DOKTORI ÉRTEKEZÉS

PHD DISSERTATION

MAJOR ANDREA

2006
Information Visualisation

PhD Dissertation
Andrea Major

Supervisor
Dr. Tibor Remzső

2006
Budapest
### Table of Context

1. The theory of visualisation ........................................................................................................5
   1.1. Introduction ..................................................................................................................5
   1.2. What is Visualisation? .................................................................................................6
   1.3. The visualisation Process .........................................................................................7
   1.4. Visualisation Mapping ..............................................................................................8
      1.4.1. AVO appearance ...............................................................................................8
      1.4.2. Selecting a good mapping .................................................................................10
   1.5. Data visualisation techniques ................................................................................10
   1.7. Historical Milestones of Visualisation ....................................................................14
      1.7.1. Charles Minard, Napoleon's failed Russian campaign of 1812 ......................14
      1.7.2. Playfair’s Chart ..............................................................................................14
      1.7.3. Florence Nightingale’s Coxcomb ..................................................................15
2. Multidimensional Data Visualisation ..............................................................................16
   2.1. Data Description .......................................................................................................16
   2.2. Visualisation Techniques .......................................................................................16
      2.2.1. Parallel Coordinates ......................................................................................16
         2.2.1.1. Brushing ..................................................................................................17
         2.2.1.2. Sample Information ...............................................................................18
         2.2.1.3. Bubble method ......................................................................................19
      2.2.2. Statistical Data Visualization - Sieve Diagrams ............................................21
      2.2.3. Pixel Bar Charts ............................................................................................22
3. Hierarchical Data Visualisation ....................................................................................24
   3.1. Data Description ....................................................................................................24
   3.2. Visualisation Techniques ......................................................................................24
      3.2.1. Treemaps .......................................................................................................24
      3.2.2. Cone Trees and Cam Trees ..........................................................................25
      3.2.3. The Information Cube ..................................................................................27
4. Network Data Visualisation ..........................................................................................28
   4.1. Data Description ...................................................................................................28
   4.2. Visualisation Techniques ......................................................................................28
1. The theory of visualisation

1.1. Introduction

Over the past few years, the role of information visualisation in supporting the analysis of data has increased dramatically. From its original role as simply a tool for presenting results, it has developed into a valuable tool for understanding data and to help grasp structure. One of the results of this has been the growth in the number of visualisation techniques.

This thesis aims to give an introduction to the information visualisation, covering the visualisation process, introducing visualisation taxonomy and presenting some of the well-known techniques.

The thesis is organised around the visualisation techniques based on data structures: multivariate, network, tree-hierarchical data structures. In addition Workplace visualisation techniques are detailed.

Individual research is built into the thesis, by introducing the enhancements to the existing techniques. Bubble method is an extension of the Parallel Coordinates and the research of different functions, enhancement around the Fisheye View technique is detailed as part of the workplace visualisation methods. In addition a new model, called Information Visualisation Classification Model has been introduced and the presented techniques mapped along the presented model.
1.2. What is Visualisation?

The visualisation process links the two most powerful information processing systems known - the human mind and the modern computer. Visualisation is a process that transforms data, information and knowledge into a visual form by exploiting people’s strength in visual pattern recognition, the ability to scan, recognise and detect changes in visual images.

Effective visual interfaces enables to observe, manipulate, search, navigate, explore, filter, discover, understand and interact with large volumes of data far more rapidly and far more effectively to discover hidden patterns. In our increasingly information-rich society visualisation research and development has fundamentally changed the way large complex data sets are presented and understand. The impact of visualisation has been widespread and fundamental, leading to new insights and more efficient decision making.

Much previous research arose from scientific community efforts to cope with the huge volumes of scientific data collected by scientific instruments or generated by massive super-computer simulations.

A new research and development focus has emerged within the visualisation community to address some of the fundamental problems associated with the new classes of data and their related analysis task. This research and development focus, called information visualisation, combines aspects of scientific visualisation, human computer interfaces, data mining, imaging and graphics. In contrast to most scientific visualisation, which focuses on data, information visualisation focuses on information, which is often abstract. Information in many cases does not automatically map to the physical world. This fundamental difference means that many interesting classes of information have no natural and obvious physical representation.

For comparative reasons, please consider the following table:
A key research problem within the Information Visualisation is than to discover new visual metaphors (AVOs see The Visualisation Process) for representing information and to understand what analysis tasks they support.

The information visualisation must enable users to get information they need, make sense of it, and reach decisions in a relatively short time. It needs to be also easy to be used.

1.3. The visualisation Process

Several models of visualisation have been proposed in the literature [11]. These vary in their scope, aims and intended audience. The thesis will be structured around the model of Haber and McNabb, which provides a useful general classification.

There are three broad processes, each of which acts on some data to produce new set of data.

The first process, data preparation, has as an input the raw data, and creates a model of the data from which a new derived data set can be produced. This covers such processes as calibration, smoothing, interpolation and the calculation of derived quantities.

The second process, visualisation mapping, creates an Abstract Visualisation Object (AVO) Each quantity in the derived data is mapped to an attribute.

The third process, presentation, is where one or more views of the AVO are rendered to produce a picture on an output device, generally the screen of the workstation, ex. view transformation, lighting and shading.
1.4. Visualisation Mapping

Visualisation mapping is the second stage of the visualisation process. The variables in the derived data set are mapped to attributes of an AVO such as spatial dimensions, time colour, vibration rate or transparency. In other words visualisation mapping is a process that maps data from the data domain to the visualisation space.

1.4.1. AVO appearance

Often the AVO is similar to a real object, but this does not need to be the case. It has the advantage that the viewers can apply their knowledge of the real world to understand the visualisation.

For example, if we consider the temperature measurements sampled over the globe, an obvious AVO is a geometric object corresponding to the map of the world. The positional data for each sampling point would map straightforwardly to positions on the AVO.
Representing temperature as colour would then give an overall picture of the actual temperature over the globe. See *Figure 1.2.*

![Europe Temperature](image)

*Figure 1.2.* Europe Temperature
Alta Vista Weather Forecast, 1998

In other cases the AVO looks nothing like a real object. It is harder to interpret, but this sort of AVO can display more information than the previous one.

For example, if the surface of the globe described above were modelled as a parametric surface, sampling position could be related to the 2D parameter space. This would free a third dimension, height, to display another variable such as pressure; temperature could still be shown as colour. The AVO would display more information than the previous one, but would look nothing like a map and so much harder to interpret. Generally, the AVO falls between the two extremes.
Frequently there are multiple AVOs, such as *Glyps*. Glyps, are small AVOs, that encode the value of several variables and can effectively used in multivariate data visualisation. Streamlines, arrow plots are examples of glyps. See *Figure 1.4. A de Leeuw and van Wijk glyp*

![de Leeuw and van Wijk glyps](image)

*Figure 1.3. A de Leeuw and van Wijk glyp*

### 1.4.2. Selecting a good mapping

There are many ways to map the derived data onto an AVO. The aim is to produce an AVO which is readily understood and clearly shows the things we are interested in looking at, while avoiding distracting details which do not contribute towards our understanding. Thus, when considering a suitable mapping it is important to understand what tasks the mapping can be used to achieve.

Using of color greatly increases the ability to efficiently communicate the meaning in data. Correctly used, color can be used to visualise one of the dimensions, to communicate in one image additional information or emphasise something. But useful though color may be, it is also a very difficult element to use effectively. Without some knowledge of the physiological and psychological relationship between the color and our perception of it, one can create confusion by using color as to contribute enlightenment.

### 1.5. Data visualisation techniques

There are several categorizations of the visualization techniques. The most well-known and practical approach has been introduced by *Ben Shneiderman* in 1996 [32].
“Overview first, zoom and filter, then details-on-demand” – this is the approach Shneiderman is using. Shneiderman’s approach has two dimensions, one is looking at the exploration of the data volume, the other at the useful display of the information. It is being called the type by task taxonomy (TTT) of information visualizations.

The seven tasks - at a high level of abstraction - are:

Overview: Gain an overview of the entire collection.
Zoom: Zoom in on items of interest
Filter: filter out uninteresting items.
Details-on-demand: Select an item or group and get details when needed.
Relate: View relationships among items.
History: Keep a history of actions to support undo, replay, and progressive refinement.
Extract: Allow extraction of sub-collections and of the query parameters.

The seven data types are as follows:

1-dimensional: linear data types include textual documents, program source code, and alphabetical lists of names which are all organized in a sequential manner.
2-dimensional: planar or map data include geographic maps, floor plans, or newspaper layouts.
3-dimensional: real-world objects
Temporal: time lines are widely used and vital enough for medical records, project management, or historical presentations to create a data type that is separate from 1-dimensional data. The distinction in temporal data is that items have a start and finish time and that items may overlap.
Multi-dimensional: data are mostly relational and statistical databases, having n attributes, which become points in a n-dimensional space.
Tree: hierarchies or tree structures are collections of items with each item having a link to one parent item (except the root).
Network: sometimes relationships between the items cannot be described with a tree structure and it is useful to have relations with other items.
Peter Young, 1996 [36], introduces a much simpler classification, not considering the data type, but putting emphasis on the interaction/mapping: mapping, presentation and dynamic information visualisation technique.

Mappings from the data domain to the visualisation space use some aspect, property or value of the data items to produce a mapping. Ex. Surface plots, cityscapes, Benediktine space.

Perspective walls, cone trees, cam trees and rooms may be classed as information presentation techniques. These visualisations concentrate on the appearance, accessibility and usability of the data and aim to provide a user friendly and intuitive interface.

Fish-eye views, self organising graphs, hierarchical browser may all be described as dynamic information visualisation techniques. These techniques allow the visualisations to respond automatically to changes in the data or to external requests.

An extension of the above mentioned taxonomies is being introduced below, based on which the most common visualisation techniques will be presented in the following chapters. These classifications are only rough and there is some degree of overlap, in addition, with the emergence of new techniques, additional dimensions, or other values along the axes can be added and as a consequence the model extended.

Let’s consider a model, called the Information Visualisation Classification Model (referred as IVCM), which has the following dimensions: Data Model, Graphical Techniques, Data Value and Workplace Techniques.

The Data Model dimension is mainly based on Shneiderman’s taxonamy:

- One dimensional or linear information, including text, algorithm
- Spacial – 2D – 3D
- Multi-dimensional
- Network
- Hierarchies

Along the dimension of Graphical techniques used, the followings can be considered:

- Standard 2D/3D: these techniques use standard 2D or 3D visualisation techniques.
Geometrical transformations: transformation, projections are being used to produce useful displays.

Icon-based displays: assign icon to the data to be visualised

Pixel displays: visualise each dimension value as a colored pixel and group these pixels into an adjacent area.

