NVIDIA®

Perspective Shadow Maps

Gary King

Shadow Mapping Review

- Image-based shadow technique
  - Lance Williams, 1978.
  - As compared to object-based stencil shadows
- First, render depth from light’s point of view
  - e.g., Z-buffer
- When rendering scene, transform fragments into shadowmap, and perform depth comparison
  - If the fragment fails test, it is shadowed

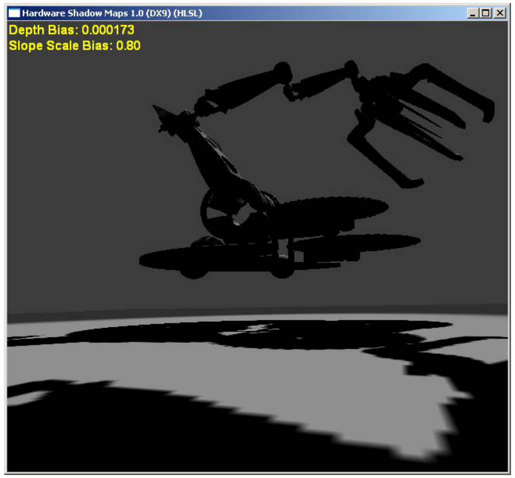
Shadow Mapping on GPUs

- On Radeon 9500+
  - Floating-point textures (R32F)
  - Pixel shader filtering and comparison

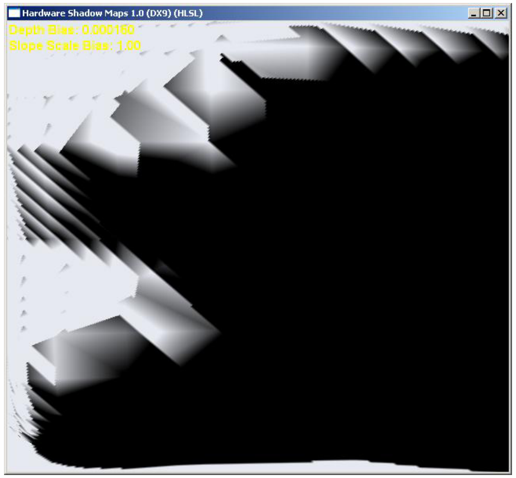
- On GeForce 3+
  - Native shadow map support (16 and 24-bit integer)
  - 2x2 bilinear percentage closer filtering for free
  - Double-speed rendering on GeForceFX and later GPUs

Shadow Mapping Problems

- Aliasing!
  - Objects distant from light may be close to viewer (perspective aliasing)



Depth Bias: 0.000173
Slope Scale Bias: 0.80

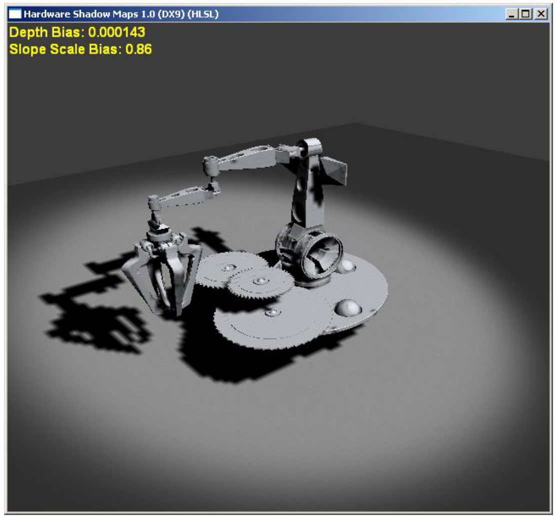


Depth Bias: 0.000160
Slope Scale Bias: 1.00

  - Receivers perpendicular to light projection plane may be parallel to view plane (projective aliasing)

