The ggland command-line simulation wrapper

Long write-up — documentation and manual

© Håkan T. Johansson, 2013
Department of Fundamental Physics
Chalmers University of Technology

Abstract

28th April 2015

Keywords:
Nuclear Physics, Simulation, Quick-n-Dirty

Contents

Chapter 1

Introduction

GGLAND[1] is a "wrapper" program around the GEANT3 [1] and GEANT4 [2] physics simulation libraries. Its task is to make it easy to perform the simulations necessary for physics analysis of the LAND/R^{3}B experiments, as well as other smaller nuclear physics experiments. It provides:

- ways to declare simple detectors, with easily adjustable sizes.
- a versatile "particle gun" to generate particles for the simulation.
- collection, pre-processing and reduction of the plentiful output of the simulation for use in analysis.

Operation is command-line based, suitable for batch processing.

1.1 Quick start

Seeing is believing!

1. Obtain the sources, using either of two options:
   - `tar`-ball (todo: to be fixed).
   - Check them out from CVS at GSI (only with account, locally):

     ```
     cvs -R -d /u/johansso/CVS co land02
     ```

     or remotely (with access to the `land` account; may first need to do `export CVS_RSH=ssh`):

     ```
     cvs -d :ext:land@lx-pool.gsi.de:/u/johansso/CVS co land02
     ```

2. GGLAND is in a subdirectory (to be fixed?).
   `cd land02/scripts/ggland/`

[1]In this text, names of programs is typeset using SMALL-CAPS, and computer input/output with `fixed-with` font.

3. Build the simulation wrappers:
   ```
   make
   ```
   or just one of them (here with GEANT4)[2]:
   ```
   make land_geant4
   ```
   Building requires that the GEANT3/4 libraries can be found. I.e. that `cernlib` and/or `geant4-config` are in the `$PATH`. At GSI[3]:

   - `. cernlogin pro`
   - `export PATH=$PATH:/u/johansso/geant4/geant4.9.6.p01/bin/`

   The system knows how to use Debian packages (e.g. `g4make` for GEANT4) out-of-the-box.

4. Get ready to run — make sure data-files can be found:
   ```
   . geant4.sh
   ```
   Or with the Debian packages:
   ```
   g4run bash
   ```

5. Run (10 events of nothing):
   ```
   ./land_geant4
   ```

6. To be able to write ROOT files (`rootlogin` at GSI):
   ```
   . rootlogin pro
   make -C ../../util/hbook/ root_writer
   ```

7. Something more interesting: a 1 MeV isotropic γ source besides a 5x5x5 cm^3 volume of Ge, data to `Co60inGe.root`:
   ```
   ./land_geant4 --gun=gamma,E=1MeV,isotropic --events=10000 \
   --test=d=5cm,z0=2.5cm,type=Ge,tree=1 --tree=Co60inGe.root
   ```

8. Open the produced data for analysis:
   ```
   root -l Co60inGe.root
   ```
   and draw!
   ```
   h102->Draw("TSTe")
   ```

9. Further options are listed by:
   ```
   land_geant4 --help
   ```
   Or for sub-options:
   ```
   land_geant4 --gun=help
   ```

[2]Copying text from pdfs viewed with XPDF seems to loose the underscores. Take care!
[3]There seems to be no centrally compiled version of GEANT4, thus the private build. Building the libraries is not very difficult, though.

10. Set up the field maps (by symlink) (if ALADiN is to be used):
    ```
    ln -s /u/sfienti/S254/momentum/tt/newmap/
    ```

1.1.1 Data for Ralf's tracker

Hits from simulated events can be used to reconstruct the generated particles using Ralf's tracker [5].

1. Obtain the TRACKER sources (todo: currently requires some patches, therefore not the official source repository):
   ```
   git clone /u/rplag/tracker
   cd tracker
   ```
2. Produce a suitable data-file (200 events):
   ```
   cd sim001
   . geant4.sh
   GGLANDDIR=path-to-land02/scripts/ggland/ ./dosim.sh
   ```
   The `dosim.sh` script contains the `land_geant4` arguments necessary to place the detectors, arrange a magnetic field, setup a particle gun and write the output to a file (`sim1.root`).
3. Build the tracker (using `sim001.hh`):
   ```
   make HAVE_QT=0 -C .. sim001/tracker_sim001
   ```
4. Set up the field maps:
   ```
   ln -s /u/sfienti/S254/momentum/tt/newmap/
   ```
5. And do the tracking:
   ```
   ./tracker_sim001 --config=sim001.txt sim1.root \
   --output=tracked1.root --track-mixed
   ```
6. Open the produced data for analysis:
   ```
   root -l tracked1.root
   ```
   and draw!
   ```
   h509->Draw("fra_A")
   ```
7. Correlations with the per-event gun parameters can be made, by opening the simulated tree as a friend[4] and building an event-number index[5]:
   ```
   h509->AddFriend("h102","sim1.root")->GetTree()->BuildIndex("Evnt")
   h509->Draw("fra_dx:gunpx")
   ```

[4] Reversing roles and adding the tracked tree as a friend to the simulated works too, but causes a warning.

[5] Needed as the tracker only writes events that were tracked.

1.2 Data flow

The overall data-flow of both analysis (calibration and reconstruction) and simulation is shown in Figure 1.1. Analysis flows downwards in the picture, from raw data produced by the data acquisition to finally the identified particles and momenta. Simulation (the left part of the figure) works in the opposite direction. Starting with particles (in the gun), they are transported in the setup by GEANT3/4 which as output gives discretised energy losses in time and space in the volumes of the setup. These discretised energy losses do however not directly match any level in the reconstruction flow.

The work of a digitiser is to combine the output from the physics simulation such that it matches and can be inserted into the normal reconstruction flow — to allow analysis with the *very same* tools and routines as are used for real data. To not try to mimic reconstruction, this additional flow should be upwards in the picture. As the discretised energy losses are already separated by volumes in the geometry, the first possible injection point is (in LAND02-speak) the `SYNC` level. The combination of discretised energy losses is performed by the `ggdigi` program.

For many quick-n-dirty simulation needs, it is however cumbersome to have to go all the way in reconstruction to reach the equivalent of `HIT` level again. When[6] it is acceptable to make some approximations of the decisions made in the reconstruction, a generic "clusteriser" can be used instead, effectively doing the work of the digitiser and reconstruction steps together. (See Section 5.2.) This does break the idea to not mimic parts of the reconstruction.

1.3 Coordinate system

The simulations use a right-handed coordinate system, with z as the default forward direction, y upwards and thus x pointing to the left.

[6]Or, due to the `SSYNC`→`SYNC` bridge not having been built yet...

