Stochastic all the things: Raytracing in hybrid real-time rendering

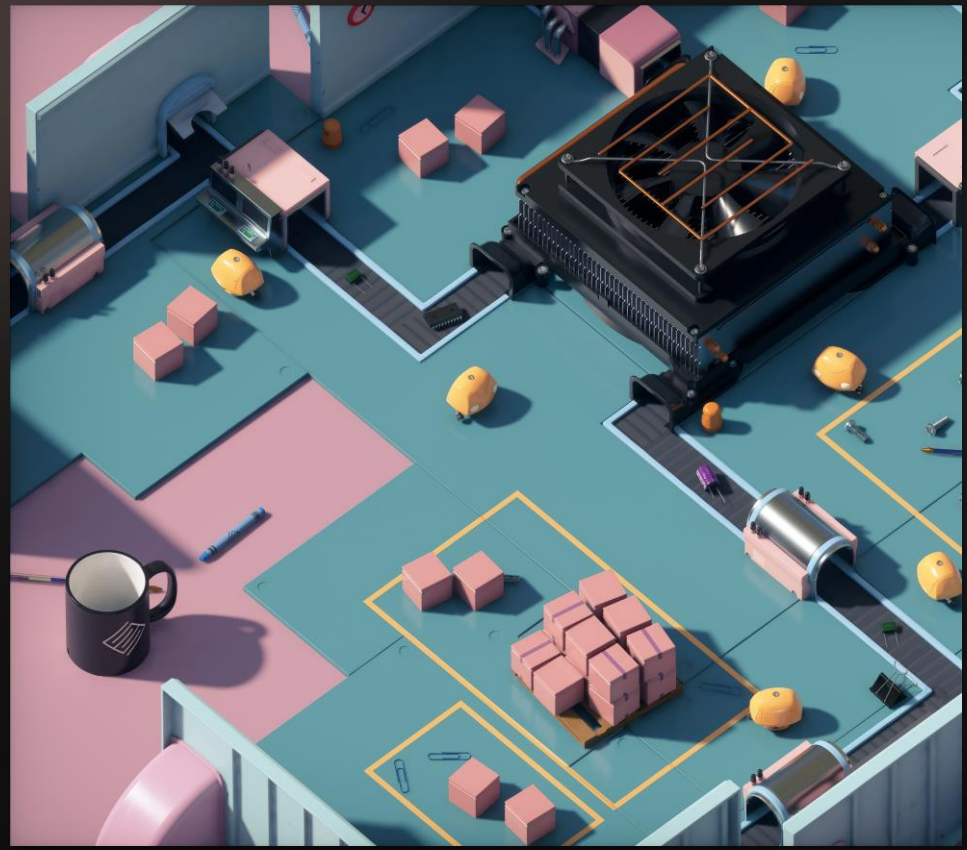
Tomasz Stachowiak
SEED – Electronic Arts

SEED

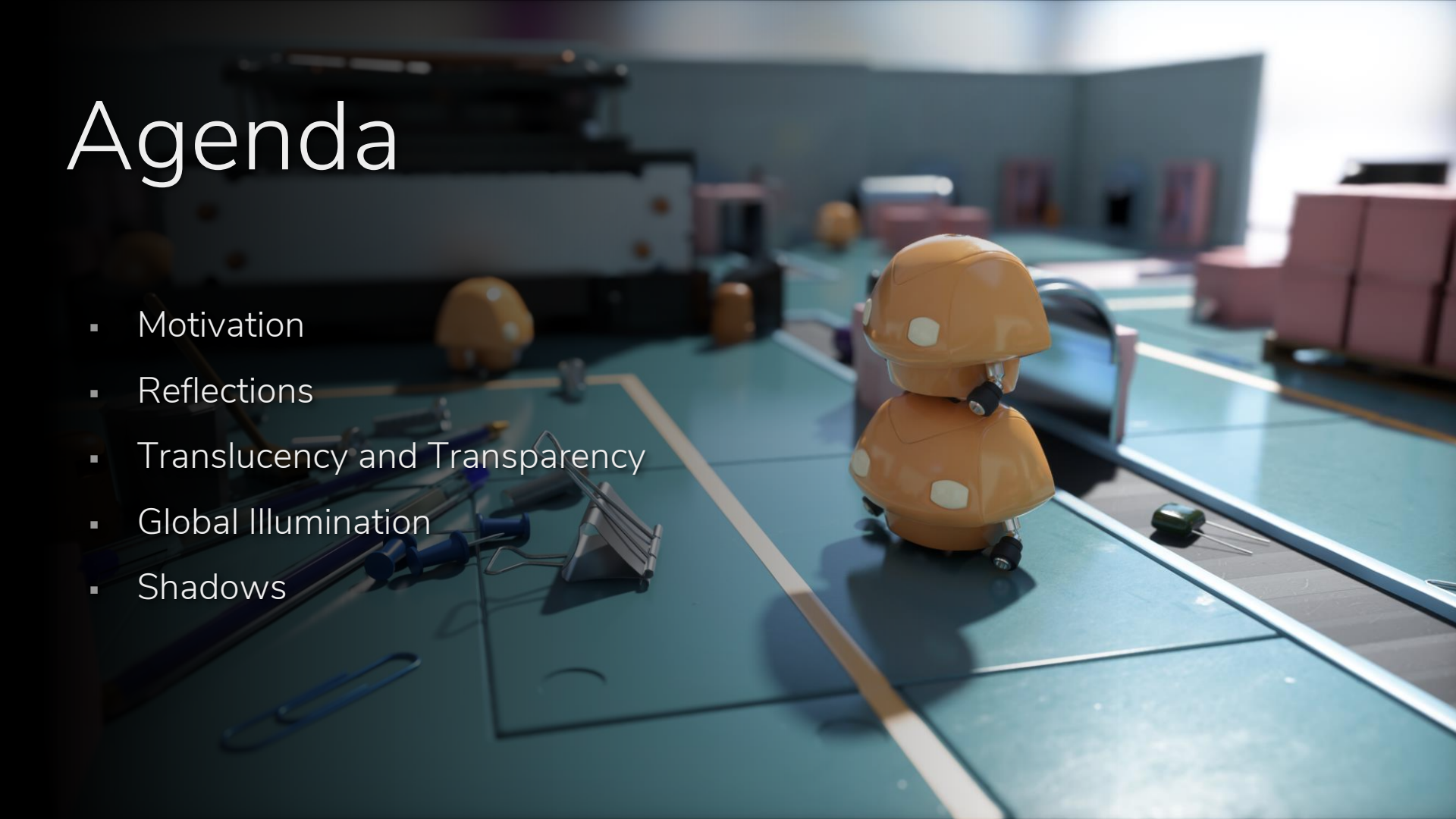
Watch the trailer here:
https://www.youtube.com/watch?v=LXo0WdlELJk

“PICA PICA”

- Exploratory mini-game & world
- For our self-learning AI agents to play, not for humans ☺
  - *“Imitation Learning with Concurrent Actions in 3D Games” [Harmer 2018]*
- Uses SEED’s **Halcyon** R&D engine
- Goals
  - Explore hybrid rendering with DXR
  - Clean and consistent visuals
  - Procedural worlds [Opara 2018]
  - No precomputation

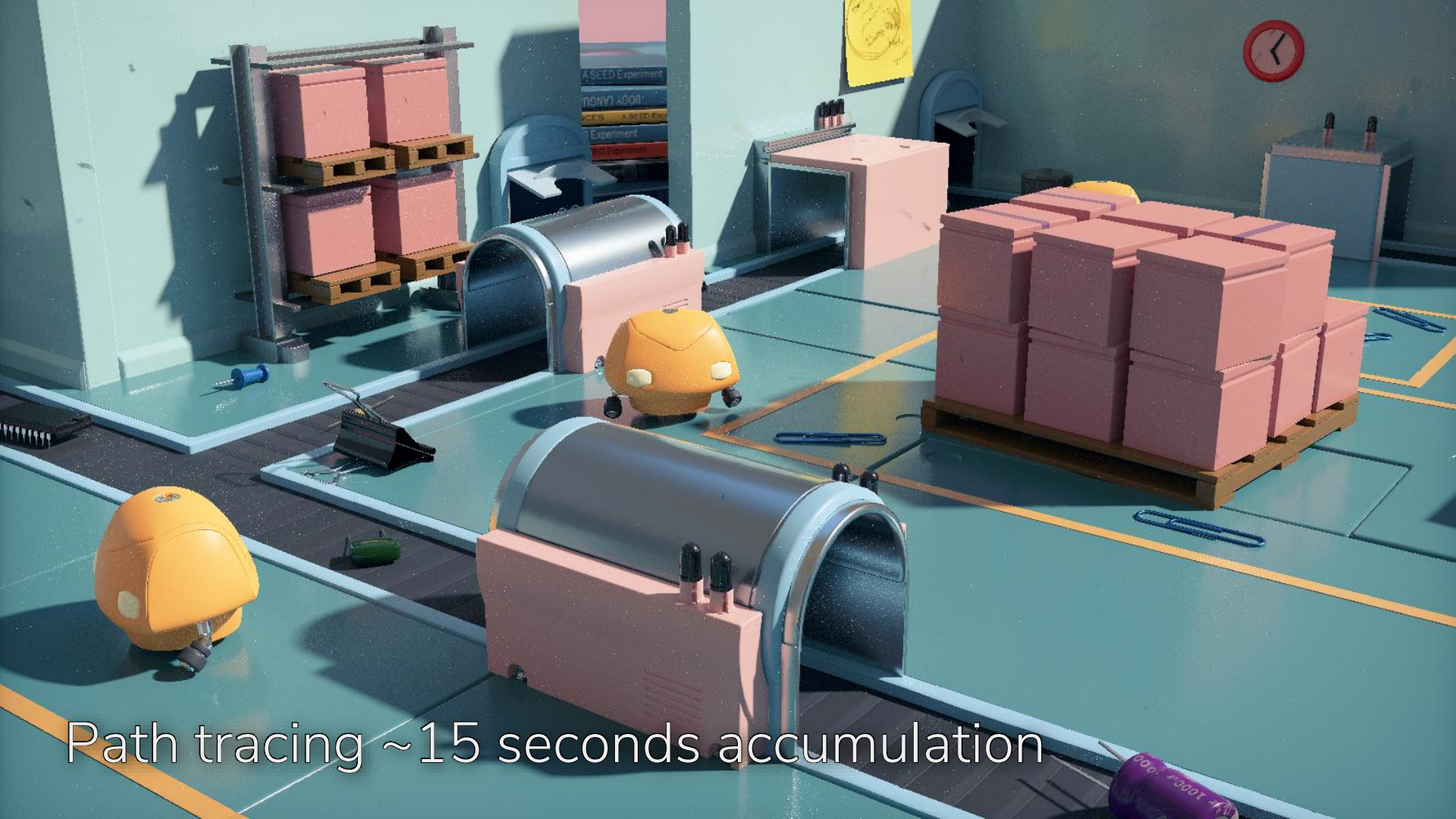


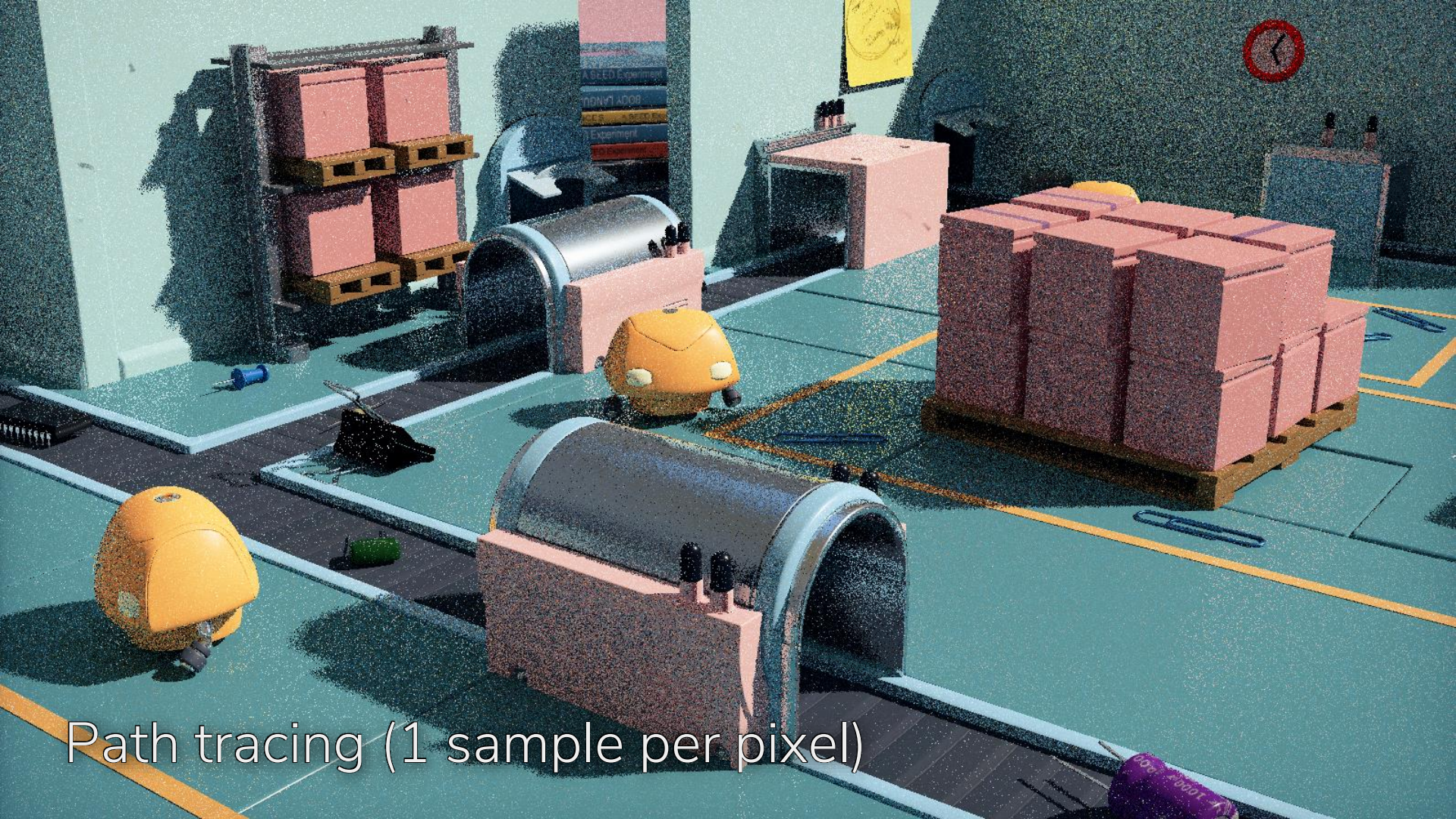
Agenda



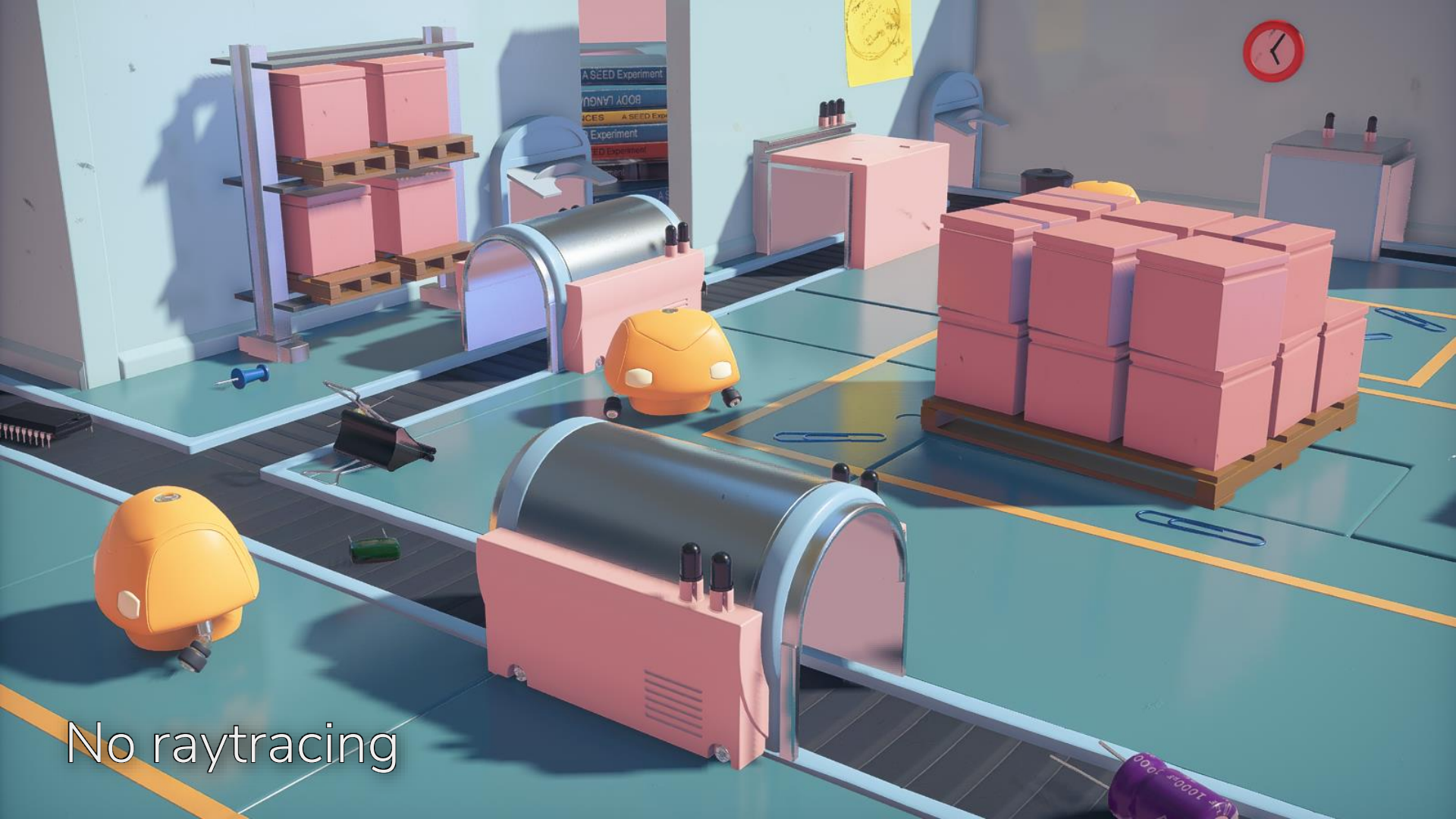
- Motivation
- Reflections
- Translucency and Transparency
- Global Illumination
- Shadows

Why use raytracing?


Path tracing ~15 seconds accumulation


Path tracing (1 sample per pixel)

Real-time hybrid with raytracing


A SEED Experiment
No raytracing

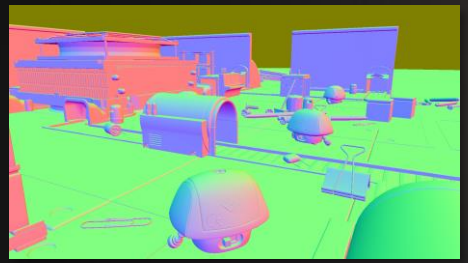
Is this a fair comparison?

