Gate 6.2 - User Fluence Image Source

F. Smekens

09 july 2013

The aim of the user fluence image (UFI) source is to generate primary particle positions according to a user-defined 2D probability distribution. This tool was initially developed for the simulation of the focal spot fluence of a low-energy X-ray tube.

1 Quick use

1.1 Macro

The UFI source uses the global particle source (gps). In the following example, a classical rectangle is replaced by the UFI source. When using this tool, the default shape is defined as a 'Rectangle' and the size is directly extracted from the image (section 1.2). Parameters related to the source orientation, particle direction and source center position are left free.

```
/gate/source/addSource mybeam gps
/gate/source/mybeam/gps/particle gamma
...
# Plane
#/gate/source/mybeam/gps/pos/type Plane
#/gate/source/mybeam/gps/pos/shape Rectangle
#/gate/source/mybeam/gps/pos/halfx 2.5 mm
#/gate/source/mybeam/gps/pos/halfy 2.5 mm

# UserFluenceImage
/gate/source/mybeam/gps/pos/type UserFluenceImage
/gate/source/mybeam/gps/pos/setImage data/reducedFocalSpot3D.mhd

/gate/source/mybeam/gps/ang/type focused
/gate/source/mybeam/gps/ang/focuspoint 0. 0. 0. mm
...
```

1.2 Description of the image

The UFI format must be supported by Gate (MetaImage, Analyse, etc.). The following elements of the header file have to be checked carefully :

1. even if we use a 2D information, Gate only supports **3D volumes** ;
2. the 'offset' defines the **center of the source in the image coordinates**. In practice, the 'offset' is defined as the translation for which the chosen pixel matches with the origin point of the image ;
3. **element spacing and dimensions** are automatically extracted from the image.

```
# header
ObjectType = Image
NDims = 3
...
Offset = -3.492 -1.188 0
...
ElementSpacing = 0.018 0.018 1
DimSize = 275 184 1
...
```

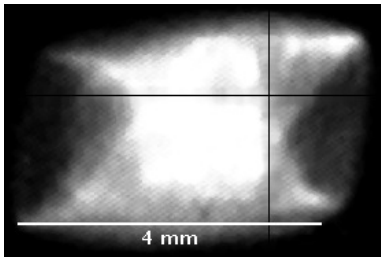

FIGURE 1 – UFI source. The black cross corresponds to the offset artificially introduced in the header file.

1.3 Example

An example of UFI source use can be found in the Gate repository. The reference image is represented in Fig. 1 and the dimensions were presented in the section 1.2. Note that the offset is set so that the green cross corresponds to the source center in the simulation. The source is placed at 80 mm from a detector plane of 8×8 mm^2 with a resolution of 200×200 pixels. The number of incident particle is set to 10^6.
The Fig. 2(a) shows the image of the source for a non-divergent source :

```
/gate/source/mybeam/gps/direction 0. 0. -1.
```

The Fig. 2(a) shows the image of the source for a focused source with a focus point equidistant from the source and the plane :

```
/gate/source/mybeam/gps/ang/type focused
/gate/source/mybeam/gps/ang/focuspoint 0. 0. 0. mm
```

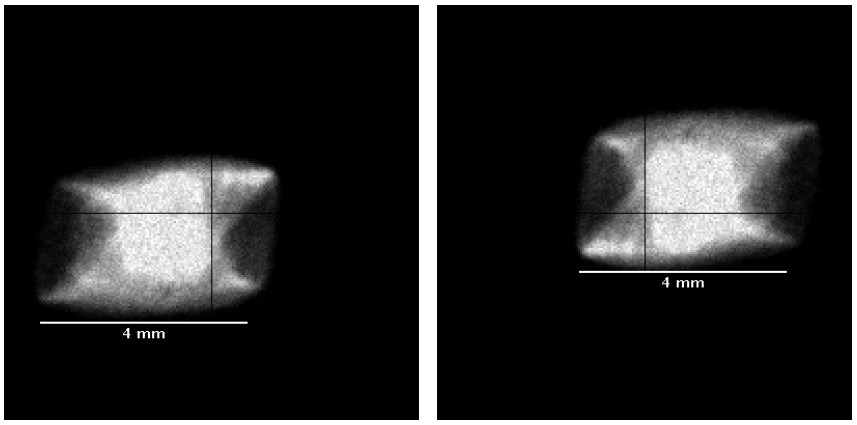

FIGURE 2 – UFI (Fig. 1) projected on a 8×8 mm^2 plane for a non-divergent source (a) and a focused source (b). The black cross corresponds to the center of image.

