Virtual Reality Course – A Natural Enrichment of Computer Graphics Classes

J. Zara

Czech Technical University in Prague, Czech Republic

Abstract

This paper shows how the Virtual Reality (VR) course can naturally extend and practice a wide range of Computer Graphics (CG) principles and programming techniques. Thanks to real-time processing requirements, the attention is paid to efficient modeling conventions, time saving rendering approaches, and user interfaces allowing a smooth navigation in a virtual environment. All these issues play an important role especially when designing virtual worlds targeted to the web, i.e. utilizing VRML/X3D standards. The paper presents a structure and related information of the VR course that has been taught at various universities during the last six years. Our experience clearly demonstrates that students appreciate its contents even if they already completed courses on 3D graphics.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Three-Dimensional Graphics and Realism]: Virtual reality K.3.1 [Computer Uses in Education]: Collaborative learning K.3.2 [Computer and Information Science Education]: Computer science education K.3.2 [Computer and Information Science Education]: Curriculum

1. Introduction

A growing interest on computer graphics (CG) is undoubtedly associated with an increasing power of computers and graphics cards. As people are familiar with graphical output and graphical user interfaces, so they meet with real-time graphics applications like computer games or 3D graphics on the web (VRML, X3D). Computer graphics education is welcome at many universities and has become a standard part of Computer Science curricula. Similarly, Virtual Reality (VR) education meets with an interest both from academic and industrial area.

The importance of CG education encouraged an organization of Eurographics/SIGGRAPH Workshops on Computer Graphics Education (CGE). The recent one, the third in a row, was held in China, 2004 [CGE04]. The final report from that event declared a necessity to provide at least introductory course on CG to *every* computer science student. This is not just a wish but a reflection of real requirements coming from many universities.

Contrary to that, virtual reality is not supported very extensively. As stated in [Bur04], VR courses are offered at only 3% of universities worldwide. The obvious difficulties observed are:

- a lack of teaching materials,
- a lack of special VR hardware,
- and a lack of experienced educators.

Indeed, high-quality online educational materials are not widely available, although certain resources already exist [Bur03]. In this paper we wish to contribute to all three items introduced above. We come out of our experience with teaching VR course for undergraduate students at several universities. All lecture notes are available on the web. Since the VR course is based on using VRML language, no special hardware support is required, thus it is not limited to universities/laboratories equipped with expensive advanced VR devices. Finally, we want to share experience gained during more than six years of teaching the course. Our belief is that VR education will play increasingly important role, since it represents a natural complement to a standard CG education.

The paper is organized as follows. A relationship between CG and VR education is discussed in Section 2. The main contribution lies in Section 3 where the VR course structure is presented in details together with the most interesting observations from practical education. A teaching experience is described in Section 4. Section 5 concludes the paper.

2. Virtual reality and Computer graphics education

A usual question is raised when discussing a relation between a VR course and other subjects: Should be a VR course taught prior to CG courses or after them? The answer is not unambiguous – both options are applicable. Phelps and Sonstein recommend teaching a course based on the VRML language as an introductory course *to excite students in the possibilities of computer graphics* [PS01]. Since VRML does not require any special VR equipment and runs on low-end hardware, this approach is definitely useful for wide range of universities. However, virtual reality techniques and applications are hardly to be educated extensively in that stage due to insufficient knowledge of students in related areas like spatial data structures, collision detection algorithms, hardware acceleration techniques, etc.

The opposite approach places VR course(s) after CG courses. Thanks to previous understanding of CG principles, the education can be more oriented towards efficient utilizing of computers and peripherals to create well-designed virtual environment and applications.

A decision where to situate a VR education within a computer graphics curriculum depends on a particular specialization of given university. A virtual reality simplified to the VRML is a perfect choice of introductory course to CG in general, while the second approach (or a combination of both) is better for universities educating computer graphics specialists. In this paper, we concentrate on the later case, i.e. on VR education for CG oriented students.

It should be mentioned that VR is also used for teaching subjects other than computer graphics. Virtual reality is getting more and more important role as *a tool* for education in the field of architecture, building constructions, art, medicine, etc. A number of contributions dealing with distant learning, collaborative work, and virtual laboratories utilizing VR can be found in *Frontiers in Education Conferences* [FIE]. A representative list of resources for such kind of education is in [ER97] (unfortunately a corresponding web page is not regularly maintained and updated).

