Report Number: WUCS-TM-92-01

1992-08-01

Pavane User's Manual

Jerome Y. Plun

This report describes the Pavane visualization system. Pavane is set to a set of tools allowing on to 1) simulate the execution of a computation specified with the Swarm model and 2) visualize the state of the computation by mean of a set of rules. Pavane also provides the means to visualize other computations. This manual describes how to initiate a Pavane session to visualize a Swarm computation as well as the role of each of the component of such as session.

School of Engineering & Applied Science

Pavane User's Manual

Jerome Y. Plun

WUCS-TM-92-01

August 1992

Department of Computer Science
Washington University
Campus Box 1045
One Brookings Drive
Saint Louis, MO 63130-4899

Abstract

This report describes the Pavane visualization system. Pavane is to a set of tools allowing one to 1) simulate the execution of a computation specified with the Swarm model and 2) visualize the state of a computation by mean of a set of rules. Pavane also provides the means to visualize other computations This manual describes how to initiate a Pavane session to visualize a Swarm computation as well as the role of each of the component of such as session.

Correspondence: All communications regarding this report should be addressed to

Dr. Gruia-Catalin Roman
Department of Computer Science
Washington University
Campus Box 1045
One Brookings Drive
Saint Louis, MO 63130-4899

office: (314) 935-6190
secretary: (314) 935-6160
fax: (314) 935-7302

roman@swarm.wustl.edu

Table of Content

List of Figures

1. Introduction

Pavane is to a set of tools allowing one to 1) simulate the execution of a computation specified with the Swarm model [5] and 2) visualize the state of a computation by mean of a set of rules [3, 4]. Pavane also provides the means to visualize other computations [1].

This manual describes how to initiate a Pavane session to visualize a Swarm computation as well as the role of each of the component of such as session.

2. Launching a Pavane Session

A Pavane session is composed of a Swarm computation, a visualization, and one or more displays in which the visualization is rendered. The Swarm Language Reference Manual [2] describes the syntax of a Swarm program and a visualization program, and how to produce the corresponding executable modules.

Once a program and a visualization have been created—referred to as `<Swarm>` and `<Visualization>` in the following— one initiate a Pavane session by typing the following command at a Unix prompt:

`<Swarm> <Visualization>` **ArgList**

where **ArgList** is a subset of the following options:

`-swarm` *<PgmArgList>*`--`	lists the parameters of the Swarm program. It can not be used in conjunction with the `-both` option.
`-visual` *<PgmArgList>*`--`	lists the parameters of the visualization. It not be used in conjunction with the `-both` option.
`-both` *<PgmArgList>* `--`	lists the parameters of both the Swarm program and the visualization. It can not be used in conjunction with either the `-swarm` or `-visual` options.
`-keep`	the rendering display will not terminate automatically once the Swarm program is exited
`-sync`	the Swarm program won't perform a step until the previous one has been fully rendered
`-go`	the Swarm program starts in Go mode (repeatedly performs steps until paused or no more steps can be performed). This cancels any previous `-step` option.
`-step` [*<N>*]	the Swarm program starts in Step mode and will perform *<N>* steps before requesting more user input. If *<N>* is not specified, one step will be performed. This cancels any previous `-go` option.

None of these options should occur more than once in **ArgList**. *<PgmArgList>* should match the list of program parameters described in the Swarm computation or visualization.

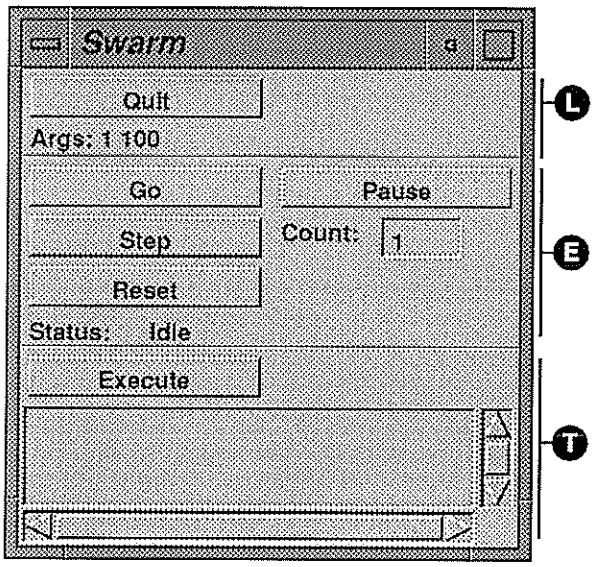
Once a Pavane session has been initiated, several windows will be created on your screen for the different compoents of the session. Most of these windows will start in iconic form. The following sections will describe the role of each window.

