Mc Graw Hill Education 大学计算机教育国外著名教材系列 影印版

Computer Graphics
Algorithms and Implementations

计算机图形学
算法与实现

D. P. Mukherjee
Debasish Jana 著

清华大学出版社

大学计算机教育国外著名教材系列（影印版）

Computer Graphics

Algorithms and Implementations

计算机图形学

算法与实现

D. P. Mukherjee
Debasish Jana
著

清华大学出版社
北京

D. P. Mukherjee, Debasish Jana
Computer Graphics: Algorithms and Implementations
EISBN: 978-81-203-4089-3

Copyright © 2011 by PHI Learning Private Limited, New Delhi.

Original language published by The McGraw-Hill Companies, Inc. All Rights reserved. No part of this publication may be reproduced or distributed by any means, or stored in a database or retrieval system, without the prior written permission of the publisher.

Authorized English language edition jointly published by McGraw-Hill Education (Asia) Co. and Tsinghua University Press. This edition is authorized for sale only to the educational and training institutions, and within the territory of the People's Republic of China (excluding Hong Kong, Macao SAR and Taiwan). Unauthorized export of this edition is a violation of the Copyright Act. Violation of this Law is subject to Civil and Criminal Penalties.

本书英文影印版由清华大学出版社和美国麦格劳-希尔教育出版(亚洲)公司合作出版。此版本仅限在中华人民共和国境内(不包括中国香港、澳门特别行政区及中国台湾地区)针对教育及培训机构之销售。未经许可之出口，视为违反著作权法，将受法律之制裁。

未经出版者预先书面许可，不得以任何方式复制或抄袭本书的任何部分。

北京市版权局著作权合同登记号　图字：01-2011-7644

本书封面贴有 McGraw-Hill 公司防伪标签，无标签者不得销售。
版权所有，侵权必究。侵权举报电话：010-62782989　13701121933

图书在版编目(CIP)数据

计算机图形学：算法与实现＝Computer Graphics：Algorithms and Implementations：英文/（印）穆克赫尔吉（Mukherjee, D. P.），（印）贾纳（Jana, D.）著. --影印本. --北京：清华大学出版社，2012.1
大学计算机教育国外著名教材系列：影印版
ISBN 978-7-302-27487-2

Ⅰ. ①计…　Ⅱ. ①穆…　②贾…　Ⅲ. ①计算机图形学－高等学校－教材－英文　Ⅳ. ①TP391.41

中国版本图书馆 CIP 数据核字（2011）第 253588 号

责任编辑：龙啟铭
责任印制：何　芊

出版发行：清华大学出版社　　**地　　址**：北京清华大学学研大厦 A 座
http://www.tup.com.cn　　**邮　　编**：100084
社　总　机：010-62770175　　**邮　购**：010-62786544
投稿与读者服务：010-62776969, c-service@tup.tsinghua.edu.cn
质　量　反　馈：010-62772015, zhiliang@tup.tsinghua.edu.cn
印 刷 者：清华大学印刷厂
装 订 者：三河市金元印装有限公司
发 行 者：全国新华书店
开　　本：185×230　　**印张**：37.25
版　　次：2012 年 1 月第 1 版　　**印　　次**：2012 年 1 月第 1 次印刷
印　　数：1~3000
定　　价：59.00 元

产品编号：044597-01

出 版 说 明

进入21世纪，世界各国的经济、科技以及综合国力的竞争将更加激烈。竞争的中心无疑是对人才的竞争。谁拥有大量高素质的人才，谁就能在竞争中取得优势。高等教育，作为培养高素质人才的事业，必然受到高度重视。目前我国高等教育的教材更新较慢，为了加快教材的更新频率，教育部正在大力促进我国高校采用国外原版教材。

清华大学出版社从1996年开始，与国外著名出版公司合作，影印出版了“大学计算机教育丛书（影印版）”等一系列引进图书，受到国内读者的欢迎和支持。跨入21世纪，我们本着为我国高等教育教材建设服务的初衷，在已有的基础上，进一步扩大选题内容，改变图书开本尺寸，一如既往地请有关专家挑选适用于我国高等本科及研究生计算机教育的国外经典教材或著名教材，组成本套“大学计算机教育国外著名教材系列（影印版）”，以飨读者。深切期盼读者及时将使用本系列教材的效果和意见反馈给我们。更希望国内专家、教授积极向我们推荐国外计算机教育的优秀教材，以利我们把“大学计算机教育国外著名教材系列（影印版）”做得更好，更适合高校师生的需要。