It is curious that a number of papers describing education *with* virtual reality exist, while papers discussing education *of* virtual reality are still rare. One aim of this paper is to alter that situation.

3. Five steps to understand real-time rendering

The VR course has been offered to students already acquainted with computer graphics in general and adequately with other subjects from computer science. Its content is briefly described in Table 1. The first nine lectures utilize the VRML language, the rest is language independent. To show the advantageous connection to CG education, we have chosen five parts from the first half of the course for a detailed description in this paper. Our aim is to demonstrate that VR education helps understanding issues coupled with a real-time 3D graphics, i.e. to concentrate on a balance between a visual quality and a rendering speed.

1	Introduction to VR
2	Navigation in 3D
3	3D objects and scene graph
4	Colors and textures
5	Lights and sounds
6	LOD and other speeding up techniques
7	Event processing and animation
8	Interaction techniques
9	EAI interface and Java applets
10	Panoramic VR, augmented reality
11	Multi-user VR
12	Special hardware for VR
13	3D on the web (X3D)

Table 1: *VR course structure*

Selected parts from the VR course curriculum are introduced in the following subsections. We intentionally do not explicitly differentiate between a lecture and a corresponding practical exercise – the idea is to analyze a selected topic from various points of view. Anyway each of the next subsections contains at least:

- main objective
- student task description,
- teacher's comments,
- curiosities,
- and lessons learned.

We point out the fact that student tasks (assignments) are designed as a progressive work starting with a simple 3D scene and finishing with a complex, highly interactive virtual environment. Since students continuously add more objects and functionality to existing (previous) data, they are naturally forced to keep all data and other program constructions well readable and manageable.

3.1. First step – Navigation and viewing

Students get familiar with camera settings and navigation paradigms in a 3D virtual environment. The topic does not tightly correspond with CG itself, but reminds of necessity to properly manage camera parameters when designing a walk/fly consisting of several viewpoints. A minor objective is to arrange basic solids into a 3D space, i.e. to work with transformations.

Task: Create a virtual garden consisting of basic solids only (Fig. 1). Make a fence, and a single shed or a pergola. Prepare a number of viewpoints forming a virtual tour. Arrange objects and viewpoints in such a way that a user has to apply all navigation paradigms to continue.

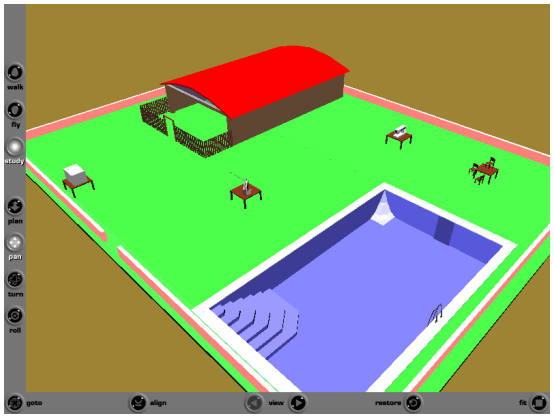

Figure 1: *A simple scene for a virtual walk-through test (rendered by the Cortona VRML browser).*

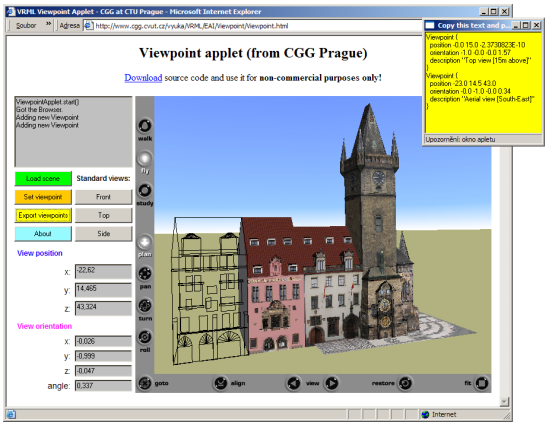

Figure 2: *A Java applet (on the left) helps to specify viewpoints for a VRML scene interactively [Zar].*

Comments: A relatively new issue for CG students is *navigation* in a virtual space. Three main paradigms based on evaluation of the couple [gravity, collision detection] are distinguished – WALK [on, on], FLY [off, on], and EXAMINE [off, off]. Another news is a concept of *avatar*, i.e. a virtual double of a user. A size of a virtual but invisible body prevents the avatar from passing through a narrow space between objects, but allows overcoming small obstacles on a ground. Gravity can take the avatar down to a canyon from which the only way back is to switch from WALK to another mode.

