



**AUTHOR:**

**TITLE:**

**YEAR:**

**OpenAIR citation:**

This work was submitted to- and approved by Robert Gordon University in partial fulfilment of the following degree:

---

**OpenAIR takedown statement:**

Section 6 of the “Repository policy for OpenAIR @ RGU” (available from <http://www.rgu.ac.uk/staff-and-current-students/library/library-policies/repository-policies>) provides guidance on the criteria under which RGU will consider withdrawing material from OpenAIR. If you believe that this item is subject to any of these criteria, or for any other reason should not be held on OpenAIR, then please contact [openair-help@rgu.ac.uk](mailto:openair-help@rgu.ac.uk) with the details of the item and the nature of your complaint.

This is distributed under a CC \_\_\_\_\_ license.

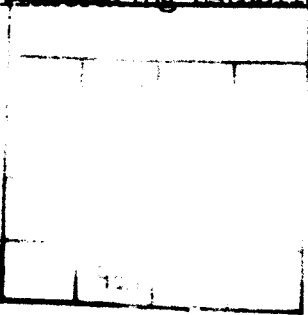
---

THE MACHINE REFINEMENT OF RAW GRAPHIC DATA FOR TRANSLATION  
INTO A LOW LEVEL DATA BASE FOR COMPUTER AIDED ARCHITECTURAL  
DESIGN (CAAD).

David Mark Leifer., B.Sc, B.Arch, Architect.

A Thesis presented in partial fulfilment of the  
requirements for the Degree of Doctor of Philosophy  
by research of the CNAA.

June 1984.

- Awarding Establishment : Council for National Academic Awards  
of the U.K.
- Sponsoring Establishment : (The Scott Sutherland School of  
Architecture),  
Robert Gordon's Institute of Technology  
Aberdeen.
- Collaborating Establishment : ABACUS, University of Strathclyde,  
Department of Architecture and Building  
Science,  
Glasgow.
- 

## Thesis Abstract

It is argued that a significant feature which acts as a disincentive against the adoption of CAAD systems by small private architectural practices, is the awkwardness of communicating with computers when compared with traditional drawing board techniques. This consideration, although not perhaps the dominant feature, may be mitigated by the development of systems in which the onus of communicating is placed on the machine, through the medium of an architect's sketch plan drawing.

In reaching this conclusion, a design morphology is suggested, in which the creative generation of building designs is set in the context of the development of a 'data-base' of information which completely and consistently describes the architect's hypothetical building solution.

This thesis describes research carried out by the author between September 1981 and September 1984, in order to produce a FORTRAN computer program to capture, enhance, decode and interpret an architect's sketch plan drawing, and subsequently reformat the data so acquired, such that it may be fed directly into existing CAAD programs, (in particular BIBLE and GOAL, developed by the ABACUS Unit of the University Of Strathclyde). A video recording which accompanies this thesis demonstrates this program, (ENIGMA), in use.

Methods of describing the geometries of real world polyhedral objects are examined, as are the data structures necessary to hold these descriptions in a machine compatible form.

Four existing computer programs which deal with geometrical descriptions of polyhedral objects are reviewed in respect of their data structures, and the limitations consequently imposed upon the program users.

The major part of the work undertaken, relates to the development and coding of algorithms to undertake the processes of the ENIGMA system, and the basic logic of the various sub-systems is described.

The significance of the ENIGMA system to the development of 'Knowledge Engineering' CAAD programs is suggested, as are areas for future work to expand and improve the ENIGMA program.

## Thesis Abstract

This thesis describes research carried out by the author between 1981 and 1984, and describes the theory, development and application of algorithms to interpret architect's sketch plan drawings, and hence permit the encoding of building geometries for CAAD applications programs.

The structure of UK architectural practices, the take-up of CAAD aids by the profession, and the range of services offered to clients by architects are noted. A design morphology is described in which the need for appraisal at the earliest stages, and the key role of geometry data are identified.

The current state of computer graphics technology is surveyed.

Methods of encoding geometry descriptions and their subsequent effects upon apposite data structures reviewed. Four existing graphics handling CAD computer programmes are examined with respect to their data structures.

The need for improved solutions to graphic data interfacing is identified, and algorithms for 'machine intelligent' interpretation of two dimensional sketch drawings described.

A suite of computer programmes to exploit these interpretation algorithms, in which crude architectural sketches are translated into a general purpose CAAD geometry database is described.

Areas for further work, improvements and inclusions into the described system are discussed, and strategies for their realisation suggested.

Two published papers, the first written by the author, the second co-jointly, both related to the research work are included.

## Preliminaries

Contents	i
List Of Tables	viii
List Of Figures	viii
Supervision Arrangements	xiii
Acknowledgements	xiv
Declarations	xv
Glossary Of Terms, Definitions And Abbreviations	xvii
Summary Of Thesis	xviii
Foreword	xix

## Contents

Section Reference	Subject	Sectional Page Number	Page Number
<u>Preliminaries</u>			
	List Of Tables	viii	
	List Of Figures	viii	
	Supervision Arrangements	xiii	
	Acknowledgements	xiv	
	Declarations	xv	
	Glossary Of Terms, Definitions And Abbreviations	xvii	
	Summary Of Scientific Method	xviii	
	Foreword	xix	
<u>Chapter 1 Architects And Computers</u>			
	Contents		1
1	Introduction	1.1	2
2	Structure Of The Architectural Profession	1.1	2
3	Why Small Practices?	1.2	3
4	Adoption Of CAAD Aids By Practice	1.3	4
5	Structure Of The Thesis	1.5	6
	References	1.6	7
<u>Chapter 2 Architectural Practice</u>			
	Contents		8
1	Introduction	2.1	9
2	The Architectural Service	2.1	9
	2.1 RIBA Work Stages	2.1	9
	2.2 Work Stage C	2.5	13
	2.3 Effort Distribution Through Work Stages	2.6	14
3	The CAD Spectrum	2.7	15
4	The Design Process	2.9	17
	4.1 The Markus/Maver Model Of Design Activity	2.10	18
	4.2 Popper's Conjectures And Refutations	2.13	21

Section Reference	Subject	Sectional Page Number	Page Number
	<u>Chapter 2 (continued)</u>		
	4.3 A Helical Model	2.15	23
	4.4 Summary	2.18	26
	References	2.19	27
	<u>Chapter 3 Computer Technology</u>		
	Contents		29
1	Introduction	3.1	30
2	Resource Requirements For CAAD Systems	3.1	30
	2.1 Finance	3.1	30
	2.2 Effects Of CAAD Upon Work Practice	3.3	32
	2.3 Staffing And Expertise	3.4	33
	2.4 Objectivity Of Systems	3.6	35
	2.5 The Role Of Drawing	3.7	36
	2.6 Draughting Systems	3.8	37
3	The Current State Of CAAD Technology	3.9	38
	3.1 Recent Developments In Computer Technology	3.10	39
4	Computer Hardware	3.11	40
	4.1 Display Screens	3.11	40
	4.2 Graphics Input Devices	3.13	42
	1 Input Via The QWERTY Keyboard	3.13	42
	2 Stepping Keys	3.13	42
	3 Thumbwheels	3.14	43
	4 Joystick	3.14	43
	5 Trackerball	3.15	44
	6 The 'Mouse'	3.15	44
	7 Flat Bed Digitiser/Plotter	3.15	44
	8 Touch Sensitive Screen	3.16	45
	9 Light Pen	3.16	45
	10 Acoustic Pen	3.17	46

Section Reference	Subject	Sectional Page Number	Page Number
-------------------	---------	-----------------------	-------------

Chapter 3 (continued)

	11	Digitising Tablet		
		a) Point Mode	3.17	46
		b) Continuous Mode	3.18	47
	4.3	Physical Models	3.20	49
	4.4	Summary	3.21	50
5		Computer Software	3.22	51
6		The Man Machine Interface	3.26	55
7		Summary	3.29	58
8		Conclusion	3.30	59
		References	3.31	60

Chapter 4 Graphic Data And Data Structures

		Contents		64
1		Graphic Data	4.1	65
2		Pictorial Data	4.1	65
	2.1	Characteristics	4.1	65
		1 Physical	4.1	65
		2 Noetic	4.2	66
		3 Imputed	4.3	67
		4 Representational	4.4	68
	2.2	Human Perception	4.4	68
3		Geometric Description Methods	4.7	71
	3.1	Point Set Method	4.7	71
	3.2	Boundary Description Method	4.10	74
	3.3	Boolean Description Method	4.12	76
4		Data Structures	4.13	77
	4.1	Point Set Method	4.15	79
	4.2	Boundary Description Method	4.19	83
	4.3	Boolean Description Method	4.26	90
	4.4	Conclusions To Data Structures	4.26	90
5		Two And Three Dimension	4.27	91
		References	4.29	93



Section Reference	Subject	Sectional Page Number	Page Number
	<u>Chapter 5</u> <u>Existing Graphics Programs</u>		
	Contents		95
1	Introduction	5.1	96
2	SKETCHPAD	5.2	97
	2.1    Operation	5.3	98
	2.2    Criticism	5.5	100
3	OXSYS/BDS	5.6	101
	3.1    Data Structure	5.8	103
	3.2    User Restrictions And Further Features	5.9	104
	3.3    Criticism	5.10	105
4	GABLE	5.10	105
	4.1    Hardware Configuration	5.11	106
	4.2    Operation	5.11	106
	4.3    Data Structure - Floors	5.11	106
	4.4    Data Structure - Roofs	5.13	108
	4.5    Criticism	5.14	109
5	GRAMP	5.15	110
	5.1    Limitations	5.15	110
	5.2    Data Structure	5.16	111
	5.3    Criticism	5.18	113
6	Summary	5.18	113
	References	5.19	114

Chapter 6    Refinement And Translation Of Raw Graphic Data

	Contents		116
1	Review	6.1	117
2	Definition Of The Problem	6.2	118
	2.1    GABLE	6.2	118
	2.2    HUNCH	6.3	119
	2.3    Criticism	6.5	121
	2.4    Proposed Solution	6.6	122

Section Reference	Subject	Sectional Page Number	Page Number
<u>Chapter 6 (continued)</u>			
3	Resource Limitations	6.7	123
	3.1 Hardware	6.7	123
	3.2 Firmware	6.8	124
	3.3 Software	6.8	124
4	ENIGMA	6.10	126
	4.1 Data Capture	6.11	127
	4.2 Line Identification	6.15	131
	Sequential Curvature	6.18	134
	Line Of Least Squares	6.18	134
	4.3 Corner Enhancement	6.23	139
	Merging Near Points	6.24	140
	4.4 Shape Identification	6.28	144
	Line Dissociation Algorithm	6.28	144
	Tab Elimination	6.30	146
	Tracing Algorithm	6.32	148
	Closure Check	6.40	156
	Duplicate Erasure	6.42	158
	Alternative Duplicate Erasure Methods	6.44	160
	Area Calculation	6.45	161
5	Summary	6.45	161
	References	6.49	165

Chapter 7    ENIGMA

	Contents		166
1	Introduction	7.1	167
2	Geometry Data File Management	7.1	167
3	Geometry Creation Devices	7.3	169
	3.1 Tablet In Continuous Digitisation Mode	7.6	172
	3.2 Tablet In Orthogonal Mode	7.6	172
	3.3 Tektronix 4010 Graphics Terminal	7.8	174

Section Reference	Subject	Sectional Page Number	Page Number
<u>Chapter 7 (continued)</u>			
4	Output Options	7.13	179
	4.1 BIBLE Output	7.13	179
	4.2 GOAL Output	7.15	181
	4.3 Crude Appraisal	7.17	183
	4.4 Crude Visualisation	7.19	185
5	Resume	7.20	186
<u>Chapter 8 Summary</u>			
	Contents		187
1	Review	8.1	188
2	Assertion	8.2	189
3	Conclusion	8.2	189
4	Derivation Of Vocabulary	8.4	191
5	Caveats	8.7	194
<u>Chapter 9 Suggested Further Work</u>			
	Contents		196
1	Introduction	9.1	197
2	Editing Functions	9.1	197
	2.1 Adding A Line	9.4	200
	2.2 Adding A Shape	9.4	200
	2.3 Repeating A Shape	9.5	201
	2.4 Deleting A Shape	9.6	202
	2.5 Deleting A Line	9.8	204
	2.6 Moving A Point	9.11	207
	2.7 Moving A Line	9.12	208
	2.8 Moving A Shape	9.13	209
3	Other Editing Functions	9.15	211
	3.1 Orthogonalisation	9.16	212
4	Interpretation Of Multi-Storey Buildings	9.17	213
5	Extensions To ENIGMA	9.17	213
6	Conclusion	9.18	214

	Subject	Page Number
	<u>Bibliography</u>	215
	<u>Appendix 1</u> <u>Publications</u>	
1	'Man Before The Machine: An Issue In CAAD' BoCAAD, University Of Strathclyde Vol.43-46 Feb 1983 pp.107-112	223
2	With B.G.Hammond	
	'A Graphics Interface To Complement Traditional Techniques' PARC83 Conference Proceedings pp.321-329 On-Line Conferénces                    Oct 1983	230
	<u>Appendix 2</u> <u>Line Of Least Squares Test</u>	238
	<u>Appendix 3</u> <u>ENIGMA User's Manual</u>	256

### List Of Tables

Table Number	Title	Section Reference	Page Number
1.1	CICA Survey Of Commercially Available Draughting Systems.	1. 3	4
1.2	AJ Computing Club Ownership Of The First 100 Members.	1. 4	5
4.1	Point Set For Object Depicted In Figure 4.8, Including Registration Of Adjacent Positive Cells.	4.18	82

### List Of Figures

Figure Number	Title	Section Reference	Page Number
2.1	Schematic Diagram Relating The Deployment Of Resources To The Importance Of Design Decisions	2. 5	13
2.2	Schematic Representation Of The Markus/Maver Model Of The Design Process	2.11	19
2.3	A Helical Model Of Design Activity Combining Both The Markus/Maver And Popper Models	2.15	23
4.1	Illustration Of Noetic Characteristics	4. 2	66
4.2	Ambiguous Imputed Characteristics	4. 3	67
4.3	A Schematic Representation Of A Perception Hierarchy	4. 6	70
4.4	The Point Set Method Of Encoding Shape Descriptions	4. 8	72
4.5	A Variegated Cell Structure	4. 9	73
4.6	The Boundary Description Method Of Encoding Shape Geometries	4.10	74
4.7	Parameters Of Three Dimensional Description For Mapping To A One Dimensional Data List	4.16	80
4.8	Solid Object To Be Encoded By The Point Set Method, To Demonstrate Identification Of Surface, Edge And Corner Cells	4.17	81
4.9	Data Set Describing Topological Relationships Of The Boundary Description Of A Polygonal Solid, With Reference To Coordinate Values For Instantiation Into Algebraic Equations	4.21	85

Figure Number	Title	Section Reference	Page Number
4.10	Tree Structure For Encoding Boundary Description Of Object From Figure 4.9	4.23	87
4.11	Further Data Structure Incorporating Lateral, Forward And Backward Pointers	4.25	89
5. 1	The SKETCHPAD Data Structure	5. 4	99
5. 2	Schematic Representation Of The OXSYS Data Structure	5. 8	103
5. 3	Schematic Representation Of The GABLE Data Structure For Recording Walls	5.12	107
5. 4	Schematic Representation Of The GABLE Data Structure For Recording Roofs	5.14	109
5. 5	Permissible Volume Types For Use With GRAMP	5.16	111
5. 6	Schematic Illustration Of GRAMP Data Structure	5.17	112
6. 1	Stylus Monitoring During Continuous Digitisation By HUNCH	6. 4	120
6. 2	Definition Of Bentness In HUNCH	6. 4	120
6. 3	ENIGMA Hardware Configuration	6. 8	124
6. 4	ENIGMA Software Configuration	6. 9	125
6. 5	Data Output From Tablet In Continuous Digitisation Mode	6.12	128
6. 6	Illustration Of Sensitivity Of Gradient To Stylus Error In Relation To Interval Length	6.13	129
6. 7	TABLIT - Data Capture And Validation Algorithm	6.14	130
6. 8	Data Format As Output From Subroutine TABLIT	6.15	131
6. 9	LINEID - Flowchart Of The Algorithm For Separating Distinct Lines From Sketch Data	6.17	133
6.10	Illustration Of Sequential Curvature	6.18	134
6.11	DEMO Program Input And Output	6.20	136
6.12	Comparison Of Sensitivities To Change In Direction By Comparing Gradients Between Segments And Line Of Least Squares	6.21	137
6.13	Output Data Format From Subroutine LINEID	6.22	138
6.14	Corners Lost As A Consequence Of Continuous Digitisation	6.23	139

Figure Number	Title	Section Reference	Page Number
6.15	Intersections Involving Near Parallel Lines	6.24	140
6.16	CORNER - Flowchart Showing The Corner Replacement Algorithm	6.25	141
6.17	MERGE - Flowchart Showing The Near Point Merging Algorithm	6.27	143
6.18	Intersection Possibilities Between Any Two Vectors	6.29	145
6.19	Flow Diagram Of Algorithm To Identify All Disected Lines And Direct Data To Splitting Routine	6.31	147
6.20	Flow Diagram Of Routine To Split Bisected Lines, In Conjunction With INTERX	6.32	148
6.21	68 Possible Relationships Between Two Joined Lines In Terms Of Their Horizontal And Vertical Increments	6.33	149
6.22	Matrix Of Possible Relationships Between Angles <u>a</u> And <u>b</u> To Calculate The Heading Of Line <u>bc</u> With Respect To Line <u>ab</u>	6.34	150
6.23	Schematic Flow Diagram Of Tracing Algorithm Showing Constituent Routines	6.36	152
6.24 A	First Section Of Flow Diagram Of Tracing Algorithm	6.37	153
6.24 B	Second Section Of Flow Diagram Of Tracing Algorithm	6.38	154
6.24 C	Third Section Of Flow Diagram Of Tracing Algorithm	6.39	155
6.25	The Relational Data Structure Employed For Storing The Geometry Data Base In ENIGMA	6.41	157
6.26	Routine For Appending Newly Validated Shape Data To The Data Structure	6.43	159
6.27	Flow Diagram Of Subroutine AREAS	6.46	162
6.28	Example Of Area Calculation For An Enclosed Polygon	6.46	162
7. 1	Schematic Representation Of ENIGMA Flow Path	7. 2	168
7. 2	File Management Flow Diagram, Omitting All Internal System Files	7. 4	170
7. 3	Flow Chart For Geometry Input Device Selection	7. 5	171

Figure Number	Title	Section Reference	Page Number
7. 4	Resolution Of Freehand Lines Into Orthogonal Equivalents	7. 7	173
7. 5	Flow Diagram Of The Subroutine To Drive The Tablet In Orthogonal Mode	7. 9	175
7. 6	Flow Diagram Of The Terminal Operating Routine	7.10	176
7. 7	Two Cases Of Superimposed Lines	7.11	177
7. 8	Logic Gate For Testing The Interaction Between Two Co-Linear Lines	7.12	178
7. 9	Shape Configuration Producing An Error In Overlap Decoding	7.12	178
7.10	Format Of Interpretation Of ENIGMA Data To BIBLE Data By Subroutine BIBOUT	7.14	180
7.11	Flow Diagram For BIBLE File Creating Routine	7.16	182
7.12	Flow Diagram Of Routine To Reformat ENIGMA File To GOAL Data File	7.18	184
8. 1	Generated Geometry Description Through Hardware	8. 5	192
8. 2	Interpreted Geometry Description By ENIGMA	8. 5	192
8. 3	Definition Of Generic Similarity Between Two Shapes	8. 7	194
9. 1	Schematic Representation Of ENIGMA Flow Path	9. 2	198
9. 2	Scheme For Determining Whether A point Lies Inside A Shape	9. 5	201
9. 3	The Effect Of Repeating An Existing Shape On The ENIGMA Data Structure	9. 7	203
9. 4	Object Described By The Data Base Shown In Figure 9.3	9. 8	204
9. 5	The Effect Of Shape Deletion On The ENIGMA Data Base	9. 9	205
9. 6	Scheme For Specifying A Line	9.10	206
9. 7	Topological Disruption Caused By Moving A Point	9.12	208
9. 8	The Effects Of Moving A Line On The ENIGMA Data Base	9.13	209
9. 9	Alteration To A Picture Through Moving A Line	9.14	210



Figure Number	Title	Section Reference	Page Number
9.10	Duplication Of Common Lines Prior To A Move Shape Operation	9.14	210
9.11	Topological Discontinuity Caused By Orthogonalisation	9.16	212

## Supervision Arrangements

### Director Of Studies

L.W.W.Laing, B.A, M.Sc, Ph.D, ARIAS

Lecturer

Scott Sutherland School Of Architecture  
Robert Gordon's Institute Of Technology  
Aberdeen

### Second Supervisors

Professor S.Wilkinson, B.Arch (Hons), RIBA, FRIAS

Head Of School

Scott Sutherland School Of Architecture  
Robert Gordon's Institute Of Technology  
Aberdeen

Professor T.W.Maver, B.Sc (Hons), Ph.D

Director

ABACUS Research Unit  
University Of Strathclyde  
Glasgow

## Acknowledgements

First and foremost, my main thanks go to Dr Lamond Laing, whose interest, enthusiasm, help and friendship have sustained me throughout the duration of this project; to Professor Tom Maver for his interest, helpful advice and constructive criticism; and to my friend and colleague Brian Hammond on who's programming advice I have found most useful, and whose current SERC Funded Research Project No. 82301583, "Application Of Dynamic Graphic Techniques To The Appraisal Of Domestic Activity Space Layouts." has contributed in no small measure to the ideas underlying my own work.

My thanks are also due to the Computer Services Unit of Robert Gordon's Institute of Technology for their prompt and free programming advice, and to the ABACUS Unit of the University of Strathclyde for creating the UTIL and TEKLIB subroutine libraries upon which I have relied.

Finally, thanks are also due to all my friends and fellow students who have given me their support and encouragement; and last but in no way least, to the Science and Engineering Research Council who have given me the opportunity to undertake this project in the first place, (Award No.81305664).

## DECLARATIONS

### 1 Content Of Work

1.1 The candidate has not, while registered for this CNAA Ph.D submission, been a registered candidate for another award of the CNAA or of a University during the research programme.

1.2 None of the original material contained in this thesis has been used in any other submission for an academic award. Acknowledgements for assistance received are given under the heading of Acknowledgements, and any excerpt from other work has been acknowledged by its source and author.

### 2 Advanced Studies

The 4 day Intensive Computer Course at Strathclyde University, Glasgow, 1982 was cancelled.

In fulfilment of the programme of related studies stated, the candidate has attended and participated in the B.Sc Hons. module in FORTRAN programming for architects at Robert Gordon's Institute of Technology, Spring Term of session 1981/82; attendance and participation at the residential SERC-Crest course, Designer/Computer, in Informatics at Teeside Polytechnic, Aug 23rd to Sept 4th 1982; CAD82 International Conference at Brighton, March 30th to April 1st 1982; and an advanced course of reading directed by the project supervisors, which sources are indicated in the Bibliography.

Additionally, the candidate has attended and participated in Eurographics82 International Conference, UMIST, Sept 8th to 10th 1982; Parc83 International Conference, Wembley, Oct 18th to 20th; visits to CAD Research Facilities at Leicester Polytechnic, and the universities of Edinburgh, Strathclyde, Heriot-Watt, Sheffield and Loughborough, and commercial CAD installations at GMW Computers, Berkhamsted, ARC Ltd, Cambridge, and ORCA Engineering, Aberdeen.

3 Copyright And Permission To Copy The Thesis

The author has the sole copyright of this thesis. The following conditions apply as regards permission to copy the thesis:

Depository and lending libraries are hereby granted powers of discretion to allow either themselves or a lender to copy the thesis in whole or in part without further reference to the author. No request for lending or copying of the thesis shall be refused by any library.

The permission to copy covers only single copies made for study purposes, subject to the normal conditions of acknowledgement. If copies are made of single sections, they should be identified by the thesis title page as the first page.

## Glossary Of Terms Definitions And Abbreviations

CAAD	Computer Aided Architectural Design.
CAD	Computer Aided Drawing, or Computer Aided Design.
CSG	Constructive Solid Geometry.
Database	The sum total of information stored, and knowledge implicit in a particular data structure
Data Structure	The conceptual arrangement of data items in related blocks of similar data designed to store and access a database.
EOF	End Of File marker which informs the computer operating system when it has reached the end of a data block held on a secondary storage device.
Holon	A node in a heirarchical data structure which is both a data set in itself and a subset of a superior holon.
Interface	The medium through which the humans representation of data is translated into that of the machine.
Line Element	A straight line of a picture which only has junctions at its end points.
Line segment	A straight line connecting two sequentially digitised points, which may be bisected by other line segments.
LOLS	Line Of Least Squares. A statistical test between points on a two dimensional graph which produces the description of a straight line which best fits through the scatter.
Microprocessor	An 8-bit word length computer processor with an internal clock cycle of less than 6 000 cycles per second.

## Summary Of Thesis

Chapters 1 and 2 establish the background to this study: Chapter 1 noting the extent of computer usage in the profession, and Chapter 2 reviewing the architect's role, and suggesting a morphological framework in which their design function may be placed.

Chapter 3 reviews the current state of computer technology; it's effects on practices, and the hardware, software and interfaces available to them.

Chapter 4 examines graphic descriptions of real objects, and the data structures necessary to describe such objects to machines when restricted to polyhedra.

Chapter 5 examines four existing computer programs which deal with the geometrical descriptions of polyhedral objects to examine how the descriptions and data structures have been accomodated, along with the restrictions thereby imposed upon the program user.

Chapters 6 and 7 describe the logic of operation of the several computer programs which make up the ENIGMA system of decoding, enhancing and interpreting an architects' sketch plan drawings. (This program is demonstrated in the video recording which accompanies this thesis.)

Chapter 8 contains a summary of what has been achieved, and suggests how the interpretation process might be used as a data-capture system for a (knowledge based' computer system.

Chapter 9 suggests areas for further work, extensions and improvements to the ENIGMA system.

The three appendices contain first, reproductions of papers published by the author which relate to the research project described herein; secondly the evidence of the erratic results of the 'Line Of Least Squares' statistical analysis as the data describes a line approaching the vertical; and finally a brief 'User Guide' to the ENIGMA suite of computer programs.

## Foreword

"A classical understanding sees the world primarily as underlying form itself. A romantic understanding sees it primarily in terms of immediate appearance.

The romantic mode is primarily inspirational, imaginative, creative, intuitive. Feelings rather than facts predominate. "Art" when it is opposed to "Science" is often romantic. It does not proceed by reason or laws. The classic mode, by contrast, proceeds by reason and by laws - which are themselves underlying forms of thought and behaviour.

Two kinds of logic are used - inductive and deductive. Inductive inferences start with observations and arrive at general conclusions...deductive inferences do the reverse. They start with general knowledge and predict a specific observation.

What we have is a conflict of visions of reality. The world as you see it right here, right now, is reality regardless of what the scientists say it might be. But the world as revealed by scientific discoveries is also reality, regardless of how it may appear. What you've got here, really, are two realities, one of immediate artistic appearance and one of underlying scientific explanation, and they don't match and they don't fit and they don't really have much of anything to do with one another."

Robert Pirsig.

Zen And The Art Of Motorcycle Maintenance.

The resolution of this artificial dichotomy between "immediate artistic appearance" and "underlying scientific explanation" is a matter of immediate concern if we are to have a technology working for all people.

Technology itself is inert, and may be used for good or ill. All too often it is the latter, because the means have become ends in themselves. Specialism, which the fantastic rush of technological progress has tended to promote, has meant the the 'priests' of the new technologies view the world through spectacles tinted by the



colour of their own pet specialism, which puts their view of the rest of the world into a false light. As John F.Kennedy said, "Man's scientific capability has outrun his spiritual capability: We have guided missiles and misguided men."

Architecture is perhaps the last of the truly liberal arts: Its study encompasses all disciplines, and there are few areas which do not bear a direct relevance to the study of man and his habitat. Architects, basing their designs upon both artistic and scientific considerations, are therefore in a position to mediate between technological progress on the one hand and its sociological implications on the other. In short, architects ought to be the promoters of Appropriate Technology, to which this thesis is devoted.

In some small measure, this work repays the debt I owe to James Freeney, who showed me what technology couldn't do, Lamond Laing who showed me what it could, Robert Pirsig for showing that the two views could be reconciled, and the Bootle family for showing that the reconciliation was necessary.

Dave Leifer

May 1984

## Chapter 1

### Architects And Computers

1	Introduction	1.1	2
2	Structure Of The Architectural Profession	1.1	2
3	Why Small Practices?	1.2	3
4	Adoption Of CAAD Aids By Practice	1.3	4
5	Structure Of The Thesis	1.5	6
	References	1.6	7

## CHAPTER 1

1 Introduction

This thesis is concerned with the description and creation of algorithms which could lead to the establishment of an interface between architect and currently available computer equipment, concentrating upon the manipulation of plan drawings. This interface being involved as the architect is in the process of developing his initial ideas about potential building form through sketch drawings: An interface more congenial to the architect, thus acting as an incentive for the adoption of CAAD aids and overcome the prevailing reticence evident through sections of the profession - particularly the small private architectural practices.

The adoption of CAAD aids facilitates the more thorough examination of design proposals before far reaching decisions are taken, hence improving the quality of the architects' finished product.

2 Structure Of The Architectural Profession

" There are about 26 000 architects in this country (the UK.). But of this number over half are 'salaried' - mainly on the payroll of local authorities or Government departments. The rest are divided between about 5 000 firms. A few of these are large, with branch offices dotted about the country and abroad, but most employ fewer than 20 people. An amazingly large number have retained the intimacy of private practice: over half the firms listed...have five or less architects on the staff." 1

In actual fact, eighty-five percent of all registered architects work in practices employing 10 or less full-time architectural staff.<sup>2</sup> From the above figures it would seem that some 11 000 architects are employed by about 4 500 practices; an average of between two and three architects per firm or office.

The predominance of small private practices goes a long way to explain why CAAD aids have not made the inroads into the

profession in comparison with others in building and construction industry: they simply do not have the investment capital to bear both capital and running costs that such systems entail.

### 3 Why Small Practices?

Four reasons may be offered to explain the proliferation of small architects practices;

a Until relatively recently, architects have tended to think of themselves as 'artists'. Trained in studios, the concept of 'the architect' under a clients' patronage has been perpetuated. It is only relatively recently that the profession has turned its attention to professionalism and technological competence.

Simultaneously, the pattern of patronage for larger projects has changed; the former individual client has largely been replaced by client committees who keep a much closer watch on the architects performance. This change appears to have slowed down the rate of establishment of new practices, but as yet, many firms, established after the Second World War have not been affected.

b Following from the above, it was the requirement for reconstruction following the Second World War that induced many architects to establish their own practices, confident in the continuity of workload.

The 'stop-go' policies of Government investment policies since the mid 1960's has had an adverse effect on architects confidence, and the present recession makes the establishment of a new practice precarious indeed.

c However, it is still comparatively inexpensive for an architect to set up his own practice. Many have done so on the strength of the expected income from just one project.

d Perhaps more than any other reason, the pattern of architectural practices seems to mirror the pattern of the cost of construction projects. The bulk of money spent annually on building construction is small scale.

The above observations are based upon the writers experience as a second generation architect, experience of working in such a small private practice, and through discussion with peers.

It is inevitable that the pattern will change with prevailing circumstances, but there is generally an appreciable time-lag between cause and effect.

#### 4 The Adoption Of CAAD Aids By Practice

The anticipated adoption of CAAD aids by the architectural profession has not happened to the extent predicted, even though such aids have been in development since the mid 1960's. Many multidisciplinary practices, who have engineering expertise, have adopted systems, but these firms are in the minority. The Table below shows the take-up nationally and worldwide of commercially available draughting systems by the construction industry.

System Supplier	System Name	Number of Installations					Launch Date
		UKci	UK	EUR	WOR	TOT	
Applicon	Image	7	51	200	1200	1400	
Autotrol	GS-1000		10	20	350	400	
ARC Ltd	GDS	20	23	3	23	49	1980
Admel/Bruning	Easidraf2	3	3		125	150	81/82
Cadam Inc	Cadam	5	20	60	200	300	71/78
Calcomp	IGS 500	3	4	25	90	120	1969
Carbs Ltd	Carbs	11	13	1	2	16	1973
Calma	Cadec		60	20	900	1000	71/72
CIS	Medusa	4	28	53	29	130	1980
Computervision	CAE	7	100	600	2400	3000	1973
Genesys*	Gable	6	8		2	10	1981
Intergraph	IGDS	10	40	90	350	500	73/81
GMW Computers	Rucaps	46	46	9	12	67	1977
McAuto	Unigraphics	1	17	18	170	200	1978
Oasys Ltd	Cadraw	4	6			6	80/81
Olivetti	IGS	3	15	60	30	100	1980
PAFEC Ltd	Dogs	3	27		7	34	1980
Sunmagraphics	Datagrid	6	10	15	120	150	79/80
Scott Wilson K	Gipsys	1	1			1	1979
Approximate Totals		140	500	1200	6000	7700	

Table 1.1 CICA Survey Of Commercially Available Draughting Systems.

Key: UKci construction industry installations in the UK  
 UK total installations in the UK  
 EUR remaining installations in Europe excluding UK  
 WOR remaining installations worldwide excluding Europe

Table 1.1 is reproduced from CICA Bulletin, Mar 82.<sup>2</sup>

Note that GABLE is an integrated CAAD system with a draughting element to it.

Whilst the figures in Table 1.1 refer to draughting systems specifically, they do give an indication of the up-take of computers generally, and also show the proportion of the total number of installations used by the UK construction industry.

Against these figures, the AJ Computer Club published a survey of the computer ownership of their first 100 members. This is shown in Table 1.2.<sup>3</sup>

Computers	No. of owners	Computers	No. of owners
Apple	28	Nascom	1
Aztech	1	Pet	19
BBC	7	Philips word processor	1
DEC minicomputer	2	Research Machines	1
Diablo	1	Sinclair ZX80	1
Equinox	1	Sinclair ZX81	2
Hewlett Packard	4	Sharp	1
IBM 3031	1	Superbrain	4
IBM Displaywriter	1	Tandy	5
ICL 2956	1	Vector graphics	1
Interdata	1	Video Genie	1
Microtan	1	Wang word processor	2
<b>Draughting Systems</b>			
Applicon	1	GDS	2
ARK II	1	Intergraph	1
Calcomp	2	RUCAPS	4
Gable	1		

Table 1.2 AJ Computing Club Computer Ownership Of The First 100 Members.

Most of the first 100 members are in the UK. The rest are in Belgium, Holland, Hong Kong, Malaysia and the Gulf States.

Obviously the tenor of the AJ Computing Club places an emphasis on microcomputers. Heavily committed organisations are more likely to be members of the CICA (Construction Industry Computer Association). The above figures do show that computers are being used by architects, if only in a more managerial or

support role.

## 5 Structure Of Thesis

Chapter 2 examines what it is that architects actually do and the service that they normally offer to clients. It enlarges upon the importance of decisions made at the early design stages, and the normal 'effort' distribution through the design process. In the light of this distribution the range of computer aids is considered.

Finally, a design morphology is proposed and discussed.

Chapter 3 discusses factors which act as disincentives to architects against the adoption of CAAD aids. This follows a consideration of the current state of computer hardware, software, and interface technologies.

Chapter 4 examines the characteristics of geometric descriptions, and the three possible mathematical description methods. The effects of these alternative descriptions on the data structures necessary to hold these descriptions is considered.

Chapter 5 briefly examines the way in which four existing graphic handling suites of computer software deal with geometric description and data structuring. The first of these is a seminal draughting coding whose fundamental ideas are central to many more recent draughting systems. The second is an example of a component based building method. The third is an example of an integrated CAAD system designed for implementation on a small desktop computer. The final example takes the graphic handling element from a system designed for a number of applications, implemented upon a large mainframe computer.

Chapter 6 summarises the arguments propounded in the previous chapters. It defines the problem central to the mis-match between CAAD systems and traditional design techniques in terms of the man-machine interface. HUNCH, a computer program conceived to

just this problem, is then critically examined. There follows a description of algorithms proposed to emulate many of the ideas contained in HUNCH, whilst rejecting others. These algorithms deal with the enhancement of crude architectural sketch plans, and their subsequent interpretation into enclosed shapes.

Included in this chapter is a description of identification using prime numbers.

Chapter 7 describes a suite of programs which utilise the algorithms described in the preceding chapter. This suite of programs are designed as a flexible interfacing system for use by architects at the early design stage to generate data necessary for more detailed future analysis and appraisal.

Chapter 8 summarises what has been achieved, and discusses the possibilities of using this interface system to automatically derive a higher level vocabulary as is used in the interrogation of expert systems.<sup>4</sup>

Finally, chapter 9 presents further extensions to the interface, principally the inclusion of editing functions.

### References

- 1 RIBA Directory Of Practices 1983  
RIBA Publications 1983 p 7
- 2 Fact Sheet On The Use Of Computers In The Construction Industry  
CICA Bulletin  
Construction Industry Computing Association Mar 1982
- 3 AJ Computing Club  
AJ 26th Jan 1983 p 70
- 4 What's What And Where's Where: Knowledge Engineering In The  
Representation Of Buildings By Computer  
John Gero, V.Tuncer Akiner, Anthony Radford  
PARC83 Proceedings  
On-Line Conferences Oct 1983 pp 205-215



## Chapter 2

### Architectural Practice

1	Introduction	2.1	9
2	The Architectural Service	2.1	9
	2.1 RIBA Work Stages	2.1	9
	2.2 Work Stage C	2.5	13
	2.3 Effort Distribution Through Work Stages	2.6	14
3	The CAD Spectrum	2.7	15
4	The Design Process	2.9	17
	4.1 The Markus/Maver Model Of Design Activity	2.10	18
	4.2 Popper's Conjectures And Refutations	2.13	21
	4.3 A Helical Model	2.15	23
	4.4 Summary	2.18	26
	References	2.19	27

## CHAPTER 2

1 Introduction

This chapter is divided into three sections: The first describes the extent of the service which architects offer to the client and the resource distribution required to realise this design service; the second summarises those areas in which a computer can provide 'aid' to the architect; whilst the third describes the processes by which the architect arrives at a building design.

2 The Architectural Service

In its Public Relations literature, the RIBA states that;

"the architects function is to translate the client's needs into a building that will serve his purpose in every way...It is the architects' aim to find for each the solution that will best serve his client's purpose while delighting the eye." 1

It is, therefore, the architect's function to conceptualise and communicate a hypothetical building form which satisfies the design criteria, both explicit in the brief, and implicit in the design problem. In deciding the priorities of the often contradictory design parameters, he is usually the arbiter.

## 2.1 RIBA WORK STAGES

The RIBA analyses the services provided to clients by architects by reference to 'work stages'. These work stages, enumerated A to L, split the design process into a number of discrete sub-processes. These are:

'Work Stage A: Inception

- 1 Discuss the client's requirements including timescale and any financial limits; assess these and give general advice on how to proceed; agree the architect's services.

- 2 Obtain from the client information on ownership and any lessors and lessees of the site, any existing buildings on the site, boundary fences and other enclosures and any known easements, encroachments, underground services, rights of way, rights of support and other relevant matters.
- 3 Visit the site and carry out an initial appraisal.
- 4 Advise on the need for other consultants' services and on the scope of these services.
- 5 Advise on the need for specialist contractors, sub-contractors and suppliers to design and execute part of the works to comply with the architects requirements.
- 6 Advise on the need for site staff.
- 7 Prepare where required an outline timetable and fee basis for further services for the client's approval.

Work Stage B: Feasibility

- 8 Carry out such studies as may be necessary to determine the feasibility of the client's requirements; review with the client alternative design and construction approaches and cost implications, advise on the need to obtain planning permissions, approvals under the building acts or regulations, and other similar statutory requirements.

Work Stage C: Outline Proposals

- 9 With other consultants where appointed, analyse the client's requirements; prepare outline proposals and an approximation of the construction cost for the client's preliminary approval.

Work Stage D: Scheme Design

- 10 With other consultants where appointed, develop a scheme design from the outline proposals taking into account amendments requested by the client; prepare a cost estimate; where applicable give an indication of possible start and completion dates for the building contract. The scheme design will illustrate the size and

character of the project in sufficient detail to enable the client to agree the spatial arrangements, materials and appearance.

- 11 With other consultants where appointed, advise the client of the implications of any subsequent changes on the cost of the project and the overall programme.
- 12 Make where required application for planning permission. The permission itself is beyond the architect's control and no guarantee that it will be granted can be given.

Work Stage E:           Detail Design

- 13 With other consultants where appointed, develop the scheme design; obtain the client's approval of the type of construction, quality of materials and standard of workmanship, co-ordinate any design work done by consultants, specialist contractors, sub-contractors and suppliers; obtain quotations and other information in connection with specialist work.
- 14 With other consultants where appointed, carry out cost checks as necessary; advise the client of the consequences of any subsequent changes on the cost and programme.
- 15 Make and negotiate where required, applications for approvals under building acts, regulations or other statutory requirements.

Work Stages F and G:    Production Information  
  and Bills of Quantities.

- 16 With other consultants where appointed, prepare production information including drawings, schedules and specification of materials and workmanship; provide information for bills of quantities, if any, to be prepared: all information complete in sufficient detail to enable contractor to prepare a tender.

Work Stage H:           Tender Action

- 17 Arrange where relevant, for other contracts to be let prior to the contractor commencing work.

- 18 Advise on, and obtain the client's approval to, a list of tenderers.
- 19 Invite tenders from approved contractors; appraise and advise on tenders submitted. Alternatively, arrange for a price to be negotiated with a contractor.

Work Stage J: Project Planning

- 20 Advise the contractor on the appointment of the contractor and on the responsibilities of the client under the terms of the building contract; where required prepare the building contract and arrange for it to be signed by the client and contractor; provide production information as required by the building contract.

Work Stage K: Operation on Site

- 21 Administer the terms of the building contract during operations on site.
- 22 Visit the site as appropriate to inspect generally the progress and quality of the work.
- 23 With other consultants where appointed, make where required periodic financial reports to the client including the effect of any variations on the construction cost.

Work Stage L: Completion

- 24 Administer the terms of the building contract relating to the completion of the works.
- 25 Give general advice on maintenance.
- 26 Provide the client with a set of drawings showing the building and the main lines of drainage; arrange for drawings of the services installations to be provided." 2

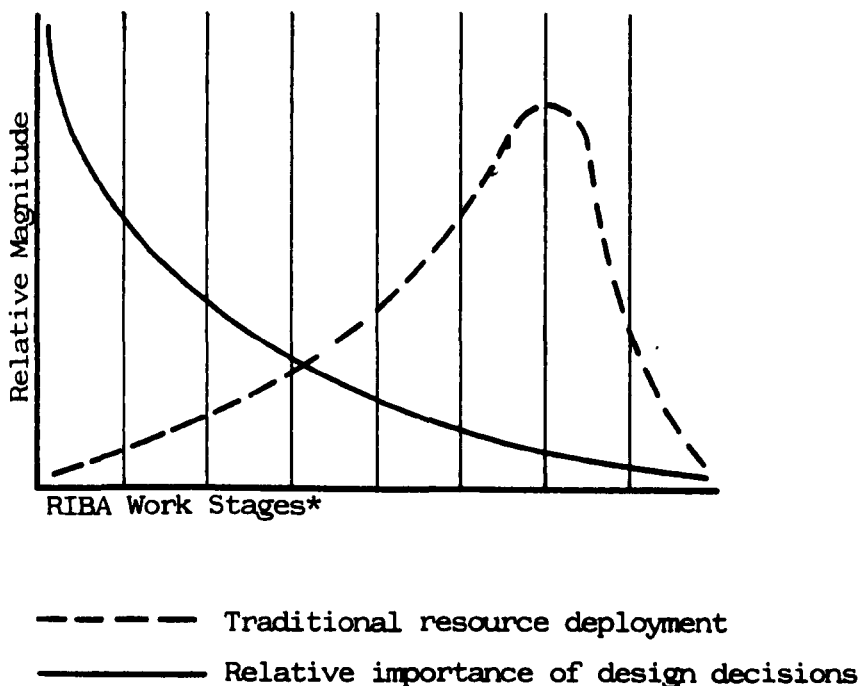
From the frequent references to liason with both client and other members of the design and construction team, an emphasis is placed upon communication. The subject being communicated is the building description which is steadily evolved under the guidance of the architect through design stages C to F. It will be noted the major determinants of both the cost and time-scale of the project are initiated by the decisions made during work stage C, that of the outline proposals.

## 2.2 WORK STAGE C

Since the major content of the architects' workload is the follow through from the decisions made during the formation of the outline design proposals, it would seem that this particular process would entail the major part of the architects design resources. This does not appear to be the case;

"crucially important initial design stages go relatively unresearched and the first proposal which appears to work is all too often seized upon." 3

This observation is illustrated by a corroborating diagram produced by Kraal<sup>4</sup>, which relates the usual deployment of the designers resources to the importance of design decisions through the work stages



\* A number of RIBA sources summarise the architects service as eight work stages, A to H.

Figure 2.1 Schematic Diagram Relating The Deployment of Resources To The Importance of Design Decisions. 4

## 2.3 EFFORT DISTRIBUTION THROUGH WORK STAGES

"Few attempts have been made to study the architectural design process..., monitoring design at the simplest level (which) entails observing the activities of design: how much time is spent thinking, drawing, looking for information." 5

Of these few, one such study undertaken using the technique of 'time-lapse' photography, records the distribution of time between the various tasks carried out by designers during their design process.<sup>6</sup> The pilot study compared the time distributions at an early and advanced stage in formulation of a design. These results are tabulated below in Table 2.1 below.

Early Design Stage

Task Description	Time Spent As %age Of Whole
writing and annotating	2
drawing	56
reviewing	14
measuring	5
other	23
	<u>100</u>

Advanced Design Stage

referencing	23
writing and annotating	14
drawing	17
measuring	4
ammendment	3
reviewing	18
other	21
	<u>100</u>

These results tend to illustrate that the time spent in reviewing, measuring and other activities stay fairly constant, whereas the major effort devoted to drawing during the early design stage gives way to the increasing requirement for referencing and annotation as the design progresses.

The diminution of the drawing effort would appear to be an argument against the adoption of computer aided draughting

systems, since the major draughting effort occurs at the formative design stage (56%), the stage at which it is notoriously awkward to describe a building to a computer with any degree of precision.

### 3 The CAD Spectrum

Lansdown<sup>7</sup> classifies the range of computer aided design aids to architects in a machine-centred manner. His validation of this classification method runs as follows:

"Because of the diverse and fragmented nature of the industry, its computing needs have necessarily been met by a wide variety of different program types, such as programmes for:

management, payroll and accounting;  
quantities and stock control;  
time scheduling and manpower  
allocation;  
design; and  
draughting.

Although all such programs perform some calculation, in many cases this is not their major function, and it is useful to sub-divide the programs into five broad but perhaps overlapping categories:

- 1 Programs in which calculation plays a major role.  
eg. dynamic thermal performance.  
beam, slab and column design.
- 2 Programs in which manipulation of text plays a major role.  
eg. word-processing.  
specification writing.
- 3 Programs in which graphical manipulation plays a major role.  
eg. draughting.  
perspective drawing.
- 4 Programs in which database creation and information retrieval plays a major role.  
eg. design.  
draughting.  
stock control and quantities.
- 5 Programs in which logical operations play a major role.  
eg. design.  
management and control." 7



This analysis makes clear the difference in the adoption by the architectural profession between microprocessors and integrated CAAD systems; whereas microprocessors\* can support those tasks in calculation, manipulation, and increasingly database manipulation predominates, they perform poorly with software for processing graphic and logical operations. Certainly the progress of technology will eliminate this difference in due course.

As will be shown in chapter 4, considerations of the problem inherent in the description of graphic representations of form means that, with the current state of the technology, computer graphics have to be dealt with as a data-structuring problem.

Davison<sup>8</sup> categorises the types of computer aids available to designers in a more 'task' oriented manner. These categories are:

#### "1 Design Generators

The purpose of design generators is to synthesise design solutions from statements of the design problem...The design process generally involves a large number of, often conflicting, requirements which define an extensive set of solutions, from within which a solution of best fit must be coaxed.

#### 2 Appraisal

Instead of looking to the computer to generate solutions, the calculating and information retrieval power (is) applied to appraising design solutions generated by the architect. Based upon the results of the appraisal the experienced designer would seem to amend his previous solution and submit the scheme again to the computer for re-appraisal and comparison against the previous solution.

\* 'Microprocessor' is used by the writer to mean 8-bit word length computer processors, with an internal clock cycle of less than 10 000 cycles per second.

### 3 Visualisation

Although conventional manual methods employ geometric constructions, the problems of projective drawing are inherently those of coordinate geometry mathematics, and as such lend themselves directly to the calculating power of computers.

### 4 Office Management

### 5 Draughting." 8

Davison goes on to argue that any activity carried out by a computer which increases the attention which the designer is able to devote to the creative process of design, constitutes a tool of CAAD. Whilst this is an alluring view, it ought to be tempered by considerations of appropriate technology which are presented in chapter 3.

Moreover, tasks such as payroll, accounting and word-processing are well tried and tested techniques which have a peripheral bearing on the problem of design per-se.

## 4 The Design Process

In the preceding section the 'architectural process' as a client service was described. It is unfortunately confusing that the term 'design process' is used ambiguously to describe both the architectural process including peripheral tasks, and the actual process of the derivation and description of a hypothetical building form proposed as a solution to the design brief.

In the remainder of this thesis, the latter interpretation is used.

In an effort to rationalise this 'ethereal' process, early design method researchers looked to Operations Research methods in an attempt to derive an acceptable design morphology. These methods tended to relegate the role of the designers' creative input, since they tended to concentrate upon techniques of optimisation applied to defined goals. Such a view of design was unpopular with practising designers who rejected such a mechanistic

view.<sup>9</sup>

With the benefit of hindsight, design methodologists are generally agreed that the 'architectural design process' is not sequential, having neither a discrete beginning or end, but is more in the nature of a continuum; that it can be indefinitely recursive and iteratively modifying; and that a learning cycle is initiated which increases the designer's understanding of the problem.<sup>10,11,12</sup>

Moreover, design methodologists have to overcome a historical antipathy to their approach because,

"the design process was influenced by the behaviour of the designer who often placed himself under the aureole of the artist and therefore was not amenable to the systematic approaches to design problems or to a scientific analysis of the design process." 13

Nevertheless, a great deal of investigation has been carried out along these lines.<sup>14</sup> The two design taxonomies currently proposed are reviewed in the following sections. These merit attention since they can be viewed in terms of 'the design of a design process'. The reader is directed elsewhere for details of historic design taxonomy descriptions.<sup>9,15</sup>

#### 4.1 THE MARKUS/MAVER MODEL OF DESIGN ACTIVITY

The Markus/Maver model of design activity has been used to explain the role of appraisal in the design process.<sup>16</sup>

The sequence of the design process is illustrated in Figure 2.2 overpage, and represents the flow of activity through the various RIBA Work Stages. Each work stage is represented as a localised activity flow comprising of analysis, synthesis and appraisal followed by a decision. The salient feature of these local activity flows, are the feedback loops between synthesis and appraisal.

##### 1 Analysis

The 'analysis' stage consists of the collection

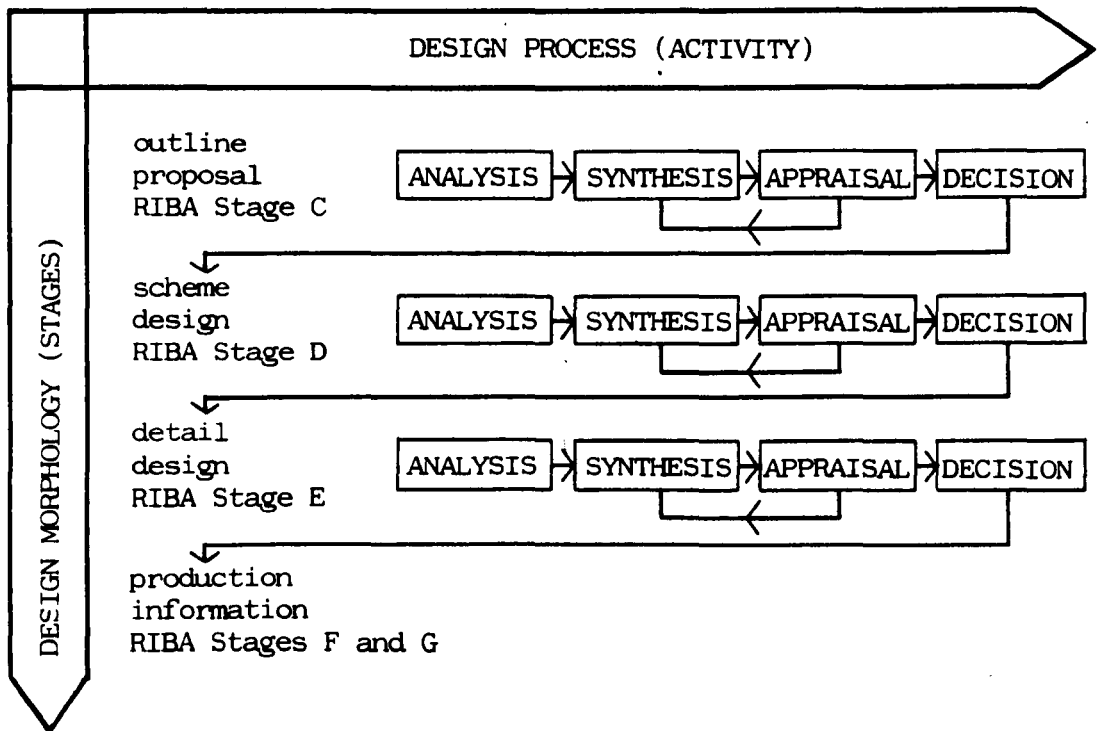


Figure 2.2 Schematic Representation Of The Markus/Maver Model Of The Design Process 16

of the salient design parameters and associated data pertaining to the design problem as it is construed at the current work stage by the designer.

## 2 Synthesis

In this phase, the various contending parameters are associated and given priorities by the designer. Having thus defined the problem, the designer postulates a design solution which he hypothesises will satisfy the criteria now set.

## 3 Appraisal

The postulated solution is tested to see if it fulfils the designers expectations and actually does satisfy the criteria set. The appraisal is not restricted to quantifiable parameters alone, but the designer also has to exercise subjective assessments too.

#### 4 Decision

The schematic representation shown in Figure 2.2 would indicate that the decision following the satisfactory completion of the appraisal loop is whether to proceed to the next stage or not. It is likely that it is intended to show that following the appraisal loop, a decision has to be made whether or not the result of the appraisal is satisfactory or not. If the result is unsatisfactory, then the proposal must be modified and re-submitted for appraisal. Only when the predicted performance is satisfactory does the designer move on to the next work stage.

The essential characteristic of the model is the effect of the appraisal. An unsatisfactory solution identifies the deficiencies in the proposed design. Not only does this give rise to alterations, but can also identify where knowledge is lacking, where further data is required, and also sub-problems which will require solution prior to proceeding with the further work stages.

Another salient feature to be recognised is that an unsatisfactory appraisal at any one work stage can lead to backtracking through preceding work stages, which although implicit is not directly shown in the schematic diagram.

Recognising perhaps the oversimplification of this design morphology, the coauthor of the description is careful to put the emphasis on appraisal;

"The crucial issue in the use of such a methodology is...the nature of the evaluative activity and how the outcome of the evaluation influences modifications of the design hypothesis. At the finish, and by definition, design decision making involves subjective value judgement. Having squarely acknowledged this fact it is important to state that subjective value judgement, by whomsoever, must be made from the best possible explicit and objective information base." 17

There is also the warning that methods and tools can be misused, no matter how sophisticated they might be. This is particularly true of computer aids in architecture and should be borne in mind in the ensuing pages.

#### 4.2 POPPER'S CONJECTURE AND REFUTATION

"The emerging third generation view (of design morphologies) is that inputs from the designer to the design process cannot be avoided, and are a necessary part of any design method. This view is usually justified by reference to Popper's 'conjectures and refutations' model of scientific method.

Translated to the design field, this model is attractive because it fits in well with what designers already do in practice." 18

In a translation of Popper's 'Conjectures and Refutations'<sup>19</sup> the architect hypothesises a conjectural solution of built form as a potential answer to the criteria set in the design problem. This hypothesised solution invites criticisms in order to refute that this is so. The attempted refutation can draw on sources beyond those specified in the brief itself, and make reference to subjective, as well as purely objective, analyses. Moreover, refutations and criticisms from laymen are just as valid as those from architects.

The procedure is always open ended, since a satisfactory design solution is always open to refutations founded on knowledge not available at the time of conjecture. All solutions are the best that could be envisaged at the time.

To paraphrase Popper by substituting 'design method' for 'scientific method', Popper enumerates several observations of a general nature relating to the growth of a 'design solution':

"1                    There are no ultimate sources of design. Every source, every suggestion is welcome; and every source, every suggestion is open to critical examination.

- 2           The proper question is not one of sources; rather we ask whether the solution suggested is true. We try to find this out as well as we can by examining or testing the solution itself; either in a direct way, or by examining or testing its consequences.
- 3           In connection with this examination, all kinds of arguments may be relevant. A typical procedure is to examine whether our theories are consistent with our observations.
- 4           Quantitatively and qualitatively by far the most important source of our design - apart from inborn flair - is tradition. Most things we know we have learnt by example, by being told, by learning how to criticise, how to take and accept criticism, how to respect truth.
- 5           Every bit of our traditional knowledge...is open to critical examination and may be overthrown.
- 6           Design cannot start from nothing. The advance of design consists mainly, in the modification of earlier designs.
- 7           Every solution to a problem raises new unsolved problems; the more so the deeper the original problem and the bolder its solution." 19

Accepting the substitution of 'design' for 'knowledge' in the above quotation, we have a widely acceptable description of what indeed happens in practice. The designer can proceed from any point which he considers useful; the major determinant of a proposed solution is whether or not it works; does it work consistently well; and design solutions can never be universally valid since the criteria we use to judge it is limited by our current knowledge. It follows then, that design is indeterminate, in which the 'solution space' is continually curtailed by the efficiency with which the solution 'works'.

Despite criticism,

"Popper's account of scientific method has proved astonishingly popular with lay audiences, and seems to have been largely

accepted as 'the picture' of what science is about." 20

#### 4.3 A HELICAL MODEL

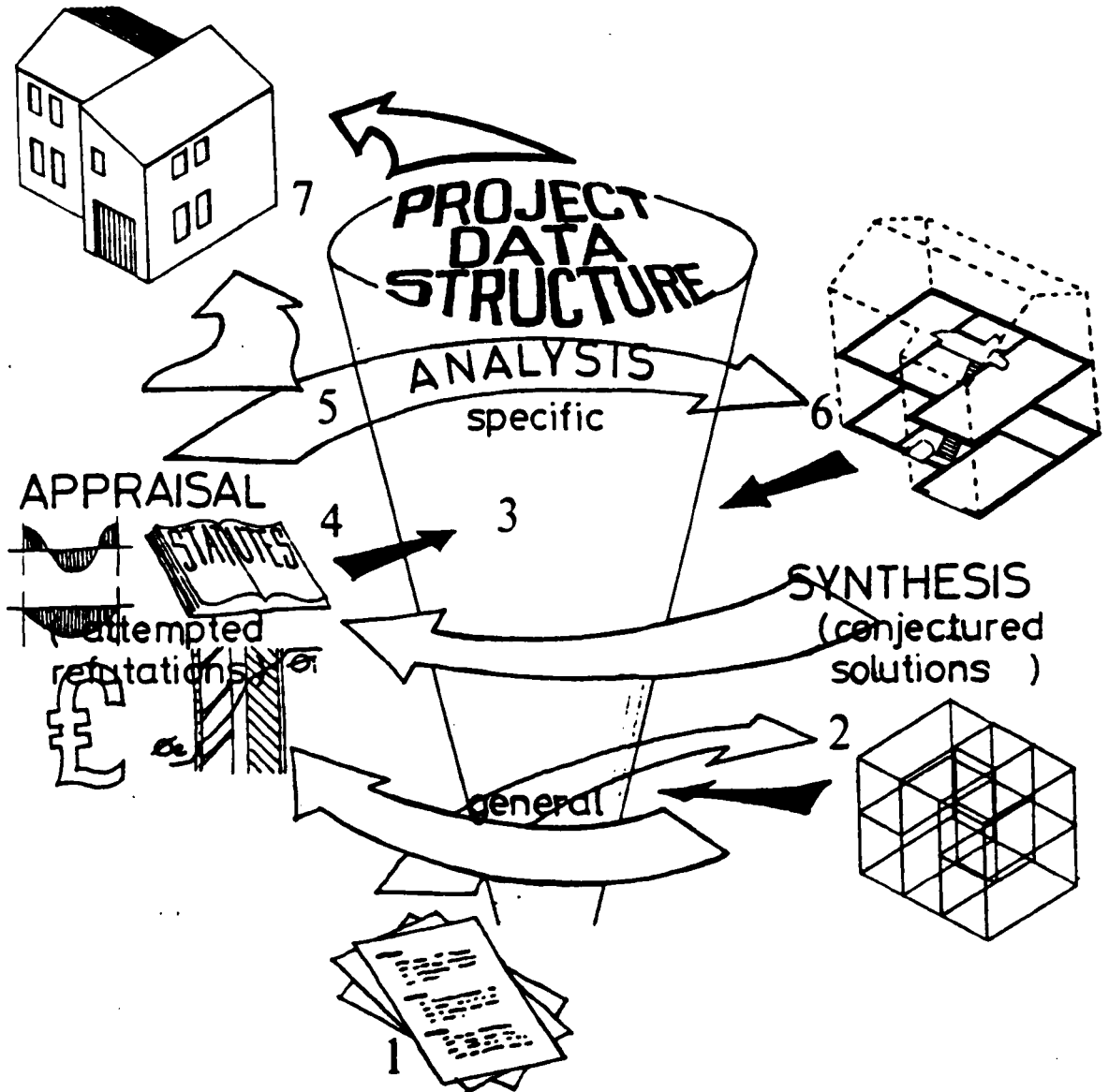


Figure 2.3 A Helical Model Of Design Activity  
Combining Both The Markus/Maver,  
And Popper Models.



