The 8th International Scientific Conference
eLearning and software for Education
Bucharest, April 26-27, 2012
10.5682/2066-026X-12-138

A SIMPLE GRAPHICAL FRAMEWORK FOR THE ACQUISITION OF BASIC C PROGRAMMING SKILLS

Srdan MIHIC, Dragan IVETIC
Faculty of Technical Sciences, University of Novi Sad, Trg Dositeja Obradovica 6, Novi Sad, Serbia
E-mail: smihic@uns.ac.rs, ivetic@uns.ac.rs

Abstract*: The introductory (and only) C programming course at the freshmen's year of Animation Engineering (AE) degree proved to be a challenge, despite 20 years of experience in teaching programming courses. Examples of data abstraction and understanding of control structures and expressions in the C programming language had to be brought closer to students who have shown interest in computer graphics and animation, and thus have chosen to pursue AE degree. Instead of a classical first program - "Hello world", their first program should show a simple graphical object even it is just a single point. Specifically, students will programmatically control the execution of C program from a console window, whereas graphical window will be used as program's output. We have chosen to use the Microsoft Visual Studio 2010 integrated development environment (IDE) for this course as it is commonly used in our other C programming courses. Therefore, it had to be adapted to the needs and skills of the AE students. This was accomplished by enabling students to draw graphical objects on a simple 2D graphics canvas but without the need to possess any knowledge of computer graphics. For that purpose, several graphical libraries, including: openBGI, GDI, OpenGL and DirectX, were considered to accommodate the graphical component in the chosen IDE. They were compared according to ease of use, debugging support and documentation availability, from the perspective of absolute programming beginners. We have found that none of them are suitable for first time programmers. Because of this, we have implemented a lightweight wrapper graphical framework that abstracts and hides the underlying complexity of libraries used for drawing on a graphical canvas and window management. The framework offers clean design and ease of use. It provides only a small subset of 2D drawing and window management functions. All framework functions have a reduced number of parameters compared to their underlying counterparts as they set as many parameters as possible to predefined values. In addition, the names of functions are shorter than their counterparts. This way, students have to memorize less and can produce working code faster which should enhance their productivity. This framework, combined with compelling examples and assignments, should help students learn programming with ease.*

Keywords: *Education, Programming, Graphical framework, C programming language*

I. INTRODUCTION

Recently, the Faculty of Technical Sciences, University of Novi Sad, created an Animation Engineering (AE) degree. An introductory (and the only) C programming course is offered at the freshmen's year. In order to accommodate this new degree we had to significantly redesign the introductory C programming course offered to our computer science students. This proved to be challenging in spite of 20 years of experience in teaching C programming courses. Examples of fundamental C programming concepts, data abstraction and control structures had to be brought closer to students who have shown interest in computer graphics and animation. We wanted to enable AE students to draw graphical objects on a simple two-dimensional canvas with no prior knowledge of computer graphics and with no intention in teaching them computer graphics for that purpose. Thus, we needed an environment in which the students would be able to draw on a simple two-dimensional canvas in easy and efficient manner. It should provide good debugging support, preferably with visual stepped debugging, and has to be well-documented. In addition, the introduction of the graphical component should have a minimal learning overhead for the students.

We have chosen to use the Microsoft Visual Studio 2010 integrated development environment (IDE) as it is commonly used in our other C programming courses. Thus, it was required to find the graphics library that would provide means for two-dimensional drawing in this environment setup. We tried to employ an existing graphics library first. Thus, we conducted a brief survey of graphics libraries. Several graphics libraries were compared according the ease of use, documentation and debugging support from first-time programmer's perspective. Alas, we have found that none of these libraries are suitable for novice programmers. Thus, we developed a lightweight graphical framework – the Simple Graphical Framework (SGF). It hides the complexity of drawing on a two-dimensional canvas and eases window management, in a way that allows first-time programmers to learn C programming with ease.

