JOURNAL OF CRITICAL REVIEWS
ISSN- 2394-5125 VOL 7, ISSUE 15, 2020

A COLLABORATIVE IDE FOR GRAPHICS PROGRAMMING

Yalmar Ponce Atencio[1], Julio Huanca Marin[2], Rembrandt Ubalde Enriquez[3], Ciro Rodriguez Rodriguez[4], Ivan Petrlik[5]

[1,2]Universidad Nacional José María Arguedas,
[3]Universidad Tecnológica del Perú,
[3,4,5]Universidad Nacional Federico Villareal
[1]yalmar@unajma.edu.pe, [2]juliohuanca@unajma.edu.pe, [3]c13198@utp.edu.pe, [4]crodriguez@unfv.edu.pe,
[5]ipetrlik@unfv.edu.pe

Received: 16 March 2020 Revised and Accepted: 17 June 2020

ABSTRACT: We present a collaborative coding environment that helps the teaching of introductory computer graphics courses, with the goal of to do teaching graphics fundamentals more effectively and lowering the excessive difficult of initiate into 3D graphics programming. Traditionally, the OpenGL Library is used for teaching computer graphics courses, since there are bindings for the most popular languages, including Python, Ruby, Java and the web (WebGL). In particular, our proposed framework provides a web IDE for collaborative coding, between classmates or workgroups, and the professor can help them or interact on it. The second component is the use of the Emscripten library, which allows the program with OpenGL on the web in the same way as it is done in a local mode. Compared with WebGL, our proposal has better benefits since the code could be used in any computer with OpenGL support, which has greatly benefited our students. It also was proposed to diminish the complexity of creating websites with 3D graphics support and better online games, which has been motivating students to learn 3D computer graphics, build educational graphical applications and articles, host them online. Compared to other online coding platforms, oriented to mainstream graphics educational materials and graphics in general, the collaborative IDE for programming computer graphics that we have developed offers the possibility of students could learn OpenGL in a fast way for both inside and outside our classrooms with the C++ language.

KEYWORDS: IDE, OpenGL, programming, education, Emscripten toolchain, computer graphics

I. INTRODUCTION

More and more extensive range of students of different ages is learning to program as part of their curricula, and formal education is a worldwide trend. In computer programming, as in other laboratory subjects, the student's active learning is very important for developing their skills in logic, math, creativity, among other subjects. In that sense, graphics programming has a special interest, because by using graphics is possible to explain the behavior of algorithms and programs, in a more detailed way. However, there are many difficulties in understanding how, when, and why practical learning has positive effects. To study the effects of hands-on for novices learning to program, frequently some graphics libraries or APIs were used, still nowadays are being used, like OpenGL, WebGL, DirectX, among others. Computer programming is part of many curricula, and nowadays, it is included from early education to college. Many times are teaching in computer labs, where a complete programming toolchain has installed and configured.

Commonly, the OpenGL library (a well known API-Application Programming Interface) and its dependent three-dimensional rendering libraries are used in a wide range of graphics applications, and many fields, including data visualization, human-computer interfaces, computer games, movie industry, CAD, education, and others [1,2,3,4].

The OpenGL programming library was designed to overcome the needs of many knowledge fields, for example, the use of CAD solutions to aerospace industries, military, medical and automotive applications. In many cases, OpenGL plays a major role in the graphical interfaces, due to the rapidly increasing need for 3D graphics in interactive applications [7,12].

In education, the OpenGL library is also widely used, and it is important to maintain a single way to program graphics since there are many different platforms. Canvas and WebGL dominate programming graphics on the web. This last one is a good tool, but the language used is JavaScript, which differs from C++, and in performance terms, it is much less efficient.

Considering that C++ is the best language to program graphics, the goal of this research is to provide and web-based IDE for programming graphics as it could be done in their local computer, by using the C++ language. OpenGL is widely used in many fields and is adequate to continue the study since desktop OpenGL is one of the most widely used graphics libraries.

The rest of the article is organized as follows: In the second section the previous works are presented, the third Section explains the proposed methodology and implementation details, Section fourth presents some achieved results; in the section fifth a discussion is presented and finally in part sixth the conclusions.

