Spatial Metric Calculator Manual

Richard Spinney

January 12, 2015

Contents

1 License information

2 Introduction

This document describes the use of the program 'Spatial Metric Calculator' designed by Richard Spinney whilst research associate at the Bartlett School of Graduate Studies as part of the Active Buildings project funded by the National Institute for Health Research's School for Public Health Research.

This program is desgined to allow users to to calculate a wide variety of graph based metrics using an intuitive editing interface to both construct & edit the structures involved as well as visually communicate & compare the results. In this program graphs are intended, in the main, to reflect spatial structure and so this program concerns undirected graphs only. The majority of such metrics are, or are generalisations of, common centrality & shortest path graph measures, but with significant flexibility in their construction meaning a very large number of different metrics can be derived with constraints/variations deriving from

- a powerful set of selection fields allowing carefully defined subsets of the graph's vertices to be considered including graph, spatial and visibility based constraints
- multiple possible path definitions along with an implementation which completely separates the definition of the shortest path from the metric associated with that path.

The environment in which the program was conceived meant that a considerable slant in it's design focuses it squarely on characterising and analysing office

based work environments. As such, vertex identifiers called 'locations' are explicitly identified as office based destinations such as desks, kitchens, printers and so on. However, we stress that such features are, in the first instance, simply labels, allowing one to conisder such identifiers as any 'location' that may reside on only a subset of vertices on a given graph, be that 'home', 'school', 'computer', 'road junction', 'facebook page' etc. as one sees fit. Further such functionality is supplementary to the core graph handling capabilities meaning location identification can simply be ignored allowing the program to easily serve as a generic spatial (or otherwise) graph editing and analysis suite. Finally we stress that the program makes no constraint on the design principles of any graph created or analysed with it despite it's use within it's parent project (along with example files) where a specific design principle was utlised. As such we suggest that potential users may be, in order of specific relevance,

- Users wishing to spatial characterise office spaces using graph based distances and measures and who would value office specific visualisation capabilities, identification of locations and consideration of visibility between separately identifiifed locations.
- Users wishing to design/assess a graph representing space of some kind and wish to calculate quantities that require identification of particular locations
- Users wishing to design/assess a graph representing space of some kind and require a large degree of flexibility in metric defintion and selection of vertices/edges for inclusion in those metrics
- Users who have or wish to design an arbitrary graph who wish to calculate standard shortest path/centrality metrics for it.

With this said the remainder of this manual will consider the program in its most applicable role where appropriate (where relevant) as it allows for complete explanation of all features.

3 Cheat Sheet: Buttons/interfaces

3.1 General UI

NOTE FOR WINDOWS USERS: In some versions, when the program has just started, the program thinks that Ctrl is held down when it is not. Simply press and release ctrl once and it should go back to normal.

- Left Click: Mark construct (point/vertex, link/edge, location, wall, floorarea polygon) (when relevant editing option selected, 2D only)
- Right Click: Select construct (point/vertex, link/edge, location, wall, floorarea polygon) (when editing option selected, 2D only). Also serves to deselect object if clicked away from object.
- Middle Click drag: Translate view

- Left Click drag: Rotate view (3D only, translates in 2D) Change view/direction (Free view only)
- Ctrl: Prevent left click marking constructs
- Mousewheel: Zoom in/out
- Ctrl + Mousewheel: Increase/decrease depth (3D perspective view only)
- W: reduce depth (3D perspective only), move forward (free view only)
- S: increase depth (3D perspective only)
- Alt: Allowing dragging of vertices/locations with left click drag (when relevant editing option selected, 2D only)
- Shift: Replace cursor with snap on cursor to add marking of constructs (2D only)/mark guidelines for marking of constructs (point/desk marking only)
- Shift+Ctrl+Left click drag: Rotate guidelines for marking points/desks.
- Marking edges/links:
  - First left click: select first vertex/point
  - Second left click: select second vertex/point and add edge
  - To connect points/vertices on differnt floor: select first point, change floor, select second point OR select menu item Edit/Add or remove link
- Walls:
  - First left click: Mark start of wall
  - Second left click: Mark end of wall
  - Alt: Walls marked whilst held down are set as transparent
  - 'T': Mark selected wall as transparent
  - 'O': Mark selected wall as opaque
- Floorareas:
  - First left click: start point for polygon
  - Subsequent left clicks: extend polygon
  - Right click: Close polygon geometry and add polygon to floor area definition
  - To cancel/undo: Close geometry and 'Clear last object' or select object and 'Clear selected object'

3.2 Shortcuts (also found in menus)

- Ctrl/⌘+ Q: Quit the application
- Ctrl/⌘+ O: Open saved project
- Ctrl/⌘+ S: Save current building selection as a project
- ALT/⌥+ P: Open and import a text file of points (vertices)
- ALT/⌥+ M: Open a text file containing an adjacency matrix and import the edges
- ALT/⌥+ L: Open a text file containing an adjacency list and import the edges
- Ctrl/⌘+ X: Load a floorplan from a .dxf file into the curent floor
- Ctrl/⌘+ Z: Undo (Only one step deep for only particular edits, do not rely on this)
- Alt/⌥+ E: Open mousefree add/remove link dialogue
- Ctrl/⌘+ E: Empty/delete entire building slot
- Alt/⌥+ C: Cancel current metric calculation
- Ctrl/⌘+ C: Copy information from a buffer (export it preferred)
- Ctrl/⌘+ Y: Check map for unconnected points and duplicate adjacency records
- Ctrl/⌘+ [: Zoom in
- Ctrl/⌘+]: Zoom out
- Ctrl/⌘+ I: Invert colours
- Ctrl/⌘+ K: Fullscreen
- Ctrl/⌘+ L: Undo fullscreen (also escape key)
- Alt/⌥+ R: Reset view (undo rotation and resize zoom limits based on active floorplan layers and spatial map points)
- Ctrl/⌘+ F: Toggle display of floorplan(s)
- Ctrl/⌘+ G: Toggle display of gridlines
- Alt/⌥+ S: Toggle display of scale
- Ctrl/⌘+ R: Toggle display of route illustration
- Ctrl/⌘+ P: Toggle display of point labels
- Alt/⌥+ G: Toggle display of segment labels
- Ctrl/⌘+ M: Toggle display of metric visualisation
- Ctrl/⌘+ U: Toggle display of metric colour bar
- Ctrl/⌘+ T: Toggle 3D visualisation of walls as transparent

4 Quick start gude

4.1 Basic concepts

To begin we start of with rough principles to using the program:

- An entire project can be encapsulating into a 'building'. This contains the graph based spatial map consisting of a separation of the space into distinct floors, points and links (vertices and edges of the graph), locations, walls/visibility barriers, floorareas and floorplans.
- Up to four separate buildings can be held in the program at once and are accessed through the 'building' selector on the top right hand side of the main tab area. Whenever a project is saved, the contents of that building slot alone is saved and whenever a project is opened a single building object is read into (overwritten into) the current building slot.
- Locations 'belong' to points. Points, walls, floorplans and floor areas 'belong' to floors. All points on a given floor have identical z values.
- Consequently: The floor must exist before floorplans, floorareas, points can be marked on it. It must also be selected using the floor selector in the top right corner of the main tab. At first one floor exists (floor 0). More are added using the 'Add' button. The default spacing of floors is 3m, but their heights can be changed with the floor height counter.
  - NOTE: Floors can be added, but not removed and whilst subsequently added floors can be moved inbetween existing floors in height the floor number label will not reflect the heights in this way if you do this. It is therefore strongly recommended your workflow incorporate you planning your structure FIRST, deciding on how many floors are needed and inserting & fixing their heights before any other editing. Remember all sets of points with mutually identical z values must exist as a floor, even if this is only one point!
- Consequently: Points must exist (ideally the spatial map is complete) before you add locations.
- Floorplans exist to:
  - Provide guidelines for drawing structures
  - Derive structures from (see list of functionality, importing from dxf file), but this is considered advanced and optional
  - They are not, as is, a construct themselves: metrics are not derived from them.

