Programming Graphics Hardware

GPU Applications

Randy Fernando, Cyril Zeller

Overview

- Per-Pixel Displacement Mapping with Distance Functions
- Percentage-Closer Soft Shadows
- Introduction to General-Purpose Computation on GPUs
- Cloth Simulation on GPU

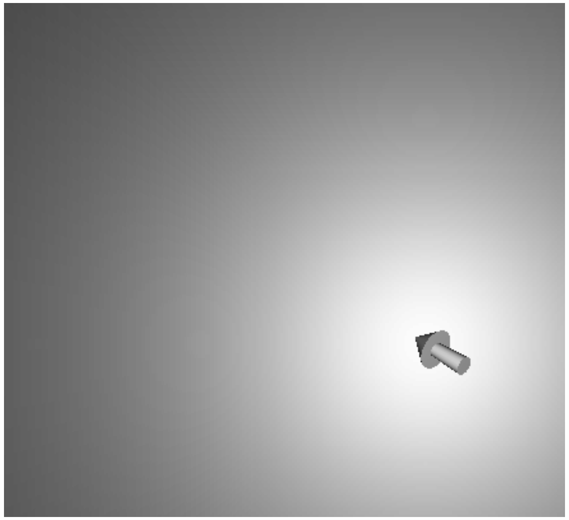
Programming Graphics Hardware

Per-Pixel Displacement Mapping with Distance Functions

Cyril Zeller

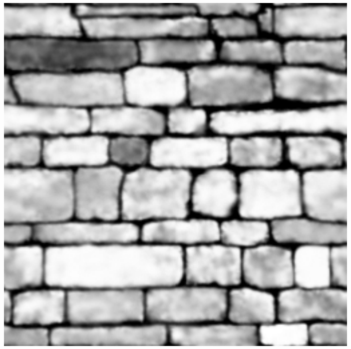
Goal

- Adding small-scale geometric details stored as a texture:
  - To reduce memory
  - To simplify authoring
- Applications: wall, grating, fence, etc.



