Teaching, Exploring, Learning - Developing Tutorials for In-Class Teaching and Self-Learning

S. Beckhaus & K.J. Blom

interactive media / virtual environments, University of Hamburg, Germany

Abstract

This paper presents an experience report on a novel approach for a course on intermediate and advanced computer graphics topics. The approach uses Teachlet Tutorials, a combination of traditional seminar type teaching with interactive exploration of the content by the audience, plus development of self-contained tutorials on the topic. Additionally to a presentation, an interactive software tool is developed by the students to assist the audience in learning and exploring the topic's details. This process is guided through set tasks. The resulting course material is developed for two different contexts: a) for classroom presentation and b) as an interactive, self-contained self-learning tutorial. The overall approach results in a more thorough understanding of the topic both for the student teachers as well as for the class participants. In addition to detailing the Teachlet Tutorial approach, this paper presents our experiences implementing the approach in our Advanced Computer Graphics course, and presents the resultant varied projects. Most of the final Teachlet Tutorials were surprisingly good, and we had excellent feedback of the students on approach and course.

Categories and Subject Descriptors (according to ACM CCS): K.3.2 [Computer and Information Science Education]: Computer science education I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism

1. Introduction

The exciting and rapid changing field of computer graphics draws the interest of many students. However, teaching advanced computer graphics topics can be a challenge. Text books for introductory computer graphics are available, but advanced topics outpace books, leaving little choice beyond other sources, such as research papers. Enabling an understanding of the sometimes highly complex mathematical, algorithmical, and geometrical concepts is also challenging, as suitable teaching and illustration methods have to be newly developed. The subject material requires a thorough understanding of diverse fundamental methods and tools. As such, students lacking the requisite background, e.g. students from other majors, may also impose additional challenges. An additional challenge is that the state of the art in the field changes so quickly, the course may have to be prepared from scratch every year.

A quick survey of advanced computer graphics course listings on the web shows that the majority of classes are structured as lecture or seminars Lectures are the classical method, making it the most familiar method. Lectures are not as costly to the educator as using a more interactive and engaging educational style. In seminars, students research existing methods, topics or posed questions and present the results in front of the audience. Seminars may ease some of the load of educators concerning the update of computer graphics lecturing material, as some of the preparation load is taken over by the students. However, this has its own drawbacks, largely in that the quality of the student lectures cannot be assured. Generally, the lecture style of teaching found in lectures and seminars does lead to basic awareness of the taught material, but not to a thorough understanding [Blo56], required to later apply some of the more advanced computer graphics principles.

In this paper we present our method to teach intermediate to advanced computer graphics principles in a combined seminar and project style course. Students develop interactive teaching units, called Teachlet Tutorials. The central component of the Teachlet Tutorial is an interactive software tool written by the student. The software tools are used in

class to interactively instruct their fellow students. The instructional method and all topic information is encapsulated, so that others can use the materials in the future to present the topic in the same manner. Finally, the software tool is also designed to be used as a stand-alone tutorial, usable either to learn or review the topic.

In the next section we present the context and educational background, in which our course and method was developed. Section 3 then introduces the course design and the Teachlet Tutorial approach. Section 4 describes the course goals and requirements and the structure of the overall course. Section 5 introduces the resulting tutorials and diverse approaches students took for their teaching concept and software tool. We then present the results of a course evaluation in Section 6. Finally, we discuss the results and conclude the paper in Sections 7 and 8.

2. Educational Context

In this section, we highlight the educational context and briefly look at the educational background surrounding the developed course. The first subsection holds a short discourse on related learning theories and teaching styles. Then, we describe relevant aspects of our general educational approach and goals in our classes. The final subsection explains the context of the course within the curriculum and the university.

2.1. Educational Background

The process of learning is subdivided into various phases. Simplified from Bloom [Blo56], we can distinguish between a first stage of "having heard of the principle" and the final stage of being able to apply the knowledge to new problems and to evaluate different solutions. Passively sitting in a lecture usually leads to, at best, *awareness* of the knowledge. Through various methods, the student can then move to the next level, having a *basic understanding* of the knowledge. In this stage the student is able to repeat what was taught, solving a problem structured exactly like what they have seen before. The final step is *mastering* the knowledge, in which the student can apply the knowledge in new and unique situations and assess the suitability of methods.

The major drawback of classic lectures and seminars is that content and facts are only presented once or twice in the presentation. Repetition, a necessary requirement for learning from lectures, only takes place in dedicated exercises and homework (if the course style allows for that) or happens when students learn for exams. To this end, many of the course offerings found throughout the web require exercises or larger class projects to be executed by the students.

