A simple data structure for interactive graphic design/drafting

A. F. Cárdenas* and R. W. Seeley†

Powerful data structures for interactive graphics systems are tediously intricate, and are indeed difficult to program, debug, maintain and modify. A simple data structure is presented to represent two- and three-dimensional objects. One of its great benefits is its ease of implementation and maintenance. Although because of its simplicity various facilities provided by more complex data structures are not available, it can be an attractive basis for a class of graphics systems, such as the operational interactive design/drafting system outlined.

(Received September 1973)

1. Introduction

A number of data structures have been proposed and implemented in interactive graphics systems to provide powerful facilities for design/drafting. Williams (1971) provides an introductory overview of various data structures and their suitability for computer graphics, and of a few selected (*a*) languages with some facilities for creating and manipulating data structures, and (*b*) existing graphics systems and their data structures. Hamilton's work (1970) is also a good survey. Both surveys provide ample bibliographies on the field. Many structures have been conceived for a variety of applications, e.g. 'inverted files' for information retrieval. Some are more suitable and convenient, and essentially only applicable, to particular applications, such as object manipulation via computer graphics.

The data structure described here is purposely a simple one. Although because of its simplicity various facilities provided by other more complex data structures are not available, it is the basis of an effective and attractive interactive graphics system currently in operation. The great benefit of this simple structure, compared to any other, is ease of implementation and maintenance. Powerful data structures for graphics are tediously intricate, and are indeed difficult to program, debug, maintain and modify—a factor too often underestimated. They tend to grow in an *ad hoc* and unpredictable manner under the pressure of more and better facilities for users and performance, and also quite often have to depend upon specific machine characteristics. Furthermore, basic systems software and hardware facilities provided with graphics terminals for applications programming are still rudimentary, besides being non-transferable across machines.

2. The simple data structure

An interactive graphics system for design/drafting called the Structure Edit Program (SEP), was implemented based on the simple structure for use by structural engineers to display and manipulate two- and three-dimensional structures. Structural engineers, when designing the frame of a building or airplane fuselage, represent the object as a three-dimensional finite element structure consisting of points in space called joints and lines called bars which connect these joints. The program was written by the author Seeley in FORTRAN IV and GSP (Graphic Subroutine Package) (IBM, 1971) and uses IBM 2250 Model terminals as the graphic device.

The basis for this system is the simple data structure shown in **Fig. 1.** It consists of two sequential arrays. One array, the joint array, contains the X, Y and Z values of each joint in the structure. The joint number of a joint is its sequential order in the array. The second array, the bar array, contains pointers pointing to the end joints, in the joint array, defining a bar. The bar number of a bar is the sequential order in the array. The pointers in the bar array are simply the joint numbers of the end joints in the joint array.

The addition of joints and bars to the arrays is kept simple by adding the data to the end of the arrays. Deletion of bars and joints becomes more complicated and time consuming. A bar is generally deleted from anywhere in the array. Since the array can be considered as a sequential list, 'holes' in the list cannot be tolerated. To avoid this complication the list is compacted to fill in the hole, and all bar numbers from the deleted bar to the end of the array must be decremented by one. **Fig. 2**(*a*) demonstrates the deletion of a bar.

Similarly, when a joint is deleted from anywhere in the array, the 'hole' is filled by compaction. The joint numbers before the

