V-Cycle Multigrid in Vulkan for Luminance Upsampling (SDR to HDR)

Richard Everhart, Advanced Application Design Inc.

DISPLAYING SDR (EASY)

Advanced Application Design, Inc.

- Don’t do anything. You’re already good to go.
- Send (optionally) OETF encoded data over wire
  - sRGB, rec.709, etc.
- Monitor SoC (System on Chip) applies Gamma 2.4 (or 2.2) EOTF

DISPLAYING HDR (ALSO EASY)

ADVANCED APPLICATION DESIGN, INC.

- First, make sure your data is actually HDR data
  - Did your image loader, tone-mapper, renderer, compositor, or GPU driver smush your dynamic range?
  - *At least one of them probably did*.
    - **Example:** Driver applies incorrect TF (srgb) during bilinear filtering
  - Did you use at least 10 bit color channels (if using rgb)?
  - Did you handle your mastering/target display discrepancies correctly?
  - Is your signal data in rec. 2100 colorspace?

- Send HDR10 metadata as a `cta_861g_infoframe`
  - Monitor is now in “HDR mode”

- Send PQ (Perceptual Quantizer) OETF encoded data over wire
  - Be mindful of your MaxCLL

- Monitor SoC applies PQ EOTF
  - Because the monitor is in HDR mode

DISPLAYING SDR AND HDR AT THE SAME TIME

- What if you are compositing or blending both HDR and SDR data?
  - Game engine output is HDR, but UI is SDR
  - Wayland compositors (because `color-management-v1` is a thing now)
  - You don't want to modeset to handle broadcasting signal discrepancies

- **The problem:** HDR and SDR are not compatible (Assuming a PQ EOTF)
  - SDR content only looks right when an SDR EOTF is applied
  - HDR content only looks right when PQ EOTF is applied
  - Your monitor only applies one EOTF to its signal data

DISPLAYING SDR AND HDR AT THE SAME TIME

- You can force everything to be SDR
  - Tone map HDR to SDR using iCAM06, or something similar
  - But then HDR looks bad!
- You can force everything to be HDR
  - Display SDR in the appropriate nit range (SDR white is 100 nits)
  - But then SDR looks bad!
- You do a weird hybrid approach
  - SDR and HDR both look bad, but not *as bad*

DISPLAYING SDR AND HDR AT THE SAME TIME

Advanced Application Design, Inc.

- **The better solution:** Inverse tone map SDR to HDR
- This makes SDR content look like it was designed/rendered originally as HDR content
- **It can make bad HDR content look much better**
- But, good inverse tone mapping is hard!
  - Requires generating new information
  - Computationally expensive
  - Requires sophisticated understanding of colorimetry and DSP to debug
  - Can produce visual artifacts if done improperly (halos, washout, flickering)

DISPLAYING SDR AND HDR AT THE SAME TIME

- **Perfect inverse tone mapping has no image quality tradeoff**
  - Often improves the quality of images by uncovering latent details
  - Has ample efficiency tradeoffs

- This presentation covers the inverse tone mapping step of a real-time SDR to HDR post processing pipeline

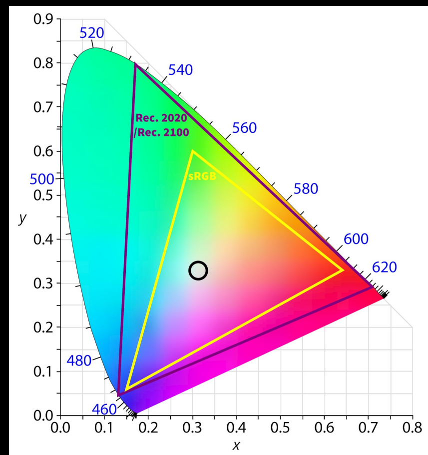
- We won't cover:
  - How to actually modulate the gradient
  - How to use the output luminance data
  - Temporal awareness
  - A bunch of other things!

INVERSE TONE MAPPING: WHERE TO BEGIN?

- First, understand what HDR is supposed to look like

- **Good HDR**
  - Drastically increases image contrast, while preserving or injecting detail
    - 1600 nits vs 200 nits leaves a lot of contrast up for grabs!
  - Whites are only bright if they are light sources
  - Preserves artistic intent
  - More vibrant colors because `rec.709` ⊂ `rec.2100`
  - Corrects the Abney effect and Bezold-Brücke shift

- **Bad HDR**
  - May inflate image contrast while nuking image details
  - Whites may be bright even if they are not light sources
  - May introduce unrealistic effects like halos
  - May demonstrate banding
  - May look washed out
  - May use dithering to hide artifacts



BAD HDR: NO DETAILS

Advanced Application Design, Inc.

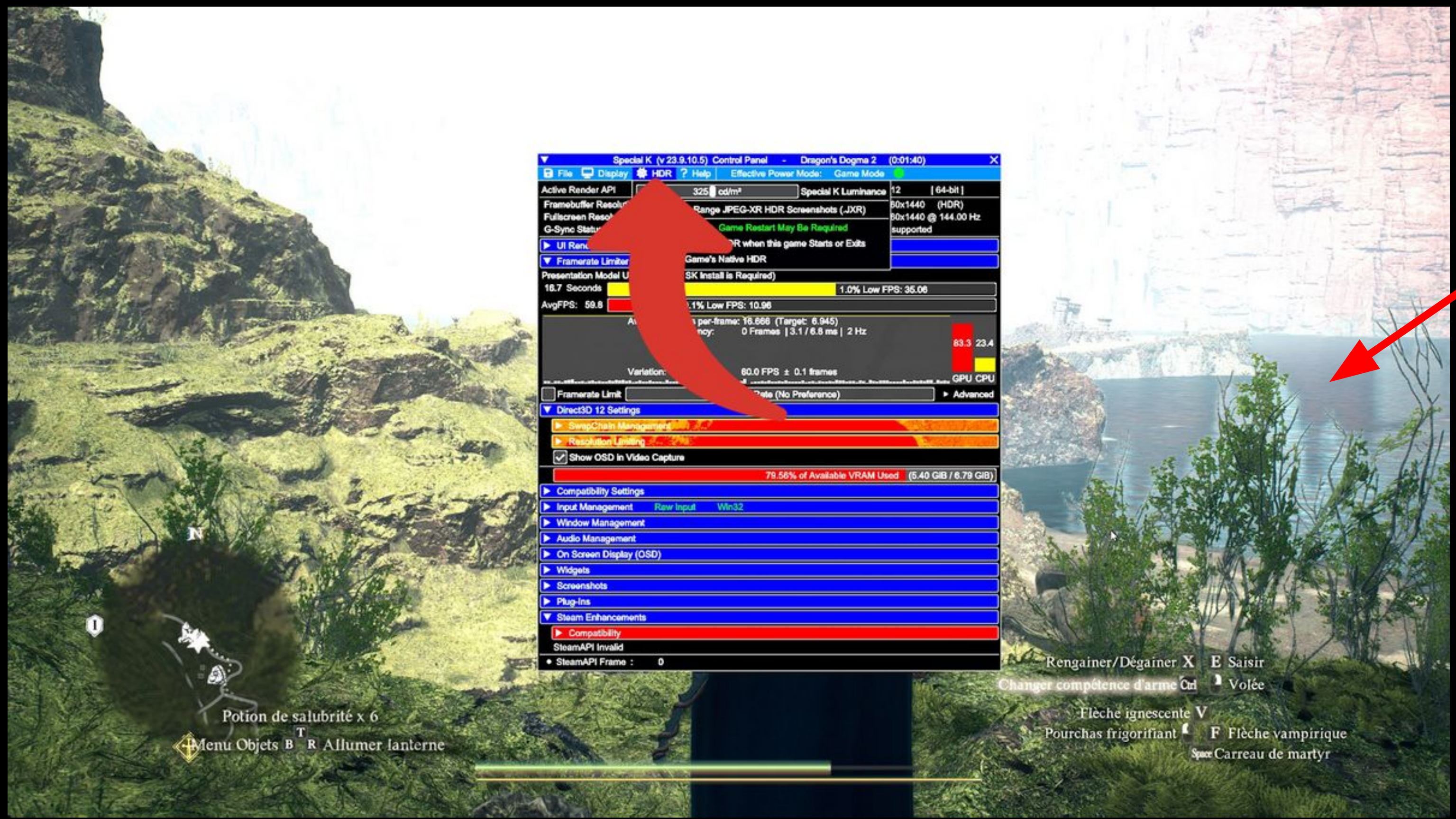
BAD HDR: NO DETAILS

Advanced Application Design, Inc.