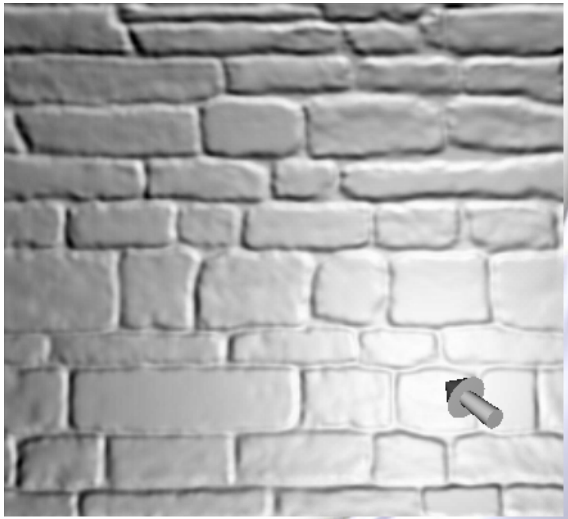
Diffuse light without bump

+



Height map

=



Diffuse light with bumps

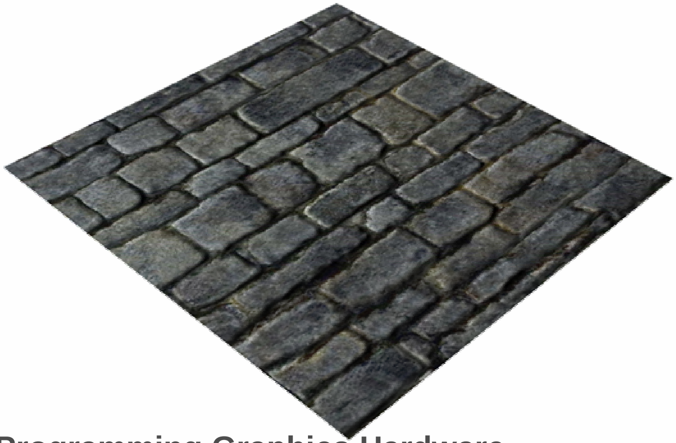
Traditional Methods

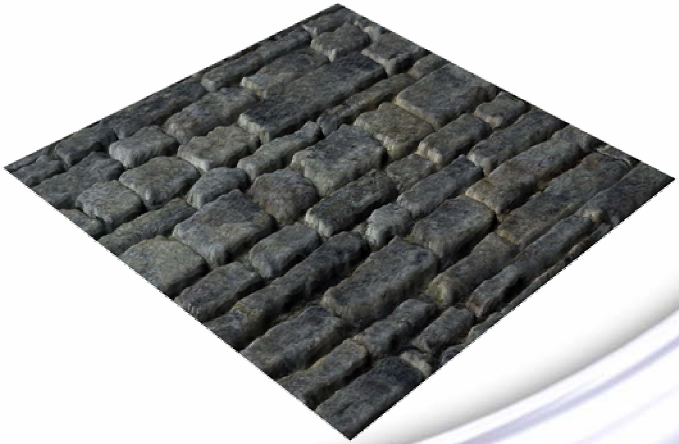
- Displacement mapping [Cook]:
  - Iteratively tessellate the mesh based on the height map

  Caveat: Requires multiple rendering passes and VTF [Bunnell]
- Bump mapping [Blinn]:
  - Shade using the normals computed from the height map

