A Lighting Model for Fast Rendering of Forest Ecosystems

Robert Geist, Jay Steele

Clemson University
Rendering Synthetic Ecosystems

Of interest in:

- architectural planning
- landscape design
- forest management
- special effects
Goal

Extend previous ray tracing approaches to include:

- diffuse leaf transparency
- inter-object light scattering
Goal

Extend previous ray tracing approaches to include:
- diffuse leaf transparency
- inter-object light scattering

while maintaining near real-time performance for scenes with hundreds of millions of primitives.
Approach draws principally from:


Use a lattice-Boltzmann solution to the volume radiative transfer equation:

\[
(\vec{\omega} \cdot \nabla + \sigma_t) L(\vec{x}, \vec{\omega}) = \sigma_s \int p(\vec{\omega}, \vec{\omega}') L(\vec{x}, \vec{\omega}') d\omega' + Q(\vec{x}, \vec{\omega})
\]

- \( L \) radiance
- \( \vec{\omega} \) spherical direction
- \( p(\vec{\omega}, \vec{\omega}') \) phase function
- \( \sigma_s/\sigma_a \) scattering/absorption coefficients
- \( \sigma_t = \sigma_s + \sigma_a \)
- \( Q(\vec{x}, \vec{\omega}) \) emissive field (in the volume)
Lattice-Boltzmann Methods

- computational alternatives to finite-element/finite-difference methods for solving PDEs

- advantages:
  - ease of implementation
  - ease of parallelization
  - ease of handling complex boundary conditions

- disadvantage: derivation (proof) can be "tedious"
Lattice-Boltzmann Methods

- computational alternatives to finite-element/finite-difference methods for solving PDEs
- simulate transport by tracing evolution of particle distributions through synchronous updates on discrete grid
Lattice-Boltzmann Methods

- computational alternatives to finite-element/finite-difference methods for solving PDEs
- simulate transport by tracing evolution of particle distributions through synchronous updates on discrete grid
- comparable (FEM) speed, stability, accuracy, storage

Advantages:
- ease of implementation
- ease of parallelization
- ease of handling complex boundary conditions

Disadvantage: derivation (proof) can be "tedious"
Lattice-Boltzmann Methods

- computational alternatives to finite-element/ finite-difference methods for solving PDEs
- simulate transport by tracing evolution of particle distributions through synchronous updates on discrete grid
- comparable (FEM) speed, stability, accuracy, storage

- advantages:
  - ease of implementation
  - ease of parallelization
  - ease of handling complex boundary conditions
Lattice-Boltzmann Methods

- computational alternatives to finite-element/finite-difference methods for solving PDEs
- simulate transport by tracing evolution of particle distributions through synchronous updates on discrete grid
- comparable (FEM) speed, stability, accuracy, storage
- advantages:
  - ease of implementation
  - ease of parallelization
  - ease of handling complex boundary conditions
- disadvantage: derivation (proof) can be “tedious”
Lattice-Boltzmann 3D Lighting

- use grid with 19 directions: all lattice points of a cube of radius 1, minus the corners

key quantity of interest: per-site photon density, $f_{\mathbf{r}; t}$ = density arriving at lattice site $\mathbf{r}$ at time $t$ in cube direction $\mathbf{c}$

update: for lattice spacing, $\Delta$, time step $\Delta t$, update is $f_{\mathbf{r} + \mathbf{c} m; t + \Delta t} = f_{\mathbf{r}; t}$

where $m$ denotes row $m$ of a $19 \times 19$ matrix, that describes scattering, absorption, and (perhaps) wavelength shift at each site

this is the entire model!
Lattice-Boltzmann 3D Lighting

- Use grid with 19 directions: all lattice points of a cube of radius 1, minus the corners.