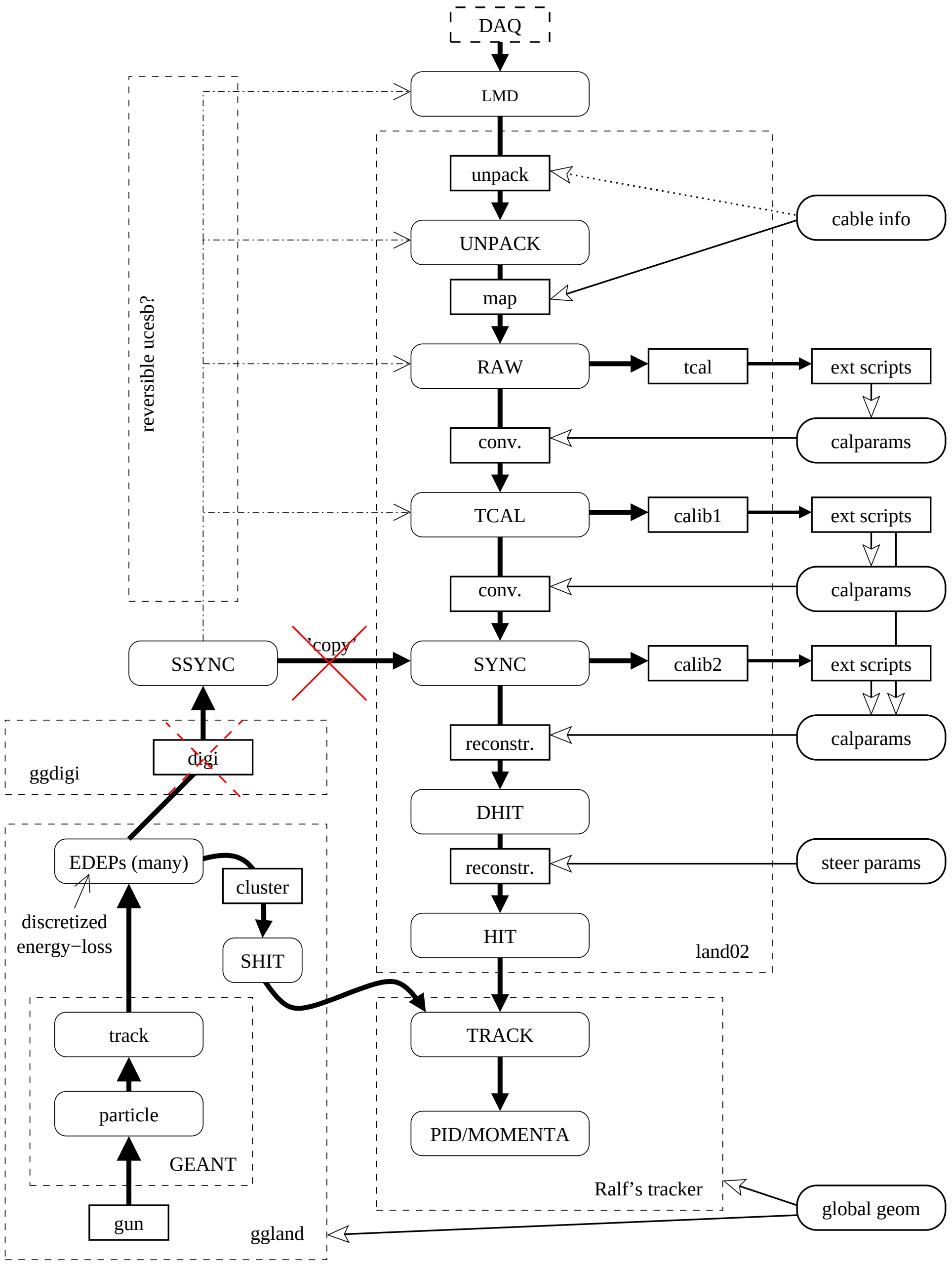

Figure 1.1: Data flow in analysis and simulation of LAND/R^{3}B experiments. (UNPACK level does not exist, is just suggested.) The SSYNC / 'copy' stage from `ggdigi` does not exist. (SHIT is to be pronounced [s-hit], as in simulated hit.)

Chapter 2

Command line options

The simulation wrapper program is driven by command line options, controlling the input (particle gun), processing and output. The following terse descriptions, explained more in detail in this chapter, are from the program itself:

```
--events=N         Number of events.
--seed=S1,S2       Random number seed.
--np[=N]           Fork simulation processes.
--quiet            Do not report progress statistics.
--verbose[=N]      Increase verbosity.
--colour=yes|no    Force colour and markup on or off.
--volume-stats     Tracking statistics for volumes.
--cmd              Interactive session.
--cmd=FILE         Initial commands.
--rangecut=Xcm     Default range cut (world).
--phys-list=NAME   Choose physics list (def: QGSP_BERT), or 'list' to show.
--geom=FILE        Read geometry from file (through cpp), or stdin as -.
-Dmacro[=defn]     Pass macro to C preprocessing of geometry file.
--list-materials   List available materials.
--list-particles[=name|pdg|pid]  List available particles (sort-order).
--hits=[OPT,]FILE|help           Output event (hit/e-deposit) data.
--pa:DET=ATTR|help        Attributes for picture of detector DET.
--picture=ATTR,FILE|help  Generate picture of (selected) geometry.
--fieldbox=SPEC|help      Homogenous magnetic field.
--tree=FILE        Generate ROOT tree for selected test volumes.
--gun=SPEC|help    Particle generator.
--gun-file=FILE    Generate events with particles from file.
--DET=SPEC|help    Specifications for detector DET, one of:
                   box_cave, cave, csi, dtf, gfi, land, mfi, ntf, pdc, pdcy,
                   pos, psp, rolu, sciland, sst, test, tfw, world, xb
--DET-ABC=SPEC     Specifications for detector DET.  Not placed.
--DET-N=SPEC       Specifications for numbered detector DET.
                   With other SPECifications: --DET-N=spec=DET-ABC[,rot,xyz0]
                   Inside root volume of another: --DET=inside=OTHER,...
```

2.1 Run control

By default, 10 events are generated using a random seed based on the current time.

`--events=n` Generate and process `n` events before terminating.

`--quiet` Do not print progress reports.

`--seed=s1[,s2]` Set the random seed for the first event to `s1,s2`. For each subsequent event, `s1` is incremented by one.

`--np[=n]` Fork and run `n` simulation processes in parallel. By default, one simulation thread is started per CPU (core) available to the parent process. The output data is stable and independent of the actual production order; it is queued and recorded in order per event.

