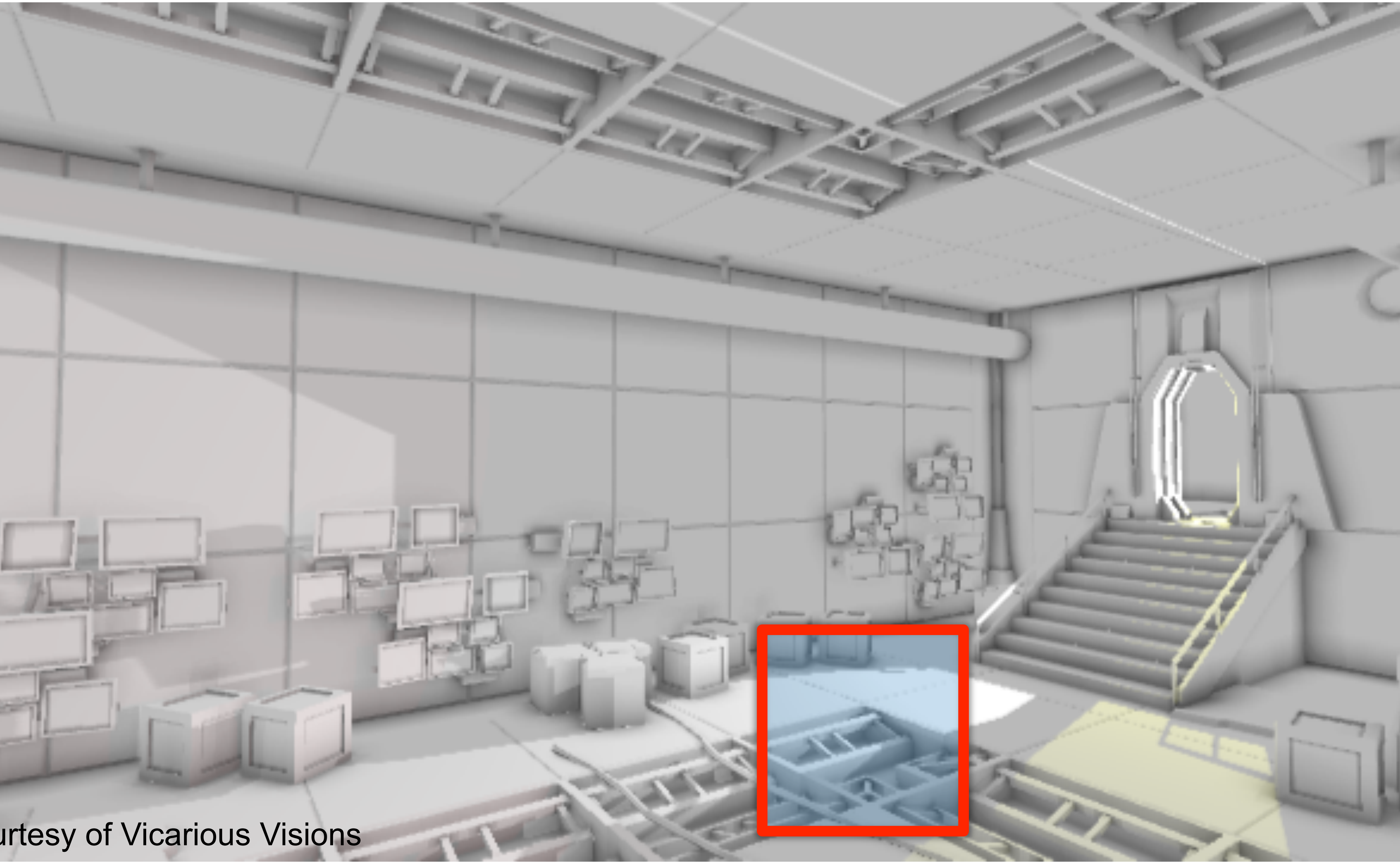
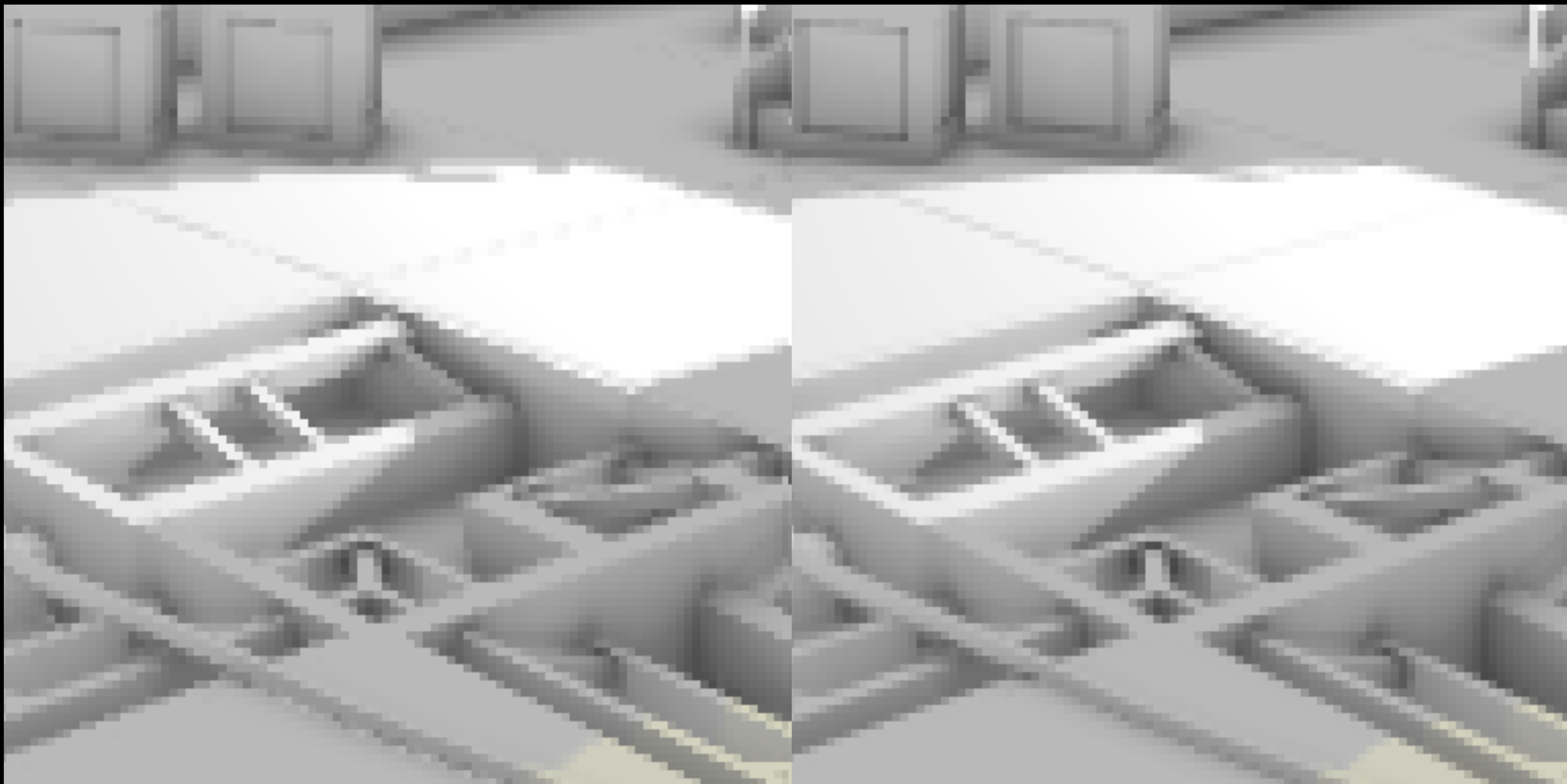
Subpixel Reconstruction Anti-Aliasing

Matthäus G. Chajdas[1,2]
Morgan McGuire[2,3]
David Luebke[2]

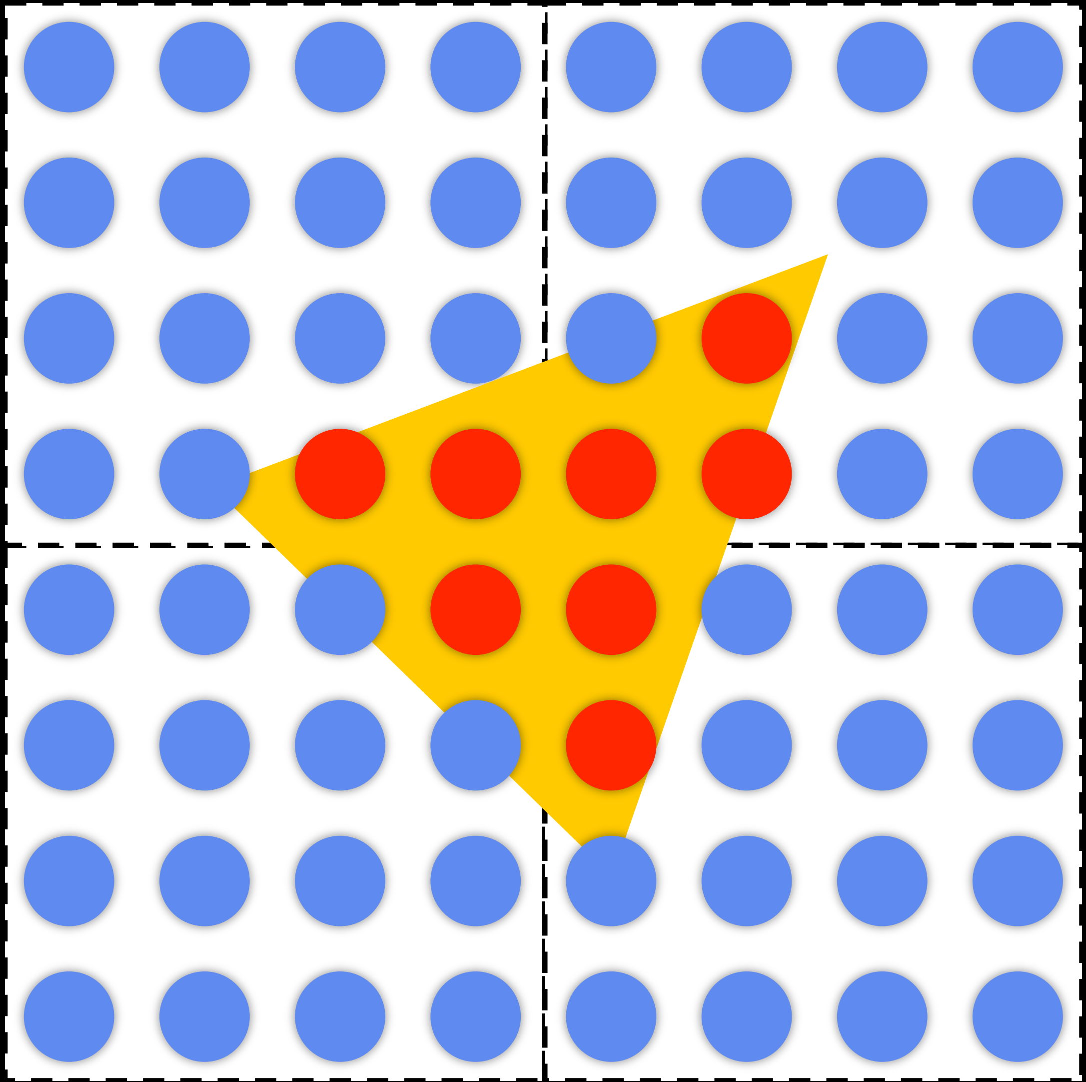
1 Technische Universität München, 2 NVIDIA, 3 Williams College