- “Classic” game solutions exist
  - Local reflection volumes
  - Planar reflections
  - Lightmaps
  - “Sky visibility” maps
  - …

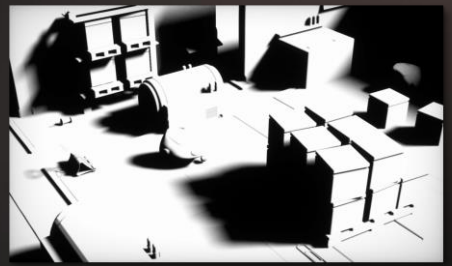
Zero artist overhead

1. Enable raytracing
2. ???
3. PROFIT!

Hybrid Rendering Pipeline



Deferred shading (raster)



Direct shadows
(raytrace or raster)

Direct lighting (compute)

Reflections
(raytrace or compute)

Global Illumination (raytrace)

Ambient occlusion
(raytrace or compute)

Transparency & Translucency
(raytrace)

Post processing (compute)

Materials

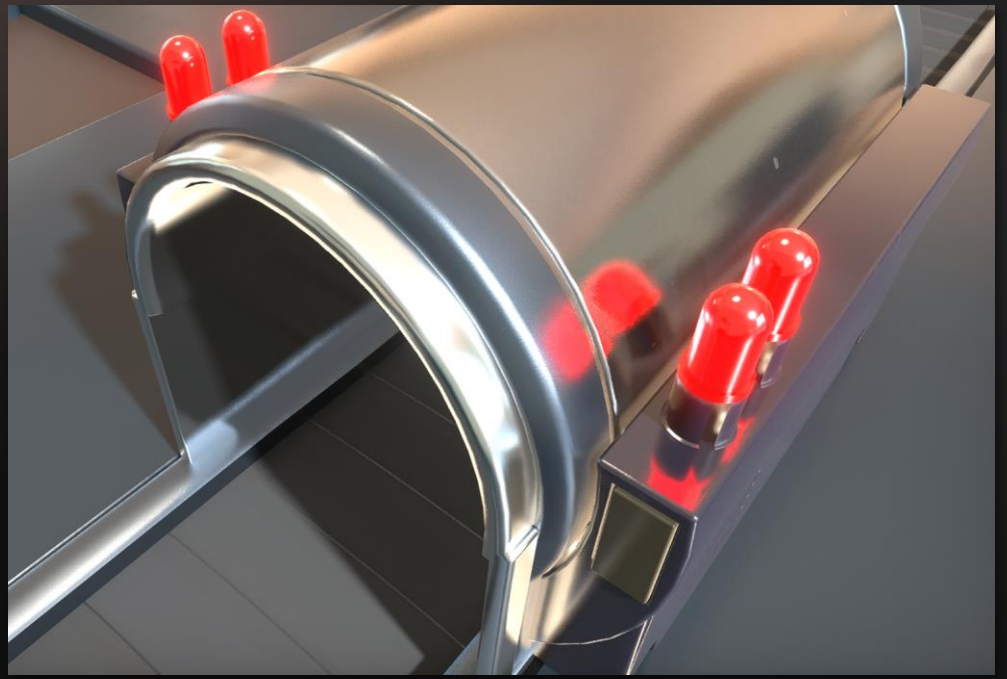
- Multiple microfacet layers
- Rapidly experiment with different looks
  - Bake down a number of layers for production
- Energy conserving
  - Automatic Fresnel between layers
- Inspired by *Arbitrarily Layered Micro-Facet Surfaces* [Weidlich 2007]
- Unified for all lighting & rendering modes
  - Raster, hybrid, path-traced reference

Objects with Multi-Layered Materials

Raytraced Reflections

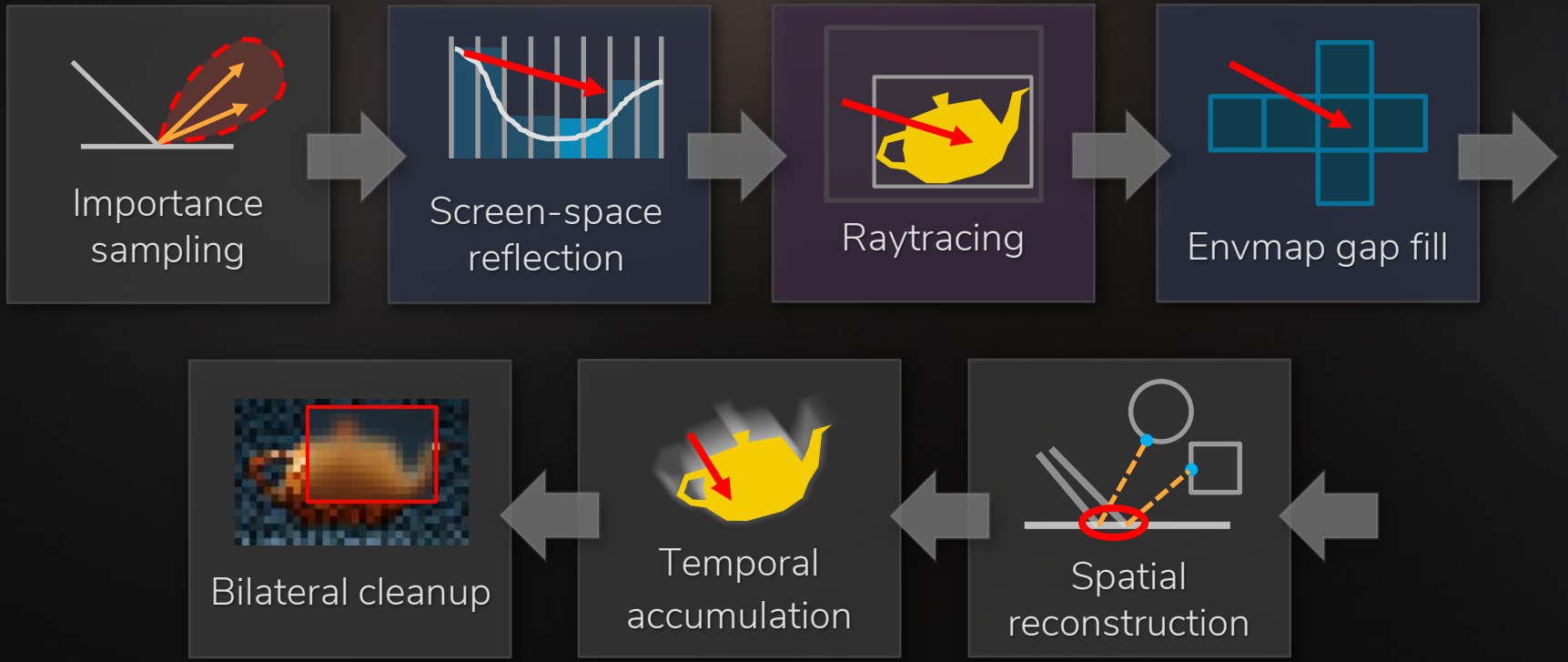
- Launch rays from G-Buffer
- Raytrace at half resolution
  - ¼ ray/pixel for reflection
  - ¼ ray/pixel for reflected shadow
- Reconstruct at full resolution

- Arbitrary normals
- Spatially-varying roughness



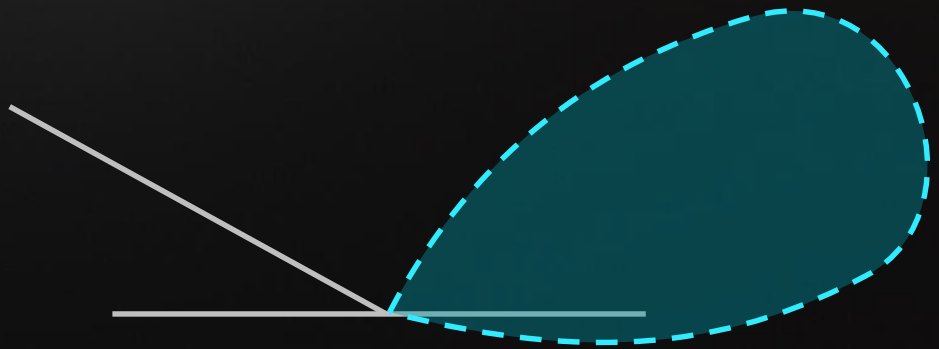
Raytraced Reflections

Reflection Pipeline



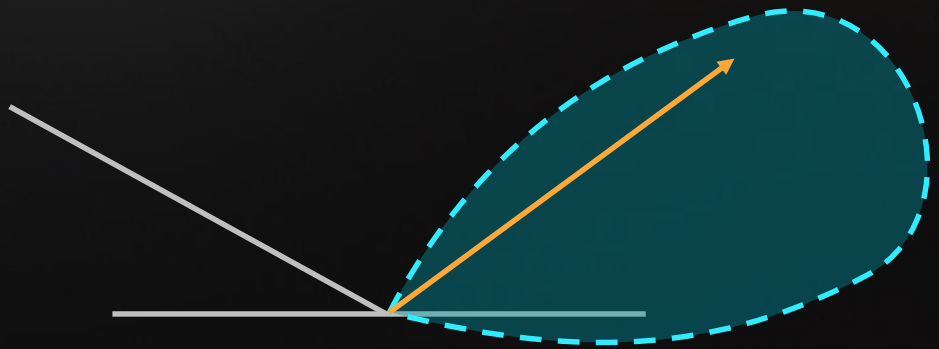
Reflection Sampling

- Microfacet importance sampling
  - Normal distribution function (NDF) part of BRDF
- Very few rays, so quality matters
  - Low-discrepancy Halton sequence
    - Cranley-Patterson rotation per pixel [PBRT]



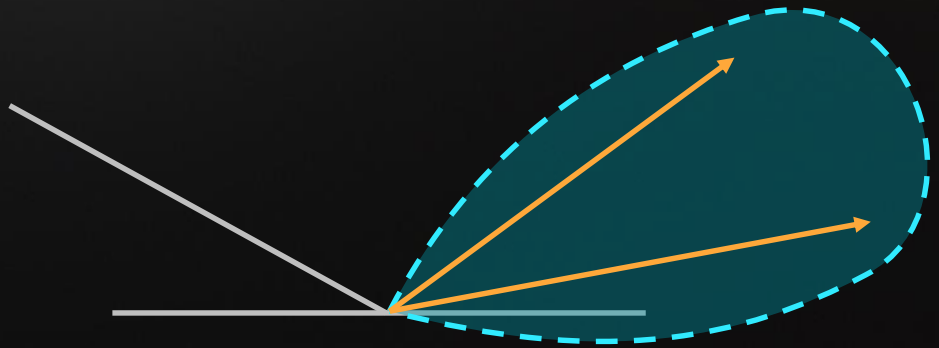
Reflection Sampling

- Microfacet importance sampling
  - Normal distribution function (NDF) part of BRDF
- Very few rays, so quality matters
  - Low-discrepancy Halton sequence
    - Cranley-Patterson rotation per pixel [PBRT]



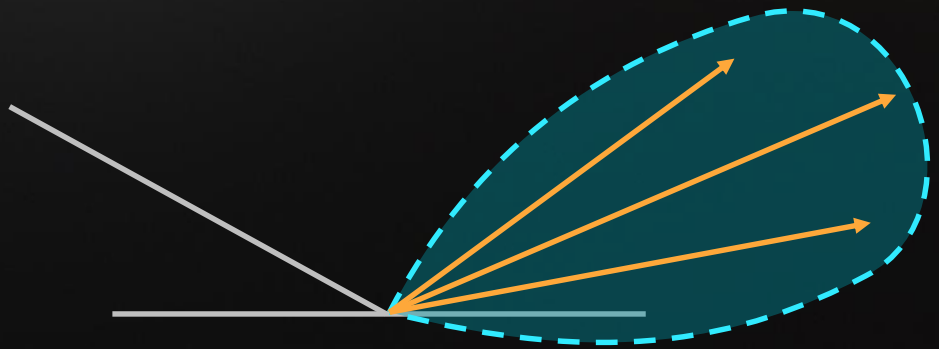
Reflection Sampling

- Microfacet importance sampling
  - Normal distribution function (NDF) part of BRDF
- Very few rays, so quality matters
  - Low-discrepancy Halton sequence
    - Cranley-Patterson rotation per pixel [PBRT]



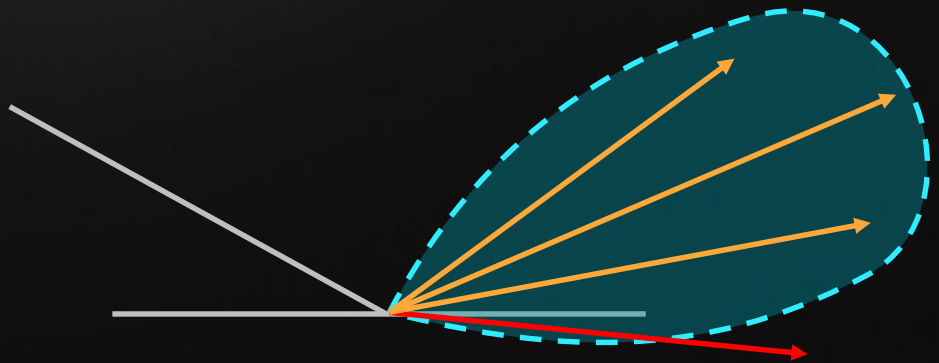
Reflection Sampling

- Microfacet importance sampling
  - Normal distribution function (NDF) part of BRDF
- Very few rays, so quality matters
  - Low-discrepancy Halton sequence
    - Cranley-Patterson rotation per pixel [PBRT]



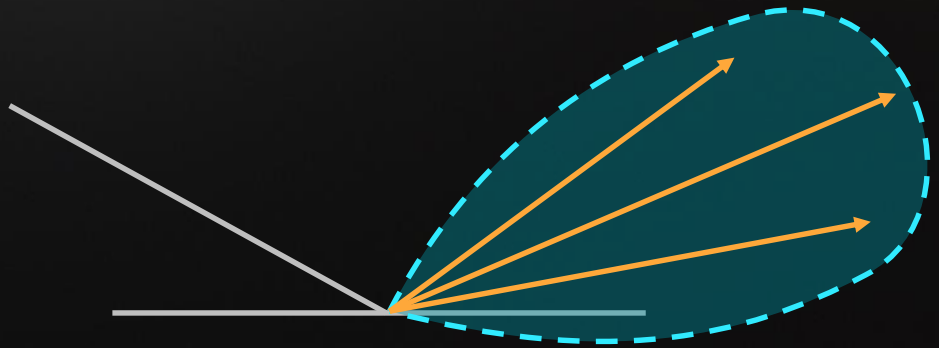
Reflection Sampling

- Microfacet importance sampling
  - Normal distribution function (NDF) part of BRDF
- Very few rays, so quality matters
  - Low-discrepancy Halton sequence
    - Cranley-Patterson rotation per pixel [PBRT]



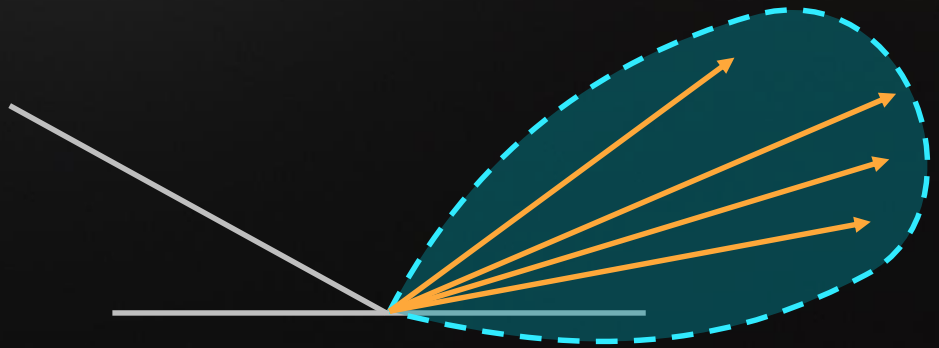
Reflection Sampling

- Microfacet importance sampling
  - Normal distribution function (NDF) part of BRDF
- Very few rays, so quality matters
  - Low-discrepancy Halton sequence
    - Cranley-Patterson rotation per pixel [PBRT]
  - Re-roll if ray below horizon



Reflection Sampling

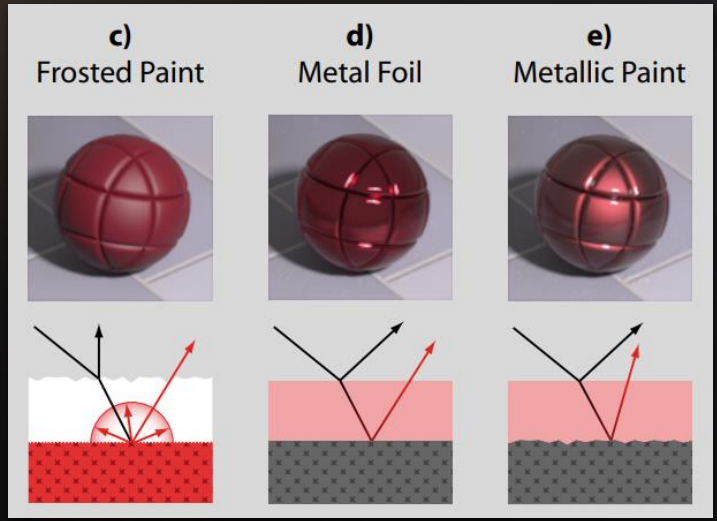
- Microfacet importance sampling
  - Normal distribution function (NDF) part of BRDF
- Very few rays, so quality matters
  - Low-discrepancy Halton sequence
    - Cranley-Patterson rotation per pixel [PBRT]
  - Re-roll if ray below horizon



Layered Material Sampling

- One ray for the whole stack
  - Stochastically select a layer
  - Based on layer visibility estimate
    - Bottom layers occluded by top layers
    - View-dependent due to Fresnel
    - Approximate with Schlick
  - Sample BRDF of selected layer

  - Multiple layers can generate same direction
    - Query each layer about generated direction
    - Add up probabilities

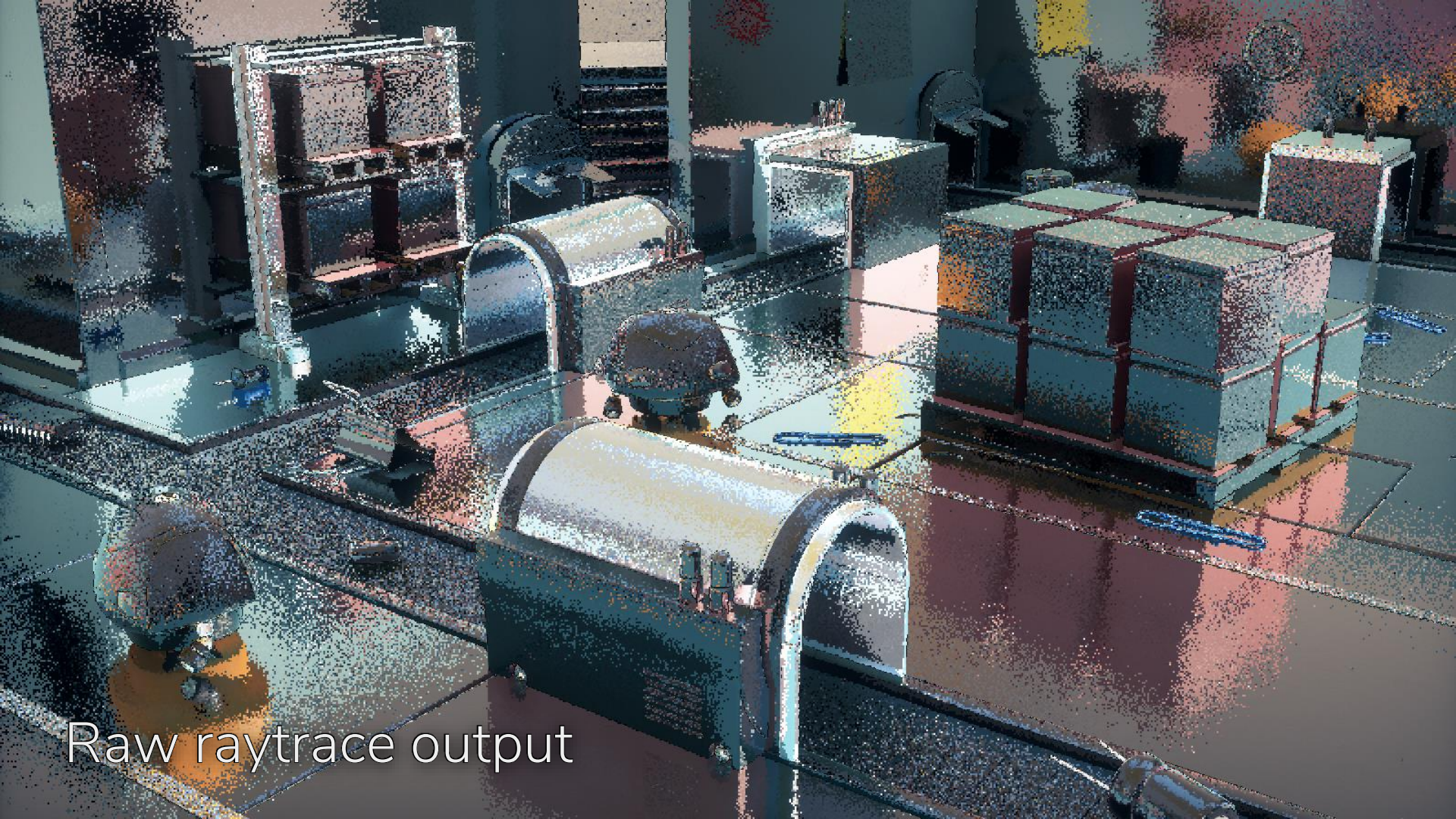


Multi-Layered Materials [Weidlich 2007]

Reflection Raygen

- Read generated ray
- TraceRay()
  - Same hit shaders as path-tracing reference
- Output packed in 2x RGBA 16F

Color R	Color G	Color B	G-buffer Depth
Dir X * Hit T	Dir Y * Hit T	Dir Z * Hit T	Inverse PDF


Raw raytrace output

Why Stochastic?

- Gives right answer, but
  - Produces noise
  - Thrashes caches
- Alternative: post-filter
  - Needs arbitrarily large kernel
  - Introduces bleeding
  - Sensitive to aliasing
- Meet in the middle?
  - Bias samplers
  - Combine with spatial filtering
  - High variance -> increase bias
  - More research needed!

