learn network inspire

Game Developers Conference 08

February 18-22, 2008
San Francisco

www.gdconf.com

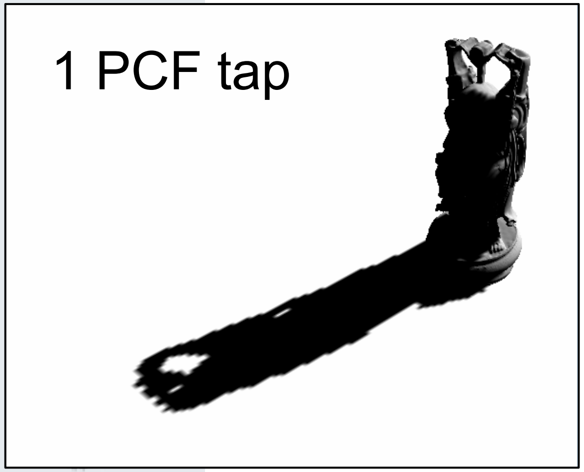
CMP United Business Media

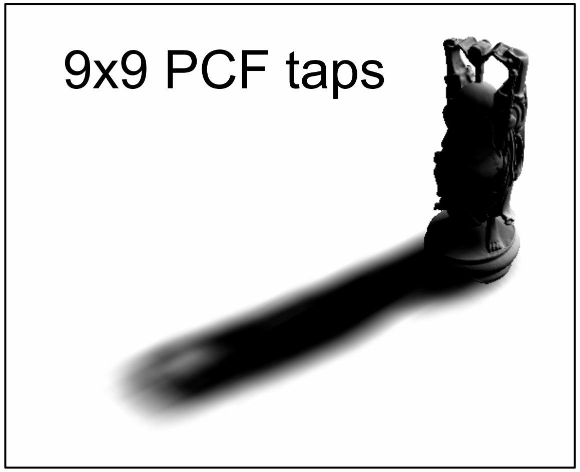
Advanced Soft Shadow Mapping Techniques

Louis Bavoil
NVIDIA Developer Technology

Why Soft Shadows?

- Antialiasing
  - Filtering the shadow edges
- Area lights
  - Cast penumbra with variable size
  - Shadows hardening on contact





Why Shadow Mapping for Soft Shadows?

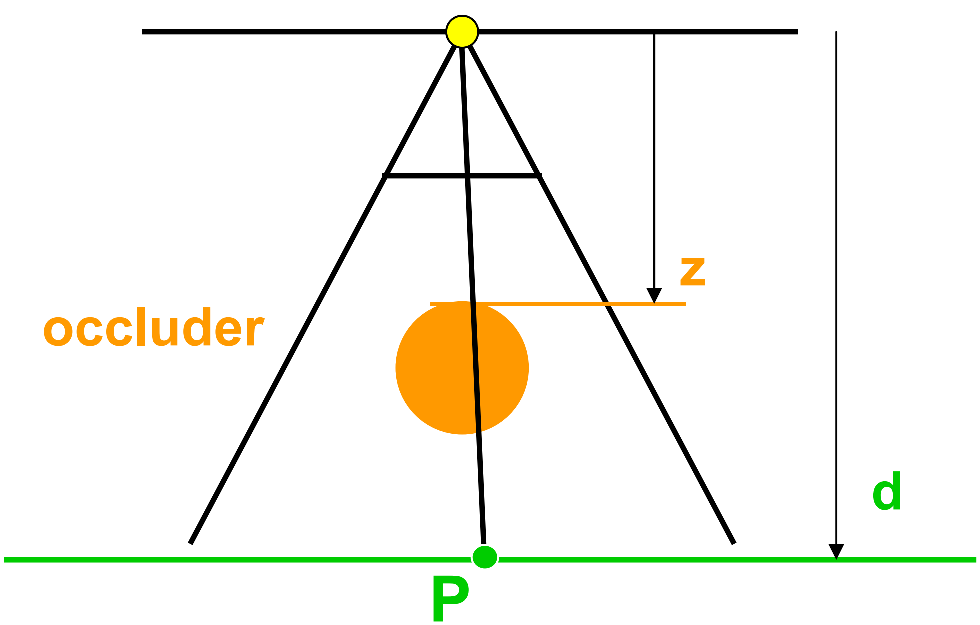
- Most popular technique for shadows in games
  - Purely image-based technique
  - Works with any rasterizable geometry
- Alternative techniques exist
  - Silhouette based techniques such as smoothies and penumbra wedges
    - Silhouette detection robustness issues
    - Do not work with alpha-tested geometry

Outline

- Fixed-Size Penumbra
    - PCF (Percentage Closer Filtering)
    - VSM (Variance Shadow Maps)
    - CSM (Convolution Shadow Maps)
    - ESM (Exponential Shadow Maps)
- Variable-Size Penumbra
    - PCSS (Percentage Closer Soft Shadows)
    - PCSS + VSM/CSM
    - Backprojection

Shadow Mapping

- Shadow map stores distance **z** to the light
- P is lit <=> (**d** < **z**)



Percentage Closer Filtering (PCF)

- Sample the result of (d<z) around projected point
  - Filter the binary results in a given kernel

0	1
1	0

- Bilinear PCF
  - NVIDIA and recent AMD GPUs implement 2x2 PCF in one fetch
  - Using same sample locations and weights as for bilinear filtering

Using Bilinear PCF with DX10

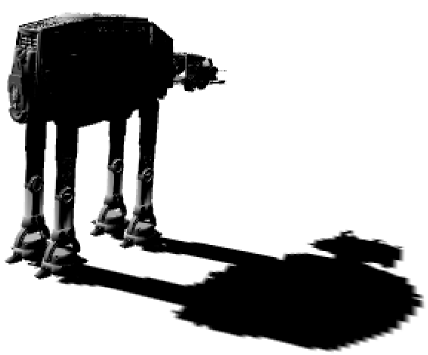
```
Texture2D<float> tDepthMap;

SamplerComparisonState ShadowSampler
{
    ComparisonFunc = LESS;
    Filter = COMPARISON_MIN_MAG_LINEAR_MIP_POINT;
};

    // …
    sum += tDepthMap.SampleCmpLevelZero(ShadowSampler,
                                         uv + offset, z);
```

PCF Filtering

- Increasing the number of PCF taps increases the softness of the shadows



1 tap

9x9 taps

17x17 taps

Irregular PCF

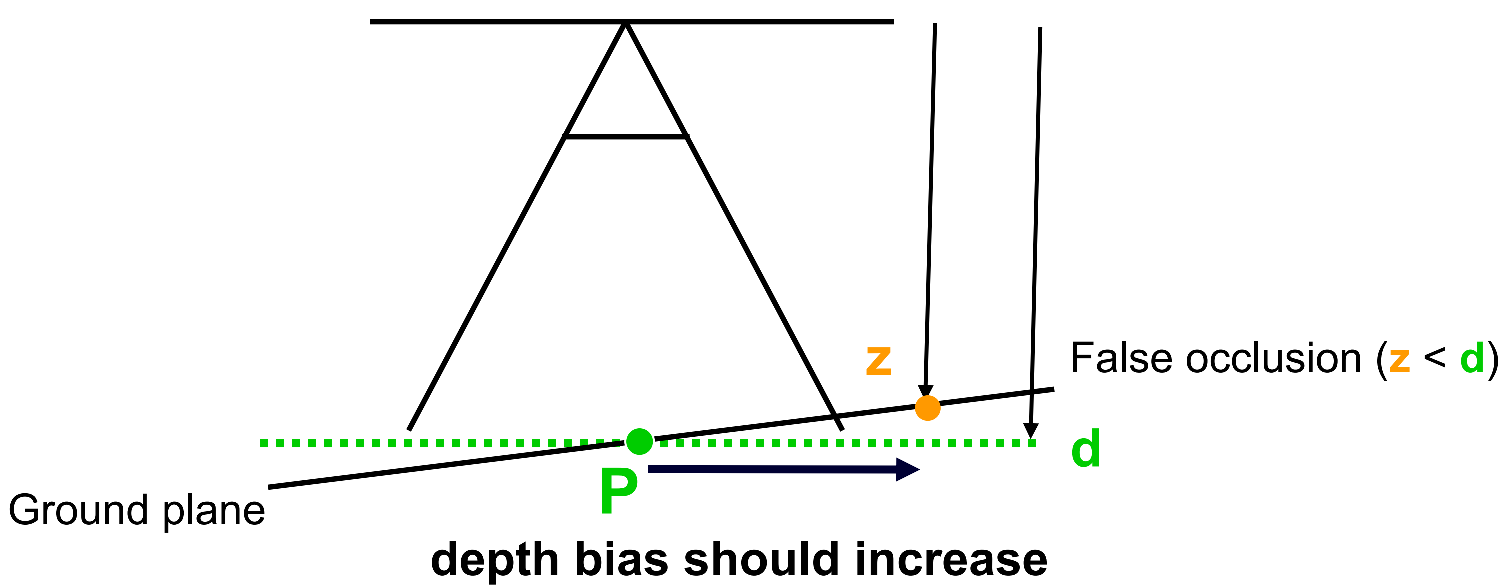
- PCF with large kernels requires many samples
- Using irregular sampling
  - Trades banding for noise

regular sampling

irregular sampling

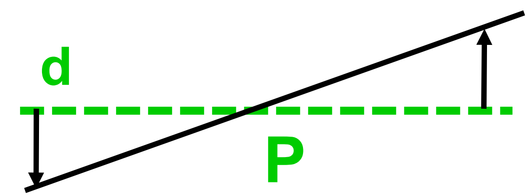
PCF Self-Shadowing Issue

- Traditional depth bias: write (z + bias) in shadow map
  - Bias = constant bias + slope-based bias
  - Issue: huge depth biases may be required for large PCF kernels

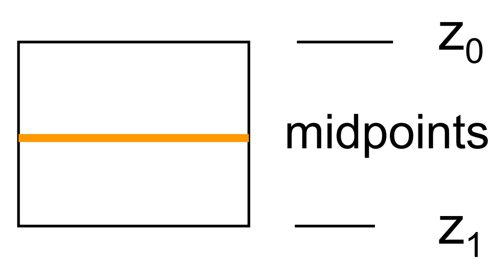


PCF Self-Shadowing Solutions

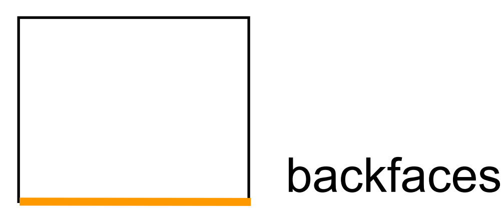
- Use depth gradient = float2(dz/du, dz/dv)
  - Make depth d follow tangent plane
  - d = d0 + dot(uv_offset, gradient)
  - [Schuler06] and [Isidoro06]



- Render midpoints into shadow map
  - Midpoint z = $(z_0 + z_1) / 2$
  - Requires two rasterization passes
    - Depth peel two depth layers
  - Still requires a depth bias for thin objects

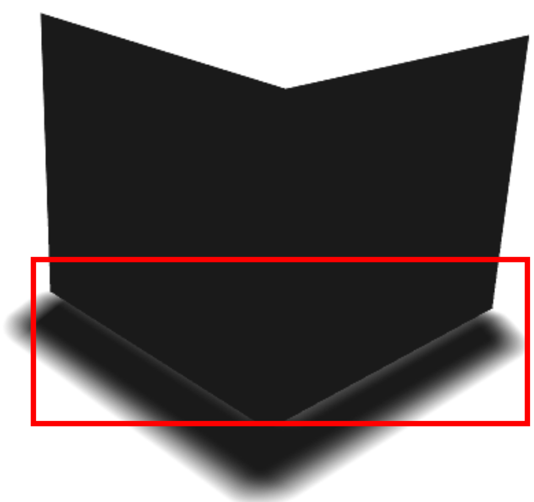


- Render backfaces into shadow map
  - Only works for closed objects
  - Light bleeding for large PCF kernels



Backfaces and Large PCF kernels

- Rendering backfaces into shadow map generates light bleeding for large PCF kernels
  - Not due to FP precision or shadow map resolution
  - But reverse of the surface acne issue



Non-Linearity of PCF

- PCF cannot be prefiltered as is
  - Average((d < z)) != (Average(z) < d)
  - Filtering the depth buffer would smooth the heightfield of the shadow map
    - Does not generate soft shadows
    - May introduce artifacts
- Solutions: Approximate shadow test by a linear function which can be prefiltered
  - Goal: blurring the shadow map to generate realistic soft shadows