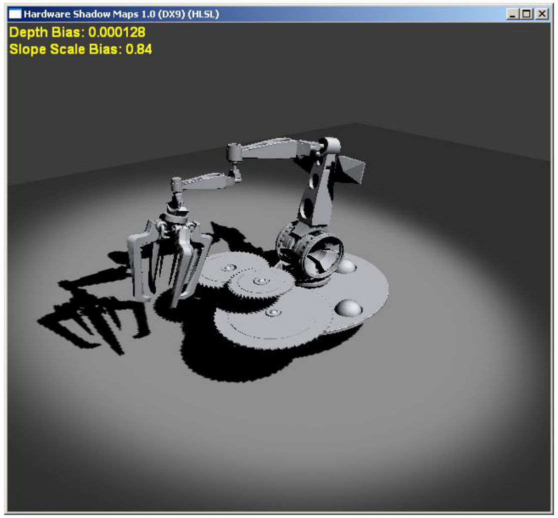
Solving Shadow-Map Problems, #1

- Easiest solution is to increase sample density
  - Just like other aliasing problems
  - This could require a huge shadow map for outdoors
  - 32k x 32k is unrealistic for hardware acceleration



Depth Bias: 0.000143
Slope Scale Bias: 0.86

128x128



Depth Bias: 0.000128
Slope Scale Bias: 0.84

512x512

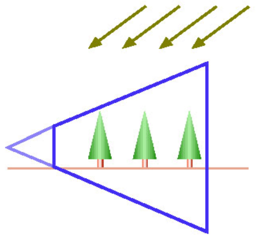
Solving Shadow-Map Problems, #2

- Redistribute samples in shadow map
  - Shadow volumes and ray tracers sample uniformly from viewer
  - Traditional shadow maps sample uniformly from light
- We need a transform that warps light space in a view-dependent way

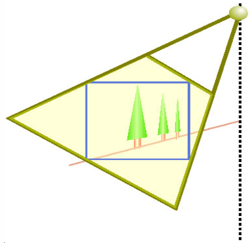
Properties of Post-Projective Space

- All visible objects squeezed into a unit cube
  - [-1,-1,0]..[1,1,1] in D3D
  - [-1,-1,-1]..[1,1,1] in OGL

- The infinity plane (w=0) has a well-defined position
  - Directional lights become point lights on this plane



Eye Space



Post-Projective Space

Perspective Shadow Maps

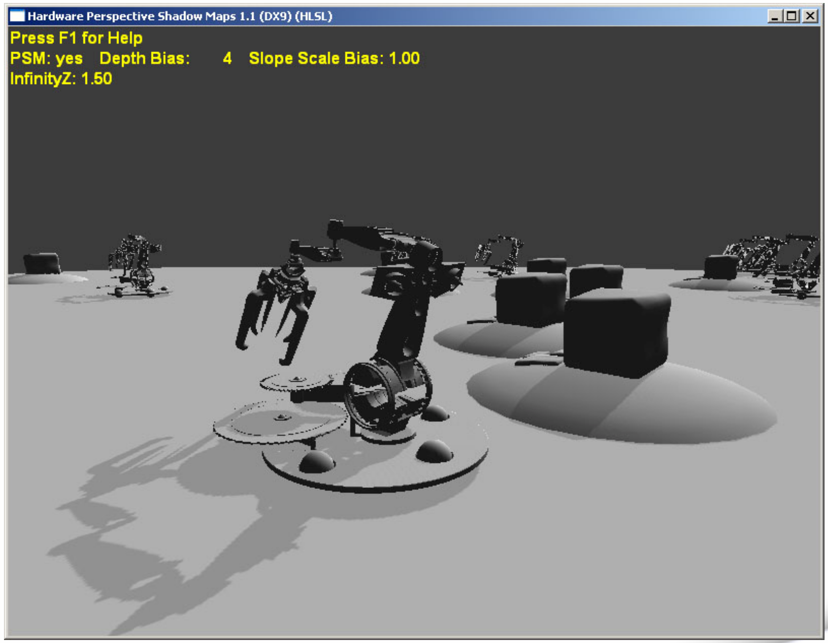
- What about viewer’s projection matrix?
  - Perspective transform makes objects near viewer larger than more distant ones
- Key insight behind Perspective Shadow Maps
  - Stamminger & Drettakis, SIGGRAPH 2002
  - Addresses perspective aliasing
- To build a shadow map for a directional light
  - `LookAt` matrix from post-projective light to view-box
  - Compose with scene `view*projection`

Unfortunately…

- PSMs, as implemented in Stamminger and Drettakis' paper, had quite a few issues
  - Lights from behind viewer
  - Temporal, view-dependent shadow quality
  - Strong near-plane dependence
  - Self-shadow artifacts
- Simon Kozlov's article in *GPU Gems*, "Perspective Shadow Maps: Care and Feeding," addresses all of these issues.