BAD HDR: WHITES ARE BRIGHT

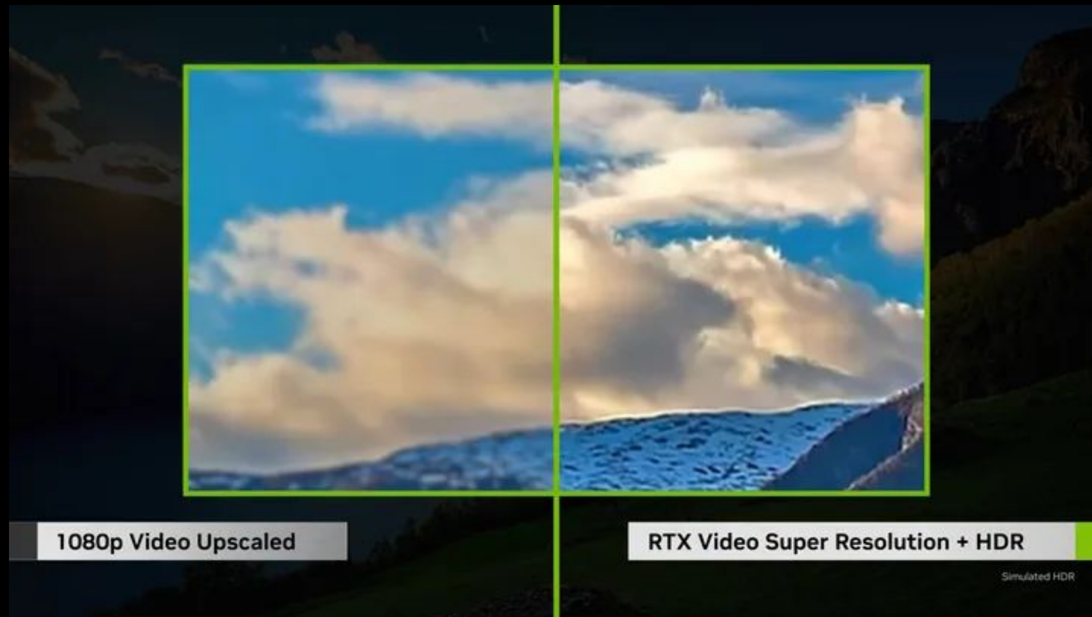
BAD HDR: HALOS

ADVANCED APPLICATION DESIGN, INC.



BAD HDR: HALOS

Advanced Application Design, Inc.



BAD HDR: BANDING

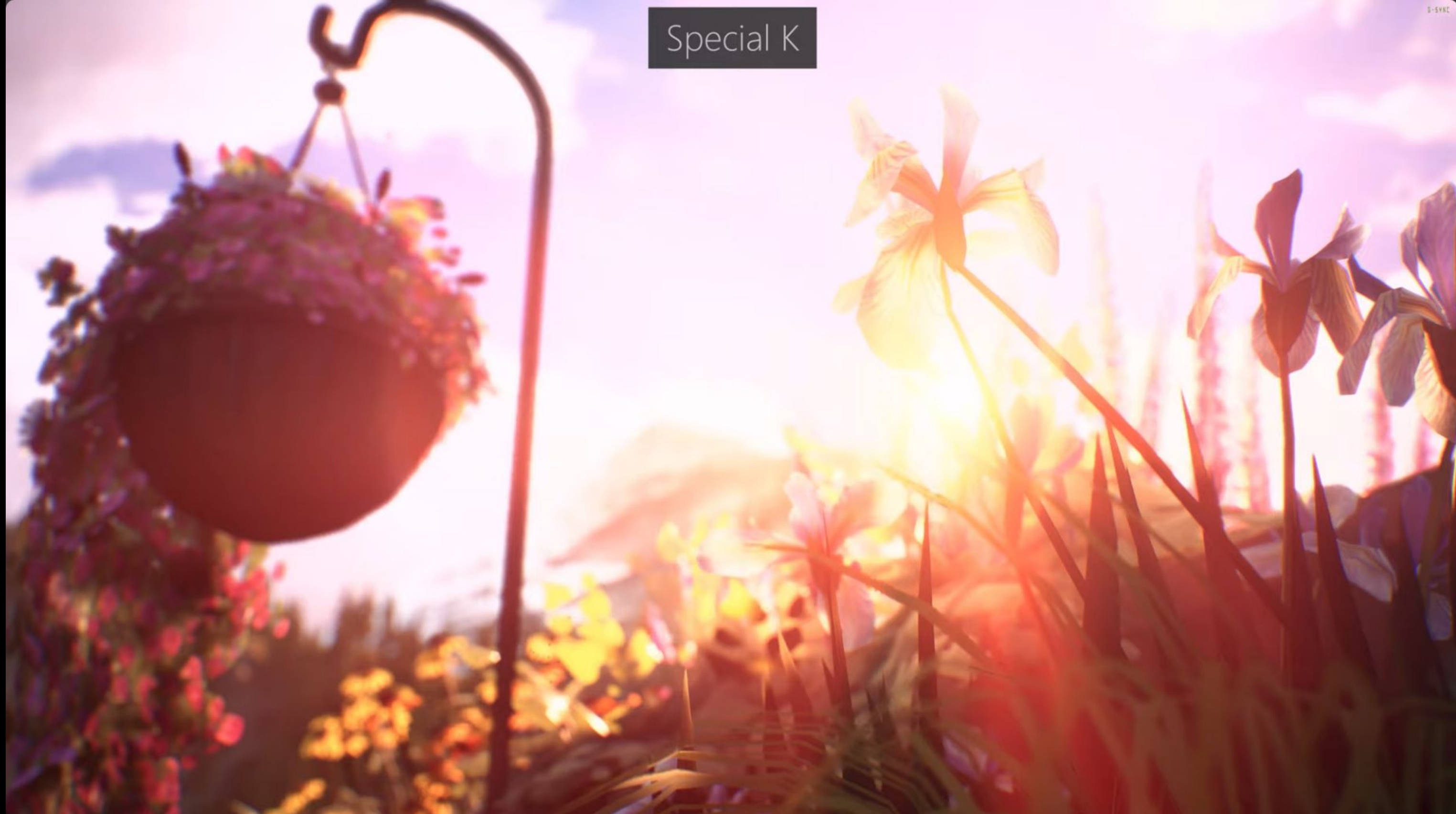
Advanced Application Design, Inc.

BAD HDR: BANDING

BAD HDR: WASHOUT

Advanced Application Design, Inc.

BAD HDR: WASHOUT



INVERSE TONE MAPPING

- Spatially-aware, perceptually weighted inverse tone mapping avoids bad HDR

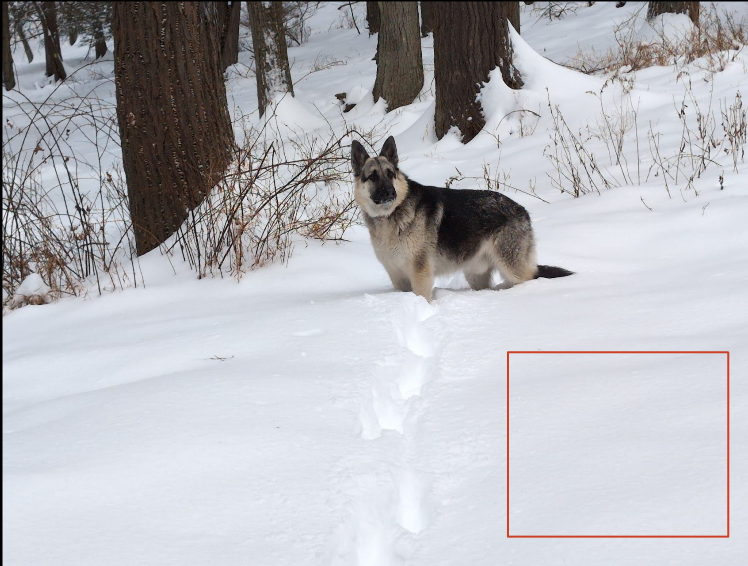
- We're going to use a *gradient domain* inverse tone mapping approach
  - Don't work in luminance, but rather luminance gradient
  - Texture can be used to determine if something is actually bright, or just white

