Available online at www.sciencedirect.com

ScienceDirect

Computers & Graphics 32 (2008) 363–370

COMPUTERS & GRAPHICS

www.elsevier.com/locate/cag

Education

Teaching 3D modelling and visualization using VTK

Paulo Dias, Joaquim Madeira*, Beatriz Sousa Santos

Department of Electronics, Telecommunications and Informatics/IEETA, University of Aveiro, Campus Universitário de Santiago, P-3810-193 Aveiro, Portugal

Received 19 April 2007; received in revised form 31 December 2007; accepted 16 January 2008

Abstract

In the last two years, we have been using the *Visualization Toolkit* (VTK) as a tool for teaching "*3D Modelling and Visualization*", an elective course offered to Computer Engineering students.

Students start by using OpenGL and, afterwards, use VTK in half of their lab classes, in order to accomplish some tasks and acquire knowledge on its features and functionalities. They are also required to develop a visualization application based on VTK.

We first present the motivation for using VTK and the main features of the "*3D Modelling and Visualization*" course. Afterwards, we describe some of the most successful projects developed by our students. Then, we globally analyse the effectiveness of using VTK, and present the results of a questionnaire handed out to the students who attended the course in the last semester.

Keywords: Computer graphics education; Visualization Toolkit; VTK

1. Introduction

In the last few years we have been witnessing a discussion on how to better teach Computer Graphics (CG) to students in different areas [1–7]. In the past, a bottom-up approach was normally used, where students had to build all necessary code (almost) from scratch. Later, many educators switched to a top-down approach, based on using a higher-level API such as OpenGL [8] or Java 3D [9], with less relevance being given to raster-level algorithms. Others developed CG courses using VRML and with some emphasis on virtual reality (VR) [10]. More recently, courses designed to take advantage of available GPUs and shading languages have also started to be offered [11,12].

Although classical CG textbooks, based on the traditional bottom-up approach (e.g., [13]), do remain useful, advances in hardware, graphics libraries and more recent API-based CG textbooks [8,14–16] offer both students and educators the possibility of exploring advanced concepts and developing useful course projects, e.g., for data visualization.

Two years ago, when planning our courses in the CG area for the first semester of 2005/2006, and given the interest shown by prospective students, we decided to offer:

- "*3D Modelling and Visualization*" (3DMV), an elective course for students already having some CG background, presenting more advanced concepts, as well as offering the possibility of working with *de facto* CG and scientific visualization standard libraries.
- Specialization courses in CG, visualization and geometric modelling, for M.Sc. students, which had to be accompanied by an integrated "*CG Laboratory*" course of 4 h per week, providing them with additional hands-on experience. Given the audience, M.Sc. students that are supposed to be more independent than graduation students, those lab classes were organized as to introduce a series of CG tools and libraries, progressing from the SVG [17] and VRML [18] languages to OpenGL [19], and then to VTK [16].

Since we consider the top-down approach as more adequate for our students, we decided to combine OpenGL with the well-known *Visualization Toolkit* (VTK). Given that we would be teaching students already possessing

*Corresponding author. Tel.: +351 234 370 504; fax: +351 234 381 128.
E-mail addresses: paulo.dias@ua.pt (P. Dias), jmadeira@ua.pt (J. Madeira), bss@ua.pt (B. Sousa Santos).

0097-8493/$ - see front matter © 2008 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cag.2008.01.005

some basic CG knowledge and, also important, having some object-oriented programming experience, we assigned part of the lab classes to VTK and required students to develop a visualization application based on that toolkit, as final assignment. In this way, they would have to use a higher-level API, in addition to the more traditional OpenGL, and would have to acquire some knowledge on data visualization, one of the most important application areas of CG.

