CONTENTS:

Preface
- What is Computer Graphics?
- What is a Graphics API?
- Why do Computer Graphics?
- Overview of the Book

Getting Started
- 3D Geometry
  - 3D model coordinate systems
  - 3D world coordinate system
  - 3D eye coordinate system
  - Clipping
  - Projections
  - 2D eye coordinates
  - 2D screen coordinates
- Appearance
  - Color
  - Texture mapping
  - Depth buffering
- The viewing process
  - Different implementation, same result
  - Summary of viewing advantages
- A basic OpenGL program
  - The structure of the main() program using OpenGL
  - Model space
  - Modeling transformation
  - 3D world space
  - Viewing transformation
  - 3D eye space
  - Projections
  - 2D eye space
  - 2D screen space
  - Appearance
  - Another way to see the program
- OpenGL extensions
- Summary
- Questions
- Exercises
- Experiments

Chapter 1: Viewing and Projection
- Introduction
- Fundamental model of viewing
- Definitions
  - Setting up the viewing environment
  - Defining the projection
  - View volumes
  - Calculating the perspective transformation
  - Clipping on the view volume
  - Defining the window and viewport
• Some aspects of managing the view
  - Hidden surfaces
  - Double buffering
  - Clipping planes
• Stereo viewing
• Implementation of viewing and projection in OpenGL
  - Defining a window and viewport
  - Reshaping the window
  - Defining a viewing environment
  - Defining perspective projection
  - Defining an orthogonal projection
  - Managing hidden surface viewing
  - Setting double buffering
  - Defining clipping planes
• Implementing a stereo view
• Summary
• Questions
• Exercises
• Experiments

Chapter 2: Principles of Modeling

Simple Geometric Modeling
• Introduction
• Definitions
• Some examples
  - Point and points
  - Line segments
  - Connected lines
  - Triangle
  - Sequence of triangles
  - Quadrilateral
  - Sequence of quads
  - General polygon
  - Polyhedron
  - Aliasing and antialiasing
  - Normals
  - Data structures to hold objects
  - Additional sources of graphic objects
  - A word to the wise

Transformations and modeling
• Introduction
• Definitions
  - Transformations
  - Composite transformations
  - Transformation stacks and their manipulation
  - Compiling geometry
• A word to the wise

Scene graphs and modeling graphs
• Introduction
• A brief summary of scene graphs
  - An example of modeling with a scene graph
• The viewing transformation
• The scene graph and depth testing
Chapter 3: Implementing Modeling in OpenGL

- The OpenGL model for specifying geometry
  - Point and points mode
  - Line segments
  - Line strips
  - Line loops
  - Triangle
  - Sequence of triangles
  - Quads
  - Quad strips
  - General polygon
  - Antialiasing
  - The cube we will use in many examples
- Additional objects with the OpenGL toolkits
  - GLU quadric objects
    - GLU cylinder
    - GLU disk
    - GLU sphere
  - The GLUT objects
  - An example
- A word to the wise
- Transformations in OpenGL
- Code examples for transformations
  - Simple transformations
  - Transformation stacks
  - Inverting the eyepoint transformation
  - Creating display lists
- Summary
- Questions
- Exercises
- Experiments
- Projects

Chapter 4: Mathematics for Modeling

- Coordinate systems and points
- Points, lines, and line segments
- Distance from a point to a line
- Line segments and parametric curves
- Vectors
- Dot and cross products of vectors
- Reflection vectors
- Transformations
- Planes and half-spaces
- Distance from a point to a plane
- Polygons and convexity
• Polyhedra
• Collision detection
• Polar, cylindrical, and spherical coordinates
• Higher dimensions?
• Questions
• Exercises
• Experiments

Chapter 5: Color and Blending
• Introduction
• Principles
  - Specifying colors
  - The RGB cube
  - Luminance
  - Other color models
  - Color depth
  - Color gamut
  - Color blending with the alpha channel
  - Challenges in blending
  - Modeling transparency with blending
  - Indexed color
  - Using color to create 3D images
• Some examples
  - An object with partially transparent faces
• Color in OpenGL
  - Specifying colors
  - Enabling blending
• A word to the wise
• Code examples
  - A model with parts having a full spectrum of colors
  - The HSV cone
  - The HLS double cone
  - An object with partially transparent faces
• Summary
• Questions
• Exercises
• Experiments
• Projects

