Evacuationz Exercise Guide

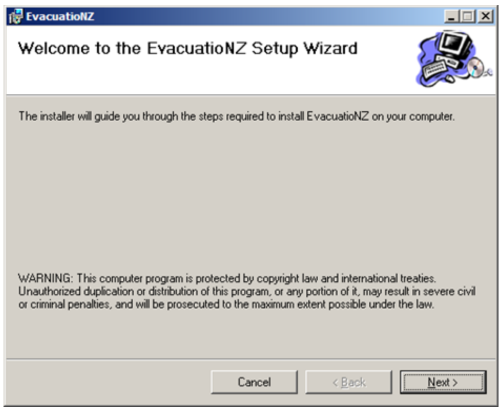
06-Jun-2022

Contents

SOFTWARE INSTALLATION

In addition to the EvacuatioNZ software you need to install the freely available yEd graph tool (http://www.yworks.com/en/products_yed_about.html), have the latest version of Smokeview (https://code.google.com/p/fds-smv/) and also a standard web browser.

Step 1
Open a command line and type 'winget install Evacuationz' (without quotes) to download the latest version from the Windows Package Manager repository and then follow the instructions. The latest version of EvacuatioNZ uses version 4 of the Microsoft .NET Framework so you may be asked to download some files. In in most cases Windows will likely already have all required files.

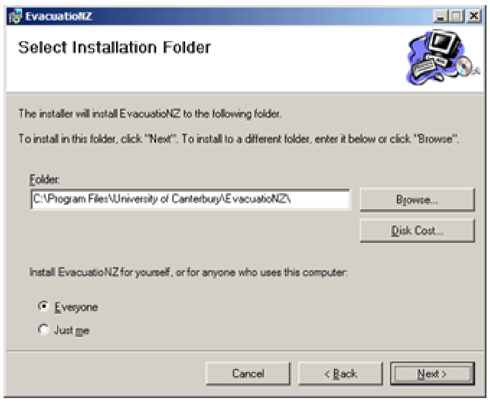

EvacuatioNZ

Welcome to the EvacuatioNZ Setup Wizard

The installer will guide you through the steps required to install EvacuatioNZ on your computer.

WARNING: This computer program is protected by copyright law and international treaties. Unauthorized duplication or distribution of this program, or any portion of it, may result in severe civil or criminal penalties, and will be prosecuted to the maximum extent possible under the law.

Step 2
Install the software to the default folder and not to any other folder.


EvacuatioNZ

Select Installation Folder

The installer will install EvacuatioNZ to the following folder.

To install in this folder, click "Next". To install to a different folder, enter it below or click "Browse".

Folder:
C:\Program Files\University of Canterbury\EvacuatioNZ\

Install EvacuatioNZ for yourself, or for anyone who uses this computer:

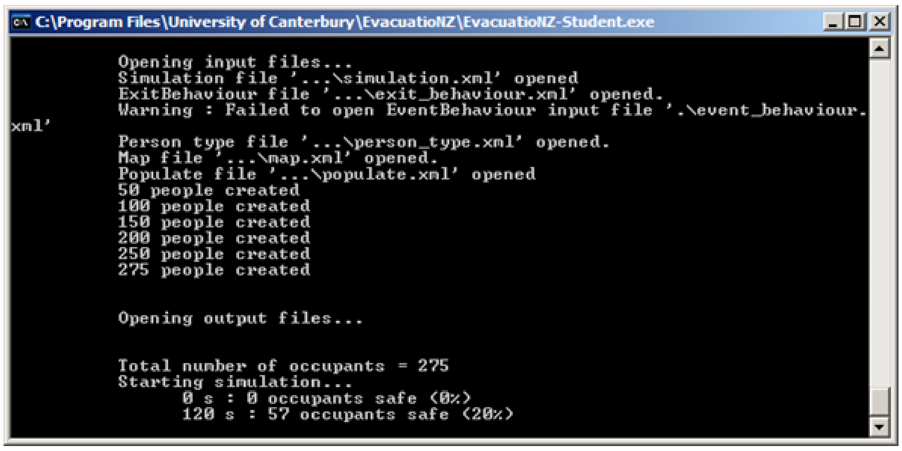
(●) Everyone
( ) Just me

The installer will also create a folder on the user's Start Menu in All Programs > EvacuatioNZ. The installation will also create an 'EvacuatioNZ' folder in the user's Documents folder in which a test project will be created.

There have been a couple of recent updates that have not yet been reflected in the installation setup so the ‘agent-type.xml’ and ‘exit-behaviour.xml’ files in the ‘…Documents\EvacuatioNZ\Test’ folder need to be replaced with new copies.

Step 3

From the Start Menu click All Programs > Evacuationz > Evacuationz to run the test file. You should see a DOS window open with a volume of text scrolling past – do not worry about not being able to read this.



```
C:\Program Files\University of Canterbury\EvacuatioNZ\EvacuatioNZ-Student.exe

          Opening input files...
          Simulation file '...\simulation.xml' opened
          ExitBehaviour file '...\exit_behaviour.xml' opened.
          Warning : Failed to open EventBehaviour input file '.\event_behaviour.
xml'
          Person type file '...\person_type.xml' opened.
          Map file '...\map.xml' opened.
          Populate file '...\populate.xml' opened
          50 people created
          100 people created
          150 people created
          200 people created
          250 people created
          275 people created


          Opening output files...


          Total number of occupants = 275
          Starting simulation...
                0 s : 0 occupants safe (0%)
              120 s : 57 occupants safe (20%)
```

Step 4

Once the program has finished you should find a file called log.html in the Documents > Evacuationz folder. You can open this in a browser and verify that the test has been successfully executed by seeing something like this:

```
Log file for EvacuatioNZ version 2.11.2 (Sep 20 2017)
Started at 08:50:48 on Wednesday 29 September 2021

                         EvacuatioNZ
                    (c) Michael Spearpoint
        version 2.11.2 - Public release (Sep 20 2017)

 This software is provided under the terms of the license available in the Program
                              Files folder.
Opened Project file
Initialising
scenario C:\Users\User\Documents\EvacuatioNZ\Test\scenario.xml

        Opened Scenario file C:\Users\User\Documents\EvacuatioNZ\Test\scenario.xml
        Random number seed set to pseudo-random default
        Opened Evacuation output file C:\Users\User\Documents\EvacuatioNZ\Test\~evac.csv
        Opened Simulation file C:\Users\User\Documents\EvacuatioNZ\Test\simulation.xml
        Agent processing procedure set to enz_sequential by default
        Maximum time set to 3600 s
        Time step set to 1.00 s
        Output frequency set to 600.00 s
        Maximum node density set to default value of 2.75 agents/m2
        Specific flow set to default value of 1.33 agents/s/m effective width
        Door flow model set to default enz_sfpe
        Opened ExitBehaviour file C:\Users\User\Documents\EvacuatioNZ\Test\exit-behaviour.xml
        Opened AgentType file C:\Users\User\Documents\EvacuatioNZ\Test\agent-type.xml
        Opened SystemType file C:\Users\User\Documents\EvacuatioNZ\Test\system.xml
        Opened Map file C:\Users\User\Documents\EvacuatioNZ\Test\map.xml
        Opened Populate file C:\Users\User\Documents\EvacuatioNZ\Test\populate.xml
        Opened Node output file C:\Users\User\Documents\EvacuatioNZ\Test\~nodes.csv
        Opened Agent                                          tracking output
        file C:\Users\User\Documents\EvacuatioNZ\Test\~agents_1.csv
        Total number of agents = 33
        Simulation #1 stopped at 323.00 s

Completed at 08:50:49 on Wednesday 29 September 2021
```

Step 5

You should also have a shortcut on your desktop to a program called ‘Runz’ (and also in All Programs > Evacuationz). Use Runz to execute the test project file ‘project.xml’ which is in the Documents > Evacuationz folder. You should see a DOS box quickly appear and disappear similarly to Step 3 and when you open the log.html file it will be same except the Started and Completed times in the file will have been updated from previously.

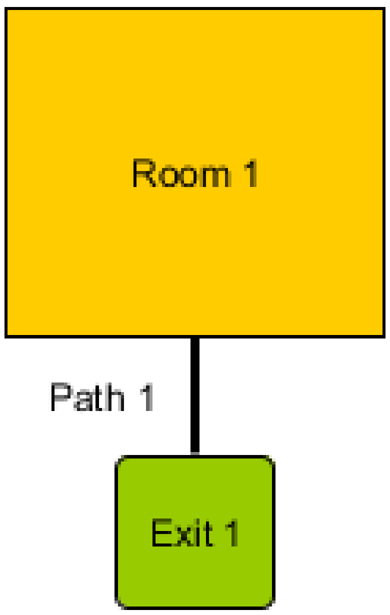
If the DOS box does not close when using Runz in Windows 8 then right-click on the desktop shortcut, go to Properties, then Compatibility and set the program to run in Windows 7 mode.

EXERCISE 1 – GETTING STARTED

This exercise introduces the basic concepts of using the Evacuationz software and how to use the map file. Each time Evacuationz is run it will generate a file called log.html in the Documents > Evacuationz folder. This file can be used to check the outcome of a simulation and track any runtime errors.

Evacuationz can be run directly from a batch file and input files can be created as XML files without the need to use any third-party software other than a text editor. However here we will create the input using a graphical network tool and run simulations using a Windows interface program.

Map files (in .graphml format) are edited using a program called 'yEd'. A simple template file is available that is setup to work with Evacuationz. To run a simulation use the Runz program to select the map file.



Use RuNZ to send the Exercise 1.graphml file to Evacuationz. Check that your log file looks similar to below…

```
Log file for EvacuatioNZ version 2.11 (Jul 5 2017)
Started at 13:37:39 on Tuesday 18 July 2017

                         EvacuatioNZ
                  (c) Michael Spearpoint
                 University of Canterbury
          version 2.11 - Student release (Jul 5 2017)

  This software is provided for research purposes only and is not intended for
                          commercial use.

Initialising scenario C:\MJS\Canterbury\ENFE 615 - Human Behaviour in
Fire\EvacuatioNZ exercises\Exercise 1\Exercise 1.graphml

      Opened Scenario file C:\MJS\Canterbury\ENFE 615 - Human Behaviour in
      Fire\EvacuatioNZ exercises\Exercise 1\Exercise 1.graphml
      Random number seed set to pseudo-random default
      Opened Simulation file C:\MJS\Canterbury\ENFE 615 - Human Behaviour in
      Fire\EvacuatioNZ exercises\Exercise 1\Exercise 1.graphml
      Opened ExitBehaviour file C:\MJS\Canterbury\ENFE 615 - Human Behaviour in
      Fire\EvacuatioNZ exercises\Exercise 1\Exercise 1.graphml
      Opened AgentType file C:\MJS\Canterbury\ENFE 615 - Human Behaviour in
      Fire\EvacuatioNZ exercises\Exercise 1\Exercise 1.graphml
      No AgentType specification found, using program defaults
      Opened SystemType file C:\MJS\Canterbury\ENFE 615 - Human Behaviour in
      Fire\EvacuatioNZ exercises\Exercise 1\Exercise 1.graphml
      Opened Map file C:\MJS\Canterbury\ENFE 615 - Human Behaviour in
      Fire\EvacuatioNZ exercises\Exercise 1\Exercise 1.graphml
      Opened Populate file C:\MJS\Canterbury\ENFE 615 - Human Behaviour in
      Fire\EvacuatioNZ exercises\Exercise 1\Exercise 1.graphml
      Total number of agents = 1
      Simulation #1 stopped at 1889.50 s

Completed at 13:37:40 on Tuesday 18 July 2017
```

As we get more into the advanced capabilities of the Evacuationz software you will need to write some XML scripts. XML is a standard markup language used for many applications and you can find out more at https://en.wikipedia.org/wiki/XML (you only really need to read the 'Key terminology' section).

EXERCISE 2 – POPULATING A NODE AND GETTING SOME OUTPUT

There are several ways in which you can populate nodes in EvacuatioNZ. This exercise illustrates the simple method to populate a node with an agent. By selecting a node in yEd and using the 'Data' in the Properties View the number of agents can be set in the Agents field.

The agent's characteristics are set to the default values: a pre-evacuation time of 1800 s; a maximum walking speed of 0.3 m/s and a 'starting' travel distance equal to the sum of the node dimensions. In this example the node is 10 m by 10 m and so the starting distance is 10 + 10 = 20 m. You can read the paper by Gwynne et al. on "Bounding defaults in egress models" in Fire and Materials if you want to know more about where the default numbers come from.

If no length is specified for the path (i.e. the path is 0.0 m) then the default distance is the square root of the half dimensions of the two connected nodes. Since the 'Exit' node is a safe node then it is not given any dimensions. In this example the node is 10 m by 10 m and thus the travel distance is $\sqrt{(10/2)^2 + (10/2)^2}$ = 7.07 m. This value is added to the starting distance making a total travel distance of 20 + 7.07 = 27.07 m.

The exercise also shows how to set up a scenario by adding XML instructions to the 'scenario' data in the Properties View. The instructions get Evacuationz to produce six different output files in the same folder as the input file.

```
<ENZ_Scenario>
        <Files>
                <!-- Create various output files in the same folder -->
                <Results />
                <Evacuation />
                <Nodes />
                <PreEvacuation />
                <Agents />
                <Log />
        </Files>
</ENZ_Scenario>
```

Given the total travel distance of 27.07 m and a walking speed of 0.3 m/s the travel time = 27.07 / 0.3 = 90 s. Adding the pre-evacuation time of 1800 s then the total egress time is 1800 + 90 = 1890 s. Check the `evac.csv` and `results.html` to confirm this result and that the `pre-evac.csv` file verifies that a pre-evacuation time of 1800 s is used.

The `nodes.csv` file shows the number of agents in each node at each time step and the `agents_1.csv` shows the node that an agent is in at each time step. The element `<Log />` simply copies the `log.html` into the same folder as the .graphml input file.

EXERCISE 3 – SETTING AGENT PROPERTIES

Following on from the previous discussion regarding uncongested movement and pre-evacuation delay we now look at how we can incorporate these parameters into Evacuationz.

Part (a) Agent Type Properties

This exercise shows how to modify the agent characteristics by using the agent type definition capability. To create an agent type definition, insert the following XML into the 'agent type' data in the Properties View:

```
<ENZ_AgentType>
    <AgentTypeDefinition>
        <Speed>1.2</Speed>
        <PreEvacuation>30</PreEvacuation>
        <StartDistance>0</StartDistance>
    </AgentTypeDefinition>
</ENZ_AgentType>
```

Here the agent characteristics have been set to a pre-evacuation time of 30 s, a maximum walking speed of 1.2 m/s and a 'starting' travel distance of 0 m. The travel distance in this exercise is now 7.07 m so the travel time = 7.07 / 1.2 = 6 s and the egress time is 30 + 6 = 36 s as there is the 30 s pre-evacuation delay.

