12

The Design of 3D Metaphors for Database Visualisation

J. Boyle [1] *and P.M.D. Gray* [2]
Robert Gordon University [1]*, Aberdeen University.* [2]

Schoolhill, Aberdeen AB9 1FR, UK.
tel: +44 (0)1224262000
fax: +44 (0)1224 262090
email: john@csd.abdn.ac.uk

Abstract

This paper discusses the designs behind Amaze, a graphical user interface to an object oriented database system. Amaze uses three dimensional graphics to visualise both query construction and result representation. The designs behind Amaze are introduced, and the visualisations of a prototype implementation are shown. It is felt that just as word processors have changed how people interact with document processing, so to could database visualisation alter how people interact with database systems.

Keywords

Database Visualisation, 3D Graphics, Interface Design

1 INTRODUCTION

Today user interface design for database systems is on the threshold of a paridigm shift. Designs are moving from a desktop menu based query retrieve style towards a means for exploration and visualisation of the database. The query process is thought of as being more than just a simple question and answer session. The focus is moving towards allowing users to interact with the data in a more intuitive way.

S. Spaccapietra et al. (eds.), *Visual Database Systems 3*
© Springer Science+Business Media Dordrecht 1995

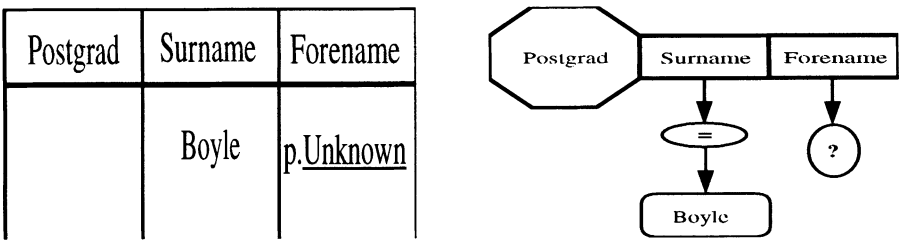
This paper outlines the designs and prototype implementation of Amaze. Amaze is a graphical user interface for an object oriented database. This GUI uses three dimensional graphics to aid the user in visualising both query construction and result representation.

It was felt by the developers that the standard desktop metaphor does not have the visualisation power required for displaying the quantity and diversity of abstract information that is needed for database visualisation. Extending the WIMP system, by embedding high performance animated three dimensional graphics, could give the enhanced expressive power whilst still keeping the familiarity and use of a menu driven environment.

2 BACKGROUND

Interfaces to Databases

Interfaces to database have developed over the last twenty years in a number of directions. However there is still no approach to a standard look and feel. QBE (Zloof 1992) was developed in 1975 for use on a character based terminal (vt100), and has since been adopted by a number of relational database manufactors as the basis of their interface designs (Access, dBase, FoxPro, Paradox, Superbase). About the same time as QBE was being developed Cupid (Macdonald & Stonebraker 1975} was proposed (a more visual approach to querying a database). These two interfaces offered a different high level approach for querying databases (Figure 1. QBE was the more powerful method, which later developed to allow updates and other data manipulation methods to be incorporated.



(a) QBE allows the user to enter the query by using a table as a template for the visual query

(b) CUPID allows the user to describe the query by using a simple diagram

Figure 1 Showing both the QBE and the CUPID representation for the query get the first names of any students who have the surname Boyle.

However QBE was designed before bitmap displays became prevalent and so the ideas behind its design naturally reflect this. With bitmap displays a new generation of user interfaces was born, these were able to exploit the new technology. Designs for interfaces to database systems soon made use of direct manipulation techniques e.g.Wong (1982), Fogg (1984).

Since then two families of interface to database have developed, those that use the *table* as

the basis of their interaction, and those that use the *schema* (Table 1). The design methodology used largely depends on the underlying data model. The relational model lends itself more easily to the QBE/Table based visualisation, which semantic data models are more easily visualised using a schema diagram approach.

Table 1 GUIs for Database systems grouped by design style

FORM BASED	*GRAPH BASED*
QBE (Zloof, 1992)	CQL (Kari et al, 1990)
Pasta 3 (Kuntz & Melchort ,1989)	GUIDE (Wong & Kuo, 1982)
CUPID (MacDonald & Stonebraker, 1975)	OdeView (Agrawal et al, 1990)
STBE (Ozsoyoglu et al, 1989)	LID (Fogg, 1984)
SBA (Zloof & deJong, 1985)	Good (Gemis et al, 1992)
G–Whiz (Heiler & Roseenthal, 1985)	Isis (Goldman et al, 1985)
ABE (Klug, 1981)	Hy+ (Consens & Mendelzon, 1993)
Vista (Sawyer et al, 1992)	SNAP (Bryce & Hull, 1985)
OBE (Whang, 1987)	Tioga (Stonebraker et al, 1994)
GQBE (Jacobs & Walczak, 1983)	QBD (Angelaccio et al, 1990)
GRADI (Kiem and Lum, 1992)	KiView (Motro et al, 1988)

The 1990s have seen an emergence of user interfaces that use three dimensional graphics as the center of the design. This approach breaks away from the bitmap displays and offers new opportunities (in much them same way as the transition from text–terminals to bitmap displays did).