清华大学出版社

Preface

Studying Computer Graphics enhances the core competence in the designing and development of algorithms which are both computationally interesting and mathematically challenging. More and more people are interested in Computer Graphics because it has emerged as a vehicle for effective communication of business data as well as scientific visualization. At the same time it has become the backbone of the modern day entertainment industry. The phenomenal increase in man-machine interface is due to the ever-expanding implementations of smarter ideas in graphical design.

Indeed Computer Graphics constitutes a core area of studies in Computer Science. Inclusion of graphical designing as an unerring component of Computer Science is already an accepted norm. There are several good books on graphics that cater to an intended audience. Therefore, our enthusiasm for writing yet another book on graphics needs justification. Our motivation for this comes mainly from our experiences in teaching graphics and conducting researches in the related areas.

Although students excel in understanding Computer Graphics algorithms and we have also seen them solving complex numerical problems of graphics, however, they perform poorly while implementing the basic ideas of graphics and executing them successfully. This is especially true in the case of handling discrete coordinates on screen or in visualizing 3D shapes on 2D screen, in specific relation to the viewpoint position. There is a widespread need to teach the theory of Computer Graphics along with the methods of its seamless implementation. We believe that this particular book will fulfil this objective. Since we have explained relevant theories alongside their simple C implementation in this book, we are confident that it will be acceptable to a wide section of undergraduate and postgraduate students. The C programs of this book generate application by targeting the Windows operating system. Dislike it or love it—we are in the Windows age and unless today's students are equipped with the know-how of Windows application—especially when we are studying a subject like Computer Graphics, they are likely to lose one of their finest sets of skill. The first chapter of the book introduces Windows programming and subsequent chapters deal with implementation of computer graphics algorithms with additional details of Windows programming as and when required.

Writing a book is considerably easy when you teach the subject. But, we are sure, it is hard for students to cope with the uncountable iterations and interpretations that authors keep on experimenting with, and which students have to accept without question, especially, in those cases where the implementation aspect of an algorithm is explained. We would however like to thank our students for their thoughtful challenges thrown at us over the years—challenges and questions that have helped us in reconstructing this manuscript. We are fortunate enough to be associated with the Institutes of national importance in India where they provide an environment in which instructors can complete a project of this nature. We would like to put

此为试读,需要完整PDF请访问: www.ertongbook.com

on record our sense of gratitude to PHI Learning who readily agreed to our book-proposal and helped in streamlining the editorial and production work.

Last but not the least, both of us are indebted to our respective families for their constant support and encouragement without which we would not have been able to come this far. We will be happy if students get benefitted from this book. Please send your feedback to **graphics.algoimpl@gmail.com.** For updates and errata, please refer to the website **http://sites.google.com/site/graphicsalgoimpl.**

D.P. Mukherjee
Debasish Jana

Contents

此为试读,需要完整PDF请访问: www.ertongbook.com

Chapter 1

Introduction to Windows Programming

~: Learning Objectives :~

The objectives of this chapter are to acquaint you with:

- Windows programming basics
- Windows operating system—The universal graphical user interface
- Data type notation in Windows programming
- Windows programming architecture
- Event-driven programming
- A simple program to create a window
- Program to display character in a window
- Menu handling

1.1 Prologue

One of the best ways to understand the computer graphics algorithms is through their implementation. This book presents the computer programs, which code some of the widely used graphics algorithms. For a wider acceptability, the programming language selected is C. However, the programs are targeted at the Microsoft's Windows ™ platform in order to derive the advantages of the features of the Windows ™. This chapter introduces some of the Microsoft's Win32 (32-bit Windows Operating System) API (application programming interface), which is the infrastructure that Microsoft offers so that the developers can build the standard Windows[1] user interface (graphical look and feel). As a result, the programs in this

[1] Throughout the book we will use Windows with capital *W* to refer to Windows OS, whereas window with small *w* for an instance of a window.

book are Windows programs written in C language. Through the example programs in this chapter, we will familiarize you with the bare minimum of Windows programming, with the objective that in subsequent chapters you can utilize this knowledge to implement algorithms of computer graphics. To run the programs given in this book, you need an access to a Windows based machine. These programs have been compiled and tested with Microsoft Visual C++ Version 6 and run under every version of Windows from Windows 95 through Windows XP and later. We assume the knowledge of C is a prerequisite.

