Monte Carlo Particle Lists : MCPL

ESS Detector Group Jamboree, DTU Risø, 2016-09-05

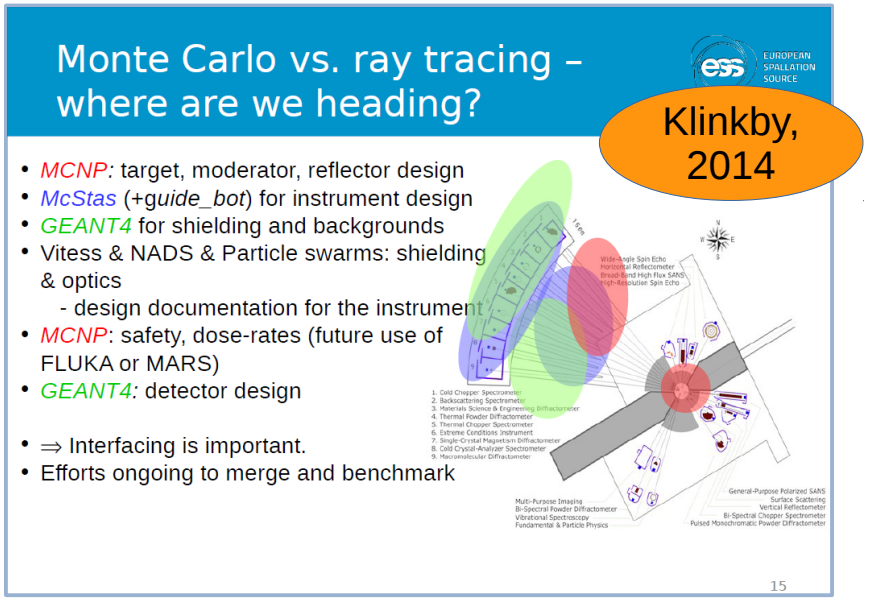
Thomas Kittelmann, ESS
(thomas.kittelmann@esss.se)

MCPL developed with contributions from:

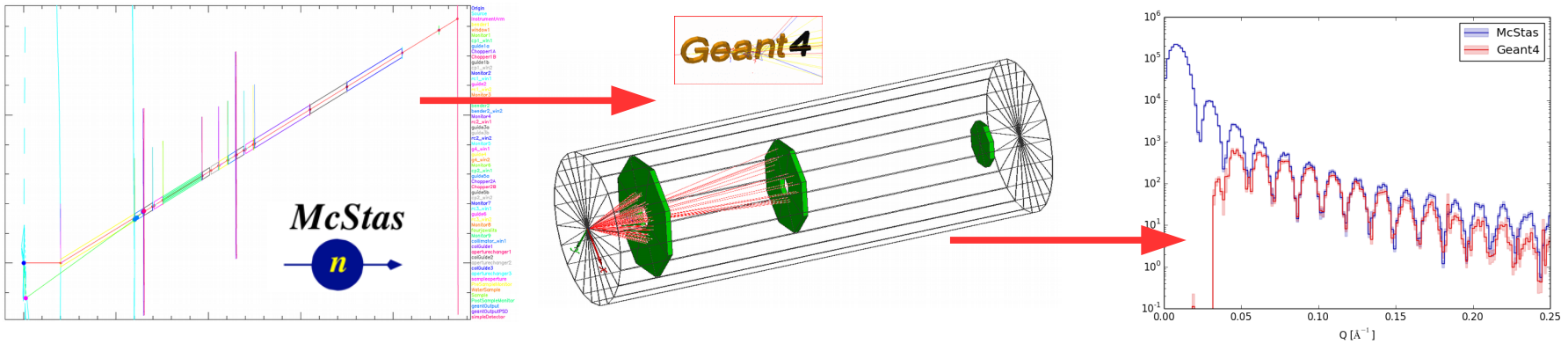
E. Klinkby (DTU), E. Knudsen (DTU), P. Willendrup (DTU, ESS), K. Kanaki (ESS), X. X. Cai (ESS, DTU)

Background / Motivation

- Many different applications in use at ESS for particle simulations.
- Desirable to be able to transfer particles between applications.
- Or reuse within a single application.



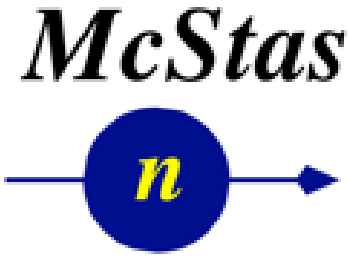
- For detector simulations in Geant4, we are interested in grabbing post-sample output of instrument simulations (usually McStas), and use those as a source.



- Or, grab background particles from MCNP or Geant4 simulations to study shielding and background issues. Outside DG, people have other needs.

How to store and transfer particles? By 2015 we had a jungle of custom solutions at ESS for just 3 apps...

NB: illustration here is surely incomplete...



- Use SSW
- Not between MCNPX & 6

- Custom McStas+MCNP code working one-way via SSW files.

Klinkby, Willendrup

- Custom McStas component?
- Reading SSW files in special format

Klinkby, Knudsen, Willendrup

- Adapted Fortran code for SSW => ASCII.
- Created custom C++ code to get into G4.
- Problems: Not all particle types worked, MCNP6 not supported, difficult work-flow.

