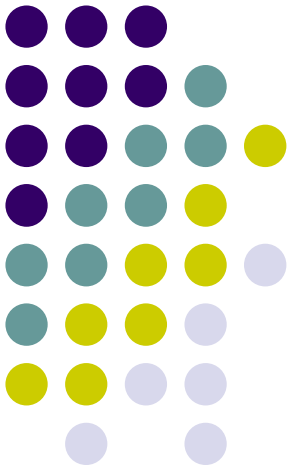
Advanced Computer Graphics
CS 563: *"Real-Time Volumetric Shadows using 1D Min-Max Mipmaps"*

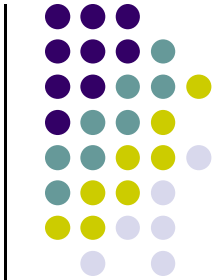
Rob Martin

Computer Science Dept.

Worcester Polytechnic Institute (WPI)

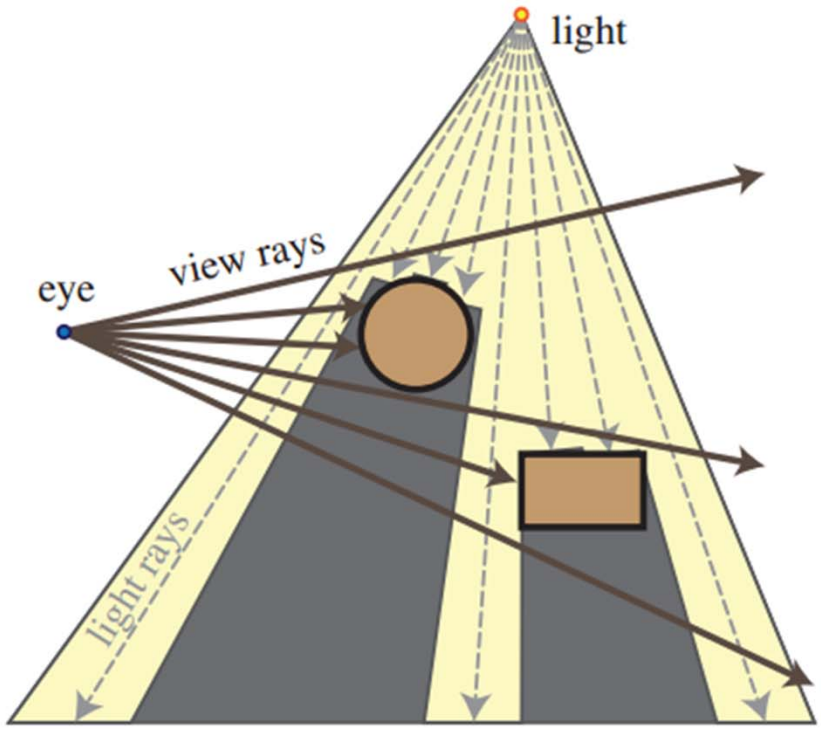


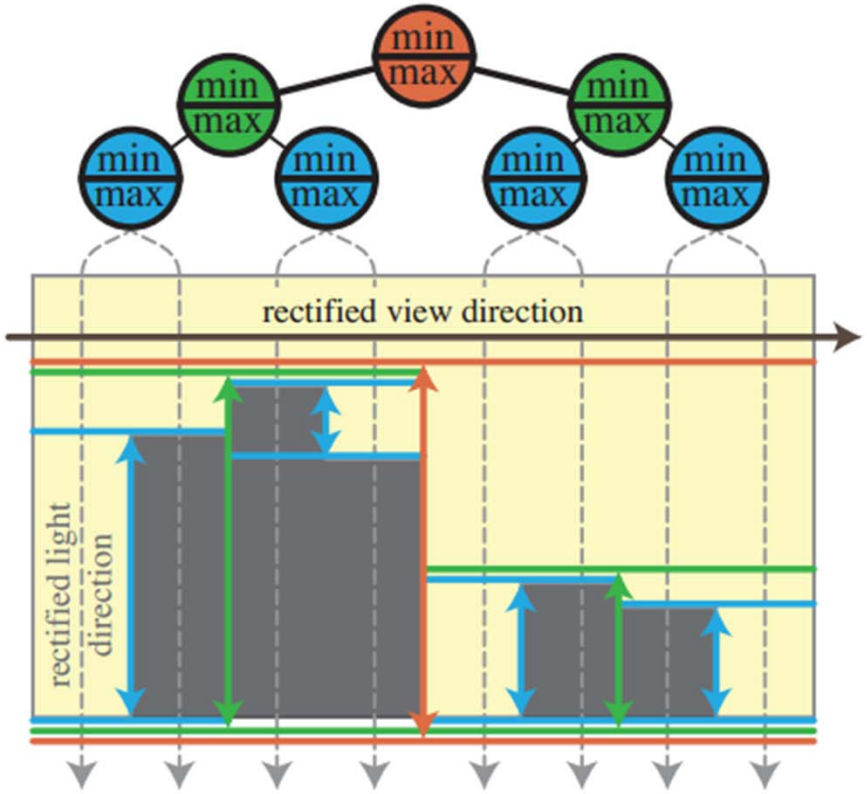
Introduction



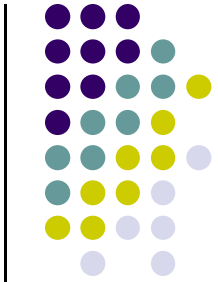
- We want to model the effects of light scattering in a *participating medium*
- Dust, moisture, etc.
- Difficult to do in real time since scattering occurs within a volume; i.e., all points along a camera ray
- This algorithm is an attempt at recreating this effect through the use of approximation techniques, epipolar shadow maps, min-map mipmaps, and parallelism.

Introduction



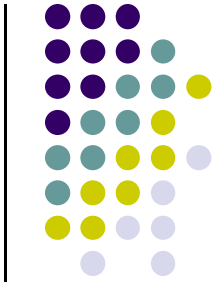


Assumptions



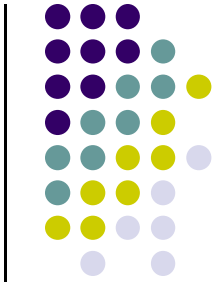
- Use single-scattering model [Blinn 1982]
  - Approximates more complex physics
