1.3 Generating complex geometries

A script to generate complex geometries is provided. An example of complex geometry is a parcel, wrapped in paper, and filled with a large number of pills distributed randomly within its volume.
The script is run with the following command:

```
$ parcel/bin/buildParcel <confIn> <confOut> <fullSetup>
```

The fullSetup flag indicates whether the other elements of the setup (dummy target, accelerator and conveyor belt) must be added. These lines are hardcoded in the code, and produce the following lines:

```
##########################################
### setup

[Lithium_target]
type = box
size = 50mm 50mm
thickness = 2mm
material = vacuum
position = 0mm 0mm 0mm
orientation = 0 0 0

[accelerator]
type = box
size = 134mm 134mm
thickness = 2000mm
material = copper
position = 0mm 0mm 1012mm
orientation = 0 0 0

[conveyer_belt]
type = box
size = 10000mm 700mm
thickness = 6mm
material = rubber
position = 0mm 0mm −353mm
orientation = 0 0 0
```

1.3.1 Configuration file

The script reads-in a configuration file (confIn). The configuration file is organized in sections, each describing a different item or set of items. Each section must follow the following syntax rules:

- it MUST START AND END with a # character (anything that follows on the same line is irrelevant); this also means that COMMENTING IS NOT ALLOWED, as it would require to use the same # character;
- each assignment is formatted as

  ```
  parameter = value
  ```

  with the all-important blank space before and after the = sign;
- it cannot contain empty lines (although empty lines between sections are allowed);
- the first parameter of a section must be the section identifier;
- for the numeric variables which are dimensional, the units are in mm, MeV and degrees;
- for vector parameters (e.g. positions), separate each value by a blank space.

Currently, the available identifiers are

- parcel
- item

The first section of the configuration file is mandatory and describes the overall parcel volume. Currently, the only possible parcel geometry type is a box. The parcel can be either filled with a single material (simple parcel) or filled with multiple items (complex parcel). In the latter case, the fill material of the parcel will be the same as the mother volume, usually air. Moreover, both the parcel and the items inside can be wrapped in one or more materials specified by the user.

Simple parcel

The parcel section is mandatory and must start with the section parameter set to parcel:

```
section = parcel
```

The mandatory parameters to be specified are the type, size, position, orientation and material. The simplest parcel is a box filled with a single material, as in the following example:

```
### parcel section start
section = parcel
name = suspiciousBox
type = box
size = 100 200 42
position = 0 0 −350
orientation = 0 0 0
material = g4_tnt
### parcel section end
```

This will generate the following configuration file:

```
##########################################
### parcel's main body

[suspiciousBox]
type = box
material = g4_tnt
size = 100mm 200mm
thickness = 42mm
position = 0mm 0mm 350mm
orientation = 0deg 0deg 0deg
```

Wrapped parcel

One can specify one or more materials to wrap the parcel by adding the wrapMaterial and wrapThickness parameters to the list. For example, the following would be a TNT box wrapped in a first layer of paper (0.1 mm thick) and a second layer of PET (0.2 mm thick):

```
### parcel section start
section = parcel
name = suspiciousBox
type = box
size = 100 200 42
position = 0 0 350
orientation = 0 0 0
material = g4_tnt
wrapMaterial = g4_office_paper g4_pet
wrapThickness = 0.1 0.2
### parcel section end
```

Notice that THE NUMBER OF THICKNESSES MUST CORRESPOND TO THE NUMBER OF MATERIALS. Each item must be separated by one simple blank.

Single items

The parcel's volume can be filled with items in a number of ways. The user specifies the parameters of the item, and the code should check whether the item is "allowed" or not. For example, it could be too large, or it could be overlapping with a pre-existing item.
The following example covers all the available features:

```
### parcel section start
section = parcel
name = suspiciousBox
type = box
size = 100 200 42
position = 0 0 −270
orientation = 0 0 0
material = g4_tnt
wrapMaterial = g4_office_paper g4_pet
wrapThickness = 0.1 0.2
### parcel section end

### item 1 section start
section = item
name = randomItem
type = box
size = random gauss 10 10 10 2 2 2
position = random uniform
```

```
orientation = random uniform
material = g4_soap
### item 1 section end

### item 2 section start
section = item
name = centeredItem
type = box
size = 10 10 10
position = center
orientation = 0 45 0
material = g4_soap
### item 2 section end

### item 3 section start
section = item
name = simpleItem
type = box
size = 10 10 10
position = 5 −10 0
orientation = 0 60 −30
material = g4_soap
### item 3 section end

### item 4 section start
section = item
name = wrappedItem
type = box
size = 10 10 10
position = −60 10 0.2
orientation = 0 0 0
material = g4_soap
wrapMaterial = g4_office_paper
wrapThickness = 0.1
### item 4 section end
```

After defining the parcel, four items are specified, which have the following characteristics:

- Item 1 gets its size, position and orientation assigned randomly. Please note the syntax of each statement from this example. Currently, only the Gaussian distribution is supported for the random generation of the size, while the position and orientation distributions can only be uniformly distributed. The distribution range is the full parcel for the position, and the full 2π for the three orientation angles. The six values after the "random gauss" statement for the size are the three mean values followed by the three sigmas of the three directions.
  Notice that the item might end up being positioned in a spot where it overlaps with a pre-existing item. In such a case, the system will keep trying by generating new positions until a "free space" is found. However, the search will be interrupted after a number of attempts[3]. This will generate a warning, but will not terminate the running of the code. Additional (maybe smaller) items which are specified later in the configuration file, might still have a chance to be allocated. As a general rule, if the parcel contains both randomly-allocated items and user-allocated items, it is better to define these latter first, so that they will be placed first and later the code will try to "find a spot" for the randomly-allocated ones.
- Item 2 has its size and orientation specified by the user, and its position is "centered". This means that it will be positioned at the center of the parcel.
- Item 3 has everything user-specified.
- Item 4 is wrapped, by following the same syntax as for the parcel wrap.

Sets of items

When the number of items to be placed is too large, one can define a set of items via the section name "itemSet", which triggers the generation of a user-defined number of items. The most complete example of an itemSet is the following:

```
### parcel section start
section = parcel
name = suspiciousBox
type = box
size = 100 200 42
position = 0 0 −270
orientation = 0 0 0
material = g4_tnt
### parcel section end

### itemSet 1 section start
section = itemSet
```

[3] This number is hardcoded in src/defaults.hpp

```
name = randomItemSet
type = box
size = random gauss 50 100 21 5 10 2.1
position = random uniform
material = g4_soap
quantity = 7
wrapMaterial = g4_office_paper
wrapThickness = 0.1
itemType = box
itemSize = random gauss 2 2 2 0.2 0.2 0.2
itemPosition = random uniform
itemWrapMaterial = g4_pet
itemWrapThickness = 0.2
### itemSet 1 section end
```

Here are the most important features:

- currently, the only allowed type is "box";
- the size and position parameters share the same syntax as for the case of a single item; if one sets

  ```
  size = all
  ```

  the item set will cover the full volume of the parcel; otherwise, the items of the set will be placed only within the user-defined range;
- the "quantity" parameter specifies the number of items to be generated in the current set;
- the item set might have wrapping materials, which are specified with the usual syntax (see parcel)
- all the parameters starting with "item" define the properties of the items of the current set.