Joint ICTP-IAEA Workshop on Monte Carlo Radiation Transport and Associated Data Needs for Medical Applications

28 October – 8 November 2024

ICTP, Trieste, Italy

Lecture 23

egs++ particle sources

Reid Townson

Metrology Research Centre
National Research Council Canada

Particle sources

Particle sources within a Monte Carlo code provide the following initial properties for particles:

- position $\vec{x}$
- direction $\vec{u}$
- energy E
- charge q
- statistical weight w
- latch variable
- time index

Some of these variables must be sampled according to a probability distribution, e.g., to reproduce the energy distribution of incident particles or their position in space for an extended source.

The egs++ particle sources

- EGSnrc C++ source classes deliver the same functionality as all other EGSnrc source functions (e.g., sources for RZ applications), but offer much more flexibility.
- Any egs++ source class provides a central function `getNextParticle()`
  - samples particle position, direction, E, q etc.
  - returns the number of independent particles generated so far
- The `getNextParticle()` function is called within every egs++ application

```
int EGS_Application::simulateSingleShower() {
  //...
  current_case =
   source->getNextParticle(rndm,p.q,p.latch,p.E,p.wt,p.x,p.u);
  //...
}
```

Common source input syntax

- The source definition is placed in the .egsinp file, between delimiters:

```
:start source definition:
    ...
:stop source definition:
```

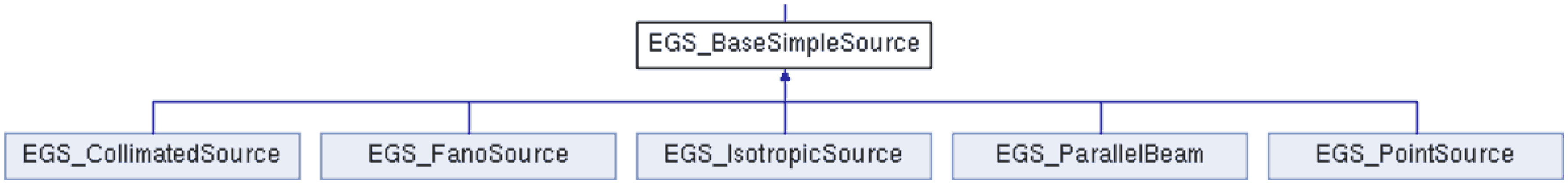
- A source object is defined using

```
:start source:
    name    = name_of_source
    library = source_library_name
    ( other source-specific input ...)
:stop source:
```

- The final simulation source can be made up of several sources

- The source definition section must contain the following key to select the simulation source: `simulation source = name_of_source`

The EGS_BaseSimpleSource class



- Simple sources share the following properties:
  - deliver a fixed charge,
  - sample energy from a distribution using a spectrum object
  - use method `getPositionDirection()`: delivering $\vec{x}, \vec{u}, w$
- Different types (parallel beam, collimated source, etc.) are derived from `EGS_BaseSimpleSource` and implement own `getPositionDirection()` functions $\Rightarrow$ definition in input files is source-specific

Source energy spectrum

Source shape

Source type

Source energy definition: Basics

Definition of the energy distribution is common to all and must be contained within

```
:start spectrum:
    .
    .
:stop spectrum:
```

The particle's energy can be either monoenergetic or sampled from an energy distribution (spectrum).

- Monoenergetic spectrum with a single discrete energy

```
:start spectrum:
    type = monoenergetic
    energy = the kinetic energy in MeV
:stop spectrum:
```

Source energy definition: polyenergetic sources

- Uniform distribution within defined energy boundaries

```
type = uniform
range = minimum and maximum energy
or
minimum energy = Emin
maximum energy = Emax
```

- Tabulated spectrum: energies and probabilities in a file (e.g. $HEN HOUSE/spectra)

```
type = tabulated spectrum
spectrum file = absolute path to a spectrum file
```

