The Rendering Pipeline in the Classroom: A Diversified Approach

Kostas Karpouzis
National Technical University of Athens
Department of Electrical and Computer Engineering
Heroon Polytechneiou 9
157 73 Zographou, Athens, Greece
+301 7722488
kkarpou@softlab.ntua.gr

Stefanos Kollias
National Technical University of Athens
Department of Electrical and Computer Engineering
Heroon Polytechneiou 9
157 73 Zographou, Athens, Greece
+301 7722488
stefanos@softlab.ntua.gr

1. ABSTRACT

In this paper we describe an integrated method of teaching an introductory computer graphics course. Most such courses are simply "art-oriented", that is they focus on getting students to use modern commercial software, so as to prepare them for a corresponding career, or concentrate on the basic concepts of graphics theory and merely provide a theoretical foundation, such as simple translations and projections; in this case, they usually fail to motivate the class by producing practical interesting examples. The curriculum that we propose combines theoretical knowledge of introductory computer graphics concepts and techniques with laboratory work in programming or modelling and animation exercises. This set of applied laboratory exercises is relevant to the material taught in class, but also extends to familiarising students with the modern uses of computer generated imagery, such as films, virtual worlds or medical imaging. The feedback from the students, combined with their success in the course, shows that this coupled teaching and immersion material is by far more interesting and challenging, while still providing them with the essential academic background.

1.1 Keywords

Computer graphics education, laboratory applets

2. FRAMEWORK OF THE COURSE

The concepts presented in the textbook and laboratory work correspond to each one of the separate processes known as the rendering pipeline for z-buffer and Phong shading [3]. Using a flow of discrete and self contained processes, such as those that form the pipeline (see Figure 1) guarantees that each of them can be implemented - and thus taught - on its own in a real or hypothetical rendering environment.

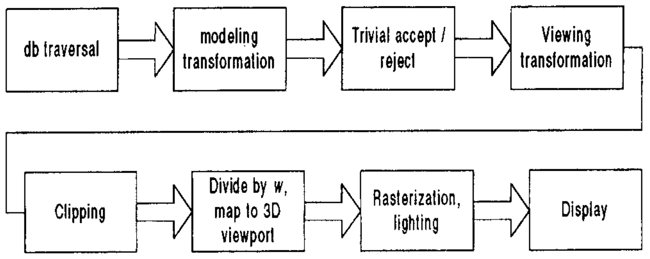

Figure 1: The rendering pipeline

The first topic covered in the textbook is vector graphics : transformations in two or three dimensions, including orthographic and perspective projections, and clipping. Even a simple transformation matrix is useful in understanding concepts that are important to a graphics programmer or user, such as screen and world space co-ordinates and transformations between different co-ordinate systems. Consider a scene with many objects : each is defined in its own local co-ordinate system, typically by a set of data structures [8] that describe points or vertices, edges and faces. When these objects interact, there is a need for a common definition scheme, so as to facilitate required transformations : this scheme utilises a world co-ordinate system and a well defined one-to-one mapping between this and the local systems. Understanding of this mapping is essential for a programmer or designer so as to accurately define the proportions of the objects in the scene. Besides that, using a common matrix representation for each of the transformations assists in implementing them in a straightforward manner.

Implementation issues regarding the transformation matrices are rather trivial for an introductory course and, on top of that, most graphics cards are satisfactory in accelerating such tasks. A special, but very usual, kind of transformation is the perspective projection : even though viewing a scene is usually taken for granted, optimising this procedure is crucial for minimising rendering times and getting unusual effects, such as panoramic viewing [2]. A related issue is that of clipping and z-buffering : the number of multiplication's needed to render an object is at least linear with respect to the number of the polygons of the object; sorting these polygons in 3D space and discarding those that are hidden or outside the view volume, reduces that amount, thereby reducing rendering times.

The next part of the textbook covers some of the most common modelling techniques : starting from the simple but effective constructive solid geometry (CSG) and the popular boundary representation (b-reps) [4, 9], students are exposed to ideas on how to model real-world objects, considering any trade-off issues between quality and size of the implementation. Such a trade-off is introduced because CSG and boundary representation techniques can only approximate detailed curved surfaces, such as those found on an organic object. This is a result of the utilisation of small polygonal facets, instead of actual curved surfaces. Despite this, these methods of 3D object representation can yield sufficient results, especially with the use of textures (see Figure 2). Since the course is targeted towards real-time applications, complex modelling methods such as spline modelling [1], are presented briefly.

