Realtime Rendering WS2020: Ants

Jonas Prohaska, Marie-Sophie Pichler

November 18, 2020

1 Description of Implementation

The prototype was implemented in C++ using the OpenGL [5] specification. We integrated GLFW to create a window and included GLEW [3] and OpenGL Mathematics [6] for the (current) basic rendering. Yet to be implemented are animations and the optimization, to draw all models with the same shader together. The models and obj files were downloaded from cgtrader [2] and were edited and assembled together in Maya. The chosen effects in the original documentation (Shadow mapping with percentage-closer filtering GPU particle system with at least 100.000 particles) have not changed for the final project but due to this being a prototype have not yet been implemented. The demo was testes on an AMD RX 5700 as well as on NVIDIA GeForce GTX 1060.

2 Additional Libraries

The positions of the models saved in a json file and get parsed using the jsoncpp library [4]. The actual loading of the obj files is done by assimp [1] while textures are loaded from PNG using Simple OpenGL Image Library 2 [7]. Other dependencies inlude GLFW, GLEW and GLM mentioned in 1.

3 Controls

- "F": Toggle between orbit camera and user controlled camera
- Mouse: Adjust camera direction (in both modes)

When using user controlled camera (default):

- "W": Move camera up
- "S": Move camera down
- "D": Move camera right

- "A": Move camera left

When using orbit camera (toggle with F):

- "W": Move camera up
- "S": Move camera down
- "D": Move camera closer
- "A": Move camera further away

References

[1] Assimp. http://www.assimp.org.

[2] cgtrader. https://www.cgtrader.com.

[3] Glew. http://glew.sourceforge.net.

[4] Jsoncpp. https://github.com/open-source-parsers/jsoncpp.

[5] Opengl. https://www.opengl.org.

[6] Opengl mathematics. https://glm.g-truc.net.

[7] Soil2. https://github.com/SpartanJ/SOIL2.