Both the Markus/Maver and Popper models of design activity can be accounted for by using the analogy of an inverted helical cone, as illustrated in Figure 2.3 on the preceding page.

Underlying this analogy is the following rationale:

"Each project can be viewed as proceeding by the performance of various functions, each marking the achievement of some identifiable goal. Performance of each function requires the execution of some design procedure which requires certain data as input, produces certain data as output, and consumes certain resources. As a design project progresses, the outputs from the procedures accumulate, and an extensive, complex project database is built up.

The project is complete when this database contains a sufficiently complete, consistent and detailed description of the proposed building to form a basis for a contract for actual construction work." 21

Whilst in practice the 'various functions' referred to are rarely discrete, as the above would suggest, two fundamental concepts have been introduced; first, that the prosecution of the design process generates a database, and second, that the process is completed when the database holds adequate detail from which the real building can be described in detail comprehensive enough to actually be realised by third parties. The detail contained in the data base must pre-empt all of the awkward problems likely to be encountered by these third parties, whether these problems be in the actual construction, or whether the building might suffer from condensation.

Returning to Figure 2.3, the design process is depicted as stemming from the design brief (1), which is a synopsis of the client's requirements which forms the starting data of the project database. Along with the stated requirements are a multitude of implicit requirements; eg. structural stability, statutory conformity, aesthetic acceptance etc.

From consideration of what is known, (site conditions etc.),

and of the problems posed by the design brief, the designer at some stage envisages an 'ethereal' form of a potential building which he conjectures as a solution (2). No matter how 'fuzzy' the conceived form might be, it adds, speculatively, to the content of the database (3) which is slowly evolving. At this stage the design is 'general', relating to the project as a whole, and decisions fixed at this point have serious knock-on effects upon subsequent development. This drawing together of the disparate parameters which form the 'design decision space', represents the analysis of the problem.

The designer can begin to speculate on the likely inter-relationships between the parameters, assigning them relative priorities. Having thus specified the subordinate problems within the overall, using the data resident in the database and including new information from without, he is able to subject his proposed design solution to testing (4). Whilst many parameters can be tested on a binary basis, eg. planning requirements and building regulation compliance, many of the parameters appraised will indicate performance ranges. This is in part due to the 'fuzziness' of the building description with which he is working, as well as the absence of detailed information which he has yet to provide.

The external data drawn in, in order to carry out the tests, as well as the results, all add to the database. Additionally, the testing procedures also indicate the deficiencies in the database to the designer. This synthesis leads to an increased awareness of the design problem on the part of the designer, as well as verifying which of the design parameters are critical (5).

Consideration of the results of the appraisal lead to modifications of the design hypothesis, as well as the extension of the data to encompass new areas of analysis (6). The modified design solution is then resubjected to the appraisals (4) in the attempt to pin-point the deficiencies in the design. This results in increasingly accurate predictions of performance, and if the design process is progressing positively, performance ranges satisfying those specified in the design brief.

Eventually the performance range of the predictions prove satisfactory. At this point, the design hypothesis from which the performance characteristics were derived, is said to be fully complete and consistent.(7). All that remains is to ensure that the description is adequate to describe the building to the third parties who will be involved in the construction.

Note that the architects function is to create the project data base in its entirety. The constant synthesis-appraisal loop allows the work and information flowing from other members of the design team to be integrated in an orderly fashion, and indeed, these specialist consultants work could be described as specific analysis-synthesis-appraisal loops 'orbiting' around those of the architect.

#### 4.4 SUMMARY

The architects' function is to conceptualise and communicate a hypothetical building design.

The design is arrived at through a process of proposing potential solutions, and then trying to demonstrate that they fail to satisfy the design problem.

As the design process continues, general and strategic problems become more specific and tactical.

The designers' solution space is increasingly constrained as the database is enlarged.

Appraisal is both objective and subjective. The solution eventually decided upon will be the compromise between the various contending design parameters, which results in the best predicted performance.

The quality of a design will improve with the rigour, range and depth of the tests to which it is subjected during its creation.

References

- 1 Working With Your Architect  
RIBA  
RIBA Publications 1971 p.5
- 2 Architect's Appointment  
RIBA/RIAS  
RIBA Publications Jun 1982 pp.3-8
- 3 The Design Of Complex Buildings Using An Integrated CAD System  
E.M.Jones  
CAD82 Proceedings  
Butterworths Mar 1982 p.355
- 4 Implementation Experiences Of Design Applications  
L.Kraal  
PArc83 Proceedings  
On-Line Conferences Oct 1983 p.69
- 5 Approaches To Computer Aided Architectural Sketch Design  
D.S.Willey  
Computer Aided Design Vol.8 No.3 Jul 1976 p.181
- 6 An Investigation Of Design Activities Using Analytic Time-Lapse  
Photography.  
C.R.Evans, P.A.Purcell, and J.Wood  
National Physical Laboratory Jul 1971
- 7 Expert Systems: Their Impact On The Construction Industry  
Report To The RIBA Conference Fund  
John Lansdown 1982 pp.3-4
- 8 Computer Aided Draughting - A means To An End  
J.A.Davison  
Paper Presented at The Institute of Advanced Architectural Studies,  
York.  
GMW Computers Ltd, Berkhamsted Nov 1981
- 9 The Coming Of Post-Industrial Design  
Nigel Cross  
Design Studies Vol.2 No.1 Jan 1981 pp.3-8
- 10 The Architecture Machine  
Nicholas Negroponte  
Computer Aided Design Vol.7 No.3 Jul 1975 pp.190-195
- 11 Analogical Reasoning In The Design Process  
N.V.Findler  
Design Studies Vol.2 No.1 Jan 1982 pp.45-51

- 12 Integrated CAAD Systems  
Aart Bijl, David Stone and David Rosenthal  
Final Report to DoE on Funded Research Project 470/12  
EdCAAD, University Of Edinburgh Mar 1979
- 13 Cybernetic Model Of The Design Process  
J.H.A.E.Amkreutz  
Computer Aided Design Vol.8 No.3 Jul 1976 p.187
- 14 Externalising The Design Process  
D.Kernohan  
Design Studies Vol.2 No.1 Jan 1981 pp.27-32
- 15 Synopses Of Recent Published Studies Of The Design Process  
S.Lera  
Design Studies Vol.4 No.2 Apl 1983 pp.133-140
- 16 A Theory Of Architectural Design In Which The Role Of The Computer  
Is Identified  
Tom W.Maver  
Building Science Vol.4 1970 pp.199-207
- 17 The Concept Of Modelling In Architectural Design  
Tom W.Maver  
Design Methods And Theories Vol.13 No.3/4 1979 p.174
- 18 The Primary Generator And The Design Process  
J.Darke  
Design Studies Vol.1 No.1 1979  
quoted by N.Cross in reference 9
- 19 Conjectures And Refutations  
Karl Popper  
Routledge and Keegan Paul 1963
- 20 Knowledge Without Authority  
Karl Popper  
A Pocket Popper Ed. D.Miller  
Fontana 1983 pp.54-55
- 21 Design Method And Scientific Method  
Nigel Cross, J.Naughton and D.Walker  
Design Studies Vol.2 No.4 Oct 1981 p.195
- 22 Computer Aided Architectural Design  
W.J.Mitchel  
Petrocelli/Charter N.Y. 1977

## Chapter 3

### Computer Technology

1	Introduction	3.1	30
2	Resource Requirements For CAAD Systems	3.1	30
	2.1 Finance	3.1	30
	2.2 Effects Of CAAD Upon Work Practice	3.3	32
	2.3 Staffing And Expertise	3.4	33
	2.4 Objectivity Of Systems	3.6	35
	2.5 The Role Of Drawing	3.7	36
	2.6 Draughting Systems	3.8	37
3	The Current State Of CAAD Technology	3.9	38
	3.1 Recent Developments In Computer Technology	3.10	39
4	Computer Hardware	3.11	40
	4.1 Display Screens	3.11	40
	4.2 Graphics Input Devices	3.13	42
	1 Input Via The QWERTY Keyboard	3.13	42
	2 Stepping Keys	3.13	42
	3 Thumbwheels	3.14	43
	4 Joystick	3.14	43
	5 Trackerball	3.15	44
	6 The 'Mouse'	3.15	44
	7 Flat Bed Digitiser/Plotter	3.15	44
	8 Touch Sensitive Screen	3.16	45
	9 Light Pen	3.16	45
	10 Accoustic Pen	3.17	46
	11 Digitising Tablet		
	a) Point Mode	3.17	46
	b) Continuous Mode	3.18	47
	4.3 Physical Models	3.20	49
	4.4 Summary	3.21	50
5	Computer Software	3.22	51
6	The Man-Machine Interface	3.26	55
7	Summary	3.29	58
8	Conclusion	3.30	59
	References	3.31	60

## CHAPTER 3

1 Introduction

There are many suggested reasons proffered to explain the reluctance of architects in small private practices to adopt CAAD aids, and this chapter discusses some of the more general state-of-the-art factors which act as dis-incentives.

The first section looks at those factors which have a direct bearing upon architects' practices. These include, obviously, the straightforward resource requirements coupled with the paucity of cost/benefit information with which to assess a systems viability, and the less obvious changes which occur in offices which employ such aids.

The second section reviews the limitations inherent in the current state of computer equipment; this includes the hardware devices through which architects communicate with graphic data, the rigidity of software systems, and the mismatch of the interface through which the hardware and software are linked between the desires of the user and the needs of the machine, in order to implement the systems.

Finally, these factors are summarised, and conclusions drawn.

2 Resource Requirements For CAAD Systems

## 2.1 FINANCE

Currently, CAAD systems are beyond the means of most small architects practices. Chapter 1 described the composition of the architectural profession, and it was seen that the majority of private practices employ two to three architectural staff with little available investment capital. Computer draughting is probably the most widely utilised computer aid which manipulates graphic information. Although these systems tend to be expensive, it is relatively easy to balance the required expenditure against probable savings in draughtsmens salaries. Indeed, it has been

known for a firm of architects to purchase a CAD system on the strength of one particular commission.<sup>1</sup>

For CAAD systems, as opposed to computer aided draughting, an investment figure of around £40,000 is typically cited for the purchase of a system comprising of processor, input and output devices and the system software\*, added to which is an average of 10% of the capital cost necessary for maintenance, as an annual overhead. Although options of hire, lease, hire-purchase and lease purchase are available, the financial viability of acquiring a system is difficult to assess, particularly when a case has to be presented in order to borrow the finance from institutions.<sup>2</sup>

Moreover,

"payback periods are very difficult to specify, but might average about 3 or 4 years; periods from 2 to 10 have been quoted." 3

In assessing the cost-effectiveness of a system, the usage that is made of the system will be a major determinant, and as we have seen in chapter 1, the continuity of the workload of most small practices would make such predictions tenuous. Many projects handled by small practices might be better dealt with by conventional methods, yet the pressure once having invested in a system to use it continuously, can induce practices to mount unprofitable projects on the machine. Projects incorporating a high proportion of repeats are more amenable to efficient computer usage than those that are 'one-offs'. Similarly, those on 'green-field' or 'open-desert' sites can be more easily dealt with on computer than those on infill sites.

As yet there is little practical experience of CAAD usage by small architects practices, consequently there have been few published studies through which architects can accurately assess the financial viability of adopting a CAAD system. A salutary

\* The GABLE system; comprising of a Tektronix 4907 File Manager, Tektronix 4662 A2 sized plotter, Tektronix 4051 desk-top graphic processor with joystick cursor control, and applications program.



statement on the cost effectiveness of computer systems is given by one user:

"recovery of development costs is dependent upon a high value being attached to improved design." 4

## 2.2 EFFECTS OF CAAD UPON WORK PRACTICE

Currently all CAAD systems represent part solutions to the design problem. These systems carry out specific tasks for which a set minimum of consistent input information must be made available in order to produce the output information, which is set in content and format. It might not produce the output in the form most convenient to the designer. This rigidity is out of accord with the flexibility offered by existing design practices, although it must be noted that such aids permit potential design solutions to be appraised to a depth and scope hitherto impractical. Systems tend to predetermine the mental processes the designer takes;

"the creative and subjective nature of design means that there is no 'right' or determinate answers to design problems. The needs of one designer are not often met by the tools which meet the needs of another designer, or even the same designer in a different situation or a different occasion. The more sophisticated the system, the more this is so." 5

Thus there is a tendency for systems to impose a set approach to design problems whenever the use of CAAD aids are contemplated. Whilst this imposed organisation might be beneficial in some cases, it might equally prove counterproductive in others.

The pressure to utilise a system once the investment has been made has already been noted. This pressure can result in an unhealthy emphasis being placed upon those tasks that the computer can perform to the prejudice of the importance attributed to those it cannot. Almost inscrutably the functions that the system can perform can become an end in themselves, rather than a means. The

consequent distortion can disturb the way in which the practice is run, particularly when people begin to see their role change from designers to that of 'machine feeders'. The smaller the office, the more pronounced this distortion is likely to be.

### 2.3 STAFFING AND EXPERTISE

Computer systems are rarely understandable to the end users who tend to consider them as 'black boxes'. This is understandable since,

"conventional programming techniques in use for CAAD can be described as imperative or prescriptive. Every step that the machine has to take has to be spelled out in stupifying detail." 6

Thus the writing of CAAD programs has in the past been undertaken by computing professionals who rarely have an appreciation of the architects problems. Their expertise lies in the areas of coding and systems analysis.

"The days when computers were used exclusively by an elite band of scientists have long gone. Unfortunately many computer specialists still do not realise this and still write programs that can only be run by users in the 'know', who can cope with the idiosyncracies of conventional programs." 7

In consequence,

"to operate these software tools, specialist staff are required. When programming CAAD applications however, other specialist skills are also required based on knowledge of design practices. It is difficult to get both skills to bear on CAAD problems satisfactorily; not least is the difficulty of avoiding interdisciplinary conflicts among staff concerned with their separate status and career prospects." 6

This also holds true within design practices themselves, where the situation is exacerbated by the salary differential between computer literate, and non-computer literate staff, who

primarily produce the same end results, albeit through differing processes.

It is difficult for the small practice lacking the necessary resident skills to 'buy-in' specialised and experienced computer literate personnel. In addition to the potential hostilities aroused, the consequent dependence upon these key personnel once employed pose a threat to the career structure of other designers. The capital investment in machinery in relation to the practices' turnover, give the computing staff undue sway in practice decisions if the principals lack sufficient understanding to be able to discuss the computing needs of the office on a parity with the operatives. This feeling of dependence upon the specialist staff can lead to a feeling of loss of control of the practice on the part of the principal and others; a feeling that their control is being usurped by technologists upon whose work the practice depends, and whose jargon they don't understand. Furthermore, there is a latent hostility to the computers 'objectivity' by designers trained with an emphasis on subjective assessment.

Staff turnover can be unusually disruptive to a small practice. Replacement staff have long lead-in times if they are required to familiarise themselves with new systems.

" Use (of CAAD systems) is most effectively accomplished by a specialist group of users. However, vulnerability to the turnover of key staff is such that adequate cover is essential." 4

The inference of the statement of a system manager is clear: The investment in specialist staff has to be sufficient to provide depth.

Due to the small number of CAAD systems in use, there is a lack of experienced personnel available. There is a consequent tendency to poach staff from existing CAAD using practices, which enhances the vulnerability of practices to the lose of staff. This applies to managerial staff as well as computer operating staff. A system badly managed can be dramatically expensive for the practice.

## 2.4 OBJECTIVITY OF SYSTEMS

Computers may be described as machines that can,

"perform logical (mathematical) operations on given information without questioning the validity of the operations. The quality that puts the modern computer in a class by itself is its automatic operation. Consequently it is able to operate by itself; it can perform a long sequence of related operations with the original information without the need for human intervention or assistance. Moreover, if so instructed by the 'program', the computer can select from alternative courses of action on a logical basis, ie. they simply compute the implications of the given data and instructions according to a fixed set of rules." 8

The alienation of the lay public from computers is a well known phenomenon. The fact that the machine questions neither the validity of data nor processes is contrary to human behaviour, hence it is disturbing. Without knowing what is going on in the 'black-box', the user is expected to have faith in the subjective overview which the programmer has encapsulated in the program. In the face of the aura projected by these machines, there is a reluctance for users to question the basis by which the results are produced by the system. Similarly, there is a reluctance to alter data which has been painstakingly fed into the system in the light of adverse results, leading to minor alterations to existing solutions, rather than a radical rethink.

"Many large systems now in use have been built up cumulatively, so that no-one knows all the rules by which they are operating. 'Decisions' may be made by computers whose programmes nobody can any longer take responsibility for." 9

As yet there has been no legal precedent set over who might be responsible in the event of decisions being taken as a result of erroneous programmes. Indeed, even well established programs used by practice for some years, still act in ways unforeseen by

the program author.

The lack of confidence on the part of the lay user is, if lamentable, at least understandable. This is particularly so where designers, trained to extract the best subjective balance from a host of competing objective considerations, are concerned. In this particular environment,

"the test of the machine is the satisfaction it gives you. There isn't any other test. If the machine produces tranquility it's right. If it disturbs you it's wrong until either the machine or your mind is changed." 10

## 2.5 THE ROLE OF DRAWING

The visualisation processes which take place in the designers mind as the various design parameters coalesce to give rise to an initial design solution, have been claimed to be the province of perceptual psychology. However, it is apparent from studies of the design activity that a large proportion of a designers time during the early design stage is involved with drawing, (see Chapter 2, section 2.3). Conventionally then, designers use drawings to manipulate their vague ideas about building layout and form. Drawings begin to establish limits and boundaries to these tentative ideas, and enable the designer to test how the various design parameters interact. In this light, a drawing can convey a wealth of positional, spatial and locational information simultaneously, and at many levels of significance.

Drawings also have further functions not related to the designers creation of designs. They have other major roles:

'We are of the view that in organisations (typical of which is our own), drawings in conventional form will, for many years yet continue to perform three vital functions:

- a defining the customer's requirements by means of layout drawings;
- b defining the details of what is to be made to the various sources of manufacture;

- c defining the manner in which detailed parts are assembled." 11

Drawings are used for record purposes, contract documents, and concise means of description, hence of continuing importance to architectural practices.

During the early design stages, drawings tend to be in the form of sketches; crude representations, generally of plan forms<sup>12</sup> are manipulated and modified.

## 2.6 DRAUGHTING SYSTEMS

Draughting systems differ from integrated CAAD systems in that the former model drawings, whilst the latter model to some extent the buildings which the drawings represent.

"In conventional draughting a drawing is built up as a collection of lines and points, each line being drawn separately by the draughtsman. A minor exception to this being the use of (dry transfer) lettering or similar techniques... Thus one major problem in trying to get to grips with computer draughting is the lack of match between traditional methods of drawing and the solutions offered by computers." 12

Draughting systems are orientated to a very specific task, although recent versions of the more widely used draughting systems now offer the capabilities of carrying out some appraisals. Component based systems have always offered the facility of automated scheduling. The specific task referred to above, is that of recording the position of lines and instances of objects in a coordinate space. In respect of this overview, these systems may be termed 'dumb' in that they merely record simple data, and do not associate it into more sophisticated structures which could then be manipulated by the user at a higher level. Dumb drawing systems

'produce drawings by following instructions from a user on where to place lines on

paper. Intelligent (CAAD) systems are, in addition, able to gather information from the arrangement of the lines to describe the real-world objects which are depicted in the drawings.. Drawing systems generally operate by aggregating detailed bits of drawing into completed whole drawings. This is evident in the use of libraries of symbols, sub-pictures or components to assemble drawings." 5

Dumb drawing systems are awkward to use for generating designs upon, unless the design is being built up from a catalogue of standardised components existent in the systems' libraries. They tend to be used 'post facto' to the outline design stage when the most far reaching decisions have already been taken. The use of these systems does not lead to a greater understanding of the design problem on the part of the designer, nor do they appear to contribute to improved quality of design. Intelligent systems on the other hand, do. For this reason, this thesis is concerned with graphics used by designers to describe buildings to intelligent systems, since it is only through increased awareness of the design problem that better qualitative design will be achieved.

### 3 The Current State Of CAAD Technology

The central issue for intelligent CAAD systems is the description of buildings to computers through the medium of drawings.<sup>13</sup> 'Pencil and paper' techniques evolved to cope with the subtle complexities of the design problem, and are tools for the manipulation of spatial, proportional and relational ideas pertaining to an embryonic evolving building form. These techniques are regarded as the vehicle for allowing the designer a wide range of possible approaches to the design problem by presenting simultaneously a range of visual cues to complex relationships<sup>14</sup>, where simple symbols possess many levels of significant information to the designer, where they encode a building form into manageable pieces of information<sup>15</sup>, and where sketchiness of a drawing contains important information about the

designers train of thought<sup>16</sup>. Significantly for intelligent systems,  
 "such a drawing cannot be easily decoded  
 by anyone other than the designer." 14

### 3.1 RECENT DEVELOPMENTS IN COMPUTER TECHNOLOGY

It is only relatively recently that applications software utilising logical programming techniques have become available. These applications programs are limited in the CAAD field, and although first reported in the mid 1970's, it was only with the advent of SeeLog<sup>17</sup> that the beginnings of a graphics interface using the PROLOG language<sup>18</sup> became available. These systems, generally known as 'expert systems', or 'knowledge engineering' systems, are in their infancy. As yet, those applications programs that do exist would appear to answer questions about the data resident in the computer, but still relies on data being manually entered through the computer keyboard. Consequently, until these systems can be approached through a graphics interface, their relevance to CAAD is limited. SeeLog became operational in August 1982, and is implemented on a VAX computer with the UNIX operating system. These facilities are unavailable at the Scott Sutherland School of Architecture. Moreover,

"SeeLog is at an early stage of development and relies on the rather inadequate message passing facilities of Berkley 4.1 UNIX. Mysterious disasters are known to have occurred." 17

Another recent development has been the construction of parallel processors, able to manage some 800 megaflops (million floating point operations per second). This computing power has only recently been applied to graphical applications, using linked registers to take note of what is happening to an adjacent pixel, which affects the current state of the pixel under scrutiny.<sup>19,20</sup> Doubtlessly, when parallel processors are used in conjunction with logical programming techniques, a new direction in many aspects of computing will be reached, as indeed recognised by the Japanese 'Fifth Generation' computing development programme.



## 4 COMPUTER HARDWARE

Electronic computers are in the process of becoming smaller, faster, cheaper and even more reliable than at present; the rate of improvement would appear to be exponential.

The actual computer processors available today do not bear scrutiny within this thesis; it is assumed that the reader knows basically what a computer is, and has a rudimentary knowledge about how they work. What is fundamental to this study, however, is the range and variety of hardware devices through which the user is able to communicate graphic data with the machine. The first consideration is of display screen types through which the user actually views the data. The second is the types of graphic data input devices, covering keys, cursor drivers and digitizers. Finally, the use of physical models to input geometric data is considered.

### 4.1 Display Screens

There are two types of display screen, vector and raster. The first type 'generates' lines and arcs between points as defined by the user, by means of an analogue generator linked to the deflection plates of the electron gun of the cathode ray tube. A raster display works like a normal television screen. The screen is traversed by the cathode ray fifty times per second, over the 625 striations into which the screen is laterally divided. The electron beams intensity gives tonal variation to the spot over which it passes. In computer implementations of raster technology the screen is divided into pixels, ie. atomic cellular subdivisions each of which may be addressed by the electron beam. This results in a granular appearance to a displayed picture.

All multi-colour display screens are based upon raster technology, although 'simulated' colour may be achieved by multi-coated phosphor layers used in vector displays.

The duration of the fluorescence of the phosphor once it has been excited by the electron beam gives rise to storage and refresh

screens. In the former, the fluorescence is of long duration, consequently the picture does not need to be overdrawn as often as the latter. However, it does mean that alterations to a picture either have to be overdrawn, or if possible, deleted lines 'undrawn'. Pictures undergoing alteration soon end up a mess. The shorter duration phosphors have the advantage of permitting instant removal of lines as a picture is altered. However, due to the speed of fading of the image, there is a maximum length of line which one can project upon a refresh screen before encountering 'flicker'.

"Raster screens at an equivalent price to either refresh or storage screens...do not offer an acceptable line quality...This problem is particularly noticeable on curves." 21

It is not only curves, but also sloping lines which are stepped on raster screens; the effect being more pronounced the nearer the lines are to the horizontal or vertical directions. To some extent these effects can be ameliorated by a technique known as 'aliasing', where the intensity of the line segments near to the step are varigated. This is illustrated in Figure 3.1 below.

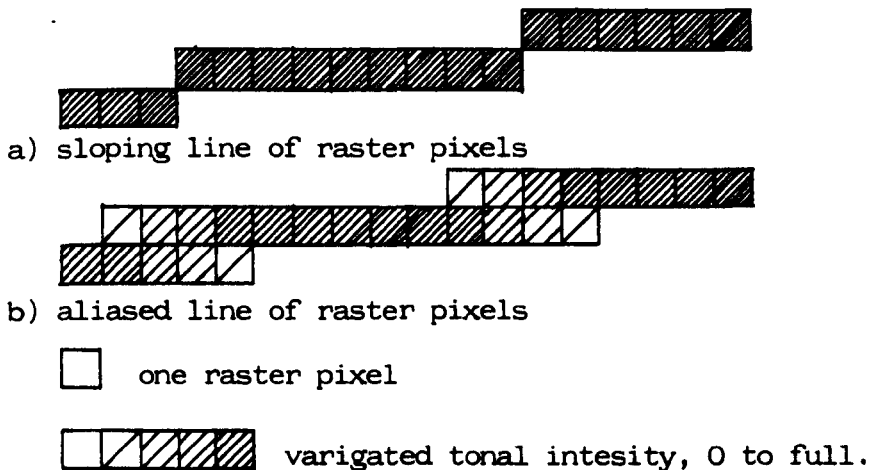


Figure 3.1 Illustration Of Aliasing Technique For Enhancing Raster Displayed Images.

In this manner, when viewed from a distance, the visual disruption is attenuated, and raster images enhanced.

## 4.2 Graphic Input Devices

### 4.2.1 Input via the QWERTY keyboard

Most if not all computer terminals are provided with a keyboard with a QWERTY character set. Besides entering text, Cartesian coordinates may be entered directly using the numeral keys. The obvious disadvantage of this method is that the user must actually compile the coordinates that he intends to input by manual methods. This means that a sketch must already exist. Not only does this entail a duplication of effort, but the process is painstakingly slow and prone to errors, both in converting the sketch to coordinates, but also in the process of typing them in. Mistakes once made are notoriously difficult to locate and correct.

The device is nevertheless cheap, and reliable.

### 4.2.2 Stepping keys

Most computer terminals are equipped with keys which control either an alpha-numeric or cross-hair cursor. Four keys control the movement of the cursor to the left, right, up and down directions. Having located the cursor in the desired position upon the screen, the pressing of a predefined key will cause the current x,y screen coordinates to be registered by the computer.

Whilst more convenient than a QWERTY keyboard, the speed of movement of the cursor can be annoyingly slow. In many of the less sophisticated terminals the cursor is incremented one pixel at a time.

To overcome this speed problem, more sophisticated terminals introduce 'fast' keys which enable the cursor position to be incremented in multiple jumps. Alternatively, the user might be limited to a grid.

These devices are accurate to one pixel. It is not all that easy

to locate the cursor at a desired point, and frequently the user has to track back and forth before positioning it correctly. The sensitivity of such devices requires a fair level of coordination between hand and eye.

These devices are usually fitted as standard to most graphics capable terminals, are cheap and robust.

#### 4.2.3 Thumbwheels

These operate in the same fashion as stepping keys. The difference being that two wheels are used to control the movement along the principal axes of the screen. These wheels are mounted at right angles, which means that effectively only one direction may be manipulated at a time. A second difference is that the thumbwheels are analogue devices whereas stepping keys are digital.

The sensitivity of thumbwheels depends upon their physical diameter, and their gearing ratio. They are perhaps more 'natural' to use than stepping keys.

Being analogue, the cursor responds more rapidly than stepping keys. The problems of over-run still apply.

Having placed the cursor, coordinate positions are again recorded by hitting a predefined hard key. These devices are also relatively cheap and robust, usually fitted, where provided, on the terminals' keyboard console.

#### 4.2.4 Joystick

The joystick is an improvement on the thumbwheel principle. This device, generally separate from the terminal, consists of a lever mounted on a universal or ball joint, and is used in much the same way as an aircraft's control column. It is a much more acceptable device than those previously mentioned, since it has the merit of being analogue, hence quick response time, and moreover, can operate along the two axes simultaneously. It also simplifies coordination between the users hand and eye.

The user still has the problem of tracking and overrun, and the devices sensitivity will depend on the friction of the device. The joystick has a finite limit to its movement due to the construction of the device, but is reasonably robust.

#### 4.2.5 Trackerball

This device consists of a ball mounted in a socket, and can be thought of as a leverless joystick. Its advantage is that there are no limits to its movement.

Although there is a more natural 'feel' to its use than the former, it can cause confusion if the user loses the reference position of the ball and the cross hair cursor goes off screen.

It is probably more sensitive than the joystick for ergonomic reasons, although subject to personal preference.

#### 4.2.6 The 'Mouse'

This is another variant of the thumbwheels principle. Here the thumbwheels are mounted on the underside of a small hand held carriage so that the movement of the mouse across a surface is picked up by the rotation of the wheels. Its accuracy is prone to errors induced by rotations of the mouse, but these errors are likely to be small. Whilst the sensitivity to movement is good, the bulk of the mouse obscures the surface on which it is being moved.

One advantage of the device is that it can be used to digitise any size of drawing, subject to the length of the flex connecting it with the terminal.

#### 4.2.7 Flat Bed Digitiser/Plotter

Some flat bed plotters may be used as digitisers by allowing the user to manually position the drawing head, and then to send coordinate data to the terminal. The positioning of the head may be achieved by thumbwheels or joystick mounted on the plotter itself.

This is a most awkward and time consuming method of digitisation, and like all devices which require the positioning of a small cross hair eyepiece over a drawing, introduces the potential problem of parallax. Plotters tend to be expensive.

#### 4.2.8 Touch Sensitive Screen

The touch sensitive screen is a laminated transparent membrane fitted in front of the display screen. When the membrane is subjected to pressure at a point, the electrical resistance between the two laminates breaks down and a current is passed which is then registered. The coordinate position of the point of contact being sent to the terminal.

Given the area of a fingertip in relation to that of the screen, the accuracy of the device would appear poor, although claims of reasonable precision have been made for it.<sup>22</sup> Also, the problem of surface obscuration applies.

Overall, it would appear to be a poor device for graphic input purposes.

#### 4.2.9 Light Pen

The light pen projects a small spot of light onto the fluorescent coating of the terminal screen, the presence of which is then registered and the coordinate position recorded. A similar scheme uses a magnetic pen which causes a small disruption in the electronic field of the screen coating.

Although the light stylus is less bulky than a fingertip, the light pen obscures the immediate area upon which the user wants to concentrate, so that accuracy is detrimentally effected. Unless the terminal screen is angled, it is both difficult and tiring to draw upon a vertical surface.

The light pen is a more natural device to use than those previously mentioned.

#### 4.2.10 Acoustic Pen

Like the light pen, the acoustic pen consists of a stylus like transponder which emits an acoustic signal. The signal is picked up by a mesh of fine wires embedded in a secondary screen in front of that of the terminal. The stylus head is generally smaller than the equivalent light pen, so obscures less of the screen surface.

#### 4.2.11 Digitising Tablet

The digitising tablet consists of a surface on which to mount drawings or plain paper, and can take drawings of up to A0 size, (1.0 x 0.75 meters approx). Below the surface is a fine mesh of wire through which is pulsed signals in both the 'x' and 'y' directions. A hand held stylus picks up these signals, and by measuring the time interval between the emission and reception of the two pulses the position of the stylus relative to the tablets' origin may be determined knowing the speed at which the signals travel. The stylus's position can be determined to an accuracy of half a mesh square which in practice is less than 0.125mm.

The tablet is an attractive input device because of its resemblance to traditional draughting methods.

**"The similarity between larger tablets and digitisers, and the conventional drawing board causes many architects and engineers to feel that they will only be able to use computers via such a device." 21**

An ink stylus is available which means that graphic data may be stored as it is created. The tablets sensitivity closely resembles that of pencil and paper techniques.

A digitising tablet may be used in two modes of operation; Point mode or Continuous mode.

##### a) Point Mode

In point mode, the tablet records the coordinate position of the single point indicated when the user presses the stylus onto the digitising surface.

In conjunction with some further control system, either hardware push buttons or software generated menus, detailed information pertaining to the point digitised can be specified. This is particularly useful for draughting systems and component based CAAD systems, as well as for digitising existing drawings.

The disadvantage of point digitisation for drawing creation is, obviously that only the node points are recorded. This means that the designer is reliant upon the image of his drawing echoed upon the terminal screen, and in this circumstance the separation of hand and eye is a disadvantage.

"The current methods of man-machine communication, (principally typed commands, screen or tablet based menus and function keys) are felt to have shortcomings. The use of dual screens often provides a partial solution." 21

Nevertheless, with the combination of point digitisation and menus, drawing editing and manipulation is possible, although it destroys the concept of 'up-to-date' hardcopy of the current design situation. Such a system is very useful in transferring existing drawings onto computer.

#### b) Continuous Mode

In continuous mode the position of the stylus is tracked all the while that it is within pick-up range of the tablet surface. Each time a pulsed signal is received by the stylus, the coordinate position is sent to the terminal, and this has the effect of sampling the coordinates over which the stylus passes. Any freehand line drawn, as represented by the sample of coordinates, can be reproduced as a series of short vectors. The sampling rate will depend upon the frequency to which the pulsed signal is set. Although rates of 200 cycles per second have been reported, this would generate too much data, and typically a rate of between 5 and 10 coordinate 'triads' produces manageable quantities of data.



The coordinate triad returned by the tablet consists of a 'header' character, followed by the 'x' and 'y' coordinates. The header character has one of four possible intentional values, these being 29,26,31, and 53. A header value of 29 indicates that the stylus has been brought into contact with the tablet surface and pressure has been applied. Thus the coordinate value following indicates that it is the first point preceeding a stream of data; it is the start point of a new line sequence.

The header value 26 refers to an intermediate coordinate point, following after a newline (29) coordinate.

A header character value of 31 indicates that at the coordinate point following, the pressure holding the stylus to the tablet surface has been released, thus indicating the end of a line coordinate data stream.

Finally, a header character of 53 indicates that although the stylus is still picking up the pulsed signals in the underlying mesh, there is no pressure on the stylus, thus no drawing is taking place.

Occasionally spurious header character values are encountered. A value of 81 appears to indicate the coordinate position at which the stylus leaves the presence of tablet surface, and no longer picks up the pulsed signals.

Other spurious triads have been encountered in practice. Due to the length of wiring beneath the tablet surface, the registration of data can be corrupted by electro-magnetic interference. Where this has happened, it would appear to be caused by the flicker of flourescent strip lights in the vicinity of the tablet.

A second source of interference can be caused by local fluctuations in the magnetic field in the close proximity of the tablet surface. To reduce this occurence, the tablet requires periodic 're-biasing' to reset the tablets magnetic field. This is done by passing a long bar magnet over the tablet surface in a prescribed manner. This is required at

roughly six monthly intervals.

Because the wires of the mesh are relatively close to the digitising surface, they are vulnerable to having their cross-section mis-shapen . This could upset the wires physical properties and thereby upset the wires attenuation of the pulsed signals. Similarly, it is possible for a wire to jump from its track and interfere with one adjacent.

In continuous digitisation mode the stylus traverses many nodes, and is therefore more liable to encounter a disruption than with point digitisation.

Disruption is seen when a corrupted data point produces 'spikes' on the drawing when the data is redrawn on the terminal screen.

Disruption due to any of the above hazards is less likely to occur than loss of data due to system buffering problems. Corruption of data is a nuisance, but not necessarily catastrophic to the data set, since some of the errors may be trapped by data validation software.

#### 4.3 PHYSICAL MODELS

In complete contrast to the data input methods outlined above, a physical model input system comprises of a set of 'building blocks', inside of which are minaturised electronic circuits which contain geometric and topological descriptions of the containing block. The designer assembles a three-dimensional model of his design, which then has a built in description of itself which can be interrogated via the computer to which it is subsequently linked. One of the components contains a mercury switch which can distinguish the 'z' dimension of the user space.

Deceptively simple, this data input method is nevertheless subject to criticisms. Despite the building blocks being un-scaled, the range of blocks necessary to facilitate the full range of a designers' potential dialogue would be very large.

In its favour, this system offers immediate visualisation,

and can be used by literally anyone, of any age. It offers a great incentive for participation of non-designers in the design process,<sup>23</sup> and is ideal for system building by the use of standard components which could be modelled to absolute scale and detail.

It is surprising that this input method has not proved more popular than it has.

#### 4.4 Summary

The methods and devices described above through which designers can describe building geometries to computers have associated levels of efficacy. All methods generate two-dimensional coordinates in space, and this is an alien way for designers to perceive drawings.

Perhaps the major determinant is the separation between hand and eye which these input devices entail. The digitising tablet, resembling in operation traditional drawing board draughting methods, provides the best opportunity for the smooth transition to computer aided design systems. Indeed, all devices where the drawing stylus, hand held, is used to draw directly mimic pencil and paper.

Devices which move a cursor involve the separation of the hand, operating the cursor moving controls, and eye, which is providing the feedback from what is actually being drawn. This is awkward, and is more convenient in terms of the machine than for the user. This does not mean to say that in course of time, cursor driven devices will not displace traditional drawing methods.

Manual digitisation requires that a drawing has already been created in order for the user to convert it to a coordinate system. This can be a tedious process, particularly where three dimensional descriptions are involved.

One factor which cannot be overlooked, is that the designer can derive a great deal of pleasure from the traditional drawing process, and possible disruption caused by the introduction of new drawing methods might well prove detrimental to the design

process as a whole.

## 5 COMPUTER SOFTWARE

A detailed discussion on the limitations inherent in the current state of computer software will be better understood in the light of an understanding of graphic data and data bases which are described in Chapter 4 following.

The following sections of this chapter discuss the more general constraints of conventional programming techniques, with particular emphasis on integrated CAAD systems. 'Dumb' draughting systems model drawings; they do not lead to an increased understanding of the design problem on the part of the designer, and consequently draughting system software will be largely ignored except to illustrate problems in common with CAAD systems.

All CAAD software is the result of a trade-off between the memory storage requirement, the amount of computation involved, the ease with which the data can be accessed from storage in the format required, and the restraints, limitations, impositions, sequences and conventions to which the user has to conform in order to use the system.

The amount of computation that a computer is required to do is largely implicit in the application for which the program is written. The amount of computation will be proportional to the amount of data which it has to process. Similarly the storage requirement will also depend on the amount of data held. However, whilst the data to be manipulated may be held in secondary storage, the program itself, and the data undergoing calculation have to be held in core memory. The larger the program, the less storage available for the data, whilst the more complex the calculation, the more data has to be reserved for data generated during the calculation process.

If data is held in secondary storage, there is a time overhead involved in its retrieval. Where much secondary storage and retrieval is involved, the time penalty might become severe.

The balancing factor to mitigate these conflicting memory storage requirements, is to impose limitations on the amount or type of data that the system will accept. The more limitations imposed, the less flexible and useful the system becomes.

Since the above factors are coded into the software systems, it is through usage that they are experienced. Obviously these programs are based upon the prediction by the programmer of what the user will want to do. Being 'post hoc',

"the programs always lag behind the aspirations of the users. A CAAD system cannot be viewed as a finished product. Building Regulations change, in-house policy changes, the needs met by the organisation change. Changes in specification occur frequently, even before the system first goes into production use." 6

A system is at best a part solution to any one designer's problems. There is always the frustration that the machine doesn't quite do what you want to do, or doesn't quite do it in the way in which you would like it done. Since the programmer's idea of 'how' operations ought to be done are those that become encoded, the user has to follow the resulting sequences precisely.

"Usually there is only one correct sequence for each function and an experienced user has to memorise many specific operations. (These systems are) built on very particular anticipations of what the user will want to do and, subsequently, prescribe to users what they may do." 5

It is not surprising therefore, that

"the user cannot normally expect to exploit the full potential of the computer, but only the potential of the programmers' knowledge of the computer. The need for the programmer as interpreter is a symptom of the sheer difficulty of getting computers to do things, of the current primitive state of software technology." 24

Where systems embody many specific operations which the user has to memorise, the operation of the system becomes a

distraction from the designers primary task; that of designing.

The means toward a solution become an obfuscation:

"That means have a tendency to become ends is a well known phenomenon in all walks of life. We come to believe that the computer has all the answers and can relieve us of effort of thinking, whereas in fact it can only answer our potentially silly questions in a way which we have taught it. The responsibility for our actions remains with us; we are, and must remain the masters." 25

As Sir Ove Arup warns us, there is a tendency among users to regard the computer as an intricate 'black-box', who's prognostications are absolute. They have to be either accepted or rejected. In view of the resources commitment that CAAD systems represent, there is an insidious pressure to accept the 'advice' which the machine offers as demonstrated fact.

For many senior architects in whose hands rests the decision whether or not to adopt CAAD systems, the resultant adverse feeling of impotence in the face of this technology is a factor often underrated by CAAD advocates.

Once a commitment has been made to adopt a CAAD system, despite frequent updates to the software, there is a limit to the amendments that the system can sustain in order to keep pace with the evolution of design practices.

"Additions become more and more difficult to integrate satisfactorily into existing code, whilst ensuring that these 'patches' do not introduce errors into the program. Dependence on the program author's 'local knowledge' of the code increases at an alarming rate.

Sooner or later some change will be required which the strained and mutilated programs simply cannot support. The remedy in theory is very simple - replace the system as one would replace any obsolete office resource. In practice, however, this implies a recognition of a limited life-span of the system such that a replacement system would have to be in development alongside the one in use. Such a

fruitful recognition of a CAAD system's lifespan is not easy in the light of the software investment that is commonly involved." 6

The rate of 'obsolescence' of software is accentuated by that of computer hardware. Accordingly there is a hesitation on the part of architects to opt for CAAD systems whilst the current state of the art in CAAD technology is so fluid.

Where modifications are successfully incorporated into CAAD systems, they generally involve increasingly complex command sequences, and hence involve the user in a time consuming familiarisation process.

Many systems attempt to mitigate the problems caused by the conflicting demands of ease of use and efficient husbandry of machine resources by limiting the user to orthogonal building geometries. Unfortunately,

"a major part of the current and foreseeable architectural workload is concerned with either the infilling of urban sites or the refurbishment and extension of existing buildings. This is in sharp contrast to the 'green-field' or 'open-desert' problems of large scale developments of the last two decades. This clearly implies that a rectilinear grid planning approach based on a system building is not sufficiently general. If CAAD systems are designed around these assumptions of regularity of form they will continue to be inapplicable to a major part of the workload of the smaller and medium sizes of office." 12

The imposition of restraints upon the designer is not simply a symptom of the machine resource conflict, but also demonstrates that many systems are machine, rather than user, centred. Systems will not become readily acceptable whilst architects have to think of design in terms of the machine perceptions rather than his own. The translation of human to machine perceptions of graphic data is achieved through the man-machine interface.

## 6 THE MAN-MACHINE INTERFACE

With respect to computer usage by architects,

"there is clear evidence that the greatest source of inhibition and frustration is right at the starting post; getting the data into the machine in the first place." 26

The computer is merely a collection of registers which, with the exception of parallel processors, can only deal with one item of data at a time, and the frustrations of communication occur as a result of the requirement to reduce the designers understanding to the simple level with which the machine can deal. It is perhaps not so much the lack of variety of input devices which the designer can use which deter, but the crudity and banality of the level at which the man-machine communication has to take place in order to describe to the machine that which is self-evident to the designer. This case has been previously argued.<sup>27</sup>

"The man-computer interface is the space of contact, the boundary layer, in which communication between human and computer takes place. This environment defines the 'form' (how) and the context (external conditions) with respect to the communication." 28

The 'how' side of the interface is dependent upon the hardware devices through which the communication is to be effected, and the command language available by which to do it. The 'external conditions' embrace not only the users understanding of the computer system, but also his conception of the data which he is trying to impart to the machine.

Newman and Sproull define the interface as comprising of four components:<sup>29</sup>

1 The User Model

This is the conceptual model formed by the user about the system: about the information he manipulates and the processes he applies to this information.



## 2 The Command Language

This is the accumulated set of command vocabulary at the users' disposal with which to communicate with the machine.

## 3 Feedback

This is the information generated by the computer as the communication takes place to provide acknowledgements, cues, advice on how to proceed, and upon the effectiveness of the users' instructions.

## 4 Information Display

This is the re-presentation of the current state of the data as a result of the users' instructions.

The user model is largely subjective, and may be modified as the user becomes more familiar with the system. The problem created here is to produce a system which is simple enough for novices to use whilst he has a crude user model concept, yet not too simplistic for users with an advanced concept.

The command language has to be consistent with the natural vocabulary that the architect generally uses. Where words are outwith the command language, they must be trapped before they cause an error, and the user given the opportunity to define the unknown word in terms of words known. This is extraordinarily difficult to achieve with conventional programming techniques, but explains why logical programming techniques hold the promise of more powerful command languages. This is discussed later in Chapter 8.

Feedback is much a case of 'negative control'<sup>30</sup>. In many implementations it is used 'interactively' to inform the user of the extent to which the data is currently short of his desired goal, and what he should do in order to rectify it. Although feedback is an essential part of a system, the problem remains that the feedback information pertains to the machines perception of the data and goals, and not necessarily that of the user. Moreover, it is still a prescribed set of rectifying actions that he is offered. Indeed,

"the word interactive is widely used within the computing fraternity, but it is used much too freely in describing many programs. Some people seem to believe that a program is interactive provided it allows the user to answer questions posed by the program whilst it is running. To be truly interactive the program must have a degree of intelligence to decide whether it can use the replies supplied by the user and, if not, to interact with the user in such a way as to find a solution to any problem that has arisen. Interaction should permit a user to present information to a program and call for operations to be performed on the information in any sequence which appears relevant to the user, within the bounds of the application encompassed within the program." 31

This is not simply a call for programs with an anticipatory element to them, but one for flexible strategies. One method for attempting to achieve this, is to use the processing power of the computer to abstract from the data possible modes of progress which the user might wish to take. This direction is very much in the fields of artificial intelligence and pattern recognition.

"In an obvious sense any device for data input involves pattern recognition of a sort, since 'reading' from a punched card for instance, involves 'recognising' the pattern of card perforations and their translation into machine language. The class of problems of interest is that which involves non-specific inputs - that is, to tolerate changes in position, size, orientation or sloppiness in the to be recognised patterns..., and, what transformations should be performed on a very large ensemble of potential input patterns to reduce them to a specifiable set of categories." 32

Indeed, it is uncertainty which characterises the early stages of the design process, when the architect is working toward a defineable geometric description.

"Whilst these characteristics are the anathema of algorithms, they are the essence of design." 33

The fundamental problem underlying computer pattern recognition is that conventionally,

"computer systems...use a laborious approach. In general they have worked on a step by step, trial and error comparative basis which sequentially rejects or confirms each of a series of variables set up for examination." 34

From the foregoing, it would appear that conventional computing techniques offer little hope of significant developments in computer pattern recognition of architects drawings.

The user model can be enhanced by more easily assimilable systems, perhaps by the automatic generation of higher level command languages.

There is a strong argument for architects to adjust their working methods in the light of the emergence of the new technologies. Conversely there is a good case for resisting the implementation of processes whose results have not yet been seen to produce better architecture as a result of their implementation. It should perhaps be reiterated that,

"quite apart from the unfamiliarity that most designers feel when confronted with an electronic drawing device, what must not be overlooked is that most architects actually enjoy drawing and are understandably reluctant to see this particularly self-satisfying part of the process being taken away from them." 26

## 7 SUMMARY

Some of the factors acting as disincentives to architects from adopting computer aided design systems are the problems of finance, disruption to traditional working methods, specialist staffing, the imposition of objectivity onto a process traditionally regarded as being essentially subjective, and the difficulty of communication of drawings between man and machine.

Despite technological innovations currently being developed,

the present state of the CAAD art is limited by the physical restrictions imposed by computer hardware, software and the interface through which users communicate with the machine.

## 8 CONCLUSION

The problem to be overcome to make CAAD more acceptable to the small private architectural practice may be stated thus:

- 1 A suitable system must allow the architect to pursue his design activities in the sequence he wishes to adopt.
- 2 The user interface must be sympathetic to the vocabulary normally used by architects, based on the medium of drawings.
- 3 The system should not intrude upon the design process.
- 4 The system should emulate existing design techniques.
- 5 A suitable system must be directed towards practical architectural problems.
- 6 Such a system should not be based upon a priori assumptions about what is being designed, but should proceed from what is being designed as represented by the architects drawings.

The following chapter examines four existing suites of computer software, written around differing applications, in an attempt to determine how they have resolved the above problem.

References

- 1     Mystery Dissolved  
      Jim Daley  
      Building Design                     20th Nov 1981   pp 32-36
  
- 2     Low Cost Computing: Finance  
      Nicholas Coutts  
      Architects Journal                 2nd Mar 1973    p 76
  
- 3     Systems: Difficult To Choose, Easy To Use  
      Architects Journal                 17th Dec 1980   p 1211
  
- 4     Towards An End User View Of Design Systems  
      R.J.Walters  
      PARC83 Proceedings  
      On-Line Conferences                 Oct 1983   pp 21-27
  
- 5     Dumb Drawing Systems And Knowledge Engineering  
      Aart Bijl  
      EdCAAD, University Of Edinburgh    Jul 1981
  
- 6     Prescriptive To Descriptive Programming  
      Peter Swinson  
      Paper for the Logic Programming Workshop,  
      John Von Neumann Society for Computer Sciences  
      EdCAAD, University Of Edinburgh    Jul 1980
  
- 7     Graphic Interaction With Database Systems  
      J.H.Nash  
      CAD82 Proceedings  
      Butterworths                        Mar 1982   pp107-118
  
- 8     Electronic Computers Made Simple  
      Henry Jacobowitz  
      W.H.Allen                            1967
  
- 9     The Man In The Belly Of The Beast  
      (Interview with Professor J.Weizenbaum, MIT)  
      John Davy  
      Observer                            15th Aug 1982
  
- 10    Zen And The Art Of Motorcycle Maintenance  
      Robert Pirsig  
      Bodley Head                         1974
  
- 11    A Computer Assisted Draughting System For Use In The Jobbing-  
      Engineering Industry  
      R.K.Hilton  
      CAD82 Proceedings  
      Butterworths                        Mar 1982   pp 425-436

- 12 ISAAC: A Technique For The Automatic Interpretation Of Spaces  
From Drawn Building Floor Plans  
Brian Lawson  
CAD82 Proceedings  
Butterworths Mar 1982 pp 663-666
- 13 Building Description Techniques To Support Computer Aided Design  
And Production  
Aart Bijl  
Section 25 of SERC Grant Application  
EdCAAD, University Of Edinburgh Dec 1981
- 14 Approaches To Computer Aided Sketch Design  
Dave Willey  
Computer Aided Design Vol.8 No.3 Jul 1976 pp 181-186
- 15 Design: A Case Study  
R.T.Krauss, J.R.Meyer  
Massachusetts Institute Of Technology 1968
- 16 The Architecture Machine  
Nicholas Negroponte  
Computer Aided Design Vol.7 No.3 Jul 1975 pp 190-195
- 17 SeeLog - A Prolog Graphics Interface  
Fernando Pereira  
EdCAAD, University Of Edinburgh Report 83/06 Aug 1982
- 18 C-Prolog User's Manual, Version 1.3  
Ed Fernando Pereira  
EdCAAD, University Of Edinburgh Report 82/11 Aug 1983
- 19 Supercomputers  
Ronald Levine  
Scientific American Jan 1982 pp 112-125
- 20 Raster Display Of A Rotating Object Using Parallel Processing  
T.Strothotte, B.Funt  
Computer Graphics Forum Vol.2 No.4 Nov 1983 pp 209-217
- 21 Computer Draughting Systems In Construction - The Buyers' Problem  
Ian Hamilton (CICA) and J.R.F.Burdett (BRE)  
CAD82 Proceedings  
Butterworths Mar 1982 pp 338-345
- 22 The Data Station: An Integrated Graphic Information Facility  
Patrick Purcell  
PARC83 Proceedings  
On-Line Conferences Oct 1983 pp 311-319

- 23 The Use Of Simplified Three-Dimensional Input Devices To Encourage  
Public Participation In Design  
John H.Frazer, J.M.Frazer and P.A.Frazer  
CAD82 Proceedings  
Butterworths Mar 1982 pp 143-151
- 24 Can Computers Understand Designers  
Aart Bijl  
PARC83 Proceedings  
On-Line Conferences Oct 1983 pp 169-181
- 25 Expert Systems: Their Impact On The Construction Industry  
foreword by Sir Ove Arup  
John Lansdown  
Report to the RIBA Conference Fund Sep 1982
- 26 ENIGMA: An Enhanced Interactive Graphics Module For Architects  
Lamond Laing, Dave Leifer, Brian Hammond  
Occasional Paper No.10  
Scott Sutherland School Of Architecture Jul 1983
- 27 Man Before The Machine: An Issue In CAAD  
Dave Leifer  
BoCAAD, University Of Strathclyde, Vol.43-6 Feb 1983 pp 107-112
- 28 Introduction To Man-Computer Communication  
J van den Bos  
Computer Graphics Forum Vol.1 No.7 Dec 1982 p 194
- 29 Principles Of Interactive Graphics  
William Newman, Robert Sproull  
McGraw Hill 1973 p 445
- 30 Computer Simulation Of Continuous Systems  
R.J.Ord-Smith, J.Stephenson  
Cambridge Computer Science Texts 3  
Cambridge University Press 1975 p 251
- 31 Graphic Interaction With Database Systems  
J.H.Nash  
CAD82 Proceedings  
Butterworths Mar 1982 pp 107-115
- 32 Visual Pattern Recognition  
P.C.Dodwell  
Holt Reinhart and Winston 1970 p 93
- 33 On Being Creative With Computer Aided Design  
Nicholas Negroponte  
Information Processing 77, IFIP Conference  
North Holland 1977 pp 695-704

34 The Patterns Of Power  
Anthony Tucker  
The Guardian

Thur 24th Sep 1981 p 11



## Chapter 4

### Graphic Data And Data Structures

1	Graphic Data	4.1	65
2	Pictorial Data	4.1	65
	2.1 Characteristics	4.1	65
	1 Physical	4.1	65
	2 Noetic	4.2	66
	3 Imputed	4.3	67
	4 Representational	4.4	68
	2.2 Human Perception	4.4	68
3	Geometric Description Methods	4.7	71
	3.1 Point Set Method	4.7	71
	3.2 Boundary Description Method	4.10	74
	3.3 Boolean Description Method	4.12	76
4	Data Structures	4.13	77
	4.1 Point Set Method	4.15	79
	4.2 Boundary Description Method	4.19	83
	4.3 Boolean Description Method	4.26	90
	4.4 Conclusions To Data Structures	4.26	90
5	Two And Three Dimension	4.27	91
	References	4.29	93

## CHAPTER 4

1 Graphic Data

The previous chapter concluded by describing the architects' role as that of conceiving of, and communicating the description of a hypothetical building. A design morphology was described, whose central feature is the derivation of a database by testing and expanding upon a building form arising from the architects' imagination. The medium of communication pertaining to these ideas of form, is almost universally that of drawings, hereafter referred to as 'graphics'.

This chapter describes pictorial graphics, and then goes on to examine the various methods for describing the geometries of real objects, as are depicted by graphical representations.

Following this, the subject of data structures with which to store such geometric descriptions, is reviewed.

Finally, observations concerning two and three-dimensional objects are made.

2 Pictorial Data

## 2.1 CHARACTERISTICS

Boreham and Edmonds<sup>1</sup> distinguish four categories of pictorial characteristics of images of real objects. These are:

## 1 Physical

"These are limited to the unambiguous features which result directly from an analysis of the physical properties of the image. They consist of tonal value and location of an atomic element in two-dimensional space, together with the features found by comparison of atomic elements." 1

In terms of the computer, the presence or absence of a mark on a piece of paper or a device is an 'atomic element' of information. Since there a very large number of combinations of atomic elements possible even in the simplest of drawings, the

"features found by comparison of atomic elements" appears rather doubtful. Unless the system knows what it is looking for in any particular picture, all possible combinations of atomic elements have equal potential significance. No relationship between any combination of marks on a sheet of paper can be deduced without some precognitive expectation, or rules, which would necessarily have had to be defined, or indeed implicit, in the computer.

For the ensuing discussion, 'atomic elements' will be taken to mean the presence of an 'event'<sup>2</sup> against a null background being described to the machine.

## 2 Noetic

"These are structural inter-relationships apprehended by the user, based on the internal organisation of the image. 'Internal' relationships are those which can be ascertained from the physical characteristics without reference to the properties or appearance of real three-dimensional objects. They are similar to mathematical relationships, and include some of the 'principles of organisation' such as proximity and similarity." 1

These 'noetic' characteristics are explained with reference to Figure 4.1 below.

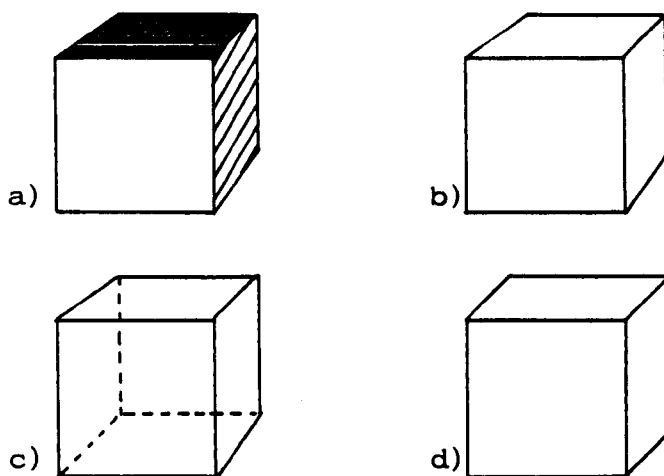


Figure 4.1 Illustration Of Noetic Characteristics

Figure 4.1 a) shows a real three-dimensional object depicted by line drawings b), c) and d). The physical characteristics are those tonal distinctions which define the presence or absence of lines. The representation in b) actually comprises of three two-dimensional quadrhedra, but they are imputed by the human observer to represent a geometry shown in c). Moreover, b) and d) illustrate two different representational techniques, perspective and orthographic respectively.

Thus Figure 4.1b) is the purely two-dimensional analysis of the picture presented by the view of the three-dimensional object in a).

### 3 Imputed

"These are derived from the attributes of physical three-dimensional objects, whilst not actually representing particular features of unique objects, They include the conceptions of overlay and transparency which are attributes of physical planes in three-dimensional space, but which are not necessarily perceived as representing the appearance of real objects." 1

As was instanced in Figure 4.1c), our experience of real objects leads us to conclude that the object represented possesses three hidden faces in addition to the three shapes existent in b). The progression from a purely geometric analysis of a representation of an object to an interpretation of it as a set of intersecting planes, is not always possible, as Figure 4.2 below illustrates.

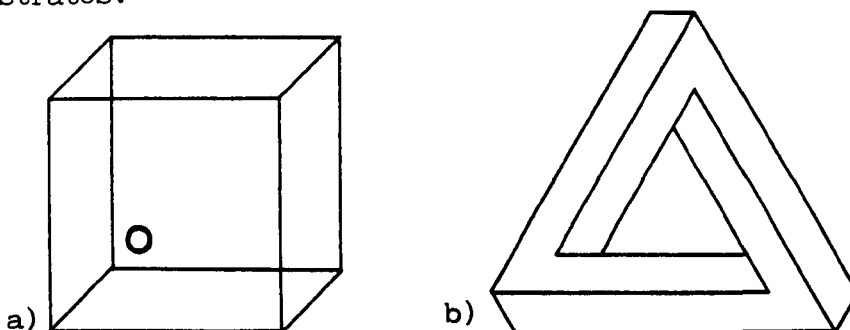


Figure 4.2 Ambiguous Imputed Characteristics

Figure 4.2a) shows the Necker Cube. The ambiguity demonstrated is the difficulty of determining whether the circle is on the fore or rearward surface of the cube. Similarly, Figure 4.2b) shows a figure which, whilst noetically consistent, would be impossible to realise as a solid three-dimensional object.

