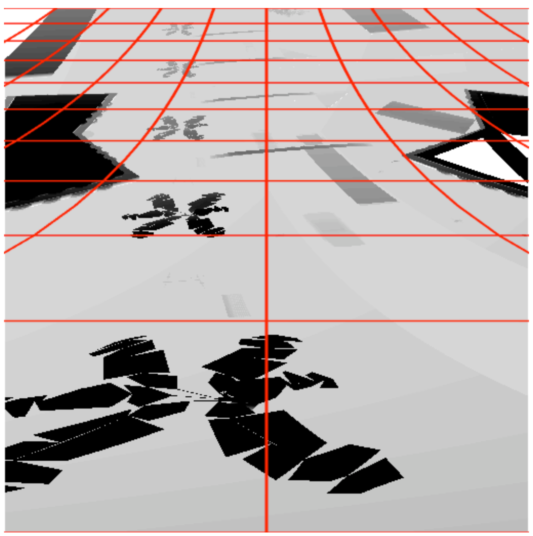
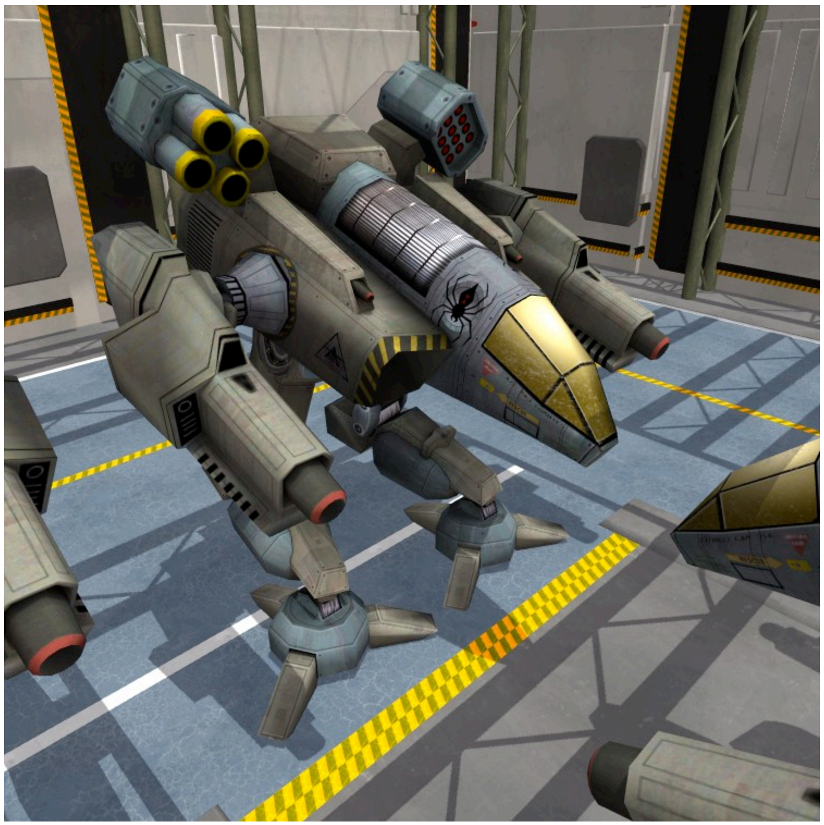
Practical logarithmic rasterization for low error shadow maps



Brandon Lloyd	UNC-CH
Naga Govindaraju	Microsoft
Steve Molnar	NVIDIA
Dinesh Manocha	UNC-CH

Shadows

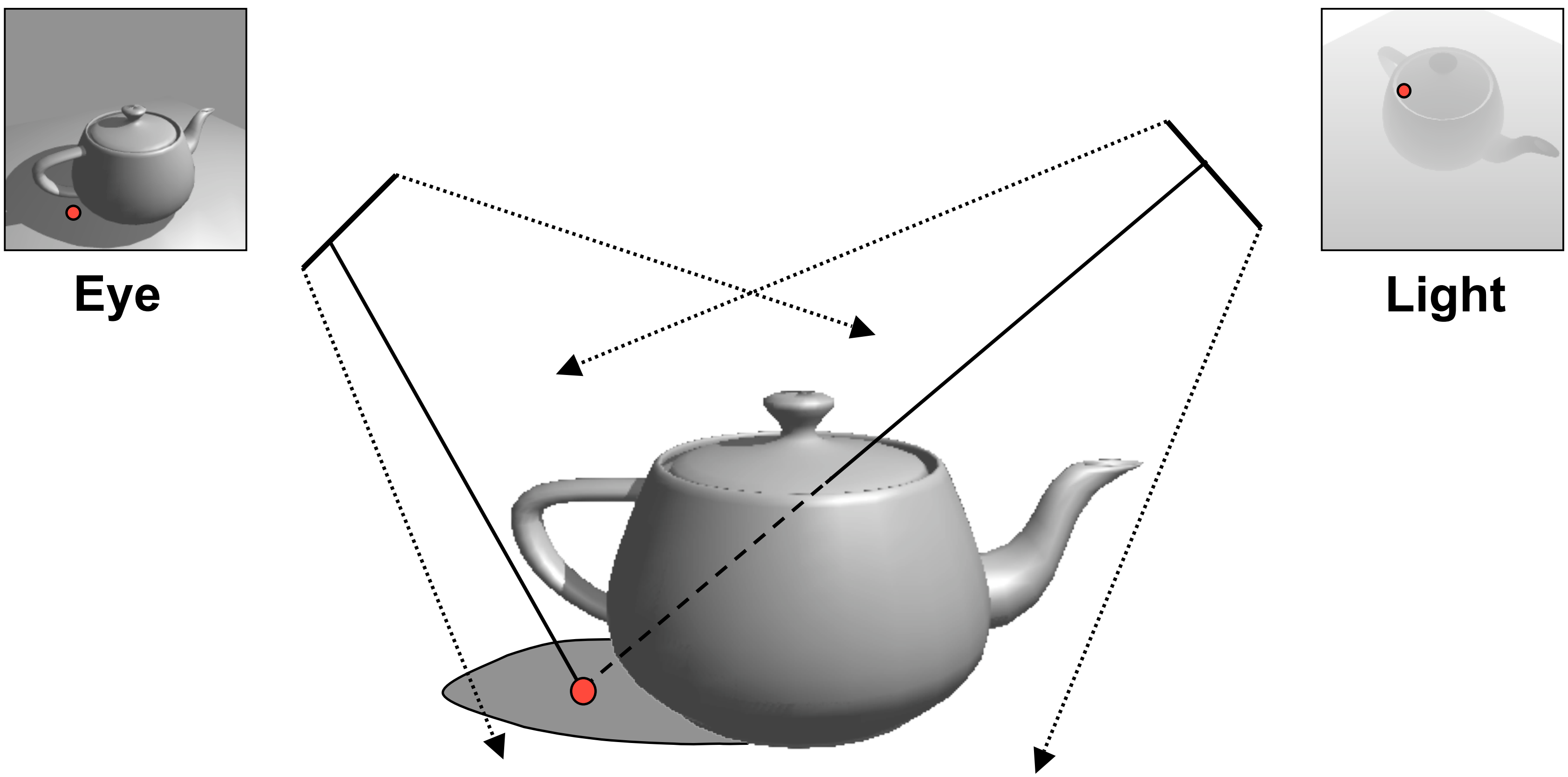


- **Shadows are important**
  - **aid spatial reasoning**
  - **enhance realism**
  - **can be used for dramatic effect**
- **High quality shadows for real-time applications remains a challenge**

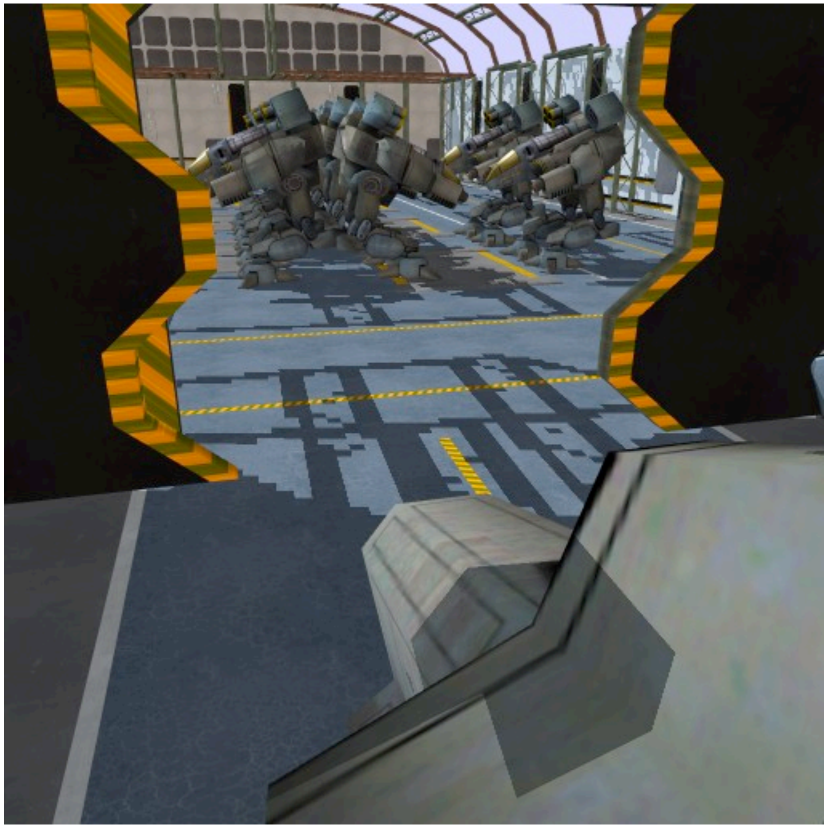
Shadow approaches

- **Raytracing** [Whitted 1980]
  - not yet real-time for complex, dynamic scenes at high resolutions
- **Shadow volumes** [Crow 1977]
  - can exhibit poor performance on complex scenes

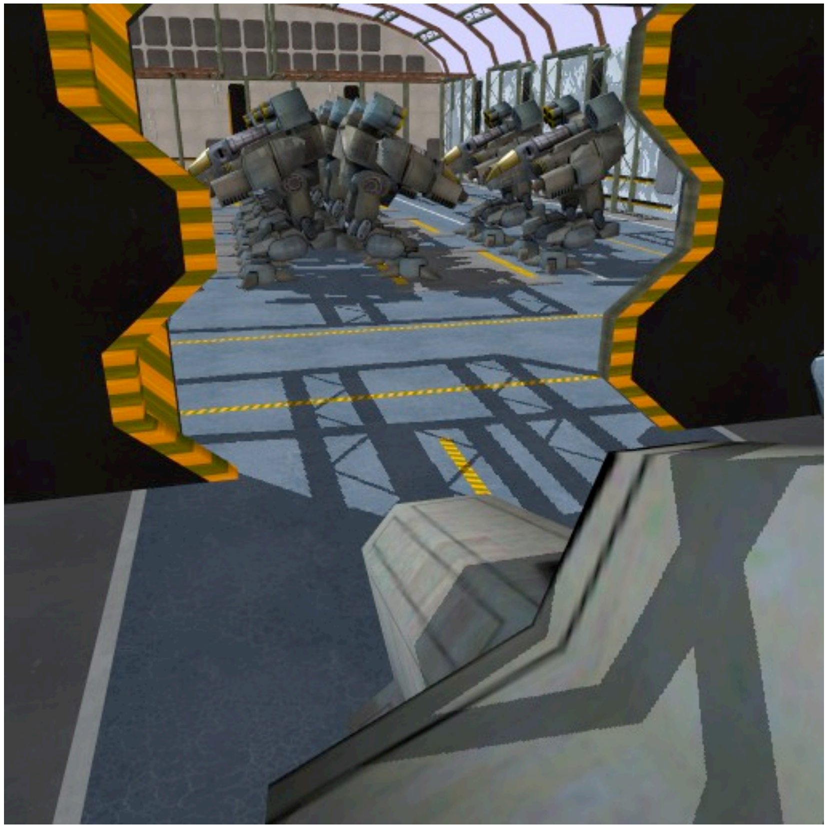
Shadow maps [Williams 1978]



