NVIDIA

Special Effects in Direct3D

Greg James

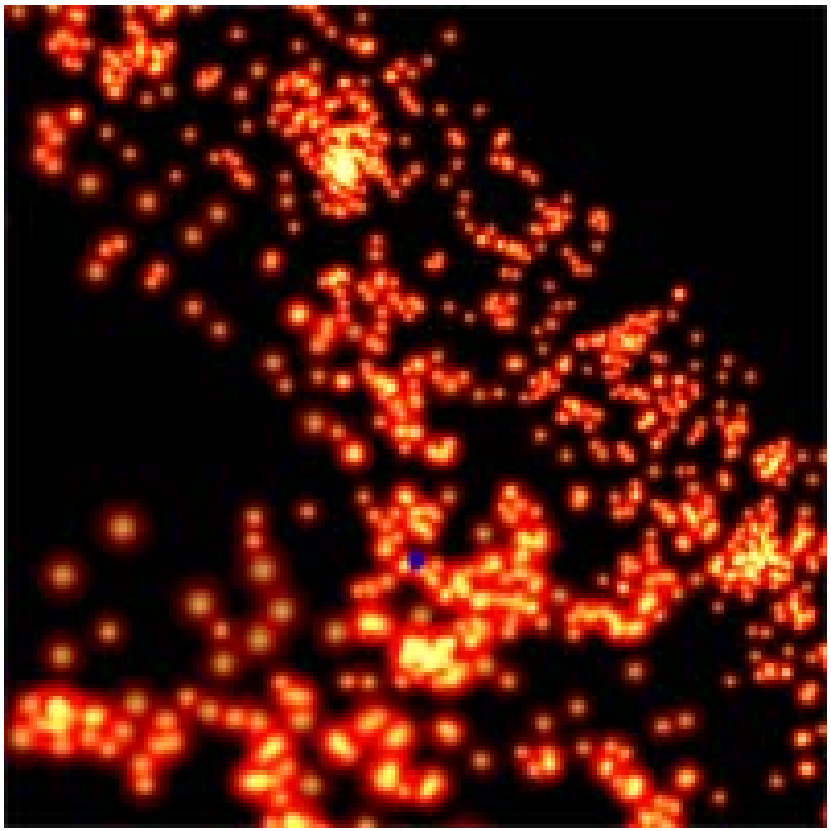
NVIDIA Corporation

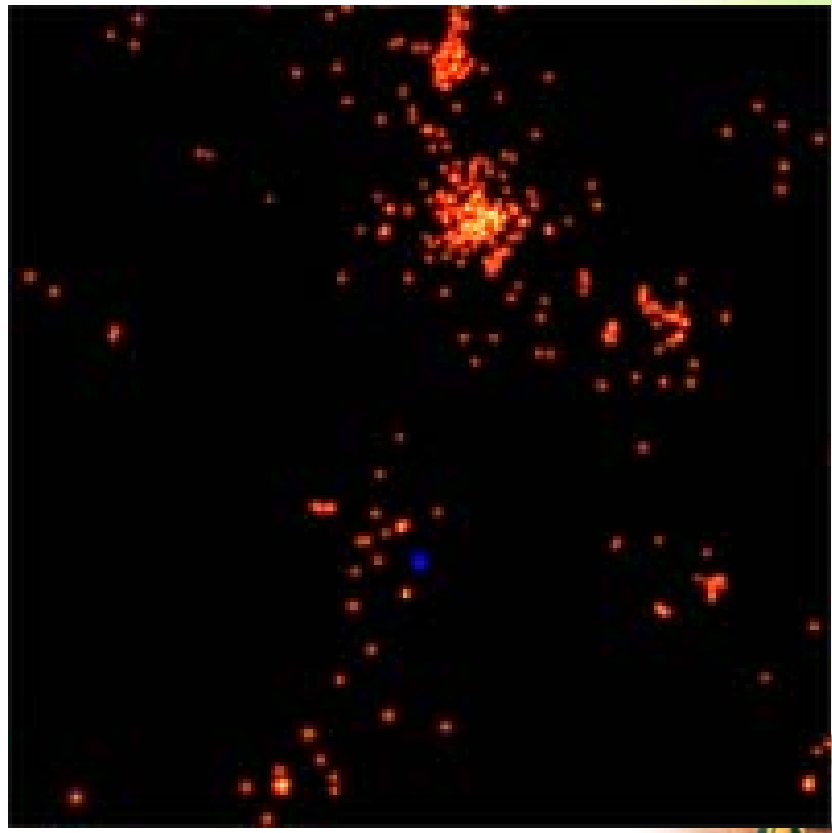
Agenda & Audience

“Good artists copy. Great artists steal.” – Pablo Picasso

- **An overview of recent D3D9 GPU work**
  - **Particle systems** (5 min)
  - **Cloth simulation** (10 min)
  - **Using anti-aliasing with post-processing** (7 min)
  - **Aniso filtering for decimation** (3 min)
  - **Ambient occlusion** (10 min)
    - **real-time!**
- **Audience**
  - **Programmers, designers**

GPU Particle Systems





GPU Particle Systems

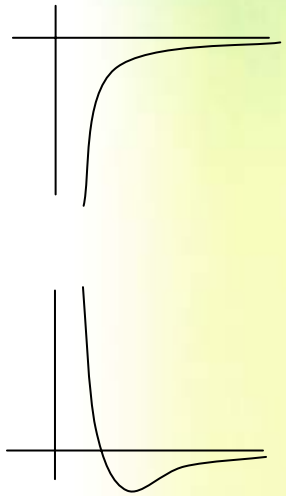
- Requires
  - Direct3D vs.3.0, ps.3.0
  - Vertex texture fetch
  - Floating-point render target textures (RTTs)
    - fp16, fp32
- Textures hold DATA
- Particle state held in rendered textures
  - Position, velocity, force
  - Encoded in R, G, B, A channels
  - 1 texel per particle, or N texels per particle