A terrain follow feature associated with a WALK navigation paradigm requires additional computational power. It is possible to show/measure how the evaluation of gravity reduces a rendering speed typically by 5% to 10% comparing with EXAMINE mode.

Curiosity: We suggest to use a well-known VRML syntax-driven text editor VrmlPad from ParallelGraphics. Contrary to WYSIWYG modelers it allows in-depth understanding of 3D models, their structure and characteristic. Unfortunately, a missing WYSIWYG feature is the most painful when defining viewpoints (camera positions). To help students, we have developed a tool for an interactive setting of viewpoints, see Fig. 2. This web application utilizes a Java applet. It reads arbitrary VRML scene and allows creation of new viewpoints defined in current users' positions as they examine the virtual world. Due to security limitations of Java applets, new viewpoints are not written back to the source file, but displayed in a text window (shown in upper right corner in Fig. 2) ready for pasting via clipboard.

Lessons learned:

- Properly designed viewpoints help non-expert users to get oriented when lost in a virtual space (controlled by a simple 2D device such as a mouse).
- It is important to fix a field of view for all viewpoints to the same angle. If not, users have troubles with guessing distances in 3D space.

3.2. Second step – Single object design and a scene graph

The aim is to practice steps from a creation of a 3D model to its efficient rendering together with a deeper understanding of a boundary representation of solids. Another issue is a scene graph that reuses already defined objects and their parts.

Task: Make a model of a garden/home technical device, e.g. lawn mower, tractor, coffee maker, washing machine, sewing machine (Fig. 3). Put several instances of that model into a scene from the previous task. Balance between a visual quality and a speed of rendering.

Comments: Students basically have two options – either to work with a full 3D modeler or to create objects 'by hand' using VRML nodes like IndexedFaceSet, Extrusion, or ElevationGrid. Both techniques have their pros and cons. An ease of interactive design using a modeler is followed by hardly maintained and usually not well-structured exported data. Manual definition of 3D data is a tedious work, but leading to optimized and well arranged VRML code suitable for further operations and programming. After a few attempts, most students prefer a combination of both approaches, i.e. to form a 3D shape in a modeler, then to tune it manually.

Students are often surprised how inefficient the export from a general 3D modeler (namely 3D Studio Max) is. The flaws belong to two categories – large files produced, and non-optimal data structures used. The former includes using triangle meshes for basic solids like a sphere or a cylinder instead of concise geometrical description (a data growth is

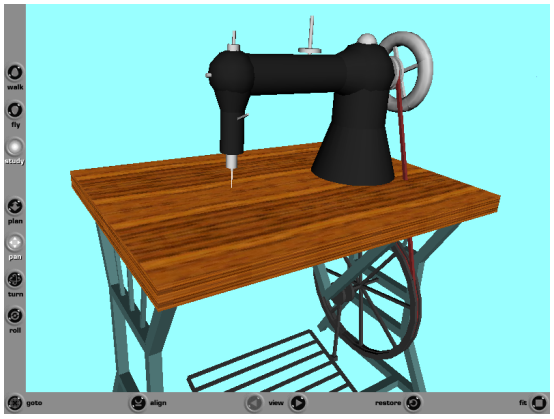

Figure 3: *More complex shaped virtual machine.*

in orders of magnitude in such cases), or unnecessary definitions of normal vectors that could be derived automatically from the mesh and tuned by proper setting of *creaseAngle* parameter that controls a surface visual smoothness. The later covers a use of two-sided faces for solids and a decomposition of general polygons into stand-alone triangles, thus avoiding more efficient data structures like triangle fans and/or strips.

General modelers usually do not offer removing cylinder/cone caps even if these solids are glued to other objects in such a way that caps cannot be viewed. A well-known example is a table with cylindrical legs. If a solid top board fully covers all legs and the table stands on a ground, all cylinder caps can be removed. This simple trick saves almost 50% of triangles.

