Guidelines for Project in EDAN35 High Performance Computer Graphics

Michael Doggett* Second Author†

Lund University
Sweden

Abstract

In this paper, we describe the two types of projects that can be done in the course EDAN35 High Performance Computer Graphics at the Department of Computer Science, Lund University. In order to pass this part of the course, the requirements for each type of project are described, and as well the rules for those of you who wish to participate in the competition. The project need also be documented, and you should use the style used in this very paper for that, so there is a section on formatting as well.

1 Introduction

As part of the course EDAN35, each group of two[1] students need to create a project as well. There are two types of projects:

- 3D Graphics applications based on a 3D engine, for example the RenderChimp framework, or
- Graphics hardware optimization (GHO)

The project is (unofficially) worth 3 points.

Any group who wants can participate in one of two separate competitions. There is one competition for each type of project.

2 3D Graphics Project

In this type of project, you have a lot of freedom. You need to write an application using a 3D Graphics API, such as OpenGL, a rendering engine, such as the RenderChimp framework, which is either cool, beautiful, interesting, useful, fun to use, or a combination of all of the above. Examples of applications may include:

- 3D graphics algorithm. From recent paper or GPU programming text, such as GPU Gems
- game
- screensaver
- 3D GUI for mobile platforms
- useful tool
- something completely different

Remember though that as a group of two, you need to spend nearly 6 weeks on this. This obviously means that if you decide to make a screen saver, it should be *very* challenging, otherwise, you will not pass.

The performance of your application is not the most important thing for the project. If you do optimize for performance, that is a good thing (especially for the competition).

You can use the code from the *Deferred Shading Assigment* when you start with this type of project.

*e-mail: mike@cs.lth.se

†second.author@cs.lth.se

[1] In some cases, groups with only three or one student have to be formed, but this should be avoided as much as possible.

2.1 Who wins the 3D Graphics competition?

A jury will decide at the last lecture (2011-12-07, 10:00-12:00).

The deadline is at 12:00 on the 6th of December, 2011, i.e., the day before the last lecture and competition. The report and code should be delivered by then. However, note that modifications to the code can be done after that (in order to further impress the jury).

3 Graphics Hardware Optimization (GHO) Project

If you choose to do this project, you will have to use the software framework that we have provided. The task description is simple: given 3072 bytes of on-chip memory, implement whichever "hardware"-algorithms (in software) you wish in order to reduce the total amount of bandwidth usage to external memory. As you have seen in *Assignment 1*, a texture cache [Hakura and Gupta 1997] can be implemented in order to reduce bandwidth related to texturing. In the lectures, other techniques have been described for reducing bandwidth usage.

It is your task to implement the techniques that you think will reduce the total amount of used memory bandwidth the most. Here is your chance to be clever, invent new algorithms, or new combinations of algorithms, and reduce memory bandwidth usage to levels never reached before. You also have a chance to analyze the amount of bandwidth in the given animated scene, and determine which part of the pipeline you want to target first.

The screen resolution must be QVGA, i.e., 320×240 pixels. The image quality does not need to be exactly the same as in the reference implementation, but it should be very similar. If you doubt whether a particular approximation is reasonable, stop by and talk to us about that.

All accesses in cached memory (3072 bytes) are assumed to be for free. The rest of the memory accesses must be counted. Obviously, there are simple ways to cheat here, but your code will be checked after the project deadline, so we will discover it sooner or later. So, be honest when counting memory accesses. When in doubt, come and ask us.

The animated scene, which you need to measure bandwidth usage for, is included in the code handout.

3.1 Who wins the GHO-competition?

The group that uses the smallest amount of accesses to external (off-chip) memory (measured in bytes).

Deadline is at 12:00 on the 6th of December, 2011, i.e., the day before the last lecture. The report and code should be delivered by then.

4 Written Report

The report should be handed in an PDF format. You can use any word processing software you like, but you need to generate a PDF for the final submission.

The typesetting for this paper was done using pdfLaTeX. It is recommended that you use this as well, and therefore the "source files" for this very document is available on the course website. If you are not familiar with pdfLaTeX, then you may use whatever other word processing program you like, as long as you mimic the

Figure 1: This is the logotype for LUGG: Lund University Graphics Group.

general style in this paper (i.e., you paper should look similar to this paper).

The source files consists of two style files: `acmsiggraph.bst` and `acmsiggraph.cls`, and these should be placed in the directory as the files `project.tex` and `project.bib`. There is also a PNG image called `lugg.png` that is shown later in this paper. It is in `project.tex`, where you should delete this document's text, and instead write your own text.

You should write your report as a scientific paper. It should have (at least) the following sections:

- Abstract [brief summary of key results]
- Introduction [why do what we did? motivation]
- Algorithms or Application [description of what you did, what algorithms you implemented]
- Results [e.g., performance, **screenshots**, usefulness etc]
- Discussion [what did not work, what worked well, what can be improved, optimizations you tried, how could we improve graphics hardware in mobile phones]
- Conclusion [any concluding remarks – skip if you do not have anything to say in addition to what you've already said]

Some groups may not have enough important material to write a "Discussion"-section, and in such cases, that section can be omitted. Also, look at scientific papers, e.g., [Aila et al. 2003; Akenine-Möller and Ström 2003; Hakura and Gupta 1997; Igehy et al. 1998], and try to follow their general style. When in doubt, come and ask us. If you need references not found in the file `project.bib`, simply add your reference to that file. The report should be 2–4 pages long.

You can include screenshots as PNGs or illustrations in the PDF format. An example is shown in Figure 1. See the source file `project.tex` for how this is done in LaTeX.

5 Conclusion

Make a great project. You're clever. Surprise us (and the jury)!

References

AILA, T., MIETTINEN, V., AND NORDLUND, P. 2003. Delay Streams for Graphics Hardware. *ACM Transactions on Graphics, 22*, 3, 792–800.

AKENINE-MÖLLER, T., AND STRÖM, J. 2003. Graphics for the Masses: A Hardware Rasterization Architecture for Mobile Phones. *ACM Transactions on Graphics, 22*, 3, 801–808.

HAKURA, Z. S., AND GUPTA, A. 1997. The Design and Analysis of a Cache Architecture for Texture Mapping. In *24th International Symposium of Computer Architecture*, ACM/IEEE, 108–120.

IGEHY, H., ELDRIDGE, M., AND PROUDFOOT, K. 1998. Prefetching in a Texture Cache Architecture. In *Workshop on Graphics Hardware*, ACM SIGGRAPH/Eurographics.