The outline of this paper is as follows. In the following section, a brief survey of graphics libraries from absolute programming beginners' perspective is presented. In Section 3, the Simple Graphics Framework (SGF) is presented, as the main contribution of this work. Finally, Section 4 concludes this paper and presents further research directions.

II. A BRIEF GRAPHICS LIBRARIES SURVEY

This survey was restricted to only those graphics libraries that have an implementation in the C programming language for Microsoft Windows XP platform. These restrictions were imposed by the platform used in laboratories and the chosen programming language. Following libraries were examined: Borland Graphics Interface family of graphics libraries, Microsoft Graphics Device Interface [1], Open Graphics Library [2] and Microsoft DirectX [3].

Borland Graphics Interface (BGI) [4] family of libraries was development as replacement for old BGI library which was bundled with/included in Borland C and provided functions for DOS graphics mode. The BGI library had many issues: i) it was extremely slow; ii) it was never intended for 32-bit machines iii) did not work on newer versions of Windows; iv) had inconsistent API and weak debugging support. Although, it had many issues it was very simple to use. Because of this, it was considered suitable for teaching programming basics, where graphics output was needed. The library and its successors were used in many programming teaching courses, e.g. [5, 6]. In following, two representatives of this family will be presented.

WinBGIm graphics library [7], version 6.0 from 2004, was developed at Colorado University for the purpose of teaching basic programming skills. WinBGIm library maps the entire functionality of the old BGI library and adds following: RGB color model, mouse event handling, etc. Since it implements the same interface as the original BGI library, it inherited its inconsistence and some of the issues. It has moderate documentation support and examples can be found on the Internet.

OpenBGI graphics library [8], version 1.5 from 2008, is an open source project intended to replace the BGI library that was included in Borland C. This library implements the same interface as BGI and adds several extensions: mouse handling, custom graphics mode, etc. One of the strong points of this library is that it was written using ANSI C, and thus, it can be compiled with practically all Win32 compilers. It has an excellent debugging support. Namely, the graphics output window is repainted debug mode, so one can see all graphics output during stepped debugging. The library comes with no documentation of its own, but can be used with the original BGI documentation. In addition, a very few examples can be found on the Internet.

Microsoft Graphics Device Interface (GDI) [1] was a core graphical component for representing graphical objects in Windows operating system until Windows XP. The library is very old and slow, but it is widespread, since it was basic library for graphics drawing in Windows for decades. Now it is deprecated and superseded by Microsoft Direct2D [9] in Microsoft Windows

Vista/Seven. The library suffers from an inconsistent interface, but has excellent documentation support and many examples can be found. In addition, the library does not offer window management functions. Thus, Win32 functions must be used for window management.

In contrast with previously presented graphics libraries, OpenGL and DirectX are modern high-performance low-level hardware accelerated graphics libraries. They work in 3D natively, but can be used for two-dimensional graphics as well. Both of them offer clean design, but they are too complex for novice beginner to use them without any knowledge of computer graphics. Similar to GDI, these libraries does not include window management functions.

Open Graphics Library (OpenGL) [2], version 4.2 from 2011, is standard specification defining a cross-language, cross-platform API for writing applications that produce two-dimensional and three-dimensional graphics. Its interface includes more than 250 different functions that can be used to draw even the most complex three-dimensional scene from simple set of primitives. OpenGL standard is supported by many hardware vendors and its implementations exist for many platforms. Its interface offers a clean design for creation of applications with graphical output. It is widely used in professional CAD/CAM applications, virtual and augmented reality, information visualization, video games, etc. Thus, it has excellent documentation support and vast number of examples. In addition, it is frequently used in academic courses for teaching computer graphics and game development courses, e.g. [11, 12].

Microsoft DirectX [3], version 9.0c from 2004, is a collection of APIs for multimedia application and game development on Microsoft platforms. It consists of many APIs: Direct3D, DirectDraw, DirectSound, DirectMusic, DirectShow etc. Direct3D API is primary used for creation of three-dimensional computer graphics, although it can be used for creation of two-dimensional graphics as well. It provides a rich set of functions for 2D/3D drawing and it is well documented with vast amount of examples. Newer versions of DirectX were not included in this survey since they are not available for Windows XP platform (only Windows Vista and Windows Seven are supported).

