

Teaching 3D Modeling and Visualization using the Visualization Toolkit

Paulo Dias, Joaquim Madeira, Beatriz Sousa Santos

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

Abstract

In the last two years, we have been using the *Visualization Toolkit* (VTK) as a tool for teaching “*3D Modeling 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 Modeling and Visualization*” course. Afterwards, we describe some of the most successful projects developed by our students. Then, we globally analyze 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.

Key words: 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

Email addresses: paulo.dias@ua.pt (Paulo Dias), jmadeira@ua.pt (Joaquim Madeira), bss@ua.pt (Beatriz Sousa Santos).

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 [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, graphical 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 Computer Graphics area for the first semester of 2005/2006, and given the interest shown by prospective students, we decided to offer:

- “*3D Modeling and Visualization*”, 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 Modeling, for M.Sc. students, which had to be accompanied by an integrated “*CG Laboratory*” course of 4 hours 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 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 computer graphics, 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, vec-

tor, tensor and volumetric methods), as well as advanced modeling techniques (e.g., implicit modeling, 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 modeling 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 “*3D Modeling and Visualization*” course during the last two years.

In what follows, we first present the main features of the “*3D Modeling and Visualization*” course, and then describe some of the most successful projects developed by our students. Afterwards, we globally analyze 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 Modeling and Visualization

The “*3D Modeling and Visualization*” course (3DMV) was introduced in the first semester of 2005/2006 as a 5th year elective for Computer Engineering students, corresponding to 2 hours of lectures and 2 hours 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 Computer Graphics 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 had 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 Computer Graphics fundamentals.
- (2) Introduction to OpenGL (Lab)
- (3) Geometric Modeling (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 was dedicated to OpenGL and the second half to VTK. OpenGL was used to illustrate CG and Geometric Modeling 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 were introduced during lab classes and consolidated using a sequence of practical exercises developed for each class. The six lab classes used to introduce VTK encompassed 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) **Visualization of 2D images, visualization and clipping of polygonal data:** manipulation and visualization 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.

In addition to the work carried out during their lab classes, each group of two students was required to develop a visualization application, corresponding to approximately 20 hours of after-class work. Most of the projects implied

that students had to visualize data from different sources using VTK, and create tools and widgets using the library to provide additional visualization capabilities (such as slicing or probing).

Several data sets have been proposed to our students, ranging from electromagnetic radiation data from antennas, to medical data (brain or lung imaging) and data from physical processes (e.g., water flow around a ship's hull or temperature values within an industrial oven). These data sets were obtained through contacts with other university departments, in order to detect data visualization needs.

The main idea is to give the students data sets to visualize, which are important to final users, and thus increase their motivation. For some of the applications and data sets, visualization tools already existed, but did not seem to completely satisfy their final users. In fact, our colleagues showed great interest when interviewed on the issue, since they considered the possibility of influencing the design and directly participating in the specification of the application's features as an advantage, instead of being limited to the use of existing commercial application software with limited possibilities.

3 Examples of Visualization Projects

We will now present some of the most successful visualization applications developed by students attending the 3DMV course in 2005/2006 and 2006/2007. The examples described encompass the application areas of medical imaging, physical simulation, acoustic data visualization, reverse engineering and virtual reality.

A — Visualization of brain data from different modalities

One of the most interesting projects consisted in developing an application to visualize, in an integrated way, data from different brain imaging and signal modalities, namely, Magnetic Resonance Imaging (MRI) and Single Photon Emission Computed Tomography (SPECT) data, Electroencephalogram (EEG) and electrical dipole data. The latter two modalities provide time-varying data sets.

The work was divided into three sub-tasks, each one allocated to two students. Given the common platform (VTK), each group implemented the visualization of a different type of data. The entire work was integrated into a single application, which allows the user to easily switch among different data and visualization methods.

Volumetric MRI and SPECT data

One group of students had to represent, in the same window, previously registered MRI and SPECT data. This simultaneous visualization provides physicians with important information concerning the location and intensity of brain activity.

A surface extracted from MRI data is presented, as well as red polygonal models representing SPECT data. The interface allows the user to activate and interactively locate up to three cutting-planes (axial, coronal and sagittal) to visualize cross-sections of the registered data (Figure 1).