Chapter 6: Visual Communication
• Introduction
• General issues in visual communication
  - Use appropriate representation for your information
  - Keep your images focused
  - Use appropriate presentation levels for your information
  - Use appropriate forms for your information
  - Be very careful to be accurate with your information
  - Understand and respect the cultural context of your audience
  - Make your interactions reflect familiar and comfortable relationships between cause and effect
• Shape
  - Comparing shape and color encodings
• Color
  - Emphasis colors
  - Background colors
  - Color deficiencies in audience
  - Naturalistic color
  - Pseudocolor and color ramps
  - Implementing color ramps
  - Using color ramps
  - To light or not to light
  - Higher dimensions
• Dimensions
• Image context
  - Choosing an appropriate view
  - Legends to help communicate your encodings
  - Labels to help communicate your problem
• Motion
  - Leaving traces of motion
  - Motion blurring
• Interactions
• Cultural context of the audience
• Accuracy
• Output media
• Implementing some of these ideas in OpenGL
  - Using color ramps
  - Legends and labels
  - Creating traces
  - Using the accumulation buffer
• A word to the wise

Chapter 7: Graphical Problem Solving in Science
• Introduction
• Examples
• Diffusion
  - Temperatures in a bar
  - Spread of disease
• Function graphing and applications
• Parametric curves and surfaces
• Graphical objects that are the results of limit processes
• Scalar fields
• Simulation of objects and behaviors
  - Gas laws and diffusion principles
  - Molecular display
  - A scientific instrument
  - Monte Carlo modeling process
• 4D graphing
  - Volume data
  - Vector fields
• Graphing in higher dimensions
• Data-driven graphics
• Code examples
  - Diffusion
  - Function graphing
  - Parametric curves and surfaces
  - Limit processes
- Scalar fields
- Representation of objects and behaviors
- Molecular display
- Monte Carlo modeling
- 4D graphing
- Higher dimensional graphing
- Data-driven graphics

• Summary
• Credits
• Questions
• Exercises
• Experiments
• Projects

Chapter 8: Lighting and Shading

Lighting
• Definitions
  - Ambient, diffuse, and specular light
  - Surface normals
• Light properties
  - Light color
  - Positional lights
  - Spotlights
  - Attenuation
  - Directional lights
  - Positional and moving lights

Materials
Shading
• Definitions
  - Flat shading
  - Smooth shading
  - Examples of flat and smooth shading
• Calculating per-vertex normals
  - Averaging polygon normals
  - Analytic computations
• Other shading models
  - Vertex and pixel shaders

Lighting, Shading, and Materials in Scene Graphs

Global Illumination

Local Illumination
• Lights and materials in OpenGL
  - Specifying and defining lights
  - Defining materials
  - Setting up a scene to use lighting
  - Using GLU quadric objects
  - An example: lights of all three primary colors applied to a white surface
  - Code for the example
• A word to the wise
  - Shading example
• Summary
• Questions
• Exercises
• Experiments
• Projects
Chapter 9: Texture Mapping

- Introduction
- Definitions
  - 1D texture maps
  - 2D texture maps
  - 3D texture maps
  - Associating a vertex with a texture point
  - The relation between the color of the object and the color of the texture map
  - Other meanings for texture maps
  - Texture mapping in the scene graph
- Creating a texture map
  - Getting an image as a texture map
  - Generating a synthetic texture map
- Texture mapping and billboards
  - Including multiple textures in one texture map
- Interpolation for texture maps
- Antialiasing in texturing
- MIP mapping
- Multitexturing
- Using billboards
- Texture mapping in OpenGL
  - Associating vertices and texture points
  - Capturing a texture from the screen
  - Texture environment
  - Texture parameters
  - Getting and defining a texture map
  - Texture coordinate control
  - Texture interpolation
  - Texture mapping and GLU quadrics
- Some examples
  - The Chromadepth™ process
  - Environment maps
- A word to the wise
- Code examples
  - A 1D color ramp
  - An image on a surface
  - An environment map
  - Multitexturing code
- Summary
- Questions
- Exercises
- Experiments
- Projects

