eSCAPE

The IO_Dencies System Design and Visualization Techniques

ABSTRACT

The IO_Dencies project is an interactive art installation that has been conceived by the art group Knowbotic Research of Koeln, Germany and is beeing designed by the ZKM, Karlsruhe, Germany. It is an eScape which sees itself as a collection of interfaces and intersections. The interaction of people within IO_Dencies creates an networked solution of a textual discourse.

The user is not represented by avatars, but is represented by the visualisation of his/her browsing activities in a virtual information space. The space is populated by documents that are localized within the space. The processes within the space are made available through a force field that is based on the constituents and their properties within the space. The visuals are to be particle streams in the force field. When one zooms in on processes in the visualised particle shreams, the representation will become more textual and keywords appear. Through choosing those keywords that actual documents that make the information space are accessible.

In an experiment to condense different fields of urban thinking in a networked environment, IO_Dencies will be placed in a local context.

Document	eSCAPE-ZKM-123
Type	Deliverable
Status	Final
Version	2.0
Date	29.4.98
Author(s)	A. Schiffler (ZKM), D. Schwabe (ZKM)
Task	3.1 – 3.3

Table of Contents

Introduction

The IO_Dencies project is an art installation that is a joint project of Knowbotic Research and the ZKM. Knowbotic research has in previous works explored the idea of locations – or rather non-locations. It is these information environments make up their work, which are of interest in the context of eScapes. In their projects, Knowbotic Research do not deal with text-based realities and can therefore only formulate conjectures about their conditions and qualities. Therefore they do not think of an information-environment as a static landscape.

Space is always articulated and constructed, before it can be observed. An eSCAPE should become a potential in which individual articulations of space are enabled. In shared environments, these actions can be combined into collective and connective experiences in order to make one conscious of co-operative action and its inherent constructive ideas. To unfold such a potentiality of eSCAPE means to extend the traditional spatial experience. This can be done by adding a sensibility for polycontextual relations through the simultaneous access to different representations of the same environment, the display of invisible connections and relationships, and the presentation of and interaction with dynamic change.

The user is not represented by avatars, but is represented by the visualisation of his/her browsing activities (i.e. moving a IO_Dencies-browser window), which are unfolding the processes inside through forcefields that are generated from each document in the information environment.

The tendencies forcefield can be represented visually, textually and audibly. The visuals are to be particle streams in the force field. When one zooms in on processes in the visualised particle shreams, the representation will become more textual and keywords appear. By choosing those keywords that actual documents that make the information space are accessible to the user of the IO_Dencies system.

The connections between IO_Dencies and eScapes and the uses for the IO_Dencies system can be

- to make the eSCAPE-idea more dynamic (information generation rather than retrieval).
- to add to the definition of space.
- to extend the aesthetic expectations beyond the visual.
- to extend the notion of visualisation by the actual.
- not to limit the interactivity to the gathering of information and communication, but to include the experience (including the aesthetic one).
- to redefine eSCAPEs as electronic urban situations, rather than electronic landscapes.
- to move from experience to urban agency; that is, to require personal activity - the articulating force in eSCAPEs - which in turn opens places.

Through the development of software and the installation of hardware, IO_Dencies aims to construct a multiuser “discourse”-machine[1] in the city of Sao Paulo which should enable a “discourse” on urban planning. In Sao Paulo, with its immense fragmented geography and ist huge illegally and too densely build cityscape, urbanists have failed in the last years to articulate public spaces where different parts of the society could meet. IO_Dencies tries to answer the question, if it is possible to articulate an urban space via electronic information processes.

[1] A “discourse-machine” is typically a system that enables several people to engage in a written or spoken discourse on a particular topic. A simple example is a telephone and its attached system, where two people can interact by talking to each other. IO_dencies resembles more the typical Usenet News systems that can be found on the internet. People interact by reading and writing text on a particular topic (the newsgroup) and the machinery behind the scene (newsserver and Internet) distributes and organizes all messages.

Features

The IO_Dencies system should incorporate a self-organizing collaborative information environment. The environment is based on the input of specially selected editors who incorporate core documents into the space, the dynamics of users and user interests about the continously updated material and a real-time evolving component that will self-organize the data that is present. This will enable the interaction with the system on different semantic levels.

IO_Dencies theme is on urban dynamics and the dynamics that is produced by the knowledge-generating processes of ist users. It tries to achieve a recontextualization analysis from urban experts (theoreticians) through city collaborators (architects, urban practitioners, ordinary people). It is also an experimental environment which takes another approach to engaged discussionfields (i.e. Internet Newsgroups) by visualising the evolving zones of intensities (information). Furthermore, it offers a hybrid environment in which human and machine (systemic) interactions cooperate.

The visualization of IO_Dencies is primarily 2D in nature. The target platform for ist use by people, is a PC with Internet connectivity and advanced graphics capabilities. A native Windows application and the necessary server programs, database and Web-Interface is beeing produced at the moment by the ZKM.

Figure 1 gives a structural overview of how the interaction of the users (or actors) with the IO_Dencies environment is organized and what results are desired.