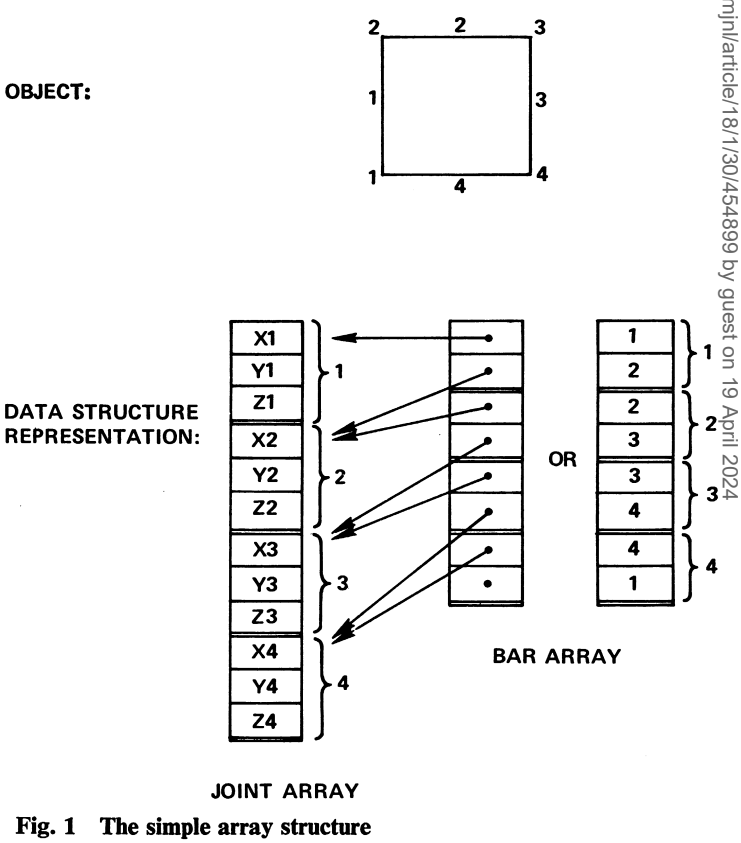

Fig. 1 The simple array structure

*Computer Science Department, Boelter Hall 3731, University of California; Los Angeles, Los Angeles, Calif. 90024, USA.
†McDonnell Douglas Automation Company, Long Beach, California: Current Affiliation: Computer Vision Corporation, Bedford, Massachusetts 01730.