Outline

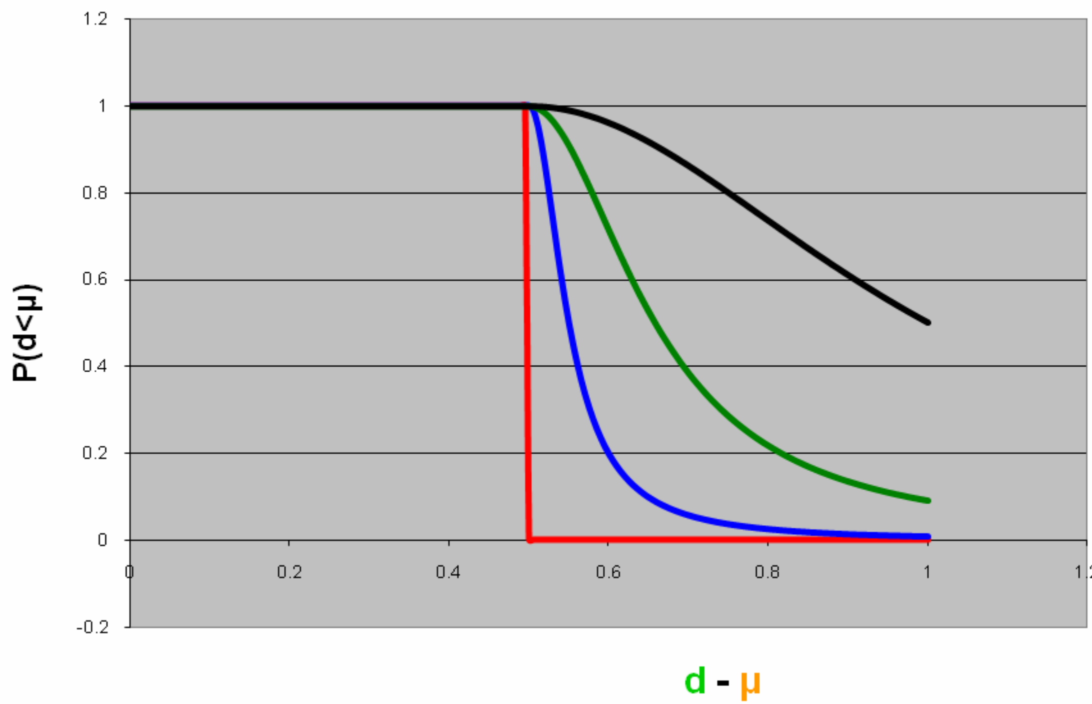
- Fixed-Size Penumbra
  - PCF (Percentage Closer Filtering)
  - VSM (Variance Shadow Maps)
  - CSM (Convolution Shadow Maps)
  - ESM (Exponential Shadow Maps)
- Variable-Size Penumbra
  - PCSS (Percentage Closer Soft Shadows)
  - PCSS + VSM/CSM
  - Backprojection

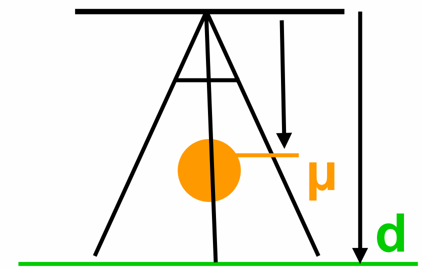
Variance Shadow Maps

- Consider depth values in the filter kernel as a depth distribution [Donnelly06] [Lauritzen07]
- Approximate the depth values in the kernel by a Gaussian distribution of mean μ and variance σ^2
  - $\sigma^2 = E(z^2) – E(z)^2$
  - $\mu = E(z)$
- Probability of being lit
  - Using Chebyshev's inequality
  - $P(d < z) <= \max(\sigma^2 / (\sigma^2 + (d – \boldsymbol{\mu})^2), (d < \mu))$
- E(X) = average of X in filter kernel
  - So the VSM (z, z^2) can be prefiltered
  - Can also be rendered using MSAA + resolve

VSM Visibility Function

$$P(d<\mu) = \max(\sigma^2 / (\sigma^2 + (d - \mu)^2), (d < \mu))$$





$\sigma^2 = E(z^2) - E(z)^2$
$\mu = E(z)$

- Self-shadowing can be handled simply by clamping σ^2 to some small MinVariance parameter

VSM Rendering

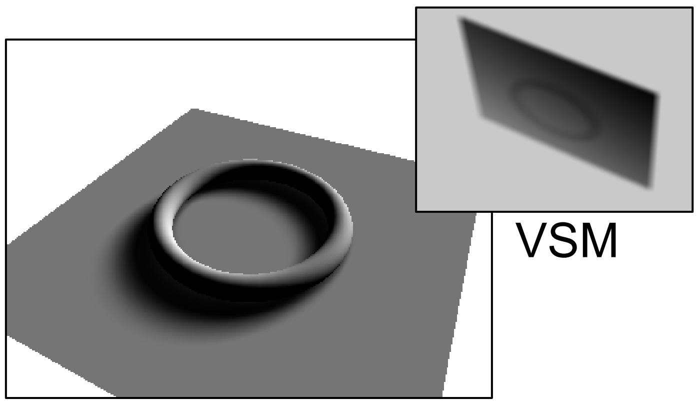
- Blur z and $z^2 \rightarrow E(z), E(z^2)$
- Fetch $E(z), E(z^2)$ with bilinear texture lookup
  - Can also use trilinear and/or anisotropic filtering to reduce aliasing
- Compute $\mu = E(z)$ and $\sigma^2 = E(z^2) - E(z)^2$
  - If $(\sigma^2 < MinVariance)\ \sigma^2 = MinVariance$
- Light intensity approximation
  - $L = \max(\sigma^2 / (\sigma^2 + (d - \boldsymbol{\mu})^2), (d < \mu))$

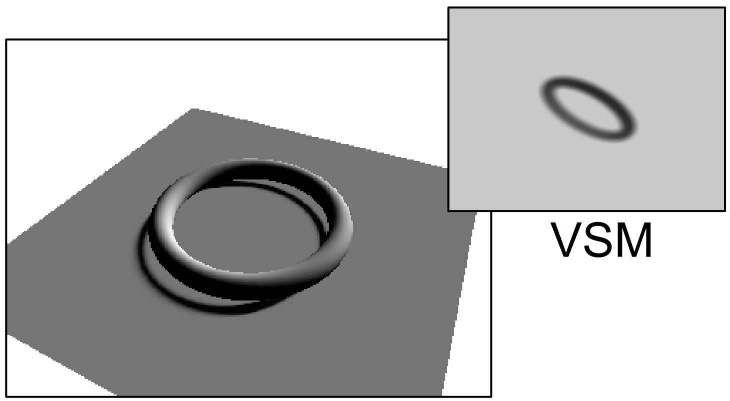
Generating the VSM

- Can use R32G32F_FLOAT format
  - Filterable format on all DX10-class graphic hardware
  - Can also use FP16 [Donnelly06]
- Render (z, z^2) to VSM texture
  - Where z is a linear distance
    - Can use an orthographic projection
    - z = mul(wPos, mObjectToLightOrtho).z
- Clear VSM to (MAX_Z, MAX_Z*MAX_Z)

Rendering everything into the VSM shadow map

- Images with and without plane rendered in VSM



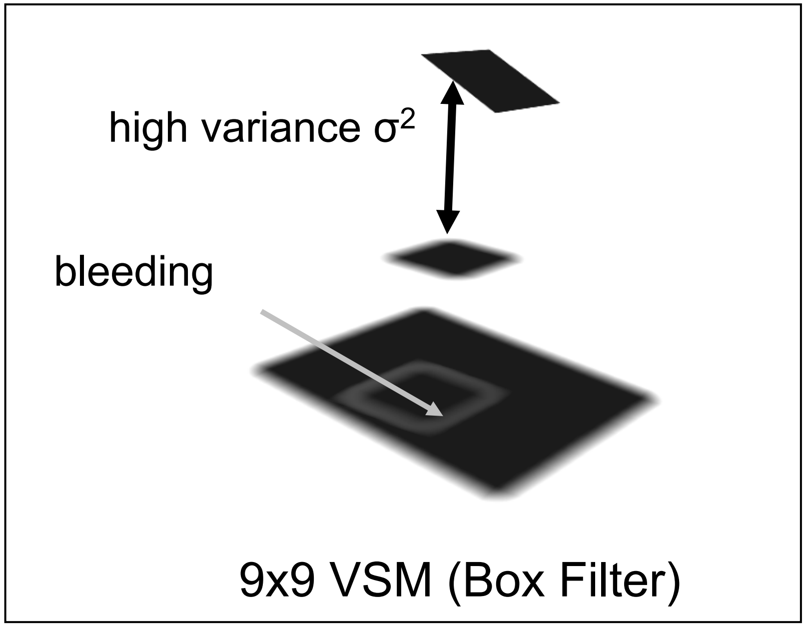


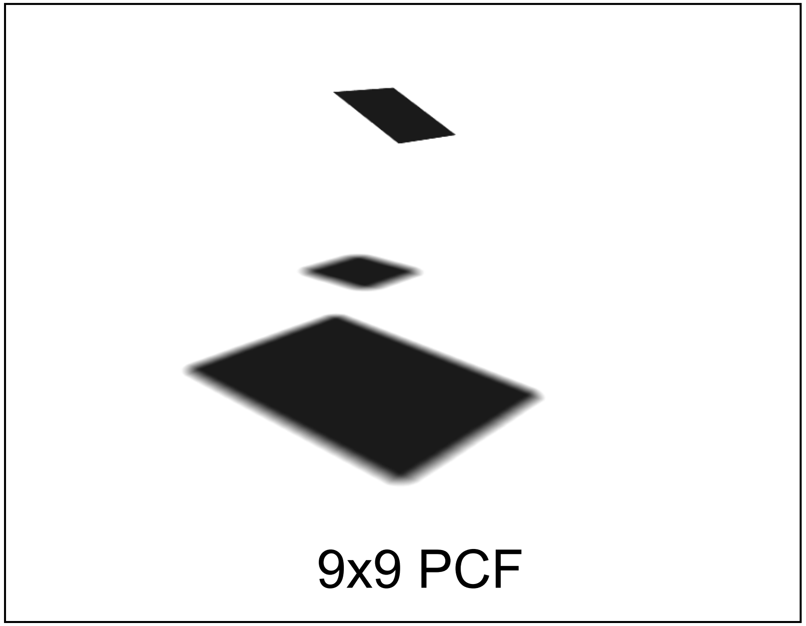
- If receivers are not in the VSM, will blur the object texels with z = MAX_Z ➔ missing occlusion

- CSMs and ESMs also have this limitation
  - Shadows look bad when blurring shadow map without everything rendered into them

VSM Light Bleeding

Two quads floating above a ground plane





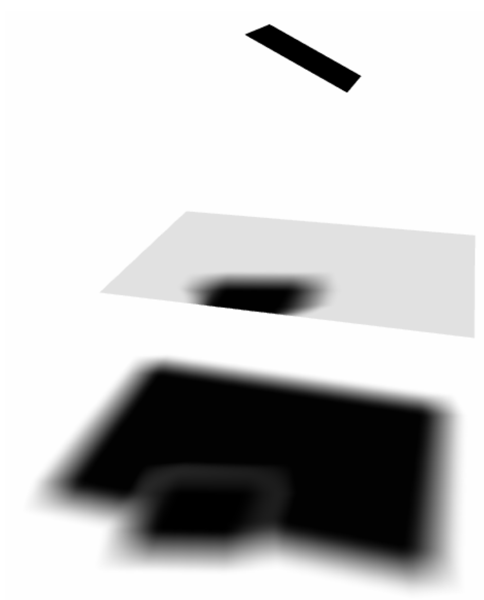
Large depth complexity in filter kernel

➔ σ^2 is large

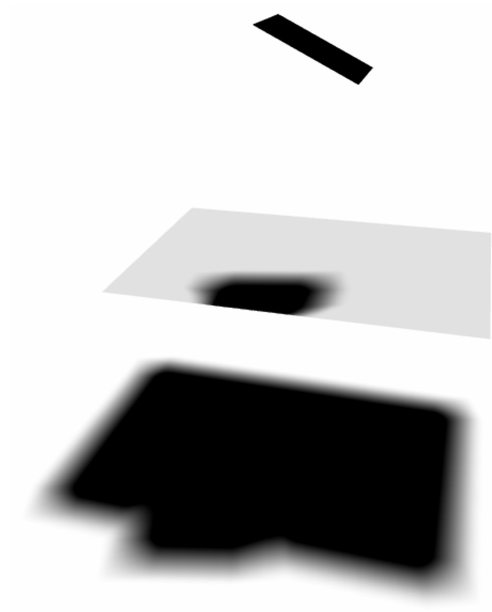
➔ Upper bound $\sigma^2 / (\sigma^2 + (d - \mu)^2) \rightarrow 1$

VSM Light Bleeding

- Proposed in GPU Gems 3 [Lauritzen07]
  - Use a threshold to remap shadow intensity



