Learning OpenGL and shader programming

Irvin Stevic*
Supervised by: Dr. Jasminka Hasic Telalovic†

Faculty of Engineering and Natural Sciences
International University of Sarajevo
Sarajevo / Bosnia and Herzegovina

Abstract

OpenGL is an API(Application Programming Interface) used for rendering 2D and 3D computer graphics. It can interact with the GPU and allows shader programming with the OpenGL Shading Language (GLSL). This paper serves as a summary of a student's experiences while studying an introductory computer graphics course. The paper will cover the contents of the course, starting with the basics of computer graphics and eventually moving to shader programming, describe the challenges and issues that appeared and how they were overcome. Examples, codes and outputs related to the lessons will also be included.

Keywords: OpenGL, shaders, programming, glsl

1 Introduction

The introduction of computer graphics courses to computer science degree programs was greatly helped by the development and spread of available graphics hardware. When mainstream machines were able to process graphics, it allowed computer graphics to become more widely accepted. Availability of graphics APIs, like OpenGL, meant that almost everyone could use it and program graphics in a higher level language. These advances and developments led to the introduction of computer graphics courses into the degree program for computer science majors.

Today most of universities offer an introductory course to computer graphics that all the computer science majors can take, as well as some advanced courses for those whishing to specialize further in the graphics field. This paper is about the undergraduaduate Computer Graphics course taught at IUS in Spring 2011/2012. Two main approaches to teaching this type of courses are top-down and the more traditional bottom up. The top-down approach which was used for this course starts teaching an overview of the system and then moves to each subsystem for detailed analysis.

Purpose of this paper is to describe the experiences of computer science students taking an introductory course in computer graphics. The paper could be used by professors so they can see how the teaching methods used reflected by the students, which were the biggest challenges and the best ways for students to master OpenGL and shader programming. It could also benefit other students taking a similar course, so they can see an approach that might be different than theirs.

2 Related work

First introduction of Computer Graphics (CG) topics within Computer Science (CS) undergraduate programs came in late 1980s [7].

The basis for creating contemporary Computer Science curricula is defined in [8]. Within the undergraduate curriculum, 14 knowledge focus groups are identified. The one that encompasses Computer Graphics education is named Graphics and Visual Computing (GV). CG was clearly outlined in this report as the new field whose topics need to be expanded within the undergraduate curriculum (through topics such as Graphics and Multimedia, and Human-computer interaction). Furthermore, the following topics are identified as relevant with GV: fundamental techniques in graphics (core), graphic systems (core), graphic communication (elective), geometric modeling (elective), basic rendering (elective), advanced rendering (elective), advanced techniques (elective), computer animation (elective), visualization (elective), virtual reality (elective), computer vision (elective), Computational Geometry (elective) and Game Engine Programming (elective) [8]. From the four major directions of Computer Science undergraduate program, CG undergraduate course was identified as required for the two of them (1. A system-based approach and 2. A web-based approach). In addition, the following advanced CG courses are identified: Advanced CG, Computer Animation, Visualization, Virtual Reality, Genetic Algorithms. It was suggested that CG courses are taken in the second year (out of three that require CS classes). Also, the need to introduce GPU and its ability to accelerate performance in computer graphics was outlined as a topic in required Computer Architecture course.

In 2008 an update for CS program was given in [9].

*irva.stevic@gmail.com
†jhasic@ius.edu.ba

One of the reasons that initiated this update was the introduction on three major new focuses in undergraduate CS education, one of them being on games and entertainment software. This gave CG greater relevance within the CS program.

At International University of Sarajevo there are three elective courses from GV knowledge focus group: CS405 Computer Graphics, CS414 Computer Vision, CS445 Human Computer Interaction [6]. Each of these courses is worth six European Credit Transfer System (ECTS) credits out of 240 ECTS needed to graduate with the undergraduate degree in computer science. The CS405 CG course has the following prerequisites: Discrete Mathematics, Linear Algebra, Algorithms and Data Structures, Advanced Programming and Introduction to Programming [6]. Out of the CG undergraduate course topics proposed in [8], the course aimed at covering the following ones:

- Graphic systems: Raster and vector graphics systems; video display devices; physical and logical input devices; issues facing the developer of graphical systems

- Fundamental techniques in graphics: Hierarchy of graphics software; using a graphics API; simple color models; homogeneous coordinates; affine transformations; viewing transformation; clipping

- Graphical algorithms: Line generation algorithms; structure and use of fonts; parametric polynomial curves and surfaces; polygonal representation of 3D objects; parametric polynomial curves and surfaces; introduction to ray tracing; image synthesis, sampling techniques, and anti-aliasing; image enhancement

The rest of the CG topics proposed in [8] are covered in other CS courses within the program.

