Status of IAEAphsp example

Miguel A. Cortés-Giraldo(*), **C. G. Okolinta**

Dep. Atomic, Molecular and Nuclear Physics

Universidad de Sevilla (Spain)

(*) miancortes@us.es

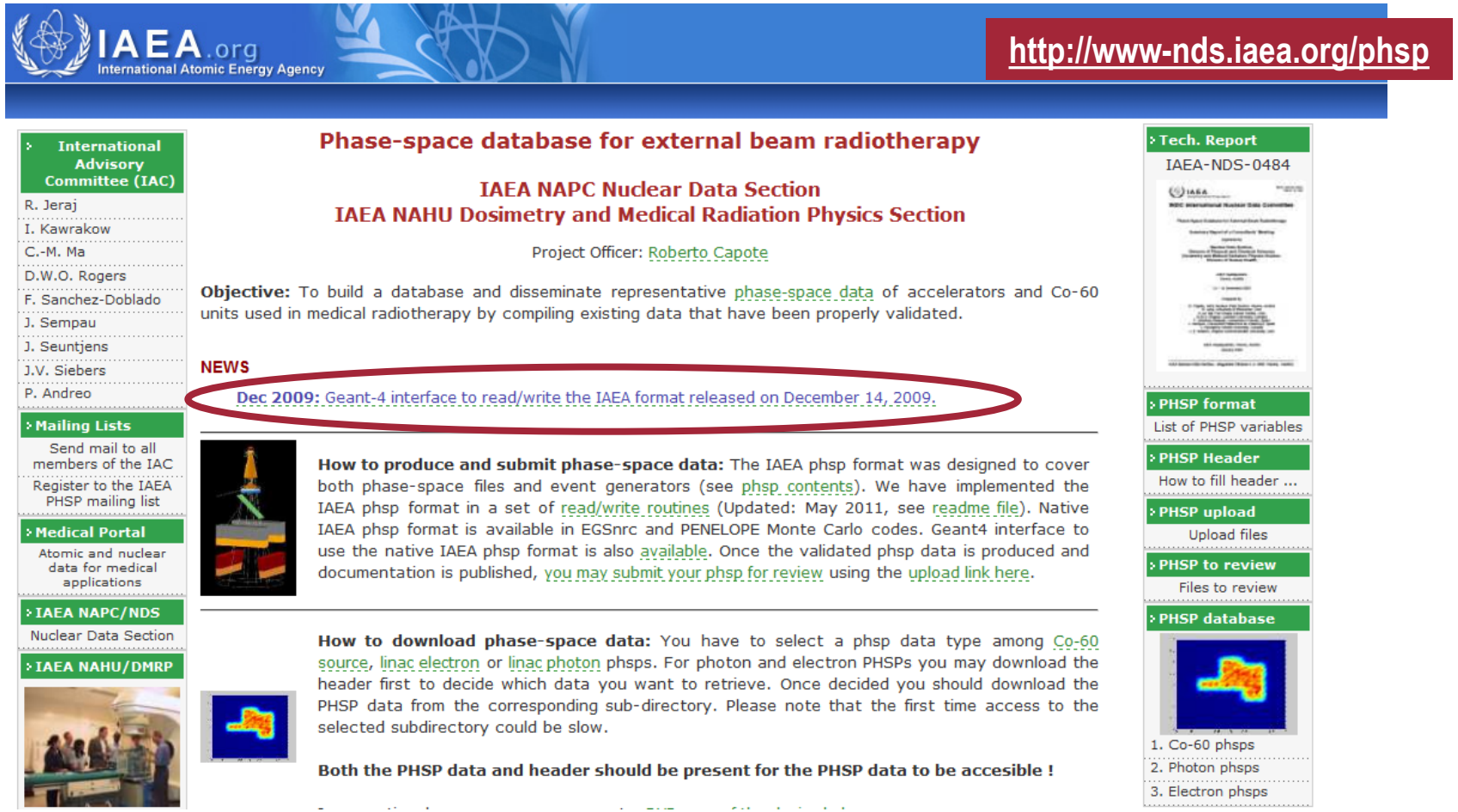
28th Geant4 Collaboration Meeting

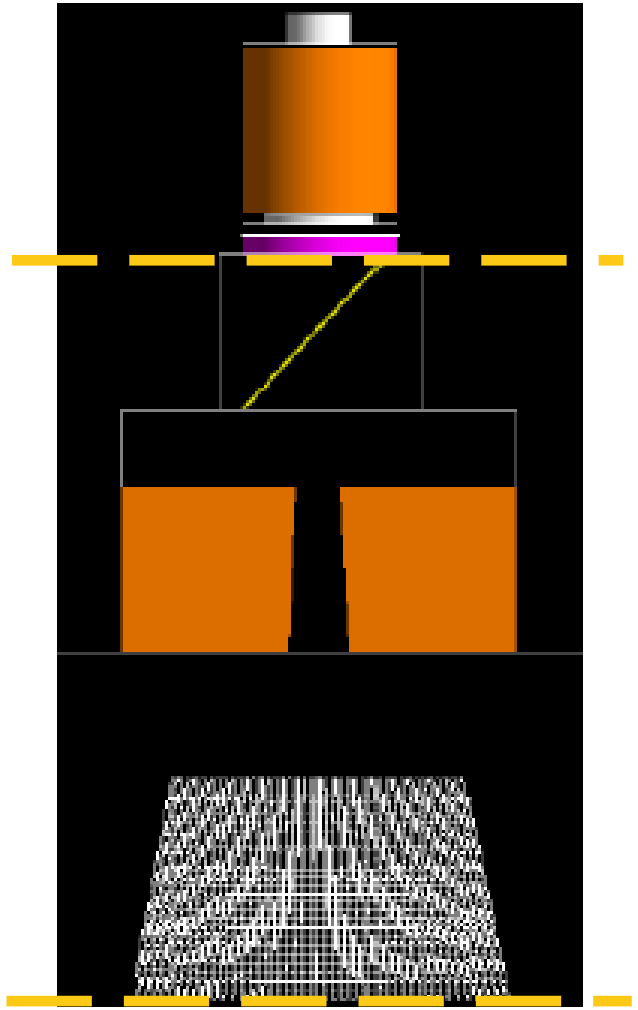
Hokkaido University

Sapporo (Japan), September 26, 2023.

Background

- **IAEAphsp:** Standardized format to use phase-space files produced from different codes.





M.A. Cortés-Giraldo *et al.*, IJRB 88: 200-8 (2012)

IAEAphsp code – current features

- **Focus:** to show clearly how to work with IAEAphsp files.
  - Very simple geometry & scoring (mostly to show that this works!)
- A controlled IAEAphsp file incorporated for testing purpose.
  - Just 1000 particles (200 of each kind: e-, e+, gamma, neutron, proton)
  - Dynamic variables are given values calculated from position within the file.
    - Energy, position, incremental history number (*n_stat*), etc.
- Compiled with latest version of IAEAphsp routines (sep-2013):

h iaea_config.h	
h iaea_header.h	C++ iaea_header.cc
h iaea_phsp.h	C++ iaea_phsp.cc
h iaea_record.h	C++ iaea_record.cc
h utilities.h	C++ utilities.cc

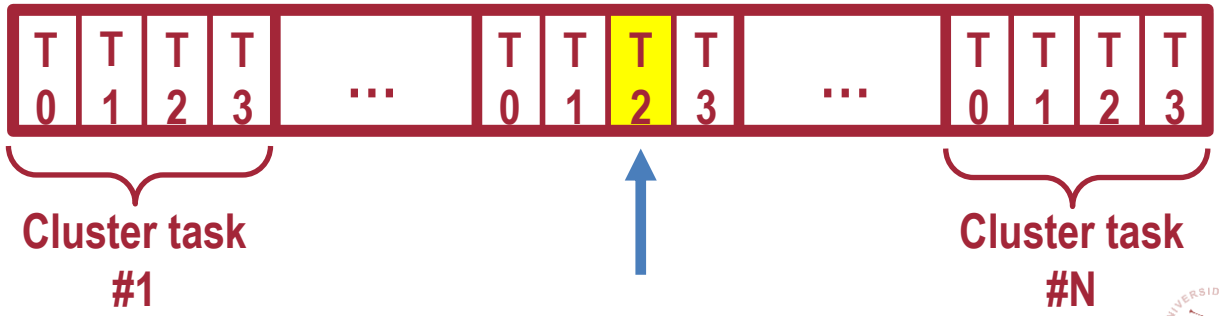
- **G4IAEAphspReader** class finished and tested.
  - **Messenger** class is also available.

G4IAEAphspReader MT class

- Derived from G4VPrimaryGenerator.
  - G4IAEApshpReader are thread-local objects.
- Object dynamically created in ActionInitialization::Build()
