# Appendix E. Glossary

**$PTOLEMY** The directory in which the Ptolemy software is installed.

**actor** An atomic (indivisible) function in a dataflow model of computation.

**ATM** (1) Asynchronous transfer mode network protocol. (2) A sub-domain of the synchronous dataflow and discrete-event domains to provide the infrastructure for simulating ATM networks.

**auto-fork** A fork star that is automatically inserted when a single output is connected to more than one input.

**base class** A C++ object that is used to define common interfaces and common code for a set of derived classes. An object may be a base class and a derived class simultaneously.

**BDF** A domain using the Boolean-controlled dataflow model of computation. This domain attempts to use compile-time scheduling, but will fall back to run-time scheduling if necessary.

**behavioral modeling**
System modeling consisting of functional specification plus modeling of the timing of an implementation (cf. functional modeling).

**Block** The base class defined in the kernel for stars, galaxies, universes, and targets.

**block** A star or a galaxy.

**Boolean-controlled dataflow**
A model of computation that includes synchronous dataflow, but adds actors that may or may not produce or consume tokens on any given input or output. Whether these actors produce or consume tokens depends on a Boolean signal.

**code generation**
The synthesis of a standalone implementation in some target language from a network of Ptolemy blocks.

**code generation domain**
A domain that supports code generation, but not simulation.

**CG** A domain that defines many of the base classes and schedulers used in code generation domains. It has no direct application by itself.

**CG56** A domain that synthesizes assembly code for the family of Motorola DSP56000 digital signal processors. It uses the synchronous dataflow model of computation.

**CG96** A domain that synthesizes assembly code for the family of Motorola DSP96000 digital signal processors. It uses the synchronous dataflow model of computation.

**CGC** A domain that synthesizes C code. It uses the synchronous dataflow or Boolean-controlled dataflow model of computation.

**CG-DDF** A code generation domain that uses the dynamic dataflow model of computation. This has not been maintained beyond version 0.4.1 of Ptolemy.

**codesign** The simultaneous design of the software and hardware composing a system.

**communicating processes**
A model of computation in which multiple processes execute concurrently and communicate with one another by passing messages.

**compile-time scheduling**
A scheduling policy in which the order of execution of blocks is precomputed when the execution is started. The execution of the blocks thus involves only sequencing through this precomputed order one or more times (cf. run-time scheduling).

**contents facet**
An oct facet that defines the physical appearance of a schematic.

**CP** A simulation domain using the communicating processes model of computation. Each star forms a process that runs under the Sun lightweight process library.

**derived class** A C++ object that is derived from some base class. It inherits all of the members and methods of the base class.

**dataflow** A model of computation in which actors process streams of tokens. Each actor has one or more firing rules. Actors that are enabled by a firing rule may fire in any order.

**DDF** A simulation domain that uses the dynamic dataflow model of computation.

**DE** A simulation domain that uses the discrete-event model of computation. In the DE domain, particles transmitted between blocks represent events that trigger changes in system state. Events carry an associated timestamp, and are processed in chronological order.

**discrete event**
A model of computation used to model systems that change state abruptly at arbitrary points in time, such as queueing networks, communication networks, and computer architectures. A block is enabled when an event at one of its inputs is the "oldest" event in the system, in that its timestamp has the smallest value. Once enabled, the block may be executed, and in the process may produce more events.

**domain** A specific implementation of a model of computation.

**Domain** The base class in the Ptolemy kernel from which all domains are derived.

**drag** The action of holding a mouse button while moving the mouse.

**dynamic dataflow**
A model of computation supporting any computable firing rule for actors. This model of computation requires run-time scheduling.

**event** A particle generated by a block in a discrete-event model of computation. This particle carries a timestamp.

**event horizon**
The interface between domains that manages the flow of particles from one domain to another.

**facet** A schematic, palette, or icon as represented in `oct`. In `pigi`, a facet is exactly that represented by one `vem` window.

**FFT** The Fast Fourier Transform is an efficient way to implement the discrete Fourier transform in digital hardware.

**firing** A unit invocation of an actor in a dataflow model of computation.

**firing rule** A rule that specifies how many tokens are required on each input of a dataflow actor for that actor to be enabled for firing.