Chapter 10: The Rendering Pipeline

- Introduction
- The pipeline
- The rendering pipeline for OpenGL
  - Texture mapping in the rendering pipeline
  - Per-fragment operations
  - Some extensions to OpenGL
  - An implementation of the rendering pipeline in a graphics card
- The rasterization process
• Some 3D Viewing Operations with Graphics Cards
• Summary
• Questions
• Exercises
• Experiments

Chapter 11: Event Handling
• Definitions
• Some examples of events
  - keypress events
  - mouse events
  - menu events
  - window events
  - system events
  - software events
• The vocabulary of interaction
• Object selection
• Events and the scene graph
• A word to the wise
• Events in OpenGL
• Callback registering
• Some details
  - More on menus
• Code examples
  - Idle event callback
  - Keyboard callback
  - Menu callback
  - Mouse callback for mouse motion
  - Mouse callback for object picking
• Details on picking
  - Definitions
  - Making picking work
  - The pick matrix
  - Using the back buffer to do picking
  - A selection example
  - A summary of picking

The MUI (Micro User Interface) Facility
• Introduction
• Definitions
  - Menu bars
  - Buttons
  - Radio buttons
  - Text boxes
  - Horizontal sliders
  - Vertical sliders
  - Text labels
• Using the MUI functionality
• Some examples
• Installing MUI for Windows
• A word to the wise
• Summary
• Questions
• Exercises
• Experiments
Chapter 12: Dynamics and Animation

- Introduction
- Definitions
- Keyframe animation
  - Temporal aliasing
  - Building an animation
- Some examples
  - Moving objects in your model
  - Moving parts of objects in your model
  - Moving the eye point or the view frame in your model
  - Changing features of your models
- Some points to consider when doing animations with OpenGL
- Code examples
- A word to the wise
- Summary
- Questions
- Exercises
- Experiments
- Projects

Chapter 13: High-Performance Graphics Techniques

- Definitions
- Techniques
  - Hardware avoidance
  - Designing out visible polygons
  - Culling polygons
  - Avoiding depth comparisons
  - Front-to-back drawing
  - Binary space partitioning
  - Clever use of textures
  - System speedups
  - Level of detail
  - Reducing lighting computation
  - Fog
  - Collision detection
- A word to the wise
- Summary
- Questions
- Exercises
- Experiments
- Projects

Chapter 14: Interpolation and Spline Modeling

- Introduction
  - Interpolations
  - Extending interpolations to more control points
  - Generating normals for a patch
  - Generating texture coordinates for a patch
- Interpolations in OpenGL
  - Automatic normal and texture generation with evaluators
  - Additional techniques
- Definitions
• Some examples
  - Spline curves
  - Spline surfaces
• A word to the wise
• Summary
• Questions
• Exercises
• Experiments
• Projects

Chapter 15: Per-Pixel Operations
• Introduction
• Definitions
• Ray casting
• Ray tracing
• Ray tracing and global illumination
• Volume rendering
• Fractal images
• Iterated function systems
• Per-pixel operations supported by OpenGL
• Summary
• Questions
• Exercises
• Experiments
• Projects

Chapter 16: Hardcopy
• Introduction
• Definitions
  - Digital images
  - Print
  - Film
  - Video
  - Digital video
  - 3D object prototyping
  - The STL file
• Summary
• Questions
• Exercises
• Experiments
• Projects

References and Resources
• References
• Resources

Appendices
• Appendix I: PDB file format
• Appendix II: CTL file format
• Appendix III: STL file format

Index
Evaluation

- Instructor’s evaluation
- Student’s evaluation

Because this is a draft of a textbook for an introductory, API-based computer graphics course, the author recognizes that there may be some inaccuracies, incompleteness, or clumsiness in the presentation and apologizes for these in advance. Further development of these materials, as well as source code for many projects and additional examples, is ongoing continuously. All such materials will be posted as they are ready on the author’s Web site:

http://www.cs.csustan.edu/~rsc/NSF/

Your comments and suggestions will be very helpful in making these materials as useful as possible and are solicited; please contact

Steve Cunningham
California State University Stanislaus
rsc@cs.csustan.edu

While the author retains copyright and other associated rights, permission is given to use this manuscript and associated materials in both electronic and printed form as a resource for a beginning computer graphics course.