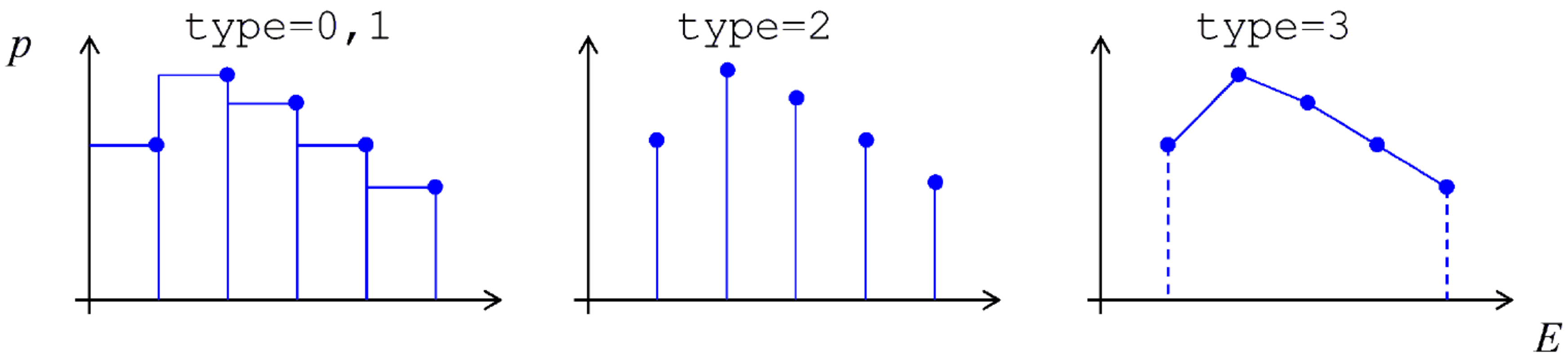
or inline with (Note: probabilities do not need to be normalized!):

```
type = tabulated spectrum
energies = list of discrete energies
           or bin edges
probabilities = list of probabilities
spectrum type = 0 or 1 or 2 or 3
```

Tabulated spectrum types

`spectrum type` (or MODE within the file) defines one of four possible interpretations:

- Histogram spectrum (type = 0,1): series of energy bins with different uniform probabilities, considered as cts per bin (type = 0) or cts per MeV (type = 1)
- Line spectrum (type = 2): series of discrete energies with corresponding probabil- ities
- Interpolated spectrum (type = 3): probabilities considered to be at bin edges and linear interpolation between bin edges

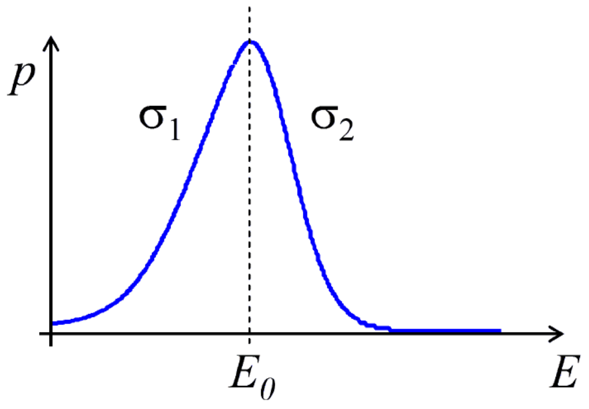


Source energy definition: Gaussian sources

- Gaussian spectrum with energies sampled from a normal distribution

```
:start spectrum:
     type = Gaussian
     energy = the mean kinetic energy in MeV
     sigma = the sigma of the spectrum
      or
     fwhm = the full-width-at-half-maximum of the spectrum
:stop spectrum:
```

- Double Gaussian spectrum: with two values for σ /fwhm, defining shape for energies less/larger than mean, joint smoothly at the mean energy.