Over the last few years, people have started looking at other designs for user interfaces to databases. These designs involve using different metaphors then those found in the desktop environment. A number of information visualisers have been developed e.g. Hemmje (1993), Robertson et al (1991), Robertson et al (1993), Fairchild (1993). Recently questions have begun arising about what is the best way to portray a database to a user, and what is the best way to show a query, and how can a user see what results they have obtained – this has lead to the idea of database visualisation. The ideas have been developed further, leading to the implementation of a number of 3D interfaces to databases e.g. Mariani et al (1992), Boyle et al (1993) , Benford et al (1994), Rapleyet al (1994)

3 THE DATABASE

The interface has been developed to the P/FDM system. P/FDM (Gray etal, 1992} is an object–oriented database system based on the functional data model (FDM) (Shipman, 1981} and implemented in Prolog.

The P/FDM system (Figure 2) allows a user to access complex data through a number of user interfaces. The user doesn't need to know the mechanism through which the data is obtained or how the data is stored. The user enters the query and the results are returned.

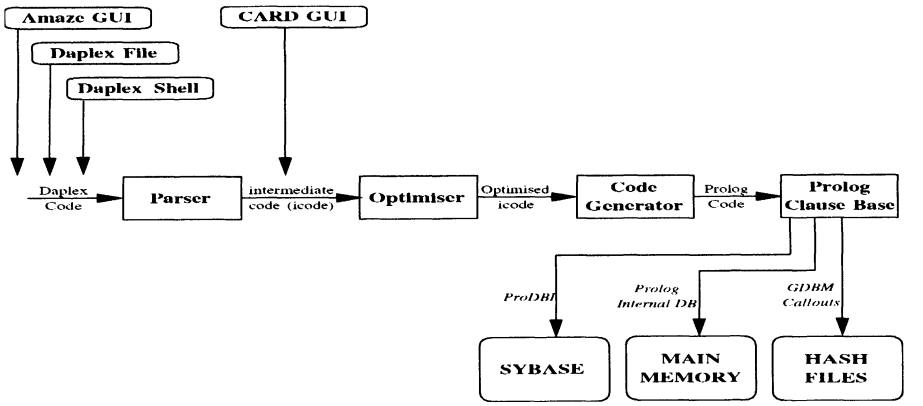


Figure 2 The Information Flow inside the P/FDM system. The user queries the database through one of the interfaces, the query is then compiled into Prolog with embedded primitive calls. The primitives access different storage schemas depending on which module the instance or function was defined.

A number of user interfaces have been developed, these vary both in complexity and ease of use:

Through the *Daplex shell* or *Daplex files* the user can write a query and then have it evaluated. At present the shell must be opened inside the Prolog environment. The user types in the query, and the system attempts to evaluate it. This offers the user a powerful mechanism through which they can query the database. Such a query language style interface is not easy to master and is difficult to use for the inexperienced. The results can only be viewed as ascii files.

The *Card* (Boyle et al, 1995) interface allows the user to graphically construct a query. The user fills in fields on *cards* to define the query, all *cards* matching other *cards* can then be viewed sequentially by the user.

The *Amaze* (Boyle et al 1993) interface allows the user to visually construct a query. This query is translated into Daplex code (actually a slightly extended version of Daplex) and then evaluated. This interface will be discussed in greater depth in the next section.

4 DESIGN OF AMAZE

The Amaze interface was developed to allow users access to the complex object-oriented database P/FDM. To do this, 3D graphics were used to visualise both the query construction and result representation processes. Both of these visualisations had to be designed and then implemented.

Query Construction

A visualisation of the conceptual schema was used as the basis of querying. The interface was designed so that the users would travel around the schema formulating the query as they moved. Each time the users arrived at an entity class they would formulate a subquery on that entity class. Each separate subquery the users formulated would be represented as a unique cube hanging beneath the relevant entity class (Figure 3).

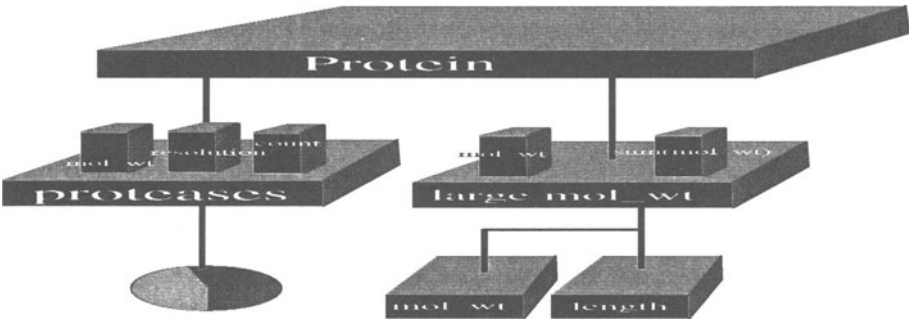


Figure 3 The design for the representation of a query on an entity class. The entity class *protein* has two subqueries attached to it: *proteases* and *large molwt*. Beneath each of the subquery box *large molwt* is the visual representation of the subquery: all instances inside the *molwt* box AND all instances inside the *length* box. Above each of the subquery boxes are those values that the users wish to know more about (either for output to file or to be used in another subquery).