3. Swarm windows

A Swarm program creates a window to control the execution of the program and a window showing the content of the dataspace.

3.1 Swarm Controls

The Swarm control window is shown in Figure 3.1. It is composed of three subareas, respectively Launch (L), Execution (E), and Transactions (T). The Launch subarea displays the parameters specified by the `-swarm` or `-both` options on the command line. The Quit button terminates the Pavane session, except for the displays if the `-keep` option was selected.



L Launch Subarea
E Execution Subarea
T Transactions Subarea

Figure 3.1: Swarm Controls

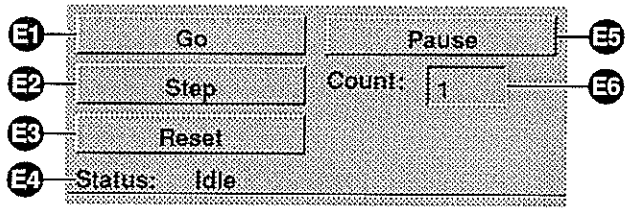
3.1.1 Execution subarea:

The Execution subarea allows the user to control the pace of execution of the Swarm program. An execution is a serie of steps, each step corresponding to a state change in the Swarm program. The components of the Execution subarea are shown in Figure 3.1.1. The Go button (E1) starts a continuous execution that can be interrupted using Pause, Step or Reset. Pause (E5) has no effect outside of a continuous execution. Step (E2) starts a limited execution and let Swarm perform the number of steps specified in E6 (the default is 1). Reset (E3) reinitializes the Swarm execution to its initial state.

Throughout the different execution modes, a meaningfull status is displayed at the bottom of the Execution subarea (E4). Possible status messages are:

`Idle`	Nothing is in progress
`Going...`	Continous execution mode
`Pausing...`	Informing the Swarm program that the continous execution is to be stopped. This status is followed by `Idle` once the Swarm program has acknowledge the command
`Stepping... x out of y`	Limited execution of `y` steps, currently execution x^{th} step. This status is followed by `Idle` once all `y` steps have been executed.

`Resetting...`	Informing the Swarm program that the current execution is to be reset. This status is followd by `Idle` once the Swarm program has acknowledged the command.



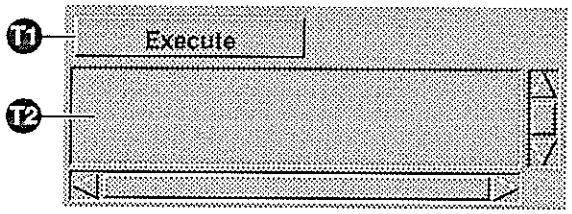
E1	Continuous execution
E2	Limited execution
E3	Reinitialization of an execution
E4	Status of the execution
E5	Interruption of a continuous execution
E6	Size of a limited execution in steps

Figure 3.1.1: *Execution* Subarea

3.1.2. Transaction subarea

The Transaction subarea allows the user to bypass the built-in scheduling mechanism to control which transaction will be selected for execution in the following step. As shown in Figure 3.1.2, the Transaction subarea contains a list of enabled transactions (T2). Clicking twice on a transaction will result in the Swarm program performing a step with the selected transaction. Clicking once on a transaction will simply highlight it. The step of the computation will then be performed only once the Execute button (T1) has been selected.

To be able to use the Transaction subarea, the current execution must be idle or paused from a Go or Step command. Selecting a transaction for execution will change the Execution status to "`Stepping... 1 out of 1`".

