Manual GATE para simulación de mamografía

David Jurado
Natalia Copete

Junio, 2021

Índice

1. Introducción

GATE es un software de código abierto desarrollado por la colaboración internacional OpenGATE dedicado al desarrollo de simulaciones en el campo de física médica. Este software está diseñado para casos de estudio como las técnicas PET, SPECT, CT y rayos X para toma de imágenes médicas. La base de este software es el uso de la herramienta Geant4 para física de altas energías y su arquitectura hace uso de macros basados en Geant4 y C++, lo que incorpora varias funciones pertinentes que abarcan interacciónes luz-materia y materia-materia. Esto le permite a GATE tener un acercamiento preciso a los casos reales por medio de modelos físicos y simulaciones Monte-Carlo ya implementadas.

Debido a la naturaleza del software que usa principalmente *scripts* en su estructura, programar una simulación en GATE no requiere de conocimientos previos en Geant4 o C++. Sin embargo, su arquitectura linux requiere del usuario competencias básicas en este sistema operativo. Es recomendable además que el usuario tenga conocimientos previos de programación en python para análisis de datos.

A pesar de que GATE es un simulador de Monte-Carlo con distintas aplicaciones en física médica, este manual se va a centrar en el caso de estudio de toma de imágenes médicas de cáncer de seno por medio de mamogramas convencionales. Sin embargo, muchas de las funciones explicadas en este manual son aplicables para otro tipo de simulaciones médicas.

2. Instalar GATE

Dentro de las posibilidades que se tiene en el Laboratorio de Altas Energías, GATE se puede utilizar de dos formas distintas. Por un lado, se puede recurrir a usar los servidores del laboratorio, en los que ya está instalado GATE en su octava versión. Por otro lado, se puede instalar la máquina virtual vGate en su computador personal. La mejor forma de familiarizarse con GATE es precisamente esta última por tres razones principales. Primero, en la máquina virtual usted tiene acceso a herramientas de visualización que no están disponibles en el servidor. Segundo, en vGate se encuentran directamente, en los archivos de la máquina virtual, diferentes ejemplos de utilización de GATE. Tercero, usted no requiere tener conocimientos sobre cómo conectarse a un servidor. Ahora bien, lo único que se requiere para usar vGate en su computador es su instalación, un proceso que es rápido y sencillo. En el siguiente enlace se encuentran descritos los pasos para realizarla: https://opengate.readthedocs.io/en/latest/vgate.html. Es importante tener en cuenta que, tal y cómo se menciona en dicho enlace, el primer paso de todo el proceso debe ser la instalación de Virtual Box, el *software* en el que se ha construido la máquina virtual:

https://www.virtualbox.org/wiki/Downloads.

Dentro de los últimos pasos del proceso de instalación, es necesario destinar cierta porción de la memoria de su equipo personal a la máquina virtual. Una pregunta natural que surge es ¿Qué porción asignarle? La respuesta depende del uso que usted le vaya a dar a su equipo anfitrión. Sin embargo, una configuración recomendable para nuevos usuarios es la de disponer de la mitad de la memoria disponible [1].

Llegados a este punto, es importante resaltar que la capacidad computacional con la que van a correr sus simulaciones está limitada por la capacidad de su computador personal y de cuánta memoria le asignó a vGate. Si observa que las simulaciones que quiere realizar resultan demasiado pesadas para su computador, considere utilizar los servidores del laboratorio. Una buena estrategia es la de construir los *scripts* y hacer un pequeño cálculo de prueba para ver si todo funciona bien en vGate, para luego correr la simulación pesada en los servidores.

3. Sintaxis de GATE

En GATE, cada acción que se realiza está asociada a un comando. Los comandos tienen estructura de árbol y pueden tomar uno o más parámetros. La estructura típica de dichos comandos es

```
/gate/X/Y/Z    parámetros
```

donde X, Y y Z son palabras que van a determinar, de forma inequívoca, el aspecto de la simulación sobre el cual se está ejerciendo la acción que se quiere. Por ejemplo, si X= fantoma; Y= geometry; Z= setHeight y parámetros=20 cm, se está definiendo la altura del fantoma en 20 cm.

Los comandos se pueden ejecutar en modo interactivo o bajo la construcción de *scripts*, llamados también *macros*, en los cuales se almacena la sucesión de comandos que se quieren llevar a cabo para realizar la simulación. El modo interactivo puede resultar útil para construir la geometría. Sin embargo, resulta más organizado y eficiente a largo plazo utilizar los *macros*.

En principio, una simulación se puede correr usando un único *macro*. Sin embargo, lo mejor es utilizar varios de ellos, uno para cada uno de los aspectos importantes de la simulación, como por ejemplo, la fuente, los procesos físicos, el fantoma, etc. Cada uno de estos *macros* "específicos" va a ser llamado por un *macro "master"*, el archivo que se va a ejecutar al correr la simulación y al cual nos referiremos de ahora en adelante como *master*. Cabe mencionar que para ejecutar un *macro* desde otro (en particular desde el *master*) se debe escribir en este ultimo el siguiente comando

```
/control/execute mimacro.mac
```

Es importante resaltar que un modelo de estructura de *macros* recomendable para aplicaciones que involucre rayos X es el del ejemplo *CT simulation classic* publicado en el GitHub de la colaboración OpenGATE.

Llegados a este punto, vale la pena mencionar que una vez construidos los diferentes *macros*, existen tres formas de ejecutar el *macro "master"*. La primera, es directamente desde la linea de comando en Linux:

```
Gate mimacromaster.mac
```

La segunda, es desde el ambiente GATE (ver sección 4), justo después de Idle>:

```
Idle>/control/execute mimacromaster.mac
```

La tercera es en modo QT. De esta se hablará en la sección 4. Estas tres formas de ejecutar GATE se pueden utilizar en la maquina virtual. Sin embargo, en el servidor del Laboratorio de Altas Energías el modo QT no se puede utilizar.