`--colour=yes|no` Force colour for errors and warnings. When piping to `less`, use `... | less -R`.

2.2 Simulation control

`--rangecut=Xcm` (GEANT4 only.) Set the range cut-off value for the world medium. See Section 6.2.

`--phys-list=NAME` (GEANT4 only.) Select a (predefined) physics list to use. Default is `QGSP_BERT`.

2.3 Placing detectors

Detectors (or other volumes) are placed in the setup geometry by invoking their constructors, giving the type, instance number (when needed) and any modifications to the default parameters. See Section 4.1 for further details and examples.

`--DET=ROTLOC,SPEC` Create an object (detector) of type `DET`, with the given parameters. Placement options:

> **`x0=Xcm, y0=Ycm, z0=Zcm`** Displace the origin of the detector in the world volume.
>
> The order of rotations and displacements is important, as they do not commute. Normally, a detector would be rotated before displacement. Multiple operations are allowed — useful for placement along the fragment or proton arms behind ALADiN.
>
> **`rotx|y|z=ANGLE`** Rotate the detector around the `x`, `y`, or `z` axis.

General options:

`spec=MODEL` Use the actual object of another `MODEL` for this item. The placement is not inherited.

`inside=MOTHER` Place this item inside (the root volume of) another `MOTHER` detector.

`help` Show the `SPEC`ification attributes available for this `DET`ector type, with default values.

`--DET-N=ROTLOC,SPEC` Create an object with instance number `N`.

`--DET-ABC=SPEC` Specify non-default parameters for an object of which possibly many are wanted. The identifier `ABC` must not start with a digit. The object is not placed in the world; `ROTLOC` is not allowed. Place using a second creation-statement (as above) with `spec=ABC,LOCROT`.

`--list-materials` List available materials.

2.3.1 Adjusting digitisation output

When recording data using the clusteriser (see Section 5.2), the output and algorithm can be adjusted per detector using `tree-` options in the `--DET=` arguments:

`tree-sort-by-t` Sort the output clusters by time instead of decreasing energy.

`tree-num-clusters=N` Include at most `N` clusters instead of the default 10.

`tree-keep-cluster-prob=[I=]P` Each cluster is discarded with probability 1-`P` in submodule `I` (one-based). Without a submodule number, it applies to all submodules.

`tree-keep-event-prob=[I=]P` All clusters within submodule `I` (one-based) are discarded with probability 1-`P`. Similar to the previous option, except that all clusters within the same submodule and event are kept or discarded together. Without a submodule number, the probability applies to all submodules.

2.3.2 Magnetic fields

The magnetic field options use the same placement arguments as the detectors (`LOCROT`, i.e. `x0`, `y0`, `z0`, and `rotx|y|z`). The magnetic fields are not constrained to any particular volumes, but are instead defined in the entire galaxy[1], erhm — world.

[1] May the force be with you?

`--fieldbox=ROTLOC,DIM` Places a box with a uniform file in the world volume.

`--aladin=ROTLOC,I=IA` Places a field according to measured ALADiN field maps for the given current in the world volume. The field is given by the current `I`, positive for Cave B (where the field-maps were measured, bending to the left) and negative for Cave C.

2.4 Particle guns

Particles are produced either internally or from sequential input.

`--gun=OPT` One item for the internal particle gun, see Section 3.2 for a description of all options.

`--gun-file=FILE` To read from a pipeline, use `-` for the filename to read from `stdin`.

`--list-particles=SORT` List available particles (as known by the underlying Monte Carlo-engine, GEANT3 or GEANT4). They can be `SORT`ed by `name`, `pdg`[2], or `pid`[3].

2.5 Output to digitisation

Output for further processing or analysis.

`--tree=OPT,FILE` The energy deposits can be combined within each detector segment by an internal digitiser/clusteriser. With this option, data from events that have at least one non-zero hit are recorded. Hits are essentially combined based on distance in space and time; see Section 5.2 for details. `FILE` gives the output ROOT file name.

`nogun` Do not include the parameters of the initial particles.

`onlygun` Only include the parameters of the initial particles.

`gunlist` When only one particle (gun) is used, the gun variables are not produced as variable length (zero-suppressed) arrays. Produce arrays also in this case.

`allevents` Include also events with no energy deposit in any active volume. Implied by `onlygun`.

`nonreduced` Include also variables not indicated by each detector as being measured by that device.

[2]Encoding by the Particle Data Group [6]. Used by GEANT4 (extended for ions).
[3]Natively used by GEANT3.

part Include information about which particle types contributed to the energy deposit of each clustered hit.

land02 Mangle variable names to mimic LAND02 output.

land02track As above, and limit the hit lists of some detectors to not overflow the tracker arrays.

minentries=N Create arrays with space for N entries, instead of the default 10.

See Section 2.3.1 for options to modify the digitised output of individual detectors.

--hits=OPT,FILE Gives all discretised energy deposits in active volumes of the detectors. To use the output in a pipeline, use `-` for the filename to write to `stdout`. `OPT`ions to adjust the output:

vertex In addition to the energy deposits, provide a full tree of the particles generated by the Monte Carlo-simulation. Energy deposits can be associated to individual particles.

origin Only give the initial (gun) particles in the particle tree format.

nohits Do not produce the energy deposit information. Useful with **origin** to only use the built-in particle gun. The output is compatible with the **--gun-file** option.

The program **ggmark** can be used to colourise the otherwise rather dense (and hard-to-read[4]) output. Usage:

```
./ggland ... --hits=[vertex],- | ./ggmark | less -R
```

2.6 Visualisation

--cmd[=SCRIPT] (GEANT4 only.) Start an interactive session with GEANT4. Useful to operate the visualisation drivers. When given a `SCRIPT`, this is executed first.

--pa (GEANT3 only)

--picture (GEANT3 only)

2.7 Examples

[4]Not seeing the forest for the trees. / Just a few numbers too many.

Chapter 3

Particle gun

Particles for each event are generated randomly by the particle gun, based on the random seed of the event. In addition, particles can be provided event-by-event by external input.

3.1 Gun overview

Each command line particle gun item (`--gun`) will be added to the list of guns to consider every event. Dependent items are handled by a "level feeding" mechanism. An item can be top-level, i.e. part of the guns to invoke each event, only subject to each one's independent probability. Alternatively, the item is one possible outcome of a fed level. Each invoked gun may feed a named level, which in turn will invoke (at most) one of the outcome levels, depending on their probabilities. Multiple subsequent particles must thus be 'unflattened' into a tree of multiple single-branches. The only exception is particle generation by the `phasespace` option.