Table 1. Graphics libraries survey

Name	Ease of use	Interface	Debugging support	Documentation
WinBGIm	Easy	Inconsistent	Yes	Good
openBGI	Easy	Inconsistent	Yes, with stepped debugging	Poor
GDI	Complex	Inconsistent	Yes	Excellent
OpenGL	Complex	Consistent	Yes	Excellent
DirectX (Direct3D)	Complex	Consistent	Yes	Excellent

Summary of this survey is presented in Table 1. From table 1, it is noticeable that none of the graphics libraries included in this survey is suitable for first-time programmers. Libraries are either too complex for a beginner to use them with no prior knowledge of computer graphics or have an inconsistent API which makes learning them a difficult task. Thus, we have implemented a lightweight wrapper graphical framework – a Simple Graphics Framework (SGF). It abstracts and hides the underlying complexity of libraries used for drawing on a simple graphical canvas and window management. The SGF is presented in detail in following section.

III. THE SIMPLE GRAPHICS FRAMEWORK (SGF)

The SGF was developed to aid acquisition of basic C programming skills for students of AE degree. We strived to make both drawing on a simple two-dimensional canvas and window management as simple as possible with minimal learning overhead for the students. The openBGI graphics library was used as basis since it is the only graphics library that supported visual stepped debugging. The SGF was implemented as a wrapper library on top of the openBGI graphics library.

The SGF enables drawing on a simple two-dimensional canvas with ease for first-time programmers. This is achieved by having a clean design, using a small subset of two-dimensional drawing functions, shortening function names, reducing number of function parameters, and hiding the complexity of the underlying graphics library.

The framework offers just two functions for window management: *openwindow* and *closewindow* that open and close graphical window respectively. For simplicity, we restricted that only one graphical window can be active, since it represents the graphical output of the program. It is set that opening of the graphical window repositions and realigns the console window so it is shown below the graphical window (figure 1). This gives a student a better overview since both windows are fully visible. In addition, this resembles a window configuration found in many CAD applications that the AE students are familiar with. To ease window management even more and boost productivity, *openwindow* function requires no parameters. It no parameters are passed to the function, it sets window dimensions to default 800x600 pixels. Still, it is possible to specify window dimensions when needed.

Unlike most of the graphical libraries, a Carthesian coordinate system was chosen where a coordinate origin is set in the center of the window. Namely, some libraries set coordinate origin in either upper-left or lower-left corner of the window (e.g. winBGIm and GDI) where others do not fixate coordinate system (e.g. OpenGL). Although each of these solutions has advantages, from perspective of first-time programmers they are not a very good choice, since students are not accustomed to them.

The SGF supports drawing of following two-dimensional graphical objects: point, line, arc, ellipse, pie slice, triangle, rectangle, quadrilateral, polygon, as well as a graphical output of text (figure 1). These graphical objects are common in many graphical applications (e.g. CorelDRAW) and in many graphics libraries. For drawing aforementioned graphical objects following functions are used: *pixel, line, arc, ellipse, pieslice, triangle, rectangle, quadrangle, polygon* and *text.*