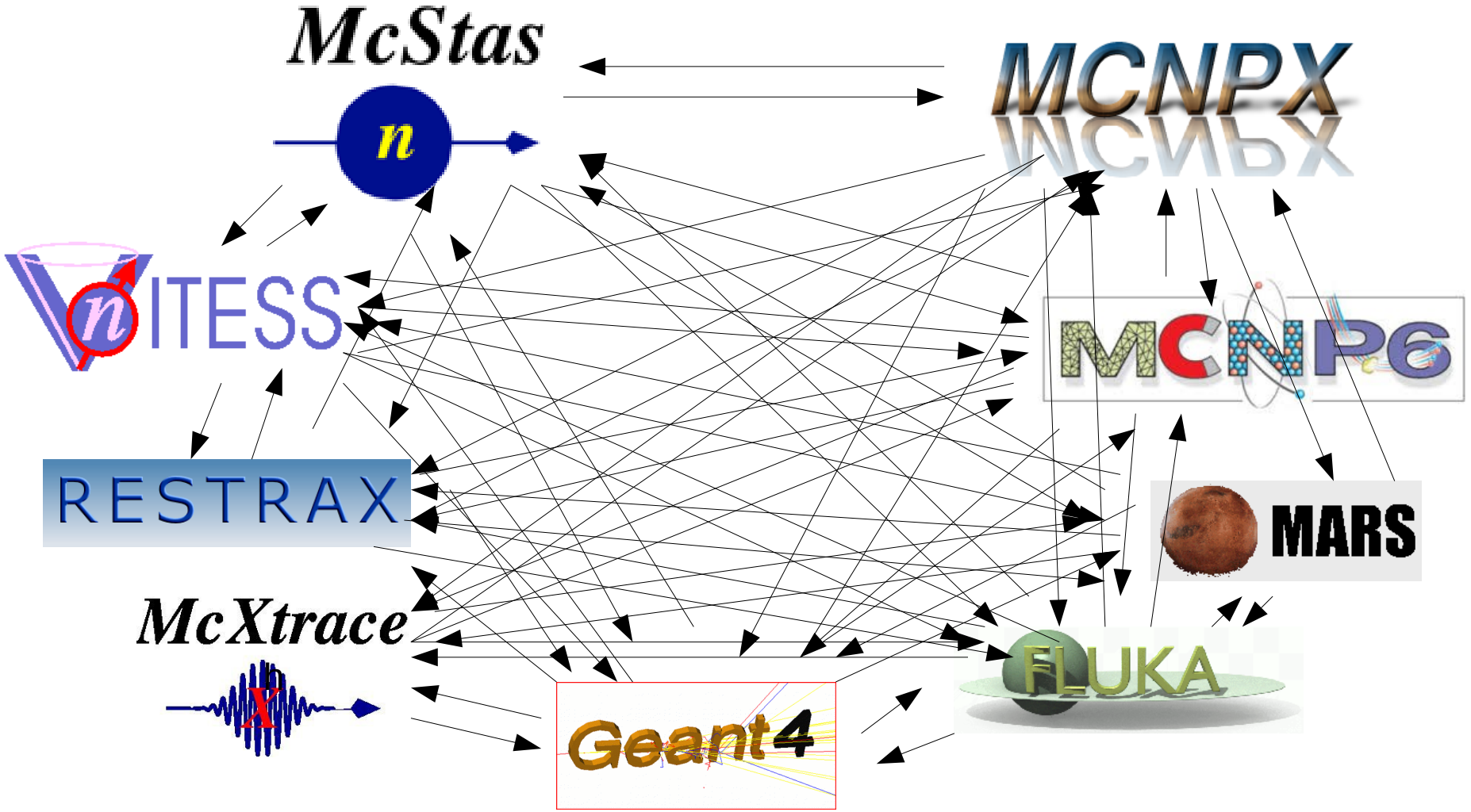
Klinkby, Kittelmann

- Used McStas MonitorND comp. => ASCII files
- Created custom C++ code to get into G4.
- Problems: MonitorND geometry, format non-constant, etc.

Kanaki, Kittelmann

- Saving particles for use within G4@ESS:
- ASCII, HDF5, GRIFF, ROOT
- Quick and dirty, non-portable, or badly suited formats...

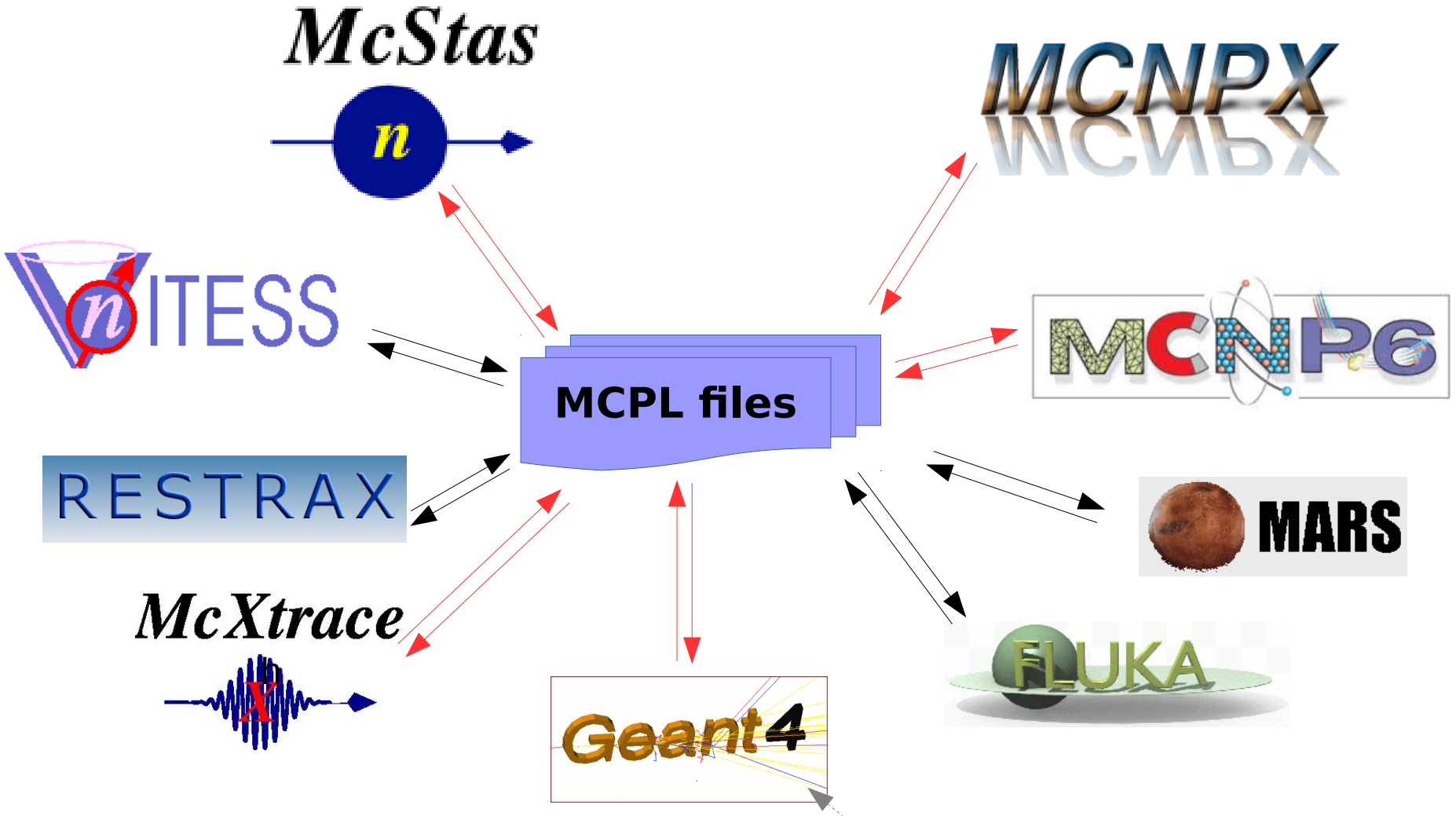
Consider more apps : The jungle gets impossibly tangled...



