Lecture 18:

Rendering for Virtual Reality

Visual Computing Systems
Stanford CS348K, Spring 2021

VR headsets

Oculus Quest 2

HTC Vive

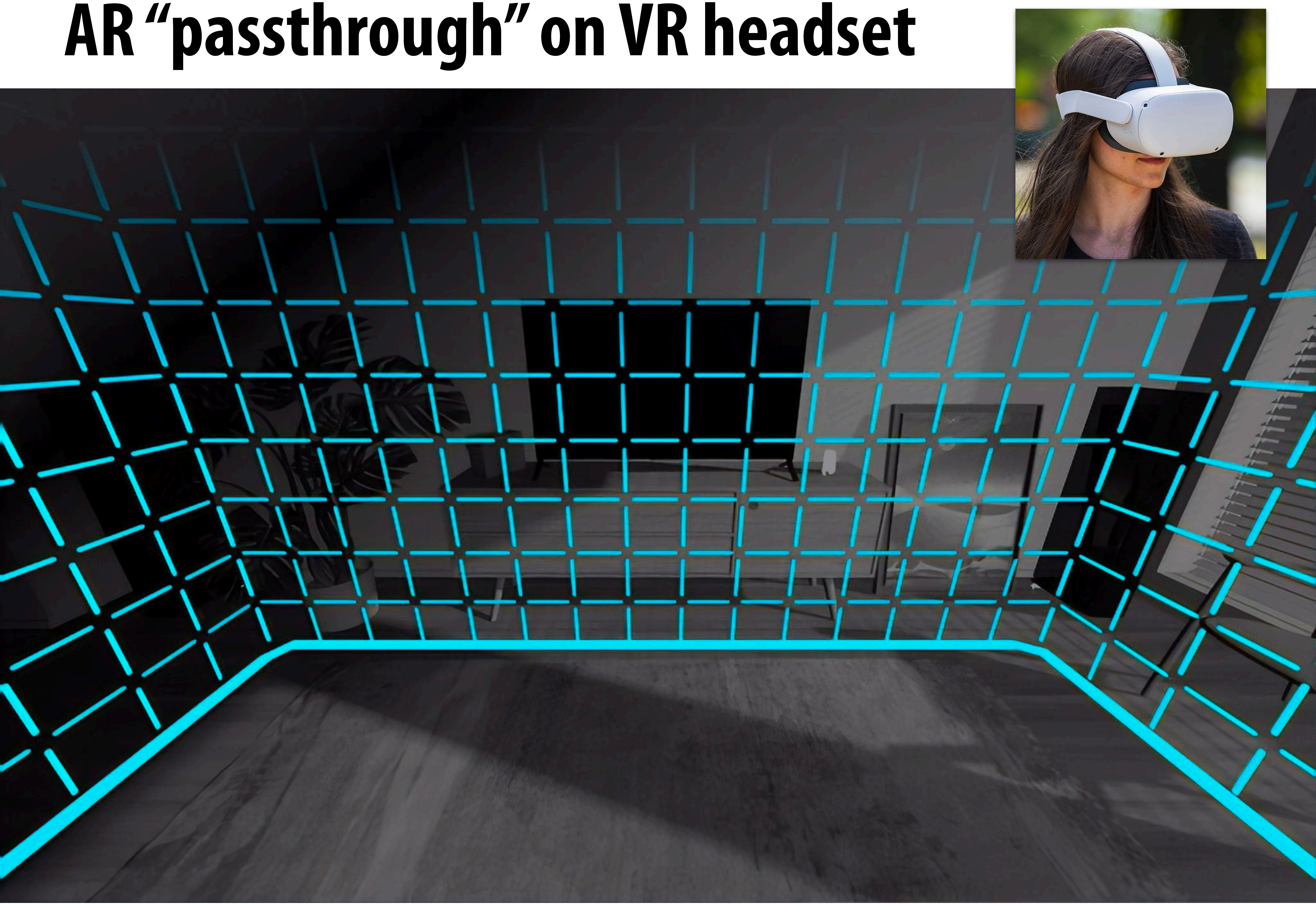
Sony Morpheus

Valve Index

Google Daydream

Google Cardboard

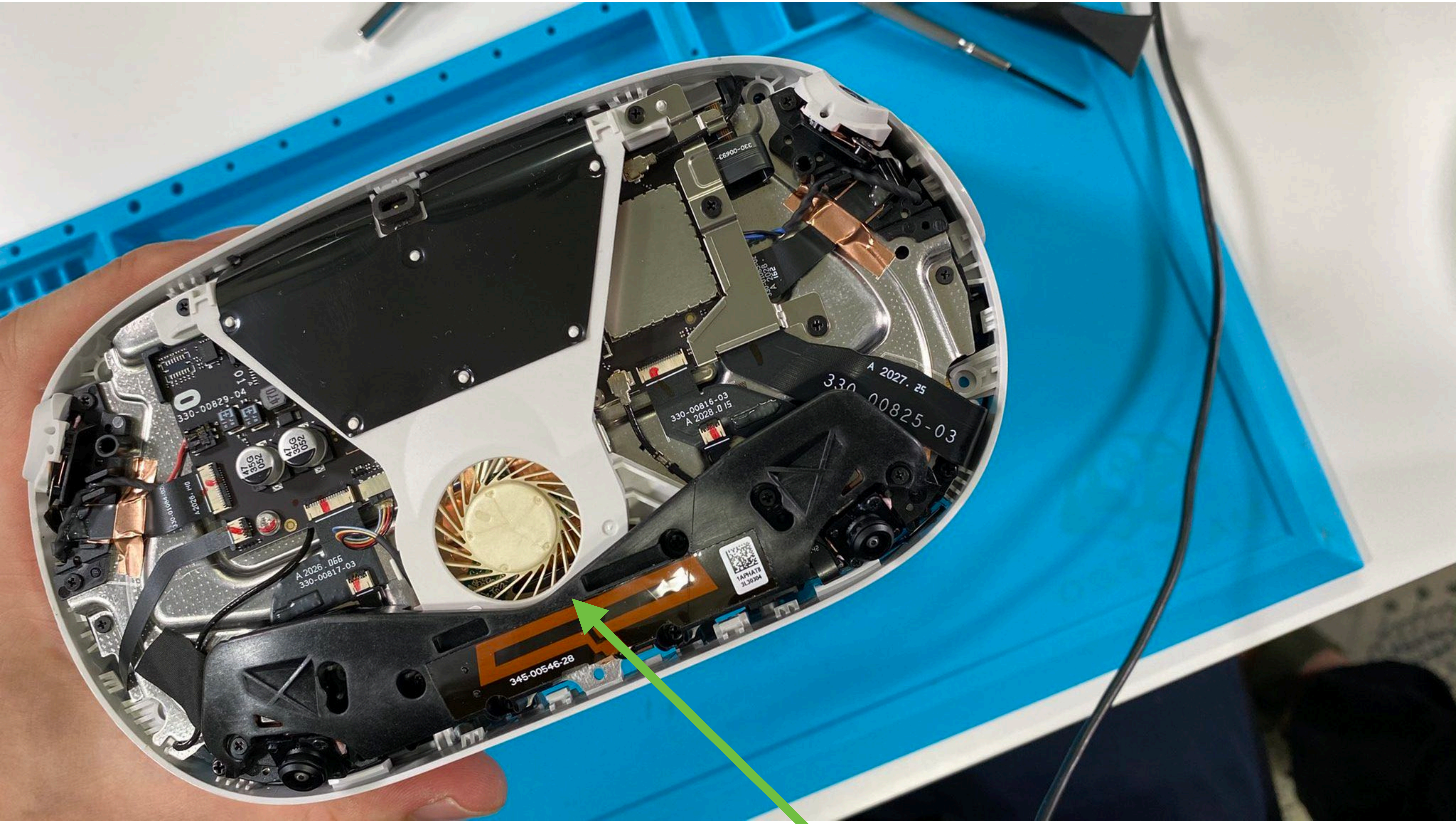
AR “passthrough” on VR headset



Oculus Quest 2 headset (2020)

Oculus Quest 2 headset (lens side view)

Oculus Quest 2 headset



Fan

Image credit: ifixit.com

Oculus Quest 2 headset

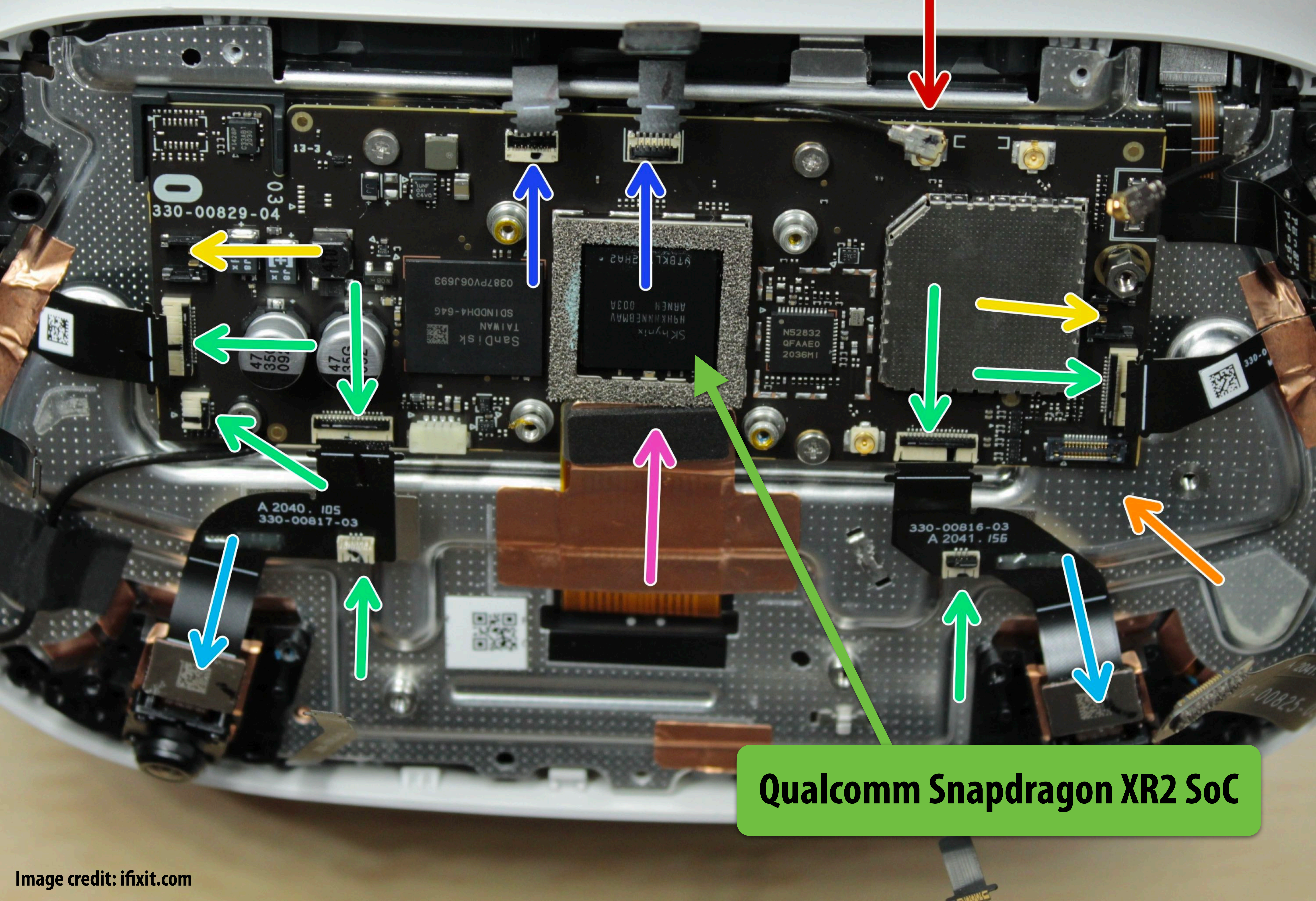
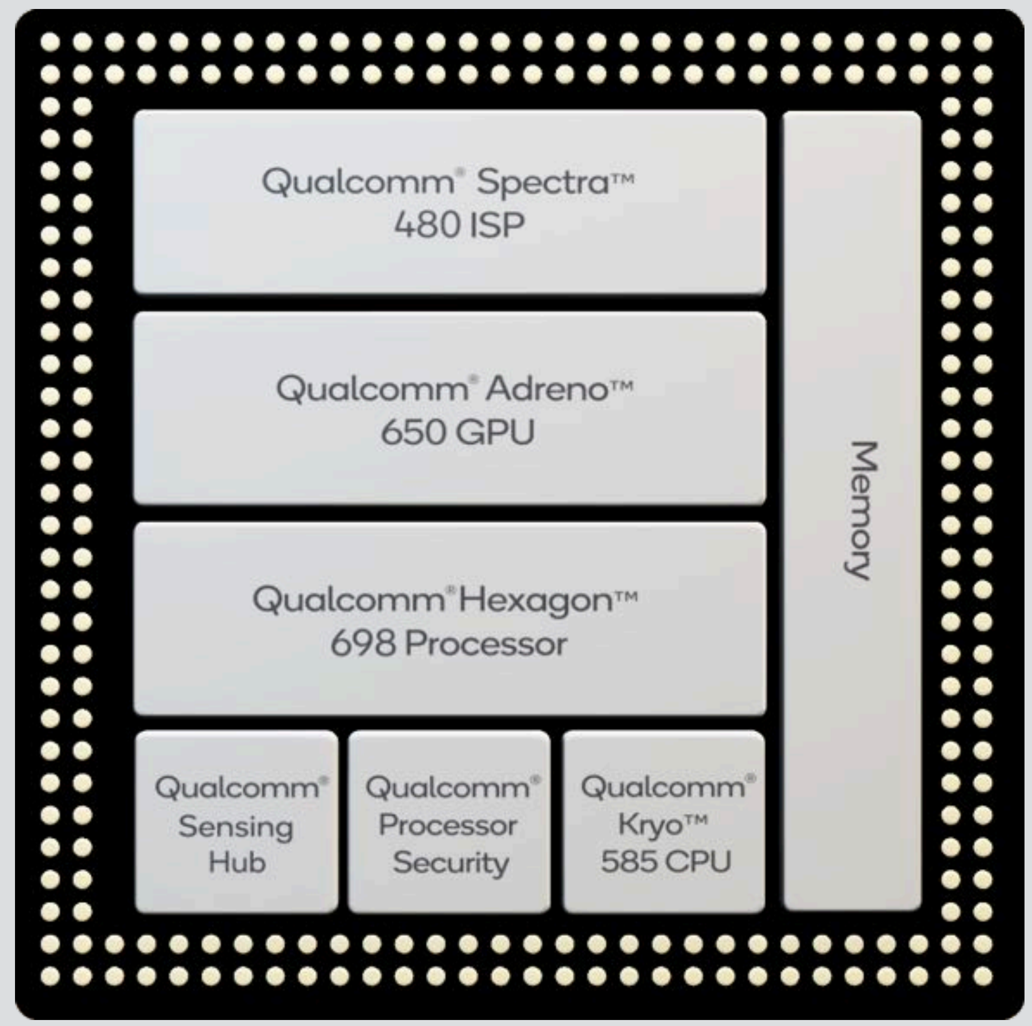


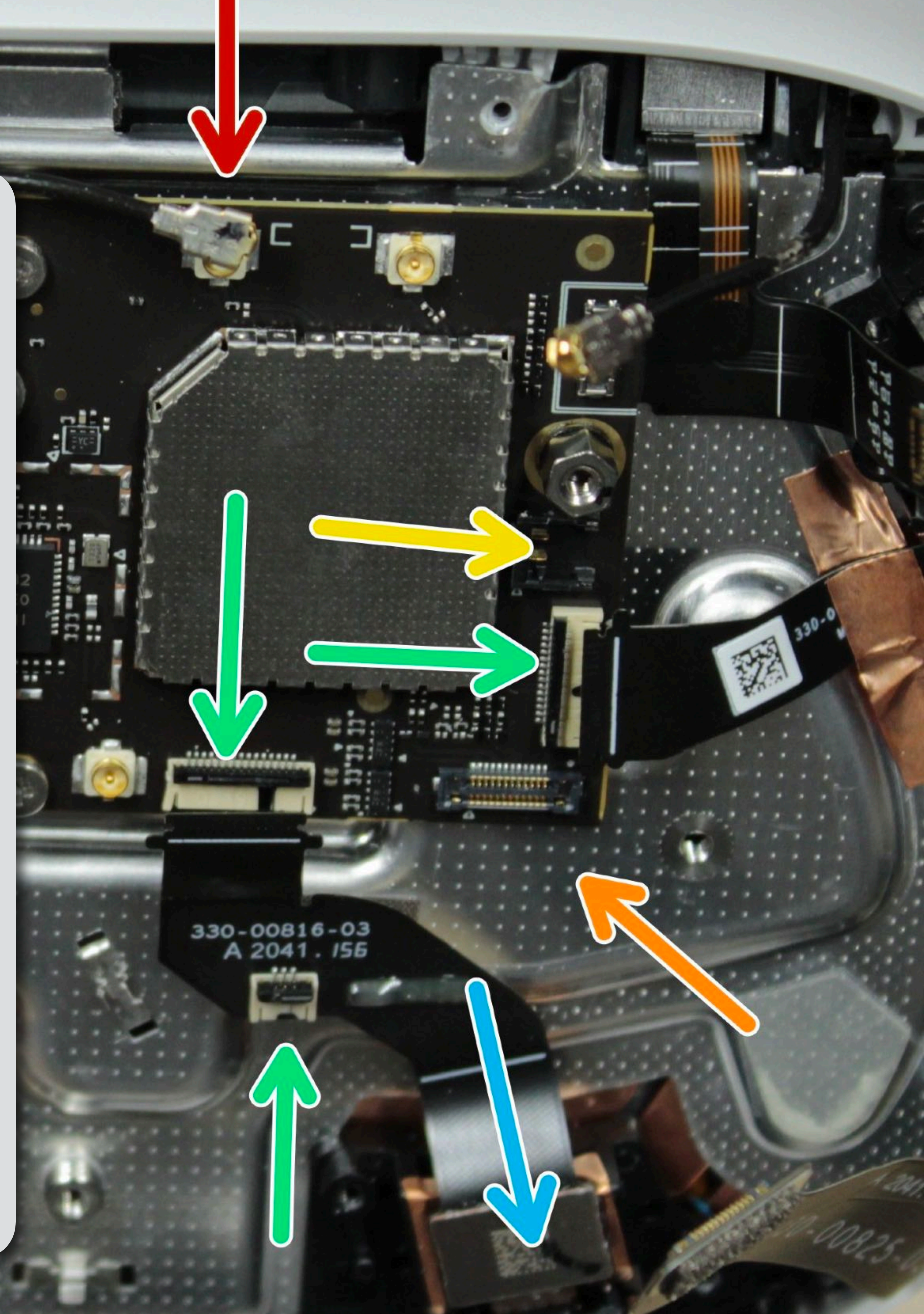
Image credit: ifixit.com

Oculus Quest 2 headset (Snapdragon SoC)



This diagram is from Snapdragon 865

4 high-performance cores
4 low-performance (low energy) cores
Image processor + DSP
Multi-core graphics processor (GPU) — up to 3000 x 3000 display @ 90 Hz
Additional processor for sensors (IMU etc)
Can process inputs from up to seven simultaneous video camera streams



Qualcomm Snapdragon XR2 SoC

Image credit: ifixit.com

Oculus Quest 2 headset

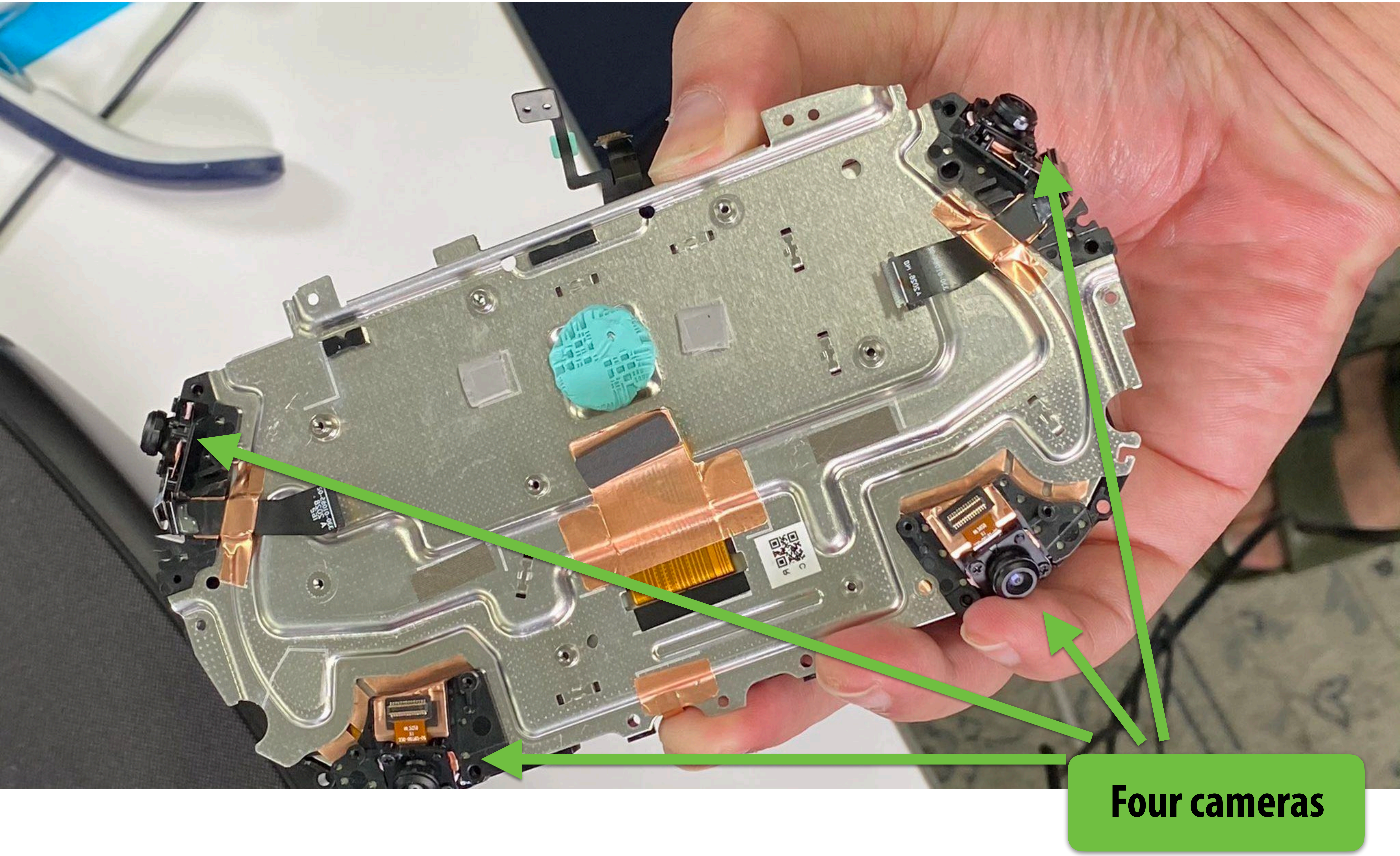


Image credit: ifixit.com

Oculus Quest 2 headset (lens assembly)