For each graphical object, stroke and fill color can be specified. Fill color can be specified for closed contours only (e.g. rectangle). The framework offers two functions: *stroke* and *fill* for stroke color and fill color specification. The system that is used is found in many graphics libraries – stroke/fill color calls set the current stroke/fill color which remains current until the next stroke/fill color call. In addition, background color can be changed with *background* function. In contrast to most of the graphics libraries, this function changes background color immediately where as others change background after a background clearing function call. This immediate effect of background change has following benefits: the change is manifested when it happened (similar to background change in graphical applications such as CorelDRAW) and there is no need to call background clearing function. All color setting functions have one parameter – a specified color. We have chosen a small set of named colors, similar to GDI, instead of color models of OpenGL and DirectX for two reasons. Firstly, one does not need many colors since the accent of the course is not on the visual component and aesthetics, but on learning C programming concepts by developing programs with graphical output. Secondly, by introducing any of color models we would shift from teaching C programming to teaching computer graphics which is not our intention. Thus, we have chosen standard Microsoft Windows 4-bit (16 colors) palette (figure 2). Each color set call is specified with named color (e.g. *WHITE*). In order to support situations when it is necessary to set the stroke and fill color to none, special color value *NONE* was introduced to the palette (e.g. *fill(NONE)* results that all closed contours are drawn as outline only).

Beside graphics-oriented functions, the SGF provides utility functions for various conversions (e.g. degree/radian conversion), and delay functions which support and ease animation creation and control.