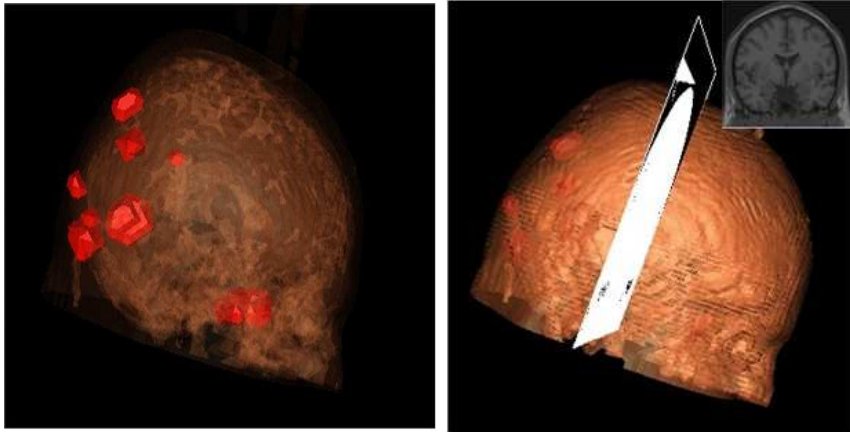


Fig. 1. MRI and SPECT data with different opacity values (left), and coronal plane with corresponding slice (right).

EEG data

A second group of students was asked to represent the location of the electrodes placed on a patient's head, as well as the EEG signal values measuring brain activity, using a mesh model of a human head. Electrodes are represented as white spheres with an associated label. The colour value assigned to each mesh vertex is defined by the signal value at the closest electrode. This results in a final representation with different colours associated to the electrical potential variations on the patient's head (Figure 2).

Since temporal information is available, the user can also navigate through different acquisition times and observe the evolution of the EEG data.

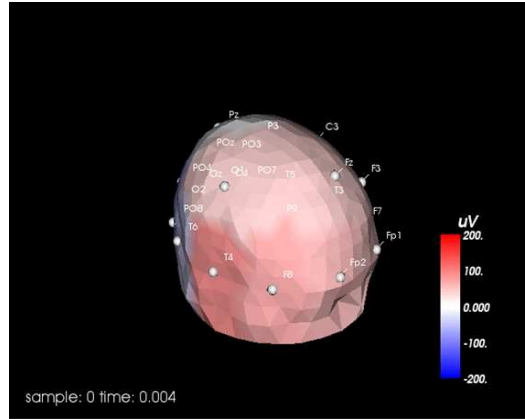


Fig. 2. Mapping EEG data on a model of a patient’s head.

Sources of electrical activity (dipoles)

A third group of students had to represent the estimated location of the sources of electrical activity (dipoles) within the brain. The dipoles are represented as arrows within a surface model of the patient’s head. Data is read from pre-processed files describing the sampling frequency, the location and the orientation of the dipoles. In addition to these features the application can also display different groups of dipoles using different colours. As for the EEG data, the user can also navigate forward or backwards in time.

Given the large range of dipole magnitudes, two visualization modes are available. In the first, the length of each arrow encodes dipole magnitude; in the second, all arrows are shown with the same length, thus simplifying the analysis and detection of clusters and patterns (Figure 3).

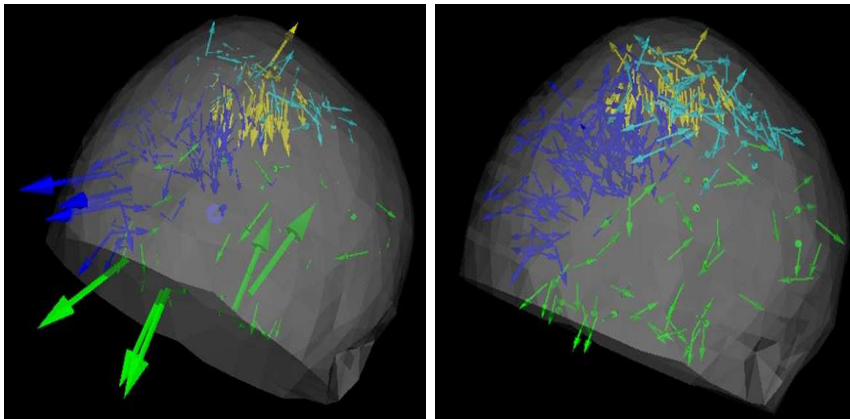


Fig. 3. Two modes of dipole representation.

