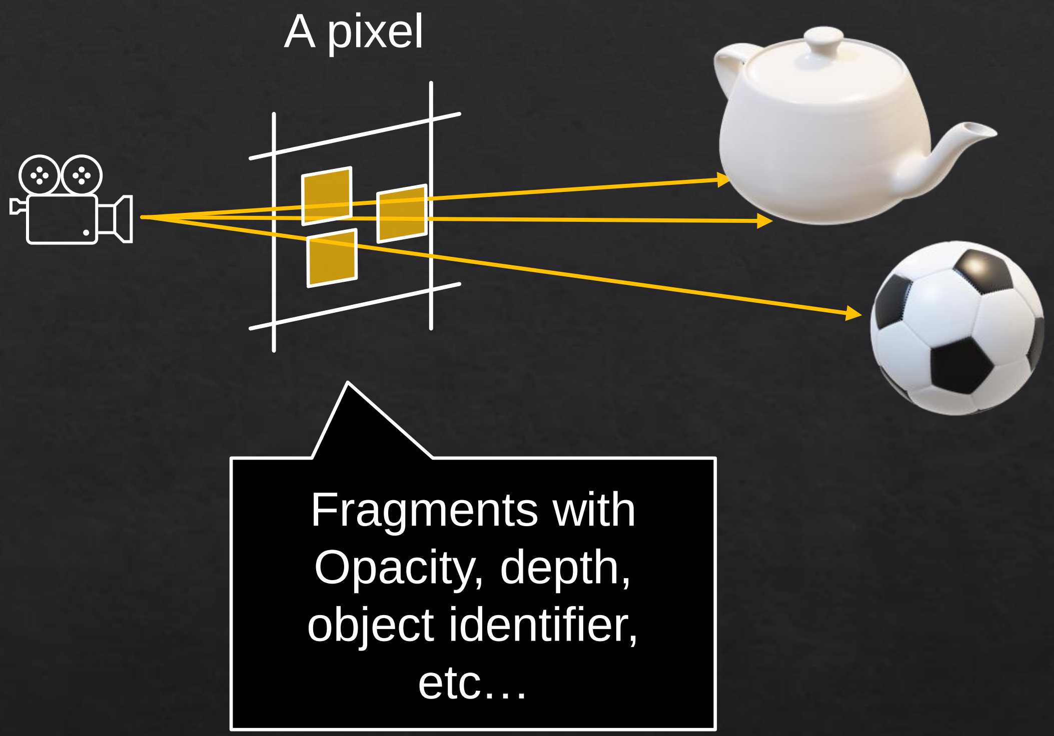
Multi-Fragment Rendering for Glossy Bounces on the GPU

Atsushi Yoshimura Yusuke Tokuyoshi
Takahiro Harada

Advanced Micro Devices, Inc.

Multi-Fragment Rendering

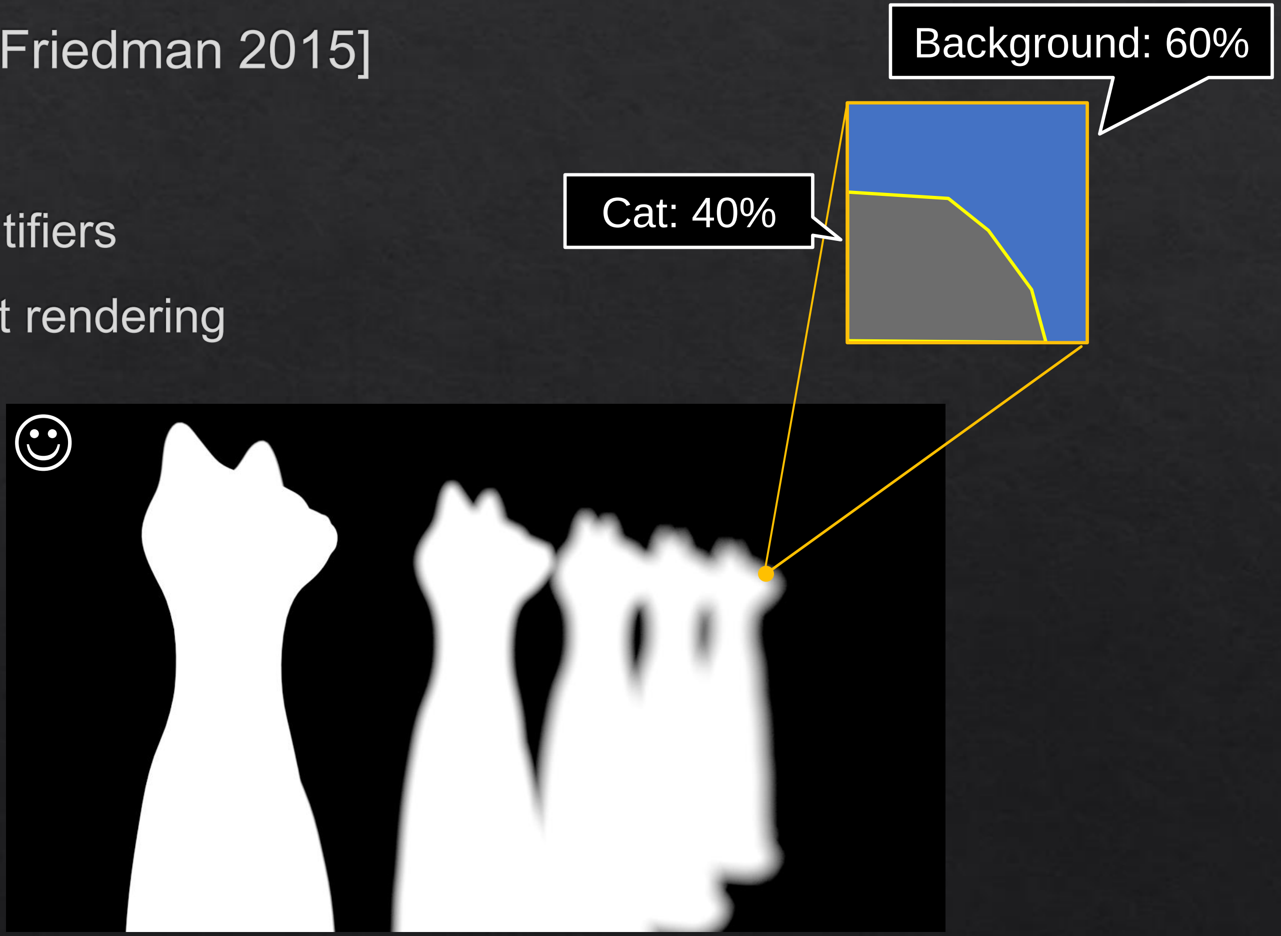
- Multiple data in a single pixel
- Wide variety of applications
  - Order-Independent Transparency
  - Anti-Aliasing
  - Defocus Blur



Matte generation in AMD Radeon™ ProRender

- We use Cryptomatte for matte generation [Friedman 2015]
- Flexible matte generation
  - Matte from a combination of any object identifiers
  - Provides a storage format for multi-fragment rendering