Disclaimer: Non-exhaustive list of applications...

The solution: A common interchange format.

MCPL: Monte Carlo Particle Lists



In red : already available now (Sep 2016).

Available for standalone Geant4, but the version in dgcode is easier to use and has more features.

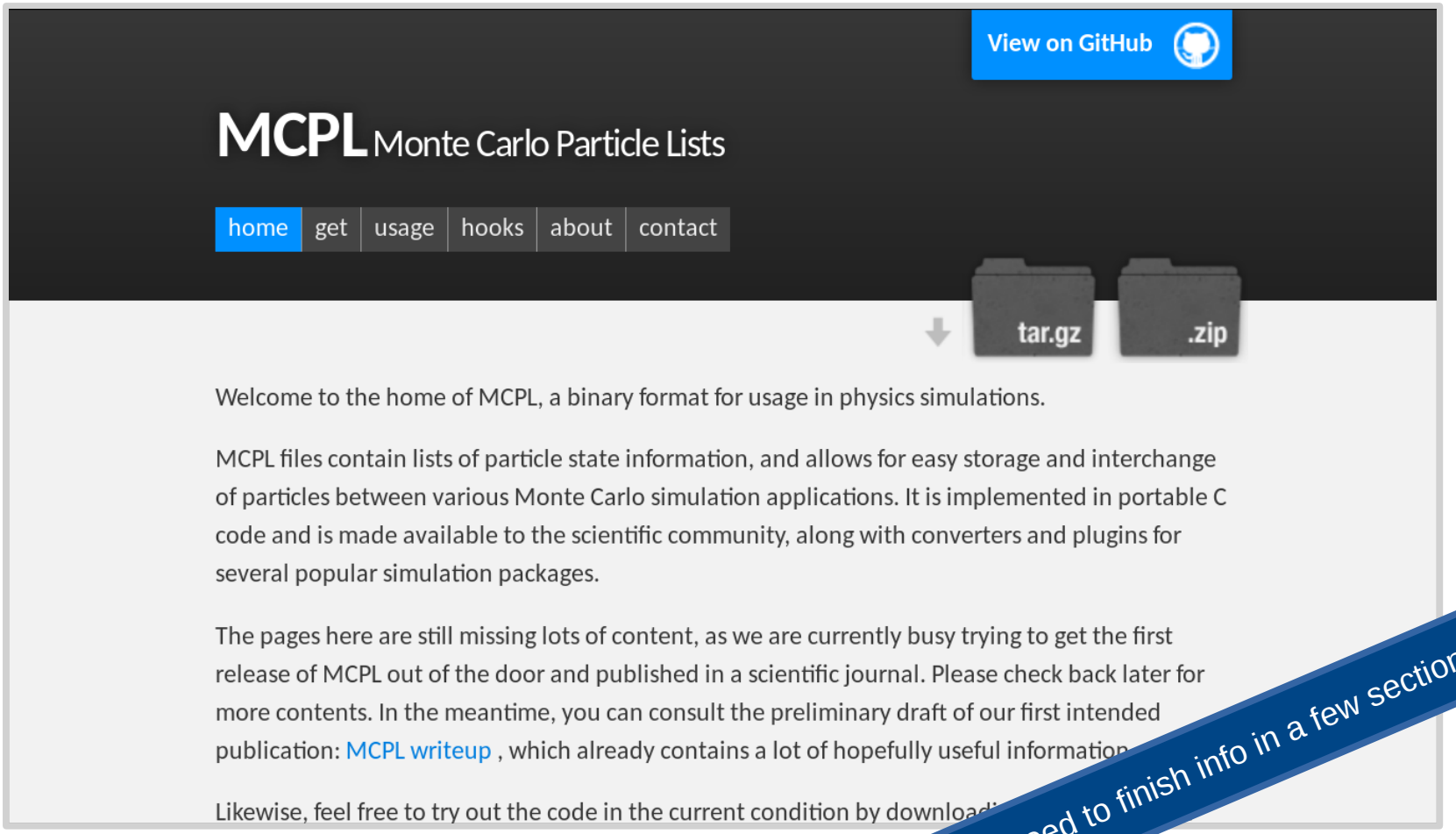
Disclaimer: Non-exhaustive list of applications...

What is MCPL?

MCPL : Monte Carlo Particle Lists

- It is a simple file-format. Each file contains a list of particles.
- The format is flexible: can contain a lot of information if needed, or can contain only minimal information if small file-size is important.
- It is easy to make code dealing with MCPL, so it is easy to make plugins&converters for the various Monte Carlo frameworks. End-users will simply use those converters.
- MCPL files can contain meta-data. This makes it possible to tell what data is in a file, where it came from, how it should be interpreted.
- MCPL comes with tools, such as for printing and plotting contents.

Official website & code @ GitHub: https://mctools.github.io/mcpl/



Still need to finish info in a few sections…

Paper describing MCPL in detail about to be submitted

Monte Carlo Particle Lists : MCPL

T Kittelmann[a,*], E Klinkby[b], E Knudsen[c], P Willendrup[c], X X Cai[a,b], K Kanaki[a]

[a] *European Spallation Source ERIC, Sweden*
[b] *DTU Nutech, Technical University of Denmark, Denmark*
[c] *DTU PHYSICS, Technical University of Denmark, Denmark*

Draft version available on MCPL website

Abstract

A binary format with lists of particle state information, for interchanging particles between various Monte Carlo simulation applications, is presented. Portable `C` code for file manipulation is made available to the scientific community, along with converters and plugins for several popular simulation packages.

Lots of details!
More than most end-users will need to know or care about :-)