B — Visualization of water pressure and velocity around a vessel’s hull

The goal of this project was to develop visualization tools to analyze the water flow around a vessel’s hull. The data consisted of the coordinates of sampled

points (unstructured grid), as well as pressure and velocity values.

Hedgehogs were used to represent velocity data and pressure was converted to a structured grid through splatting [16]. The final application gives the user the possibility to manipulate a cutting plane where the pressure data is displayed through colour mapping. A view of the final visualization is presented in Figure 4.

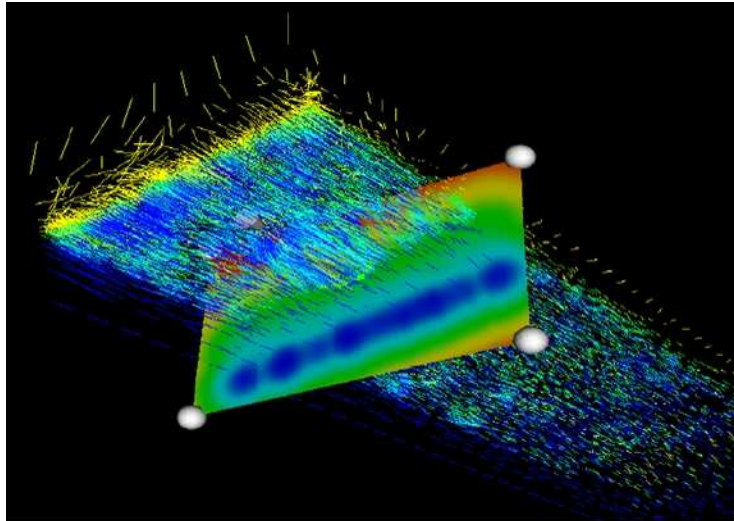


Fig. 4. Water velocity and pressure around a vessel's hull.

C — Visualization of acoustic data

Another interesting set of projects consisted in developing applications to visualize the propagation of sound waves, given time-varying data sets containing the results of sound propagation simulations.

2D sound wave propagation

This application allows the analysis of 2D data obtained from an acoustic modeling software, which simulates the interaction among neighboring points, and thus the propagation of sound waves, resulting from the injection of a sound pulse at an arbitrary point of a given environment, using a numerical model based on finite differences in the time domain (FDTD).

Two visualization modes are offered: for a given instant in time t , the sound wave values, as well as the associated zero-valued iso-lines are displayed (see Figure 5); as an alternative, the maximum values recorded so far, for time t , can also be represented. As usual, the user can navigate forward or backwards in time.

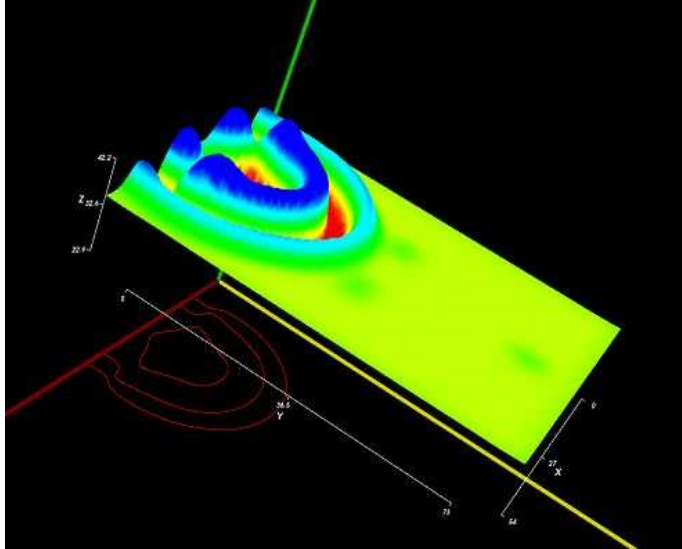


Fig. 5. Sound wave at a given time ($t = 39$): visualization of 2D propagation, as well as the zero-valued iso-lines.

3D acoustic pressure

The second application allows the analysis of data resulting from 3D acoustic simulations, namely pressure values and the frequency response for a given 3D environment.

