base class called `Star`. A hierarchical block that consists of multiple stars is a `Galaxy`, and a top-level system representation is a `Universe`. Unlike physical galaxies, Ptolemy galaxies can also contain other galaxies.

The domain used in our DSP classes, the synchronous dataflow (SDF) domain [Lee87a,b], is one of the most mature in Ptolemy. It is used for signal processing and communications algorithm development, and has particularly good support for multirate algorithms [Buc91]. A dynamic dataflow (DDF) domain, which is not used in our DSP classes, extends SDF by allowing data-dependent flow of control. Many other Ptolemy domains provide alternative modeling and implementation techniques.

A key idea in the Ptolemy project is to mix modeling techniques, implementation languages, and design styles, rather than trying to develop one, all-encompassing technique. The Ptolemy software is used as a laboratory by the Ptolemy research project for experimenting with such mixtures. Such mixtures, however, are not needed for straightforward modeling of DSP algorithms.

1.5 Additional documentation

The main Ptolemy documentation, called *The Almagest*, is divided into three volumes. It is available on-line at http://ptolemy.eecs.berkeley.edu. The first volume is a user's manual. It is sufficient for users who do not plan to extend the system by adding code. It includes brief documentation of the most commonly used domains, and brief summaries of stars, galaxies, and demonstration programs that are distributed with the system.

The second volume is a programmer's manual. It includes chapters on writing new stars, writing targets, defining customized user interfaces by writing new Tcl/Tk code, and defining new domains.

The third volume is the kernel manual. It details every C++ class defined in the Ptolemy kernel. It also gives full documentation for the classes supporting code generation. These classes provide the utilities used to build application-specific environments.

1.6 Acknowledgments

Ptolemy is a team effort in every sense. See *The Almagest* for a full description of individual contributions.

Berkeley before many of the features that we have come to expect from user interfaces. It was very innovative for its time, and has served us well, but it is clearly obsolete. Nonetheless, with a little patience, you get used to it. It helps to learn to use the keyboard accelerators rather than the popup menu.

Ptolemy crash more frequently than most commercial software. FIXME.

1.4 Organization of the Software

The overall organization of the Ptolemy system is shown in figure 1-1. A typical use of Ptolemy involves starting two Unix™ processes, as shown in figure 1-1(a), by running `pigi` (Ptolemy interactive graphical interface). The first process contains the `vem` user interface and the `oct` design database [Har86], and the other process contains the Ptolemy kernel.

The `pigi` program, which is aptly named, contains all Ptolemy domains. As such, it is a rather large program that requires a good bit of memory. For the DSP classes, a much smaller version, called `ptiny`, is sufficient. This is the version you should use.

In Ptolemy, applications are described as interconnections of blocks, much like the block diagrams you find in a DSP textbook. Blocks can be hierarchical, as shown in figure 1-2. The lowest level of the hierarchy, as far as Ptolemy is concerned, is derived from a kernel

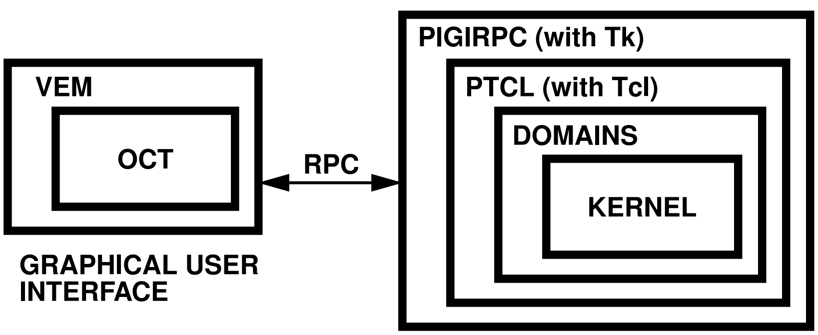

FIGURE 1-1: The overall organization of Ptolemy, showing two processes that communicate.

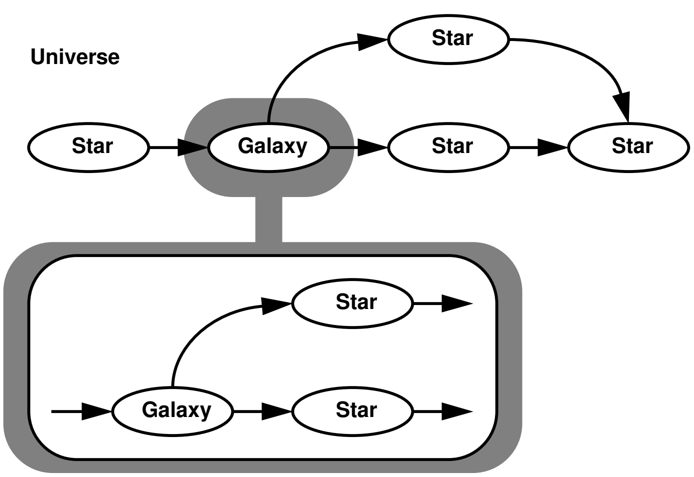

FIGURE 1-2: A complete Ptolemy application (a Universe) consists of a network of Blocks. Blocks may be Stars (atomic) or Galaxies (composite).

Chapter 1. Micro Overview

1.1 Introduction

Ptolemy is a research software infrastructure upon which specialized design environments (called *domains*) can be built. The key part of the software infrastructure, called *the Ptolemy kernel*, is made up of a family of C++ class definitions. Domains are defined by creating new C++ classes derived from the base classes in the kernel. Domains can operate in either of two modes:

- Simulation — A scheduler invokes code segments in an order appropriate to the model of computation.
- Code generation — Code segments in an arbitrary language are stitched together to produce one or more programs that implement the specified function.

This greatly shortened version of the manual is concerned with only one of the simulation domains, the so-called "SDF domain." This domain is used for exploring signal processing algorithms, and is used at Berkeley in EE123 and EE225a, the undergraduate and graduate signal processing classes. For more information about other parts of Ptolemy, see *The Almagest*, the full Ptolemy manual.

1.2 Why use Ptolemy?

A few comments about the rationale for using Ptolemy in these classes is in order. Many alternatives exist for exploring signal processing algorithms. For example, the Matlab program provides an excellent computational engine with a rich built-in library of signal processing functions (in the Signal Processing Toolkit). In fact, it is possible to exploit Matlab from within Ptolemy. At the other extreme, signal processing implementation can be explored using programmable DSPs, specialized microprocessors used in embedded signal processing systems.

Matlab, with all its strengths, is a traditional, imperative computation engine that, while useful for algorithm exploration, is not very useful for real-time or embedded system implementation. In view of this, the signal processing industry has largely migrated to a different paradigm for modeling systems, one based on block-diagram visualization coupled with hardware and software synthesis.

The Ptolemy SDF domain is a representative of a genre of such signal processing software environments. Notable commercial representatives include SPW, from the Alta Group at Cadence Design Systems, and Cossap from Synopsys. Why don't we use the commercial versions? Frankly, we find them harder to use and the software harder to maintain.

1.3 Idiosyncrasies

Ptolemy has many idiosyncrasies, so some tolerance and patience is requested. A better user interface is under development, but will probably not be ready yet for some time. The current user interface is more than 10 years old, having been developed in the CAD group at