---

This work was supported by National Science Foundation grant DUE-9950121. All opinions, findings, conclusions, and recommendations in this work are those of the author and do not necessarily reflect the views of the National Science Foundation. The author also gratefully acknowledges sabbatical support from California State University Stanislaus and thanks the San Diego Supercomputer Center, most particularly Dr. Michael J. Bailey, for hosting this work and for providing significant assistance with both visualization and science content. Ken Brown, a student of the author’s, provided invaluable and much-appreciated assistance with several figures and concepts in this manuscript. The author also thanks students Ben Eadington, Jordan Maynard, and Virginia Muncy for their contributions through examples, and a number of others for valuable conversations and suggestions on these notes.
Preface

Computer graphics is one of the most exciting ways that computing has made an impact on the world. From the simple ways that spreadsheets allow you to create charts to see data, to the ways graphics has enhanced entertainment by providing new kinds of cartoons and special effect, to the ways graphics has enabled us to see and understand scientific principles, computer graphics is everywhere we turn. This important presence has come from the greatly improved graphics hardware and software that is found in current computing systems. With these advances, computer graphics has emerged from a highly technical field, needing very expensive computers and frame buffers and requiring programmers to master all the mathematics and algorithms needed to create an image. It has now become a field that allows the graphics programmer to think and work at a much higher level of modeling and to create effective images that communicate effectively with the user. We believe that the beginning computer graphics course should focus on how the student can learn to create effective communications with computer graphics, including motion and interaction, and that the more technical details of algorithms and mathematics for graphics should be saved for more advanced courses.

What is Computer Graphics?

Computer graphics is involved in any work that uses computation to create or modify images, whether those images are still or moving; interactive or fixed; on film, video, screen, or print. It can also be part of creating objects that are manufactured from the processes that also create images. This makes it a very broad field, encompassing many kinds of uses in the creative, commercial, and scientific worlds. This breadth means that there are many kinds of tools to create and manipulate images for these different areas. The large number of tools and applications means that there are many different things one could learn about computer graphics.

In this book, we do not try to cover the full scope of work that can be called computer graphics. Rather, we view computer graphics as the art and science of creating synthetic images by programming the geometry and appearance of the contents of the images, and by displaying the results of that programming on appropriate devices that support graphical output and interaction. This focus on creating images by programming means that we must learn how to think about how to represent graphical and interaction concepts in ways that can be used by the computer, which both limits and empowers the graphics programmer.

The work of the programmer is to develop appropriate representations for the geometric objects that are to make up the images, to assemble these objects into an appropriate geometric space where they can have the proper relationships with each other as needed for the image, to define and present the look of each of the objects as part of that scene, to specify how the scene is to be viewed, and to specify how the scene as viewed is to be displayed on the graphic device. The programming may be done in many ways, but in current practice it usually uses a graphics API that supports the necessary modeling and does most of the detailed work of rendering the scene that is defined through the programming. There are a number of graphics APIs available, but the OpenGL API is probably most commonly used currently.
In addition to the creation of the modeling, viewing, and look of the scene, the programmer has two other important tasks. Because a static image does not present as much information as a moving image, the programmer may want to design some motion into the scene, that is, may want to define some animation for the image. And because the programmer may want the user to have the opportunity to control the nature of the image or the way the image is seen, the programmer may want to design ways for the user to interact with the scene as it is presented. These additional tasks are also supported by the graphics API.

**What is a Graphics API?**

An API is an *Application Programming Interface*—a set of tools that allow a programmer to work in an application area. The API’s tools are oriented to the tasks of the application area, and allow a programmer to design applications using the concepts of the area without having to deal with the details of the computer system. Among the advantages of an API is that it hides the details of any one computer system and allows the programmer to develop applications that will work on any of a wide range of systems. Thus a **graphics API** is a set of tools that allow a programmer to write applications that include the use of interactive computer graphics without dealing with system details for tasks such as window handling and interactions.

