Program a Game Engine from Scratch

Mark Claypool

Chapter 2 - Setup

This document is part of the book "Dragonfly – Program a Game Engine from Scratch", (Version 10.0). Information online at: `http://dragonfly.wpi.edu/book/`

Chapter 2

Setup

Dragonfly is a text-based game engine, primarily designed to teach about game engine development. That is not to say Dragonfly cannot be used to make games – it certainly can! – but rather Dragonfly is developed with the purpose of *teaching* how to make a game engine.

2.1 Development Environment

Dragonfly development can be done on Microsoft Windows, Linux or Apple Mac OS. Although not as thoroughly tested, Dragonfly development should also work with other Unix operating systems, such as FreeBSD.

Regardless of the development environment, the following tools and libraries need to be installed for Dragonfly development:

- A C++ compiler (e.g., `g++`)
- Standard development libraries (e.g., `stdc++`)
- The Simple and Fast Multimedia Library[1] (SFML).[2]

In addition, the development environment should include:

- Project development software (e.g., `make`)
- A debugger (e.g., `gdb`)

Integrated development environments (also known as IDEs), such as Microsoft Visual Studio, can be used for editing, compiling and debugging. Such systems provide the ability to enter code, but also browse multiple source files, do syntax highlighting and even compile the code without leaving the IDE. However, the lowest common denominator is basically anything that allows entering code, such as a text editor, with compilation done through the command line.

[1] `http://www.sfml-dev.org/`

[2] An earlier version of Dragonfly used Curses.

Instructions for installing the above software depends upon the operating system platform used for development. Some platform-specific setup instructions can be found online at `http://dragonfly.wpi.edu/engine/`. Other Internet forums may provide installation assistance appropriate for the chosen development platform.

2.2 Dragonfly Header Files

All header files (`.h`) for the finished Dragonfly engine are provided in downloadable form, as well as shown in Listings in the Appendix A (page 279). These header files reveal the design of the engine, providing the exact methods and attributes that need to be implemented. While it might be tempting to copy the headers and start implementing the `.cpp` files to support them, this approach is not advised. The header files represent Dragonfly in its final, fully implemented, full-featured form. Development should not start at this point, but instead incrementally build to this final form. Starting from the final header files may be confusing, at best, and render one hopelessly lost, at worst. Instead, the recommended path is to build the engine following the sections in Chapter 4 in order, using the downloaded header files as a reference only.

All header files can be downloaded from the Dragonfly book Web page:

`http://dragonfly.wpi.edu/book/`

Lastly, while the header files are provided, as well as hundreds of code samples in the form of Listings, it is worth noting that the hard part of programming Dragonfly (and most other complex pieces of software) is *understanding* the design. Understanding takes the most effort and the most time. The second hardest part is still not entering the code – rather, it is debugging the code that implements the design. Not just compilation errors that are usually relatively easy to fix, but functional and logical errors that arise (often while trying to understand the Dragonfly design being implemented!). After those two parts, then actually entering the code, typing it in or cutting and pasting it into the right files, may take the next most time. Typically, this will be relatively quick compared to the first two parts. Such is the nature of programming.