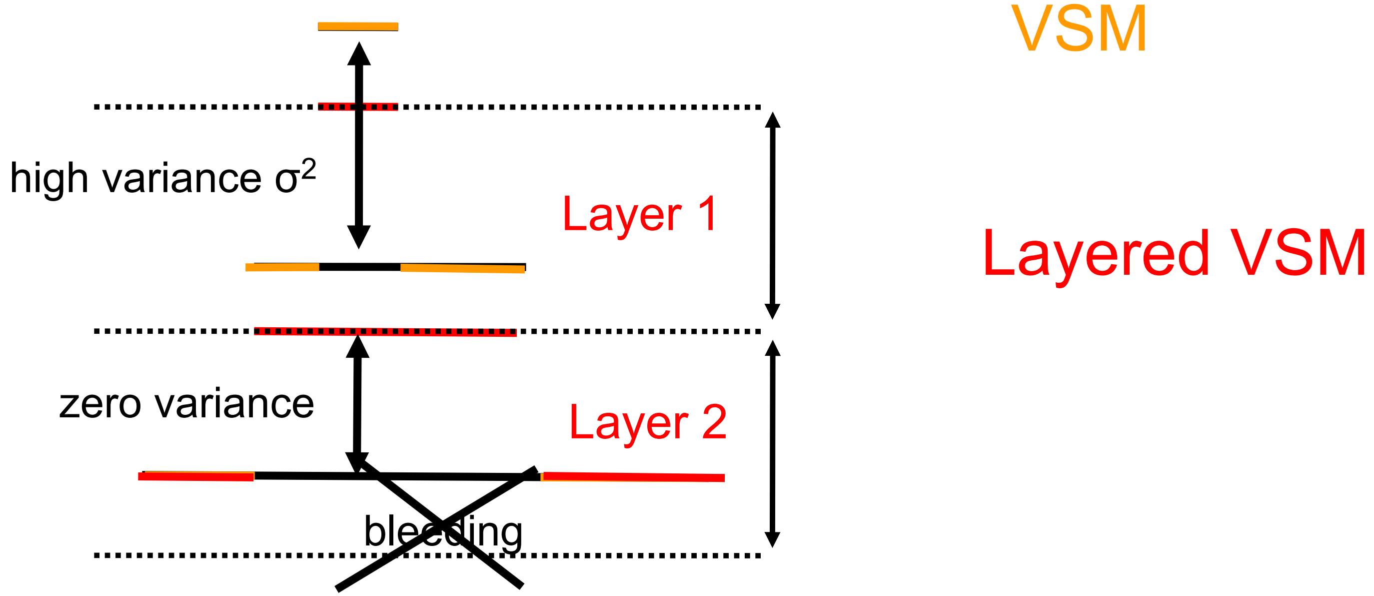
threshold=0



threshold=0.2

Layered VSM

- [Lauritzen08] I3D 2008
  - Partition shadow-map frustum into multiple depth ranges and clamp z to each range



Layered VSM

- Compared to VSMs, LVSMs
  - Reduce or eliminate light bleeding artifacts
  - Do not require 32-bit float texture filtering
  - Can still be generated in one pass (using MRTs) and need only one texture sample per pixel

Variance Shadow Map

Layered Variance Shadow Map

Outline

- Fixed-Size Penumbra
  - PCF (Percentage Closer Filtering)
  - VSM (Variance Shadow Maps)
  - CSM (Convolution Shadow Maps)
  - ESM (Exponential Shadow Maps)
- Variable-Size Penumbra
  - PCSS (Percentage Closer Soft Shadows)
  - PCSS + VSM/CSM
  - Backprojection

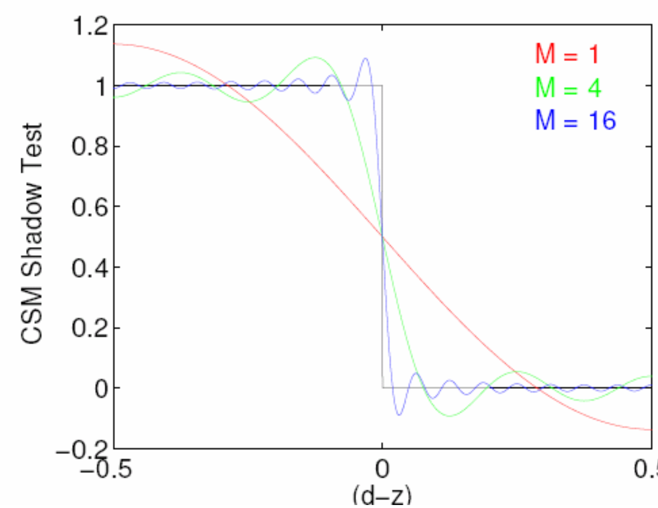
Convolution Shadow Maps

- Approximate the step function (function of depth) by 1D Fourier expansion [Annen07]

$$f(d,z) \approx \frac{1}{2} + 2\sum_{k=1}^{M} \frac{1}{c_k}\cos(c_k d)\sin(c_k z) - 2\sum_{k=1}^{M} \frac{1}{c_k}\sin(c_k d)\cos(c_k z)$$

Function of d
Reconstruct when rendering shadows

Function of z
Write in shadow map and blur it



Excerpted from [Annen07]

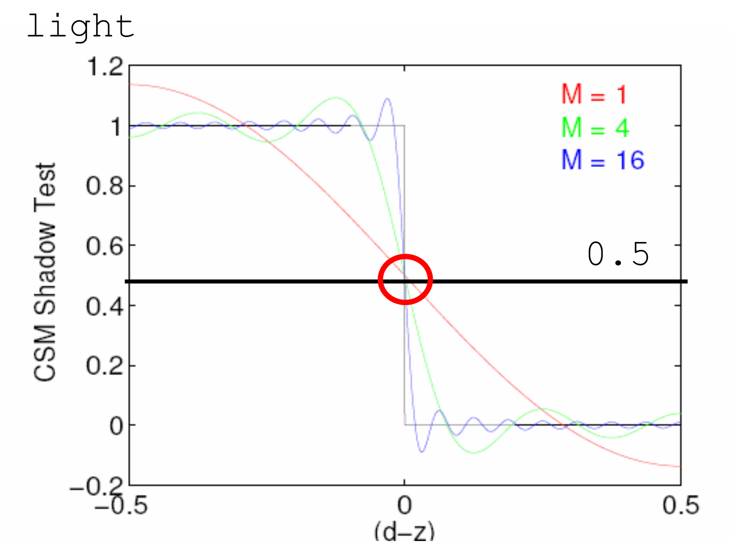
Generating a CSM

- Assuming a linear depth z is available for every texel in the shadow map
  - z needs to be normalized in [0,1] because the Fourier expansion is periodic
- Can generate CSM with full screen shader pass
  - Rendering to R8G8B8A8_UNORM MRTs
  - 4 MRTs (M=8) is usually a good tradeoff
- Background texels cleared to CSM(1.0f)

CSM Rendering

- Fetch cos(ck*z) and sin(ck*z) from the CSM textures and compute light intensity

$$f(d,z) \approx \frac{1}{2} + 2\sum_{k=1}^{M} \frac{1}{c_k} \cos(c_k d)\sin(c_k z) - 2\sum_{k=1}^{M} \frac{1}{c_k} \sin(c_k d)\cos(c_k z)$$



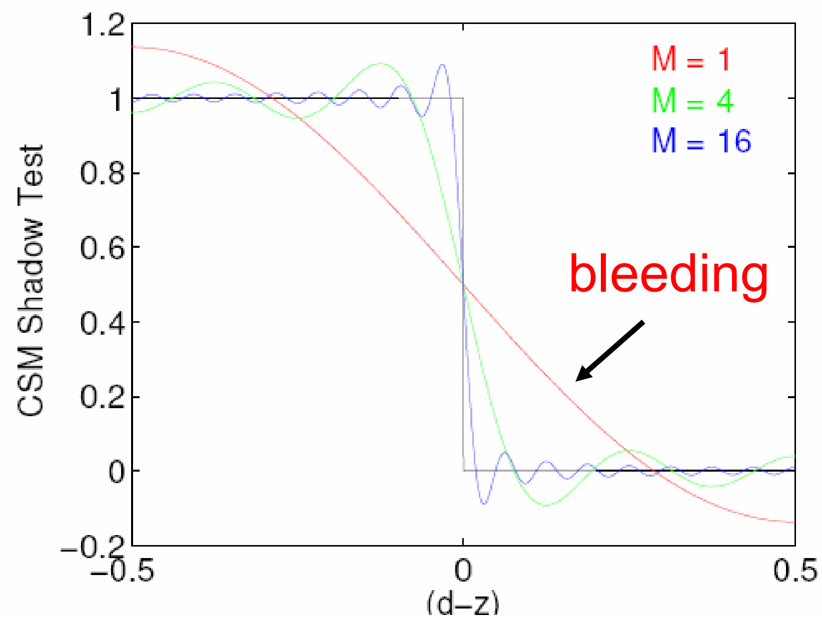
- Final light intensity

  Scale by 2 otherwise f(d,z) == 0.5 for (d==z)

  L = saturate(2.0 * f(d,z))

CSM Light Bleeding

- Limited bleeding may be seen as a feature (deep shadow look)
- Light bleeding when should be in shadow (d>z) but light > 0

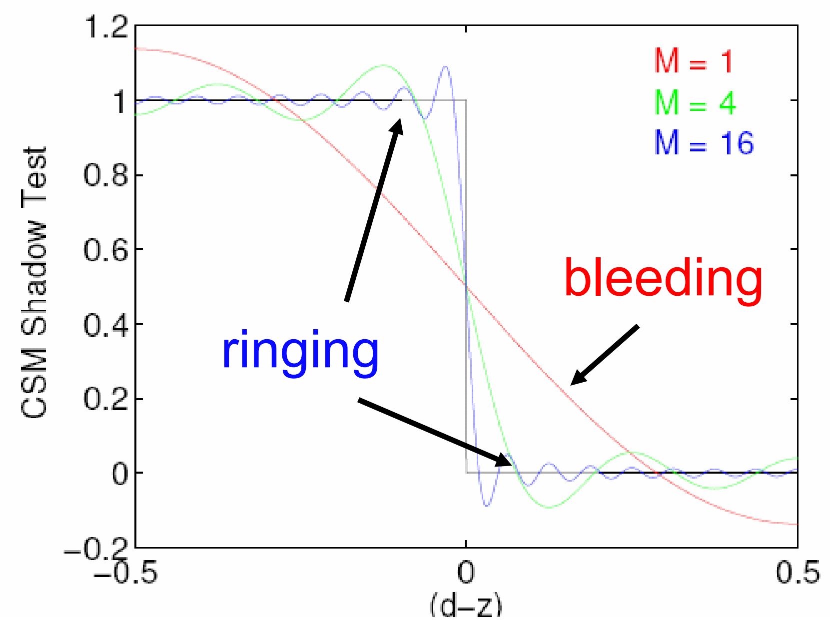


CSM Ringing

- Gibbs phenomenon
  - High order Fourier terms generate ringing
- Ringing workaround
  - Attenuate terms based on their order (k)

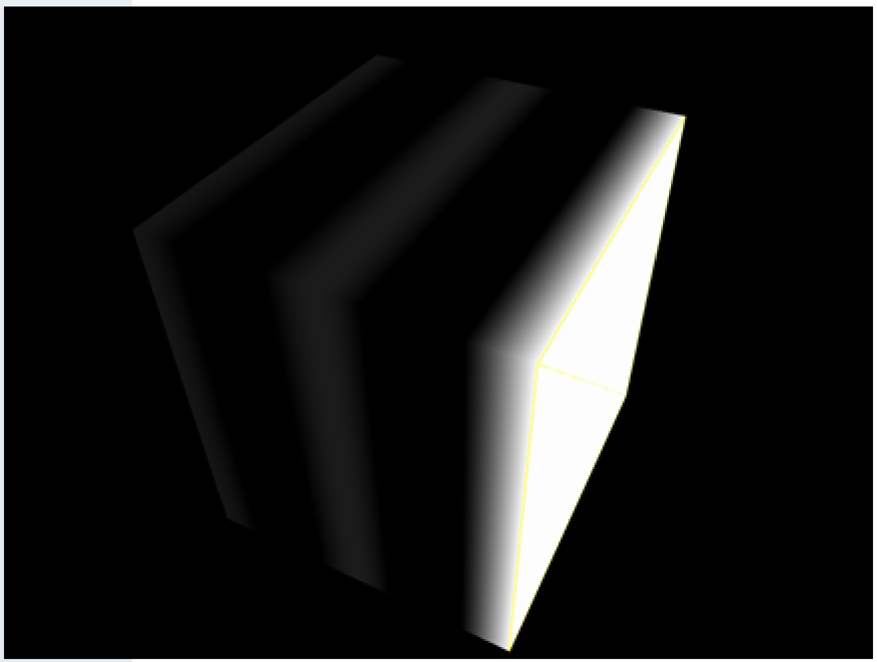
$$f(d,z) \approx \frac{1}{2} + 2\sum_{k=1}^{M} \boxed{\frac{1}{c_k}\cos(c_k d)\sin(c_k z)} - 2\sum_{k=1}^{M} \boxed{\frac{1}{c_k}\sin(c_k d)\cos(c_k z)}$$

