TU Clausthal

GPU Programming **WiSe16/17**
Prof. Thorsten Grosch, Johannes Jendersie

Exercise 2: Textures and Samplers

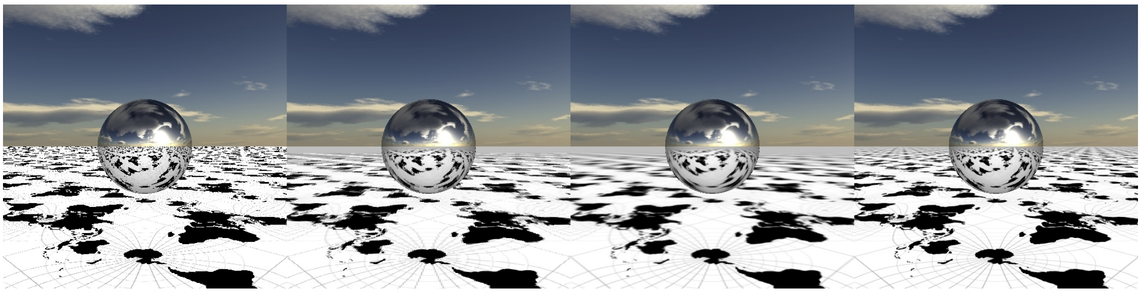

Figure 1: A 2D texture and a cube map with different samplers: nearest no mipmap, all nearest, all linear and all linear with 16 anisotropic samples.

Task 1: Implement the Textures 3P

Start with the implementation of textures in the framework (*`texture.cpp`*). The texture class is designed to be compatible with all types of textures (1D, 2D, 3D, Arrays, ...). The concept is always the same: Generate the texture, allocate memory and fill that memory. The second and third step are separated in the framework to allow the loading of array textures and cube maps which require multiple loads of 2D images. In the allocation step we will use the `glTexStorage..` family. It differs from the `glTexImage` calls in that it allocates the entire mipmap chain and does not need the image data at this point in time. Hence, glTexStorage is a bit easer and cleaner. To upload the data use `glTexSubImage`.

To avoid tedious work most of the types are not included this week, but will be added for you later. Only 2D textures and cube maps must be implemented. Therefore, some parts of the code (e.g. switches) look useless in the current state.

Task 2: Implement the Sampler Objects 2P

Sampler objects control how a texture is read on GPU side. Most important they define filtering operators, if a texture is read in a location between texels. Further, filtering can integrate over areas in the texture, if the pixel covers more than one texel. Border handing and texel comparisons make sense in some scenarios like shadow mapping.

In *`pipeline.cpp`* the implementation of the sampler class must be completed.

Note: the pipeline will grow in the next week and will make much more sense.

Task 3: Bring them Together 2P

Finally, you can complete the demo itself. Load the textures and create 8 samplers for different combinations of filters (which is given in the source code).

Theory

Be prepared to answer the following questions.

Q1 How is a texture be created?

Q2 Why is not using a mipmap so noisy (flickering animation)?

Q3 Is using mipmaps faster or slower?

Q4 How does a cube map lookup work?

Q5 In old OpenGL sampler states were set per texture. What are the dis/advantages of the separated approach?

Q6 What is anisotropic filtering? How does it work?