INTELLIGENT ARCHITECTURE

new tools for the three dimensional analysis of space and built form

30

Alan Penn, Ruth Conroy, Nick Dalton, Laura Dekker, Chrion Mottram and Alastair Turner
University College London, London, England

30.1

0 Abstract

This paper describes Pangea, a new software workbench developed to enable the flexible analysis of three dimensional 'worlds' composed of objects and the spaces between them. By developing a simple application in which 3-D shapes can be created and edited, and in which each shape holds arbitary length lists of attribute data and a software 'script', Pangea provides a customisable tool for the analysis of 3-D spatial relations. Descriptions are given of the use of Pangea to develop a range of analytic and design support tools including the development of isovists and axial maps within three-dimensional models.

Keywords: configuration, computer, design, Pangea, programming

Alan Penn
The Bartlett School of Graduate Studies
(Torrington Place Site)
University College London, Gower Street
London, WC1E 6BT
England
tel: (44) (0) 171 391 1739
fax: (44) (0) 171 916 1887
e-mail: a.penn@ucl.ac.uk

1 Introduction

Although in principle space syntax techniques can be easily adapted to represent and quantify aspects of the three dimensional form of built space, there are at present relatively few examples of this type of analysis. The analysis of three dimensional built space is of interest for a number of reasons. The first is that although we are constrained mainly to move in two dimensions the world that we perceive is three dimensional. If we are interested in the way that buildings and cities are 'intelligible' to us then it is likely that an analysis of the three dimensional scene will be relevant. Second, although many of the social aspects of a building's function may depend essentially on two dimensional relations in plan, the function of buildings as shelters, environmental and climatic modifiers, the way they cast light and shadow and their structural function depends on their three dimensional configuration. In this sense, their total 'architecture' must respond to aspects of three dimensional configuration, and there can be little doubt that this affects strategic design choices (Hillier & Penn, 1993). Third, although most 'conventional' multi storey buildings effectively 'stack' two dimensional plans on top of each other, architects are increasingly experimenting with ways of dissolving the barriers between floors, using atria, ramps, sloping floors and complex circulation strategies. It is likely that in order to analyse and understand this sort of architecture that we shall be driven to analyse three dimensional configurations of space. Whilst all these reasons may seem relatively prosaic, it is certainly true that the architectural medium is three dimensional and any form of analysis that really aims to unpack architectural design fully will need eventually to represent and quantify three dimensional formal and spatial relations.

One of the main barriers to the analytic investigation of three dimensional aspects of spatial configuration up until now has been the complexity of the analytic problem. While representations remain fairly simple it is possible to construct examples by hand, or at least to verify computer calculations by hand. However, as configurations become more complex, even two dimensional overlapping convex spaces become

impossible to handle without computers. Any of the potentially more interesting forms of representation, such as isovists, become completely unmanageable for realistically sized buildings.

At the same time, the range of possible forms of analysis in three dimensions is vast. If anything it is possible to think of a larger number of potentially interesting forms of representation and analysis for three dimensional than for two dimensional space. And yet we know from experience that it is almost impossible to judge in advance which the empirically 'useful' forms of analysis will be. This creates a problem for the would-be analyst. Where should they begin?

The Pangea 3-d workbench recently developed with EPSRC/DTI funding is one response to this problem. Given the investment needed in programming software for any form of three dimensional analysis we decided to separate out, as far as possible, the representation of three dimensional form, its construction, editing and viewing, from the analytic parts of the software. We decided also that the analysis should be 'user enhanceable'. That is, that the user at run time should be as free as possible to amend and even create completely new forms of analysis without having to rewrite - or even modify and recompile - the source code. The intention was that people who do not have the programming skills to create three dimensional graphics packages and user interfaces should be able to interact with and interrogate three dimensional built form in ways that we could not possibly anticipate. In this sense our aim was to create a 'tool for thinking with' aimed at analysis of three dimensional form and space.

In this paper we first review 'conventional' space syntax and in particular the type of representations or 'analysis maps' that must be developed if we are to automate the procedure. Next we look at a fundamental dilemma at the basis of space syntax methodology. Many of its successes seem to lie in the move from continuous space, which is different at every point, to discrete graph representations in which individual 'spaces' are held somehow to be uniform across a defined but finite area. We define a methodological need, therefore, to be able to treat three dimensional built form in terms of the continuous space pattern it defines which gives rise to different views at every point, at the same time as being able to develop from it analytic representations which can be considered as bounded volumes of space and which can be treated as the elementary nodes in an analysis of spatial relations. Then we turn to the Pangea workbench, describing its main features and the scripting language on which it is based as one attempt to respond to these methodological needs. Finally we describe the use of the workbench to create tools for spatial analysis. These descriptions are purely by way of example. The uses to which Pangea has been put so far only scratch the surface of what is possible.

2 Spatial representations

Conventional space syntax is based on a three step procedure. First a representation is constructed; a map which reduces continuous open space to a finite number of discrete 'elements' or 'spaces' which can somehow be considered to be equivalent to each other. Conventional examples are axial, boundary and convex maps. Next a graph is constructed in which each 'space' is represented by a node, and in which nodes are linked together if there is some relationship between the spaces. The

relationship is defined according to a rule such as the existence of a relationship of permeability or visibility between the two spaces. For instance, if a space is a room and the relationship we are considering is permeability then a doorway between two rooms would be represented as a link in the graph. Finally, we measure the pattern properties of the graph constructed in this way. Depth and rings in the graph are commonly measured in both local and global terms. Local measures consider the way a space and its immediate neighbours relate and global measures consider a space in relation to all other spaces in the system. The most empirically useful global measure of depth is integration radius n. This might best be visualised as a measure of the shape of the justified graph (in terms of its mean depth) from the point of view of a particular node.

It is worth considering the construction of the representation maps carefully. During the mid to late 1980's the UAS devoted considerable effort to the automation of this step of the process. This was seen to be of theoretical as well as practical importance in that it would be hard to maintain that a methodology was objective unless it could be satisfactorily automated. The main findings covered here were arrived at in the 'Space Syntax as an Interactive Design Tool' project between 1985-87, and were embodied first in the Syntactica software package (1987) and then rewritten in SpaceBox (1989). The theoretical value of these techniques has received its first major test with the publication of Space is the Machine, and has yet to deliver real empirical results although the trials that have been carried out to date are promising.

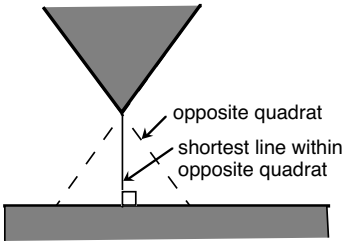


Figure 1. The simple rule to give a minimal 'fattest' convex subdivision of space.

Put simply, the research discovered that conventional representation maps - the fewest axial line map and the discrete convex space map, which were simple enough for human researchers to construct and work with, were in certain cases undecidable by computer. A number of approaches were tried, first to produce discrete 'fewest and fattest' convex spaces. The algorithm to do this is apparently simple entailing finding the nearest point on the surface of a building within the opposite quadrat for each convex vertex (Figure 1).

The problem is that under certain conditions the simple algorithm needs to be modified if the requirement to derive as few convex spaces as possible is to be met. In particular there are occasions in which it is possible to reduce the number of convex spaces by relaxing the requirement for the edge between spaces to be shortest which is entailed by the 'fattest' rule. This alone would not necessarily have made the procedure anything but somewhat inelegant, however there are occasions in which 'second pass' modifications to eliminate an edge and which result in other edges being re-aligned, can have further implications for yet another edge in the system, and the algorithm, faced with particular morphological conditions, can enter an infinite loop. This appears to be a direct result of the two competing requirements - fewest and fattest. Whilst it might seem possible to develop ever more sophisticated rule systems to decide on the 'optimal' subdivision on the basis that human researchers appear generally to be able to come to an agreement on this, it is possible to show circumstances in which a particular convex subdivision is actually undecidable. The obvious example is the simple cruciform room in which each wing is exactly the same width and depth (Figure 2).