In addition to the visualisation technique, for an effective visualisation addition, workplace technique is necessary to be applied, which allow dynamic changes

Interaction technique: allow dynamic changes to the visualisation.

Distortion techniques help in the data explorations by providing means for focusing while preserving an overview of the data.

Data values dimension can take the following “values”: Data Value as well as Modified or Additional Value. Visualisation techniques may generate additional data, relation, find patterns and visualise these in order to help spotting trends, relations, aggregation, sorting etc,

The next chapters present some of the most well-known technique. These are mainly organised around the data model they are usually applied on.

Below the presented visualisation techniques are categorised along the IVCM model:

<table>
<thead>
<tr>
<th>Technique</th>
<th>Data Type</th>
<th>Graphical technique</th>
<th>Workplace</th>
<th>Data Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perspective Wall</td>
<td>Linear</td>
<td>Geometric transformation</td>
<td>Distortion</td>
<td>Real Values</td>
</tr>
<tr>
<td>Parallel Coordinates</td>
<td>Multidimensional</td>
<td>Geometric transformation</td>
<td></td>
<td>Real Values</td>
</tr>
<tr>
<td>Bubble</td>
<td>Multidimensional</td>
<td>Geometric transformation</td>
<td></td>
<td>Additional</td>
</tr>
<tr>
<td>Brushing</td>
<td>Multidimensional</td>
<td>Geometric transformation</td>
<td>Filter/restrict area of interest</td>
<td>Real Values</td>
</tr>
<tr>
<td>Sieve Diagram</td>
<td>Multidimensional</td>
<td>Geometric transformation</td>
<td></td>
<td>Additional</td>
</tr>
<tr>
<td>Pixel Bar Charts</td>
<td>Multidimensional</td>
<td>Pixel Displays</td>
<td></td>
<td>Additional</td>
</tr>
<tr>
<td>Treemap</td>
<td>Hierarchical</td>
<td>Geometric transformation</td>
<td></td>
<td>Real Values</td>
</tr>
<tr>
<td>Cone Tree/Cam Tree</td>
<td>Hierarchical</td>
<td>Geometric transformation</td>
<td>Transparency</td>
<td>Real Values</td>
</tr>
<tr>
<td>Information Cube</td>
<td>Hierarchical</td>
<td>Geometric transformation</td>
<td></td>
<td>Real Values</td>
</tr>
<tr>
<td>Hyperbolic Browser</td>
<td>Hierarchical</td>
<td>Fisheye</td>
<td></td>
<td>Real Values</td>
</tr>
</tbody>
</table>

Table 1.2. Visualisation techniques along the IVCM Model
1.7. Historical Milestones of Visualisation

1.7.1. Charles Minard, Napoleon's failed Russian campaign of 1812

Charles Minard [6,7]– French engineer, illustrated the disastrous result of Napoleon's failed Russian campaign of 1812. The size of the army is shown by the width of the band – both outward and return legs, with temperature on the retreat shown on the line graph at the bottom.

![Napoleon's Russian Campaign of 1812](image)

**Figure 1.4.** Minard: Napoleon’s Russian Campaign

Source: Gallery of Data Visualisation

1.7.2. Playfair’s Chart

William Playfair [6,7] is viewed as the inventor of most of the common graphical forms used to display data: the scatterplot, line plots, bar chart and pie chart. His *The Commercial and Political Atlas*, published in 1786, contained a number of interesting time-series charts. One of them is presented below.

![Playfair: Population and Taxes](image)

**Figure 1.5.** Playfair: Population and Taxes
1.7.3. Florence Nightingale’s Coxcomb

Florence Nightingale [6,7] is remembered as the mother of modern nursing. But few realize that her place in history is at least partly linked to her use, following Playfair, of graphical methods to convey complex statistical information dramatically to a broad audience.

In her „Notes on Matters Affecting the Health, Efficiency and Hospital Administration of the British Army” (1858), she included several graphs of her own design, which she called "Coxcombs".

Nightingale's Coxcomb is notable for its display of frequency by area, like the pie chart. Unlike the pie chart, the Coxcomb keeps angles constant and varies radius.

![Florence Nightingale: Coxcomb](image)

**Figure 1.6.** Florence Nightingale: Coxcomb
Source: Gallery of Data Visualisation
2. Multidimensional Data Visualisation

2.1. Data Description

Multi-dimensional information describes items with more than two attributes. These attributes form an n-dimensional space, within which each item is placed, based on the value of its attributes. Multi-dimensional visualisation is being used for example in stock market statistics, statistical information, and various database mining exercises.

One, two, three dimensional and temporal information visualisation schemes can be viewed as a subset of multi-dimensional information visualisation. Attributes in multi-dimensional visualisations should have no explicit structure or relations between them.

2.2. Visualisation Techniques

2.2.1. Parallel Coordinates

The system of *parallel coordinate* is a non-projective mapping between N-Dimensional and 2-Dimensional sets. The simplified methodology developed by Alfred Inselberg [13] is presented below.

In Parallel Coordinates each dimension of the multidimensional data corresponds to an axes, and the N axes are organised as uniformly spaced vertical lines. A data element in N-dimensional space manifests itself as a connected set of points, one on each axes. *See Figure 2.1.*
Above the axes are placed the variable identifiers. The possible variable values are uniformly mapped on the axes. A polyline across the N-axes represents the data-elements of one sample, and in our example it is being drawn in green.

Additional enhancement methods can be used on the parallel coordinate method: brushing, sample, bubble.

2.2.1.1. Brushing

Holding the information regarding the spatial relationship of the data N-D brushing method is used to decrease, increase or manipulate the range of each dimension, focussing in this way on specific variable-values.
It is used to highlight data points, which fall into a user-defined sub-space. Light blue shaded area represents the brushed area. See Figure 2.2.

2.2.1.2. Sample Information

It is very important to get information on-line, without looking for the meaning of one axes or the samples in a book. This is made possible by pointing on the objects within the geometry viewer.

The poly-lines carry the data sample elements and the axis the name of the variable, its minimum, maximum values and the information related to the brush. See Figure 2.3.
The above method is suitable for spotting correlation in data. It is not the best viewing technique for finding out trend in the dataset.

Based on the parallel coordinates, an enhancement of the technique has been developed, in order to provide additional information to the already visualised data. It is not a simple extension of the dimension, but it is about adding additional information to the already visualised data set. This helps to spot trends, feature, which would require detailed data research.

2.2.1.3. Bubble method

It is based on the main idea of the Parallel Coordinate method. Instead of viewing individual data, the distribution of the whole data-set is mapped along the axes. The method below can be viewed as one example of the extension of the Bubble method. Based on specific needs, variants of the method can be applied.

As in case of the Parallel Coordinates, the axes represent the dimensions within the multidimensional dataset. For each value a “bubble” is being placed along the axis, holding multiple information. The radius of the sphere gives the relative frequency number of the variable value on the specific axes. From each sphere there are two lines pointing out, holding the most demanded next variable choice (red line) and the least preferred (purple colour) one for the next variable from the actual variable-value set. See Figure 2.4.
The figures presented above represent the outcome of a project, which involved an investigation of the suitability of a number of techniques for the visualisation of multi-dimensional multi-variate data.

The visualisation methods were written using the AVS Application.
2.2.2. Statistical Data Visualization - Sieve Diagrams

Statistics is one of the areas where multidimensional data visualisation is of high demand and is being applied with success. In addition, these techniques do tend to add additional values to the visualisation data, which are in majority a statistical function (frequency, mean etc.) of the data being visualised.

The table below shows data on the relation between hair color and eye color among 592 subjects collected by Snee (1974). The question is how to understand the nature of the association between hair and eye color.

<table>
<thead>
<tr>
<th>Eye/Hair Color</th>
<th>Black</th>
<th>Brown</th>
<th>Red</th>
<th>Blond</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brown</td>
<td>68</td>
<td>119</td>
<td>26</td>
<td>7</td>
<td>220</td>
</tr>
<tr>
<td>Blue</td>
<td>20</td>
<td>84</td>
<td>17</td>
<td>94</td>
<td>215</td>
</tr>
<tr>
<td>Hazel</td>
<td>15</td>
<td>54</td>
<td>14</td>
<td>10</td>
<td>93</td>
</tr>
<tr>
<td>Green</td>
<td>5</td>
<td>29</td>
<td>14</td>
<td>16</td>
<td>64</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>108</strong></td>
<td><strong>286</strong></td>
<td><strong>71</strong></td>
<td><strong>127</strong></td>
<td><strong>592</strong></td>
</tr>
</tbody>
</table>

Table 2.1. Hair-color eye-color data

The expected frequencies under independence can be represented by rectangles whose widths are proportional to the total frequency in each column, and whose heights are proportional to the total frequency in each row. Figure 2.5. shows the expected frequencies for the hair and eye color data.

![Figure 2.5. Expected frequencies under independence.](image1)
Reiedwyl and Schüpbach [26] proposed a sieve diagram (later called a parquet diagram) based on this principle. In this display the area of each rectangle is proportional to expected frequency and observed frequency is shown by the number of squares in each rectangle. Hence, the difference between observed and expected frequency appears as the density of shading, using color to indicate whether the deviation from independence is positive or negative. The sieve diagram for hair color and eye color is shown in Figure 2.6.

![Sieve diagram: Hair Eye Color Data](image)

**Figure 2.6.** Sieve diagram for hair-eye color

In the sieve diagram the foreground (rectangles) shows expected frequencies; deviations from independence are shown by color and density of shading.

### 2.2.3. Pixel Bar Charts

Pixel bar charts [16] use the basic idea of bar charts but at the same time use the available screen space to present more information. Pixels within the bars are coloured according to the values of the data. To make the display more meaningful, two parameters of the data records are used to order the pixels within the bar in x- and y- dimension.
Figure 2.7. Pixel Bar Charts, Konstanz University Research
3. Hierarchical Data Visualisation

3.1. Data Description

Hierarchies, - tree structures - , are collections of data nodes where each node has a unique parent (above it in the hierarchy), but may have many siblings (below it in the hierarchy). In general, the nodes and the links between them can have multiple attributes.

Hierarchies may arise in taxonomies, organisation’s structure or disk space management. Hierarchies can be used in finding a particular node, viewing a node in the context of the entire hierarchy, examining the overall structure and relations of the tree, and even finding duplicates or anomalies within the tree structure.

The traditional presentation of hierarchies is a 2-D representation where child nodes are positioned under their parents in wedge-like formations. This technique is applicable for a small hierarchy, which can be drawn on a computer screen. For large hierarchies this would not work due to disorientation and lack of visibility of the tree.

Hierarchical visualisation techniques attempt to show many more nodes -if not the entire tree itself - as well as providing mechanisms for navigation and searching which allows the user to keep the context of the entire tree in mind as well as reducing the disorientation. These techniques may use 3D graphics in order to visualise more information in the same space.