With this in mind here is a list of what constructs are needed for what functionality

- Floorplan: Needed as a guide for marking constructs or importing other constructs. Not essential but recommended
- Spatial graph/map: Essential for all metrics

- Walls/visibility barriers: Essential for determining visibility (Additionally needs locations for visbility to be relevant)
- Locations: Needed for increased flexibility of certain metrics by constraining included vertices (also needed for visibility)
  - Location type 'Desks' are required for alternative basis for metric whereby each desk is assigned a metric rather than each point or segment. Visibility functionlity applies only to this basis.
  - Floorareas: Not needed for metrics, but can be used to derive global or floor based densities of points, links and locations.

4.2 Starting a building

So let's look at what a typical starting set up might be

- I want to characterise a building with two floors and have two dxf floorplans, one of each floor
  - Note: If the spatial graph needs to represent connections between those floors as anything other than single edges/links between the floors extra floors would need to be considered in order to house any intermediate points.
- So I click 'add' in the floor dialogue so that I have two working floors (0 and 1)
- I use the floor height counter to adjust the heights of the two floors
- I select floor 0 and then press Ctrl/⌘+ x or (file/read pre-existing structures/open dxf floorplan) and select the dxf floorplan which loads into the floor
- I select floor 1 and then press Ctrl/⌘+ x or (file/read pre-existing structures/open dxf floorplan) and select the dxf floorplan which loads into the floor
- I check the units of each floorplan in each floor and scale the floorplan if there are any errors and use the shift x and shift y counters to ensure the two floorplans are on top of each as I intend. These are found in the floorplan subtab on the right hand side of the main tab area. Both floors can be viewed at the same time using the 'All' option in the floor option selector so ensure alignment. Alternatively gridlines and xy coordinates written to the screen can be used.
- For each floorplan I open the layer dialogue and activate/deactivate any layers I wish to include/exclude

At this point the spatial map or walls/visibility barriers of different types can be imported from the floorplan (specifically all active layer of all the foors for walls or all active layers for the current selected floor for points/links), but we delay discussion until the list of functionality section.

After this point the spatial map/graph, floorareas and walls/visibility barriers can be drawn/marked.

4.3 Marking/editing the spatial map manually

Once the floors are set up we can mark and edit the spatial map. The editing rules are as follows

- On the relevant floor:
- In the main edit subtab on the right hand side of the main tab area select 'spatial graph: points' in the edit selector.
- Left click wherever a vertex/point is desired.
- Hold shift whilst the cursor is near an existing point to overlay 90 degree guidelines for the placement of more.
- Hold down Ctrl whilst shift is depressed to rotate those guidelines by dragging the mouse. Resetting the edit selector will reset the guidelines to 90 degrees.
- Drag a point around by holding down alt before clicking and dragging
- Drag a point along guidelines by then subsequently (and simultaneously) pressing shift near the point you want to overlay guidelines over.
- Select a point with right click when the edit selector is set to 'spatial graph: points'
- Remove selected point with 'remove selected object' button in main edit subtab.
  - Note: Any links associated with a removed point will be deleted as will any locations that belong to that point.
- In the main edit subtab on the right hand side of the main tab area select 'spatial graph: links' in the edit selector.
- Add links by first clicking on one point and then clicking on the point it will be connected to
- Right click to select and press 'remove selected object' to delete
- To add links across floors either change floor between left clicks or select the menu view/point labels, identify the numbers of the points you wish to connect and then select the menu edit/add or remove link (mousefree) and enter the points then press add.
- To remove links across floors use the edit/add or remove link (mousefree) method.

An example image of marking a link between two points is shown in figure 1 An example image showing a highlighted point is shown in figure 2

NOTE: The spatial map can also be imported through a pair of text files containing the points and adjacency information. Alternatively just the points can be read in. This is so that one can design the spatial map in other software or derive it from other data and then load it in to the program.

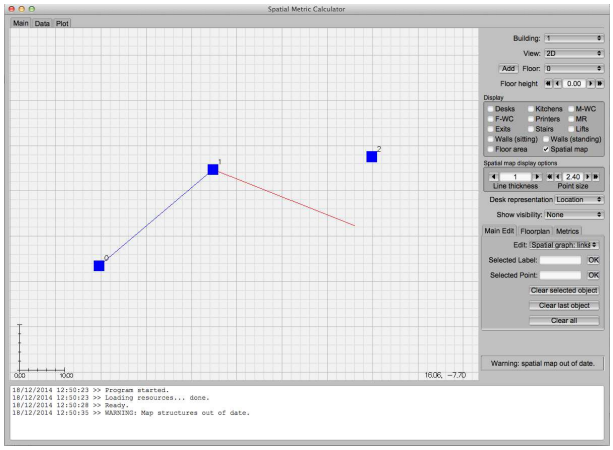

Figure 1: An edge/link being marked between point/vertex 1 and 2. A link already exists between point 0 and 1.

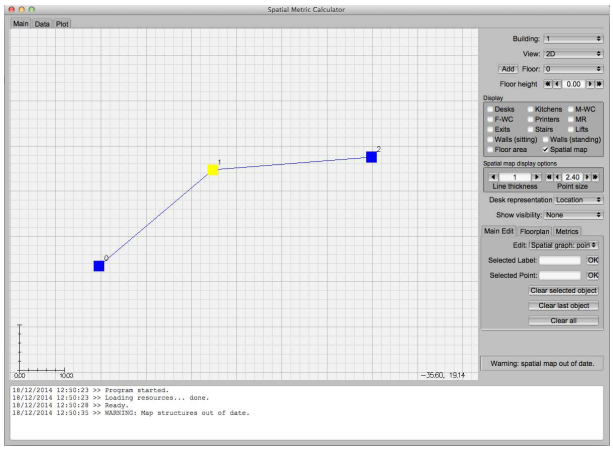

Figure 2: Point/vertex 1 is highlighted. The point can then be selectively deleted

4.4 Marking/editing walls

Once the floors are set up we can mark and edit visibility barriers/walls. The editing rules are as follows

- On the relevant floor:
- In the main edit subtab on the right hand side of the main tab area select 'visibility barriers: sitting/standing' in the edit selector (for different height barriers).
- Left click to start and wall
- Right click to end a wall
- Holding alt will mark new walls as transparent.

- Select sitting/standing barrier with right click when the edit selector is set to the appropriate option
- When selected:
  - Press O: make opaque (walls are opaque by default)
  - Press T: make transparent
  - To delete press 'remove selected object'

4.5 Marking Floor areas

Once the floors are set up we can mark floorareas (If one wishes to calculate net internal areas, calculate densities etc.)

- On the relevant floor:
- In the main edit subtab on the right hand side of the main tab area select 'floor areas' in the edit selector
- First left click: Start polygon
- Subsequent left clicks: Extend polygon
- Right click: Close polygon geometry
- Right click whilst no polygon not starting: select polygon which can then be deleted with 'remove selected object'
- Polygons cannot intersect themselves, but polygons can intersect other polygons which should be avoided.
- Area of floorplan is the sum of areas of individual polygons hence why overlapping polygons should be avoided.
- Existing polygons cannot be edited and all polygons are simple polygons without holes so complex geometries must be formed of multiple polygons.

4.6 Marking/editing location markers

Once the spatial map has been set up we can add location markers. To do so we follow these rules:

- On the relevant floor:
- In the main edit subtab on the right hand side of the main tab area select the type of location to be added in the edit selector eg 'Desk'.
- Add location marker with left click
- Remove location marker by selecting with right click and then pressing 'clear selected object'
- Location markers are automatically associated with a point (the closest) to which it belongs. To over ride the point which it has been associated select the location marker with right click, enter the new point in the 'selected point' field and press OK.

- Similarly, to give the location marker a label (useful for identifiying specific location markers, particularly desks for desk based metrics) right click the location marker then enter the label into the label field and press OK.

Note: The point to which the location marker belongs is what is used to calculate any graph measures, whereas the postion of the location marker determines the point in space visibility information is calculated from. By allowing the owning point to be far away from this position (through overriding the associated point) we can manually dictate the rules of visibility according to the user. For instance I may believe that it is possible to 'see' a location by being in line of sight with the door that leads to it, but insist that the location itself resides firmly beyond the door in terms of any graph measures.