En relación con la ejecución de GATE en el servidor, resulta útil saber que se pueden ejecutar varias simulaciones al mismo tiempo. Para esto, se debe utilizar el comando *screen*. El manejo de este último se puede resumir en los siguientes pasos:

1. Ingresar al servidor y verificar que para cada *job* que se va a correr el archivo de salida tenga un nombre distinto. Lo anterior, para que un sólo archivo de salida no se sobrescriba con las diferentes simulaciones.
2. Escribir el comando *screen*. A partir de este, se va a generar una nueva ventana. Dentro de dicha ventana ejecutar GATE normalmente, tal y como se describió en párrafos anteriores.
3. Una vez el código comience a correr, es posible salirse de la ventana ejecutando ctrl+A+D.
4. Repetir el proceso anterior tantas veces como *jobs* necesite ejecutar.

Ahora, para mirar cuántos *jobs* están corriendo se debe escribir en la línea de comandos

```
screen -ls
```

A partir de lo anterior, va a salir una lista con todas las ventanas en las que se está ejecutando un *job*; cada una de ellas tiene asociada una identificación de 5 dígitos, por ejemplo, 70996. Para volver a entrar a cualquier ventana, por ejemplo, aquella asociada a este último número, escribir *screen -r* seguido de su identificación:

```
screen -r 70996
```

Finalmente, para salir definitivamente de una ventana, escribir *exit* en la linea de comando.

4. Primeros pasos

Para entrar en el entorno GATE desde una máquina virtual, basta con introducir "Gate" en la consola ubicándose en un directorio con permisos de lectura y escritura, al entrar en el entorno, se pueden introducir líneas de código después de *Idle*> permitiendo así definir objetos y características de la simulación, además de comandos de visualización como el modo QT. Sin embargo, esta no es una forma eficiente de iniciar una simulación, y al igual que otros lenguajes de programación como *Python* o *C++* que usan archivos compilables para manejar múltiples líneas de código (como son el formato *.py* y *.cpp* respectivamente) en este caso usaremos archivos macro *.mac* para ejecutar las múltiples líneas de código como se mostró en la sección anterior. El entorno GATE ejecuta estos archivos en el mismo orden de entrada, por lo que es necesario definir una estructura para no tener errores en los procedimientos de la simulación. Además de contener funciones de GATE, estos archivos macro pueden tener sub archivos macro que sean específicos a cada elemento de la simulación, por ejemplo un archivo específico para las condiciones de la fuente *source.mac* o bien para la definición del detector *detector.mac*. A continuación se muestra un ejemplo de la estructura que puede tener la simulación

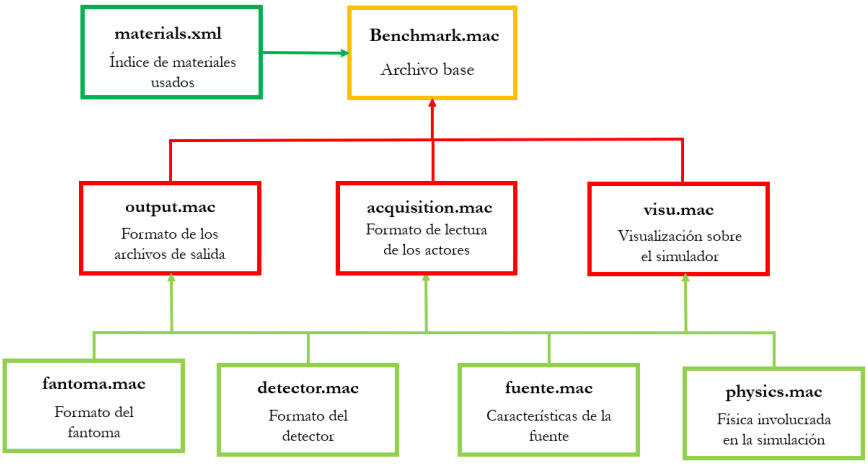

Figura 1: Ejemplo de jerarquía que deben seguir los archivos en la simulación, donde en la parte superior se encuentran los archivos que se definen primero y contienen los archivos en niveles inferiores. El archivo *Benchmark* es el archivo *macro* principal (*master*), el cual contiene todos los demás archivos en un orden apropiado para su ejecución, los archivos de la parte inferior definen geometrías y configuraciones de los objetos, así como características de la simulación como los fenómenos físicos involucrados

Esta estructura puede variar para distintas aplicaciones y no es única, dado que se pueden añadir o remover archivos. Sin embargo, es recomendable que la simulación tenga un número adecuado de archivos tanto para que su manejo no se vuelva tedioso, como para mantener un orden de compilación adecuado y evitar errores dentro de la semántica del código.

Una forma de estructurar la simulación y visualizar el procedimiento mientras se programa es el modo *qt* de GATE, este se ejecuta de la misma forma en que abrimos el entorno, pero añadiendo el parámetro de la forma

```
1 Idle > Gate --qt
```

Esto nos muestra otra ventana con las siguientes funciones

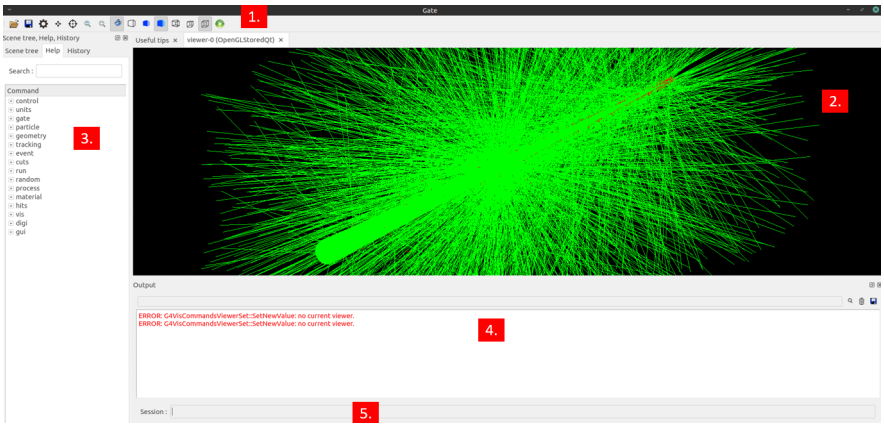

Figura 2: Interfaz del modo QT, 1. Barra de herramientas de visualización, 2. Visualización de la simulación en tiempo real, 3. Barra de comandos, 4. Consola, 5. Cuadro de input del usuario.