**fork star** A star that reads one input particle and replicates it on any number of outputs.

**functional modeling**
System modeling that specifies input/output behavior without specifying timing (cf. behavioral modeling).

**galaxy** A block that contains a network of other blocks.

**Galaxy** The class (derived from Block) in the Ptolemy kernel that represents a network of other blocks.

**Gantt chart** A graphical display of a parallel schedule of tasks. In Ptolemy, the tasks are the firings of stars and galaxies.

**higher-order functions**
Functional programming constructs that apply a function a determined number of times to one or more streams of inputs. Examples of higher-order functions from Lisp include `mapcar` and `apply`.

**HOF** A domain implementing higher-order functions that are expanded at compile-time and incur no run-time overhead. HOF stars are typically embedded in other domains, and provide graphical expression of parameterized parallel, cascaded, and recursive structures.

**homogeneous synchronous dataflow**
A particular case of the synchronous dataflow model of computation, where actors produce and consume exactly one token on each input and output.

**icon** A graphical object that represents a single block or palette.

**interface facet**
A facet that defines the physical appearance of an icon (cf. contents facet).

**Itcl** [incr Tcl], an object oriented extension to Tcl.

**iteration** A set of executions of blocks that constitutes one pass through the precomputed order of a compile-time schedule.

**kernel** The set of classes defined in the directory `$PTOLEMY/src/kernel`.

**layer** In vem, a color with a given precedence. Colors with higher precedence will obscure colors with lower precedence.

**master** In `vem`, the interface and contents facet referred to by an icon. The master is represented by an absolute Unix path name pointing to the directory in which the facet is stored.

**masters** A program for examining and changing the list of masters (see above) that make up a schematic or palette.

**MDSDF** A simulation domain that uses a multidimensional extension to the synchronous dataflow model of computation. Actors in MDSDF consume data defined on rectangular grids, e.g. a subblock in an image.

**member** A C++ object that forms a portion of another object.

**method** A function defined to be part of an object in C++.

**model of computation**
A set of semantic rules defining the behavior of a network of blocks.

**net** A graphical connection between ports in vem.

**object** A data type in C++ consisting of members and methods. These members and methods may be private, protected, or public. If they are private, they can only be accessed by methods defined in the object. If they are protected, then they can also be accessed by methods in derived classes. If they are public, then they can be accessed by any C++ code.

**oct** A design database developed by the CAD Group at U. C. Berkeley. Oct is used to store graphical representations of Ptolemy applications.

**octtools** A collection of CAD tools based on the `oct` database. Some programs from the octtools distribution are used within Ptolemy.

**palette** A facet that contains a library of icons.

**parameter** The initial value of a state.

**particle** A datum (e.g. a floating-point value) communicated between blocks.

**pepp** The program that translates stars written in the Thor domain to C++.

**pigi** The Ptolemy interactive graphical interface. Pigi is implemented as a shell script that starts `vem` and `pigiRpc`.

**pigiRpc** The program that forms the core of the Ptolemy graphical user interface. PigiRpc communicates with `vem` via a remote procedure call interface. It is mainly responsible for handling Ptolemy-specific commands from `vem` and translating `oct` representations of Ptolemy systems into a form executable by the Ptolemy kernel.

**Plasma** A class in the Ptolemy kernel that serves as a repository for used particles of any particular types. When new particles of the appropriate type are needed, they are taken from the Plasma, if possible, thus avoiding memory allocation.

**PN** A simulation domain based on the process networks computational model. Each star forms a process under this domain.

**port** An input or output of a star or galaxy.

**PortHole** The base class in the Ptolemy kernel for all ports.

**ptcl** A textual, interactive command interpreter for Ptolemy. As the name implies, ptcl is based on Tcl.

**ptlang** (1) A schema language used to define stars in Ptolemy. (2) The program that translates stars written in the ptlang language to C++.

**ptplay** A program to play sound on the workstation speaker.

**PTOLEMY** An environment variable with value equal to the name of the directory in which the Ptolemy system is installed.