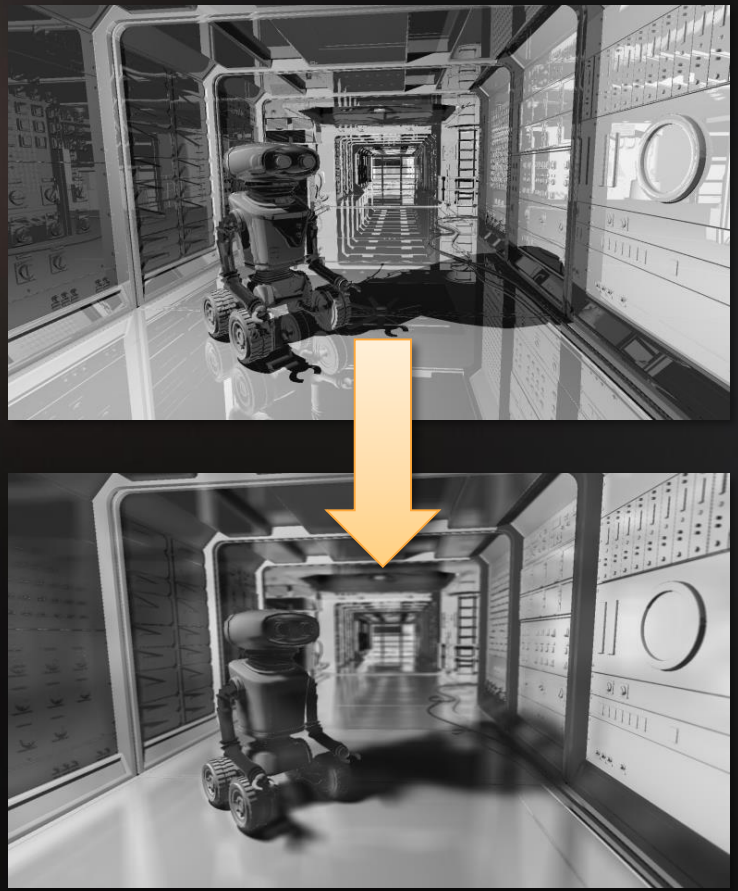
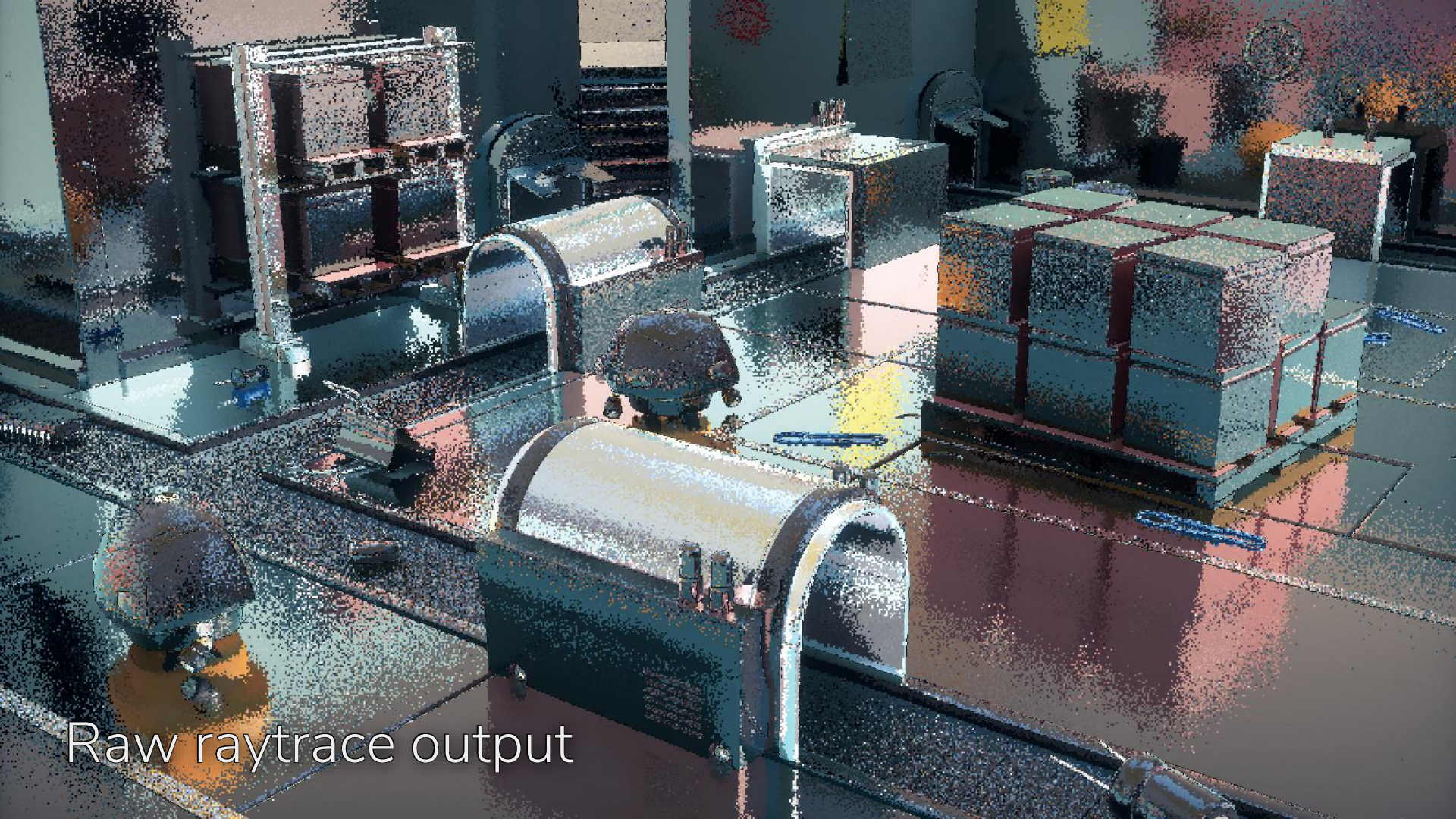


Image Space Gathering [Robison 2009]


Raw raytrace output

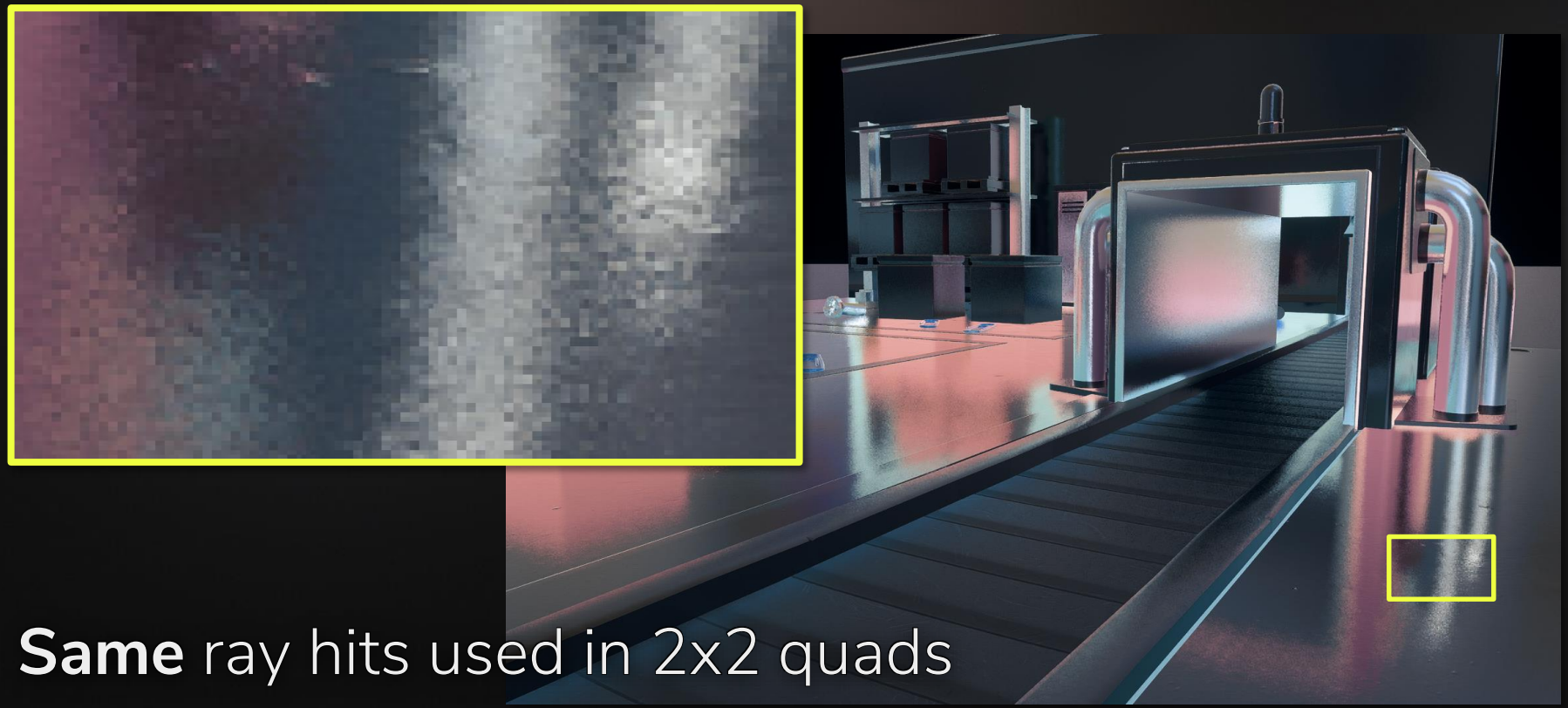

+Spatial reconstruction

- Based on Stochastic Screen-Space Reflections [Stachowiak 2015]
- For every full-res pixel, use 16 half-res ray hits
  - Poisson-disk distribution around pixel
  - Scale by local BRDF
  - Divide by ray PDF
- Ratio estimator [Heitz 2018]

```
result    = 0.0
weightSum = 0.0
for pixel in neighborhood:
    weight = localBrdf(pixel.hit) / pixel.hitPdf
    result += color(pixel.hit) * weight
    weightSum += weight
result /= weightSum
```

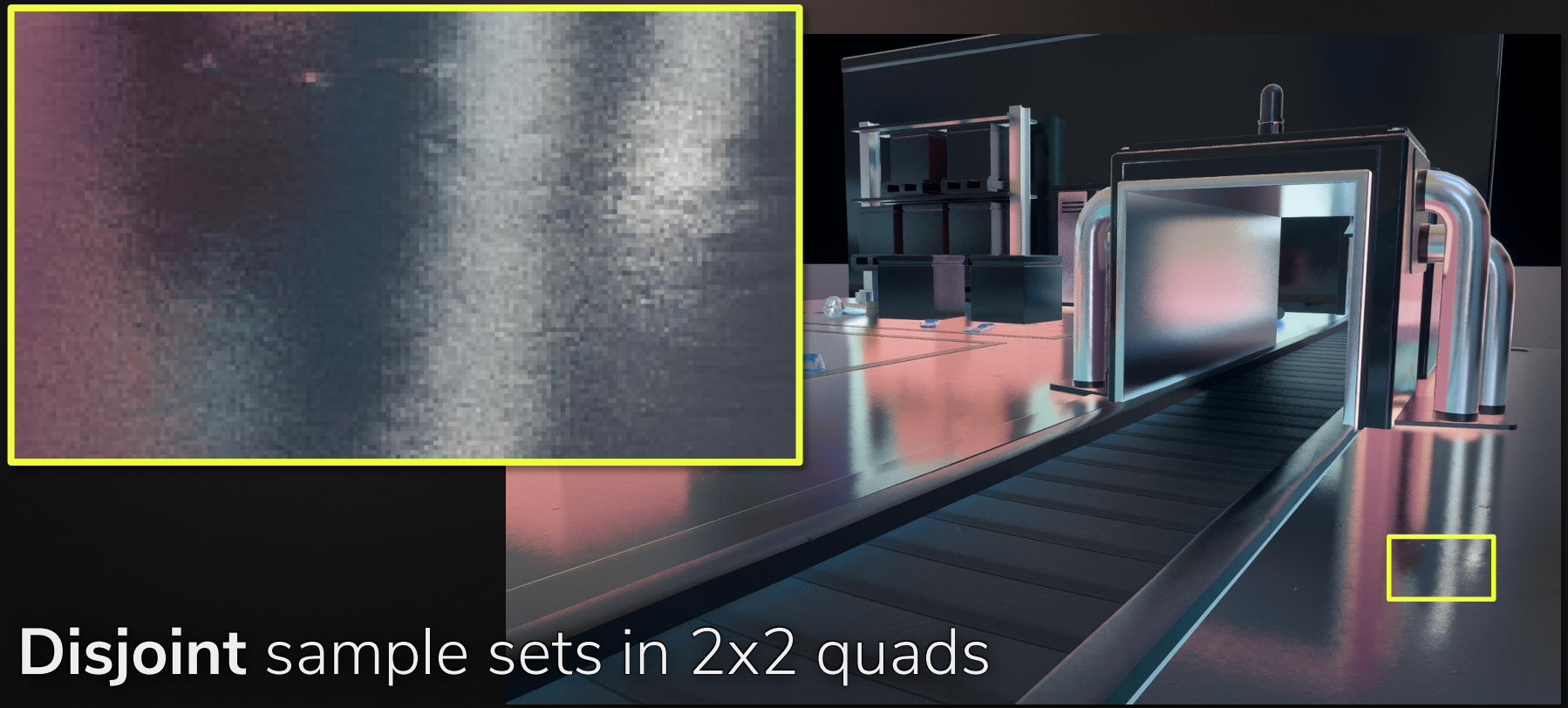
+Spatial reconstruction

Spatial upsampling



Same ray hits used in 2x2 quads

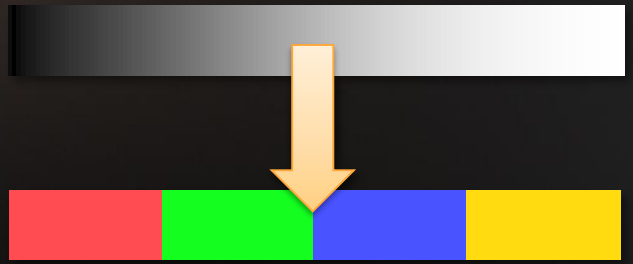
Spatial upsampling

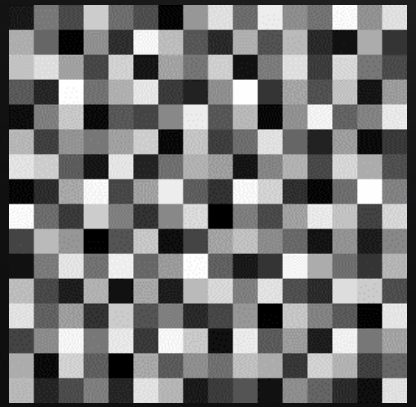


Disjoint sample sets in 2x2 quads

Spatial Sample Pattern

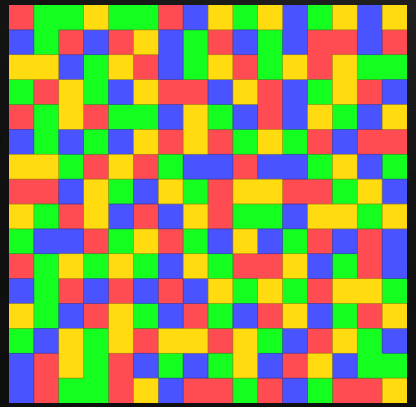
- Four sample sets for pixels in 2x2 quad
- Threshold blue noise into four classes





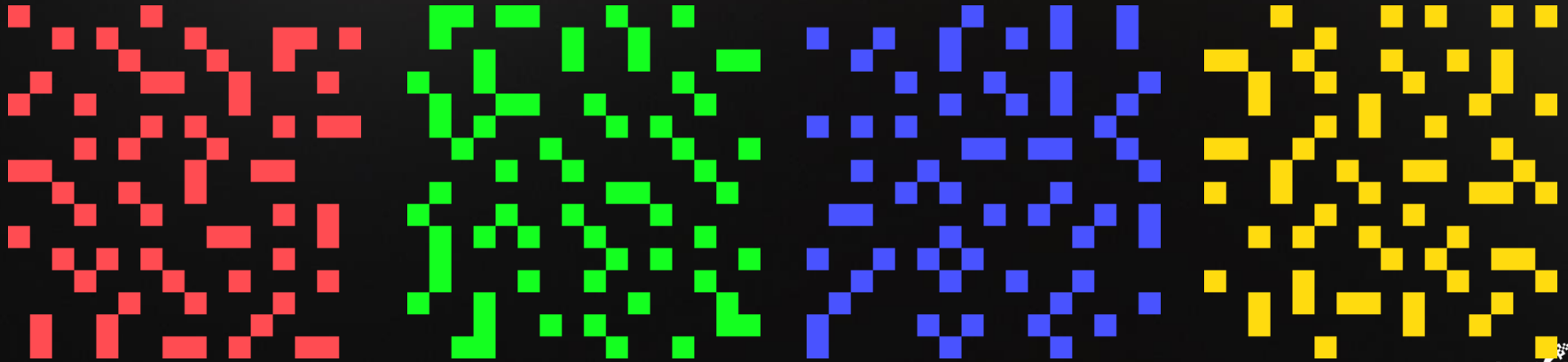
Spatial Sample Pattern

- Four sample sets for pixels in 2x2 quad
- Threshold blue noise into four classes



Spatial Sample Pattern

- Four sample sets for pixels in 2x2 quad
- Threshold blue noise into four classes
- Class determines sample set



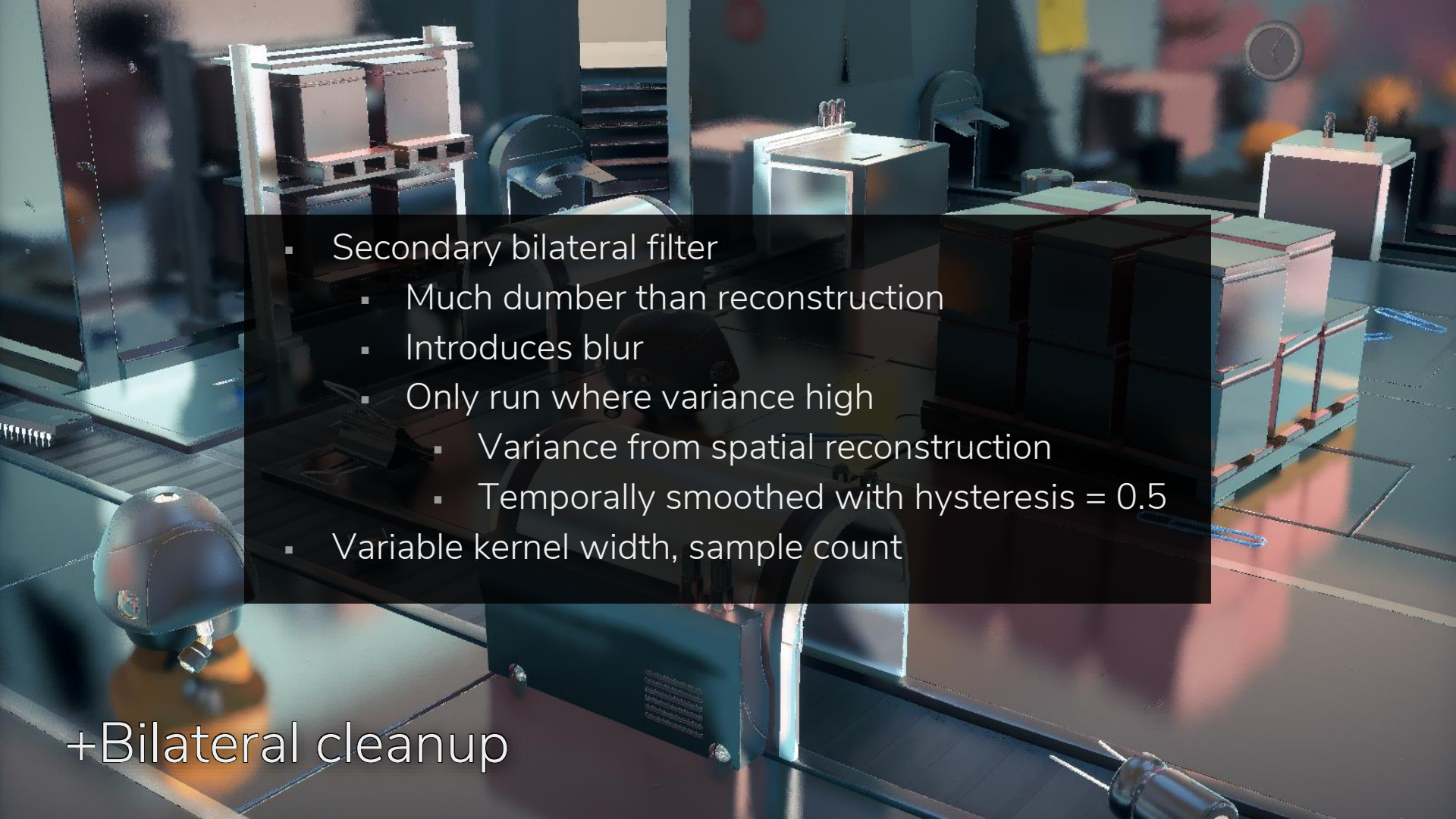
Spatial Sample Pattern

- Four sample sets for pixels in 2x2 quad
- Threshold blue noise into four classes
- Class determines sample set
- Take 16 samples from each set in a disk
  - Disjoint high-quality samples


