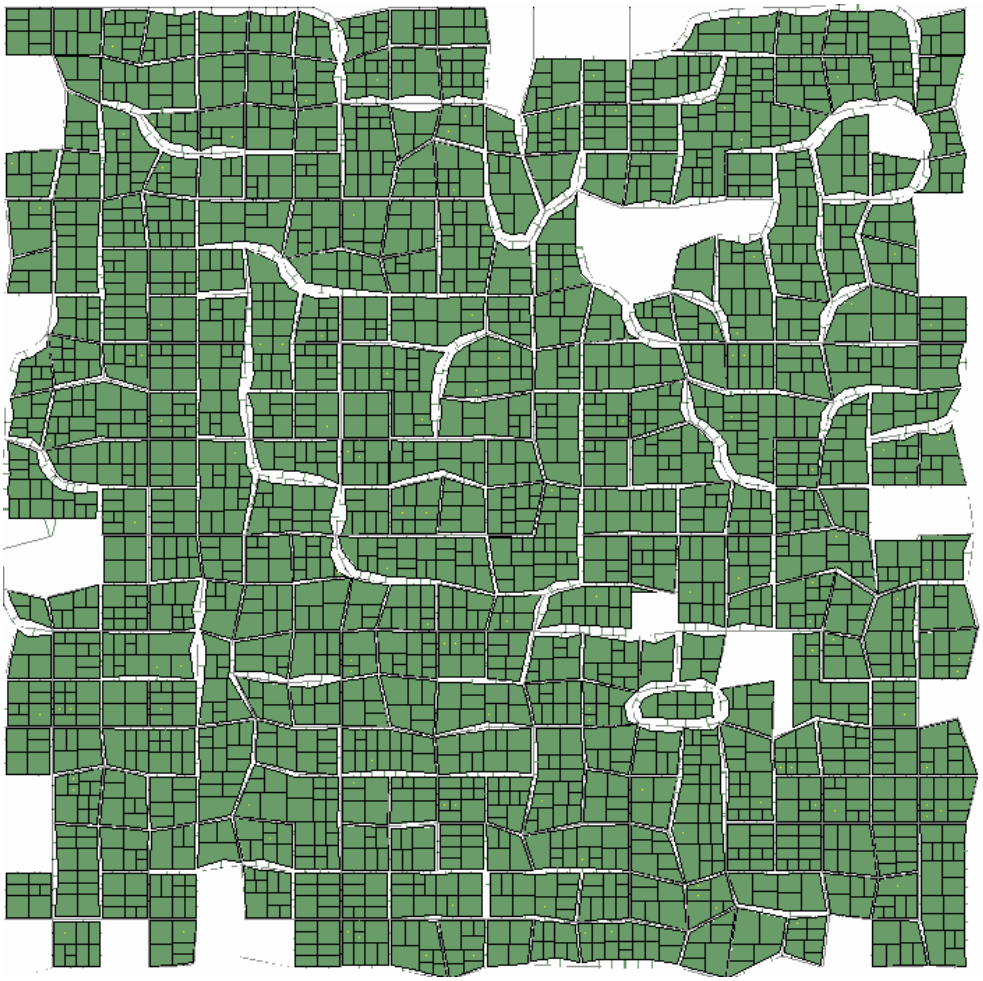
Random Map Generator v1.0

User's Guide



Jonathan Teutenberg

2003

1 Map Generation Overview

The map generator is designed to generate random city maps that can be used for testing agents designed for the Robocup Rescue domain. It is included as part of the rescuecore package provided on http://sourceforge.net/projects/rescuecore/. The full path of the package is rescuecore.tools.mapgenerator, which contains the main MapGenerator class and several supporting classes.

1.1 Command Line

The map generator is written entirely in Java and is designed to be run from the command line. Minimally, it can be run with default settings using

```
java rescuecore.tools.mapgenerator.MapGenerator
```

Optionally, a number of switches can be added to customise the production of maps. A listing of valid switched is given when any unknown argument is given to the map generator. The effects of each of these switches will be covered in later chapters.

1.2 Operation Flow

The operation of the map generator is a linear process that can be broken into a number of discrete steps.

1. Create an initial set of nodes and roads.
2. Partially randomise this set.
3. Connect any overlapping roads.
4. Weight the roads – give them a number of lanes.
5. Smooth main roads.
6. Add buildings.
7. Write to file.

All of these steps, with the exception of step 8, can be affected through the application of various switches or the inclusion of customised classes.

2 Map Initialisation

This is the first step taken by the random map generator. It involves the creation of a set of nodes, and connecting some of the nodes to form roads. The initialisation scheme can be specified by the –init switch at the command line. Currently there is only one available initialisation scheme, grid, which is the default.

