

# SPHRAY-1.0 Users Guide

Gabriel Altay

March 4, 2008



# Contents

<b>1</b>	<b>Introduction</b>	<b>5</b>
1.1	Requirements . . . . .	5
1.1.1	Fortran 90 Compiler . . . . .	5
1.1.2	GLUT . . . . .	5
1.1.3	f90gl . . . . .	5
1.2	Obtaining SPHRAY . . . . .	6
1.2.1	Source Code . . . . .	6
1.2.2	Example Snapshots . . . . .	6
1.3	Unpacking . . . . .	6
<b>2</b>	<b>Configuration</b>	<b>7</b>
2.1	Makefile Options . . . . .	7
2.1.1	Preprocessor Macros . . . . .	7
2.1.2	Compiler Flags . . . . .	8
2.1.3	OpenGL Paths . . . . .	9
2.2	Config File . . . . .	9
2.2.1	Initialization . . . . .	9
2.2.2	Input . . . . .	11
2.2.3	Raytracing . . . . .	12
2.2.4	Gas Composition . . . . .	13
2.2.5	Output . . . . .	14
2.2.6	SPH . . . . .	15
<b>3</b>	<b>Usage</b>	<b>17</b>
<b>4</b>	<b>Data Structures</b>	<b>19</b>
4.1	Basic Data Structures . . . . .	19
4.2	Advanced Data Structures . . . . .	20



# Chapter 1

## Introduction

The **S**moothed **P**article **H**ydrodynamics **R**aytracer (SPHRAY) is a code written by Gabriel Altay and Inti Pelupessy designed to perform 3D, time dependent, radiative transfer calculations on precomputed SPH density fields. The source code is written in standard Fortran 90 and is made publically available under the terms of the latest GNU General Public License ([www.gnu.org/copyleft/gpl.html](http://www.gnu.org/copyleft/gpl.html)).

### 1.1 Requirements

#### 1.1.1 Fortran 90 Compiler

SPHRAY has been designed so as not to require external libraries for basic operation, however the source code must be compiled by the user. A Makefile template is provided. SPHRAY has been successfully tested with the Intel Fortran 90 compiler and with the free g95 compiler ([www.g95.org](http://www.g95.org)) developed by Andy Vaught.

#### 1.1.2 GLUT

In addition to basic operation, SPHRAY can be run with an interactive OpenGL visualization tool. This allows the user to view the ionization or temperature state of the gas and rotate, zoom, or translate the computational volume while the code is running. This is accomplished using the OpenGL Utility Toolkit (GLUT). If GLUT is not already on your system you can download it at [www.opengl.org/resources/libraries/glut/](http://www.opengl.org/resources/libraries/glut/).

#### 1.1.3 f90gl

You will also need f90gl to run the interactive OpenGL visualization tool. f90gl is a public domain implementation of the official Fortran 90 bindings for OpenGL. It can be downloaded at <http://math.nist.gov/f90gl/>.

## 1.2 Obtaining SPHRAY

Source code, example snapshots, and information about the performance of SPHRAY can be found at [www.sphray.org](http://www.sphray.org).

### 1.2.1 Source Code

The SPHRAY source code is expected to evolve fairly rapidly. Stable versions will be periodically posted to the website [www.sphray.org](http://www.sphray.org) in the form of compressed tarballs. The version number will be indicated in the name of the tarball in the standard way (e.g. `sphray-1.0.tar.gz`).

### 1.2.2 Example Snapshots

In order for SPHRAY to run, it requires precomputed SPH density fields. Some example density fields, created to perform scaled down versions of the tests described in the Radiative Transfer Comparison Project of Iliev et. al. 2006, are available in the tarball `iliev-snapshots-small.tar.gz` on the SPHRAY web page [www.sphray.org](http://www.sphray.org).

## 1.3 Unpacking

Once you have acquired the source code tarball you should unpack it using the command `tar xvfz sphray-x.x.tar.gz`. This will create a directory called `Sphray` in your current directory. If you downloaded any example snapshot tarballs, you should also unpack them in your current directory.

# Chapter 2

## Configuration

### 2.1 Makefile Options

To create an executable you must compile the SPHRAY source code. To do this you will need to make a copy of the file `Sphray/trunk/MakefileTemplate`. Name the copy `Makefile` and open the copy in your favorite text editor. Once you have finished editing the Makefile, simply type `make` and your machine will attempt to create a basic usage executable named `sphray`. To compile the OpenGL executable type `make glsphray` instead.