Introducing shaders programming in CG education is described in [5, 2, 4] and more recently in [1] at introductory level and in [3] on intermediate level.

3 Contents of the course

The course was given in the Spring 2011/2012 semester at the International University of Sarajevo, and the textbook used for the course was [1]. The grading was done based on assignments, two exams and student participation in the classes.

Programming assignments were made of two parts, one part which was done in labs with the help of the instructor, and the second part was done as homework individually by the students. The assignments were related to the lectures and the chapters covered that week. Some of the assignments were programming a random maze generator and making a 3D object that responded to various user inputs. For the final assignment, we used a camera and did a brief project related to HDR (High Dynamic Range) photography.

The course was started with some basic definitions and introduction to computer graphics. Applications, history, the basics of how imaging systems and display devices work were covered, as well as hardware architecture of graphic processors and the graphics pipeline. All of these basics helped us comprehend and understand what was following in the course.

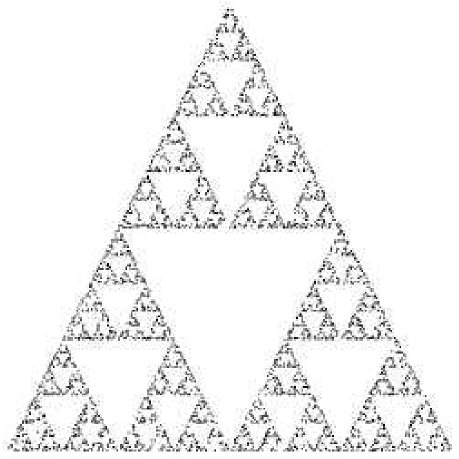

Figure 1: The Sierpinski Gasked generated with 5000 random points

After the introduction, we moved to OpenGL programming. We used Microsoft Visual C++ IDE, with the GLUT (OpenGL Utility Toolkit) which enabled easy writing of OpenGL programs. Both the Visual Studio(Express version) and GLUT were freely available for download on the internet, so there was no need for us to spend any money on software. Since the course was based on a top-down approach, we started with a premade code that drew a Sierpinski gasket, which can be seen in Figure 1. At first the code was overwhelming but after going through it a few times with the help of the instructor, it all started to make sense. Using this code we learned about the coordinate system used in OpenGL, polygons, triangulation, approximating curved objects and simple coloring functions.

After the Sierpinski gasket, we moved on to simple 3D primitives and objects and used a sample code for a colored cube. Using this code we learned about the 3D coordinate system, modeling the faces of the cube and coloring the faces differently. After we finished with a simple program that just diplayed the cube, we added some functionality to the program, so the user could interact with it and also perform transformations on it. The program allowed the cube to be moved, rotated and scaled. Using this new program we learned the basics of transformations.

Although the translation, rotation and scaling can be done using C++ coding and OpenGL functions, shader programming and GLSL were introduced at this point. We learned how to pass information to the shader and GPU, as well as how to process them using GLSL. In the end, the final results, computed using the GPU, were displayed. For exercise and as an assignment, we wrote programs that implemented rotating, scaling and translating using both OpenGL functions as well as GLSL. Although the end result looked the same, we were able to see the difference between programming the shaders and just passing already computed data to the shaders.

From this point, the course focused on viewing and ways of displaying the scene. Differences between different types of viewing were covered, as well as their representation in OpenGL. Camera position as well as different projections were explained and covered. Near the end of the course, we learned about the lighting techniques, light sources, reflection models and shading.

Finally, we covered the field of HDR photography. Although brief, this section introduced the basics of HDR photography and some of its applications. We also used a camera and created some of our own HDR photos by using Adobe Photoshop in the computer labs.

4 Assignments

Perhaps the most important part of the course were the assignments. Most of the assignments involved and were focused on programming and the final one was oriented toward photography. The assignments gave us the opportunity to apply what we learned and see how it all comes together. Programming was done in C++ with GLUT, and in total, there were five major assignments.