urtesy of Vicarious Visions

1x Regular vs. 16x SSAA vs. 4x SRAA

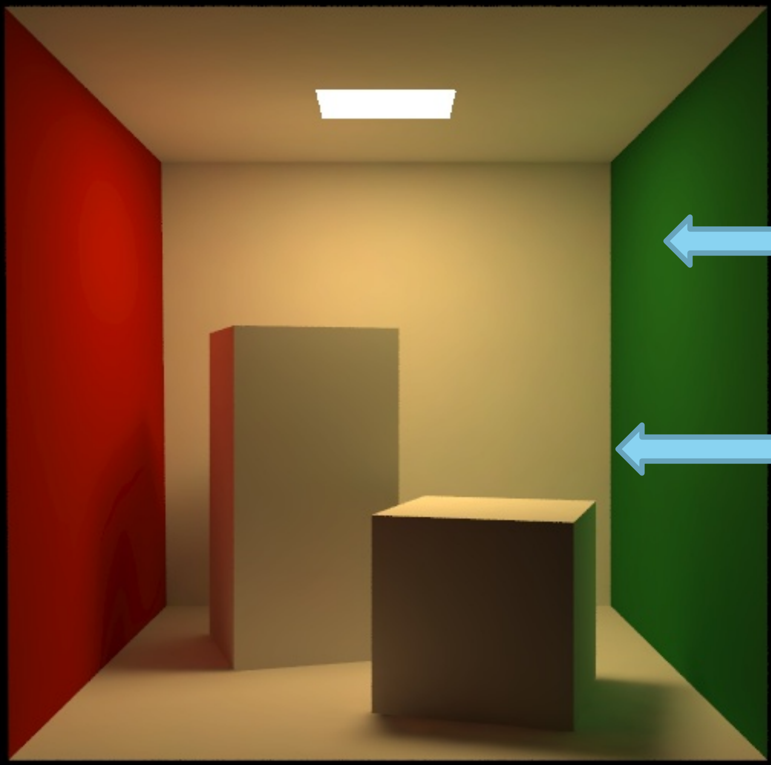


Super-Sample Anti-Aliasing



SRAA – Key observation

- Shading frequency is typically lower than geometric frequency

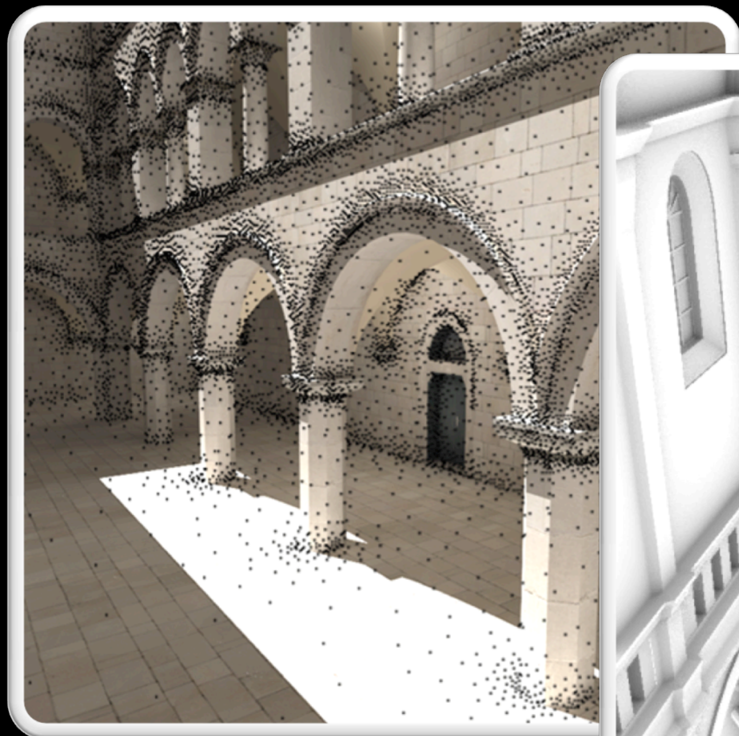


Shading changes smoothly

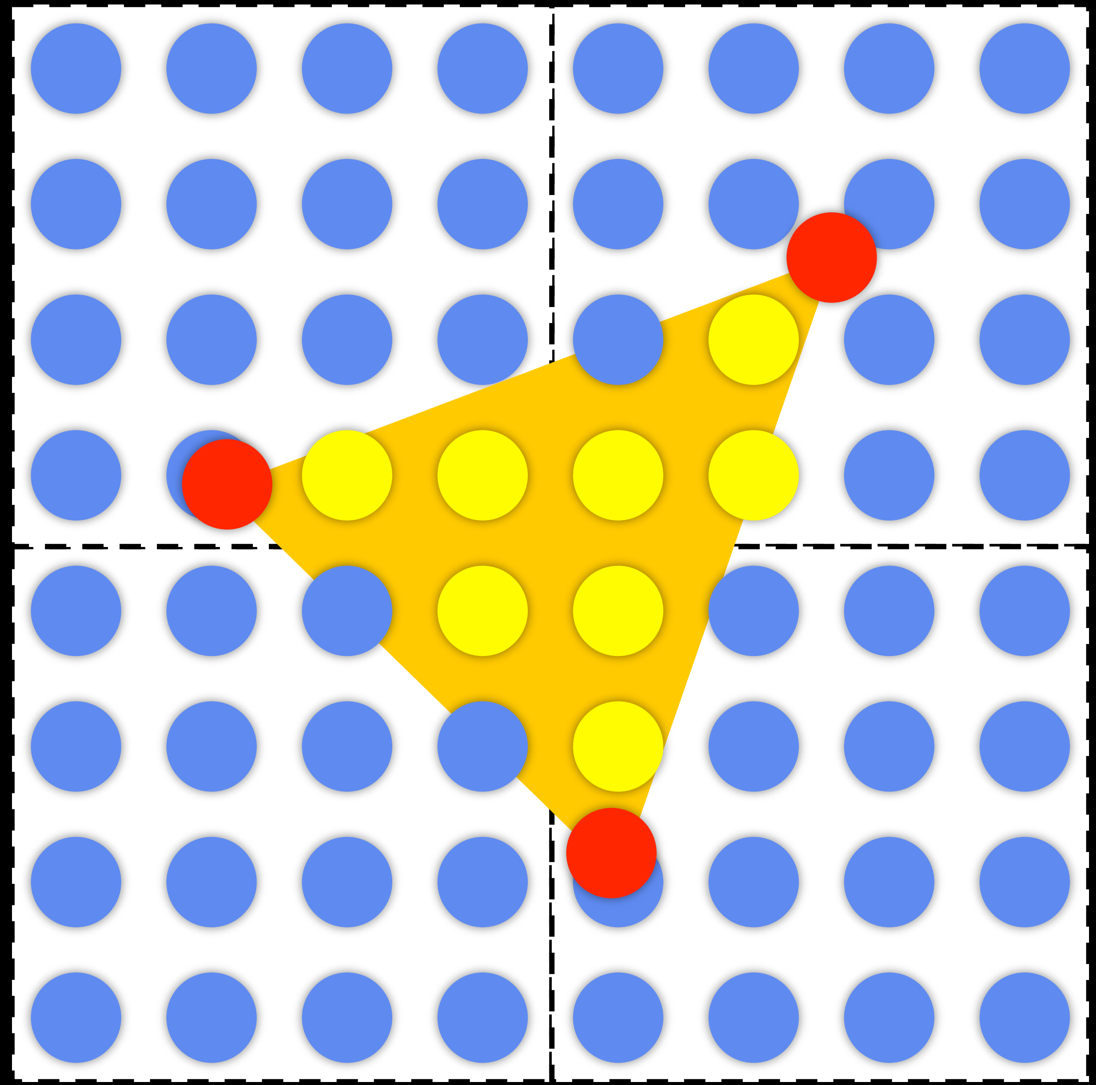
Geometry changes introduce high frequencies

Previous work

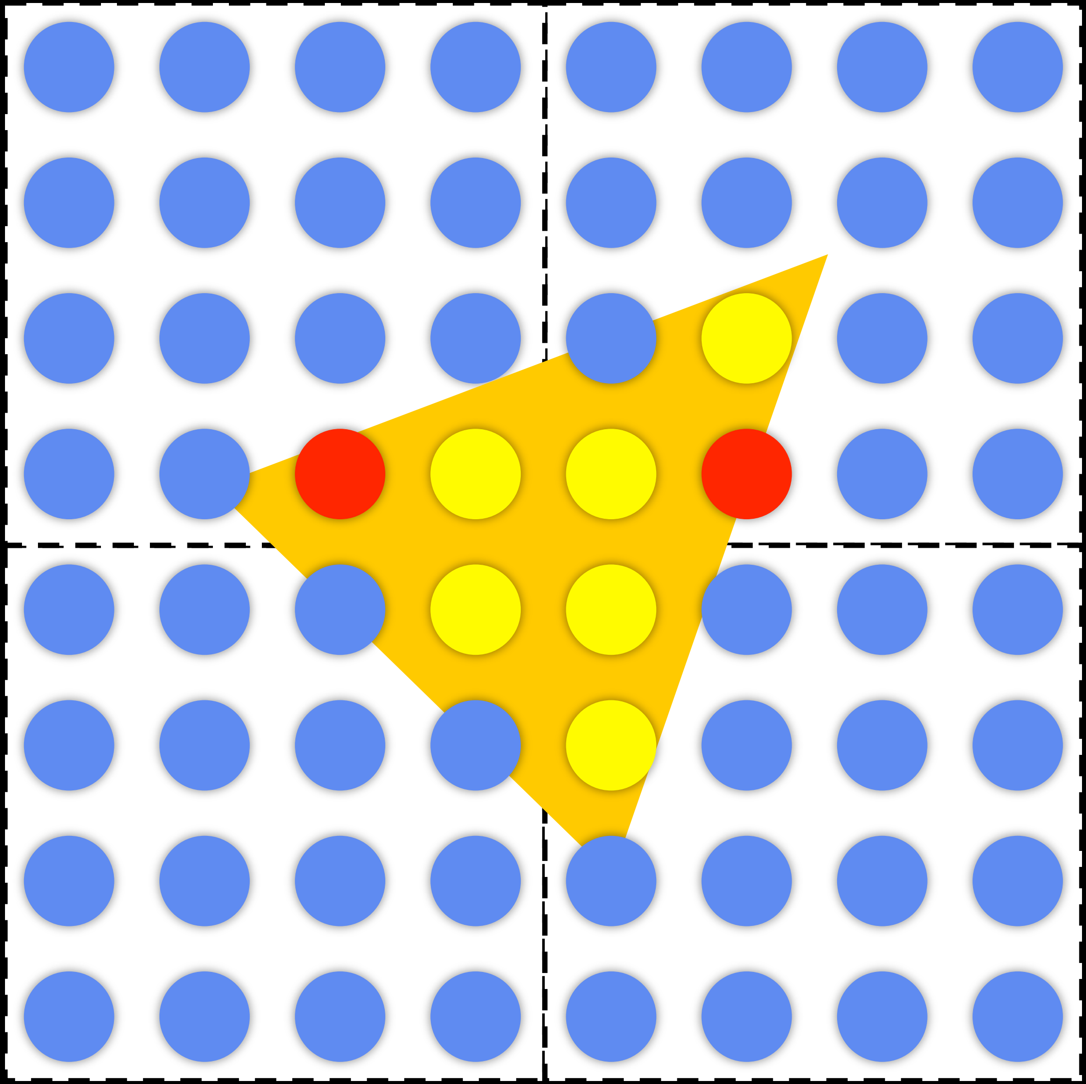
- Several algorithms use the fact that shading varies slower than geometry
  - REYES
  - Irradiance Caching
  - Upsampling for SSAO



