PBRT: Introduction

Tuesday, November 10, 2020 11:37 PM

1 Introduction

- A rendering system's basic abstractions and interfaces.
- "We believe that deep understanding of a small number of algorithms in this manner provides a stronger base for further study of computer graphics than does superficial understanding of many." - The authors.
- Ray tracing algorithms follow the path of ***infinitesimal*** rays of light through the scene until they intersect a surface.
- Practical issues such as antialiasing, robustness, numerical precision, and the ability to render efficiently complex scenes: it's important to consider these from the start of the system's design, since these features can have subtle implications for all components of the system and can be quite difficult to retrofit in to the system at a later stage of implementation.
- The system is small enough for one person to understand completely.
- Pbrt can compute images that are physically correct; they accurately reflect the lighting. If different lighting algorithms give different results for the same scene, there is certainly an error in one or some of them.
- Numerical stability must be handled carefully, and algorithms that ***don't waste floating-point precision*** are critical.
- Making the system both flexible and robust is a difficult task.
- The rewards for developing a system that addresses all these issues are enormous. It is a great pleasure to write a new renderer or add a new feature to an existing renderer and use it to create an image that couldn't be generated before.

1.3 System overview

There's only a single list of Lights in the Scene. Renderers that aren't physically based usually associate lights with objects, so that those lights only illuminate those objects, but lights in the real world affect all the objects in the scene that are visible to them and within reach.

- [ ] TODO: Rest of section; only covered Scene.

1.3.3 Integrator interface and SamplerIntegrator

Integrators are the ones that render the scene. The abstract class Integrator is the interface and includes a Render(Scene) method.

Some implementations of the Integrator interface are SamplerIntegrator, WhittedIntegrator, and DirectLightingIntegrator, which is a subclass of SamplerIntegrator.

SamplerIntegrator has a rendering process driven by a stream of samples (points on the image at which to compute radiance) from a Sampler.

13.4 The main rendering loop

With SamplerIntegrator: the Sampler generates a CameraSample (film plane pixel position in raster coordinates, time, lens sample position, etc.), which the Camera uses to generate a Ray; SamplerIntegrator then invokes Li() to compute the amount of light arriving at the film plane position along the ray. This is just 1 contribution, which Film stores and accumulates.

PBRT can render an image in parallel by breaking it up in tiles. For simplicity, it always uses 16x16 tiles.