Chapter 7

Simple Graphics and Image Processing

At a Glance

Instructor's Manual Table of Contents

Lecture Notes

Overview

Chapter 7 introduces simple graphics and image processing with Python. Some basic concepts, including the RGB system, analog and digital information, and sampling, are introduced. Students learn to work with graphics through many examples using two non-standard, open-source modules: `turtlegraphics` and `images`.

Objectives

After completing this chapter, students will be able to:

- Use the concepts of object-based programming—classes, objects, and methods—to solve a problem
- Develop algorithms that use simple graphics operations to draw two-dimensional shapes
- Use the RGB system to create colors in graphics applications and modify pixels in images
- Develop recursive algorithms to draw recursive shapes
- Write a nested loop to process a two-dimensional grid
- Develop algorithms to perform simple transformations of images, such as conversion of color to grayscale

Teaching Tips

Simple Graphics

1. Explain that *graphics* is a discipline that underlies the representation and display of geometric shapes in two- and three-dimensional space.

2. Note that a *Turtle graphics* toolkit provides a simple and enjoyable way to draw pictures in a window. Point out that the `turtlegraphics` module used in this chapter is a non-standard, open-source Python module.

Teaching Tip	In addition to `turtlegraphics`, the `turtle` module provides turtle graphics primitives, in both object-oriented and procedure-oriented ways. For more information, visit: www.python.org/doc/2.3/lib/module-turtle.html.

Overview of Turtle Graphics

1. Provide a brief introduction to turtle graphics. Explain that it uses a Cartesian *coordinate system* with the origin (0, 0) located at the center of a window.

Teaching Tip	Students may find it useful to experiment with Logo before using the `turtlegraphics` module. If possible, allocate some time for them to experiment with this programming language. For an online Turtle Logo Applet, visit: www.mathsnet.net/logo/turtlelogo/index.html.

2. Use Table 7.1 to describe the most important attributes of a turtle. Stress that together, these attributes make up a turtle's state.

Turtle Operations

1. Explain that types of objects are called classes and each class includes methods relating to the specific object type.

2. Use Table 7.2 to provide a brief description of some of the most important `Turtle` class methods.

3. Explain that an *interface*, which is the set of methods of a given class, is used to interact with an object. Remind students how to use the docstring mechanism to view an interface.

4. Point out that it is usually sufficient to know the geometry of the desired graphic and the interface of the appropriate object to create graphics in Python.

5. Explain that the `Screen` and `Canvas` classes are also important when working with the Turtle graphics system, and outline the uses of each of these two objects.

Object Instantiation and the `turtlegraphics` Module

1. Explain that before you apply any methods to an object, you must create an *instance* of a class. Introduce the terms *instantiation* and *constructor*.

2. Use a few examples to show how to instantiate and use the `Turtle` class.

3. Note that an attempt to manipulate a turtle whose window has been closed raises an error.

Drawing Two-Dimensional Shapes

1. Explain that many graphics applications use *vector graphics*, which is the drawing of simple two-dimensional shapes such as rectangles, triangles, and circles.

2. Use the example provided in the book to show how to create a function that draws a polygon using the `turtlegraphics` module.

Taking a Random Walk

1. Use the example provided in the book to show how to simulate an animal taking a random walk.

Colors and the RGB System

1. Explain how the *RGB system* encodes color information. Introduce the terms *pixel* and *true color*.

2. Use Table 7.3 to help explain how RGB encodes pixel color information. Note that the total number of bits needed to represent a color value in this system is 24= (8 × 3).

Example: Drawing with Random Colors

1. Use the example provided in the book to show how to use the `penColor` method.

Examining an Object's Attributes

1. Explain the difference between mutator methods and *accessor methods*, which return the value of an object's attributes without altering its state.

2. Give examples of times in which accessor methods could be useful.

Manipulating a Turtle's Screen

1. Remind students of the `Screen` and `Canvas` classes and their uses with respect to a `Turtle` object.

2. Use a few examples to illustrate the characteristics of the `Screen` and `Canvas` classes and how their methods can be used to manipulate the background of a Turtle graphic.

Setting up a `cfg` File and Running IDLE

1. Explain that a configuration file and a different way of launching IDLE are required to run a program with the `turtle` module.

2. Show an example of the turtle.cfg file, and point out some of the initial settings for attributes of `Turtle, Screen,` and `Canvas` objects.

3. Demonstrate to students how an IDLE window should be opened in order to use the `turtle` module. Show students what happens when the IDLE window is opened the regular way and you attempt to use the `turtle` module.