Image credit: ifixit.com

Oculus Quest 2 display + lens assembly

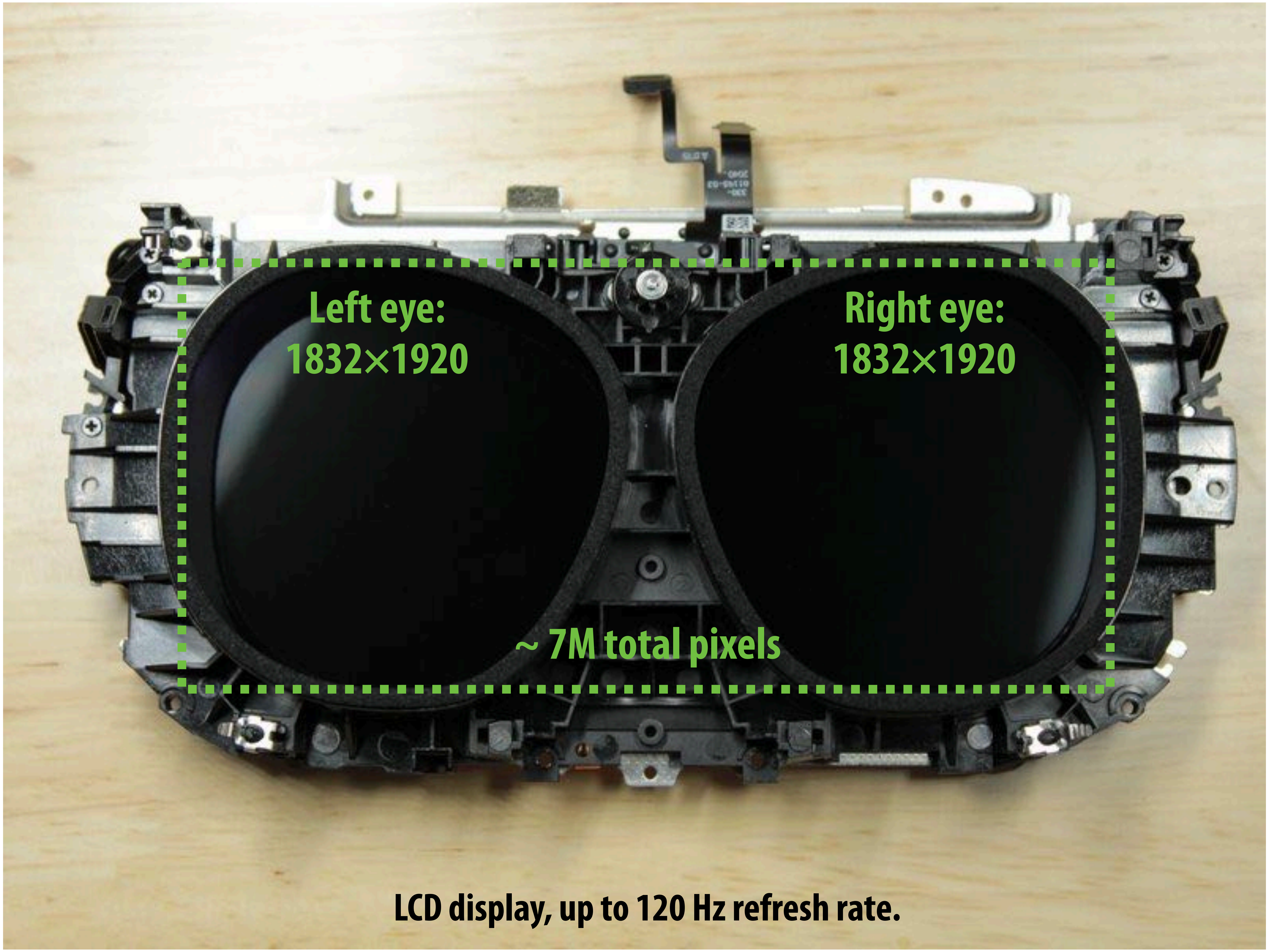
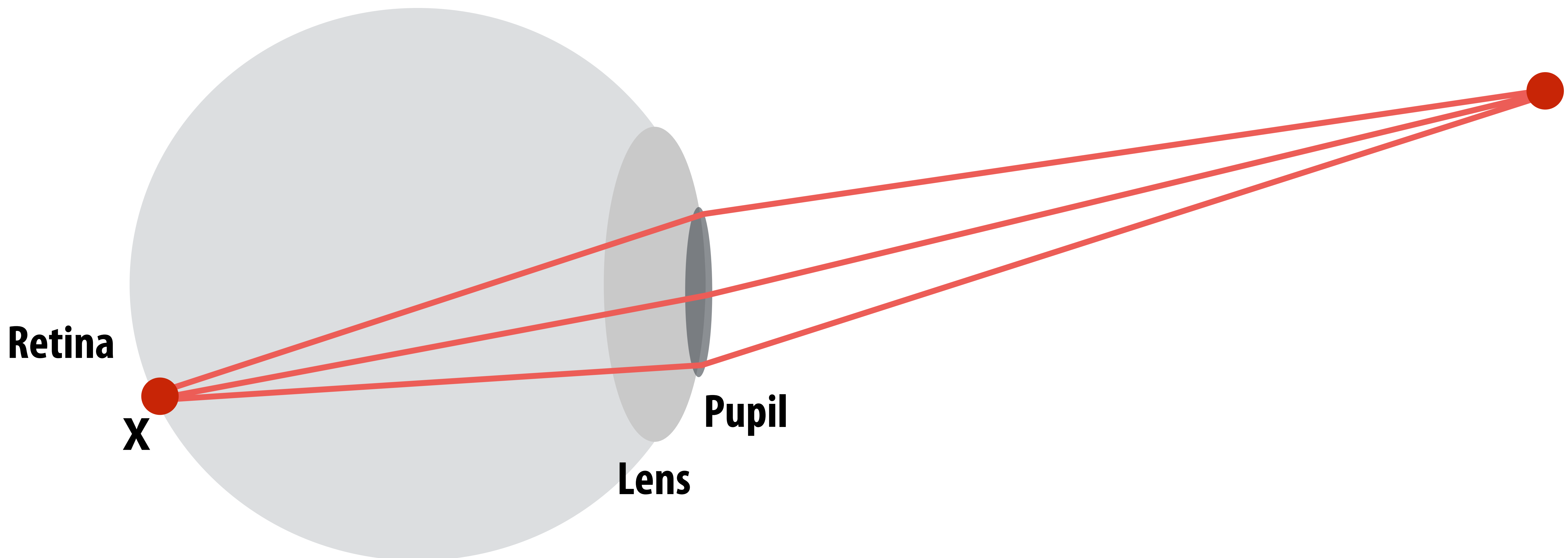


Image credit: ifixit.com

Consider projection of scene object on retina

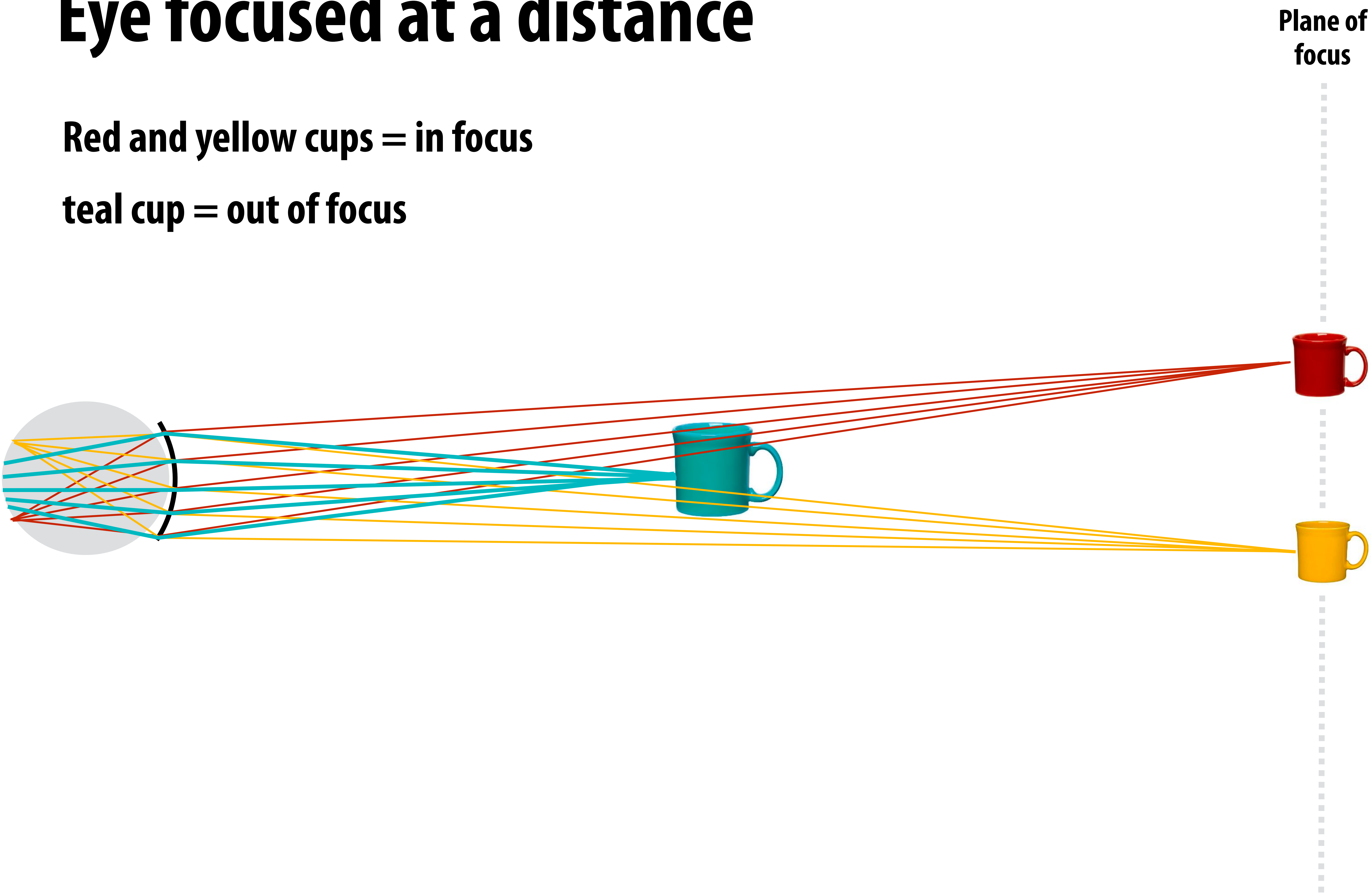


Here: object projects onto point X on back of eye (retina)

Eye focused at a distance

Red and yellow cups = in focus

teal cup = out of focus

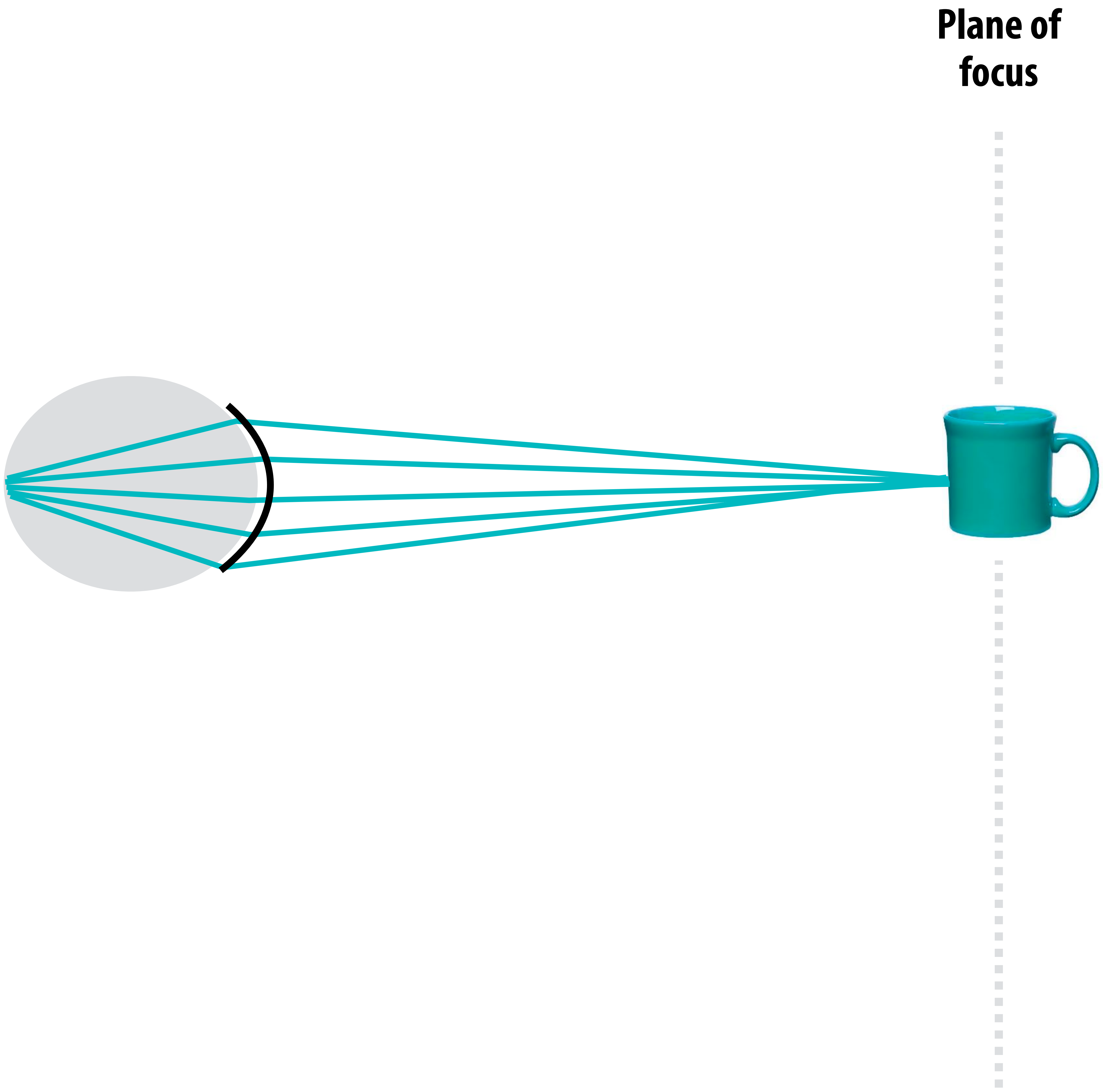


Eye focused at a distance

teal cup = out of focus

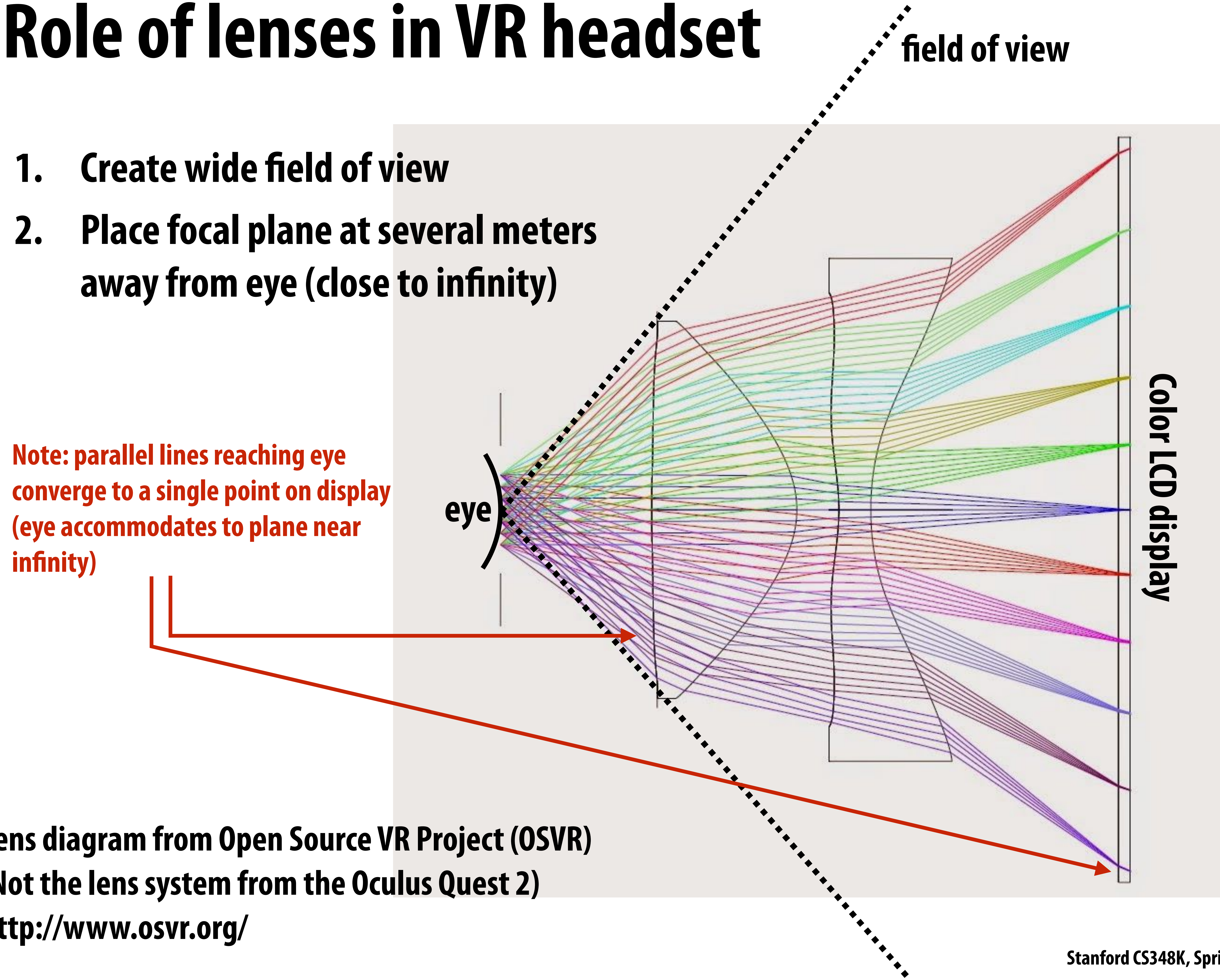
Eye focused up close

teal cup = in focus



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)



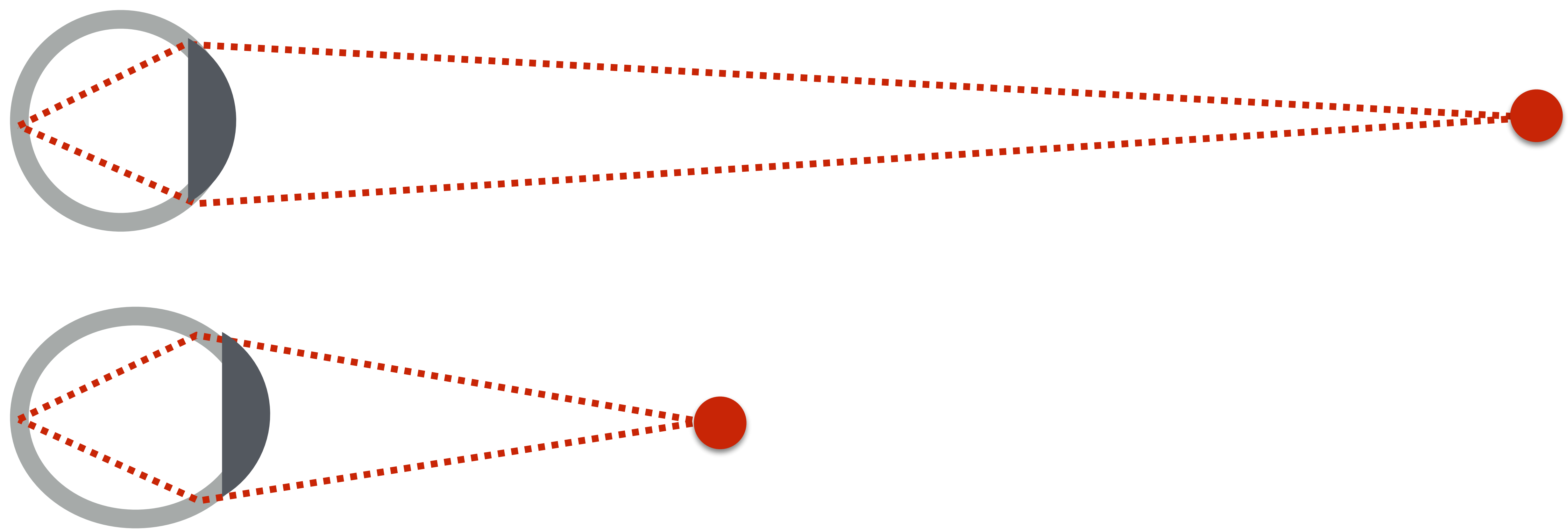
Lens diagram from Open Source VR Project (OSVR)
(Not the lens system from the Oculus Quest 2)
http://www.osvr.org/

Accommodation and vergence

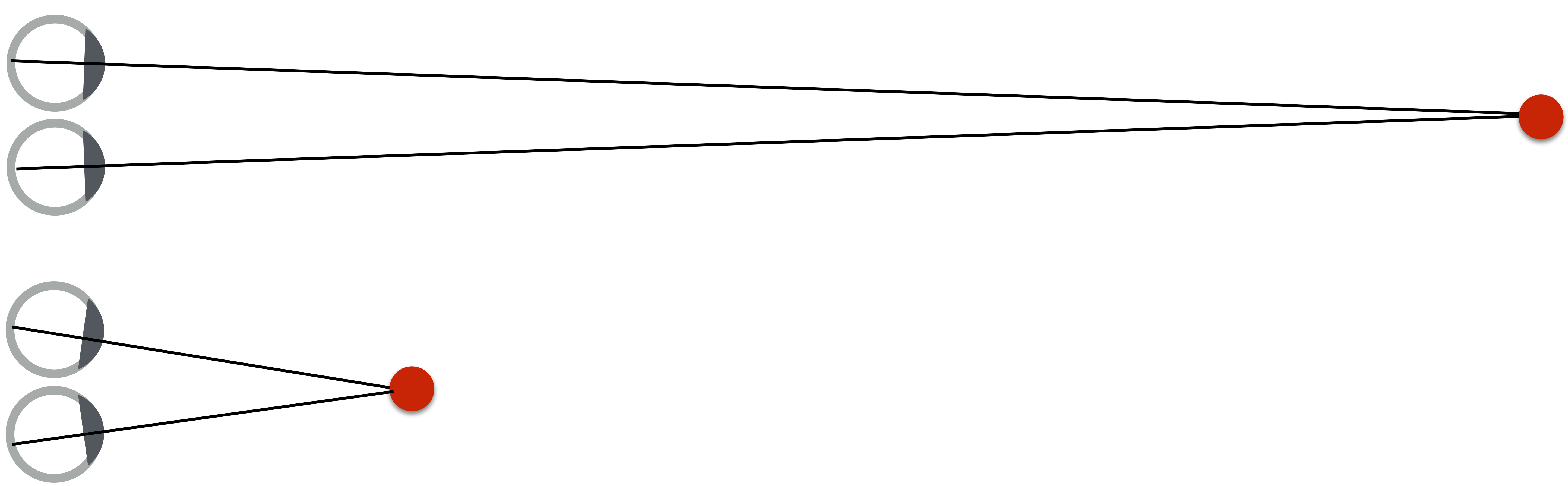
Accommodation: changing the optical power of the eye to focus at different distances

Eye accommodated at far distance

Eye accommodated at near distance

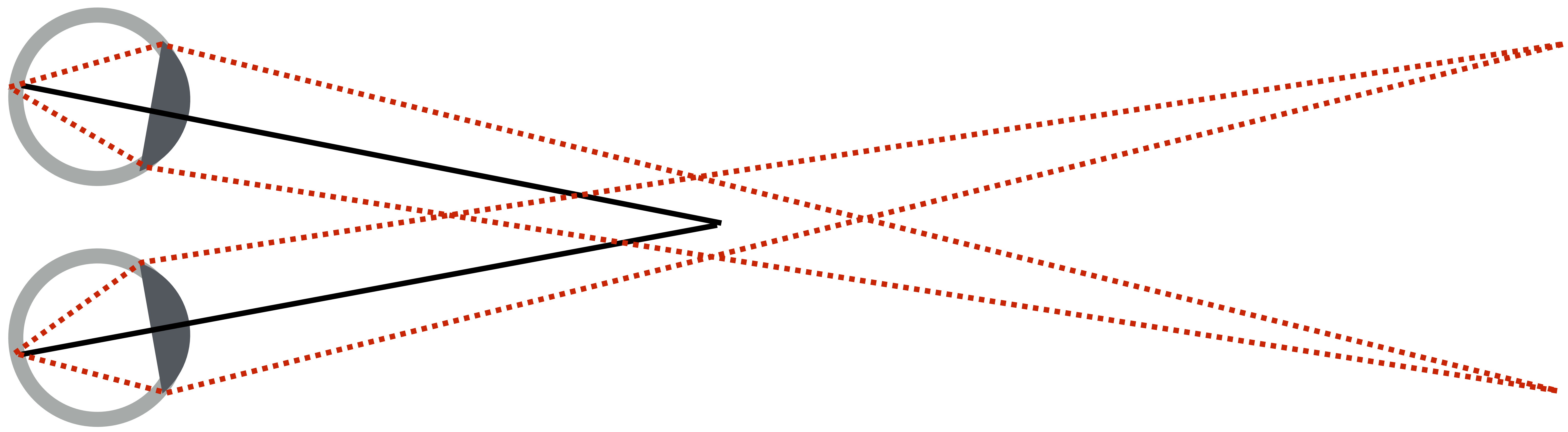


Vergence: rotation of eye to ensure projection of object falls in center of retina



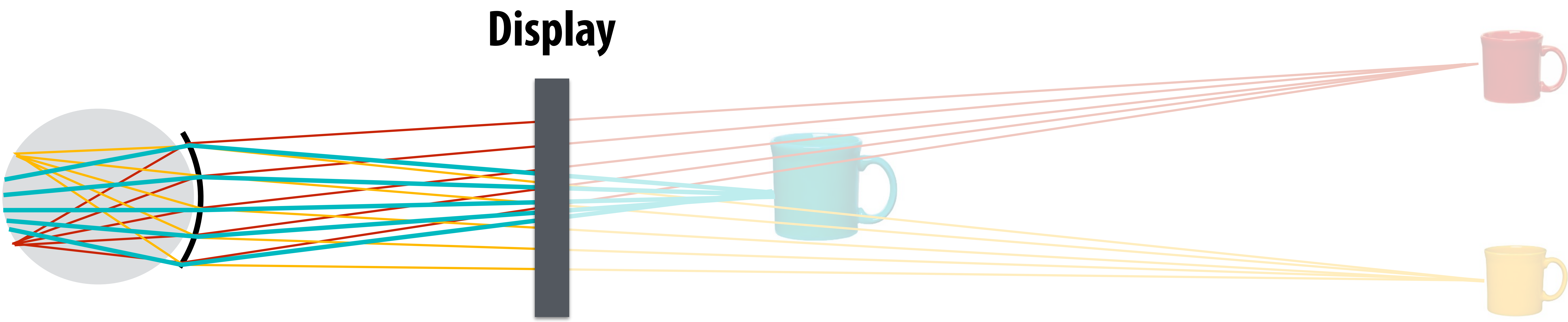
Accommodation/vergence conflict

- 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…

A better (future) display



Note how this hypothetical display creates the same rays of light as what would be seen in the real environment.