Source energy spectrum

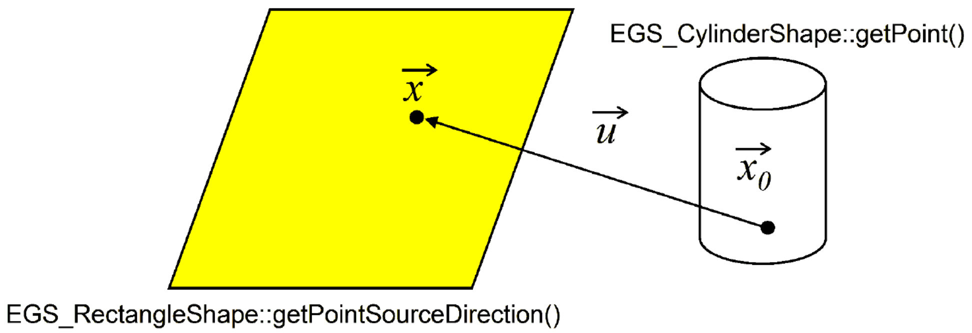
Source shape

Source type

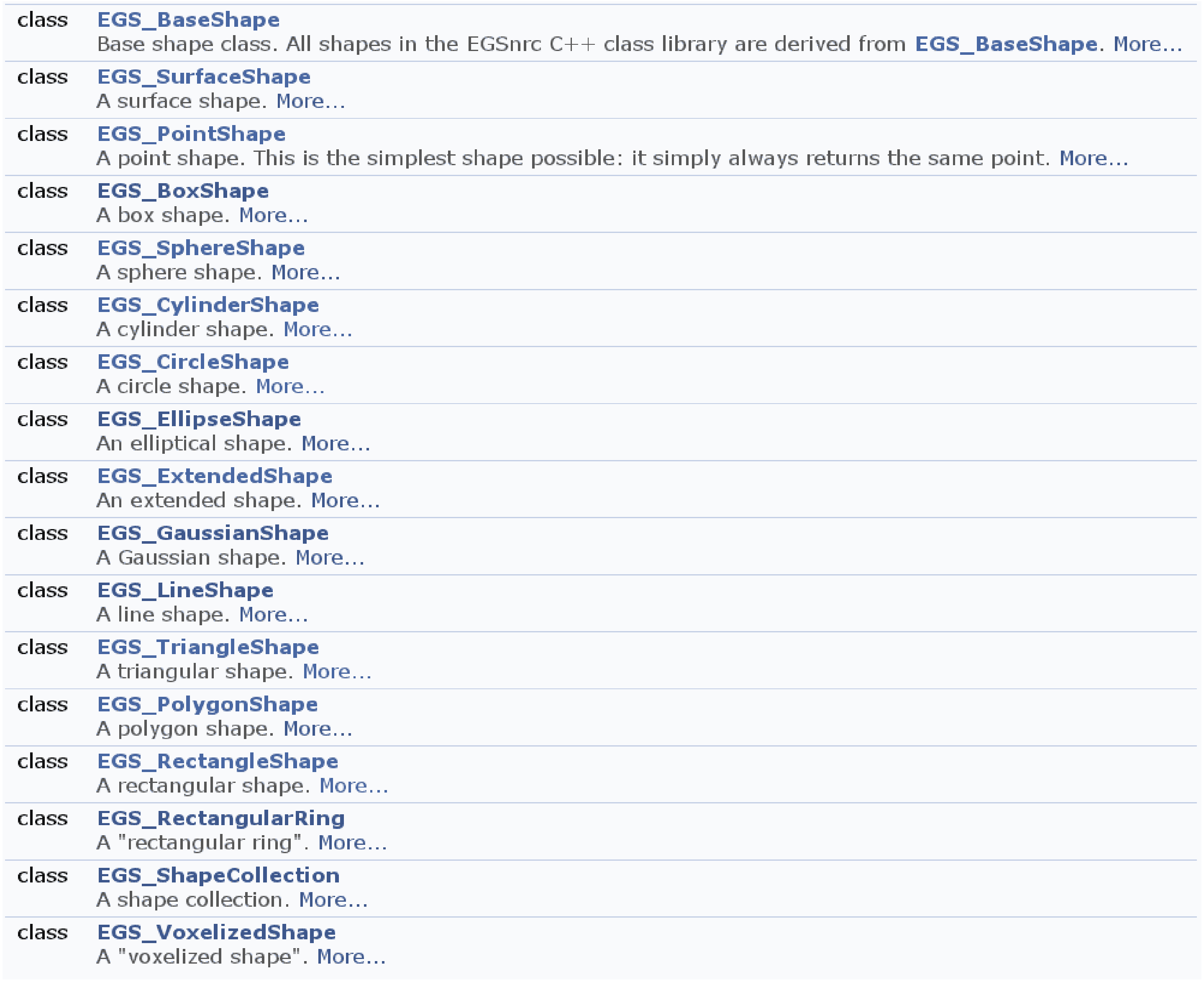
EGS_Shapes

- Sources derived from EGS_BaseSimpleSource require some method for the determination of direction and position of a particle:
  - position $\vec{x}_0$ is fixed or picked randomly within a certain region in space
  - direction $\vec{u}$ can be isotropic around $\vec{x}_0$, have fixed direction or aim to another point $\vec{x}$, thus $\vec{u} = (\vec{x} - \vec{x}_0) \, / \, |\vec{x} - \vec{x}_0|$

- EGS_shapes
  - deliver random positions in space
  - internal function getPoint() samples and returns $\vec{x}$
  - surface shapes deliver $\vec{u}$ with the getPointSourceDirection() function, called with $\vec{x}_0$


EGS_CylinderShape::getPoint()
$\vec{x}$
$\vec{u}$
$\vec{x}_0$
EGS_RectangleShape::getPointSourceDirection()

EGS_Shapes types



class	**EGS_BaseShape** Base shape class. All shapes in the EGSnrc C++ class library are derived from **EGS_BaseShape**. More...
class	**EGS_SurfaceShape** A surface shape. More...
class	**EGS_PointShape** A point shape. This is the simplest shape possible: it simply always returns the same point. More...
class	**EGS_BoxShape** A box shape. More...
class	**EGS_SphereShape** A sphere shape. More...
class	**EGS_CylinderShape** A cylinder shape. More...
class	**EGS_CircleShape** A circle shape. More...
class	**EGS_EllipseShape** An elliptical shape. More...
class	**EGS_ExtendedShape** An extended shape. More...
class	**EGS_GaussianShape** A Gaussian shape. More...
class	**EGS_LineShape** A line shape. More...
class	**EGS_TriangleShape** A triangular shape. More...
class	**EGS_PolygonShape** A polygon shape. More...
class	**EGS_RectangleShape** A rectangular shape. More...
class	**EGS_RectangularRing** A "rectangular ring". More...
class	**EGS_ShapeCollection** A shape collection. More...
class	**EGS_VoxelizedShape** A "voxelized shape". More...

