Towards Automatic Band-Limited Procedural Shaders

Jonathan Dorn, Connelly Barnes,

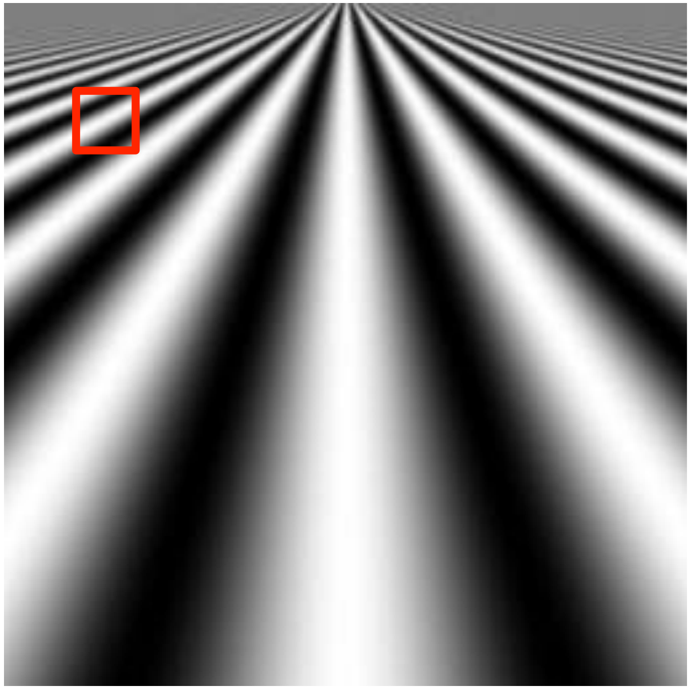
Jason Lawrence, Westley Weimer

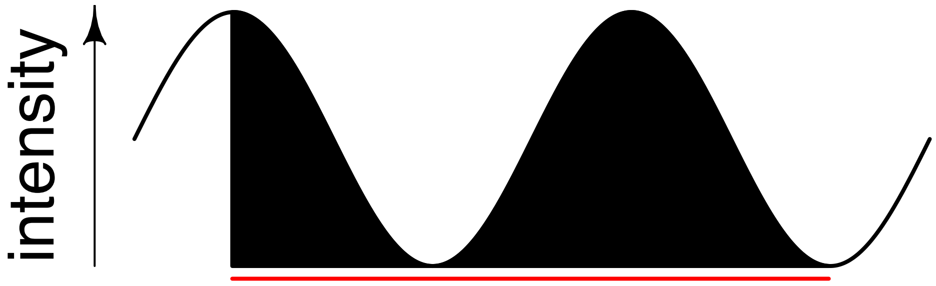
University of Virginia

7 October 2015

PACIFIC GRAPHICS 2015

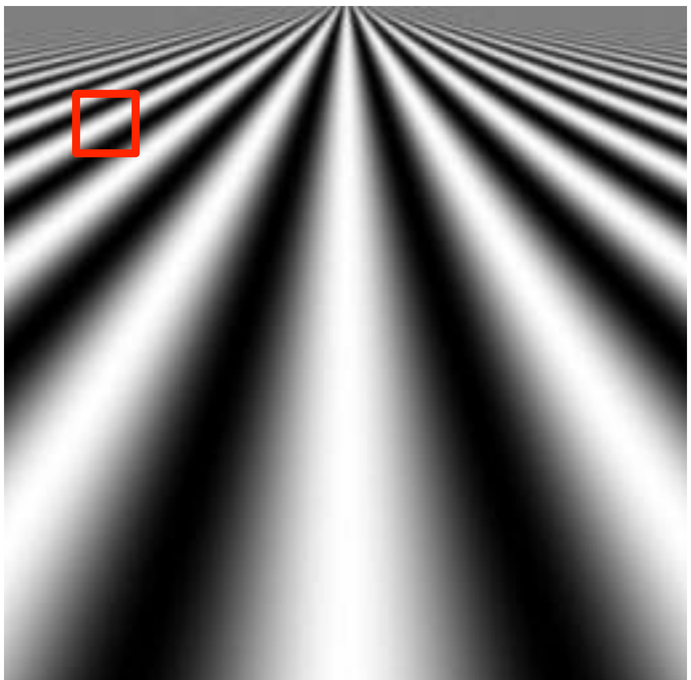
Rendering Textures

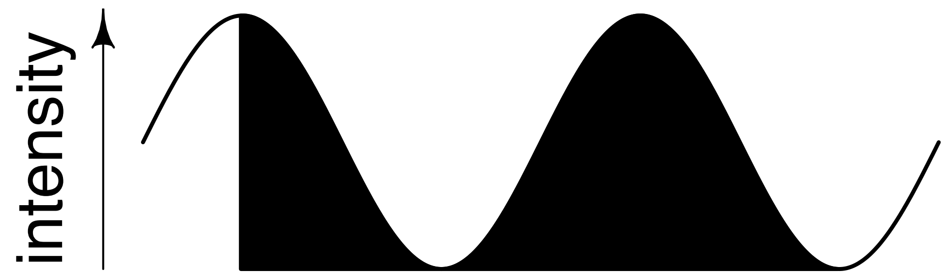




$$\int \cos(s)k(s,w)\,ds$$

Rendering Textures

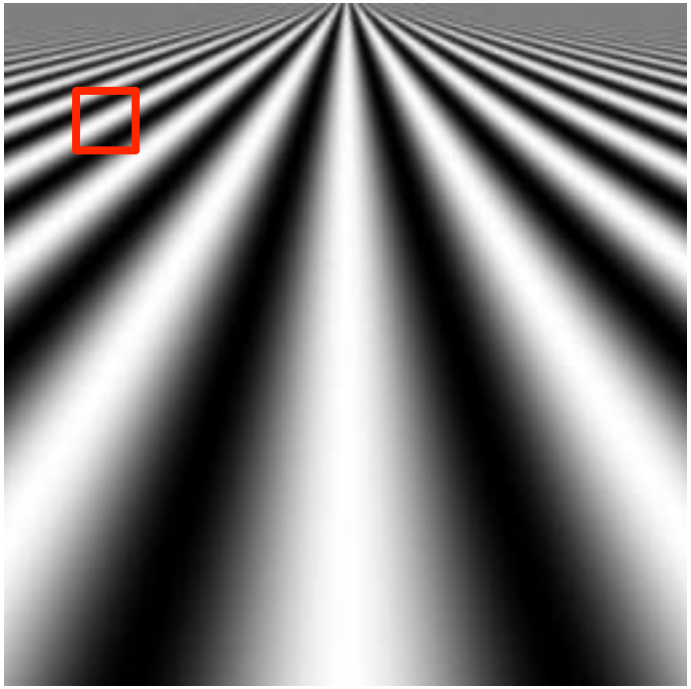


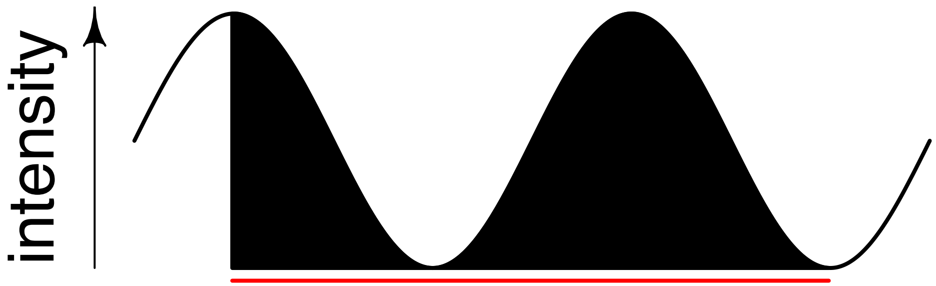


$$\int \cos(s) k(s, w) \, ds$$

Texture function

Rendering Textures

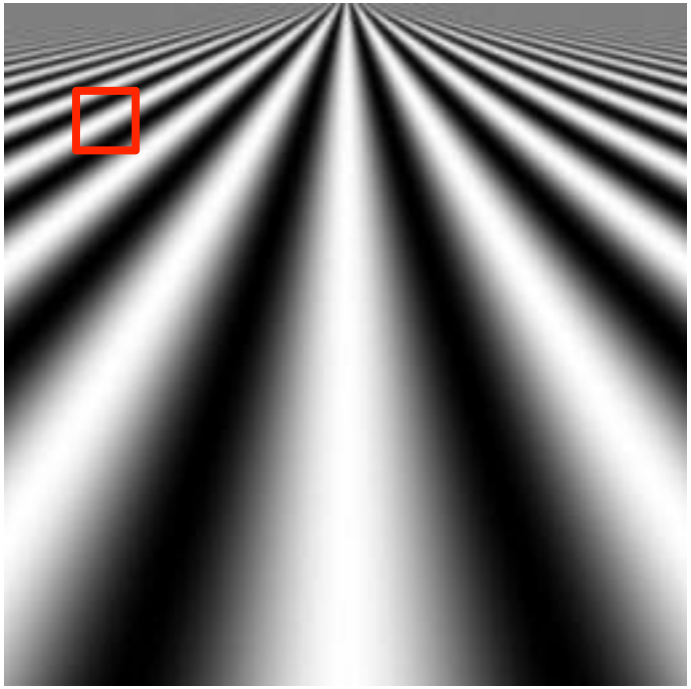


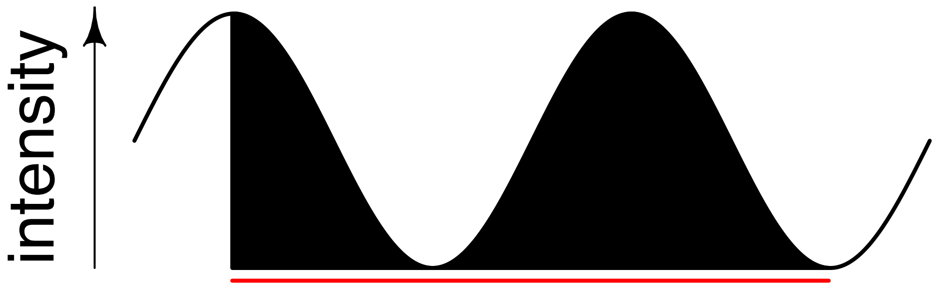


$$\int \cos(s)\,k(s,w)\,ds$$

Pixel footprint

