Realtime 3D Graphics Programming Using the Quake3 Engine

Wagner Daniel, Bernhard Kainz, Dieter Schmalstieg
Graz University of Technology
{ wagner | kainz | schmalstieg } @ icg.tugraz.at

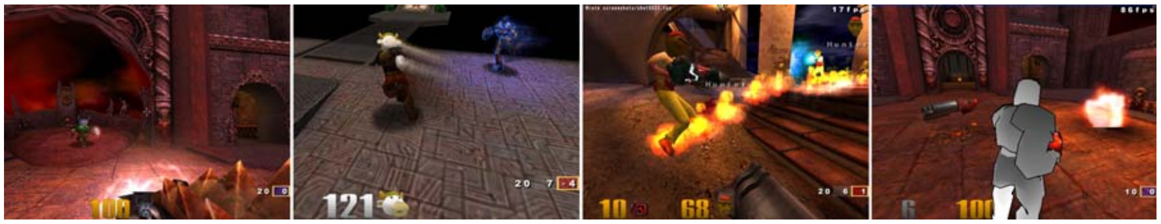

Figure 1: Various 3D effects implemented by students in 2006. Left to right: Environment mapping (level reflecting on a sphere), Motion blur (using Render-to-Texture), Particle effect, Toon shader (fragment shader).

Abstract: *We present a lab assignment that accompanies a complete module called "Real-time Graphics". The students' task is to get familiar with content creation and programming a (previously) commercial 3D engine. In a first task, students have to create 3D content, which is integrated into the Quake3 engine. In a second task, the students have to implement a simple animation and finally add an impressive 3D graphics effect to the Quake3 engine. The lecture has been taught four times from 2004 to 2007. We present the assignment and report on experiences that we have gained.*

Keywords: Real-time graphics, 3D engine, Quake3

1 Introduction

The lecture "Echtzeit-Graphik" (Real-time Graphics) is taught every winter semester at the Graz University of Technology. In contrast to most lectures of this kind we aim at letting the students experience problems and typical solutions faced in industry. The lecture is accompanied with a lab assignment. During the lab assignment the students have to extend the (now open source) Quake3D engine with new content and graphical effects. During the course, the students have to get familiar with the Quake3 engine, which is the basis for several dozens of commercial games. The original Quake3 code is written in pure C and not well documented. However, it serves as a good example for clean and efficient engine design, and has been extended by us with sample material that minimizes the learning curve of students and lets them focus on the graphics-specific problem set. Figure 1 shows outcomes of the lab assignments from 2006.

2 Educational Goals

The goal was to let the students experience programming a (previously) commercial 3D engine during a lab assignment. The students are expected to have visited basic lectures on 2D and 3D graphics and have to be able to implement C/C++ applications on their own.

In order to facilitate diving into using the Quake3 engine, we provide two example applications that are discussed in detail during the lecture. A framework for integrating 3D effects, containing the two example applications, is presented. The framework allows the development of simple 3D effect without having to deal too much with the inner workings of the Quake3 render engine. The sample code provides hooks that enable the developer to parse content before it is rendered and to store important data such as projection or modelview matrices. We also provide project files for Microsoft Visual Studio and Linux makefiles, so the students can choose their preferred platform.

3 Methodology

The semester project is split into four submissions. The first submission requires the student to select a 3D object that she will model later on, select an animation to implement and to decide for a 3D effect that shall be integrated into the render engine. This first assignment acts as a final registration for the course and forces the student to think about the upcoming tasks. Therefore the submission for this task has to be done in the first week of the lecture and is graded with 5% of the overall points.

For the second assignment students have to model a head for a Quake3 player model. The goal is to make students familiar with modeling tools and raise awareness for important real-time graphics topics such as polygon count. We recommended the freeware modeling tool gmax due to its similarity to professional modeling tools like 3D-Studio Max and available export plugins for the Quake3 mesh file format (*.md3). We also give a Tutorial with the basics of gmax and present known pitfalls for the mesh export. However, the students are also allowed to use tools they already know or prefer more.

