Cinematic Depth Of Field

How to make big filters cheap

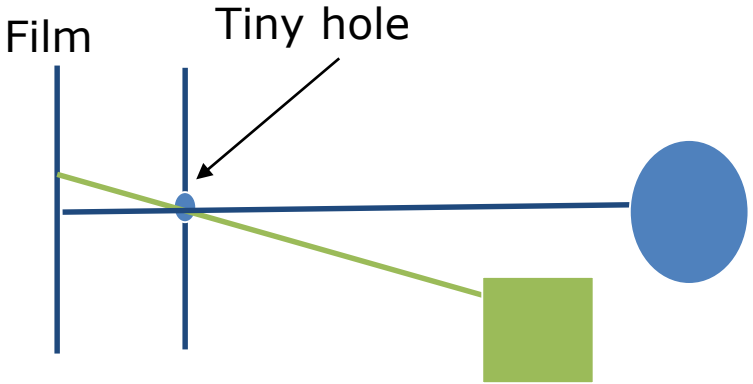
Karl Hillesland
Sean Skelton
AMD

Outline

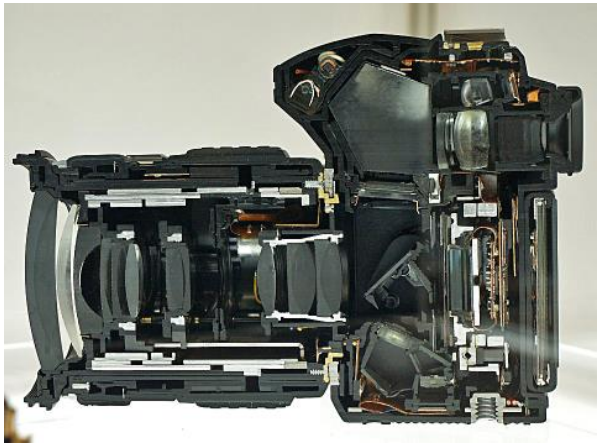
- Depth of Field
- Real-time approximations
- Fast Filter Spreading
- Implementation

Depth of Field

Depth of Field Optics



Pinhole Camera

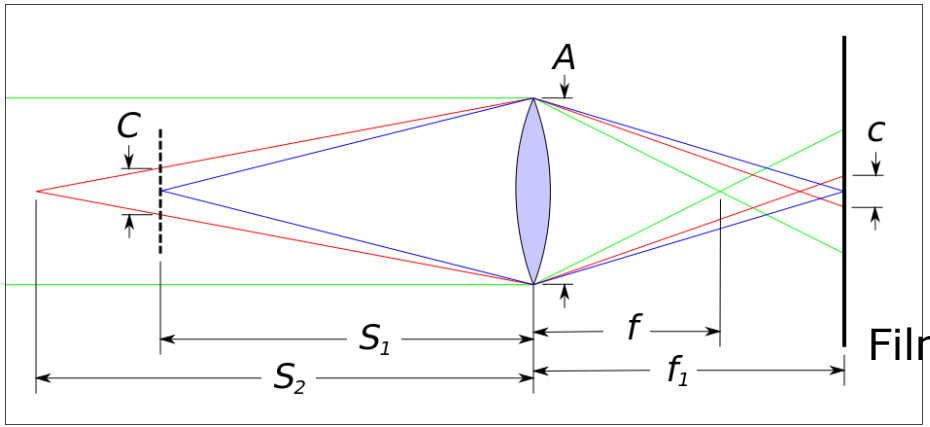


Real Camera

Thin Lens Model

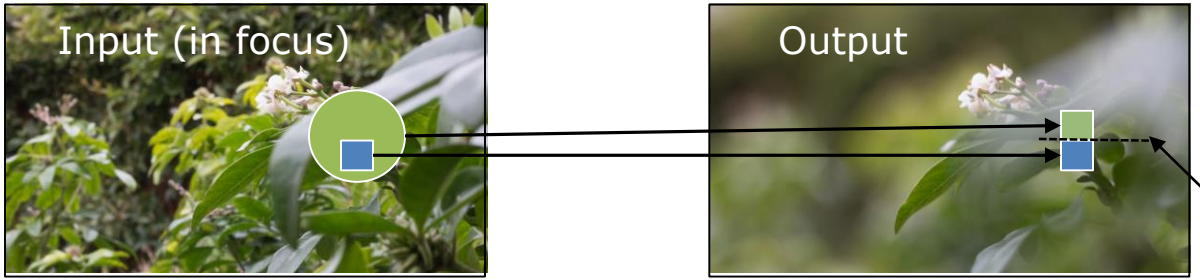
Lens focuses a distance onto film (blue in this case)

f = focal length
A = aperture width
c = circle of confusion (CoC) on film
C = CoC in world