In this chapter, we will explore Windows programming; we will learn how to create a window, how to write or draw in a window and above all how to implement the most common feature of the Windows—a menu. These will act as the building blocks for the implementation of graphics algorithms in subsequent chapters. However, an experienced reader familiar with the Windows program can certainly skip this chapter and can get a head start from the Chapter 2. If you are not in that bracket, read along.

Programming Windows is a bit different than the way we are familiar with the conventional character-oriented application development (for example, in MS-DOS or UNIX). As the programming guru Charles Petzold puts it, "Windows are easy to learn but difficult to program". Not because Windows is a multitasked[2] environment, not because it resolves the screen in terms of pixel resolution rather than characters, but because it defines every action to and due to a window as *messages*[3]. Therefore, programming Windows is programming messages with which we are not usually familiar. Messages are generated due to actions or activities related to a window, for example, when the window is resized, or repositioned, or mouse has been moved over the window area, so on and so forth. A Windows application expects to service messages and also generates messages to be serviced by the Windows Operating System (OS) or any other Windows applications. Is the Windows programming difficult? Not really—like any other programming, once we appreciate the core issues of message driven architecture, once we understand that from where these messages are generated and how they are serviced in a program, we are the *experts* in Windows programming. Let us then explore the message-driven architecture of Windows programming. We first recapitulate some major features of Windows followed by writing a simple Windows based application.

1.2 Windows Operating System–The Universal Graphical User Interface

It is indeed a waste of effort to prove the acceptance of Windows operating system as the universal Graphical User Interface (GUI). Probably, it is the main factor behind the popularity of Windows. Windows has standardized the user interface, may be chronologically it is not the first to do so—but let us face it—it is now the de facto standard. The interface is standardized, the buttons are defined, and mouse actions are fixed. Well, in one way, it is a constraint and we are not permitted to deviate too much from the set standards.

[2] More than one task is being serviced by the operating system.

[3] Messages can be treated same as commands or keywords in any other programming language or operating system.

As a developer or user of a particular OS, we need to understand the services offered by the OS, so that we can leverage our work. So, before developing any application on the Windows platform, let us explore some architectural features of underlying platform. We will explore briefly the nature and working of Windows, how applications use the Windows OS, and the facilities provided by the Windows platform. Well then, let us sniff a bit of the architectural pearls of the Windows.

1.3 Windows OS—Internals

Since there are many versions of the Windows operating system available, let us explore them briefly. Windows was originally a 16-bit graphical operating environment (not a full-fledged bootable operating system) on top of the MS-DOS that was developed by Microsoft. The first official release was Windows 1.0 in 1985 (in fact, originally called *Interface Manager*) with many limitations and some bare bone functionalities. Windows 1.0 did not gain popularity in the market. Microsoft Windows 2.0 came out in 1987 with some added functionalities but still with many limitations and could not capture much market popularity. First broad commercial success came in 1990 with the release of Windows 3.0 as graphical user interface oriented operating environment on top of DOS with pseudo-multitasking[4]. Later the variant of Windows 3.11 came as operating environment on top of DOS, added with basic required services with more sophistication for file handling, printing, GUI, etc. and minimal networking options. Windows 95 and its subsequent upgraded version, Windows 98 emerged as desktop GUI based 32-bit operating system with reduced dependencies and relationships with DOS and features like plug-and-play hardware support, sturdy file system FAT32, etc. and became spectacularly popular.

The 32-bit operating system maintained downward compatibility to 16-bit operating system applications for some time. However, the execution of such *old apps* (16-bit applications) may not be that efficient on a 32-bit operating system kernel. From NT (New Technology) versions, Windows relied solely on the NT kernel instead of the Windows MS-DOS subsystem. Windows NT emerged as a server-centric operating system and came with many important technical features and services, for example, file and print services, scalability, multiprocessor support, more sophisticated security, wide area connectivity, emulated DOS functionality, API support for the Portable Operating System Interface for UNIX (POSIX), government certifiable C2[5] security for resource protection, access rights, user quotas, etc.