Browsing an MCPL file with the MCPL tool

in dgcode, just run: "`ess_mcpl_tool <name-of-MCPL-file>`"

```
Opened MCPL file myfile.mcpl.gz:

  Basic info
    Format             : MCPL-2
    No. of particles   : 5037156
    Header storage     : 818 bytes
    Data storage       : 181337616 bytes

  Custom meta data
    Source             : "Geant4"
    Number of comments : 8
          -> comment 0 : "Created with the Geant4 MCPLWriter in the ESS/dgcode framework"
          -> comment 1 : "MPCLWriter volumes considered : ['RecordFwd']"
          -> comment 2 : "MPCLWriter steps considered : <at-volume-exit>"
          -> comment 3 : "MPCLWriter write filter : <unfiltered>"
          -> comment 4 : "MPCLWriter user flags : <disabled>"
          -> comment 5 : "MPCLWriter track kill strategy : <none>"
          -> comment 6 : "ESS/dgcode geometry module : G4StdGeometries/GeoSlab"
          -> comment 7 : "ESS/dgcode generator module : G4StdGenerators/SimpleGen"
    Number of blobs    : 2
          -> 74 bytes of data with key "ESS/dgcode_geopars"
          -> 231 bytes of data with key "ESS/dgcode_genpars"

  Particle data format
    User flags         : no
    Polarisation info  : no
    Fixed part. type   : no
    FP precision       : single
    Endianness         : little
    Storage            : 36 bytes/particle

index     pdgcode   ekin[MeV]       x[cm]       y[cm]       z[cm]         ux          uy          uz    time[ms]      weight
    0        2112  4.0061e-08     -11.518      -2.744          40   -0.60697   -0.093797     0.78917     0.22354           1
    1        2112     2.5e-08           0           0          40          0           0           1      0.1829           1
    2          22      7.7251      7.8603     -6.7903          40   0.072796    -0.20272     0.97653     0.33498           1
    3        2112  1.8481e-08     -21.168      4.4662          40   -0.70384      0.1485     0.69466     0.24732           1
    4          22       0.511      27.191      7.7111          40    0.12641   -0.034978     0.99136     0.13778           1
    5          22       0.031     -30.093      19.067          40    0.10979     0.84395     0.52507     0.27059           1
    6          22       1.592         -50      2.7616      27.847   -0.66425     0.66981     0.33186     0.27059           1
    7          22      1.4402      16.313     -15.255          40   0.062836    -0.14628     0.98724     0.11248           1
```

Browsing an MCPL file with the MCPL tool

in dgcode, just run: "`ess_mcpl_tool <name-of-MCPL-file>`"

```
Opened MCPL file myfile.mcpl.gz:

  Basic info
    Format             : MCPL-2
    No. of particles   : 5037156
    Header storage     : 818 bytes
    Data storage       : 181337616 bytes

  Custom meta data
    Source             : "Geant4"
    Number of comments : 8
          -> comment 0 : "Created with the Geant4 MCPLWriter in the ESS/dgcode framework"
          -> comment 1 : "MPCLWriter volumes considered : ['RecordFwd']"
          -> comment 2 : "MPCLWriter steps considered : <at-volume-exit>"
          -> comment 3 : "MPCLWriter write filter : <unfiltered>"
          -> comment 4 : "MPCLWriter user flags : <disabled>"
          -> comment 5 : "MPCLWriter track kill strategy : <none>"
          -> comment 6 : "ESS/dgcode geometry module : G4StdGeometries/GeoSlab"
          -> comment 7 : "ESS/dgcode generator module : G4StdGenerators/SimpleGen"
    Number of blobs    : 2
          -> 74 bytes of data with key "ESS/dgcode_geopars"
          -> 231 bytes of data with key "ESS/dgcode_genpars"

  Particle data format
    User flags         : no
    Polarisation info  : no
    Fixed part. type   : no
    FP precision       : single
    Endianness         : little
    Storage            : 36 bytes/particle
```

Columns of particle data (1 row = 1 particle)
In this file: No *userflags* or *polarisation*

index	pdgcode	ekin[MeV]	x[cm]	y[cm]	z[cm]	ux	uy	uz	time[ms]	weight
0	2112	4.0061e-08	-11.518	-2.744	40	-0.60697	-0.093797	0.78917	0.22354	1
1	2112	2.5e-08	0	0	40	0	0	1	0.1829	1
2	22	7.7251	7.8603	-6.7903	40	0.072796	-0.20272	0.97653	0.33498	1
3	2112	1.8481e-08	-21.168	4.4662	40	-0.70384	0.1485	0.69466	0.24732	1
4	22	0.511	27.191	7.7111	40	0.12641	-0.034978	0.99136	0.13778	1
5	22	0.031	-30.093	19.067	40	0.10979	0.84395	0.52507	0.27059	1
6	22	1.592	-50	2.7616	27.847	-0.66425	0.66981	0.33186	0.27059	1
7	22	1.4402	16.313	-15.255	40	0.062836	-0.14628	0.98724	0.11248	1

Browsing an MCPL file with the MCPL tool

in dgcode, just run: "`ess_mcpl_tool <name-of-MCPL-file>`"

```
Opened MCPL file myfile.mcpl.gz:

  Basic info
    Format             : MCPL-2
    No. of particles   : 5037156
    Header storage     : 818 bytes
    Data storage       : 181337616 bytes

  Custom meta data
    Source             : "Geant4"
    Number of comments : 8
          -> comment 0 : "Created with the Geant4 MCPLWriter in the ESS/dgcode framework"
          -> comment 1 : "MPCLWriter volumes considered : ['RecordFwd']"
          -> comment 2 : "MPCLWriter steps considered : <at-volume-exit>"
          -> comment 3 : "MPCLWriter write filter : <unfiltered>"
          -> comment 4 : "MPCLWriter user flags : <disabled>"
          -> comment 5 : "MPCLWriter track kill strategy : <none>"
          -> comment 6 : "ESS/dgcode geometry module : G4StdGeometries/GeoSlab"
          -> comment 7 : "ESS/dgcode generator module : G4StdGenerators/SimpleGen"
    Number of blobs    : 2
          -> 74 bytes of data with key "ESS/dgcode_geopars"
          -> 231 bytes of data with key "ESS/dgcode_genpars"

  Particle data format
    User flags         : no
    Polarisation info  : no
    Fixed part. type   : no
    FP precision       : single
    Endianness         : little
    Storage            : 36 bytes/particle
```

index	pdgcode	ekin[MeV]	x[cm]	y[cm]	z[cm]	ux	uy	uz	time[ms]	weight
0	2112	4.0061e-08	-11.518	-2.744	40	-0.60697	-0.093797	0.78917	0.22354	1
1	2112	2.5e-08	0	0	40	0	0	1	0.1829	1
2	22	7.7251	7.8603	-6.7903	40	0.072796	-0.20272	0.97653	0.33498	1
3	2112	1.8481e-08	-21.168	4.4662	40	-0.70384	0.1485	0.69466	0.24732	1
4	22	0.511	27.191	7.7111	40	0.12641	-0.034978	0.99136	0.13778	1
							0.84395	0.52507	0.27059	1
							0.66981	0.33186	0.27059	1
							-0.14628	0.98724	0.11248	1

Columns of particle data (1 row = 1 particle)
In this file: No *userflags* or *polarisation*

PDG codes: 2112 = neutron, 22 = gamma
More at http://pdg.lbl.gov/2015/reviews/rpp2015-rev-monte-carlo-numbering.pdf