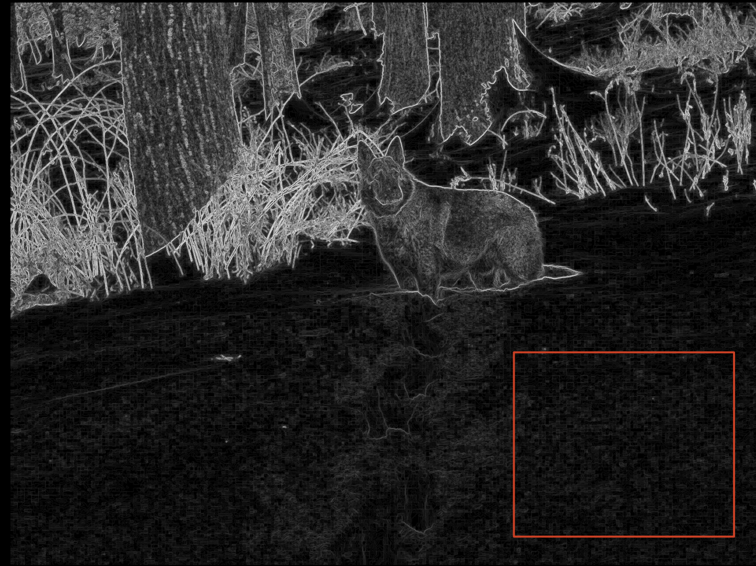
- After modulating the gradient, we can construct the final image by solving

$$\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right)\varphi(x, y) = f(x, y)$$

- Combine new luma with original chroma, without changing hue

INVERSE TONE MAPPING



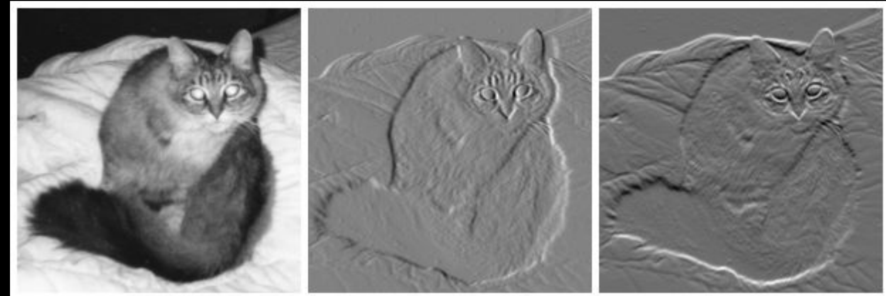


Super inflate invisible texture, force PDE solver to converge on a less bright solution

INVERSE TONE MAPPING

- **Tone:** Gradient of luminosity
- **Inverse Tone Map:** Transformation of lower dynamic luminosity gradient to higher dynamic luminosity gradient
    - f is the divergence of our modulated luminance gradient
    - φ is our source luminance
    - $\nabla^2 = \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right)$ is the divergence of luminance

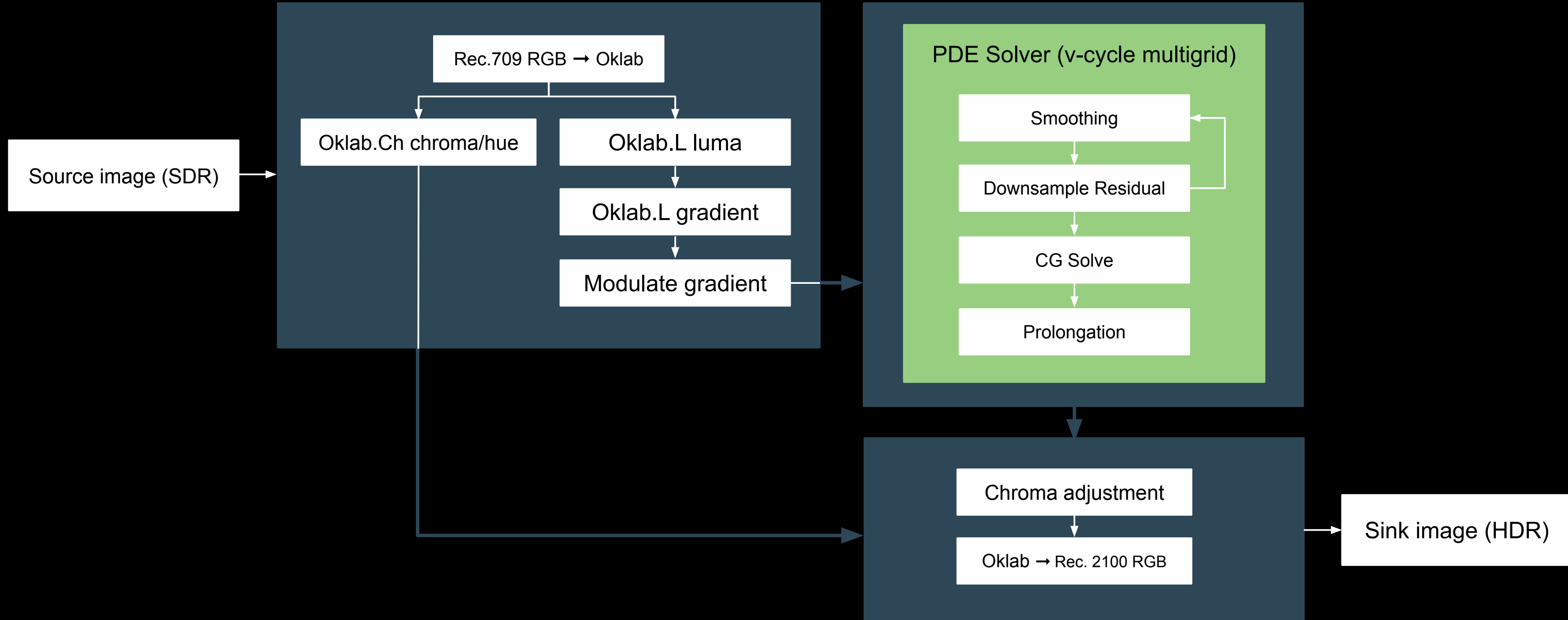
$$\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right)\varphi(x, y) = f(x, y)$$



COLOR VOLUME CONVERSION

Advanced Application Design, Inc.

Use a perceptual color space (Oklab) so that human perception doesn't discard luminance details.



INVERSE TONE MAPPING

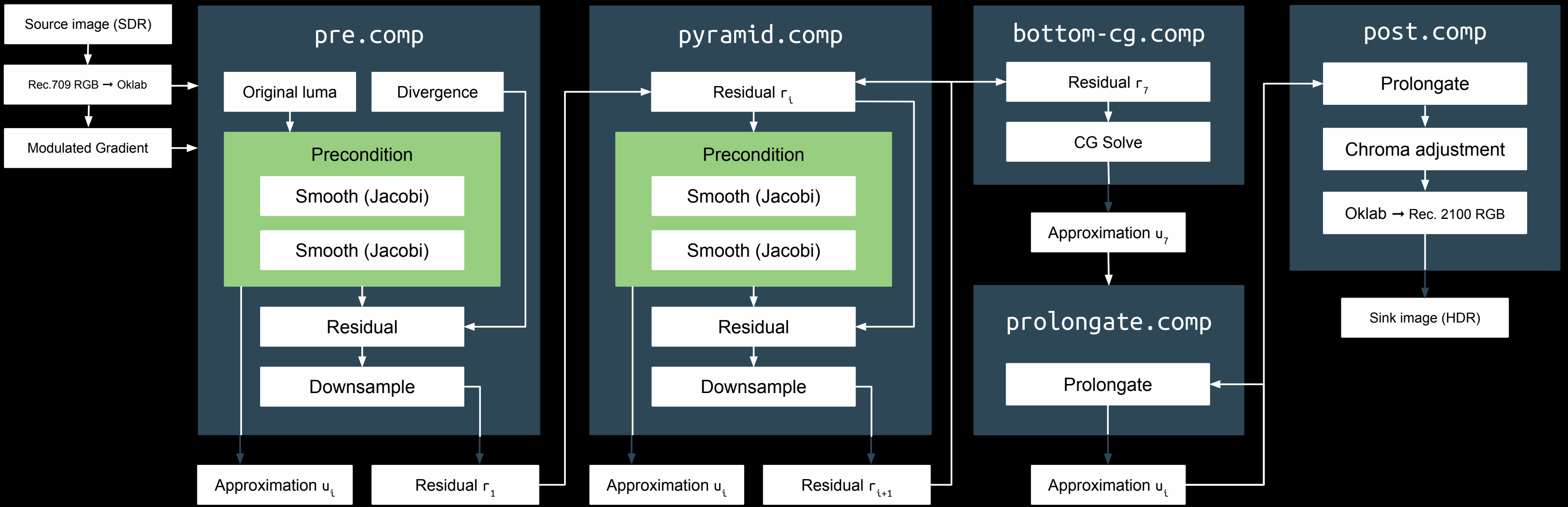
- **Goal:** Solve Poisson in <1ms on AMD’s Van Gogh (RDNA V2) APU

- **Problem:** Solving PDEs is expensive
  - Amounts to solving a linear system with `IMAGE_SIZE` number of variables
  - “Easy” methods like Jacobi, Gauss-Seidel are no-goes here

- **Solution:** Use *v-cycle multigrid*
  - VkFFT is too slow (took 5-9ms)!
  - Variable, due to periodic boundary conditions requiring more global reads

