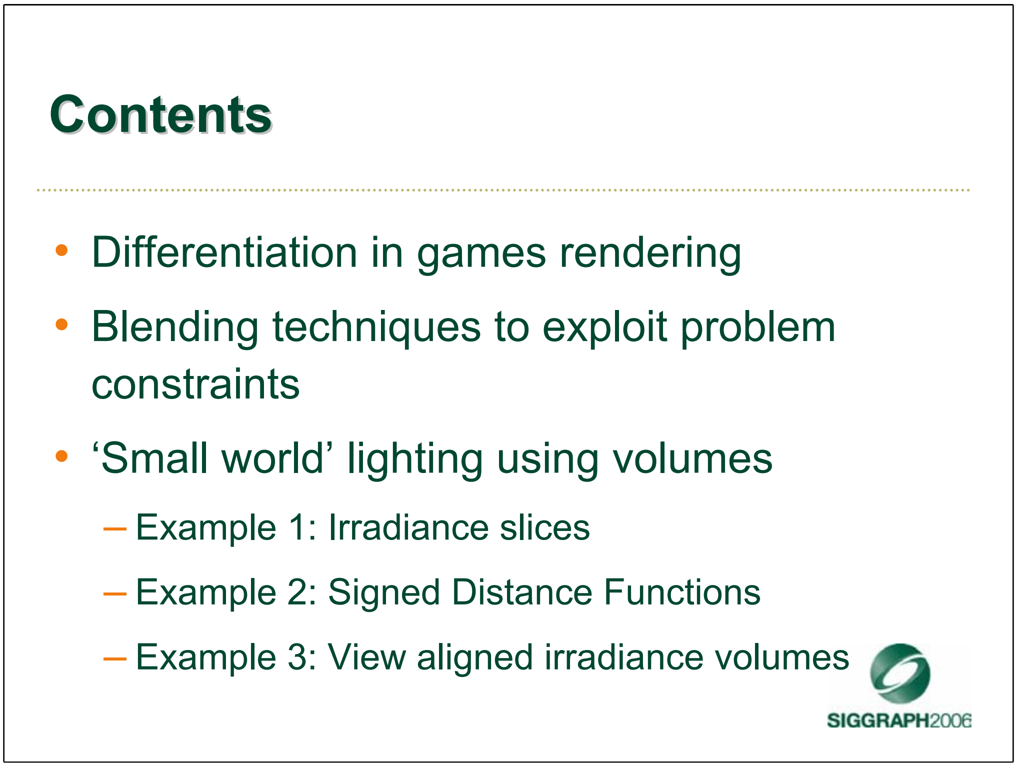
Fast Approximations for lighting of Dynamic Scenes

Alex Evans

Media Molecule Ltd

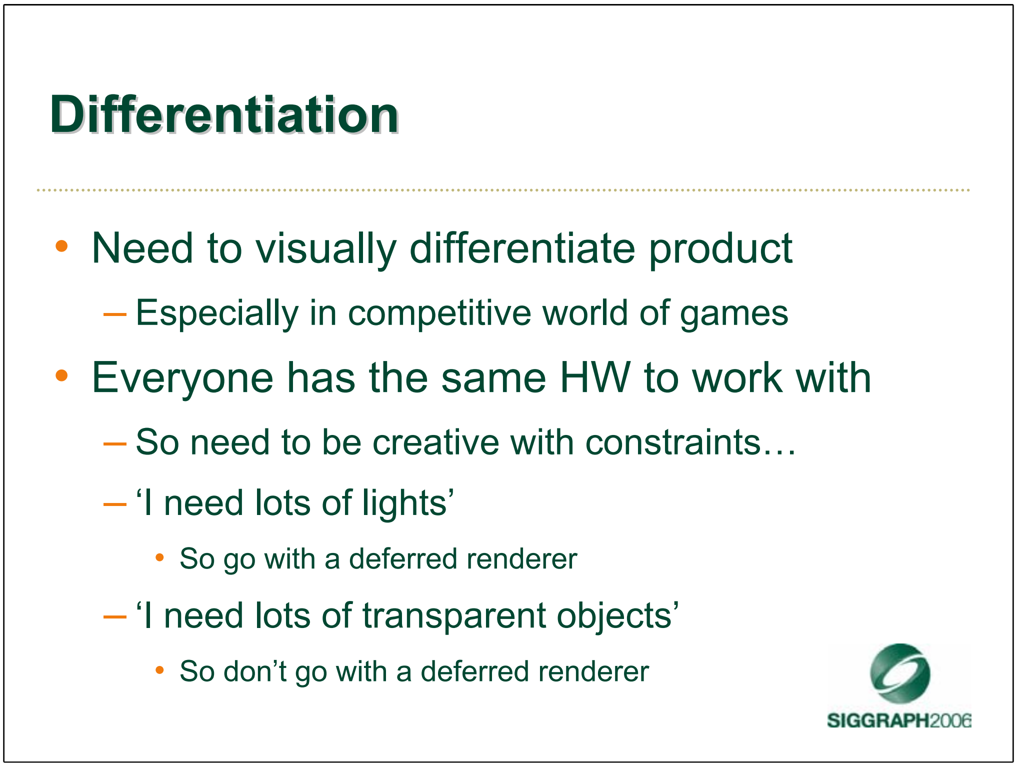
SIGGRAPH2006



Contents

- Differentiation in games rendering
- Blending techniques to exploit problem constraints
- ‘Small world’ lighting using volumes
  - Example 1: Irradiance slices
  - Example 2: Signed Distance Functions
  - Example 3: View aligned irradiance volumes

SIGGRAPH2006



Differentiation

- Need to visually differentiate product
  - Especially in competitive world of games
- Everyone has the same HW to work with
  - So need to be creative with constraints…
  - ‘I need lots of lights’
    - So go with a deferred renderer
  - ‘I need lots of transparent objects’
    - So don’t go with a deferred renderer

SIGGRAPH2006

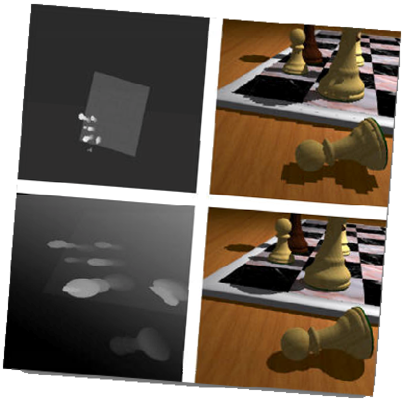
Ultimately all these algorithms are about servicing a look, art director or game designer.

Exploiting techniques by blending them

Always worth thinking of one technique in terms of another. eg first order SH are kinda AO, and 1st order are like bent normals, oh right, zonal harmonics, oh right so this relates to eg horizon maps and occlusion discs and form factors.

.... and having made those connections, gpu acceleration papers are popping up all over the place and so you begin to exploit them.in the same way, intuitions gained in one algorithm can be re-applied to another one, when they are thought about using the same ‘language’, and get ‘blended’ ☺

Exploiting techniques by blending them



← shadow maps.
lots of cool variations...

SIGGRAPH2006

Always worth thinking of one technique in terms of another. eg first order SH are kinda AO, and 1st order are like bent normals, oh right, zonal harmonics, oh right so this relates to eg horizon maps and occlusion discs and form factors.

.... and having made those connections, gpu acceleration papers are popping up all over the place and so you begin to exploit them.in the same way, intuitions gained in one algorithm can be re-applied to another one, when they are thought about using the same ‘language’, and get ‘blended’ ☺

Exploiting techniques by blending them

shadow maps.

lots of cool

Always worth thinking of one technique in terms of another. eg first order SH are kinda AO, and 1st order are like bent normals, oh right, zonal harmonics, oh right so this relates to eg horizon maps and occlusion discs and form factors.

.... and having made those connections, gpu acceleration papers are popping up all over the place and so you begin to exploit them.in the same way, intuitions gained in one algorithm can be re-applied to another one, when they are thought about using the same ‘language’, and get ‘blended’ ☺

Exploiting techniques by blending them

VARIANCE shadow maps.

lots of cool

Always worth thinking of one technique in terms of another. eg first order SH are kinda AO, and 1st order are like bent normals, oh right, zonal harmonics, oh right so this relates to eg horizon maps and occlusion discs and form factors.

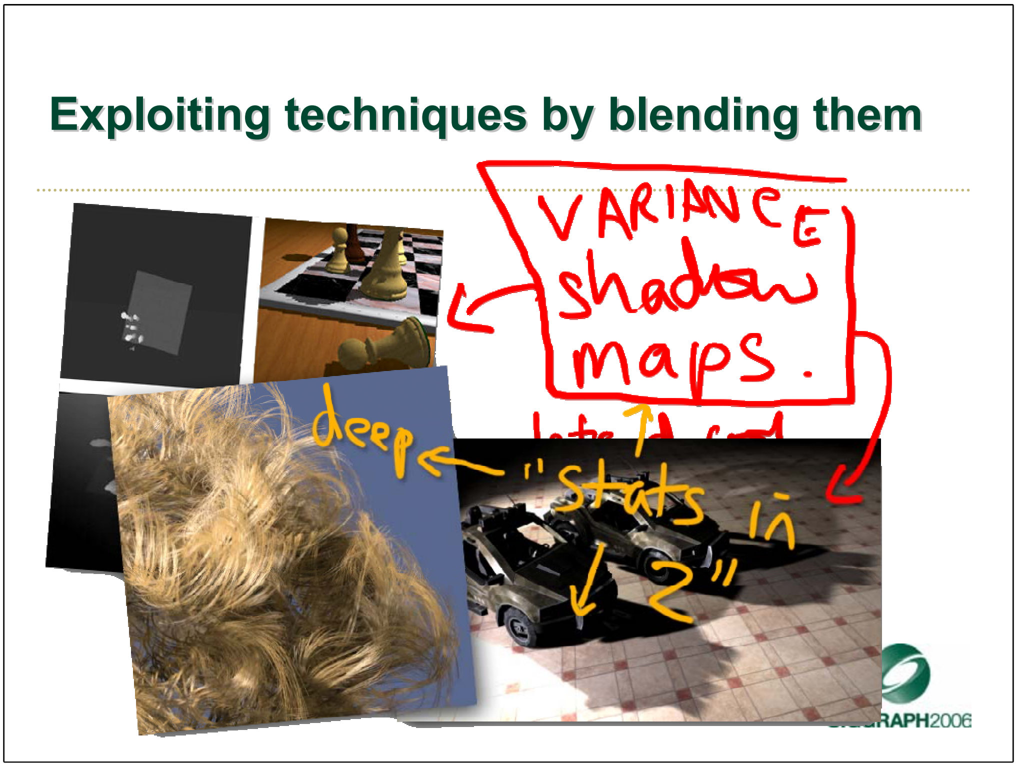
.... and having made those connections, gpu acceleration papers are popping up all over the place and so you begin to exploit them.in the same way, intuitions gained in one algorithm can be re-applied to another one, when they are thought about using the same ‘language’, and get ‘blended’ ☺

Exploiting techniques by blending them

VARIANCE shadow maps.

Always worth thinking of one technique in terms of another. eg first order SH are kinda AO, and 1st order are like bent normals, oh right, zonal harmonics, oh right so this relates to eg horizon maps and occlusion discs and form factors.

.... and having made those connections, gpu acceleration papers are popping up all over the place and so you begin to exploit them.in the same way, intuitions gained in one algorithm can be re-applied to another one, when they are thought about using the same ‘language’, and get ‘blended’ ☺



Exploiting techniques by blending them

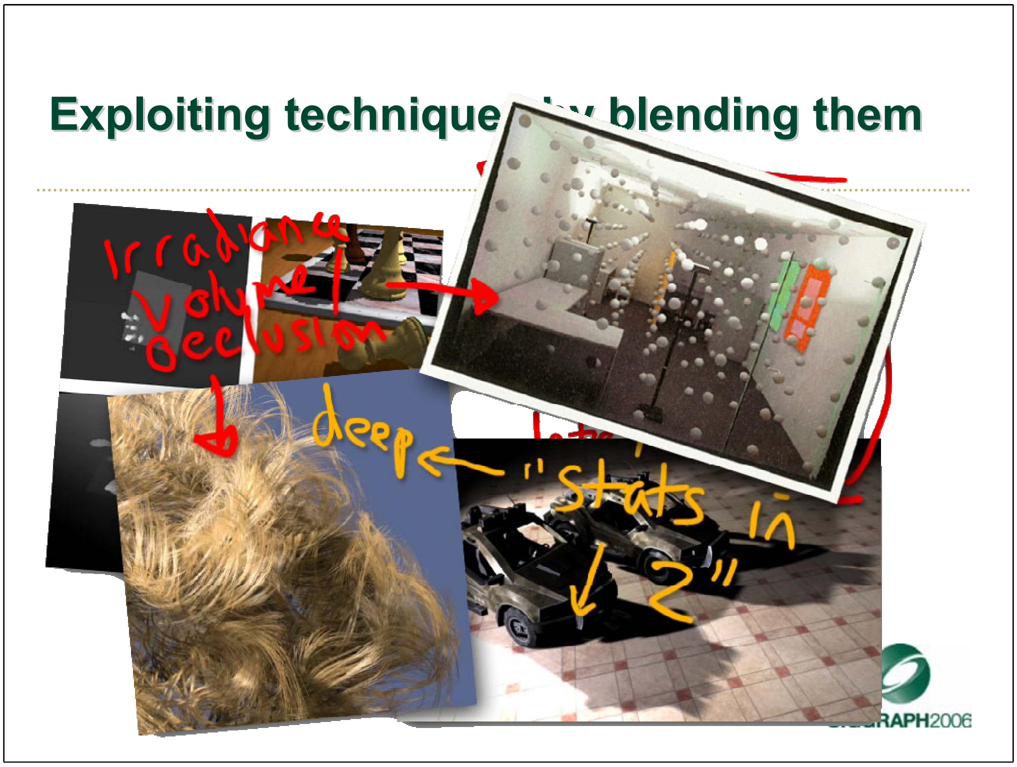
Always worth thinking of one technique in terms of another. eg first order SH are kinda AO, and 1st order are like bent normals, oh right, zonal harmonics, oh right so this relates to eg horizon maps and occlusion discs and form factors.