The *same position* on the display emits light with *different colors* in *different directions*. (Current LCD displays emit same color in all directions from each pixel)

The display generates the same “light field” in front of the eye as present in the real scene.

Need for high resolution

Recall: Oculus Quest 2 display

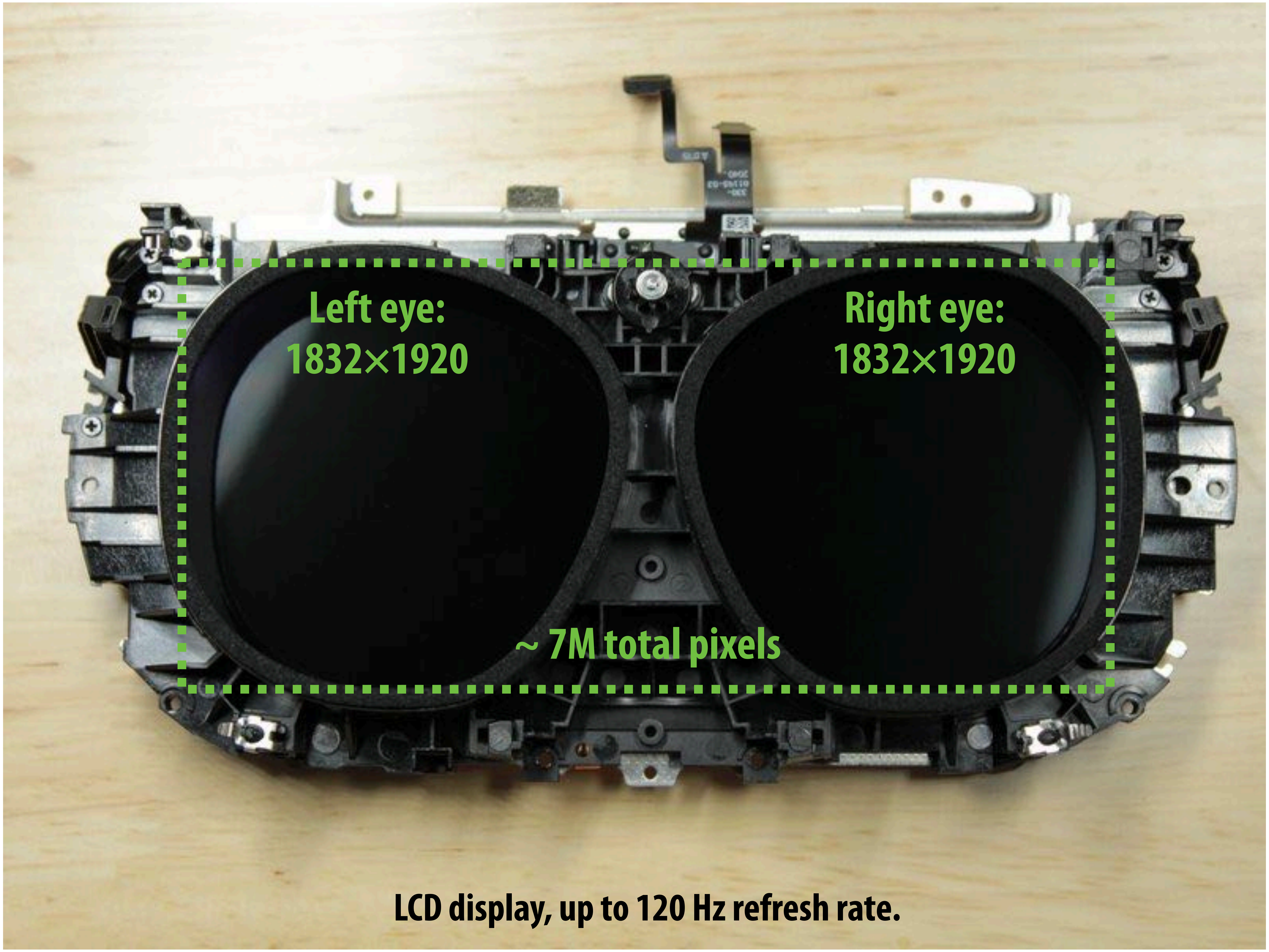
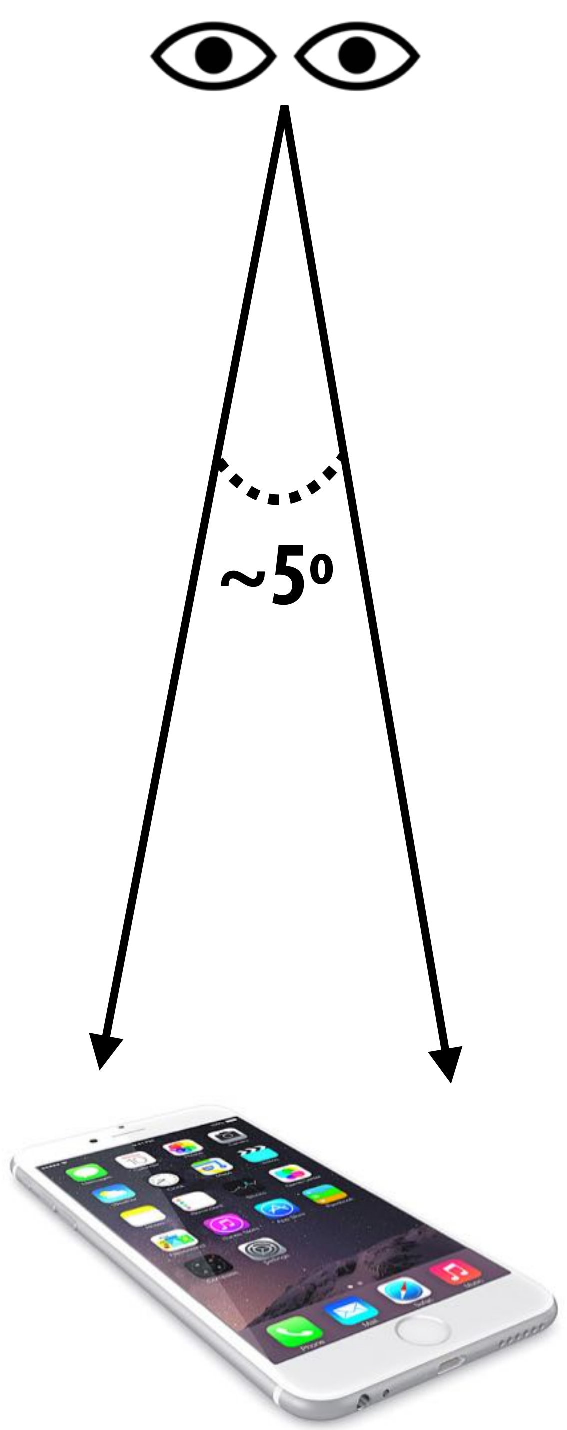


Image credit: ifixit.com

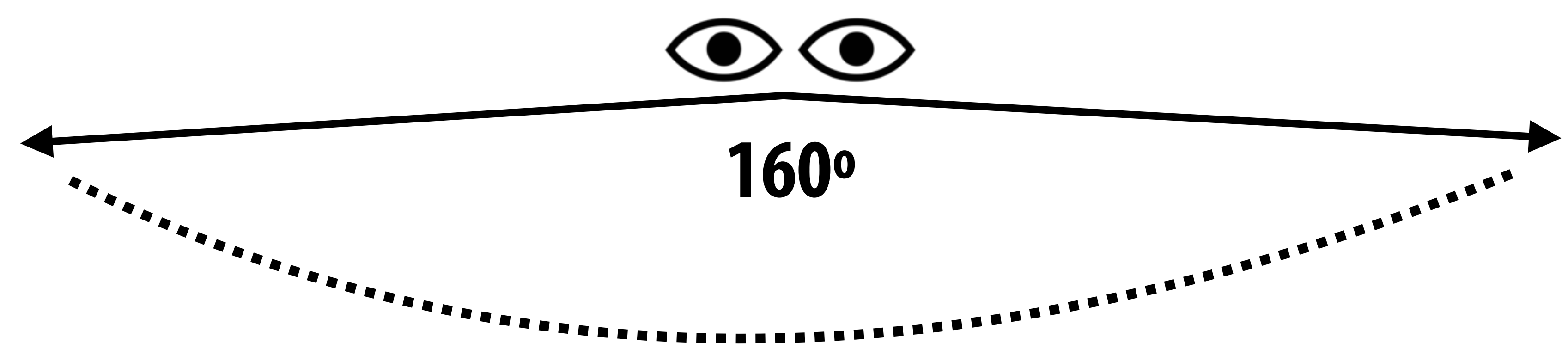
Need for high resolution



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:

~ 8K x 8K display per eye

= 128 MPixel

16K TVs!!!

15,360 x 8,640 resolution…

~ 132 Mpixel 🤯🤯🤯🤯

Forget 8K, Sony's New 63-Foot 16K Crystal LED TV Is Now Available—for a Few Million

The ballpark figure is $5 million.

By RACHEL CORMACK

Courtesy of Sony

When a new gogglebox drops, it's always the same drill: The screen gets bigger, the resolution gets better and the design gets bolder. Indeed, it's difficult for a brand to stand out. Unless you're Sony and the new TV your peddling is the size of a New York City public bus and also happens to boasts an unheard-of 16K screen.

Earlier this year when Sony unveiled the colossal 63-foot TV—the biggest 16K screen of its kind—it had commercial cinemas in its sights. But, hey, why should theaters have all the fun?

Consider bandwidth cost of getting pixels to display

- 132 Mpixel @ 120 Hz x 24 bpp = 354 Gbits/s
- Note: modern display compression technologies (such as Display Stream Compression — DSC 1.2a) provide ~ 3:1 compression
  - Reduces need to 118 Gbits/s bandwidth

- Now consider *energy cost* of *transmitting pixels* to display at this rate
  - Rough estimate: ~ 100 pico-Joules per bit transferred *
  - 100 Pj/bit x 118 Gbit/s = 11.8 J/s = 11.8 W
  - Snapdragon SoC in Oculus Quest 2 designed for TDP of ~ 5W

* Signaling technologies undergo rapid improvement, feasible to see 1pJ/bit in the next decade

Display stream compression (DSC)

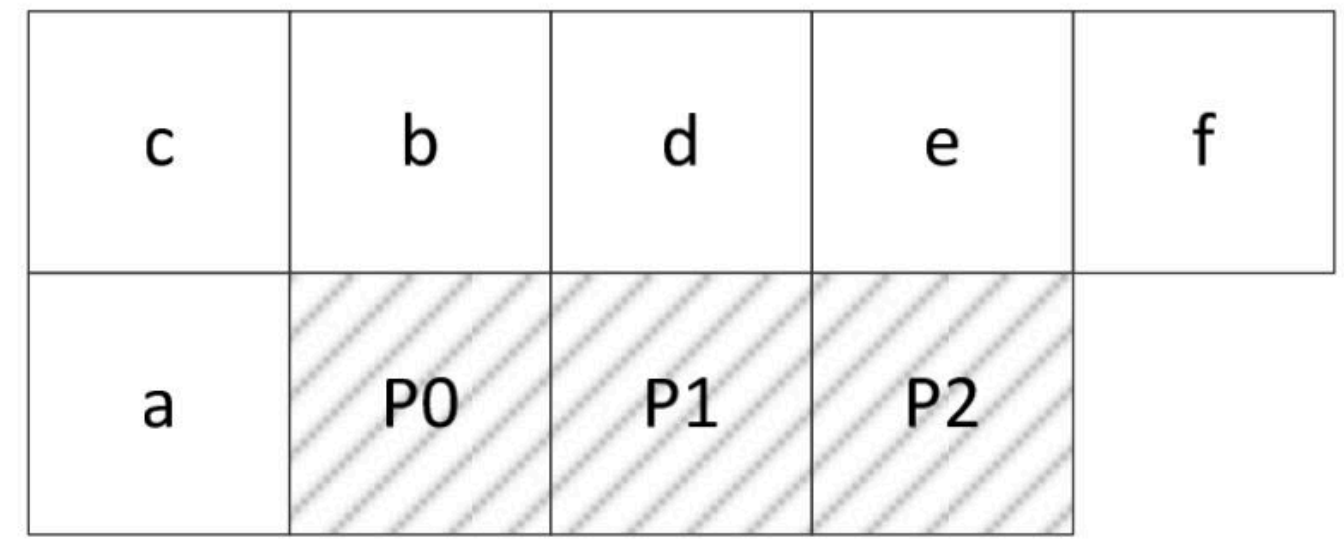
- **Goal: high compression ratio but, but cheap to encode so compression can be performed at high data rate.**
- **Example modes:**
  - **MMAP (Modified median-adaptive prediction)**

$$P0 = CLAMP(a + \tilde{b} - \tilde{c}, MIN(a, \tilde{b}), MAX(a, \tilde{b}))$$

$$P1 = CLAMP(a + \tilde{d} - \tilde{c} + R0, MIN(a, \tilde{b}, \tilde{d}), MAX(a, \tilde{b}, \tilde{d}))$$

$$P2 = CLAMP(a + \tilde{e} - \tilde{c} + R0 + R1, MIN(a, \tilde{b}, \tilde{d}, \tilde{e}), MAX(a, \tilde{b}, \tilde{d}, \tilde{e})) \quad (2)$$

Where $\tilde{b}$, $\tilde{c}$, $\tilde{d}$, and $\tilde{e}$ are the results of the blending operation for the pixels in the previous line, R0 and R1 are the inverse quantized residuals of the first two pixels in the group,



  - **ICH (indexed history mode): Retain buffer of the last 32 pixels and encoding is index into that buffer.**

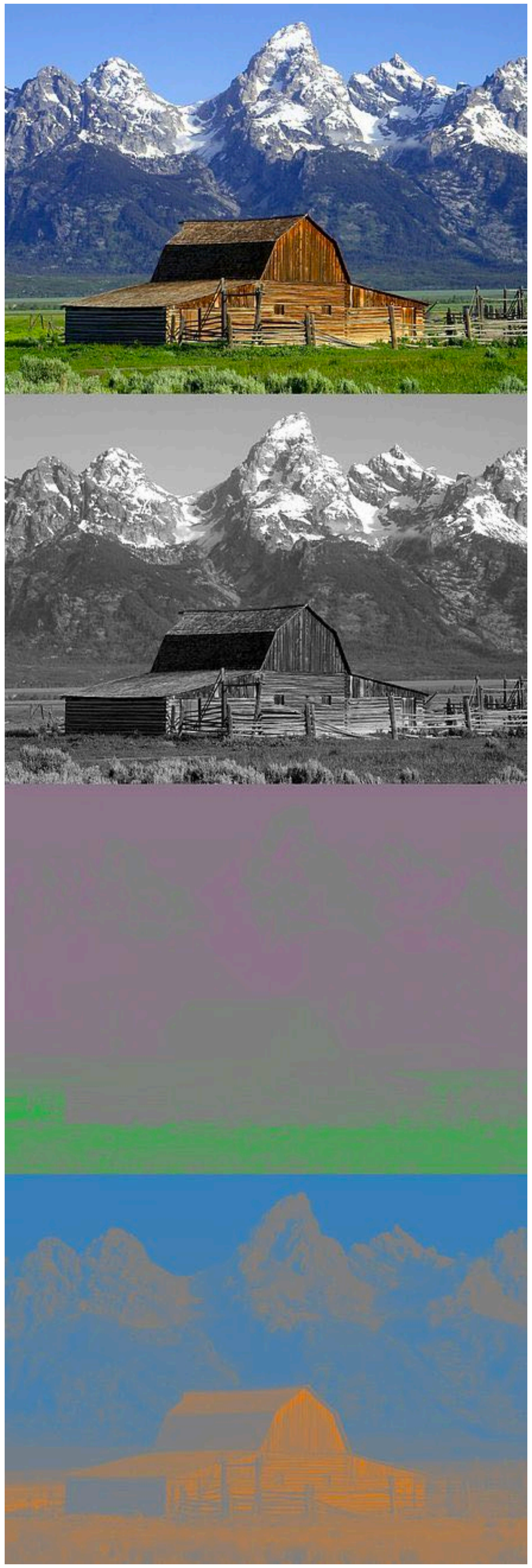
- **Encoding performed on luma/chroma representation**
  - **YCgCo-R (see next slide)**

YCgCo

- Luma, chrominance green, chrominance orange
- Conversation from RGB to luma/chroma representation with minimal hardware (only additions and shifts)
- YCgCo-R is a slight modification that supports lossless (bit precise) conversation from and back to RGB

$$\begin{bmatrix} Y \\ Co \\ Cg \end{bmatrix} = \begin{bmatrix} \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \\ \frac{1}{2} & 0 & -\frac{1}{2} \\ -\frac{1}{4} & \frac{1}{2} & -\frac{1}{4} \end{bmatrix} \cdot \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 & 1 & -1 \\ 1 & 0 & 1 \\ 1 & -1 & -1 \end{bmatrix} \cdot \begin{bmatrix} Y \\ Co \\ Cg \end{bmatrix}$$



RGB

Y

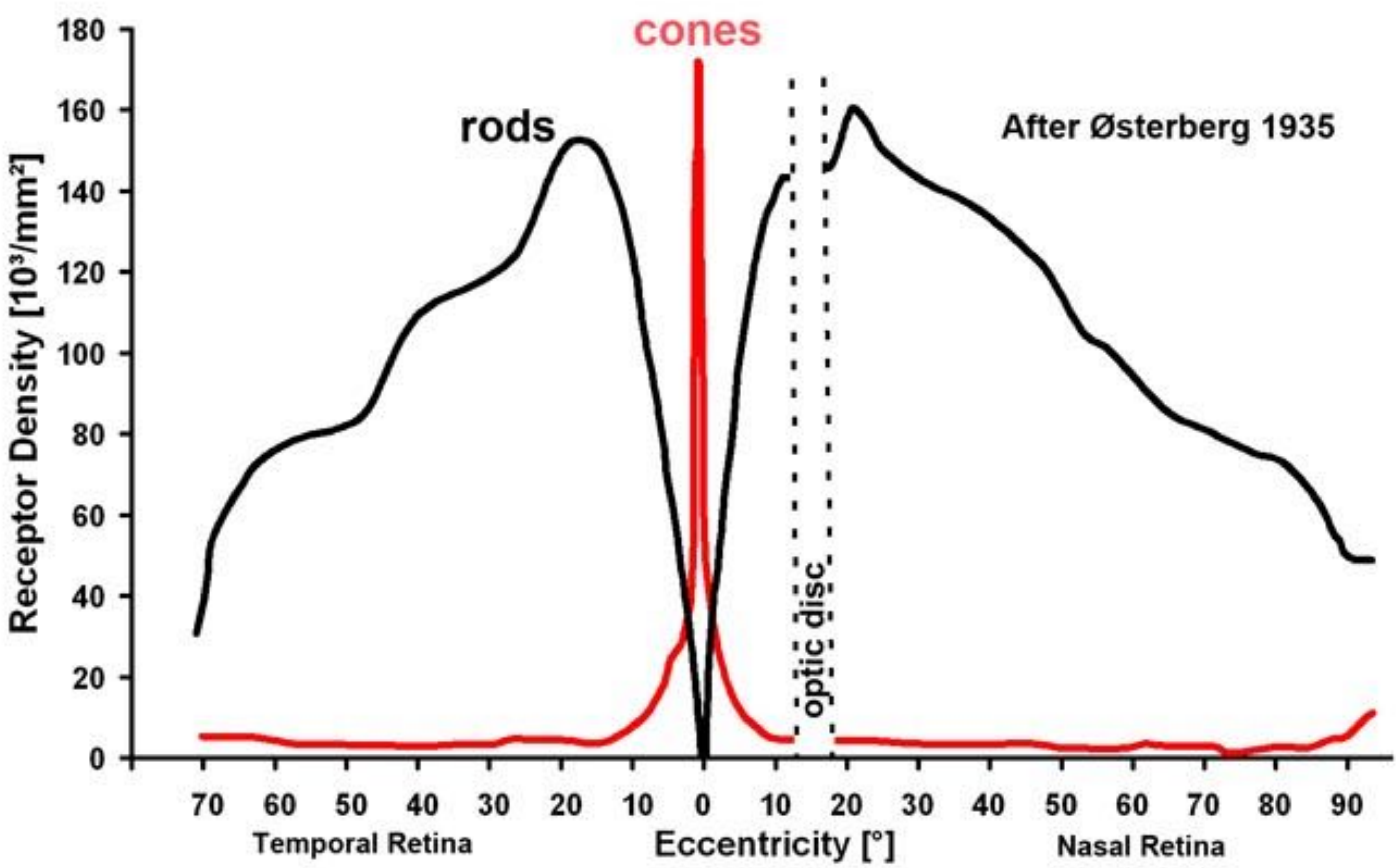
Cg

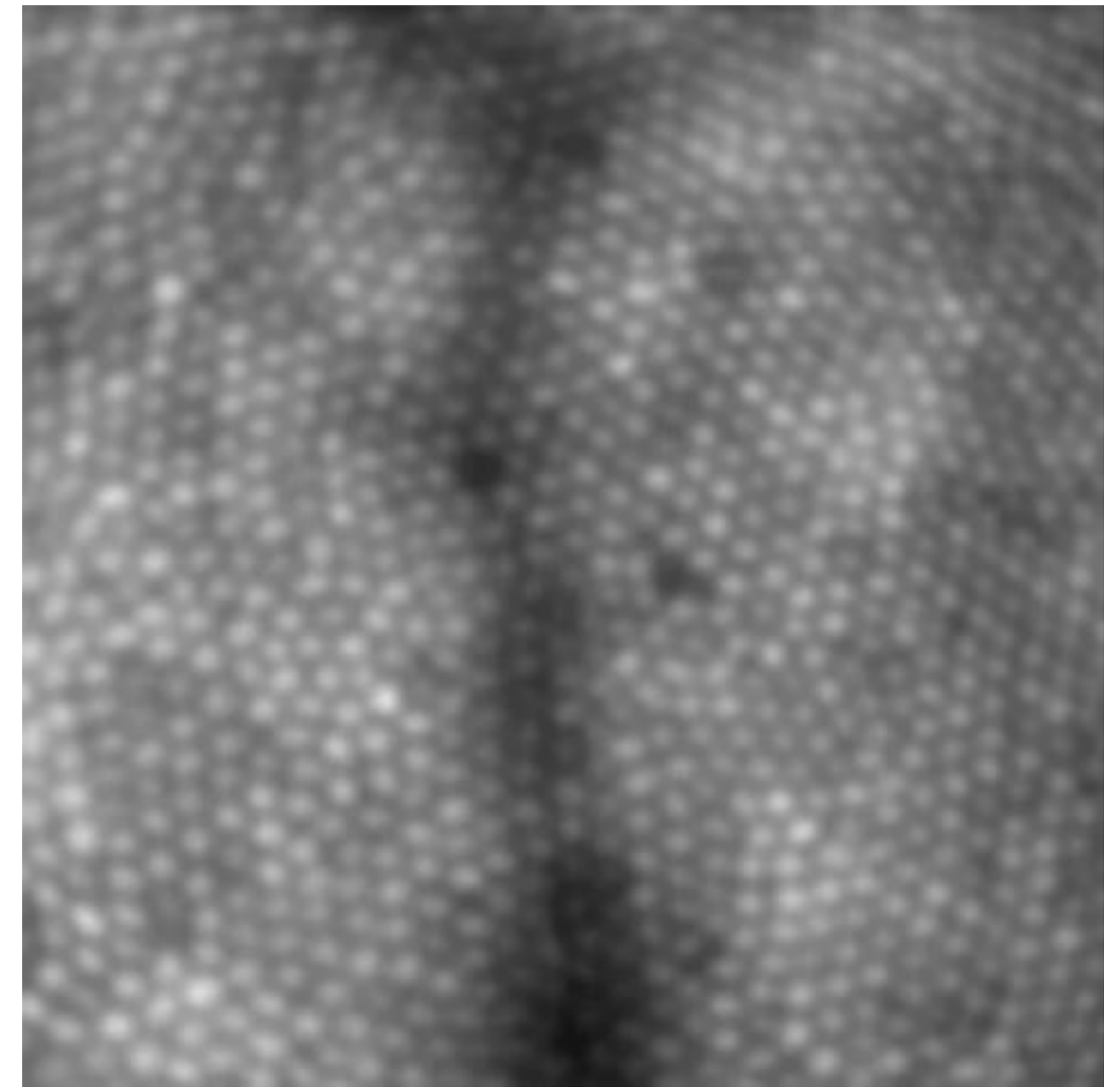
("chrominance green")