Browsing an MCPL file with the MCPL tool

in dgcode, just run: "`ess_mcpl_tool <name-of-MCPL-file>`"

```
Opened MCPL file myfile.mcpl.gz:

  Basic info
    Format              : MCPL-2
    No. of particles    : 5037156
    Header storage      : 818 bytes
    Data storage        : 181337616 bytes

  Custom meta data
    Source              : "Geant4"
    Number of comments  : 8
          -> comment 0  : "Created with the Geant4 MCPLWriter in the ESS/dgcode f
          -> comment 1  : "MPCLWriter volumes considered : ['RecordFwd']"
          -> comment 2  : "MPCLWriter steps considered : <at-volume-exit>"
          -> comment 3  : "MPCLWriter write filter : <unfiltered>"
          -> comment 4  : "MPCLWriter user flags : <disabled>"
          -> comment 5  : "MPCLWriter track kill strategy : <none>"
          -> comment 6  : "ESS/dgcode geometry module : G4StdGeometries/GeoSlab"
          -> comment 7  : "ESS/dgcode generator module : G4StdGenerators/SimpleGe
    Number of blobs     : 2
          -> 74 bytes of data with key "ESS/dgcode_geopars"
          -> 231 bytes of data with key "ESS/dgcode_genpars"

  Particle data format
    User flags          : no
    Polarisation info   : no
    Fixed part. type    : no
    FP precision        : single
    Endianness          : little
    Storage             : 36 bytes/particle
```

Custom meta-data

- This file is from Geant4 in dgcode
- Comments reminding us of setup used to create file
- Binary "blobs" keep more complete configuration details (here dgcode geo/gen parameters, could be McStas instrument file or MCNP input deck).

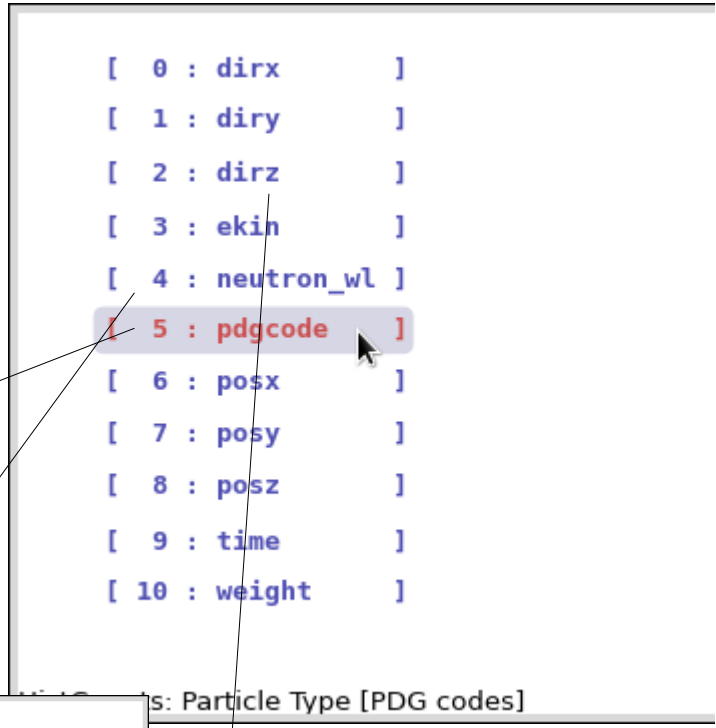
Columns of particle data (1 row = 1 particle)
In this file: No *userflags* or *polarisation*

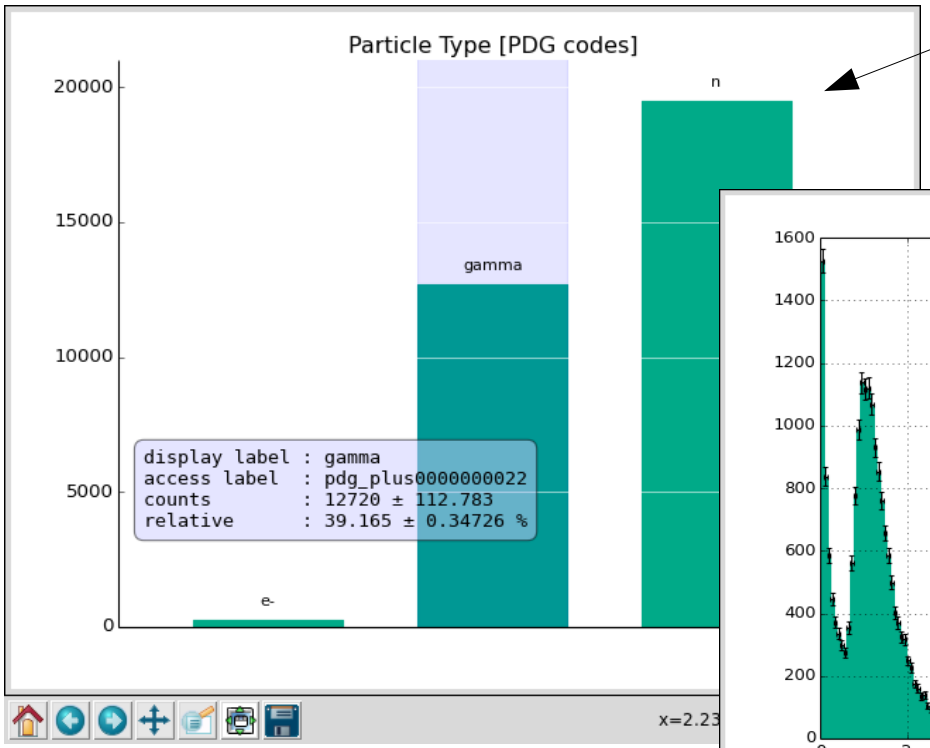
index	pdgcode	ekin[MeV]	x[cm]	y[cm]	z[cm]	ux	uy	uz	time[ms]	weight
0	2112	4.0061e-08	-11.518	-2.744	40	-0.60697	-0.093797	0.78917	0.22354	1
1	2112	2.5e-08	0	0	40	0	0	1	0.1829	1
2	22	7.7251	7.8603	-6.7903	40	0.072796	-0.20272	0.97653	0.33498	1
3	2112	1.8481e-08	-21.168	4.4662	40	-0.70384	0.1485	0.69466	0.24732	1
4	22	0.511	27.191	7.7111	40	0.12641	-0.034978	0.99136	0.13778	1
							0.84395	0.52507	0.27059	1
							0.66981	0.33186	0.27059	1
							-0.14628	0.98724	0.11248	1