EGS_Shapes specifics

- All have their own specific key values (see PIRS898).
- Some shapes are available as dso/dll and require the library key, others are directly compiled into egspp and defined with the type key.
- examples:

```
:start shape:
     type = point
     position = px, py, pz
:stop shape:
```

or

```
:start shape:
     library = egs_rectangle
     # left-upper, right-lower corners
     rectangle = x1 y1 x2 y2
:stop shape:
```

EGS_Shapes are not physical materials

- All shapes are placed at the origin by default and might be translated/rotated with the proper transformation.
- The shapes just define the position sampling space for sources (2D or 3D) and do not have any physical manifestation in the geometry
- In other words, shapes do not have a material assigned!
- Source shapes can overlap geometries and are completely independent

Source energy spectrum

Source shape

Source type

EGS_PointSource

The simplest type of source ...

```
:start source:
     library = egs_point_source
     name = some_name
     position = Px, Py, Pz
     :start spectrum:
           definition of the spectrum
     :stop spectrum:
     charge = -1 or 0 or 1 (electrons, photons, positrons)
:stop source:
```

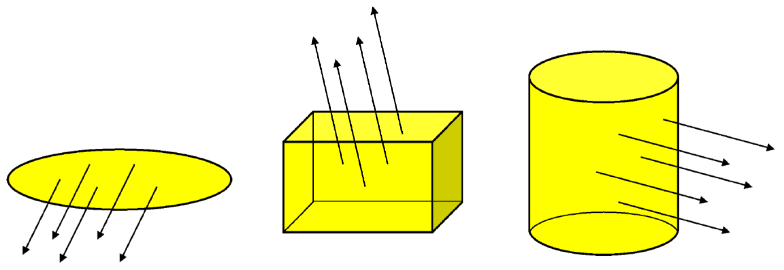
EGS_PointSource: Example

```
:start source definition:
    :start source:
        library      = egs_point_source
        name         = my_source
        position     = 0 0 0
        charge       = 0

        :start spectrum:
            type     = monoenergetic
            energy   = 1
        :stop spectrum:
    :stop source:

    simulation source = my_source

:stop source definition:
```

EGS_ParallelBeam

- EGS_ParallelBeam delivers particles, all having same direction with random positions within *any* shape
- Usually planar shapes are used, but 3D-shapes are possible.

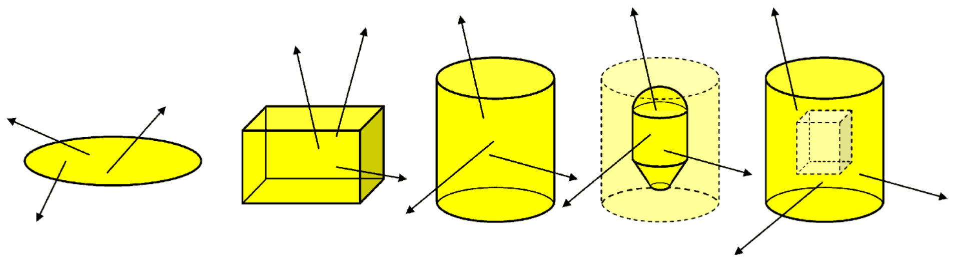
```
:start source:
   library=egs_parallel_beam
   name    = some_name
   :start shape:
        the shape
   :stop shape:
   :start spectrum:
        the spectrum
   :stop spectrum:
   direction = Ux Uy Uz
   charge = -1 or 0 or 1
:stop source:
```



- Functionality of SOURCE NUMBER=0,2,10,13 from the RZ applications and isource=0,1 in DOSXYZnrc can be reproduced.

EGS_IsotropicSource

- `EGS_IsotropicSource` delivers particles with directions uniformly distributed in 4π emitted from any shape.
- In case of 3D-shape, particles are emitted from any point within shape.
- Further, you can use a previously defined egspp geometry (or some selected regions of it) for sampling particle positions
  - IncludeAll, ExcludeAll, IncludeSelected or ExcludeSelected
    $\Rightarrow$ very complex isotropic sources shapes can be realized!
- Remember, the source shapes do not exist in the geometry and have no material.
- Source can reproduce the functionality of the SOURCE NUMBER=3 of the RZ-codes and isource=6 of DOSXYZnrc.