As an alternative to the lecture style, constructivist approaches to teaching use methods, where knowledge is actively constructed by students [BA98]. The advantage is that knowledge acquired by active participation is much deeper internalized than by merely listening to a talk. This approach is a method, which may enable the students to achieve, at least, the state of basic understanding of the material in an easier way. Literature and studies in other educational area indicate that students receive benefits from such an educational paradigm. Taxén applied this to teach students fundamental computer graphics (CG) knowledge and reported a perceived improvement over lecture style learning [Tax03].

Schmolitzky developed the concepts of Teachlets in a Teachlet Laboratory, primarily to teach design patterns in a software engineering curriculum [Sch05]. Students develop teaching units, called Teachlets, which they present to their fellow students in class. A teaching unit consist of a slide based introductory lecture, a task set for the audience, a software framework or software tool, in which the task is solved, and an interactive, guided, "solution-finding" process jointly with the student audience. During this solution finding process, the audience advises the student teacher, who acts merely as an operator of the software framework. Through this processe, the audience solves the task, guided by the student teacher if necessary. The Teachlet Laboratory successfully combines classic lecture-style teaching of content with application of this content to real problems.

2.2. Our General Educational Approach

In our courses, we pursue several goals, more thoroughly described in [Bec06]. One of our major concerns is how to motivate students and how to create a friendly environment. Students who feel comfortable and relaxed in a learning environment may understand the material at a deeper level and may also be able to present it in a lively, unconventional, and interesting way to their fellow students. Unmotivated students may only learn to fulfill the requirements for a class in order to receive a passing mark.

We found that conducting projects with visible outcome and useful application is highly motivating to students. Motivated students achieve more, learn better, and the outcome of the course is more satisfying both for teachers and students. Highly motivated students may develop new viewpoints, approaches, and tools, and may even contribute to current research questions.

Including projects in the education process is not only motivating, but teaches the necessary skills of project management - vital for pursuing projects but hardly focused on in teaching -, encourages a strong focus of the students on the topic, motivates an understanding of the diversity of users, needs, and expectations, and brings the necessity to validate one's assumptions into focus. Furthermore, by applying the newly acquired knowledge, the topic is much more reflected on and subsequently better internalized.

2.3. University Context

The students in our specific course on advanced computer graphics topics were third to fifth years students, pursuing a five year German Diploma in Computer Science (comparable with MSc level). By their third year, they already possess a solid theoretical base in mathematics, programming principles, and theory. A prerequisite to our course is fundamental computer graphics knowledge, as, for example, taught in a dedicated course CGB (Computergrafik und Bildsynthese, Computer Graphics and Image Synthesis), plus working knowledge of C++. The CGB course focuses primarily on ray-tracing fundamentals, with only rudimentary real-time CG coverage.

At the University of Hamburg, the diploma courses range from dedicated projects to seminars and lectures. Seminars are 2 hour per week courses, conducted mostly in the styles expected. Projects comprise of 6 hour per week work on a dedicated topic. The unusual format, "project-seminars," combines a seminar with project work in a 4 hours per week course. In projects, seminars, and project-seminars, students either actively participate to receive specific project and seminar credits which they need for their studies, or they have the choice to only attend the course to learn for a later combined exam on a larger topic area.

3. Course Design and Teachlet Tutorials

For the rough structure of the course we chose to create a project-seminar, as described in the previous section. This format allows for a combination of taught content (seminar) and applied project work, which we believe is not only a very successful way of teaching and learning, but also most rewarding for students. A commitment of four hours per week time in class, plus working hours outside class, provides sufficient time to both teach and explore a subject and apply the knowledge. However, having sufficient time does not automatically overcome the traditional problem with seminars. Students are typically well motivated to explore the materials in computer graphics, but assuring that the material is well covered in the presentation is difficult.

To address this we looked to the Teachlets method [Sch05], described in Section 2.1, and adopted and adapted the approach for our "Advanced Computer Graphics" (ACG) class. We felt that this approach brought a number of advantages. The foremost is that the students were forced, through the interactive requirements, to really deeply understand the material. Creating a tool to interactively clarify the concept they were teaching also forces the students to approach the materials from different ways than they might have otherwise.

While Schmolitzky uses Teachlets for teaching in a classroom context only, we extended this method to also reuse the produced material for later reference and self-learning. We call this combined method *Teachlet Tutorials*. An additional advantage compared to standard Teachlets is that all relevant background information on the topic itself, necessary for conducting a Teachlet, is included in the tutorial material. No other sources should be necessary to adopt the Teachlet Tutorial for later reuse in class. This extension was inspired by the CGEMS [FEJ03] concept. If suitable results were achieved in our course, they could be submitted to the CGEMS program. We also hoped this would contribute significantly to the students motivation to produce a high quality work.