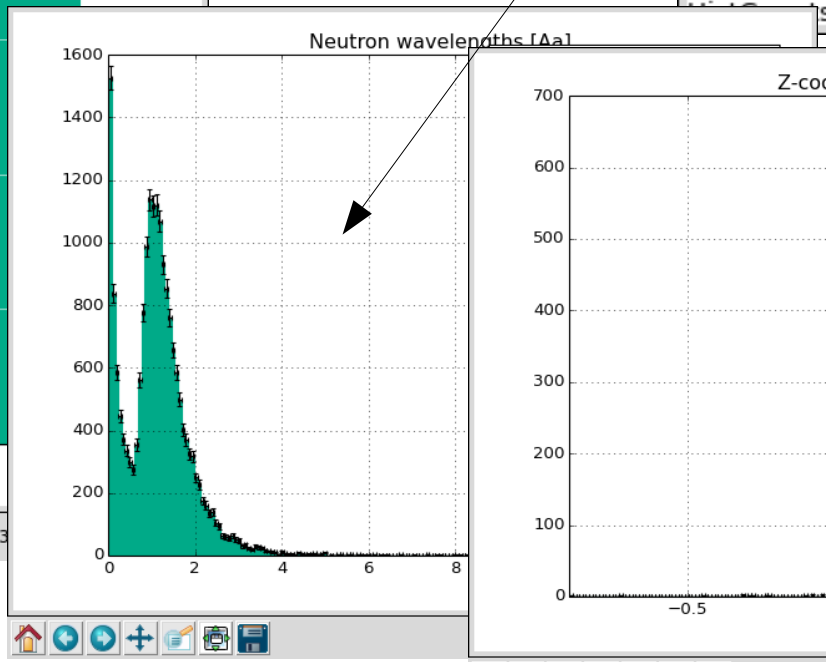
PDG codes: 2112 = neutron, 22 = gamma
More at http://pdg.lbl.gov/2015/reviews/rpp2015-rev-monte-carlo-numbering.pdf

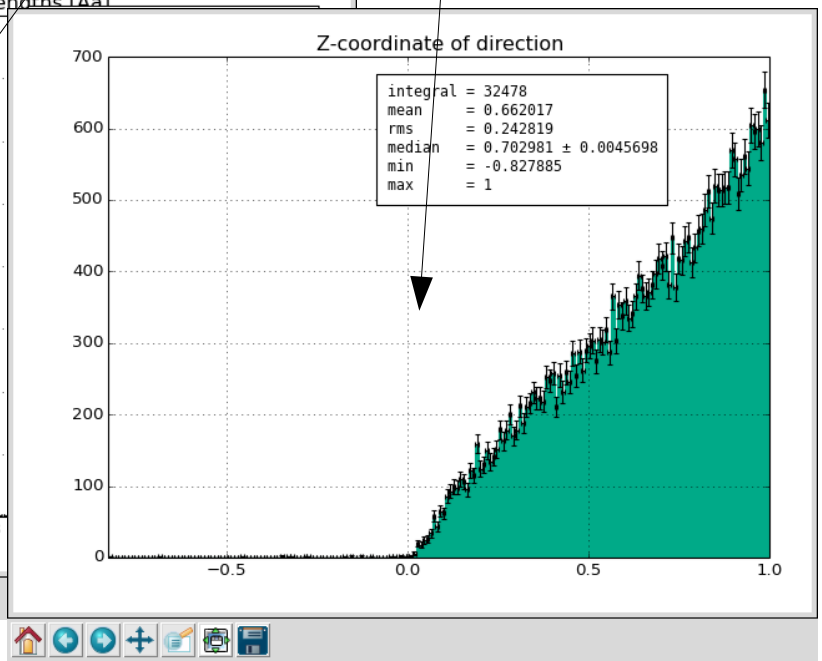
In dgcode: Extra MCPL tools

```
$> ess_mcplextra_browse myfile.mcpl.gz
```







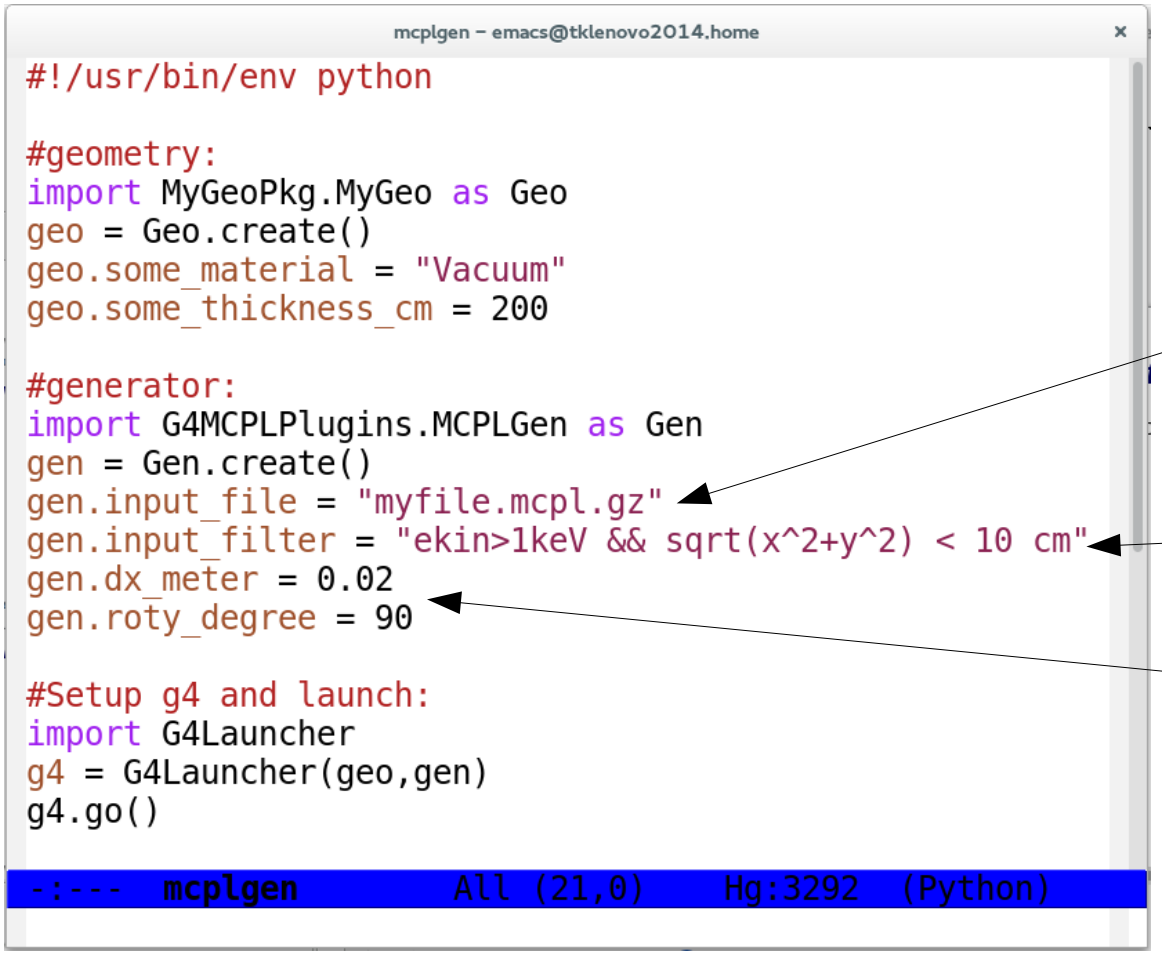


Can also produce 2D plots of variable correlations...