**Ptolemy** A design environment that supports simultaneous mixtures of different models of computation. Ptolemy has been developed at the University of California at Berkeley. The Ptolemy design environment is named after the second century Greek astronomer, mathematician, and geographer.

**pxgraph** A plotting program used by several standard Ptolemy stars.

**real time** The actual time (cf. simulated time).

**RTL** Register-transfer level description of digital systems. This kind of description is used by the Thor domain.

**run-time scheduling**
A scheduling policy in which the order of execution of the blocks is determined "on-the-fly," as they are executed (cf. compile-time scheduling).

**Scheduler** An object associated with a domain that determines the order of execution of blocks within the domain. Domains may have multiple schedulers.

**schematic** A block diagram.

**SDF** A simulation domain using the synchronous dataflow model of computation.

**Silage** (1) A functional language developed by Paul Hilfinger at U. C. Berkeley for specifying signal processing systems. It is used primarily as input for VLSI synthesis tools. (2) A code generation domain in Ptolemy that synthesizes Silage code and uses the synchronous dataflow model of computation.

**simulated time**
In a simulation domain, the real number representing time in the simulated system (cf. real time).

**simulation** The execution of a system specification (a Ptolemy block diagram) from within the Ptolemy process (i.e., without generating code and spawning a new process to execute that code).

**simulation domain**
A domain that supports simulation, but not code generation.

**snap** In `vem`, an invisible grid defining the points at which graphical objects can have endpoints or corners.

**star** An atomic (indivisible) unit of computation in a Ptolemy application. Every Ptolemy simulation ultimately consists of executing the methods of the stars used to define the simulation.

**Star** The base class in the Ptolemy kernel for all stars.

**state** A member of a block that stores data values from one invocation of the block to the next.

**State** The base class in the Ptolemy kernel for all states.

**stop time** Within a timed domain, the time at which a simulation halts.

**synchronous dataflow**
A dataflow model of computation where the firing rules are particularly simple. Every input of every actor requires a fixed, pre-specified number of tokens for the actor to fire. Moreover, when the actor fires, a fixed, pre-specified number of tokens is produced on each output. This model of computation is particularly well-suited to compile-time scheduling.

**target** An object that manages the execution of a simulation or code generation process. Thus, for example, in code generation, the target would be responsible for compiling the generated code and spawning the process to execute that code, if desired.

**Target** The base class in the kernel for all targets.

**Tcl** Tool command language, a textual, interpreted language developed by John Ousterhout at U.C. Berkeley. Tcl is embedded in both `pigi` and `ptcl`.

**Thor** A register transfer level digital hardware simulator from Stanford University. Thor is incorporated as a domain within Ptolemy.

**timestamp** A real number associated with a particle in timed domains that indicates the point in simulated time at which the particle is valid.

**timed domain**
A domain that models the evolution of a system in time.

**Tk** An X windows toolkit for Tcl. Tk is embedded in `pigi`, which uses it extensively. The interactive sliders, buttons, and plotting capabilities of `pigi` are implemented in Tcl/Tk.

**tkoct** An experimental front-end to Ptolemy that replaces `vem`. It is a Tk interface to schematics stored as `oct` facets. It executes universes by calling `ptcl` to evaluate the `ptcl` description of the universe.

**token** A unit of data in a dataflow model of computation. Tokens are implemented as particles in Ptolemy.

**Tycho** A graphical development environment for Ptolemy that is implemented in Itcl.

**universe** An entire Ptolemy application.

**URT** The Utah Raster Toolkit for image and video processing. It is used by the image processing stars in the synchronous dataflow domain. The multidimensional synchronous dataflow domain treats images as matrices and does not use the Utah Raster Toolkit.

**vem** A graphical editor for objects stored under `oct`. The `vem` schematic capture interface is part of the octtools distribution from U. C. Berkeley, and forms a significant part of `pigi`.

**VHDL** The VHSIC hardware description language, a standardized language for specifying hardware designs at multiple levels of abstraction.

**VHDLF** A code generation domain for functional modeling of hardware. This domain synthesizes a system description in VHDL.

**VHDLB** A code generation domain for behavioral modeling of hardware. This domain synthesizes a system description in VHDL

**wormhole** A star in a particular domain that internally contains a galaxy in another domain.