Figure 3 shows two desk marker placement/association and selecting Note: We

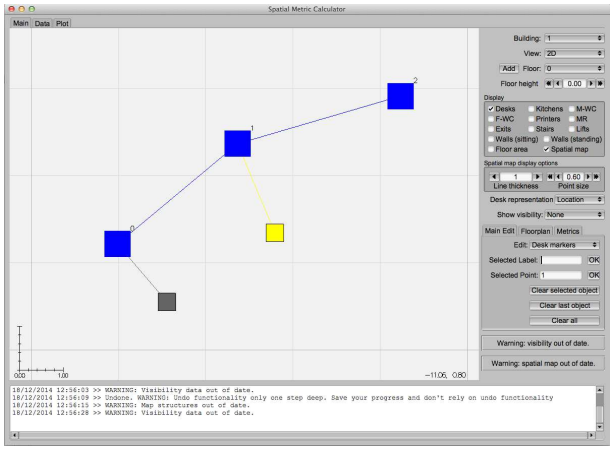

Figure 3: A desk location marker is associated with point 0 and another with point 1. The marker associated with point 1 is selected. It's label and associated point can be edited in the appropriate fields on the right.

make careful distinction between location *markers* which we have just described and locations which are interpreted *from* location markers. In this regard desks behave differently to all other types of destination. Each desk location marker counts as a distinct desk location regard of the spatial map point to which it belongs. However, for all other destination types the location marker is simply a visibility point and all location markers of a given type, all belonging to one spatial map point, *together* count as a single destination. This is so that these destinations' visbility behaviour can extend beyond that of a point in space. Explicitly: two desk location markers associated with point 1 are interpreted as two desks each associated with point 1, each with their own single visibility point. On the other hand two kitchen location markers associated with point 1 are interpreted as a single kitchen with two visibility points. The program illustrates this by connecting non-desk location markers together if associated with the same point. For an example see figure 4.

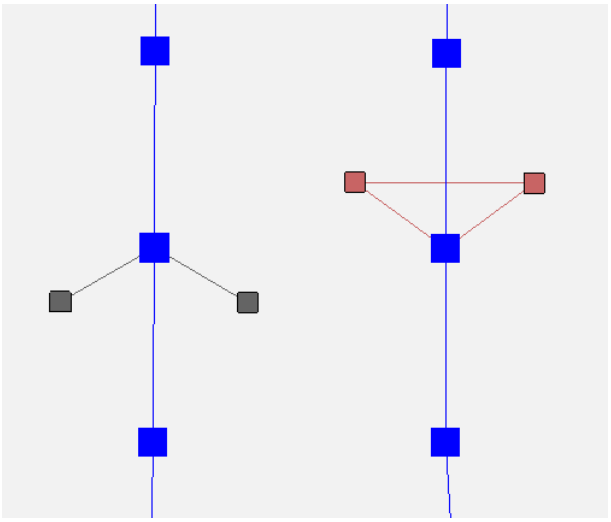

Figure 4: Two separate desk markers (grey) on the left, each belonging to the same vertex/point (blue) count as two separate desk locations. Two kitchen markers (red) on the right, each belonging to the same vertex/point count as two visibility points for one kitchen location.

5 List of functionality

Here we go through additional functionality excluding metric generation

5.1 Saving/loading

Building objects can be saved to files which can be read in later to continue work on them. The program is designed to read files with a .proj ending so save them with that ending or edit them in your OS to ensure they get this ending at some point.

5.2 Data/Exporting

All constructs and calculated metrics (eg. the list of desks with labels, positions and owning points) can be viewed in the 'Data' tab. The contents of which is copyable, but we recommend exporting to a text file of your choice with the 'export' button. The data comes with headers (including important descriptions of the metrics for the relevant selection option) which can be removed in any text editor. The remaining data is comma separated.

5.3 Visualising metric data

In addition to the colour map visualisation facility of the metric onto the graph the user can histogram or scatterplot the metric data from any of the derived metrics from any of the four building slots. Up to three metrics can be scatterplot and four histogramed. When three variables are scatterplot 3D rotation is possible and when 2 variables are scatterplot regression is possible if there

are an equal number of items in each metric. When more than one metric is histogramed 3D rotation is possible. With the histogram and interpolation can be shown in addition to or instead of the standard bar chart. The interpolation is transparent so eases comparison of multiple metrics.

This facility can be used to examine changes in metrics between buildings in distribution etc. or relationships between differnt metrics in the same building. For example I could save my current building in slot 1 and then open it again in slot 2. I could then make edits in slot two and calculate the same metric for both buildings. The changes that edit made would be observable in these plots. Figure 5 shows an example histogram of a metric, figure 6 two metrics histogrammed and figure 7 two metrics scatter plot.

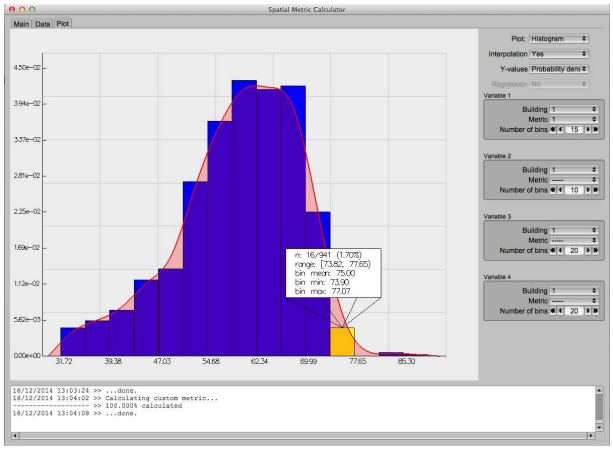

Figure 5: A histogram of a single metric variable.

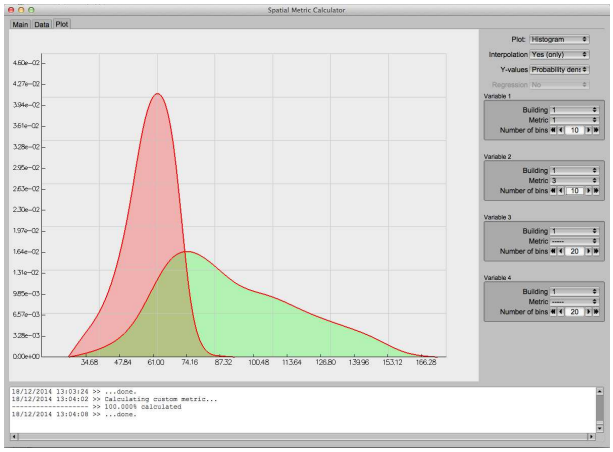

Figure 6: A histogram of two metric variables.

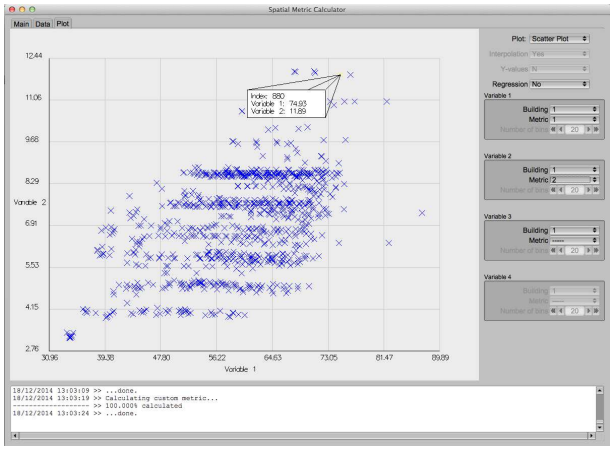

Figure 7: A scatter plot of two metric variables.

