A Lighting Model for Fast Rendering of Forest Ecosystems

Robert Geist, Jay Steele

Clemson University
Analytics

- Consider a sheaf of germs of holomorphic functions, \( \gamma \), for which we have the following:
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
- complete CUDA-based implementation
- distribution across multiple GPUs

Maintain (sort of, almost, near) real-time performance.
Background

Approach draws principally from:

Overview

Components:

- hierarchical kd trees, with geometric instancing of Xfrog plant models
- CUDA-based ray tracing engine incorporating the short-stack kd traversal algorithm
- global illumination effects via modification of local ambient component
- global lighting model to generate these effects
Lighting Model

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

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

- \(L\) 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
- comparable in speed, stability, accuracy, and 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
- Comparable in speed, stability, accuracy, and storage
- Simulate transport by tracing evolution of particle distributions through synchronous updates on discrete grid

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
- comparable in speed, stability, accuracy, and storage
- simulate transport by tracing evolution of particle distributions through synchronous updates on discrete grid

- 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
- comparable in speed, stability, accuracy, and storage
- simulate transport by tracing evolution of particle distributions through synchronous updates on discrete grid
- advantages:
  - ease of implementation
  - ease of parallelization
  - ease of handling complex boundary conditions
- disadvantage: derivation (proof) can be “tedious”
Lattice-Boltzmann 3D Lighting

- use 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 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.
Lattice-Boltzmann 3D Lighting

- use 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!
In general,

- conserve mass, \( \sum_m (\mathbf{\Omega}_m \cdot f) = 0 \)
- conserve momentum, \( \sum_m (\mathbf{\Omega}_m \cdot f) \mathbf{v}_m = 0 \), where \( \mathbf{v}_m = (\lambda/\tau) \mathbf{c}_m \)
- \( \mathbf{\Omega}_{i,j} \) controls scattering from direction \( \mathbf{c}_j \) into direction \( \mathbf{c}_i \)
- directional density \( f_0 \) holds the absorption/emission
Lighting Model (isotropic case)

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

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

\[ \Omega_{ij} = \begin{cases} 
1/24 & j = 0 \\
\sigma_s/24 & j > 0, \ j \neq i \\
-\sigma_t + \sigma_s/24, & j = i 
\end{cases} \]

i = 1, ..., 6:

i = 7, ..., 18:
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

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

$$\frac{\partial \rho}{\partial t} = D \nabla^2_{\vec{r}} \rho$$

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
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)
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


To 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?

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... 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 $X_s = 1$.
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
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:

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

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:
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: 
  \[ f s^X + bs^X = \sigma_S^X \]

- normalize:
  \[ g^X = \frac{f s^X - bs^X}{f s^X + bs^X} \quad \text{for} \quad X = R, G, B \]
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: 
  \[ fs^X + bs^X = \sigma^X_S \]

- normalize:
  \[ g^X = \frac{fs^X - bs^X}{fs^X + bs^X} \quad \text{for } X = R, G, B \]

- 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: 
  \[ fs^X + bs^X = \sigma^X_S \]

- normalize:
  \[ g^X = \frac{fs^X - bs^X}{fs^X + bs^X} \]
  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 \(X\)

A Lighting Model for Fast Rendering of Forest Ecosystems – p.17/29
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)
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
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
- modulate LB value with texture and add to standard, local illumination
CUDA basics ...

- code organized around *kernels*, 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*, 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 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
- 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 to match
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>total scene</td>
<td>962</td>
<td>273,376,528</td>
</tr>
</tbody>
</table>

A Lighting Model for Fast Rendering of Forest Ecosystems – p.25/29
### 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>

**single GPU:**

**multiple GPUs:**

<table>
<thead>
<tr>
<th>count</th>
<th>execution time (1 ray/pixel)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1.162 s</td>
</tr>
<tr>
<td>4</td>
<td>0.666 s</td>
</tr>
<tr>
<td>8</td>
<td>0.351 s</td>
</tr>
<tr>
<td>16</td>
<td>0.170 s</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>

### single GPU:

<table>
<thead>
<tr>
<th>G80 count</th>
<th>execution time (1 ray/pixel)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1.162 s</td>
</tr>
<tr>
<td>4</td>
<td>0.666 s</td>
</tr>
<tr>
<td>8</td>
<td>0.351 s</td>
</tr>
<tr>
<td>16</td>
<td>0.170 s</td>
</tr>
</tbody>
</table>
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 relatively fast pre-processing step.
- Mapping the ray tracing engine to CUDA is promising: 16 G80s delivered 6 fps at resolution $896 \times 448$ on a scene with 273M triangles.
- 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. Quality?
  - 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 (single ray vs multiple triangles) at kd leaves is an interesting possibility.
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. Quality?
  - 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 (single ray $\rightarrow$ multiple triangles) at kd leaves is an interesting possibility.
Thanks!

- CISE Directorate of the US NSF - award EIA-0305318
- NVIDIA Corporation - graduate Fellowship
- NVIDIA Corporation - G200 EES