#### 4 Representational

"These depict features of real three-dimensional objects, usually by a two-dimensional projection of their apparent qualities when viewed monocularly. Thus they either refer to a conceptual definition of an object feature, an object, or a class of objects, or employ features of monocularly viewed objects as graphic conventions." 1

One example already cited, is that between Figures 4.1b) and 4.1d). In the case of the former, the cube is drawn in single point perspective, whilst the latter is in parallel projection.

An example of a 'graphic convention' may be given by the example where a non-existent three-dimensional object, such as a mathematical surface may be depicted.

These characteristics of pictorial data are not as discrete as the above paragraphs would suggest. They are of course related through mutual reinforcement, nor can one think of representational characteristics existent without the noetic, for example.

The theoretical analysis of drawn information might well be useful for discussion in terms of 'machine perception', but become less relevant when considered from the point of human perception.

## 2.2 HUMAN PERCEPTION

By some unknown process the image of a drawing falling onto the retina of the human eye is converted into a perception. The physiological processes involved up to the point where the image is transmitted to the brain by electrical impulses, and has been well documented.<sup>3</sup> However, we are at a loss to explain how this

train of pulses are recognised.

"The most fundamental problem concerns the human tendency to perceive images as 'wholes' rather than as a sum of the parts.

There is a clear distinction between (the) two stages (of) registration of the data, and the interpretation of the data. Where only one interpretation is possible, we may regard the data upon which it is based as having unambiguous properties, but wherever more than one interpretation is possible, the data can be said to have ambiguous properties. In the latter case, the percept will result from one interpretation being selected from the possibilities." 3

That is, that visual perception cannot be explained by the simple 'one-to-one' mapping of the environment onto the receptor of the perceiving organism. It is suggested that a 'perceptual hierarchy' must be involved wherein,

"the input of the receptor...is more and more 'de-particularised', stripped of irrelevencies during its ascent (from stimulus to percept). The output hierarchy concretises, the input hierarchy abstracts. The former operates by means of triggering devices, the latter by means of filtering or scanning devices. Triggers release complex outputs by means of a simple coded signal. Scanners work the opposite way: they convert complex inputs into a simple coded signal." 4

"In order to identify the input, the brain must activate some sort of memory trace. Some very complex scanning process must be involved which first identifies characteristic simpler features in the complex whole (visual holons like loops, triangles etc.); then abstracts the relations between the features; and then the relations between the relations." 5

In a computer system, the base-holons (or atomic elements) of information are easily defined, whereas the 'abstracting' hierarchy is far more tenuous. Whilst we cannot define an open

ended system which could cater for all eventualities, by proscribing various options we might at least achieve a system which is manageable if closed.

An example of this would be the process of recognising enclosed polygonal shapes from a picture composed of lines. Arcs would have to be proscribed, and the rule that all shapes have to be enclosed spaces enforced. However, this will be described more fully in due course.

Figure 4.3 below illustrates a schematic representation of a perception hierarchy.

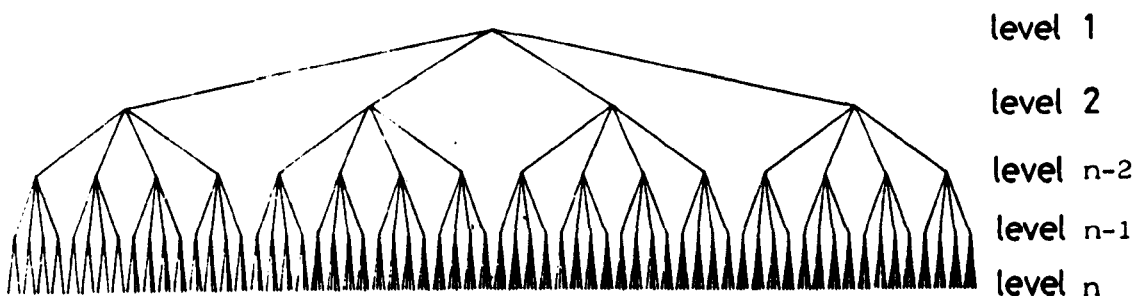


Figure 4.3 A Schematic Representation Of A Perception Hierarchy

At the lowest level  $n$  are the physical characteristics, (marks on the paper). These are associated into discernable lines which are associated into shapes, (the noetic characteristics), and so on up the hierarchy, culminating in the precept.

From the top down, the node at level 1 represents the percept of the drawing. This is achieved through the indefinite number of intermediate levels, involving the synthesis of the four pictorial characteristics described above.

The illustration would seem to suggest that each percept may be definitively mapped, but this is not so. Only by rigorous and comprehensive definition of the trigger and filtering mechanisms can the crudest of machine pattern recognition systems be evolved.

This thesis deals with the abstraction of some noetic characteristics of pictures from the physical; that is, those mathematical relationships and principles of organisation deductible from the drawing. However, chapter 8 describes some potential processes by which a limited range of 'fuzzy' principles of organisation, eg. generic similarity, may be ascertained.

Before proceeding to the algorithms and processes to achieve this low level interpretation of line drawings by computer, it is necessary to examine the different ways in which one can describe solid geometries, (ergo line drawings), to a computer, and the effects these descriptions have on methods of computer data storage.

### 3 Geometric Description Methods

Ultimately the purpose of a geometric description of a real object is to record the presence or absence of materials at locations in space.

There are three methods for describing geometries. These are;

- 1 The Point Set Method,
- 2 The Boundary Description Method, and
- 3 The Boolean Method.<sup>6</sup>

#### 3.1 THE POINT SET METHOD

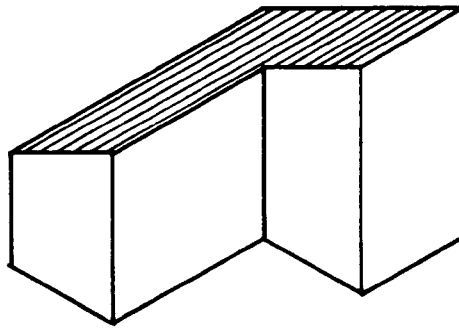
With the point set method, a space is described by means of a three-dimensional matrix of contiguous orthogonal cells, each of which may or may not contain material, (see Figure 4.4 overpage).

##### Resolution

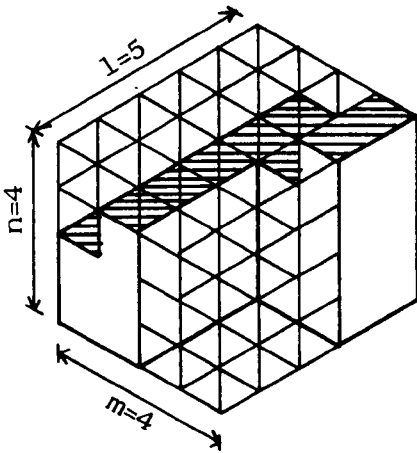
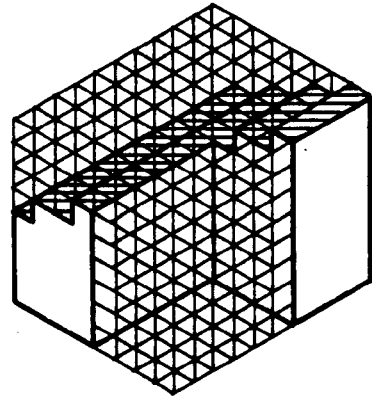
The finest resolution which may be achieved by this method of description is one cell unit. A doubling of the resolution requires an eightfold increase in the number of cells involved.

One method of reducing this exponential increase in





a) Object to be described

b) Object resolved in a  
5 x 4 x 4 cell universec) Object resolved in an  
10 x 8 x 8 cell universe

n=1

1	1	1	1	1
1	1	1	1	1
0	0	0	1	1
0	0	0	1	1

n=2

1	1	1	1	1
1	1	1	1	1
0	0	0	1	1
0	0	0	1	1

n=3

0	0	0	0	0
1	1	1	1	1
0	0	0	1	1
0	0	0	1	1

n=4

0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	1	1

d) Storage array (1 x m x n), containing coded description  
of object resolved in b)Figure 4.4 The Point Set Method Of Encoding Shape  
Description.

storage requirement is the utilisation of a variegated cell structure as shown in Figure 4.5 below.

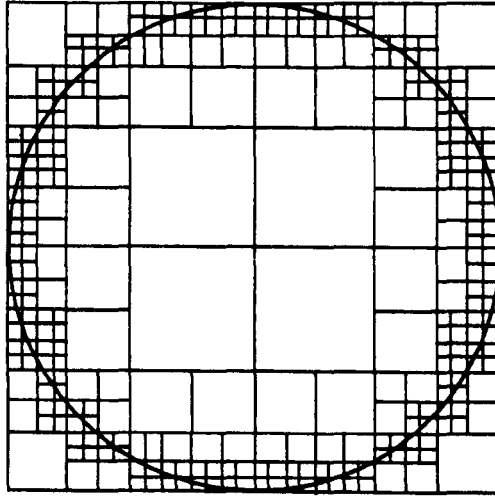


Figure 4.5 A Variegated Cell Structure

Economy of storage is achieved at the expense of more complex mapping functions. In the above example, each axis would have to be located by a five-bit coordinate description in order to keep track of the cell divisions.

#### Computation

With this method of geometry description it is very easy to check whether a point in object space lies within a described object by interrogation of the cell within which the point lies, hence clash checking is easily accomplished. However, the description is very basic, and it is not so straightforward to generate shapes from their algebraic definition, and in most cases requires converting to other description systems.

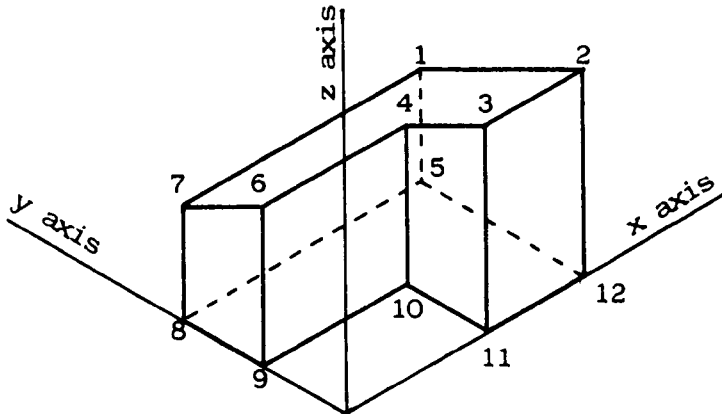
#### Application

Whilst this method of modelling is of great value in

applications where a solid is to be reduced from a homogeneous material, eg. the automatic control of machine tools, the typical CAAD object is not only of far greater physical size than an engineering component, but is also composed of a variety of different materials. In addition, the CAAD user is equally concerned with the spaces within the object, as with the fabric of the object itself. Consequently this method is not often used for CAAD applications.

### 3.2 THE BOUNDARY DESCRIPTION METHOD

With this method of geometry description, the lines bounding the edges of the surfaces of an object are described exactly by their mathematical algebraic definitions. As illustrated in Figure 4.6 below, the object is construed as filling the object



a) Object to be described in Cartesian Space

Surface i.d	General Equation	Values
1	$l_1x + m_1y + n_1z = c_1$	$l_1, m_1, n_1, c_1$
2	$l_2x + m_2y + n_2z = c_2$	$l_2, m_2, n_2, c_2$
3	$l_3x + m_3y + n_3z = c_3$	$l_3, m_3, n_3, c_3$
..	..	..
8	$l_8x + m_8y + n_8z = c_8$	$l_8, m_8, n_8, c_8$

b) Data storage scheme for planar boundary description.

Figure 4.6 The Boundary Description Method Of Encoding Shape Geometries

space described by the 'wire-frame' construction obtained by instantiating values into the equations describing the boundaries.

#### Resolution

This description method is accurate to the limit of the size of number capable of being held in the computer, since equations can be reduced to infinitesimal increments. All surfaces can be reduced to a set of small surface elements which may be accurately defined as a 'B-spline' curve in three dimensional space.

#### Storage

The storage requirement is much less than for the equivalent point set, although it requires a more complex data structure to do so. Only the coordinate positions of the nodal points are stored, along with records of which nodes connect which lines, and which lines comprise which surfaces. Fairly sophisticated database management systems are needed to keep track of these related sets of data.

#### Computation

Most operations performed using this description method entail large amounts of computation of coordinate geometry. As mentioned above, since alterations to the description can lead to catastrophic inconsistencies corrupting the database, a fair amount of computation has also to be carried out to validate the data.

#### Application

Nearly all existing CAAD systems utilise this form of geometry description method. Topological relationships can be derived from the description, and consistency checking can be accomplished by application of Euler's formulation,

$$V + S - ( E + 2 ) = 0$$

where for a polygon;

V is the number of vertices,

S is the number of surfaces, and

E is the number of Edges.

This method also accords well with the data format required to drive drawing devices.

### 3.3 THE BOOLEAN DESCRIPTION METHOD

"The Boolean method... is in essence similar to the point set method except it is the directed boundaries between point sets that are stored; shapes are then defined by performing the logical operations of union, intersection and difference on the point sets." 6

One implementation of Boolean shape description has been evolved by Giraud<sup>7</sup>.

"Today's 3D wireframe systems...exhibit some serious deficiencies: they are not valid because they tolerate 'nonsense' objects with missing edges; they are ambiguous in that different objects may have the same representation." 7

As an alternative, two Boolean based methods have been tried;

"CSG (constructive solid geometry) and the boundary representation. In CSG 3D objects are considered as the union, intersection and/or difference of elementary volumes. In the boundary representation 3D objects are represented by their faces.

What is needed is a representation scheme which is valid and unambiguous like CSG, which has uniqueness like the boundary representation and from which one can conceive interfaces to satisfy all designers' needs, in time and effort which are not prohibitive." 8

For a full description of the "Presque Half Space" Boolean geometry description method, the reader is referred to references number 7 and 8.

### Resolution

As with the boundary description method, the description is accurate to the limit of the number capable of being held in the host computer.

### Storage

Similar in content and context to the boundary description method.

### Computation

It is only relatively recently with the advent of SeeLog, a PROLOG language based system, that a general purpose logic based Boolean graphics interface has been written.

The computation is carried out by interrogating the database to see which facts fit the rules and conditions set by the problem. A great deal of data searching takes place as opposed to direct calculation, and consequently the processing time increases with the amount of data stored in an exponential progression. Currently the response time of Boolean systems leaves much to be desired, but it is likely that the application of these systems on parallel processors might overcome this time overhead.

### Application

As yet general purpose Boolean systems are still under investigation and development. As mentioned above, SeeLog<sup>9</sup> is in its infancy, but its potential in conjunction with 'knowledge engineering' systems holds great promise.

## 4 Data Structures

The terms 'database' and 'data structure' tend to be used synonymously in CAAD literature. For the purposes of this thesis these two terms will be defined as follows:

**Data Structure** - The organisational arrangement for storing

data in the computer memories for subsequent use by application programs.

Database - The total aggregation of explicit knowledge resident in a data structure.\*

Whilst the data structure employed will depend upon the application for which the program is written, the geometric description method employed will also have a very direct bearing. Other relevant factors include the amount of core storage available in the host computer, and the amount and speed of access available as secondary storage.

Most CAAD application programs facilitate some degree of geometric modelling. This model is described through an interactive graphics input medium in the general case. Williams identifies four categories of problem pertaining to graphic data structures:

"First there is a representation problem. Graphical data, diagrams etc. are two-dimensional but computer storage is one dimensional. Consequently there is a mismatch and the two-dimensional data must be mapped into a one-dimensional store. Furthermore there are many implicit relationships in a two-dimensional plane that must be preserved in a computer representation.

Secondly there is a computational problem. Computation proceeds serially in one dimension and so there is a problem of converting essentially two-dimensional operations into sequences of processing steps.

Thirdly, problem solving and computer aided design with graphics is an environment where highly complex interactions take place. This is a problem because it is very difficult to manage dynamic data structures.

Finally there are data presentation problems. In most applications users want

\* Where in the following text quotations occur, the terms 'database' and 'data structure' will be replaced as necessary, to conform to the definitions so made.

to work at different levels of detail and they also want to vary the scope of the data presentation at each level. Therefore it is important to be able to define data subsets and selectively alter the detail associated with such a subset. Conversely, it must be possible to combine or aggregate data from different sources into a displayable entity. In both cases an ability to explore and peruse the data is desirable." 10

The four problems identified are thus,

- 1 mapping the data into storage,
- 2 extracting the data in the form wanted by the user,
- 3 associated problems effecting the ease of editing and manipulating the data structure, and
- 4 ease of access to the database.

These four facets of data structuring are causally related, and are manifested in the implementation of the methods of geometry descriptions.

#### 4.1 POINT SET METHOD

As seen from the preceding description in section 3.1, an object described as a point set is reduced to a matrix of cells either containing material or null.

##### Mapping

Mapping is fairly easily achieved, particularly with the use of high level languages which have macros for dealing with subscripted multi-dimensional arrays. The users perception of the data structure exactly matches that of the object, although not identical to the storage sequence in the machine. In lieu of array handling macros, mapping functions may be implemented. A typical mapping function for converting three dimensional arrays into a list is shown in Figure 4.7 overpage.



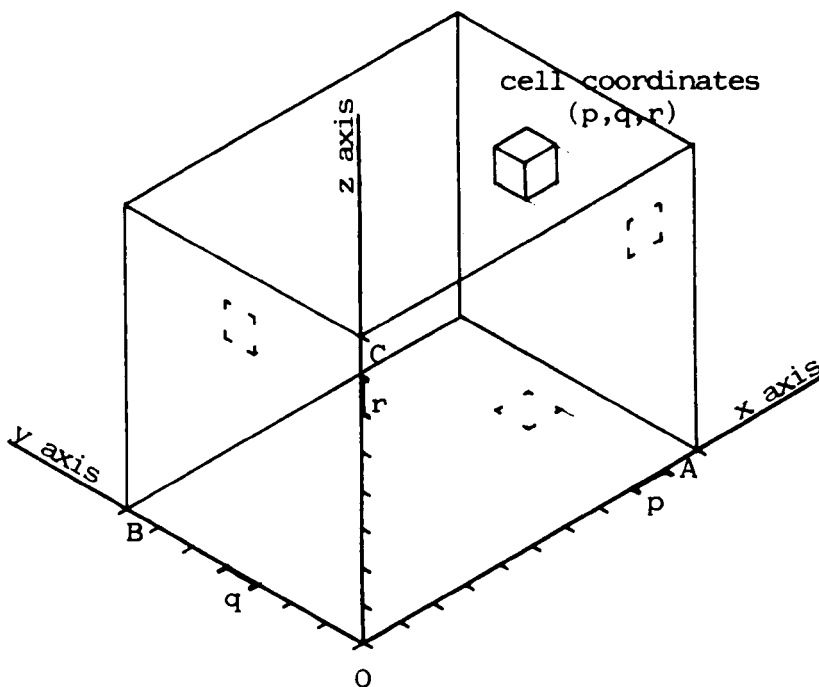


Figure 4.7 Parameters Of Three-Dimensional Description For Mapping To One Dimensional Data List.

The cell at coordinate position  $(p,q,r)$  in a volume of  $A \times B \times C$  cells, is given as a position in a list by the function

$$f = (A \times B) \times (r - 1) + A \times (q - 1) + p$$

#### Extraction

The actual data bit held in storage can hold information both of the material specification of the cells content, and pointers to adjacent cell locations, depending upon the user programs application. Data is extracted by calling for information from the list by use of the mapping function.

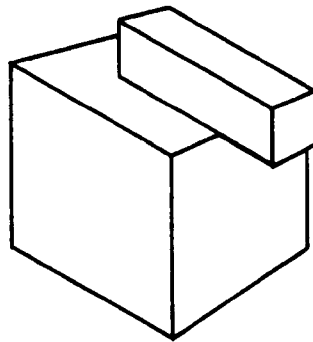
Derivation of topological relationships could require the comparison of each cell to every other, although surface and edge conditions may be found directly as explained in due course.

#### Manipulation

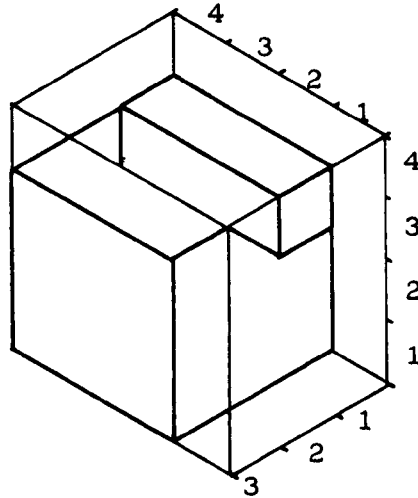
Changing or amending the geometric description

involves producing algorithms whose variables are transformed via the mapping functions, which inevitably involve a large amount of calculation, although not a significant time overhead. The situation becomes more complex when non-orthogonal planes are involved, as these manipulations require the stepping effects to be resolved.

Cells constituting corners, edges and surfaces can be identified quite easily. Figure 4.8 below shows a simple solid described in a 4x3x4 object space.



a) Object to be described



b) Object in 4x3x4 object space.

Figure 4.8 Solid Object To Be Coded By Point Set Method To Demonstrate Identification Of Surface, Edge And Corner Cells

If for each cell we record the presence or absence of material in adjacent cells in contact with each of its six faces, the total number of surfaces in contact indicates surface, edge and corner cells. The data set for Figure 4.8 is shown in Table 4.1 below.

Mapping function	Coordinates			Value	There is an adjacent cell to the:						Total
	p	q	r		left	right	above	below	fore	behind	
1	1	1	1	0							
2	2	1	1	1	0	1	1	0	0	1	3 corner
3	3	1	1	1	1	1	1	0	0	1	4 edge
4	4	1	1	1	1	0	1	0	0	1	3 corner
5	1	2	1	0							
6	2	2	1	1	0	1	1	0	1	1	4 edge
7	3	2	1	1	1	1	1	0	1	1	5 surface
8	4	2	1	1	1	0	1	0	1	1	4 edge
9	1	3	1	0							
10	2	3	1	1	0	1	1	0	1	0	3 corner
11	3	3	1	1	1	1	1	0	1	0	4 edge
12	4	3	1	1	1	0	1	0	1	0	3 corner
13	1	1	2	0							
14	2	1	2	1	0	1	1	1	0	1	4 edge
15	3	1	2	1	1	1	1	1	0	1	5 surface
16	4	1	2	1	1	0	1	1	0	1	4 edge
17	1	2	2	0							
18	2	2	2	1	0	1	1	1	1	1	5 surface
19	3	2	2	1	1	1	1	1	1	1	6 internal
20	4	2	2	1	1	0	1	1	1	1	5 surface
21	1	3	2	0							
22	2	3	2	1	0	1	1	1	1	0	4 edge
23	3	3	2	1	1	1	1	1	1	0	5 surface
24	4	3	2	1	1	0	1	1	1	0	4 edge
25	1	1	3	0							
26	2	1	3	1	0	1	0	1	0	1	3 corner
27	3	1	3	1	1	1	0	1	0	1	4 edge
28	4	1	3	1	1	0	0	1	0	1	3 corner
29	1	2	3	0							
30	2	2	3	1	0	1	1	1	1	1	5 surface
31	3	2	3	1	1	1	1	1	1	1	6 internal
32	4	2	3	1	1	0	0	1	1	1	4 edge
33	1	3	3	0							
34	2	3	3	1	0	1	0	1	1	0	3 corner
35	3	3	3	1	1	1	0	1	1	0	4 edge
36	4	3	3	1	1	0	0	1	1	0	3 corner
37	1	1	4	0							
38	2	1	4	0							
39	3	1	4	0							
40	4	1	4	0							
41	1	2	4	1	0	1	0	0	0	0	1
42	2	2	4	1	1	1	0	1	0	0	3 membrane
43	3	2	4	1	1	0	0	1	0	0	2
44	4	2	4	0							
45	1	3	4	0							
46	2	3	4	0							
47	3	3	4	0							
48	4	3	4	0							

Table 4.1 Point Set For Object In Fig.4.8, Including Registration Of Adjacent Positive Cells

From consideration of the total number of surfaces of any one cell in contact with other filled cells (physical characteristics<sup>11</sup>), the following noetic characteristics may be defined:

No. of Surfaces in contact	Noetic Characteristic
6	The cell under consideration is internal, and will only be seen in a section through the solid.
5	The cell lies on a surface.
4	If the two free faces of the cell are opposite, the cell is part of a membrane: Otherwise it lies on an edge.
3	The cell is either the edge of a membrane or a corner.
2	If the two connected faces are opposite, the cell is part of a 'rod': Otherwise it is at the junction of such a rod.
1	The cell is the termination of a rod.
0	The cell is independent.

Particular surfaces and edges may be determined by examining which of the three coordinate values each edge or surface cell hold in common.

Distinguishing surfaces facing a particular orientation may be effected by examining which of the faces of any particular surface cell is null, ie. left, right etc. However, the situation becomes exceedingly complex when analysis of pitched surfaces is required.

#### 4.2 BOUNDARY DESCRIPTION METHOD

"Clearly, the most direct method of describing a polyhedral form to a computer is by listing the three-dimensional coordinates of each point and then specifying which points are connected by edges." 12

Systems based on the entities of faces, edges and vertices are known as 'graph based' systems.

"These and the links between them, (points, faces and edges), may be included into the the data structure directly. This allows the user to proceed in a unified manner down beyond the part as a primitive unit to examine its constituent geometry and topology." 13

This is consistently so because;

"any shape may be considered a polyhedron. The faces of a polyhedron may be planar or curved. The complete description of any polygon consists of a topology of the adjacency relations between its constituent parts and a geometry specifying the dimensional aspects of the shape. Without both aspects, a shape description is incomplete and some operations on it are not possible. In most data structures, these two aspects are not distinguished." 14

Graph based systems assume that boundaries are described in their algebraic form, with the physical coordinates of the vertices acting as delimiters or alternatively calculated from the interaction of the algebraic descriptions dictated by the stored topology.

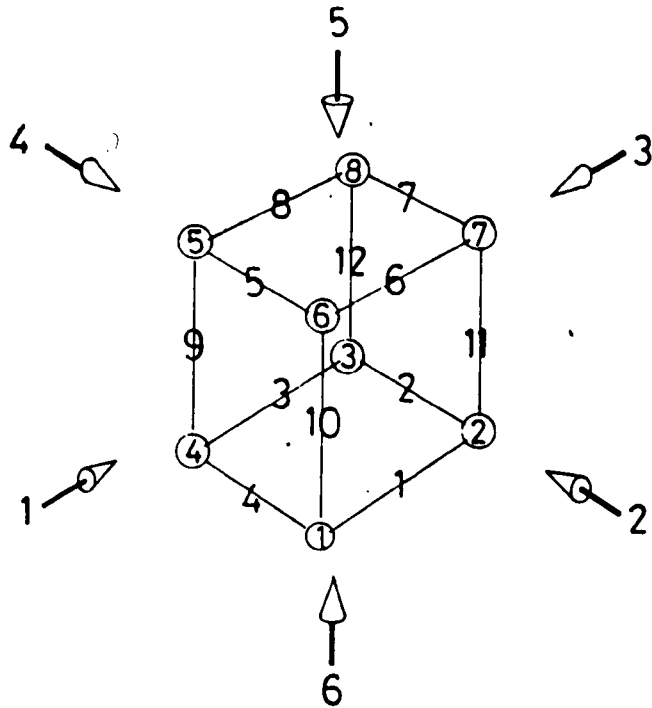
The general equation for a surface enjoys three degrees of freedom. Constrained in one degree it describes an edge, and in two, a point.

#### Mapping

Unlike the point set, mapping data into the computer cannot take place on a one-to-one basis, and indeed the advantage of the boundary description is that it condenses the amount of data to be stored, virtually irrespective of resolution. However, this economy is achieved at the loss of simplicity, since a hierarchical structure is needed.

The complete mapping of a polygonal solid requires the inclusion of the coordinate positions of the vertices, a specification of which vertex pairs describe which boundary line, and which boundary lines contain which surfaces. At the next higher level might be a description of which surfaces form which solid.

Such a data set is shown in Figure 4.9 below.



Vertex coordinates	Edges vertex nos.	Faces edge nos.
1 $x_1, y_1, z_1$	1 1 2	1 4, 10, 5, 9
2 $x_2, y_2, z_2$	2 3 3	2 1, 11, 6, 10
3 $x_3, y_3, z_3$	3 3 4	3 2, 12, 7, 11
4 $x_4, y_4, z_4$	4 4 1	4 3, 9, 8, 12
5 $x_5, y_5, z_5$	5 5 6	5 5, 6, 7, 8
6 $x_6, y_6, z_6$	6 6 7	6 1, 2, 3, 4
7 $x_7, y_7, z_7$	7 7 8	
8 $x_8, y_8, z_8$	8 8 5	
	9 4 5	
	10 1 6	
	11 2 7	
	12 3 8	

Figure 4.9 Data Set Describing Topological Relationships Of The Boundary Description Of A Polygonal Solid, With Reference To Coordinate Values For Instantiation Into Algebraic Equations

### Extraction

This is achieved by a process passing through the levels of the data hierarchy. Top down, vertices are extracted from the surfaces or superior levels, whilst bottom up, surfaces are referenced from the vertices. These related data lists effectively cross reference storage addresses of the vertex data items, and is indeed known as a 'relational data structure'.

### Manipulation

As has been seen, instantiation of coordinate values into the general algebraic equations gives precise data for application calculations.

There is an underlying assumption, unless all solids are described in their triangulated form, that all vertices on a surface lie in the same plane. Consequently problems arise when the user moves a vertex during editing or manipulation. Either the whole plane is moved, or it is tilted. In both cases other vertices related to the transient point have also to be moved. The algorithms necessary to monitor these changes can become very complex, particularly where a plane so moved is common to more than one data set. Similarly additions and deletions effect the relational data structure. Unless solids interact at their vertices, calculation intensive routines (clash checking) have to be invoked to ensure data consistency.

### Access

Access can be effected from any level of the data hierarchy.

The data structure presented in Figure 4.9 , comprising of related lists is perhaps the simplest implementation. An alternative approach is the tree structure, also heirarchical in effect, but where each level of data forms a subset of that at the superior level. Such a heirarchical tree structure is illustrated in Figure 4.10 overpage.

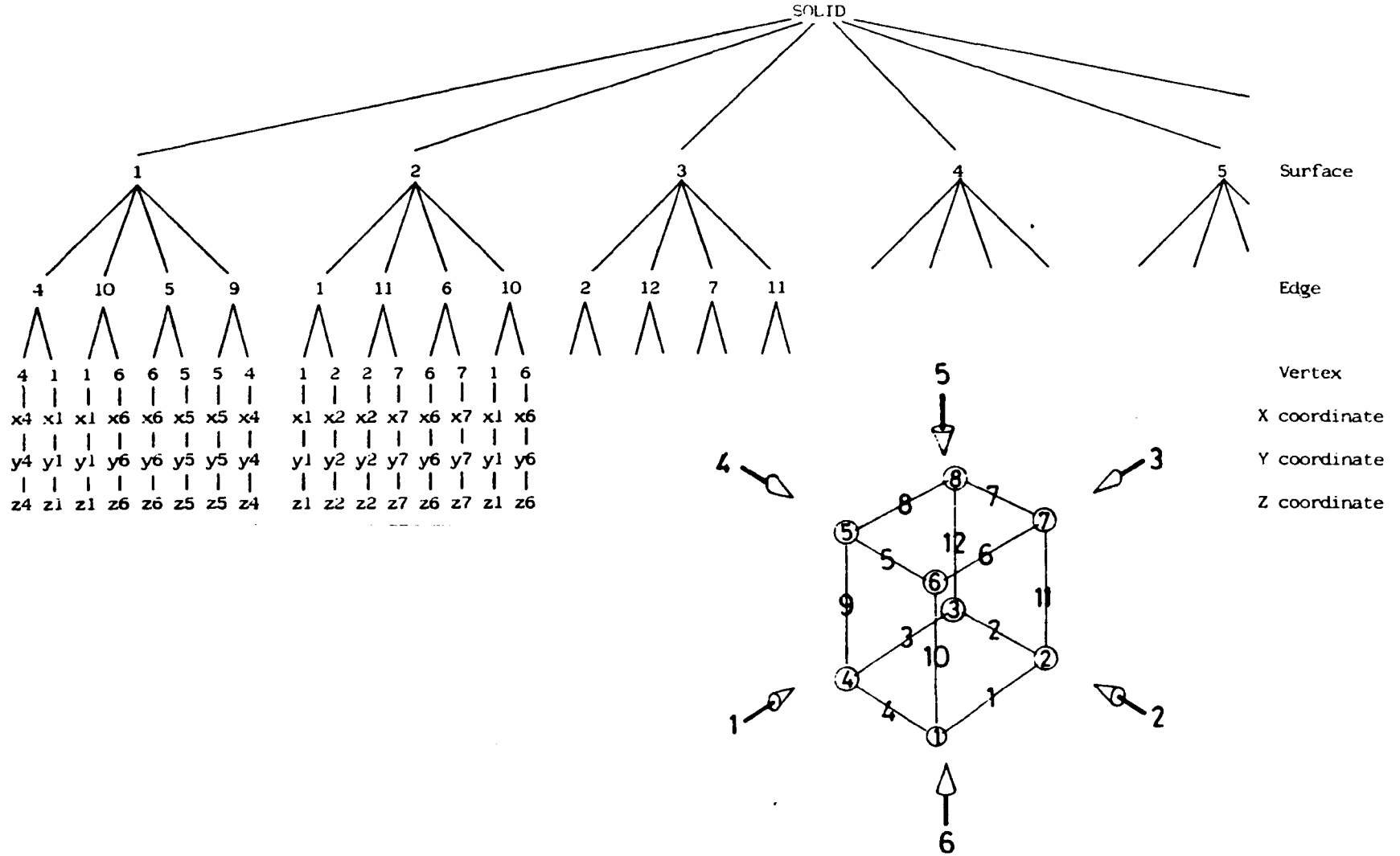


Figure 4.10 Tree Structure For Encoding Boundary Description Of Object (inset) From Figure 4.9



"Hierarchical data structures may be represented by directed graph structures. Blocks of data (records) are represented by directed arcs between the nodes. A list then is a chain of nodes and arcs. A ring is a closed chain of nodes and arcs called a circuit. A tree structure corresponds to a directed graph which has no circuits and which can, therefore, be redrawn to exhibit the levels in the tree. A hierarchical organisation corresponds to a general directed graph which does not contain circuits. As a result each structure may be classified as a sub-class or special case of the next more general structure." 15

In Figure 4.10, for example, each node of the tree has links to both superior and inferior nodes. The data recorded can have pointers to both 'parent' and 'daughter' nodes, thereby allowing changes to one node to be followed down through the hierarchy into the bottom level of the tree. However, the effects a change to a node has on the higher levels of the hierarchy might invalidate definitions of the superior nodes themselves, eg. the alteration in position of a vertex might move it out of the plane of the surface under which it is 'filed' in the tree structure.

At the price of greater complexity ring structures can be incorporated into daughter subsets of the tree hierarchy at the same level. Moreover, these can be extended to include pointers both forward and backward. This is illustrated in Figure 4.11 overpage, where pointers have been added to companion files to the data set accompanying Figure 4.9 on page 4.21.

However,

"a database on a given subject is a collection of data on that subject that obeys three criteria; completeness, non-redundancy, and appropriate structure." 16

By these criteria, consistency is achieved at the cost of non-redundancy. Even so, there is no guarantee that consistency at one level of the hierarchy will ensure consistency at superior levels.

## Connections

lateral		forward		back	forward	back	lateral								
2	4	6	1	1	4	10	10	5	9	4	1	4	5	2	6
1	3	7	2	1	2	11	11	6	10	1	2	1	5	3	6
2	4	8	3	2	3	12	12	7	11	2	3	2	5	4	6
1	3	5	4	3	4	9	9	8	12	3	4	3	5	1	6
4	6	8	5	5	8	9	5	6	7	8	5	2	3	4	1
1	5	7	6	5	6	10	1	2	3	4	6	1	4	3	2
2	6	8	7	6	7	11									
3	5	7	8	7	8	12									

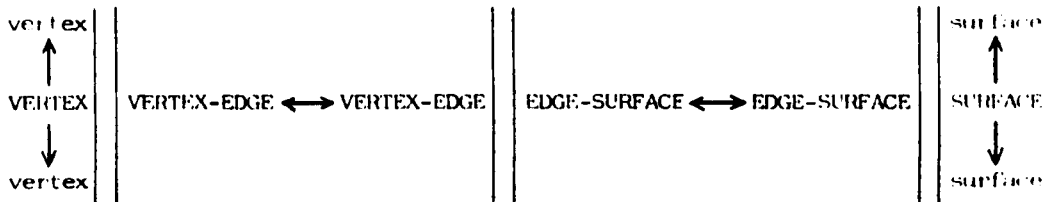


Figure 4.11 Further Data Structure Incorporating Lateral, Forward And Backward Pointers.

(Refer to Figure 4.9)

Once a ring structure has been incorporated into a data structure, a structured data acquisition system is imposed, either upon the user directly, or by routines to ensure that data is passed into the data structure in the correct sequence. Where for example, a shape is to be drawn into the data structure, most CAAD software requires that the user trace out each shape ending the input sequence at the point where he started. However, once shapes start aggregating, shapes which may be relevant to the picture might be formed by the composition of the other declared shapes. Tracing over these shapes now involves the repetition of line data, which may cause failure in subsequent computation.

"One may conclude that, although simple data organizations can be designed quite easily on a basis of searching and updating times, complex data structures cannot. This is because flexible, versatile data structures are tediously intricate and may grow unpredictably in any direction.

Data structures are dependent upon the machines characteristics and, more

importantly, on the applications for which they are to be used." 15

#### 4.3 BOOLEAN DESCRIPTION METHOD

Unlike either of the previous two methods, data is not mapped as such into the computer storage, other than as lists. Each list is treated as a subset of variables with related attributes as defined by an embracing label. A data type, given as a general clause defines the 'arity' of the data type that follows; eg. in the PROLOG language, the statement

point(X,Y,Z)

defines a variable type, 'point', which has associated with it three values. These values may themselves be labels referring to further data types. Consequently, the label

line(Point,Point)

when invoked will cause the system to begin a process of instantiation to find all instances of 'point' which satisfies the conditions set by the statement in which 'line' appears.

The data itself has no meaning other than that its label declares. The machine itself maintains track of the various data sets.

#### 4.4 CONCLUSIONS TO DATA STRUCTURES

There are three methods of geometry description, each having a significant effect on the data structure that may be used.

The point set method involves a one-to-one mapping of the occupancy of cells into a storage list. The larger the object space and the greater the resolution, the greater the storage requirement. Data storage by this method has great simplicity since data is held at only one level. However, its usefulness is limited, and it is rarely used for CAAD applications.

There are a range of data structures which can be used with the boundary description method. The more robust the data

structure, the increasingly complex the database management system in order to maintain data consistency. The data is held in a hierarchy of at least three levels, unless user geometries are restricted to predefined forms. Data storage tends to be more efficient than with the point set, and application programs of great sophistication can be supported. Boundary descriptions are almost universally used for CAAD applications, hence the predominance of relational hierarchical data structures. Such a data structure also accords well with the structure required to drive vector graphic display units and incremental plotters.

The data structures appropriate to Boolean descriptions are still the subject of investigation. CAAD applications in which such descriptions are used, tend currently not to involve mathematical calculation.

A data structure may be described as,

"a collection of interrelated data stored together with controlled redundancy to serve one or more applications in an optimal fashion; the data are stored independent of programmes which use the data; a common controlled approach is used in adding new data and modifying and retrieving existing data within the data base." 17

## 5 Two And Three Dimension

Real objects exist in three dimensions; drawings are two-dimensional representations of them.

The discussion of geometry description methods and the resulting data structures have dealt with three dimensional models. However, with one notable exception (see Chapter 3, section 4.3), all graphics input devices are two-dimensional. Therefore, input data has to be translated into three dimensional form if solids are to be modelled.

Programs which require a three dimensional description at input, eg. GRAMP (see Chapter 5, section 5), either restrict the forms which the user may describe to orthogonal and prismatic forms, thereby treating the solid as an extrusion of its plan form,

or demands painstaking attention by the user to specify the height coordinate of each vertex in turn, eg. RODIN (see Chapter 5, section 4).

Fortunately for most CAAD applications, buildings tend to be extrusions of their plan forms between a floor and ceiling level.

An alternative approach to translating from two to three dimensional input data consists of the simultaneous interpretation of three orthogonal projections of a solid. This approach is known as  $2\frac{1}{2}D$ . Although such systems are known to exist, there is doubt about their ability to handle complex polyhedra, and it is likely that these systems only handle solids on the basis that they have vertical sides.

Almost all draughting systems use two dimensional descriptions. Consequently they do not model solid bodies, only their two-dimensional aspects.

In summary, the vast majority of CAAD systems use the boundary description method for encoding solid geometry, treating objects as extrusions of the plan forms. Apart from the advantages outlined above, the restriction decreases the amount of calculation that would otherwise be required to process irregular polyhedra.

### References

- 1    Extracting Shapes From Grey-Scale Images  
Dominic Boreham, and Ernest Edmonds  
International Journal Of Man-Machine Studies  
  No.16            1982 p.136
- 2    Introduction To The Graphic Kernel System - GKS  
F.R.A.Hopgood, D.A.Duce, J.R.Gallop, and D.C.Sutcliffe  
A.P.I.C Studies in Data Processing, No.19  
Academic Press   1983
- 3    The Working Brain  
A.R.Luria  
Penguin   1973 pp.229-244
- 4    Janus: A Summing Up  
Arthur Koestler  
Picador   1978 pp.23-56
- 5    The Ghost In The Machine  
Arthur Koestler  
Hutchinson    1967
- 6    Integrated CAAD Systems  
Aart Bijl, David Stone, and David Rosenthal  
Final Report to DoE Funded Research Project DGR 470/12  
EdCAAD, University of Edinburgh                 Mar 1979 p.2-9
- 7    The PHP Representation  
Christian Giraud  
EdCAAD, University of Edinburgh                 Feb 1984
- 8    Presque Half-Planes: Towards A General Representation Scheme  
Christian Giraud  
Computer Aided Design                             Vol.16 No.1 Jan 1984
- 9    SeeLog - A PROLOG Graphics Interface  
Fernando Pereira  
Report No.83/06  
EdCAAD, University of Edinburgh                 Aug 1982
- 10   On The Application Of Relational Data Structures In Computer Graphics  
Robin Williams  
Data Structures, Computer Graphics And Pattern Recognition  
Eds A.Klinger, K.S.Fu, and T.L.Kunii  
Academic Press   1977 pp.154-155
- 11   How To Handle An Image  
Steve Scrivener  
Design Studies   Vol.4 No.1 Jan 1983 pp.35-41

- 12 RODIN - A System For Modelling Three Dimensional Roof Forms  
J.P.Riley and Brian R.Lawson  
CAD82 Proceedings  
Butterworths  
Apl 1982 pp.667-670
- 13 A Spatially Segmented Solids Database - Justification And Design  
K.M.Quinlan and J.R.Woodwark  
CAD82 Proceedings  
Butterworths  
Apl 1982 pp.126-132
- 14 The Concise Structuring Of Geometric Data For Computer Aided Design  
Charles M.Eastman  
Data Structures, Computer Graphics And Pattern Recognition  
Eds A.Klinger, K.S.Fu, and T.L.Kunii  
Academic Press  
1977 pp.33-34
- 15 A Survey Of Data Structures For Computer Graphics Systems  
Robin Williams  
Data Structures, Computer Graphics And Pattern Recognition  
Eds A.Klinger, K.S.Fu, and T.S.Kunii  
Academic Press  
1977 p.118
- 16 Database Design And Implementation  
Daniel Martin  
Van Nostrand Reinhold  
1980
- 17 An Introduction To Database Systems  
C.J.Date  
Addison Wesley  
1976

## Chapter 5

### Existing Graphics Programs

1	Introduction	5.1	96
2	SKETCHPAD	5.2	97
	2.1 Operation	5.3	98
	2.2 Criticism	5.5	100
3	OXSYS/BDS	5.6	101
	3.1 Data Structure	5.8	103
	3.2 User Restrictions And Further Features	5.9	104
	3.3 Criticism	5.10	105
4	GABLE	5.10	105
	4.1 Hardware Configuration	5.11	106
	4.2 Operation	5.11	106
	4.3 Data Structure - Floors	5.11	106
	4.4 Data Structure - Roofs	5.13	108
	4.5 Criticism	5.14	109
5	GRAMP	5.15	110
	5.1 Limitations	5.15	110
	5.2 Data Structure	5.16	111
	5.3 Criticism	5.18	113
6	Summary	5.18	113
	References	5.19	114



## CHAPTER 5

1 Introduction

Chapters 2 and 3 discussed the problems and concepts pertaining to the practice of architecture and the process of building design. Chapter 4 introduced the problems of describing building geometries to computers, and the various consequent approaches to geometric data storage.

In this chapter, four existing suites of computer programs dealing with graphic data will be examined, to see how they have handled the problems previously described.

The four programs examined are:

- 1 SKETCHPAD,
- 2 OXSYS/BDS,
- 3 GABLE, and
- 4 GRAMP.

The first of these, SKETCHPAD, is probably the oldest interactive computer graphics program developed as a Ph.D thesis in 1963.

"Many of the basic ideas that the computer graphics industry now relies upon were in (SKETCHPAD), Ivan Sutherland's original thesis. What has been developed since is the capability to deal with complexity, and making images that are believable to the eye.

The kind of data structure he used for representing the object was very different from what had been used before. It was a structure based on the topology of the subject, describing the relationships between the parts of the object." 1

SKETCHPAD is the seminal work on computer draughting, yet with a data structure in advance of many used by CAD systems today.

The second suite of computer programs involves the suite GDS.

"During the sixties Oxford Regional Health Authority commenced the development of a 'building system', utilising a limited

number of components based on a steel structural system. The objectives, common to other systems of the era, were the use of prefabrication to increase the efficiency of site operations." 2

Having achieved a standardised component based system, the creators built an integrated CAAD system around the Building Description System (BDS), pioneered by ARC Ltd, Cambridge, and has been in use since 1974.

The third, GABLE is a CAAD design and appraisal package developed for implementation on a small desk-top computer. It is an interesting suite of programmes since the problems inherent in CAAD systems are exacerbated by the 'shortage' of machine resources.

Finally, GRAMP is the geometric data handling core used in the SPACES and BIBLE programs for building evaluation and visualisation developed by the ABACUS unit at Strathclyde University.

## 2 SKETCHPAD

"The primary idea...was that computers were simulators. Such simulators can be used by people who are doing design to replace real objects on occasions when a simulation can be built more cheaply than the physical model. You can make changes more easily, and the advantages of a simulator are very strong. We did graphics first because we thought they were an essential link between the human user and the simulation." 1

SKETCHPAD was written in 1963 as a Ph.D thesis by an electrical engineer. At the time, computer processors were still under development, and SKETCHPAD was run on the then experimental TX-2 mainframe computer via a time sharing graphics terminal.

SKETCHPAD is a two-dimensional graphics drawing and manipulation package, although later versions were used as the nucleus of pattern recognition software such as HUNCH (see chapter 6), by the Architecture Machine Group at the Massachusetts Institute of

Technology. The basic format of the data structure used in SKETCHPAD is still in use in some draughting systems eg. SORCE\*.

Although not a 'dumb drawing system' by Bijl's definition<sup>3</sup> it is not far removed from one. Whilst storing data from which to reproduce drawings, it also keeps track of the topological relationships of the lines constituting shapes, which can be modified without disrupting the database.

It is machine centred insomuch that the user,

'will (have to) issue specific commands with a set of push buttons, turn functions on and off with switches, indicate position information and point to existing parts of the drawing with the light pen, rotate and magnify picture parts by turning knobs, and observe the drawing on the display system.' 4

## 2.1 Operation

Drawings are entered as a set of geometric entities such as lines, arcs or predefined shapes. Each new entry has to be preceded by specifying to the system which entity is about to be input by means of a hard key. Once specified, the coordinate data anticipated by the system has then to be entered in the correct sequence. Figure 5.1 overpage shows part of the SKETCHPAD data structure.

The geometric data is stored in a series of lists. At the apex of the hierarchical structure is a list of records holding a shape type descriptor, and a storage address of the start of a list of data pertaining to that shape. In Figure 5.1 the address A refers to a block holding further addresses of the storage locations of the terminal coordinates of the line A. In addition, this block has storage address locations of the line data for the lines adjoining at both ends, ie. The line list holds firstly the address of the lines start point, secondly its stop point, thirdly the storage address of the line list pertaining to

\* SORCE, written by Norie Hill Ltd, London, for implementation on Hewlett Packard 9600 series desk-top computers.

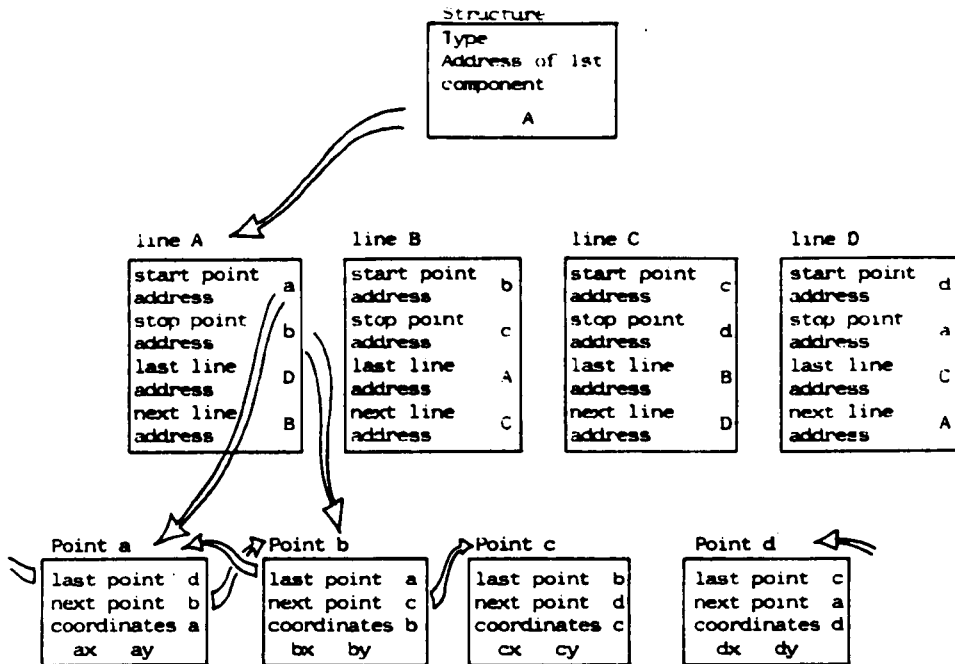


Figure 5.1 The SKETCHPAD Data Structure

the line which meets line A at the start coordinate, and finally the storage address of the line list pertaining to that line which meets line A at the stop coordinate.

The interesting feature of this storage regime, is that the line lists are cross referenced to each other when they form a closed shape. Such a contained relational structure is defined as a 'ring'. These rings were designed to facilitate the ease of data editing, since lines can be inserted or deleted by simply removing a link in the ring, and then connecting together the two loose ends.

Similarly, the lists of the end coordinates of the lines are also connected by a ring data structure; each point also recording the storage reference addresses of both the pro and pre-ceeding points connected by the two lines intersecting at that point.

The lists have slightly differing capacities in order to accomodate the differing line types, ie. lines or arcs. Enclosed shapes, once defined in its own ring structure can be manipulated

as one entity. Merging nor splitting of shapes so defined can be carried out with this system.

## 2.2 Criticisms

Understandably, in view of its age, SKETCHPAD has attracted a fair amount of criticism.

"Sutherland, in his thesis on computer graphics introduced a method for dealing with the variable shape of physical (two-dimensional) elements at the level of their automatic generation. His thesis relied on closed curves to depict the boundaries of physical elements. Each point on the perimeter of an element was considered a variable to be manipulated. Sutherland defined the desired shape of an element as algebraic equations. A feasible solution is any set of perimeter points that satisfy these equations." 5

Thus the data structure, whilst effective for recording and manipulating drawings, requires more complex algorithms and data structures in order to process the data; the system is thereby a typical dumb draughting system, good at recording line drawings, permitting drawing amendments, but awkward for feeding in to calculation procedures.

"The seeming limitations of this approach are that many elements require an undefined number of points to describe their periphery. Variable shape generation should be able to determine the number of points and add or subtract them as needed. No effective procedure for doing this has been proposed in Sutherland's scheme. Spatial overlaps, connectivity between elements of irregular shape and other important criteria seem computationally expensive by this approach. Whilst the shortcomings may be resolvable, a further reason for not following this approach is that the point vector has not been a common representation for space planning." 5

On this point, even Sutherland's coworkers are forced to concur:

"To think in terms of vectors is an archaic

way of thinking about graphics. We only think about lines because we have hands, and with pencil and hand movements we draw lines.

Seeing surface intersections as lines is the wrong approach and is counter productive...A lot of people are interested in geometrical modelling in terms of surfaces and solids." 6

The desire to write-off vector graphics seem premature, since vectors are most compatible with the majority of graphic output devices. Pending the development of logical language graphic systems, no serious alternative has yet been demonstrated. Yet understandably the prevalence of raster graphic display systems would appear to be a great stimulus for change.

Sutherland himself states in his thesis that,

"every system which is devised for programming on computers has little problem areas which give humans more trouble than other parts; it took months before all the nuances of these problems were learned." 4

Perhaps the most salutary criticism comes from Sutherland as he concludes,

"It is only worthwhile to make drawings on the computer if you get something more out of the drawing than just a drawing." 4

### 3 OXSYS/BDS

Hospital design entails a number of accentuated problems atypical of most other architectural design projects. The sheer size of such problems led to the development of a design system by the Oxford Regional Health Authority, which was subsequently transferred in part to computer some ten years after the systems introduction. At the core of the computer design system is the Building Description System (BDS) developed by Applied Research of Cambridge, which is currently commercially available. The current system runs on a minicomputer time sharing between a number of

dedicated graphics workstations. The system was first implemented in 1974.

"The Oxford Method of Building has been developed over some twelve years and is intended to cover a range of health programme buildings. The method employs a structural steel frame, a well defined set of components, a modular planning grid and a high degree of modular coordination." 7

"The fundamental framework of both the building and CAD systems is the overall geometry which acts as both a coordinating discipline and a referencing system.

The computer system consists of essentially three main elements - a Codex which is a file of all the parts of the building and engineering systems and their properties, a building file which is a description of the building being modelled, and an extensive range of application programmes which, through commands, act upon the building description without the need for further extensive input." 8

The facilities provided by the system include programs for the description and analysis of:

- a Building Organisation - input grids, bay dimensions, levels, vertical sections, zone perimeters.
- b Codex - create/edit a codex, create/edit components and component data, create/edit component properties.
- c Site Description - input site data, locate/rotate a building, create/edit a site obstruction, draw site.
- d Room Data And Schedules - create/edit zone and zone data, output zone or bay parts.
- e Component Selection and Location - general purpose component selection, location and deletion; selection, location and deletion for internal partitions, internal doors, steelwork, precast floor slabs, roof-lights and cladding.
- f Structural Steel Design - design a beam, load analysis, update structure.
- g Interactive Production Drawing - Tektronix and plotter

output, production drawing format and output.

- h Evaluation - element cost, sunlight and heat loss.
- j Services - low temperature hot water design.

In addition to the component building system and the CAAD system, a third part of the OXSYS design method is a project management programme not yet on the BDS computer system.

### 3.1 Data Structure

The BDS data structure as it appears to be implemented in OXSYS, is in the form of a four level index to a list of components as schematically depicted in Figure 5.2 below.

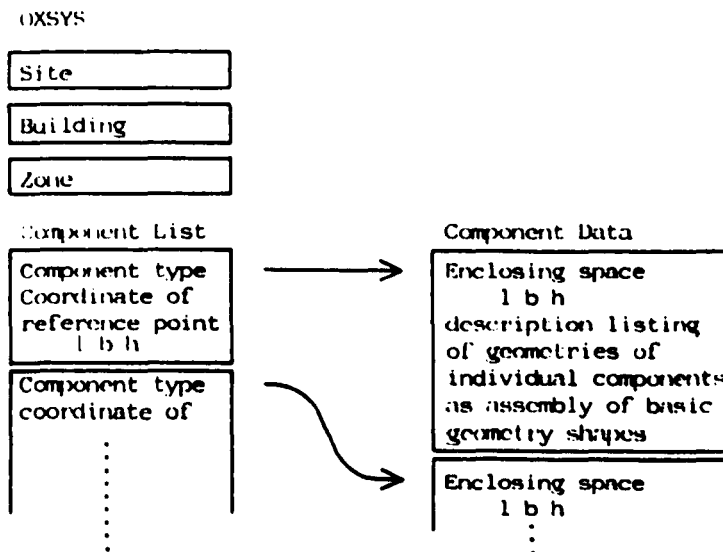


Figure 5.2 Schematic Representation Of OXSYS Data Structure

At the top of the hierarchy is a three dimensional description of the site upon which are buildings. These buildings are located relative to the origin of the site. The building is an assembly of zones, and the zones are defined as areas containing the aggregation of components therein.

Components are referred to by identifying type labels, each component being unique but capable of being duplicated. These components reside in an enclosing box, so that in the first instance



zones are modelled as a collection of cuboids. This is in order to facilitate economy of calculation when the system undertakes clash checking. Unless the containing cuboids intersect, there is no reason to proceed and check for interaction between the potentially more complex items contained within them. The component identifying labels also refer to associated attribute files which contain physical data about each item which is required when the evaluation programs are run; eg. information such as weight and thermal properties.

The component list can be thought of as a series of overlays, each one of which records the positions of each instance of an item. These overlays can be organised to suite any user required indexing format, including the CI/Sfb system.

Zones share a common environment, thus the heating loads are calculated on the enclosing surfaces of the zone. This may run counter to zones described in terms of usage.

### 3.2 User Restrictions And Further Features

The use of this simple data structure is possible because of the limitations accepted by the system users. The most onerous of these is the restriction of building geometries to a planning grid and the imposition of rectangular geometries.

Only certain classes of zone may overlap. This might prove awkward in practice, but seems trivial as a limitation.

Implicit in the data structure is the principle of containment. Sub-zones must always be contained within their superior zone, which presumably means that amendments to the database must be monitored at two levels simultaneously.

Functional relationships between members of zone classes and between zone and zone components are again implicit in the zone hierarchy.

Some components may have an 'adjacency code' which will limit the types of other component with which it may form a junction, thus limiting the designers choice of combinations of component.

### 3.3 Criticism

Although a number of the limitations outlined above are trivial, in practice an overall assessment reveals that whilst the OXSYS system solves some of the problems of hospital design for ORHA, component based systems are not relevant to the requirements of most architectural practices. How much is attributable to the paucity of component based systems, and how much to the computer system remains speculative.

The practical disincentives militating against the wider adoption of the OXSYS system may be summarised as follows:

- "a The OXSYS rationale (resource distribution and shared image) does not take adequate account of the limitations of traditional techniques and of design team attitudes. (Implementation) remains an objective rather than an accomplishment.
- b OXSYS/BDS is restrictive in the system concepts, implementation tools and user interface design employed, but the system is powerful and shows improvement upon traditional techniques.
- c Choice of implementation tools is crucial.
- d System support for the content (as opposed to the fabric) is better due to the lack of relational complexity.
- e The system facilities provided for high level input and data consistency are unusual and much needed, but current facilities are inadequate in practice.
- f Detailed design systems associated with the fabric are generally not robust enough to withstand building system development."

## 4 GABLE

GABLE is the acronym for a General Aid to Building Layout and Evaluation, and is an integrated CAAD system developed at the University of Sheffield and commercially available since 1981. The most significant feature of the system is that it is designed to run on an 'inexpensive' desk-top computer.

The GABLE software consists of nine separate application

programme modules interacting with a common database. Each application programme is held on a separately loadable magnetic tape.

Ignoring the draughting program module, the GABLE software is in three basic sections; data input, data interpretation and appraisal, and data output.

#### 4.1 Hardware Configuration

GABLE is designed to be implemented on the Tektronix 4050 series graphics terminals; ie. the 4051, 4052 or 4054. The Tektronix 4907 file manager system and disk drive unit is also needed. Data input is effected, in the Tektronix 4054 option by the built in thumb wheels, or joystick in the case of the others. Hardcopy graphic output is created via a Tektronix 4663 A2 sized flatbed plotter.

The total purchase cost of the system described above is less than forty thousand pounds at 1983 prices, which includes the GABLE software.

#### 4.2 Operation

The system is structured to permit the user to design at the computer terminal. The building geometry is entered in two parts; roofs are entered as a set of pitched surfaces, whilst the remainder of the geometry is treated as an extrusion of the floor plans. The requisite tape is loaded into the computer, which enables access to the relevant geometry creation program. Separate tapes exist for the description of external walls, internal partitions, doors and windows. Doors and Windows are specified with reference to a moderate components library.

#### 4.3 Data Structure - Floors

Although precise details of the data structures used by commercially available software is unavailable, from the operation of the system one can surmise the approximate features, and these are illustrated schematically in Figure 5.3 overpage.

