GPU-based Monte Carlo Ray Tracing Simulation For Solar Power Plants

GTC 2012

Michel Izygon, Ph.D., Claus Nilsson, Ngoc Vu, Peter Armstrong
Tietronix Software
www.tietronix.com
Headquartered in Houston, TX

Main Activities:

- **Software Development for NASA JSC**
  - Onboard and Mission Control Center
  - Simulation systems

- **Medical Device Software**

- **3D Graphics for Simulation, Training, Education**

- **Solar Energy Engineering Software and Services**
  - TieSOL suite of software
Concentrated Solar Power systems use mirrors to concentrate sunlight onto a small receiver. The concentrated light heats a fluid (molten salt or water), which drives a steam turbine connected to an electrical power generator.

Multiple projects are being developed in the southwest (Nevada, California, Arizona), as well as in the rest of the world (Spain, India, China, South Africa, Australia, Morocco, Saudi Arabia, Egypt,…)

Major players:
- **Manufacturing/Suppliers:** Schott, St Gobain, Flabeg, ABB, Alstom,
- **Technology Providers:** BrightSource Energy, SolarReserve, Abengoa, Sener, Areva, Siemens, eSolar, Acciona, …
- **Engineering Construction:** Bechtel, ACS Cobra,…
- **Utilities:** Pacific Gas and Electric, South California Edison, Nevada Power, SDG&E,
- **Funding:** NRG, Google, DoE
All Shapes and Forms
How does it work?

HOW THE SOLAR TOWER WORKS:

1. The solar tower is 115m (377ft) tall and surrounded by 600 steel reflectors (heliostats). They track the sun and direct its rays to a heat exchanger (receiver) at the top of the tower.

2. The receiver converts concentrated solar energy from the heliostats into steam.

3. Steam is stored in tanks and used to drive turbines that will produce enough electricity for up to 6,000 homes.
Generating Electricity
Heliostat
Next Generation Solar Thermal

Bright Source Concept

Abengoa Plants
Software for Central Receiver System
- First generation: Mirval, Helios, Rcell, Delsol …
- Second Generation: Soltrace, WinDelsol, Hflcal, Nspoc, ..

Always constrained by processing power
- i.e., use Hermite Coefficients, Convolution, …
- Ray Tracing usually considered too slow to be used for optimization or for very large systems

Current / future Central Receiver Systems
- Much larger systems (many thousands mirrors)
- Need better tools for design, optimization, operations

=> Needs for next generation software that uses State-of-the-Art Computing Model.
- Provide accurate shading and blocking percentages
- Calculate the incident flux on the receiver
- Calculate efficiencies of all optical losses
- Compute fast enough to enable large range of design parameters assessment/trade-offs
- Model all potential optical effects such as sun shape, facets focusing, facets canting, mirror tracking errors, mirror surface errors, pedestal alignment errors, atmospheric attenuation, ....
Solar Ray Tracing Engine

What it is:

- Customized Ray Tracer
- Highly specialized for CSP
- Double precision
- Fast
1. Generate sun ray
2. Trace ray from sun to mirror
3. Reflect ray
4. Trace ray from mirror to receiver
5. Calculate the flux map
Shading and Blocking

Definitions:

- **Shading occurs at low sun angles** when a heliostat casts its shadow on a heliostat located behind it. Therefore, not all the incident solar flux is reaching the reflector.

- **Blocking occurs** when a heliostat in front of another heliostat blocks the reflected flux on its way to the receiver.
Requirement: “Provide accurate shading and blocking percentages”

Solution: Target rays uniformly across all mirrors.
Requirement: “Calculate the incident flux on the receiver”

Solution:

- Each ray has a Power value
- Losses are subtracted from the Power value as the ray travels through the system.
Requirement: “Calculate efficiencies”

Solution: Each ray has a

- *Power value*
- *State, e.g.*
  - Is alive or dead
  - Has been reflected
  - Has hit receiver
Specialization

- Specialization = speed
- Non-recursive
- Very simple objects
  - Limited planes
  - Limited cylinders
- No acceleration structure
  - Intersection objects are found prior to ray tracing
Limited Planes

- Used for mirrors and some receiver types
- Consist of:
  - 3D normal
  - Width and height
  - Center position
- Smaller memory footprint
  - 64 bytes vs. 192 bytes for triangles (d.p.)
Limited Cylinders