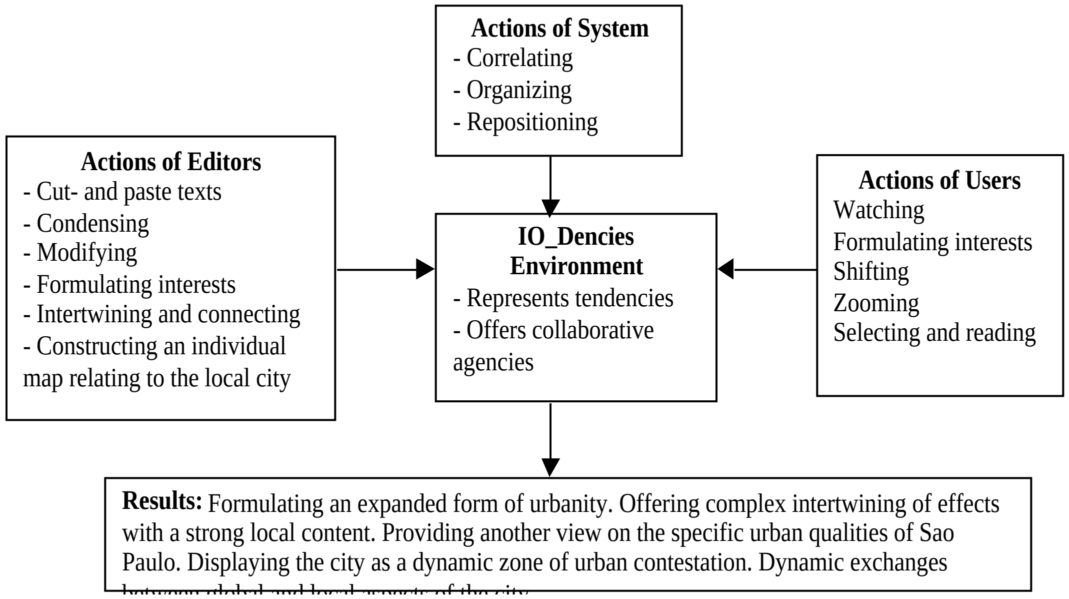

Actions of System
- Correlating
- Organizing
- Repositioning
Actions of Editors
- Cut- and paste texts
- Condensing
- Modifying
- Formulating interests
- Intertwining and connecting
- Constructing an individual map relating to the local city
IO_Dencies Environment
- Represents tendencies
- Offers collaborative agencies
Actions of Users
Watching
Formulating interests
Shifting
Zooming
Selecting and reading
Results: Formulating an expanded form of urbanity. Offering complex intertwining of effects with a strong local content. Providing another view on the specific urban qualities of Sao Paulo. Displaying the city as a dynamic zone of urban contestation. Dynamic exchanges

Figure 1: Structure of IO_Dencies

System Design

Software Overview

The system

There are distinct software components to the whole system. The IO_Dencies system will consist of a server and several client computers in a classic client-server structure (see Figure 2).

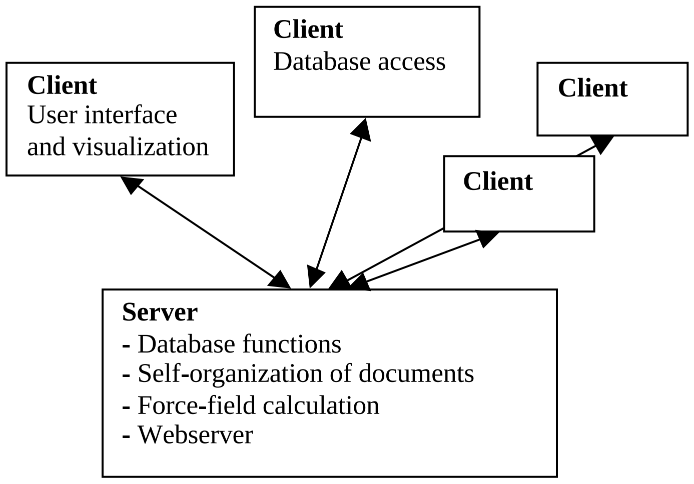


Figure 2: Basic client-server architecture of IO_Dencies.

A single central server controls the IO_Dencies “world”, stores the documents and meta data and provides information to the clients for visualization and reading. Unlike a standard webserver, the IO_Dencies server get a considerable feedback of information – details on the browsing activities of ist users – which it puts back into the system. The clients are the “windows” into the IO_Dencies world. They visualize the information and provide access to the database.

The server computer is running a database engine, a webserver, a self-organization engine and a forcefield engine with client-access capabilities. The database engine to store the documents as well as the meta information linked to these documents. The documents alone are in such a form that easy retrieval through the webserver running on the same computer is possible. Thus all data can be viewed on a user (client) computer through a web browser. The meta data is information that has been added to the document by the editors or the system. The database also contains information that is used for the overall administration of the IO_Dencies system, such as access control information. The self-organization software is a seperate program which will continuously read and update the meta data, trying to “organize” it. One important piece of information is a “location” of each document in the virtual IO_Dencies environment. The self-organization engine will try to change these locations to create new groups and fields. The forcefield engine also reads the meta data but does not change the information (such as the location). It will make use of the documents meta data to generate a force field in the IO_Dencies environment relative the the documents current position. The field is then processed and send to the requesting user for use in the visualization.