#### 2.1.1 Preprocessor Macros

The first choices you will have to make concern the Makefile preprocessor macro definitions. These provide memory and speed efficiency by determining the minimum amount of data that needs to be tracked for each particle. In addition they instruct the compiler to ignore certain sections of code if they are not needed (e.g. involving Helium for a Hydrogen only run). The different options are detailed in Table 2.1

Table 2.1: Makefile Options

Macro Name	Description
<code>incid</code>	unique integer label (unused in public version)
<code>incvel</code>	velocity (unused in public version)
<code>incT</code>	variable temperature
<code>incmass</code>	variable masses
<code>incHe</code>	Helium ionization fractions
<code>increc</code>	recombination fractions (unused in public version)

**incid**

Uncommenting this macro definition adds an 8 byte integer field to the particle type to uniquely label particles. There is no functionality in the public version that requires this field, but it may be useful to users for their own purposes.

**incvel**

Uncommenting this macro definition adds three 4 byte reals to the particle type for velocity information. It is not used in the public version of SPHRAY.

**incT**

Uncommenting this macro definition adds one 4 byte real to the particle type for temperature information. If this macro is left commented out each particle will be assumed to have the (constant) temperature specified in the config file (see section 2.3).

**incmass**

Uncommenting this macro definition adds one 4 byte real to the particle type for mass information. If this macro is left commented out, each particle will be assumed to have the (constant) mass specified in the config file (see section 2.3).

**incHe**

Uncommenting this macro definition adds two 4 byte reals to the particle type for Helium ionization fraction information. This also tells the compiler to include all code used to calculate the Helium evolution.

**increc**

If the incHe macro is (un)defined, uncommenting this macro definition adds (one) three 4 byte reals to the particle type for recombination fraction information. In the public version of SPHRAY only the "On The Spot" approximation is used (for now) and this information is not needed.

### 2.1.2 Compiler Flags

The next section of the Makefile allows the you to define a system type. Each system consists of a command to call a Fortran 90 compiler and the flags to pass to that compiler. An intel compiler with generic optimization flags is the default choice. If you are using a different compiler or want to pass a different set of flags to the compiler you will have to redefine the `SYSTEM` variable in the Makefile.

The systems I have predefined have descriptive names consisting of 3 parts. The first part describes the computer system the code will run on. There are three options here `psycho`, `toto`, and `generic`. `Psycho` and `Toto` are the names

of two Beowulf clusters at CMU and these definitions take advantage of the -x flags for the intel compiler (type `ifort --help | less` into your command line for more about these). The generic options should work for just about any setup. The second part of the name describes the compiler. The `ift` tag refers to the intel fortran 90 compiler v 10.1 and the `g95` tag refers to the g95 compiler. The last part of the name refers to either optimization flags or debugging flags.

My advice is to use the generic optimized choice for your compiler. If that doesn't work then it would help me improve SPHRAY if you contacted me with a description of the error. If you are designing your own set of flags please note that the flags that cause the generic variable type to be double precision (`-d8` for g95 and `-r8 -i8` for ifort) must always be included.

### 2.1.3 OpenGL Paths

If you would like to run SPHRAY with the OpenGL viewer you will have to specify the paths to the GLUT and f90gl libraries. This is done through the Makefile variables `F90GLDIR` and `GLUTDIR`. The defaults are those for my office computer at CMU. You will need to modify them according to the directory structure of your system and the specific versions of the libraries you have installed.

## 2.2 Config File

Many of the options in SPHRAY are set in a configuration file that is passed to the executable at runtime. This file consists of a series of keywords followed by their values. Several examples of configuration files are provided in the directory `Sphray/data/config`. The meaning of the keywords is described below.

### 2.2.1 Initialization

#### **DoTestScenario** [Logical]

Indicates if the run is taken from one of the predefined tests. This informs SPHRAY to make certain test specific standard outputs.

#### **TestScenario** [Character]

Indicates the specific predefined test to be performed. Must be one of the following (`iliev_test1`, `iliev_test2`, `iliev_test3`, `iliev_test4`)

#### **JustInit** [Logical]

If set to True, instructs SPHRAY to stop after the initial particle and source snapshots are read in, otherwise normal operation. This can be useful for run planning.

**Verbose [Logical]**

Mostly used for debugging. Even if set to false a certain amount of standard output will occur (and could be redirected to a file if convenient).