The command for each item is parsed upon startup, producing an entry in an internal structure. For each event, where the item is invoked, a "firing" routine will create the particle, invoking randomisation routines as instructed; see Figure 3.1.

When investigating the effect of different independent sources of e.g. background, analysis is much easier with separate simulation runs for each independent source. When probing the response to random coincidences, the `tdelay` option is useful to ensure that the particles are not generated at exactly the same time. The time range would correspond to the larger of the experiment trigger coincidence time, and shaping/gate times for converters.

3.2 Options

Each bullet of the versatile particle gun can be modified by a number of options. By default, particles are emitted in the z direction (or along the

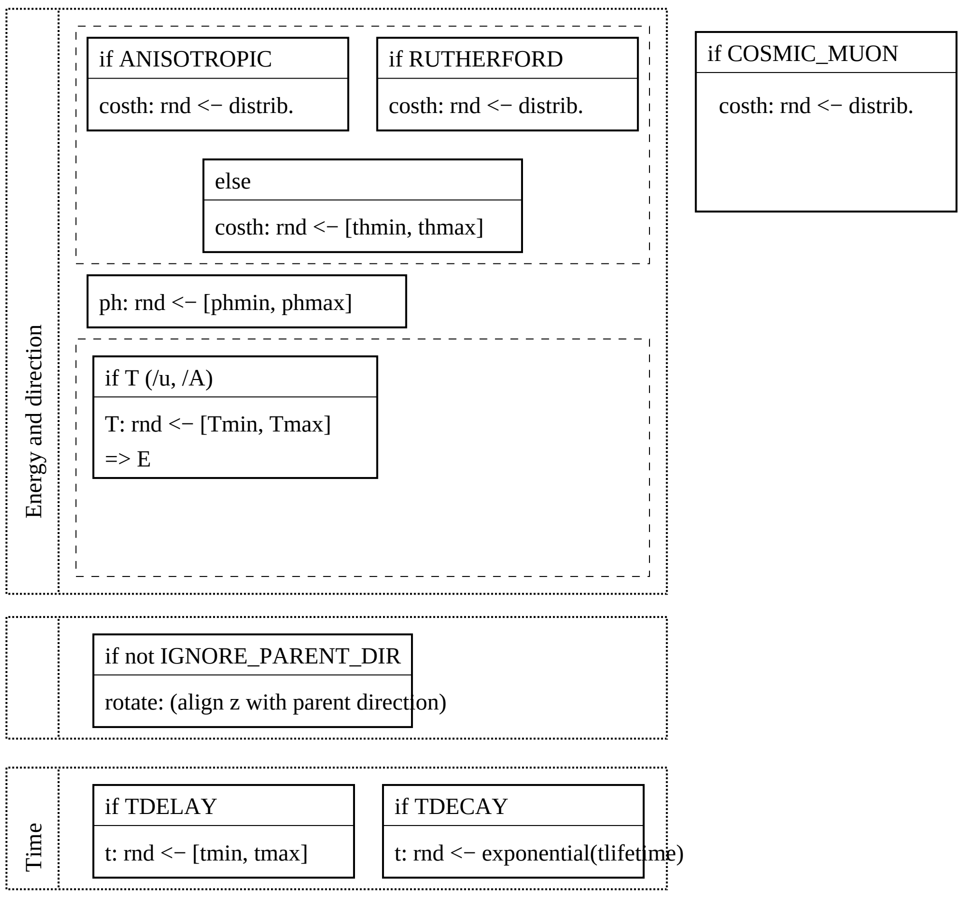

Figure 3.1: Actual firing sequence for each item of the particle gun. (Figure is incomplete / work-in-progress.)

parent particle direction if applicable). At minimum, the type and energy (possibly as momentum, or β) of the particle has to be given. Options as described by `--gun=help`:

```
Parameter:    Default:  Description:

tdelay:tmax   [0 ns]    Delay t.
tdecay        [0 ns]    Delay by exponential decay (mean).  (thalf for t_1/2).
x0, y0, z0    [0 cm]    Origin.
sx, sy, sz    [0 cm]    Source extent (box,ellipsoidals).
sr            [0 cm]    Source extent (disc,sphere,shell),
box/sphere/shell/disc   Shape of sx/sy/sz source extent.
                        Disc perpendicular to nominal direction.
                        (sphere is solid, shell is surface.)
shapenorot              Do not rotate source shape with theta & phi.
                        Shape of sx/sy/sz source extent.
noverttochild           Do not give vertex to subsequent particle (feed).
                        (Child using a source extent forces this.)
```

```
ignoreparentdir          Ignore parent direction.
theta, phi     [0, 0]    Nominal direction, (e.g. 45deg for degrees).
dtheta         [0]       Spread of beam, (e.g. 45deg for degrees).
                         (Also as range [min:max].)  Or as dcostheta.
dphi           [2pi]     Phi section.  (Also as range [min:max].)
isotropic                Shorthand for dtheta=1pi.
anisotropic=a2:a4        Anisotropic distribution (rel. to parent).
backtoback               Generate in opposite direction to parent.
T,Tmin:Tmax    [MeV]     Kinetic energy, fixed or equidistributed min -- max.
                         Also as T/u and T/A, with MeV/u and MeV/A.
p,pmin:pmax    [MeV/c]   Momentum, fixed or equidistributed min -- max.
px, py, pz     [MeV/c]   Momentum, with direction.
E,Emin:Emax    [MeV]     Total energy, fixed or equidistributed min -- max.
                         (Units [kev] or [me] also allowed.
beta,betamin:max         Beta, fixed or equidistributed min -- max.
Brho,Brhomin:max         Brho (magnetic rigidity), fixed or min -- max.
cosmmuonthetaE=lowEcut   Energy and angular distribution for cosmic muons.
                         (Or cut given as concrete shield thickness [cm].)
fromworldedge            Move particle backwards to start from world edge.
feed=name                Activate subsequent particles (name for level).
prob=w                   Probability, (e.g. 45% for percent, multiplication and
                         division also allowed; dt to scale by tdelay range).
from=name:w              Given particle level with 'name' was fed.  Probability
                         w.  Items are mutually exclusive.
boost                    Boost particle by set parent particle boost.
setboost                 Use four-vector as boost source.
phasespace               Multiple particles share the given energy. (theta, phi,
                         dtheta, isotropic applies to *last* particle).
                         Particle can be followed by :setboost :dummy :feed.
particle/isotope         The particle to generate (e.g. gamma,e-,p,n,d,t,Li6).
dummy                    Fire blank - no particle generated.
```

Incompatible or senseless combinations of options are refused by the command line parser. For readability in the following, spaces have been inserted between values and units. This is accepted by the parser, but require the arguments to be enclosed in quotation marks, so is generally not used in practice.

