Particle3 User's Guide

Rev 2.2

Derek Plansky
Massachusetts Institute of Technology

January 5, 1995

Table of Contents

Chapter 1

Introduction

Particle3 is an interactive scientific visualization tool that was developed to fulfill the growing need for the visualization of particle based data. With the rapid advances in computational power, both at the workstation and the massively parallel computing levels, fields such as computational plasma physics and non-equilibrium chemistry have been performing increasingly large particle based simulations.

Present state of the art applications include the plasma particle-in-cell (PIC) method and Discrete Monte Carlo Simulation (DSMC) method using approximately 10^6 to 10^8 particles.

Particle3 can be used to efficiently visualize and interrogate particulate data by allowing a user to examine both steady and unsteady data in both position and velocity spaces. Particle3 includes a number of tools that enables the user to interrogate the systems of particles so that regions of interest may be discovered and examined.

Particle3 is based upon the visualization package VISUAL3. VISUAL3 has a very general structure that allows it to visualize almost any type of continuous field (i.e. scalar, vector, steady or unsteady). Particle3 is designed as an extension to VISUAL3 to allow the visualization /interrogation of non-continuous particle data. As an extension, Particle3 was modeled after VISUAL3 as much as possible, in order to prevent a user familiar with VISUAL3, from having to learn an entirely new visualization interface. This means that like VISUAL3, Particle3 is a collection of routines, that when initialized through well defined access points, allows the user to visualize his or her data.

1.1 Requirements

There are certain minimum requirements that must be satisfied in order to use Particle3. The primary requirement is to have a copy of VISUAL3, version 2.20 or higher. This in itself imposes several system and hardware sub requirements. These are included in the appendix of this manual. In order to operate Particle3, VISUAL3 must be running simultaneously. The data set requires an arbitrary grid and at least one scalar parameter or field defined over that grid in addition to the particulate data.

1.2 Assumptions

In the following sections of the manual, it has been assumed that the reader has some familiarity with VISUAL3 and will understand such terms as cutting plane, iso-surface, and 3D window. For those less familiar with the program, a glossary of explanations has been included later in this manual. All italicized words are included in the glossary

Chapter 2

Particle3 User Interface

2.1 Basic Structure

VISUAL3 can visualize almost any form of continuous data, but it can not visualize particle based data. Continuous data is traditionally discretized into a mesh or grid, with the data being stored either at the mesh nodes or in the mesh cells. The VISUAL3 term for the data structures is *space*, i.e. the data is stored in *node space* or *cell space*. Particle based data, or particulate data, is quite different in structure. Each particle has a corresponding position and velocity, in addition to some other data. Since the data is based on particles rather than grid nodes or cells, the set of all particulate data is called the *particle space.*

Each element in the particle space is one particle, and has associated with it three Cartesian position components (x, y and z), three Cartesian velocity components (v_x, v_y and v_z) and possibly other particle based quantities. The six dimensional space spanned by the position and velocity components is referred to as the *phase space.*

The remaining data belongs to the particle *parameter space* and can be any particle based parameter that the Particle3 user deems of interest. In Particle3 this data is stored and handled separately from the *phase space* data and generally is used to distinguish groups of particles from one another.

Although the particle space data is stored as real numbers, for generality, the *parameter space* can store real valued and integer valued data. The real type is used to represent non discrete valued parameters associated with each particle, such as energy, charge or mass . The integer type is for discrete data, such as particle species or type.

Although the user may specify as many different data types or fields as desired (barring memory constraints) only one field may be active at any one time.

2.2 Tools

In Particle3 a tool is anything that changes the state of visualization either internally or externally. For convenience the tools have been divided into two categories for easier reference. The first category is visualization tools, tools that allow the user to examine the entire particle space on the screen, in the various VISUAL3 windows. The second category is interrogation tools, tools that allow the user to query and select subsets of the *particle space* to provide more detailed views.

The following table contains a list of all the tools included in Particle3 and which which category the tool belongs to. Following the list is a detailed description, by category, of the purpose and the use of each tool.