Este modo es de utilidad al momento de modelar los objetos en el mundo, ya que nos da una descripción geométrica rápida del mundo. Sin embargo, al momento de hacer simulaciones con muchos eventos, es recomendable usar la ejecución por shell unicamente, ya que parte de la memoria se invierte en la visualización de los eventos. Es necesario además aclarar que los objetos son visibles únicamente si se definen los parámetros de visualización para cada estos, es decir, su dibujo y su color

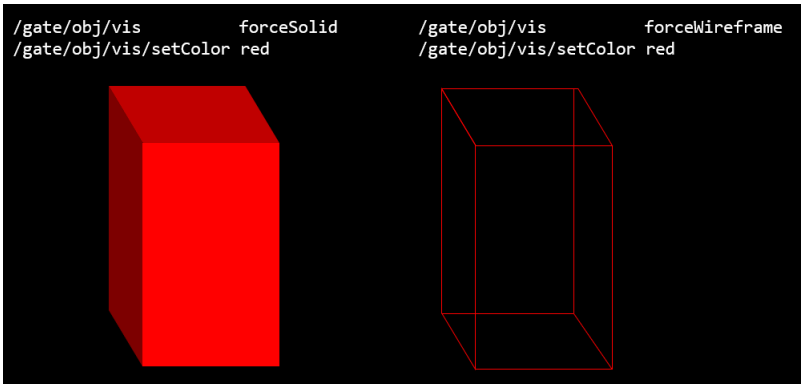

Figura 3: Formas en que GATE dibuja los objetos sobre la visualización, cabe además mencionar que el modo *forceSolid* no presenta transparencia a menos de que se defina un *alpha*, por lo que elementos al interior del objeto no son visibles en el display, para más información, es recomendable leer la sección *visualization* en el manual oficial de GATE.

5. Definir volúmenes y geometrías

La definición de volúmenes es importante para dos aspectos de una simulación mamográfica: el fantoma y el detector. Sin embargo, antes

de definir cualquiera de ellos es necesario declarar el volumen que los contiene a todos, el *world*. La geometría del *world* es rectangular centrada en el origen y sus dimensiones deben ser fijadas en el *master*. Para definir el volumen del fantoma o del detector es necesario establecerlos como "hijos" del *world*. Y, cualquier volumen nuevo que se quiera incluir dentro de estos debe ser establecido como "hijo". La forma de establecer "hijos" con respecto a sus volúmenes "madre", así como los comandos para definir las dimensiones, el material y la posición de los diferentes volúmenes, se encuentran bien explicados en la sección *Defining a geometry* del manual oficial de GATE en su novena versión. Sin embargo, vale la pena resaltar dos cosas. Primero, que las coordenadas de los volúmenes hijo siempre se definen con respecto al centro del volumen madre. Segundo, el material que se establezca para un volumen determinado debe encontrarse declarado en el archivo de materiales de la simulación (ver sección 6).

Ahora bien, usualmente al construir el fantoma y el detector se hace necesario incluir un mismo volumen N veces. En dicho caso, no es necesario declarar el volumen N veces. En cambio, se puede hacer uso de los repetidores, comandos que permiten repetir un volumen determinado tantas veces como se quiera. Existen distintos tipos de repetidores, dependiendo de la simetría de la repetición. En la sección *Defining a geometry* del manual oficial de GATE se presentan de forma adecuada los diferentes tipos de repetidores y la sintaxis para definirlos. Sin embargo, resulta pertinente presentar un ejemplo adicional a los que se muestran en dicho manual. A continuación se presenta un ejemplo de código, en lenguaje GATE, en el que se declara el volumen del fantoma y en este se incluye el arreglo de cubos de la Figura 4.

```
/gate/world/daughters/name Fantoma
/gate/world/daughters/insert box
/gate/Fantoma/placement/setTranslation  0.0  0.0
     0.0 cm
/gate/Fantoma/geometry/setXLength 1.1 cm
/gate/Fantoma/geometry/setYLength 1.1 cm
/gate/Fantoma/geometry/setZLength 44 mm
/gate/Fantoma/setMaterial PMMA
/gate/Fantoma/daughters/name crystalOX
/gate/Fantoma/daughters/insert box
/gate/crystalOX/placement/setTranslation -2.94
    3.62  0.0  mm
/gate/crystalOX/geometry/setXLength 0.54 mm
/gate/crystalOX/geometry/setYLength 0.54 mm
/gate/crystalOX/geometry/setZLength 0.54 mm
/gate/crystalOX/setMaterial HA
/gate/crystalOX/repeaters/insert cubicArray
/gate/crystalOX/cubicArray/setRepeatNumberX 2
/gate/crystalOX/cubicArray/setRepeatNumberY 2
/gate/crystalOX/cubicArray/setRepeatNumberZ 1
/gate/crystalOX/cubicArray/setRepeatVector 1.080
     1.080 0.0  mm
```