  Caveat: Bumps don’t occlude each other

Bump mapping

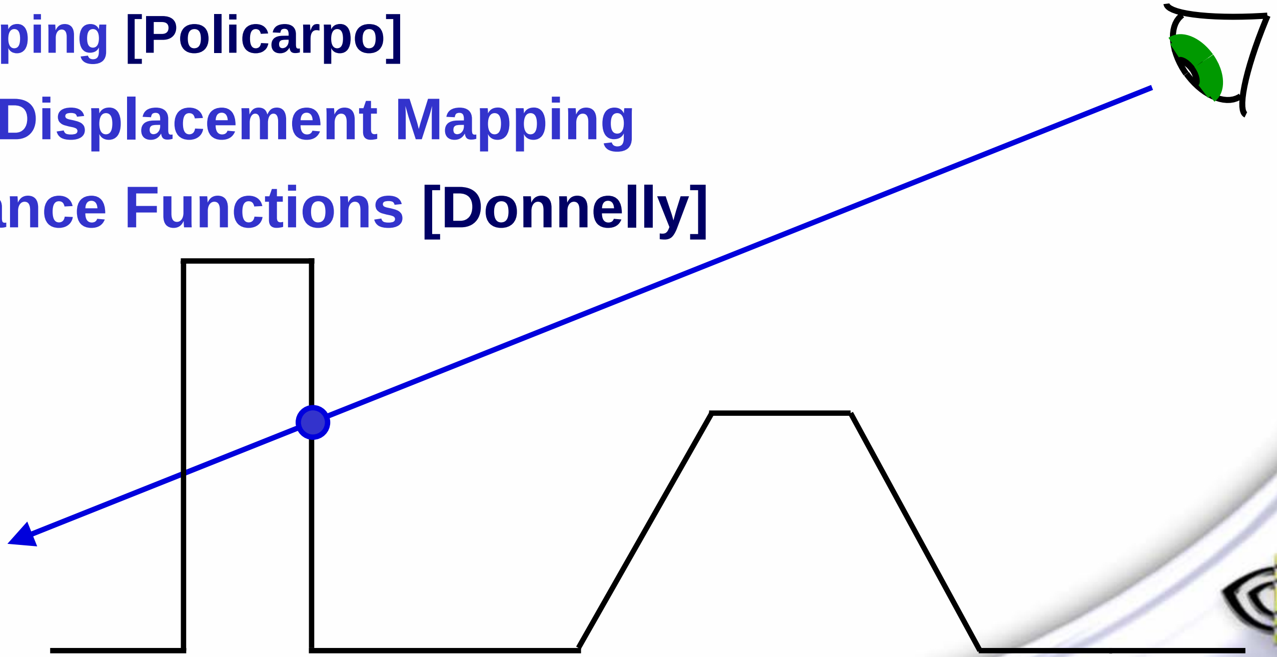




Per-pixel displacement mapping

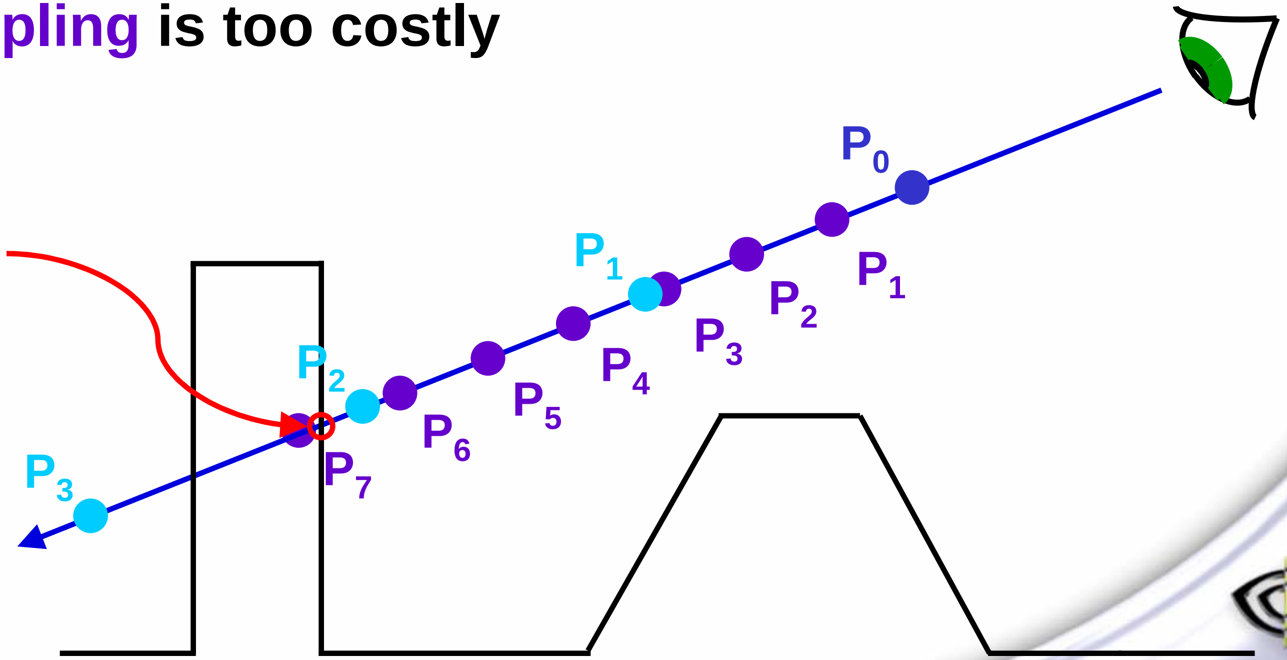
Ray Tracing Based Methods

- To handle bump occlusion, we need to intersect the view vector with the height field:
  - View-dependent displacement mapping [LWang]
  - Generalized displacement mapping [XWang]
  - Parallax mapping [Kaneko, Welsh]
  - Relief mapping [Policarpo]
  - Per-Pixel Displacement Mapping with Distance Functions [Donnelly]



Finding the Intersection: Uniform Sampling

- One way of finding the intersection is to sample the height map at uniformly spaced locations
- But:
  - Undersampling is too risky (missed intersections)
  - Oversampling is too costly