---

[4] You may refer a book on Operating Systems, e.g. Operating Systems by Siverchatz and Galvin to know more on multitasking or pseudo-multitasking.

[5] Trusted Computer System Evaluation Criteria (TCSEC), a United States Government Department of Defense (DoD) standard defines four divisions of security: D, C, B and A where division A has the highest security. The subdivision C1 provides Discretionary Security Protection with separation of users and data and Discretionary Access Control (DAC) capable of enforcing access limitations on an individual basis. The subdivision C2 provides Controlled Access Protection with more finely grained DAC, individual accountability through login procedures, audit trails, resource isolation with requisite System Documentation and user manuals.

Later variants Windows NT 4.0, Windows 2000, Windows XP, Windows Server, and ME are all 32-bit operating systems based on NT Kernel. All of them support both process and thread-based multitasking. These operating systems allow multiple applications or processes to utilize a single CPU in a time multiplexed way. Every process is allotted a certain time slice, after which another process is run so that even with a single processor system, the OS gives a feeling of multitasking. The operating system scheduler determines the runtime priority and the time slice allocated to a process. Threads are lightweight processes and several threads can be executed in preemptive[6] multitasked environment and in the same address space[7] of that of the parent process.

The kernel of the Windows operating system, WinNT and its future variants provide basic functionality through the file NTOSKRNL.EXE with dynamic link library (DLL) file HAL.DLL (supports inter hardware communication) supported by two important files NTDLL.DLL (implements many kernel-mode[8] APIs or "Native Windows API" for file input/output functions, multithreading, etc.) and Win32K.SYS (performs most of the low-level tasks in Windows involving graphics and user interface). The dynamic link libraries (DLLs) are far more efficient unlike the executable files (.exe) containing APIs and kernels for Windows programming and message servicing. The user-mode APIs popularly known as Win32 APIs (32-bit API routines and libraries) are callable from user application programs written on top of the Windows. The Win32 API is supported by three important dynamic link libraries (DLL): Kernel32.DLL (user-mode equivalent of NTDLL.DLL), WinGDI.DLL (GDI stands for Graphics Device Interface, a core component of Microsoft Windows operating system and provides basic functions for drawing bitmaps and shapes like circles, rectangles, etc.) and User32.DLL (implements the familiar user interface of Windows on top of WinGDI and Win32K) supported by MSVCRT.DLL (implements C standard library functions) and WS2_32.DLL (contains the API for communicating through sockets on the Internet, also referred to as standard Berkeley socket). Older versions were named WINSOCK.DLL or WSOCK32.DLL.

We have reviewed briefly some of the relevant features of popular 32-bit Windows operating system. However, in this book, when we are referring to Windows features, we mean the generic features of Windows, not specifically Windows 2000 or Millennium version. The graphics algorithms implemented in this book are independent of any specific version of Windows OS.

Well then, let us start Windows programming.

---

[6] Preemption allows operating systems to preempt or stop a currently running task to process a higher priority task. Interrupt handling should be non-preemptive. Preemption helps system to be more responsive to handle multiple tasks in conjunction.

[7] An address space defines a range of discrete memory addresses for storing data or code applicable to a particular program in execution (process) or a subprogram (thread) in execution.

[8] Windows operating system and its modern variants are protected mode operating systems which segregate virtual memory into kernel space and user space. Kernel space is strictly reserved for running the kernel, device drivers and kernel extensions. Each user space process normally runs in its own virtual memory space, and, normally cannot access the memory of other processes. User-written application programs run in user mode and not in kernel or privileged mode.