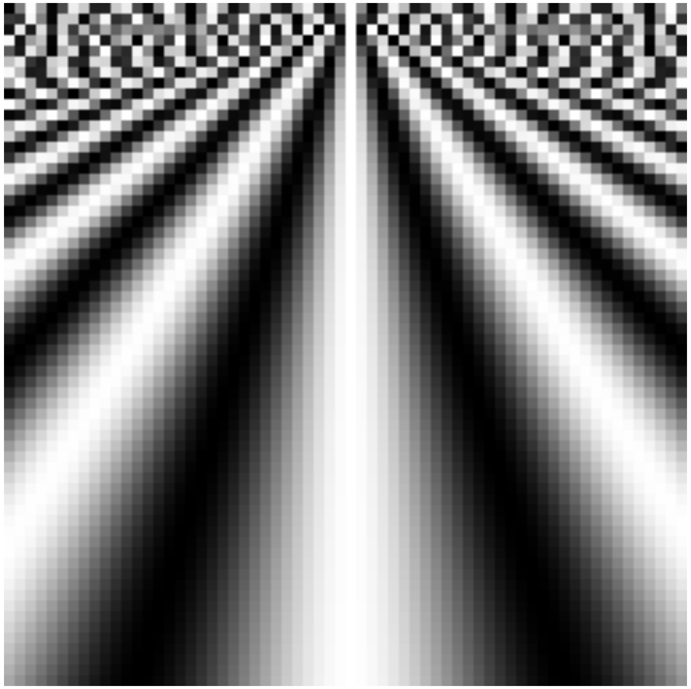
Rendering Textures

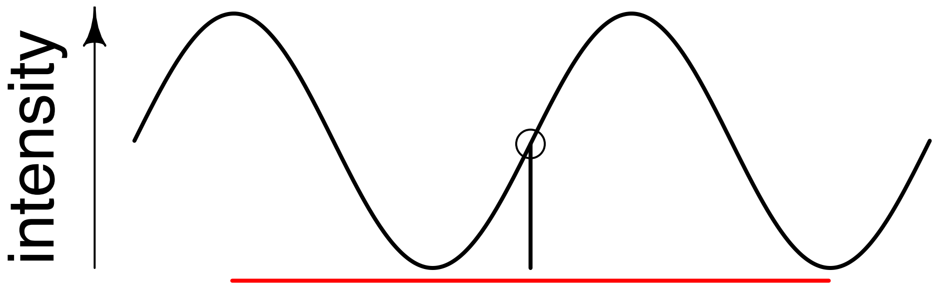




$$\int \cos(s)k(s,w)\,ds$$

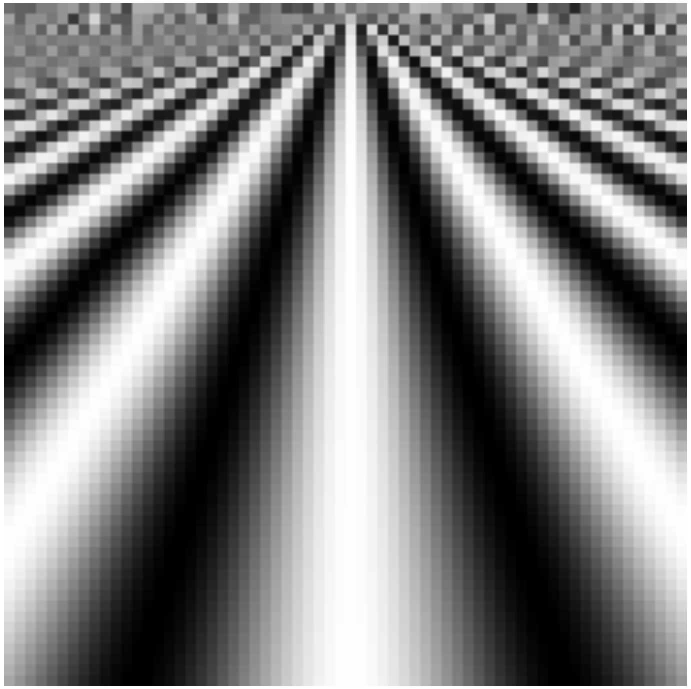
Single-Sample Approximation

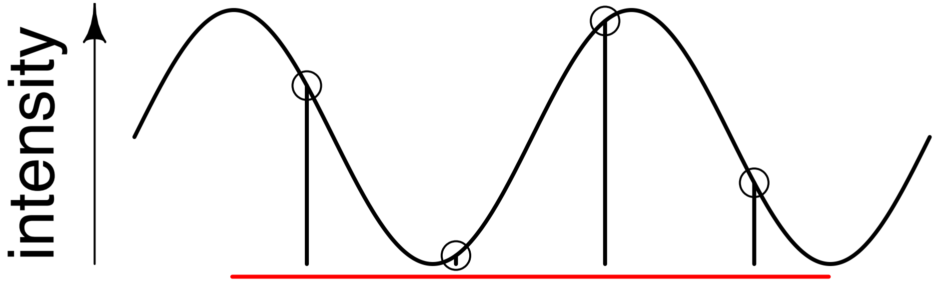




$$\cos(s)k(s,w)$$

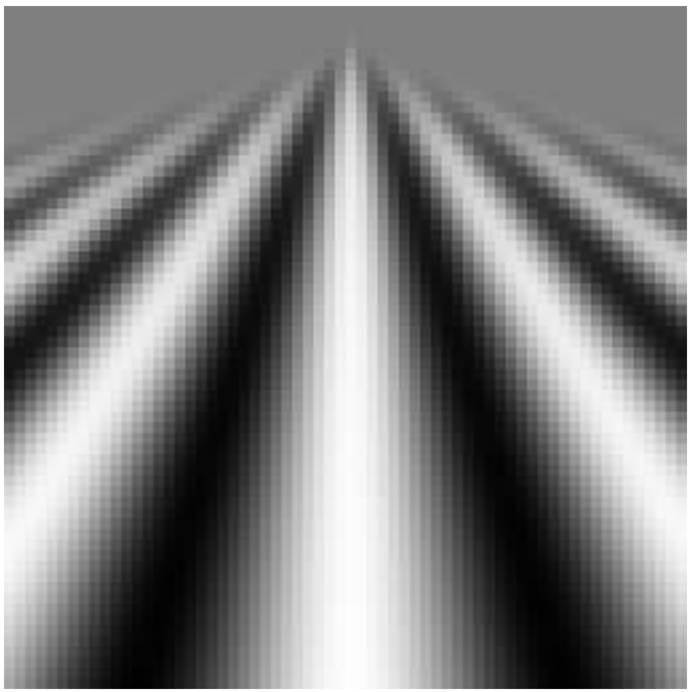
Multi-Sample Approximation

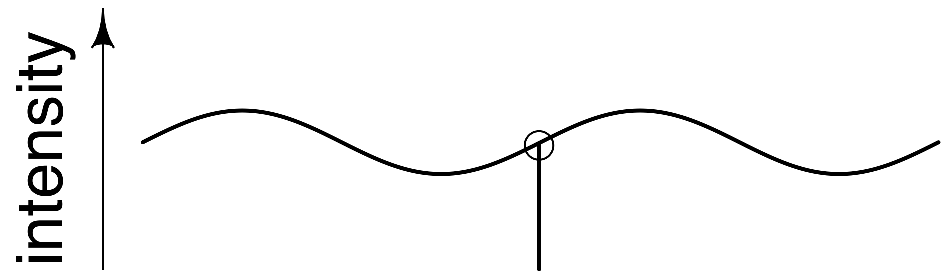




$$\frac{1}{n}\sum_{i=1}^{n}\cos(s_i)k(s_i, w)$$

Band-Limited Functions





$$\widehat{cos}_k(s, w)$$

Band-Limited Procedural Shaders

- Given a procedural shader, generate a new shader that is:
  - ***Visually faithful*** to original,
  - A ***band-limited*** function of sampling rate,
  - ***Efficient*** to compute.

