An Extensible Scene Graph Library for Teaching Computer Graphics along the Programmable Pipeline*

Volker Ahlers
University of Applied Sciences and Arts Hannover
Faculty IV, Dept. of Computer Science
Ricklinger Stadtweg 120, 30459 Hannover, Germany
volker.ahlers@hs-hannover.de

ABSTRACT

Computer graphics is a subject which is typically enjoyed by students and which has the potential to attract pupils to consider studying computer science. Although the programming methods used by computer graphics have significantly changed in recent years due to the integration of programmable shaders into the graphics rendering pipeline, a lot of computer graphics courses still start with the fixed-function pipeline. In view of future applicability, however, it is desirable to teach students modern concepts of computer graphics from the beginning. One problem with teaching shader-based computer graphics is that a lot of technical tasks lie in the hand of the programmer: loading and compiling shader programs, managing buffer objects, defining transformations by means of matrices, etc. This poster presents a scene graph library which is fully based on the programmable rendering pipeline. It uses the OpenGL 3.2 core profile, which does not allow deprecated fixed-function functionality. The teaching approach combines the high-level abstraction of a scene graph with the low-level programming of shader cores, which are attributed to scene graph nodes. The presented scene graph library has a simple and clear structure and is extensible in order to let students implement advanced concepts taught in the lecture, like shadows or particle systems. Finally, the poster presents code samples, results of student projects, and student evaluation results.

Categories and Subject Descriptors

I.3.4 [**Computer Graphics**]: Graphics Utilities; K.3.2 [**Computers and Education**]: Computer and Information Science Education

Keywords

Computer Graphics; Scene Graph; Rendering Pipeline; Programmable Shaders; Software Library; Laboratory

*Poster proposal, revision January 2, 2014

SIGCSE'14 March 5–8, 2014, Atlanta, Georgia, USA

1. SIGNIFICANCE AND RELEVANCE OF THE TOPIC

Computer graphics is a classic course of many computer science curricula which is typically enjoyed by students. Video games and animated feature films often arouse interest in the subject, and visual 3D results of project-oriented laboratory classes are a welcome diversion from more abstract topics of other courses. In consequence, computer graphics is well-suited to attract pupils to consider studyding computer science.

On the technical side, the programming methods used by computer graphics have changed a lot in recent years due to the integration of programmable shaders into the graphics rendering pipeline. Since Direct3D 9.0 (released 2002) and OpenGL 2.0 (2004) it is possible to program certain pipeline stages in high-level shading languages. Since Direct3D 10.0 (2007) and OpenGL 3.0 (2008) the fixed-function pipeline is declared obsolete or at least deprecated [4, 8]. Nevertheless, a lot of computer graphics courses still start with the fixed-function pipeline, mainly because it is easier to obtain first results without having to learn how to program shaders. If shaders are treated at all, they are introduced as a means to achieve special effects. For OpenGL, this is in part supported by the compatibility profile which still exists in latest OpenGL versions (but not in newer developments like OpenGL ES for embedded systems and WebGL). In view of future applicability, however, it is desirable to teach students modern concepts of computer graphics from the beginning. As a side effect, programmable shaders provide insight into the hardware of graphics processing units.

One problem with teaching shader-based computer graphics is that a lot of technical and potentially boring tasks lie in the hand of the programmer: loading and compiling shader programs, storing and managing buffer objects, defining transformations by means of matrices and linear algebra operations, etc. Existing approaches thus provide libraries for facilitating some of these tasks [2, 1, 3].

2. CONTENT OF THE POSTER

The poster presents a scene graph library which is fully based on the programmable rendering pipeline. It uses the OpenGL 3.2 core profile, which does not allow deprecated fixed-function functionality [9]. The teaching approach combines the high-level abstraction of a scene graph with the low-level programming of shader cores, which are attributed to scene graph nodes (cf. figs. 1 and 2).