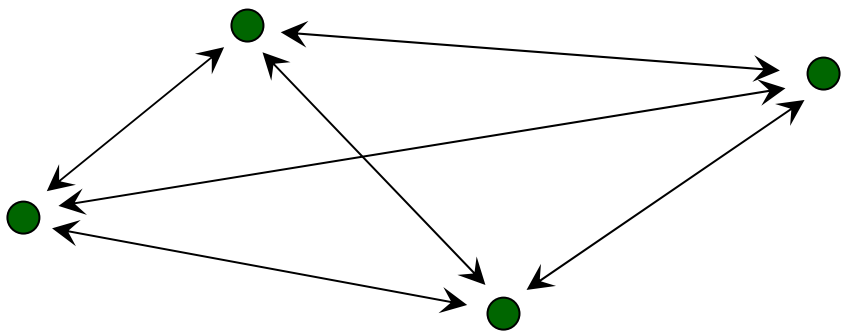
GPU Particle Systems

- **Render-to-texture updates particles**
  - Samples particle state from textures
  - Computes forces, new state
  - Writes new state to textures
  - New state used in next step of simulation
  - No CPU work!
- **N-body forces**
  - All particles attract eachother
  - Complex gravitational interactions
- **Particle attractor systems**
  - force = func(position)
  - func() via math or textures

N-body Forces

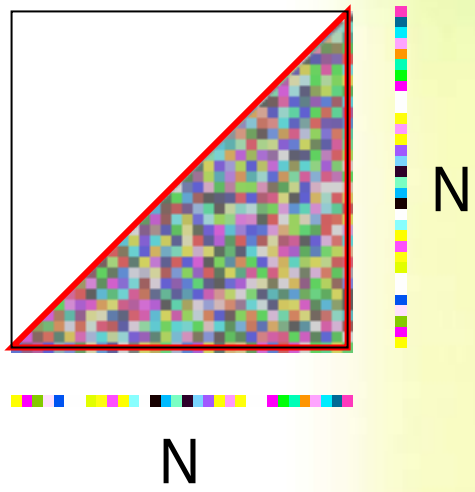
- Every particle attracts every other particle
- Gravitation: $F = -k M_1 M_2 / (dist^2)$
- Gravitation & repulsion: $k/d^4 - k/d^2$

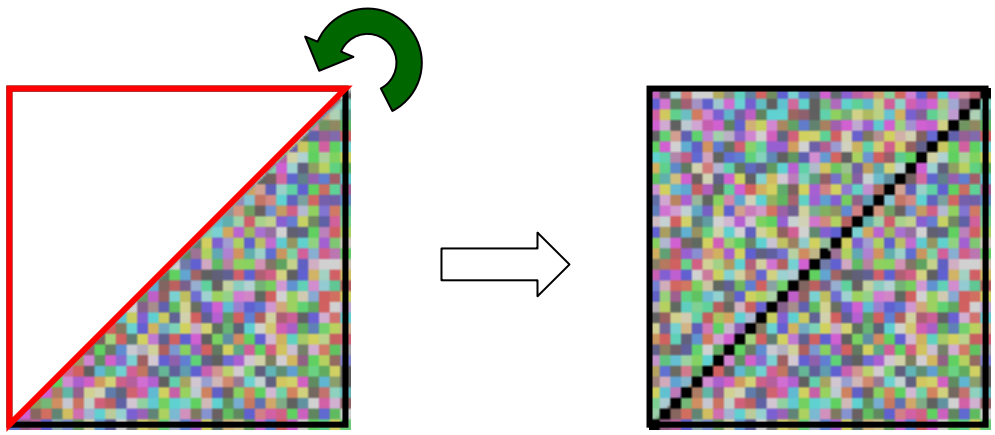




N-body Forces

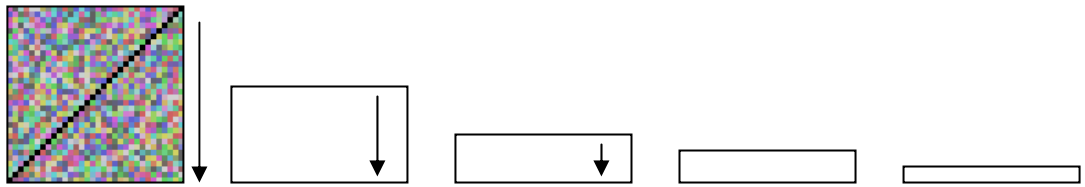
- Forces between N particles rendered to NxN fp16 texture
- Render 1 triangle
  - Interacts all particles!
- Force(a,b) = -force(b,a)
  - Flip a copy * -1 to fill in the rest