  - Simple, realistic results
- Assume the scattering medium is homogeneously distributed.

Naïve approach: Ray marching



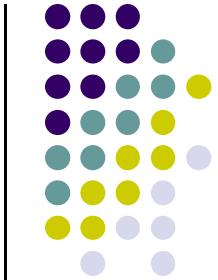
- Render shadow map from light source
- Shoot a ray from the camera for each pixel
- Walk down the ray at equal intervals from the camera to the first intersection
- Take sample at each interval
- Compute final radiosity from samples

Algorithm Overview



- Render depth map from the camera and a shadow map from the light source
- Perform ‘epipolar rectification’ on the shadow map
- Compute approximate scattering integral
- Build min-max mipmaps from shadow map
- Traverse mipmaps for each camera ray to determine lit segments
- (If epipole is on or near screen, use ray marching

Single Scattering



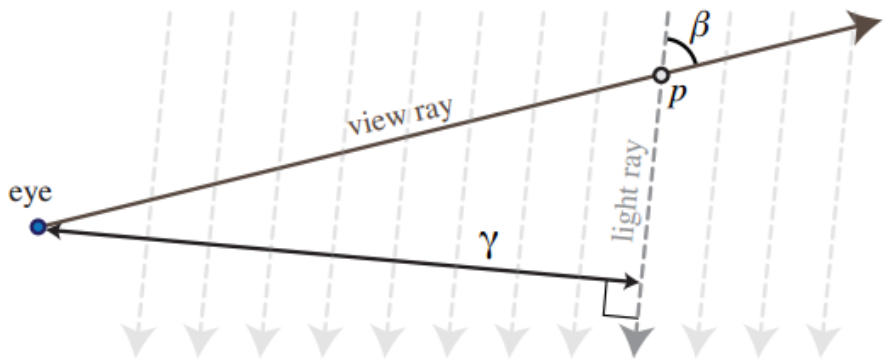
$$L(\mathbf{v}) = \int_0^d e^{-\sigma_t s}\, V(s\mathbf{v})\, \sigma_s\, \rho(\theta)\, L_{\text{in}}(s\mathbf{v})\, \mathrm{d}s, \qquad (1)$$