Table 1 Particle3 Tools

Tool	Type
1. Particle position plots	visualization
2. Particle phase plots	visualization
3. Particle Histogram	visualization
4. Data field change	visualization
5. Data field limits.	interrogation
6. Projection Plane	interrogation
7. Filtering structure definition	interrogation
8. Activate filtering	interrogation

2.2.1 Interactive assistance

Although this is not really a tool as defined above, the help function of Particle3 has been included here. The description of its use has been included in the same format as the other tools for consistency. In order to remain distinct from VISUAL3, where possible the tool activation keys have been kept as upper case letters where possible ('P' instead of 'p' etc.).

• Get help

This command allows the user to get a listing of all commands and their corresponding keys, that are activated from a particular window. This works identically to the help command in VISUAL3 with the addition of the Particle3 commands prepended to the list of available keys.

Use: Striking '?' in any window will list all the VISUAL3 and Particle3 commands available from that window.

2.2.2 Visualization Tools

As mentioned previously, visualization tools are tools which allow the user to view the entire data set at once, and provide general views and information about the data viewed.

2.2.2.1. Particle position plots

This tool draws all the particles, or all the unfiltered particles if filtering is activated, in the 3D window. The particle s are located according to their x, y and z positions. The particles are coloured according to the active data field and its current limits.

Use: Pressing 'P' in the 3D window *toggles* this tool on and off.

2.2.2.2. Particle phase plots

This tool allows the user to look at the phase space representation of the particles in 2 dimensions. The axes are chosen by the user and may be any combination of x, y, z, v_x, v_y or v_z. The plots are normalized to fit in the square 2D window, and the zero axes are drawn in the window as a reference (unless either zero axis is out of the window, in which case the axes is drawn along the appropriate edge of the window). This tool will only activate if the 2D window is empty.

Use: The function is activated by pressing 'P' in the 2D window. Particle3 then asks which state will be shown on the x and y axis of the plot and then renders the particles in the 2D window. As with the 3D phase plot the particles are coloured according to the active data field and its current limits.

2.2.2.3. Particle Histogram

A histogram, showing the distribution of particle density versus the active data field, is drawn in the 1D window. The routine divides the active parameter's range into evenly spaced discrete units, and then determines the number of particles in each subunit. This number of particles is used to represent the density. If the filtering is activated in the 2D window the histogram shows filtered data. This tool will only activate if no other tool is active in the 1D window.

Use: 'H' is the toggle character for the histogram when pressed in the 2D window. Particle3 queries the user to determine if the count axis should be logarithmically scaled. If no, the scaling is linear.

2.2.2.4. Data field change

This fundamental 'tool' allows the user to select any predefined data field as the active one. Data fields are used to colour the particles and as parameters by which to filter. When Particle3 is initialized the user specifies the number of data fields, their titles, their initial limits and the ASCII of the character that will activate that field. When a predefined key is pressed, the corresponding data field becomes the active one. Only one field may be active at any one time.

Use: When any of the keys ,corresponding to the user defined data fields, are pressed in the 3D window, that data field becomes the active field (changing the colouring, etc.).

2.2.3 Interrogation Tools

2.2.3.1 Data field limits.

The colour scale limits for the active data field are changed through this command. The initial limits are set when Particle3 is initialized. The limits determine the colour variation of the particles based on a linear variation on the colour scale in the key window. Any value that is greater than the upper limit or less than the lower limit is coloured as though it was at the upper or lower limit respectively.

Use: Pressing 'L' in the key window will cause Particle3 to ask the user, in the text window, for the lower , then the upper limit. In each case the default for each limit is its current value.

2.2.3.2 Projection Plane

The projection plane allows the user to select an arbitrary rectangular volume in the 3D window and to project all of the particles within it to one of the faces in the volume.