3.2.1 Source location

The source is by default located at the origin, at time zero.

`x0=X cm`, `y0=Y cm`, `z0=Z cm` Move the centre of the source to the location specified.

`tdelay=T ns|TMIN:TMAX ns` Emit the particle at the time (or within the range) specified. When a range is used together with `cosmmuonthetaE`, the probability is adjusted so that an realistic amount of muons are generated in the given time interval.

`tdecay=TLIFE ns` Delay the emission of the particle according to an exponential decay with the given lifetime.

`thalf=T1/2 ns` Same as above, but using the half-life.

3.2.2 Source shape

The source is by default point-like.

`box`, `sphere`, `shell`, `cyl` Make the source extended in space. While a `sphere` is a volume, the `shell` emits only from the surface. The `cyl`inder is by default along the z axis, and with `sz` it becomes a disc.

`sx=X cm`, `sy=Y cm`, `sz=Z cm`, `sr=R cm` Give the size of the source volume. Cartesian directions are used for the `box`, or to make a round shape elliptical. The cartesian directions give half-widths, akin to the `h`-options for the test volume. (todo: elliptical shapes have constant emission per volume/area?)

`surround` Distribute the particle origin on the surface of a shell. The inward direction distribution ($\cos^2\theta_{\text{inward}}$) ensures that all points within the shell see the same amount of particles, and isotropically, if there is no material.

`shapenorot` The source is normally rotated to be aligned with the nominal gun direction (`theta, phi`) (in turn aligned with any parent direction). This option prevents that.

`noverttochild` When a subsequent source is point-line, the particle is for administrative purposes with `geant3/4` normally associated with (being emitted from) the same vertex as the parent particle. This option prevents that.

3.2.3 Emittance

The particles are by default emitted in the local z-direction. If the gun has a parent, the local z-direction is event-by-event equal to the actual parent emission. This works well with the particle having an anisotropic or conical emission envelope (`dtheta`), but breaks down if a range or fixed `dphi` is specified (as the choice of new x- and y-directions is ambiguous).

`ignoreparentdir` Do not rotate the emission direction to align the local z direction with the actual emission direction of the parent particle.

`theta=THETA`, `phi=PHI` Centre emission around the direction given by the θ and φ angles. Units for angles are by default radians, unless given as `ANGLE deg` or `ANGLE pi`.

`dtheta=DTHETA|THMIN:THMAX`, `dcostheta=DCT|CTMIN:CTMAX` Give a range in θ in which to emit the particles. This θ is relative to the axis given by `theta`, `phi`. With one value, the range is opening from the forward direction.

`dphi=DPHI|PHIMIN:PHIMAX` Give a range in φ in which to emit the particles. This is useful when simulating reaction products for hitting detectors with limited coverage. Is 2π by default.

`isotropic` Alias for `dtheta=1pi`; isotropic emission.

`anisotropic=a2:a4` Anisotropic emission, relative to the parent direction, using the relative weight function

$$w(\theta) = 1 + a_2 \cos^2\theta + a_4 \cos^4\theta. \tag{3.1}$$

`backtoback` Emission in the direction opposite to the parent particle. Useful for the second photon of an annihilation pair.

3.2.4 Energy

The energy for the particle(s) must be given explicitly using one of the following directives. When a range is given, the value is chosen randomly with a uniform distribution. Instead of a uniform range, a gaussian distribution can be obtained by `mu=MU:sigma=SIGMA` instead of `MIN:MAX`.

`T=T MeV|TMIN:TMAX MeV` Give the available kinetic energy.

`T/u=T MeV/u|TMIN:TMAX MeV/u`, `T/A=T MeV/A|TMIN:TMAX MeV/A` As above, but per mass or nucleon.

`p=P MeV/c|PMIN:PMAX MeV/c` Give the (absolute value of the) momentum of the particle.

`px=P MeV/c`, `py=P MeV/c`, `pz=P MeV/c` Give the momentum vector of the particle. Incompatible with any other option specifying the emittance or direction.

`E=E MeV|EMIN:EMAX MeV` Give the total available energy (including rest mass).

`beta=BETA|BETAMIN:BETAMAX` Give the fractional speed of light of the particle, $\beta = \frac{v}{c}$. The particle must not be massless.

`Brho=BRHO|BRHOMIN:BRHOMAX` Give the magnetic rigidity of the particle, $B\rho$. The particle must neither be massless nor uncharged.

3.2.5 Level feeding

Each gun item is either independently generated each event, subject to an independent probability, or generated as a particle from a named “level”. The latter is useful for implementing realistic γ sources.

`prob=W` The particle is a top-level gun item, generated with probability `W`.

`feed=NAME` The particle feeds a level called `NAME`, i.e. one of the particles from that level will be issued. (Possibly none, if their summed probability is less than 1.)

`from=NAME[:W]` Set the particle up to be one of the candidates for going from level `NAME`, with probability `W`. Multiplications and divisions are allowed, like `0.0075/99.88`, as well as percentages, like `90.326%`. A specifier of `tot%`, as in `85.1tot%`, gives the total probability for the particle, given all paths from top-level items to this level.

3.2.6 Boosting

Particles can be Lorentz-boosted into the frame of parent particles. Useful for creating reaction products or other assorted debris at beam-velocity.

`setboost` Set the boost (essentially velocity vector) to use with so wishing subsequent (dependent) particles. Child particles most likely want to specify the `ignoreparentdir` option, as they otherwise will align their direction with this one, and then boost using it (again).

`boost` Boost the particle by the velocity vector set by a parent.

`dummy` Do not emit the particle as such; useful when setting a boost (e.g. with an incoming beam particle that should not be simulated).

3.2.7 Particle type

Except when using the special distribution options, one particle type has to be specified. (For e.g. setting a boost, `dummy` can be enough, together with `beta` or `T/u`.)

`PARTICLE` Determines the particle to emit, some examples are shown in Table 3.1. Use the command-line option `--list-particles` for a full list (except ions in GEANT4). Note that GEANT3 only handles a few (mostly stable) ions.

`pdg=PDG` todo! (implement…)

3.2.8 Special distributions

Some more involved but useful correlations or distributions of particles which cannot be described using combinations of the above options are implemented explicitly and are requested thus:

γ	`gamma`	p	`p`
e^-	`e-`	n	`n`
e^+	`e+`	d	`d`
μ^-	`mu-`	α	`alpha`
μ^+	`mu+`	^{12}C	`C12`

Table 3.1: Names of some common particles.