II. PREVIOUS WORKS

Younger children nowadays teach programming and computer science to college. Still, the most recommended methodology for programing is usually graphical programming. Depending on the situation, commonly are used environments such as Scratch and Alice in early education and more complex programming languages using the commonly OpenGL library in school and college. There are many studies about teaching programming concepts comparing several techniques to graphical programming [13].

Emscripten is a source-to-source compiler that runs as a back end to the LLVM compiler and produces a subset of JavaScript known as asm.js, was presented by Alon Zakai [5], and opens up two avenues for running code written in languages other than JavaScript on the web. It allows us to compile code directly into LLVM assembly, and then compile that into JavaScript using Emscripten, or Compile a language's entire runtime into LLVM and then JavaScript, as in the previous approach, and then use the compiled runtime to run code written in that language [8]. In our research, for example, the former method can work for C and C++, while the latter can work for Python; all three examples open up new opportunities for running code on the web [6].

In the past, the WWW was designed to only display documents written in HTML. Later, the style sheet (CSS), and programming using JavaScript were incorporated to get some dynamic functionalities. The main trend was achieved fully functional applications. Therefore, JavaScript is the way to programming applications on the Web, and thus it is needed to share a program on the Web, it must be written in that language. However, the problem is that JavaScript is a dynamic language. This language belongs to a group of scripting languages that tend to be easy to write but runs slower, but as the programmer writes less, the virtual machine running the code must figure out more, and this task takes additional time.

These problems have avoided high-performance programs running well on the Web. For example, game companies would love to be able to run their high-performance games on the Web; digital libraries like the Internet Archive want people to easily run emulated versions of old computer hardware and software so that they are not forgotten; and in general, if you have something that requires high performance, like physics simulation, it can be useful to share it through the Web. All you need to do is give people a link, and no matter what device they are using, they could easily run it. Then, there are many reasons to want to be able to run code on the Web at high speed [9].

C++ is one of the most popular programming languages, and the choice for many developers, mainly when it is want to implement high-performance applications. This often includes applications that require to display 3D graphics. Nowadays, 3D graphics are typical in medical applications, CAD applications, video games, among others. All of these types of applications demand responsiveness as a key usability feature. This makes the C++ language a perfect choice for this type of program because programmers can target and optimize for specific hardware platforms. However, when it looks at writing a simple 3D program, the best choice is to use the OpenGL API, which is supported on Windows, OSX, and most Linux distributions, then it's a perfect choice to the needs because you might be using any of these operating systems. Nevertheless, one of the more tedious aspects of OpenGL programming is the requirement to set up and manage windows in multiple operating systems if you want your application to run in more than one [10]. This job is made much easier by using windows management libraries. Some of them are the GLUT, GLFW, and SDL packages, which abstracts this task away behind an API, so you don't have to worry about how the window management is done [11,12].

III. METHODOLOGY AND IMPLEMENTATION DETAILS

For the implementation, we use the XP agile methodology, since our main goal was developing a rapid collaborative application.

For the web user interface, the programming languages JavaScript, CSS3, and HTML5 have been used. Additionally, for the achieve the collaboration functionalities, the Node.Js, jQuery, Socket.io, Spectrum, and Filesaver frameworks were used.

To accomplish the interaction, each client connects to the server through a socket connection (previously a socket_id is assigned to each connection created between a client and the server). A socket is a bidirectional connection, it means, each user is connected to the server, and the server is connected to each user in such a way it appears that each user is connected to all others in a room. The connection is maintained until the user disconnects from the application (closing the tab in the browser). Later, any changed piece of code (by any user) will be caught by the socket and, the server will emit this change to all the connected (to the server) users in the same room.

Furthermore, the implemented application has many useful features and functionalities to allow users to make really fast coding.

