The exercises will take place in room G40 in Mühlenpfordtstrasse 23. Your y-account is sufficient to login and access all tools. Ctrl+Alt+T gives you a terminal and g++ is your GNU C++ compiler.

Each week you must complete the assignments and hand in your commented source code for the practical tasks, as well as your solutions to the theoretical tasks (with drawings/formulas). Please use different colors in your drawings and also make sure that formulas are recognizable in your source code. Your group must present the completed assignments on Friday, 13.01.2017, 9:45.

From now on it is also required that you send me your source code to foerster@cg.cs.tu-bs.de (without the data folder) before Friday 9:45!

---

7.1 KD-Tree Acceleration (70 Points)

So far our ray tracer has not used any acceleration structure for reducing the number of ray/primitive intersections. This works well for small scenes, but for larger ones we need a data structure to speed up the process of finding the first hit of a ray with the primitives. In the lecture you have learned about kd-trees. Have a look at common/kdmtree.cpp and implement the missing parts. Also have a look at the ObjectModel::intersect(Ray * ray). When determining the split dimension use the dimension in which your local bounding area is the widest. When determining the split position, use the median of the minimum bounds of the primitives. If everything is implemented correctly, the result should still look like the image below but render a lot faster.
7.2 Supersampling (20 + 10 Points)

a) Until now, we have used exactly one ray per pixel, with the pixel being a rectangular area \( p = a \times a \) and the ray going through its upper left corner. However, as you may have already noticed this leads to aliasing artifacts for large ray distances. To counteract this effect, we use multiple rays per pixel and calculate the average. Subdivide the pixel into \( n = s \times s \) equally sized regions, all of which are sampled once. The final pixel color is the average over all samples. Take a look at the SimpleRenderer and create your own SuperRenderer class (files and everything), which allows setting \( s \) and which renders a super sampled image.

Remember to add your superrenderer files to the Makefile.

b) Create a scene in which the differences between normal sampling and super sampling becomes very obvious (aliasing artifacts). Render both images for comparison.

A little help for the Raytracing Challenge:

I have included a Qt-based Texture class, which you can find in common/texture_qt. If you are using Qt you can use this texture class instead of the old one. This allows you to load more image formats (.png, .jpg, ...) instead of just .ppm. There is also a savePPM function, which allows you to convert textures to .ppm. This should help you loading models with custom textures and creating pretty scenes.

Check the internet for free models that you can use. For example:

http://www.turbosquid.com
http://www.blendswap.com
https://3dwarehouse.sketchup.com

IT'S A CHRISTMAS TREE WITH A HEAP OF PRESENTS UNDERNEATH!

... WE'RE NOT INVITING YOU HOME NEXT YEAR.