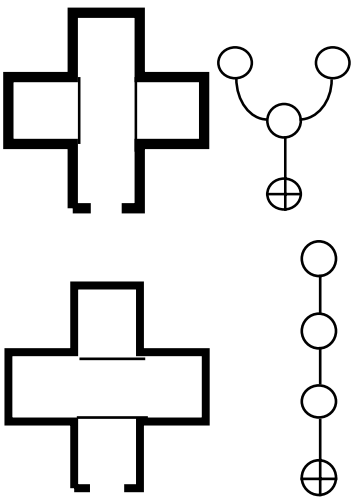


Figure 2. In the simple cruciform building which wing 'wins' only matters when we consider it as a part of an asymmetric global system, say with an entrance on one wing only.

In this case it is impossible to decide which wing is to 'win' and become a single space and which is to lose and become a pair of spaces. At first sight this might appear to be a trivial case, however when the system is automated the computer will either be in a position of randomly assigning the 'winner', or the algorithm will effectively be dependent on the order in which the original plan is digitised. In either case, presented with exactly the same plan data two researchers could in principle arrive at different mappings and results, and this would have dire consequences for the claim of 'reproducibility' of the methodology. The problem becomes critical when we analyse perfect grid cities.

30.4

The solution to this paradox was to eliminate the requirement that convex spaces be discrete, and to allow them to 'overlap' (Figure 3). In one step this allowed a much simpler algorithm, elimination of the undecidable situation, as well as making 'intuitive' sense of the nature of space - there are situations, as in the central space in the cruciform room, where we are effectively in two convex spaces at once. It became clear that there was a much closer relationship than we had previously realised between built form and spatial subdivision, and between the continuous view of space in which every point is different to all others and the discrete subdivision of space into 'overlapping' convex areas of essentially similar points. It is the faces of buildings and their vertices that define convex spaces and areas of overlap.

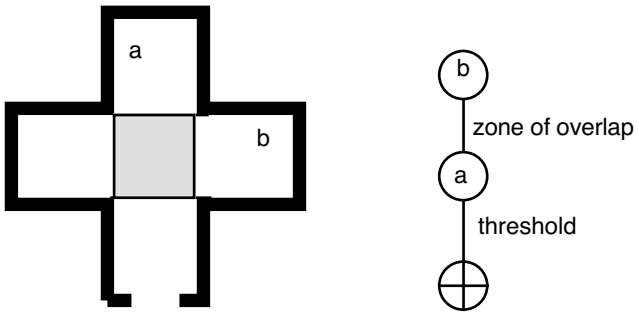


Figure 3. Overlapping convex space in which all spaces are maximal and overlaps are considered as relations or links in the graph.

When we turned to the automation of axial mapping we were surprised to find almost exactly the same situation facing us. Although under certain conditions it was possible to define algorithms to produce very convincing 'fewest/longest' line maps, there are a class of plans where it is possible to show that there is no single 'correct' mapping. The procedure we developed entailed two stages. First we construct all possible longest lines passing through the open space of a configuration, and then we remove shorter lines that made no additional contribution to the reduction of depth in the map, to leave the fewest, longest lines. The production of all candidate lines is relatively simple, if exhausting. Longest lines will tend to be those that pass diagonally through space, just touching and passing-by acute vertices on built form. It turns out that there are also occasions where longest lines pass along the face of a form, or where they terminate at a concave vertex on a form. To produce the full set of all possible candidate lines merely requires one to join all pairs vertices to each other that can be joined through open space and to extend lines past the vertices, if this is possible without passing into a solid form, until they hit the nearest solid object. In fact we can reduce the task slightly by only keeping lines that extend beyond at least one of their generating vertices, unless both vertices are concave (the longest lines within a closed square room will be from corner to corner). The problems occur at the second

stage of the procedure, where we wish to remove lines to produce the 'fewest' lines needed to cover the system. At this point it turns out that there are particular cases, especially in geometrically regular systems, where two lines may be of exactly the same length and so it requires additional rules to be applied to define which line should be eliminated. The most apparently reasonable rules pass through a definition requiring the elimination of lines that make fewest depth reducing connections between other pairs of lines in the map. The problem here is that this turns out to be a global problem, where the decision to eliminate a line on one side of the map can have effects on subsequent decisions elsewhere in the system. It is possible to show that these rules are essentially dependent on the order in which the problem is processed, and this means that given precisely the same map, but with its vertices represented in different orders the algorithm could come up with more than one distinct 'fewest line' axial mapping.

Our response to this paradox was to adopt the algorithmically simplest mapping - the all line map - and to do away with the elimination stage all together. Again it seems that this map, which represents all possible lines of sight and movement, may actually be more closely related to our perceptual experience of space, as well as to certain fundamental mathematical properties of spatial configurations. For instance, the pattern of integration in all line maps distinguish between the segments along a long axial alignment in a grid, with those at the centre being more integrated than those at the ends of the alignment. It is also possible to show that all metrically shortest routes between distant pairs of points in large systems pass predominantly along segments of lines in the 'all line' map.

Two important realisations followed from this research. The first was that apparent simplicity of the representation - discrete convex maps and fewest line maps certainly look simpler than overlapping maps or all line maps - did not necessarily correspond to algorithmic simplicity. The second was that the simple algorithm - an analytic equivalent of the 'short model' - could give rise to a representation that was intuitively richer and perhaps in closer accord to our perceptions of space. This is somewhat surprising, and suggests an important distinction is to be drawn between reductionism in the representation itself and methods based on applying the principle of parsimony to the process of analysis. It seems plausible that algorithmic simplicity may equate to an 'antireductionist' richness of representation.

The principles involved in this new suite of representational maps began to strike chords with some of the fundamental notions which gave rise to 'syntax' in the first place. The early work on generative modelling of beady ring systems turned around the notion of the 'short model' and algorithmic simplicity in a process. Why invoke a longer model when a short model accounts for the phenomenon? In this there is a fundamental similarity between the generative and analytic approaches to syntax which turns on the distinction between 'forms' - the elementary cell-space diad, for instance - and 'rules' - such as the rules of aggregation which determine the placement of the next cell in the process. Part of the stimulus to develop analytic representations lies in the hope that by representing and quantifying the pattern properties of spatial systems it may prove possible to retrieve the 'rules' governing aggregation that determine their global form.

In this way analysis aims at an explicit form of description retrieval in which the simpler the analytic algorithm, the greater the range of possible rules that can be retrieved, and possible morphologies that can be dealt with. At the centre of the development of the new representations were the anomalies and paradoxes faced by the 'conventional' representations. And yet there still remain at least two issues - possible anomalies - which confront even the new representational maps. These are the resolution of spatial deformations with which we deal, and the related issue of continuous smooth curving form. Although the majority of built

form consists of straight facets and acute vertices when we are faced with continuous smooth curves we have to approximate them by polygonal forms.

At the basis of these issues lies a fundamental dilemma faced by space syntax methodology. Many of its empirical successes seem to lie in the move from continuous space, which is different at every point, to discrete graph representations in which individual 'spaces' are considered to be uniform across a finite area. Yet our experience of space as an observer is largely of smooth and continuous transitions and changes as we move. It seems entirely possible that the discrete graph is yet another example of the apparently simple, yet algorithmically more complex representation. The fact that we have to convert all continuous curves to faceted polygons, provides an example in which it would be possible to devise situations where the resulting maps could differ depending on the starting point and order with which one generated the facets. This shows that the step is essentially arbitrary and suggests that it should ideally be eliminated.

