Controlling the Data Generator, Version 2.1

Thomas Brinkhoff, IAPG, Jade University Oldenburg
http://iapg.jade-hs.de/personen/brinkhoff/
September 2003

The data generator can be run as a java application or as a java applet.

The data generator is controlled by a control file. The default name of the control file is `properties.txt`. Running the generator as applet, the name of the control file can be defined by the applet parameter `propertyfile`, e.g.:

```
 <PARAM NAME=propertyfile VALUE="propOL.txt">
```

Running the generator as application, this name can be defined by the first argument of the program, e.g.:

```
  java –classpath … generator2.DefaultDataGenerator propOL.txt
```

Instead of running the class `generator2.DefaultDataGenerator` as application, predefined (or own) subclasses like `generator2.NodeDataGenerator` or `generator2.ConstantDataGenerator` can be started.

Example of a Control File (! indicates a comment line)

```
! limits for input fields
MIN_MAXTIME = 5
MAX_MAXTIME = 400
MAX_OBJCLASSES = 20
MAX_OBJPERTIME = 40
MAX_OBJBEGIN = 100
MAX_EXTOBJCLASSES = 10
MAX_EXTOBJPERTIME = 3
MAX_EXTOBJBEGIN = 10

! settings of the generator
!urlnez = file:/C:/data/oldenburg/oldenburgGen
!urlnez = C:\\data\\oldenburg\\oldenburgGen
urlnez = ..\\data\\oldenburg\\oldenburgGen
waitingPeriod = 1
!DSO
VIZ
outputFile = OldenburgOut.txt
seed = 123

! settings of the map viewer
baseScaleFactor = 1250
maxScale = 1
minScale = 64
scale = 64
viewWidth = 500
viewHeight = 500
language = E
color = white
mapColor = white
```

Parameters Setting the Limits of the Input Fields

All supported parameters are listed in the example above. Their values may be changed for the required extreme values.

Parameters Concerning the Data Generation

Parameter *urlne*

The basic name of the network files (without `.node` and `.edge`), if the files are not compressed. Exact one of the parameters `urlne` or `urlnez` must be set. *Note:* the files ***must*** have names like `<Basicname>.<NodeOrEdge>`

Parameter *urlnez*

The basic name of the network files (without `.node` and `.edge`), if the files are compressed as zip files. Exact one of the parameters `urlne` or `urlnez` must be set. *Note* the files ***must*** have names like `<Basicname>.<NodeOrEdge>.zip`

Examples for filenames are given in the control file above. Note that Java interprets `\\` as `\`.

Parameter *DSO*

Indicates that the data-space oriented approach should be used for generating nodes. If this parameter is not set, the network-oriented approach will used. For details, see the method `computeNode` of the class `generator2.ObjectGenerator`.

Parameter *VIZ*

Indicates that the generated data be visualised. Should only be set for smaller datasets because this mode slows down the generator and increases the memory demands. Default value is no visualization.

Parameter *waitingPeriod*

Waiting period between two time stamps in milliseconds. Default value is 0, which means no waiting at all. If `VIZ` is set and a waiting period larger than 0 is set, the generator visualizes the moving objects after each time stamp that has been passed!

Parameter *outputFile (modified in Version 2.1)*

The name of the file that stores the generated points. Should only be set if the generator runs as application or as an applet in an appletviewer (but not in a browser) (in the case of an applet, the java policy file must be modified in order to allow writing into the destination directory).

If `generator2.DefaultDataGenerator` is started, it uses the class `generator2.PositionReporter`. Then, the positions of the objects at a time stamp are reported. If the name of the output file ends with ".mpf", a binary file is generated. Otherwise, a text file is reported.
A line of the ***text file*** represents the generated position of an object; it is described by the following fields (separated by tabulators):

- `newpoint` (for the first position of a new moving object), `point` (for the following positions of a moving object), or `disappearpoint` (if a moving object has reached its destination)
- the id of the point
- the sequence number (starts with 1)
- the id of the object class
- the time stamp (as integer)
- the x-coordinate (as floating-point number)
- the y-coordinate (as floating-point number)
- the current speed (in space units per time unit as a floating-point number; note: the speed may change several times between two reported positions)
- the x-coordinate of the next node that will be passed (as integer)
- the y-coordinate of the next node that will be passed (as integer)

