# Worksheet 12

For many scattering materials we are only interested in the radiance emerging on the surface. This is especially true for highly scattering materials like skin, where we definitely will not place the camera inside the volume. Since the material still scatters light beneath the surface, we would in principle have to do a volume rendering, but the fact that we only need to worry about light emergent on the surface means that we can use an approximate analytical expression to describe the light diffusion under the surface. The following exercises are about using the dipole approximation for computing subsurface scattering.

## Learning Objectives

- Simulate subsurface scattering using diffusion.
- Use the dipole approximation for rendering translucent materials.
- Integrate the dipole approximation with path tracing.
- Explain the BSSRDF and how it relates to volume rendering.

## Subsurface Scattering

The material properties used in subsurface scattering are the same as the material properties used in volume rendering (complex index of refraction $n$, phase function $p$ or asymmetry parameter $g$, and scattering coefficient $\sigma_s$).

- Load the Cornell box (`CornellBox.obj`) and the blocks inside it (`CornellTallBlock.obj` and `CornellShortBlock.obj`) into your ray tracer and set the scene scale to millimetres. Ensure that the material of the tall block is marble and render a reference image using your Monte Carlo volume rendering shader (illumination model `illum 13`, implemented in the previous set of exercises). As we need a very large number of paths per pixel to render the volume, it is important not to use splitting at diffuse surfaces (in `RenderEngine.cpp`, set the number of splits to 1 for the `mc_lambertian` shader and set the number of area light samples to 1). Store the resulting image such that you can compare it to the subsurface scattering result later.
- Subsurface scattering involves four major steps. The first step is to compute a single scattering contribution. This is a simplification of the full multiple scattering implemented in the previous set of exercises. The trick is simply to stop the multiple scattering computation after one scattering event. Render the marble block using single scattering only. (In the framework, single scattering can be implemented in the `single_shade` function of the file `MCSubsurf.cpp`. Switch the illumination model of the marble material to `illum 14` and render it again.) Store the resulting image.
- The three remaining steps evaluate the diffusion part of the BSSRDF as described in **P** and by Jensen et al. [2001, see reference below].[1] Do the following: (a) sample points of incidence on the surface of the translucent object, (b) sample incident illumination at these points of incidence, and (c) compute the diffusion of the incident illumination to the point where light is emergent using the Fresnel transmittances and the dipole approximation. (In the `pathtrace` project of the framework, implement the missing parts of the functions `init_sample_surface` and `diffusion` in `MCSubsurf.cpp`.)
- Choose the number of surface samples per pass and render an image showing the result for the diffusion term only (in `RenderEnging.cpp`, set the number of samples to 500, for example, for the

[1] If you use the original article [Jensen et al. 2001], be sure to check all formulae against another reference [Donner 2006, for example] as the original article has a couple of errors.

`mc_subsurf` shader). Store the result. Then render an image which shows the result of single scattering and diffusion combined. Store the result.

- Load the scene with a bunny in an outdoor environment (see Worksheet 2). Set the material of the bunny to one with subsurface scattering (`illum 14`). Render the scene without single scattering, but with sampling of both direct illumination and environment illumination. Note that the sampling of incident radiance will be different for these two cases. (In the framework, implement simplistic multiple importance sampling that picks one case or the other with equal probability in the function `init_sample_surface` of `MCSubsurf.cpp`.) Store the resulting image and compare it to the case of a Lambertian bunny. You may want to increase the intensity of the sun to display the translucency of the object (you can do this in `SunSky.cpp`).

## Worksheet 12 Deliverables

Reference images of the marble block in the Cornell box rendered using volume rendering with multiple scattering. Rendered images of the marble block showing the single scattering term, the diffusion term, and the combined result, respectively. A final image showing a translucent bunny in an outdoor environment. Resolution down to $128 \times 128$ is acceptable for the Cornell box images. In addition, include a comparison of the Cornell box subsurface scattering result with the reference image, and a comparison of the translucent bunny with a Lambertian bunny.

## Reading Material

The curriculum for Worksheet 12 is

**P** Sections 11.4. *The BSSRDF*.
**P** Section 15.4, 15.5. *Subsurface Scattering*.
- Dal Corso, A., and Frisvad, J. R. Point cloud method for rendering BSSRDFs. Technical report, 2018.

Alternative literature available online or uploaded to CampusNet:

- Jensen, H. W., Marschner, S. R., and Levoy, M., and Hanrahan, P. A practical model for subsurface light transport. In *Proceedings of ACM SIGGRAPH 2001*, pp. 511–518, ACM, 2001.

Additional resources:

- Donner, C. Subsurface scattering of light using diffusion. In *Towards Realistic Image Synthesis of Scattering Materials*, Chapter 5, PhD dissertation, University of California, San Diego, 2006.
- Frisvad, J. R., Hachisuka, T., and Kjeldsen, T. K. Directional dipole model for subsurface scattering. *ACM Transactions on Graphics 34*(1), pp. 5:1-5:12, November 2014. Presented at SIGGRAPH 2015.