Five different roles can be identified in this model, where the last is only applicable, if the final Teachlet Tutorial is later reused in subsequent presentations:

- the *instructor* (educator), who introduces the concept of Teachlet Tutorials in class and sets the topics, plus guides the students teachers through their development phase and may add missing details in the presentation, if necessary.
- the *student developer and teacher*, who adopts a topics, researches it, builds an interactive tutorial, presents the fundamentals to the audience in a suitable way, moderates or guides through the interactive, task oriented exploration of the topic, operating the tutorial software, and, finally, prepares the final stand-alone tutorial for later reference and reuse. Student teachers prepare the Teachlet Tutorials and also use them for their presentation.
- the *student audience*, who listens and interactively explores the subject guided by the set tasks and using the tutorial tool.
- the *self-learner*, who either revisits the topic (mainly participants from the audience) or learns the topic from scratch (anyone interested in the topic).
- the *instructor of existing Teachlet Tutorials*, who later adopts existing Teachlet Tutorials in her own course. In this case, the instructor simply uses the pre-prepared material to present the tutorial in a similar way as the student teacher had envisioned. This reuse in class is a further potential of the method.

In the Teachlet Tutorial approach, three separate levels of learning activity can be identified as results:

- the student teachers, those students who conduct specific topics, thoroughly research their topics and reflect both on the topic as well as on potential ways to apply it. They learn to program the principles and apply them in their software tool. They also reflect on good ways of teaching the material.
- the class audience learns from the student teacher about advanced CG topics and has a chance to interactively explore the content. This allows for a deeper understanding of CG principles and potentially results in an immediate, more effective learning effect.
- the class audience, plus any interested student, can later use the materials to either revisit or learn the topic. As the results are presented in a standardized and well designed

form, the CGEMS style tutorial form [FEJ03], the results of the class are potentially available to a wider community. If made public, it can then be used as a self-tutorial or as instructional component on the topic by anybody.

4. Our Course's Structure

The goal of the ACG project-seminar, as announced in the curriculum [Uni06], was to research and implement advanced computer graphics topics, to build educational material suitable for others to learn this topic, and to teach the fellow students with the developed material. Furthermore, this material was planned to be submitted to the CGEMS program, if suitable results were achieved.

The course was structured in the following way. In the first meeting, we introduced the list of topics and the goals for this project-seminar. We had three requirements for students to pass the class. First, we defined that each topic had to be presented in class using an interactive software tool, which illustrates the concepts of the topic. Secondly, we expected a suitable interactive teaching style, making the presentation more lively and, potentially, allowing the exploration of the topic by the student audience. As a final requirement, students were asked to create a version of their tutorial, which incorporates the software tool and the print-style description to form a self-learning unit. This was to be submitted to us in the CGEMS style, including a paper describing objectives, requirements and the tool, as outlined on the CGEMS webpage (`http://cgems.inesc-id.pt/`).

Each of the final Teachlet Tutorials was to consist of the following components:

- a *teaching concept*, including a concept for the presentation, the interactive demonstrations, and the tasks for the audience. This naturally included making the topic accessible, understandable, and easily assimilated,
- the interactive *software tool* for demonstration and exploration,
- the *oral presentation*, potentially completed with slides or a blackboard concept,
- a *full topic report*, describing all principles, methods, and mathematics in print-form, including text and images (mainly useful for self-learners and later reuse),
- a brief overall topic and requirements description (CGEMS *paper*) to accompany the final set of components,

The individual topics were tailored to the students present in our course. Factors contributing to the selection included the number of students, their limited experience with real-time graphics, and our orientation toward virtual reality and real-time issues. Based on these factors, the partly intermediate, partly advanced topics were Lighting and Materials Methods, Vertex and Pixel Shader Techniques, Intersection Testing, Shadows, Binary Space Partitioning, an overview of Acceleration Techniques (Portals etc.), and Collision Detection.

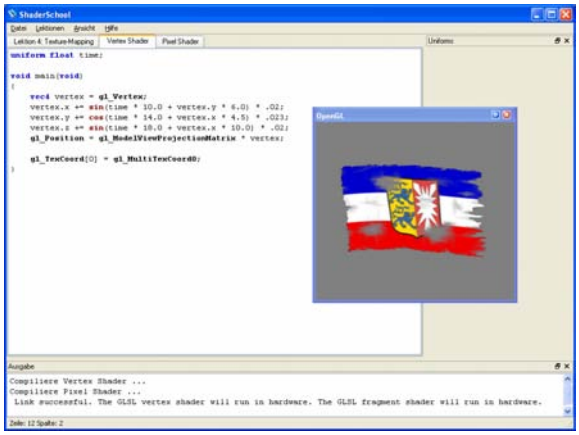

Figure 1: *ShaderSchool - Ragged Flag.*

To further motivate the students, we tried to create a friendly working environment, as we have found that this is a major contributing factor to their level of motivation. To this end, we generally have a major focus on the first meeting, where we encourage creative approaches in the set frame of the class and we try to form a group out of a 'crowd' of individuals. We did this here by setting a small fun task, which had to be cooperatively solved [Bec06].

