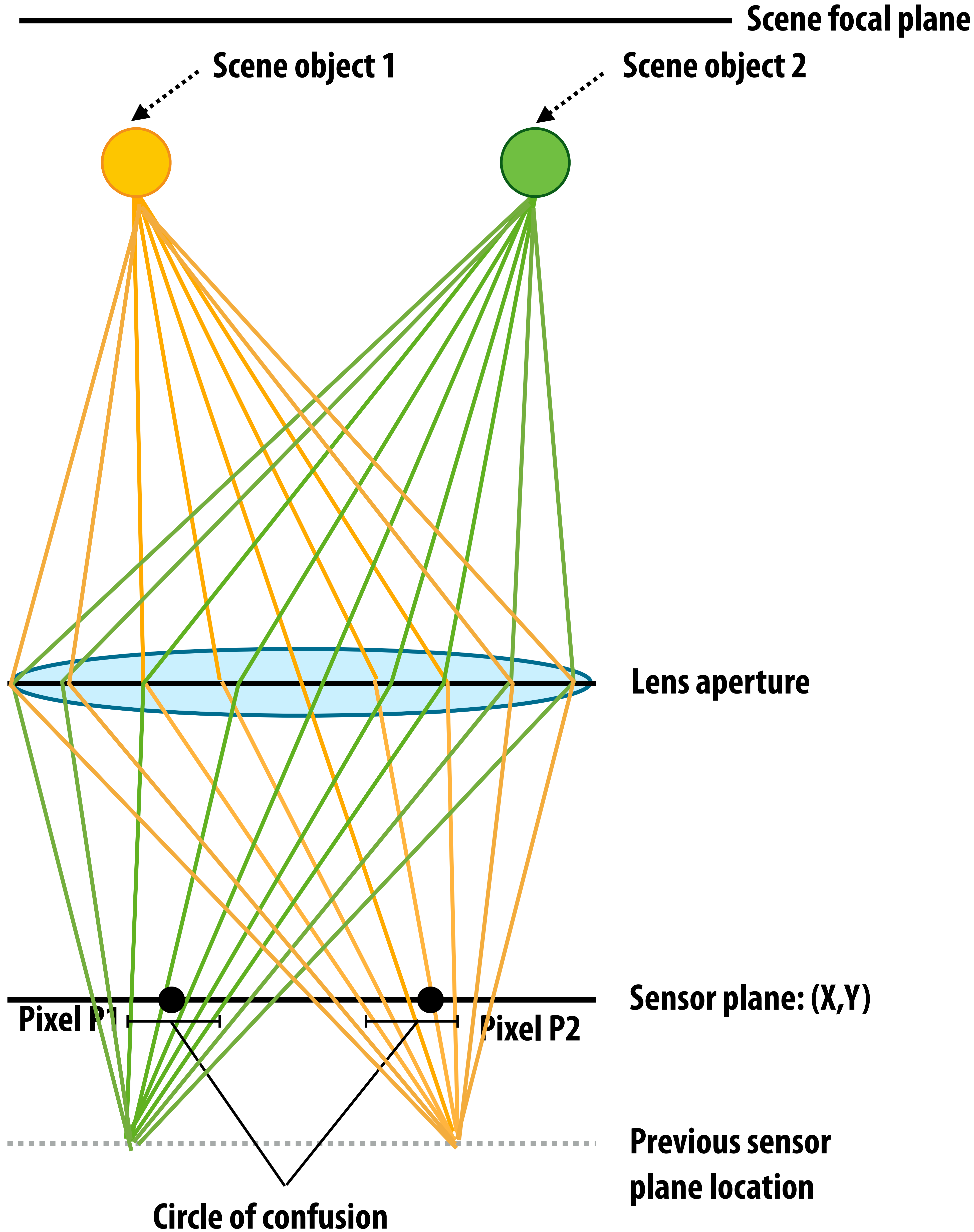
Every pixel accumulates all rays of light passing through lens aperture and refracted to location of pixel

When camera is in focus:
all rays of light from one point on focal plane in scene arrive at one point on sensor plane



Out of focus camera

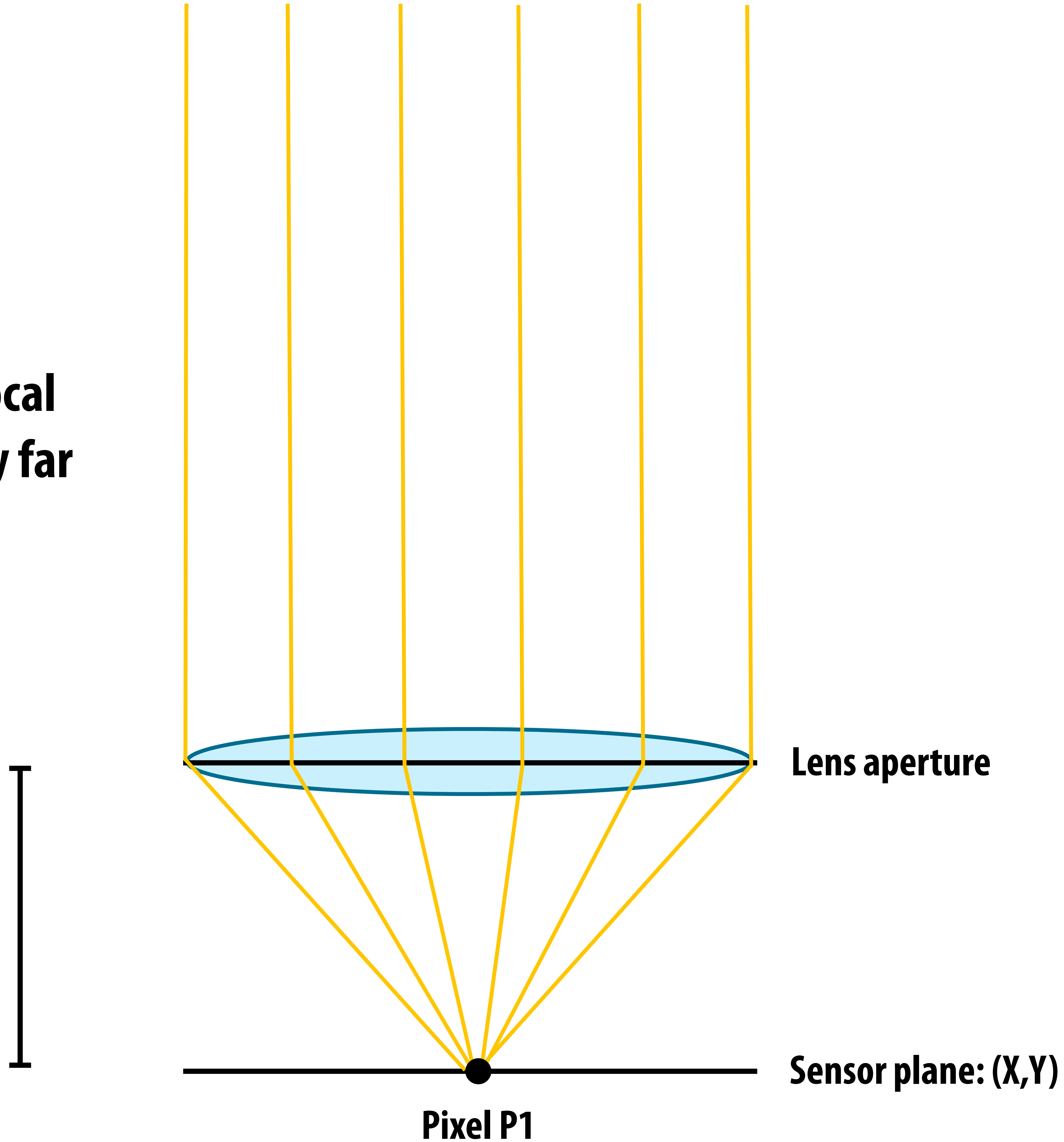
Out of focus camera: rays of light from one point in scene do not converge at point on sensor



Bokeh

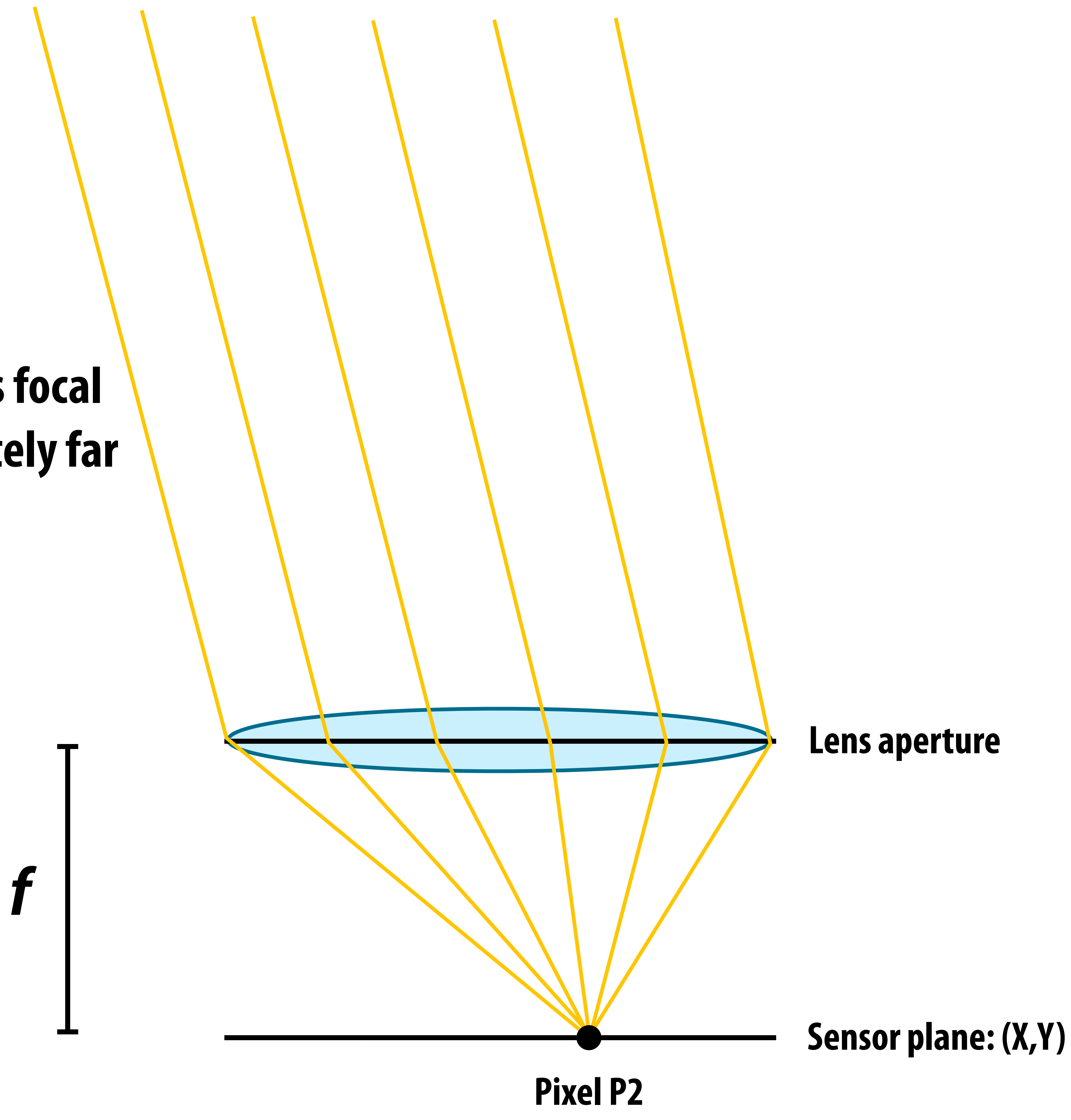
Lens focal length (*f*)

Here: camera's focal plane is infinitely far away.

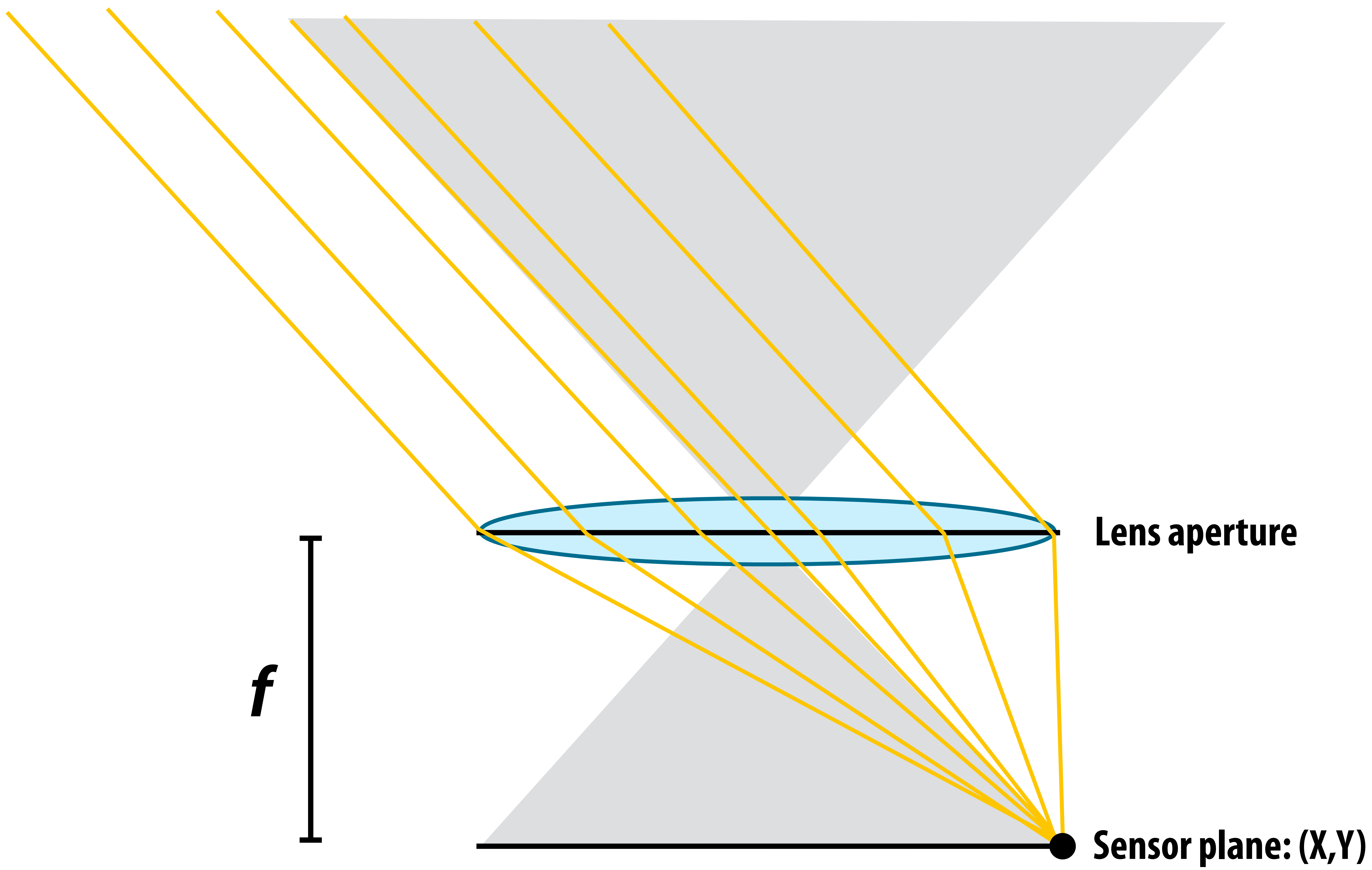


Lens focal length (*f*)

Here: camera's focal plane is infinitely far away.