Spatial reconstruction

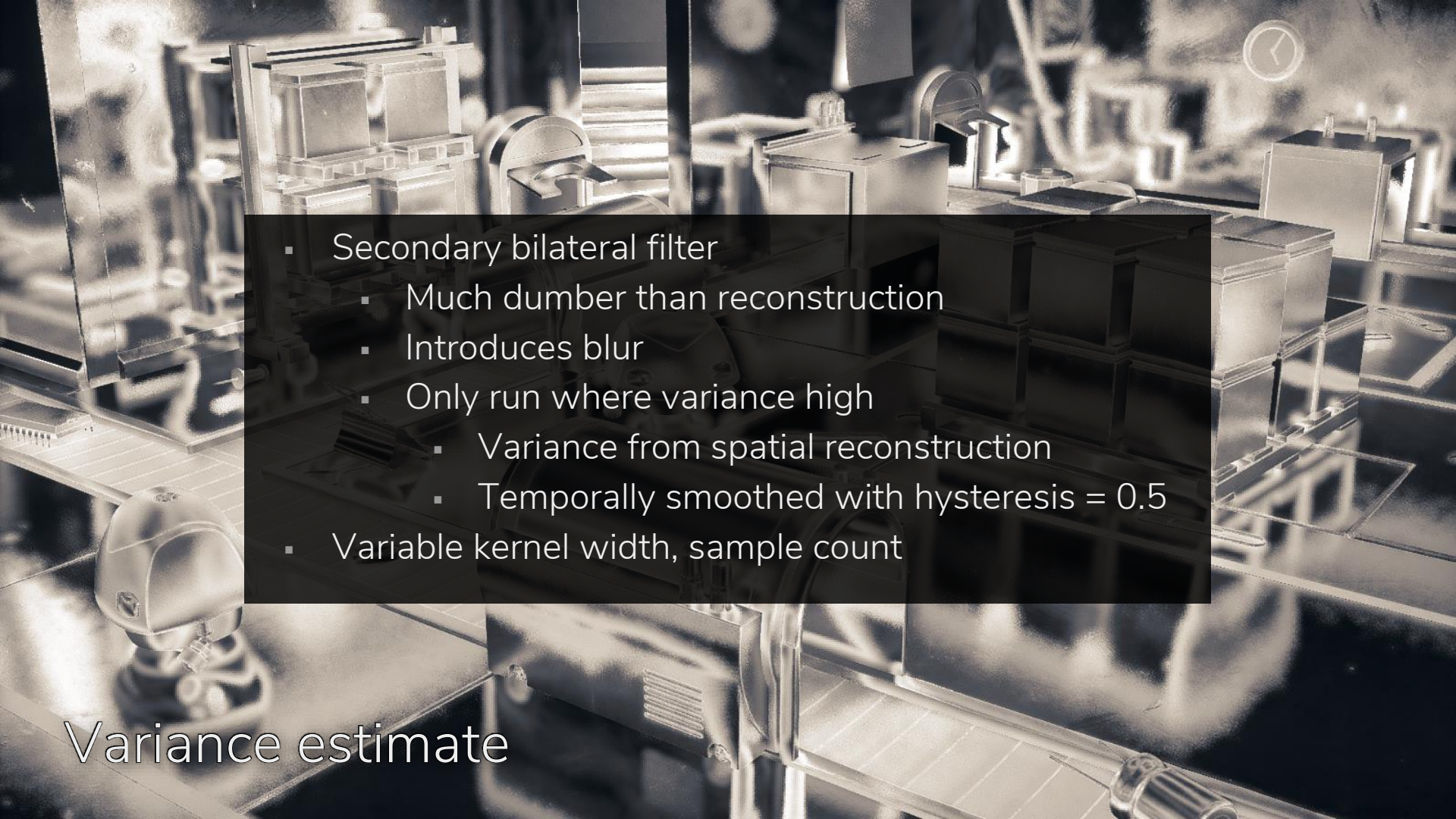

+Temporal accumulation


+Bilateral cleanup



- Secondary bilateral filter
  - Much dumber than reconstruction
  - Introduces blur
  - Only run where variance high
    - Variance from spatial reconstruction
    - Temporally smoothed with hysteresis = 0.5
- Variable kernel width, sample count

+Bilateral cleanup



- Secondary bilateral filter
  - Much dumber than reconstruction
  - Introduces blur
  - Only run where variance high
    - Variance from spatial reconstruction
    - Temporally smoothed with hysteresis = 0.5
- Variable kernel width, sample count

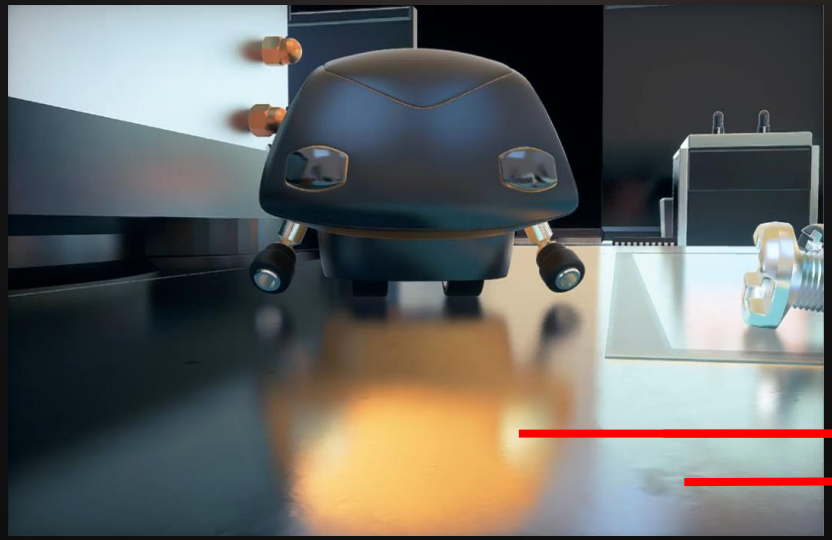
Variance estimate

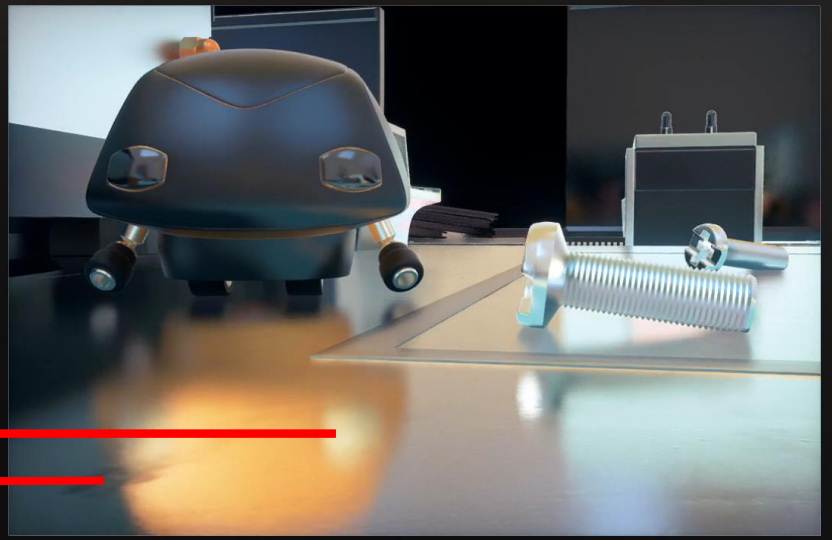

+Bilateral cleanup


+Temporal anti-aliasing

Temporal Reprojection

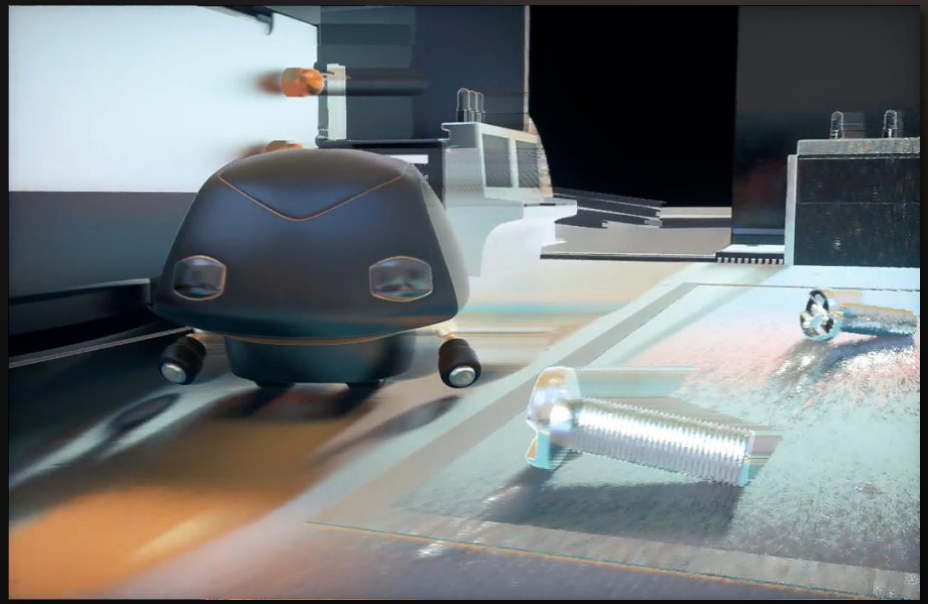
- Two sources of velocity
  - Per-pixel motion vector
  - Reprojection of hit point



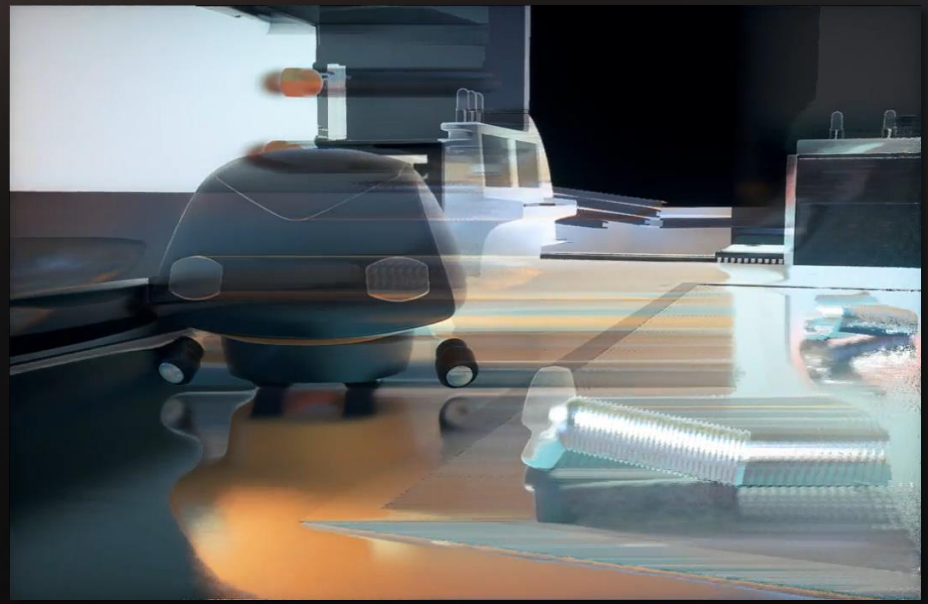


Temporal Reprojection

- Neither reprojection method robust



Per-pixel motion vector



Reprojection of hit point

Dual-Source Reprojection

- Sample history using both reprojection strategies
- Estimate local statistics during spatial reconstruction
  - RGB mean
  - RGB standard deviation
  - Of used hit samples
- Weigh contributions
  - rgb_dist = (rgb – rgb_mean) / rgb_dev
  - dist = (rgb – rgb_mean) / rgb_dev
  - w = exp2(-10 * luma(dist))

Reprojection Clamp

- Build color box from local statistics
  - rgb_min = rgb_mean – rgb_dev
  - rgb_max = rgb_mean + rgb_dev
- Clamp reprojected values to box
  - rgb = clamp(rgb, rgb_min, rgb_max)
  - Same idea as in temporal anti-aliasing [Karis 2014]
- Weigh clamped contributions
  - rgb_sum += val * w

- Introduces bias
- Fixes most ghosting

Transparency & Translucency

- Raytracing enables accurate light scattering
- Refractions
  - Order-independent (OIT)
  - Variable roughness
  - Handles multiple IOR transitions
  - Beer-Lambert absorption
- Translucency
  - Light scattering inside a medium
  - Inspired by Translucency in Frostbite [Barré-Brisebois 2011]

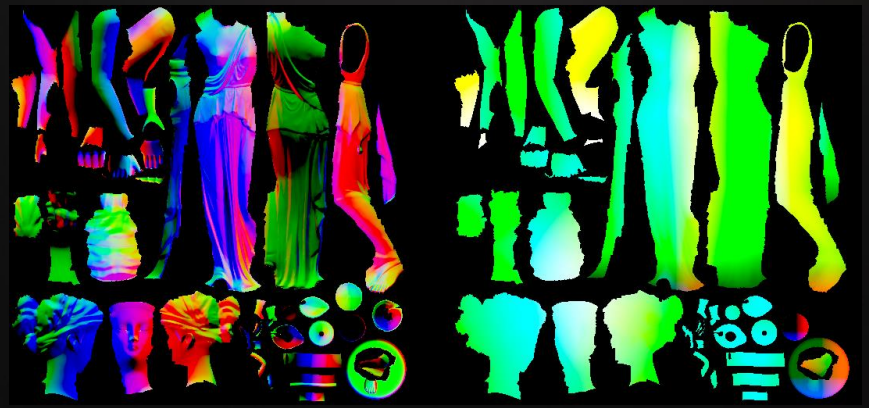
Glass and Translucency

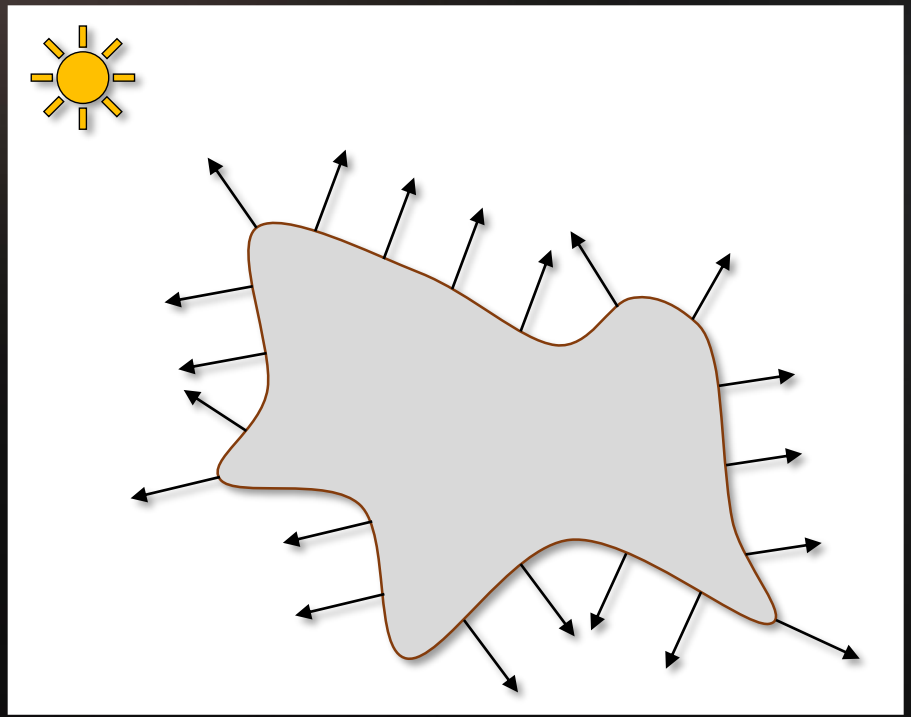
Integration

- Multiple samples required for rough refractions and translucency
- Multi-layer screen-filtering complicated
  - Layered framebuffers?
  - Per-pixel linked lists?
  - Filter in 2D or between layers?
  - More research needed!
- Accumulate in texture-space instead
  - Stable integration domain
- Static texture allocation
  - 512*512 per object
  - Time-sliced, ~1M rays/frame

Translucency Breakdown

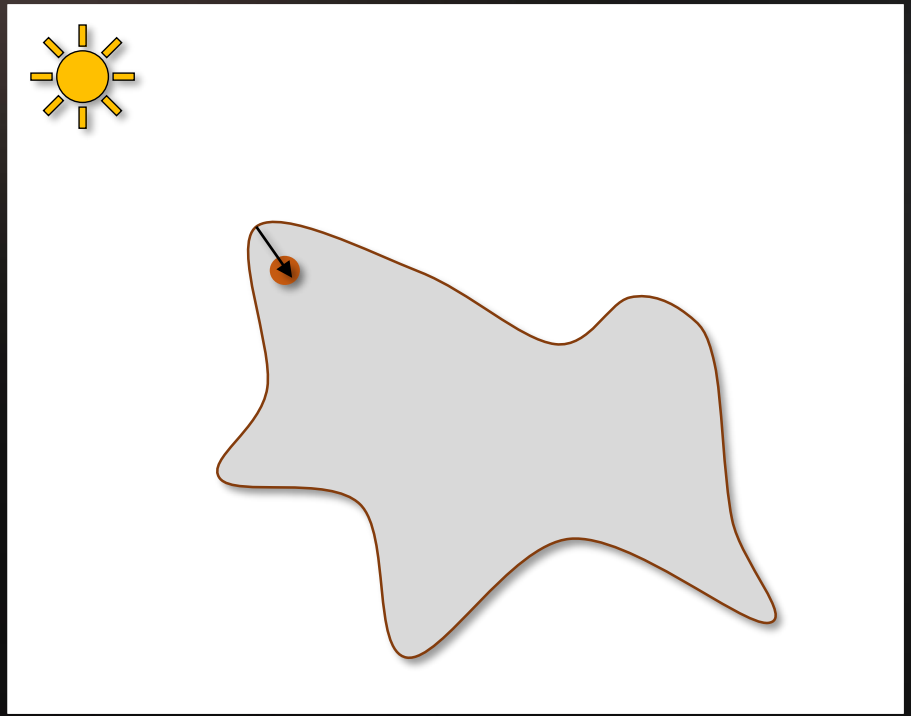
- For every valid position & normal





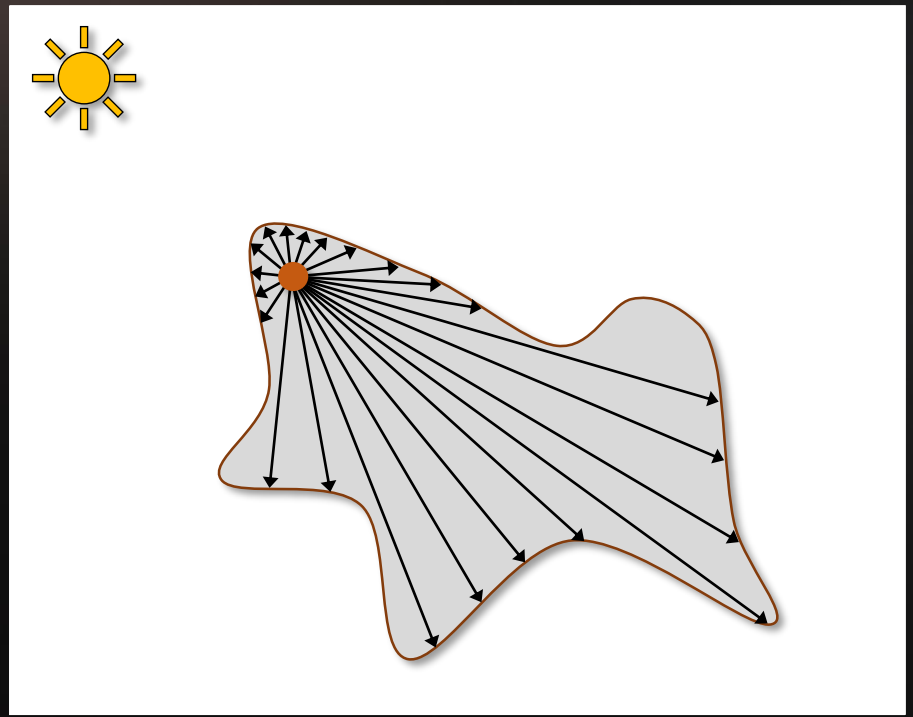
Translucency Breakdown

- For every valid position & normal
- Flip normal and push (ray) inside



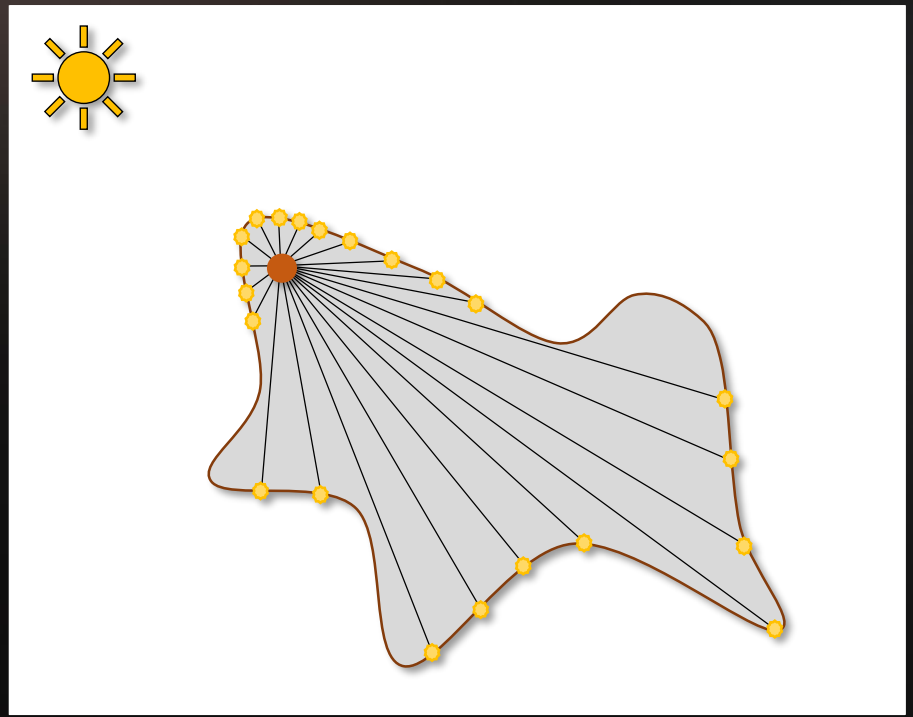
Translucency Breakdown

- For every valid position & normal
- Flip normal and push (ray) inside
- Launch rays in uniform sphere dist.
  - *(Importance-sample phase function)*



