http://dx.doi.org/10.18623/rvd.v22.n2.3462

METHODOLOGY AND TECHNIQUES USED IN THE DEVELOPMENT OF A GRAPHICS LIBRARY WITH OPENGL

METODOLOGIA E TÉCNICAS UTILIZADAS NO DESENVOLVIMENTO DE UMA BIBLIOTECA GRÁFICA COM OPENGL

Article received on: 6/23/2025
Article accepted on: 9/29/2025

Tzvetomir Vassilev*
***University of Ruse (UR)
Ruse, Bulgaria
Orcid: https://orcid.org/0000-0002-0067-7316
tvassilev@uni-ruse.bg

Svetlozar Iliev**
****SIRMA OOD
Ruse, Bulgaria
svetlozar.iliev@sirma.bg

The authors declare that there is no conflict of interest

Abstract
This paper describes and reviews the development stages of a graphics library, called by its developers GFX, and also summarizes its main characteristics. This paper also describes the core techniques used in the development stages of such library and all additional libraries required for the completion of the target functionality. It covers all aspects and requirements of a traditional graphics library as well existing solutions and their applications in the industry. The paper's main objective is to explain and lay down the basics of a traditional computer graphics system as well as specifics regarding the actual library implementation. The authors of this project were motivated by the potential acquisition of knowledge and experience in this field and also the chance of developing a library which addresses some of the most common shortcomings in present day state of the art graphics libraries.

Keywords: Computer Graphics. OpenGL. Linear Algebra. Rendering & Optimisation Techniques.

Resumo
Este artigo descreve e revisa as etapas de desenvolvimento de uma biblioteca gráfica, chamada por seus desenvolvedores de GFX, e também resume suas principais características. Este artigo também descreve as principais técnicas utilizadas nas etapas de desenvolvimento dessa biblioteca e todas as bibliotecas adicionais necessárias para a conclusão da funcionalidade desejada. Abrange todos os aspectos e requisitos de uma biblioteca gráfica tradicional, bem como soluções existentes e suas aplicações na indústria. O principal objetivo do artigo é explicar e estabelecer os fundamentos de um sistema de computação gráfica tradicional, bem como as especificidades da implementação da biblioteca. Os autores deste projeto foram motivados pela potencial aquisição de conhecimento e experiência nesta área, bem como pela oportunidade de desenvolver uma biblioteca que solucione algumas das deficiências mais comuns nas bibliotecas gráficas de última geração.

Palavras-chave: *Computação Gráfica. OpenGL. Álgebra Linear. Técnicas de Renderização e Otimização.*

1 INTRODUCTION

The paper aims to describe the stages of developing a typical graphics library similar to well-known libraries such as Ogre3D, Irrlicht, Horde3D, Unity, Unreal, etc., as well as to summarize its important characteristics. The idea of creating such a library was born for several reasons, some of which are the potential acquisition of more detailed knowledge in the field of computer graphics and the creation of an atypical, flexible library that solves some of the main problems present in the above-mentioned existing solutions to date.

1.1 Origin and purpose of graphic libraries

The term graphics library (or graphics engine) is a common part of a larger unit known in the world of 3D graphics as a “game engine” (GREGORY, 2019) and originated in the mid-1990s in connection with one of the most famous games in the so-called “first person shooter” genre, called “Doom”. This is one of the first popular games known to the market, in which a visible boundary or division is clearly built between the main core and the components of the graphics library. The importance of this division quickly became apparent when developers began to reuse existing graphics libraries without any or minimal changes together with other components to create completely new and different virtual games and graphics applications. This also gave rise to the so-called “mod” studios, which created new games using and modifying existing ones. Games like Quake III Arena and Unreal are typical examples of virtual games created with the idea of being easily modified and flexible, using scripting languages to manage the internal logic of the game without the need to change the underlying components (part of which is the graphics library) for their creation.

1.2 Structure of graphics libraries

At its core, a graphics library consists of several core components or layers of abstraction (GREGORY, 2019) that work together to efficiently, realistically, and optimally achieve the ultimate goal of rendering computer-generated images.

- *Layer 1* – the lowest level of abstraction in a graphics library, this is the level that communicates and works directly with the hardware or with existing Graphics APIs such as OpenGL, DirectX, Vulkan, which take care of communication with the graphics driver.
- *Layer 2* – an intermediate layer that wraps the first layer of abstraction and serves to remove or hide specific limitations and features existing in the used API, driver or layer of the previous level. This level allows the library to become independent of the type, features, limitations or peculiarities of the used graphics pipeline and hardware. This level builds the basic functionalities underlying the graphics library such as creating, deleting, copying video buffers, textures, shaders, graphics contexts, interface buffers, graphics layers, hardware requests, etc.
- *Layer 3* – this level is characterized by high abstraction, it usually includes functionalities that the end user of a graphics library works with. Most often, this layer is characterized by the fact that it contains a much larger number of functionalities compared to the lower-level layers. This layer encompasses elements for visualization and importing objects, lighting, shadows, texturing, 3D selection, serialization, particle simulation, fluid simulation, removing invisible objects, transparency, changing projections, applying filters, input and output management, etc.

