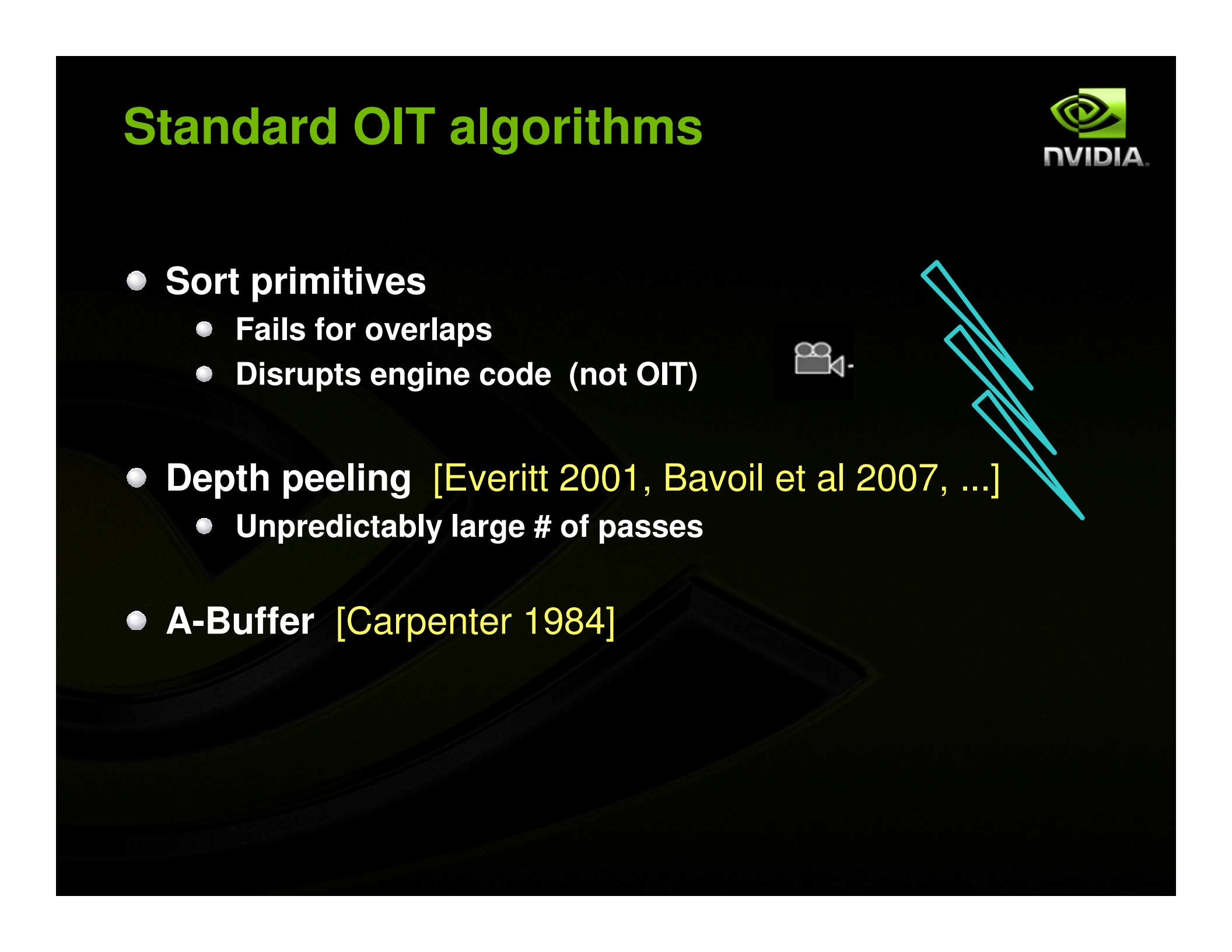

Stochastic
Transparency
Eric Enderton
Erik Sintorn
Pete Shirley
David Luebke
I3D 2010
NVIDIA

Order Independent Transparency

- hair
- foliage
- particle
- windows

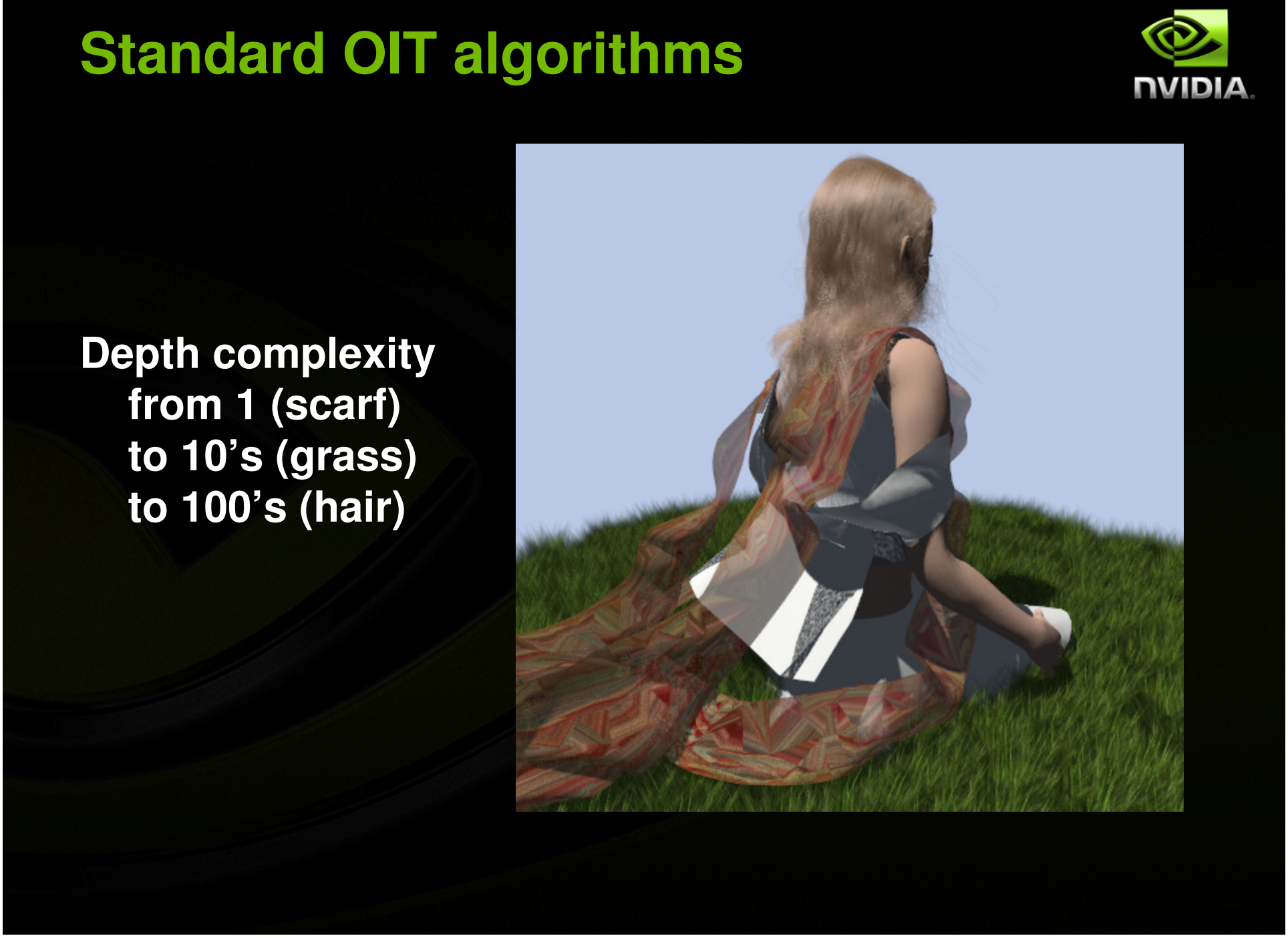
- shadows thereof

Standard OIT algorithms



- **Sort primitives**
  - **Fails for overlaps**
  - **Disrupts engine code (not OIT)**
- **Depth peeling** [Everitt 2001, Bavoil et al 2007, ...]
  - **Unpredictably large # of passes**
- **A-Buffer** [Carpenter 1984]

Standard OIT algorithms



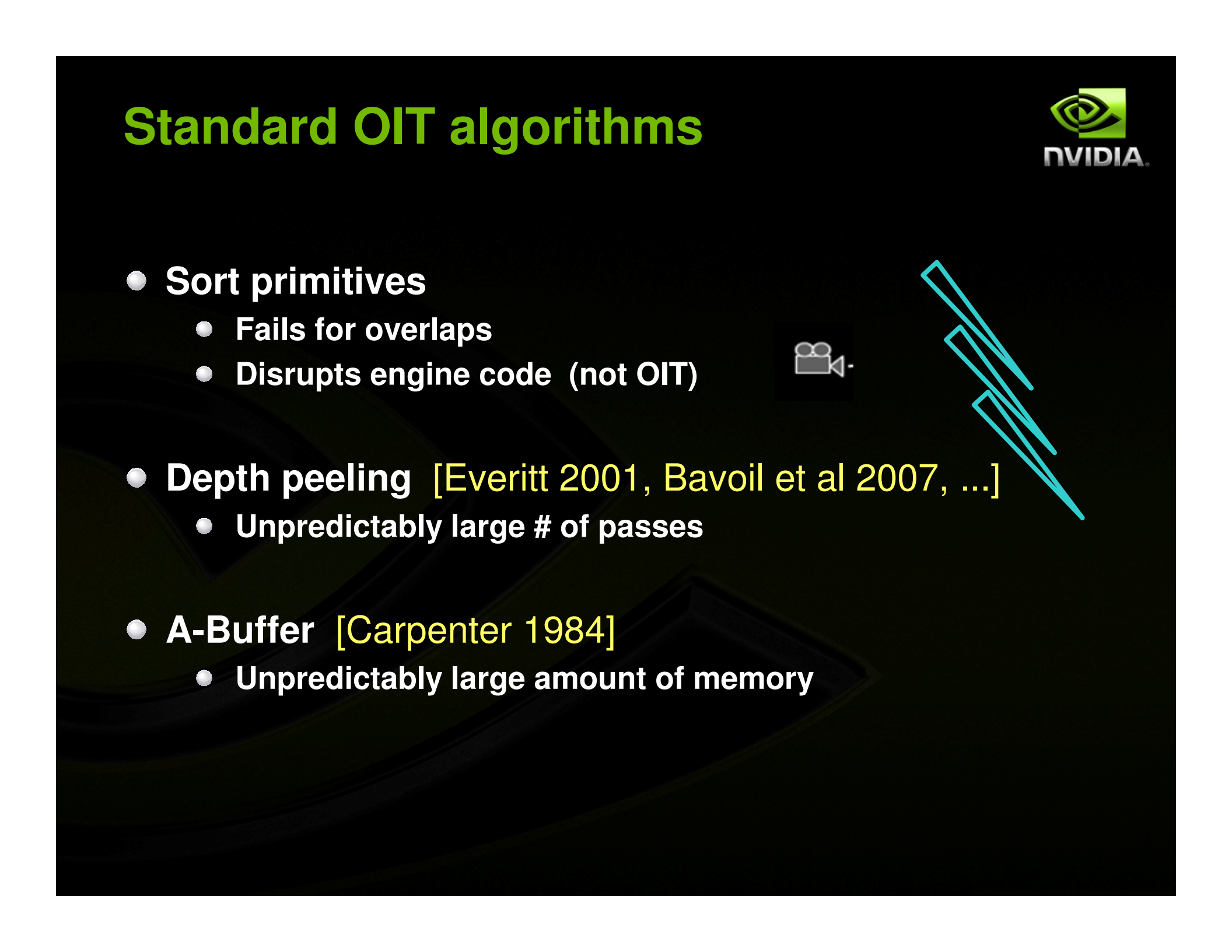
Depth complexity
from 1 (scarf)
to 10's (grass)
to 100's (hair)

