(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) Publication number: **0 145 817**
B1

(12) # EUROPEAN PATENT SPECIFICATION

(45) Date of publication of patent specification: **10.08.88**

(51) Int. Cl.[4]: **G 09 G 1/16,** G 09 G 1/00

(21) Application number: **83307697.9**

(22) Date of filing: **19.12.83**

(54) **A data display system.**

(43) Date of publication of application:
26.06.85 Bulletin 85/26

(45) Publication of the grant of the patent:
10.08.88 Bulletin 88/32

(84) Designated Contracting States:
DE FR GB IT

(56) References cited:
IBM TECHNICAL DISCLOSURE BULLETIN, vol. 20, no. 10, March 1978, pages 4161-4162, New York, USA; D.P. ATTWOOD: "Graphic display protect scheme"

(73) Proprietor: **International Business Machines Corporation**
Old Orchard Road
Armonk, N.Y. 10504 (US)

(72) Inventor: **Impey, Alan A.**
"Chancels" The Drive Whinwhistle Road
E. Wellow Romsey Hants SO5 0BN (GB)
Inventor: **Powell, Colin V.**
103 Kingsway
Chandlers Ford Hants SO5 1FD (GB)
Inventor: **Simmons, John**
13 Parkway Gardens
Chandlers Ford Hants SO5 2EN (GB)
Inventor: **Easton, Colin J.**
"Bywood" Moorhill Road West End
Southampton S03 3AX (GB)

(74) Representative: **Appleton, John Edward**
IBM United Kingdom Limited Intellectual Property Department Hursley Park
Winchester Hampshire SO21 2JN (GB)

Courier Press, Leamington Spa, England.

EP 0 145 817 B1

Description

This invention relates to a data display system and particularly to a display system which is used as an input-output terminal of a data processing system which may be used for concurrent processing of application programs.

Viewporting is the generic name given to the technique of defining a particular screen area as the viewport on which an application writes and displays data-graphic or alphanumeric. When a user is using a display terminal to interact with more than one application program then different areas of the screen will be allocated to different applications this is multiple view porting. This concept is explained in Fundamentals of Interactive Computer Graphics by Foley and Van Dam published by Addison Wesley 1982.

A further development has been the so-called 'messy desk' concept in which multiple viewports overlap and the user regards the view which overlays the others as that which highest priority and the one which is currently being used.

Viewporting designs for current raster displays use the concept that only the viewport which is of highest priority, i.e. on top of, or overlaying, all others can be modified at any one time. This, in effect, corresponds to a single application situation and requires the complete re-drawing of a viewport whenever it is given highest priority after it has previously been overlaid.

An example of such a technique is described in European Patent EP—A—0121015.

This application describes a multi-viewport system in which the writing of application data into overlapping viewports is controlled by a screen manager. The screen manager maintains a series of priority flags for each pixel (bit in the screen buffer) relating to the layers of the viewports, and a viewport order list. Only the current, or higher priority viewport is written into by an application. There is no provision for having more than one application writing into a lower priority viewport overlapped by the current viewport other than serially, i.e. writing to one viewport is completed before processing the next one.

IBM Technical Bulletin Vol 20, No. 10, March 1978 page 4161 "Graphic display protect scheme" by D. P. Attwood shows a scheme in which the parity bit in a data register associated with each display pel is used to inhibit rewriting in the display buffer. However as in the above mentioned application there is not provision for having more than one application writing into a lower priority viewport other than serially.

According to the invention there is provided a data display system in which a display screen may simultaneously display data relating to different applications in overlapping viewports having different priority levels including a pixel buffer in which the picture data to be displayed on the screen in stored, a mask buffer, setting means to set the contents of the mask buffer and logic means that uses the contents of the mask buffer to determine whether a pixel in the pixel buffer should be written during vector or character drawing, characterised in that the mask buffer has a plurality of bit positions equal to the total display screen pixels each one associated with a corresponding display screen pixel, the setting means sets the contents of the mask buffer when an application sends data to an associated viewport in accordance with the viewport areas having a high priority and controls the operation of the ligic means whereby when data is to be written to a viewport the area of the mask buffer corresponding to the viewport to be written is reset and then for each higher priority viewport in sequence a bit plane write enable register is set so that writing only takes place to the mask buffer plane and each bit in the mask buffer corresponding to a pixel within the higher priority viewport is set.