In both cases, 3D time-varying data are simultaneously visualized through two representations, using the same colour scale: one allows slicing through the model volume using two orthogonal plane widgets, the other shows the iso-surfaces (see Figure 6).

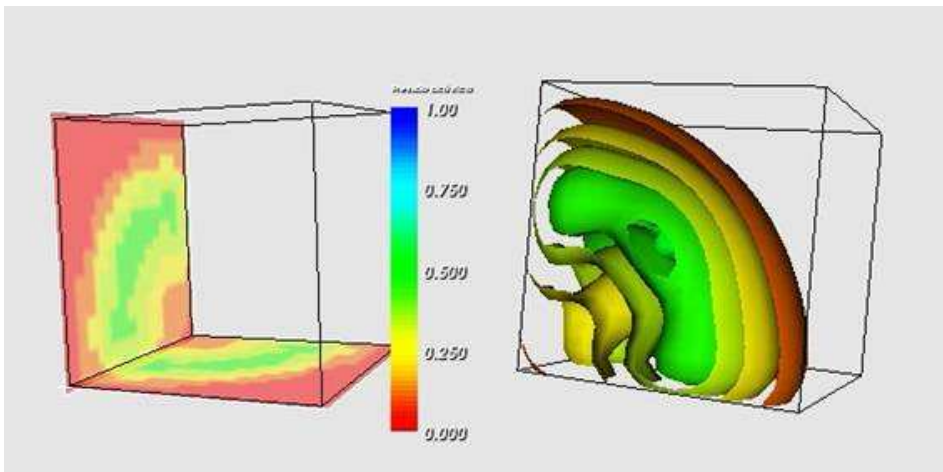


Fig. 6. Acoustic pressure in a cubical volume at a given time ($t = 29$): visualization using orthogonal planes (left) and iso-surfaces (right).

D — Visualization of Oporto underground tunnels

The goal of this project was to allow the visualization of point clouds, acquired with a 3D laser scanner, representing some tunnels of the city of Oporto underground network, in order to assess particular tunnel surface features.

A visualization application was developed allowing the representation of the scanned data in various ways: as a point cloud, a 3D Delaunay triangulated surface or a Gaussian splattered volume. In all these representations the user can interact with the data using a plane widget to define appropriate tunnel cross-sections (see Figure 7).

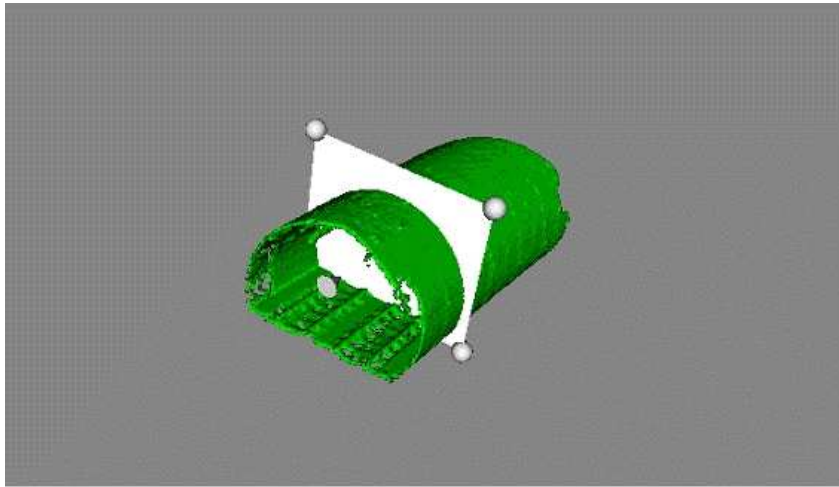


Fig. 7. Partial representation of the tunnel data using Gaussian splatter; the cutting plane is also visible.

E — Visualization of a football game in a VR environment

A different kind of project, *footVR*, was also developed: the goal was to visualize the dynamics of a football game using a Virtual Reality (VR) environment under development at our laboratory.

The application reads the log-file from a simulation of a robotic football game and allows the user to visualize the game in the VR environment. The developed software allows watching the game (robots/players are represented as simple triangular prisms) either in a desktop, by controlling the viewing angle (there is an option to follow automatically the ball), or in the VR environment, which includes a Head Mounted Display (HMD) and a tracker. In the latter, the user's head motion is registered and the camera parameters updated accordingly, as shown in Figure 8.