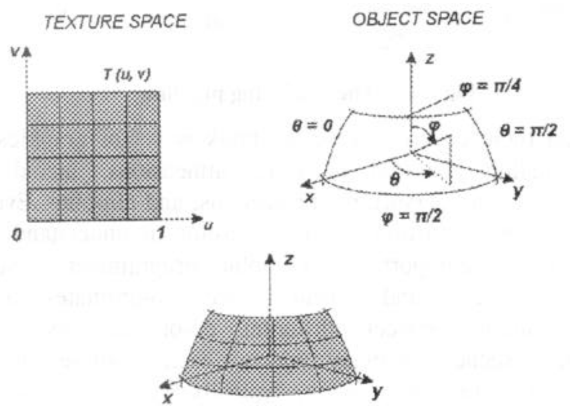

Figure 2: Texture mapping

The final chapters of the textbook concentrate on texture mapping and ray tracing [3]. When creating either a photorealistic scene or a simple virtual world, texturing can greatly enhance the final result; this is achieved because the mapping of images onto the surface of an object can help conceal any modelling shortcomings and recreate the effect of non-existent geometry. Consider a scene that consists of a model of a house : one has the choice of explicitly modelling the geometry of each window, or texture mapping photographs of real windows onto to the house model. In most cases, while the viewer maintains ample distance from the model, the latter method will be sufficient in terms of realism; on top of that, one has managed to keep the polygon count of the model quite low and is thus able to provide the viewer with real time feedback.

Ray tracing is a very popular, but also time consuming, method of rendering scenes into images : since this is an introductory computer graphics course, implementation issues are not included in the textbook. Instead, students are presented with the algorithms used in a fundamental ray tracing scheme, as well as a handful of ideas on how to extend it. These ideas include the incorporation of coloured light, instead of just monochromatic, or pre-processing with bounding volumes in order to speed up the ray tracing process. Besides resulting in attractive, photorealistic images, ray tracing algorithms can familiarise a student with navigating through and viewing 3D scenes, without having to take into account issues related to modelling or programming.

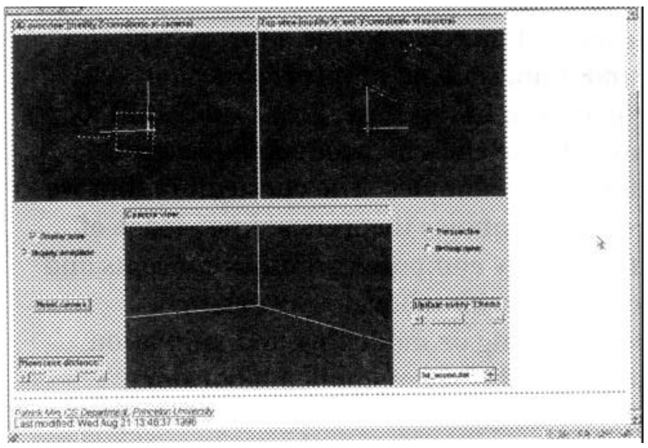

Figure 3: A Java applet that implements the projection transformation

Using printed material in order to visualise graphics usually has its limits, as such material is rather static and not interactive at all. Hence, the lecturer utilises the Java applets, written by Patrick Min for Princeton University [5]. These applets (see Figure 3) help illustrate specific issues related to the textbook content, such as the significance of the order in which different transforms are applied to a vector drawing, as well as more generic notions, such as 3D viewing and 2D line clipping. The most common use of Java applets is the review of their result by the students, while the lecturer interacts with them. However, Java code is structured and easy to understand, as a result of its being object oriented and cross platform. This means that it is organised in several distinct packages, which are reusable and do not contain platform-specific instructions. Furthermore, one can download both the source and the interpreted code and use them in a common web browser, such as Netscape Communicator or Internet Explorer. Also, if a student feels confident enough to experiment with the source code and create his own applets, the Internet can provide him with a rich selection of free, easy to use Java development tools.

3. LABORATORY EXERCISES

The other parts of the original rendering pipeline (lighting, texture mapping and rendering) are covered during the laboratory exercises. Moving from theory to practice motivates students, because they can actually review the significance of each process and see the results of changing different parameters without any programming effort. Besides that, computer art involves not only scenes with static objects, but also objects with changing shapes, appearance and position. This is collectively known as animation and is what actually brought the public's attention to computer graphics. Students are taught how to use real or painted images as textures on top of objects, in order to achieve realism and how to use lighting to supply the scene with the appropriate atmosphere.