$* \ \exp(-\alpha(k/M)^2)$

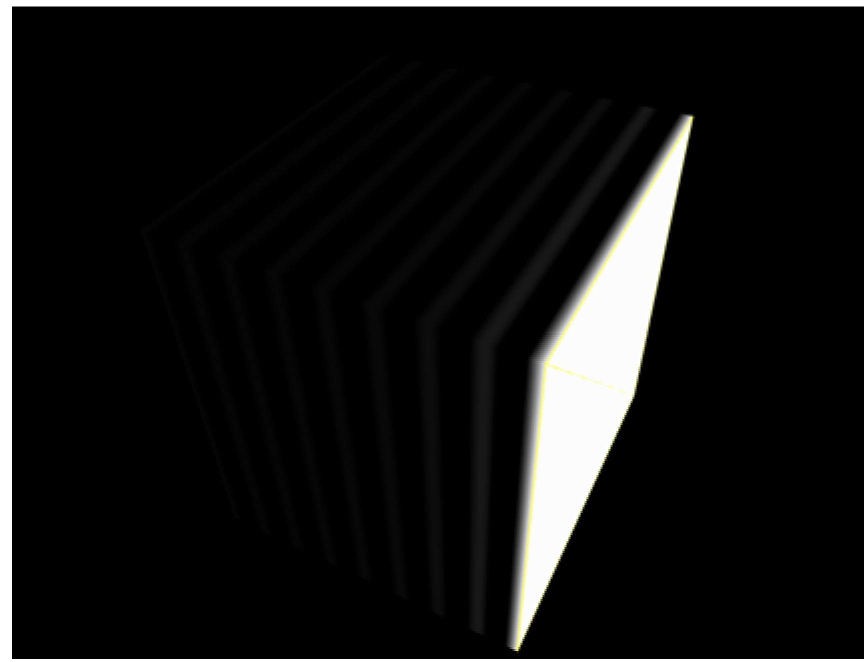


CSM Ringing

- Light source facing inwards the cube



2 RGBA8

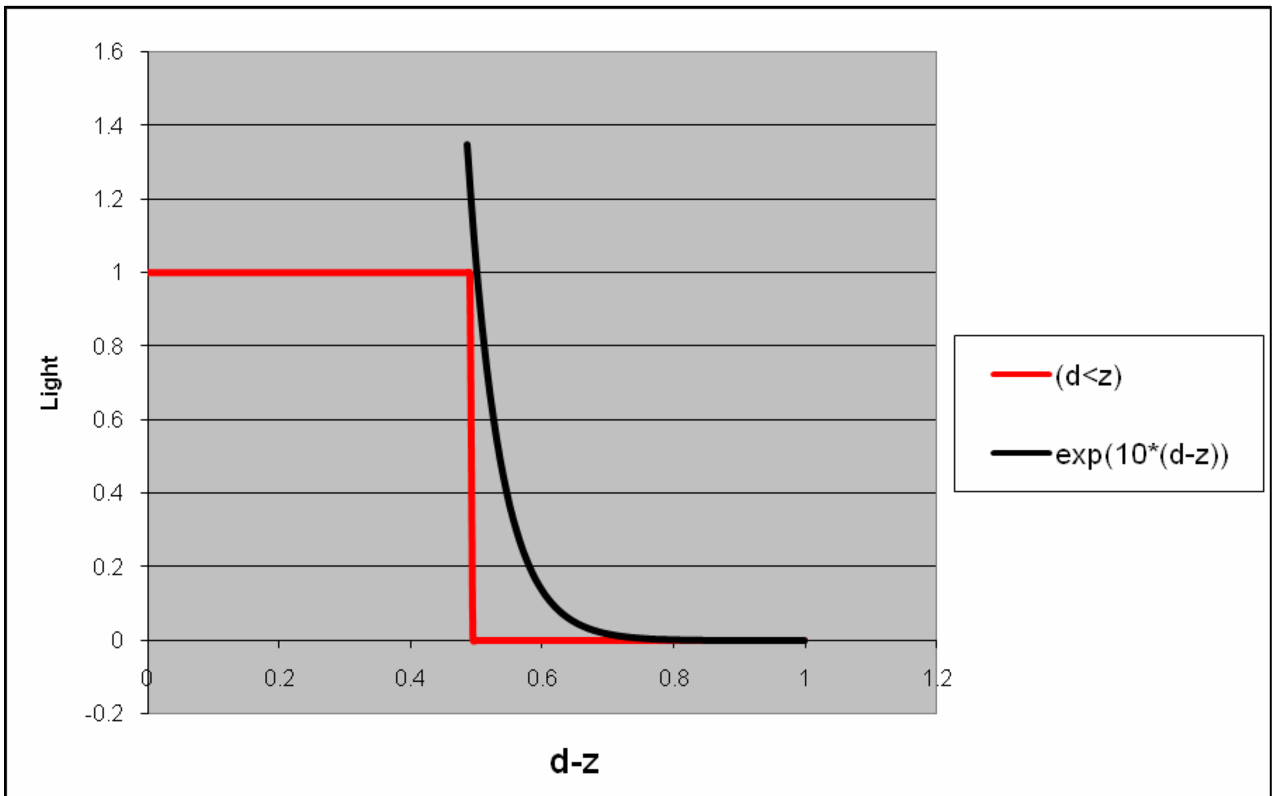


8 RGBA8

Exponential Shadow Maps

- [Salvi08] ShaderX[6]
- Approximate step function (z-d > 0) by

  exp(k*(z-d)) = exp(k*z) * exp(-k*d)



ESM Parameter Tuning

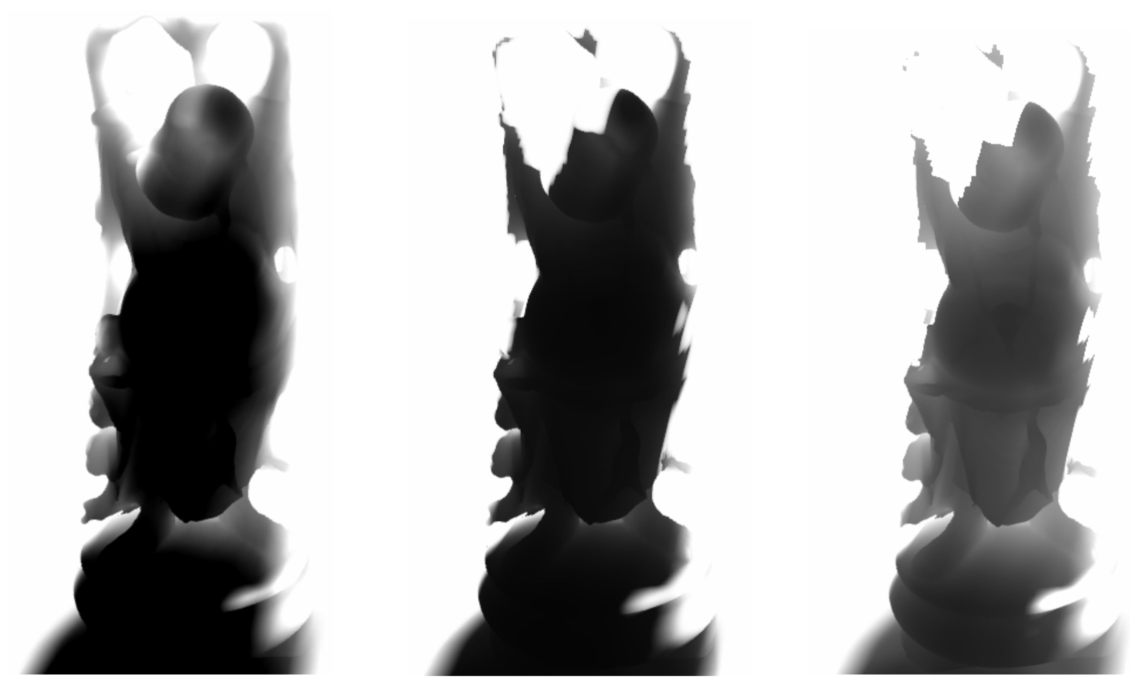
$$L = \exp(k*z) * \exp(-k*d)$$

k=10

k=30

Deep Shadow Look

- Shadows without any shading
  - Soft shadow test gives translucent look



CSM VSM ESM

VSM/CSM/ESM

- Parameters
  - VSM: MinVariance, BleedingReductionFactor
  - CSM: NumTextures, AbsorptionFactor
  - ESM: ScaleFactor
- Storage

VSM	CSM	ESM
R32G32	N * (R8G8B8A8)	R32

For CSM, N>=4 is usually required to avoid bleeding

- The less storage, the faster the prefiltering

VSM/CSM/ESM

VSM – 87 fps

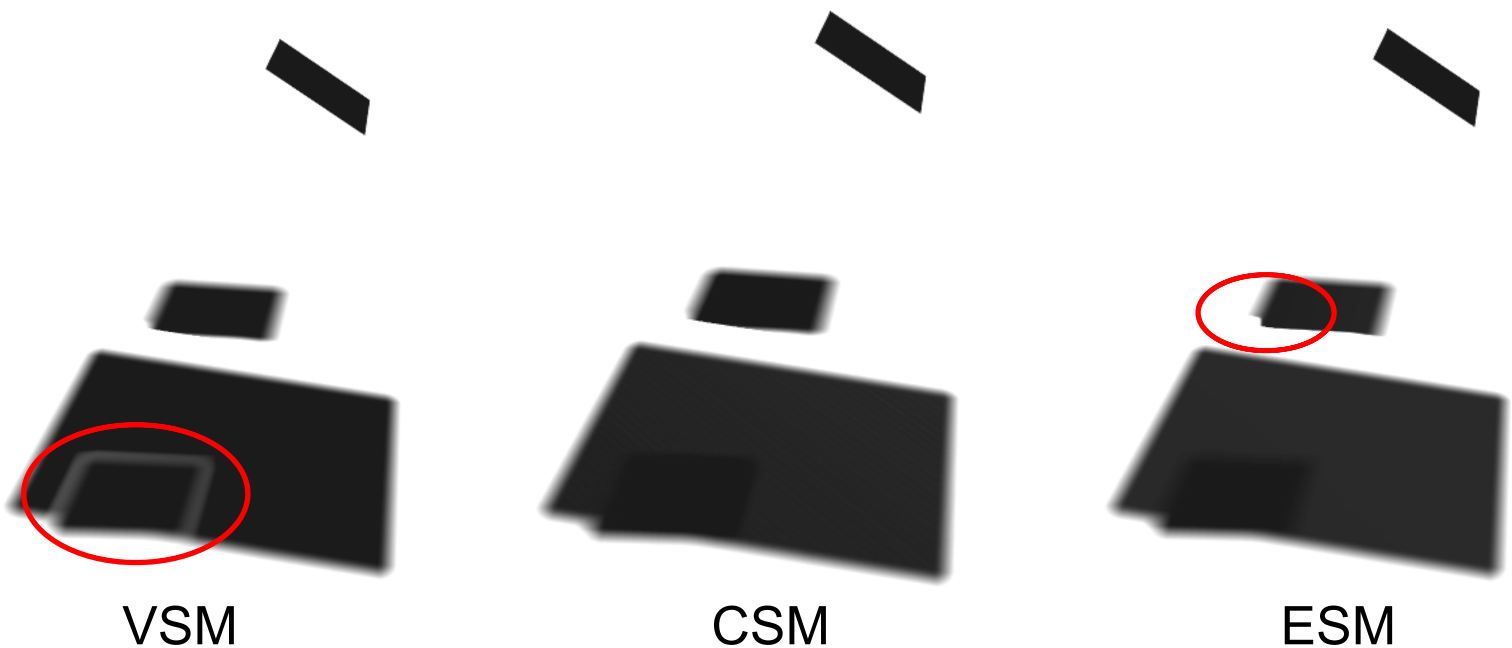
CSM – 38 fps

ESM – 93 fps

8800 GTX Ultra

VSM/CSM/ESM

- 2 quads floating above a ground plane



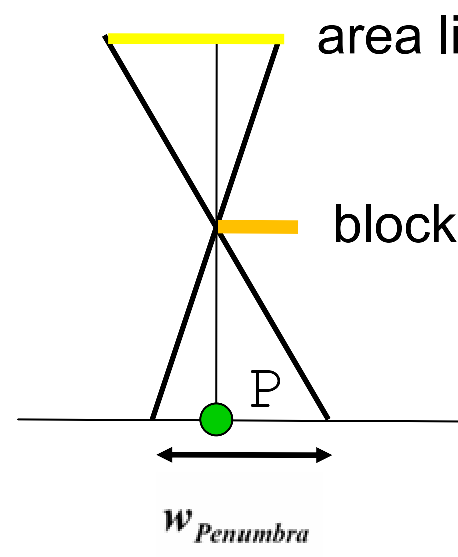
- The minor ESM artifacts can be handled with another parameter which overdarkens the shadows [Salvi08]

Outline

- Fixed-Size Penumbra
  - PCF (Percentage Closer Filtering)
  - VSM (Variance Shadow Maps)
  - CSM (Convolution Shadow Maps)
  - ESM (Exponential Shadow Maps)
- Variable-Size Penumbra
  - PCSS (Percentage Closer Soft Shadows)
  - PCSS + VSM/CSM
  - Backprojection

Percentage Closer Soft Shadows

- PCSS [Fernando05]
- Assume a square light centered at the shadow map center
- Assuming some parallel blocker to receiver
  - Compute penumbra width using similar triangles



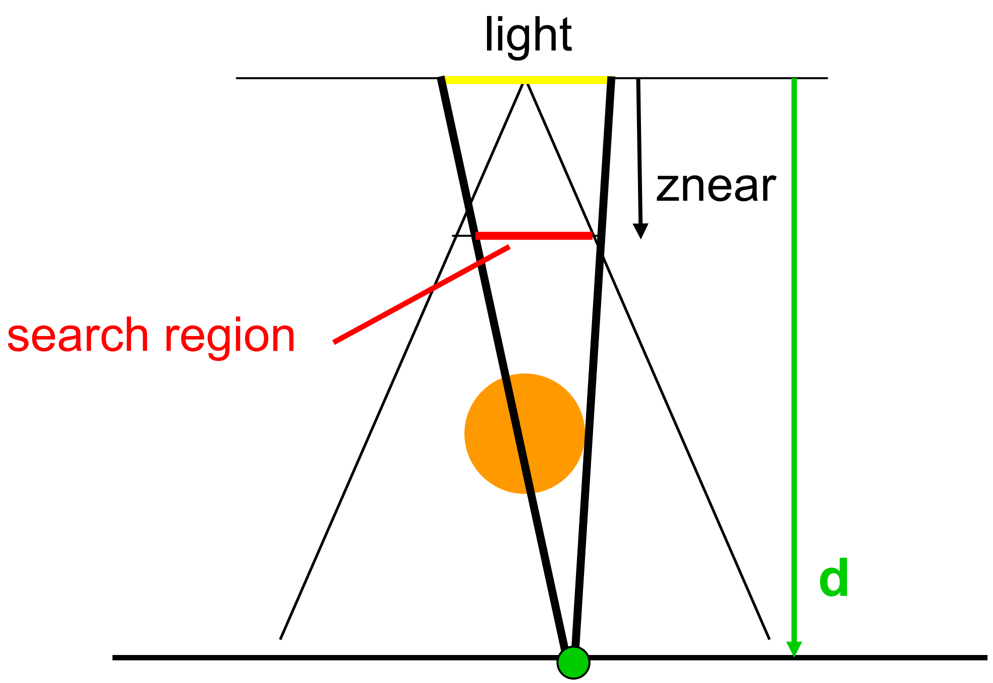
$$w_{Penumbra} = \frac{(d_{Receiver} - d_{Blocker}) \cdot w_{Light}}{d_{Blocker}}$$