.... and having made those connections, gpu acceleration papers are popping up all over the place and so you begin to exploit them.in the same way, intuitions gained in one algorithm can be re-applied to another one, when they are thought about using the same ‘language’, and get ‘blended’ ☺

Exploiting technique by blending them

Always worth thinking of one technique in terms of another. eg first order SH are kinda AO, and 1st order are like bent normals, oh right, zonal harmonics, oh right so this relates to eg horizon maps and occlusion discs and form factors.

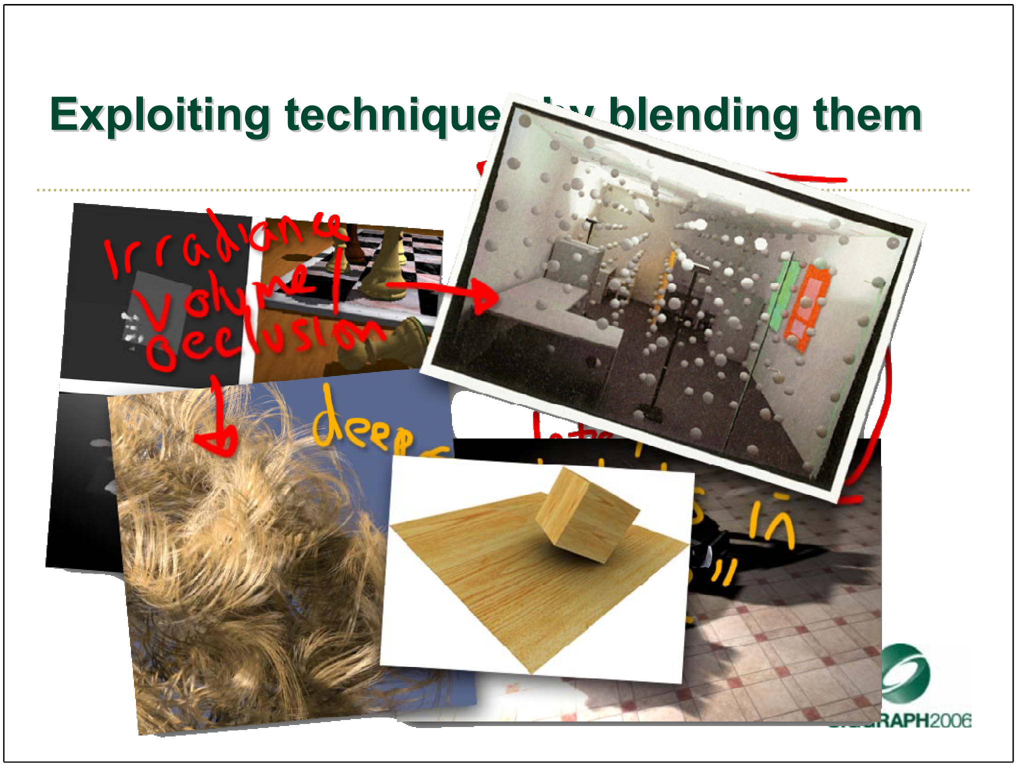
.... and having made those connections, gpu acceleration papers are popping up all over the place and so you begin to exploit them.in the same way, intuitions gained in one algorithm can be re-applied to another one, when they are thought about using the same ‘language’, and get ‘blended’ ☺



Exploiting techniques by blending them

Always worth thinking of one technique in terms of another. eg first order SH are kinda AO, and 1st order are like bent normals, oh right, zonal harmonics, oh right so this relates to eg horizon maps and occlusion discs and form factors.

.... and having made those connections, gpu acceleration papers are popping up all over the place and so you begin to exploit them.in the same way, intuitions gained in one algorithm can be re-applied to another one, when they are thought about using the same ‘language’, and get ‘blended’ ☺

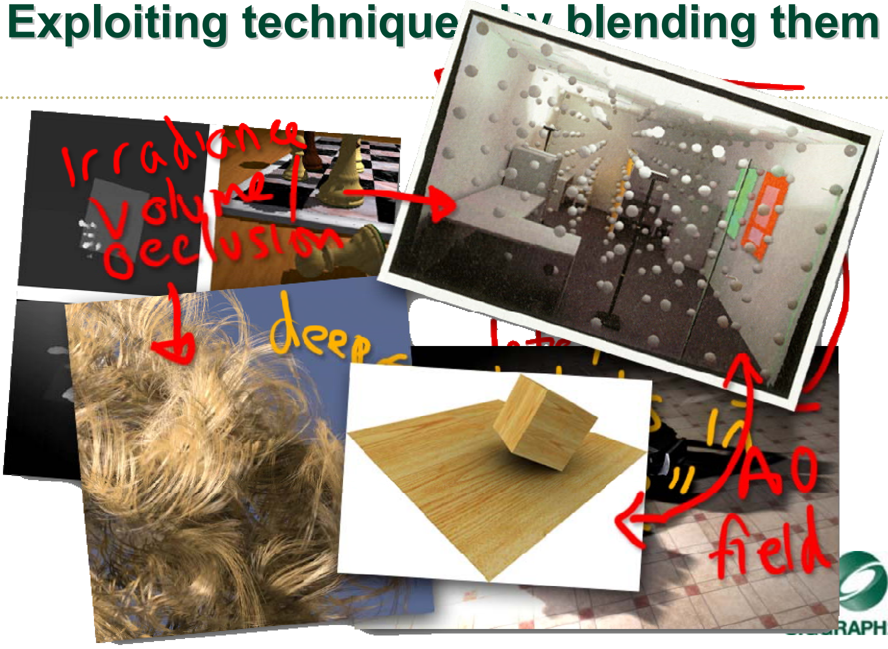


Exploiting techniques by blending them

Always worth thinking of one technique in terms of another. eg first order SH are kinda AO, and 1st order are like bent normals, oh right, zonal harmonics, oh right so this relates to eg horizon maps and occlusion discs and form factors.

.... and having made those connections, gpu acceleration papers are popping up all over the place and so you begin to exploit them.in the same way, intuitions gained in one algorithm can be re-applied to another one, when they are thought about using the same ‘language’, and get ‘blended’ ☺

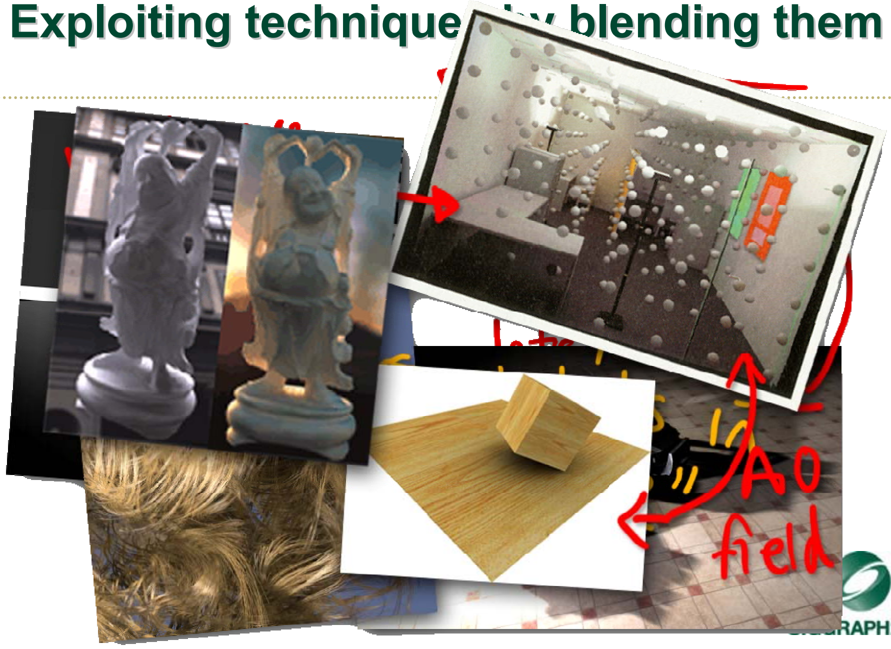
Exploiting techniques by blending them



Always worth thinking of one technique in terms of another. eg first order SH are kinda AO, and 1st order are like bent normals, oh right, zonal harmonics, oh right so this relates to eg horizon maps and occlusion discs and form factors.

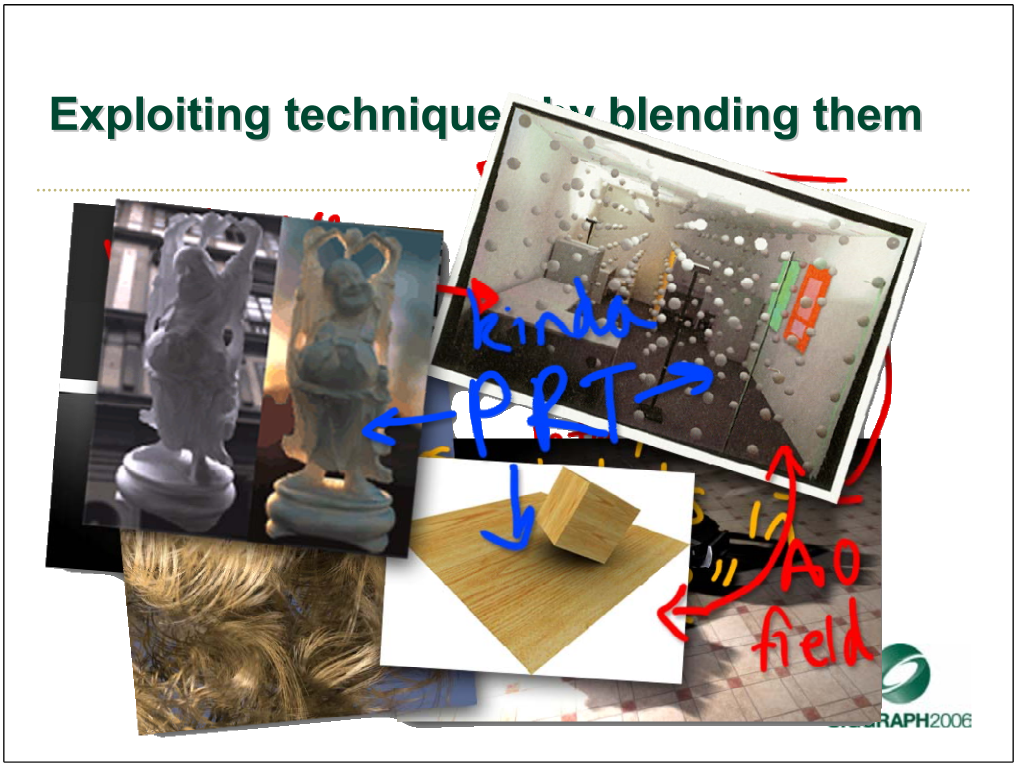
.... and having made those connections, gpu acceleration papers are popping up all over the place and so you begin to exploit them.in the same way, intuitions gained in one algorithm can be re-applied to another one, when they are thought about using the same ‘language’, and get ‘blended’ ☺

Exploiting technique by blending them



Always worth thinking of one technique in terms of another. eg first order SH are kinda AO, and 1st order are like bent normals, oh right, zonal harmonics, oh right so this relates to eg horizon maps and occlusion discs and form factors.

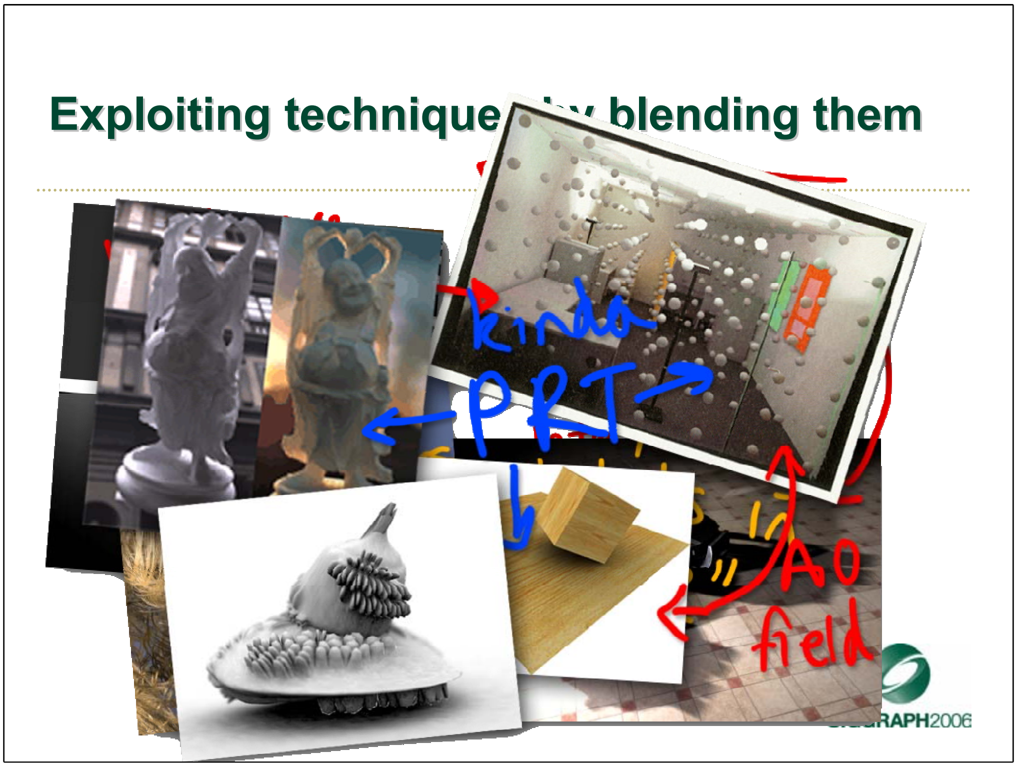
.... and having made those connections, gpu acceleration papers are popping up all over the place and so you begin to exploit them.in the same way, intuitions gained in one algorithm can be re-applied to another one, when they are thought about using the same ‘language’, and get ‘blended’ ☺



Exploiting techniques by blending them

Always worth thinking of one technique in terms of another. eg first order SH are kinda AO, and 1st order are like bent normals, oh right, zonal harmonics, oh right so this relates to eg horizon maps and occlusion discs and form factors.

