NVIDIA®

BloodShader

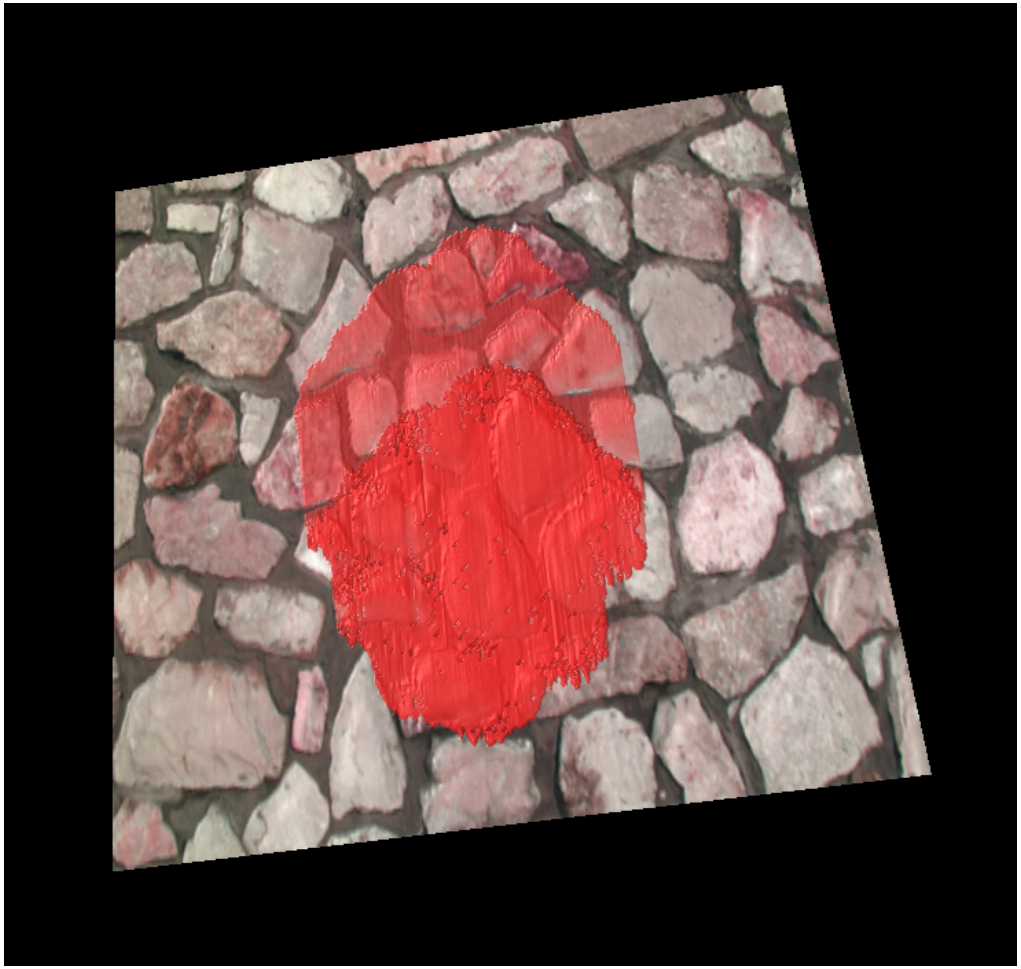
Real-Time Adaptive Animation

Kevin Myers

Overview

- Demo
- Per-pixel physics
- Normal Maps as Gravity Maps
- Oozing
- Demo
- Questions

Demo



Per-Pixel Physics

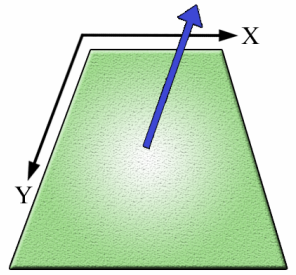
- Dot3 bump mapping gives us per-pixel lighting
- Side effects of increased aesthetic realism
  - Per-Pixel control
    - How will the user effect this pixel's color?
  - More accurate model of real life physics
- GPU can now be used for physics