Rendered image

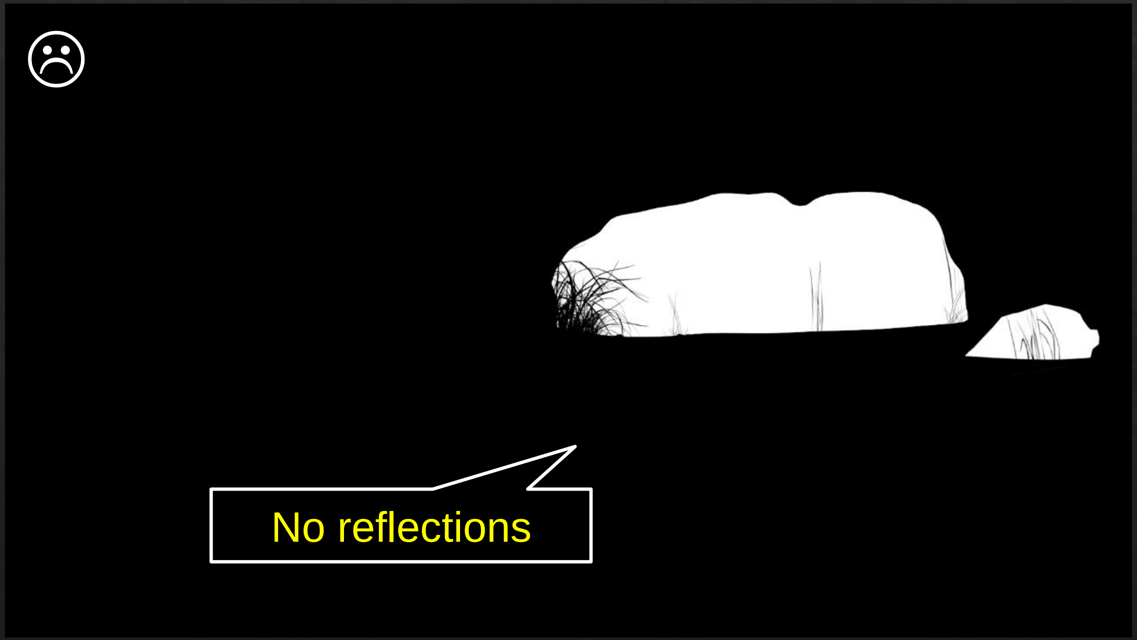


Matte

Limitation: Glossy bounces

- Existing work mainly focus the primary visible fragments
  - But the matte should represent the object

Rendered image



Matte of the rocks

⇨ Extend to glossy bounces including indirect visibility

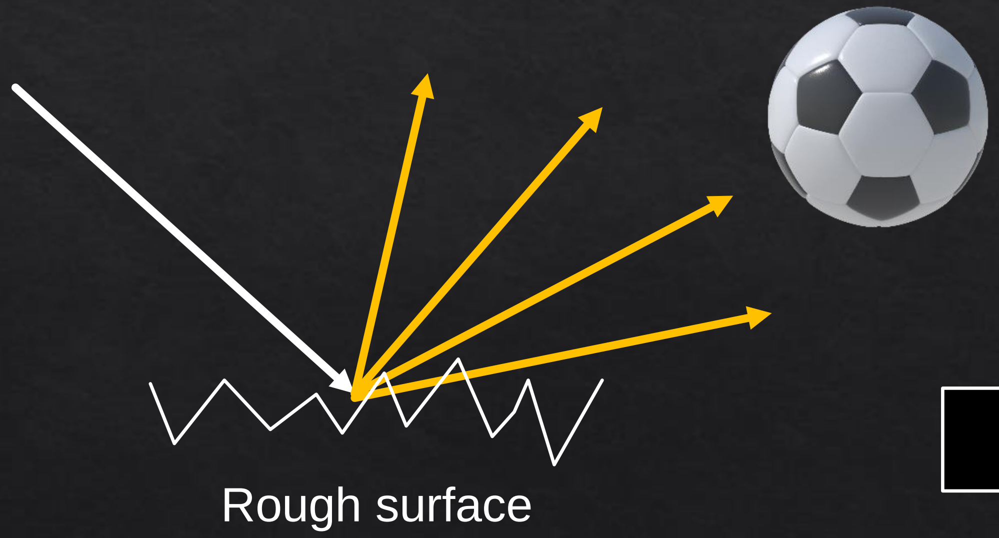
Our contributions

- A computation method for fragment coverage in multiple bounces using a weighting function
- A weighting function that penalizes the coverage of a fragment with less visibility of object details
- The implementation details of our coverage update process on the GPU
  - Please refer to our paper

