Nikolai Hofmann, Marc Stamminger *Erlangen, 12/06/2025*

Global Illumination Tutorials (exercise sheet 3)

In this assignment we will look into importance sampling and common applications, such as sampling environment maps and selecting light sources for shading. Both are important tools to handle complex lighting, for example from an HDR environment map captured by a natural photograph.

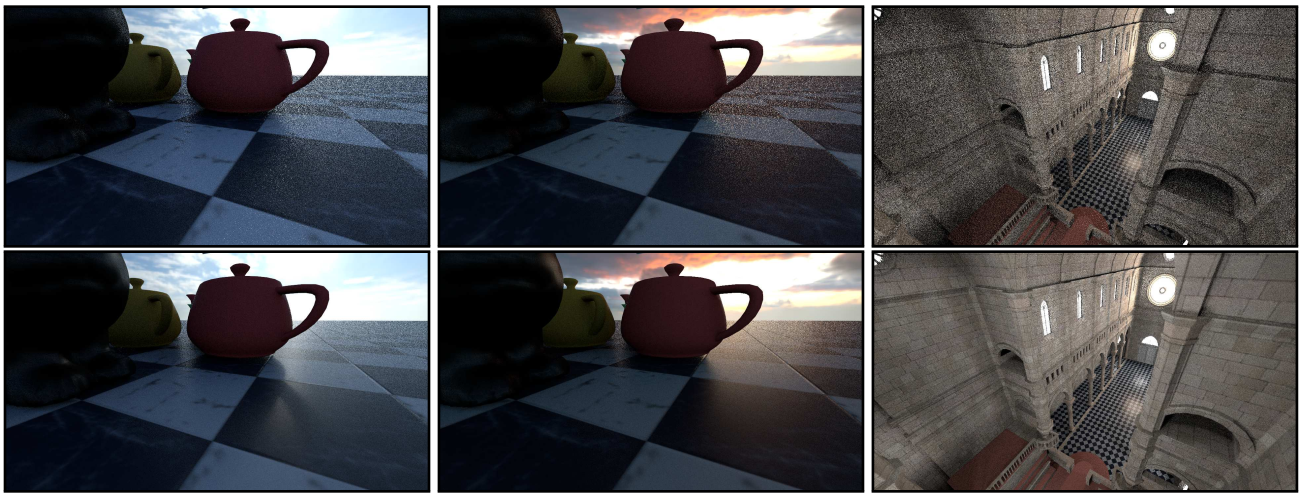
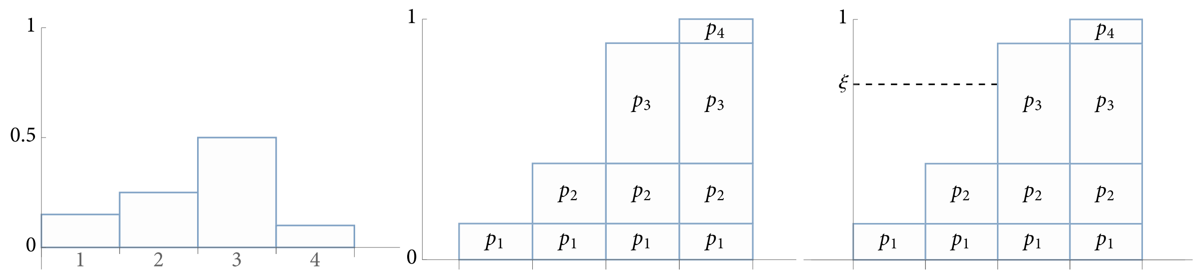


Figure 1: Comparison of direct illumination using uniform sampling (top row) and importance sampling (bottom row) at equal number of samples. Left column: an environment map with a very bright spot (sun). Middle column: Same scene lit by a sunset environment map. Right column: The Sibenik cathedral where each triangle of each window is an area light source, relying on a discrete 1D distribution for sampling. Note how importance sampling improves convergence for all scenes.