PCSS Overview

- Step 1: Blocker Search
  - Sample the depth buffer using point sampling
  - Average all blockers with (depth + bias < receiver) in search region / kernel
  - Early out if no blocker found
- Step 2: Filtering
  - Use filter radius from step 1
  - Clamp filter width to be >= MinRadius for antialiasing
  - Filter the shadow map with PCF or VSM/CSM/ESM

Search Region

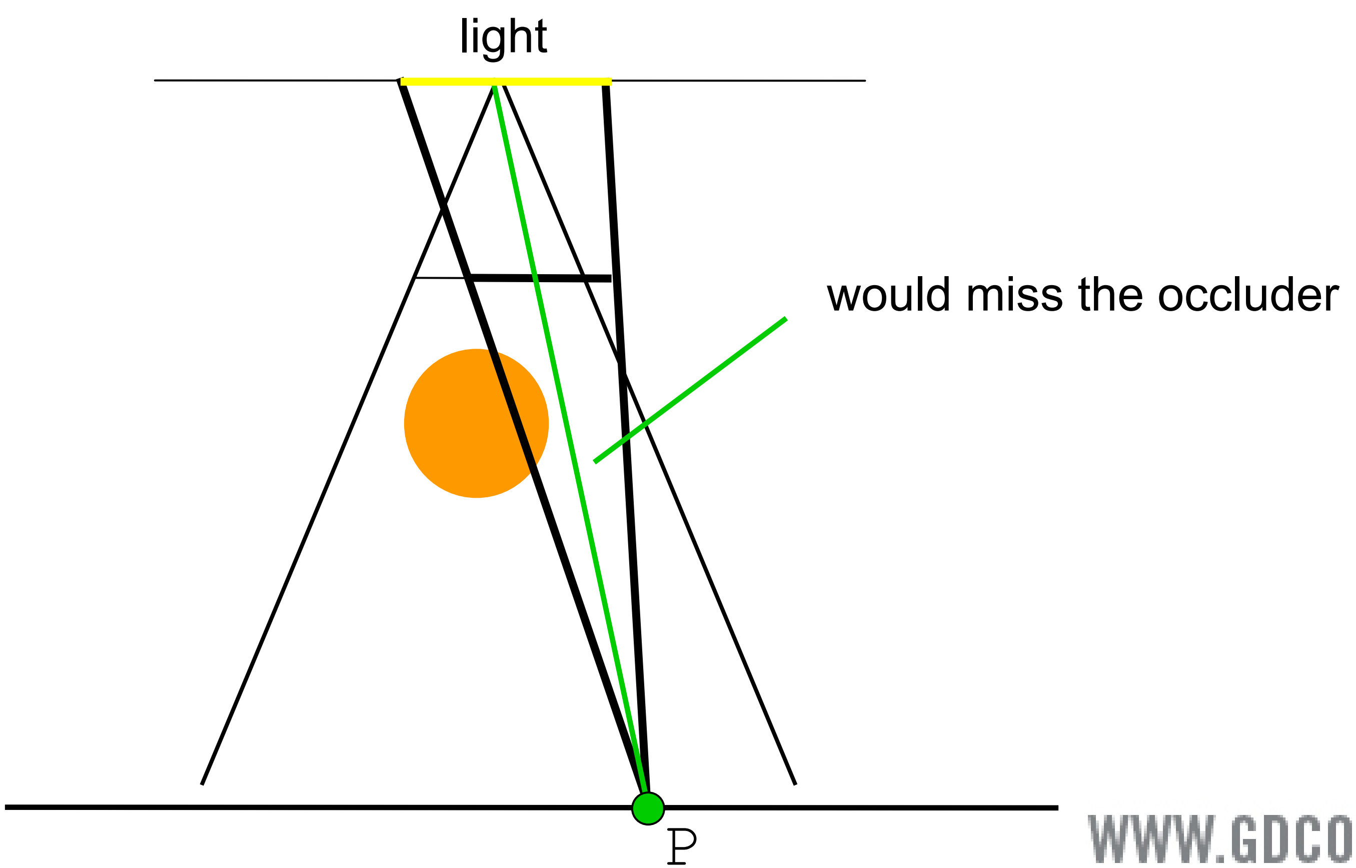
- Where to find blockers?
- Conservative search radius using similar triangles



LightRadius / d = SearchRadius / (d-znear)

Blocker Search

- Why not doing just one sample?



Blocker Search

- The more samples in the blocker search, the less noisy artifacts in the soft shadows
  - In practice, 4x4 or 5x5 samples is sufficient

Blocker Search with 3x3 taps

Blocker Search with 5x5 taps

Hellgate_DX10_vs_DX9_Large.jpg...

PCSS In Hellgate: London

PCSS

PCF

16 POINT taps for the blocker search

16 PCF taps for the PCF filtering

Outline

- Fixed-Size Penumbra
  - PCF (Percentage Closer Filtering)
  - VSM (Variance Shadow Maps)
  - CSM (Convolution Shadow Maps)
  - ESM (Exponential Shadow Maps)
- Variable-Size Penumbra
  - PCSS (Percentage Closer Soft Shadows)
  - PCSS + VSM/CSM
  - Backprojection

PCSS + VSM/CSM/ESM

- Compute filter size using blocker search
- Solutions to prefilter the shadow map to support fast blurs of variable size
  - Summed Area Tables (SAT)
    - Fast Box Filters
  - Mipmapping
    - Level[i] = Blur(Level[i-1])

Summed Area Tables

- For a 1D texture, each texel S[n] of the SAT is
  - S[n] = Sum(T[i], i=0..n)
- Average in interval (i,j] is
  - (S[j]-S[i]) / (j-i)
- Also works in 2D
  - With more 4 fetches instead of 2
- Fast algorithm for building SATs on GPUs [Hensley05]
  - Recursive doubling
  - For NxN shadow map, log(N) passes

Summed Area Shadow Maps

- Using UINT32
  - Advantages: more bits available than FP32, and support arbitrary shadow map resolution
  - Drawback: need to implement bilinear filtering in shader
  - See GPU Gems 3 chapter for details [Lauritzen07]
- Best fitting formats for DX10 GPUs
  - For SAVSM - R32G32_UINT
  - For SACSM - multiple R32G32B32A32_FLOATs
  - For ESM (storing exp(k*z)) - R32_UINT

SAVSM vs MIPCSM

- Disadvantage of SAT: can only do box filter
- Mipmap with recursive blurs looks more like a Gaussian

SACSM

MIPCSM

SATs with variable size kernel

- Rounding the filter radius generates banding

    float2 moments = FilterMomentsBilinear(uv, **round(filterRadius)**);

- Better quality: trilinear filtering

    float2 moments0 = FilterMomentsBilinear(uv, **floor(filterRadius)**);

    float2 moments1 = FilterMomentsBilinear(uv, **ceil(filterRadius)**);

    float2 moments = lerp(moments0, moments1, **frac(filterRadius)**);

SATs for VSM/CSM/ESM

- Cost of building a SAT (on G80 Ultra)
  - SAVSM
    - 0.9 ms for 512^2, 3.3 ms for 1024^2
  - SACSM with 16 FLOATs per texel
    - 6.3 ms for 512^2 and 27.5 ms for 1024^2
  - SAESM
    - 0.6 ms for 512^2, 2.2 ms for 1024^2

- CSM is not well suited for SATs
  - Mipmaps are a better fit for them

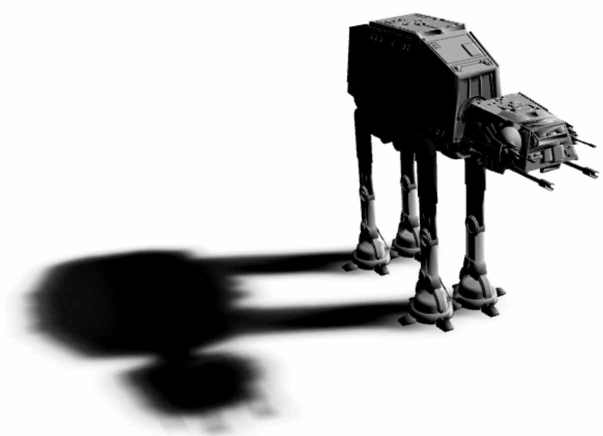
MIPCSM

- Build a mipmap pyramid on top of CSM
  - Implemented as texture array with mip levels
  - How to generate the mipchain
    - Default 2x2 mipchain generates aliasing
    - Use a larger box filter (at least 7x7) to generate each LOD, double spacing every level
    - Lookup CSM using trilinear filtering
- Advantages of MIPCSM over SACSM
  - Can use RGBA8 instead of RGBA32F
  - Much faster to build than SACSM

MIPCSM

- To reduce issues to discretized blurs, do a small blur pre-pass over the CSM

without pre-blur

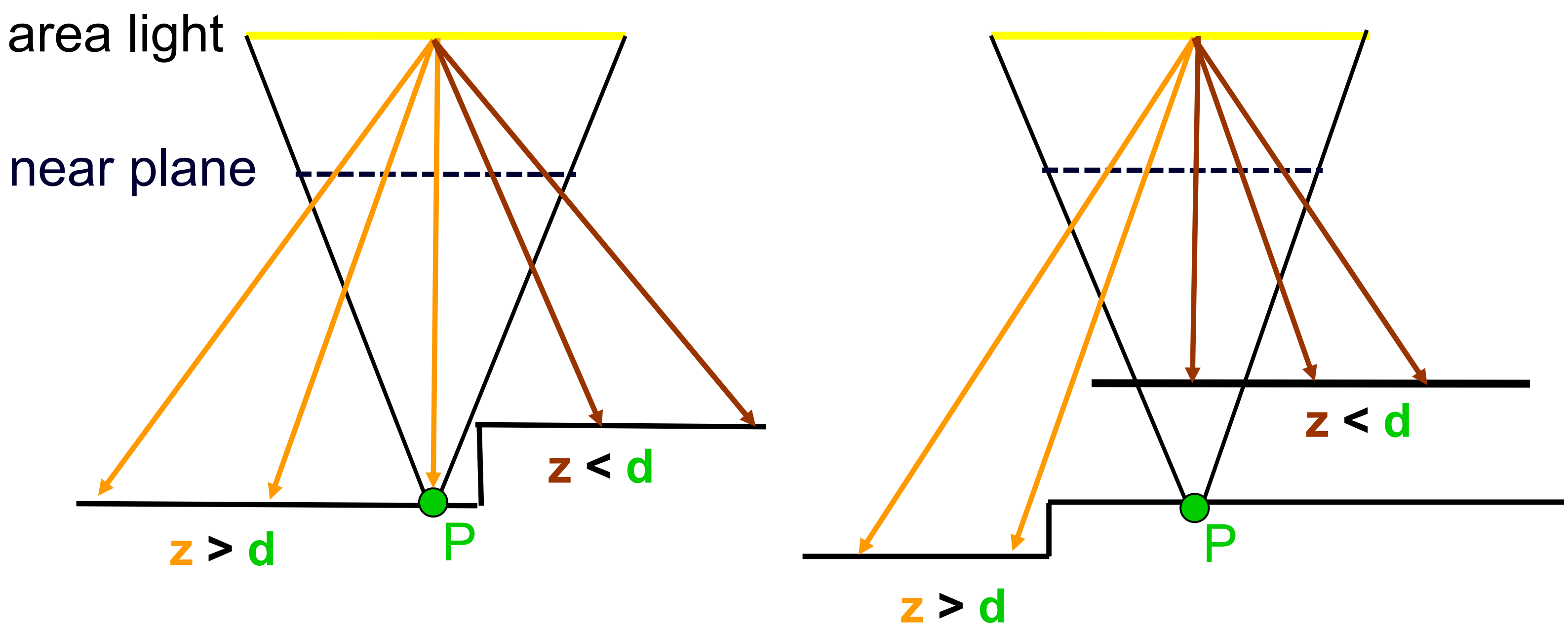


with 5x5 pre-blur

Outline

- Fixed-Size Penumbra
  - PCF (Percentage Closer Filtering)
  - VSM (Variance Shadow Maps)
  - CSM (Convolution Shadow Maps)
  - ESM (Exponential Shadow Maps)
- Variable-Size Penumbra
  - PCSS (Percentage Closer Soft Shadows)
  - PCSS + VSM/CSM
  - Backprojection

