Volume 3
Programmer’s Manual
software organization
writing stars
infrastructure
data types
Tcl/Tk
domains
code generation

Volume 4
Kernel Manual
detailed documentation
of all C++ classes
defined in the kernel.
Essential for defining
Targets and Domains.
The Almagest

Volume 1
User’s Manual
pigip
tcldomains
vem
pxgraphinstallation

Volume 2
Star Atlas
detailed documentation
of all stars
(not up-to-date)
Within Each Domain Directory

$PTOLEMY/src/domains/xxx

demo
icons
kernel
stars
targets

applications in the domain
the icons used by pigi
core code defining the domain
stars in the domain
(optional) additional targets

The stars in a domain are an excellent resource for learning about how to write custom stars for the domain.

READ THE SOURCE
The Domains Directory

- bdf  Boolean dataflow domain
- cg   base classes for code generation
- cg56 Motorola DSP56000
- cg96 Motorola DSP96000
- cgc  code generation in C
- cp   communicating processes (sun)
- ddf  dynamic dataflow
- de   discrete-event
- mq   message queue
- sdf  synchronous dataflow
- silage Silage code generation
- sproc Sproc DSP (obsolete)
- thor hardware simulation
- vhdlb behavioral modeling in VHDL
- vhdlf functional modeling in VHDL
- xxx demonstration new domains
The Source Tree

$PTOLEMY/src

compat
header files for odd configurations
domains
the code for each of the domains
filters
outside filter design programs
kernel
the Ptolemy kernel
libgantt
the Gantt chart display tool
octtools
our subset of the oct tools
pigiExample
example to make a custom pigi
pigiRpc
source for pigiRpc program
pigilib
source for most of pigi
ptcl
source for ptcl
ptklib
shared Tcl/Tk code
ptlang
the preprocessor for star writing
tcltk
source for Tcl and Tk
thread
code used by the CP domain
utils
utilities
xgraph
the pxgraph program
To run ptolemy, set the environment variable PTOLEMY to the root directory of the installation and put $PTOLEMY/bin in your path.

The preferred installation location is /users/ptolemy.
Ptolemy is an Extensible Open Architecture

Extensions that are possible:

• applications
• a library of galaxies
• application builders (using ptcl)
• a library of custom stars
• customized user interfaces (using Tcl/Tk)
• simulation managers (Targets)
• domains
• foreign simulators and/or synthesis tools
Domains in Ptolemy

- Code generation domains:
  - CGC
  - CG56
  - CG96
  - Silage
  - VHDLF
  - VHDLB
  - Sproc
  - CG

- Code generation kernels:
  - PTOLEMY KERNEL
  - SDF
  - BDF
  - DDF
  - PN

- Additional domains:
  - MDSDSDF
    - multidimensional SDF
  - Thor
    - circuit simulation
  - DE
    - discrete-event
  - CP
  - DMM
    - communicating processes
  - design methodology management

- Descriptions:
  - process networks
  - dynamic dataflow
  - Boolean dataflow
  - synchronous dataflow
Standard forms of the Ptolemy Executables

pigi
• All domains and targets plus the GUI.

ptrim
• SDF, BDF, DDF, DE, CGC and HOF domains plus GUI.

ptiny
• SDF and DE domains plus GUI.

ptcl
• All domains, no GUI.
The user interface runs as two communicating Unix processes.

A minimal configuration has only a command-line interface.
Assume "XXX" and "YYY" are two domain names.
Hierarchical Abstraction

Examples of Derived Classes

- class Star:: Block
- class XXXStar:: Star
- class Galaxy:: Block
- class Universe:: Galaxy, Runnable

XXXStar

Galaxy

XXXStar

Universe

XXXStar

Galaxy

XXXStar

XXXStar

XXXStar
Example: NamedObject

class NamedObj {
public:
    NamedObj ();
    NamedObj (const char* n, Block* p, const char* d);

    const char* name () const;
    const char* descriptor () const;
    Block* parent () const;
    void setParent (Block* my_parent);
    void setName (const char* my_name);

    virtual const char* className () const;
    virtual int isA (const char* cname) const;
    virtual void initialize () = 0;

private:
    const char* nm;
    Block* prnt;
    const char* myDescriptor;
};
Example: NamedObject

class NamedObj {
    public:
    NamedObj ();
    NamedObj (const char* n, Block* p, const char* d);
    
    const char* name () const;
    const char* descriptor () const;
    Block* parent () const;
    
    void setParent (Block* my_parent);
    void setName (const char* my_name);
    
    virtual const char* className () const;
    virtual int isA (const char* cname) const;
    
    virtual void initialize () = 0;
    
    private:
    const char* nm;
    Block* prnt;
    const char* myDescriptor;
}
Example: NamedObject

class NamedObj {
public:
    NamedObj ();
    NamedObj (const char* n, Block* p, const char* d);

    const char* name () const;
    const char* descriptor () const;
    Block* parent () const;

    void setParent (Block* my_parent);
    void setName (const char* my_name);

    virtual const char* className () const;
    virtual int isA (const char* cname) const;

    virtual void initialize () = 0;

private:
    const char* nm;
    Block* prnt;
    const char* myDescriptor;
}
Some Kernel Classes (Booch Notation)

Abstract base class: Cannot be instantiated.