In order that the invention may be fully understood preferred embodiments thereof will now be described with reference to the accompanying drawings in which:

FIG. 1 is a block schematic of a data processing configuration including a data display system incorporating the invention.

FIGS. 2—4 illustrate multiple viewports on a display.

FIG. 5 is a block screen schematic of a data display system.

FIG. 6 illustrates the content of the mask buffer before drawing data to a view port.

Until recently, a user has only been able to use a terminal for a single, interactive session at any one time. This session might be conducted with a host application, or with an application running in stand-alone workstation. The range of applications available is very broad and includes word-processing, data base operations, communications and graphics design. These applications might be commercially written, or coded by the user.

The growing complexity of users, applications and workstations, has brought about the requirement to interact with one or more applications at one time. Thus, while writing a letter, with a word-processing system, a user may wish to obtain information from a data base or perform some financial calculation with a spread-sheet application, and include this information in the letter. To handle these concurrent, multiple applications, the concept of the "messy-desk" has evolved. This concept allows a user to allocate a subarea, or viewport, of the workstation screen, to each of the concurrently running applications.

In general, each application generates output to a conceptual presentation space, which is considered to be mapped onto the entire workstation screen. Each viewport provides a window onto the corresponding presentation space. The contents of the presentation space, contained within this window, are displayed on the screen viewport. While the viewport remains in a fixed position on the screen, the window may be moved, or scrolled, over the presentation space, under user control, thus allowing the whole of the

presentation space to be accessed and viewed. For maximum flexibility in screen layout, the user may change the size of a viewport and its position on the screen.

While the user is directly interacting with a particular application, one or more of the other applications may be computing the results of a previous user request. Thus, for example, a communications application may be providing a continuous report on the movement of stocks and shares, or monitoring the status of a major power station; a data base application might be searching and reporting occurrences of a subject in a world wide collection of information systems. This multi-application system allows the user to retrieve information when it is needed, rather than when the current application has terminated.

In the full "messy desk" concept, these viewports may overlap on the screen, just as several documents might overlap on someone's desk. They would be arranged, by the user, so that the portions of interest are visible, even if other viewports partially obscure them. Thus, while a user is interacting with one viewport, information, from other viewports, is immediately available for reference. However, unlike a document on a desk the content of these other viewports may be dynamically changing, depending on the corresponding applications.

In the "messy desk" concept, the active viewport, with which the user is currently interacting, overlays the other viewports. The user can chose to interact with one of the other viewports, when it will "pop" to the top so that it overlays the others.

The purpose of this invention is to provide a technique whereby an application, associated with an overlaid viewport, may dynamically update its entire presentation space, have this new data immediately displayed on the visible, non-overlayed, portion of the viewport, and yet not interfere with the overlaying viewports.

FIG. 1 shows in schematic form a data processing system in which a host data processor 10 which can be a multi-processor such as an IBM 3033 will process data relating to many application programs concurrently. The processor can support many display systems such as that through a link 12. The display system comprises a display terminal 14, a keyboard 16, a light pen 18 or interactive device such as a mouse or a tablet.

The display system may be communicating with the host processor with data relating to up to four application programs at a time. A fifth application program may be running on a locally processor 20, such as an IBM Personal Computer.

In the configuration shown in Figure 1 data-streaming may take place to one or more viewports other than the viewport with which the operator is currently interacting. In general, these lower priority viewports may be overlaid by the active viewport and possibly by one or more other viewports.

When only a single viewport is in use, or when the viewport receiving alphanumeric or graphic data overlays all other viewports, for example viewport B in Figure 2 then output may be drawn directly, using normal clipping procedures within the viewport boundary. However, when output is to be drawn in viewport A, which is partially overlaid by viewport B, then the normal rectangular clipping algorithms become more complex.