Our Approach

Our coverage computation

- Scattering of rays blurs details

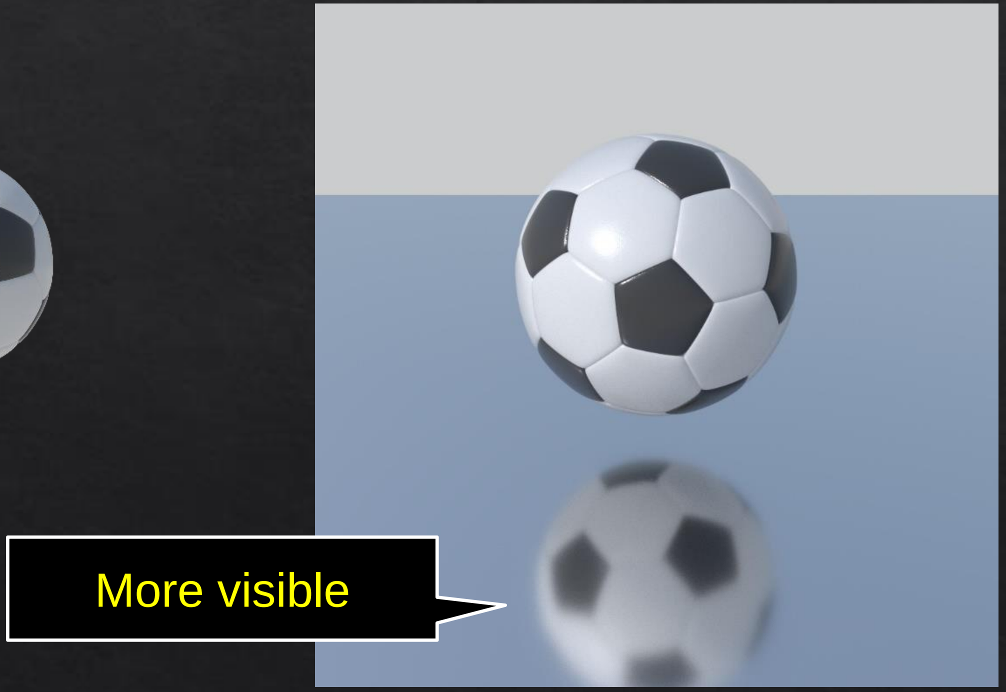


Moderately glossy

Our coverage computation

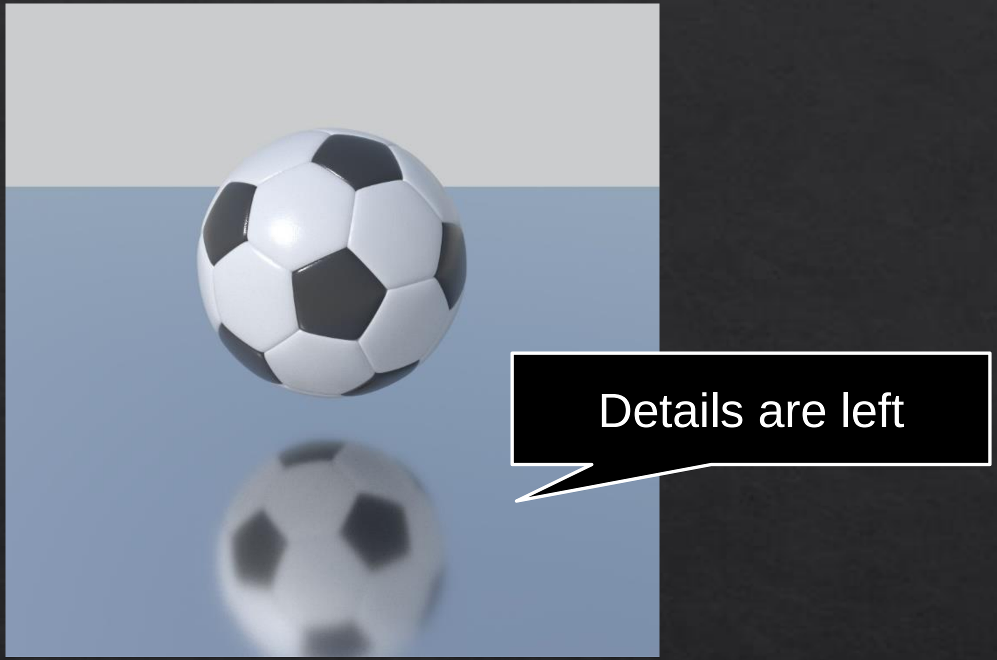
- Scattering of rays blurs details
- The coverage of fragments for the object as an amount of visibility of details