3.1. Programming OpenGL on the web

Currently, programming graphics on the web is possible by using WebGL (Web Graphics Library). WebGL is a JavaScript OpenGL binding for rendering interactive 2D and 3D graphics that run within any compatible web browser. The most interesting thing is that this allows to run OpenGL applications without the use of additional complements than the native HTML5 with the canvas element. WebGL is a fully integrated OpenGL framework for the web, accomplish with other web standards, allowing GPU-accelerated features. This allows to implement many kinds of graphics applications, from physics simulations to image processing, allowing nice effects as part of the web page canvas. WebGL elements can be mixed with other HTML elements and composited with other parts of the page or page background. WebGL programs are written in JavaScript, and it gives the programmer complete control of the application, including shader programming support. The used OpenGL is not the native but a subset called OpenGL ES. Similarly, the Shading Language is ES (GLSL ES). This last one is a language similar to C or C++ and is executed on a computer's graphics processing unit (GPU). Currently, WebGL is designed and maintained by the non-profit Khronos Group.

In the education field, the main inconvenience about to use WebGL is the need to use another language (JavaScript) different from C++, since many courses use C++ for teaching [14].

Our main goal was to provide a way to program graphics in a collaborative environment. This kind of environments nowadays is highly demanded, since virtual education is present in many education systems.

For graphics programming, our proposed system works essentially by using the Emscripten toolchain. This allows compiles from native languages like C/C++ into a full functionality web application based on the JavaScript language (see the diagram in Figure 1).

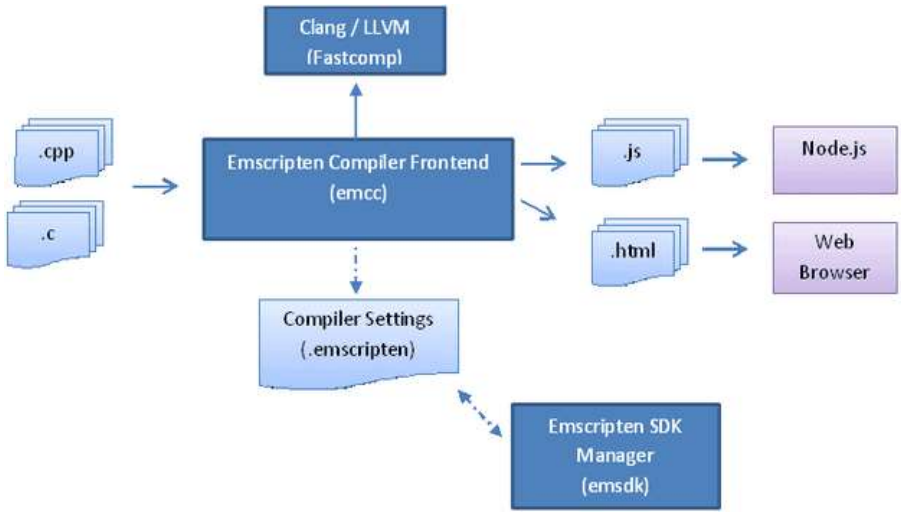

Figure 1. How the Emscripten framework works.

By using the Emscripten framework allow us to program in the same way as it will bring in our local computer. Related to OpenGL, the code is the same; then, it is a good choice for teaching graphics programming because the students could continue to work on their local computer before attending the classroom session.