There appears to be a need, therefore, to be able to treat three dimensional built form in terms of the continuous space pattern it defines which gives rise to different views at every point, at the same time as being able to develop from it analytic representations which can be considered as bounded volumes of space and which can be treated as the elementary nodes in an analysis of spatial relations. However, eventually we shall need to be able to move around and analyse three dimensional form in arbitrarily small steps. Taken together with the fundamental distinction between 'forms' and 'rules' this gave rise to a specification for a new kind of analytic tool or 3-d 'workbench' as well as to the first analytic uses to which such workbench could be put. In particular Gibson's (1979) notion of the 'optic array' and Benedikt's (1979) translation of this into the 'isovist' provided useful tests of the ability of the workbench to help develop and carry out new forms of analysis.

3 The Pangea workbench

The Pangea workbench is a simple programmable 3D CAD application (Penn et al 1996a; 1996; 1995). The application is geared to producing 3D models, and has no conventional 2D drawing capability. The user interface is straight forward (Figure 4). When the application is launched and a new 'world' opened a toolbar appears at the top of the screen containing a number of simple primitive solid objects such as cubes, cones and spheres, as well as more complex tools for building 'walls' and extruded polygonal shapes. A statusbar at the bottom of the screen gives feedback to the user on his actions and precise metric information on mouse location, or the location and size of selected shapes. The main body of the screen contains four windows each of which shows a view of the start-up world containing a floor, a light source and the cameras (whose view is shown in the other three windows).

Shapes are created in the world simply by selecting the appropriate tool and then clicking and dragging on the floor to set the dimensions of the shape. Once a shape has been created it can be selected and moved, resized or rotated merely by dragging at the appropriate handles. Other tools in the toolbar allows a shape's vertices to be selected and moved independently ('tweaked'); new vertices can be added and 'windows' cut through objects. Once a world has been constructed it is possible to steer a camera through it in a real time walk through.

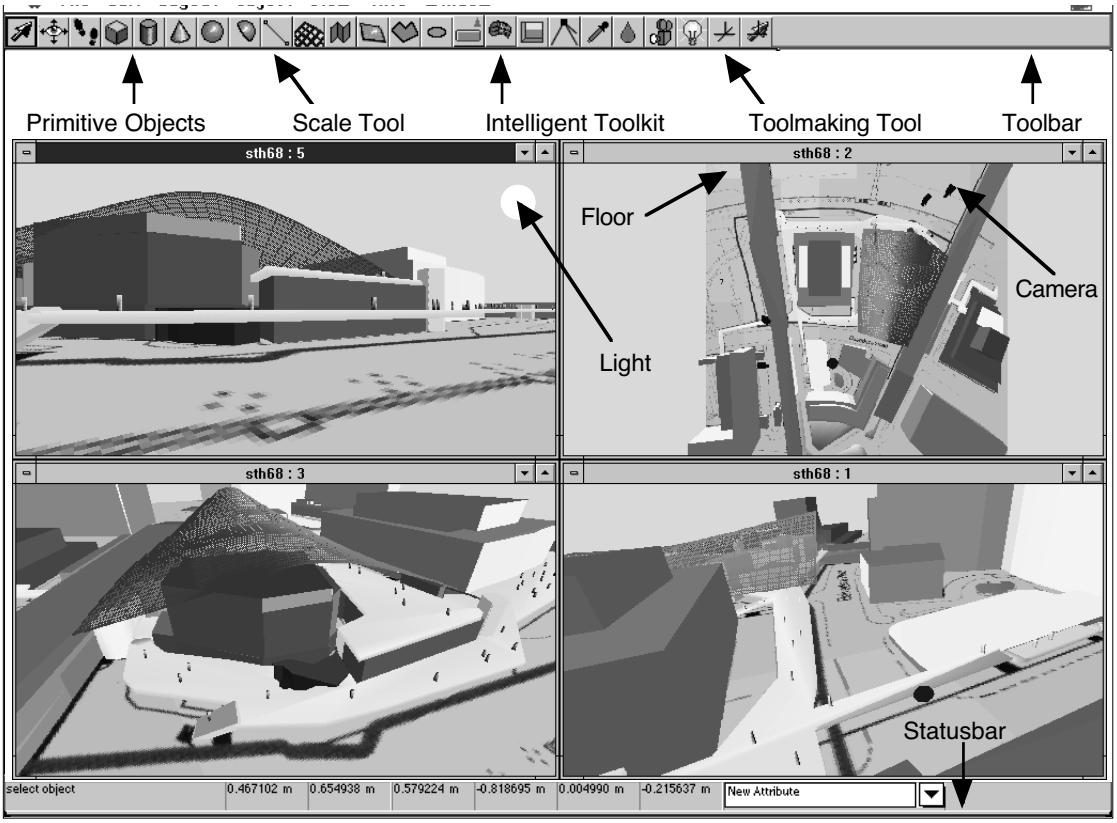


Figure 4. The Pangea user interface.

3.1 Attribute properties and scripts

Every shape in the world has a list of properties which we call its 'attributes'. These can be viewed and edited in the statusbar. Attributes include a shape's location and the dimensions of its bounding box, its colour and opacity, the type of tool that created it, its number of faces and a unique personal identifier (PID). The list of attributes can be extended by the user, and any type of data can be stored as an attribute and modified by the user.

Pangea differs from other 3D CAD packages, firstly, in that it is relatively easy to use, and secondly, in the way that it is possible to programme the world by allowing every 'shape' to carry a 'script' which gives it a behaviour. In this it is similar to programs like HyperCard and SuperCard, but with the difference that instead of buttons or graphics we have fully three dimensional shapes. A Pangea script is based on the concept of 'message handling'. Every object in Pangea including each shape, window and the Pangea application itself, has a script whose job it is to respond to messages from the user or from other objects. For example, when the user clicks on a shape with the 'browse' cursor the shape receives 'mousedown' and 'mouseup' messages. If the shape's script contains a mousedown or mouseup message handler it will carry out its instructions. A message handler takes this form:

```
on mouseup
      beep
end mouseup
```

If the shape's script does not have the appropriate handler it passes the message up to the current window, which in turn will either execute its instructions if it has the handler or pass the message up to the Pangea application's script. There are a large

number of standard messages that the scripting language can respond to ('mouseup' is one), but a user can simply declare a new message by writing its handler into a shape's script as follows:

```
on jump
      moveby 0,.5,0
      paint
      moveby 0,-.5,0
      paint
end jump
```

The shape will now respond to the message 'jump', when it receives it, by jumping up and down. If we draw a second cube and give it the following script it will send the 'jump' message for us:

```
on mouseup
      send jump to shape id 1492 [use PID of the cube with the 'jump'
script]
end mouseup
```

Messages can be sent to and received from four main sources: the user - mouse and key board events are all sent as messages and a dialogue box can be used to ask the user explicit questions; the system - for instance when two objects bump into each other an 'intersected' message is sent to each of them; other objects through scripting, and; external applications. This allows data to be sent to and received from spreadsheets, CAD packages and databases as well as purpose made external applications.

The ability of shapes in the world to send and respond to messages is at the core of Pangea's functionality. A wide range of behaviours can be constructed in Pangea worlds, including the ability to analyse the properties of two and three dimensional spatial configuration. It is by means of messages and the responses they engender in shapes that 'rules' can be attached to the behaviour of 'forms'.

3.2 Structuring information and collection

In order to handle large and relatively complex data sets we had to develop ways of allowing the user to structure that data flexibly. It seemed likely that apart from data on building form, a dataset might contain different types of object - furniture, shell and core might all need to be looked at separately and together for instance - or empirical data on patterns of space use or occupancy. To help do this Pangea incorporates two features targeted at the information creation and storage requirements of 3-d models: a method for grouping components into sets, and an automatic inferencing tool for clustering similar objects together.