Co

("chrominance orange")

Density of rod and cone cells in the retina





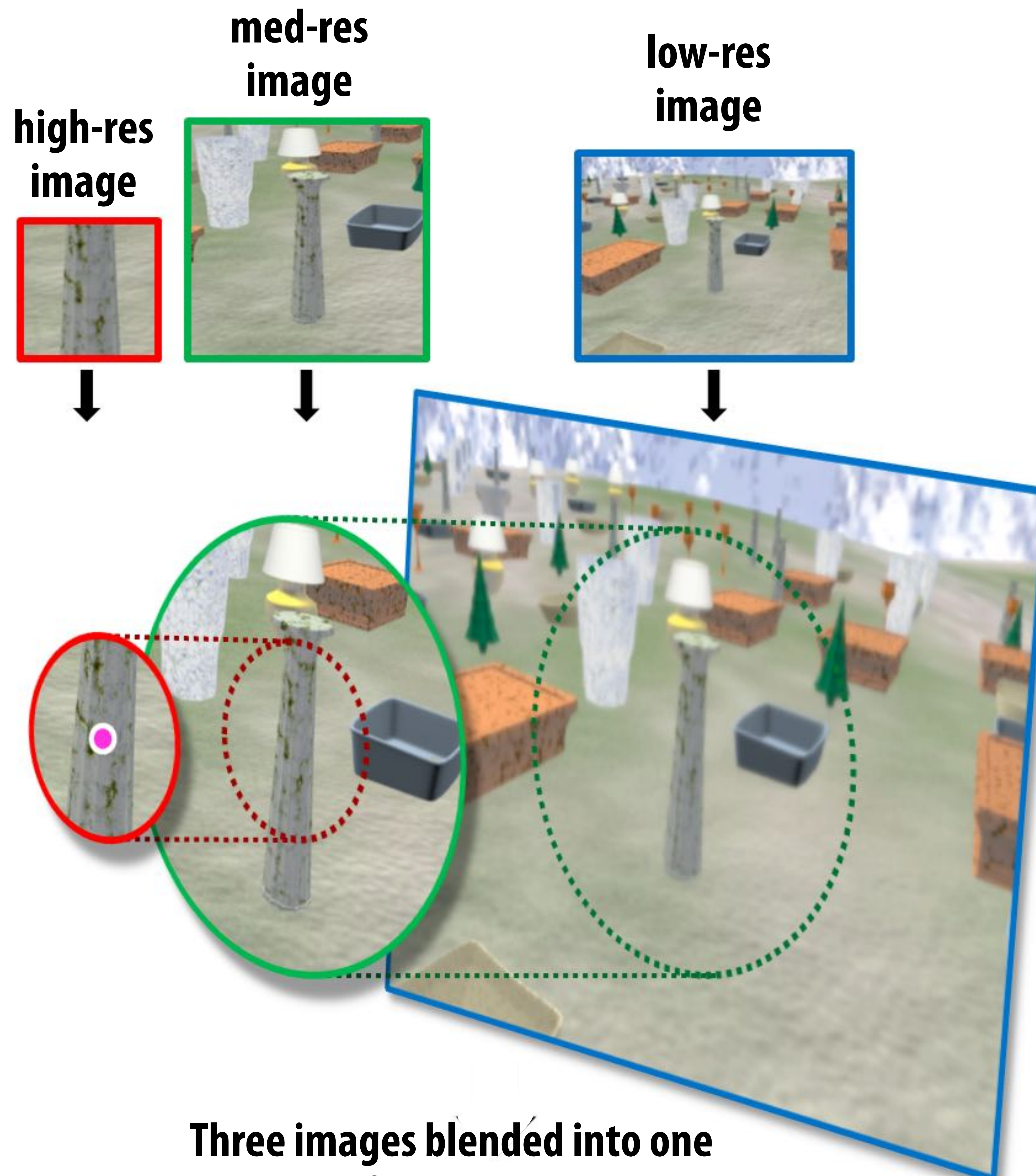
[Roorda 1999]

- **Cones are color receptive cells**
- **Highest density of cones is in fovea**
  (best color vision at center of where human is looking)
- **Implication: human eye has low spatial resolution away from fovea *(opportunity to reduce computation by computing less in these areas)***

Reducing rendering cost via foveated rendering

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

VR headset with eye tracker:
HTC Vive Pro Eye



Three images blended into one for display

Eye tracking based solutions

- Given gaze information, many rendering-cost reducing strategies
  - Use low resolution rendering away from point of gaze
  - More practical: perform part of the rendering computation at lower frequency (lower-rate shading, reduce texture LOD etc.) *

- Fundamental problem: accurate low-latency eye tracking is challenging
  - Abnormal eyes, etc.

* We'll come back to this in a second when we talk about lens matched shading

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

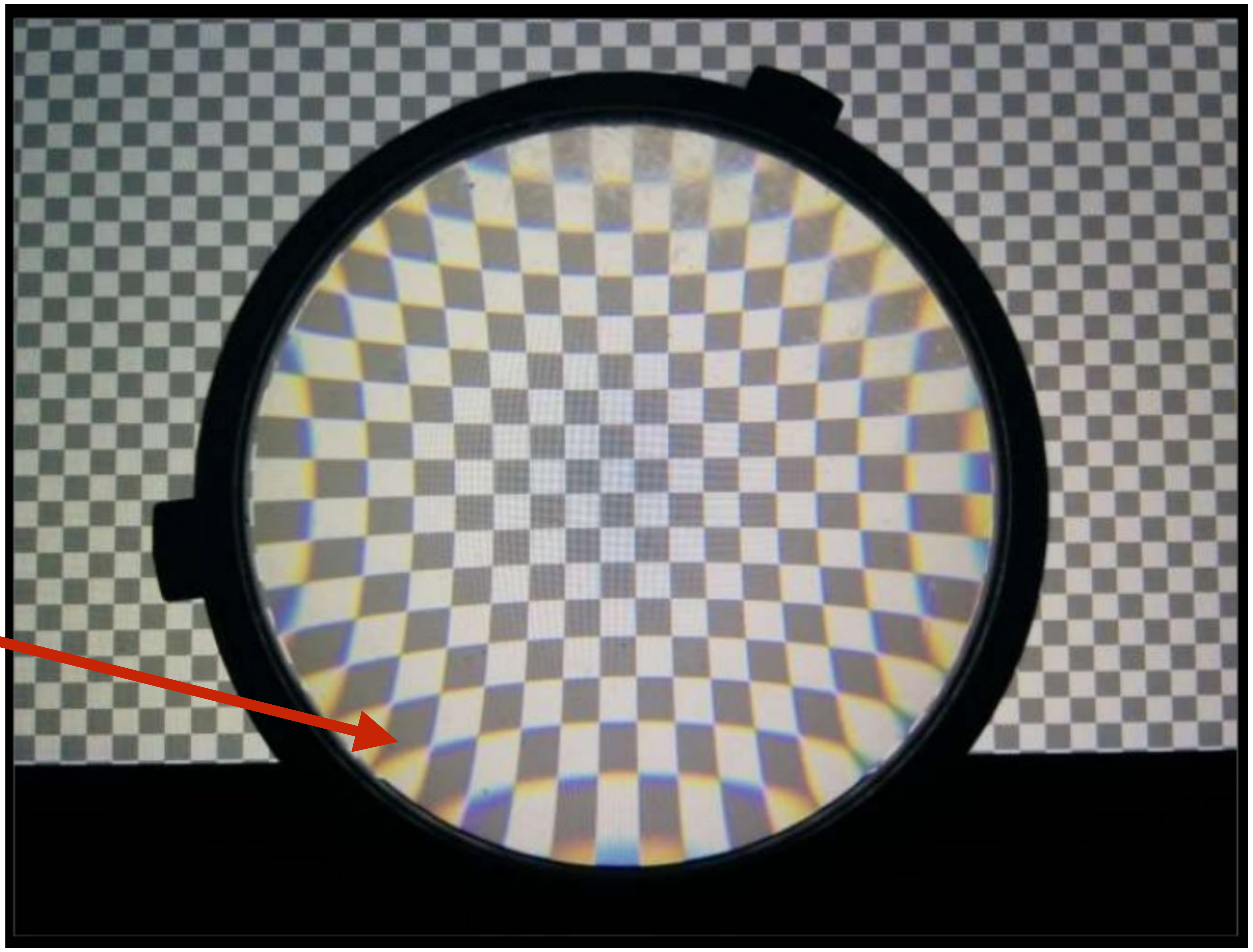
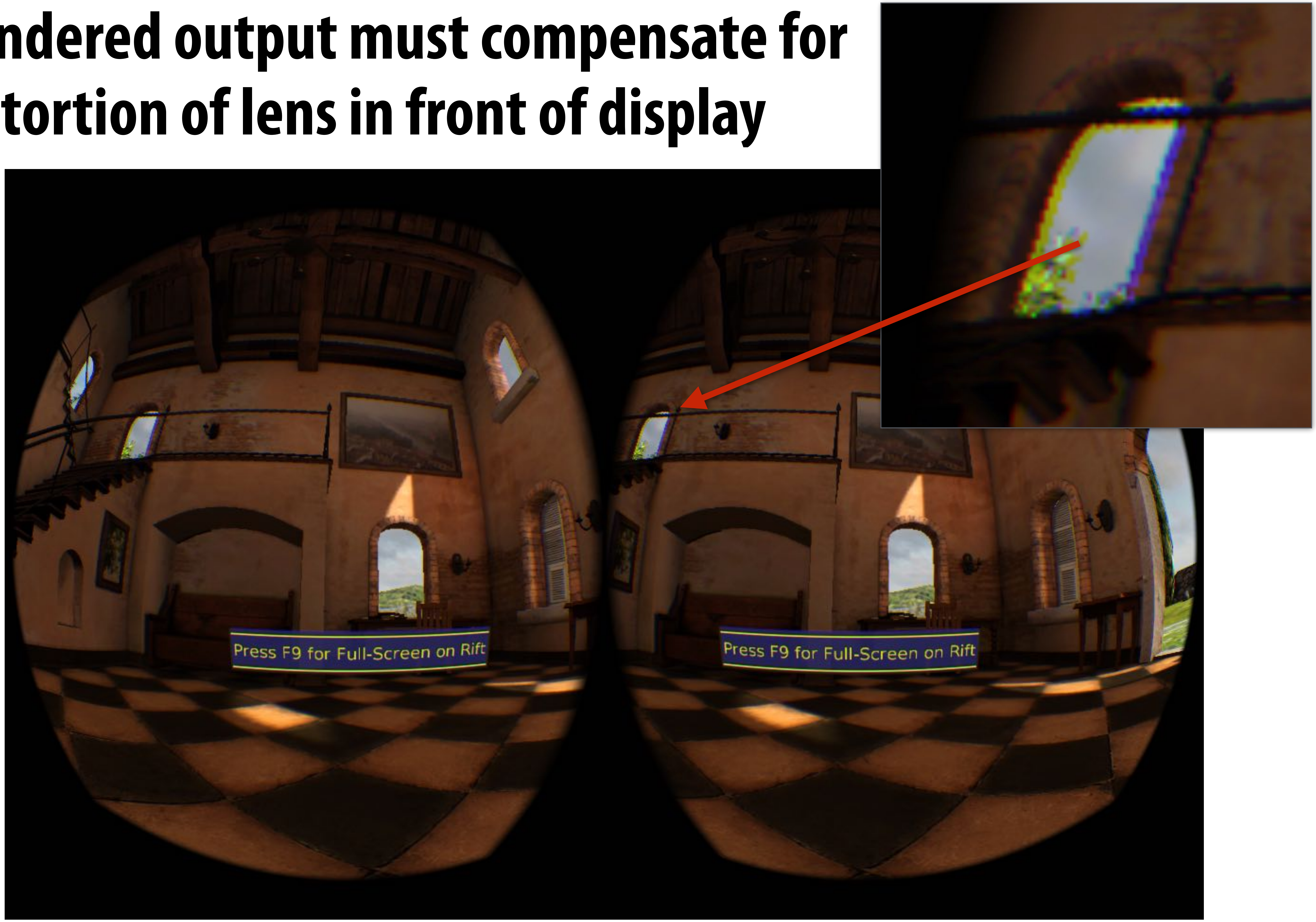


Image credit: Cass Everitt

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



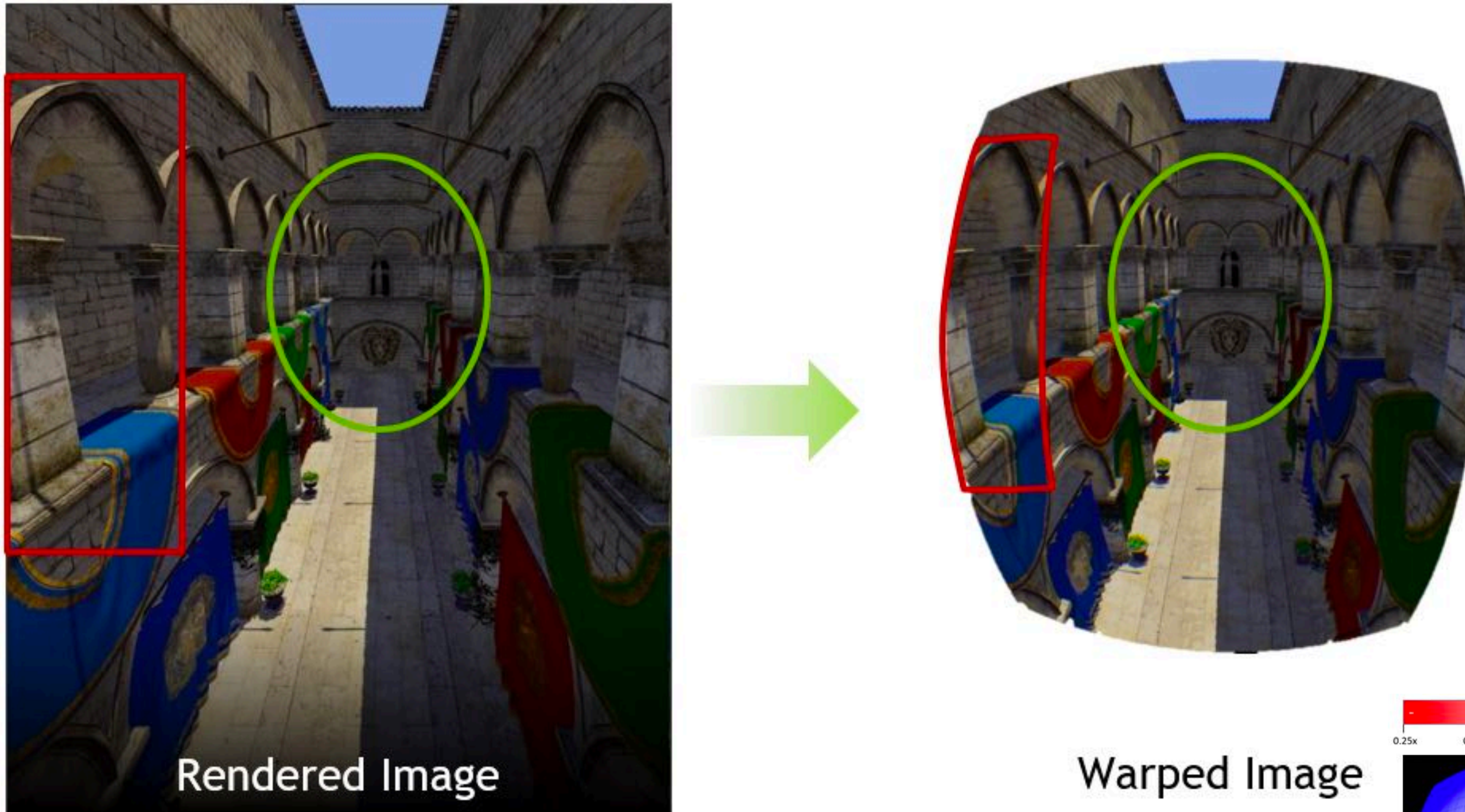
Step 1: render scene using traditional graphics pipeline at full resolution for each eye

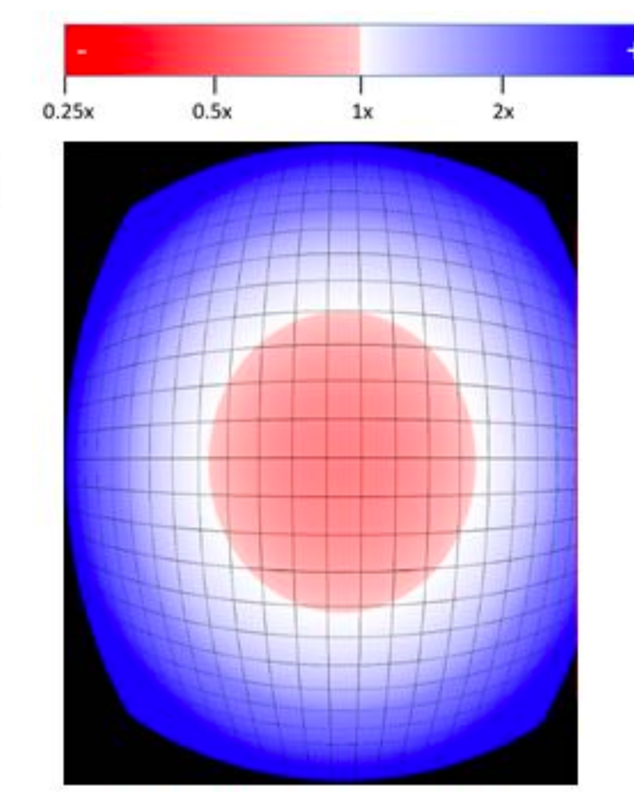
Step 2: warp images so rendering is viewed correctly when screen viewed under lens distortion

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

Image credit: Oculus VR developer guide

Problem: rendering at higher resolution than needed at periphery





Shading Rate After Lens Warp

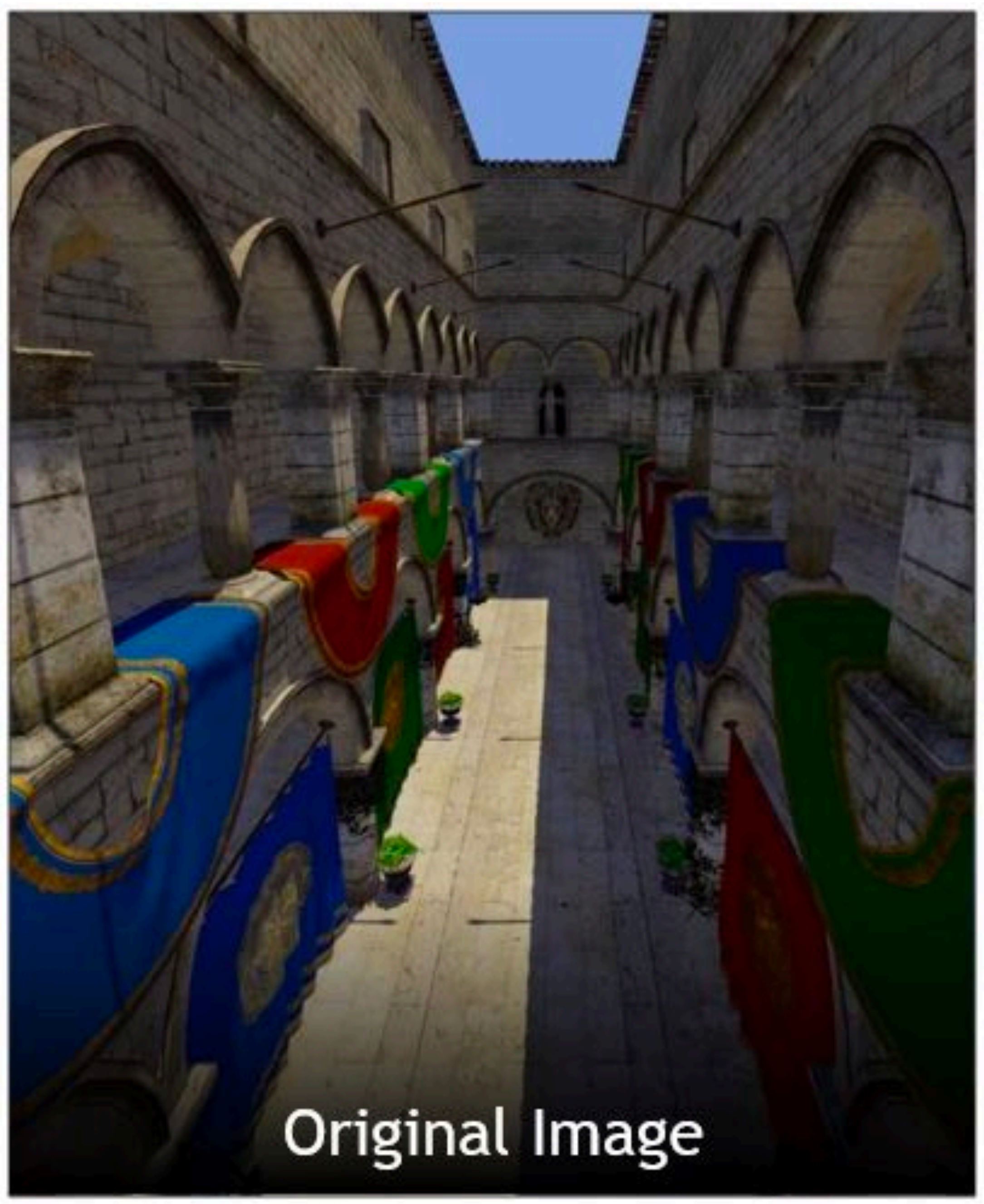
Performing unnecessary rendering work in the periphery due to:

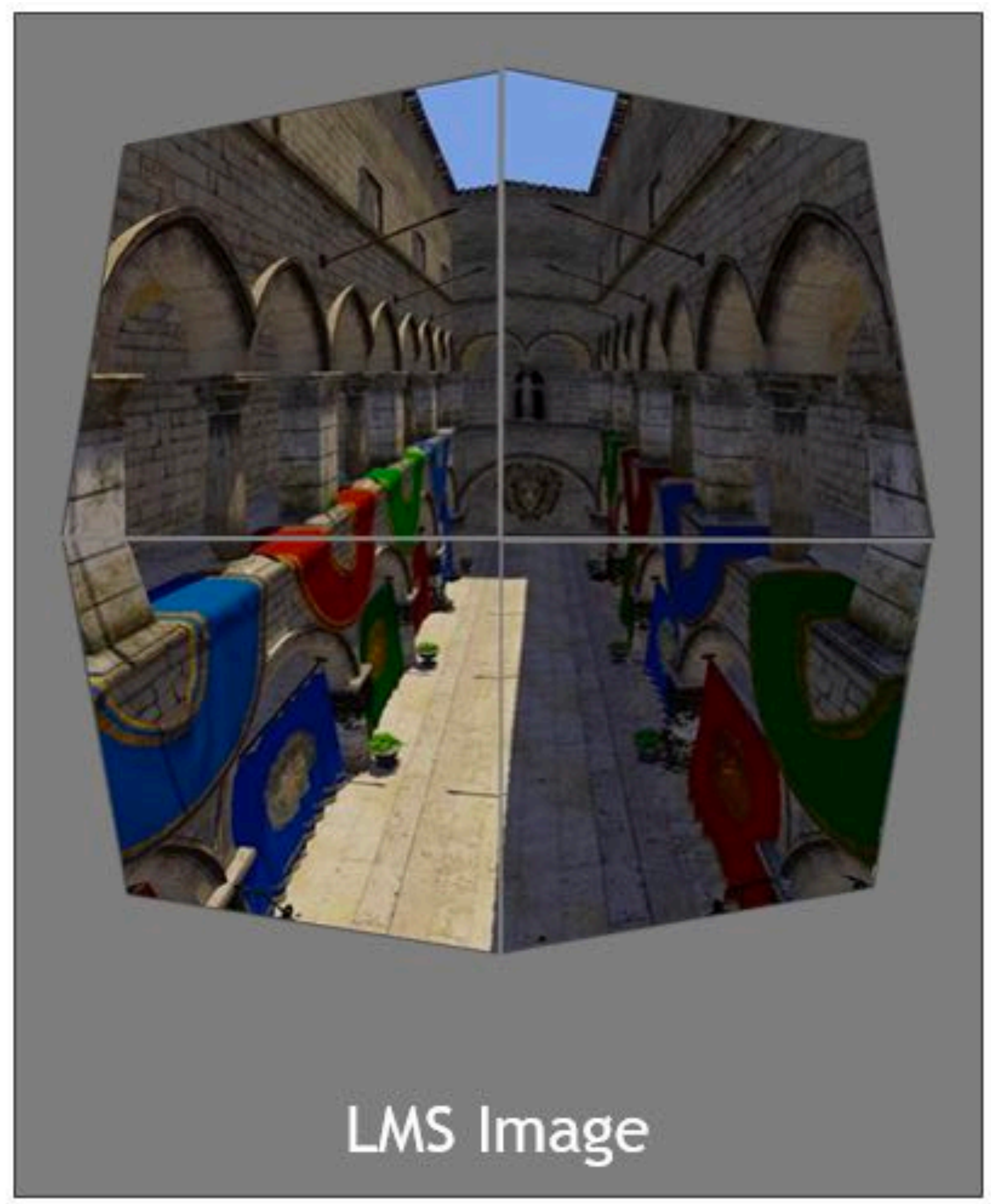
1. **Warp to reduce optical distortion (result: sample shading more densely in the periphery than in center of screen)**
2. **Eye has less spatial resolution in periphery (assuming viewer's gaze is toward center of screen)**

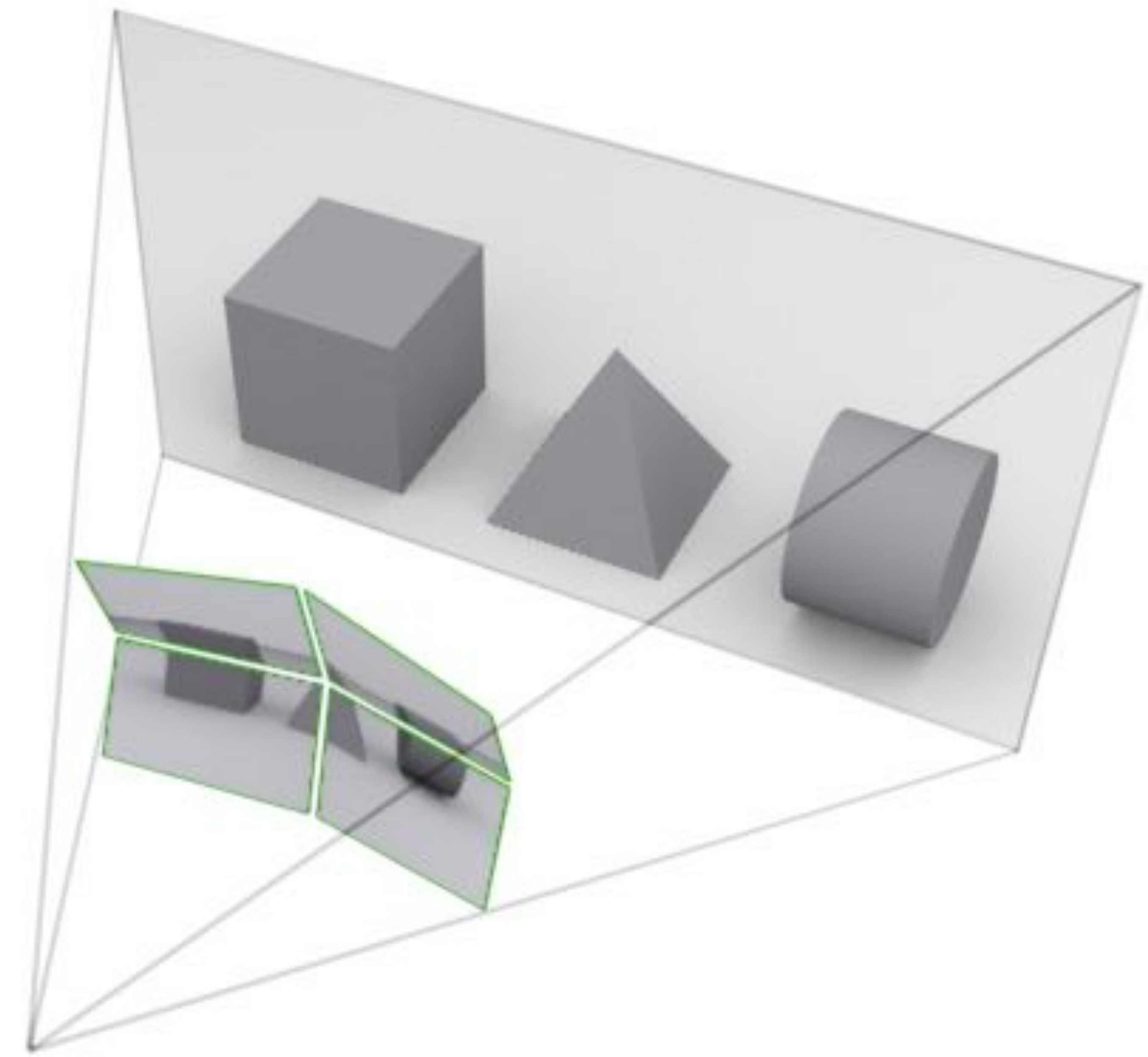
[Image credit: NVIDIA]

Modern solution: lens matched shading

- Render scene with four viewports, each has different projection matrix
- "Compresses" scene in the periphery (fewer samples), while not affecting scene near center of field of view







Note: lens matched shading results in more shading work toward the center of the screen (since users typically look to center, yields many benefits of more advanced eye tracking)

[Image credit: NVIDIA]

Need for low latency
(End-to-end head motion to photon latency)

Need for 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 from the display hit your eyes
    - Measure user’s head movement
    - Update scene/camera position
    - Render new image
    - Perform any distortion corrections
    - Transfer image 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

“Outside in” tracking: Oculus CV1 IR camera and IR LEDs (Early headset technology)

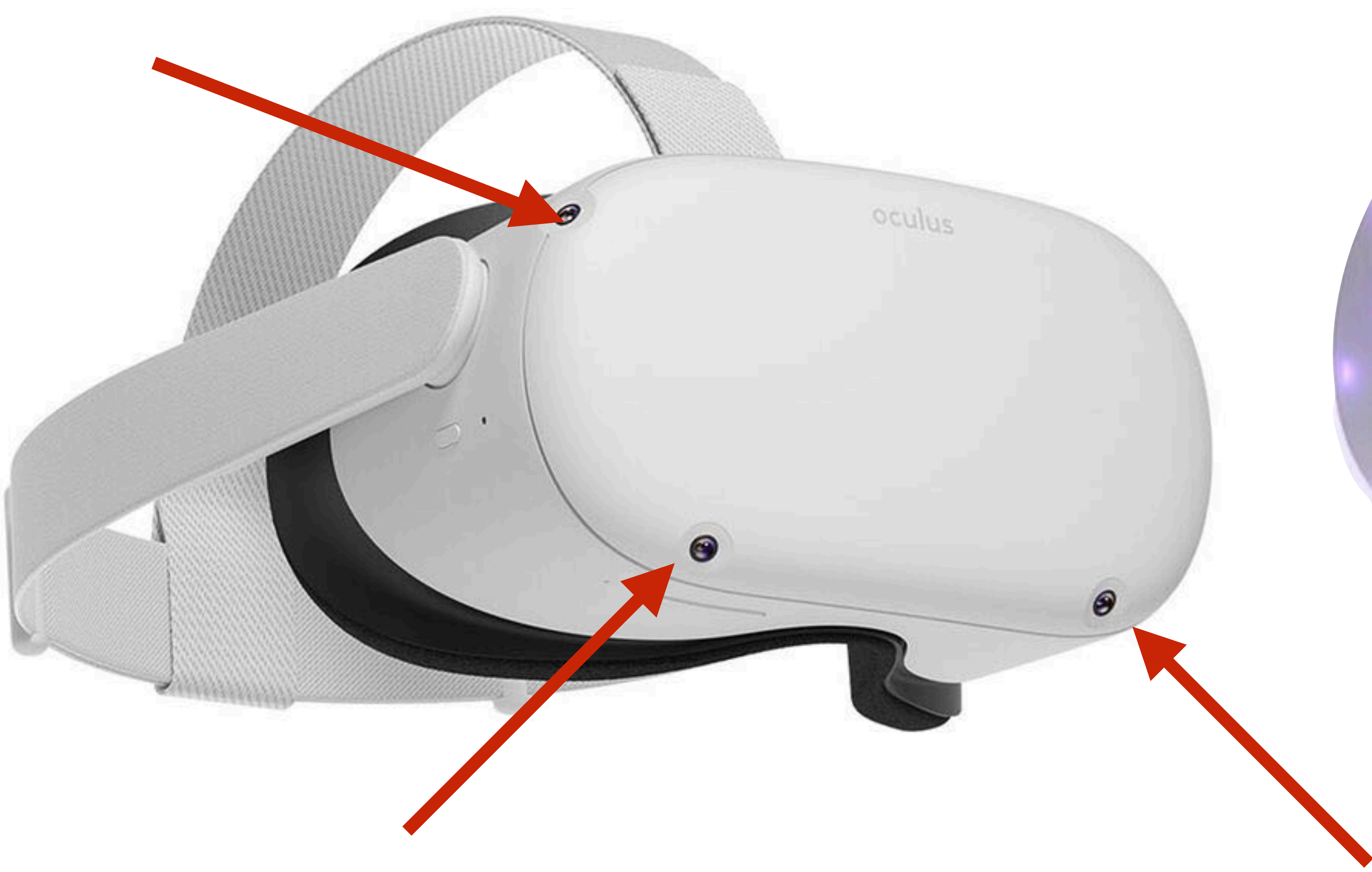
Headset contains:
IR LEDs (tracked by camera)
Gyro + accelerometer (1000Hz). (rapid relative positioning)

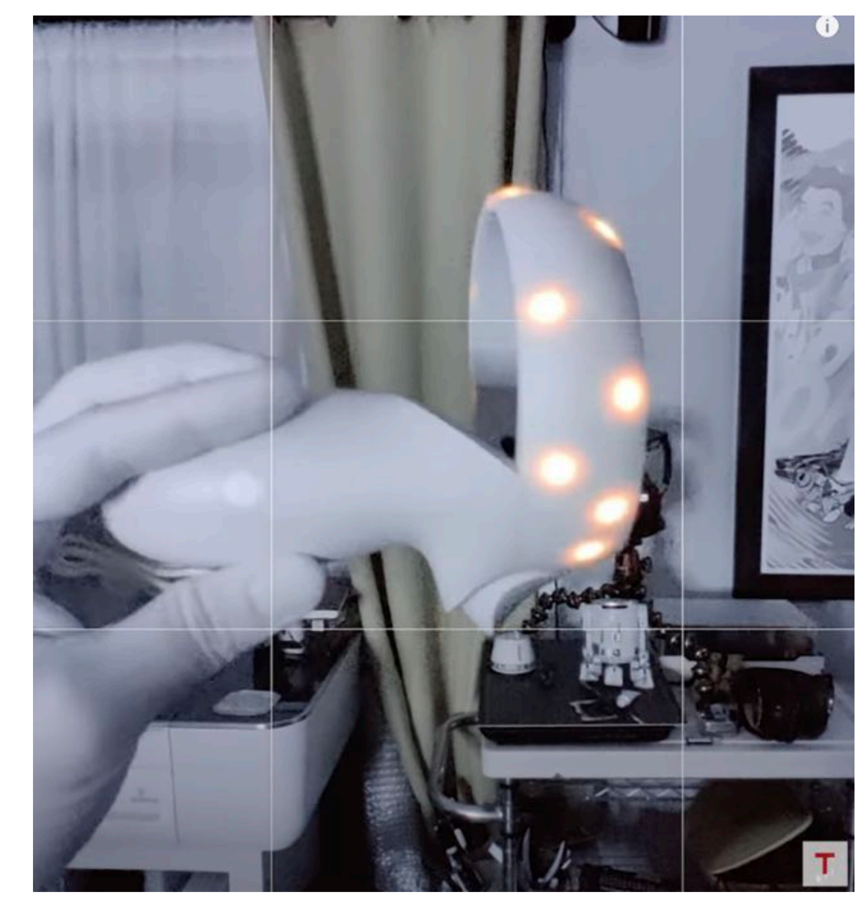
60Hz IR Camera (measures absolute position of headset 60 times a second)

Image credit: ifixit.com

Most modern systems use “inside out” tracking

- Wide-angle cameras look outward from headset
- Use computer vision (SLAM) to estimate 3D structure of world and position/orientation of camera in the world
- These cameras also track the position/orientation of the controllers
    - Quest 2 controllers have 15 infrared LEDs to aid tracking

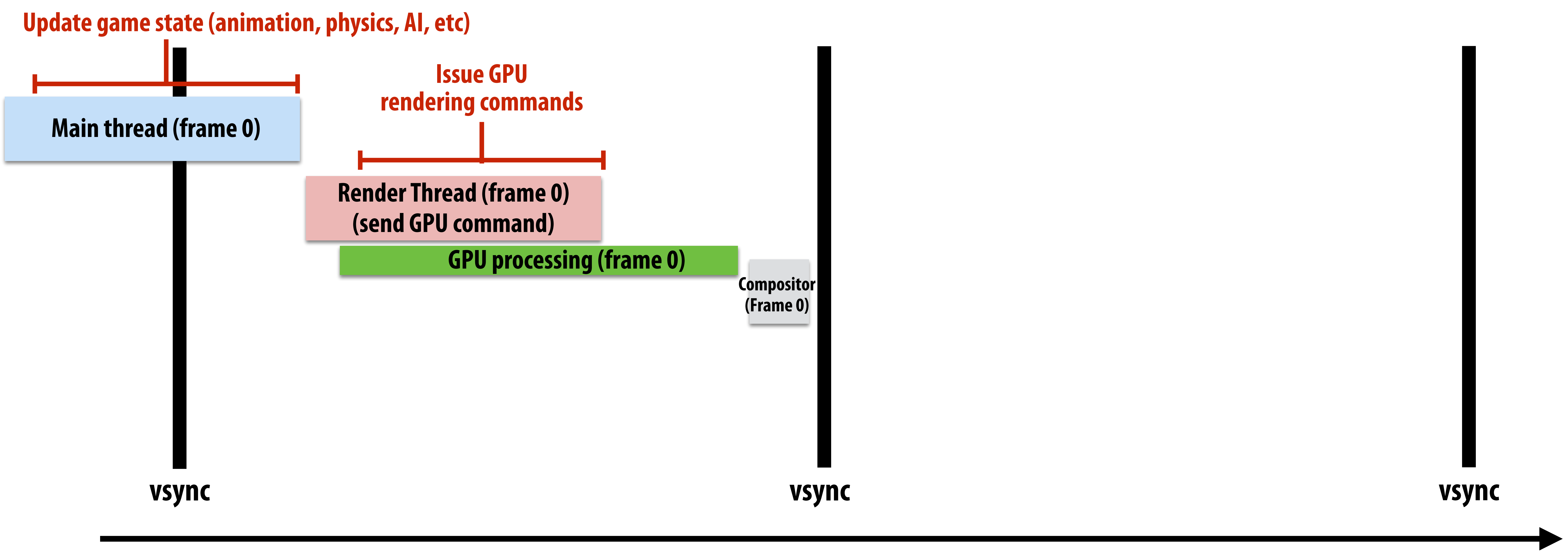




View of controller through infrared camera (credit Adam Savage’s Testbed)

Frame life cycle

- Goal: maintain as low latency as possible under challenging rendering conditions:
  - Battery-powered device (not a high-end desktop CPU/GPU)
  - High-resolution outputs (+ both left and right eye views)
  - Implication: can take awhile to render a frame 🤮



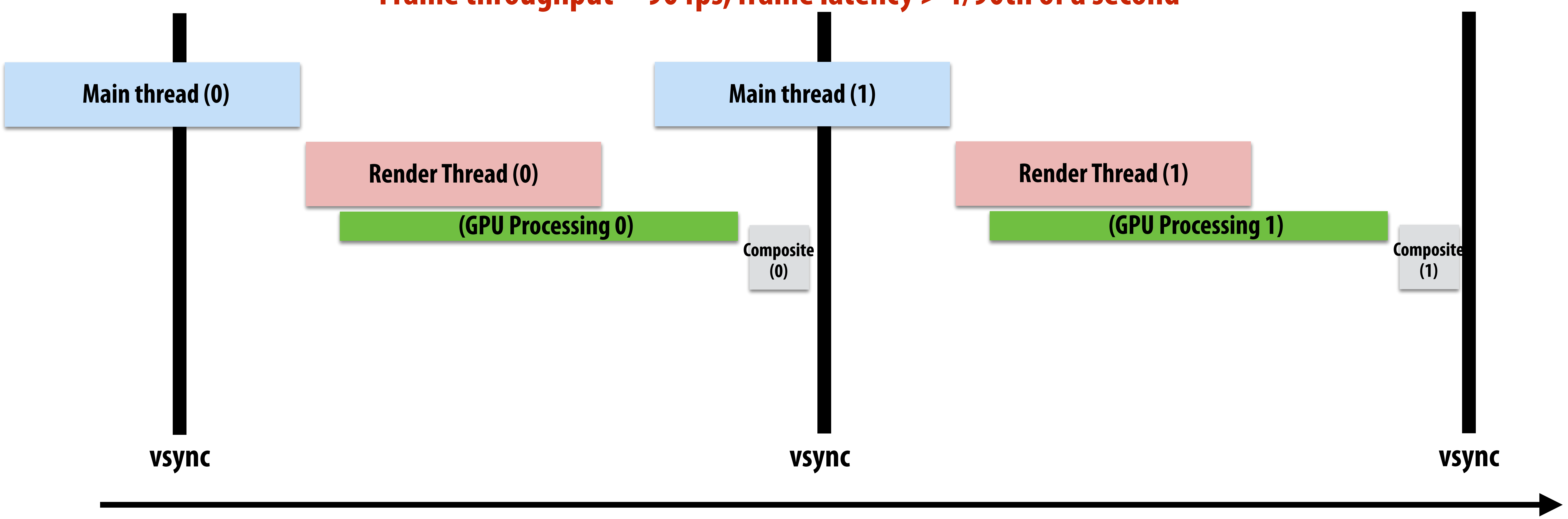
Frame life cycle

- **Goal: maintain as low latency as possible under challenging rendering conditions:**
  - Battery-powered device (not a high-end desktop CPU/GPU)
  - High-resolution outputs (+ both left and right eye views)
  - Implication: can take awhile to render a frame 🤮

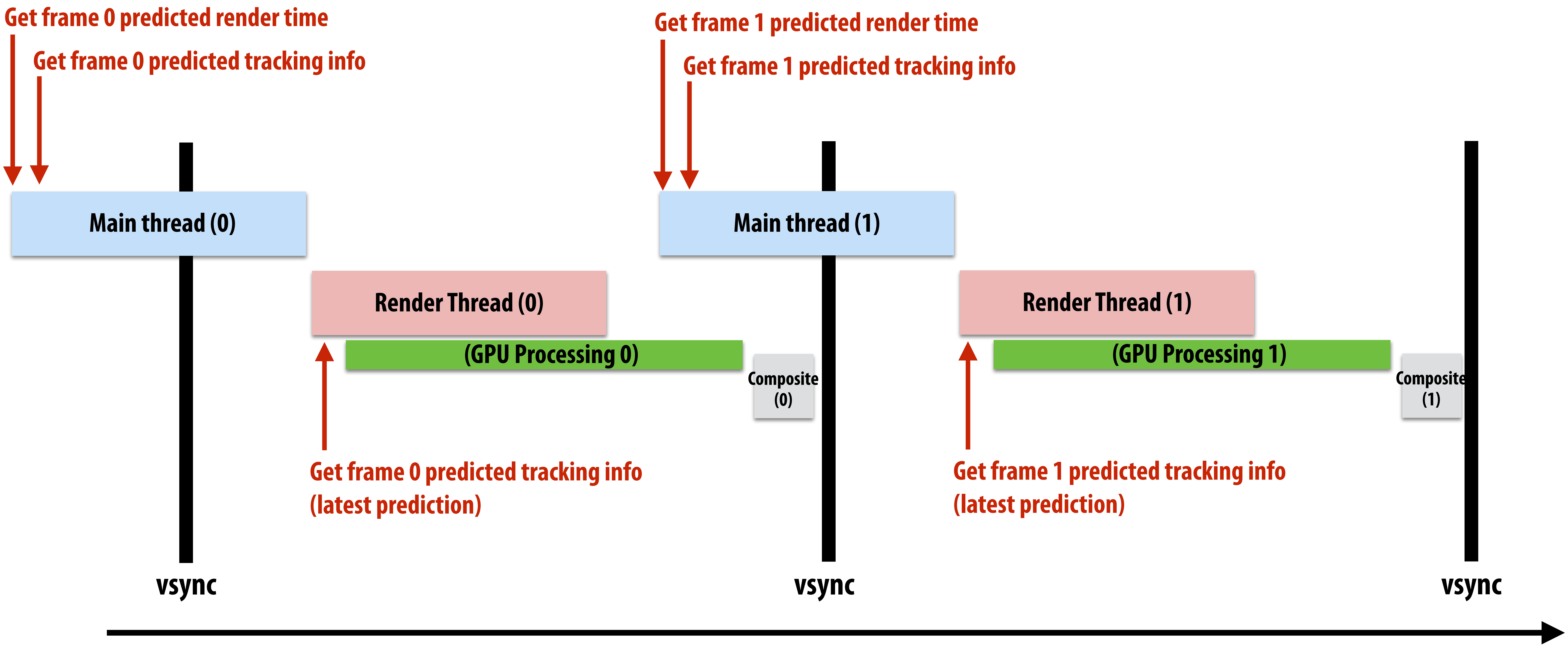
Pipelined view:

Assume vsync fires at 90 Hz

Frame throughput = 90 fps, frame latency > 1/90th of a second

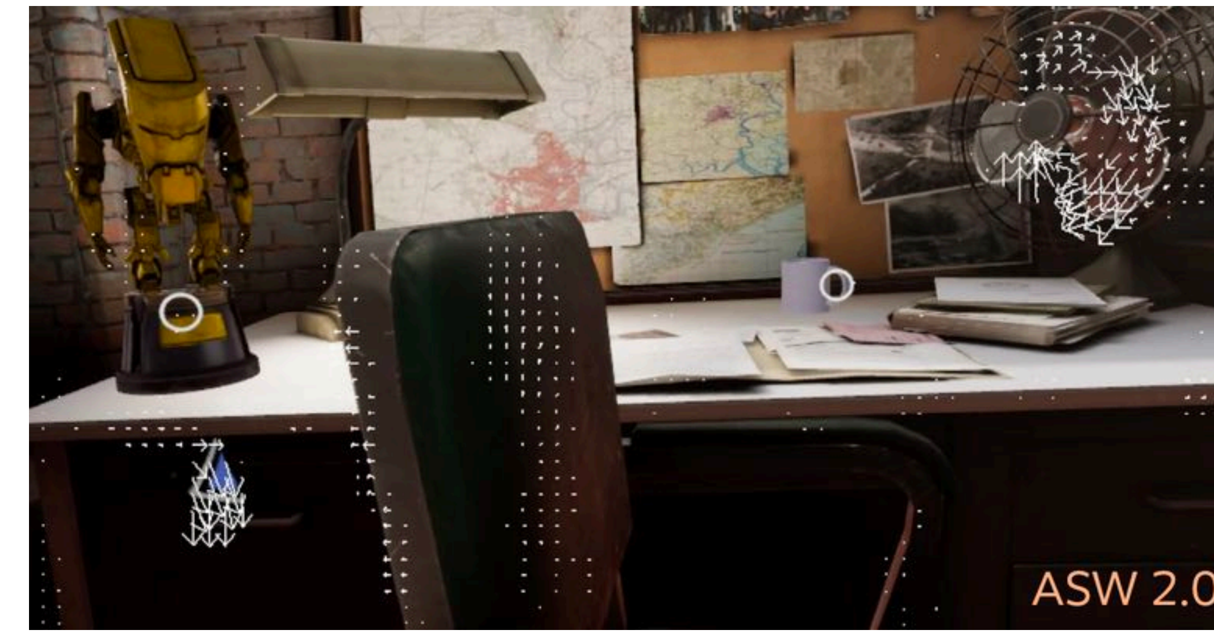


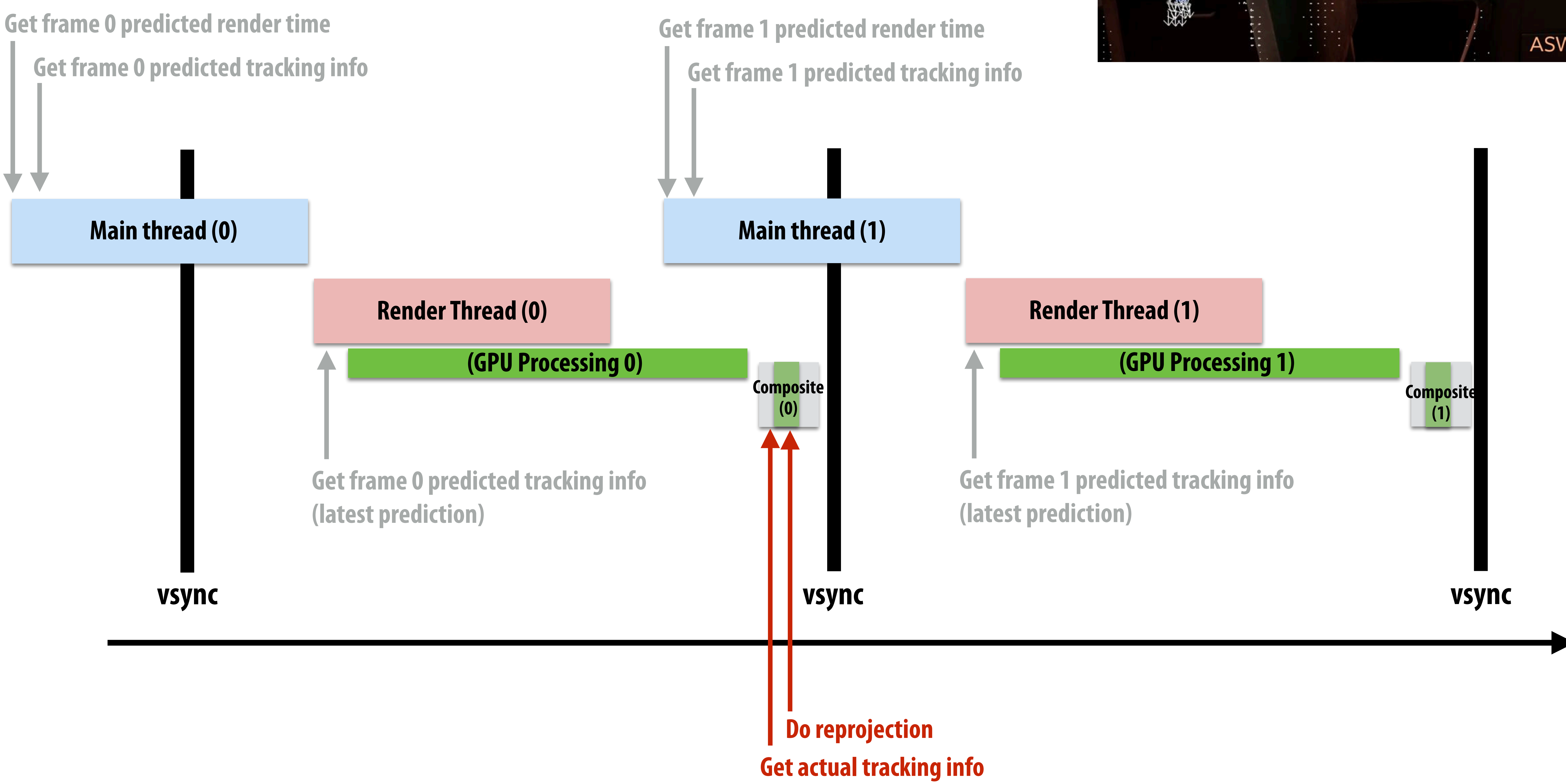
Frame life cycle



- Key ideas:
  - Game stated updated on "predicted" tracking info
  - Re-update head/controller tracking predictions right before drawing
  - Start next frame (frame 1 in this example) at last possible moment that gives it time to finish before target display time

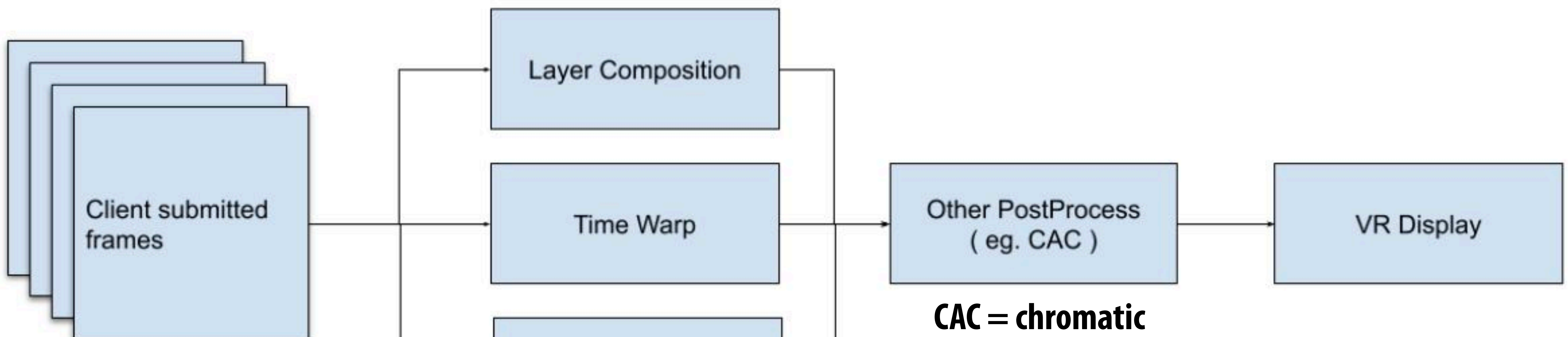
Reducing latency via reprojection





- Key idea ("time warp"): after rendering is complete, re-project rendered image to produce view of scene from most recent head position
- Accurate re-projection requires both rendered image and its depth map

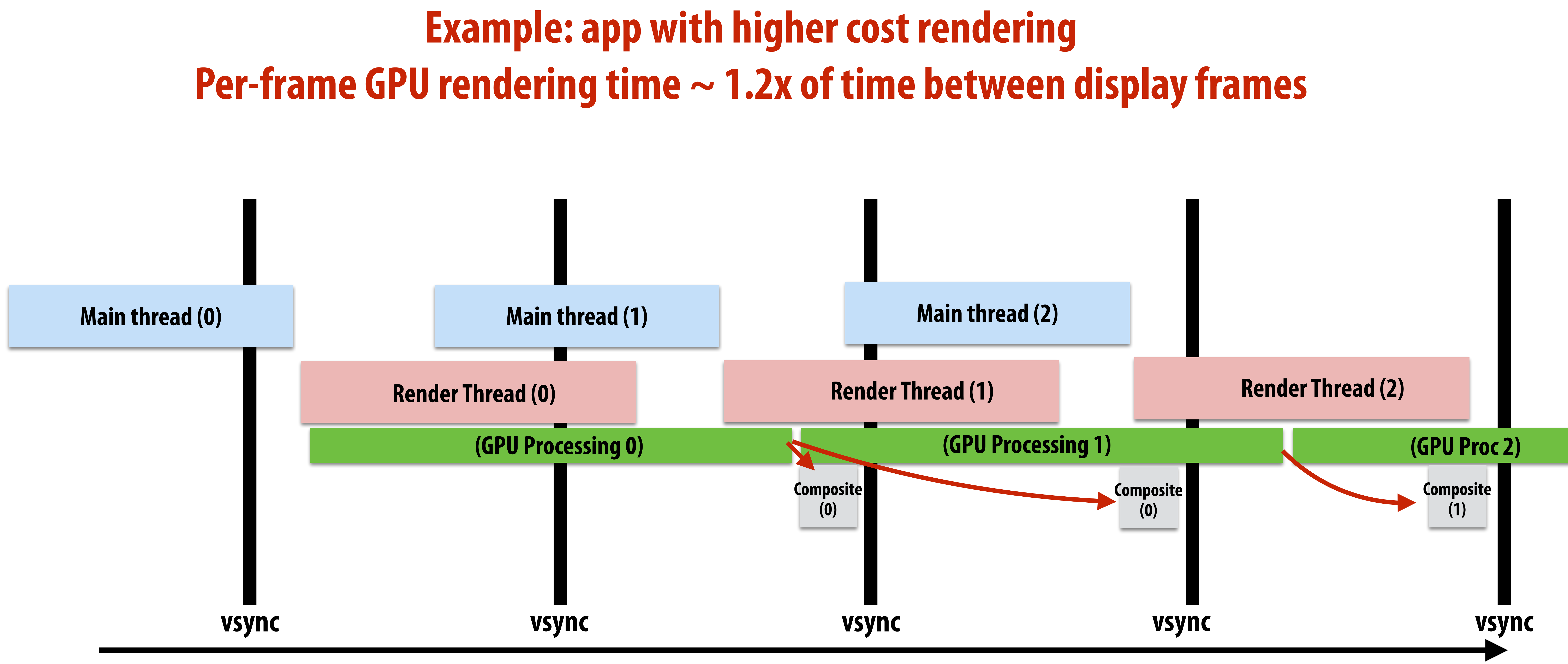
Oculus compositing pipeline



Frames come from multiple Oculus applications (Main app, Gaurdian, etc.)

CAC = chromatic aberration correction

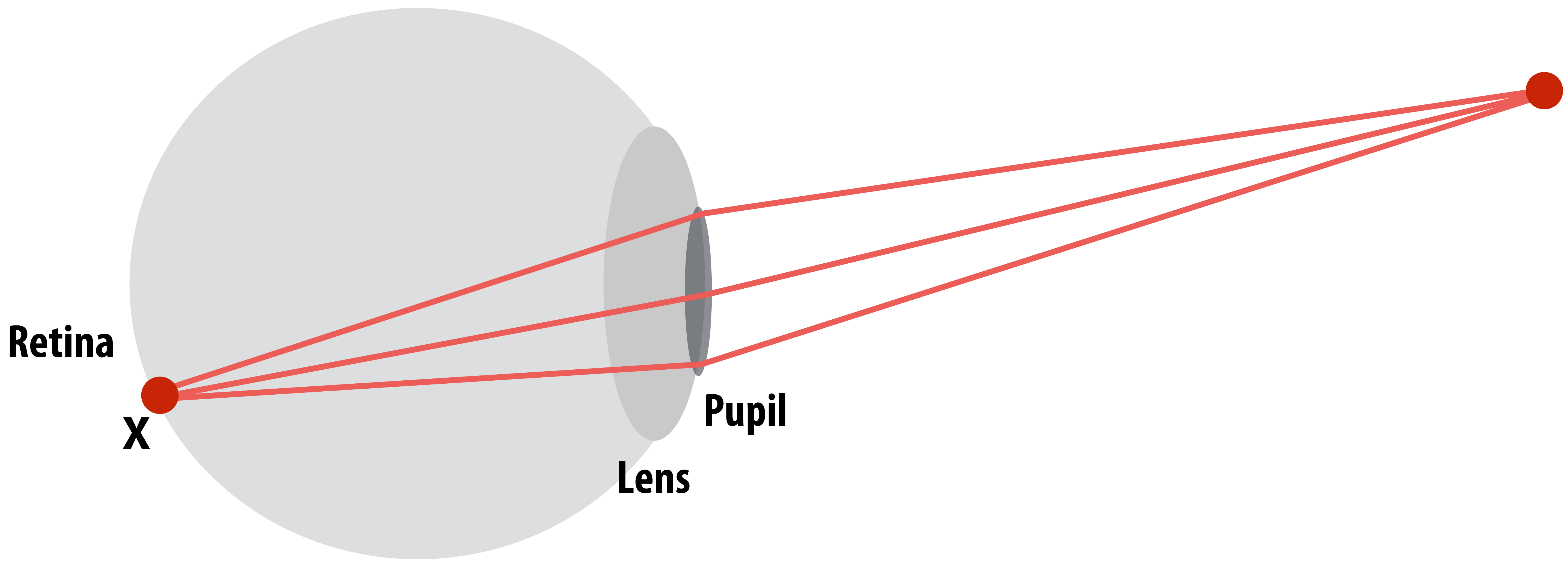
Increasing *frame rate* via reprojection



- Store last rendered frame
- If new frame not ready at time of next display, warp that last completed frame

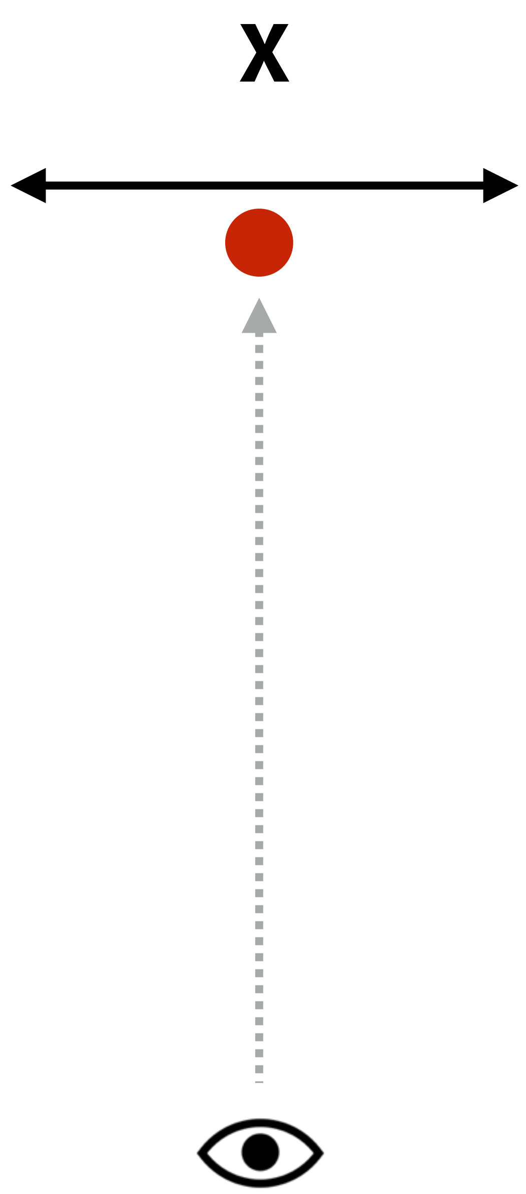
Accounting for interaction of:
display update +
display attached to head

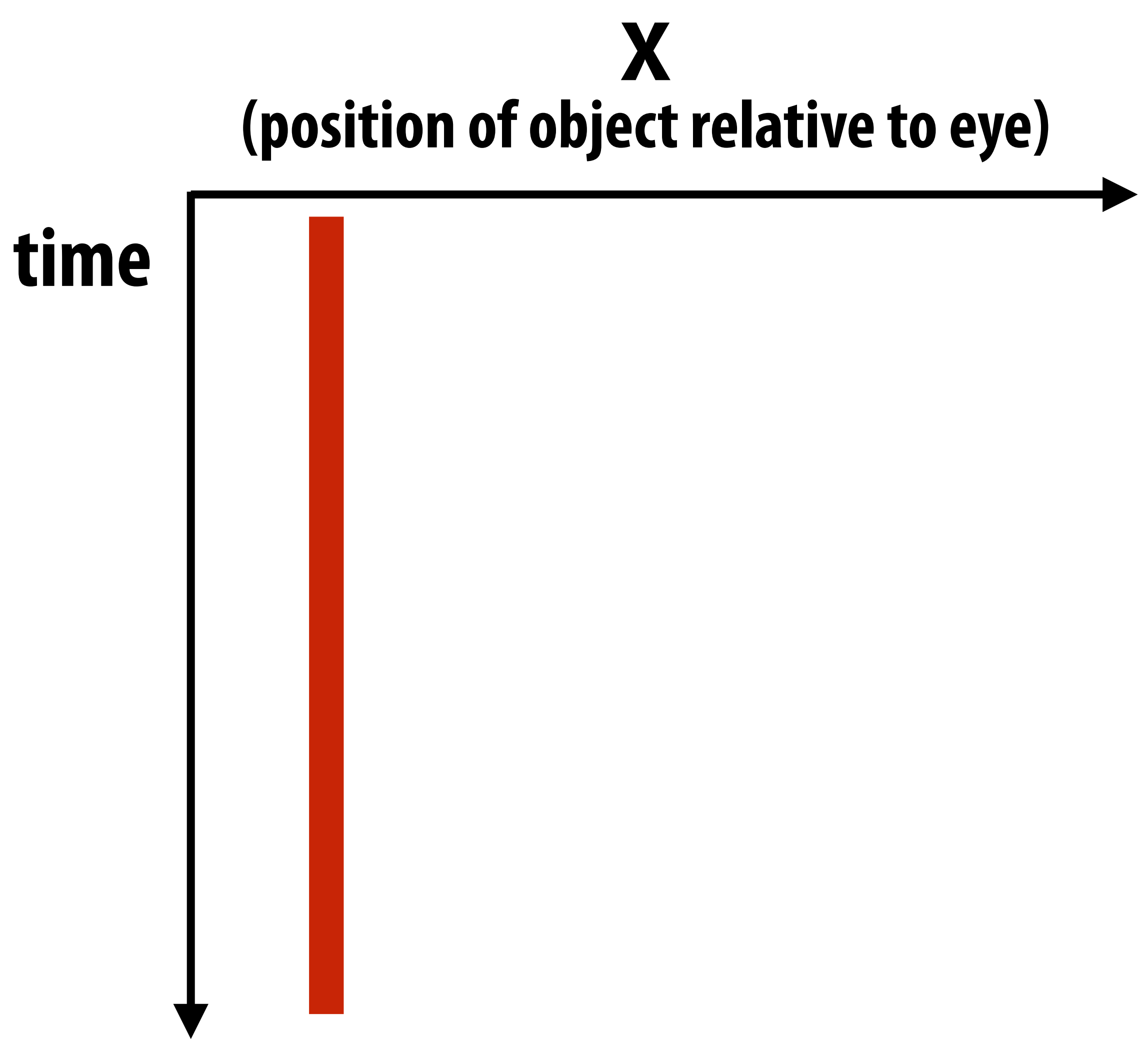
Consider projection of scene object on retina



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

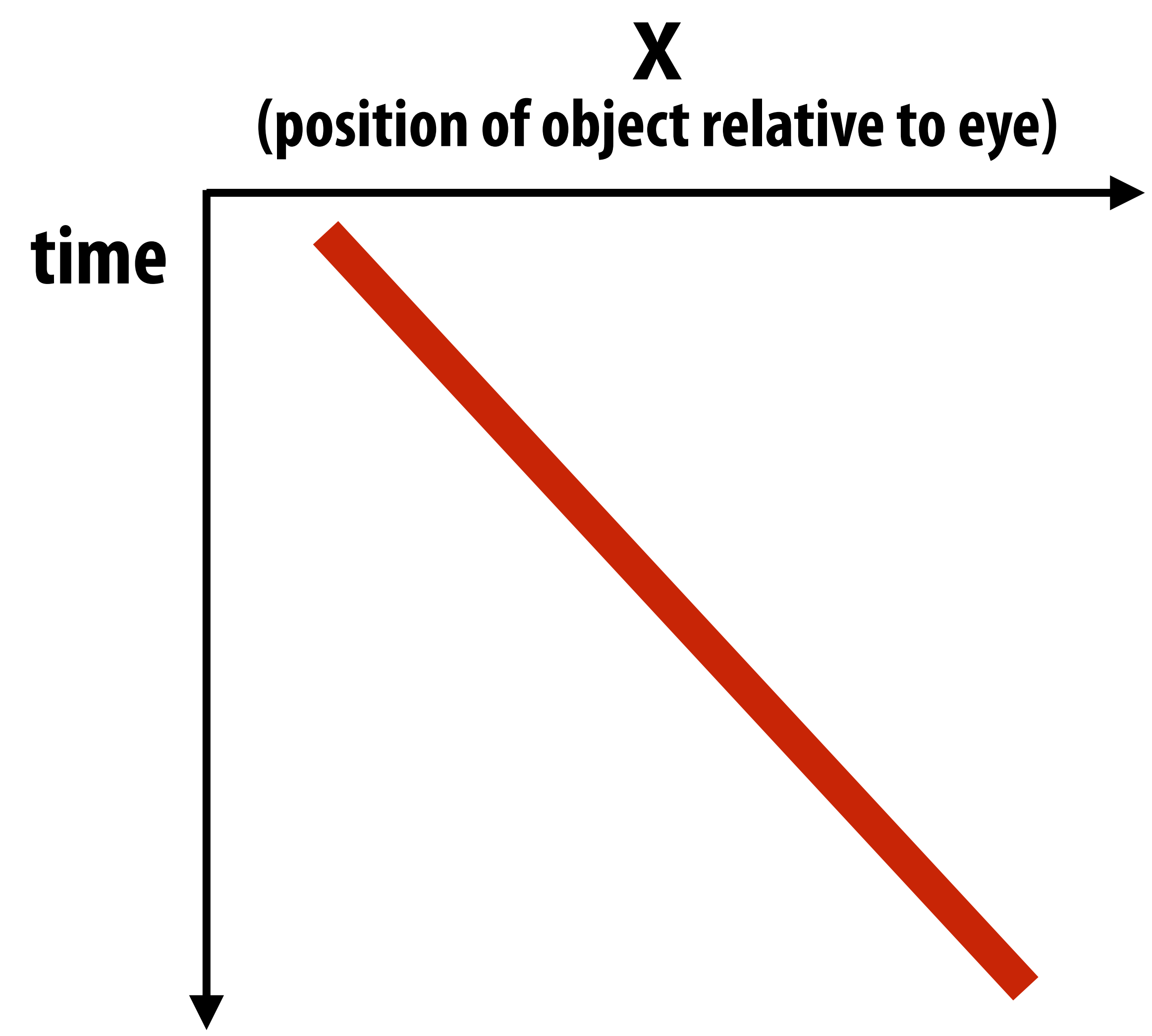
Consider object position relative to eye





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:

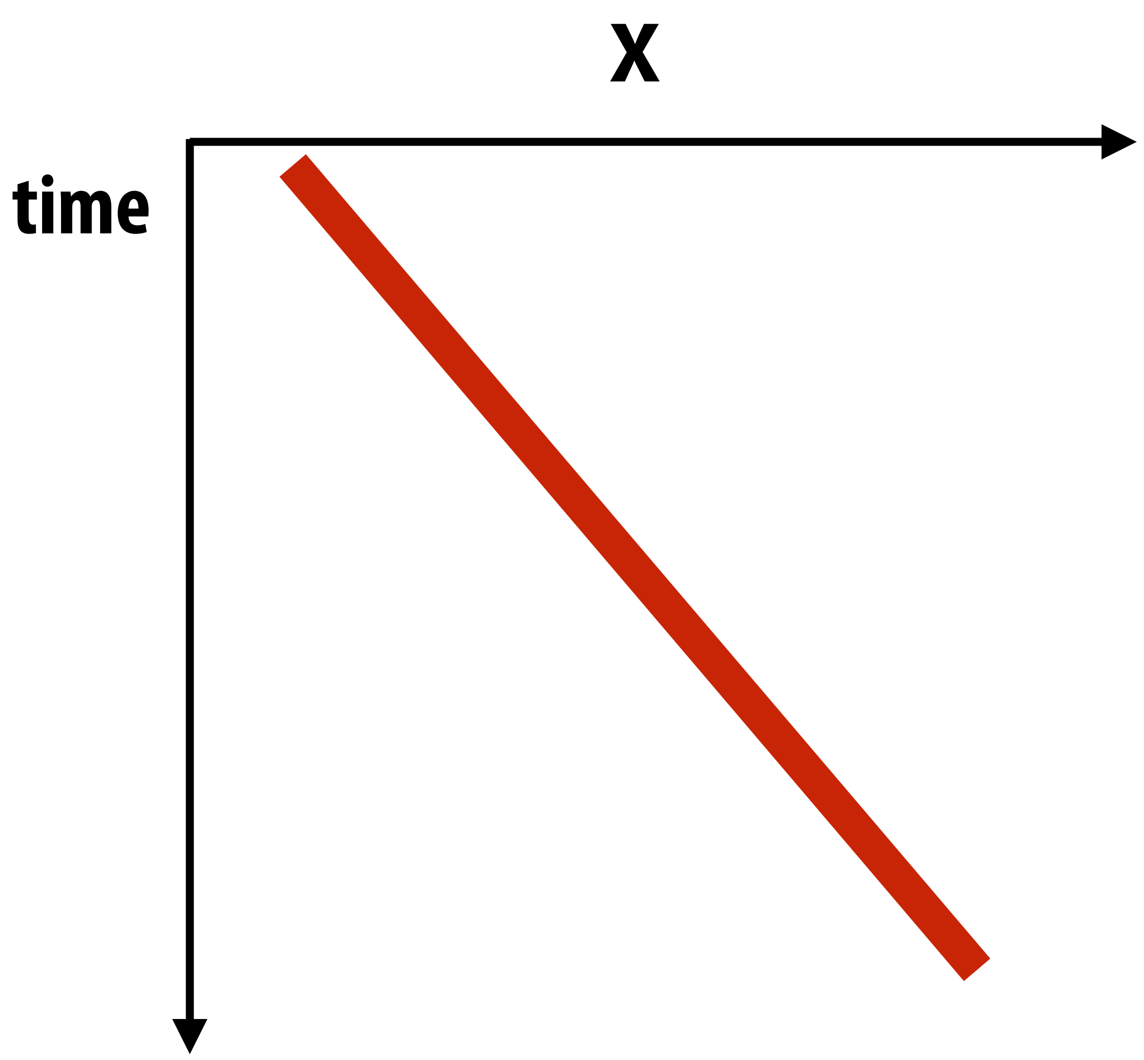
(red object moving from left to right but
eye stationary, i.e., it's focused on a different
stationary point in world)

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!!!