REYES

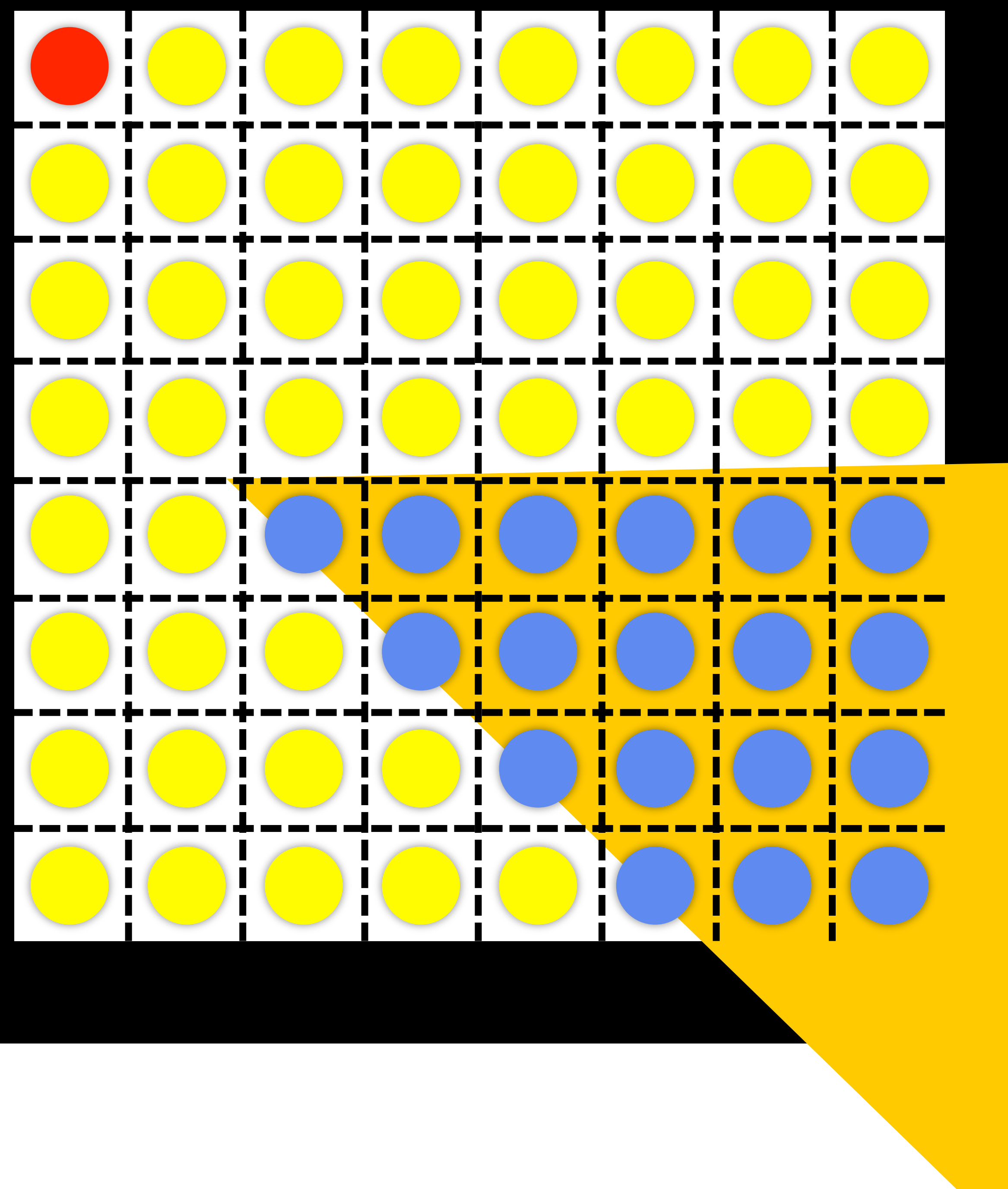


Irradiance caching

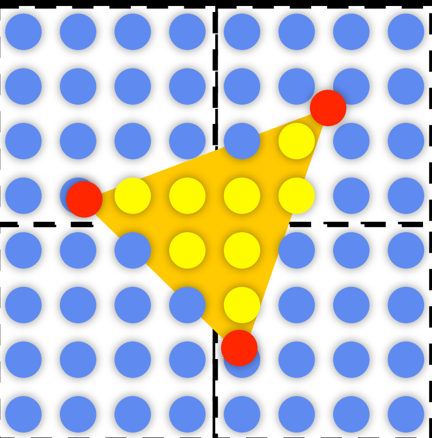


Upsampling

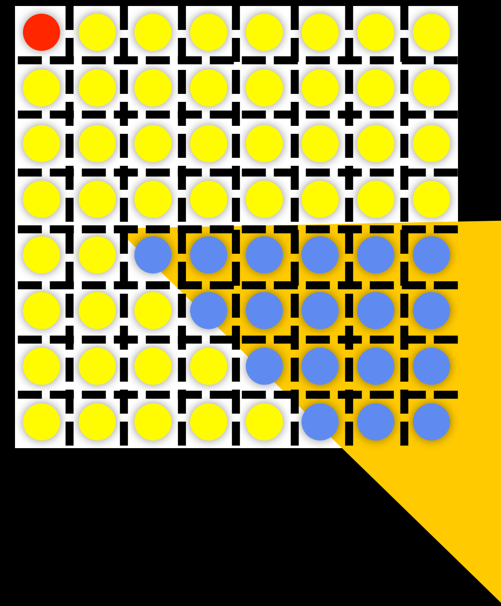
Great, but no AA



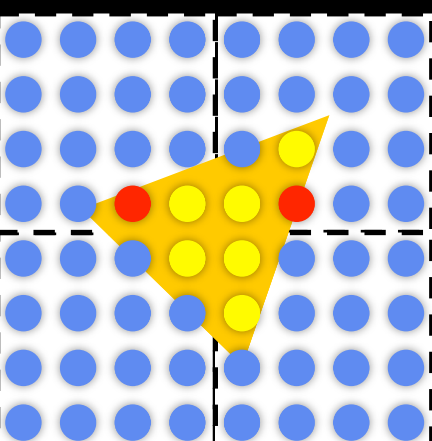
Previous work



REYES: Lots of geometry, not efficient on GPUs yet



Upsampling: Low frequency only

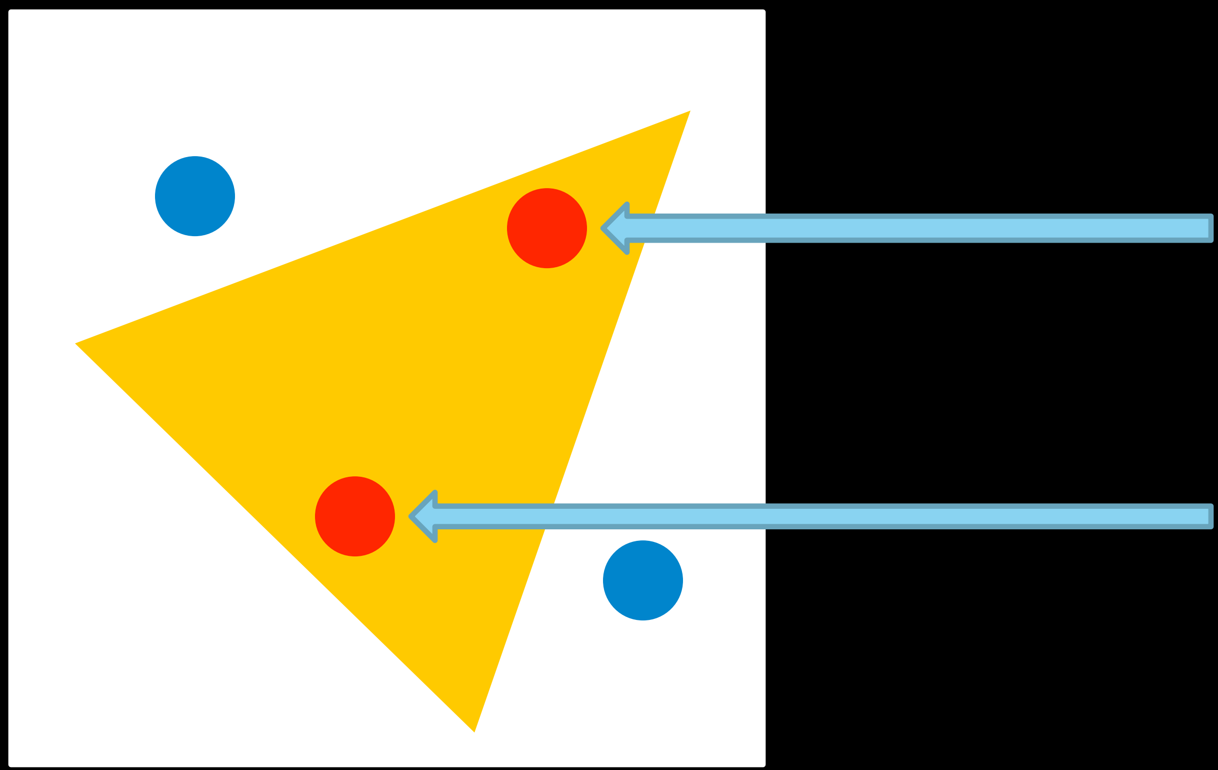


Irradiance Caching: Difficult to parallize well

Previous work

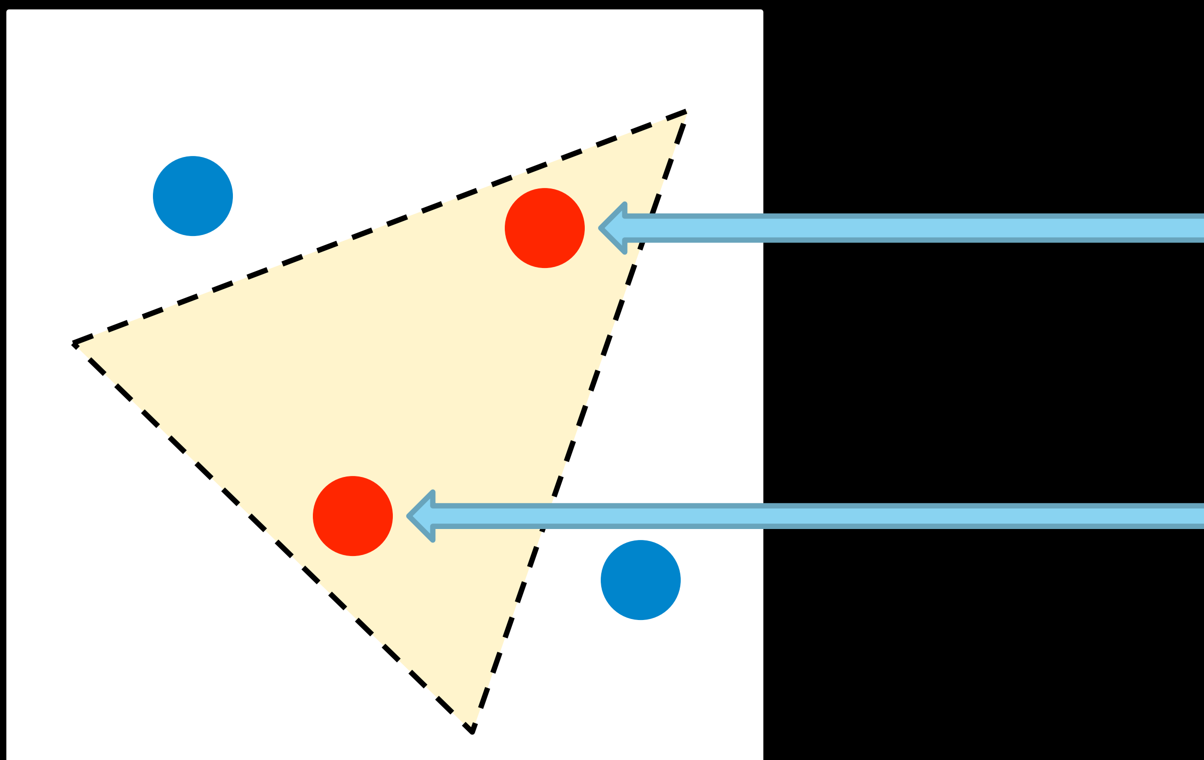
- Interpolate in world-space and check if the source/target locations are similar enough
- In screen-space, use a cross-bilateral filter when upsampling
  - Guarantees that shading does not get smeared across geometry edges
  - Requires geometric information
- Cross-bilateral interpolation is equivalent to the error metric in irradiance cache
  - Only use a sample if source “location” is similar to target
- Upsample low-frequency information like SSAO
  - Upsampling complex shading results usually in very blurry output

MSAA, CSAA



Shader runs only once!

MSAA is incompatible with deferred shading

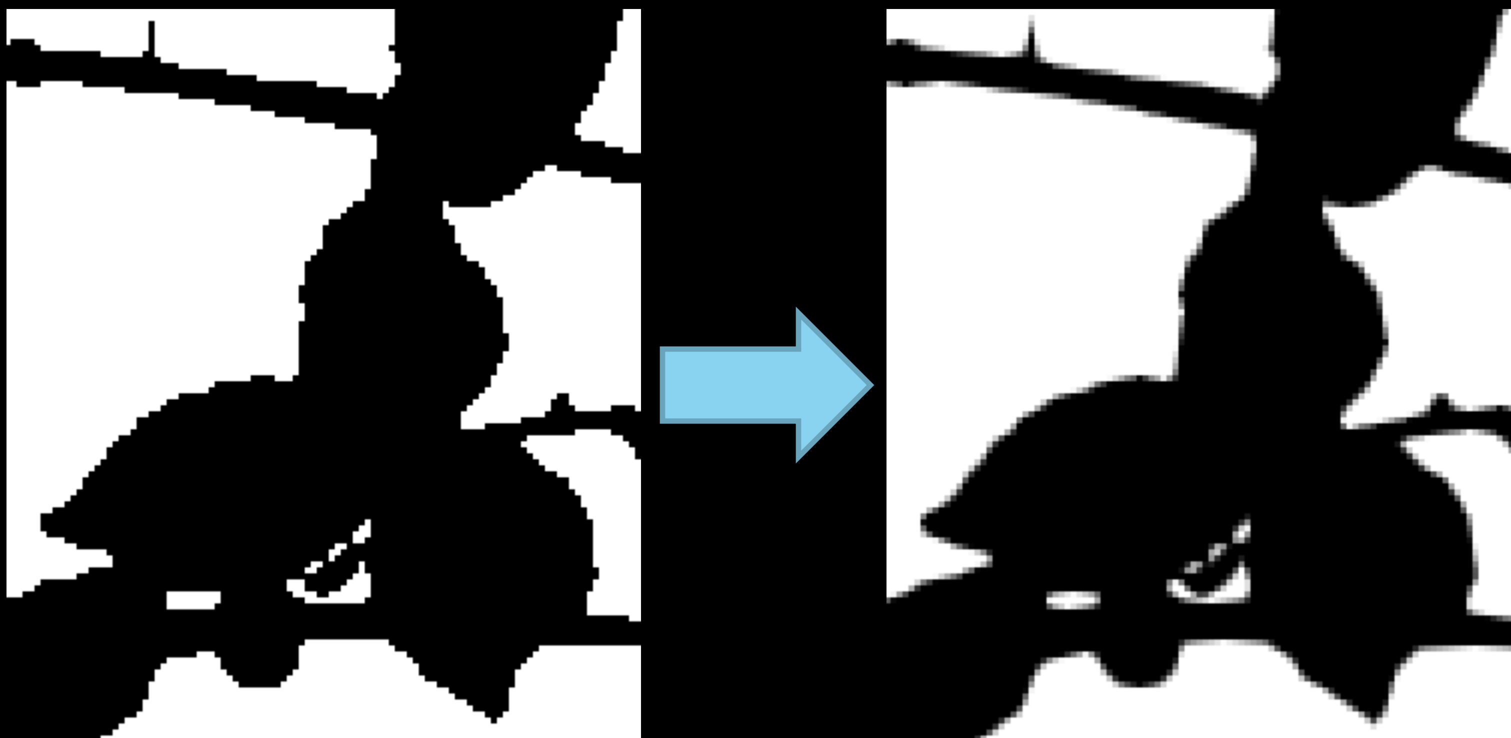


Shader must
run per-sampl

MSAA vs. deferred rendering

- MSAA: Great technique to get anti-aliasing without super-sampling the shading
  - Shade each primitive once per pixel, independent of sample count
- Deferred rendering
  - Has to shade all incoming samples
  - No efficient way to reconstruct which samples come from the same primitive
  - With MSAA, deferred shading degenerates to SSAA (!)
  - Stencil mask tricks work against warp-packing and don’t solve all issues