Standard OIT algorithms

Depth peeling:

in the same time as our algorithm, 5 passes

Standard OIT algorithms



- **Sort primitives**
  - **Fails for overlaps**
  - **Disrupts engine code (not OIT)**
- **Depth peeling** [Everitt 2001, Bavoil et al 2007, ...]
  - **Unpredictably large # of passes**
- **A-Buffer** [Carpenter 1984]
  - **Unpredictably large amount of memory**

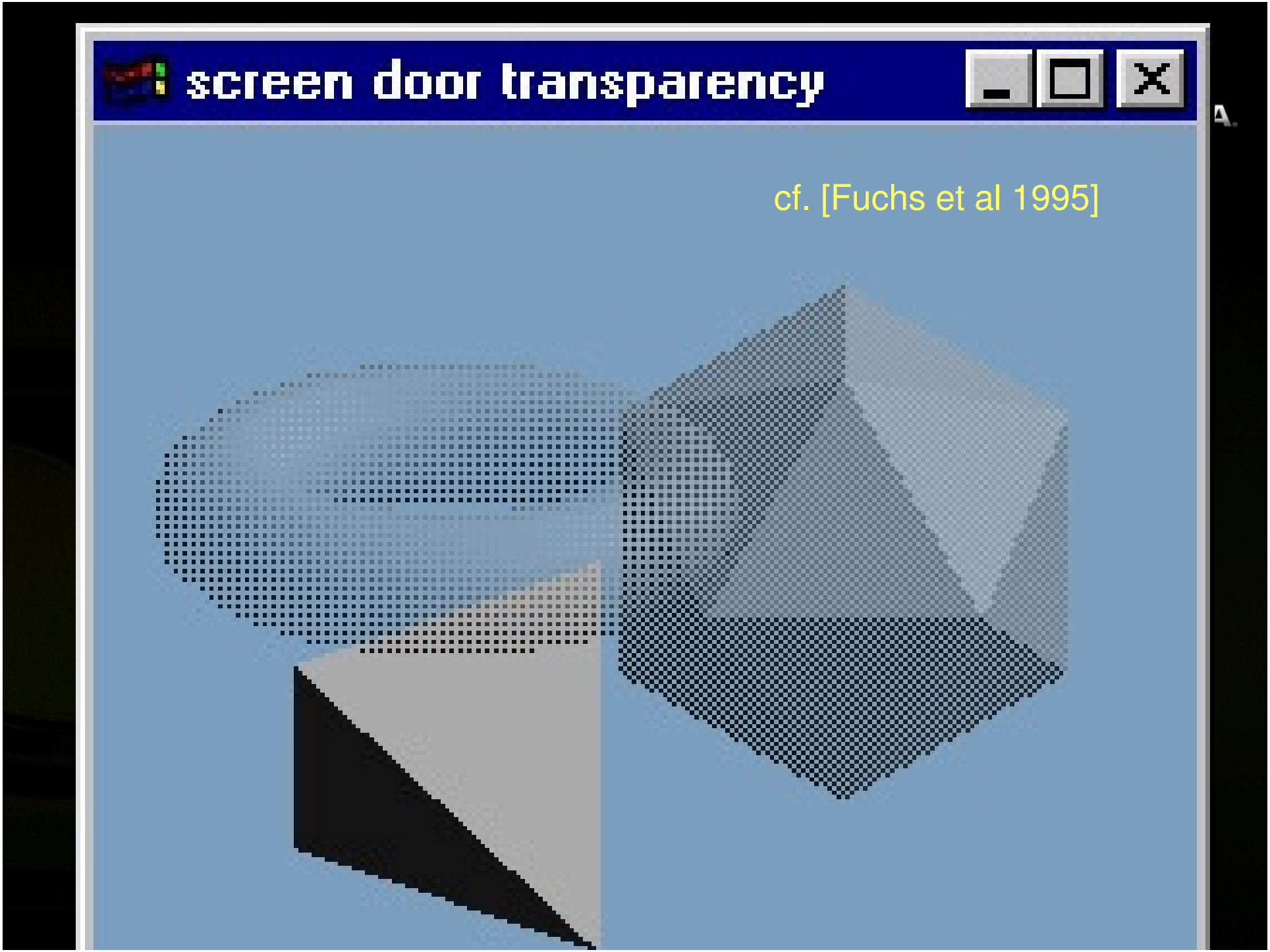
Transparency Without Sorting

- For each pixel sample, collect **statistics** about the transparent fragments along that ray
  - min z, max z, count, total opacity, stranger things
  - Estimating the parameters of a model
- **Fast:** Fixed passes, fixed memory
- **Approximate**

Transparency Without Sorting

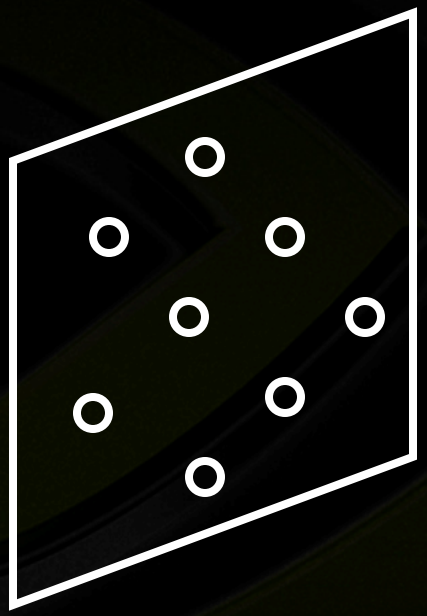
- **Variance Shadow Maps** [Donnelly + Lauritzen I3D 2005]
  - **collect mean, variance of z**
- **Occupancy Maps** [Sintorn + Assarsson I3D 2009]
  - **collect counts, occupancy bit mask**
  - **assumes equal alphas; trouble with multiple clumps**
- **Fourier Opacity Maps** [Jansen + Bavoil I3D 2010 – next!]

Stochastic Transparency: Basic Method


screen door transparency
cf. [Fuchs et al 1995]

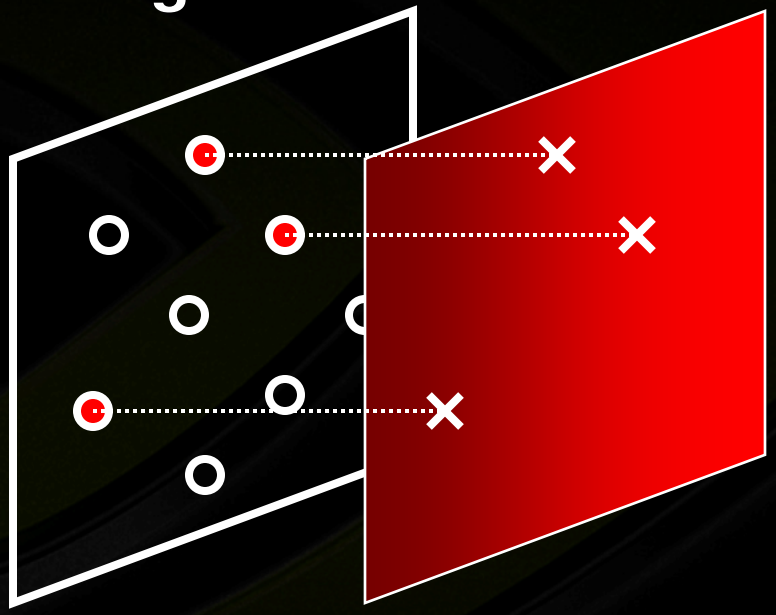
Alpha-to-Coverage [Akeley 1993]

- MSAA with S samples per pixel (S=8)



Alpha-to-Coverage [Akeley 1993]

