Graphics and Imaging Lab

Path tracing

Informática Gráfica

Grado en Ingeniería Informática

Goal

The goal of this practical work is to implement a path tracer [Kaj86, FHF+17][1] from scratch. It can be done either individually or in groups of two students.

1 Previous assignment

The path tracer builds on top of previous assignments, from which it should borrow the following:

- A pinhole camera model.
- The image description and saving routine.
- Spheres and infinite planes as geometrical primitives.
- The emission property (RGB tuple) for each primitive.
- Any optional extensions implemented.

This assignment will then add reflectance properties and provide the rendering algorithm which deals with them properly.

2 Path tracer

The path tracer will enable a full light transport simulation. Each primitive will therefore include reflectance properties that can later be used by the algorithm, additionally to the emission. Specifically, the reflectance properties will be a combination of:

- A Lambertian diffuse BRDF.
- A perfect specular reflectance according to the law of reflection (delta BRDF).
- A perfect refraction according to Snell's law (delta BTDF).

[1] https://jo.dreggn.org/path-tracing-in-production/

Each of those terms will be modulated by a specific RGB coefficient, and, for physical correctness, the addition of all the coefficients will be less than 1. Not all objects will have the three properties activated (an object with all the three properties would be weird) so some of the RGB coefficients must be 0. Typical materials can be:

- A pure diffuse material (no specular reflection nor refraction).
- A plastic material, with diffuse and specular reflection (no refraction). The specular RGB components should be equal (gray-white specular reflection).
- Dielectric materials, with specular reflection and refraction (no diffuse). The coefficients would be better modulated by the Fresnel equations.

It can be assumed that an emitting object does not reflect light and viceversa (a reflecting object does not emit light).

The path tracer will have only one specific parameter (set by command line): the **number of paths per pixel**. The path interactions will follow Monte Carlo sampling with Russian Roulette. Each path will have the standard termination conditions (not hitting anything, hitting an object with no reflectance properties or by a Russian Roulette termination condition). The output image will have also its own command line parameters: width, height, file name and color resolution (for storing the image as an HDR, see the tone mapping assignment). The color resolution parameter can have a default and should be high enough $> 2^24$ for avoiding resolution pitfalls and artifacts when tone mapping.

It is not required that the geometry is loaded from a file. Therefore, the scenes can be hard-coded (defined in source code). One of the scenes (the *default* one) should be a **Cornell Box**: five walls forming a box (with an open side for the camera to watch), all of them diffuse gray except the left one (red) and the right one (green). Also, it should have two spheres on the floor. One of the spheres should have a plastic-like material while the other should have a dielectric-like material. The lighting should be an area light source on the top of the box (could be the whole upper wall).

3 Point lights and next event estimation

Once the standard path tracing is done, point lights will be added to the model. Each point light will have a power (RGB tuple) and a specific position in space, and the scene might have as many light sources as needed. At each interaction of each path, the path tracer will estimate direct lighting from all the point light sources by casting shadow rays (next event estimation). Notice that the number of light sources will affect rendering time unless you perform some kind of Russian Roulette between the light sources.

In order to compare the effect of next event estimation, another test scene will be a **Cornell Box** in which the light source is a point light source in the upper half of the box (instead of an area light source), making a total of two Cornell Boxes to test with different illumination

conditions. There is no requirement of applying next event estimation to area light sources.

If all these required features (path tracer + next event estimation with point lights) are accomplished correctly, the students will get a mark of 7. For higher marks, the students should do any of the optional extensions.

4 Optional extensions

The students can choose one (or more) of the following extensions (including the ones from the previous assignment):

Other geometries. Add other (several) different geometrical primitives defined by their implicit equation, such as cones, cylinders, ellipsoids, disks or triangles. You can get creative here.

Acceleration structures. For scenes with many objects, implement Bounding Volume Hierarchies or Kd-trees for accelerating rendering. The improvement on rendering time should be evaluated.