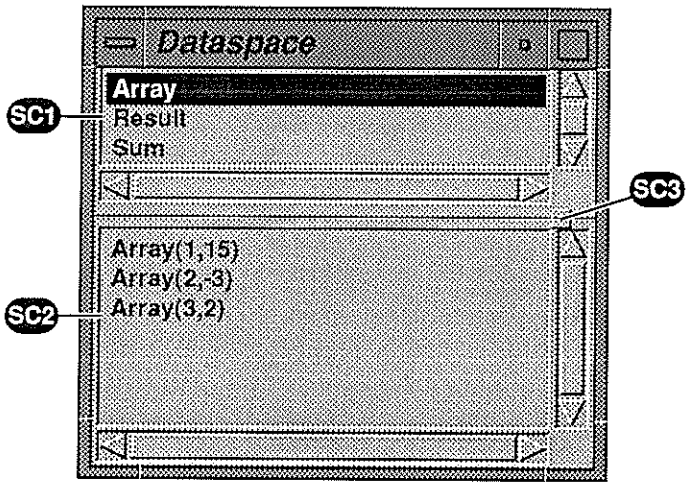


T1	Execute the highlighted transaction
T2	List of enabled transactions

Figure 3.1.2: *Transactions* Subarea

3.2 Swarm Dataspace

As shown in Figure 3.2, the Swarm dataspace window provides a list of all possible tuple and transaction types (SC1) and, for the selected types, a list of current existing tuples and transactions (SC2). The selected types are highlighted. Both lists can be widened by widening the window. Expanding the window vertically will only expand the list of instances. The separator (SC3) between the lists can be used to modify the amount of space allocated to each list.



SC1 List of tuple and transaction types
SC2 Tuple and transaction instances
SC3 Separator

Figure 3.2: Swarm Dataspace

4. Visualization windows

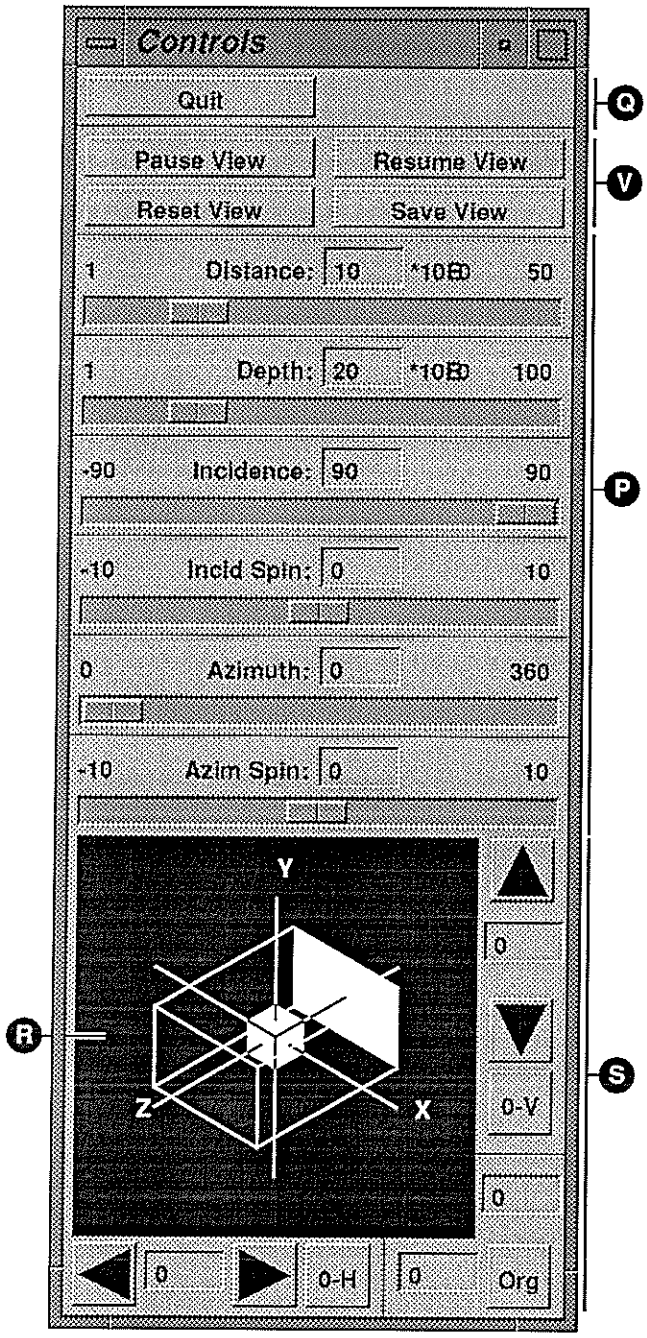
A visualization program creates two windows, initially iconic, which allows the user to look at the current and previous content of the visual dataspace. These windows are titled "Visual" and "old.Visual". Their appearance and behavior are entirely similar to the Swarm dataspace window.