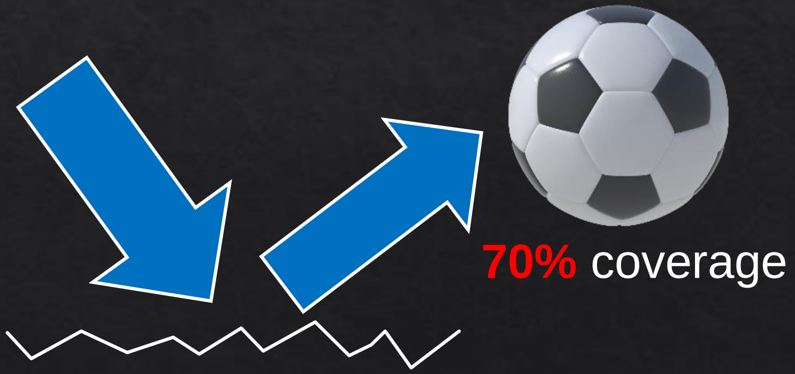
Smoother surface



Highly glossy

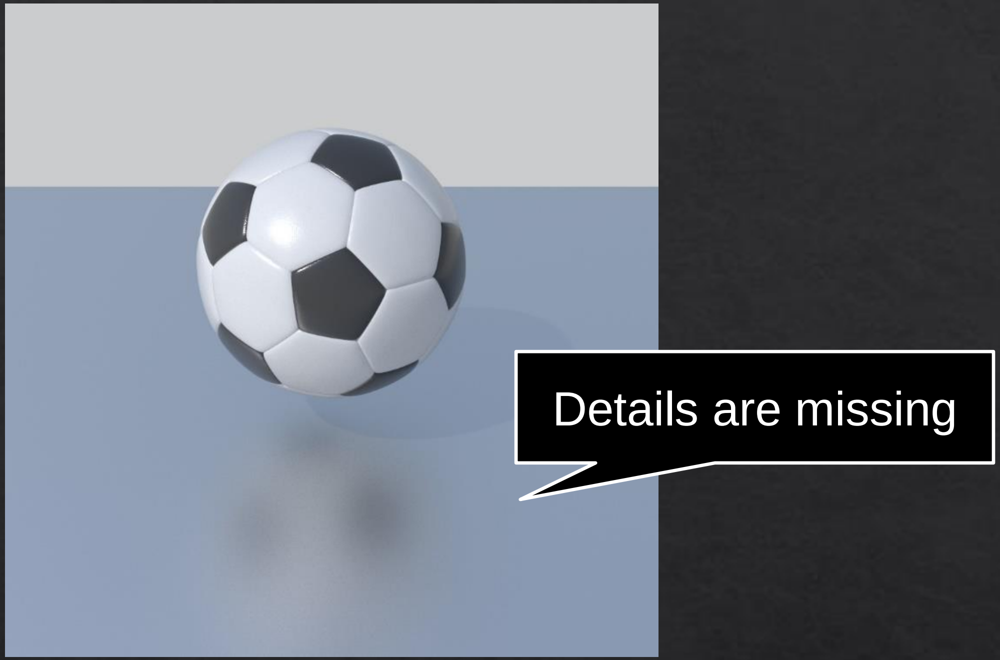
Highly glossy

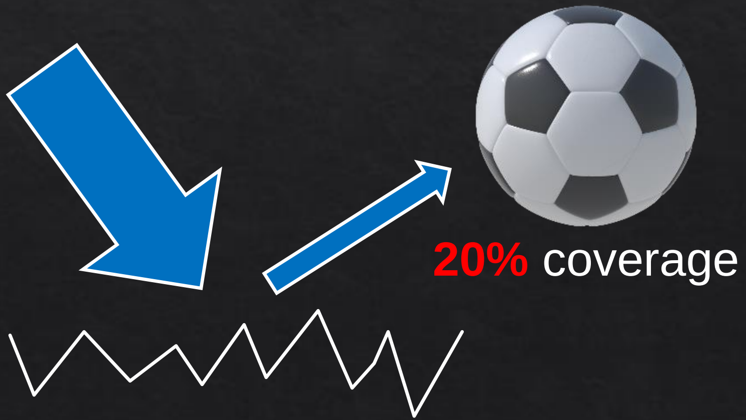




100% - 70 % = 30% coverage

Moderately glossy



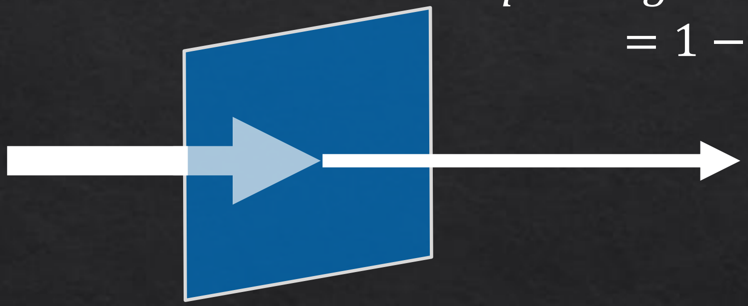


100% - 20 % = 80% coverage

The amount of passing through

Existing work

$$passing\ through = 1 - opacity$$



$$coverage = opacity$$

Our passing through on glossy bounce

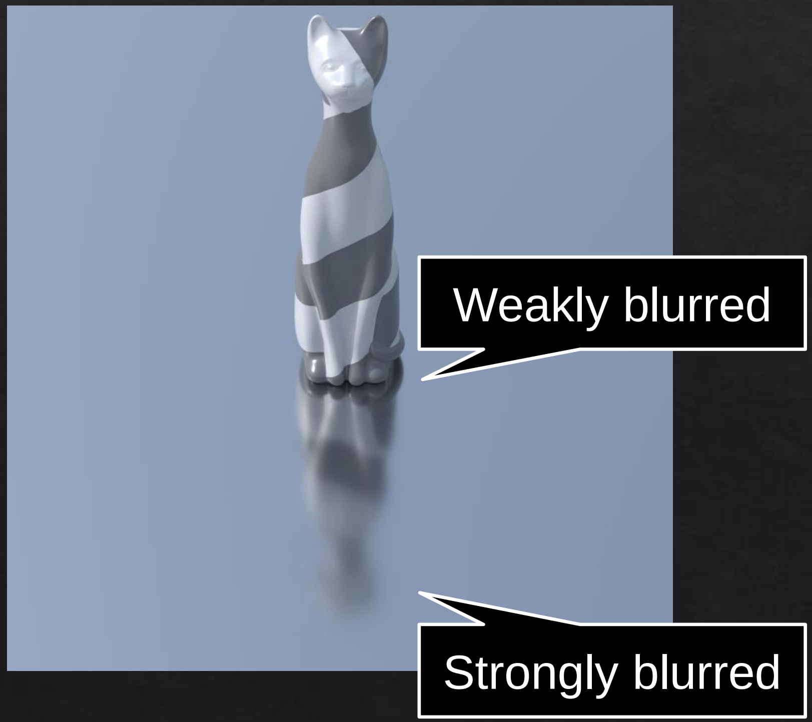
$$passing\ through = \int_{\Omega} W(x,\omega,\omega')f(x,\omega,\omega')(\omega' \cdot n)d\omega'$$

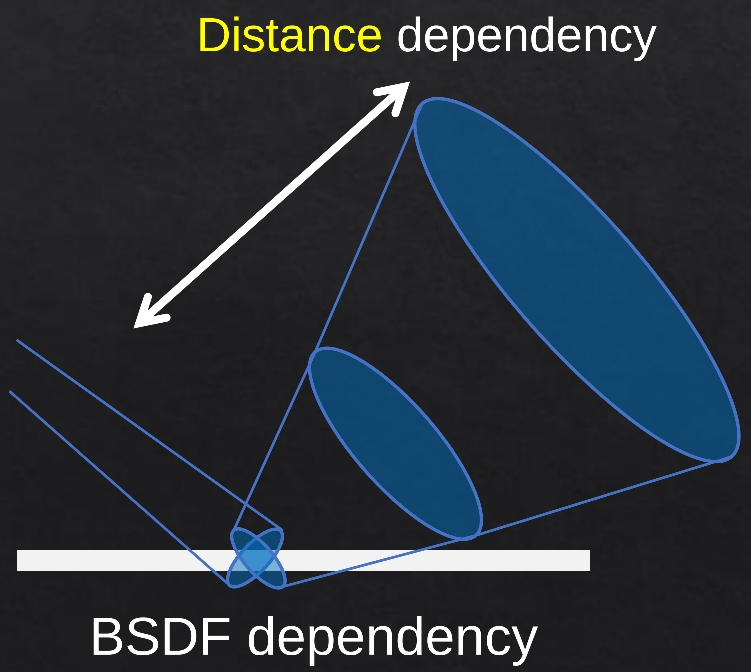
$$coverage = 1 - \int_{\Omega} W(x,\omega,\omega')f(x,\omega,\omega')(\omega' \cdot n)d\omega'$$

- Describe the amount of passing through using the integral of the product of a BSDF, cosine term, and weighting function W
- The weighting function W is designed to controls the passage

Our metric for the weighting function

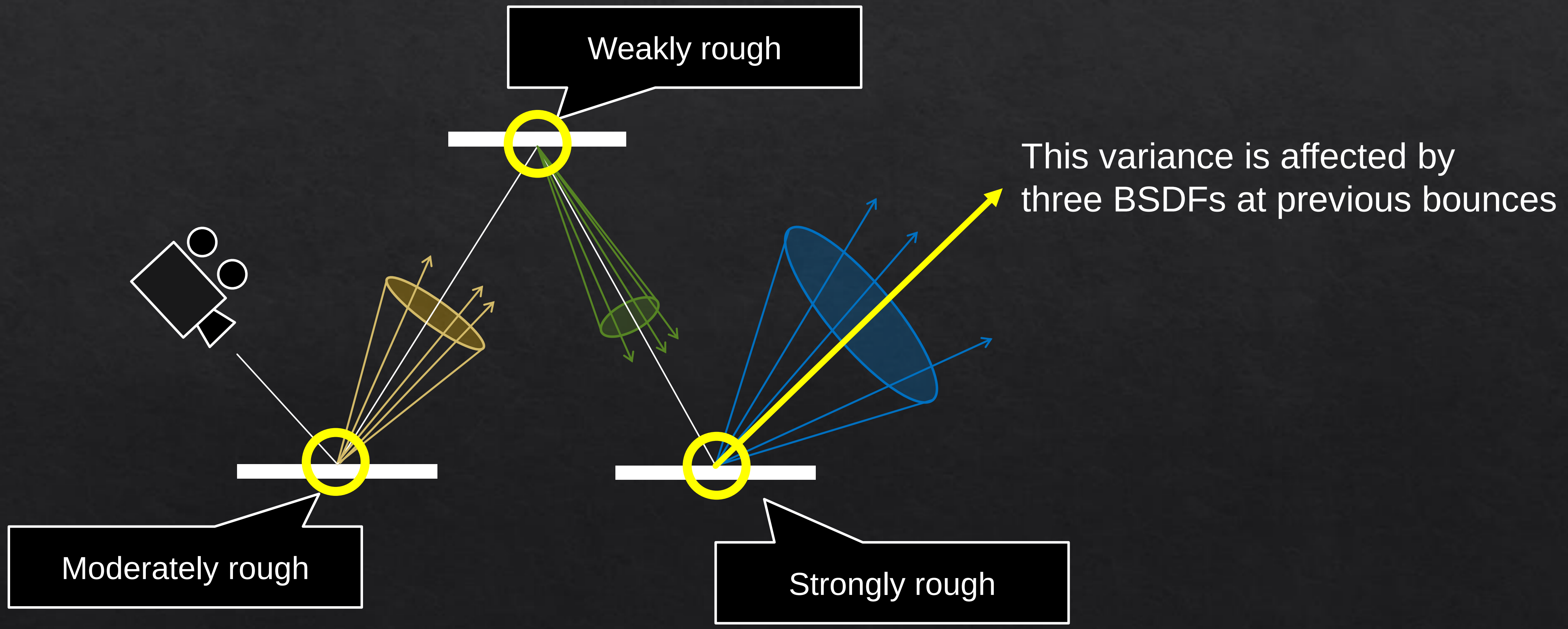
- Need a new metric - how much object details are lost due to the BSDF
- Depends on the BSDF and **the distance** from previous shading point
- Use the approach of ray cones [Akenine 2021] and an idea of ray variance from the BSDF