5.4 Importing data

5.4.1 Importing points/vertices from a file

You can import a text file of points. To do so go to menu file/read pre-existing structures/open points. The format must be one of

- x y z
- i x y z
- i x y z floor

where i is the index or number of the point. Each row must be on a separate line or encased within (), [] or {}. The individual values can be separated by a space, a comma or a semi colon. Reading in points in this way will reset the buiding project and so it is recommended this is done first, and in this instance, before the loading of any floor plans (If you are reading in points it is assumed you have designed the floor structure elsewhere).

5.4.2 Importing links/edges with an adjacency list

You can import a text file of an adjacency list to build the internal adjacency list. To do so go to menu file/import pre-existing structures/open adjacency list. The list must be contain an entry for each link you wish to add and each edge entry must consist of two integers: the indices of the points that are being connected. The index of the points must match the order of the points/vertices that have been read in/marked. Each entry must be on a separate line of the text file or be encased within (), [] or {}. The individual values can be separated by a space, a comma or a semi colon. Checks for duplicates are performed so be patient if the process is taking some time.

5.4.3 Importing links/edges with an adjacency matrix

You can import a text file of an adjacency matrix to build the internal adjacency list. To do so go to menu file/import pre-existing structures/open matrix. The

matrix must be square (although only the top right triangle is read) and have entries of 0 for no edge and 1 for the existence of an edge. The index of the adjacency matrix must match the order of the points/vertices that have been read in/marked. Each line of the matrix must be on a separate line of the text file or be encased within (), [] or {}. The individual values can be separated by a space, a comma or a semi colon. If the matrix size exceed the number of points no further reading is performed after that point. If the matrix is smaller it assumes all unaddressed points are unconnected.

5.4.4 Importing walls from a dxf

You can import walls/visibility barriers in from the current active layers of the floorplan. To do so we recommend specifically designing a particular layer in a CAD application with an appropriate and modest number of lines taken as visibility barriers. Navigating to menu file/read pre-existing structures/floorplan to walls gives four options based on the different height and transparency/opacity options. Accepting will read all lines in active layers on the current floor into the building object as walls on that floor in addition to any existing walls. Note: Excessive numbers of walls will cause considerable performance slow down (particularly in 3D views, and especially if using transparency effects, where it may grind to a halt completely) which can happen if one simply exports all lines in a floorplan (often in the tens of thousands) in as walls. We recommend careful use of this feature, ideally with a small number of specifically drawn layers for this purpose designed to capture basic visibility features and shape of walls rather than small details.

5.4.5 Importing spatial map from a dxf

It is possible to import points and links into the spatial map from a dxf file. In this case each unique line in every active layer of the floorplan of the current floor, consisting of a start and end point, is imported as an edge/link and all end points which are within a floating point tolerance of $1 \times 10^{-6} m$ of each other are counted as individual points/vertices. They are read into the current floor in addition to any points that already exist on that floor.

5.4.6 Importing location markers

Given an existing set of spatial map points/vertices it is possible to read in text files containing visibility marker information. The format must be one of

- label x y z point
- label x y z
- x y z point
- point

where 'label' is a string label and 'point' is an integer instructing the program which spatial map point to associate the location marker with. If the point is out of bounds or not provided the location marker is associated with the point closest to the visibility location x,y,z. Each row must be on a separate line or

encased within (), [] or {}. The individual values can be separated by space, a comma or a semi colon.

6 Metrics

6.1 Basics

Once the spatial map, visibility barriers and location markers has been set preliminary functions need to be run in order to create structures used for calculating metrics. Buttons that run these functions can be found in the metrics subtab on the right hand side of the main tab area.

The 'create' button generates the structures needed for metric calculation based on the spatial map. Creating these structures also requires the definition of a turn to be set which is achieved through the turn definition counter as is defined in degrees. This quantity represents an angular deviation which if exceeded in any single angular deviation, counts as a single turn. When this calculation is completed the progress bar will turn green and any visible warnings wil disappear. Any changes to the spatial map or turn definition will cause the structures to go out of date, the progress bar will turn red and a warning box in the lower right hand side will appear. No metric calculations can occur in this state and the structures will need to be recalculated before any further metrics are generated.

The 'visibility' button calculates the structures which hold the visibility information for the locations with respect to all desk locations based on the location and type of all marked visibility barriers. When this calculation is completed the associated progress bar will turn green and any visibe warnings will disappear. Any changes to location markers or visibility barriers will cause these structure to go out of date, the progress bar to turn red and a warning box to appear in the lower right hand side. Metric calculations *can* occur when these structures are out of date, but none that depend on visibility information.

The 'find lines' button identifies segments that can represent sequences of segments that can be traversed without incurring any turns. Such destinations can only be utilised for segment based metrics (see next section), have restrictions placed upon the nature of the metrics and can be costly to compute if the self count option is set to anything other than 'Yes (if qualifying)'. Metric calculation can occur when these have not been identified, but turnless lines are not available as a destination. Any changes to the spatial map or turn definition causes the identification of these segments to go out of date.
The 'clear' button clears all created graph structures AND clears all calculated metrics.

The 'route' button opens a dialogue allowing for the display of shortest paths, based on a number of different metrics which can be minimised, between two specified points. If more than one minimum path exists they are also shown up to a limit of 1000. The number of equivalent paths and the metric length are reported in the main program dialogue. The route display can be turned

on and off in the menu options/route. Routes can only be displayed if the map structures have been created.

The 'metric dialogue' button opens a new window which provides the main interface for metric calculation. In this window, at the top, is a selector labeled 'metric no./display metric'. The option allows selection of which metric, already calculated, to display, overlaid on the spatial map if the 'Display/Cancel' button is pressed or selects which metric slot (or new metric slot) to overwrite with the next calculated metric if the 'calculate' button is pressed.
Figure 8 shows visualisation of a segment based metric.

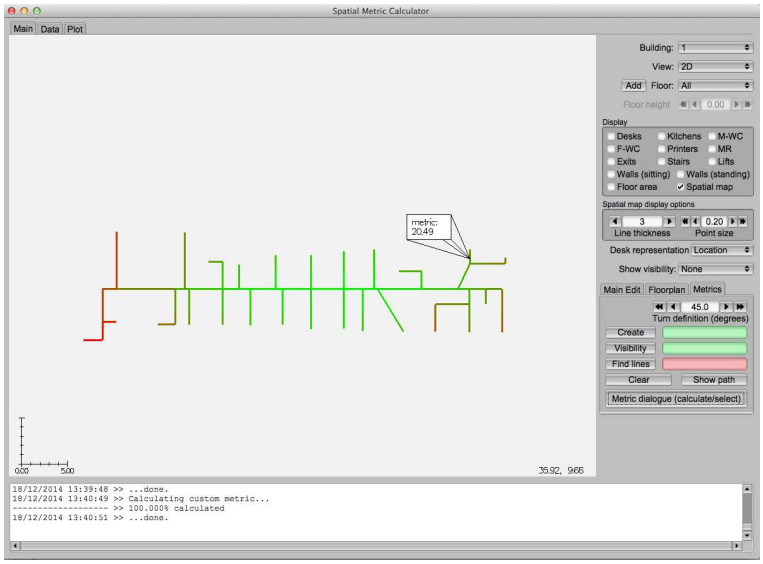

Figure 8: A visualisation of a segment based metric.

6.2 Procedure and definitions

The first thing to be appreciated is to realise that the specification of any metric is anywhere from a 2 to 4 step procedure with 1) metric basis and 2) metric measure always needing to be specified. The next two are only required for certain metric selections and are 1) destination selection and 2) route selection. Very broadly (though not exactly) one can think of all the spatial metrics as being based upon various measures of distance either directly (expiclty returning distance) or indirectly (finding properties of the paths that minimise that distance for instance). In such terms one can think of all such specifications as achieving the following

- Metric basis: What are we calculated the metric for (what is the produced number associated with?)
- Metric measure: What are we measuring (e.g. what kind of distance)
- Destination selection: Where are we going/from what subset of points/segments etc. is what we are measuring calculated from

- Route selection: How are we getting to the destination. What route finding criteria are we using.