Parallelization. Parallelize the algorithm between several threads of execution. Use a parallelization strategy at image level: subdivide the image into regions (lines / rectangles / columns / pixels) and use a thread-safe task queue to set up the render regions. Enqueue all the render tasks (single producer) and have multiple threads popping tasks and performing them (multiple consumers). Try different strategies: different image regions and image regions sizes, different thread-safe queue implementations and different combinations between parallelization and acceleration structures. **Additionally :** Explore more sophisticated parallelization strategies, including having differents kinds for consumers for different time-consuming tasks (such as separating intersection calculations from light transport calculations).

Textures. Add textures that modulate the emission of an object or any of the coefficients that model its appearance (Phong BRDF, perfect specular or perfect refraction). Other textures (bump maps or normal maps, for instance) can be added as well but they are harder to model because they need a tangent field.

Spectral rendering. Instead of having RGB emission properties for each object, include a spectral representation of color that is later converted to RGB using the standard CIE RGB curves. The spectral representation can be a vector of 8, 16 or 32 values, or even a spectral function (a function that depends on the wavelength) that is sampled in the corresponding

8, 16 or 32 values.

BSDFs. Add new BSDF models (BRDFs or BTDFs) that deal with other phenomena: Fresnel dielectric interactions (including perfect reflection and refraction), Phong or Ward specular lobes, microfacet models... Anisotropic BSDFs can be added as well, but they need more work because they require a tangent field.

Motion blur. Add animations to the geometry model: simple ones such as translations along a straight line, will suffice. Then add the temporal dimension to the sampling and integrate along a small temporal frame (exposition time).

Depth of field. Expand the pinhole camera model into a camera model that includes an aperture (adding two extra dimensions) and a focal distance that enable depth of field effects (blurring due to distance to the focus point).

Importance sampling next event estimation. Instead of all the point light sources, find a way to importance sample their contribution at each interaction, and maybe include area lights (emitting objects) into the next event estimation.

Bidirectional path tracing. Implement a simple version of Bidirectional path tracing in which paths are generated also from the light source with the corresponding shadow rays.

Participating media. Extend path tracing so it takes into account participating media such as fog, smoke or water, including absorption and scattering events.

Other extensions. Any other extension considered by the students may be added as well, but they must be first discussed with the teacher.

Generally speaking, before undertaking any optional extension, the students should contact the teacher so they are assessed about the optimal way of addressing the specific extension and they extension can be toned down in case it surpasses the expected workload.

5 Implementation details

Programming language. The students can choose any programming language they consider. However, we recommend C++ due to the following:

- The compiled code should be more efficient than other languages.

- The code that will be provided for the second practical work is in C++ (unless you choose to stick with your own code for both of them).
- C++ enables to define operators such as + or * for any new data type. This makes the code more readable for data types such as vectors.

File formats. As you have used it before in previous assignments, we recommend the `.ppm` image format for writing images (in the HDR-like format with large color space resolution, as discussed in previous assigments). Find the details in `http://netpbm.sourceforge.net/doc/ppm.html`. Also it can be used to load images as input if the optional extensions of textures or environment maps are included.

If the students decide to include the optional extension of triangle meshes and want to load them, we recommend the `.ply` file format. It is again a very simple text file. The description of such format, along with some free `.ply` models can be found in `https://people.sc.fsu.edu/~jburkardt/data/ply/ply.html`.

External libraries. The use of any kind of external library or downloaded source code is strictly forbidden. The students are only allowed to use the standard library that comes with the chosen programming language. The only exception to this rule is to include code that is your own previous work (maybe coming from previous assigments at previous computer science subjects). If you include code that is your own previous work, you are required to document it propperly (when did you generate it, for what assignment and what subject).