.... and having made those connections, gpu acceleration papers are popping up all over the place and so you begin to exploit them.in the same way, intuitions gained in one algorithm can be re-applied to another one, when they are thought about using the same ‘language’, and get ‘blended’ ☺

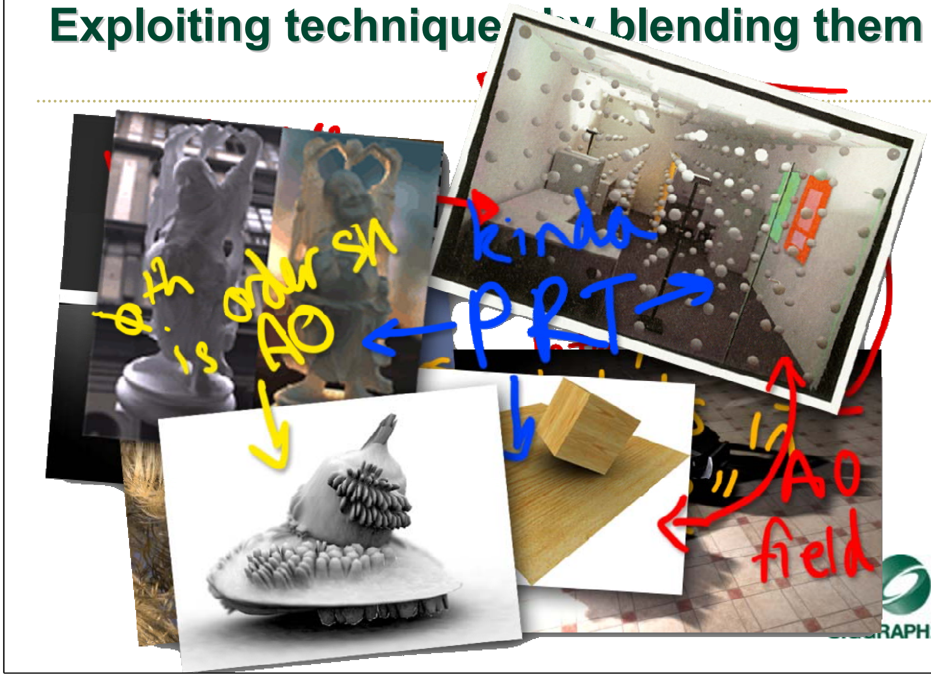


Exploiting techniques by blending them

Always worth thinking of one technique in terms of another. eg first order SH are kinda AO, and 1st order are like bent normals, oh right, zonal harmonics, oh right so this relates to eg horizon maps and occlusion discs and form factors.

.... and having made those connections, gpu acceleration papers are popping up all over the place and so you begin to exploit them.in the same way, intuitions gained in one algorithm can be re-applied to another one, when they are thought about using the same ‘language’, and get ‘blended’ ☺

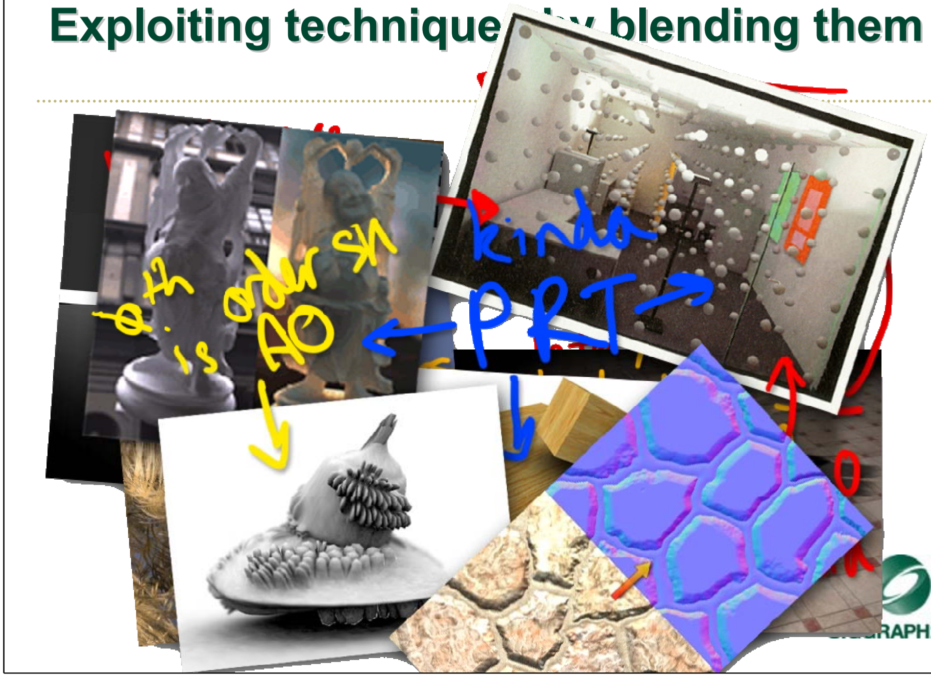
Exploiting techniques by blending them



Always worth thinking of one technique in terms of another. eg first order SH are kinda AO, and 1st order are like bent normals, oh right, zonal harmonics, oh right so this relates to eg horizon maps and occlusion discs and form factors.

.... and having made those connections, gpu acceleration papers are popping up all over the place and so you begin to exploit them.in the same way, intuitions gained in one algorithm can be re-applied to another one, when they are thought about using the same ‘language’, and get ‘blended’ ☺

Exploiting techniques by blending them



Always worth thinking of one technique in terms of another. eg first order SH are kinda AO, and 1st order are like bent normals, oh right, zonal harmonics, oh right so this relates to eg horizon maps and occlusion discs and form factors.

.... and having made those connections, gpu acceleration papers are popping up all over the place and so you begin to exploit them.in the same way, intuitions gained in one algorithm can be re-applied to another one, when they are thought about using the same ‘language’, and get ‘blended’ ☺

Exploiting techniques by blending them

0th order SH is AO
kinda PRT
1st order is bent normal

Always worth thinking of one technique in terms of another. eg first order SH are kinda AO, and 1st order are like bent normals, oh right, zonal harmonics, oh right so this relates to eg horizon maps and occlusion discs and form factors.

.... and having made those connections, gpu acceleration papers are popping up all over the place and so you begin to exploit them.in the same way, intuitions gained in one algorithm can be re-applied to another one, when they are thought about using the same ‘language’, and get ‘blended’ ☺

Exploiting techniqu blending them

APH2006

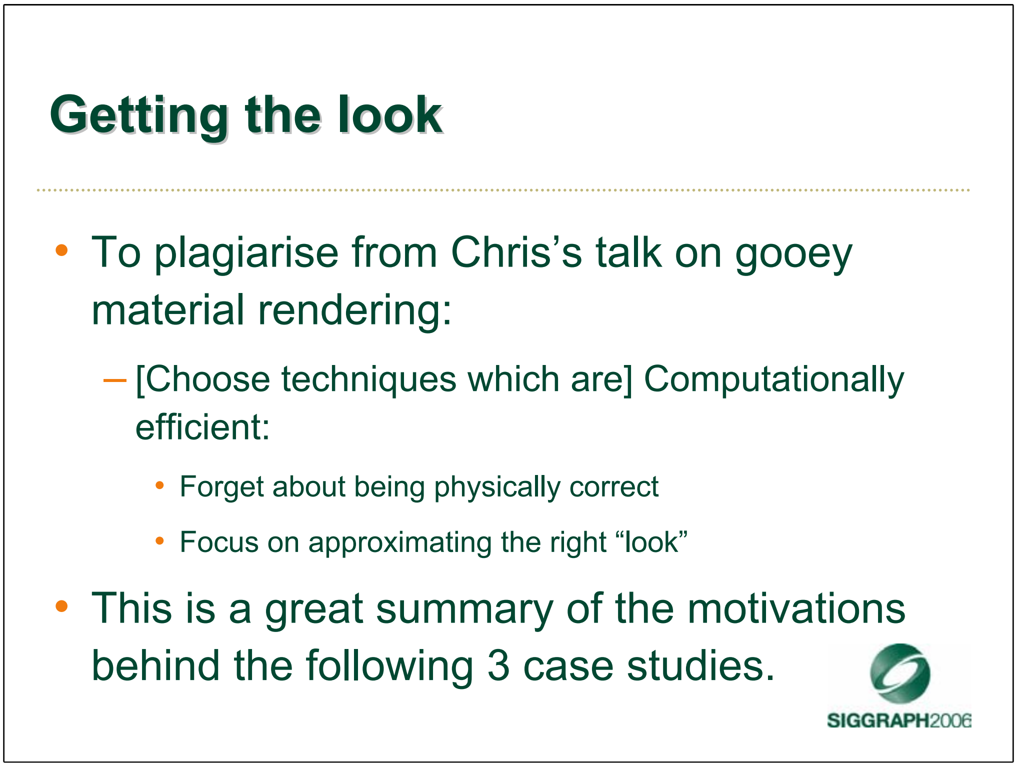
Always worth thinking of one technique in terms of another. eg first order SH are kinda AO, and 1st order are like bent normals, oh right, zonal harmonics, oh right so this relates to eg horizon maps and occlusion discs and form factors.

.... and having made those connections, gpu acceleration papers are popping up all over the place and so you begin to exploit them.in the same way, intuitions gained in one algorithm can be re-applied to another one, when they are thought about using the same ‘language’, and get ‘blended’ ☺

Exploiting techniques by blending them

Always worth thinking of one technique in terms of another. eg first order SH are kinda AO, and 1st order are like bent normals, oh right, zonal harmonics, oh right so this relates to eg horizon maps and occlusion discs and form factors.

.... and having made those connections, gpu acceleration papers are popping up all over the place and so you begin to exploit them.in the same way, intuitions gained in one algorithm can be re-applied to another one, when they are thought about using the same ‘language’, and get ‘blended’ ☺



Getting the look

- To plagiarise from Chris’s talk on gooey material rendering:
  - [Choose techniques which are] Computationally efficient:
    - Forget about being physically correct
    - Focus on approximating the right “look”
- This is a great summary of the motivations behind the following 3 case studies.

Case study: Fake GI in small worlds

- Bounce light effects aid realism greatly
  - – But come at a cost. The real tricky part is always the same: efficient computation of visibility point to point or area to area is hard
  - – Different constraints make it tractable:
    - PRT – static scenes
    - Irradiance Caching / Vertex lighting – reduce number of samples of hard-to-evaluate visibility function

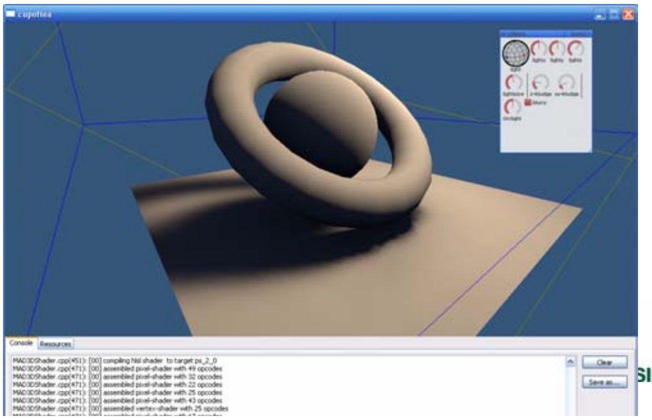
SIGGRAPH2006

Unusual constraint: ‘Small world’

- If the world was small…
  - I could compute/store information about the whole world, dynamically
  - I could store that data in GPU friendly data structures
    - Volume textures!
  - Pros/cons:
    - May make traditionally hard queries like visibility, fast
    - Simpler to represent scene-wide effects rather than just 'light interacts with surface‘
    - May be low or even constant cost regardless of scene complexity
      - This is one of the most useful properties of PRT for games.