3.2. Visualisation Techniques

3.2.1. Treemaps

The Treemap visualisation technique is a 2D space filling technique, which makes 100% use of the available space by mapping the full hierarchy onto a rectangular region. The method is able to depict both the structural and content information as well.

The Treemap concept was invented by Ben Shneiderman [31] in response to the common problem of a filled hard disk. See Figure 3.1.
Each node is represented by a rectangle, which size is in line with the weight associated to the node. In case of the above example each file appears as a rectangle which size is proportional to the file size, enabling users to spot large files at any level in the hierarchy.

Figure 3.1. Tree-Map of a Directory Structure

Applications of the Treemap visualisation technique are: TreeViz™ for Macintosh models and the WinSurfer™ implementation, by Marko Teittinen, available for the Windows environment.

The application uses color to show file type – attribute of the node, e.g. text, picture, application, etc. By pointing and clicking a rectangle, users can bring up detailed information about nodes such as filename, path, creation date, etc.

3.2.2. Cone Trees and Cam Trees

Cone trees are 3D extension of the familiar 2D hierarchical tree structures. Cam trees are similar to cone trees except that they grow horizontally. The aim of cone and cam trees is to allow a greater amount of information to be displayed using the ability of 3D perception.
Cone trees are constructed by placing the root node at the top of the display area. All child nodes are then distributed at the next level. The height and diameter of the cones are calculated as a function of the display size and the tree depth. Transparency shading is used for the body of the cones in order that the cones in the back are visible to the user.

The original cone and cam tree visualisations produced at Xerox PARC enabled the tree to be rotated smoothly to bring any particular node into focus. The smooth animation was found to be critical in maintaining the viewer’s cognitive model of the structure. Sudden changes in orientation of the tree would require a significant amount of time to re-orientate the user’s cognitive model.

Consider Figure 3.2. and Figure 3.3. as visualisation examples of a cam and cone tree, representing directory structure.

![Figure 3.2. Cam Tree, Source Inxight](image)

![Figure 3.3. Cone Tree, Source Dave Snowdon, Nottingham University](image)
3.2.3. The Information Cube

Using nested translucent cubes Rekimoto and Green [27] invented the Information cube to visualise hierarchical information.

The information cube technique effectively extends 2D treemaps into 3D. Hierarchical information is presented as nested translucent cubes, where the level of transparency varies with the ‘depth' and the amount of information presented. Transparency and shading is the main technique used to control the information content of the cube visualisations. This transparency allows the user to view the contents of the cubes and their children, while hiding inner information gradually.

Figure 3.4. An information cube visualisation
By Jun Rekimoto, Sony Computer Science Laboratory Inc.
4. Network Data Visualisation

4.1. Data Description

The final goal of network information visualization is to get insight into a structure. Networks consist of nodes – data points - and links - representing a relationship between two nodes.

At the beginning, due to the simple structure the visualisation of the networks were simple graph drawings. These simple graphs were easy to draw and understand, however large data sets may require more sophisticated visualisation tools. Large set of data use to have much of their information hidden. Finding a structure or hierarchy among a set of data points is not straightforward. Based on the structure of the, they can be categorised into different networks (acyclic, lattices, rooted versus un-rooted, directed versus undirected) and visualisation tools are developed accordingly.

Application areas of network visualisation include telecommunication and computer networks, Web pages and links between them, geographic relations between locations – map of a tube system.

4.2. Visualisation Techniques

As mentioned before, visualisation techniques make use of the structure of the network. One technique is using the physical analogy of the graph in order to lay the nodes. Examples are the telecommunication networks. Other techniques reduce the networks to a tree structure and visualise them using hierarch visualisation techniques.

The structure of world trade between the countries is visualised on the figure below. The size of the nodes gives the volume of flows in dollars (imports and exports) for each country. The size of the links stands for the volume of trade between any two countries. Colors give the regional respectively memberships in different trade organisations.
Figure 4.1. World Trade in 1981
5. Workspace Visualisation

5.1. The Fisheye View

5.1.1. Introduction

With the increase of information visualisation the size of display remains constant. If one would like to visualise extra information, than it happens at the expense of the other information.

Displaying detailed pictures has the advantage of showing the global, entire picture but has the drawback that details are too small to be seen. Alternatively, zooming into a part of the picture and panning to other parts does show local details but loses the overall structure of the picture.

Using two or more views – one of the entirely picture and the other of a zoomed portion-gives one the advantage of seeing both local detail and overall structure but has the drawbacks of requiring extra screen space and forcing the viewer to mentally integrate the views. The multiple view approach has also the disadvantage that parts of the picture adjacent to the enlarged area are not visible at all in the enlarged view.

A fisheye-view of a picture shows an area of interest quite large and with detail and shows other areas successively smaller and in less detail. It achieves this smooth integration of local detail and global context by repositioning and resizing elements of the picture.

A fisheye-view seems to have all the advantages of the other approaches for viewing and browsing a picture without any drawback.

William Farrand was the first who introduced the technique of the fish-eye view in 1973. He has also applied the concept of the fisheye-view on the computer screen.

In 1982 Spence and Apperley developed a new technique the bifocal-display. It consists of information grouped horizontal streaks. One can differentiate one normal view streak and two detailed streaks on both sides of the first one.

5.1.2. A formal framework

Generating a fisheye view involves magnifying the area of greater interest and correspondingly de-magnifying the areas of lower interest. Intuitively, the position of the point in the fisheye view depends on its position in normal view and its distance from the focus. Furthermore, by applying a relative importance within the global structure to each element of the original layout, this relative importance is preserved after the transformation process.

The process consists of two steps. First, a geometric transformation is applied on normal view in order to reposition the elements of the view and magnify or de-magnify areas. Second, the size of the elements is obtained, using their relative importance to the focus.

5.1.2.1. Computing position

5.1.2.1.1. Introduction

A function, which transfers each element of the original layout to the fisheye-view, should have the following features:

- the focus and the points along the sides of the bounding box do not change their positions
- the points closer to the focus will be distorted with a greater amount than the others
- the function is continuous, monotone increasing and the degree of the increase of the function is decreasing.

In order to compute the function the application is simplified. The view consists of elements in a grid of a shape of a circle. In the first step the centre of the elements will be transformed.

The following notations and notions are used:
normal-view : the original layout
fisheye-view : the layout after applying the function
normal co-ordinate : the co-ordinate of the element in normal-view
fisheye co-ordinate : the co-ordinate of the element in fisheye-view
focus co-ordinate : the co-ordinate of the focus in both normal and fisheye-view

\[ P, Q \] : the element in normal view
\[ P_x, P_y \] : the P element coordinate in normal view
\[ P', Q' \] : the element in fisheye-view
\[ P'_x, P'_y \] : the P element coordinate in fisheye-view
\[ F \] : the focus
\[ F_x, F_y \] : the focus coordinates
\[ f, g, g_1, g_2, g_3 \] : applied functions
\[ d \] : distortion factor
\[ D_{\text{norm}}, x \] : the distance between the focus and the element along the OX axes
\[ D_{\text{max}}, x \] : the distance of the focus from the boundary of the picture along the OX axes
\[ r \] : the distance of the element from the focus in normal-view
\[ r' \] : the distance of the element from the focus in fisheye-view
\[ \theta, \theta' \] : the angle between the element’s normal vector and the Ox axe in normal and fisheye-view
\[ r_{\text{max}} \] : the distance from the focus along the \( \theta \) to the picture’s boundary
\[ \text{min}_x, \text{min}_y, \text{max}_x, \text{max}_y \] : the co-ordinates of the picture along the X,Y axes

5.1.2.1.2. The transformation function

Let’s note \( f \), the applied transformation function. The position of the elements in the fisheye-view depends on their original, normal co-ordinates and their distances from the focus.

\[ P' = f(P, F) \]

The requirements of the function are the followings:
\[ f(F) = F \]
\[ f(P_{\text{max}_x}) = P_{\text{max}_x} \]
\[ f(P_{\text{max}_y}) = P_{\text{max}_y} \]
\[ f(P_{\text{min}_x}) = P_{\text{min}_x} \]
\[ f(P_{\text{min}_y}) = P_{\text{min}_y} \]

- if \( |P - F| \leq |Q - F| \) then \( |P' - P| \geq |Q' - Q| \)

### 5.1.2.1.3. Cartesian transformation

Let’s introduce the \( g \) function. The \( g \) function is characterised by the followings:

- \( g : [0,1] \rightarrow [0,1] \)
- \( g(0) = 0 \) and \( g(1) = 1 \)
- monotone increasing and the degree of increase is decreasing

Let’s follow its behaviour on **Figure 5.1**.

\[
\begin{align*}
g_1(x) &= x \\
g_2(x) &= 1 + \frac{x - 1}{dx + 1} \\
g_3(x) &= \log_2(x + 1)
\end{align*}
\]

**Figure 5.1.** Function \( g \), Cartesian transformation

If \( g_1 \) function is applied there will be not distortion, ie. for each point \( P = P' \).

Function \( g_2 \) meets the previously listed requirements:
- the points along the boundary and the focus keep their positions after applying the function
• the distortion is more emphasised near the focus than at the other points.

\( d \), the distortion factor, gives the degree of distortion. If \( d = 0 \), then \( g_2 = g_1 \).

Function \( g_3 \) is based on the features of the logarithmic function; however, it has only slight effect on the transformation.

In order to preserve the shape of the bounding box in the fisheye-view the \( g \) function is applied on the normalised values. In other words it is applied on the proportion of the distance between the focus and the element and of the distance of the focus from the boundary of the picture.

\[
f(P) = g \left(\frac{\frac{D_{\text{norm}}.x}{D_{\text{max}}.x}}{x}\right) \cdot D_{\text{max}}.x + F
\]

**if** \( P.x \leq F.x \) **then**

\[ P'.x := [-g \frac{(F.x-P.x)}{(F.x - \text{min}_x)\cdot(D_{\text{max}}.x - \text{min}_x)}]+F.x \]

**else**

\[ P'.x := [g \frac{(P.x-F.x)}{(\text{max}_x - F.x)\cdot(D_{\text{max}}.x - \text{min}_x)}]+F.x \]

**if** \( P.y \leq F.y \) **then**

\[ P'.y := [-g \frac{(F.y-P.y)}{(F.y - \text{min}_y)\cdot(D_{\text{max}}.y - \text{min}_y)}]+F.y \]

**else**

\[ P'.y := [g \frac{(P.y-F.y)}{(\text{max}_y - F.y)\cdot(D_{\text{max}}.y - \text{min}_y)}]+F.y \]

In order to present the effect of the functions, let’s consider **Figure 5.2**. On Picture #1 function \( g_3 \) is applied and obviously the effect is nearly nothing. On Picture #2 function \( g_2 \) with \( d = 2.1 \) distortion degree is applied.
A Cartesian transformation has the property that all the vertical and horizontal lines remain vertical and horizontal after the transformation. Because of this property Cartesian transformation are especially well suited for layouts with edges consisting of mostly horizontal and vertical line segments, for example the computer networks.