In a second meeting, the lecturers presented a topic, including methods demonstrating one of the many possible ways of giving an interactive tutorial. After these two initial meetings, the students had several weeks time to prepare their topics and tutorial software tools. In this time, we offered assistance, if requested. Then, the students presented their tutorials, with a small winter-break after the first presentation. Finally, the course was evaluated (described in Section 6), and students handed in their project work.

5. Overview of Resultant Projects

In this section, we briefly describe the Teachlet Tutorials created by the students during the course. We discuss both the in-class presentation style and the tutorial. An overview of the technical aspects of their software is include in the discussion. For the first project, shown in Figure 1, we will present a bit more in depth look than for the others. Due to their individual natures, we have chosen to include portions about each project, as we feel that they may contribute to design ideas of future tutorials. The first of the projects described here was also the first presented in the course. This is of importance, as it greatly influenced the others, which were presented more than a month later due to a mid-semester break.

We have striven to highlight, through the descriptions and

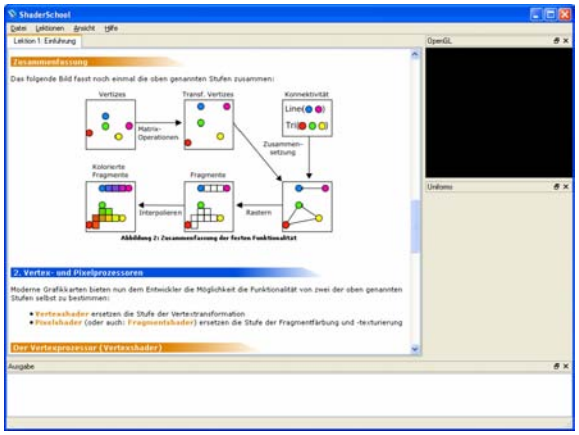

Figure 2: *ShaderSchool - Introduction.*

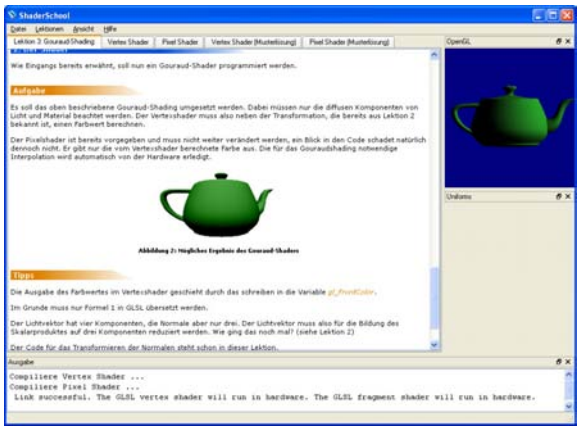

Figure 3: *ShaderSchool Tutorial - Gouraud Shading.*

figures in this chapter, the two different formats of Teachlet Tutorials, each of which is comprised of the interactive tutorial software components and the presentation styles using them. The first format is a software tool that is a self-contained tutorial. It includes both the topic description and the interactive component. We will refer to this as a *tutorial tool*. The second format is a stand-alone software tool, which was accompanied by traditional blackboard or slide presentation. The Teachlet Tutorial is completed by an accompanying print document for traditional tutorial usage. The stand-alone software is here called a *software tool*.

5.1. Project 1: ShaderSchool

The ShaderSchool was the first of the presented Teachlet Tutorials. This project was a combined project for two students, one focused on vertex shaders and the second on pixel shaders. The students worked together as the topics were so tightly inter-related.

The students gave their in-class presentation using primarily their tutorial tool. Figures 2 and 3 show the format of the tool and presentation. The students did have a short introduction to the topic consisting of a few slides. This was largely to introduce OpenGL pipeline concepts that not all students were familiar with. The slide presentation is not necessary when all the students have the appropriate background. The remainder of the presentation was done within the tutorial tool. The tutorial is divided into lessons, where an textual introduction to the new information is followed by a integrated programming assignment over the new materials.

The students added more in-depth information verbally than was present in the tutorial text and presented most of the information outside of diagrams without reading from the text. Interspersed throughout were interactive tutorial pieces. These were performed in the Teachlets style, where the student teachers did little more than typing the solution thought out by the participants of the class. The interactive nature of this lead to a number of obtuse experiments being performed, at the behest of the audience. This interactive exchange also led to new insights, even for the student teachers, on what can be done with shaders and the tutorial tool. For example, following an audience request, a colored flag waving in the wind was coded in a joint effort. The results can be seen in Figure 1.