The client computer requires a IO_Dencies client program and a standard web browser. The client program will contact the IO_Dencies server, retrieve the force field data and use it to visualize the IO_Dencies environment with particle streams. It will also server as a launch point for the browser to show selected documents. The editors will also require a special Editor program for database access since only they are allowed to add and change information. This will be a Java application which can be aquired with the browser through the Internet.

Initially a 2D version of the visualiser (client) for use on a standard PC is beeing made. In the future the development will concentrate on broadening the spectrum of clients – be it a 3D version, a mobile input unit or sophisticated sound output.

Distinct Software Components

Server OS: RedHat Linux 5.0 with upgrades and custom SMP kernel.

Database: PostgreSQL database server with webserver plugin and Java tools.

Webserver: Apache.

Self organization engine: custom C-program running on server.

Forcefield engine: custom C-program running on server.

Forcefield and keyword visualizer: custom C-program running on client.

Editor tool: custom Java programm running on browser.

Document display: standard web browser with Java capabilities (Netscape, Internet Explorer).

Hardware Specifications

The server system specifications are based on the need for much computational power and sufficient memory to server several clients at once. It is currently set as follows: Dual Pentium II @ 266MHz, 256MB RAM, 4.3GB UW-SCSI2 harddrive, 10/100Mbit ethernet card.

The target client computer should have the following components: Windows 95 compatible PC, SVGA graphics, sufficient memory to run a Java enabled browser (32MB RAM), Internet connection (PPP via modem, ISDN or ethernet).

Details on Software Components

The Editor Tool

There will be several specialists (urbanists, architects, city planners, sociologists, etc.) collaborating on IO_Dencies as editors. Their duty is to create the initial collection of documents and the analysis of the documents on the theme of urban planning and urbanity in general. A document can be a text, a URL, an image, video clip or an audio clip, whichever is appropriate and desired by the editor. There will be the only restriction that it has to be uploadable and storable by the database and must therefore be a single file which can be viewed through a web browser.

Each one of the documents will pass through the assignment of keywords to the respective content by the editors. Furthermore, the editors will implicitly define a set of rules for the self-organization by arranging keywords on a two-dimensional map. These tasks are done by the editors as they see fit; they are therefore not any user of the system, but are selected beforehand by us.

The editor tool will be the main software component for the editors to accomplish these tasks:

- **Document Uploading**

  The uploading of the original documents (texts, images, movies, audio) into the database. The editor will have to pre-process the documents on his/her local computer into a form that is compatible with the IO_Dencies system. This will be accomplished by external programs (OCR software, paint program, audio-recorder and –encoder).

- **Database Maintenance**

  During a work-session of an editor, the tool has to keep track of changes and must organize fetching and updating the data from and to the database. This is mostly for changing already uploaded text and access to some of the meta data that is associated with each document.

- **Keyword Assignment**

  Texts will be divided into segments (or fragments) and each of these is assigned a keyword. The same procedure applies to other types of documents (individual images or audio clips for example). The tool will only have to make sure that keywords are not assigned twice. Once this step is complete, all editors will have access to the document through the keyword.

- **Keyword Placement and Implicit Rule Definition**

  Gives the editor the ability to place an already existing keywords onto a 2D map (thereby creating local groups). Following the placement the Editor can then relate keywords by dragging connecting lines between them. The locations and the connections are implicit rules that will be used by the IO_Dencies system for the self-organization and the force-field generation.