Check both the `evac.csv` and `results.html` files to confirm this result, where an additional 0.5 s needed to complete the simulation. Check that the `pre-evac.csv` file verifies that a pre-evacuation time of 30 s is used.

There is more information regarding the agent ‘start distance’ in a later exercise.

Part (b) Units

Note how no units were specified for the walking speed etc. If no units are given then EvacuatioNZ uses a default set. The fundamental method to specify the units is to use the units="*unit type*" attribute. A list of unit categories, types and defaults is given in the EvacuatioNZ manual. Units can be freely mixed in the input and the format provides for alternate specifications.

We could have written (for example):

```
<ENZ_AgentType>
	<AgentTypeDefinition>
		<Speed units="m/s">1.2</Speed>
		<PreEvacuation units="min">0.5</PreEvacuation>
		<StartDistance units="ft">0</StartDistance>
	</AgentTypeDefinition>
</ENZ_AgentType>
```

For the output files the following elements allow units, for example:

```
<ENZ_Scenario>
	<Files>
		<Evacuation units="minutes"/>
		<PreEvacuation units="hours"/>
		<Nodes units="s"/>
	</Files>
</ENZ_Scenario>
```

where units are explicitly specified Evacuationz checks that they are appropriate for the given parameter. For example, assigning a length unit to a time parameter will cause an error.

A list of unit categories and types is given below:

Time

- Seconds$^+$: seconds, s
- Minutes: minutes, min
- Hours: hours, hr

Length

- Metres$^+$: metres, meters, m
- Centimetres: centimetres, centimeters, cm
- Millimetres: millimetres, millimeters, mm
- Feet: feet, ft
- Inches: inches, in

Velocity

- Metres per second$^+$: metres per second, meters per second, metres/second, meters/second, m/s
- Metres per minute: metres per minute, meters per minute, m/min
- Feet per minute: feet per minute, feet/minute, ft/min

Agent Density

- <u>Agents per metre squared</u>$^{+}$: agents per metres squared, agents per meters squared, agents/m2
- Square metres per agent: metres squared per agent, meters squared per agent, m2/agent

+default unit

EXERCISE 4 – NODE AND PATH DIMENSIONS

The previous exercise introduced the mechanisms used by Evacuationz to represent nodes and paths. This exercise illustrates these mechanisms in more detail as we will need to understand more about the way they work as we move through the following exercises.

Part (a) Node Dimensions

Recall from Exercise 2 that if a path length is set to 0.0 m then Evacuationz assumes a distance of the square root of the half dimensions of the two connected nodes. This effectively represents a travel distance from the centre of one node to another. Since safe nodes do not have any dimensions then these do not contribute to the default path length calculation.

The floor area of a node can be modified by editing the node's 'Length' and 'Width' items in Properties View > Data. For example if the node dimensions are set to 100 m with both the agent pre-evacuation and start distance set to zero the resulting path length is $\sqrt{(100/2)^2+(100/2)^2}$ = 70.71 m with a corresponding total evacuation time of ~60 s.

Part (b) More Node Dimensions

If a new node is added with length 50 m and width 10 m and connected to the existing node then the default path length between Room 2 and Room 1 is $\sqrt{(50/2)^2+(10/2)^2}+\sqrt{(100/2)^2+(100/2)^2}$ = 96.21 m and the path length between Room 1 and the safe node is 70.71 m as before. An agent placed in the new node has to travel 96.21 + 70.71 = ~167 m and at 1.2 m/s takes 139 s to evacuate.

Part (c) Path Lengths

The path length can be changed by using the connection's 'Path length' item in Properties View > Data. For example, if we wanted to represent an agent standing immediately adjacent to a connection then the path length could be set to something small.

Using the map from part (b) we can set the path length to 0.01 m for the connection leading to the safe node and placing an agent in the neighbouring node gives an evacuation time of 1 s even with the node having 100 m dimensions. However, if an agent is located in the farthest node then their travel distance is now 96.22 m and so at 1.2 m/s this gives an evacuation time of ~81 s. This means there has been no account of the travel distance across the second node.

Part (d) More Path Lengths

Conversely to part (c), we could make the path length 1000 m between the safe node and neighbouring node and the evacuation time for an agent in farthest node becomes ~914 s.

Summary

The important point to note from these exercises is that it is possible to set path lengths that are independent of the node dimensions. It is therefore possible to have a path length many times more or less than the node size. The only time the path lengths are determined by the node dimensions is when the path length is 0.0 m. In general the ‘centre-to-centre’ approach to path lengths (i.e. using 0.0 m in the yEd map) is a reasonable approximation for most applications. The agent ‘start distance’ property can be used to further modify the travel distance and this is discussed in a later exercise.

EXERCISE 5 – DOOR WIDTHS

Part (a) Specified Door Width

This exercise illustrates how to specify an exact path length and how to create a door constriction along a path. To set a path length, modify the 'Path length (m)' data associated with the 'Path' edge. In this example the length is set to 0.01 m to over-ride the default path distance algorithm.

To create a 600 mm wide door add the following XML to the 'Path' edge 'Evacuationz XML' data:

```
<ConnectionType type="enz_door">
        <Width>0.6</Width>
</ConnectionType>
```

Here we can compare the Evacuationz prediction for 50 agents to pass through the door with the had calculation approach Since there is effectively no travel distance or pre-evacuation delay then the egress time is the queuing time:

The effective width of the door, W_e = (0.60 - 2 × 0.15) = 0.30 m.

Actual flow, $F_a = W_e \times F_c$ = 0.30 × 1.3 = 0.40 persons/s

Time = N / F_a = 50 / 0.40 = 125 s (about 2.1 min)

compared with 126 s from Evacuationz.

Part (b) Unspecified Door Width

If a width is not defined then the flow is set to 0.67 agents/s (i.e. equivalent to a 0.8 m wide door). Rerunning the example with no width defined gives a time of 75.5 s and you can check your own hand calculation.

[**Note:** I would suggest never leaving a door width undefined as the default could be changed in the future, for example, if I were to decide that a smaller equivalent width was appropriate.]

Part (c) Door Leaves

To represent the effect of self-closing devices (closers) door flows can also be specified in terms of the number of door leaves using the <Leaves> element. The flow is fixed to 50 agents/min per leaf, any <Width> element is ignored and fractional leaf values are rounded to the nearest integer. Thus in Evacuationz we write:

```
<ConnectionType type="enz_door">
        <Leaves>1</Leaves>
</ConnectionType>
```

When Evacuationz runs it changes the effective door width to get the required flow. If the example is modified to include a single leaf we get a time of 60 s as expected given we have 50 agents. Try adding a width element and or using fractional leaf values to see what happens.

Part (d) Door Closer

An alternative to using the door leaf functionality is to associate a closer with a door using the following

```
<ConnectionType type="enz_door" closer=”enz_true”>
        <Width>1.0</Width>
</ConnectionType>
```

to get the same result as Exercise 5c. Note that in the `results.html` file that the description of the connection says that the flow has been restricted by the presence of the closer.

If the defined width of the door means the flow is less than with a leaf with a closer then the lesser value is used. Therefore, if we had defined the door width as 600 mm as before the time would be the same as in Exercise 5a.

Part (e) Openings

To create an opening (i.e. a theatre chair or stadium bench) rather than a door, use the following XML for the 'Path' edge 'Evacuationz XML' data:

```
<ConnectionType type="enz_opening">
        <Width>0.6</Width>
</ConnectionType>
```

Openings differ from doors in that they do not have any boundary layer widths included. In this example we get a simulated time of 63.5 s and you can check this with a hand calculation.

[Note: In a future release I will include the ‘theatre_chair’ and ‘stadium_bench’ terminology into the software as well as ‘opening’.]

EXERCISE 6 – CONGESTED MOVEMENT

We will now create a similar scenario in Evacuationz using the simple map created in Exercise 5. The agent characteristics are set to a pre-evacuation time of 0 s, a maximum walking speed of 1.2 m/s and a 'starting' travel distance of 20 m. The node is populated with 200 agents and the door width is set to 1000 m, an arbitrarily wide opening to make sure no queue is created.

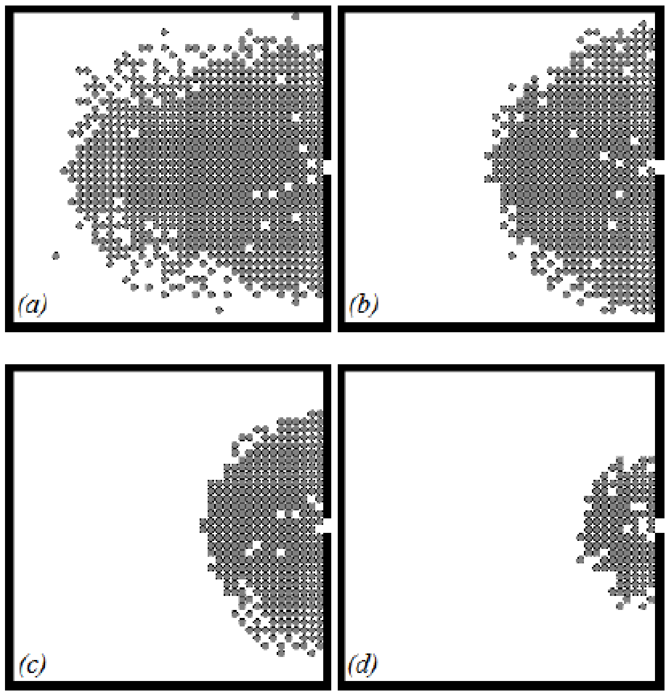
By hand calculation the occupant density is 200 / (10 × 10) = 2 persons/m^2.

From the SFPE Handbook figure at 2.0 persons/m^2 the walking speed is 40 m/min = 0.66 m/s or using the equation $S = 84 \times (1 - 0.266 \times D)$

$S = 84 \times (1 - 0.266 \times 2.0) = 39.3$ m/min = 0.66 m/s

Therefore, egress time is 20 / 0.655 = 30.5 s

Running the Evacuationz input the simulation gives a total egress time of 22.5 s which is clearly faster than the hand calculation. The reason for this difference is that even though agents do not 'take up space' in a node, Evacuationz includes a 'congested node' algorithm to adjust for the effect that some agents will have to be closer to the door than others when the queue to the door is formed. This can be illustrated in the diagram below (from Xu X, Song W, Ma Y, 'Analysis of detour behavior in pedestrian dynamics by a cellular automata model with perceptual range')



We can see that an agent joining the back of the queue does not need to move the whole distance across the room and the distance to travel depends on the number of agents in the space. Once the agent is in the queue then the queuing time essentially includes the remaining travel distance to the door. Obviously these effects are dealt with directly in models that use fine nodal mesh and continuous approaches but not with a tool such as Evacuationz.

If this ‘congested node’ algorithm is disabled then an egress time of 31.5 s is obtained for this scenario but benchmarking the software has shown we get simulation times that are conservative if this algorithm is disabled when compared to equivalent hand calculations in more complex map geometries.

EXERCISE 7 – STAIRS

Representing stairs in Evacuationz (and yEd) is a bit trickier than doors as we have to account for the constricting effect as agents enter the stairs as well as the effect of the steps on their movement speed. Even after many years investigating this aspect I still do not feel that this element is as complete as I would like it to be in Evacuationz.

Part (a) Movement speed

In this example we will look at how fast will an agent travels in Evacuationz if the tread is 305 mm (12 inches) and the riser is 165 mm (6.5 inches). First we can carry out a hand calculation using the FEDG approach such that:

$$S = k\,(1 - 0.266\,D)$$

$$k = 51.8\sqrt{T/R}$$

$$= 51.8\sqrt{305/165}$$

$$= 70.4$$

When D is small use 0.54 persons/m^2

$$S = 70.4 \times (1 - 0.266 \times 0.54)$$

$$= 60.3 \text{ m/min} \quad \text{or } 1.0 \text{ m/s}$$

This is the same as the speed given in C/VM2, Table 3.4 for the given tread and riser combination.

In this exercise the map is amended to include a new node and connection type. The new node is created to represent a 2 m by 10 m long stair space, named 'Stairs'. A stair 10 m long connection is created between the 'Room' and the 'Stairs' with a width of 2.00 m using the following XML:

```
<ConnectionType type="enz_stairs">
        <Width>2.00</Width>
        <Tread>0.305</Tread>
        <Riser>0.165</Riser>
</ConnectionType>
```

Another connection is created between the 'Stairs' node and 'Exit' with a length of 0.01 m and the following XML added to the 'Evacuationz XML' data:

A single agent is defined in the 'Room' node so that we do not have any congestion on the stairs which would slow the movement. The agent is set to have a 0 s pre-evacuation delay and a 0 m start distance so they only travel down the stairs. The result from the simulation is 10.0 s compared with the expected time of 10 s (i.e. the time to travel 10 m at 1 m/s).

Part (b) Flow

In this example we now investigate the flow and travel of multiple agents. We now place 10 agents in the room node and we can find the flow time from:

> W = 2 m thus $W_e = W - (2 \times 0.15) = 1.7$ m
> From SFPE Handbook Table 59.5, F_s = 1.09 agents/s/m eff. width
> $F_a = 1.7 \times 1.09 = 1.85$ persons/s
> $t_{flow} = 10/1.85 = 5.4$ s

The travel time of each agent is again 10 s since the occupant density in the stair node is 0.54 agents/m^2 so the time for the last agent to reach the exit is 10.0 + 5.4 = 15.4 s. Running the simulation gives a total time of 15.5 s and it takes 6 s for the room to clear corresponding to the 5.4 s in our calculation above.

Now let us see what happens if we have more agents in our simulation, for example increase the number to 20. The scenario now becomes more complex because the stair will fill to capacity before all of the agents have left the room. Evacuationz also will adjust the movement speed of the individual agents each time step as the occupant density in the stair node increases and decreases as agents enter and leave. We will return to the capacity of nodes when we look at different ways to populate nodes.

Part (c) Handrails

In general it is fairly easy to determine the overall effective width of a stair by hand however EvacuatioNZ does have a method to do the calculations for the user. This one day might be useful if someone wanted to examine the sensitivity of a simulation if the handrail arrangements are allowed to vary stochastically. Using the elements described below EvacuatioNZ will calculate the overall effective width of the stairs as a combination of walls, intruding handrails and open stair handrails.

The sides of stairs can be allocated handrails and/or be defined as not having a wall so that no boundary layer is applied. Both, one or neither side can have handrails and additional handrails can be defined in the open part of the stairs. Similarly both, one or neither side of the stairs can be defined as having a wall.