Código 1: Código en lenguaje GATE para definir el fantoma y generar

el arreglo de cubos de la Figura 4

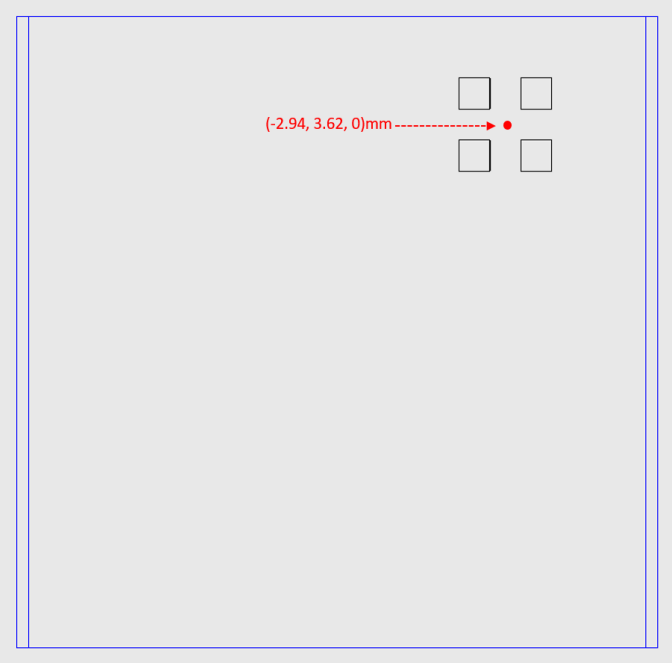

Figura 4: Fantoma y arreglo de cubos resultantes del código 1.

En las líneas 1 a 7 del código 1 se define el volumen *Fantoma* y se coloca el centro de este en las coordenadas (0,0,0)cm con respecto al *world*. Además, se establece el material del fantoma como PMMA, material que debe estar definido en la base de datos de materiales, como se mencionó anteriormente. Por otro lado, en las líneas 8 a 14 del código se define un volumen *crystalOX*, un cubo de lado 0.54 mm, el cual se sitúa en las coordenadas (-2.94, 3.62, 0)mm con respecto al centro del fantoma (el cual en este caso coincide centro del *world*). Finalmente, en las líneas 15 a 19 se realiza la repetición del volumen *crystalOX*. En la línea 15, se indica que la repetición se realiza mediante una geometría cúbica, cartesiana si se quiere. Las líneas 16 a 18 determinan cuántos volúmenes del tipo *crystalOX* se desea repetir en cada eje de coordenadas. Por su parte, la linea 19 indica, mediante un vector de repetición, la densidad de repetición del volumen en cada una de las direcciones cartesianas. Es importante tener en cuenta que, al no haber escrito ninguna opción de auto-centrado, en este caso se toma la opción por defecto, es decir, aquella en la que el arreglo de cubos se encuentra centrado en la posición incial del volumen *crystalOX*, tal y como se muestra en la Figura 4. Cabe mencionar que la otra opción de auto centrado disponible en GATE consiste en dejar el volumen inicial que se está repitiendo centrado en sus coordenadas iniciales; en este caso, (-2.94, 3.62, 0)mm (ver *Linear repeater*).

Finalmente, en la sección 10 se muestra otro ejemplo de utilización de

los repetidores.

6. Definir los archivos de materiales

Las propiedades de los materiales necesarias para que GATE defina el comportamiento de la radiación al interactuar con estos se deben encontrar en un archivo determinado, el cual va ser llamado en el *master*. En la sección *Materials* del manual de GATE se describe bastante bien cómo definir un nuevo material en dicho archivo, dependiendo de si se trata de un elemento, de un compuesto o de una mezcla. Ahora bien, uno de los recursos que entrega la colaboración OpenGATE es el archivo *GateMaterials.db* que contiene una cantidad considerablemente amplia de materiales ya predefinidos. Este archivo viene incluido en vGate. Sin embargo, también se puede encontrar en la carpeta *GateContrib/misc/* del Github de la colaboración OpenGATE.

Una pregunta natural al enfrentarse a la necesidad de definir un nuevo material es ¿dónde buscar las propiedades del material? Le podemos recomendar 3 lugares donde comenzar a buscar: 1) en el *Handbook of mineralogy* de la Sociedad Mineralógica de Estados Unidos [2]; 2) en la base de datos de los Institutos Nacionales de Salud de Estados Unidos (NIH, por sus siglas en inglés) [3]; o 3) en las bases de datos del Instituto Nacional de Estándares y Tecnología (NIST, por sus siglas en inglés) [4].

7. Definir la física del sistema

En GATE los procesos físicos tienen tres estados: "Disponible", "Habilitado" e "Inicializado". Por defecto, todos los procesos se encuentran únicamente en estado "Disponible". Para habilitar uno de estos, se debe utilizar el comando *addProcess* y se debe escoger uno de los modelos disponibles para dicho proceso. En la sección *Setting up the physics* del manual de GATE se encuentran los diferentes procesos físicos disponibles, así como los modelos que se les pueden asociar. En el caso de imágenes de rayos X, los procesos recomendables se encuentran especificados en el archivo *physics.mac* del ejemplo *CT simulation classic* publicado en el GitHub de la colaboración OpenGATE. De acuerdo con este último, los procesos físicos a incluir, junto con sus modelos específicos en lenguaje GATE escritos entre paréntesis, son:

1. Efecto fotoeléctrico (*StandardModel*):
   Los rayos X incidentes desplazan un electrón de una capa interna del átomo, de tal forma que dicho electrón es expulsado con una energía igual a la diferencia de energías entre el rayo X incidente y la energía de enlace del electrón. Luego, un electrón de una capa superior ocupa el puesto del electrón que ha sido expulsado, de tal forma que se libera un fotón con energía igual a la diferencia de energías entre las dos capas. Sin embargo, dicho fotón posee una energía tan baja, que es absorbido por el tejido [5].

2. Efecto Compton(*PenelopeModel*):
   Consiste en la transferencia de energía por parte de los rayos X incidentes a un electrón de una capa exterior del átomo. Dicha transferencia tiene dos consecuencias: el electrón en cuestión es expulsado del átomo y el fotón de rayos X incidente es desviado de su trayectoria inicial [5].

3. Esparcimiento Rayleigh (*PenelopeModel*):
   El campo eléctrico de la radiación incidente genera oscilaciones en todos los electrones del átomo, lo cual origina la emisión de radiación con la misma energía, pero con una pequeña desviación [6].