And can also be used to study effects of filters (more in a few slides)

In dgcode : Use MCPL files as input to Geant4 simulation

- Exists as generator module “MCPLGen” for your sim-script (resulting G4Event's will have 1 primary particle each):



Can read compressed (.gz) files directly.

Optional set an input_filter to select particles from file (more next slide)

Optionally apply coordinate transformations, if needed

- Normally simulations default to 10 events, but when using MCPL files as input, the default will be to run over all the (selected) particles in the file. An upper limit can be requested with the usual `--nevts` or `-n` command line flag.

In dgcode: `input_filter` *examples*

(this feature brought to you by our new ExpressionParser)

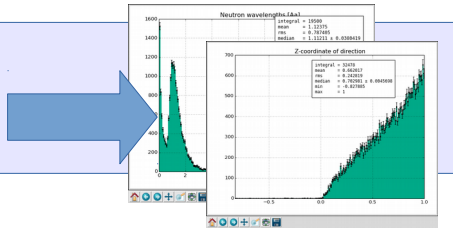
- **`"pdgcode==22 && ekin<2MeV"`** : *Gammas only, Ekin<2MeV*
- **`"is_neutron && neutron_wl > 1.0Aa"`** : Long-wavelength neutrons only.
- ***`"userflags==2"`*** *: Select on value of userflags field, whatever this means (in the files from Loki-McStas, this would mean "neutron interacted with the sample").*
- ***`"time > 2.0ms"`***
- ***`"sqrt(x^2+y^2)<10cm"`***
- ***`"acos(uz) < 5degree"`***

Expressions can (should) use physics units and constants, as well as most mathematical functions and logical operators.

Available variables
x, y, z : position
ux, uz, uz : direction (normalised)
polx, poly, polz : polarisation
weight : statistical weight
userflags : custom integer
time & ekin : time stamp and Ekin
pdgcode : particle type integer
is_gamma : shortcut for "pdgcode==22"
is_neutron : shortcut for "pdgcode==2112"

Fine-tune and test filters before simulation:

```
ess_mcplextra_browse myfile.mcpl.gz where "time>2ms"
```



In dgcode : Create new MCPL files from Geant4 simulations

- No need for code-changes, just supply `--mcpl` to any sim-script:
  - **`ess_myproject_sim --nevts=100000 --mcpl=SomeVol`**
  - This results in capture (and halt further simulation of) any particle entering Geant4 volumes named "SomeVol" to an MCPL file.
  - By default, captured particles are also “killed” (i.e. won't be simulated further). This is to prevent accidental double-counting.
  - Details can be fine-tuned, run with `--mcpl=help` for instructions.
- Of course, can also be done from python instead of the command line:
  - Example in `packages/Validation/UnitTests/G4MCPLTests/scripts/mcplwrite`
- Courtesy of our new ExpressionParser, MCPL userflags can be composed with user-provided expressions, here embedding two separate pieces of info:
  - **`--mcpl=”MyVol withflags trk.is_primary+10*step.volcopyno”`**
  - The provided expression will also be added as a comment in the MCPL header for later reference.

In dgcode : integrated with multiprocessing

```
(thki@tklenovo2014 tkdgcode_mcpl)> ess_lokisim_tube_sim --mcpl=CountingGas -j4

*************************************************************
 Geant4 version Name: geant4-10-00-patch-03    (31-October-2014)
                      Copyright : Geant4 Collaboration
                      Reference : NIM A 506 (2003), 250-303
                            WWW : http://cern.ch/geant4
*************************************************************

G4Launcher:: Pre-init started
(-------------------snip-------------------)
=============================================================
G4Launcher:: Forking into 4 processes.
G4Launcher::[proc1] Begin simulation of event 1 [seed 541357962368]
G4Launcher::[proc2] Begin simulation of event 1 [seed 1082592467947]
G4Launcher::[proc1] Begin simulation of event 2 [seed 10746979579748687815]
G4Launcher::[proc0] Begin simulation of event 1 [seed 123456789]
G4Launcher::[proc3] Begin simulation of event 1 [seed 1623826973526]
G4Launcher::[proc2] Begin simulation of event 2 [seed 1080350825
G4Launcher::[proc1] Begin simulation of event 3 [seed 2465056555
G4Launcher::[proc2] Simulation done
G4Launcher::[proc1] Simulation done
G4Launcher::[proc0] Begin simulation of event 2 [seed 353863685956056017 3]
G4Launcher::[proc3] Begin simulation of event 2 [seed 963218188694824 5369]
G4Launcher::[proc3] Simulation done
G4Launcher::[proc0] Begin simulation of event 3 [seed 1482835384047974398]
G4Launcher::[proc0] Simulation done
G4Launcher::[proc0] Simulation done in all processes
MCPLWriter: Merging output of proc1 into particles.mcpl
MCPLWriter: Merging output of proc2 into particles.mcpl
MCPLWriter: Merging output of proc3 into particles.mcpl
MCPLWriter: Done merging output from 4 processes into file particles.mcpl
MCPL: Attempting to compress file particles.mcpl with gzip
MCPL: Succesfully compressed file into particles.mcpl.gz
(thki@tklenovo2014 tkdgcode_mcpl)>
```

Nothing special to do: Simply request both MP and MCPL...

User still gets just a single output file (here just with 10 events)

Using MCPL files as *input* to multi-processing jobs is equally easy: Particles gets distributed automatically to the available processes.

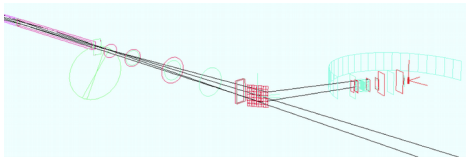
In dgcode : extract subset of particles into new file with the `ess_mcplextra_filterfile` command

(another feature brought to you by our new ExpressionParser)

```
$> ess_mcplextra_filterfile in.mcpl new.mcpl <OPTIONS>
```

- Example **<OPTIONS>**:
  - **“is_neutron”**
    new.mcpl will contain all neutrons from in.mcpl
  - **“is_neutron || is_photon”**
    new.mcpl will contain all neutrons or photons from in.mcpl
  - **“is_neutron && neutron_wl > 2.0 Aa”**
    new.mcpl will contain all long-wavelength neutrons from in.mcpl
  - **-l1000**
    new.mcpl will contain first 1000 particles from in.mcpl
  - **-l1000 “ekin > 10 MeV”**
    new.mcpl will contain first 1000 high energy particles from in.mcpl
- Meta-data transferred and augmented with record of transformation.