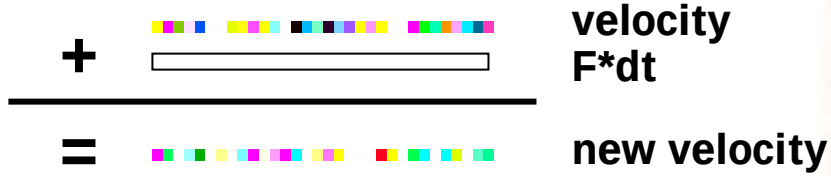
N-body Forces

- Sum all forces using reduction
- Additive blend all rows



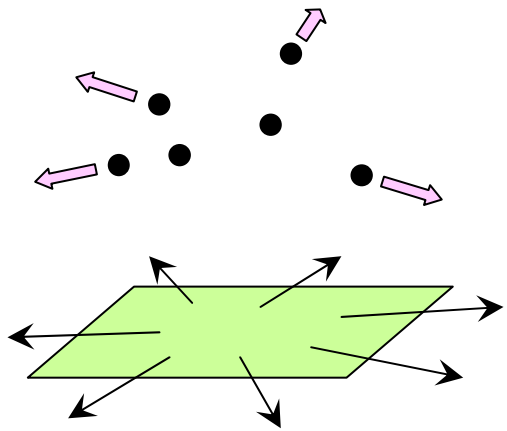
N forces

- Add forces to particles
  - Nx1 or NxM textures



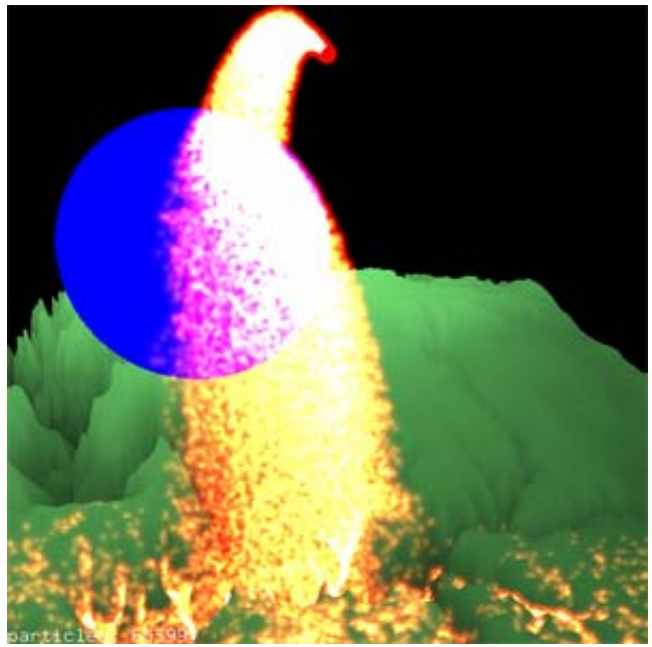
Particle Attractor Systems

- Force(i)= func(pos(i), vel(i), mass(i), etc.(i))
- No particle-particle interaction
- Force calculated with
  - pixel shader math
  - texture lookups
- Can handle a lot of particles



Lots of particles!

- n-body : 2048 30 fps
- attractor : 1,000,000 20 fps
- Terrain & geometric primitive collision
  - planes, boxes, cylinders, ellipsoids



65535 particles

More Information

- Mark Harris, Simon Green
- Talk: Tues. ~3pm
- References coming soon!

GPU Cloth Simulation

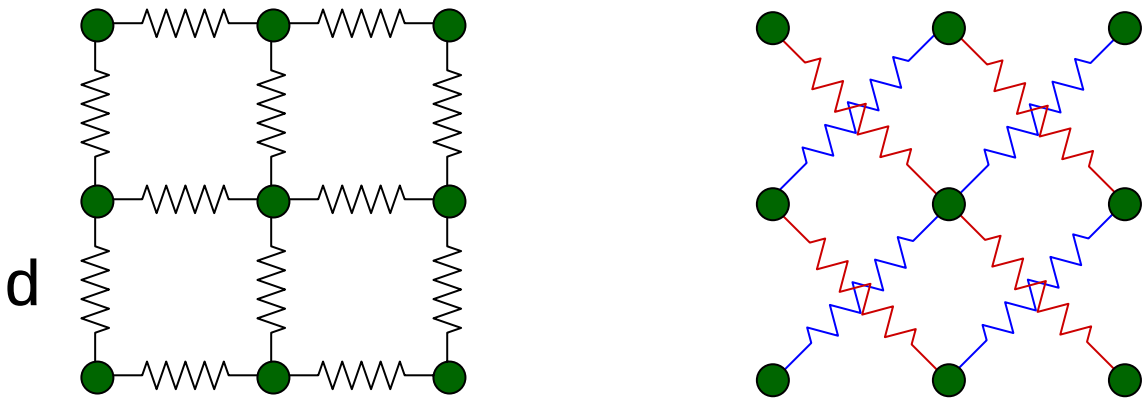
- Cyril Zeller
- DEMO

GPU Cloth Simulation

- Requires
  - Direct3D vs.3.0, ps.3.0
  - Vertex texture fetch
  - Floating-point render targets (fp32)
- 70 fps, 3 simulations
  - 32x32, 32x32, 26x14
- No CPU vertex buffer work!
- Linked particles

Cloth Simulation Model

- **Cloth: array of nodes joined by springs**



- **Interaction between neighboring nodes**
  - **spring forces**
  - **distance constraints: d, sqrt(2d²)**

Verlet Integration

- Less costly than Runge-Kutta
- More stable than Euler
- $P_{t+dt} = P_t + k*(P_t - P_{t-dt}) + dt^2*F(t) / mass$

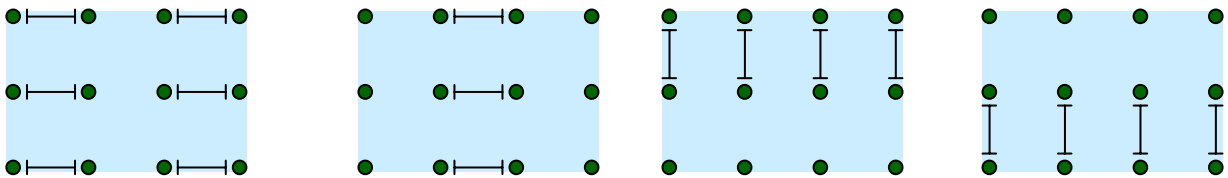
- Relaxation used to converge to state where all constraints are satisfied

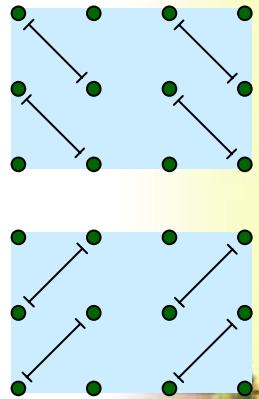
Cloth Simulation Data

- Cloth state in 3 render-target textures
  - node positions, time t : 1 fp32
  - node positions, time t-1 : 1 fp32
  - node positions, time t+1: 1 fp32
- Cloth of WxH nodes held in texture that is WxH texels
- Render a quad to update all nodes
- Texture coordinates sample neighbors at each pixel rendered