- Key quantity of interest: per-site photon density,
  \[ f_m(\vec{r}, t) = \text{density arriving at lattice site } \vec{r} \in \mathbb{R}^3 \text{ at time } t \text{ in cube direction } \vec{c}_m, m \in \{0, 1, \ldots, 18\} \]
Lattice-Boltzmann 3D Lighting

- use grid with 19 directions: all lattice points of a cube of radius 1, minus the corners
- key quantity of interest: per-site photon density, $f_m(\vec{r}, t) = $ density arriving at lattice site $\vec{r} \in \mathbb{R}^3$ at time $t$ in cube direction $\vec{c}_m$, $m \in \{0, 1, \ldots, 18\}$
- update: for lattice spacing, $\lambda$, time step $\tau$, update is

$$f_m(\vec{r} + \lambda \vec{c}_m, t + \tau) - f_m(\vec{r}, t) = \Omega_m \cdot f(\vec{r}, t)$$
use grid with 19 directions: all lattice points of a cube of radius 1, minus the corners

key quantity of interest: per-site photon density,
\[ f_m(\vec{r}, t) = \text{density arriving at lattice site } \vec{r} \in \mathbb{R}^3 \text{ at time } t \text{ in cube direction } \vec{c}_m, m \in \{0, 1, \ldots, 18\} \]

update: for lattice spacing, \( \lambda \), time step \( \tau \), update is

\[ f_m(\vec{r} + \lambda \vec{c}_m, t + \tau) - f_m(\vec{r}, t) = \Omega_m \cdot f(\vec{r}, t) \]

where \( \Omega_m \) denotes row \( m \) of a 19 \times 19 matrix, \( \Omega \), that describes scattering, absorption, and (perhaps) wavelength shift at each site
use grid with 19 directions: all lattice points of a cube of radius 1, minus the corners

key quantity of interest: per-site photon density,
\[ f_m(\vec{r}, t) = \text{density arriving at lattice site } \vec{r} \in \mathbb{R}^3 \text{ at time } t \text{ in cube direction } \vec{c}_m, m \in \{0, 1, ..., 18\} \]

update: for lattice spacing, \( \lambda \), time step \( \tau \), update is
\[
f_m(\vec{r} + \lambda \vec{c}_m, t + \tau) - f_m(\vec{r}, t) = \Omega_m \cdot f(\vec{r}, t)
\]
where \( \Omega_m \) denotes row \( m \) of a \( 19 \times 19 \) matrix, \( \Omega \), that describes scattering, absorption, and (perhaps) wavelength shift at each site

this is the entire model!
Ω not unique!

In general,

- conserve mass, \( \sum_m (\Omega_m \cdot f) = 0 \)
In general,

- conserve mass, \( \sum_m (\Omega_m \cdot f) = 0 \)
- conserve momentum, \( \sum_m (\Omega_m \cdot f) \vec{v}_m = 0 \), where 
  \[ \vec{v}_m = (\lambda/\tau) \vec{c}_m \]
Omega not unique!

In general,

- conserve mass, $\sum_m (\Omega_m \cdot f) = 0$
- conserve momentum, $\sum_m (\Omega_m \cdot f) \vec{v}_m = 0$, where $\vec{v}_m = (\lambda/\tau) \vec{c}_m$
- $\Omega_{i,j}$ controls scattering from direction $\vec{c}_j$ into direction $\vec{c}_i$
In general,

- conserve mass, \( \sum_m (\Omega_m \cdot f) = 0 \)
- conserve momentum, \( \sum_m (\Omega_m \cdot f) \nu_m = 0 \), where
  \( \nu_m = (\lambda/\tau) c_m \)
- \( \Omega_{i,j} \) controls scattering from direction \( c_j \) into direction \( c_i \)
- directional density \( f_0 \) holds the absorption/emission
Lighting Model (isotropic case)

\[ \Omega_{0j} = \begin{cases} 
-1 & j = 0 \\
\sigma_a & j > 0
\end{cases} \]

\[ i = 1, \ldots, 6 : \quad \Omega_{ij} = \begin{cases} 
\frac{1}{12} & j = 0 \\
\sigma_s/12 & j > 0, \ j \neq i \\
-\sigma_t + \sigma_s/12, & j = i
\end{cases} \]

\[ i = 7, \ldots, 18 : \quad \Omega_{ij} = \begin{cases} 
\frac{1}{24} & j = 0 \\
\sigma_s/24 & j > 0, \ j \neq i \\
-\sigma_t + \sigma_s/24, & j = i
\end{cases} \]
Lighting Model (derivation)

