Proseminar Computer Graphics
Summer Semester 2015

Programming Assignment 4

Hand-out: June 5th, 2015

Hand-in: June 26th, 2015, 11:59pm

Topics

- Texture mapping
- Billboarding and transparency
- Advanced effects

Outline

The overall goal of the programming project during the semester is to create a shader-based OpenGL program showing an animated merry-go-round. This target will be realized in four steps, each representing a programming assignment. In each step the complexity of the scene will be further improved, until a final full 3D animated model with several effects is realized.

The general idea of the fourth and final assignment is the further enhancement of the visual fidelity of the previously implemented animated scene. The 3D scene is composed of differently shaped and animated objects on the merry-go-round, including an external reference geometry. A focus of the assignment is on using texture mapping, as well as integration of advanced effects to improve visual appearance. For this some of the concepts introduced in the lecture will be implemented. Appropriate code should be developed following the shader-based OpenGL programming paradigm. The implementation should build on the previous programming assignments. In addition to the previous assignments, a brief development report shall be prepared, as outlined below.

Tasks

1. Add basic texture mapping to your scene. Include at least two visually different textures, mapped to two differently shaped basic geometries. One of these should comprise a non-planar surface section for the mapping. Appropriately combine the already implemented scene lighting with the texture mapping.

2. Add at least one billboard to your scene. The billboard should include transparency (e.g. via alpha blending). The billboard can be located at a fixed position, e.g. the root of the mobile.

3. Implement an advanced visual effect in your scene. You are free in the choice of the selected method(s). One option could be an advanced texturing effect, such as normal mapping or environment mapping. Another option is the inclusion of basic shadowing or mirroring. Please feel free to suggest other effects. Get in touch in case of doubt or questions on appropriate methods. The selected visual effect and its implementation shall be described in a brief report.

Submission of your solution is due on June 26th, 2015 at 11:59pm. Please submit all developed source code (i.e. no executables) as well as the report in PDF in a ZIP file via OLAT.

Development should be carried out in teams of 2-3 students – please continue in the same teams or get in touch if any changes are required. Make sure to acknowledge all team members in the code and submission (please name the ZIP file with all surnames). Only one submission per team is required. Provide adequate information on program usage.

Finally, please respect the academic honor code. Each team should work on its own on the solution of the assignment. Any attempts at plagiarism or collusion will lead to 0 marks and further scrutiny. Please get in touch in case of questions or problems with regard to the assignment.

In total there are 20 marks achievable in this assignment as well as 15 marks for the report.

Implementation Remarks

Your solution should be developed using OpenGL 3.3 and GLSL 3.3. You can check available drivers and versions on Linux with the command `glxinfo`. All required libraries are installed on the machines in the ZID computing labs, e.g. RR14, RR15, RR19, RR20.

While you are free to develop in any environment, the final submission should compile and run in Linux. As reference, the machines in RR15 will be considered. Make sure to provide a Makefile with your submission, so that after unpacking your solution can be compiled directly in your top-level directory via `make`.

Test early enough in case you develop in a different hardware environment. If you intend to work on your own setup, it is preferable to employ vendor-specific (e.g. Nvidia, AMD) driver implementations (instead of using e.g. Mesa OpenGL). Also note that in addition to OpenGL, the packages freeglut and GLEW are used in the examples. For the solution a shader-based programming approach should be targeted. Follow a proper coding style and provide appropriate comments in the code.

Resources

- Lecture and Proseminar slides as well as code and information is available via http://igs.uibk.ac.at/ (→ Teaching)
- OpenGL homepage
  http://www.opengl.org
- OpenGL 3.3 reference pages
  http://www.opengl.org/sdk/docs/man3
- Legacy information
  http://www.opengl.org/wiki/Legacy_OpenGL
- OpenGL Tutorial
  http://www.opengl-tutorial.org
- GLUT implementation freeglut
  http://freeglut.sourceforge.net/
- OpenGL Extension Wrangler GLEW
  http://glew.sourceforge.net/
- D. Shreiner, G. Sellers, J. Kessenich, B. Licea-Kane, “OpenGL Programming Guide: The Official Guide to Learning OpenGL, Version 4.3” Addison-Wesley, 8th edition, 978-0321773036, 2013.

Note: Be mindful of employed OpenGL and GLSL versions!