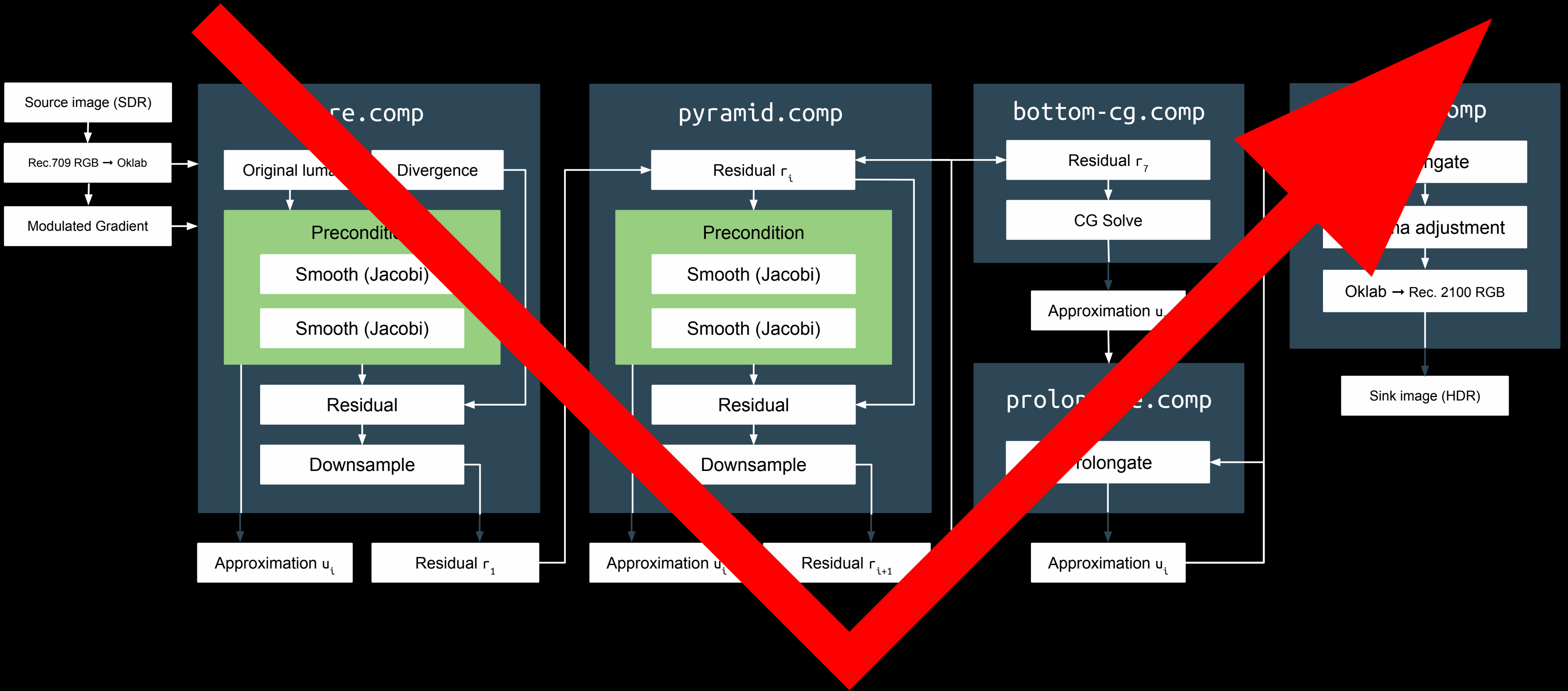
V-CYCLE MULTIGRID

Advanced Application Design, Inc.



V-CYCLE MULTIGRID

Advanced Application Design, Inc.



SETUP (VULKAN)

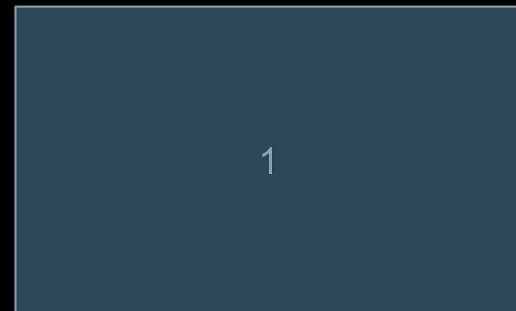
ADVANCED APPLICATION DESIGN, INC.

- Wayland client asks for buffer to be created
  - `zwp_linux_buffer_params_v1::create/create_immed`
  - Force the client to use non-scanout tranche
  - **Note:** No Wayland framework allows this! We had to roll our own.

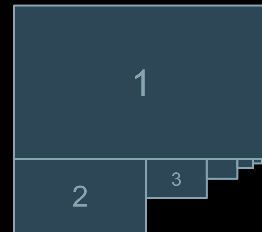
- Allocate `VkImage` (or `VkBuffer`) and export as DMABUF for client
  - **`VK_EXT_external_memory_dmabuf`**
  - **Note:** Vulkan has no way of getting the size of buffers/images that have DRM modifiers
    - See Vulkan spec, `vkGetImageSubresourceLayout()`
    - Go through gbm as needed

- Create storage images/buffers
  - `source (SRC_WIDTH, SRC_HEIGHT)`
  - `fine (SRC_WIDTH, SRC_HEIGHT)`
  - `pyramid_ui (SRC_WIDTH / 2, SRC_HEIGHT / 2 + SRC_HEIGHT / 4)`
  - `pyramid_ri (SRC_WIDTH / 2, SRC_HEIGHT / 2 + SRC_HEIGHT / 4)`
  - `sink (SRC_WIDTH, SRC_HEIGHT)`

- Setup synchronization primitives:
  - **`VK_KHR_external_memory_fd`**, `IN_FENCE_FD`, `OUT_FENCE_PTR`



`source, fine, sink`



`pyramid_ui, pyramid_ri`

SETUP (VULKAN)

- Command buffer will dispatch 13 shaders
  - `pre.spv`
  - `pyramid-1.spv`
  - `pyramid-2.spv`
  - `pyramid-3.spv`
  - `pyramid-4.spv`
  - `pyramid-5.spv`
  - `bottom-cg.spv`
  - `prolongate-5.spv`
  - `prolongate-4.spv`
  - `prolongate-3.spv`
  - `prolongate-2.spv`
  - `prolongate-1.spv`
  - `post.spv`

- `vkCmdPipelineBarrier2()` between each dispatch

- Only 6 grids
  - `1920 / 2^{5+1} = 30, 1080 / 2^{5+1} ~= 16`
  - We use AMD FSR, NVIDIA NIS, or NVIDIA DLSS for 4K

SETUP (VULKAN)

- We also use timestamp queries

```
vkCmdResetQueryPool(
    rend_multigrid->vulkan_cmd_buf, rend->vulkan_query_pool, 0,
    LK_MAX_TIMESTAMPS
);
vkCmdWriteTimestamp(
    rend_multigrid->vulkan_cmd_buf, VK_PIPELINE_STAGE_TOP_OF_PIPE_BIT,
    rend->vulkan_query_pool, 0
);

// Dispatches

vkCmdWriteTimestamp(
    rend_multigrid->vulkan_cmd_buf, VK_PIPELINE_STAGE_BOTTOM_OF_PIPE_BIT,
    rend->vulkan_query_pool, 1
);
```

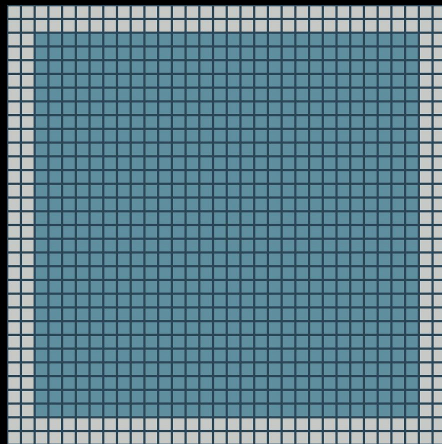
GENERAL OPTIMIZATIONS

- Load tiles in LDS (shared memory)

- Tile size of 32 * 32
  - Divergence, Jacobi requires (x, y) difference calculations
  - Inner: 28 * 28 = 784px
  - Halo: 2px width, or 240px (~23% of our tile)

- Roughly 23% of our calculations are redundant here
  - Redundancy is better than global memory access

- This could be lowered with a bigger tile size, but it isn’t worth it
  - Selected tile size avoids bank conflicts



```
shared float tile_0[(TILE_W) * (TILE_H)];
shared float tile_1[(TILE_W) * (TILE_H)];
shared float tile_2[(TILE_W) * (TILE_H)];
```

BANK CONFLICTS

- Shared memory banks can only address one dataset at a time
- If multiple threads in a wavefront/warp go through the same bank, then data access is serialized
  - Very bad!
- On AMD RDNA V2, bank address is given as LDS offset (in 32-bit DWORDs) modulo bank count (32)

pre.comp

ADVANCED APPLICATION DESIGN, INC.