The <Walls> element can be used to specify whether walls are located along the stairs where integer values of 0, 1 or 2 are valid. By default it is assumed that both sides of the stairs have walls and so the 0.15 m boundary applies, i.e. a value of 2 is applied.

```
<ConnectionType type="enz_stairs">
        <Walls>Integer</Walls>
        . . . width, tread, riser elements. . .
</ConnectionType>
```

Handrails are defined as either being along a wall or open, i.e. in the open part of the stairs. Both types of handrails can be applied to a set of stairs. For open handrails the enz_open attribute type is applied with valid <Count> element integers greater than zero.

```
<ConnectionType type="enz_stairs">
        <Handrails type="enz_open">
                <Count>Integer</Count>
        </Handrails>
        . . . width, tread, riser elements. . .
</ConnectionType>
```

For wall handrails the enz_wall attribute type is applied with valid <Count> element integers being 0, 1 or 2. A <Distance> element is used to specify the distance of the handrail away from the wall/s. If two wall handrails exist then the distance is the same and, by default, the dimension is in metres. If the distance of the handrail is less than 0.15 m then the boundary layer is set to 0.15 m whereas if the handrail distance is greater than 0.15 m then its dimension is used.

```
<ConnectionType type="enz_stairs">
        <Handrails type="enz_wall">
                <Count>Integer</Count>
                <Distance>Value</Distance>
        </Handrails>
        . . . width, tread, riser elements. . .
</ConnectionType>
```

Part (d) Floor-to-floor Height

Rather than specifying the diagonal travel length of stairs the other method available is to define the floor-to-floor height *H*, along with tread *T* and riser *R* dimensions. The diagonal path length *L* is then calculated from

$$L = \sqrt{H^2 + \left(T\frac{H}{R}\right)^2}$$

Applying a floor-to-floor height will over-ride any length dimension given to a path and is achieved using the <Height> element as shown below:

```
<ConnectionType type="enz_stairs">
        <Width>2.00</Width>
        <Tread>0.305</Tread>
        <Riser>0.165</Riser>
        <Height>3.0</Height>
</ConnectionType>
```

After running the Exercise 7d input file have a look at the `Results.html` file and you will see that the length of the Stairs connection is no longer 10.0 m but 6.3 m to correspond with the 3.0 m floor-to-floor height.

Summary

The important thing to remember when modelling stairs in Evacuationz is that you need to have a node to represent the stair's physical space and a path that accounts for the stair configuration in terms of its width, tread and riser dimensions. If a stair has landings then you should consider modelling those as separate nodes or accounting for them by some other means.

[**Note:** One day I would like to define a special 'stair and landing' node that would simplify the creation of stairs but I have yet to find the time to do it or a willing volunteer to do it for me.]

EXERCISE 8 – PRE-EVACUATION DISTRIBUTION

In this exercise the fixed pre-evacuation time defined in Exercise 3 is replaced with a distribution. The PreEvacuation element has an attribute that instructs Evacuationz to expect a distribution which is then followed by the appropriate XML structure for a Normal distribution.

```
<ENZ_AgentType>
        <AgentTypeDefinition>
                <Speed>1.2</Speed>

                <!-- Specify user-defined pre-evacuation distribution -->
                <PreEvacuation type='enz_distribution'>
                        <Distribution type='enz_normal'>
                                <Mean>50</Mean>
                                <StandardDeviation>30</StandardDeviation>
                        </Distribution>
                </PreEvacuation>

                <StartDistance>0</StartDistance>
        </AgentTypeDefinition>
</ENZ_AgentType>
```

After Evacuationz has run the simulation the pre-evacuation times applied to the agents can be found in the `pre-evac.csv` file. You can use Excel to get the actual mean and standard deviation of the values which should be close to the specified values.

There are many places within Evacuationz where distributions can be used, for example we could apply distributions to the speed and starting distance in the above example. There are a number of in-built distribution functions that can be used for selecting model parameters. Where selecting a value less than 0 is not logical (e.g. a negative pre-evacuation time), distributions will be truncated at zero. The general form of a distribution element is

```
<Distribution type='type'>
        <ParameterElement/s>
</Distribution>
```

The following distributions are available:

Uniform

```
<Distribution type='enz_uniform'>
        <Min>value</Min>
        <Max>value</Max>
</Distribution>
```

Normal

```
<Distribution type='enz_normal'>
        <Mean>value</Mean>
        <StandardDeviation>value</StandardDeviation>
</Distribution>
```

Log-normal

```
<Distribution type='enz_lognormal'>
        <Mean>value</Mean>
        <StandardDeviation>value</StandardDeviation>
</Distribution>
```

Triangular

```
<Distribution type='enz_triangular'>
        <Min>value</Min>
        <Max>value</Max>
        <MostLikely>value</MostLikely>
</Distribution>
```

Weibull

```
<Distribution type='enz_weibull'>
        <Alpha>value</Alpha>
        <Beta>value</Beta>
</Distribution>
```

Fixed

The fixed 'distribution' is mainly included for consistency and completeness.

```
<Distribution type='enz_fixed'>
        <Value>value</Value>
</Distribution>
```

Discrete

This function allows the creation of a distribution from a series of discrete values with associated probabilities of their selection and an optional interval range. The csv value list for each element must have the same number of items and the items in the <Percent> element must add up to 100.

```
<Distribution type='enz_discrete'>
        <Bins>csv value list</Bins>
        <Percent>csv value list</Percent>
        <Intervals>csv value list</Intervals>
</Distribution>
```

Truncation
Any of the above distributions can be forced to have an upper and/or lower truncated limit by including either or both the `<Upper>value</Upper>` and `<Lower>value</Lower>` elements in the parameter list. For example:

```
<Distribution type='enz_weibull'>
        <Alpha>value</Alpha>
        <Beta>value</Beta>
        <Upper>value</Upper>
        <Lower>value</Lower>
</Distribution>
```

EXERCISE 9 – OCCUPANT DENSITY

Part (a) Populating in a Node

This exercise shows a more advanced method to populate nodes using XML embedded in a node. The advantage of doing this over the method shown in the previous Study Guide will become apparent below. The following XML is added to the 'Room' node and the 'Agents' data in the Properties View is set to 0 agents.

```
<PopulationDefinition>
        <Agents>20</Agents>
</PopulationDefinition>
```

It is possible to populate a node by using both the method previously described and the method shown here. In that case the total number of agents in the simulation will be the sum of the two. It is not recommended to use both methods in conjunction as it may confuse users or reviewers.

Part (b) Metres Squared per Agent

This is an exercise is an example of how to specify a population by using an occupant density where the default units are agents/m^2. Since the area of the node is 10 m by 10 m = 100 m^2 then with an occupant density of 0.5 agents/m^2 we will get 50 agents created. To set an occupant density, update the 'Evacuationz XML' data in the 'Room' node to read as follows:

```
<PopulationDefinition>
        <!-- Set the agent density to 0.5 agents/m² -->
        <Density>0.5</Density>
</PopulationDefinition>
```

Part (c) Metres Squared per Agent

Alternatively to above, to use metres squared per agent we need to specifically set the units:

```
<PopulationDefinition>
        <!-- Set the agent density to 4 m²/agent -->
        <Density units="m2/agent">4</Density>
</PopulationDefinition>
```

When using any of the above methods to specify the occupant numbers you need to be aware that nodes have a default maximum occupant density of 2.75 agents/m^2 so if you try to specify greater numbers the excess will be disregarded.

Part (d) Named Density

Another method to set the occupant density is to use the definitions given in a particular document. Currently only the C/VM2 definitions are available as shown below. The definitions are case insensitive and the trailing ‘s’ can usually be omitted so for example ‘Museums’ or ‘Museum’ will both work.

Definition	**Density (m^2/agent)**
Area without seating or aisles	1
Art galleries, museums	4
Art galleries	4
Museums	4
Bar sitting areas	1
Bar standing areas	0.5
Classrooms	2
Consulting rooms	5
Dance floors	0.6
Day care centres	4
Dining, beverage and cafeteria spaces	1.25
Dining spaces	1.25
Beverage spaces	1.25
Cafeteria spaces	1.25
Exhibition areas, trade fairs	1.4
Exhibition areas	1.4
Trade fairs	1.4
Fitness centres	5
Gaming and casino areas	1
Gaming areas	1
Casino areas	1
Indoor games areas, bowling alleys	10
Indoor games areas	10
Bowling alleys	10
Libraries – stack areas	10
Libraries other areas	7
Lobbies and foyers	1
Lobbies	1
Foyers	1
Mall areas used for assembly uses	1
Reading or writing rooms and lounges	2
Reading rooms	2
Writing rooms	2
Lounges	2
Restaurants, dining rooms	1.1
Restaurants	1.1
Dining rooms	1.1
Shop spaces and pedestrian circulation areas including malls and arcades	3
Shop spaces	3
Pedestrian circulation areas	3
Shop spaces for furniture, floor coverings, large appliances, building supplies and Manchester	10

Shop spaces for furniture	10
Shop spaces for floor coverings	10
Shop spaces for large appliances	10
Shop spaces for building supplies	10
Shop spaces for Manchester	10
Showrooms	5
Spaces with loose seating	0.8
Spaces with loose seating and tables	1.1
Sports halls	3
Stadiums and grandstands	0.6
Stadiums	0.6
Grandstands	0.6
Staffrooms and lunch rooms	5
Staffrooms	5
Lunch rooms	5
Stages for theatrical performances	0.8
Standing spaces	0.4
Swimming pools: water surface area	5
Swimming pools: surrounds and seating	3
Teaching laboratories	5
Vocational training rooms in schools	10
Vocational training rooms	10
Aircraft hangars	50
Bulk storage including racks and shelves (warehouses etc)	100
Bulk storage	100
Retail and trading (storage >3.0 m high)	5
Retail (storage >3.0 m high)	5
Trading (storage >3.0 m high)	5
Call centres	7
Commercial laboratories, laundries	10
Commercial laboratories	10
Laundries	10
Computer server rooms	25
Heavy industry	30
Interview rooms	5
Commercial kitchens	10
Manufacturing and process areas	10
Manufacturing areas	10
Process areas	10
Meeting rooms	2.5
Offices	10
Personal service facilities	5
Reception areas	10
Workrooms, workshops	5
Workrooms	5
Workshops	5
Boiler rooms, plant rooms	30
Boiler rooms	30
Plant rooms	30
Parking buildings, garages	50

Parking buildings	50
Garages	5

To use this method in Evacuationz enter:

```
<PopulationDefinition>
        <!-- Set the agent density to C/VM2 definition -->
        <Density type="enz_vm2">Museum</Density>
</PopulationDefinition>
```

EXERCISE 10 – MULTIPLE AGENT TYPES

Part (a) Defining Agent Types

By using the ability to specify a population from within a node we now have a method to create and specify multiple agent types with different characteristics. To expand the agent type definitions, use the following XML in the 'agent type' data in the Properties View:

```
<ENZ_AgentType>

        <AgentTypeDefinition>
                <!-- Specify user-defined agent type name -->
                <Name>my-agent-type-1</Name>

                <Speed>1.2</Speed>
                <PreEvacuation type="enz_distribution">
                        <Distribution type="enz_normal">
                                <Mean>50</Mean>
                                <StandardDeviation>30</StandardDeviation>
                        </Distribution>
                </PreEvacuation>
                <StartDistance>0</StartDistance>
        </AgentTypeDefinition>

        <!-- Specify second user-defined agent type -->
        <AgentTypeDefinition>
                <Name>my-agent-type-2</Name>
                <Speed>1.2</Speed>
                <PreEvacuation>0</PreEvacuation>
                <StartDistance>0</StartDistance>
        </AgentTypeDefinition>

</ENZ_AgentType>
```

The content of the <Name> element in the <AgentTypeDefinition> structure can be specified as any text characters (except for single or double quotes) so long as each name is unique within the input. In the full version of Evacuationz there is no limit to the number of different agent types that can be created although there is a limit in the student version.

The specific type of agent can then be declared in the <PopulationDefinition> element in the following way:

```
<PopulationDefinition>
        <Agents>20</Agents>
        <AgentType>
                <Name>my-agent-type-2</Name>
        </AgentType>
</PopulationDefinition>
```

Part (b) Several Agent Types

If you want to have agents with more than type in a node then you can have multiple <PopulationDefinition> elements in a node such as:

```
<PopulationDefinition>
        <Agents>5</Agents>
        <AgentType>
                <Name>my-agent-type-1</Name>
        </AgentType>
</PopulationDefinition>
<PopulationDefinition>
        <Agents>10</Agents>
        <AgentType>
                <Name>my-agent-type-2</Name>
        </AgentType>
</PopulationDefinition>
```

Part (c) Using Distributions

Just as we saw with pre-evacuation time, we can use a distribution when populating a node. For example we can have:

```
<PopulationDefinition>
        <Agents type="enz_distribution">
                <Distribution type="enz_normal">
                        <Mean>20</Mean>
                        <StandardDeviation>10</StandardDeviation>
                </Distribution>
        </Agents>
        <AgentType>
                <Name>my-agent-type-2</Name>
        </AgentType>
</PopulationDefinition>
```

Similarly, a distribution can be applied to an occupant density instead of an occupant number.

Part (d) Applying Probabilities

By creating multiple agent types and using the method to populate nodes directly it is also possible to apply probabilities to the creation of agents. For example, the following XML is used in the 'Room' node to specify the new population so that there is a 30% chance that an agent will be 'my-agent-type-1' and a 70% chance that it will be 'my-agent-type-2':

```
<PopulationDefinition>
        <Agents>20</Agents>
        <AgentType>
                <Name>my-agent-type-1</Name>
                <Probability>30</Probability>
        </AgentType>
        <AgentType>
                <Name>my-agent-type-2</Name>
                <Probability>70</Probability>
        </AgentType>
</PopulationDefinition>
```

You can assess the number of agents created of each type by looking at the `pre-evac.csv` file since 'my-agent-type-1' agents have a distribution and 'my-agent-type-2' have a 0 s value.

The sum of the probabilities associated with the <AgentType> elements within a <PopulationDefinition> must add up to 100% otherwise Evacuationz will raise an error.

EXERCISE 11 – AGENT LOGS

Once we start creating multiple agent types and populating multiple nodes it is often useful to get more detailed information on the agents. This example shows how we can obtain a detailed log file of each agent that is included in a simulation. The 'scenario' data in the Properties View is modified as follows:

```
<EvacuatioNZ_Scenario>
    <Files>
        <Results />
        <Evacuation />
        <Nodes />
        <PreEvacuation />

        <!-- Add agent detailed log -->
        <Agents log="enz_true" />

    </Files>
</EvacuatioNZ_Scenario>
```

Update the 'Evacuationz XML' data in the 'Room' node to read as follows:

```
<!--Enable logging for this population-->
<PopulationDefinition log="enz_true">
    <Density>0.5</Density>
    <AgentType>
        <Name>my-agent-type-1</Name>
        <Probability>30</Probability>
    </AgentType>
    <AgentType>
        <Name>my-agent-type-2</Name>
        <Probability>70</Probability>
    </AgentType>
</PopulationDefinition>
```

After Evacuationz has finished an HTML file will have been created for each agent that can be opened in a browser window. If you want to combine all of the individual HTML files into a single file then add the combined="enz_true" attribute to the <Agents> element:

```
<Agents log="enz_true" combined="enz_true" />
```

EXERCISE 12 – OCCUPANT DISTRIBUTIONS

Part (a) Using the project Properties View

We have already seen two methods which can be used to specify the number of occupants in a node: either using the Data field in yEd or by adding some XML to the 'Evacuationz XML' field in a node. The third method is to insert XML into the 'populate' field in the project Properties View field.

```
<ENZ_Populate>
        <PopulationDefinition>
                <Agents>20</Agents>
                <NodeRef>Room</NodeRef>
                <AgentType>
                        <Name>my-agent-type-1</Name>
                </AgentType>
        </PopulationDefinition>
</ENZ_Populate>
```

Here we need to specify the name of the node to populate using the <NodeRef> element.