OpenGL [19] is the current *de facto* CG educational standard: it is open-source, with a low-level entry barrier and appropriate documentation, as well as offering multi-platform support. VTK [16] is also an open-source, freely available toolkit not only for 3D CG, but also for image processing and data visualization. It offers a higher-level of abstraction than other rendering libraries, such as OpenGL, making it much easier for the knowledgeable user to create graphics and visualization applications. In addition, it also offers a wide variety of data visualization algorithms (including scalar, vector, tensor and volumetric methods), as well as advanced modelling techniques (e.g., implicit modelling, polygon reduction, mesh smoothing and contouring).

Other CG tools were also considered as possible choices: in spite of their relative ease of coding, JOGL [20], the Java wrapper library for OpenGL, and Java 3D [21], the scene graph-based 3D API for Java, were ruled out since our students have no Java background; Microsoft's Direct3D, part of the DirectX API [22], has a higher-level entry barrier and does not offer multi-platform support. We also did not want to limit our students to the use of a modelling language such as VRML or X3D; instead we deemed important for them to design and develop software applications.

The first results of using VTK as a tool for teaching and applying CG, during the first semester of 2005/2006, were presented earlier at the *EUROGRAPHICS 2006 Education Programme* [23]. In this paper we report on the experience of teaching the 3DMV course during the last two years.

In what follows, we first present the main features of the 3DMV course, and then describe some of the most successful projects developed by our students. Afterwards, we globally analyse the effectiveness of using VTK, and present the results of the questionnaire handed out to the students who attended the course in the last semester. Finally, we present some conclusions.

2. 3D modelling and visualization

The 3DMV course was introduced in the first semester of 2005/2006 as a fifth year elective for Computer Engineering students, corresponding to 2 h of lectures and 2 h of lab classes per week.

The 3DMV course was offered as an answer to the strong interest shown by prospective students; moreover, we believe that students benefit from additional exposure to advanced topics in the CG area. Also, medical imaging and data visualization are long-established research areas in our department, with relevance in many graduation and post-graduation projects and/or R&D activities.

Students attending the 3DMV course mostly have an introductory background on the fundamental CG concepts and some experience using VRML, obtained in their third year *Human–Computer Interaction* course, but usually have no experience in using any CG API. Therefore, the main topics addressed throughout the course are:

(1) Review of CG fundamentals.
(2) Introduction to OpenGL (lab).
(3) Geometric modelling (polygonal meshes and free-form curves and surfaces).
(4) Techniques conducing to higher realism (ray-tracing, radiosity, textures).
(5) Introduction to volume visualization (surface extraction and direct volume rendering).
(6) Introduction to VTK (lab).

Regarding lab classes, the first half of the semester is dedicated to OpenGL and the second half to VTK. OpenGL is used to illustrate CG and geometric modelling concepts addressed during the lectures, and to provide the students with their first hands-on experience using a CG API, which facilitates the later transition to VTK.

The fundamentals of VTK are introduced during lab classes and consolidated using a sequence of practical exercises developed for each class. The six lab classes used to introduce VTK encompass the following topics:

(1) *First examples*, *interactors*, *cameras and lighting*: Compiler configuration; visualization of a simple cone; using interactors and different interaction techniques; using several cameras and light sources.
(2) *Actor properties*, *multiple actors and renderers*, *transformations*, *shading and textures*: Modifying actor properties (colour, opacity, etc.); managing multiple actors and multiple renderers in the same window; using transformations to change location and orientation; applying different shading techniques and textures.
(3) *Observers and callbacks*, *glyphing and picking*: Using callbacks and managing events; application examples of glyphing, picking and coordinate visualization.
(4) *Widgets*, *implicit functions*, *contouring and probing*: Using widgets; definition and visualization of quadrics using contouring; probing a quadric with a plane and visualizing the resulting iso-lines.
(5) *Viewing of 2D images*, *visualization and clipping of polygonal data*: Manipulation and viewing of 2D images; importing VRML polygonal data and simple manipulation of the resulting polydata objects.
(6) *Visualization of non-structured grids and volumetric data*: Creation and manipulation of a simple non-structured grid; association of scalar information to the data; visualization and reslicing of medical data.