GPUFlux – a new radiation model using the GPU

Gerhard Buck-Sorlin

UMR1345 Institut de Recherche en Horticulture et Semences (IRHS)
Équipe Arboriculture Fruitière
AGROCAMPUS OUEST Centre d’Angers - INRA - Université d’Angers.

Introduction

So … why yet another light model?

- CPU: Processing a MC model is *THE* bottleneck in any FSPM

Introduction

Dietger van Antwerpen

* Technical University Delft, the Netherlands
* Internship at WUR, 1.3.11 – 31.8.2011
* Project « Biosolar Cells »

Introduction

- Monte-Carlo light tracer as part of GroIMP
- Spectral light transport simulation
- Conversion of absorbed light spectrum into products of photosynthesis at individual leaf level
- High performance
- Platform independence

Methods

- light tracer utilizes available computing resources through *OpenCL*
- OpenCL (Open Computing Language):
  - first open, royalty-free standard for general-purpose parallel programming of heterogeneous systems.
  - provides uniform programming environment for software developers to write efficient, portable code…
  - …using a diverse mix of multi-core CPUs and other parallel processors.
- During simulation: each object keeps track of the amount of light it absorbs
- Computation of:
  - a fully discretized absorption spectrum or
  - several integrated weighted spectra

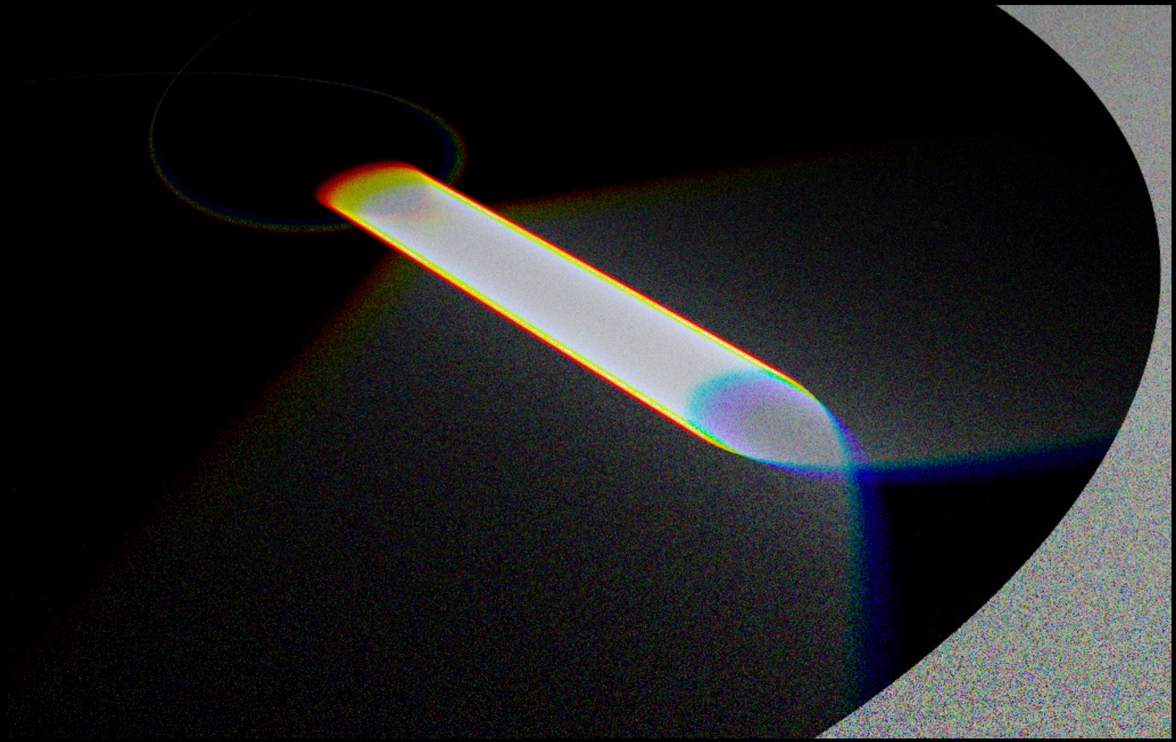


Figure 1. GroIMP supports full spectral rendering.