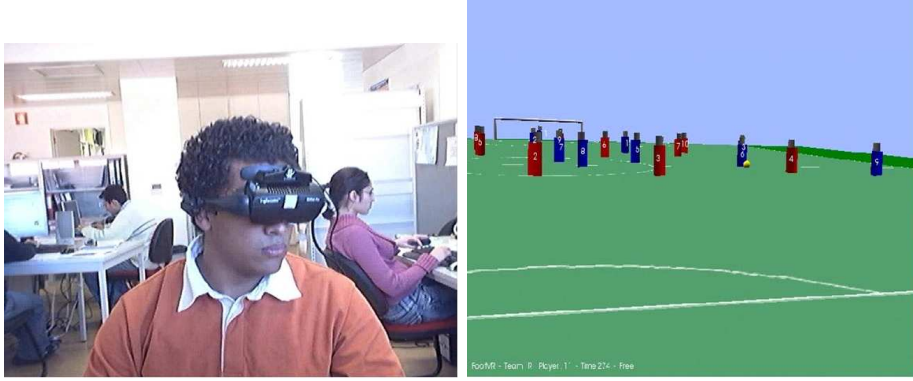


Fig. 8. User wearing the HMD (left), and his view of the game (right).

4 Using VTK: An Assessment

Since it is not usual to choose VTK as a tool supporting 3D modeling and visualization classes, we decided to assess the effectiveness of using this toolkit taking into account three different vantage points: (1) comparing the evaluation results of the OpenGL and VTK projects developed by our students; (2) analyzing the answers given to a questionnaire handed out to the students in 2006/2007; and (3) collecting our impressions on the use of VTK, given the questions and reactions from the students during the VTK lab classes.

4.1 Evaluation of the OpenGL and VTK projects

In addition to developing and implementing their 3DMV projects, one using OpenGL, the other using VTK, students were required to write a report describing the main steps of the work carried out, as well as present their work to their colleagues. Their projects were then individually evaluated by the authors (see Table 1 for the global results in the last two years).

Instead of directly comparing the grading of individual projects, we performed a relative comparison and identified students having similar evaluation results in both projects, or performing significantly better or worse in VTK than in OpenGL (see Table 2).

While, for 2005/2006, the results in Table 2 seem to validate our initial idea that using VTK would not be an excessive load for most students, and would be an excellent way of further motivating interested students, this is no longer true for the results obtained in 2006/2007: less students had excellent results in both APIs or performed significantly better when using VTK; moreover, one third of the students performed significantly worse when using VTK, than when using OpenGL.

	2005/2006		2006/2007	
	OpenGL	VTK	OpenGL	VTK
Weak	4	4	2	3
Average	4	4	6	7
Good	4	—	6	4
Very Good	6	10	4	4
Number of students	18	18	18	18

Table 1
Evaluation of the OpenGL and VTK projects in 2005/2006 and 2006/2007.

	2005/2006	2006/2007
Weak evaluation in both OpenGL and VTK	4	2
Average evaluation in both OpenGL and VTK	4	4
Significantly worse evaluation in VTK than in OpenGL	—	6
Significantly better evaluation in VTK than in OpenGL	4	2
Excellent evaluation in both OpenGL and VTK	6	4
Number of students	18	18

Table 2
Analysis of the evaluation results of the OpenGL and VTK projects.

Clearly, the results in Table 2 seem to be inconclusive. Although a possible explanation for the worse results obtained in 2006/2007 with VTK might be the excessive workload, during the second half of the semester and due to other courses. Further work is needed to try to better identify global student shortcomings, as well as particular questions and needs regarding the use of VTK.

4.2 Questionnaire handed out in 2006/2007

A questionnaire was handed out to the students at the end of their 3DMV course: the main goal was to gather their own opinion regarding the work carried out when developing the two projects, as well as additional information on their use of OpenGL and VTK, and on their experience previous to the course.

The questionnaire was anonymous and we had 16 (out of 18) respondents: 13 declared having average or above average programming experience, and 13 had

never used a CG programming API before. We globally analyzed their grades in the various programming and algorithms and data structures courses they had attended before, and concluded that less than half of them could really claim having average or above average programming experience.

