Post Processing in The Orange Box

Alex Vlachos
Valve

February 18, 2008

Outline

- sRGB – DX9, DX10, XBox 360
- Tone Mapping
- Motion Blur

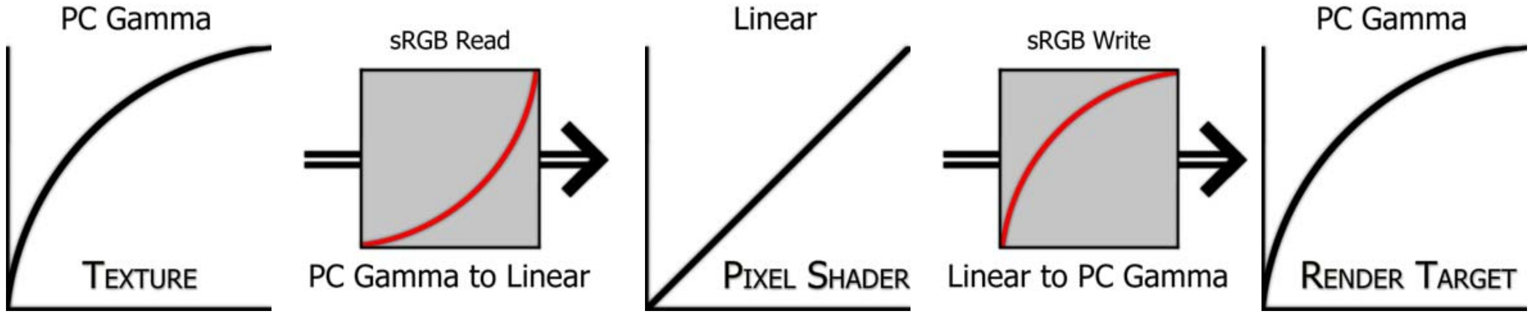
sRGB Outline

- sRGB & gamma review
- Alpha Blending: DX9 vs. DX10 & XBox 360
- sRGB curve: PC vs. XBox 360

sRGB Review

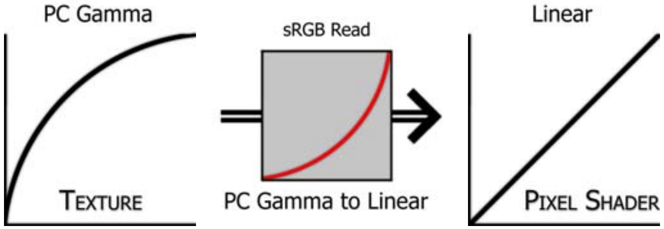
Terminology:

- Color textures are stored in “gamma space”
- Want our pixel shader to run in “linear space”

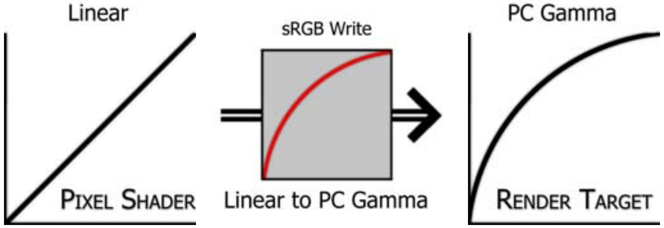


sRGB & Gamma Conversions

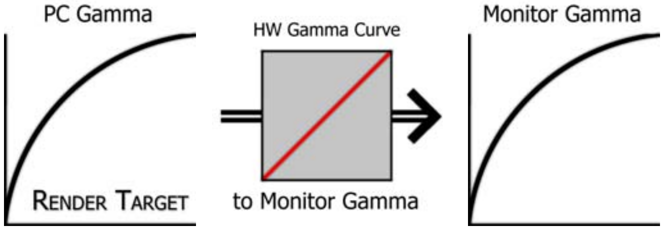
sRGB Read:



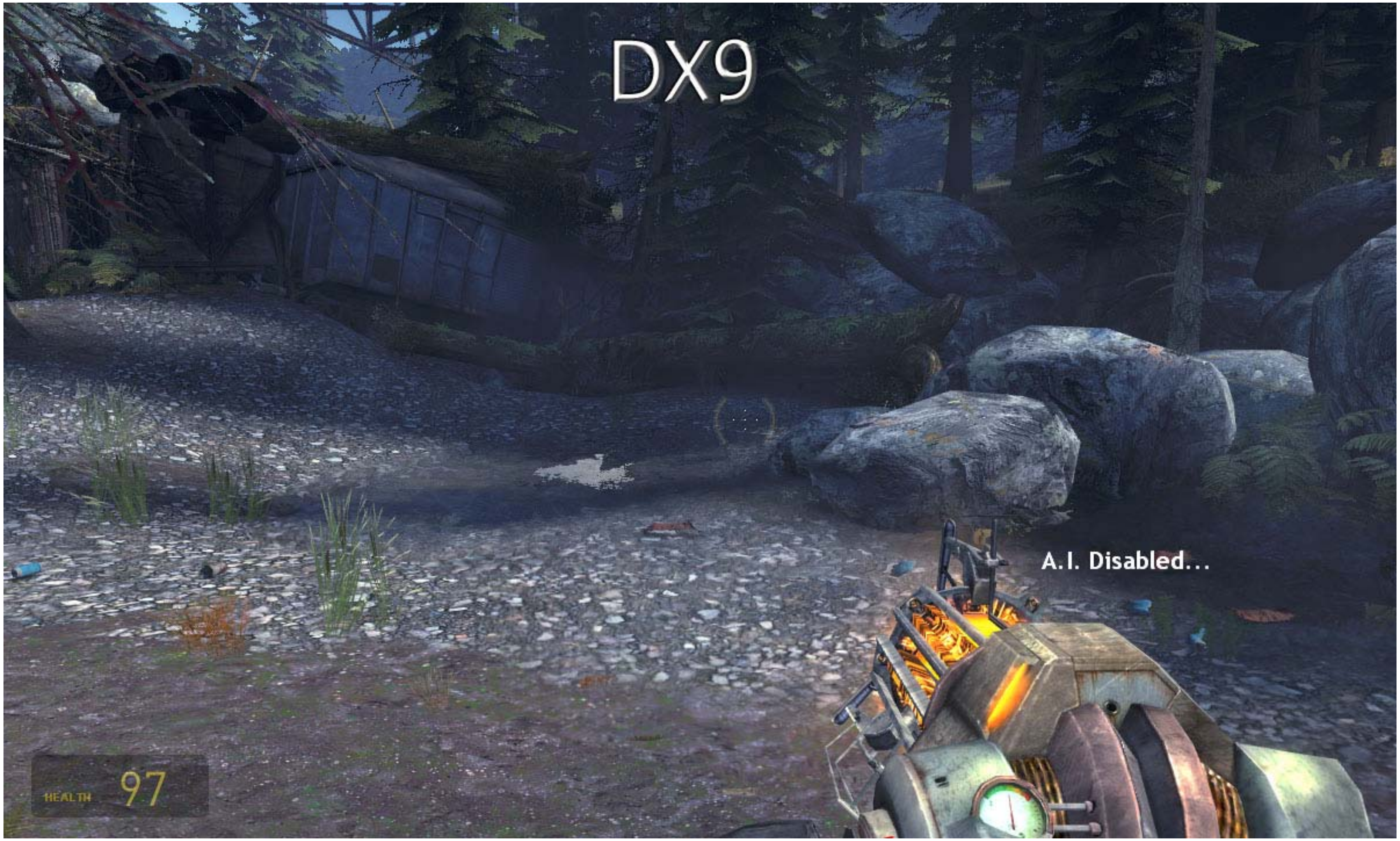
sRGB Write:



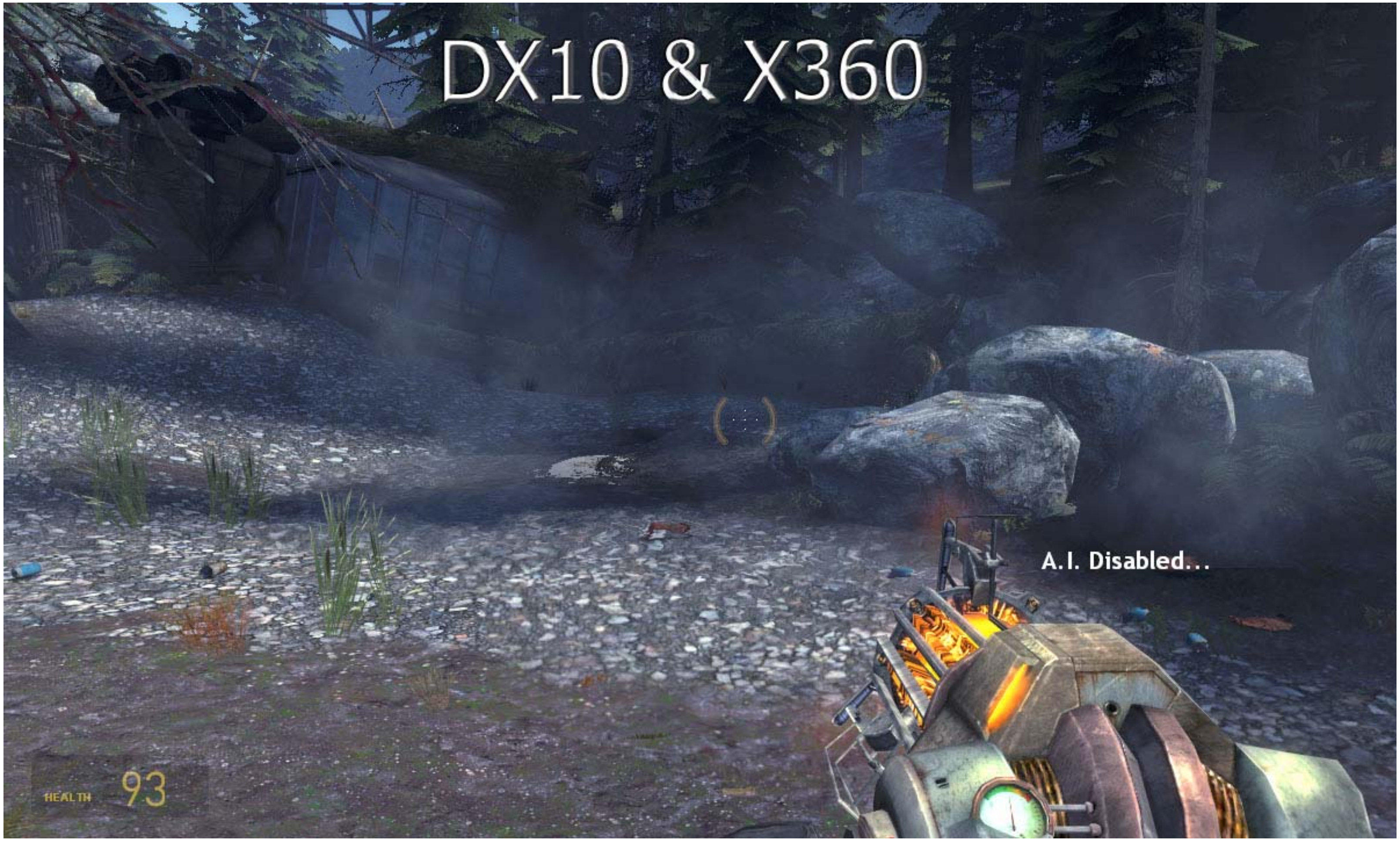
Hardware
Gamma Curve:



Alpha Blending w/ sRGB Writes DX9 vs DX10 & XBox 360



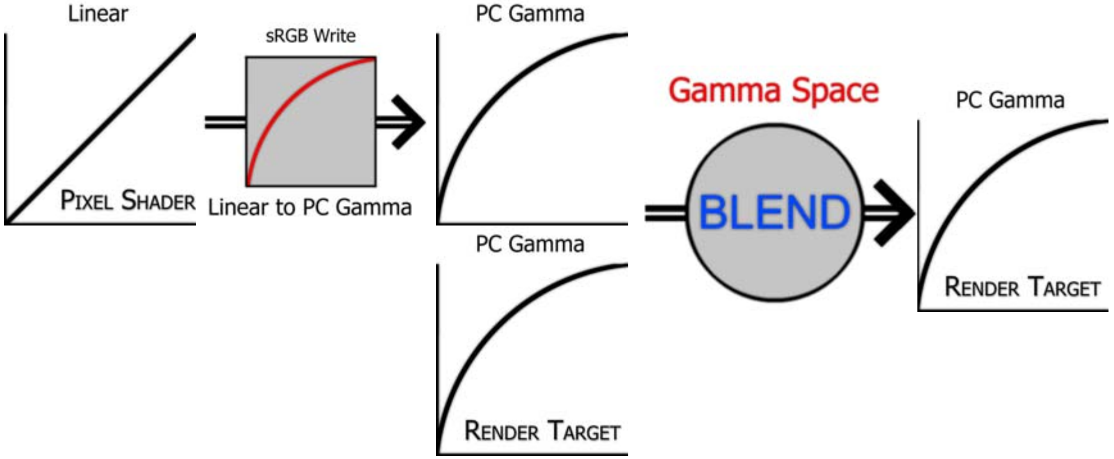
Alpha Blending w/ sRGB Writes DX9 vs DX10 & XBox 360