Curiosity 1: To support unbiased measurement of time (fps – frames per second) we have developed a Java applet monitoring a speed of rendering and providing average fps and a variance. The applet was also used for evaluation and comparison of rendering speed of various VRML browsers [ZK01].

Curiosity 2: A specific feature of the VRML language is a possibility to build complex scenes from objects stored on a (distant) web server. This is highly useful for a team work when team members create their models locally, then a final scene references them via URLs. Once, a team made an array of instances of the same original object using multiple references to a distant server. A download of the whole scene took about 1 minute, since every reference required a client-server communication. Then a structure of a scene was changed in order to reference a distant object only once and to link all other references to that object locally in a scene graph. A download time dropped down to 4 sec. This applies also for other scene resources like distant textures (images) and audio clips.

Lessons learned:

- Efficiently designed models substantially decrease a file size, thus saving both transfer time (when downloaded from the web) and computer memory.
- Most export filters (to VRML) produce a mess of data that is hard to manage and often contains redundant information.
- About 50% of time can be saved by using one-sided faces instead of two-sided ones for solids (thanks to back-face culling). Another savings can be achieved by removing parts that are never visible, because fully covered by other objects.
- Reusing parts of already defined scene graph makes a code both manageable and efficient.
- A scene distributed over the web introduces networking issues for CG students.

3.3. Third step – Colors and textures

Geometrical objects managed in a previous lecture are enhanced by colors and textures. The attention is paid to a selection of proper images applicable as textures with respect to a texture memory organization. A relationship between 3D graphics algorithms and their implementation in a GPU (Graphics Processing Unit) is presented.

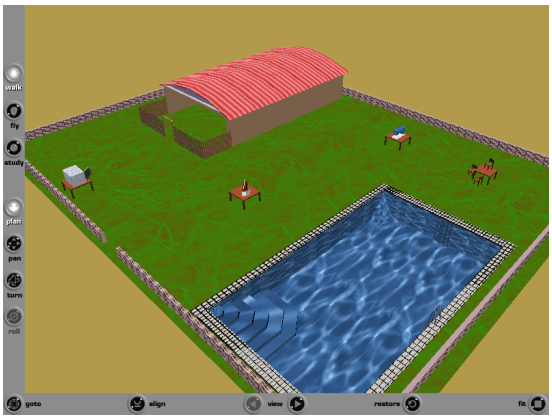

Figure 4: *Textured scene from Fig. 1.*

Task: Improve the visual quality of previously created objects using various colors and textures.

Comments: Most students like to make objects semitransparent to simulate (colored) glass (see Fig. 5). Indeed, a possibility to see background objects through foreground ones brings an interesting and nice design results. However, the speed of rendering is influenced negatively, since the alpha blending (used for transparent faces) is not a commutative operation. Semitransparent objects have to be sorted and rendered in back to front order exactly like in a well known

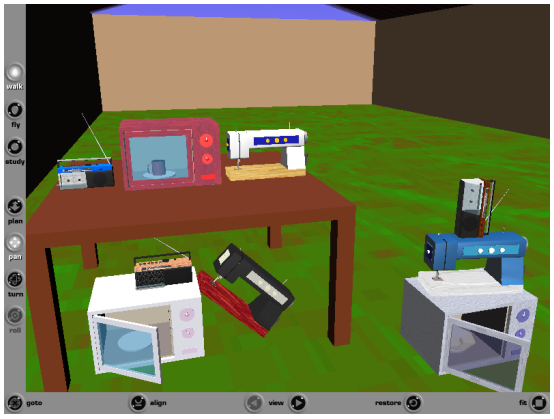

Figure 5: *A test scene with semitransparent and textured appliances.*

painter's algorithm and in contrast to a z-buffer visibility algorithm implemented in GPU. This knowledge should warn students of using too many transparent objects.

It is useful to discuss properties of the most frequently used file formats (GIF, JPEG, and PNG) in terms of color resolution, image quality, and transparency. Since a texture memory of a graphics card has always a limited size, its optimal occupation plays an important role. How many CG students know that the most important issue for textures is not a file size but an image resolution? How many of them are acquainted with a block structure of a texture memory managed by OpenGL that leads to prefer images with a size that is a multiple of power of 2, usually 64, 128, or 256 pixels?