Part (b) Applying distributions

Just the same as using a <PopulationDefinition> element in a node we can also apply distributions such as the in the following:

```
<ENZ_Populate>
        <PopulationDefinition>
                <Agents type="enz_distribution">
                        <Distribution type="enz_normal">
                                <Mean>20</Mean>
                                <StandardDeviation>10</StandardDeviation>
                        </Distribution>
                </Agents>
                <NodeRef>Room</NodeRef>
                <AgentType>
                        <Name>my-agent-type-1</Name>
                </AgentType>
        </PopulationDefinition>
</ENZ_Populate>
```

Part (c) Spreading agent populations across nodes

As we have noted in the discussion regarding the location of occupants in a building at a given point in time, we might want to be able to assess the effect of having agents spread across different nodes. Evacuationz has a command that allows the user to spread agents across two or more specified nodes. The example below shows we can spread a specified number of agents across two nodes:

```
<ENZ_Populate>
        <PopulationDefinition>
                <Agents>20</Agents>

                <!-- Spread 20 agents across the two nodes -->
                <NodeRef type="enz_spread">Room 1,Room 2</NodeRef>

                <AgentType>
                        <Name>my-agent-type-1</Name>
                </AgentType>
        </PopulationDefinition>

</ENZ_Populate>
```

Note how we can use spaces in the names of the nodes but be careful not to add extra leading and/or trailing spaces. For example, if we had used `<NodeRef type="enz_spread">Room 1, Room 2</NodeRef>` then an error would occur as Evacuationz will search for a node called Room 2 with a leading space.

Part (d) Populating a range of nodes

Evacuationz also has a command that allows the user to place agents into a range of nodes and the example below shows how to set this up:

```
<ENZ_Populate>

        <PopulationDefinition>
                <Agents>5</Agents>

                <!-- Place 5 agents in each of the two nodes -->
                <NodeRef type="enz_range">Room 1,Room 2</NodeRef>

                <AgentType>
                        <Name>my-agent-type-2</Name>
                </AgentType>
        </PopulationDefinition>

</ENZ_Populate>
```

Part (e) Complex populations

Using all of the population methods described above means we can create complex definitions that combine the use of a distribution for the number of agents, spreading the agents across nodes and assigning different agent types with probabilities. Have a look at the XML script below and consider what this will do when it is run.

```
<ENZ_Populate>
    <PopulationDefinition>
        <Agents type="enz_distribution">
            <Distribution type="enz_normal">
                <Mean>20</Mean>
                <StandardDeviation>10</StandardDeviation>
            </Distribution>
        </Agents>
        <NodeRef type="enz_spread">Room 1,Room 2</NodeRef>
        <AgentType>
            <Name>my-agent-type-1</Name>
            <Probability>30</Probability>
        </AgentType>
        <AgentType>
            <Name>my-agent-type-2</Name>
            <Probability>70</Probability>
        </AgentType>
    </PopulationDefinition>
</ENZ_Populate>
```

Just be aware that mixing the three different population methods (i.e., the simple property approach, using in-node <PopulationDefinition> elements and using the <ENZ_Populate> element) is not recommended – in any one project stick to one way or you (or Evacuationz) are likely to get confused.

EXERCISE 13 – MERGING FLOWS

Part (a) Default Behaviour

For the first exercise we will see how Evacuationz performs without explicitly accounting for merging. After you have run the exercise plot the results from the nodes file to examine the number of agents in the two room nodes. You should see that agents flow at an equal rate from both nodes i.e., equivalent to a 50:50 merge.

Part (b) Controlling Merge Rates

The following method can be used to control the merge rate into a node through a group of connections. The `<Merge>` element can be added to the connection XML as follows:

```
<Merge>1</Merge>
<ConnectionType type="enz_door">
        <Width>5</Width>
</ConnectionType>
```

By placing the Merge element on several connections leading to a node with different merge values the flow through each connection is the ratio of the individual values. In Exercise 13b we have two connections with a merge value of 1 on ‘path 1’ and a value of 3 on ‘path 2’. After running the exercise plot the number of agents in the two room nodes. You should now see that the agents leave Room 2 faster than from Room 1. When all 500 agents have left Room 2 there are still 319 agents in Room 1, i.e. 500 – 319 = 181 have exited so the relative flow rate is 500:181 or 2.8:1 which (reasonably) matches our desired ratio of 3:1. You will also note that once all of the agents have left Room 2 then the flow from Room 1 increases since no more merging is taking place.

The merging can be set across more than two connections from more than two nodes with different merge values on each connection. This means it is possible to create very complex arrangements of nodes, connections, agents and merge values. A word of warning, creating such complex situations could reach a point that Evacuationz might end up showing some unexpected outcomes.

EXERCISE 14 – ADVANCED AGENT TYPES

Part (a) Additional Settings

We can use the following XML in the 'agent type' data to set the gender (sex), age, BMI and how these affect the maximum walking speed of the agents:

```
<ENZ_AgentType>

        <AgentTypeDefinition>
                <Name>my-agent-type-3</Name>

                <PreEvacuation>0</PreEvacuation>

                <!-- Set speed using Ando’s data for age and gender -->
                <Gender type="enz_female">100</Gender>
                <Age>63</Age>
                <Speed model="enz_ando" />

                <!-- Set body mass index -->
                <BMI>40</BMI>

                <!-- Set agent’s height -->
                <Height>1.75</Height>

        </AgentTypeDefinition>
</ENZ_AgentType>
```

The number associated with the Gender element is the probability that gender is selected. Distribution functions can be used with the age and BMI elements. Previously we set the maximum speed of agents with a fixed value or we could also apply a distribution. By using the <Speed model="enz_ando" /> element EvacuatioNZ determines the agent speed using their age, gender and the relationships given by Ando. If a BMI value is applied to an agent then their maximum speed is further reduced using the relationship previously shown in Spearpoint and MacLennan.

You will also notice that it is possible to set the height of the agent although this currently has no effect on the model’s calculations.

We can get an extended pre-evacuation output file by setting the type attribute in the scenario data as well as using the individual agent log to verify our settings

```
<ENZ_Scenario>
        <Files>
                <Results />
                <PreEvacuation type="enz_full" />
                <Agents log="enz_true" />
        </Files>
</ENZ_Scenario>
```

Run the exercise and open the `pre_evac.csv` file and will see some additional information regarding the agent. Similarly, you can open the individual agent log files to view the effect of the settings.

Part (b) Start Distance

We will now return to the further examine the StartDistance element. The start distance capability allows us to define an additional distance to the travel distance in the node that the agent starts in. The element also has several pre-defined options available where the enz_disperse attribute randomly disperses the agents around the node. The other options are: enz_minimum where the distance is 0 m, enz_maximum where the distance is the sum of the node length and width (i.e., the default setting) and enz_distribution where a distribution can be defined by the user.

```
<ENZ_AgentType>

        <AgentTypeDefinition>

                <!-- Use one of the pre-defined StartDistance options -->
                <StartDistance type="enz_disperse" />

                ...other elements...

        </AgentTypeDefinition>
</ENZ_AgentType>
```

The disperse function randomly sets start distances in the range $\pm\sqrt{(L/2)^2 + (W/2)^2}$ where *L* and *W* are the dimensions of the node. This function is particularly useful when coupled with path lengths that are obtained when a value of 0.0 is defined in yEd on node and path dimensions. The two algorithms simulate agents dispersed around their starting node.

We can see how the start distance affects movement by going back to Exercise 6 on congested movement. If we set <StartDistance type="enz_disperse"/> then the total egress time reduces from 22.5 s to 12.5 s because the agents have different starting distances from the connection. This means they that leave the node at different times which in turn reduces the occupant density and hence those agents that remain move faster.

Part (c) Individuals

So far we have created groups of agents from a given agent type definition but in some cases we might want to create individual agents with specified characteristics. The example below shows how we can override the base agent type settings with specific values:

```
<PopulationDefinition log="enz_true"  type="enz_individual">

        <!-- Set base agent type -->
        <AgentType>my-agent-type-3</AgentType>

        <!--Override base settings -->
        <Name>Mike</Name>
        <StartDistance>4.5</StartDistance >
        <Speed>1.75</Speed>
        <PreEvacuation>20</PreEvacuation>
        <Gender type="enz_male" />
        <Age>50</Age>
        <BMI>22</BMI>

</PopulationDefinition>
```

Any characteristic that is not specifically defined will be taken from the agent type definition. The list in the above example all are the characteristics that can be individually specified at the moment. Only fixed values for parameters can be defined in an individual not calculated / probability-based values.

After running the exercise open the agent log file and you will see that the name of the agent will be given in the title and the initialisation description will show that the individual settings have been applied.

If you want to use the <Speed model="enz_ando" /> then this needs to be defined in the base agent type definition not where you create the single agent as shown below:

```
<ENZ_AgentType>
        <AgentTypeDefinition>
                <Name>my-agent-type-3</Name>
                <Speed model="enz_ando" />
        </AgentTypeDefinition>
</ENZ_AgentType>
```

and then

```
<PopulationDefinition log="enz_true"  type="enz_individual">

        <!-- Set base agent type -->
        <AgentType>my-agent-type-3</AgentType>

        <!--Override base settings -->
        ...other desired settings such as age, gender, etc...

</PopulationDefinition>
```

If the specific age and gender is set then the Ando model will be applied using those as input.

EXERCISE 15 – ADVANCED SCENARIO SETTINGS

The exercise shows how to set up additional scenario settings by adding XML instructions to the 'scenario' data in the Properties View. We can use the `<Simulations>` element to tell Evacuationz to run the input multiple times.

The pseudo-random number generator can be seeded with a user-defined value so that the same series of random numbers are generated each time. If no seed is specified or the seed is set to 0 then the start of the random number sequence is effectively random.

The instructions get Evacuationz to generate four different output files in various folders. The **`%MyDocsFolder%`** places files in the My Documents folder, the **`%InputFolder%`** places files in the same folder as the original input file and the `<RootPath>` element can be used to create a user-defined folder path that can be called up using the **`%ROOT%`** variable. Before you run the exercise make sure you have a folder called **`C:\tmp`**.

```
<ENZ_Scenario>

        <!-- Specify the number of simulations to execute -->
        <Simulations>2</Simulations>

        <!-- Seed the random number generator -->
        <Seed>1</Seed>

        <Files>
                <!-- Send output file to specified path -->
                <PreEvacuation>c:\tmp\my-pre-evac.csv</PreEvacuation>

                <!-- Send output files to special folders -->
                <Evacuation>%MyDocsFolder%\my-evac.csv</Evacuation>
                <Results>%InputFolder%\my-results.html</Results>

                <!-- Send output files to user-defined folder -->
                <RootPath>C:\tmp</RootPath>
                <Nodes minmax="enz_true" safe="enz_true">
                        %ROOT%\my-nodes.csv</Nodes>
        </Files>
</ENZ_Scenario>
```

Using the log file

Since we have now placed files in various folders we will have to navigate to them. However, recall we can also use the `log.html` file to open the files directly.

Node output files

When we look at the nodes output we will find that two sequentially numbered files have been created, one for each simulation. Only a single pre-evacuation file is created by Evacuationz for the last simulation that was run.

The safe attribute associated with the <Nodes> element generates an additional output file that gives the number of agents that are in each safe node at each time step split by their agent types.

The minmax attribute associated with the <Nodes> element generates an additional output file that lists the minimum and maximum number of agents in each node for each time step across all simulations. It also lists the minimum and maximum total number of agents that have reached all safe nodes for each time step. This file is gives a useful summary of your simulations without having to process each individual node output file.

EXERCISE 16 – SIMULATION SETTINGS

Part (a) Times

We can make changes to the simulation time settings by creating XML instructions in the 'simulation' data section of the Properties View.

```
<ENZ_Simulation>
        <TimeStep>1</TimeStep>
        <TimeMax>10</TimeMax>
</ENZ_Simulation>
```

Here we have changed the simulation time step to 1 s and the maximum simulation time to 10 s. When you open the `evac.csv` file you will find that the final evacuation time is 10 s since not all of the agents have reached a safe node by the time the simulation terminated. Similarly the final time is the `nodes.csv` and in the `results.html` files are 10 s.

Part (b) Agent Processing

When Evacuationz runs it needs to process each agent during the period of a single time step. By default, the agents are processed in the same sequential order each time step and the sequence is determined by the order the agents were originally created. This means the first agent processed always gets to move first however we might want the change processing order to get more randomness into the simulations.

The order of agent processing can be changed by adding the agent_process to the <ENZ_Simulation> element and there are three possible values for the attribute. The enz_sequential value means the same agent sequence is used each time step, i.e., the default behaviour. The enz_random value means the order that the agents are processed is random each time step. Finally, the enz_randon_once value means that the order of the agents is randomised at the start of a simulation and then this same order is then used for the remainder of the simulation.

```
<ENZ_Simulation agent_process='enz_random'>
        ...other simulation elements...
</ENZ_Simulation>
```

If you go back to Exercise 6 and run it several times you will find that in the `results.html` file the last agent to leave Room 1 is always Agent #1. Now if we add the above XML to the simulation as in Exercise 16b then you will find that the last agent is always different since the agent order is changed each time step.

Part (c) Sampling Method

In addition to setting the agent processing, the sampling method used by Evacuationz can be specified using the sampling attribute. The attribute can be set to enz_monte_carlo (i.e., the default if no sampling attribute is given) or to enz_stratified, e.g.

```
<ENZ_Simulation sampling='enz_monte_carlo'>
```

or

```
<ENZ_Simulation sampling='enz_stratified'>
```

Application of the stratified sampling can be used on the fixed, discrete or uniform distributions although it has no effect on the outcome. By default the number of strata is set to 10. Evacuationz has other ways in which to conduct and control sampling but this is beyond the scope of this guide.

Part (d) Counterflow

To do…

```
<ENZ_Simulation counterflow='enz_true'>
```

EXERCISE 17 – YED OUTPUT

yEd is not only a way to create input but can also be used to generate simple graphical output.

Part (a) Nodes and Paths

This example shows how yEd can be used to view node and connection properties. The 'scenario' data in the Properties View is modified as follows:

```
<ENZ_Scenario>
        <Files>
                <Results />
                <Evacuation />
                <Nodes />
                <PreEvacuation />
        </Files>

        <!-- Add yEd output -->
        <yEd>
                <ResultDiagram displayDimensions.Text="enz_true" />
        </yEd>

</ENZ_Scenario>
```

When the displayDimensions.Text attribute is set to enz_true the node and connection details are written into the appropriate yEd labels to allow the user to check their geometry. Details are only provided for yEd elements that have a visible text label. The details are only written to the output file generated at time equals zero.