(effects due to subsurface scatting and participating media are ignored)

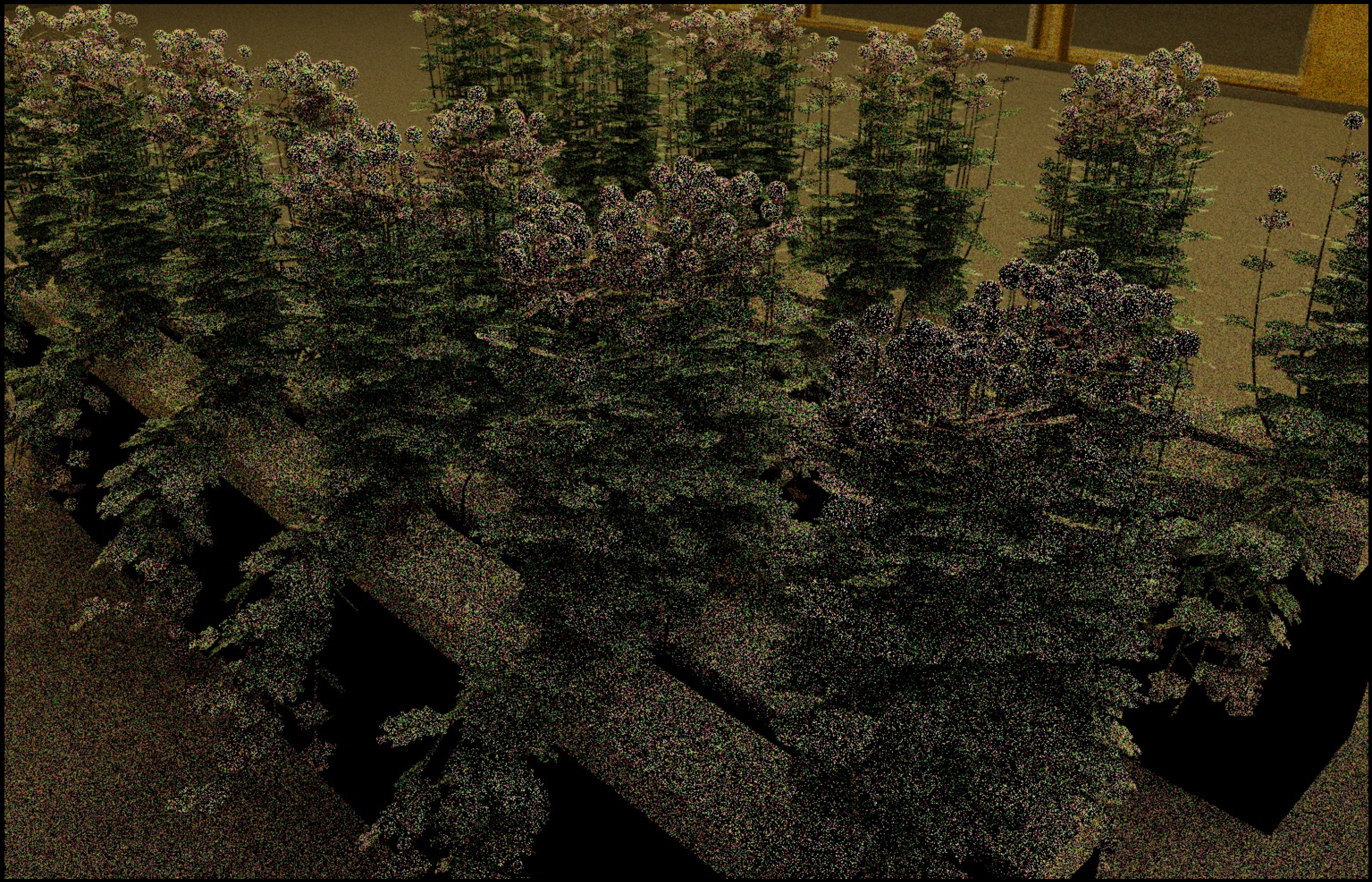
