University of Glasgow

Department of Computing Science
Lilybank Gardens
Glasgow G12 8QQ

University of St Andrews

Department of Computational Science
North Haugh
St Andrews KY16 9SS

An Implementation of Multiple Inheritance in a Persistent Environment

Persistent Programming
Research Report 49
November 1987

An Implementation of Multiple Inheritance in a Persistent Environment

P.J.Benson, E.B.D'Souza, I.S.Rennie, S.J.Waddell

Department of Computational Science
University of St. Andrews
North Haugh
St.Andrews
KY16 9SS

Introduction

An Image Data Processing System [2] (IDPS) is documented as an object-oriented pictorial database system comprising an interactive graphical toolkit and an hierarchical database model incorporating multiple inheritance.

The database hierarchy is based on the notion of a taxonomy of superobjects and subobjects and composition inheritance links within the actual pictorial representation of an object. A query language permits retrieval of objects by matching textual attributes in a given query with the values of database objects.

Using the atomic graphical objects line, circle and box, arbitrarily complex pictures maybe built up to define a object's graphical representation. This method of specifying objects allows for graphical inheritance through inclusion which may be recursive.

IDPS is a Senior Honours (1987) project implemented in PS-algol [7] at the University of St. Andrews as a course requirement for the B.Sc. degree in Computational Science.

1. Historical Perspective

Sutherland's "SKETCHPAD" [18] provided the inspiration for our system. Advances in human-computer interfaces have provided us with conceptually simpler and more flexible graphical development systems and toolkits. Improved workstation technology offers a wide range of facilities to the programmer and hence any implementation of the IDPS should be able to support a comprehensive set of picture manipulating functions to provide that higher degree of flexibility.

Hierarchical data models [3,9,13] provide a natural mechanism whereby subclasses of objects inherit all the attributes of their superclasses. The Smalltalk-80 implementation [10] offers only single-valued attribute inheritance. Subsequent proposals for multiple inheritance are discussed in [3]. The semantics for this latter model and a means of inheritance type-checking are discussed in the implementations of Galileo [1] and Amber [6]. Many existing hierarchical systems such as Trellis [15] successfully incorporate a particular solution to the problem.

Traditional database management methods are not ideal for manipulating pictorial data as

1. such data is continuous and two dimensional,
2. the result of a database query is alphanumeric rather than graphical,
3. different operations are required to deal with pictorial data.

Consequently several difficulties arise when attempting to formalise the combination of these two models of data. Particular techniques are required to provide graphics functions. Symbol hierarchies, modelling transformations, display procedures and symbol operations are essential. The user interface requires a user model, a command language, on-line help facilities, quick response times to all operations, systematic menu design and facilities for feedback and screen output and layout [10,11,16].

2. IDPS: Design, Objectives & Problems

In IDPS, the traditional notion and definition of classes [8] as super- and subclasses of each other is deviated from. Instead, database records (*objects*) reside in an environment which exhibits *object/value* inheritance rather than attribute inheritance.

We have extended the hierarchical database model to incorporate a simple but powerful notion of multiple inheritance. Records of information stored in the database table are treated as objects, operations upon which are as follows:

1. creation of new objects,
2. deletion of objects.
3. updating of objects, e.g. addition of further detail to objects: extension of an object's attibute list or addition of verisimilitude to graphical representations,
4. identification of a relational formalism; through creation or editing of an object in the hierarchy a specification of parent and child relationships between objects is possible,

5. retrieval of objects by named or predicate search.

In order that the inheritance mechanism operates efficiently, all database `OBJECTs` have

1. a unique identifying name,
2. an optional short textual description,
3. links to its superobjects,
4. links to its subobjects (**not** user definable when creating a new subobject; subobject links only exist as superobject-object links when creating a *new object*),
5. a pictorial representation (the object's "scene"),
6. links to all of its instances,
7. a list of attribute name-value pairs, e.g. `(size,medium)`. IDPS treats all names and values as having type **`string`**.

The design specifications of the system are four-fold, namely

1. *database model* (requirements analysis); a general purpose pictorial database system with inheritance mechanisms. Pictorial data is generated interactively, and an abstract data representation of the pictorial information is stored together with textual information on each object.
2. *data needs* (information and object oriented structure); the definition of a database object is specified using the necessary intra- and interobject relationships.
3. *processing needs* (object structure orientation); all access operations on the database records, modes of operation and frequency of use, and graphics processing requirements are considered as internal to the schema representation. This governs the types of functions possible.
4. *storage level representation* of data in the given system; the IDPS design is concerned with superobject-subobject relationships. The actual storage level design of the data is dependent on the abstract structure for storing the pictorial representation of each object. This is based on the needs of the graphics model; simple line drawings based on atomic component manipulation are sufficient to build complex pictures [11,13]. The interface mechanism of the database schema is

used to incorporate existing scenes in the current representation. As a large number of operations are permissible upon such attributes it is necessary to strictly control the use of mouse buttons, their combinations and usage semantics. To ease the use of IDPS as little interaction is made via the keyboard as possible (textual key entries, object descriptions, etc.) while all subsystem menu calls are WIMP and menu oriented.

The conceptual model encompasses

1. *view modelling* which generates an abstract representation of the database using the specification of the requirements analysis,
2. *view integration* which integrates all possible user views into a global view of the database, i.e. the ubiquitous `UNIVERSE` object,
3. *view restructuring* which maps an integrated view of the logical structures onto the target system.

Automatic graphical inheritance [9,10] is meaningless in our implementation. It is the user's perogative to make a logical application dependent decision which defines the appearance of the object's scene; certain basic superobjects may already exhibit suitable pictorial characteristics and may therefore be included into the current scene. However, the following example (Fig.2.1) clearly demonstrates how this cannot be achieved with automatic graphical inheritance.

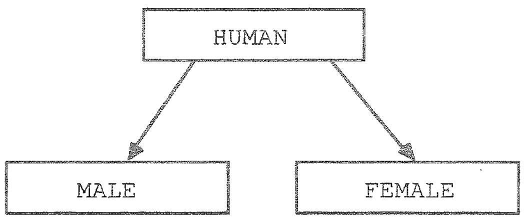


Figure 2.1

If one were to draw a skeleton for the `HUMAN`'s scene, then it should be anatomically sexed. Automatic graphical inheritance would give the `MALE` a `FEMALE` skeleton, or vice-versa (whichever is not specified). Even if the `MALE` were suitably clothed, a

closer examination (by parsing the `MALE`s inherited scenes) would reveal an inconsistency in the database inheritance mechanism. This consistency is at the root of semantic interpretation of the user model in the hierarchy. For this reason only very basic scenes need be created for each object. At any instant in time, the creation of an object defines a new subobject leaf node; this object may later become a superobject by virtue of another creation. Because of this, tangled multiple inheritance is not possible. For example in Fig.2.2 the superobject-subobject link from `D` back to `B` is not possible.