Part (b) Agents

We can also use the yEd output to visualise the agent movement by setting the 'scenario' data in the Properties View is modified as follows:

```
<ENZ_Scenario>
        <Files>
                <Results />
                <Evacuation />
                <Nodes />
                <PreEvacuation />
        </Files>
        <!-- Add yEd output -->
        <yEd>
                <ResultDiagram
                        frequency="30"
                        displayDensity="enz_true"
                        displayDensity.Text="enz_true"
                        displayPercentage.Text="enz_true"
                        displayAgents.Text="enz_true" />
        </yEd>
</ENZ_Scenario>
```

Each attribute does the following:

frequency : This attribute sets the simulation time frequency for each output file where the default value is every 10 s. If the value is set to 0 then the only file generated will be at the 0 s time step. A yEd output file is also created at the last time step irrespective of the frequency.

displayPercentage.Text : When set to enz_true this attribute displays the percentage of the total number of agents in each node at the given time step. The total percentage of agents that have reached a 'safe' node is displayed in the progress bar label.

displayDensity : When set to enz_true this attribute shades nodes as a function of the agent density at each time step where darker shades indicate a higher density. A node is not shaded if the fill has no colour.

displayDensity.Text : When set to enz_true this attribute displays the agent density in each node at the given time step.

displayAgents.Text : When set to enz_true this attribute displays the number of agents in each node at the given time step. The total number of agents that have reached a 'safe' node is displayed in the progress bar label.

displayAgents : When set to enz_true this attribute displays icons in each node to represent individual agents. The simplest approach to tracking individual agents is to create a single icon in yEd that will be associated with all of the agents. In the yEd

In the yEd Evacuationz template attach an icon to the 'progress timer' bar using the steps given below:

- Select the 'progress timer' bar.
- Right-click and from the menu select Add label.
- Type in the name of the label and press Return.
- Select the label that has just been created and from the Properties View window select Icon Properties.

- Here the icon can be selected and its size, position etc. specified.
- The Properties View can also be used to specify other properties associated with the label and it is often a good idea to set the Placement to 'free' so that the label can be positioned anywhere that is convenient.

Individual agents can be tracked by their agent type so that different icons can be used for each agent type. To ensure a particular agent type is tracked, the AgentTypeDefinition element must have the <yEdIcon> attribute set to enz_true:

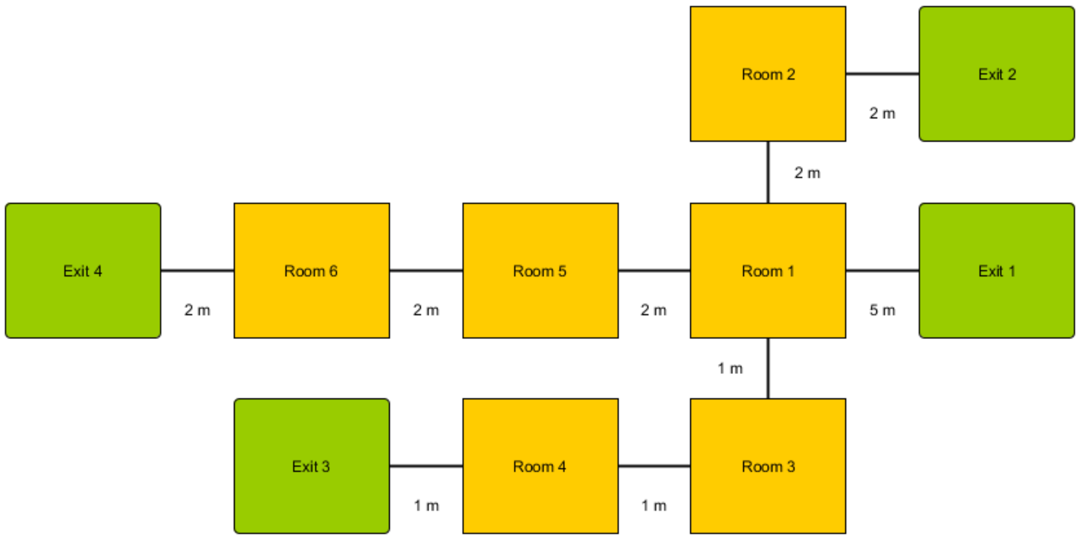
```
<AgentTypeDefinition yEdIcon="enz_true"/>
        <Name>Use same name for the yEdIcon text</Name>
</AgentTypeDefinition>
```

Part (c) Creating Animations

We can create simple animated GIF files using the yEd output and some other free tools from the web. Use the File > Export menu to save several .gif images. Then you can use a tool such as UnFREEz 2.1 to stich the individual gif files into an animated file that can be viewed in a browser.

EXERCISE 18 – EXIT BEHAVIOURS

Evacuationz uses the term ‘exit’ behaviour but really it might have been better to have called it route selection behaviour or something similar. For the following set of exercises to demonstrate the exit behaviour capabilities in Evacuationz we will use variations of the node map shown below. The green coloured nodes are all designated ‘safe’ nodes.



We are not interested in the movement mechanics of the agents other than the distances they can potentially travel and these distances are marked on the diagram. In general, we will only simulate a single agent in the exercises that starts in Room 1.

Part (a) Minimum Total Path Travel

There are two steps to associating an exit behaviour with agents. The first is to define the behaviour type and then to link that behaviour to an agent type. For the first example we will look at the current default behaviour in which agents follow the shortest total travel distance to a safe node.

We can specifically create the behaviour using the XML example below. The user can define the name as any unique text avoiding the use of quotation marks, commas and other characters that might confuse the software.

```
<ENZ_ExitBehaviour>
        <ExitBehaviourDefinition>
                <Name>MinDistance</Name>
                <ExitBehaviourType type='enz_min_distance_to_safe' />
        </ExitBehaviourDefinition>
</ENZ_ExitBehaviour>
```

The exit behaviour is then linked to an agent type as illustrated below.

```
<ENZ_AgentType>
        <AgentTypeDefinition>
                <Speed>1.2</Speed>
                <PreEvacuation>0</PreEvacuation>
                <StartDistance>0</StartDistance>
                <ExitBehaviour>MinDistance</ExitBehaviour>
        </AgentTypeDefinition>
</ENZ_AgentType>
```

By opening up the agent log file we can see that the exit behaviour has been appropriately set during the initialisation and that as the agent progresses through the simulation how the behaviour is applied as they pass between nodes. We can also see from the node map that Exit 3 has the least total path distance of all of the safe node options.

Part (b) Minimum Node Travel

The second behaviour to examine is one in which agents travel the least number of nodes to a safe node. By running the exercise we see that the agent travels to Exit 1 as expected.

```
<ENZ_ExitBehaviour>
        <ExitBehaviourDefinition>
                <Name>MinNodes</Name>
                <ExitBehaviourType type='enz_min_nodes_to_safe' />
        </ExitBehaviourDefinition>
</ENZ_ExitBehaviour>
```

Part (c) Minimum Neighbouring Path Travel

The first behaviour we looked at used the total travel distance to a safe node whereas here the agent selects the path that has the shortest distance to a neighbouring node.

```
<ENZ_ExitBehaviour>
        <ExitBehaviourDefinition>
                <Name>MinPath</Name>
                <ExitBehaviourType type='enz_shortest_path_to_next_node' />
        </ExitBehaviourDefinition>
</ENZ_ExitBehaviour>
```

When the agent selects the path from Room 1 it chooses to go to Room 3. From Room 3 it goes to Room 4 as the software checks that an agent will not go back to a node it has just come from (unless there is no other option). Then from Room 3 it goes to Exit 3.

Part (d) Minimum Total Path Travel to Specified Safe Node

With this behaviour we can designate a specific safe node (in this case Exit 2) that the agent is to travel to, using the minimum travel distance.

```
<ENZ_ExitBehaviour>
        <ExitBehaviourDefinition>
                <Name>Specified</Name>
                <ExitBehaviourType type='enz_min_distance_to_specified'>
                        <NodeRef>Exit 2</NodeRef>
                </ExitBehaviourType>
        </ExitBehaviourDefinition>
</ENZ_ExitBehaviour>
```

Part (e) Narrowest Final Exit

This exit behaviour directs agents to the safe node that has the narrowest final path constriction width. When we examine the door widths on the paths leading to the safe nodes the narrowest is that leading to Exit 4.

```
<ENZ_ExitBehaviour>
        <ExitBehaviourDefinition>
                <Name>Narrowest</Name>
                <ExitBehaviourType type='enz_narrowest_exit'/>
        </ExitBehaviourDefinition>
</ENZ_ExitBehaviour>
```

If two safe nodes have the same narrowest width then agents travel to the one with the shortest travel distance.

Part (f) Maximised Path Distances

This exit behaviour determines which node 'maximises' the distance for agents to reach any safe node and creates exit paths that correspond to that node. It is fairly complex to follow how this behaviour works and so it is best illustrated by example.

```
<ENZ_ExitBehaviour>
        <ExitBehaviourDefinition>
                <Name>Conservative</Name>
                <ExitBehaviourType type='enz_maximised_path_distances'/>
        </ExitBehaviourDefinition>
</ENZ_ExitBehaviour>
```

First the algorithm determines which node has the greatest total travel distances from all of the available safe nodes. So for our example map we have the following total distance from the sum of the travel for each room and the four safe nodes:

Room 1 : 5 + 4 + 3 + 6 = 15
Room 2 : 7 + 2 + 5 + 8 = 17
Room 3 : 6 + 5 + 2 + 7 = 20
Room 4 : 7 + 6 + 1 + 8 = 22
Room 5 : 7 + 6 + 5 + 4 = 22
Room 6 : 9 + 8 + 7 + 2 = 26

Thus, we can see that Room 6 is the node that is at the 'maximum' location with the route to Exit 1 being 9 m, Exit 2 at 8 m etc. Once the node with the greatest cumulative score has been found the algorithm assigns scores along each path with respect to Room 6:

Room 6 and Exit 4 = 2 with no more subsequent nodes in this direction
Room 6 and Room 5 = 2
Room 5 and Room 1 = 2 (i.e. the previous Room 6 to Room 5) + 2 = 4
Room 1 and Room 2 = 4 + 2 = 6
Room 2 and Exit 2 = 6 + 2 = 8
Room 1 and Exit 1 = 4 + 5 = 9
Room 1 and Room 3 = 4 + 1 = 5
Room 3 and Room 4 = 5 + 1 = 6
Room 4 and Exit 3 = 6 + 1 = 7

Once the scores along each path have been determined then agents move along the paths with the highest value to get to their safe node. Thus:

An agent starting in Room 1:
Room 1 and Room 2 = 6; Room 1 and Exit 1 = 8; Room 1 and Room 3 = 4 so agent goes to Exit 1
An agent starting in Room 2:
Room 2 and Exit 2 = 8; Room 2 and Room 1 = 6 so agent goes to Exit 2
An agent starting in Room 3:
Room 3 and Room 1 = 4; Room 3 and Room 4 = 5 so agent goes to Room 4 and then on to Exit 3

The purpose of this exit behaviour is to try to create a reasonably conservative analysis of the scenario but avoiding agent counter-flows were they all to just to try to move to their most remote exits.

Part (g) Random Choices

The previous exit behaviours are all deterministic in that the user knows which exit will be used by an agent each time a simulation is run. Evacuationz also has the option to get agents to randomly select paths and safe nodes.

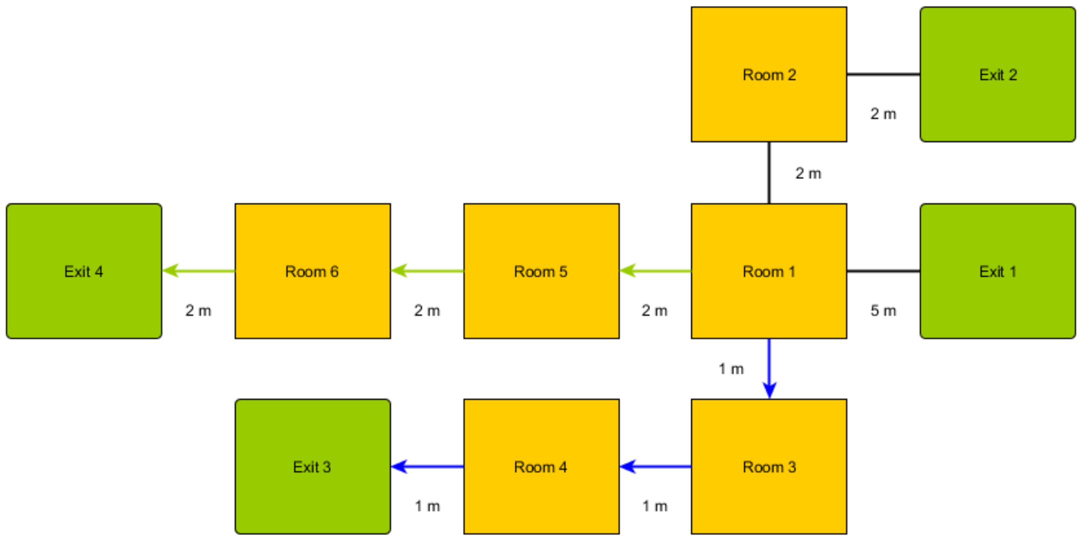
```
<ENZ_ExitBehaviour>
```

```
        <!-- Randomly choose next node -->
        <ExitBehaviourDefinition>
                <Name>Random</Name>
                <ExitBehaviourType type='enz_random' />
        </ExitBehaviourDefinition>

        <!-- Randomly choose next node but do not allow backtracking -->
        <ExitBehaviourDefinition>
                <Name>RandomNoBackTrack</Name>
                <ExitBehaviourType type='enz_nobacktrack_random' />
        </ExitBehaviourDefinition>

        <!-- Randomly select a safe node that all agents travel to -->
        <ExitBehaviourDefinition>
                <Name>RandomSafe</Name>
                <ExitBehaviourType type='enz_random_single_safe' />
        </ExitBehaviourDefinition>

    </ENZ_ExitBehaviour>
```

Randomly selecting the choice of the next node is probably not particularly useful however the ability to have all agents randomly select a single safe node might be useful as part of a sensitivity study. You might need to run this exercise a few times and look at the combined log file to see how each exit behaviour performs.

Part (h) Exit Sign and Preferred Routes

We can also set up specific paths that represent those marked by exit signs and/or preferred routes using coloured paths and arrows. The yEd arrowhead must be the first one in the drop-down list. For an exit route paths are coloured green (colour #99cc00 in yEd), preferred paths are blue (#0000ff).