5.1.2.1.4. Polar co-ordinate transformation

Cartesian transformation seems somehow unnatural, especially when applied to familiar objects, such as maps. The framework allows one to address this issue by using domain-specific transformations. A more natural fish-eye view of such a map might be achieved by distorting the map into a hemisphere. To do so, one may use a transformation based on the polar co-ordinate system with the origin in the focus.

Each point can be specified by its distance from the focus and the angle between its normal co-ordinate and the $Ox$ axis. The angle remains unchanged; consequently, the point at equal distance from the focus will be distorted by the same degree.

Let's consider a polar co-ordinate system with the origin in the focus and its axis parallel with the boundary of the picture.

Let's use the following notations:

- $r$ : distance of element $P$ from the origin in normal view
- $\theta$ : angle between element $P$’s normal vector and the $Ox$ axis
- $r_{\text{max}}$ : the distance of the focus from the boundary of the box along the $Ox$ axis
\( r' \) : the distance between the element \( P \) and the focus in the fisheye-view

\( \theta = \theta' \) : \( \theta \) remains unchanged after the transformation.

\[
\begin{align*}
    r' &= g\left( \frac{r}{r_{\text{max}}} \right) \times r_{\text{max}} \\
    \theta' &= \theta \\
    P'.x &= r' \times \cos(\theta') \\
    P'.y &= r' \times \sin(\theta')
\end{align*}
\]

On Figure 5.3 the effect of the polar co-ordinate transformation is presented. The focus is placed in the middle of the picture and \( d = 2.1 \) distortion factor has been applied.

![Figure 5.3. Polar co-ordinate transformation](image)

5.1.2.2. Computing the size of the elements

Each element has a position, specified by its normal co-ordinates and a size, which is the radial of the circle. Each element is also assigned a number to represent the relative importance in the global structure. This number is called the \emph{a priori importance} (API) of the element.

Each element in fisheye view is assigned a \emph{visual worth} (VW), computed based on its distance from the focus (in normal co-ordinates) and its API. The VW will determine the final size of the element in the fisheye view.
Let’s consider the following notations:

\( v, v' \): element in normal view and in the fisheye-view;

\( f \): focus;

\( s, s' \): size of the element in the normal view and in the fisheye-view;

\( vw, vw' \): visual worth of the element \( v \) in normal and in fisheye view;

Using the notations previously defined the following concepts are formalised:

- the position of the element \( v \) in the fisheye view is a function of its position in normal co-ordinates and the position of the focus:

\[
v' = f_1(v, f)
\]

- the size of the element in the fisheye view is a function of its size and its position in normal co-ordinates, the position of the focus:

\[
s' = f_2(s, v, f)
\]

- the visual worth of the element depends on the distance between the element and the focus in normal co-ordinates and on the element API's:

\[
vw' = f_3(v, f, API)
\]

On the following figure the size of the elements are computed based on their distance from the focus.
5.1.2.3. Other transformation strategies

5.1.2.3.1. Limited transformation area

The previous examples have in common the fact that they have been applied on the whole area of the picture. In the followings let’s consider the case of a limited transformation area around the focus.

Figure 5.5 is an example of polar co-ordinate transformation on a circle area and Figure 5.6 is an example of a cartesian transformation on a square area.
### 5.1.2.3.2. Logical Fisheye-Views

In the previous examples we have described distance as the Euclidean distance separating two vertices in a graph, whereas Furnas defined the distance as an arbitrary function between two objects in a structure.

Let's consider a graph and a *Degree of Interest* (DOI) function which assigns each node in the structure a number which tells you how interested the user is in seeing that point.

The generalised fisheye views arise by decomposing the DOI into two components: *a priori* importance and distance. The DOI function is as follows:

\[
\text{DOI}_{\text{fisheye}}(x \mid y) = \text{API}(x) - D(x,y),
\]

where

- \( \text{DOI}_{\text{fisheye}}(x \mid y) \): the users DOI in a point \( x \), given that the current point of focus is \( y \);
- \( \text{API}(x) \): the global *a priori* importance of \( x \);
- \( D(x,y) \): the distance between \( x \) and \( y \);

The nodes are displayed if the DOI is above a some threshold (choosing \( k \),

\[ \text{DOI}_{\text{fisheye}}(x \mid y) \geq k \].

A rooted tree structure is considered to present the logical fisheye-view. \( D(x,y) \) has a natural instantiation as \( d_{\text{tree}}(x,y) \), the path length distance between \( x \) and \( y \) in the tree. Similarly,
API(x) can become $-d_{\text{tree}(x, \text{root})}$, the distance of $x$ from the root, under the approximating assumptions that the points at levels closer to the root are intrinsically more important.

$$\text{DOI}_{\text{fisheye}}(\text{tree})(x | y) = -d_{\text{tree}(x, y)} + d_{\text{tree}(x, \text{root})}$$

Figure 5.7 illustrates these two components and how they add together point by point to form the fisheye DOI function for the tree.

$\text{d}_{\text{tree}(x, y)}$ - distance $x$ from $y$

$\text{API}(x) = -d_{\text{tree}(x, \text{root})}$
By choosing a threshold, $k$, and only displaying those points with $\text{DOI}(x) \geq k$, one can obtain fisheye views of different size. For example, letting $k = -3$ selects only the most interesting subset which, by the fisheye DOI, turns out to be the direct ancestral lineage between $y$ and the root of the tree (i.e. zero order fisheye view). This subset is interesting basically because points on that lineage increase in \textit{a priori} importance in exact compensation for their increase in distance. If the threshold is lowered ($k = -5$, i.e. first order fisheye view) the next most interesting subset the ancestral line and its siblings are included. See Figure 5.8.

$$\text{DOI}_\text{fisheye(tree)}(x \mid y) = -d_{\text{tree}}(x,y) + d_{\text{tree}}(x,\text{root})$$

**Figure 5.7.** Distance, API and DOI of a fisheye view for a rooted tree
Due to the fisheye view exceptional feature, that it provides detailed visualisation of the selected area on the expense of the surrounding area, but keeping the overall structure of the picture, it is widely applied in different visualisation applications.

Fisheye technique has the powerful advantage that it can be applied on different data structures: network, hierarchies or on data without any data structure.

**5.1.3. Applications of the fisheye-view technique**

This visualisation technique lays out a hierarchy on a hyperbolic plane and maps this plane onto a circular display region. The hyperbolic plane is a non-Euclidean geometry in which parallel lines diverge away from each other. This leads to the convenient property that the circumference of a circle on the hyperbolic plane grows exponentially with its radius, which means that exponentially more space is available with increasing distance. As a consequence
hierarchies, which tend to expand exponentially with depth, can be laid out in hyperbolic space in a uniform way, so that the distance (as measured in the hyperbolic geometry) between parents, children, and siblings is approximately the same everywhere in the hierarchy.

The hyperbolic browser [19] initially displays a tree with its root at the center, but the display can be smoothly transformed to bring other nodes into focus, as illustrated in Figure 5.9. In all cases, the amount of space available to a node decreases with its distance in the tree from the point in the center. As a consequence, the view will include several generations of hierarchies, making it easier for the user to explore the data/hierarchy without getting lost.

**Figure 5.9.** Hyperbolic tree, initial display

**Figure 5.10.** Hyperbolic tree, re-focusing
The concept of hyperbolic tree has several applications in real world hierarchical data visualisation. In the case of the World Wide Web hierarchy's visualisations, the particular nodes of the tree are the URLs. See Figure 5.11.

![Hyperbolic representation](image)

**Figure 5.11.** Hyperbolic representation

### 5.1.3.3. Perspective Wall

Perspective walls [21] are a technique for viewing and navigating large, linearly structured information, allowing the viewer to focus on a particular area while still maintaining some degree of location or context. Perspective walls are similar to the fisheye views in that they allow a particular area of information to be viewed in detail while information close to this is still visible in lesser detail thus giving an idea of position and orientation within the data. One can arrange along both X and Y axes by one criteria. By clicking on the objects the wall moves into focus to on the specific cluster of document or data of interest. See Figure 5.12.
5.1.3.4. Sphere visualisation

The sphere visualisation is a 3D version of the 2D perspective wall. The objects are mapped onto the surface of a sphere with highly related objects placed close to the object of interest. Unrelated objects are displayed further from the object of interest and thus become less visible as they move round to the opposite side of the sphere.

The information is presented on the surface of a number of nested spheres. This provides a mechanism for representing different levels of information. The object of interest is displayed on the outermost sphere with objects directly related to it surrounding around the surface of the sphere. Objects which are indirectly related to the object of interest are considered lower level objects and are displayed on spheres nested within the outer sphere. The colour of the spheres becomes darker with increasing depth of nesting to give the user visual cues to their current location within the visualisation. Navigation through the visualisation is facilitated by rotating the sphere to bring nodes of interest into view and by traversing links to lower level spheres.
Figure 5.13. Sphere visualisation produced using VizNet.
Kim Fairchild, Institute of Systems Science, National University of Singapore
List of Figures and Tables