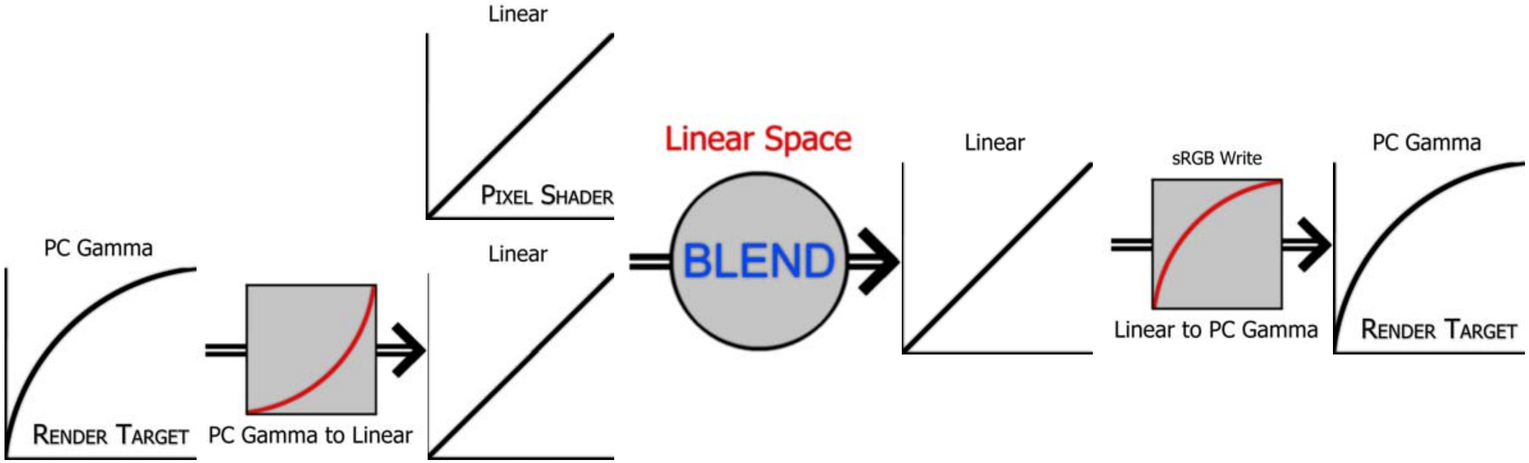
Notice the thicker smoke, glows on the gravity gun, and health GUI

Alpha Blending on DX10 HW and XBox 360

DX9 with sRGB writes



DX10 & XBox 360 with sRGB writes



Important Details

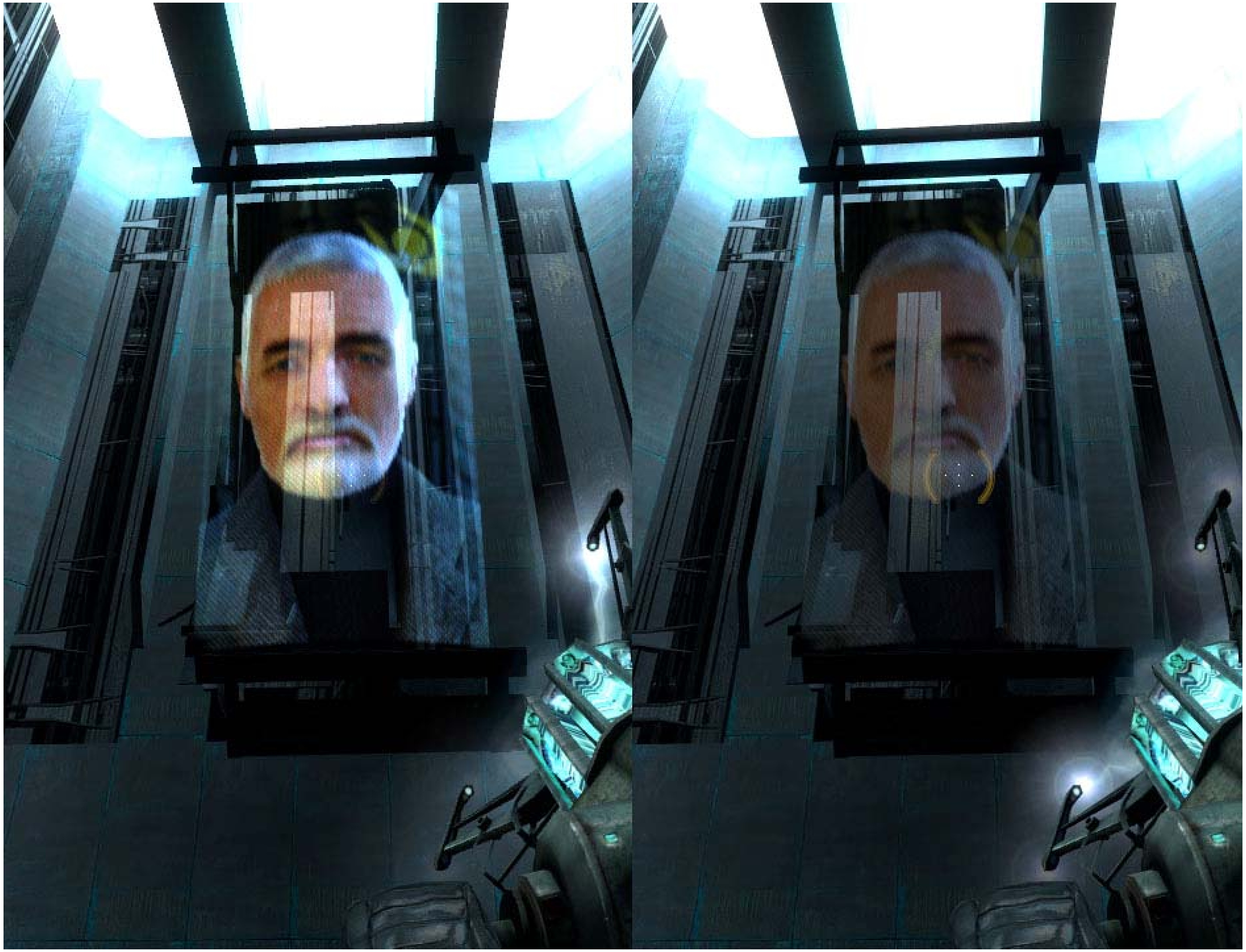
- DX10 hardware running on DX9 will blend with DX10's behavior!
- This affects DX9 games that have already shipped!

Solutions

1. Detect DX10 behavior and simulate the sRGB write in shader code forcing gamma blending
2. Let your artists tweak your materials for the obvious cases

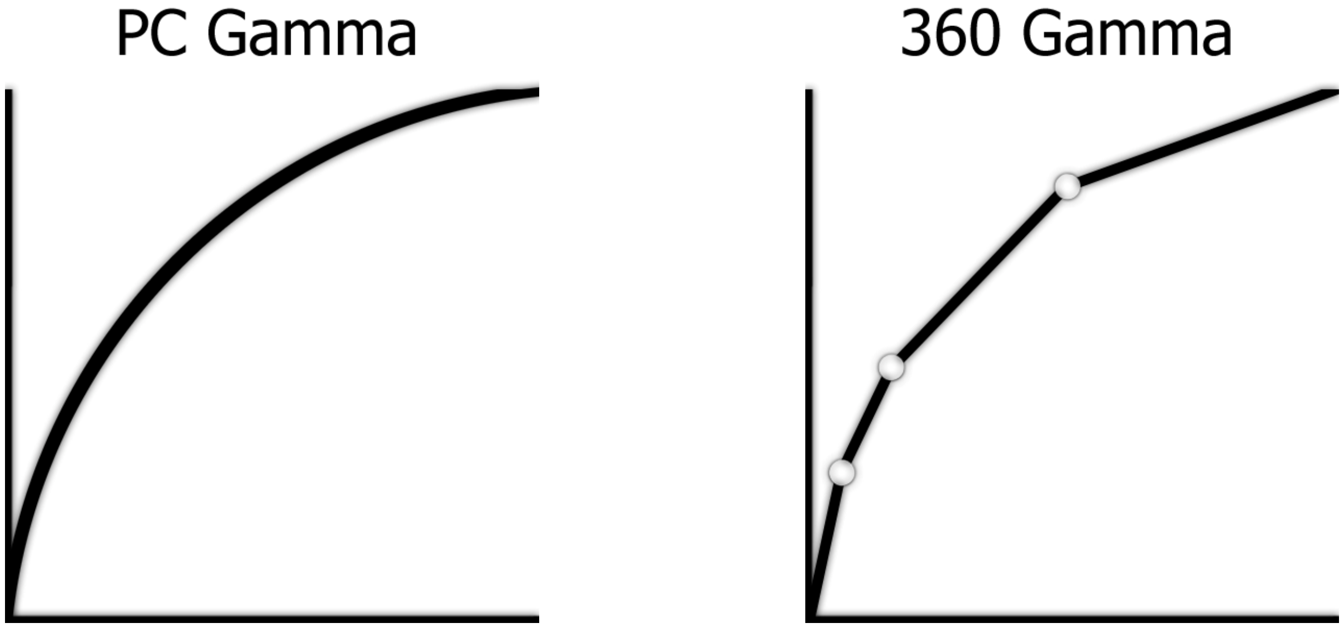
We chose #2, and the artists only modified 40 out of thousands of materials in HL2, Ep1, Ep2, TF2, and Portal

Example From Half-Life 2



sRGB Curve: PC vs. XBox 360

- PC hardware uses the actual sRGB curve
- XBox 360 uses a piecewise linear approximation to the sRGB curve

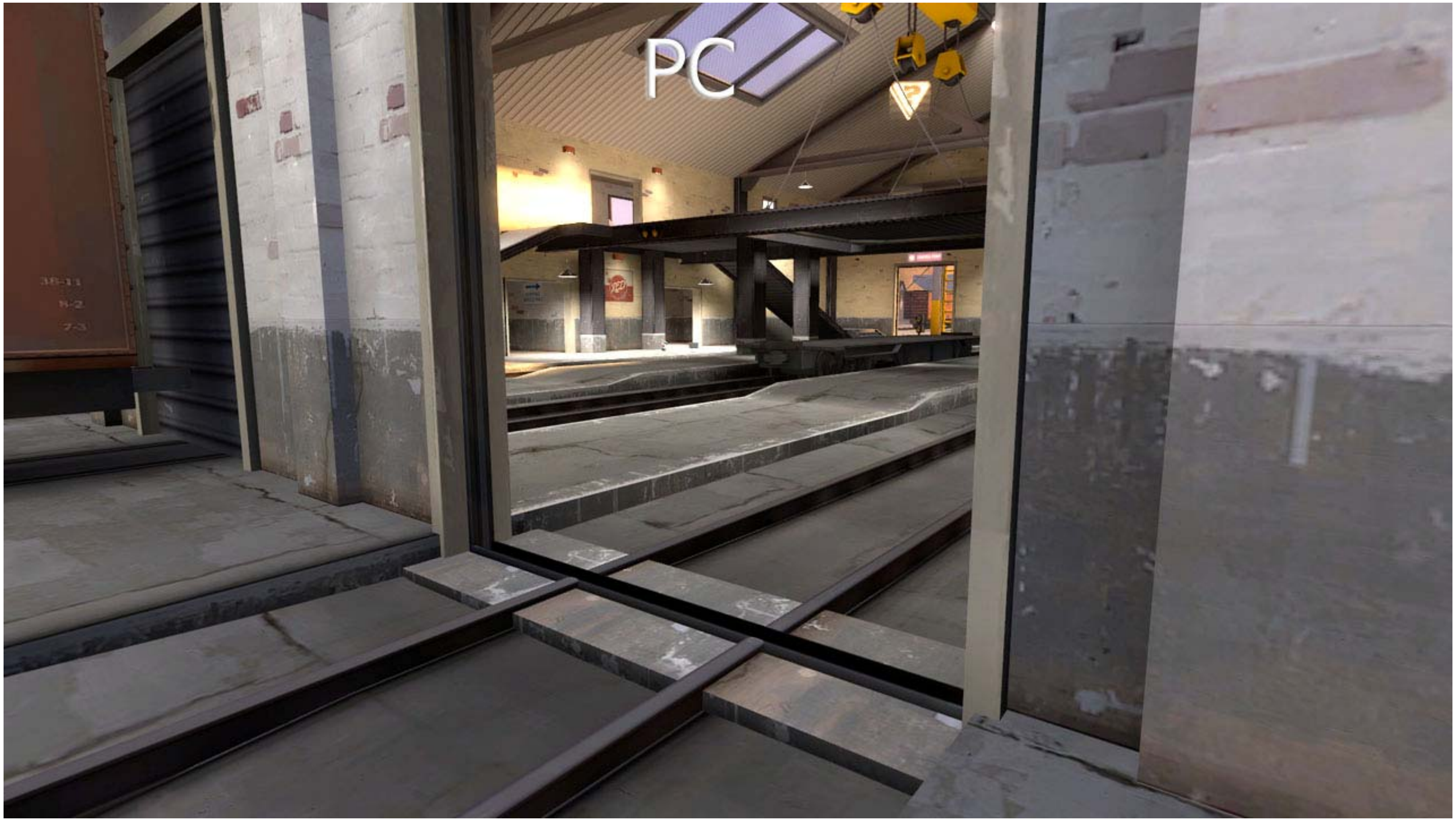


NOTE: The smooth curve representing the PC sRGB curve in these slides doesn't accurately represent the actual sRGB curve that is linear in the low end.