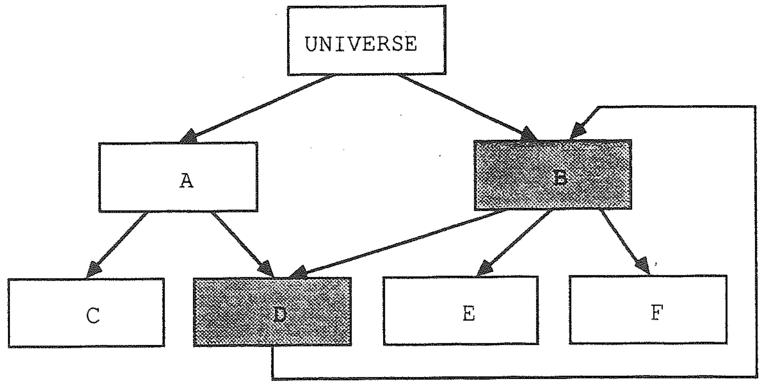


Figure 2.2

If `B` represented a barn door and `D` a barn, this example would tell us that a barn door is

1. a component of a barn, is a component of a barn door, is a component of a barn, etc. i.e. a recursive definition,
2. the attributes of the barn are (in part) derived from a barn door, but also the attributes of a barn door include those of a barn itself.

Such definitions of subobject inheritance are always recursive. The meaning of such definitions is not clear. For this reason they are not permitted.

Graphical inheritance is achieved by the user specifying scenes to be included in a object's `SCENE`. This mechanism allows a created scene to be included within itself to produce a conceptually infinitely regressive scene which is recursively defined with scale tending to zero, and positional value relative to the regression in the picture. The aim is for the user to utilise "skeletal" scenes in the hierarchy when creating a new scene. More well-defined subobjects whose scenes are inherited from their "parents" are fleshed out using the graphical editor until it meets the object's specification.

Declaration of attributes *local* to a object is possible only in a restricted sense. Any such name-value pair will be automatically inherited by a subobject specification. Local attributes of the same name as one inherited further up the graph (as in Smalltalk's single-value inheritance) are not permitted.

The abstraction techniques of *generalisation* and *aggregation* [17] are used to define the requirements of the database. Generalisation can be considered to suppress the differences between objects in a category, e.g. the domain of objects {elephant, dog, cat} and the generic object ANIMAL. Aggregation suppresses the component names and can be thought of as a relationship between objects, e.g. a person reserves a room in a hotel (relationship) and the corresponding aggregate object could be RESERVATION. Repetition of these produces an hierarchy of objects.

3. Database Model Implementation

In IDPS the generalisation is taxonomic and the aggregation can be considered as an "is-composed-of" semantic edge for the pictorial representation of an object. The conceptual model includes the generalisation and aggregation hierarchies for all concepts supporting the application view. Thus modularity is introduced. A necessary distinction is made between object and graphical attribute structuring; both views are concrete abstractions over the unified model with which the user is concerned. The object inheritance graph is acyclic (a tree) as recursive object definitions are not possible. Cycles are permitted when specifying graphical attributes of objects, i.e. the scene. Consequently the pictorial hierarchy only exists to provide an optional inheritance link for the scene attribute of an object.

Links in the database model are hence defined in two forms as

1. superobject-subobject links used for general database operations:

   <universe> ::= ø | <object>$^+$

   <object> ::= ø | <object>$^+$

   A newly initialised database has only one object, the UNIVERSE. Subsequently new objects are *all subobjects of the UNIVERSE*. In this sense, UNIVERSE is in fact a unique superclass representing the whole database.
2. composition links; each object has a pictorial representation (attribute of SCENE):

   <scene> ::= <scene>$^+$ | <atom>$^+$

   <atom> ::= <line>$^+$ | <circle>$^+$ | <box>$^+$

   i.e. SCENEs may be composed of other SCENEs or atomic objects.

These composition links are utilised in the graphical operations of the system. The scene of a object contains lists of constituent scenes (with scale and location information) and its own atomic objects. Fig.3.1 shows an example of this arrangement where TREE and ROAD denote individual collections of LINEs, CIRCLEs and BOXes which are linked to the object STREET. The object HOUSE has been graphically inherited into the new scene three times. This does not mean that STREET has three instances of the object inherited,

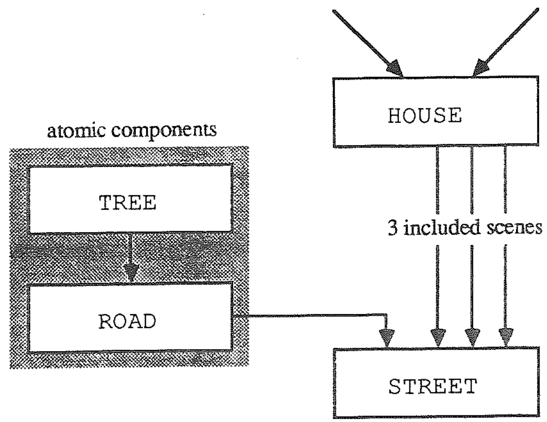


Figure 3.1

but only three instances of the *scene*. HOUSE has two superobjects in the hierarchy.

Subobjects in the database achieve automatic inheritance of single valued attributes through the generalisation hierarchy as in Fig.3.2. An inheritable attribute is one that applies to any instance (subobject only) by virtue of its membership in the superobject.