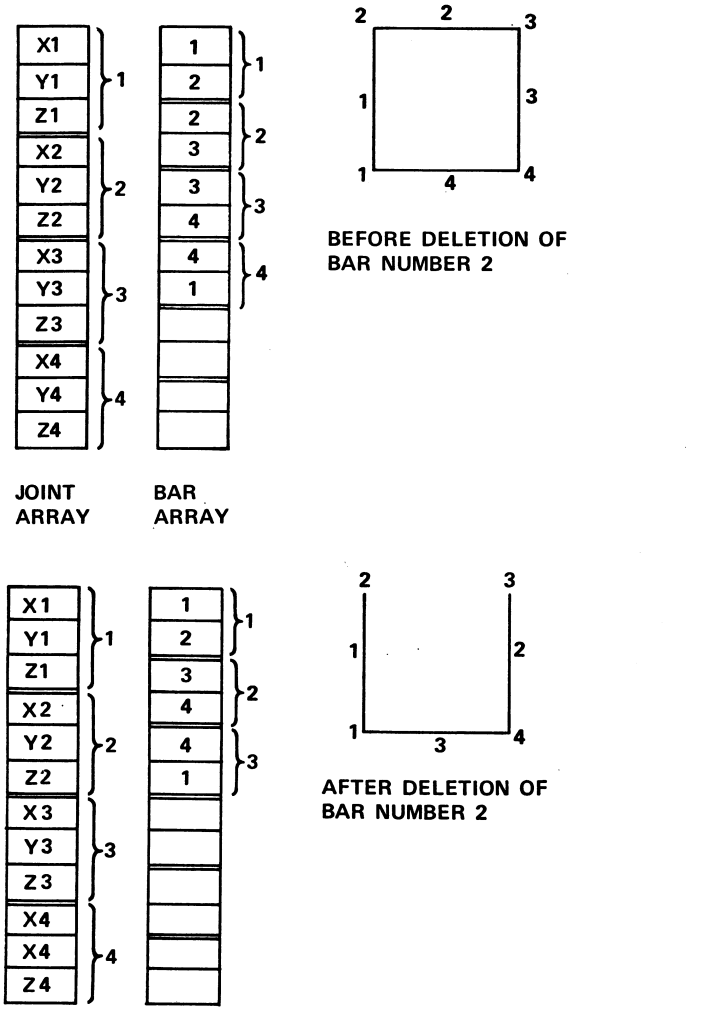

Fig. 2(*a*) The simple array structure—bar deletion

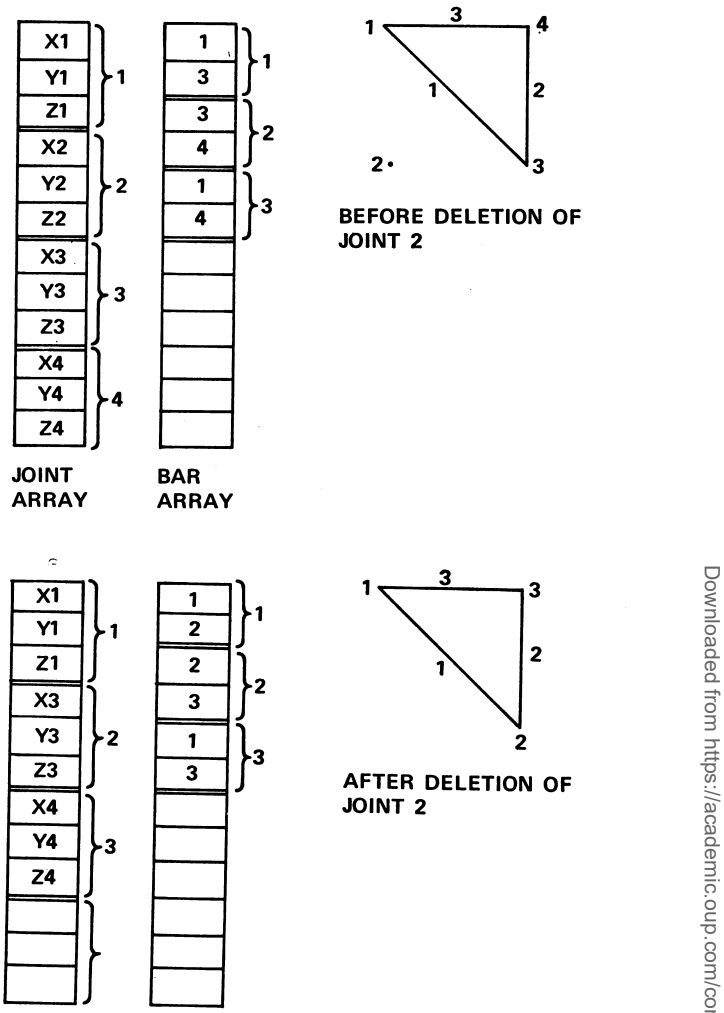

Fig. 2(*b*) The simple array structure—joint deletion

deleted joint are decremented by one due to their shift in the array and the joint numbers in the bar array greater than the deleted joint number must be decremented by one. It is a rather time consuming process for the deletion of one joint. **Fig. 2**(*b*) demonstrates the deletion of a joint.

A second method for deletion reduces the time required to delete an element. Instead of compacting the bar array to eliminate a 'hole', the hole can be 'plugged' with the last bar element in the array. A joint may be deleted in a similar manner; however, the numbering scheme in the bar array must be updated. This requires one pass through the bar array searching for all occurrences of the last joint number (joint element used to fill the 'hole'). This joint number is replaced by the joint number of the deleted joint. This second method has not been implemented. Since the data structure is core resident, the decrease in computer time would be insignificant to the observer.

3. Advantages and disadvantages

The great benefit of the simplicity of the data structure is ease of implementation. This ease is relative. The implementation of interactive graphic systems is inherently intricate and costly. SEP consists of about 8,000 source statements. It is written in FORTRAN IV H and graphic communication is accomplished through GSP. It is made up of 130 subroutines, 63 overlay segments, and runs in a region of 164K bytes of core. Among these subroutines are the subroutines to carry on the basic data manipulation function: define and insert a joint into the joint array; delete a bar and compress the bar array; retrieve the coordinates of a joint; etc. Another group of subroutines performs other essential tasks such as provide convenient error messages to users, indicate dialogue options to users, etc. This type of task is often not regarded enough by system designers. The remaining subroutines may be termed as 'overhead' (but essential) subroutines; they support the system, often 'glue it together' in actual practice. Other structures reported in the literature (starting with SKETCHPAD, CORAL, etc. (Williams, 1971; Hamilton, 1970)) require sophisticated routines to allocate storage, deallocate storage, trace pointers for insertion and deletion of blocks, format each block type, update pointers, and carry on other 'overhead' tasks. Graphics systems based on these structures involve larger and more intricate programs.

Another advantage of the simple structure is the speed of retrieval. A joint or bar can be retrieved directly by computing the location in the array. With other structures, retrieval time will be longer since usually long lists of pointers or rings must be traced. On the average, half of the blocks on a ring in ringed structures have to be searched to retrieve a specific block.