Beneath each subquery would be a tree. The tree would represent the separate nodes that make up the total subquery. For example if the users wished to generate all the protein that have a high molecular weight and are of a certain length range, then the subquery *large molwt* could be generated.

The subquery would be made up of two nodes:

The molwt node. Would represent all proteins that have a molecular weight greater than 10kDa.

The length node Would represent all proteins that have a length greater than 200, but less than 10000.

The users may wish to use the results of the subquery to form the basis of another subquery (or may wish to output them to file). This would be a similar mechanism to projections in QBE (Zloof, 1975}, such projections could be represented graphically as objects on top of the relevant subquery.

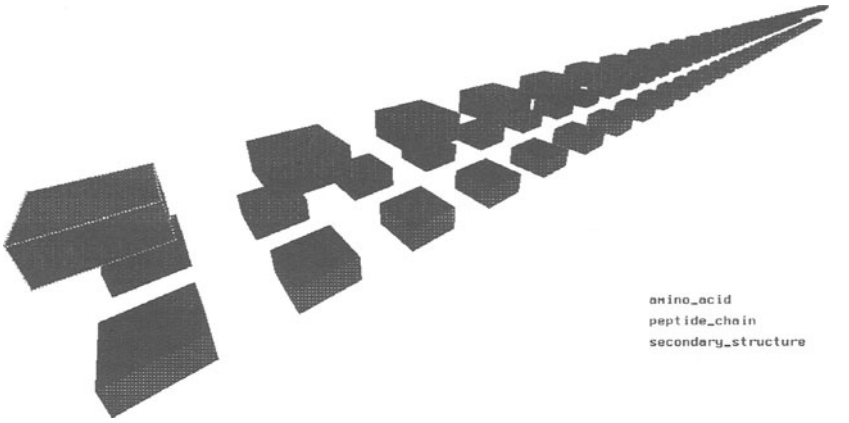


Figure 4.a: The *Maze,* each result instance is viewed as a unique cube, the cubes are clustered by entity class. A set of instances related to an instance from another class are clustered spacially It is possible to see at most two relations ahead

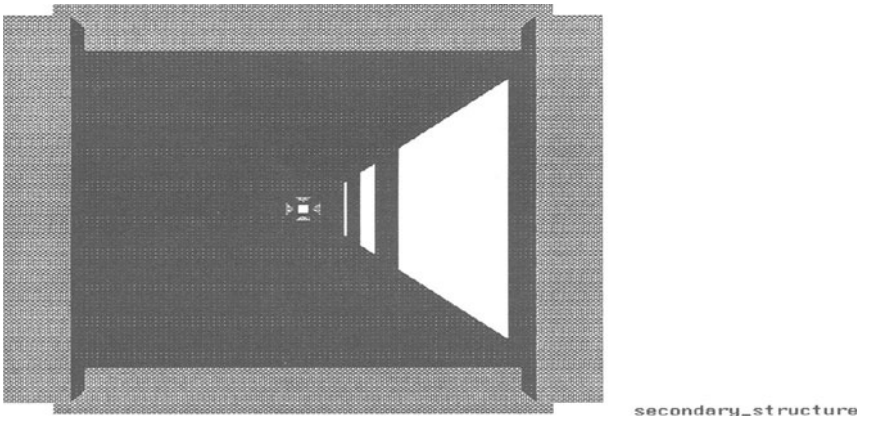


Figure 4.b: The *Walkthrough,* each set of results can be visualised as a corridor. The user can then walk down the corridor and can turn to see a related set of results

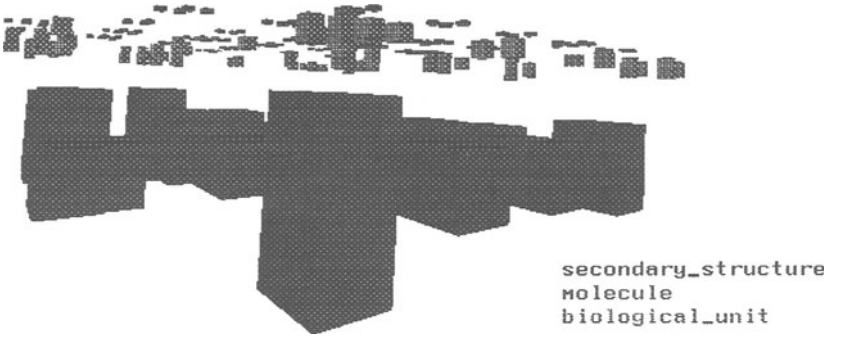


Figure 4.c: The Sea, the results are clustered by entity class, with results being further grouped into subsets to show relations between the different entity classes. The sea shows all the results – spread out in a continuous manner

