Prof. Dr. Leif Kobbelt
Kersten Schuster, Philip Trettner

Visual Computing Institute | RWTH AACHEN UNIVERSITY

Real-time Graphics Assignment 9

Date Published: January 17th 2017, Date Due: January 24th 2018, 11:44

- The assignments have to be done in groups of 2 students.
- Hand in the solutions to the exercises via L^{2}P.
- You are only allowed to change code inside the marked strips (STUDENT CODE BEGIN/END)!
- Any questions? → L^{2}P discussion forum or rtg@cs.rwth-aachen.de!

If not done yet, obtain the (publicly accessible) exercise framework and assignments from `https://www.graphics.rwth-aachen.de:9000/Teaching/rtg-ws17-assignments/`.
Use **git pull** to fetch the newest changes of the framework (including the code for this exercise).

The **only** files that you should modify and **upload**:

- Assignment09.cc
- fullscreen.bloom-downsample.fsh
- fullscreen.bloom-kawase.fsh
- fullscreen.bright-extract.fsh
- fullscreen.tone-mapping.fsh

Description and Further Help In this assignment you will add an HDR/Tonemapping pipeline with Bloom. Please note that the code strip for the HDR pipeline (`Assignment09::renderHDRPass()`) is not devided into smaller parts (brightness extraction, downsampling, Kawase blur, and tone-mapping). Note that you will have a black / corrupted screen until you render to the appropriate targets. However, you can tell the TweakBar to output a specific render target to screen. As always, you find code strips in the framework with more detailed comments and hints. You can find some screenshots in the folder `screenshots`.

Performance Hints If your hardware is somewhat weak, the following might help:

- Reduce shadow map size (or disable shadows completely)
- Keep the render distance small
- Reduce your window size
- Disable point light sources

Exercise 1 HDR/Tonemapping Pipeline with Bloom [10 Points]

- Integration / C++: Implement the method `renderHDRPass()` in `Assignment09.cc` which consists of 4 "sub passes" (brightness extraction, downsampling, Kawase blur, and tone-mapping). Note that the Kawase blur actually consists of 5 blur iterations with different kernel sizes (0, 1, 2, 2, 3).
- Shader Code / GLSL: Write the respective shaders.
  - fullscreen.bright-extract.fsh
  - fullscreen.bloom-downsample.fsh
  - fullscreen.bloom-kawase.fsh
  - fullscreen.tone-mapping.fsh