In traditional CAD drawing sets 'layering' conventions are used to store information of different types separately, and to give access to appropriate parts of the whole information store to allow concurrent work by different members of the design team. Layering effectively reproduces the paper systems that predate CAD in which different layers of tracing paper would be used to store different categories of information; steelwork, concrete work and masonry, for instance, or piped water services, drainage and wiring. By storing these on separate layers of paper different people could work

on the same part of the building at the same time, the data could be structured into smaller and more manageable groups and visual clutter could be reduced. The problem is that by separating the different layers conflicts between them become harder to detect.

Pangea has replaced the concept of the layer with the concept of the 'collection'. Collections behave rather like mathematical sets. A single shape can be a member of a number of collections - a partition may be in the 'impermeable', the 'internal' and the 'full height' collections all at once. This makes for a far more flexible way of storing data without the need to reproduce new object instances in different layers so reducing the size of the 'world'. Collections can be hidden to remove visual clutter, and their members can be addressed directly - it is possible to iterate through collections and send script messages to each member in turn. They have turned out to provide an important method for structuring information, and yet are generic in that they do not impose a particular convention, but allow almost any indexing system to be implemented.

3.3 Clustering

The second feature tackles a critical problem in selecting an classifying the objects in a dataset. The approach we have adopted in Pangea is to build a component inferencing tool which can begin to recognise similar and different groups of components, based on their geometry and any other attributes. The clustering tool allows the user to cluster objects in the world into groups according to their similarity in an n-dimensional vector space in which each attribute is a dimension. Figure 5 shows the clustering tool dialog which allows selection of attributes and clustering

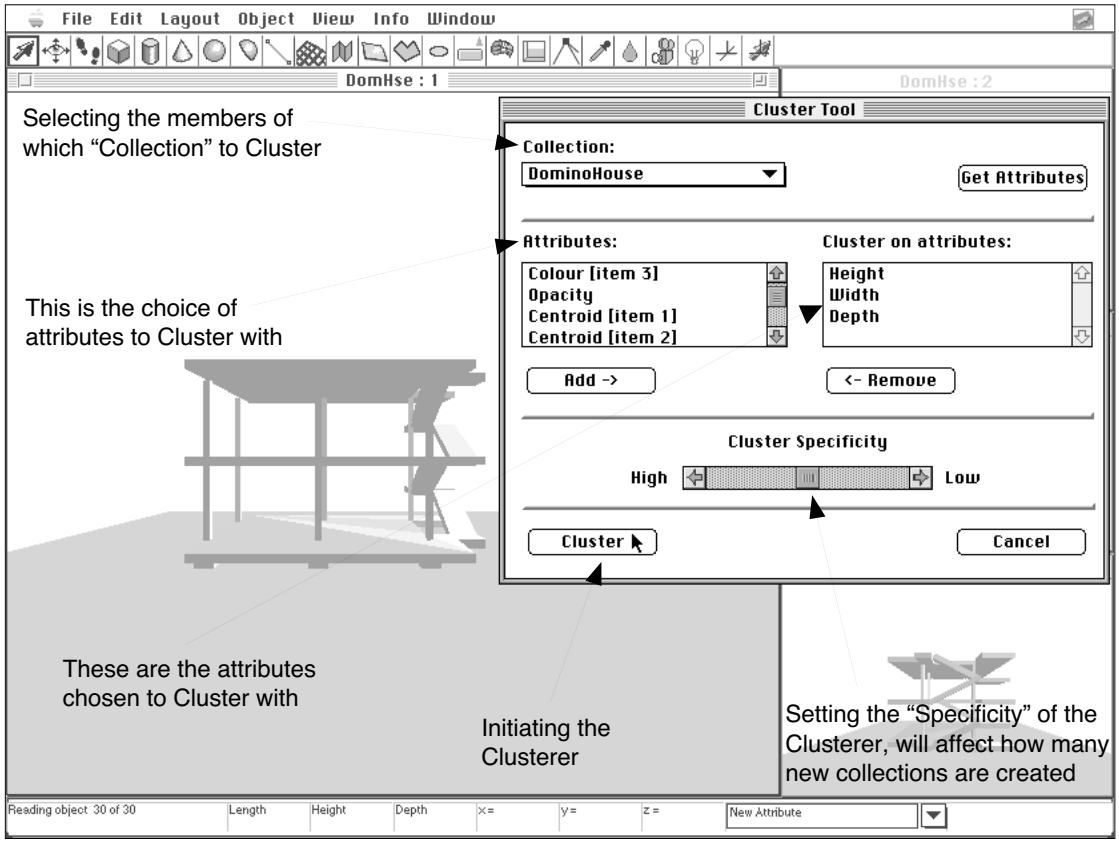


Figure 5. The clustering tool dialog box.

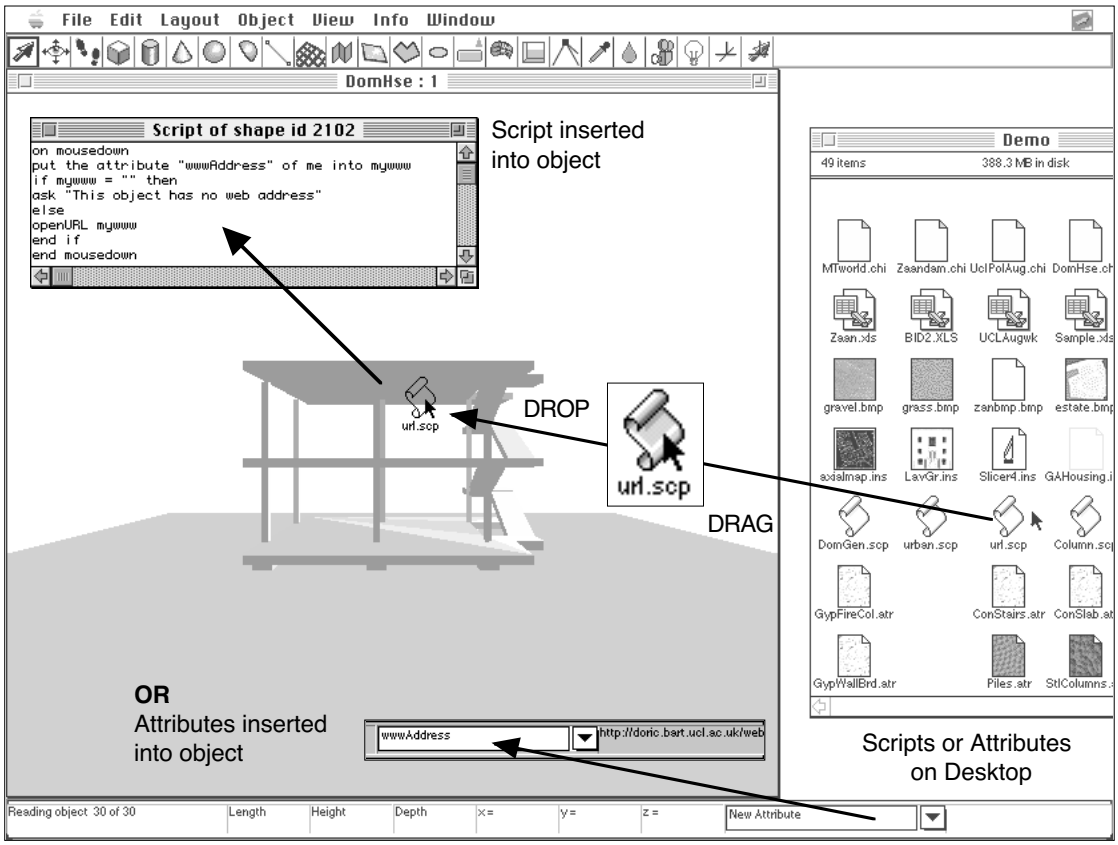
specificity. The tool automatically retrieves the attributes common to a collection, and the user chooses the particular attributes which they feel help define the set of objects they are interested in. For example, the user may feel it is helpful to start by looking at the overall dimensions of objects, and so cluster the objects in the world on the attributes of height, width and depth. In a building this would sort out the tall thin, the wide flat and the cuboid components, and insert each set of collections according to aspect ratio.