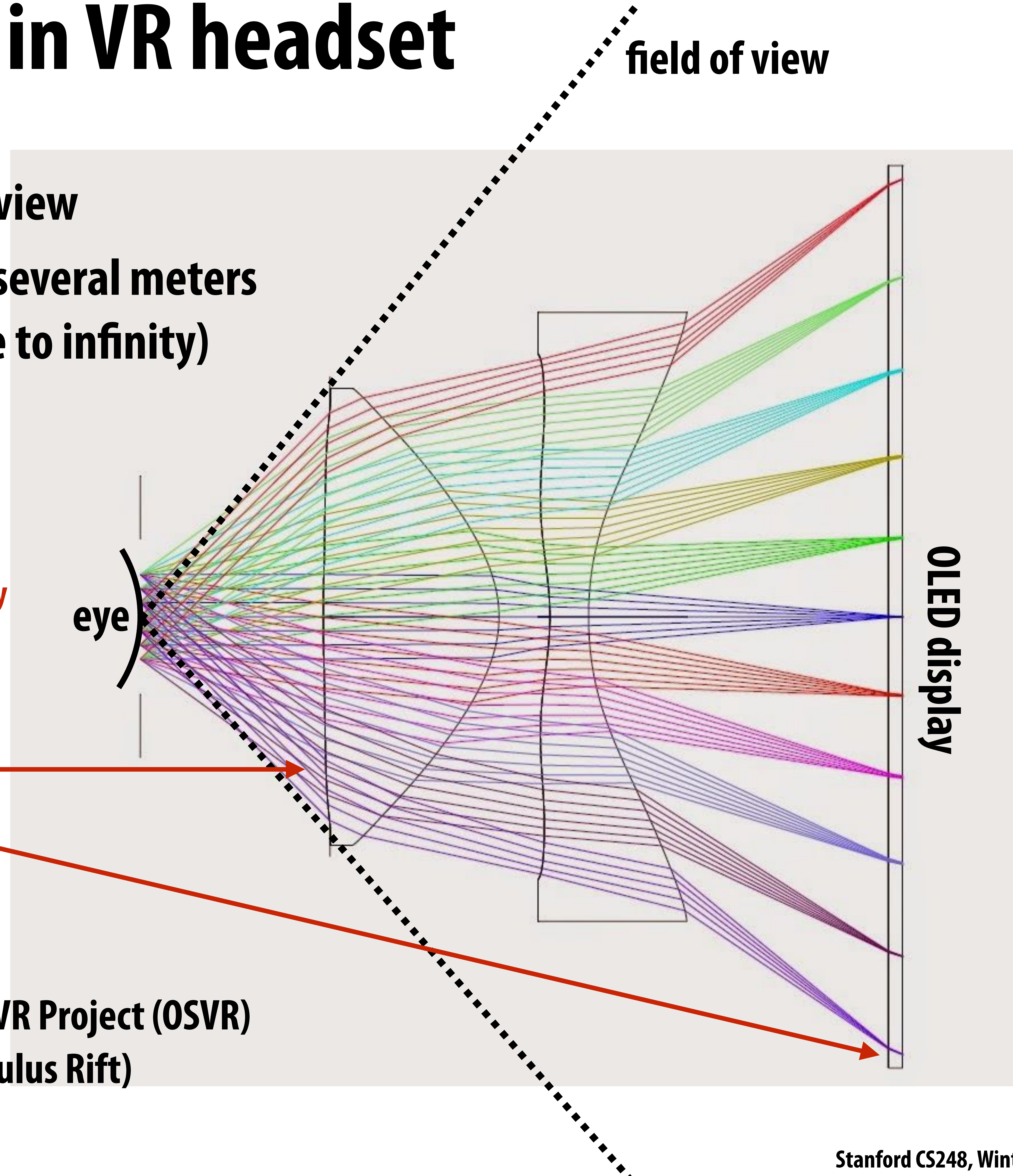
Lens field of view



Role of lenses in VR headset

1. **Create wide field of view**
2. **Place focal plane at several meters away from eye (close to infinity)**

Note: parallel lines reaching eye converge to a single point on display (eye accommodates to plane near infinity)



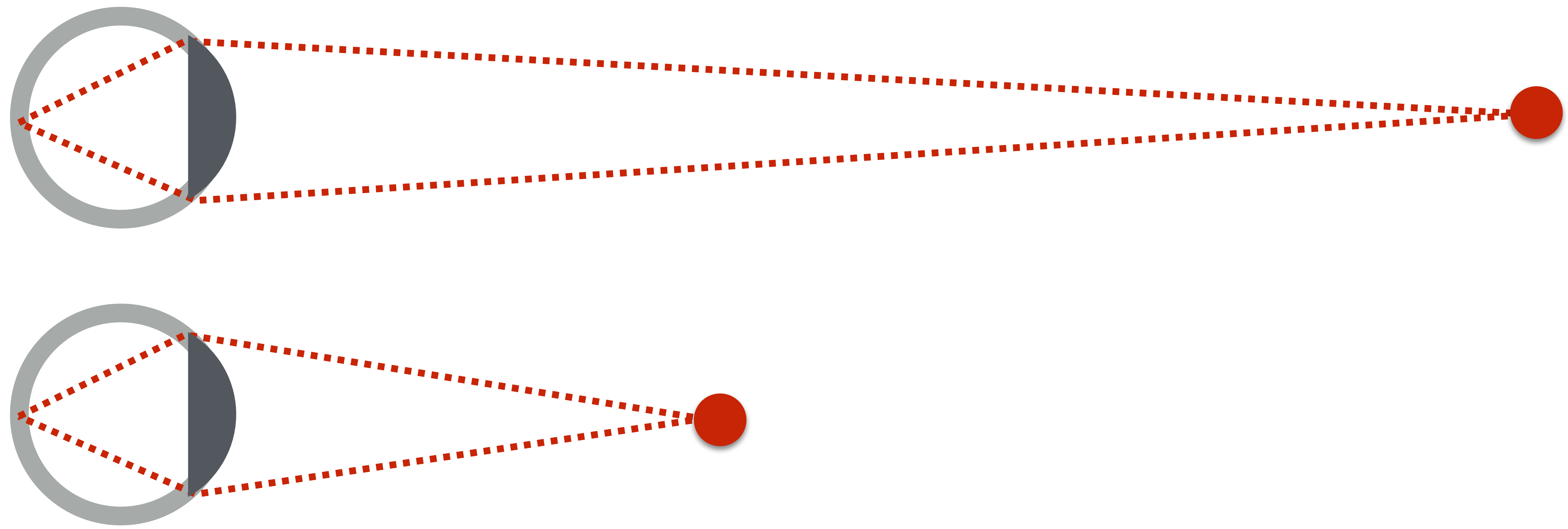
Lens diagram from Open Source VR Project (OSVR)
(Not the lens system from the Oculus Rift)
http://www.osvr.org/

Accommodation and vergence

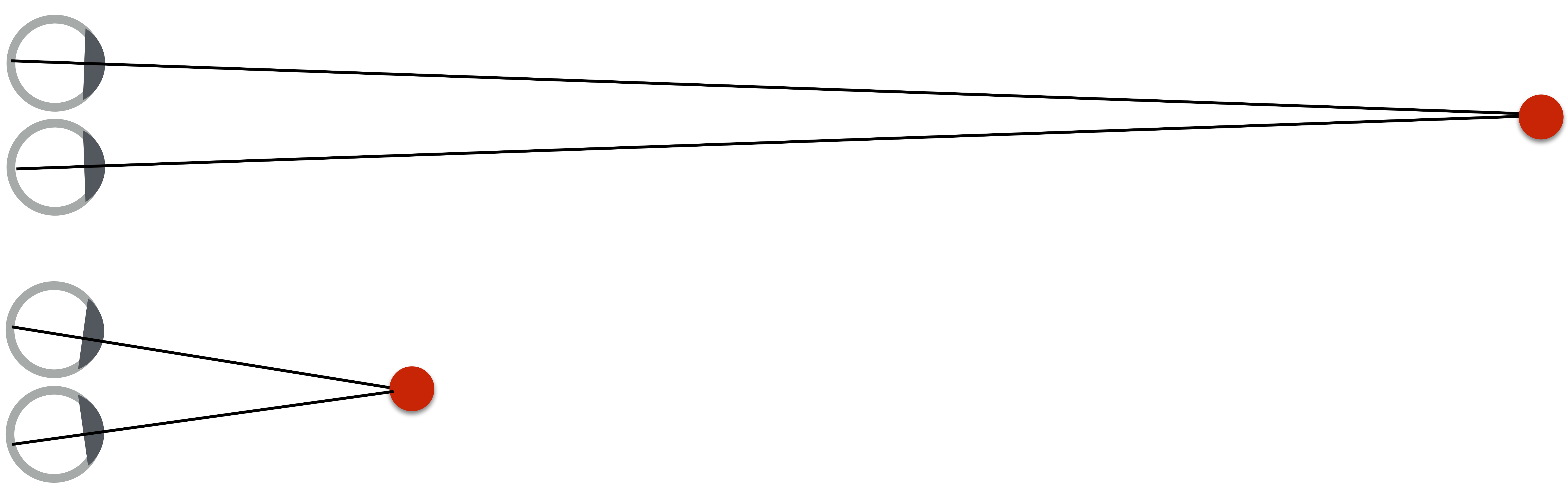
Accommodation: changing the optical power of the eye to focus at different distances

Eye accommodated at far distance

Eye accommodated at near distance

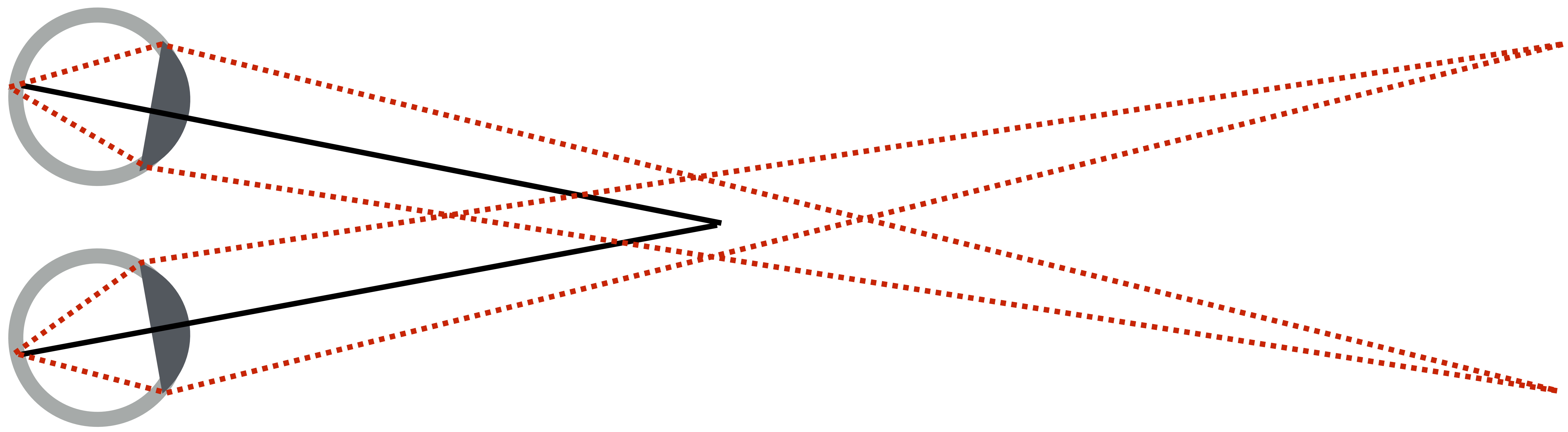


Vergence: rotation of eye to ensure projection of object falls in center of retina



Accommodation/vergence conflict

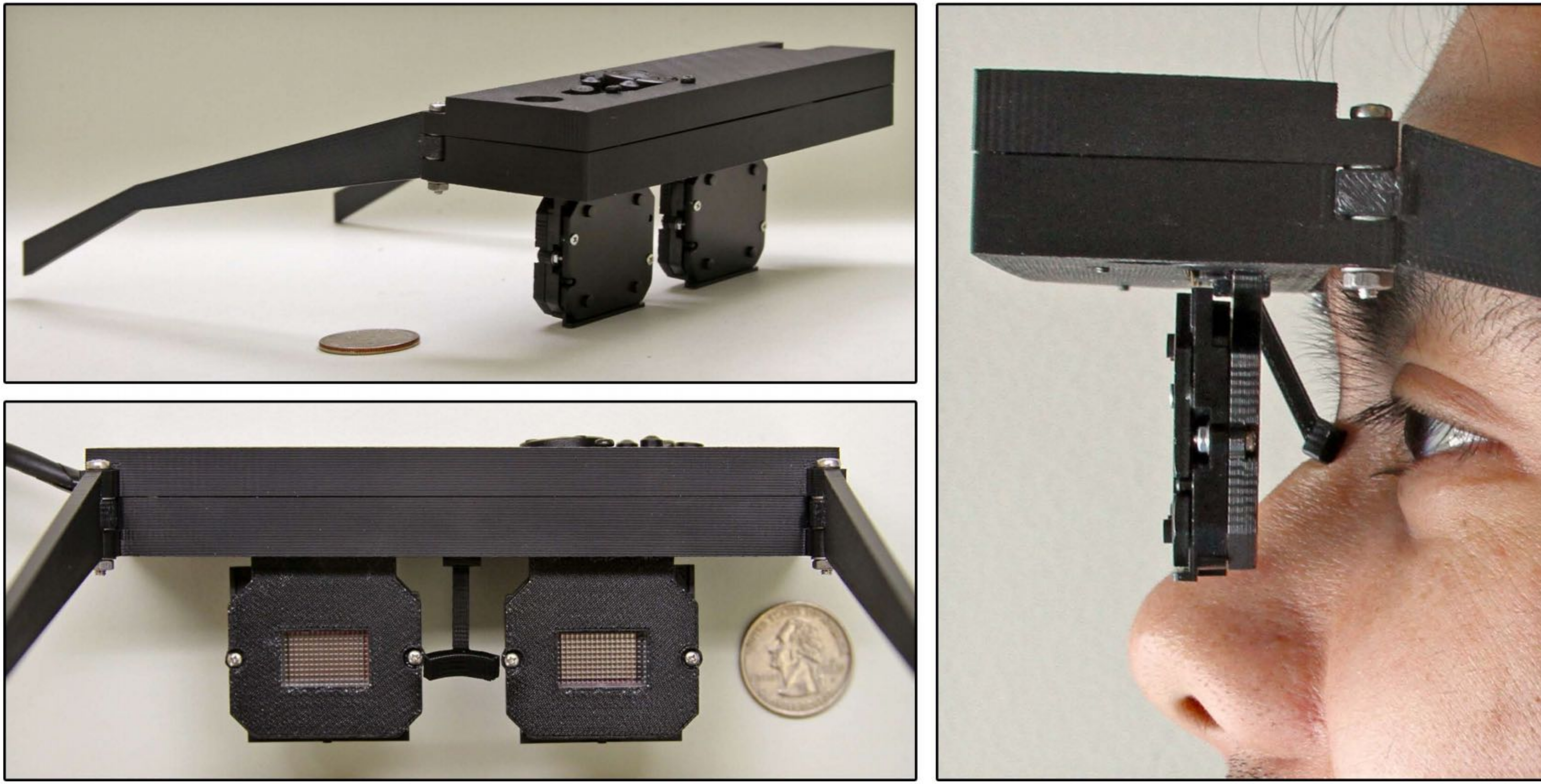
- **Given design of current VR displays, consider what happens when objects are up-close to eye in virtual scene**
  - **Eyes must remain accommodated to near infinity (otherwise image on screen won't be in focus)**
  - **But eyes must converge in attempt to fuse stereoscopic images of object up close**
  - **Brain receives conflicting depth clues… (discomfort, fatigue, nausea)**



This problem stems from nature of display design. If you could just make a display that emits the same rays of light that would be produced by a virtual scene, then you could avoid the accommodation - vergence conflict…

Aside: near-eye “light field” displays

Attempt to recreate same magnitude and direction of rays of light as produced by being in a real world scene.