Why PCF is wrong for soft shadows

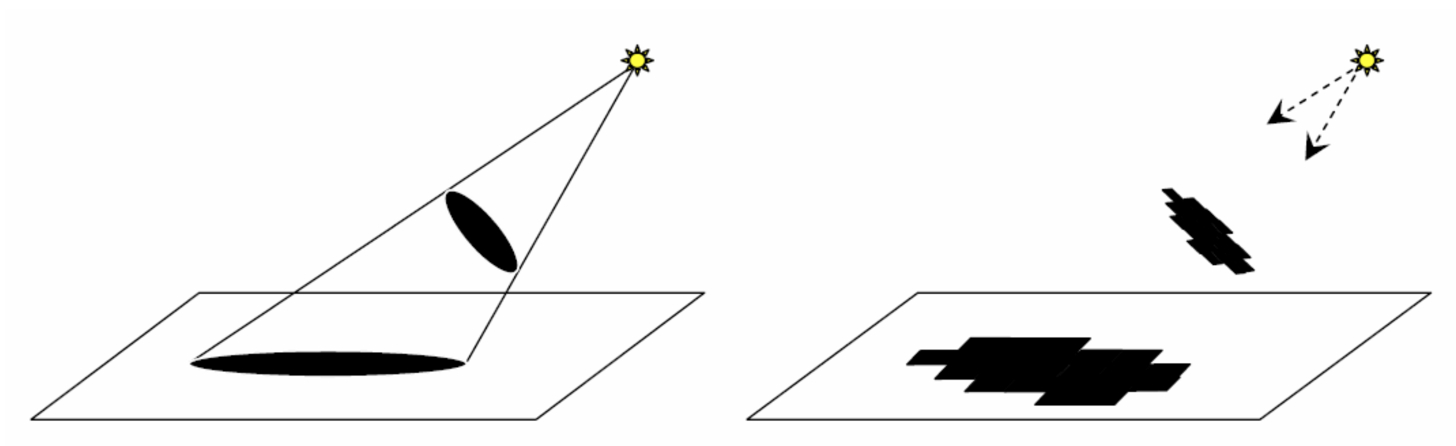


PCF: Shadow(P) > 0
Ground Truth: Shadow(P) == 0

PCF: Shadow(P) < 1
Ground Truth: Shadow(P) == 1

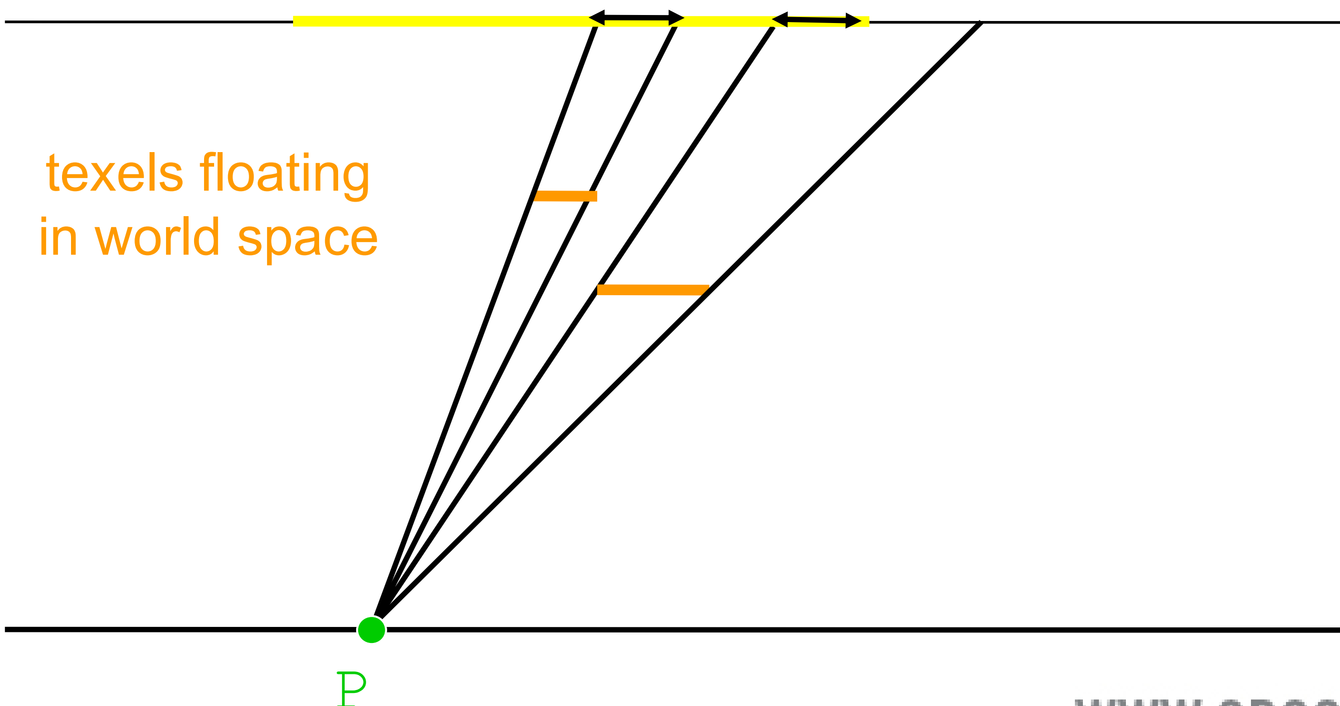
Unprojecting the shadow map

- Unproject shadow map texels into world space [Atty05] [Guennebaud06]
- Discretized approximation of the scene geometry



Backprojecting texels

- Backproject texels onto light source
- Clamp to light borders and accumulate area

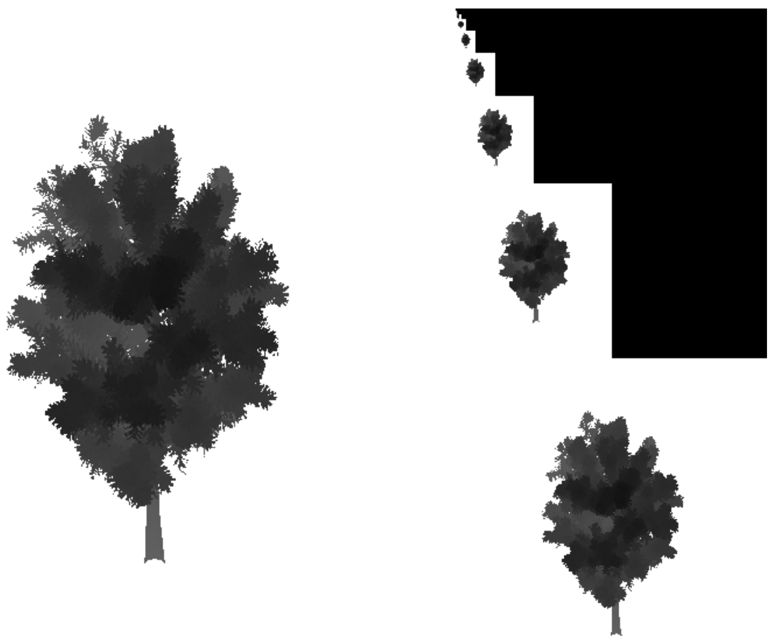


Min-Max Mipmap Shadow Map

- Can represent shadow-map data in hierarchical fashion [Dmitriev07]
  - Hierarchical traversal allows for efficient pruning of subtrees
- Build a hierarchical shadow map first
  - R32G32_FLOAT mipmap
  - Each level stores min and max z of 2x2 subpixels
- Links
  - SDK sample
  - GDC 2007 talk

Flattening the mipmap

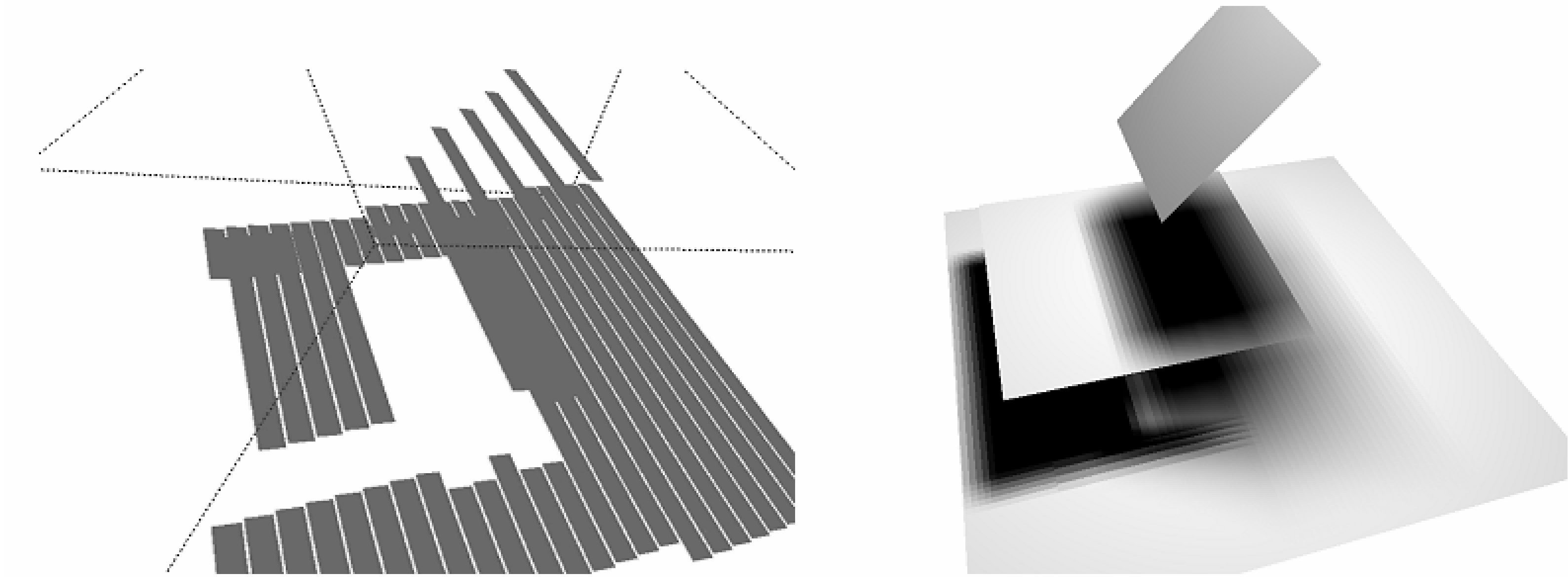
- Generate mipmap and flattened 2D texture
  - Load() from texture without mipmaps is faster because it guarantees that all pixels access the same mipmap level.



Flattened mipmap

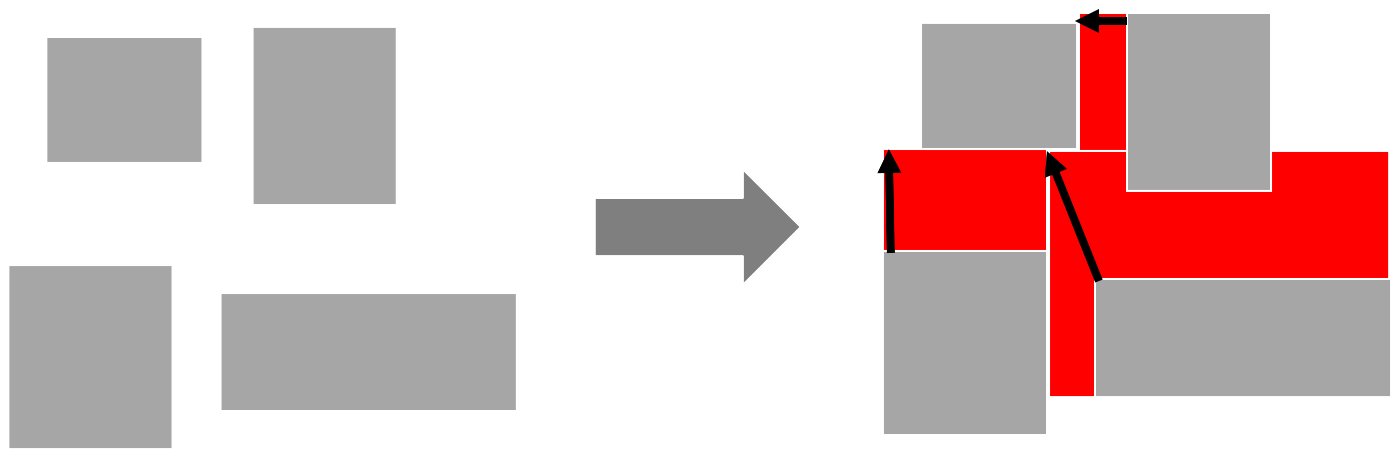
Light Bleeding

- Light bleeding is caused by gaps between micropatches



Gap Filling

- Extend backprojected quads on the sides to touch their left and top neighbors and the diagonal
  - Similar to [Guennebaud06]