It may also be useful to group by other attributes, for example, colour. The clustering tool allows the user to cluster on any set of attributes at the same time: so they may cluster on dimensions, colour and a user defined attribute such as component cost. Alternatively, they may cluster on the colour independently of the dimensions, so the original set of objects is sliced in different ways. The user can then combine these sets as they desire. The user may choose an existing subcluster, such as 'Large Objects' and cluster within that set for colour. All these possibilities make the clusterer a flexible tool for searching and grouping operations of the sort that need to be done when moving from the purely geometric model needed for visualisation towards the intelligent type of model needed for analysis.

3.4 Drag and drop

Pangea supports 'drag and drop' for properties such as attributes, scripts and textures, so the process of adding information into a collection of objects is very simple (Figure 6). This makes it possible to select each collection in turn - say the 'tall thin' components and give all the components attribute information needed for, say, columns in one drag and drop action. The wide flat components might be given slab attribute information, or an appropriate script, and so on.

Figure 6. Drag and drop of attributes, scripts, textures or whole components such as the 'isovist camera'.



An extension of the 'drag and drop' metaphor allows the user to export a whole building or component, including specific attributes, textures and scripts as a file. This file can then be dragged from the desktop into a world and inserted complete in one action. An 'on inserted' message handler in the component's script allows it to set up specific requirements it may have of the 'world' into which it has been placed. This allows an inserted component to create appropriate collections required by its script, or to make checks on its placement and 'snap to' other components or grid locations. This allows one to develop specific objects that carry out a kind of analysis and then to drop them into a model world in order to execute the analysis.

3.5 Communications with external applications

In order to exploit existing software and to make the most of the potential offered by the Pangea 3-d interface without having to completely rewrite existing analysis applications, Pangea has a range of methods for importing and exporting data, including reading and writing files from scripting, DXF import and export to allow file transfer to most CAD packages and open communications using Dynamic Data Exchange (DDE) and AppleEvents. Using DDE items of data can be simply imported and exported using the 'fetch' and 'post' operators in the scripting language and this is useful for simple communications with spreadsheets or databases (see the urban masterplanning example below). The scripting 'tell' operator has been devised to provide a simple and explicit method for setting up sequences of interactions with external applications. For instance, scripts can be run in one of two ways, either directly, although this is limited to scripting calls that have one or less parameters, or by loading the whole script as the data part of the communication and the keyword 'runscript' as the command. An example is given below of the kind of commands Pangea can accept as a Server. First, a connection is opened to Pangea with a specified document, then data are sent in multiple and single parameter forms, finally the connection is closed.

```
on mousedown
      tell "pangea","aab.chi" --open connection with Pangea 'aab.chi'
            senddata "send rotateby to shape id 10156 with 0,10,0"
         to "runscript"            --multiple parameters
            senddata "0" to "paint" --only one parameter
      end tell                     --close connection with Pangea
end mousedown
```

A feature of DDE that we have found valuable is that it allows communications with Visual Basic applications. This gives access to a simple interface for 2-d GUI design and construction. Dialog boxes, text fields and simple charts are all easily set up in Visual Basic and can be directly linked to Pangea both as client and server.

3.6 The 3-d crosshair and the scriptable tool

In order for an application to be simple to use it is important to minimise the number of tools and modes available to the user. However, in the kind of applications envisaged for Pangea, being too restrictive could make it hard for particular user requirements to be fulfilled. In order to try and overcome this we have allowed access within the scripting language to a 3-d crosshair cursor which 'sticks' to, and tracks over, surfaces of objects in the world.

30.12

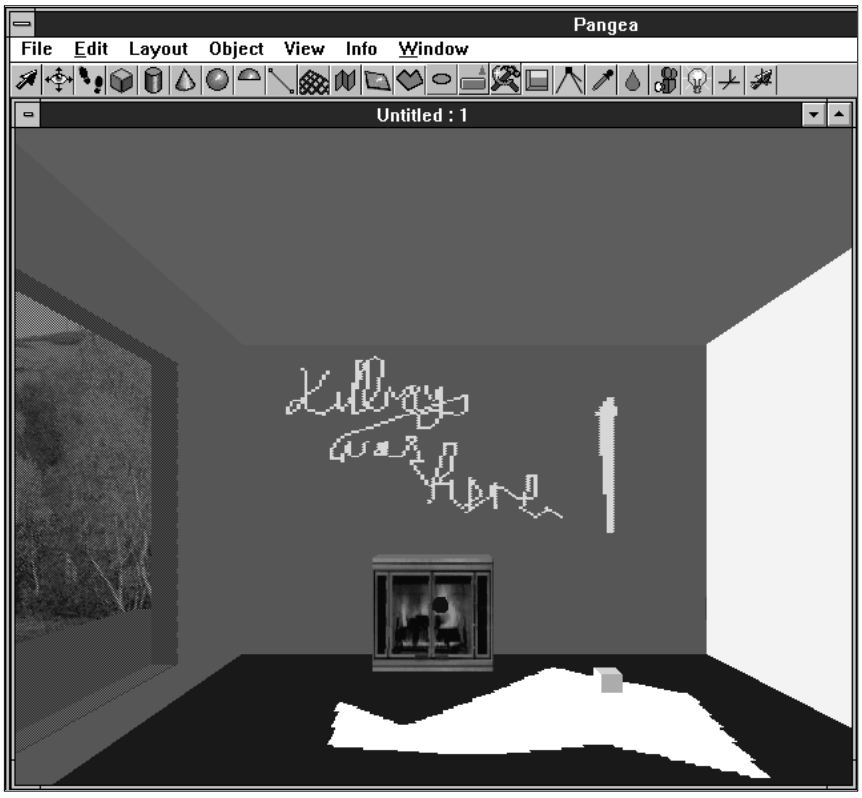


Figure 7. In-situ bitmap editing using the scriptable tool.

By catching mouse and keyboard events via scripting the user can access information about the 3-d world underneath the crosshair. For instance, the location of the point in the world underneath the crosshair, the object underneath the crosshair and the index of the closest vertex to the 3-d point can all be caught. One also receives messages when the crosshair moves from one polygon to another over the surface of objects. Using this information it is possible to create interactive tools to accept mouse and keyboard events and to create new primitive objects or new kinds of operation. Tools have been developed in scripting that 'paint' on the bitmap on the surface of shapes in the world, Figure 7, and that create quite new kinds of tools such as lathes or blankets. A simple script showing how one can access 3-d information about the world when a user presses the mouse button and use this to create a new tool is described in Penn et al 1996, and demonstration examples are given in Penn et al 1996a.

This feature provides a means by which users can create their own tools, and implicitly their own applications, using Pangea as the basis. This could be especially flexible if used in conjunction with the 'dynamic data exchange' methods described previously.The Pangea workbench is a simple programmable 3D CAD application (Penn et al 1996a; 1996; 1995). The application is geared to producing 3D models, and has no conventional 2D drawing capability. The user interface is straight forward (Figure 4). When the application is launched and a new 'world' opened a toolbar appears at the top of the screen containing a number of simple primitive solid objects such as cubes, cones and spheres, as well as more complex tools for building 'walls' and extruded polygonal shapes. A statusbar at the bottom of the screen

3.7 *Optimisation for urban masterplanning*

A range of test applications aimed at strategic design problems have now been built using the workbench. For instance, Pangea applications have been developed for engineers Battle McCarthy and architects Richard Rogers Partnership to investigate environmental implications of building density and use in a large urban developments. In a masterplanning project for Zaanstad, a town on the northern periphery of Amsterdam, an analysis of likely energy demands resulting from land use mixes was carried out in conjunction with conventional space syntax analysis and advice on spatial structure (Battle McCarthy, 1996). By linking a model in Pangea to a spreadsheet assessment package, information on land use and density acquired directly from the model was used to produce expected daily and annual profiles for population, electricity, cooling and heating demand. Figure 8 shows the Zaanstad masterplanning model linked to population prediction calculations and those for energy demands through time of day in an Excel spreadsheet, on the basis of quanta of different land uses assigned to building blocks. As land uses are changed by changing a block's colour, or building height is changed, the spreadsheet calculates the overall impact on the energy demand profile through time of day. The analysis was fed back in graphical form to the designers and so allowed a series of factors to be taken into account at once during the early 'strategic design' stage when both building form and land use distribution were being decided.