Scene graphs as high-level graphics approaches allow struc-

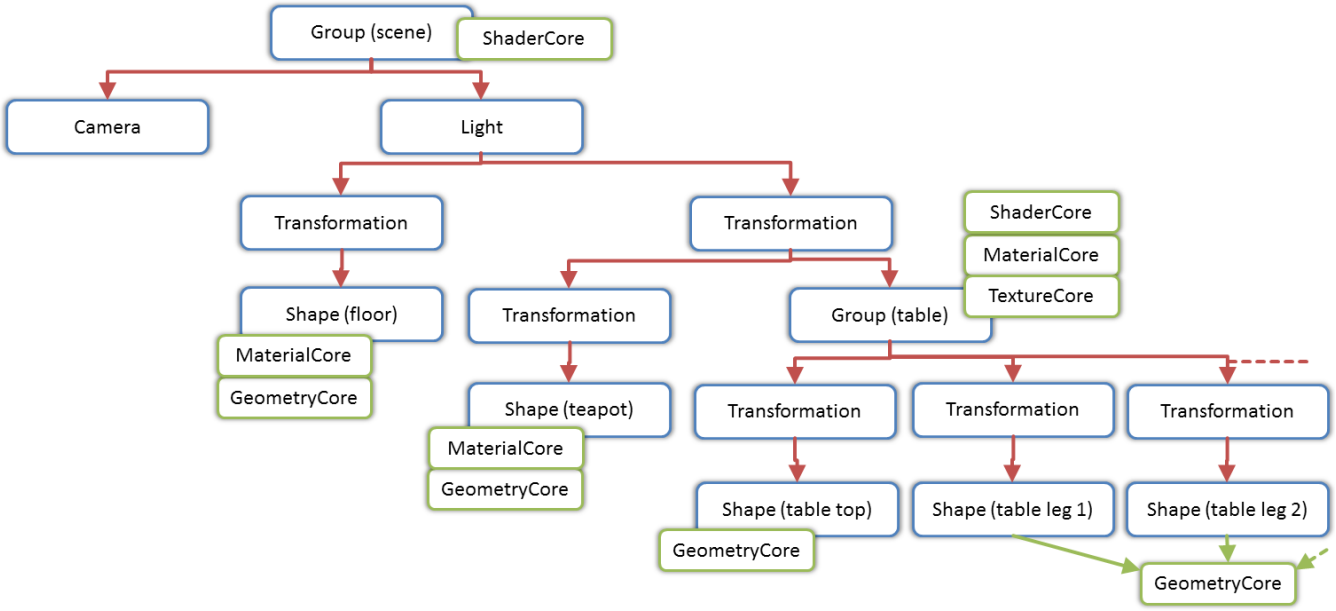

Figure 1: Sample scene graph consisting of nodes and cores.

Figure 2: Sample scene with pixel-based Phong shading corresponding to scene graph of fig. 1.

tured geometric modeling, make use of various design patterns, and offer a possibility to review and apply tree and graph data structures and algorithms. Existing scene graph libraries like OpenSceneGraph [7], however, are very complex and difficult to comprehend by students. A further danger in project-oriented laboratory classes is that students may look for existing code in the internet instead of trying to find their own solutions.

The presented scene graph library has a simple and clear structure that can be explained in a 90 minute lecture unit. It is extensible in order to let students implement advanced concepts taught in the lecture, like shadows or particle systems. It uses modern technologies including C++11, GLFW [5], and GLM [6]. Finally, the poster will present code samples, results of student projects, and student evaluation results. The code of the scene graph library will be published as an open-source project.

3. ACKNOWLEDGMENTS

The author gratefully acknowledges fruitful discussions with Ingo Ginkel, Frauke Sprengel, Henrik Tramberend, and numerous students of his computer graphics classes.

4. REFERENCES

[1] E. Angel. Teaching computer graphics starting with shader-based OpenGL. In P. Cozzi and C. Riccio, editors, *OpenGL Insights*, pages 3–16. CRC Press, Boca Raton, FL, 2012.

[2] E. Angel and D. Shreiner. Teaching a shader-based introduction to computer graphics. *IEEE Computer Graphics and Applications*, 31(2):9–13, 2011.

[3] M. Bailey. Transitioning students to post-deprecation OpenGL. In P. Cozzi and C. Riccio, editors, *OpenGL Insights*, pages 17–26. CRC Press, Boca Raton, FL, 2012.

[4] D. Blythe. The Direct3D 10 system. *ACM Transactions on Graphics*, 25(3):724–734, 2006.

[5] GLFW. Retrieved November 20, 2013 from `http://www.glfw.org/`.

[6] GLM: OpenGL Mathematics. Retrieved November 20, 2013 from `http://glm.g-truc.net/`.

[7] OpenSceneGraph. Retrieved November 20, 2013 from `http://www.openscenegraph.org/`.

[8] M. Segal and K. Akeley. The OpenGL Graphics System: A Specification (Version 3.0). Technical report, Khronos Group, 2008. Retrieved November 20, 2013 from `http://www.opengl.org/registry/doc/glspec30.20080923.pdf`.

[9] M. Segal and K. Akeley. The OpenGL Graphics System: A Specification (Version 3.2 (Core Profile)). Technical report, Khronos Group, 2009. Retrieved November 20, 2013 from `http://www.opengl.org/registry/doc/glspec32.core.20091207.pdf`.