Even though this material is starting to move away from a strictly academic curriculum, it provides the lecturer with the foundation needed to discuss optimising the texture mapping scheme or the different lighting models. Course material consists of a set of tutorials and exercises, which includes scenes and objects that students are asked to replicate; during this effort, they master the different techniques that they read about in the textbook and experience the optimisation issues discussed. This drill can be realised with the use of either commercial software (Newtek Lightwave 3D) or the coding of stand-alone applications, based on the OpenGL or HOOPS [6] libraries, by the students. This choice is offered because not all the students that take the course are CS majors; asking from non-CS majors to create a rendering environment, even with the simplicity that these libraries offer, would necessitate tackling both the graphics and the rendering part, as well as coding and creating the user interface.

During the first sessions of the laboratory exercises, the class concentrate on modelling and transforming simple geometrical objects. Getting to know the different tools of the software and the relevant methods of the software library, makes the students feel confident to delve into advanced notions, such as spline modelling and texture mapping. At the end of the semester, most students have come to master the content of the textbook, mainly because of the practical results that they experienced during these exercises.

The part of the exercises that regards animation consists mainly on teaching the use of "bones" to move human-like object hierarchies and inverse kinematics (IK) procedures [6].

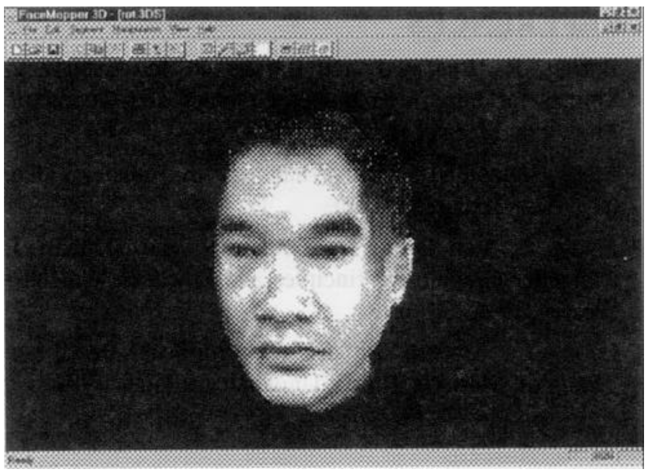

Figure 4: An example 3D application

The notion of bones originates in kinematic physics : they are used to calculate the transformation of rigid objects that are parts of a hierarchy. A jointed model, such as that of a human body, is a collection of different objects that follow certain rules and restrictions. Hence, knowing the path that parent objects follow, the use of bones that connect the joints can help animating the remaining parts as well.

Rendering a realistic animation of a scene which includes organic objects is a challenging task and can only be taught through paradigm. Although creating an animation application from scratch is possible, commercial software (Kinetix 3D Studio MAX) is employed again, so as to let students focus on modelling and animating and not on coding issues.

4. CONCLUSIONS

We present a practical approach to teaching an introductory computer graphics course to undergraduate students, both CS and non-CS majors. The textbook and laboratory exercises are based on the concept of the rendering pipeline, a set of sequential processes that implement modelling, transforming and rendering scenes with 3D objects and effects. The combination of printed material and laboratory exercises provides students with both the theoretical background and the applied prowess to continue research on related areas or, alternatively, follow a career in producing scientific or entertainment material. Feedback from the students shows that this kind of diversity is both challenging and pedagogical. In addition to that, the majority of the students that took the course achieved high grades at the end of the semester, in both the written exam and the evaluation of their laboratory work, indicating that this combination succeeded in introducing them to several computer graphics issues.

5. REFERENCES

[1] Bartles, R., Beatty, J. and Barsky, R. Introduction to Splines for Use in Computer Graphics and Geometric Modeling, Morgan Kaufmann, 1987

[2] Black Diamond Inc. What is Surround Video, http://www.bdiamond.com

[3] Foley, J., Van Dam, A., Feiner, S. and Hughes, J. Computer Graphics : Principles and Practice, Addison - Wesley, 1990

[4] Loop, C. Smooth Subdivision Surfaces Based on Triangles, Master's Thesis, University of Utah, 1987

[5] Min, P. Computer Graphics Applets, http://www.cs.princeton.edu/~min/cs426/applets.html

[6] Techsoft Inc., HOOPS Technical Overview, http://www.hoops3d.com

[7] Thalmann, N. and Thalmann, D., eds. Interactive Computer Animation, Prentice Hall, 1996, 40 - 70

[8] Vince, J. 3-D Computer Animation, Addison - Wesley, 1992, 81 - 84

[9] Vince, J. 3-D Computer Animation, Addison - Wesley, 1992, 45 – 81