Mathematical Methods for Image Synthesis - Reading/Project 1

Nicolas Bonneel

In this class you will need to do one project and one article reading that will be presented at the end of the semester. If you take a project with me, you will need to take a reading from Julie's part, and conversely, if you take the reading with me, you'll need to take a project from Julie. A project is expected to take between 15 and 30 hours, and can be implemented with any *imperative* programming language you want. Readings go much more in-depth than the class lectures, and are thus more complex.

1 Readings – weeks 1-2

You can choose between these 3 options :

- Implicit Visibility and Antiradiance for Interactive Global Illumination, Dachsbacher et al. 2007 `https://www-sop.inria.fr/reves/Basilic/2007/DSDD07/ImplicitVisibilityAndAntiradiance.pdf`
- Manifold Exploration : A Markov Chain Monte Carlo Technique for Rendering Scenes with Difficult Specular Transport, Jakob and Marschner, 2012 `http://www.cs.cornell.edu/projects/manifolds-sg12/manifolds-sg12.pdf`
- A Frequency analysis of Light Transport, Durand et al. 2005 `https://people.csail.mit.edu/fredo/PUBLI/Fourier/`

2 Project (for those who haven't done it already)

For this project, we require you to implement a simple path tracer, dealing with spheres and a static camera. The following lectures notes will take you to the implementation of such a path tracer. For this project, we will

implement a path tracer with diffuse, pure specular and pure transparent materials, using only point lights, handling shadows, indirect lighting and gamma correction. This corresponds to all parts described before the "optional part", and took 3 classes of 4 hours to implement (with some supervision) for Centrale students :)

— French version : `https://goo.gl/hWNL6l`
— English version : `https://goo.gl/3a9X7z` This is currently a very poor google translation of the french version, but I will improve the translation over the next few weeks.

While we allow for any imperative language, the runtime cost can be expensive, and using slow languages (i.e., matlab or python) is not recommended. You will not be penalized if you choose these languages, but this can make debugging harder for you (especially when launching multiple rays per pixel).

3 Project (for the others)

You are asked to implement a naive radiosity solver. For that, you will need to implement :

— An OBJ mesh loader. You may use a library or copy existing code.
— A ray-Geometry intersector. You will need to implement a BVH datastructure to accelerate these intersections, as your code will require many such intersections (in the order of N^2). See `https://youtu.be/Fr328siHuVc?t=4212` + erratum.
— These intersections will be used to evaluate a form factor $G_{i,j}$ between triangles i and j, as $G_{i,j} \approx \frac{1}{n}\sum_{k=1}^{n} \frac{<u_k,N_i><-u_k,N_j>A_j}{d^2(x_{i,k},x_{j,k})}$ where $u_k = \frac{x_j - x_i}{d(x_{i,k},x_{j,k})}$, $x_{i,k}$ is the k'th sample on triangle i (resp j), d is the distance function, and N_i is the normal of triangle i (resp. j), A_j is the area of triangle j (this term is a simplification of $\frac{A_i A_j}{A_i}$ that is due to the pdf of the two sampled triangles and the area of triangle A_i). Formulas for generating samples over a triangle can be found in Philip Dutré's online book : Global Illumination Compendium `https://people.cs.kuleuven.be/~philip.dutre/GI/`.
— Use the formulas seen during the lecture to implement an iterative solver of the radiance, and store the radiance as a color property of triangles in a mesh (while the OBJ mesh format does not support colors per triangles, the OFF format allows that and is as simple).