The ShaderSchool software is built in a QT framework. The main pane of the interface is a tabbed textual area, and is the main working area. Each lesson is composed of three tabbed components. The first tab contains the tutorial text and explanations. The second tab contains the interactive code. The interactive code piece is in every case a shader. The shader is, with a key press or menu selection, compiled and inserted into the active OpenGL context, in a pane in the upper right corner, on the fly. The underlying OpenGL environment is predefined, but also available to look into. In the version presented, this was only available outside of the ShaderSchool software. Additionally, there is an example solution to the code; however, it must be explicitly loaded by the student to prevent accidental access. After completion of a task, the next lesson in the tutorial can be selected, loading new tabs into the text and a new OpenGL environment.

5.2. Project 2: Lighting and Materials Methods

The OpenGL Lighting tutorial followed largely the pattern set by the ShaderSchool. However, the delivery and technical aspects of this project were different and well worth noting. This topic, a classical introductory CG topic, was included to see how effective this format would be for teaching it. We hoped to integrate the end result in our introductory classes.

As with the ShaderSchool, the student presented the information through the tutorial tool. In this case the student used only the interactive tool to present the information, again re-

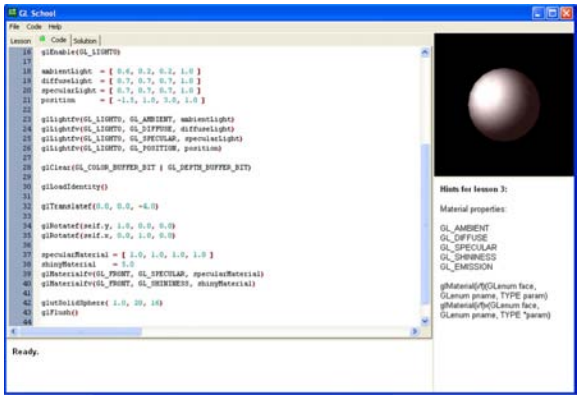

Figure 4: *Lighting and Materials Methods Tutorial.*

lying mostly on the pictures and diagrams and presenting the information verbally to the audience. The presentation tool can be seen in Figure 4. Again, the student also included the interactive stepping through of the tutorial coding with the class. It is interesting to note here, that beyond the first interactive task, the student audience did not immediately come to the correct solutions.

In order to have a system, with the similar interactive nature to the ShaderSchool, the student took a slightly unconventional route. Basing the tool on Python with an OpenGL extension, he managed to create an interactive tool, where code can be modified in one pane and then compile. The compiled code is automatically placed into the view window. The view window supports a limited amount of interaction also, in the form of positional and rotation manipulation.

5.3. Project 3: Acceleration Techniques

The Acceleration Techniques project was on the topic of algorithms and concepts designed to speed up real-time environments. The concepts are commonly used in many modern graphics environments. From the possible list of concepts for acceleration in real-time environments, the student selected to present portal-culling, level-of-detail, and billboards.

The presentation was, once again, delivered through use of the tutorial tool, shown in Figure 5. Although there was no interactive coding portion to this tutorial, the presenting student used a more traditional interactive teaching style. Largely this was based on questions to the audience on how concepts would be manifested in the environment or how changes look. However, the tutorial tool did include portions which helped to explain the concepts. In each case, additional viewports gave views, highlighting the concept visually.

The tutorial tool developed is a departure from the style of the ShaderSchool. It was written in java for delivery as a web-plugin. As the user scrolled down the tutorial text in the

Figure 5: *Acceleration Techniques Tutorial.*

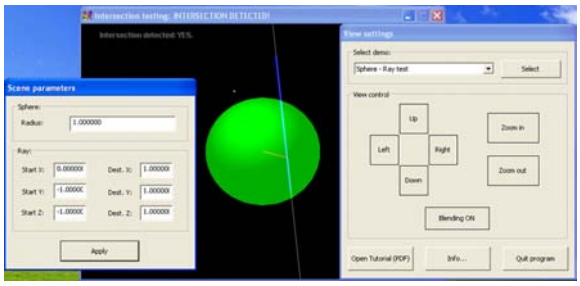

Figure 6: *Intersection Testing Tool.*

left pane, the interactive environment on the right updated to display the corresponding environment. In each case the environment highlighted the discussed algorithm or concept. The user was free to interact with the environment in various ways, including coordinated interaction with the different perspective views of the concept.

