CSC4140 Final Projects

Computer Graphics

May 3, 2022

The final is 30% of the total mark.

We encourage to help eachother but do not show the same thing in your report and do not cheat!

Strict Due Date: 11:59PM, May 23^{th}, 2022

Student ID:

Student Name:

This assignment represents my own work in accordance with University regulations.

Signature:

1 Hair Simulator

Problem Description

Based on the code of assignments 6 and 7(or Mitsuba, recommended), statically render realistic hair and (ideally) fur. The reflectance models are basically hard to simulate because hair and fur have complex light-scattering properties that make them difficult to emulate via ray-tracing. Hair/fur also has many physical properties like thickness, length, and bend radius, which makes it hard to render when applying physics(wind, motion). We can use the Marschner model to generate hair and Lingqi's state-of-the-art fur reflectance model as our primary resources.

Goals and Deliverbles

For this project, deliver a series of images generated with the Mitsuba Renderer. You can first use the renderer to generate some basic images of a head of hair without any advanced lighting models. Then after implementing the Marschner model, we will generate several images of much more realistic human hair to compare to the basic renderings.

When working on this project, you will have to figure out how to render realistic renderings of hair. Therefore, you have a couple of stretch goals depending on the difficulty of the hair rendering and what issues we come across. First, you need to use the Linq-Qi model for the fur to try to generate realistic, fun renderings for animals. If this is also successful, you can choose to set up some interactive real-time rendering demo using OpenGL.

For the fur rendering deliverables, try to generate several images of one or two animals.

At this time, it isn't easy to measure the quality of our renderings using any objective methods since it is more about a realistic appearance. Since you are only using one model for each render, you do not need to check for improved performance since you have no baseline. Therefore, the quality of the rendered images will be mostly subjective.

Resources

1. Mitsuba Renderer OpenGL (Optional)
2. egsr hair
3. paper fur2
4. Mitsuba

2 Realize BDPT (better to use cuda)

Problem Description

Based on the code of assignment 6 and 7, realize your own Bi-Directional Path Tracer.

Goals and Deliverbles

Render the given scenes in assignments 6 and 7 using your own BDPT and compare the difference with the current one in your report. We encourage you to use CUDA to implement them so as to avoid the hours-long rendering process. Nowadays, CUDA has become very easy, and it's just a library for parallel computing and rendering.

Resources

M. Clark, "CUDA Pro Tip: Kepler Texture Objects Improve Performance and Flexibility", NVidia Accelerated Computinig, 2013. [Online]

T. Karras, "Thinking Parallel, Part III: Tree Construction on the GPU", NVidia Accelerated Computing, 2012. [Online]

T. Karras, "Thinking Parallel, Part II: Tree Traversal on the GPU", NVidia Accelerated Computing, 2012. [Online]

E. Veach, "Robust Monte Carlo Methods for Light Transport Simulation", Ph.D, Stanford University, 1997.

3 Realize Spectral Ray Tracing

Problem Description

The current implementation of the raytracer cannot model dispersion and chromatic aberrations because its light model is not wavelength-dependent. Currently, indices of refraction are constant rather than different for each wavelength. You can implement your code based on assignment 7.

Goals and Deliverbles

Implement spectral ray tracing by tracing rays of different wavelength sampled using the human eye's wavelength profile for each color (RGB). By modeling different indices of refraction based on those wavelengths for glass-like materials, we hope you to reproduce effects such as the dispersion of light through a prism, the changing colors based on viewing angle for a lens on a reflective surface with a thin film (such as a DVD), as well as model chromatic aberrations present in real

camera systems with lenses. Additionally, you need to create wavelength dependent bsdf's and lighting, we hope to model different temperature lights.

1. Prism scene rendering
2. Disk/bubbles scene rendering (Add different environment maps (potential source from Light Probe Library). Images from the light probe library are in HDR format, suitable for spectral ray-tracing since you have the more realistic spectrum distribution of each scene pixel).
3. Correctly simulates chromatic aberration of different lenses.
4. Compare rendered images with real photos we take of the objects (e.g., disk).
5. Compare rendering under different temperature lights.
6. Finally, we hope you deliver a synthesized image that harmoniously combines objects that best illustrate the effectiveness of our spectral ray tracer. (e.g., gemstones, etc., suggestions on this would be helpful!)
7. Optional: Add fog/volumetric scattering so that rainbows can be seen.

Tasks:

1. Change lenstester to also include wavelength argument that the user can set. (mainly for debugging purposes)
2. Refactor code so that rays have a wavelength argument that can be passed in and checked as well as that functions that return Spectrums now return a single intensity value
3. Change raytrace_pixel to ask for multiple ray samples for each color channel, then combine those color channels
4. Change camera.generate ray to take in a color channel argument and sample that color channel's wavelength distribution (Gaussian) to change the ray's wavelength
5. Change lens_camera's tracing through the lens to use the wavelength argument to change indices of refraction when tracing through the lens
6. Change sample_L of lights to have a wavelength-dependent intensity to simulate different colors of lights (maybe initialize lights with a temperature argument and model them as ideal black bodies to get the intensities for each color)
7. Rewrite BSDFs of colored objects to return a wavelength-dependent magnitude instead of a constant spectrum argument.