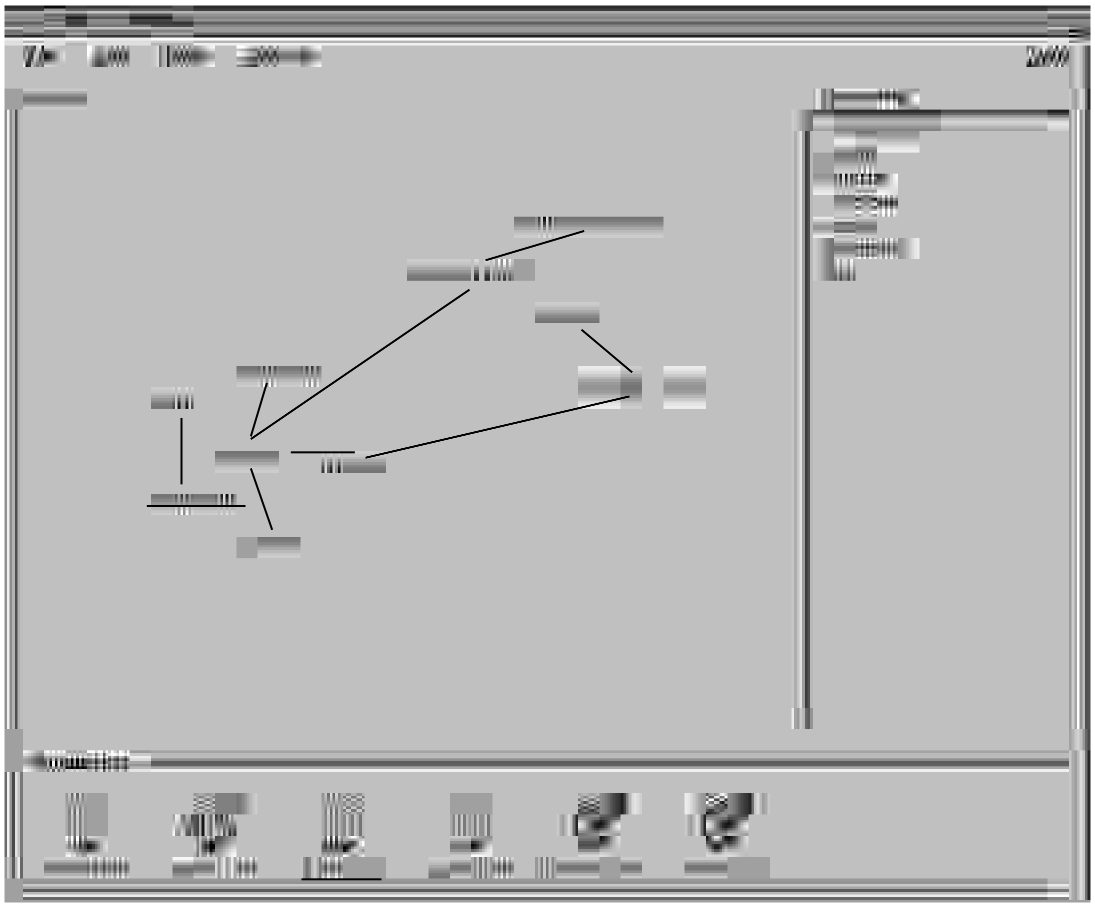


Figure 3: The main panels of the editor tool

In Figure 3, the three main panels of the editor interface are shown.

- **Document Panel**

  All original documents residing in the database are shown as icons, revealing the type of data. A double-click on one of the icons, opens a second window (browser), showing the content and offering the ability to edit the keyword of the document. If the document is text, marking of smaller text blocks to further fragment the document is possible.

- **Keyword Panel**

  Each keyword is shown in this panel. The keywords can originate by the editor himself or can relate to keywords introduced by other editors. By clicking on a keyword and dragging it to the map panel, the keyword can be placed on a two-dimensional rule map. The visible rule map is smaller than the available map space to allow for the placement of hundreds of keywords without interference. Just selecting one keyword in this panel highlights the corresponding original document, a double click opens this document and shows the content related to the keyword (text, image, movie or audio).

- **Map Panel**

  The map panel is the working area, where the editor can place and arrange all keywords from the database. He/she is able to place all listed keywords. By arranging keywords in relation to the position of others, the editor implicitly defines a set of local rules for each keyword which are simply defining the preferred distance to its local neighbours. Additional lines connecting keywords can strengthen the relationships and hirarchies further. It is up to the editor to make those decisions based on the content of the documents behind each keyword.

A sample input panel for text and other data is shown in Figure 4.