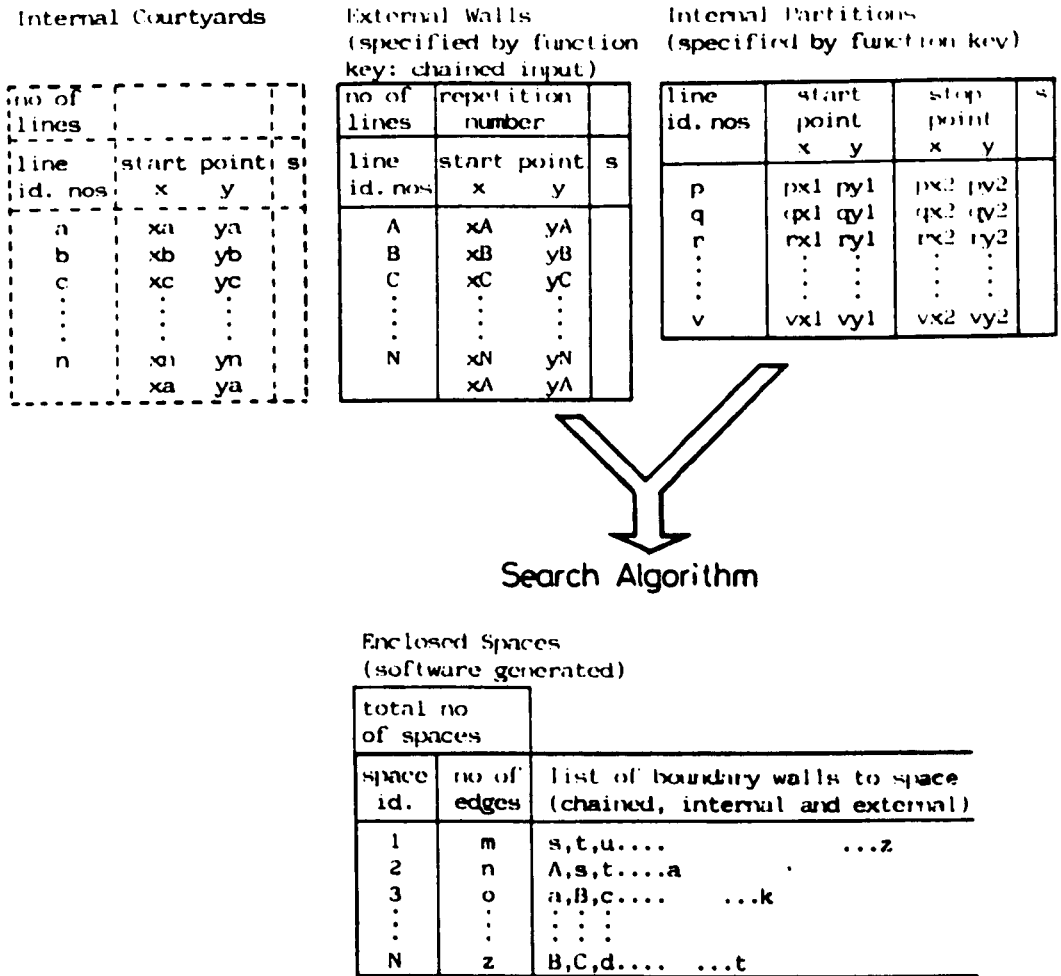


Figure 5.3 Schematic Representation Of GABLE Data Structure For Recording Walls

The schematic data structure depicted above shows two basic lists containing locational data for the end coordinates of lines depicting internal and external walls, each wall having a type code which defines its thickness and materials of construction with associated physical properties by reference to an attribute file.

In actual operation, hard function keys are used to specify the type of component that the user is about to input. In the case of external walls, the system will chain the input lines by default until such time as the user indicates that the chain is complete by entering a point instead of a length of wall. Although

it is possible to break away from this convention, interpretation of floor plans cannot be carried out until the external envelope is completed.

The software has the facility for deducing the presence of courtyards, and sets up the necessary data structures to hold this additional external wall information.

Internal partitions are held in a structure which holds the coordinates of the line vector by which the partition is represented on the screen, and also its associated specification identifier label. A limitation is imposed by the system that in order to keep calculation of internal room enclosures down to a manageable level, only three internal partitions may meet at any one junction. This suggests that the data structure for internal partitions is organised in a ring structure once the plan has been interpreted.

Windows and doors are selected from a user defineable library of up to twenty-six item types. Once specified, the scale symbol is superimposed on the wall outline. All windows and doors have to be located on walls and partitions already described to the system, and the user cannot specify a free standing aperture. Consequently, during editing routines, doors and windows are moved with their 'host' wall surface.

The geometry creation is carried out in two-dimensions, as indeed is the data structure depicted in Figure 5.3. On interpretation, or consigning a floor plan to the plan chest, floor and ceiling levels are specified by the user. In addition, localised raised areas within a floor can be specified.

ISAAC, the sub-routine for interpreting the geometry description of the building storeys, is described elsewhere.<sup>10</sup>

#### 4.4 Data Structure - Roofs

Whilst the storeys of a building design are treated as an extrusion of the floor plans, the treatment of roof geometries is dealt with as a collection of pitched planes. The user specifies the edges of each plane in turn as either a hip, valley, gable or

an eaves. The roof geometry interpretation program, RODIN<sup>11</sup>, runs concurrently keeping track of the input and warning the user of data inconsistencies. The system permits the user to input the boundaries of the roof planes by the spot heights of the nodal points as an alternative to assigning roof pitches. The data structure used is schematically depicted in Figure 5.4 below.

#### Roofs

Vertices (x,y coordinates)				Edge type (Ridge, Valley, Eave)				Height (by pitch or height)		
plane id nos.	start point	...	stop point	plane id. nos.	edge type			plane id. nos.	pitch	vertex height
1	ax,ay	bx,by	...rx,ry	1	Tub	Tbc	... Tna	1		az bz ... nz
2	px,py	qx,qy	...mx,my	2	Tpq	Tqr	... Tmp	2		pz qz ... mz
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮		⋮
N	rx,ry	sx,sy	...tx,ty	N	Trs	...	Ttr	N		rz sz ... tz

Figure 5.4 Schematic Representation Of GABLE Data Structure For Recording Roofs

#### 4.5 Criticism

GABLE represents a significant achievement in applying micro-computer technology to the problems of integrated CAAD programs, which tend naturally to require large amounts of computer resources. GABLE is available at the Scott Sutherland School Of Architecture, and the following criticisms are levelled from the writer's 'hands-on' experience of using the system.

The imposed division of the geometric data input system into individual components intrudes upon the normal flexibility available with traditional pencil and paper techniques. The user has to start with the external envelope and work inwards. Frequently, the generation of a design happens in the converse manner, working from the inside out.

The necessity for swapping the different program tapes can become a significant nuisance, disruptive to the user's train of thought.

The geometry input system is (necessarily) contrived, and can become frustrating for the user. Although a novice can start producing results from the system in a very short while, the quality of the graphic results produced leaves much to be desired. Drawings can be enhanced by titilating the design drawing by the use of the draughting package program tape, making ammendments to the enhanced drawing is painstakingly awkward, as the drawing enhancements are independent of the design drawing database.

The design system is based upon small load bearing brick-work buildings. There is little provision for framed constructions.

Nevertheless, GABLE remains a usable low cost CAAD system which would appeal to non-expert users. perhaps as an example of 'appropriate technology'.

## 5 GRAMP

GRAMP is the acronym the Graphical Manipulation Package which accepts and manipulates graphic data used for geometry input in SPACES and GOAL programs; the former a planning efficiency evaluation aid, and the latter a general appraisal package. All three of the above mentioned programmes were written and developed by the ABACUS Unit of the University of Strathclyde.

### 5.1 Limitations

GRAMP restricts the permissable geometries to building forms which may be described by the combination of eight volume types illustrated in Figure 5.5 overpage. The common feature of the volumes is their orthogonality; all are right-angular prisms. Subsequently, all geometries are restricted to orthogonal building forms.

The data input files can be created either by manual digitisation, or created at the graphics terminal by use of menus and cross-hair cursor. The GRAMP system is implemented on a DEC10 mainframe computer, using the Tektronix 4010 graphics terminal. There is also the facility for digitising existing plan drawings

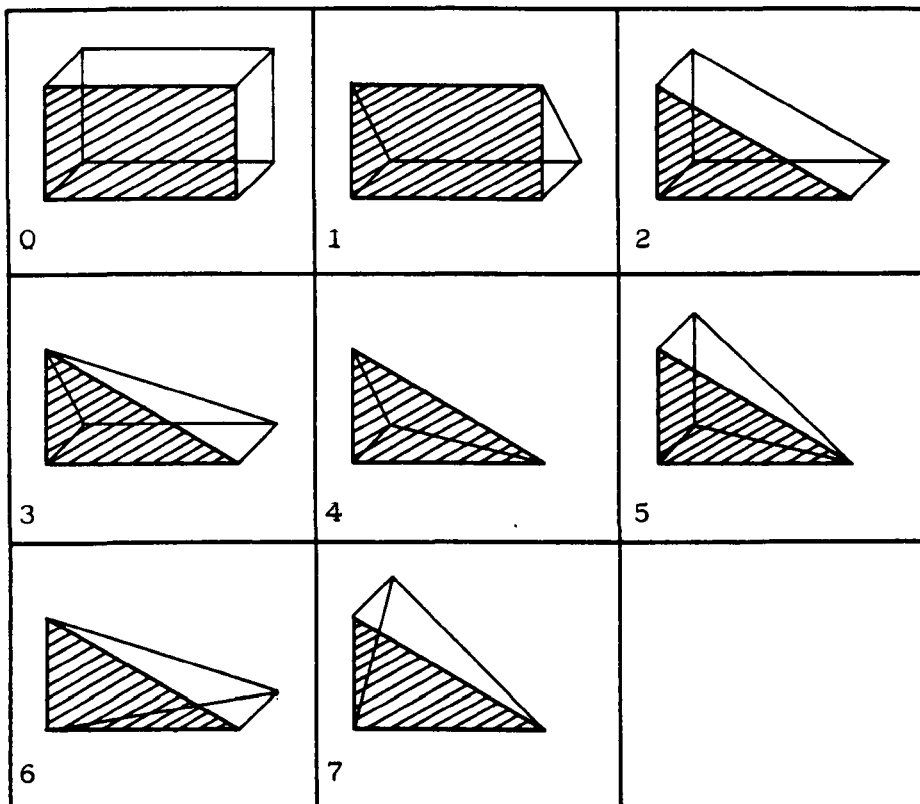


Figure 5.5 Permissible Volume Types For Use With GRAMP

## 5.2 Data Structure

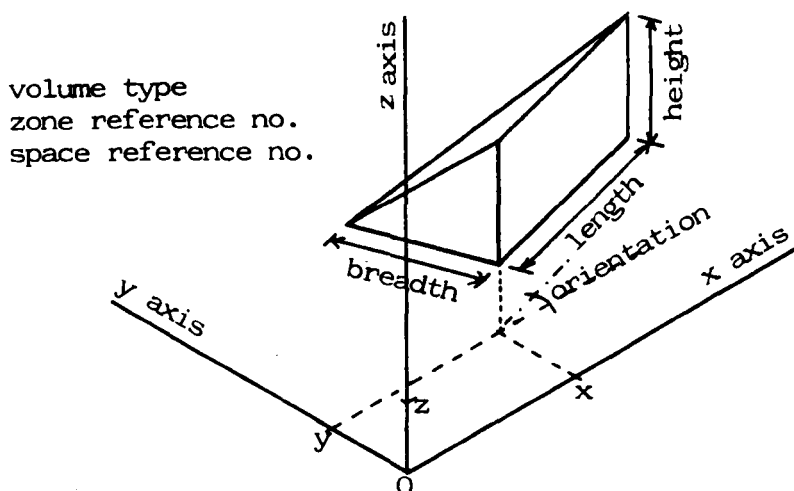
The use of volume types enables the necessary data needed to describe a building volume to be minimised. The volume types themselves need only four bits of data to describe them precisely; the type code, and its length breadth and height. In addition to these three attributes, four variable values are required to enable complicated volumes to be described; the relative displacements along the three principle axes of the object space, and the rotation of the volume.

The volume types are assumed to have their reference point at the bottom left hand corner as the shapes appear in Figure 5.5 above, and lie coaxially to the object space. Rotation is taken by convention to be anticlockwise.

Two additional data bits are added to identify each volume which may be aggregated to a zone.



The GRAMP data structure is summarised in Figure 5.6 below.



a) Illustration Of Attributes and Variables for GRAMP geometry description

zone reference number	space reference number	volume type	angle of orientation	
coordinates of origin		length	breadth	height
x	y z			

b) Format of input data for each volume

Figure 5.6 Schematic illustration of GRAMP Data Structure

A complete data file comprises of a list of data blocks as shown in Figure 5.6b).

Editing and manipulation is carried out on the plan projections of the volumes; shapes may be added, deleted, moved, stretched, and repeated. Volumes are dealt with as discrete entities, and editing of individual lines is not possible. There is no data consistency monitoring, and the user is informed of data file errors at run time. The user is informed which aberrant volume description caused the failure, but he then has to locate and manually rectify the error, by direct edit or graphical interaction.

### 5.3 Criticism

Both GOAL and SPACES program suites are available at the Scott Sutherland School of Architecture, and the ensuing criticisms raised stem from the writer's 'hands-on' experience. However, a fuller critique of GOALS' use for a large construction project may be found elsewhere.<sup>12</sup>

The primary criticisms of GRAMPs' data acquisition and manipulation facilities relate to the restriction to orthogonal building forms, and the aggregation of a limited range of primary shapes into larger aggregate complex volumes. It is often difficult for the architect to construe his design in these terms.

The potential GRAMP user has to have a fairly well advanced concept of his building before he can describe it to the system, and therefore design generation at the keyboard compares unfavourably with traditional pencil and paper techniques. The effort expended in generating the input data often acts as a disincentive for the user to make subsequent alterations, although this is not directly attributable to the computer program.

GRAMP is nevertheless a simple, if stylised, graphic data handling system, which is easily understood. It perhaps works better in theory than in practice.

## 6 Summary

In this chapter the data structuring systems of four graphics based systems have been briefly described insofar as source code of commercially available software is available.

All of the systems described impose either limitations on the user's building geometries, or drawing conventions. They represent a trade-off between simplicity and generality.

References

- 1 Pioneer Of Graphics  
John Lewell  
CADCAM International Dec 1982 p.25
- 2 The Organisational Implications Of CAAD  
Steve Little  
CAD84 Proceedings  
Butterworths Apl 1984 pp.156-164
- 3 Dumb Drawing Systems And Knowledge Engineering  
Aart Bijl  
EdCAAD, University Of Edinburgh Jul 1981
- 4 SKETCHPAD, A Man-Machine Graphical Communication System  
Ivan Sutherland  
Ph.D Thesis, Massachusetts Institute Of Technology  
Jan 1963
- 5 Methods For Treating Variable Shaped Objects In Computer Aided  
Design  
Charles Eastman, and Michael Schwartz  
BoCAAD, University Of Strathclyde No.13 Oct 1973
- 6 Visual Communication And Interaction  
Nicholas Negroponte  
Methodology Of Interaction  
North Holland Publishing 1980 p.46
- 7 Integrated CAAD Systems  
Aart Bijl, David Stone and David Rosenthal  
Final Report to the DoE Funded Research Project 470/12  
EdCAAD. University Of Edinburgh Mar 1979
- 8 The Design Of A Complex Building Using An Integrated CAD System  
E.M.Jones  
CAD82 Proceedings  
Butterworths Apl 1982 pp.353-364
- 9 Towards An End User View Of Design Systems  
R.J.Walters  
PARC83 Proceedings  
On-Line Conferences Oct 1983 pp.17-27
- 10 ISAAC - A Technique For The Automatic Interpretation Of Spaces  
From Drawn Building Floor Plans  
Brian R.Lawson, and J.P.Riley  
CAD82 Proceedings  
Butterworths Apl 1982 pp.663-666

- 11 RODIN - A System Of Modelling Three-Dimensional Roof Forms  
J.P.Riley, and Brian R.Lawson  
Cad82 Proceedings  
Butterworths  
Apr 1982 pp.667-670
- 12 Implementation Experiences Of Design Applications  
L.Kraal  
Parc83 Proceedings  
Oct 1983 pp.65-79

## Chapter 6

### Refinement And Translation Of Raw Graphic Data

1	Review	6.1	117
2	Definition Of The Problem	6.2	118
	2.1 GABLE	6.2	118
	2.2 HUNCH	6.3	119
	2.3 Criticism	6.5	121
	2.4 Proposed Solution	6.6	122
3	Resource Limitations	6.7	123
	3.1 Hardware	6.7	123
	3.2 Firmware	6.8	124
	3.3 Software	6.8	124
4	ENIGMA	6.10	126
	4.1 Data Capture	6.11	127
	4.2 Line Identification	6.15	131
	Sequential Curvature	6.18	134
	Line Of Least Squares	6.18	134
	4.3 Corner Enhancement	6.23	139
	Merging Near Points	6.24	140
	4.4 Shape Identification	6.28	144
	Line Dissociation Algorithm	6.28	144
	Tab Elimination	6.30	146
	Tracing Algorithm	6.32	148
	Closure Check	6.40	156
	Duplicate Erasure	6.42	158
	Alternative Duplicate Erasure Methods	6.44	160
	Area Calculation	6.45	161
5	Summary	6.45	161
	References	6.49	165

## CHAPTER 6

1 Review

The structure and nature of the UK architectural profession has been noted and a description of the architectural design process advanced.

The problems inherent in describing geometries to computers have been examined, as have the follow-on effects of these descriptions to the data structures required to hold the descriptions for CAAD applications.

Examples of the data structures of existing graphic handling computer programmes have been reviewed to illustrate the inter-relationship between data structures and user limitations.

It has been noted in chapter 1, that the uptake of CAAD systems by architects has not been as rapid as that of other members of the building industry. Of the many reasons advanced to explain this reticence, the following factors may be given pre-dominance:

- 1           The relative cost of CAAD systems in comparison to the turnover of the majority of architectural practices.
- 2           The mismatch between CAAD processes and the architects' traditional manner of undertaking his design work, and
- 3           the inherent difficulty of describing geometry to the computer, which requires the architect to think of building description in unfamiliar ways; terms which become clear after the most far-reaching decisions have already been taken.

Whilst there is little that architects can do to influence the cost of computer systems, there is scope for mitigating these latter two disadvantages.

## 2 Definition Of The Problem

To be an aid to architects, a computer system must offer the following features:

- 1           It must make available reliable appraisals of potential design solutions at the stage at which the architect requires them, acting on the information which he has at hand.
- 2           It should not involve the architect in unfamiliar descriptions of building form.
- 3           Operation of the computer system should not distract the architect from his primary task; design.
- 4           It should allow for the evolutionary expansion of the design data base, and above all
- 5           The computer aids must be appropriate to the architects' needs.

The problem then, is to produce a system which fulfils the above requirements.

Several attempts have been made to address these problems, most notably, GABLE, (described in chapter 5), and HUNCH. As will be suggested, these two computer programs are part-solutions.

### 2.1 GABLE

The GABLE suite of programs satisfies the first of the above criteria; the integrated data base permits the user to carry out relatively sophisticated performance appraisals with very little additional effort to that of creating the input geometry data.

It is in the latter area that the system is most open to criticism. Users are limited to creating geometries on the graphics terminal screen through crosshair cursor.

The operation of the system requires constant attendance on the machine which inevitably detracts the designer from the task of designing.

The interstorey interpretation program, ISAAC, is an

excellent and practical concept, if perhaps requiring a too structured data input procedure. The roof interpreting program is even more constrained.

GABLE would be enhanced by a data input system more in line with traditional drawing techniques, such as are described in the remaining chapters of this thesis.

## 2.2 HUNCH

The problem of advanced forms of graphic data input was investigated by the Massachusetts Institute of Technology, and in 1971 produced HUNCH<sup>1,2,3,4,5</sup>, which permits graphical input through the continuous digitisation of sketch drawings. The overall intention was to get the computer to make useful interpretations of sketches without employing a knowledge of the subject being drawn; ie. to generate an understanding from the limited data created by the process of digitisation.

The programs monitor the coordinate positions of either a light pen on a cathode ray tube, or an ink pen on a digitising tablet, and the pressure with which the stylus is pressed against the drawing surface. Further facilities display and allow the manipulation of the data base at various stages in the interpretation process.

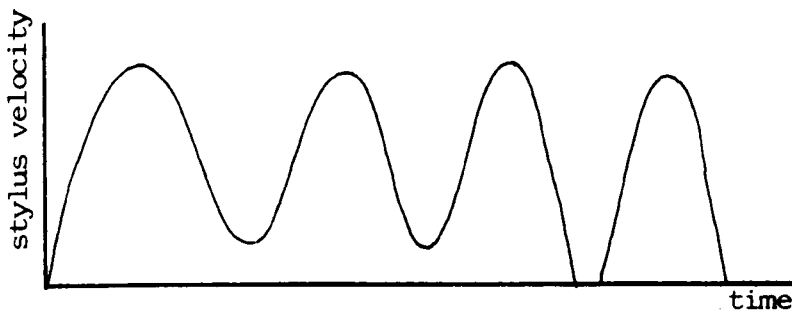
The digitising tablet or screen can be adjusted to send coordinate positions at a fixed rate which can vary between 16 and 200 points per second. Consequently the speed of the stylus between each coordinate pair is determined. Monitoring the velocity of the stylus enables corners to be deduced on the assumption that the minima of the velocity time graph of the stylus accords with a change of direction as illustrated in Figure 6.1 overpage.

Having interpolated the point in time at which the turning points are passed through, the coordinate pairs separated by two such corners can be subjected to a line of closest fit algorithm and straight lines superimposed on the sketch drawing.





a) Coordinate points monitored by the digitiser



b) Velocity/time plot of the stylus during digitisation

Figure 6.1 Stylus Monitoring During Continuous Digitisation Of Sketches By HUNCH

However, to take account of curves, the path of the stylus is also monitored for 'bentness'. Bentness is defined in Figure 6.2 below.

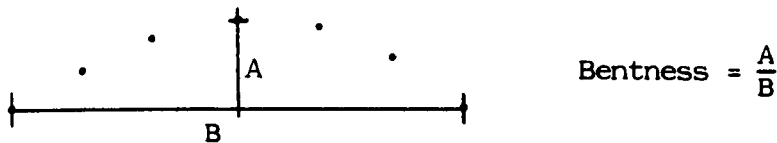


Figure 6.2 Definition Of Bentness In HUNCH

Where supposed corners coincide with a high degree of bentness, or bentness greater than a preset critical value, changes in direction are confirmed. However, the number of points in the base line (B) used in the bentness calculation will have a significant effect upon the nature of the curve which the system will

act upon; the greater the base line the more severe will be the curvature necessary to confirm a change of direction. Consequently, the number of points is a parameter whose value has to be set to suit the users 'fist'. Clearly the HUNCH system is construed as dealing with linear drawings, and the identification of slow arcs seems tenuous. Slightly overlapping lines and line ends in close proximity are made to join at their intersections. Again a user entered tolerance factor determines how small a separation will be before this enhancement is enacted.

The other feature of the drawing monitoring, is the pressure applied to the stylus, and the overscoring of lines. In the first case, a high pen pressure indicates greater importance being attached to the line by the user than those drawn with a lesser pressure. Similarly, overscored lines are allocated greater importance than single lines.

The enhanced line drawings are processed to produce enclosed spaces, and the data is held in a hierarchical data structure at four levels; object, space, enclosing lines, and node points.

### 2.3 CRITICISM

Both GABLE and HUNCH attempt to interpret two-dimensional representations of three-dimensional objects. GABLE overtly treats solids as extrusion of their plan forms, whilst the later versions of HUNCH relied on the user to supply additional information to the system interactively to enable the computer to decode the drawing database.

"The next question to be asked is 'What balance can be struck between an intelligent but unwieldy system, and a tiresome but practical one?'" 5

ie. between HUNCH and GABLE.

Whilst it is recognised that a total building description is an end in itself so far as the architect is concerned, the derivation of the description requires that specified attributes

'indexed' by the geometric description: The specification defines 'what' a thing is, and the geometric description describes 'where' it is, and how 'it' relates to all of the other things in the data base. Later implementations of HUNCH incorporated a logical data structure based upon the CONNIVER logical language, which attempted to permit the simultaneous top-down decomposition of the data and the bottom-up association of graphic drawing data, and perhaps this later development influenced the earlier interpretation system. In the event, the concern with intent and emphasis of lines in a drawing would appear to make HUNCH inappropriate for practical applications.

GABLE is a closed system. The data acquisition serves the needs of its embodied appraisal and other applications programs. As such it is a success, but it would be a great advantage if the geometric description were 'portable' for implementation in other systems, and there seems to be no reason why this should not be done. The shortcoming of the GABLE system is in its fixed method of data entry which does not accord well with traditional techniques. There is no reason to consider traditional techniques sacrosanct, but nevertheless, the problem identified is the need to encourage the use of CAAD by architects.

The idea of contextual data structures, as embodied in the more recent versions of HUNCH are still experimental. Whilst not invalid, there is the danger that the context becomes rigid, as for example, the context in GABLE which although unstated, limits building descriptions to loadbearing wall structures.

## 2.4 PROPOSED SOLUTION

In order to satisfy the problem stated on page 118, the approach of HUNCH is accepted insofar as the interpretation of sketch data. The circuitous methods by which HUNCH does so are rejected in favour of straightforward geometric testing. The delineation between inner and outer enclosures used in GABLE are also rejected in favour of automatic determination.

Such an interpretation system would help encourage

the adoption of computer aids by;

- a            permitting the description of building form to the computer at the earliest stages of the design process, thus permitting access to appraisal aids at the point at which they can provide the most benefit.
- b            exactly matching the architects' traditional pencil and paper techniques, although providing him with the option of alternative forms of data entry if he so wishes.
- c            in the case of continuous digitisation of sketches, not distracting the designer from his design task, and
- d            allowing for the open ended expansion of the buildings geometric description.

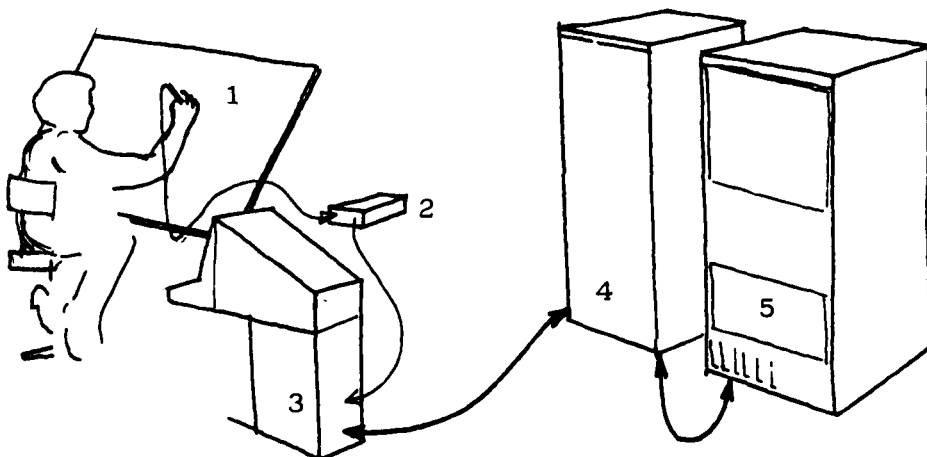
### 3    Resource Limitations

All projects are necessarily constrained by the resources available for their realisation. For projects involving computer graphics, the limitations fall under the headings of hardware, firmware and software.

#### 3.1    HARDWARE

Although there are three types of computer processor in the Scott Sutherland School of Architecture, ( a range of small 8-bit microcomputers, a range of Tektronix 4050 series desk-top computers, and a DEC20 36-bit mainframe computer installation located at a central RGIT facility ), selection of the computer system was dictated by an external consideration. The digitising tablet, (a Tektronix 4954,A0 size), is controlled by a circuit board resident in a Tektronix 4010 graphics terminal. The terminal has no local processing capacity, thus the user is constrained to use the DEC20 mainframe computer accessed by modem.

The resulting hardware configuration is illustrated in Figure 6.3 overpage.



Key:

- 1 Tektronix 4954 digitising tablet
- 2 pen/tablet link unit
- 3 Tektronix 4010 vector graphics terminal
- 4 Gandalf Modem
- 5 DEC20 mainframe computer processor and secondary data storage.

Figure 6.3 ENIGMA Hardware Configuration

### 3.2 FIRMWARE

The DEC20 computer is operated by the TOPS20 operating system. It also has compilers for most high level programming languages, and also machine code facilities. The operating system also permits access to public access subroutine libraries.

Several hundred remote terminals distributed throughout all RGIT sites are linked to the DEC20 by an intelligent modem, GANDALF.

### 3.3 SOFTWARE

As previously stated, the DEC20 supports most high level languages. However, although the digitising tablet may be used to draw on the Tektronix 4010 terminal off-line as a dumb drawing system, when used on-line it is driven and accessed by calling routines written in FORTRAN.

Figure 6.4 below illustrates the interdependence of the

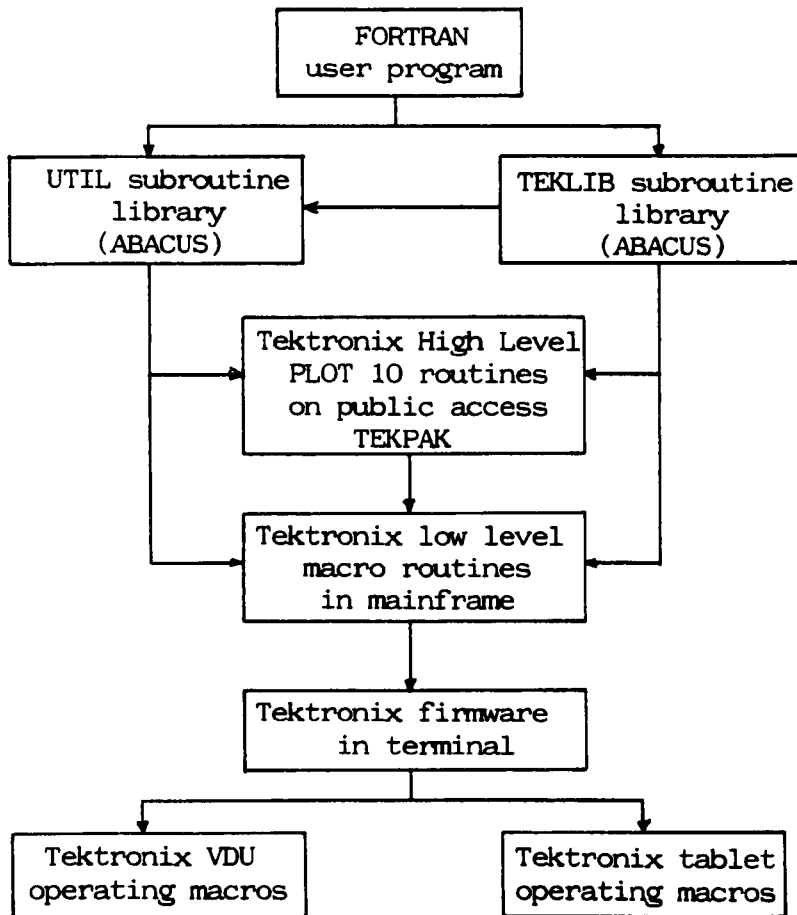


Figure 6.4 ENIGMA Software Configuration

various sources of coding. A users graphic application program may draw on existent routines in public access libraries. In this case, there are two sources of FORTRAN subroutines written by the ABACUS Unit of the University Of Strathclyde, UTIL and TEKLIB. The former contains subroutines which carry out such functions as the setting up of control parameters to initiate the correct presets for the various hardware devices in the configuration, opening secondary files and setting the communication baud rate. These functions have to be invoked only once in the program. The latter library contains routines such as drawing arcs, chained lines etc. However, some TEKLIB routines call upon routines from UTIL. Both of these libraries routines call upon a further library of routines resident in the system. These are the Tektronix PLOT 10

high level language graphics routines. However, some UTIL and TEKLIB routines are written directly into machine code or similar low level language.

On compilation the body of user code and subroutine libraries are interpreted into machine code, which will concatenate those subroutines already in machine code. The machine code sequences, or macros, correspond with the pulsed signals which communicate with the terminal.

The terminals firmware decodes these sequences into device operating instructions, and opens channels to other devices as and when called.

Other libraries of subroutines are available. However, those in GINO-F and PICASSO are designed more for application to draughting systems. They require more complex information, eg. line thickness and type etc. whilst limiting access to peripheral devices.

#### 4 ENIGMA

Using the resources outlined above, computer programs have been developed as a step towards overcoming the problems which discourage architects from adopting CAAD aids.

The suite of computer programs is called by its acronym ENIGMA; an Enhanced Interpretive Graphics Module for Architects. It interprets raw graphic sketch data created by the designer either at the digitising tablet or on a graphics terminal, and compiles a hierarchical data structure which is transportable between differing application programs. Graphic data may be drawn in any order prior to the interpretation taking place. Building geometries are treated as extrusions of their plan forms.

The logic of the program is explained in the remainder of this chapter. The processes described fall into four sequential procedures; data capture, line identification, picture enhancement, and shape identification.

#### 4.1 DATA CAPTURE

This section of software, written as a subroutine, operates the tablet through the use of subroutine libraries as previously described.

A sketch is a freehand line drawing on a two-dimensional surface. Lines may be drawn in any order, with any degree of curvature. Moreover, several lines may be drawn continuously without lifting the pen from the paper. Individual lines may cross, almost meet, or be totally independent of one another.

In the context of an architects drawing carried out by the continuous digitisation of points, we may make three assumptions;

- a all freehand sketched lines can be described as a sequence of small straight line segments joining each pair of sequentially recorded points,
- b lines which almost intersect are intended by the drawer to intersect, and
- c all lines which cross leaving a short tab protruding are intended by the drawer to meet exactly.

The significance of these assumptions will become apparent in due course.

The Tektronix 4954 digitising tablet has a sensitised surface measuring approximately 1 000 x 750 mm. Beneath a protective top surface is a mesh of wires which divide the surface into a grid of 4096 x 3072 squares, ie. roughly four wires per millimeter. This provides a resolution to half a grid square; just less than  $\pm 0.125$  of a millimeter. In the 'continuous mode' of operation, the tablet returns a three part locational record at an approximate rate of five positions per second.

The three parts of the locational record consist of a 'header' character, followed by an 'x' and a 'y' coordinate relative to the bottom left hand corner of the digitising surface. The coordinate values are given as integers.

Figure 6.5 overpage illustrates the function of the header characters.



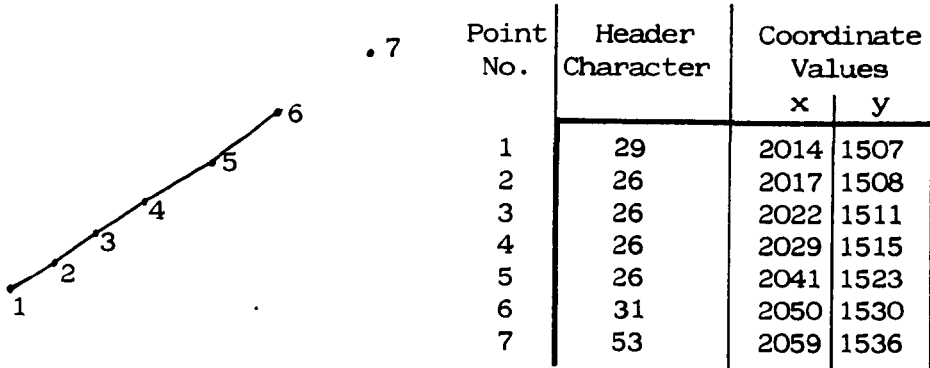


Figure 6.5 Data Output From Tablet In Continuous Digitisation Mode

Here, the typical output produced by the digitisation of the line shown on the left, is presented in the table on the right, (enclosed by the heavy line).

The header character value 29 indicates that the stylus has just been brought into contact with the digitising surface and pressure has been applied; ie. the first point in a sequence. The value of 26 indicates intermediate points along the line. The value 31 indicates that pressure has been released although the stylus is still in contact with the surface; ie. the last point in a line. The value 53 indicates that the stylus is still being tracked by the tablet, but it is not in contact with the surface.

As described in chapter 3, section 4.2.11b) on page 47, this data triad can become corrupted due to several causes. Consequently, a data validation routine has been included in the data capture program to mitigate the occurrence of corrupted data. This is done by filtering out all header characters with their associated coordinate data other than those with the values of 29 and 26. This means that separate lines are distinguished by the occurrence of a 'new line' header in the data list.

A second feature which requires some controlling action is the rate at which data can be generated by the tablet, too much of which might rapidly fill the available data storage, and moreover increase the amount of computation required for the analysis of the data. Furthermore, small incremental changes in position due to

a high frequency rate is more likely to induce logic errors in the line identification routine. The slower the stylus travels across the drawing surface, given a fixed rate of data sampling, the smaller the space intervals between sampled points. At the start of a line when the stylus is being accelerated, and again at the end when it is decelerated, the intervals between the sampled points decreases accordingly. Moreover at these periods the users hand might be shaking, and the lateral movements become significant in relation to the interval. The sensitivity of the gradient of a line to the ratio of lateral movement and base length is demonstrated in Figure 6.6 below.

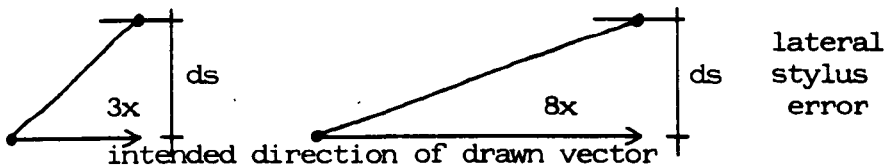


Figure 6.6 Illustration Of Sensitivity Of Gradient To Stylus Error In Relation To Interval Length

Without rectification, the ends of a line would be less reliable than its middle. However, in practice the nodal points of the lines are generally more significant than their centres.

To eliminate this source of potential error, a minimum base length interval is introduced into the algorithm. Unless the interval between the last recorded point and that currently monitored is greater than 10 tablet units (2.5mm approximately), the current point is ignored. This ensures that recorded points do not bunch up at the start and end of a line.

Figure 6.7 overpage shows the logic flow diagram for the subroutine 'TABLIT' which controls the data capture system, eliminates spurious header characters and imposes the minimum interval. The algorithm also draws the picture as it is digitised onto the terminal screen so that the user can check that the picture has been correctly captured.

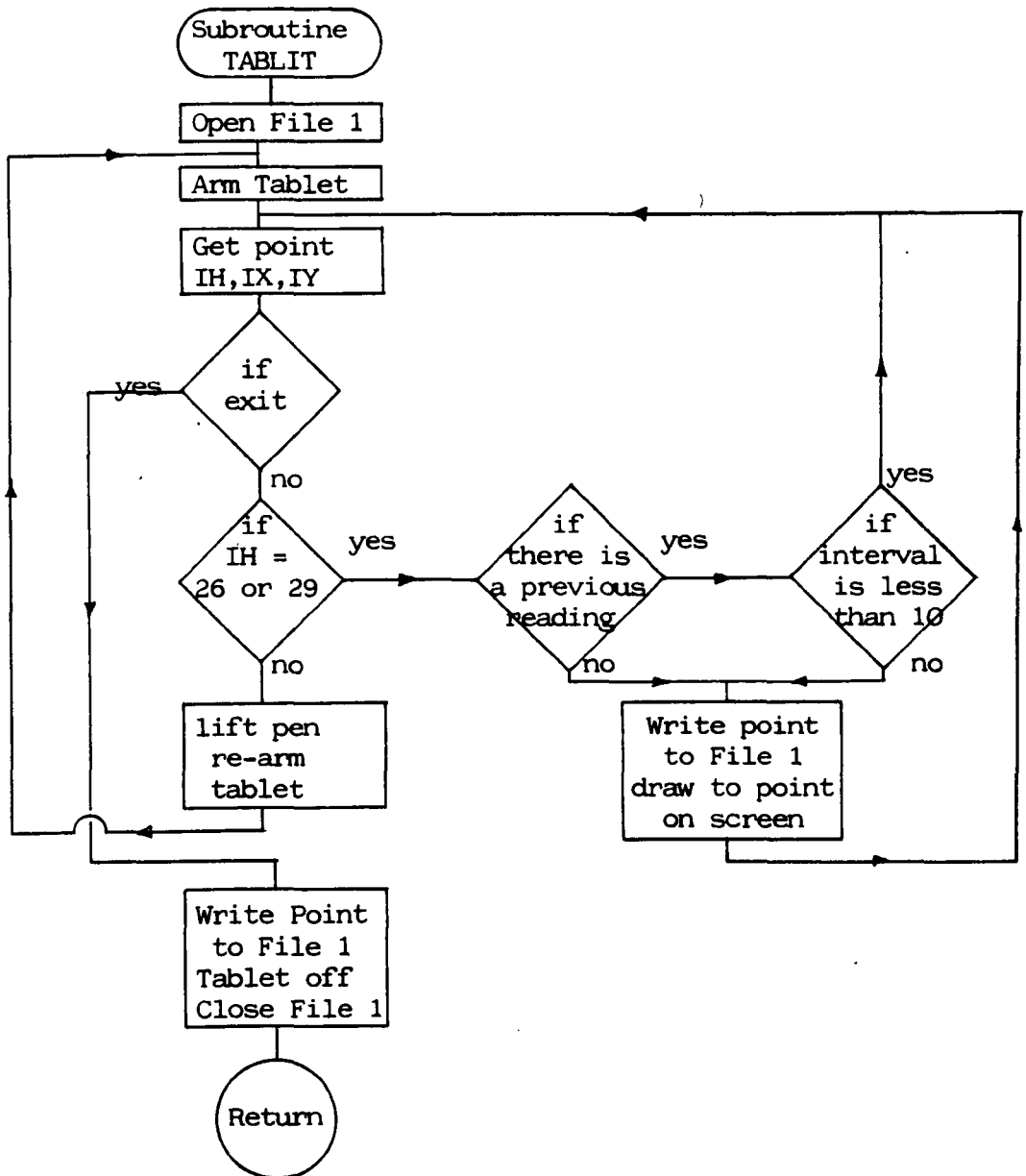


Figure 6.7 TABLIT - Data Capture And Validation Algorithm

Once the stylus has been taken out of 'presence', i.e. the stylus breaks contact with the digitiser, the tablet has to be re-initiated before further digitisation can begin. This involves a small time delay, and the user has to watch the 'ready' light on the tablet link box before he can proceed.

One problem yet to be overcome is the occasional loss of the last buffer of data returned to the sequential access data file. The data is buffered in the graphics terminal until the DEC20 computer is ready to receive and process the data, (as is the case with all time-sharing computers). It is not yet clear whether this problem resides with the Tektronix 4010 terminal, the TOPS20 operating system, or the GANDALF modem. This problem has only been experienced since additional core memory was added to the DEC20 in December 1983. This loss of data is awkward, requiring the picture to be redrawn until it is successfully captured. Current work is being carried out to overcome the problem by permitting the picture to be overdrawn.

#### 4.2 LINE IDENTIFICATION

Following the digitisation process, the triad coordinate data are stored in a list on a sequential access file in the form illustrated in Figure 6.8 below.

FILE TBDAT:DAT				
(start header for 1st line) →	29	850	1439	
	}	26	849	1422
(intermediate headers of		26	848	1394
1st line)		26	846	1368
		⋮	⋮	⋮
		⋮	⋮	⋮
	26	1122	966	
	26	1123	949	
( start header for 2nd line) →	29	831	958	
	}	26	848	953
( intermediate headers of		26	873	948
2nd line)		⋮	⋮	⋮
	⋮	⋮	⋮	
('x' coordinates)		→	→	
('y' coordinates)		→	→	

Figure 6.8 Data Format As Output From Subroutine TABLIT

It will be seen that all coordinate values are positive, integer and have values between 0 and 4096 (on the 'x' axis), and 3072 (on the 'y').

It is unlikely that an architect would draw to a scale of more than 1 to 2500, (the scale of an Ordnance Survey map). Thus the maximum numerical values of distances represented on the digitising tablet will be no greater than 2560.000 meters, ie. the width of the tablet being 1.024 meters at a scale of 2.5 m per millimeter to a scale of 1:2500). Consequently, when values are converted from integer to real, a FORTRAN data format of F9.3 will be adequate.

The picture as recorded is in the form of a great number of small line segments which represent a fewer number of straight lines. The segments are combined into these larger straight lines by the subroutine LINEID, whose logic flow diagram is shown in Figure 6.9 overpage.

Each line segment has its gradient calculated and compared to that of the segment preceeding. If the two gradients, expressed as an angle are equal to within a user specified tolerance value, the two segments are concatenated, and the line of closest fit through the points calculated, whose gradient is then taken as the test for the next segment. This will be explained more fully in due course. The tolerance value between the gradients within which concatenation takes place has a default value of  $\pm 5^\circ$ , although an experienced user may adjust this value to suit his 'fist'.

In actual fact the gradient calculation is not straightforward. Vertical lines produce an infinite Tangent value, and therefore have to be filtered out for special treatment. Any occurrence of a zero interval on the 'x' axis indicates a vertical line, and by examining the direction of the 'y' interval,  $+90^\circ$  and  $-90^\circ$  are easily deduced.

A second problem which has to be overcome occurs because Tangent functions do not distinguish between the first and third, and second and fourth quadrants, which is a potential problem when comparing the gradients of the line segments. The problem is overcome by the introduction of a simple test. The angles which the two segments make with the horizontal are multiplied together

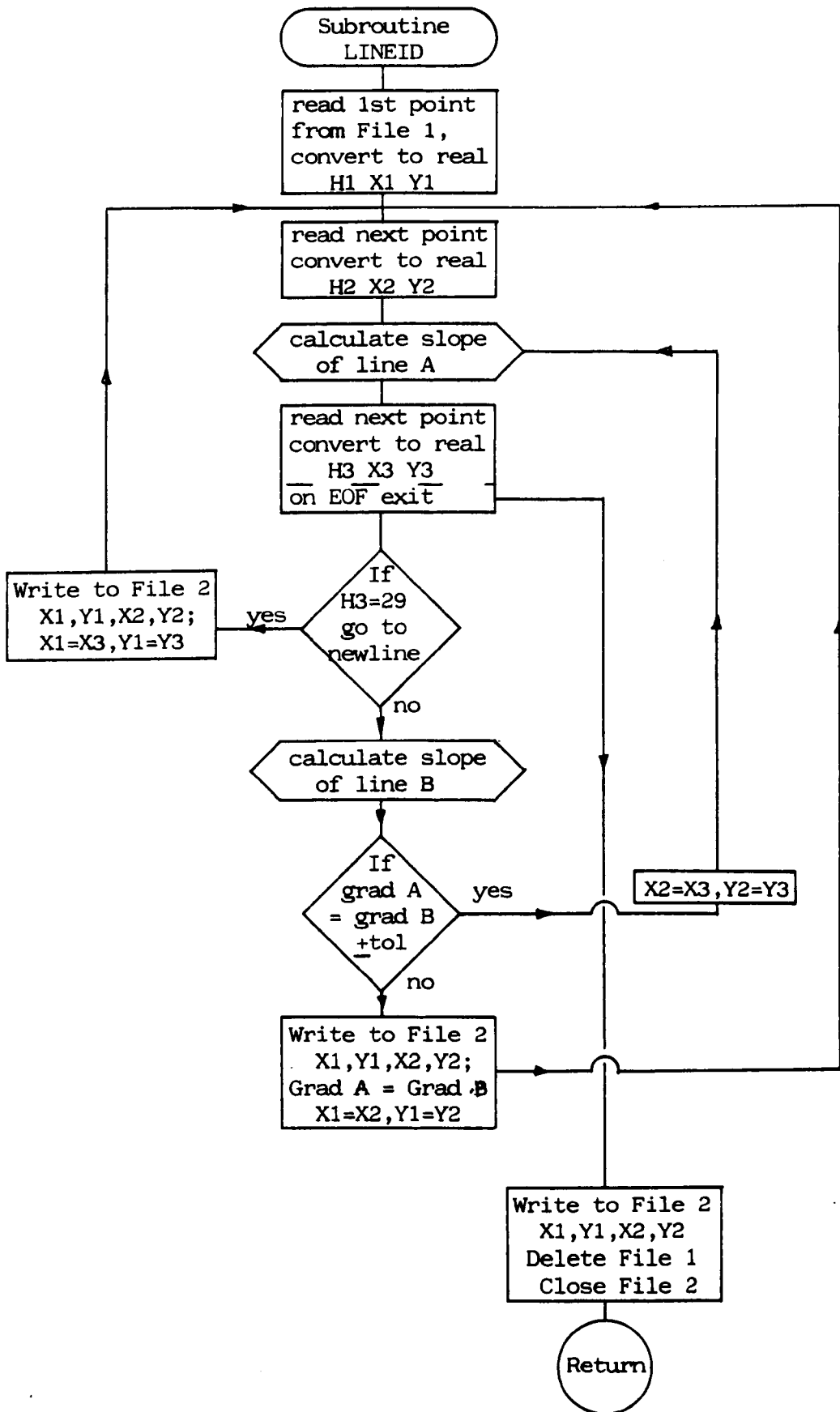


Figure 6.9 LINEID - Flowchart Of The Algorithm For Separating Distinct Lines From Sketch Data.

and, if the product is negative, then the segments are in adjacent quadrants and their angles are added to give their angular deviation, otherwise they are subtracted.

Figure 6.9 therefore gives a highly simplified illustration of the calculation involved, and the calculation of the segments slope is a subprogram in its own right.

Further problems involve the treatment of arcs, and sequential curvature. These relate to the difficulty of deciding when a slow curve should be interpreted as a straight line and visa versa. The category of problem is illustrated in Figure 6.10 below.

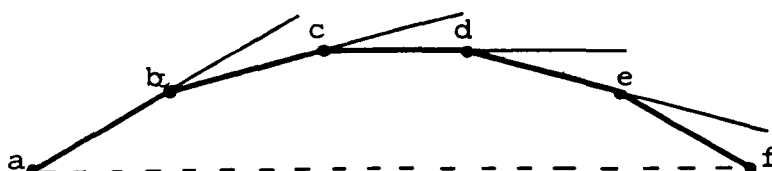


Figure 6.10 Illustration Of Sequential Curvature

Here the differences in gradient between any sequence of two cojoined line segments is sufficiently small as to be interpreted as being members of the same straight line, thus the series abcdef is interpreted instead as the line vector af.

To mitigate the effects of sequential curvature in the line identification algorithm, the LOLS (line of least squares) through the previous points on the straight line was incorporated. This has the effect of attenuating curvature, thus increasing the sensitivity of the algorithm to changes of direction.

The Line Of Least Squares' gradient, (m), is given by the equation -

$$m = \frac{\sum XY \times N - \sum X \sum Y}{\sum X^2 \times N - (\sum X)^2}$$

Where X and Y are the numerical coordinate values of

line graph, and N the number of points upon it.

It will be noted that when encoded into a calculation routine there are occasional instances of misleading results being generated. This may be verified by the test program whose coding is listed below, and the input data fed to, and the results generated by it, are presented in Figure 6.11 overpage.\*

```

          DEMO.FOR
0100      OPEN (UNIT=20,FILE='PLANO1')
0200      FD=1
0300 15:   FORMAT(3I)
0400 35   FORMAT(2I,1F)
0500      READ (20,15) IH,IX,IY
0600      SX=IX ;SY=IY ;SX2=SX*sx ;SXY=SX*SY
0700 30   READ (20,15,END=10) IH,IX,IY
0800      IF (IH.EQ.29) GO TO 10
0900      SX=SX+IX ;SY=SY+IY ;SX2=SX2+IX*IX
1000      SXY=SXY+IX*IY ;FD=FD+1
1100      CC=SXY*FD-SX*SY ;DD=SX2*FD-SX*SX
1200      IF (DD.EQ.0) GO TO 20
1300      GD=ATAN(CC/DD)*57.2957805
1400 25   WRITE (5,35) IX,IY,GD
1500      GO TO 30
1600 C
1700 20   GD=90.0 ;GO TO 25
1800 C
1900 10   CLOSE (UNIT=20,FILE='PLANO1')
2000      STOP ;END

```

#### Listing Of Line Of Least Squares Demonstration Program

The graph of the input data shown in Figure 6.11 shows what is clearly a 'vertical' freehand line, whilst the line of least squares calculation produces, over the first eight data points, angles which fail to describe the line. One possible reason for this anomaly is the diminishing confidence with which the 'x' values may be placed upon a line as it approaches the vertical, which may lead to ambiguous cases being encountered. Although this is not a frequently occurring problem, the possibility of these ambiguities exist, and for that reason it is necessary to employ both the precedingly described tests in sequence, in order to deal with arcs.

Of the two tests of line continuity, the one by comparison of the consecutive changes in the segments' gradient, and the other by comparison of a segments gradient to that of the LOLS through

\* A fuller discussion is presented in Appendix 2, page 238.



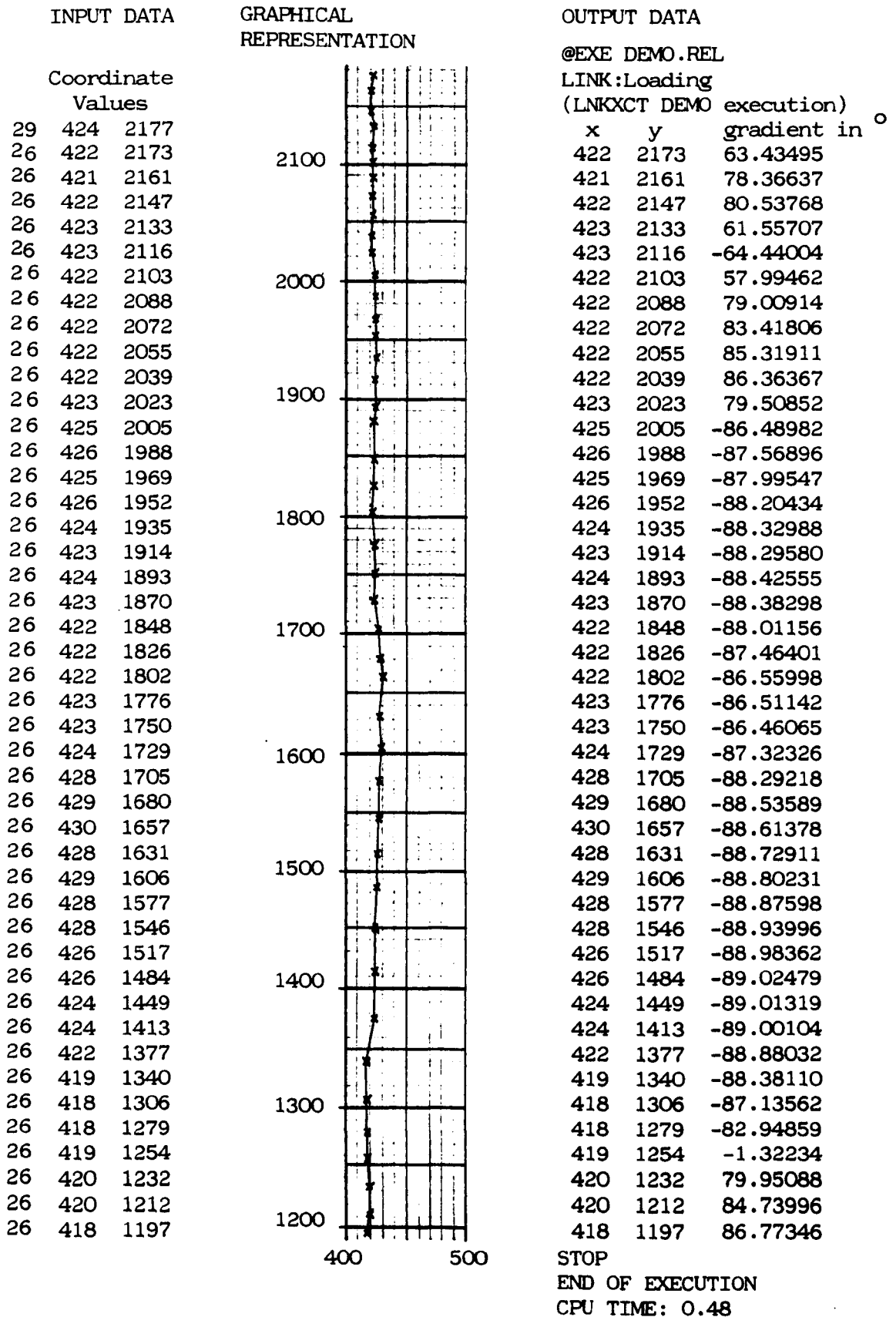


Figure 6.11 DEMO Program Input And Output

the preceding points, the former is more sensitive to changes in direction than the latter. This can be demonstrated by reference to Figure 6.12 below.

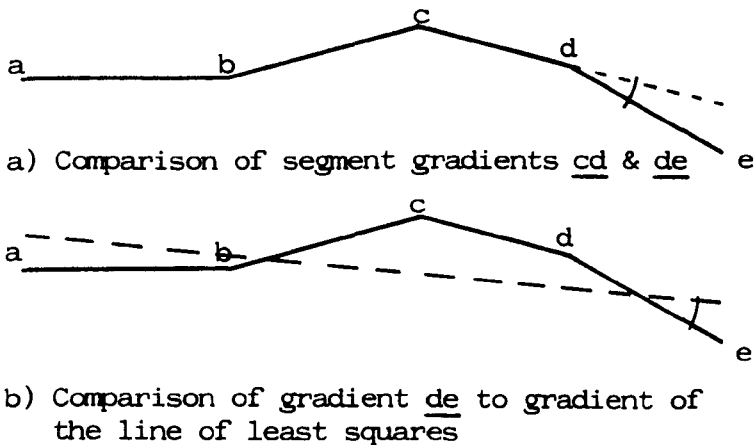


Figure 6.12 Comparison Of The Sensitivities To Changes In Direction By Comparing Gradients Between Segments And LOLS.

The deviation to produce a significant swing in the LOLS by a segment will have to be sufficiently large such that it would have already failed the test by comparison of segments. Thus any test by b) in which a swing greater than the declared tolerance value is detected must necessarily also be detected by method a). Therefore, by applying both tests in the line identification algorithm, spurious gradients produced by the LOLS process will be apprehended, and its effect attenuated.

When a change of direction is discovered to occur, the coordinates of the start of the line formed by the concatenation of the line segments recognised as continuities, and the coordinates of the point at which the change of direction occurs are written away to a sequential access file as the start and stop points of the discovered line. Unless a new line header is the next value to be encountered, the stop point of the last line becomes the start point of the next.

Similarly, when a new line header is encountered, the prior coordinates become the end point of the last line irrespective, and the new line point becomes the start coordinates of the next.

If neither a new line header is encountered, nor is a change of direction determined, then the point is accounted a continuity of the existing line and the segment concatenated to it.

The algorithm also compiles a companion file which stores additional information about each discovered line. The data stored at this juncture is shown in Figure 6.13 below.

position in sequential	LINE DATA				COMPANION FILE			
	coordinate data				flag setting 1,2,3,4	number of points	Line of Least Squares Data	
	start point		stop point				gradient	constant of intersection
x	y	x	y					
1	Xa1	Ya1	Xa2	Ya2	1	A	m1	k1
2	Xa2	Ya2	Xb2	Yb2	2	B	m2	k2
3	Xb2	Yb2	Xc2	Yc2	3	C	m3	k3
4	Xc1	Yc1	Xc2	Yc2	1	D	m4	k4
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
n	Xn1	Yn1	Xn2	Yn2	3	N	mn	kn

Figure 6.13 Output Data Format From Subroutine LINEID

The 'flag' setting indicates one of four possibilities; the line is the first in a chained sequence, it is intermediate in a sequence, it is the end of a sequence, or it is independent of any sequence. This information is necessary for enhancing corners of line intersections as will be described in the next section.

Also recorded are the number of points lying on each identified line. This reflects the confidence with which each line's existence is asserted, and is also necessary for use in the enhancement routines, as are the values for the general equations of the LOLS.

It will be noted that this scheme records arcs as a series of line segments. Slow curves will be described as longer lines forming a polygonal shape, whilst sharp curves, particularly over substantial arcs, might well be lost.

## 4.3 CORNER ENHANCEMENT

Because in continuous digitisation mode the tablet is sampling the mesh nodes traversed by the stylus, there is a high probability that junctions between two lines will become rounded or lost as illustrated in Figure 6.14 below.

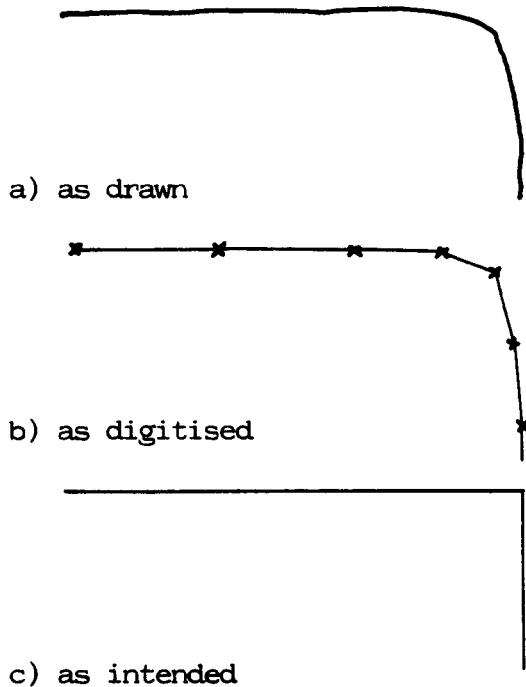


Figure 6.14 Corners Lost As A Consequence Of Continuous Digitisation

When the coordinate data has been analysed by the LINEID subroutine, these 'lost' corners will be represented by a small number of line segments with only two or three points along their lengths, (including those at their ends). Additionally, they have a flag value of 2, ie. they are in a sequence or chain of lines, in the companion file.