<table>
<thead>
<tr>
<th>Table/Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 1.1</td>
<td>Scientific and Information Visualisation</td>
<td>7</td>
</tr>
<tr>
<td>Figure 1.1</td>
<td>The Visualisation Process, After Haber and McNabb</td>
<td>8</td>
</tr>
<tr>
<td>Figure 1.2</td>
<td>Europe Temperature</td>
<td>9</td>
</tr>
<tr>
<td>Figure 1.3</td>
<td>A de Leeuw and van Wijk Glyn</td>
<td>10</td>
</tr>
<tr>
<td>Table 1.2</td>
<td>Visualisation Techniques Along the IVCM Model</td>
<td>13</td>
</tr>
<tr>
<td>Figure 1.4</td>
<td>Minard: Napoleon's Russian Campaign</td>
<td>14</td>
</tr>
<tr>
<td>Figure 1.5</td>
<td>Playfair: Population and Taxes</td>
<td>14</td>
</tr>
<tr>
<td>Figure 1.6</td>
<td>Florence Nightingale: Coxcomb</td>
<td>15</td>
</tr>
<tr>
<td>Figure 2.1</td>
<td>Parallel Coordinates</td>
<td>17</td>
</tr>
<tr>
<td>Figure 2.2</td>
<td>Brushing the Area</td>
<td>18</td>
</tr>
<tr>
<td>Figure 2.3</td>
<td>Sample Information</td>
<td>18</td>
</tr>
<tr>
<td>Figure 2.4</td>
<td>Bubble Method</td>
<td>20</td>
</tr>
<tr>
<td>Table 2.1</td>
<td>Hair-Color Eye-Color Data</td>
<td>21</td>
</tr>
<tr>
<td>Figure 2.5</td>
<td>Expected Frequencies Under Independence.</td>
<td>21</td>
</tr>
<tr>
<td>Figure 2.6</td>
<td>Sieve Diagram for Hair-Eye Color</td>
<td>22</td>
</tr>
<tr>
<td>Figure 2.7</td>
<td>Pixel Bar Charts, Konstanz University Research</td>
<td>23</td>
</tr>
<tr>
<td>Figure 3.1</td>
<td>Tree-Map of a Directory Structure</td>
<td>25</td>
</tr>
<tr>
<td>Figure 3.2</td>
<td>Cam Tree</td>
<td>26</td>
</tr>
<tr>
<td>Figure 3.3</td>
<td>Cone Tree</td>
<td>26</td>
</tr>
<tr>
<td>Figure 3.4</td>
<td>An Information Cube Visualisation</td>
<td>27</td>
</tr>
<tr>
<td>Figure 4.1</td>
<td>World Trade in 1981</td>
<td>29</td>
</tr>
<tr>
<td>Figure 5.1</td>
<td>Function G, Cartesian Transformation</td>
<td>33</td>
</tr>
<tr>
<td>Figure 5.2</td>
<td>Cartesian Transformation</td>
<td>35</td>
</tr>
<tr>
<td>Figure 5.3</td>
<td>Polar Co-ordinate Transformation</td>
<td>36</td>
</tr>
<tr>
<td>Figure 5.4</td>
<td>Computing the Size</td>
<td>38</td>
</tr>
<tr>
<td>Figure 5.5</td>
<td>Polar Co-ordinate Transformation on a Circle Area</td>
<td>38</td>
</tr>
<tr>
<td>Figure 5.6</td>
<td>Cartesian Transformation on a Square Area</td>
<td>39</td>
</tr>
<tr>
<td>Figure 5.7</td>
<td>Distance, API and DOI of a Fisheye View for a Rooted Tree</td>
<td>41</td>
</tr>
<tr>
<td>Figure 5.8</td>
<td>Zero, First and Second Order Fisheye Views for a Rooted Tree</td>
<td>42</td>
</tr>
<tr>
<td>Figure 5.9</td>
<td>Hyperbolic Tree, Initial Display</td>
<td>43</td>
</tr>
<tr>
<td>Figure 5.10</td>
<td>Hyperbolic Tree, Re-Focusing</td>
<td>43</td>
</tr>
<tr>
<td>Figure 5.11</td>
<td>Hyperbolic Representation</td>
<td>44</td>
</tr>
<tr>
<td>Figure 5.12</td>
<td>Perspective Wall</td>
<td>45</td>
</tr>
<tr>
<td>Figure 5.13</td>
<td>Sphere Visualisation</td>
<td>46</td>
</tr>
</tbody>
</table>
Bibliography


[12] Clay Hellerg: Pitfalls of Data Analysis, University of Wisconses, School of Nursing and Medicine


[14] Dr. Mikael Jern: Market Trends in Graphics and visualisation in the late 90s


[32] Ben Schneiderman: They eyes have it: A task by Data Type Taxanomy for Information visualisation, 1996, University of Maryland

[33] Onel Daniel Troe: Spacial Distortion Techniques, 1999


[36] Peter Young: Three Dimensional Information Visualization, Visualisation Research Group, Centre for Software Maintenance. Department of Computer Science, University of Durham, 1996

[37] Dr. Yu, Chong Ho and Dr. John Behrens : The Alignment Framework for Data Visualization: Relationships Among Research Goals, Data Types, and Multivariate Visualization Techniques

Appendix A – Source Code for Bubble Method (AVS Application)

```
#ifndef COMMONSTDLIB_H
#define COMMONSTDLIB_H

#include "../Common/StdLib.h"
#include "BubbleObj.h"

int FirstRun;

void CPPCompute(upstream_geom * pick_info, struct SDimensionalData * Input, GEOMedit_list *Output, char * InfoBox);

int ButtonStatus;
int AxisInOperation;

extern "C"
{
  int ModuleDescriptionFunction()
  {
    int in_port, out_port, param, iresult;
    extern int ModuleComputeFunction();
    FirstRun = TRUE;
    AVSset_module_name("Bubble", MODULE_RENDER);
    // Port creation
    in_port = AVScreate_input_port("pick information", "struct upstream_geom", OPTIONAL | INVISIBLE);
    AVSset_input_class(in_port, "upstream_geom");
    // Input Port Specifications:-
    in_port = AVScreate_input_port("Input", "struct SDimensionalData", REQUIRED);
    // Output Port Specifications:-
    out_port = AVScreate_output_port("Output", "geom");
    param = AVSadd_parameter("Sample Information", "string_block", ",", ",", ",");
    AVSconnect_widget(param, "text_block_browser");
    // End of port creation bit.
    AVSload_user_data_types("/home/cgu/zzptmam/Uni/Common/DimUniData.h");
    AVSset_compute_proc(ModuleComputeFunction);
    // Initialise the brushing stuff:-
    ButtonStatus = BUTTON_UP;
    AxisInOperation = -1;
    return(1);
  }
  // The modules compute function:-
  int ModuleComputeFunction(upstream_geom * pick_info, struct SDimensionalData *Input, GEOMedit_list *Output, char * InfoBox)
  {
    char CommandBuffer [256];
    char * OutputBuffer, * ErrorBuffer;
    // Call the CPP compute function, which is at the end of the code:-
    *Output = GEOMinit_edit_list(*Output);
    if (FirstRun)
      
```
// Delete the top object:-
sprintf((char *) &CommandBuffer, "geom_delete_obj -object top");
// Send the command:
AVScommand("kernel", (char *) &CommandBuffer, & OutputBuffer,
& ErrorBuffer);
}
// Create a new one:-
CPPCompute(pick_info, Input, Output, InfoBox);

// Set the camera view if it's the initialisation run of the module:-
if (FirstRun)
{
  FirstRun = FALSE;
  // Make the background white:-
  sprintf((char *) &CommandBuffer, "geom_set_background 1 1 1");
  // Send the command:
  AVScommand("kernel", (char *) &CommandBuffer, & OutputBuffer,
  & ErrorBuffer);
  // Make the background white:-
  sprintf((char *) &CommandBuffer,
    "geom_set_camera_params -camera 1 -from 50 50 100 -up 0 1 0 -at 50 50 0
    -fov 45 -wsize 55");
  // Send the command:
  AVScommand("kernel", (char *) &CommandBuffer, & OutputBuffer,
  & ErrorBuffer);
}

sprintf((char *) &CommandBuffer, "geom_set_cur_obj -object top");
// Send the command:
AVScommand("kernel", (char *) &CommandBuffer, & OutputBuffer,
& ErrorBuffer);

return(1);
} // Extern "C" end.

// Initialization for modules contained in this file:-
static int ((*mod_list[ ]))() =
{ ModuleDescriptionFunction
};
#define NMODS (sizeof(mod_list) / sizeof(char *))

extern "C"
{
  AVSinit_modules()
  {
    AVSinit_from_module_list(mod_list, NMODS);
  }
  // Extern "C" end.
  // This is the CPP compute function:-

  void CPPCompute(upstream_geom * pick_info, struct SDimensionalData * Input,
  GEOMedit_list *Output, char * InfoBox)
  {
    // Construct, and pass the MD data pointer to the object:-
    CPCObj Object(Input);
  }
}
// Sort out the dimensions:-
Object.PlaceDimensions();

if (pick_info)
{
    int MouseInfo = Object.HandleMouseFlags(pick_info);
    Object.HandleAxis(pick_info, MouseInfo);
}

// Render the data:-
Object.DrawDimensions(Output);
Object.DrawBubbles(Output);
Object.DrawBoundingBox(Output);
Object.DrawLabels(Output);

// Set the selectable objects:-
Object.SetSelectableObjects(Output);

// This modules holds all the Bubble Method
// viewing extentions to the multidimentional data.
//
// Written by Andrea Major

#include "../Common/StdLib.h"
#include "BubbleObj.h"
#include "math.h"
#include <strings.h>

// The code:-
CPCObj::CPCObj(void)
{
}
CPCObj::~CPCObj(void)
{
    // The functional member variables:-
    CPoint CPCObj::GetCoord(int Dim, double Value)
    {
        CPoint temp(ScreenLocation [Dim], (Value - Min(Dim)) * ScreenHeight
            / (Max(Dim) - Min(Dim)));
        return temp;
    }

    // Routines to allocate the dimentions:-
    void CPCObj::PlaceDimensions(void)
    {
        int i;
        double inc;

        // Get the ammount of incrament for each axis:-
        if (NumDimensions() > 2)
        {
            inc = ScreenWidth / (NumDimensions() - 1);
        }
        // Calculate the position of the axis on
        // the screen:-
        for (i = 0; i < NumDimensions(); i++)
        {
            if (NumDimensions() == 1)
            {
                ScreenLocation [i] = ScreenWidth / 2;
            }
        }
    }
}
else {
    // There are 2, or more dimensions:-
    if (i == 0) // Is it the first dimension?
        ScreenLocation [i] = 0;
    else // Is it the last dimension?
        if (i == (NumDimensions() - 1))
            ScreenLocation [i] = ScreenWidth;
        else
            // This is a mid dimension:-
            ScreenLocation [i] = i * inc;
}
}