Acquiring VR content

Google's Jump VR video:
Yi Halo Camera (17 cameras)

Facebook Manifold
(16 8K cameras)

Name of the game, part 1: low latency

- The goal of a VR graphics system is to achieve "presence", tricking the brain into thinking what it is seeing is real

- Achieving presence requires an exceptionally low-latency system
  - What you see must change when you move your head!
  - End-to-end latency: time from moving your head to the time new photons hit your eyes
    - Measure user's head movement
    - Update scene/camera position
    - Render new image
    - Transfer image to headset, then to transfer to display in headset
    - Actually emit light from display (photons hit user's eyes)
  - Latency goal of VR: 10-25 ms
    - Requires exceptionally low-latency head tracking
    - Requires exceptionally low-latency rendering and display

Thought experiment: effect of latency

- Consider a 1,000 x 1,000 display spanning 100° field of view
  - 10 pixels per degree

- Assume:
  - You move your head 90° in 1 second (only modest speed)
  - End-to-end latency of graphics system is 33 ms (1/30 sec)
    - In other words, the time from you moving you head to the display emitting light for a frame that reflects that movement.

- Therefore:
  - Displayed pixels are off by 3° ~ 30 pixels from where they would be in an ideal system with 0 latency

Example credit: Michael Abrash

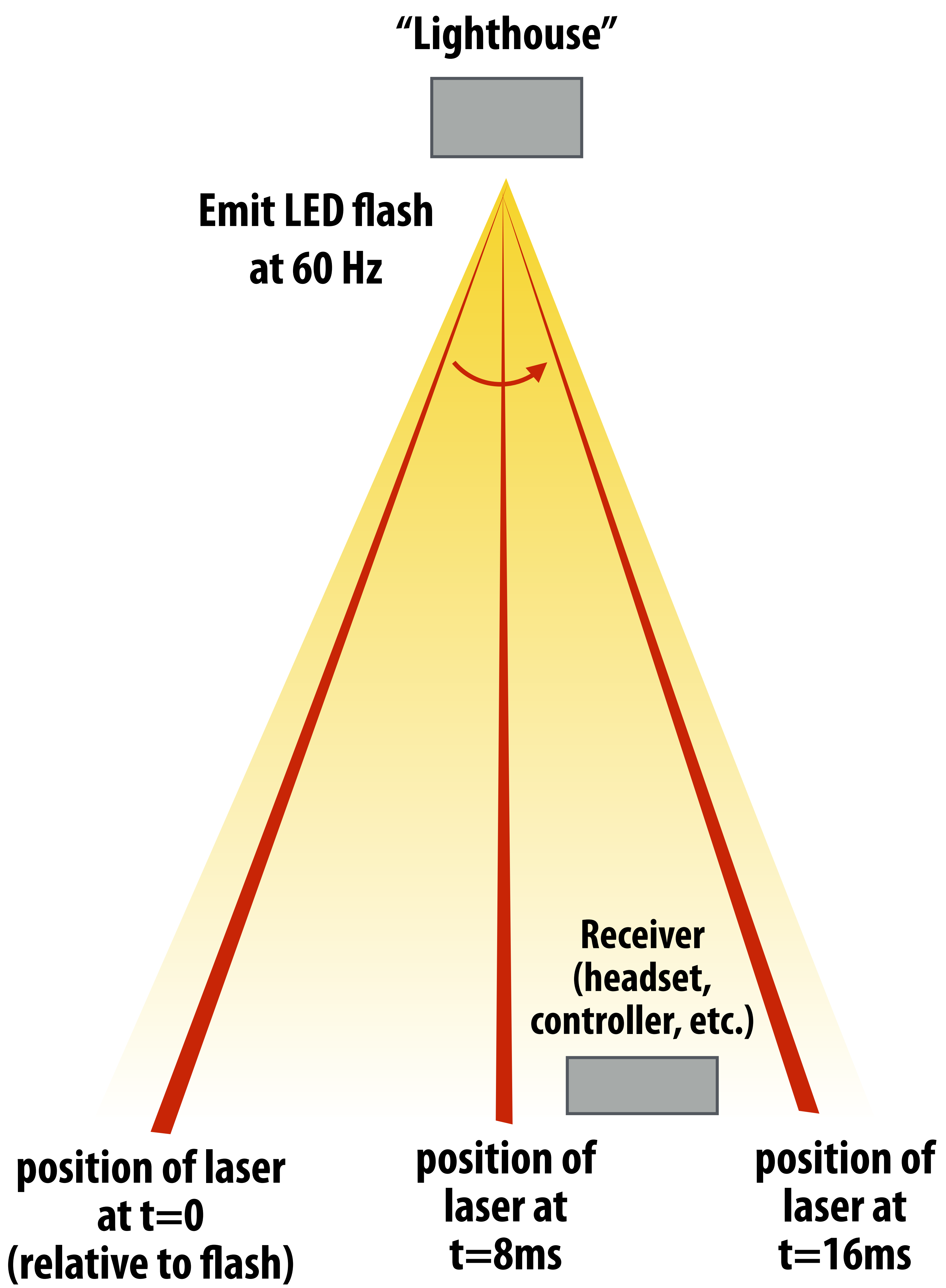
Oculus CV1 IR camera and IR LEDs

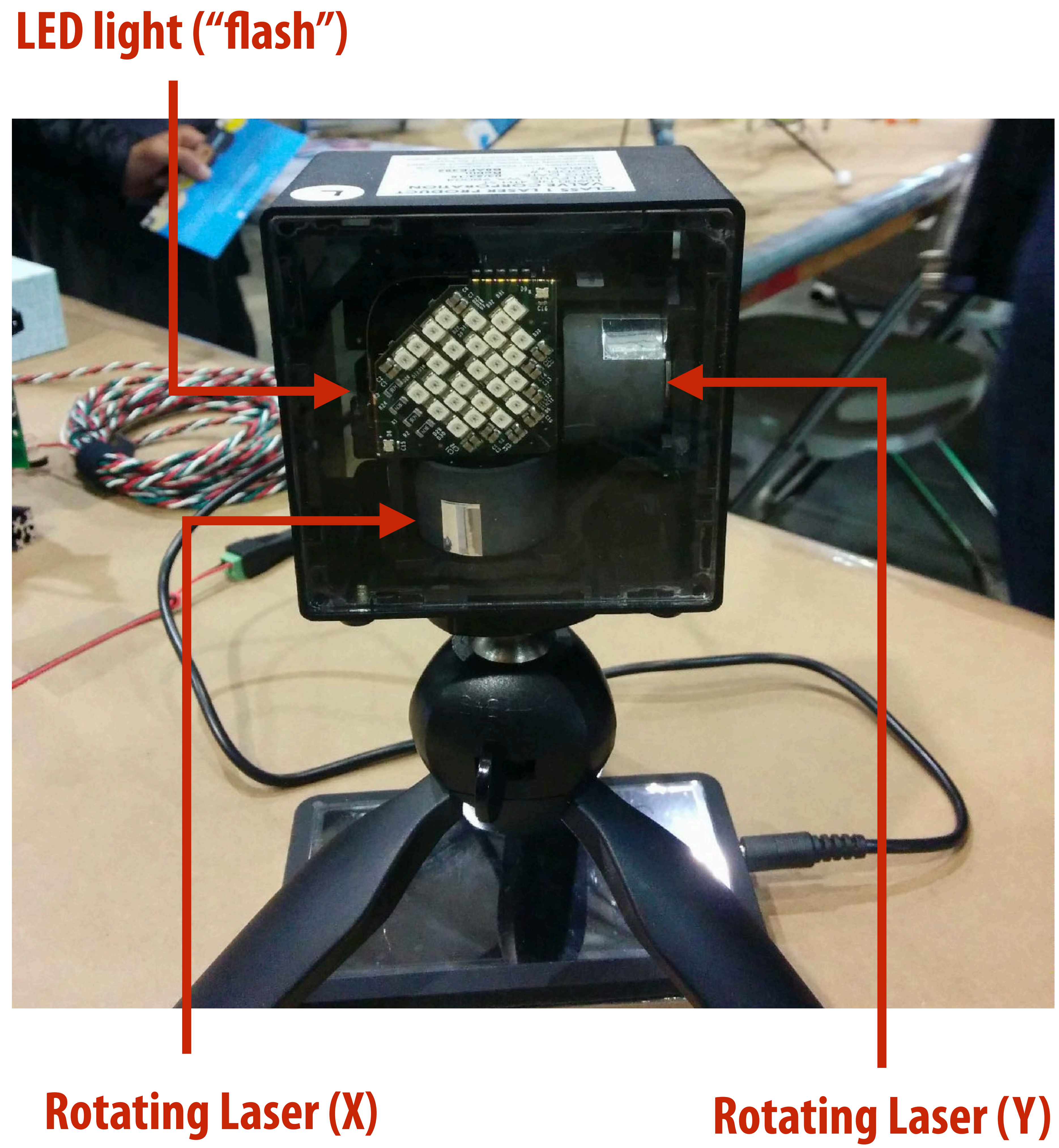
Headset contains:
IR LEDs (tracked by camera)
Gyro + accelerometer (1000Hz). (rapid relative positioning)

60Hz IR Camera
(measures absolute position of headset 60 times a second)

Image credit: ifixit.com

Valve's Lighthouse: cameraless position tracking



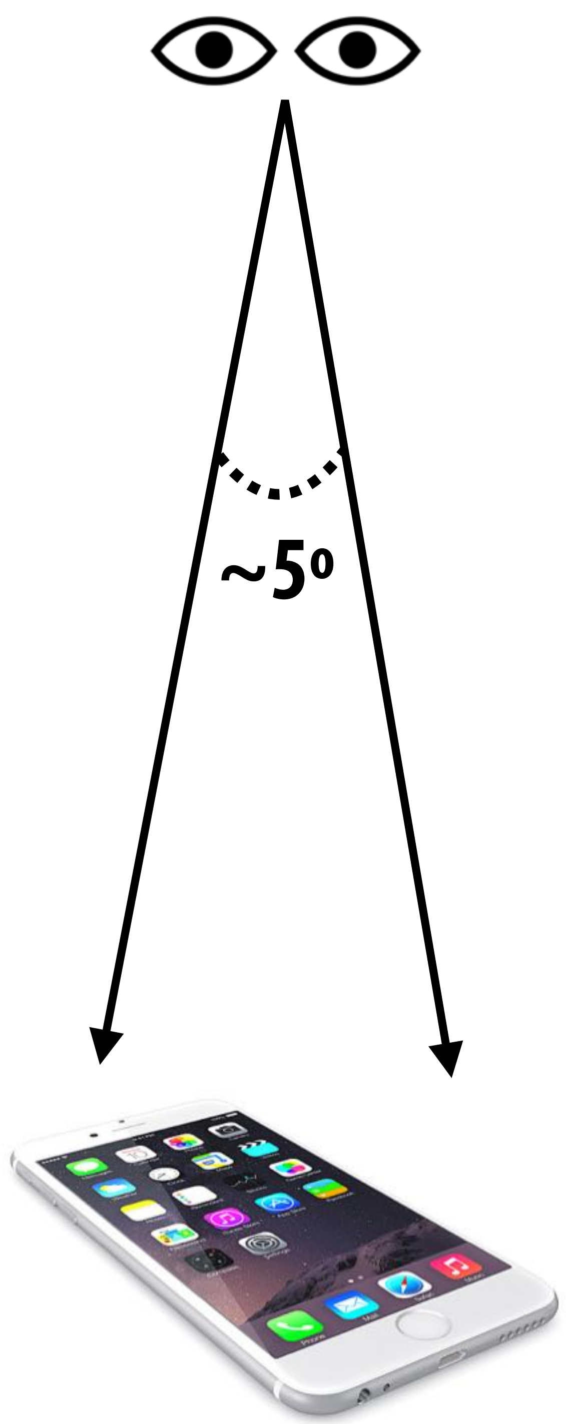


No need for computer vision processing to compute position of receiver: just a light sensor and an accurate clock!

Image credit: Travis Deyle
http://www.hizook.com/blog/2015/05/17/valves-lighthouse-tracking-system-may-be-big-news-robotics

Accounting for resolution of eye

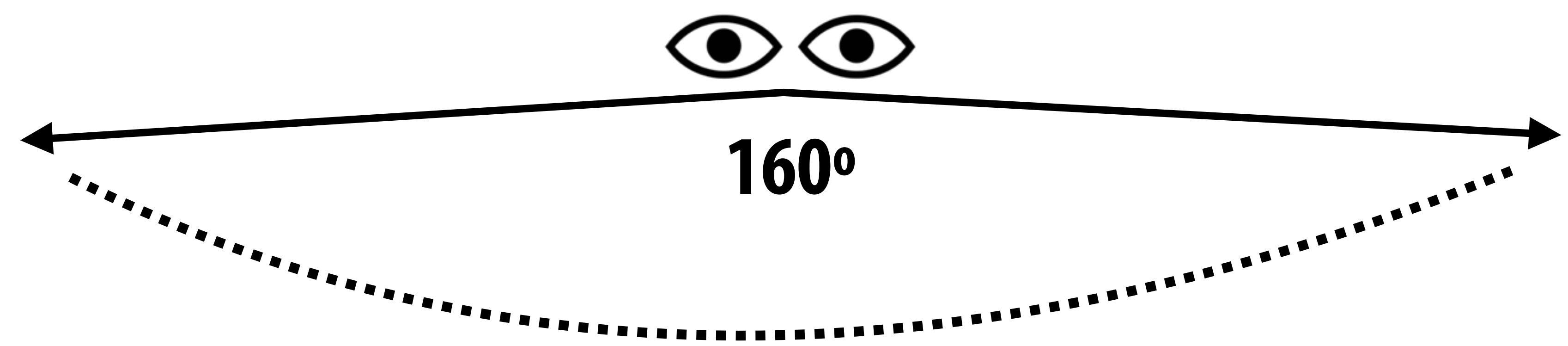
Name of the game, part 2: high resolution



iPhone 7: 4.7 in "retina" display:

1,334 x 750 (1 Mpixel)

326 ppi → 65 ppd



Human: ~160° view of field per eye (~200° overall)

(Note: this does not account for eye's ability to rotate in socket)

Future "retina" VR display:

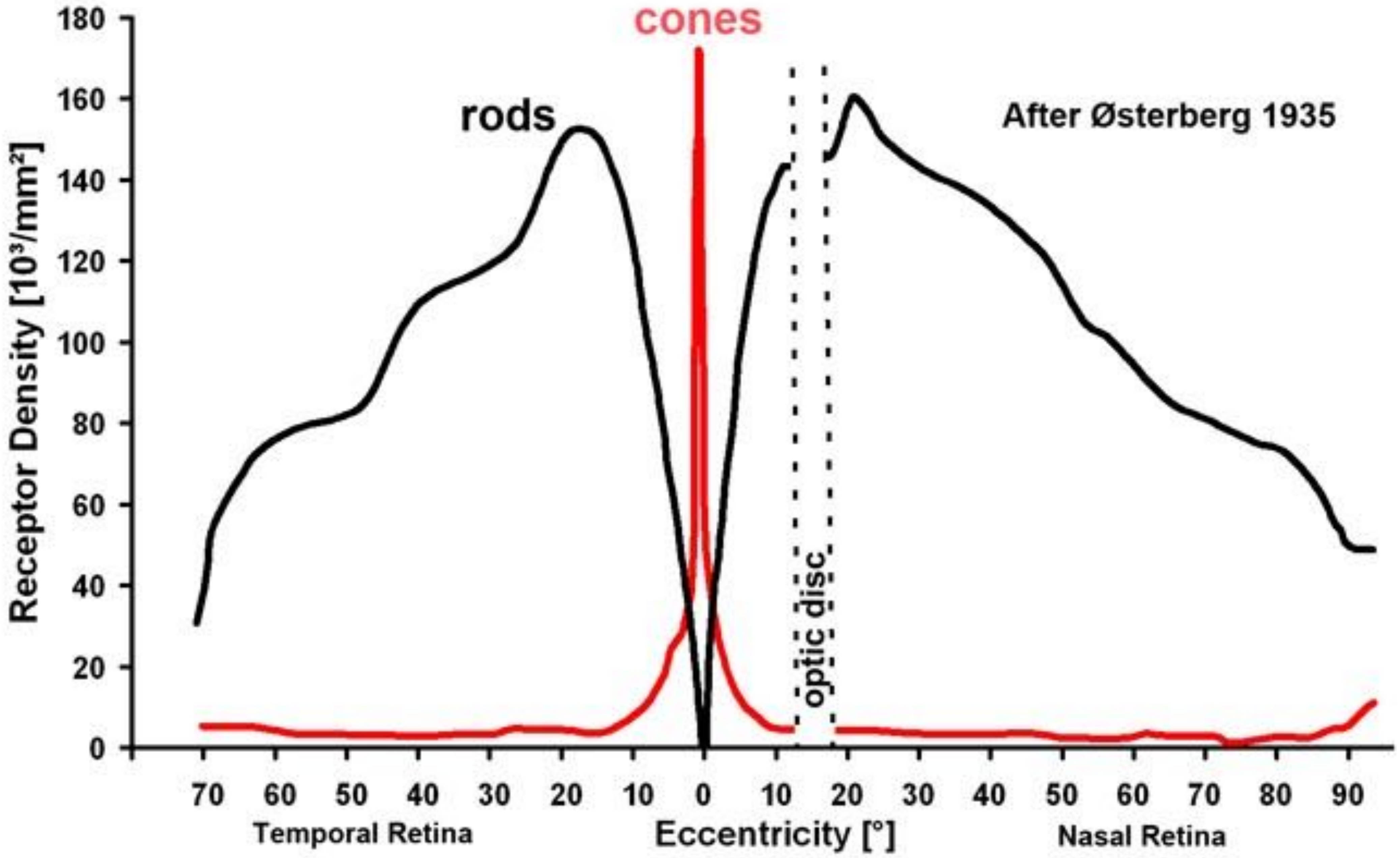
65 ppd covering 200°

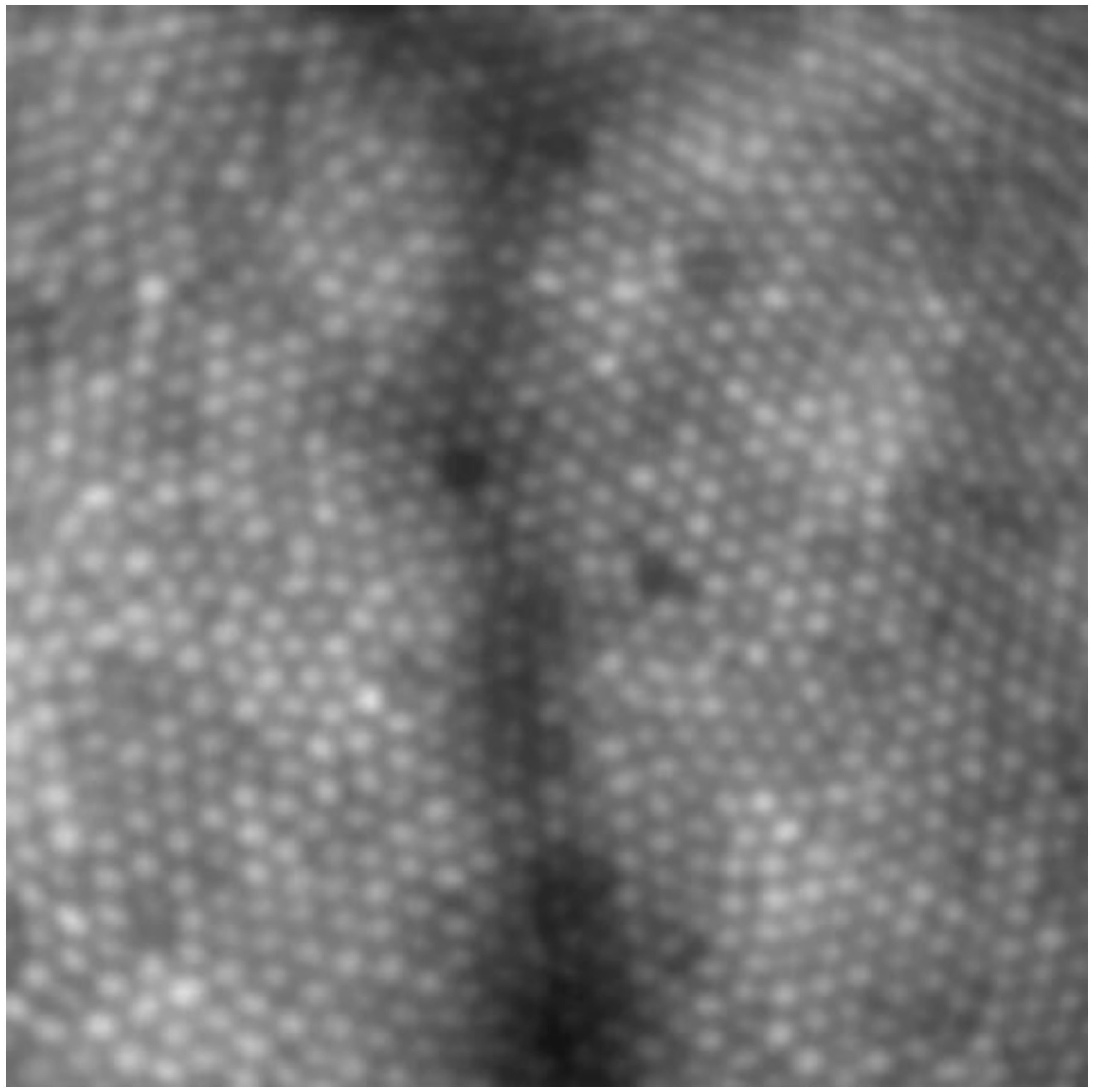
= 13K x 13K display per eye

= 170 MPixel per eye

Strongly suggests need for eye tracking and foveated rendering (eye can only perceive detail in 5° region about gaze point)

Density of rod and cone cells in the retina



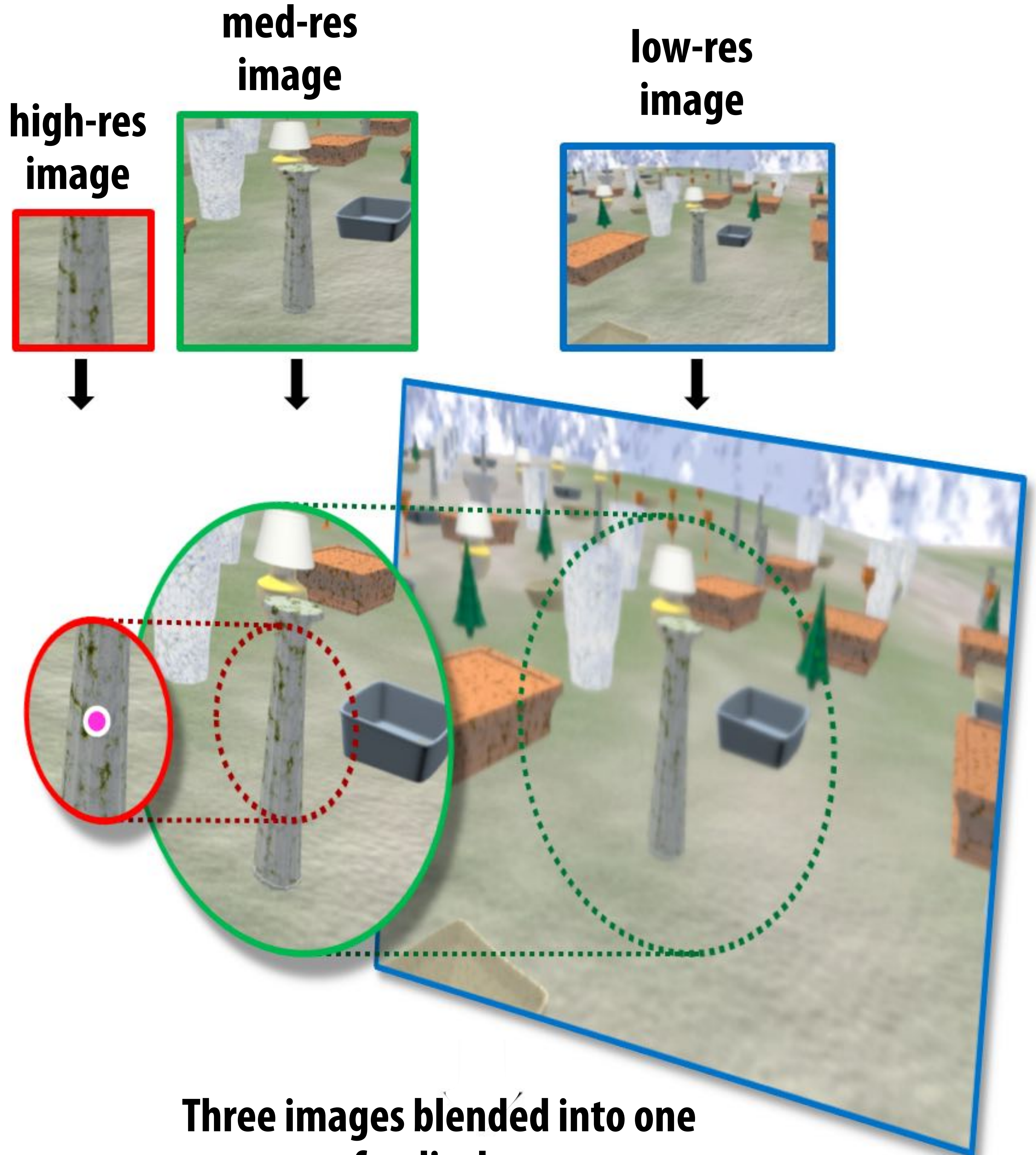


[Roorda 1999]

- Cones are color receptive cells
- Highest density of cones is in fovea
  (best color vision at center of where human is looking)

Addressing high resolution and high field of view: foveated rendering

Idea: track user's gaze, render with increasingly lower resolution farther away from gaze point



Three images blended into one for display

Traditional rendering (uniform screen sampling)

[Patney et al. 2016]

Low-pass filter away from fovea

In this image, gaussian blur with radius dependent on distance from fovea is used to remove high frequencies

[Patney et al. 2016]

Contrast enhance periphery

Eye is receptive to contrast at periphery

[Patney et al. 2016]

Accounting for distortion due to design of head-mounted display

Lenses introduce distortion

Lenses introduce distortion

- Pincushion distortion
- Chromatic aberration (different wavelengths of light refract by different amount)

View of checkerboard through Oculus Rift lens

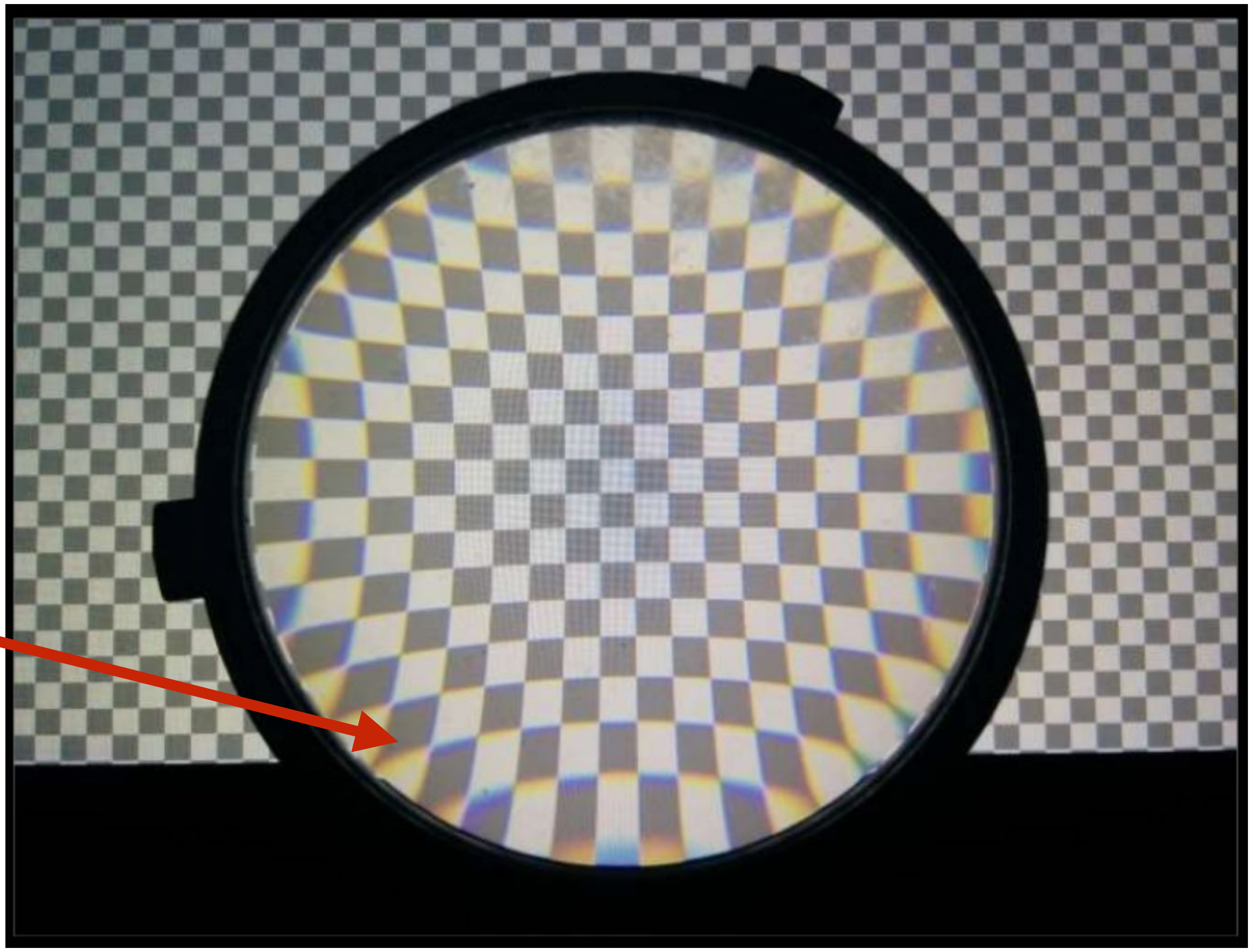
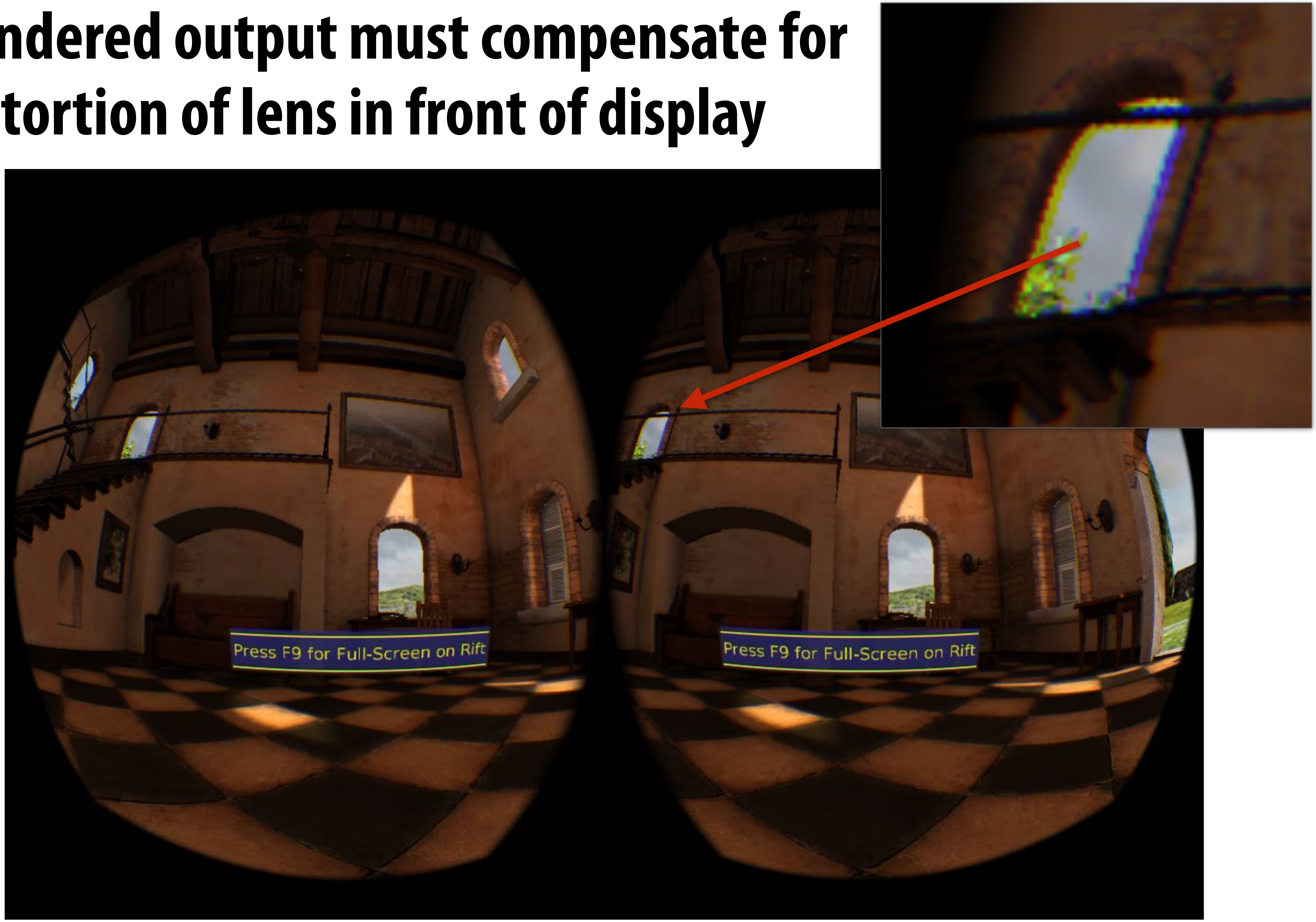