Using the exit behaviours below will get agents to travel along either of the two paths.

```
<ENZ_ExitBehaviour>

        <!-- Follow a route marked with exit signs -->
        <ExitBehaviourDefinition>
                <Name>ExitRoute</Name>
                <ExitBehaviourType type='enz_exit_sign' />
        </ExitBehaviourDefinition>

        <!-- Follow a specified preferred route -->
        <ExitBehaviourDefinition>
                <Name>PreferredRoute</Name>
                <ExitBehaviourType type='enz_preferred' />
        </ExitBehaviourDefinition>

</ENZ_ExitBehaviour>
```

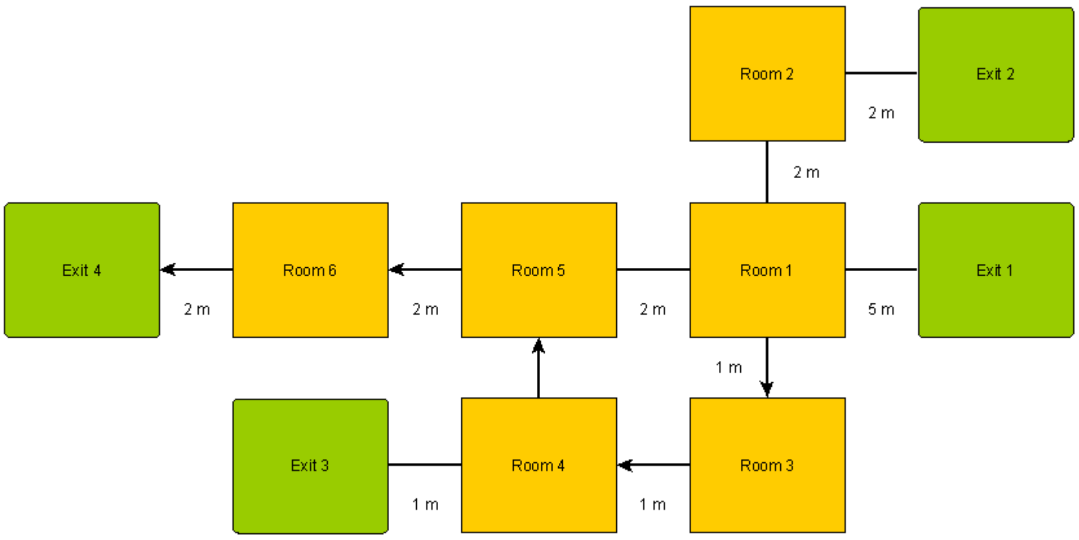
If you want a path to be both an exit route and a preferred route then the paths are coloured purple (#800080).

Part (i) Required Routes

We can force agents to follow a 'required' path by using the enz_required attribute. The yEd arrowhead must be the first one in the drop-down list and the paths are coloured black.

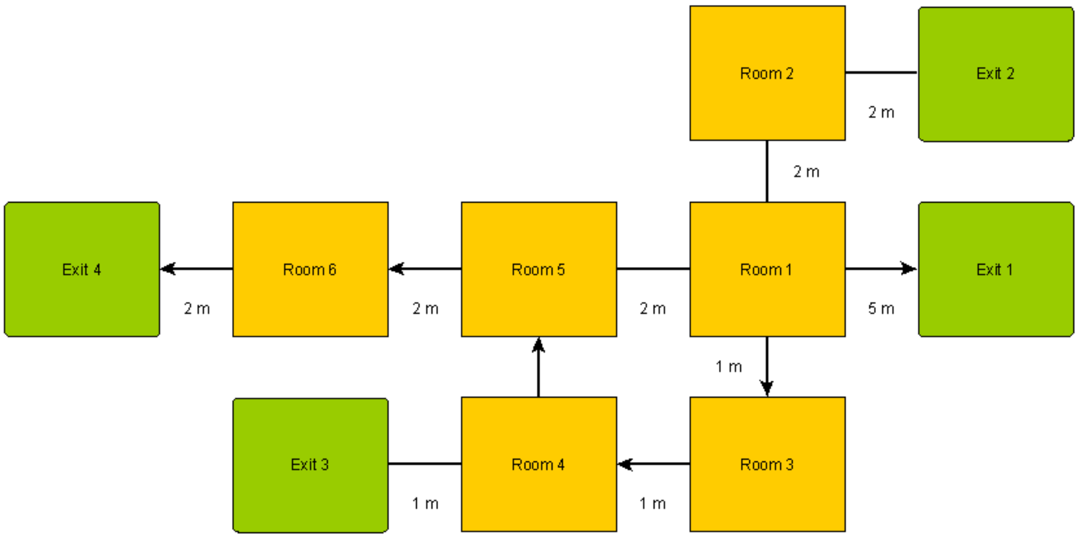
```
<ENZ_ExitBehaviour>
        <ExitBehaviourDefinition>
                <Name>Required</Name>
                <ExitBehaviourType type='enz_required' />
        </ExitBehaviourDefinition>
</ENZ_ExitBehaviour>
```

For the revised map below we can see how an agent starting in Room 1 is forced to travel through Room 3 and Room 4 etc. before reaching Exit 4.



Part (j) Weighted Routes

If two or more required paths lead out from a node then Evacuationz will randomly select one of them with an equal probability. If we take the map from the previous exercise and add another required route from Room 1 to Exit 1 and place 10 agents in Room 1 then about a half end up at Exit 4 and half at Exit 1.



The probability of selecting a required path can be adjusted by using the <Weighting> element in the path XML.

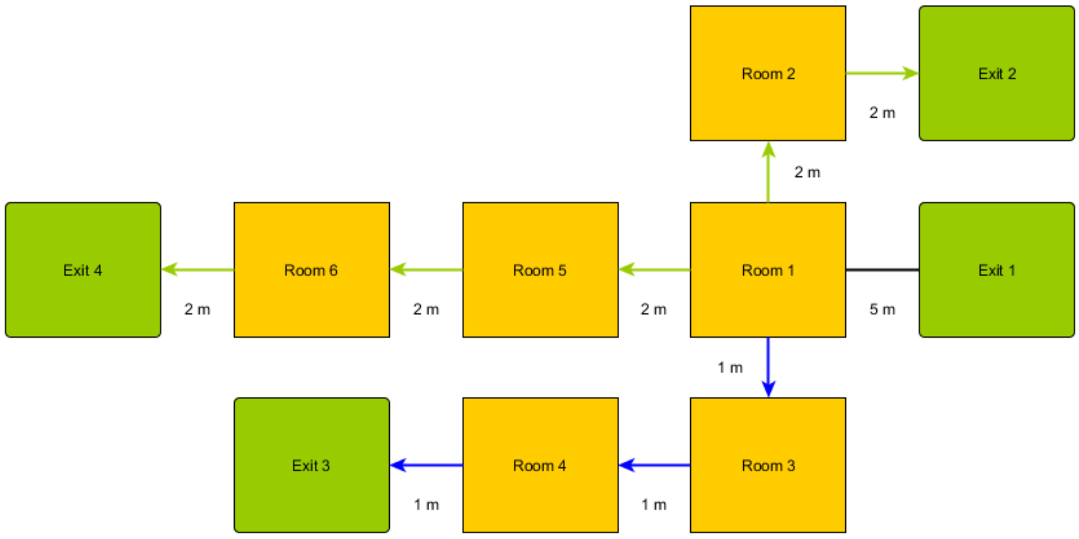
```
<Weighting>my-value</Weighting>
```

By placing weightings on several paths Evacuationz proportionally distributes agents across the path options. The absolute values used in the weighting do not matter as they provide proportions to the total leading from the node although it is sensible to have the total add up to 100 so as to effectively give percentages.

If we add a weighting of 10 on the path to Exit 1 and a weighting of 90 on the path to Room 3 then on average 1 agent goes to Exit 1 and the remainder go to Exit 4.

Part (k) Subtypes

Here we modify the map used in part (h) and make the paths to both Exit 2 and Exit 4 as exit sign routes. Now when an agent leaves Room 1 there are two options that are equal (i.e., travel to Room 2 or Room 5) so it randomly chooses one.



However, we can add an exit behaviour subtype to a behaviour that is employed as a secondary decision on the primary one as shown below:

```
<ENZ_ExitBehaviour>
        <ExitBehaviourDefinition>
                <Name>ExitRoute-2</Name>
                <ExitBehaviourType type='enz_exit_sign'
                        subtype='enz_min_distance_to_safe' />
        </ExitBehaviourDefinition>
</ENZ_ExitBehaviour>
```

What this behaviour does is first to check for any exit sign routes and if there are more than one then it selects the one which has the minimum travel distance to a safe node. Here the agent travels to Exit 2 as this is the exit sign route with the shortest travel distance.

Part (l) Behaviour Lists

We now examine what happens if agents have an exit behaviour, but no path is available. Following on from the previous exercise, if we gave an agent a required behaviour it would not know what path to select since there are none leading from Room 1. Instead we can create a group (or list) of behaviours which are then checked in order. In the example below we have created two separate behaviours similar to what we have used before and then created a group with the two behaviours. Finally, we have created a new exit behaviour that refers to the group. So here the agent first checks whether there is a required path and if not, the agent chooses the path with the fire exit defined with a further check on the path with the shortest travel distance.

```
<ENZ_ExitBehaviour>
        <!-- Follow a route marked with exit signs with minimum distance -->
        <ExitBehaviourDefinition>
                <Name>ExitRoute</Name>
                <ExitBehaviourType type='enz_exit_sign'
                        subtype='enz_min_distance_to_safe' />
        </ExitBehaviourDefinition>

        <!-- Minimum path travel to an exit -->
        <ExitBehaviourDefinition>
                <Name>Required</Name>
                <ExitBehaviourType type='enz_required' />
        </ExitBehaviourDefinition>

        <!-- Create a group using the two definitions above -->
        <ExitBehaviourGroup>
                <Name>my-group</Name>
                <ExitBehaviourName>Required</ExitBehaviourName>
                <ExitBehaviourName>ExitRoute</ExitBehaviourName>
        </ExitBehaviourGroup>

        <!-- Create a definition that is the group -->
        <ExitBehaviourDefinition>
                <Name>ExitRouteGroup</Name>
                <ExitBehaviourType type='enz_group'>
                        <GroupName>my-group</GroupName>
                </ExitBehaviourType>
        </ExitBehaviourDefinition>
</ENZ_ExitBehaviour>
```

Creating behaviour lists is quite advanced and so is not something I would expect many users to need to work with.

Part (m) Probabilistic Selection

Here we will examine how we can assign probabilities of selecting exit behaviours to an agent type. An example is shown below:

```
<ENZ_ExitBehaviour>
        <ExitBehaviourDefinition>
                <Name>my-exit-behaviour</Name>
                <ExitBehaviourType type='enz_min_nodes_to_safe'>
                        <Probability>60</Probability>
                </ExitBehaviourType>
                <ExitBehaviourType type='enz_preferred'>
                        <Probability>40</Probability>
                </ExitBehaviourType>
        </ExitBehaviourDefinition>
</ENZ_ExitBehaviour>
```

Using the map we had in Exercise (k) and create 100 agents then, as expected, around 60 go to Exit 3 (as this has the shortest number of nodes) and rest go to Exit 4 (as the preferred path).

EXERCISE 19 – REASSESSMENT

In this exercise we will concentrate on allowing agents to reassess their route selection and we will return to how we might modify their pre-evacuation choice in a later exercise.

The exit behaviour of an agent can be set using the <Reassessment> element with an associated range of attributes. The default behaviour is

```
<Reassessment method='enz_none'/>
```

which means an agent makes its exit behaviour choice once at the start of the simulation. We do not need to investigate this any further.

Part (a) Entering a Node

In EvacuatioNZ we can get agents to reassess their egress path each time they enter a new node in the following way:

```
<ENZ_ExitBehaviour>
        <ExitBehaviourDefinition>
                <Name>my-exit-behaviour</Name>
                <Reassessment method='enz_enter_node'/>
                <ExitBehaviourType type='enz_min_distance_to_specified'>
                        <Probability>60</Probability>
                        <NodeRef>Exit 4</NodeRef>
                </ExitBehaviourType>
                <ExitBehaviourType type='enz_preferred'>
                        <Probability>40</Probability>
                </ExitBehaviourType>
        </ExitBehaviourDefinition>
</ENZ_ExitBehaviour>
```

After running the simulation, if we examine the log file for the agent we find that it makes another exit behaviour choice each time it enters a new node.

Part (b) Defined Time

The second type of reassessment allows the user to force the exit type behaviour to be reassessed at a given time for example:

```
<Reassessment method='enz_time_delay'>10</Reassessment>
```

where the value is the specified time. This means that a reassessment is made every 10 s during a simulation. In the example exercise the agent has been given a slow walking speed to ensure the simulation lasts long enough to illustrate the method working.

Part (c) Entering a Node and Defined Time

Next we can combine the previous two methods so that a reassessment occurs both when an agent enters a node or after a given time using:

```
<Reassessment method='enz_enter_node_time_delay'>10</Reassessment>
```

Part (d) Entering a Node and Defined Time with Reset

Finally, it is possible to reset the timer on an agent when they enter a new node using:

```
<Reassessment method='enz_enter_node_time_delay_reset'>10</Reassessment>
```

This means that the next timed reassessment takes place 10 s after the agent first entered a new node and every subsequent 10 s until they enter a new node and the timer is reset.

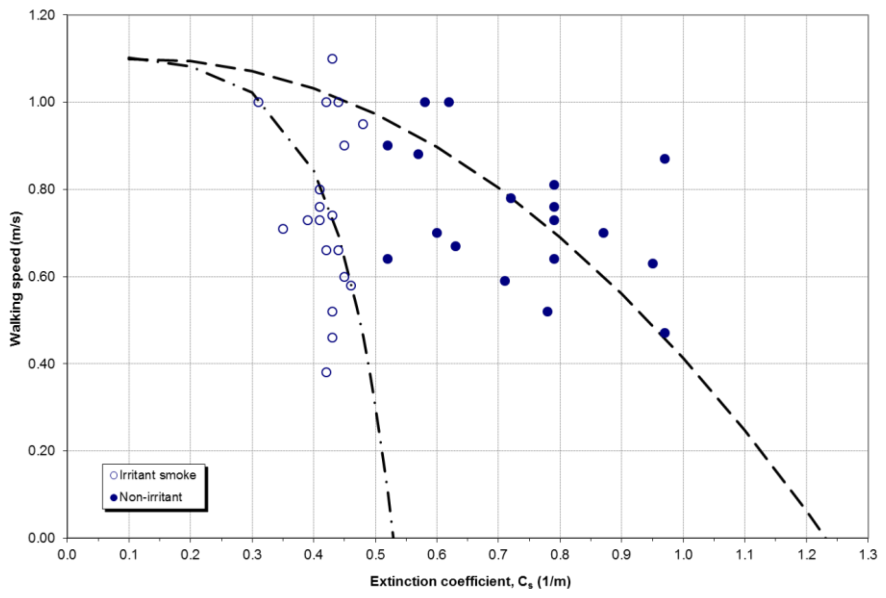
There is even more the user can do with reassessment such as assigning probabilities and using a distribution on the time.