Curiosity 1: If a virtual world stored on a server references tens of texture images, every image file requires an individual network handling what leads to a significant delay at the beginning of a rendering. A solution is to arrange all textures into one composite image (known as *texture atlas*) and download it at once. An optimal arrangement of a big number of images of various sizes into a composite image is an interesting optimization problem, not yet fully solved. A networked virtual reality turns attention to a computational geometry.

Curiosity 2: Students often try to take photos of real objects and environment and apply them as textures or background images. Due to rapidly growing resolution of commonly available digital cameras, e.g. 5 Mpixels, students easily overfill a texture memory of a common GPU if they apply several non-processed photos with original resolution. One such image occupies 20 MB of texture memory when a mip-mapping is required. If a concept of *textures* was explained in previous CG courses as something what can replace a huge number of polygons representing a (micro) structure of a given surface and thus to significantly increase a rendering speed, now students may be surprised by reversed effect of using textures. A theory fails when wrongly used textures reach hardware (GPU) limitation.

Lessons learned: A knowledge of visibility algorithms and organization of texture memory helps to discover reasons for performance reduction.

3.4. Fourth step – Lights and sounds

The topic covers a use of light and sound sources, their proper positioning in a 3D scene and settings their parameters.

Task: Enhance every instance of a virtual appliance by one individual color light source (usually placed above the model). Record real sounds and play them in a virtual space attached to typical user's activities – manipulation with an object (drag), and basic interaction with it (click).

Comments: It is useful to remind the fact that light equation is basically evaluated only for vertices, while the rest of the surface is shaded using interpolation (Gouraud, Phong). If a face of an object is too large comparing with a range of a local light source then unexpected visual results can occur.

An example is a point light source with a maximal range of 3 m that is placed 2 m above a larger rectangular ground. Since the distance from the light to corners of the ground is bigger than 3 m, corners are evaluated as completely dark and subsequently all ground as well. No light area can be seen on a ground until the rectangular geometry of the ground is replaced by a mesh with certain density. To see local light effects in a good quality, more polygons have to be rendered. Students should be also informed that local lights are more computational expensive than infinite lights.

Curiosity 1: Virtual reality provides a stereo sound generated when navigating through a virtual environment. The *stereo effect* is caused by a spatial relation between a position of a user and a position of sound source(s) but not by a stereo record stored in source audio files. Students frequently download stereo audio clips from the web and play them from a point sound source in a virtual environment. A file size (in case of uncompressed WAV format) is twice as big as necessary – a good example of wasting space for data.

Curiosity 2: Students are always unpleasantly surprised by a low number of light sources supported by a majority of graphics cards. A common limit of 8 lights is reached pretty fast in their virtual scenes and additional lights always cause troubles. Either a slower software rendering takes the place of the hardware accelerated one or extra lights are simply ignored. To keep a rendering speed satisfactory, students have to carefully and dynamically manage an overall number of light sources concurrently switched on. A knowledge of a position of a light model evaluation in a rendering pipeline helps to understand the reason why additional lights cannot

be simply processed in advance by CPU, while having the first 8 lights evaluated in GPU as before.

Lessons learned: Without knowledge about GPU limitations, theoretical expectations can easily fail.

3.5. Fifth step – Interaction and Animation

Basic interaction techniques are discussed. Animation based on a key-frame (key value) method is introduced together with interpolation techniques. A concept of *event* and its application for simulation purposes is introduced.

Task: Enhance a model by animation of selected parts. Change positions, orientations, a scale, and colors of relevant geometrical components. Allow direct manipulation (e.g. dragging) with specific parts. Visualize a process of assembling individual components into a final product.