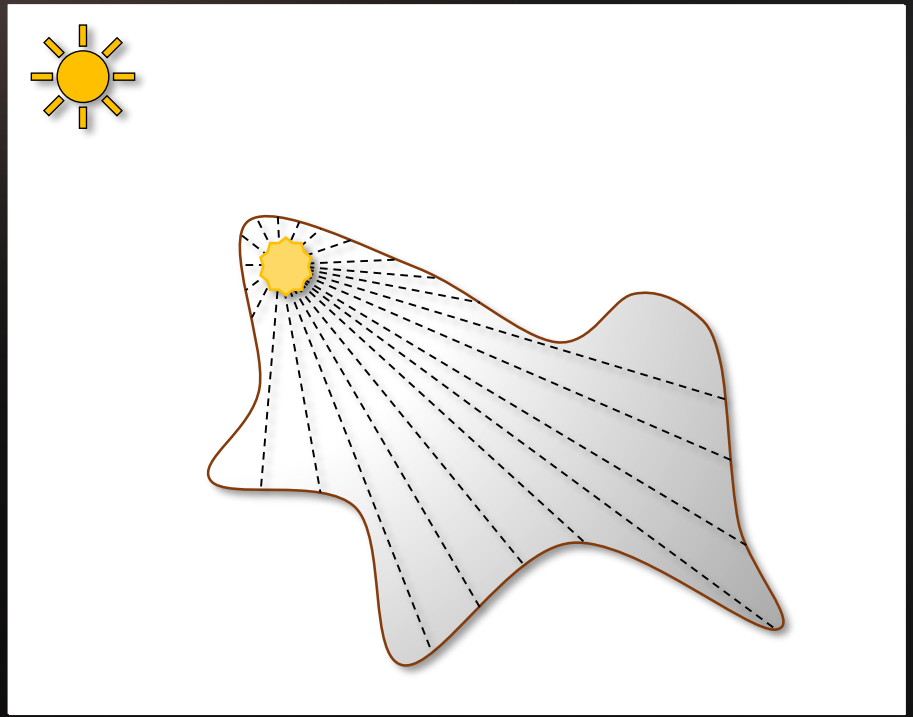
Translucency Breakdown

- For every valid position & normal
- Flip normal and push (ray) inside
- Launch rays in uniform sphere dist.
  - *(Importance-sample phase function)*
- Compute lighting at intersection



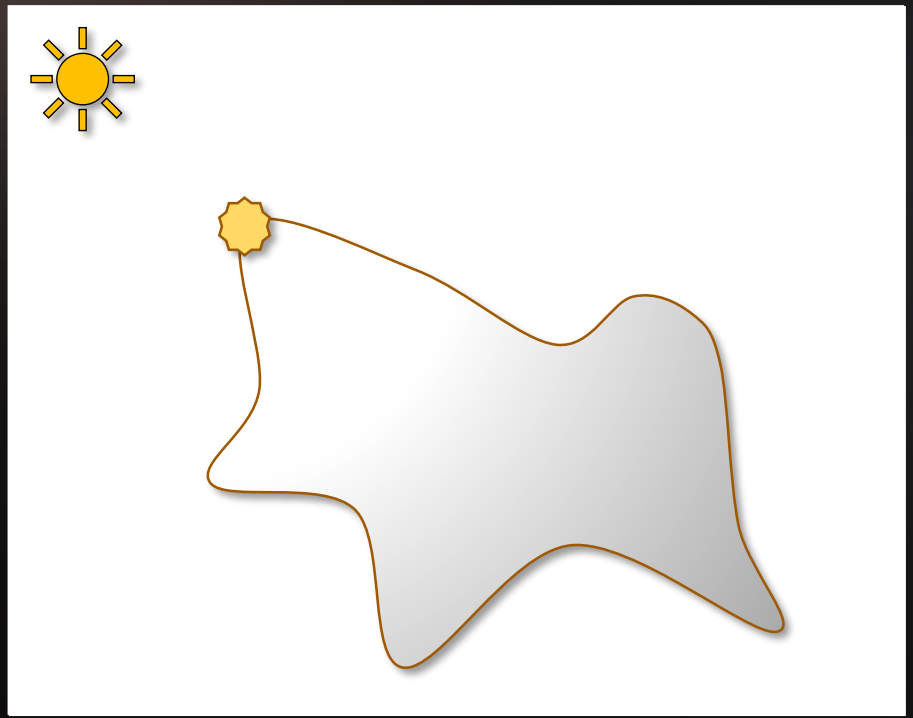
Translucency Breakdown

- For every valid position & normal
- Flip normal and push (ray) inside
- Launch rays in uniform sphere dist.
  - *(Importance-sample phase function)*
- Compute lighting at intersection
- Gather all samples



Translucency Breakdown

- For every valid position & normal
- Flip normal and push (ray) inside
- Launch rays in uniform sphere dist.
  - *(Importance-sample phase function)*
- Compute lighting at intersection
- Gather all samples
- Update value in texture



Translucency Filtering

- Can denoise spatially and/or temporally
- Temporal: build an update heuristic
  - Reactive enough for moving lights & objects
  - Exponential moving average can be OK
- Variance-adaptive mean estimation
  - Based on exponential moving average
  - Adaptive hysteresis

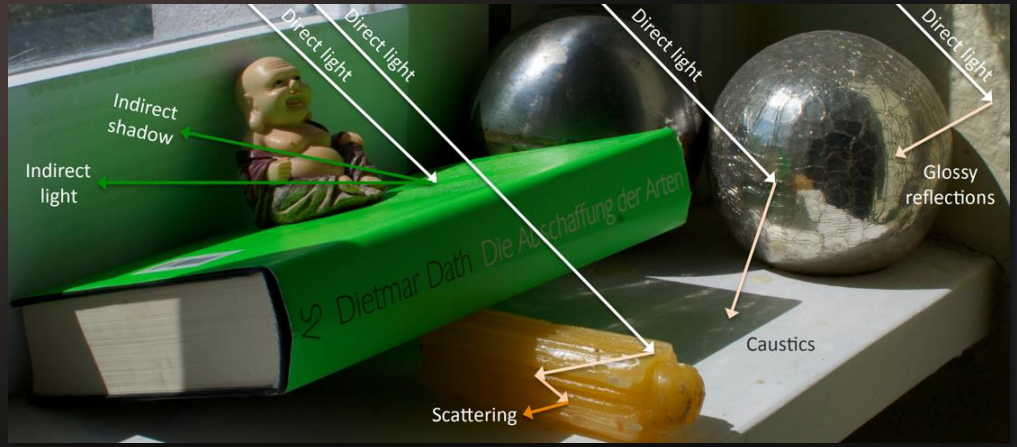
Transparency

- Similar approach to translucency
- Launch ray using view's origin and direction
- Refract based on medium's index of refraction
  - Sample a BSDF for rough refraction
- Trace a ray in the scene & sample lighting
- Tint the result
  - Chromatic aberration from interface
  - Beer-Lambert absorption in medium

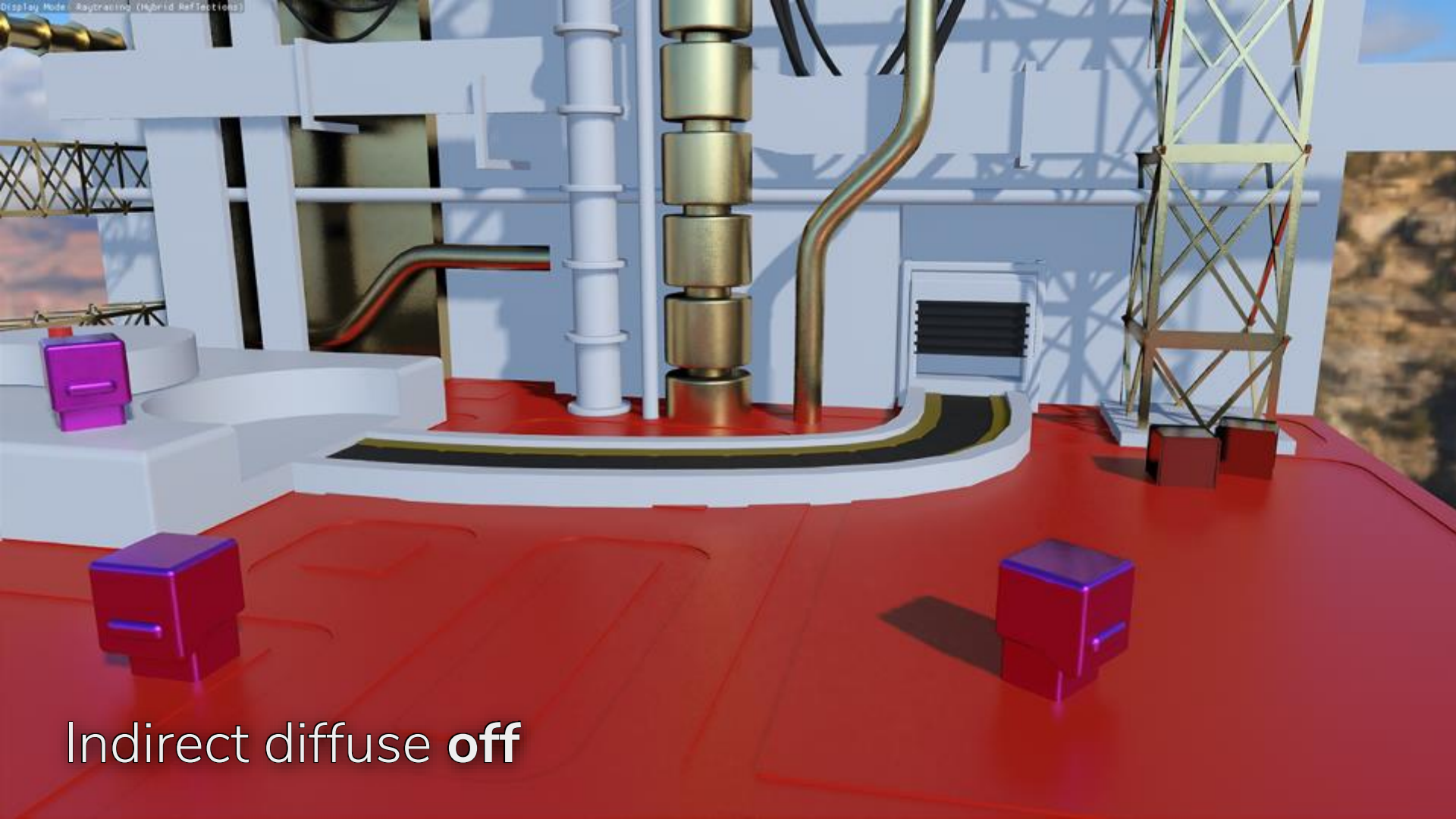
Transparency

- Similar approach to translucency
- Launch ray using view’s origin and direction
- Refract based on medium's index of refraction
  - Sample a BSDF for rough refraction
- Trace a ray in the scene & sample lighting
- Tint the result
  - Chromatic aberration from interface
  - Beer-Lambert absorption in medium
- Pen: we don’t handle transparent shadows yet

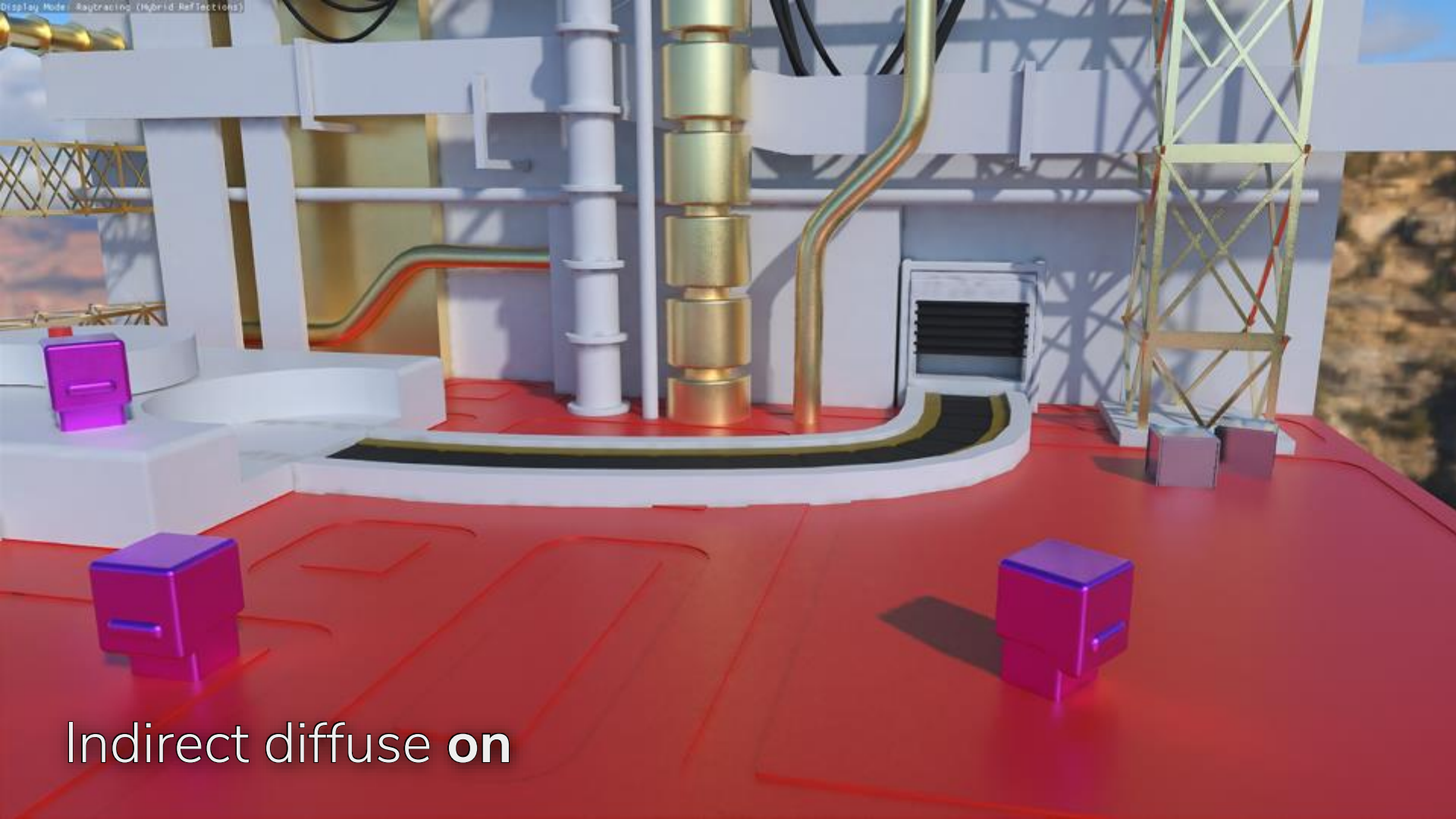
Global Illumination



- Important for consistency
- Minimize artist overhead
- We want a technique with:
  - No precomputation
  - No parametrization (UVs, proxies)
  - Support for static and dynamic scenes
  - Adaptive refinement



Indirect diffuse **off**

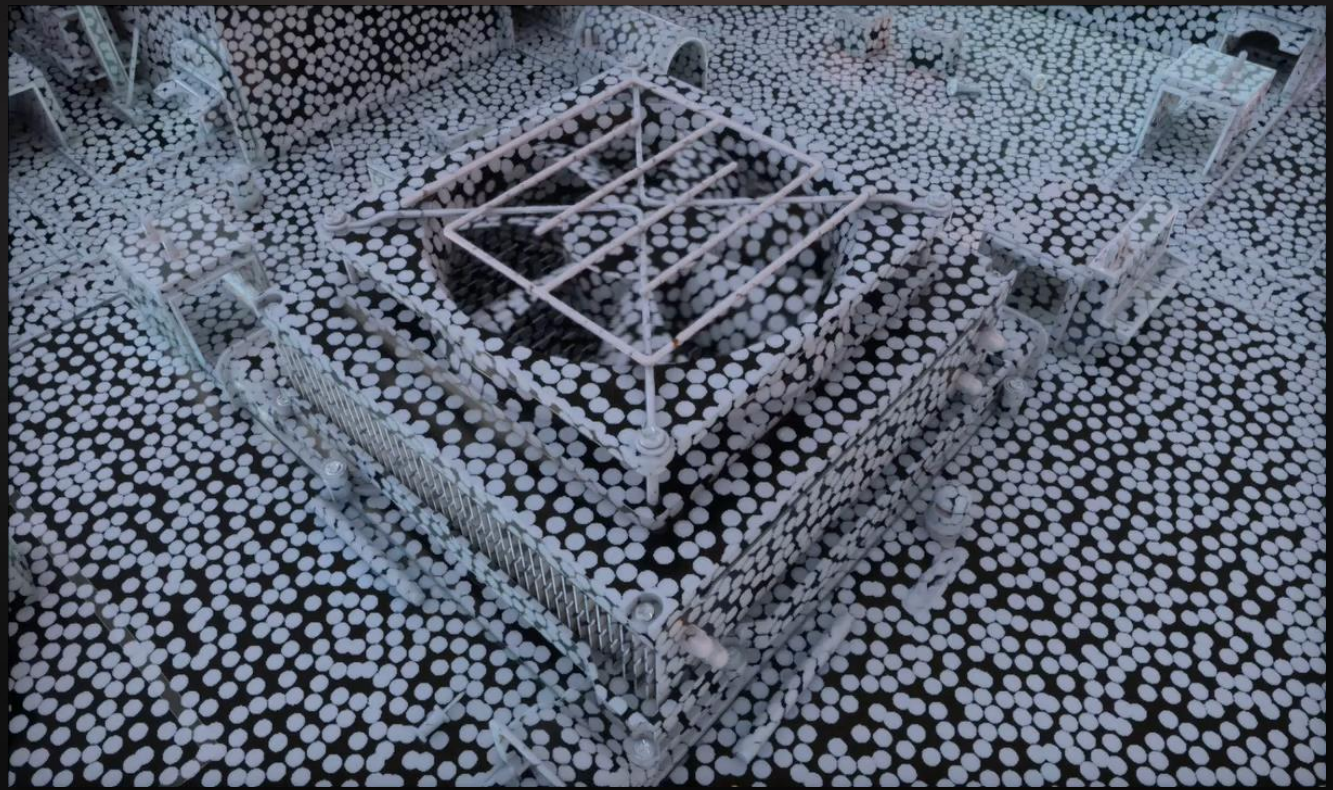


Indirect diffuse **on**

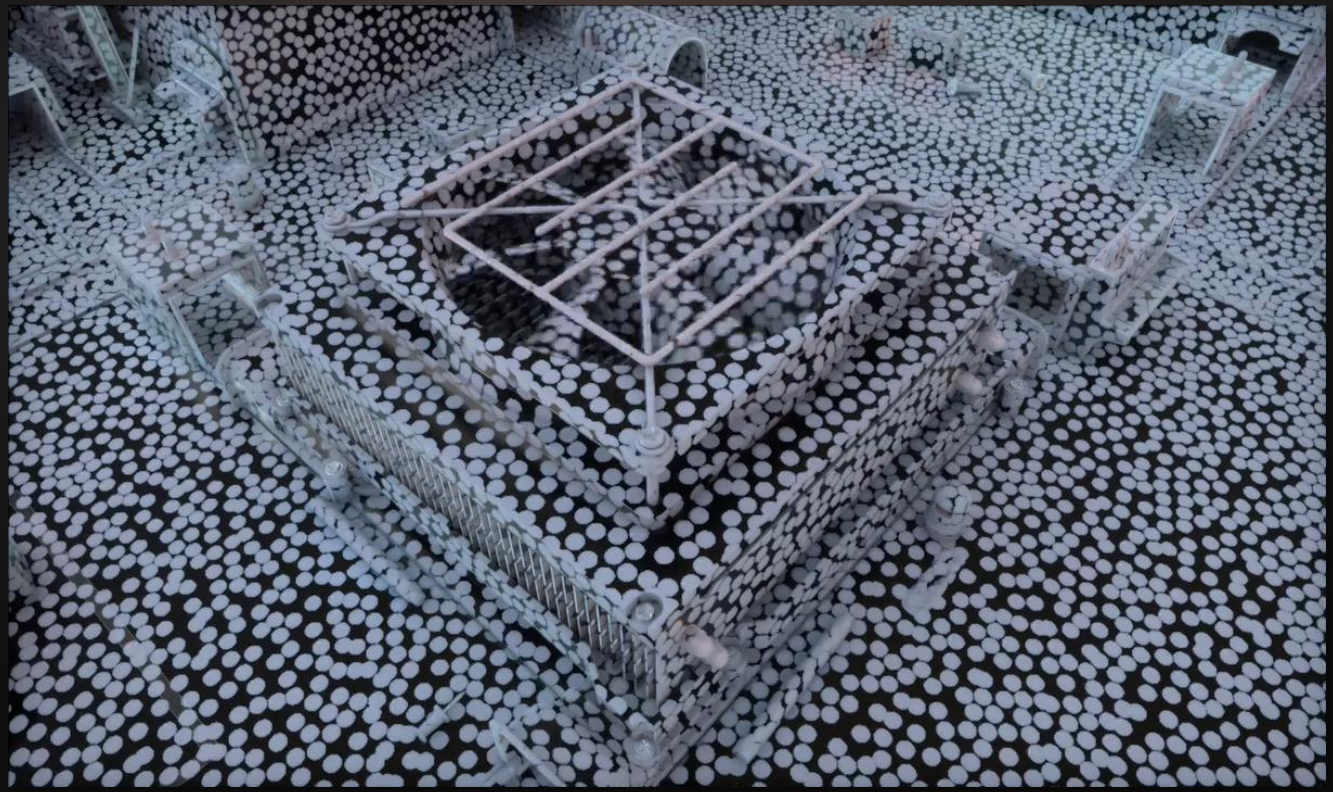
Spatial Storage

- Can’t afford to solve every frame
  - Our GI budget: 250k rays / frame
- World space vs screen space
  - World space is stable
  - Screen-space fully dynamic
- World space surfels
  - Easy to accumulate
  - Distribute on the fly
  - No parameterization
  - Smooth result by construction
  - Position, normal, radius, animation info