Image credit: Cass Everitt

Rendered output must compensate for distortion of lens in front of display



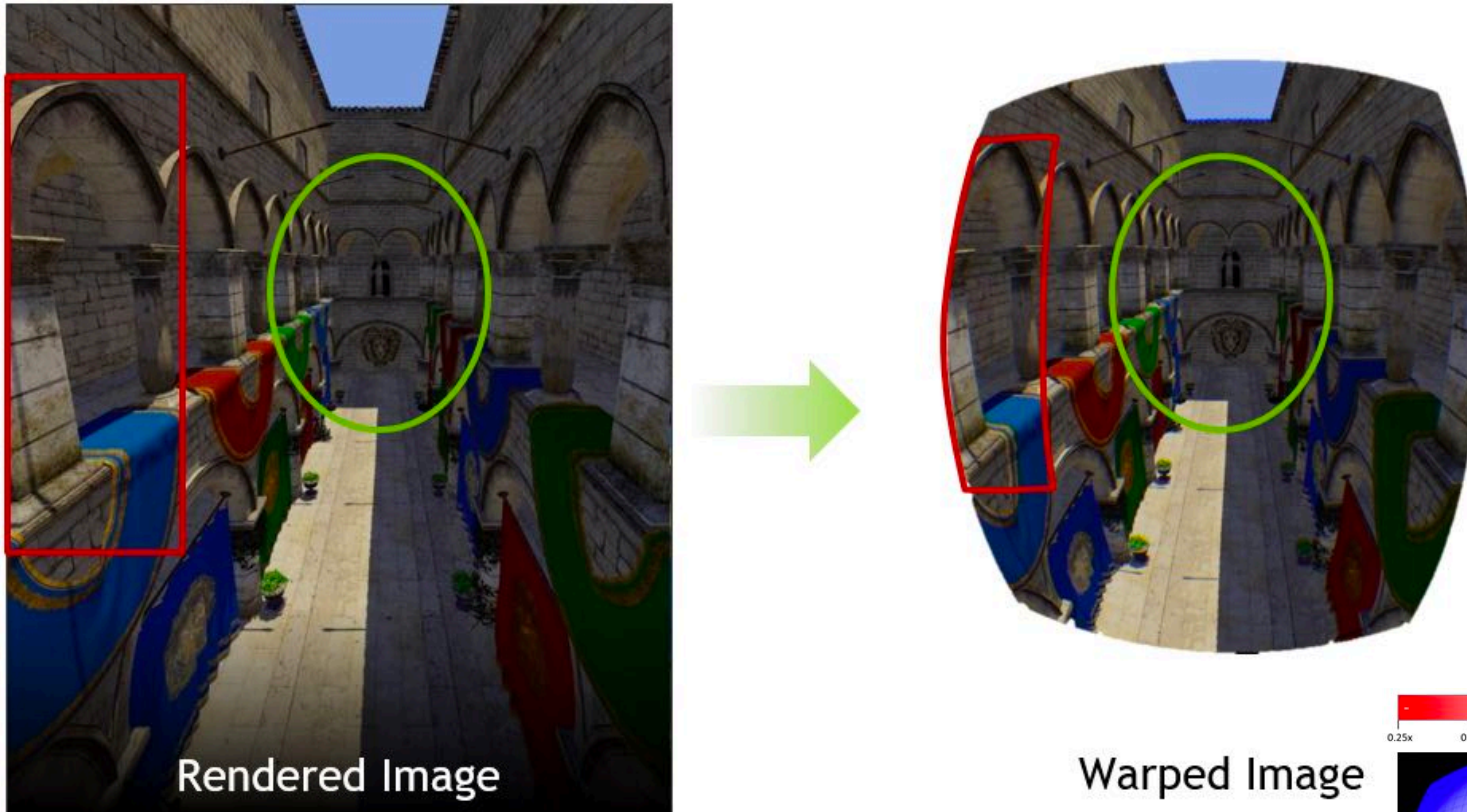
Step 1: render scene using traditional graphics pipeline at full resolution for each eye

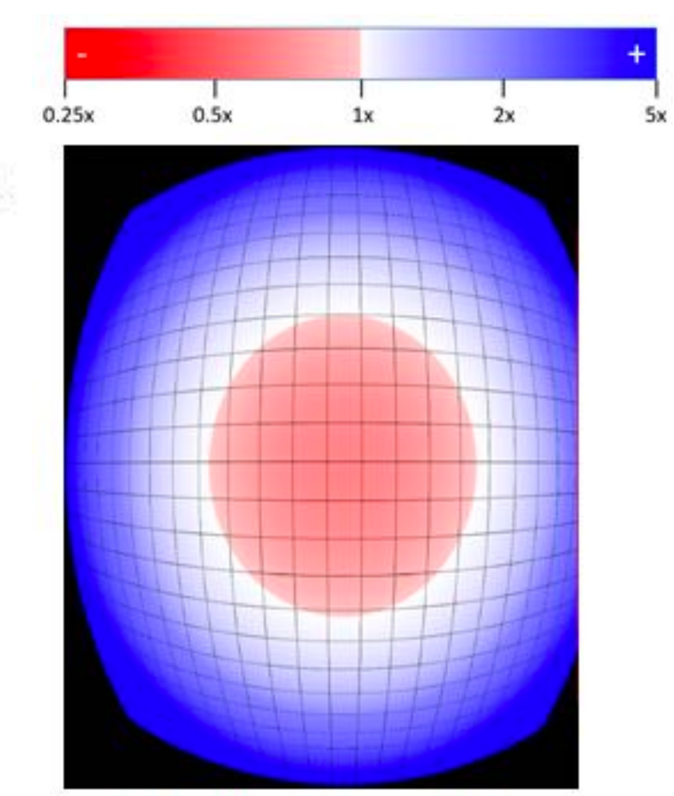
Step 2: warp images so rendering is viewed correctly when screen viewed under lens distortion

(Can apply different distortion to R, G, B to approximate correction for chromatic aberration)

Image credit: Oculus VR developer guide

Problem: oversampling at periphery





Shading Rate After Lens Warp

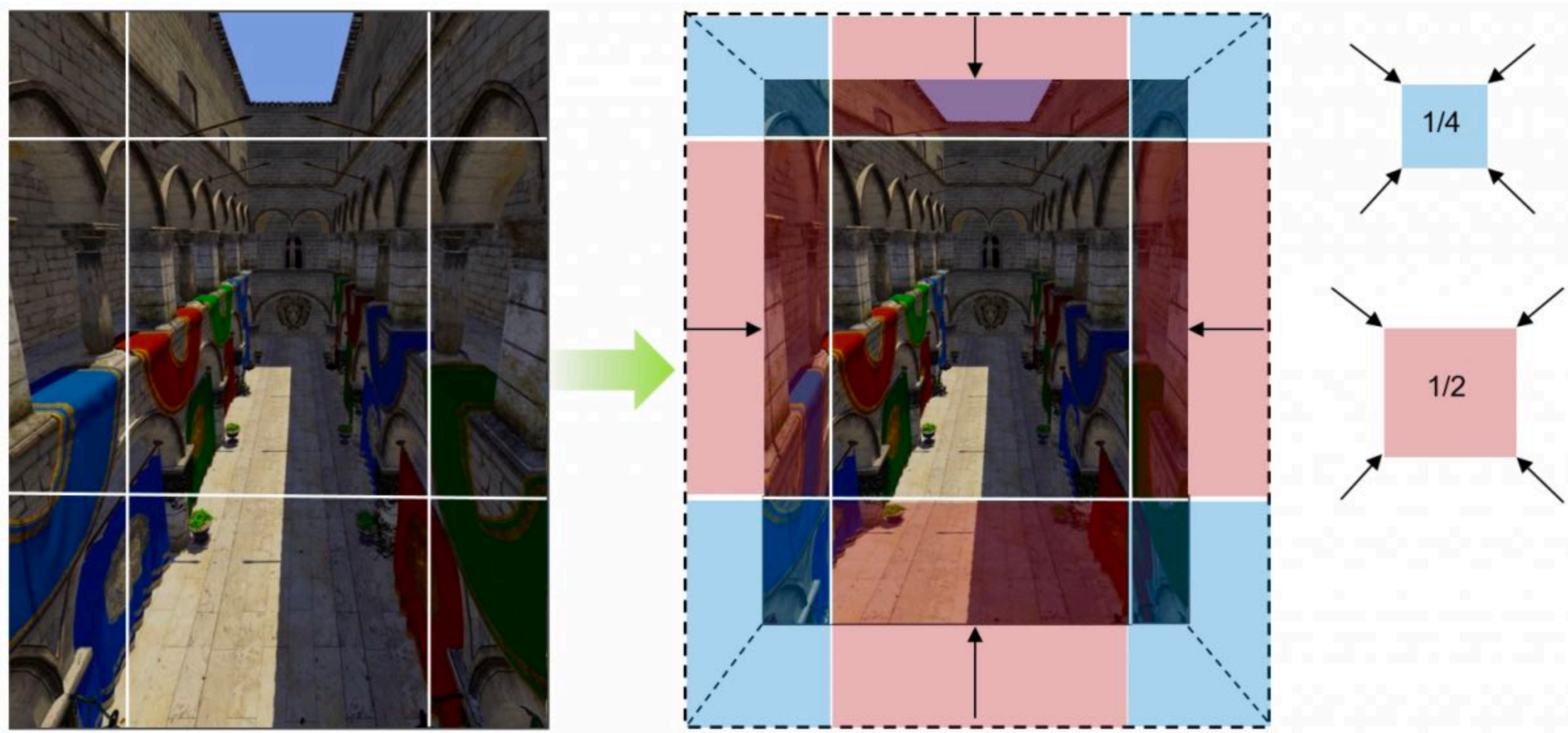
Due to:

Warp to reduce optical distortion (sample shading densely in the periphery)

Also recall eye has less spatial resolution in periphery (assuming viewer's gaze is toward center of screen)

[Image credit: NVIDIA]

Multi viewport rendering



Render the scene once, but graphics pipeline using different sampling rates for different screen regions ("viewports")

[Image credit: NVIDIA]

Lens matched shading

- Render with four viewports
- Modify w prior to homogeneous divide as: $w' = w + Ax + By$
- "Compresses" scene in the periphery (fewer samples), while not affecting scene near center of field of view

Original Viewport

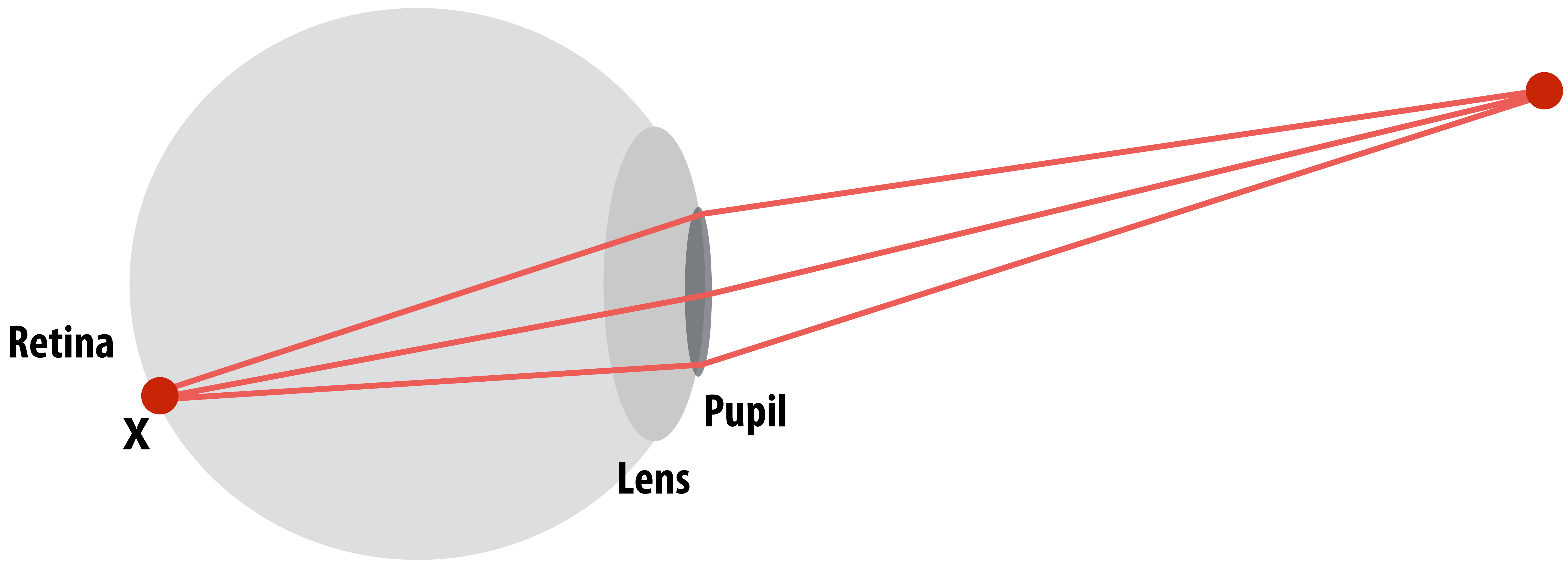
Enlarged Viewport
Shading Rate Increased

With Modifed W
Periphery Shading Reduced
Center Shading Rate Still Increased
Overall Shading Reduced

[Image credit: NVIDIA]