5.4. Project 4: Intersection Testing

The Intersection Testing presentation was given with a more traditional approach. The student did a lecture style introduction, using traditional media such as the black board to interactively explain the principles. At the point of introducing each of the individual tests presented (Sphere-Sphere, Sphere-Ray, Sphere-Axis aligned box, Ray-Box, Ray0-Axis aligned box), the student demonstrated the idea with the developed software tool. The tool was interactive, in that the user could move the objects in question for the testing. Included in the visuals were construction objects, which helped to clarify the principles. The construction objects typically reflected steps in the calculations, clarifying visually the mathematically complex steps. Examples of this are coloring of segments of intersection, normals to lines and faces, and explicitly showing separating axes.

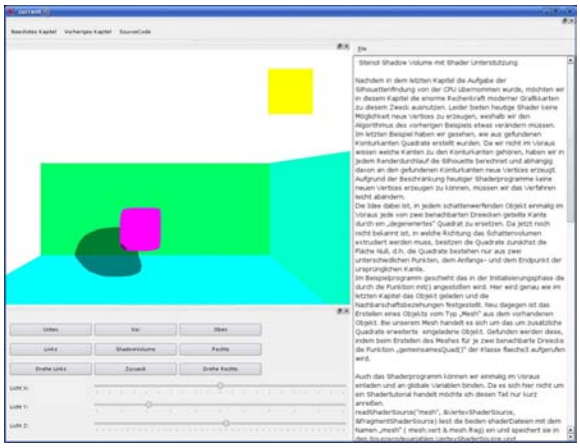

Figure 7: *Shadows Tutorial - gpuStencilShadows.*

5.5. Project 5: Shadows

The presentation of the Shadows topic followed a mixed format. The student began with a slide presentation of the basic topic and explained the main concepts. This was followed by an interactive section using the developed tutorial tool, shown in Figure 7. During this part of the presentation, explanation of the details of implementing shadows was given and was highly interactive with the student audience. The tutorial tool is a QT based application, with a structure similar to the ShaderSchool. In contrast to the Lighting and ShaderSchool tools, the framework does not allow for interactive changes to the code. Instead, the tool helps by visually assisting the understanding. The code segments relevant to the method being highlighted are shown and explained through the text tutorial and the comments in the code. Stand-alone versions of the code in C++ are available for the user of the tutorial to interactively change, without having to deal with the entire QT application and build process.

5.6. Project: Binary Space Partitions

The Binary Space Partitions (BSP) presentation was a free-form lecture, first using the black-board and then presenting the learning tool he created (see Figure 8). This was a good example of a topic where the audience had difficulties following the complex mathematical principles, until the software tool was utilized. The tool is a small java application, which demonstrates the tree construction. Vertices are entered in an application window, forming polygons. When the polygons are entered, they appear in the visual field and a second window displays the resultant BSP. The materials presented in the presentation are available in written form, as a text tutorial to work through.

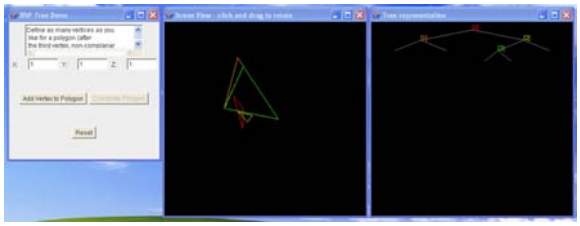

Figure 8: *Binary Space Partitioning Tool.*

5.7. Interactive Tutorial Toolkit Frameworks

Each of the students prepared a unique toolkit, as can be seen throughout this section. This was done, even though the ShaderSchool was offered by the originating students to the others as framework for reuse. With the winter break in-between the ShaderSchool project and the other projects, it gave ample opportunity for the others to use the ShaderSchool framework. When asked, each of the following students, for various reasons, had felt they would achieve more developing either own software. In total, three of the tutorial tools were created as generic frameworks, designed to be further used:

- The ShaderSchool can be easily expanded for shader lessons. This is accomplished simply by placing appropriate files into a specified directory. These files, through naming convention, become the text, examples, and example solutions. The one portion that requires compilation is for new OpenGL basics.
- The Shadows project created a similar reusable and extendable framework, also based on QT. In this framework, the different textual files are also placed in a lesson directory and automatically loaded.
- The Lighting and Materials tool is written in Python, with an external library to allow OpenGL calls. The interesting part of this tool is, of course, that the code can be recompiled "on the fly" in a quasi-interactive manner. This adds flexibility to the inclusion of new lessons not available in the QT-based frameworks. The framework is similarly extensible via text files, which make up the individual lessons components.