`phasespace` The specified particles (at least two) are emitted according to a random phase-space distribution, using the Raubold-Lynch method [9]. Note that the direction of the last particle listed is according to the gun emittance; use e.g. `isotropic` to randomise also this direction.

Each `PARTICLE` can have one or more attributes appended:

`:setboost` Use this particle to set the boost of subsequent particles.

`:dummy` Do not emit the particle.

`:feed` Use this particle for feeding subsequent particles. (In particular, its direction.) This option also requires the gun to specify the level name using a normal `feed` option.

The phase-space distribution can e.g. be used to easily crate an approximate β-spectrum (ignoring Coulomb and angular momentum effects).

`cosmmuonthetaE=LOWECUT MeV|SHIELD cm` Generate muons with a realistic distribution [7] of energy and zenith-angle (θ_y). The muons are generated with emission points on a disc tangential to a sphere of radius `sr` (which must be given). For each event, the tangential point is randomised according to the zenith-angle. With this method, the distribution is the same for all points within the sphere of radius `sr`. `dtheta` and `dphi` can be used to limit the incoming direction. Unless `mu-` or `mu+` is explicitly specified, the particle type is sampled according to [8] up to 4.15 GeV/c, after which a constant ratio is used (from PDG).

Either a low-energy cut-off (`LOWECUT`) must be used, or a concrete `SHIELD`ing thickness. In the latter case, the distribution is modified accordingly without actually including the concrete in the geometry (then with no reactions or angular straggling).

`fromworldedge` When producing a distribution that is valid in a certain volume (like the cosmic muons) it may still be interesting to make the particles penetrate some outer material first. With this option, the particle direction is unchanged, but the emission point is back-tracked to the world boundary.

`thermal` The momenta are randomised in 3 dimensions, with a Gaussian distributions. With any of `px`, `py`, or `pz` given, the widths (sigma) of the distributions are taken as those. With `p` given, each dimension has $\sigma_p = p/\sqrt(3)$. Given the kinetic energy, the distribution is adjusted such that the average kinetic energy is as given. (With a slight reduction due to relativity being taken into account in the momentum conversion, but not in the distribution.) Note that `T` is used to give kinetic energy, not a temperature!

3.3 Predefined guns

Some common scenarios of particle generation are predefined, and will instantiate a tree of guns using the normal descriptions. They are invoked as `--gun=[OPT,]PREDEF`. The only allowed `OPT`ions are `x0`, `y0`, `z0`, `tdelay`, as well as top-level `prob`. Available `PREDEF`ined guns:

`decay-Co-60` The γ decay following β^--decay of ^{60}Co. The β^--particle is not generated[1]. The anisotropy between the 1.173 MeV and 1.332 MeV γs is implemented.

`decay-Na-22` The γ decay following β^+-decay of ^{22}Na. The β^+-particle is not generated, but the annihilation photons are (except if the randomisation allowance for electron capture struck).

`decay-Cs-137` The γ decay following β^--decay of ^{137}Cs. The β^--particle is not generated.

3.4 Examples

Level-feeding.

Phase-space.

[1]The electron would normally not escape the source container, which thus need not be included in the geometry.

Chapter 4

Geometry definitions

The geometry of the setup is described using the basic shapes available in GEANT3/4. The construction of the detector is described programmatically in the program source, and instantiated in the actual simulation geometry by a command-line option[1]. This Chapter describes both these aspects, beginning with the second.

4.1 Using a detector

The creation of a detector is invoked by the command-line option `--det=...` as described in Section 2.3, with `det` being the name of the device. Each detector is described by a number of variables that control the actual dimensions and materials of it, see Section 4.3.1. The default values would usually correspond to the real device, but can be modified on the command line, as in `--test=d=10cm,type=plastic`. This is in addition to the placement options `x0`, `y0`, `z0`, `rotx|y|z`, that determine the location and orientation of the device in the simulation world (defaulting to unrotated at the origin). To get a list of the variables with default values for a detector, ask for: `--det=help`. Flags can be prepended with ~ for not. Several detectors with the same constructor can be created (possibly with different modifications), but need then to use instance numbers for identification, like `--pos-1=z0=-100cm`.

An item can use the same object as another item (but at a different location), by using the `spec=MODEL` option. Note that any subvolumes (including other items placed inside the model object will appear also in the duplicate. An item can also be placed inside (the root volume) of another detector, using `inside=MOTHER`. It is the responsibility of the user to ensure that any child volumes do no extend outside the mother volume, and that no child volumes overlap.

[1]Shell scripts are powerful tools to keep repeatedly used commands and options in.

Energy deposits in the volumes registered as active by the creation function are automatically recorded by GGLAND, and propagated to any requested output files (cf. Section 2.5).

4.1.1 Test detector

A simple but useful "detector" consisting of a single shape (usually box) is the test detector. It is fully customisable from the command line. In its default configuration is is a cube with full side length 1 cm. The only required parameter is the material type. Available modifications:

`type=MAT` Specify the `MAT`erial of the volume. For a list of available materials, use `--list-materials`. Use `blackhole` to make the volume absorb (discard) all incoming particles.

`d=Dcm` Specify the full-width dimension of the volume. (Individually as `dx`, `dy`, `dz`.)

`h=Hcm` Alternative to the above, but giving the half-width dimension of the volume. (Individually as `hx`, `hy`, `hz`.)

`dm=RHOg/cm2` Specify the dimension (thickness) of the volume in terms of the surface density of the chosen material. (Also as `dmx`, `dmy`, `dmz`.)

`r=Rcm` Specify the radius of the volume. Makes it a sphere instead of box, or a tube (cylinder) if `dz` (or similar) is given.

`ir=IRcm` Give the inner radius of a sphere or tube, making it hollow.

`theta1=TH1, theta2=TH2, phi1=PH1, phi2=PH2` Limit the sphere in θ and/or φ (also tube).

`rangecut=Rcm` Modify the range cut tracking parameter in the volume. See Section 6.2. Mostly useful for testing.

`out_col=COL` Dump the recorded total energy deposit per event on `stdout`, as one line per event. The `COL`umn gives which column to accumulate to (for dumping from several test volumes). Useful post-processing pipeline:

```
./land_geantX ...  | \
grep TSTDMP | sed -e 's/.*://' -e 's/\#.*//' > OUTFILE
```

`tree=1` Include the hit clusters from this volume into an output `--tree` file. The data is not included by default, as the test volume also can be useful to create passive objects in the setup geometry. (todo: make the argument change the clusteriser combine-distance. for now: dummy) `enter` Include a list of particles entering into this volume in the output `--tree` file. The format is similar to that of the list of gun-produced particles, cf. Table 5.2.