It is assumed that where a couple of such lines exist in between two longer lines having significantly more data points along their lengths, the intention of the user was that the two longer lines were to meet at their intersection, and that the intervening segments are due to the digitisation error. If, however, there is a chain of more than three small three point segments, it

is assumed that these represent an intended arc. These assumptions are reasonable, since the length of these segments are short in comparison to the concatenated vectors between which they lie. The number of segments, and number of points thereon, may be specified by the user to suit his 'fist' so that the corner enhancement subroutine can filter out these segments, and reconstruct the corners.

Where a lost corner is detected, the point of intersection given by the interaction of the two lines of least squares, whose constant values are held in the companion file, is calculated. The list of lines in the sequential store are rewritten to a second line file, inserting the new line end coordinates and skipping the intermediate segments. The logic flow diagram for the subroutine, CORNER, which rebuilds lost corners is given in Figure 6.16 overpage.

#### Merging Near Points

Originally it was assumed that it would be adequate at this stage of the process, to enhance the drawing simply by moving the end points of the recorded lines to the intersection points given by their line of least square equations, thereby replacing the inexactitude of a freehand sketch with the precision of its line drawing equivalent. The first implementation of an algorithm to achieve this revealed a class of problem that required solution. This condition is illustrated in Figure 6.15 below.

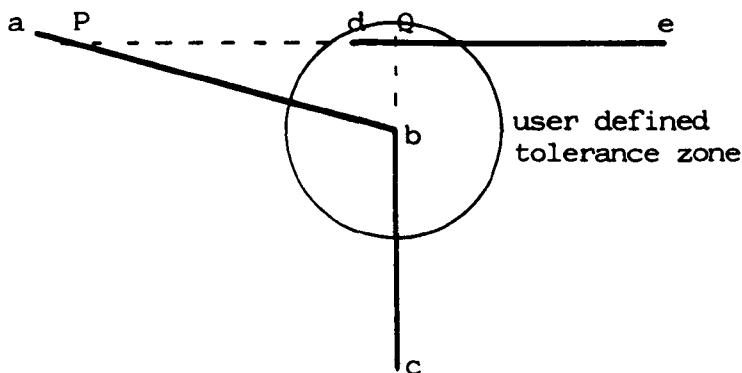


Figure 6.15 Line Intersections Involving Near Parallel Lines

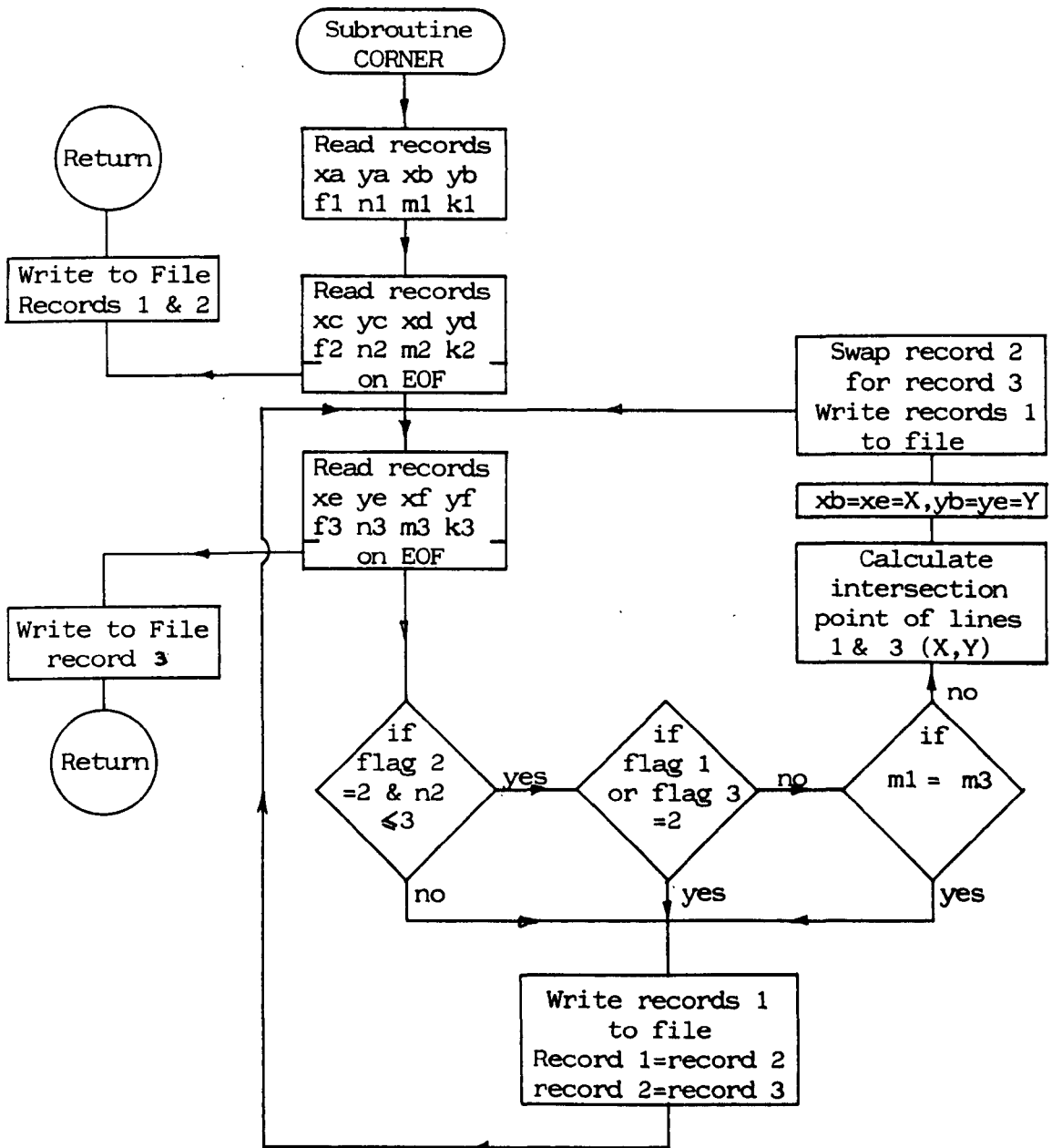


Figure 6.16 CORNER - Flowchart Showing The Corner Replacement Algorithm.

Suppose that the least squares analyses describe the three lines ab, bc, and de as shown in in Figure 6.15 (page 6.24). Point b has been reconstructed, and the calculations give the points of intersection between ab and de as P, and bc and de as Q. If the interval bQ is small, we may conclude that the user intended all three lines to meet at a point. Projecting all the lines to their intersection points would produce a spurious addition to the triangle PQb. On the other hand, taking the centroid of the triangle as the junction of the three lines would produce serious distortions to the picture.

The problem is resolved by a subroutine, MERGE, which reads through the line data list, taking one line at a time, looking for end coordinates 'near' to those of the line under consideration. 'Near' in this case is specified by the user, otherwise a default value proportional to the scale at which the drawing is being carried out is used. Once all the near points have been identified, they are averaged out, and the new values overwritten in the line data list.

Should any near points be encountered, then the line of least squares description of the lines involved are made void. Inevitably the implementation of the MERGE subroutine introduces some distortion to the drawing. However, it is less than would be produced by the straightforward implementation of the alternative. There is a possibility that if too large a tolerance value is selected, serious distortion can arise, since distant points might be 'dragged' in to the mergeing sequence. In practice it would seem that the advantage of dealing with the interaction of near parallel lines, a situation frequently encountered in architectural plan sketch drawings, outweighs the potential disadvantages.

If, however, near points lie outside the tolerance zone, the lines will remain unaltered.

The logic flowchart of the MERGE subroutine is given in Figure 6.17 overpage.

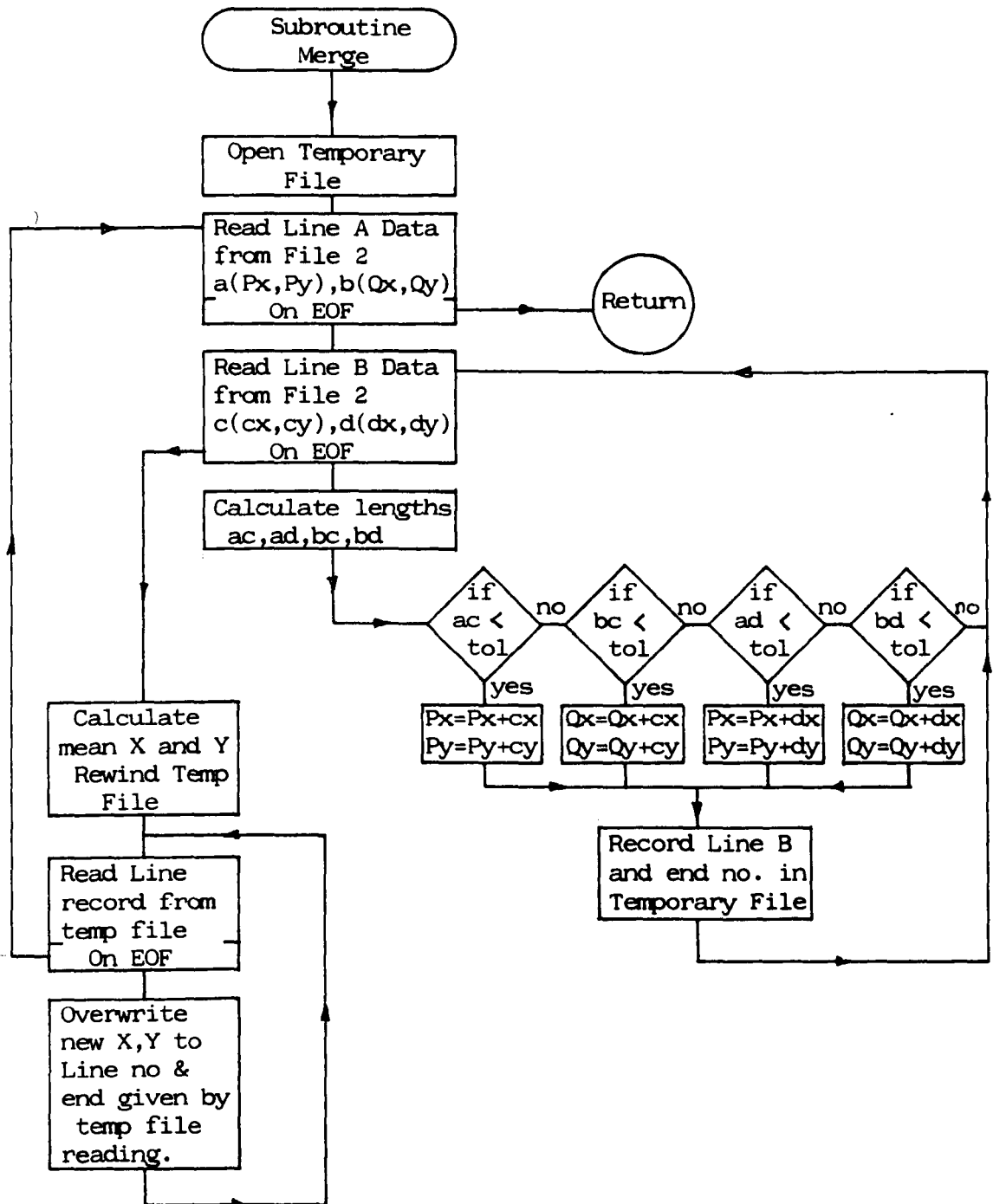


Figure 6.17 MERGE - Flowchart Showing The Near Point Merging Algorithm.



#### 4.4 SHAPE IDENTIFICATION

At the early design stage, many of the appraisals which the architect might want to carry out involve processing data relating to the individual spaces as represented by enclosed shapes on a plan drawing. However, the data as stored by the system is unlikely to be in a form from which shapes can be extracted until a further series of manipulations are carried out upon it.

First the lines have to be dissociated. The lines of the drawing might cross and intersect one another, and these lines have to be split into discrete line elements.

Second, lines not forming part of a shape have to be sorted; ie. lines connected at only one end. These lines will be referred to hereafter as 'tabs'.

Thirdly, shapes have to be traced. This process itself entails four sub-processes;

- a a trace navigation system based upon a heading calculation,
- b the generation and management of a secondary data file,
- c a checking system to determine when a shape has been circumnavigated by the trace, and
- d a checking and elimination system to ensure that a shape is discovered only once.

Finally, the areas of each shape is determined in order to identify the enclosing drawing outline, ie. the shape with the greatest area.

##### Line Dissociation Algorithm

The principle behind the trace logic is to select any start line and then determine the lines which join it at its far end. The line which forms the least, (or greatest), angle between the pair is selected, and the trace moves on to the far end of the line and repeats the process until the trace arrives back at its starting point.

There are six possible relationships between any pair of

lines held in the data set so far described, and these are shown in Figure 6.18 below.

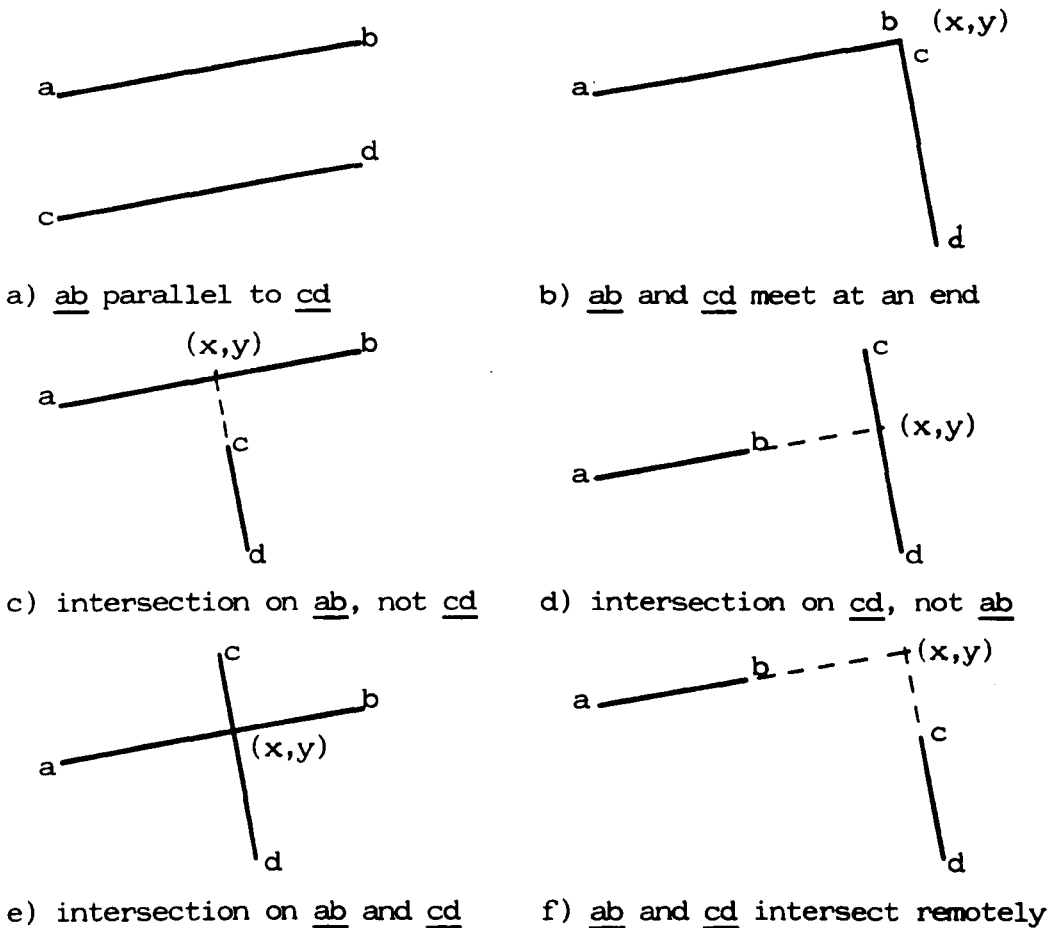


Figure 6.18 Intersection Possibilities Between Any Two Vectors

Of these possible relationships, only cases a), b) and e) are of direct interest to the tracing algorithm.\* Parallel lines have to be noted since they will not intersect, and must be prevented from being accessed by the point of intersection calculation, thereby causing a program failure. Case b) can be determined simply by comparing end point coordinates.

\* Cases of indirect intersection c), d) and f) will be of interest if the system is extended to take account of drawing construction lines.

The intersection determination subroutine, INTEREX, takes each line in turn, and then compares it sequentially with every other line. Only if the two lines under consideration intersect either at a common end, or along their lengths is any further action taken. In this situation, the number of nodes which lie along the premier line under consideration are tallied, (noting each nodes (x,y) coordinate positions). These nodes are then checked to ensure no duplications, and rank ordered from their minimum (x,y) values up. If there are more than two nodes, ie. one at each end, after this ordering process, the line is then split into its constituent elements. The original line is rewritten as the line connecting the first and second nodes in the list, and the additional line elements are given an identifier number and appended to the back of the line data list. This process effectively nullifies the sequence in which the lines were originally drawn.

The logic flow diagram for the section of the INTEREX subroutine dealing with intersections within the length of both lines is shown in Figure 6.19 overpage, and the sub-subroutine for splitting dissected lines in Figure 6.20 on page 6.32.

#### Tab Elimination

During the course of the above process, it is possible to encounter lines with only one, or no, lines connected to it. In the former case the line is identified as a 'tab', and in the latter as an independent line.

In both these cases a label is attached to the line in a companion file so that when the tracing routine is invoked, lines bearing either a tab or independent label are skipped, thus reducing the amount of calculation time involved. Should the tab have a length smaller than the user declared tolerance value, it may be eliminated completely, since it represents an overshoot where two lines cross.

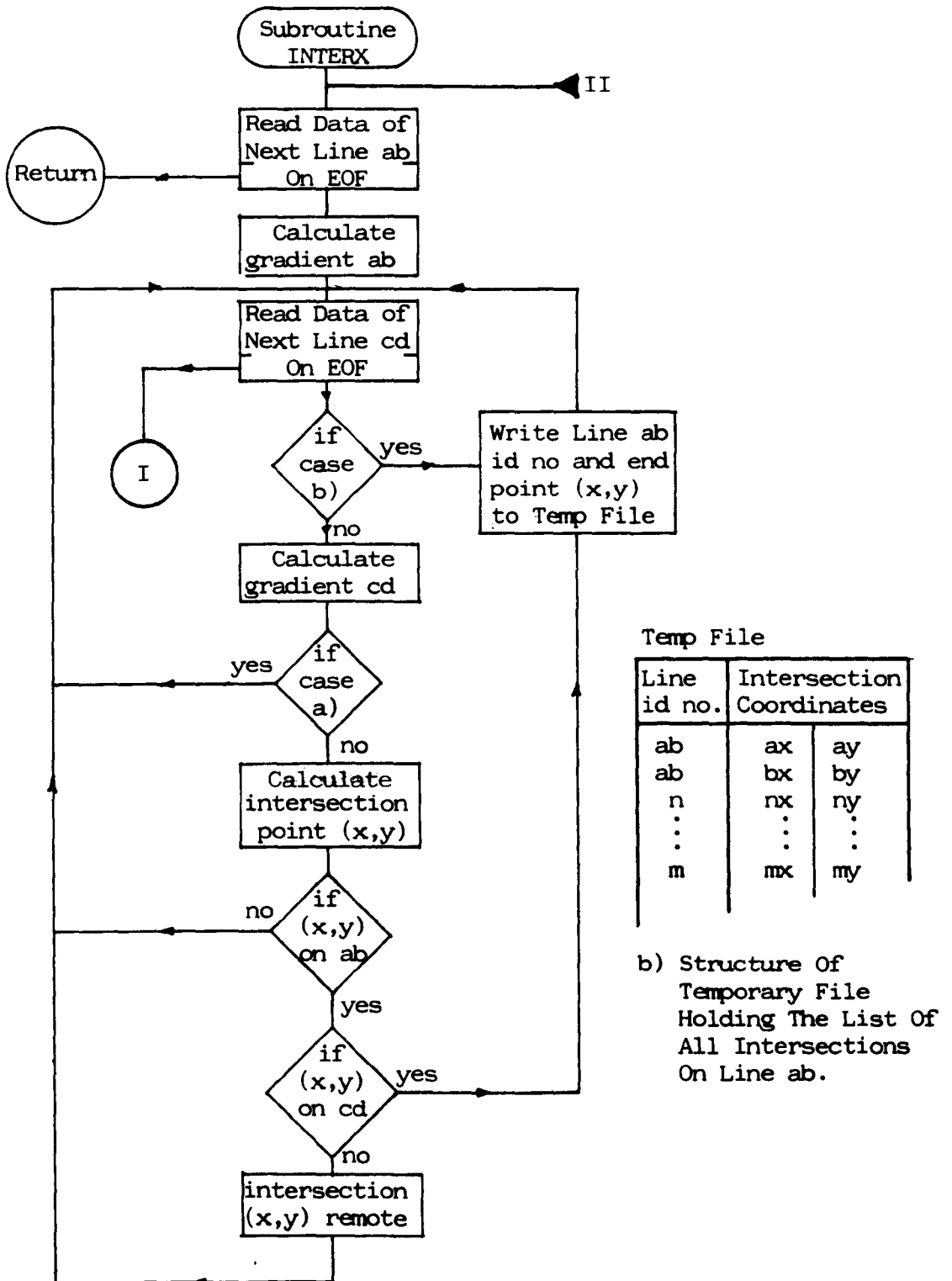


Figure 6.19 Flow Diagram Of Algorithm To Identify All Disected Lines And Direct Data To Splitting Routine.

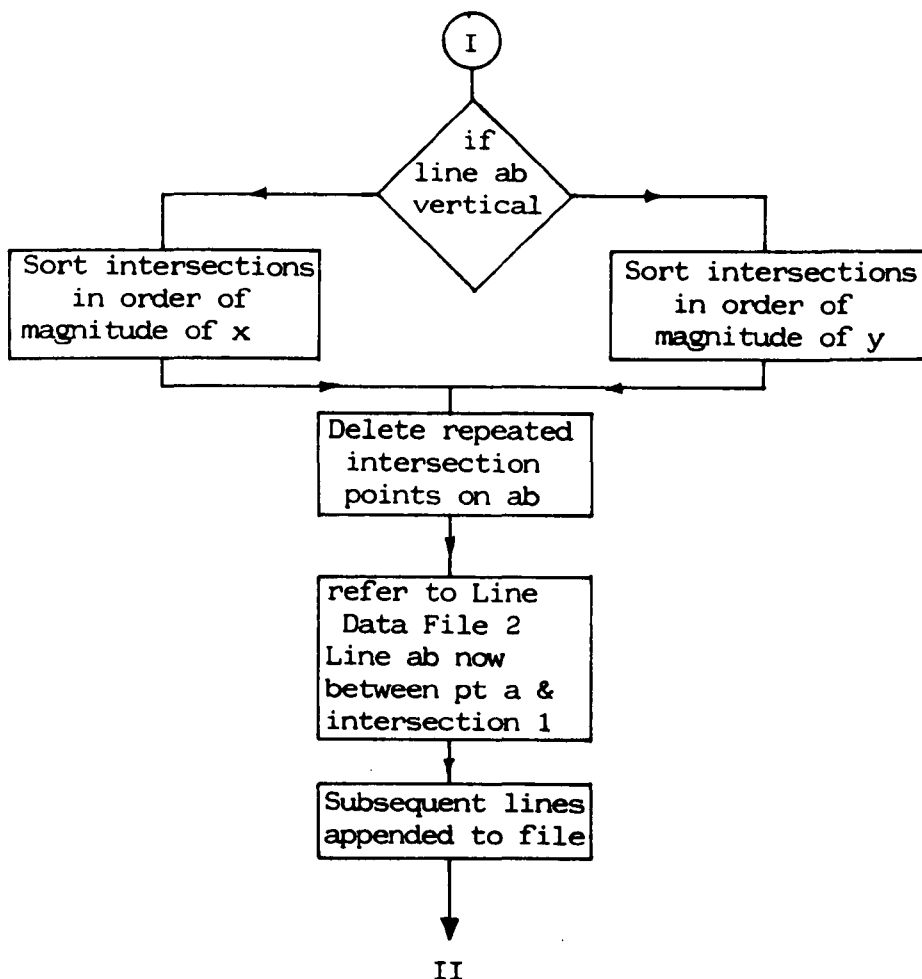


Figure 6.20 Flow Diagram Of Routine To Split Bisected Lines, In Conjunction With INTERX.

### Tracing Algorithm

#### a) Trace Navigation System

The trace follows the path around a shape by arriving at a node and determining the least (or greatest) angle between the edge and possible routes from it. On the first pass through the data, the trace takes the smallest angle, and on the second, the greatest, thus ensuring that all possible enclosed spaces are covered.

The process involves several major nested loops.

There are 68 possible geometric relationships between two lines, as shown in Figure 6.21 overpage when considered in terms

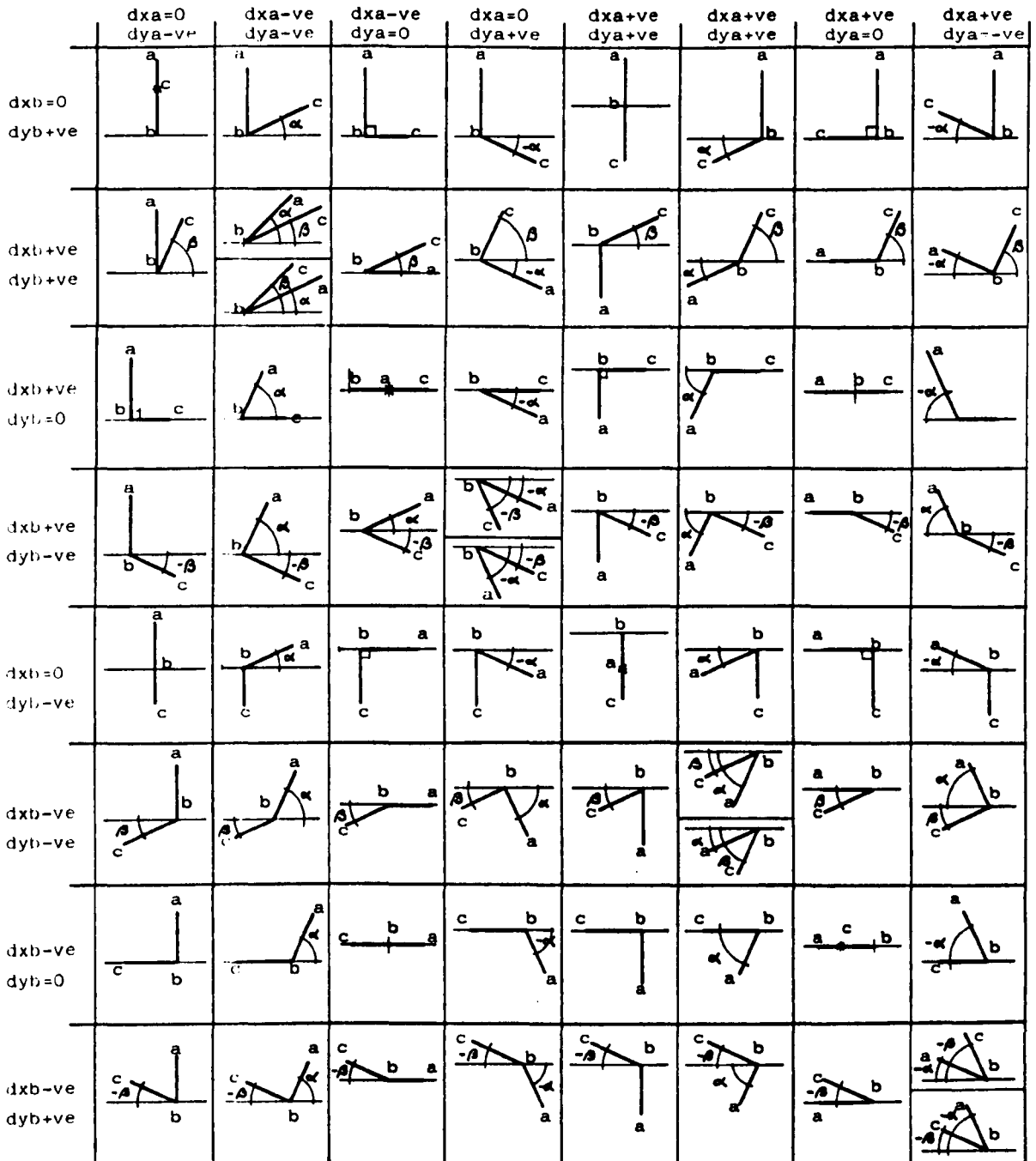


Figure 6.21 68 Possible Relationships Between Two Joined Lines In Terms Of Their Horizontal And Vertical Increments

of the relative heading between them, as described by their respective horizontal and vertical increments. This matrix discounts the other seventeen possibilities when either line has both of its increments of zero value, since this possibility has been filtered out of the data during the validation process. The matrix includes four ambiguous cases where the heading determination depends upon the relative magnitudes of the lines gradients.

Analysis of the 68 conditions leads to the derivation of a matrix showing the function of both lines gradients which gives the relative heading of the second line relative to the first. The angle is expressed in the sense of a compass heading, and the base line is facing south. This matrix is shown in Figure 6.22 below.

	dxa=0 dya=ve	dxa=ve dya=ve	dxa=ve dya=0	dxa=ve dya=ve	dxa=0 dya=ve	dxa=ve dya=ve	dxa=ve dya=0	dxa=ve dya=ve
dxb=0 dyb=ve	a+b	360+a-b	360+a-b	360+a-b	a+b	a+b	a+b	a+b
dxb=ve dyb=ve	180+a-b	a-b 360+a-b	360+a-b	360+a-b	180+a-b	180+a-b	180+a-b	180+a-b
dxb=ve dyb=0	180+a-b	a-b	a-b	360+a-b	180+a-b	180+a-b	180+a-b	180+a-b
dxb=ve dyb=ve	180+a-b	a-b	a-b	a-b 360+a-b	180+a-b	180+a-b	180+a-b	180+a-b
dxb=0 dyb=ve	-a-b	a-b	a-b	a-b	-a-b	180+a-b	180+a-b	180+a-b
dxb=ve dyb=ve	360+a-b	180+a-b	180+a-b	180+a-b	a-b	a-b 360+a-b	360+a-b	360+a-b
dxb=ve dyb=0	360+a-b	180+a-b	180+a-b	180+a-b	a-b	a-b	a-b	360+a-b
dxb=ve dyb=ve	360+a-b	180+a-b	180+a-b	180+a-b	a-b	a-b	a-b	a-b 360+a-b

Figure 6.22 Matrix Of Possible Relationships Between Angles a And b To Calculate The Heading Of Line bc With Respect To Line ab.

Fortunately, many of the heading descriptions are duplicated. Consequently a logic gate was written which, by simply comparing the arities of the axial intervals, determines the value of the heading by the sum or difference of their gradient angles relative to the compass, eg. if both the increments in the 'x' and 'y' directions are both positive, and those of the second line both negative, then the value of the heading is given by the angular gradient of the second subtracted from the first.

Once the relative headings of all the lines meeting at a junction have been assessed, the trace navigation system selects which route to take and moves up to the next junction having first checked to see whether it has completed a circuit.

To reduce the amount of calculation involved in this trace, use is made of the fact that one line element can occur as a boundary to only two shapes. Therefore, whenever a shape is traced around, a counter for each line set at an initial value of 2, is decremented by 1. Thus once it has been identified twice, the lines counter value is decremented to zero, and may not be used in subsequent trace attempts. A second feature is that at the termination of the trace algorithm, the remaining data list will contain a number of lines with a counter value of 1. These lines must be those forming the external boundary or silhouette of the picture. Note that tabs and independent lines were preset to a zero value, so that they too are ignored by the trace routine.

- b) Secondary Data Management
- c) Enclosure Check
- d) Duplication Detection

These three systems, although having discrete functions, are highly interrelated and consequently have to be described as one system. Figure 6.23 shows schematically the interaction between the four parts of the trace program overpage. The logic flow diagram for the whole tracing system is shown in Figure 6.24 a), b), and c) on pages 6.37-39.



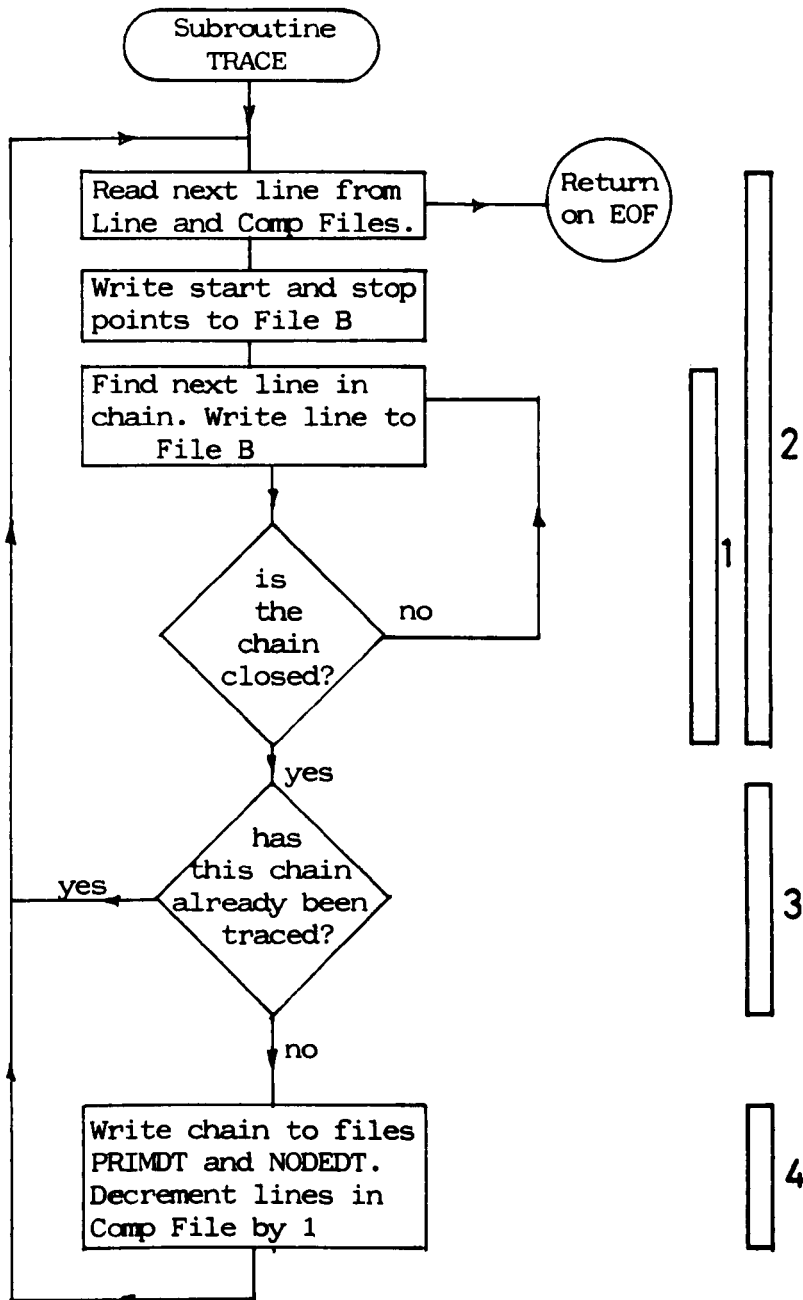


Figure 6.23 Schematic Flow Diagram Of Tracing Algorithm Showing Constituent Routines.

Key:

- 1 Secondary Data Manager
- 2 Trace Navigator
- 3 Enclosure Check
- 4 Duplication Detection And Output

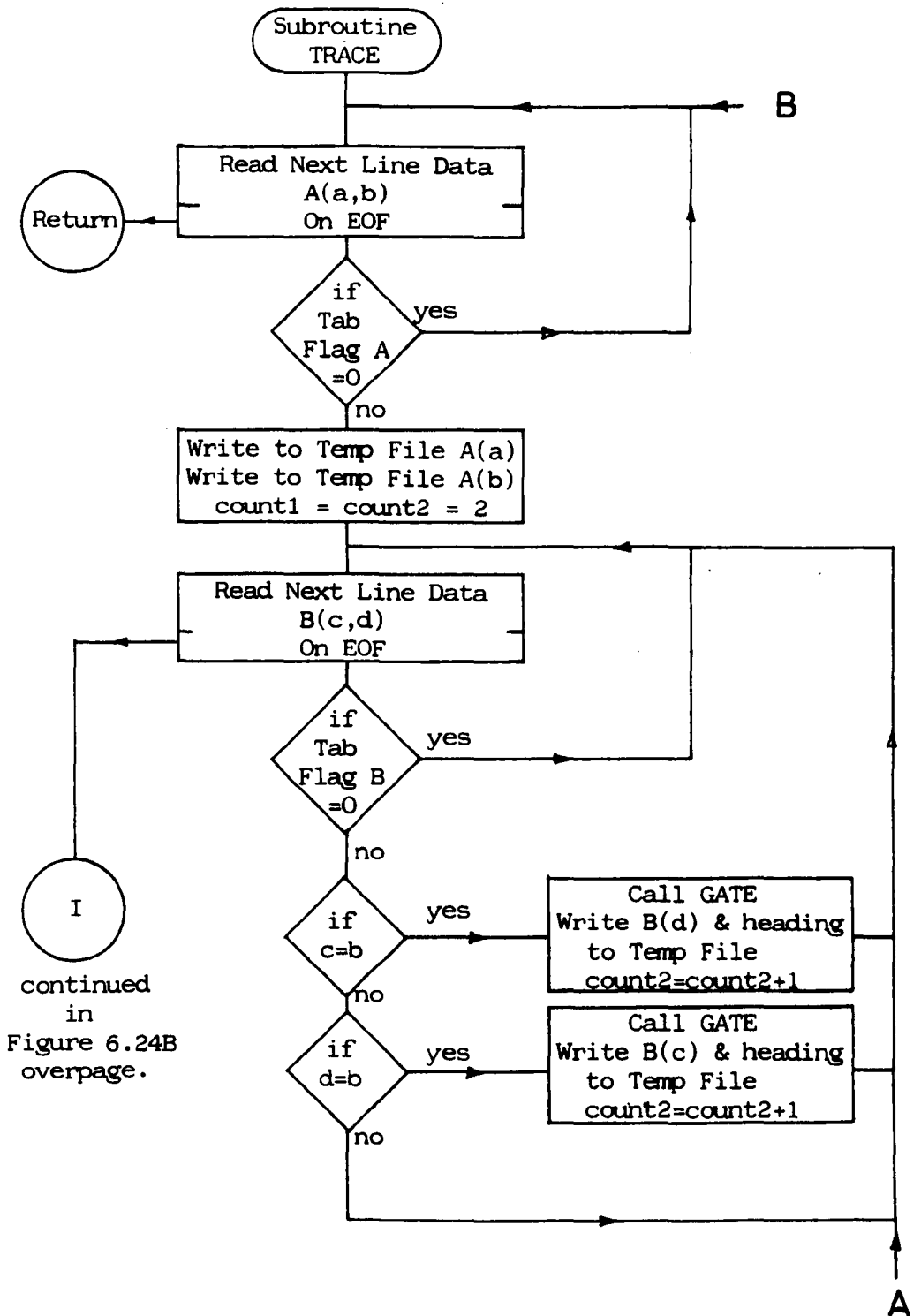


Figure 6.24A First Section Of Flow Diagram Of Tracing Algorithm.

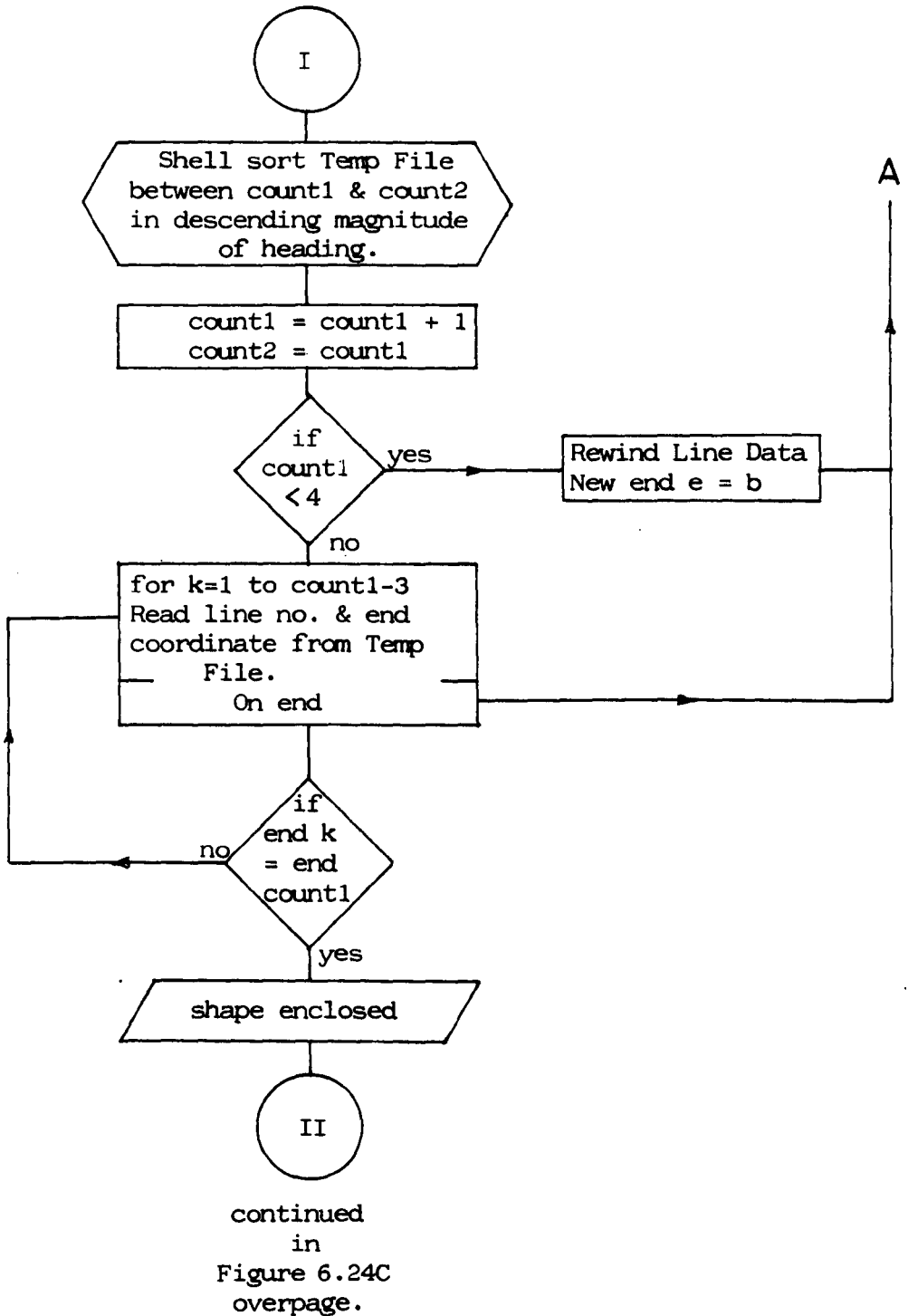
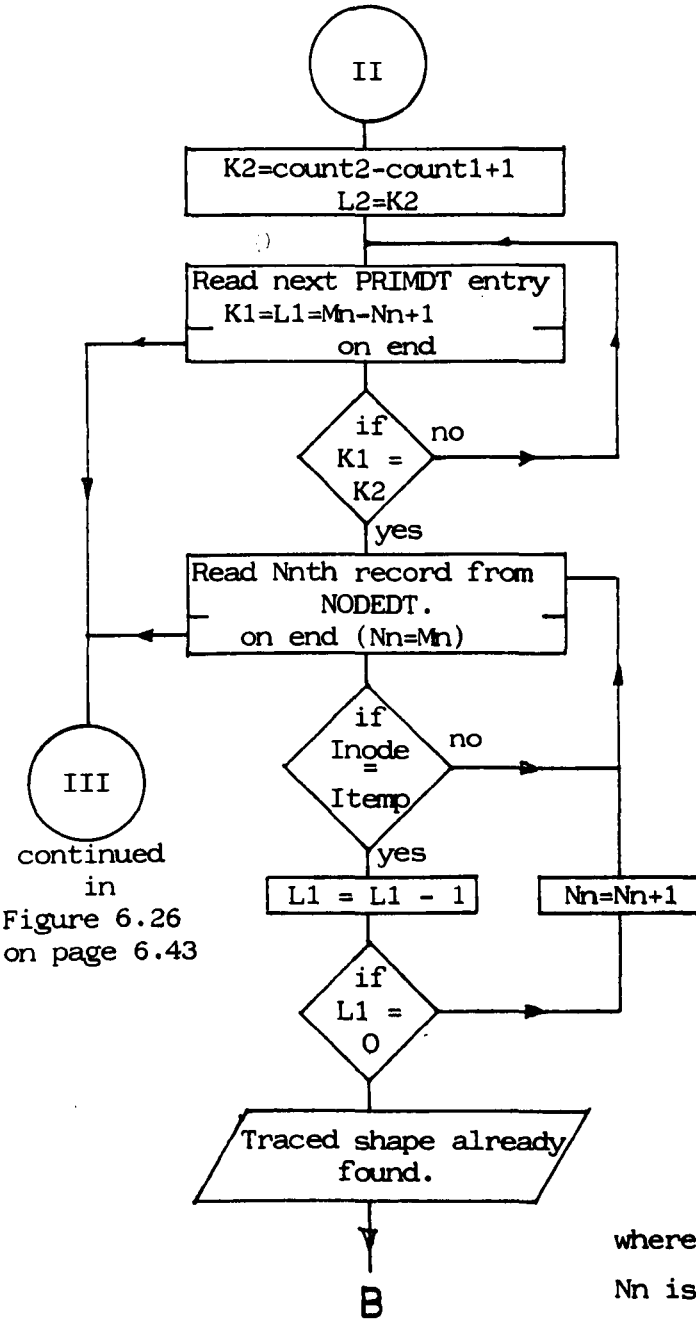


Figure 6.24B

Second Section Of Flow Diagram  
Of Tracing Algorithm.

From Figure 6.24B  
overpage



continued in  
Figure 6.26  
on page 6.43

where:  
 Nn is the start address of shape  
 already recorded in PRIMDT  
 Mn is the stop address of shape  
 in PRIMDT  
 Inode is the id. no. of the line  
 in mth record of NOEDT  
 Itemp is the id. no. of the  
 current line in Temp file

See Figure  
6.24A

Figure 6.24C Third Section Of Flow Diagram  
Of Tracing Algorithm.

When the trace algorithm starts with a line from the line data file, the end point is examined for connections by comparison of coordinates with all the other lines in the file. When a connection is discovered, the heading is calculated and two bits of information are written off to a temporary file, the connected line identification number, (its position in the file) and the heading. A counter (which we will call concount) is incremented by 1. Having processed all lines in the file, the temporary file is re-examined and the line with the least heading overwritten in the first available list position, and a second counter ( which we will call sidecount) incremented by 1. The first line selected will be placed in the first location. Concount is now made equal to sidecount. The end coordinates of the attached line are substituted for those of the starting line, and the algorithm reinvoked. Concount minus sidecount gives the number of possible routes from the end of the line under consideration, of which the one with the least heading is selected etcetera, etcetera.

Once the first three lines surrounding the shape under detection have been identified, it is necessary for the newly arrived-at end point to be compared with the others already in the temporary file list, to determine whether the shape has been enclosed; the minimal shape being a triangle.

If the closure check reveals that the current point held by the trace process is the same as the start point from which the trace began, the sequence of edge line identification numbers held in the temporary file are compared with any other previously determined shapes. It will be appreciated that any one shape might be traced starting with any of its sides. If it is found to be different to those already detected, it is stored. If not it is ignored, and the temporary file cleared in readiness for the commencement of the next trace. This comparison of sequences is handled by the duplication detection algorithm.

Before proceeding to describe the operation of the shape duplication procedure, it is necessary to describe the data storage structure in which the shape data is held. Figure 6.25

below illustrates the data structure employed.

LINEDT

Line id (position in sequence)	start point		stop point	
	X1	Y1	X2	Y2
1	X1.1	Y1.1	X2.1	Y2.1
2	X1.2	Y1.2	X2.2	Y2.2
3	X1.3	Y1.3	X2.3	Y2.3
"	"	"	"	"
n	X1.n	Y1.n	X2.n	Y2.n
n-1	X1.n-1	Y1.n-1	X2.n-1	Y2.n-1
n-2	X1.n-2	Y1.n-2	X2.n-2	Y2.n-2

PRIMDT

Primitive id (position in sequence)	NOEDT start address	NOEDT stop address	Reference point	
			X	Y
1	N1	M1	X1	Y1
2	N2	M2	X2	Y2
3	N3	M3	X3	Y3
"	"	"	"	"
n	Nn	Mn	Xn	Yn
n-1	Nn-1	Mn-1	Xn-1	Yn-1

NOEDT

Primitive id (position in sequence)	LINEDTids	End Point	
		X	Y
1	a1	xa1	ya1
	b1	xb1	yb1
	c1	xc1	yc1
	"	"	"
2	m1	xm1	ym1
	a2	xa2	ya2
	b2	xb2	yb2
	c2	xc2	yc2
"	"	"	"
"	"	"	"
"	"	"	"
n	an	xan	yan
"	bn	xbn	ybn
"	cn	xcn	ycn
"	"	"	"
"	mn	xmn	ymn
n-1	an-1	xan-1	yan-1
"	bn-1	xbn-1	ybn-1
"	cn-1	xcn-1	ycn-1
"	"	"	"
"	mn-1	xmn-1	ymn-1

Figure 6.25 The Relational Data Structure Employed For Storing The Geometry Data Base

The end point coordinates of the line elements following the line dissociation subroutine are held in a random access file, called LINEDT. Each line is identified by its record position in the file.

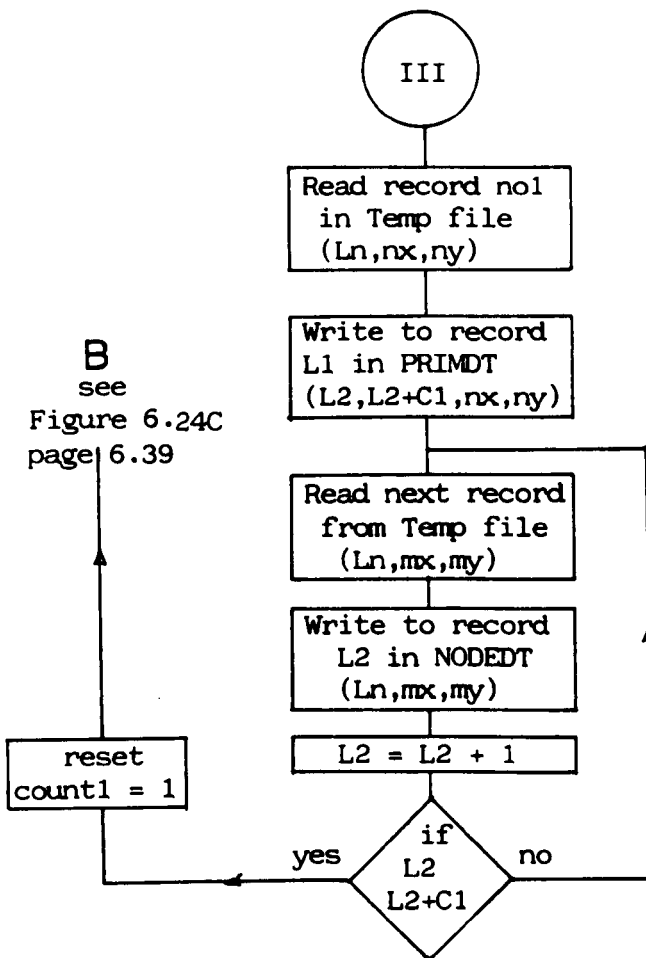
Shape data are accessed through a file, PRIMDT, which may be thought of as an index holding addresses of the shape data held in a third file, NODEDT. NODEDT holds a list of the coordinates which define the lines sequentially surrounding a particular space. The space is itself identified by the position of its index record in PRIMDT. Additionally, PRIMDT also holds a reference coordinate by which the shape is positioned on the drawing. It is incidentally also the same as the last point traced by the algorithm when the shape was being identified. This has the additional advantage that when a shape is to be drawn by a plotter, for example, the pen head can be instructed to move to the reference point, and then do a draw sequence between all of the points specified as enclosing the shape. The two addresses held in PRIMDT are those of the first and last coordinates in the sequence, held in NODEDT.

Significantly, the data in LINEDT is capable of sustaining modification by editing. Any alterations which effect the topological arrangement of the picture and relations between the spaces will corrupt the data in NODEDT.

The sequence of line identification numbers, (ie. their LINEDT record positions) for two polygons will be chained; that is that they end at their start point. However, two sequences of line numbers surrounding the same polygon will be out of phase since the search starts from a different point each time. The duplication detection algorithm therefore looks first for polygons having the same number of sides as the polygon just traced. If such a equal faced shape is encountered, two counters are set equal to the number of sides of both polygons. The line identification numbers in the newly traced sequence are checked against those of the existent shape, and if similar, the traced shape counter is

decremented by one, until one or both of the counters are equal to zero. All polygons have to be compared since it is possible for one shape to be enclosed by another with a dissimilar description, eg. a circle contained within a figure of eight.

Once it is established that the newly traced shape is not a duplication, it is written away to the files PRIMDT and NODEDT. The logic flow diagram below illustrates the subroutine to do this. It should be read in conjunction with Figure 6.24, pages 6.37-39



where:

- L1 is the next free record address in PRIMDT
- L2 is the next free record address in NODEDT
- C1 is the number of records in the temporary file.

Figure 6.26 Routine for Appending Newly Validated Shape Data To Data Structure.



In the course of the investigations into methods, alternative methods for shape duplication discovery were evolved. These methods are superior to the one described, but await improvements in the word lengths of available computers before they can be implemented. The first method is based upon the use of prime numbers, and can detect both duplicated and imbedded shapes; the second, using binary numbers can detect only duplications.

#### a Prime Number Method

Instead of assigning line numbers on the basis of a lines position in the LINEDT data store, each line is assigned a prime number sequentially, so that each is unique. Consequently, when a shape has been traced, the line identification numbers, primes, are multiplied together. This produces a unique product.

Any other shape producing the same product must be identical. Moreover, if the division of two products results in a whole number, the shape represented by the denominator must be embedded in the shape represented by the numerator.

The problem preventing the application of this method, is that the product obtained by the multiplication of the first fourteen primes, (starting with 3), produces a figure in excess of 64 digits, which is too large to be held in the computer without recourse to scientific notation; exponents of base 10. This defeats the object of using prime numbers in the first place.

#### b Binary Number Method

Here, the line identifiers are assigned in the decimal form of binary powers, ie. 1,2,4,8,16.... . Instead of multiplying the line identification numbers, the binary forms are added. This produces a unique binary pattern of a list of 1's and 0's which is unique to that particular shape. If the subtraction of the binary totals for any two shapes results in a zero answer, then the two shapes are identical.

The problem with this method is that again, only a limited number of line identifiers can be accomodated because of the

computers word length. Although with a double precision variable, this scheme would cope with drawings containing no more than 64 lines, (compared with the prime methods' 13), it is still too limiting for most CAAD applications. Moreover, this method cannot deal with embedded shapes.

#### Area Calculation

It has already been described how, at the end of the trace process, lines will be indicated in the companion file as bordering only one shape. These lines constitute the silhouette of the picture.

It is possible, however, that the external boundary might be picked up by the trace algorithm. In this case it is necessary to distinguish it from the other recognised shapes. This is done by calculating the areas of all the shapes by integration of the part areas subtended by each line of the boundary, one shape at a time. The total area of a shape may be positive or negative, depending on the direction in which the trace algorithm 'moved'. The integrals modulus is taken. The external envelope is then defined as the shape with the largest area, and to cross check, it should be equal to the sum of the areas of the other shapes.

Figure 6.27 overpage shows the logic flow diagram for the AREAS algorithm which does the calculation.

Figure 6.28 overpage demonstrates that irrespective of direction, whether the nodes are followed in a clockwise or anti-clockwise direction, the absolute integral gives the magnitude of the area.

The area data is essential to many CAAD applications.

## 5 Summary

This chapter began by defining the problems which detract from the uptake of CAAD systems by architects.

Examples of systems designed to overcome these problems were described and discussed. A system was proposed to extend the

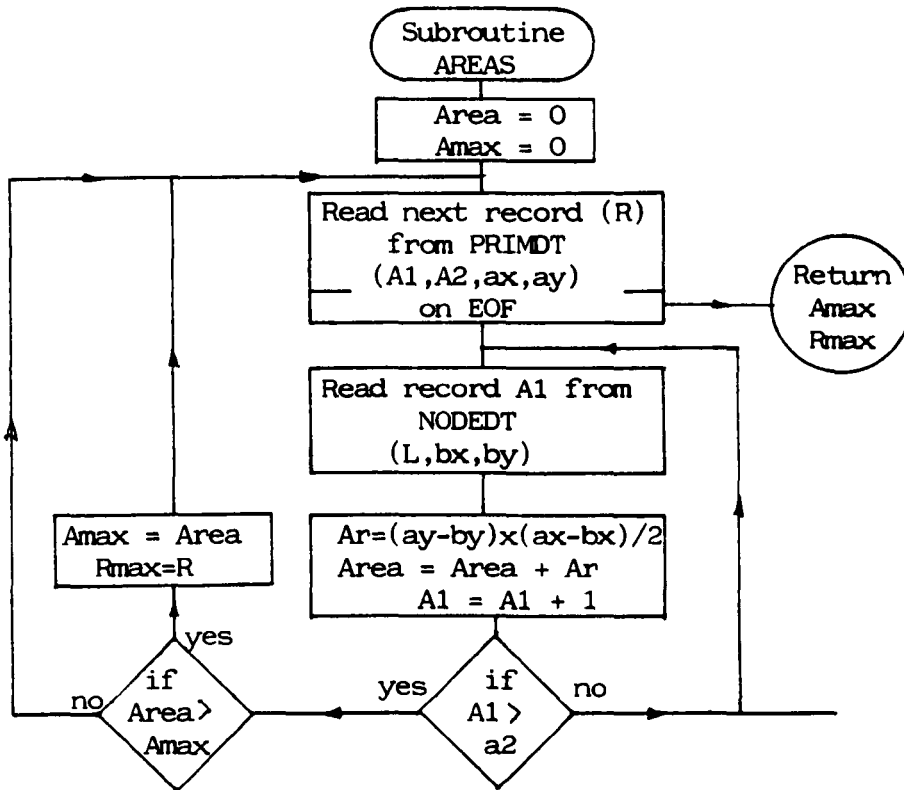


Figure 6.27 Flow Diagram Of Subroutine AREAS.

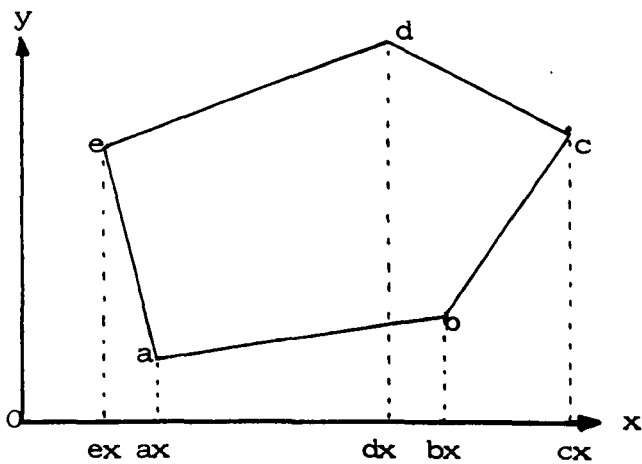


Figure 6.28 Example Of Area Calculation For An Enclosed Polygon.