1.4 Windows Programming

Two different approaches can be used for Windows programming. The first approach could be that we can use the application programming interface (API) routines as and when necessary in the application program written in C. The other approach is to use C++ class libraries defining actions to and due to Windows, commonly referred to as Microsoft Foundation Classes (MFC). While the former gives the complete control of message flow in the application, the latter generates a bit cryptic code at the first glance as quite a significant message and Windows-related actions are masked in the class libraries. There is of course a quicker way to accelerate the second type of programming; however, we will not be using that in this book. We purposely avoid the second type of programming, using MFC, Object Window Library (OWL), and other such quicker ways. Instead, we will be using bare Win32 system. Through the example programs spread throughout this book, we will take you to eventually learn and specialize in graphics algorithms and their implementations. As such, we will purposefully omit certain concepts and practices related to Windows programming in order to stay focused to computer graphics algorithms. All the codes given in this book have been written, tested and compiled, using Microsoft's Visual C++ 6.0 compiler. You may, however, use any other compiler of your choice on Windows platform.

So, we will take the *first* approach in this book as described in the earlier paragraph—using APIs to write Windows program in C. This gives direct control over the basic actions of the Windows and the application the operating system points to. We assume, we are the first generation Windows programmer. This experience will help us in implementing the graphics gizmos—the main focus of this book.

Description of Windows program in this book is incremental in nature. As we will be exploring through the chapters, we will explain the additional features of Windows programming over and above the one that we will be covering in this chapter. The program description includes presentation of the problem and the associated theory, followed by listing of codes necessary to develop the application. The explanation of code attempts to relate the code with the associated theory. The related resource and header files are explained as and when required. Let us first familiarize ourselves with the notation conventions used in the Windows program.

1.4.1 Data Type Notation

The Windows programmers use a bit strange notation to represent variables used in a Windows program. This is courtesy legendary Microsoft programmer, Charles Simonyi and because of his Polish descendence, the notation is often referred to as Polish notation. At a first glance, they seem strange but as we understand the *rhythm* of the notation, we will be able to use it at ease. Usually the name of the variable will be prefixed with letter(s) describing the data type. So the notation is useful once we are familiar with it. The Polish notation actually increases the readability of the code. For example, a variable *szWinName* is a character string (prefix *s*) ending with zero (prefix *z*). A Windows message generated on pressing the left button of the mouse is specified by *WM_LBUTTONDOWN*, the prefix *WM* stands for Windows Message. Rather than listing all possible prefix notations, we will introduce them as and when we encounter them; but here is a glimpse of a few in the following section.

1.4.2 Predefined Constants

Windows uses an exclusive list of predefined constants (styled in upper cases) used as messages, flag values, and other operational parameters. These constants usually include a two or three-letters prefix set, as described in Table 1.1.

Table 1.1 Some predefined prefix values

Prefix	*Category*	*Example as used in window messages*
CS	Class style[9]	*CS_HREDRAW* — redraws the entire window if a movement or size adjustment changes the width of the window area
		CS_VREDRAW — redraws the entire window if a movement or size adjustment changes the height of the window area
CW	Create window	*CW_USEDEFAULT* — system selects the default position or size of the window, as applicable
DT	Draw text	*DT_CENTER* — centres text horizontally in the bounding rectangle or window
IDC	Cursor ID or identifier	*IDC_ARROW* — sets the arrow cursor style for displaying cursors or mouse pointers
IDI	Icon ID or identifier	*IDI_APPLICATION* — default application icon
WM	Window message	*WM_PAINT* — sent when the system or another application makes a request to paint a portion of an application's window
WS	Window style	*WS_OVERLAPPEDWINDOW* — this style gives the window a title bar, a window menu, a sizing border, and minimize and maximize buttons.

With this preliminary idea of data type notations, let us see the Windows programming architecture.

1.4.3 Windows Programming Architecture

A Windows program has two main sections. Let us call them window generation section and message serving section. Whereas window generation part is fairly universal for most of the applications that we discuss in this book, the message servicing part is application-specific and programmer's responsibility. As explained earlier, writing a Windows program is essentially designing the messages and associated actions depending on the application.

In the window generation section, we need to define the window—basically the generic window class with which the application is going to work. We have to specify the colour of the window, the cursor, the menu type, the icon, etc. associated with the window class that we will be using. The definition is followed by the registration of the window class that we have just defined—this is in fact an instantiation of the window—a "new" window is now born with

[9] A window class represents a structure as a template, from which several window instances may be created. The structure specifies icons, menu, background colour, position, size and other applicable attributes. It also holds a pointer to a procedure that controls how the window behaves in response to window events caused due to user interaction or generated internally.

此为试读,需要完整PDF请访问: www.ertongbook.com