02562: Project Proposals

Jeppe Revall Frisvad

November 2023

Material appearance modelling (using a “principled BSDF”)

- Pick a material and characterize its intrinsic visual properties (use photos).

- Explore the parameters of a principled BSDF (in Blender, for example) and find parameter settings or build shader trees that reproduce the discovered visual properties of the material.

- Provide a physical explanation of the link between shader parameters and visual properties (physical properties are recommended over artistic properties).

- Construct an appearance model that defines reasonable variation of the parameters without violating the visual properties of the material.

OptiX: GPU accelerated ray tracing

- Try out your newly acquired ray tracing skills in another ray tracer.
- Re-implement the rendering effects of the course in kernels running on the GPU.
- Get progressive updates at real-time frame rates.
- Perhaps add environment mapping and a shader for rough transparent materials.

- There is an OptiX version of the render framework.

- A *project initiator* worksheet is available.
- Next lecture is on GPU ray tracing with OptiX.

Importance sampling environment maps

- Sampling of a cosine-weighted hemisphere is inefficient when a high-dynamic range (HDR) environment map is the light source.

- A method for generating distribution functions from tabulated data:
  - Sampling Piecewise-Constant 1D Functions
    `https://pbr-book.org/4ed/Sampling_Algorithms/Sampling_1D_Functions#SamplingPiecewise-Constant1DFunctions`
  - Piecewise-Constant 2D Distributions
    `https://pbr-book.org/4ed/Sampling_Algorithms/Sampling_Multidimensional_Functions#Piecewise-Constant2DDistributions`
  - Image Infinite Lights
    `https://pbr-book.org/4ed/Light_Sources/Infinite_Area_Lights#ImageInfiniteLights`

- This can be used for importance sampling an HDR environment map. Demonstrate its more efficient convergence when shading objects and that it enables appropriate casting of shadows onto a holdout geometry.

Rendering with a microfacet-based BSDF

- Extend the shader for transparent objects to include reflection and refraction from a rough surface.
- Sample a microfacet normal and use this instead of the macroscopic surface normal.

- Include explicit evaluation of direct illumination (which requires evaluation of the microfacet BSDF expression).

- Explore the range of materials that this model can represent.
- Could you render both dielectrics (glass) and conductors (metal)?

Reference

- Walter, B., Marschner, S. R., Li, H., and Torrance, K. E. Microfacet models for refraction through rough surfaces. In *Proceedings of Eurographics Symposium on Rendering (EGSR 2007)*, pp. 195–206. Eurographics Association, 2007.

Rendering with a measured BRDF

- Implement a shader that uses measured material appearance.
- Rendering such a material in an HDR environment can lead to highly realistic images, but rendering convergence may be slow.
- Adjust the tone mapping of the rendered result.

- Extra:
  - Consider use of importance sampling to make the rendering more efficient.
  - A method for generating distribution functions from tabulated data is available (other project).
  - This can be used for importance sampling an HDR environment map.
  - It can also be used for importance sampling a tabulated BRDF.

- A lecture note is available (and a *project initiator* worksheet).

Other projects

- Ray-object intersection for a mathematically defined object (like a spline surface).
- Rendering with depth of field (https://pbr-book.org/4ed/Cameras_and_Film/Projective_Camera_Models#TheThinLensModelandDepthofField).
- Procedural solid texturing (like a wood texture applied to the Stanford bunny) or procedural mesoscale geometry (defined by a signed distance field).
- Proper tone mapping of rendered HDR images (http://www.advancedhdrbook.com/).
- Rendering with the matrices that a camera calibration produces.
- Rendering with textured light sources (representing monitors or projectors).
- Comparing rasterized and ray traced rendering results.
- Exploring neural radiance fields (NeRF) (https://www.matthewtancik.com/nerf, https://github.com/bmild/nerf, https://jatentaki.github.io/portfolio/gaussian-splatting/).
  - Could you render images in your WebGPU renderer using a NeRF or scene data retrieved from a NeRF (e.g. NeRFactor)?
  - Could you generate a synthetic dataset for NeRF using Blender or your WebGPU renderer?