// Routines to draw the dimensional grid into
// The geom object:-

void CPCObj::DrawDimensions(GEOMedit_list *Output) {
    int i, j;
    GEOMobj *obj;
    float verts [2] [3], colors [2] [3];
    char Name [256];
    // Create our GEOM object to add polygons to:-
    obj = GEOMcreate_obj(GEOM_POLYTRI, NULL);
    sprintf((char *) & Name, "Dimention Axis");
    for (i = 0; i < NumDimensions(); i++) {
        verts [0] [0] = ScreenLocation [i];
        verts [0] [1] = 0;
        verts [0] [2] = 0;
        verts [1] [0] = ScreenLocation [i];
        verts [1] [1] = ScreenHeight;
        verts [1] [2] = 0;
        for (j = 0; j < 2; j++) {
            colors [j] [0] = 0.0;
            colors [j] [1] = 0.0;
            colors [j] [2] = 0.0;
        }
        GEOMadd_polyline(obj, (float *) &verts, (float *) &colors, 2, GEOM_COPY_DATA);
    }
    // Now we replace the geometry for the object named
    // "polygon" to this obj:-
    GEOMedit_geometry(*Output, Name, obj);
    GEOMdestroy_obj(obj);
}

void CPCObj::DrawBoundingBox(GEOMedit_list *Output) {
    GEOMobj* BoxObj;
    int i;
    char Name [256], temp [256];
    // Create our GEOM object to add the labels to:-
    BoxObj = GEOMcreate_obj(GEOM_POLYTRI, NULL);
    // Add the bounding box:-
    float verts [5] [3], colors [5] [3];
    verts [0] [0] = 0;
    verts [0] [1] = 0;
    verts [0] [2] = 0;
    verts [1] [0] = 0;
    verts [1] [1] = ScreenHeight;
    verts [1] [2] = 0;
    verts [2] [0] = 0;
    verts [2] [1] = ScreenHeight;
    verts [2] [2] = 0;
    verts [3] [0] = 0;
    verts [3] [1] = ScreenHeight;
    verts [3] [2] = ScreenHeight;
    verts [4] [0] = ScreenWidth;
    verts [4] [1] = ScreenHeight;
    verts [4] [2] = ScreenHeight;
    colors [0] [0] = 1.0;
    colors [0] [1] = 0.0;
    colors [0] [2] = 0.0;
    colors [1] [0] = 0.0;
    colors [1] [1] = 1.0;
    colors [1] [2] = 0.0;
    colors [2] [0] = 0.0;
    colors [2] [1] = 0.0;
    colors [2] [2] = 1.0;
    colors [3] [0] = 0.0;
    colors [3] [1] = 0.0;
    colors [3] [2] = 1.0;
    colors [4] [0] = 1.0;
    colors [4] [1] = 1.0;
    colors [4] [2] = 1.0;
    GEOMadd_polyline(obj, (float *) &verts, (float *) &colors, 5, GEOM_COPY_DATA);
    // Now we replace the geometry for the object named
    // "bounding box" to this obj:-
    GEOMedit_geometry(*Output, Name, obj);
    GEOMdestroy_obj(obj);
}
verts [2] [0] = ScreenWidth;
verts [2] [1] = ScreenHeight;
verts [2] [2] = 0;

 verts [3] [0] = ScreenWidth;
 verts [3] [1] = 0;
 verts [3] [2] = 0;

 verts [4] [0] = verts [0] [0];
 verts [4] [1] = verts [0] [1];
 verts [4] [2] = verts [0] [2];

for (i = 0; i < 5; i++)
{
  colors [i] [0] = 0.0;
  colors [i] [1] = 0.0;
  colors [i] [2] = 0.0;
}

// Set the bounding box name:-
sprintf((char *) & Name, "Bounding Box");
// Add the bounding box to the BoxObj:-
GEOMadd_polyline(BoxObj, (float *) &verts, (float *) &colors, 5, GEOM_COPY_DATA);
// Add the BoxObj to the GEOM:-
GEOMedit_geometry(*Output, Name, BoxObj);
// Destroy the object:-
GEOMdestroy_obj(BoxObj);
}

void CPCObj::DrawLabels(GEOMedit_list *Output)
{
  GEOMobj *LabelObj;
  int i;
  char Name [256], temp [256];
  float verts [2] [3], colors [3];

  int Labels = GEOMcreate_label_flags(GEOMget_font_number("Courier", 0, 0),
                                      0, 1, 0, GEOM_LABEL_CENTER, 0);

  // Create our GEOM object to add the labels to:-
  LabelObj = GEOMcreate_label(GEOM_NULL, Labels);

  // Iterate through each dimension:-
  for (i = 0; i < NumDimensions(); i++)
  {
    // Use verts to represent offset, and the reference point:-
    verts [0] [0] = ScreenLocation [i];
    verts [0] [1] = ScreenHeight + 1;
    verts [0] [2] = 0;
    verts [1] [0] = 0;
    verts [1] [1] = 0;
    verts [1] [2] = 0;
    colors [0] = 0.0;
    colors [1] = 0.0;
    colors [2] = 0.0;

    // Add the label to the LabelObj object:-
    GEOMadd_label(LabelObj, Label(i), verts [0],
                  verts [1], 0.06, colors, -1);
  }

  // Set the object name:-
  sprintf((char *) & Name, "Labels");
  // Pass the new object to the GEOM:-
  GEOMedit_geometry(*Output, Name, LabelObj);
  // Destroy the object:-
  GEOMdestroy_obj(LabelObj);
}

// Draws the Bubble Objects and the Best/Worst Lines
void CPCObj::DrawBubbles(GEOMedit_list *Output)
{
```c
int NRBubbles, NrGroup, Min, Max, MinValue, MaxValue, i, j, k;
CPoint ThisValue;
char Name[256], LName[256], FName[256];
CHoldData FLine, LLine;

float BCoord[400][3], BRadii[400];
float FColor[2][3], LColor[2][3];
float BColors[400][3];

NRBubbles = 0; NrGroup = 10;

FLineObj = GEOMcreate_obj(GEOM_POLYTRI, NULL);
LLineObj = GEOMcreate_obj(GEOM_POLYTRI, NULL);
sprintf((char *) &FName, "Fist Best Line");
sprintf((char *) &LName, "Last Best Line");

for (i = 0; i < NumDimensions() - 1; i++)
{
    for (j = 0; j < ValueOnAxis(i); j++)
    {
        ThisValue = GetCoord(i, Data->Min[i] + j + 1);
        FLine.Line[0][0] = ThisValue.x;
        FLine.Line[0][1] = ThisValue.y;
        FLine.Line[0][2] = 0;
        LLine.Line[0][0] = ThisValue.x;
        LLine.Line[0][1] = ThisValue.y;
        LLine.Line[0][2] = 0;
        FColor[0][0] = 0.90;
        FColor[0][1] = 0.0;
        FColor[0][2] = 0.0;
        LColor[0][0] = 0.20;
        LColor[0][1] = 0.0;
        LColor[0][2] = 0.90;
        // Computes the Best and Worst lines
        Min = 0; Max = 0; MinValue = 1700; MaxValue = -1;
        for (k = 0; k < ValueOnAxis(i + 1); k++)
        {
            if (BestLines(i, j, k) >= MaxValue)
            {
                MaxValue = BestLines(i, j, k);
                Max = k;
            }    
            if (BestLines(i, j, k) <= MinValue)
            {
                MinValue = BestLines(i, j, k);
                Min = k;
            }
        }
        // For further extensions the lines are polylines
        ThisValue = GetCoord(i + 1, Data->Min[i + 1] + j + 1);
        FLine.Line[1][0] = ThisValue.x;
        FLine.Line[1][1] = ThisValue.y;
        FLine.Line[1][2] = 0;
        FColor[1][0] = 0.90;
        FColor[1][1] = 0.0;
        FColor[1][2] = 0.0;
        GEOMadd_polyline(FLineObj, (float *) &FLine.Line,
                         (float *) &FColor, 2, GEOM_COPY_DATA);
        ThisValue = GetCoord(i + 1, Data->Min[i + 1] + j + 1);
        LLine.Line[1][0] = ThisValue.x;
        LLine.Line[1][1] = ThisValue.y;
        LLine.Line[1][2] = 0;
        LColor[1][0] = 0.20;
        LColor[1][1] = 0.0;
        LColor[1][2] = 0.90;
        GEOMadd_polyline(LLineObj, (float *) &LLine.Line,
                         (float *) &LColor, 2, GEOM_COPY_DATA);
    }
}
```

(float *) &LColor, 2, GEOM_COPY_DATA);
}
}

// The frequencies are computed relative to the whole data-set
GEOEdit_geometry(*Output, FName, FLineObj);
GEOEdit_geometry(*Output, LName, LLineObj);
GEOMdestroy_obj(FLineObj);
GEOMdestroy_obj(LLineObj);
for (i = 0; i < NumDimensions(); i++)
{
    for (j = 0; j < ValueOnAxis(i); j++)
    {
        ThisValue = GetCoord(i, Data->Min[i] + j + 1);
        BCoord[NRBubbles][0] = ThisValue.x;
        BCoord[NRBubbles][1] = ThisValue.y;
        BCoord[NRBubbles][2] = 0;
        BColors[NRBubbles][0] = 0.3;
        BColors[NRBubbles][1] = 5.0;
        BColors[NRBubbles][2] = 0.8;

        // The relative radii of the Bubbles
        BRadii[NRBubbles] = 0.4 +
            0.2 * BestPoint(i, j) / NumDimensions() / NrGroup;
        NRBubbles++;
    }
}
sprintf((char *) &Name, "BUBBLE");
BubbleObj = GEOcreate_sphere
    (GEOM_NULL, (float *) &BCoord, (float *) &BRadii,
    GEOM_NULL, (unsigned long *) &BColors, NRBubbles++, GEOM_COPY_DATA);
GEOEdit_geometry(*Output, Name, BubbleObj);
GEOMdestroy_obj(BubbleObj);

// Routines to pick objects:-
void CPCObj::SetSelectableObjects(GEOEdit_list *Output)
{
    // Set the axis to selectable:-
    GEOEdit_selection_mode(*Output, "Dimension Axis", "notify", BUTTON_DOWN |
        BUTTON_MOVING | BUTTON_UP);
}

int CPCObj::HandleMouseFlags(upstream_geom * pick_info)
{
    if (pick_info->flags & BUTTON_DOWN)
        return BUTTON_DOWN;
    if (pick_info->flags & BUTTON_UP)
        return BUTTON_UP;
    if (pick_info->flags & BUTTON_MOVING)
        return BUTTON_MOVING;
    return 0;
}

// Functions to handle axis.
int CPCObj::GetAxisPoint(upstream_geom * pick_info, float & YPosition)
{
    int PickedAxis, i = 0;
    if (! strncmp ((char *) & pick_info->picked_obj, "Dimension Axis", 14))
while (fabs(ScreenLocation[i] - pick_info->mscoord[0]) > 0.0001) && (i < NumDimensions())
    i++;
YPosition = pick_info->mscoord[1];
if (i < NumDimensions())
    return i;
else
    return -1;
}
return -1;
}
void CPCObj::HandleAxis(upstream_geom * pick_info, int MouseInfo)
{
    float IntersectedAxisPos, Percentage;
    int IntersectedAxis, BrushSideToMove, i;
    char Out[10000], Temp[256];

    if (!strncmp((char *)&pick_info->picked_obj, "Dimension Axis", 14))
    {
        // Button just gone down code:-
        if (MouseInfo == BUTTON_DOWN)
        { // The mouse button has just been pressed:-
            AxisInOperation = GetAxisPoint(pick_info, IntersectedAxisPos);
            // Add the dimension data to the text box:-
            sprintf(Out, " %s
", Question(AxisInOperation));
            for(i=0; i < ValueOnAxis(AxisInOperation); i++)
            {
                sprintf((char *)&Temp," %i - %s = %i
", RealValues(AxisInOperation, i), RealLabel(AxisInOperation, i),
                        BestPoint(AxisInOperation, i));
                strcat(Out, (char *)&Temp);
            }
            strcat(Out, "\n");
            AVSmodify_parameter("Sample Information", AVS_VALUE, (char *)&Out, "", "");
        }
    }
}
Appendix B – Source Code for Fisheye View