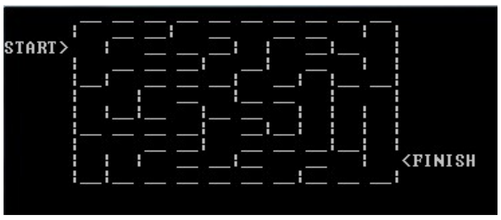

Figure 2: Random maze drawn using simple outputs

4.1 Random maze generator

The first assignment that we were given was programming a random maze generator using C++. The program was to randomly generate a maze of user selected size that would then be drawn using simple line characters such as underscore and vertical line. Although the program was written using only C++ functions, it would later be used for a more complex 3D assignment. This program served as a good starting point and a good way to get familiar with C++ and Microsoft Visual Studio, especially for students that didn't have a chance to work with it before. The result can be seen in Figure 2. Also, this assignment was a good test of student knowledge from prerequisite classes (Data Structures and Algorithms and Advanced Programming).

4.2 Turtle graphics

The second assignment involved making an API of graphics functions, which would use an object to draw various shapes using API functions. The program required implementation of functions that imitated the movements of a turtle with a pen attached. The turtle would turn in place and then move forward, the process would be repeated until the desired shape was finished. The functions implemented were turning left and right by a certain degree and moving forward a certain length. The program also allowed the pen to be lifted up in order to move the turtle to a new position without a line trail. This assignment allowed us to get familiar with the coordinate system and basic of drawing primitives.

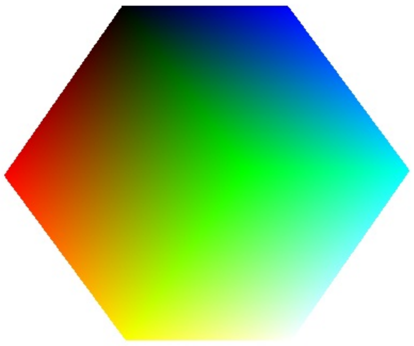

Figure 3: Modeled 3D cube using shader programming for rotation, translation and scaling

4.3 Cube transformations

After the turtle graphics assignment, we were given the task of modeling a 3D cube that responded to user input, so that it could be rotated, zoomed in or out and moved. This task was to be accomplished in two ways, the first one using OpenGL functions for rotation, scaling and translating, and also by implementing those functions in GLSL and programming the shaders directly. The user could interact with the cube using combinations of mouse movements and button clicks. Directions and length of mouse movement determined the direction and degree of transformation while mouse buttons determined what type of transformation was being performed(e.g. left-click for rotation). The final output of this assignment is shown in Figure 3. This assignment was the first encounter with shader programming, and although the language was relatively easy to get used to, the biggest challenge was sending of data to the GPU for processing. The functions used to accomplish this looked messy and complex, and it took some time to get comfortable with using them.

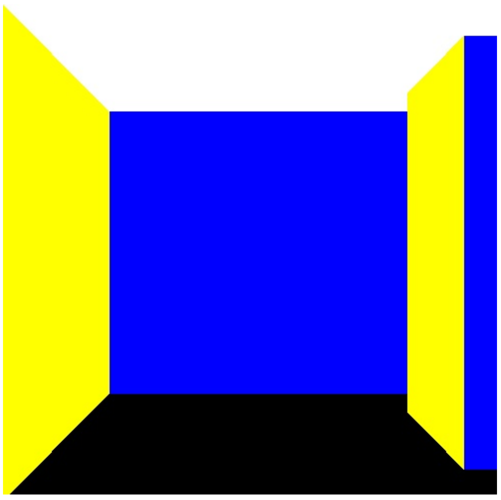
Figure 4: Interactive 3D maze with different colored walls for easier differentiation

4.4 Interactive 3D maze

The next programming assignment was to use the random maze generator from the first assignment to generate a 3D maze using OpenGL. First part of the assignment was to display the 2D maze using primitives and then modify it to 3D and upgrade it to so it could be navigated and explored. This task helped us understand the basics of viewing and camera positioning, since the maze needed to be interactive. The result of this assignment is shown in Figure 4. This assignment was probably the hardest of all since the viewing functions were challenging and hard to use in the code.

4.5 HDR photography

The final assignment was related to HDR photography. We needed to write a short essay on the topic and include photos that we took, as well as the produced HDR photo that we made using Adobe Photoshop. This assignment was a nice change of routine of programming and allowed us to explore this interesting field more on our own, as there was not much time for it during the classes. The process of finding good scenes and taking the pictures proved to be the most interesting since we spent some time with the instructor outdoors in a more relaxed atmosphere. Our end results looked great and this proved to be a great experience. One of the results is shown in Figure 5.