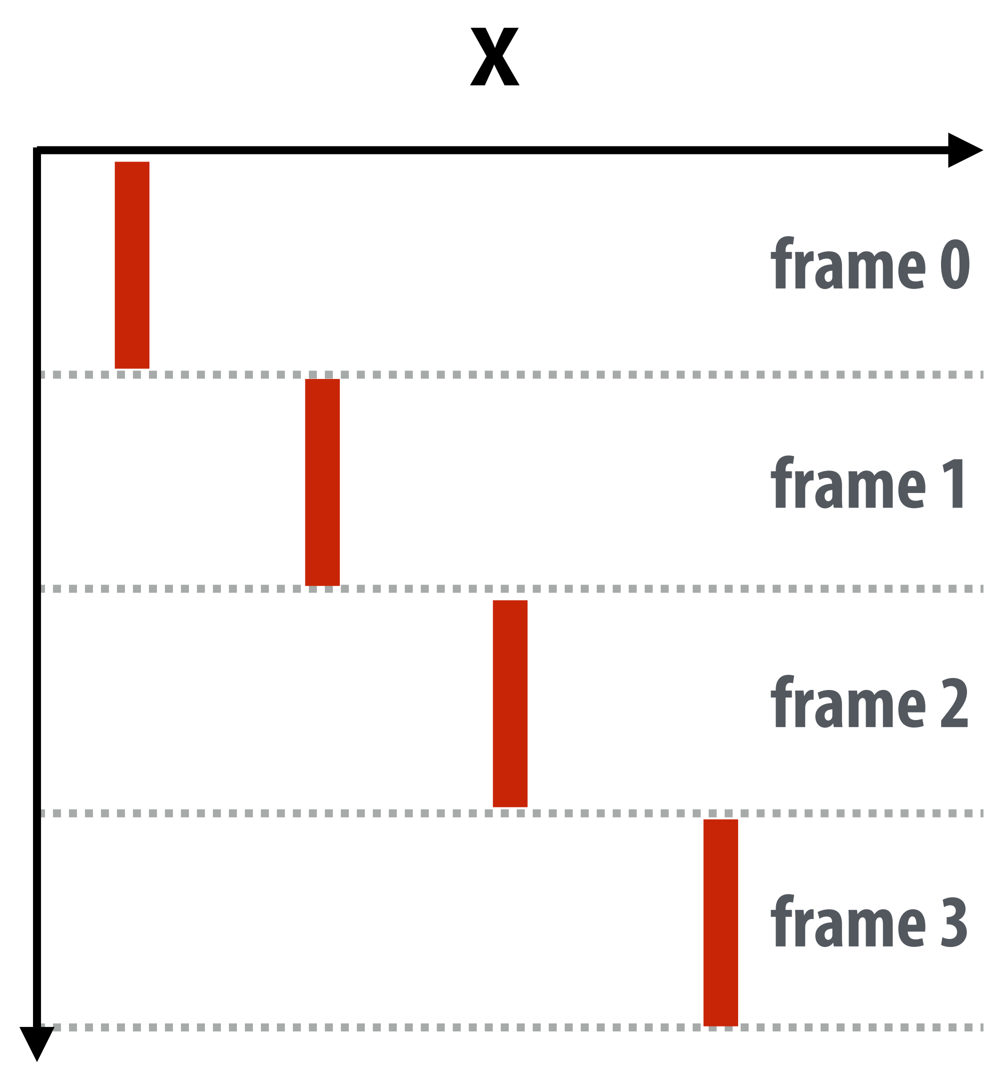
Spacetime diagrams adopted from presentations by Michael Abrash
Eyes designed by SuperAtic LABS from the thenounproject.com

Effect of latency: judder



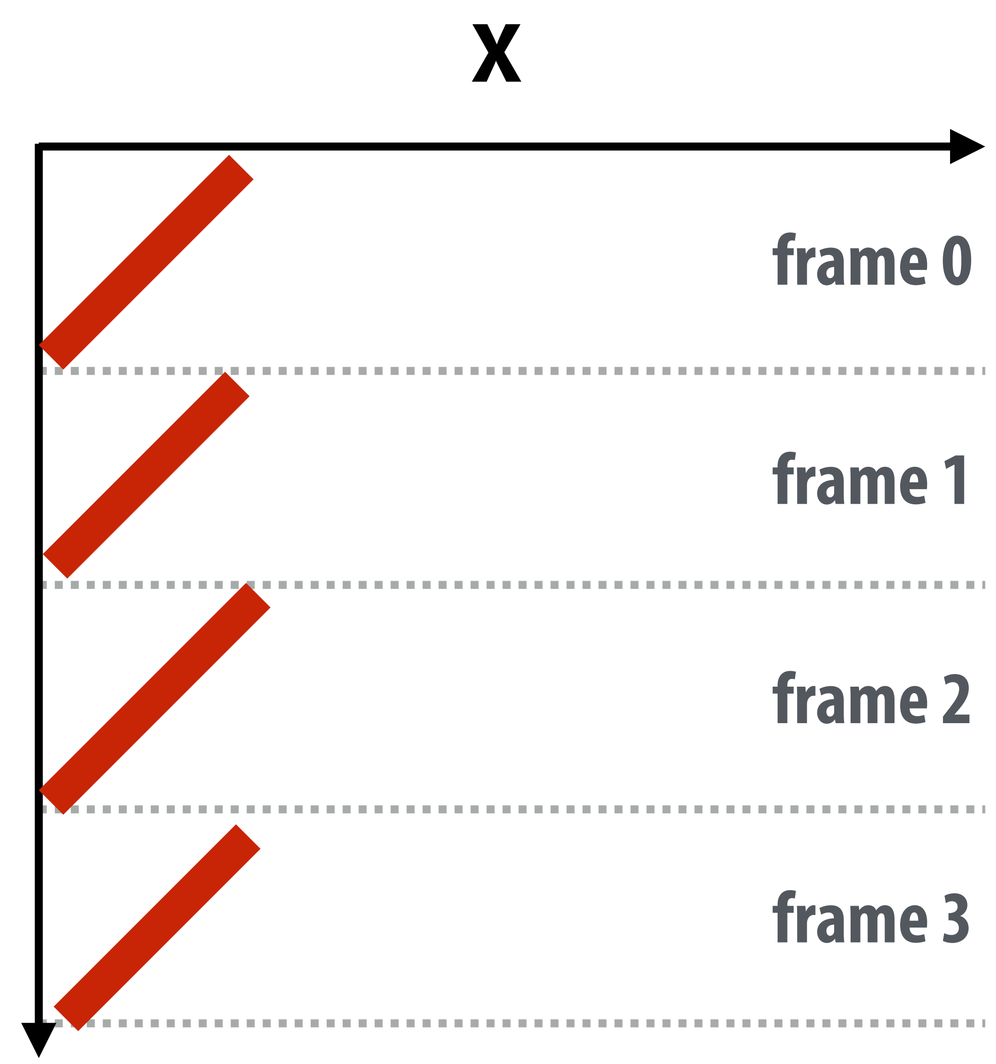
Case 2: object moving from left to right, eye stationary
(eye stationary with respect to display)

Continuous representation.



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

Light from display
(image is updated each frame)

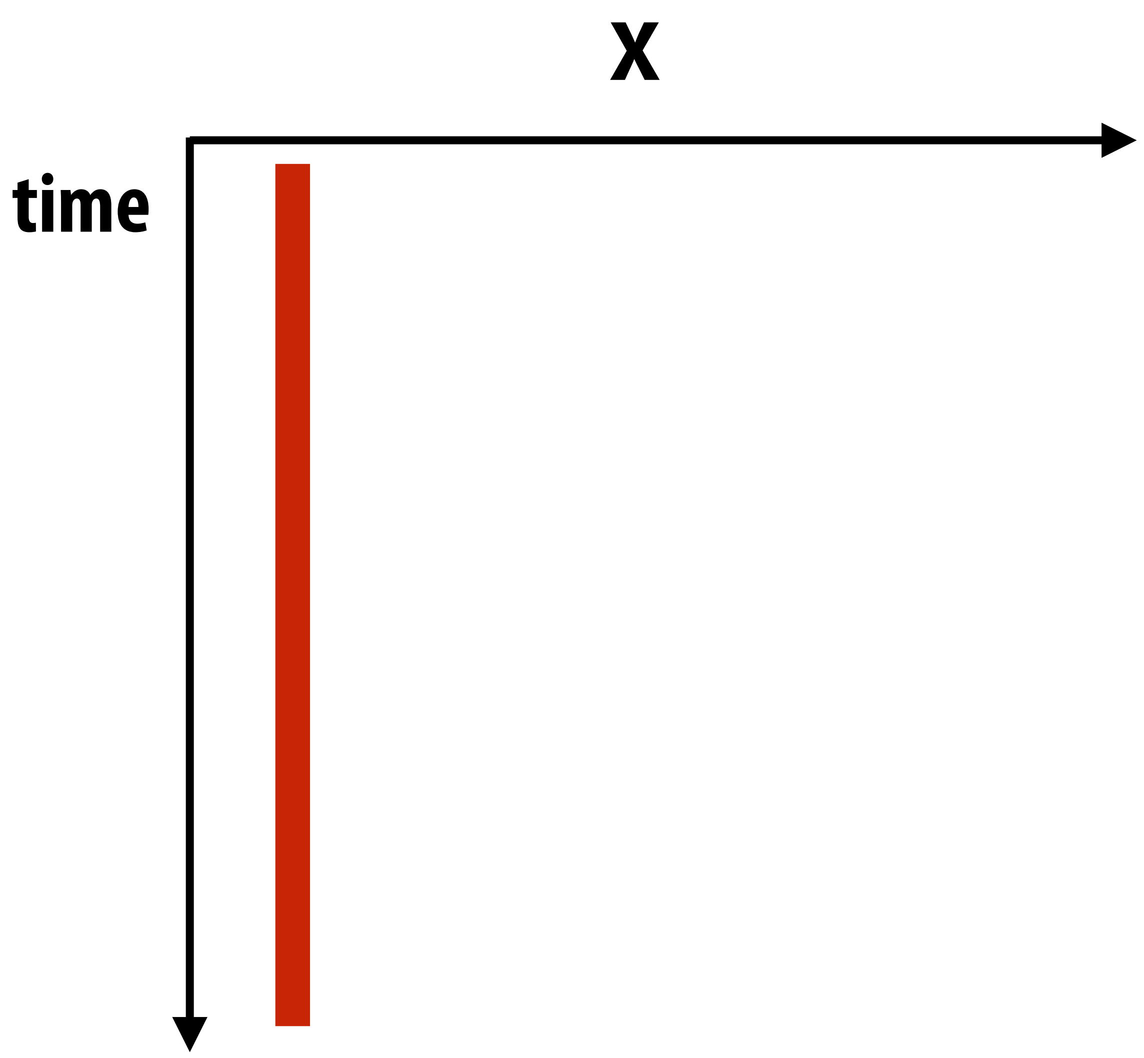


Case 1: object moving from left to right, eye moving continuously to track object
(eye moving relative to display!)

Light from display
(image is updated each frame)

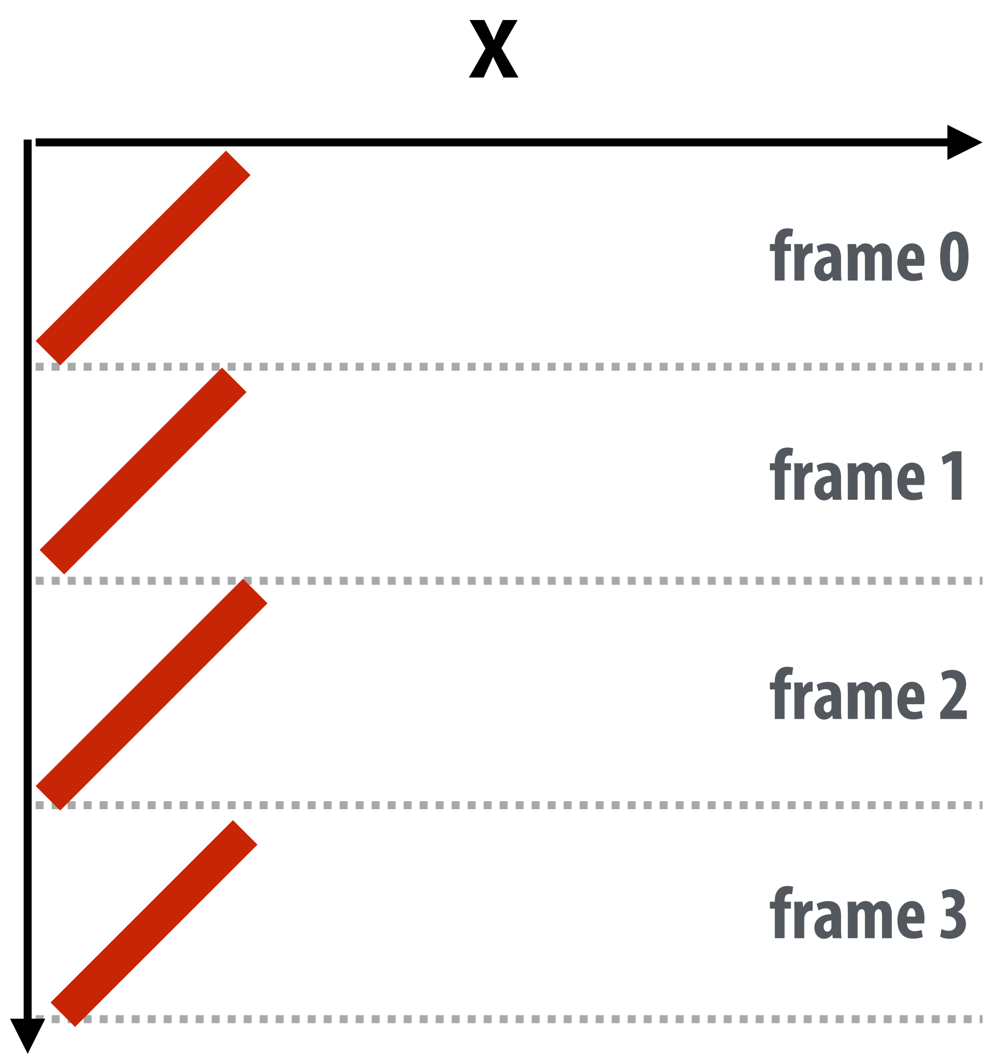
Case 1 explanation: since eye is moving, object's position is relatively constant relative to eye (as it should be since the eye is tracking it). But due discrete frame rate, object falls behind eye, causing a smearing/strobing effect ("choppy" motion blur). Recall from earlier slide: 90 degree motion,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

Reducing judder: increase frame rate

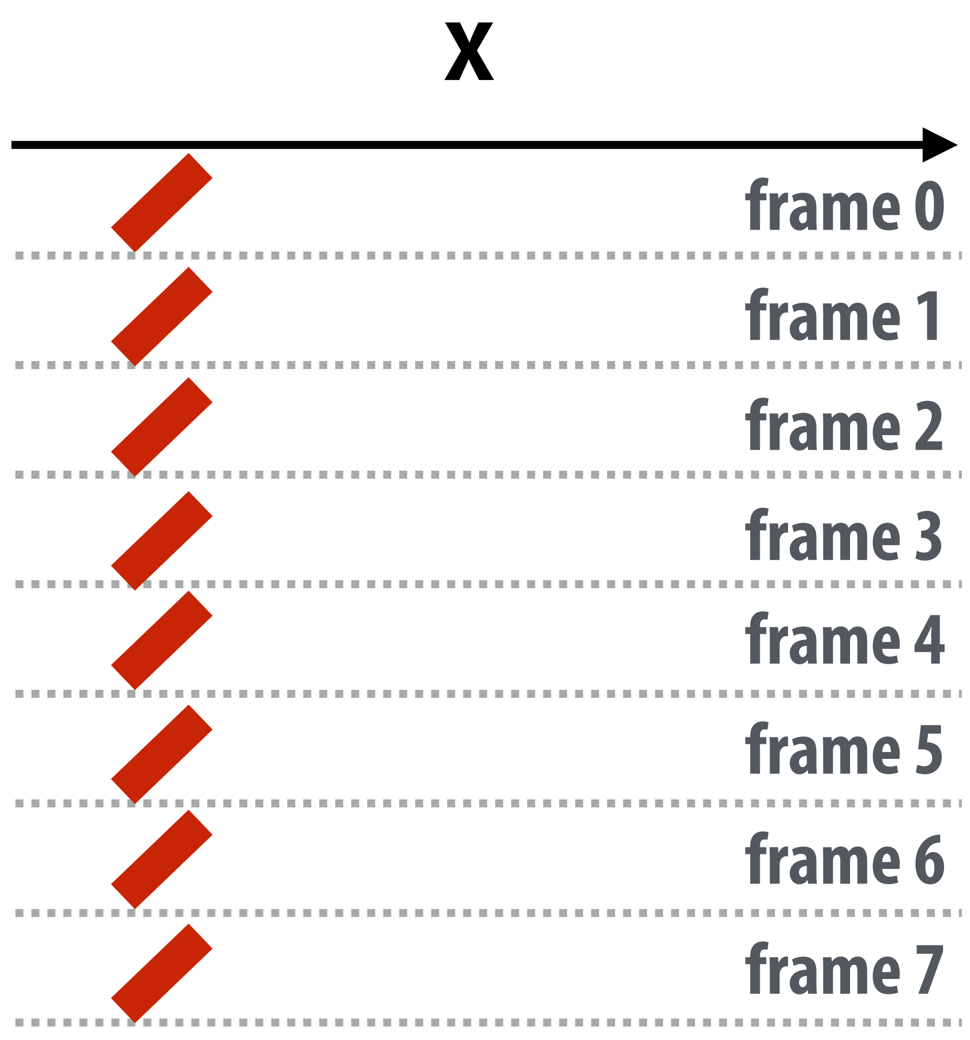


Case 1: continuous ground truth

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)

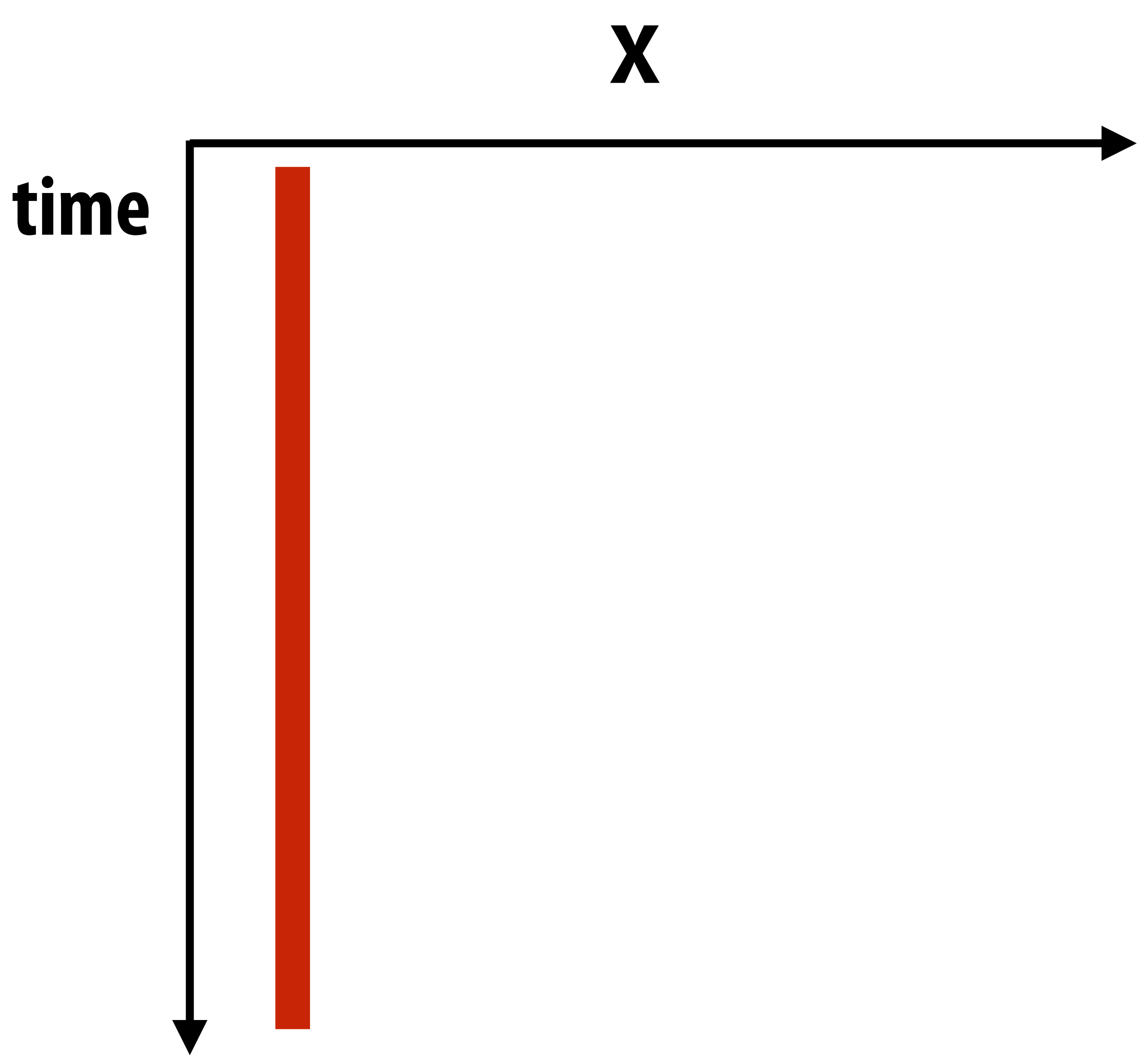


Light from display
(image is updated each frame)