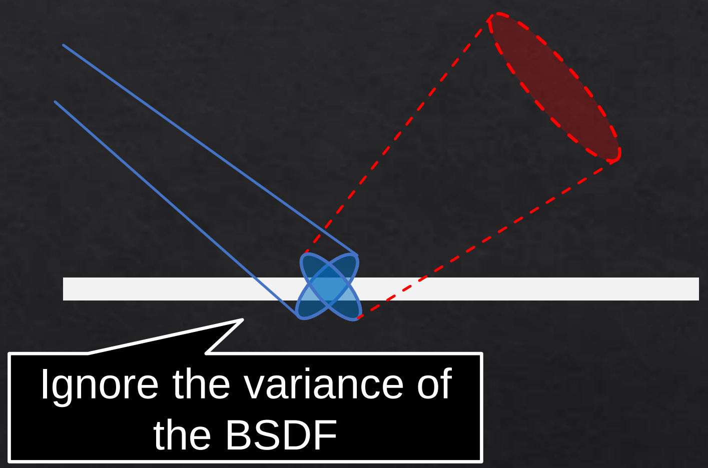
The variance of rays

- The variance of rays increases at every bounce due to the BSDF



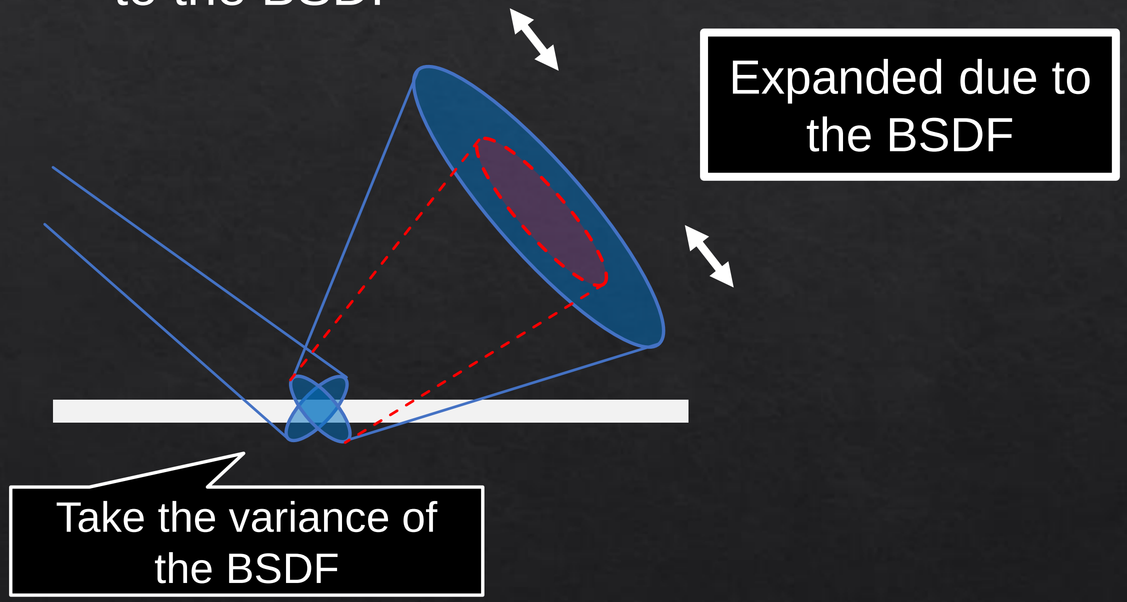
Our metric – cone ratio

Cone without the spread due to the BSDF



:

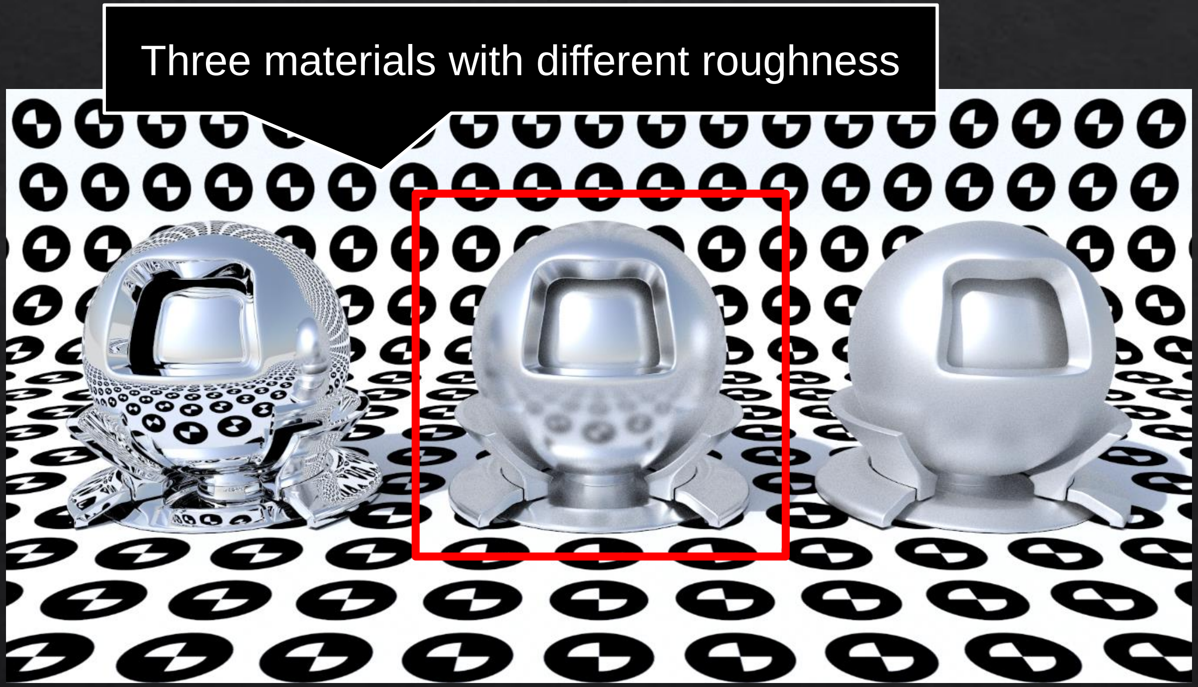
Cone with the spread due to the BSDF



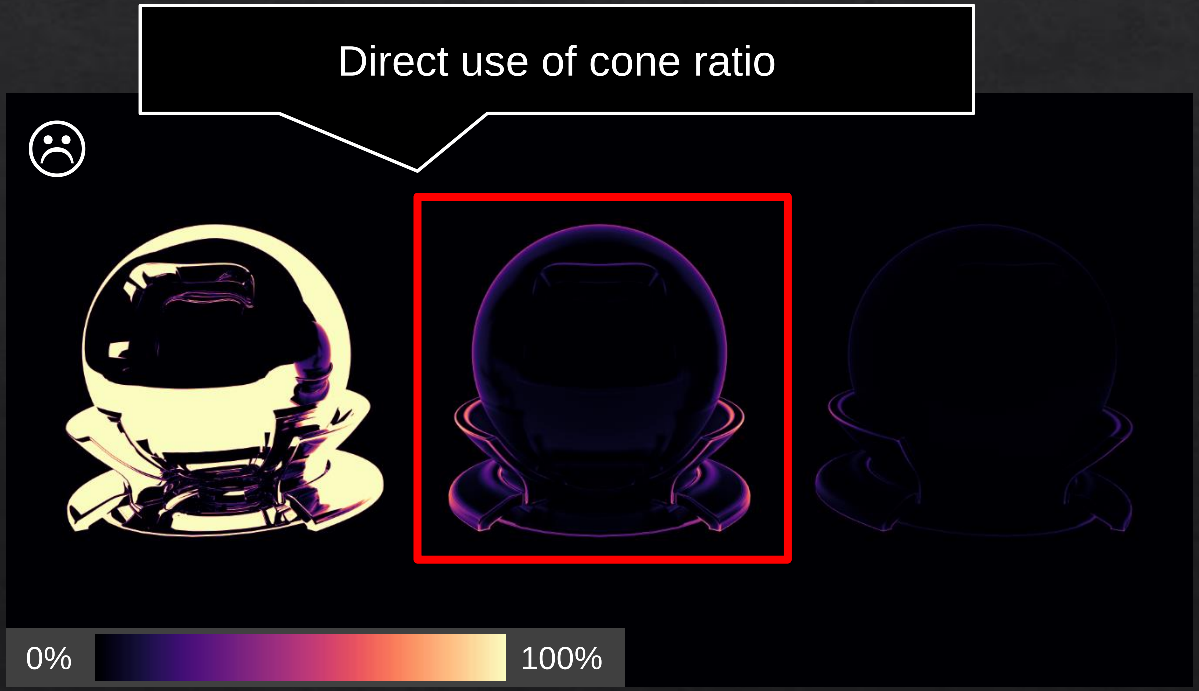
⇨ The metric represents how much pixel blur is occurred due to the BSDF

