Prof. Dr. Leif Kobbelt
Kersten Schuster, Philip Trettner

Visual Computing Institute | RWTH AACHEN UNIVERSITY

Real-time Graphics Assignment 7

Date Published: January 16th 2020, Date Due: January 30th 2020, 14:29

If not done yet, obtain the (publicly accessible) exercise framework and assignments from
`https://www.graphics.rwth-aachen.de:9000/Teaching/rtg-ws19-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**:

- `Assignment07.cc`
- `common.glsl`
- `fullscreen.transparent-resolve.fsh`

Description In this assignment you will migrate the framework to a deferred shading pipeline and implement order-independent transparancy.

Controls You can place blocks with left mouse button (green overlay). Keep Ctrl pressed to remove blocks (red overlay). Keeping Shift pressed while clicking (yellow overlay) stores the clicked block's material (pipetting). The currently existing materials are sand, grass, dirt, rock, snow, snowrock, crystal, gold, copper, bronze, water, lightsource, and air.
W, A, S and D keys can be used for navigation while the right mouse button allows you to rotate the camera. Press space bar for jumping and hold shift for faster walking. To toggle between freecam mode and character controller, press key F.

Further Help As always, you find code strips in the framework with more detailed comments and hints. You can find some screenshots in the folder `screenshots`. A summary of the intended pipeline can be found in `rendering-pipeline.jpg`. The UI has a lot of settings to play with. Especially the "Output" setting in the "debug" group might be helpful for debugging.

Exercise 1 Deferred Shading [$1 + 1 + 2 = 4$ Points]

The code strips for this exercise are in `Assignment07.cc`.

(a) Clear the G-Buffer properly and perform the opaque rendering pass. Pay attention to the correct `depthFunc` state and be aware that we need to write to an sRGB buffer.

(b) Implement the depth pre-pass. Do not forget to clear the depth and set the `depthFunc` state.

(c) Implement the light pass

- Do a fullscreen lighting pass (ambient light + directional (sun) light)
- Render the point lights on top

Exercise 2 Transparent Objects [$1 + 3 + 2 = 6$ Points]

(a) Render the background (skybox) in `fullscreen.transparent-resolve.fsh`.

(b) Render translucent objects with “weighted, blended order-independent transparency” (transparent pass in `Assignment07.cc` and shader functions in `common.glsl` and `fullscreen.transparent-resolve.fsh`)

(c) Implement refraction via a distortion map (shader functions in `common.glsl` and `fullscreen.transparent-resolve.fsh`)

Exercise 3 Shadows (Theory) [0 Points]

(a) Briefly explain how basic shadow mapping works.

(b) Explain the simple idea behind *Percentage-Closer Filtering* and give an example for why it cannot create realistic soft shadows.

(c) Can you simply blur the shadow map to create soft shadows? Why/why not?

(d) A typical one-channel color sampler can be used as shadow map. The alternative is to use a shadow sampler. Explain its advantage.

(e) Sketch a situation where the screen size of a shadow texel becomes really large.

(f) What is the idea of cascaded shadow mapping and how does it work?