



# Chapter 31

## Next Steps

There are lots of other things I'd like to discuss in this book, but I don't want it to get too long and expensive. So now that you've gotten this far, I'll suggest some other books to read next.

As I've mentioned before, much of this book is based on *How to Design Programs* [?]. I recommend getting that book (it's available on the Web for free at <http://www.htdp.org>) and reading the following chapters:

- 14-16 about binary and  $n$ -ary trees,
- 25-32 about “generative recursion” and “accumulative recursion”, which enable you to do a number of things more easily than you could with the “structural recursion” we've used in this book
- 39-43 about object-oriented programming and how it works.

(Some of these chapters refer to an old “draw.ss” teachpack, which has since been replaced by the `picturing-programs` teachpack. Ignore the graphics parts of the book.)

As I write this, the authors of *How to Design Programs* are working on a second edition. It's not finished yet, but it should be good. Do a Web search on “htdp2e” and you'll find it.

If you want more exercises involving animation, see *How to Design Worlds* [?], on the Web for free at <http://world.cs.brown.edu>.

If you want to learn Java, there are lots of good (and not-so-good) books on the market, but there's one that builds on what you've learned in this book: *How to Design Classes* [?]. As I write this, *How to Design Classes* isn't finished, but you can find out about it and request a draft copy at <http://www.ccs.neu.edu/home/vkp/HtDCH>.

Racket comes with lots of powerful tools we haven't discussed. Most of these are documented in the Help Desk. Here are some that might interest you:

- The `picturing-programs` library allows you to write programs that run on several computers at once, communicating over a network. It takes a *client/server* approach: one computer (the “server”) keeps track of information that all the computers share, and each “client” computer shows its own view of that shared information. This could be used, for example, to develop a multi-player game. To find out more about this, look up the `2htdp/universe` module in the Help Desk, then look for the section entitled “The World is not Enough”.

- Racket comes with a built-in Web server, with which you can easily write powerful, interactive Web applications. A Racket feature called *continuations*, which most languages don't have, makes it *much* easier to write Web programs. Open the Help Desk and follow the link “Continue: Web Applications in Racket”.
- In this book, we've written “contracts” in comments. In fact, Racket allows you to write contracts as part of the code, and will automatically enforce them for you: look in the Help Desk for `define/contract` and `define-struct/contract`. Or open the Help Desk, follow the link “Guide: Racket” and look for the chapter on “Contracts”.
- Modules allow you to break a large program into separate source files, each with a well-defined interface (in the same way that each individual function has a well-defined interface). Open the Help Desk, follow the link “Guide: Racket”, and look for the chapter on “Modules”.
- Racket supports class-based object-oriented programming, similar to classes in Java and C++ but more flexible. Open the Help Desk, follow the link “Guide: Racket”, and look for the chapters on “Classes and Objects” and “Units”.
- One of Racket's most powerful features is called “macros”: they're basically functions which, rather than taking in and returning values, instead take in and return Racket code. They allow you to write functions that don't pre-evaluate their arguments (like `or`, `and`, `define`, and `define-struct`), and even completely change the syntax of the language. Most programming languages don't offer anything similar, so if you're trying to solve a problem that lends itself to macros, it may be *much* easier and simpler to solve in Racket than in most other languages. Open the Help Desk, follow the link “Guide: Racket”, and look for the chapter on “Macros”.
- If you want to write Racket programs that run on their own, without the user needing to know about DrRacket, open the “Guide: Racket”; first read the chapter on “Modules” (above) and then read the chapter on “Running and Creating Executables”.
- The graphics and GUI programming we've been doing in this book are pretty simple: great for a beginning programming course, but if you want to do industrial-strength graphics programming, especially 3-D graphics, you'll need more flexibility. Open the Help Desk and look for the section on “GUI and Graphics Libraries”.