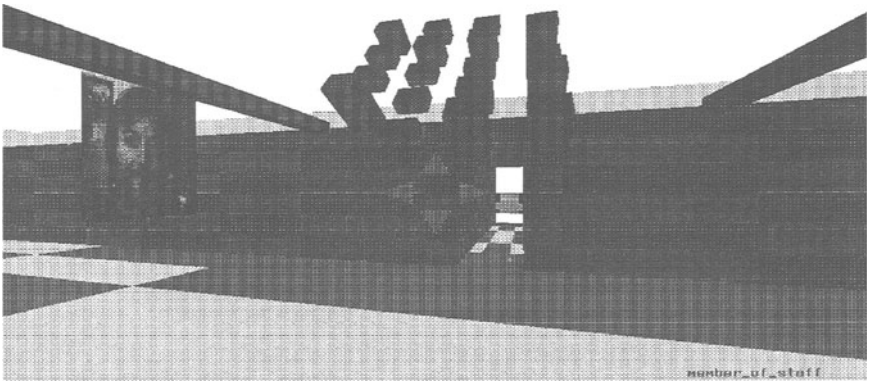


Figure 4.d: The Room, the results can be placed inside a room environment. The user can either move the results around or can move their own location. To find information about a relationship the user has to walk down the corridor to the next room. the room can also be used to show other information, in this case image data is included

Result representation

Once the query has been evaluated it was felt that a mechanism for being able to visualise the results would be important.

Each resulting instance would be returned and visualised – the results would be grouped by entity class and the relationships between sets of instances would be shown spatially. This would be a generic mechanism through which the users could explore the results to find out more about the quantity of results and how they are interrelated. The users should also have the power to be able to explicitly ask for more information (in a number of visual representations) about any of these results.

Four basic designs have been implemented:

The Maze. Results are clustered by entity class. A set of instances related to an instance from another class are clustered spatially. It is possible to see at most two relations ahead (as constricted by three dimensions), if the users want to see another related set of instances they choose an instance and the other information is faded out, and the next instance set is faded in. (Figure 4.a)

Walkthrough. Each set of result instances can be visualised as a corridor. The users can then walk down the corridor and can turn to see a related set of results.(Figure 4.b)

The Sea. This is a mechanism that allows all of the results to be visualised. The *sea* of results is an expanding result set, which the users view my moving over it – as the users move into the sea more results appear. The size of each instance is proportional to the number of instances that are related to it. (Figure 4.c)

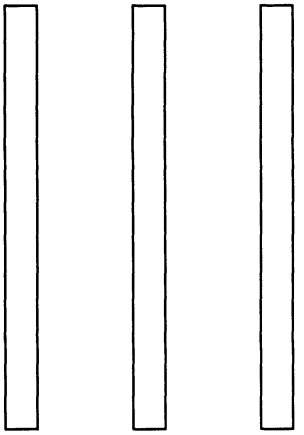
The Room. The users are placed inside a *room* with the results. The users can move their relative viewpoint in the room by selecting directional arrows that are rendered on the screen, If the users are interested in a related set of results then they can move down the corridor to the next room – placed in this room are the related set of results.(Figure 4.d)

The users can select which method they wish to see the results visualised with by selecting the corresponding button at the top of the result visualisation window. The users can change the method at any time during the querying session.

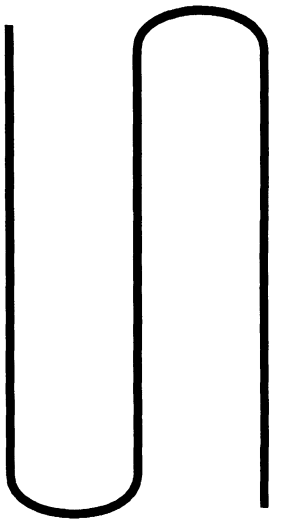
5 IMPLEMENTATION

Amaze has been implemented. While it is still being developed, it is a fully functional GUI to the underlying database. To embed the 3D graphics windows inside a menu driven system, a 'through the window' approach (as opposed to an immersion virtual reality system), with a 2D (mouse) not 6D (spaceball/bat) pointing device was used. To illustrate how Amaze can be used to answer a user's query, an example will be used.

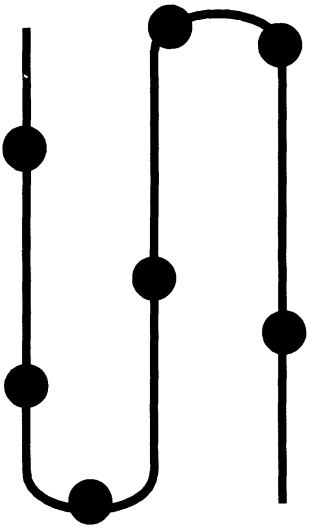
The example query will be to the protein database (Gray et al 1990), as this is an example of a more realistic use of the database. The schema contains a description of the modelled data .



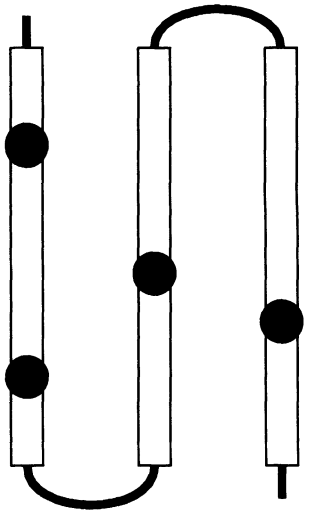
Getting all the strands in the database.
In Daplex:
for each s in sstructure
such that name(s)="S"



Getting all the chains that have strands in them. In Daplex:
for each s in sstructure
such that name(s)="S"
for each c in has_chain(s)