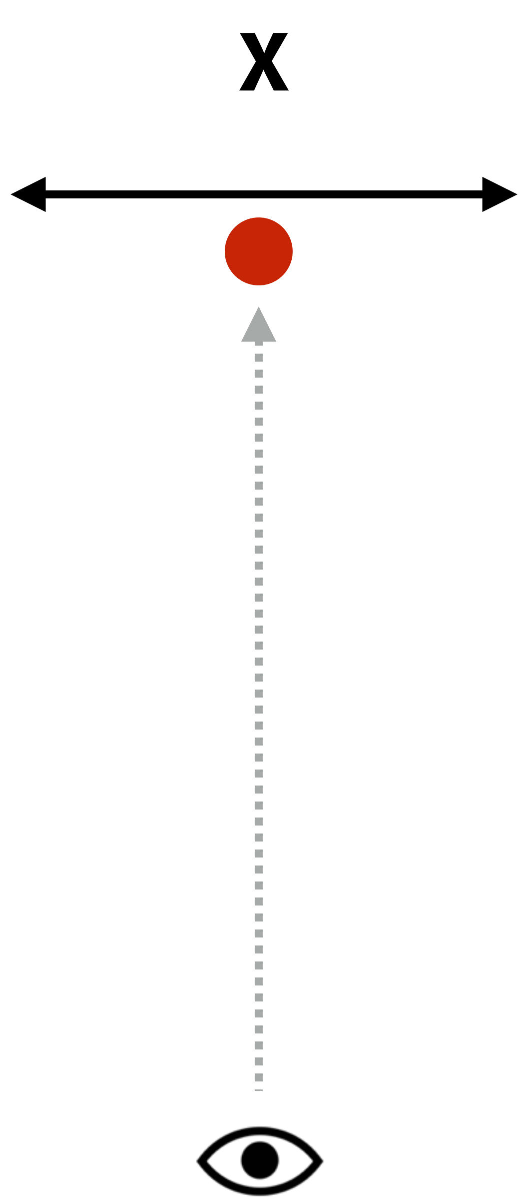
Accounting for interaction of:
display update +
display attached to head

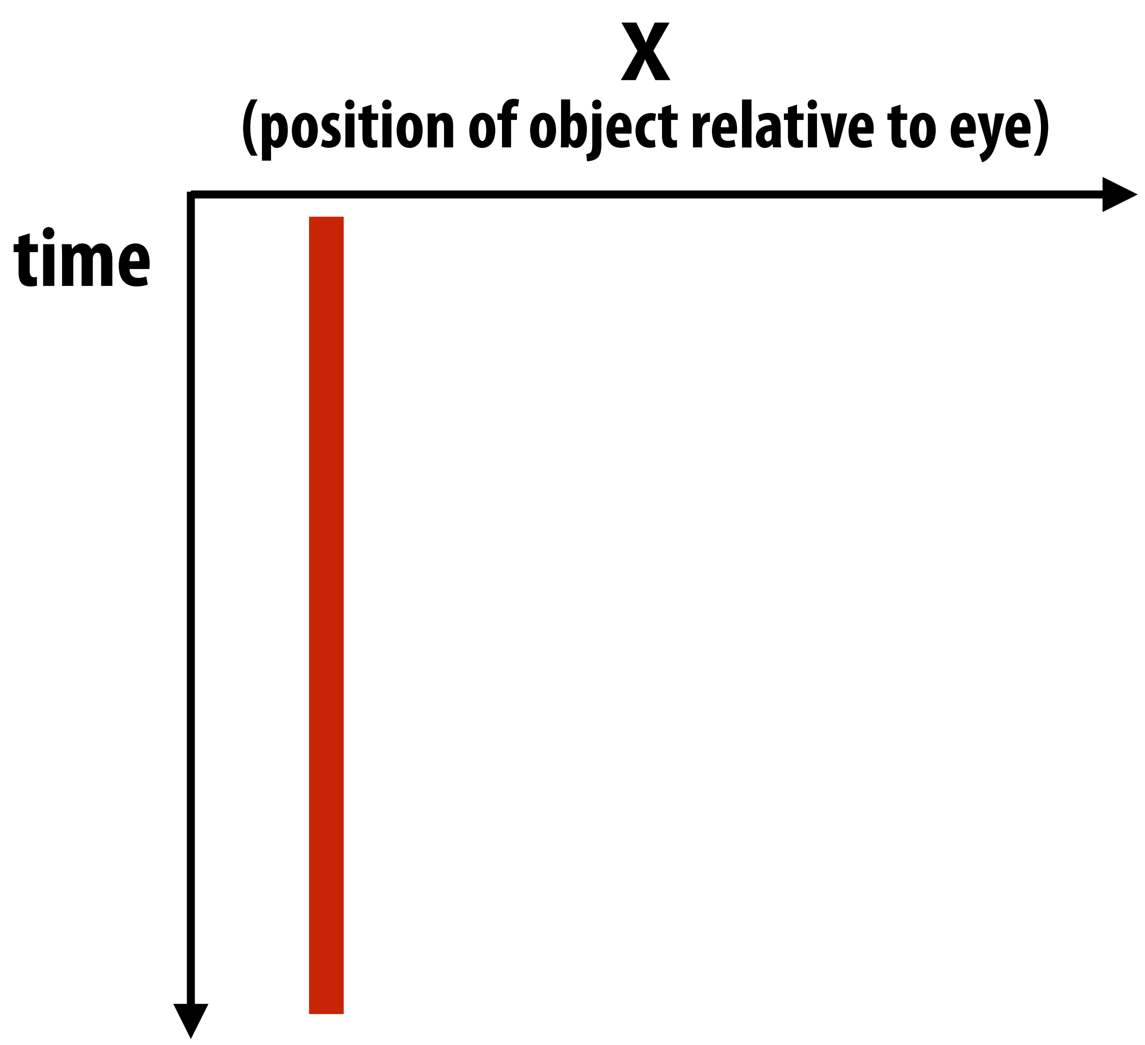
Consider projection of scene object on retina



Here: object projects onto point X on back of eye (retina)

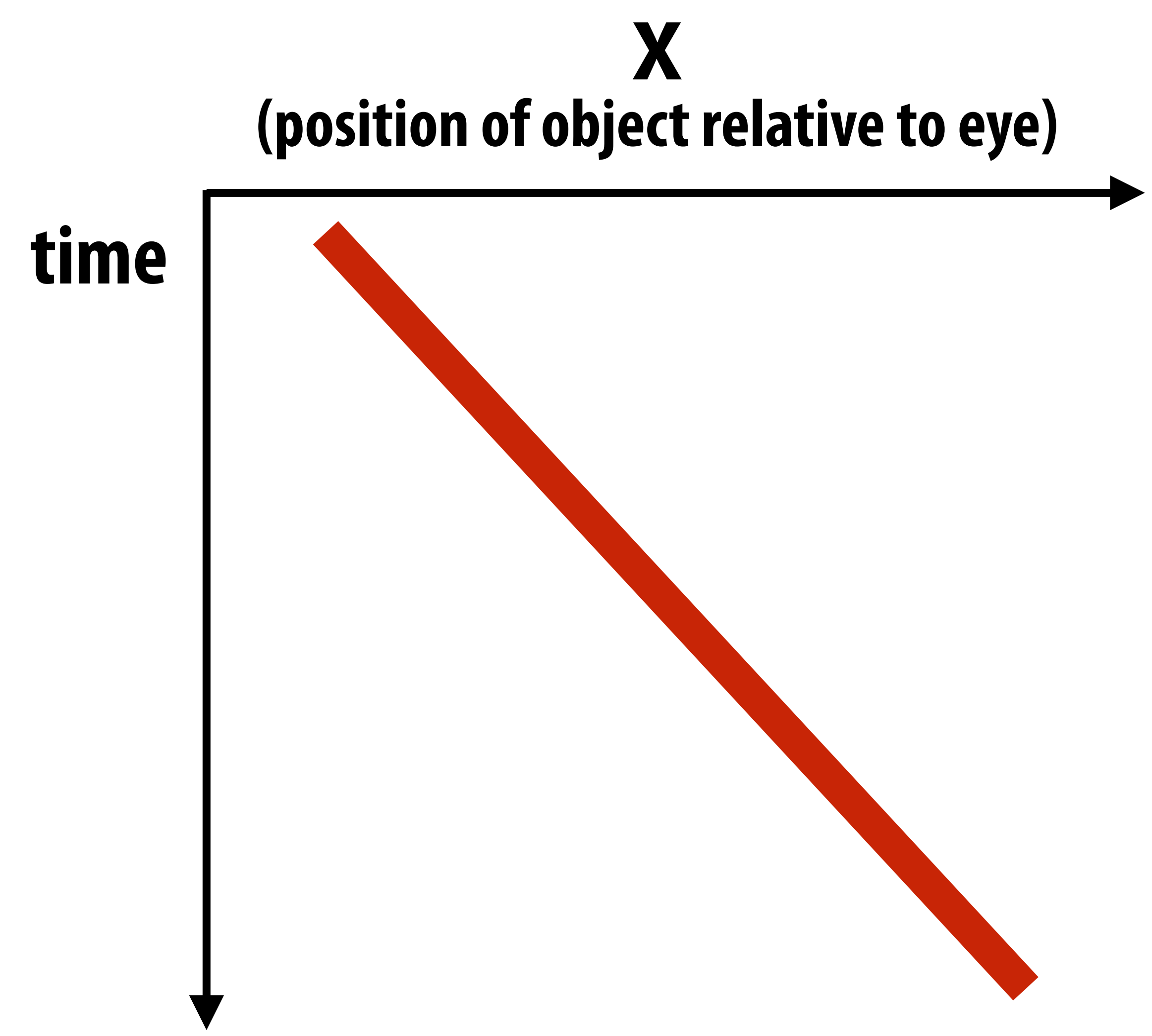
Consider object position relative to eye





Case 1: object stationary relative to eye:

(eye still and red object still
OR
red object moving left-to-right and
eye rotating to track object
OR
red object stationary in world but head moving
and eye rotating to track object)



Case 2: object moving relative to eye:

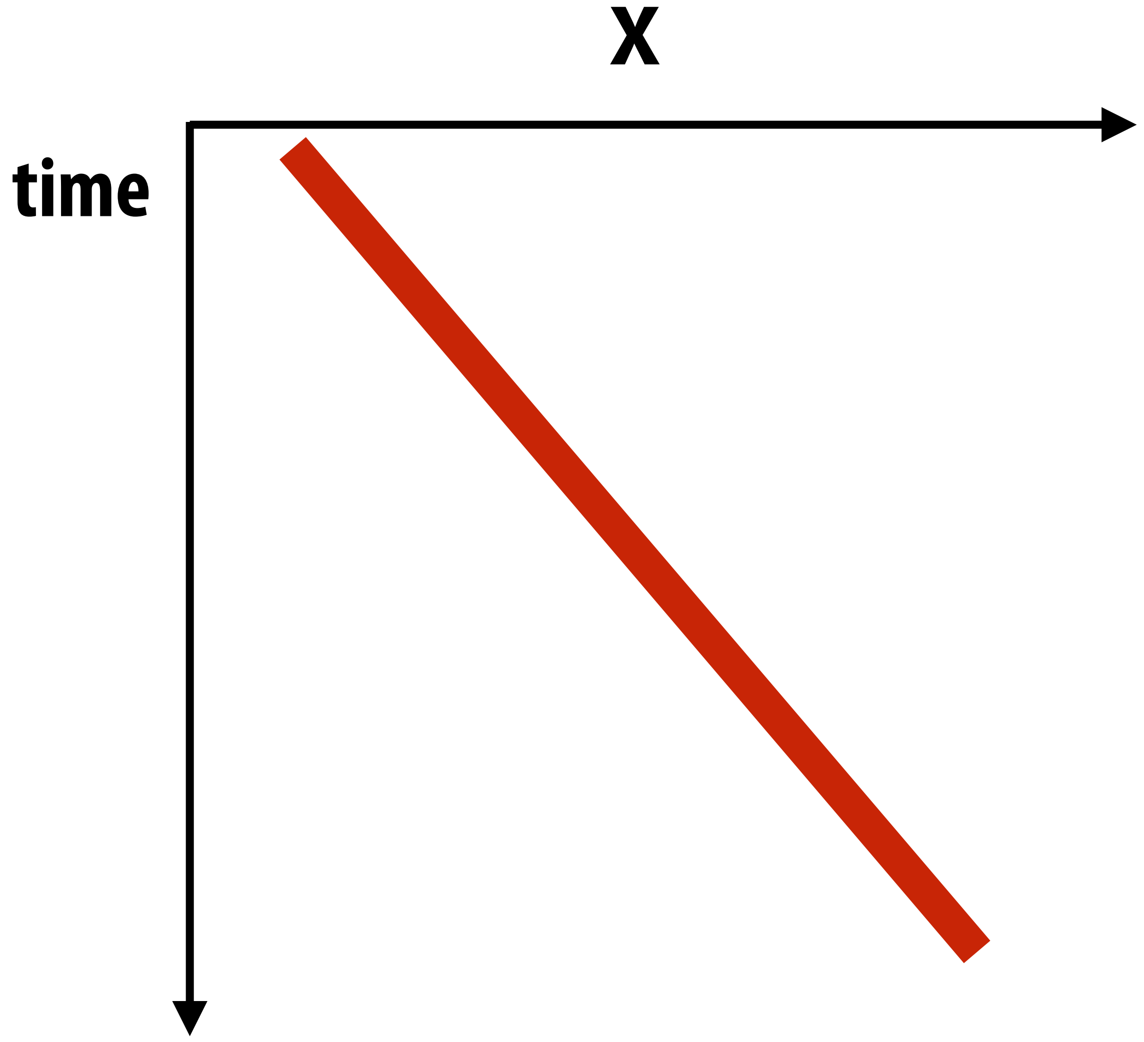
(red object moving from left to right but
eye stationary, i.e., it's focused on a different
stationary point in world)

NOTE: THESE GRAPHS PLOT <u>OBJECT POSITION</u> RELATIVE TO EYE

RAPID HEAD MOTION WITH EYES TRACKING A MOVING OBJECT IS A FORM OF CASE 1!!!

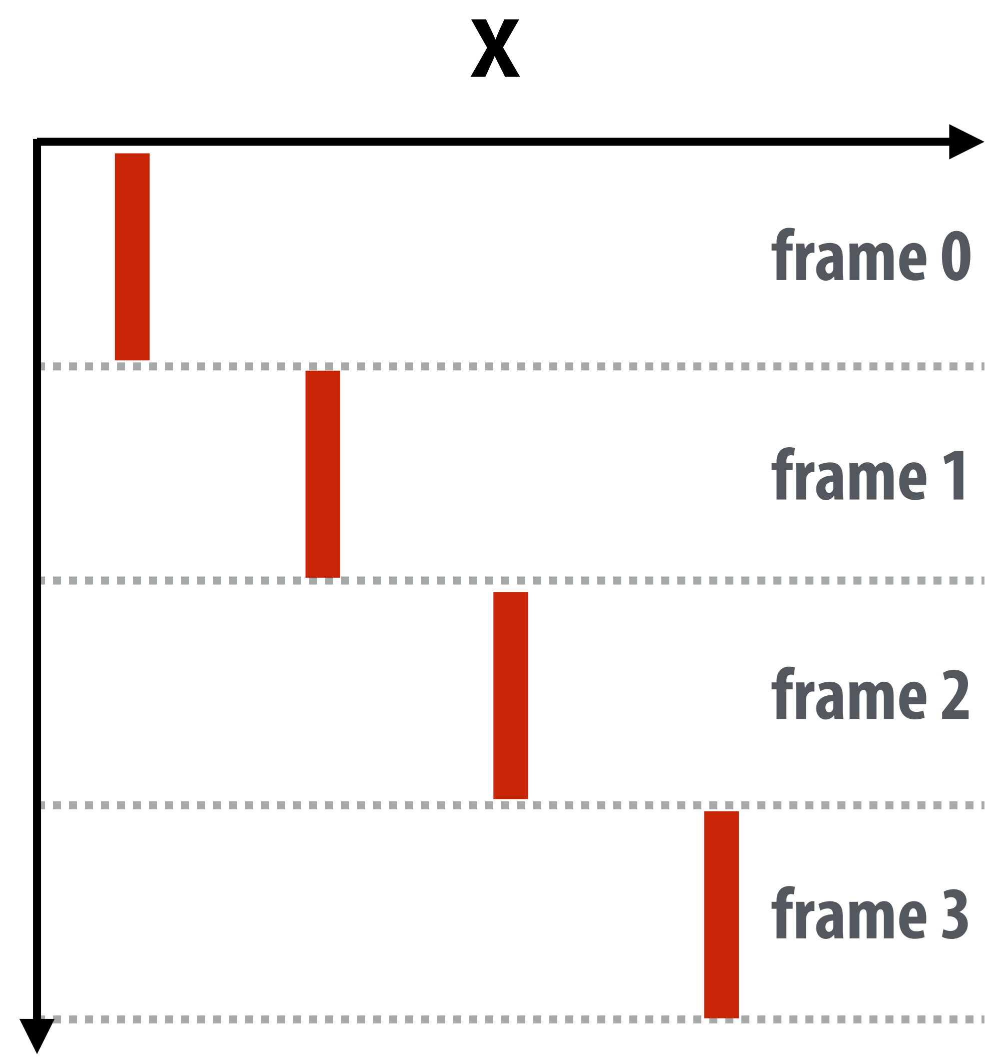
Spacetime diagrams adopted from presentations by Michael Abrash
Eyes designed by SuperAtic LABS from the thenounproject.com