MLAA

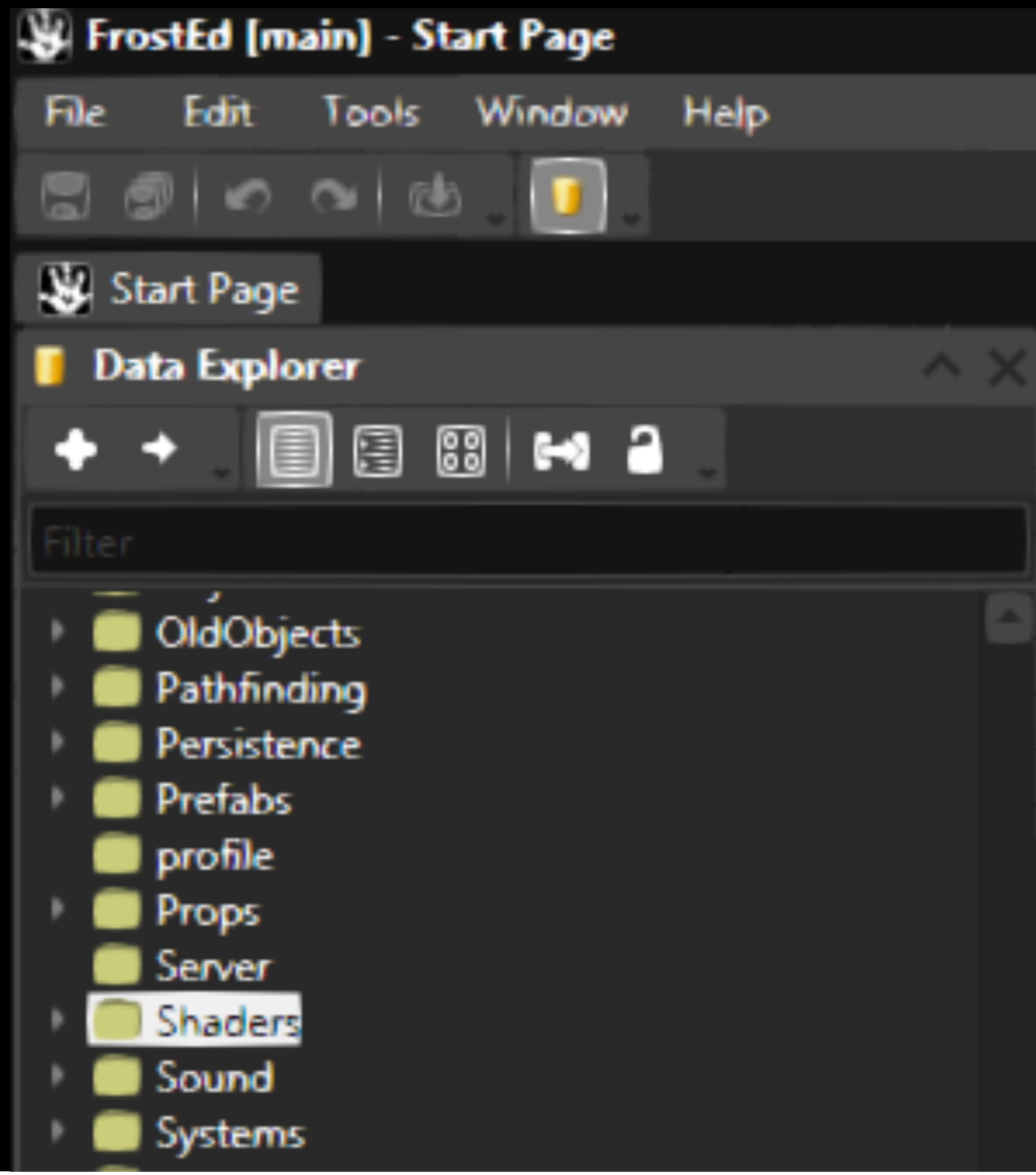


MLAA

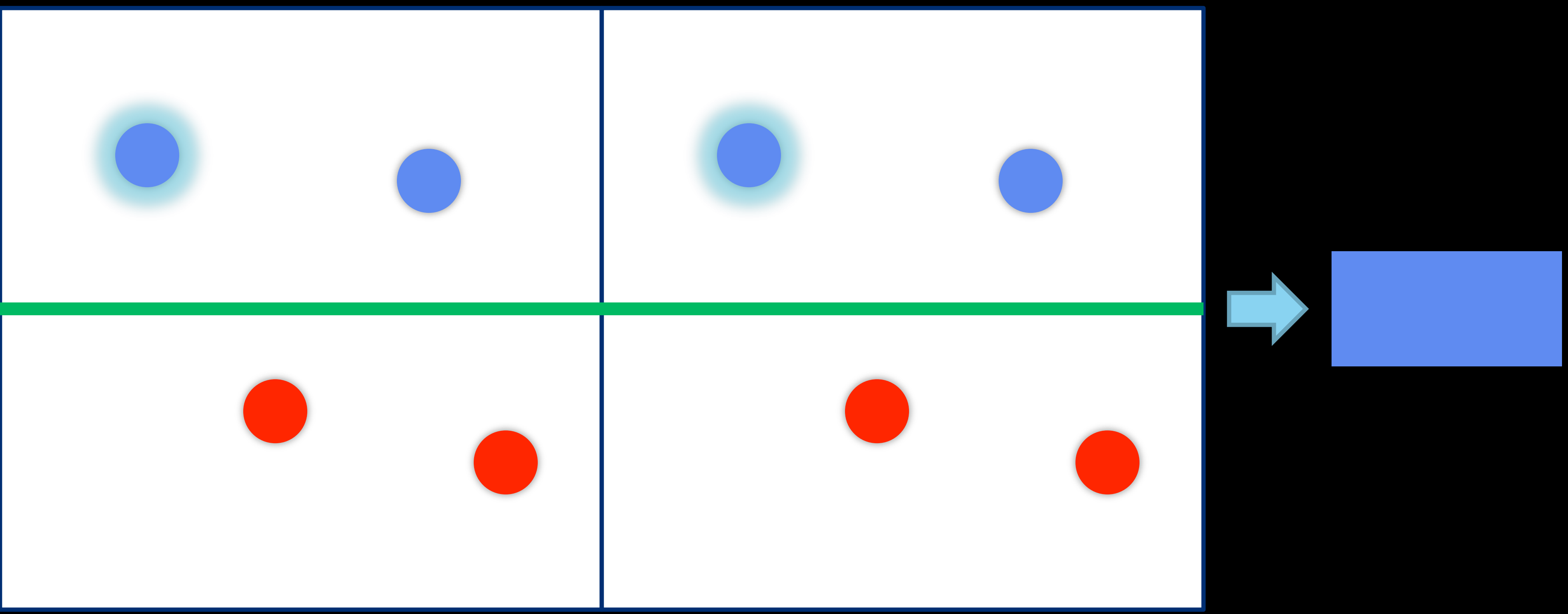
- Pure post-process
- Analyses the image content and blurs if something edge-like is found
  - Finds geometric and shading edges!
  - Text usually suffers worst (no information that this area should be excluded)
  - Runtime depends on edge count: Even though strictly a post-process, the runtime cannot be bound easily (x5 between best/worst case is common)!
- Can be easily used on any kind of pipeline
- Has some artifacts

MLAA vs. Text

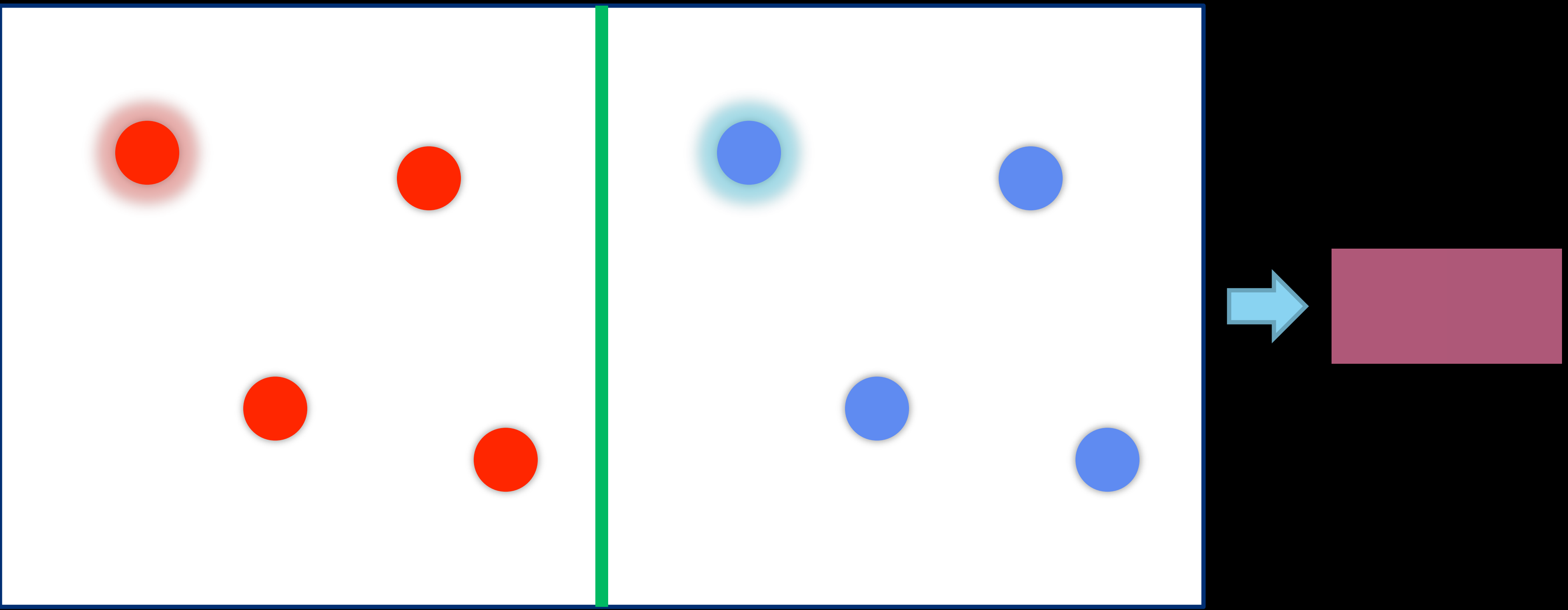
Eww, enabled AMD's driver-based GPU MLAA filter and it attacked our innocent 'FrostEd' editor that uses WPF