A record of the ***binary file*** represents the generated position of an object; it is described by the following fields:

- (byte) `0` (for the first position of a new moving object), `1` (for the following positions of a moving object), or `2` (if a moving object has reached its destination)
- (long) the id of the point
- (int) the sequence number (starts with 1)
- (int) the id of the object class
- (int) the time stamp
- (double) the x-coordinate
- (double) the y-coordinate
- (double) the current speed (in space units per time unit; note: the speed may change several times between two reported positions)
- (double) the distance the object has done since the last reporting
- (int) the x-coordinate of the next node that will be passed
- (int) the y-coordinate of the next node that will be passed

If `generator2.NodeDataGenerator` is started, it uses the class `generator2.NodeReporter`. Then, the positions of the objects are reported, when they pass a node of the network. If the name of the output file ends with ".mof", a binary file is generated. Otherwise, a text file is reported.
A line of the ***text file*** represents the position of node passed by an object; it is described by the following fields (separated by tabulators):

- `0` (for the first position of a new moving object), `1` (for the following positions of a moving object), or `2` (if a moving object has reached its destination)
- the id of the point
- the id of the object class
- the time (as floating-point number)
- the current x-coordinate of the object and the node (as integer)
- the current y-coordinate of the object and the node (as integer)
- the current speed (in space units per time unit as a floating-point number; note: the speed may change several times between two reported positions)
- the x-coordinate of the next node that will be passed (as integer)
- the y-coordinate of the next node that will be passed (as integer)

A record of the ***binary*** represents the position of node passed by an object; it is described by the following fields:

- (byte) `0` (for the first position of a new moving object), `1` (for the following positions of a moving object), or `2` (if a moving object has reached its destination)
- (long) the id of the point
- (int) the id of the object class
- (double) the time
- (int) the current x-coordinate of the object and the node
- (int) the current y-coordinate of the object and the node
- (double) the current speed (in space units per time unit; note: the speed may change several times between two reported positions)
- (int) the x-coordinate of the next node that will be passed
- (int) the y-coordinate of the next node that will be passed

The external objects are not reported by the above reporter classes – by changing or replacing the reporter classes, this feature and the reported parameters for moving objects may be modified or extended by an user.

Parameter seed (new in Version 2.1)

The seed for the random generators in the class `generator.RandomGenerator`.

Parameters Concerning the Map Visualization

The following parameters concern the visualization. In the example of the control file, some of these parameters are set on reasonable values. In the other cases, the default values can be taken.

Parameter baseScaleFactor

Factor of the basic scale. This number corresponds to factor which must be applied to the scale in order to get the real scale number.

Parameter *color*

The color of the frame. This parameter is optional. The default value is special blue. Supported values are: `black`, `blue`, `cyan`, `darkGray`, `gray`, `green`, `lightGray`, `magenta`, `orange`, `pink`, `red`, `white`.

Parameter *mapColor*

The background color of the map. This parameter is optional. The default value is white. Supported values are: `black`, `blue`, `cyan`, `darkGray`, `gray`, `green`, `lightGray`, `magenta`, `orange`, `pink`, `red`, `white`.

Parameter *language*

The language used by the generator. This parameter is optional. The default value is `E` = English. Supported values are: `E` for English and `D` for German.

Parameters *minScale* / *maxScale*

The least / most detailed scale. The parameters are optional; default values are 1 / 320.

Parameters *mapWidth* / *mapHeight*

Width and height of the map (in pixel) at the basic scale 1 (i.e. in basic coordinates).

Parameter *scale*

The starting scale. This parameter is optional; the default value is the value of `maxScale`.

Parameters *viewWidth* / *viewHeight*

The width / height of the depicted part of the map in pixel. These parameters are optional; the default values are 300.

Parameters *viewX* / *viewY*

The width / height of the border of the map in pixel. These parameters are optional; the default values are 5.

Extensions

User-defined classes may require additional parameters. An example for such an extension is the class `generator2.OracleReporter`, which inserts the computed moving objects and the computed external objects into an Oracle database. This class requires the following parameters:

`dbDriverClassName`: default value: `oracle.jdbc.driver.OracleDriver`
`dbConnectionName`: default value: `jdbc:oracle:oci8:@geodata`
`dbUserName`: default value: `scott`
`dbPassword`: default value: `tiger`