Problem on the direct use of cone ratio

- Our cone ratio measures the loss of object details due to the BSDF ☺
- May not perceptually plausible for artists ☹



Rendered image

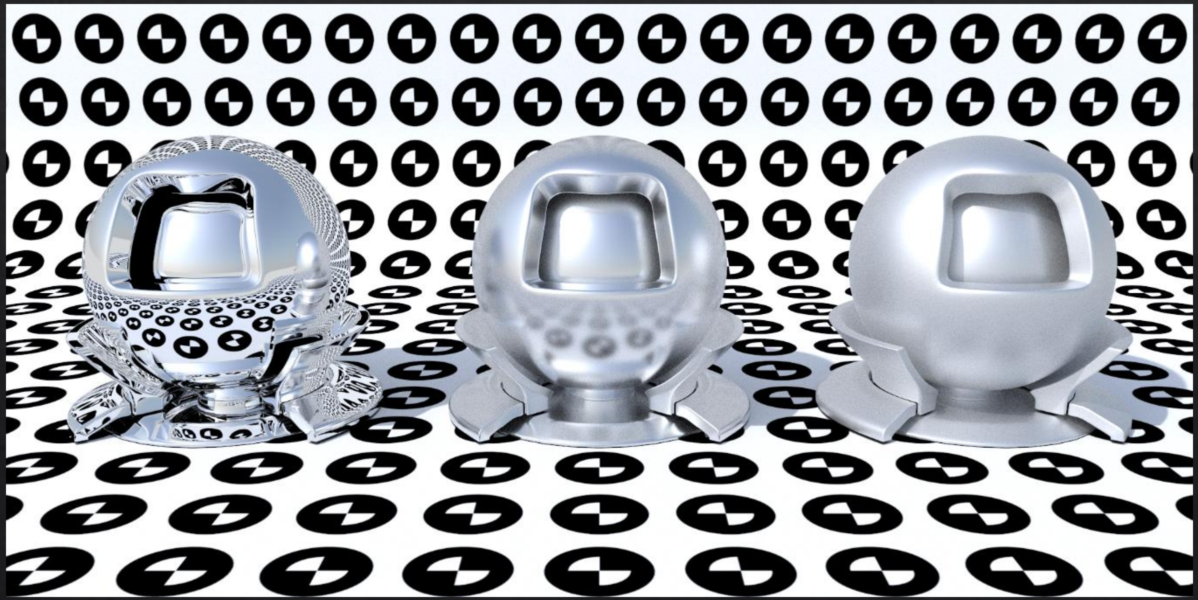


The coverage of the reflected floor

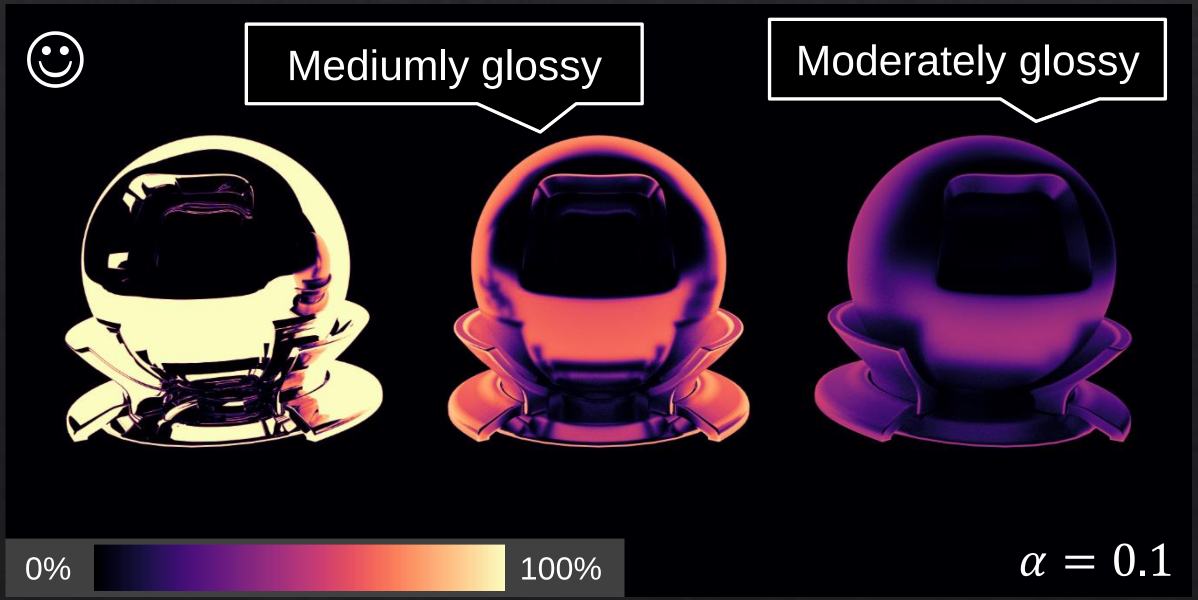
Tweakable mapping