Logarithmic perspective shadow maps (LogPSMs) [Lloyd et al. 2007]

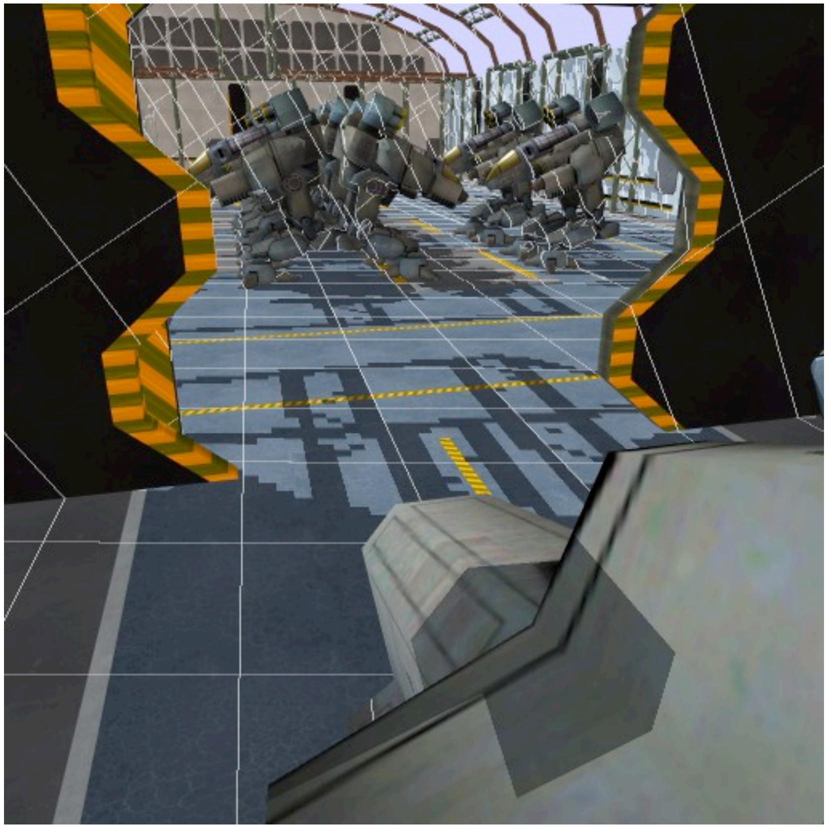


Standard shadow map

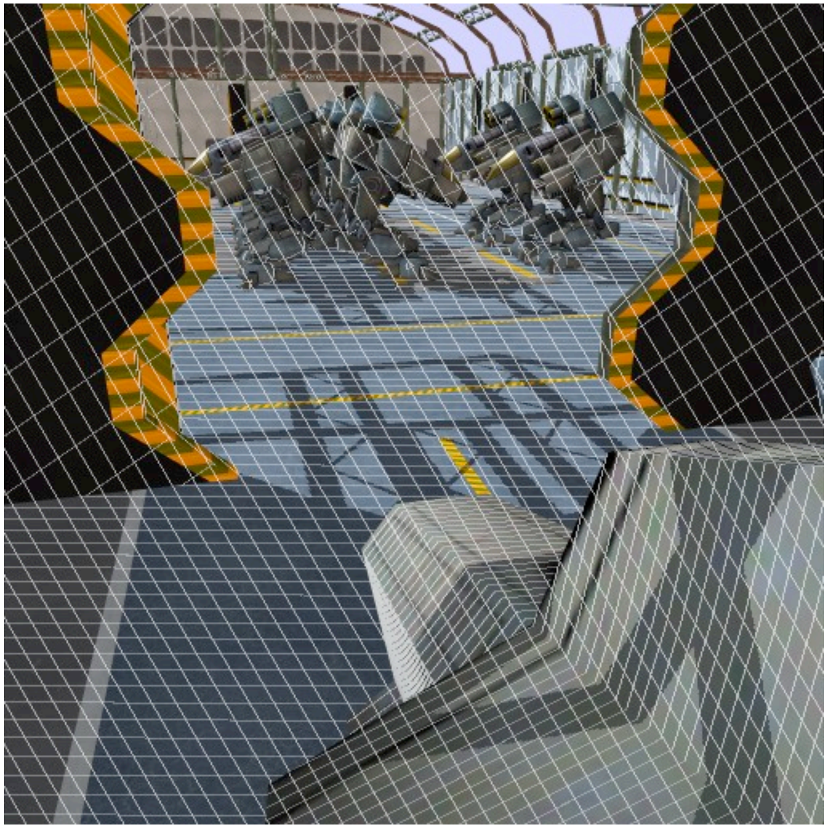


LogPSM

Logarithmic perspective shadow maps (LogPSMs) [Lloyd et al. 2007]

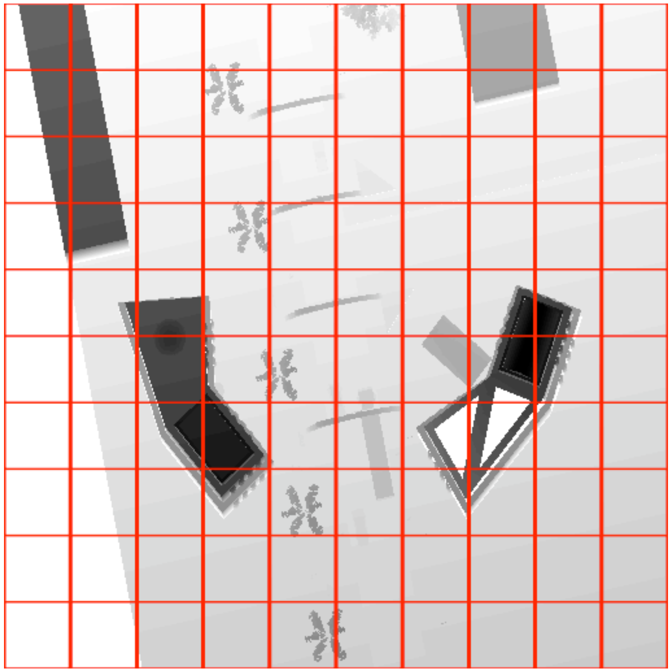


Standard shadow map

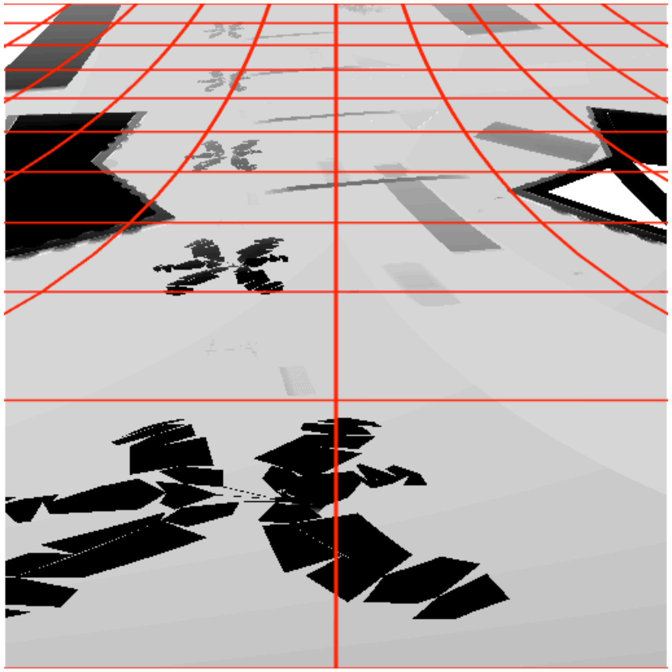


LogPSM

Goal



linear rasterization



logarithmic rasterization

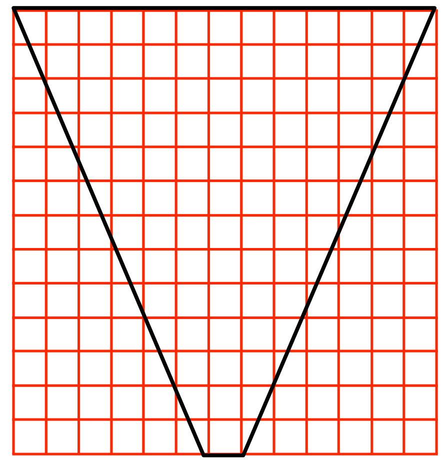
Perform logarithmic rasterization at rates comparable to linear rasterization

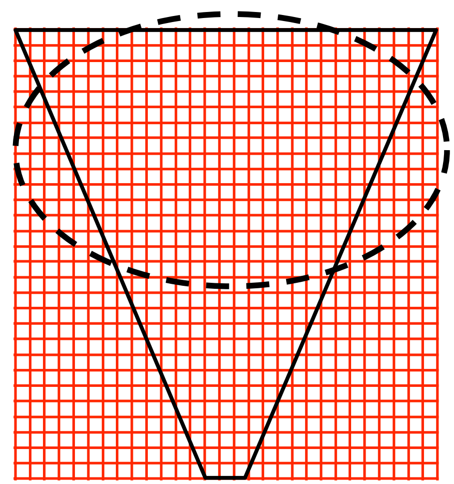
Outline

- Background
  - Handling aliasing error
  - LogPSMs
- Hardware enhancements
- Conclusion and Future work

High resolution shadow maps

- Requires more bandwidth
  - Decreases shadow map rendering performance
- Requires more storage
  - Increased contention for limited GPU memory
- Decreased cache coherence
  - Decreases image rendering performance