4. Bremsstrahlung (*StandardModel* e^-):
   Es un proceso mediante el cual las partículas cargadas en aceleración emiten radiación. En este caso, se trata de los electrones liberados mediante los efectos fotoeléctrico y Compton, los cuales son acelerados por los campos generados por los núcleos de los átomos [7].

5. Ionización electrónica (*StandardModel* e^-):
   Surge por la perdida de energía que sufre una partícula cargada (en este caso electrones resultantes de otros procesos) al pasar a través de la materia. Esto, mediante colisiones inelásticas con los electrones de los átomos del material. Dicha perdida de energía resulta en la ionización o en la excitación del átomo en cuestión [7].

6. Esparcimiento múltiple (*MSC model*):
   Hace referencia a los múltiples procesos de esparcimiento que se generan entre una partícula cargada que pasa a través de la materia y los núcleos que la conforman. Dichos esparcimientos tienen la característica de ser Coulómbicos y elásticos [7].

Finalmente, cabe mencionar que los procesos habilitados pasan a estar inicializados una vez se ejecuta el comando */gate/run/initialize*, el cual típicamente debe estar incluido en el *macro master*.

8. Definir una fuente

La fuente es otra parte esencial de la simulación la cual como se mostró en la estructura, se puede definir en un archivo a parte con sus características, donde podremos definir su geometría, su rango de energías o si es monocromática, e incluso podemos definir semillas de eventos para así estudiar casos conocidos. GATE permite usar diferentes partículas del modelo estándar como eventos que provienen de la fuente. Sin embargo, en este caso nos centraremos en fotones (γ) sobre el rango de energía de rayos X usados en radiología. Para definir el objeto, debemos primero definir una fuente vacía a la cual le añadiremos las propiedades correspondientes. Entre estas propiedades, es esencial definir la particula emitida, la energía de dicha partícula, su espectro de energía, la apertura de la fuente y su geometría. A continuación se mostrará un

ejemplo para definir una fuente convencional de mamograma.

```
/gate/source/addSource xraygun
/gate/source/verbose 0
/gate/source/xraygun/setActivity 6500 Bq
/gate/source/xraygun/gps/verbose 0
/gate/source/xraygun/gps/particle gamma
/gate/source/xraygun/gps/energytype UserSpectrum
/gate/source/xraygun/gps/setSpectrumFile
    InterpolationSpectrum.txt
/gate/source/xraygun/gps/emin 19.5 keV
/gate/source/xraygun/gps/emax 21 keV
/gate/source/xraygun/gps/type Plane
/gate/source/xraygun/gps/shape Rectangle
/gate/source/xraygun/gps/halfx 0.015 mm
/gate/source/xraygun/gps/halfy 0.015 mm
/gate/source/xraygun/gps/mintheta 0  deg
/gate/source/xraygun/gps/maxtheta  0.9 deg
/gate/source/xraygun/gps/centre 0. 0. -63.63 cm
/gate/source/xraygun/gps/angtype iso
/gate/source/list
```

Este código define un objeto fuente llamado *xraygun* como se muestra en la primera linea, luego de esto, se procede a definir el modo de "Verbose", que como se explica en el tutorial de GATE en la sección *verbosity*, es la forma en que el simulador reporta los eventos Monte-Carlo generados. Es recomendable mantener el modo en cero para un menor consumo de memoria siempre y cuando no se requiera.

La siguiente línea hace referencia a la partícula a utilizar que, como ya se mencionó, se va a utilizar fotones. En el caso donde se usen partículas que pueden decaer durante el proceso, es necesario además definir su tiempo de vida media. Las siguientes dos líneas definen el tipo de input del espectro de energía y los valores de este espectro. Este espectro nos da información de cómo vienen distribuídas las energías de las partículas emitidas por la fuente. El tipo de energía "UserSpectrum" le dice al simulador que el usuario va a introducir el espectro de energía de la fuente manualmente por medio de un archivo de texto o un archivo *.dat*. Para otros usos, hay espectros de energía pre definidos que no requieren un archivo de texto. Para mas información sobre el formato de los espectros de energía y los modos de espectro, refierase al tutorial de GATE en la sección *Defining the energy*, otra fuente que puede ser muy útil para obtener espectros de fuentes con ánodo de Tungsteno es el uso de la herramienta TASMICS, la cual genera espectros por medio de métodos Monte Carlo [8]. El laboratorio de altas energías dispone de un Excel para este espectro que puede ser solicitado para su uso.

Ahora, llegando a las propiedades geométricas de la fuente, tenemos la forma de emisión, la cual puede ser rectangular o radial. Esto nos dice la forma en que se dispersan las partículas emitidas. Y también tenemos las funciones "mintheta" y "maxtheta" que se refieren al ángulo de apertura de la fuente. Para efectos de una simulación sin artefactos,

es recomendable mantener esta apertura lo más baja posible, ya que aperturas considerables dan paso al efecto Heel encontrado en procedimientos de radiología. Finalmente, la función "Centre" define la ubicación de la fuente en el mundo. Ahora, lo último que falta es definir cuántas partículas salen de la fuente, a estos eventos Monte-Carlo se les denomina primaries y generalmente no se definen en el archivo fuente, sino en el archivo principal. Para simular alrededor de sesenta millones de partículas, introducimos la siguiente línea

```
/gate/application/setTotalNumberOfPrimaries
    60.4915e6
```

Cabe aclarar que la cantidad de "primaries" a simular determina en gran parte la calidad de los resultados sobre el detector (A mayor estadística, mejores resultados) Sin embargo, esta también afecta el tiempo de cómputo, por lo que es importante lograr balancear buena calidad en las simulaciones y tiempo de procesamiento. Se recomienda realizar simulaciones preliminares con pocos eventos (Alrededor de diez millones) mientras se configuran otras características de la simulación y las imágenes finales producirlas con la mayor cantidad de eventos posibles.