Cloth Simulation Steps

- Apply forces, gravity, wind
  - moves particles
- Set anchors (constrain fixed particles)
- Spring force & node distance constraints
  - rows & columns in pairs, many steps





- Environment constraints
- Render cloth mesh using Vertex Tex Fetch

Additional Info

- Cyril's docs
  - SDK\DEMOS\Direct3D9\src\Cloth\Docs\Cloth.pdf
- Cyril's demo
  - SDK\DEMOS\Direct3D9\src\Cloth
- Jakobsen, T. "Advanced Character Physics" Game Developer's Conference 2001
  - www.gdconf.com/archives/2001/jakobsent.doc
- Literature…

Anti-aliasing with Post-processing

Elder Scrolls : Oblivion

Courtesy of Bethesda Softworks

http://www.elderscrolls.com/art/obliv_screenshots_01.htm

Post Processing

- It's hot!
- Full-screen glow
  - A8R8G8B8, fp16, or fp32 HDR
  - Wreckless, Tron 2.0, World of Warcraft, EverQuest2, Elder Scrolls: Oblivion, Far Cry, UE3, Halo 2, S.T.A.L.K.E.R., ...
- Video effects
  - TV noise, night-vision, sepia tone, B&W
- Render-to-texture 2D image processing

Carsten's Work (Stick around!)

Far Cry

Carsten Wenzel, Crytek www.crytek.com

Academic Foundations

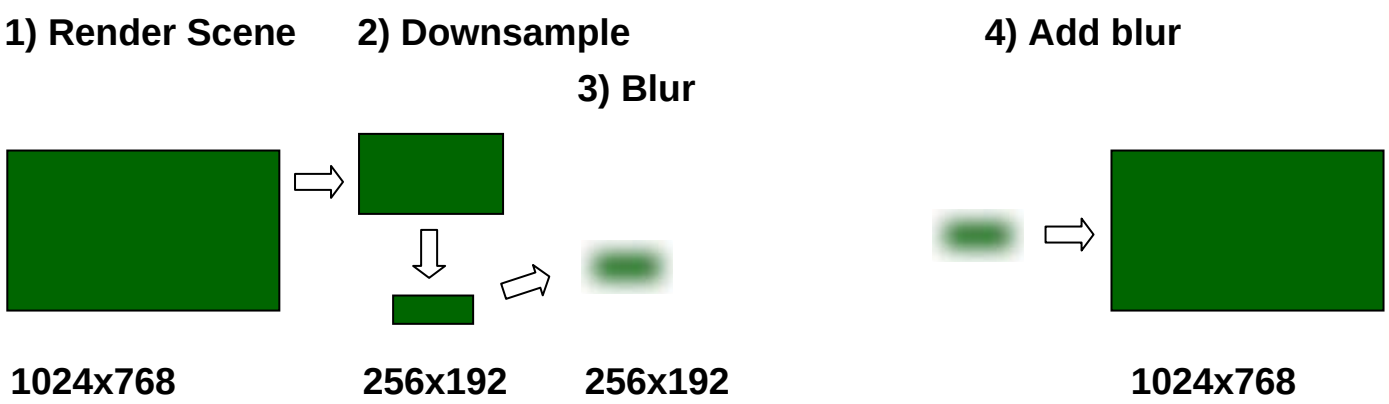
- Glow & light halos caused by
- Scattering from the atmosphere (poetic sense)
- Scattering in the eye
- G. Spencer, P. Shirley, K. Zimmerman, D. Greenberg, “Physically Based Glare Effects for Digital Images.” SIGGRAPH 95, pp. 325-334
- Complex physical models
  - All we’re after is the effect – hello!
  - Approximate

Anti-aliased RTTs? No way!

- Anti-aliased RTTs are not allowed
- AA not supported for fp16, fp32 targets
- How to get AA with post-processing?

Example

- **Post-process a scene for full-screen glow**



Temptation

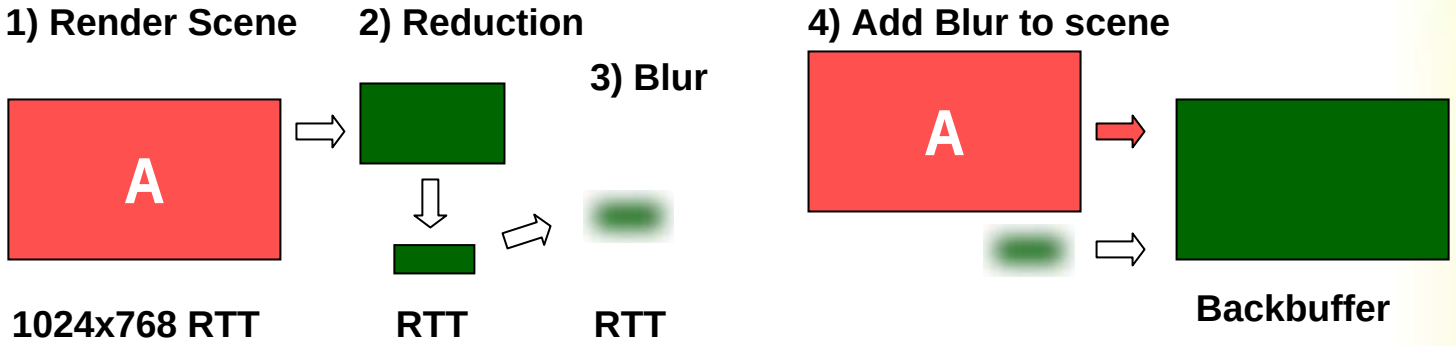
- Render the scene to a RTT
  - Can't use anti-aliasing here
- Downsample texture to smaller RTTs
- Process the small RTTs
- Composite result into the scene
  - The full resolution RTT
- Render the full res RTT to the backbuffer