Example 1: Irradiance Slices

- Goal: mixed soft and sharp shadows from dynamic planar light source with no precomputation

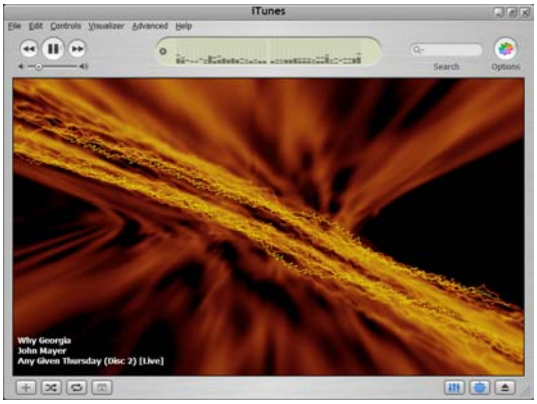


```
MAD3DShader.cpp(451): [00] compiling hlsl shader  to target ps_2_0
MAD3DShader.cpp(471): [00] assembled pixel-shader with 49 opcodes
MAD3DShader.cpp(471): [00] assembled pixel-shader with 32 opcodes
MAD3DShader.cpp(471): [00] assembled pixel-shader with 22 opcodes
MAD3DShader.cpp(471): [00] assembled pixel-shader with 25 opcodes
MAD3DShader.cpp(471): [00] assembled pixel-shader with 43 opcodes
MAD3DShader.cpp(471): [00] assembled vertex-shader with 25 opcodes
```

SIGGRAPH2006

Example 1: Irradiance Slices

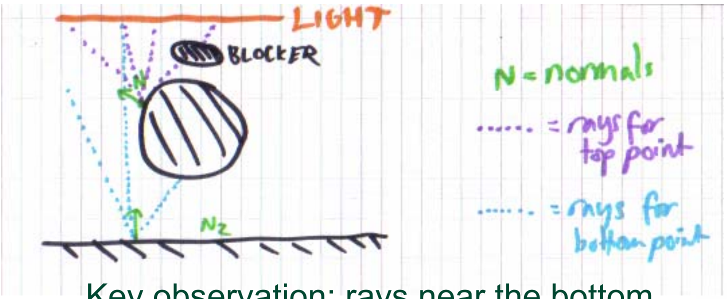
- Inspired by the ‘repeated zoom and blur’ used in every MP3 player ever


iTunes
File Edit Controls Visualizer Advanced Help
Search
Options
Why Georgia
John Mayer
Any Given Thursday (Disc 2) [Live]

SIGGRAPH2006

Irradiance slices: the setup

- Imagine a single, large area light
  - To shade a point, we want to average many ‘random’ rays
    - All ending at the light. They either get there, or they don’t



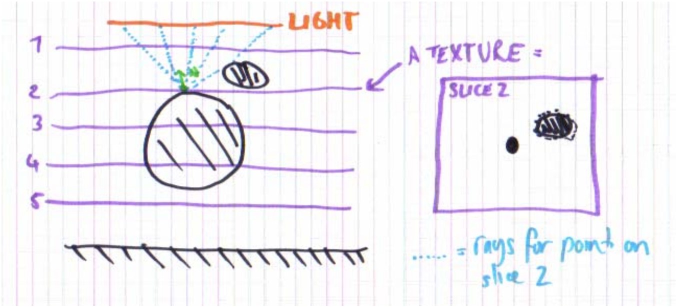
Key observation: rays near the bottom…

…pass through many of the same places rays near the top did

SIGGRAPH2006

Irradiance slices: the idea

- So, slice world parallel to light
  - Store a texture at each slice

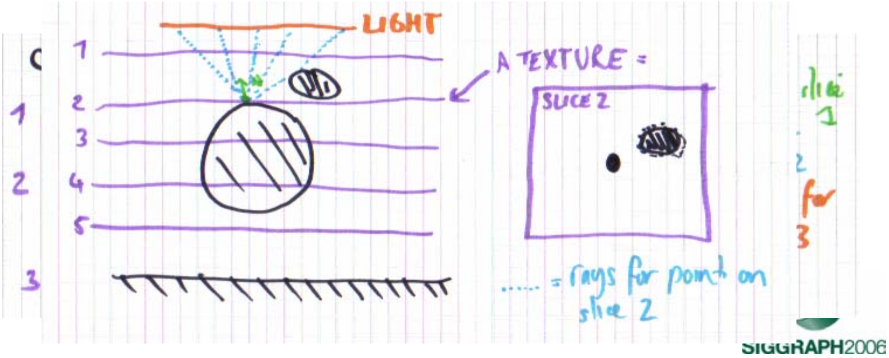


SIGGRAPH2006

Irradiance slices: the idea

Trace rays from each plane in turn, starting nearest the light

- If a ray gets to the previous plane, stop tracing
  - And just return the result at the previous plane

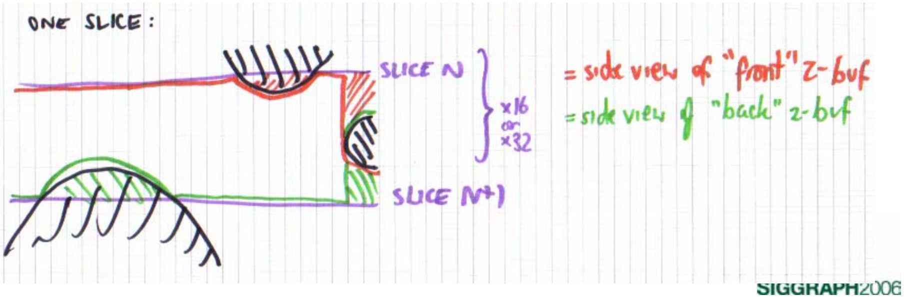


Demo

- That was pretty abstract so before I describe how to do it on GPU, have a demo…
  - The artefacts as you move the light around
    - Are caused by only updating ¼ of the slices per frame.
      - Demo written for a 9600 mobility– on modern kit, no need any more!
  - Features this has that are nice:
    - Sharp and soft shadows can intermix correctly
    - Real inner and outer shadow penumbras
    - Completely dynamic. Fixed cost. Low geometry Cost

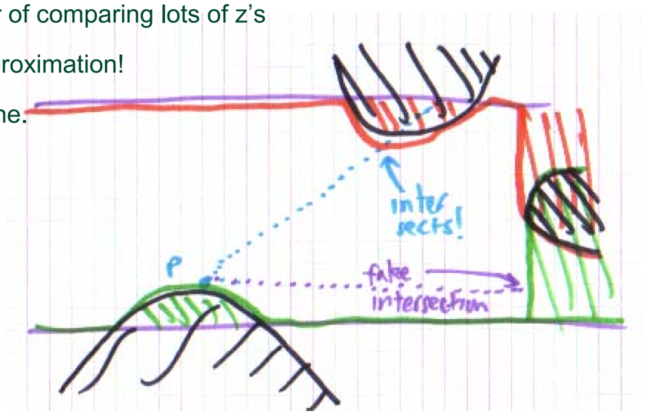
Irradiance slices on the GPU

- How to represent the scene geometry?
  - Render a front and back zbuffer for each slice
    - Using MRT’s, with min-blending, can render 16 per pass
    - Take care over slices that are inside geometry



Irradiance slices on the GPU 2

- Now we have for each slice,
  - A min and max z. so tracing a ray within a slice…
    - …is just a matter of comparing lots of z's
    - This is all an approximation!
    - But it's a good one.



Irradiance slices on the GPU 3

- Each slice is stored as a texture…
    - A PS2.0 shader can trace one ray per pixel
        - We trace a 4x4 block and average by downsampling
    - Sadly we can’t write directly to a volume texture
        - So the slices are laid out in 2d on a huge 2048x2048 map
- Then at render time, shading is just a texture lookup!
    - Just find nearest point in slice volume and sample
        - Will become more efficient as GPUs get more orthogonal

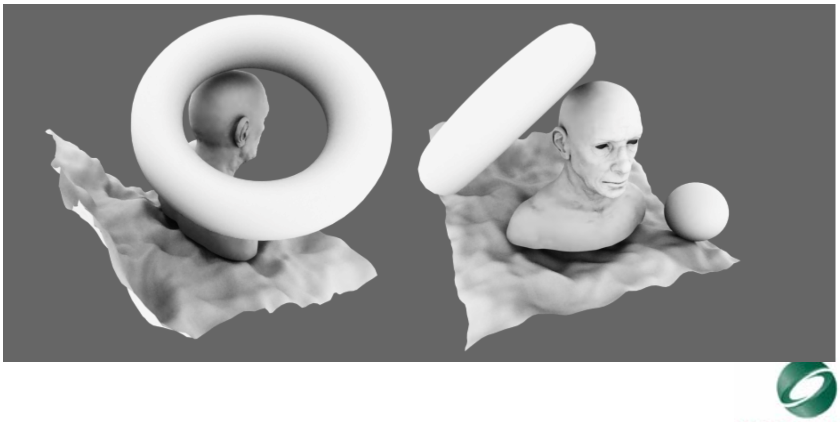
That’s it for Irradiance Slices

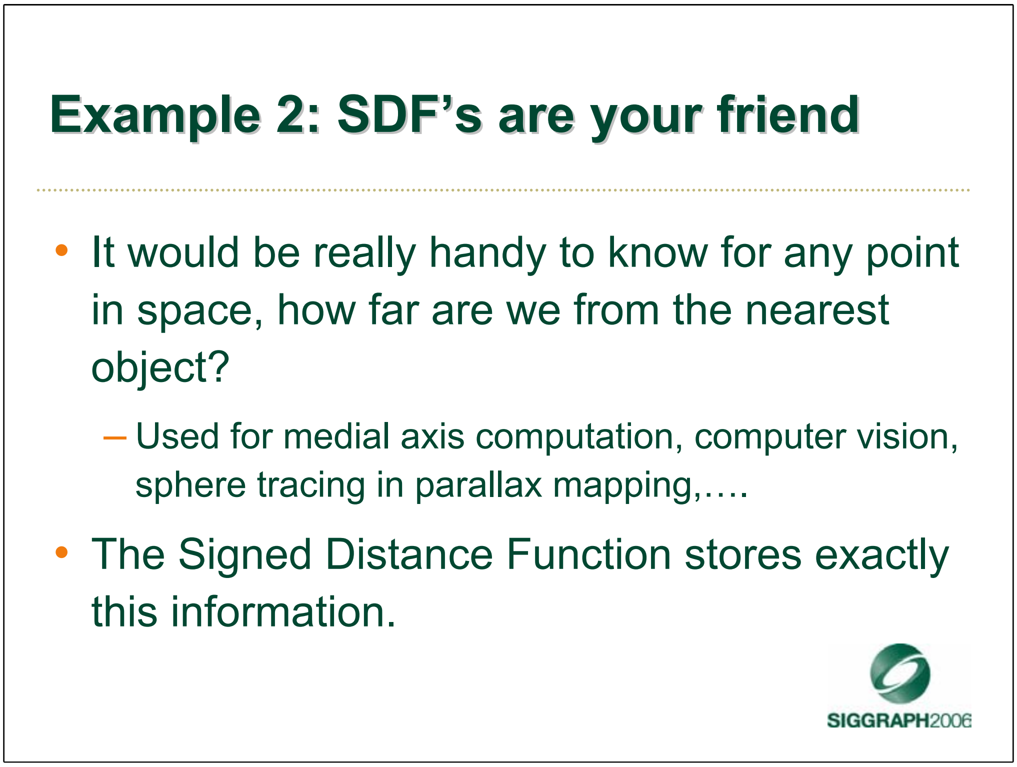
- Post-Mortem:
  - Fixed cost, huge pixel fill rate
    - but great results. Not practical…. Yet
  - Main source of error is z-slice approximation to scene geometry
    - But it allows completely dynamic geometry
  - It’s only one light at a time
    - Could implement a system using a grid of slices
      - Allows arbitrary # of lights in fixed time
        - › Also bounce light! (Every wall can be a light)

SIGGRAPH2006

Example 2: Skylight for dynamic scenes

- The goal:

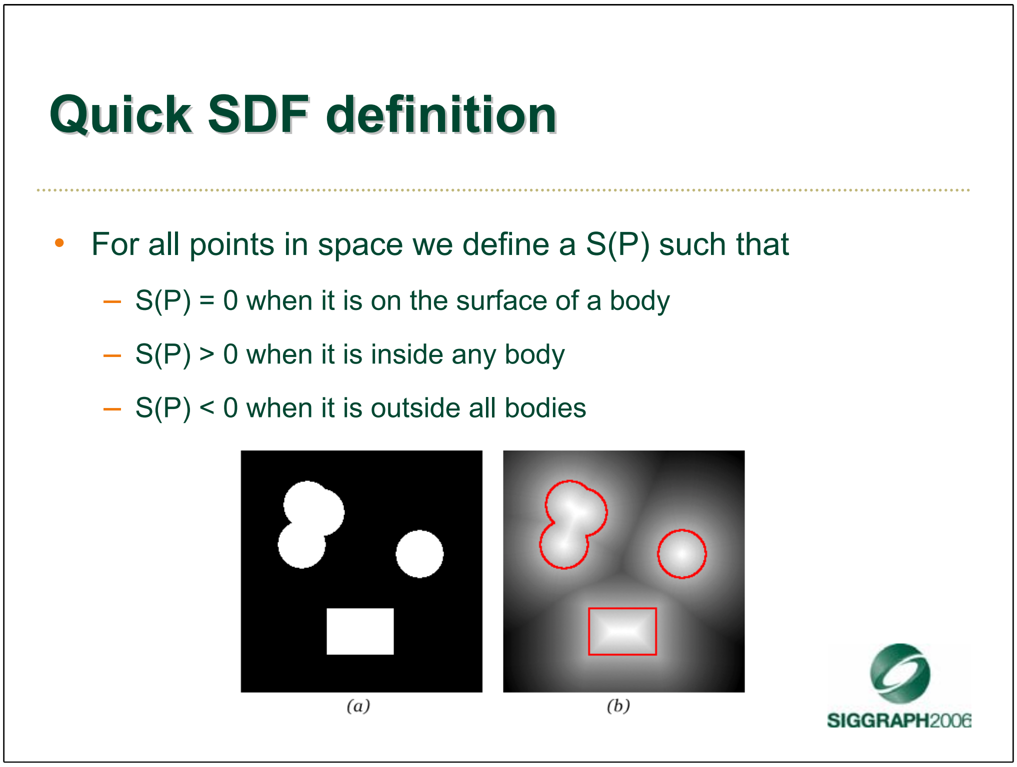




Example 2: SDF’s are your friend

- It would be really handy to know for any point in space, how far are we from the nearest object?
  - Used for medial axis computation, computer vision, sphere tracing in parallax mapping,….
- The Signed Distance Function stores exactly this information.

SIGGRAPH2006



Quick SDF definition

- For all points in space we define a S(P) such that
  - S(P) = 0 when it is on the surface of a body
  - S(P) > 0 when it is inside any body
  - S(P) < 0 when it is outside all bodies

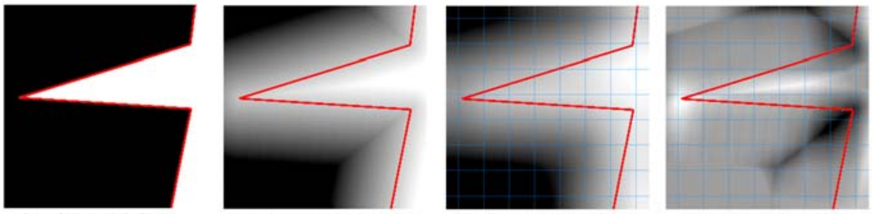
(a) *(b)*

SIGGRAPH2006

Can use many distance metrics. It turns out for our purposes pretty much any old distance function will do – in particular a gaussian blurred version of the binary image actually accentuates the affect we want, namely surface curvature.