9. Definir un detector

Se pueden mencionar dos formas distintas de considerar el detector.Una, es mediante la definición de un sistema. La otra, es usando el *DoseActor*. La primera, permite definir un detector más realista puesto que admite la posibilidad de definir una serie de parámetros que buscan emular la cadena de procesamiento de señales en un detector. Para construir el detector de dicha forma, se deben seguir cinco pasos, los cuales están bastante bien ilustrados en los archivos *CTscanner.mac* y *digitizer.mac* del ejemplo *CT simulation classic*. El primero paso, es definir el volumen del detector con las dimensiones deseadas (ver Sección 5). El segundo, es definir el sistema (*Defining a system*) que se va a utilizar. En el caso de las imágenes mamográficas, y en general de las imágenes de rayos X, el sistema que resulta útil es *CTscanner*. Al definir este último, se deben declarar tres sub-volúmenes: un módulo, un *cluster* y un píxel. Este último, se debe repetir mediante la herramienta de repetición mencionada en le Sección 5. Esto, con base en el número de píxeles que se quiera tener en el detector. En un tercer paso, se deben unir (*attach*) los volúmenes módulo, *cluster* y píxel a los comandos **module**, **cluster_0** y **pixel_0**. Esta acción de unir dichos volúmenes a esos comandos es la que determina el tipo de sistema que se está utilizando. En cuarto lugar, se tienen que unir los volúmenes en los que se quiere detectar la información de las partículas a la herramienta llamada **sensitive detector** (*Attaching the sensitive detectors*). Dicha herramienta es la que permite almacenar la información relacionada con las interacciones entre la radiación y el detector. Típicamente, los volúmenes que se unen al **sensitive detector** son los píxeles, puesto que estos definen la

resolución con que se quiere almacenar información. Finalmente, en un quinto paso, se deben determinar los parámetros de la cadena de procesamiento de señales en un detector. Dichos parámetros pueden ser muy simples o bastante complejos, eso depende de usted. En la sección *Digitizer and readout parameters* se encuentra explicado todo lo relacionado con la definición de la cadena de procesamiento de señales. Una configuración básica de dicha cadena que funciona es la del archivo *digitizer.mac* del ejemplo *CT simulation classic*.

Ahora bien, existe otra forma más sencilla (pero menos realista) de emular el detector. Esta es mediante el Actor de dosis (*DoseActor*) mencionado anteriormente. Dicha herramienta, permite medir, entre otras cosas, la energía y la dosis depositada en el detector durante la simulación. Esto, con una cierta resolución definida por el operador (la cual debería corresponder con la resolución del detector que se quiera simular). En la sección 11 se explican más a profundidad las características de esta herramienta.

Llegados a este punto, vale la pena mencionar que el enfoque de definir un sistema es más realista pero toma más recursos computacionales (y por lo tanto más tiempo) que el enfoque del Actor de dosis. Esto, puesto que se tienen que definir los volúmenes correspondientes a los píxeles y se tiene que simular la cadena de procesamiento de las señales. Por lo tanto, escoger un enfoque o el otro depende de la rigurosidad con la que usted quiera tratar la señal en el detector y de los recursos y el tiempo que usted disponga.

10. Definir fantomas

Definir un fantoma que represente nuestro caso de estudio de forma precisa es una de las partes más importantes al hacer simulaciones de imágenes médicas, por lo que es necesario para este punto tener conocimiento de las funciones para definir geometrías y mover objetos sobre el mundo, además de tener conocimiento de cómo estos objetos y geometrías tienen un análogo a nuestro caso de estudio. Los fantomas son herramientas reales que tienen diferentes usos en diagnósticos y evaluaciones de toma de imágenes médicas, y su uso se limita a la prueba de toma de imágenes, algoritmos y procesamientos sobre las mismas. Sin embargo, no son representaciones completamente fiables del cuerpo humano y todas las interpretaciones hechas sobre una simulación de fantoma se deben hacer teniendo esto último en cuenta.

Para entender su construcción, vamos a usar un ejemplo en el cual construimos una región modificada de un fantoma comercial CIRS015, que tiene un arreglo de microcalcificaciones de trióxido de aluminio Al_2O_3 y de Hidroxiapatita sobre un medio de PMMA con grosor conocido [9].

```
######   Contenedor   ######

/gate/world/daughters/name Contenedor
/gate/world/daughters/insert box
/gate/Contenedor/placement/setTranslation 0.0
    0.0  0 cm
/gate/Contenedor/geometry/setXLength 4 cm
/gate/Contenedor/geometry/setYLength 4 cm
/gate/Contenedor/geometry/setZLength 4 cm
/gate/Contenedor/setMaterial Breast
/gate/Contenedor/vis/forceWireframe
/gate/Contenedor/vis/setColor blue

######   Cristales HA   ######
/gate/Contenedor/daughters/name crystalHAs
/gate/Contenedor/daughters/insert sphere
/gate/crystalHAs/placement/setTranslation -4.5 0
     0 mm
/gate/crystalHAs/geometry/setRmin 0 mm
/gate/crystalHAs/geometry/setRmax 0.24 mm
/gate/crystalHAs/geometry/setPhiStart 0 deg
/gate/crystalHAs/geometry/setDeltaPhi 360 deg
/gate/crystalHAs/geometry/setThetaStart 0 deg
/gate/crystalHAs/geometry/setDeltaTheta 360 deg
/gate/crystalHAs/setMaterial HA
/gate/crystalHAs/vis/forceWireframe
/gate/crystalHAs/vis/setColor red

/gate/crystalHAs/repeaters/insert ring
/gate/crystalHAs/ring/setRepeatNumber 5
/gate/crystalHAs/ring/setPoint1 0. 0. 10. mm
/gate/crystalHAs/ring/setPoint2 0. 0. 0. mm


######   Cristales AT   ######

/gate/Contenedor/daughters/name crystalOXs
/gate/Contenedor/daughters/insert sphere
/gate/crystalOXs/placement/setTranslation 4.5 0
    0 mm
/gate/crystalOXs/geometry/setRmin 0 mm
/gate/crystalOXs/geometry/setRmax 0.24 mm
/gate/crystalOXs/geometry/setPhiStart 0 deg
/gate/crystalOXs/geometry/setDeltaPhi 360 deg
/gate/crystalOXs/geometry/setThetaStart 0 deg
/gate/crystalOXs/geometry/setDeltaTheta 360 deg
/gate/crystalOXs/setMaterial AT
/gate/crystalOXs/vis/forceWireframe
/gate/crystalOXs/vis/setColor red

/gate/crystalOXs/repeaters/insert ring
/gate/crystalOXs/ring/setRepeatNumber 5
/gate/crystalOXs/ring/setPoint1 0. 0. 2. mm
/gate/crystalOXs/ring/setPoint2 0. 0. 0. mm
```