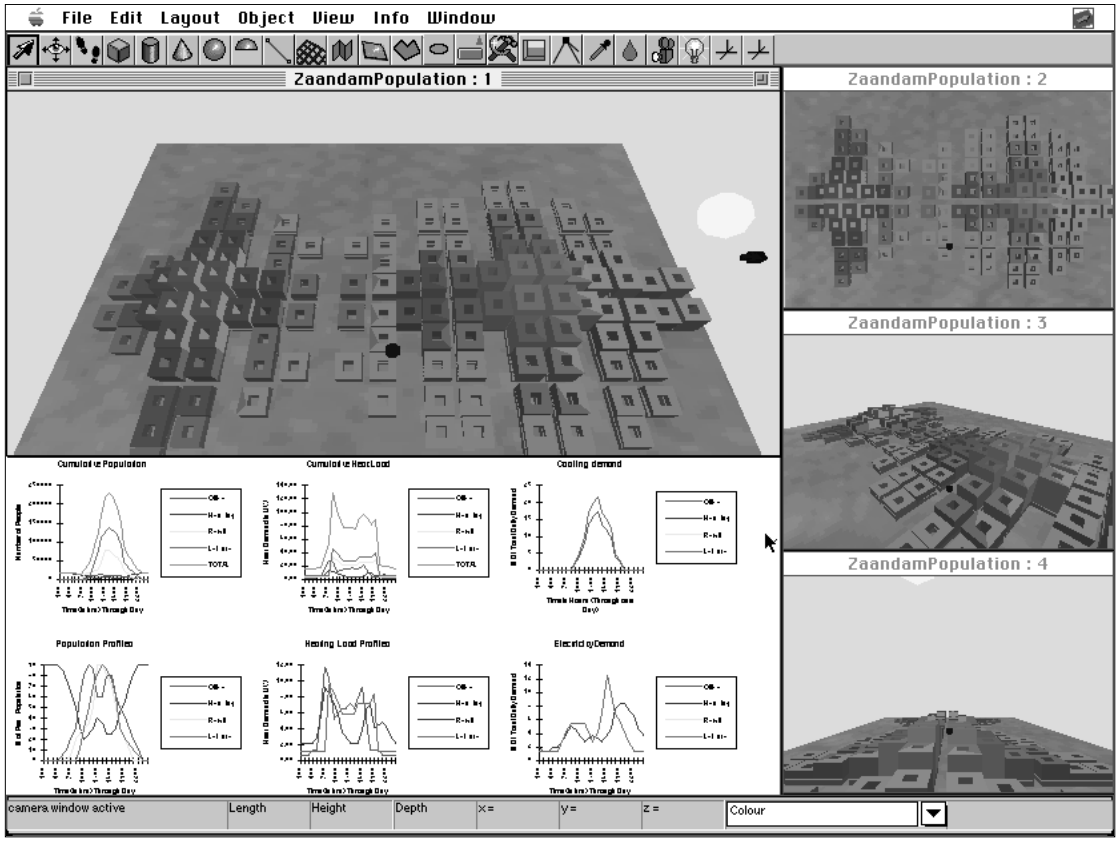


Figure 8. The Zaanstad masterplanning model.

3.8 Intelligent toolkits

Although the system described above allows the user to assess the impact of their decisions, due to the number of building units involved, the time involved in assessing even a small range of the possible layout options would be enormous. In order to help tackle this type of problem Pangea includes a range of 'intelligent' toolkits which can be set up and called using scripts. Three main types of tool are available: Genetic Algorithms which, analogously to genes and the 'survival of the fittest', mutate, recombine and test successive populations, and so evolve relatively 'good' solutions to particular problems; Neural Networks, which are loosely based on the way that neurons connect up in the brain, and lend themselves to pattern spotting, classification and simple control procedures; and, Morphic Searches which provide very simple way of finding optima in rugged fitness landscapes. In addition to these, scripting lends itself to writing simple and fuzzy rulebased systems.

For the Zaanstad project a Genetic Algorithm (GA) was developed to investigate a large number of land use allocations and to 'evolve' these to produce a 'good' land use plan based on the features assessed by the spreadsheet. Genetic Algorithms in Pangea are objects, similar to a shape - a cube or a sphere - and are scripted in exactly the same manner. The land use of each building block in a prospective plan was easily encoded for a GA since all the information needed to evaluate a design had previously been passed to the spreadsheet. The fitness of the plan was assessed using the spreadsheet and so the details of the fitness function are relatively open to scrutiny and modification by the engineers who wrote and are comfortable using spreadsheets. We have found that the discipline imposed on the user by the spreadsheet as a means of defining the fitness functions is an effective means of satisfying the constraints for using a GA - that we can encode the problem (the specific properties of a design in this case), and that we are able to assess the value, or fitness, of an encoding.

Once a decision has been made to use a GA from the Intelligent Tools, the Pangea scripting language offers the user an easy method to interact with the GA (see example script below). The user can adjust the attributes of the GA as necessary (such as deciding the number of generations the GA should run for or the size of gene pools). However, relatively little understanding of the internal workings of the GA code is required of the user. They are encouraged to experiment with the system through the simple scripting front end. The user must also provide a fitness function, called GetFitness. This allows the user complete freedom in the way they specify the fitness, either internally within scripting or through calls to external applications or by means of a combination of these.

For the Zaanstad problem, the GA optimised a combination of population flux and electricity demand to maintain a constant balance throughout the day. As such, this is only a simple example problem. Battle McCarthy describe how they intend to extend their analysis to cover aspects such as balancing the cost of social housing while maintaining a high quality of public space. This involves incorporating multi-objective analysis in which several criteria are optimised at the same time (Kingdon & Dekker, 1995).

```
on mouseDown      -- Initialise the GA attributes
   set the attribute "maxgenerations" to 50
   set the attribute "numpools" to 1
   set the attribute "poolsize" to 40
   set the attribute "numchromosomes" to 1
   set the attribute "numgenes" to 256
   set the attribute "genetype" to "integer"
   set the attribute "minvalue" to 1
   set the attribute "maxvalue" to 4
   set the attribute "crossover" to "Uniform Crossover"
   RunGA    -- Run the GA
end mouseDown
on GetFitness pool, individual
      -- Find out the fitness of a chromosome
      -- First update the building's usage  according to the
      --values of the genes on the chromosome
   put "1" into i
   repeat for each shape s in collection "z"
      GetGeneValue pool, individual, 1, i
      put the result into TheUsage
      set the attribute "usage" of shape id s to TheUsage
      put (i + 1) into i
   end repeat
-- Second, work out the area used for each usage type in the development
   put 0 into OfficeArea
   put 0 into RetailArea
   put 0 into HousingArea
   put 0 into LeisureArea
   repeat for each shape s in collection "z"
      put the attribute "usage" of shape id s into TheUsage
      put the attribute "area" of shape id s into TheArea
      if TheUsage is "1" then
         add TheArea to OfficeArea
      end if
      if TheUsage is "2" then
         add TheArea to RetailArea
      end if
      if TheUsage is "3" then
         add TheArea to HousingArea
      end if
      if TheUsage is "4" then
         add TheArea to LeisureArea
      end if
   end repeat -- Third, tell Excel the area for each usage type
   post "Excel", "Zaanstad", 4,  8, OfficeArea
   post "Excel", "Zaanstad", 4,  9, HousingArea
   post "Excel", "Zaanstad", 4, 10, RetailArea
   post "Excel", "Zaanstad", 4, 11, LeisureArea
            -- Finally, ask Excel how 'fit' this mix of areas is
   fetch "Excel", "Zaanstad", 43, 25
   put the result into Fitness
   return Fitness
end GetFitness -- Excel is a registered trademark of Microsoft Corp.
```


Genetic Algorithms lend themselves not only to optimisation problems, but also to encoding and evolving generative models and rule sets as an analytic tool, however, to date we have not experimented seriously with this.

