Acknowledgement/Background

Ray Tracing: The Rest of Your Life

Peter Shirley
edited by Steve Hollasch and Trevor David Black

Version 3.2.3, 2020-12-07

Copyright 2018-2020 Peter Shirley. All rights reserved.

Contents

1 Overview

2 A Simple Monte Carlo Program
   2.1 Estimating Pi
   2.2 Showing Convergence
   2.3 Stratified Samples (Jittering)
Acknowledgment

This lecture originally prepared and delivered by Ben Sattelberg for Fall 2019
Today’s Transition
What Color for Cyan Circle
Direct Vs. Indirect Lighting

Step Through Indirect: Step 0
Step Through Indirect: Step 1
Step Through Indirect: Rest
The Rendering Equation

\[ L_o(x, \omega_o, \lambda, t) = L_e(x, \omega_o, \lambda, t) + \int_{\Omega} f_r(x, w_i, w_o, \lambda, t) L_i(x, w_i, \lambda, t) (\omega_i \cdot n) \, d\omega_i \]
The Rendering Equation

\[ L_0(x, \omega_0, \lambda, t) = L_e(x, \omega_0, \lambda, t) + \int_{\Omega} f_r(x, w_i, \omega_0, \lambda, t) L_i(x, w_i, \lambda, t)(\omega_i \cdot n) d\omega_i \]

Light coming out = Light emitted this direction +
The amount of incoming light to this point that is reflected this direction
Monte Carlo Methods

• Exact solutions are difficult (impossible?)

• Compute time is cheap

• Use many different samples to approximate true solution

• Shoot many rays per pixel and let them bounce “randomly”
SageMath Example
Nice Animations 1

https://www.youtube.com/watch?v=frLwRLS_ZR0
Nice Animations 2

https://www.youtube.com/watch?v=frLwRLS_ZR0
Monte Carlo Illumination

Don’t calculate illumination from each light at each point. Instead:

1. Shoot many rays per pixel
2. Have each pixel bounce according to material properties
3. Keep track of the running albedo
4. When the ray collides with a light, return
Bouncing Rays

- Specular Reflection Stays the Same
- Lambertian – What to do?
- Recall from our Illumination Lecture

Light per unit area arriving depends upon angle to light source.
Start Simple – Uniform Directions

• Want to sample uniformly from points in the sphere

• Complex to do directly without bias

• Instead, sample from a cube and remove samples outside the sphere
Better: Lambert’s Cosine Law

Modified From: https://upload.wikimedia.org/wikipedia/commons/2/25/ Lambert_Cosine_Law_1.svg
A Few More Details

• Ignore Ka, Kd, Ks, and Kr terms — only use the albedo and emittance of the object

• Have materials determine the direction of a bounced ray

• Lights must have volume to be collided with

• Lights may need emittance > 1
Sample MC Ray Tracer

Monte Carlo Ray Tracing
Ben Sattelberg, November 19, 2020

In this notebook is an implementation of a recursive Monte Carlo ray tracer.

In [1]:
import numpy as np
import multiprocessing
import time
import itertools

from IPython import display

import matplotlib.pyplot as plt

Some helper methods are useful for later calculations. Making vectors unit length is done very fast, uniformly from the unit sphere makes the Lambertian bouncing easier to work with, and having a Lambertian bounce makes mirror-like objects easier to work with.

WARNING: MANY HOURS TO RUN!
Accuracy Over Time

From https://upload.wikimedia.org/wikipedia/commons/e/ea/Path_tracing_sampling_values.png
A Few Efficiency Thoughts

• Ray culling — ignore rays that have low albedo and boost the rest
• Early stopping — skip pixels that have converged
• Bidirectional path tracing — shoot rays from the camera and lights and connect them
• Explicit light sampling — cast rays directly towards the lights some proportion of the time
• AI denoising — render a partial image and feed it to an algorithm that gives a complete image