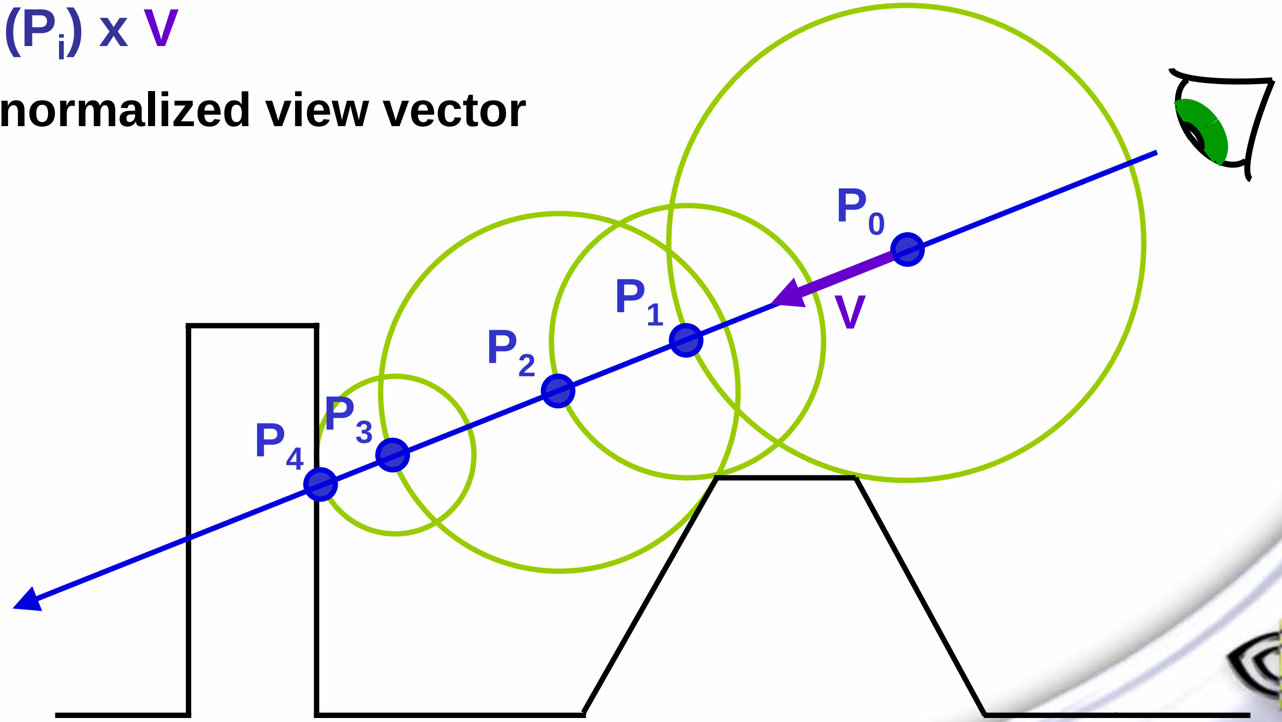
Missed when undersampling!

Finding the Intersection: Sphere Tracing

- Sphere tracing is used to accelerate raytracing of implicit surfaces [Hart]
- We pre-compute a distance function d(P) that maps any 3D point P to its distance to the height field *H*: d(P) = distance(P, *H*)
- Then at run time, we step along the ray using the following formula:

$$P_{i+1} = P_i + d(P_i) \times V$$

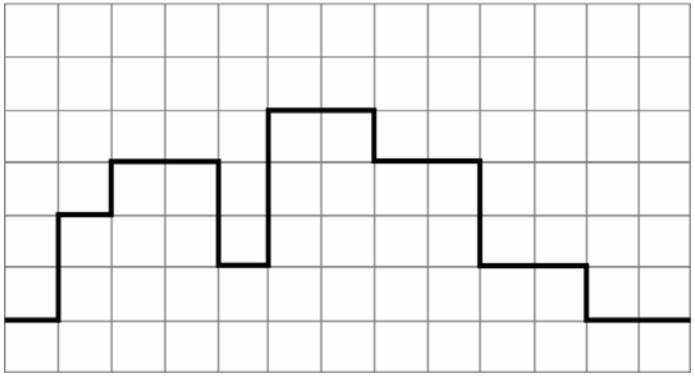
where V is the normalized view vector



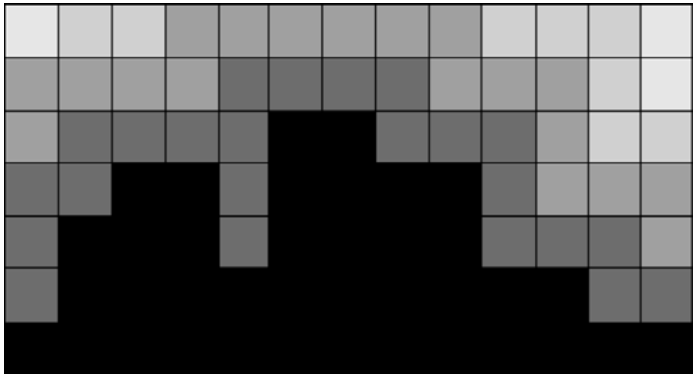
Distance Map

- The distance function is stored in a “thin” 3D texture (e.g. H_{width} x H_{height} x 16) called a distance map
- The computation of the distance map is based on [Danielsson]