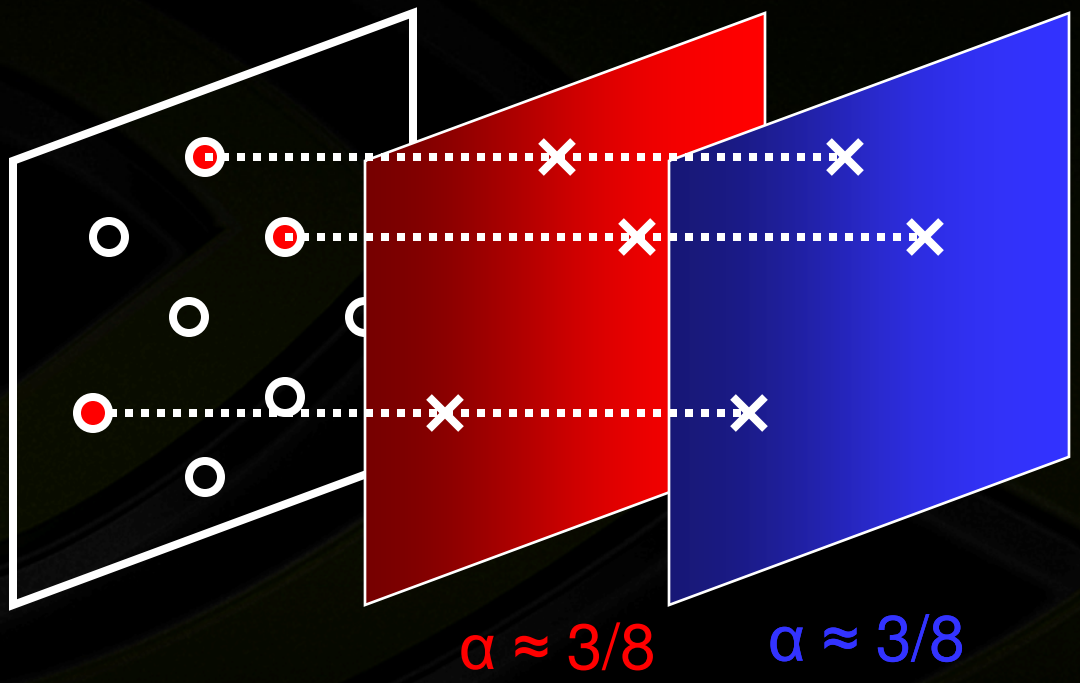
- Kill all but α*S samples
- “coverage mask”



α ≈ 3/8

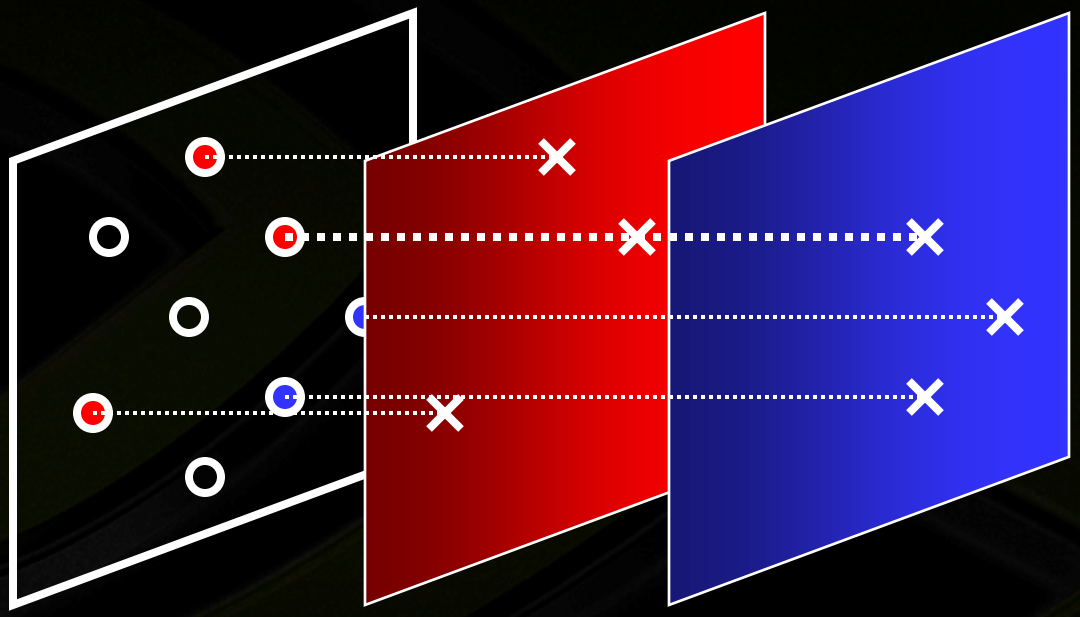
Alpha-to-Coverage

- Two fragments with similar alpha cover the same samples -- oops



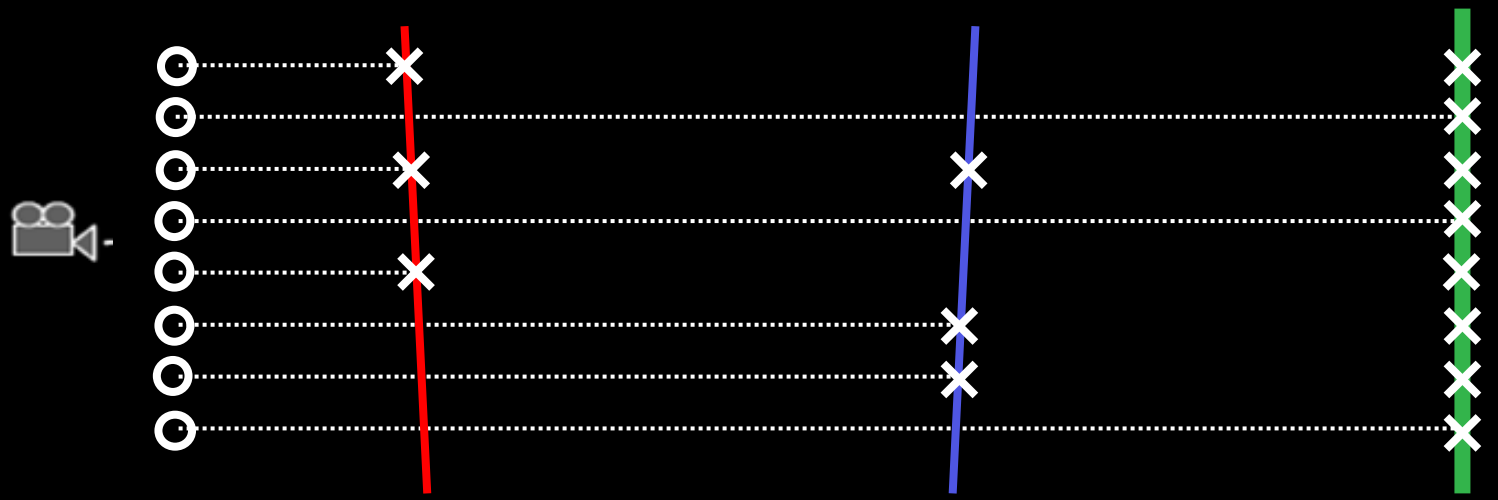
Idea

- Choose sample masks randomly [OpenGL 1993]



- Correct on average, in all cases

Correct on average



$$c = \alpha_1 c_1 + (1-\alpha_1)(\alpha_2 c_2 + (1-\alpha_2)\alpha_3 c_3)$$

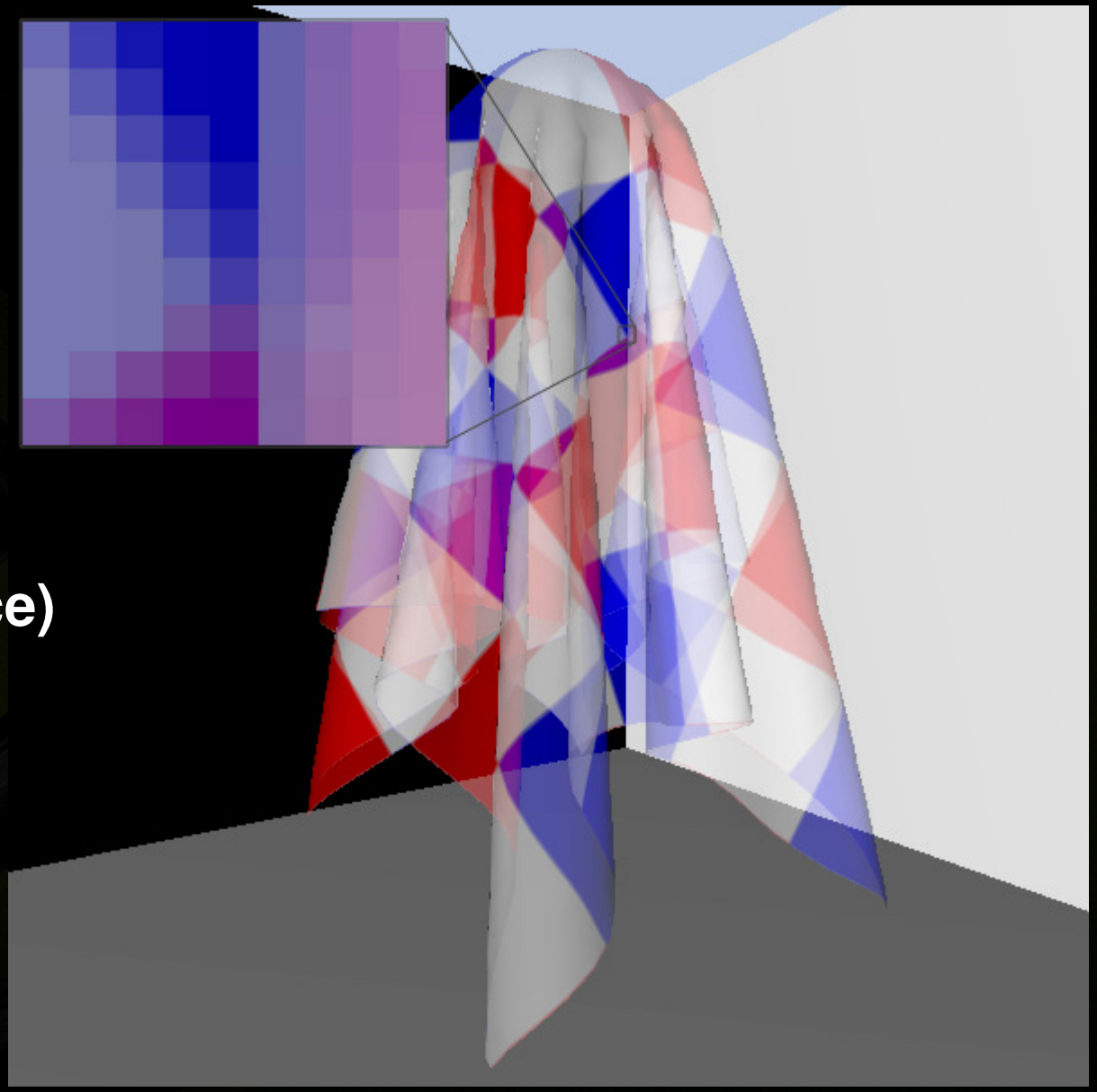
$$= \text{“over”}$$

Stochastic Transparency

Screen-door + multi-sampling + random masks.