The Righteous Path

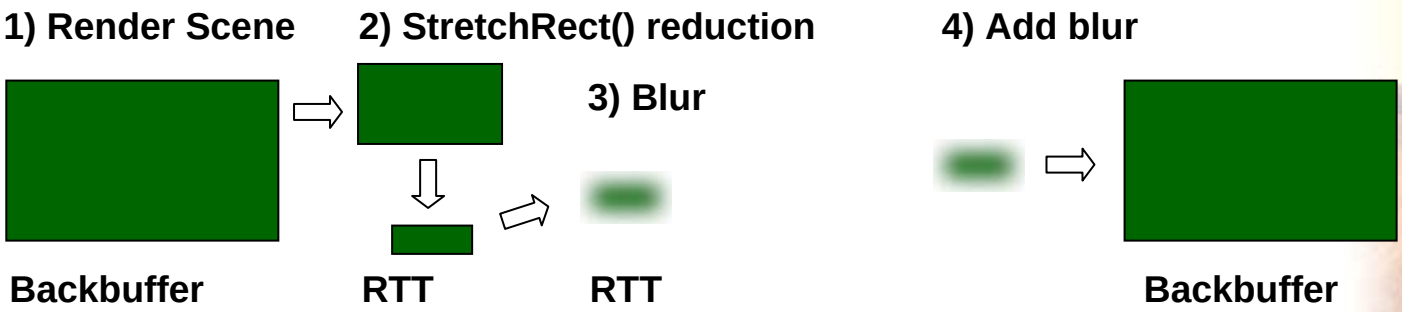
- Render first to the backbuffer
- Copy backbuffer to a smaller RTT
  - IDirect3DDevice9::StretchRect (..)
  - Resolves anti-aliasing in the RTT
- Use the RTT as source for post-processing
- This is always better

Post-processing for Glow

- **Wrong way: Render to full res RTT**



- **Right way: Render to backbuffer**



Why is it better?

- Rendering first to the backbuffer
  - Enables anti-aliasing
  - Uses anti-aliasing HW
    - No need for do-it-yourself supersampling
  - Does not require a full-resolution RTT
  - Avoids full-resolution copy from RTT to backbuffer
- StretchRect(..) is fast
  - As fast as ordinary render-to-texture

Resolving Anti-aliasing

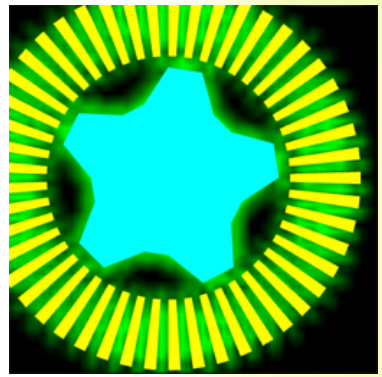
- Multisample AA held in bloated surface
- If src and dest are same size
  - MxN 4xAA → MxN RTT
  - StretchRect() resolves AA
  - AA target is downsampled, dest pixels have AA
  - Additional rendering to dest is NOT AA'd
- If src and dest are different size
  - MxN 4xAA → M / 2 x N / 2 RTT
  - StretchRect() may or may not resolve
  - Use bilinear to make up for it

Demos

http://download.developer.nvidia.com/developer/SDK/Individual_Samples/samples.html

- **Anti-aliasing with post-processing**

  http://download.developer.nvidia.com/developer/SDK/Individual_Samples/samples.html#AntiAliasingWithPostProcessing



- **Glow**

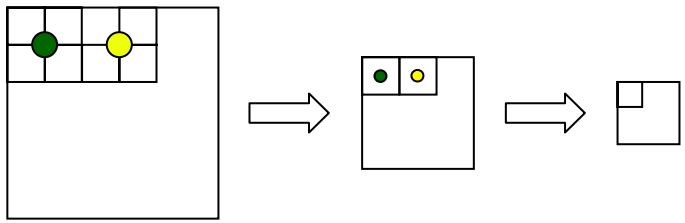
  http://download.developer.nvidia.com/developer/SDK/Individual_Samples/samples.html#Glow

Gamey References

- Greg James, “Special Effects with Direct3D 9,” Game Developer’s Conference 2003, also developer.nvidia.com
- Masaki Kawase, “Frame Buffer Postprocessing Effects in DOUBLE-S.T.E.A.L (Wreckless),” Game Developer’s Conference 2003
  - HDR demo: http://www.daionet.gr.jp/~masa/
- Greg James, “Real-Time Glow : Tron 2.0 and Beyond,” Microsoft Meltdown 2003, also developer.nvidia.com
- Carsten Wenzel, “Far Cry and DirectX,” Game Developer’s Conference 2005
- Literature…
  - References coming soon

Anisotropic Filtering for Decimation

- Gary King, NVIDIA
- Part of full-screen glow is decimation
  - Reduce NxM scene to N/4 x M/4 texture
  - Blur the N/4 x M/4 texture
- You could take several bilinear samples
  - Texture cache suffers



- 30% speedup if you use aniso

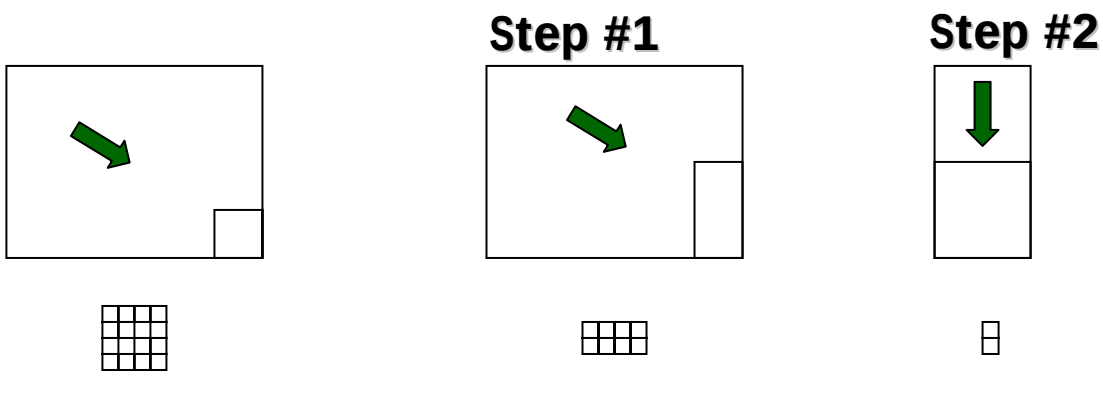
What's Aniso?

