BornAgain - Refactoring #1733

Avoid multiple calculations of same RT coefficients

12 Dec 2016 09:02 - wuttke

Status:	Resolved	**Start date:**	12 Dec 2016
Priority:	High	**Due date:**	
Assignee:	herck	**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:	Sprint 33		

Description

Currently, SpecularMatrix::calculateUpFromLayer is called within loops

... - // scattering from particles
... - for (auto& layer_comp: m_layer_computation)
... - - for (const DecoratedLayerComputation* comp: layer_comp) // loop over layouts
... - - - DecoratedLayerComputation::eval(const SimulationOptions&, ProgressHandler*, bool polarized, [SimulationElement iterators])
... - - - - for (std::vector<SimulationElement>::iterator it = begin_it; it != end_it; ++it)
... - - - - - IInterferenceFunctionStrategy::evaluate(const SimulationElement& sim_element)
... - - - - - - IInterferenceFunctionStrategy1::precomputeParticleFormfactors(const SimulationElement&)
... - - - - - - - for (auto ffw: m_formfactor_wrappers)
... - - - - - - - - FormFactorWrapper::evaluate(const SimulationElement&)
... - - - - - - - - - ScalarSpecularInfoMap::getOutCoefficients(const SimulationElement&)
... - - - - - - - - - - ScalarSpecularInfoMap::getCoefficients(kvector_t kvec)
... - - - - - - - - - - - SpecularMatrix::execute(const MultiLayer&, const kvector_t k, MultiLayerCoeff_t&)
... - - - - - - - - - - - - calculateUpFromLayer(SpecularMatrix::MultiLayerCoeff_t&, sample, kmag, layer_index) <--HERE
... - // scattering from roughness
... - RoughMultiLayerComputation::eval(std::vector<SimulationElement>::iterator begin_it, .. end_it)
... - - for (std::vector<SimulationElement>::iterator it = begin_it; it != end_it; ++it)
... - - - RoughMultiLayerComputation::evaluate(const SimulationElement& sim_element)
... - - - - for (size_t i=0; i<mp_multi_layer->getNumberOfLayers()-1; i++)
... - - - - - RoughMultiLayerComputation::get_sum8terms(size_t ilayer, const SimulationElement& sim_element)
... - - - - - - LayerSpecularInfo::getOutCoefficients(sim_element)
... - - - - - - - ScalarSpecularInfoMap::getOutCoefficients(const SimulationElement&)
... - - - - - - - - ScalarSpecularInfoMap::getCoefficients(kvector_t kvec)
... - - - - - - - - - SpecularMatrix::execute(const MultiLayer&, const kvector_t k, MultiLayerCoeff_t&)
... - - - - - - - - - - calculateUpFromLayer(SpecularMatrix::MultiLayerCoeff_t&, sample, kmag, layer_index) <-- HERE

(for full call stack see http://apps.jcns.fz-juelich.de/redmine/projects/bornagain/wiki/GISASSimulation_call_stack).

I see no reason for embedding the RT computation in loops over layers, layouts, formfactors. Precompute the RT coefficients, and store them as members of SimulationElement.

History

#1 - 12 Dec 2016 09:02 - wuttke

- Description updated

#2 - 02 Feb 2017 14:58 - herck

The current interface for the Fresnel maps (ScalarSpecularInfoMap above, but more generally FullFresnelMap now) is agnostic about how to retrieve these coefficients. A solution that requires no moving around of any of the objects would be to provide an implementation that caches previously calculated values in a hash table, at the cost of memory usage. The biggest gain is to be expected for the scalar case, where only the outgoing inclination angle determines the R and T coefficient, thus reducing the number of computations of Fresnel coefficients drastically.

#3 - 02 Feb 2017 15:14 - herck

One extra remark concerning the caching of Fresnel coefficients: the Fresnel coefficients of all layers/slices have to be cached as one single structure. Right now, ScalarSpecularInfoMap is a map for a single layer only.

#4 - 03 Feb 2017 09:35 - herck

- Subject changed from Precompute RT coefficients to Avoid multiple calculations of same RT coefficients

- Assignee set to herck

- Target version set to Sprint 33

#5 - 09 Feb 2017 16:47 - herck

- Status changed from New to Resolved

#6 - 18 Sep 2020 17:32 - wuttke

- Parent task deleted (#1119)