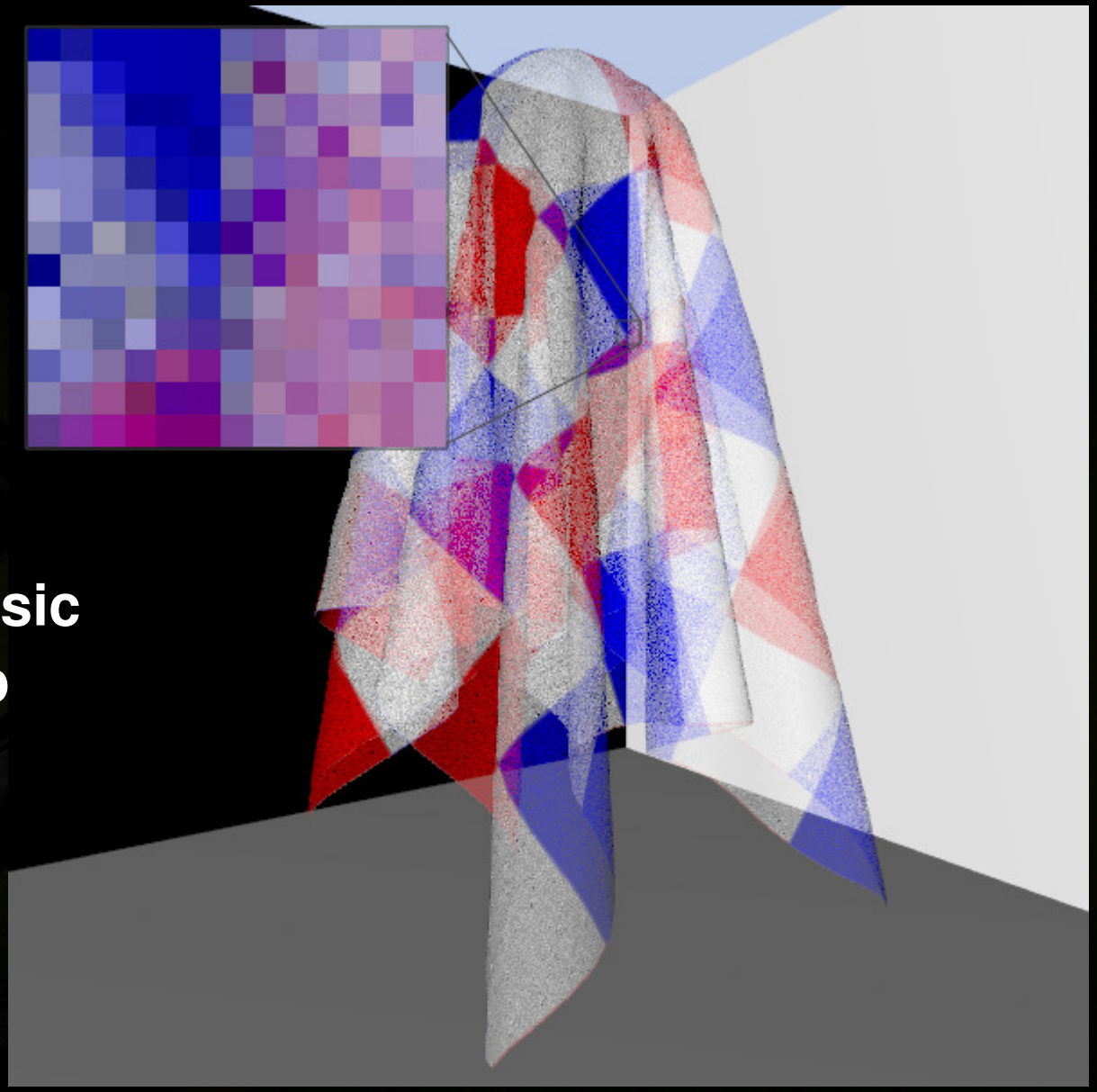
- Correct on average, in all cases
  - Foliage, Smoke, Hair, Glass
  - Mixed together
- Fast
  - One order-independent pass
  - One MSAA z-buffer
- But noisy
  - → More samples
  - → More algorithms

Stochastic Transparency



(Reference)