- Texture sampling adapts
  - Based on footprint in the texture
- Hardware takes up to N samples along the long axis
- Must have a long axis!
  - Size does matter

```
SetSamplerState(., D3DSAMP_MAXANISOTROPY, 1-N);
SetSamplerState(., D3DSAMP_<>FILTER,
                  D3DTEXF_ANISOTROPIC);
```

So Easy to Implement!

- Just choose the right size reductions
- Big performance boost
- Two steps needed so that aniso kicks in



Example

- 1600x1200 → 256x256
- Set Tex0 as source 1600x1200
  - Set Tex0 filter to aniso, max aniso = 8
  - Use two steps for proper HW aniso
- Render to RTT 1 256x600
  - 8x aniso in horiz
  - 1 bilinear sample in vertical
- Set RTT 1 as source
- Render to RTT 2 256x256

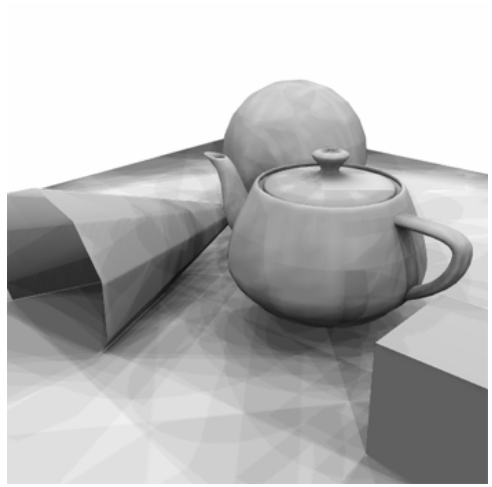
Additional Info

- **Gary's demo**
  - **SDK\DEMOS\Direct3D9\src\AnisoDecimation**
- **Gary's paper**
  - **AnisoDecimation\docs\AnisoDecimation.pdf**
- **Literature**
  - **There is none! muahahha**

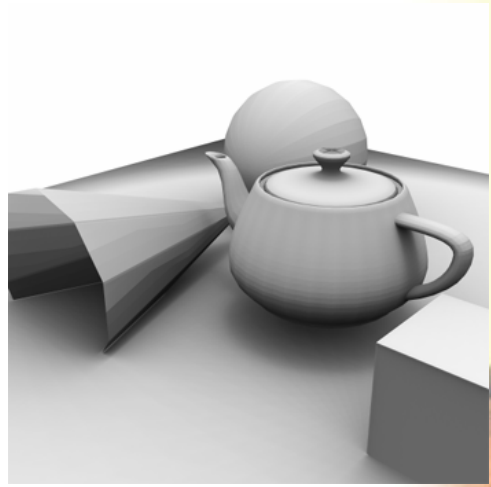
GPU Ambient Occlusion

- Simon Green, Mark Harris, Doug Rogers, NVIDIA
- Shadow maps and fp16 render targets to accumulate occlusion