With Reference To Figure 6.28:

Area Of Polygon a,b,c,d,e = Area (ex,e,d,c,cx) - Area (ex,e,a,b,c,cx)

A Starting at random point d in a clockwise direction

$$\text{Area}(d,c,cx,dx) = \frac{dy+cy}{2} + (dx-cx) \quad (\text{result } -ve)$$

$$\text{Area}(c,b,bx,cx) = \frac{cy+by}{2} + (cx-bx) \quad (\text{result } +ve)$$

$$\text{Area}(b,a,ax,bx) = \frac{by+ay}{2} + (bx-ax) \quad (\text{result } +ve)$$

$$\text{Area}(a,e,ex,ax) = \frac{ay+ey}{2} + (ax-ex) \quad (\text{result } +ve)$$

$$\text{Area}(e,d,dx,ex) = \frac{ey+dy}{2} + (ex-dx) \quad (\text{result } -ve)$$

$$\text{Area}(ex,e,d,c,cx) = \text{Areas}(ex,e,d,dx) + (dx,d,c,cx)$$

$$\text{Area}(ex,e,a,b,c,cx) = \text{Areas}(ex,e,a,ax) + (ax,a,b,bx) + (bx,b,c,cx)$$

Therefore;

$$\begin{aligned} \text{Total Area} &= \text{Area}(e,d,c,cx,ex) + \text{Area}(e,a,b,c,cx,ex) \\ &\quad (-ve) \qquad \qquad \qquad (+ve) \\ &= \text{Area}(abcde) \quad (-ve) \end{aligned}$$

B Starting at random point d in an anti-clockwise direction

$$\text{Area}(d,e,ex,dx) = \frac{dy+ey}{2} + (dx-ex) \quad (\text{result } +ve)$$

$$\text{Area}(e,a,ax,ex) = \frac{ey+ay}{2} + (ex-ax) \quad (\text{result } -ve)$$

$$\text{Area}(a,b,bx,ax) = \frac{ay+by}{2} + (ax-bx) \quad (\text{result } -ve)$$

$$\text{Area}(b,c,cx,bx) = \frac{by+cy}{2} + (bx-cx) \quad (\text{result } -ve)$$

$$\text{Area}(c,d,dx,cx) = \frac{cy+dy}{2} + (cx-dx) \quad (\text{result } +ve)$$

$$\text{Area}(ex,e,d,c,cx) = \text{Areas}(ex,e,d,dx) + (dx,d,c,cx)$$

$$\text{Area}(ex,e,a,b,c,cx) = \text{Areas}(ex,e,a,ax) + (ax,a,b,bx) + (bx,b,c,cx)$$

Therefore;

$$\begin{aligned} \text{Total Area} &= \text{Area}(e,d,c,cx,ex) + \text{Area}(e,a,b,c,cx,ex) \\ &\quad (+ve) \qquad \qquad \qquad (-ve) \\ &= \text{Area}(abcde) \quad (+ve) \end{aligned}$$

Proof Of Area Calculation Shown In Figure 6.28

principles upon which these programs were based to provide an easily used, general and transportable CAAD data structure for use in disparate CAAD applications.

The remainder of this chapter described in some detail how the proposed algorithms operate.

The following chapter will describe how these algorithms have been applied, and the ENIGMA suite of programmes are presented.



## Chapter 7

### ENIGMA

1	Introduction	7.1	167
2	Geometry Data File Management	7.1	167
3	Geometry Creation Devices	7.3	169
	3.1 Tablet In Continuous Digitisation Mode	7.6	172
	3.2 Tablet In Orthogonal Mode	7.6	172
	3.3 Tektronix 4010 Graphics Terminal	7.8	174
4	Output Options	7.13	179
	4.1 BIBLE Output	7.13	179
	4.2 GOAL Output	7.15	181
	4.3 Crude Appraisal	7.17	183
	4.4 Crude Visualisation	7.19	185
5	Resume	7.20	186

## CHAPTER 7

1 Introduction

The algorithms comprising the 'intelligent' routines in the ENIGMA suite of programs has been described in chapter 6. This chapter looks at the ENIGMA suite of programs as a whole.

The structure of ENIGMA is shown in Figure 7.1 overpage. The overall structure may be considered in three sections;

- a Geometry data file creation and manipulation,
- b selection of, and access to geometry creation devices, and,
- c selection of output modules.

2 Geometry Data File Management

An architect, when implementing the ENIGMA system, might wish to create a new geometry data file, or to amend an existing geometry description created in a previous invocation.

At the time of writing, the editing functions and routines for ENIGMA are being developed independently as part of a Ph.D thesis by a colleague\*in the Scott Sutherland School of Architecture. These routines exist independently of ENIGMA, and are discussed in Chapter 8. Consequently, editing functions in the current ENIGMA implementation are restricted to either adding additional data to a drawing, or scrubbing the picture entirely in order to restart.

A user might have an existing ENIGMA, GOAL or BIBLE File which, created during a previous ENIGMA session, he wishes to amend. In the first case, the data will be held in a line data file, (LINEDT). On the new run, the LINEDT file is copied to the ENIGMA system file, and left open in append mode. This procedure removes the end-of-file (EOF) marker from the end of the file, and leaves the pointer at the next free record.

Work is currently being done to establish both BIBLE and GOAL file interpretation procedures. Here, each shape specified is described in terms of the individual lines surrounding the

\* SERC Funded Research Project No.82301583 (see acknowledgements).



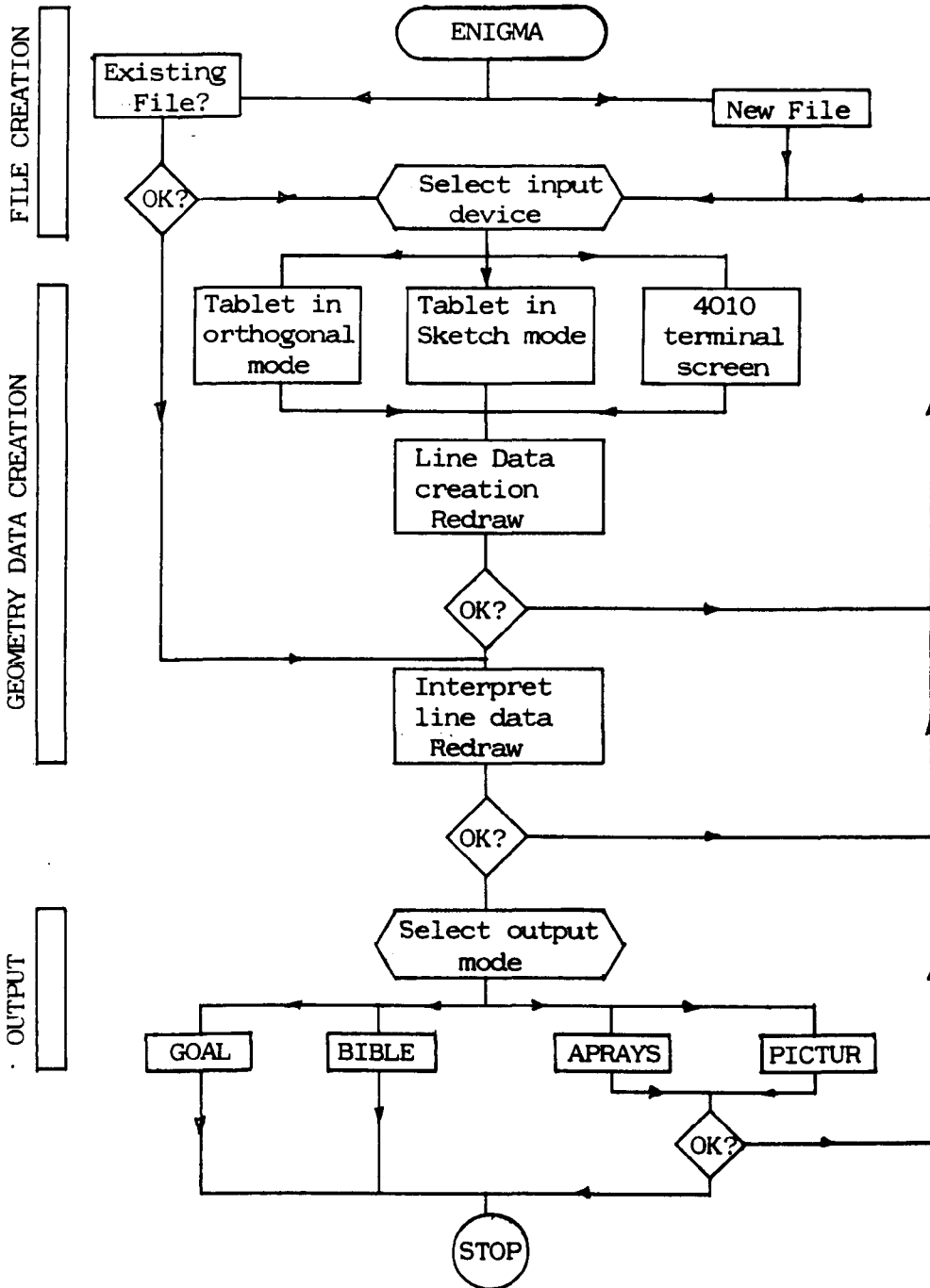


Figure 7.1 Schematic Representation Of ENIGMA Flow Path.

plan forms of each space. These lines are written to a transitory line data file, from which duplicated lines are eliminated. Methods are being considered to write data directly to the shape data files, thereby eliminating the need for the implementation of the ENIGMA interpretation routines. Because these routines are not yet fully operational, they are shown as dotted in the logic flow diagram of the file management system shown in Figure 7.2 overpage.

Essentially, existing files are copied, so that the original is not lost should the user wish to revert to a former version, and then with the additional or amended information, the line data file is subjected to the ENIGMA interpretation routines. The consequent application of editing routines, (when they become fully integrated), particularly the deletion of lines and shapes, will necessitate the removal and compaction of null records from the data structure and the consequent renumbering of the remaining records in the final output files. These sequences are discussed in Chapter 8.

### 3 Geometry Creation Devices

Three geometry input devices or modes are available to the ENIGMA user. The digitiser may be used in point or continuous mode, or the drawing may be effected through the use of the Tektronix 4010 graphics terminal using crosshair cursor and thumbwheels. The major reason for including the graphics terminal as an input medium is the anticipation of the editing capabilities. Clearly, a sketch done in ink on paper cannot be easily changed. Once the enhanced sketch has been echoed upon the terminal screen, using the manipulation functions through the terminal, an updated picture may be created. However, the new picture bears little relation to that still shown on the tablet, and the tablet ceases to be the input medium.

Figure 7.3 on page 7.5 shows the flow chart for the selection of geometry input devices. It will be noted that all routes to any particular device entails the definition of scale and default

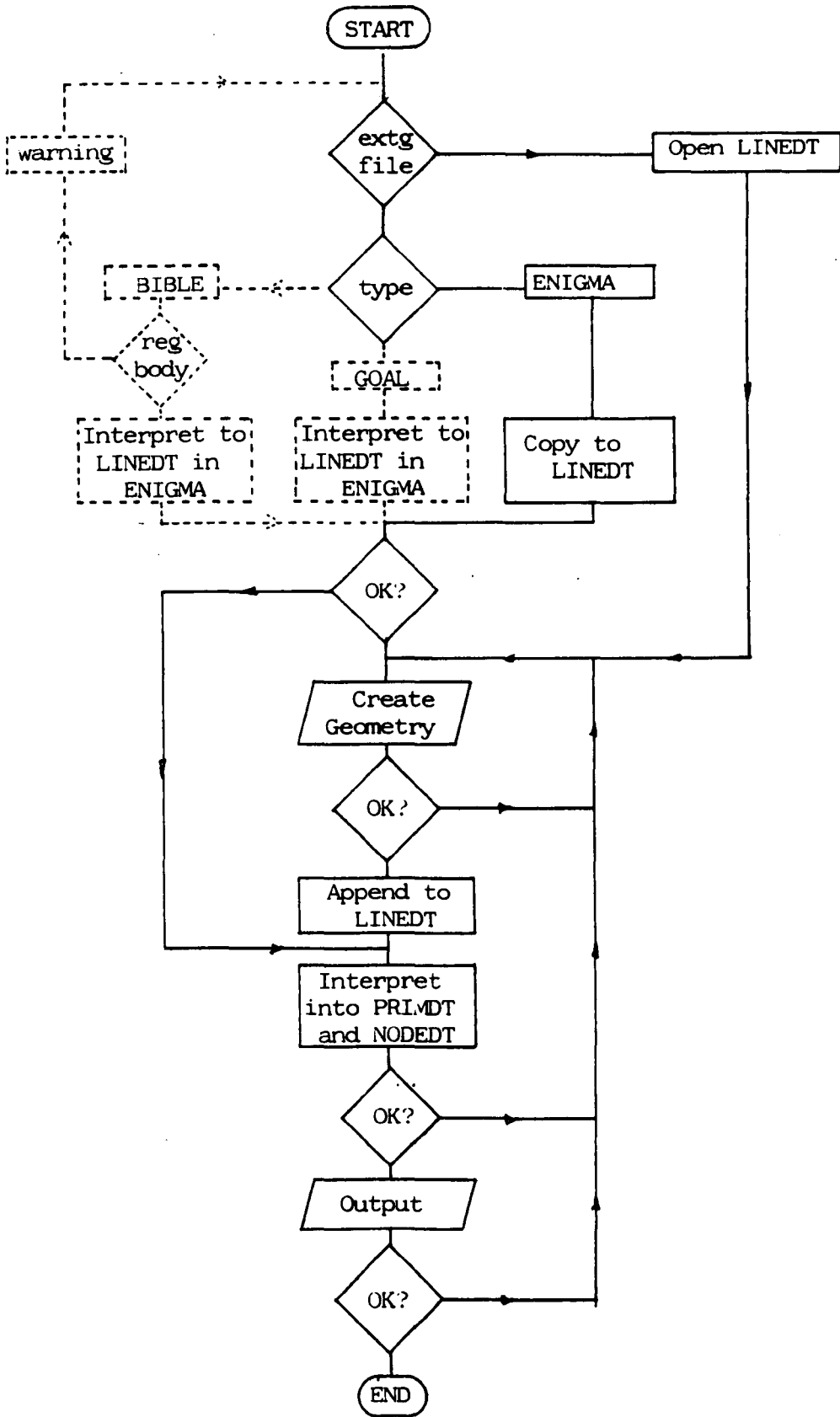


Figure 7.2 File Management Flow Diagram Omitting All Internal System Files.

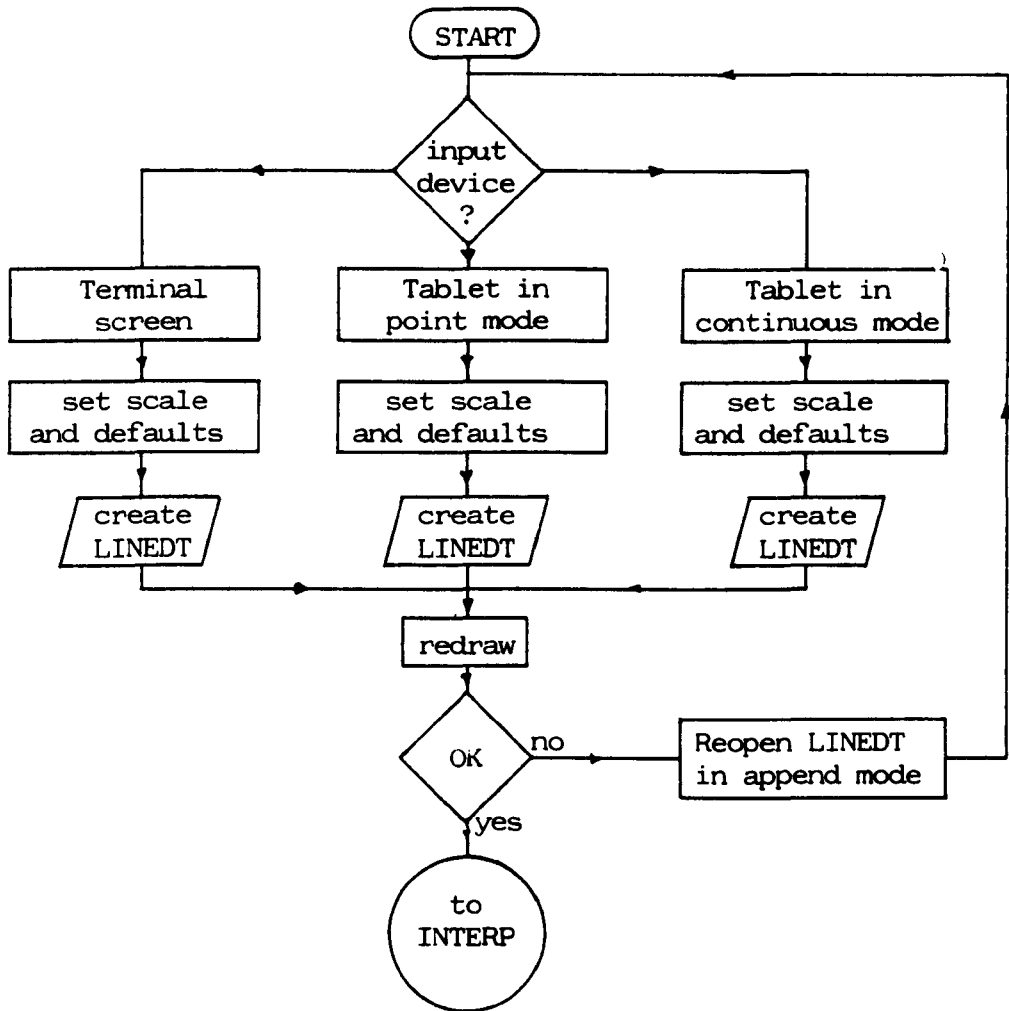


Figure 7.3 Flow Chart For Geometry Input Device Selection.

values which will be required first in the conversion of tablet or screen units to user units of measurement, and secondly during the interpretation process. The two main values to be settled are the scaling factor appropriate to each particular device, and secondly the tolerance value specifying how close two points are to be before the system can assume that they are joined together. Other options include the specification of the angle of deviation between a pair of line segments recorded on the tablet in continuous mode within which the latter is to be considered a continuation of the former, the number of short arcs to be exceeded before an intentional arc is recognised, and the number of points thereon. Whilst these latter values are necessary for the freehand sketch mode, for brevity the same routine is used on all device preset routines.

### 3.1 TABLET IN CONTINUOUS DIGITISATION MODE

The algorithm which drives the tablet in continuous point digitisation mode has been described in Chapter 6, section 4.1.

The output produced by this routine is in the form of a line data file as depicted in LINEDT shown in Figure 6.25 on page 6.41, for subsequent processing.

### 3.2 TABLET IN ORTHOGONAL MODE

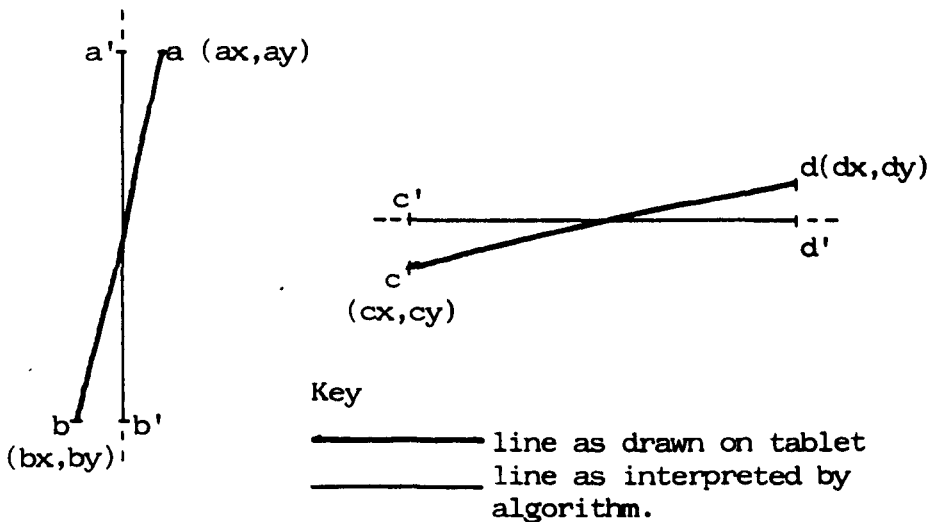
Some application software programs, (most notably GOAL), restrict the permissible geometry descriptions to orthogonal forms. In this case much of the calculations carried out by the interpretation calculations becomes superfluous, since obviously the line gradients encountered will either be horizontal or vertical. Consequently, the data input procedure need only concern itself with these two classes of line.

Usually, such rectilinear geometries are created on a computer system by point digitisation. However, in the ENIGMA system it was considered important to retain the freehand sketch capability of the data generation system.

From the continuous digitisation of a freehand sketch in the case of rectilinear geometries, we are interested

only in the start and stop points of each line, since these will define along which axis the line lies. In freehand sketches the lines of the drawing are almost certain to have small gradients due to the drawing error, unless the digitising tablet is fitted with some form of parallel motion device.

In lieu, or anticipation of the availability of such a draughting device, it is assumed that the user will input each straight line individually. In this case, where the stylus is still being monitored in continuous digitisation mode, we are interested only in the start and stop points; ie. those characterised by a header value of either 29 or 31. Between these two points, within a small margin of error, either a horizontal or vertical line exists. The relative magnitude of the 'x' interval to the 'y' will define to which axis any line is collinear. The line is then reproduced as that vertical or horizontal line passing through the midpoint of that drawn. Although this will introduce a small error at the node points of the lines, as illustrated in Figure 7.4 below,



Interpretation:

ab is the line

$$\left(\frac{ax+bx}{2}, ay\right), \left(\frac{ax+bx}{2}, by\right)$$

cd is the line

$$\left(cx, \frac{cy+dy}{2}\right), \left(dx, \frac{cy+dy}{2}\right)$$

Figure 7.4 Resolution Of Freehand Lines Into Orthogonal equivalents.

where, for example, the point a is moved laterally to point a', in practice these displacements will be small, and will in fact be eliminated by the tolerance interaction on interpretation.

If the gradient of a line is calculated to be greater than 1, it is assumed vertical; if less than 1, horizontal; and if equal to one, an error message is displayed to the user.

The adjusted end point coordinates are then written direct to the line data file for subsequent dissociation prior to interpretation.

The logic flow diagram of the orthogonalised sketch data input subroutine is shown in Figure 7.5 overpage.

### 3.3 TEKTRONIX 4010 GRAPHICS TERMINAL

With this mode of graphic data input, the user can create geometries using predefined shape primitives, ie. rectangles, triangles and lines. Since rectangles are always coaxial, only two coordinate points need be input; those defining a diagonal. Lines occurring individually are rectified, if necessary during the input procedure by the process outlined above if an orthogonal picture is required. In this case, of course, the use of the triangle drawing routine would be prohibited. The logic flow diagram of the terminal driving subroutine is shown in Figure 7.6 on page 7.10.

Once created, there is a potential problem with the geometry data which has to be resolved. When a picture is produced by the aggregation of shape primitives, there is the probability that several shapes may share a common edge as depicted in Figure 7.7 on page 7.11. Such occurrences would result in the common line data being recorded several times, which would cause confusion and failure in the following interpretation process.

Before any two lines can interact in the manner illustrated, they must both have the same general equation; ie. gradient and constant of intersection. Thus during the digitisation process, each line has its values calculated and stored in a companion file. If the line is vertical, its 'x' axis intercept is recorded, otherwise its 'y'. Each newly described line is compared with those already existing in the line data file. If the two lines

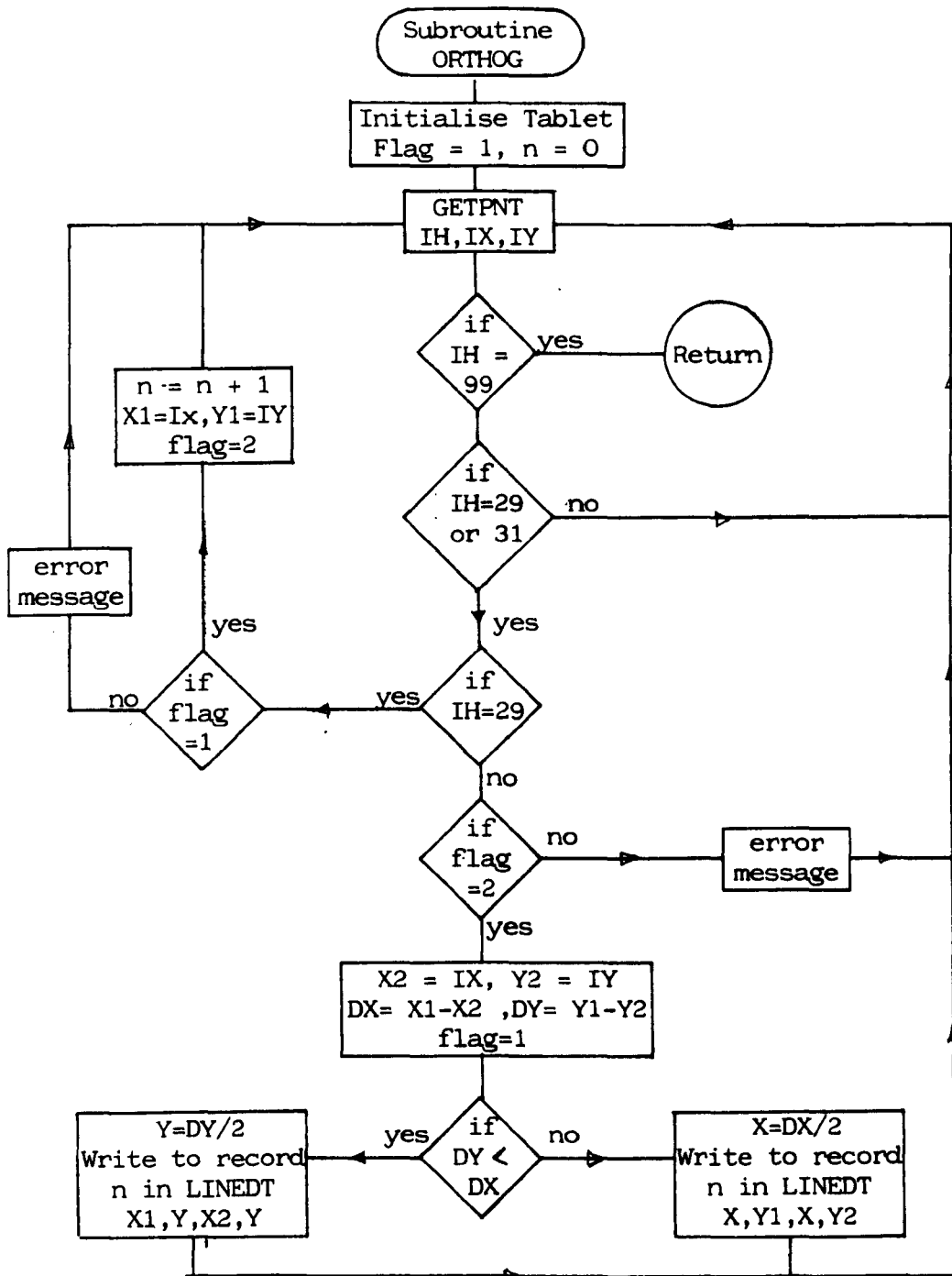


Figure 7.5 Flow Diagram Of The Subroutine To Drive The Tablet In Orthogonal Digitisation Mode.



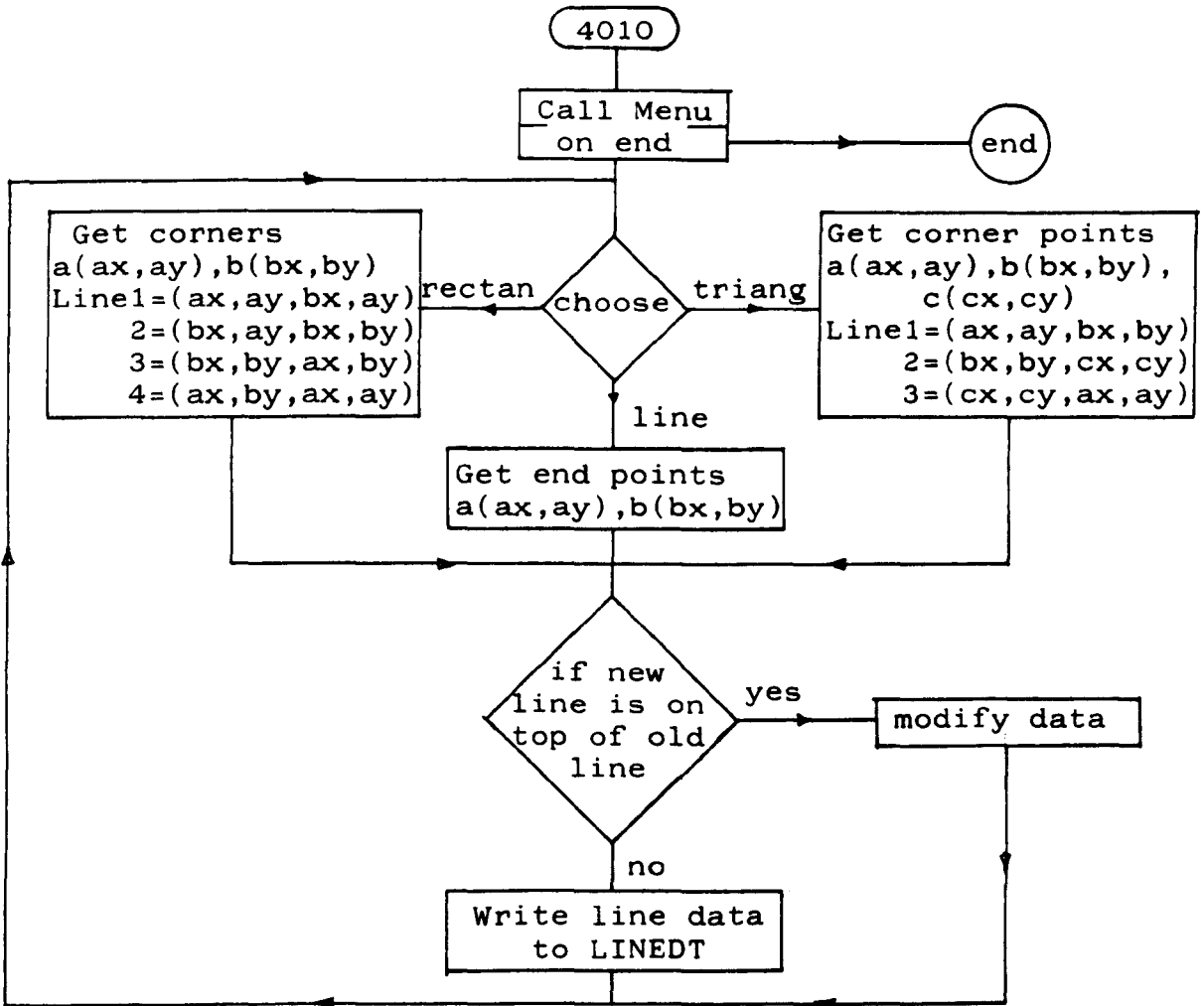


Figure 7.6 Flow Diagram Of The Terminal Operating Routine.

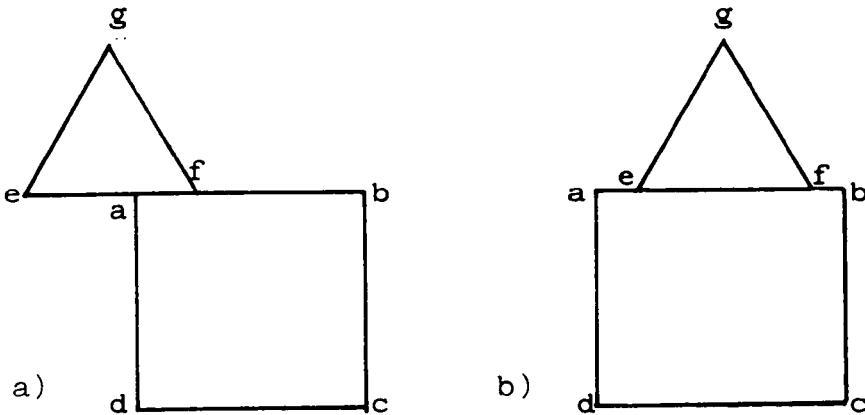


Figure 7.7 Two Cases Of Superimposed Lines

are collinear, then the line's end points are fed through a 'logic gate' to determine if they interfere with one another, and if so, modify the data accordingly. In the above figure for example, the two lines ab and ef interfere. In the first case, assuming that abcd is already described in the LINEDT file, the line ab needs to be changed to the line eb, and line ef discarded. In the latter case, line ab remains unchanged, but still line ef needs to be discarded.

This logic gate operates by first testing whether the two lines under consideration are described in the same direction, ie. whether ex-fx, and ax-bx are both negative or positive. Vertical lines are considered similarly in terms of their 'y' coordinates. Following this, the end points of the most recent line (line ef) is tested to see whether they lie between the end points of the stored line. If both ends of ef lie between the end points of ab, then the most recent line is discarded. If the lines meet at an end, or they slightly overlap, the data held in store is modified, and the recent line discarded. The logical decisions are listed in Figure 7.8 overpage.

For two non-vertical lines, AB  $((ax, ay), (bx, by))$  and CD  $((cx, cy), (dx, dy))$ , which are colinear and have the same constant of intersection on the 'y' axis, then;

If  $(ax-bx)$  -ve and  $(cx-dx)$  -ve, then  $p=ax, q=bx, r=cx, s=dx$

If  $(ax-bx)$  -ve and  $(cx-dx)$  +ve, then  $p=ax, q=bx, r=dx, s=cx$

If  $(ax-bx)$  +ve and  $(cx-dx)$  +ve, then  $p=bx, q=ax, r=dx, s=cx$

Otherwise  $p=bx, q=ax, r=cx, s=dx$

Moreover,

If  $(r \geq p)$  and  $(s \leq q)$  then line CD lies within AB, or

If  $(p \geq r)$  and  $(q \leq s)$  then line AB lies within CD, or

If  $(p \leq r \leq q)$  then line pq = line given by  $(p,s)$ , or

If  $(p \leq s \leq q)$  then line pq = line given by  $(r,q)$ ,

Otherwise the lines do not interact.

Figure 7.8 Logic Gate For Testing The Interaction Between Two Co-linear Lines

Although the above logic gate can 'recognise' and rectify two overlapping collinear lines, in certain circumstances, more than two interacting collinear lines would cause a failure. Such an example is shown in figure 7.9 below, where a third shape 'bridges' two already existing.

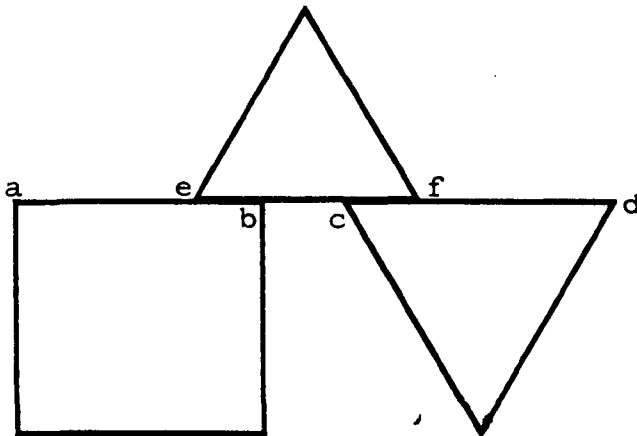


Figure 7.9 Shape Configuration Producing An Error On Overlap Decoding

These errors could be removed by subjecting the whole content of the line data file to comparison on completion of the drawing sequence by gathering all colinear lines together, rank ordering their least 'x' or 'y' coordinates of their end points, and then sequentially dealing with each pair of lines. This clearly involves much calculation.

#### 4 Output Options

There are four modes of output which the user can select from; the creation of a GOAL formatted output data file for direct input into the geometry description of the GOAL suite of appraisal programmes; the creation of a BIBLE formatted output data file for direct implementation into the BIBLE visualisation program; a rudimentary isometric visualisation program; and a rudimentary building performance appraisal routine based upon the external envelope of the building plan.

All output options require the line data to undergo the interpretation programme, and for the user to input floor and ceiling heights for the storey or space.

At the termination of either the GOAL or BIBLE output options, the ENIGMA program terminates: This is based upon the assumption that the user will want to implement his created geometry files in these programmes immediately. At the termination of the other options, control is returned to the output selection should the user wish to specify further output choices, or indeed, to amend his geometry data.

##### 4.1 BIBLE OUTPUT

The structure of a BIBLE geometry data file has been described in Chapter 5, section 5. In the ENIGMA programme, all objects for a BIBLE description are treated as 'regular' bodies. By the definition in the BIBLE user manual,

"a general body is a vertical prism, ie. a body with exactly two horizontal surfaces, a floor and a roof, which are parallel and equal polygons, and the remaining faces of

being vertical and joining corresponding edges of the floor and roof. Data for a regular body consists of:-

- a line whose first three characters are REG
- a line containing the (integer) number (L) of sides of the floor (and roof), and the (real) Z-coordinate of the floor, and the (real) Z-coordinate of the roof.
- a list of pairs of the (real) X- and Y-coordinates of the (L) vertices of the floor (and roof) in clockwise order as viewed from above."

In ENIGMA, a space is treated as an extrusion of its plan form through a base to ceiling height. If the user fails to specify these levels, the floor is taken to be at a level of 0.000 meters, and the ceiling at 3.000 meters by default.

Consequently, each space detected by the interpretation subroutines are checked to see whether they are recorded in a clockwise or anticlockwise direction, and the number of sides for each space assessed. Having written to file a line containing the 'REG' tag, number of sides (given by the difference between the stop and start addresses held in the PRIMDT file), and the upper and lower surface levels, it then remains to write the sequence of node coordinate pairs from the start to stop address in NODEDT if the shape has been traced clockwise, or from stop to start address if anti-clockwise.

Each shape so translated is concatenated into a list of data, and the list given a file name, BIBOUT.DAT by default.

The data format for each encoded shape is shown in Figure 7.10 below.

```
REG
L,Zf,Zc
ax,ay,bx,by,cx,cy.....Lx,Ly
```

Figure 7.10 Format Of Interpretation Of ENIGMA Data To BIBLE Data By Subroutine BIBOUT

The logic flow diagram of the subroutine which translates the geometry data from its ENIGMA form to BIBLE form, is shown in Figure 7.11 overpage.

#### 4.2 GOAL OUTPUT

It will be seen from Chapter 5, section 5, that GOAL geometry data files are those compiled by the GRAMP graphic handling system, which constrains the user to pictures consisting of the aggregation of simple shape primitives. Several shapes may be aggregated to describe one zone of up to 65 separate shapes, and some two dozen different zone types may be specified. The shapes may be entered with an angle of rotation relative to the 'east' of the drawing device.

In the current implementation, ENIGMA assumes that shapes are drawn coaxially, ie. the shapes are oriented north-south, and are exclusively rectangular.

GOAL allows the user to describe multistorey building geometries by assigning a floor index to each element. The current implementation of ENIGMA does not allow the direct entry of more than one storey at a time, and to encode multistorey building forms, the user is required to describe each storey in turn, and then to append each resulting data file into a master by use of the TOPS20 file editing system outwith the ENIGMA programme.

After entering the storey index which will relate to the entire picture currently held in the ENIGMA picture, each shape in turn is read from the PRIMDT file, and drawn in outline on the screen by reference to the NODEDT file. If there are more than 4 edges to a shape, an error message is displayed to the user, and that particular shape is ignored. Otherwise, the corner of the rectangle nearest to the origin is abstracted.

The shape's area, held in the companion file is divided by the shapes width. This is done in anticipation of routines to be written to enable non-rectilinear pictures to be 'orthogonalised' at some point in the future. For each shape drawn upon the terminal screen, the user is required to enter; first the zone's function

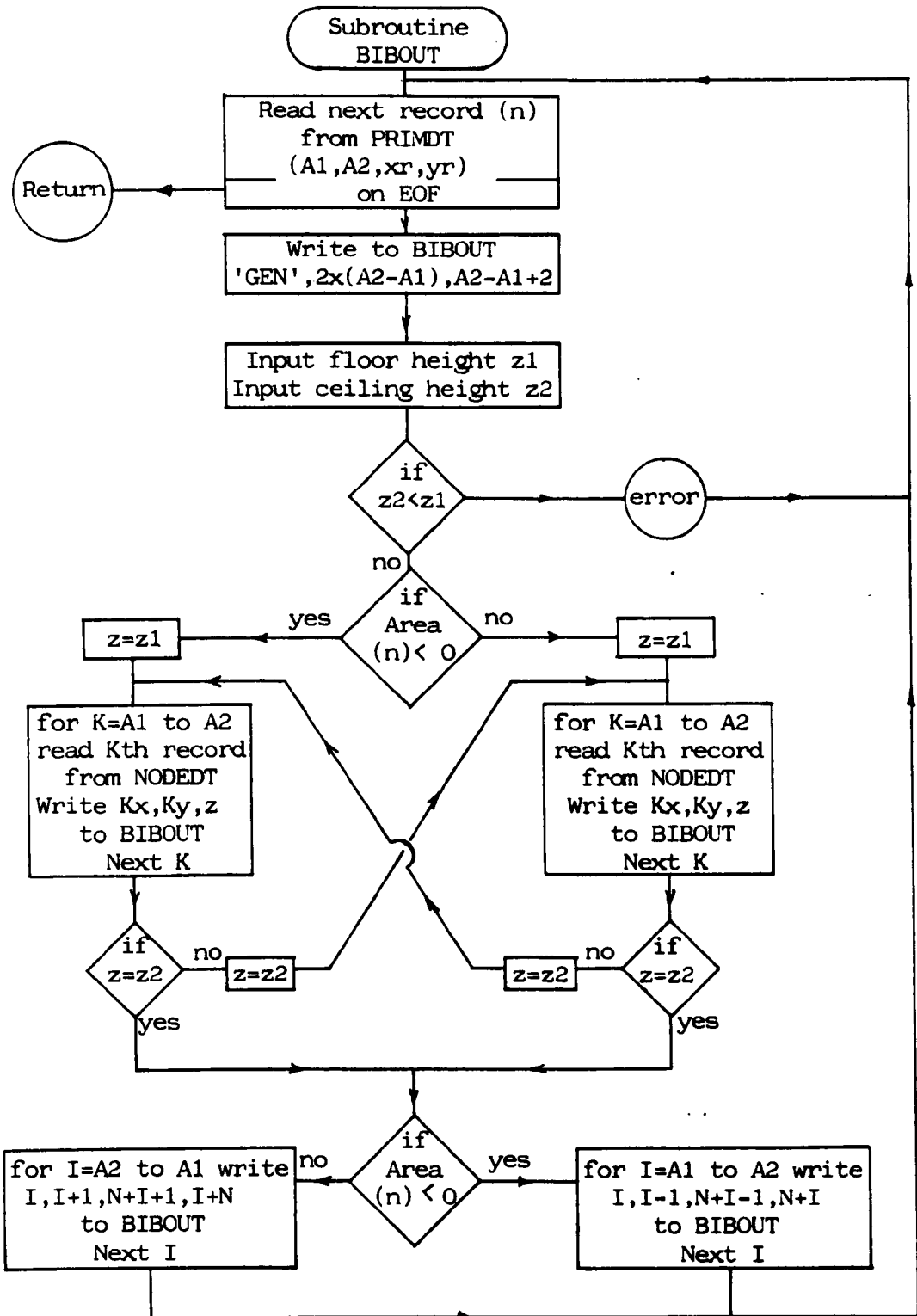


Figure 7.11 Flow Diagram For BIBLE File Creating Routine

type; second, the shape reference number within the specified zone, third, the floor level, and finally, the ceiling/roof level. In assigning zone types, the user is familiar with, or has the GOAL operating manual to hand.

The data so accrued is then formatted into GOAL compatible form, and written to a file, named GOALDT.DAT on default.

The logic flow diagram for the subroutine which translates from the ENIGMA data structure to that of GOAL, is illustrated in Figure 7.12 overpage.

#### 4.3 CRUDE APPRAISAL

Utilising the data contained in the ENIGMA database, crude appraisals may be undertaken. These appraisals are not intended to be accurate performance predictions, but enable relative comparisons to be made between alternative solutions, giving an indication of which of the alternatives is likely to perform better, not to define how much better.

The appraisals undertaken test:

- a the compactness ratio of the building form,
- b the relative elemental costs, and
- c the maximum heat demand to maintain an internal temperature at a defined minimum temperature of 0°C.

The Compactness Ratio is defined as the ratio between the external surface area of the proposed building, (excluding the area of the floor), to the surface area of a hemisphere which contains an equal volume.

Assuming a uniform storey height, the external exposed surface area of the proposed building may be calculated, and the equivalent surface area of an equal volume enclosing hemisphere deduced.

The elemental costs are based upon cost indices, which will have to be estimated by reference to current cost data. The programme measures the areas of roof, floor, opaque wall and window, (based upon the maximum permitted glazing ratios permitted by the statutory regulations), and a foundation assumed to



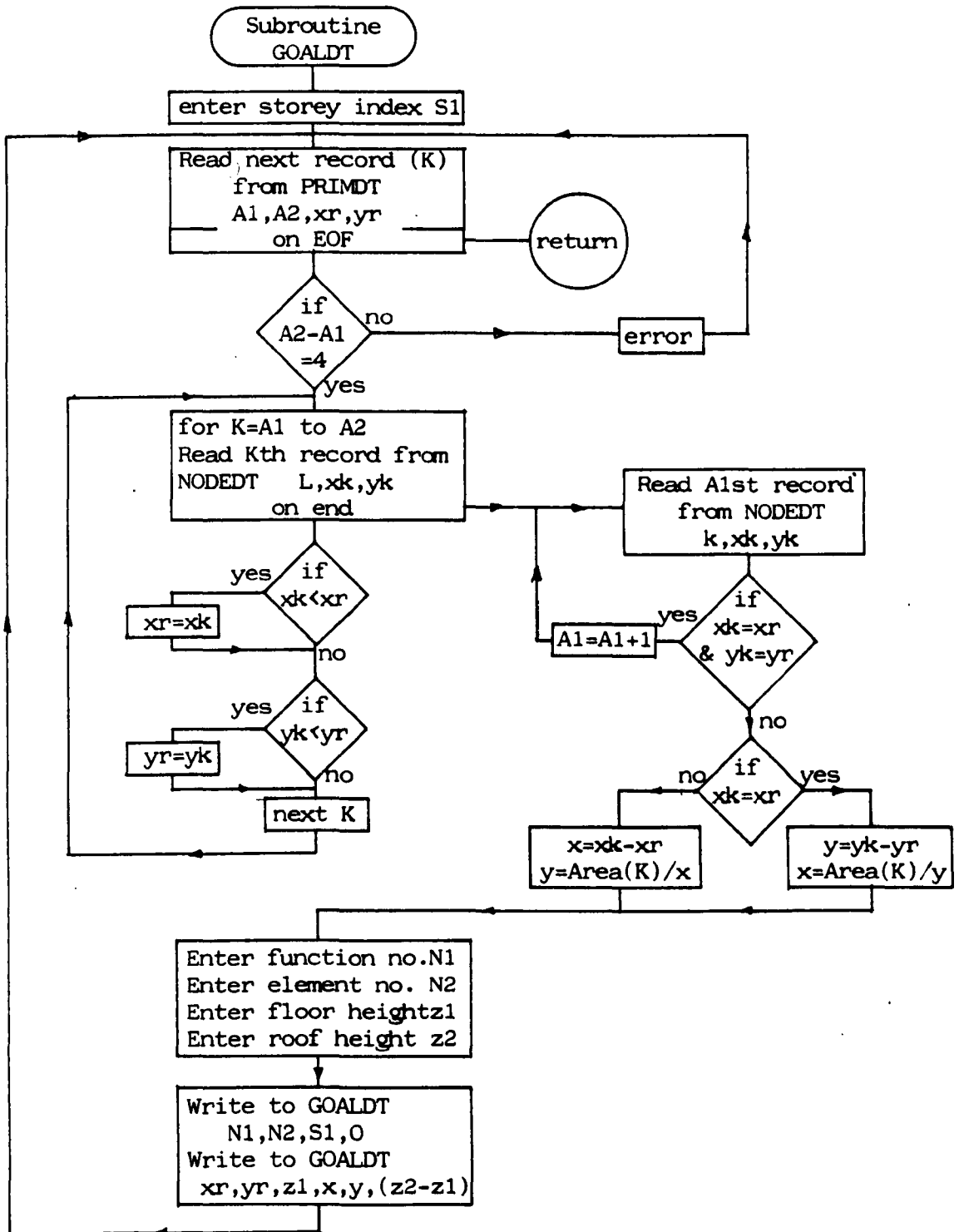


Figure 7.12 Flow Diagram Of Routine To Reformat ENIGMA File To A GOAL Data File.

be a strip footing 1 meter deep.

Similarly, the heat demand for a design day also assumes glazing ratios and 'U-values' as specified in the statutory regulations, the glazing being evenly distributed on all four facades. The calculations are carried out simultaneously with the cost appraisal, since they both require the same elemental breakdown.

#### 4.4 CRUDE VISUALISATION

The data held in the ENIGMA database may be re-presented in the form of a simple wireframe isometric projection.

The viewpoint is automatically taken to be from the south-west of the building, looking towards the north-east. The picture has to be scaled to fit within the terminal view 'window', hence the maximum and minimum plan coordinate points are abstracted from the line data file. When tilted, the picture has to be scaled by roughly one half in order to ensure that it fits on screen. Once the user has specified the floor and ceiling heights, a scaling factor (s) is calculated.

The coordinates of the external boundary shape of the building are then transformed to describe the two-dimensional projection of the three-dimensional solid by the following transform functions:

For converting 3D to 2D in the 'x' direction;

$$f(x) = s \cdot \cos 30^\circ \times (X - Y),$$

for the 'y' direction;

$$f(y) = s ( \sin 30^\circ \times (X + Y) + Z ).$$

These simplify to;

$$f(x) = 0.7071 \times s \times (X - Y), \text{ and}$$

$$f(y) = s \times ( 0.5 \times (X + Y) + Z ),$$

where s is the scaling factor,

X, Y, and Z are the three dimensional coordinates of  
of a point on the building model.

These functions are then applied to the drawing routine which will plot the two-dimensional picture on the terminal screen.

5 Resume

The ENIGMA programme is intended primarily to 'front-end' to other applications software programmes. However, the data held in the ENIGMA data structure can be used for crude appraisal and visualisation with little additional information requiring to be added to the database.

In order to be compatible with programmes requiring orthogonal geometry descriptions, ENIGMA requires that the user work within that regime; non-orthogonal geometries will fail if such an application programme is attempted. Similarly it is assumed that all three dimensional geometries can be described as an extrusion of their plan forms.

The inclusion of the draughting option using the Tektronix 4010 terminal is in anticipation of the inclusion of the editing and manipulation routines currently under development.

The ability to interpret geometry descriptions from one data structure is due to ENIGMA's power to decode drawings. There is little reason to prevent ENIGMA being used to front-end further applications programmes.

## Chapter 8

### Summary

1	Review	8.1	188
2	Assertion	8.2	189
3	Conclusion	8.2	189
4	Derivation Of Vocabulary	8.4	191
5	Caveats	8.7	194

## CHAPTER 8

1 Review

In the preceding chapters of this thesis, the structure and distribution of the architectural profession has been described, noting the predominance of a large number of small practices. It was also noted that few architectural practices have adopted computer aids.

From the range of services which architects offer to clients, the actual design function was singled out as being the most awkward aspect in which to incorporate computer aids. A design morphology was described in which the design function was analysed.

The inherent difficulties in attempting to describe geometry descriptions of buildings to computers was examined with particular reference to the mismatch between the human's and machine's 'perception' of graphic data, and methods of geometry description. The effects of the various methods of geometry description upon the data structures necessary to hold graphic data so generated were examined. It was noted that drawings are two-dimensional representation of three-dimensional objects, and that this could cause difficulties describing real objects to computers.

Having isolated the factors which produce the mis-match between computer systems and the architects traditional methods of working, four examples of existing graphic handling programs were described in order to ascertain how they dealt with them.

Following this survey, algorithms were described which together enable traditionally drawn architectural plan sketches to be refined and enhanced, and then translated into a general hierarchical data structure without the need for further operator intervention.

A suite of computer programmes which utilise these picture translation techniques was subsequently introduced and described.

2 Assertion

The algorithms described in Chapter 6 which, carried out at the earliest stages in the design process utilising traditional pencil and paper techniques, would interpret line drawings into a general data structure would have the following advantages:

- 1 Such a system would be in keeping with architect's traditional working practices, thus reducing the impact of 'technological alienation'.
- 2 It would permit the computer to be consulted at the earliest design stages when performance predictions would be of greatest benefit.
- 3 Eliminate the need for painstaking manual digitisation methods currently required in the use of most CAAD software.
- 4 It would allow the designer to concentrate upon design rather than upon the operation of the computer.
- 5 It would produce a database which could be tapped by other CAAD programmes requiring a geometric description of the proposed building.
- 6 It would be easy for inexperienced users to master, and
- 7 Would require minimal interaction on the users part, thus reducing the possibility of the introduction of errors.

3 Conclusion

In order to realise the above advantages, a graphics interface system has been developed as described in Chapters 6 and 7.

The system, ENIGMA, is the acronym standing for an ENhanced Interpretive Graphics Module for Architects. The system conceptually imposes few limitations on the user, working with sketch drawings. The system requires greater interaction should the user wish to generate geometries through the use of the Tektronix 4010 graphics terminal. The major reason for the inclusion of this option is the anticipation of the editing and data manipulation routines, which cannot be sensibly carried out on the digitising tablet.

CAAD systems interpreting building plans using the graphics terminal to create the geometry data are frustrating to the user, whilst those which allow the user freedom of input methods, including for the continuous digitisation of sketch drawings, either generate inflexible data structures, or have failed to apply their data structures to useful purposes so far as architects are concerned.

ENIGMA also offers an orthogonal mode of operation, anticipating that the user will be using the digitiser as a drawing board, complete with parallel motion devices, in order to draw orthogonal plan drawings. However, ENIGMA can cope with freehand sketches which are near orthogonal provided that each line is drawn separately in emulation of drawing with 'T' and set-square.

The various parameters effecting the clarity and extent of the drawing enhancement and interpretation of the drawings can be altered to suit the individual user's drawing style and fist.

The particular value of the ENIGMA system, is its ability to cope with drawings produced at the very formative stages in the design process, when the architect is beginning to formulate a building form as a potential solution to the design problem. As with traditional techniques the architect is encouraged to manipulate ideas through the medium of rough sketches, discarding unpromising solutions and amending those which show potential. Because the generation of the building model is automatic, there is no penalty for rejecting models which would otherwise involve a great deal of time and effort to produce.

Once a promising idea has been roughed out, the designer can very quickly get a refined and enhanced straight line interpretation of his tentative design solution, and receive from the appraisal routines an indication of this solutions performance. Moreover, the data reformatting capabilities of ENIGMA enables him to quickly subject his design proposal to more stringent and detailed appraisals, by recourse to other software application programmes.

Unlike other CAAD graphic interfaces, the ENIGMA system permits the user to subdivide shapes. This is illustrated in Figure 8.1 overpage, where for example, two rectangles, defined perhaps as shape primitives, are made to overlap. Generally, such a picture would result by defining two rectangles, and would be recorded as such in the consequent data structure. If the user then wanted to refer to the common area bounded by the overlap, he would be unable to do so. The interpretation capability of ENIGMA dissociates the drawing into its constituent shapes, irrespective of the manner in which they were defined when drawn. This means that the designer can create his building designs by either dividing spaces, as with for example, an infill site, or by aggregating together individual spaces. These strategies which designers often use simultaneously are not catered for by the CAAD systems so far examined.

The result of the interpretation of the picture shown in Figure 8.1 is shown in Figure 8.2. The resulting interpretation means that each individual space is uniquely identified, not as a fixed shape type, but as a description which is open to re-interpretation with each modification.

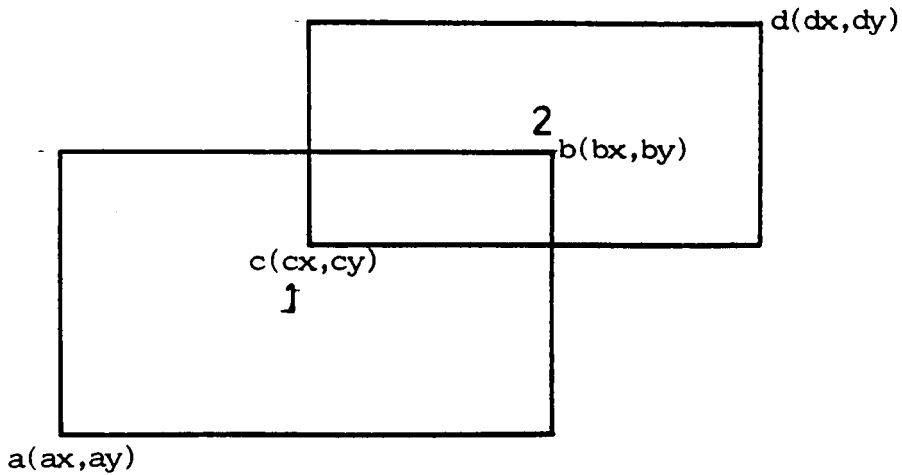
The ENIGMA system is intended as a graphics handling interface, and not as a self contained set of programmes as a finished, and complete product. Its purpose is to allow the designer to draw naturally, and to automatically develop a wider communication language than that with which existed on data input. This means that the system 'knows' significantly more about the data than the designer has given it; this knowledge being implicit in the data structure.

As an interface, the intent is to extend the range of dialogue between the man and machine by placing the onus of interpretation upon the machine. The communication language could possibly be extended further by more automatic procedures, some of which are discussed below.

#### 4 Derivation Of Vocabulary

From the ENIGMA database as described in Chapter 6, page 6.41,

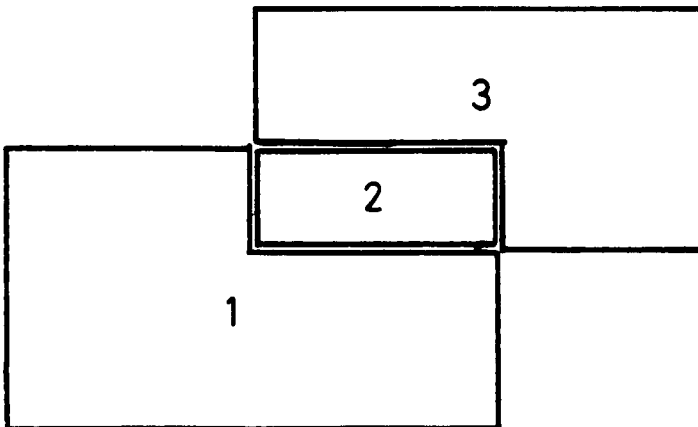




Description:

Space 1  $ax, ay, bx, ay, bx, by, ax, by$   
 Space 2  $cx, cy, dx, cy, dx, dy, cx, dy$

Figure 8.1 Generated Geometry Description Through Hardware



Description:

Space 1  $ax, ay, bx, ay, bx, cy, cx, cy, cx, by, ax, by$   
 Space 2  $cx, cy, bx, cy, bx, by, cx, by$   
 Space 3  $bx, cy, dx, cy, dx, dy, cx, dy, cx, by, bx, by$

Figure 8.2 Interpreted Geometry Description By ENIGMA

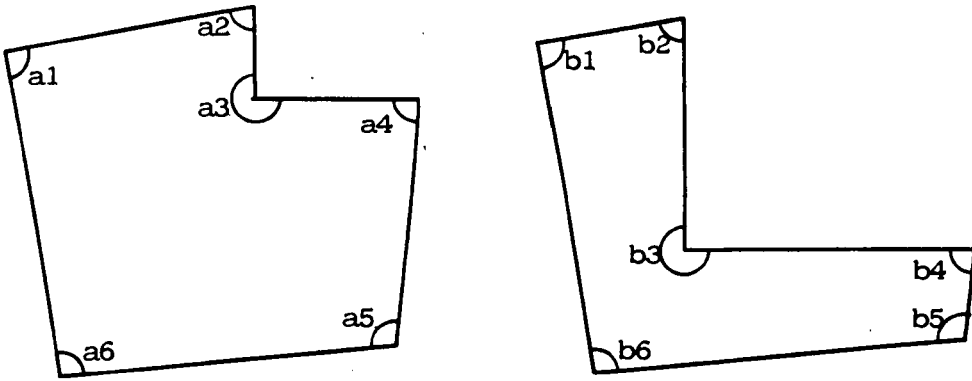
it will be seen that from crude line sketch drawings, the user vocabulary has been extended to include noetic characteristic descriptions of shapes, and the ability to abstract the surrounding silhouette of the plan form.

Although the concept of 'shape' is ambiguous, since we cannot deal with the 'shapeness' involved in the distinction between for example, those between a rectangle and square as the rectangle approaches a state of 'nearly square', the concept of outline or silhouette is exact. (It is of interest to note that the human can only distinguish between nearly square rectangles and squares only by mensuration). Consequently, the user can refer to outline in a machine cognisable sense.

Extending these concepts further, any two shapes which have a line as a common boundary are adjacent: If any two shapes have an equal number of sides, the sides having respectively equal lengths and subtending equal internal angles, and enclosing equal areas, then the two shapes are congruent: If an inversion about the 'x' axis of one shape produces a congruent version of another, then one shape is an inversion of the other: Similarly, if an inversion of one shape about the 'y' axis produces a congruent version of another, then one shape is a handed version of the other.

There are further propositions which may be postulated to deduce a class of 'fuzzy' or inexact relationships which may, with an associated level of certitude, be defined: Two shapes are 'generically similar' if they have the same number of sides which subtend the same angles as illustrated in Figure 8.3 overpage. Two spaces are exactly similar if in addition to being generically similar, the ratios of their respective edges remains constant.