A simple scheme would be to sub-divide viewport A into two rectangles A1 and A2 as shown in Figure 3. The alphanumeric and graphic data would be written twice, first clipping to the boundary of area A1 and then a second time, clipping to the boundary of area A2. This involves two passes over the data which can be time consuming.

In a more complex situation for example viewport D in Figure 4, the viewport would need to be split into five rectangular areas for the purposes of clipping requiring five passes over the data. In addition there is the complexity of determining the clipping boundaries in the optimum way for a general set of overlapping viewports.

A preferred embodiment of the invention will now be described with reference to Fig. 5. Fig. 5 shows schematically the component parts of the display unit including a drawing order store 50 which receives data from up to four applications from the host processor 10 and one from the local processor 20 on lines 51.

Graphic and alphanumeric data is received in the form of orders which may include a vector list. A formatter 52 tranforms the order list to a form suitable for a raster scan and passes the rastered data to a bit plane logic device 54. A pixel buffer 56 includes bit positions one associated with each pixel on the display screen and is read out for each refresh cycle of the display unit. A mask buffer 58 is connected to the bit plane logic device 54 and together control the writing of raster formatted information into the pixel buffer. The display unit components are controlled by a control unit 60 which contains a microprocessor and its associated ROM and RAM stores with the microcode for controlling the operation of the unit.

The actual picture data for all viewports is stored in the pixel buffer 56. Typically this buffer is made up of one or more bit planes. The mask buffer 58 comprises a single bit corresponding to each pixel of the main pixel buffer 56. Typically the mask buffer comprises an additional bit plane to those used for the pixel buffer. The logic of the bit plane logic 54 uses the content of the mask buffer 58 to determine whether a pixel should be written in the pixel buffer during character or vector drawing.

The control unit 60 enables or disables the mask buffer so that when the mask buffer is disabled all pixels, including the mask buffer bit of the pixel may be written regardless of the current mask content. However, when in the enable mode a mask bit having a '1' value inhibits the writing into the associated pixel in the pixel buffer planes.

Pixel data is written to the bit planes of the pixel

and mask buffers under the control of the control unit 60. The bit plane logic device 54 includes a bit plane enable register which selects the bit planes that are to be updated. If a bit of the register is set, then the corresponding bit plane may be written. A bit for the mask buffer is included in the register.

A single control bit enables the use of the mask buffer as an overall control of whether a particular pixel is actually updated. When enabled, data from the mask buffer is read by the update logic and used to determine whether the corresponding pixel is to be updated.

The operation under the control of the control unit 60 of the embodiment is as follows.

The area of the mask buffer corresponding to the viewport to be written is reset to zero. This allows pixel drawing anywhere within the viewport. This reset operation is implemented using normal 'area fill' methods simplified for speed by taking account of the rectangular viewport shape.

Then for each higher priority viewport in sequence:

1. Set the Bit Plane Write Enable register so that writing only takes place to the mask buffer plane.

2. Turn on (set) each bit in the mask buffer corresponding to a pixel within the higher priority viewport. Again, a simplified, 'area fill' method may be used.

Then the control unit enables the use of the mask buffer to control the writing of pixel data.

The Bit Plane Write Enable register is set to allow writing to the pixel buffer, but not the mask buffer.

The data is then written into the low priority viewport, but only into these pixels whose corresponding mask bit is zero.

The operation is concluded by the disabling of the mask buffer to allow normal writing operations to the higher priority viewport.

FIG. 6 illustrates the content of the mask buffer just before the start of writing data to viewport D of FIG. 4. It shows how the overlaid areas of viewport D are indicated by the mask of '1's in the mask buffer and consequently inhibit the further modifications of pixels in these areas. Each '1' in the mask buffer inhibits pixel modification while viewport D data is being written. Normal clipping operation for the whole of viewport D will prevent pixels outside of the viewport being modified.

The hardware implementation of the invention described above may be simulated in software.

This second embodiment is applicable for display systems where the additional bit plane is not available, or where existing hardware cannot be modified to allow one of the existing bit planes to be used for the "mask" buffer.

