This chapter focuses on a graphics package called SRGP.

SRGP was written by the authors to demonstrate some of the basics of Raster Graphics primitives.

Since you will be writing programs in OpenGL, and you will learn the primitives there, we will not spend much time in this chapter.

1. Drawing with SRGP

Drawing in an integer raster graphics package is like plotting graphs on graph paper with a very fine grid.

- First you figure out where the origin (0,0) is.
- Then what primitives are provided to you.
- And, what attributes these primitives can have.
2. Basic Interaction Handling

Once you know how to draw basic shapes and text, your next step is to learn how to write interactive programs that communicate effectively with the user.

First we will look at some general guidelines, then we will discuss input devices, and finally we will look at mechanisms for dealing with various aspects of interaction handling.

2.1 Human Factors

The designer of an interactive program must deal with many matters that do not arise in a noninteractive, batch program. They are the so-called human factors of a program, such as its look and feel and its ease of learning and of use.

These are as important as, if not more so than, its functional completeness and correctness.

Techniques for user-computer interaction that exhibit good human factors are studied in more detail in Chapter 8. Those guidelines include:

- Provide simple and consistent interaction sequences.
- Do not overload the user with too many different options and styles.
- Show the available options clearly at every stage of the interaction.
- Give appropriate feedback to the user.
- Allow the user to recover gracefully from mistakes.
2.2 Logical Devices

- A major goal in designing graphics packages is device independence, which enhances portability of applications.
- For output:
  ◆ Primitives are specified in terms of an abstract integer coordinate system,
  ◆ The application does not need to set the individual pixels in the frame buffer.

- For input:
  ◆ Support for a set of logical input devices
    • this shields the application from the details of the physical devices available.
  ◆ Two basic types of devices:
    ◆ Locator:
      • a device for specifying screen coordinates and the state of one or more associated buttons.
    ◆ Keyboard:
      • a device for specifying character string input.
  ◆ Chapter 8 elaborates further on properties of logical devices.

2.3 Sampling Versus Event-Driven Processing

- Two fundamental techniques are used to receive information created by user interactions:
  ◆ Sampling
  ◆ Interrupt-driven interaction
  ◆ Event Processing
**In sampling** (also called **polling**)

- The application program queries the current value of a logical input device (called the **measure** of the device).

- This is performed regardless of whether the device’s measure has changed since the last sampling.

- This is costly for interactive programs.

An alternative is **interrupt-driven** interaction.

- The application enables one or more devices for input and continues normal execution until interrupted by some input event.

- Control then passes asynchronously to an interrupt procedure, which responds to the event.

- For each input device, an **event trigger** is defined (the user action that causes the event to occur).

  - Typically, the trigger is a button push, such as a press of the mouse button, or a press of a keyboard key.

**Event-driven interaction.**

- To free applications programmers from the tricky and difficult aspects of asynchronous transfer of control, many graphics packages offer **event-driven interaction** as a synchronous simulation of interrupt-driven interaction.

- In this technique, the application enables devices and then continues execution.

- In the background, the package monitors the devices and stores information about each event in an event queue.
◆ The application, at its convenience, checks the event queue and processes the events in temporal order.

◆ When an application checks the event queue, it typically can specify whether it would like to enter a wait state if the queue is empty.

◆ In effect, event mode replaces polling of the input devices with the much more efficient waiting on the event queue.

◆ Event driven applications typically spend most of their time in a wait state, since interaction is dominated by think time during which the user decides what to do next.