Practically Oriented Introduction to Computer Graphics

– Schedule –

1. Introduction

In this lesson, the participants will get a general overview of the knowledge domain of computer graphics and its various fields of application. Thereby at first, basic terms like ’graphics’, ’computer graphics’ and ’rendering’ will be explained.

A historical outline – heading back to 1950 and presenting important milestones of the field of computer graphics from that time until today – will show the development of this subject to an independent discipline of computer science.

The lecture is assigned to the branch of generative computer graphics, but also intersections with other scientific disciplines such as image analysis and image processing are taken into consideration.

The last part of the lecture will focus on the necessity and on contents and applications of the most important standards in computer graphics. Thereby the historical development of the OpenGL standard will be introduced among others.

2. Basic concepts of OpenGL

The technical and methodical basis of this lesson will be the industrial standard OpenGL. While its commonalities and differences with other standards will be pointed out, certain examples demonstrate to the participants that the algorithms and processes being conveyed in the lecture are basic for all current standards for rendering of three-dimensional scenes i.e. also for DirectX.

The introduction to OpenGL will be top-down. Beginning with the general structure of the processing scheme of OpenGL, the central instrument of the rendering pipeline will be explained. The definition of important terms such as ’context’, ’GL status’ and ’shader’ shall lead to an extensive discussion of functions and structures of the different levels of the GL pipeline. Due to the introduction of the state machine, the quasi- sequential processing – from 3D geometries up to pixels in the framebuffer – will be explained from a global view.

3. The Template Program

OpenGL does not provide any support for the implementation of the user interface. For instance, no functions are given for the handling of windows, dialogue elements or for the recall of the keyboard or the mouse, as rather these tasks lie in the responsibility of the user. Accordingly to that, depending on the actual operating and window system, the approaches to the programming of a

framework clearly differ. But this is not the issue of this lesson.

For this reason, a template program in which the tasks above are provided is available and can be downloaded for the experiments made by the participants with OpenGL. The particular structure and functionality of this program will be presented in this lesson. All the following lectures and exercises will be based on this template. Besides the use of GL Code for the drawing of geometry, the programming of keyboard and mouse interactions or starting points for the realization of animations (e.g. counter variables or the access to frame counters) will be explained.

4. OpenGL Primitives

Each graphical scene is based on fundamental description elements, which for OpenGL applications are the so-called Vertex Data. Building up on the basic concepts that will already have been introduced, the central function of vertices in OpenGL will be explained. The discussion of the simple primitive surfaces (triangles, quadrilaterals) will be followed by the introduction of the GL polygon. Thereby, the limiting conditions that OpenGL puts on the specifications of polygons will comprehensibly be justified from the viewpoint of a simple pipeline design that can easily be translated into hardware.

Geometrical basic figures such as globe, cube or cone and curved surfaces are not being standardized in the GL core, but are being provided by the Utility Libraries. The discussion of these primitive characters such as quadrics and NURBs is targeted at the quick prototyping of graphical scenes.

5. Combined Primitives, Arrays and Buffer Objects

Building up on the basic GL primitives, the use of combined primitives for the drawing of complex polygonal surfaces is motivated by certain efficiency considerations. In this lesson, the application and efficiency as well as the restrictions of the standardized OpenGL fans and strips shall be presented.

Combined primitives are only one of two possible solutions for the efficient upload of geometrical data to the GL pipeline. The other, even more efficient way, is represented by the so-called Vertex Arrays (VA) or Vertex Buffer Objects (VBO) between which the GL differentiates. This lesson shall familiarize the participants with the basic concepts and program-specific technical applications of VA and VBA.

6. Transformations

Transformations are, apart from primitives and attributes, the most important basis for the realization of graphical models. In this lesson, the transformation chain of OpenGL will be illustrated. Based on mathematical fundamentals such as affine transformations and practical application sceneries, the different steps towards transformation by MODELVIEW, PROJECTION and VIEWPORT are being presented and their programming will be explained. Thereby, an important

aspect is the positioning of geometrical components in the 3D scene through manipulation of the GL coordinate system (coordinate transformation). The second aspect hereby affects the implementation of a camera model (perspective, projection).

In addition to these basic functions for setting and influencing the transformation matrix for geometry and camera, which is being stored in the Server State, particularly the matrix stack as a tool for the control of complex transformation chains will be presented. Finally, techniques for camera movements and animations – in terms of accumulated transformations – will be demonstrated on the template.

7. Scene Graph

With the knowledge gained from the lessons 1-6, the participants shall basically be able to write a 3D OpenGL application. But in order to operate with bigger models or programs, suitable concepts are needed. Generally, the means for this are scene graphs. OpenGL, however, does not offer any functionality for this purpose.

As an appropriate structuring of big models is essential for a successful technical implementation with OpenGL, the participants will be familiarized in this lesson with the basic concepts of a scene graph. Terms such as ’root node’, ’structures’, ’spatial location’ and ’inheritance’ are being introduced and, based on examples, useful basic approaches will be conveyed. A special attention is also paid to the structuring of the geometrical transformations in the scene as well as to their functional realization with the help of the OpenGL transformational concepts and the matrix stack. Independent from OpenGL, the concept of the scene graph will also be regarded under aspects of distributed implementation, efficiency considerations and commercial libraries.

8. Graphical Exchange Formats

OpenGL represents a versatile and efficient rendering interface to the graphical hardware. The previous lessons will have conveyed the basic concepts for the rendering of three-dimensional geometry. But OpenGL does not provide any functionality for modeling. Demanding, detailed and animated 3D scenes are usually created with specialized programming systems, the so-called modelers. With the help of transfer formats, the graphical models that have been created can be interchanged between the programming systems and can be imported into self-generated programs.

In this lesson, at first the problem of the exchange of graphical data is basically discussed. This also focuses on the aspects of the incompatibility of certain modeling techniques and proprietary file formats.

In the second part of this lesson, the exchange format WaveFront will be presented. The explanation of the syntax will be followed by a draft of useful data structures as well as of an import function for WaveFront files, the content of which shall then be rendered by the means of OpenGL. The programming of the rendering function – with the aim of deepening the knowledge from lessons 3-5 –

takes place with simple GL primitives and arrays as well as with the buffer concept (VBO).

9. Lighting with OpenGL

The discipline of computer graphics makes use of lighting models and texturing methods in order to increase the closeness to reality of the rendering. In this lesson, basic aspects of lighting computation will be presented. Firstly, global and local lighting models shall be contrasted. Thereby, the focus will be on the different possibilities such as reflections, shades or the physical correctness as well as on the implementation (algorithms, requirements, operating times) on the software side. Illustrating images will be available as examples.

In the second part of the lesson, the local lighting model of OpenGL will be demonstrated by vertex lighting and its application will be comprehensively described. An important component for the calculation of the light situation at vertices is the normal vector. Its calculation for the different primitives and for the WaveFront loader will be explained.

10. Texturizing

Textures are inevitable for current graphical programs. Aside from the lighting, their various forms of application in the rendering process offer closeness to reality of the computed images.

With its current shader concepts and extensions OpenGL provides an extensive support of advanced texturing processes, including multi-texturing functions. To deal with all these concepts would go beyond the scope of this lecture. For this reason, the basic technique of the application of image textures will be conveyed and demonstrated on practical exercises in combination with the OpenGL lighting system.