The algorithm used is similar to that described above for the hardware solution, except that no hardware assist is used. It is entirely a software solution. In this case, it is assumed that the system processor, or possible a second slave processor, is used to draw the vectors in a set of bit planes. A pseudo "mask" buffer is built-up in normal processor storage. It is used during drawing operations to determine whether a particular pixel is to be drawn in the bit planes as described for the hardware implementation. If a second processor is used to draw the actual vectors, storage within its own address space might be used to contain the "mask" buffer.

Claims

1. A data display system in which a display screen (14) may simultaneously display data relating to different applications in overlapping viewports having different priority levels including a pixel buffer (56) in which the picture data to be displayed on the screen in stored, a mask buffer (58), setting means (60) to set the contents of the mask buffer and logic means (54) that uses the contents of the mask buffer to determine whether a pixel in the pixel buffer should be written during vector or character drawing, characterised in that the mask buffer (58) has a plurality of bit positions equal to the total display screen pixels each one associated with a corresponding display screen pixel, the setting means (60) sets the contents of the mask buffer when an application sends data to an associated viewport in accordance with the viewport areas having a high priority and controls the operation of the logic means (54) whereby when data is to be written to a viewport the area of the mask buffer (58) corresponding to the viewport to be written is reset and then for each higher priority viewport in sequence a bit plane write enable register is set so that writing only takes place to the mask buffer plane and each bit in the mask buffer corresponding to a pixel within the higher priority viewport is set.

2. A data display system as claimed in claim 1 in which the pixel buffer (56) comprises a plurality of bit planes, each plane having a number of bits equal to the number of picels on the display screen and the mask buffer is a bit plane of an equal number of bits to the pixel bit planes.

3. A data display system as claimed in claim 1 or claim 2 in which the operation of the system is under the control of a control unit (60) comprising a microprocessor and associated read only store and random access stores.

4. A data display system as claimed in any one of claims 1, 2 or 3 in which the operation of the mask buffer is simulated in a data processor.

5. A data display system as claimed in claim 1 further characterised by including a drawing order store (50) for receiving commands relating to information to be displayed on the display screen and a formatter (52) to convert drawing order information from a vector to a raster scan form.

Patentansprüche

1. Datenanzeigesystem, bei dem auf einem Bildschirm (14) Daten, welche sich auf unterschiedliche Anwendungsbereiche beziehen, gleichzeitig in sich überlagernden Sichtfenstern

mit unterschiedlichen Prioritätsebenen erscheinen können, mit einem Bildpunktzwischenspeicher (56 zum Speichern der auf dem Bildschirm darzustellenden Bilddaten, einem Maskenzwischenspeicher (58), einer Aufsetzeinrichtung (60) zum Aufsetzen des Inhalts des Maskenzwischenspeichers und mit einer logischen Einrichtung (54), welche mittels des Inhalts des Maskenzwischenspeichers feststellt, ob ein Bildpunkt im Bildpunktzwischenspeicher während des Zeichnens von Vektor oder Zeichen geschrieben werden sollte, dadurch gekennzeichnet, daß der Maskenzwischenspeicher (58) eine Vielzahl von Bitpositionen enthält, die gleich der Anzahl sämtlicher Bildpunkte des Bildschirms sind, wobei jede einem entsprechenden Bildpunkt des Bildschirms zugeordnet ist, daß die Aufsetzeinrichtung (60) den Inhalt des Maskenzwischenspeichers aufsetzt, wenn eine Anwendung Daten zu einem zugeordneten Sichtfenster entsprechend den Sichtfensterbereichen hoher Priorität schickt und den Betrieb der logischen Einrichtung (54) steuert, so daß beim Schreiben von Daten in ein Sichtfenster der Bereich des Maskenzwischenspeichers (58), welcher dem Sichtfenster entspricht, das beschrieben werden soll, rückgesetzt wird und dann für jedes Sichtfenster höherer Priorität in Folge ein Biebenen-Schreibfreigaberegister gesetzt wird, so daß nur in der Ebene des Maskenzwischenspeichers geschrieben wird und jedes Bit im Maskenzwischenspeicher, welches einem Bildpunkt innerhalb der Sichtfenster höherer Priorität entspricht, gesetzt wird.