- Load modulated luminance gradient tile into LDS
- Compute divergence of modulated gradient
  - Dirichlet boundary conditions
- Run Jacobi twice on original luminance, using ping-pong LDS buffers
  - Dirichlet boundary conditions
- Calculate residual as $r_i = f_i - \Delta u_i$
  - f_i is divergence of modulated gradient (`tile_2`)
  - u_i is current “best guess”, smoothed luminance (`tile_0`)

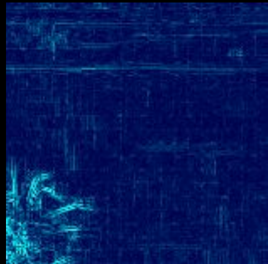
```
if (is_in_image) {
    vec2  G  = vec2(tile_1[ci], tile_2[ci]);
    float Gx = tile_1[li];
    float Gy = tile_2[di];

    tile_0[ci] = (G.x - Gx) + (G.y - Gy);
    tile_2[ci] = tile_0[ci];
}
barrier();

/* Jacobi iteration 1 / 2 (unrolled).*/
if (is_in_image) {
    float Du = tile_0[ri];
    float Dd = tile_0[li];
    float Dr = tile_0[ui];
    float Dl = tile_0[di];

    tile_1[ci] =
        0.25f * (Du + Dd + Dl + Dr - tile_3[ci] * h);
}
barrier();

/* Jacobi iteration 2 / 2 (unrolled).*/
if (is_in_image) {
    float Du = tile_1[ri];
    float Dd = tile_1[li];
    float Dr = tile_1[ui];
    float Dl = tile_1[di];

    tile_0[ci] =
        0.25f * (Du + Dd + Dl + Dr - tile_3[ci] * h);
}
barrier();
```

`pre.comp`

- **Problem**: Divergence calculation and Jacobi method calculates x, y differences
  - 1px halo for divergence, 1px halo for Jacobi iteration 1, and 1px halo for Jacobi iteration 2
  - But, we only have a 2px halo!
  - We're even worse off if we also do gradient calculation in `pre.comp`

- **Solution:** Ignore this problem, handle DC shift in post
  - If we use another shader for div calculation, we pay multiple global memory access penalties
    - Jacobi requires *divergence* and *original luminance* (two reads)
  - If we use another shader for div calculation, we pay a global synchronization penalty
  - If we expand our halo, we decrease coverage
    - Results in 423 (~24%) more workgroups for a 1080p image

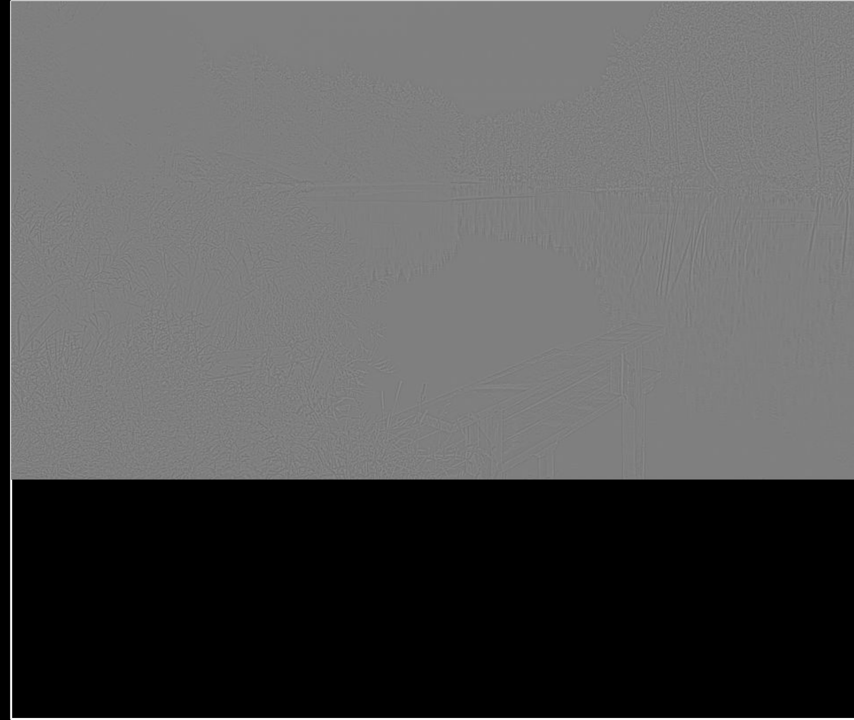
- We save up to ~0.3ms (~0.5ms) this way
  - Arguably significant!



$e^{3x} - 1$

Divergence of modulated gradient

Smoothed luminance written to `fine` storage buffer



Downsampled residual written to `pyramid_ri` storage buffer

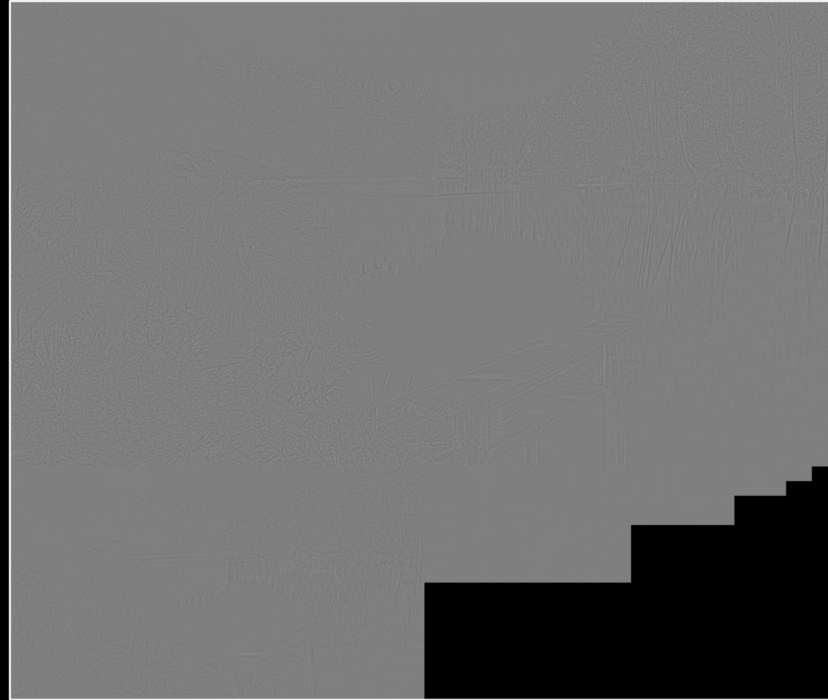
pyramid.comp

ADVANCED APPLICATION DESIGN, INC.

- Load previous residual into LDS
- Run Jacobi twice on previous residual data, using ping-pong LDS buffers
  - Dirichlet boundary conditions
- Calculate residual as $r_i = f_i - \Delta u_i$
  - f_i is previous residual r_{i-1} (`tile_2`)
  - u_i is the previous residual, smoothed (`tile_0`)
- Do this for 5 (progressively coarser) grids

```
/* Jacobi iteration 1 / 2 (unrolled).*/
if (is_in_image) {
    tile_1[ci] =
        0.25f * (0.0f - tile_3[ci] * h2);
}
barrier();

/* Jacobi iteration 2 / 2 (unrolled). */
if (is_in_image) {
    float Du = tile_1[ui];
    float Dd = tile_1[di];
    float Dr = tile_1[ri];
    float Dl = tile_1[li];

    tile_0[ci] =
        0.25f * (Du + Dd + Dl + Dr - tile_3[ci] * h2);
}
barrier();

/* Residual calculation */
if (is_in_image && is_in_inner) {
    float Yu = tile_0[ui];
    float Yd = tile_0[di];
    float Yr = tile_0[ri];
    float Yl = tile_0[li];

    tile_1[ci] =
        tile_2[ci] -
        (Yu + Yd + Yr + Yl - 4.0f * tile_0[ci]) / h2;
}
barrier();
```