How to activate in McStas



- Just add two lines in your McStas instrument file at the appropriate position (for instance, right after the sample component):

```
COMPONENT mcplout = MCPL_output(filename="myfile")
AT(0,0,0) RELATIVE PREVIOUS
```

- This captures into myfile.mcpl.gz the full state of all neutrons as they leave the previous component (with coordinates relative to that component).
- So put the two lines above after the sample component in McStas, and you can fire neutrons into a Geant4-based detector simulation by putting the Geant4 sample position as parameters to the MCPLGen module in dgcode.
- For how to add custom userflags, or use MCPL as *input* to McStas, see the official examples:
  - mcstas-comps/examples/Test_MCPL_output.inst
  - mcstas-comps/examples/Test_MCPL_input.inst

More info on website & in section 3.3 of writeup!

NOTE: The MCPL code is already part of McStas 2.3, but a few bugs were fixed late, so need to copy a fixed version of MCPL_output.comp into your rundir (will not be needed in future releases).

Summary and outlook

- Collaboration between us (dgcode/Geant4), McStas developers & the MCNP community at ESS, have resulted in a new standard particle interchange format.
- It can be (and is) used already now!
- dgcode users have a bunch of extra handy tools for visualisation, filtering & editing of files.
- Interest from community (presentations to IAEA, ~~IWSMT~~, SINE2020, ...)
- Still a few loose ends to tidy up:
  - A lot of documentation exists, but needs polish in DG wiki + MCPL website.
  - Submit publication (this week!)
- Could imagine adding more tools for investigating and editing MCPL files. Or making the ones we have available to a wider audience.

Additional material

Reference: meta-data in MCPL header

File header information	
Field	*Description*
File type magic number 0x4d43504c ("MCPL")	All MCPL files start with this 4-byte word.
Version	File format version.
Endianness	Whether numbers in file are in little- or big-endian format.
Number of particles in file	64 bit integer.
Flag : Particles have polarisation info	If false, all loaded particles will have polarisation vectors (0,0,0).
Flag : Particles have "userflags" field	If false, all loaded particles will have userflags 0x00000000.
Flag : Particle info use double-precision	If true, floating points storage use double-precision.
Global pdgcode	If this 32 bit integer is non-zero, all loaded particles will have this pdgcode.
Source name	String indicating the application which created the MCPL file.
Comments	A variable number of comments (strings) added at file creation.
Binary blobs	A variable number of binary data blobs, indexed by keys (strings). This allows arbitrary custom data to be embedded.

Table 1: Information available in the header section of `MCPL` files.

Reference: Particle state data in MCPL

Total: 32-96B (before compression)

Particle state information		
Field	*Description*	*Bytes of storage used per entry (FP = 4 or 8 bytes)*
PDG code	32 bit integer indicating particle type.	0 or 4
Position	Vector, values in centimetres.	3FP
Direction	Unit vector along the particle momentum.	2FP
Kinetic energy	Value in MeV.	1FP
Time	Value in milliseconds.	1FP
Weight	Weight or intensity.	1FP
Polarisation	Vector.	0 or 3FP
User flags	32 bit integer with custom info.	0 or 4

Table 2: Particle state information available and uncompressed storage requirements for each entry in the data section of `MCPL` files.

Reference: C-code for reading MCPL file

Note: Normally most users would instead activate pre-written converters & plugins for their applications

Listing 1: Simple example for looping over all particles in an existing MCPL file

```
#include "mcpl.h"

void example()
{
  mcpl_file_t f = mcpl_open_file("myfile.mcpl");

  const mcpl_particle_t* p;

  while ( ( p = mcpl_read(f) ) ) {

    /* Particle properties can here be accessed
       through the pointer "p":

       p->pdgcode
       p->position[k] (k=0,1,2)
       p->direction[k] (k=0,1,2)
       p->polarisation[k] (k=0,1,2)
       p->ekin
       p->time
       p->weight
       p->userflags
    */

  }

  mcpl_close_file(f);
}
```

Reference: C-code for creating MCPL file

Note: Normally most users would instead activate pre-written converters & plugins for their applications

Listing 2: Simple example for creating an MCPL file with 1000 particles.

```
#include "mcpl.h"

void example()
{
  mcpl_outfile_t f = mcpl_create_outfile("myfile.mcpl");
  mcpl_hdr_set_srcname(f,"MyAppName-1.0");

  /* Tune file options or add custom comments or
     binary data into the header:

     mcpl_enable_universal_pdgcode(f,myglobalpdgcode);
     mcpl_enable_userflags(f);
     mcpl_enable_polarisation(f);
     mcpl_enable_doubleprec(f);
     mcpl_hdr_add_comment(f,"Some comment.");
     mcpl_hdr_add_data(f,"mydatakey",
                       my_datalength, my_databuf)
  */

  mcpl_particle_t* p = mcpl_get_empty_particle(f);

  int i;
  for ( i = 0; i < 1000; ++i ) {

    /* The following particle properties must
       always be set here:

       p->position[k] (k=0,1,2)
       p->direction[k] (k=0,1,2)
       p->ekin
       p->time
       p->weight

       These should also be set when required by
       file options:

       p->pdgcode
       p->userflags
       p->polarisation[k] (k=0,1,2)
    */

    mcpl_add_particle(f,p);
  }

  mcpl_close_outfile(f);
}
```

Reference: C-code for extracting subset of particles from one MCPL file into a new one

Listing 3: Example extracting low-energy neutrons (pdgcode 2112) from an MCPL file.

```c
#include "mcpl.h"

void example() {

  /* open files, transfer meta-data, add comment */

  mcpl_file_t fi = mcpl_open_file("myfile.mcpl");
  mcpl_outfile_t fo = mcpl_create_outfile("new.mcpl");
  mcpl_transfer_metadata(fi, fo);
  mcpl_hdr_add_comment(fo,"Extracted neutrons with ekin<0.1MeV");

  /* transfer selected particles */

  const mcpl_particle_t* particle;
  while ( ( particle = mcpl_read(fi) ) ) {
    if ( particle->pdgcode == 2112 && particle->ekin < 0.1 )
      mcpl_add_particle(fo,particle);
  }

  /* finish up */

  mcpl_closeandgzip_outfile(fo);
  mcpl_close_file(fi);
}
```