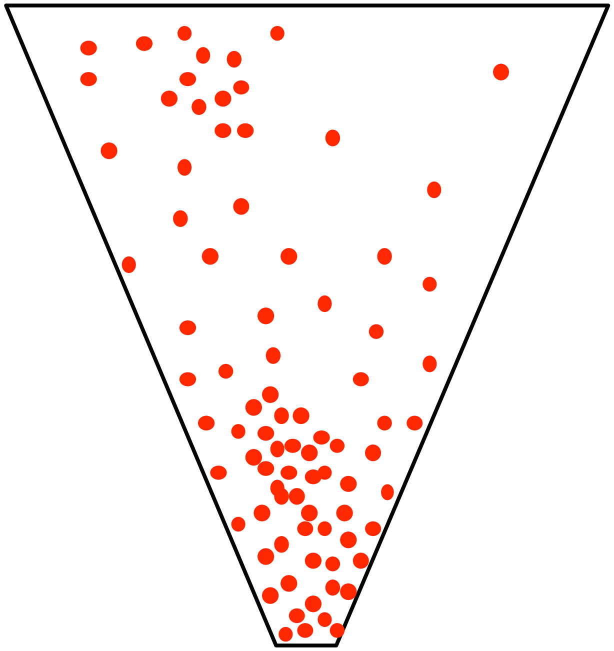




Poor shadow map query locality

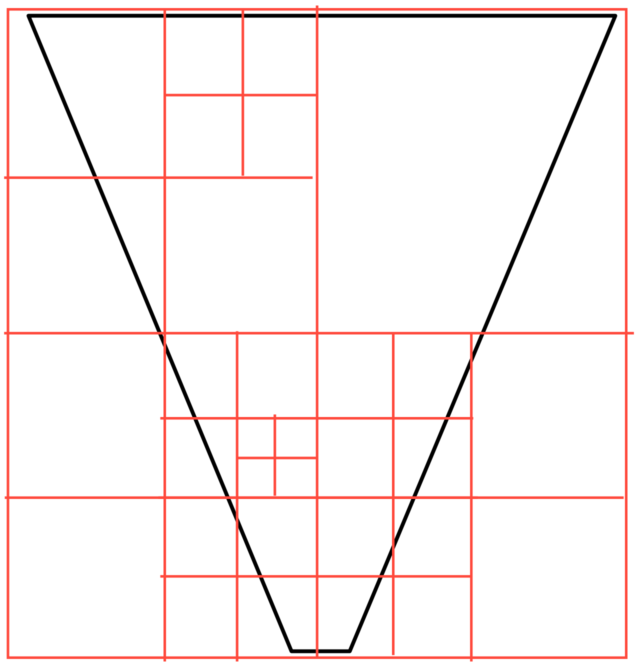
Irregular z-buffer [Aila and Laine 2004; Johnson et al. 2004]

- Sample at shadow map query positions
  - No aliasing
- Uses irregular data structures
  - requires fundamental changes to graphics hardware [Johnson et al. 2005]



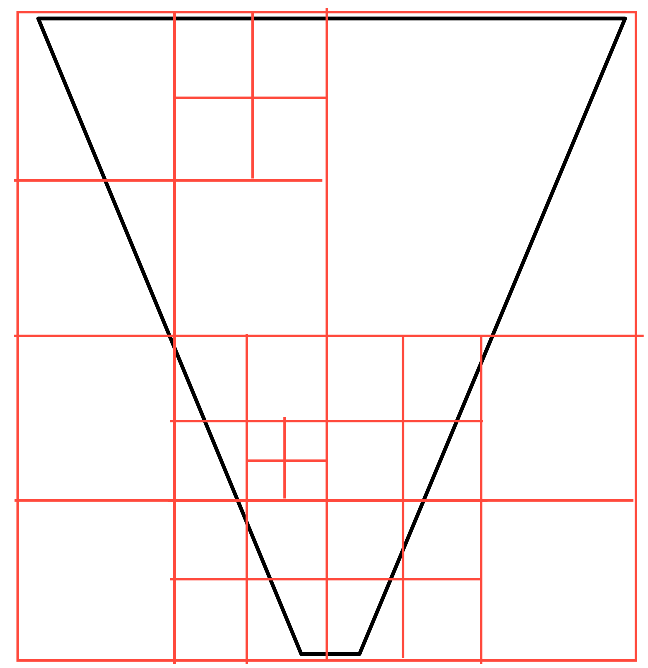
Adaptive partitioning

- **Adaptive shadow maps**
  [Fernando et al. 2001]
- **Queried virtual shadow maps**
  [Geigl and Wimmer 2007]
- **Fitted virtual shadow maps**
  [Geigl and Wimmer 2007]
- **Resolution matched shadow maps**
  [Lefohn et al. 2007]
- **Multiple shadow frusta**
  [Forsyth 2006]



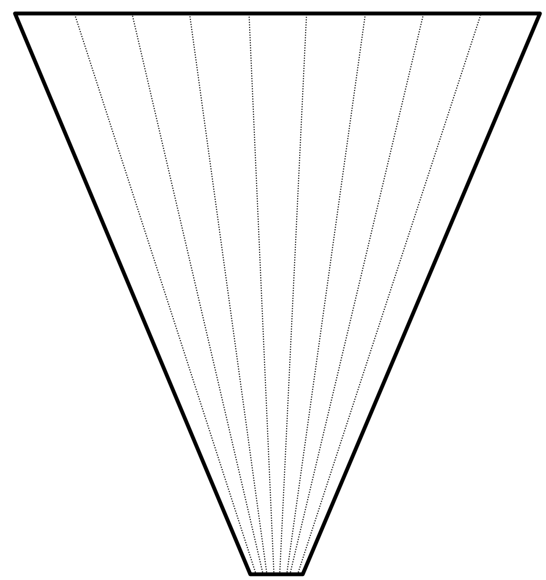
Adaptive partitioning

- Requires scene analysis
- Uses many rendering passes



Scene-independent schemes

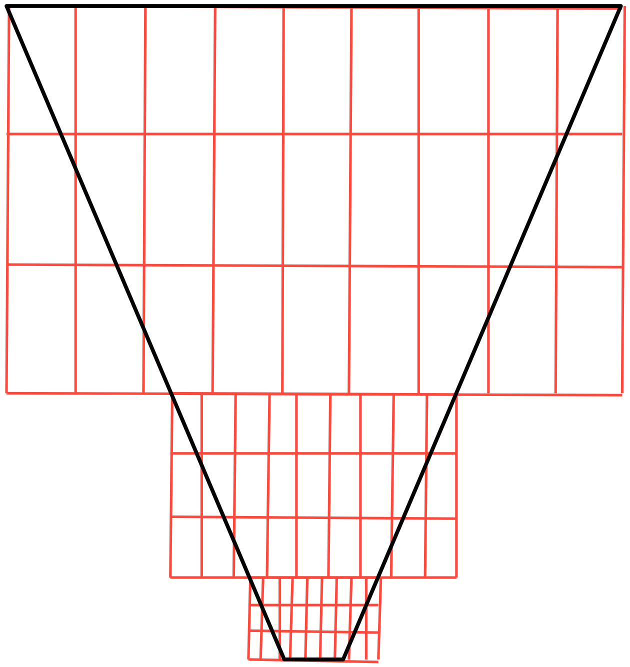
- **Match spacing between eye samples**
- **Faster than adaptive partitioning**
  - **no scene analysis**
  - **few render passes**



eye sample spacing

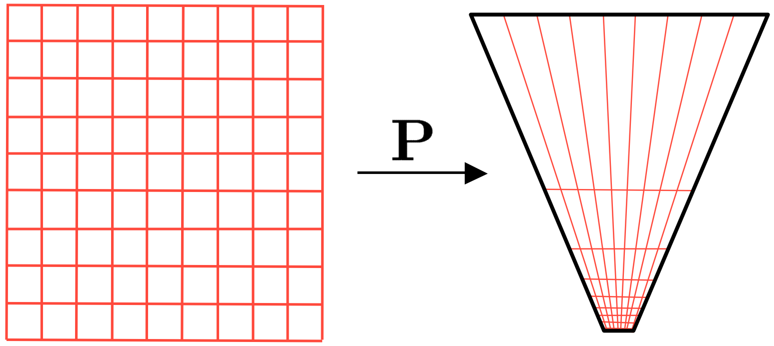
Cascade shadow maps

- **Cascaded shadow maps [Engel 2007]**
- **Parallel split shadow maps [Zhang et al. 2006]**



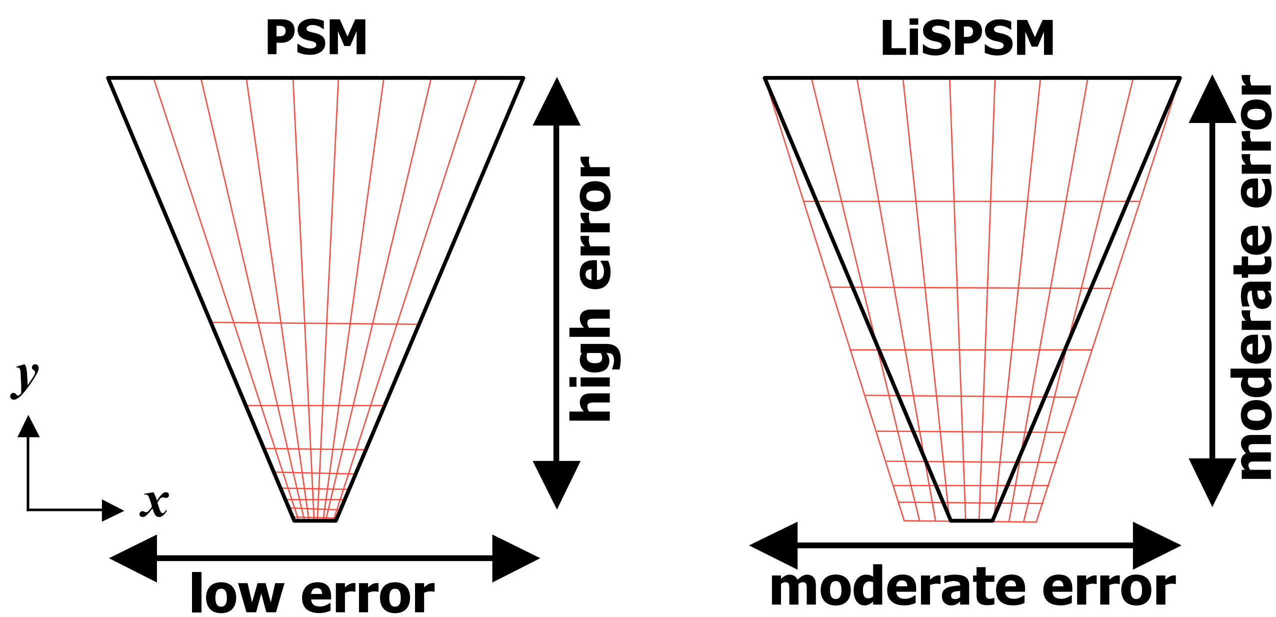
Projective warping

- **Perspective shadow maps (PSMs)** [Stamminger and Drettakis 2002]
- **Light-space perspective shadow maps (LiSPSMs)** [Wimmer et al. 2004]
- **Trapezoidal shadow maps (TSMs)** [Martin and Tan 2004]
- **Lixel for every pixel** [Chong and Gortler 2004]