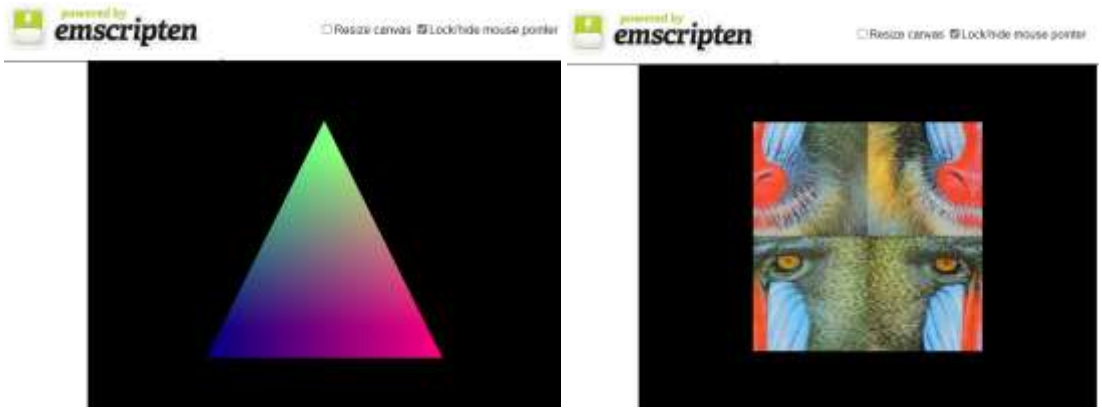

Figure 2. OpenGL examples compiled with Emscripten.

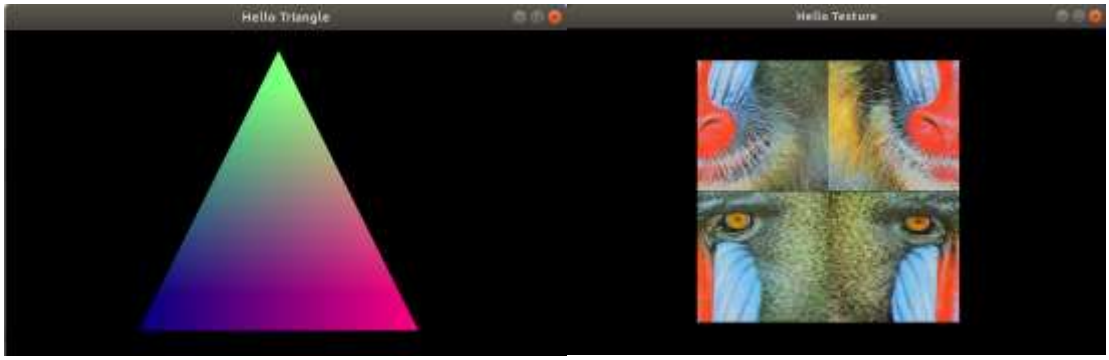

Figure 3. OpenGL examples compiled with C++ in an Ubuntu system.

As can be seen in the Figures above, we can write unique code to get the same result on the web and the desktop computer.

3.2. Proposed application architecture

The figure below shows the complete, implemented architecture.

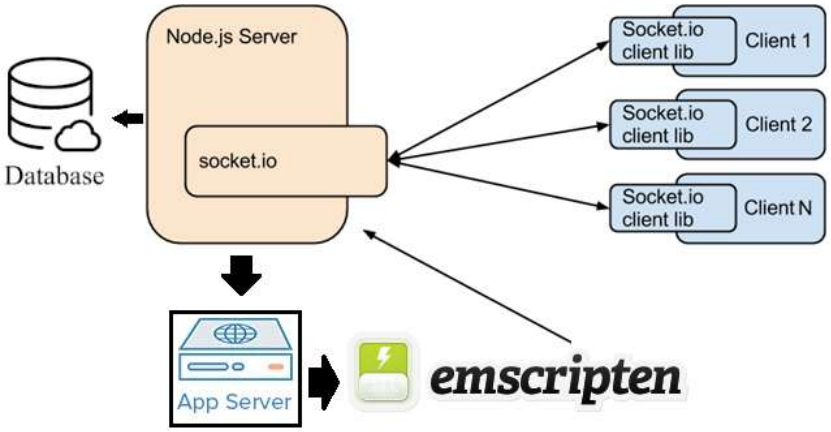

Figure 4. The proposed collaborative environment architecture.

Our application was designed as a collaborative tool to supply teaching needs to computer programming, more specifically, graphics programming. In the past, some graphics programming libraries were used; usually, tools and APIs incorporated on commercial IDEs, like Java Graphics or Visual Studio API for painting. All of these tools were designed for 2D graphics, is much more complicated the 3D programming. To overcome these needs and difficulties were created DirectX and OpenGL libraries. Still, the programming is very different from the traditional 2D graphics, mainly because it is needed to use additional window manager libraries and for many operating systems. From the 90s, OpenGL is being used in many different fields, since this graphics API is open source and was designed for multiplatform. But even being very popular, OpenGL is difficult to configure and install in most operating systems and for the different languages. Then, it is for many reasons that it is important to have an environment to teach graphics programming by using a unique language programming and mainly focusing on learning the most important issues like geometry, behavior, physics, maths, etc.