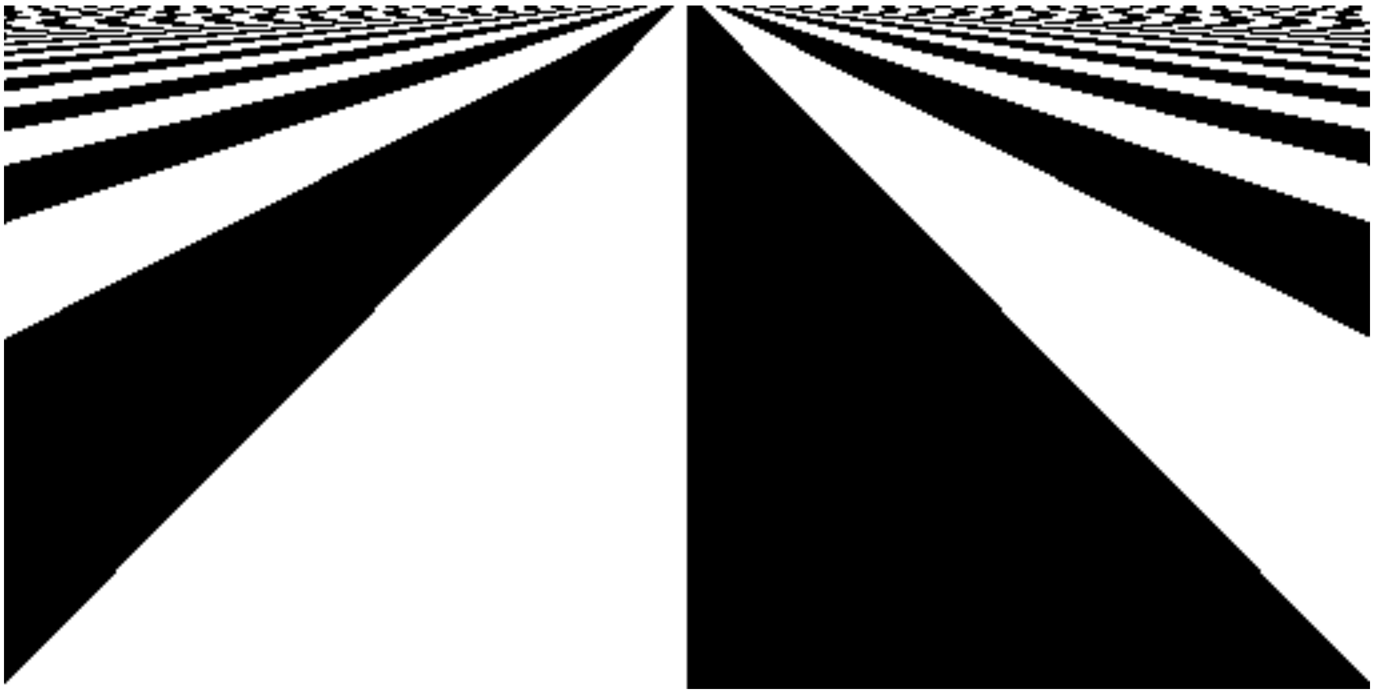
Band-Limited Primitives

1. Solve by hand.
   - See paper and supplemental material for examples.
2. Published solutions.
   - E.g. Gabor noise, summed area tables.

$f(x)$	$\hat{f}(x,w)$
x	x
x^2	x^2+w^2
$fract_1(x)$	$\frac{1}{2}-\sum_{n=1}^{\infty}\frac{\sin(2\pi nx)}{\pi n}e^{-2w^2\pi^2n^2}$
$fract_2(x)$	$\frac{1}{2w}\left(fract^2\left(x+\frac{w}{2}\right)+\left\lfloor x+\frac{w}{2}\right\rfloor-fract^2\left(x-\frac{w}{2}\right)-\left\lfloor x-\frac{w}{2}\right\rfloor\right)$
$fract_3(x)$	$\frac{1}{12w^2}\left(f'(x-w)+f'(x+w)-2f'(x)\right)$
	where $f'(t)=3t^2+2fract^3(t)-3fract^2(t)+fract(t)-t$
$\lvert x\rvert$	$x\,\mathrm{erf}\,\frac{x}{w\sqrt{2}}+w\sqrt{\frac{2}{\pi}}e^{-\frac{x^2}{2w^2}}$
$\lfloor x\rfloor$	$x-\widehat{fract}(x,w)$
$\lceil x\rceil$	$\widehat{floor}(x,w)+1$
$\cos x$	$\cos x\, e^{-\frac{w^2}{2}}$
$saturate(x)$	$\frac{1}{2}\left(x\,\mathrm{erf}\,\frac{x}{w\sqrt{2}}-(x-1)\,\mathrm{erf}\,\frac{x-1}{w\sqrt{2}}+w\sqrt{\frac{2}{\pi}}\left(e^{-\frac{x^2}{2w^2}}-e^{-\frac{(x-1)^2}{2w^2}}\right)+1\right)$
$\sin x$	$\sin x\, e^{-\frac{w^2}{2}}$
$step(a,x)$	$\frac{1}{2}\left(1+\mathrm{erf}\,\frac{x-a}{w\sqrt{2}}\right)$
$trunc(x)$	$\widehat{floor}(x,w)-\widehat{step}(x,w)+1$