Besides covering the basic ideas of interactive computer graphics, this book will introduce you to the OpenGL graphics API and to give you a number of examples that will help you understand the capabilities that OpenGL provides and will allow you to learn how to integrate graphics programming into your other work.

**Why do Computer Graphics?**

Computer graphics has many faces, so there are many reasons why one might want to use computer graphics in his or her work. Many of the most visible uses of computer graphics are to create images for the sciences (scientific visualization, explanations to the public), entertainment (movies, video games, special effects), for creative or aesthetic work (art, interactive installations), for commercial purposes (advertising, communication, product design), or for general communication (animated weather displays, information graphics). The processes described in this book are all fundamental to each of these applications, although some of the applications will want the kinds of sophistication or realism in images that are not possible through simple API programming.

In all of these application areas, and more, there is a fundamental role for computer graphics in solving problems. Problem solving is a basic process in all human activity, so computer graphics can play a fundamental role in almost any area, as shown in Figure 1. This figure describes what occurs as someone:
- identifies a problem
- addresses the problem by building a model that represents it and allows it to be considered more abstractly
- identifies a way to represent the problem geometrically
- creates an image from that geometry so that the problem can be seen visually
uses the image to understand the problem or the model and to try to understand a possible solution.

Figure 1: Computer graphics in the problem-solving process

The image that represents a problem can be made in many ways. One of the classical uses of images in problem solving is simply to sketch an image—a diagram or picture—to communicate the problem to a colleague so it can be discussed informally. (In the sciences, it is assumed that restaurants are not happy to see a group of scientists or mathematicians come to dinner because they write diagrams on the tablecloth!) But an image can also be made with computer graphics, and this is especially useful when it is important to share the idea to a larger audience. If the model permits it, this image may be an animation or an interactive display so that the problem can be examined more generally than a single image would permit. That image, then, can be used by the problem-solver or the audience to gain a deeper understanding of the model and hence of the problem, and the problem can be refined iteratively and a more sophisticated model created, and the process can continue.

This process is the basis for all of the discussions in a later chapter on graphical problem solving in the sciences, but it may be applied to more general application areas. In allowing us to bring the visual parts of our brain and our intelligence to a problem, it gives us a powerful tool to think about the world. In the words of Mike Bailey of the San Diego Supercomputer Center, computer graphics gives us a “brain wrench” that magnifies the power of our mind, just as a physical wrench magnifies the power of our hands.

Overview of the Book

This book is a textbook for a beginning computer graphics course for students who have a good programming background, equivalent to a full year of programming courses. We use C as the programming language in our examples because it is the most common language for developing applications with OpenGL. The book can be used by students with no previous computer graphics experience and less mathematics and advanced computer science studies than the traditional computer graphics course. Because we focus on graphics programming rather than algorithms and techniques, we have fewer instances of data structures and other computer science techniques. This means that this text can be used for a computer graphics course that can be taken earlier in a student’s computer science studies than the traditional graphics course, or for self-study by anyone with a sound programming background. In particular, this book can be used as a text for a computer graphics course at the community college level.
Many, if not most, of the examples in this book are taken from sources in the sciences, and we include a chapter that discusses several kinds of scientific and mathematical applications of computer graphics. This emphasis makes this book appropriate for courses in computational science or in computer science programs that want to develop ties with other programs on campus, particularly programs that want to provide science students with a background that will support development of computational science or scientific visualization work. It is tempting to use the word “visualization” somewhere in the title of this book, but we would reserve that word for material that is primarily focused on the science with only a sidelight on the graphics; because we reverse that emphasis, the role of scientific visualization is in the application of the computer graphics.

The book is organized along fairly traditional lines, treating projection, viewing, modeling, rendering, lighting, shading, and many other aspects of the field. It also includes an emphasis on using computer graphics to address real problems and to communicate results effectively to the viewer. As we move through this material, we describe some general principles in computer graphics and show how the OpenGL API provides the graphics programming tools that implement these principles. We do not spend time describing in depth the algorithms behind the techniques or the way the techniques are implemented; your instructor will provide these if he or she finds it necessary. Instead, the book focuses on describing the concepts behind the graphics and on using a graphics API (application programming interface) to carry out graphics operations and create images.