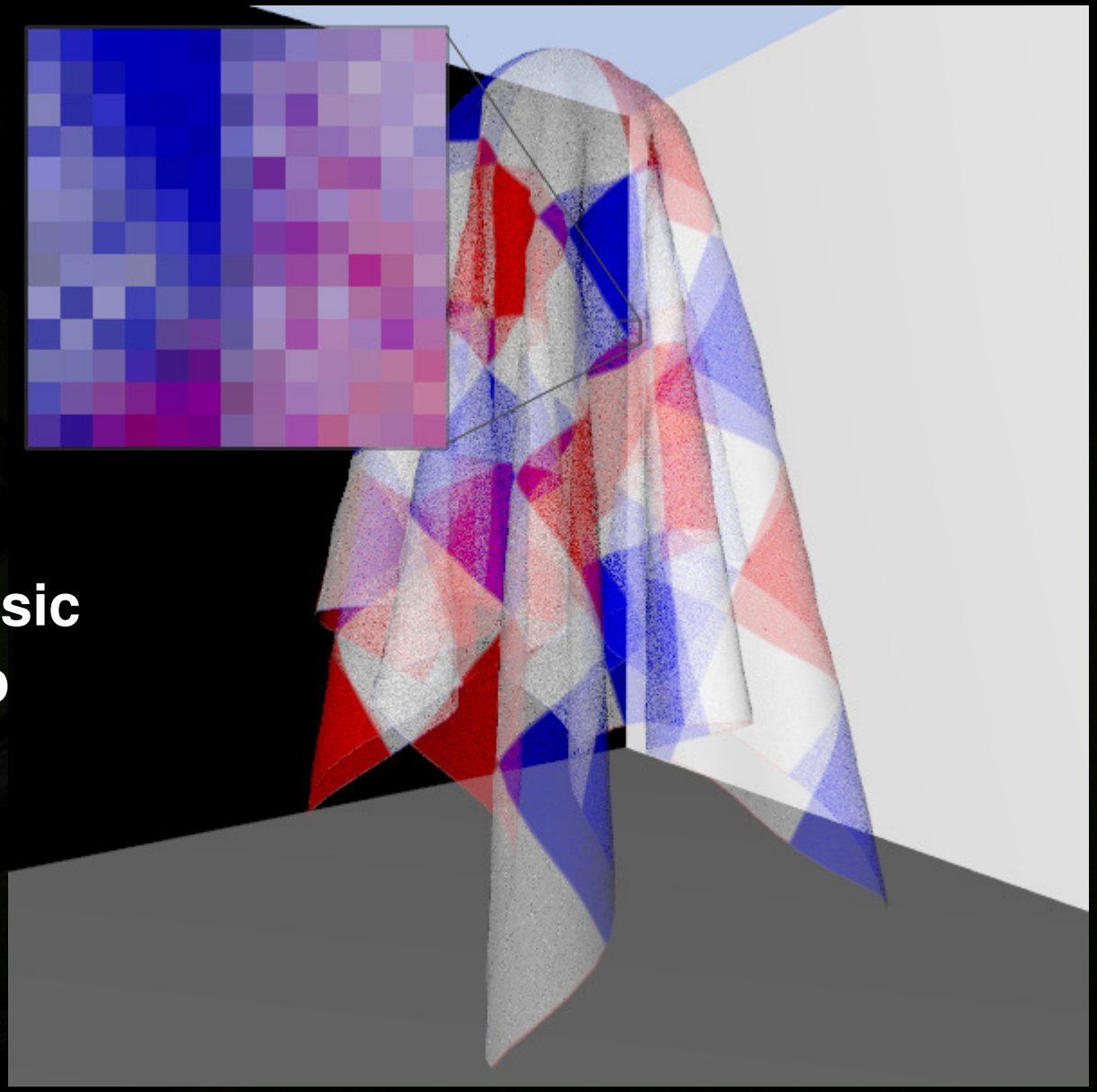
Stochastic Transparency



Alg 1. Basic

8 spp

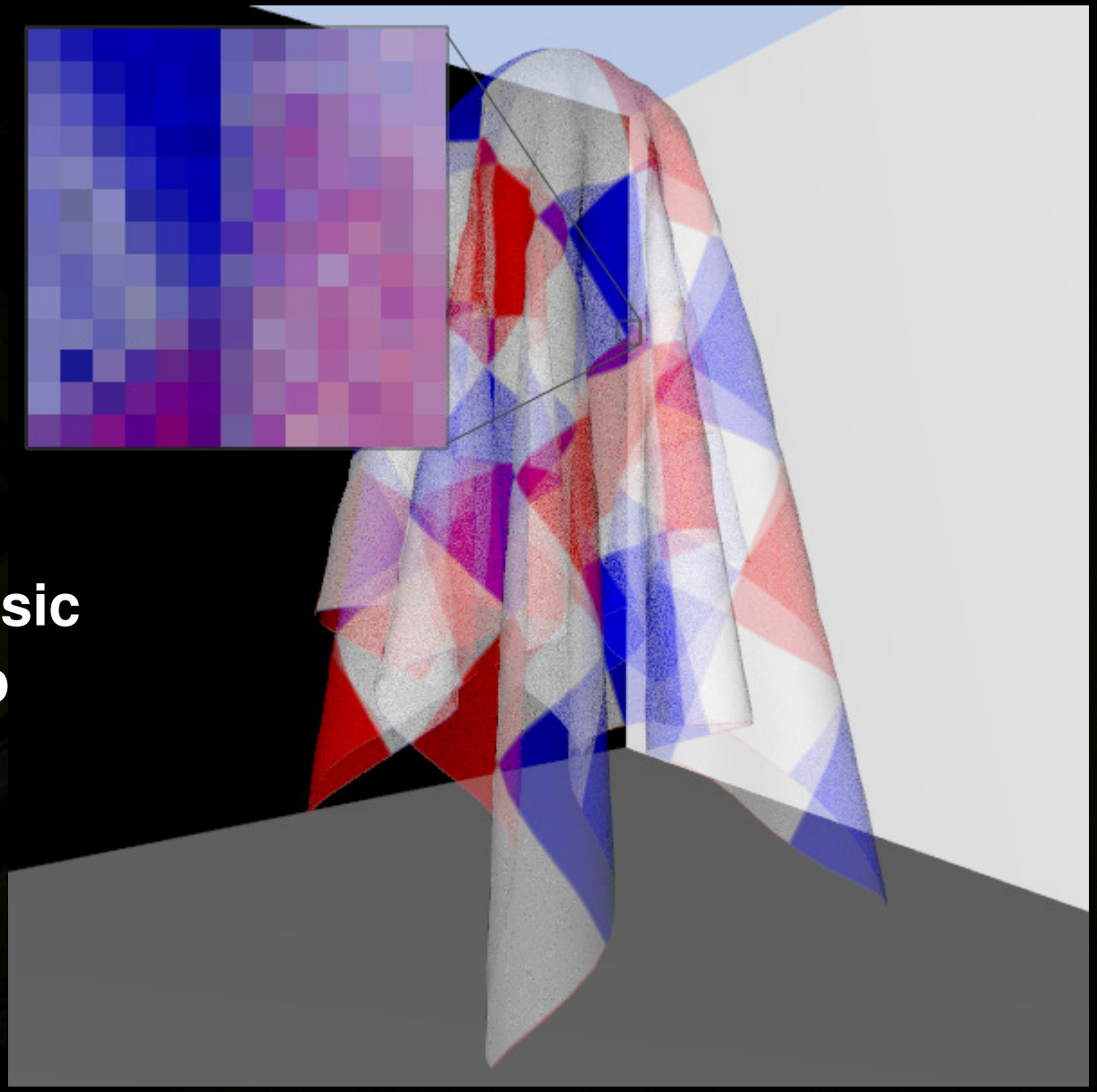
Stochastic Transparency



Alg 1. Basic

16 spp

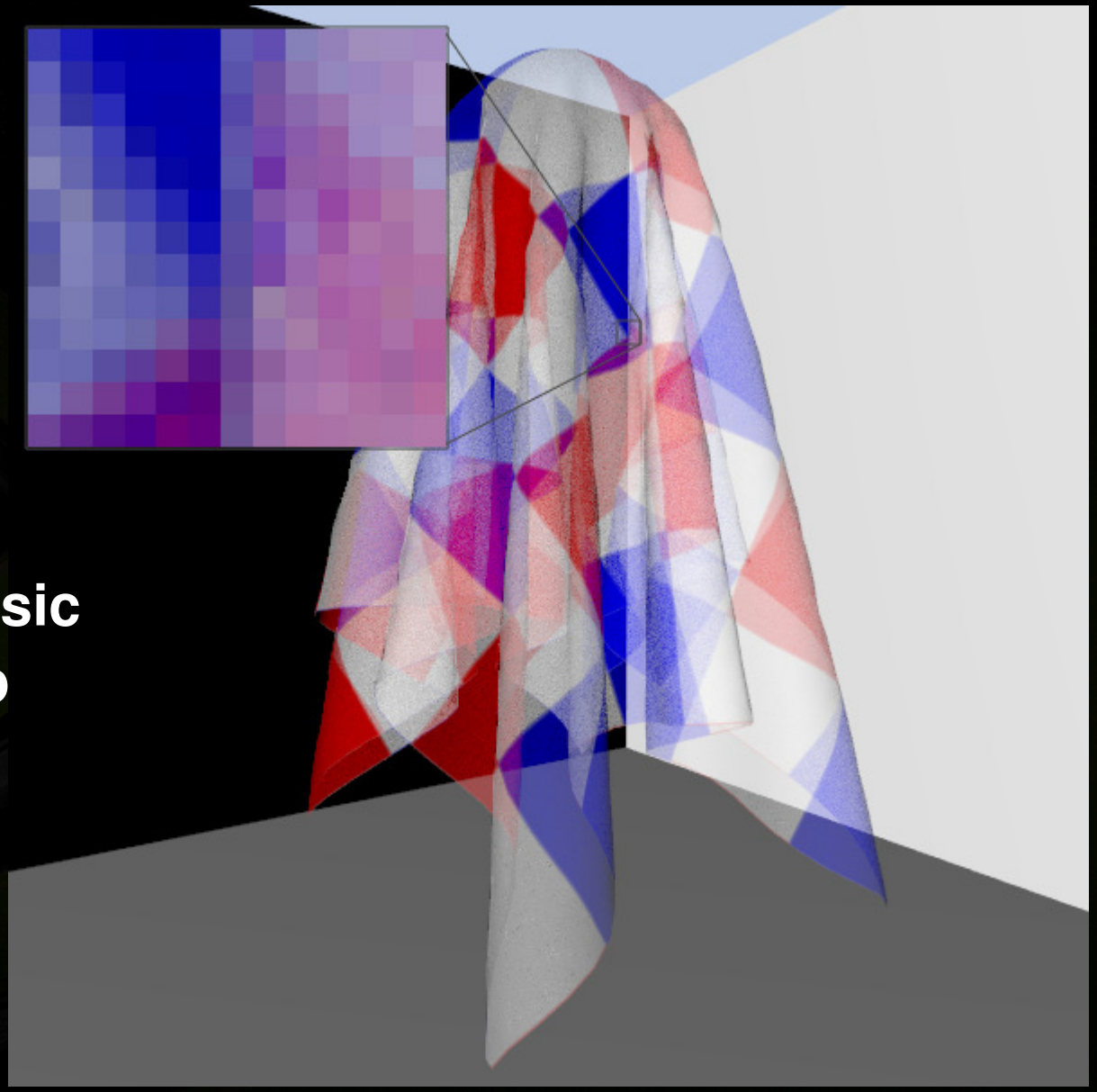
Stochastic Transparency