Different Gamma Spaces

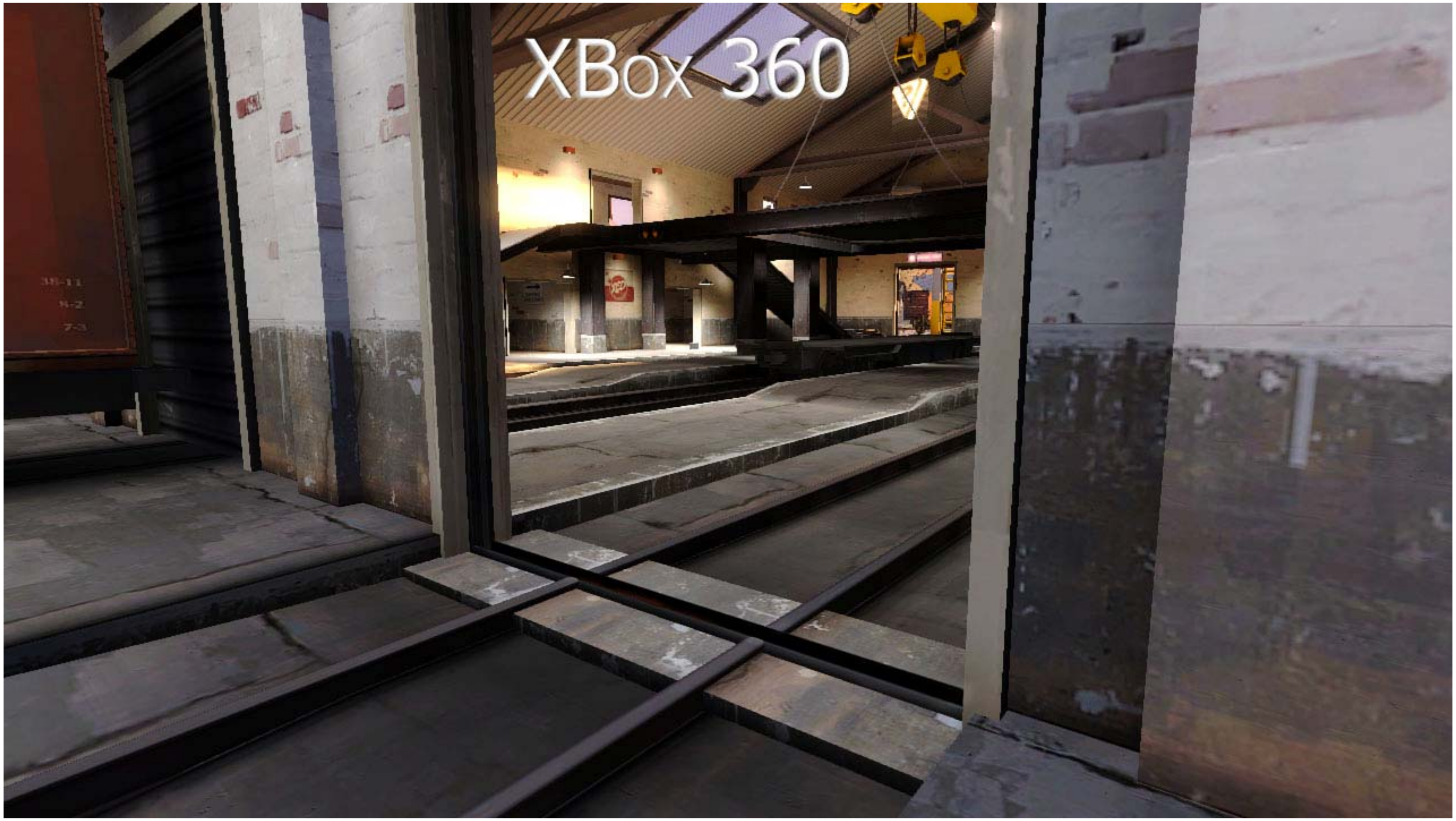
- We stopped using the term “Gamma Space” and instead…
  - “**PC Gamma Space**” – Official sRGB curve
  - “**360 Gamma Space**” – Piecewise linear sRGB approximation found on the XBox 360
  - “**Linear Space**”

PC Gamma on PC



(Using PC Gamma textures on the PC)

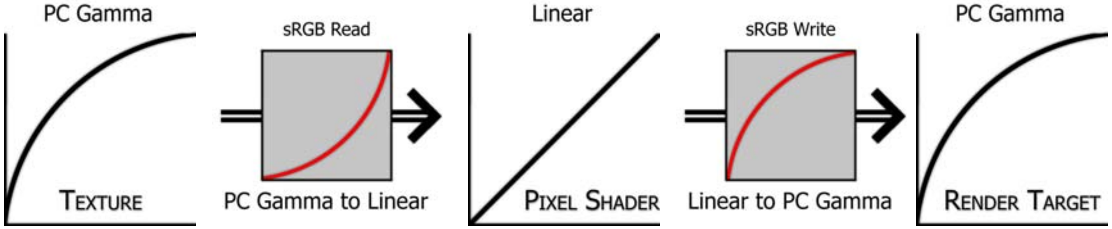
Uncorrected 360 Results



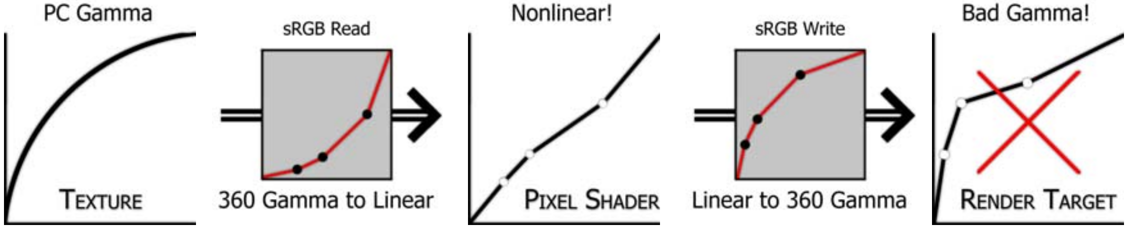
(Using PC Gamma textures on the XBox 360)

What Just Happened?

On PC, linear in pixel shader:



On 360, nonlinear in pixel shader:



Solutions

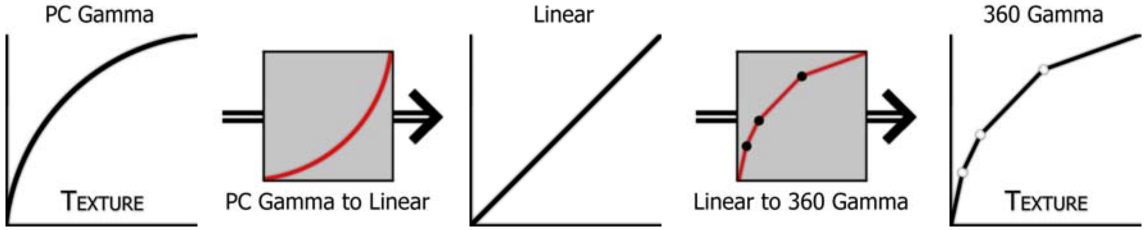
1. XBox 360-only developers: Use a Photoshop color space plug-in
2. Simulate sRGB reads and writes in shader code (Performance!)
3. Convert color textures at tool time and use the hardware gamma curve

Orange Box uses #3. Let’s take a closer look…

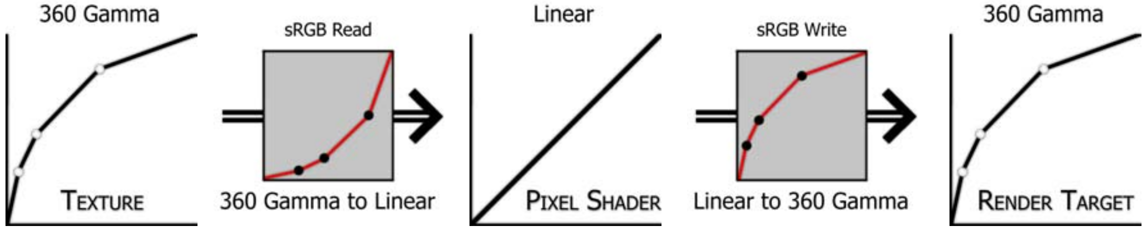
The Orange Box Solution for XBox 360

- Want to use the hardware “sRGB” reads & writes
- We can modify the input textures so that the 360’s piecewise linear read gets us to linear space!

Preprocess:

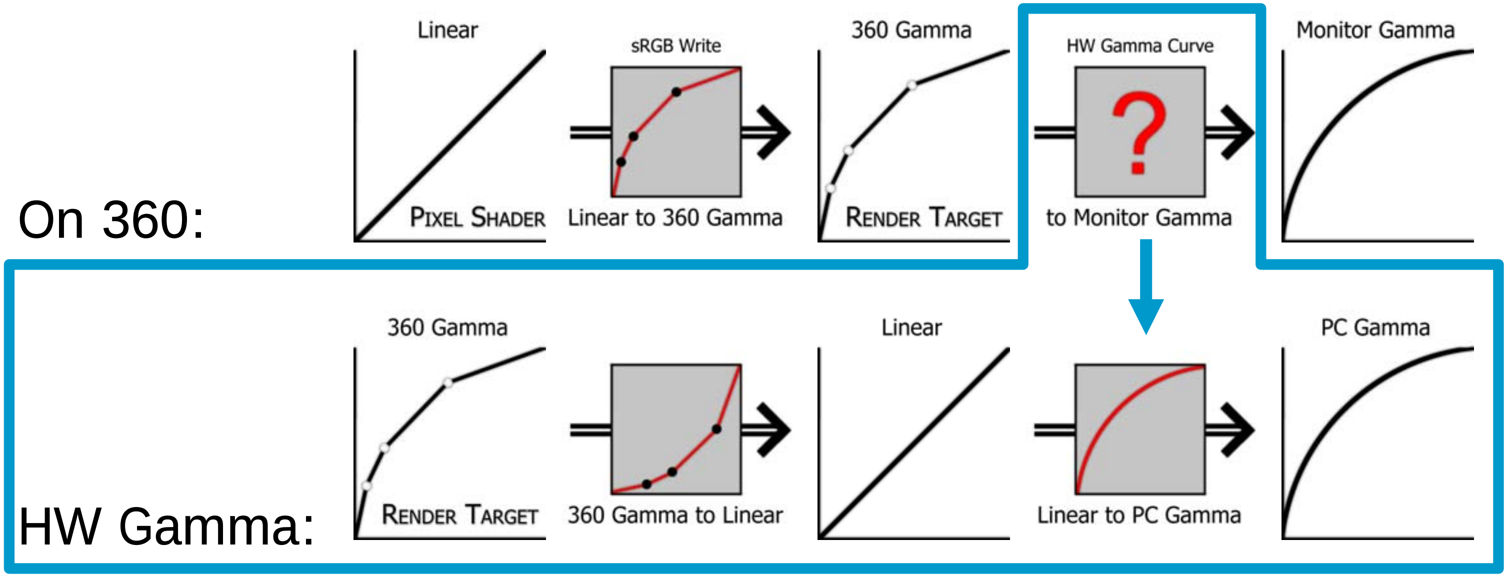


Run-time:



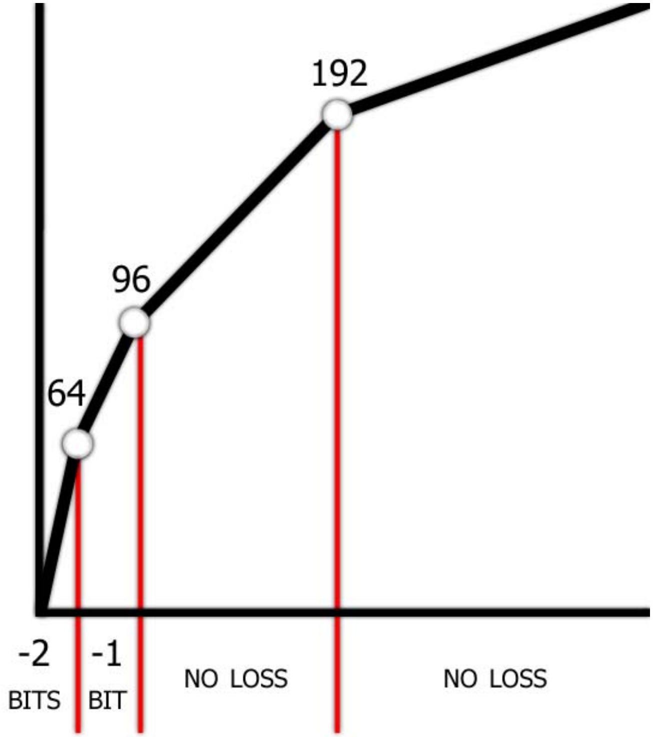
- But, 360 gamma space looks wrong on a TV or monitor! Need to get back to PC gamma space…

Using the Hardware Gamma Curve



We also use the hardware gamma curve to optionally correct for “blacker-than-black” colors displayed on a television and the deeper gamma of televisions.

XBox 360 Lossy sRGB Read



- In linear space, the lossy range is 0.0-0.14, so generally OK. This caused very few issues for us, but…
- Don’t use the hardware sRGB reads for post processing or feedback effects! Simulate the piecewise linear sRGB read in shader code…it’s only ~11 asm instructions.

sRGB Summary

- Alpha blending differences exist
  - We let the artists tweak around this
- XBox 360 has a different gamma space
  - Convert color textures PC -> 360 Gamma Space
  - Set hardware gamma ramp for end correction
- XBox 360 HW sRGB read is lossy at the dark end (in linear space, 0.0-0.14)

Tone Mapping Outline

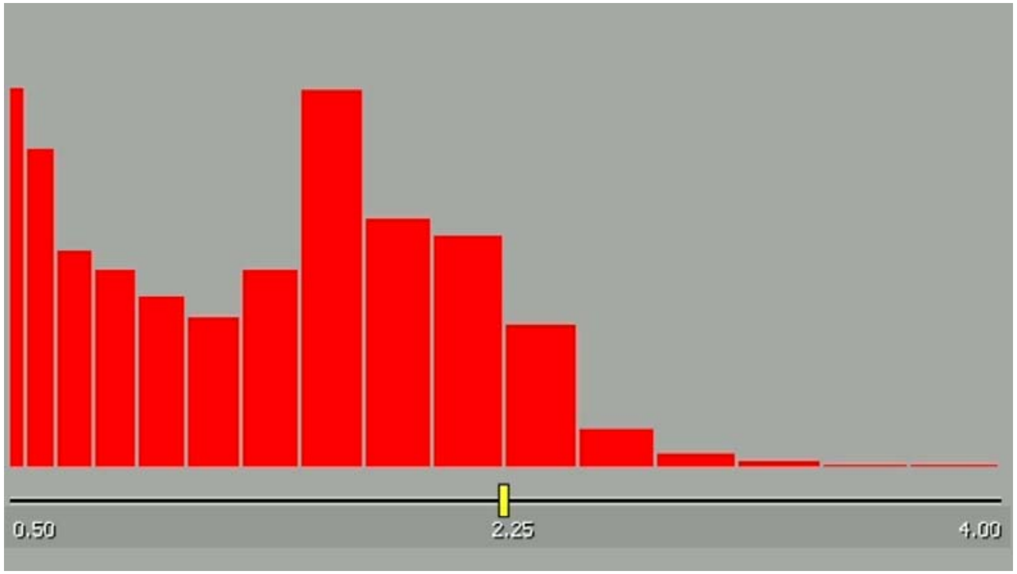
- Brief overview of Valve’s HDR rendering
- Building the histogram
- Computing the tonemap scalar

Overview of Valve's HDR Rendering

- Lighting data and environment maps are stored in HDR linear space
- Every pixel shader scales the linear HDR value by our tonemap scalar (Back buffer is RGBA8888!)
- Incrementally build histogram each frame
- Tonemap scalar is generated from the current histogram each frame