In the third assignment the students have to integrate the newly modeled head into the Quake3 engine and animate it using C/C++ programming. This assignment prepares the students for the upcoming major task of programming a 3D effect and forces them to learn about the actual C++ development environment.

The final and major task (credited with 2/3 of all points) requires the students to program a custom 3D effect and integrate it into the render engine. The student can choose between different difficulty levels which are sketched by topics like “Particle Effects”, “Motion Effects” or “Shader Effects”. A thorough tutorial that explains the internals of the Quake3 engine is held shortly after the third submission and presents the students with all details to start working.

The schedule as well as the points credited for each submission put the focus on the last assignment. Yet we decided that was important to start slowly and also include content creation rather than just programming.

4 Assessment

The presented lab has been taught four times since 2004. In the first year, Quake3 was not yet available as open source; so Quake2 was used. In the first year, students were asked to model a level for the Quake2 engine. Although we advised students to focus mostly on the programming task, many students complained later on that the level modeling took too much time.

Shortly before the second year started, Quake3 was released as open source, which allowed us to switch to a more modern and powerful 3D engine. The modeling task was reduced to creating a completely new player model. Yet again this task was rated as too time consuming and demanding.

In the third year we further reduced the modeling task to creation of a player model’s head only. Due to positive ratings we kept this exact mode for the fourth year.

The only evaluation criterion for the first assignment is the submission in time. After this assignment first hints for the feasibility of the model and the desired effect can be given. In some cases the submission has to be rejected due to a too demanding or impossible proposition and an alternative model or effect is worked out with the student. The first Tutorial introduces the students in gmax and shows up how to produce a real time usable model. This includes the very restrictive specification in Quake3, that a closed mesh may not have more than 1000 Vertices. Submissions for the second assignment which do not fulfill this restriction will be rejected. The quality of texturing is besides the vertex count the main assessment criteria for this task. The student can get 10% of the overall points for a well designed model.

The third task seems to be easy since only the model has to be exported into an md3-mesh, integrated in a Quake3 readable archive (.pk3) and animated with some matrix transformations. Nevertheless, students who are not used to use a more complex software development environment may run into troubles. Submissions are graded due to the following criteria: A running binary file (to test this we provide a reference system in our labs) has to be submitted with a correctly integrated model, which implies the submission of a working Quake3 add-on archive. Correct matrix transformations relatively to the body disposition are together with creativity and prettiness required to get all points for this assignment.

These first three tasks have to be submitted within the first two months of the exercise lecture. For the fourth task the students have another two months. Depending on the proposed effect the participants have to implement various difficulty levels. Simple particle effects or Cg-Shader effects can be fully implemented within our framework whereas other effects require some new changes in the Quake3 rendering engine. Therefore this task is assessed by two main criteria: Complexity and creativity.

In general the submissions showed the same split in the last years. About 25% were excellent work, including a good story-line (matching character model and effect) and presentable effects. However, complexity and creativity not always call for each other. A good effect done with a simple implementation can also be graded with all points.

To guarantee the communication between the students and for direct contact to the supervisors we established, besides a mailing list and the TU-Graz online system, a newsgroup in 2007. The newsgroup turned out to be the best way to come along with uprising problems. Not only that this relieves the lecturers because of students answering their questions themselves, the problems from one class are also archived and can be studied in the next year.

Further motivation is given by a student competition. This competition can be won by the student whose assignments are voted by the supervisors as the most beautiful submission. Yet, this ranking does not influence the overall grade. In the last years the award was a quite big piece of chocolate and the first place on the "Hall Of Fame" website published on our public web server for one year.

5 Conclusions

Using a commercial 3D engine such as Quake3 turned out to be highly motivating for students. The fact that most of the students played Quake3 and several other games based on the Quake3 engine before raised a lot of interest in the lecture. The circumstance that the engine was extremely expensive (several 100k dollars for a single title) until shortly before it was made open source further raised the students interest.