Lecture: on Tuesdays from 12:15 - 13:45
Tutorials: on Thursdays from 09:00-12:00
Submission deadline: Wednesday 02.07.2025, 23:55
Grading period: Thursday 3.07.2025, 09:00-12:00

The goal of importance sampling a spherical environment map, i.e. a 2D texture, is to select brighter pixels – which represent small light sources – more frequently than darker ones. To this end, we need to map uniform random samples to the brightness distribution in the texture. In order to still converge to the correct mean, we also need to compute the respective PDF for each pixel. We will separately look at sampling 1D and 2D distributions, as sampling a 2D distribution relies heavily on the former.

Render the `configs/a03.json`, `configs/a03_sunset.json`, or `configs/a03_sibenik.json` configuration for the left, middle and right column in Figure 1. Have a look at the interface in `src/gi/distribution.h`. You may also consult the PBR book (third edition) in chapters 13.3 and 13.6. For grading, please pre-render the images from Figures 1 and 3 using your solution.



(a) Discrete Probabilities (b) Discrete CDF (c) Inversion Method

Figure 2: Sampling from a discrete distribution consisting of four probabilities p_1 to p_4 with $\sum_i p_i = 1$.

Assignment 1 [5 Points] (Sampling 1D Distributions)
[Files: `src/gi/distribution.cpp`]

To start, implement sampling a piecewise constant 1D function from a given array of function values, similar to Figure 2. To this end, build a CDF from function values while ensuring a probability density and implement the lookup of a real, piecewise linear inverse PDF. Take care to normalize to $[0, 1]$ when building your CDF, e.g. $\int_0^1 f(x)dx$, and avoid division by zero.

There are two distinct sampling schemes: *continuous*: return a floating point value in $[0, 1)$ according to the PDF, and *discrete*: return an integer in $[0, N)$ according to the PDF, where N is the number of function values. Implement the continuous case in `Distribution1D::sample_01` and the discrete case in `Distribution1D::sample_index`. Set the out parameter `pdf` to the probability density of this respective sample. As the CDF is monotonically increasing, implement a *binary search* algorithm to deal with large arrays performantly. Take care to adjust the PDF for the continuous and discrete case accordingly. Note the generated debug output, which you can compare to Figure 3.

Assignment 2 [3 Points] (Sampling 2D Distributions)
[Files: `src/gi/distribution.h, src/gi/distribution.cpp`]

Now, extend the previously implemented 1D sampling scheme to 2D, to be able to sample from a 2D texture. Fill in the data structures required for sampling 2D distributions via conditional and marginal densities. In order to do this, create a `Distribution1D` for each row of pixels in the image and set up a marginal distribution over the individual rows. Finally implement the *continuous* sampling scheme for a 2D distribution in `Distribution2D::sample_01`. Similarly, set the out-parameter `pdf` to the respective PDF. You can now compare your renders to Figure 1 and debug output to Figure 3, however it will not exactly match, yet.

Assignment 3 [1 Points] (Countering Spherical Distortion)
[Files: `src/gi/light.cpp`]

Due to mapping a rectangular 2D texture onto a unit sphere, areas close to the poles become distorted. Since we effectively consider each pixel in the environment map as a small light source illuminating the scene, the surface area – and thus emitted light – of each pixel changes with the distortion. To this end, we need to skew the brightness distribution of the environment map by applying a scaling factor before building the `Distribution2D` in the `SkyLight` constructor. Find and apply the appropriate scaling factor for spherical mapping[1], which is one at the equator and zero at the poles.

[1] `https://en.wikipedia.org/wiki/Spherical_coordinate_system`

Assignment 4 [1 Points] (Applicability to BRDFs)

The inversion method we just implemented is a general importance sampling approach that is valid for any arbitrary discrete function. In the last assignment, however, we relied on analytical inversion of the continuous GGX microfacet distribution to enable importance sampling. Are the two approaches interchangeable? I.e. can we importance sample the BRDF from last assignment using the approach from this assignment, and vice-versa for environment maps? If this is the case, which approach should be preferred? Please prepare your answers for grading.