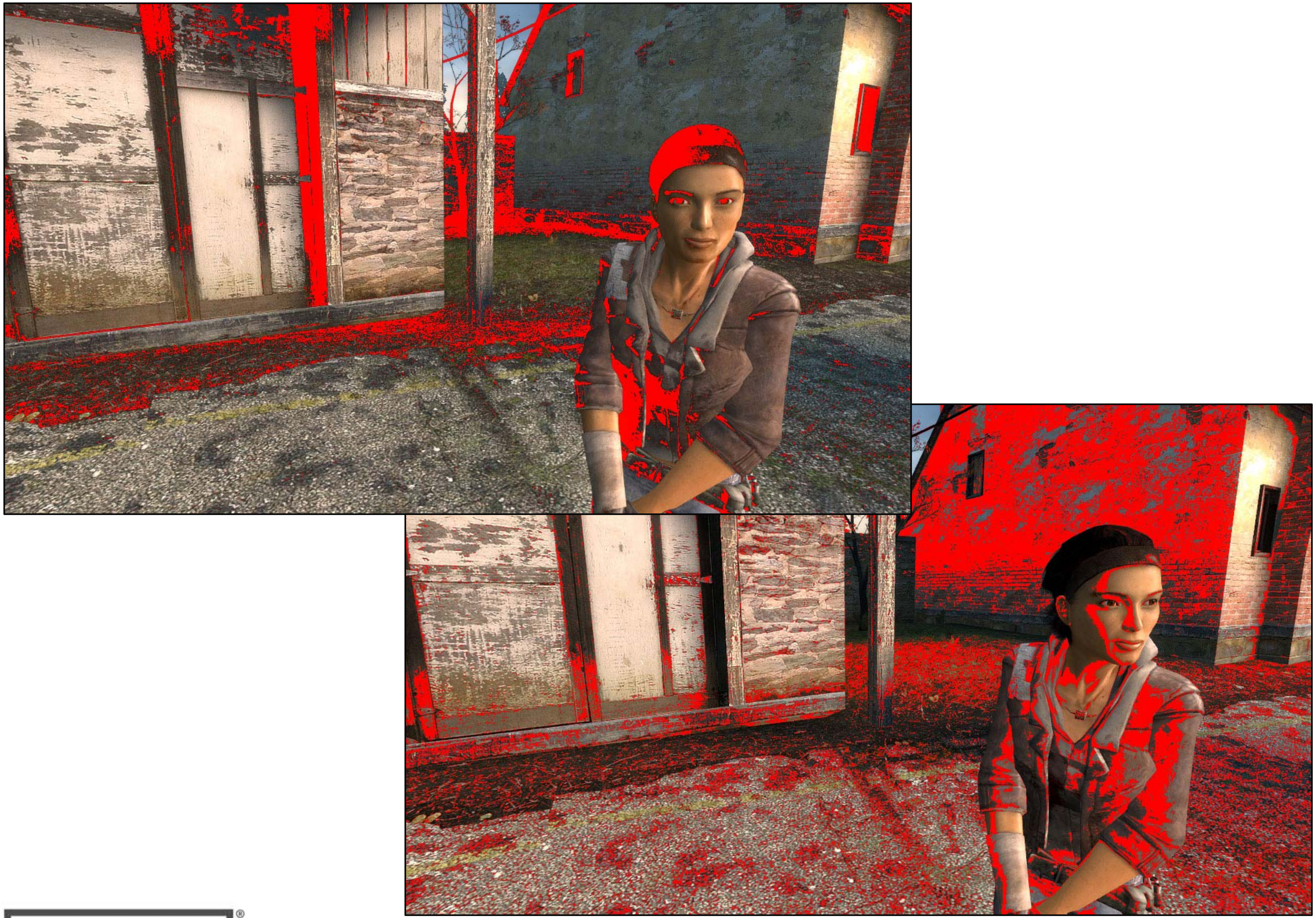
- More details on the first 2 points: http://www.valvesoftware.com/publications.html

Building the Histogram



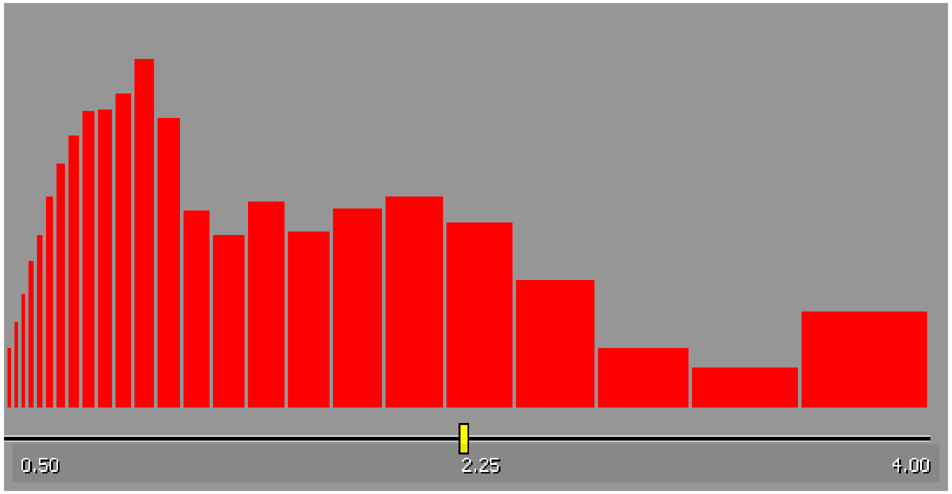
- Amortize the cost of building the histogram over 16 frames
  - Update one bucket per frame
  - Sample post-tonemapped frame
- Use an asynchronous occlusion query to count pixels in range

Sampling Each Histogram Bucket



Evaluating the Histogram

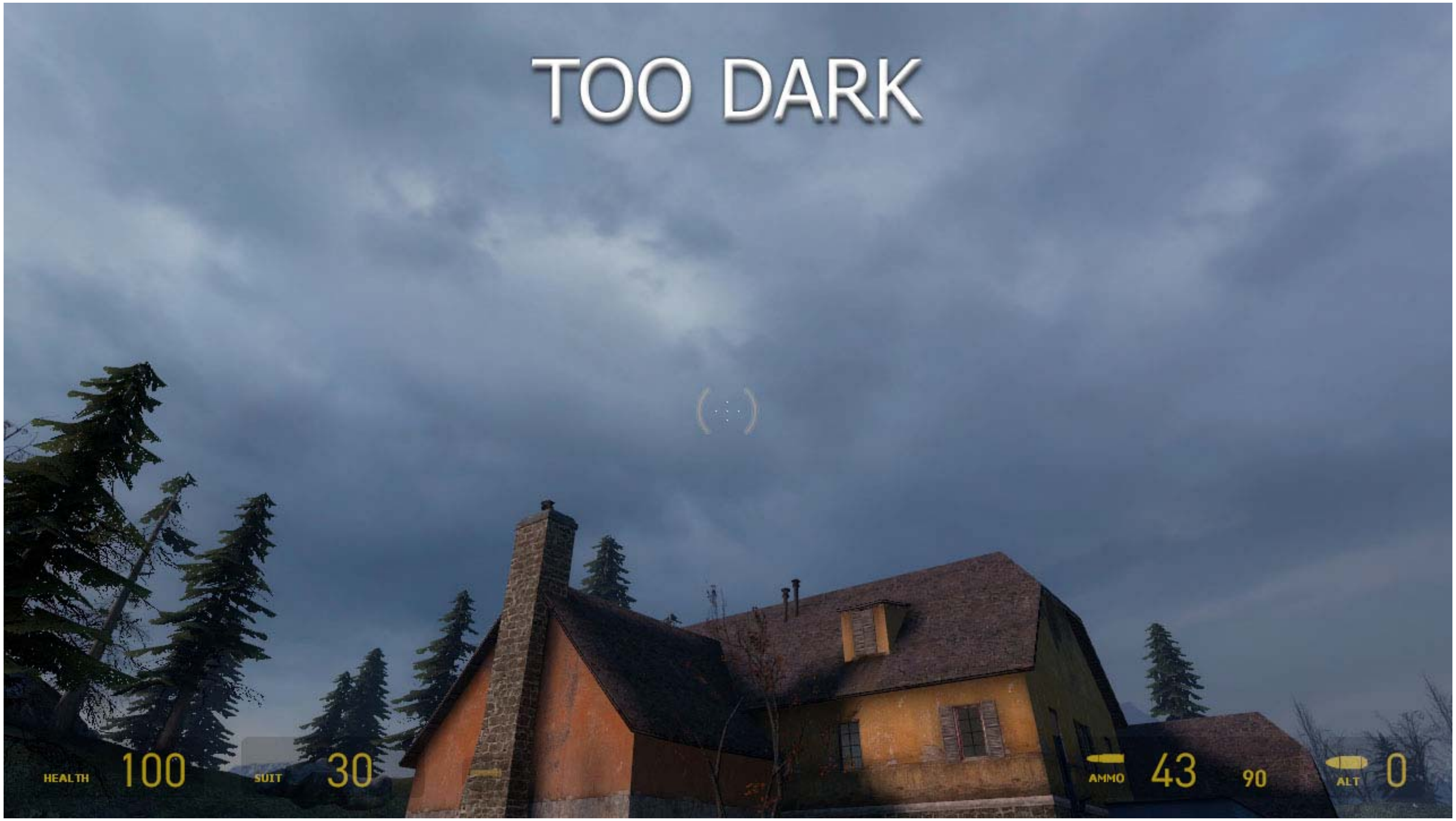
- Our first implementation was based on median luminance (Shipped in HL2: Episode One, Day of Defeat, Lost Coast)



(NOTE: All histograms are in linear space!)

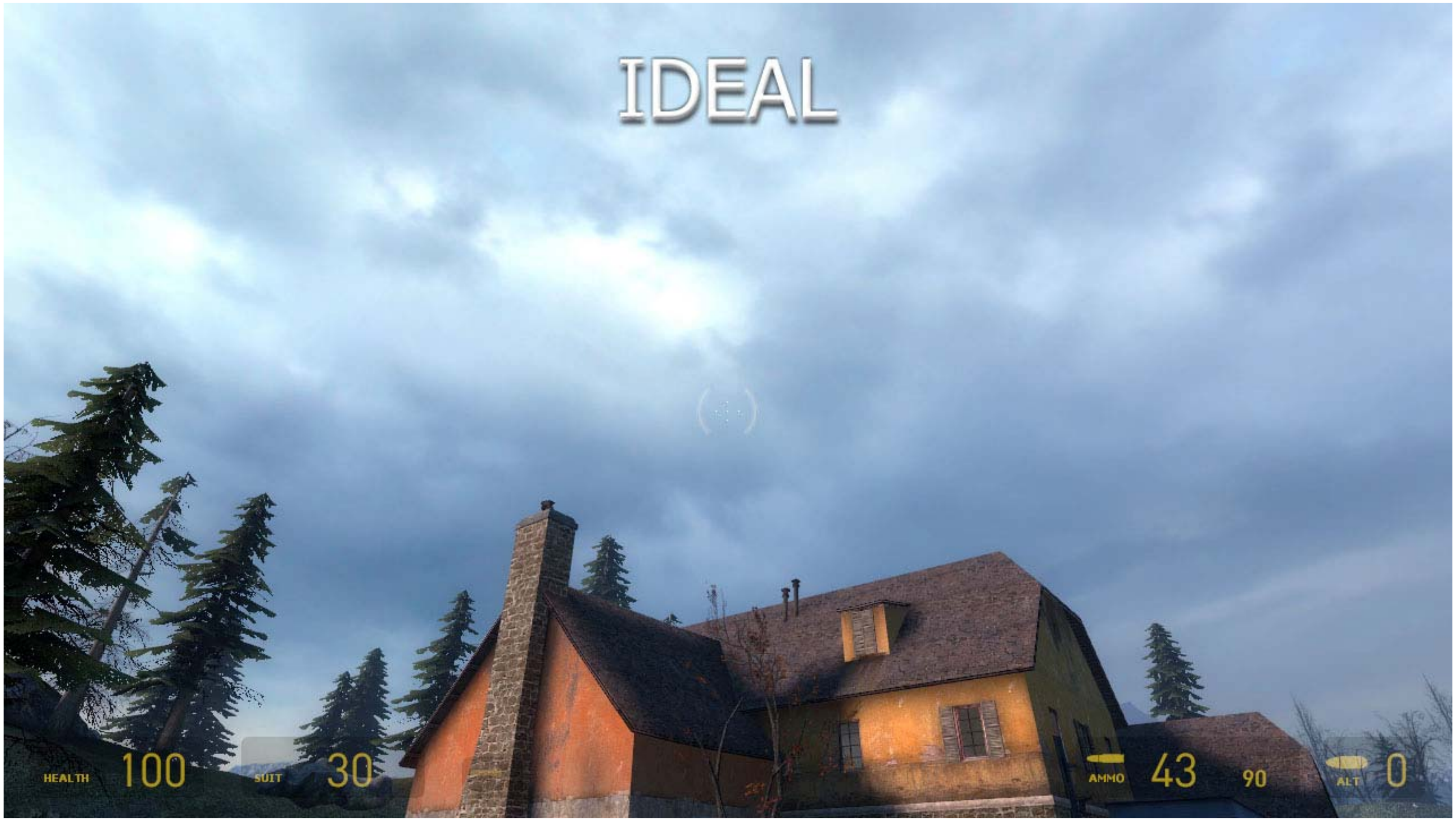
- But, we ran into too many cases in The Orange Box that caused tonemapping to behave strangely.

Dark Skies!



This environment was tonemapping too dark

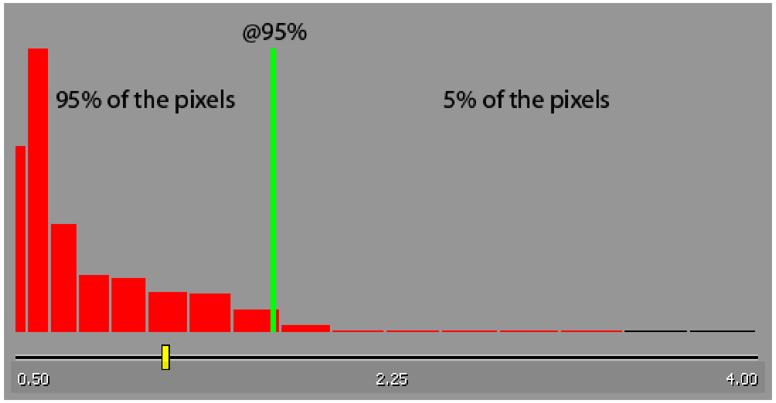
Dark Skies!