We have tried to match the sequence of chapters in the book to the sequence we would expect to be used in a beginning computer graphics course, and in some cases the presentation of one module will depend on your knowing the content of an earlier chapter. However, in other cases it will not be critical that earlier chapters have been covered. It should be pretty obvious if other chapters are assumed, and we may make that assumption explicit in some modules.

The book focuses on computer graphics programming with a graphics API, and in particular uses the OpenGL API to implement the basic concepts that it presents. Each chapter includes a general discussion of a topic in graphics as well as a discussion of the way the topic is handled in OpenGL. However, another graphics API might also be used, with the OpenGL discussion serving as an example of the way an API could work. Many of the fundamental algorithms and techniques that are at the root of computer graphics are covered only at the level they are needed to understand questions of graphics programming. This differs from most computer graphics textbooks that place a great deal of emphasis on understanding these algorithms and techniques. We recognize the importance of these for persons who want to develop a deep knowledge of the subject and suggest that a second graphics course can provide that knowledge. We believe that the experience provided by API-based graphics programming will help you understand the importance of these algorithms and techniques as they are developed and will equip you to work with them more fluently than if you met them with no previous background.

This book includes several features that are not found in most beginning textbooks. These features support a course that fits the current programming practice in computer graphics. The discussions in this book will focus on 3D graphics and will almost completely omit uniquely 2D
techniques. It has been traditional for computer graphics courses to start with 2D graphics and move up to 3D because some of the algorithms and techniques have been easier to grasp at the 2D level, but without that concern it is easier to begin by covering 3D concepts and discuss 2D graphics as the special case where all the modeling happens in the X-Y plane.

Modeling is a very fundamental topic in computer graphics, and there are many different ways that one can model objects for graphical display. This book uses the standard beginning approach of focusing on polygon-based modeling because that approach is supported by OpenGL and most other graphics APIs. The discussion on modeling in this book places an important emphasis on the scene graph as a fundamental tool in organizing the work needed to create a graphics scene. The concept of the scene graph allows the student to design the transformations, geometry, and appearance of a number of complex components in a way that they can be implemented quite readily in code, even if the graphics API itself does not support the scene graph directly. This is particularly important for hierarchical modeling, but it also provides a unified design approach to modeling and has some very useful applications for placing the eye point in the scene and for managing motion and animation.

A key feature of this book is an emphasis on using computer graphics to create effective visual communication. This recognizes the key role that computer graphics has taken in developing an understanding of complex problems and in communicating this understanding to others, from small groups of working scientists to the general public. This emphasis is usually missing from computer graphics textbooks, although we expect that most instructors include this somehow in their courses. The discussion of effective communication is integrated throughout several of the basic chapters in the book, because it is an important consideration in graphics modeling, viewing, color, and interaction. We believe that a systematic discussion of this subject will help prepare students for more effective use of computer graphics in their future professional lives, whether this is in technical areas in computing or is in areas where there are significant applications of computer graphics.

This book also places a good deal of emphasis on creating interactive displays. Most computer graphics textbooks cover interaction and the creation of interactive graphics. Historically this was a difficult area to implement because it involved writing or using specialized device drivers, but with the growing importance of OpenGL and other graphics APIs this area has become much more common. Because we are concerned with effective communication, we believe it is critically important to understand the role of interaction in communicating information with graphics. Our discussion of interaction includes a general treatment of event-driven programming and covers the events and callbacks used in OpenGL, but it also discusses the role of interaction in creating effective communications. This views interaction in the context of the task that is to be supported, not just the technology being studied, and thus integrates it into the overall context of the book.

This book’s approach, discussing computer graphics principles without covering the details of the algorithms and mathematics that implement them, differs from most computer graphics textbooks that place a much larger emphasis on understanding these graphics algorithms and techniques. We recognize the importance of these ideas for persons who want to develop a deep knowledge of the subject and suggest that a second graphics course can provide that knowledge.
We believe that the experience provided by API-based graphics programming will help the student understand the importance of these algorithms and techniques as they are developed and will equip someone to work with them more fluently than if they were covered with no previous computer graphics background.