6 Report

As a deliverable of this work, each group should write a report regarding the design and implementation of the path tracer. There are no requirements regarding page count or structure of such report. However, we have the following recommendations:

- **Structure** your document before writing it. You will avoid redundancies or missing information.
- When showing renders, **discuss** them. What are the renders illustrating? Which phenomena is relevant?
- It can be useful to **compare renders** side by side to illustrate a specific feature (with and without it).
- Do **not include source code**. You are already submitting source code.
- It is normal to draw inspiration from external sources (papers, online information, Wikipedia or even source code). When this happens **add bibliographic references** to the papers, books or material you got inspiration from. Note that there is a fine

line between "getting inspired" and "copying and pasting code".

The content of the report should include:

- A description of the **decisions** that you have made during the design and implementation of the path tracer, relating them to the teaching materials and linking them to images (renders) of different scenes that illustrate each design decision.
- An analysis of your workload: how have you managed your own work, which tasks each of the members of the group has undertaken, how did you share and improve the code, which methodology did you use...
- A set of nice looking renders, including information about render time and a discussion about the features illustrated by the render. At least, two of them should be a Cornell Box (one with an area light, and the other one with a point light) with indirect illumination.

Besides that required content, the report should answer the following questions:

1. Write and describe **the Render Equation** and discuss how does your path tracer accout for it.
2. Regarding the **convergence** of path tracing:
    a) How fast does path tracing converge with respect to the number of paths per pixel? Illustrate with renders. Bonus point if a numerical analysis is included.
    b) Of the implemented materials, which ones make convergence slower? In which circumstances? Why?
    c) Which light sources make path tracing converge slower? Area lights or point lights? In which circumstances? Why? Illustrate it with renders.
3. Regarding the **global illumination** effects (hard shadows, soft shadows, color bleeding, caustics):
    a) Which of these global illumination effects are easily obtained with path tracing? Why? Illustrate them with renders and insets (zoom in) to specific areas.
    b) Which scenes do you need to obtain each of this effects? Discuss scene requirements for each of the effects: material properties, light source properties, geometrical distribution of objects...
    c) Which of these global illumination effects cannot be obtained (or are very hard to obtain regarding convergence time)? Why?

Additionally, answer the following **for each implemented extension**:

1. Describe the implemented extension, including specific design decisions.

2. Relate the implemented extension with the teaching materials and the bibliography: has it been discussed in class? When? If not, add bibliographic references of the materials that have helped you implement the extension.
3. How does the extension improve over the basic version of the path tracer? Illustrate it with side-by-side renders, comparing rendering times...

The quality of the report will also be considered for the evaluation. A longer report does not necessarily equal a better report.

7 Submission

The submission must be done through Moodle (`https://moodle2.unizar.es/add`) by one (and just one) of the students.

Source code. All the source code of the ray tracer should be compressed in a `.zip` file, with the following file name:

- `pathtracer_<nip1>_<nip2>.zip` (where `<nip1>` and `<nip2>` are the 6-digit student IDs) if the work has been done between two students.
- `pathtracer_<nip>.zip` (where `<nip>` is the 6-digit student ID) if the work has been done individually.

It should also include a *readme* file with clear compilation and execution instructions.

Report. The report should be delivered as a `.pdf` file, with the following file name:

- `pathtracer_<nip1>_<nip2>.pdf` (where `<nip1>` and `<nip2>` are the 6-digit student IDs) if the work has been done between two students.
- `pathtracer_<nip>.pdf` (where `<nip>` is the 6-digit student ID) if the work has been done individually.

References

[FHF+17] Luca Fascione, Johannes Hanika, Marcos Fajardo, Per Christensen, Brent Burley, and Brian Green. Path tracing in production - part 1: Production renderers. In *ACM SIGGRAPH 2017 Courses*, SIGGRAPH '17, pages 13:1–13:39, 2017.

[Kaj86] James T. Kajiya. The rendering equation. In *Computer Graphics (Proc. of SIGGRAPH)*, 1986.