Regarding the OpenGL project, 14 students considered themselves satisfied with the applications they had developed, and 11 classified their work as being above average. This overall evaluation is quite similar to our own grading of their work.

Regarding the VTK project, the main conclusions that can be extracted from the answers to the questionnaire are:

- 13 students considered themselves satisfied with the applications they had developed, and 11 classified their work as being above average.
- although there is no significative difference, students globally dedicated slightly less time to the VTK project than to the OpenGL project.
- although there is no significative difference, students globally declared having had slightly less difficulty in developing their VTK project than their OpenGL project.
- students considered that, on average, the information available for VTK was mostly insufficient, and poorer than the information available for OpenGL.

Regarding our own evaluation of the work carried out by the students, their answers to the questionnaire revealed that most of them had expected better grades. This was probably due to VTK allowing them to develop applications providing visually pleasing results with less work, when compared to OpenGL.

We were quite surprised to find out that students seemed to have dedicated slightly less time to their VTK projects, and found it slightly less difficult, in comparison to the OpenGL projects. In our opinion, the VTK projects are more demanding and would have required more time and effort for the students to attain above than average results. Such an attitude by the 2006/2007 students might also explain why the global evaluation in Table 2 is worse than that for 2005/2006.

4.3 Our own impression

In spite of the results in Table 2, our overall evaluation on the use of VTK is encouraging. We were positively surprised by the quality of the visualization applications developed by some of our students, since the time allocated to the final VTK projects was relatively reduced. Using VTK allowed us to propose challenging and motivating tasks, and students were able to develop relatively complex applications in a short time. This was certainly rewarding for most

students.

For the students, the object-oriented structure of VTK and its modularity were also important advantages. However, many students complained about the lack of appropriate documentation to help them use VTK. The available documentation is very often insufficient to clearly understand the features of the classes used, and examples are missing for many functions.

Even with the help of the user's guide [24], it is often difficult to understand at first how VTK classes behave: this is certainly a strong limitation of the toolkit, which does not recommend its use by students with less programming experience or reduced knowledge of the object-oriented paradigm. In some way, using VTK can even be frustrating for a student, since final solutions to some programming or development difficulties are often compact (a few lines of code), but difficult to attain. A possible way to mitigate this problem might consist in providing the students with a set of additional code examples.

The above mentioned shortcomings of VTK force students to a somewhat important effort, during their first contact with the toolkit, in order to overcome first difficulties. For students with low motivation this was a major drawback, and a few did not succeed in developing satisfactory work.

Nevertheless, most students particularly appreciated the use of a higher-level tool as VTK, which allows developing working prototypes and provides some degree of interaction and appropriate visualization functionalities. Some students were also asked to write a short paper describing the main features of their work, to be published in the internal journal of our department. Despite the fact this additional work was asked for after the conclusion of the semester, almost all of them agreed. This exercise was, after all, a nice introduction to more challenging research projects that might be proposed to some of them later.

5 Conclusion

We have presented our experience regarding the use of VTK, in the last two years, in the context of the elective “*3D Modeling and Visualization*” course at the University of Aveiro. In spite of the global results in 2006/2007 being somewhat poorer than expected, when compared to 2005/2006, our overall evaluation on the use of VTK is encouraging.

Using OpenGL in the first lab classes provides a valuable first step for those students with no previous CG programming experience. Then, the object-oriented structure of VTK and its modularity, as well as the visualization and

interaction functionalities available, are of great advantage for most students, who are able to develop complex visualization applications in a relatively short time.

However, it is also clear for us that the learning-curve of VTK might delay or even prevent the progress of some students, and we still have to devise appropriate strategies to mitigate this problem. Since we intend to keep using VTK in our CG courses, both at graduate and post-graduate level, this will be our next challenge.

6 Acknowledgments

First of all, we wish to thank all the students who developed the visualization applications presented here.

A word of gratitude goes to our colleagues: José Maria Fernandes, José Rocha Pereira, Filipe Teixeira-Dias, Guilherme Campos, Augusto Silva and José Silvestre Silva, who provided us with some of the data sets used in the various assignments. Thanks are also due to the 3D laser digitalization company *ARTESCAN*, for providing us with the Oporto underground tunnels data.