(Source code in Pascal for the different distortion techniques for fisheye view)

Var FishFld : Byte;

Constructor TSphereChild.Init(AParent: PWindowsObject; ChildNum: Integer);
Var
  ChildNumStr : array [0..5] of Char;
  TitleStr : array [0..12] of Char;
begin
  Str(ChildNum, ChildNumStr);
  StrCat(StrECopy(TitleStr, 'Picture #', ChildNumStr);
  Num := ChildNum;
  FishFocus.X := 1;
  FishFocus.Y := 1;
  Mode := FishFld;
  inherited Init(AParent, TitleStr);
  With Attr Do Begin W:=400; H:=250; End;
end;

Procedure TSphereChild.SetupWindow;
begin
  inherited SetupWindow;
end;

Procedure TSphereChild.ReDraw;
Begin
  InvalidateRect(HWindow,nil,truc);
  UpdateWindow(HWindow)
End;

procedure TSphereChild.Paint(DC: HDC; var PS: TPaintStruct);
Var
  CH: Char;
begin
  SaveDC(DC);
  Symbol := '**';
  DistFactor := 1.5;
  GaussDistF := 5;
  PatternSize := 2;
  SetTextColor(DC, RGB(0, 0, 0));
  If Mode = 1 Then Cartezian(DC, 1,FishFocus.X,FishFocus.Y);
  If Mode = 2 Then Polar(DC, 1,FishFocus.X,FishFocus.Y);
  If Mode = 3 Then Circle(DC, 1,FishFocus.X,FishFocus.Y);
  If Mode = 4 Then Gauss(DC, 1,FishFocus.X,FishFocus.Y);
  If Mode = 5 Then Rectangle(DC, 1,FishFocus.X,FishFocus.Y);
  RestoreDC(DC, -1);
end;
Procedure TSphereChild.Cartezian (DC:HDC; ViewMod: Byte; FocusX, FocusY: Word);

Function G( Arg : Real) Real;
Begin
  G := (DistFactor + 1)*Arg/(DistFactor*Arg + 1)
{function g2 in cartezian transformation}
End;

Var X,Y,FXX,FYY : Integer;
I,J : Integer;
SX : String;
SXF,SYF : PChar;

Begin
  I:= 0;
  While I < Attr.H Do
    Begin
      J := 0;
      While J < Attr.W Do
        Begin
          If J <= FocusX Then
            Begin
              FXX := FocusX;
              X := Trunc(-G((FocusX-I)/FXX)*FXX+FXX)
              {normalising the fisheye position}
            End
          Else
            Begin
              FXX := Attr.W – FocusX;
              X := Trunc(G((J – FocusX)/FXX)*FXX + FocusX);
            End;
          If I <= FocusY Then
            Begin
              FYY := FocusY;
              Y := Trunc(-G((FocusY-I)/FYY)*FYY + FYY)
            End
          Else
            Begin
              FYY := Attr.H-FocusY;
              Y := Trunc(G((I - FocusY)/FYY)*FYY+FocusY);
            End;
          If Not Ellipse (DC,X-PatternSize, Y-PatternSize, X+PatternSize, Y+PatternSize)
            Then Begin End;
          J:=J+10;
End;
I := 1 + 10;
End;
End;

Procedure TSphereChild.Polar( DC: HDC; ViewMod: Byte; FocusX, FocusY : Word);
{polar coordinate transformation}
Function Pol( ArgM, ArgN : Extended) : Extended;
Var At : Extended;
Begin
   At := (DistFactor+1)*ArgM*ArgN/(DistFactor*ArgN/ArgM + l)/ArgM;
P = At;
End;

Var A,D1,D2,SinTeta, RNorm, RMax : Extended;
X,Y,FX,FY,I,J : Integer;
Vl,V2: Extended;
Begin
FX := FocusX;
FY := FocusY;
I := 0;
While I < Attr.H Do
Begin
   J := 0;
   While J < Attr.W Do
   Begin
      Vl := J-FX; V2 := I-FY;
      RNorm := Sqrt(Vl*Vl + V2*V2);
      If (FX < > J) And (FY < > I) Then
      Begin
         A := (FX-J)/(FY-I);
         SinTeta := Abs(FY-I)/RNorm;
         IF (J > FX) And (I < FY) Then
         Begin
            Vl := (FX+A*I-J); V2 := FY;
            D1 := SQRT(Vl*Vl + V2*V2);
            Vl := (FX-Attr.W); V2 := (FY-I-(Attr.W-J)/A);
            D2 := SQRT(Vl*Vl+V2*V2);
            IF D1 < D2 Then RMax := D1 Else RMax := D2;
            X := FX + Trunc(Pol(RMax, RNorm) * SQRT(1-SQR(SinTeta)));
            Y := FY - Trunc(Pol(RMax, RNorm)*SinTeta);
         End;
         IF (J > FX) And (I > FY) Then
         Begin
            V1 := (FX-Attr.W); V2 := (FY-I-(Attr.W-J)/A);
            D1 := SQRT(V1*V1+V2*V2);
         End;
      End;
   End;
End;
V1 := (FX - J - A*(Attr.H - I)); V2 := (FY - Attr.H);
D2 := SQRT(V1*V1 + V2*V2);
IF D1 < D2 Then RMax := D1 Else RMax := D2;
X := FX + Trunc(Pol(RMax, RNorm)*SQRT(1 - SQR(SinTeta)));
Y := FY + Trunc(Pol(RMax, RNorm)*SinTeta);
End;
If (J < FX) And (I < FY) Then
Begin
V1 := (FX + A*I - J); V2 := FY;
D1 := SQRT(V1*V1 + V2*V2);
V1 := (FY - I + J/A); V2 := FX;
D2 := SQRT(V1*V1 + V2*V2);
IF D1 < D2 Then RMax := D1 Else RMax := D2;
X := FX - Trunc(Pol(RMax, RNorm)*SQRT(1 - SQR(SinTeta)));
Y := FY - Trunc(Pol(RMax, RNorm)*SinTeta);
End;
If (J < FX) And (I > FY) Then
Begin
V1 := (FX - A*Attr.H - I) - J);
V2 := (FY - Attr.H);
D1 := SQRT(V1*V1 + V2*V2);
V1 := (FY - I + J/A); V2 := FX;
D2 := SQRT(V1*V1 + V2*V2);
IF D1 < D2 Then RMax := D1 Else RMax := D2;
X := FX - Trunc(Pol(RMax, RNorm)*SQRT(1 - SQR(SinTeta)));
Y := FY + Trunc(Pol(RMax, RNorm)*SinTeta);
End;
Else
Begin
If FX = J Then
Begin
If I < FY Then RMax := FY Else RMax := Attr.H - FY;
Y := FY + Trunc(Pol(RMax, RNorm)*SinTeta);
X := FX
End;
If FY = I Then
Begin
If J < FX Then RMax := FX Else RMax := Attr.W - FX;
X := FX - Trunc(Pol(RMax, RNorm)*SQRT(1 - SQR(SinTeta)));
Y := FY
End;
End;
If Not Ellipse(DC, X-PatternSize, Y-PatternSize, X+PatternSize, Y+PatternSize)
Then Begin End;
End;
I = 1 + 10;
End;

Procedure TSphereChild.Circle( DC : HDC; ViewMod: Byte; FocusX, FocusY : Word);
{applying on a circle area the polar transformation function}
Function Pol( ArgM, ArgN : Extended) : Extended;
Var At : Extended;
Begin
At := (DistFactor + 1)*ArgM*ArgN/(DistFactor*ArgN/ArgM + 1)/ArgM;
Pol := At;
End;

Var A, D1, D2, SinTeta, RNorm, RMax : Extended;
X, Y, FX, FY, I, J : Integer;
V1, V2 : Extended;
Begin
FX := FocusX;
FY := FocusY;
I := 0;
If FX < Attr.W - FX Then D1 := FX Else D1 := Attr.W - FX;
If FY < Attr.H - FY Then D2 := FY Else D2 := Attr.H - FY;
If D1 < D2 Then RMax := D1 Else RMax := D2;
While I < Attr.H Do
Begin J := 0;
While J < Attr.W Do
Begin
V1 := J - FX; V2 := I - FY;
RNorm := SQRT(V1*V1 + V2*V2);
If RNorm < RMax Then
Begin
If (FX = J) And (FY = I) Then
Begin
A := (FX-J)/(FY-I);
SinTeta := Abs(FY-I)/RNorm;
If (J > FX) And (I < FY) Then
Begin
X := FX + Trunc(Pol(RMax, RNorm)*SQRT(1 - SQR(SinTeta)));
Y := FY - Trunc(Pol(RMax, RNorm)*SinTeta);
End;
If (J > FX) And (I > FY) Then
Begin
X := FX + Trunc(Pol(RMax, RNorm)*SQRT(1 - SQR(SinTeta)));
Y := FY + Trunc(Pol(RMax, RNorm)*SinTeta);
End;
End;
If \((J < FX)\) And \((I < FY)\) Then

Begin

\[
X := FX - \text{Trunc}(\text{Pol}(RMax, RNorm) \cdot \sqrt{1 - \sin^2 \theta})
\]
\[
Y := FY - \text{Trunc}(\text{Pol}(RMax, RNorm) \cdot \sin \theta)
\]

End;

If \((J < FX)\) And \((I > FY)\) Then

Begin

\[
X := FX - \text{Trunc}(\text{Pol}(RMax, RNorm) \cdot \sqrt{1 - \sin^2 \theta})
\]
\[
Y := FY + \text{Trunc}(\text{Pol}(RMax, RNorm) \cdot \sin \theta)
\]

End;

Else

Begin

If \(FX = J\) Then

Begin

\[
Y := FY + \text{Trunc}(\text{Pol}(RMax, RNorm) \cdot \sin \theta)
\]
\[
X := FX
\]

End;

If \(FY = I\) Then

Begin

\[
X := FX - \text{Trunc}(\text{Pol}(RMax, RNorm) \cdot \sqrt{1 - \sin^2 \theta})
\]
\[
Y := FY
\]

End;

End;

Else

Begin

\[
X := J; Y := I;
\]

End;

If Not Ellipse(DC, X-PatternSize, Y-PatternSize, X+PatternSize, Y+PatternSize) Then Begin End;

\[
J = J + 10;
\]

End;

\[
I = I + 10;
\]

End;

Procedure TSphereChild.Rectangle( DC : HDC; ViewMod: Byte; FocusX, FocusY : Word);

{applying on a rectangle area the cartesian transformation function}

Function Pol( ArgM, ArgN : Extended) : Extended;

Var At: Extended;

Begin

\[
At := (\text{DistFactor} + 1) \cdot \text{ArgM} \cdot \text{ArgN} / (\text{DistFactor} \cdot \text{ArgN} / \text{ArgM} + 1)/\text{ArgM};
\]