**JustPhoto [Logical]**

Restricts the evolution of particles to photoionizations and photoheating (in the case of variable temperature runs) ignoring atomic processes such as recombinations and collisional ionizations. This may be useful for coupling SPHray to a hydrodynamic code that accounts for atomic cooling, ionizations due to collisions, and recombinations.

**Expanding [Logical]**

If set to true the positions, velocities, densities, and smoothing lengths in the particle snapshot files are considered to be comoving (peculiar in the case of velocity) and are adjusted to physical coordinates according to the scale factor in the particle header upon read in. This also causes the snapshots to be associated with physical times since the big bang assuming a flat  $\Lambda$ CDM cosmology. If set to false the snapshot data is not rescaled (regardless of what the scale factor in the header is)

**IsoTemp [8 byte real]**

If the Makefile macro `incT` is undefined, all particles will have a fixed temperature equal to `IsoTemp`. If the Makefile macro `incT` is defined then this variable is ignored and temperatures are expected to be in the initial particle snapshot.

**IsoMass [8 byte real]**

If the Makefile macro `incmass` is undefined, all particles will have a mass equal to `IsoMass`. If the Makefile macro `incmass` is defined then this variable is ignored and masses are expected to be in the initial particle snapshot.

**IntSeed [8 byte Integer]**

The integer seed that is passed to the Mesenne Twister random number generator on initialization.

**StaticFieldSimTime [8 byte real]**

For runs involving a single particle snapshot this variable determines the physical time the simulation should be run for. `StaticFieldSimTime` should be specified in code units (default code time unit = 10 Myr). Note that for runs involving multiple snapshots it is important that the time variables in the particle snapshot headers be set correctly as they will be used to determine the physical time a snapshot should be raytraced for.

### 2.2.2 Input

#### **SnapPath** [Character]

Path (relative to the SPHRAY executable) to the directory containing the particle snapshots (e.g. `../..//iliev-small-snapshots`).

#### **SourcePath** [Character]

Path (relative to the SPHRAY executable) to the directory containing the source snapshots (e.g. `../data/sources`).

#### **SpectraFile** [Character]

Path (relative to the SPHRAY executable) to the file containing the user defined spectra (e.g. `../data/spectra/BlkBdyT1e5Bins1e3.txt`).

#### **b2cdFile** [Character]

Path (relative to the SPHRAY executable) to the file containing the impact parameter to column depth conversion table (e.g. `../data/column_depth/ColumnDepthTable.txt`).

#### **AtomicRatesFile** [Character]

Path (relative to the SPHRAY executable) to the file containing the atomic rates table (e.g. `Sphray/data/atomic_rates.txt`).

#### **ParFileBase** [Character]

The particle snapshots in `SnapPath` should all be named according to the following convention: `ParFileBase_SnapNum.FileNum`. `SnapNum` should range from `StartSnapNum` to `EndSnapNum` and `FileNum` should range from 1 to `ParFilesPerSnap`.

#### **SourceFileBase** [Character]

The source snapshots in `SourcePath` should all be named according to the following convention: `SourceFileBase_SnapNum.FileNum`. `SnapNum` should range from `StartSnapNum` to `EndSnapNum` and `FileNum` should range from 1 to `SourceFilesPerSnap`.

#### **StartSnapNum** [8 byte Integer]

The number (3 digits from the format described in the previous entry) of the first snapshot to be read in. SPHRAY will use this information when constructing a list of snapshot files to raytrace. For single snapshot runs `StartSnapNum` should be set equal to `EndSnapNum`

**EndSnapNum [8 byte Integer]**

The number of the last snapshot. SPHRAY will use this information when constructing a list of snapshot files to raytrace. For single snapshot runs **StartSnapNum** should be set equal to **EndSnapNum**

**ParFilesPerSnap [8 byte Integer]**

The number of files each particle snapshot is spread across (must be set to 1 in the public version).

**SourceFilesPerSnap [8 byte Integer]**

The number of files each source snapshot is spread across (must be set to 1 in the public version).

**2.2.3 Raytracing****RayScheme [Character]**

Must be set to one of the following, [**header**, **raynum**]. For **header**, the number of rays to be traced in each snapshot is taken from the source snapshot headers. This method must be used for runs that involve ray tracing multiple snapshots, but also works for single snapshot runs. For **raynum**, the number of rays to be traced is equal to **ForcedRayNumber** (regardless of what is specified in the source snapshot header. This option cannot be specified for runs involving multiple snapshots.

**ForcedRayNumber [8 byte integer]**

See item above. Ignored if **RayScheme** is not **raynum**.