Lo primero en este caso es definir un contenedor sobre el mundo, este contenedor será el volumen en el cual se define nuestro fantoma y por definición su desplazamiento será respecto al cero del mundo definido en el archivo *benchmarkCT.mac*. Generalmente, su geometría

va a ser una caja que contendrá los elementos necesarios del fantoma, sin embargo se pueden hacer fantomas con diferentes geomentrías que se acomoden a nuestro caso de estudio. Definimos entonces una caja con las especificaciónes geométricas que van desde la linea 6 a la línea 8. Dado que el desplazamiento es de 0 en todas las direcciones, el fantoma se va a ubicar en el centro de nuestro mundo, por la definición de su geometría, este será un cubo de 4 centímetros de largo sobre sus aristas. Luego se define el material, como ya se ha mencionado antes, hay varios materiales pre definidos en los ejemplos de gate y en archivos descargables sobre la documentación, estos materiales incluyen tejidos humanos que simulan distintos órganos de interés, en este caso se usa PMMA para simulaciones sobre el seno. Posterior a esto, se define la forma de visualización del objeto, los comandos *forceWireFrame* y *setColor* no afectan en ningún modo la simulación y son únicamente para visualizar los objetos en el espacio.

Posterior al contenedor, se definen los dos objetos *hijos* que representan los arreglos de microcalcificaciones de distintos materiales, ambos arreglos se definen con una geometría esférica y un radio de 240 μM. Para definir el volumen completo, es necesario que ambas coordenadas ϕ y θ completen una revolución. Finalmente, se define el material y un repetidor. Este repetidor es un objeto hijo de la primera calcificación, y lo que hace es replicar esta calcificación en serie sobre una geometría definida, en este caso, insertamos un repetidor en forma de anillo que dispone los objetos en forma circular con respecto al eje x, hay tres clases de repetidores que se pueden usar

Para las microcalcificaciones de hidroxiapatita se sigue exactamente el mismo proceso, pero tomando la primera calcificación en el lado opuesto del fantoma, esto resulta en la siguiente imagen al añadir una fuente y un detector.

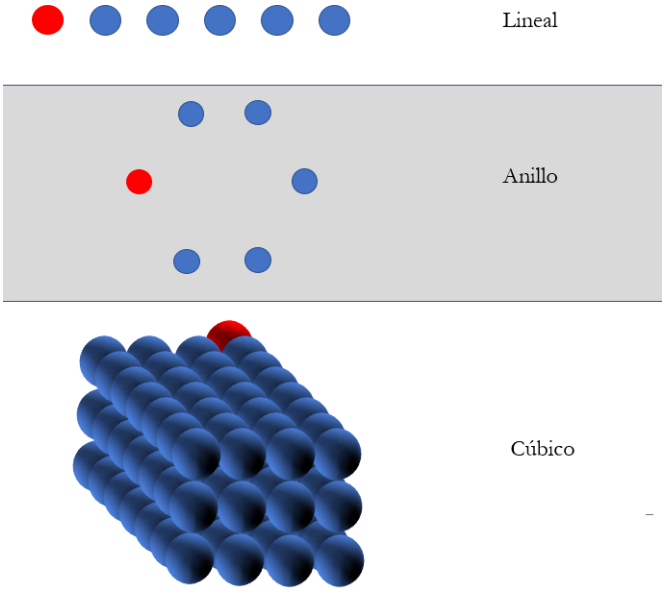

Figura 5: Disposiciones pre configuradas para repetidores en GATE, estas tres configuraciones se definen por vectores de repetición, los cuales nos dice la cantidad de microcalcificaciones en cada dirección. Todos los repetidores se definen a partir de un objeto padre representado en color rojo, y los objetos hijos del repetidor replican las características del objeto padre

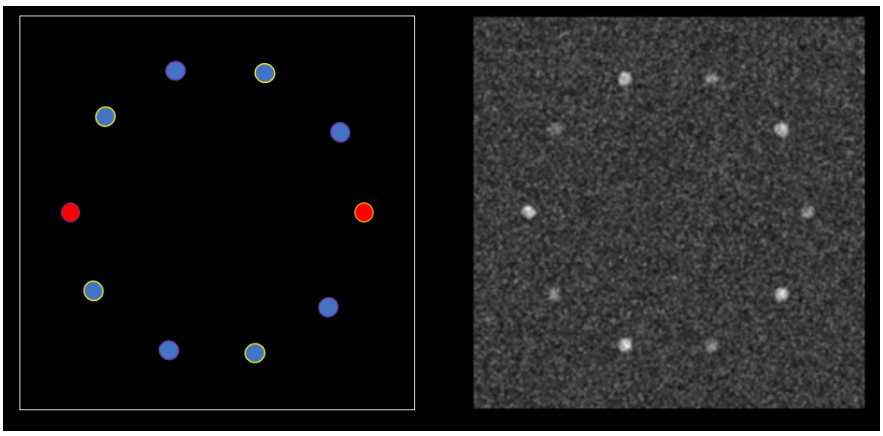

Figura 6: Estructura del fantoma simulada en GATE a partir del código anterior, a la izquierda se encuentra la disposición de las microcalcificaciones con sus repetidores, los objetos rojos son las microcalcificaciones padre y las azules son las microcalcificaciones del repetidor, el borde amarillo denota las microcalcificaciones de hidroxiapatita, mientras que el borde púrpura representa las microcalcificaciones de trióxido de aluminio, a la derecha se encuentra el patrón de absorción dado por un detector, el medio de este fantoma es PMMA