2.1 Initialisation Parameters

The initialisation of the map is affected by three switches (other than the switch selecting the initialisation scheme). These switches are:

-w *xxxxxxx*

This specifies the width of the rectangle bounding the city. At the end of the map generation process there is no guarantee that parts of the city will not lie outside the bounds of this rectangle, hence it is only an approximation. The unit in which the width is given is millimetres. Default width is 1000000 = 1km.

-h *xxxxxxx*

This specifies the height of the rectangle bounding the city. Also approximate, and also given in millimetres. Default height is 1000000 = 1km.

-rd *xxx*

This gives the road density. A high value should indicate many short roads, a lower number should indicate fewer, longer roads. This is an integer value between 0 and 100. Default road density is 60.

2.2 Grid Initialisation Scheme

Currently this is the only initialisation scheme provided by the map generator. It lays nodes out in a grid between the given width and height. The distance between adjacent nodes in the grid is between 60 and 90 meters depending on the road density. Nodes are connected to the nodes immediately adjacent to them vertically and horizontally.

3 Randomisation

In this step positions of some nodes are adjusted and some roads are removed. These are chosen randomly, with the extent of the movement and the number of nodes and roads affected dependent on the values for various switches.

3.1 Randomisation Parameters

The randomisation step is affected by four switches.

-u *xxx*

This is map uniformity. Between 0 and 100% nodes will have their locations shifted, this is directly related to the uniformity value which is also between 0 and 100. Default uniformity is 75.

-m *xxx*

This is node movement. It is the maximum distance any one node will be shifted and is measured in meters. By specifying 0 movement, the node movement section of randomisation can be suppressed. Default movement is 20m.

-rd *xxx*

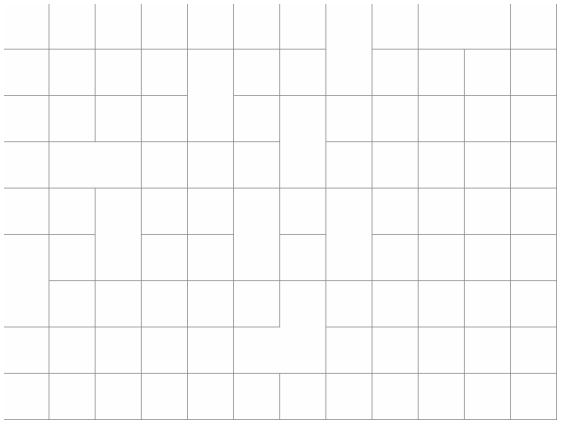
This is road density, which here affects the number of roads that will be randomly removed. Every road will have between 5 and 30% chance of being removed. If a road is chosen, a check is made to see whether the road graph is still connected. If this check passes, the removal of the road is performed. If a road is not chosen and one-way streets are allowed, there is a further 1 to 6% chance that one direction will be removed. Default road density is 60.

-nooneway

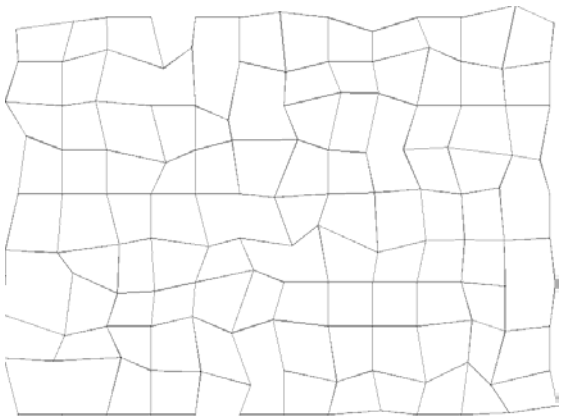
If this is enabled the chance of removing a road in only one direction is zero.

3.2 Sample Output

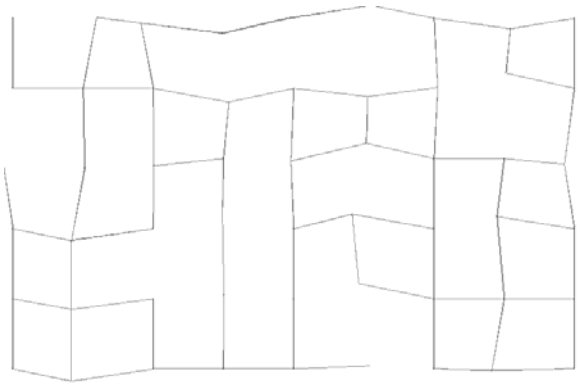
The following are samples of output of the randomisation step with various switch values. All maps are 600m by 800m (using –w 600000 –h 800000).