- If \( \rho(\vec{r}, t) = \sum_m f_m(\vec{r}, t) \), limiting case of

\[
f_m(\vec{r} + \lambda \vec{c}_m, t + \tau) - f_m(\vec{r}, t) = \Omega_m \cdot f(\vec{r}, t)
\]

as \( \lambda, \tau \to 0 \) is
If $\rho(\vec{r}, t) = \sum_{m} f_{m}(\vec{r}, t)$, limiting case of

$$f_{m}(\vec{r} + \lambda \vec{c}_{m}, t + \tau) - f_{m}(\vec{r}, t) = \Omega_{m} \cdot f(\vec{r}, t)$$

as $\lambda, \tau \to 0$ is

$$\frac{\partial \rho}{\partial t} = D \nabla^{2}_{\vec{r}} \rho$$
If \( \rho(\vec{r}, t) = \sum_m f_m(\vec{r}, t) \), limiting case of

\[
f_m(\vec{r} + \lambda \vec{c}_m, t + \tau) - f_m(\vec{r}, t) = \Omega_m \cdot f(\vec{r}, t)
\]

as \( \lambda, \tau \to 0 \) is

\[
\frac{\partial \rho}{\partial t} = D \nabla^2 \rho
\]

where the diffusion coefficient

\[
D = \left( \frac{\lambda^2}{\tau} \right) \left[ \frac{(2/\sigma_t) - 1}{4(1 + \sigma_a)} \right]
\]
Lighting Model (derivation)

If \( \rho(\vec{r}, t) = \sum_m f_m(\vec{r}, t) \), limiting case of

\[
f_m(\vec{r} + \lambda \vec{c}_m, t + \tau) - f_m(\vec{r}, t) = \Omega_m \cdot f(\vec{r}, t)
\]

as \( \lambda, \tau \to 0 \) is

\[
\frac{\partial \rho}{\partial t} = D \nabla^2 \rho
\]

where the diffusion coefficient

\[
D = \left( \frac{\lambda^2}{\tau} \right) \left[ \frac{(2/\sigma_t) - 1}{4(1 + \sigma_a)} \right]
\]

consistent with previous approaches to modeling multiple photon scattering events
enclose each tree ("leaf cloud") in a $128^3$ lattice
Lighting Model (application)

- enclose each tree ("leaf cloud") in a $128^3$ lattice
- multiply entries of $\Omega$ by mean biomass density per lattice site
  - density 0 yields straight pass-through
  - density 1 yields full scattering
enclose each tree ("leaf cloud") in a $128^3$ lattice

multiply entries of $\Omega$ by mean biomass density per lattice site

- density 0 yields straight pass-through
- density 1 yields full scattering

label each site "green" (allow forward scattering) or "brown" (backscattering only)
Lighting Model (application)

- enclose each tree (“leaf cloud”) in a $128^3$ lattice
- multiply entries of $\Omega$ by mean biomass density per lattice site
  - density 0 yields straight pass-through
  - density 1 yields full scattering
- label each site “green” (allow forward scattering) or “brown” (backscattering only)
- still must determine $\sigma_a$ and $\sigma_s$
Capturing Leaf Transparency

- absorption, reflection, transmission are wavelength dependent


Restrict wavelength dependence to three components scale absorptance values from Knapp and Carter to obtain per-component model absorption coefficients, \( X_a, X_g, X_b \); then \( X_s = 1 \).
Capturing Leaf Transparency

- absorption, reflection, transmission are wavelength dependent
- species dependent?


Restrict wavelength dependence to three components: scale absorptance values from Knapp and Carter to obtain per-component model absorption coefficients, $X_a$, $X_s = 1 - X_a$, then
absorption, reflection, transmission are wavelength dependent

species dependent?