The array structure uses no pointers to indicate the next logical block (joint or a bar). Without the use of pointers, the array structure has no ability to form hierarchies or share common data. In fact, no subpictures can be defined. There is only one picture and that is the entire data structure. Operations such as scaling, rotation and translations must be performed on the entire structure. This is where sophisticated structures show their advantages. However, the user is often unaffected by some of these internal factors since the simple structure does permit such functions, although at a higher processing cost.

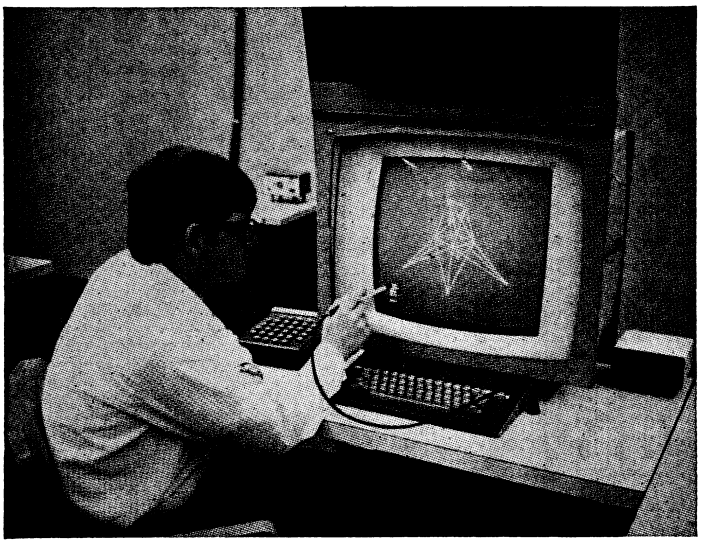

Fig. 3 Photograph of author using the structure edit program SEP on an IBM 2250

Fig. 3 is a photograph of the author Seeley using SEP on the IBM 2250. Hardcopy of displays is obtained (*a*) instantly by means of a Polaroid camera attached to a swing-down hood or (*b*) by means of a Stromberg Datagraphix 4060 plotter or a Gerber plotter. Simple as the basic structure is, the implemented SEP offers powerful and useful facilities to conveniently build and manipulate complex object structures. Users may, for example:

1. Design and modify frame structures through facilities to add, modify and delete joints and line (bar) segments via light pen and keyboard.
2. Rotate, scale and translate objects to acquire better viewing, and correct such errors as missing points and lines, lines connecting the wrong points, etc. often incurred in designing complicated objects.
3. Mirror structures about a plane, by light pen detecting on three joints or typing-in their coordinates; this is a powerful operation in building up a structure from basic building blocks.

Figs. 4(*a*), **4**(*b*) and **4**(*c*) demonstrate the mirroring techniques. Fig. 4(*a*) shows an object with two bottom bars divided into five bars each. In Fig. 4(*b*) it has been scaled down and mirrored about the vertical plane. The object in Fig. 4(*c*) was built by simply mirroring Fig. 4(*b*) about its top horizontal plane.

The SEP has been developed to the limits of the capabilities of the array type data structure. Further development is not warranted. Other more powerful data structures would have to be implemented to provide more facilities than current ones in SEP.

4. Concluding remarks

Graphics systems are tediously intricate and difficult to implement, debug, maintain and modify. The use of the simple data structure as the basis for the operational and satisfactory interactive graphics system outlined was prompted by its ease of implementation and maintenance compared to other more complex structures. The simple structure formalised here is a practical and attractive basis for a class of graphics systems.

Acknowledgement

Thanks are due to Dr. Robin Williams, IBM Research Laboratory, San Jose, California, for his helpful comments in this article. The authors appreciate also the criticism of Mr. Malcolm Atkinson, Computer Laboratory, University of Cambridge.