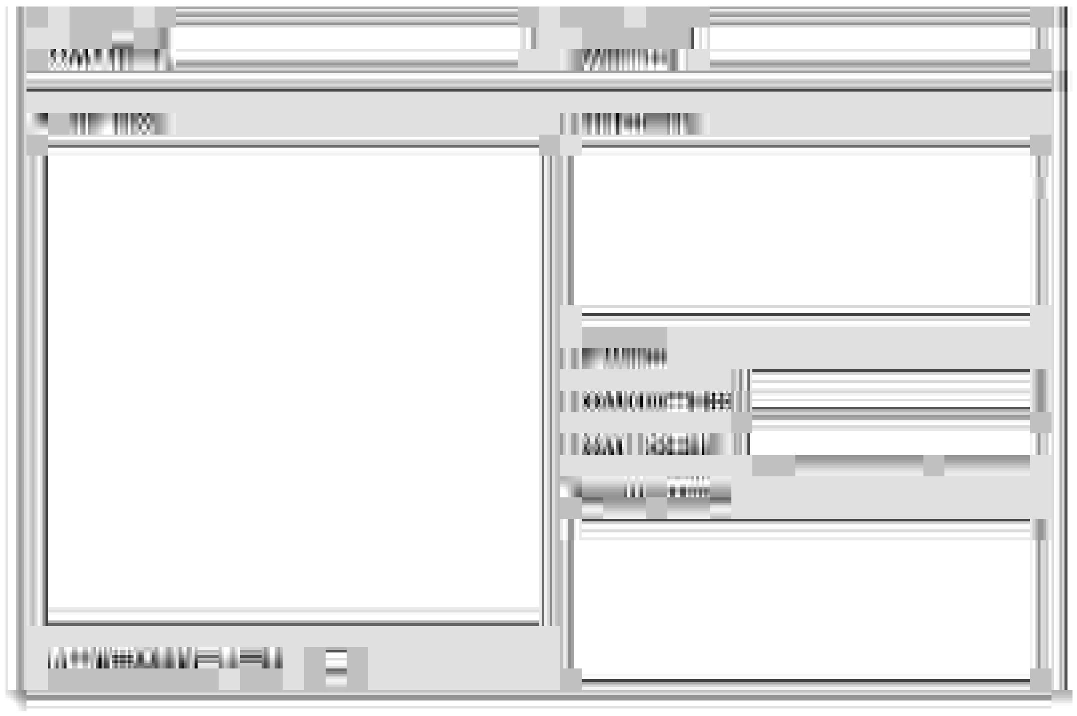


Figure 4: Original document upload panel for the editors.

Self-organization

Based on the two-dimensional keyword maps created by each editors, the self-organisation algorithm will create local behaviour rules for each keyword object. As a first approach the distances of the 5-10 closest neighbours to a specific keyword in each map will define a set of different spring forces for that object. Then, all objects are distributed randomly into a 2D- (or possibly 3D-) environment where their motion equations are solved by simple integration of the total acceleration over time. A reference point will always be the current location and state of a keyword in the IO_Dencies system (as opposed to the editors map).

A diagram explaining how the relations between the keywords are derived, is shown in Figure 5. The keywords in the local editing view form a zone or group merely by their proximity to each other. They are also connected (shown as solid lines) with each other. The difference between their locations in the personal editing view and the locations in the selforganisation state (as indicated by the dashed lines) is the bases for the spring forces. These forces tend to drive the locations of each keyword to the one selected by the editor. Since there are many editors, there are many more spring forces acting on each keyword than are shown here. In effect, the differences between each editors location and the world location are combined.

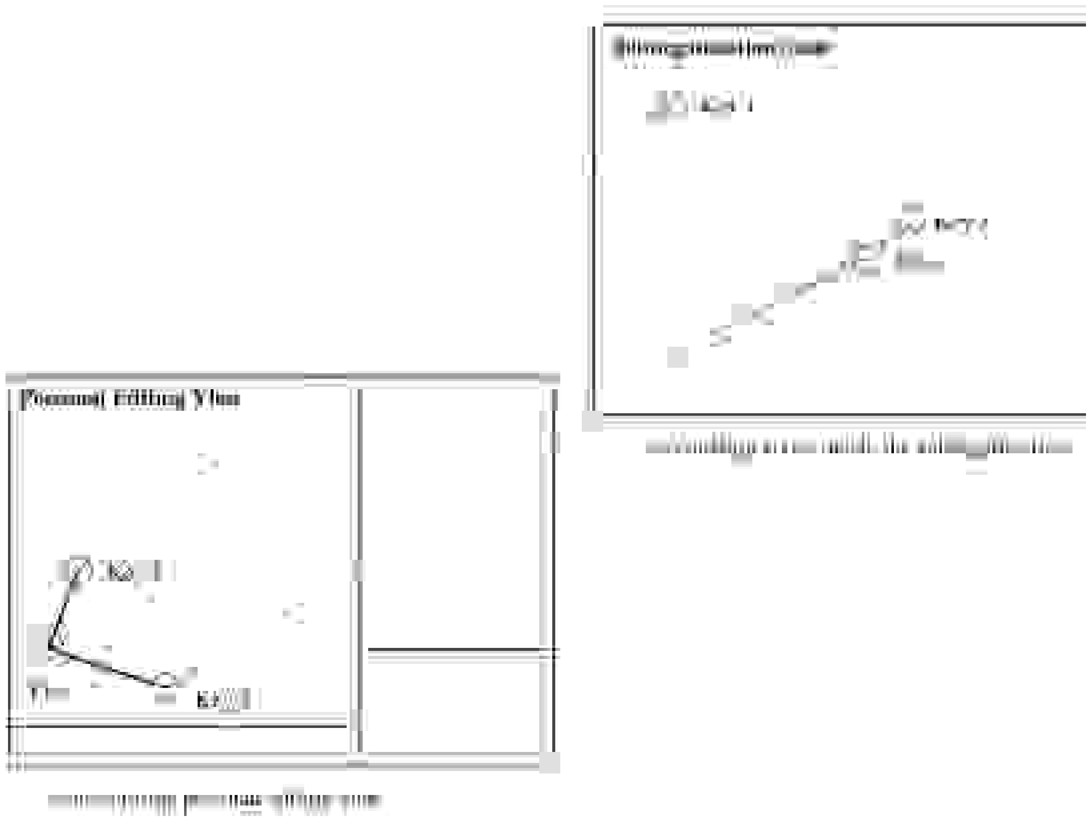


Figure 5: Relations between a personal keyword layout and the system keyword locations define self-organization.