Both simulations give the expected result. For the non-divergent source, we get a simple copy of the UFI source and, for the focused source, the entire image is reversed. In both cases, the source is well centered.

2 Key points of the implementation

2.1 Description of the problem

2D biased distribution Geant4 already gives the possibility to generate 1D biased distributions (*G4SPSRandomGenerator* class). This tool is often used to include in the simulation a particular energy spectrum (see radiotherapy examples in Gate). The main problem of the UFI source is the necessity to create a 2D biased distribution, i.e. generate 'x' knowing 'y' or inversely, which do not exist in Geant4.

Gate source management The management of the source properties in Gate is quite simple. Actually, in the most generic cases, Gate directly calls the Geant4 functions whether to set parameters or to generate particles. This could be an advantage in many situations but, in our case, it is a real problem since a lot of steps concerning the primary particles creation can not be accessed. This includes the biased distribution generator of Geant4.

2.2 Implementation in *GateVSource*

The choice have been made to create a parallel position generator dedicated to the UFI system. The generator is composed of an initialization function (*InitializeUserFluence()*) and a position generator function (*UserFluencePosGenerateOne()*).

2D biased distribution In practice, a biased x-index position generator is created according to the cumulated fluence along the y-axis. For each index (i.e. pixel number along the x-axis), a specific biased index generator is created according to the fluence along the y-axis :

```
// Generate XBias and "YBias knowing X" according to fluence image
mUserPosGenY.resize(resX);
for(int i=0; i<resX;i++) {
  ...
  sum = 0.0;
  for(int j=0; j<resY; j++) {
    sum += userFluenceImage.GetValue(i,j,0);
    mUserPosGenY[i].SetYBias(G4ThreeVector(j,userFluenceImage.GetValue(i,j,0),0.));
    ...
  }
  mUserPosGenX.SetXBias(G4ThreeVector(i,sum,0.));
  ...
}
```

Note : *'mUserPosGenX/Y'* are *G4SPSRandomGenerator* objects. Notice that the use of a real 2D biased system is necessary since the bin number of a biased distribution is limited to 1024 in Geant4 code. **Therefore, the user is limited to a 1024 $\times$ 1024 image size.**

Gate source management As our 2D position generator can not directly replace the Geant4 one, we use the following trick. When a new event have to be processed, the 2D position generator is called. The random position is then set as source center of the **Geant4 official source which was artificially transformed into a point.** The two main advantages of the point type source are (1) the simplicity of steps for generating a position and (2) the non-attendance of any knowledge about the position for generating an angle (rotation matrix, translation, unit vectors, etc.).

```
# UserFluencePosGenerateOne
int i = floor(mUserPosGenX.GenRandX());
int j = floor(mUserPosGenY[i].GenRandY());

// uniform rand in pixel
double x = mUserPosX[i] + (G4UniformRand()-0.5)*mUserFluenceVoxelSize.x();
double y = mUserPosY[j] + (G4UniformRand()-0.5)*mUserFluenceVoxelSize.y();
double z = 0.;
...
// Moving the source (point) and call GenerateOne function
m_posSPS->SetCentreCoords(randPos);
```

However, since the UFI is a plane-like generator, the generator has to take into account the rotation and translation of the plane.

3 Files and functions

Modifications concerning the UFI source have been made in the following classes and functions :

- **GateVSource** (source/physics/...)

  ```
  void GateVSource::GeneratePrimaryVertex(G4Event* aEvent)
  void GateVSource::InitializeUserFluence()  (NEW)
  G4ThreeVector GateVSource::UserFluencePosGenerateOne()  (NEW)
  void GateVSource::SetCentreCoords(G4ThreeVector coordsOfCentre)  (NEW)
  void GateVSource::SetPosRot1(G4ThreeVector posrot1)  (NEW)
  void GateVSource::SetPosRot2(G4ThreeVector posrot2)  (NEW)
  ```

- **GateSingleParticleMessenger** (source/physics/...)

  ```
  "pos/setImage"  (NEW)
  "pos/type"
  "pos/centre"
  "pos/rot1"
  "pos/rot2"
  "centre"
  "posrot1"
  "posrot2"
  ```