8. Rewrite/write glass BSDF to have wavelength-dependent indices of refraction (similar code as lens_camera's tracing)

9. Write a bubble/ thin-film interference BSDF that uses wavelength, thickness, and light to determine if the interference occurs (integer multiples of wavelength)

10. Write new scene/dae files (using Blender)/mess with the parser to create a triangular prism would want a small area of light create a disk + reflective surface + transparent coating

Resources

1. Prisms and Rainbows: a Dispersion Model for Computer Graphics

2. Iridescent Surface Rendering with Per-channel Rotation of Anisotropic Microfacet Distribution

3. Rendering Iridescent Colors of Optical Disks

4. Derive spectrum from RGB triple

5. soap bubbles 1

6. soap bubbles 2

   Other useful links: [1] refractive index [2] refractive indices [3] glassner [4] hyperphysics [5] Morris, Nigel. "Capturing the Reflectance Model of Soap Bubbles." University of Toronto (2003).

4 Point Cloud to Mesh

Problem Description

Based on the code of assignment 5, convert point clouds into 3D meshes. Specifically, given an input file of format .plz/.ply, your program will output a 3D triangle mesh in the format .dae.

The purpose of a 3D scanner/Depth camera is to analyze a real-world object and collect data on the object's shape and appearance. The data collected by a 3D scanner is used to construct virtual models for various purposes, including but not limited to industrial design, medical prosthetics, engineering prototyping, quality control, entertainment production, and historic artifact documentation. These devices commonly scan the surface of the object and output data in the form of point clouds. Then the point clouds must be processed to created a 3D virtual mesh. The creation of the mesh is a challenging task. There are many ways to connect the points within the point cloud to create a mesh: which method is the fastes? Which method is the most accurate? How are

the terms “fast” and “accurate” best defined for this task? Just seek to answer these questions in this project, and in doing so we hope you to construct a point cloud processing method that rivals the current standards. The main references for this project are this paper on the ball-pivoting algorithm , and this paper on Poisson surface reconstruction.

Goals and Deliverbles

Your task is to solve the problem of generating a 3D mesh that connects all the points from a given set of points (a point cloud).

To create a mesh reconstruction from a point cloud, you need to implement "The Ball-Pivoting Algorithm for Surface Reconstruction algorithm" by using the HalfEdge data structure. Essentially, this algorithm will give us a function that converts from .xyz to .dae.

Compare the performance graphs between the two algorithms. The performance will involve timing the system on specific 3D scanned files. For a file input, we want the algorithm to have fast mesh creation but at the same time be accurate. A precise metric would be the number of FLOPs required by each algorithm. You can use a timer (for simplicity) between the start of the mesh creation and the end using different algorithms (either the basic implementation or modified implementations of the basic one) to see which ones obtain the fastest runtime. There are some relatively advanced statistical calculations for judging mesh accuracy for accuracy. The lecture on geometry processing gives a good basis for determining the “goodness” of a mesh rendering. If you check every mesh triangle and determine if it holds for a certain threshold, say, 80% of triangles. You have a list of triangles in your mesh, and if you iterate through and find at least 80% pass, we can stop and declare the mesh "good".

Implement the more advanced “Poisson Surface Reconstruction” algorithm to create a mesh reconstruction from a point cloud. After completing the performance analysis algorithm for the ball-pivoting algorithm, you need to answer these questions: Is the "Ball-Pivoting Algorithm" or "Poisson Surface Reconstruction" better for constructing a mesh from a point cloud? Is there a tradeoff between speed and accuracy for the algorithms?

Resources

For test files, you can use the http://graphics.stanford.edu/data/3Dscanrep/. Specifically, use MeshEdit to parse your model files and visually display your meshes to check the reconstruction algorithm’s accuracy. Use the HalfEdge data structure to implement our mesh reconstruction algorithms. The models used to test your mesh reconstruction algorithms are from the Stanford 3D scanning repository. It would help if you also used RPly, a small ANSI C library for parsing .ply files. This library made it possible for us to create the ply2xyz converter necessary within

a short time for creating your point cloud rendering program. By converting to .xyz format (a format that Michael made up for this project), you can view your files in a format that is easier to parse.

5 Realize image processing pipeline for night sight

Problem Description

Taking photographs in low light using a mobile phone is challenging and rarely produces pleasing results. Aside from the physical limits imposed by read noise and photon shot noise, these cameras are typically handheld, have small apertures and sensors, use mass-produced analog electronics that cannot easily be cooled, and are commonly used to photograph subjects that move, like children and pets. Low light videos processing are a hot topic for today's mobile phone camera.

Goals and Deliverbles

Based on the code of hdr plus, improve and implement your own image processing pipeline for night sight images and videos. To real-timely process a video sequence, the idea is to set the raw sequence just like a FIFO (which is different from single image processing taught in the class). You can set the "FIFO" to store 6-13 raw sequences and then process them like the "night sight look" paper. Try your best to get the best visual pleasuring results.

Resources

1. Handheld Mobile Photography in Very Low Light
2. hdr plus paper
3. hdr plus

Final Note

You have achieved a milestone in Computer Graphics. Here your task left is to make some fancy results and report! Computer Graphics is not only a science of producing graphical images with the aid of a computer but also a fancy **art**! Again, always be creative!