5. Display windows

A display program is started for each "window" defined in the visualization program. Each one is composed of two windows, one containing all the controls necessary to modify the viewing conditions, and the other providing a rendering area in which the visualization is displayed.

5.1 Display Controls

The control window for a display is shown in Figure 5.1. It is composed of 5 subareas, respectively Quit (Q), View (V), Polar (P), Scroll (S), and Referential (R).



Q Quit Subarea
V View Subarea
P Polar Subarea
S Scroll Subarea
R Referential Subarea

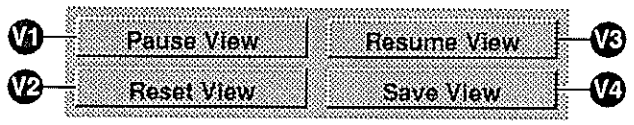
Figure 5.1: Display Controls

5.1.1. Quit subarea

The "Quit" button is provided to terminate the corresponding display. It is enabled only once the Swarm program and visualization have terminated but the displays were kept (as specified with the command line option `-keep`). If this option is not selected, terminating the Swarm program will also terminate all the displays. When the option is selected, each display must be terminated independently.

5.1.2. View subarea

The View subarea, shown in Figure 5.1.2., is used to control the execution of the Display module. "Pause View" (V1) suspends the updating of the display until resumed using "Resume View" (V3). "Reset Display" (V2) resets all the display parameters to the values defined at the start of the visualization. These parameters are modified using the Scroll and Polar subareas. "Save View" (V4) copies of the current display into a file.



V1	Suspend display of Visualization
V2	Reset display of Visualization
V3	Resume display of Visualization
V4	Record display of Visualization

Figure 5.1.2: *View* Subarea

5.1.3. Polar subarea

The position of the viewpoint in the visualization world is defined using a polar coordinate system, i.e. a distance from the origin and two angles: incidence and azimuth. Figure 5.1.3.1 is a representation of the components of the coordinate system.

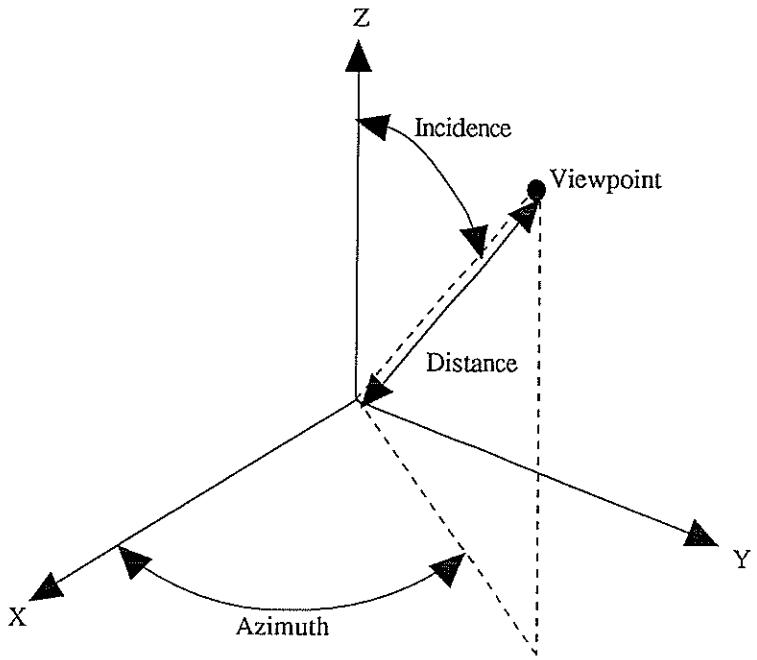
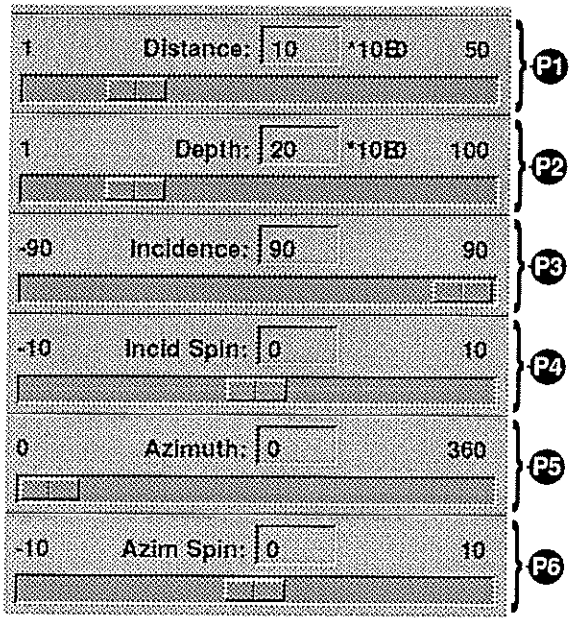