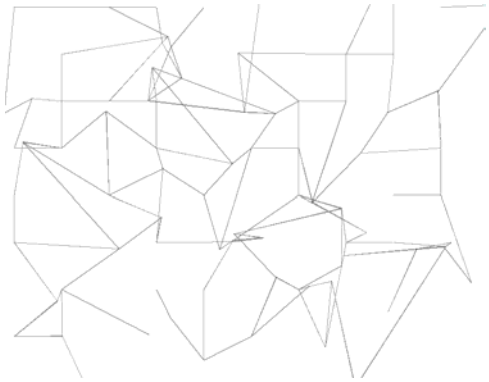
MapGenerator -d 100 –u 100



MapGenerator -d 100 –u 50



MapGenerator -d 10 –u 50



MapGenerator -d 50 –u 50 –m 100

Currently outputs with high movement, like the bottom right example, are not recommended. This is due to the fact that an error in the checking of overlapping roads means that some will not have a node placed at their overlapping point. This creates a city that is 'layered' in parts – such cities are not considered reasonable.

4 Road Weighting

The road weighting step alters the number of lanes in the roads. This may include the addition or removal of roads depending on the weighting scheme used. Currently there are two possible schemes, ‘none’ which leaves all roads as one lane and ‘ants’ which simulates a number of trips in the city to decide on each road’s importance. The road weighting scheme can be chosen using the –weight switch at the command line, with the default being ‘none’.

4.1 Road Weighting Parameters

Only two switches apply to road weighting. These are:

-u *xxx*

The map uniformity. A uniform map can be expected to have highways (several lanes wide) that extend across the map. A non-uniform map is more likely to have differing lanes throughout its roads. Default uniformity is 75.

-nooneway

If this is enabled, any given road will have the same number of lanes in each direction.

4.2 Ants Road Weighting Scheme

The ants road weighting scheme works by picking 10,000 random pairs of nodes from the map. For each pair it finds the shortest path between them, and keeps track of the number of times each road is used. If nooneway is enabled, then the counter for a road is incremented if it is traversed in either direction.

Once all runs are completed, the roads are given a number of lanes according to their usage. The 10% of most used roads are given 3 lanes, the next 20% most used are given 2 lanes and the remaining 70% are left as single lane roads. The uniformity switch has no effect on this scheme.

5 Smoothing Roads

The smoothing of roads is included to make series of main roads (those with many lanes) more natural looking. It removes sharp edges and intersections so as to make them more of a highway or motorway, rather than a suburban street.

5.1 Smoothing Parameters

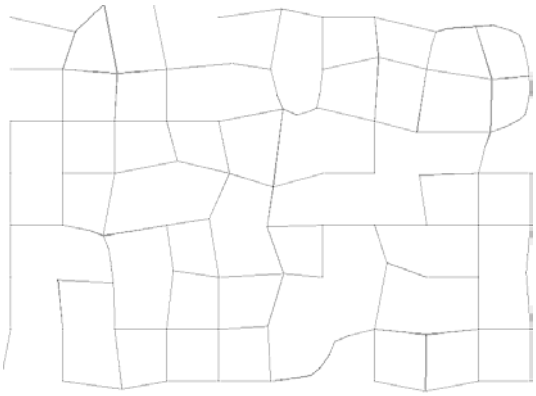
Road smoothing uses a single switch which has been specially hand crafted for this very purpose.

-sm *x*

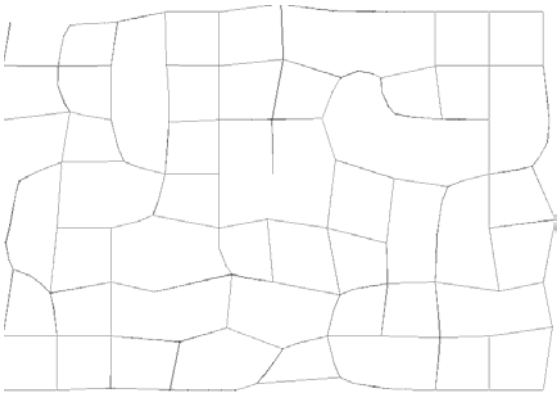
This switch indicates what sized roads should be considered highways (and hence smoothed). The number is the total number of lanes (sum of both directions) to make a road a highway. The default value is 5. For maps with nooneway enabled, this means only those with 3 lanes in each direction (or more) are smoothed.