5 Course Evaluation

The course was a good addition to the Computer Science degree and gave us the opportunity to learn about the graphics and the basics of how the stunning graphics we see in video games are created. Although most of the students were not interested in exploring this topic in depth and specializing in computer graphics, it still proved to be beneficial for all.

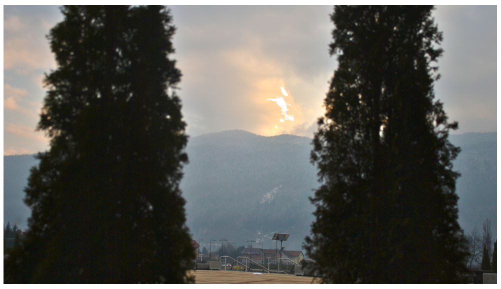
Figure 5: A HDR photo created from multiple LDR photos

The best part of the course was its diversity, and covering different aspect of CG including software as well as hardware principles and HDR photography. This way, the course remained interesting and kept our attention. A lot of exercises and practical examples helped us understand the theoretical concepts we studied and make sense of it all. The tutorials gave us the opportunity to work on the exercises with the instructor, discuss different ideas and compare them with our peers.

Unfortunately, when the course was offered, it did not get much attention from students since the majority considered it to be too difficult. As a result we ended up with a small group of students, but all of us that took the final exam, passed the course. The pre-requisites for this course were Linear Algebra and Algorithms and Data Structures courses [6]. Linear Algebra course prepared us for the matrix operations used in transformations as well as usage of homogenous coordinates, and after a short review all of us were comfortable with matrix operations. Unfortunately, not many of the previous courses used C++ for programming(but Java) and some of the students were not familiar with it. However with some help from the instructor, soon all of us were able to start programming in it. The previous courses involving math and programming courses, were of great help in learning computer graphics and all of its aspects.

The computer graphics course itself was very interesting and well adjusted for everyone able to take it. It was very well organized and was not extremely difficult, but still managed to cover all the main areas and give us a good basis of computer graphics knowledge to build upon. Top-down approach was well suited for our needs and allowed us to start writing some programs and experimenting as soon as possible, which was much better, since most of the students were more programming oriented.

6 Conclusions

This paper summarized an introductory computer graphics course, and told the story of learning computer graphics basics from a student's perspective. The course used already written codes and entire programs to teach us how to use OpenGL and how to apply the theory we studied. Although different approaches probably work better or worse depending on the students, the approach used in this course helped us easily master computer graphics basics and prepared us for advanced computer graphics courses.

References

[1] Angel E. and Shreiner D. *Teaching a Shader-Based Introduction to Computer Graphics*. Computer Graphics and Applications, IEEE, 2011.

[2] Owen G.S. and Zhu Y. *Teaching programmable shaders: lightweight versus heavy-weight approach*. SIGGRAPH 2005 Educators program, 2006.

[3] Fink H., Weber T., and Wimmer M. *Teaching a Modern Graphics Pipeline Using a Shader-based Software Renderer*. Eurographics 2012 Education program, 2012.

[4] Talton J.O. and Pitzpatrick D. *Teaching graphics with the OpenGL shading language*. ACM SIGCSE Technical Symposium on Computer Science Education, 2007.

[5] Bailey M. and Cunningham S. *A hands-on environment for teaching GPU programming*. ACM SIGCSE Technical Symposium on Computer Science Education, 2007.

[6] Haris Memic, Rasit Koker, Emir Karamehmedovic, Kanita Hadziabdic, Akif Yaman, and Alma Husagic-Selman. *COMPUTER SCIENCE AND ENGINEERING PROGRAMS*. International University of Sarajevo, 2012.

[7] Ohlson M.R. *The Role and Position of Graphics in Computer Science Education*. SIGCSE Technical Symposium on Computer Science Education, 1986.

[8] The Joint Task Force on Computing Curricula. IEEE Computer Society and Association for Computing Machinery. *Computing Curricula 2001 Computer Science*. IEEE CS and ACM, 2001.

[9] The Joint Task Force on Computing Curricula. IEEE Computer Society and Association for Computing Machinery. *Computer Science Curriculum 2008:An Interim Revision of CS 2001*. IEEE CS and ACMl, 2008.