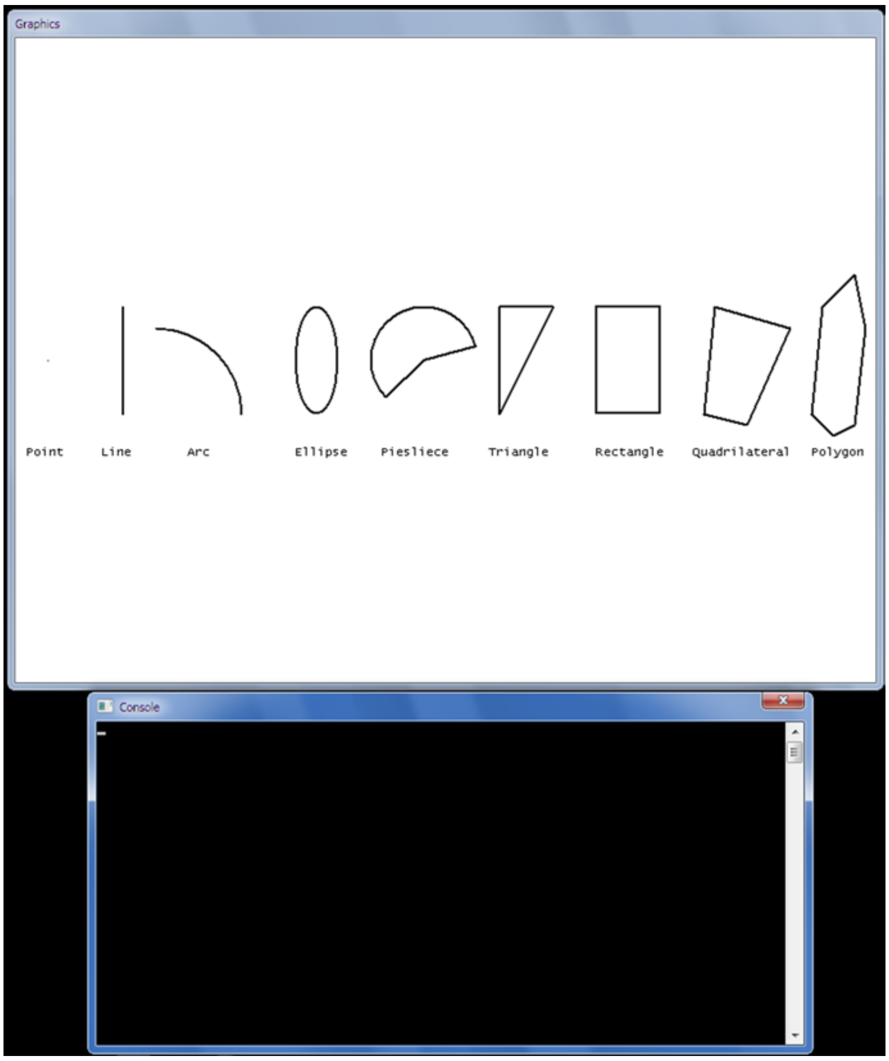

Figure 1. The SGF window setup presenting supported graphical objects

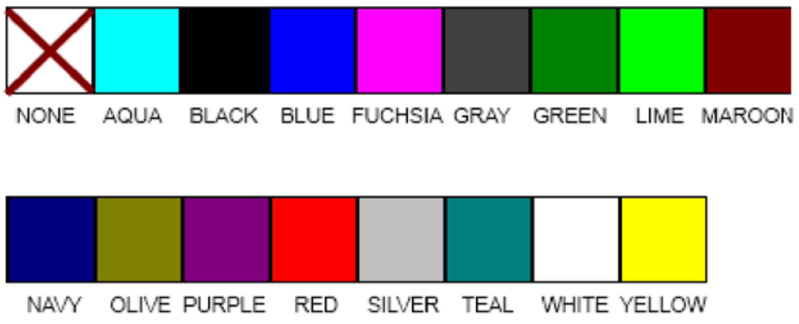

Figure 2. The SGF color palette

IV. CONCLUSIONS

This paper presented the Simple Graphics Framework (SGF) for the acquisition of basic C programming skills. It provides the means to draw on a simple 2D canvas with ease for first-time programmers. This is achieved by: i) providing a clean API; ii) supporting only a small set of two-dimensional drawing functions with a reduced number of parameters; iii) including a graphical window management as an integral part of the framework. This way, students have to memorize less and the learning of the framework introduces a minimal learning overhead. Thus, students can produce working code faster which should enhance their productivity. This framework, combined with compelling examples and assignments, should help students learn programming with ease.

As future work, we want to: i) evaluate the SGF framework; ii) compare it with similar approaches; iii) investigate how to improve the color palette that is used.

Acknowledgements

This research is financial supported by Ministry of Science and Technological Development, Republic of Serbia; under the project number III47003 "Infrastructure for Technology Enhanced Learning in Serbia", 2011-2014.

References

[1] Qingyuan, L., Hai, T., 2010.The Study on GDI/GDI+ Rendering Function Defects and How to Avoid Them. In 2nd International Conference on Information Engineering and Computer Science (ICIECS). IEEE Explore Digital Library, pages 1 – 5, ISBN: 978-1-4244-7939-9

[2] Wright, R.S., Haemel, G., Sellers, G., Lipchak B., 2010. OpenGL Super Bible – comprehensive tutorial and reference. 5th Ed., Addision-Wesley Professional.

[3] Thorn, A., 2005. DirectX 9 Graphics: The Definitive Guide to Direct 3D, Wordware Publishing, Inc., ISBN-10: 1-55622-229-7

[4] Ravichandran, D., 2003. Programming with C++, 2nd Ed., Tata McGraw-Hill Publishing Company Limited, ISBN: 0-07-049488-6

[5] CSCI 1300 - Spring 2007 Introduction to Computer Science Syllabus and Information, http://www.cs.colo-rado.edu/~main/intro/, last accessed on 21 February 2012

[6] Intro to Computer Science, CSCI 105 Fall 2005, Biola Unversity, http://csci.biola.edu/csci105/, last accessed on 21 February 2012

[7] WinBGIm library homepage, http://www.cs.colorado.edu/~main/cs1300/doc/bgi/index.html, last accessed on 21 February 2012

[8] OpenBGI library homepage, http://openbgi.sourceforge.net/, last accessed on 21 February 2012

[9] MSDN page for Microsoft Direct2D, http://msdn.microsoft.com/en-us/library/windows/desktop/dd370990 (v=vs.85).aspx, last accessed on 21 February 2012.

[10] Introduction to Computer Graphics, Spring 2004, http://www.cs.virginia.edu/~gfx/Courses/2004/Intro.Spring.04/, last accessed on 21 February

[11] CS148: Introduction to Computer Graphics and Imaging, Summer 2010, http://graphics.stanford.edu/courses/ cs148-10-summer/, last accessed on 21 February 2012