MLAA



MLAA



SRAA

- Unlike MLAA, SRAA knows where sub-pixel edges are
- Blur only where necessary

Reference, 16384x AA

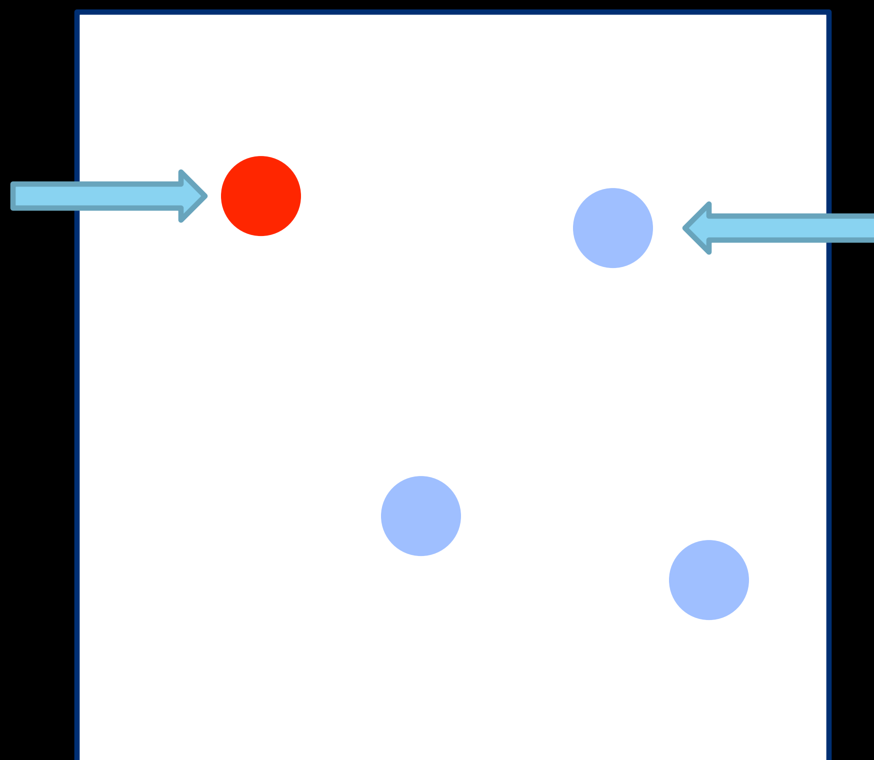
MLAA

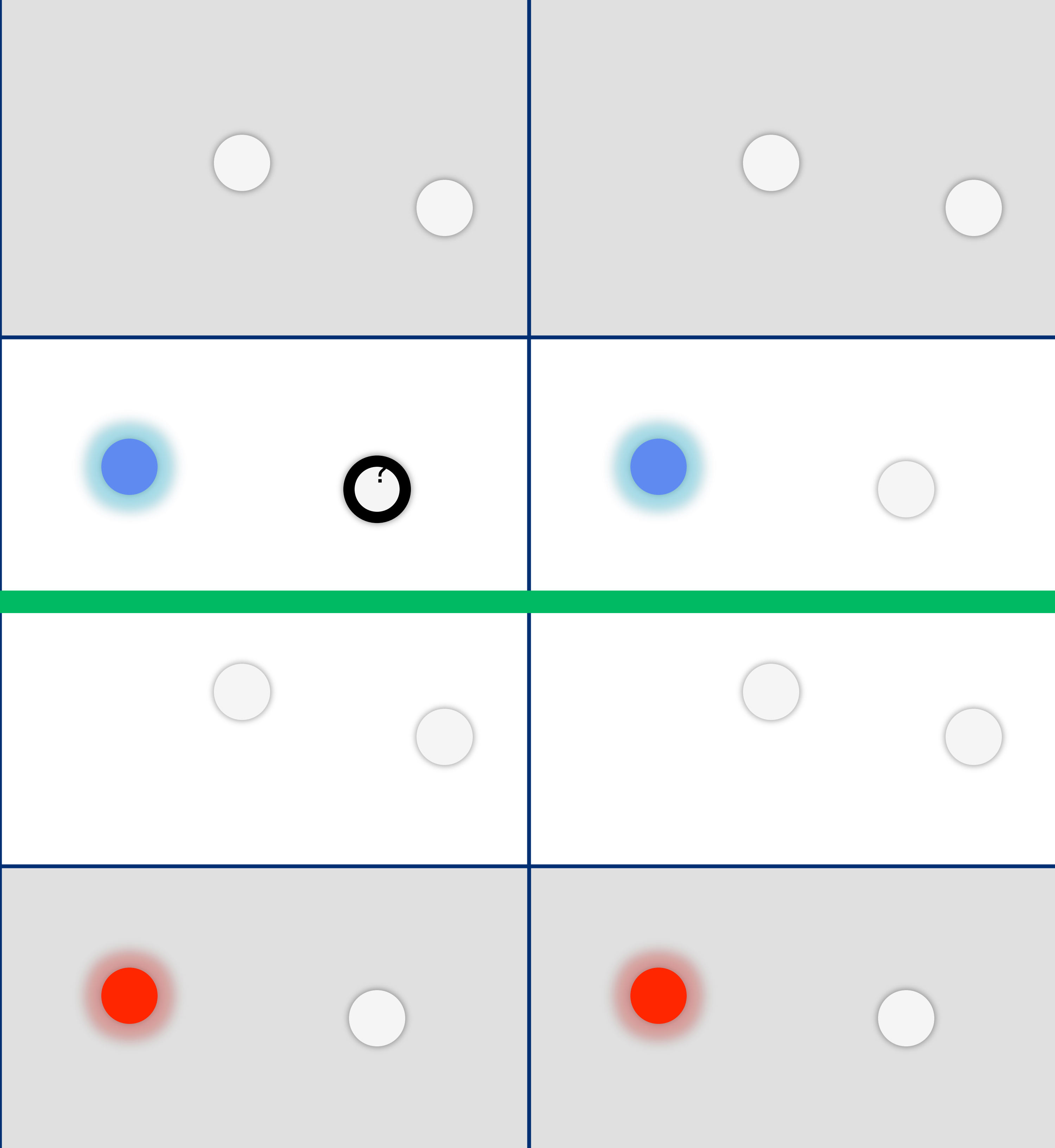
SRAA

SRAA

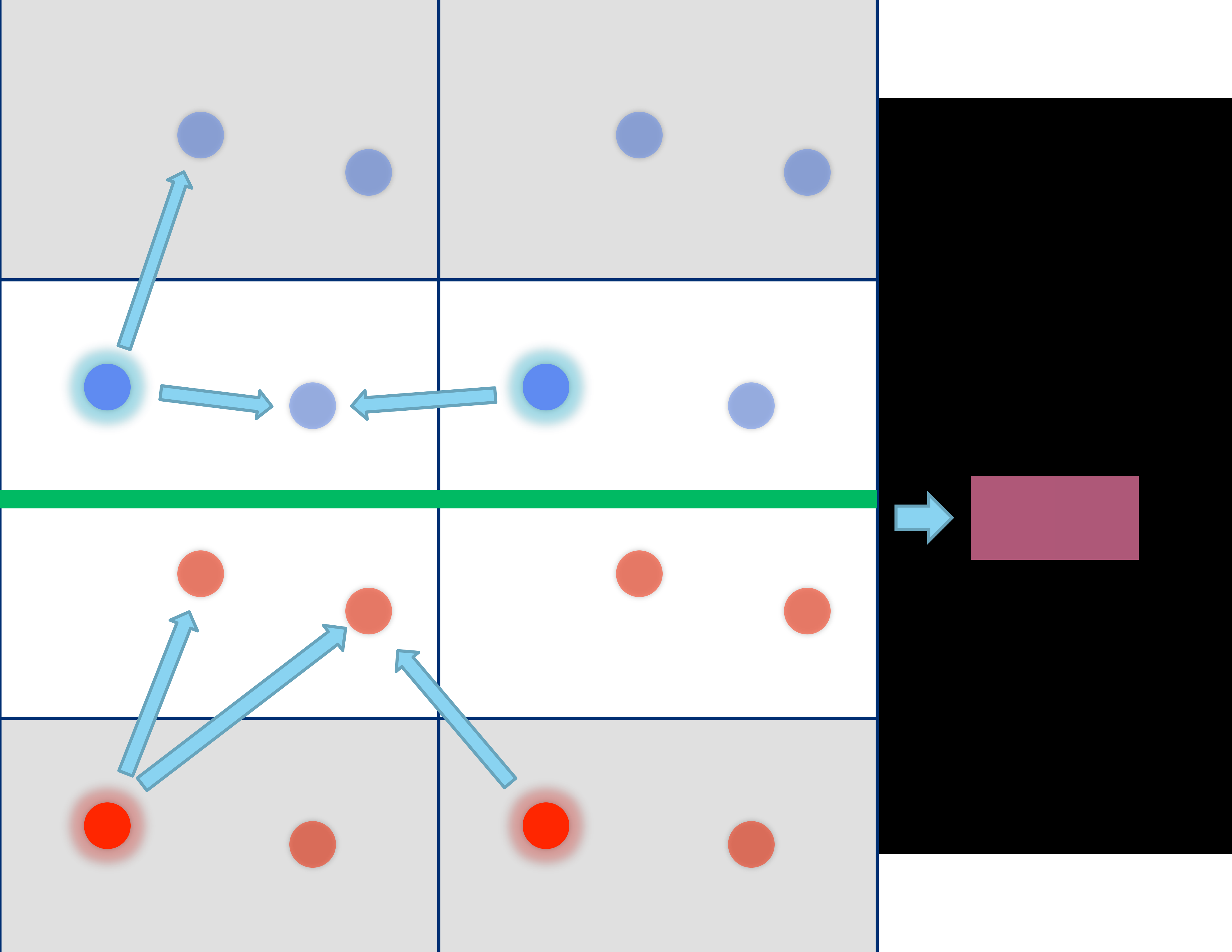
- Capture shading and geometry information at different frequencies
  - Geometry information is comparatively cheap to get (MSAA'ed G-Buffer has very little overhead)
  - Shading information is expensive (texture lookups, complex shaders, ray-tracing, you name it)
- Using high-frequency geometric information, try to estimate which shading sample corresponds to each geometric sample
- Works directly with MSAA
- Can be used with both deferred and forward rendering

SRAA

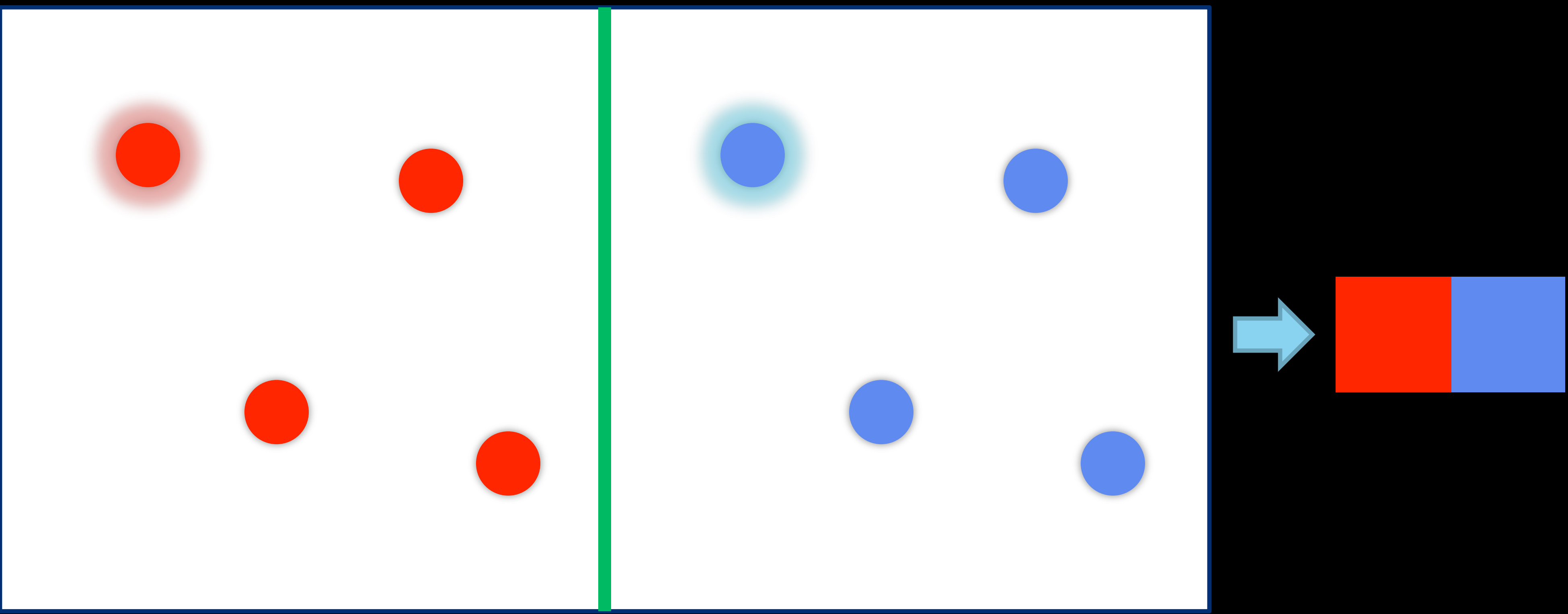




Like
Like
Like
Social netwoRk
Anti-Aliasing



SRAA



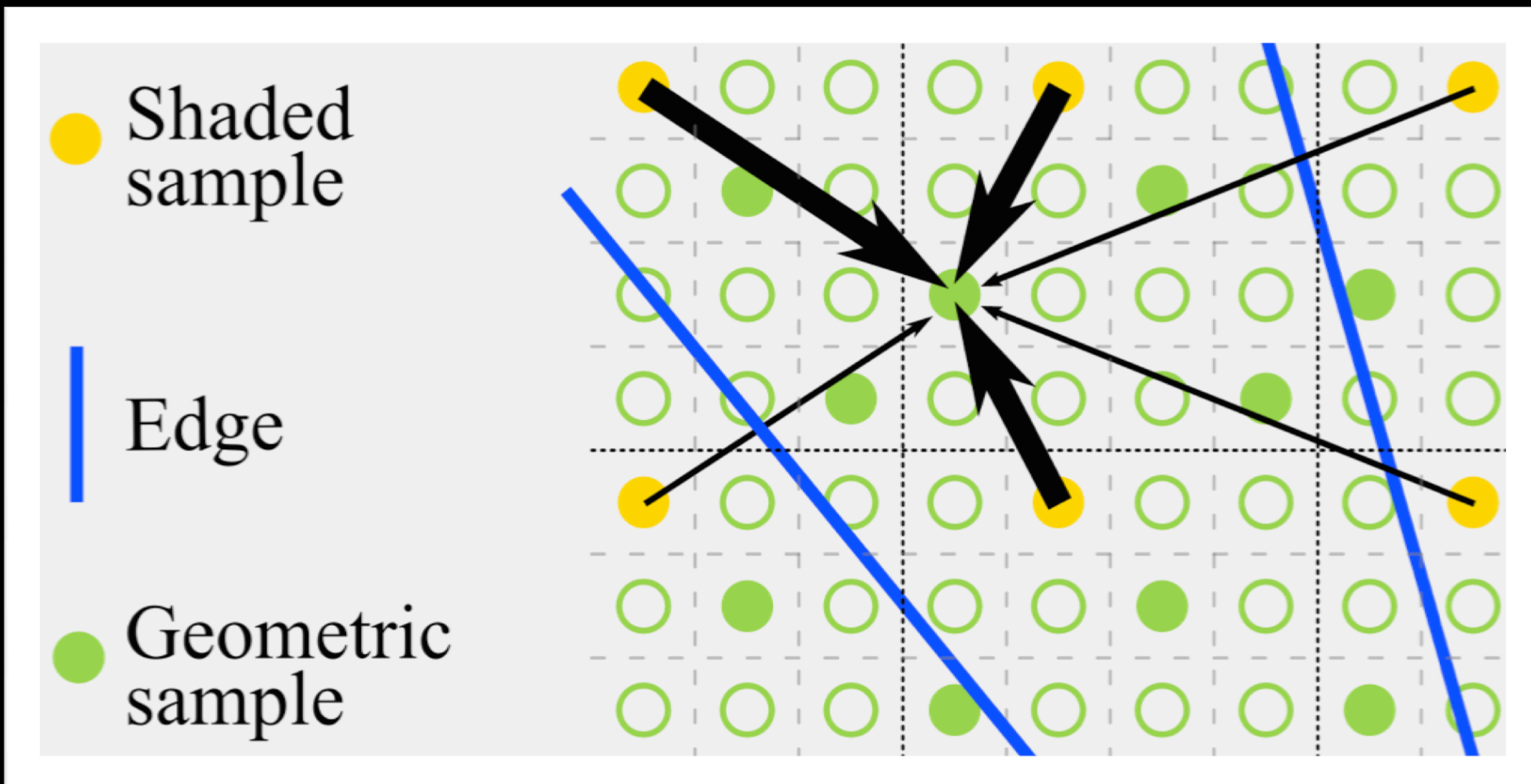
SRAA

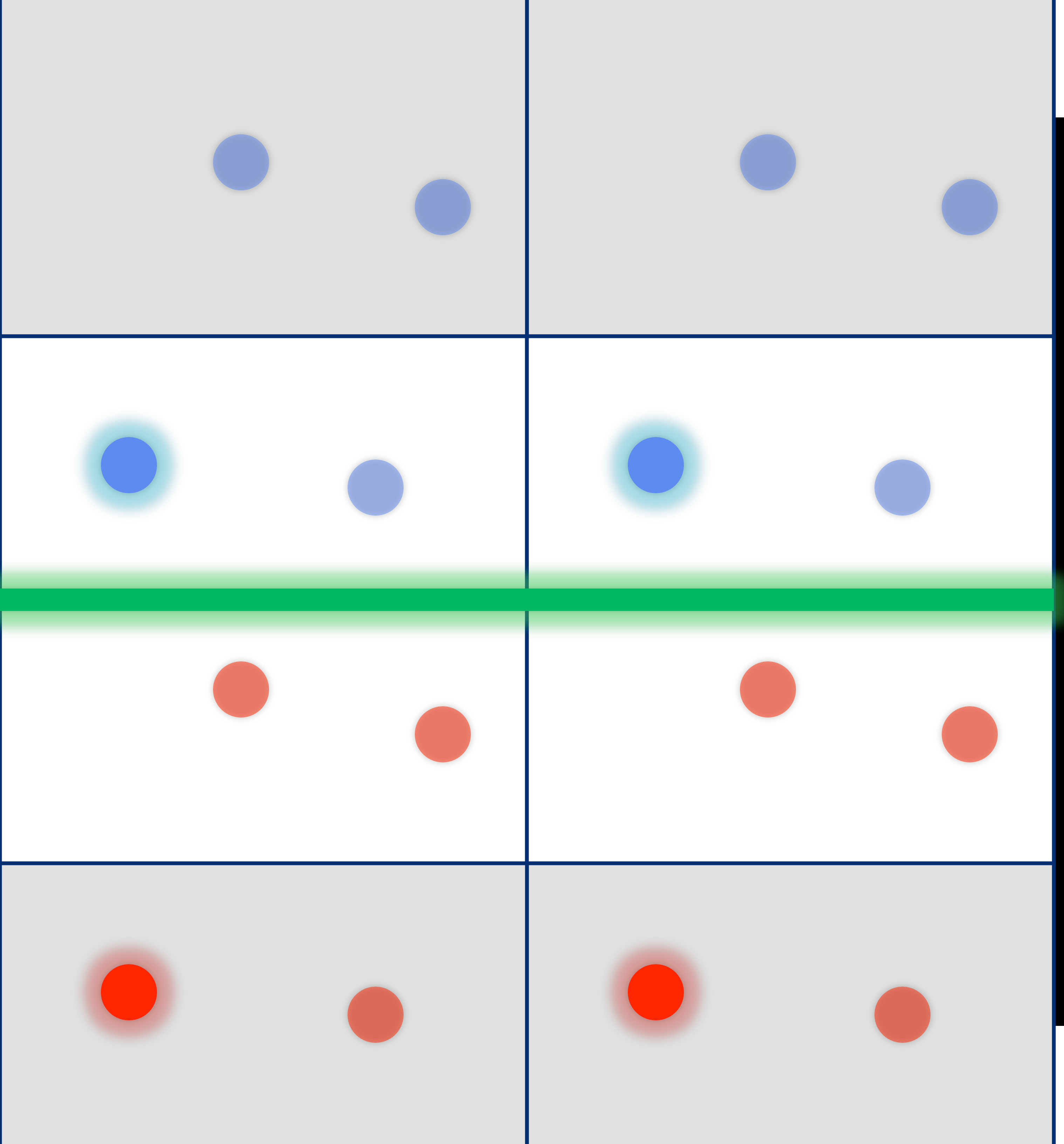
- We introduce geometry and shading samples
- A pixel can contain N geometry samples and M shading samples (M <= N)
- Geometry samples capture local surface properties: Position & Normal
- Shading samples capture color
- SRAA 4: N = 4, M = 1
- SRAA 16: N = 16, M = 1