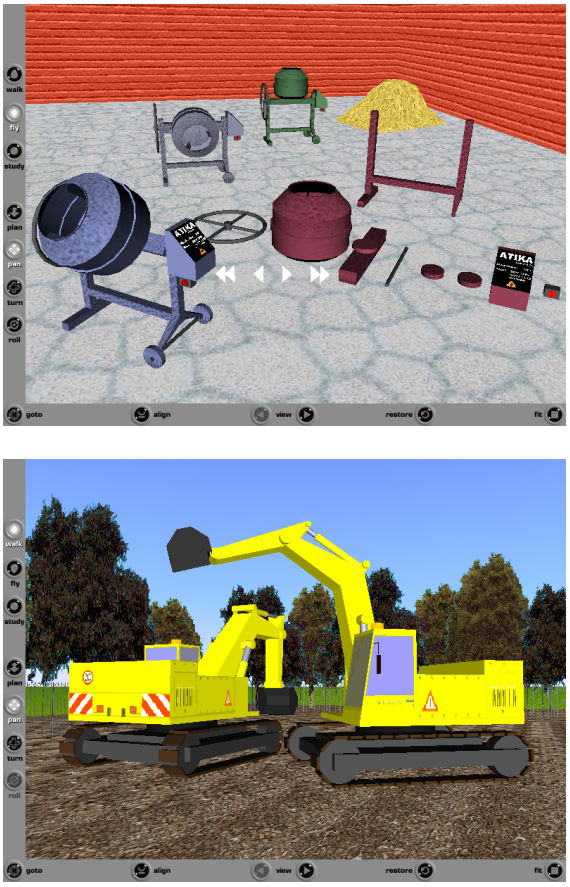

Figure 6: *Two examples of interactive and animated models. Some objects can be disassembled/assembled either step by step or in a single animation (top image).*

Comments: Although this lecture does not deal with advanced animation methods like humanoid animation, it still contains interesting issues. Since the interactive manipulation with one part of a device usually propagates position/rotation changes to other parts, students have to manage a chain of transformations properly. The example is a synchronized movement of arms and pistons of a crane in Fig. 6 (bottom image).

Curiosities: No techniques for interaction and a design of user interfaces applicable to a 3D virtual space have been standardized and generally accepted yet contrary to the well-established WIMP paradigm known from 2D window-based applications. A lot of students are disappointed by the situation when a number of manipulation possibilities and rich animations have been designed and implemented in their own virtual worlds but no user notices those features! That is why we recommend to enhance the virtual environment by unnatural objects like flickering text messages flying over interactive objects ("Drag me!)", pulsing arrows pointing to sensitive pieces, or at least by highlighting objects (changing their color/brightness) when a mouse pointer is moved over them. Another recommended approach is to use a HUD (Head-Up Display) panel – a virtual control device that follows a user's movement in a scene and is displayed at the same position on a screen, in front of the virtual scene (see Fig. 7, bottom images).

Lessons learned: Users are not familiar with direct interaction and manipulation with objects in a 3D virtual environment. Various visual clues are highly important here.

3.6. Other parts of the course

The course deals with a number of other interesting issues like mesh simplification techniques (required for Level of detail management), utilizing billboards and impostors, collision detection algorithms, humanoid animation, special user interfaces suitable for immersive or augmented reality, etc.

It is up to a teacher how deep he/she wants to go in those advanced topics related to a real-time rendering, a simulation, and a human-computer interface.

4. Teachers and students

The VR course described in previous sections has been taught at various universities – Czech Technical University in Prague (CZ), Charles University in Prague (CZ), Masaryk University in Brno (CZ), Stuttgart University of Applied Science (DE), and Tecnologico de Monterrey, Mexico City (MX).

The course has been organized in various forms. The most common case was a regular course scheduled to one semester. For special cases, duration of the course was reduced to one week, but the number of contact hours, especially lectures, was kept at the similar level as in the case of the regular course. We call that track *intensive course*.

Figure 7: *A scene with virtual musical instruments that can produce music both interactively and using pre-recorded melodies is a result of a team consisting of three students (top). Bottom images show individually created interactive models controlled by HUD user interface.*

Most participants attending the course were master program students specialized to Computer Graphics. The others were educated in Computer Science in general. On overview and the most important teacher's experience is presented in the following sections.

4.1. Timing issues

The intensive one week course requires 100% student concentration on the subject thus it should be open only if students are free of other duties. We have organized intensive courses either out of semester (e.g. last week before the beginning of a semester) or as a special event for short-term foreign student visitors (in ATHENS programme [ATH]). In such a case, morning time is optimal for lectures, while afternoons are suitable for practical work on (sub)projects.

Students attending intensive course has a limited time for doing projects comparing with a regular, full semester course. Their results should be evaluated with respect to that fact. We recommend to prefer a one semester course. Ample time enables not only to implement all assignments in a good quality, but opens students' minds in terms of novel ideas, designs and solutions. We meet many times with student VR projects that incorporated a number of additional features not required by a teacher, but developed by students themselves just because they were excited by the subject and possibilities to do *something new, nice and interesting*.