Knapp and Carter (Am. J. Botany 1998): amazing lack of variability across a wide range of species from a wide range of habitats

conclude: single set of wavelength dependent parameters will suffice to determine $\sigma_a$ and $\sigma_s$
Capturing Leaf Transparency

- absorption, reflection, transmission are wavelength dependent
- species dependent?
  - Knapp and Carter (Am. J. Botany 1998): amazing lack of variability across a wide range of species from a wide range of habitats
  - conclude: single set of wavelength dependent parameters will suffice to determine $\sigma_a$ and $\sigma_s$
- restrict wavelength dependence to three components
Capturing Leaf Transparency

- absorption, reflection, transmission are wavelength dependent
- species dependent?
  - Knapp and Carter (Am. J. Botany 1998): amazing lack of variability across a wide range of species from a wide range of habitats
  - conclude: single set of wavelength dependent parameters will suffice to determine $\sigma_a$ and $\sigma_s$
- restrict wavelength dependence to three components
- scale absorptance values from Knapp and Carter to obtain per-component model absorption coefficients, $\sigma_a^X, X = R, G, B$; then $\sigma_s^X = 1 - \sigma_a^X$
Capturing Leaf Transparency

- scattering is anisotropic (and wavelength dependent)
Capturing Leaf Transparency

- scattering is anisotropic (and wavelength dependent)
- multiply $\sigma_s$ in $\Omega_{i,j}$ by normalized phase function:

$$p_{n_{i,j}}(g) = \frac{p_{i,j}(g)}{\left(\sum_{i=1}^{6} 2p_{i,j}(g) + \sum_{i=7}^{18} p_{i,j}(g)\right) / 24}$$
Capturing Leaf Transparency

- scattering is anisotropic (and wavelength dependent)
- multiply $\sigma_s$ in $\Omega_{i,j}$ by normalized phase function:

$$pn_{i,j}(g) = \frac{p_{i,j}(g)}{\left( \sum_{i=1}^{6} 2p_{i,j}(g) + \sum_{i=7}^{18} p_{i,j}(g) \right) / 24}$$

where (Henyey-Greenstein)

$$p_{i,j}(g) = \frac{1 - g^2}{(1 - 2gn_i \cdot n_j + g^2)^{3/2}}$$

$n_i$ is normalized direction, $\vec{c}_i$; $g \in [-1, 1]$ controls scattering direction
Capturing Leaf Transparency

per-component phase function parameter ($g$) values:

transmittance and reflectance ratios from Knapp and Carter determine forward and backward scattering components by constraint:

$$f_s X + b_s X = S$$

normalize:

$$g X = \frac{f_s X}{b_s X}$$

for $X = R, G, B$

note: identical transmittance and reflectance values for color component $X$ yield isotropic scattering if node is classified as “brown,” $g X = 1$ all $X$
Capturing Leaf Transparency

per-component phase function parameter \((g)\) values:

- transmittance and reflectance ratios from Knapp and Carter determine forward and backward scattering components by constraint: 
  \[
  f s^X + b s^X = \sigma^X_S
  \]
  
  note: identical transmittance and reflectance values for color component 
  \(X\) yield isotropic scattering if node is classified as “brown,” 
  \[g^X = 1\] all 
  \(X\)
Capturing Leaf Transparency

per-component phase function parameter \((g)\) values:

- transmittance and reflectance ratios from Knapp and Carter determine forward and backward scattering components by constraint: 
  \[ f s^X + b s^X = \sigma_X^X \]
- normalize:
  \[ g^X = \frac{f s^X - b s^X}{f s^X + b s^X} \quad \text{for } X = R, G, B \]
Capturing Leaf Transparency

per-component phase function parameter \((g)\) values:

- Transmittance and reflectance ratios from Knapp and Carter determine forward and backward scattering components by constraint:
  \[ f s^X + b s^X = \sigma_s^X \]

- Normalize:
  \[ g^X = \frac{f s^X - b s^X}{f s^X + b s^X} \quad \text{for } X = R, G, B \]

- Note: identical transmittance and reflectance values for color component \(X\) yield isotropic scattering
Capturing Leaf Transparency

per-component phase function parameter \((g)\) values:

- transmittance and reflectance ratios from Knapp and Carter determine forward and backward scattering components by constraint: 
  \[
  f s^X + b s^X = \sigma_S^X
  \]