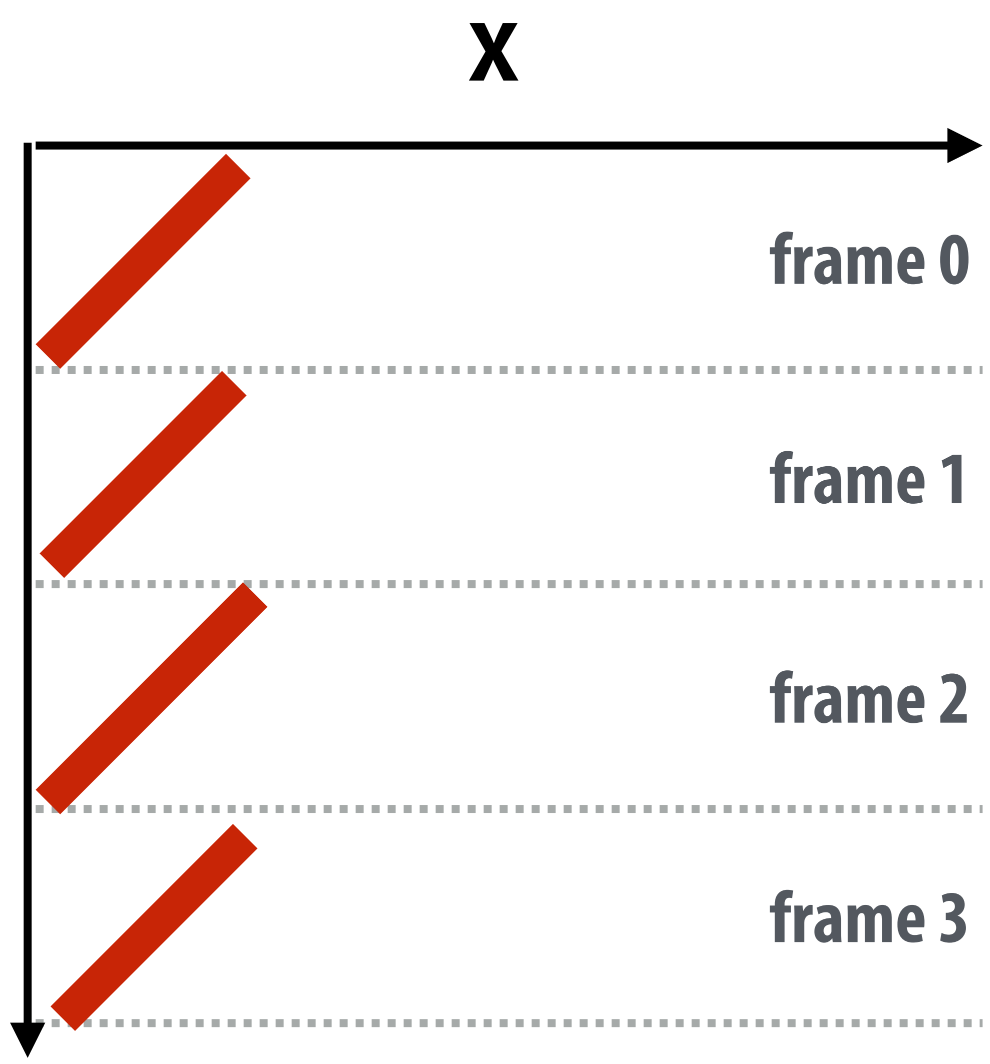
Higher frame rate results in closer
approximation to ground truth

Spacetime diagrams adopted from presentations by Michael Abrash

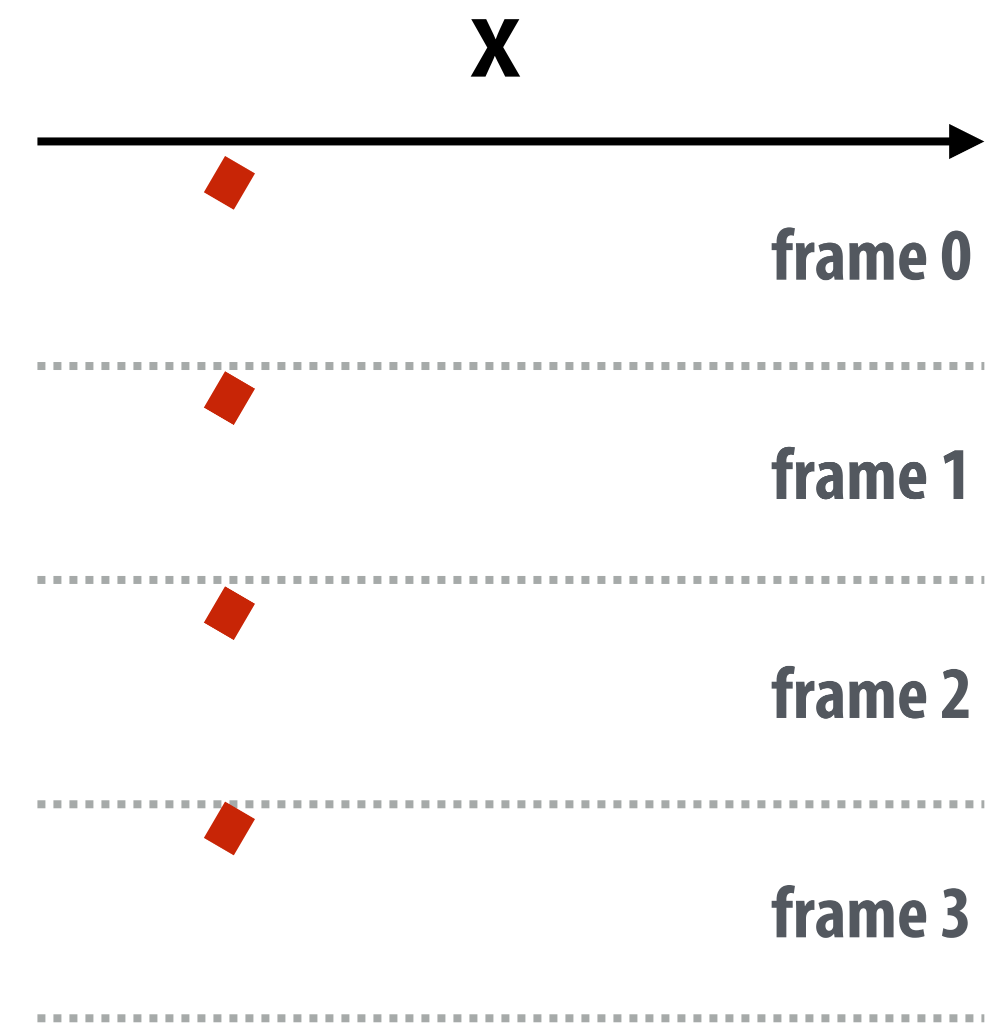
Reducing judder: low persistence display



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

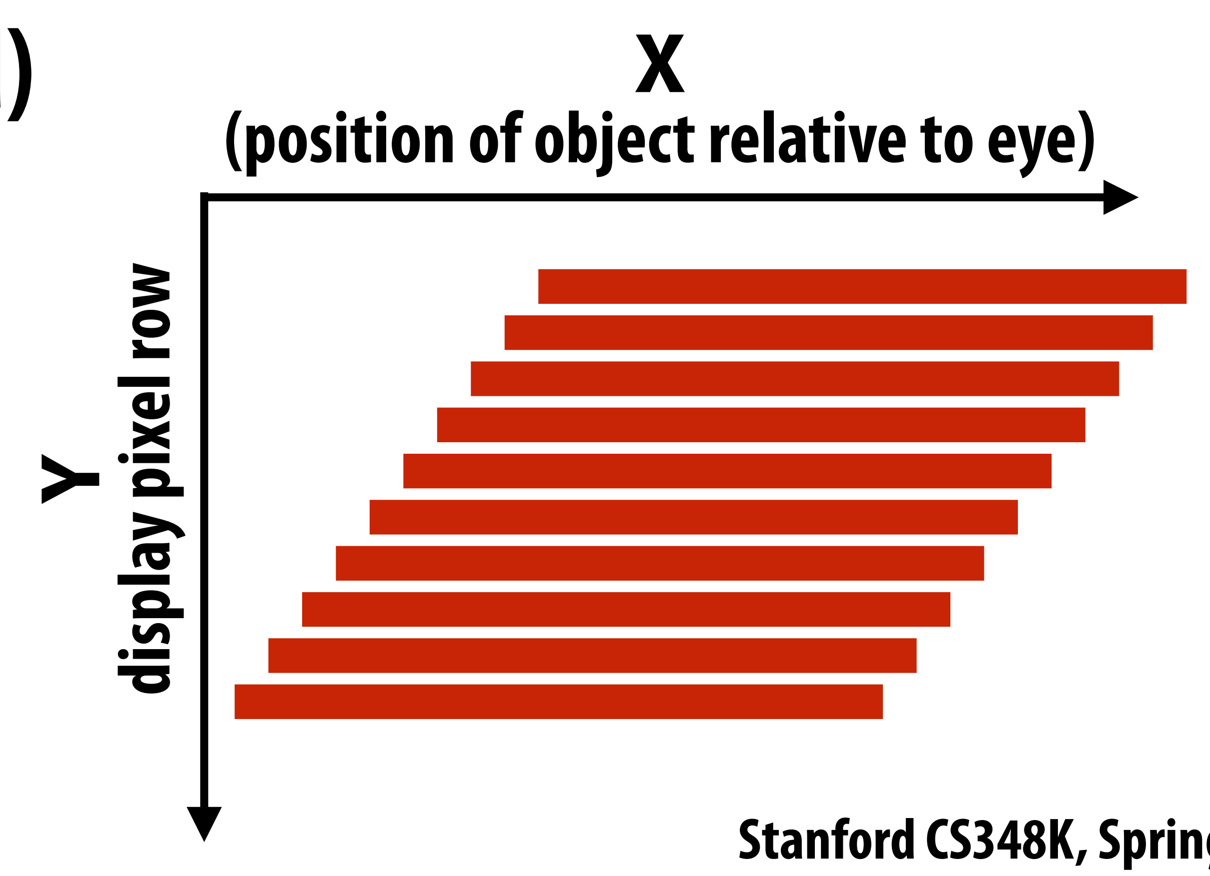
Full-persistence display: pixels emit light for entire frame
Low-persistence display: pixels emit light for small fraction of frame
Oculus Rift CV1 low-persistence display

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

Spacetime diagrams adopted from presentations by Michael Abrash

Artifacts due to rolling 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 eye moving horizontally relative to display (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.

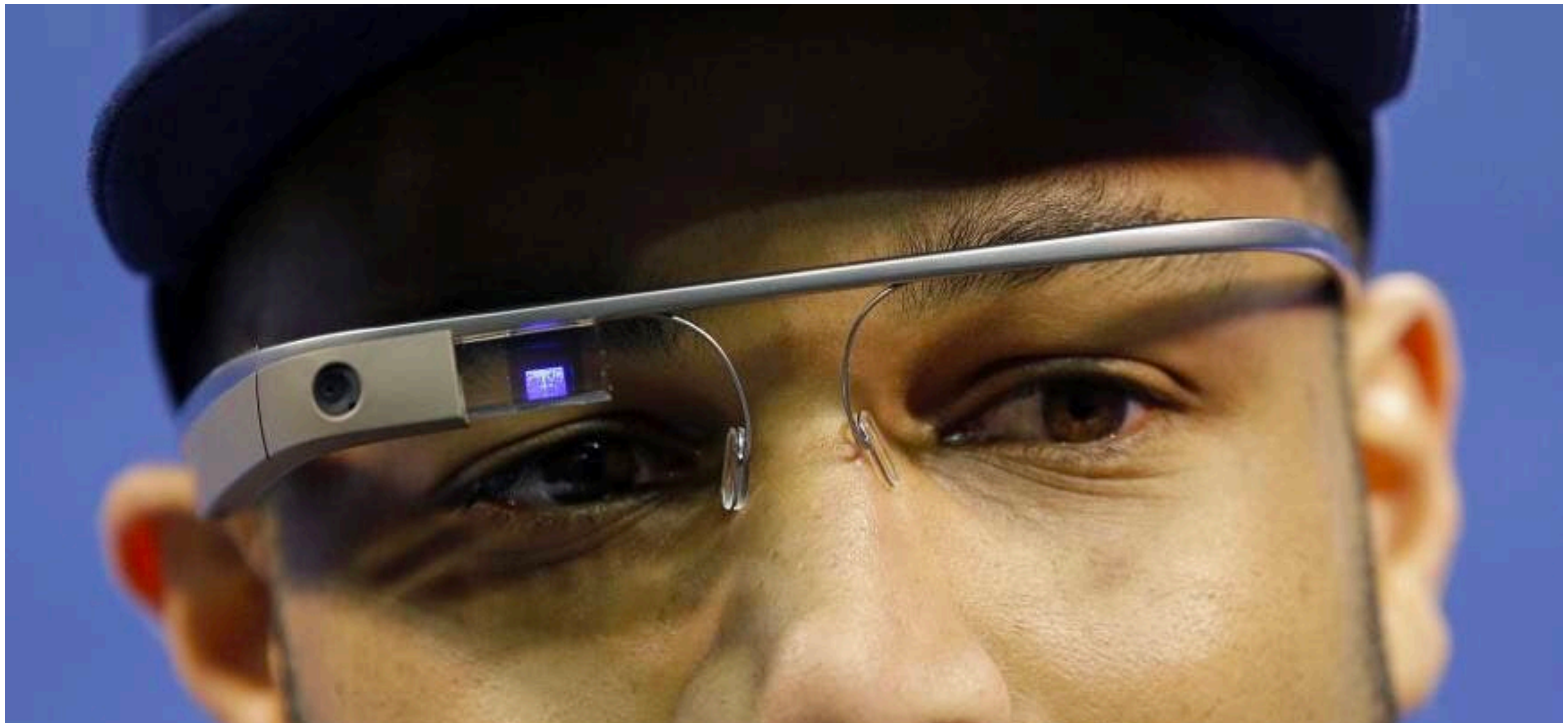
Compensating for rolling backlight

- 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 tracing? (each row of camera rays samples scene at a different time)

Reducing bulky form factor + AR

Glasses form factor (for AR applications)



Google Glass (2013)

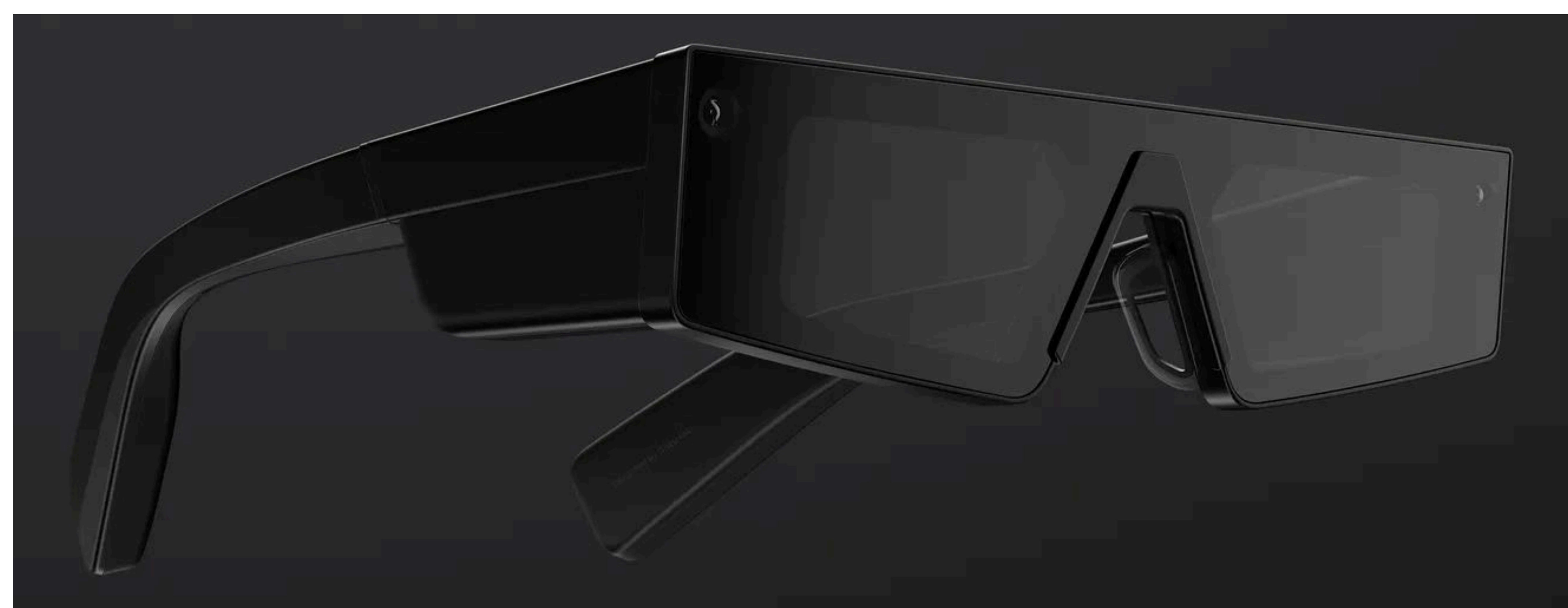
Qualcomm "AR smartviewer" glasses (tethered to smartphone)

Ray-Ban | FACEBOOK

Additional announcements (or rumors) by Google, Apple, etc all suggesting they are making AR glasses.

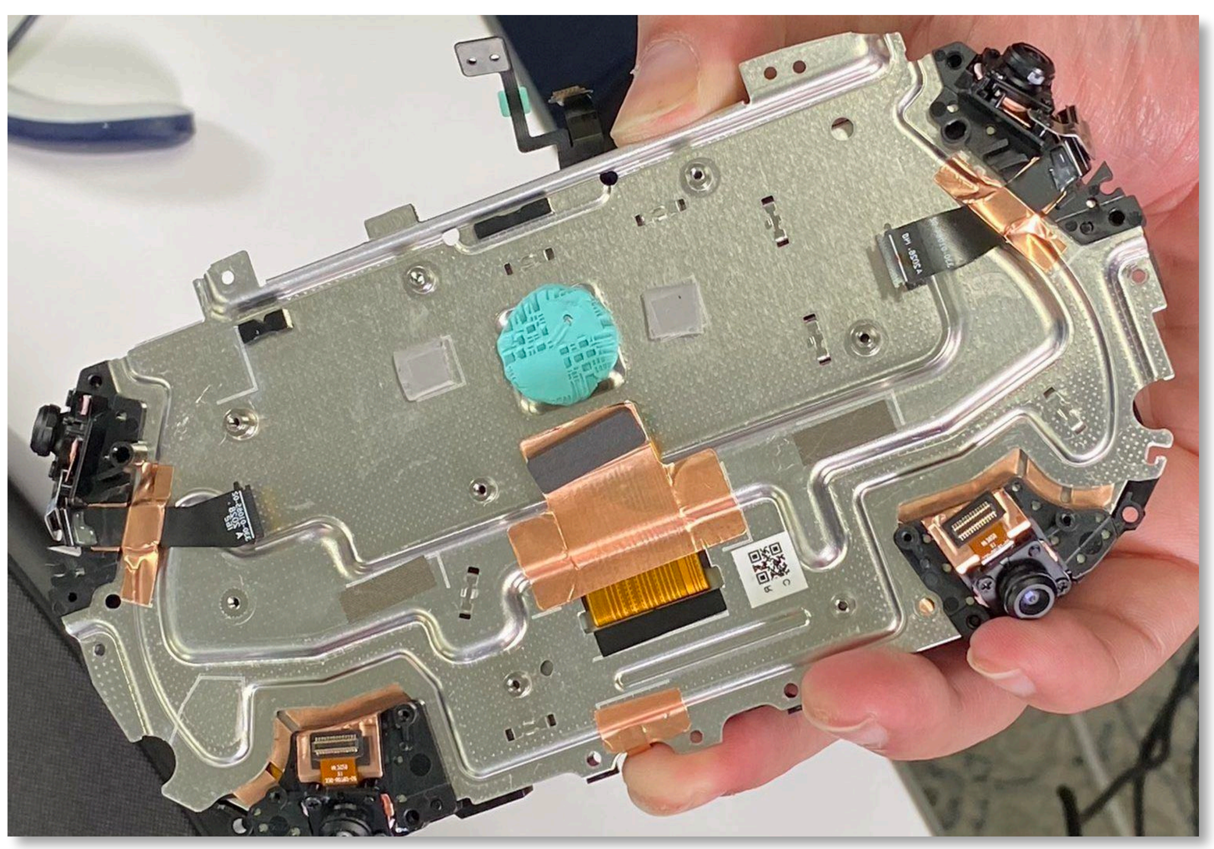
Snap Spectacles v4 (2021)

(Snap reports 30 minute battery life)

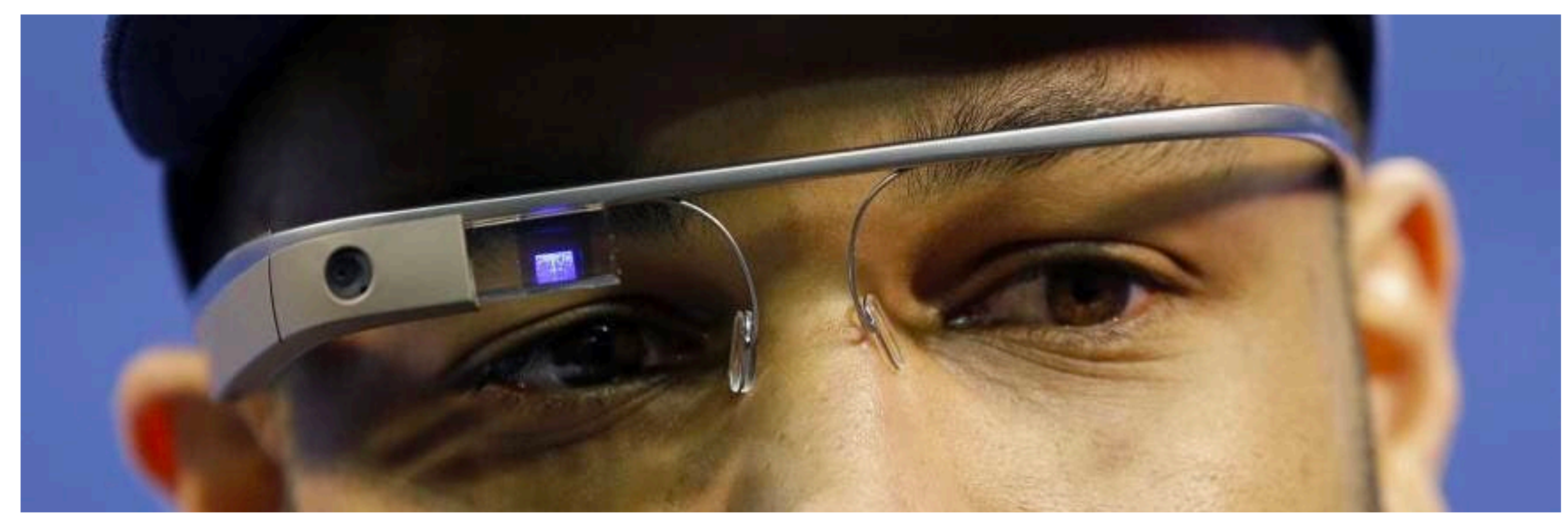


AR / VR summary

- Very difficult technical challenge
- Interest in glasses form factor will place considerably more pressure on system energy efficiency



vs.



- Many new challenges of AR:
  - Rendering to a display that "overlays" on the real world (how to draw black?)
  - Intelligently interpreting the world to know what content to put on the display
  - Ethical/privacy questions about applications