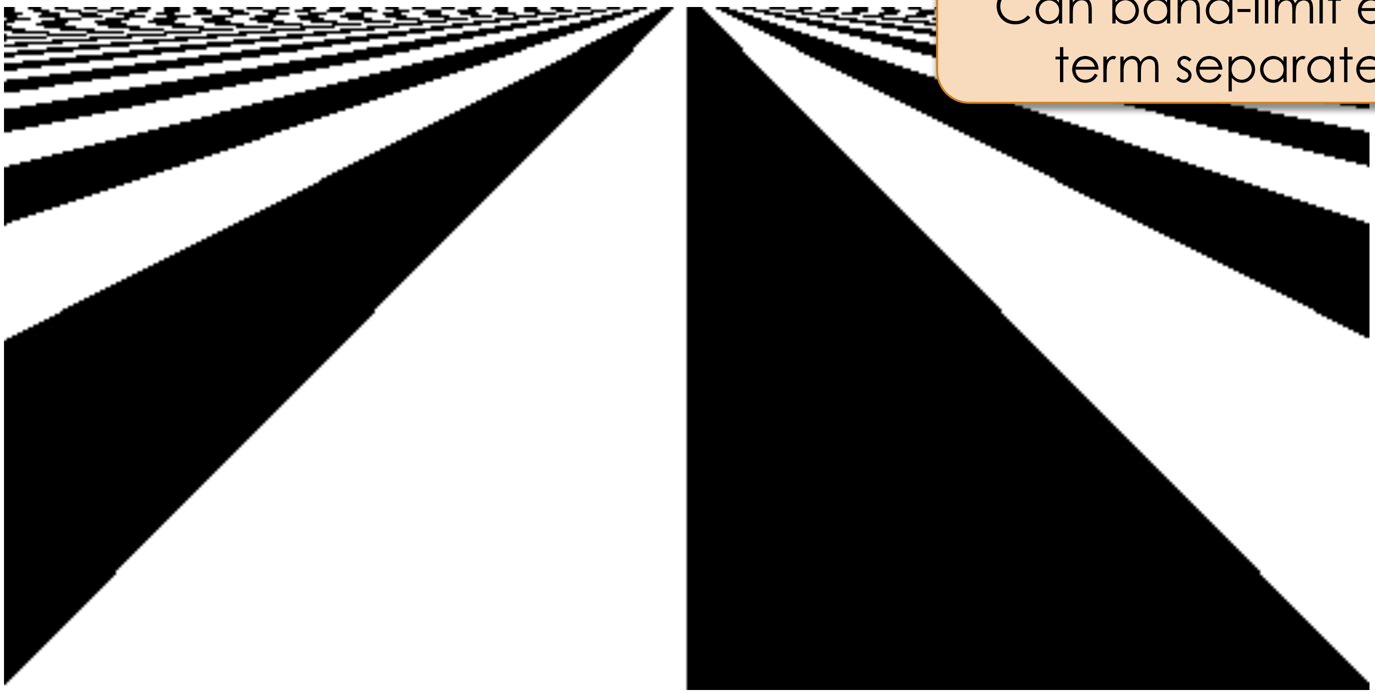
Linear Combinations

$$\lfloor s + 0.5 \rfloor - \lfloor s \rfloor$$



Linear Combinations

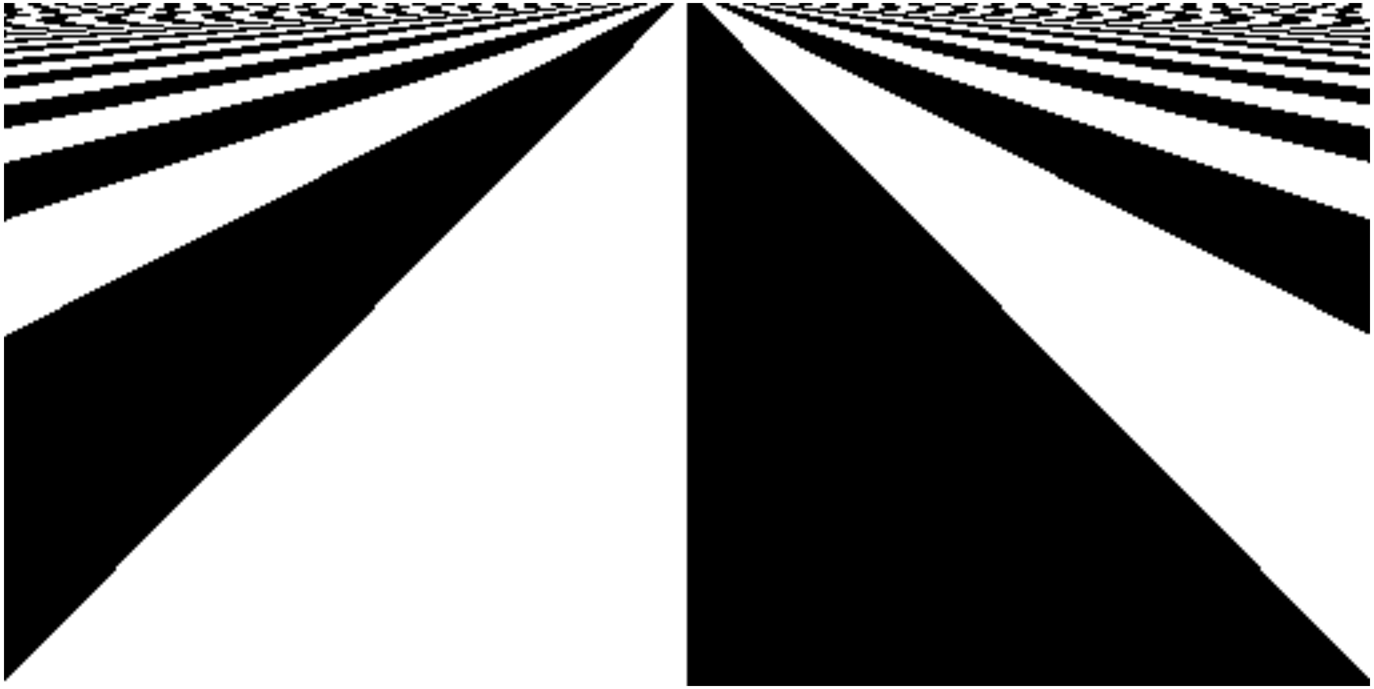
$$\lfloor s + 0.5 \rfloor - \lfloor s \rfloor$$