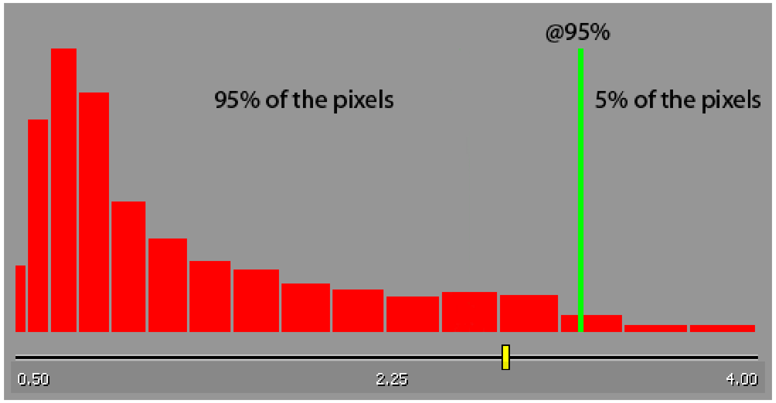


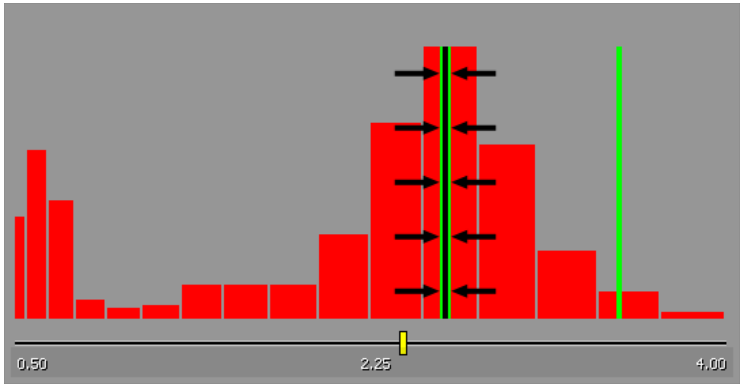
This is what we wanted

Bright Pixels Matter

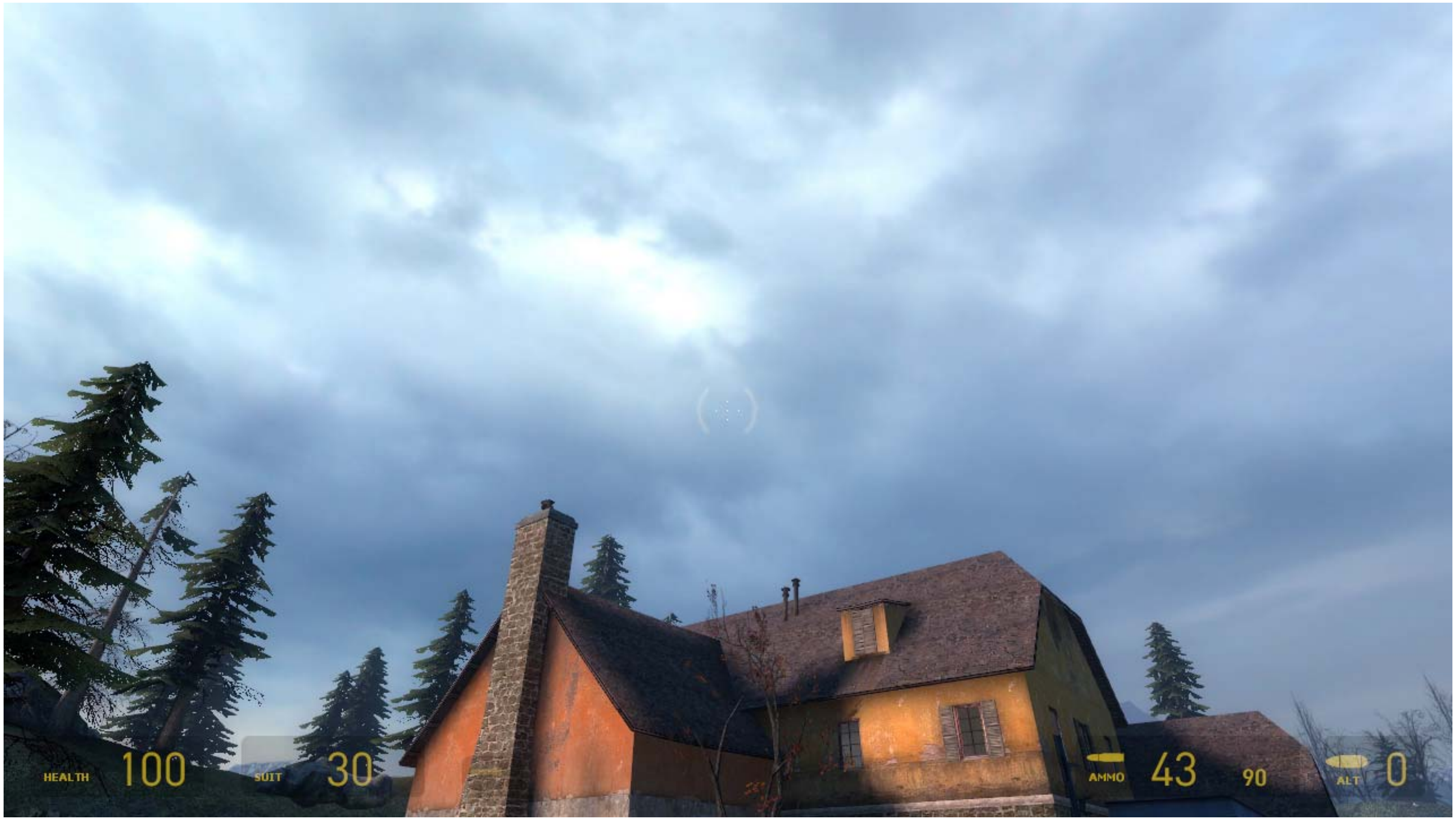
- Experiment: Don’t use median luminance (50%)
- Use a different histogram threshold: Keep 5% of bright pixels in top bins





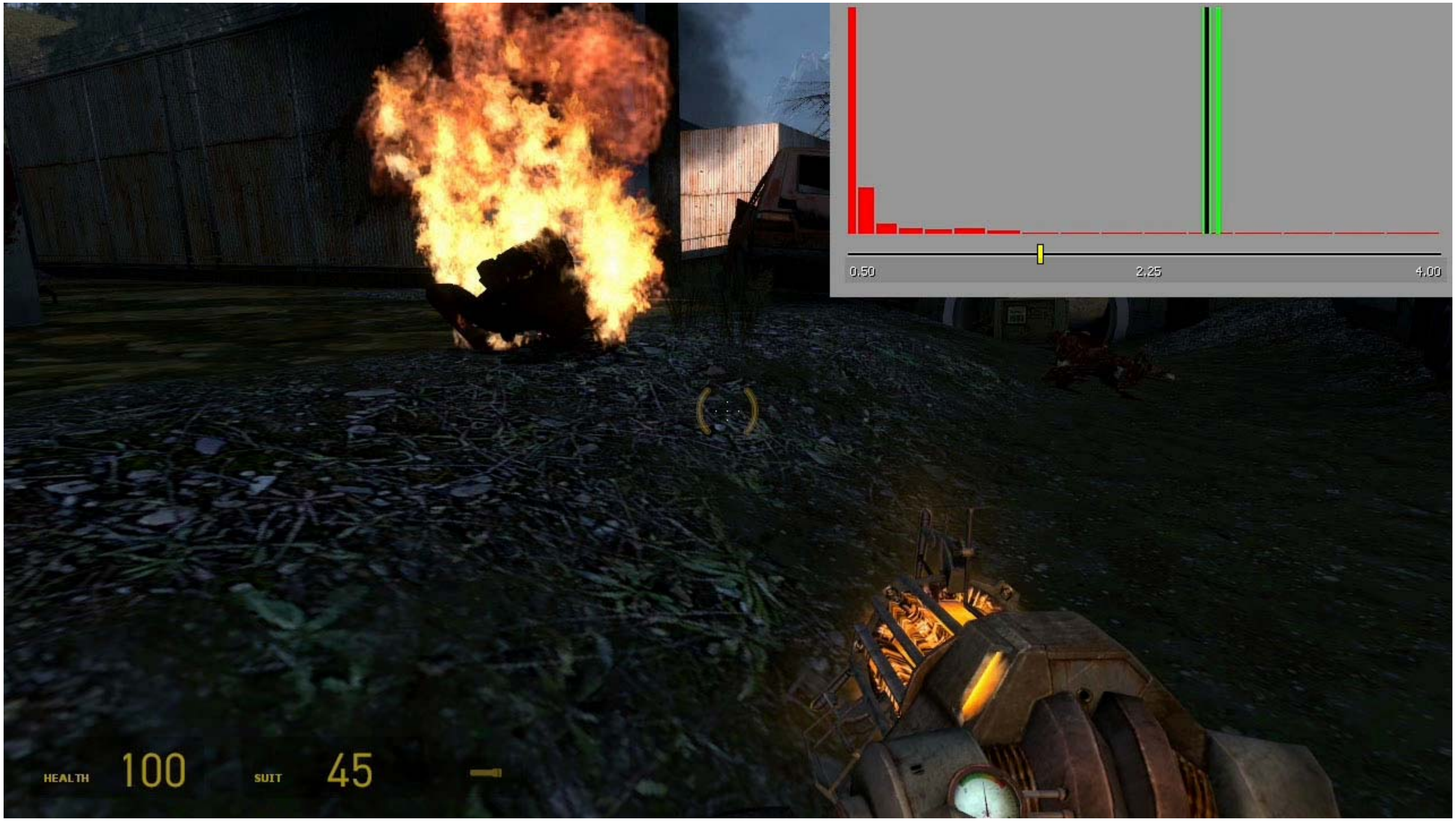


Results From Using 95% Threshold



This worked great! Except for…

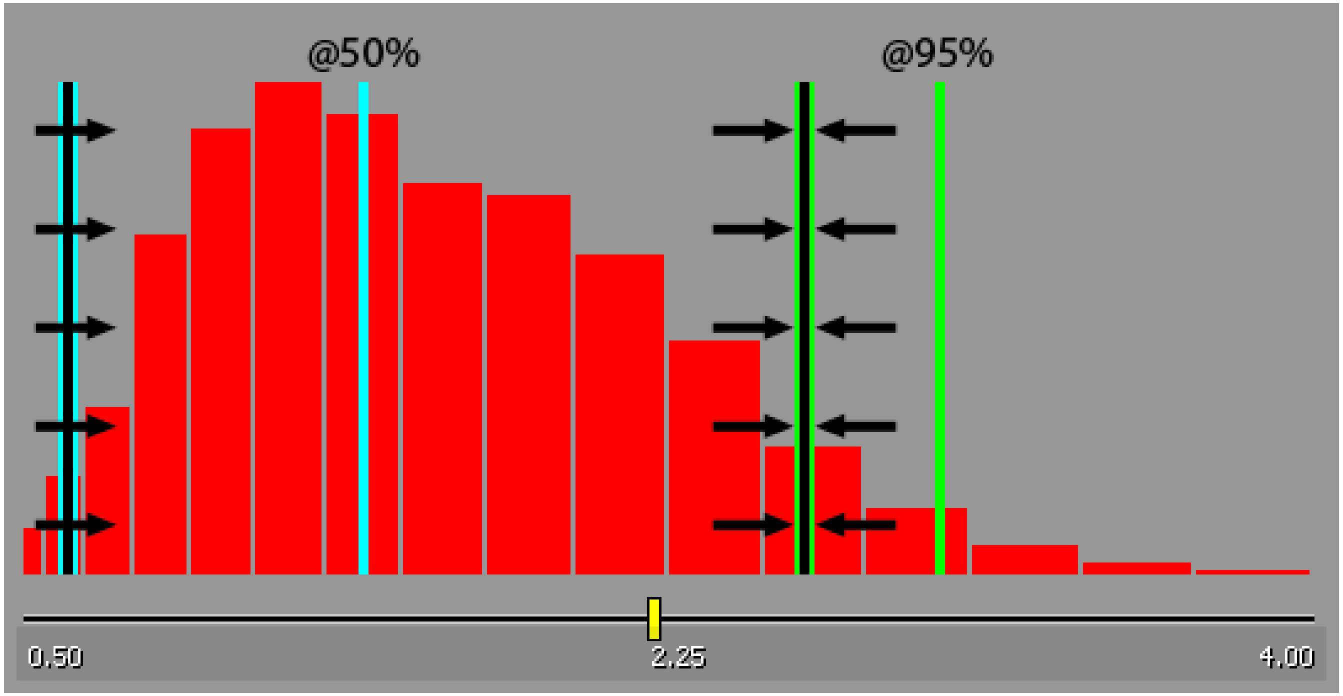
Zombies on Fire!



Bright pixels from the fire caused
tone mapping to over darken the screen!