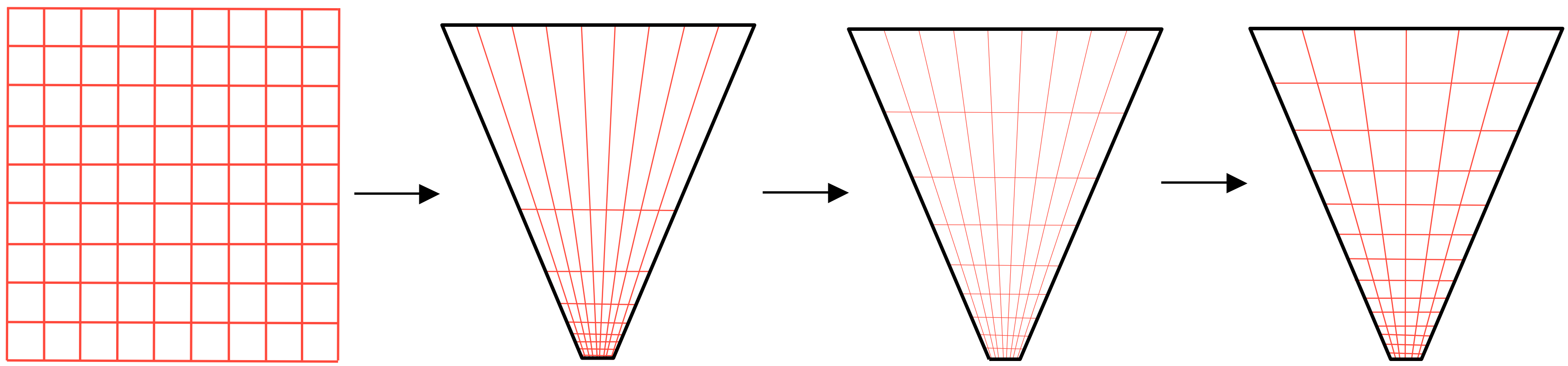


Projective warping

- **Not necessarily the best spacing distribution**



Logarithmic+perspective parameterization



Perspective projection

Logarithmic transform

Resolution redistribution

$$F(y) = c_0 \log(c_1 y + 1)$$

$$c_0 = \frac{-1}{\log(f/n)}, \quad c_1 = \frac{1-(f/n)}{(f/n)}$$

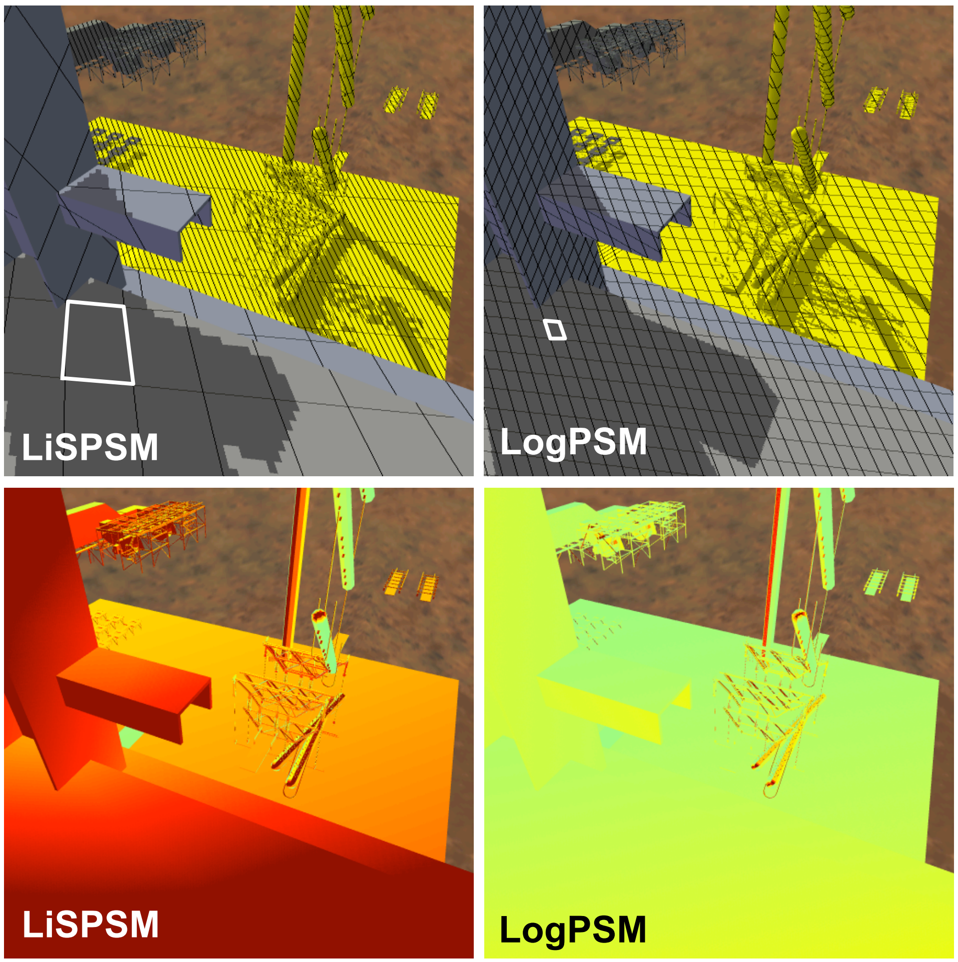
Bandwidth/storage savings

Size of the shadow map* required to remove aliasing error (ignoring surface orientation)	
Uniform	$O\left((f/n)^2\right)$
Perspective	$O\left(f/n\right)$
Logarithmic + perspective	$O\left(\log(f/n)\right)$