Surfel Skinning

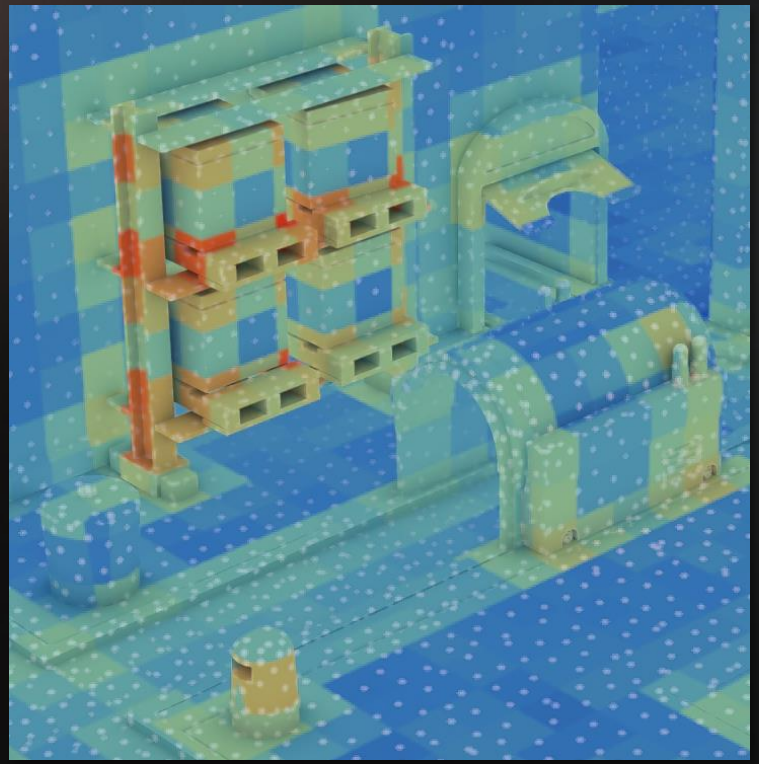


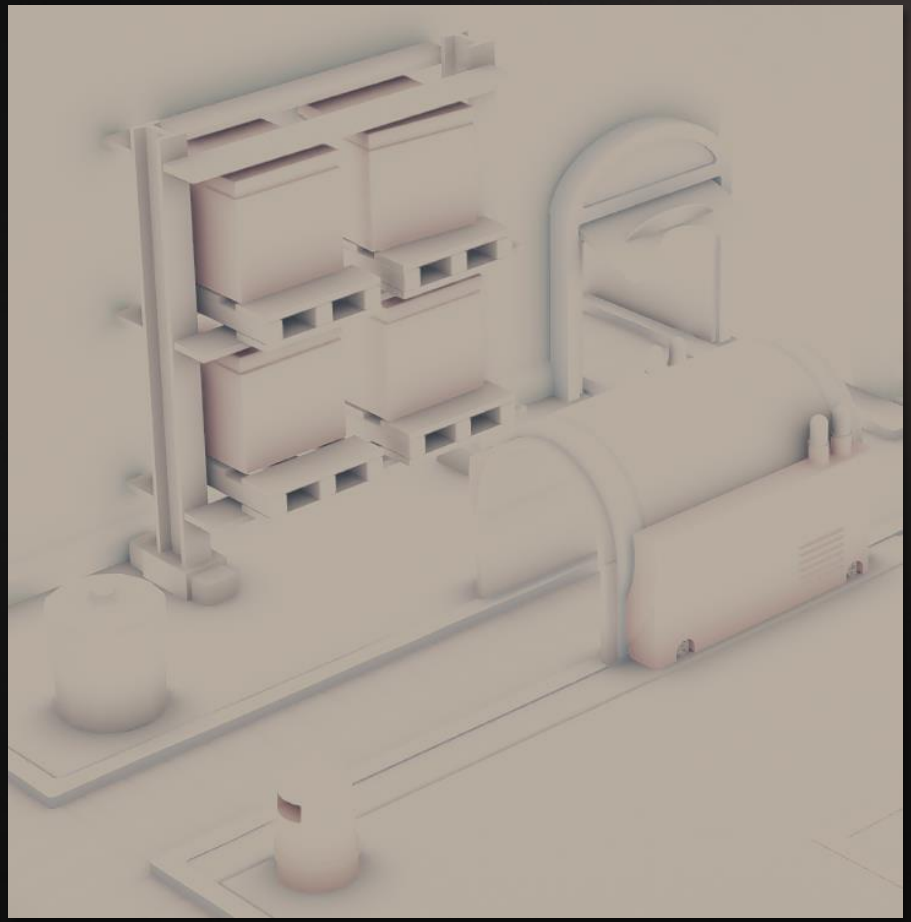
Surfel Skinning

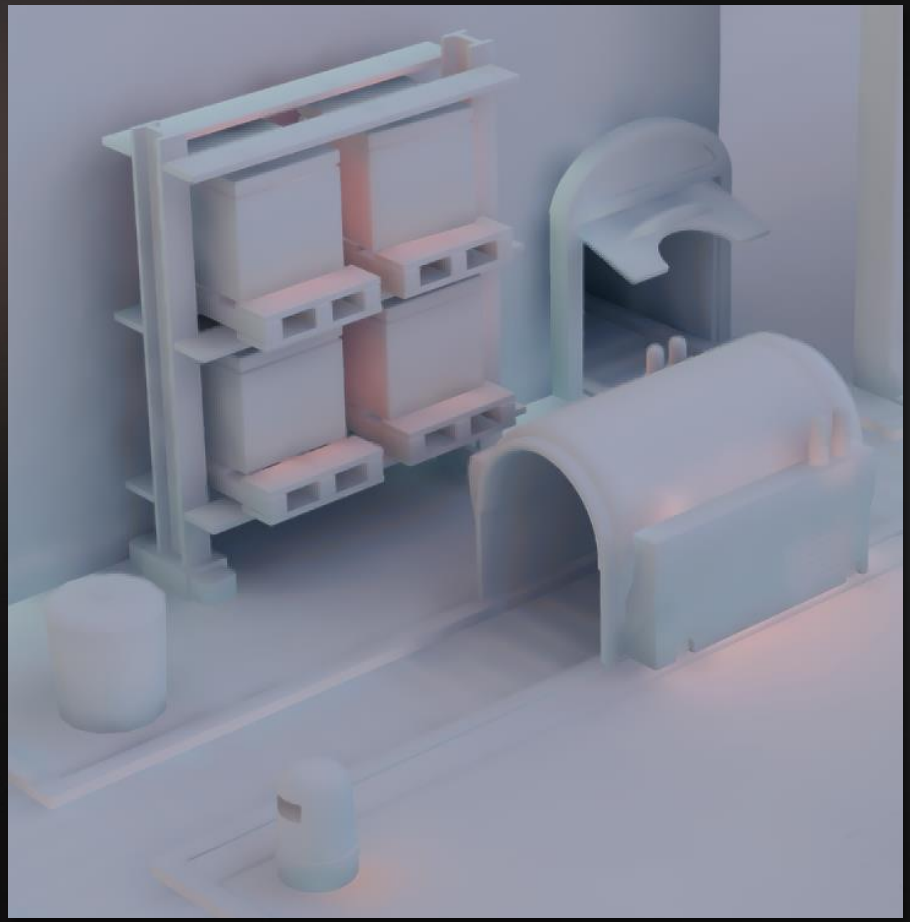


Surfel Screen Application

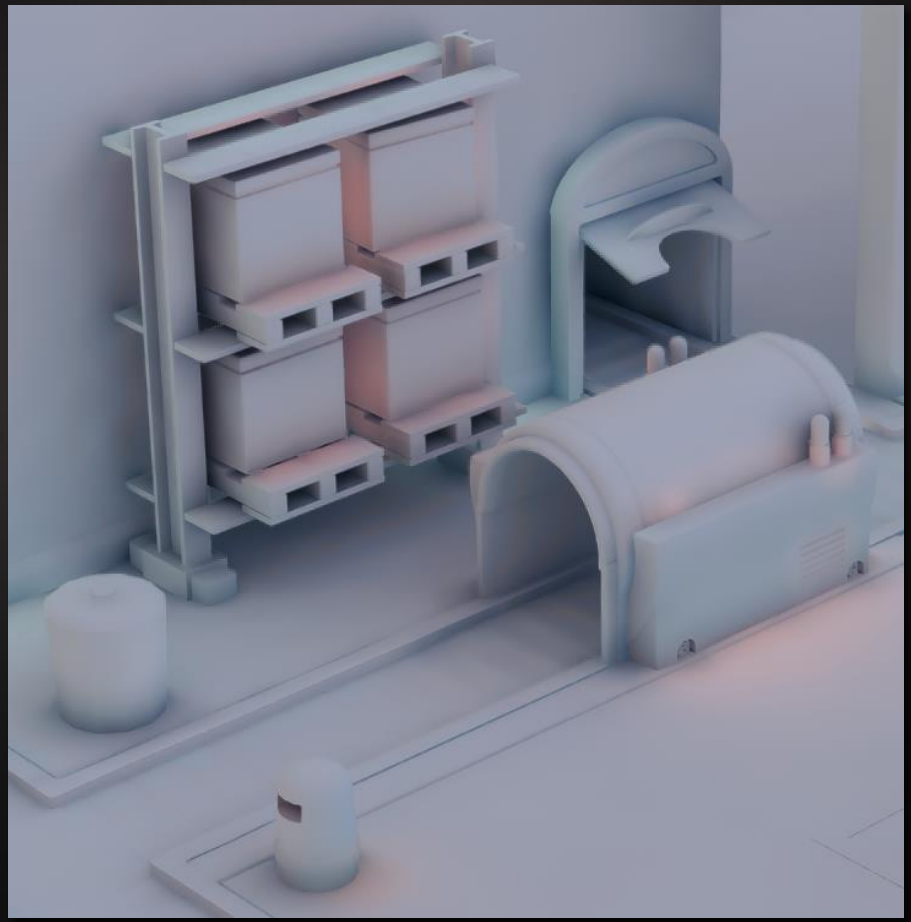
- Render like deferred light sources
  - Diffuse only
  - Smoothstep distance attenuation
    - Mahalanobis metric
  - Squared dot of normals angular weight
  - Inspired by [Lehtinen 2008]
- World-space 3D culling
  - Uniform grid
  - Each cell holds list of surfels
  - Find one cell per pixel, use all from list







Combine with Screen Space AO [Jimenez 2016]

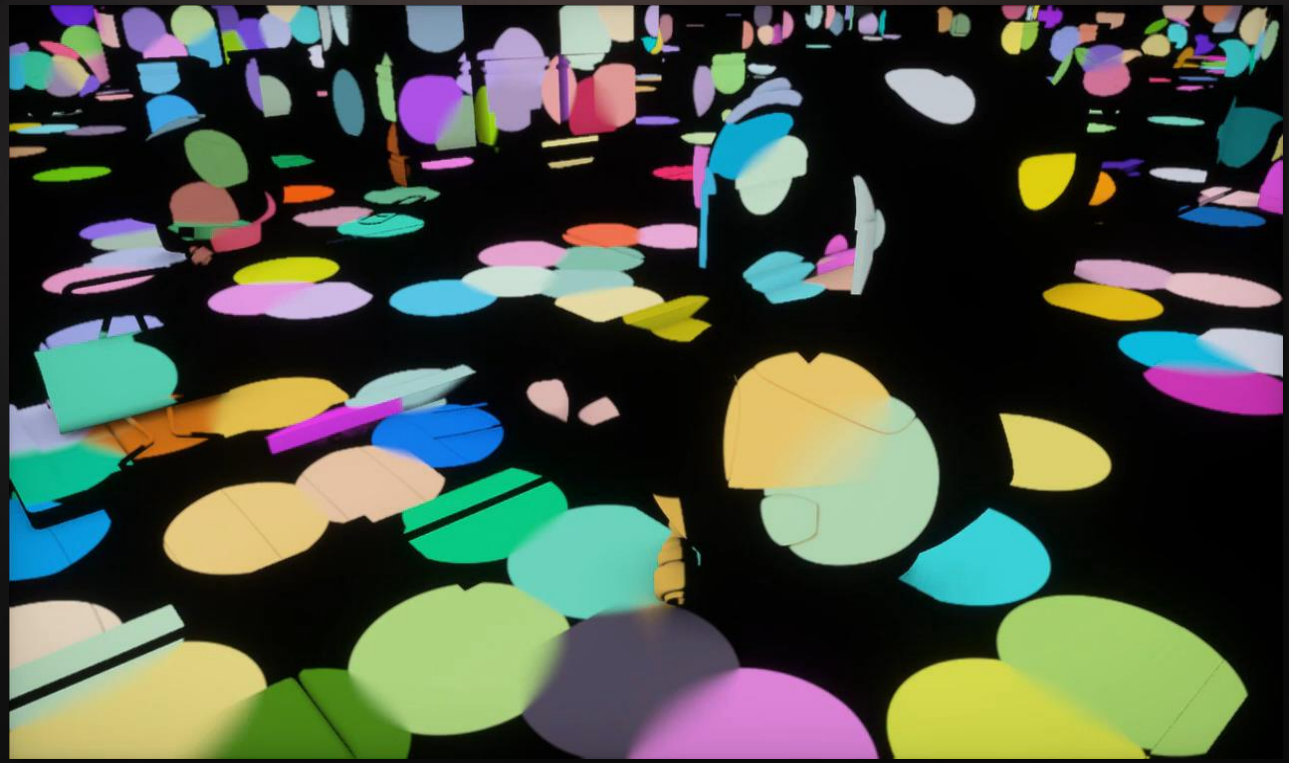


Combine with Screen Space AO [Jimenez 2016]

Surfel Placement

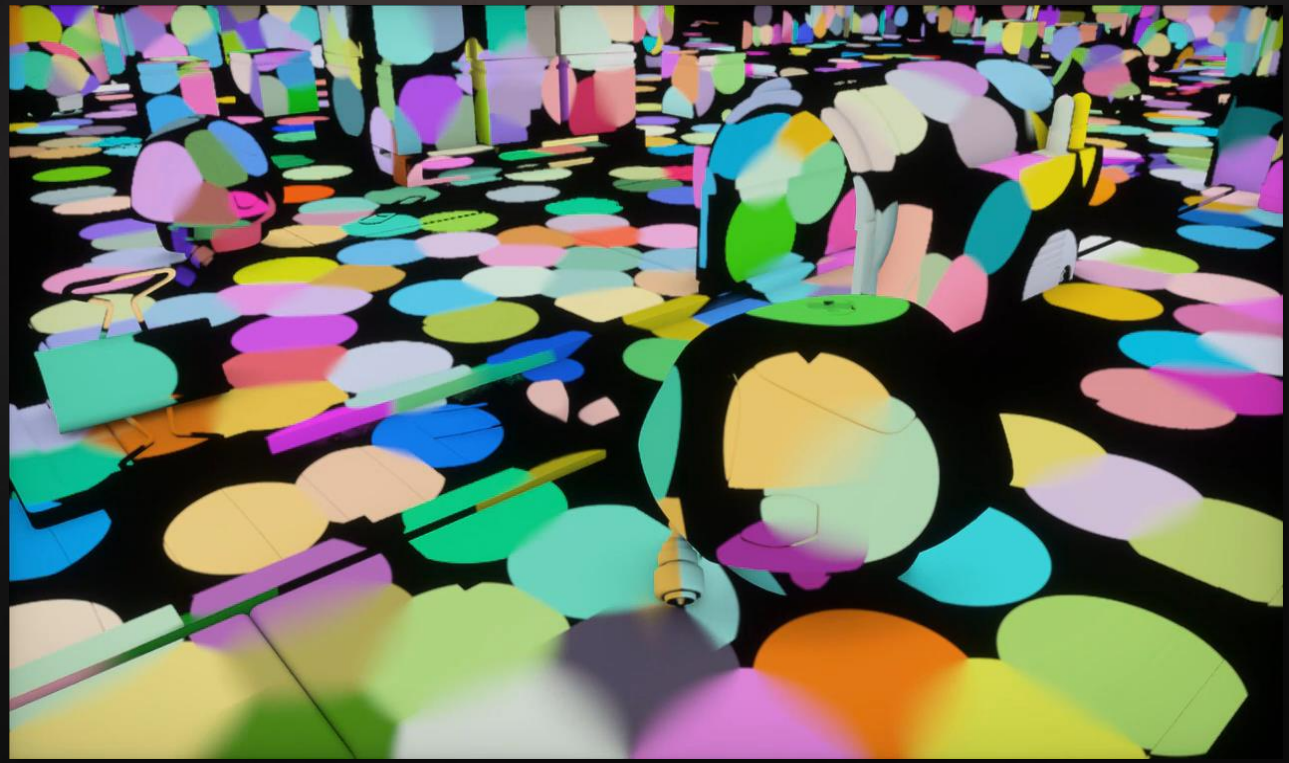
Surfel Spawning From Camera @ 1% speed

Surfel Placement



Surfel Spawning From Camera @ 1% speed

Surfel Placement



Surfel Spawning From Camera @ 1% speed

Surfel Placement Algorithm

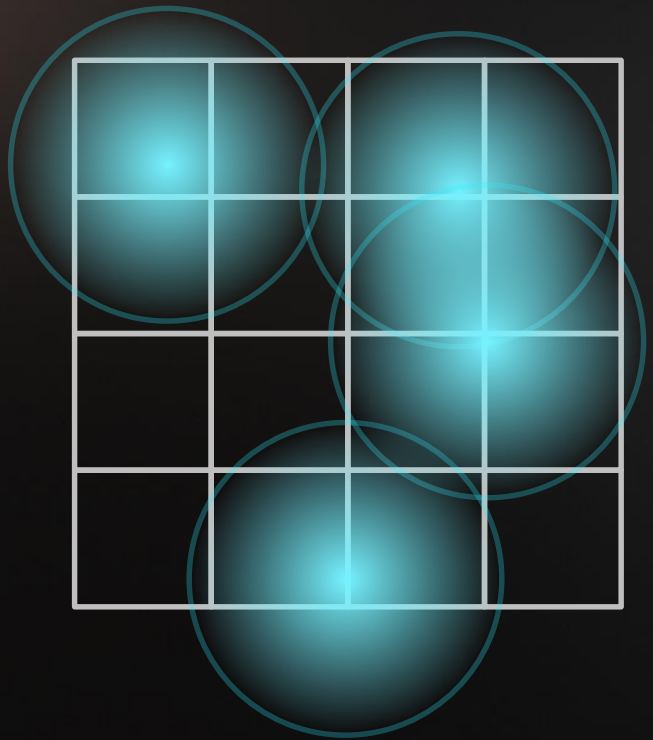
- Iterative screen-space hole filling

Surfel Placement Algorithm

- Iterative screen-space hole filling
- Calculate surfel coverage for each pixel

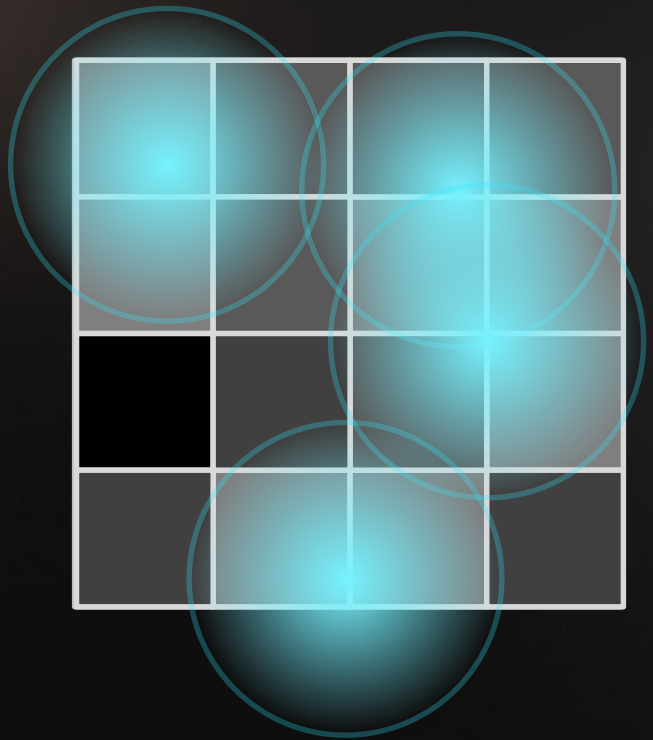
Surfel Placement Algorithm

- Iterative screen-space hole filling
- Calculate surfel coverage for each pixel



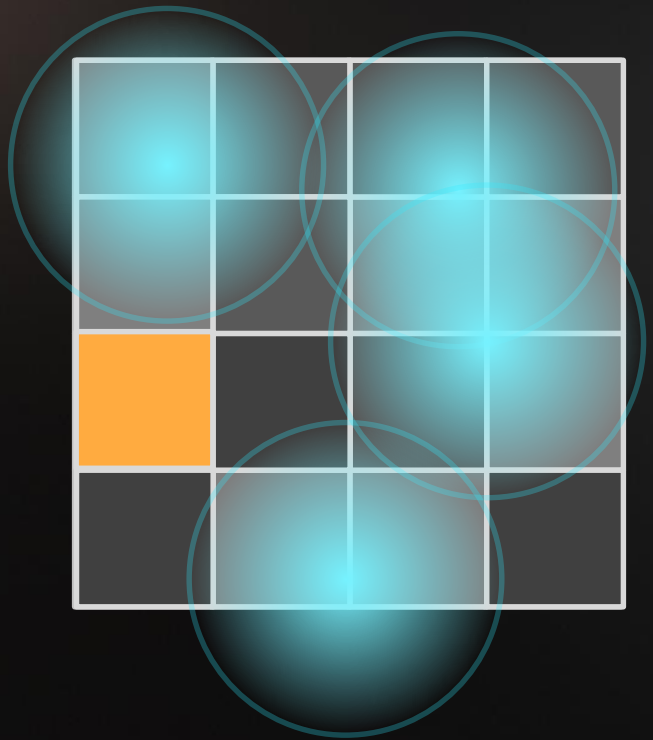
Surfel Placement Algorithm

- Iterative screen-space hole filling
- Calculate surfel coverage for each pixel



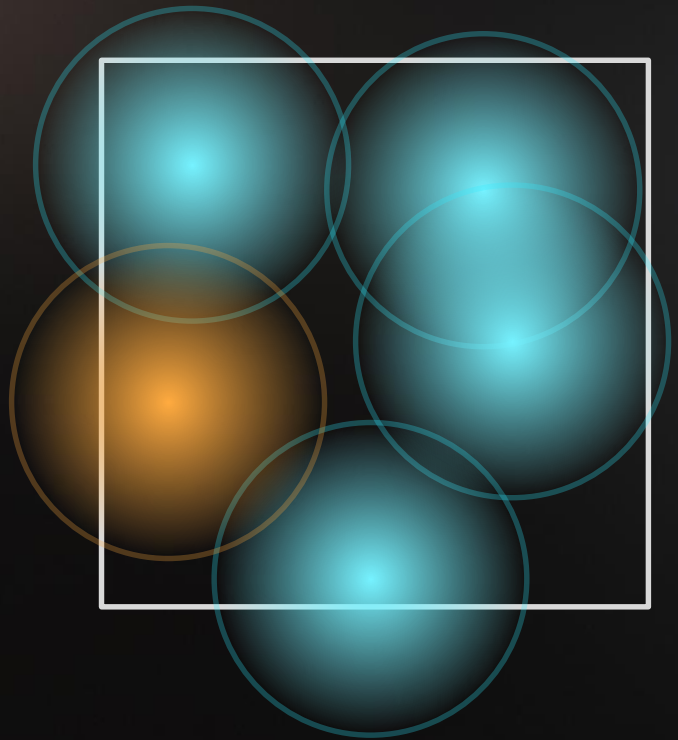
Surfel Placement Algorithm

- Iterative screen-space hole filling
- Calculate surfel coverage for each pixel
- Find lowest coverage in tile (16x16 pixels)