2. Datenanzeigesystem nach Anspruch 1, dadurch gekennzeichnet, daß der Bildpunktzwischenspeicher (56) eine Vielzahl von Bitebenen aufweist, welche jeweils eine Anzahl von Bits enthält, die der Anzahl von Bildpunkten auf dem Bildschirm entspricht und daß der Maskenzwischenspeicher eine Bitebene mit der gleichen Anzahl von Bits wie die Bildpunkt-Bitebenen ist.

3. Datenanzeigesystem nach Anspruch 1 oder 2, dadurch gekennzeichnet, daß der Betrieb des Systems durch eine Steuereinheit (60) gesteuert wird, welche einen Mikroprozessor sowie einen zugeordneten Festspeicher und Speicher mit wahlfreiem Zugriff enthält.

4. Datenanzeigesystem nach einem der Ansprüche 1, 2 oder 3, dadurch gekennzeichnet, daß der Betrieb des Maskenzwischenspeichers in einem Datenprozessor simuliert wird.

5. Datenanzeigesystem nach Anspruch 1, gekennzeichnet durch einen Zeichnungsbefehls-Speicher zum Empfang von Kommandos, die auf dem Bildschirm darzustellender Information zugeordnet ist, und durch einen Formatwandler, der Zeichenbefehls-Informationen vom Vektor- zum Raster-Abtast-Format wandelt.

Revendications

1. Système de visualisation de données dans lequel un écran de visualisation (14) peut visualiser simultanément des données concernant différentes applications dans des fenêtres de visualisation se chevauchant ayant des niveaux de priorité différents, comportant une mémoire tampon (56) de pixels dans laquelle les données d'image à visualiser sur l'écran sont mémorisées, une mémoire tampon (58) de masques, des moyens de mise à un (60) pour mettre à un les contenus de la mémoire tampon de masques et des moyens logiques (54) qui utilisent les contenus de la mémoire tampon de masque pour déterminer si un pixel dans la mémoire tampon de pixels doit être écrit durant le tráce d'un caractère ou d'un vecteur, caractérisé en ce que la mémoire tampon (58) de masques possède une pluralité de positions de bits égales à tous les pixels de l'écran de visualisation, chacun d'eux étant associé à un pixel correspondant de l'écran de visualisation, les moyens de mise à un (60) mettent à un les contenus de la mémoire tampon de masques lorsqu'une application envoie des données à une fenêtre de visualisation associée en fonction des zones de fenêtres de visualisation ayant une priorité élevée et commande le fonctionnement des moyens logiques (54), grâce à quoi, lorsque les données doivent être écrites dans une fenêtre de visualisation, la zone de la mémoire tampon (58) de masques correspondant à la fenêtre de visualisation à écrire est remise à zéro, et, ensuite, pour chaque fenêtre de visualisation de priorité plus élevée dans la séquence, un registre de validation d'écriture de plans de bits est mis à un de façon à ce que l'écriture ne se produise que vers le plan de la mémoire tampon de masques correspondant à un pixel à l'intérieur de la fenêtre de visualisation de plus grande priorité soit mis à un.

2. Système de visualisation de données selon la revendication 1 dans lequel la mémoire tampon de pixels (56) comporte une pluralité de plans de bits, chaque plan ayant un nombre de bits égal au nombre de pixels sur l'écran de visualisation et la mémoire tampon de masques est un plan de bits avec un nombre de bits égal aux plans de bits de pixels.

3. Système de visualisation de données selon la revendication 1 ou la revendication 2 dans lequel le fonctionnement du système est sous le contrôle d'une unité de commande (60) comportant un microprocesseur et des mémoires morte et vive associées.

4. Système de visualisation de données selon l'une quelconque des revendications 1, 2 ou 3 dans lequel le fonctionnement de la mémoire tampon de masques est simulé dans un processeur de données.

