Advanced Topics in Computer Graphics II

Universität Bonn
Institut für Informatik II
November 5, 2019

Winter term 2019
Prof. Dr. Reinhard Klein
Alexander Dieckmann, Lukas Bode

Sheet R2 - AxF and Shading
UPDATE

Parsing the AxF File

In order to load the AxF files, we provide parser code, that relies on libhdf5. Therefore, at first you have to install it. On most Linux based operating systems you can just install it from the package manager. E.g. on Ubuntu you can type:

```
$ sudo apt-get install libhdf5-dev
```

Now you can test your setup by doing the following in the folder where AxfReader.zip is located:

```
$ unzip AxfReader.zip
$ mkdir build
$ cd build
$ cmake ..
$ make
$ ./AxfTest
```

This compiles and runs the AxfTest test code. It utilizes AxfReader.hpp to load the happy_birthday.axf file we produced in the first exercise meeting and prints the shapes and some values of the various data textures it contains. A "test successful!" in the console indicates that everything works fine.

In order to use the AxfReader in your own project, you have to copy the "HighFive" folder and the AxfReader.hpp file into your project's source folder. To make the code work you now have to modify your project's CMakeLists.txt file, such that it can find libhdf5 and HighFive:

```
...
set(CMAKE_CXX_STANDARD 11)
set(CMAKE_CXX_STANDARD_REQUIRED ON)

find_package(HDF5)

...

add_executable(${PROJECT_NAME} ...)

target_include_directories(${PROJECT_NAME} PRIVATE
    ${CMAKE_SOURCE_DIR}/HighFive/include/
    ${HDF5_INCLUDE_DIRS}
    ...
)

target_link_libraries(${PROJECT_NAME}
    ${HDF5_LIBRARIES}
    ...
)
...
```

The provided CMakeLists.txt file that builds AxfTest can be used as reference.

Since the code consists of header files only, you do not have to compile anything in advance. You can just include it in your program and it will be compiled on demand.

Using AxfReader.hpp

The AxF file contains (among other stuff) the following textures:

Data	Symbol	String Key
Diffuse Albedo	ρ_d	diffuse
Normals	N	normal
Specular Albedo	ρ_s	specular
Roughness	α	lobes
Anisotropic Rotation	φ_α	aniso
Fresnel F_0	F_0	fresnel
Displacement	D	displacement
Transparency	T	transparency

The anisotropic rotation φ_α rotates the specular highlights in tangent space and F_0 is the Fresnel reflection coefficient for 0 degree inclination. The other ones should be self-explanatory.

By including the AxfReader.hpp file and using the AxfReader::readAxf(...) function, you can read them from the AxF file. AxfReader provides two maps which contain all the data. One of those maps a string key of a texture to its dimensions and the other one maps the same string key to the texture data as one large std::vector<float>. If you need other details, take a look at the AxfReader.hpp source code.

Ward Shading Overview

In order to avoid confusion, this is a quick overview of how you should use the textures mentioned above to implement your Ward shading. Therefore, we assume that $\cdot(u, v)$ samples texture $\cdot$ at position u, v, V is the viewing direction, and L is the incoming light direction. Both V and L have to be normalized and in the local surface coordinate frame. Information on how to transform them from world/view space into the needed tangent space based on well defined texture coordinates can be found here: `http://foundationsofgameenginedev.com/FGED2-sample.pdf`.

So, for a surface point with texture coordinates u, v, we can describe the shading calculation as follows:

$$\hat{H} = L + V \tag{1}$$

$$H = \frac{\hat{H}}{|\hat{H}|} \tag{2}$$

Fresnel term (Schlick's approximation):

$$w_F = F_0(u, v) + (1 - F_0(u, v)) \cdot (1 - \cos\theta_V)^5 \tag{3}$$

$$\cos\theta_V = H * V \tag{4}$$

Anisotropy (rotate unnormalized half vector):

$$H' = \begin{pmatrix} \cos\varphi_\alpha(u, v) & \sin\varphi_\alpha(u, v) & 0 \\ -\sin\varphi_\alpha(u, v) & \cos\varphi_\alpha(u, v) & 0 \\ 0 & 0 & 1 \end{pmatrix} \hat{H} \tag{5}$$

Lambert diffuse:

$$k_d = \frac{\rho_d(u,v)}{\pi} \tag{6}$$

Ward specular:

$$k_s = \rho_s(u,v) \cdot \frac{1}{\pi \alpha_x(u,v)\alpha_y(u,v)} \cdot \frac{1}{4|L * H'|^2 (H'_z)^4} \cdot \exp\left(- \frac{\left(\frac{H'_x}{\alpha_x(u,v)}\right)^2 + \left(\frac{H'_y}{\alpha_y(u,v)}\right)^2}{(H'_z)^2} \right) \tag{7}$$

Final color (not accounting for transparency):

$$C = \sum_{i \in \text{Lights}} (k_d + k_s \cdot w_F) \cdot L_i \cdot \cos\theta_i \tag{8}$$

For further details please look at `https://www.xrite.com/-/media/xrite/files/whitepaper_pdfs/axf/axf_whitepaper_en.pdf` Section 4.1.1 and Appendix B.

Remarks

Implementing this will not look very realistic for most materials. Therefore, you should also implement at least some sort of transparency. If you have other techniques in mind, that would improve on the rendering of your material, please give them a try as well.

Good luck !