UNIVERSITÄT TÜBINGEN
Prof. Dr.-Ing. Hendrik P.A. Lensch
Lehrstuhl Computergrafik
Dennis Bukenberger (dennis.bukenberger@uni-tuebingen.de)
Zabiha Kehribar (zabiha.kehribar@student.uni-tuebingen.de)

7. December 2020

Graphische Datenverarbeitung
Assignment 5

Submission deadline for the exercises: 14. December 2020 6.00 am

Source Code Solutions

- Upload **only** the source code files listed in the description in Ilias.
- Upload them one by one and **don't** zip them.

The source code must run on `cgpool120[0-7].informatik.uni-tuebingen.de` by extracting your submitted `.tar.gz` file to a certain folder and running `scons`. You can log onto this machine via ssh by using your WSI account. From outside the WSI network, you may have to use a ssh gateway, e.g. `cgcontact.informatik.uni-tuebingen.de`
Attention: Do not use `cgcontact` for working, but only for ssh-ing to the `cgpool` machines.

The framework you get for completion already compiles and runs as requested above and you only have to modify source and header files - no files have to be created.

Written Solutions

Written solutions have to be submitted digitally as one PDF file via Ilias.

Framework Update

Download the new framework from the course website. There are some new improvements:
Scenes now contain texture coordinates in a `.uv` file. Additionally, the filenames of textures are stored with the material information in the `.scn` files. The framework loads this information and was extended with texture objects and UV texture coordinates.
Beside textures, the updated framework supports some more shaders:

- Key 1: Normal debug shader.
- Key 2: UV coordinate debug shader.
- Key 3: Mip level debug shader.
- Key 4: Diffuse color only shader.
- Key 5: Simple shader.
- Key 6: Path shader.

Scenes

To load scenes, call `./coRT SceneName EnvmapFile`. The framework contains the following three scenes:

Bunny is the already known Bunny scene which doesn't contain any light sources and should therefore be rendered with an environment map to be lightened. Load it with `./coRT Bunny pisa_oct.hdr`

CornellBox is a box with some geometrical objects and a light emitting plane in it and can be rendered without environment map. Is is loaded per default or with `./coRT CornellBox`

Checkerboard approximates the checkerboard scenes of the lecture's slides. Load it with `./coRT Checkerboard`

5.1 Textures (40 Points)

Add textures to the rendering process.

a) Interpolate the uv-coordinates. The uv-coordinates are accessed and interpolated the same way as vertex normals.

b) Use the interpolated uv-coordinates to access the texture. To do that, replace any use of the diffuse material color during shading (`Render::shade_noshading(...)`, `Render::shade_simple(...)` and `Render::shade_path(...)`) with a componentwise product of the diffuse material color and the texture color (obtained with `Material::GetTextureColor(coords)`).

5.2 Mipmapping (60 Points)

a) Implement the helper methods of the texture that are necessary for mip mapping.

b) Extent the `Texture`'s constructor so that it calculates all mip levels of the texture.

c) Implement the retrieval of a texture sample from mipmaps.

d) Incorporate the mipmap textures into the current ray tracing framework by calling `Material::GetTextureColor(...)` with two parameters. The second mip level parameter must be $\log_2(\text{rec.dist} \cdot c)$ where c is a constant that has to be tuned manually for this exercise.
Hint: You may use constant mip levels to debug the creation and sampling of mip levels.

e) Render the Checkerboard scene with and without mipmapping, each way with only one camera ray sample per pixel. Send the resulting images along with the code.

5.3 Bonus: Procedural Shaders (15 Points)

a) Implement the 2D Perlin Noise function.

b) Use it to add procedural textures to the test scene from exercise 4.

c) Render some nice images.

Don't submit code for this exercise! Just send some nice images.