5.2 Sample Output

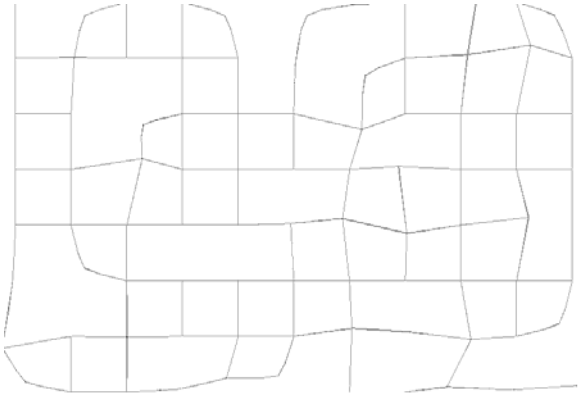
The following outputs have all had roads weighted by the ants road weighter (-weight ants) and width of 800m, height 600m (-w 800000 –h 600000). All other unspecified switches are set to the default values.



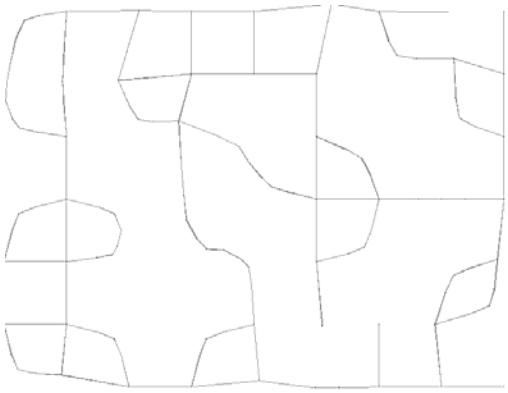
MapGenerator –sm 5 –nooneway



MapGenerator –sm 4 –nooneway



MapGenerator –sm 1 -nooneway



MapGenerator –sm 1 –rd 30 -nooneway

The first two examples show how moving from 5 to 4 (requiring only 2 lanes in each direction to make a highway, rather than 3) increases the number of smoothed roads significantly. Further reducing it to 1 (as in the third example) actually reduces the number of smoothed corners. This is because all roads are now considered equal, and it can no longer decide on a direction to smooth in at nodes which form intersections. The final example shows how this no longer causes a problem when there are few or no intersections in the map.

6 Adding Buildings

Here buildings are added to the map, the final step in the generation process. The addition of buildings includes creating entrance nodes if necessary, creating apexes, types and number of floors of buildings. Two schemes are available for use, ‘none’ which adds no buildings and ‘block’ which adds buildings by city block. These can be selected using the switch –build, with the default being block.

6.1 Building Generation Parameters

Two switches are used in building generation.

-bd *xxx*

This gives the building density. This is a measure of how tightly packed the buildings of a city are, with a high density indicating many, small buildings and a low density indicating fewer, larger buildings. This is an integer value from 0..100. Default building density is 50.

-u *xxx*

Map uniformity. A highly uniform map should have areas of tall buildings separate from areas of low level buildings. Types may also be clustered on a very uniform map. A less uniform map may have building types and heights randomly distributed around the city. Default value is 75.

6.2 Blocks Building Generator

The blocks building generator is actually given as an abstract class. It has been provided to assist in the creation of further building schemes. The concrete instantiation that is actually used when the block scheme is selected will be talked about in the next section (6.3).

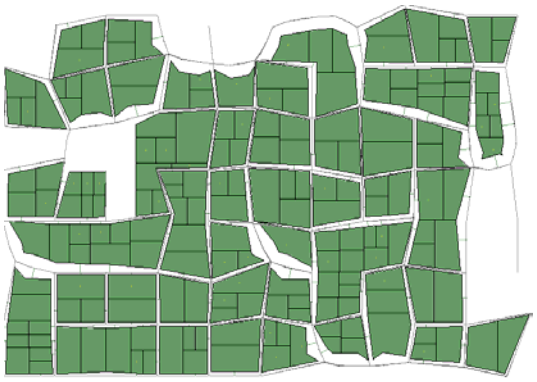
The blocks building generator finds all city blocks as a list of nodes, plus a special block representing the outer road that surrounds the city. Classes using this are expected to decide on a scheme to fill in any given city block. Entrances are then created at the closest point on the closest road to each building’s centre.