4 The Isovist Camera

A simple analytic tool, the 'Isovist Camera', has been developed in Pangea specifically for use in analysis of visual fields in three dimensional space. The isovist was first proposed by Benedikt as the "set of all points visible from a given vantage point in space and with respect to an environment." (Benedikt, 1979). The notion of the isovist is partly based on concepts developed by Gibson (1979), whose research into perception psychology considers a person standing in a room containing different objects and asks how that person perceives the environment. Gibson suggests that the 'optic array' describing the sum total of all of the rays of light bouncing off the surrounding surfaces, entering the eye, and forming a 2-d image on the retina must be both what actually enables us to perceive the environment in the first place (for this is how the eye works), yet can also be used to represent what is either visible or occluded to an observer.

The isovist camera is based on a Pangea camera, a scriptable shape which brings a window with it showing what can be seen from its particular location and viewing direction. The camera's script rotates the camera about a vertical axis through 360 degrees, in small steps, and at each step it picks the point at the centre of its view and finds the location of the face of the object it can see. It uses this series of points to construct a new extruded polygon representing the 'isovist' from that camera location as a new shape in the world.

```
on mousedown
      --SET UP THE SAMPLING RESOLUTION OF THE ISOVIST BOUNDARY
      put the attribute "Resolution" of me into steps
      put (360/steps) into degs
      repeat with counter = 1 to steps
            rotateby 0,degs,0
            paint 4
      --PICKING THE POINT
            getwindowsize
            put the result into ws
            put (item 1 of ws)/2 into wx
            put (item 2 of ws)/2 into wy
            put wx & "," & wy into fred
            pickpoint fred
            put the result into pt
      --ADDING THE POINT TO THE LIST OF VERTICES OF THE ISOVIST
            if pt = "NO" then
                  donothing
            else
                  put (item 4 of pt) into testid
                  put (item 1 of pt) into testx
                  put (item 2 of pt) into testy
                  put (item 3 of pt) into testz
                  put testx & "," & testy & "," & testz into testVtx

                  if testid = previd then  --pickpoint on same face:
ignore
                        put "same poly"
                        put testVtx into prevVtx
                  else
                  if counter = 1 then  --first point
                        put "first point!"
                        put testid into previd
                        put testVtx into prevVtx
```

```
                        put testVtx into firstVtx
                        put testVtx into lastPt
                  else
                        put "new poly" --new face so record the location
                        put list & prevVtx & "," & testVtx  & "," into
list
                        put testid into previd
                        put testVtx into prevVtx
                        put testVtx into lastPt
                  end if
            end if
      end if
      end repeat
      --MAKING THE ISOVIST SHAPE
      make "extrudedpolygon",list,0.01
      put the result into s
end mousedown
```


The isovist camera can then be scripted to move along a predefined path and construct isovists at regular intervals to produce a 'Minkowski model' (Benedikt, 1979), or different parameters of the isovist can be calculated such as its area perimeter ratio. The latter turns out to be particularly interesting in the first analyses that have been carried out of using the new isovist camera to look at the distribution of interacting groups in a building interior. A model of a floor of Company X's building (see Penn, Desyllas & Vaughan, 1997, in these proceedings) including furnishings, storage and partitions at different heights (Figure 9).

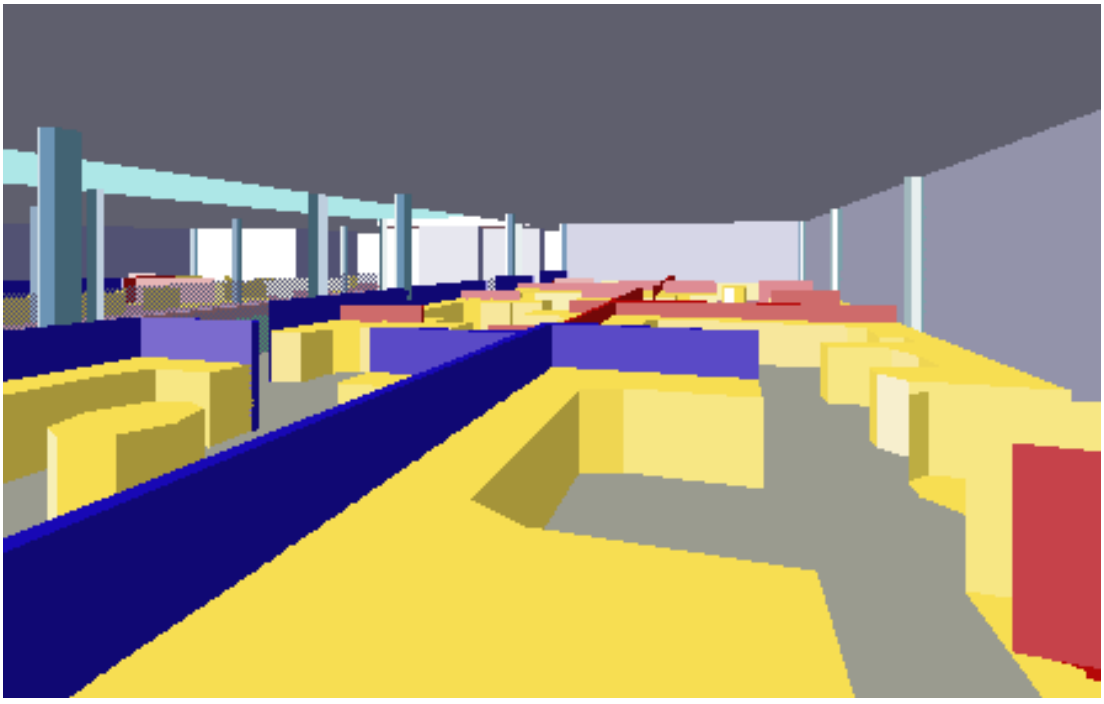


Figure 9. Standing height view of Company X's open plan office areas.

The isovist camera was then dragged into the model and located at different plan positions (Figure 10), as well as at seated and standing eye heights. Some positions are more strategic and obtain better views than others, and measures of the shape and size of the isovist can quantify these differences. In particular, the area-perimeter ratio measures the relative 'fatness' or 'spikiness' of an isovist. If we consider the effects of the built form in constructing or obstructing our awareness of people as we move through an environment, it seems possible that this type of measure may be informative.