Linearity of integration: Can band-limit each term separately

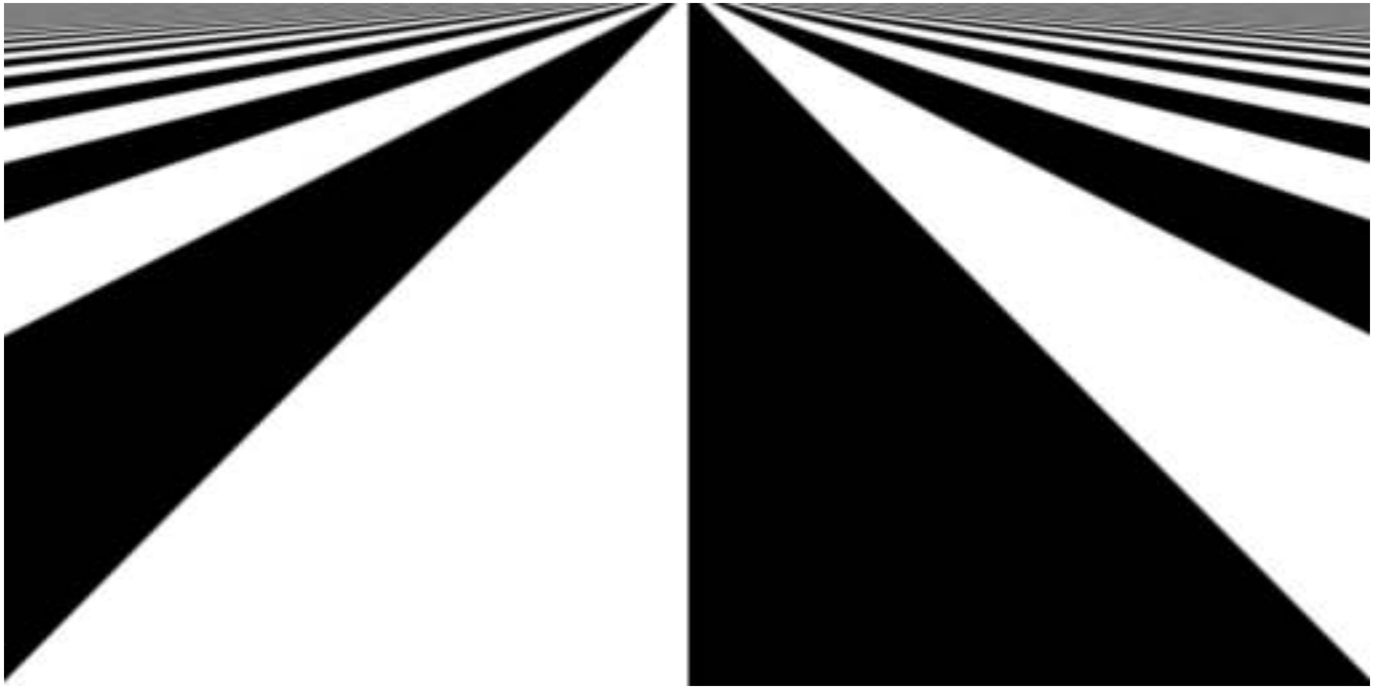
Linear Combinations

$$\lfloor s + 0.5 \rfloor - \lfloor s \rfloor$$



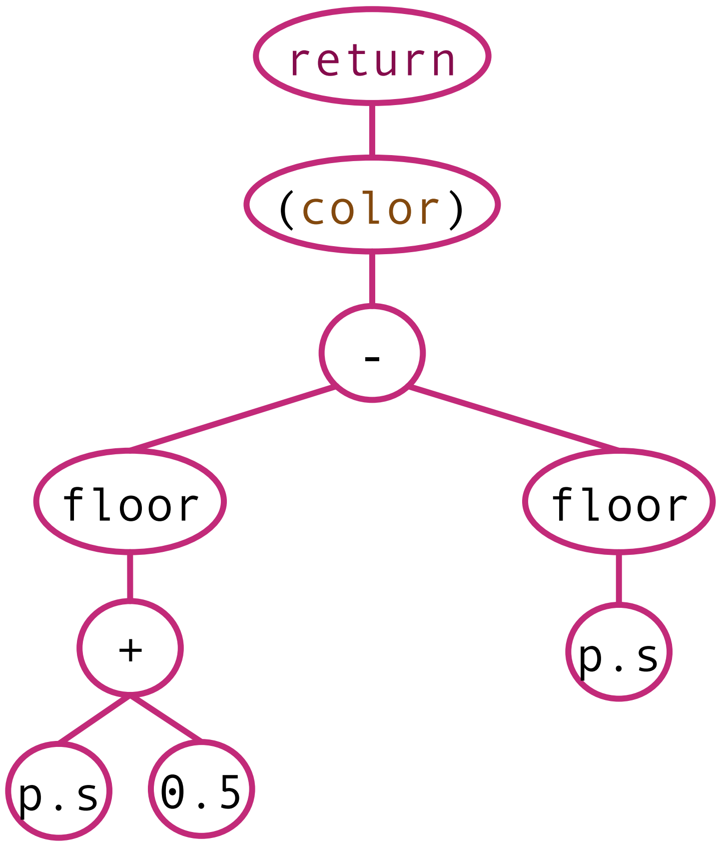
Linear Combinations

$$\widehat{floor}_k(s + 0.5, w) - \widehat{floor}_k(s, w)$$



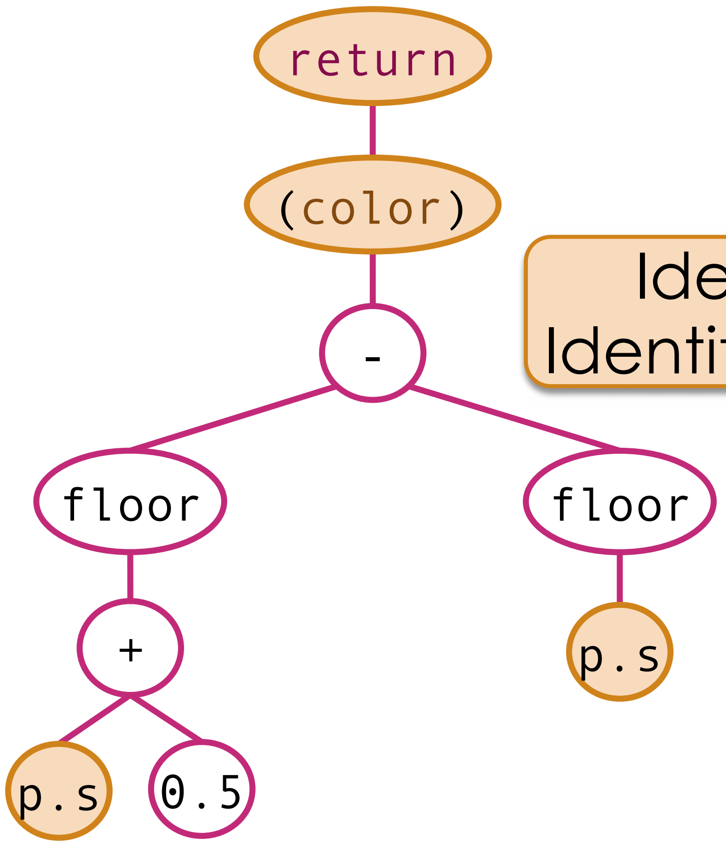
Abstract Syntax Trees (ASTs)

```
color stripe(float2 p) {
  return (color)(floor(p.s + 0.5) - floor(p.s));
}
```



Abstract Syntax Trees (ASTs)

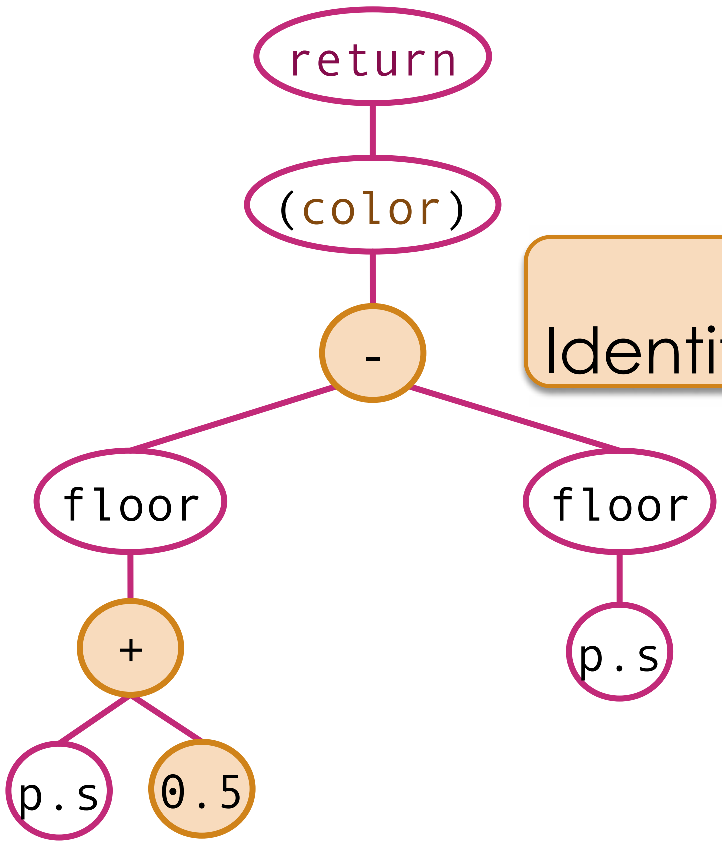
```
color stripe(float2 p) {
  return (color)(floor(p.s + 0.5) - floor(p.s));
}
```



Identity function:
Identity transformation

Abstract Syntax Trees (ASTs)

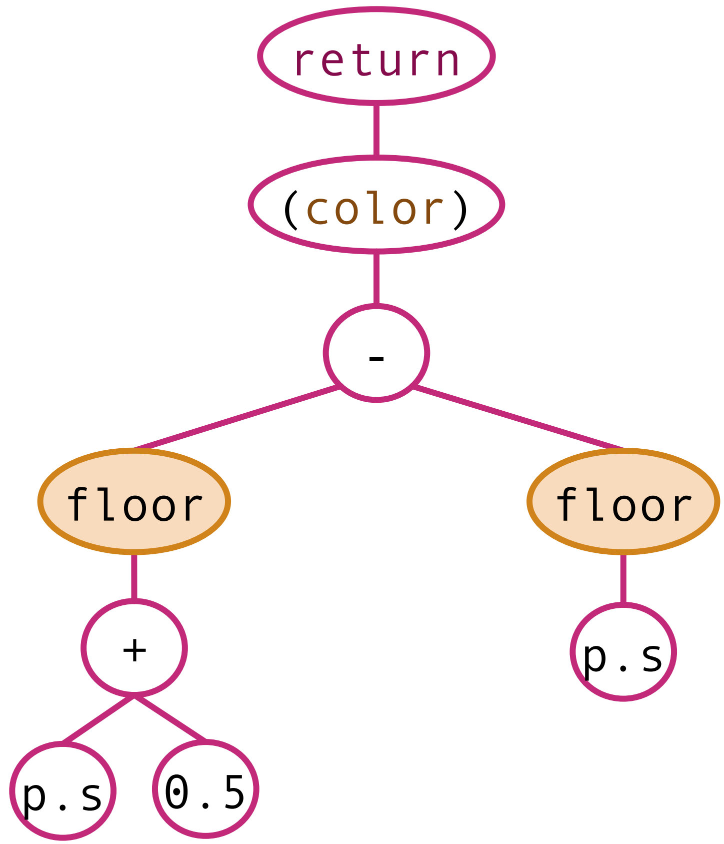
```
color stripe(float2 p) {
  return (color)(floor(p.s + 0.5) - floor(p.s));
}
```



Linearity:
Identity transformation

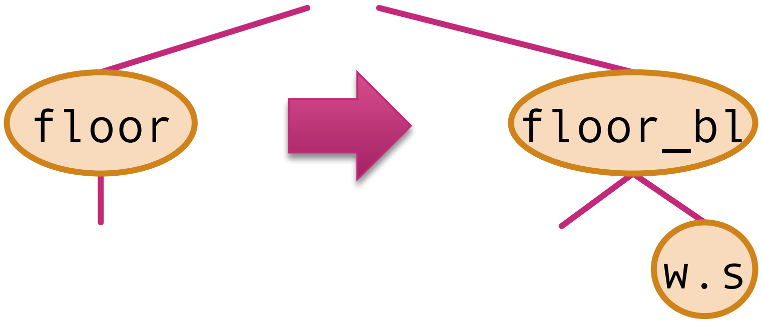
Abstract Syntax Trees (ASTs)

```
color stripe(float2 p) {
  return (color)(floor(p.s + 0.5) - floor(p.s));
}
```



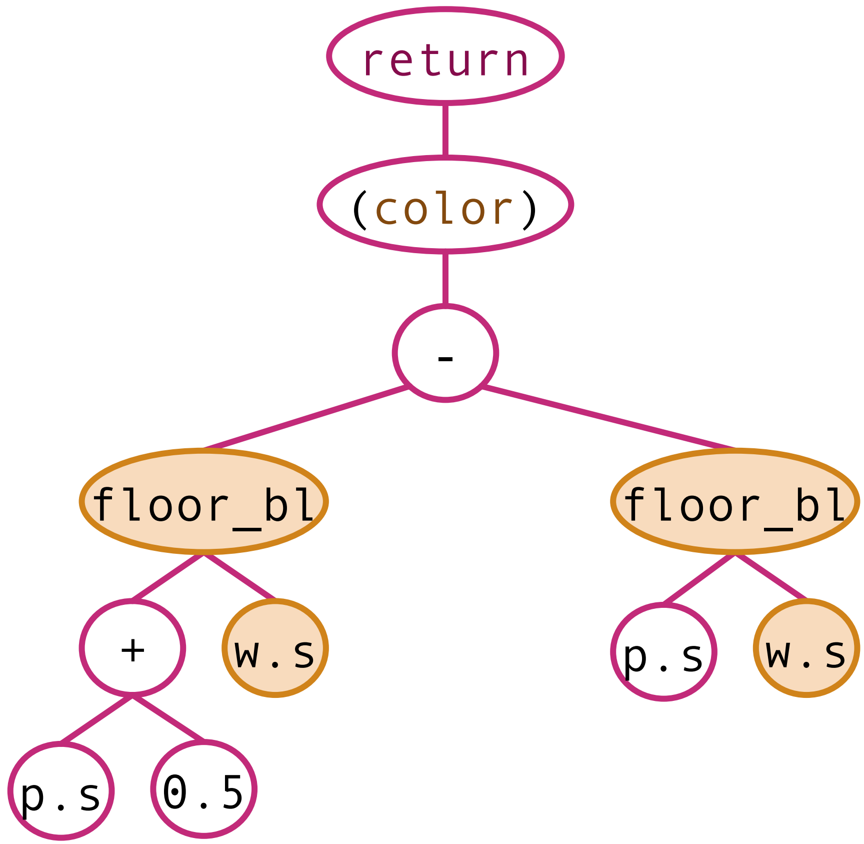
Abstract Syntax Trees (ASTs)

- Transform AST nodes locally.
  - Replace as child of parent.
  - Replace as parent of children.