6. Course Evaluation

The course concept was evaluated through an initial oral feedback from the students for the instructors, plus a written three page evaluation form filled out by the students. Questions were asked concerning content, approach, results, motivation, support, and general comments. Questions were rated on a 5 point Linkert scale (1: not at all/very bad, 5: very much/very good). The results to the questions listed below are shown in Table 1. The scores reported here are for the seven students who attended the class and created Teachlet Tutorials. The results showed that the course was perceived

Question number	Median	Deviation
q1	5	0.73
q2	5	0.49
q3	5	0.41
q4	5	0.61
q5	5	0.65
q6	5	0.41
q7	5	0.41
q8	4	0.73
q9	5	0.65
q10	5	0.24
q11	2	1.35
q12	3	0.49
q13	4	0.61
q14	5	0.49

Table 1: *Evaluation results to the questions described in the text.*

as being of high personal interest (q1), challenging (q2), that the atmosphere was very good (q3), that the course was fun (q4), and that the students felt they had learnt something useful (q5).

The tutorial style approach in the course was rated very good (q6). Nobody preferred to have standard slide talks or disliked the interactive form of presentation. Students estimated that, compared to classic style courses, they have understood the topic better (q7), they assimilated the material better (q8), and they are more able to apply learned materials (q9). All students felt that, for their own tutorial, they understood their topic much better than in a traditional education setting (q10).

The responses to the questions about the presentations were more mixed. Three students felt that, without the tutorials, they would have been able to present more content, while five students did not have this feeling (q11). Students indicated that they learnt more about how to better give a talk in general (q12) and how to improve talks by including interactive components (q13).

The verbal feedback affirmed most of what was in the formal evaluation. One of the main oral and free form comments were concerning the good atmosphere in the course. All but one students also liked the exact choice of topics. It was justly noted that the order of topics, which unfortunately had to be rearranged during the course, could be improved. The overall rating of the course was very good (q14). One student even claimed that this was the first course, he really looked forward to attend to every week and felt sad, when one presentation had to be canceled.

7. Discussion

The resultant projects demonstrate that the developed tutorial and software tools required for computer graphics are more technically complex than those described in the original Teachlet paper. We had been concerned that the students may be overwhelmed by the technical aspects. The interactive tutorial part requires not only the development of a suitable task and its preparation in a piece of software, but requires caring for OpenGL environment, shader compilation, and such. Furthermore, we required an additional focus on the reusability of the final tutorial both for self-learners and future teachers. Nevertheless, this additional load did not keep the majority of students from finalizing the software tools and completing the tutorials. All but one student, who well presented his topic, but did not develop a tool due to time constraints, successfully finished their projects and handed in their final tutorial.

It should be noted that for Teachlet Tutorial projects like the ShaderSchool, the Lightning and Materials Method Tutorial, and the Acceleration Techniques Tutorial, no change of the tutorial from presentation mode to self-learning mode was necessary, as the tutorial already included all necessary information for a full topic report. Only accompanying instructions on how to use the tutorials were required. All other projects needed to adapt their oral presentation and presentation material to form a full topic report.

We believe that the successful outcome of the projects and the course as a whole was based on the students motivation and has several causes beyond the task we set and the generally high interest in computer graphics topics:

- The task itself was set with a clear goal, but with space for the students to develop their own ideas. The task included the list of requirements, described in Section 3, and a final date for the presentation. Beyond that, we left a wide space for students to develop their own ideas and explicitly encouraged creative and novel approaches. We have found this to be very motivating in previous courses of various styles.
- The overall task was not obviously easy to accomplish and had several layers to it, which may have presented new challenges for the students. The students seem to have enjoyed taking up the challenge.
- The outcome of the project was immediately used by the student teacher in class for the benefit of the audience. Immediate acknowledgment and feedback followed from the presentation. This reinforced the point that the project had been a useful and needed project, which left students feeling the work had been worth the effort.
- The project may be useful to a wider audience, due to the standardized outcome. This could be in the CGEMS context, if appropriate and submitted, or for internal university use.
- Having a finished, published project is an impressive line in students' CVs.

- Last, but not least, the atmosphere in the course was excellent. Students felt comfortable in this working environment, which in turn lifted some of the pressures of presenting in front of others, encouraged experimenting with teaching concepts, and enabled a lively, friendly discussion and exploration of the software tools among the students. Encouraging this atmosphere needs specific focus by the instructors. This takes time and effort to establish, but is greatly rewarding, as it strongly influences the motivation and commitment of the participants in a class.

The evaluation presented in the previous section included self report impressions of the students to their perceived educational gains from the course. In a curriculum where it is appropriate, testing could naturally be included at the end of the semester to better ascertain what the students learned and if they retain the information better than with another instructional method. As the topics in this course were taught by us for the first time, we can make no comparison to other teaching styles at this point.