A system that is controlled in such a way, tends to stabilize and come to rest over time. Thus, if there are no new documents added to the system, each document would assume a final resting position at some point and as such the information environment would not behave as desired by the artists. To not finally end up with a stable, unchanging system, additional rules will apply for the self-organisation process that tend to keep a constant reorganization process going.

- **Local "Vision"**

  Only those spring forces that have been calculated for a particular keyword will apply for which the related keywords are currently inside a certain surrounding region. This means from a physical point of view, that the force law used is discontinuous.

- **Constant Speed**

  The scalar velocity of the position of a document will be kept at a constant value. Only the direction of the movement is influenced by the spring forces acting on it. This will mean, that all document locations are changing at all times. Physically this corresponds to a high „temperature“ of the system.

- **Behaviour Cycle**

  To additionally make the total behaviour of the keyword more interesting, attraction rules (organisation) will slowly alternate with distraction rules (disorganisation). This cycle is like a virtual day and night and could be linked to actual ephemeris times of the place under discussion.

The Database

The central element for maintaining all project data and for handling the update and exchange of the data between the different software modules is a SQL-driven relational database system. We decided to use POSTGRES 95 for the project, a RDBMS which is in the public domain.

The database will be holding and maintaining the following data:

- **Personal Editor Data**

  The name of every editor participating in the project, along with his user name, password and contact information are maintained in the database.

- **Original Documents**

  Document formats can be text (HTML, ASCII), image (JPEG, GIF), movie (RealVideo,Quicktime) or audio (RealAudio, AIFF/C). During document creation, documents are choosen, digitized and uploaded to the database. Editors can add personal comment and annotations to the document.

- **Editor Specific Segments**

  To process text documents further, a segmentation process is possible. This allows a given longer text to be fragmented into smaller segments. Segments can be chunks of text from original which the editor has chosen to give a particular keyword. Every segment the editor creates, is entered into the database and keeps a reference to the original document and to the editor.

- **Rule Objects**

  The editor is able to place his created keywords onto a two-dimensional area, arranging keywords in spatial relation to other and hence creating a keyword relation map. Every placed keyword becomes a rule object which, in addition to a reference to the corresponding keyword and to the editor, stores the position on the editors map. Rule objects have also references to links to other keywords that have been drawn by the editor

- **Global Objects**

  All local rules defined on each editors relation map are combined and then fed into a self-organizing system which reorganizes the documents in the database spatially. These global objects continously update their global 2D position in the database through the self-organization engine. They are also accessible to the force field engine and contain the relevant meta data.

Theoretically there is no limit on the number of individual documents present in the system – the database is expandable, the visualization does not depend on the number of documents and the document processing is done asyncronous on the server. Practically there will be a limit posed by the server speed, available harddrive space and network speeds with which the clients connect. IO_Dencies is estimated to be fully functional with 5000 documents organizing in the system.

How to self-organize

Based on the two-dimensional keyword maps created by the editors, (see Figure 3) the self-organisation algorithm will create local behaviour rules for each keyword object. As a first approach the distances of the 5-10 closest neighbours to a specific keyword in each map will define a set of different spring forces for that object. Then, all objects are distributed randomly into a 2D- (or possibly 3D-) environment where their motion equations are solved by simple integration of the total acceleration over time.

To not finally end up with a stable, unchanging system, additional rules will apply for each object:

- **local "vision"**

  Only those spring forces will apply for which the related object is currently inside a certain surrounding region.

- **constant speed**

  The scalar velocity of an object will be kept at a constant value. Only the direction of the movement is influenced by the spring forces.

- **behaviour cycle**

  To additionally make the total behaviour of the keyword objects more interesting, attraction rules (organisation) will slowly alternate with distraction rules (disorganisation).

Visualization

Forcefield engine

The forcefield engine facilitates the visualization of the IO_Dencies “world” on the clients by providing supporting data on demand through the Internet. A virtual pane with x and y coordinates from 0.0 to 1.0 is populated with forces that represent the locations of the keywords (and related documents). The server aquires this information by querying the database in regular intervals. The position of a document in the IO_Dencies virtual environment - as determined by the self-organization engine - is used to position the force center in the pane in a one-to-one fashion. Meta-data associated with the document is mapped to one of the variables of a force point.

Each force has several attribute that control the look and feel of it. The varibles that control the forces are: type, strength, range and direction. The following table lists possible mappings from the meta data to the force variables:

Force variable	**Possible mapping sources**
Type	Age of keyword-object; text content
Strength	Number of selections by viewers
Range	Length of text; number of relations
Direction	Current angle of motion

The type variable controls the shape of the force field which is derived from a finite set of predefined fields (A,B,C, ... ,Z). The variable value can change from 0.0 to 1.0 and smoothly changes through all possible field types, mixing always two field types in a linear fashion (see Figure 6). The ordering of individual fields on the scale is done in such a way as to create “interesting” mixed-fields.