The projection plane is based on the cutting plane from VISUAL3. The volume is defined by the cutting plane, a square surface, and a thickness, which represents the height of the resulting rectangular prism. Once defined, an outline of the volume will appear in the 3D window as a reference. Any particle inside the volume is projected onto the cutting plane along the shortest distance to the plane. The particles are the drawn in the 2D window.

In order to see the particles clearly it may be necessary to turn off VISUAL3's cutting plane rendering in the 2D window.

Use: The user must first position the cutting plane and activate it. Pressing 'J' then activates the projection plane. Particle3 will request a thickness, and once that is entered the projected particles will be rendered in the 2D window. Pressing 'J' again deactivates this tool.

2.2.3.3 Filtering structure definition

This tool allows the user to select a subset of the particle space based on one or more filtering criterion. The subset is used when the users desires to view just the selected particles rather than the entire set. The criteria are based on bounds or limits. Any particle that falls inside the specified bounds, becomes a member of the subset. Multiple criteria are handles through layering. Each criterion is a layer, and for a particle to be a member of the subset it must pass through each layer, i.e. it must satisfy each criterion.

There are two types of filter available to the user, the parameter based filter and the volumetric based filter. The parameter based filter allows the user to select all particles which have a parameter value between an upper and lower limit. The filtering parameter is always the active data field. The volumetric filter allows the user to select all particles within a user specified rectangular volume (the same way the projection plane defines a volume).

In addition to adding layers the user may also remove layers, either individually or all at once. When removing layers individually, the most recently defined layer is always removed first. Also, since up to 10 layers of filtering may be used, the user may print a list of the current filter limits, including parameter name, layering order and limits.

Use: All key presses are in the 2D window unless otherwise noted. To add parameter filter layer the user presses '*', then enters in the lower and upper limit, when prompted. To remove levels the '-' (minus) key removes the most recently added layer and the '/' (slash) key clears all the filter information. To get a list of all current filtering levels press 'S'.

A volumetric filter layer is activated from the 3D window. To add a volume based filter, the user must first activate the projection plane. Once the outline of the volume is visible, pressing 'V' will select all the particles inside the defined volume as the next filter layer.

2.2.3.4 Activate filtering

The particle selection during the filter definition is usually transparent to the user. This command allows the user to see the effect of the filtering on any of the data shown in the 3, 2 and 1D windows. The user may *toggle* the filtering in either the 3D or 2D window.

Once activated only the subset of particles that met all the filter criteria will be shown in that window. If activated in the 2D for example, the phase plot (if active) will only show the unfiltered particles and the histogram will only show the distribution of the unfiltered particles etc. Although the default is off, the first press of '*' in the 2D window activates the 2D window filtering. Also clearing all levels with '/' or removing the last level with '-' deactivates the filtering in both windows.

Use:Pressing 'T' in either the 3D or the 2D window *toggles* the filtering in that window.

Chapter 3

Programming Particle3

3.1 Programming Notation

As with VISUAL3, Particle3 was designed to be accessible from both FORTRAN and C. FORTRAN is more restrictive in argument passing and naming, and therefore it has shaped the programming interface. Also the following routines are from the FORTRAN programmer's viewpoint. The internal subroutines of Particle3 have names that begin with p3_ and have common blocks which begin with 'P3': such names should be avoided in the application program. Provided with the Particle3 release are several example source codes. These are well documented and can be used as a template for future applications.

In describing the arguments of the routines in the next sections, the notation from the VISUAL3 reference manual has been used. For convenience a description of the notation has been included here. Variable names are followed by either 'i' or 'o', indicating the variable is an input or an output of that routine, or whether it is set by the routine. The variable is then followed by a letter representing the variable type (C=character, I=integer, L=logical, R=real) and its dimensionality.

3.2 Programmer called routines

There are two predefined routines in Particle3 that the user can interface with to activate the utility, p3_init and p3_update. p3_init is a required routine, for it links the users code to the Particle3 subroutines. The other routine, p3_update, is only required when the particle data is unsteady. The routine parameters and calling protocols are described below.