Figure 5.1.3.1: Polar coordinate system

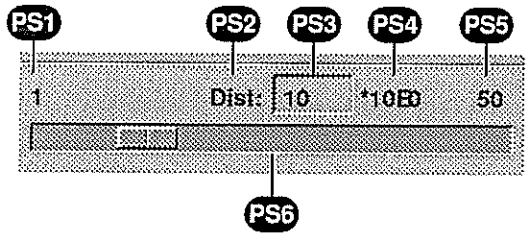
The polar subarea, shown in Figure 5.1.3.2, provides the user with the controls necessary to position the viewpoint by setting the distance (P1), the incidence (P3) and the azimuth (P5), as well as specify the depth of vision (P2)—which is measured from the viewpoint toward the origin—and a possible spin (P4 and P6) which corresponds to a temporal modification of the incidence or the azimuth. The parameter values shown in Figure 5.1.3.2 correspond to the default values.



P1 Controls for *Distance* parameter
P2 Controls for *Depth* parameter
P3 Controls for *Incidence* parameter
P4 Controls for *Incidence Spin* parameter
P5 Controls for *Azimut* parameter
P6 Controls for *Azimuth Spin* parameter

Figure 5.1.3.2: *Polar* Subarea

As shown in Figure 5.1.3.3, a parameter, whose name and value are indicated in (PS2) and (PS3), is defined between a minimum (PS1) and maximum value (PS5), and can be modified by typing a legal value in (PS3) or by moving the scale (PS6). An exponent (PS4) can be provided for the distance and depth to modify the scaling factor of the corresponding parameter. The actual value of the parameter is Value*10^{Exp}.

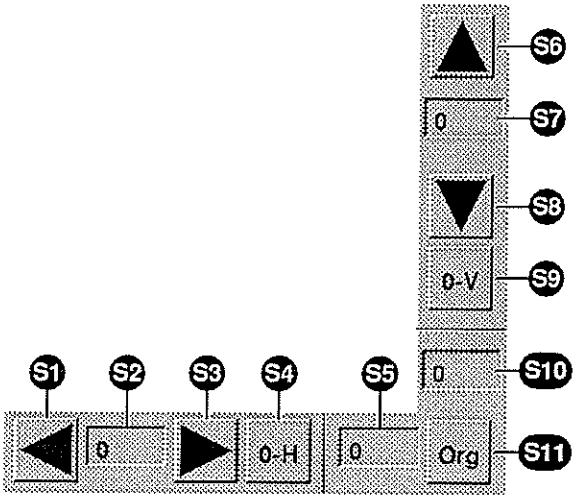


PS1 Minimum value of the parameter
PS2 Name of the parameter
PS3 Current value of the parameter
PS4 Exponent value of the parameter (if any)
PS5 Maximum value of the parameter
PS6 Scale of values from min to max