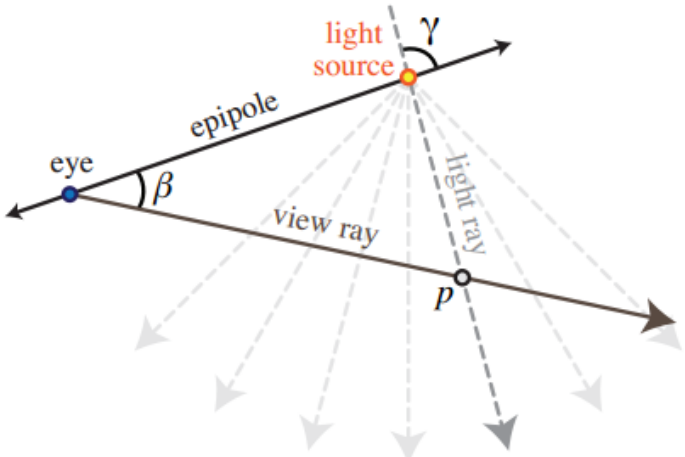
$$L_{\text{in}}(s\mathbf{v}) = \frac{I e^{-\sigma_t d(s\mathbf{v})}}{d(s\mathbf{v})^2}, \quad d(s\mathbf{v}) = \|\mathbf{x} - s\mathbf{v}\|, \qquad (2)$$

v : normalized direction vector
d : distance to first occluder
σ_s : scattering coef.
σ_t : σ_s - σ_a extinction coef.
V(s**v**) : 1 if in light, 0 otherwise
L_{in}(s**v**) : radience assuming no occusion
$\rho(\theta)$: scattering phase function

Epipolar Rectification

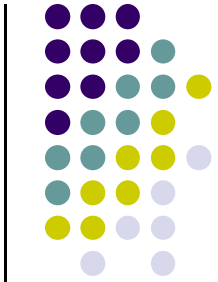
- Conversion from cartesian coordinate system to epipolar coordinates
- 3-dimensional space
- Every point has α,β,γ
- Camera rays are indexed by (α,β)
- Light rays by (α,γ)
- α determines the ‘slice’



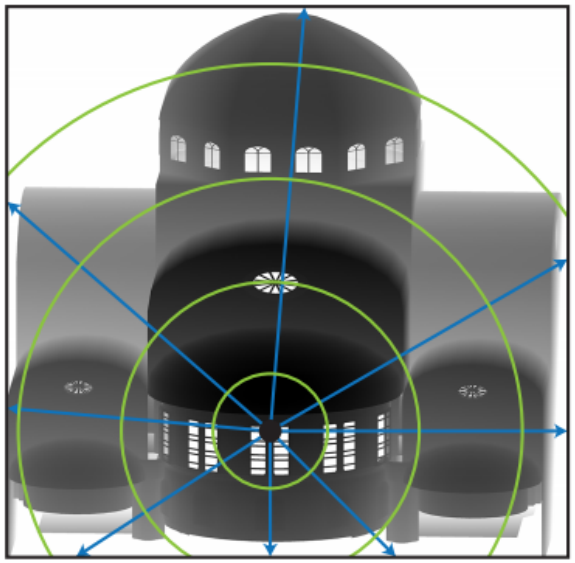


Draw me

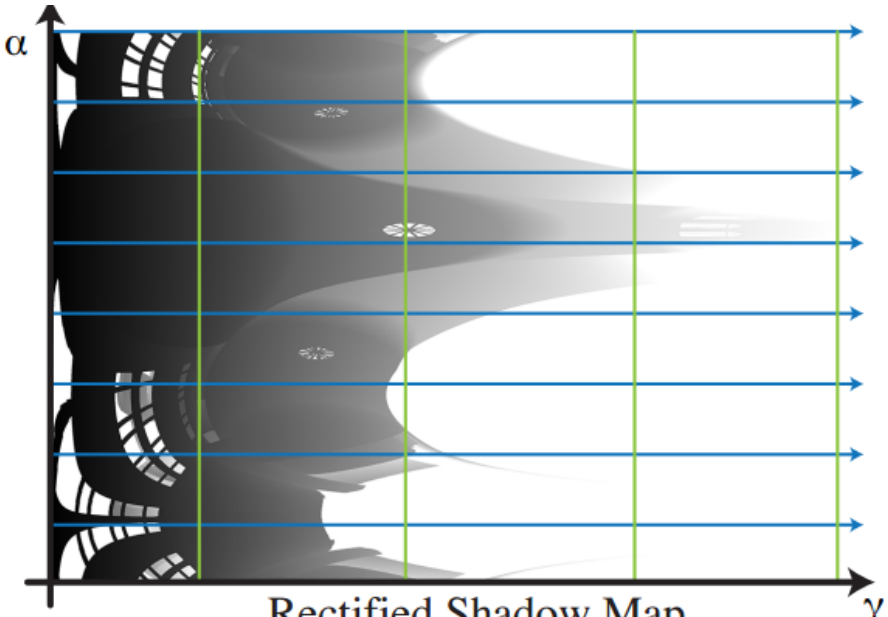
Epipolar Rectification, cont.