2 REVIEW OF EXISTING SOLUTIONS

Nowadays, there is a large set of graphics libraries. They are practically divided into two main categories. On the one hand, there are libraries that are quite flexible and their advantage is that they provide only the basic graphics functionalities that can be easily extended and customized in any direction. A major disadvantage of this type of libraries is that they require a large set of knowledge to be used or modified. Another disadvantage is the amount of time required to build a minimal, functioning project. Such libraries often require users to have deep programming knowledge, they do not offer an easy and convenient way of interaction, such as a graphical interface, and they often do not include additional systems such as an audio system or a physics simulation system. A few examples of such libraries are:

- *Ogre3D* is a basic graphics library that provides the essential elements needed for the graphics pipeline to work. Ogre3D offers various methods for visualization and scene management including spatial division, paging, and portals. Another characteristic feature is the ability to work with dynamic level of detail, animations, working with shaders in different languages (GLSL, HLSL, Assembly, Cg)
- *Cocos2d* is an open-source library that offers functionalities for modeling, visualizing and simulating two-dimensional graphical environments. It is known for being one of the libraries with a minimal set of features, but also one of the fastest to date. It allows developers to create projects for any device, both mobile and desktop. Its speed allows developing applications for devices with weaker hardware compared to modern standards.
- *Irrlicht* is an open-source library offering the largest set of functionalities compared to Ogre3D or Cocos2d. It is characterized by being relatively small, which allows for easy and quick mastering, it is supported by a large community and is free for a large set of programming languages such as C++, Java, Lua, Pascal, Basic, Python, etc. Irrlicht supports OpenGL, DirectX 8, 9, and 11, as well as shaders for different languages (GLSL, HLSL). Compared to its rivals, it offers not only a set of graphics functionalities, but also the ability to read, write and play a large set of audio, video formats. Irrlicht also includes a physics simulation module.

At the other end of the spectrum, there are libraries that include a huge range of functionalities, enabling rapid development of projects of any scale. The advantage of these is that they require significantly less knowledge in the field of computer graphics, the user usually interacts with them through a graphical user interface, and often do not require direct knowledge in the field of programming. Another important advantage is that they often include not only a set of methods for graphic visualization, but also other elements such as an audio system or a physics simulation system. The disadvantage is that they cannot be easily customized or modified for work requiring specific needs or techniques (ANDRADE, 2015).

- *Unreal Engine* is one of the most successful and most famous libraries of developers from Epic Games, which entered the Guinness Book of Records. It is intended for the development of large projects. A characteristic of Unreal Engine is that it supports multiple platforms such as Windows, Mac, Linux, iOS, Android, Playstation, Xbox, etc. It is widely used by famous companies for the development of games and graphic applications, some of which are Capcom, Activision, Ubisoft, Microsoft Studios, Nintendo, etc.

- *Unity* is another widely used library, which is mainly used and popular in the circles for creating mobile graphics applications or games. It is adapted for easy and quick development and use, both by professionals and novice developers. It supports the development of two-dimensional and three-dimensional graphics applications, has a flexible modeling and animation system. Like its rival Unreal Engine, Unity is a multi-platform engine developed for Windows, Mac, iOS, Android, Playstation, Xbox, Windows Phone, Tizen, etc. Some of its most famous users are Electronic Arts, LEGO, Ubisoft, Square Enix, etc.
- *CryEngine*. Like the aforementioned giants in the industry, CryEngine is no slouch in terms of the available set of functionalities it offers. One of the great advantages of this engine over its competitors is that its developers offer its full source code completely free of charge. It is also characterized by its multiplatform support for iOS, Android, Windows, Linux, Playstation, Xbox, and Wii. Famous companies such as Poppermost Productions, CI Games, Obsidian Entertainment are its users.

3 DESIGN OF THE GRAPHICS LIBRARY

3.1 Main objectives

The main objective of this work is to develop a graphics library for the purpose of training students, which can also serve as a basis for developing a complete library for real-time simulation. Another goal of the development is to introduce modern techniques for creating computer-generated images in real time (AKENINE et al, 2018), optimizations, etc. This library should possess the following basic functionalities:

- Reading external files containing graphic models or elements;
- Reading external files containing graphic images, textures, atlases, etc.;
- Reading, compiling and executing shader programs contained in external files. Loading data into variables and shader programs;