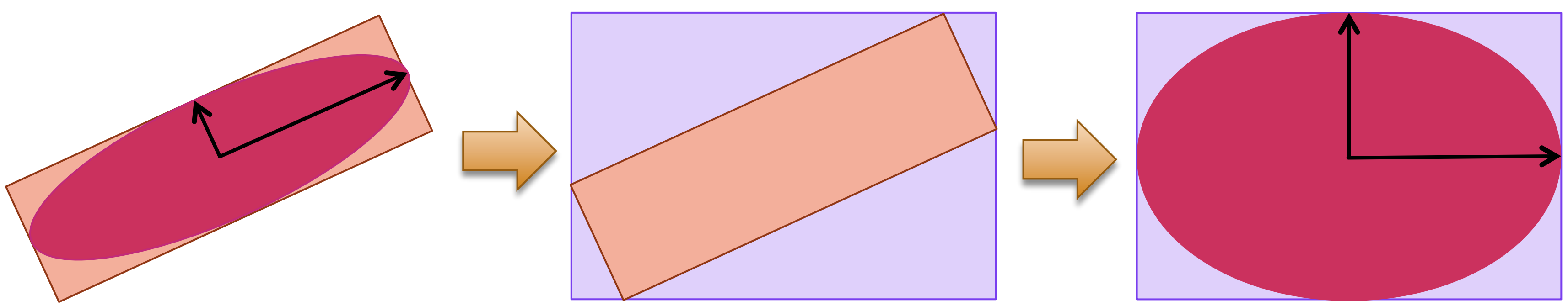
Band-Limited Source Code

```
color stripe_bl(float2 p, float2 w) {
  return (color)(floor_bl(p.s + 0.5, w.s) - floor_bl(p.s, w.s));
}
```



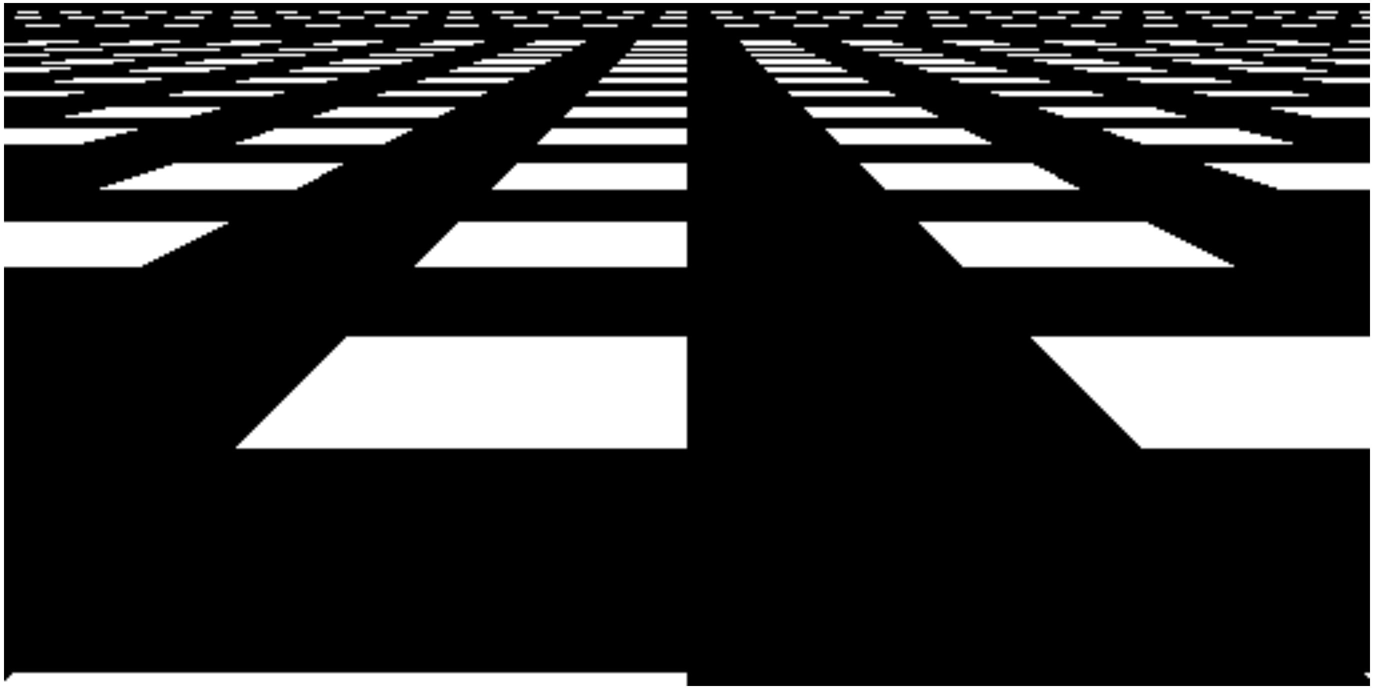
Multiplicative Combinations

- If shader ***and*** pixel footprint are multiplicatively separable, the separate parts can be band-limited separately.
- We ensure separable pixel footprint with a bounding box approximation.



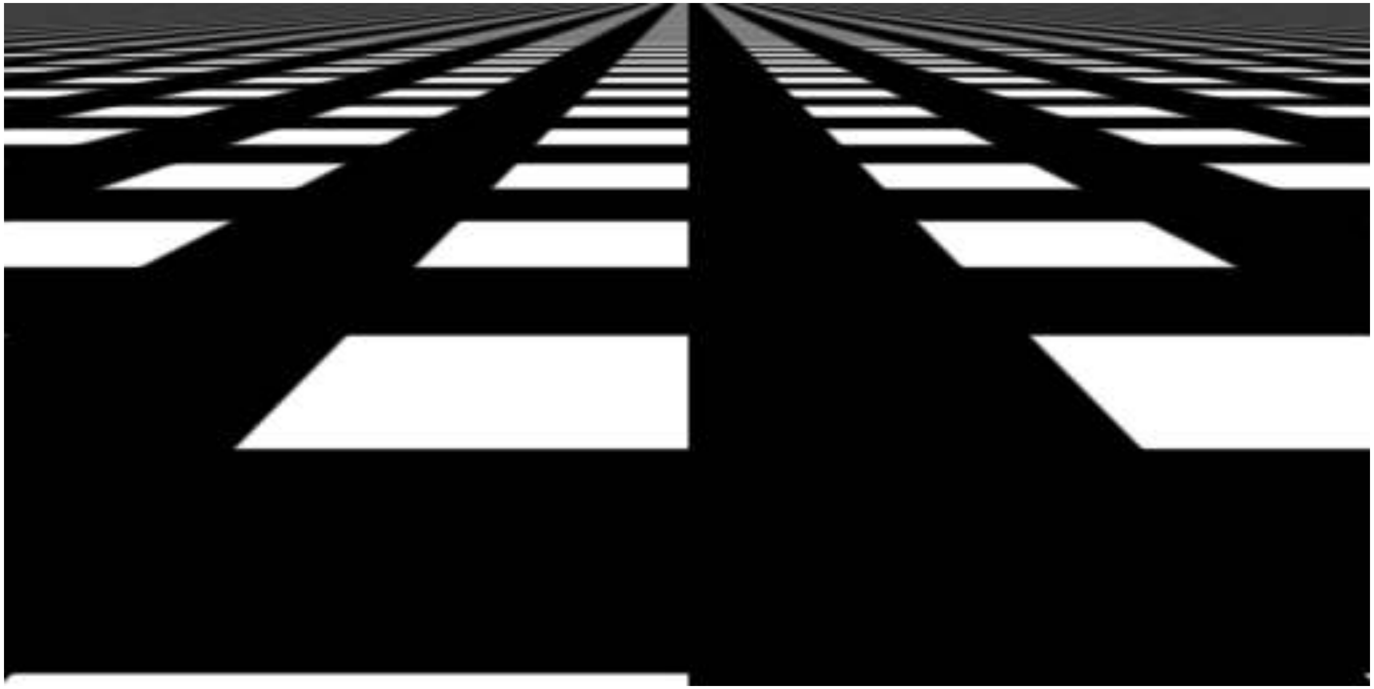
Multiplicative Combinations

$$\big(\lfloor s + 0.5 \rfloor - \lfloor s \rfloor\big)\big(\lfloor t + 0.5 \rfloor - \lfloor t \rfloor\big)$$