Getting all the tryptophon residues in the chain. In Daplex:
for each s in sstructure
such that name(s)="S"
for each c in has_chain(s)
for each r in res_by_name(c,"TRP")



Getting all the tryptophan residues that lie in the strand regions of the chains. In Daplex:
for each s in sstructure
such that name(s)="S"
for each c in has_chain(s)
for each r in res_by_name(c,"TRP")
such that pos(r) >= start(s)
and pos(r)=< end(s)

Figure 5 Showing a possible step by step mechanism for dividing the query into subqueries

Asking the question

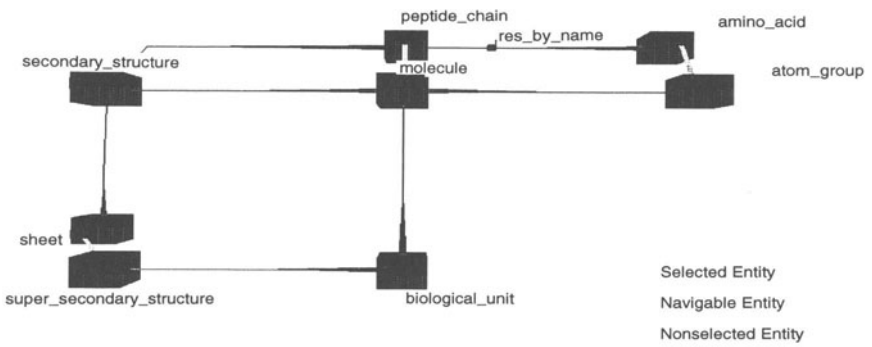


Figure 6 A subschema of the protein structure schema. A large number of the entity classes have been hidden.

If the users wish to examine all the strands of chains that contain Tryptophan (Figure 5)

The query involves navigating through three entity classes. Amaze supports the idea of subschemas – so the users doesn't have to display the whole schema if they are only interested in part of it. The subschema is defined in data files on the client side, it defines which entity classes and which scalar attributes should be shown.

Figure 6 shows a subschema of the main protein schema. The users can select the entity class *Secondary Structure* from the subschema and choose to query it. The users are presented with a dialogue box, from which they can choose to ask for all structures that have a *dssp_name* equal to "E" (extended strand) (Figure 7).

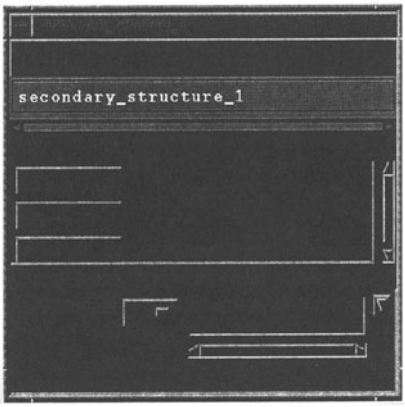


Figure 7. The user can enter the query on the entity class *secondary_structure* by filling in the dialogue box. The user has selected all the instances of *secondary_structure* that have the *dssp_name* E

The users also explicitly states that they are interested in where the strands start and stop. This is represented graphically above the subquery (Figure 8).

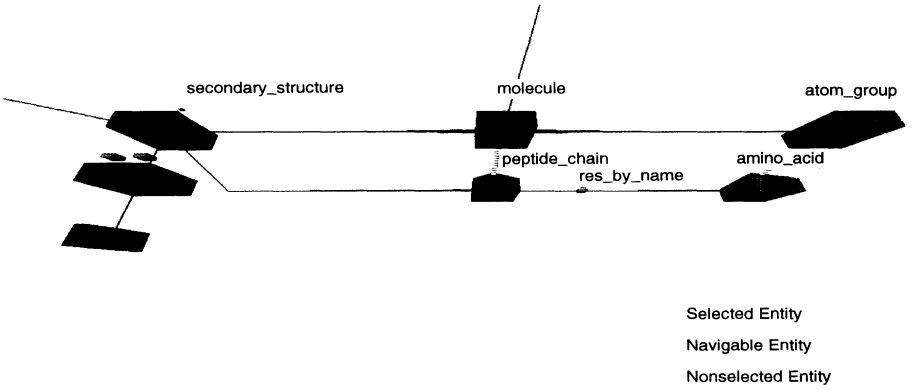


Figure 8 The subquery to *secondary_structure* is represented as a dangling cube. The users enter constraints which make up the subquery are shown as a graphical tree beneath the subquery cube (in this case all dssp_names that have a value "E". An values the users wish to export from the subquery are shown above the subquery cube – in this case the *start* and *end* values of the *secondary_structure* instances.

The users then navigate to the next entity class of interest, *peptide_chain*. selecting all of the instances of the class.

e.g.

for each s in secondary_structure such that dssp_name(s)="E"
for each c in has_chain(s)

Now the users are interested in all tryptophan amino acids. There is a two argument function between *peptide_chain* and *amino_acid*. This takes a name of an amino acid as its second argument (*res_by_name(Peptide_chain, String)*). The users can select any names for this value (if the second argument was an integer then the users could select a range of numbers).