SRAA

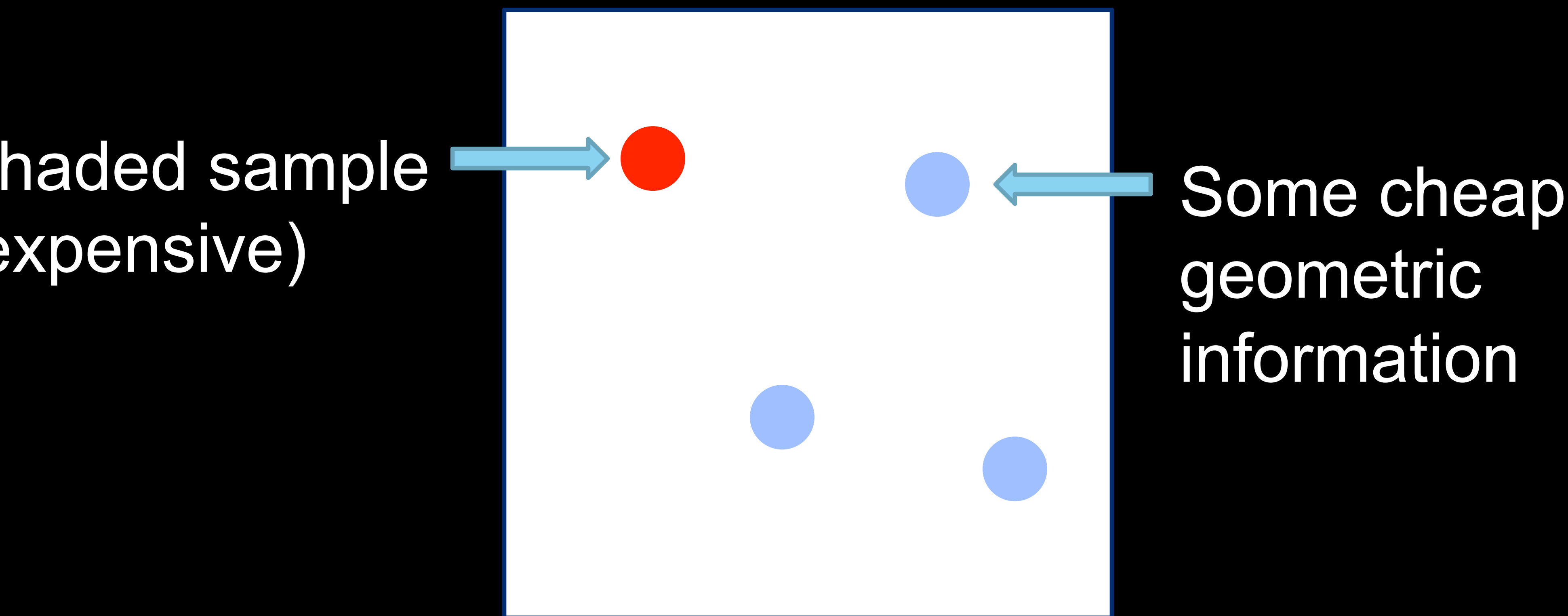
- Two pass algorithm
  - Render the depth/normals for the complete scene
- Shade a subset of the samples (typically, only the first)
- Run the SRAA filter which combines the MSAA'ed depth/normals with the shaded data
- Post-process the data as usual
- For deferred renderers, the only change is to generate the G-Buffers with MSAA
- For forward renderers, augment the z-Pre-Pass with normals

SRAA: Secret sauce





SRAA

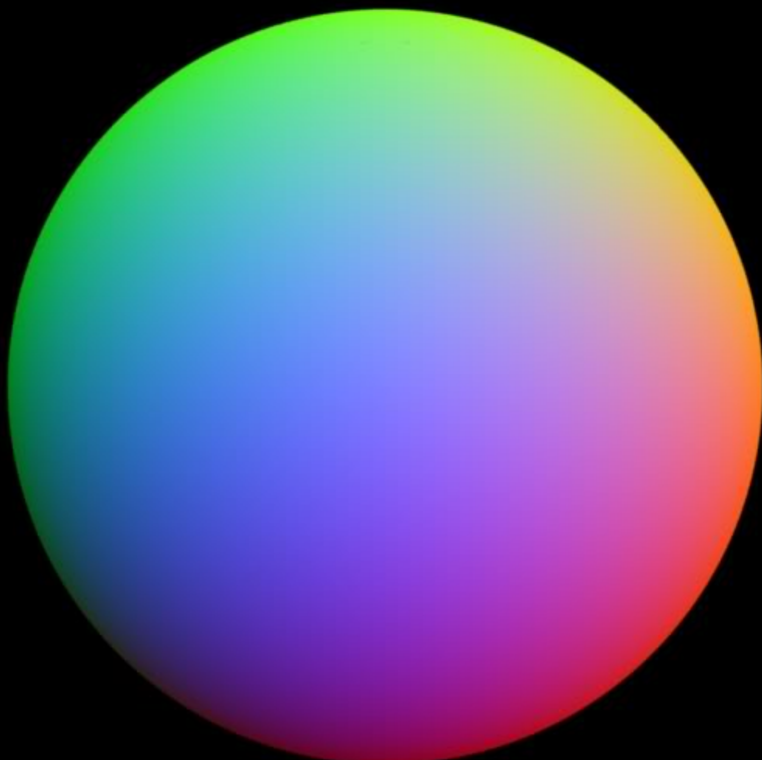


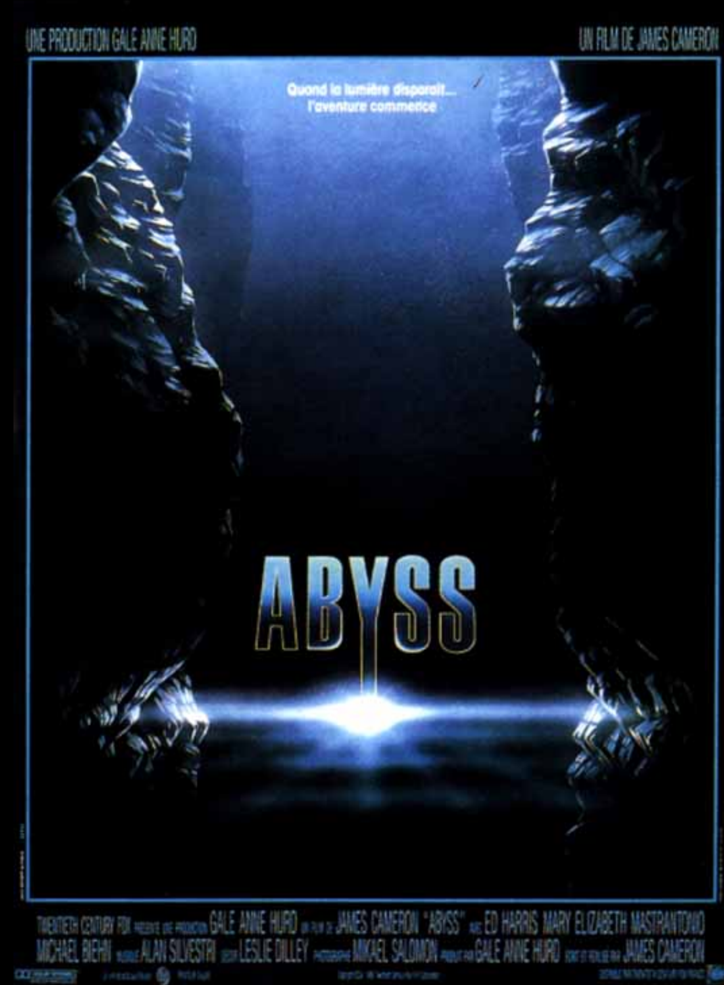
SRAA

- What‘s cheap?

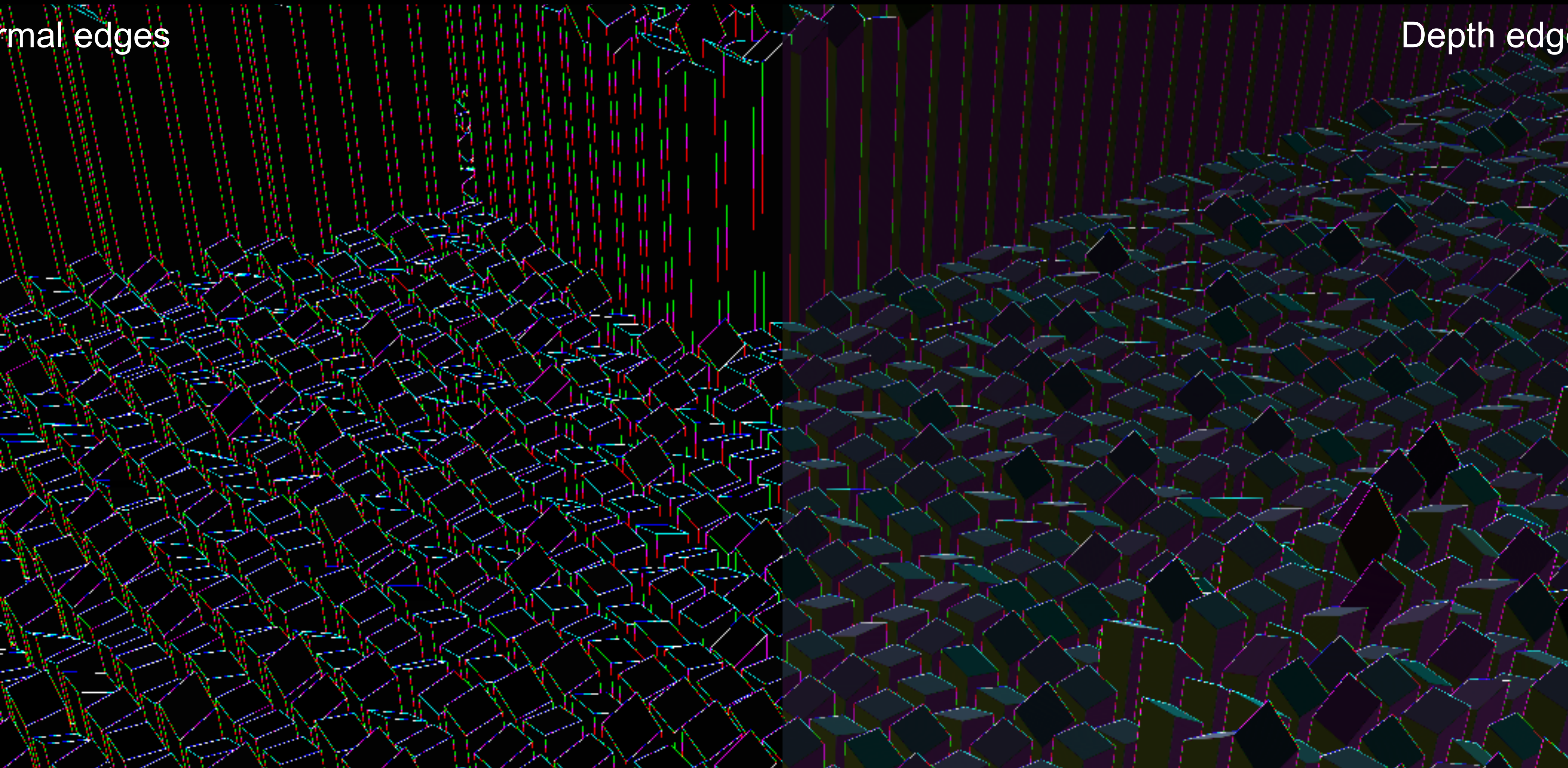
MSAA!

Cheap geometric information





SRAA: Edge detection



SRAA: Secret sauce

- Magic happens in SRAA kernel
- Looks at every geometric sample in a pixel, analyses all surrounding shaded samples
- Compute a weight for each shaded sample
- Reconstruct color for each geometric sample
- Box-Filter
  - Could use more advanced filters here!