  - This was (for me!) the cleanest solution.
    - The file name has to be set prior object creation (no “empty” phsp reader is allowed).
- Each thread-local object defines a *source_id* defined internally in IAEA routines’. (MAX_NUM_SOURCES=30)

- Currently, up to 30 threads are admitted.
  - But each fstream is associated to a source_id (either **reading** or **writing** a phsp file!).
  - ➢ The limit is lower when using a sources “slot” to write a phsp file.

- Support for array-job of MT executions:
  - The fragment in file is limited accordingly.



Current status & next steps

- Currently working on **G4IAEAphspWriter** class.
  - Cleaning up code and redesigning to make it MT-compliant.
- To decide between these approaches:
  1. Consider the G4IAEAphspWriter objects as **thread local**.
     - Each thread produces a file, merge them at end of run.
       - Remember that each G4IAEAphspWriter object will ‘eat’ one source_id.
  2. Consider a **shared** G4IAEAphspWriter object.
     - Particles dumped in bunches from worker threads to G4IAEAphspWriter.
     - G4IAEAphspWriter object receives ***N*** particles to dump it into the phsp file.
     - The bunch size should be set by user. Optimal value of ***N*** depends on use case.

This work receives funding from Grant PID2021-123879OB-C21 funded by MCIN/AEI/10.13039/501100011033 and by "ERDF A way of making Europe", European Union.

@ miancortes@us.es