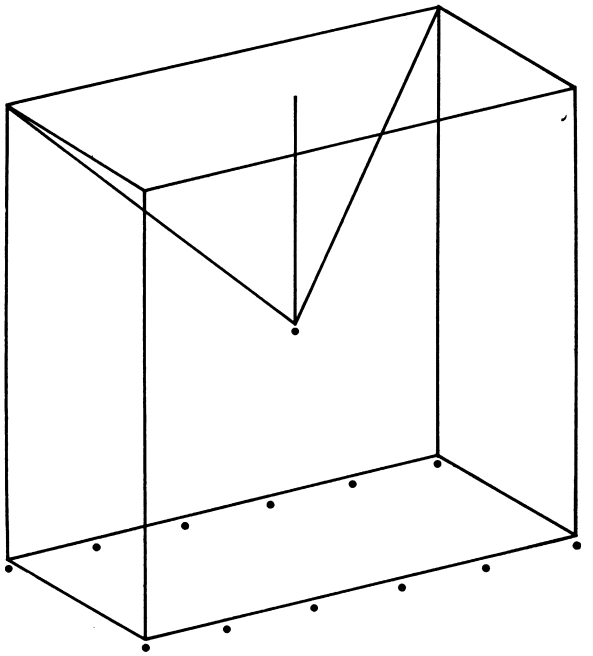

Fig. 4(*a*) Bars divided into bar segments

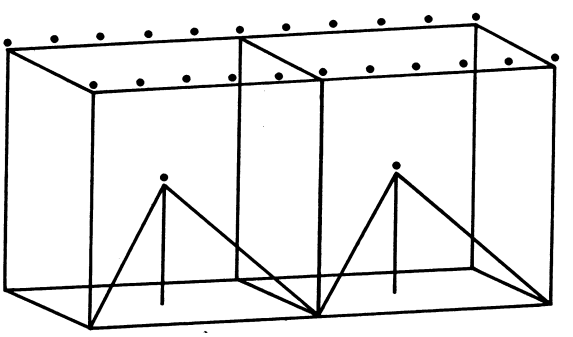

Fig. 4(*b*) Structure mirrored.

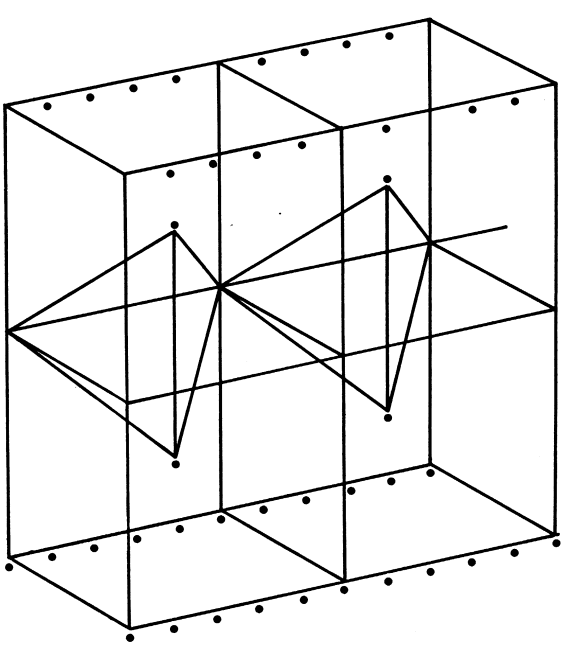

Fig. 4(*c*) Structure mirrored

References

HAMILTON, J. A. (1970). A Survey of Data Structures for Interactive Graphics, The Rand Corporation, *Report RM-6145-ARPA*, April 1970, Santa Monica, California.

IBM System/360 Operating System Graphic Subroutine Package (GSP) for FORTRAN IV, COBOL and PL/I, Form C27-6932, IBM Corporation, Programming Publications, Kingston, New York.

WILLIAMS, R. (1971). A Survey of Data Structures for Computer Graphics Systems, *Computing Surveys*, March 1971, Vol. 3, No. 1, pp. 1-21.

Book reviews

Computer Data Processing, by G. B. Davis (2nd edition), 1973; 662 pages, (*McGraw-Hill*, £6·35.)

This is a text-book of data processing aimed at students having their first encounter with computing. Substantial changes have been made to the original edition.

The book is clearly-written and copiously illustrated. There are simple exercises at the end of every chapter and three self-test quizzes. The claim that it is suitable for self-study is justified.

Until I reached page 122, I read every word and mentally answered the exercises. This diligence was founded more on a desire to be scrupulously fair than any other reason. I then realised, with dismay, that there were still 540 pages to go and the Journal would be lucky to get my review before the 3rd edition appeared, unless I changed my tactics. I therefore apologise in advance for the less intense examination of the balance of the book.