Courtesy of DIC

SRAA, regular input

SRAA, output


Courtesy of DIC
SRAA, regular input


Courtesy of DIC
16x SSAA reference


Courtesy of DIC
SRAA, output


16x SSAA reference
Courtesy of DIC

SRAA, regular input

Courtesy of DIC

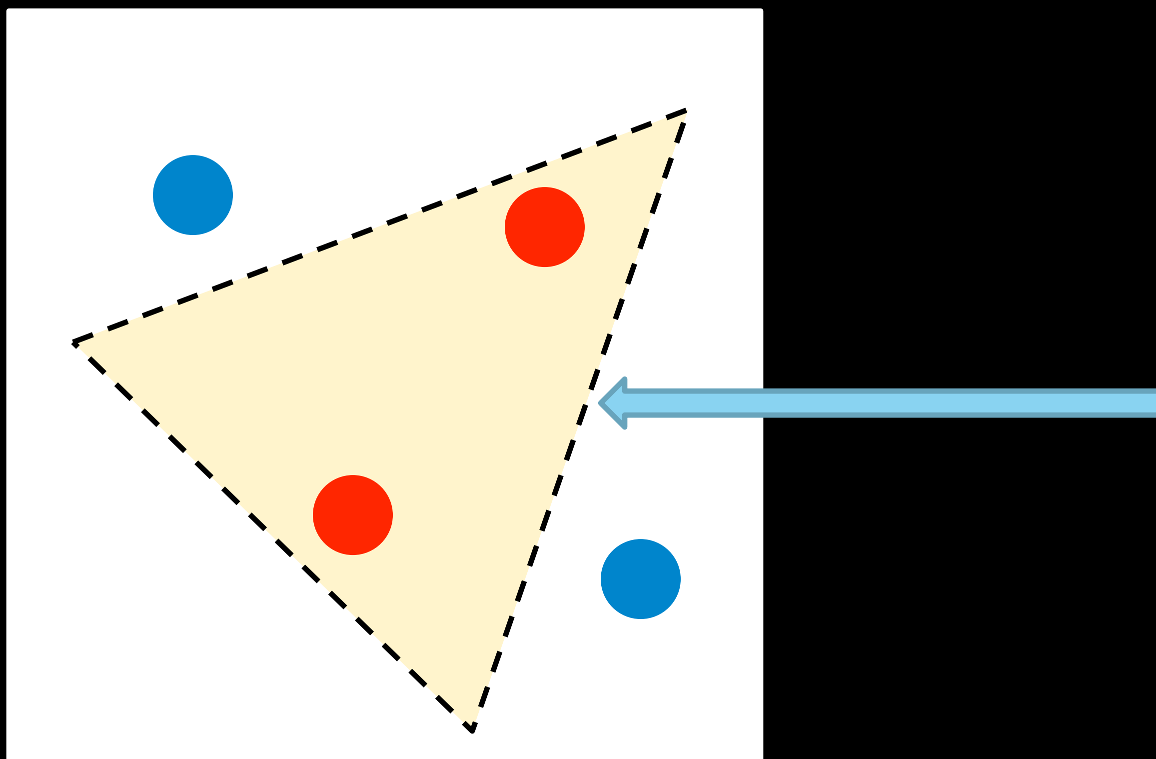

Courtesy of DIC
16x SSAA reference


SRAA, output
Courtesy of DIC

Performance

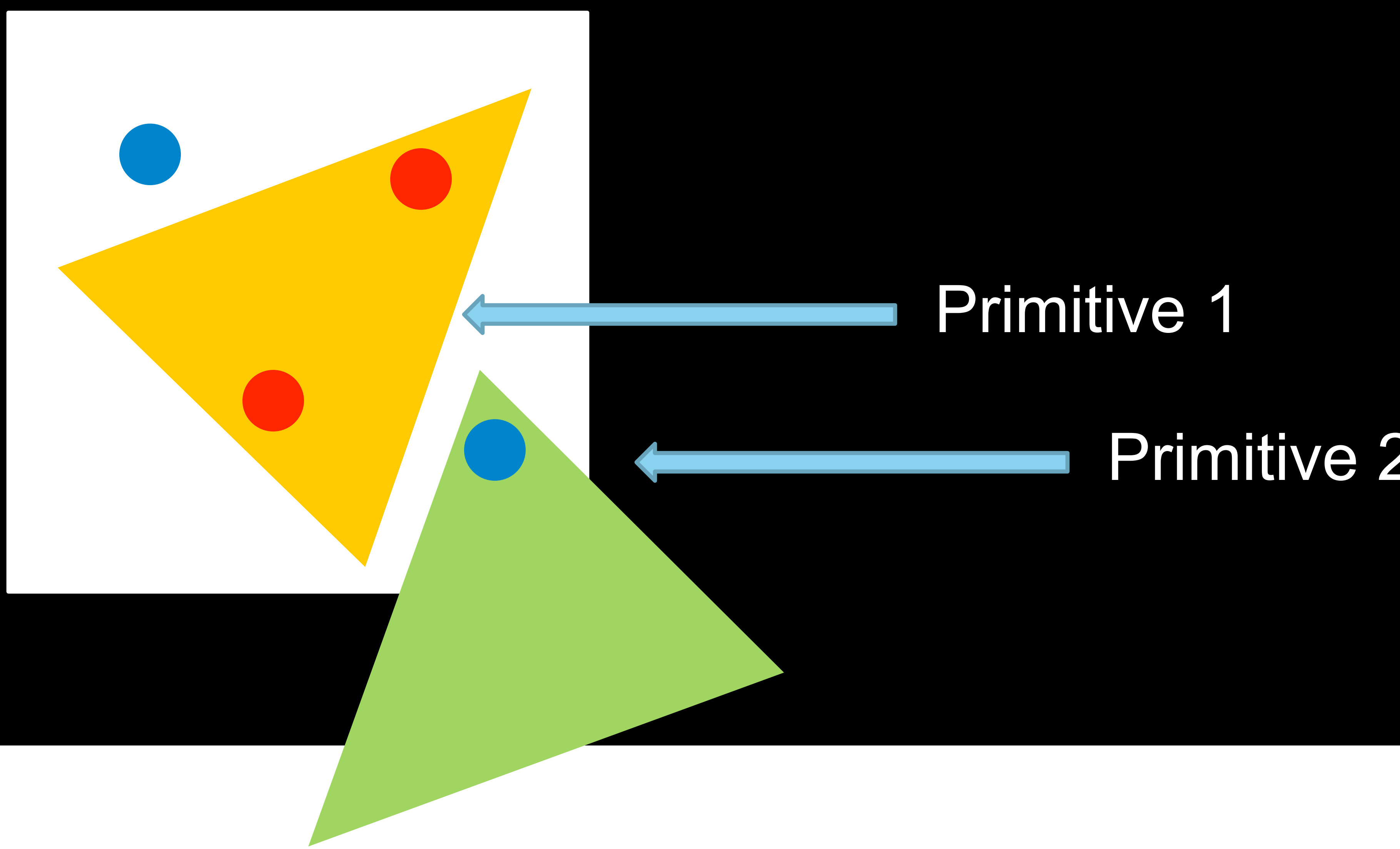
- High-quality with depth/normals, SRAA pass only
  - On a GTX 480, SRAA at 1920x1200 takes ~2 ms
  - On 1280x720, ~1 ms

MSAA vs. deferred shading



All we want to know is which samples belong together

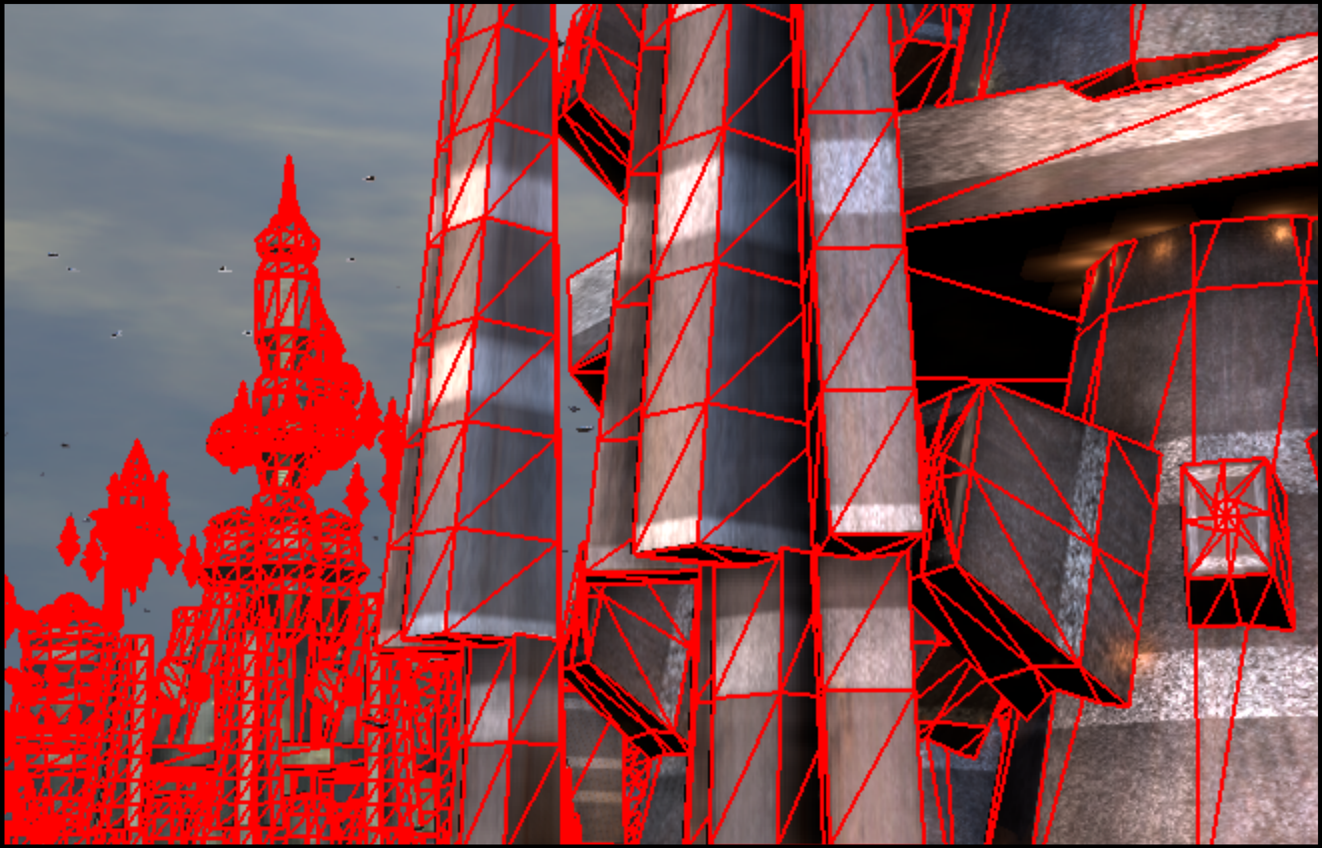
Geometry samples



Geometry samples

- We use the geometry samples to reconstruct surfaces
- Ideally, we want triangle Ids with adjacency information …
  - That‘s what MSAA computes actually, but doesn‘t give us access to
- Can use basically anything as „geometry samples“ as long as it changes at geometry edges

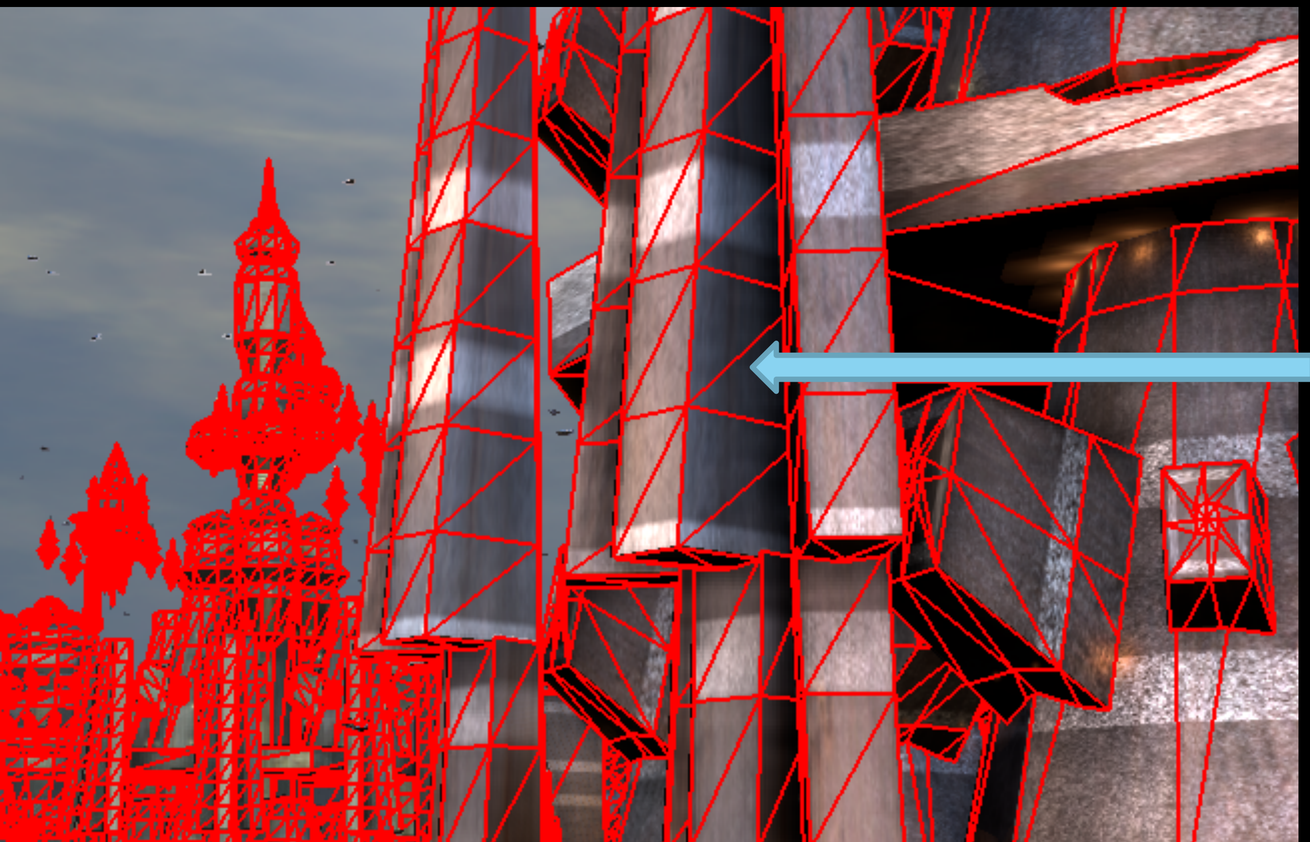
SRAA PrimitiveID



Performance

- High-quality with depth/normals, SRAA pass only
  - On a GTX 480, SRAA at 1920x1200 takes ~2 ms
  - On 1280x720, ~1 ms
- `SV_PrimitiveID`: 1 ms for 1920x1200 on a **GTX 460**
- Ready to deploy as DX11 pixel shader
  - Sample MSAA’ed depth/normal/primitive buffers
- MSAA makes the G-Buffer creation slightly more expensive

SRAA PrimitiveID



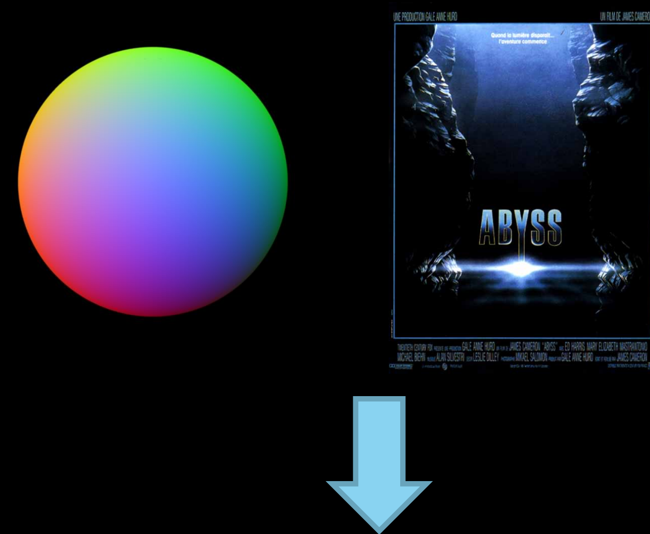
We actually don‘t want that one … ☹

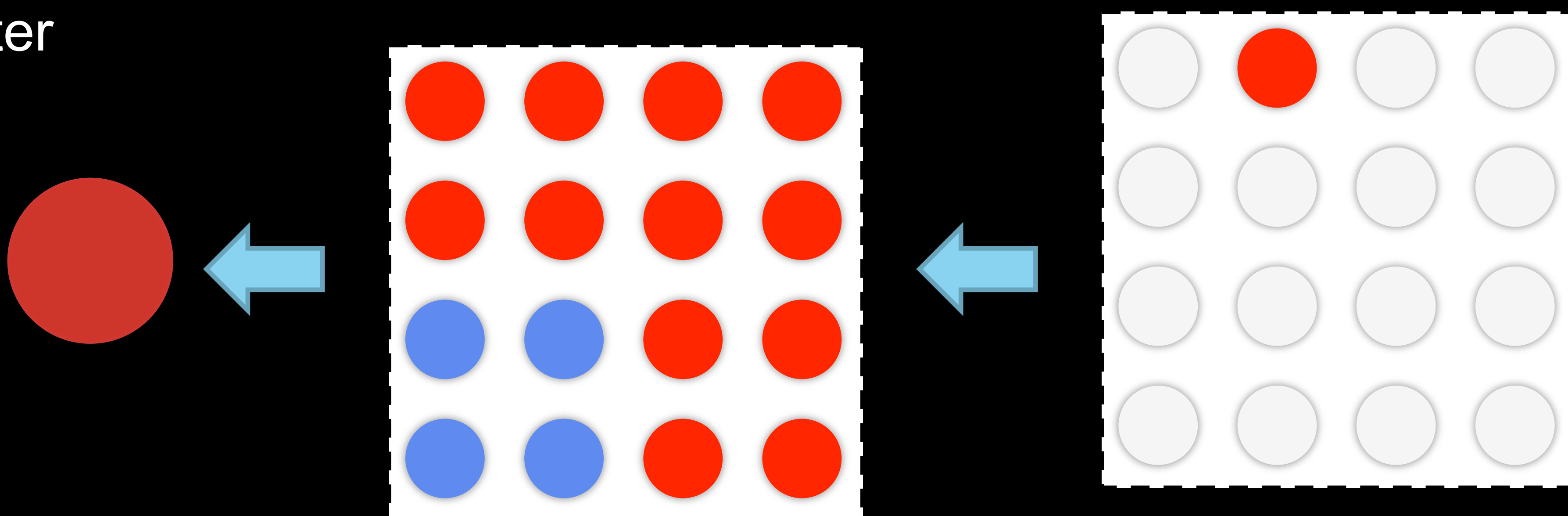
SRAA Optimisations

- Instead of using depth/normal to estimate discontinuities …
  - Use just depth
    - Finds most edges
    - Depending on the depth range, can work with 8 bit depth buffer (See Crysis 2 images in paper)
  - Use an object/primitive ID
    - `SV_PrimitiveID` does the job quite well, hash it to 8 bit
    - SRAA becomes very similar to MSAA here!
  - Any other source of discontinuities
    - Material IDs
    - UVs
    - …