n, f - near and far plane distances of view frustum

*shadow map texels / image pixels

Single shadow map LogPSM



- **LogPSMs have**
  - **lower maximum error**
  - **more uniform error**

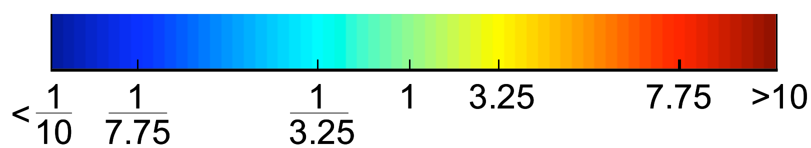
Image resolution: 512^2

Shadow map resolution: 1024^2

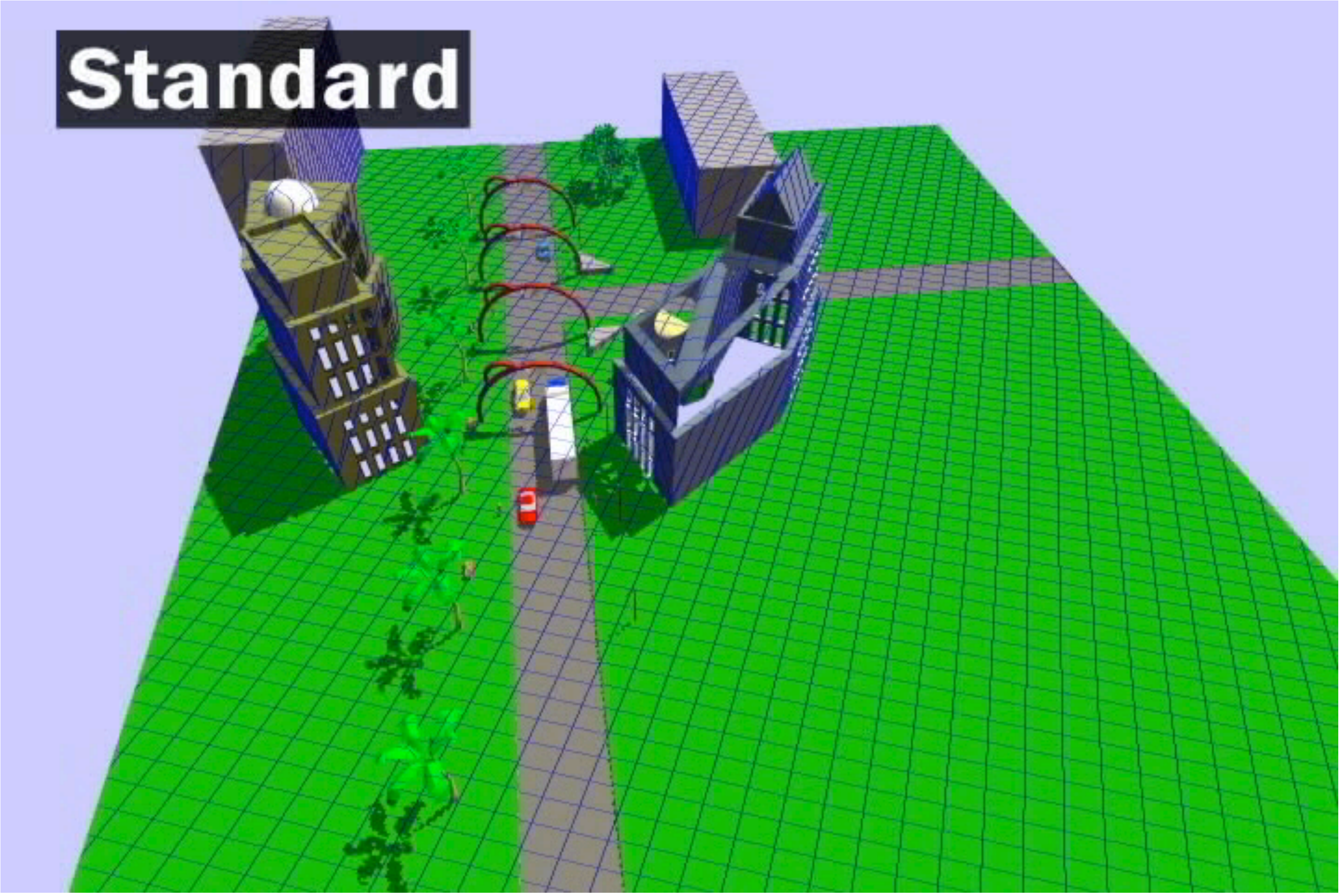
f/n = 300

Grid lines for every 10 shadow map texels

Color coding for maximum texel extent in image



Comparisons



More details

Logarithmic perspective shadow maps

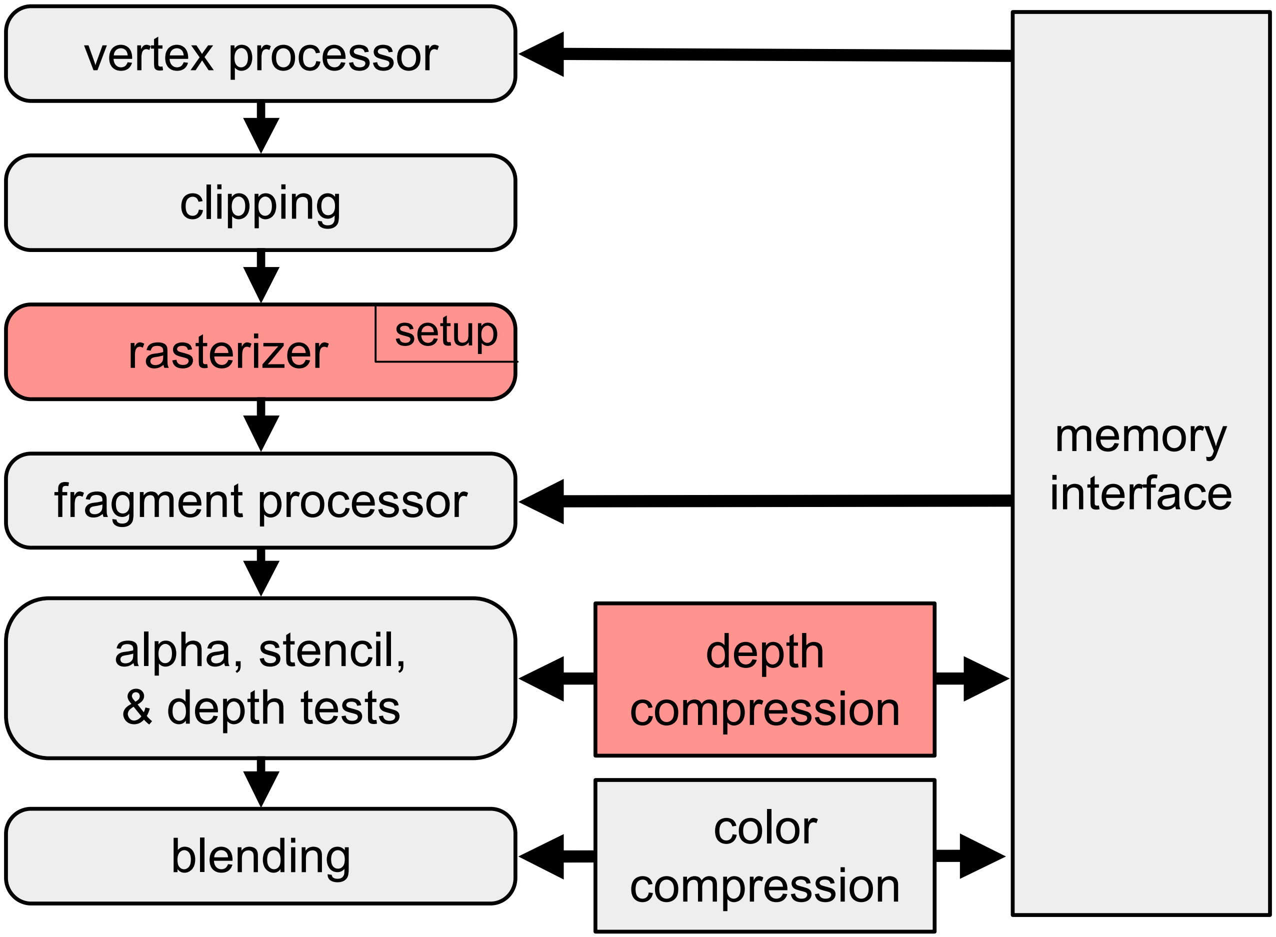
UNC TR07-005

http://gamma.cs.unc.edu/logpsm

Outline

- Background
- **Hardware enhancements**
  - **rasterization to nonuniform grid**
  - **generalized polygon offset**
  - **depth compression**
- Conclusion and Future work

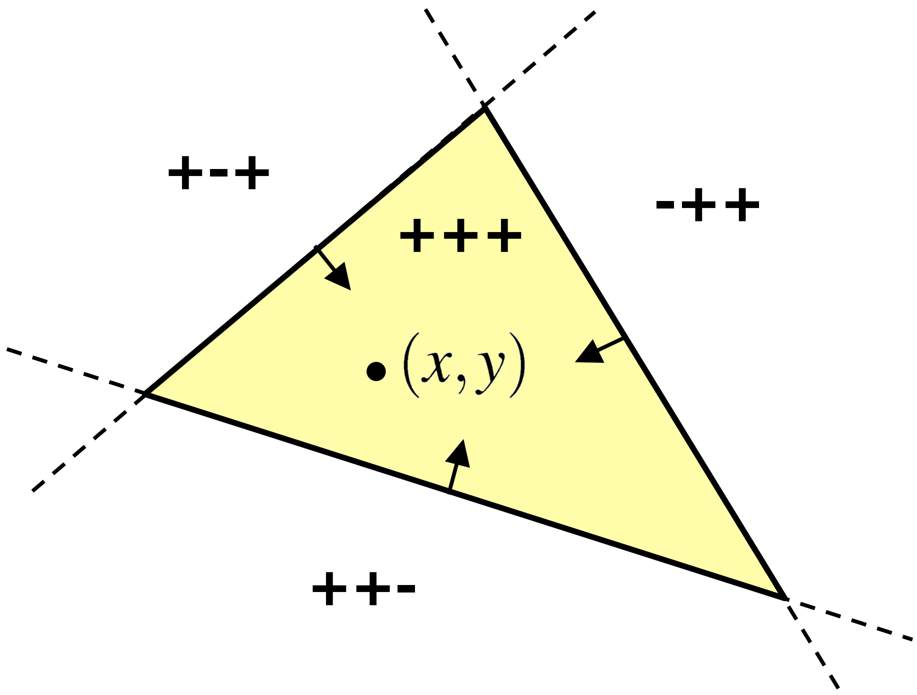
Graphics pipeline



Rasterization

- **Coverage determination**
  - **coarse stage – compute covered tiles**
  - **fine stage – compute covered pixels**
- **Attribute interpolation**
  - **interpolate from vertices**
  - **depth, color, texture coordinates, etc.**

Edge equations



$$E(x,y) = Ax + By + C$$

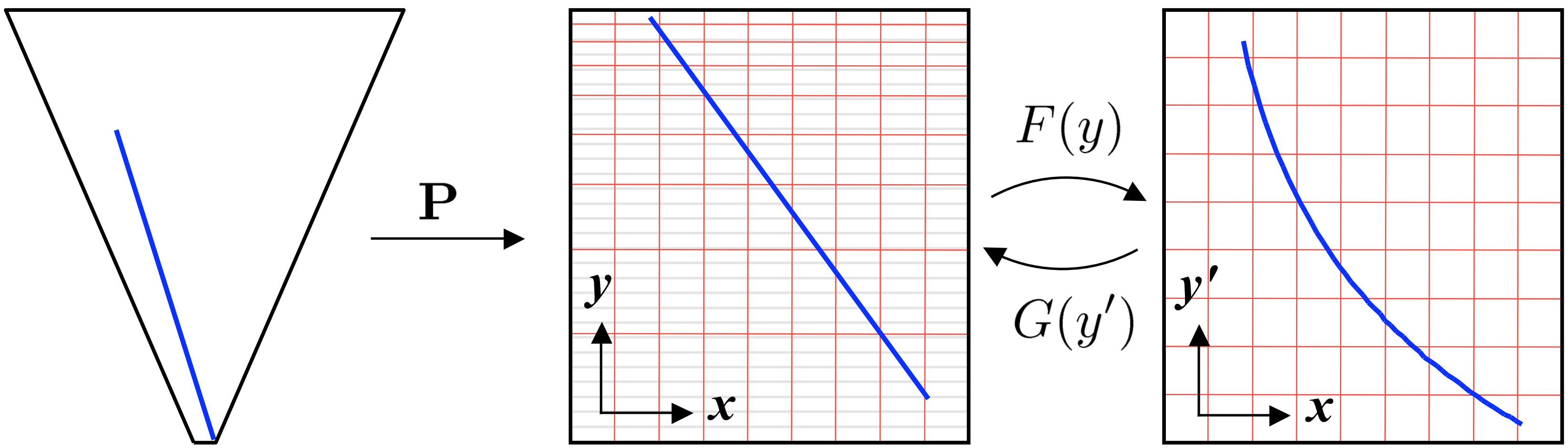
- **Signs used to compute coverage**
- **Water-tight rasterization**
  - **Use fixed-point**
  - **fixed-point “snaps” sample locations to an underlying uniform grid**

Attribute interpolation

- **Same form as edge equations:**

$$\boxed{depth(x, y) = Ax + By + C}$$

Logarithmic rasterization



light space **linear shadow map space** **warped shadow map space**

Linear rasterization with nonuniform grid locations.

Edge and interpolation equations

$$
\begin{aligned}
E'(x,y') &= E(x,G(y')) \\
&= Ax + BG(y') + C
\end{aligned}
$$

$$
G(y') = \frac{(f/n)(1-(f/n)^{-y'})}{(f/n)-1}
$$

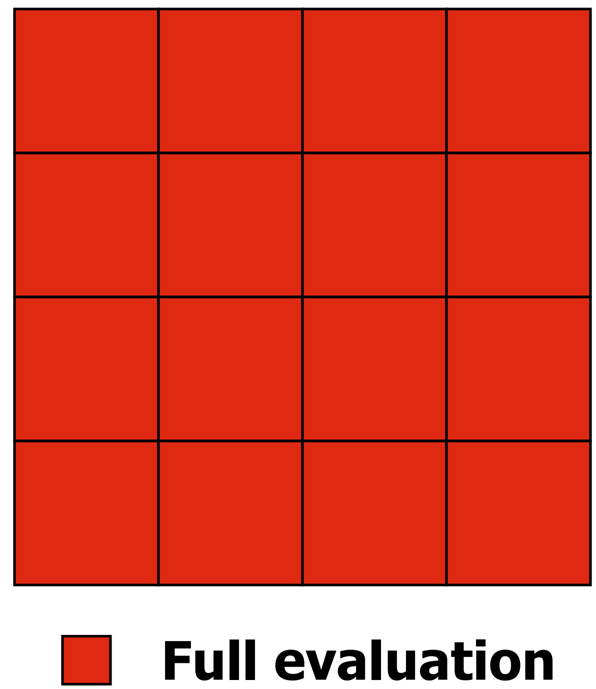
- Monotonic
  - existing tile traversal algorithms still work
  - optimizations like z-min/z-max culling still work