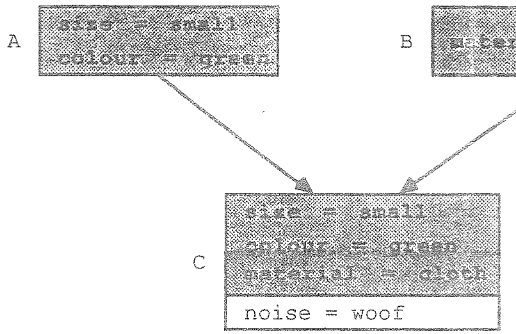


Figure 3.2

A subobject inherits all the attributes of its superobjects (multiple inheritance as above) but not their SCENE representations. The last attribute of object C happens to have been defined locally so now any object which has specified C as its superobject will inherit all four attributes. An object may have it's attribute values edited or indeed extended as part of the database maintenance process. Attribute value changes to objects in the universe will filter through to all inheriting subobjects.

Changing the representation of LEG in Fig.3.3 accordingly alters the pictorial representation of DOG and HUMAN (by composition rules). This is given that DOG and HUMAN have the same BODY and LEG. The alternative and better approach would be to create DOG.LEG and DOG.BODY, HUMAN.LEG and HUMAN.BODY as new objects and then redefine the composition of DOG and HUMAN. This example shows the care which must be taken in generalising the attributes of a object as all its subobjects will inherit these values. Superobject-subobject links between the highlighted objects can be considered a reasonable proposition, but this would mean that now BODY, LEG, ARM and MAMMALS are superobjects of HUMANS and DOGS. If this point were an absolute necessity, then the above description of the application model is incorrect - the user

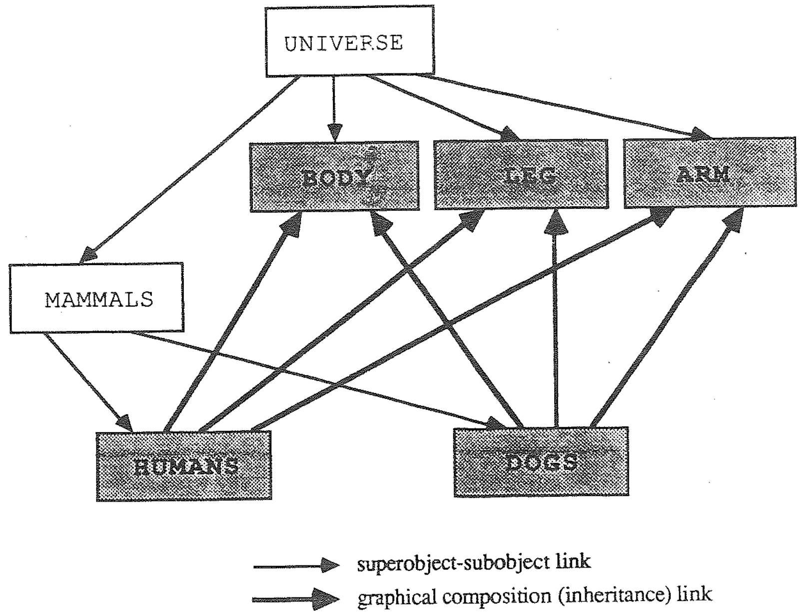


Figure 3.3

should have created a superobject BONE.STRUCTURES for example, specifying BODY, LEG and ARM as having this superobject then specifying the all new superobject relationships for objects from here. It is possible that the above hierarchy gives a more accurate description of what the user is attempting to model, and merely the graphical attributes of the three superobjects are required to complete the definition.

Duplication of what the user considers to be a submodel structure, i.e. part of the hierarchy, or creation of new objects can be used to overcome this problem. A data dictionary which may be manipulated is included within the conceptual design.

Scenes derived from the atomic LINEs, CIRCLEs and BOXes may be arbitrarily complex attributes of an object and are stored using the following abstract data representations:

```
OBJECT          (string object.name, object.description;
                 pntr super.objects, sub.objects, scene,
                      instances, attributes)

LIST            (pntr element, next)

ATTRIBUTE       (string attribute.name, attribute.value)

CONSTITUENT     (pntr object;
                 real locx, locy, scale.c)

SCENE           (pntr scenes, circles, lines, boxes)

LINE            (real x1.l, y1.l, x2.l, y2.l)

CIRCLE          (real x1.c, y1.c, radius.x.c, radius.y.c;
                 int rotate.c)

BOX             (real x1.b, y1.b, x2.b, y2.b)
```

4. Conventions & Constraints

Four windows (see Fig.4.1) are used throughout interaction, each defined as an abstract data type with capabilities for input and output of text, clearing and closing operations so that any function using them may treat the display in a uniform and selective manner.

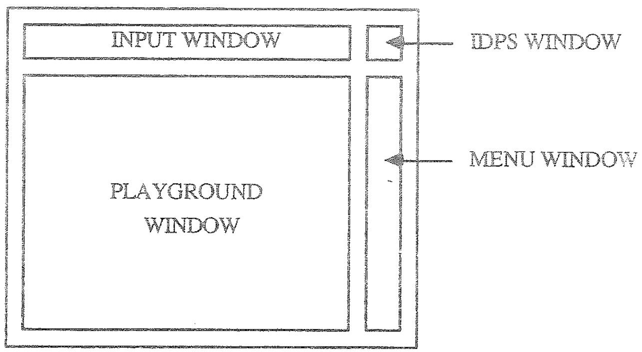


Figure 4.1

Messages to be acted upon always appear in the input window whilst displays of various forms are shown in the main playground area. All options menus appear in the rightmost window leaving the introductory system menu displayed in the playground.

The special case of the UNIVERSE object has neither a scene nor attributes; it is impossible to specify a general and *useful* attribute for this object which can still apply to subobjects.

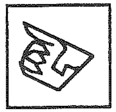
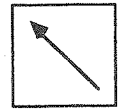
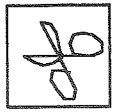
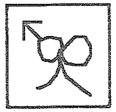
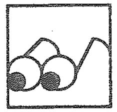
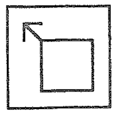
ICON	MEANING
	menu option selection
	graphics pencil
	cut out (delete) atom or scene within a scene
	tie (inherit) two scenes together
	view a scene
	move or select a portion of the scene

Figure 4.2

Six icon images shown in Fig.4.2 are used at selected points during run-time in place of the standard mouse pointer. Each indicates what the system is doing or is able to do at that point. For a user, it is far clearer what is happening at run-time by using this method. Icons are easier to understand than numerous textual messages and attention is localised at all times on one aspect of the interface.

Mouse control of system functions revolves around a simple rule-set defined as follows:

1 : select or start a function,
2 : abort, returning to the start of the current function,
3 : terminate with results and return to the previous function.