We now carefully itemise all possible options to allow for complete unambigous specification of any given metric. Figure 9 shows the metric dialogue illustrating all possible options

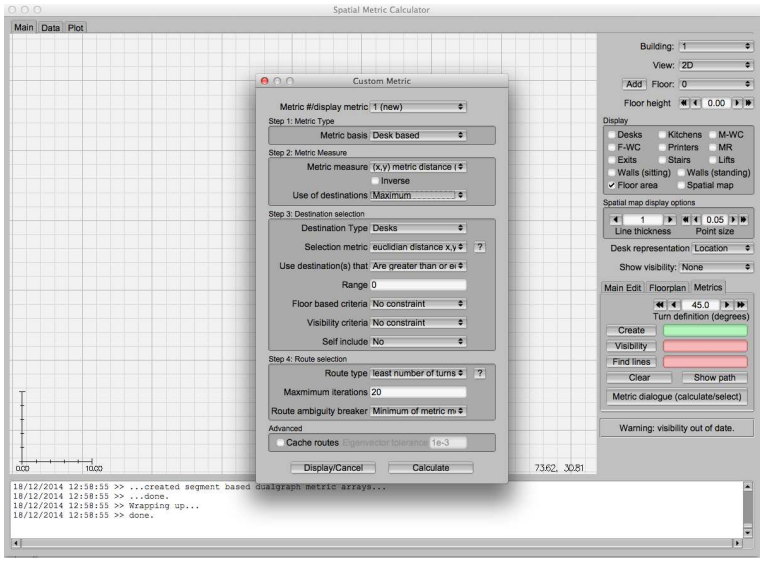

Figure 9: The metric dialogue

6.2.1 Metric basis

This option determines what each metric number is associated with. The options are

- Desk based: This generates a metric for every desk location which is equal to the number of desk markers regardless of if they occupy the same spatial map point. Metric information for each desk is calculated by using the point with which they are associated.
- Point based: This generates a metric for every point in the spatial map
- Segment based: This generates a metric for every segment in the spatial map. Note: calculation of metrics for segments relies on an alternative graph definition. This is automatically generated and makes no difference to the procedure, but means locations cannot be identified as locations are associated with points, not segments.

6.2.2 Metric Measure

When determining the metric measure anywhere between 2 and 3 options are taken. The first is the main metric measure choice, the second an inverse choice and the third a statistic choice. The statistic choice is only required if the metric

measure can be applied from the basis point/segment to any other individual point/segment. The main metric measure options are as follows

- Euclidian distance(x,y)
  - The metric is defined between pairs of points or segments and so is defined between the basis point/segment and any qualifying destinations
  - So a statistic of the metric applied to all qualfying destinations is needed
  - The metric is the planar euclidian (as the crow flies) distance, not counting any vertical distance, between the basis point/segment and the destination's point/segment where the position of segment is taken as it's midpoint
  - Since the distance is as the crow flies no route finding is required
- Euclidian distance(x,y,z)
  - The metric is defined between pairs of points or segments and so is defined between the basis point/segment and any qualifying destinations
  - So a statistic of the metric applied to all qualfying destinations is needed
  - The metric is the three dimensional euclidian (as the crow flies) distance, including vertical distance, between the basis point/segment and the destination's point/segment where the position of segment is taken as it's midpoint
  - Since the distance is as the crow flies no route finding is required
- (x,y) metric distance
  - The metric is defined between pairs of points or segments and so is defined between the basis point/segment and any qualifying destinations
  - So a statistic of the metric applied to all qualfying destinations is needed
  - The metric is the length accumulated in metres whilst traversing the graph between the basis point/segment and the destination's point/segment, not counting any vertical movement, where the start/end point of a segment is taken as it's midpoint
  - Since the distance depends on the route taken route finding is required
- (x,y,z) metric distance
  - The metric is defined between pairs of points or segments and so is defined between the basis point/segment and any qualifying destinations
  - So a statistic of the metric applied to all qualfying destinations is needed

  - The metric is the length accumulated in metres whilst traversing the graph between the basis point/segment and the destination's point/segment, including any vertical movement, where the start/end point of a segment is taken as it's midpoint
  - Since the distance depends on the route taken route finding is required

- angular distance
  - The metric is defined between pairs of points or segments and so is defined between the basis point/segment and any qualifying destinations
  - So a statistic of the metric applied to all qualfying destinations is needed
  - The metric is the number of degrees passed through, where all turns count as positive contributions, whilst traversing the graph between the basis point/segment and the destination's point/segment
  - Since the distance depends on the route taken route finding is required

- number of segments
  - The metric is defined between pairs of points or segments and so is defined between the basis point/segment and any qualifying destinations
  - So a statistic of the metric applied to all qualfying destinations is needed
  - The metric is the number of distinct steps between point/segments accumulated whilst traversing the graph between the basis point/segment and the destination's point/segment. Eg a move from one point to an immediately adjacent point with which it is connected or a move from a segment to another with which it is connected counts as one step.
  - Since the distance depends on the route taken route finding is required

- number of turns
  - The metric is defined between pairs of points or segments and so is defined between the basis point/segment and any qualifying destinations
  - So a statistic of the metric applied to all qualfying destinations is needed
  - The metric is the number of turns, defined as any individual deviation greater in magnitude than the user set turn definition, accumulated whilst traversing the graph between the basis point/segment and the destination's point/segment.
  - Since the distance depends on the route taken route finding is required

- number of qualifying destinations
  - The metric is defined between the basis point/segment and all qualifying segments
  - So a statistic of the metric applied to all qualfying destinations is *not* needed
  - The metric is the number of destinations (points, segments, locations) which meet the specified criteria
  - No paths are involved so no route finding is required
- number of equivalent weighted route metric minimising paths to destinations
  - The metric is defined between pairs of points or segments and so is defined between the basis point/segment and any qualifying destinations
  - So a statistic of the metric applied to all qualfying destinations is needed
  - The metric is the number of distinct routes which minimise a specified metric used when routefinding
  - The metric explicitly concerns routes so routefinding is required
- number of shortest paths through basis point/segment
  - The metric is defined between the basis point/segment and all qualifying segments
  - So a statistic of the metric applied to all qualfying destinations is *not* needed
  - The metric is the number of distinct shortest paths between each distinct (not doubled counted) pair of qualifying destinations that pass through the basis point/segment
  - The metric explicitly concerns routes so routefinding is required
- Betweeness centrality normalised on $[0, 1]$ and generalised
  - The metric is defined between the basis point/segment and all qualifying segments
  - So a statistic of the metric applied to all qualfying destinations is *not* needed
  - The metric is defined as follows. If the set of qualifying destinations runs from $i = 1, \ldots, N$, and the number of shortest paths between destination s and t is σ_{st} and the number of shortest paths between s and t that also pass through v is $\sigma_{st}(v)$ it is defined as

$$BCG(v) = \frac{2}{N(N-1)} \sum_{s=1}^{N} \sum_{t>s}^{N} \frac{\sigma_{st}(v)}{\sigma_{st}} \tag{1}$$

    We note destinations, as shown later, can be locations, points or segments and are selected based on individual criteria, and the metric

defining the path can be defined thus generalising the betweeness centrality. The self counting selection seen later should be off to replicate the original betweeness centrality

  - The metric explicitly concerns routes so routefinding is required

- Length of subgraph formed from shortest paths between qualifying destinations
  - The metric is defined between the basis point/segment and all qualifying segments
  - So a statistic of the metric applied to all qualfying destinations is *not* needed
  - The metric is the combined total length in metres of all links/segments that are involved in any shortest path between all pairs of qualifying destinations.
  - The metric explicitly concerns routes so routefinding is required

- Eigenvector centrality
  - The metric is defined between the basis point/segment and all qualifying segments
  - So a statistic of the metric applied to all qualfying destinations is *not* needed
  - The metric is the ith element of the eigenvector associated with the maximum eigenvalue of the adjacency matrix of the relevant graph structure.
  - The metric does not concern paths so routefinding is *not* required
  - The metric utilises the entire adjacency matrix and so destination selection is not required.