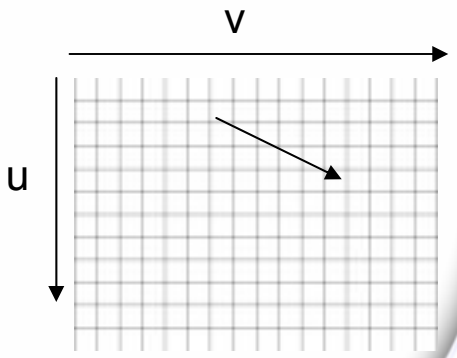
Animation and the GPU

- Per-pixel animation
- Need surface details defined in a normal map
- Surface normals and gravity define surface movement
  - Object moves along the surface in the direction of both gravity and the normal
- Write resulting 2D vector out to a texture

Normal Maps as Gravity Maps

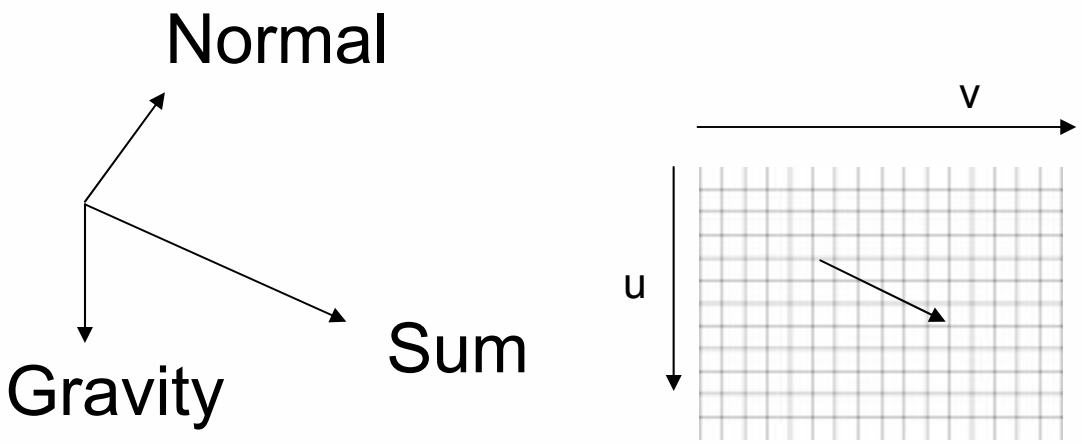
- We only care about tangent space
  - Ignore the z-component
- Normal Map can then be viewed as a gravity map
  - Gravity map defines gravity's pull in tangent space
  - X and Y components individually describe gravitational force at each point





World Of Gravity

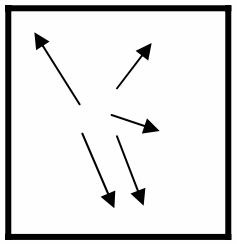
- Gravity is a constant vector in World Space
  - Move Gravity into tangent space
- 2D vector Normal.xy + Gravity.xy defines actual gravitational pull
- Resulting force follows normal map contours
- “Viscosity” is increased by raising gravity to a power



NVIDIA.

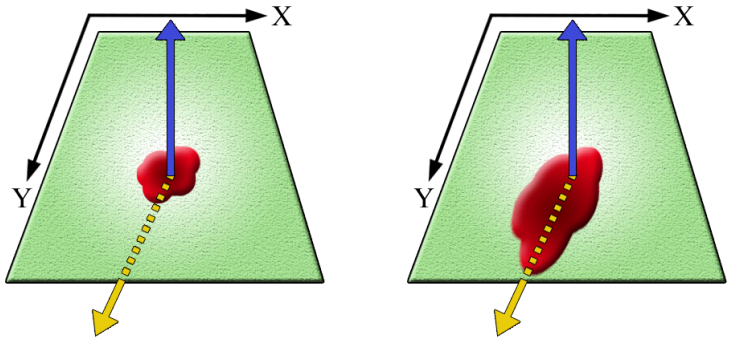
Oozing

- Gravity map now defines 2D directional movement
- Fluid is stored in the unused z component of the map
- Every frame
  - Read texel's gravity direction
  - Compute fluid lost based on direction
  - Subtract value from height
- The final sum is the amount of fluid that the texel will keep this frame