Using SDF’s to measure surface curvature

- We want that ‘AO’ look – where regions inside creases and valleys receive less sky light.
  - Surface curvature is a great start.

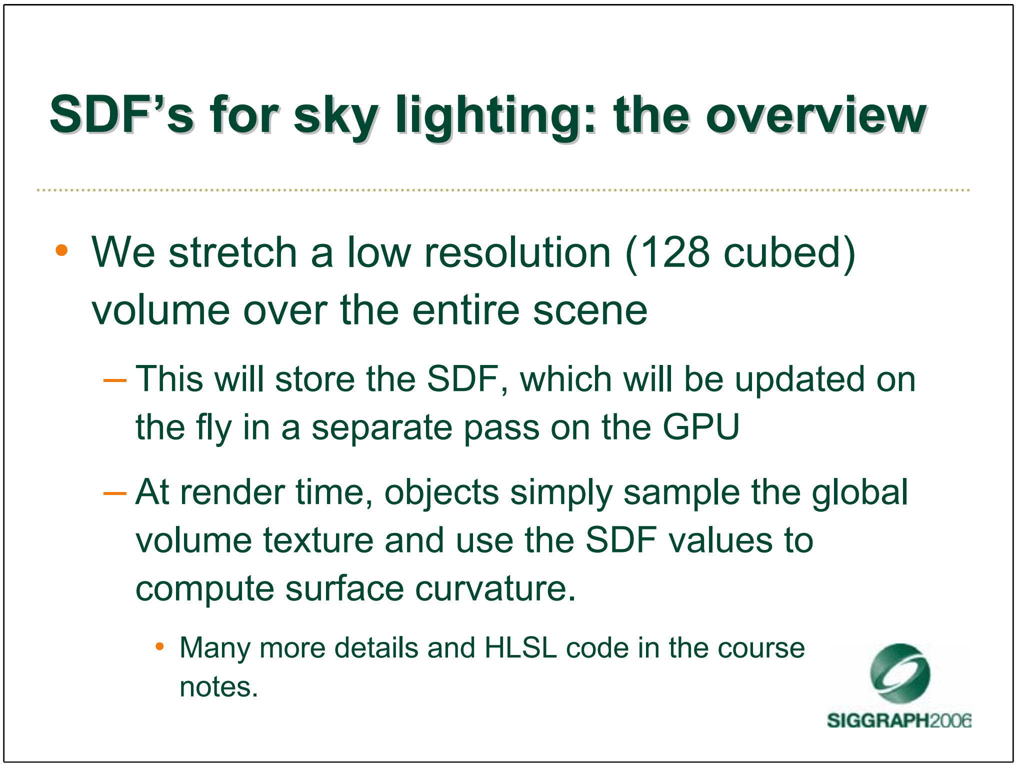


(a) - the original object

(b) - the euclidian SDF of (a)

(c) - SDF downsampled to 8x8 and then reconstructed bilinearly

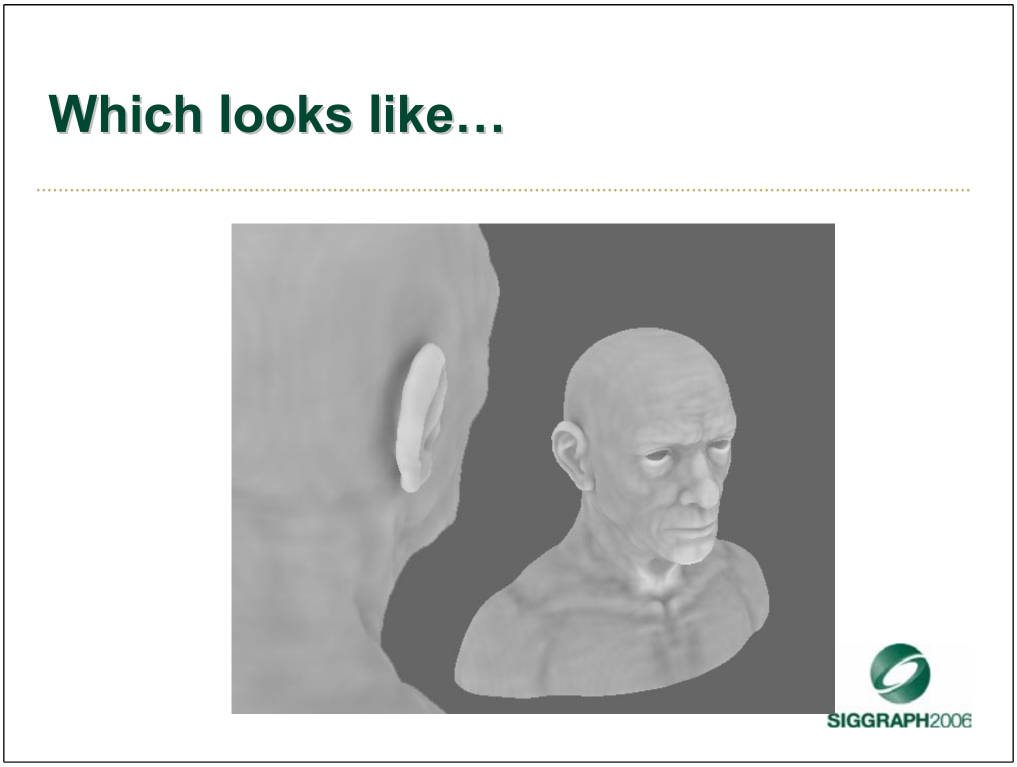
(d) - difference between (a) and (b) [exaggerated]



SDF’s for sky lighting: the overview

- We stretch a low resolution (128 cubed) volume over the entire scene
  - This will store the SDF, which will be updated on the fly in a separate pass on the GPU
  - At render time, objects simply sample the global volume texture and use the SDF values to compute surface curvature.
    - Many more details and HLSL code in the course notes.

SIGGRAPH2006

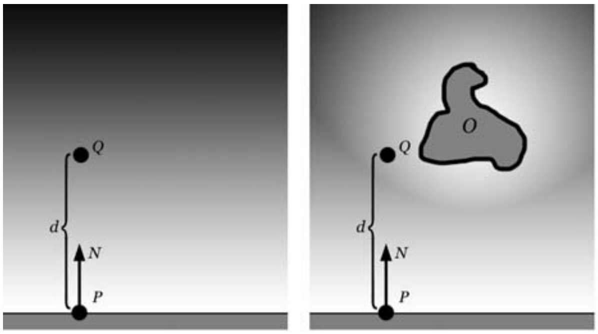


Which looks like…

Not a bad start. But this is still direction free, there is no sense of an up/sky vector. this is a case where experimentation and messing around can really get you a long way.

SDF as occlusion map

- If we sample SDF at a distance ‘d’ from surface, we expect result ‘d’ (or –d)…
  - Unless there’s another object in the way.

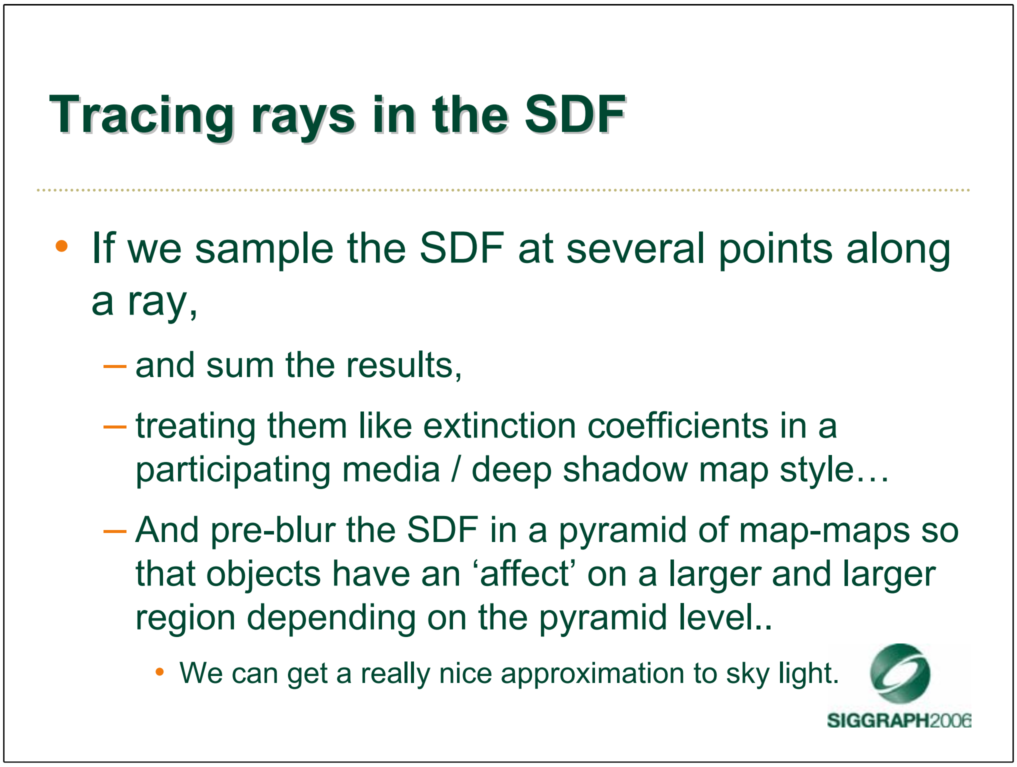


A step back: what’s going on here?

- The SDF is really serving as a easy-to-sample measure of the ‘occludedness’ of points in space
  - We’re not actually making explicit use of it’s distance properties
  - Suggests an analogy with deep shadow maps: we just get a blurry picture of how occluded a particular point is.
  - What if we were to trace rays through this volume?

SIGGRAPH2006

The intuition is: the SDF here is really acting for us like a measure of occludedness



Tracing rays in the SDF

- If we sample the SDF at several points along a ray,
  - – and sum the results,
  - – treating them like extinction coefficients in a participating media / deep shadow map style…
  - – And pre-blur the SDF in a pyramid of map-maps so that objects have an ‘affect’ on a larger and larger region depending on the pyramid level..
    - We can get a really nice approximation to sky light.

This is the ‘blender’ idea I was talking about in the opening. Despite having arrived at this point via the (well studied) idea of SDF’s, we’re now in territory that has a lot in common with deep shadow maps, participating media, volume rendering, and irradiance volumes. Which is all good, because we can take intuitions from there and apply them here.

Tracing rays in the SDF

- In the absence of occluders we expect

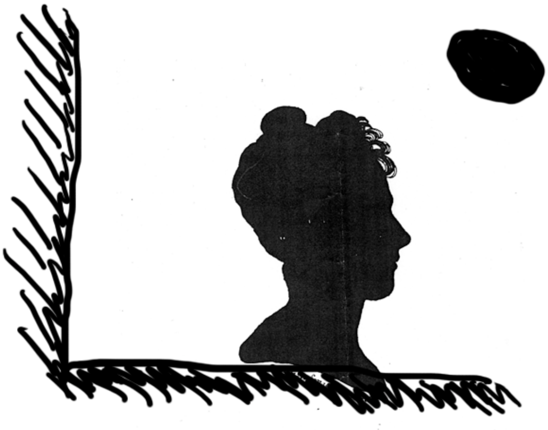
$$\sum_{i=0}^{n} SDF(P + d_i N) = -\sum_{i=0}^{n} d_i$$

- So in the shader we do:

$$C = \exp(k\sum SDF(P + d_i N) + d_i)$$

- When you skew ‘N’ towards the sky
  - So that you’re tracing rays slightly upwards,
  - Upward facing faces naturally get a brighter, less occluded shade. And we complete the ‘AO’ look we wanted.

Sketch of what's going on.



SIGGRAPH2006

Sketch of what's going on.

SIGGRAPH2006

Sketch of what's going on.

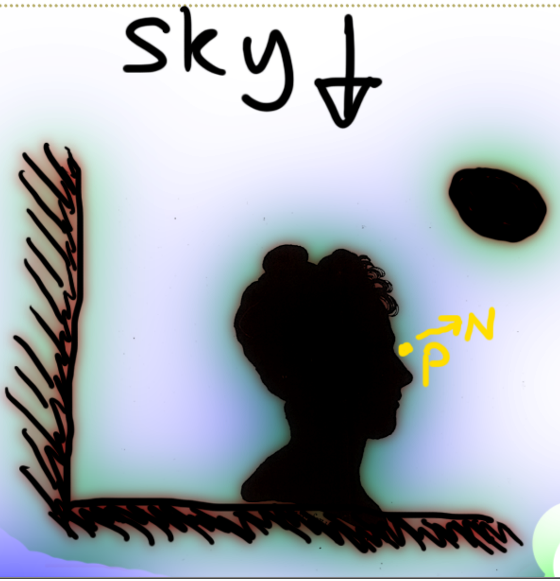
Sketch of what's going on.

SIGGRAPH2006

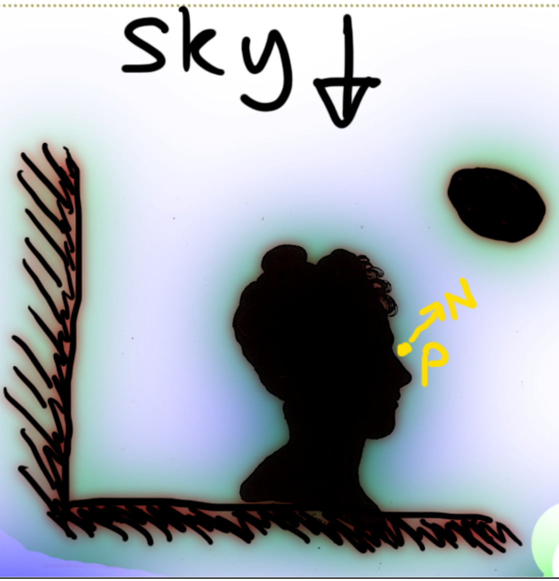
Sketch of what's going on.