Final Comment

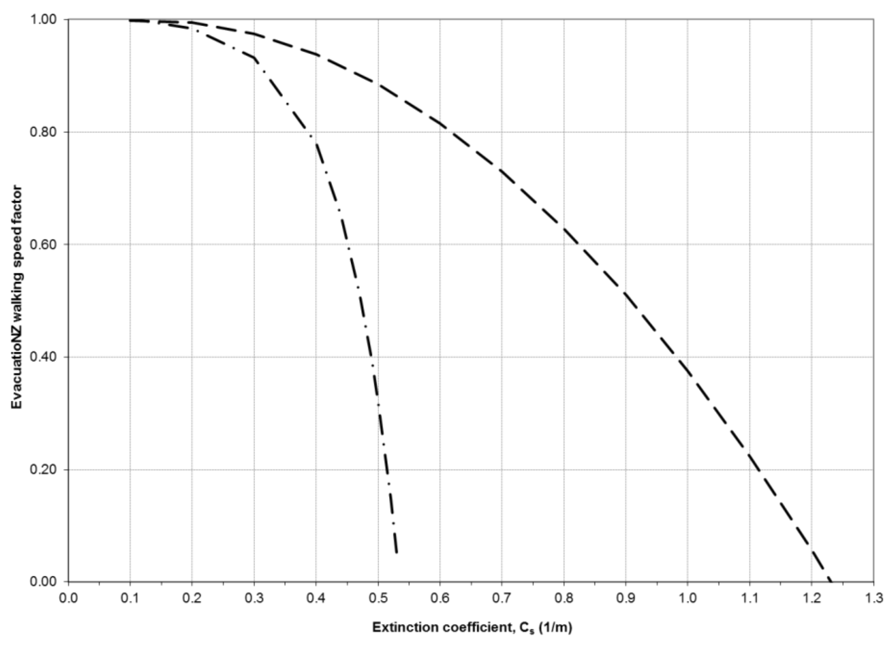
Mixing many exit behaviours, groups, probabilities and reassessment options can make things very complex to the point Evacuationz could do some unexpected things that become very difficult to decipher to the point where not only does the capability of the model overwhelm the capability of the user to understand what it is doing it will also overwhelm the capability of the developer!

EXERCISE 20 – MOVEMENT IN SMOKE

The effect of smoke on walking speed can be incorporated in Evacuationz in which the curves from Jin have been extended down to a walking speed of 0 m/s for both irritating and non-irritating smoke. Any extinction coefficient that exceeds the limit of the curve is assumed to give a walking speed of 0 m/s. This effectively allows for the model to predict fire casualties as at 0 m/s the agent never reaches a safe node.



Evacuationz assumes Jin's curve can be normalised and then an appropriate reduction factor is applied to the agent's uncongested walking speed. Since the walking speed of agents can be defined directly by the user or through using the Ando approach then this normalised reduction factor is applied as the method to account for the effects of smoke.



Part (a) No Smoke

Here we set the path length to 20 m so at a walking speed of 1.2 m/s the travel time is 20 / 1.2 = ~17 s.

Part (b) Jin's Model

We can add some smoke into the node to apply Jin's reduction factor as follows, for example

```
<!-- Set irritant smoke for 20 s -->
<Smoke>
        <Time>0,20</Time>
        <Density model='enz_jin' type='enz_extinction'
                kind='enz_irritant'>0.5,0.0</Density>
</Smoke>
```

This XML tells EvacuatioNZ to use irritant smoke as defined by its extinction coefficient. The <Time> and <Density> elements define the extinction coefficient such that from 0 to 20 s the extinction coefficient is 0.5 m^{-1} and from 20 s to the end of the simulation it is 0 m^{-1}. If the kind attribute is not defined then Evacutionz uses irritant smoke.

If we rerun our scenario from Part (a) the total egress time is 32 s. The agent's speed is reduced to 73% of its maximum for the first 20 s (i.e., 1.2 × 0.27 = 0.32 m/s) so they travel 6.4 m. Then the speed returns to 1.2 m/s so it takes (20 – 6.4) / 1.2 = 12 s and hence the total time is 20 + 12 = 32 s.

Part (c) Reduction Factor

An alternative way to reduce the waking speed is to use a factor directly, for example:

```
<Smoke>
	<Time>0,200</Time>
	<Density type='enz_factor'>0.5,0.0</Density>
</Smoke>
```

So now the agent walks at 0.5 of their maximum speed for 20 s and then 0 times thereafter so they never egress.

Part (d) Layer Height

The previous examples assumed the smoke filled the node but we can account for the layer height by including a set of height values corresponding to each time step, for example:

```
<Smoke>
	<Time>5,10,15</Time>
	<Density type='enz_factor'>0.5,0.5,0.2</Density>
	<Height>2.0,1.7,1.5</Height>
</Smoke>
```

The calculation checks the height of the agent against the smoke layer height and the smoke reduction is only applied if the layer height is less than the agent height. In the example the agent height has been set to 1.6 m so the smoke only has an impact after 15 s.

The default height of an agent is currently taken to be 1.8 m so if no agent height is specified then in this example the smoke would have an impact after 10 s.

EXERCISE 21 – ALARM SYSTEMS

Evacuationz allows for the inclusion of two types of alarm notification systems; namely sounders and voice. To define a system the following is added to the 'scenario' XML data, for example:

```
<ENZ_System>
        <SystemTypeDefinition system="enz_alarm" type="enz_sounder">
                <Name>my-system-name-1</Name>
                <Begin>10</Begin>
                <End>20</End>
        </SystemTypeDefinition >
</ENZ_System>
```

where the type can be either enz_sounder or enz_voice.

The <Begin> element defines the time the system become active and this can have a distribution specified. Correspondingly the optional <End> element defines when the system becomes inactive and again can have a distribution specified. Now thinking about this the user has to be careful that the end time does not occur before the start time if distributions are defined so it would be better to have 'begin' and 'duration' times in the next iteration of the software.

To add one or more systems to a node use the following:

```
<System>my-system-name-1</System>
<System>my-system-name-2</System>
```

The specification of an alarm system in itself does not affect the simulations but we will need this when we come to use the Evacuation Decision Model. We also need to understand how systems are defined in Evacuationz when we want include the effects of lighting on agent movement in the next exercise.

EXERCISE 22 – MOVEMENT IN REDUCED LIGHTING

Part (a) Proulx's Model

Evacuationz uses Proulx et al.'s results for the three lighting types where the XML in a node is as follows, for example:

```
<ENZ_System>
        <SystemTypeDefinition system="enz_lighting" model="enz_proulx"
                        type="enz_emergency">
                <Name>my-system-name</Name>
        </SystemTypeDefinition >
</ENZ_System>
```

which reduces the agent's maximum walking speed by 0.75. So if we rerun our scenario from the previous exercise, Part (a) the total egress time is 22.5 s since 20 / (0.75 × 1.2) = 22.2 s. The type attribute can be set to the following for the Proulx model: enz_emergency, enz_photoluminescient or enz_ photoluminescient_emergency.

Part (b) Reduction Factor

An alternative to using Proulx et al.'s results a factor between 0.0 and 1.0 can be set with the following XML:

```
<ENZ_System>
        <SystemTypeDefinition system="enz_lighting" model="enz_factor">
                <Name>my-system-name</Name>
                <Factor>0.5</Factor>
        </SystemTypeDefinition >
</ENZ_System>
```

Part (c) Combined Effects of Smoke and Lighting

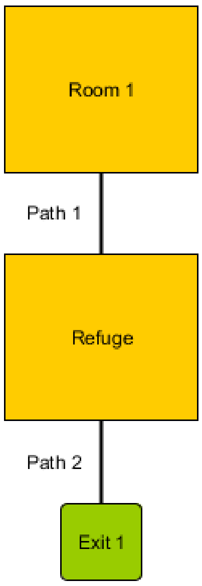
We can see that there are several factors than can reduce an agent's maximum movement speed be it density, smoke and/or lighting. The question arises as to what takes precedence? Should a congested flow also be reduced by smoke and/or lighting, do the effects of lighting and smoke combine? It would appear that the literature is not clear on this unless someone knows something I do not.

In Evacuationz where lighting is concerned a simulation uses the minimum of either the congested speed or the reduced maximum walking speed. This minimum value is then further reduced if smoke is present, i.e., congested movement is additionally affected by smoke. This algorithm might change in the future subject to finding new material in the literature.

EXERCISE 23 – NODE DELAYS

Part (a) Fixed Times

A 'refuge floor' capability can be represented in Evacuationz by creating 'node delays'. In the following examples we will create two nodes connected together that lead to an exit. The second node is used to demonstrate the delay function.



The simplest version of the node delay function is to add a fixed delay time into the node by using the following:

```
<Delay type="enz_fixed">
        <Value>100</Value>
</Delay>
```

As an agent enters a node they wait 100 s before continuing with movement (travel and/or queuing).

Part (b) Agent Types

The node delay can be applied to one or more specified agent types and can also have other functions as shown:

```
<Delay type="enz_distribution">

        <!-- Set a probability that the delay is applied to an agent -->
        <Probability>50</Probability>
        <!-- Set a distribution for the delay time -->
        <Distribution type="enz_normal">
                <Mean>50</Mean>
                <StandardDeviation>30</StandardDeviation>
        </Distribution>

        <!-- Delay only applies to specific agent type/s -->
        <AgentType include="enz_true">my-agent-type-1</AgentType>

    </Delay>
```

The <AgentType> element means that this delay is only applicable to a particular agent type. More than one agent type can be defined by creating a list of <AgentType> elements. The <Probability> element means that there is a 50% chance that the delay will occur and the value for that delay is taken from a defined distribution.

Part (c) Other Capabilities

Additional capabilities are shown with the following:

```
    <Delay type="enz_infinite">

        <!-- Delay applies all agent type/s except this -->
        <AgentType exclude="enz_true">my-agent-type-1</AgentType>

        <!-- Set a time when the delay ends -->
        <ClearTime>500</ClearTime>

    </Delay>
```

Here a specific agent type is excluded and the delay is infinite except in this case all delayed agents in the node will continue with their evacuation after 500 s. The clearance time can also have a distribution defined. Elements from Part (b) and Part (c) can be used in combination. Any <AgentType> exclusions take priority over inclusions so excluding and including an agent type will mean it is excluded from the delay.

EXERCISE 24 – EVACUATION DECISION MODEL

Part (a) Without Smoke

Evacuationz includes a modified version of the Evacuation Decision Model (EDM), based on the original work by Reneke. The Evacuationz implementation of EDM has been calibrated against several of the C/VM2 pre-travel activity times hence the addition of a vm2='enz_true' attribute when enabling this capability. Thus, to enable EDM the simulation settings need to be modified:

```
<ENZ_Simulation>
        ...other elements...
        <EDM active='enz_true' vm2='enz_true'/>
</ENZ_Simulation>
```

To use EDM we also need to set some characteristics of the agents, for example

```
<AgentTypeDefinition>
        <Name>my-agent-type-1</Name>

        <!-- Set attributes used by EDM -->
        <Attribute type='enz_familiar' />
        <Attribute type='enz_awake' />
                ...other elements...
</AgentTypeDefinition>
```

The enz_awake (or its opposite enz_asleep) attribute sets the waking state of the agent and the enz_familiar (or its opposite enz_unfamiliar) element sets the building familiarity state.

We can set the alarm type to a node using the system capability as discussed in a previous Study Guide. For the first example we will not include any smoke in the node (i.e., this represents the 'remote' scenarios in C/VM2). With the alarm sounder, when the agent is Awake and Familiar we get a time to reach the evacuating state of 61 s (c.f. 60 s in C/VM2); Asleep and Unfamiliar we get a time of 600 s (c.f. 600 s in C/VM2).

When running a simulation we can get the EDM 'level' of agents by adding the edm attribute to the <Agents> element in the scenario:

```
<ENZ_Scenario>
        <Files>
                <Agents edm='enz_true' />
                        ...other elements...
        </Files>
</ENZ_Scenario>
```

Part (b) With Smoke

Now we will add some smoke to the node so that the agents become 'intimate' with the fire. When the agent is Awake and Familiar we get a time to reach the evacuating state of 31.5 s (c.f. 30 s in C/VM2) and when the agent is Asleep we get 63 s (c.f. 60 s in C/VM2). By calibrating EDM to some C/VM2 scenarios we can find some others do not match so well and also we can suggest values for situations that C/VM2 does not address (e.g. sleeping, unfamiliar with standard or voice alarm).

Part (c) Realistic Situations

If we try something more 'realistic', say having smoke visible after 10 s and the alarm is raised at 30 s then when the agent is Awake and Familiar we get a time to reach the evacuating state of 36.5 s.

EDM is still a work-in-progress, for example not all of the calibration values for the C/VM2 combinations give the same values for the pre-travel activity time. There is no adjustment for the intensity of a signal, for example how loud an alarm is.

Note: EDM does not replace a pre-evacuation time if one is defined therefore a combined value is used. To only use EDM, the pre-evacuation time must be defined as zero.

Part (d) Moving Away From C/VM2

Recent work has investigated how EDM might be applied to non-C/VM2 situations where distributions could be applied as input to EDM inputs to get pre-evacuation times as distributions and there is a paper available if you are interested. For the moment if the vm2='enz_false' attribute is defined then you will find EDM will not work as there are other settings that need to be defined which is beyond the scope of this discussion.

EXERCISE 25 – AGENT INTERACTION

Part (a) Notification

So far agents in EvacuatioNZ have acted independently but we might want to simulation some level of inter-agent interaction. EDM has the capability to include inter-agent behaviour where the state of agents in the same room influences other agents.

Using the previous Exercise we can have the agent in Room 2 as Asleep / Unfamiliar while the agent in Room 1 has the Awake / Familiar characteristics to get them evacuating first. When the agent from Room 1 enters Room 2 they affect the EDM level of the agent already there and that effect continues until Agent 1 eventually leaves Room 2. The result is that the agent in Room 2 enters their evacuation mode in 571.5 s rather than 600 s as we found previously.

Part (b) Notification with Node Delay

Now if we modify the previous example to include a 60 s delay in Room 2 then the agent from Room 1 remains in Room 2 and so has a greater influence on the agent in that room so that their time to start evacuation reduces to 315.5 s.

Part (c) Groups

EvacuatioNZ also has an ability to create rudimentary groups of occupants that are likely to stick together.

```
<PopulationDefinition type='enz_group'>
        ...other elements...
</PopulationDefinition>
```

Using this attribute means that the agents that are created will group together if they start in the same node. When agents start to move their uncongested walking speed will be taken to be the slowest of any in the group.

We can create an example in which we have two populations, one which is grouped and the other not where the agent type speed is set to a distribution. When a simulation is run the two grouped agents move at the slowest speed of the pair while the other two move at whatever speeds were generated. By examining the agent log we can also see that EvacuatioNZ notes when agents are grouped and who with.