11. Definir actores

Los actores son herramientas en GATE que nos permiten interactuar con la simulación para así recolectar datos en la simulación, como pueden ser depósitos de energía y partículas producidas en un volumen dado. De hecho, como se mencionó en la sección 9, una forma en que el detector puede leer conteos y dosis es por medio de un actor, por lo que una parte fundamental para nuestras simulaciones puede ser la configuración de estos actores para recopilar archivos de salida que nos permitan tener resultados manejables. Vamos a ver un ejemplo para un actor que cuenta el número de colisiones sobre una superficie cuadrada, que es la representación de un detector.

```
/gate/actor/addActor DoseActor
    Detector
/gate/actor/Detector/save
    Dosis.mhd
/gate/actor/Detector/attachTo
    CTscanner
/gate/actor/Detector/stepHitType
    random
/gate/actor/Detector/setDoseAlgorithm
    VolumeWeighting
/gate/actor/Detector/setResolution
    512 512 1
/gate/actor/Detector/enableNumberOfHits
    true
```

Dado que los actores son directamente las herramientas que definen nuestro output de la simulación, es importante que estos actores representen de forma concisa la estadística y propiedades de los detectores usados en un estudio real. Para definir un actor, primero se define su tipo y luego se fija a algún objeto sobre la geometría, para la lista completa de tipos de actores, refiérase a *list of avaliable actors*. Además, es necesario definir un archivo de salida. Para imágenes médicas en GATE, el formato mhd es ideal, ya que estas guardan los puntos espaciales del mundo físico simulado y su proyección sobre el actor, lo que nos da una reconstrucción mas realista del caso de estudio. Para leer estos archivos *.mhd* en python, es necesario instalar el paquete SITK.

Las últimas cuatro líneas de este ejemplo se refieren a propiedades del actor, estas pueden variar dependiendo del tipo de actor que se esté usando. Por ejemplo, para el algoritmo que calcula la dosis, podemos usar tanto *VolumeWeighting* como *MassWeighting*. El primero calcula la dosis ponderándola sobre el volumen total del material, lo cual es más eficiente para detectores que sabemos son completamente uniformes. El segundo calcula la dosis de cada píxel tomando la energía depositada en su región y dividiéndola por su masa, este algoritmo puede ser preferible en algunos casos en los que se esté estudiando el detector. Por último, la resolución de nuestro actor debe ser la misma resolución de nuestro detector real, por lo que el comando *setResolution* se modifica

a conveniencia, y el comando *enableNumberOfHits* nos da un output adicional, que no cuenta la dósis sino el número de fotones que llegan al detector sin considerar la energía. Este último es completamente análogo a multiplicar la dosis por la energía en el caso donde tenemos una fuente ideal uniforme.

Cabe aclarar que los actores son objetos que pueden añadirse a cualquier volumen en nuestra simulación, por lo que no necesariamente son objetos que van ligados a los detectores y pueden tener utilidades extras, como lo pueden ser medir espectros de energía sobre el detector o sobre el fantoma.

12. Definir *outputs*

Si usted toma la decisión de simular el detector mediante el enfoque de definición de sistema, en la Sección *How to connect the geometry to a system*, se mencionan los tipos de archivo de salida en los que se puede guardar la información relacionada con las señales obtenidas en el detector durante la simulación. Y, en la Sección *Data output* se describe el contenido de estos archivos de salida y la forma en que toca declararlos. Es recomendable construir un *macro* únicamente para la declaración de los *outputs* y llamar a este archivo desde el *macro master*.

Ahora bien, algo importante a mencionar es que a partir de la Versión 9 de GATE se tiene la posibilidad de generar los archivos con extensión .npy. En la Sección *New unified Tree output (ROOT, numpy and more)* se describe como declarar este tipo de *outputs*. Los archivos con extensión .npy pueden resultar bastante amigables puesto que se pueden leer directamente con la librearía numpy de Python. Sin embargo, resulta aún más cómodo utilizar la librería Pandas, también disponible en Python.

Por otro lado, tal y como se mencionó en la sección 11, si se decanta por simular el detector con el *DoseActor*, un formato de *output* amigable es el *.mdh*.

Referencias

[1] "How much ram is recommended to be assigned to a virtual machine in virtualbox?." https://forums.virtualbox.org/viewtopic.php?f=1&t=78132.

[2] J. W. Anthony, R. A. Bideaux, K. W. Bladh, and M. C. Nichols, "Handbook of mineralogy, mineralogical society of america, chantilly, va 20151-1110, usa." http://www.handbookofmineralogy.org/.

[3] "Pubchem." https://pubchem.ncbi.nlm.nih.gov/.

[4] "Libro del web de química del nist." https://webbook.nist.gov/chemistry/.

[5] N. B. Smith and A. Webb, *Introduction to medical imaging: physics, engineering and clinical applications*. Cambridge university press, 2010.

[6] J. T. Bushberg and J. M. Boone, *The essential physics of medical imaging*. Lippincott Williams & Wilkins, 2011.

[7] "Setting up the physics." https://opengate.readthedocs.io/en/latest/setting_up_the_physics.html#.

[8] J. M. B. Andrew M Hernandez, "Tungsten anode spectral model using interpolating cubic splines: unfiltered x-ray spectra from 20 kv to 640 kv," *Med Phys.*, vol. 41, no. 4, 2014.

[9] "Mammographic accreditation phantom model 015 (data sheet)." http://www.cirsinc.com/wp-content/uploads/2020/07/015-DS-071620.pdf.