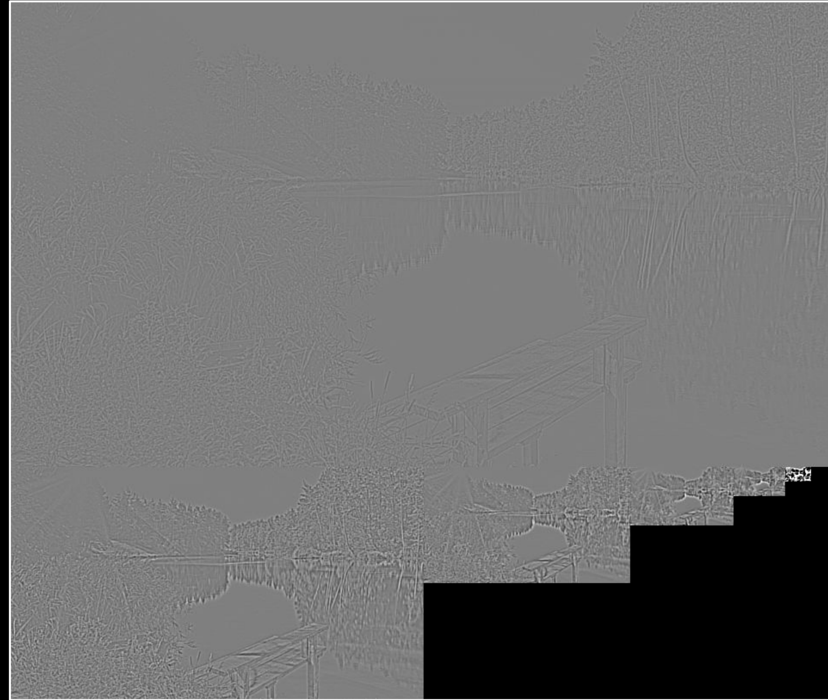
pyramid.comp



Downsampled residual written to `pyramid_ri` storage buffer

pre.comp

Advanced Application Design, Inc.



Smoothed luminance written to `pyramid_ui` storage buffer

bottom-cg.comp

ADVANCED APPLICATION DESIGN, INC.

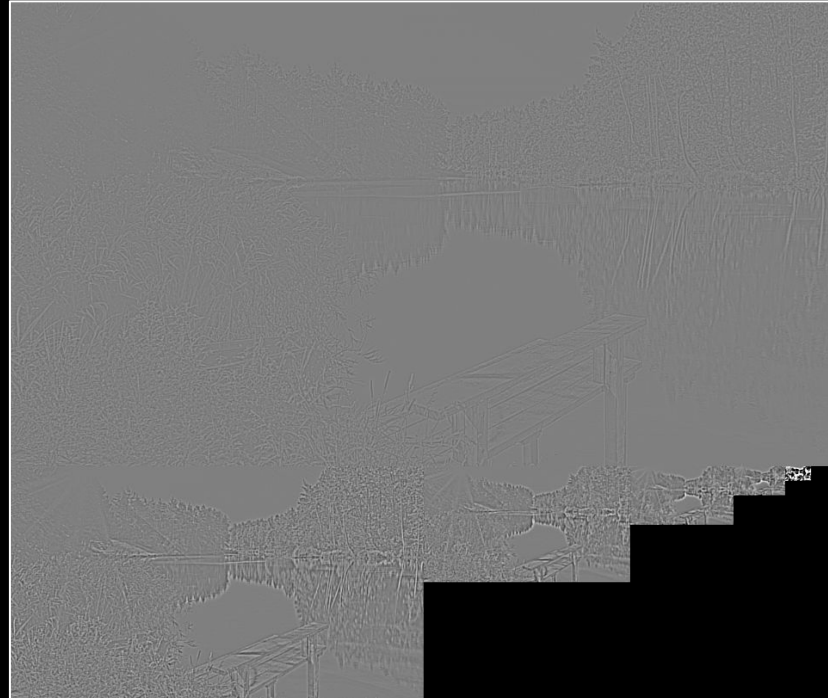
- Bottom level fits into one LDS tile

- Solve using whatever system of linear equations solver you want!
  - Jacobi
  - Gauss-Seidel
  - FFT
  - Conjugate Gradient

- We use Conjugate Gradient
  - `GLSL_SUBGROUP_ADD` really helps out here

bottom-cg.comp

Advanced Application Design, Inc.



Solved luminance written to `pyramid_ui` storage buffer

prolongate.comp

ADVANCED APPLICATION DESIGN, INC.

- Store the previous (more coarse) level in LDS

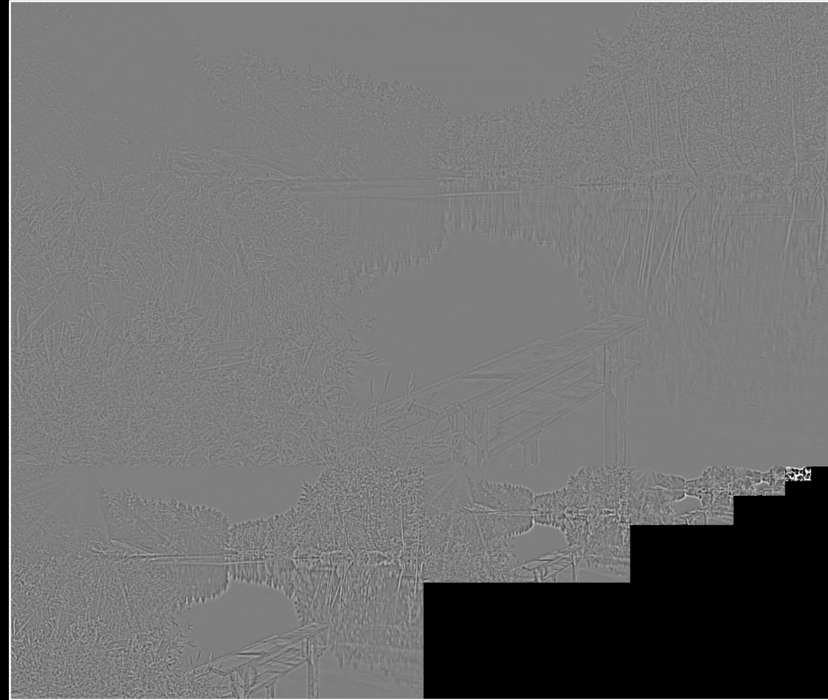
```
if (is_in_image) {
    if ((tx <= (TILE_W / 2 + 1)) && (ty <= (TILE_H / 2 + 1))) {
        tile_0[TILE_IDX(tx, ty)] = ri_pyramid_data[PYRAMID_i_plus_1_IDX(px / 2, py / 2)];
    }
}
```

- Prolongate to get luminance correction, and update pyramid

```
if (is_in_image) {
   float v = 0.0f;

   if ((tx % 2 == 0) && (ty % 2 == 0)) {
       v = tile_0[ci];
   }
   else if ((tx % 2 == 1) && (ty % 2 == 0)) {
       v = (tile_0[ci] + tile_0[ri]) / 2.0f;
   }
   else if ((tx % 2 == 0) && (ty % 2 == 1)) {
       v = (tile_0[ci] + tile_0[ui]) / 2.0f;
   }
   else {
       v = (tile_0[ci] + tile_0[ri] + tile_0[ui] + tile_0[rui]) / 4.0f;
   }

   ui_pyramid_data[PYRAMID_i_IDX(px, py)] += v;
}
```

prolongate.comp

Advanced Application Design, Inc.



Solved luminance written to `pyramid_ui` storage buffer

post.comp

Advanced Application Design, Inc.

- Prolongate to final luminance (not correction)
- Perform 0-2 additional v-cycles
  - Not 100% necessary
- Finish up
  - Account for DC
  - Adjust chroma
  - Oklab ➞ Rec. 2100 RGB
  - Apply PQ OETF as needed
  - `VK_FORMAT_B16G16R16_UINT` or `VK_FORMAT_A2B10G10R10_UINT_PACK32`
    - Depending on driver, whether buffers or images were used, etc.