Demo

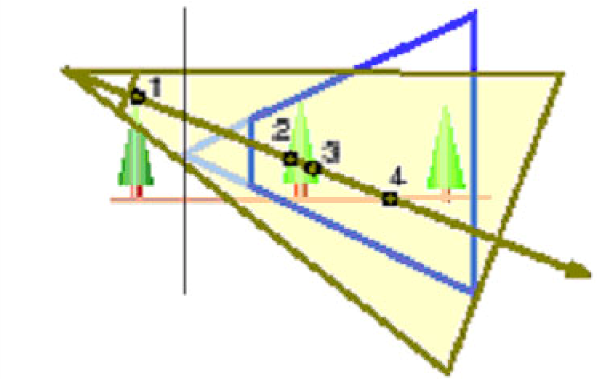
- Large-Scale Full-Scene Shadow Mapping
  - 1600m x 1600m terrain (Z_{near}= 1m, Z_{far}=800m)
  - 40 shadow-casting objects
  - One 1536 x 1536 shadow map



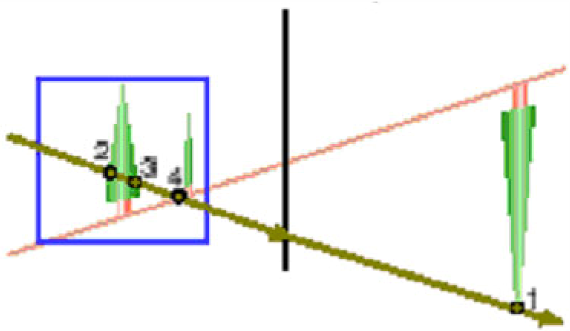
Press F1 for Help
PSM: yes   Depth Bias: 4   Slope Scale Bias: 1.00
InfinityZ: 1.50

Lights from Behind Viewer

- Significant problem with original PSM implementation
  - Objects behind viewer (w<0) cast shadows into scene
  - (w<0) is on opposite side of infinity plane



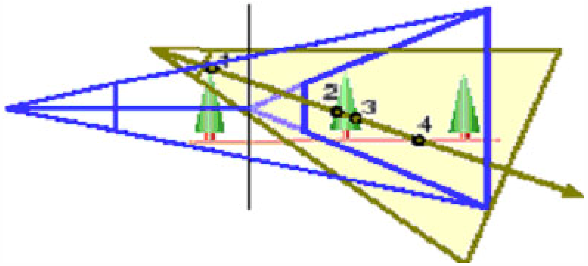
Eye-space



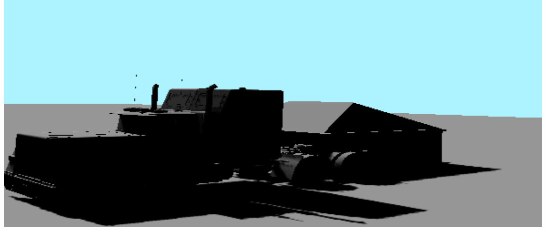
Post-projective space

Lights from Behind Viewer: Old Solution

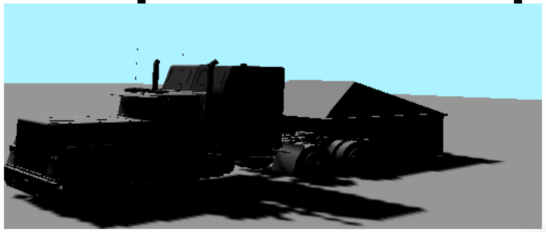
- Expand view volume
  - Keep all shadow casters on positive side of $Z_{infinity}$
  - “Slide back” virtual viewer to include all casters



- But increasing view volume decreases texel density
  - Large, instantaneous drop in shadow quality