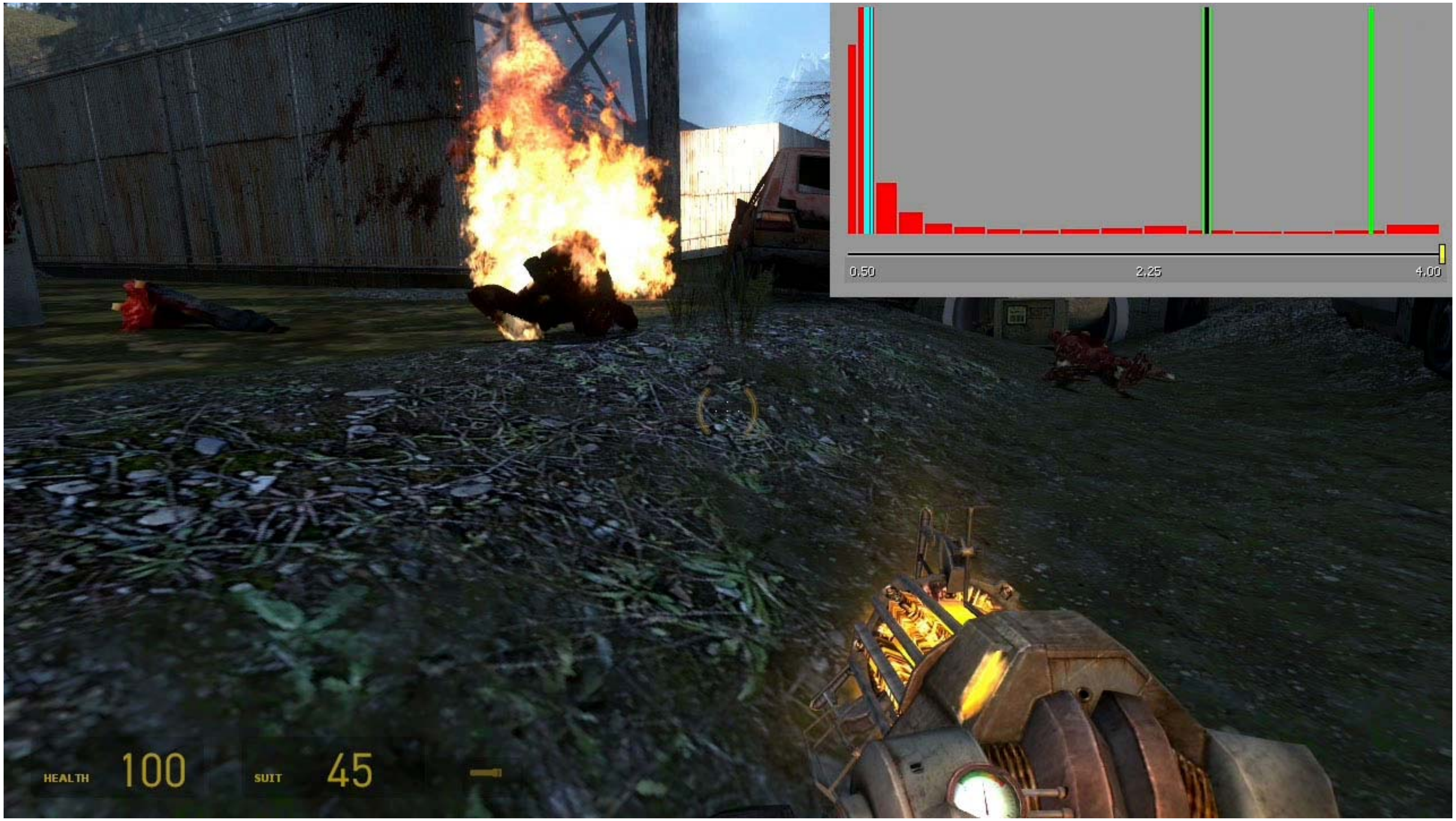
Need a Secondary Rule

- Primary rule: Brightness threshold
- Secondary rule: Use median luminance as a darkness barrier



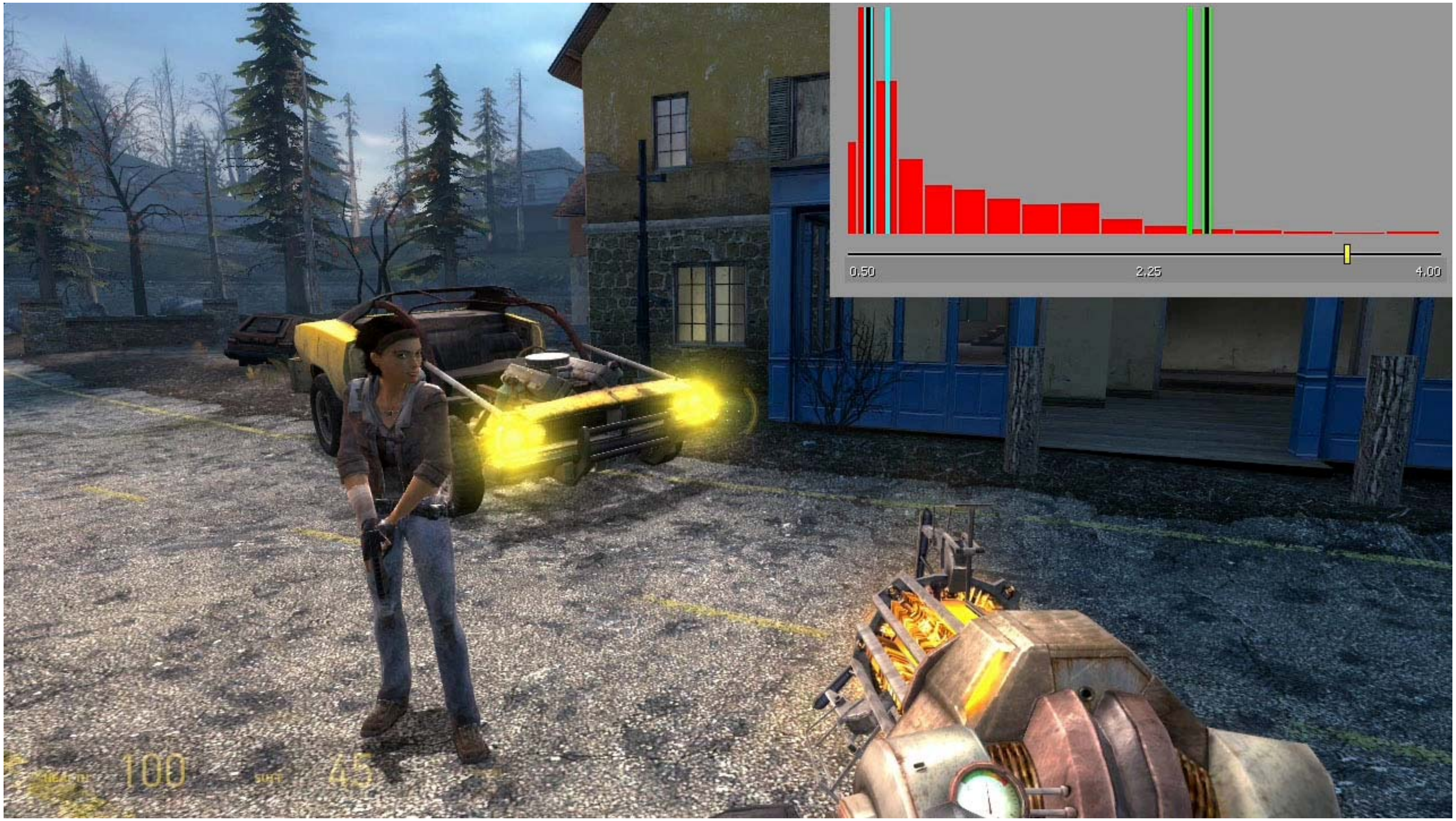
(NOTE: All histograms are in linear space!)

Zombies Fixed With Both Rules



This worked! But we still had one issue…

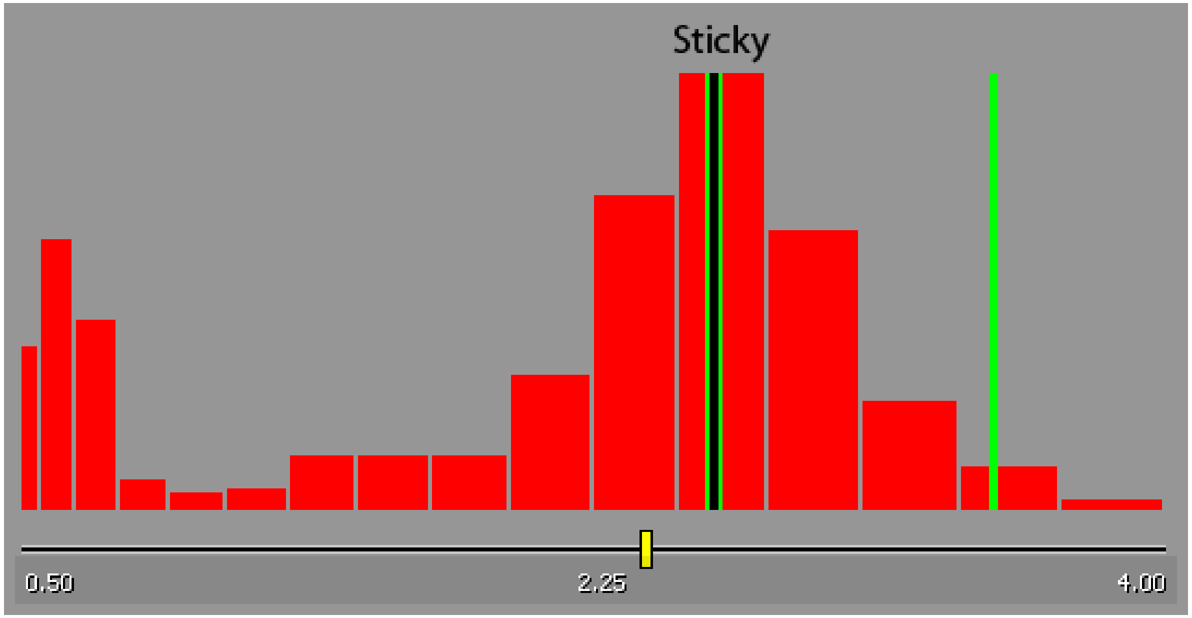
Oscillations From Blinking Lights



Blinking lights cause oscillations in the histogram that cause unwanted oscillations in the final tonemap scalar!

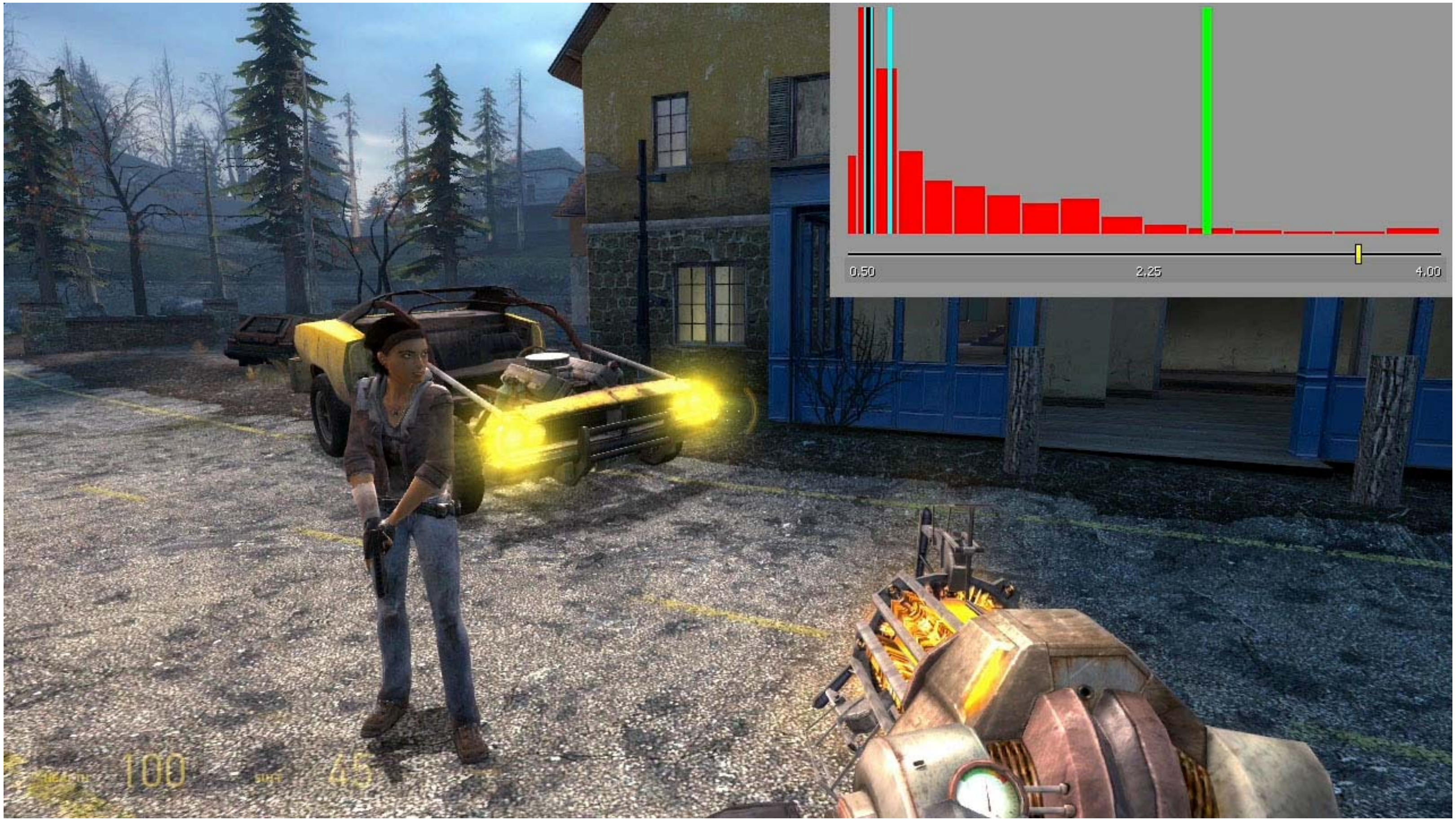
The “Sticky Bin”

- Make bin containing 95% target “sticky”



- This causes minor variations in light to have no effect until passing threshold