3.3.1 P3_INIT

Parameter structure: p3_init (maxnum, iopt, pnkeys, pkeycode, ptitle, ptype, plims)

This routine allocates the memory required for the particle data storage. It must be called from the main program before V3_INIT is called.

maxnum(i) :I: The maximum number of particles that will be in memory at any time in the visualization

iopt (i):i: A system parameter

iopt = 0 steady data

iopt ≠ 0 unsteady data

pnkeys(i):I: the number of different particle data fields (keys)

pkeycode (i):I(pnkeys): The ascii code for each key

ptitle(i):C(pnkeys): The key title. not to exceed 20 characters

ptype(i): I(pnkeys): The data type

ptype(m)=1 real data

ptype(m)=2 integer data

plims(i):R(2, pnkeys): The initial parameter limits

plims(1,m) = the initial minimum for key 'm'

plims(2,m) = the initial maximum for key 'm'

3.3.2 P3_UPDATE

Parameter structure: p3_update (numpart)

This routine controls the data calls to P3load and P3data during unsteady applications. It also ensures that the filter information is current for each time step. It should be called from within V3_UPDATE.

numpart (i):I: number of particles in memory at the current time step
(must be less than or equal to MAXPART)

3.3 Programmer defined routines.

The following routines are called from within the Particle3 code but are left for the user to define.

3.3.1 P3LOAD

Parameter structure: p3load(number,phspace)

This routine loads in the basic 6 Cartesian components into the array phspace.

numpart (o):I: number of particles in memory at the current time step

phspace (i):R(6,numpart): The particle space array

phspace(1,i): x position of particle 'i'

phspace(2,i): y position of particle 'i'

phspace(3,i): z position of particle 'i'

phspace(4,i): x velocity of particle 'i'

phspace(5,i): y velocity of particle 'i'

phspace(6,i): z velocity of particle 'i'

3.3.2 P3DATA

Parameter structure: p3data (numpart, rspace, ispace, phspace, currkey)

This routine is used to load the desired data field. It the programmer's responsibility to ensure that either rspace or ispace is loaded. Also phspace is provided for access, but it should not be changed in this routine. Any changes to phspace should be done in p3load.

numpart (o):I: The number of particles in memory at the current time step

rspace(i):R(numpart): The array for the storage of real data (ptype (currkey) =1)

ispace(i):I(numpart): The array for the storage of integer data (ptype(currkey)=2)

phspace(o):R(6, numpart): the phase space array (see P3load)

currkey(o):I:the current key index

3.4 Advanced Programming Notes

3.4.1 Multiply defined VISUAL3 routines

For programmers familiar with VISUAL3, incorporating Particle3 into existing VISUAL3 applications will be very straightforward. There may however be some slight problems if the existing application codes use certain

VISUAL3 routines from The Advanced Programmer's Guide. Since Particle3 also uses these routines, there is a possibility that, between the user defined application and Particle3, there will be some multiply defined routines. This should be avoided to avoid compilation errors. The four VISUAL3 programmer defined routines that are used in Particle3 are: V3PEVENTS, V3DRAW3D, V3DRAW2D and V3PROBE.

If Particle3 is to be added to an existing VISUAL3 based application, and the previously mentioned VISUAL3 routines are defined in that code, the programmer must incorporate their versions of the VISUAL3 routines into the Particle3 versions of those routines. For convenience the Particle3 file P3_events.f contains all four of the mentioned routines. The file P3_events.f is well documented, in order to make modifications straightforward.

3.4.2 Archive Libraries

In order to simplify the linking process and the generation of executables, archive libraries are often used. If all the Particle3 routines are put into one library and linked at compile time a slight problem may arise. Due to the structure of VISUAL3, which provides access to customize almost any parameter of a visualization, VISUAL3 provides its own 'dummy' routines if some of the customization routines remain unused. If all of the Particle3 routines are in a library the necessary linking dependencies are not established when the program is compiled. This potential problem may be solved by including the P3_events.o object file in the link list, after the main program object files, rather than in an archive library.