- Simple mapping function with a user-specified parameter

$$W(x, \omega, \omega') = (cone\ ratio)^{\alpha}$$



Rendered image



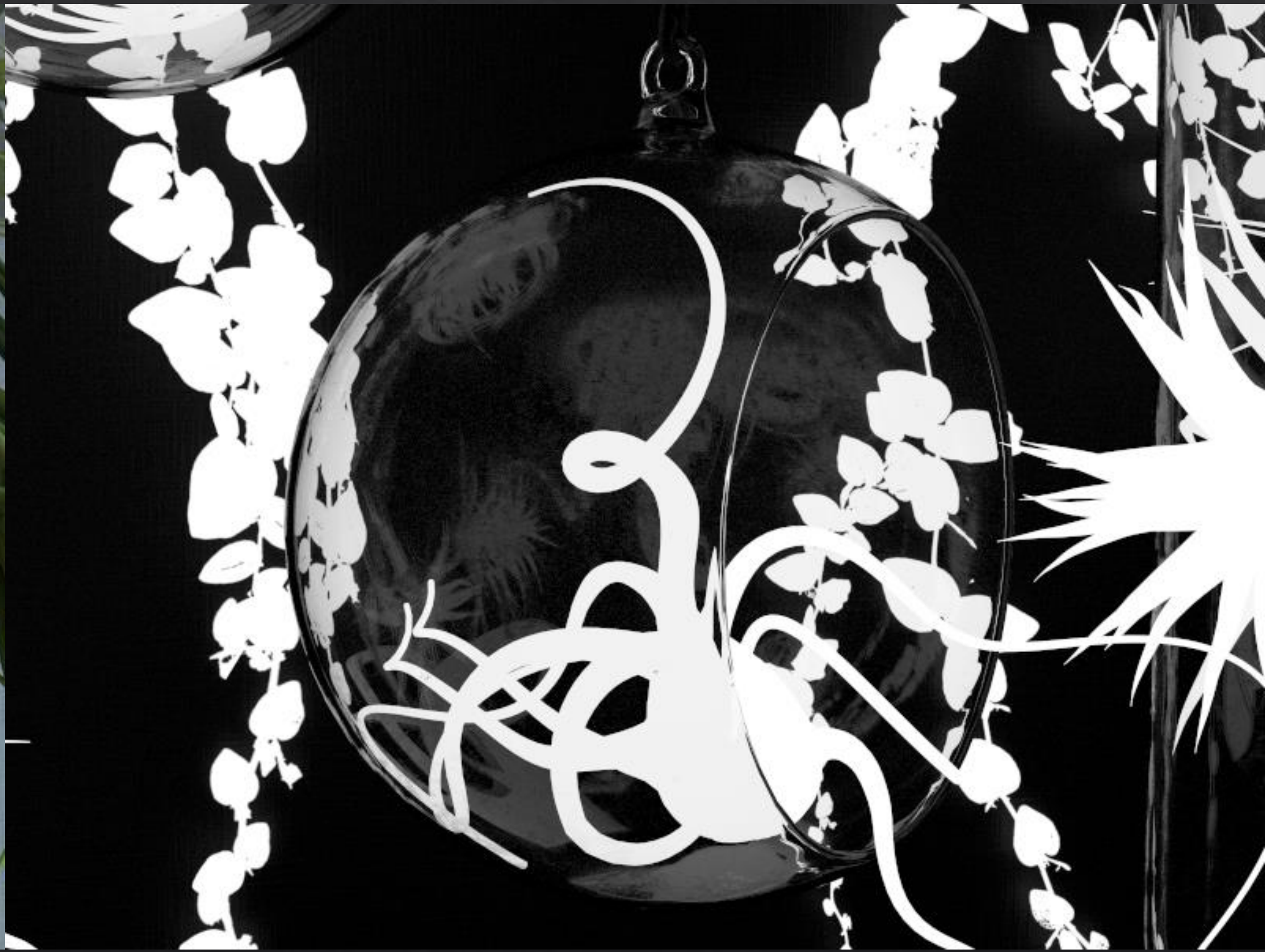
The coverage of the reflected floor

Results

Rendered image

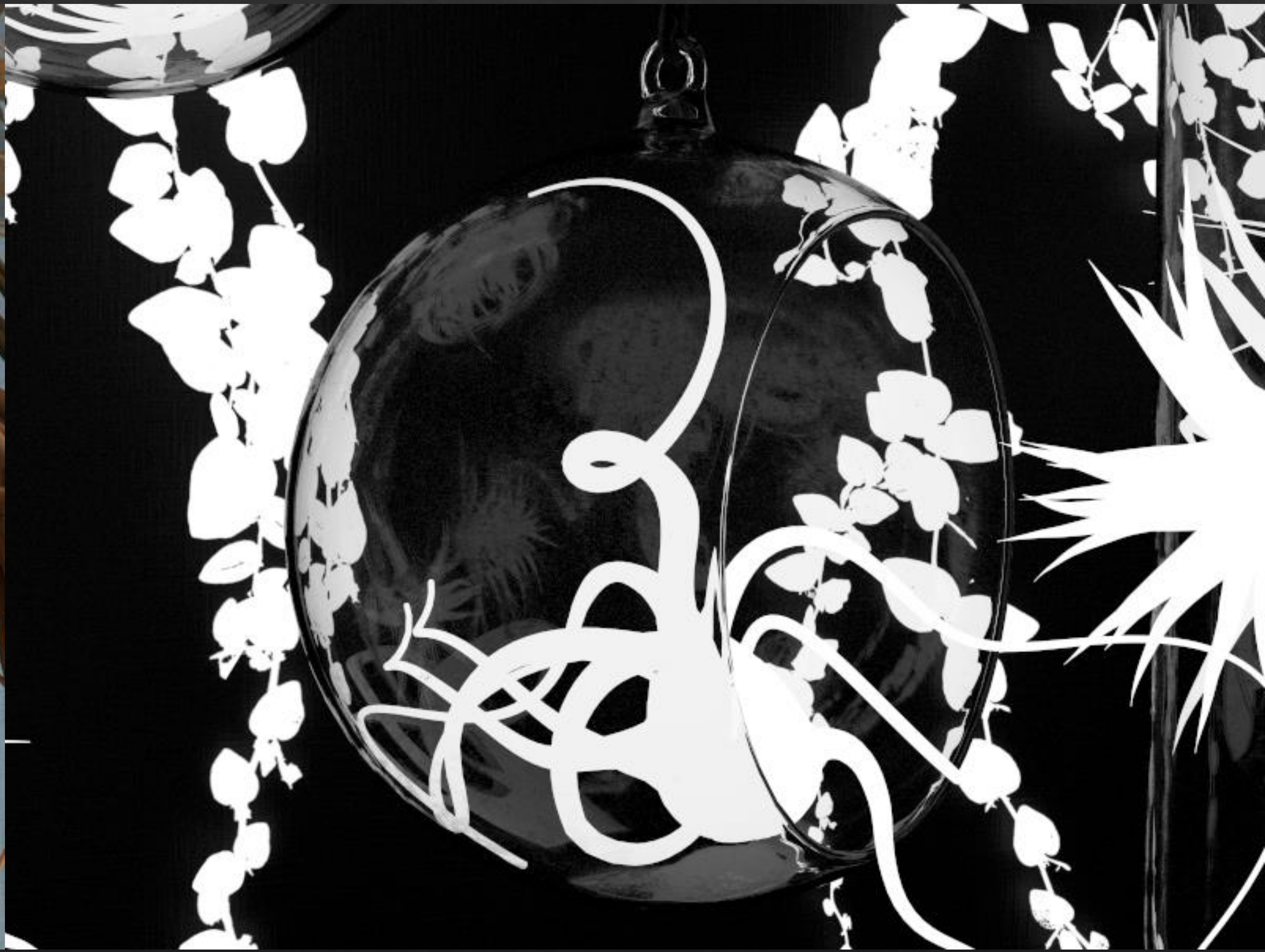
Matte image of plants with previous work [FRIEDMAN 2015]

