Worksheet 13 (optional)

Camera and eye models are an important part of realistic rendering. We cannot make a rendering of a 3D world without such a model. Usually, we employ the simplistic modified pinhole camera model, where the camera is described by the parameters listed in Table 1 (and it is assumed that a pixel is a square). In this set of exercises, the assignment is to explain the theory behind more advanced camera and eye models by annotating code that implements these models.

Tabel 1: User specified camera parameters.

Outer parameters		Inner parameters	
$\boldsymbol{e}$	Eye point	ϕ	Field of view (vertical)
$\boldsymbol{p}$	View point	d	Camera constant
$\vec{u}$	Up direction	W, H	Camera resolution

Learning Objectives

- Use advanced camera and eye models.
- Render an image that exhibits depth of field and explain the camera model behind it.
- Render an image that exhibits glare and explain the eye model behind it.
- Explain the theoretical models behind GPU-based demo programs.

Depth of Field

The simplistic camera model does not include any lens effects. The lens is modelled only by an angle and a distance: the field of view ϕ and the camera constant d (see Table 1). This means that objects at all distances from the camera will appear sharp in the rendered image. A real lens has a *depth of field*, that is, an interval of distances from the camera in which objects will appear sharp. The farther away from this interval an object is placed, the more blurry will it appear in the image. To include the depth of field effect, we need a more elaborate camera model. In particular, we need parameters to describe the shape of the lens. The parameters most commonly used for camera lenses are the focal length f and the aperture stop k. The focal length is the distance from the lens where collimated incident light (parallel rays) are focused in a point. The aperture stop $k = f/\ell$ is the ratio between the focal length f and the diameter of the lens ℓ. Including these two parameters, or similar parameters holding the same information, allows rendering of images that exhibit depth of field effects. An extended camera model for rendering depth of field effects using ray tracing was presented by Cook et al. [1984, see references below]. This model has now been implemented using GPU ray tracing and is included as one of the sample programs in the NVIDIA OptiX SDK. The assignment is to try out this sample program and explain how it works by annotating relevant parts of the code.

- Locate the OptiX SDK on your computer. Ensure that you have writing access to the location where the OptiX SDK is placed on the computer (in the VR-lab, you need to copy the OptiX SDK folder to the Desktop). Load the SDK into CMake (run CMake (cmake-gui) and choose the SDK subfolder as the location of the source). Specify an SDK/build folder as the location where you would like to build the binaries. Press Configure a few times. Then press Generate and build to Visual Studio. The SDK folder will now have a build folder which contains a Visual Studio solution that holds all the OptiX SDK samples. Open the solution, choose the `cook` project as start-up project, and find out how it works (run it and check the usage information in the command prompt window).
- Render an image exhibiting depth of field with no aliasing artifacts. (When you press 'r' the image will improve as long as a frame rate is printed in the lower left corner.)

- Read the code and use the theory from the lecture on depth of field and the references to explain the relevant parts of the code. Find out how the depth of field camera works, what the camera configuration is, and how it is implemented in OptiX. In principle, you should annotate the relevant code.

Glare and Fourier Optics

There are two approaches we can take in realistic rendering. We can either try to render an image as if it were taken with a digital camera or we can try to render it as if it were seen through the eyes of a human observer. While cameras and eyes are similar in some ways (they both use a lens for focusing light), they are also quite different as a camera is composed of homogeneous, inorganic materials while the eye is made of organic, particulate substances. In daylight conditions, the heterogeneous, particulate nature of the ocular substances is rarely noticed by the brain. However, in high contrast lighting environments such as a bright light source observed at night time, the diffraction pattern from the particles in the eye will be quite obvious to the viewer. This diffraction phenomenon is called *glare*. We can simulate glare using Fourier optics. This has been implemented by Ritschel et al. [2009, see references below] as an image processing technique using GPU shaders. A demo program is available at DTU Inside File Sharing (in the code folder).

- Build and run the glare demo program. The program depends on GLUT and GLEW, but headers and binaries are included, so you should be able to build executables without too many problems. Running the program should result in a candle light with a simulated glare surrounding it. Take a screenshot.
- Read the keyboard function (or at least the comments) in `glare.cpp` to find out what other images you can get from the program. Run the program again and take screenshots of the different components behind the final result.
- Use the screenshots, theory from the lecture, and relevant parts of `glare.cpp` to explain how the glare phenomenon is simulated.
- If you have extra time, try to find a different input image and draw a source overlay image (start with a black background and draw white pixels where you would like to have intense sources in the original image). Run the program with this new set of images as commandline arguments.

Worksheet 13 Deliverables

Annotated code and screenshots explaining how to simulate depth of field effects in ray tracing and glare effects as a post-processing technique.

Reading Material

The curriculum for Worksheet 13 is

P Sections 6–6.2. *Camera Models*.
P Sections 13.6.6. *Sampling Cameras*.
- Ritschel, T., Ihrke, M., Frisvad, J. R., Coppens, J., Myszkowski, K., and Seidel, H.-P. Temporal glare: real-time dynamic simulation of the scattering in the human eye. *Computer Graphics Forum (Proceedings of Eurographics 2009) 28*(2), pp. 183–192, April 2009.

Alternative literature available online or uploaded to CampusNet:

- Cook, R. L., Porter, T., and Carpenter, L. Distributed ray tracing. *Computer Graphics (Proceedings of ACM SIGGRAPH 84) 18*(3), pp. 137–145, July 1984.