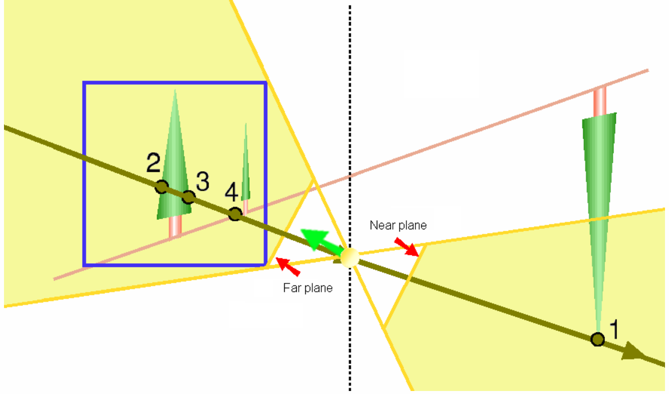
default



Slide back by Z_{near}

Lights from Behind Viewer: New Solution

- Shadow matrix looks at both sides of infinity plane
  - Near=`-a`, far =`a` (`a` = distance from light to view box)
  - Shadow projection “wraps around” infinity
  - Requires high-precision depth buffer (R32F, D24X8)



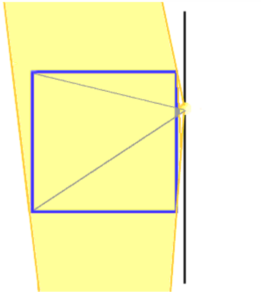
```
D3DXMatrixPerspectiveFovLH(
    &matrix,fovy,aspect,-a,a );
```

- No view volume expansion required
  - No instantaneous drops in shadow quality

View-Dependent Shadow Quality

- Light and scene transformed by `view*projection`
  - Relative post-projective position depends on viewer

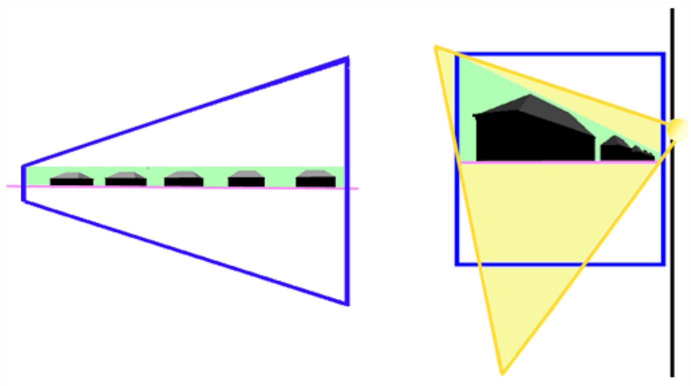
- When $Z_{infinity}$~1.0, shadow field-of-view ~180°
  - Happens when light is close to view-box



- This greatly reduces texel density
  - Increases shadow map perspective aliasing

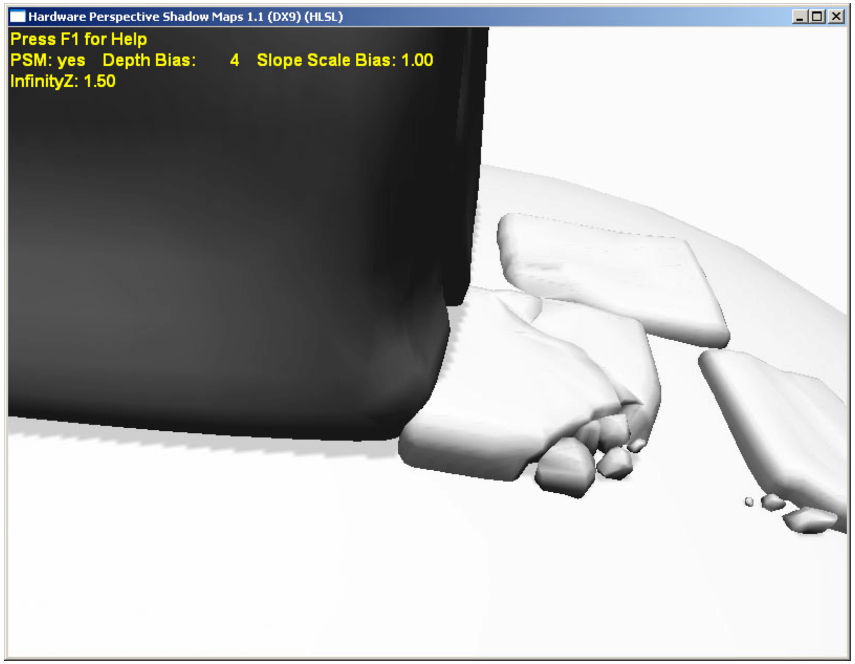
View-Dependent Shadow Quality

- Reduce artifacts by optimizing shadow frustum
  - Get bounding volume of shadow *receivers* in view
  - Build tight bounding frustum from this point list
  - Analogous to clipping the view box