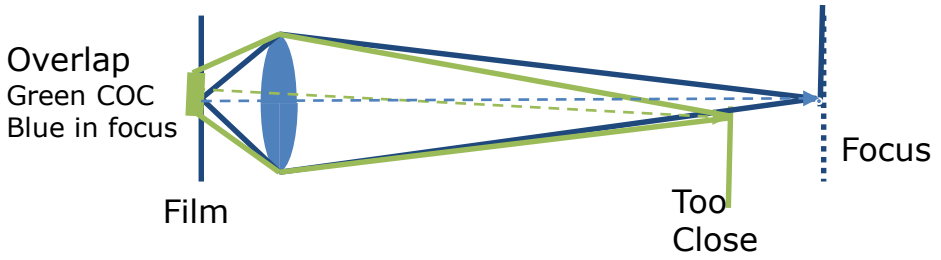


From Wikipedia "Circle of Confusion"

Gather Methods

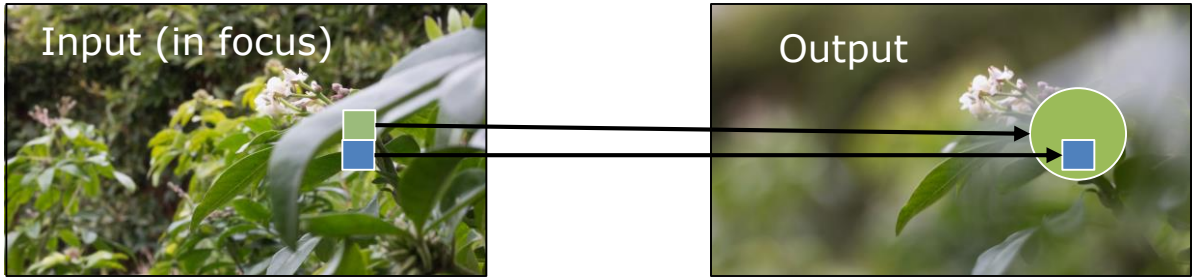



```
for i,j:
    CoC = f ( z(i,j) – zInFocus )
    Out_ij = Gather( CoC );
```



Scatter Methods

Where does each input go?




```
for i,j:
    CoC = f ( z(i,j) - zInFocus );
    Spread( Input_ij, CoC );
```