Oozing Some More

- Check for incoming fluid
  - Sample four nearest neighbors
  - Check for gravitational force pointing in this direction
    - i.e. Check the texel above
      - If y < 0 then abs(y) * B gives fluid contribution
- Net result animates the fluid

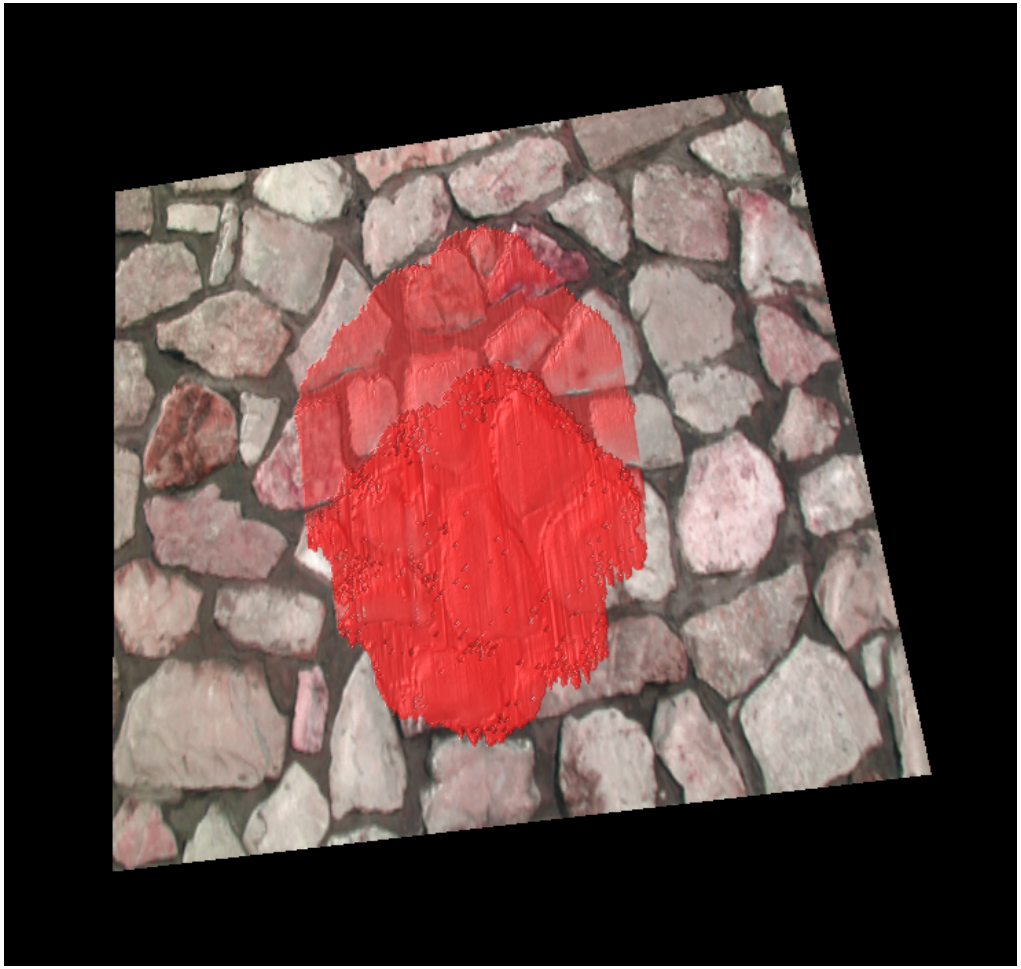


Eye Candy

- Compute height deltas for the fluid
  - dx = Right – Left
  - dy = Below – Above
  - Add these values to the surface normal
  - This gives a thickness to the fluid
- Final color =

  BloodHeight * BloodColor + SurfaceColor * (1 – BloodHeight)