Appendix A

How to get VISUAL3

Instructions for getting VISUAL3, pV3 and Particle3 distributions and the minimum requirements for each.

Currently Available (at no charge):

Visual2 DECStation SGI 4D-IRIS [1] HP 9000/700 IBM RS/6000

VISUAL3 DECStation[2] SGI 4D-IRIS[3] HP 9000/700[4]

pV3 SGI 4D-IRIS[3] as server; many clients

Particle3 same as VISUAL3

[1] Must be running IRIX 4.0 (or higher)

[2] Must be running ULTRIX 4.2a (or higher) and have PXG/Turbo graphics or a PXG with ZBuffer and 24-bit color.

[3] Must be running IRIX 4.0 (or higher) and have at least 24 bits of color. Indigos must have either XS24 graphics with ZBuffer, XZ, Elan or Extreme graphics.

[4] Must have CRX24, CRX24Z or CRX48Z and PEX runtime system.

BY NETWORK:

1) ftp onto "raphael.mit.edu" (18.78.0.66) using the (anonymous) ftp account (the user name is "ftp"). Use any password (not a null string).

2) This is a restricted ftp account, you will not be able to get a directory listing. Get the ASCII file "VERSIONS". This file lists the current releases with the associated files.

3) Get the binary files that you want. These are compressed "tar" files that must be uncompressed and extracted to get to the distributions (which contain post-script files for the manuals). NOTE: In the future you can periodically check "VERSIONS" to see if new releases have issued.

4) Send me by e-mail; (1) your complete name and mailing address, (2) FAX number if appropriate, (3) telephone number, (4) e-mail address, and (5) which distributions you have taken.

BY MAIL:

1) Send me a tape with return shipping. I can handle most types of tapes. FedEx, DHL and etc. work best, because you can pre-pay the return envelope and all I have to do is call them to pick up the package when the tape is cut.

2) Request the specific distributions for the supported machines and include your telephone number, e-mail address, and FAX number if appropriate.

Bob Haimes email: haimes@orville.mit.edu
MIT 37-467 FAX: (617) 253-0823
70 Vassar St Tel: (617) 253-7518
Cambridge, Ma 02139

Appendix B

GLOSSARY

The following is a collection of words that are assocated with VISUAL3 and Particle3. They are intended as a simple definition that should be sufficient for the reader to understand the concept/purpose behind them. For a more detailed description it is suggested that the reader reference the VISUAL3 User's and Programmer's Manual .

Cell space: The data structure containing all of the field information associated with each cell. Used when data is cell based.

Cutting plane: A tool that allows the user to interrogate the internal 'structure' of a scalar field using a flat square plane. Once positioned and activated by the user it shows, in the 2D window, how the scalar field varies on the plane.

Iso-surface: A 3D surface of a constant scalar parameter. The user selects the scalar field and the value of the constant and VISUAL3 draws the resulting surface in the 3D window.

Node space: The data structure containing all of the information associated with each node. Each element represents a different node. Only used when the data is node based.

*Particle space:** The data structure used by Particle3 to store the particulate data. It is divided into two subsets, the *phase space* and the *parameter space.*

*Phase space:** The data structure containing the cartesian velocity and position comp-onents.Each element is one particle. The data is used for positioning the particles when rendered.

*Parameter space:** The data structure containing the data used to color and filter the particles.It may contain elements of the phase space.

Windows: Refers to any one of the six windows that may be activated by VISUAL3. The *3D window* shows the spatial orientation of the user's data. The *2D window* shows some

form of 2 dimensional view of the data (from the cutting plane for example). The *1D window* is similar to the 2D window, except it shows a 1D representation of the current data set (the pressure variation along a streamline for example). The *dialbox window* contains the controls for the orientation of the data within the 3D, 2D and key windows.

The *key window* contains the colour scale and the associated numerical values as well as several other useful parameters. Finally the *text window* simply contains the text queries and messages that VISUAL3 outputs.

* These terms are defined in this document, rather than the VISUAL3 User's Guide