Planning Hair rendering thesis

Jeffrey Lemein

1. **Setting up the project** 32 hours
    - Set up build system by using cmake, and set up proper directory structure.
        - Keep all builds and renderings out-of-source, so that source files and directories never get cluttered.
    - Prepare scripts to:
        - simplify hair models by Cem Yuksel (for faster rendering)
        - A render script, that preprocesses PBRT scene files to set render properties.
            - Render frame script should render a single frame
            - All settings must be able to be specified via this script (using a property file with key value pairs, or on command line by key=value).
            - Each rendering of a frame should result in a new directory in the output directory together with all data to re-render on the fly when needed.

2. **Use PBRT to render example scene using path tracing (reference material)** 32 hours
    http://www.pbrt.org
    - Set up PBRT
    - Find/Create a realistic example scene using realistic lighting
    - buy additional memory to have at least 16GB RAM, because:
    - Curly hair model is ~750 MB, containing more than 3 million hair strands.
        - I asked forum and it is likely that I need more RAM.
    - Be able to render a path-traced image (that serves as the reference material for my final renderings). See below what is included in PBRT. Cem Yuksel has more hair models on his website.
        - http://www.cemyuksel.com/research/hairmodels/

3. **Use OpenVDB to create a voxel density grid** 40 hours
    http://www.openvdb.org/
    - Needed to store densities of the hair per voxel
    - Should be run once per hair model and can be reused for different renderings.
    - Used to quikly find the density when a ray is propagated through the hair volume.

4.	**Migrate marschner/dual scattering shader code to PBRT** • Should be relatively painless, both are written in C++. • The only difference is that before it was linked as plugin to Pixar's Renderman and now it should be part of PBRT scene.	32 hours
5.	**Adjust code to create Marschner/Dual Scattering lookup table** • Lookup data contains precomputed values for Marschner and Dual-Scattering algorithms. • Code is already there, but there were flaws with it	24 hours
6.	**Render scattering responses to file** • use visualization tool to plot results, • investigate visualization tools (probably Matlab is used) • a file with properties must also be passed as argument	32 hours
7.	**Analyze response data** • fit mathematical formula to scattering distribution • invert scattering distribution formula • Find out how to use discrete approach to finding the inverted function	80 hours
8.	**Code sampling strategy in PBRT**	60 hours
9.	**Render samples using my optimized implementation (with importance sampling)** • Render same scenes as in step 2.	16 hours
10.	**Evaluate performance of algorithm** • measure rendering time and quality (noise) • compare versus dual scattering papers • compare versus path tracing result → compare to path noise example using 1024 samples and 32 integration steps.	40 hours
11.	**Add everything in thesis**	40 hours
	Total 428 hours (11 weeks full time)	