Our hope is that teaching materials available on our web helped students a lot. It concerns not only lecture notes, but also additional tools developed (e.g. the Viewpoint applet [Zar]) and previously completed student projects published on the web as examples.

4.2. Team work

Progressively implemented student projects can be considered as either individual or team works. Our observation is that properly organized team work is much better than the individual one in case of this course.

An optimal number of team members seems to be two or three – all team members have the same position, no special team manager is established. All students do the same assignments (*individual effort*), but resulting virtual models are placed in a common virtual environment with the same style of control/interaction and navigation techniques (*team effort*). The evaluation considers both team and individual results which leads to cooperation among team members – students within a team help each other to receive the best team rating. Selected results can bee seen in Fig. 7.

Since the VRML/X3D supports network-based virtual worlds, students naturally utilize the web similarly like in the case of modern *distant learning* approach.

5. Conclusion

The aim of this paper was to share the most interesting experience obtained when teaching Virtual Reality course for more than six years. An overall number of participants (mostly master students) have been over 650.

A majority of them judged a difficulty from 2.5 to 4 in the scale of 1 (low) – 5 (high), but the usefulness from 4 to 5. A correlation (inversely proportional) between a difficulty and a number of previously attended Computer Graphics courses was evident. Generally positive opinion about the usefulness shows that Computer Graphics students find Virtual Reality course highly beneficial. They claimed that the VR course provided additional, practical, and in several meanings novel information to their previous knowledge. Many of them were motivated and enthusiastic about the fact that VRML/X3D brings a 3D graphics to anybody connected to the web.

The Virtual Reality course extended classic Computer Graphics lectures and opened new ways of thinking about CG applications and programming.

This paper can be also considered as a contribution to the idea of creation of the *refereed repository for computer graphics educational materials* as proposed in Eurographics/SIGGRAPH Workshop on Computer Graphics Education in Bristol, UK, 2002 [CGE02]. All lecture notes dealing with discussed VRML course are freely available on the web [Zar97].

Acknowledgements

We express our thanks to all students visiting our VR course(s) in previous years and providing a valuable feedback that helped us to tune a course structure and improve its contents. Special thanks belong to students who developed nice virtual models and scenes whose images were used in illustrative figures in this paper.

References

[ATH] *The ATHENS European Network*. http://www.intersek.ntnu.no:591/athensreg/. 7

[Bur03] BURDEA G. C.: VR instructor's resource page for Virtual Reality Technology. http://www.caip.rutgers.edu/vrtechnology/resources.html. 1

[Bur04] BURDEA G. C.: Teaching virtual reality: Why and how? *Presence 13*, 4 (Aug. 2004), 463–483. 1

[CGE02] *CGE02 – Eurographics/SIGGRAPH Workshop on Computer Graphics Education*. Bristol, UK, 2002. http://www.siggraph.org/education/bristol/bristol.htm. 8

[CGE04] *CGE04 – Eurographics/SIGGRAPH Workshop on Computer Graphics Education*. Hangzhou, China, 2004. http://www.cad.zju.edu.cn/cge04/. 1

[ER97] EMERSON T. C., REVERE D.: *Virtual Reality in Training and Education (EdVR): Resource guide to citations and online information*. HITL Technical Publications: B-94-1, University of Washington, 1997. http://www.hitl.washington.edu/kb/edvr/. 2

[FIE] *Frontiers in Education Conferences*. http://www.fie-conference.org/. 2

[PS01] PHELPS A. M., SONSTEIN J.: Recommended practices for vrml education. http://www.it.rit.edu/~jxs/VRML/recommended_practices_draft2.html. 2

[Zar] ZARA J.: Viewpoint applet. http://www.cgg.cvut.cz/vyuka/VRML/EAI/Viewpoint/Viewpoint.html. 3, 7

[Zar97] ZARA J.: Lecture notes for VRML course. http://www.cgg.cvut.cz/~zara/courses/links.html. 8

[ZK01] ZARA J., KRIVANEK J.: Graphics performance benchmarking based on VRML browsers. In *VRIC 2001 Proceedings* (2001), Laval : ISTIA Innovation, pp. 111–120. 4