**BndryCond [8 byte integer]**

Determines the boundary conditions for rays and particles. Set **BndryCond** = 0 for vacuum boundaries or **BndryCond** = 1 for periodic boundaries. In the current implementation, rays are terminated after one box length in periodic cases.

**OnTheSpot [Logical]**

Set to true to use the On The Spot approximation. In this approximation recombination photons are assumed to be absorbed within the particle they are produced. This eliminates the need for ray tracing recombination photons. If set to false, rays will be traced when recombination photons are produced, however the user must specify a scheme to modify atomic rates. In the public version only the **OnTheSpot** approximation is implemented and setting this to **False**

has no effect. However, there are subroutines provided to trace recombination rays in the source code if a user wishes to modify it to call them.

**Tfloor [8 byte real]**

The minimum temperature a particle can have in the simulation. Usually set to coincide with the fitting formulas used to approximate the temperature dependence of the atomic rates.

**Tceiling [8 byte real]**

The maximum temperature a particle can have in the simulation. Usually set to coincide with the fitting formulas used to approximate the temperature dependence of the atomic rates.

**RecRayTol [8 byte real]**

A number between 0.0 and 1.0 representing the minimum recombination fraction a species in a particle must have in order for a recombination ray to be traced. Ignored if `OnTheSpot = true`

**RayPhotonTol [8 byte real]**

A photon packet is considered completely absorbed if the number of photons it contains divided by the number of photons originally in it drops below this number. This represents the level of photon conservation and is usually set at  $1.0 \times 10^{-10}$ .

**NeBackGround [8 byte real]**

Represents a constant free electron background from ionized metals. This provides a physical mechanism that ensures the atomic rates will not blow up for completely neutral gas. `NeBackGround` should be specified in free electrons per physical  $\text{cm}^3$ .

**UpdateAllRays [8 byte integer]**

Everytime this number of rays is traced, SPHRAY will loop through all the particles in the simulation and update those that have not been intersected by a ray in the last `UpdateAllRays` assuming a photoionization rate of zero.

## 2.2.4 Gas Composition

**H\_mf [8 byte real]**

A number from 0.0 to 1.0 that represents the Hydrogen mass fraction of the particles in the simulation. `H_mf` and `He_mf` need not add to 1.0 but the sum must be less than or equal to 1.0. This is useful if the masses read in from the

particle snapshot(s) are only partly due to baryons. For example, if the density field (and particle masses) come from a dark matter only simulation possible values would be, `H_mf`= $\Omega_b/\Omega_m$  and `He_mf`=0.0.

#### **He\_mf [8 byte real]**

A number from 0.0 to 1.0 that represents the Helium mass fraction of the particles in the simulation. The sum of `H_mf` and `He_mf` must be less than or equal to 1.0 (see item above).

### **2.2.5 Output**

#### **OutputDir [Character]**

Path (relative to the SPHRAY executable) to the directory where the output snapshots should be written (e.g. `./`).

#### **OutputFileBase [Character]**

The output snapshots in `OutputDir` will all be named according to the following convention: `OutputFileBase_OutNum.FileNum`.

#### **OutputType [Character]**

Set to either `standard` or `forced`. For `OutputType = standard`, outputs are made when  $t = n \times dt$  where  $n$  runs from 0 to `NumStdOuts` and  $dt = \text{SimTime} / \text{NumStdOuts}$ . For `OutputType = forced`, outputs are made at times specified in the file `ForcedOutFile`

#### **NumStdOuts [8 byte integer]**

If `OutputType = standard`, SPHRAY will produce `NumStdOuts`+1 where the extra output will be labeled as output number zero and reflect the initial conditions of the run. This is done because the initial conditions might be different from the particle snapshot actually read in for certain configuration choices (`IsoMass` for example). Ignored otherwise.

#### **IonFracOutRays [8 byte integer]**

This sets the number of smaller outputs. Every `IonFracOutRays` rays traced, a small standard screen output will be made. In addition, the code time, number, volume, and mass weighted global ionization fraction will be output to the file `ionfrac.log` in the directory `OutputDir`.

#### **ForcedOutFile [Character]**

This file specifies the output times if `OutputType = forced` (see `OutputType`). Ignored otherwise.

**ForcedUnits [Character]**

The time units for the entries in `ForcedOutFile`. Must be set to `codetime` in the public version.