- normalize:
  \[
  g^X = \frac{f s^X - b s^X}{f s^X + b s^X}
  \]
  for \(X = R, G, B\)

- note: identical transmittance and reflectance values for color component \(X\) yield isotropic scattering

- if node is classified as “brown,” \(g^X = -1\) all \(X\)
Lighting Model (implementation)

- run LB lighting model (CUDA) to steady-state as pre-processing step; store values
Lighting Model (implementation)

- run LB lighting model (CUDA) to steady-state as pre-processing step; store values
- ray trace (CUDA)
Lighting Model (implementation)

- run LB lighting model (CUDA) to steady-state as pre-processing step; store values
- ray trace (CUDA)
- at each intersection point, read LB values at surrounding lattice nodes and interpolate
run LB lighting model (CUDA) to steady-state as pre-processing step; store values

- ray trace (CUDA)

- at each intersection point, read LB values at surrounding lattice nodes and interpolate

- modulate LB value with texture and add to standard, local illumination
CUDA basics ...

- code organized around kernels, functions invoked on CPU, executed on GPU
- kernels invoked simultaneously by multiple threads
- threads organized (by programmer) into blocks
- each block is mapped to a multiprocessor (8 cores)
- minimum scheduling unit is a warp (32 threads)
- each MP executes a warp in 4 clock cycles
CUDA basics ...

- code organized around *kernels*, functions invoked on CPU, executed on GPU
- kernels invoked simultaneously by multiple threads
- threads organized (by programmer) into *blocks*
- each block is mapped to a *multiprocessor* (8 cores)
- minimum scheduling unit is a *warp* (32 threads)
- each MP executes a warp in 4 clock cycles
- memory management important!
CUDA basics ...

- code organized around *kernels*, functions invoked on CPU, executed on GPU
- kernels invoked simultaneously by multiple threads
- threads organized (by programmer) into *blocks*
- each block is mapped to a *multiprocessor* (8 cores)
- minimum scheduling unit is a *warp* (32 threads)
- each MP executes a warp in 4 clock cycles
- memory management important!
- avoid control flow divergence within warps!
CUDA Lattice-Boltzmann

straightforward...
CUDA Lattice-Boltzmann

straightforward...

- $128^3$ threads
- update is synchronous matrix multiplication per site
- effectively zero control flow divergence
- entire model fits in device memory
CUDA Ray Tracing

- store all kd trees in texture memory (cached)
CUDA Ray Tracing

- store all kd trees in texture memory (cached)
- thread block (one thread per ray) will trace $8 \times 8$ tile; warp receives $8 \times 4$ tile
CUDA Ray Tracing

- store all kd trees in texture memory (cached)
- thread block (one thread per ray) will trace $8 \times 8$ tile; warp receives $8 \times 4$ tile
- kd traversal by *short-stack*; stack size 5, stored in shared memory
CUDA Ray Tracing

- store all kd trees in texture memory (cached)
- thread block (one thread per ray) will trace $8 \times 8$ tile; warp receives $8 \times 4$ tile
- kd traversal by *short-stack*; stack size 5, stored in shared memory
- 4 kernels:
  - primary rays (leaf shape from alpha of texture)
  - shadow rays
  - shading
  - tone mapping and down-sampling
CUDA Ray Tracing

- store all kd trees in texture memory (cached)
- thread block (one thread per ray) will trace $8 \times 8$ tile; warp receives $8 \times 4$ tile
- kd traversal by *short-stack*; stack size 5, stored in shared memory
- 4 kernels:
  - primary rays (leaf shape from alpha of texture)
  - shadow rays
  - shading
  - tone mapping and down-sampling
- OpenMPI distributes across multiple GPUs
Results

- full LB scattering
- local plus ambient
Results

local illumination only  volume visualization of LB
Results (Beech Forest Scene)
Results (Pine Forest Scene)
## Beech Forest Scene Composition