4 lights



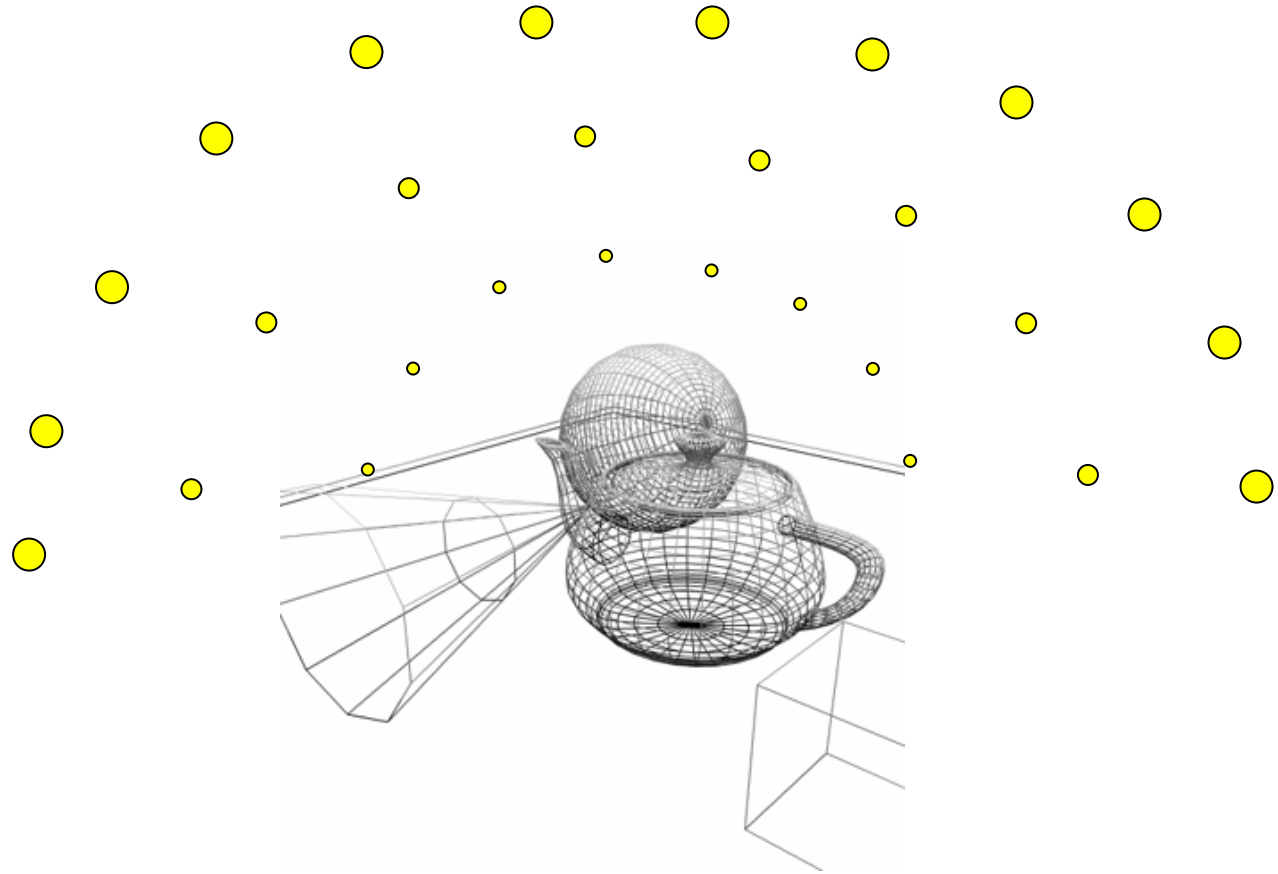
32 lights



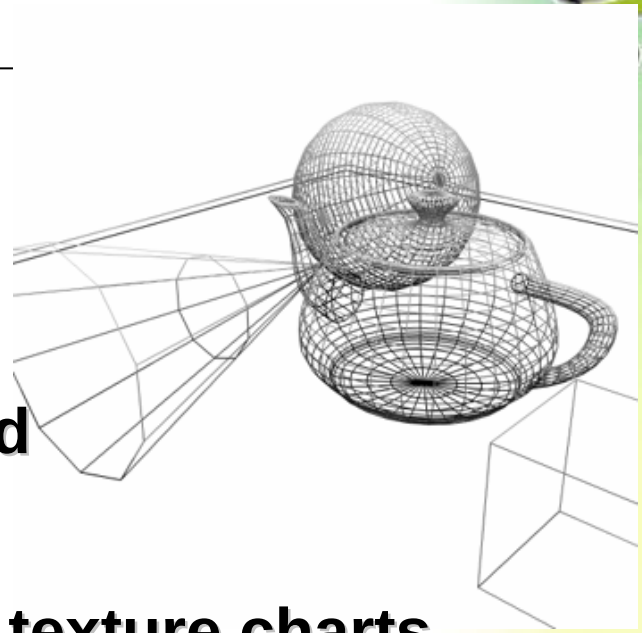
2048 lights

Array of lights around objects

- ~1000 lights



GPU Ambient Occlusion



- Screen space version
  - No extra tesselation required
- Baked lighting version
  - Doug baking illumination to texture charts
- Each light contributes 1 or 0
- Could also weight by N dot L illumination
  - tends to darken

Not real-time

- Almost
  - 128 samples, 2 fps
  - 2048 samples, 9 secs / frame
- Easy to add to your game
  - Generate very high-quality illumination
  - Great in-game stills, marketing! ☺
- Doug's tool for generating high-quality light maps
  - GPU accelerates your off-line pre-processing

Additional Info

- Simon Green & Matt Pharr, ed. Randy Fernando, “Ambient Occlusion,” GPU Gems, Addison-Wesley, 2004
- http://Developer.nvidia.com
- Literature…
  - References coming soon

Real-time Ambient Occlusion

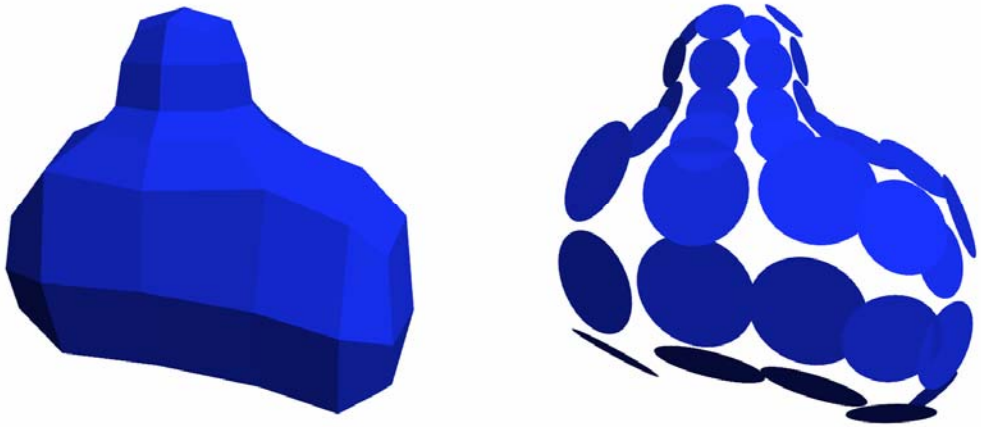
- **Mike Bunnell, NVIDIA**
- **Tuesday, NV sponsored session**

Environment Map + Ambient Occlusion + Indirect Lighting

New Radiance Transfer Algorithm

- For ambient occlusion and indirect lighting
- Deformable bodies
- Dynamic environments!
  - Dynamic lights
- Real-time on the GPU
- Efficient and parallelizable