The users then ask for all *amino_acids* that have a position inside a strand. The query (Figure 9) is shown graphically, and corresponds to the Daplex:

for each s in secondary_structure
such that dssp_name(s)="E"
for each c in has_chain(s)
for each r in res_by_name(c,"TRP")
such that pos(r) >= start(s) and pos(r) =< end(s)

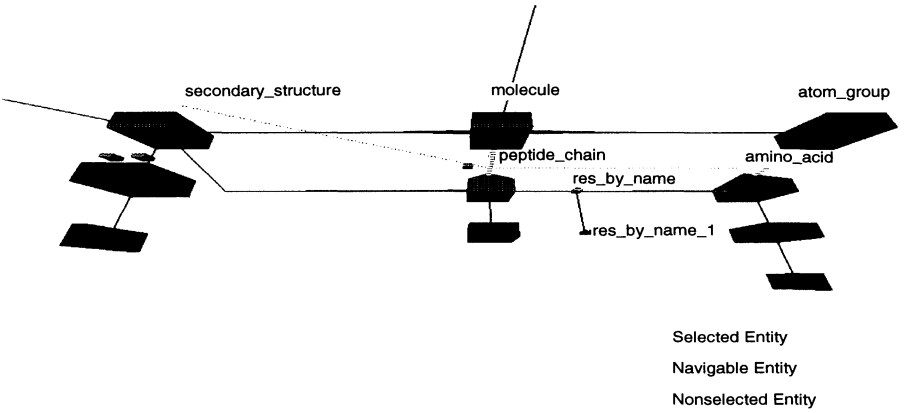


Figure 9 The whole visual representation of the query asking for 'all tryptophans that lie inside a strand'.

Viewing the results

Once the query has been formulated the users can send the query off to the database. Once the results have been returned, a visualisation of them is rendered to the screen (Figure 4). The users can then browse the results.

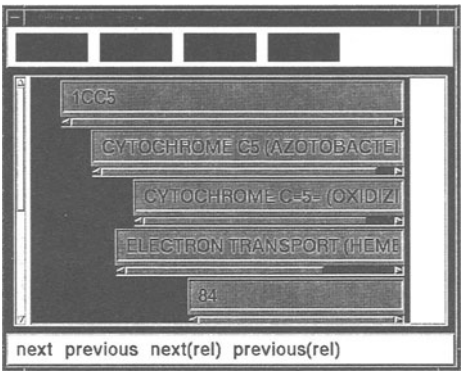
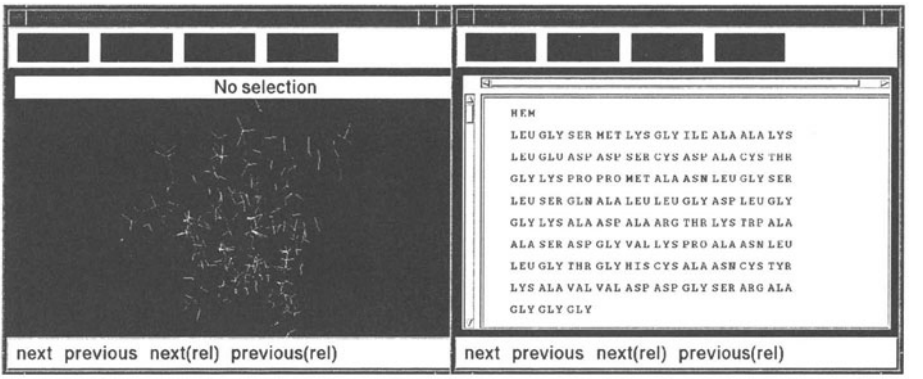


Figure 10 The browsing tool enables the users to visualise each of the token abstracts in the result maze. By using the *next* and *previous* buttons it is possible to browse the data. It is also possible to have the data visualised in alternative forms by using the top row *display action* buttons. At present the users can see the data as text (default), or for the protein data base as sequence or 3D molecule data, or for the personnel data base image data can be retrieved.}

The users can obtain more information about each result by using the browser tool (Figure 10). The browser tool has a dual purpose. It allows the users an alternative method for moving through the results (by use of the *next* and *previous* buttons) and also allows the users to visualise the results in a number of different methods (Figure 11).



The molecule displayer allows the users to see the 3D representation of the molecule they are interested in. The displayer shows the protein Cytochrome C

The sequence displayer shows the amino acid sequence of the protein. Again the displayer shows the protein Cytochrome C

Figure 11 Amaze has been customised for biological data by defining two display actions. As well as being able to see the scalar attributes of the data, the users can also get the sequence information and the 3D protein visualisation.

6 DISCUSSION

A number of extensions are planned for Amaze – to develop it from an advanced prototype into a fully functional and reliable interface.

Improve the querying power by:

Allow users to save and reload subqueries.

Allow quantifiers to be used (e.g. least, most)

Allow Daplex to be used as part of a subquery. So they can import complex Daplex (sub)queries and encorperate them visually into the main query.

Improve the result representation by:

Allow users to actually query from inside the *result maze*. So that users can extend the query to include other entity classes. Also a mechanism for sorting the results by a scalar value will be introduced.

Implement the other visualisations for the results.

Improve the usability by:

Distribute Amaze for evaluation/comments.