A teaching concept strongly depends on the number of students in the audience. Our Teachlet Tutorial approach is most useful with a limited class size. A large factor is that each student, who wants to get credit for the course, must develop and present a Teachlet Tutorial and that the number of slots available is limited by the number of times the class meets. This limits the number of students who can take part in all aspects of the course, but not how many can attend the presentations. We had 8-12 students in the audience, but believe that similar results can be achieved with a larger audience. Direction and questions addressed to the student teacher, who operates the tool, works in the same way as in small group. Thus, the interactive exploration of the principles with the tool should be possible in much larger groups. However, creating the friendly, motivating atmosphere we had in this course in settings with hundreds of students may pose a problem and make it unattractive for students to take an active role in the course. Therefore, for initial development of the Teachlet Tutorials, we recommend small groups, with up to 20 people.

8. Summary and Conclusion

We have presented an approach to a computer graphics course, where students develop interactive tutorials, Teachlet Tutorials, for computer graphics topics. These Teachlet Tutorials can be used for both in-class presentation as well as for stand-alone reference and self-learning units. To accomplish this, students have to: learn about the topic, develop a teaching concept involving the interactive participation of the class, develop a suitable software tool to assist the interactive presentation and exploration of their concept, and assure that the software tool could stand as a complete stand-alone tutorial.

Several different approaches to developing and conducting Teachlet Tutorials were presented. Three of them included the development of generic frameworks which allow easy extension of the tutorials and reuse for other topics. Methods were found, for example, to interactively recompile shader code, to use Python to quasi-interactively recompile OpenGL code, or to connect the position in a web-based tutorial text with the perspective three-dimensional interactive presentation of the principles.

The students were highly motivated. Motivation is naturally high due to the subject, but in class presentation delivery is often sub-interesting, as it is much more fun to learn on the topic than to present it. To fulfill the minimal course requirements, simple software tools would have been sufficient, adding just one interactive component to the presentation. Nevertheless, most students chose to create extensive software tools, demonstrating as much as possible for the subject. Half of the students went even further, creating reusable frameworks.

The students in our course produced excellent results. Students, as well as teachers, gained a lot from this combination of teaching, exploring, and learning a topic in the class room context. The course was always highly instructive, enjoyable, and pleasant to attend.

The high quality of the Teachlet Tutorial concepts, the resulting software tools, and the final self-learning tutorials were mostly unexpected by us. We believe that a significant portion of that success was due to the additional motivational factors beyond the topic itself. These motivational factors included: the potential reward of developing and presenting a new tool, the motivational atmosphere, the feedback of their fellow students, the learning success of the students, the possibility of submitting their project to CGEMS, and having a presentable project, which looks good on their CVs. This is much more motivating than normally in seminar style classes, where topics are presented once and the seminar paper, summarizing the talk, is read by hardly anybody beyond the instructors.

We expect that subsequent courses using Teachlet Tutorials will be successful and that the method will work with varied topics. The developed frameworks can be reused by other students or instructors in later projects, easing the programming load of future developers. Also, we expect to have at least some of them to be submitted to the CGEMS project.

Acknowledgements

This paper would not have been possible without the great commitment of our motivated and motivating students: Christian Alfonso (BSPs), Bastian Boltze (Acceleration Techniques), Jan-Tajo Fittkau, Matthias Grandis (Lighting and Materials), Björn Kühl (Shadows), Greg Paperin (Intersection Testing), Ulf Reimers (ShaderSchool), and Malte Thiessen (ShaderSchool), who designed and developed the diverse tutorials.

References

[BA98] BEN-ARI M.: Constructivism in computer science education. In *SIGCSE '98: Proceedings of the twenty-ninth SIGCSE technical symposium on Computer science education* (New York, NY, USA, 1998), ACM Press, pp. 257–261.

[Bec06] BECKHAUS S.: Seven factors to foster creativity in university HCI projects. In *Inventivity: Teaching theory, design and innovation in HCI, Proceedings of HCIEd.2006-1* (Limerick, Ireland, 2006), BCS/IFIP WG13.1/ICS/EU CONVIVIO, pp. 91–95.

[Blo56] BLOOM B.: *Taxonomy of Educational Objectives: Handbook I: Cognitive Domain*. Longmans, Green and Company, 1956.

[FEJ03] FIGUEIREDO F. C., EBER D. E., JORGE J. A.: CGEMS: Computer graphics educational materials server. In *GRAPH '03: Educators program from the 30th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2003), ACM Press, pp. 1–7.

[Sch05] SCHMOLITZKY A.: A laboratory for teaching object-oriented language and design concepts with teachlets. In *OOPSLA '05: Companion to the 20th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications* (New York, NY, USA, 2005), ACM Press, pp. 332–337.

[Tax03] TAXÉN G.: Teaching computer graphics constructively. In *GRAPH '03: Educators program from the 30th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2003), ACM Press, pp. 1–4.

[Uni06] UNIVERSITY OF HAMBURG: Curriculum WS 2005/2006: Advanced computer graphics course. http://imve.informatik.uni-hamburg.de/WS0506ACG.htm, last accessed: 22th of June, 2006.