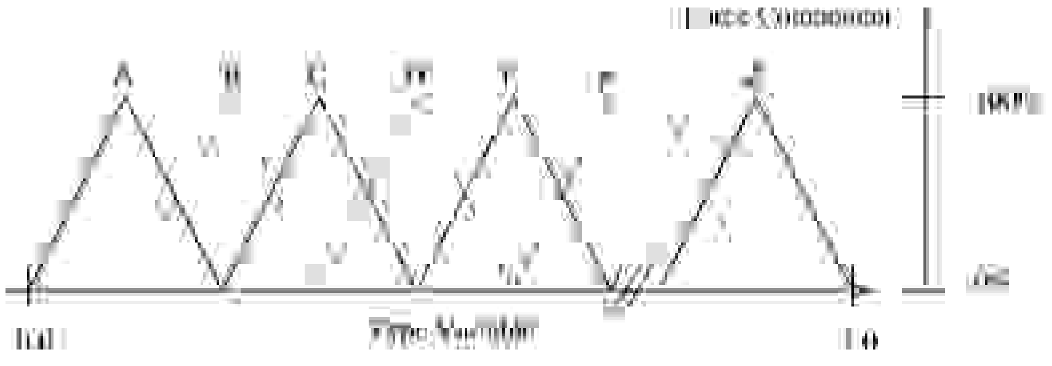


Figure 6: Linear mixing of field types is controlled by a single variable

Field Types

The fields that were designated in the previous chapter as A,B, ... Z are prototype fields. They are mathematically defined for each point around a center. The following graphics (see Figure 7 and Figure 8) show some of the field plots that are used in IO_Dencies[2]. Additional fields can be added easily, since the field shape is described by a single variable as described above. Note also, that these "pure" fields are seldom visible, since there is always a mixing between two field types active and the fields themselves mix when they overlap (i.e. when force centers are in close proximity).

[2] The following variables are used in the equations: r = radius of testpoint from center, dx = x-distance of testpoint from center, dy = y-distance of testpoint from center, PI = 3.1415927, fx = x-component of force, fy = y-component of force.

Figure 7: Sample field derived from the formula: *scale=1-r fx=scale*(1.0/0.82)*cos(10*dy+PI/2) fy=scale*(1.0/0.9)*

Figure 8: Sample field derived from the equations: *scale=1-r*
fx=scale(1.0/0.84)*cos(8.0*(dx*dx+dy*dy))*
fy=scale(1.0/0.06)*dy/(dx*dx+dy*dy+4)*

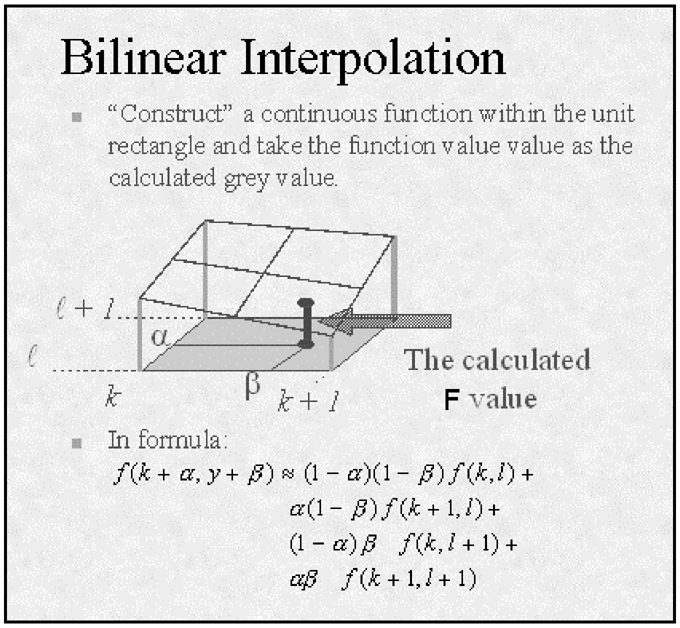
From field ...

These fields will not be visualized by vector diagrams as shown above but through streaming particles (pixels). This method makes the visual interface much more dynamic and interesting in nature, it is not as abstract to the viewer as a vector plot and allows for a greater adaptability to network conditions.

The visualizing client will first request a force field for a region of interest from the server. The server will then calculate the x and y components of the force field on a grid of fixed size (say 100 x 100) and return it to the client. During this process, all force centers (representing keyword-objects) will be considered. Since the fields are defined as functions, a wide range of region requests can be satisfied. If more than one force center is present in the viewing area (which is the normal case), each contributing force will be added together. From this calculation result two bitmaps of force components – one for the X direction and one for the Y direction. The bitmaps will be compressed using a Huffman encoder or PNG (Portable Network Graphics) image compression algorithm and transfered to the client. The client can then decompress the component bitmaps and use the force information to move points on the screen showing the field as a flow of particles. Additional information that will be transferred from the server to the client, will be the exact locations of all force centers in the viewing area plus their keywords and the last known locations and viewing areas of all other users currently using the system.