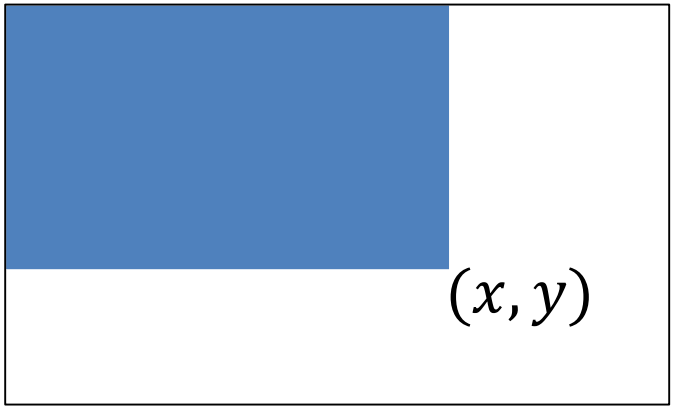
Draw a sprite for each input pixel

Gather from SAT

Compute Summed Area Table

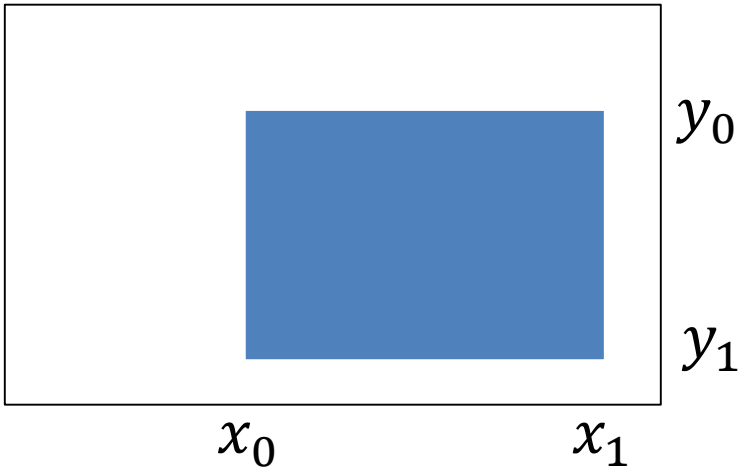
Gather with 4 lookups

Summed Area Table



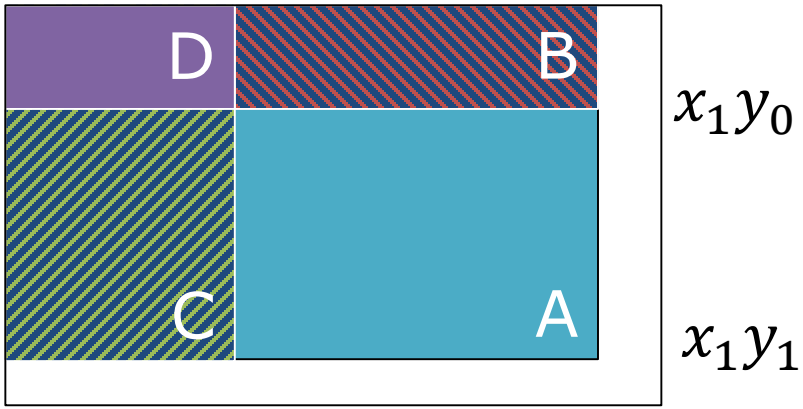
(x,y) contains $\sum_{i=0}^{x}\sum_{j=0}^{y} I_{i,j}$

Gather from SAT



$$S_A - S_B - S_C + S_D$$

=



$$S_A = \sum_{i=0}^{x_1} \sum_{j=0}^{y_1} I_{i,j} \qquad S_B = \sum_{i=0}^{x_1} \sum_{j=0}^{y_0} I_{i,j}$$

Gather SAT

Niceness

- O(1) in filter size

Problems

- Box filter
- Gather method
- Precision Problem

Fast Filter Spreading

- *Fast Filter Spreading and its Applications*
  Kosloff, Hensley, Barsky, UC Berkeley tech report.
- Scatter Method
- SAT in reverse
  - Poke (add) deltas at the corners
  - Then sum

Simple Example

Desired Result

1	1		
1	1		

+

	2	2	
	2	2	

=

1	1		
1	3	2	
	2	2	

Simple Example

Adding Deltas

1		-1	
-1		1	

+

	2		-2
	-2		2

=

1		-1	
	2		-2
-1		1	
	-2		2

Simple Example

Summing for the result

1		-1	
	2		-2
-1		1	
	-2		2

→

1	1		
	2	2	
-1	-1		
	-2	-2	

↓

1	1		
1	3	2	
	2	2	

Still Only 4 Each

1					-1		
			2				-2
-1					1		
			-2				2

1	1	1	1	1			
1	1	1	1	1			
1	1	1	1	1			
1	1	1	3	3	2	2	
1	1	1	3	3	2	2	
			2	2	2	2	
			2	2	2	2	

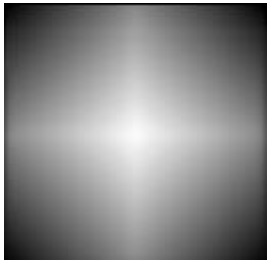
Bartlett

1	-2	1
-2	4	-2
1	-2	1

2 integrations

1	-1	0
-1	1	0
0	0	0

After 1st integration



Implementations

Ladybug

Big demo

Closed source

DOF FX

Simple demo

Open source

Implementation Topics

- Weights and Normalization
- Fixed point math
- Atomics

Weights and Normalization

- Box filter: divide by area
- Bartlett: divide by area*area
- Other filters/weights possible
- Catch all:
  - Accumulate weights in alpha
  - Divide at the end

Fixed Point Math

- Atomics
- Precision
- Scale examples @ ~ 64 x 64, 32 bits
  - 6*2 (box area) = 12 bits -> 20 remaining
  - 6*4 = 24 bits -> 8 remaining
  - 6*3 = 18 bits -> 14 remaining