6.3 Basic Building Generator

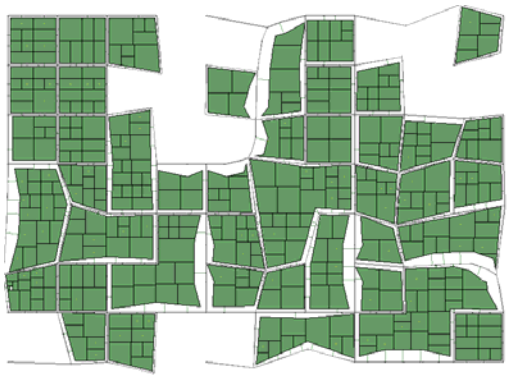
This is the default scheme used by the map generator. Blocks are filled by creating a polygon lying inside the roads of the city block. The polygon is then recursively split in half until a building's area is smaller than the minimum size (between 5,000 and 500,000 depending on building density). In addition, there is a chance that the recursion will finish earlier, leaving some larger buildings. Finally, on some occasions a block will not be filled, for now this happens arbitrarily but in future it should be tied to the building density. Building type is always 0 and building height is randomly generated between 1 and 5 floors. The uniformity is not used by this building scheme.

6.4 Sample Output

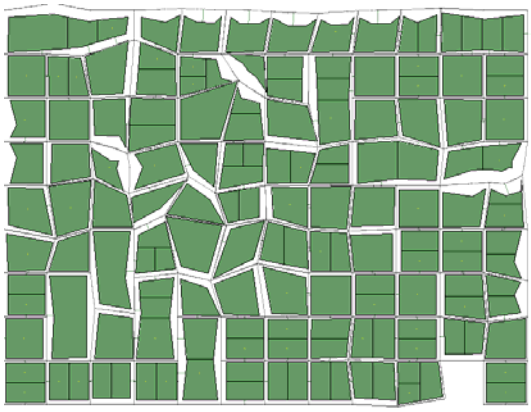
We use the same width and height values as in earlier examples. Also the ants road weighting scheme will be used (-weight ants) and nooneway enabled.



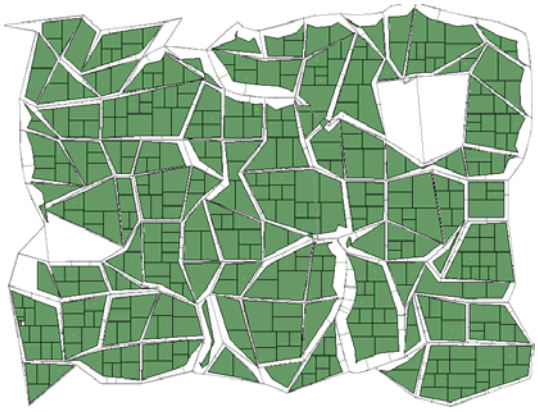
MapGenerator –bd 50



MapGenerator –bd 100



MapGenerator –rd 100 –bd 20



Map Generator –rd 100 –bd 100 –u 40 –m 40

In the first two examples we see the effect of the building density on building size. The third example shows how a map with low building density and small blocks will be given a very chunky look, and is probably not so useful for simulation purposes. The fourth example shows how the building generator deals with very randomised roads. In this case, while it looks very good, several invalid buildings have been created which would require editing by hand. Most of these cases (but not all) are due to the error in the algorithm for joining overlapping roads.

7 File Output

Once map generation is complete, the internal representation is written to file. The building.bin, node.bin and road.bin files will be created in the directory from which the map generator is run.

7.1 Road File

- IDs are assigned from (number of nodes+2) up to (nodes+roads+2)
- The length of a road is the Euclidean distance between its head and tail
- All roads are kind 0
- The width of each road is 2000mm per lane

7.2 Building File

- IDs are assigned from (nodes+roads+3) to (nodes+roads+buildings+3)
- Each building is given only one entrance
- The floor area is exactly the area defined by its apexes in 0.01m units
- The total floor area is the floor area times number of floors

7.3 Node File

- IDs are assigned from 1 to (nodes+1)
- No signals, pockets or shortcuts exist
- Lists of edges are computed during file output – this can be the slowest operation in the whole map generation process

8 Additional Notes

This section contains any information that has not yet been covered, future versions and known issues.

One Way Streets

It appears that the current traffic simulator cannot handle one way streets. In addition, it does not seem to be able to handle roads with differing values for lines to head and lines to tail (this may need confirming). For now, all maps should be constructed with the –nooneway flag set.

Rotation

One switch that has not been covered is the –rotate flag. This will rotate the nodes about their centre before buildings are added. This is included for aesthetic purposes only.

Overlapping Roads

As mentioned earlier, overlapping roads are not yet handled correctly. For now, it is best to keep node movement limited so that roads do not overlap (movement less than 30 should guarantee this).

Bad Buildings

Sometimes there will be a few bad buildings in a map (up to 3 have been found in some large maps). It is most likely that these buildings have two identical apexes (although this hasn't been checked yet). It is recommended that maps are checked with the JGIS editor before use, although these buildings do not significantly affect simulation.