- To rectify the shadow map, rows are indexed by α, columns by γ; each pixel contains the β
- Do not need to rectify the camera

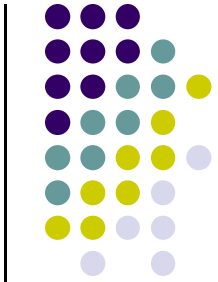


Shadow Map



Rectified Shadow Map

Revamped Scattering Integral

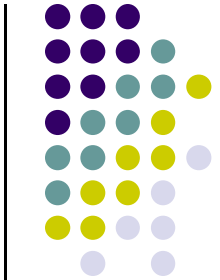


$$L(\alpha,\beta) = \int_0^{D(\alpha,\beta)} e^{-\sigma_t s(\beta,\gamma)} V(\alpha,\beta,\gamma)\,\sigma_s\,\rho(\beta,\gamma)\,L_{\text{in}}(\beta,\gamma)\,\frac{\mathrm{d}s}{\mathrm{d}\gamma}\mathrm{d}\gamma, \quad (3)$$

$$L(\alpha,\beta) = \int_0^{D(\alpha,\beta)} V(\alpha,\beta,\gamma)\,\mathcal{I}(\beta,\gamma)\mathrm{d}\gamma, \quad (4)$$

D(α,β) : the γ coordinate where the camera ray hits the first occluder

Integral approximations

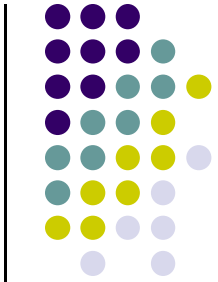


$$L(\alpha,\beta) \approx \sum_{i}^{N} \left(\mathrm{B}_i(\beta) \int_{0}^{D(\alpha,\beta)} V(\alpha,\beta,\gamma)\Gamma_i(\gamma)\mathrm{d}\gamma \right). \quad (5)$$

$$L(\alpha,\beta) \approx \sum_{i}^{N} \left(\mathrm{B}_i(\beta) \sum_{\substack{\gamma<D(\alpha,\beta) \\ S[\alpha,\gamma]>\beta}} \Gamma_i(\gamma)\Delta\gamma \right), \quad (6)$$

- The I term is too complex to compute, so approximate by sampling in a 64x64 grid
- Approximate integral in equation 5 by using a Riemann sum; integrate in the visibility term (V)

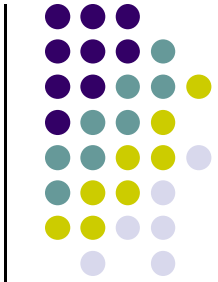
Textured Lights



$$L(\alpha,\gamma) \approx \sum_{i}^{N} \left(\mathrm{B}_i(\beta) \sum_{\substack{\gamma < D(\alpha,\beta) \\ S[\alpha,\gamma] > \beta}} \Gamma_i(\gamma) T(\alpha,\gamma) \Delta\gamma \right). \qquad (7)$$