Quick Quiz 1

1. What is a Turtle graphics toolkit?
   Answer: A Turtle graphics toolkit provides a simple and enjoyable way to draw pictures in a window and gives you an opportunity to run several methods with an object. Turtle graphics were originally developed as part of the children's programming language Logo, created by Seymour Papert and his colleagues at MIT in the late 1960s. The name is intended to suggest a way to think about the drawing process. Imagine a turtle crawling on a piece of paper with a pen tied to its tail.

2. The process of creating an object is called _____.
   Answer: instantiation

3. True or False: Many graphics applications use Turtle graphics, or the drawing of simple two-dimensional shapes, such as rectangles, triangles, and circles.
   Answer: False

4. In the RGB system, each color component can range from 0 through _____.
   Answer: 255

Case Study: Recursive Patterns in Fractals

1. Explain that fractals are highly repetitive or recursive patterns. Stress that, even though a *fractal object* appears geometric, it cannot be described with ordinary Euclidean geometry.

2. Use Figure 7.6 to explain that one example of a fractal curve is the *c-curve*.

Teaching Tip	"The geometric characterization of the simplest fractals is self-similarity: the shape is made of smaller copies of itself. The copies are similar to the whole: same shape but different size." *Reference:* http://classes.yale.edu/fractals/

Request

1. Ask students to write a program that allows the user to draw a particular c-curve in varying degrees.

Analysis

1. Use Figure 7.7 to show the user interface for the c-curve program.

Design

1. Use Figure 7.8 and the pseudocode provided in the book to explain how to implement a c-curve function in Python. Point out that because fractals are repetitive, their implementation lends itself to recursive programming.

Implementation (Coding)

1. Explain that the program includes the three function definitions of `cCurve`, `drawLine`, and `main`. Because `drawLine` is an auxiliary function, its definition is nested within the definition of `cCurve`.

Quick Quiz 2

1. What is a fractal?
   Answer: Fractals are highly repetitive or recursive patterns.

2. True or False: A fractal object appears geometric and can be described with ordinary Euclidean geometry.
   Answer: False

3. True or False: One example of a fractal curve is the c-curve.
   Answer: True

4. In Python, the definition of an auxiliary function can be _____ within the definition of another function.
   Answer: nested

Image Processing

1. Describe the uses of digital image processing.

Teaching Tip	For more information on digital image processing, visit: http://www.ciesin.org/docs/005-477/005-477.html.

Analog and Digital Information

1. Explain the difference between *analog* and *digital information*. Introduce the terms: *discrete values*, *continuous range*, and *sampling*.

Sampling and Digitizing Images

1. Provide some insight into how sampling and digitizing of images is performed.

Image File Formats

1. Explain that once an image has been sampled, it can be stored in one of many file formats. Introduce the terms: *raw image, lossless compression, lossy scheme,* and *color palette*.

2. Briefly describe the difference between the JPEG and GIF image formats.

Image-Manipulation Operations

1. Explain that image-manipulation programs either transform the information in the pixels or alter the arrangement of the pixels in the image. List some examples of some common image-manipulation operations.

The Properties of Images

1. Explain that the coordinates of pixels in the two-dimensional grid of an image range from (0, 0) at the upper-left corner to (width-1, height-1) at lower-right corner. Stress that this *screen coordinate system* differs from the standard Cartesian coordinate system used by turtle graphics.

2. Point out that colors of images are typically represented in RGB.

The `images` Module

1. Note that the `images` module is a non-standard, open-source Python tool.

2. Use the example provided in the book to explain that the `Image` class represents an image as a two-dimensional grid of RGB values.

3. Using Table 7.4, briefly describe some of the most useful `Image` methods. Use the example provided in the book to show how to use some of these methods.

A Loop Pattern for Traversing a Grid

1. Describe the difference between a *linear loop structure* and a *nested loop structure*. Stress that the latter is commonly used to traverse a two-dimensional grid of pixels (see Figure 7.11).

2. Use the pseudocode provided in the book to explain that *row-major traversal* is used to develop many of the algorithms that follow.

A Word on Tuples

1. Use a few examples to explain show that a pixel's RGB values are stored in a tuple.

Converting an Image to Black and White

1. Use the example provided in the book to show how to convert an image to black and white.

Converting an Image to Grayscale

1. Use the example provided in the book to show how to convert an image to *grayscale*. Introduce the term *luminance*.

Copying an Image

1. Use one or more examples to show that the method `clone` builds and returns a new image with the same attributes as the original one, although with an empty string as the filename.

Blurring an Image

1. Explain that *pixilation* can be mitigated by *blurring* the affected areas of an image.

2. Provide an overview of the code for blur provided in the book, pointing out the design considerations listed in bullet points on Pages 280-281.

Edge Detection

1. Explain that *edge detection* removes the full colors to uncover the outlines of the objects represented in the image.