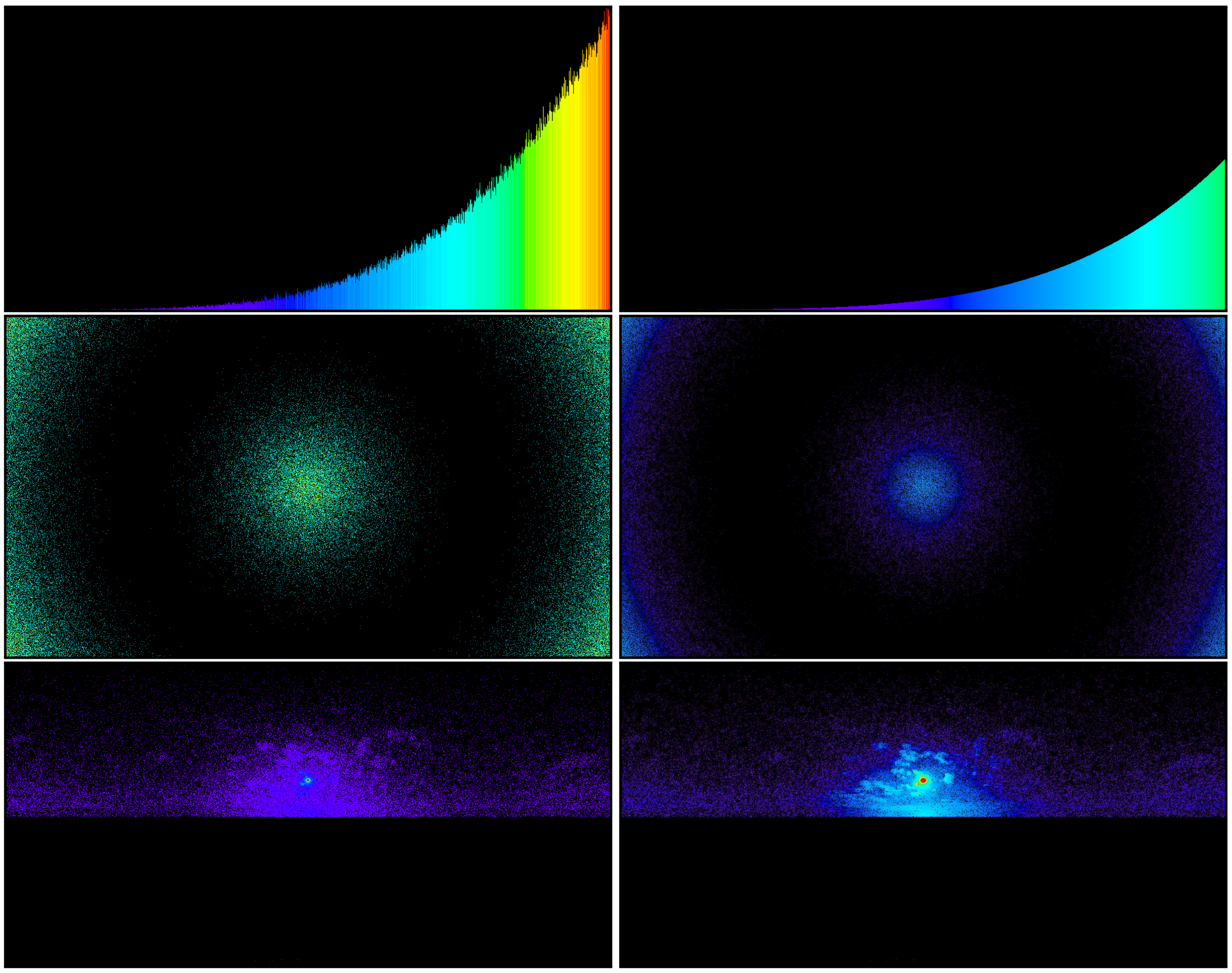


Figure 3: Heatmaps of the resulting sampling patterns (left) and their PDFs (right) from different distributions for debugging. Top row: 1D distribution from a power function. Middle row: 2D distribution from a circle SDF. Bottom row: 2D distribution from the sun environment map.

Happy Hacking! :)