Effect of latency: judder



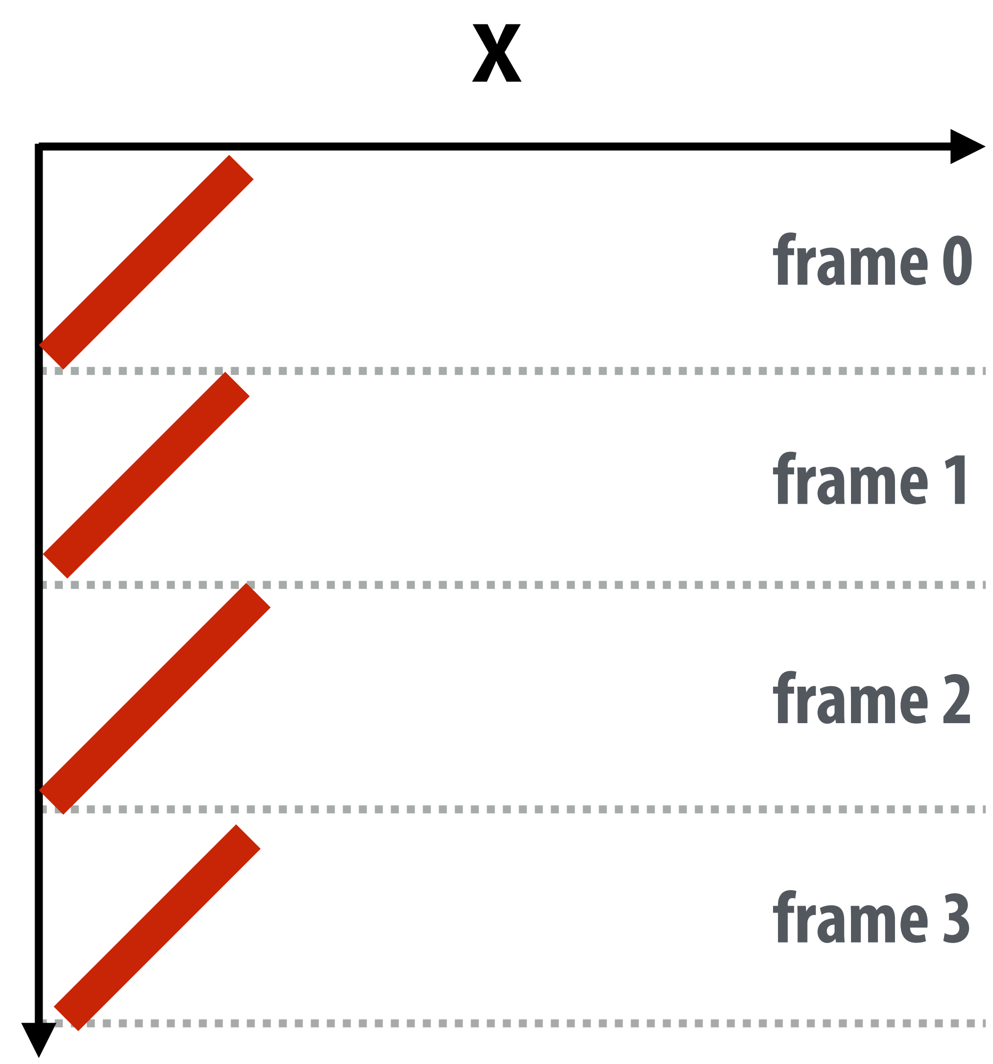
Case 2: object moving from left to right, eye stationary
(<u>eye stationary with respect to display</u>)

Continuous representation.



Case 2: object moving from left to right, eye stationary
(<u>eye stationary with respect to display</u>)

Light from display
(image is updated each frame)



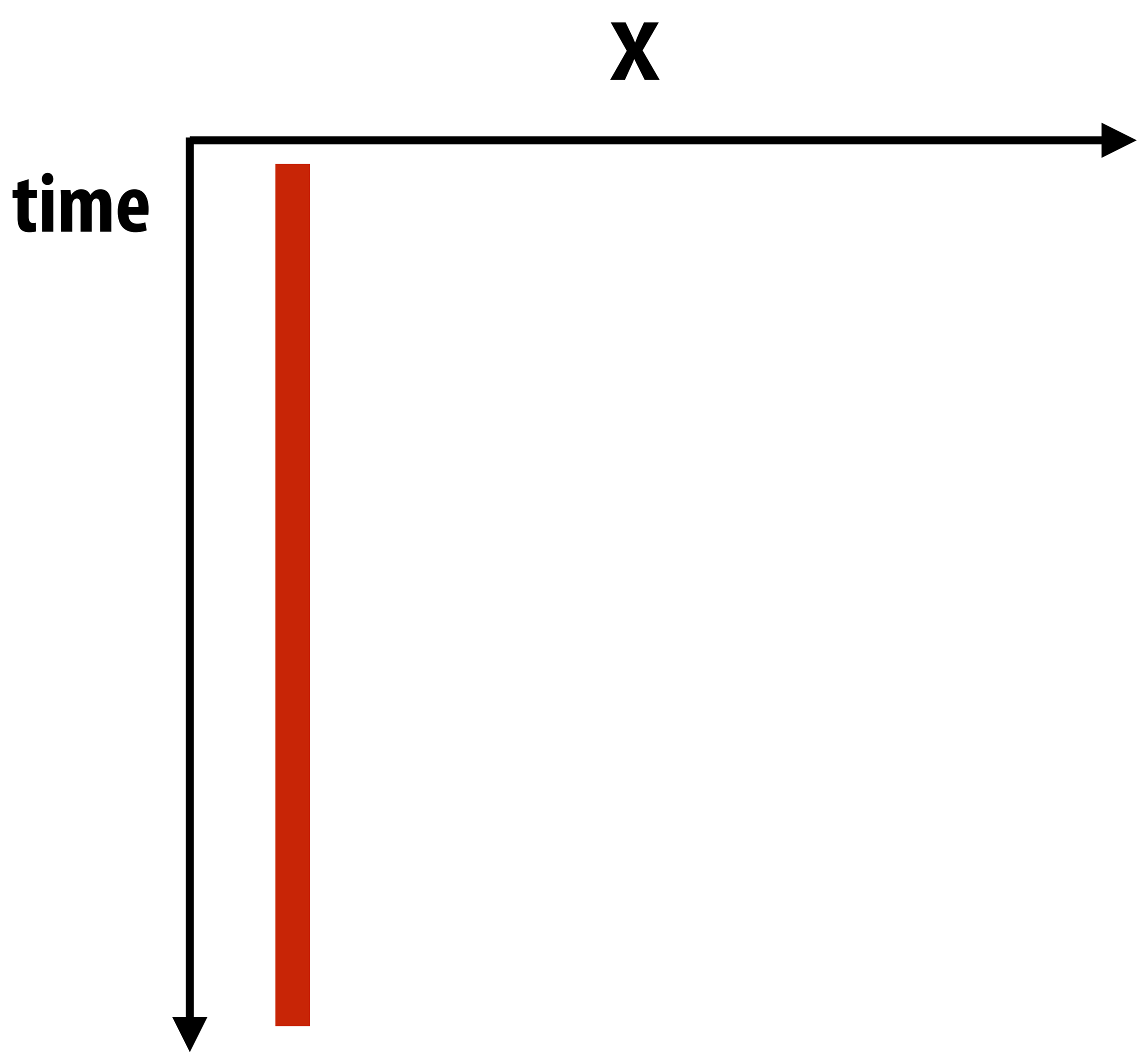
Case 1: object moving from left to right,
eye moving <u>continuously</u> to track object
(<u>eye moving relative to display!</u>)

Light from display
(image is updated each frame)

Case 1 explanation: since eye is moving, object's position is relatively constant relative to eye (as it should be since the eye is tracking it). But due discrete frame rate, object falls behind eye, causing a smearing/strobing effect ("choppy" motion blur). Recall from earlier slide: 90 degree motion, with 50 ms latency results in 4.5 degree smear

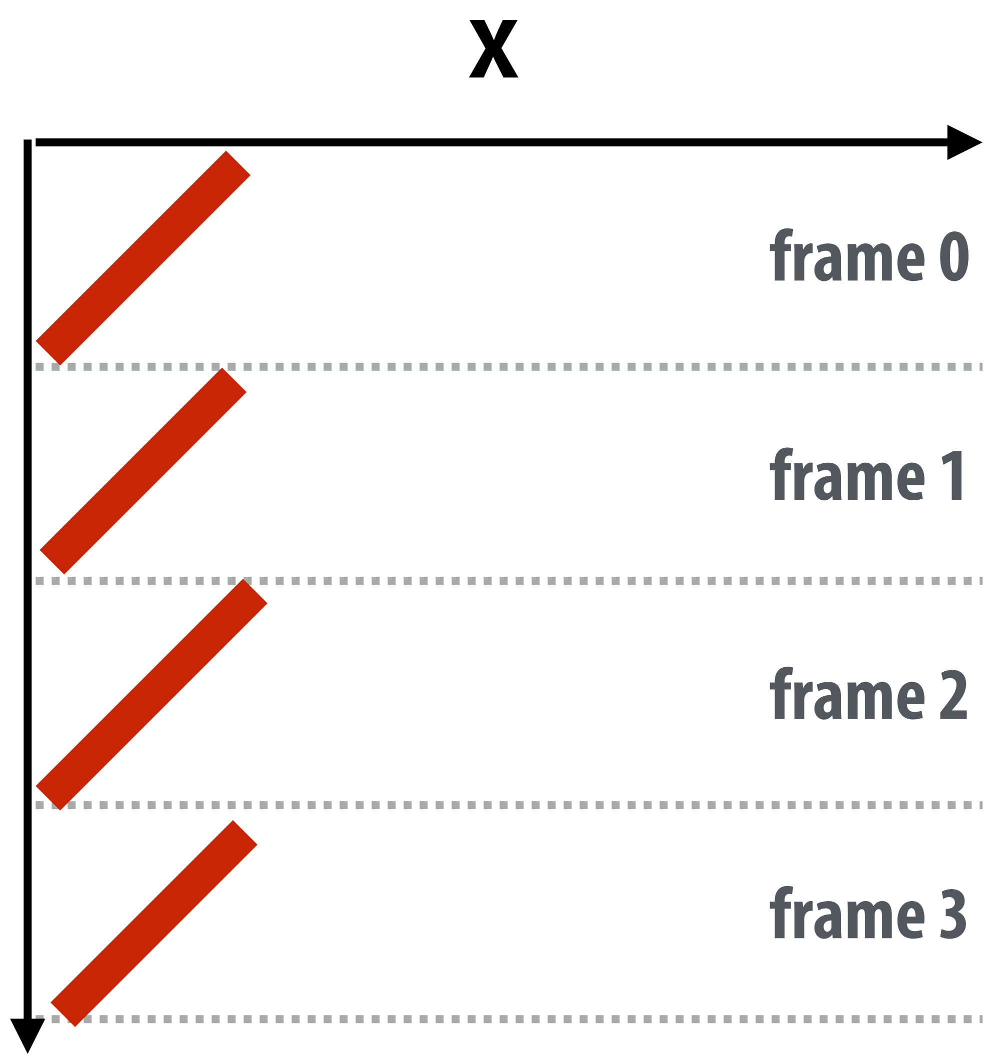
Spacetime diagrams adopted from presentations by Michael Abrash

Reducing judder: increase frame rate

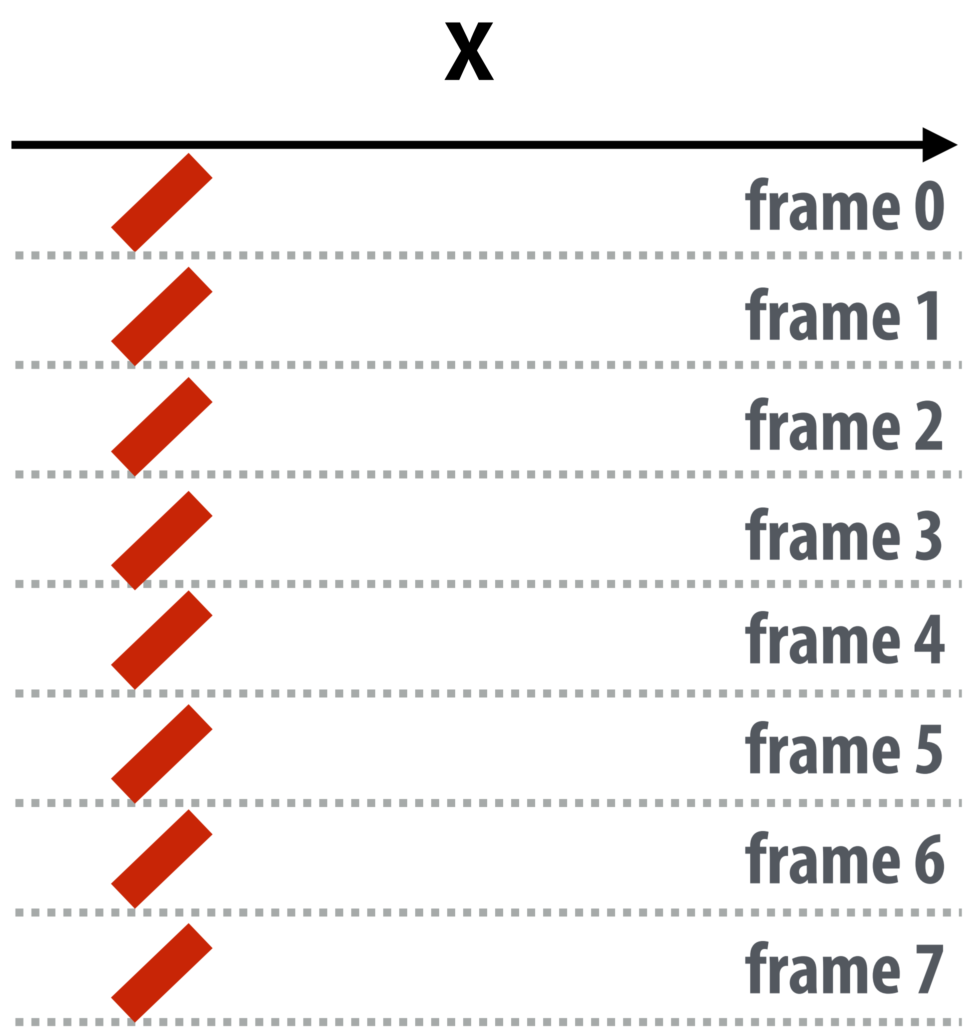


Case 1: continuous ground truth

red object moving left-to-right and eye moving to track object
OR
red object stationary but head moving and eye moving to track object