5. Système de visualisation de données selon la revendication 1, caractérisé de plus en ce qu'il comporte une mémoire d'ordres de tracé (50) pour recevoir des ordres concernant l'information à visualiser sur l'écran de visualisation et un formateur (52) pour convertir l'information d'ordre de tracé d'un vecteur sous la forme d'un balayage de trame.

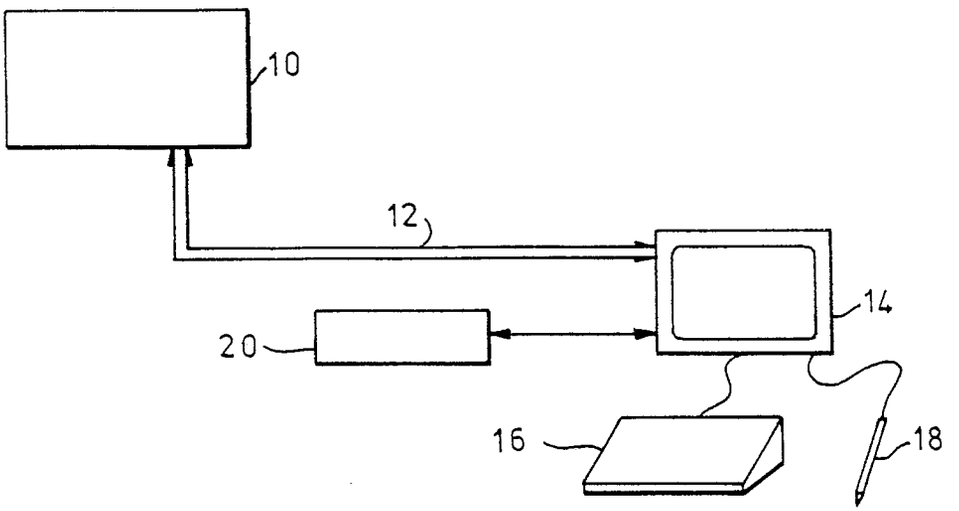

FIG. 1

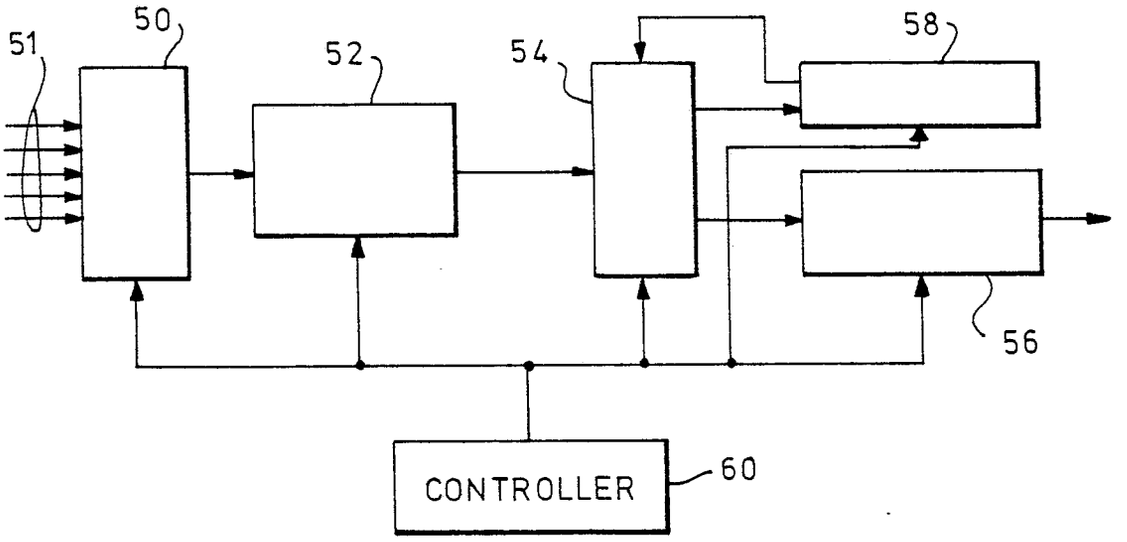

FIG. 5

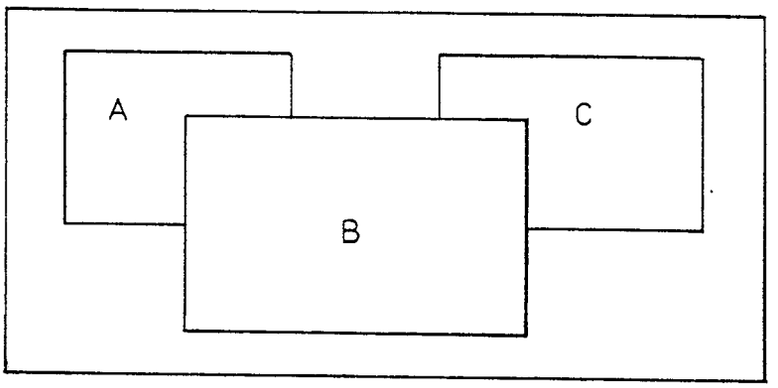

FIG. 2

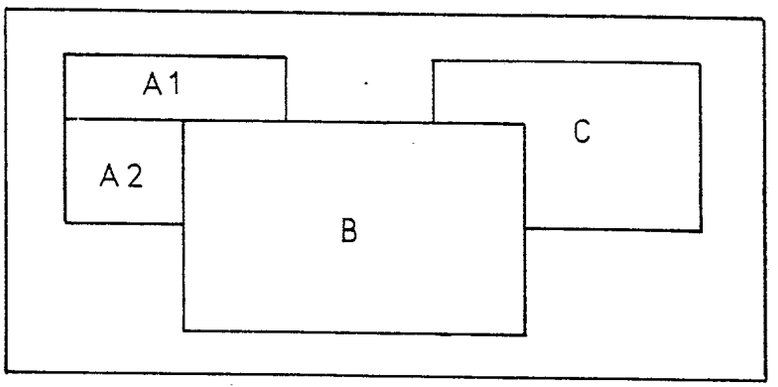

FIG. 3

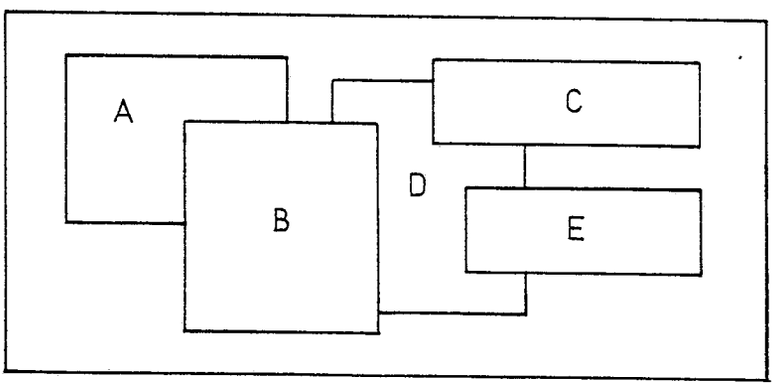

FIG. 4

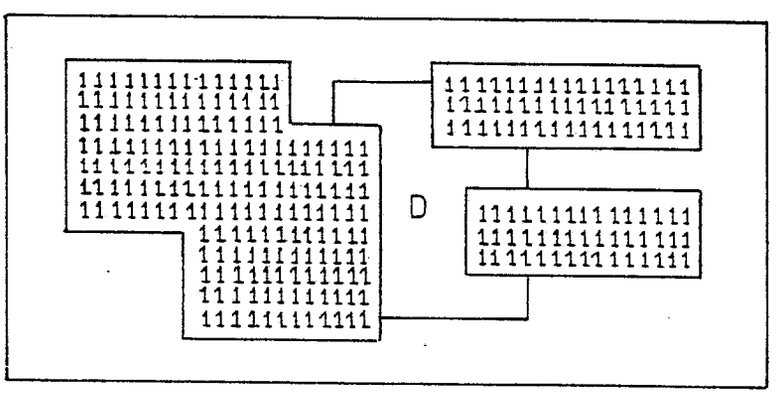

FIG. 6