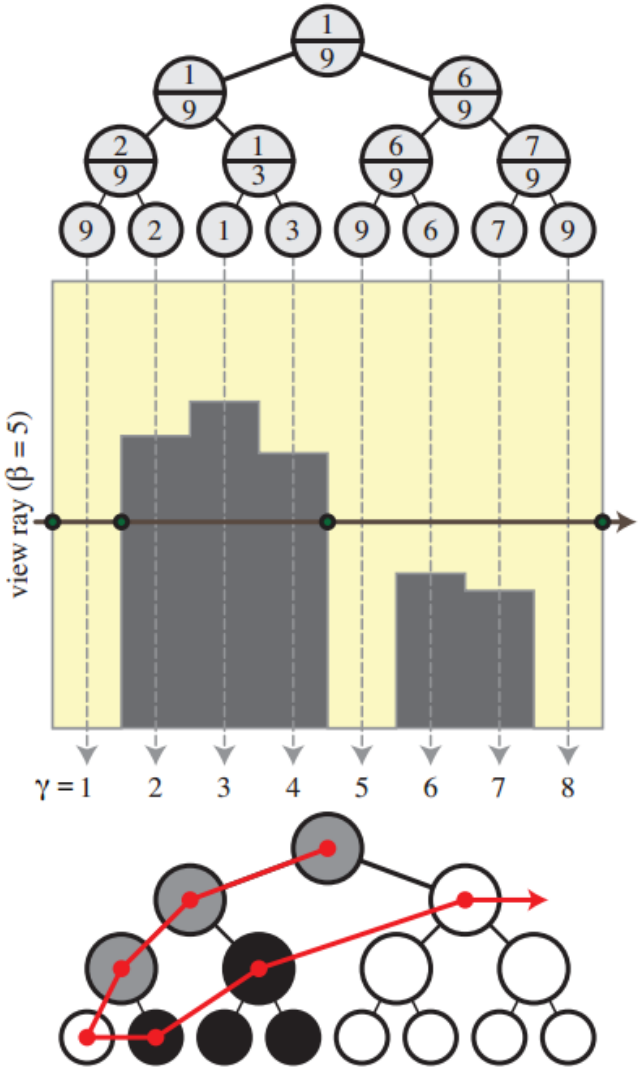
- To enable textured lights, must also rectify the texture map
- New texture coordinates given by $\mathrm{T}(\alpha,\gamma)$
- Must compute prefix sums of $\Gamma_i(\gamma)\mathrm{T}(\alpha,\gamma)$
- Must do this for each α

1D Min-Max Mipmaps



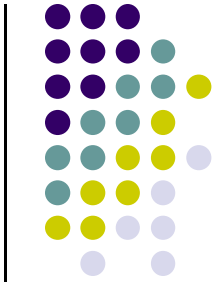
- Binary tree structure
- Each node holds the minimum and maximum value of all child nodes
- Each row of the rectified shadow map corresponds to an α slice in epipolar coordinates
- Create a min-max mipmap for each row/slice

Min-Max Mipmaps, cont.



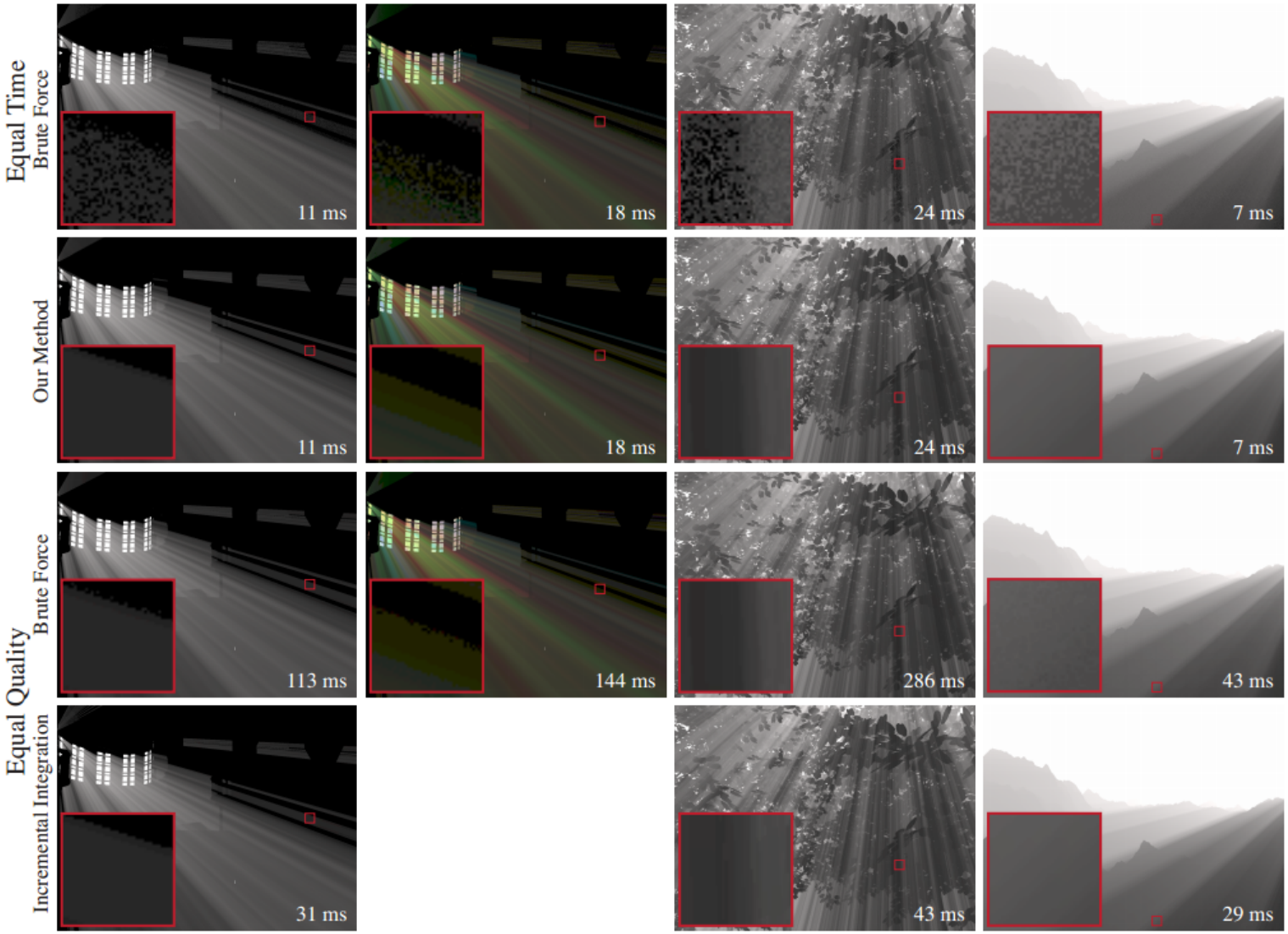
- Use the mipmaps to quickly find lit/unlit areas of the camera ray
- For a ray with angle β, if β is less than minimum, then the area is lit. If greater than the maximum, the area is in shadow.