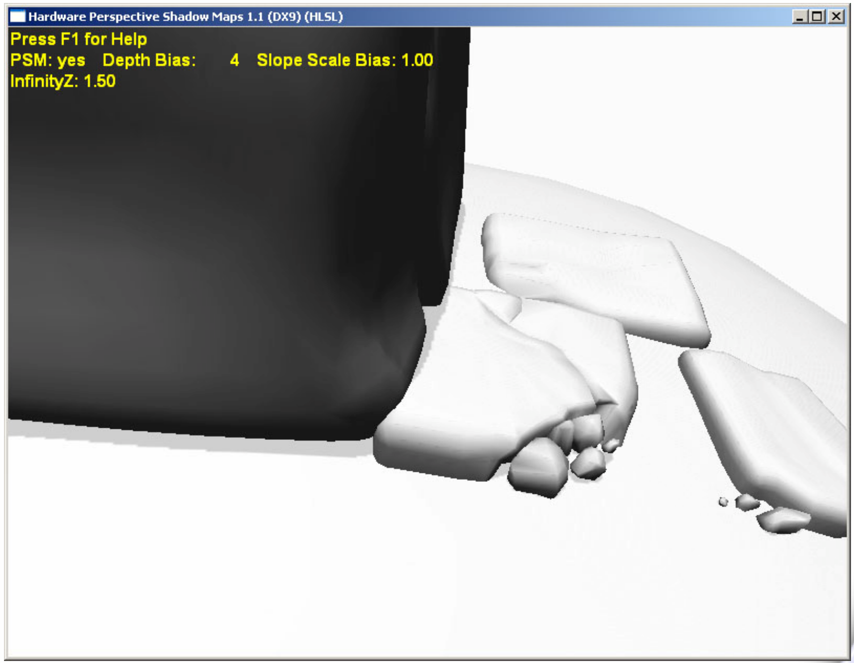
- Do not include shadow casters in this list
  - “Inverted” matrix will see everything

View-Dependent Shadow Quality



```
Press F1 for Help
PSM: yes   Depth Bias:     4   Slope Scale Bias: 1.00
InfinityZ: 1.50
```

Without clipping



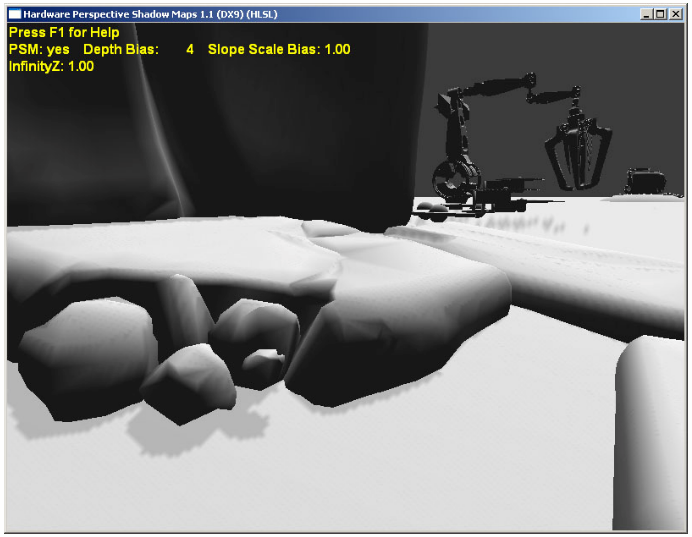
```
Press F1 for Help
PSM: yes   Depth Bias:     4   Slope Scale Bias: 1.00
InfinityZ: 1.50
```

With clipping

Near-Plane Dependence

- Post-projective Z distribution affects shadow quality
  - Great near camera, much worse far away

- If $Z_{infinity}$~1.0, quality in distance will be unacceptable
  - Happens when Z_{far}>>Z_{near}