For functions whose nature demands a more complicated set of button commands, combinations of the above three are used in a consistent manner.

5. Applications & Use

The IDPS menu hierarchy shown in Fig.5.1a indicates the buildup of the system from ideas to specifications to menu entries to program (function) module development and finally implementation of tested code as live menu options.

Those functions which when selected cause visual or database updates have a secondary menu function (Fig.5.1b) which traps any accidental menu selections. At all times the OPTIONS and EDIT menus, when called, remain displayed in the rightmost window with the current option highlighted. This allows for an entirely visual option selection and verification process, keeping the user's attention focussed where it is needed.

On-line help is available at every point of keyboard data entry and also as a main menu function; each IDPS function has a screenful of information available for consultation with guides to related functions. It is important that the contents of the database are able to be catalogued in some meaningful and unambiguous form. To this end both a simple textual listing of object names and a full graphical browser are available (see also [4]). The browser, as Fig.5.2 shows, details the selected object and its respective super- and subobjects. Where there are too many super- or subobjects to be displayed in one go, a cyclic horizontal scrolling list of objects is presented. The database is *parsed* by repeatedly selecting a super- or subobject to be the new central

object (located in the centre of the display); any selected object may be "exploded" to reveal both its graphical representation and inherited attribute list.

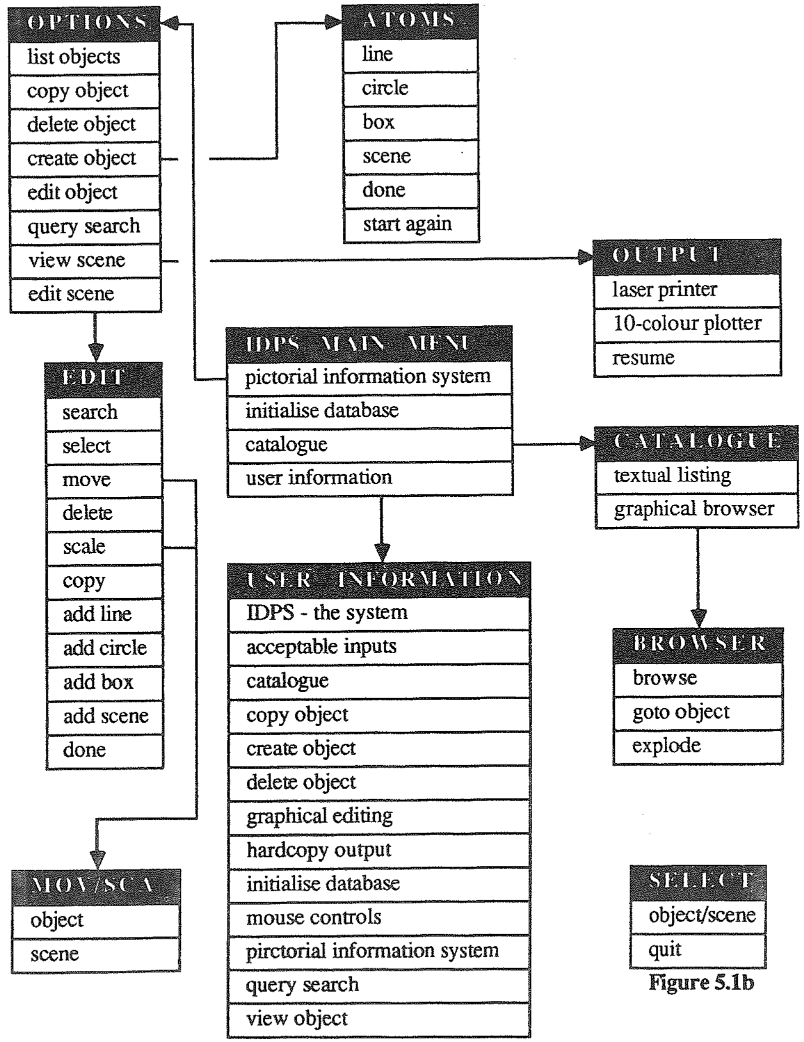


Figure 5.1b

Figure 5.1a

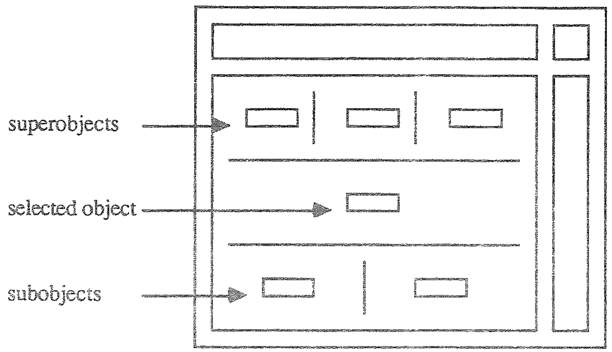


Figure 5.2

Several application areas for an IDPS were considered and these ideas have shaped the final piece of software. Of these the following are most significant: a graphical toolkit (albeit very simple) with inheritance, electronic schematic circuit modelling (transistors, capacitors, etc. may be created as resource objects to be included in a large diagram), "architectural" design, chemical modelling, cartographic modelling, biological classification, overhead slide preparation (through the hardcopy function), and database tutorials.

6. Conclusions

We have described an implementation of multiple inheritance using a model comprising two inheritance graphs. Object relationships must be kept distinct from the possibility of recursive graphical attributes.

Attributes are both textual strings and line drawings. Repetition of the database model functions upon objects allows refinement of these attributes.

Iconic cursor images used during interaction with IDPS show system status at all times. This is preferred to additional textual message displays and windows. As keyboard data entry is minimised, continuous on-line help is available (also at every new

level of menu selection).

Examination of database contents is by use of textual catalogues or more explicitly through the graphical browser; in this, a map of the precise super- and subobject relationships can be viewed in a logical manner.

Early evidence suggested that for a system as large as IDPS modular development of program source is necessary. *Static* binding of modules was chosen rather than *dynamic* (closures defined at run-time) because of the large number of permissible functions available at each stage and speed of module execution (encompassing link times, database pattern matching and closure definition). This, however, means there exists an update precedence for program modules in the database during development time. At run-time the benefits are only marginal in terms of speed but the chosen design principle holds; for smaller systems dynamic binding is preferred. Without such techniques, development of the project would have been very difficult.