Alg 1. Basic

32 spp

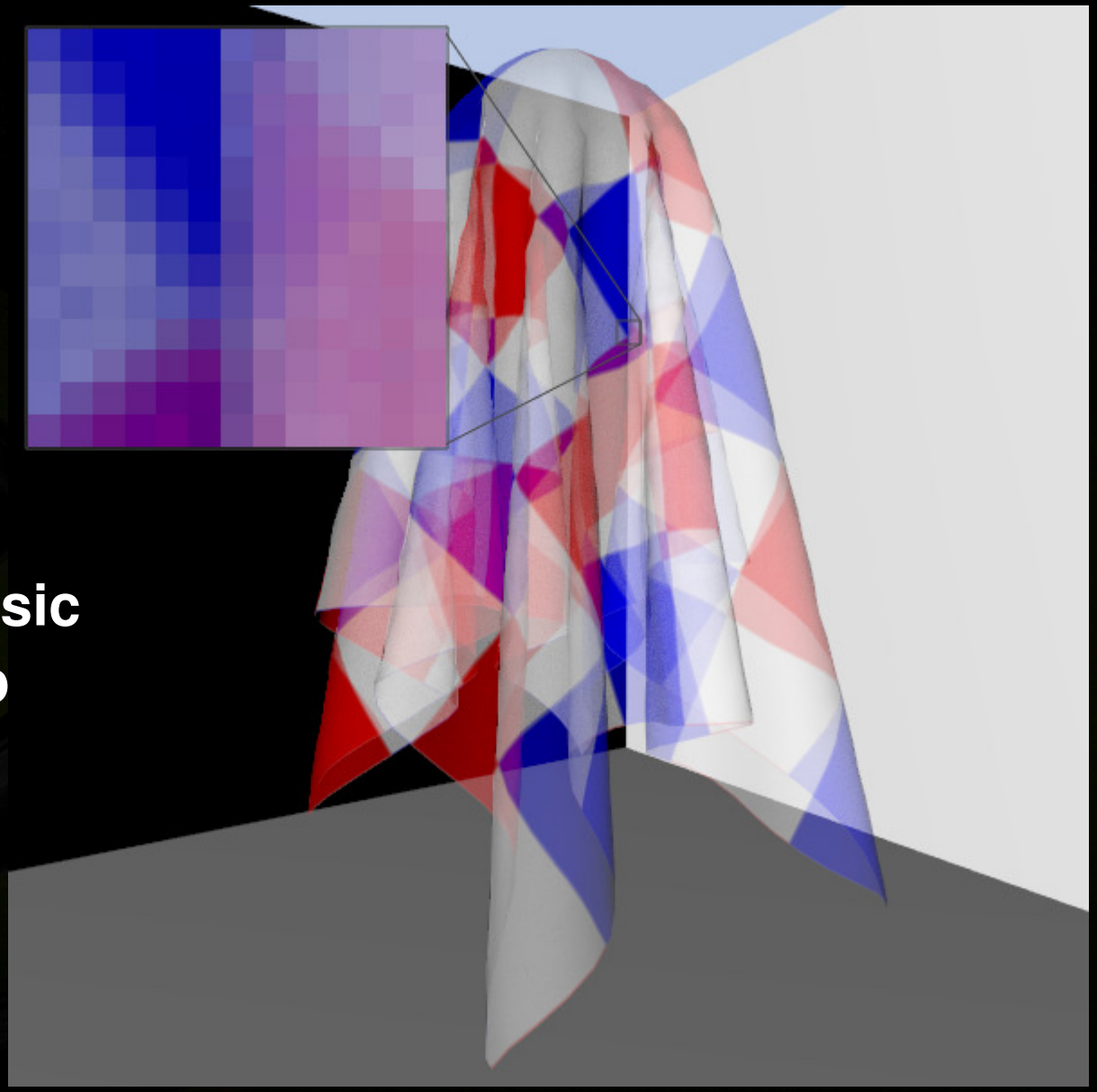
Stochastic Transparency



Alg 1. Basic

64 spp

Stochastic Transparency



Alg 1. Basic

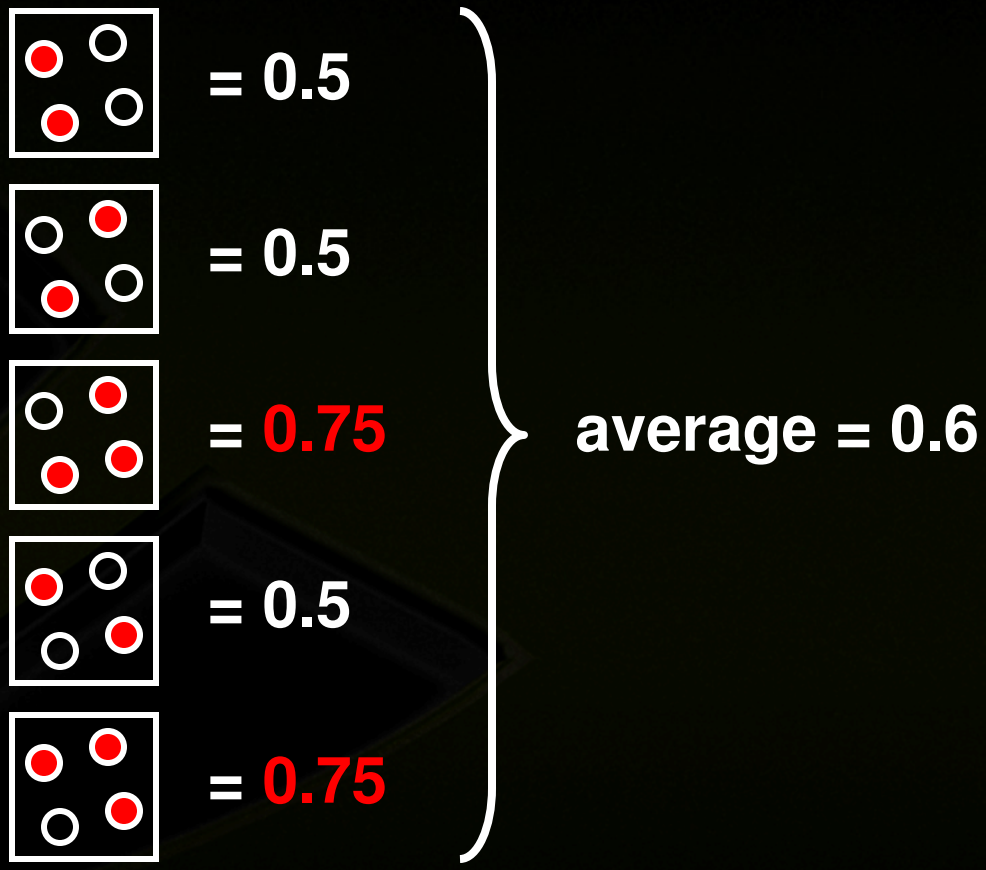
512 spp


NVIDIA
Motion
(video #1)

Quantization noise

Example:

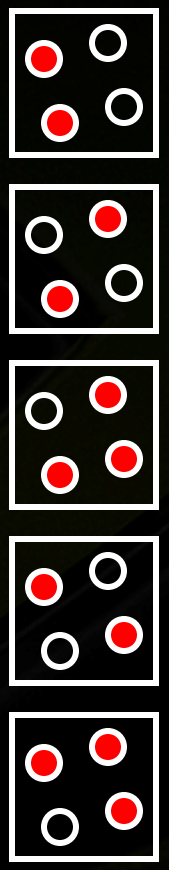
- 4x MSAA
- α = 0.6



Alpha correction

- One extra pass to render correct total α
→ Correction factor

- One layer → exact
- More layers → still noisy



0.5 * 0.6/0.5 = 0.6

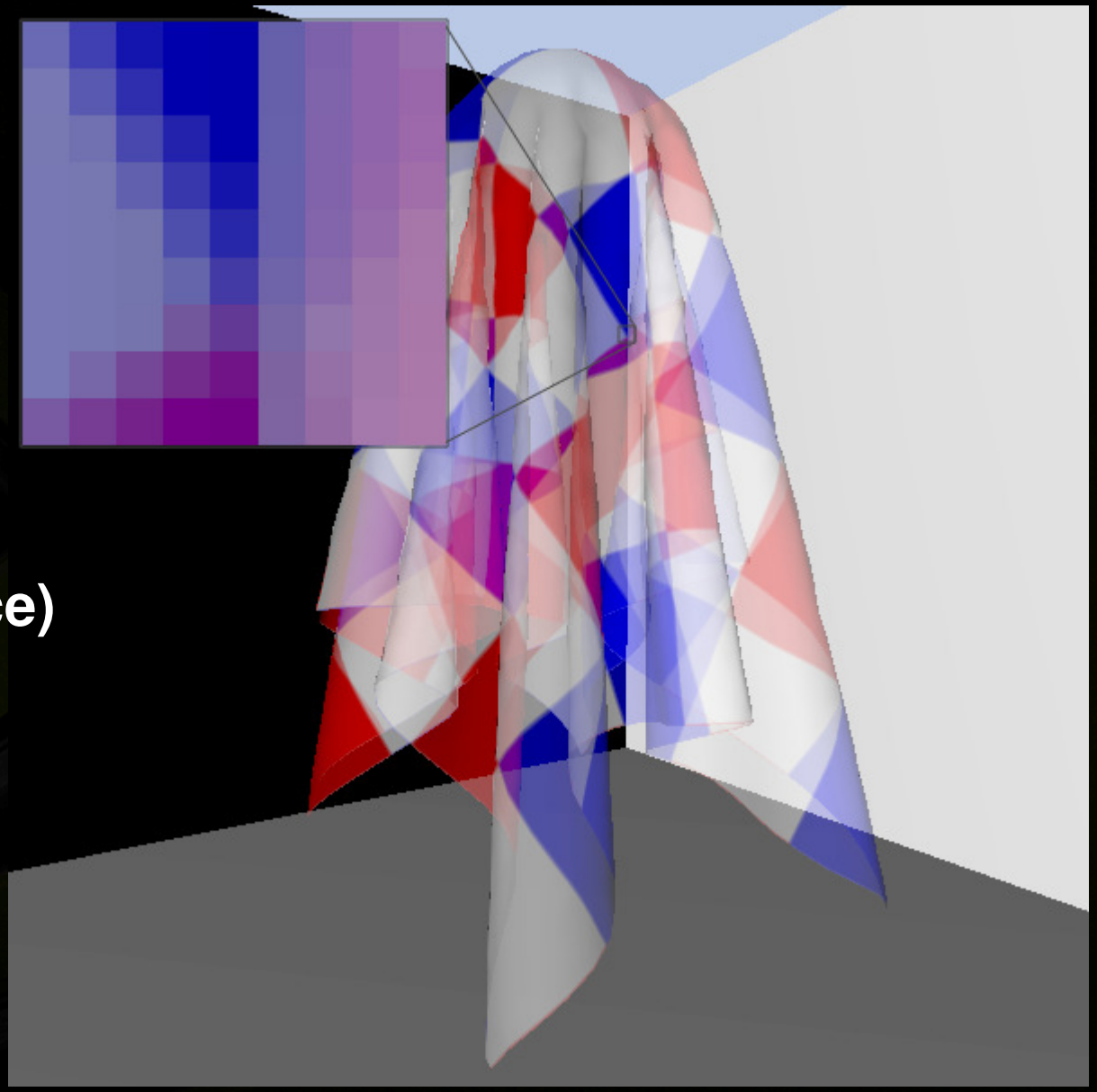
0.5 * 0.6/0.5 = 0.6

0.75 * 0.6/0.75 = 0.6

0.5 * 0.6/0.5 = 0.6

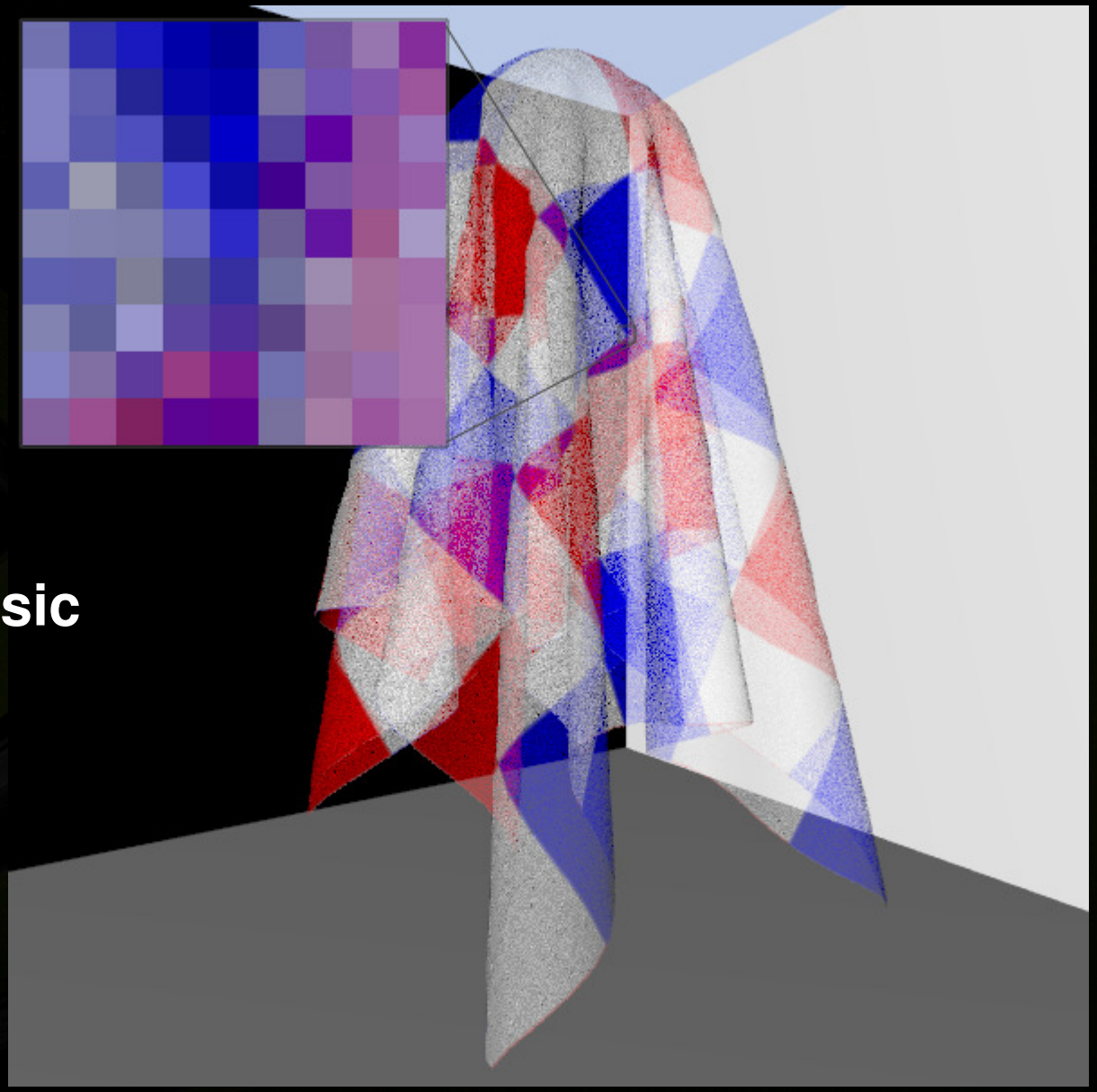
0.75 * 0.6/0.75 = 0.6

Stochastic Transparency



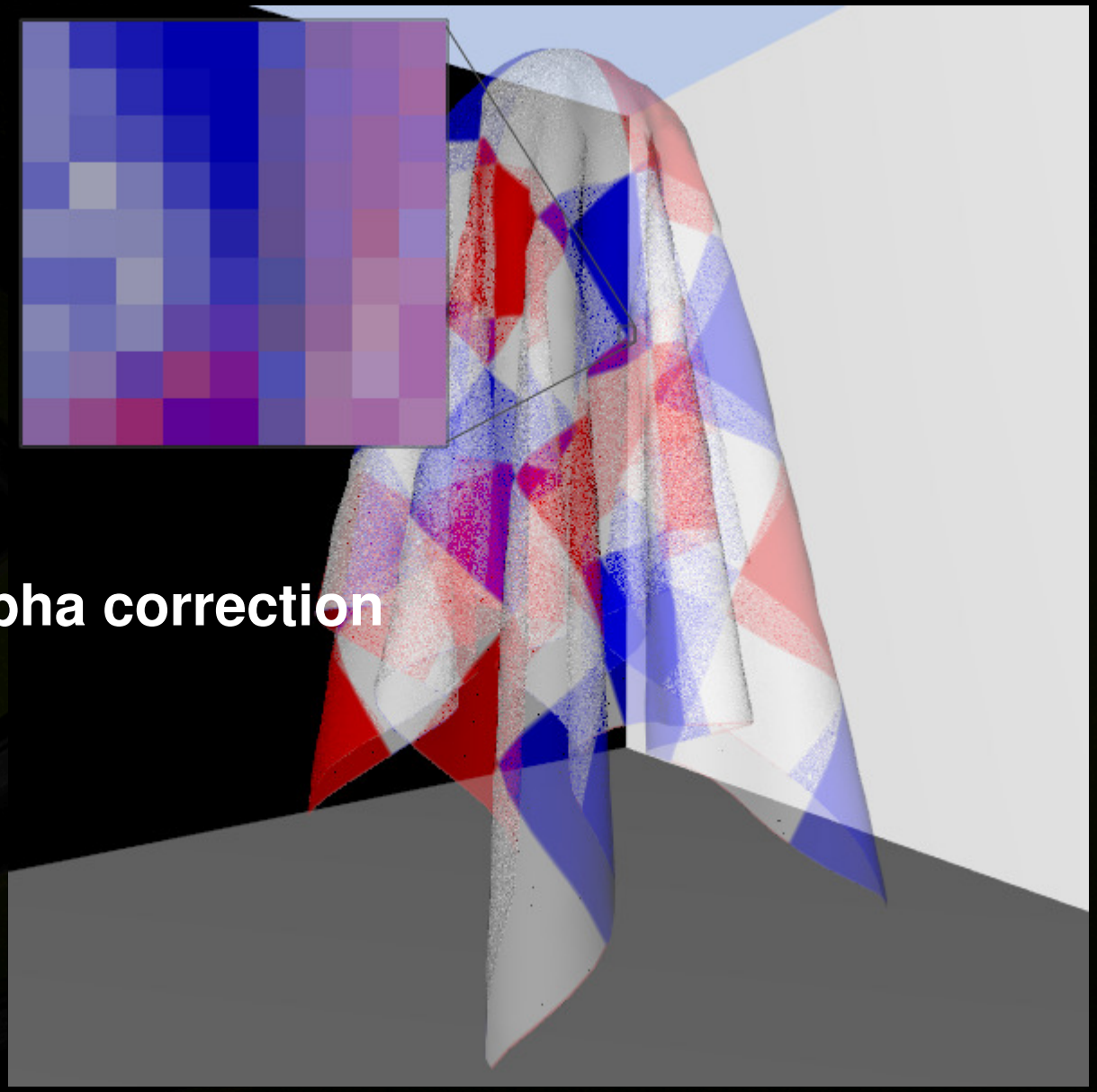
(Reference)