One shape may be said to lie to the right or left of another by consideration of their respective maximum and minimum 'x' coordinates, or indeed by consideration of their respective centroids. Similarly, by consideration of their coordinates relative to the 'y' axis, we may perceive an appreciation of the ideas of up and down with respect to the page on which the picture



Two shapes of equal number of sides where;  
 $a_1=b_1$ ,  $a_2=b_2$ ,  $a_3=b_3$ ,  $a_4=b_4$ ,  $a_5=b_5$ , and  $a_6=b_6$

Figure 8.3 Definition Of Generic Similarity  
 Between Two Shapes

is drawn.

The rationale outlined above holds the possibility of the automatic derivation of an interface system which would include in its user vocabulary the adjectives and nouns underscored above.

Not only does this extended range of vocabulary permit a wider and more general range of discourse between man and machine, but it would appear to resemble the type of exchange in use with some knowledge engineered systems. If this seeming compatibility is indeed real, then ENIGMA might prove to be a good interface to front-end onto a logic based system to handle the graphic data capture. More interestingly, it would also seem to bridge between the calculation power of conventional prescriptive programming techniques and the data interogation power of the descriptive logical language based systems.

## 5 Caveats

The ENIGMA system described in the forgoing chapters of this thesis is currently a prototype. It is recognised that only through observation of its use in practice will the hypothesis that it will make CAAD systems more accessible to architectural practitioners, be confirmed or refuted.

The fundamental research work contained in this thesis was devoted to the automatic data capture and interpretation routines described in Chapter 6. Consequently, and in view of the time constraints of the project, it is recognised that parts of the ENIGMA suite of programmes could be much improved.

The following chapter describes those areas where more work is needed, particularly in the editing routines, despite the fact that many of the algorithms have already been considered as part of another research project within the Scott Sutherland School of Architecture, (SERC Funded Research Project No.82301583).

Additionally, ENIGMA is capable of expansion in order to do some functions in a better and more efficient manner, and to include procedures to increase its versatility.

## Chapter 9

### Suggested Further Work

1	Introduction	9.1	197
2	Editing Functions	9.1	197
	2.1 Adding A Line	9.4	200
	2.2 Adding A Shape	9.4	200
	2.3 Repeating A Shape	9.5	201
	2.4 Deleting A Shape	9.6	202
	2.5 Deleting A Line	9.8	204
	2.6 Moving A Point	9.11	207
	2.7 Moving A Line	9.12	208
	2.8 Moving A Shape	9.13	209
3	Other Editing Functions	9.15	211
	3.1 Orthogonalisation	9.16	212
4	Interpretation Of Multi-Storey Buildings	9.17	213
5	Extensions To ENIGMA	9.17	213
6	Conclusion	9.18	214

## CHAPTER 9

1 Introduction

ENIGMA, as previously described is a prototype graphics interface system, and consequently is capable of being much improved. The incorporation of these improvements, and the extension of the system to handle more tasks and become more flexible as a result, has not proved possible within the time constraints within which this project was carried out.

Apart from the removal of local idiosyncracies and quirks inherent in any new computer software, the most immediate requirement for inclusion into the body of the code are the routines necessary for the editing and manipulation of the database once captured.

Secondly, the ENIGMA system might be expanded to cater for the description of multi-storey buildings.

Thirdly, there is great scope for improving the crude appraisals which ENIGMA carries out. However, since the ENIGMA system has been devised as a front end for use with other applications programmes, it would be preferable if translation routines were developed such that data files could be created for direct input into already existent appraisal programmes, thus avoiding the duplication of effort.

These improvements and extensions, when included, might result in a system as depicted in Figure 9.1 overpage, which for example makes reference to two further appraisal programmes, ESP, an Environmental Systems Performance, and GABLE which has already been described in Chapter 5, section 4.

2 Editing Functions

The creation of geometry data through the use of the tablet in continuous digitisation mode with an ink stylus precludes the implementation of editing or manipulation functions direct upon the digitiser. The sketch drawing represents the current state of the drawing data, and unless an erase function can be devised

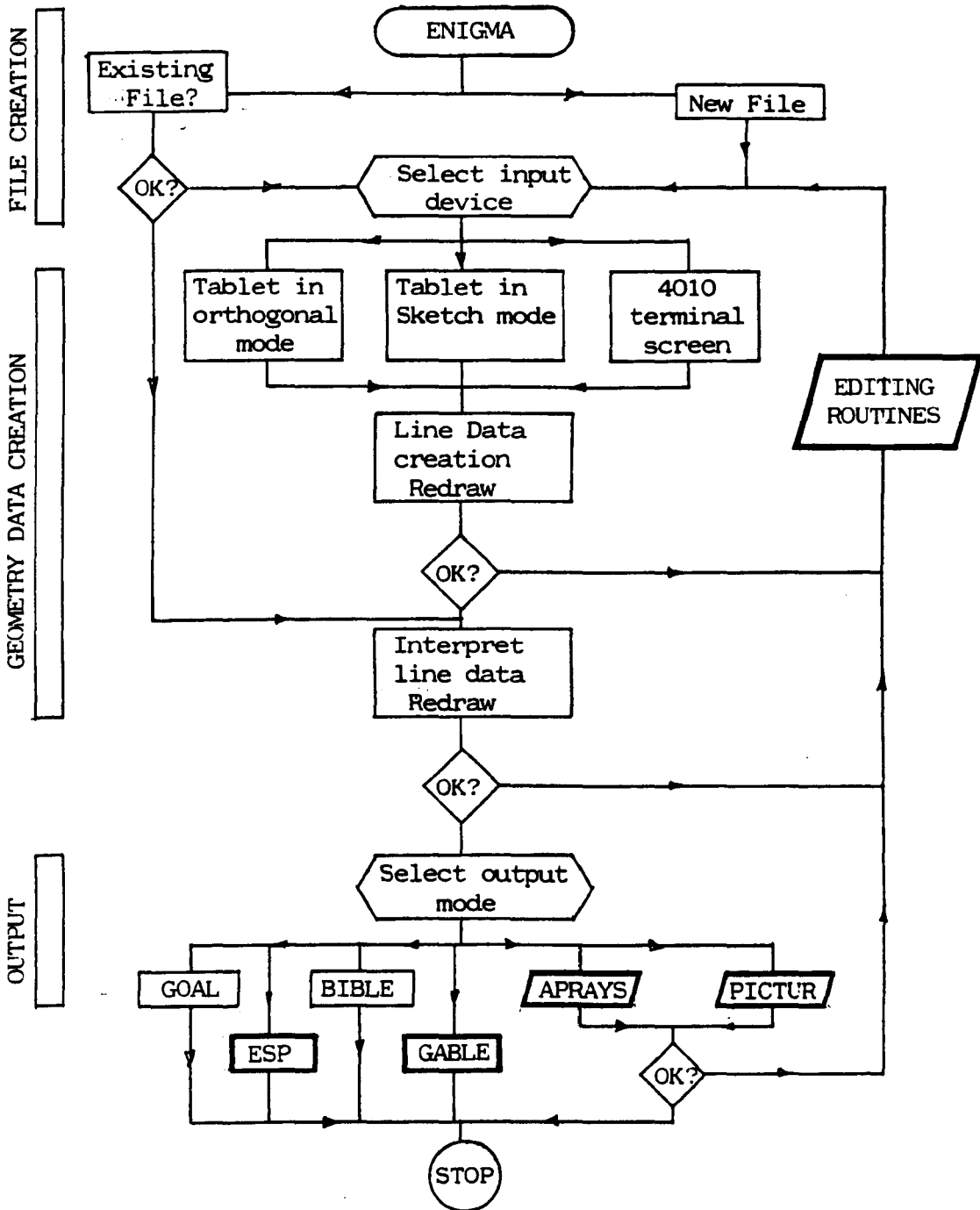


Figure 9.1 Schematic Representation Of ENIGMA Flow Path.

which could simultaneously clear the data referred to from the data structure, and remove the inked line from the paper, any amendments will cause confusion on the ink drawing. Consequently, editing functions would have to be carried out on the Tektronix 4010 graphics terminal.

Reference has already been made to a current research project part of which deals with editing algorithms which could be utilised to manipulate the ENIGMA database. The algorithms devised have been developed for use upon a Tektronix 4054 graphics terminal to take advantage of the dynamic graphics which this piece of equipment offers. The Tektronix 4054 has its local processor, and has been developed for use in stand alone mode.

The system as envisaged requires that ENIGMA data files be sent through the DEC20 system and loaded into the Tektronix 4054 local tape drives. The 4054 is then disconnected from the DEC20 system, and the editing and manipulation functions could then be enacted off-line. Moreover, these editing and manipulation routines are written in BASIC, as opposed to ENIGMA's FORTRAN, and also relies on firmware functions not available on the TOPS20 system.

One consequence of the relatively limited storage capacity of the 4054 is that the user is permitted a maximum of six junctions with any one line. This may not be adequate for all architectural plan drawings.

An attempt was made to link the Tektronix 4054 directly with the 4010, such that the latter would act as a host computer to the former terminal and so control the digitising tablet. This attempt was unsuccessful although the manufacturers believe it possible.

The functions covered in this Tektronix 4054 coding permits the user to:

- a create and store pictures in a plan chest,
- b locate a line,
- c locate a shape,
- d add a line,
- e delete a line,



- f delete a shape,
- g move a line,
- h stretch a shape, and
- i repeat a line.

However, the 4054 editing and manipulation routines convert the ENIGMA data files into an entirely different data structure, and it would be useful if these routines were rewritten to operate directly with that of ENIGMA. The following section discusses some of the problems and strategies involved in doing so.

## 2.1 ADDING A LINE

The ability to add a line already exists in the current implementation of ENIGMA. This is effected on the digitising tablet directly. However, to ensure data consistency, the overlap check, (see Chapter 7, section 3.3), would have to be implemented, and overcome the potential problem of 'bridging'.

## 2.2 ADDING A SHAPE

If the digitising tablet is being used, shapes may be added as a series of lines as described above.

If, however, the 4010 terminal is being used, it is first necessary to specify which shape primitive the user wishes to add. Again, those shapes not amongst the options offered by the terminal may be drawn as a series of lines, as with the digitiser.

Once specified, and coordinate values entered by use of the thumbwheel driven cross-hair cursor, it is necessary to ensure that any overlaps, should they occur, are resolved. The newly added lines are appended to the line data list, and on completion of the addition, further shape and line additions may be made.

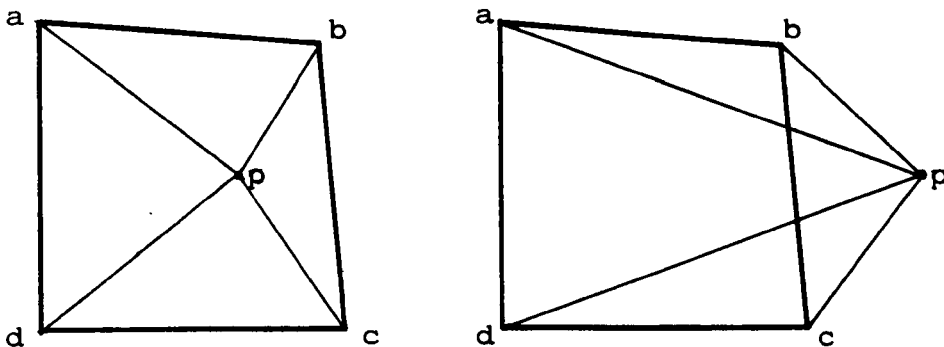
All of the following routines are assumed to operate through the use of the Tektronix 4010 graphics terminal.

## 2.3 REPEATING A SHAPE

The first necessity in this instance is to identify which existing shape the user wishes to repeat. It is assumed that the original picture has undergone the interpretation process, otherwise the ENIGMA data base will have no shape data to refer to.

Having informed the system that the user is about to point to an existent shape, he positions the cursor inside the shape he is attempting to identify for the computer.

A scheme for identifying the selected shape is illustrated in Figure 9.2 below.



Point p declared by the user.

a) Point inside abcd

$$\begin{array}{l} \text{sum of angles} \\ \text{apb} + \text{bpc} + \text{cpd} + \text{dpa} = 360^\circ \end{array}$$

b) Point outside abcd

$$\begin{array}{l} \text{sum of angles} \\ \text{apb} + \text{bpc} + \text{cpd} + \text{dpa} < 360^\circ \end{array}$$

Figure 9.2 Scheme For Determining Whether A Point Lies Inside A Shape

The shape data held in the ENIGMA files NODEDT and PRIMDT are a chained path of node points defining the periphery of each shape. If the angles subtended between the edges of the shape and a given point are summed, then the total angle turned through, given that a negative line direction will return a negative angle subtended, will be a complete revolution, ie.  $360^\circ$ , if the point is within the shape, despite concavities. Any point outside will return a total angle of less than one revolution. This process would not determine which shape has been pointed to if shapes are

nested. The process would however, stop as soon as the first shape is encountered within which the point lies, so the user could instruct the computer to continue the search until it arrives at the correct shape, should its most recent choice be wrong.

Having specified which shape the user wants repeated, he would then have to relocation the reference point, (held in the latter two data bits of the located shapes PRIMDT record), to where he wants to place the repeated shape. The modifications which must take place in the ENIGMA database to contend with the alterations so wrought are shown in Figure 9.3 overpage.

If, for example, the user wishes to reproduce shape number 2 (as recorded in the data structure shown in the figure), at a new location given by the new reference point coordinates (50,50), then the differences between the new and old reference point coordinates are carried through all of the node points held in NODEDT an appended at the end of the file as new shape data, with the requisite index record appended to PRIMDT.

The above schema demonstrates the principle of shape repetition which assumes that the repeated shape does not interact with any existing shapes in the database. Should this happen, then first, the new line must be checked for overlapping, secondly the entire line data list would have to be subjected to the line dissociation routine, and the interpretation process re-invoked in order to clarify the new shape subdivisions.

Since in the following descriptions we will be referring to the effects various editing and manipulation processes have upon the database shown in Figure 9.3, the picture to which this data refers is shown in Figure 9.4 on page 9.8.

## 2.4 DELETING A SHAPE

Deletion of a shape from the database will introduce records containing null values as the data is erased, and these voids have to be removed. Additionally, removing a shape might well effect the topology of the remaining shapes in the picture, which means that

## LINEDT

	Start Pt x1	Stop Pt y1	Stop Pt x2	Stop Pt y2
1	580	3000	580	2320
2	580	3000	1550	3000
3	1550	3000	1550	2320
4	1060	1650	1550	1660
5	580	2320	1060	2320
6	1060	1650	1060	2320
7	580	1640	580	2320
8	1550	2320	1550	1660
9	580	1640	1060	1650
10	1060	2320	1550	2320
11	50	50	540	60
12	540	60	540	520
13	540	20	50	520

new lines generated  
by repeating a shape.

## PRIMDT

	Address first	Address last	Location x ref	Location y ref
1	1	5	580	3000
2	6	9	1060	1650
3	10	13	580	2320
4	14	17	50	50

shape 2 repeated as 4  
produces a difference  
of -1010 on the X-axis,  
and -1600 on the y

## NODEDT

	line.i.d	End Node x	End Node y
1	1	580	2320
2	5	1060	2320
3	10	1550	2320
4	3	1550	3000
5	2	580	3000
6	4	1550	1660
7	8	1550	2320
8	10	1060	2320
9	6	1060	1650
10	5	1060	2320
11	6	1060	1650
12	9	580	1640
13	7	580	2320
14	11	540	60
15	12	540	520
16	13	50	520
17	14	50	50

+ dx of -1010 and  
dy of -1600.  
produces the node  
coordinates stored  
in locations 14-17.

Figure 9.3 The Effect Of Repeating An Existing Shape On The ENIGMA Data Structure.

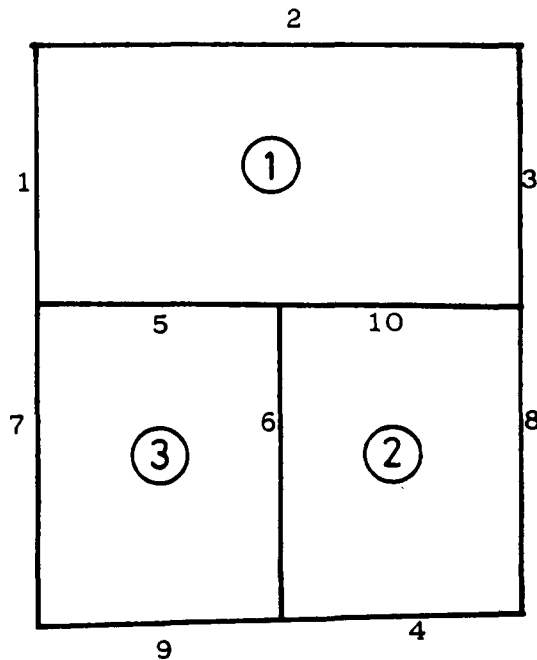


Figure 9.4      Figure Described By The Data Base  
Described In Figure 9.3.

the picture which remains following the deletion will have to be reinterpreted. It is possible, therefore, that a shape totally surrounded by others, can never be deleted, since on reinterpretation, that shape will be traced out anew.

In actual fact, only the null records in the line data need be removed. This is made apparent in Figure 9.5 overpage, where an example of shape deletion is given.

A further problem arises, since those lines common to another shape, besides that to be deleted, must nevertheless be preserved in the line data list.

Several shapes can be deleted in sequence without the need for re-interpretation, provided the data consistency of the line data list is preserved.

## 2.5 DELETING A LINE

Prior to modifying the database by the deletion of a line, the computer must be told which line is the subject of scrutiny.

LINEDT

	Start Pt x1	y1	Stop Pt x2	y2
1	580	3000	580	2320
2	580	3000	1550	3000
3	1550	3000	1550	2320
4	1060	1650	1550	1660
5	580	2320	1060	2320
6	1060	1650	1060	2320
7	580	1640	580	2320
8	1550	2320	1550	1660
9	580	1640	1060	1650
10	1060	2320	1550	2320

LINEDT

	Start Pt x1	y1	Stop Pt x2	y2
1	580	3000	580	2320
2	580	3000	1550	3000
3	1550	3000	1550	2320
4				
5	580	2320	1060	2320
6	1060	1650	1060	2320
7	580	1640	580	2320
8				
9	580	1640	1060	1650
10	1060	2320	1550	2320

becomes

PRIMDT

	Address first	last	Location x ref	y ref
1	1	5	580	3000
2	6	9	1060	1650
3	10	13	580	2320

PRIMDT

	Address first	last	Location x ref	y ref
1	1	5	580	3000
2				
3	10	13	580	2320

becomes

NODEDT

	line id	End Node x	y
1	1	580	2320
2	5	1060	2320
3	10	1550	2320
4	3	1550	3000
5	2	580	3000
6	4	1550	1660
7	8	1550	2320
8	1060	1060	2320
9	1060	1060	1650
10	5	1060	2320
11	1060	1060	1650
12	9	580	1640
13	7	580	2320

NODEDT

	line id	End Node x	y
1	1	580	2320
2	5	1060	2320
3	10	1550	2320
4	3	1550	3000
5	2	580	3000
6			
7			
8			
9			
10	5	1060	2320
11	6	1060	1650
12	9	580	1640
13	7	580	2320

becomes

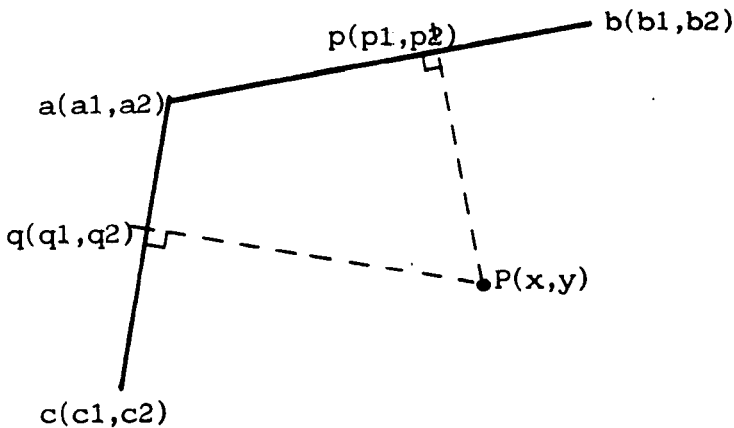
Figure 9.5 The Effect of Shape Deletion On The ENIGMA Data Base.

Note that lines common to two shapes must be preserved in LINEDT despite deletion of a shape to which it is a boundary.

This specification may be achieved by calculating the tangential distance between a point placed by the user manipulating the cursor, near to the line in question. The line data list is then read in sequence, and the perpendicular length to all lines calculated. The line nearest to the point is identified as that for deletion.

Further possibilities arise, in that should the perpendicular strike the line in question near to one of its ends, then the system could construe that the user wishes to edit that end; if in the middle, that the user wishes to manipulate the entire line.

Figure 9.6 illustrates the principle of line selection.



$$p1 = (y-a2) + (x+a1) \cdot \frac{(a2-b2)}{(a1-b1)} \quad \frac{(a2-b2)^2 + (a1-b1)^2}{(a1-b1) \cdot (a2-b2)}$$

$$p2 = y + x \cdot \frac{(a2-b2)}{(a1-b1)} - \frac{p1 \cdot (a1-b1)}{(a2-b2)}$$

$$q1 = (y-a2) + (x+a1) \cdot \frac{(a2-c2)}{(a1-c1)} \quad \frac{(a2-c2)^2 + (a1-c1)^2}{(a1-c1) \cdot (a2-c2)}$$

$$q2 = y + x \cdot \frac{(a2-c2)}{(a1-c1)} - \frac{q1 \cdot (a1-c1)}{(a2-c2)}$$

If  $(a1-b1)/(p1-a1) \geq 4$  reference is to end point a or

If  $(a1-b1)/(p1-a1) < 1.33$  reference is to line ab

Else reference is to end point b.

Figure 9.6 Scheme For Specifying A Line

Reference to an end point might be used should the user wish to 'drag' an end point to a new position. This would introduce further problems, since it would effect all shapes which held the point to be moved in common, as will be discussed in due course.

The deletion of a line from the LINEDT data list may well produce 'knock-on' effects upon the consistency of the data held in both PRIMDT and NODEDT, since it is likely that a line deleted is common to two shapes. Unlike the case of deletion of an entire shape, the remaining boundaries of the two formerly adjacent shapes now describe one new shape by virtue of the removal of the separating line. Several line deletions may be done in sequence, but the line data list would have to be cleared of the void records before attempting a re-interpretation of the amended picture.

## 2.6 MOVING A POINT

As demonstrated above, the user can point with the cursor to a node which he wants to move to a new position.

Once the lines describing a drawing have been split into discrete line elements, (see Chapter 6, section 4.4), data consistency may be maintained in LINEDT when a point fixing the end position of a line (or lines) is moved. When the point has been specified and the new point position defined by means of the new position of the cross-hair cursor, the old coordinates of each occurrence of the old point are substituted by the new. However, there are situations in which the overall topology is seriously disrupted. Such a situation is illustrated in Figure 9.7 overpage.

In this example, the point b has been moved such that the lines ab and ac now intersect several existing lines, creating a number of additional new shapes. It is also possible that the point moved is the reference point held in PRIMDT.

To ensure that these possible consequences are filtered out, the data newly created in LINEDT would have to be reinterpreted. If no additional line segments are generated, the system might be able to modify the corresponding changes throughout the rest of the data structure.



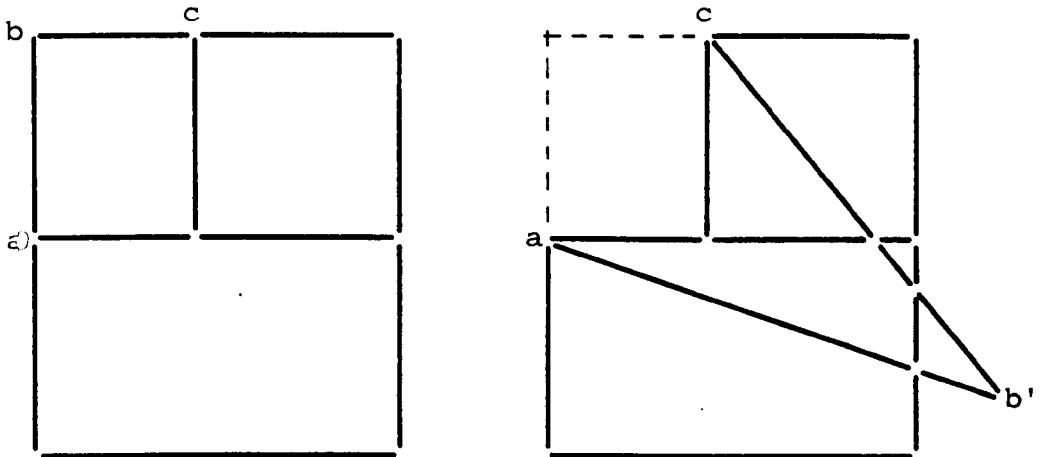


Figure 9.7 Topological Disruption Caused By Moving A Point

## 2.7 MOVING A LINE

This operation is similar to the previous point moving process, but here, obviously, two points are moved simultaneously. The line to be moved is specified as described previously, and the new point selected. The line will be kept at the same length, and moved to a position parallel to its old, but passing through the specified point. Those line connected to the old lines end points are extended to the new positions. There is a likelihood of far greater topological disruption to the data base than with a single point move.

To overcome these problems, the line overlay check must be carried out on all altered lines in case the side lines to that moved have regressed. The line data set would subsequently have to be reinterpreted.

Should these effects not occur, then Figure 9.8 on the following page illustrates the effect of a line movement upon the database. In the example shown, line number 10 is to be moved. However, from NODEDT it is seen that line 10 is connected to lines number 3, 5, 6, and 8, and any change to the coordinate positions of line 10 will need to be followed through to these other lines. The maintenance of the topology of the picture

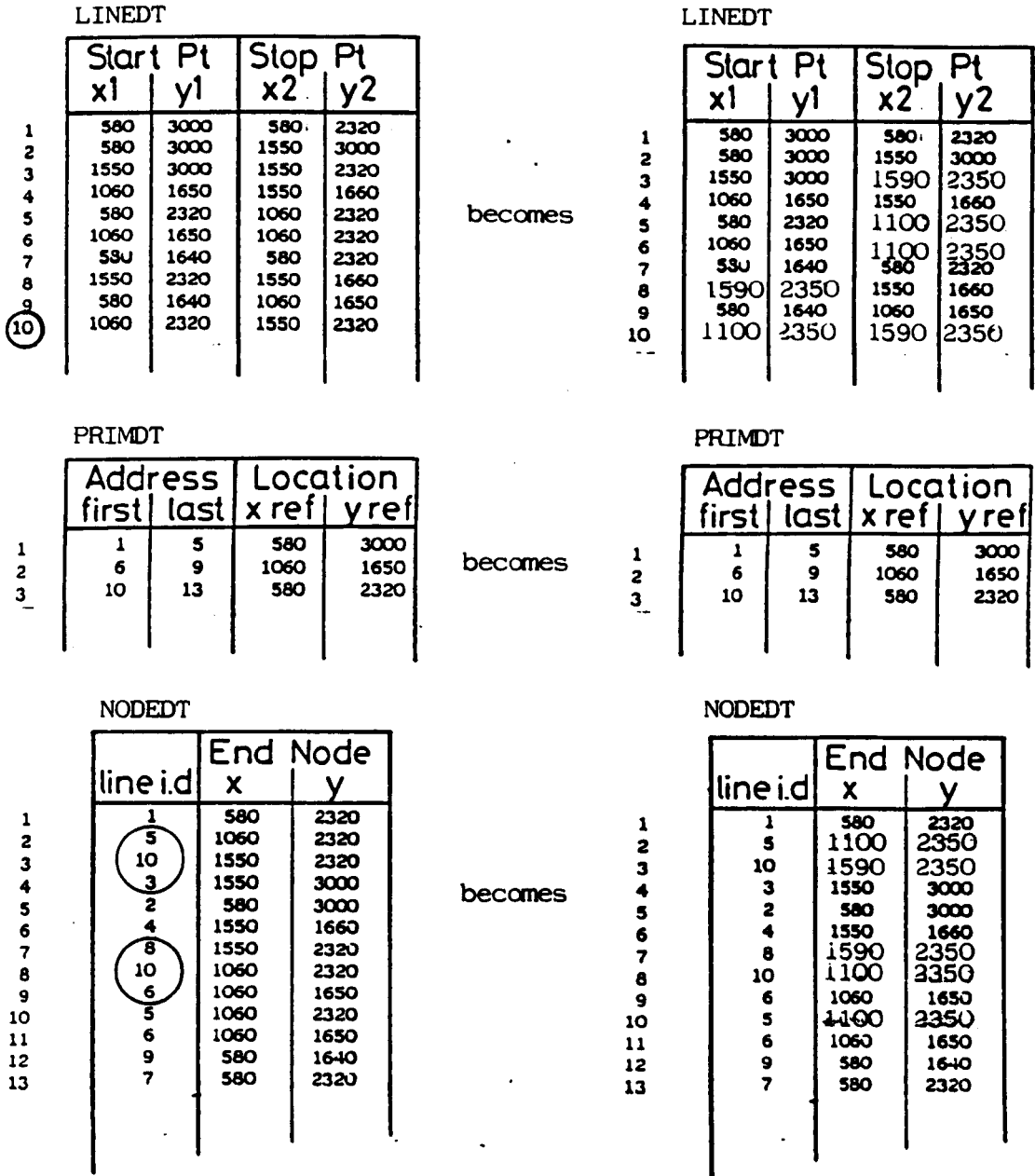


Figure 9.8 The Effects Of Moving A Line On The ENIGMA Data Base.

Line 10 is to be moved.

causes distortions to be induced as illustrated in Figure 9.9 overpage.

## 2.8 MOVING A SHAPE

There are two possible implementations for moving a shape. The first is illustrated in Figure 9.10 overpage. Here, the shape

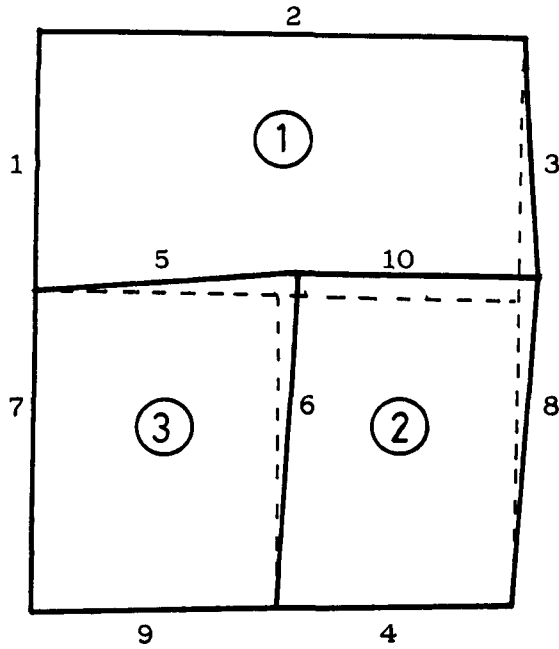


Figure 9.9 The Alteration To A Picture Through Moving A Line

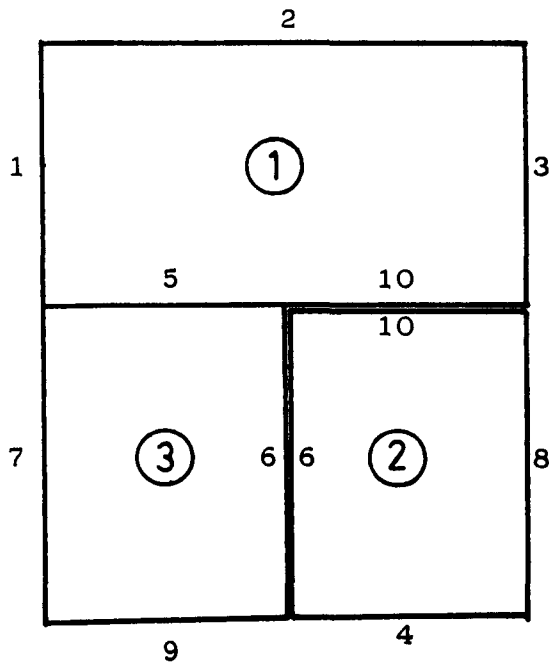


Figure 9.10 Duplication Of Common Lines Prior To A Move Shape Operation

to be moved is 'detached' from the picture, and common lines duplicated in the LINEDT list. The shape can then be moved as a complete unit.

The second option would maintain the original topological links, such that when the shape was moved, the adjoining lines would be dragged with it, as with the line move process. All nodes connected with the moved shape would be moved through a similar 'x' and 'y' increment, dictated by the difference between the reference point and the newly specified position.

The first option requires rather more calculation than the second. Detaching a shape from the remainder of the picture involves alterations to all three ENIGMA data lists. Additionally, there is a potential problem where the shape is moved adjacent to two existing shapes. The line overlap checking routine has to be invoked, before subjecting the line data file with the new data to the interpretation routine.

These problems will also apply where an existing shape is to be duplicated.

### 3 Other Editing Functions

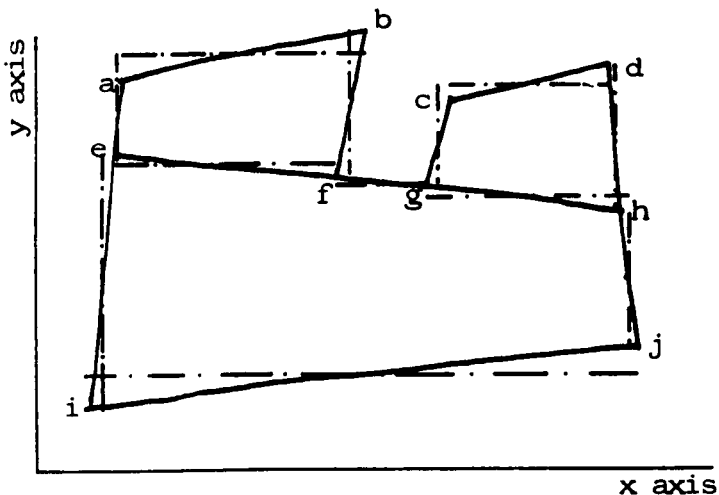
The editing and manipulation functions so far considered are those which effect parts of the data structure, and whose application is intended to make marginal adjustments to the picture. There are, however, a set of desirable data manipulation processes which are of a higher order. Those which might be considered in this category are the processes to orthogonalise a non-orthogonal picture, those to hand or invert individual shapes, and perhaps those to reproduce aggregations of individual shapes. Similarly, rotations applied to shapes, and of course, the treatment of arcs and circles.

These latter operations have not been considered to any depth. However, the process for orthogonalisation is considered a very worthwhile addition to ENIGMA, although the problems associated with its implementation are particularly onerous.

## 3.1 ORTHOGONALISATION

Many CAAD applications programmes process rectilinear geometry descriptions, ie. all lines drawn parallel to the two axes. By virtue of the inaccuracies inherent in freehand sketching, it is unlikely that true vertical and horizontal lines will be drawn, despite the ENIGMA enhancing routines. It would also be of use to be able to convert nonorthogonal drawings into their rectilinear equivalents, so that programmes limited to this geometry discipline could be run using the approximated graphic data.

In order to make a line coaxial it must be rotated about a fixed point along its length to bring it parallel to either the 'x' or 'y' axis. This necessarily will displace at least one end point of a pitched line. Figure 9.11 demonstrates the effect orthogonalisation would have on the topology of a drawing caused by rotating its constituent lines about their mid-points.



Key:

- original sketch
- - - - orthogonalised sketch

Figure 9.11 Topological Discontinuity Caused By Orthogonalisation

Several consequences will be noted: Firstly, unless the gradients of the lines are small, the junctions become separated beyond the point which the ENIGMA tolerance values can rectify the separations. Secondly, some lines have become stepped. This

results in, thirdly, simple shapes becoming more complex, with the introduction of many more sides to formerly simple figures.

If as in the GRAMP graphic regime, only rectangular figures are permitted, there is the further problem of reducing these more complex shapes to an aggregation of constituent rectangles.

#### 4 Interpretation Of Multi-Storey Buildings

Provided that each storey is treated as a separate drawing, the resulting interpretation in ENIGMA can be built up by concatenating the output files by use of the TOPS20 file editing routines, although this has to be done, currently, independently of the ENIGMA system. It would therefore, be more convenient if these facilities were included within the body of the main program. The first requirement is for a planchest in which to store the different drawings in an organised manner.

The planchest would treat a building description as a set of overlays which could be superimposed, yet each drawing referring to a specific level. However, the major problem remains where a space extends through more than one level, or is not a simple extrusion of its (lowest) plan form throughout its height.

Potential problems exist where the various storey plans have differing silhouettes. This problem relates to overhangs. There is also the potential problem where a plan includes unconnected shapes, such that there is a multipart silhouette.

#### 5 Extensions To ENIGMA

One method to deal with many of the possibilities outlined above could be circumvented by looking to other existent programs which already cope with the eventualities described. Indeed, the intention behind the ENIGMA system is the desire to create geometry data files for a range of such programmes.

The translation of the ENIGMA database into input files for both the GOAL and BIBLE programs has been described previously in Chapter 7. An examination of the GABLE data structure, (insofar as

it is possible to get access to commercially available source code), would suggest that the data required exists in the ENIGMA data structure. Consequently it should be possible to produce a translation program, such that access is gained to GABLE's multi-storey shape interpretation routines, planchest, and appraisal modules.

Similarly, a cursory inspection of ESP, (Environmental Systems Performance), and VISTA, (Visual Impact Simulation Technical Aid), from the ABACUS Unit, University of Strathclyde, suggest that it might be possible to produce similar interpretation modules for these programmes. This is particularly true of VISTA which uses an advanced form of the data structure utilised by BIBLE.

## 6 Conclusion

The ENIGMA system so far described is deficient in graphics editing and manipulation facilities. The most pressing future work on ENIGMA should be aimed at making good this deficiency.

A useful extension to the system would be produced by the inclusion of an orthogonalisation programme.

Perhaps the most fruitful development would be the inclusion of further translation routines to enable access to other application programmes. This would have the great advantage of avoiding the duplication of effort involved in reconstructing existent software.

## Bibliography



## Bibliography

<u>Author</u>	<u>Title</u>	<u>Source</u>
Ankrentz.J.H.A.E	"Cybernetic Model Of The Design Process"	Design Studies Vol.2 No.1 Jan 1981
Angell.I.O	"A Practical Introduction To Computer Graphics"	Macmillan 1981
Architects Journal	"AJ Computing Club"	26th Jan 1983
Architects Journal	"Systems: Difficult To Choose, Easy To Use"	17th Dec 1980
Banham.R	"Theory And Design In The First Machine Age"	Architectural Press 1960
Becker.E	"The Birth And Death Of Meaning"	Penguin 1962
Benton.T, and Benton.c	"Form And Function"	O.U Press 1975
Bijl.A	"Can Computers Understand Designers"	PARC83 Proceedings On-Line Conferences Oct 1983
Bijl.A	"Dumb Drawing Systems And Know- ledge Engineering"	EdCAAD, Univ. Of Edinburgh Jul 1981
Bijl.A	"Integrated CAAD Systems"	EdCAAD, Univ. Of Edinburgh Mar 1979
Boreham.D, and Edmonds.E	"Extracting Shapes From Grey- Scale Images"	Int.J Of Man-Machine Studies, No.16 1982
Broadbent.G	"Design In Architecture"	John Wiley & Sons 1973
Capra.F	"The Tao Of Physics"	Shambhala Books 1976
Charlesworth.A.S, and Fletcher.J.R	"Systematic Analogue Computer Programming"	Pitman 1967
CICA	"Fact Sheet On The Use Of Comput- ers In The Construction Industry"	CICA Mar 1982
Clocks.in.W.F, and Mellish.C.S	"Programming In Prolog"	Springer-Verlag 1981
Collingwood.R.G	"The Idea Of Nature"	Oxford U.P 1945
Corcoran.D.W	"Pattern Recognition"	Penguin 1971
Coutts.N	"Low Cost Computong: Finance"	AJ 17th Dec 1980
Cross.N etal	"Design Method And Scientific Method"	Design Studies Vol.2 No.4 Oct 1981

<u>Author</u>	<u>Title</u>	<u>Source</u>
Cross.N	"The Coming Of Post-Industrial Design"	Design Studies Vol.2 No.1 Jan 1981
Daley.J	"Mystery Dissolved"	Building Design 28th Nov 1981
Darke.J	"The Primary Generator And The Design Process"	Design Studies Vol.1 No.1 1979
Date.C.J	"An Introduction To Database Systems"	Addison Wesley 1976
Davison.J.A	"Computer Aided Drawing - A Means To An End"	GMWComputers Ltd Nov 1981
Davy.J	"Man In The Belly Of The Beast"	Observer 15th Aug 1982
Dodwell.P.C	"Visual Pattern Recognition"	Holt Reinhart & Winston 1970
Eastman.C, and Schwartz.M	"Methods For Treating Variable Shaped Objects In Computer Aided Design"	BoCAAD, Univ. Of Strathclyde No.13 Oct 1973
Eurographics82	"Proceedings"	North Holland 1982
Evans.C.R et al.	"An Investigation Of Design Activity Using Analytic Time-Lapse Photography"	National Physical Laboratory Jul 1971
Findler.N.V	"Analogical Reasoning In The Design Process"	Design Studies Vol.2 No.1 Jan 1982
Flegg.G	"Boolean Algebra"	Transworld Student Library 1964
Giloi.W ed.	"Interactive Computer Graphics"	Prentice Hall 1978
Giloi.W	"Computer Graphics"	Berlin 1971
Giraud.C	"The PHP Representation"	EdCAAD, Univ. Of Edinburgh Feb 1984
Giraud.C	"Presque Half-Planes: Towards A General Representation Scheme"	Computer Aided Design Vol.16 No.1 Jan 1984
Guedj.R.A et al.	"Methodology Of Interaction"	North Holland 1980
Hamilton.I, and Burdett.J.R.F	"Computer Draughting Systems In Construction - The Buyers Problem"	CAD82 Proceedings Butterworths Apr 1982
Herot.G.F	"Graphical Input Through Machine Recognition Of Sketches"	ACM-SIGGRAPH Proceedings Vol.10 Part.2 1976
Hilton.R.K.	"A Computer Assisted Draughting System For Use In The Jobbing-Engineering Industry"	CAD82 Proceedings Butterworths Apr 1982

<u>Author</u>	<u>Title</u>	<u>Source</u>
Hopgood.F.R.A et al.	"Introduction To The Graphic Kernel System - GKS"	APIC Studies In Data Processing No.19 Academic Press 1983
Hoskins.E.M	"Integrated Computer-Aided Building And The OXSYS Project"	ARC Ltd June 1976
Jacobowitz.H	"Electronic Computers Made Simple"	W.H.Allen 1967
Jero.J et al.	"What's What And Where's Where: Knowledge Engineering In The Representation Of Buildings"	PARC83 Proceedings Oct 1983
Jone.E.M	"The Design Of Complex Buildings Using An Integrated CAD System"	CAD82 Proceedings Butterworths Apr 1982
Kernohan.D	"Externalising The Design Process"	Design Studies Vol.2 No.1 Jan 1981
Klinger.A et al.	"Data Structures, Computer Graphics And Pattern Recognition"	Academic Press 1977
Koestler.A	"Janus: A Summing Up"	Picador 1978
Koestler.A	"The Act Of Creation"	Picador 1964
Koestler.A	"The Ghost In The Machine"	Hutchinson 1967
Koyre.A	"From The Closed World To The Infinite Universe"	John Hopkins Univ. Press 1957
Kraal.L	"Implementation Experiences Of Design Applications"	PARC83 Proceedings On-Line Conferences Oct 1983
Kraus.R.T and Meyer.J.R	"Design: A Case Study"	MIT 1968
Lansdown.J	"Expert Systems: Their Impact On The Construction Industry"	RIBA Conference Fund 1982
Lawson.B	"How Designers Think"	Architectural Press 1980
Lawson.B	"ISAAC - A Technique For The Auto- matic Interpretation Of Spaces From Drawn Building Floor Plans"	CAD82 Proceedings Butterworths Apr 1982
Lera.S	"Synopsises Of Recent Published Studies Of The Design Process"	Design Studies Vol.4 No.2 Apr 1983
Levine.R	"Supercomputers"	Scientific American Jan 1982
Lewell.J	"Pioneer Of Graphics"	CADCAM International Dec 1982
Little.S	"The Organisational Implications Of CAAD"	CAD84 Proceedings Butterworths Apr 1984

<u>Author</u>	<u>Title</u>	<u>Source</u>
Luria.A.R	"The Working Brain"	Penguin 1973
Mallen.G	"The Analysis Of Architectural Design Activity In The Working Environment"	Report No.108/3 RCA Dec 1973
Martin.D	"Database Design And Implementation"	Van Nostrand Reinhold 1980
Massey.H	"The New Age In Physics"	Elek Books 1966
Maver.T.W	"A Theory Of Architectural Design In Which The Role Of The Computer Is Identified"	Building Science Vol.4 1970
Maver.T.M	"The Concept Of Modelling In Architectural Design"	Design Methods And Theories Vol.13 No.3/4 1979
Miller.D ed.	"A Pocket Popper"	Fontana 1983
Mitchel.R	"Computer Aided Architectural Design"	Petrocelli/Charter 1977
Nash.J.H	"Graphic Interaction With Database Systems"	CAD82 Proceedings Butterworths Apr 1982
Negroponte.N	"The Architecture Machine"	Computer Aided Design Vol.7 No.3 Jul 1975
Negroponte.N	"On Being Creative With Computer Aided Design"	IFIP Conference Proceedings North Holland 1977
Negroponte.N	"A Computational Paradigm For Personalised Searching"	MAGIC Reference Manual, MIT 1976
Negroponte.N	"Computer Aids To Design And Architecture"	Petrocelli/Charter 1975
Negroponte.N	"Recent Advances In Sketch Recognition"	IFIP Conference Proceedings North Holland 1973
Newman.W, and Sproull.R	"Principles Of Interactive Graphics"	McGraw Hill 1973
Ord-Smith.R.J	"Computer Simulation Of Continuous Systems"	Cambridge Computer Science Texts No.3 1975
Papanek.V	"Design For The Real World"	Granada 1974
Pereira.F	"SeeLog - A Prolog Graphics Interface"	EdCAAD, Univ.Of Edinburgh Aug 1982
Pereira.F	"C-Prolog User's Manual. Version 1.3"	EdCAAD, Univ.Of Edinburgh Aug 1983

<u>Author</u>	<u>Title</u>	<u>Source</u>
Pirsig.R	"Zen And The Art Of Motorcycle Maintenance"	Bodley Head 1974
Popper.K	"Conjectures And Refutations"	Routledge and Keegan Paul 1963
Purcell.P	"The Data Station: An Integrated Graphic Information Facility"	PARC83 Proceedings On-Line Conferences Oct 1983
Quinlan.K.M, and Woodwark.J.R	"A Spatially Segmented Solids Database - Justification And Design"	CAD82 Proceedings Butterworths Apr 1982
RIBA	"Directory Of Practices 1983"	RIBA Publications 1983
RIBA	"Working With Your Architect"	RIBA Publications 1971
RIBA	"Architects Appointment"	RIBA Publications Jun 1982
Radford.A.S	"Computer Programming In FORTRAN"	Teach Yourself Books 1975
Riley.J.P and Lawson.B.R	"RODIN: A System For Modelling Three Dimensional Roof Forms"	CAD82 Proceedings Butterworths Apr 1982
Rorvik.D	"As Man Becomes Machine"	Abacus 1975
Russell.B	"The Impact Of Science On Society"	Unwin 1968
Scrivener.S	"How To Handle An Image"	Design Studies Vol.4 No.1 Jan 1983
Strothotte.T and Funt.B	"Raster Display Of A Rotating Object Using Parallel Processing"	Computer Graphics Forum Vol.2 No.4 Nov 1983
Sutherland.I	"SKETCHPAD: A Man-Machine Graphical Communication System"	MIT 1963
Swinson.P	"Prescriptive To Descriptive Programming"	EdCAAD, Univ.Of Edinburgh Jul 1980
Tucker.A	"The Patterns Of Power"	The Guardian Thur 24th Sept 1981
Tuckey.C.O	"Coordinate Geometry"	Longmans 1953
Van den Bos.J	"Introduction To Man-Computer Communication"	Computer Graphics Forum Vol.1 No.7 Dec 1982
Walters.R.J	"Towards An End User View Of Design Systems"	PARC83 Proceedings Oct 1983
Weeramantry.C.G	"The Slumbering Sentinels"	Penguin 1983

<u>Author</u>	<u>Title</u>	<u>Source</u>
Willey.D.S	"Approaches To Computer Aided Sketch Design"	Computer Aided Design Vol.8 No.3 Jul 1976
Wilson.I.R and Addyman.A.M	"A Practical Introduction To PASCAL"	Macmillan 1978
Zukav.G	"The Dancing Wu Li Masters"	Fontana 1979

Appendix 1

Publications

1	'Man Before The Machine: An Issue In CAAD'	216
2	with B.G.Hammond 'A Graphics Interface To Complement Traditional Techniques'	230

The author is a Registered Architect having been employed in private practice in London. His introduction to computers came through thermal modelling whilst a student at the Welsh School Of Architecture, and he is currently undertaking an SERC funded Research Degree at the Scott Sutherland School Of Architecture in Aberdeen under the direction of Dr L.W.W.Laing, looking at aspects of data-inputting for CAAD systems.

## Man Before The Machine : An Issue In CAAD

" The test of the machine is the satisfaction it gives you. There isn't any other test. If the machine produces tranquility it's right. If it disturbs you it's wrong until either the machine or your mind is changed. "

R.Pirsig

Zen And The Art Of Motorcycle Maintenance.

### INTRODUCTION

In-advance of any other life form man is able to predict his<sub>1</sub> future circumstances and take action to mitigate future contingencies. These endowments have enabled him to develop tools, to modify his environment through the building of habitations and to develop technologies. The ability involved in these endeavours is mans ability to design.

The design of cities, buildings or infact "any edifice on a scale large enough to house human activity"<sub>2</sub> goes under the generic term of 'Architecture'. That same creative force that gave rise to architecture also gave rise to the design and construction of the digital electronic computer.

Although applications can be found for computer aids in the practice of architecture as a business process<sub>3</sub> "computing is less in evidence in the work of architects than in any other sector of the (building) industry. There could be many reasons for this, including cost, lack of suitable input and output devices and a feeling that computers can only deal with a small part of the architects workload"<sub>4</sub>

### Aspiration : The Days Of Optimism

When the pioneering CAAD work was undertaken it was generally imagined that the end product would be a system where the architect would specify the problem to the computer and then select his preferred solution from a number automatically generated by the machine ; solutions incorporating his specialists' advice and the automatic 'optimisation' of the various contradictory design parameters.

This optimism was in part a consequence of the enthusiasm for



Operations Research methods ( developed to compensate for manpower shortages during the early 1940's ) which were subsequently applied to the 'design process'. With the benefit of hindsight design methodologists are generally agreed that architectural design is too complex to be described by a process having a 'beginning' and an 'end'<sup>5,6</sup>; that it can be infinitely recursive ; and that a learning process is initiated which increases the designers understanding of the problem as the process continues.<sup>6,7</sup>

#### Actuality : Design As A Continuum As Opposed To A Sequence

However it is phrased, the goal of architectural endeavour is the conception and communication of a hypothesised building model. The strategy invoked, on a simplistic level, is the recursive cycle of 'analysis-synthesis-appraisal',<sup>8</sup> proceeding from the general to the specific as described by Markus . This model is compatible with the theory of the development of scientific knowledge by the method of conjecture and refutation described by Popper . This leads to a helical model of the design process summarised in Fig 1.

The process then, may be seen as a developing description of the evolving building model which may, at any point in time, include incomplete and inconsistent data. The goal of the process is to arrive at a complete and thorough data set pertaining to the hypothetical building<sup>10</sup> from which it can ultimately be constructed. The self evidence of this is noteworthy since the process that is invoked is nothing less than the creation of a data base by the utilisation of parts of itself to test, regulate and expand and modify itself : It is thereby a system in which each data set is implicitly related to each and every other, and although there are many ways of crossreferencing the data, the common feature interrelating it all is the building geometry to which the data applies. It is this feature which pervades the entire data-base and explains why architects manipulate their ideas by means of graphic representations. Significantly it is this feature which currently has the weakest input/output support in CAAD systems.

#### Communication : Talking Down To Machines

A computer may be described as a high speed moron ; and perhaps it is not so much the lack of variety of building geometry input/output devices which deter architects from CAAD systems. but the crudity and banality of the level at which the man-machine interaction has to take place. This 'blandness' is manifest in those areas of architecture where consideration of qualitative aspects - which indeed many practitioners feel are the essence of architecture - which require channels of communication, languages and interfaces currently unavailable to computers<sup>5</sup>. Moreover, where machines are programmed to handle 'qualities', we find that they are actually transposed to quantities, ie. coefficients of 'goodness' arrived at by subjective assessment. Under these circumstances it is often hard to see whether a computer aid might not in fact, be an obfuscation.

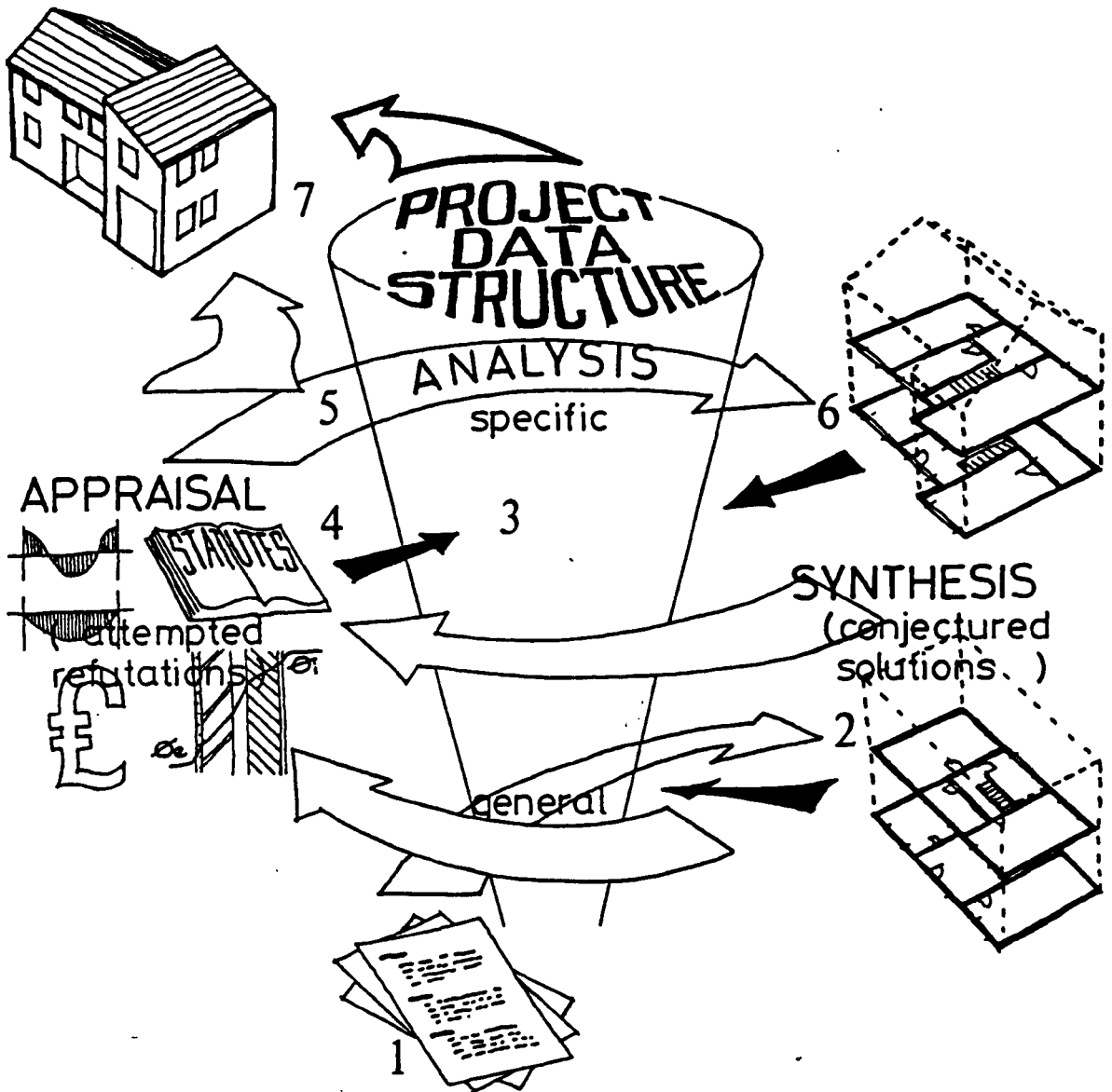


Diagram 1. A Model Of The Design Process

An initial analysis of the design problem results in a Design Brief (1). A hypothetical 'building' is conceived by the architect (2) which 'spins off' information to the Project Data Structure (3). This hypothesised solution is then appraised in part or full as a solution to the brief (4). Again this adds to the data structure. Analysis (5) of the appraisal leads to the synthesis of new or modified solutions (6). This cycle of analysis - synthesis - appraisal is continued until the appraisal no longer 'refutes' the 'conjectured' design solution and the Project Data Structure is sufficiently detailed to communicate the design in full. (7)

A significant feature of this model is that the process is seen to proceed from general to specific problems and statements.

## Evolution : Traditional Methods

Implicitly then, the central issue for CAAD systems is that it is necessary "to describe buildings to computers...before architects ...can use computers to perform any task related to the design and production of buildings"<sup>1</sup> and to do so through an interface that is attractive to the user. 'Pencil and paper' techniques evolved to cope with the subtle complexities of the design problem. They are tools for the manipulation of spatial, proportional and relational ideas. Pencil and paper are regarded as the vehicle for allowing the designer a wide range of possible approaches to the problem by presenting simultaneously a range of visual cues to complex relationships<sup>2</sup>; where simple symbols possess many levels of significant information; where they encode a building form into manageable pieces of information<sup>3</sup>; and where sketchiness of a drawing contains important information about the designers train of thought<sup>5</sup>. Significantly "such a drawing cannot be easily decoded by anyone other than the designer"<sup>12</sup>

## Specification : A Desirable Interface

The development of CAAD systems, if they are to become less incongruous to their potential users, must be endowed with qualities that enable them to assist the designer to find entry points to the design problem; assist him in generating possible design solutions from which he can select; and to assist him to manipulate building form. The attributes such a system should possess include the ability to focus on form; the ability to allow the designer to select areas to examine and proceed with operations in the order he chooses; the capacity to deal with a large number of variables; the ability to keep the designer in close contact with the problem solving process; and finally it should compliment the strengths of the designer<sup>13</sup>.

A machine must therefore possess a high degree of intelligence and sophistication. Only by doing so can a computer system<sup>12</sup> offer the designer the freedom enjoyed using traditional techniques<sup>2</sup>.

## Peroration : Technological Indigestion

It is not clear just how long man has been designing buildings and monuments, but during this time it is evident that those design techniques that evolved were a direct consequence of both the knowledge and the scale of human abilities prevalent at the time the designers lived and worked.

In contrast the development of the digital computer and its associated applications technology has been so rapid that in only a few fields of human endeavour has a body of user experience been accumulated. The impact of this technology and its associated learning curve has been too abrupt to have been comfortably assimilated<sup>7</sup>; Judging by the antagonisms generated by the impact of CAAD into the architectural profession, for example, there is good reason to believe that the professions metabolism is insufficiently developed to do the job and until it is we are suffering from technological indigestion<sup>14, 15, 16</sup>.

Arguably, the rate of hardware development has been so rapid that it will take a long time for software applications to catch up with the hardware's potential abilities, and a longer time still before sufficient user experience exists for the profession to undertake a

critical qualitative assessment of CAAD benefits. In this context we must think in terms of decades pending a software revolution.

Perhaps the most unfortunate legacy of the computers origins in the fields of scientific application has been the emphasis on numeracy in the design process - strengthened perhaps in the building industry by the typically 'numeric' input from the architects special consultants. This has tended to obscure the fact that the architects final design solution is itself an optimisation of both qualitative and quantitative parameters. The emphasis on 'quality,' argueably differentiates 'Architecture' from 'mere engineering'<sup>2</sup>.

#### Summation : The Last Word

The controversy within the architectural profession has moved on from whether computers have anything to offer them : Computers are here and there is no precedent to suggest that a technology once invented can be uninvented.

The issue upon which architects now have to set their minds is whether to reformulate design problems and procedures to make the most cost effective use of those computeraids which exist ; or to exert pressure to develop computer equipment more sympathetic to the existing design process and so preserve for the designer the pleasure of design.