One further benefit of such a scheme comes when type-checking module signatures. With dynamic closures, only at run-time can updates to modules' parameter lists be tested by an activation of the function. Should a type error arise data may be lost or indeed corrupted. By statically binding modules together (through the module update precedence list) this is overcome at the point where the closures are defined.

A slide presentation of the project to colleagues in the department included a demonstration of the possibilities of recursion in graphical attributes. A view of a house may be presented and its contents browsed by parsing the inheritance graphs. A journey through a window into a bedroom to look at a poster on a wall is possible. The picture shows our solar system which we can examine to discover the Earth, and through clouds to reveal the British Isles. Selecting this scene shows a number of cities, one of which details a map of the area. Homing in on a street scene we are confronted with several houses, the rightmost of which we recognise as out starting point. Many paths may exist in from one scene.

For more serious applications, IDPS would clearly need a far more extensive range of tools in the graphical toolkit. Extending the system for such a purpose as this is not a problem. The persistent database is updated to include a new procedure and respective menus are extended to offer the new function. The dynamic capabilities of

IDPS to match changing requirements of users is clear evidence of how important module persistence is.

References

[1] ALBANO, A., CARDELLI, L., ORSINI, R. Galileo: a Strongly Typed, Interactive, Conceptual Language. *ACM Transactions on Database Systems*, June 1985

[2] BENSON, P.J., D'SOUZA, E.B., RENNIE, I.S. & WADDELL, S.J. An Image Data Processing System. *University of St. Andrews, Department of Computational Science, B.Sc.(Hons) Project Report*, 1987

[3] BORNING, A.H. & INGALLS, D.H.H. Multiple Inheritance in Smalltalk-80. *Proceedings at the National Conference on Artificial Intelligence, Pittsburgh, PA*, 1982

[4] BROWN, A.L. & DEARLE, A. Safe Browsing in a Strongly Typed Persistent Environment. *PPRR-33-87 University of Glasgow & University of St.Andrews*, 1987 (to appear in *Computer Journal*)

[5] CARDELLI, L. A Semantics of Multiple Inheritance. *Semantics of Data Types, Lecture Notes in Computer Science 173*, Springer-Verlag, 1984

[6] CARDELLI, L. Amber. *AT&T Bell Labs Technical Memorandum, 11271-840924-10TM*, 1984

[7] MORRISON, R. & ATKINSON, M. PS-algol Reference Manual, 4th edition. *PPRR-12-87 University of Glasgow & University of St.Andrews*, 1986

[8] DATE, C.J. An Introduction to Databases, 3rd edition. *Addison Wesley*, 1981

[9] FOLEY, J.D. & Van DAM, A. Fundamentals of Interactive Computer Graphics. *Addison Wesley*, 1984

[10] GOLDBERG, A. & ROBSON, D. Smalltalk-80: The Language and its Implementation. *Addison Wesley*, 1983

[11] HANGEN, Ø. & SKIFJELD, K. Class Graphics - An Object Oriented Approach to Effective Computer Graphics. *Mach-S Project Group, Norwegian Computing Center*, 1983

[12] HARRINGTON, S. Computer Graphics: a Programming Approach. *McGraw-Hill*, 1985

[13] RICH, E. Artificial Intelligence. *McGraw-Hill*, 1985

[14] RODGERS, D.F. Procedural Elements for Computer Graphics. *McGraw-Hill*, 1985

[15] SCHAFFERT, C., COOPER, T. & WILPOLT, C. Trellis Object Based Environment. *DEC TR-372, Digital Eastern Research Laboratory*, 1985

[16] SHEIL, B. Power Tools For Programmers, *Datamation*, February 1983

[17] SMITH, J.M., & SMITH, D.C.P. Database Abstractions - Aggregation & Generalisation. *ACM Transactions on Database Systems*, vol.2(2), June 1977

[18] SUTHERLAND, I. SKETCHPAD: A Man Machine Graphical Communications System. *MIT Lincoln Laboratory Technical Report 296*, May 1965

Bibliography

Copies of documents in this list may be obtained by writing to:

The Secretary,
Persistent Programming Research Group,
Department of Computing Science,
University of Glasgow,
Glasgow G12 8QQ
Scotland.

or

The Secretary,
Persistent Programming Research Group,
Department of Computational Science,
University of St. Andrews,
North Haugh,
St. Andrews KY16 9SS
Scotland.

Books

Davie, A.J.T. & Morrison, R.
"Recursive Descent Compiling", Ellis-Horwood Press (1981).

Atkinson, M.P. (ed.)
"Databases", Pergammon Infotech State of the Art Report, Series 9, No.8, January 1982. (535 pages).

Cole, A.J. & Morrison, R.
"An introduction to programming with S-algol", Cambridge University Press, Cambridge, England, 1982.

Stocker, P.M., Atkinson, M.P. & Gray, P.M.D. (eds.)
"Databases - Role and Structure", Cambridge University Press, Cambridge, England, 1984.

Published Papers

Morrison, R.
"A method of implementing procedure entry and exit in block structured high level languages". Software, Practice and Experience 7, 5 (July 1977), 535-537.

Morrison, R. & Podolski, Z.
"The Graffiti graphics system", Proc. of the DECUS conference, Bath (April 1978), 5-10.

Atkinson, M.P.
"A note on the application of differential files to computer aided design", ACM SIGDA newsletter Summer 1978.

Atkinson, M.P.
"Programming Languages and Databases", Proceedings of the 4th International Conference on Very Large Data Bases, Berlin, (Ed. S.P. Yao), IEEE, Sept. 78, 408-419. (A revised version of this is available from the University of Edinburgh Department of Computer Science (EUCS) as CSR-26-78).

Atkinson, M.P.
"Progress in documentation: Database management systems in library automation and information retrieval", Journal of Documentation Vol.35, No.1, March 1979, 49-91. Available as EUCS departmental report CSR-43-79.

Gunn, H.I.E. & Morrison, R.
"On the implementation of constants", Information Processing Letters 9, 1 (July 1979), 1-4.

Atkinson, M.P.
"Data management for interactive graphics", Proceedings of the Infotech State of the Art Conference, October 1979. Available as EUCS departmental report CSR-51-80.

Atkinson, M.P. (ed.)
"Data design", Infotech State of the Art Report, Series 7, No.4, May 1980.