Parallelism

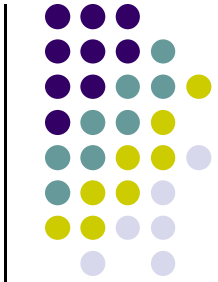


- With the rectified shadow maps, min-max mipmaps, and precomputed scattering integrals, each camera ray can be computed in parallel on a fragment/pixel shader
- Key for algorithm performance

Results

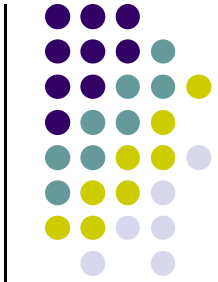

Equal Time
Brute Force
11 ms
18 ms
24 ms
7 ms
Our Method
11 ms
18 ms
24 ms
7 ms
Equal Quality
Brute Force
113 ms
144 ms
286 ms
43 ms
Incremental Integration
31 ms
43 ms
29 ms

Results, cont.



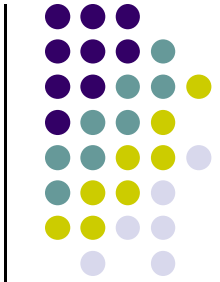
- Incremental integration is a very similar algorithm
- Main difference is the data structure used to detect lit/unlit areas
- Uses a dynamic data structure called a partial sum tree
- Is not static, nor is it parallelizable

Limitations



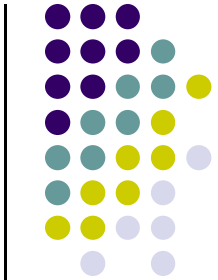
- Homogeneous scattering medium
- Anisotropic mediums mess up the integral approximations
- Shadow maps are susceptible to aliasing
- Must brute force near the epipole

Conclusions



- Effective real-time algorithm for producing the effect of shadow volumes and participating medium
- Mathematical techniques simplify computations by pre-computing static values used in the integration
- Simplified data structure allows for parallism
- Algorithm can run on most hardware (DX9)

References



- Blinn, J. F. 1982. Light reflection functions for simulation of clouds and dusty surfaces. 21–29
- Jiawen Chen, Ilya Baran, Fredo Durand, Wojciech Jarosz. *Real-Time Volumetric Shadows using 1D Min-Max Mipmaps*. Proceedings of the 2011 Symposium on Interactive 3D Graphics and Games, 2011.