Figure 5.1.3.1: Components of a *Polar* Scale

5.1.4. Scroll subarea

The Scroll subarea, shown in Figure 5.1.4, allows the user to move the display parallelepiped based on the planar coordinate system of the viewing rectangle. The "Left" (S1), "Right" (S3), "Up" (S6) and "Down" (S8) scrolling buttons force the display to shift accordingly. A single click on a button moves the display by 1 unit. If the mouse is maintained down in a scrolling button, the display is repeatedly moved by increasingly larger amounts (starting with 1). The horizontal (S2) and vertical (S7) offset of the display relative to the current origin is indicated between the respective scrolling buttons. The horizontal (S4) and vertical (S9) reset buttons shift the display so that its center corresponds to the display origin. The display origin is initially on the line perpendicular to the display plane and intersecting the visualization origin. It can be set to any value by scrolling the display using the scrolling buttons and selecting the origin setting button (S11) at the appropriate location. The offset of the current origin based on the default one is shown in two fields (S5 and S10).



S1 Shift display to the left
S2 Horizontal offset of the display
S3 Shift display to the right
S4 Reset horizontal offset of the display
S5 Horizontal offset of the origin
S6 Shift display up
S7 Vertical offset of the display
S8 Shift display down
S9 Reset vertical offset of the display
S10 Vertical offset of the origin
S11 Set display origin

Figure 5.1.4: *Scroll* Subarea

5.1.5. Referential subarea

The Referential subarea, shown in Figure 5.1.5., provides a 3D representation of the displayed volume of the visualization. The three axis labelled X, Y and Z indicate the 3D coordinate system used in the visualization rules. The wire frame parallelepiped represents the portion of 3D space shown on the display, with the rearmost plane indicated by a solid rectangle and the viewer's eye centered in the opposite side. The position of the wire frame is updated as display parameters are modified using the Polar subarea. There is no scaling or scrolling information.

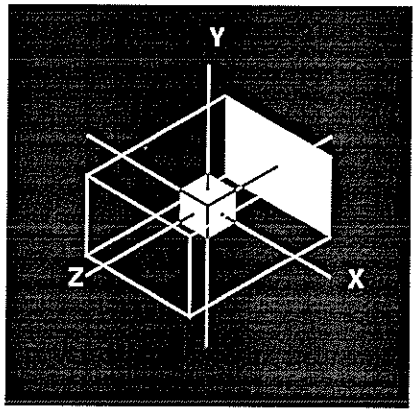


Figure 5.1.5: *Referential* Subarea

5.2 Display Rendering

When a visualization is initiated, a display window is created based on the dimensions given at start-up time. The viewing area displayed corresponds to a parallelepiped which "front" is the display and with the depth defined in the Polar subarea. The content of the display corresponds to the part of the visualization falling in this parallelepiped. The use of the Scroll and Polar subarea allows the user to select what part of the visualization should be visible.

The rendering window can be resized but the same viewing area will stil be displayed (i.e., the previous display is expanded, not widened). The resizing capabilities of the rendering window can be specified in the visualization program (no resizing, ratio of width and height, ...).

6. Bibliography

[1] Cox, K. C., "CtoVis: An Interface between C Programs and Pavane Visualizations," Washington University, Dept of Computer Science, Technical Memo WUCS-TM-92-04, 1992.

[2] Plun, J. Y., Cox, K. C., and Wilcox, C. D., "Swarm Language Reference Manual," Washington University, Dept of Computer Science, Technical Memo WUCS-TM-92-03, 1992.

[3] Roman, G.-C., and Cox, K., "A Declarative Approach to Visualizing Concurrent Computations," *Computer*, vol. 22, no. 10, pp. 25-36, 1989.

[4] Roman, G.-C., and Cox, K., "Declarative Visualization in the Shared Dataspace Paradigm," *11th International Conference on Software Engineering*, IEEE, Pittsburgh, PA, pp. 34-43, 1989.

[5] Roman, G.-C., and Cunningham, H. C., "Mixed Programming Metaphors in a Shared Dataspace Model of Concurrency," *IEEE Transactions on Software Engineering*, vol. 16, no. 12, pp. 1361-1373, 1990.