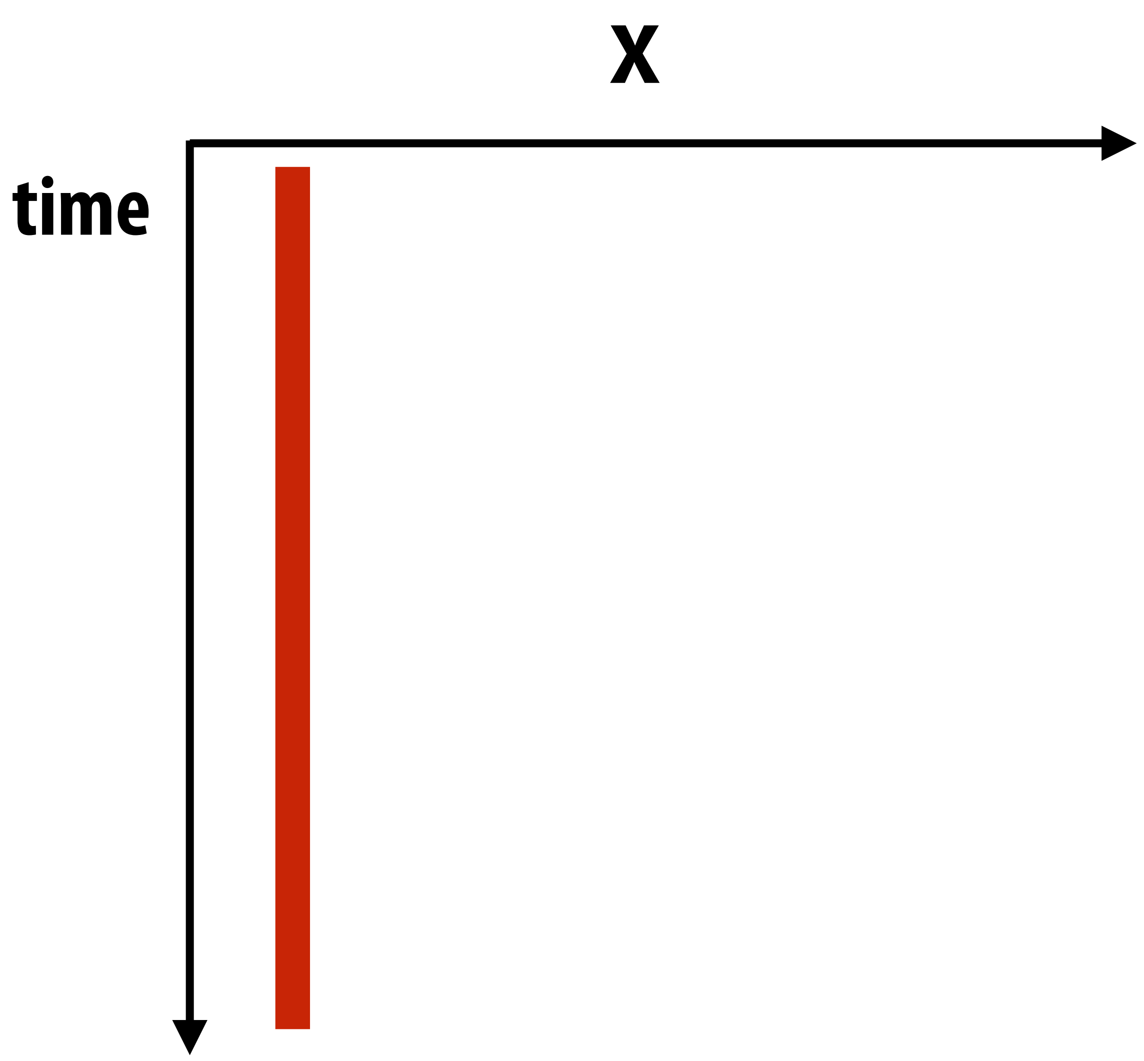
Light from display
(image is updated each frame)



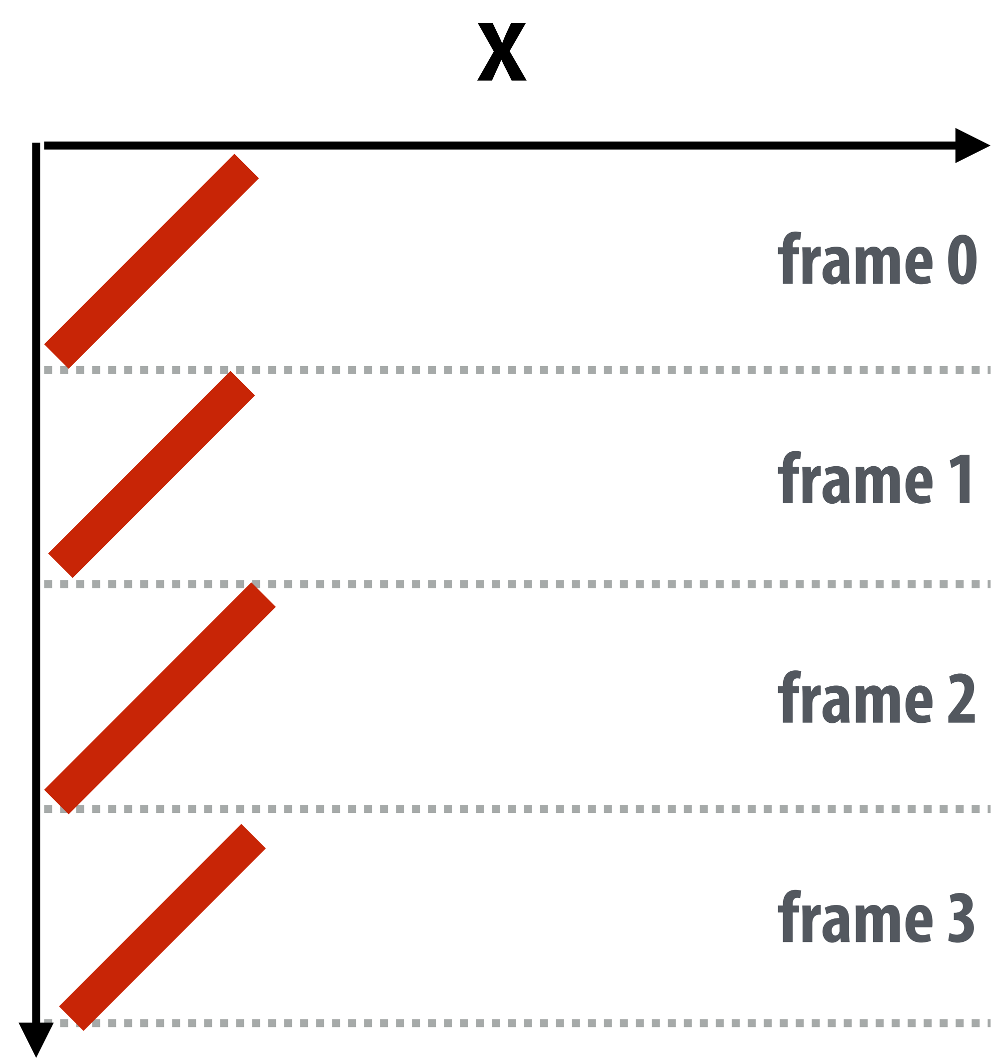
Light from display
(image is updated each frame)

Higher frame rate results in closer approximation to ground truth

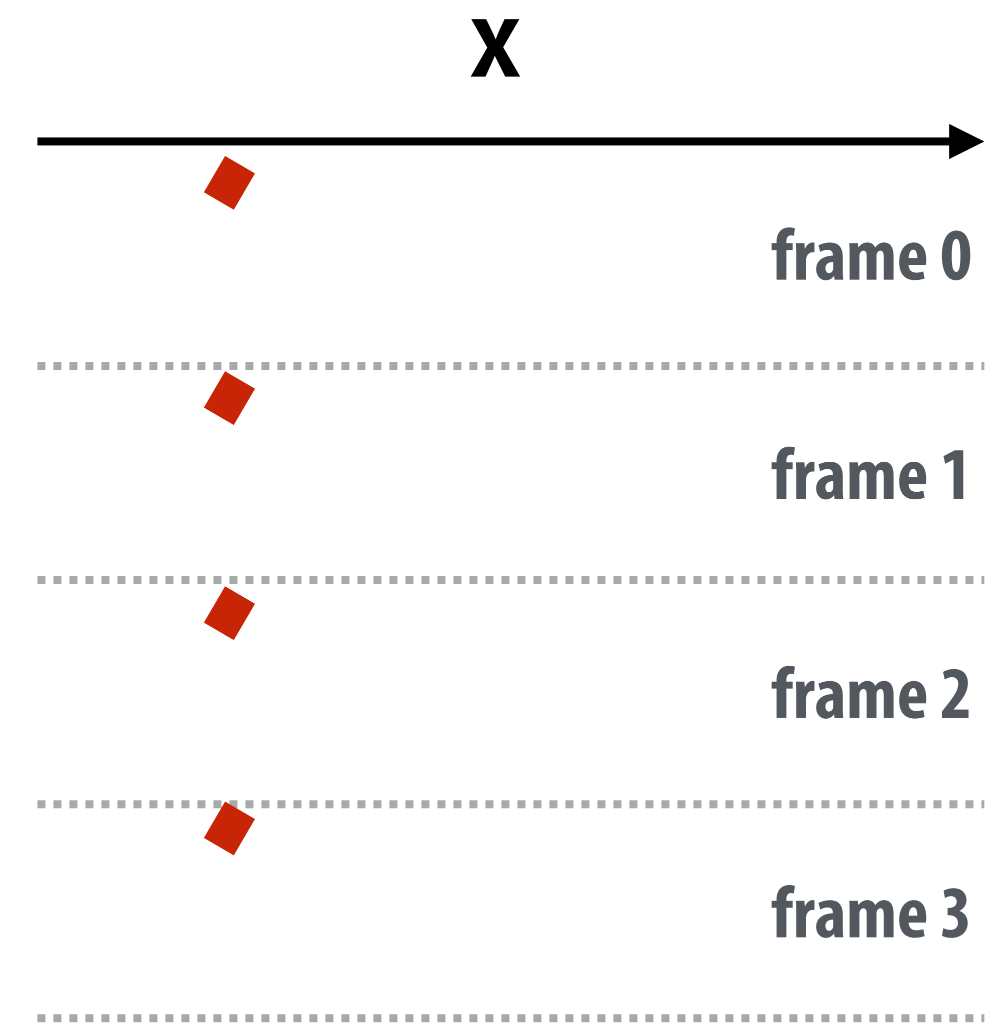
Reducing judder: low persistence display



Case 1: continuous ground truth



Light from full-persistence display



Light from low-persistence display

red object moving left-to-right and
eye moving to track object
OR
red object stationary but head moving
and eye moving to track object

Full-persistence display: pixels emit light for entire frame

Low-persistence display: pixels emit light for small fraction of frame

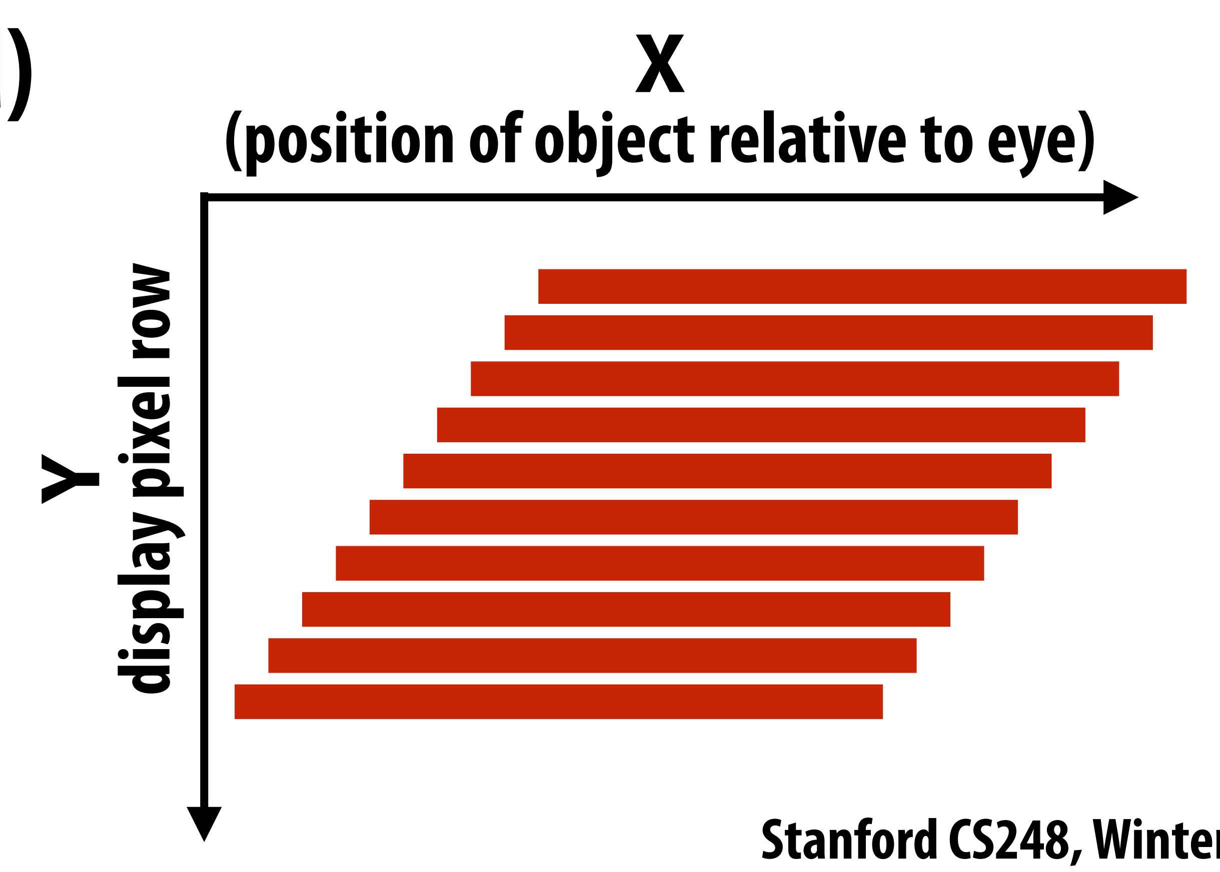
Oculus Rift CV1 low-persistence display

- 90 Hz frame rate (~11 ms per frame)
- Pixel persistence = 2-3ms

Spacetime diagrams adopted from presentations by Michael Abrash

Artifacts due to rolling OLED backlight

- Image rendered based on scene state at time t_0
- Image sent to display, ready for output at time $t_0 + \Delta t$
- "Rolling backlight" OLED display lights up rows of pixels in sequence
  - Let *r* be amount of time to "scan out" a row
  - Row 0 photons hit eye at $t_0 + \Delta t$
  - Row 1 photos hit eye at $t_0 + \Delta t + r$
  - Row 2 photos hit eye at $t_0 + \Delta t + 2r$
- Implication: photons emitted from bottom rows of display are "more stale" than photos from the top!
- Consider <u>eye moving horizontally relative to display</u> (e.g., due to head movement while tracking square object that is stationary in world)



Result: perceived shear!

Similar to rolling shutter effects on modern digital cameras.

Compensating for rolling backlight

- Perform post-process shear on rendered image
  - Similar to previously discussed barrel distortion and chromatic warps
  - Predict head motion, assume fixation on static object in scene
    - Only compensates for shear due to head motion, not object motion

- Render each row of image at a different time (the predicted time photons will hit eye)
  - Suggests exploration of different rendering algorithms that are more amenable to fine-grained temporal sampling, e.g., ray caster? (each row of camera rays samples scene at a different time)

Increasing frame rate using re-projection

- **Goal: maintain as high a frame rate as possible under challenging rendering conditions:**
  - Stereo rendering: both left and right eye views
  - High-resolution outputs
  - Must render extra pixels due to barrel distortion warp
  - Many "rendering hacks" (bump mapping, billboards, etc.) are less effective in VR so rendering must use more expensive techniques

- **Researchers experimenting with reprojection-based approaches to improve frame rate (e.g., Oculus' "Time Warp")**
  - Render using conventional techniques at lower frame rate (e.g., 30 fps), then reproject (warp) image to synthesize new frames based on predicted head movement at high frame rate (75-90 fps)
  - Potential for image processing hardware on future VR headsets to perform high frame-rate reprojection based on gyro/accelerometer

Near-future VR system components

Low-latency image processing
for subject tracking

High-resolution, high-frame rate,
wide-field of view display

Massive parallel computation for
high-resolution rendering

Exceptionally high bandwidth connection
between renderer and display:
e.g., 4K x 4K per eye at 90 fps!

In headset motion/accel
sensors + eye tracker

On headset graphics
processor for sensor
processing and re-
projection