SIGGRAPH2006

Sketch of what's going on.

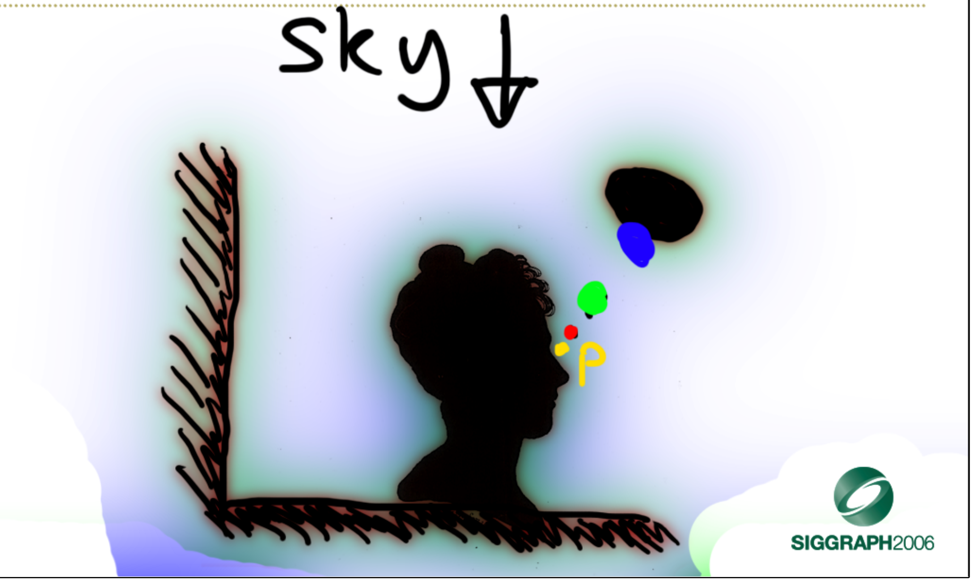


Sketch of what's going on.



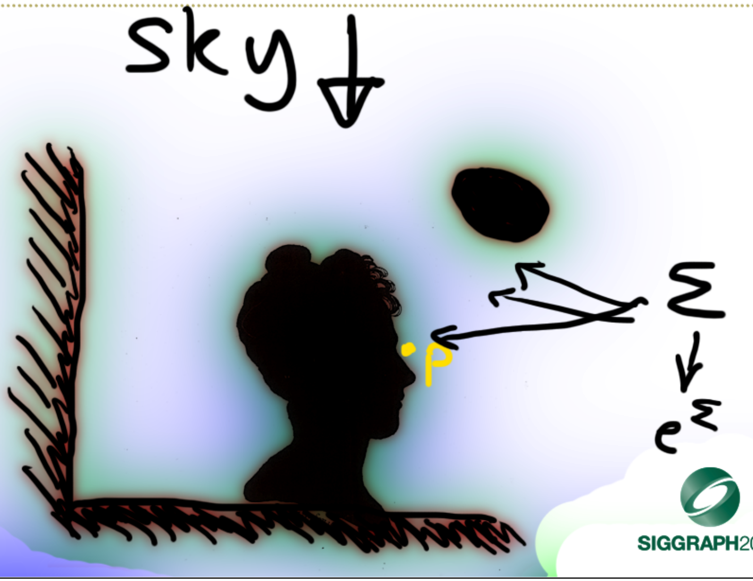
SIGGRAPH2006

Sketch of what's going on.

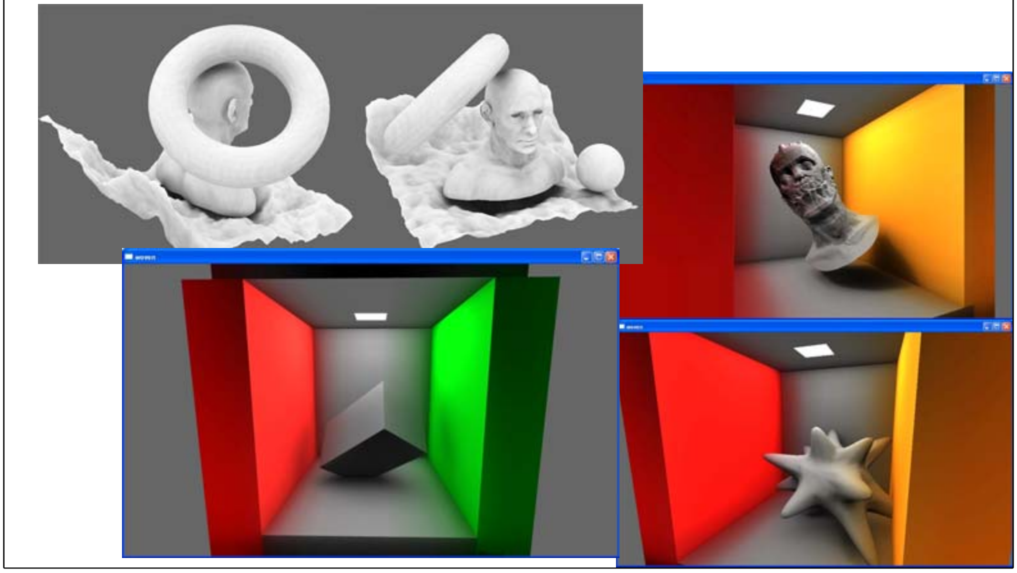


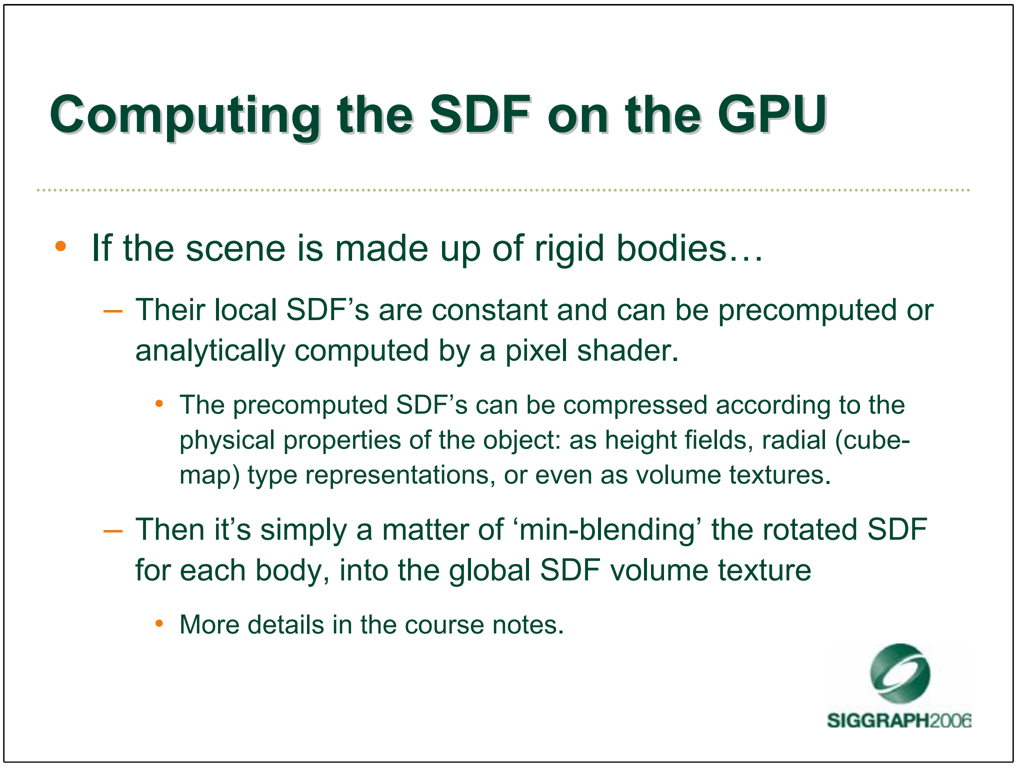
SIGGRAPH2006

Sketch of what's going on.



Picture & Demo

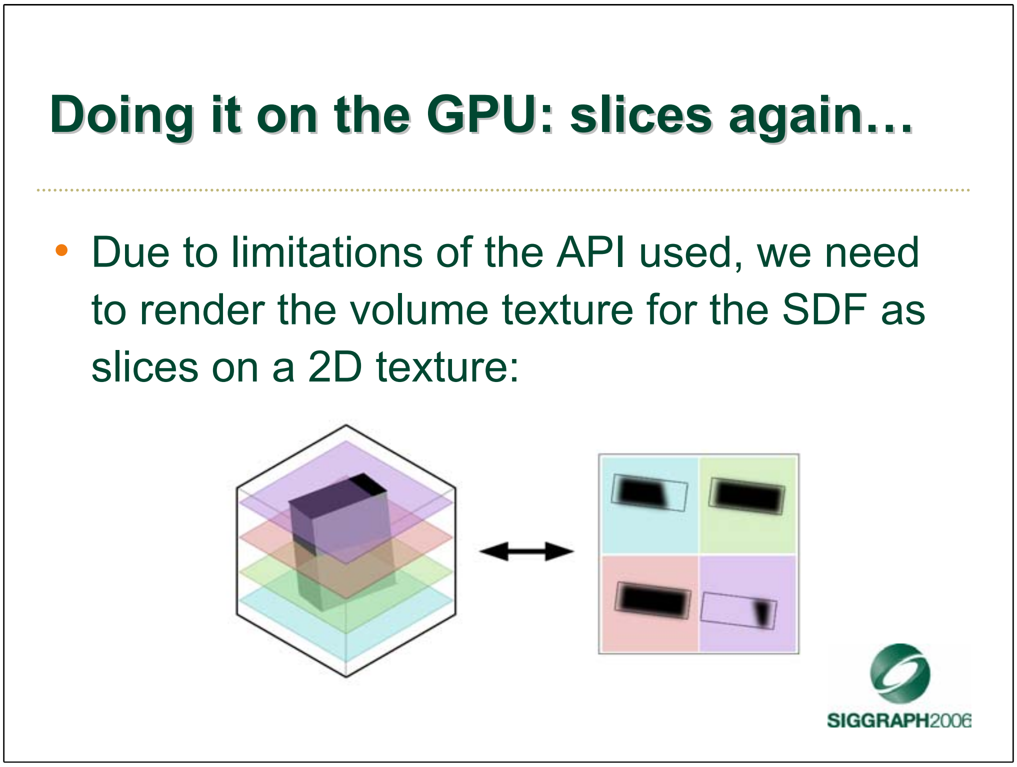




Computing the SDF on the GPU

- If the scene is made up of rigid bodies…
  - Their local SDF’s are constant and can be precomputed or analytically computed by a pixel shader.
    - The precomputed SDF’s can be compressed according to the physical properties of the object: as height fields, radial (cube-map) type representations, or even as volume textures.
  - Then it’s simply a matter of ‘min-blending’ the rotated SDF for each body, into the global SDF volume texture
    - More details in the course notes.

SIGGRAPH2006



Doing it on the GPU: slices again…

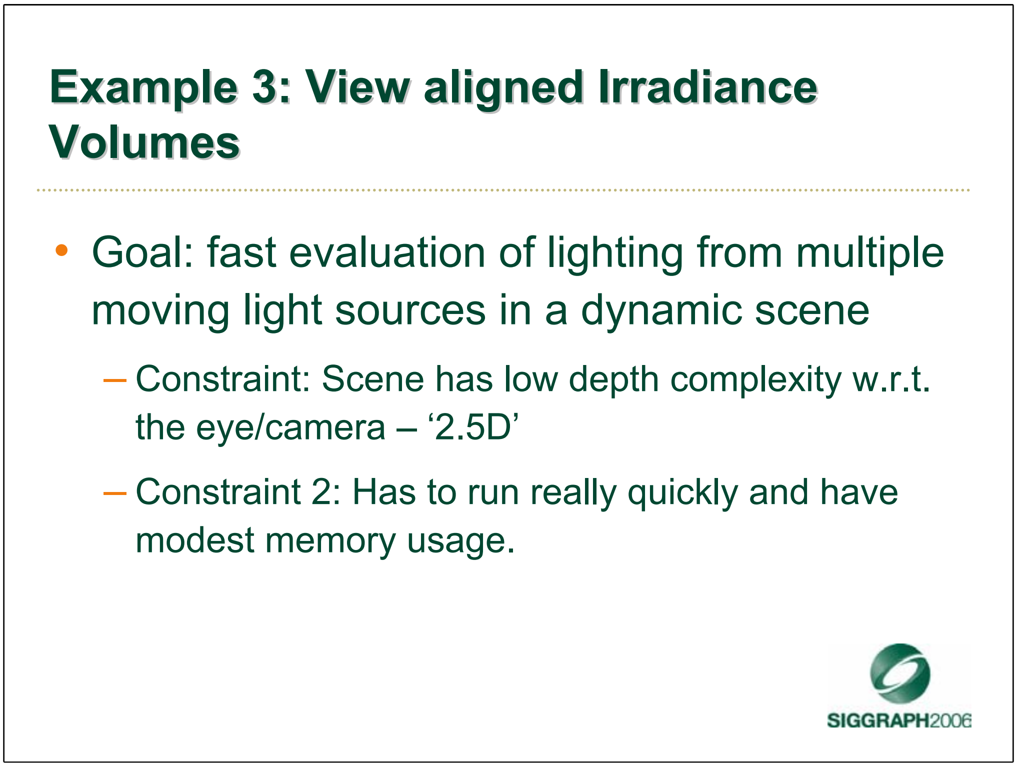
- Due to limitations of the API used, we need to render the volume texture for the SDF as slices on a 2D texture:

SIGGRAPH2006

A vertex shader computes the 4 vertices of the eges of the most aligned axis of the OBB of the object to be rendered, then a pixel shader evluates the SDF. ‘min blending’ allows multiple objects to be composited together into the SDF, and then mip map generation gives us the pyramid of blurred volumes used in the sampling of the volume. More details in the course notes.