Morrison, R.
"Low cost computer graphics for micro computers", Software Practice and Experience, 12, 1981, 767-776.

Atkinson, M.P., Chisholm, K.J. & Cockshott, W.P.
"PS-algol: An Algol with a Persistent Heap", ACM SIGPLAN Notices Vol.17, No. 7, (July 1981) 24-31. Also as EUCS Departmental Report CSR-94-81.

Atkinson, M.P., Chisholm, K.J. & Cockshott, W.P.
"Nepal - the New Edinburgh Persistent Algorithmic Language", in Database, Pergammon Infotech State of the Art Report, Series 9, No.8, 299-318 (January 1982) - also as EUCS Departmental Report CSR-90-81.

Morrison, R.
"S-algol: a simple algol", Computer Bulletin II/31 (March 1982).

Morrison, R.
"The string as a simple data type", Sigplan Notices, Vol.17,3, 46-52, 1982.

Atkinson, M.P., Bailey, P.J., Chisholm, K.J., Cockshott, W.P. & Morrison, R.
"Progress with Persistent Programming", presented at CREST course UEA, September 1982, revised in "Databases - Role and Structure", see PPRR-8-84.

Morrison, R.
"Towards simpler programming languages: S-algol", IUCC Bulletin 4, 3 (October 1982), 130-133.

Atkinson, M.P., Chisholm, K.J. & Cockshott, W.P.
"Problems with persistent programming languages", presented at the Workshop on programming languages and database systems, University of Pennsylvania. October 1982. Circulated (revised) in the Workshop proceedings 1983, see PPRR-2-83.

Atkinson, M.P.
"Data management", in Encyclopedia of Computer Science and Engineering 2nd Edition, Ralston & Meek (editors) January 1983. van Nostrand Reinhold.

Atkinson, M.P., Chisholm, K.J. & Cockshott, W.P.
"Algorithms for a Persistent Heap", Software Practice and Experience, Vol.13, No.3, 259-272 (March 1983). Also as EUCS Departmental Report CSR-109-82.

Atkinson, M.P., Chisholm, K.J. & Cockshott, W.P.
"CMS - A chunk management system", Software Practice and Experience, Vol.13, No.3 (March 1983), 273-285. Also as EUCS Departmental Report CSR-110-82.

Atkinson, M.P., Bailey, P.J., Chisholm, K.J., Cockshott, W.P. & Morrison, R.
"Current progress with persistent programming", presented at the DEC workshop on Programming Languages and Databases, Boston, April 1983.

Atkinson, M.P., Bailey, P.J., Chisholm, K.J., Cockshott, W.P. & Morrison, R.
"An approach to persistent programming", The Computer Journal, 1983, Vol.26, No.4, 360-365 - see PPRR-2-83.

Atkinson, M.P., Bailey, P.J., Chisholm, K.J., Cockshott, W.P. & Morrison, R.
"PS-algol a language for persistent programming", 10th Australian Computer Conference, Melbourne, Sept. 1983, 70-79 - see PPRR-2-83.

Morrison, R., Weatherill, M., Podolski, Z. & Bailey, P.J.
"High level language support for 3-dimension graphics", Eurographics Conference Zagreb, North Holand, 7-17, Sept. 1983. (ed. P.J.W. ten Hagen).

Cockshott, W.P., Atkinson, M.P., Chisholm, K.J., Bailey, P.J. & Morrison, R.
"POMS : a persistent object management system", Software Practice and Exerience, Vol.14, No.1, 49-71, January 1984.

Kulkarni, K.G. & Atkinson, M.P.
"Experimenting with the Functional Data Model", in Databases - Role and Structure, Cambridge University Press, Cambridge, England, 1984.

Atkinson, M.P. & Morrison, R.
"Persistent First Class Procedures are Enough", Foundations of Software Technology and Theoretical Computer Science (ed. M. Joseph & R. Shyamasundar) Lecture Notes in Computer Science 181, Springer Verlag, Berlin (1984).

Atkinson, M.P., Bocca, J.B., Elsey, T.J., Fiddian, N.J., Flower, M., Gray, P.M.D. Gray, W.A., Hepp, P.E., Johnson, R.G., Milne, W., Norrie, M.C., Omololu, A.O., Oxborrow, E.A., Shave, M.J.R., Smith, A.M., Stocker, P.M. & Walker, J.
"The Proteus distributed database system", proceedings of the third British National Conference on Databases, (ed. J. Longstaff), BCS Workshop Series, Cambridge University Press, Cambridge, England, (July 1984).

Atkinson, M.P. & Morrison, R.
"Procedures as persistent data objects", ACM TOPLAS 7, 4, 539-559, (Oct. 1985) - see PPRR-9-84.

Morrison, R.,Bailey, P.J., Dearle, A., Brown, P. & Atkinson, M.P.
"The persistent store as an enabling technology for integrated support environments", 8th International Conference on Software Engineering, Imperial College, London (August 1985), 166-172 - see PPRR-15-85.

Atkinson, M.P. & Morrison, R.
"Types, bindings and parameters in a persistent environment", proceedings of Data Types and Persistence Workshop, Appin, August 1985, 1-24 - see PPRR-16-85.

Davie, A.J.T.
"Conditional declarations and pattern matching", proceedings of Data Types and Persistence Workshop, Appin, August 1985, 278-283 - see PPRR-16-85.

Krablin, G.L.
"Building flexible multilevel transactions in a distributed persistent environment, proceedings of Data Types and Persistence Workshop, Appin, August 1985, 86-117 - see PPRR-16-85.

Buneman, O.P.
"Data types for data base programming", proceedings of Data Types and Persistence Workshop, Appin, August 1985, 291-303 - see PPRR-16-85.

Cockshott, W.P.
"Addressing mechanisms and persistent programming", proceedings of Data Types and Persistence Workshop, Appin, August 1985, 363-383 - see PPRR-16-85.

Norrie, M.C.
"PS-algol: A user perspective", proceedings of Data Types and Persistence Workshop, Appin, August 1985, 399-410 - see PPRR-16-85.

Owoso, G.O.
"On the need for a Flexible Type System in Persistent Programming Languages", proceedings of Data Types and Persistence Workshop, Appin, August 1985, 423-438 - see PPRR-16-85.