Coverage determination for a tile

- **Full parallel implementation**

$$E'(x, y') = Ax + BG(y') + C$$

$$G(y') = \frac{(f/n)(1 - (f/n)^{-y'})}{(f/n) - 1}$$



Coverage determination for a tile

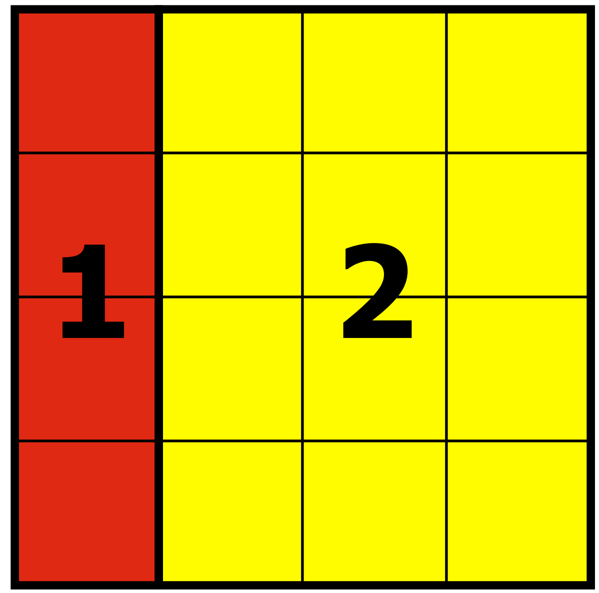
- **Incremental in x**

$$E'(x, y') = Ax + BG(y') + C$$

$$E'(x_0 + k\Delta x, y') = E'(x_0, y') + Ak\Delta x$$

$$k \in \{1, 2, 3\}$$

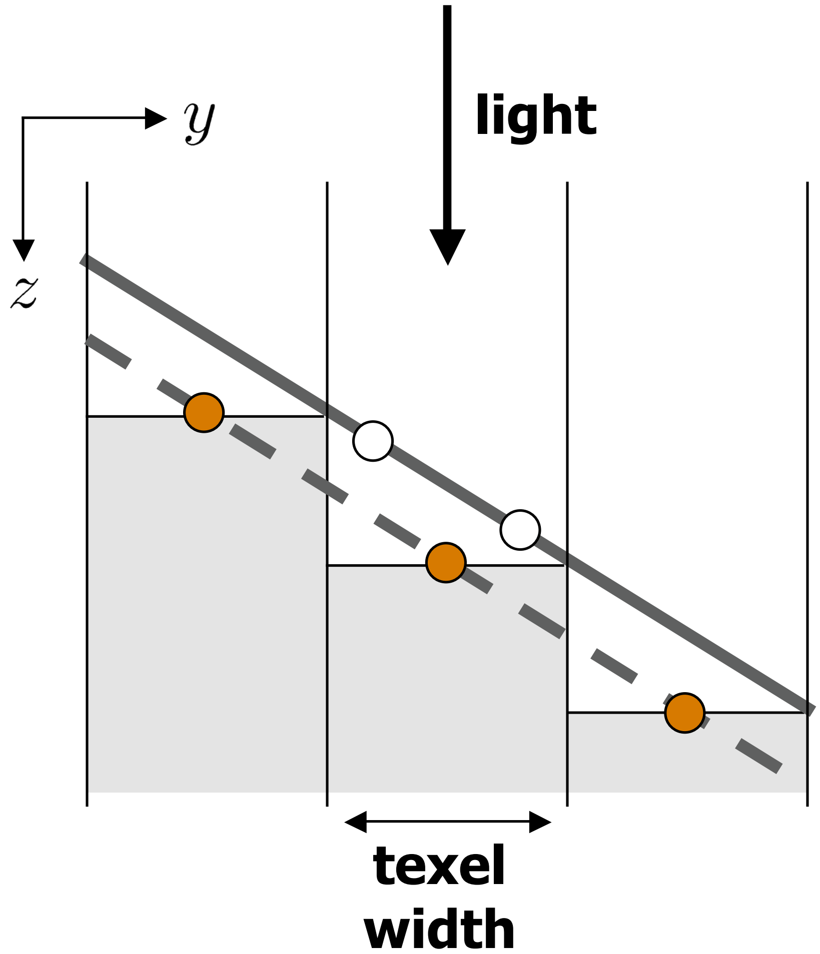
Per-triangle constants



Full evaluation

Incremental x

Generalized polygon offset



$$offset = scale \cdot m_z + bias \cdot \Delta d_{\min}$$

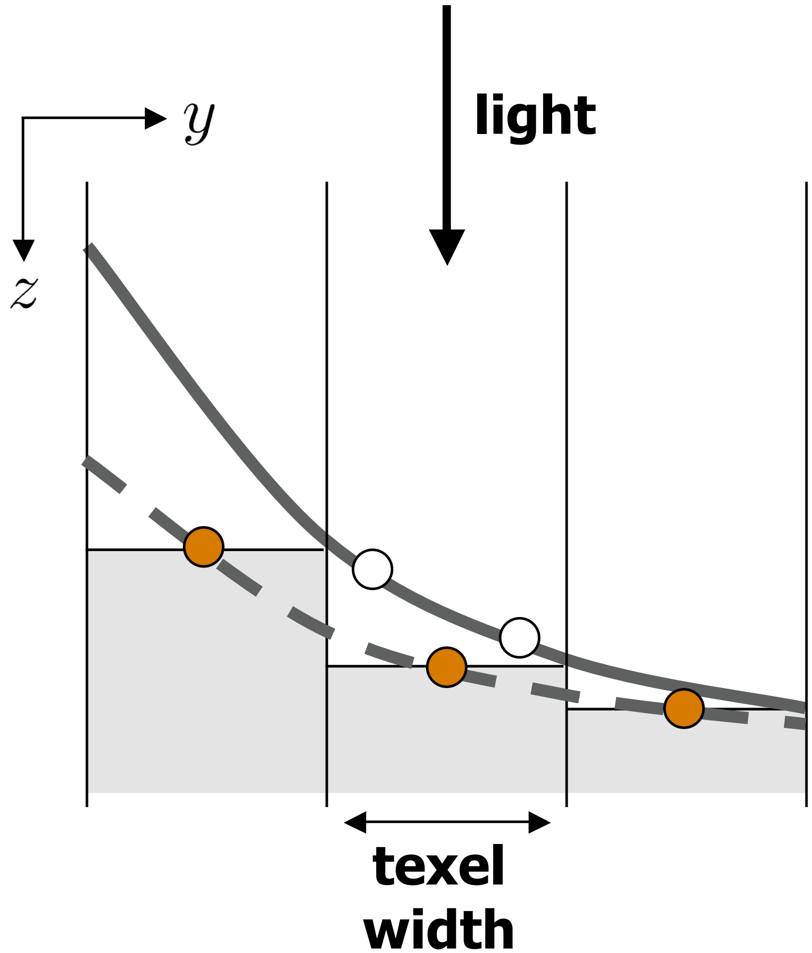
m_z **- depth slope**

$\Delta d_{\min}$ **- smallest representable depth difference**

$$m_z = \max\left(\left|\frac{dx}{dz}\right|, \left|\frac{dy}{dz}\right|\right)$$

constant

Generalized polygon offset



$$offset = scale \cdot m_z + bias \cdot \Delta d_{\min}$$

m_z **- depth slope**

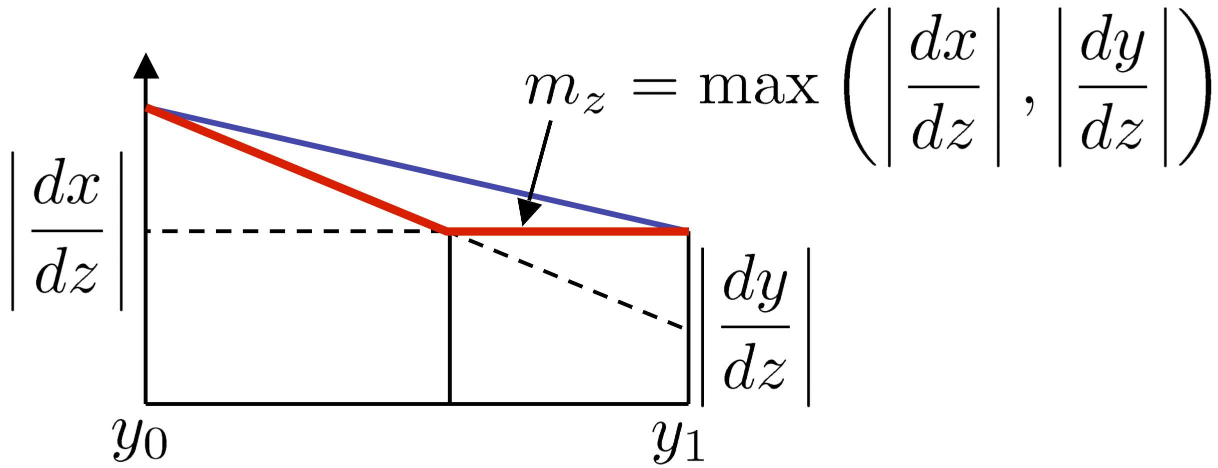
$\Delta d_{\min}$ **- smallest representable depth difference**

$$m_z = \max\left(\left|\frac{dx}{dz}\right|, \left|\frac{dy}{dz}\right|\right)$$

constant

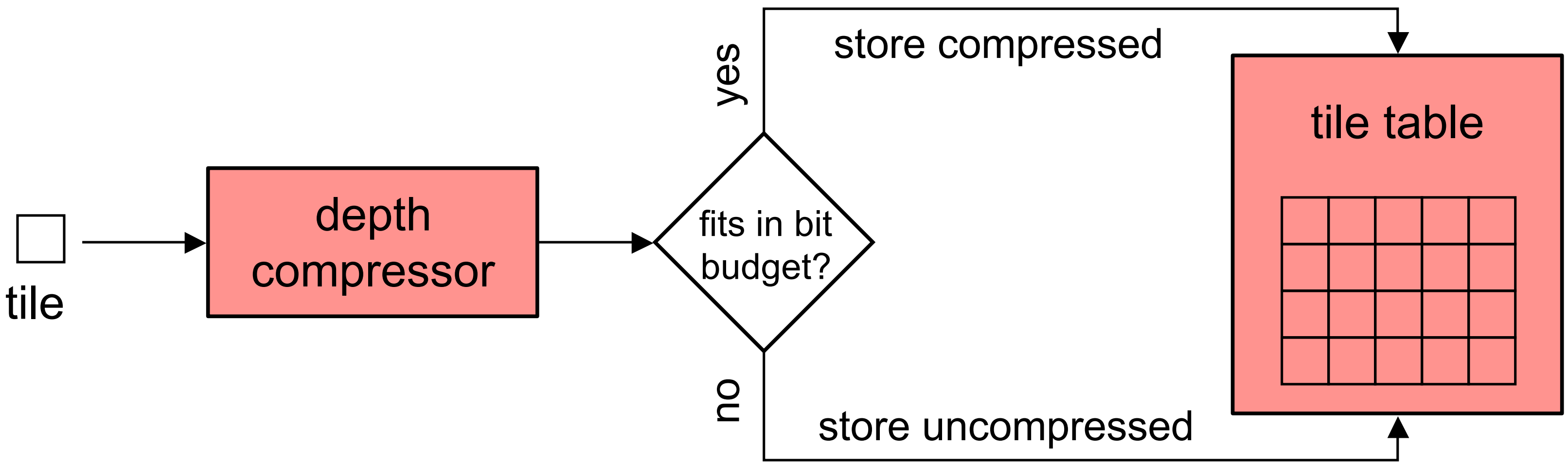
not constant

Generalized polygon offset



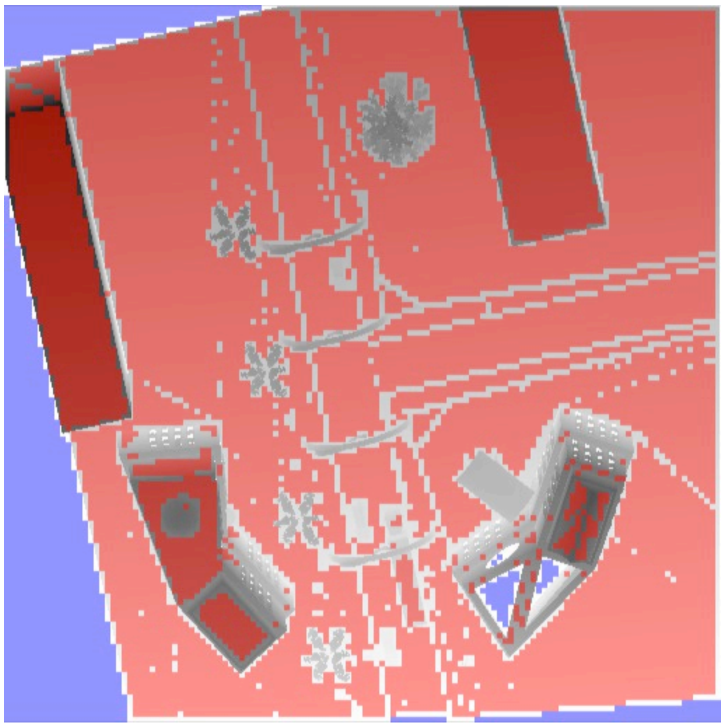
- Do max per pixel
- Split polygon
- Interpolate max at end points