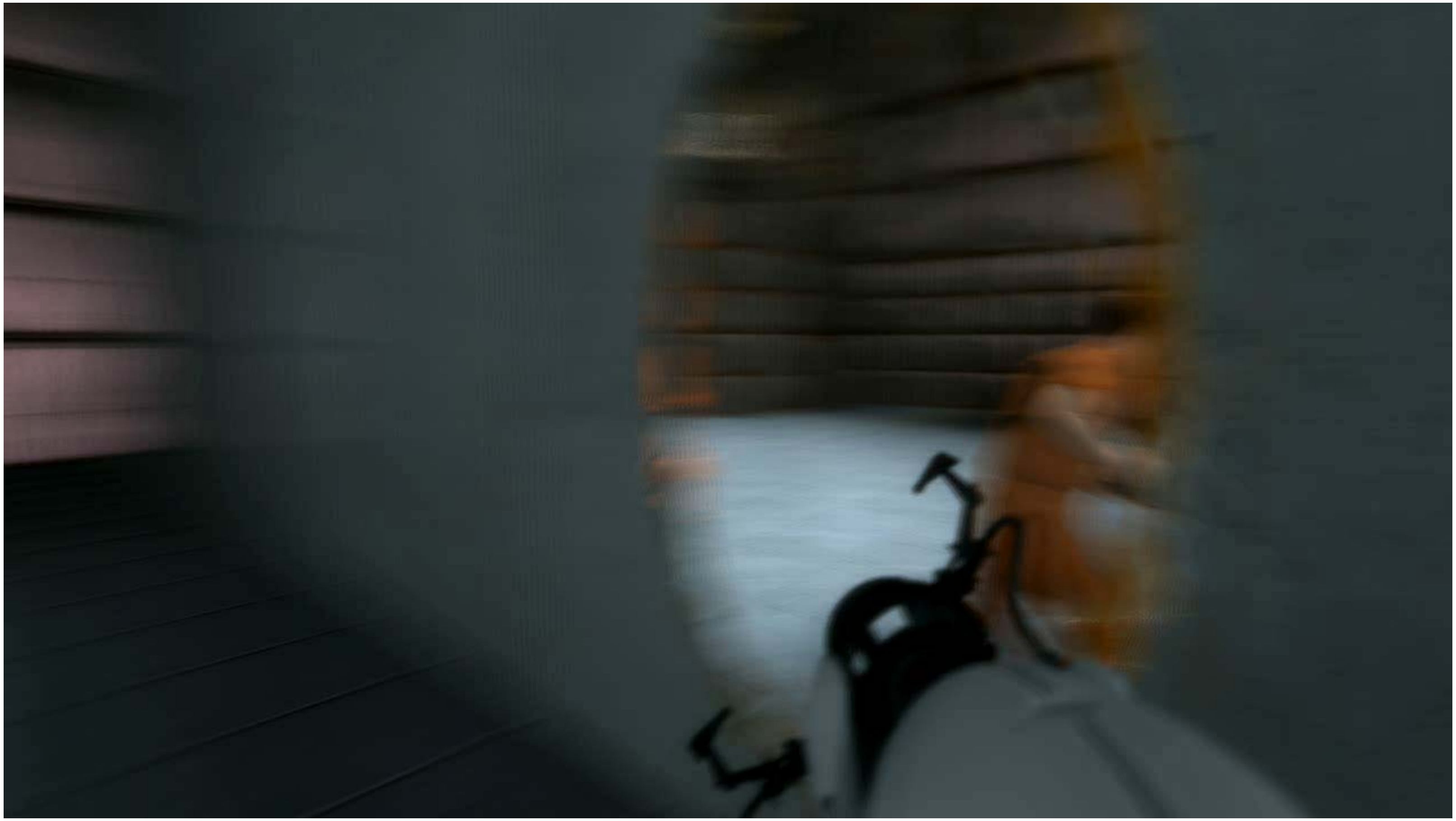
“Sticky Bin” Fixes Oscillations



Final Tonemapping Heuristics

1. Bright pixel threshold
2. Median luminance (darkness barrier)
3. Sticky bin

Motion Blur



(A section from the non-real-time Portal trailer 2006)

Motion Blur Goals

- Isolated, self-sufficient system
- Shader models 2.0, 2.0b, 3.0
- No additional memory (system or video)
- Performance!
- I don’t want to spend more than one week

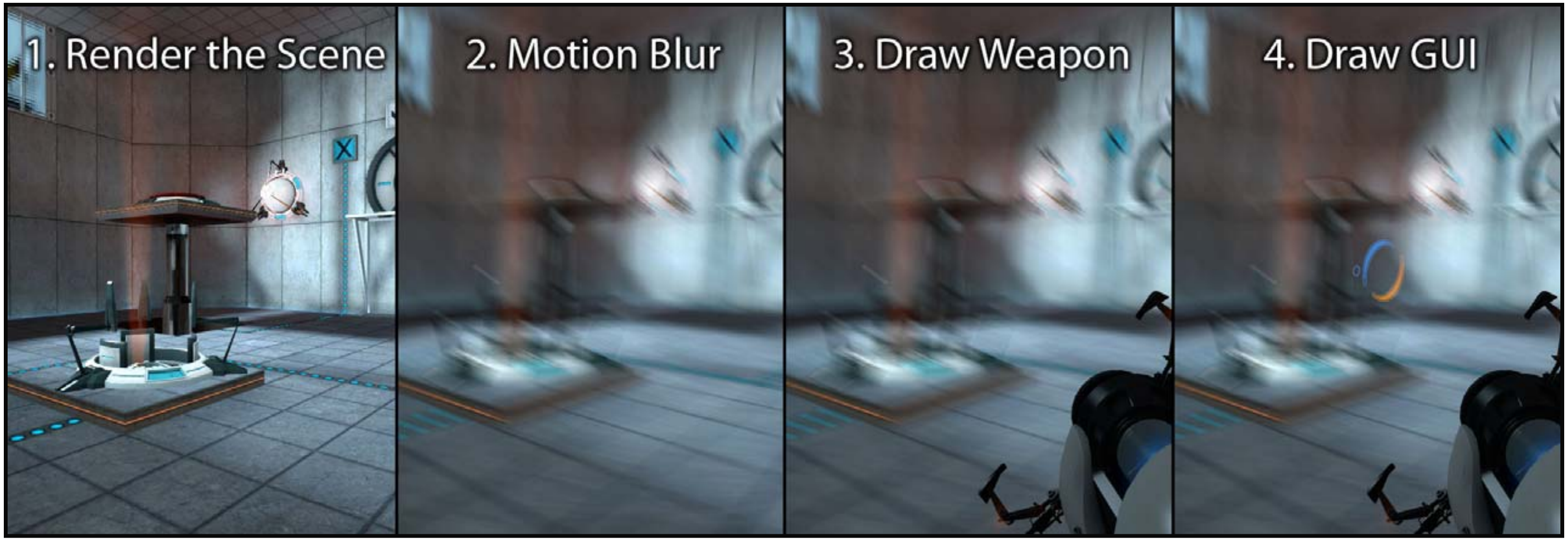
Evaluating Types of Motion Blur

1. Camera rotations – Can be done in post
2. Camera translations – Needs depth or vector image for correct parallax
3. Object translations – Needs vector image or “fins”
4. Object rotations & animation – Needs vector image or “fins”

- We chose #1 with some of #2

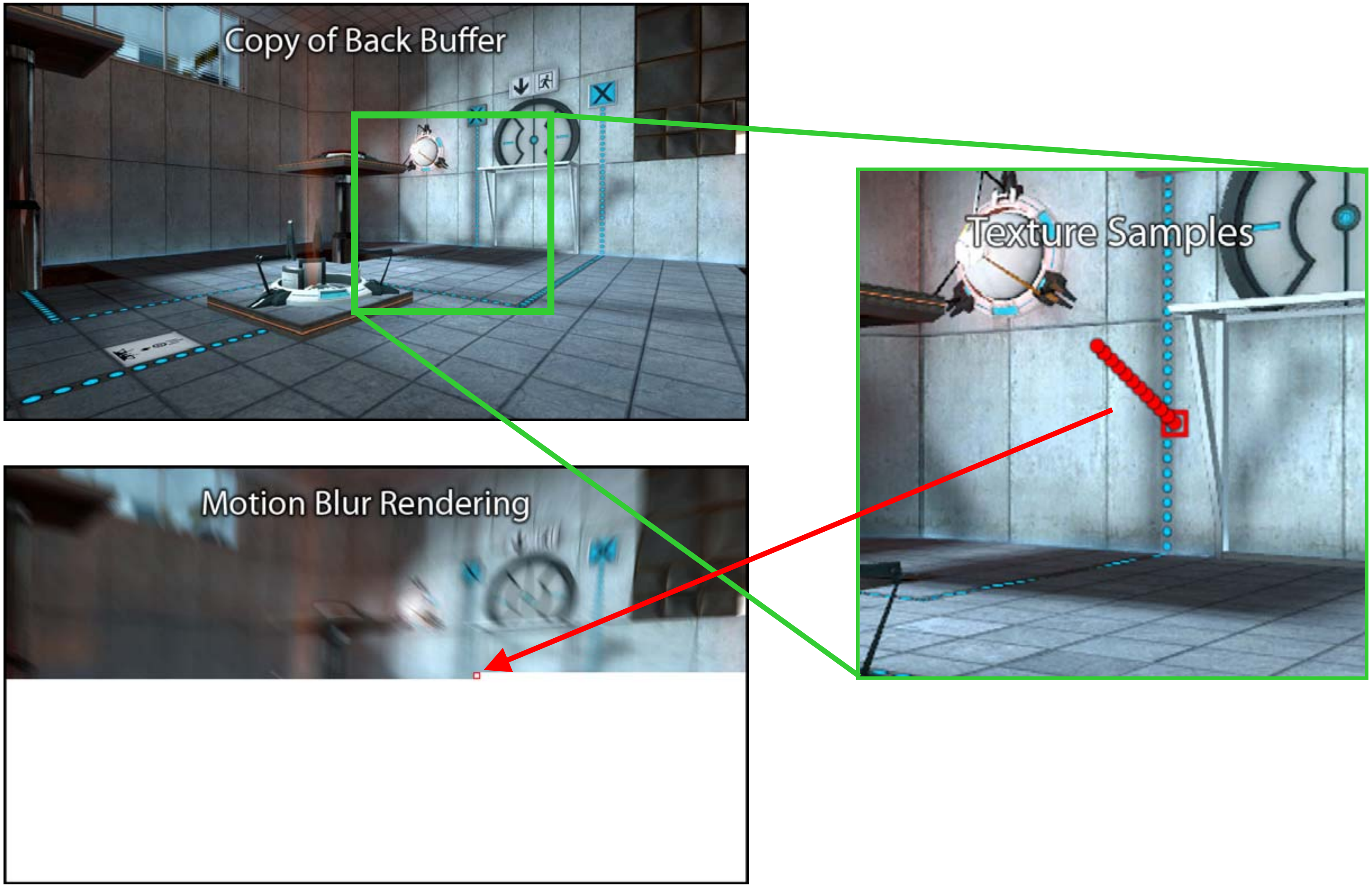
Motion Blur: Where in the Pipeline?

We don't want to blur the weapon!



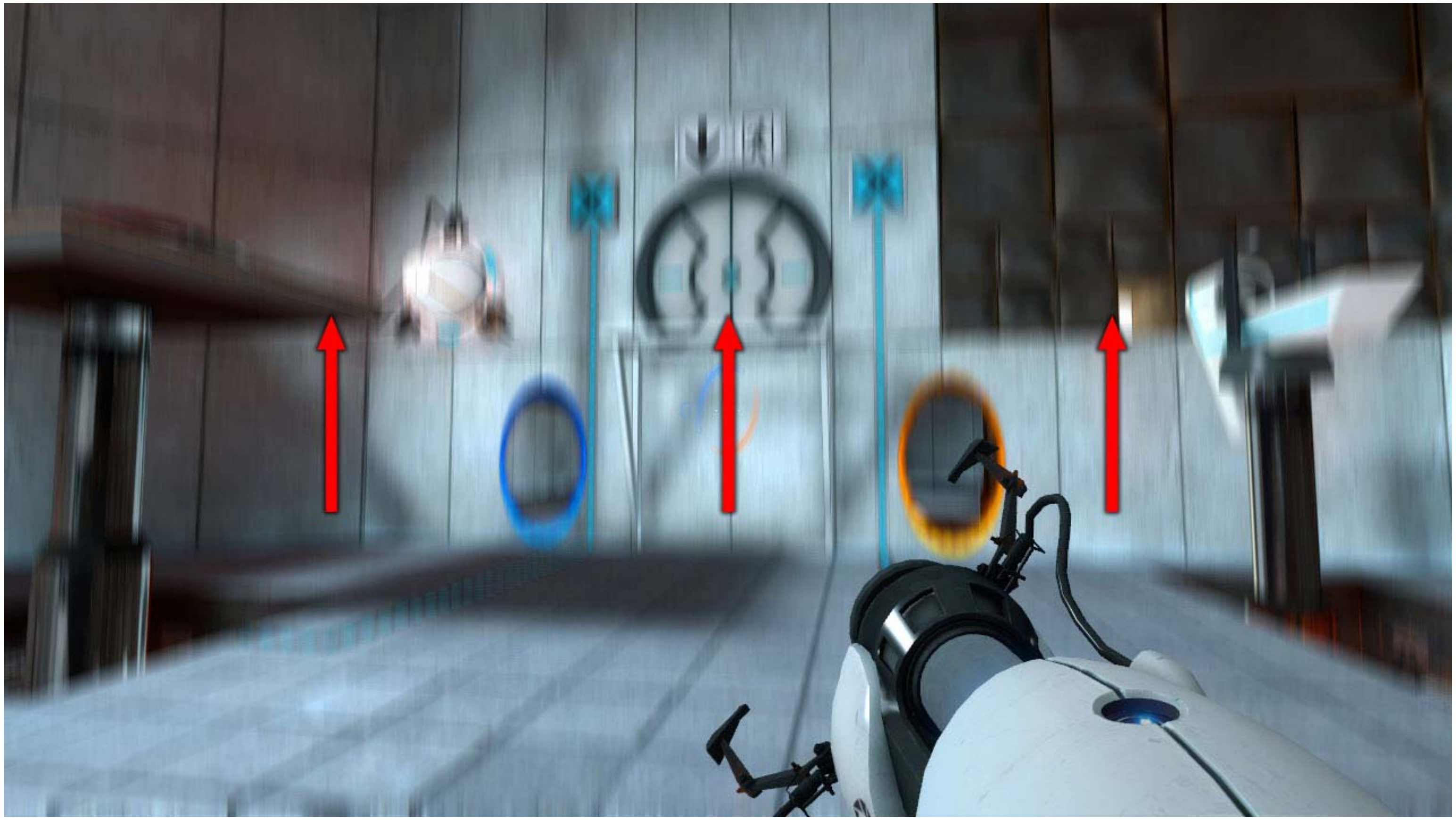
1. Render full scene
2. **Motion blur**
3. Render view model / weapon
4. Render GUI