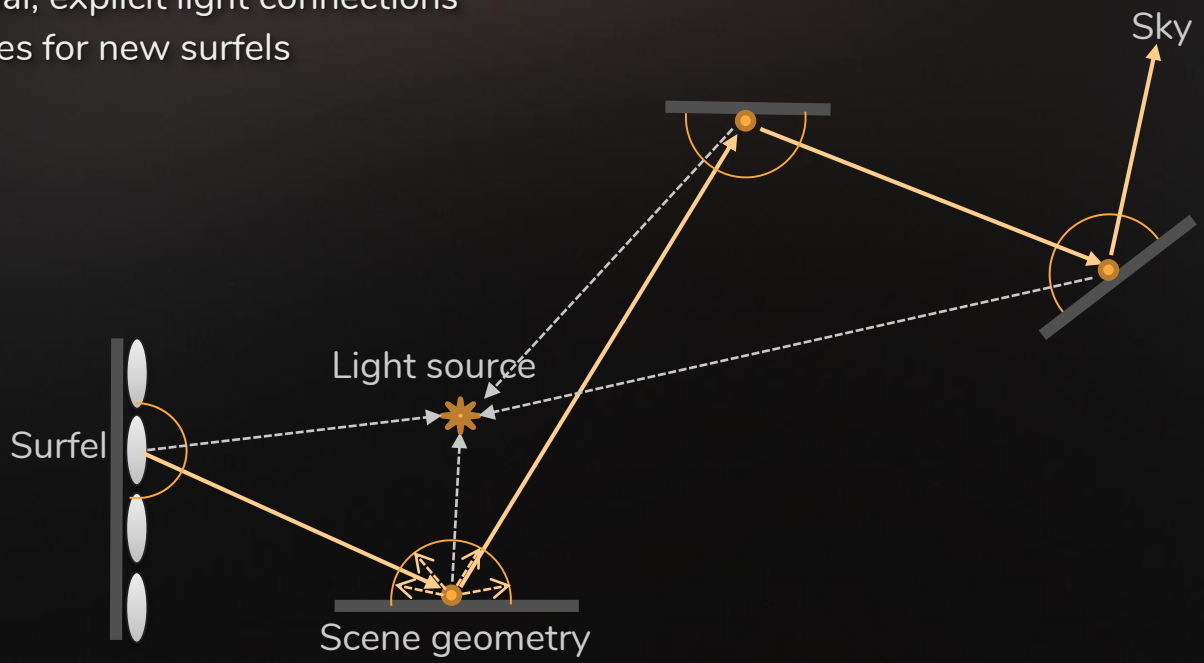
Surfel Placement Algorithm

- Iterative screen-space hole filling
- Calculate surfel coverage for each pixel
- Find lowest coverage in tile (16x16 pixels)
- Probabilistically spawn surfel on pixel
  - Chance proportional to pixel's projected area
  - G-Buffer depth and normal
- Continue where coverage below threshold



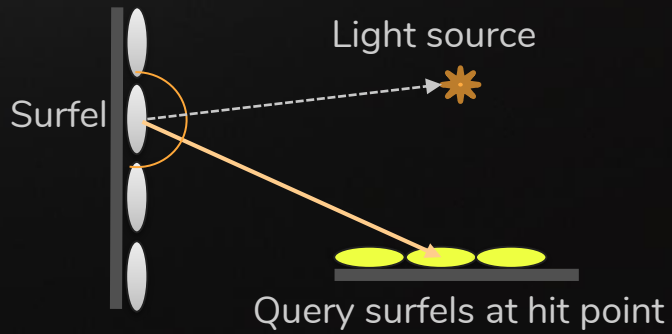
Irradiance Calculation

- Path trace from surfels
  - Unidirectional, explicit light connections
  - More samples for new surfels



Irradiance Calculation

- Path trace from surfels
  - Unidirectional, explicit light connections
  - More samples for new surfels
- Limit light path length
  - Sample previous GI at last bounce
- "Infinite" bounces amortized over frames



Monte Carlo Integration

- Monte Carlo assumes immutable integrand
- Plain old mean estimation

$$\bar{x}_0 = 0$$

$$\bar{x}_{n+1} = lerp\left(\bar{x}_n, x_{n+1}, \frac{1}{n+1}\right)$$

- Can't use in dynamic GI
- Hard reset undesirable

Interactive GI in Frostbite [Hillaire 2018]

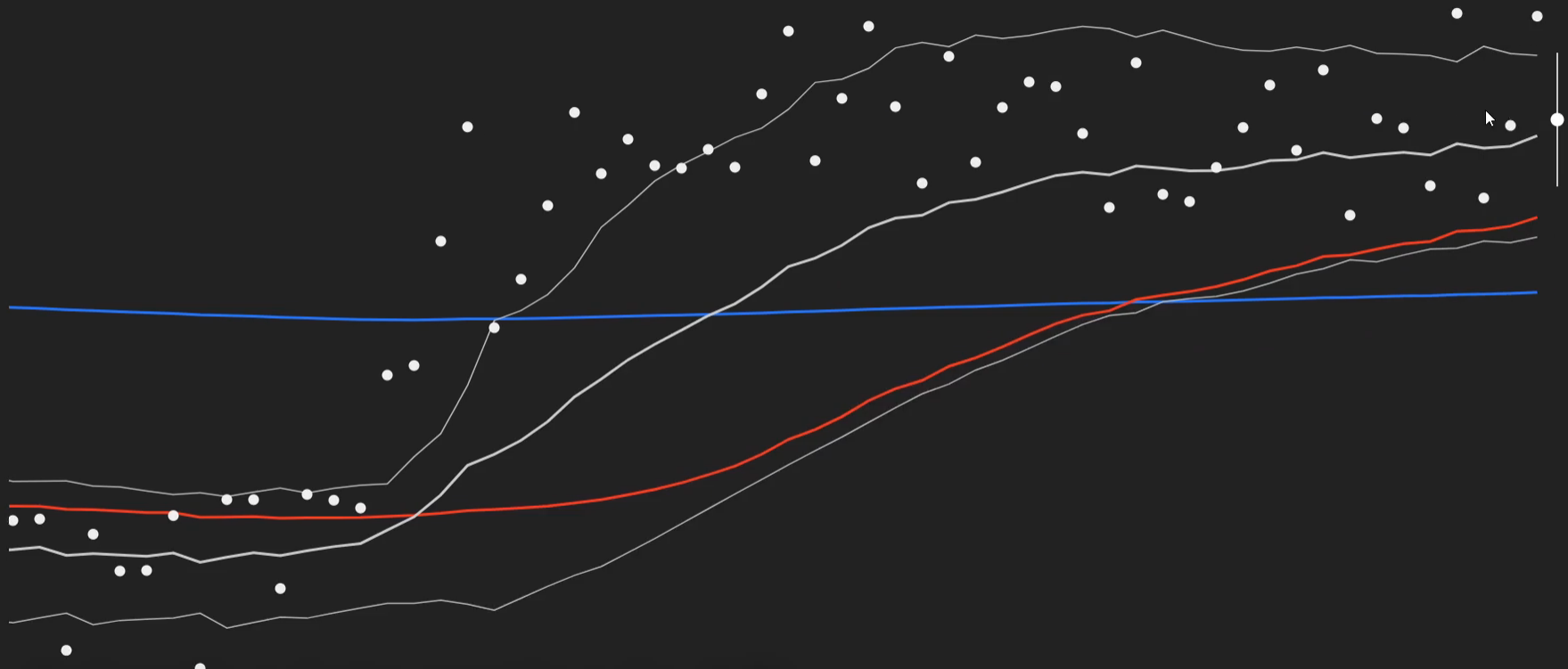
Adaptive Integration

- Give up on fully converging
- Exponential moving average

$$\bar{x}_0 = 0$$

$$\bar{x}_{n+1} = lerp(\bar{x}_n, x_{n+1}, k)$$

- Estimate short-term statistics
  - Mean, variance
- Adapt blend factor k
  - Reduce if variance high
  - Increase if mean far from short-term statistics



Adaptive integration in 1D

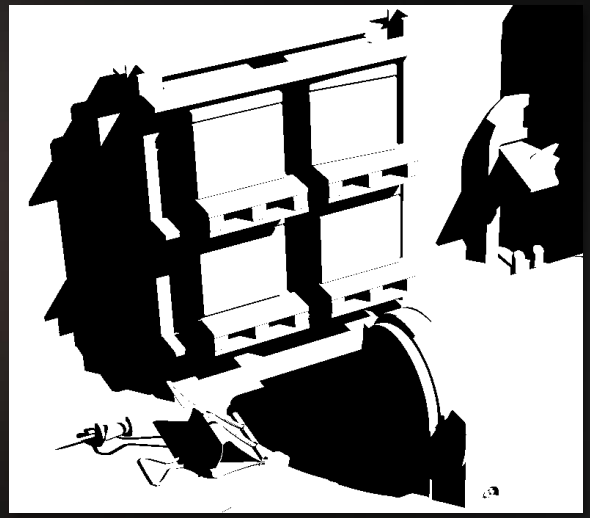
Adaptive mean

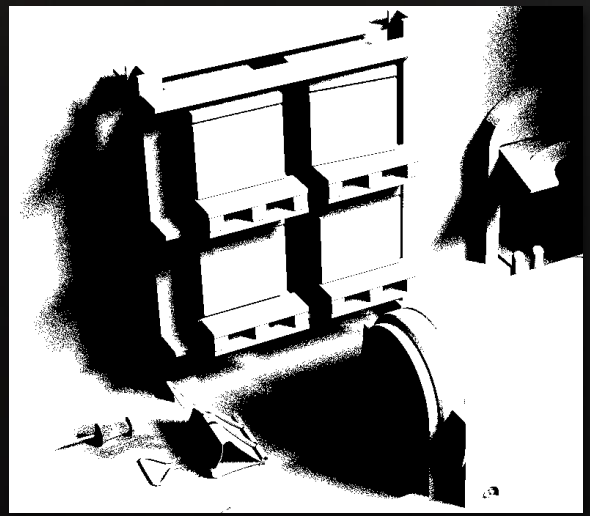
Exponential moving average

Short-term
distribution bounds

Shadows

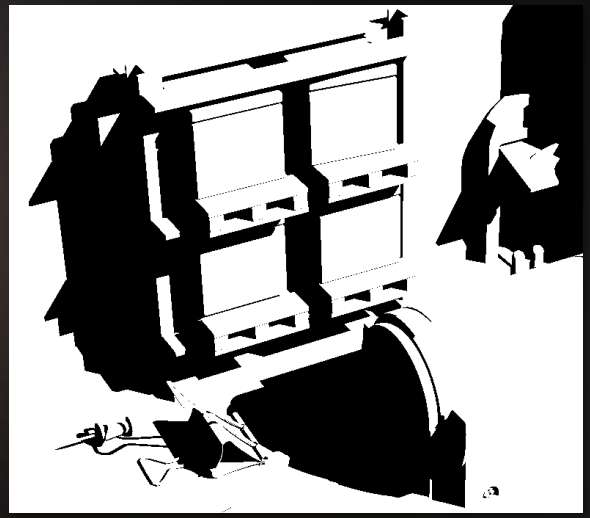
- Hard shadows trivial
  - Launch a ray towards light position
  - Check for any hit
- Soft shadows easy
  - Sample direction from uniform cone [PBRT]
  - Cone width determines penumbra

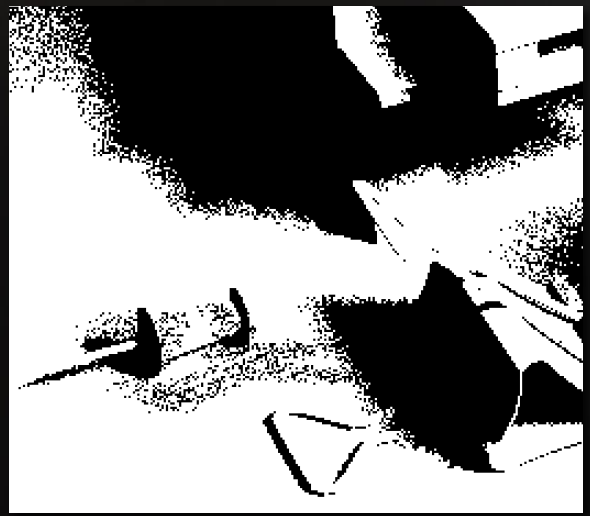




Shadows

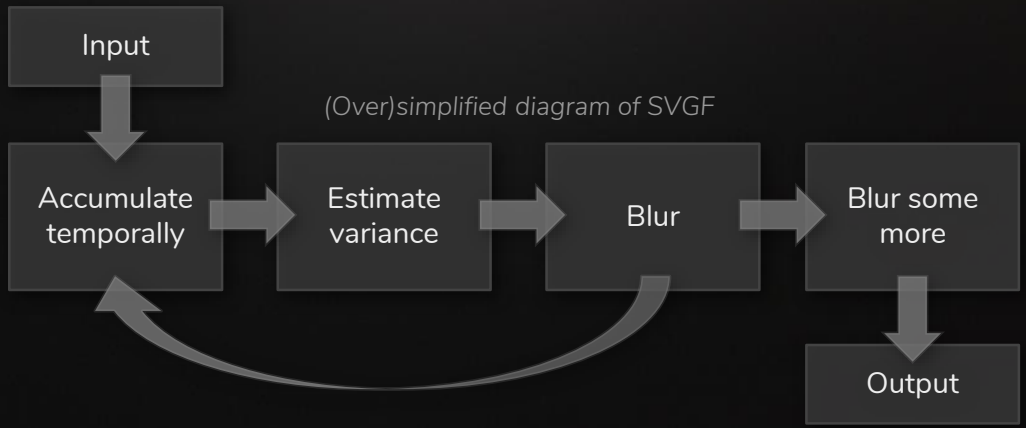
- Hard shadows trivial
  - Launch a ray towards light position
  - Check for any hit
- Soft shadows easy
  - Sample direction from uniform cone [PBRT]
  - Cone width determines penumbra
- … Easy but noisy
  - 1 visibility sample per pixel not enough
  - Filter instead of shooting more

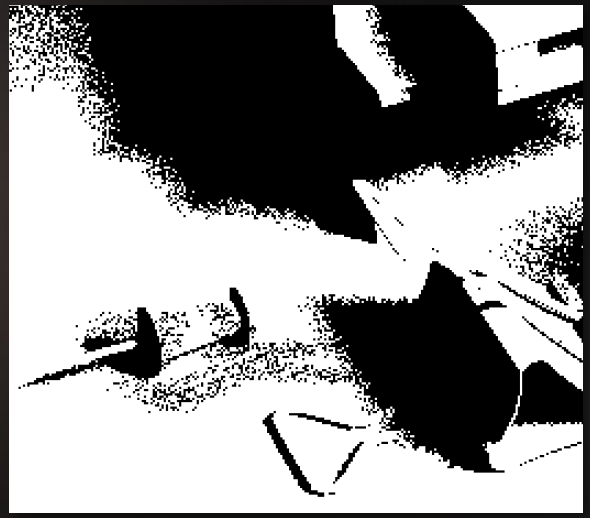


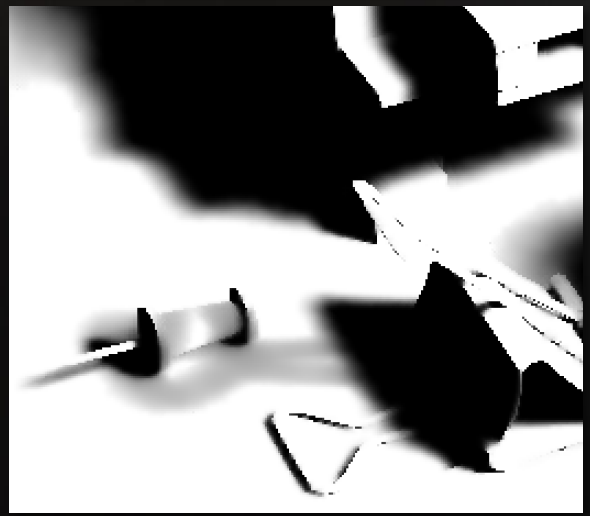


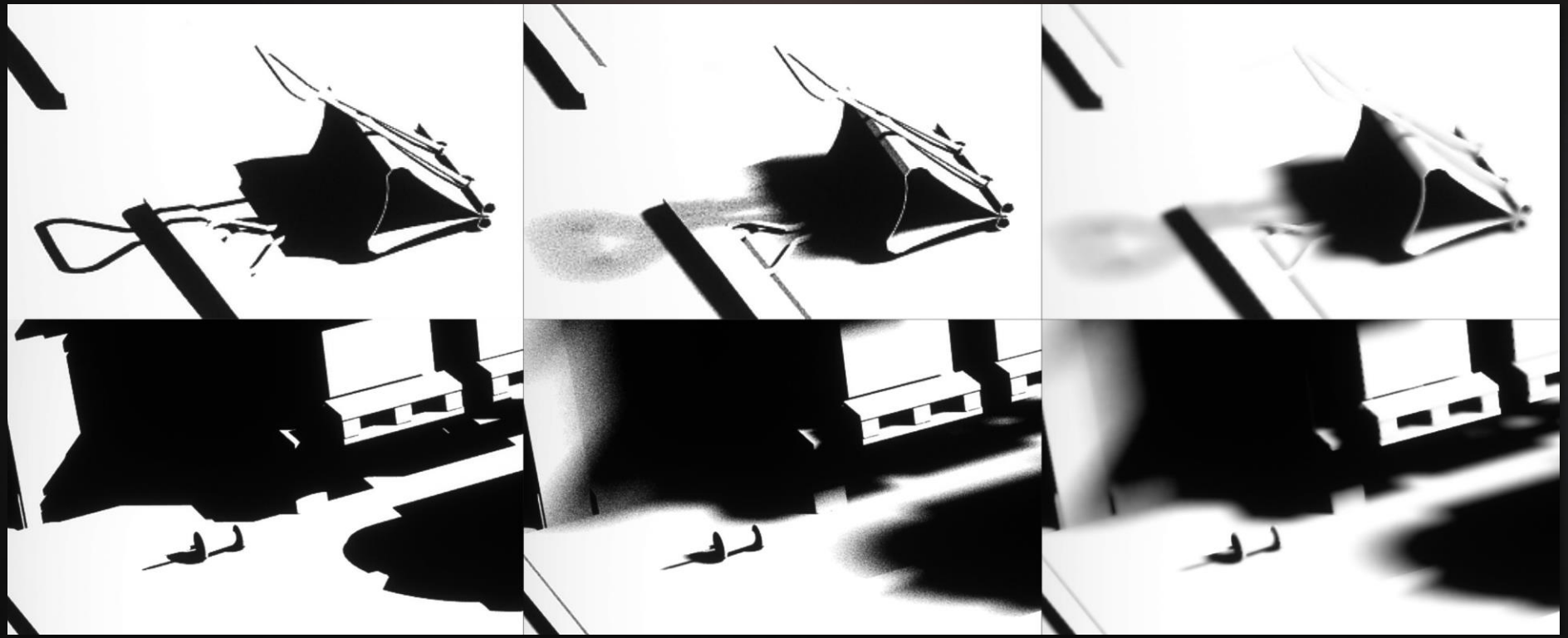
Shadow Filtering

- Based on Nvidia's SVGF [Schied 2017]
  - Temporal accumulation
  - Multi-pass weighted blur
    - Kernel size driven by variance estimate
  - No shadow-specific heuristics