Morrison, R., Brown, A.L., Bailey, P.J., Davie, A.J.T. & Dearle, A.
"A persistent graphics facility for the ICL PERQ", Software Practice and Experience, Vol.14, No.3, (1986) - see PPRR-10-84.

Atkinson, M.P. and Morrison R.
"Integrated Persistent Programming Systems", proceedings of the 19th Annual Hawaii International Conference on System Sciences, January 7-10, 1986 (ed. B. D. Shriver), vol IIA, Software, 842-854, Western Periodicals Co., 1300 Rayman St., North Hollywood, Calif. 91605, USA - see PPRR-19-85.

Atkinson, M.P., Morrison, R. and Pratten, G.D.
"A Persistent Information Space Architecture", proceedings of the 9th Australian Computing Science Conference, January, 1986 - see PPRR-21-85.

Kulkarni, K.G. & Atkinson, M.P.
"EFDM : Extended Functional Data Model", The Computer Journal, Vol.29, No.1, (1986) 38-45.

Buneman, O.P. & Atkinson, M.P.
"Inheritance and Persistence in Database Programming Languages"; proceedings ACM SIGMOD Conference 1986, Washington, USA May 1986 - see PPRR-22-86.

Morrison R., Dearle, A., Brown, A. & Atkinson M.P.; "An integrated graphics programming environment", Computer Graphics Forum, Vol. 5, No. 2, June 1986, 147-157 - see PPRR-14-86.

Atkinson, M.G., Morrison, R. & Pratten G.D.
"Designing a Persistent Information Space Architecture", proceedings of Information Processing 1986, Dublin, September 1986, (ed. H.J. Kugler), 115-119, North Holland Press.

Brown, A.L. & Dearle, A.
"Implementation Issuses in Persistent Graphics", University Computing, Vol. 8, N0. 2, (Summer 1986) - see PPRR-23-86.

Kulkarni, K.G. & Atkinson, M. P.
"Implementing an Extended Functional Data Model Using PS-algol", Software - Practise and Experience, Vol. 17(3), 171-185 (March 1987)

Cooper, R.L. & Atkinson, M.P.
"The Advantages of a Unified Treatment of Data", Software Tool 87: Improving Tools, Advance Computing Series, 8, 89-96, Online Publications, June 1987.

Atkinson, M.P, Morrison, R. & Dearle, A.
"A strongly typed persistent object store", 1986 International Workshop on Object-Oriented Database Systems, Pacific Grove, California (September 1986).

Atkinson, M.P., Morrison, R. & Pratten G.D.
"PISA : A persistent information space architecture", ICL Technical Journal 5, 3 (May 1987),477-491.

Atkinson, M.P. & Morrison, R.
"Polymorphic Names, Types, Constancy and Magic in a Type Secure Persistent Object Store". Presented at the 2nd International Workshop on Persistent Object Stores, Appin, August 1987.

Cooper, R. & Atkinson, M.P.
"Requirements Modelling in a Persistent Object Store". Presented at the 2nd International Workshop on Persistent Object Stores, Appin, August 1987.

Wai, F.
"Distribution and Persistence". Presented at the 2nd International Workshop on Persistent Object Stores, Appin, August 1987.

Philbrow, P.
"Associative Storage and Retrieval: Some Language Design Issues". Presented at the 2nd International Workshop on Persistent Object Stores, Appin, August 1987.

Guy, M.R.
"Persistent Store - Successor to Virtual Store". Presented at the 2nd International Workshop on Persistent Object Stores, Appin, August 1987.

Dearle, A.
"Constructing Compilers in a Persistent Environment". Presented at the 2nd Internaional Workshop on Persistent Object Stores, Appin, August 1987.

Carrick, R. & Munro, D.
"Execution Strategies in Persistent Systems". Presented at the 2nd International Workshop on Persistent Object Stores, Appin, August 1987.

Brown, A.L.
"A Distributed Stable Store". Presented at the 2nd International Workshop on Persistent object Stores, Appin, August 1987.

Cooper, R.L., Atkinson, M.P., Dearle, A. & Abderrahmane, D.
"Constructing Database Systems in a Persistent Environment". Proceedings of the Thirteenth Internaional Conference on Very Large Databases, Brighton, September 1987.

Atkinson, M.P. & Morrison, M.
"Polymorphic Names and Iterations", presented at the Workshop on Database Programming Languages, Roscoff, September 1987.

Internal Reports

Morrison, R.
"S-Algol language reference manual", University of St Andrews CS-79-1, 1979.

Bailey, P.J., Maritz, P. & Morrison, R.
"The S-algol abstract machine", University of St Andrews CS-80-2, 1980.

Atkinson, M.P., Hepp, P.E., Ivanov, H., McDuff, A., Proctor, R. & Wilson, A.G.
"EDQUSE reference manual", Department of Computer Science, University of Edinburgh, September 1981.

Hepp, P.E. and Norrie, M.C.
"RAQUEL: User Manual", Department of Computer Science Report CSR-188-85, University of Edinburgh.

Norrie, M.C.
"The Edinburgh Node of the Proteus Distributed Database System", Department of Computer Science Report CSR-191-85, University of Edinburgh.

Theses

The following theses, for the degree of Ph. D. unless otherwise stated, have been produced by members of the group and are available from the address already given,

W.P. Cockshott
Orthogonal Persistence, University of Edinburgh, February 1983.

K.G. Kulkarni
Evaluation of Functional Data Models for Database Design and Use, University of Edinburgh, 1983.

P.E. Hepp
A DBS Architecture Supporting Coexisting Query Languages and Data Models, University of Edinburgh, 1983.

G.D.M. Ross
Virtual Files: A Framework for Experimental Design, University of Edinburgh, 1983.

G.O. Owoso
Data Description and Manipulation in Persistent Programming Languages, University of Edinburgh, 1984.

J. Livingstone
Graphical Manipulation in Programming Languages: Some Experiments, M.Sc., University of Glasgow, 1987.

Persistent Programming Research Reports

This series was started in May 1983. The following list gives those which have been produced at 17th September 1987. Copies of documents in this list may be obtained by writing to the addresses already given.