The statistic option called 'use of destinations' has the following options:

- Minimum: Find the smallest value of the metric from the basis point/segment to all qualifying destinations
- Maximum: Find the largest value of the metric from the basis point/segment to all qualifying destinations
- Mean: Find the mean value of the metric from the basis point/segment to all qualifying destinations
- Media: Find the median value of the metric from the basis point/segment to all qualifying destinations
- Geometric mean: Find the geometric mean value of the metric from the basis point/segment to all qualifying destinations defined as $GM = (\prod_i^n x_i)^{1/n}$
- Harmonic mean: Find the harmonic mean value of the metric from the basis point/segment to all qualifying destinations defined as $HM = n/\sum_i^n x_i^{-1}$

- Variance: Find the variance of the values of the metric from the basis point to all qualifying destinations

Finally the inverse option takes the final returned value for the basis point/segment as 1 divided by the normal value, performed after the statistic 'use of destinations' operation where appropriate.

6.2.3 Destination selection

Here a series of criteria allow for systematic inclusion of destination on the graph in the above metric calculations. They include destination type, metric distance to basis point/segment constraints, floor based constraints and visibility constraints. They are as follows:

- Destination type:
  - Selects what type of point or segment to consider including as destinations: required for all metrics but eigenvector centrality
  - For desk or point based metrics can be: any type of location or every point in the graph. If desks are selected each is treated as an individual destination even if at the same spatial graph point.
  - For segment based metric can be: every segment in the spatial map or segments that represent turnless lines (if found). If the latter is chosen the destination and route metrics MUST be number of turns and the metric measure MUST be number of turns, number of destinations, number of equivalent weighted paths (based on turns) or eigenvector centrality (for which destinations are irrelevant). Hoever, the program insists on these options being selected so no invalid metrics can be calculated.
- Selection metric:
  - Required if the 'use destination(s) that' field is set to anything other than (don't exclude based on selection metric)
  - Specifies metric by which destinations can be included or excluded quantitatively based on measures of distance
  - These measures of distance can be
    - Euclidian (x,y) from basis point/segment to destination
    - Euclidian (x,y,z) from basis point/segment to destination
    - Shortest graph (x,y) distance from basis point/segment to destination
    - Shortest graph (x,y,z) distance from basis point/segment to destination
    - Least angular deviation from basis point/segment to destination
    - Least number of segments from basis point/segment to destination
    - Least number of turns from basis point/segment to destination based on user defined turn definition
- 'Use destination(s) that' field

  - Specifies how to use selection metric
  - Options are
    * Don't exclude based on selection metric
    * Use destinations that minimise selection metric
    * Use destinations that maximise selection metric
    * Use destinations that are equal to or less a given cut off in the selection metric
    * Use destinations that are equal to or greater a given cut off in the selection metric
    * Use destinations that maximise the selection metric whilst being less than or equal to a given cut off in the selection metric
    * Use destinations that minimise the selection metric whilst being greater than or equal to a given cut off in the selection metric
    * Use destinations that are exactly equalt to a specified value of the selection metric

- Range
  - Is the cut off/specified value for the last 5 of the above choices
  - Not required otherwise

- Floor based criteria
  - Choose to only include destinations on the same floor as the basis point/segment or not to constrain based on floor.
  - Segments which broach floors are defined as floorless and will have no valid destinations

- Visibility criteria
  - Choose to only include destinations which are visible from the basis desk (desk based metrics only)
  - Options
    * Define visbility by blocking line of sight with only ceiling height barriers, but both opaque and transparent barriers
    * Define visbility by blocking line of sight with both ceiling height and 1m high barrier of both opaque and transparent types
    * Define visibility by blocking line of sight with only ceiling height barriers that are opaque
    * Define visibility by blocking line of sight with both ceiling height and 1m high barriers that are opaque

- Self include
  - This option allows for consideration of the basis desk/point/segment as a destination
  - The options are

    * No: Do not consider the basis desk/point/segment as a destination. This is relevant when the metric is desk based and the destinations are desk, when the metric is desk based and the destinations are point, the metric is point based and the destinations are point or when the metric is segment based and the destinations are segments or turnless lines. With turnless lines it is slightly more complicated in that the segment that represents the turnless line which can be traversed to from the basis segment in no turns is not included.
    * Yes (if qualifying): Consdier if the basis desk/point/segment qualifies as a destination according to all the above criteria. This means that the desk, the desk's point, the point or the segment are included *if* the destination is of type desk, point, point and segment resepctively and the basis desk/point segment fulfils all selection metric criteria
    * Yes (force): Include the basis desk/point/segment/turnless line by appending it to the list of destinations regardless of destination type or any other criteria.

6.2.4 Route selection

Here we describe the options used for implementing route finding, needed in the situation described above in the metric measure sub section.

- Route metric
  - Defines the metric which is minimised in order to produce the route. For instance the route which minimises the number of turns or angular distance may be a different route than that which minimises the total accumulated distance.
  - Options are
    * Route minimises (x,y) graph distance
    * Route minimises (x,y,z) graph distance
    * Route minimises angular distance
    * Route minimises number of segments
    * Route minimises number of turns according to user defined turn definition
- Iterations
  - Is an integer number which specifies how long to keep enumerating/searching for distinct paths that all minimise the route metric
  - The number is the maximum number of equivalent shortest paths it can find
  - Warnings are given if this number is reached and not all shortest paths are found
  - Not finding all shortest paths when large numbers are used suggests an ill-formed graph: check that the turn definition isn't greater than 90 degrees

  - Only required when
    * The metric concerns numbers of paths or requires all paths eg betweeness centrality or length of subgraph
    * When the metric measure is of an accumulated type (eg distance, number of turns, etc.) and is being counted along a route which is defined by minimising a different metric since then different routes which all minimise the route metric may lead to different metric measure values

- Route ambiguity breaker
  - Only required when the metric measure is of an accumulated type (eg distance, number of turns, etc.) and is being counted along a route which is defined by minimising a different metric since then differnt routes which all minimse the route metric may lead to different metric measure values
  - In such cases to find a single unique number we need to select a particular metric measure out of all the metric measures associated with each equivalent route (In practise this may be required only rarely as there may only be one equivalent metric minimising path)
  - The options are:
    * Choose the minimum metric measure out of all the metric measures generated by the equivalent routes
    * Choose the maximum metric measure out of all the metric measures generated by the equivalent routes
    * Use the mean of the metric measures from all the metric measures generated by the equivalent routes

6.2.5 Some examples

Here we describe how one might produce some specific graph measures.

Metric example 1: The distance traversed from each desk to the (radially closest) kitchen that is visible (using all tall barriers), by taking the route which incurs the least angular deviation.

Options:
Metric basis: Desk based.
Metric measure: (x,y) graph distance to destinations.
Inverse: No.
Usage of destinations: Minimum.
Destination type: Kitchens.
Selection metric: (x,y) euclidian distance.
Use destination(s) that: Minimise selection metric.
Range: N/A.
Floor based constraint: No constraint.
Visibility constraint: Standing visible (all barriers).
Self include: No.
Route metric: Least angular distance.

Iterations: 20 (Set higher if errors returned: unlikely however).
Route ambiguity breaker: Minimum.

Metric example 2: The mean distance traversed from each desk to all other desks, by taking the route which incurs the least distance traversed

Options:
Metric basis: Desk based.
Metric measure: (x,y) graph distance to destinations.
Inverse: No.
Usage of destinations: Mean.
Destination type: Desks.
Selection metric: N/A.
Use destination(s) that: Don't exclude based on selection metric.
Range: N/A.
Floor based constraint: No constraint **Visibility constraint:** No constraint
Self include: No.
Route metric: (x,y) least metric graph distance.
Iterations: N/A.
Route ambiguity breaker: N/A.

Metric example 3: Number of visible desks whilst standing from each desk

Options:
Metric basis: Desk based.
Metric measure: Number of destinations.
Inverse: No.
Usage of destinations: N/A **Destination type:** Desks.
Selection metric: N/A.
Use destination(s) that: Don't exclude based on selection metric.
Range: N/A.
Floor based constraint: No constraint.
Visibility constraint: Standing visible (all barriers).
Self include: No **Route metric:** N/A.
Iterations: N/A.
Route ambiguity breaker: N/A.