<table>
<thead>
<tr>
<th>species</th>
<th>instances</th>
<th>triangles/instance</th>
</tr>
</thead>
<tbody>
<tr>
<td>red maple</td>
<td>12</td>
<td>115,529</td>
</tr>
<tr>
<td>ohio buckeye</td>
<td>285</td>
<td>168,520</td>
</tr>
<tr>
<td>paper birch</td>
<td>291</td>
<td>372,896</td>
</tr>
<tr>
<td>southern catalpa</td>
<td>206</td>
<td>155,342</td>
</tr>
<tr>
<td>american beech</td>
<td>168</td>
<td>496,719</td>
</tr>
<tr>
<td><strong>total scene</strong></td>
<td><strong>962</strong></td>
<td><strong>273,376,528</strong></td>
</tr>
</tbody>
</table>
## Beech Forest Scene Execution Time

<table>
<thead>
<tr>
<th>Platform</th>
<th>1 ray/pixel</th>
<th>4 rays/pixel</th>
<th>LB lighting</th>
</tr>
</thead>
<tbody>
<tr>
<td>G80</td>
<td>2.277 s</td>
<td>8.044 s</td>
<td>32.1 s</td>
</tr>
<tr>
<td>G200 EES</td>
<td>1.151 s</td>
<td>-</td>
<td>15.9 s</td>
</tr>
</tbody>
</table>
## Beech Forest Scene Execution Time

<table>
<thead>
<tr>
<th>single GPU:</th>
<th>1 ray/pixel</th>
<th>4 rays/pixel</th>
<th>LB lighting</th>
</tr>
</thead>
<tbody>
<tr>
<td>platform</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G80</td>
<td>2.277 s</td>
<td>8.044 s</td>
<td>32.1 s</td>
</tr>
<tr>
<td>G200 EES</td>
<td>1.151 s</td>
<td>-</td>
<td>15.9 s</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>multiple GPUs:</th>
</tr>
</thead>
<tbody>
<tr>
<td>G80 count</td>
</tr>
<tr>
<td>----------------</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td>16</td>
</tr>
</tbody>
</table>
Conclusions

- Ray tracing forest ecosystems in real-time remains a difficult task.
Conclusions

- Ray tracing forest ecosystems in real-time remains a difficult task.
- Global illumination (leaf transparency and inter-object light scattering) can be approximated by a lattice-Boltzmann model, executed as a pre-processing step, followed by interpolation.
Conclusions

- Ray tracing forest ecosystems in real-time remains a difficult task.

- Global illumination (leaf transparency and inter-object light scattering) can be approximated by a lattice-Boltzmann model, executed as a pre-processing step, followed by interpolation.

- Mapping both LB model and ray tracing engine to CUDA delivers reasonable performance; 16 G80s delivered 6 fps at resolution $896 \times 448$ on a scene with 273M triangles.

Safe conjecture: 24 G200s (full clock) would provide real-time.
Conclusions

- Ray tracing forest ecosystems in real-time remains a difficult task.
- Global illumination (leaf transparency and inter-object light scattering) can be approximated by a lattice-Boltzmann model, executed as a pre-processing step, followed by interpolation.
- Mapping both LB model and ray tracing engine to CUDA delivers reasonable performance; 16 G80s delivered 6 fps at resolution 896×448 on a scene with 273M triangles.
- Safe conjecture: 24 G200s (full clock) would provide real-time.
Conclusions

- Drawbacks (directions for future work):
  - LB execution is not real-time. Reducing the lattice to $64^3$ would make it sub-second, and it is easily distributed.
  - Device memory must hold models of all species. Hundreds of species could not be supported.
  - Adaptive transparency control (as yet) interferes with quality.
  - Ray tracing engine performance has room for improvement. Exploiting additional parallelism and using better acceleration structures will probably improve results.
Conclusions

■ Drawbacks (directions for future work):
  ■ LB execution is not real-time. Reducing the lattice to $64^3$ would make it sub-second, and it is easily distributed.
  ■ Device memory must hold models of all species. Hundreds of species could not be supported.
  ■ Adaptive transparency control (as yet) interferes with quality.
  ■ Ray tracing engine performance has room for improvement. Exploiting additional parallelism and using better acceleration structures will probably improve results.
Thanks!

- NSF - CISE 0722313
- NVIDIA Corporation - graduate Fellowship
- NVIDIA Corporation - G200 EES