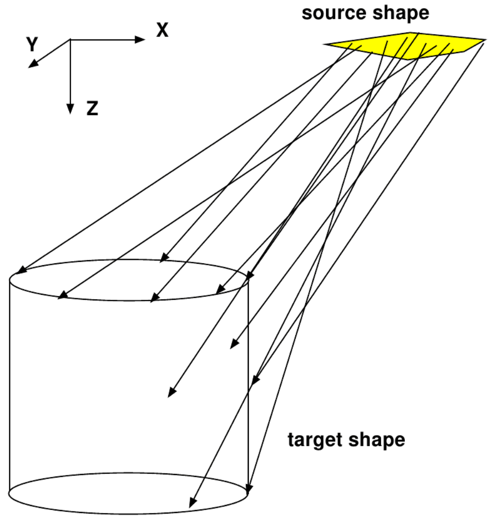


EGS_IsotropicSource: input definition

```
:start source:
    library = egs_isotropic_source
    name = some_name
    :start shape:
         definition of the shape
    :stop shape:
    :start spectrum:
         definition of the spectrum
    :stop spectrum:

    # Only emit if sampled within this geometry
    geometry = some_geometry
    # Specify some regions to include or exclude
    region selection = IncludeAll or ExcludeAll or
                  IncludeSelected or ExcludeSelected

    charge = -1 or 0 or 1 #for electrons, photons, positrons
:stop source:
```

EGS_CollimatedSource

- EGS_CollimatedSource is an isotropic source, collimated to irradiate only a certain area or solid angle.



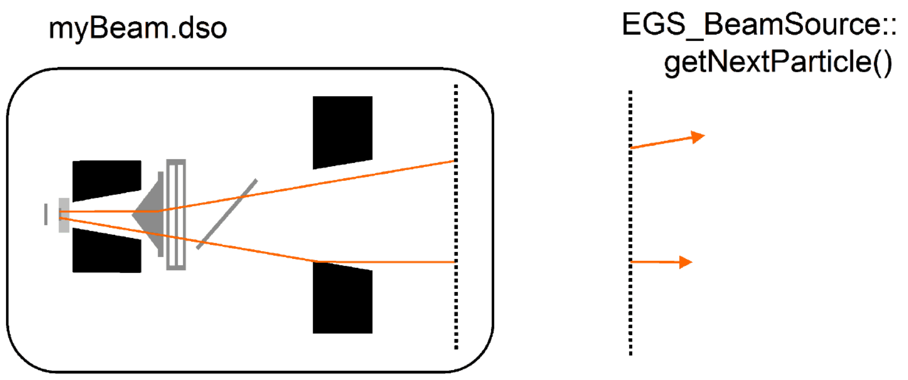
- Internally, EGS_CollimatedSource calls getPoint() function of a “source shape” and getPointSourceDirection() of a “target shape” $\Rightarrow$ only shapes supporting this function can be used as targets.

EGS_CollimatedSource: example

```
:start source:
    name          = the_zero_collimated_source
    library       = egs_collimated_source
    distance      = 100
    charge        = 0
    :start source shape:
        type      = point
        position = 0, 0, -100
    :stop source shape:
    :start target shape:
        library    = egs_rectangle
        rectangle = -15 -15 15 15
        # (in z=0 plane; use affine transformation to change)
    :stop target shape:
    :start spectrum:
        type   = monoenergetic
        energy = 0.06
    :stop spectrum:
:stop source:
```

EGS_BeamSource

- `EGS_BeamSource` uses a `BEAMnrc` application, pre-compiled into shared library (dso/dll).
- Particles are extracted and used as source particles upon crossing a defined scoring-plane within `BEAMnrc`.
- When starting the simulation a complete `BEAMnrc` simulation is initialized.
- Internally, a container provides single particles of a primary history. When the container is empty, a new shower within `BEAMnrc` is called, filling the container.
- `getNextParticle()` function of `EGS_BeamSource` takes single particles from container.