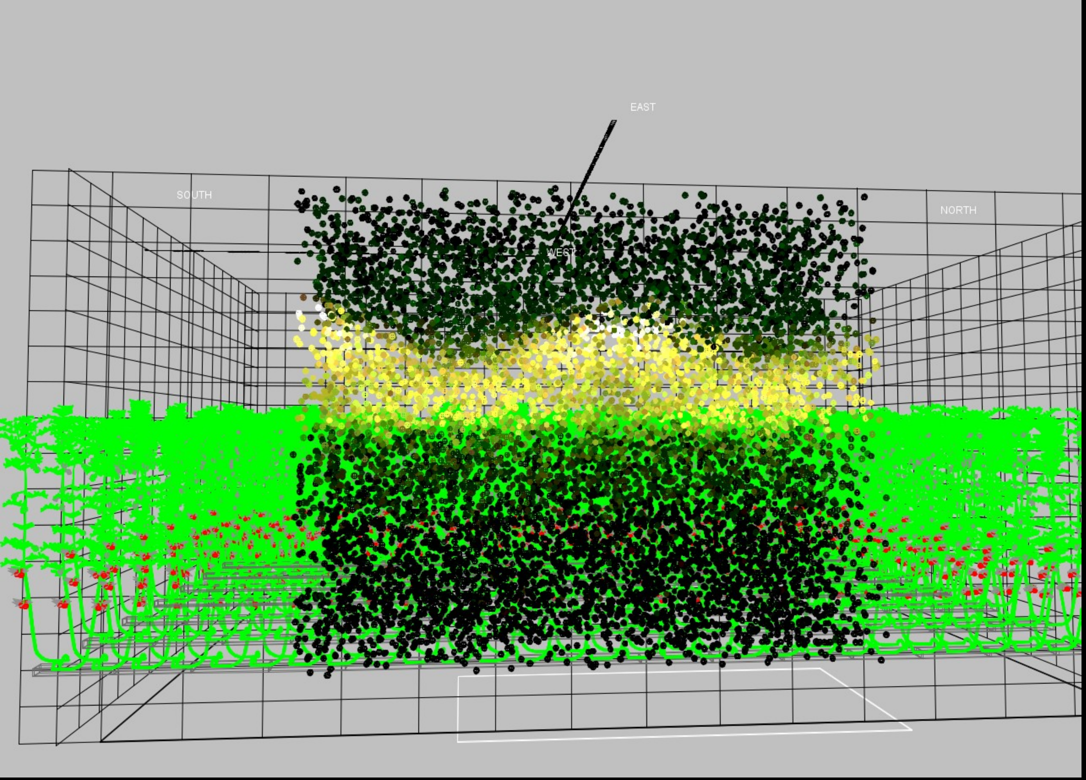
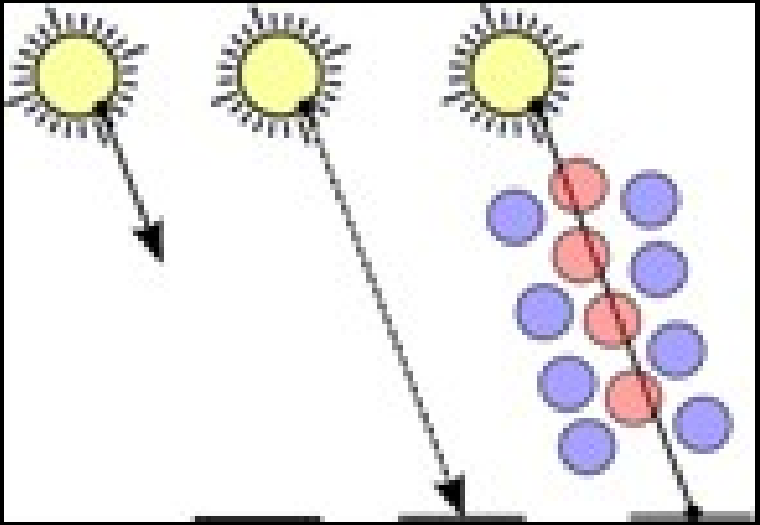