1	3	4	4	2	5	5	4	4	2	2	1	1



A sample 1D height field

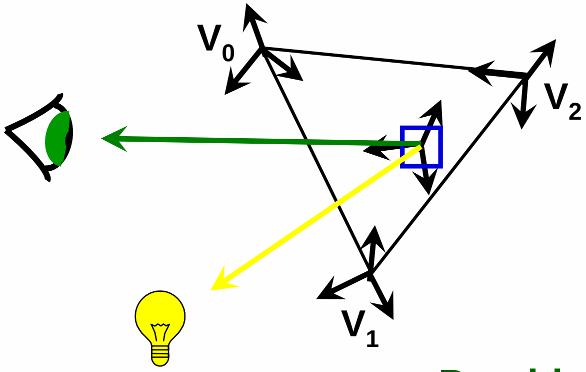


The corresponding 2D distance map

Vertex Shader

```
void VertexShader(
  float4 position, // In model space
  float3x3 modelToTangent, // Local tangent, normal and binormal
  float4 eye, // In model space
  float4 light, // In model space
  uniform float4x4 modelToProjection,
  out float4 positionProj,
  inout float2 texCoord,
  out float3 viewVector,
  out float3 lightVector
)
{
  // Transform position to projection space
  positionProj = mul(modelToProjection, position);

  // Transform view and light vectors to tangent space
  viewVector = mul(modelToTangent, eye - position);
  lightVector = mul(modelToTangent, light - position);
}
```



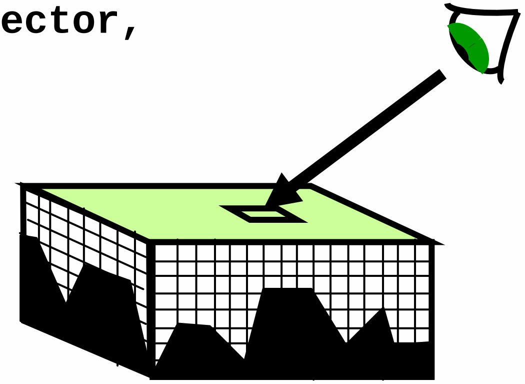
Provide the pixel shader with the view and light vectors in tangent space

Pixel Shader

```
float4 PixelShader(
  float2 texCoord, float3 viewVector, float3 lightVector,
  uniform sampler2D colorMap,
  uniform sampler2D normalMap,
  uniform sampler3D distMap
)
{
  // Normalize interpolated vectors
  viewVector = normalize(viewVector);
  lightVector = normalize(lightVector);

  // Find intersection with height field,
  // assuming the surface is locally planar
  float3 point = float3(texCoord, 1);
  for (int i = 0; i < NUM_ITERATIONS; ++i)
    point += tex3D(distMap, point) * viewVector;

  // Compute final color
  float4 color = tex2D(colorMap, point.xy);
  float3 normal = tex2D(normalMap, point.xy);
  return dot(normal, lightVector) * color;
}
```



Start at the top of the height field

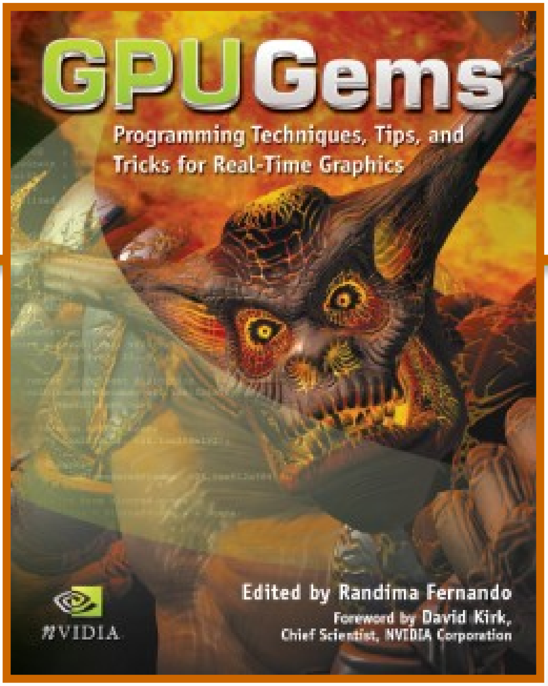
Performance

- `NUM_ITERATIONS` depends on
  - Distance map resolution
  - Smoothness of data
- For the demo (256x256x16), using 16 iterations is more than enough
- Note that each iteration is `{tex; mad;}`, which runs in a single cycle on GeForce FX and GeForce 6
- On a GeForce 6800 GT, the previous shader runs at around 70 fps on 1280 x 1024 pixels with 16 iterations