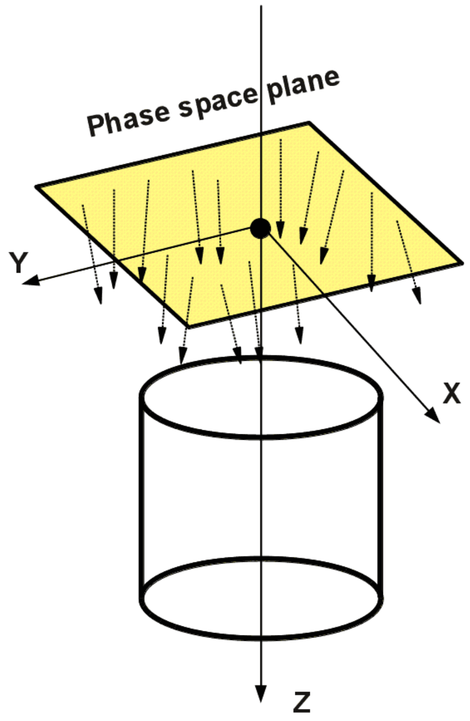
EGS_BeamSource: definition input

```
:start source:
    library = egs_beam_source
    beam code = BEAMnrc application name
    pegs file = PEGS file name for BEAMnrc simulation
    input file = BEAMnrc input file name
    cutout = x1 x2 y1 y2 (optional)
    particle type = 'all' or 'electrons' or 'photons' or
        'positrons' or 'charged' (optional)
    # use to reject "fat" particles
    weight window = wmin wmax (optional)
:stop source:
```

BEWARE: When restarting calculations using a BEAMnrc shared library source, make sure that the `restart` calculation option is defined in both the source and the application input files.

EGS_PhaseSpaceSource

A phase-space file source reads and delivers particles from a BEAMnrc phase-space file. Because the phase-space file only contains the x- and y- positions, the z-position is set to 0.



EGS_PhaseSpaceSource

A phase-space file source is defined as follows:

```
:start source:
    library = egs_phsp_source
    name = some_name
    phase space file = phsp file name
    particle type =  'all' or 'electrons' or 'photons' or
        'positrons' or 'charged' (optional)
    cutout = x1 y1 x2 y2  (optional)
    reuse photons = number to RECYCLE a photon
    reuse electrons = number to RECYCLE a charged particle
:stop source:
```

EGS_TransformedSource

A transformed source takes a particle from any other source and applies an `affine transformation` to the position and the rotation of to the direction.

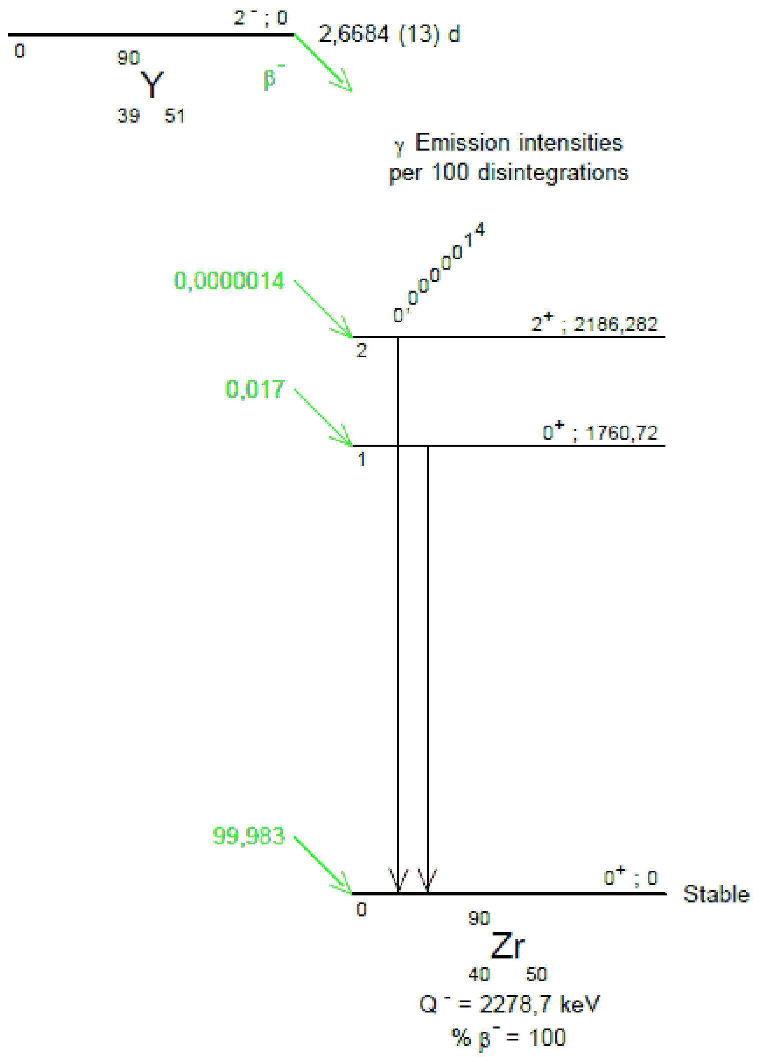
A transformed source is defined as follows:

```
:start source:
    library = egs_transformed source
    name = some_name
    source name = the name of a previously defined source
    :start transformation:
           translation = tx, ty, tz
           rotation = 2, 3 or 9 floating point numbers
             or
           rotation vector = 3 floating point numbers
    :stop transformation:
:stop source:
```

EGS_PhaseSpaceSource redux

- Note that a phase-space source can be used as the source in a transformed source permitting in this way arbitrary transformations to be applied to the particle positions and directions.
- It is also worth noting that, together with a transformation, the phase-space source can reproduce the functionality of any phase-space file based source in the RZ series of applications and in DOSXYZnrc.