We thank Samuel Silva for his help in preparing the final version of this paper. And the anonymous referees for their comments and suggestions.

References

- [1] S. Grissom, J. Bresenham, B. Kubitz, S. Owen, and D. Schweitzer. Approaches to teaching Computer Graphics. In *Proc. SIGCSE 1995*, pages 382–383, Nashville, Tenn., 1995.
- [2] L. E. Hitchner and H. A. Sowizral. Adapting Computer Graphics curricula to changes in graphics. *Computers & Graphics*, 24:283–288, 2000.
- [3] D. J. Bouvier. From pixels to scene graphs in introductory Computer Graphics courses. *Computers & Graphics*, 26:603–608, 2002.
- [4] E. Paquette. Computer Graphics education in different curricula: analysis and proposal for courses. *Computers & Graphics*, 29:245–255, 2005.
- [5] E. Angel, S. Cunningham, P. Shirley, and K. Sung. Teaching Computer Graphics without raster-level algorithms. In *Proc. SIGCSE 2006*, pages 266–267, Houston, Texas, 2006.

- [6] C. R. McCracken. Issues in Computer Graphics education. In *Proc. SIGGRAPH 2006 Educators Program*, Boston, Mass., 2006.
- [7] K. Sung, P. Shirley, and B. R. Rosenberg. Experiencing aspects of games programming in an introductory Computer Graphics class. In *Proc. SIGCSE 2007*, pages 249–253, Covington, Kent., 2007.
- [8] E. Angel. *Interactive Computer Graphics: A Top-Down Approach using OpenGL*. Addison-Wesley, 4th edition, 2006.
- [9] R. Tori, J. L. Bernardes Jr., and R. Nakamura. Teaching introductory Computer Graphics using Java 3D, games and customized software: a brazilian experience. In *Proc. SIGGRAPH 2006 Educators Program*, Boston, Mass., 2006.
- [10] J. Zara. Virtual Reality course — A natural enrichment of Computer Graphics classes. *Computers Graphics Forum*, 25:105–112, 2006.
- [11] M. Bailey. Teaching OpenGL shaders: Hands-on, interactive, and immediate feedback. In *Proc. EUROGRAPHICS 2006 Education Programme*, pages 57–60, Vienna, Austria, 2006.
- [12] J. O. Talton and D. Fitzpatrick. Teaching graphics with the OpenGL shading language. In *Proc. SIGCSE 2007*, pages 259–263, Covington, Kent., 2007.
- [13] J. Foley, A. van Dam, S. Feiner, J. Hughes, and R. Phillips. *Introduction to Computer Graphics*. Addison-Wesley, 1994.
- [14] D. Hearn and M. P. Baker. *Computer Graphics with OpenGL*. Prentice-Hall, 3rd edition, 2004.
- [15] S. Cunningham. *Computer Graphics: Programming in OpenGL for Visual Communication*. Prentice-Hall, 2006.
- [16] W. Schroeder, K. Martin, and B. Lorensen. *The Visualization Toolkit — An Object Oriented Approach to 3D Graphics*. Kitware, 4th edition, 2006.
- [17] J. Frost, S. Goessner, and M. Hirtzler. *Learn SVG: The Web Graphics Standard*. Self Publishing, 2004.
- [18] A. L. Ames, D. R. Nadeau, and J. L. Moreland. *The VRML 2.0 Sourcebook*. John Wiley & Sons, 2nd edition, 1997.
- [19] D. Shreiner, M. Woo, J. Neider, and T. Davis. *OpenGL Programming Guide*. Addison-Wesley, 5th edition, 2005.
- [20] G. Davis. *Learning Java Bindings For OpenGL (JOGL)*. Author House, 2004.
- [21] D. Selman. *Java 3D Programming*. Manning Pub., 2002.
- [22] F. D. Luna. *Introduction to 3D Game Programming with DirectX 9.0*. Wordware Pub., 2003.
- [23] P. Dias, J. Madeira, and B. Sousa Santos. Using VTK as a tool for teaching and applying Computer Graphics. In *Proc. EUROGRAPHICS 2006 Education Programme*, pages 61–67, Vienna, Austria, 2006.

[24] Kitware Inc. *The VTK User's Guide — Version 4.2*. Kitware Inc., 2003.