Part (d) Spreading Groups

When EvacuatioNZ creates a population it can spread the agents across several nodes. If we use the group attribute then this will likely result in two separate groups (unless all of the agents happen to be placed in one node, which is unlikely) as the agents must start in the same node. So if we try the following we will see that agents are grouped by the node they are created in and not all together.

```
<ENZ_Populate>
        <PopulationDefinition>
                <Agents>10</Agents>
                <NodeRef type='enz_spread'>Room 1,Room 2</NodeRef>
                <AgentType>
                        <Name>my-agent-type-1</Name>
                </AgentType>
        </PopulationDefinition>
</ENZ_Populate>
```

Part (e) Following Leaders

A recent addition to EvacuatioNZ is the ability to have 'leader' agents that will direct other agents to follow them to the same neighbouring node. Leaders are designated by adding an attribute:

```
<AgentTypeDefinition>
        <Name>my-agent-type-1</Name>
                ...other elements...
        <Attribute type="enz_leader" />
</AgentTypeDefinition>
```

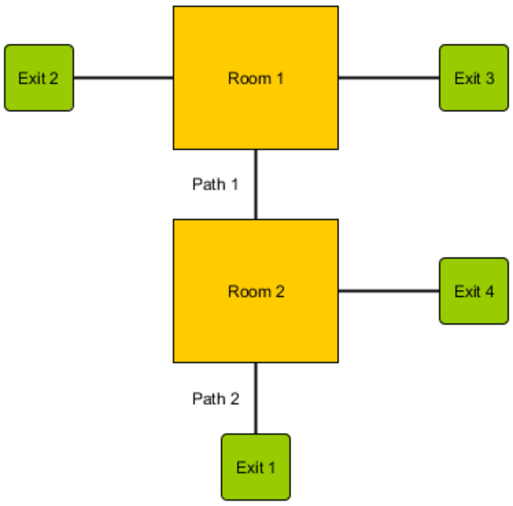
The exit behaviour of agents that are to follow a leader agent is set to enz_follow_any_leader as shown:

```
<ExitBehaviourDefinition>
        <Name>Follow</Name>
        <ExitBehaviourType type='enz_follow_any_leader' />
</ExitBehaviourDefinition>
```

If no agent with the leader attribute is found then the agent reverts to travelling towards the nearest exit node. The behaviour is likely to change and it is suggested that when using the enz_follow_any_leader behaviour that a behaviour list is employed with an alternative behaviour to follow if no leader is available.

Here we try an example in which a leader agent is given a random node selection behaviour and then the other agents that start in the same node will follow the leader.

However some follower agents are likely to get separated and so do not eventually arrive at the same exit.



Part (f) Following or Avoiding Crowded Exits

Another method which Evacuationz can consider agent interaction is with the selection of connections related to the number of agents already using them. The 'least populated' algorithm assesses the number of agents already using a connection when the agent makes its selection (and the converse for the 'most populated'). The algorithm is applied by an agent when they first enter a new node

```
<ExitBehaviourDefinition>
        <Name>Least</Name>
        <ExitBehaviourType type="enz_least_populated_connection" />
</ExitBehaviourDefinition>
```

and

```
<ExitBehaviourDefinition>
        <Name>Most</Name>
        <ExitBehaviourType type="enz_most_populated_connection" />
</ExitBehaviourDefinition>
```

There is also the ability for the user to define when the least and most populated algorithms are applied based on the number of agents at each connection being assessed, for example:

```
<ExitBehaviourDefinition>
        <Name>Least</Name>
        <ExitBehaviourType type="enz_least_populated_connection" />
        <LeastPopulatedLimit>20</ LeastPopulatedLimit >
</ExitBehaviourDefinition>
```

and

```
<ExitBehaviourDefinition>
        <Name>Most</Name>
        <ExitBehaviourType type="enz_most_populated_connection" />
        <MostPopulatedLimit>20</ MostPopulatedLimit >
</ExitBehaviourDefinition>
```

Note: The group and leader capabilities need more work. For example, it is possible for agents in a group to be given different exit choices so that they will travel to different nodes and yet their walking speed will still be constrained. The main point at the moment is that we can demonstrate that such behaviours can be modelled in Evacuationz albeit in a rudimentary way.

EXERCISE 26 – VARIABLE DOOR FLOWS

We can specify a specific flow through a door (or opening) by using the <SpecificFlow> element, for example:

```
<ConnectionType type="enz_door">
        <Width>1.80</Width>
        <SpecificFlow type="enz_distribution">
                <Distribution type="enz_triangular">
                        <Min>1.0</Min>
                        <Max>2.3</Max>
                        <MostLikley>1.33</ MostLikley>
                </Distribution>
        </SpecificFlow >
</ConnectionType>
```

When we run this simulation the specific flow changes each time and so the egress time changes accordingly.

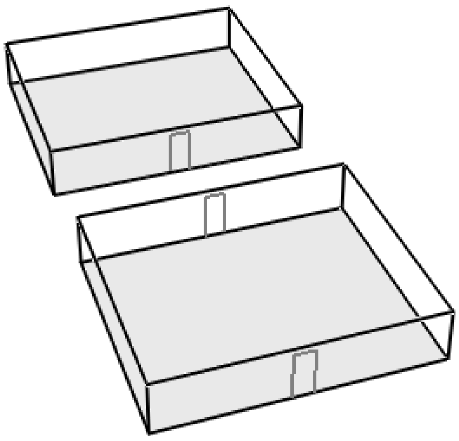
Note: There is no capability to change the specific flow for stairs although Evacuationz could be modified to do so in future.

EXERCISE 27 – SMOKEVIEW

Evacuationz has the ability to generate animated output using NIST's Smokeview (as can FDS, CFAST and B-RISK). The animations are not 'perfect' but can be a helpful way to visualise results. To create the required output files the minimum following XML is added to the 'scenario' data

```
<ENZ_Scenario>
        <Files>
                ...other elements...
        </Files>

        <Smokeview>
                <ResultFile />
        </Smokeview>

</ENZ_Scenario>
```

Running Evacuationz creates a file in the same folder called 'smokeview.smv' which can be directly opened in Smokview.



To view the agent movement, the associated particle file needs to be loaded from the right-click menu (*Load/unload > Particle file > particles*). If the animation runs too fast then use the Options > Max frame rate menu. To change the particles from dots to an agent shape use *Show/hide > Particles > Draw* menu.

Part (a) Room Dimensions etc.

The Smokeview output replicates the original yEd geometry. If the user does not want the gaps between rooms then the yEd nodes need to be butted together. The positions of the doors replicate the location of the yEd edge connectors. By repositioning the edges the door positions will change appropriately.

To get the Smokeview output to appropriately reproduce the yEd map Evacuationz scales the room dimensions from yEd with Width and Height units 10 times the dimensions wanted in metres (see Properties View > General in yEd when you select a node). This will over-ride any dimensions specified in the node Properties > Data.

To change the height and/or elevation of rooms the following elements can be added to a node XML:

```
<Height>5</Height>
<Elevation>5</Elevation>
```

The names of rooms can be shown in Smokeview from the *Show/hide > Labels* menu. The colour of a name label is inherited from the colour used in yEd.

Part (b) Agent Bodies

To change the body colour of the agent graphic, add the following to the agent type XML:

```
<AgentTypeDefinition>
                ...other elements...
        <!-- Set Smokeview agent body colour -->
        <SmokeviewColour colour="enz_blue" />

</AgentTypeDefinition>
```

where the colours available are: enz_grey, enz_purple, enz_green, enz_blue, enz_red, enz_black, enz_white. Alternatively, the colour can be set using RGB values, for example

```
<AgentTypeDefinition>
                ...other elements...
        <!-- Set Smokeview agent body colour -->
        <SmokeviewColour>0,255,255</SmokeviewColour>

</AgentTypeDefinition>
```

If an agent has a height defined in their agent type properties then this will be illustrated by the Smokeview body size. If no height property is defined then the body will be a 'standard' height.

Part (c) Smoke

If a room has smoke defined then this can be animated using the Zone file capability in Smokeview. If no height is defined then the smoke layer is shown at floor level.

EXERCISE 28 – CREATING BASE FILES

The exercises have used yEd to create the Evacuationz input files but it is possible to just use XML files directly (see later exercises). There might be occasions when someone wants to create the XML input independent from yEd, for example one might want to share the simulation file with a client or peer reviewer but not to give them the original yEd file. A 'base' input file can be generated using the <Base> element in the list of files as shown here:

```
<EvacuatioNZ_Scenario>
        <Files>
                ...
                <Base/>
        </Files>
</EvacuatioNZ_Scenario>
```

One other advantage of the raw file is that it protected from modification by a self-checking 'hash' algorithm. If someone modifies the file then the next time it is run in Evacuationz an error will occur to say the file has been changed. If the 'hash' attribute is removed from the input file then a 'missing hash code' error is given.

EXERCISE 29 – ADVANCED MAP CREATION

Part (a) Node and Path Configuration Files

Place the full path name of the external configuration file in the URL field for those nodes and paths that you want to use an external configuration file. For nodes set length and widths to 0.0

In this example we have created the following configuration file:

```
<ENZ_Map>
        <Node>
                <Name>Room 1</Name>
                <Name>Room 2</Name>
                <Width>2.5</Width>
                <Length>2.5</Length>
        </Node>

        <Connection>
                <Name>5 m</Name>
                <ConnectionType type="enz_door">
                        <Width>2.0</Width>
                </ConnectionType>
                <Length>4</Length>
        </Connection>
</ENZ_Map>
```

We have created lists of the node and connection names we want to configure from the file along with the parameters we wish to set. Currently the <Length> element will not affect the connection (although it should), but this might get updated in a future release.

Rather than repeating the path to the external configuration file in every node and connection we can instead use a reference to a common definition that holds the path name. In the main template insert a <UrlDefinition> into the config property as shown in the example:

```
        <UrlDefinition>
                <Name>my-definition</Name>
                <Path>D:\MJS\Canterbury\EvacuatioNZ\Projects\Config\
                        Test 2\my-config-2.xml</Path>
        </UrlDefinition>
```

This creates a name for the definition and the full path to the external configuration file.

Then in the URL field in the required nodes and/or connection use a reference to the name preceded by ‘config://’, i.e., for this example ‘config://my-definition’.

The purpose of this approach is that rather than having to go through each and every node and connection to update the full path name to the external configuration file only the path associated with the <UrlDefinition> needs to be changed.

Part (b) Map Lists

To come…

Part (c) Defining Node Areas

To come…

Part (d) Connection Lengths

To come…

EXERCISE 30 – XML FILE SCRIPTING

Rather than using yEd to create the input for Evacuationz an alternative is to script the XML directly in a tool such as NotePad (or I prefer NotePad++). Sometimes it is faster to write and modify the XML than it is to use yEd.

The example below shows how to create a simple map file similar to the first exercise.

```
<ENZ_Map>
        <Node>
                <!-- Node name has been omitted -->
                <Ref>1</Ref>
                <Length>10</Length>
                <Width>10</Width>
        </Node>

        <!-- The exit node -->
        <Node type='enz_safe'>
                <!-- Node name has been omitted -->
                <Ref>99</Ref>
                <!-- dimensions are not required for a safe node -->
        </Node>

        <Connection>
                <!-- Connection name has been omitted -->
                <NodeRef>1</NodeRef>
                <NodeRef>99</NodeRef>
                <Length>20.0</Length>
        </Connection>
</ENZ_Map>
```

Similarly, simple Agent Type and Populate files are shown below:

```
<ENZ_AgentType>
        <AgentTypeDefinition>
                <Name>adult</Name>
                <Speed>1.00</Speed>
                <!-- Use default exit behaviour -->
                <ExitBehaviour>enz_exit_behaviour</ExitBehaviour>
        </AgentTypeDefinition>
</ENZ_AgentType>
```

and

```
<ENZ_Populate>
        <PopulationDefinition>
                <Agents>1</Agents>
                <NodeRef refstyle='enz_ref'>1</NodeRef>
                <AgentType>
                        <Name>adult</Name>
                </AgentType>
        </PopulationDefinition>
</ENZ_Populate>
```

To run this example a Scenario file is then needed:

```
<ENZ_Scenario>
        <Files>
                <Map>D:\MJS\Canterbury\EvacuatioNZ\Projects\
                        Minimum\Example 1/map.xml</Map>
                <Populate>D:\MJS\Canterbury\EvacuatioNZ\Projects\
                        Minimum\Example 1/populate.xml</Populate>
                <AgentType>D:\MJS\Canterbury\EvacuatioNZ\Projects\
                        Minimum\Example 1/agent_type.xml</AgentType>
        </Files>
</ENZ_Scenario>
```

To run the example, select the Scenario file from the Runz tool.

To create more complex inputs, the XML scripts shown in the previous exercises can be used in the same manner.

EXERCISE 31 – RUN FROM A BATCH FILE

As an alternative to using Runz to execute a simulation you can use the command line or use a batch file (.bat). Batch files can be edited in Notepad and run from a window by double-clicking it.

For our previous example, the batch file might look something like

```
D:\MJS\Canterbury\EvacuatioNZ\Release\EvacuatioNZ.exe "D:\MJS\Canterbury\EvacuatioNZ\Projects\Minimum\Example 1\scenario.xml"
```

An advantage of having the batch file in the same folder as the input XML files is that the path names are not needed in the scenario file. Thus, this will work fine:

```
<ENZ_Scenario>
        <Files>
                <Map>map.xml</Map>
                <Populate>populate.xml</Populate>
                <AgentType>agent_type.xml</AgentType>
        </Files>
</ENZ_Scenario>
```

Another advantage is that several scenarios can be set up to run in sequence from a single batch file, for example:

```
D:\MJS\Canterbury\EvacuatioNZ\Release\EvacuatioNZ.exe "D:\MJS\Canterbury\EvacuatioNZ\Projects\Minimum\Example 1\scenario.xml"
D:\MJS\Canterbury\EvacuatioNZ\Release\EvacuatioNZ.exe "D:\MJS\Canterbury\EvacuatioNZ\Projects\Minimum\Example 2\scenario.xml"
```

Batch files have a range of commands and here is how I script mine

```
@ECHO OFF
SET path=D:\MJS\Canterbury\EvacuatioNZ\Projects\Minimum\Example 1
SET enz=D:\MJS\Canterbury\EvacuatioNZ\Release\EvacuatioNZ.exe

SET "file= Example 1\scenario.xml
%enz% "%file%"

SET "file= Example 2\scenario.xml
%enz% "%file%"
```

Yet another advantage of batch files is that they can be used to clean up any old output files that already exist in a folder that you may not want. The command *del .\"~*.csv"* would remove all of the csv files in the current folder that begin with the tilde (~). I would recommend naming all of your input files starting with the tilde as it makes it easy to identify them.

EXERCISE 32 – ADVANCED OUTPUT

Part (a) Evacuation file

To come…