Colour Bleeding and wrap up

- By computing SDF's for each colour channel, and offsetting each object's SDF values by their surface albedo,
  - Fake bounce light effects are introduced 'automatically'
    - The bounce light isn't physically correct, or shadowed – but gives visual queues which viewers are happy to accept as secondary diffuse reflections.
- This technique gives pleasing 'AO' and sky-type lighting at very low cost even for high geometric complexity scenes
  - But the approximations used are so coarse that the unphysical nature of the lighting makes it tricky to use in complex scenes



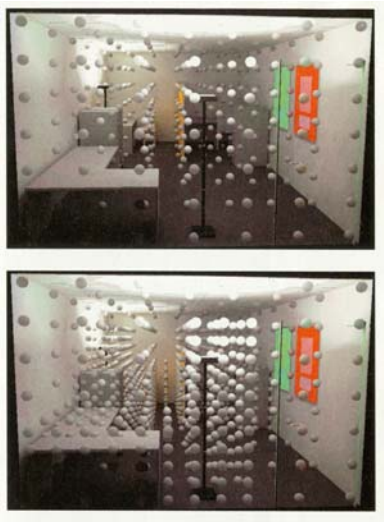
Example 3: View aligned Irradiance Volumes

- Goal: fast evaluation of lighting from multiple moving light sources in a dynamic scene
  - Constraint: Scene has low depth complexity w.r.t. the eye/camera – ‘2.5D’
  - Constraint 2: Has to run really quickly and have modest memory usage.

SIGGRAPH2006

Irradiance Volumes

- Many games use irradiance volumes to store (and sample) the light flowing through any point in a scene
  - Normally they are world aligned and precomputed at coarse resolution
  - They store the irradiance flowing in any direction, often compressed using a spherical harmonic representation

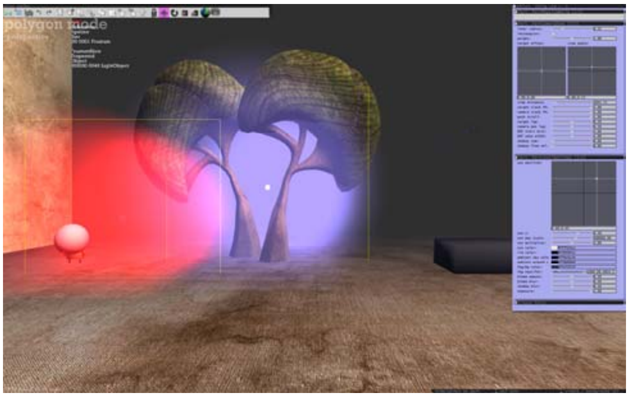


View aligned Irradiance Volumes

- In a dynamic scene, it’s not possible to precompute the irradiance volume
  - So we are going to recompute it on the fly using the GPU, at low resolution, based on a potentially large number of light emitters
- Since we’re recomputing it every frame,
  - It makes sense to compute it in screen space.
  - In this example, the target constraint was for a ‘2.5D’ thin world, so a small number (16) of slices are used, parallel to the screen. They are evenly spaced in post projective space, ie evenly spaced in ‘w’. (1/z)

The setup

- Here's a view of the example scene, as seen in a game editor:

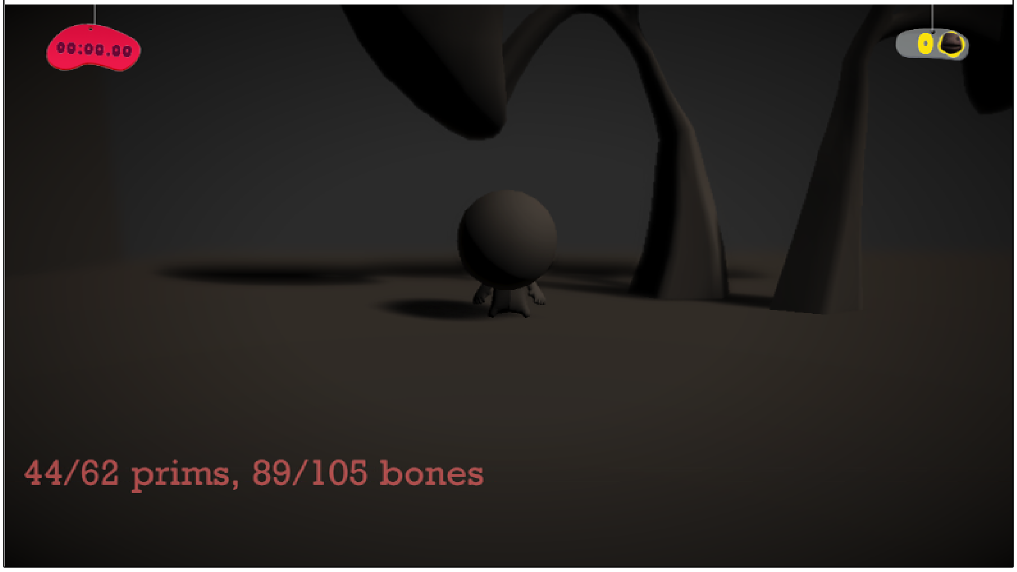


Rendering of the lights into the volume

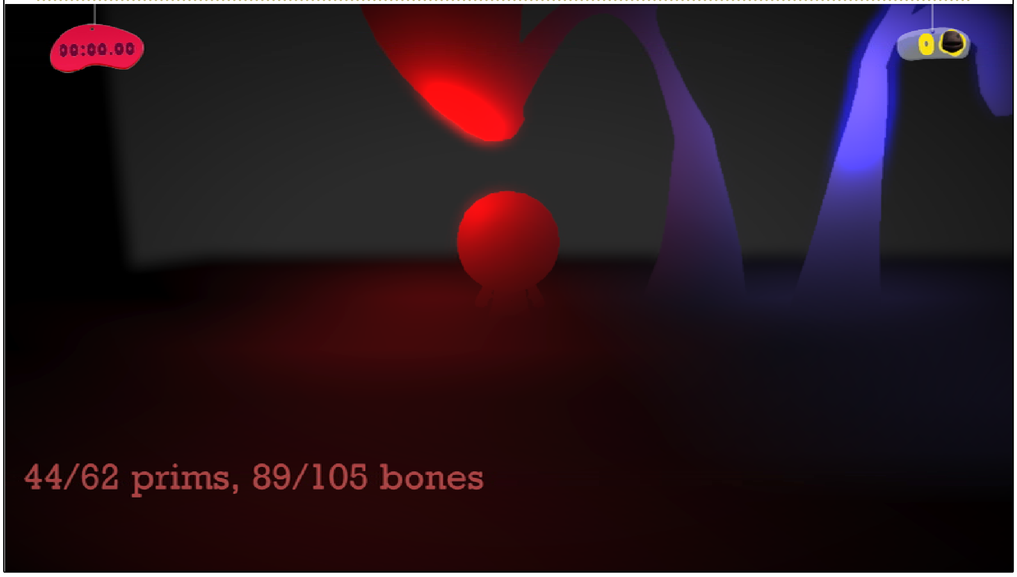
- We additively composite each visible light into the volume by intersecting its OBB with the slices of the volume.
- Irradiance is a function of both position and direction…
  - I(p,ω)
- The dependence on direction of the irradiance at a given point is often represented using spherical harmonics
  - We can get away with a simpler approximation: we first render just the average Irradiance over all directions, at each point in space…
  - Directional variation is then approximated from the grad (∇) of this ‘average irradiance’ field.

SIGGRAPH2006

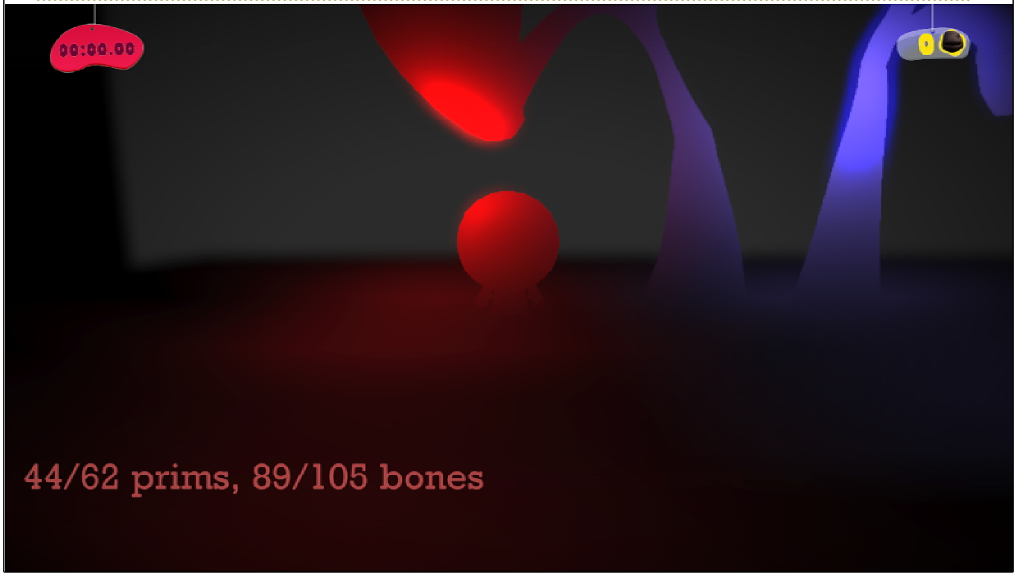
The scene lit by a single point source



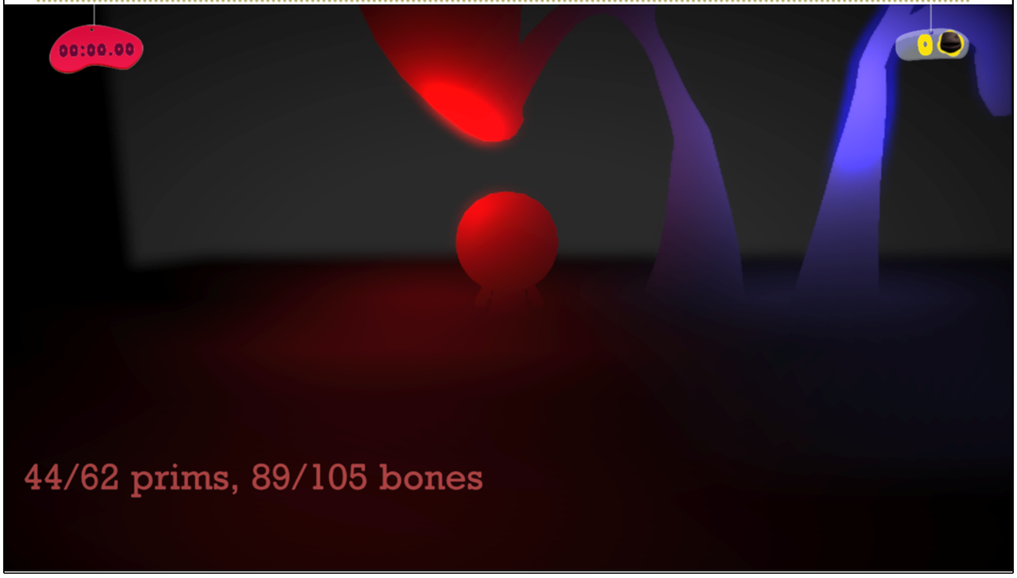
Irradiance volume visualisation



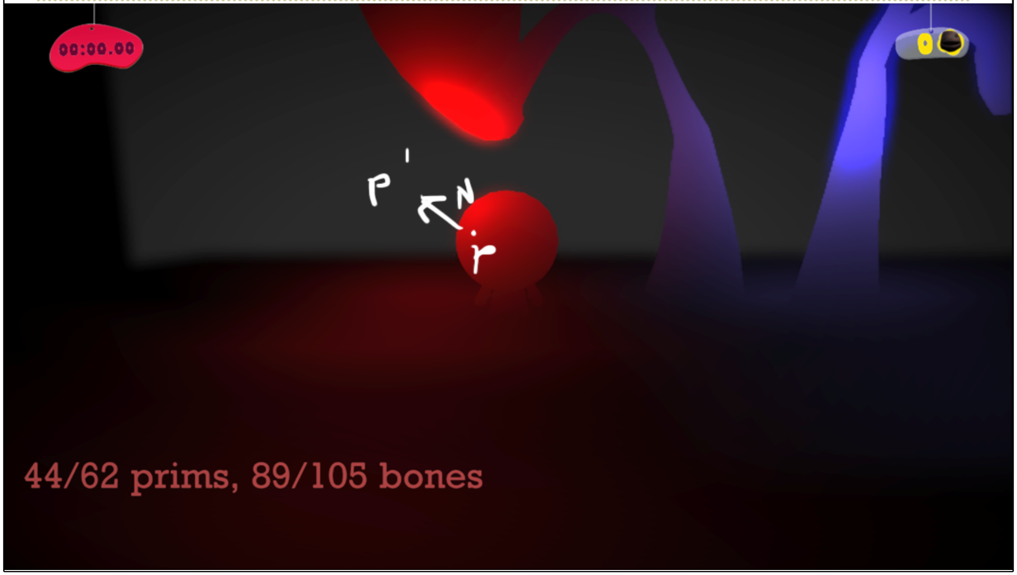
Using ∇ to approximate directional dependence