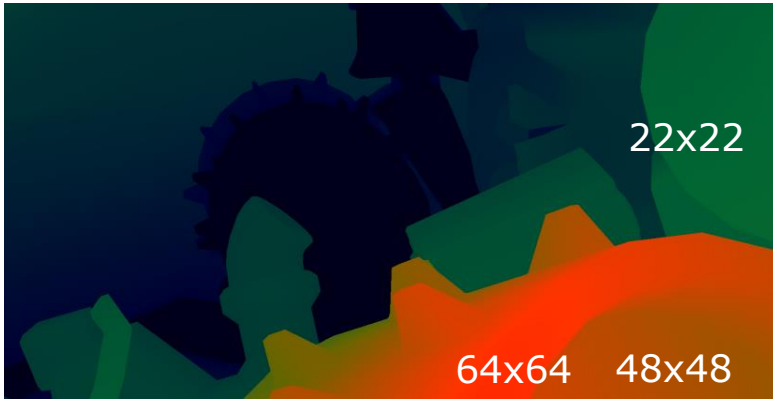
Compute Shader Sample

```
// Scale for fixed point and filter size
int4 intColor = normalizeBlurColor(float4(vColor, 1.0), blur_amount);
for (int i = 0; i < 9; ++i)
{
    // Offset the location by location of the delta and padding
    // Need to offset by (1,1) because the kernel is not centered
    const int2 delta = bartlettData[i].xy * (blur_amount + 1);
    int2 bufLoc = loc.xy + delta + padding + uint2(1, 1);

    // Filter weight
    const int delta_value = bartlettData[i].z;
```

Compute Shader Sample

```
for (int i = 0; i < 9; ++i)
{
    // Offset (previous slide)
    // Weight (previous slide)

    // Write the delta
    // Use interlocked add to prevent the threads from stepping on each other
    // 4 atomics, including alpha channel for normalization.
    InterlockedAddToBuffer(deltaBuffer, bufLoc, intColor * delta_value);
}
```

Results



DOF FX Performance

- Radeon™ RX 480
  - ~5 ms for 1080 p
  - ~3 ms for 1080 p with quarter res
- Looking for further optimizations

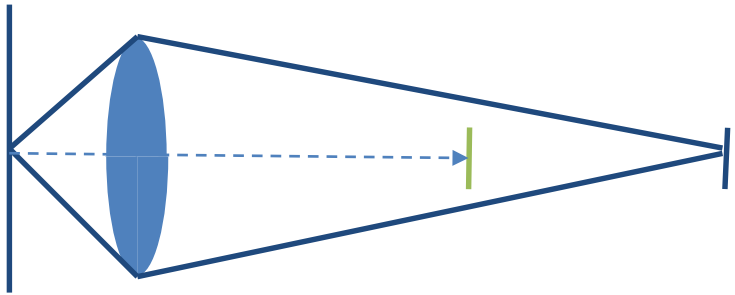
Results

64x64 = 4k / pixel

DOF FX Performance

- Radeon™ RX 480
  - ~5 ms for 1080 p
  - ~3 ms for 1080 p with quarter res
- Looking for further optimizations

Depth of Field



Summary

- Scatter based DOF
- Fixed, but high-ish overhead
- Limited by precision
- Source available soon

Questions?

The information presented in this document is for informational purposes only and may contain technical inaccuracies, omissions and typographical errors.

The information contained herein is subject to change and may be rendered inaccurate for many reasons, including but not limited to product and roadmap changes, component and motherboard version changes, new model and/or product releases, product differences between differing manufacturers, software changes, BIOS flashes, firmware upgrades, or the like. AMD assumes no obligation to update or otherwise correct or revise this information. However, AMD reserves the right to revise this information and to make changes from time to time to the content hereof without obligation of AMD to notify any person of such revisions or changes.

AMD MAKES NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE CONTENTS HEREOF AND ASSUMES NO RESPONSIBILITY FOR ANY INACCURACIES, ERRORS OR OMISSIONS THAT MAY APPEAR IN THIS INFORMATION.

AMD SPECIFICALLY DISCLAIMS ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE. IN NO EVENT WILL AMD BE LIABLE TO ANY PERSON FOR ANY DIRECT, INDIRECT, SPECIAL OR OTHER CONSEQUENTIAL DAMAGES ARISING FROM THE USE OF ANY INFORMATION CONTAINED HEREIN, EVEN IF AMD IS EXPRESSLY ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

ATTRIBUTION