4.2 Creating a detector

Before constructing a detector, it is strongly suggested to study the available shapes of both GEANT3 and GEANT4, as well as how the simulation libraries optimise navigation in the geometry.

Most often, the same organisation works for both libraries, especially when the geometry is simple and made of (multiple layers of) boxes. The LAND/R^{3}B setup fits this bill very well. In some cases however, e.g. the Crystal Ball, variations on the construction strategy can be beneficial. It can even be that an efficient layout in GEANT3 is not even allowed in GEANT4, and vice versa. The differences mainly stem from the different handling of unions and intersections of basic shapes.

...

Detectors are created from the basic shapes by direct `C++` code, invoking routines that wrap the relevant calls to GEANT3 and GEANT4. The idea is that these creation routines work with variables and avoid using hard-coded values[2] like the plague.

4.3 Code organisation

The geometry of a detector is specified in one file: `geom_det.cc`, `det` being the (shorthand) name of the detector. This file contains three important elements:

- A structure `struct spec_DET_t` declaring the variables, with default values, that affect the dimensions and other aspects of the detector.
- An `#include "auto_gen/spec_info_det.hh"` to include a file generated using the contents of the above structure. The file creation and updating is automatically handled by the GGLAND `Makefile`.
- A normal function `make_DET(...)` to create the detector using the the basic shapes, and set up volumes for being active elements that should record energy deposits.

4.3.1 Structure of controlling variables

Each variable has a name, default value, unit and description.

[2]Even if one does not intend to ever change a value, using variables improves readability.

4.3.2 Auto-include

4.3.3 Creation function

Half width vs. full width

Chapter 5

Digitisation

The discretised Monte Carlo-generated energy deposits need to be combined within each segment of the detectors before data comparable to measured experimental quantities is obtained.

5.1 General digitiser

[unfinished]
program: `ggdigi`

5.2 Clusteriser

As an alternative to the full-fledged digitiser described above that takes data upwards in Figure 1.1, a simplified one-clusteriser-fits-all internal digitiser can be employed. The strategy by design combines digitisation and a reconstruction mimic. It is invoked with the command line option `-tree=...`, see Section 2.5 for a detailed description of the arguments.

The hits in each segment of active volumes are collected separately. The energy deposits are recorded as line segments from each simulated particle. After each event, the clusteriser combines all line segments within the same volume segment into clusters. Two line segments are combined if they are anywhere[1] closer than

$$d^2_{\text{space}} + d^2_{\text{time}} \leq d^2. \tag{5.1}$$

The clusters are then sorted by descending energy deposit. The 10 hits with largest deposit per detector (not segment) are stored in the output tree, zero-suppressed, as listed in Table 5.1. When an active volume is registered with two different indexing paths (u and v)[2], they are treated

[1]todo: Presently, the implementation is a bit cheap, and works with the bounding-boxes instead of the actual lines.

[2]Dual reporting is useful for e.g. double-sided silicon strip detectors, as the actual active volume is the same for the read-outs on both sides.

Name	Unit	Description
`n`		Number of hits.
`t[n]`	ns	Energy weighted time of the hit.
`e[n]`	MeV	Deposited energy of the hit.
`x[n]` `y[n]` `z[n]`	cm	Energy weighted position of the hit.
`i[n]`		Index of the detector subsegment (u).
`j[n]`		Index of alternative detector sub-segmentation (v).
`majprt[n]`		Integer describing the particle type contributing the most energy. (todo: `pdg`)
`maje[n]`	MeV	Energy deposited by particles of type `majprt[n]`.
`othprt[n]`		Integer mask describing all other contributing particles.

Table 5.1: Variables describing each hit by the clusteriser. The name is prefixed by the detector name, and any detector index, like `TST1n`, `TST1t[TST1n]`.

separately. This means that the energy deposits will be reported twice, once for each indexing. Which often corresponds to actual measurements. Special post-processing may be used for some detectors countering this (not implemented yet). The x, y, z coordinates are in the local coordinate system of the detector, i.e. not per detector segment.

The parameters of the gun-produced (original) particle parameters are also stored in the tree, as described in Table 5.2. In order to simplify plotting[3], these variables are by default not zero-suppressed when the gun only can produce one particle per event (only has one item).

[3]Specifically, to counter that ROOT does not perform a double-loop when two array variables with independent indexing are plotted together.

Name	Unit	Description
`gunn`		Number of hits.
`gunpdg[gunn]`		Particle PDG ID.
`gunt[gunn]`	ns	Time of particle emission.
`gunx[gunn]`, `guny[gunn]`, `gunz[gunn]`	cm	Location of the particle emission.
`gunpx[gunn]`, `gunpy[gunn]`, `gunpz[gunn]`	MeV/c	Momentum of the emitted particle.
`gunT[gunn]`	MeV	Kinetic energy of the emitted particle.

Table 5.2: Variables describing each particle produced by the particle gun.

Chapter 6

Simulation overview

6.1 Random number seeds

...

6.2 Range cut (geant4 only)

"Timing underwater. Speed underwater.
That is what half our assignments are about."
Rufus Excalibur ffolkes, North Sea Hijack,
Universal Pictures, 1979.

This section deals with one simulation parameter which alone can affect the simulation time of heavy ions in the LAND setup with an order of magnitude: the range cut — the explicit particle production threshold.

Heavy ions produce a lot of low-energy (delta) electrons. With the default range production cut of GEANT4, (1 mm with the used physics list), this produces a lot of low-energy electrons in gases, e.g. air. Figure 6.1 shows the simulation speed implications. These low-energy particles anyhow almost always deposit their energy just besides the track, so need not be produced at all. Especially when they (or their descendants) do not make it to any active volume. Note that even as the delta-particles are not explicitly produced, the associated energy-loss of the passing particle is accounted for.