Metric example 4: Degree centrality (point based)

Options:
Metric basis: Point based.
Metric measure: Number of destinations.
Inverse: No.
Usage of destinations: N/A.
Destination type: Points.
Selection metric: minimum number of segments between basis and destination.
Use destination(s) that: Are equal to 'range' from orgin based on selection metric.
Range: 1.

Floor based constraint: No constraint.
Visibility constraint: No constraint.
Self include: No.
Route metric: N/A.
Iterations: N/A.
Route ambiguity breaker: N/A.

Metric example 5: Distance centrality utilising turn based distance (segment based)

Options:
Metric basis: Segment based.
Metric measure: Number of turns to destinations **Inverse:** Yes.
Usage of destinations: Mean.
Destination type: Segments.
Selection metric: N/A.
Use destination(s) that: Don't exclude based on selection metric.
Range: N/A.
Floor based constraint: No constraint.
Visibility constraint: No constraint.
Self include: No.
Route metric: Least number of turns.
Iterations: N/A.
Route ambiguity breaker: N/A.

Metric example 6: Graph centrality using 3D metric graph distance (point based)
Options:
Metric basis: Point based.
Metric measure: (x,y,z) graph distance to destinations **Inverse:** Yes.
Usage of destinations: Maximum.
Destination type: Points.
Selection metric: N/A.
Use destination(s) that: Don't exclude based on selection metric.
Range: N/A.
Floor based constraint: No constraint.
Visibility constraint: No constraint.
Self include: No.
Route metric: (x,y,z) Least metric graph distance.
Iterations: N/A.
Route ambiguity breaker: N/A.

Metric example 7: Betweeness centrality, number of step/number of edges based distance (point based)

Options:
Metric basis: Point based.
Metric measure: Beteeness centrality (normalised) generalised to qualifying destinations.
Inverse: No.

Usage of destinations: N/A.
Destination type: Points.
Selection metric: N/A.
Use destination(s) that: Don't exclude based on selection metric.
Range: N/A.
Floor based constraint: No constraint.
Visibility constraint: No constraint.
Self include: No.
Route metric: Least number of segments.
Iterations: N/A.
Route ambiguity breaker: N/A.

Metric example 8: Space syntax-like 'choice' variable at radius 100 metres, based on metric graph based distance

Options:
Metric basis: Segment based.
Metric measure: Beteeness centrality (normalised) generalised to qualifying destinations
Inverse: No.
Usage of destinations: N/A.
Destination type: Segments.
Selection metric: (x,y) Euclidian distance
Use destination(s) that: Are less than or equal to 'range' based on selection metric.
Range: 100.
Floor based constraint: No constraint.
Visibility constraint: No constraint.
Self include: No
Route metric: (x,y) least metric graph distance.
Iterations: N/A.
Route ambiguity breaker: N/A.

Metric example 9: Space syntax-like 'integration' variable based on axial lines

Options:
Metric basis: Segment based.
Metric measure: Number of turns to destinations
Inverse: No.
Usage of destinations: Mean.
Destination type: Turnless lines.
Selection metric: N/A.
Use destination(s) that: Don't exclude based on selection metric.
Range: N/A.
Floor based constraint: No constraint.
Visibility constraint: No constraint.
Self include: No
Route metric: Least number of turns.
Iterations: N/A.
Route ambiguity breaker: N/A.

Metric example 10: Find the length of the subgraph, connected by shortest paths defined by least number of turns, between the basis point and all desks that fall within 50 metres of the basis point, for all points

Options:
Metric basis: Point based.
Metric measure: Length of subgraph of all minimum paths between qualifying destinations
Inverse: No.
Usage of destinations: N/A.
Destination type: Desks.
Selection metric: (x,y) (x,y) Euclidian distance.
Use destination(s) that: Are within'range' of according to the selection metric.
Range: 50.
Floor based constraint: No constraint.
Visibility constraint: No constraint.
Self include: Yes (Force).
Route metric: Least number of turns.
Iterations: N/A.
Route ambiguity breaker: N/A.

6.2.6 Notes on performance

We quicky explain performance differences that one can expect when calculating different metrics.

When calculating metrics the time taken depends on many things

- The size of the graph
- The density of the graph
- The metric measure
- The number and selection of qualifying destinations
- Any difference between the metric measure and the selection metric
- Any difference between the metric measure and the route metric

We assume the first two are fixed for any graph and so decribe the effect of the others

- The metric measure
  - This has probably the largest effect
  - Fastest measures are non angular deviation/ non turn based distance metrics and, in many instances, eigenvector centrality too (though noticeably not so in certain cases)

  - Using angular deviation/turn based metrics requires using larger graph structures and so the process is slower
  - Number of equivalent path metrics requires iteration of all paths and so is slower
  - Number of shortest paths through basis and betweeness centrality requires initial calculation of the weights and number of destinations between all points/segments so is slower
  - Length of subgraph requires the above plus consideration of all possible points/segments for each basis point/segment so is slower still

- The number and selection of qualifying destinations
  - Larger numbers of qualifying metrics will make the calculation slower for the number of equivalent paths, number of shortest paths through basis, betweenness centrality and most noticeably the length of subgraph metrics

- Any difference between the metric measure and the selection metric
  - If the selection metric for a distance based metric measure does not match the metric measure the code cannot optimise by utilising the same metric calculation twice so this makes the metric calculation slower

- Any difference between the metric measure and the route metric
  - If the selection metric for a distance based metric does not match the route metric then the metric measure cannot be simply identified as the weight returned by the main path finding algorithm, but must iterate along all possible paths that minimise the route metric to calculated the final metric measure. This makes the metric calculation slower.

7 Appendix A: What is actually going on

For the interested we give a very rough, hand waving guide to the algorithms/approaches at the core of this program.

The core algorithm that this program uses is known as Dijkstra's algorithm which, for a specified vertex on an graph with non-negative weights, and for a map of 'weights' between vertices connected by edges finds the minimum distance (being the sum of all relevant weights) needed to travel from the specified vertex to all other vertices in the graph. By use of Bellman's criterion one can keep a record, from any vertex in the graph, which direction one needs to go to keep minimising the distance to the origin vertex and thus iteratively deduce what the actual sequence of vertices that comprise the path which minimises that distance is. In this work we generalise allowing a list of vertices one might be able to move in the direction of at each vertex without violating the Bellman criterion so that *all* equivalent shortest paths can be enumerated from one vertex to another by systematically running through these lists of possible vertex choices in the appropriate sequence (Note: it may be tempting to do this

recursively, but be warned stack overflow can be reached faster than you might think!). Such an algorithm therefore allows us to derive the minimum distance between two vertices according to a specified set of weights and also all the sequences of vertices that comprise all paths that do so.

From this information a great many different things can be calculated. For instance we can define a path by minimising one measure of distance, but then trace out that path whilst summing up a different map of weights meaning we can separate the measured 'distance' of the path with the 'routefinding' that is used to define the path. Alternatively we can include or exclude destinations based on a given measure of distance. We can also find out how many shortest paths go through any given vertex using this information.

However, more information needs to be provided to explain all the workings of the program. The implementation of Dijkstra's algorithm uses a minimum binary heap (probably the single biggest way to improve performance) to effect a priority queue which replaces a naive search of all vertices. Technically a Fibonnaci heap is more effective still, but the improvement would be very slight for anything but the densest of graphs which are not those typically under consdieration in this work. Further performance in both time and space is achieved by building and utilising individual adjacency lists for each vertex in the graph (very effective for real world space graphs) with all weight information from a given vertex being stored in a hash table with keys equal to the index of the vertices to which the considered vertex is connected via an edge. This in constrast, for example, to storing such constructs in a matrix which would scale as $|V|^2$ in memory usage.