Press F1 for Help
PSM: yes Depth Bias: 4 Slope Scale Bias: 1.00
InfinityZ: 1.00

Near-Plane Dependence: Old Solution

- Find optimal near plane position
  - Read depth buffer onto CPU, find nearest point
  - Or, use bounding volumes to approximate
- CPU read-back is a *bad* idea
  - Forces synchronization between CPU & GPU
  - D24X8 is an opaque format
- Bounding volumes often insufficient
  - In outdoor scenes, every 1m of Z_{near} helps

Near-Plane Dependence: New Solution

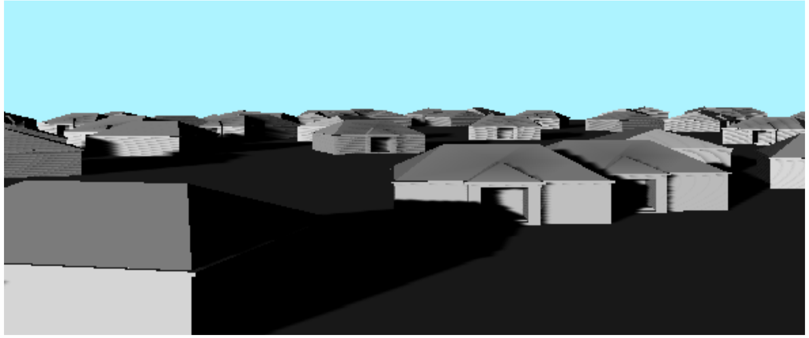
- “Virtually” slide back near plane
  - Translate “virtual” eye by $Z_{slideback}$
  - Move “virtual” eye plane forward by $Z_{slideback}$
  - Shrink virtual field-of-view
  - Increases view volume, but improves Z distribution
  - I choose $Z_{slideback}$ based on a fixed minimum for $Z_{infinity}$

```
View' = View * D3DXMatrixTranslate(0,0,Z_sb);
theta = max( atan(h_f/(f+Z_sb)), atan(h_n/(n+Z_sb)) );
D3DXMatrixPerspectiveFovLH(&Proj',2*theta),aspect,n+Z_sb,f+Z_sb);
```

- Good results without any scene analysis
  - Simple analysis can further improve quality

Self-Shadow Artifacts

- Simple constant bias is ineffective for PSMs



- Depth slope scale bias works great
  - But only applies to depth shadow maps (e.g., D24X8)
- Or, calculate bias in the vertex shader
  - Based on the texel size in world space

Summary

- Perspective Shadow Maps are (finally) useful
- Some CPU analysis is required for best results
  - But limited to bounding boxes and O(N) algorithms
- Use hardware shadow maps on NVIDIA GPUs
- This presentation focused on directional lights, PSMs are applicable to point lights, too
  - See original paper and Kozlov’s article for details

Questions

- Email: gking@nvidia.com
- Web: http://developer.nvidia.com

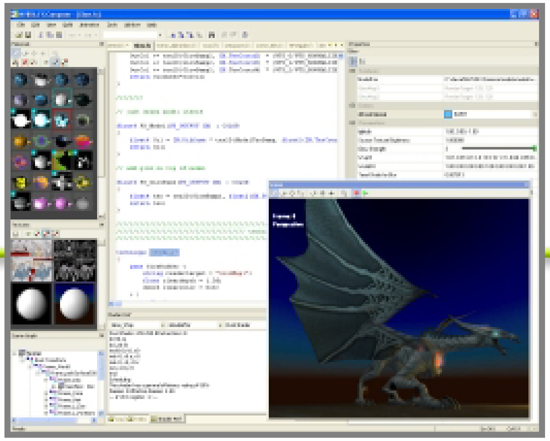
References

- Kozlov, S. Perspective Shadow Maps: Care and Feeding. *GPU Gems*, 2004
- Stamminger, M and G. Drettakis. Perspective Shadow Maps. *SIGGRAPH 02*, 2002
- Williams, L. Casting curved shadows on curved surfaces. *SIGGRAPH 78*, 1978