Depth compression

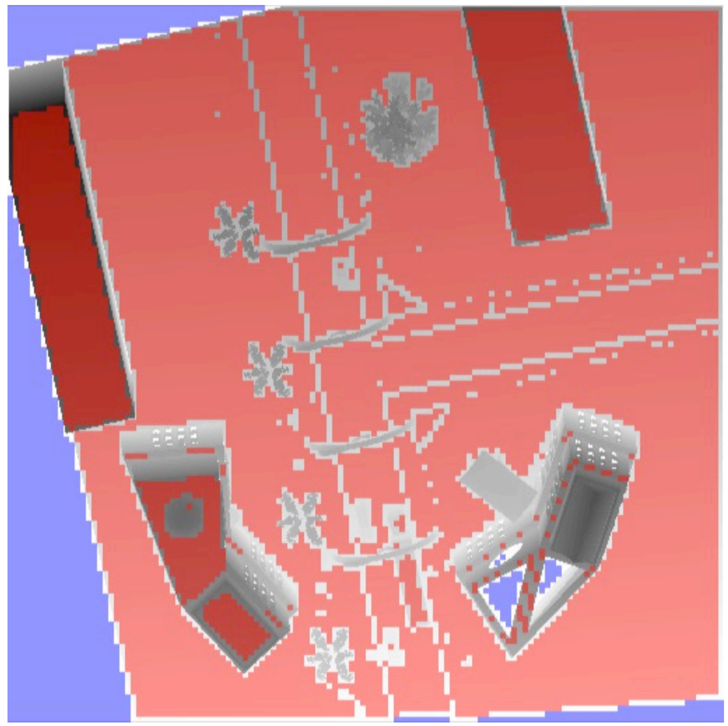


- **Important for reducing memory bandwidth requirements**
- **Exploits planarity of depth values**
- **Depth compression survey [Hasselgren and Möller 2006]**