PPRR-1-83	The Persistent Object Management System - Atkinson,M.P., Bailey, P., Chisholm, K.J., Cockshott, W.P. and Morrison, R.	£1.00
PPRR-2-83	PS-algol Papers: a collection of related papers on PS-algol - Atkinson, M.P., Bailey, P., Cockshott, W.P., Chisholm, K.J. and Morrison, R.	£2.00
PPRR-5-83	Experimenting with the Functional Data Model - Atkinson, M.P. and Kulkarni, K.G.	£1.00
PPRR-6-83	A DBS Architecture supporting coexisting user interfaces: Description and Examples - Hepp, P.E.	£1.00
PPRR-7-83	EFDM - User Manual - K.G.Kulkarni	£1.00
PPRR-8-84	Progress with Persistent Programming - Atkinson,M.P., Bailey, P., Cockshott, W.P., Chisholm, K.J. and Morrison, R.	£2.00
PPRR-9-84	Procedures as Persistent Data Objects - Atkinson, M.P. and Morrison, R.	£1.00
PPRR-10-84	A Persistent Graphics Facility for the ICL PERQ - Morrison, R., Brown, A.L., Bailey, P.J., Davie, A.J.T. and Dearle, A.	£1.00
PPRR-11-85	PS-algol Abstract Machine Manual	£1.00
PPRR-12-87	PS-algol Reference Manual - fourth edition	£2.00
PPRR-13-85	CPOMS - A Revised Version of The Persistent Object Management System in C - Brown, A.L. and Cockshott, W.P.	£2.00
PPRR-14-86	An Integrated Graphics Programming Environment - 2nd edition - Morrison, R., Brown, A.L., Dearle, A. and Atkinson, M.P.	£1.00
PPRR-15-85	The Persistent Store as an Enabling Technology for an Integrated Project Support Environment - Morrison, R., Dearle, A, Bailey, P.J., Brown, A.L. and Atkinson, M.P.	£1.00
PPRR-16-85	Proceedings of the Persistence and Data Types Workshop, Appin, August 1985 - ed. Atkinson, M.P., Buneman, O.P. and Morrison, R.	£15.00
PPRR-17-85	Database Programming Language Design - Atkinson, M.P. and Buneman, O.P.	£3.00

PPRR-18-85 The Persistent Store Machine -
Cockshott, W.P. £2.00

PPRR-19-85 Integrated Persistent Programming Systems -
Atkinson, M.P. and Morrison, R. £1.00

PPRR-20-85 Building a Microcomputer with Associative Virtual Memory -
Cockshott, W.P. £1.00

PPRR-21-85 A Persistent Information Space Architecture -
Atkinson, M.P., Morrison, R. and Pratten, G.D. £1.00

PPRR-22-86 Inheritance and Persistence in Database Programming Languages -
Buneman, O.P. and Atkinson, M.P. £1.00

PPRR-23-86 Implementation Issues in Persistent Graphics -
Brown, A.L. and Dearle, A. £1.00

PPRR-24-86 Using a Persistent Environment to Maintain a Bibliographic Database -
Cooper, R.L., Atkinson, M.P. & Blott, S.M. £1.00

PPRR-25-87 Applications Programming in PS-algol -
Cooper, R.L. £1.00

PPRR-26-86 Exception Handling in a Persistent Programming Language -
Philbrow, P & Atkinson M.P. £1.00

PPRR-27-87 A Context Sensitive Addressing Model -
Hurst, A.J. £1.00

PPRR-28-86b A Domain Theoretic Approach to Higher-Order Relations -
Buneman, O.P. & Ochari, A. £1.00

PPRR-29-86 A Persistent Store Garbage Collector with Statistical Facilities -
Campin, J. & Atkinson, M.P £1.00

PPRR-30-86 Data Types for Data Base Programming -
Buneman, O.P. £1.00

PPRR-31-87 An Introduction to PS-algol Programming (third edition) -
Carrick, R., Cole, A.J. & Morrison, R. £1.00

PPRR-32-87 Polymorphism, Persistence and Software Reuse in a Strongly Typed Object Oriented Environment -
Morrison, R, Brown, A, Connor, R and Dearle, A £1.00

PPRR-33-87 Safe Browsing in a Strongly Typed Persistent Environment -
Dearle, A and Brown, A.L. £1.00

PPRR-34-87 Constructing Database Systems in a Persistent Environment -
Cooper, R.L., Atkinson, M.P., Dearle, A. and Abderrahmane, D. £1.00

PPRR-35-87 A Persistent Architecture Intermediate Language -
Dearle, A. £1.00

PPRR-36-87 Persistent Information Architectures -
Atkinson, M.P., Morrison R. & Pratten, G.D. £1.00

PPRR-37-87 PS-algol Machine Monitoring -
Loboz, Z. £1.00

PPRR-38-87 Flexible Incremental Bindings in a Persistent Object Store -
Morrison, R., Atkinson, M.P. and Dearle, A. £1.00

PPRR-39-87 Polymorphic Persistent Processes -
Morrison, R., Barter, C.J., Brown, A.L., Carrick, R., Connor, R., Dearle, A., Hurst, A.J.and Livesey, M.J. £1.00

PPRR-40-87 Andrew, Unix and Educational Computing -
Hansen, W. J. £1.00

PPRR-41-87 Factors that Affect Reading and Writing with Personal Computers and Workstations -
Hansen, W. J. and Haas, C. £1.00

PPRR-42-87 A Practical Algebra for Substring Expressions -
Hansen, W. J. £1.00

PPRR-43-87 The NESS Reference Manual -
Hansen, W. J. £1.00

PPRR-44-87 Persistent Object Systems: their design, implementation and use. (proceedings of the Appin workshop August 1987)
ed. Atkinson, M.P., Buneman, O.P. and Morrison, R. £20.00

PPRR-45-87 Delayed Binding and Type Checking in Database Programming Languages -
Atkinson, M.P., Buneman, O.P. & Morrison, R. £1.00

PPRR-46-87 Transactions and Concurrency -
Krablin, G.L. £1.00

PPRR-47-87 Persistent Information Space Architecture - PISA Club Rules
Atkinson, M.P., Lucking, J.R., Morrison, R. and Pratten, G.D. £1.00

PPRR-48-87 An Event-Driven Software Architecture
Cutts, Q. and Kirby, G. £1.00

PPRR-49-87 An Implementation of Multiple Inheritance in a Persistent Environment
Benson, P.J., D'Souza, E.B., Rennie, I.S., Waddell, S.J. £1.00

PPRR-50-87 A Distributed Stable Store
Brown, A.L. £1.00

PPRR-51-87 Constructing Compilers in a Persistent Environment
Dearle, A. £1.00