Dynamic Ambient Occlusion



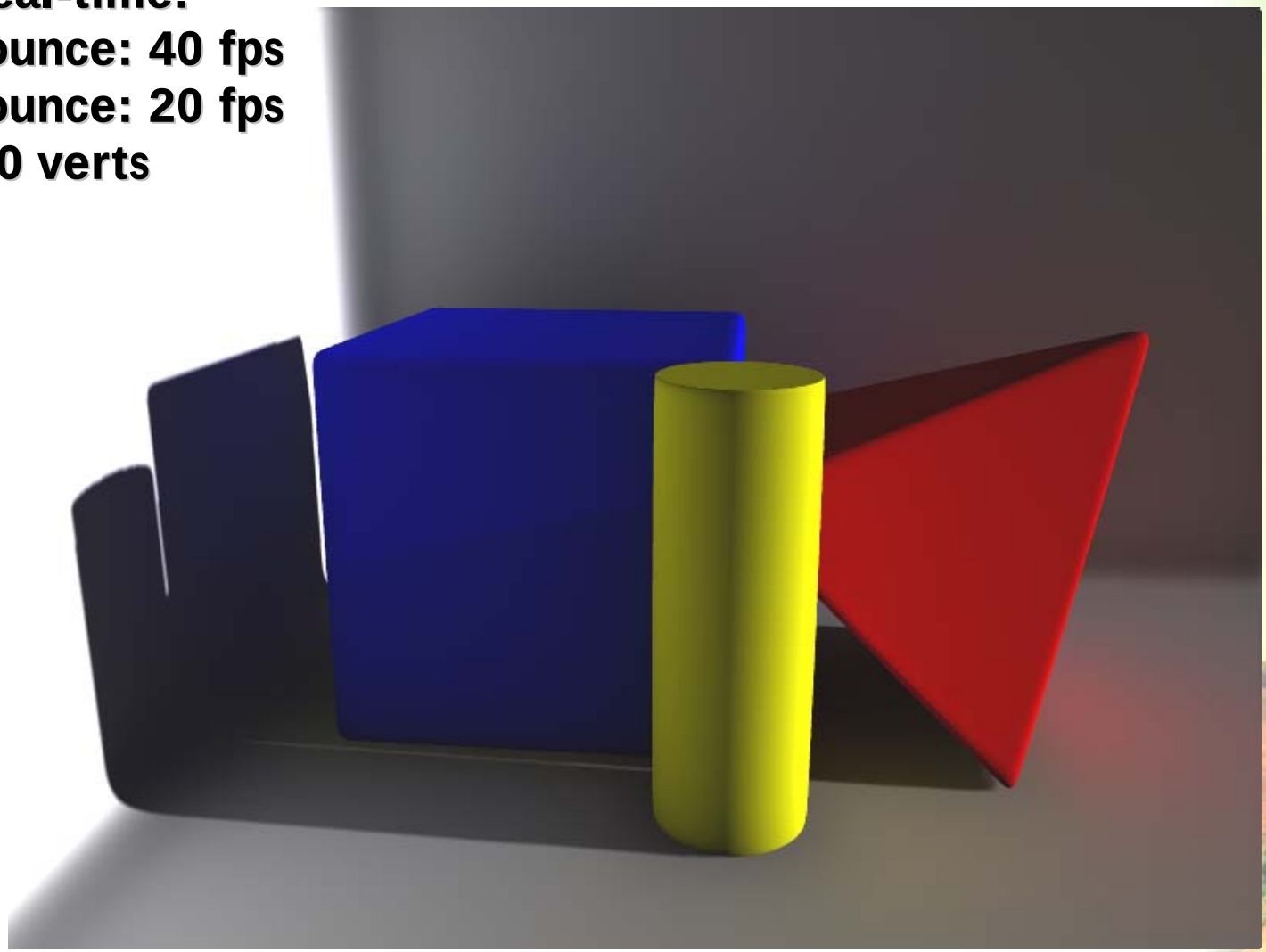
- Oriented disk representation for lighting
- Disks have:
  - position, normal, area
- Disk-to-disk occlusion calculated per frame
  - fp32 render targets

Hierarchy of Elements

- **Easy to generate**
  - **no tesselated geometry**
- **Only traverse children when close to parent**

Direct & Indirect Lighting

In real-time!
1 bounce: 40 fps
2 bounce: 20 fps
6000 verts



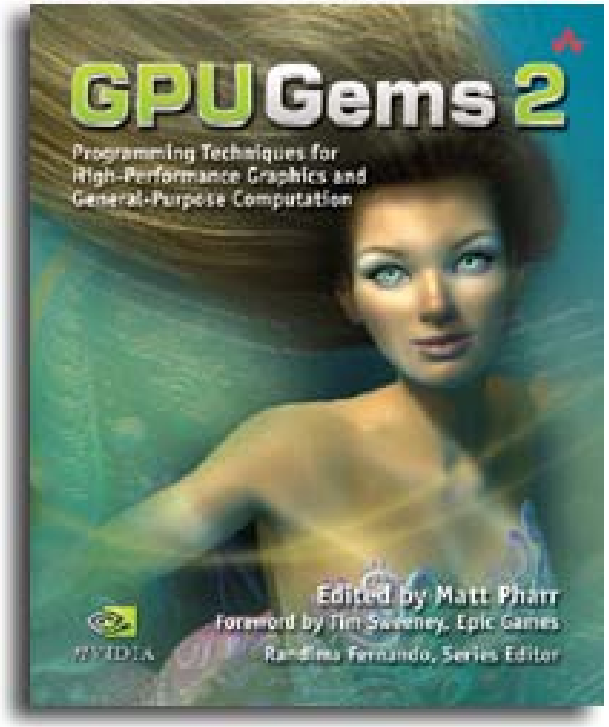
Additional Information

- **Mike's talk**
  - **Tuesday, NV sponsored session**
- **Mike's paper in "GPU Gems 2"**
  - **http://download.nvidia.com/developer/GPU_Gems_2/GPU_Gems2_ch14.pdf**

Questions?

GPU Gems 2

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



- **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