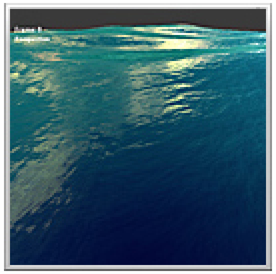
developer.nvidia.com

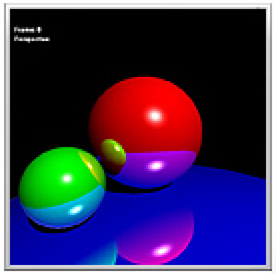
The Source for GPU Programming

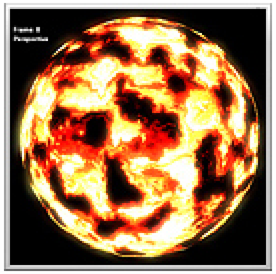
- Latest documentation
- SDKs
- Cutting-edge tools
  - Performance analysis tools
  - Content creation tools
- Hundreds of effects
- Video presentations and tutorials
- Libraries and utilities
- News and newsletter archives

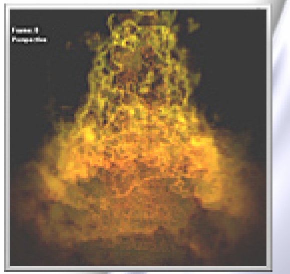


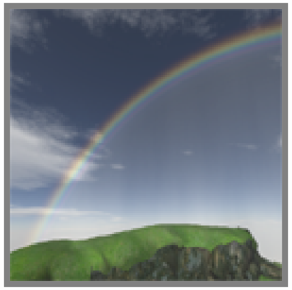
EverQuest® content courtesy Sony Online Entertainment Inc.

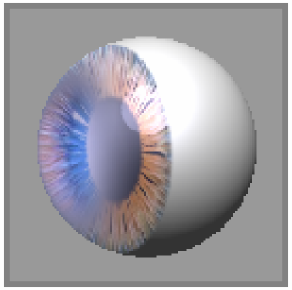




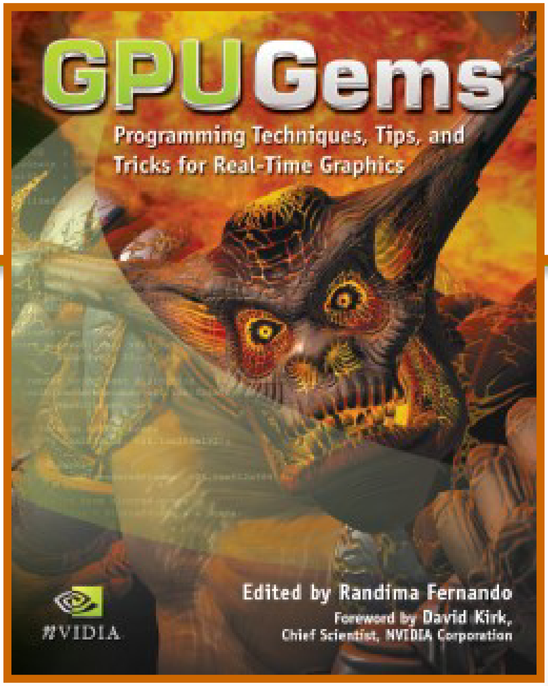








GPU Gems: Programming Techniques, Tips, and Tricks for Real-Time Graphics


GPU Gems
Programming Techniques, Tips, and Tricks for Real-Time Graphics
Edited by Randima Fernando
Foreword by David Kirk, Chief Scientist, NVIDIA Corporation
NVIDIA

- Practical real-time graphics techniques from experts at leading corporations and universities
- Great value:
  - Contributions from industry experts
  - Full color (300+ diagrams and screenshots)
  - Hard cover
  - 816 pages
  - Available at GDC 2004

For more, visit: http://developer.nvidia.com/GPUGems

"*GPU Gems* is a cool toolbox of advanced graphics techniques. Novice programmers and graphics gurus alike will find the gems practical, intriguing, and useful."

Tim Sweeney

Lead programmer of *Unreal* at Epic Games

"This collection of articles is particularly impressive for its depth and breadth. The book includes product-oriented case studies, previously unpublished state-of-the-art research, comprehensive tutorials, and extensive code samples and demos throughout."

Eric Haines

Author of *Real-Time Rendering*