Visualise the actual query evaluation process. This has been partially done, however some mechanism to inform users approximatively how much longer the query will take to be evaluated is needed.

6 SUMMARY

The development of Amaze was driven primarily by the short comings of Card (a form based GUI for P/FDM). Amaze was designed to allow users to query large volumes of data, and to enable them to interact with the results returned from the query. The power of different visualisations was explored, both for query construction and result visualisation. Each of the two 3D visualisation have their own drawbacks, this are discussed below

6.1 The Schema

Large Schema Representation

One central question we have not attempted to provide an answer to is that of representing large schemas visually, and allowing users to navigation easily between distant entities. Amaze has a significant advantage over its two dimensional counterparts – due solely to the increased expressive power of three dimensions. More information can be portrayed to the users, and the whole schema representation can be manipulated by the users to focus in on the areas of interest.

Long distance navigation

One problem that will have to be dealt with is that of long distance navigation. Is it desirable for the users to see all the possible paths that can be navigated from one entity class to another. As the distance between the entity class grows so does the number of possible paths. Amaze can give the users all the possible paths between two entity classes – this is done by passing a meta data (Embury & Gray 1992) query to the database server which returned all the possible paths. However the most informative way to represent all the large number of possible paths has yet to be investigated.

Another approach is to have predefined paths – which the users can alter and save as they wish. These paths would be for queries the users are likely to ask at frequent intervals.

Locating the scalar attributes

With any schema representation, which shows the users the entity classes and their relationships, the users are always going to have difficulty locating the attributes they are interested in finding more information about. When the users becomes more familiar with the schema then this problem does not arise as much. A well designed schema, with well thought

out classifications, will help alleviate this problem. Visualisation of the attribute values (either automatically or under some user controlled event) is also worth some investigation – as it appears feasible in the three dimensional environment.

6.2 The Result Maze

Objectcentric approach

The main application database for Amaze is one which contains protein structure information. So it is desirable to study how biochemists interact with 3D packages, however as the interfaces had been developed with flexibility and generality in mind the interfaces should be usable by all – no matter the discipline or experience. This means that the interface should not be restricted to an object centric approach (where the users stay still and the objects move), and a egocentric approach (where the users 'flys' over the data) should be investigated for the result maze. It is felt that with a large volume of data the egocentric users movement may offer a more natural mechanism for data exploration – in the very least the users should be allowed to choose between modes of movement.

Smooth animation

One of the most important aspects of 3D interface design is that of smooth unconfusing animation. The result maze does smooth the transition of 'turning the corner' by fading one dimension out and then fading the next dimension in. However this does not give a smooth feel to the transition, and is a limitation of this visual metaphor.

Better result representation

Ideas and prototypes for other ways of representing the results has been suggested previously. Building these visual representations and their behaviour still takes some time, and judging their usability will be an important next step. The visual representations of the data are still relatively primitive, and more complex representations could offer better expressive power and usability.

7 THE FUTURE OF AMAZE

Further designs will be implemented and the drawbacks of each visualisation will be considered in the next design cycle. A successor to Amaze is presently being implemented, which will address the shortcomings in the previous version, this will be developed past the prototype stage into a more robust and versatile database visualisation tool.

References

[Agrawal et al., 1990] Agrawal, R., Gehani, N., & Srinivasan, J. (1990) Odeview: The graphical interface to ode, in *SIGMOD.*

[Angelaccio et al., 1990] Angelaccio, M., Catarci, T., & Santucci, G. (1990) *IEEE Transactions on Software Engineering 16*(10), 1150–1163.

[Benford & Mariani, 1994] Benford, S. & Mariani, J. (1994) Virtual environments for data sharing and visualisation populated information terrains in *Proceedings of Interfaces to Database Systems,* (P.Sawyer, Ed.). Springer Verlag.

[Boyle et al., 1993] Boyle, J., Fothergill, J., & Gray, P. (1993) Design of a 3d interface to a database in *Database Issues for Data Visualization,* (Lee, J. & Grinstein, G., Eds.) pp 173–183. Springer Verlag.

[Boyle et al., 1994] Boyle, J., Leishman, S., & P.Gray (1994) Wimps to 3d: design of a visual language for a database, in *Technical Report (submitted for publication).* Aberdeen University.

[Bryce & Hull, 1986] Bryce, D. & Hull, R. (1986) Snap: a graphics-based schema manager, in *Proc. Int'l Conf on Data Engineering.*

[Consens & Mendelzon, 1993] Consens, M. & Mendelzon, A. (1993) Hy+: A hygraph-based query and visualization system, in *Proceedings of the ACM-SIGMOD.*

[Cooper, 1992] (Cooper, R., Ed.) (1992) *The 1st International Workshop on Interfaces to Databases (IDS92).* Springer Verlag.

[Embury et al., 1992] Embury, S., Jiao, Z., & Gray, P. (1992) Using prolog to provide access to metadata in an object-oriented database, in *Proc. 1st Int'l Conf. on Practical Applications of Prolog.* Applied Workstations Ltd.

[Fairchild et al., 1993] Fairchild, K., Serra, L., Hern, N., Hai, L., & Leong, A. (1993) Dynamic fisheye information visualisations in *Virtual Reality Systems,* (R. Earnshaw, M. G. & H.Jones, Eds.). London Academic Press.