As a result, most students do much more work voluntarily than required for the lecture. Even though we emphasize that a simple 3D effect is enough to get a high grade, many students invest hundreds of hours into programming their assignments. Figure 1, Figure 2 and Figure 3 show results of the work these highly motivated students. It is also noteworthy that the lab assignment can be concluded with a LAN game party with the objective of fragging the lecturer as much as possible.

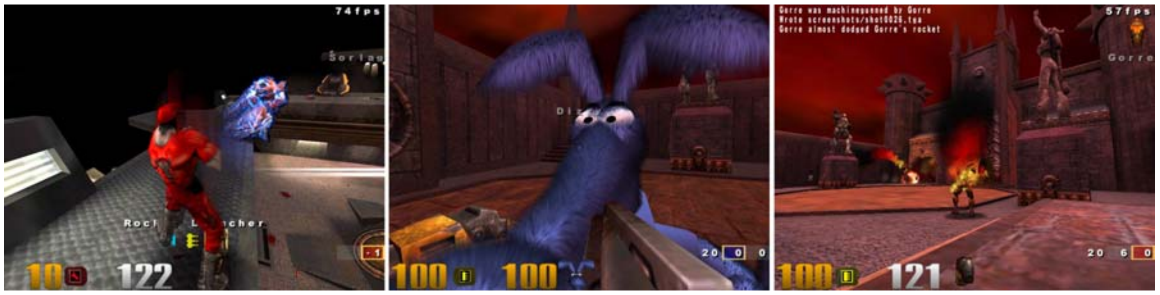

Figure 2: Student hand-ins of 2005. Left to right: Motion blur (using render-to-texture), Fur shader (using a pixel shader), Burning player models (particle effect).

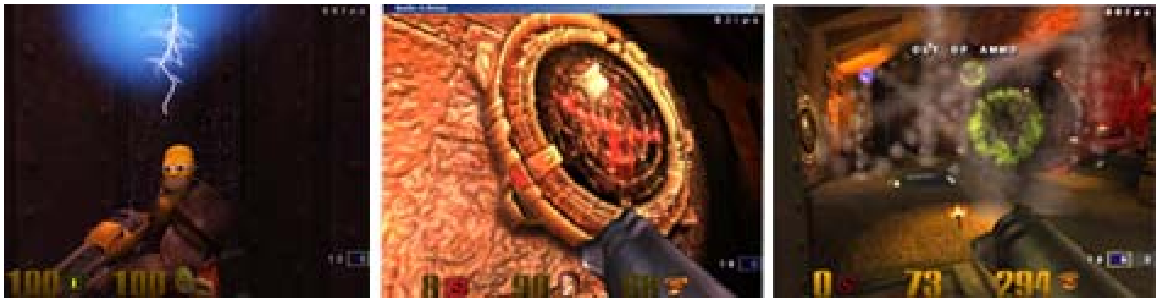

Figure 3: Student hand-ins of 2007. Left to right: Particle Effect (using billboarding), Parallax Mapping shader (using a pixel and a vertex shader), Fireworks Rocked Launcher (particle effect and physics manipulation).

References

Moeller, T.A., Haines, E., Real-time Rendering, A K Peters, ISBN 1568811829, 2002

Eberly, D.H., 3D Game Engine Design. A Practical Approach to Real-Time Computer Graphics, Elsevier LTD, ISBN 0122290631, 2006

Shreiner, D., Woo, M., Neider, J., OpenGL Programming Guide: The Official Guide to Learning OpenGL, Version 2.1, Addison-Wesley Longman, ISBN 0321481003, 2007

Real time Graphics Hall of Fame Website (2007), http://www.icg.tu-graz.ac.at/courses/lv710.079/HallOfFame/2007/index.htm