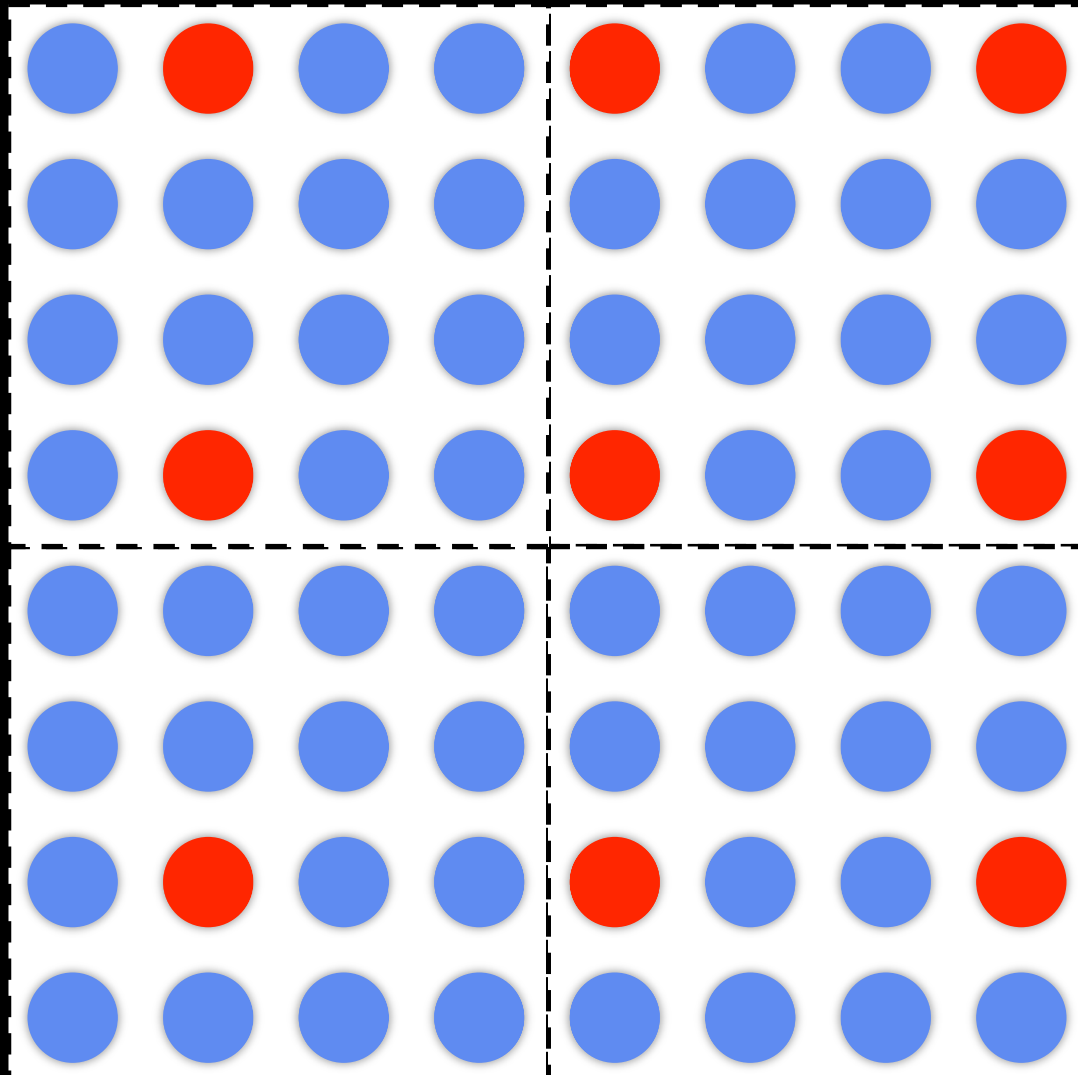
Recap

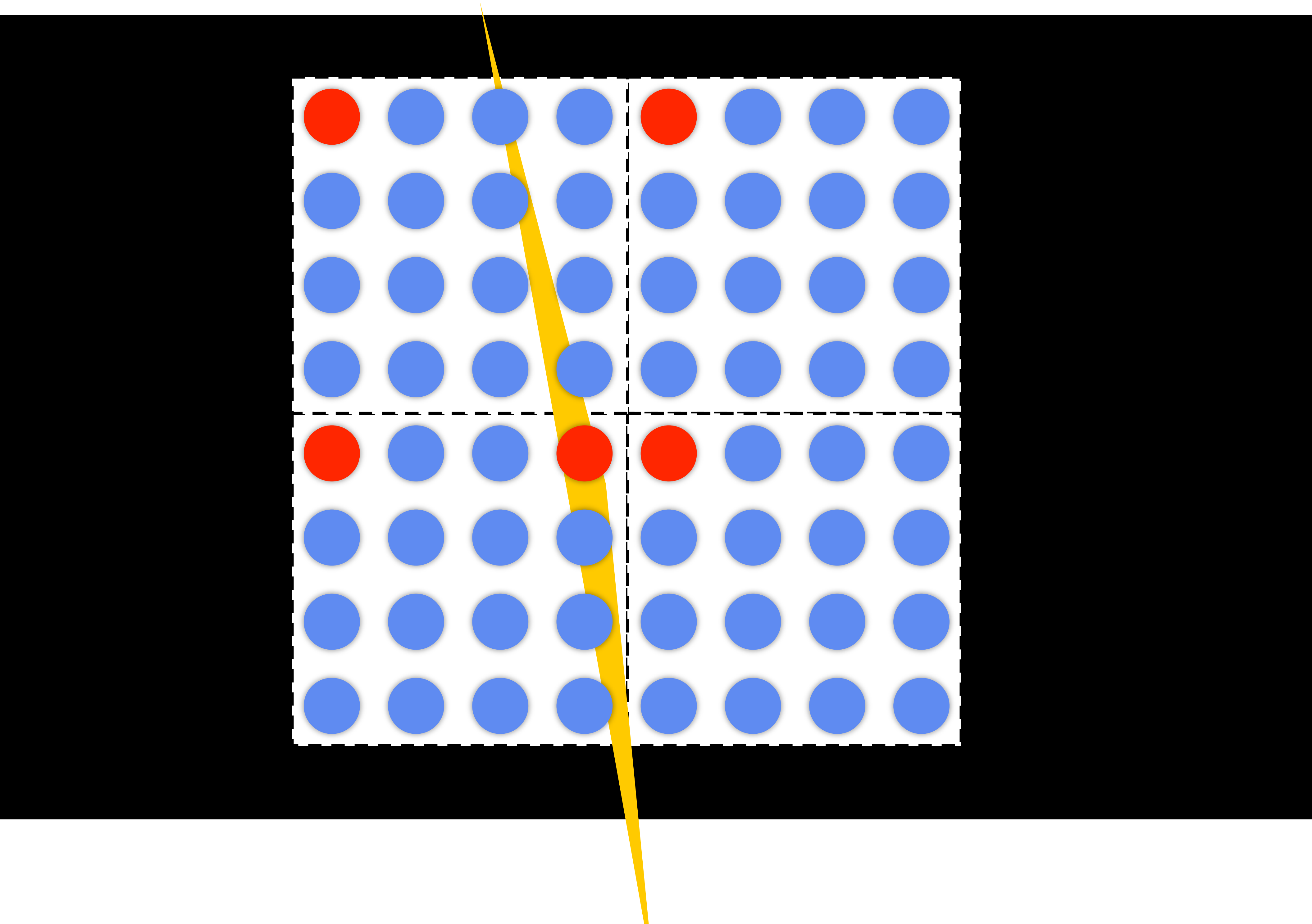
1. Generate MSAA Depth/Normal
2. Shade a subset of all samples
    - Forward or deferred!
3. Reconstruct per-sample color
4. Filter






The Future
NEXT EXIT





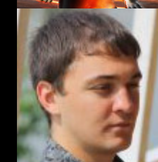
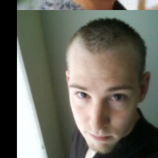
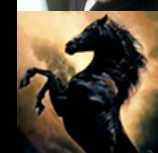
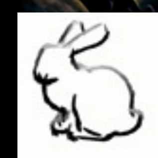
Future work

- Use SRAA to guide MLAA
  - Help MLAA to find all sub-pixel edges
  - Use MLAA to clean up after SRAA removed sub-pixel aliasing
- Investigate higher-quality modes
  - We have 1.5 shading samples at 16 geometry samples, which starts to look equal to 16x SSAA
  - Both are fully decoupled: Can shade any subset of the geometric samples
  - Only shade interesting samples
- Better edge finder
  - Tessellation makes `SV_PrimitiveID` miss in-patch edges
  - Depth/Normal can fail if depth-range is extremely large

Thanks!

- Johan Andersson, DICE, @repi
- Anton Kaplanyan, Crytek, @tofic
- Timothy Lottes, NVIDIA, @TimothyLottes

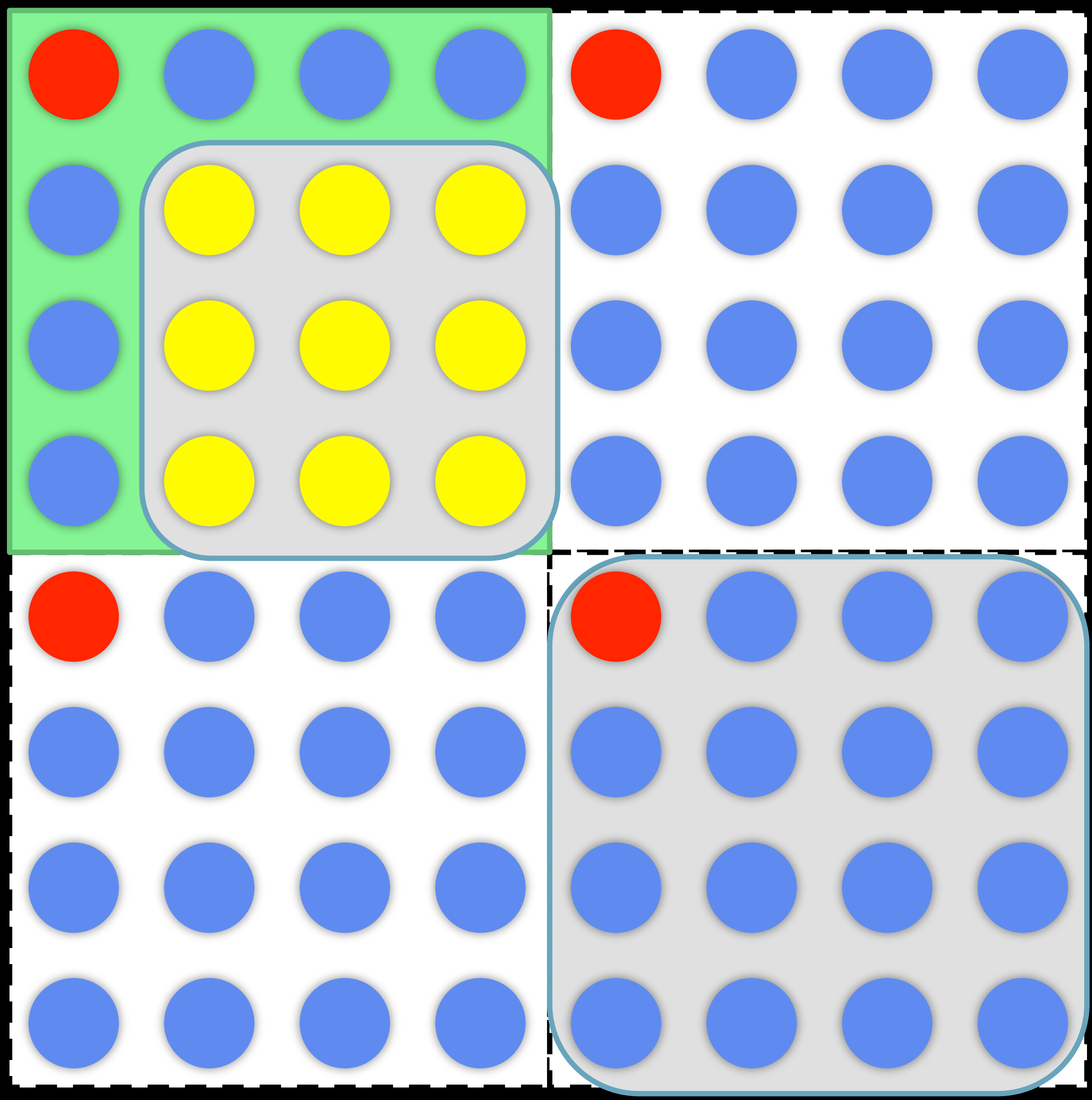
- Authors
  - Morgan McGuire, NVIDIA, @morgan3d
  - David Luebke, NVIDIA, @davedotlubke
  - Matthäus Chajdas, TUM, @NIV_Anteru



AA, 4x super-sampled G-Buffer, 8-bit depth
Questions?
Courtesy of Cry


AA, 4x super-sampled G-Buffer, 8-bit depth
Backup
Courtesy of Cry



Depth/Normal distance