Demo

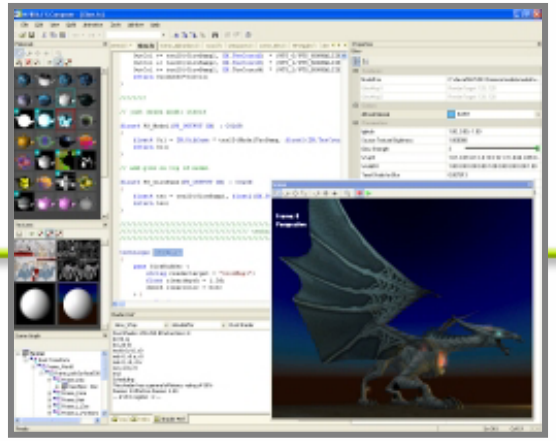


Questions?

- kmyers@nvidia.com
- http://developer.nvidia.com

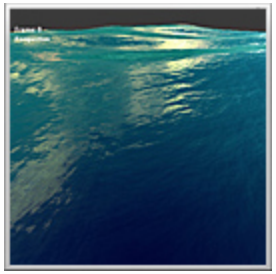
developer.nvidia.com

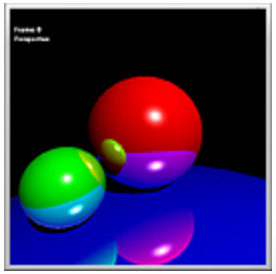
The Source for GPU Programming

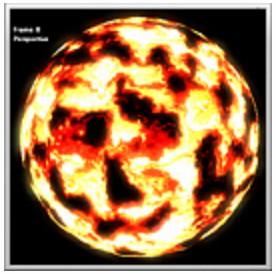


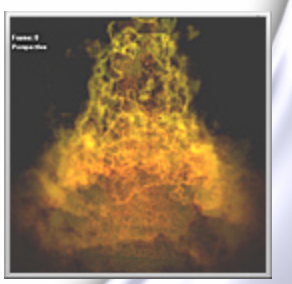
EverQuest® content courtesy Sony Online Entertainment Inc.

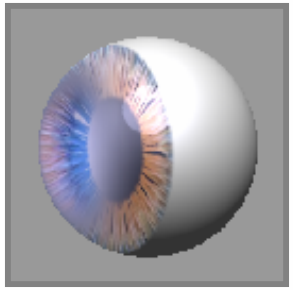
- Latest documentation
- SDKs
- Cutting-edge tools
  - Performance analysis tools
  - Content creation tools
- Hundreds of effects
- Video presentations and tutorials
- Libraries and utilities
- News and newsletter archives

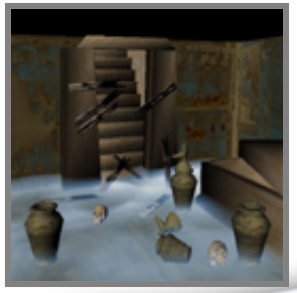






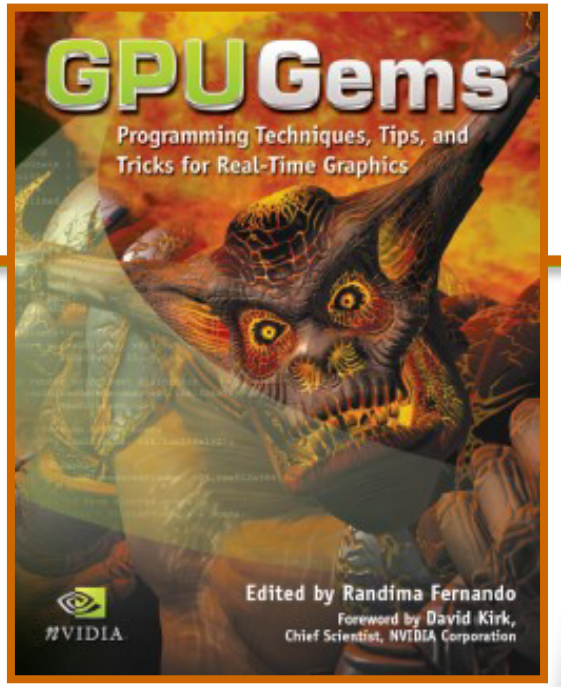






NVIDIA.

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



- 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*