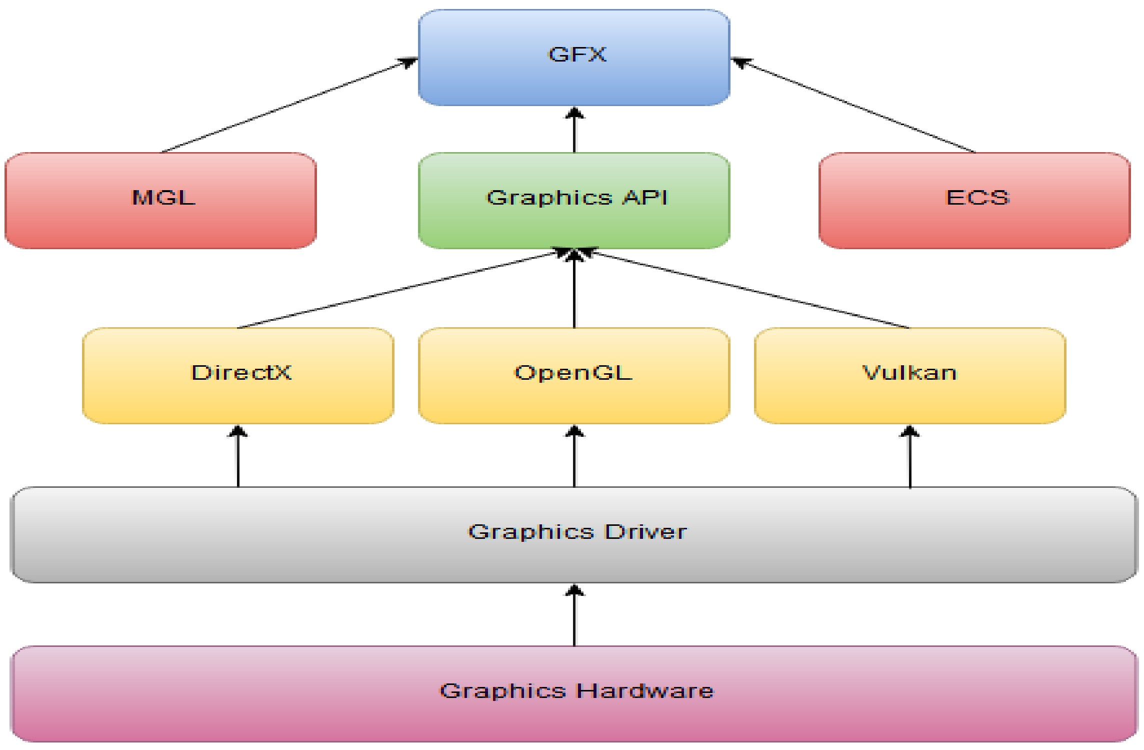
- A set of different options for visualizing images and geometric elements in real time;
- Image filtering and processing: smoothing, brightening, scaling, enhancing high and low frequencies, adjusting contrast, etc.;
- Working with dynamic geometric level of detail for graphic models
- Support for basic rendering techniques: lighting, texturing, shadows, rendering, excluding invisible objects;
- Support for advanced techniques for rendering and simulating ocean, atmosphere, clouds, environment, etc.;
- Dynamic real-time interaction with a virtual scene: selecting, transforming and processing objects;
- Recognition of signals and commands from a set of peripheral devices: mouse, keyboard, joystick, trackball, etc.;
- Serialization and deserialization of a virtual scene on an external medium or permanent memory.

3.2 Design

The graphics engine, described in this work, is a set of three independent libraries that are developed and designed to work together to implement the main aim and objectives of the project. It is planned that the library develops its own API and its own design of the modules, without relying on existing libraries, as far as possible. Figure 1 systematizes the structure of the library and the modules working together to implement the set goal. A detailed description and introduction to the three main modules of the graphics library follows.

Figure 1.

A structural diagram of the GFX library



- **MGL (Math for Graphics)** is a library designed to work with mathematical operations. It is the first of the three main libraries, which aims to provide easy work with basic mathematical constructs. CORE is the main module, containing the basic constructs such as: vectors, matrices, quaternions, while the STRUCT module contains a large set of three-dimensional primitives and their two-dimensional equivalents (sphere, cone, parallelepiped, cube, ray, planes, truncated pyramid, etc.), as well as structures for spatial partitioning. The library supports all commonly used methods for implementing basic mathematical operations (vector and dot product, matrix transformation, rotation and interpolation between vectors and quaternions, normalization, transposition, determinant and finding the inverse matrix, etc.), located in the FUNCTIONS module, as well as a large set of methods for testing for intersection between basic geometric primitives, such as sphere, parallelepiped, plane, ray, truncated pyramid, etc., located in the Geometric Test module. An important component of this library is the structures for “accelerating” the management and processing of geometric primitives. The library supports two such structures called: Octree and Point Cloud Tree, located in the Space Partition module. An Octree is a special type of tree in which each parent node has eight children. Each node is a cube with dimensions of powers of 2 (4, 8, 16, 32, etc.) and is divided into 8 identical cubes, which become its descendants. Each node stores its descendants and a set of primitives inscribed in it, this type of structure is often used to remove invisible objects from dynamic scenes. Another important module that MGL offers is the API module, providing a set of programming interfaces. There are several main books about math for computer graphics (DUNN, et al, 2011, LENGYEL, 2012).