Figure 2. Splatted visualization of the light tracer after a few seconds on an NVIDIA GeForce GTX 480. The rendered image shows a cut-rose production system with upright and bent shoots, with a total of 48696 objects: leaves, internodes, flowers, plus inanimate objects (slabs, benches).



Dense sensor clouds lead to large variations in path depths. This significantly reduces SIMD* efficiency on the GPU.

To improve performance, sensors and geometry are handled using separate acceleration structures: Each ray is first intersected with the geometry, after which the corresponding ray segment is traced against the sensors.

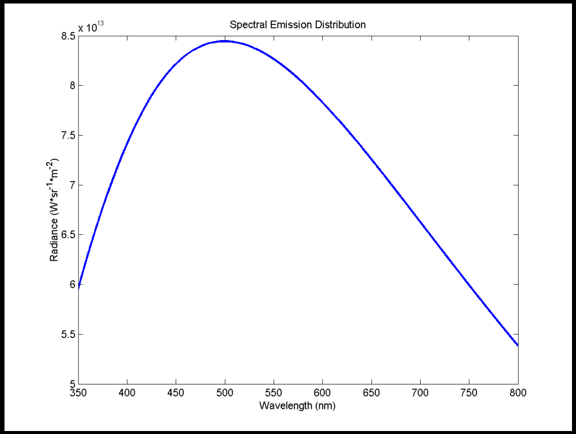


*Single Instruction stream Multiple Data stream. The instruction execution architecture of a vector processor (a CPU or GPU that performs one operation on multiple sets of data simultaneously).

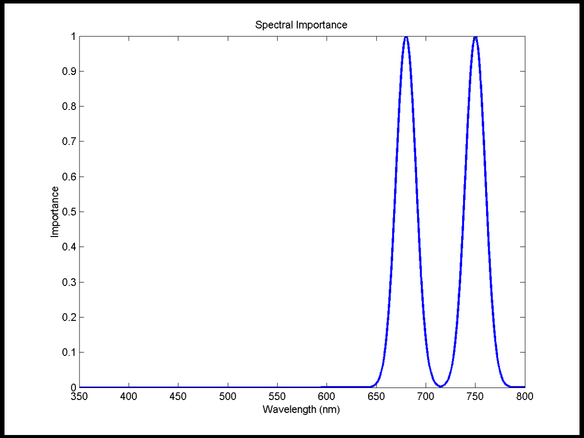
Spectral Importance Sampling

To focus computing power, spectral wavelengths are sampled proportional to a user specified spectral importance function and the spectral emission distribution of each light source:

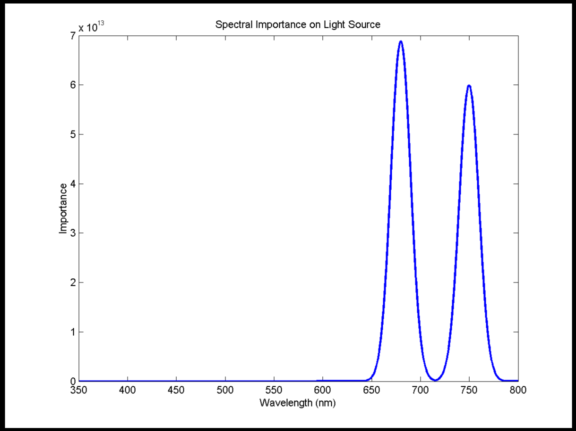
Spectral emission distribution:



Spectral importance:



Spectral importance combined with light source:



Platform Independence
The combination of Java and OpenCL results in near-platform independence with high performance on heterogeneous systems.

Some disadvantages:

- Little room for platform specific low level optimizations
- Platform specific bugs
- OpenCL + Java complicates debugging
- OpenCL kernel compilation is slow for large kernels

- only useful when all materials and light sources are defined over the entire simulated spectrum (issue of data availability)

```
FluxLightModel lm = new FluxLightModel(200000,10);

lm.setMeasureMode(MeasureMode.RGB);
lm.setMeasureMode(MeasureMode.FULL_SPECTRUM);
lm.setMeasureMode(MeasureMode.INTEGRATED_SPECTRUM);

lm.compute(true, true); // compute light rebuilding ALL scene objects
lm.compute(true, false); //compute light rebuilding only light sources
```

1) define lamp module:

```
module SONTlamp (float power) extends LightNode()
{
{setLight(new SpectralLight(getSpectralCurve()).(
      setPower(power),
     setLight(new PhysicalLight().(setDistribution(ldi)))
      )  // end SpectralLight
      ); // end setLight
      }
}  // end lamp

protected static SpectralCurve getSpectralCurve()
{
IrregularSpectralCurve spdr = new
    IrregularSpectralCurve(wb,pd);
return spdr;
}

static float[] wb = 380,385,390,395,…,765,770,775,780};
static float[] pd = {0.000967721,0.000980455,…,
0.001973642,0.001986376};
static LightDistribution ldi = new LightDistribution(breed);
```

(breed: double array with measured light distribution values per solid angle)

2) insert lamp into scene: `SONTlamp(pow(time, mar1))`