**2.2.6 SPH****PartPerCell [8 byte integer]**

The minimum number of particles in a cell to stop oct-tree refinement. This determines the coarseness of the oct-tree and can be useful when memory consumption is an issue. A value that has proved reliable is 12.

**NsphNnb [8 byte integer]**

The number of neighbors used to determine the smoothing lengths of the particles in the simulation. It is important that this number be specified correctly as it is used to recalculate the density of the particles when a particle snapshot is read in for runs in which the `incmass` Makefile macro is undefined. This is necessary because the particle masses can be changed through the configuration variable `IsoMass`

$$\rho = \frac{\text{NsphNnb} \times \text{IsoMass}}{\frac{4}{3}\pi h^3} \quad (2.1)$$

Here the smoothing length,  $h$ , is the distance to the `NsphNnb` nearest neighbor. Note that this definition of the smoothing length differs from some others in the literature which define the distance to the `NsphNnb` nearest neighbor to be  $2h$ .



# Chapter 3

## Usage

To successfully run SPHRAY, there are several prerequisites. They are,

- SPHRAY executable
- configuration file
- impact parameter to column depth file
- spectra file
- atomic rates file
- particle snapshot file(s)
- source snapshot file(s)

The first two are the easiest to come by. If the Makefile template provided is not producing an executable for you, then please contact me with the error and I will attempt to update the template. The configuration file is described in chapter 2 and there are several examples in the directory `/trunk/data/config`.

The next three entries are provided in the SPHRAY distribution, but I list them here so that a user who wants more options can modify them. The first is the impact parameter to column depth file which describes the smoothing kernel. It is a look up table that takes an impact parameter (in units of a particle's smoothing length) into the line integral through the smoothing kernel with  $h = 1$ . The default smoothing kernel in SPHRAY is that detailed in Monaghan and Lattanzio 1985, specifically,

$$W(r, h) = \frac{8}{\pi h^3} \begin{cases} 1 - 6(\frac{r}{h})^2 + 6(\frac{r}{h})^3 & 0 \leq r \leq \frac{h}{2} \\ 2(1 - \frac{r}{h})^3 & \frac{h}{2} < r \leq h \\ 0 & r > h \end{cases} \quad (3.1)$$

This is used to quickly convert impact parameters to column depths. It is good practice to use the same smoothing kernel as was used in the SPH code that produced the density fields being raytraced.

The next item describes the possible spectra of the sources of ionizing radiation. It is a collection of look up tables that take a random number between 0 and 1 into an energy in Rydbergs. Two simple black body spectra files are provided in the SPHRAY distribution. It is not necessary to specify a look up table for monochromatic spectra. Each entry in the source snapshot file(s) includes a real number to specify the spectral type of the source. If the number is positive, it is converted to an integer and determines which user defined spectra should be used. If the number is negative, it produces a monochromatic source emitting photons whose energy is the absolute value of the spectral type in Rydbergs (for example a spectral type of -2.4 produces a monochromatic source that emits 32.64 eV photons, a spectral type of 3 produces a source that has a spectrum defined by the third look up table in the spectra file.

The atomic rates file provides look up tables that take a temperature in Kelvin to an atomic rate coefficient. The file provided in the SPHRAY distribution provides rates for Hydrogen and Helium which are taken from fits discussed in the SPHRAY code paper (a copy of which is in the directory `trunk/docs`). The fits are from  $10^0 - 10^9$  K, as some of these fits disagree with other published atomic rates a user may wish to use their own rates file.

The final two items in the list above describe the density field and the sources of ionizing radiation. The particle snapshot file(s) must be in the SPHRAY format described in chapter ???. The first (and possibly only) snapshot describes the initial conditions of the particles including temperature and ionization state. The subsequent snapshot files are only used to update the positions, velocities, densities, and smoothing lengths of the particles. For each particle snapshot file, the user must also provide a source snapshot file that describes the positions, luminosities, spectral types and emission profiles of the sources. Examples of source files are provided in the directory `trunk/data/sources`. Examples of (the usually larger) particle snapshot files are provided as a separate .tar file on the SPHRAY website [www.sphray.org](http://www.sphray.org).

Once all of these files are in place, SPHRAY can be started by passing the configuration file to the executable as a command line argument. The following example would begin the second verification test (a Hydrogen only, variable temperature, Stromgren sphere created by a source with a 100,000 K blackbody spectrum).

```
sphray ../data/config/iliev_small_tests/illiev_test2_N64_R6.config
```

The config file need not follow any special naming conventions, but the one used here is that N64 indicates a glass density field with  $64^3$  particles and R6 indicates that  $1.0 \times 10^6$  rays will be traced.