- Used for cylindrical receivers
- Consists of:
  - 3D center
  - Height
  - Radius
- Always aligned with vertical axis
Each heliostat has two lists:
- *Potential shading object*
- *Potential blocking object*

- Fixed number of neighbors
- Neighbor search partially reusable
Pseudorandom numbers need

- Uniform distribution
- Normal (or Gaussian) distribution

Uniform distribution RNGs

- Rand48 from Molecular Dynamics simulation framework (MDGPU)
- CURAND library from NVIDIA CUDA Toolkit

Normal distribution RNG

- Box-Muller transformation
### Random Number Generator

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Rand48</th>
<th>CURAND</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear Congruential Generator (LCG)</td>
<td>$2^{48}$</td>
<td>XORWOW (xor-shift family)</td>
</tr>
<tr>
<td>Period</td>
<td>$2^{192}$</td>
<td></td>
</tr>
<tr>
<td>Performance (ms/100 million double-precision numbers)</td>
<td>200</td>
<td>30</td>
</tr>
<tr>
<td>Inline generation support</td>
<td>NO</td>
<td>YES</td>
</tr>
</tbody>
</table>

(*) Generated on GTX 570
Random Number Generator

- **Block generation**
  - Cost device memory
  - Global state can be deleted after block generation

- **Inline generation**
  - Increase registers used in kernel
  - Global state remains in memory for multiple kernels

TieSOL uses block generation for uniform distribution random numbers and uses inline generation for normal distribution random numbers
## BLOCK GENERATION

<table>
<thead>
<tr>
<th>Method</th>
<th>GPU time (ms)</th>
<th>%GPU time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accumulate_Rays_to_Heliostats</td>
<td>203.146</td>
<td>62.93</td>
</tr>
<tr>
<td>Trace_Rays</td>
<td>92.609</td>
<td>19.53</td>
</tr>
<tr>
<td>Generate_Rays</td>
<td>39.759</td>
<td>8.36</td>
</tr>
<tr>
<td>Accumulate_Rays_to_FluxNodes</td>
<td>15.099</td>
<td>3.03</td>
</tr>
<tr>
<td>generate_seed_pseudo</td>
<td>12.959</td>
<td>2.71</td>
</tr>
<tr>
<td>gen_sequenced</td>
<td>2.175</td>
<td>0.45</td>
</tr>
<tr>
<td>Orient_Field</td>
<td>0.108</td>
<td>0.02</td>
</tr>
<tr>
<td>Initialize_Field</td>
<td>0.023</td>
<td>0</td>
</tr>
<tr>
<td>Find_Shaders_and_Blockers</td>
<td>0.023</td>
<td>0</td>
</tr>
<tr>
<td>Initialize_Facets</td>
<td>0.019</td>
<td>0</td>
</tr>
<tr>
<td>Total</td>
<td><strong>365.921</strong></td>
<td></td>
</tr>
<tr>
<td>Performance boost</td>
<td><strong>33%</strong></td>
<td></td>
</tr>
</tbody>
</table>

## INLINE GENERATION

<table>
<thead>
<tr>
<th>Method</th>
<th>GPU time (ms)</th>
<th>%GPU time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accumulate_Rays_to_Heliostats</td>
<td>203.032</td>
<td>35.94</td>
</tr>
<tr>
<td>Trace_Rays</td>
<td>142.243</td>
<td>25.18</td>
</tr>
<tr>
<td>Generate_Rays</td>
<td>131.434</td>
<td>23.27</td>
</tr>
<tr>
<td>Accumulate_Rays_to_FluxNodes</td>
<td>65.994</td>
<td>11.68</td>
</tr>
<tr>
<td>Init_RNG</td>
<td>0.637</td>
<td>0.11</td>
</tr>
<tr>
<td>Orient_Field</td>
<td>0.107</td>
<td>0.01</td>
</tr>
<tr>
<td>Initialize_Field</td>
<td>0.023</td>
<td>0</td>
</tr>
<tr>
<td>Find_Shaders_and_Blockers</td>
<td>0.023</td>
<td>0</td>
</tr>
<tr>
<td>Initialize_Facets</td>
<td>0.020</td>
<td>0</td>
</tr>
<tr>
<td>Total</td>
<td><strong>543.5138</strong></td>
<td></td>
</tr>
</tbody>
</table>

(*) 10 million rays simulated and 50 million random numbers generated on GTX 570
Two distinct regions exist when examining a single sun shape profile: the distribution within the solar disc, and that of the circumsolar region.