Multiplicative Combinations

$$\left(\widehat{floor}_k(s+0.5, w_s) - \widehat{floor}_k(s, w_s)\right)\left(\widehat{floor}_k(t+0.5, w_t) - \widehat{floor}_k(t, w_t)\right)$$



Function Composition

$$\int f(g(x))k(x,w)\,dx = \boxed{?}$$

- Our approach: transform subsets of functions.
  - E.g., $\widehat{f_k}(g(x), w)$
  - Genetic search.

$$\widehat{f \circ g}(x, \boxed{?})$$

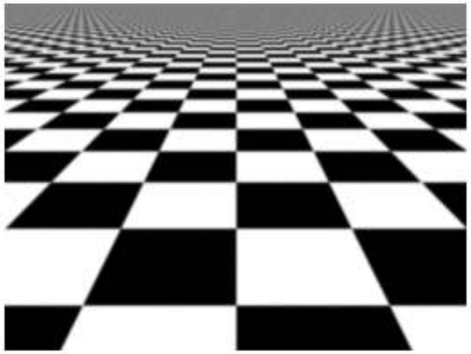
- Our approach: linear combination of sampling rates.
  - $w = c_0 + c_1 w_x + c_2 w_y$
  - Simplex search.

Approach Overview

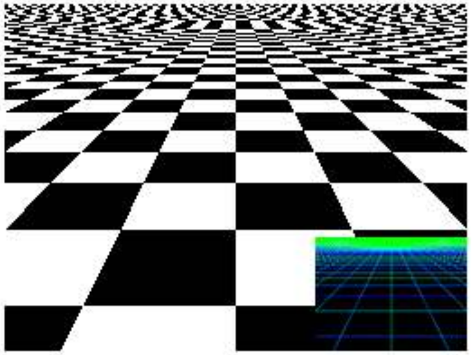
1. Identify AST nodes that may be transformed.
2. Select subset with genetic search.
3. Fit sampling rate coefficients with simplex search.
4. Replace selected nodes using fitted sampling rates.

Results: Checkerboard

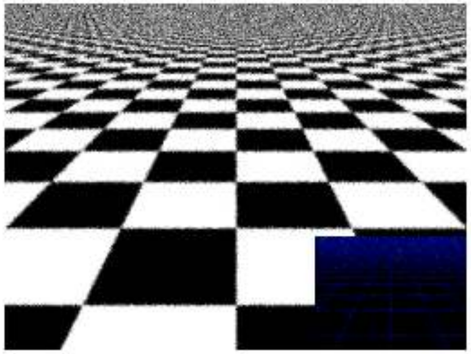
Target Image



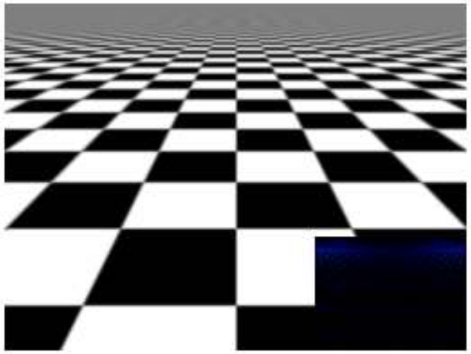
No Antialiasing



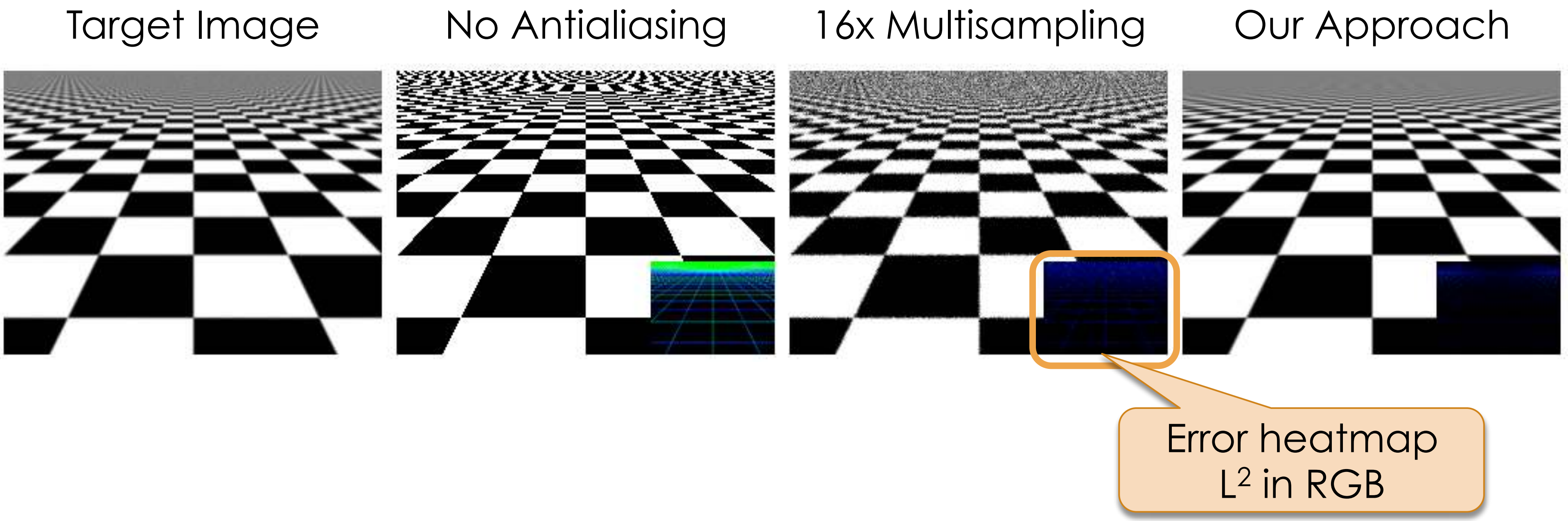
16x Multisampling



Our Approach

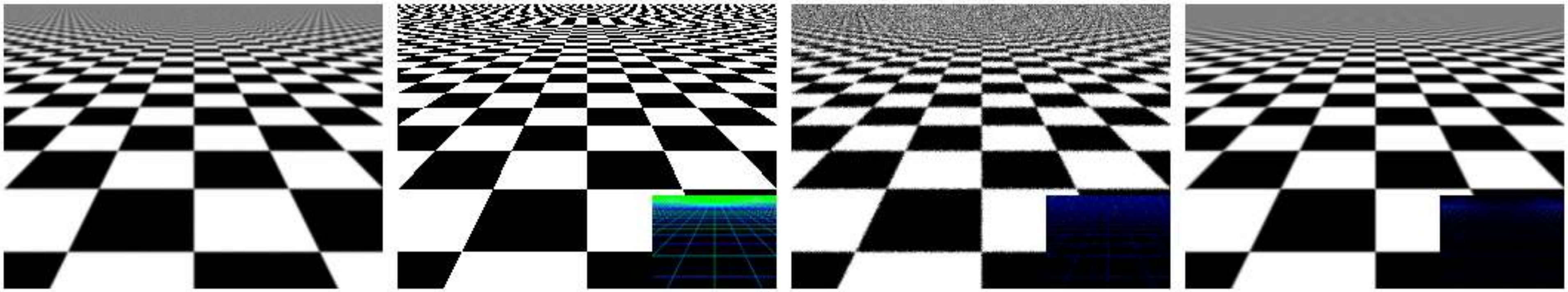


Results: Checkerboard



Results: Checkerboard

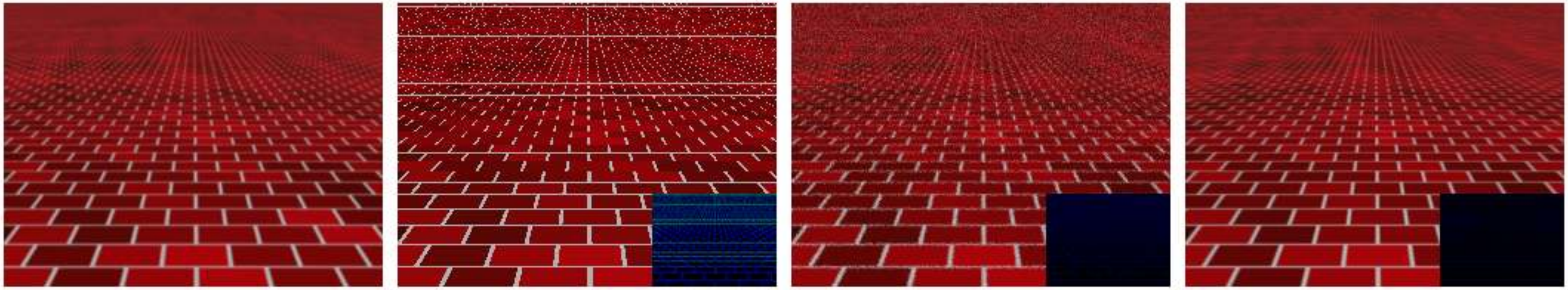
Target Image | No Antialiasing | 16x Multisampling | Our Approach