Rendering Motion Blur



Camera Rotation: Pitch

- Blur vector is just vertical

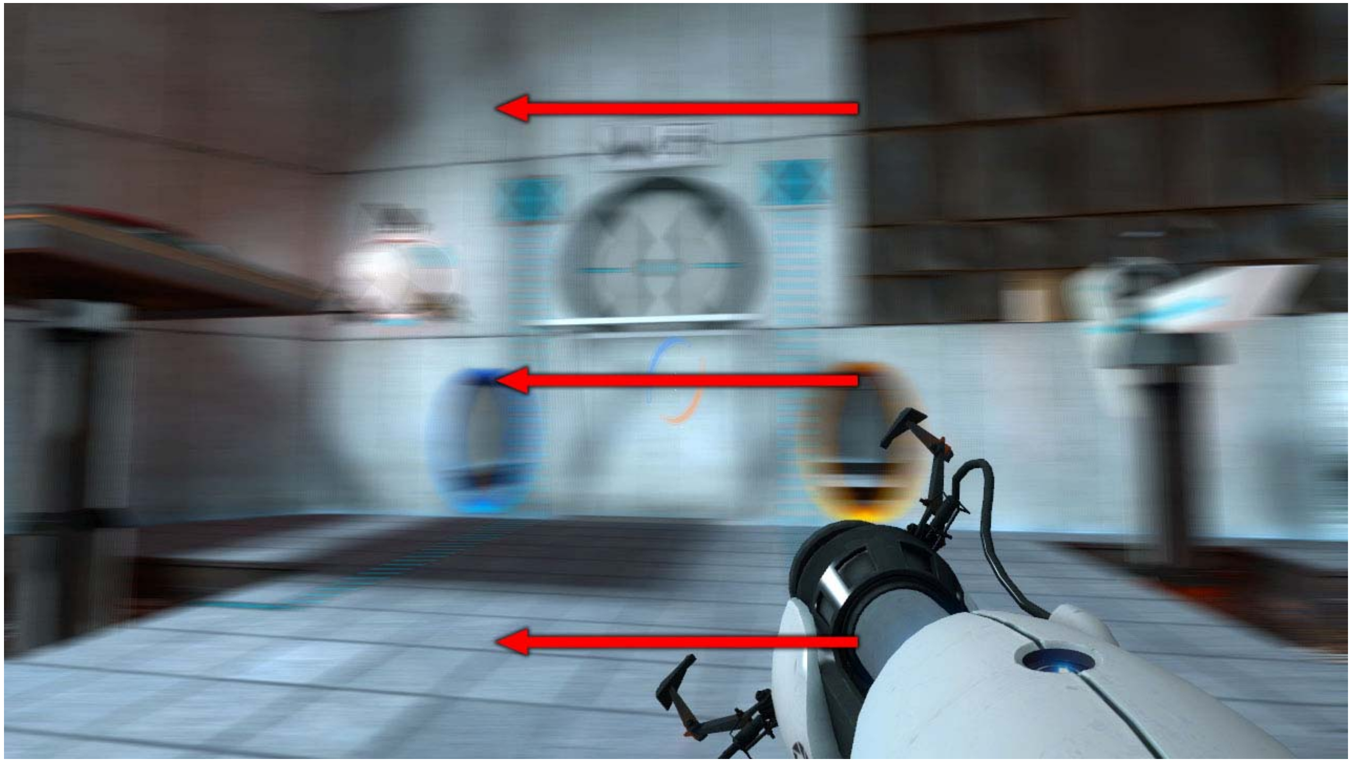


Camera Rotation: Yaw

- Not as simple as pitch
- Need two separate solutions
- We roll when we turn left/right while looking down!

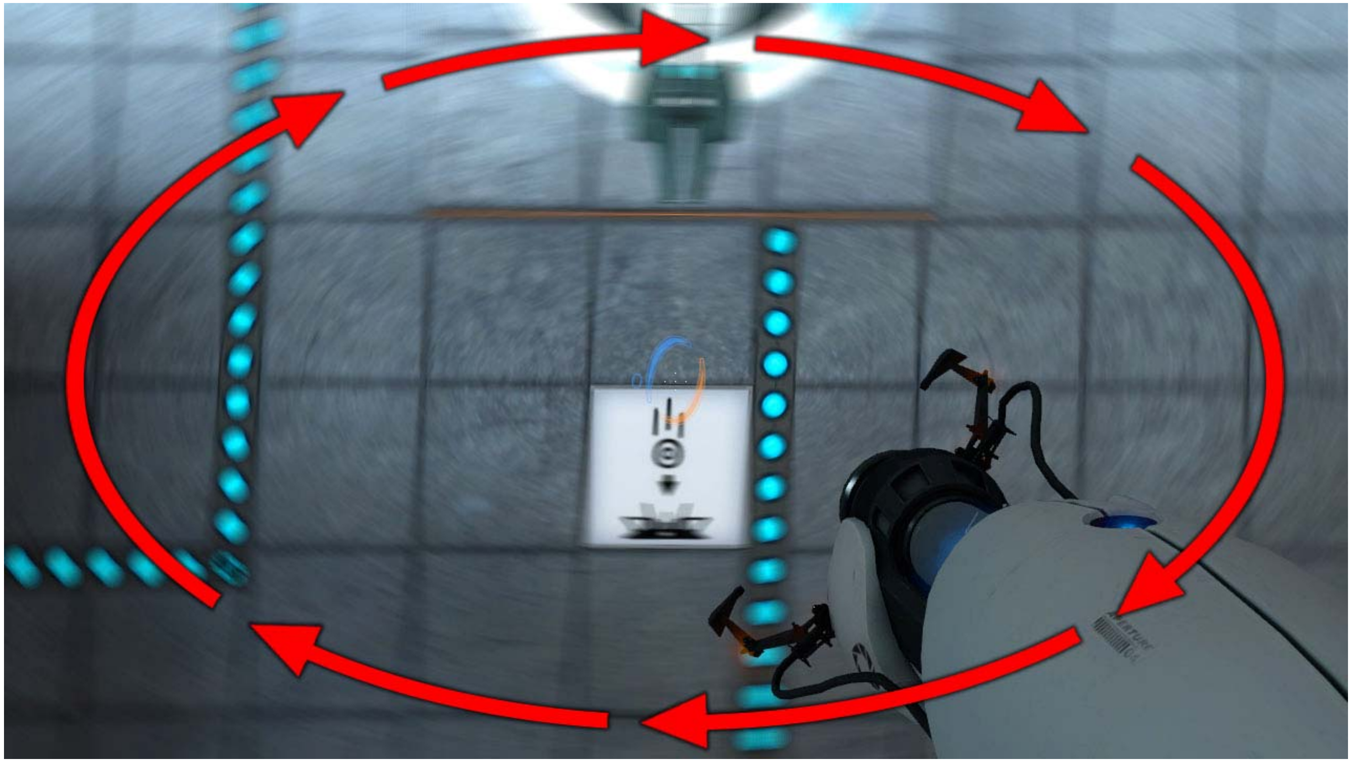
Camera Rotation: Yaw (Part A)

- Blur vector is horizontal
- This fades in/out with pitch



Camera Rotation: Yaw (Part B)

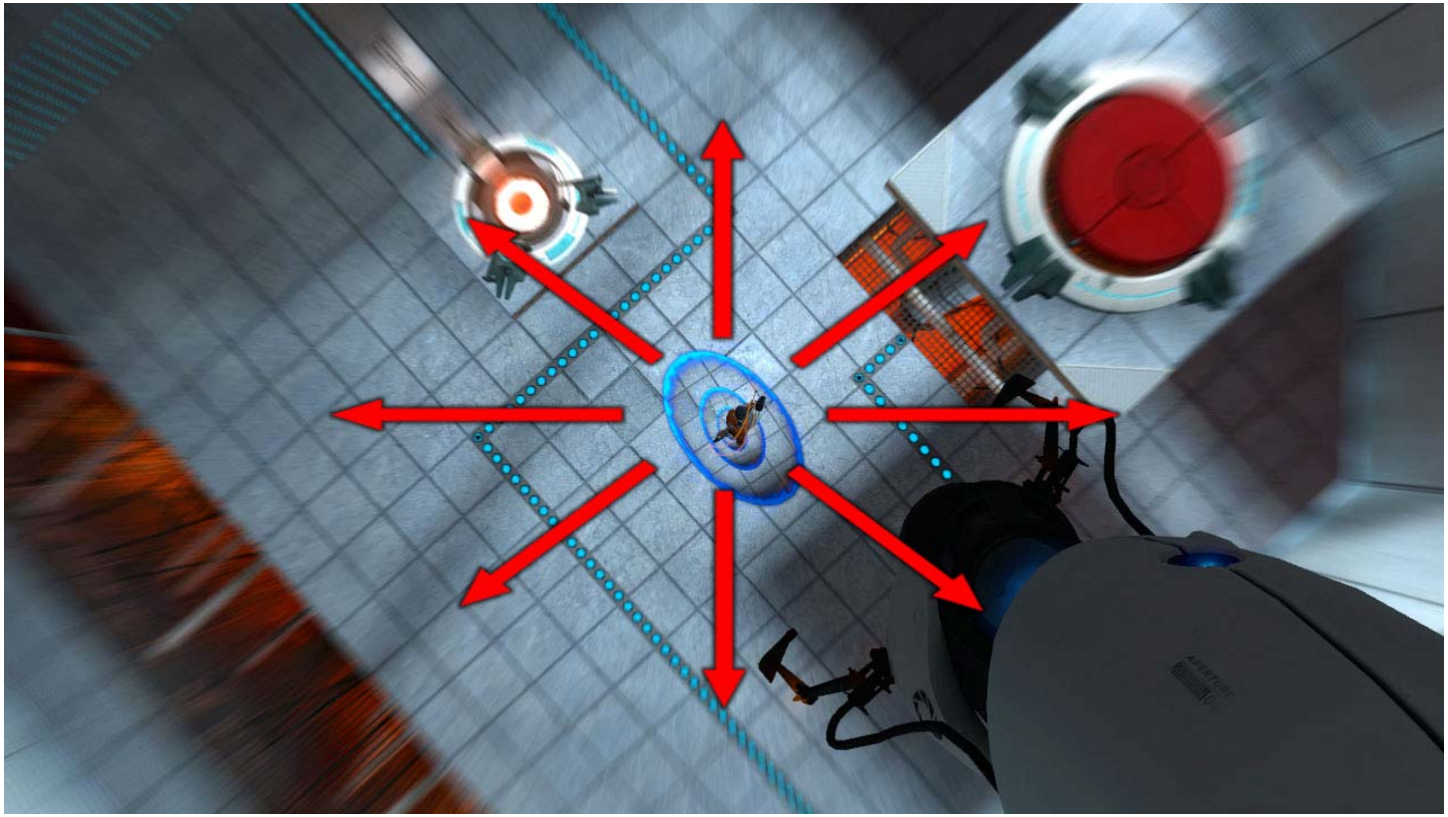
- Roll motion blur
- This fades in/out with pitch



This approximation is very efficient to implement!

Portal Falling Blur

- When falling and looking down generate forward motion vectors



Generating the Final Blur Vector

- Blur vectors computed per-pixel:
  - Pitch: Full screen vertical vector
  - Yaw: Full screen horizontal vector
  - Yaw: Roll vector
  - Falling: Inside/out vector
- Combine these individually weighted vectors
- Sample along the vector and average

Special Case: Portal Transitions

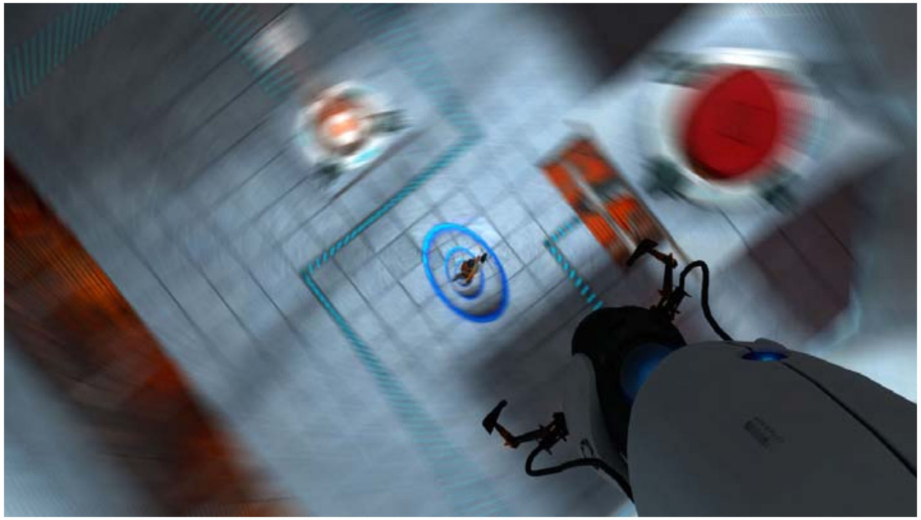
- Moving through portals caused a jolt

- Use last frame’s blur values when moving a far distance in a single frame

Special Case: System Hitches

- Another process stole CPU cycles from the game and caused a hitch
  - “You’ve got mail!”
  - “Time to update your software!”
- System hitches can cause one very blurry frame
- Time lapse between frames greater than 0.1 seconds, we disable motion blur for that frame

Special Case: Headache



- Variable frame rate and blur made people sick
  - Only an issue when frame rate is low with variable frame rate (Does not apply to the 360 since we’re vsync’d!)
  - Motion blur vector is globally scaled down as frame rate drops from 50-30 fps
  - Use minimal motion blur to achieve the effect. We only use 15% of full-frame shutter!
  - Limit blur to 4% of screen width

Motion Blur Summary

- Isolated system
- Blur from camera rotation only
- Special case Portal falling blur
- Acceptable performance & no additional memory
- 90% of Orange Box customers

Summary

- sRGB – DX9, DX10, XBox 360
- Tone Mapping
- Motion Blur
- Additional details about our rendering: http://www.valvesoftware.com/publications.html

Thanks!

Alex Vlachos, Valve

alex@valvesoftware.com