Stochastic Transparency



Alg 1. Basic
8 spp

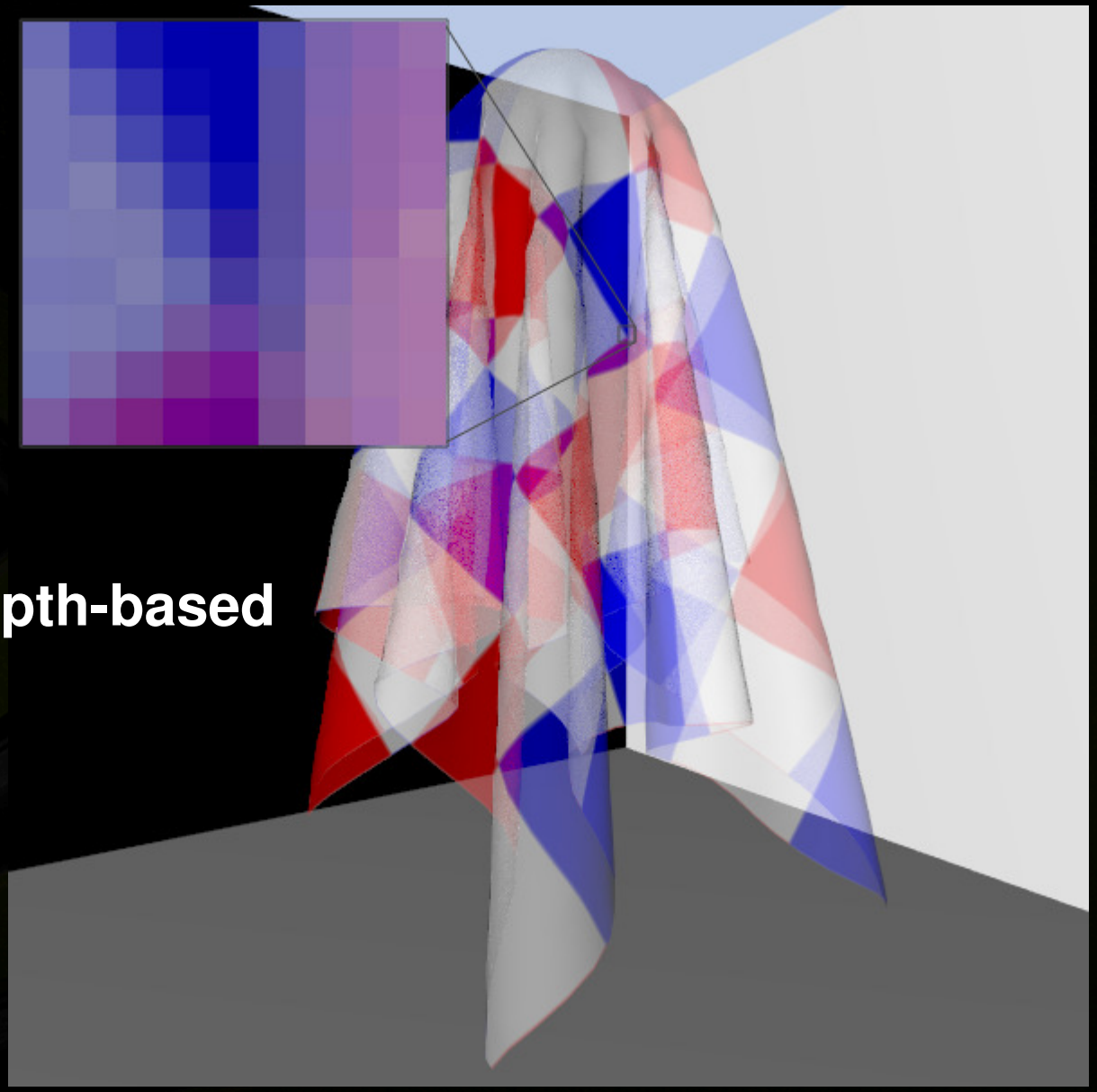
Stochastic Transparency



Alg 2. Alpha correction

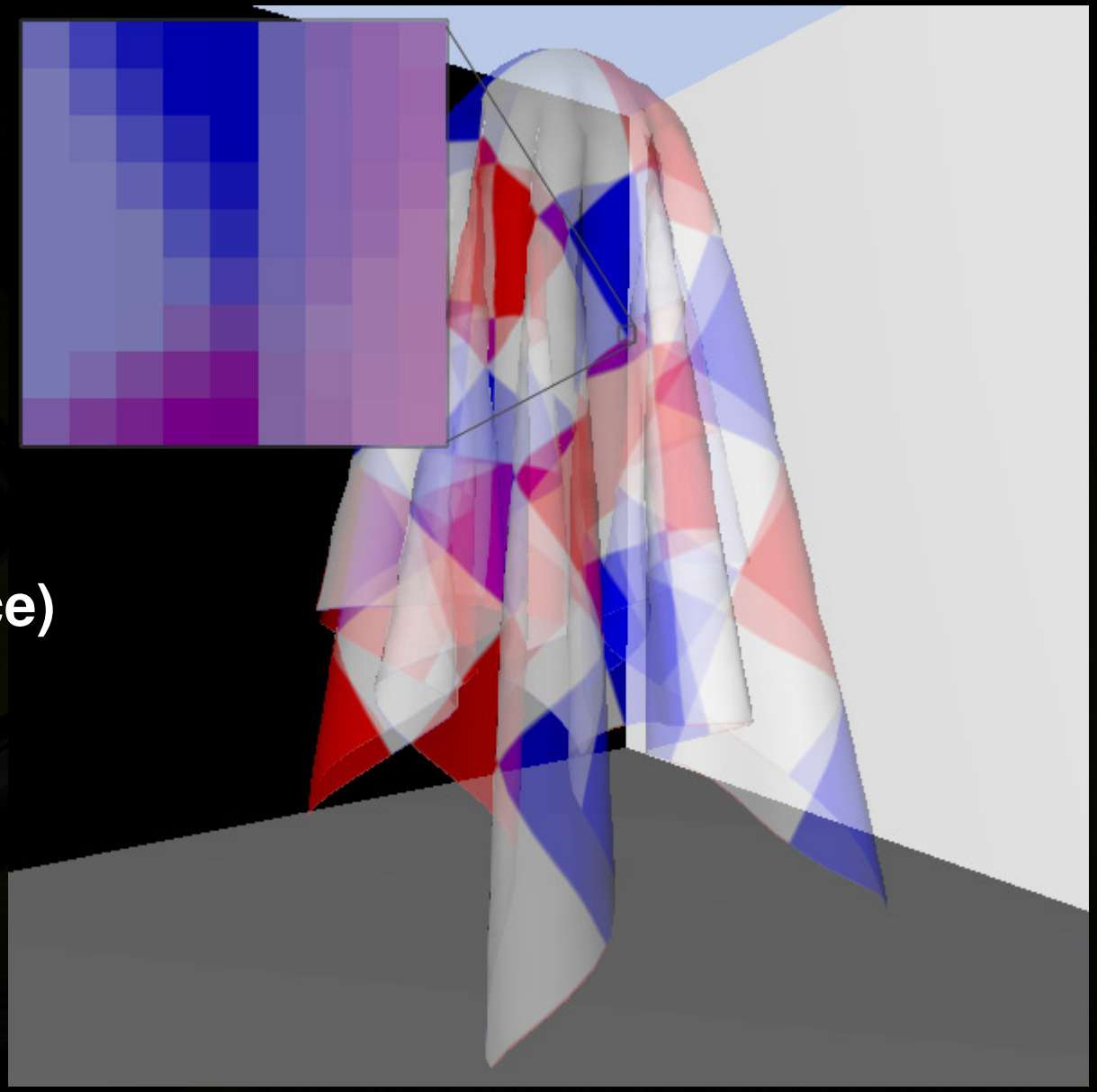
8 spp

Stochastic Transparency



Alg 3. Depth-based
8 spp

Stochastic Transparency



(Reference)

Stochastic Shadows

Stochastic Shadow Map

- A shadow map, with screen-door transparency
- Noise → higher-res map
- Look-up is just PCF

Stochastic Shadow Map

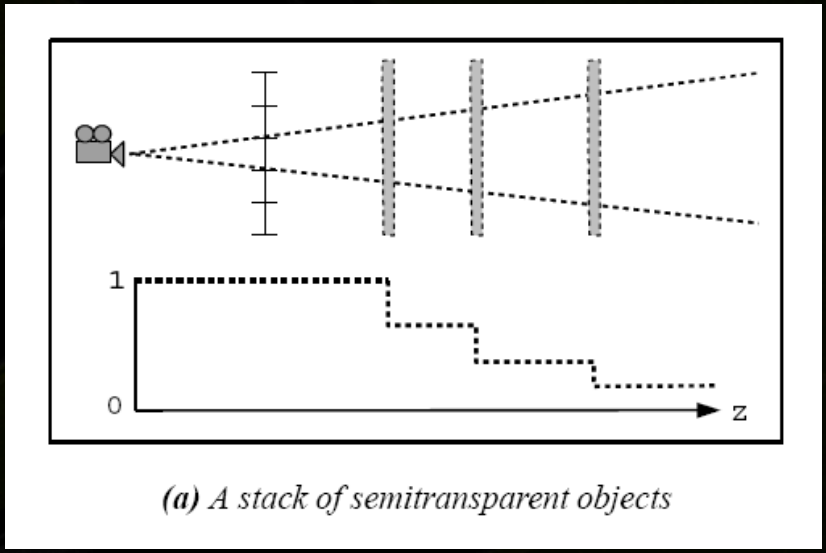
Stochastic Shadow Map

- Optional: Render with MSAA hardware

→ Each map pixel contains S depth values

- Models vis(z) =

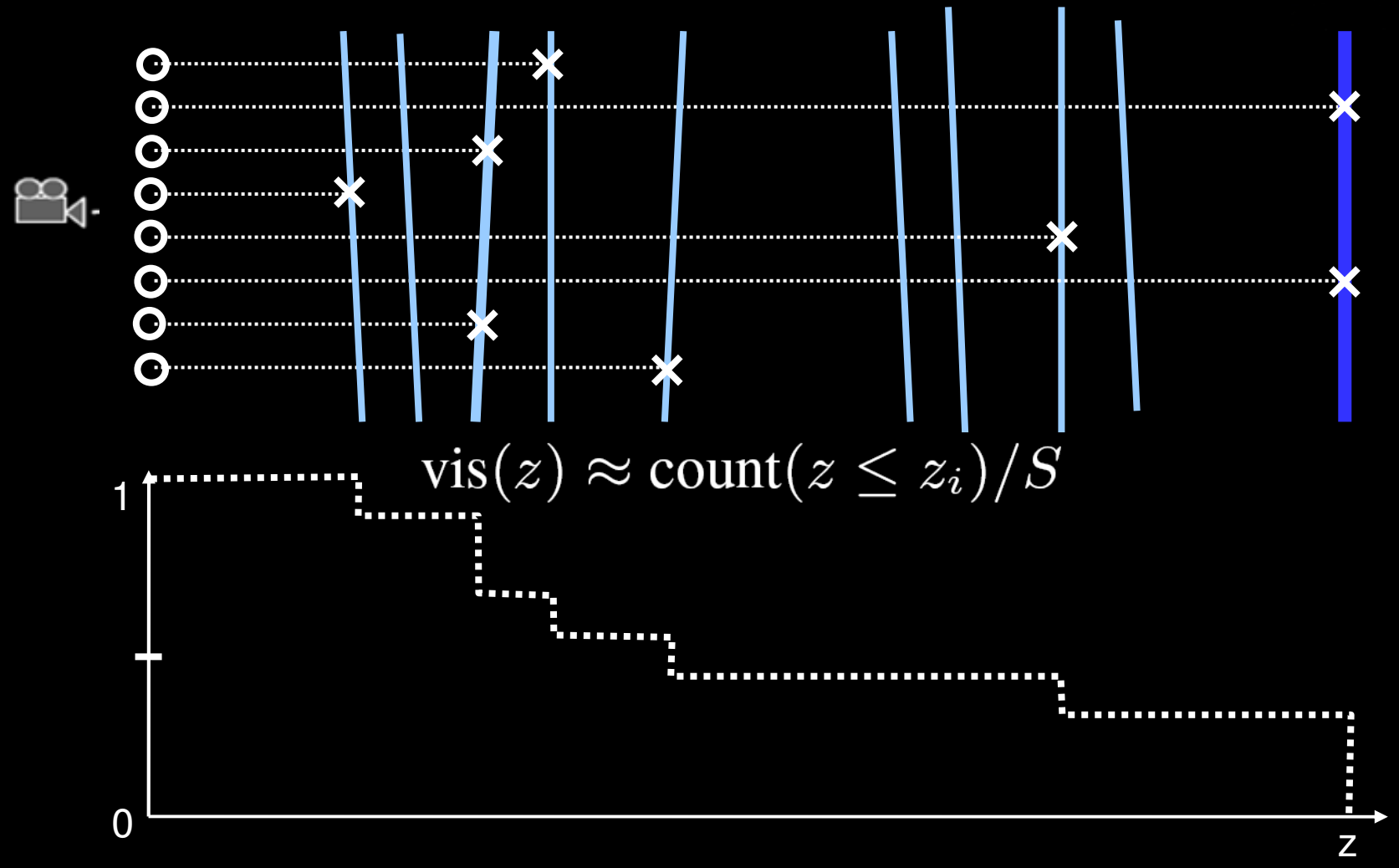
  How much light gets from camera to depth z



(a) A stack of semitransparent objects

- Cf. Deep Shadows [Lokovic and Veach 2000]

Stochastic Shadow Map



Stochastic Shadow Map

Crude, but compact, regular, and parallel:

- Every pixel looks the same
  - S z-values
  - z's not sorted
- Look-up is just PCF
  - S comparisons per shadow-map pixel

Depth-Based Stochastic Transparency

Transparent Shadow Map

- How much light gets from camera to depth z
  = How much light gets from depth z to camera
  = Contribution of fragment at depth z

  → Compute c as a weighted sum of fragment colors

  → Any transparent shadow method
  is also an OIT method.

Stochastic Transparency Algorithms

- "Depth based" stochastic transparency
  - Render stochastic shadow map from the camera
  - Accumulation pass
  - Alpha correction pass

- Basic: 1 pass
- Alpha Corrected: 2 passes
- Depth Based: 3 passes

  (Per 8 spp. Add 2 passes per 8 additional spp.)

Motion

(video #2)

Discussion (1 of 2)

Stochastic Transparency is

- Fast: Fixed passes, fixed memory
- Unified
- Simple
- Parallel
- No sorting!

Discussion (2 of 2)

- 64 spp? “The elegance of brute force”
- Connections to Deep Shadow Maps
- Connections to Monte Carlo Ray Tracing
- Turns transparent stuff into opaque stuff

Thank you!

Thanks also to James Reilley, Lars Nordskog, John Tran and Steve Parker at NVIDIA.

Contact:

eenderton@nvidia.com

erik.sintorn@chalmers.se

www.nvidia.com/research