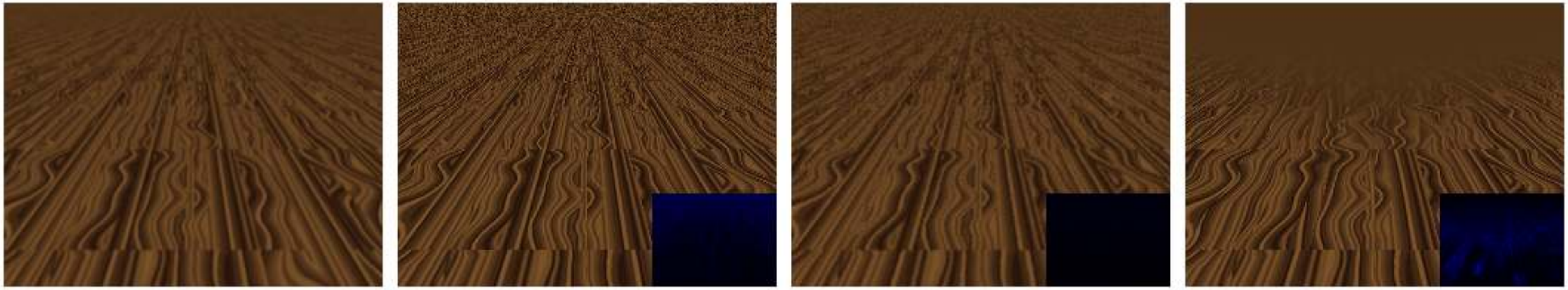
- **4x faster** than multisampling.
- **2x less L^2 (RGB) error** than multisampling.
  - Error is due to separable kernel approximation.

Results: Brick and Wood

Target Image | No Antialiasing | 16x Multisampling | Our Approach



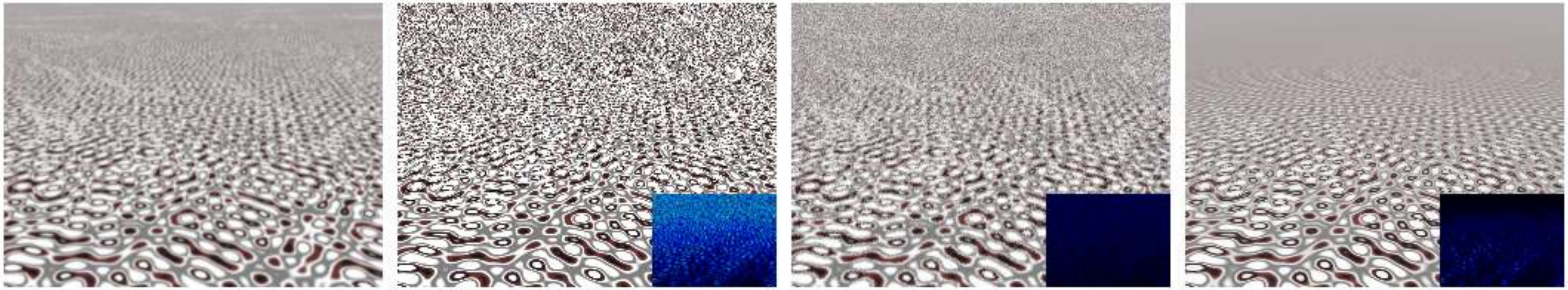
6x faster, 2x less L^2 error than multisampling.



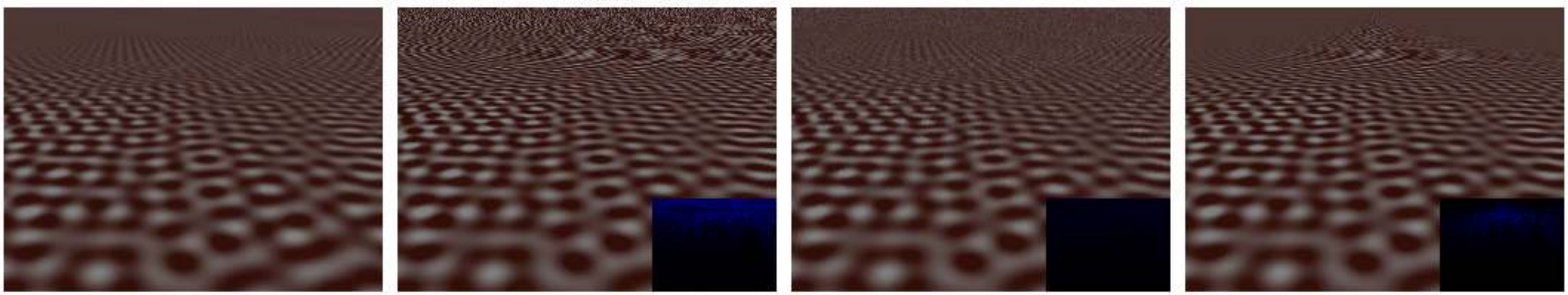
5x faster, 3x more L^2 error than multisampling.

Results: Procedural Noise

Target Image | No Antialiasing | 16x Multisampling | Our Approach



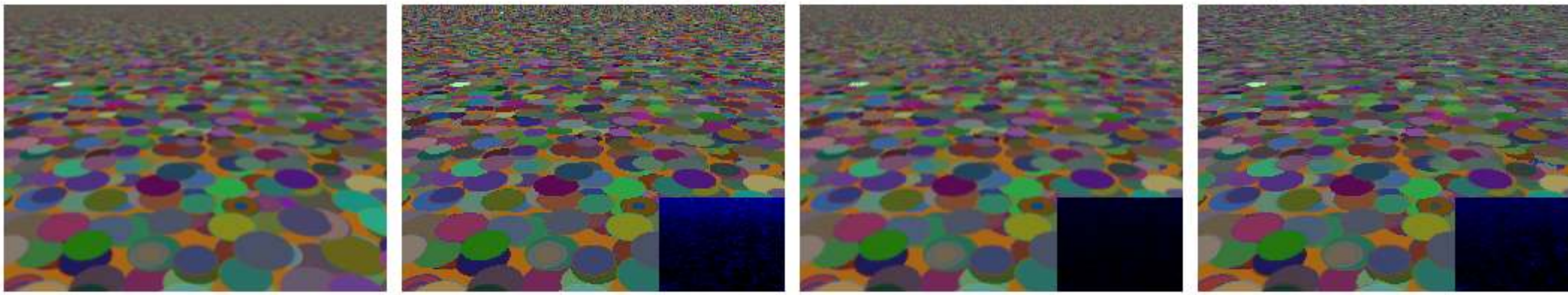
6x faster, equivalent L^2 error to multisampling.



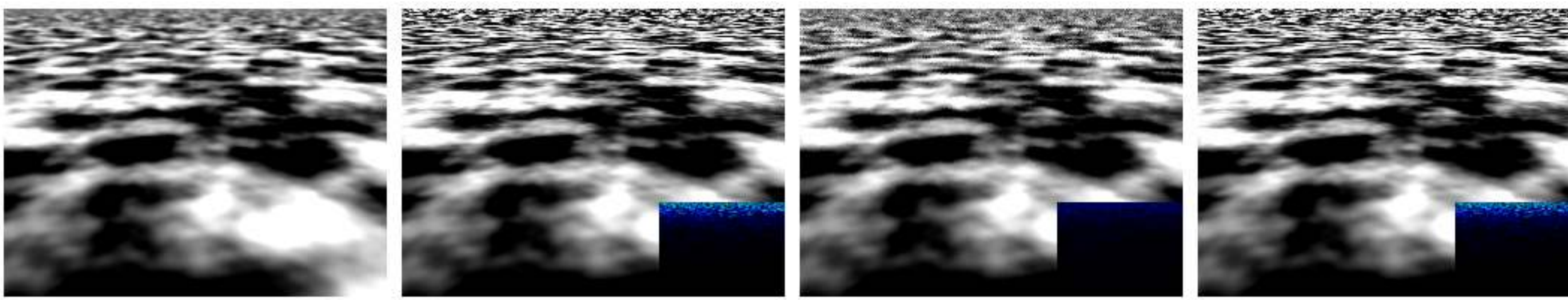
6x faster, equivalent L^2 error to multisampling.

Results: More Complex Procedures

Target Image | No Antialiasing | 16x Multisampling | Our Approach



20x faster, 2x more L^2 error than multisampling.



17x faster, 2x more L^2 error than multisampling.

Future Work

- More sophisticated fitness functions.
- Spatially varying loop bounds.
- Merging conditional branches.
- Investigate alternative transformations.
- Language support.

Conclusion

- Explore problem of automatically band-limiting general texture shaders.
- Band-limited shaders are faster than multisampling.
- Work remains to scale up.

Questions?

PACIFIC GRAPHICS 2015