Depth compression - Standard



Linear depth compression



Our depth compression

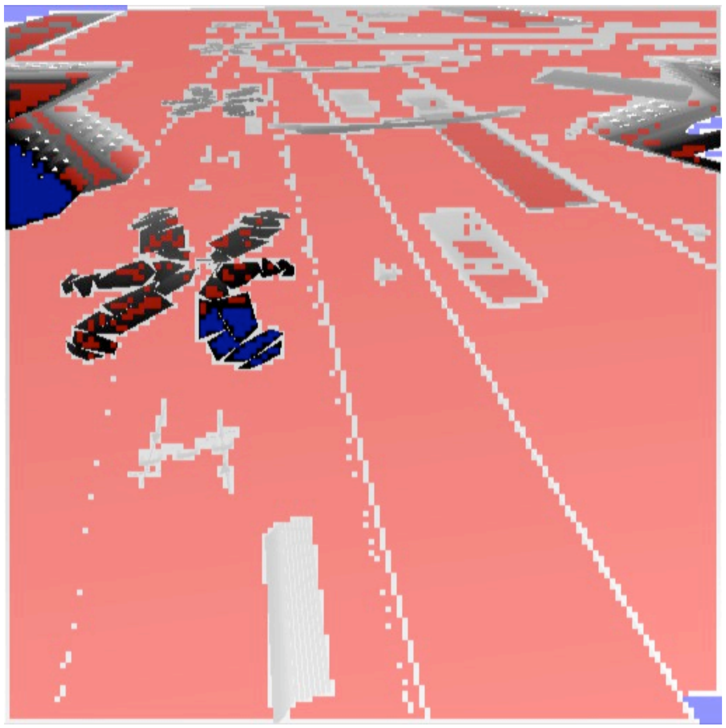
compressed

untouched

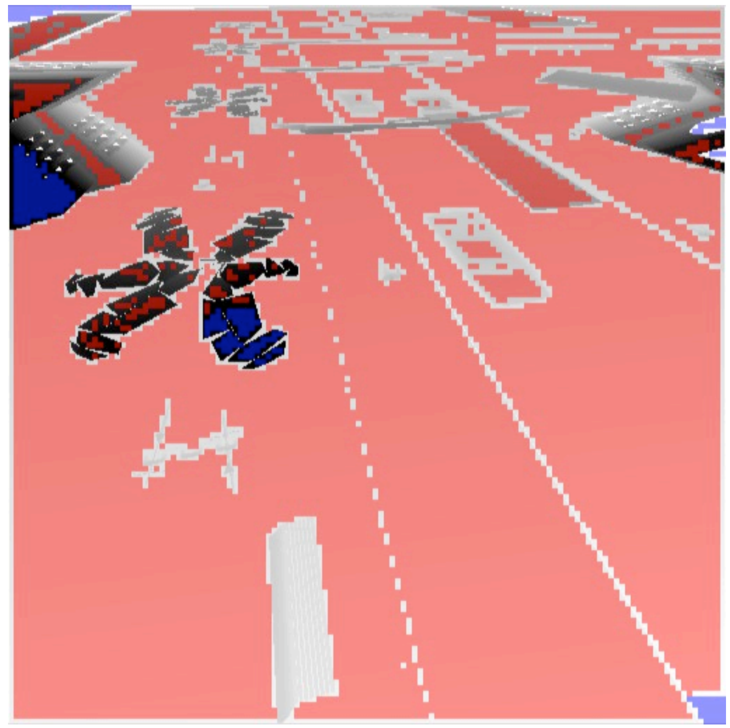
clamped

Resolution: 512x512

Depth compression - LiSPSM



Linear depth compression

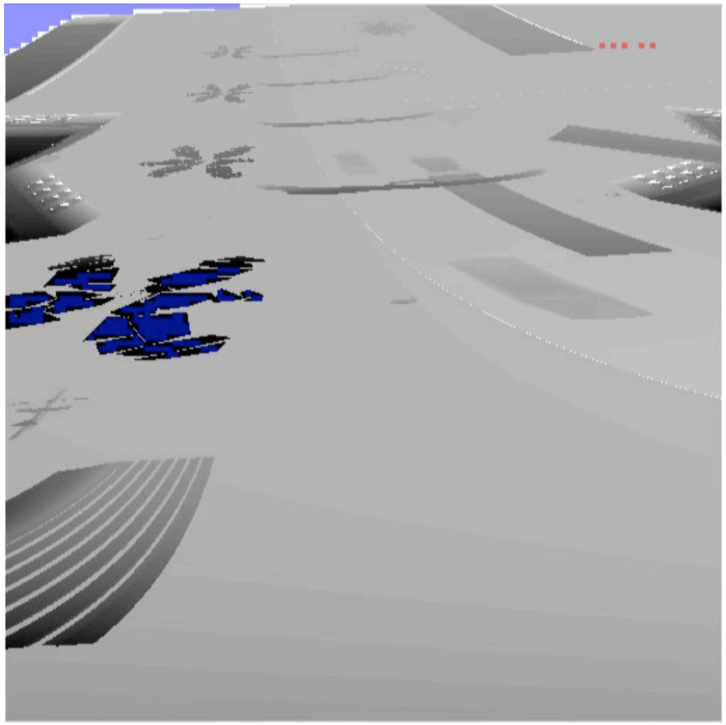


Our depth compression

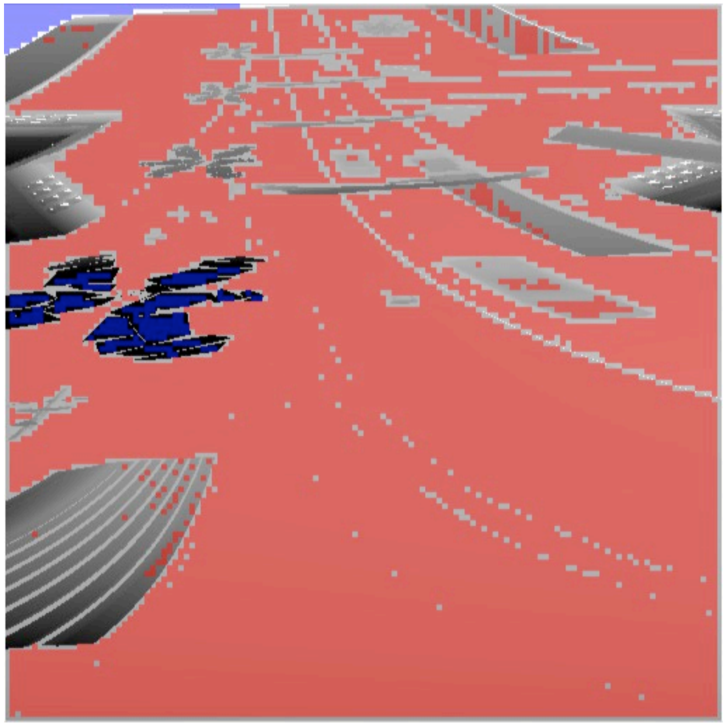
- compressed
- untouched
- clamped

Resolution: 512x512

Depth compression - LogPSM



Linear depth compression



Our depth compression

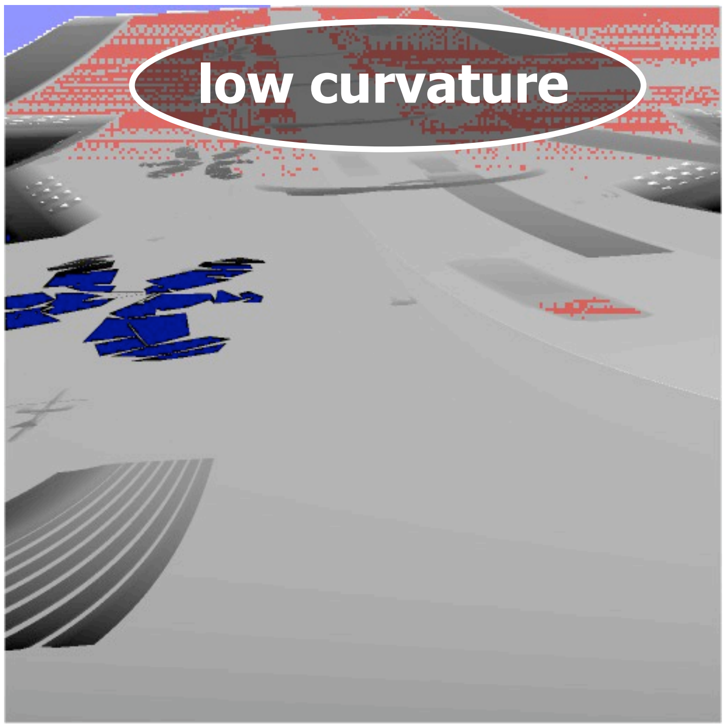
compressed

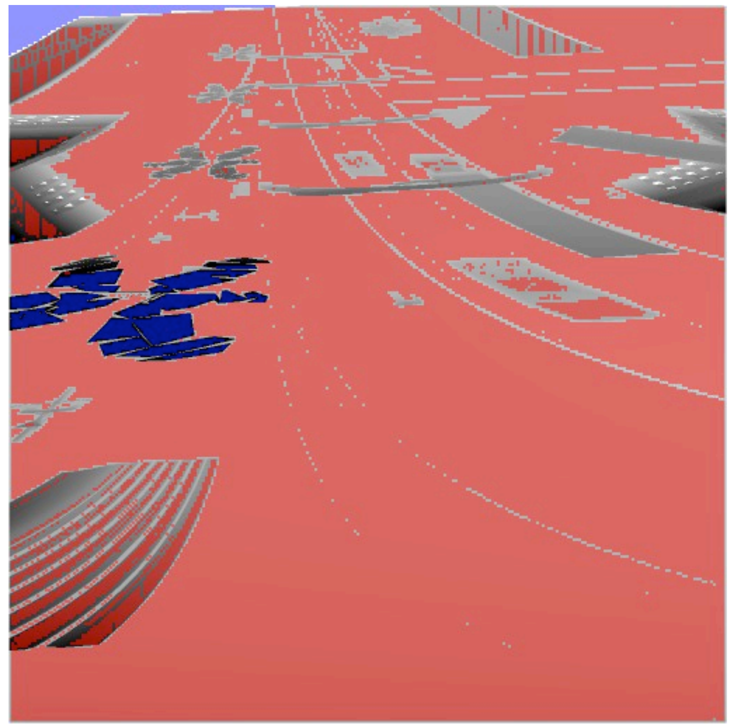
untouched

clamped

Resolution: 512x512

Depth compression – LogPSM
Higher resolution





- compressed
- untouched
- clamped

Resolution: 1024x1024

Linear depth compression **Our depth compression**

Our compression scheme

z_0	Δx	Δx	Δx
Δy	Δx	Δx	Δx
Δy	Δx	Δx	Δx
Δy	Δx	Δx	Δx

Differential encoding

z_0	d	Δy	d
d	d	a_1	Δx
a_0	d	Δy	d
Δy	d	Δy	d

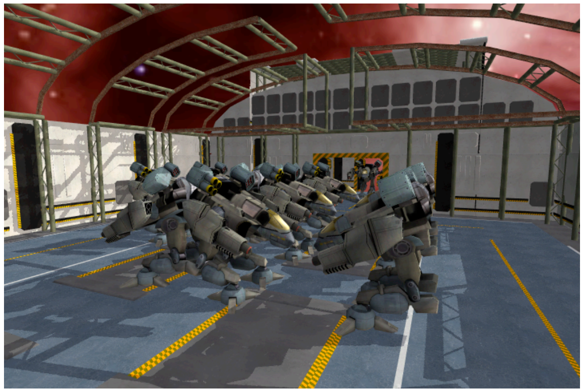
Anchor encoding

24	3	6	3
10	3	16	3
18	3	10	3
10	3	9	3

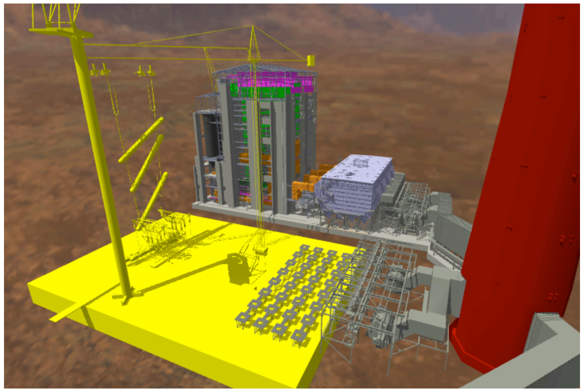
128-bit allocation table

Test scenes

Town model
- **58K triangles**



Robots model
- **95K triangles**

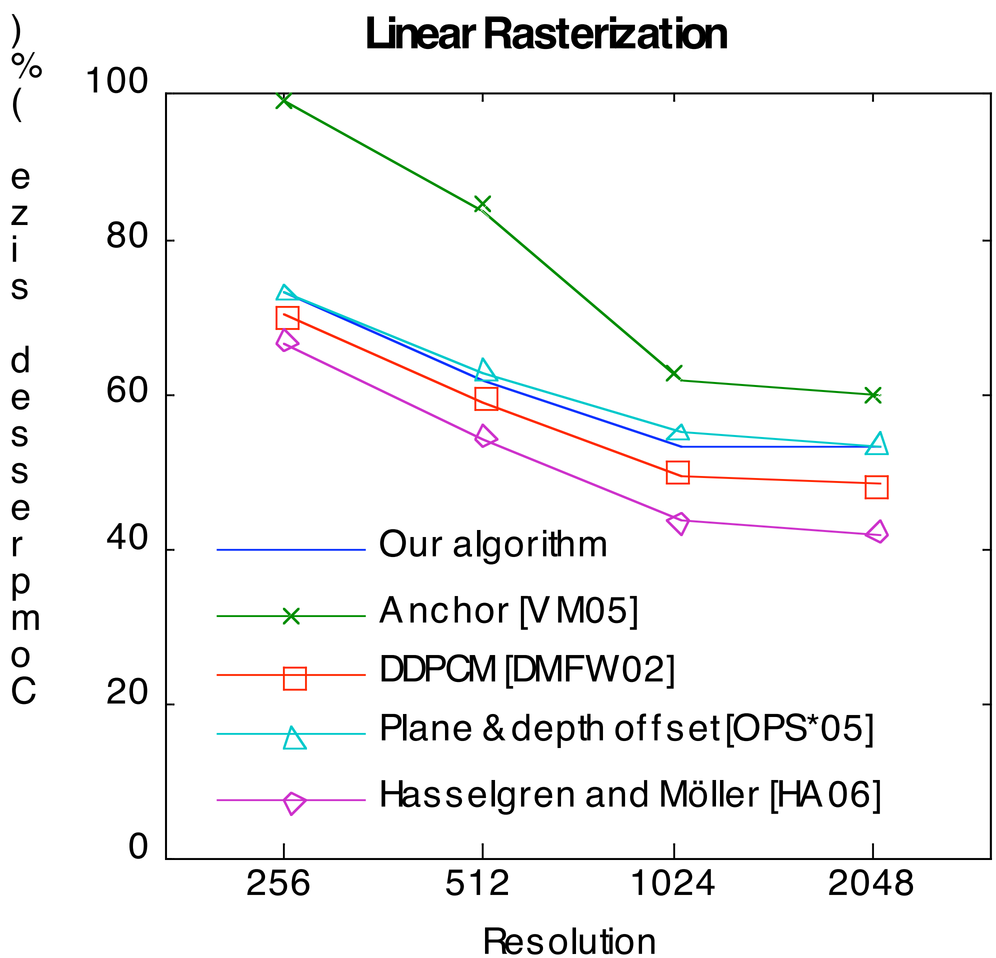


Power plant
- **13M triangles**
- **2M rendered**

Compression methods tested

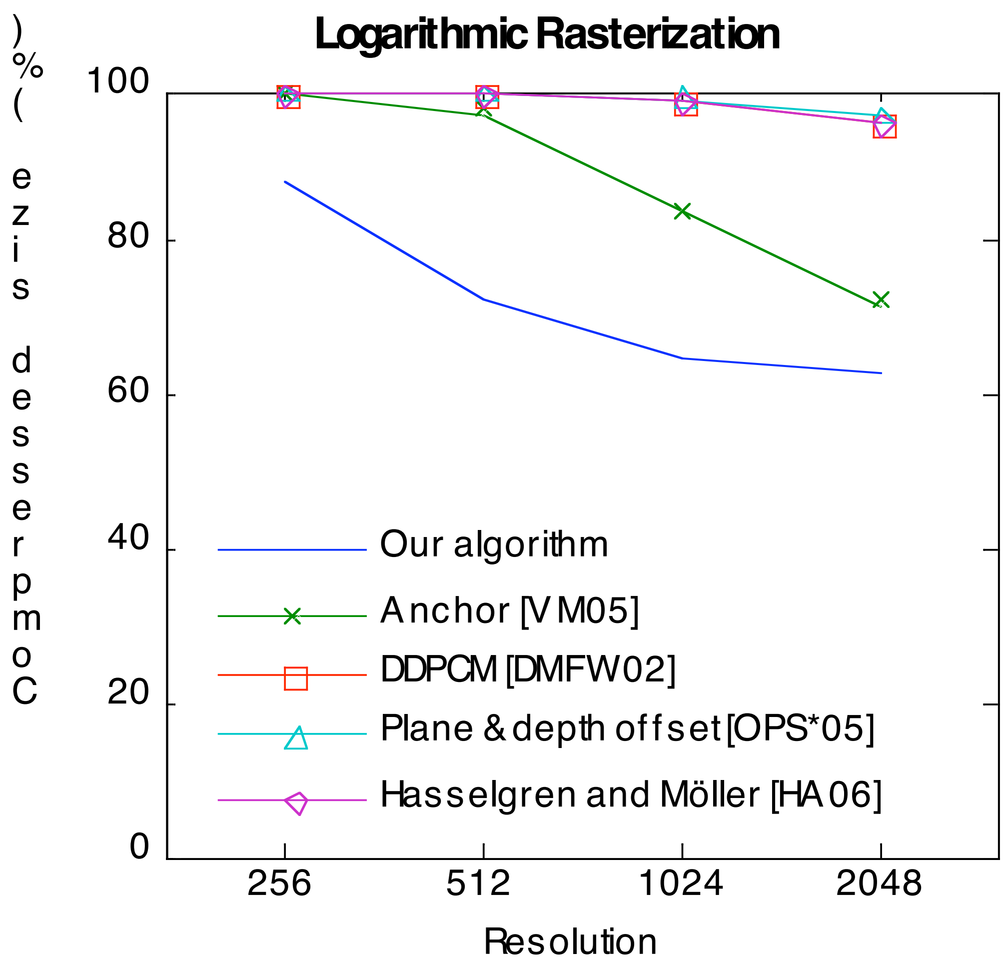
- **Anchor encoding**
  [Van Dyke and Margeson 2005]
- **Differential differential pulse code modulation (DDPCM)**
  [DeRoo et al. 2002]
- **Plane and offset**
  [Ornstein et al. 2005]
- **Hasselgren and Möller** [2006]

Compression results



- **Average compression over paths through all models**
- **Varying light and view direction**

Compression results



- **Anchor encoding best linear method**
  - **higher tolerance for curvature**

Summary of hardware enhancements

- Apply F(y) to vertices in setup
  - log and multiply-add operations
- Evaluators for G(y')
  - exponential and multiply-add operations
- Possible increase in bit width for rasterizer
- Generalized polygon offset
- New depth compression unit
  - can be used for both linear and logarithmic rasterization

Feasibility

- Leverages existing designs
- Trades computation for bandwidth
- Aligns well with current hardware trends
  - computation cheap
  - bandwidth expensive

Conclusion

- Shadow maps
  - Handling errors requires high resolution
- Logarithmic rasterization
  - significant savings in bandwidth and storage
- Incremental hardware enhancements
  - Rasterization to nonuniform grid
  - Generalized polygon offset
  - Depth compression
- Feasible
  - leverages existing designs
  - aligns well with hardware trends

Future work

- Prototype and more detailed analysis
- Greater generalization for the rasterizer
  - reflections, refraction, caustics, multi-perspective rendering [Hou et al. 2006; Liu et al. 2007]
  - paraboloid shadow maps for omnidirectional light sources [Brabec et al. 2002]
  - programmable rasterizer?

Acknowledgements

- Jon Hasselgren and Thomas Akenine-Möller for depth compression code
- Corey Quammen for help with video
- Ben Cloward for robots model
- Aaron Lefohn and Taylor Halliday for town model

Acknowledgements

- Funding agencies
  - NVIDIA University Fellowship
  - NSF Graduate Fellowship
  - ARO Contracts DAAD19-02-1-0390 and W911NF-04-1-0088
  - NSF awards 0400134, 0429583 and 0404088
  - DARPA/RDECOM Contract N61339-04-C-0043
  - Disruptive Technology Office.