Next a description of all bases and route types is needed. To calculate segement based metrics a separate graph structure needs to be created where, in effect, each edge becomes a vertex and these new vertices are joined together with a new-type edge if the one old-type vertex that is part of the the old-type edge (which defines the new-type vertex) matches an old-type vertex that is part of another new-type vertex. To calculate metrics that utilise paths and weights based on angular change or number of turns new graphs are needed. The angular distance from a vertex to any other vertex it is connected to is strictly zero and so use of the original graph would assign weight zero for all paths. This however, does not reflect real world behaviour. The new-type edge based graph is favourable so we can use this as a starting point, but this too has problems, namely without distinguishing direction the 'walker' can backtrack and shortcircuit the graph producing weights of unphysically low reported angular and turn deviations. Consequently to deal with angles and turns we produce a new-type edge centric graph but produce a new-type vertex for each direction, meaning that we now have twice as many new-type vertices as there were old-type edges: one going in one direction and the other in the other direction. Using this we can calculate the angular/turn cost from any location to another, but are faced by the problem of having multiple new-type vertices equivalently representing our start and end points, or in the case of the original vertex basis, no new-type vertices representing our true start and end points. To achieve this we can add further new type vertices, but deliberately artificial ones which are designed to act as sources and sinks. They simply connect into and out of the wider graph,

but do so with zero angular/turn weight and do so in such a way that means they reflect the original vertex/edge basis behaviour. One final adjustment is needed, whenever a calculation is performed the weights *out* of these source sink vertices must be set to infinity if the source/sink vertex does not correspond to the start vertex of the calcualation (it must still have zero weight in in order to reach the sink destinations). In other words the start node must be set exclusively as a source and the destination nodes must be set exclusively as sinks otherwise a similar shortcut behaviour would be observed. All of this requires mapping from this new graph type into the old graph type which is achieved through a hash table with an object pair of indices acting as the key in this instance. Care must be taken when calculating metrics with these alternative graphs. For instance the number of paths through an old type vertex is equal to all the paths through all the new type vertices which contain the old type vertex in their definition, but the whole number must then be halved becasue the old type vertex will feature in the 'leading in' and 'leading out' vertices from every turn.

Betweeness centrality and subgraph calculations are calculated by using the common Bellman criterion which works sufficiently well for all tested graphs. Further work might include implementing Brandes algortihim.

To calculate the eigenvector centrality the power iteration method is used. Tests for convergence utilised the following. When converged we should have

$$Mx = \lambda x \tag{2}$$

such that

$$\lambda = \frac{\sum_j M_{ij} x_j}{x_i} \tag{3}$$

Since the sequence will not have totally converged we have therefore, an estimate of x_i, x_j and of λ which will be differnt for each x_i such that we can define

$$\lambda_i = \frac{\sum_j M_{ij} \bar{x}_j}{\bar{x}_i} \tag{4}$$

where bars indicate estimates. When the maximum and minimum values of λ_i fall within a user specified fractional tolerance the sequence is deemed converged.

8 Appendix B: Development and wish list

8.1 Development

Here we note that, despite binaries being available and what we suggest most users utilise, the program is open source and so users are free, if they wish, to modify, play with, adapt or improve the source code of this work so long as they obide by the relevant licensing agreements. They may also just wish to compile their own version.

With this in mind there are two main points to cover.

First is an acknowledgement by myself, the author, that the work represents something of an odyssey of learning under strong time constraints (the work being taken from concept to completion, by a solitary researcher, whilst part of a wider active research role in around 9 months with *all* concepts [basic GUI, serialisation, multithreading, OpenGL graphics, dxf handling etc. etc.] except graph based algorithms being learnt both from absolute scratch and very much 'on the go'). As such best programming practise, or the most modern APIs, is/are not always followed/used and the code is far from readable all of the time. However trivial a refactor might have been, once a feature was working and was tested, there was usually simply not enough time to implement it. To anyone wishing to adapt the code, use approaches etc. I say 'by all means' and welcome any such attempt, but perhaps might be tempted to follow it up with 'good luck!'.

Secondly I list coding information/dependencies additional to the source code provided (all are free software with open source or permissive licenses) if one wishes to compile a version for themselves

- The program is written in C++. Mac binaries were compiled with the version of the gnu c++ compiler version 4.2 included in the developer tool and the Windows version compiled with visual studio 2013.
- The main GUI functionality is provided by FLTK version 1.3.3 which can be obtained here http://www.fltk.org/index.php
- The dxf functionality is provided by dxflib which can be obtained here http://www.ribbonsoft.com/en/what-is-dxflib
- The serialisation and storage facilities are provided by boost which can be obtained here http://www.boost.org/

Down to the nuts and bolts of the compilation one would need to link against the following libraries (options are for different linking methods and operating systems)

- libboost_serialization.a/ libboost_serialization.dylib/ libboost_serialization.so/ boost_serialization.lib/ boost_serialization.dll
- libboost_iostreams.a/ libboost_iostreams.dylib/ libboost_iostreams.so/ boost_iostreams.lib/ boost_iostreams.dll
- libdxflib.a/ libdxflib.dylib/ libdxflib.so/ dxflib.lib/ dxflib.dll
- libfltk.a/ libfltk.dylib/ libfltk.so/ fltk.lib/ fltk.dll
- libfltk_gl.a/ libfltk_gl.dylib/ libfltk_gl.so/ fltk_gl.lib/ fltk_gl.dll
- libfltk_forms.a/ libfltk_forms.dylib/ libfltk_forms.so/ fltk_forms.lib/ fltk_forms.dll

If compiling your own version I recommend using the latest stable release of each, unless compliation is only successful with legacy versions (not the case at time of writing). All other functionality, is contained within the provided source code.

8.2 Wish list

There are three main areas which, had I had more time, I would have implemented in this software. Maybe one day I'll implement them, maybe not. Being open-source anyone is free to attempt these additions (see above comments).

- Generalisation to directed graphs. This in theory is a simple extension, though perhaps tricky to merge with the current implementation, but the only reason for its absense is lack of time and priority being set on the the type of environments the program was commissioned for which require only undirected graphs. This would require the following adaptions
  - Adpation of the editing system to allow for adding of directed and undirected links. I envisage the same functionalty, but with the user holding alt to add a directional link.
  - Adaption of all adjacency list consistency checks and updates based on user input (ie so that adjacency A-B is not considered a duplicate of B-A)
  - A sensible way of representing, visually, undirected links and directed links and an efficient was of identifying them. Perhaps simply adding direction arrows in each relevant direction would suffice as this could be done without searching for B-A when considering A-B
  - A facility to allow for selective link deletion. I suspect deleting all links between a pair of points would suffice as the user could always add directional links back in.
  - Most importantly: Careful application of a now non-symmetric adjacency matrix when constructing the graph structures (ie the index lists and hash tables of weights) in particular for the (themselves directional) extended graphs used for angular deviation and turn metrics.
  - Care would have to be taken with segment based graphs as one would be obliged to use a graph with twice the number of nodes (representing segemnts) like is done for turn based and angular based metrics. Consideration of this performance drop off and maybe an option to consider any drawn graph as directed or undirected might be preferable so that this difference can be optimized away.
- Adaption of the betweeness centrality type calculations so they use Brandes algorithm.
  - On all graphs I have used (up to 1000 vertices) with this program the amount of time taken was acceptable (5 seconds for non turn/angular route definitions and 38 seconds for turn/angular route definitions when using all points as destinations), but this could be improved particularly if larger graphs are used. Brandes algorithm is the natural step in this direction. Some care needs to be taken to ensure the algorithm still works with the extended graph representations required for turn/angular routes, but I can't think why not.
- Slightly more flexibility in the choice of destinations

- At the moment one can only select destinations from one type of location.
- Ideally I would include some functionality that allows selection of locations, based on the selection criteria, from each of a user selected set containing chosen sets of locations. For example, so that the closest, kitchen & printers could consitute the qualifying destinations. Or that the closest kitchen *or* printer could be selected from the set of kitchens and printers.
- I imagine implementing this would involve reimagining the destinaton type selector as either 'custom locations' or 'points' for desk and point based metrics which the custom option allowed the opening of a modal window containing checkboxes for each of the location types.
- The metric calculation/adaption would not be the challenging part here, but rather the implementation of another layer of complexity into an already quite dense metric dialogue.