This experience serves to emphasise the encyclopaedic proportions of the work. There are very few topics, relevant to the subject, which do not receive at least a mention. In his preface, Professor Davis argues that 'given the rate at which we forget, it is usually better to err on the side of telling too much than too little'.

The result of this philosophy might be catalogued as follows: unit record equipment and systems—45 pages; computer hardware and number systems—122 pages; information system design and development—108 pages; programming—138 pages; management—108 pages; miscellaneous—121 pages.

In achieving this breadth, it is understandable that depth has suffered. I do not think the treatment of most of the topics included would satisfy a specialist in that area, even as an introduction, at any rate because of what is omitted if not for what is said. For this reason, I cannot see students of, say, a degree level course with a major data processing content being advised to buy the book; they would be better served by a collection of more specialised books.

A weakness in the text is its inadequate exposure of practical, human problems facing data processing personnel, whether in their dealings with each other and users or in the design of man/machine interfaces. Also, to a reader who is accustomed to the elegant simplicity of the NCC standards, the ANSI/ISO flowcharting symbols used throughout the book will appear cluttered and cumbersome. A small point is that to understand some examples and exercises a knowledge of American terms is required (e.g. 'sophomore', 'non-alumni contributors', 'FICA-tax').

A particular strength of the book is its description of hardware devices; I cannot put my finger on another work which explains such a variety so well. The whole book is also commendably up to date.

As a text-book, then, it might be set for someone not specialising in the area but who needs a wide appreciation of data processing; for example, a student of business subjects. As general reading, it would interest a newcomer to the computing profession who wished to broaden his knowledge. A teacher might want a copy on his library shelves for his students specialising in data processing, if only for the factual hardware content and the well selected bibliography. For all these uses, the price per page gives good value for money.

ANDREW PARKIN (Leicester)

Automated Analysis of Drugs and Other Substances of Pharmaceutical Interest, by C. T. Rhodes and R. E. Hone, 1974; 291 pages. (*Butterworths*, £5·95 hardcover, £3·95 paperback.)

This book gives a wide and brief coverage of automated analysis of drugs, which is of great value to analysis in the pharmaceutical industry, to pharmacologists and to clinical chemists. There is a growing demand for such analyses with the rapidly increasing number of drugs, and with the increase in statutory requirements for quality control of drug manufacture.

The scope of application of automated analysis of drugs (or metabolites) in biological fluids is wide; it includes studies on new drugs, on new formulations (for bioavailability), and on all drugs where a delicate control of dosage is required.

The book provides a general outline of the principal approaches, citing only a few examples of each. Thus, it first deals with scope of the subject, then the use of automated apparatus for chemical and physical measurements, data processing, evaluation of dosage forms, and the scope of automated procedures in drug analysis in biological fluids. This is followed by a narrower but deeper coverage of analysis of selected groups of drugs, particularly antibiotics (including specialised microbiological techniques), steroids and vitamins. Specific examples of the methods of estimation of certain elements such as halogens, drugs containing certain chemical groups (e.g. amine groups), and certain enzyme analyses are cited.

The concluding chapter partly answers the expected questions from those who wish to make the best of their presently available facilities for the development of some kind of automated drug analysis. Many interested research workers would have liked to see some coverage of analysis of a few new or relatively new groups of drugs, of automated biological and immunoassay procedures, and of the comparison of drug analysis by various methods.

J. W. BLACK (London)

An Introduction to Automated Electrocardiogram Interpretation, by P. W. MacFarlane and T. D. V. Lawrie, 1974; 115 pages. (*Butterworths Computers in Medicine Series*, £2·20.)

This short book makes an admirable introduction to the present state of the art. The authors describe briefly the principles of electrocardiography and techniques of ECG recording and data transmission, and then go on to explain the general principles of economical analogue to digital conversion. They discuss in somewhat greater detail the algorithms underlying the automatic interpretation of the trace and illustrate this in particular by the technique of analysis of arrhythmias.

The reader should not assume that there is any detail in the text which would be sufficient to help someone to program a computer to analyse electrocardiograms. However the book can be strongly recommended to cardiologists and hospital administrators who are concerned with the economics and cost/benefit aspects of automated electrocardiographic interpretation, and the book contains at the end some realistic assessment of what this technique has to offer to a group of cooperating hospitals.

C. J. DICKINSON (London)