backprojected texels

filling the gaps

PCSS with PCF

Frame Rate: 120 fps (8.3 ms)
Shadow Map Generation: 1.4 ms

Blocker Search: 5x5 taps
PCF: 5x5 taps

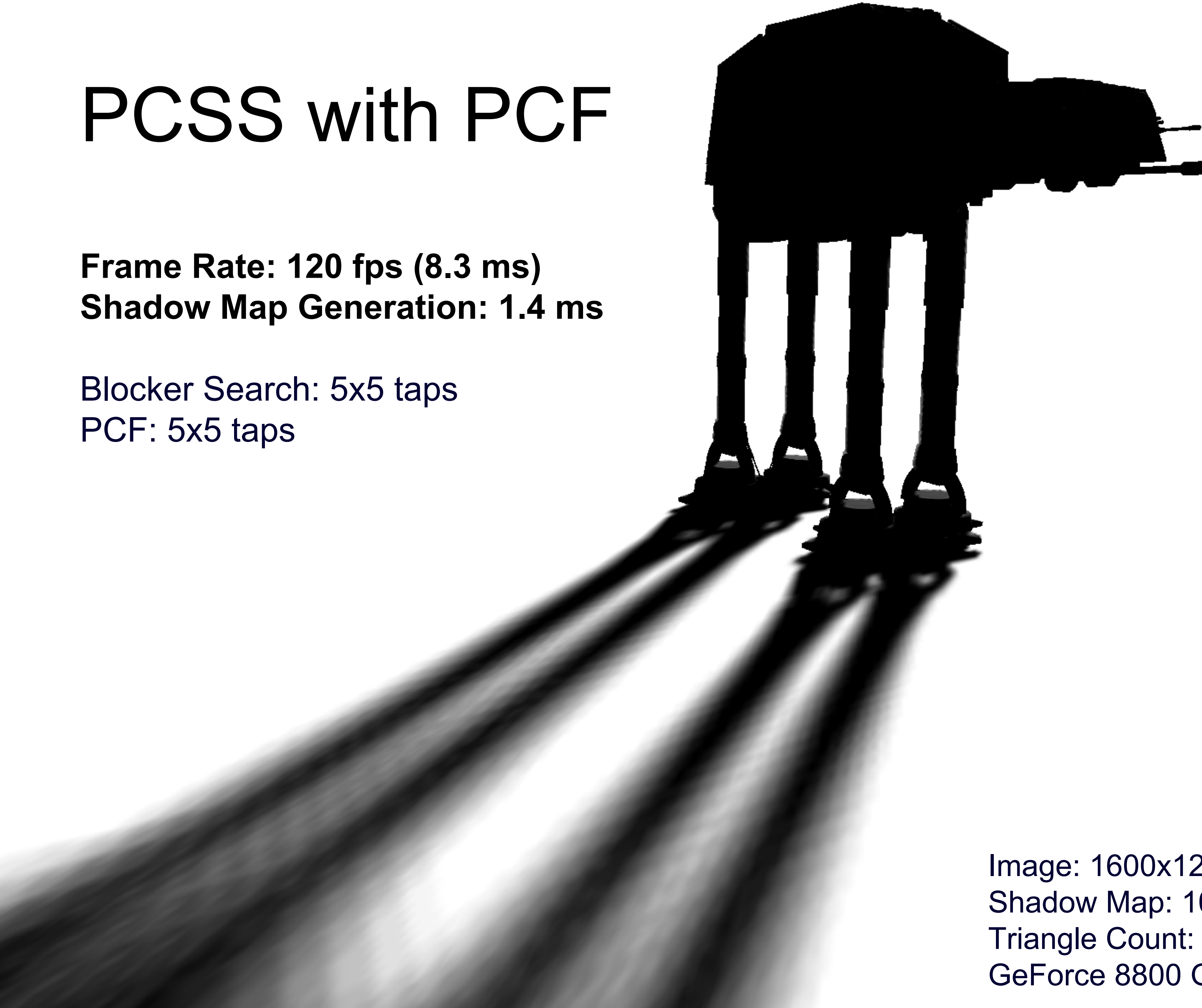


Image: 1600x1200
Shadow Map: 1024^2
Triangle Count: 211k
GeForce 8800 GT

PCSS with PCF

Frame Rate: 96 fps (10.4 ms)
Shadow Map Generation: 1.4 ms

Blocker Search: 5x5 taps
PCF: 9x9 taps

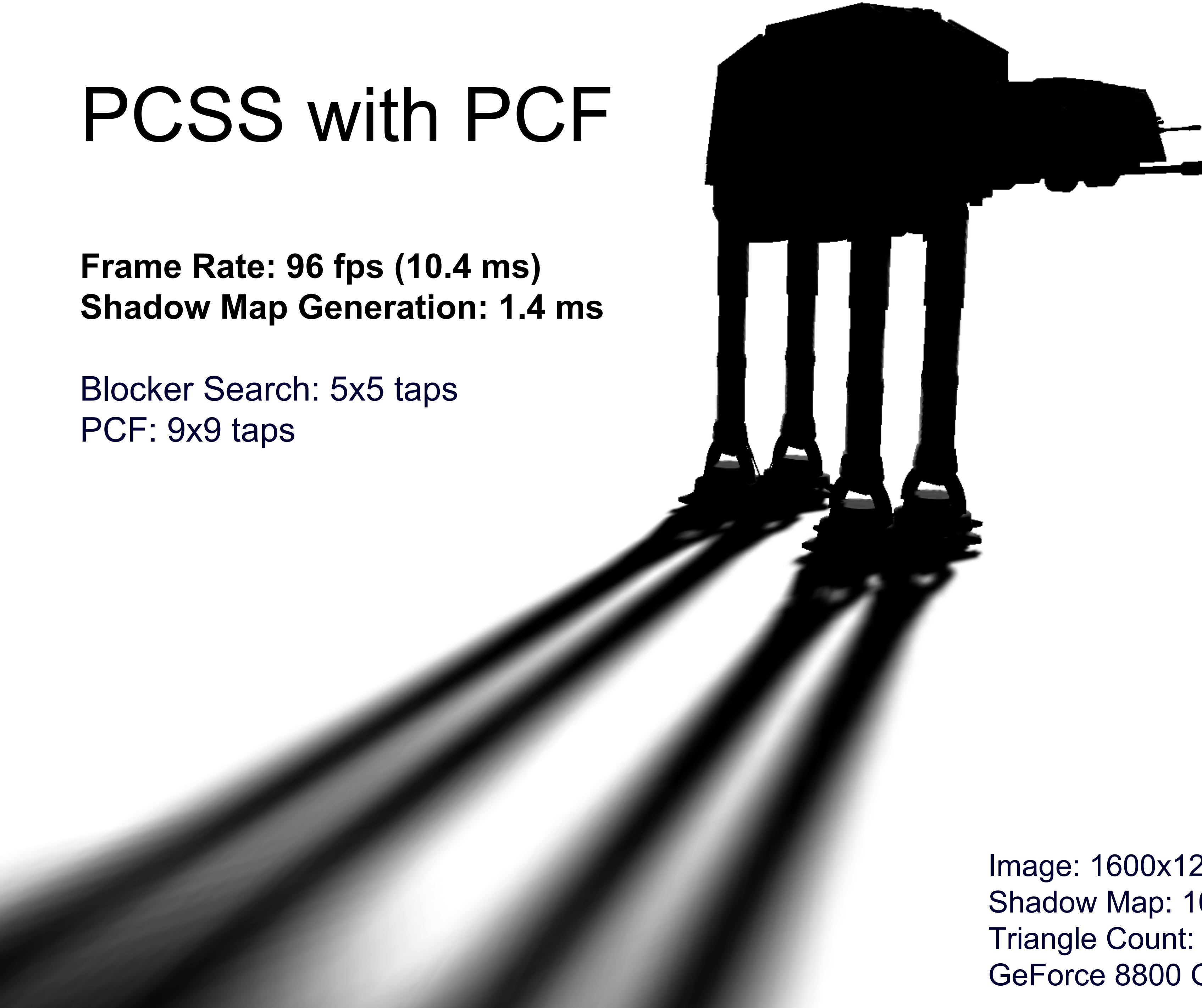


Image: 1600x1200
Shadow Map: 1024^2
Triangle Count: 211k
GeForce 8800 GT

PCSS with PCF

Frame Rate: 54 fps (18.5 ms)
Shadow Map Generation: 1.4 ms

Blocker Search: 5x5 taps
PCF: 17x17 taps

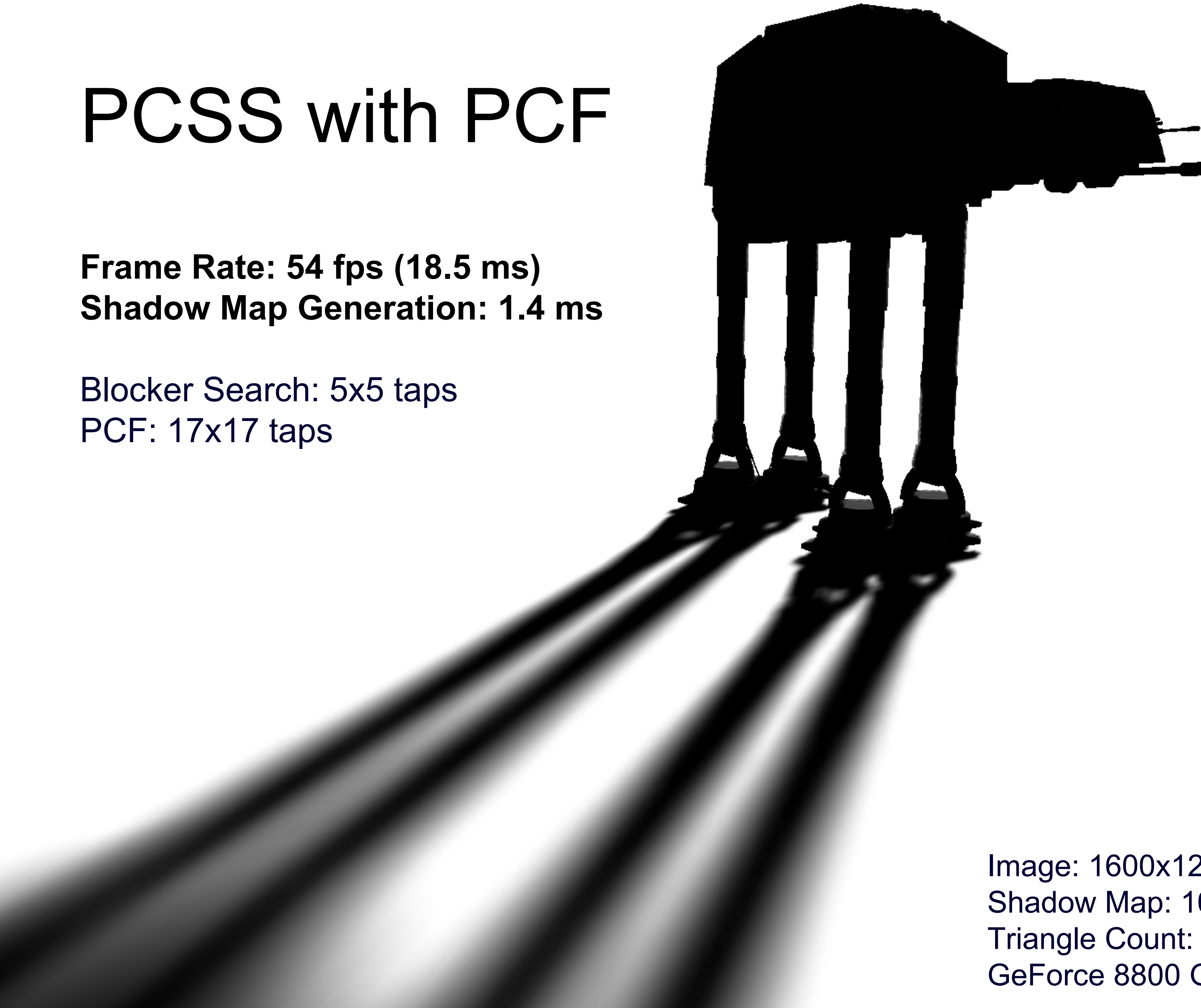


Image: 1600x1200
Shadow Map: 1024^2
Triangle Count: 211k
GeForce 8800 GT

PCSS + SAVSM

Frame Rate: 68 fps (14.7 ms)
SAVSM: 6.5 ms total (44%)