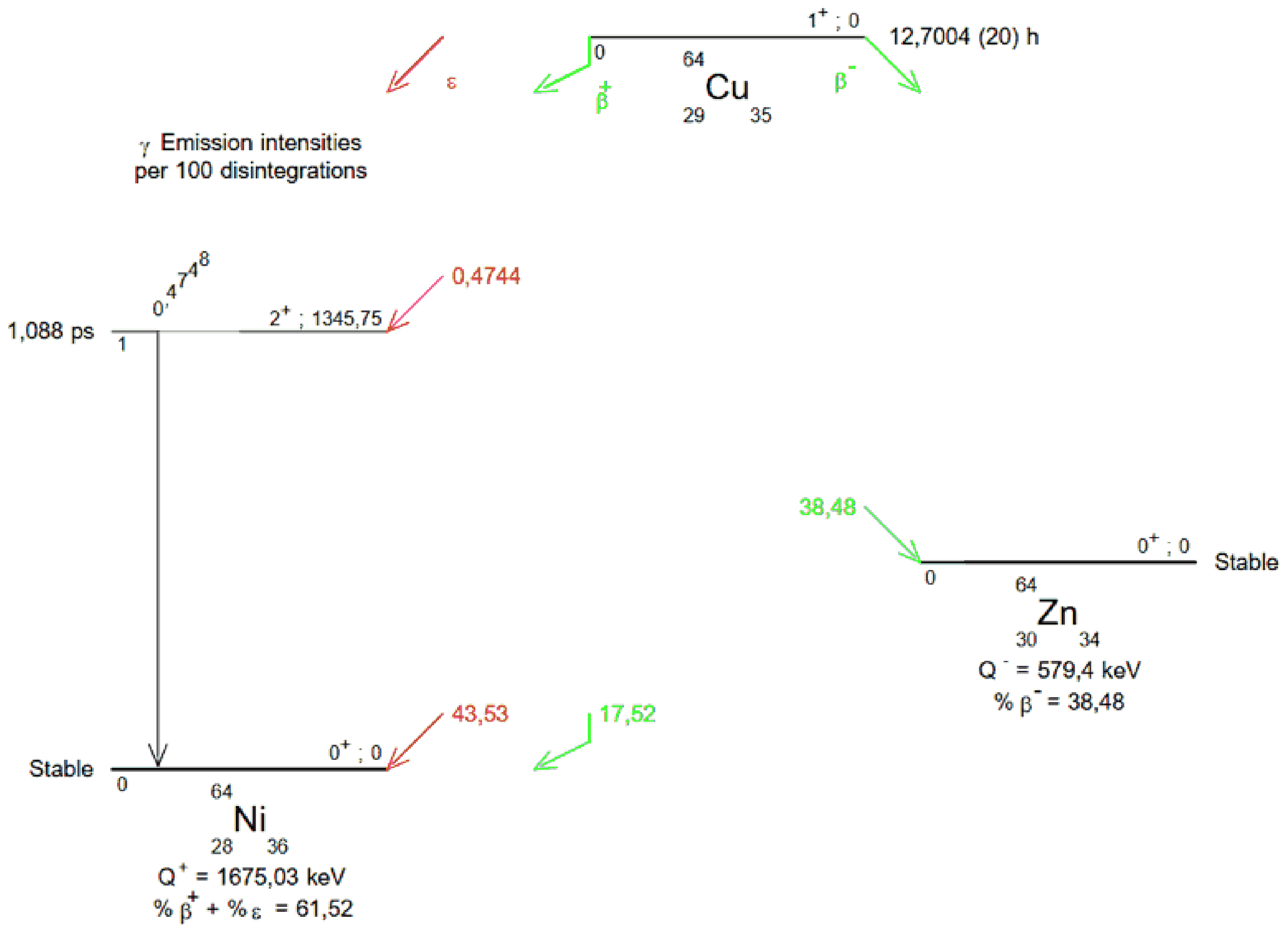
EGS_RadionuclideSource



- A new source of particles, following the decay scheme of a radionuclide.
- EGS_RadionuclideSource extends EGS_BaseSource, and allows for the correlated emissions of photons, electrons and positrons.