The solar disc profile is created from the processes of solar limb darkening, atmospheric extinction and atmospheric scattering.

The circumsolar region is simply created from the small angle scattering of the incoming beam off large particles in the troposphere.
Sun shape model: function of sun brightness on sun disc versus sun angle

TieSOL:
- More rays in an area of sun disc means more brightness
- Use cumulative distribution function to generate rays on sun disc
Effect of Sunshape on flux

Point sun

Kuiper

V.B. 2
Compute error for normal vector at each impact point

- The error is first broken into 2 error components (X, Y)
- Apply normal distribution on each component with given standard deviations
Mirror Surface Irregularity

X: 1 Y:1 mRads
Surface Irregularity Images

X: 0 Y:0 mRads  X: 1 Y:2 mRads  X: 1 Y:5 mRads
Requirement: “Calculate efficiencies of all optical losses”
Performance Results

Results for:
- TieSOL::CudaFlux
- TieSOL::AnnualPerformance

Comparison between ASAP ray-tracing simulation and TieSOL::CudaFluxLite
- 1 Million Rays
  - ASAP: 25 s
  - CudaFluxLite: 0.2 s

<table>
<thead>
<tr>
<th># rays</th>
<th># runs</th>
<th>GTX 480</th>
<th>GTX 570</th>
<th>GTX 580</th>
<th>GTX 590</th>
<th>3 GTX 570</th>
</tr>
</thead>
<tbody>
<tr>
<td>100 million</td>
<td>1</td>
<td>1.986</td>
<td>1.849</td>
<td>1.547</td>
<td>1.132</td>
<td>0.887</td>
</tr>
<tr>
<td>8 Million</td>
<td>365</td>
<td>17:42</td>
<td>14:52</td>
<td>12:20</td>
<td>8:50</td>
<td>5:25</td>
</tr>
</tbody>
</table>
Next Generation Code: TieSOL

- Suite of code that leverages GPU technology
  - *TieSOL::CudaFlux*
    - Ray Tracing software with extremely fast implementation on Nvidia CUDA
  - *TieSOL::CudaFluxLite*
    - Heliostat Analysis
    - Use Ray Tracing engine following Heliostat Flux map over day
  - *TieSOL::Annual Performance*
    - Use Ray Tracing engine for multiple days per year
  - *TieSOL::CudaSAB*
    - Geometrical computation of S&B using combination of GPU and Multi-Core CPU capabilities
  - *TieSOL::SkyFlux*
    - Computes the flux reflected by the mirrors in the sky, while the heliostats are in a standby mode.
  - *TieSOL:: Solar 4D*
    - Graphical Simulation of Heliostat Field using GPU
CUDA based Monte Carlo Ray Tracing code

- Extremely fast implementation: 50 - 80 Million rays < 1 s
- Accounts for all the optical losses (Solar shape, S&B, Cos, Reflectivity, Pedestal Tilt, Random Errors in Heliostat tracking, canting, focusing, and on mirror surface, atmospheric attenuation, Spillage, Aiming Distribution)
- Graphical User Interface provides ways to control the computation and get the results for whole field or section of it or for each heliostat
- Heliostat geometry and configuration defined through GUI
- Computes the flux map on cylindrical or flat plane receiver
TieSOL:: Annual Performance

- Uses same Monte Carlo Ray Tracing engine as CUDAFlux
  - Computes Annual Performance through integration of results from number of date/time
  - User controls number of computations and get the results for Yearly Energy and for individual Date/Time
TieSOL::CudaFluxLite

- Uses same Monte Carlo Ray Tracing engine as CUDAFlux
  - Compute optical losses and flux map for single heliostat
  - Continuous flux computation over time
  - Allow user to compare heliostat flux for different conditions
Conclusion

- TieSOL suite of software provides toolset for CSP designers, analyst, operators
  - *Extreme performance*
  - *Expand the range of trade offs studies*

- Computing on GPU for CSP is a disruptive innovation

- Extending to other types of CSP systems