On the other hand, the developed collaborative system allows interacting between a group of classmates or when the professor teaches practical examples, where all the students could see the code edition on the collaborative editor.

3.3. Necessary Hardware and Software

The implemented collaborative system has two parts: The first is a common collaborative application where it allows the interaction between the connected users, have a code editor for interacting and to collaborate by correcting errors and aggregate code. Typically, it could be useful when a group of classmates needs to implement exercises as part of homework. The second part consists of installing the Ecmascript toolchain on the applications server. Then, when a compile task is required, from the collaborative IDE, this request is sent to the Node.Js server, and then this call to Ecmascript (emcc or em++), with the received code from the browser, to compile and generate the results that will be shown on the collaborative IDE.

The used application server has installed the Ubuntu 19.04 operating system, with all the needed software for supporting the graphics application generation, mainly the Ecmascript toolchain.

Related to hardware, we have used a server equipped with an Intel i7 5820K, 16GB DDR4 RAM, SSD 480GB.

For the clients, we have used many different devices, for instance, a cellphone with 6" touchscreen and 8GB RAM, a laptop with intel i7 5400U and 16GB RAM, Desktop with intel i3 and 8GB RAM, a tablet with 10.1" screen and 8GB RAM, and others devices. It is possible since the proposed collaborative IDE works entirely on the web.

Chrome, Firefox, and Chromium Edge browsers were used to test the collaborative environment.

IV. Results

Below are shown some examples that were implemented by using our collaborative IDE. Figures 5 and 6 show two examples. The first shows an only triangle, but in the implementation of this example was used the current OpenGL features, including program shaders.