Blocker Search: 5x5 taps
UINT32 SAVSM

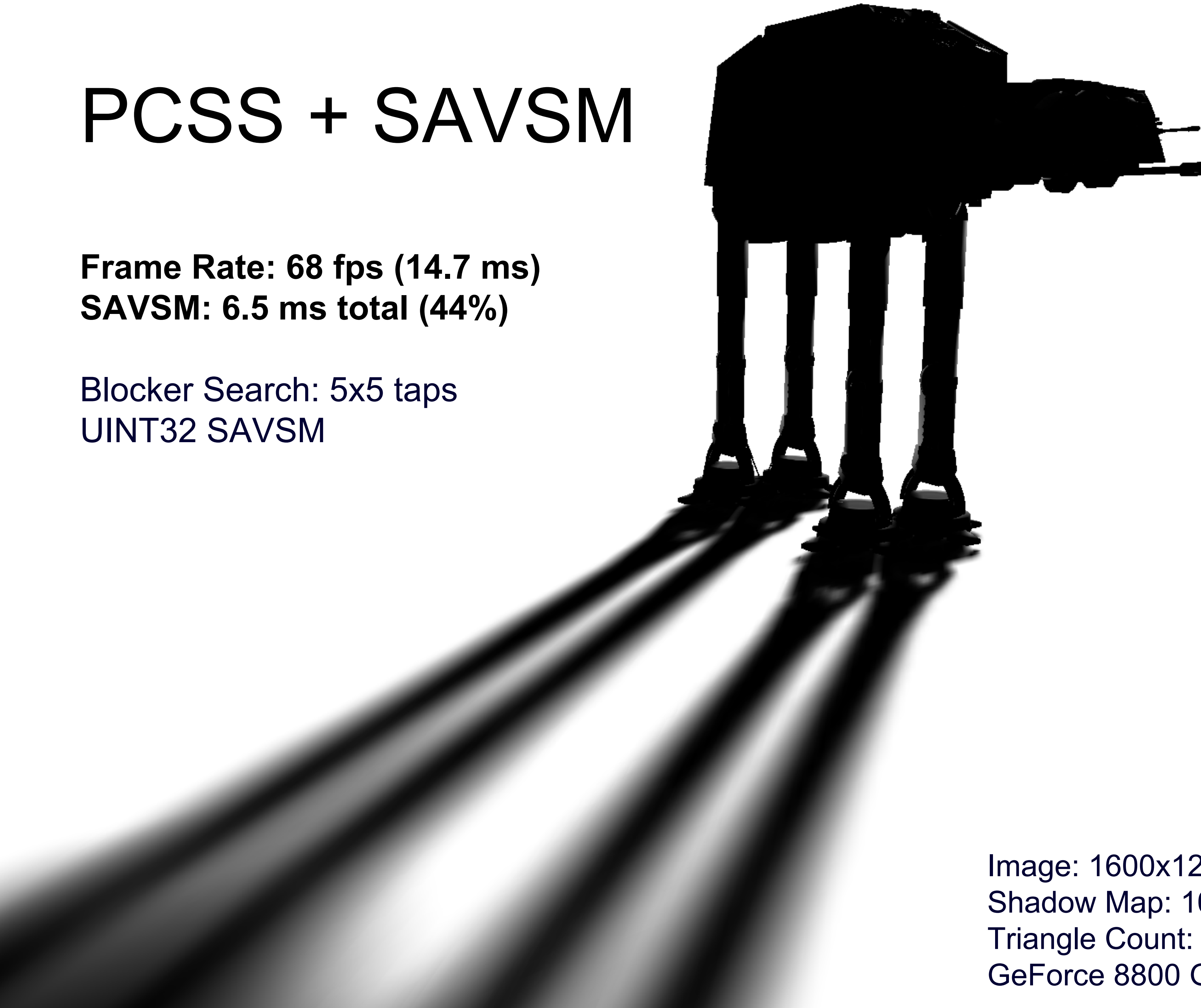


Image: 1600x1200
Shadow Map: 1024^2
Triangle Count: 211k
GeForce 8800 GT

PCSS + MIPCSM

Frame Rate: 57 fps (17.5 ms)
MIPCSM: 9.4 ms total (54%)

Blocker Search: 5x5 taps
MIPCSM: 4 RGBA8

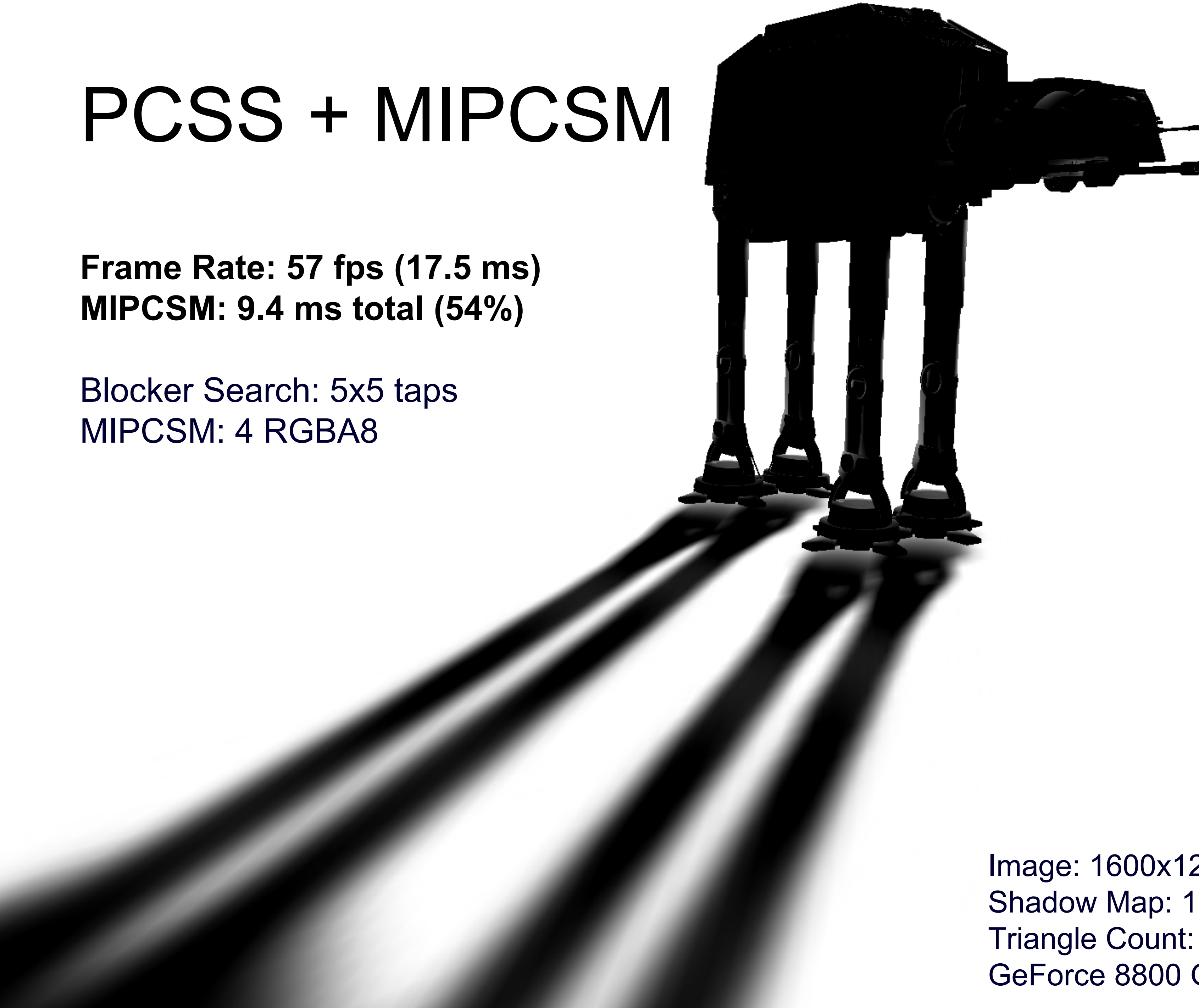


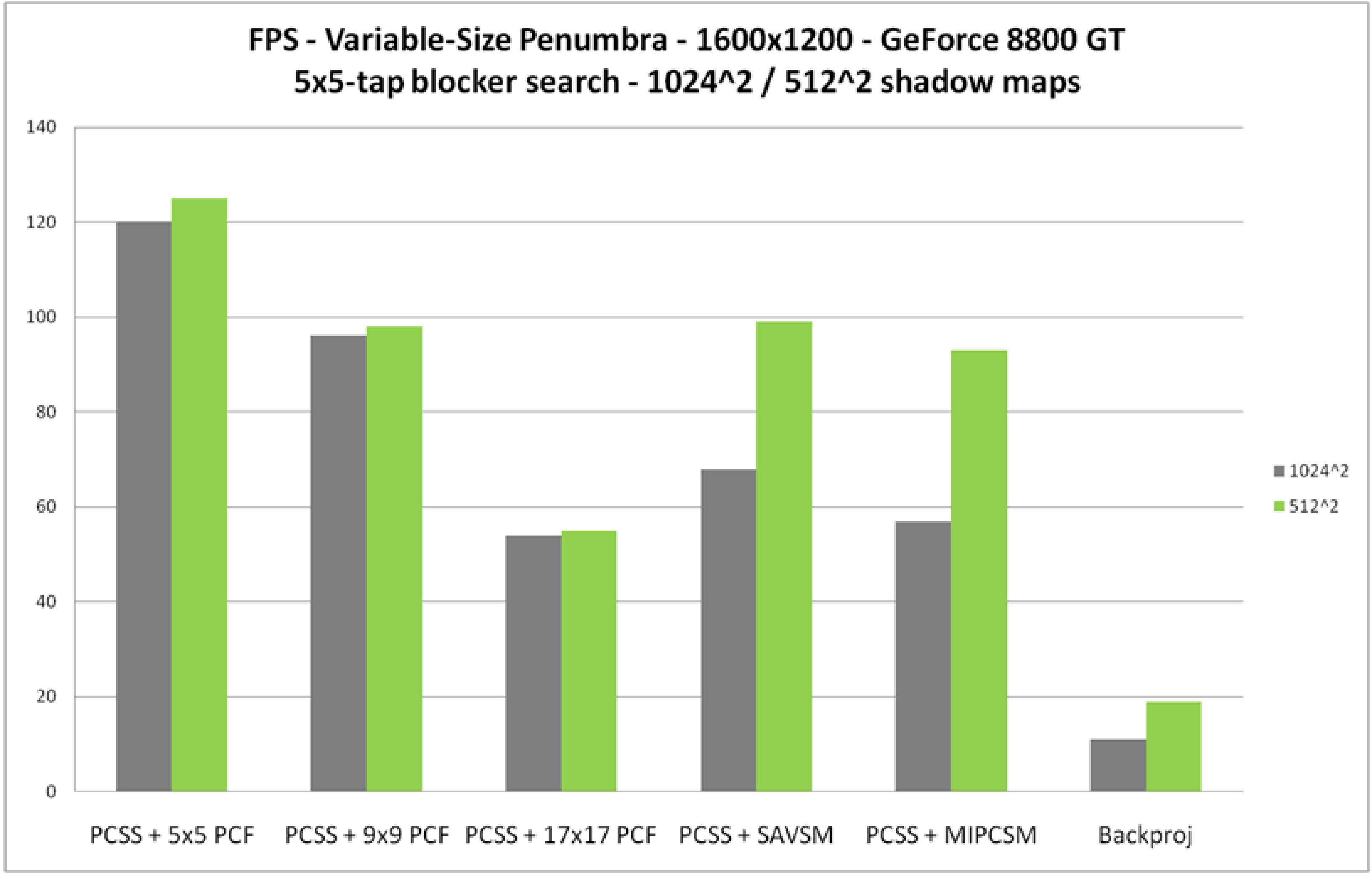
Image: 1600x1200
Shadow Map: 1024^2
Triangle Count: 211k
GeForce 8800 GT

Hierarchical Backprojection

Frame Rate: 11 fps (91 ms)
Min-Max Mipmap: 2.6 ms (3%)

Image: 1600x1200
Shadow Map: 1024^2
Triangle Count: 211k
GeForce 8800 GT

Performance Comparison



Conclusions

- VSM/CSM/ESM all have some sort of light bleeding
  - There are ways to control it
    - VSM: threshold and/or layered VSM [Lauritzen08]
    - CSM: more coefficients [Annen07]
    - ESM: scale factor [Salvi08]
- Percentage Closer Soft Shadows (PCSS) is good stuff
  - Used in shipping game (Hellgate: London)
  - See our latest PCSS whitepaper (February 2008)
  - Two step process:
    - 1. compute filter size using point sampling
    - 2. filter the shadow map using PCF

Conclusions

- Can swap PCF filtering with other filtering methods
  - PCSS + SAVSM works great
    - See GPU Gems 3 Chapter for SAVSM (including source code)
    - Can reduce light bleeding using layered VSMs
- Ideas for accelerating the blocker search
  - Min-max mipmap hierarchical traversal
  - Summed Area Table in the case where the shadow receivers are not in the PCSS shadow map
- Shadow map generation recommendation
  - Use a separate stream for the depth pre-pass with the minimum set of vertex attributes

Acknowledgments

- NVIDIA
  - Kevin Myers
  - Randy Fernando
  - Kirill Dmitriev
  - Yury Uralsky
  - Per Vognsen
- Marco Salvi (ESM)
- Andrew Lauritzen (VSM)

Resources

- Implementation details
  - PCSS
    - PCSS integration guide (February 2008)
    - DirectX 10 sample to be released soon
  - Hierarchical backprojection
    - SoftShadows sample in our SDK10
  - SAVSM
    - GPU Gems 3 chapter
- Models
  - AT-AT: Brad Blackburn / www.scifi3d.com
  - Trees: Generated using Dryad
  - Buddha: Stanford Mesh Repository

References

- [Salvi08] Marco Salvi, "Rendering Filtered Shadows with Exponential Shadow Maps", ShaderX6
- [Lauritzen08] Andrew Lauritzen, Michael McCool, "Layered Variance Shadow Maps", I3D 2008 (poster)
- [Annen07] T. Annen, T. Mertens, P. Bekaert, H.-P. Seidel, J. Kautz, "Convolution Shadow Maps", EGSR, June 2007
- [Dmitriev07] Kirill Dmitriev, Yury Uralsky "Soft shadows using hierarchical min-max shadow maps", GDC 2007
- [Schuler06] Christian Schueler, "Normal Mapping without Pre-Computed Tangents", ShaderX5
- [Isidoro06] John Isidoro, "Shadow Mapping: GPU-based Tips and Techniques", GDC 2006

References

- [Donnelly06] William Donnelly, Andrew Lauritzen, “Variance Shadow Maps, I3D 2006
- [Guennebaud06] G. Guennebaud, L. Barthe, M. Paulin, “Real-time Soft Shadow Mapping by Backprojection”, EGSR 2006
- [Fernando05] Randy Fernando, “Percentage-Closer Soft Shadows”, SIGGRAPH 2005
- [Hensley05] J. Hensley, T. Scheuermann, G. Coombe, M. Singh, A. Lastra, “Fast Summed-Area Table Generation and its Applications”, Computer Graphics Forum, 2005
- [Atty05] L. Atty, N. Holzschuch, M. Lapierre, J.-M. Hasenfratz, C. Hansen, F. Sillion, “Soft Shadow Maps: Efficient Sampling of Light Source Visibility”, Technical Report, 2005