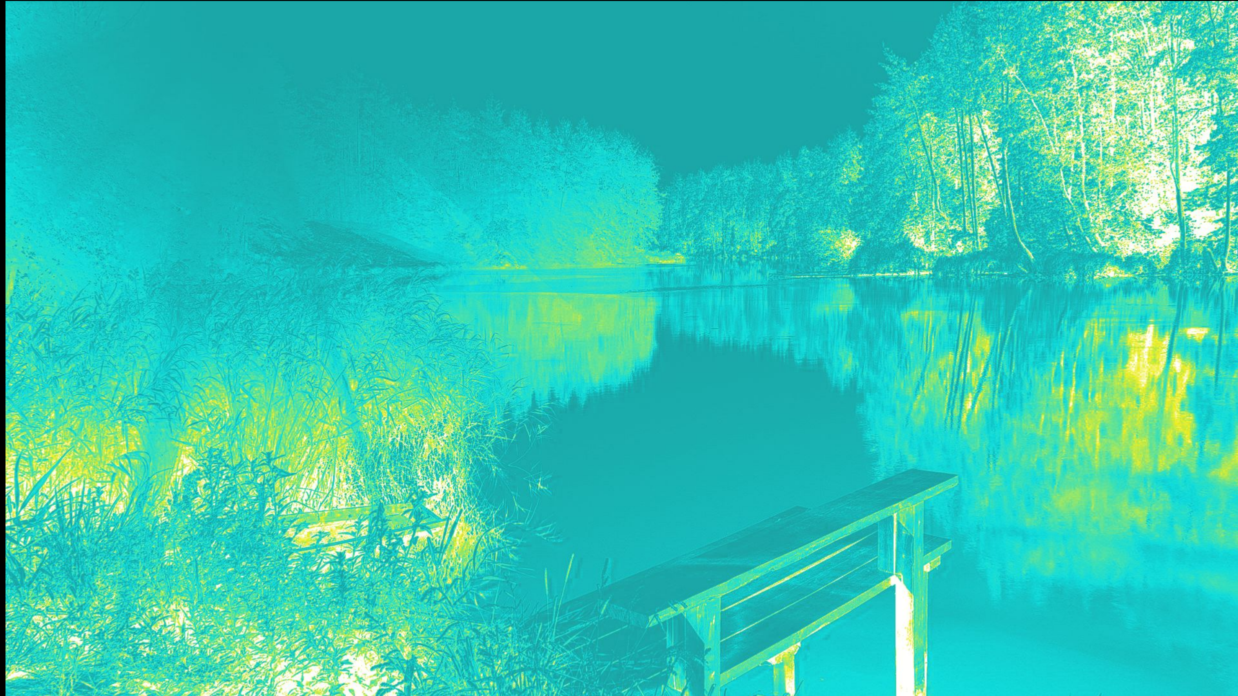
```
if (is_in_image) {
    float v = 0.0f;

    if ((tx % 2 == 0) && (ty % 2 == 0)) {
        v = tile_0[ci];
    }
    else if ((tx % 2 == 1) && (ty % 2 == 0)) {
        v = (tile_0[ci] + tile_0[ri]) / 2.0f;
    }
    else if ((tx % 2 == 0) && (ty % 2 == 1)) {
        v = (tile_0[ci] + tile_0[ui]) / 2.0f;
    }
    else {
        v = (tile_0[ci] + tile_0[ri] + tile_0[ui] + tile_0[rui]) / 4.0f;
    }
}
```

Final image written to `sink` storage buffer

Noticeably higher contrast!

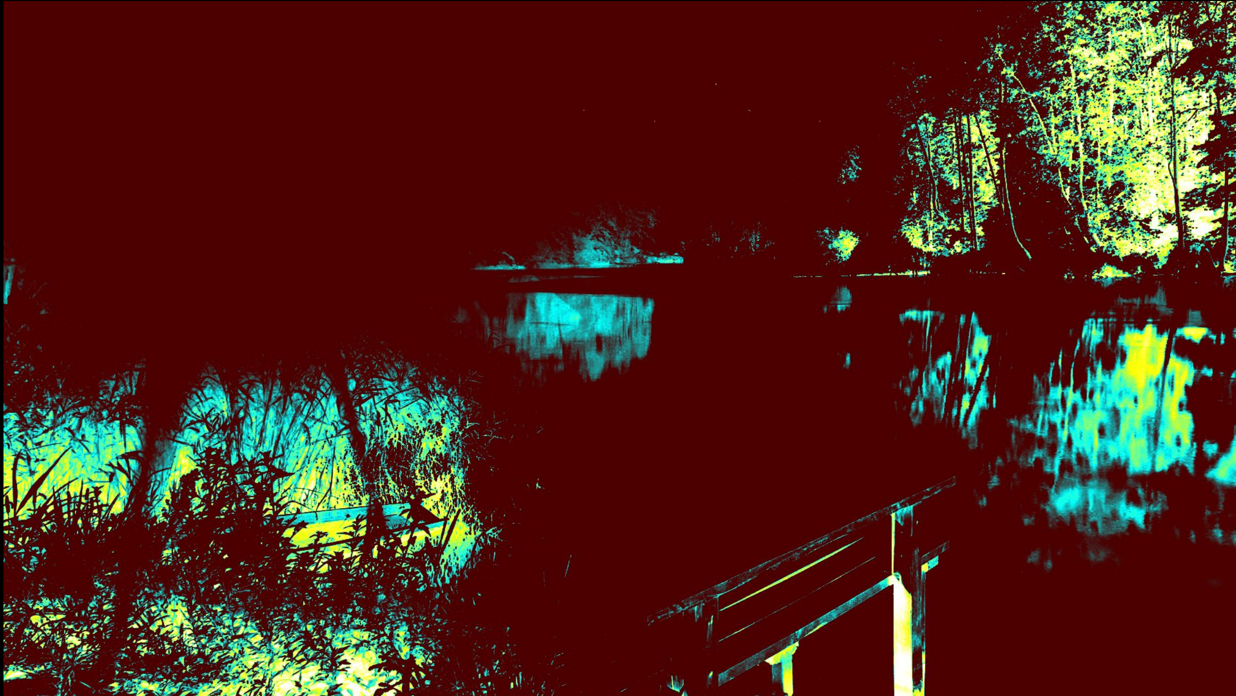
Noticeably higher contrast!

Relative darkening map (false color)

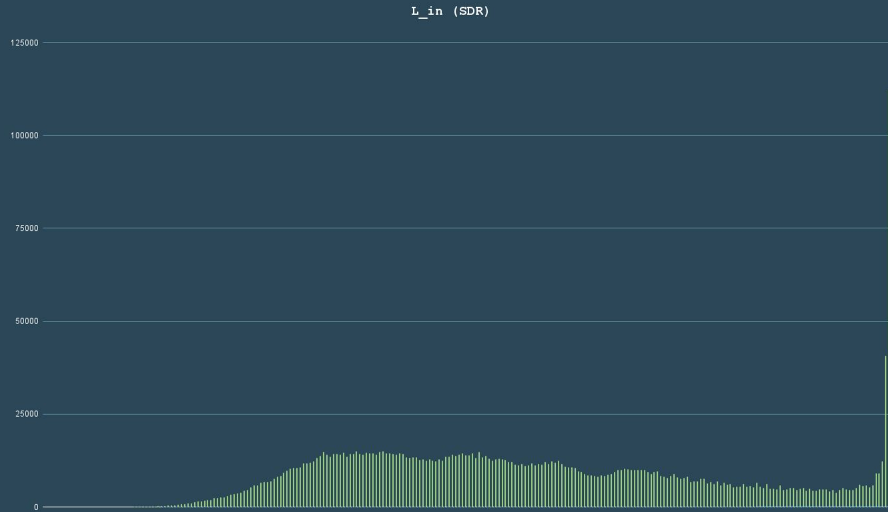


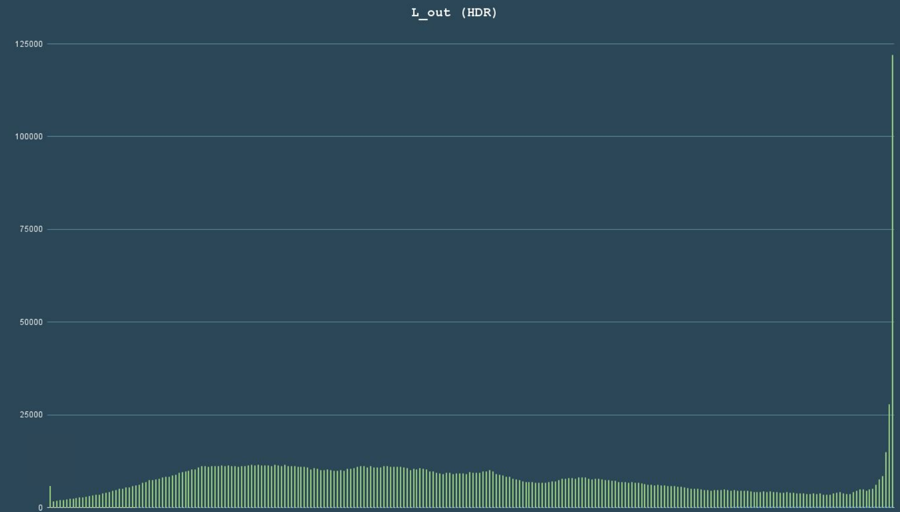
Relative darkening map (false color)

Relative darkening map (false color, beautified)



Shadow emphasis map (false color)





Histograms of luminance (256 bins)

$$||\nabla L_{\text{in}}||$$

Advanced Application Design, Inc.

$$||\nabla L_{\text{out}}||$$

Advanced Application Design, Inc.

$$||\nabla L_{\text{out}}|| - ||\nabla L_{\text{in}}||$$

Output image is best viewed in HDR

Output image is best viewed in HDR

Output image is best viewed in HDR

post.comp

Advanced Application Design, Inc.

post.comp

Advanced Application Design, Inc.