[Fogg, 1984] Fogg, D. (1984) Lessions from living in a database, in *SIGMOD.*

[Gemis et al., 1992] Gemis, M., Paredaens, J., & Thyssens, I. (1992) A visual database management interface based on good, In [Cooper, 1992] pp 155–175.

[Goldman et al., 1985] Goldman, K., Kanellakis, P., & Goldman, S. (1985) Isis: Interface for a semantic information system, in *SIGMOD.*

[Gray et al., 1992] Gray, P., Kulkarni, K. G., & Paton, N. W. (1992) *Object Oriented Databases - A Semantic Data Model approach,* Prentice Hall.

[Gray et al., 1990] Gray, P., Paton, N., & Fothergill, G. K. J. (1990) *Protein Engineering 3*, 235–243.

[Heiler & Roseenthal, 1985] Heiler, S. & Roseenthal, A. (1985) G-whiz, a visual interface for the functional model with recursion, in *VLDB*.

[Hemmje, 1993] Hemmje, M. (1993) Lyberworld: A 3d graphical user interface for fulltext retrieval in *Proceeding of the Workshop on Database Issues for Visualization*, (Grinstein, G., Ed.). Springer Verlag.

[Jacobs & Walczak, 1983] Jacobs, B. & Walczak, C. (1983) *IEEE Trans. on Software Engineering 9*(1), 40 57.

[Kari et al., 1990] Kari, S., Kangassalo, H., & Pösö, J. (1990) Conceptual query language - cql: A visual user interface to application databases in *Information Modelling and Knowledge Bases*, (Kangassalo, H., Ohsuga, S., & Jaakkola, H., Eds.) pp 608–623. IOS Press.

[Keim & Lum, 1992] Keim, D. & Lum, V. (1992) Gradi: A graphical database interface for a multimedia dbms, In [Cooper, 1992] pp 95 112.

[Klug, 1981] Klug, A. A. (1981) Abe: a query language for constructing aggregates by example, in *Proc. Int'l Workshop on statistical Database Management* pp 190–205.

[Kuntz & Melchort, 1989] Kuntz, M. & Melchort, R. (1989) Pasta-3's graphical query language: Direct manipulation, cooperative queries, full expressive power, in *VLDB* pp 97–105.

[MacDonald & Stonebraker, 1975] MacDonald, N. & Stonebraker, M. (1975) *ACM Transactions on Database Systems 5*, 127 131.

[Mariani & Lougher, 1992] Mariani, J. & Lougher, R. (1992) *Interacting with Computers 4*(2), 147 162.

[Motro et al., 1988] Motro, A., D'Atri, A., & Tarantino, L. (1988) Design of kiview: An object-oriented browser, in *Proc. 2nd Intl. Conf. Expert Database Systems*.

[Özsoyoğlu et al., 1989] Özsoyoğlu, G., Matos, V., & Özsoyoğlu, Z. (1989) *ACM Transactions on Database Systems 14*(4), 526 573.

[Rapley & Kennedy, 1994] Rapley, M. & Kennedy, J. (1994) Three dimensional interface for an object oriented database in *Proceedings of Interfaces to Database Systems*, (P.Sawyer, Ed.). Springer Verlag.

[Robertson et al., 1993] Robertson, G. G., Card, S. K., & Mackinlay, J. D. (1993) *Communications of the ACM 36*(4), 57–71.

[Robertson et al., 1991] Robertson, G. G., Mackinlay, J. D., & Card, S. K. (1991) Cone trees: Animated 3d visualizations of hierarchical information, in *Proceedings of ACM CHI'91 Conference on Human Factors in Computing Systems and Graphics Interface, ACM SIGCHI.* ACM-Press.

[Sawyer et al., 1992] Sawyer, P., Colebourne, A., Sommerville, I., & Mariani, J. (1992) Object-oriented database systems: A framework for user interface development, In [Cooper, 1992] pp 25–38.

[Shipman, 1981] Shipman, E. (1981) *ACM Trans. on Database Systems 6*, 140–173.

[Stonebraker et al., 1994] Stonebraker, M., Nathan, C., C.Paxson, Su, A., & Wu, J. (1994) Tioga:a database-oriented visualization tool, in *IEEE Visualization '93* pp 86–93.

[Whang, 1987] Whang, K. (1987) *ACM Trans. on Office Information Systems 5*(4), 393–427.

[Wong & Kuo, 1982] Wong, H. & Kuo, I. (1982) Guide: Graphical user interface for database exploration, in *VLDB* pp 22–32.

[Zloof, 1992] Zloof, M. (1992) Query by example, in *National Computer Conference* pp 431–437.

[Zloof & deJong, 1977] Zloof, M. & deJong, P. (1977) *Comm. ACM 20*(6), 385–396.

BIOGRAPHY

Peter Gray is a Professor in Computing Science at Aberdeen University, and is a credited author and researcher in the field of Database Management Systems.

John Boyle completed his graduate studies under the supervision of Peter Gray in 1994, it was during this period of study that most of the work discussed in this paper was carried out. He then worked briefly at EMBL in Heidelberg, and has now taken up a full-time lecturing position at Robert Gordon University in Aberdeen.