Dave Leifer

Aberdeen

Feb 1983

Validation : References

- 1 Eastman C.M Carnegie-Mellon University Pittsburgh U.S.A.  
 " Recent Developments In Representation In The Science Of Design "  
 Design Studies Vol 3 No 1 Jan 82
- 2 Pevsner N  
 " Outline Of European Architecture "  
 Penguin 1950
- 3 Davison J.A GMWComputers Berkhamsted  
 " Computer Aided Draughting : A Means To An End "  
 Paper presented to The Institute Of Advanced Architectural Studies  
 York. Nov 1981
- 4 Standing Committee On Computing And Data coordination  
 " Computing And Communication In The Building Industry "  
 Department Of The Environment Mar 1979
- 5 Negroponte N Massachusetts Institute Of Technology U.S.A.  
 " The Architecture Machine "  
 Computer Aided Design Vol 7 No 3 Jul 1975
- 6 Findler N.V State University Of NewYork U.S.A.  
 " Analogical Reasoning In Design Process "  
 Design Studies Vol 2 No 1 Jan 1981
- 7 Bijl A , Stone D , Rosenthal D EdCAAD Edinburgh University  
 " Integrated CAAD Systems "  
 Final Report To DoE On Funded Research Project DRG 470/12 Mar 79
- 8 Markus T.A ABACUS University Of Strathclyde  
 " The Role Of Building Performance Measurement And Appraisal In  
 Design Methods "  
 Architects Journal Vol 146 No 25 1967
- 9 Popper K  
 "Conjectures And Refutations : The Growth Of Scientific Knowledge"  
 Routledge And Keegan Paul 1963
- 10 Phillips R.J , Beaumont M.J , Richardson D , Bartley J  
 " Geometry For CAAD "  
 Computer Aided Design Vol 13 No 2 Mar 1981
- 11 Bijl A et al. EdCAAD University Of Edinburgh  
 Building Description Techniques To Support Computer-Aided Design  
 And Production "  
 Section25 of SERC Grant Application Dec 1981
- 12 Willey D.S University Of Liverpool  
 " Approaches To Computer-Aided Architectural Sketch Design  
 Computer Aided Design Vol 8 No 3 Jul 1976



The authors are currently engaged in SERC funded research degrees at the Scott Sutherland School Of Architecture in Aberdeen under the direction of Dr L.W.W.Laing. They are currently examining aspects of computer graphics with particular emphasis on the man-machine interface. Both authors have experience of working in private practice in London, and one of whom was last employed in the capacity of Job Architect.

## A Graphics Interface to complement Traditional Techniques

### ABSTRACT

Noting the reluctance of architects in small private practices to adopt CAAD aids, the crudity of existing graphic interfaces is identified as an inhibiting factor.

A suite of computer programmes currently under development are described which are designed to permit the input of geometric plan forms by traditional pencil and paper techniques, whilst utilising the computers processing power to edit and manipulate the data so 'captured'.

### INTRODUCTION

Many reasons have been offered to explain why architects have failed to utilise computers more fully than they have done<sup>1</sup>. Despite the promise of the micro-chip revolution, most architects' design work is still carried out by manual methods; computer aids being viewed as a specialised adjunct applicable to a few atypical projects. This is perhaps understandable in a profession where some 85% of all registered architects work in practices employing 10 or less architectural staff<sup>2</sup>, and where the stability of workload is sufficiently uncertain to make large capital investment precarious<sup>3</sup>.

To most architectural practitioners who are accustomed to, and moreover enjoy using, drawing board and set-square, computers represent an alien technology requiring unfamiliar and sometimes inappropriate languages and working methods. The relative magnitude of the capital investment that computers represent to the small architectural practice requires extremely efficient and close management of the system to ensure that it is run cost-effectively. Not only is this expertise expensive, but it is at present rarely available. More insidiously, principals in small practices may feel a potential loss of overall

control posed by the inherent complexity of such systems. Such fears cannot be easily allayed.

It has been argued elsewhere that one of the most positive ways with which to promote CAAD would be to utilise the computers processing power to take on more of the burden of man-machine communication<sup>4</sup>. The machines must become more approachable by the non-computer literate user, leaving him free to devote his energies to designing rather than on communicating with, and operating the system.

### THE ARCHITECTS ROLE

The architects task may be construed as the conception and communication of a hypothetical building model. Infact, the design process is one in which the designer compiles ever increasingly accurate data pertaining to the hypothetical model until such time as it is sufficiently complete and consistent to allow the client to 'experience' it and the builder to build it. ( This pragmatic description does not belittle the implicit importance attached to the role of the architect as an aesthete ). The common feature underlying all of the various architect generated data sets is the building models geometry; whilst the material specifications state 'what' it is and the performance specifications state what 'it' is to do, the drawings state where all of this other information is to apply, and how it is interrelated.

The corollary to this process is constant appraisal to ensure that the sub-systems do what they are supposed to, be it the adequacy of the structural system or that the project may be built within budget.

### DATA MANIPULATION

The magnitude and complexity of the data generated for any one building project, not least of which is the project drawings, can only be 'guesstimated' in advance. This puts extraordinary demands on any computer system which would handle such a large and various data-set. Although some data-basing systems have been evolved and utilised by the profession<sup>5</sup>, none have been sufficiently effective for machine implementation. Perhaps one of the most significant developments in data-manipulating techniques has been the development of logical programming languages such as PROLOG<sup>6</sup> which offer a powerful means for interrogating large data-structures. The application of these languages to graphic data is currently under investigation elsewhere<sup>7</sup>.

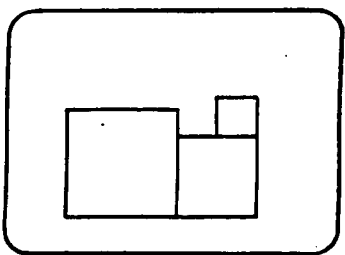
### COMPUTER GRAPHICS

Within the overall CAAD context, one area of concern is the machines data-aquisition rather than its data-manipulation. This is particularly relevent to drawn information, since it is at this level that architects 'experience' computer systems. Current systems require inhibiting draughting conventions and crude levels of communication.

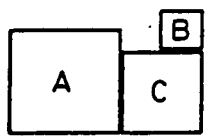
Drawings represent simultaneously many different levels of signif-



cance to the the user<sup>8</sup>. Each level is 'distilled' from it's contextual relationship to other parts of the drawing. Thus the fundamental problem with computer graphics is the radical difference between mans and machines perception of drawings. Whilst the human can relate parts of a 'picture' to the whole, the machine is simply a collection of registers which can only cope with one data item at a time. The only 'context' the machine has for construing a drawing is the artificial one implicit in the structure of the data-base. To illustrate this, consider the example below.



a) Picture to be created



b) Association of Graphic Primitives



c) Primitives

Primitive	A	B	C
scale factor	$S_a$	$S_b$	$S_c$
reference point x	$x_a$	$x_b$	$x_c$
y	$y_a$	$y_b$	$y_c$

d) Input Data

move pen to	$(x_a, y_a)$	$(x_b, y_b)$	$(x_c, y_c)$
draw line to	$(x_a + S_a, y_a)$	$(x_b + S_b, y_b)$	$(x_c + S_c, y_c)$
draw line to	$(x_a + S_a, y_a + S_a)$	$(x_b + S_b, y_b + S_b)$	$(x_c + S_c, y_c + S_c)$
draw line to	$(x_a, y_a + S_a)$	$(x_b, y_b + S_b)$	$(x_c, y_c + S_c)$
draw line to	$(x_a, y_a)$	$(x_b, y_b)$	$(x_c, y_c)$

e) Machine Draughting Commands

Diagram 1 Example Of A Data Structure

Consider a simple CAAD graphics input system where drawings are created as combinations of squares. The user wishes to input a picture shown in 1a. The picture comprises of three squares A,B and C. Ignoring the interrelationship of the squares with each other, the machine must be given sufficient information about each geometric primitive not only to carry out the calculations that might be required in the subsequent applications programme, but simply to draw them on the screen. In the case illustrated the necessary input data includes the coordinate position of a point of reference for each square ( here taken as the lower left-hand corner ), and a scaling factor for each square. ( This input data is shown in the table, Diagram 1d.). In order to draw these squares, the machine must convert this input data such that the drawing routines may be enacted. ( These drawing routines are shown in Diagram 1e.)

It will be seen that in such a system it is necessary to refer to the lower levels of the data-structure if the user wants to address any particular line. This level is of course different to the 'vocabulary' used to create the picture in the first place. This cumbersome method of dealing with drawings does not bear comparison to the ease of paper and pencil techniques.

THE PROPOSED GRAPHICS INTERFACE

To overcome the disincentive outlined above, an opposite route may be considered. Accepting the premise that architects generally design in the initial stages by 'toying' with freehand sketches, it is legitimate to place the onus on the computer to derive higher levels of significance ( ie. recognition of graphic primitives ), from the lowest level of input data ( ie. the continuous digitisation of architects sketches ); in effect to deduce the data-structure from the act of drawing.

The system being developed is summarised in Diagram 2. Architectural sketches are digitised on a graphics tablet (Tektronix 4954 ) which is connected via a Tektronix 4010 interface to a micro-computer with graphics display screen (Tektronix 4054).

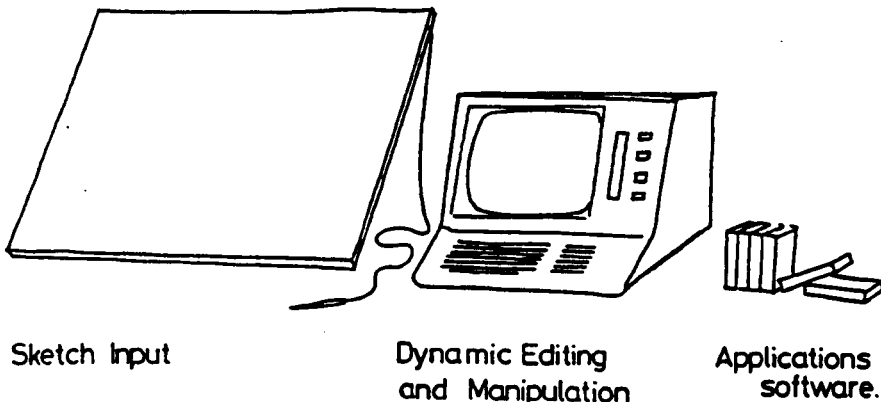


Diagram 2 Configuration Of Proposed Interface

SYSTEM DESCRIPTION

The proposed system operates in five sections.

1. As described the designer sketches his intentions on the digitising tablet, an example of which is shown in Diagram 3. The rate of data capture by the machine will depend upon the speed at which the user draws and the cycle time of the digitiser.

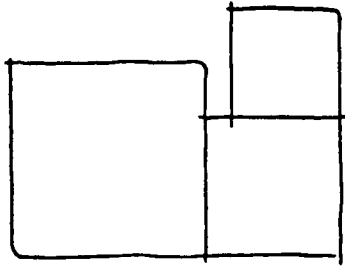


Diagram 3 Sketch As Drawn On Tablet

The drawing is echoed on the screen of the graphics terminal to confirm to the user that the drawing has been captured. The echo on the screen resulting from the sketch shown in Diagram 3, and the format that the digitised data takes is shown in Diagram 4.

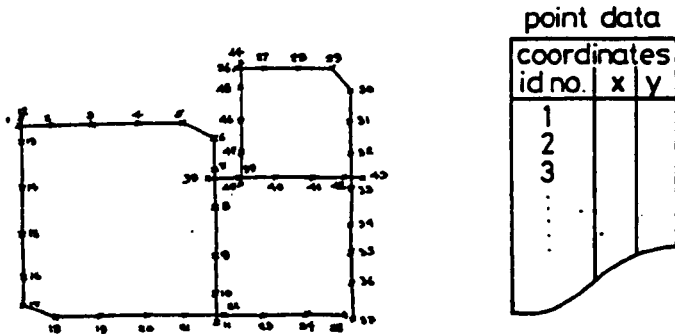


Diagram 4 Digitised Information Echoed On Screen

2. The digitised tablet data is processed, and the data is sorted into discrete lines by comparing the gradient of each successive line element to the line of closest fit through the preceding data points. If the deviation is greater than a user defined tolerance a new line is deemed to have been encountered.

Moreover the 'image' is enhanced: Not only are freehand lines straightened, but overlapping lines are removed, clipped corners reconstructed, and almost touching lines made to touch. The information about the lines composing the drawing are stored in an array for further processing, and the enhanced image displayed on the screen as shown in Diagram 5.

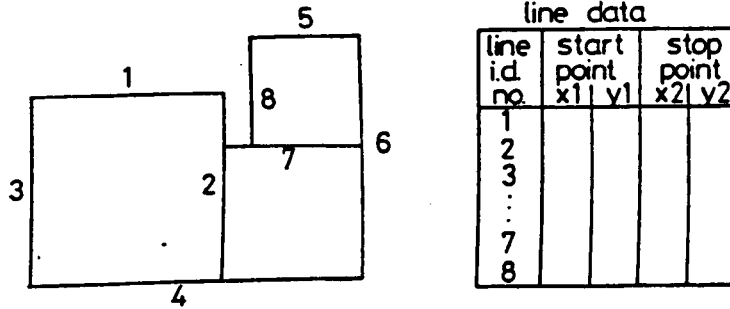


Diagram 5 Computer Interpretation Of Lines

3. The user may wish to edit the interpretation the computer has generated. Using the dynamic graphic facilities of the terminal, he may delete or add lines by means of the cross-hair cursor, or indeed add information via the tablet.

Manipulation via the terminal has the advantage that the user can, if he wishes, make the drawing orthogonal, or place the lines in a range of sectors (ie. 0,15,30,45,60,75 or 90 degrees). With this editing process via the terminal, the dynamic feedback technique mentioned earlier is available to ensure the accurate placing of lines on the screen.

4. The line data generated at the completion of the above editing process is then dissasociated into discrete single line sections. Thus a line disected by another is split into two individual lines. The result of this dissasociation is shown in Diagram 6.

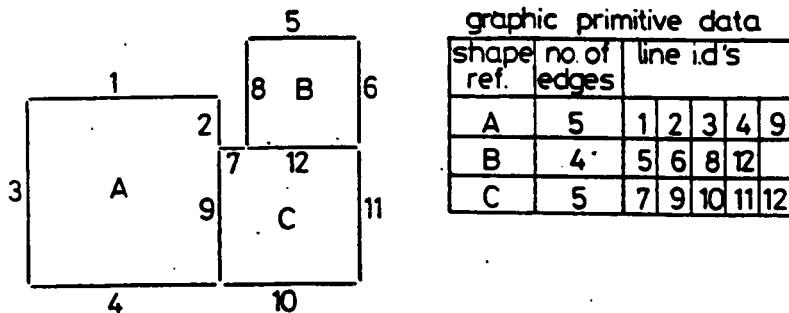


Diagram 6 Line Disassociation And Primitive Identification

By evoking a search algorithm, the perimeter of each enclosed space is traced and the line identifiers for the boundaries stored in an array. The system has thus abstracted geometric primitives from the data in a form which may be used directly by the advanced graphics capabilities of the graphics terminal.

Utilising the terminals capabilities, the user can manipulate complete primitives. He may repeat, move, mirror, rotate, and scale. An indication of these facilities is demonstrated in Diagram 7.

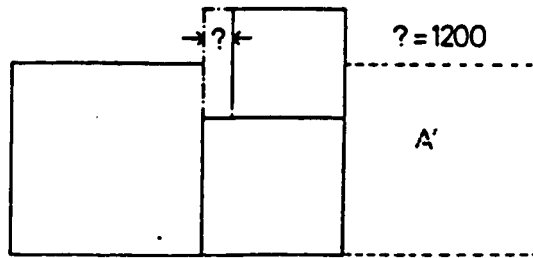


Diagram 7 Graphic Manipulation Option Utilising Dynamic Graphics Offering Instant Feedback And Rubber Banding

5. The final step for the system is to reformat the graphic data into structures suitable for input into other applications programs. Since most applications software deal with polyhedral geometries the data-structure produced by the above system contains the necessary information for transmutation to other forms.

#### SUMMARY

The development project described above is intended to ease the communication of drawings between the architectural user and the machine in the following ways :

1. By placing the onus of interpretation on the machine, the designer can devote his time more fully to the task of designing.
2. By automating the interpretation system the machine is made accessible to the non-computer literate user.
3. Such a system supplements the existing drawing board techniques generally used by the architectural profession at present, and can make the new technology less obtrusive.
4. Such a system frees the designer from the constraints of draughting conventions common with many existing graphics handling applications programmes.
5. Appraisal programs may be instigated much earlier in the design process, which can maximise the benefit of the advice thus rendered. Moreover there is less overhead involved in the time consuming task of 'digitising' drawings done in advance by manual methods.

#### REFERENCES

1. Standing Committee On Computing And Data Coordination  
"Computing And Communication In The Building Industry"  
D.o.E March 1979
2. CICA  
"Fact Sheet On The Use Of Computers In The Construction Industry"  
CICA Bulletin Mar. 1982

3. Bijl.A, Stone.D, Rosenthal.D EdCAAD, University Of Edinburgh  
 "Integrated CAAD Systems"  
 Final Report To D.o.E On Funded Research Project DRG 470/12 Mar.79
4. Leifer.D Scott Sutherland School Of Architecture, Aberdeen  
 "Man Before The Machine : An Issue In CAAD"  
 BoCAAD University Of Strathclyde, Glasgow Vol 43-46 Feb 1983
5. Ray-Jones.A.R, Clegg.D Sfb Agency  
 "CI/Sfb Construction Indexing Manual"  
 RIBA Publications 1976
6. Clocksin.W.F., Mellish.C.S.  
 "Programming In PROLOG"  
 Springer-Verlag 1981
7. Work currently being undertaken at EdCAAD, Edinburgh University.
8. Willey.D.S University Of Liverpool  
 "Approaches To Computer-Aided Architectural Sketch Design"  
 Computer Aided Design Vol 8 No.3 Jul 1976

Brian Hammond B.Sc  
 Dave Leifer B.Sc, B.Arch

Apl 83.

Appendix 2

Line Of Least Squares Test

## APPENDIX 2

It was noted in section 6, page 219, that the least squares line describing the line of closest fit through a spread of data coordinate points, is capable of producing misleading results.

" The least square line approximating the set of points  $(X_1, Y_1), (X_2, Y_2), \dots, (X_N, Y_N)$  has the equation

$$Y = a_0 + a_1 X$$

where the constants  $a_0$  and  $a_1$  are determined by solving simultaneously the equations

$$\begin{aligned} Y &= a_0 N + a_1 X \\ XY &= a_0 X + a_1 X^2 \end{aligned}$$

which are called the 'normal equations for the least square line'.

The constants  $a_0$  and  $a_1$  can, if desired, be found from the formulas

$$a_0 = \frac{(Y)(X^2) - (X)(XY)}{N X^2 - (X)^2}, \text{ and}$$

$$a_1 = \frac{N XY - (X)(Y)}{N X^2 - (X)^2} \quad "$$

"Schaums Outline Series: Theory And Problems  
Of Statistics."  
Murray.R.Spiegel  
McGraw Hill 1961 p.220

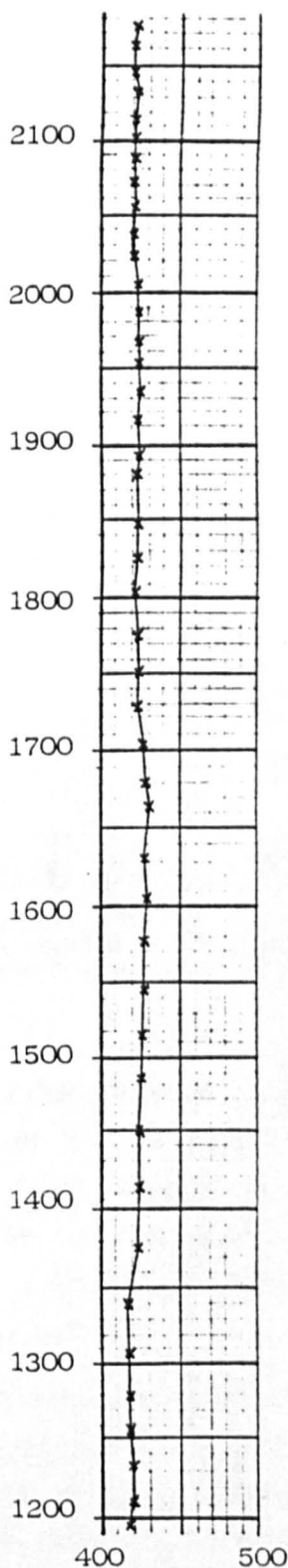
The input and output data which was used and produced by the line of least squares program, (the listing of which is presented on page 135), clearly shows that the gradient of the closest line wavers, when as in this example, the data points indicate a line close to the vertical. Suspecting that the algorithm was at fault, The first seven data points were fed into a regression analysis available on the Stat Pack computer programme run on the DEC 20 mainframe computer. This process confirmed the results generated by the DEMO program. The results of the STATPK computer run is



INPUT DATA

Coordinate Values		
29	424	2177
26	422	2173
26	421	2161
26	422	2147
26	423	2133
26	423	2116
26	422	2103
26	422	2088
26	422	2072
26	422	2055
26	422	2039
26	423	2023
26	425	2005
26	426	1988
26	425	1969
26	426	1952
26	424	1935
26	423	1914
26	424	1893
26	423	1870
26	422	1848
26	422	1826
26	422	1802
26	423	1776
26	423	1750
26	424	1729
26	428	1705
26	429	1680
26	430	1657
26	428	1631
26	429	1606
26	428	1577
26	428	1546
26	426	1517
26	426	1484
26	424	1449
26	424	1413
26	422	1377
26	419	1340
26	418	1306
26	418	1279
26	419	1254
26	420	1232
26	420	1212
26	418	1197

GRAPHICAL REPRESENTATION



OUTPUT DATA

@EXE DEMO.REL  
LINK:Loading  
(LNKXCT DEMO execution)

x	y	gradient in °
422	2173	63.43495
421	2161	78.36637
422	2147	80.53768
423	2133	61.55707
423	2116	-64.44004
422	2103	57.99462
422	2088	79.00914
422	2072	83.41806
422	2055	85.31911
422	2039	86.36367
423	2023	79.50852
425	2005	-86.48982
426	1988	-87.56896
425	1969	-87.99547
426	1952	-88.20434
424	1935	-88.32988
423	1914	-88.29580
424	1893	-88.42555
423	1870	-88.38298
422	1848	-88.01156
422	1826	-87.46401
422	1802	-86.55998
423	1776	-86.51142
423	1750	-86.46065
424	1729	-87.32326
428	1705	-88.29218
429	1680	-88.53589
430	1657	-88.61378
428	1631	-88.72911
429	1606	-88.80231
428	1577	-88.87598
428	1546	-88.93996
426	1517	-88.98362
426	1484	-89.02479
424	1449	-89.01319
424	1413	-89.00104
422	1377	-88.88032
419	1340	-88.38110
418	1306	-87.13562
418	1279	-82.94859
419	1254	-1.32234
420	1232	79.95088
420	1212	84.73996
418	1197	86.77346

STOP  
END OF EXECUTION  
CPU TIME: 0.48

Figure 1 DEMO Program Input And Output

shown in Figure 2 on overpage.

In the course of further investigation, the data was fed in to the STATPK program in reverse order, which produced identical results as shown in Figure 3 on page 243. Omitting the first reading, however, produces a markedly different result, as shown in Figure 4 on page 244.

Having identified this 'aberant' situation, which would appear to arise due to the initial data values of 'x' and 'y', the area immediately around this first point was 'mapped', ie, lines of least squares were calculated for data sets where the first coordinate point was moved by a value of 1 to its eight surrounding cells. These STATPK calculation runs are shown in Figures 5 to 12 on pages 245 to 252 respectively. The results are summarised in the matrix below.

	423	424	425
2178	<u>-81°27'</u> 84°18' -87°33'	<u>61°57'</u> 86°12' -84°54'	<u>77°36'</u> 86° -79°12'
2177	<u>-81°42'</u> 84°03' -87°33'	<u>58°</u> 86°20' -85°	<u>76°57'</u> 85°57' -79°36'
2176	<u>-81°57'</u> 83°50' -87°36'	<u>53°03'</u> 86° -85°03'	<u>76°09'</u> 85°48' -80°

The figures underlined are the gradients of the lines of least squares, whilst the accompanying two figures show the upper and lower angular limits respectively, for each line of least squares gradient. The 'x' and 'y' coordinate values for the calculations first data point are given along the top, and down the left hand side respectively.

It will be noted that the values calculated under the 423, and 425 columns show a fair degree of stability, whilst those in the central column fluctuate markedly. Also, the sign changes from negative to positive between the 423 and 424 columns.

Throughout, the upper and lower limits within which the line of least squares is 'confidently' predicted to occur remain stable







RUN PUB:STATPK

WHICH COMMAND? DATA

HOW MANY INPUT VARIABLES? 2

ENTER INPUT DATA

423,2178,  
422,2173,  
421,2161,  
422,2147,  
423,2133,  
423,2116,  
422,2103,

WHICH COMMAND? NAME

VAR 1? EX

VAR 2? WY

WHICH COMMAND? REGR

ENTER OPTIONS SEPERATED BY COMMAS RESID

WHICH VARIABLES ARE THE REIDUALS TO BE STORED UNDER? 3

LIST THE INDEPENDENT VARIABLES? EX

WHICH IS THE DEPENDENT VARIABLE? WY

\*\*\*\*\* MULTIPLE LINEAR REGRESSION \*\*\*\*\*

SAMPLE SIZE: 7  
DEPENDENT VARIABLE: WY  
INDEPENDENT VARIABLE: EX  
COEFFICIENT OF DETERMINATION: 0.03123  
MULTIPLE CORRECTION COEFFICIENT: 0.17671  
ESTIMATED CONSTANT TERM: 4959.6666  
STANDARD ERROR OF ESTIMATE: 30.748442

ANALYSIS OF VARIANCE FOR THE REGRESSION:

SOURCE OF VARIATION	DF	S. SQ	M.S.	F	PROB
REGRESSION	1	152.381	152.381	.1612	0.7047
RESIDUALS	5	4727.33	954.467		
TOTAL	6	4879.71			

REGRESSION	S. E. OF	F-VALUE	CORR. COEF.
VAR. COEFFICIENT	REG. COEF.	DF (1, 5)	WITH WY
EX -6.666667	16.61	.1612	0.7047
			-0.1767

Gradient=  $-81^{\circ}27'$

+ve limit=  $+84^{\circ}18'$   
-ve limit=  $-87^{\circ}33'$

Figure 5 First Data Point Moved One Point To The NW







RUN PUB:STATPK

WHICH COMMAND? DATA

HOW MANY INPUT VARIABLES? 2

ENTER INPUT DATA

423,2177,  
422,2173,  
421,2161,  
422,2147,  
423,2133,  
423,2116,  
422,2103,

WHICH COMMAND? NAME

VAR 1? EX

VAR 2? WY

WHICH COMMAND? REGR

ENTER OPTIONS SEPERATED BY COMMAS RESID

WHICH VARIABLES ARE THE REIDUALS TO BE STORED UNDER? 3

LIST THE INDEPENDENT VARIABLES? EX

WHICH IS THE DEPENDENT VARIABLE? WY

\*\*\*\*\* MULTIPLE LINEAR REGRESSION \*\*\*\*\*

SAMPLE SIZE: 7  
DEPENDENT VARIABLE: WY  
INDEPENDENT VARIABLE: EX  
COEFFICIENT OF DETERMINATION: 0.03367  
MULTIPLE CORRECTION COEFFICIENT: 0.18349  
ESTIMATED CONSTANT TERM: 5047.50000  
STANDARD ERROR OF ESTTMATE: 30.50041

ANALYSIS OF VARIANCE FOR THE REGRESSION:

SOURCE OF VARIATION	DF	S. SQ	M.S.	F	PROB
REGRESSION	1	162.054	162.054	.1742	0.6937
RESIDUALS	5	4651.38	930.275		
TOTAL	6	4813.43			

REGRESSION	S. E. OF	F-VALUE	CORR. COEF.
VAR. COEFFICIENT	REG. COEF.	DF (1, 5) PROB	WITH WY
EX -6.875000	16.47	.1742 0.6937	-0.1835

Gradient= -81°42' +ve limit= +84°03'  
-ve limit= -87°33'

Figure 8 First Data Point Moved One Point To The West

RUN PUB:STATPK

WHICH COMMAND? DATA

HOW MANY INPUT VARIABLES? 2

ENTER INPUT DATA

425,2177,  
422,2173,  
421,2161,  
422,2147,  
423,2133,  
423,2116,  
422,2103,

WHICH COMMAND? NAME

VAR 1? EX

VAR 2? WY

WHICH COMMAND? REGR

ENTER OPTIONS SEPERATED BY COMMAS RESID

WHICH VARIABLES ARE THE REIDUALS TO BE STORED UNDER? 3

LIST THE INDEPENDENT VARIABLES? EX

WHICH IS THE DEPENDENT VARIABLE? WY

\*\*\*\*\* MULTIPLE LINEAR REGRESSION \*\*\*\*\*

SAMPLE SIZE: 7  
DEPENDENT VARIABLE: WY  
INDEPENDENT VARIABLE: EX  
COEFFICIENT OF DETERMINATION: 0.03747  
MULTIPLE CORRECTION COEFFICIENT: 0.19357  
ESTIMATED CONSTANT TERM: 323.49998  
STANDARD ERROR OF ESTIMATE: 30.440347

ANALYSIS OF VARIANCE FOR THE REGRESSION:

SOURCE OF VARIATION	DF	S. SQ	M.S.	F	PROB
REGRESSION	1	180.355	180.355	.1946	0.6775
RESIDUALS	5	4633.07	926.615		
TOTAL	6	4813.43			

REGRESSION	S. E. OF	F-VALUE	CORR. COEF.
VAR. COEFFICIENT	REG. COEF.	DF (1, 5)	WITH WY
EX	4.308824	9.767	0.1936

Gradient=  $76^{\circ}57'$       +ve limit=  $+85^{\circ}57'$   
   -ve limit=  $-79^{\circ}36'$

Figure 9 First Data Point Moved One Point To The East



RUN PUB:STATPK

WHICH COMMAND? DATA

HOW MANY INPUT VARIABLES? 2

ENTER INPUT DATA

424,2176,  
422,2173,  
421,2161,  
422,2147,  
423,2133,  
423,2116,  
422,2103,

WHICH COMMAND? NAME

VAR 1? EX

VAR 2? WY

WHICH COMMAND? REGR

ENTER OPTIONS SEPERATED BY COMMAS RESID

WHICH VARIABLES ARE THE REIDUALS TO BE STORED UNDER? 3

LIST THE INDEPENDENT VARIABLES? EX

WHICH IS THE DEPENDENT VARIABLE? WY

\*\*\*\*\* MULTIPLE LINEAR REGRESSION \*\*\*\*\*

SAMPLE SIZE: 7  
DEPENDENT VARIABLE: WY  
INDEPENDENT VARIABLE: EX  
COEFFICIENT OF DETERMINATION: 0.00211  
MULTIPLE CORRECTION COEFFICIENT: 0.04596  
ESTIMATED CONSTANT TERM: 1584.4250  
STANDARD ERROR OF ESTIMATE: 30.785792

ANALYSIS OF VARIANCE FOR THE REGRESSION:

SOURCE OF VARIATION	DF	S. SQ	M.S.	F	PROB
REGRESSION	1	10.0321	10.0321	.1059E-01	0.9221
RESIDUALS	5	4738.82	947.765		
TOTAL	6	4748.86			

REGRESSION	S. E. OF	F-VALUE	CORR. COEF.
VAR. COEFFICIENT	REG. COEF.	DF (1, 5) PROB	WITH WY
EX	1.325000	12.88 .1059E-01 0.9221	0.0460

Gradient= 53°03' +ve limit= +86°  
-ve limit= -85°03'

Figure 11 First Data Point Moved One Point To The South

RUN PUB:STATPK

WHICH COMMAND? DATA

HOW MANY INPUT VARIABLES? 2

ENTER INPUT DATA

425,2176,  
422,2173,  
421,2161,  
422,2147,  
423,2133,  
423,2116,  
422,2103,

WHICH COMMAND? NAME

VAR 1? EX

VAR 2? WY

WHICH COMMAND? REGR

ENTER OPTIONS SEPERATED BY COMMAS RESID

WHICH VARIABLES ARE THE REIDUALS TO BE STORED UNDER? 3

LIST THE INDEPENDENT VARIABLES? EX

WHICH IS THE DEPENDENT VARIABLE? WY

\*\*\*\*\* MULTIPLE LINEAR REGRESSION \*\*\*\*\*

SAMPLE SIZE: 7  
DEPENDENT VARIABLE: WY  
INDEPENDENT VARIABLE: EX  
COEFFICIENT OF DETERMINATION: 0.03370  
MULTIPLE CORRECTION COEFFICIENT: 0.18357  
ESTIMATED CONSTANT TERM: 428.99998  
STANDARD ERROR OF ESTIMATE: 30.294631

ANALYSIS OF VARIANCE FOR THE REGRESSION:

SOURCE OF VARIATION	DF	S. SQ	M.S.	F	PROB
REGRESSION	1	160.034	160.034	.1744	0.6936
RESIDUALS	5	4588.82	917.765		
TOTAL	6	4748.46			

REGRESSION	S. E. OF	F-VALUE	CORR. COEF.		
VAR. COEFFICIENT	REG. COEF.	DF (1, 5)	WITH WY		
EX	4.058824	9.720	.1744	0.6936	0.1836

Gradient=  $76^{\circ}09'$  +ve limit=  $+85^{\circ}48'$   
-ve limit=  $-80^{\circ}$

Figure 12 First Data Point Moved One Point To The SE

although the total range of angle these limits represent is only a few degrees short of the entire compass.

Despite the constancy of the limits or gradient, the simple fact is that the gradient predicted is wrong. This is clearly shown in Figure 13 overpage. The actual gradient is, by visual inspection, greater than even the upper limit of the calculation.

One explanation offered for this phenomenon, is the increasing difficulty of predicting the 'y' coordinates on a near vertical line from the 'x's. This would not appear to accord with the preceding observations, for, as the gradient was increased by moving the 'x' coordinate of the first data point to the left, the predicted gradient became increasingly vertical, contrary to the result to be expected if that explanation were true.

The preceding observations seem to demonstrate that in certain localised spots, the gradient prediction reaches some point of inflexion which causes a disturbance. It remains in the province of the mathematician or statistician to produce an explanation.

The reason why this phenomenon does not cause problems with the ENIGMA line interpretation routine, apart from the rarity of encountering such a disturbance, is the two-part test which has been explained in section 6. To illustrate this in practice, with reference to Figure 1 on page 240, a change of direction is noted only if a) the difference between a line segments gradient and that of the preceding line of least squares is greater than a given value, and b), the line segments gradient must also differ from the gradient of the preceding segment by more than the same value.

Finally, Figure 14 on page 255 shows the actual results of applying the line identification routine to a series of curves.

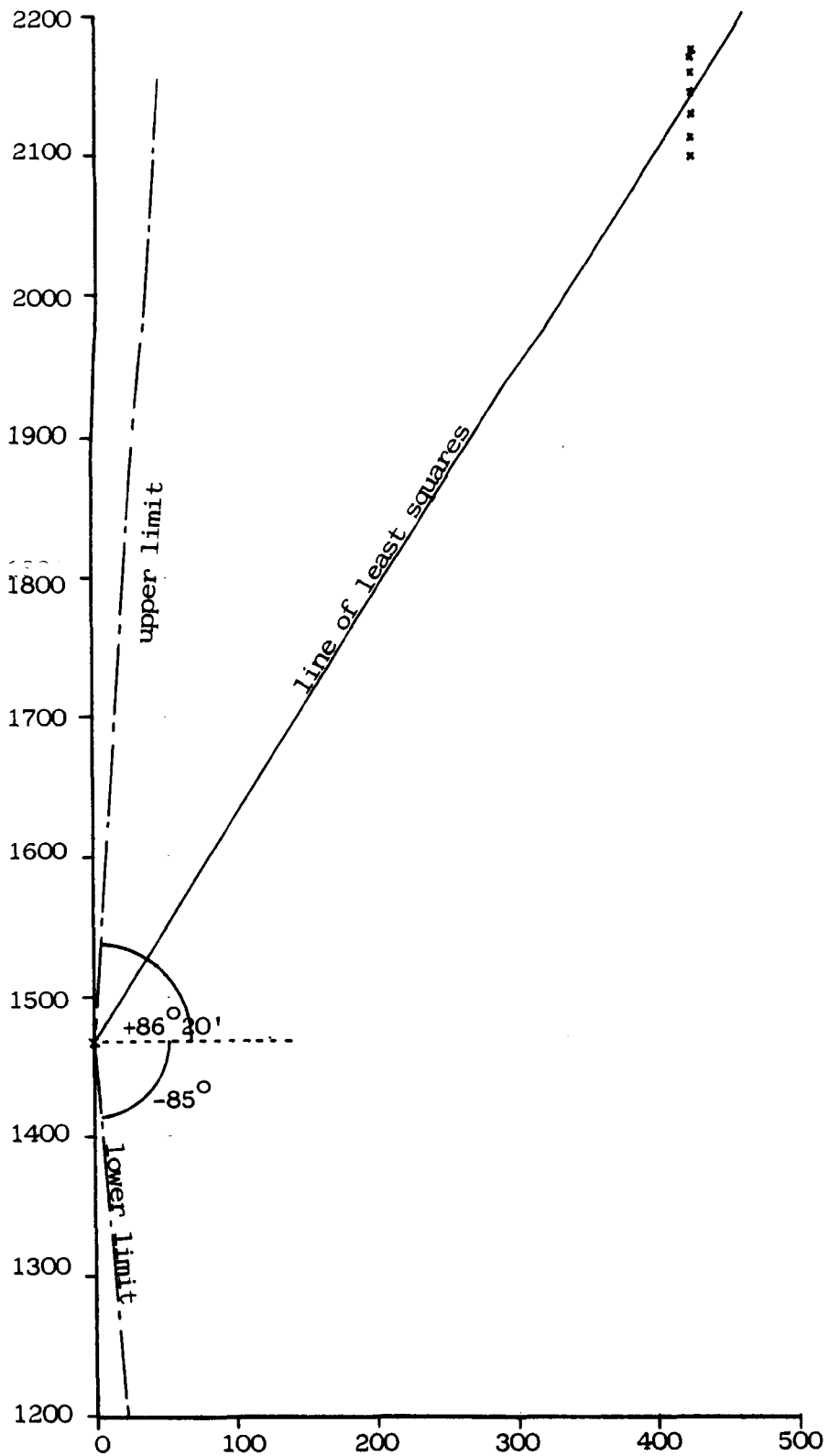


Figure 13 Illustration of the results of the STATPK Line Of Least Squares Calculation listed on Page 242.

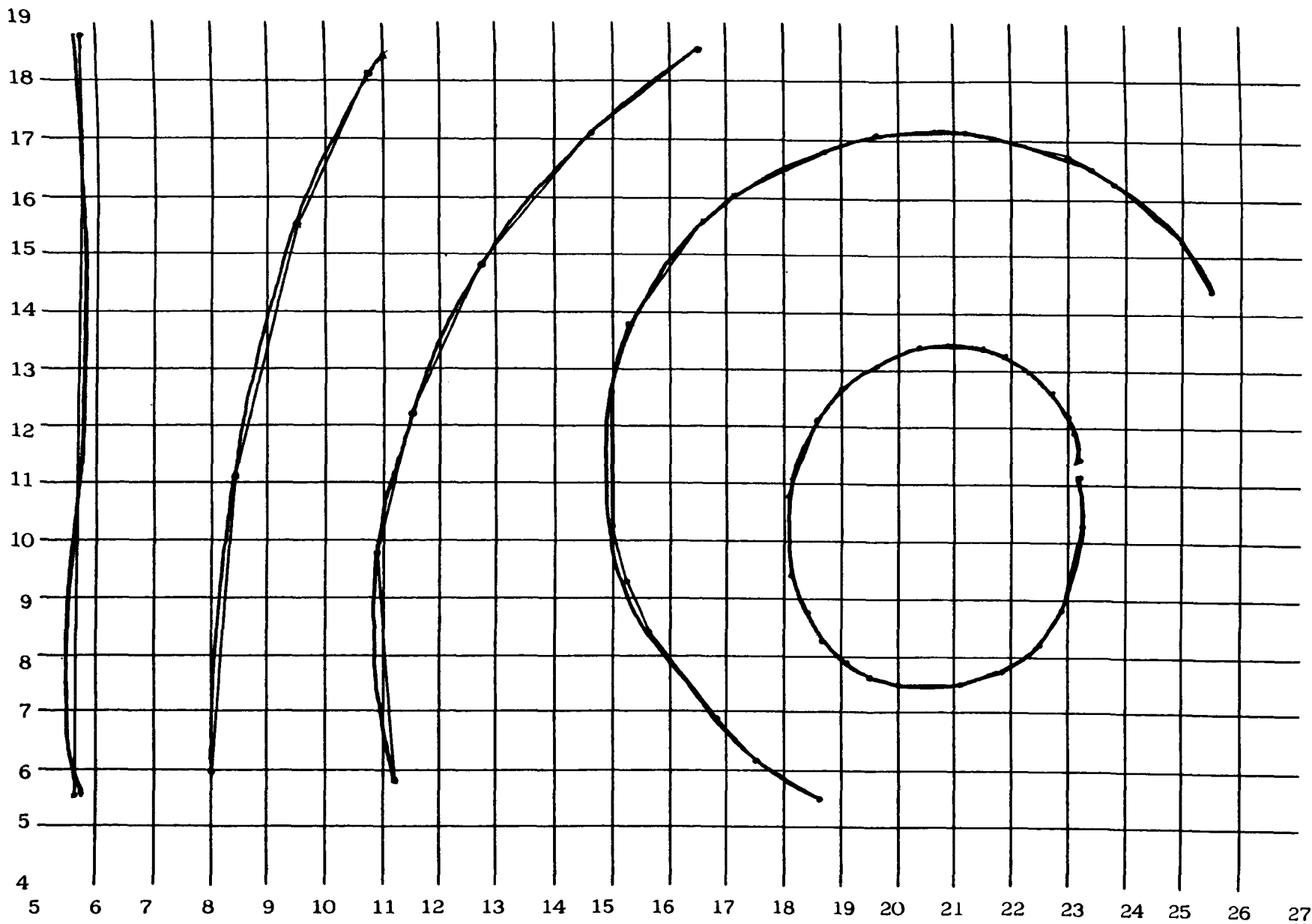


Figure 14 Demonstration Of Sequential Curves Identified By The LINEID Routine.



Appendix 3

ENIGMA User's Manual

NOTE:

Continued development of the software will imply further expansion and alteration of the notes contained within this manual.

Last Revision

July 1984

## APPENDIX A

### ENIGMA User's Manual

May 1984

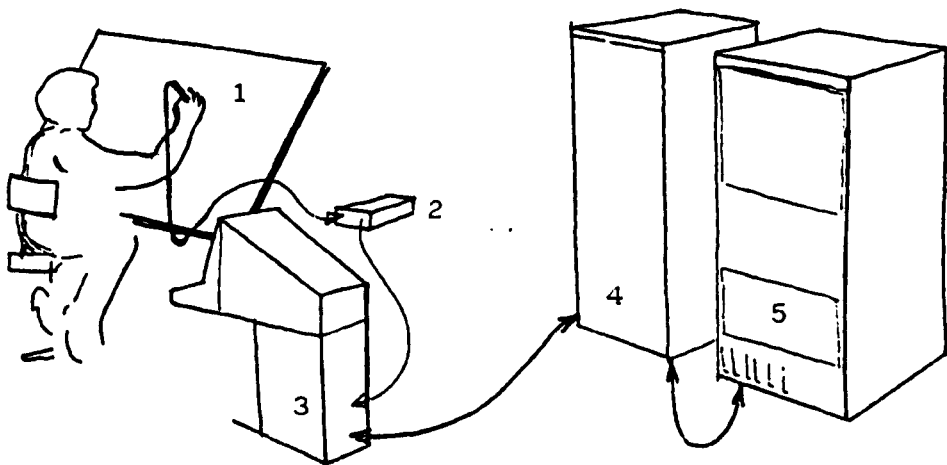
#### 1 Introduction

ENIGMA stands for an ENhanced Interpretive Graphics Module for Architects.

It has been designed to be used for two purposes:

- 1 To serve as a graphics interface for the compilation of geometry data files for input into applications programmes available in the Scott Sutherland School Of Architecture, and
- 2 To permit geometry data files to be created by the interpretation of freehand sketches carried out with an ink pen on a digitising tablet. This system is intended to match the architects' traditional pencil and paper techniques.

#### 2 Hardware Configuration



Key:

- 1 Tektronix 4954 digitising tablet
- 2 pen/tablet link unit
- 3 Tektronix 4010 vector graphics terminal
- 4 Gandalf Modem
- 5 DEC20 mainframe computer processor and secondary data storage.

The ENIGMA suite of programs has been written to run on a DEC20 host computer, using a Tektronix 4010 graphics terminal, and a 4954 A0 size digitising tablet.

The program calls for PLOT 10 graphics library routines, and also requires the Tektronix TEKPAK library to operate the tablet. Additionally, UTIL and TEKLIB routine libraries from the ABACUS Unit, University of Strathclyde are required.

### 3 Software Assumptions

Underlying the design of the ENIGMA software is the assumption that the user wishes to create building geometries by freehand sketches, without much initial idea of the form the building design is likely to take. Unsatisfactory sketches which are drawn should be consigned to the bin, and only those attempts which hold promise should be pursued in order to minimise machine overheads.

The user wishing to create GOAL data files is expected to be familiar with that programs conventions; ie restricting user geometries to an assembly of coaxial rectangles, and the zone and space numbering systems.

The ENIGMA system carries out four functions:

- 1) Data Capture,
- 2) Data Enhancement,
- 3) Interpretation, and
- 4) Data Output.

This separation will not be aparent to the user.

### 4 Starting The System

After logging in to the DEC 20 computer system, type

EXE ENIGMA,UTIL/LIB,TEKLIB/LIB,TEKPAK

followed by a carriage return. This will produce the following introduction:

```

          X
          X
X   X   X   XX   X   XX   XX   XX   XX   XX   XX   XX   XX
X   X   X   X   X   X   X   X   X   X   X   X   X   X   X
X   X   X   XXXX  X   X       X   X   X   X   X   XXXX
X   X   X   X   X   X   X   X   X   X   X   X   X   X
XX  XX   XXX  X   XX   XX   X   X   X   XXX   X   XX

```

```

XXXXXX  XX   XX  XX   XXXXX   XXX   XXX   XX
XX      XXX  XX  XX  XX   XX  XX  XX  XX   XX  X
XX      XX  X  XX  XX  XX   XX  XX  XX  XX   XX  X
XX      XX  XXX  XX  XX       XX  XX  XX  XX   X
XXXX    XX   XX  XX  XX   XX   XX  XX  XX  XX   X
XX      XX   XX  XX  XX   XXX  XX  XX  XX  XX   X
XX      XX   XX  XX  XX   XX  XX  XX  XX  XXXXXXXX
XX      XX   XX  XX  XX   XX  XX  XX  XX  XX   X
XXXXXX  XX   XX  XX   XXXXX  XX  XX  XX  XX   X

```

IF YOU HAVE AN EXISTING DATA FILE TO WORK ON,  
PLEASE ENTER IT'S (6 CHARACTER MAX) NAME.

It is possible that you will have an ENIGMA data file dating from a previous session, or perhaps an existing GOAL or BIBLE file upon which you want to work. In this case, the user is requested to enter the file's name. When a name is entered, the system will respond by asking;

IS IT A BIBLE, GOAL OR ENIGMA FILE?  
(ENTER -1, 0 OR 1 RESPECTIVELY)

In the case of either a BIBLE or GOAL file, the system will then interpret your file into an ENIGMA file, called PLAN03.DAT, which will be opened in 'append' mode, such that any addition of data will be added to the back of the file. The original file will remain in your user area undisturbed.

An existing ENIGMA file will simply be copied into PLAN03, and the same conditions will apply as with the preceding cases.

Following this, or following a carriage return should you not have an existing file upon which you wish to work, the system will prompt;

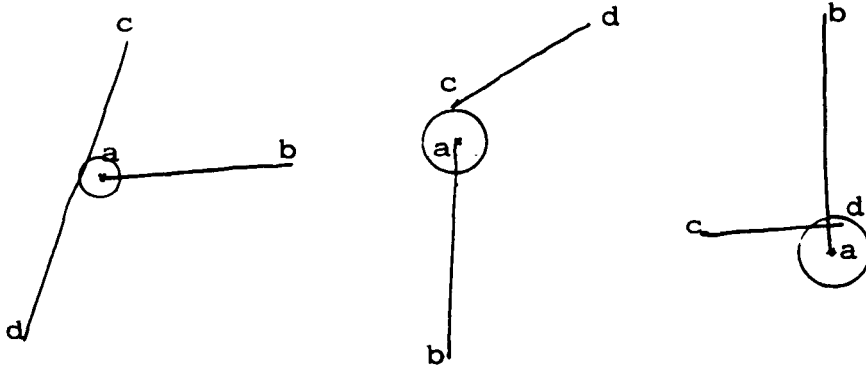
WHAT SCALE DO YOU WISH TO DRAW TO? (1 TO ?)

The user then types in the integer value of the scale, ie. 50 (as in 1:50), 100 (as in 1:100) etc.

The system will respond by asking;

DO YOU WISH TO DEFINE A TOLERANCE VALUE?

This value is important, since it defines how close two lines may be drawn before the computer considers them to meet. This is illustrated below.



A)

B)

C)

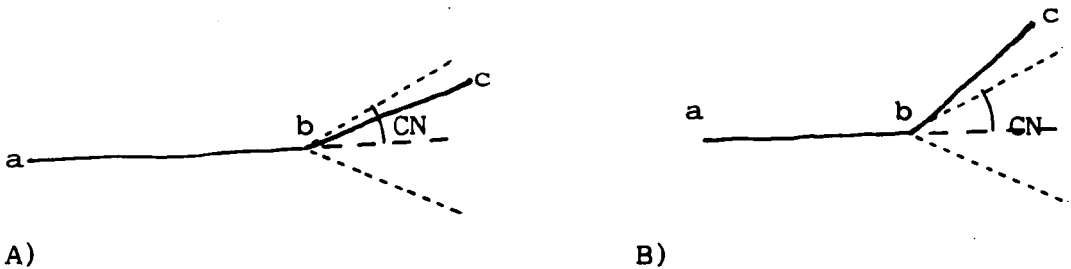
In these instances, the Tolerance Value defines a circular area around the end 'a' of the line ab. In the first instance, A), because line cd passes through the zone, the system will recognise that line ab is intended to meet the line cd. In example B), because the end 'c' of line cd does not lie in the tolerance zone, the system will not recognise that the two lines ab and cd are intended to meet. In the final example, C), the point 'd' of line cd does lie within the tolerance zone. Consequently, their point of intersection is recognised as being the terminal point of both lines.

The tolerance value is entered as a real number to scale, ie. a value of 0.5 at a scale of 1:50 will represent a tolerance radius of 50cm, or in real terms, 10mm on the drawing board. Should you not wish to specify a tolerance value, the system will create a value for you, which will be 5mm in real terms on the drawing board. If you do not wish to select a tolerance value, simply type NO, followed by a carriage return.

The system will then prompt,

DO YOU WISH TO ENTER A CONTINUITY ANGLE?

This value is also important, since the continuity angle defines the zone in which the pen must move, relative to the line being drawn, in order for the computer to determine an intentional change of direction. This is illustrated below.



In the first example, A), point 'c' lies within the cone defined by the continuity value CN, and consequently line abc is treated as one straight line. In the second case, B), the point falls outwith the cone, and the system recognises the existence of two separate lines, ab and cd.

If the user does not wish to specify a value, the machine will assume a value by default of 5°, on the instance of the user typing NO followed by a carriage return.

NOTE

The majority of system failures encountered by users will be the result of incompatible values for defining the Tolerance, and the Continuity Angle.

Users might be required to experiment with various values until they find the combination of values which suit their drawing style.

Finally. the system will prompt,

DO YOU WISH TO DRAW IN ORTHOGONAL MODE?

This requires a simple yes/no response depending on whether you wish to generate rectilinear drawings or not.

#### NOTE

It is essential to respond YES to this prompt if you wish to generate GOAL data files as output. In this case, you MUST restrict your building geometries to an assembly of rectangles. (Refer if necessary to the GOAL users' manual).

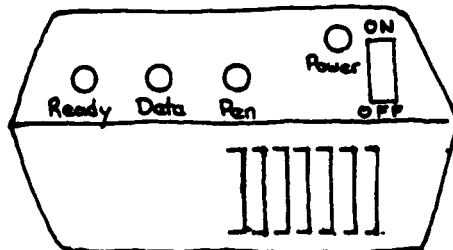
The system will then respond with;

START DRAWING - TYPE 'X' TO EXIT

The terminal screen will then clear, and the READY light on the Tablet control box will come on.

#### 5 Data Capture

The Tablet control box has three red lights on its face in addition to a green indicator neon as shown below.



- 1 Ensure that the pen is connected to the control box.
- 2 Ensure that the Tablet and Terminal are connected to the control box.
- 3 Ensure that the green power indicator neon is on before starting the drawing.

As soon as the pen nib comes close to the tablet surface, the red 'Ready' light will go out, and the 'Data' light will flicker as the pens position is sent back to the terminal. The pen should be held upright like a Rapidograph in order to ensure the best contact Tablet and stylus. Although the pen is being tracked, no data will be recorded by the computer until the pen is pressed against the

paper mounted on the tablet. It is important that the pen is pressed positively, but not too firmly against the tablet surface all the while during the drawing process. When the pen is pressed down on the paper, the red 'Pen' light will come on, and the 'Data' light will flicker as the coordinate data is transmitted to the terminal. The picture being recorded will be echoed upon the terminal screen. As soon as the pen is taken away from the tablet, or the pressure on the pen reduced, the 'Pen' light will go out and the data transmission cease. When this happens, you must wait until the 'Ready' light comes on again before you can proceed with the drawing.

Regularly check the monitor echo to ensure that the data has been correctly captured. If the echo is incorrect, press

Control and C simultaneously,

followed by

STA (return),

which will rerun the programme necessitating the redrawing of the sketch plan.

#### NOTE

Do not go over lines twice. This will result in a garbled interpretation of the drawing at a later stage. If in doubt, restart the drawing process by the above procedure.

When the drawing echo on the monitor has been recorded to your satisfaction and you have no additions to make press the 'X' key on the terminal to exit.

## 6 Enhancement

Following the exit from the data capture sequence the screen of the monitor will clear. The system will request from you

TERMINAL TYPE) to which the response is 1, and  
LINE SPEED) to which the response is 1200.

The system will then redraw upon the terminal monitor the straight line version of the freehand drawing, followed by the query



IS THE PICTURE O.K.?

This query currently exists in anticipation of editing routines to be made available following further development. At present, if the line drawing is incorrect, then you will have to restart the programme. Following an affirmative response, the system will split any bisected lines. It is in this routine that system generated errors, if any, are likely to occur. Consequently the system will redraw the new interpretation of the picture on the terminal screen following the response to the `TERMINAL TYPE` and `LINE SPEED` requests. This is again followed by the prompt

IS THE PICTURE O.K.?

## 7 Interpretation

The interpretation sequence is entered automatically on a positive response to the last prompt. On completion the system will go on to the output sequence.

## 8 Output

The system will display the following message on the terminal screen:

YOU NOW HAVE THE FOLLOWING OUTPUT OPTIONS:

	SELECTION	EFFECT	CONTROL
1	BIBLE	CREATES A BIBLE DATA FILE.	TERMINATES ENIGMA.
2	GOAL	CREATES A GOAL DATA FILE.	TERMINATES ENIGMA.
3	APPRAISAL	PROVIDES CRUDE RELATIVE PERFORMANCE DATA ON THE CURRENT DESIGN.	RETURNS TO OUTPUT OPTIONS.
4	DRAWOUT	DRAWS A SIMPLE ISOMETRIC PROJECTION.	RETURNS TO OUTPUT OPTIONS.
5	PRINTOUT	LISTS THE CONTENTS OF THE CURRENT ENIGMA FILES.	RETURNS TO OUTPUT OPTIONS
6	STOP	HALTS ENIGMA.	TERMINATES ENIGMA.

You are now expected to enter a value of 1 to 5 corresponding to the output option of your choice. Any other number will terminate ENIGMA, saving the current files. These saved files will be stored in your user areas under the names `LINEDT.DAT`, `NODEDT.DAT`, `PRIMDT.DAT`, and `FMCOMP.DAT`. These files contain the list of lines comprising

the picture, the list of points defining the individual spaces, the index to the preceding point list, and the areas of the spaces listed in the preceding file, respectively.

## 8.1 BIBLE

On the selection of BIBLE, a BIBLE geometry data file will be created. When the file has been created, ENIGMA will end, and you may commence running BIBLE by entering the instruction,

RUN PUB:BIBLE

The BIBLE data file for input will be in your area under the name and extension - BIBLE.DAT.

In ENIGMA, after selecting BIBLE as the output option, each individual space will be drawn on the terminal screen in turn. When each space is drawn, the prompt

Z1=?,Z2=?

will appear in the centre. To this you respond with the floor and ceiling heights (in meters), separated by a comma. When this has been done for all of the discovered enclosed spaces, ENIGMA will stop.

## 8.2 GOAL

On the selection of GOAL, a GOAL geometry data file will, where possible be created. On completion, ENIGMA will stop permitting you to directly run GOAL by entering the instruction,

RUN PUB:GOAL

The GOAL data file for input will be in your user area under the name and extension - GOAL.DAT.

In ENIGMA, after selecting GOAL as the output option, the system will prompt,

WHICH STOREY?

You are required to enter the storey reference number of the plan upon which you are working.

The system will then ask,

ARE THE FLOOR AND CEILING LEVELS COMMON THROUGHOUT?

to which you are required to respond YES or NO. It will save time and effort if the floor and ceiling heights are common, but each space can be entered separately if not. If your response has been YES to the previous question, the system will prompt,

ENTER FLOOR AND CEILING LEVELS

to which you respond with the floor and ceiling levels (in meters) separated by a comma

If your previous response was NO, then each detected enclosed space will be drawn out on the terminal screen separately, and the prompt

Z1=?,Z2=?

will appear in the centre of the shape after it is drawn. The floor and ceilings levels of that particular room should be entered in the same format as above. Consequently this will be followed by the prompt

COMP & ID. NOS?

This requires the USE TYPE reference and ELEMENT numbers. (Refer to the GOAL User Handout or Manual if these concepts are unfamiliar).

When all the data has been collected for all the recorded spaces, ENIGMA will stop.

#### NOTE

Should the system encounter a non rectangular or non-four sided space the following message will be printed on the terminal screen:

I'VE ENCOUNTERED A NON-RECTANGULAR SPACE.  
I'M RETURNING CONTROL TO THE OUTPUT OPTIONS.

In this version of ENIGMA, due to the absence of the editing routines, control will return to the data capture routines, and the sketch redrawn.

### 8.3 Appraisal

Using crude cost data, selection of the Appraisal output option will allow you to make relative comparisons between your various design schemes. The appraisal will print out

- \* the Compactness Ratio,
- \* The maximum heating energy requirement based upon the following assumptions:
  - 1 a temperature difference of 20°C between internal and external conditions.
  - 2 a fabric 'U-value' of 0.6 Wm<sup>2</sup>/°C,
  - 3 single glazing distributed evenly around all four facades, in ratios given in the Building Regulations,
- \* The construction costs in terms of the following elements:
  - 1 Wall construction (assumed cavity brickwork),
  - 2 Glazing (assumed single glazing in timber frames),
  - 3 Floor (assumed to be concrete),
  - 4 Roof (assumed flat construction), and
  - 5 Foundation (assumed to be a 1m deep strip footing).

The system will ask you to

ENTER FLOOR LEVEL (IN METERS), followed by  
ENTER ROOF/CEILING LEVEL (IN METERS).

The following output will then be printed upon the terminal screen:

COMPACTNESS RATIO =  
ENERGY REQUIREMENT =

ASSUMING *	IS THE COST PER METER SQUARE OF WALL
*	IS THE COST PER METER SQUARE OF GLAZING
*	IS THE COST PER METER SQUARE OF FLOOR
*	IS THE COST PER METER SQUARE OF ROOF
*	IS THE COST PER METER SQUARE OF FOUNDATION

THE COST OF	METERS OF WALL	IS
THE COST OF	METERS OF GLAZING	IS
THE COST OF	METERS OF FLOOR	IS
THE COST OF	METERS OF ROOF	IS
THE COST OF	METERS OF FOUNDATION	IS

TOTAL

The system will then rewrite the list of output options, enabling you to extract further information from ENIGMA.

#### 8.4 Drawout

Selection of the Drawout option will produce an isometric projection of your enhanced sketch upon the terminal screen, after you have entered the floor and roof/ceiling levels.

The system will assume that these levels are common to all the spaces, and will take as the base of the projection the corner on plan nearest the origin of the tablet, ie. the bottom left hand corner.

The image will remain on the terminal screen until you press the return key, following which the output option list will be redrawn upon the screen ready for your next command.

#### 8.5 Printout

The Printout option will list the information known about your sketch space by space.

The output format will be displayed in a list under the following heading;

AREA ID	REFERENCE POINT	LINE ID	START POINT	STOP POINT	AREA IN SQUARE METERS	LENGTH OF EDGE
	XR	YR	X1	Y1	X2	Y2

Finally, the output will list

#### EXTERNAL ENVELOPE

AREA ID	REFERENCE POINT	LINE ID	START POINT	STOP POINT	AREA IN SQUARE METERS	LENGTH OF EDGE
	XR	YR	X1	Y1	X2	Y2

The list of output options will then be listed.