Pol := At;

End;

Var A, D1, D2, SinTeta, RNorm, RMax : Extended;

X, Y, FX, FY, I, J : Integer;
V1, V2, Diam: Extended;

Begin
FX := FocusX;
FY := FocusY;
I := 0;
If FX < Attr.W - FX Then D1 := FX Else D1 := Attr.W - FX;
If FY < Attr.H - FY Then D2 := FY Else D2 := Attr.H - FY;
If D1 < D2 Then Diam := D1 Else Diam := D2;
While I < Attr.H Do
Begin J := 0;
While J < Attr.W Do
Begin
V1 := J - FX; V2 := I - FY;
RNorm := SQRT(V1*V1 + V2*V2);
If RNorm < Diam Then
Begin
  If (FX > J) And (FY > I) Then
  Begin
    A := (FX - J)/(FY - I);
    SinTeta := Abs(FY - I)/RNorm;
    IF (J > FX) And (I < FY) Then
    Begin
      V1 := FX - A*(FY - Diam - I) - J; V2 := Diam;
      D1 := SQRT(V1*V1 + V2*V2);
      V1 := Diam; V2 := (FY - I - (FX + Diam - J)/A);
      D2 := SQRT(V1*V1 + V2*V2);
      IF D1 < D2 Then RMax := D1 Else RMax := D2;
      X := FX + Trunc(Pol(RMax, RNorm)*SQRT(1 - SQRT(SinTeta)));
      Y := FY - Trunc(Pol(RMax, RNorm)*SinTeta);
    End;
    If (J > FX) And (I > FY) Then
    Begin
      V1 := Diam; V2 := (FY - I - (FX + Diam - J)/A);
      D1 := SQRT(V1*V1 + V2*V2);
      V1 := (FX - J - A*(FY + Diam - I));
      V2 := Diam;
      D2 := SQRT(V1*V1 + V2*V2);
      IF D1 < D2 Then RMax := D1 Else RMax := D2;
      X := FX + Trunc(Pol(RMax, RNorm)*SQRT(1 - SQRT(SinTeta)));
      Y := FY + Trunc(Pol(RMax, RNorm)*SinTeta);
    End;
    If (J < FX) And (I < FY) Then
    Begin
      V1 := (FX - A*(FY - Diam - I) - J); V2 := Diam;
      D1 := SQRT(V1*V1 + V2*V2);
      V1 := (FY - I + (FX - Diam - J)/A); V2 := Diam;
    End;
  End;
End;
D2 := SQRT(V1*V1+V2*V2);
IF D1 < D2 Then RMax := D1 Else RMax := D2;
X := FX - Trunc(Pol(RMax,RNorm)*SQRT( 1 - SQR(SinTeta)));
Y := FY - Trunc(Pol(RMax,RNorm)*SinTeta);

End;

If (J < FX) And (I > FY) Then Begin

V1 := (FX - A*(FY+Diam-I)-J);
V2 := Diam;
D1 := SQRT(V1*V1 + V2*V2);
V1 := (FY-I+(FX-Diam-J)/A); V2 := Diam;
D2 := SQRT(V1*V1+V2*V2);
IF D1 < D2 Then RMax := D1 Else RMax := D2;
X := FX - Trunc(Pol(RMax,RNorm)*SQRT( 1 - SQR(SinTeta)));
Y := FY + Trunc(Pol(RMax,RNorm)*SinTeta);

End;

Else Begin

If FX = J Then Begin

If I < FY Then Rmax := FY Else RMax := Diam;
Y := FY + Trunc(Pol(RMax,RNorm)*SinTeta);
X := FX

End;

If FY = I Then Begin

If J < FX Then RMax := FX Else RMax := Diam;
X := FX-Trunc(Pol(RMax,RNorm)*SQRT( 1 - SQR(SinTeta)));
Y := FY

End;

End;
Else Begin

X := J; Y := I

End;

If Not Ellipse(DC,X-PatternSize, Y-PatternSize, X+PatternSize, Y+PatternSize) Then Begin End;

J := J+10;
I := I + 10;

End;

Procedure TSphereChild.Gauss( DC : HDC; ViewMod: Byte; FocusX, FocusY : Word);
Function GaussF( ArgM, ArgN : Extended) : Extended;
{ the gauss transformation function }
Var Beta : Extended;
Begin
Beta := (Ln(GaussDistF) – Ln(ArgM))/(ArgM*ArgM);
GaussF := ArgM – GaussDistF*Exp(-Beta*ArgN*ArgN);
End;

Var D1, D2, SinTeta, RNorm, RMax : Extended;
X, Y, FX, FY, I, J, K : Integer;
V1, V2 : Extended;
Begin
FX := FocusX;
FY := FocusY;
I := 0;
If FX < Attr.W - FX Then D1 := FX Else D1 := Attr.W - FX;
If FY < Attr.H - FY Then D2 := FY Else D2 := Attr.H - FY;
If D1 < D2 Then RMax := D1 Else RMax := D2;
While I < Attr.H Do Begin J := 0;
While J < Attr.W Do
Begin
V1 := J - FX; V2 := I - FY;
RNorm := Sqrt(V1*V1 + V2*V2);
If RNorm < RMax Then
Begin
If (FX < J) And (FY > I) Then
Begin
SinTeta := Abs(FY - I)/RNorm;
If (J > FX) And (I < FY) Then
Begin
X := FX + Trunc(GaussF(RMax, RNorm) * SQRT(1 - SQR(SinTeta)));
Y := FY - Trunc(GaussF(RMax, RNorm) * SinTeta);
End;
End;
End;
End;
End;
End;
End;
End;
End;
End;
End;
End;
End;
End;
End;
End;
Y := FY + Trunc(GaussF(RMax,RNorm)*SinTeta); End; End Else Begin If FX = J Then Begin
Y := FY+Trunc(GaussF(RMax,RNorm)*SinTeta);
X := FX End; If FY = I Then Begin
X := FX-Trunc(GaussF(RMax,RNorm)*SQRT(1-SQR(SinTeta)));
Y := FY End; End; End Else Begin
X := J; Y := I; End; If Not Ellipse (DC,X-PatternSize,Y-PatternSize,X+PatternSize,Y+PatternSize) Then Begin End; J := J+10; End; I := I + 10; End;

Procedure TSphereChild.WMLButtonDown(var Msg: TMessage); Begin
FishFocus.X := Msg.LParamLo;
FishFocus.Y := Msg.LParamHi;
Redraw; End;

Procedure TSphereChild.WMSize(Var Msg: TMessage); Begin
Inherited WMSize(Msg); End;

Procedure TFishMDIWindow.SetupWindow; Var ARect: TRect;
NewChild: PSpereWindow
Begin
inherited SetupWindow;
End;
procedure TFishMDIWindow.Cartezian;
  Begin
    FishFld := 1;
  End;

procedure TFishMDIWindow.Polar;
  Begin
    FishFld := 2;
  End;

procedure TFishMDIWindow.Circle;
  Begin
    FishFld := 3;
  End;

Procedure TFishMDIWindow. Gauss;
  Begin
    FishFld := 4;
  End;

Procedure TFishMDIWindow. Rectangle;
  Begin
    FishFld := 5;
  End;

Procedure TFishMDIWindow. CMHelpAbout(var Msg: TMessage);
  Begin
    Application ^ ExecDialog(New(PDialog, Init(@SeIf, PChar(id_About))));
  End;

Function TFishMDIWindow.CreateChild: PWindowsObject;
Var   ChildNum: Integer;

Function NumberUsed(P: PSphereWindow): Boolean; far;
  Begin
    NumberUsed := ChildNum = P^.Num;
  End;

Begin
  ChildNum := 1;
  While FirstThat(@NumberUsed) <> nil do Inc(ChildNum);
CreateChild := Application ^ MakeWindow(New(PSphereWindow, Init(@SeIf, ChildNum)));
End;

Procedure TFishApp.InitMainWindow;
Begin
MainWindow:New(PMyMDIWindow,Init('Fisheye Application',LoadMenu(HInstance, PChar(100))));

End;

var
MDIApp: TFishApp;
Begin
  MDIApp.Init('FishEye Application');
  MDIApp.Run;
  MDIApp.Done;
End.
Appendix C - Publication List

Articles


*Majör Andrea: Imagini Stereo, PCReport* 02. 1995 Nr. 29, Románia


Books

*Majör Andrea - Major Levente: Számítástechnikai feladatok*, Mentor kiadó, Marosvásárhely, 1995

Appendix D - Summary

The study covers the topic of information visualisation, from the definition to the introduction of some of the visualisation techniques. Besides presenting the most well-known techniques, individual research is placed within the framework set.

The study starts with an introduction of the term, then details steps of the visualisation method and introduces a categorisation of the information visualisation. Given various taxonomies, a comprehensive model, called Visualisation Classification Model has been introduced. It is based on existing taxonomies as well as on trends in information visualisation. The aim of this model is to place the presented techniques along a framework.

The techniques presented are grouped based around the data model they are mainly applied on. One of the individual research area detailed in the study falls under the multidimensional data visualisation, as an extension of the Parallel Coordinates technique. When developed further the most important factor was the usability and applicability of the enhancement. The Bubble Method, arose from the need of adding additional information to the already visualised data, in order to help decision makers. It was not a simple extension of the dimension, but was about adding additional information to the already visualised data set. This helped to spot trends, feature, which would have required detailed data research.

In the case below, taken a marketing database with over 250 dimensions, the size of the
bubbles were given by the relative frequency of that value within the range of values for that dimension. In addition the colour of the lines connecting the values on the parallels have their meanings as well. In my case, this was assigned to the most demanded next variable choice (red line) and the least preferred (purple line).

With the increase of information visualisation the size of display remains constant. If one would like to visualise extra information, than it happens at the expense of the other information.

As part of the study a separate area has been devoted for the workplace visualisation. Displaying detailed pictures has the advantage of showing the global, entire picture but has the drawback that details are too small to be seen. Alternatively, zooming into a part of the picture and panning to other parts does show local details but loses the overall structure of the picture. A fisheye-view of a picture shows an area of interest quite large and with detail and shows other areas successively smaller and in less detail. It achieves this smooth integration of local detail and global context by repositioning and resizing elements of the picture.

The individual research of this topic touched two areas of interest: the distortion function – computing the position of the elements and the area of applicability.

An application has been coded, providing interactivity and allowing to explore distortion features of different functions. In addition, this application has been extended on the area of applicability, ie the fisheye has been applied on either a rectangle area or a circle on, but not on the whole image. This approach had the benefit of increasing speed, as only a limited area needed re-calculation.
The study does not aim to cover all the existing techniques, but aims to present a framework to help pulling together a nowadays highly demanded area of applicability of the computer graphics.