Nuclear data from LNHB: http://www.nucleide.org/DDEP_WG/DDEPdata.htm

EGS_RadionuclideSource (cont.)



EGS RadionuclideSource (cont.)

```
:start source:
    name                   = my_mixture
    library                = egs_radionuclide_source

    base source            = name of the source used to generate decay locations

    activity               = [optional, default=1] total activity of mixture,
                             assumed constant. The activity only affects the
                             emission times assigned to particles.
    charge                 = [optional] list including at least one of -1, 0, 1, 2
                             to include electrons, photons, positrons and alphas.
                             Filtering is applied to ALL emissions (including
                             relaxation particles).
                             Omit this option to include all charges - this is
                             recommended.
    experiment time        = [optional, default=0] time length of the experiment,
                             set to 0 for no time limit. Source particles generated
                             after the experiment time are not transported.

    :start spectrum:
        # See next slide...
    :stop spectrum:
```

EGS_RadionuclideSpectrum

```
    :start spectrum:
        type              = radionuclide
        nuclide           = name of the nuclide (e.g. Sr-90), used to look up the
                              ensdf file as $HEN_HOUSE/spectra/lnhb/ensdf/{nuclide}.ensdf
                              if ensdf file not provided below
        ensdf file        = [optional] path to a spectrum file in ensdf format,
                              including extension
        atomic relaxations = [optional, default=eadl] eadl, ensdf or off
                                      By default, 'eadl' relaxations use the EGSnrc
                                      algorithm for emission correlated with
                                      disintegration events. Alternatively, 'ensdf'
                                      relaxations statistically sample fluorescent
                                      photons and Auger emission using comments
                                      in the ensdf file. Turning this option off
                                      disables all relaxations resulting from
                                      radionuclide disintegration events.
        output beta spectra = [optional, default=no] yes or no
                                  whether or not to output beta spectra to files.
                                  Files will be named based on the nuclide and
                                  maximum energy of the beta decay:
                                  {nuclide}_{energy}.spec
        # See next slide...
```

EGS_RadionuclideSpectrum (cont.)

```
        alpha scoring       = [optional, default=none] none or local
                               Whether or not to deposit alpha particles locally.
                               Since alpha particles are not transported in EGSnrc,
                               there are only two options. Either discard the alpha
                               particles and their energy completely, or deposit
                               the energy immediately after creation in the
                               local region.
        extra transition approximation = [optional, default=on] on or off
                               If the intensity away from a level in a radionuclide
                               daughter is larger than the intensity feeding the
                               level (e.g. decays to that level), then additional
                               transitions away from that level will be sampled if
                               this approximation is on.
                               They will not be correlated with decays, but the
                               spectrum will produce emission rates to match both
                               the decay intensities and the internal transition
                               intensities from the ensdf file.
    :stop spectrum:
:stop source:
```

Source normalization

Sources provide a method for extracting the final fluence via the `getFluence()` method. However, the returned value depends on the source type:

- `EGS_IsotropicSource` returns the number of histories N
- `EGS_PointSource` returns the number of histories N
- `EGS_BeamSource` returns the number of histories N
- `EGS_PhspSource` returns $N_{\mathrm{read}}/N_{\mathrm{particles}} \times N_0$, where N_0 is the number of primary histories and $N_{\mathrm{particles}}$ number of particles in phsp file.
- `EGS_ParallelBeam` returns N/A, where A is the target shape area
- `EGS_CollimatedSource` returns N/d^2, where N is the number of particles hitting the target and d the distance defined by the user in the input file. If isotropic, you can set $d = 1/\sqrt{2\pi}$ to recover a normalization to the number of histories. *Note that d is set to 1 by default!*
- `EGS_RadionuclideSource` returns the number of disintegration events

It is important to be mindful of normalization when comparing results with other Monte Carlo simulation packages, or with experimental measurements!

... more sources

- EGS_SourceCollection consists of arbitrary number of any other sources with user-defined weights.

```
:start source:
    library = egs_source_collection
    name = some_name
    source names = names of previously defined sources
    weights = list of weights for the sources
:stop source:
```

- EGS_AngularSpread takes particle from other source and applies rotation to direction by angle sampled from a Gaussian distribution.

```
:start source:
    library = egs_angular_spread_source
    name = some_name
    source name = the name of a previously defined source
    sigma = positive: sigma in degrees
            negative: fwhm in degrees
:stop source:
```

Finally, if you think this is not enough...

$\Rightarrow$ implemented your own sources, deriving it from `EGS_BaseSource` or `EGS_BaseSimpleSource` class!

Browse the PIRS-898 html-documentation!

`http://nrc-cnrc.github.io/EGSnrc/doc/pirs898/index.html`