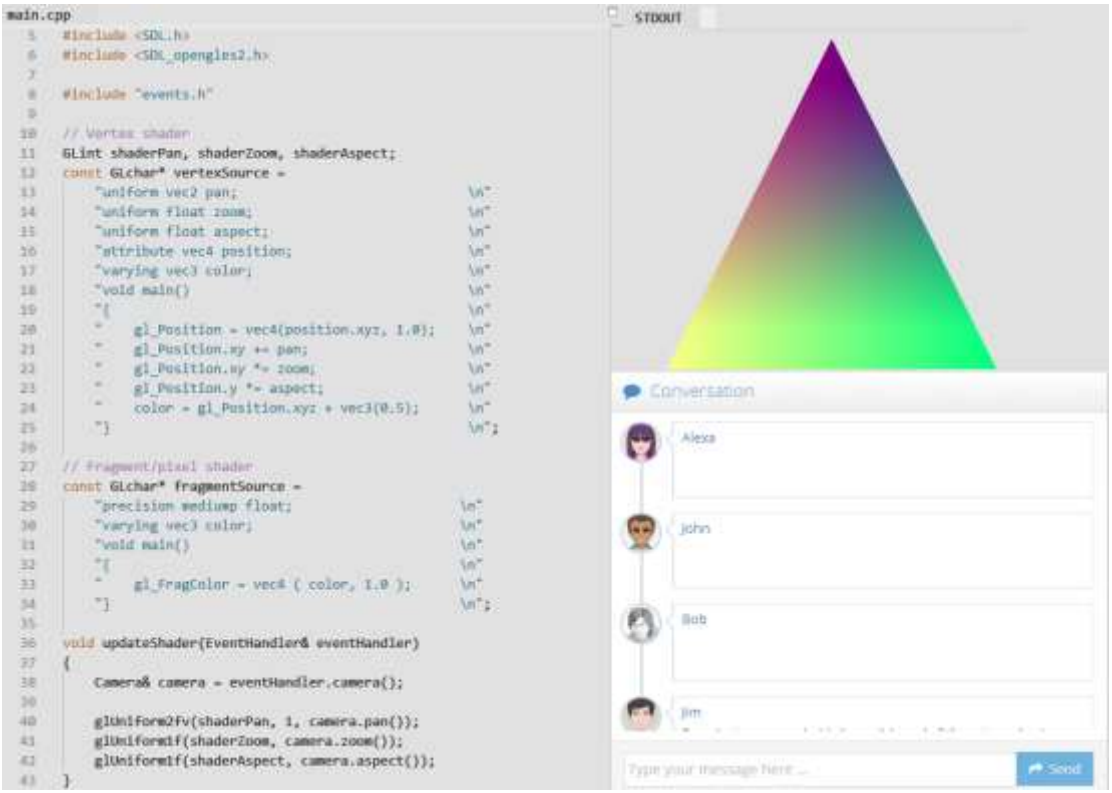

Figure 5. A first example is drawing a triangle.

The second example is more complex and uses textures. Still, additionally, notice that another OpenGL advanced feature was used, the VAO (vertex array objects), and, in consequence, more complex applications can be implemented.

Although in these two examples, basic 2D examples were shown, notice that OpenGL is enabled to implement complex 3D graphics applications, mainly because it is enabled shader programs and VAOs

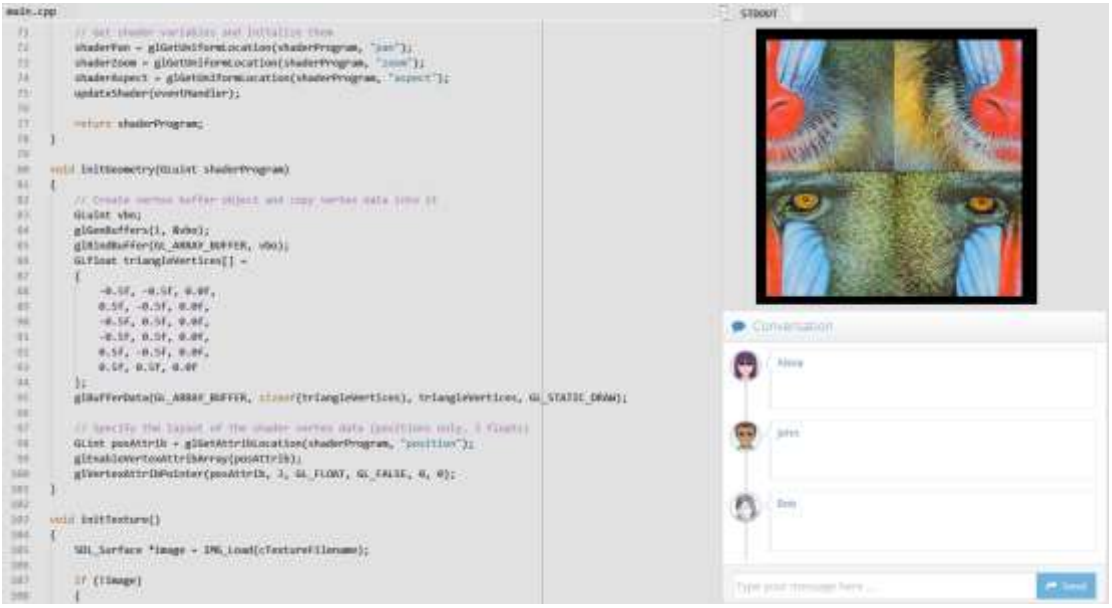

Figure 6. Example drawing a quad using a loaded texture.

V. DISCUSSION

A collaborative development environment for computer programming courses has been implemented. The reason for this research is because, in these times, where virtual education is growing up, there is a growing need to develop collaborative applications for education and mainly for teaching. In this scenario, in computer programming courses, teachers usually work in computer labs, where all computers have the same and necessary software, and the examples the teacher teaches must work on all student computers because the university has a support team for this. Then at home, the student should also be able to reproduce the examples developed in class, as long as they have the necessary tools and applications installed. In this sense, the collaborative application implemented in this research work aims to support the teacher's job to develop his classes online, in a similar way as it is done in a computer lab, and where the students also may be able to reproduce the results on their local computer.