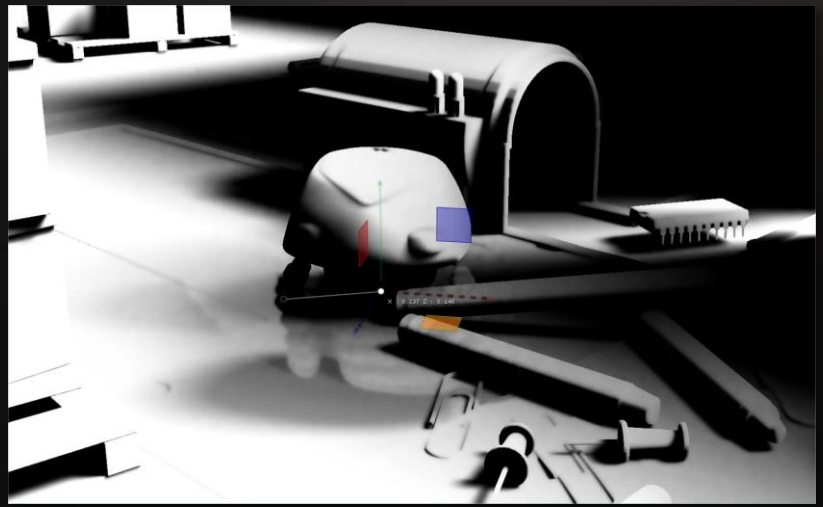
Hard Raytraced Shadows

Soft Raytraced Shadows (Unfiltered)

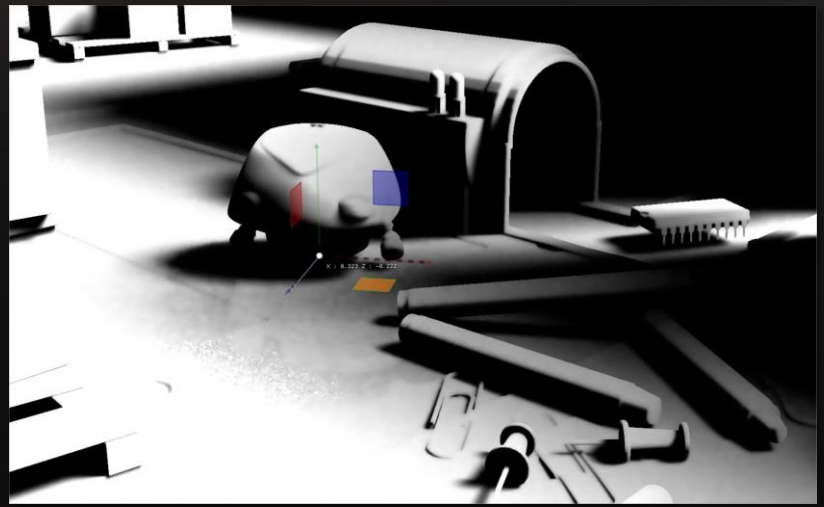
Soft Raytraced Shadows (Filtered)

Shadow Filtering

- Modified to reduce ghosting
  - TAA-like color clip
    - Variance-based bounds (5x5 kernel) [Salvi 2016]



Baseline SVGF



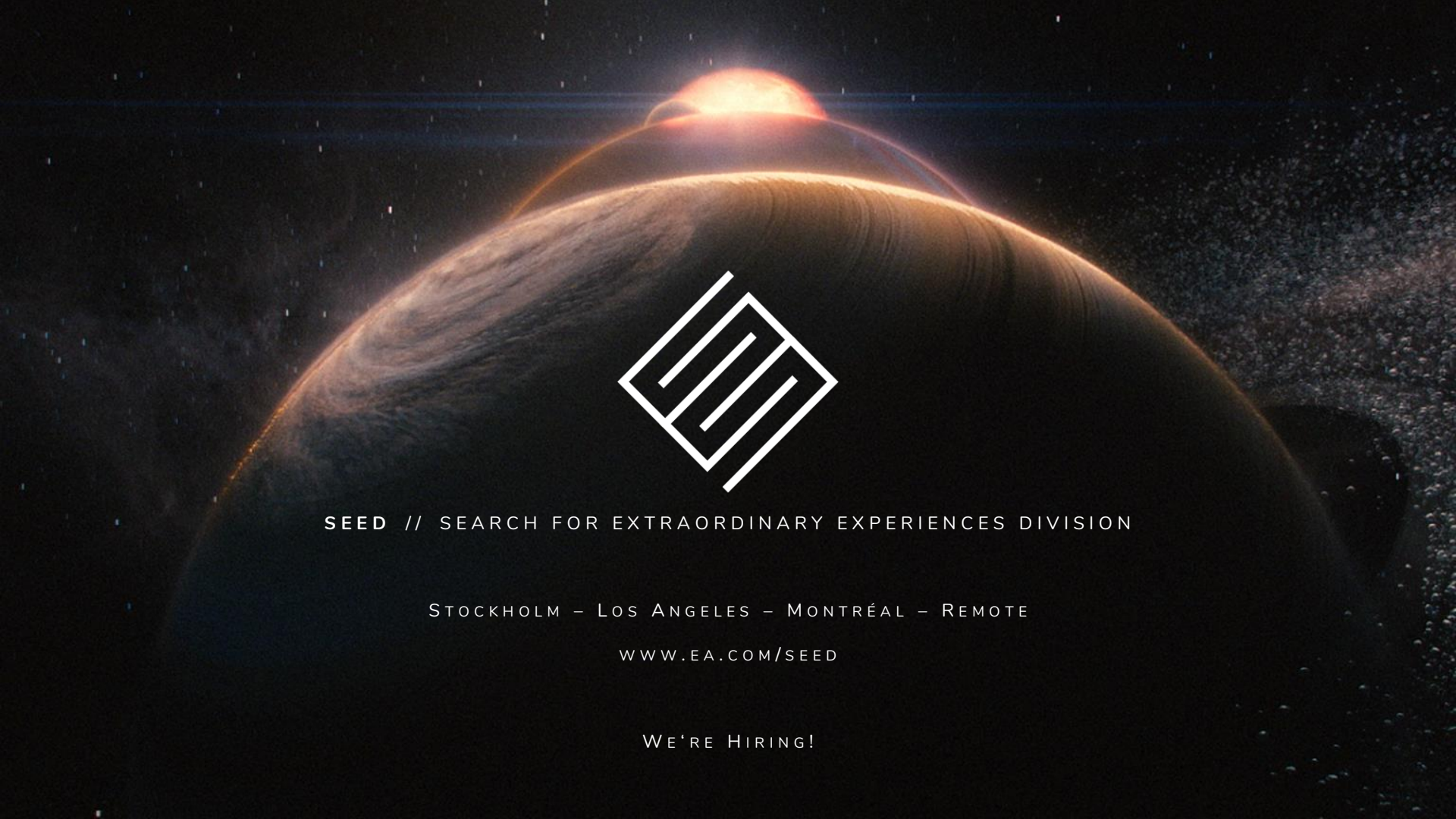
+TAA-like clip

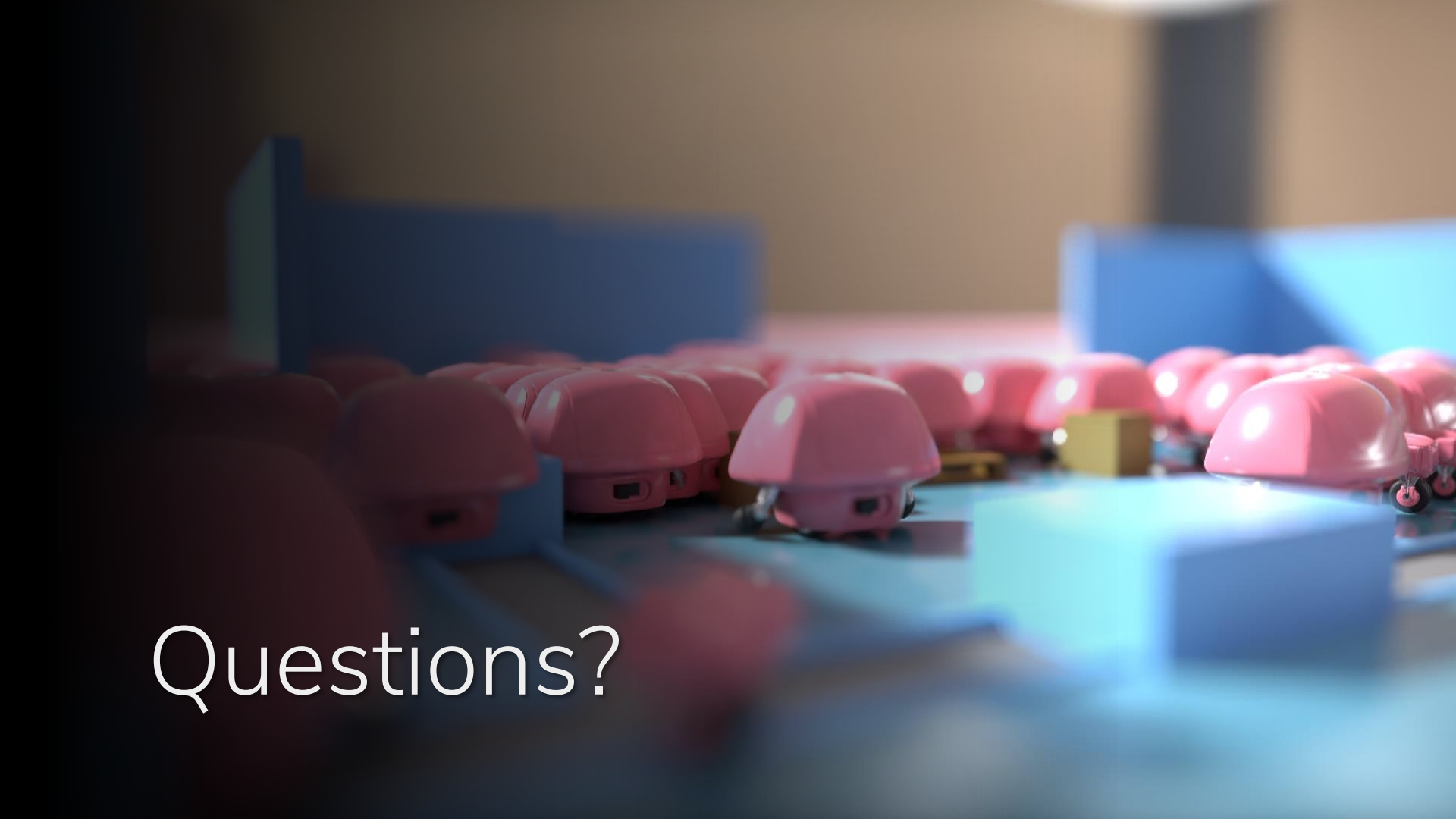
Summary

- Replace fine-tuned hacks with unified approaches
  - Reconstruction and filtering instead
- New tool in the box
  - Solve sparse and incoherent problems
  - Not a silver bullet
- Use rays wisely
  - Pica Pica shoots ~2.25 rays per pixel
- Real-time visuals with (almost) path traced quality achievable today

Thanks

- SEED
  - Johan Andersson
  - Colin Barré-Brisebois
  - Jasper Bekkers
  - Joakim Bergdahl
  - Ken Brown
  - Dean Calver
  - Dirk de la Hunt
  - Jenna Frisk
  - Paul Greveson
  - Henrik Halen
  - Effeli Holst
  - Andrew Lauritzen
  - Magnus Nordin
  - Niklas Nummelin
  - Anastasia Opara
  - Kristoffer Sjöö
  - Ida Winterhaven
  - Graham Wihlidal
- Microsoft
  - Chas Boyd
  - Ivan Nevraev
  - Amar Patel
  - Matt Sandy
- NVIDIA
  - Tomas Akenine-Möller
  - Nir Benty
  - Jiho Choi
  - Peter Harrison
  - Alex Hyder
  - Jon Jansen
  - Aaron Lefohn
  - Ignacio Llamas
  - Henry Moreton
  - Martin Stich


SEED // SEARCH FOR EXTRAORDINARY EXPERIENCES DIVISION
STOCKHOLM – LOS ANGELES – MONTRÉAL – REMOTE
WWW.EA.COM/SEED
WE'RE HIRING!


Questions?

References

- **[Harmer 2018]** Jack Harmer, Linus Gisslén, Henrik Holst, Joakim Bergdahl, Tom Olsson, Kristoffer Sjöö and Magnus Nordin“ Imitation Learning with Concurrent Actions in 3D Games”. available online
- **[Barré-Brisebois 2011]** Barré-Brisebois, Colin and Bouchard, Marc. “Approximating Translucency for a Fast, Cheap and Convincing Subsurface Scattering Look”, available online.
- **[Barré-Brisebois 2017]** Barré-Brisebois, Colin. “A Certain Slant of Light: Past, Present and Future Challenges of Global Illumination in Games”, available online.
- **[Igehy 1999]** Igehy, Homan. “Tracing Ray Differentials”, available online.
- **[PBRT]** Pharr, Matt. Jakob, Wenzel and Humphreys, Greg. “Physically Based Rendering”, Book, http://www.pbrt.org/.
- **[Schied 2017]** Schied, Christoph et. Al. “Spatiotemporal Variance-Guided Filtering: Real-Time Reconstruction for Path-Traced Global Illumination”, NVIDIA Research, available online.
- **[Stachowiak 2015]** Stachowiak, Tomasz. “Stochastic Screen-Space Reflections”, available online.
- **[Weidlich 2007]** Weidlich, Andrea and Wilkie, Alexander. “Arbitrarily Layered Micro-Facet Surfaces”, available online.
- **[Williams 1983]** Williams, Lance. “Pyramidal Parametrics”, available online.
- **[Robison 2009]** Austin Robison, Peter Shirley. "Image Space Gathering", available online.
- **[Heitz 2018]** Eric Heitz, Stephen Hill, Morgan McGuire. "Combining Analytic Direct Illumination and Stochastic Shadows", available online.
- **[Karis 2014]** Brian Karis. "High-Quality Temporal Supersampling", available online.
- **[Lehtinen 2008]** Jaakko Lehtinen, Matthias Zwicker, Emmanuel Turquin, Janne Kontkanen, Frédo Durand, François Sillion, Timo Aila. "A Meshless Hierarchical
- Representation for Light Transport“, available online.
- **[Jimenez 2016]** Jorge Jimenez, Xian-Chun Wu, Angelo Pesce, Adrian Jarabo. "Practical Realtime Strategies for Accurate Indirect Occlusion", available online.
- **[Opara 2018]** Anastasia Opara. “Creativity of Rules and Patterns”, available online.
- **[Salvi 2016]** Marco Salvi. "An excursion in temporal super sampling", available online.
- **[Hillaire 2018]** Sébastien Hillaire. "Real-time Raytracing for Interactive Global Illumination Workflows in Frostbite", available online.

Bonus

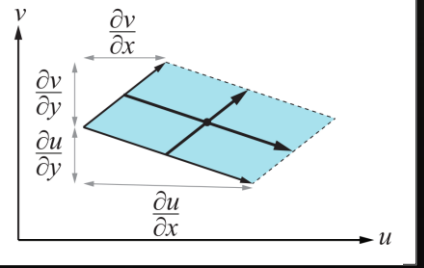
Texture Level-of-Detail

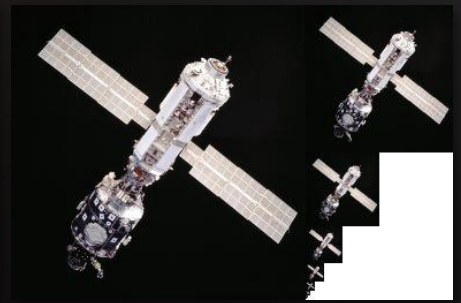
What about texture level of detail?

- Mipmapping [Williams 1983] is the standard method to avoid texture aliasing:

$$\lambda(x,y) = \log_2 \lceil \rho(x,y) \rceil$$

$$\rho(x,y) = \max\left\{ \sqrt{\left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial x}\right)^2}, \sqrt{\left(\frac{\partial u}{\partial y}\right)^2 + \left(\frac{\partial v}{\partial y}\right)^2} \right\}$$





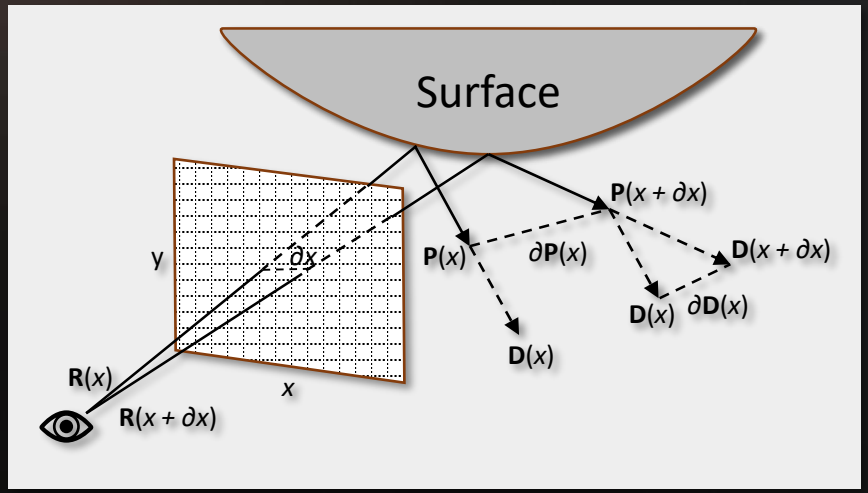
Left: level-of-detail (λ), partial derivatives and the parallelogram-approximated texture-space footprint of a pixel. Right: mipmap chain

- Screen-space pixel maps to approximately one texel in the mipmap hierarchy
- Supported by all GPUs for rasterization via shading quad and derivatives

Texture Level-of-Detail

No shading quads for ray tracing!

- Traditionaly: *Ray Differentials*
  - Estimates the footprint of a pixel by computing world-space derivatives of the ray with respect to the image plane
  - Have to differentiate (virtual offset) rays
  - Heavier payload (12 floats) for subsequent rays (can) affect performance. Optimize!



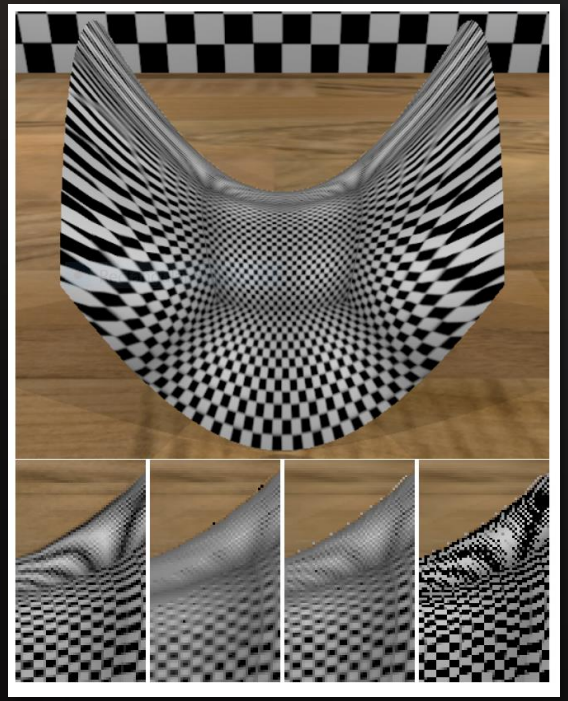
Ray Differentials [Igehy99]

- Alternative: always sample mip 0 with bilinear filtering (with extra samples)
  - Leads to aliasing and additional performance cost

Texture Level-of-Detail

Together with NVIDIA Research, we developed a texture LOD technique for raytracing:

- Heuristic based on ***triangle properties***, a ***curvature estimate***, ***distance***, and ***incident angle***
  - Similar quality to *ray differentials* with single trilinear lookup
  - Single value stored in the payload for subsequent rays
- Upcoming publication by:
  - *Tomas Akenine-Möller (NV), Magnus Andersson (NV), Colin Barré-Brisebois (EA), Jim Nilsson (NV), Robert Toth (NV)*



Ground Truth, Ray Differentials, Ours, Mip0

mGPU

- Explicit Heterogenous Multi-GPU
- Parallel Fork-Join Style
- Resources copied through system memory using copy queue
- Minimize PCI-E transfers
- Approach
  - Run ray generation on primary GPU
  - Copy results in sub-regions to other GPUs
  - Run tracing phases on separate GPUs
  - Copy tracing results back to primary GPU
  - Run filtering on primary GPU