References

- James **Blinn**. "Simulation of Wrinkled Surfaces", *SIGGRAPH 1978*
- Per-Erik **Danielsson**. "Euclidean Distance Mapping", *CGIP 1980*
- Robert **Cook**. "Shade Trees", *SIGGRAPH 1984*
- John **Hart**. "Sphere Tracing: A Geometric Method for the Antialiased Ray Tracing of Implicit Surfaces", *The Visual Computer, 1996*
- Tomomichi **Kaneko**, et al. "Detailed Shape Representation with Parallax Mapping", *ICAT 2001*
- Lifeng **Wang**, et al. "View-Dependent Displacement Mapping", *SIGGRAPH 2003*
- Xi **Wang**, et al. "Generalized Displacement Maps", *EG 2004*
- Terry **Welsh**. "Parallax with Offset Limiting: A Per-Pixel Approximation of Uneven Surfaces"
- Fabio **Policarpo**. "Relief Mapping in a Pixel Shader Using Binary Search"
- Michael **Bunnell**. "Adaptive Tesselation of Subdivision Surfaces with Displacement Mapping", *GPU Gems 2, 2005*
- William **Donnelly**. "Per-Pixel Displacement Mapping with Distance Functions", *GPU Gems 2, 2005*

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:**
  - **Full color (300+ diagrams and screenshots)**
  - **Hard cover**
  - **816 pages**
  - **CD-ROM with demos and sample code**

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

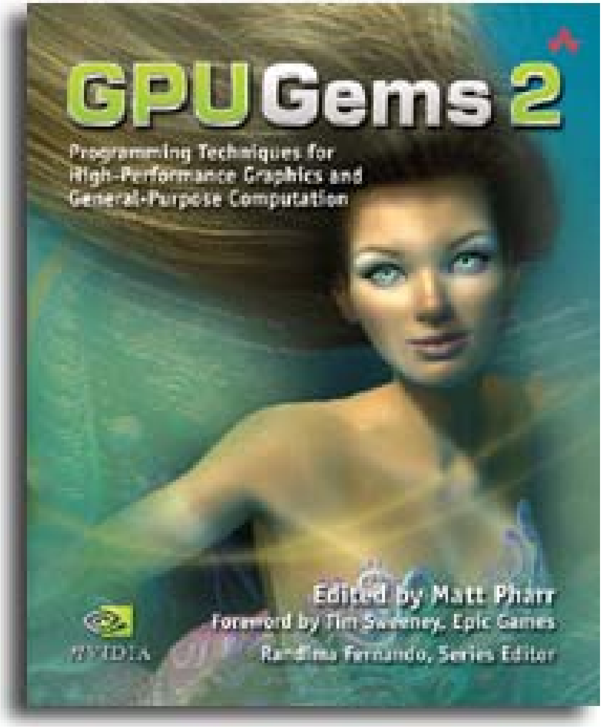
"This collection of articles is particularly impressive for its depth and breadth. The book includes product-oriented case studies, previously unpublished state-of-the-art research, comprehensive tutorials, and extensive code samples and demos throughout."

Eric Haines

Author of *Real-Time Rendering*

GPU Gems 2

Programming Techniques for High-Performance Graphics and General-Purpose Computation


GPU Gems 2
Programming Techniques for High-Performance Graphics and General-Purpose Computation
Edited by Matt Pharr
Foreword by Tim Sweeney, Epic Games
Randima Fernando, Series Editor
NVIDIA

- **880 full-color pages, 330 figures, hard cover**
- **$59.99**
- **Experts from universities and industry**

"The topics covered in *GPU Gems 2* are critical to the next generation of game engines."

— Gary McTaggart, Software Engineer at Valve, Creators of Half-Life and Counter-Strike

"*GPU Gems 2* isn't meant to simply adorn your bookshelf—it's required reading for anyone trying to keep pace with the rapid evolution of programmable graphics. If you're serious about graphics, this book will take you to the edge of what the GPU can do."

—Rémi Arnaud, Graphics Architect at Sony Computer Entertainment