Karlsruhe Institute of Technology
Computer Graphics Group

Max-Gerd Retzlaff * Emanuel Schrade † Tamás Szép ‡

Graphics Programming Lab, Winter Term 2015/2016
Assignment 5

Freestyle Project

Abstract

In this final assignment of our course, you can demonstrate your graphics programming skills on a completely custom application. This problem can be different from the topics covered in the previous assignments, but you are also free to build upon and extend your existing solutions.

Deadlines/Timeline:

8.12.2015	Call for Proposals
15.12.2015	Topic mail deadline
18.12.2015	Topic finalization
5.1.2016	Draft proposals
12.1.2016	Final proposals
2.2.2016	Assignment upload deadline
9.2.2016	Presentation and evaluation

1 Overview

The goal of this assignment is the creative solution of a custom problem, using the previously demonstrated graphics programming techniques. For this reason, the assignment is organized in a different way than the previous ones. Instead of a strict and exact specification for implementation and evaluation, we only provide general guidelines for this task, and we leave the entire concept to your choice.

Thanks to the completion of the preceding problems you already have a basic knowledge about the OpenGL graphics pipeline. You are also familiar with widely used techniques in terrain rendering, volume visualization, basic material models (Phong shading) and design concepts of interactive modeling applications. The assignment consists of four main steps:

1. Initial concept
2. Qt and OpenGL implementation
3. Evaluation
4. Presentation

*retzlaff@kit.edu
†schrade@kit.edu
‡tamas.szep@kit.edu

2 Concept

As a first step, you will need to write a short specification (about 1-2 pages length). The specification should define the problem and how it can be solved using the graphics hardware. If you are going to extend a previous assignment, you should clarify the improvements compared to the existing solution.

We encourage you to search for literature references and tutorials on your preferred technique. Presumably, you will find very useful resources for your work, including theoretical background, optimization hints and best practices.

You will need to get your plans accepted before you start the implementation itself. The deadline for submitting your specification is the 6th of January 2015 (via email). We will accept your specification if it has similar complexity to the previous assignments, and the definition of the problem is clear enough. Every concept should address a unique problem (we will not accept two concepts on ocean rendering, for example).

The overall structure of the concept should be organized as follows:

1. **Introduction.** Summarize the motivation and goals of your project.
2. **Rendering techniques.** Short description of the chosen problem, with references to the used literature or presentations related to the used rendering techniques.
3. **Implementation.** Outline how do you plan to implement your methods in OpenGL and Qt. This is a very short summary.

2.1 Teams

You might have chosen a problem that you find too complex for a single person, but you are still motivated to work on it. As a specialty of this assignment, we allow you to seek partners and work in a small team instead of individual assignment. In this case, the length of the specification, and ideally the size of the problem should grow linearly with the number of team members.

Please note that we do not accept any excuse regarding team-management problems. We require you to work on independent parts of your project, so if one person is late with an implementation detail, it should not block the progress of the team-mates.

Each team member will present her / his own work individually, and must have a separate scoring table in the evaluation part of the specification.

3 Presentation

As part of the grading, you should prepare a short presentation (no longer than 10 minutes per person) to introduce your solution to the other members of the course. Each implementation will be fundamentally different, therefore we feel it beneficial to share your results and discuss your problems together. We will make no evaluation in the ATIS lab: until the 3rd of February 2015 you should submit your solutions electronically and the presentation (including the grading) will take place before the end of the semester (most probably the same day).

The presentation itself is only a brief summary followed by a short discussion. The structure of the presentation should roughly follow the concept, but extended with demo(s) and results of your running application (an OpenGL 4.2 capable GPU will be provided in the system used for the presentation). The presentation should conclude with performance measurements, similarly to the recent assignments.

4 Evaluation

Akin to the former assignments, you can receive 20 points in total, in the following parts:

- 2 points: initial concept
- 15 points: implementation (distributed by you)
- 3 points: short presentation

The distribution of the 15 points will go to your custom problem. You should propose a scoring table for your assignment as well, as part of the initial concept. Using the previous assignments as guidelines, you should break down your implementation into smaller steps and assign points for each step. We will use this scoring table during the evaluation, so this assignment is the most flexible one. However, we will correct unrealistic scoring tables before we accept the specification.

5 Examples

Without limiting your possible applications, we summarize a few potential freestyle projects:

5.1 Advanced post-processing effects

This is a large field of rendering techniques applied after the scene has been already rasterized and the render targets contain all visible surfaces. The main advantage of these methods is that they operate on the full rendered image, therefore their computational cost does not depend on the scene complexity. A good example for advanced post-processing is the rendering of Depth of Field (DoF), which is too expensive in a physically accurate way for current games.

`http://http.developer.nvidia.com/GPUGems3/gpugems3_ch28.html`

`http://dxp.korea.ac.kr/homewiki/images/f/f5/PG-PointDof-submit.pdf`

5.2 Realistic water rendering

`http://madebyevan.com/webgl-water/`

5.3 Parallax occlusion mapping

In nature, surfaces are almost never completely smooth, but real-time models lack the necessary detail to capture fine details, like bumps or small stones on the ground. Such rough phenomena is usually modelled in a displacement texture, and sampled during rendering to increase the detail on the surface. Parallax occlusion mapping is a per-pixel displacement method, a very popular technique in current games, because it creates the illusion of displacement without increasing the geometric complexity.

This technique is mostly used in terrain rendering, where the base mesh is approximately planar.

`http://sirkan.iit.bme.hu/~szirmay/egdisfinal3.pdf`

5.4 3D tanks game

You are probably familiar with the classic “Tanks” or “Worms” games, where multiple players are placed on a simple 2D terrain and are allowed to fire at each other with various weapons. The projectiles explode as they hit the ground, and create large craters after them. An example implementation can be played on this link:

`http://www.mathsisfun.com/games/tanks.html`

As an option, you can use your existing terrain implementation as a base of a 3D tanks game. You can create a randomly generated terrain and scatter the models of the players over it. You can directly use the 2D height-map during the gameplay, which is also base of your terrain rendering technique: this is how you can detect collisions with the projectile, and you can also create the craters easily by reducing the height values in the map.

Be creative, but also try to put the main focus on graphics (since we are a graphics programming course). You should model explosions using animated particle systems, and creatively use your blended fracture textures when rendering burnt craters. In your modelling framework, you have already implemented multiple viewports; it would be a good idea to have a split-screen multiplayer mode, as a trivial application of the concept.