## Chapter 4

# Data Structures

The data structures used in SPHRAY are described in this chapter. A user who would simply like to use the public version of SPHRAY should be familiar with those data structures described in the first section. Users who would like to modify SPHRAY for their own purposes will find the other sections useful.

### 4.1 Basic Data Structures

A complete description of the density field to be raytraced and the sources producing ionizing radiation are stored in an object called a particle system. The particle system is a derived type whose structure is outlined in Table 4.1. Those entries representing arrays show the size of the arrays and how they are indexed in the `Name` field.

Table 4.1: Particle system data structure

Name	Description	Type
<code>npar</code>	number of particles in the system	8 byte int
<code>nsrc</code>	number of sources in the system	8 byte int
<code>box</code>	describes the computational volume	box type
<code>par(1:npar)</code>	all particles	particle type
<code>src(1:nsrc)</code>	all sources	source type

A particle system contains three derived types. The `box` derived type describes the extent and boundary conditions of the computational volume as outlined in Table 4.2. The first, second, and third elements of the array `top` refer to the upper x, y, and, z coordinates of the computational volume. A similar pattern is used in the arrays `bot`, `tbound`, and `bbound`. The boundary conditions (which can be specified on each face, but in practice are usually all the same) are set to either -1 (reflecting), 0 (vacuum), or 1 (periodic).

The source derived type describes the sources of ionizing radiation as outlined in Table 4.3. The spectral type has a dual function. If it is set to a negative

Table 4.2: Box data structure

Name	Description	Type
top(1:3)	upper coords. of the computational volume	4 byte reals
bot(1:3)	lower coords. of the computational volume	4 byte reals
tbound(1:3)	upper boundary conditions	4 byte ints
bbound(1:3)	lower boundary conditions	4 byte ints

number, then the source has a monochromatic spectrum at that frequency in Rydbergs. For example a spectral type of  $-2.0$  would produce photons of 27.2 eV. If the spectral type is positive it is converted to an integer which references a user defined lookup table to convert a random number to a frequency. Lookup tables for several blackbody spectra are provided with SPHRAY. The emission profile references another lookup table to convert a random number to a direction. SPHRAY is initially configured so that an emission profile equal to 0 produces isotropic emission and an emission profile equal to -1 turns the lower x face into a source. The `Lcdf` and `lastemit` entries are used by SPHRAY to randomly select sources based on their luminosity and calculate the number of photons to place in a photon packet produced by the source. These entries do not need to be set by the user.

Table 4.3: Source data structure

Name	Description	Type
pos(1:3)	position	4 byte reals
L	luminosity	4 byte real
SpcType	spectral type	4 byte real
EmisPrf	emission profile	4 byte int
Lcdf	luminosity cumulative distribution function	4 byte real
lastemit	id of last ray emitted	8 byte int

The particle derived type is special in that the components that make it up are determined by pre-processor macros set in the Makefile. Those fields marked `always included` are necessary and are part of the particle type even when no Makefile pre-processor macros are defined. The other fields are necessary only for certain types of simulations and the necessary Makefile macro for each field is described in Table 4.4. The first column indicates the data block in which the variable will appear in snapshots produced by SPHRAY.

## 4.2 Advanced Data Structures

Table 4.4: Particle data structure

#	Name	Description	Makefile Macro	Type
1	id	unique integer identification	incid	8 byte int
2	xpos	x position	always included	4 byte real
3	ypos	y position	always included	4 byte real
4	zpos	z position	always included	4 byte real
5	xvel	x velocity	incvel	4 byte real
6	yvel	y velocity	incvel	4 byte real
7	zvel	z velocity	incvel	4 byte real
8	hsml	smoothing length	always included	4 byte real
9	rho	density	always included	4 byte real
10	mass	mass	incmass	4 byte real
11	temperature	temperature	incT	4 byte real
12	xHII	ionized fraction of HII	always included	4 byte real
13	xHeII	ionized fraction of HeII	incHe	4 byte real
14	xHeIII	ionized fraction of HeIII	incHe	4 byte real
15	xHIIrc	recombination fraction of HII	increc	4 byte real
16	xHeIIrc	recombination fraction of HeII	increc and incHe	4 byte real
17	xHeIIIrc	recombination fraction of HeIII	increc and incHe	4 byte real
18	lasthit	ID of last ray to intersect this particle	always included	8 byte int