Rendered image

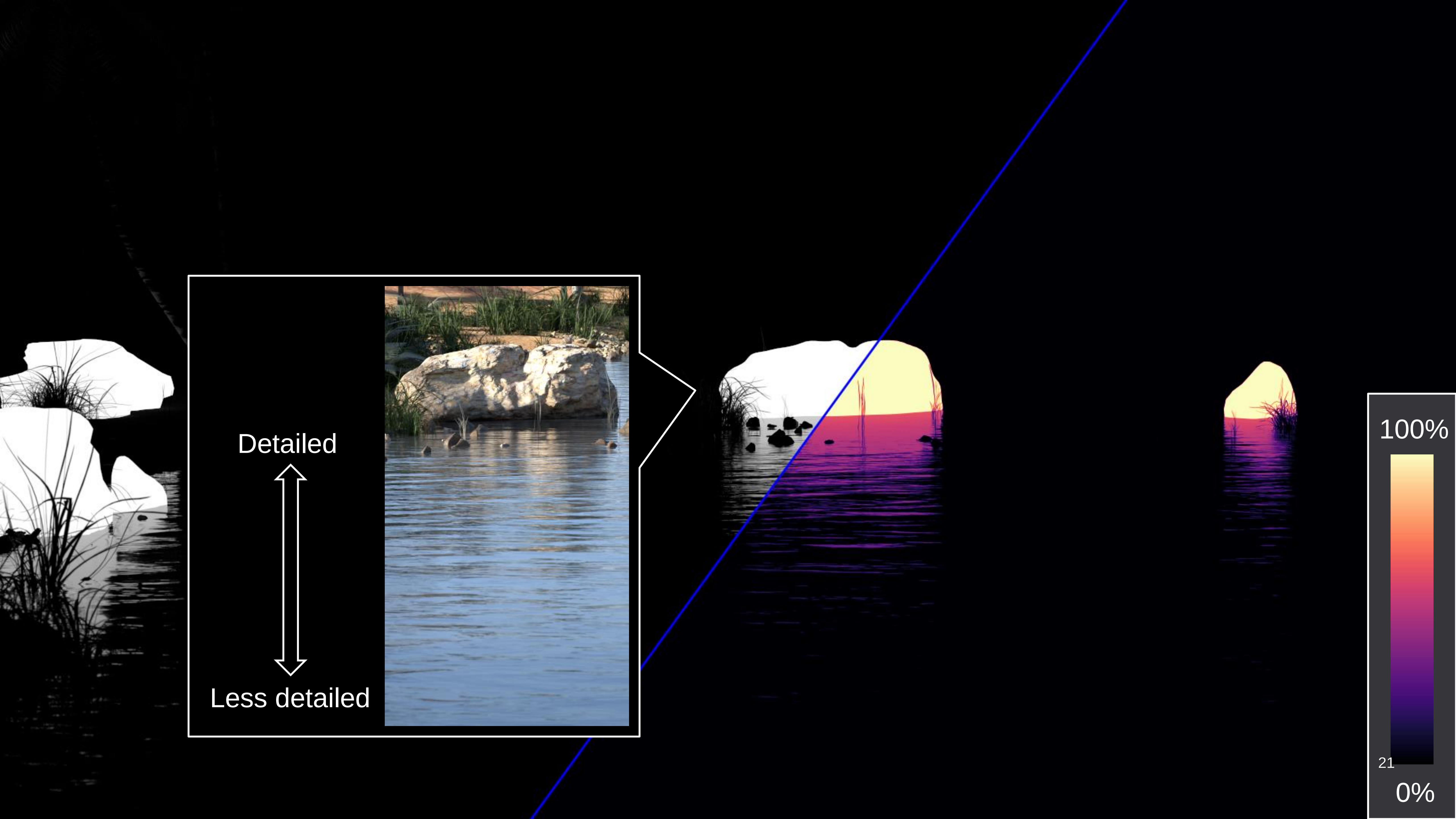


Matte image of plants with ours

Re-colored image



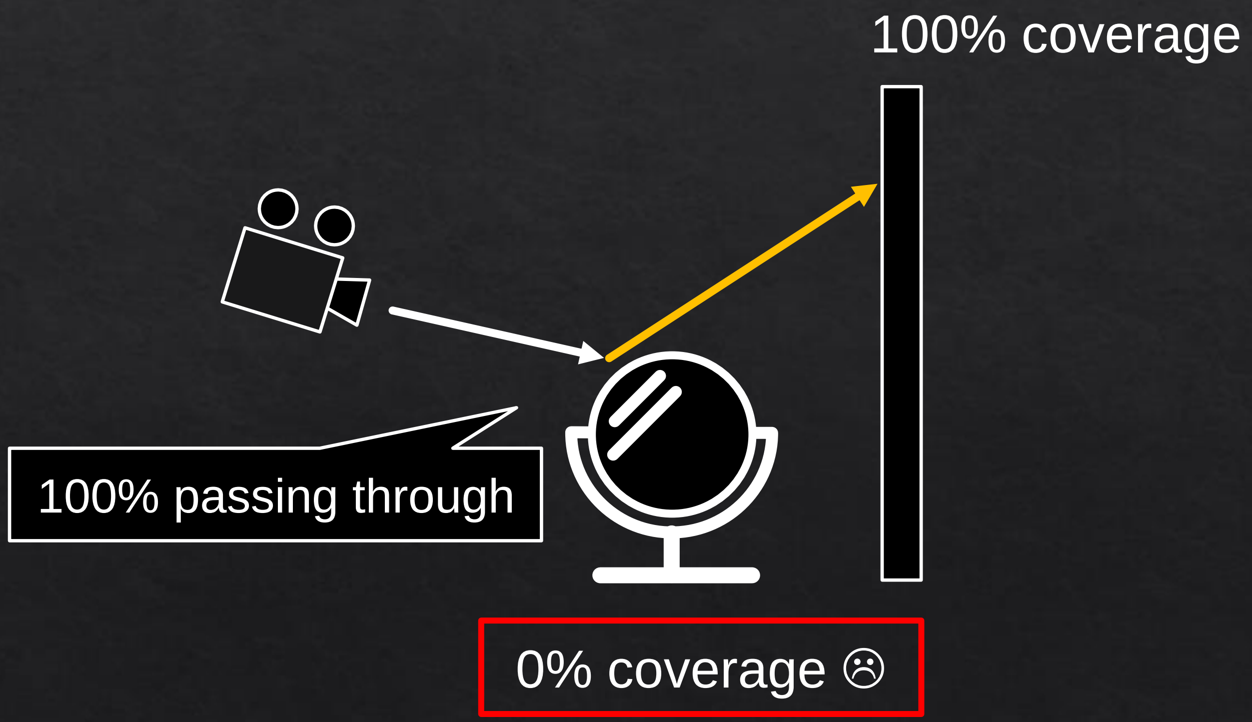
Matte image of plants with ours


Detailed
Less detailed
100%
21
0%


Detailed
Less detailed
100%
0%

Limitations

- Our method produces 0% coverage for perfect specular surface
  - We provide another practical option (please see our paper for more details)



Limitations

- Our method produces 0% coverage for perfect specular surface
  - We provide another practical option (please see our paper for more details)
- Fragment loss due to fixed-size storage
- The granularity of our matte generation is limited to object identifiers
  - Additional data such as depth, opacity, bounce type can be used

Conclusions

- Extend multi-fragment rendering to glossy bounces for matte generation
  - Our weighting function considering the diffusion of rays
- Our implementation of the coverage update on the GPU

References

- [Friedman 2015] Friedman, J. and Jones, A. C. “Fully Automatic ID Mattes with Support for Motion Blur and Transparency”
- [Akenine 2021] Akenine-Möller , T. and Crassin, C. and Boksansky, J. et al. “Improved shader and texture level of detail using ray cones”

Děkuji

AMD

ARR
Advanced Rendering Research Group