For the non-explicitly produced delta-electrons to (if produced) have had a non-negligible effect on measured quantities, the active volumes need to be unshielded, i.e. surrounded (preceded) by material thin enough that it would not have absorbed the secondary electrons. Figure 6.2 show how the range cut value where effects are seen (mainly in secondary (minor) detector hits) varies with the surrounding material thickness. There is approximately a factor of 1000 between the absorbing material thickness and the cut-off

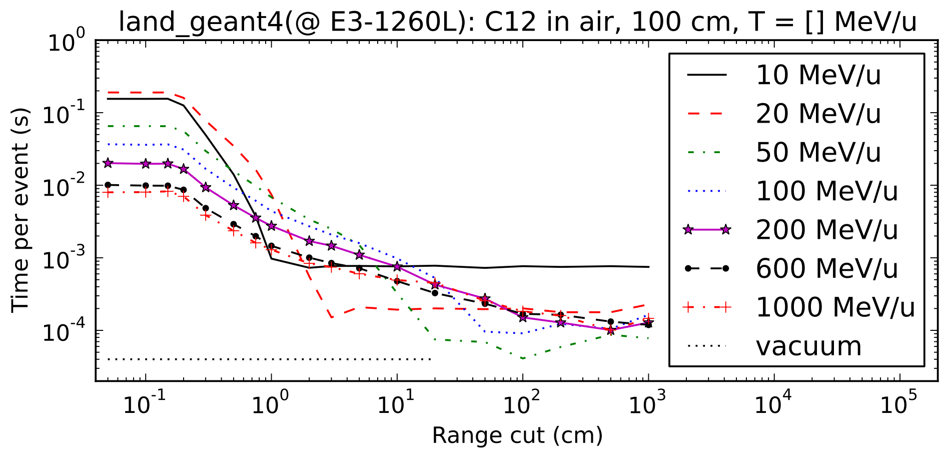

Figure 6.1: Simulation speed of ^{12}C at different energies in 100 cm air, as a function of the range cut, i.e. threshold for explicit particle generation. With range cuts between 0.1 cm – 100 cm, the simulation speed of heavy ion propagation in air can be affected by about two orders of magnitude. (The ordinate might as well have been normalised to the simulation speed in vacuum.)

value, which is quite consistent with the density ratio between air and solids (in particular plastic).

For vacuum, there is no delta-ray production, and thus no need to have an increased production threshold. For detectors operated in air, there is generally always an inactive wrapper material. Either as black-tape wrapping of scintillators, or foils for gaseous detectors. GGLAND sets the range cut to 1 cm for the world volume[1], and 0.1 cm for other volumes. Some detectors may set even lower cut-offs to handle the fact that they have very fine segmentation (e.g. silicon-strip detectors.).

[1]todo: 5 cm seems quite safe.

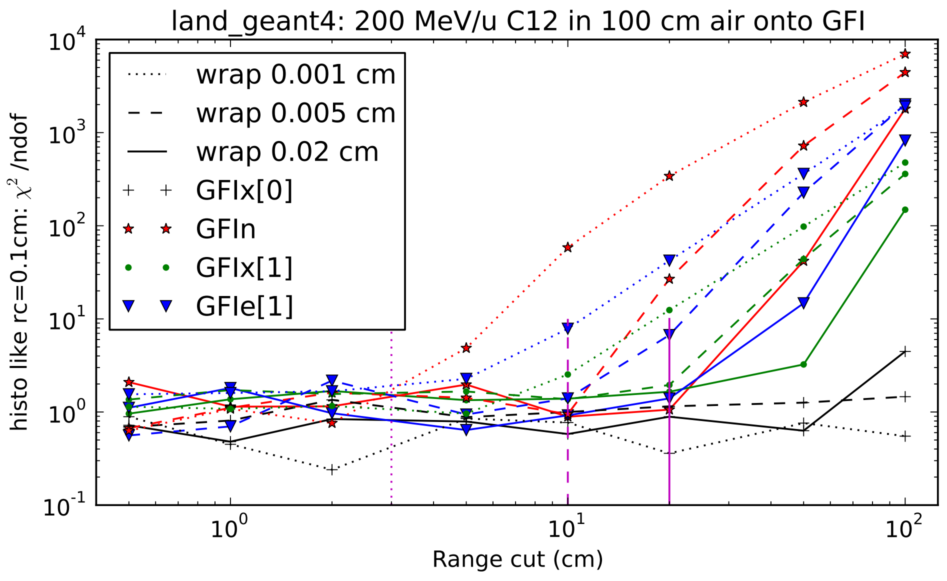

Figure 6.2: Effect on data recorded in active volumes of the GFI detector, as a function of the range cut in air, for three different values of the inactive plastic wrapping thickness. The effect is measured as the similarity of histograms of a few variables to those collected with a low range-cut (0.1 cm). The histograms deviate when $\chi^2/n_{dof} \gg 1$, as indicated by the vertical lines. Effects are seen in the secondary hits (`[1]`), and the number of hits, and not appreciably in the primary hits (`[0]`).

Chapter 7

Internals

The primary goal is to keep the code size down — avoiding code duplication at almost any cost — letting the compiler and helper scripts do the job instead! The second ambition is to use as generic processing schemes (and thereby data storage) as possible and feasible.

It is not by chance that each new detector definition just requires the addition of one file, and a quite small one!

7.1 Forked simulation processes

Chapter 8

Outlook

Having just become operational for simulations going all the way to tracking via the quick-n-dirty general clusteriser, there are several enhancement possibilities:

Capability

- Recording of particles entering selected volumes, to effectively make it possible to see simulated cross-sections "experimentally". Like the `gun` variables for `--tree`, but reversed.
- Include more advanced (approximate) particle guns:
  - Repair the Rutherford distribution calculation.
  - Mimic quasi-free scattering.
- Include more dead material around detectors. (Details less important than having about the right amount of material.)
- Semi-automatic generation of vacuum vessels, beam pipe etc.

Performance

- Use magnetic-free media for volumes (detector) located (completely) outside magnetic fields. (Faster tracking.)
- Adjust production thresholds (ranges in GEANT4) such that far fewer low-energy electrons are created when passing inactive gas volumes, in particular air and/or Helium bags. (Done! But care for the He-volumes too.)
- Handle spawning of processes on other machines, for massive parallelism.

Bibliography

[1] *GEANT, Detector Description and Simulation Tool, CERN Program Library Long Write-up W5013*
http://wwwasdoc.web.cern.ch/wwwasdoc/geant_html3/geantall.html.

[2] S. Agostinelli et al., *Geant4—a simulation toolkit*, NIM A 506 (2003), 250–303.

[3] *The ROOT system homepage* http://root.cern.ch (2013-08-26).

[4] *CERN Program Library*
http://cernlib.web.cern.ch/cernlib/ (2013-08-26).

[5] *Ralf's tracker*
http://ralfplag.de/tracker/.

[6] Particle Data Group, *Monte Carlo Particle Numbering Scheme*
http://pdg.lbl.gov/2012/mcdata/mc_particle_id_contents.html.

[7] J. Kempa, A. Krawczynska, *Low energy muons in the cosmic radiation*, Nuclear Physics B (Proc. Suppl.) 151 (2006) 299-302.

[8] S. Haino et al., *Measurements of primary and atmospheric cosmic-ray spectra with the BESS-TeV spectrometer* Physics Letters B 594 (2004) 35–46.

[9] F. James, *Monte Carlo Phase Space*
http://cdsweb.cern.ch/record/275743/files/CERN-68-15.pdf.