... to flow

The particle flow animation works on a time-step basis where each particle moves a distance dx and dy according to the force present at its original location at time t0. Its new location at time t+dt is then simply x+dx and y+dy and the process repeats. To make this process independent of computer speed, the client will first determine how many of those elementary particle updates can be achived (including drawing them to the screen) during the refresh intervall (1/25th of a second for a 25Hz refresh rate). The graphics routine will then proceed to animate exactly this number of particles. There are of course hard upper and lower limits on this automatic setting. The determination of the actual force involves a bilinear interpolation (see Figure 9) of the gridded force components. The quality of the moving pixel can be enhanced by using antialiasing. This feature is available through the OpenGL graphics library and is useful on platforms that have hardware accelerated 3D graphics capabilities. If a particle moves out of bounds or comes to close to a force center, it will be removed and should reappear on a random position to keep the total number of particles in view constant. Particles will also require a „camper“ timer that will remove particles that - due to the nature of the force field – do not move from a location (they „camp“).



Bilinear Interpolation

- "Construct" a continuous function within the unit rectangle and take the function value value as the calculated grey value.
- In formula:

$$\begin{aligned} f(k+\alpha, y+\beta) \approx\ & (1-\alpha)(1-\beta)\, f(k,l) + \\ & \alpha(1-\beta)\, f(k+1,l) + \\ & (1-\alpha)\beta \quad f(k,l+1) + \\ & \alpha\beta \quad f(k+1,l+1) \end{aligned}$$

Figure 9. Interpolation of gridded field data

Field changes

Since the field is based upon the dynamics of the documents (as encapsulated by changes of their locations and meta data) and user actions, it will change and the clients need to received new versions of the field frequently. Since both client-load, server-load and the networking conditions can change widely, a simple polling of information at a fixed rate is employed. If the transmission finishes faster than its allotted time, the client will wait until the next rate interval comes about. If the transmission takes longer that the rate interval, the next transmission request will follow immediately and the rate intervall will adjust to the conditions (become longer).

To smooth out drastic field changes that might occur in highly active regions (field centers move fast, field parameters change fast) a new field will not be applied instantaneously to the moving particles. The old field will be interpolated linearly over several (particle) time steps to the new field. A similar condition could occur also when the update interval is large due to slow connection speeds.

User Interaction

The user will have several ways of interaction with the field visualization:

- motion of the viewable area (left, right, up, down)

- zoom of the viewable area (in, out)
- selection of visible keywords

To avoid a lag period during which the client to wait for new field information after a motion or zoom-out event, the requested field area is choosen to be larger than required for a non-moving viewable area. Thus the motion and zoom speed depends on the available transfer speed and the server load. If the client cannot get updated field information fast enough, it has to slow the motion or zoom requested by the user – this allows for a smooth dynamic updating according to the actual connection and server conditions. During zoom-in events the grid resolution simply changes. For consistency, the zoom-in speed is the same as the zoom-out speed. Moving and zooming the viewable area in the field will also create a feedback to the field itself. This is handled on the server side, since the server has knowledge of the views location as well as its motion. Special force centers will be generated from the motion of the viewable area (trace). These „disturbing“ forces will only be stored on the server, have a limited lifetime and there will be a limited number of them as well.

At a sufficient zoom level, the keywords that are associated with the force centers become visible. They are clickable and should “spawn” an external viewer (web browser) to display the text. Thus the field visualizer is not responsible for displaying the actual information hidden behind the force centers and keywords.

Future Work

Future work will concentrate on two major areas: firstly, the implementation of the system in the context of Sao Paulo’s urban planning. This process has already started as prospective editors have been contacted and the software has been partially developed. The main task here is to make the system usable by everyone involved. This migh require design changes in the user interface and display technology to adjust for the local situation (networks, computers) and from user feedback. Secondly the IO_Dencies system will be expanded with new forms of clients that make use of 3D graphics and audio. The following extensions to IO_Dencies are considered:

- The 2D force field visualization using points will be extended by using lines, splines or 3D objects that are controlled by the force field. The goal is to show more clearly regions of complex dynamics and high activity.

- The relation of the content of information in time (not space) will be visualized as a 3D thread going through the IO_Dencies space. Alhough it is not possible to record the complete state of the system to allow a peek back into time, it is feasible to record a partial state of the system (keywords and their locations). This can be used as a navigational tool as well as an informational view on the system

- Audiovisual Tools will be developed that encode the activities of the users and the self-organizing system.

Conclusion

In summary, IO_Dencies is not a navigation interface for different chunks of information and differently designed databases, but is a place where different infrastructures, concepts, economies of time, behaviours, feelings, etc. are being confronted. The artists, Knowbotic Research, see IO_Dencies as a construction of an experiment to condense different fields of urban thinking in a networked environment. One could call it “heterogeneous places of the mind”. From this extended form of urbanity it should be elaborated if possibly bridges to the physical city could be built. The primary goal is, to test and experiment of how one can identify the developing structures of intensity, to then create spaces where those structures can grow. This happens in IO_Dencies - in an exemplary way - through the interaction with the selforganising system and the special visualisation techniques.