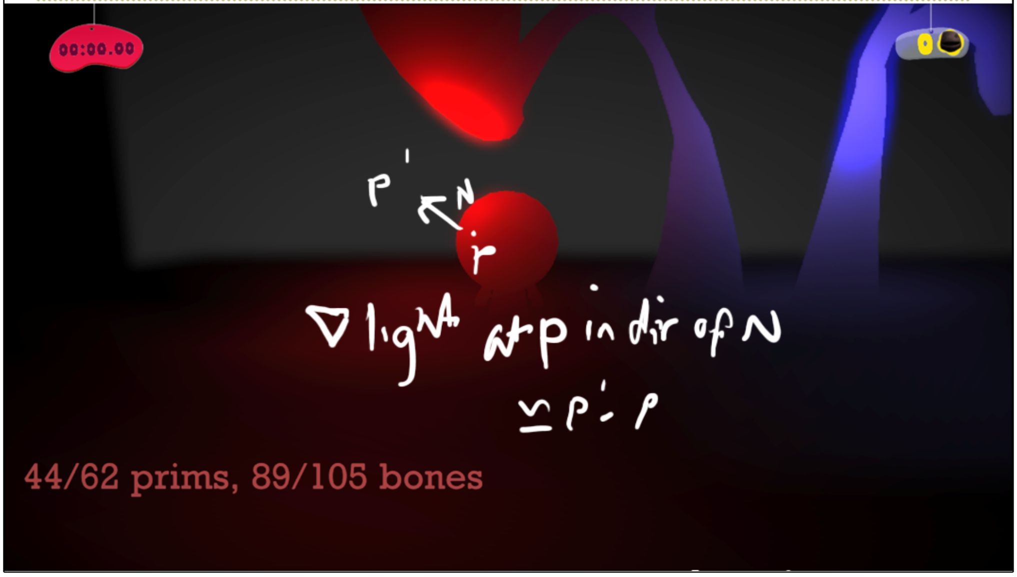
Using ∇ to approximate directional dependence



Using ∇ to approximate directional dependence



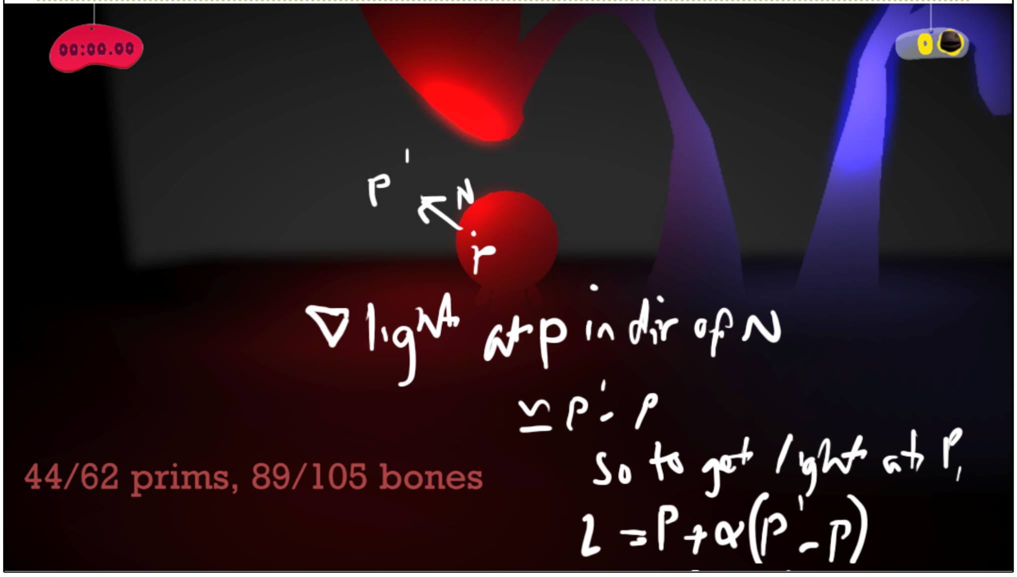
Using ∇ to approximate directional dependence



∇ light at P in dir of N

$\simeq P' - P$

Using ∇ to approximate directional dependence

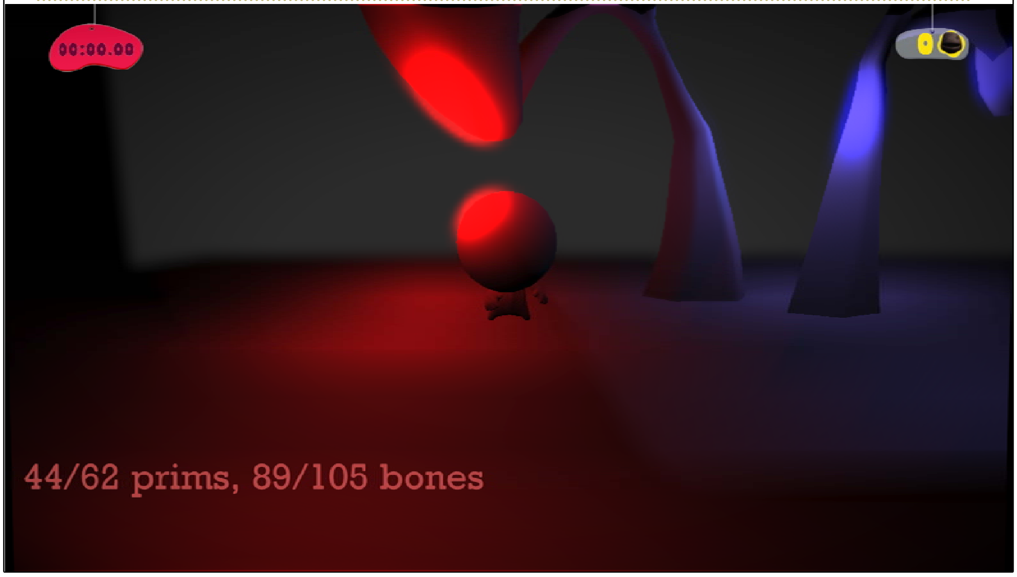


∇ light at P in dir of N $\simeq P' - P$

So to get light at P,

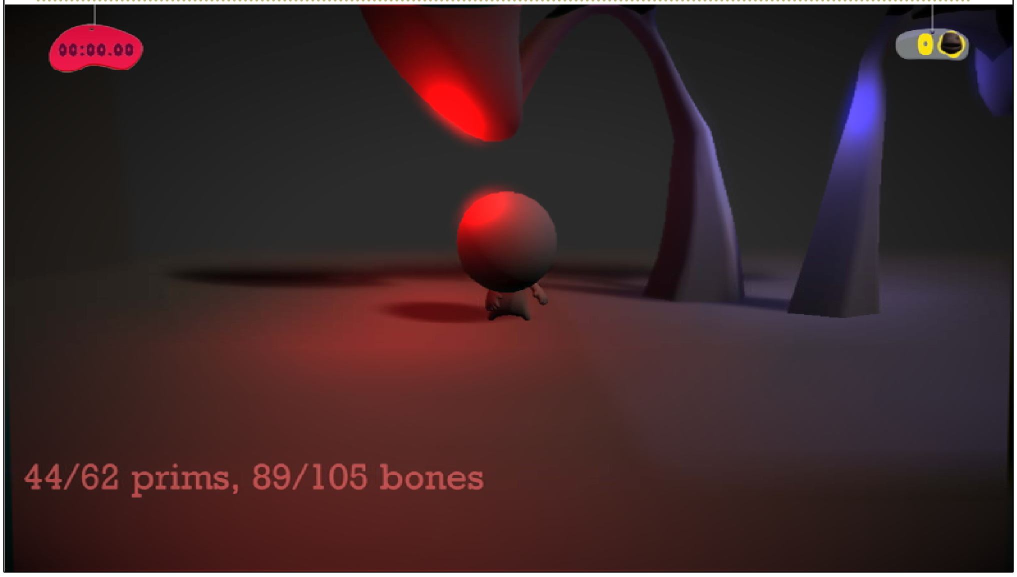
$L = P + \alpha(P' - P)$

Results (no sun)



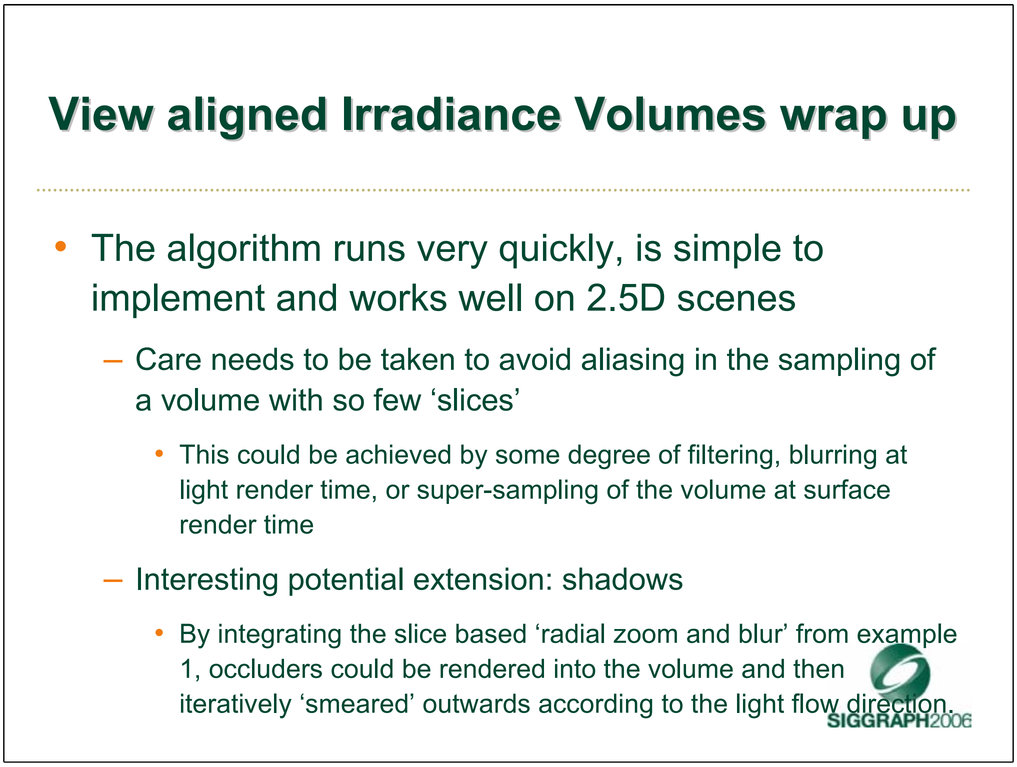
44/62 prims, 89/105 bones

Results (with sun)



44/62 prims, 89/105 bones

Video demo



View aligned Irradiance Volumes wrap up

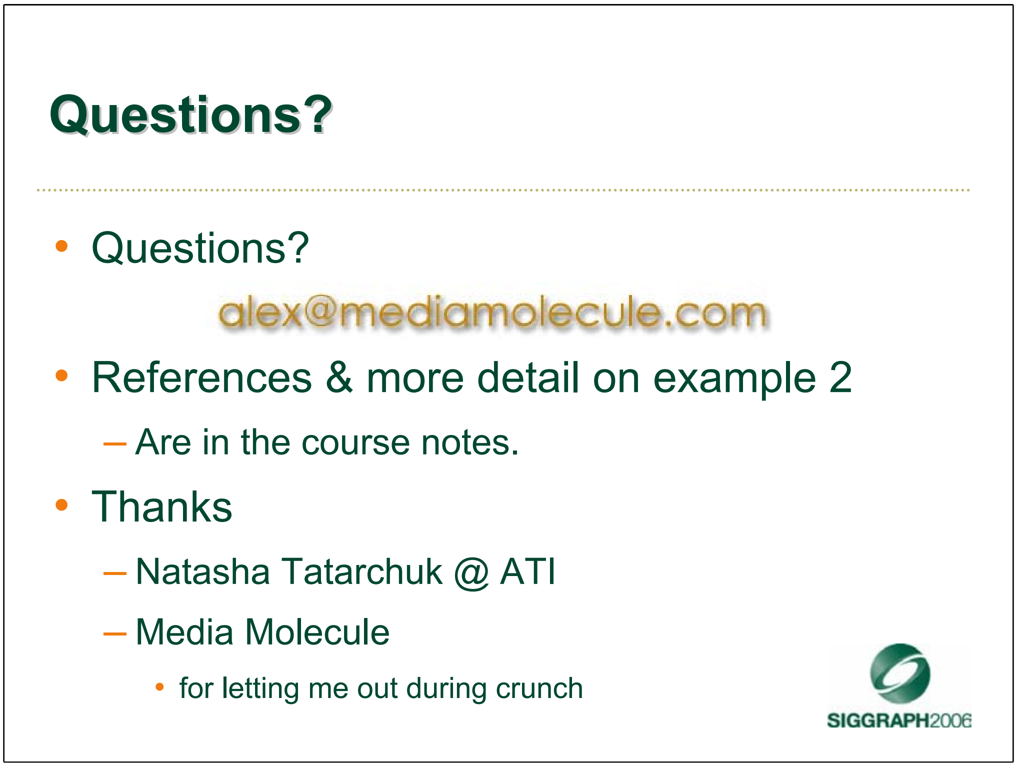
- The algorithm runs very quickly, is simple to implement and works well on 2.5D scenes
  - Care needs to be taken to avoid aliasing in the sampling of a volume with so few ‘slices’
    - This could be achieved by some degree of filtering, blurring at light render time, or super-sampling of the volume at surface render time
  - Interesting potential extension: shadows
    - By integrating the slice based ‘radial zoom and blur’ from example 1, occluders could be rendered into the volume and then iteratively ‘smeared’ outwards according to the light flow direction.

SIGGRAPH2006

Conclusion

- 3 novel techniques were presented using volume representations of small scenes to ‘get a nice look’
  - 1 – repeated zooming and blurring of slices through the scene can give convincing penumbra effects
  - 2 – a GPU updated volume texture was used to rapidly compute occlusion information from a ‘skylight’ to give a nice AO look with some bounce light effects
  - 3 – the gradient of a screen aligned ‘irradiance volume’ was used to rapidly compute lighting from a potentially large number of moving light sources.

SIGGRAPH2006



Questions?

- Questions?

  alex@mediamolecule.com

- References & more detail on example 2
  - Are in the course notes.
- Thanks
  - Natasha Tatarchuk @ ATI
  - Media Molecule
    - for letting me out during crunch

SIGGRAPH2006

lex lex mediamolecule.com