2. Use Figure 7.14 to show how setting different luminance threshold values affects the results of the edge detection algorithm.

Reducing the Image Size

1. Explain that the size and the quality of an image on a display medium depend on two factors: the image's width and height in pixels and the display medium's resolution (measured in pixels or dots per inch).

2. Note that a higher DPI causes a sampling device to take more samples (pixels) through the two-dimensional grid.

3. Use the example provided in the book to show how to shrink an image. Note that a size reduction usually preserves an image's *aspect ratio*. Stress that reducing the size of an image throws away some pixel information.

Quick Quiz 3

1. Computers use digital information, which consists of _____ values such as individual integers, characters of text, or bits in a bit string.
   Answer: discrete

2. True or False: GIF stands for Graphic Interchange Format.
   Answer: True

3. Many image-processing algorithms use a(n) _____ loop structure to traverse a two-dimensional grid of pixels.
   Answer: nested

4. A pixel's RGB values are stored in a(n) _____.
   Answer: tuple

Class Discussion Topics

1. Ask your students which module they think they will use more frequently in their programs: `turtlegraphics` or `images`. Why?

2. Have students seen other fractal designs besides the c-curve studied in this chapter? Have them suggest algorithms for coding such fractal designs. Are the suggestions similar to the code for the c-curve fractal?

Additional Projects

1. Ask students to do some research on fractals. They should write a one to two page report on their origins and include one to three examples of their uses.

2. Ask your students to write a function that uses the `turtlegraphics` module to draw a simple button like the one shown below. The function should receive the width and height of the button to be drawn, as well as the x and y coordinates of the bottom-left corner.

Additional Resources

1. Online Logo Turtle Applet:
   www.mathsnet.net/logo/turtlelogo/index.html

2. Fractal:
   http://classes.yale.edu/fractals/

3. Digital Image Processing:
   http://www.ciesin.org/docs/005-477/005-477.html

4. Image Processing with Python SIG:
   www.python.org/community/sigs/current/image-sig/

Key Terms

- **accessor method(s):** A method used to examine an attribute of an object without changing it.
- **analog information:** Information containing a continuous range of values.
- **aspect ratio:** The ratio between the width and height of an image.
- **blurring:** making jagged edges in an image appear softer. Involves loss of information.
- **c-curve:** A fractal shape that resembles the letter C.
- **constructor:** A method that is run when an object is instantiated, usually to initialize that object's instance variables. This method is named _init_ in Python.
- **coordinate system:** A grid that allows a programmer to specify positions of points in a plane or of pixels on a computer screen.
- **discrete value(s):** Individual information values, such as individual integers, characters of text or bits in a bit string.
- **edge detection:** Removing the full colors from an image to uncover the outlines of the objects represented in the image.
- **fractal geometry:** A theory of shapes that are reflected in various phenomena, such as coastlines, water flow, and price fluctuations.
- **fractal object:** A type of mathematical object that maintains self-sameness when viewed at greater levels of detail.
- **graphics:** A discipline that underlies the representation and display of geometric shapes in two- and three-dimensional space.
- **grayscale:** A color scheme that contains various shades of gray in addition to black and white.
- **instance:** A computational object bearing the attributes and behavior specified by a class.
- **instantiation:** The process of creating a new object or instance of a class.
- **interface:** A formal statement of how communication occurs between the user of a module (class or method) and its implementer.
- **lossless compression:** A compression scheme in which no information is lost.
- **lossy scheme:** A compression scheme in which some of the original color information is lost.
- **luminance:** The amount to which the human eye is sensitive to specific colors.

- **mutator method(s):** A method used to change the value of an attribute of an object.
- **nested loop:** A loop as one of the statements in the body of another loop.
- **object-based programming:** The construction of software systems that use objects.
- **origin:** The point (0,0) in a coordinate system.
- **pixel(s):** A picture element or dot of color used to display images on a computer screen.
- **pixilation:** The condition in which an image appears to contain rough, jagged edges.
- **raw image file:** A file that saves all the information sampled from an image.
- **resolution:** A measure for a display medium's accuracy of display. Measured in pixels or DPI.
- **RGB system:** A system for representing colors which represents the intensity of each of the colors red, green, and blue as integers in the range 0-255.
- **row-major traversal:** A grid traversal scheme in which the rows are traversed in the outer loop and the columns in each row are traversed using an inner loop.
- **screen coordinate system:** A coordinate system used by most programming languages in which the origin is in the upper-left corner of the screen, window, or panel, and the y values increase toward the bottom of the drawing area.
- **Turtle graphics:** A set of methods that manipulate a pen in a graphics window.
- **vector:** A one-dimensional array that supports resizing, insertions, and removals.