- **ECS (Entity Component System)** is the second main library, providing an implementation of an architectural pattern often used in the development of virtual games and graphical applications. This pattern is characterized by the fact that it emphasizes the use of composition over inheritance, which provides great flexibility. It consists of three elements called Object, Component and System. A Component in this pattern is a simple minimal structure in which only data is stored, e.g. a component describing the position of an object would be a structure containing only one member - a 3D vector describing the position of the object in space. Each component has a unique identifier or number and describes a certain characteristic or action. Each Object is a unique identifier or mask, which is obtained based on what components are bound to it. Depending on what components a given object is bound to, it has different properties, capabilities and characteristics. What controls, manages and uses the Components of an Object are the Systems, they are functions or classes describing a set of actions that process certain Components belonging to each of the Objects.
- **GFX (Graphics Effects)** is the third, last and main element that combines the previous two. It represents the core of the graphics library along with the full set of functionalities needed to fulfill the goal of this project. Several critical modules are distinguished here, necessary for the operation of the library. The first of them is the module containing all the Components describing and defining the behavior and character of the Objects. Another important element is the module describing the structure and method of presenting the graphic models, needed for loading and interpreting external resources containing geometric shapes as a mesh, elements or models. The main module contains the set of Systems performing the actions necessary for the full implementation of the already defined project goal. The group of systems here is divided into several main categories. Core is a set of basic

systems that take on solving basic tasks such as transformations, compositing, selection, level of detail management, removal of invisible objects, etc. Another group is the systems responsible for lighting and illumination, as well as the group of systems responsible for generating dynamic shadows in real time. Next is the group of systems for controlling and simulating particles, water, ocean, clouds and climate environment.

The following important references were used for the development of this project: GENNARO, 2023, MEYERS, 2010, MEYERS, 2017, SHREINER, 2013, WOLFF, 2011.

4 CONCLUSIONS

Creating a graphics library for educational purposes is a feasible task, even within a few months by a group of a couple of people. It is also quite possible to develop the library to a state that is ready to be used for practical or real-world tasks.

In order for the development of a given product, beyond a research purpose, to make sense, it is necessary to find a suitable niche to which the product should be directed. At this stage, it can be argued that the niche for graphics systems and libraries is successfully occupied by established products. A large set of these libraries have a huge number of people working on them, constantly adding new functionalities, expanding them in various directions, including not only graphic visualization.

A future direction for development, research and experimentation with various concepts and techniques for graphic visualization can be adding animations, techniques for realistic texturing, dynamic tessellation of objects in real time, etc.

ACKNOWLEDGEMENTS

This work was financially supported by the research fund at the University of Ruse, Bulgaria, and performed under the project FNI-2025-FPNO-04.

REFERENCES

Akenine, T., Haines, E., Hoffman, N., Pesce, A., Iwanicki, M. Hillaire, S. Real-Time Rendering 4th Edition, CRC Press, 2018

Andrade, A. Game engines: a survey, *EAI Endorsed Transactions on Serious Games*, Vol. 2, 2015, http://dx.doi.org/10.4108/eai.5-11-2015.150615

Dunn, F., Parberry, I. *3D Math Primer for Graphics 2nd Edition*, CRC Press, 2011

Gennaro, D. D. *Advanced Metaprogramming in Classic C++*, Berkeley, CA, 2023

Gregory, J. *Game Engine Architecture 3rd Edition*, CRC Press, 2019

Lengyel, E. *Mathematics for 3D Game Programming and Computer Graphics, Third Edition*, Course Technology Press, 2012

Meyers, S. *Effective STL*, Addison-Wesley, 2010

Meyers, S. *Effective C++ 3rd Edition*, Addison-Wesley, 2017

Shreiner, S., Sellers, G., Kessenich, J., Licea-Kane, B. *OpenGL Programming Guide 8th Edition*, Addison-Wesley, 2013

Wolff, D. *OpenGL 4.0 Shading Language Cookbook*, PACKT Publishing, 2011

Authors' Contribution

Both authors contributed equally to the development of this article.

Data availability

All datasets relevant to this study's findings are fully available within the article.

How to cite this article (APA):

Vassilev, T., & Iliev, S. (2025). METHODOLOGY AND TECHNIQUES USED IN THE DEVELOPMENT OF A GRAPHICS LIBRARY WITH OPENGL. Veredas Do Direito, 22(2), e223462. https://doi.org/10.18623/rvd.v22.n2.3462