30.18

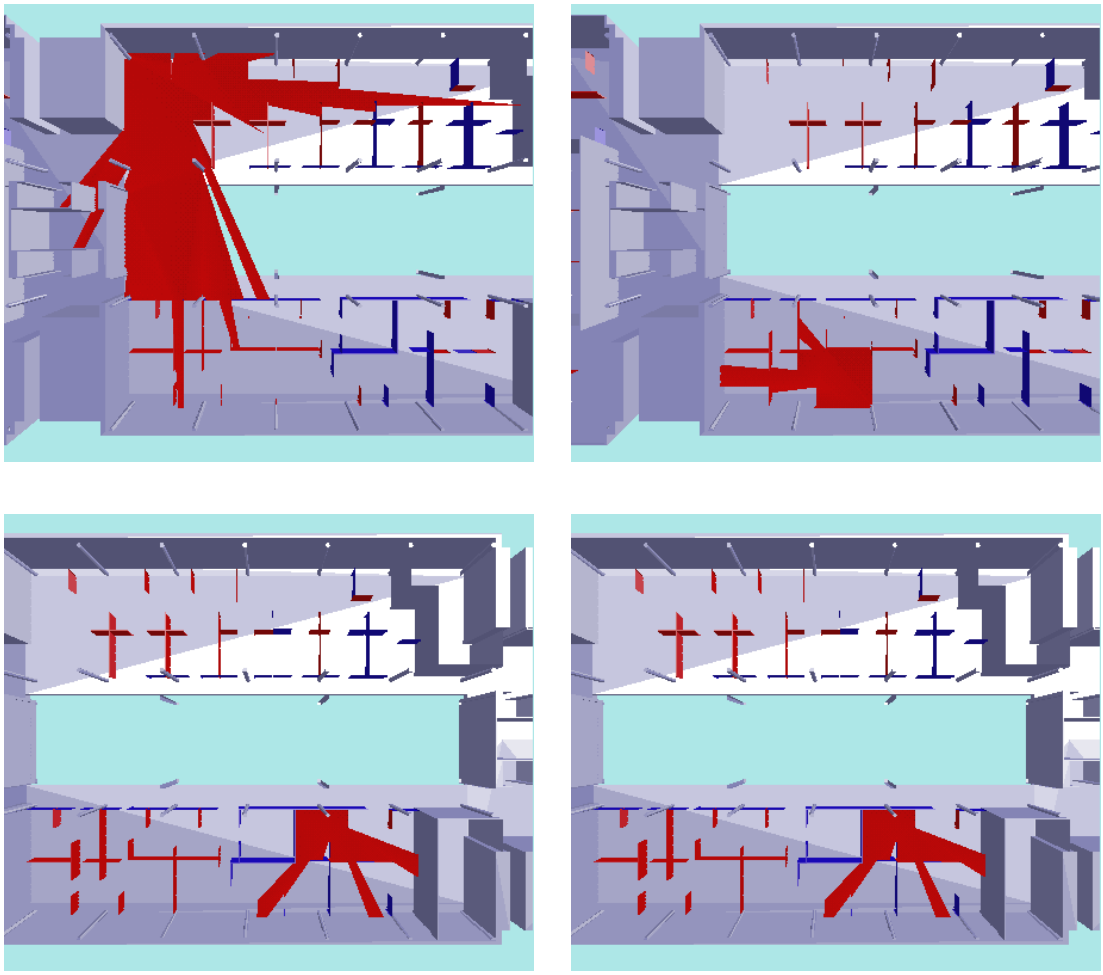


Figure 10. Seated eyeheight isovists at different desk locations in Company X.

In an early study Conroy counts of numbers of standing people at each of those locations were found to correlate well the area-perimeter ratio of the isovist, albeit with some anomalous results (Figure 11). It seems possible that the area-perimeter ratio may be measuring the ability of those passing down the main through-route adjacent to the atrium to be seen and 'recruited' into conversation by those at desk locations. If one passes through a spiky isovist one only gains glimpses of those one is passing, whilst if one passes through a 'fat' isovist one may be visible for long enough to allow recruitment to take place. If this is the case, then it suggests that we need to understand the way that the pattern of movement brings one into the isovist field of those seated at their workstation, and how this varies temporally as one moves about the system. This would require 'conventional' syntax representations that have been found to relate powerfully to patterns of movement to be brought into the same frame as 'static' isovist fields from workstations. Although these results are purely preliminary, they suggest that there may be considerable scope in extending the range of syntax methods from discrete to continuous representations and in unifying static and dynamic descriptions of configurational properties within a single analytic framework.

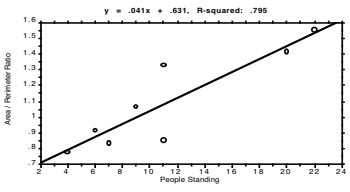


Figure 11. Scattergram of total observed people standing, plotted against the area/ perimeter ratio of all isovist locations, taken at seated eye height, (excluding isovist locations C & I). r=.89, p=.002.

One of the main objectives in developing Pangea was to begin to prototype these kinds of analyses. By constructing a full model of the spatial and formal configuration of an environment, and giving the researcher the means to develop new analytic tools that can move about the environment and analyse what is visible from any point within it, Pangea may make it possible to address issues more directly related to the egocentric and perceptual than have been possible to date.

Acknowledgments

Pangea was developed with funding from the Engineering and Physical Sciences Research Council of the UK and the Department of Trade and Industry under the Intelligent Systems Integration Programme by the Intelligent Architecture Project (EPSRC GR/J33609 IED4/8003). The project consortium consists of Avanti Architects, Broadgate Properties PLC, Bovis Construction Ltd., Criterion Software Ltd., DEGW London Ltd., Oscar Faber, PowerGen PLC, Richard Rogers Partnership, Qualum Ltd. and the Bartlett School of Architecture and the Department of Computer Science at University College London.

References

Battle, G. & McCarthy, C., (1996) *Multi-source synthesis: Dynamic Cities, in Architectural Design, Architecture on the Horizon*, Academy Press, London, July-Aug, 1996, III-IX;

Benedikt, M.L. (1979) To take hold of space: isovists and isovist fields, *Environment and Planning B, Planning and Design*, Pion, London, 1979;

Dalton, N. (1989), *SpaceBox software package developed at the UAS*, UCL, London, 1989;

Gibson, J. (1979), *The ecological approach to visual perception*, Houghton, Boston, USA, 1979;

Hillier B., & Penn A., (1993) Virtuous Circles, Building Sciences and the Science of Buildings: using computers to integrate product and process in the built environment in *Informing Technologies for Construction, Civil Engineering and Transport*, in Eds., Powell J. A., & Day, R., Brunel with SERC, London, 1993, pp. 283-300. republished in The International Journal of Construction Information Technology, 1:4, 69-92, Salford, 1993;

Kingdon, J. & Dekker, L., (1995) *The Shape of Space, Technical Report*, RN95/23, Department of Computer Science, University College London, 1995;

Penn, A. (1987), *Syntactica, software package developed at the UAS*, UCL, London, 1987;

Penn, A., Conroy, R., Dalton, N., Dekker, L., Mottram, C., Turner, A., (1995) Intelligent Architecture: User interface design to elicit knowledge models, in *Applications and Innovations in Expert Systems III*, Macintosh, A. & Cooper, C. (eds), SGES Pubs, Oxford, 1995, 335-348; ISBN 1 899621 03 2;

Penn, A., Treleaven, P., Hillier, B., Conroy, R., Dalton, N., Dekker, L., Mottram, C., Turner, A., (1996) *PANGEA: AN INTELLIGENT WORKBENCH FOR ARCHITECTURAL SKETCH DESIGN* Watson, I. & Morgan, A. (eds), The Intelligent Systems Integration Prgogramme ES96 Papers, SGES Pubs, Oxford, 1996, 127-139; ISBN 1 899621 14 8

Penn, A., Treleaven, P., Hillier, Bull, L., B., Conroy, R., Dalton, N., Dekker, L., Mottram, C., Turner, A., (1996a) *PANGEA v2.1b CD-ROM and Pangea User Manual*, UCL, London, ISBN 0902137 38 7;

Penn, A., Desyllas, J. & Vaughan, L., (1997) The space of innovation: Interaction and communication in the work environment, *Proceedings of the First International Space Syntax Symposium*, UCL, London, 1997;