On the other hand, a collaborative application is especially useful because it allows a group of colleagues to interact on a task, mainly because they can complete an exercise much faster or correct mistakes quickly.

VI. CONCLUSIONS AND FUTURE WORKS

We have introduced an online collaborative IDE for graphics programming by using the OpenGL library and the C++ programming language. All of these for supporting teaching in educational systems. The code could be edited by many users who belong to a classmate group or students of a classroom. The system allows implementing native OpenGL code in the C++ language, with support to modern graphics programming. This is mainly important since, frequently, on many curricula, the C++ programming language is used for most programming courses. Then it is recommendable to reinforce learning in the same programming language. Another advantage is to allow full code compatibility with desktop platforms. For future works, we consider using fewer complex configurations for the Emscripten cross compiler.

Another important thing is to test with much more complex implementations; it is also important to measure some aspects such as response time, memory consumption, and efficiency compared to WebGL.

VII. REFERENCES

[1] Chatzopoulos D, Bermejo C, Huang Z, Hui P (2017) Mobile augmented reality survey: from where we are to where we go. IEEE Access 5:6917–6950.
[2] Loseille A, Feuillet R (2018) Vizir: high-order mesh and solution visualization using OpenGL 4.0 graphic pipeline. In: 2018 AIAA Aerospace Sciences Meeting.
[3] Woods J, Christian J (2016) Glidar: an OpenGL-based, real-time, and open-source 3D sensor simulator for testing computer vision algorithms. J Imaging 2:5.
[4] Ridge, Garett & Terzopoulos, Demetri. (2019). An Online Collaborative Ecosystem for Educational Computer Graphics. Web3D '19: The 24th International Conference on 3D Web Technology. 1-10. 10.1145/3329714.3338133.
[5] Zakai, Alon. (2011). Emscripten: an LLVM-to-JavaScript compiler. 301-312. 10.1145/2048147.2048224.
[6] McAnlis, Colt & Lubbers, Petter & Jones, Brandon & Tebbs, Duncan & Manzur, Andrzej & Bennett, Sean & d'Erfurth, Florian & Garcia, Bruno & Lin, Shun & Popelyshev, Ivan & Gauci, Jason & Howard, Jon & Ballantyne, Ian & Freeman, Jesse & Kihira, Takuo & Smith, Tyler & Olmstead, Don & McCutchan, John & Austin, Chad & Pagella, Andres. (2014). HTML5 Games in C++ with Emscripten. 10.1007/978-1-4302-6698-3_18.
[7] Guha, Sumanta. (2018). An OpenGL Toolbox. 10.1201/9780429464171-3.
[8] Baek, Nakhoon. (2018). An emulation scheme for OpenGL SC 2.0 over OpenGL. The Journal of Supercomputing. 10.1007/s11227-018-2399-1.
[9] Zakai, Alon. (2018). Fast Physics on the Web Using C++, JavaScript, and Emscripten. Computing in Science & Engineering. PP. 1-1. 10.1109/MCSE.2018.110150345.
[10] Baek, Nakhoon. (2020). A Simplified Implementation of the Fixed-Function Graphics Pipeline: DRM Approach. International Journal of Advanced Trends in Computer Science and Engineering. 9. 1551-1555. 10.30534/ijatcse/2020/98922020.
[11] Khan, Mehbuba & Hashem, M.M.A.. (2019). A Comparison between HTML5 and OpenGL in Rendering Fractal. 1-6. 10.1109/ECACE.2019.8679427.
[12] Browning, J. & Sutherland, Bruce. (2020). 3D Graphics Programming. 10.1007/978-1-4842-5713-5_14.

[13] R. Lindén, A. García Díaz, E. Kaila, E. Lokkila, M.J. Laakso (2017) COMPARING PLAY-BASED LEARNING TO GRAPHICAL PROGRAMMING IN PROGRAMMING EDUCATION, EDULEARN17 Proceedings, pp. 2176-2184.
[14] von Hausswolff, Kristina. (2017). Hands-on in Computer Programming Education. 279-280. 10.1145/3105726.3105735.