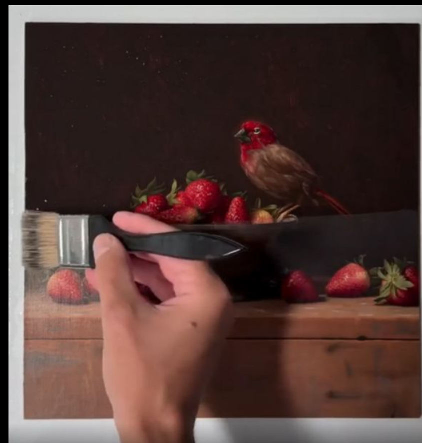
Note: Not the same as sharpening the image

Advanced Application Design, Inc.

Note: Not the same as saturating the image

RESULTS

- Time: 0.89ms - 0.92ms on an AMD Van Gogh APU
  - 0.39ms on a NVIDIA GTX 1080
  - 0.26 ms on a NVIDIA RTX 3080
  - **Note:** Compute can be done asynchronously
- Restores optical intent of signal data
  - Does not stylize the image
  - Overcomes Abney effect and Bezold-Brücke shift
  - Output looks better when rendered in SDR, or HDR
- Digital varnish
  - Varnish naturally ensures rays scatter less, which allows details to pop
  - We mathematically increase detail, and force them to pop

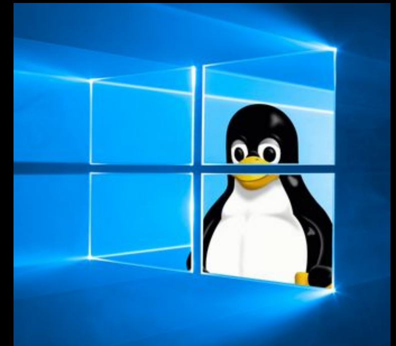


RESULTS

- Spatially aware, perceptually weighted reconstruction is substantially better than existing approaches

- Windows Auto HDR and Special K are neither perceptual, nor spatially aware
  - scRGB is not perceptual
  - Rely on “smart” curves which are largely spatially agnostic
  - These are essentially Photoshop filters

- Configurable for various display types and conditions
  - We can account for various conditions in gradient modulation
  - OLEDs are delicate in the blacks
  - Makes “dimmed” (low power) displays look better

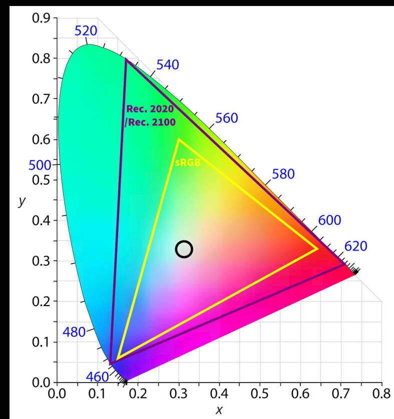


EXAMPLES

- To fairly see the SDR/HDR difference, we need to render SDR and HDR side by side ***without distorting either***
  - *Not an issue for HDR/HDR*

- This generally isn't possible, but on some hardware the pipe is big enough to:
  - Place output in SDR mode
  - Calculate `inv_sdr_eotf(pq_eotf(pq_oetf(x))` in shader without crushing blacks
    - Because `sdr_eotf(inv_sdr_eotf(pq_eotf(pq_oetf(x))) = pq_eotf(pq_oetf(x))`
  - Super brittle! Not for production

- This comparison is weighted in the original SDR image's favor
  - Our HDR output must use a restricted gamut
  - Our gradient modulation method must be more conservative
  - We can't send HDR metadata



EXAMPLES (SDR-MODE)

EXAMPLES (SDR-MODE)

EXAMPLES (SDR-MODE)

Advanced Application Design, Inc.

EXAMPLES (SDR-MODE/HDR-MODE)

Advanced Application Design, Inc.

EXAMPLES (SDR-MODE/HDR-MODE)

EXAMPLES (SDR-MODE/HDR-MODE)

Advanced Application Design, Inc.

EXAMPLES (HDR-MODE/HDR-MODE)

EXAMPLES (SDR-MODE/HDR-MODE)

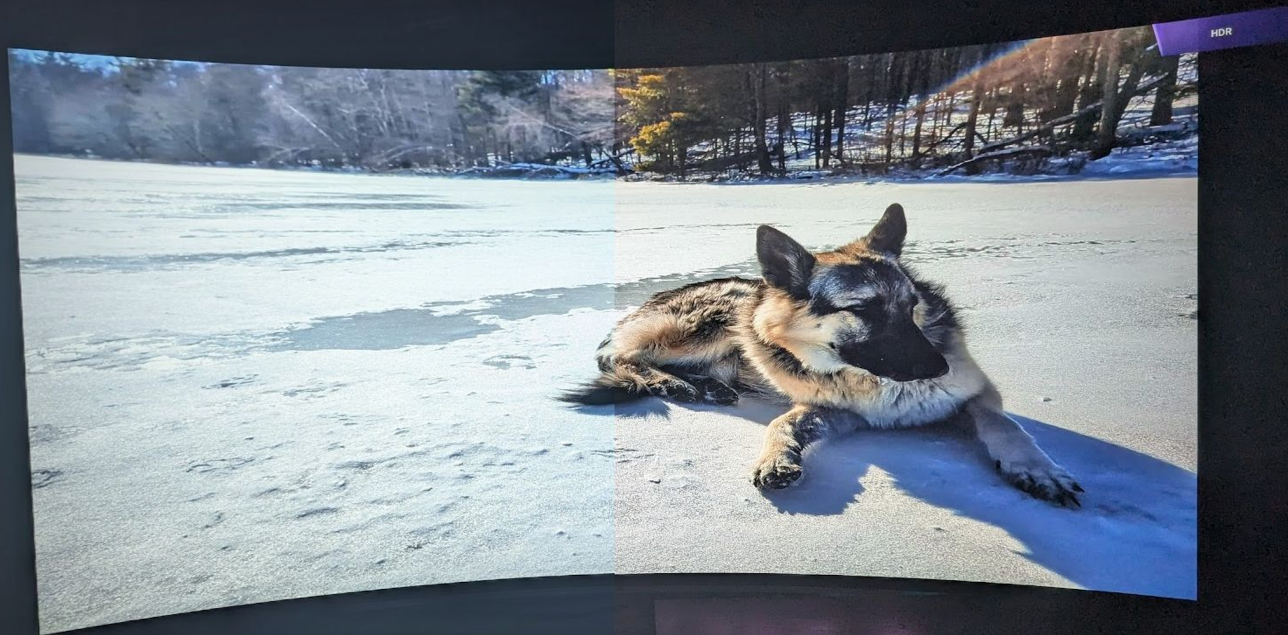
ADVANCED APPLICATION DESIGN, INC.

EXAMPLES (HDR-MODE/SDR-MODE)

Advanced Application Design, Inc.

EXAMPLES (SDR-MODE/HDR-MODE)

Advanced Application Design, Inc.



RESULTS (SDR-MODE/HDR-MODE)

ADVANCED
APPLICATION
DESIGN, INC.

RESULTS (HDR-MODE/SDR-MODE)

Advanced Application Design, Inc.

RESULTS (HDR-MODE/SDR-MODE)

ADVANCED APPLICATION DESIGN, INC.

RESULTS (SDR-MODE/HDR-MODE)

ADVANCED APPLICATION DESIGN, INC.

RESULTS (SDR-MODE/HDR-MODE)

Advanced Application Design, Inc.

RESULTS (SDR-MODE/HDR-MODE)

Advanced Application Design, Inc.

RESULTS (HDR-MODE/SDR-MODE)

Advanced Application Design, Inc.

RESULTS (SDR-MODE/HDR-MODE)

Advanced Application Design, Inc.

RESULTS (SDR-MODE/HDR-MODE)

Advanced Application Design, Inc.

RESULTS (SDR-MODE/HDR-MODE)

ADVANCED APPLICATION DESIGN, INC.

SEE FOR YOURSELF!

See for yourself in Lumakit (lumakit.io)! Coming soon for Linux.

A complete, from-scratch rewrite of Valve's Gamescope, with this stuff, NVIDIA support, and more.

0 validation errors, as opposed to, like, 20.

CONCLUSION

ADVANCED APPLICATION DESIGN, INC.

- Graphical rendering keeps getting better, but color largely remains a forgotten topic

- There is ample quality left on the table
  - As display technology improves, the quality left on the table will increase even more
  - This is already true! More modern media (paradoxically) resulted in more quality gain

- We've spent billions on ray tracing the geometry; it is time we spend a millisecond restoring the color volume

Thank you!

Contact: richard@aadcorp.com