C++ class:
- Members & Methods
- Public, Protected, Private

Inheritance: A derived class has all the properties of the parent, plus more.

Data abstraction: Hiding the irrelevant and amplifying the essential.

Polymorphism: Objects with identical interface have different implementations.
Object-Oriented Programming

- inheritance
- data abstraction
- polymorphism

The idea is to use a heterogenous software environment to develop heterogeneous designs. The interaction between different modules in the software environment is managed through object-oriented principles.
Ptolemy C++ Base Classes Support this Abstract Syntax

Block
- initialize()
- run()
- wrapup()

PortHole

Geodesic

Plasma

Particle
- type()
- print()
- initialize()
A Flexible Abstract Syntax

Hierarchical Graphs
Heterogeneity in System-Level Design

System-level modeling:
- Imperative
- FSMs
- Dataflow
- Discrete event

Synthesis:
- Partitioning
- Compiler
- Software synthesis
- ASIC synthesis
- Logic synthesis

Detail modeling and simulation:
- Execution model
- ASIC model
- Logic model

Cosimulation:
- Symbolic
Ptolemy Simulation

- ATM network with three 4x4 switches
- Detailed model of each switch with queueing and routing protocols.
- Dummy traffic (Poisson arrivals) to create congestion.
- Test traffic (video and audio) to measure subjective performance.

Multiple models of computation may be used in the same system. Here, dataflow is used for signal processing, while a timed discrete-event system models a communication network.
Heterogeneous Real-Time Prototyping

Sparc C

DSP Card M56K
Baseline Cosimulation

Features Today

- Instruction set model of the processor.
- RTL model of chip input/output.
- High-level software development.
- Compatibility with ASIC simulators.

Tomorrow

- Compiled simulation.
- More abstract modeling.
- Cycle-based simulation.

The image above shows a Ptolemy simulation of a multiprocessor system where the software is synthesized (including partitioning) from a block diagram.
## Major Activities

- **Formal methods**
  - **Dataflow** (process networks, synchronous, Boolean, multidimensional, ...)
  - **Control** (hierarchical FSMs, Esterel, synchronous languages, ...)
  - **Partitioning and scheduling of dataflow graphs** (optimize IPC, memory, ...)
  - **Programming languages** (higher-order functions, polymorphism, ...)

- **Algorithm-level design methodology**
  - **Mixing models of computation** (discrete-event, FSMs, dataflow, imperative, ...)
  - **Animation and design visualization** (Tcl/Tk, Matlab, xv, ...)
  - **Mixing domain-specific tools** (filter design, Matlab, Mathematica, ...)
  - **Visual programming** (dataflow, FSMs, regularity, recursive, functional, ...)

- **System-level design methodology**
  - **Synthesis of embedded software** (high-level, assembly, ...)
  - **Design complexity management** (data, tool, flow, methodology, ...)
  - **Hardware/software codesign** (DesignMaker, GCLP partitioning, ...)
  - **Architecture design and performance modeling** (OT principle, VHDL, ...)

---

**UNIVERSITY OF CALIFORNIA AT BERKELEY**
Ptolemy as a Tool and as a Laboratory

Ptolemy is
- Extensible
- Publicly available
- An open architecture
- Object-oriented

Allows for experiments with:
- Models of computation
- Domain-specific tools
- Design methodology
- Software synthesis
- Hardware synthesis
- Cosimulation
- Codesign

Rationale for heterogeneity: specialized models are
- More useful to the system-level designer
- More amenable to hardware and software synthesis.
Agenda (Afternoon)

1:30

  9. Writing Tcl/Tk blocks (15 minutes)          Lee
     10.Code generation concepts (30 minutes)    Williamson
     11.Writing blocks for the CGC and VHDL domains (45 min.) Williamson

2:45 — Break

3:15

  12.Defining Targets (15 minutes)              Lee
  13.Defining Domains (30 minutes)              Wilson
  14.Interfacing to foreign design environments (30 minutes) Wilson
  15.Debugging (15 minutes)                    Lee
  16.Preview of Coming Attractions (15 minutes) Lee

5:00 Adjourn
Agenda (Morning)

8:30
1. Overview of the Ptolemy project (45 minutes) Lee
2. Guide to the GUI (30 minutes) Evans
3. Higher-Order Functions (15 minutes) Evans

10:00 — Break

10:15
4. Dataflow models of computation (30 minutes) Lee
5. The synchronous dataflow domain (15 minutes) Lee
6. Writing custom stars for the SDF domain (15 minutes) Lee
7. Boolean and dynamic dataflow domains (15 minutes) Lee

11:30
8. The discrete-event domain (30 minutes) Evans

12:00 — Lunch
Ptolemy Tutorial

Brian Evans
Edward A. Lee
Mike Williamson

UC Berkeley

Dave Wilson

Berkeley Design Technology, Inc.