COMPUTERISATION OF THE GRAPHICS SECTION OF THE FCDA PRINTING UNIT, ABUJA

BY

SAMUEL, JOSHUA ISSA
PGD/MCS/2001/1096

DEPARTMENT OF MATHEMATICS/COMPUTER SCIENCE

FEDERAL UNIVERSITY OF TECHNOLOGY, MINNA, NIGER STATE

NOVEMBER 2003

COMPUTERISATION OF THE GRAPHICS SECTION OF THE FCDA PRINTING UNIT, ABUJA

BY

SAMUEL, JOSHUA ISSA
PGD/MCS/2001/1096

A PROJECT SUBMITTED TO THE DEPARTMENT OF MATHEMATICS/COMPUTER SCIENCE IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF POST GRADUATE DIPLOMA IN COMPUTER SCIENCE

FEDERAL UNIVERSITY OF TECHNOLOGY, MINNA,
NIGER STATE

NOVEMBER 2003

CERTIFICATION

This is to certify that this work carried – out by Joshua Samuel Issa (PGD/MCS/2001/2002/1096) of the Department of Maths/Computer science, Federal University of Technology, Minna, meets the requirement for the award of Post Graduate Diploma in Computer Science.

....

Mal. Abubakar Y. U **Date**

....

L. N. Ezeako **Date**

(Head of Department)

....

(External Examiner) Date

DEDICATION

This project is dedicated to my kids, Thomas, Dorcas and Paul.

ACKNOWLEDGEMENT

My sincere appreciation goes to the Almighty god for His guidance, protection and for seeing me through the duration of the course.

I wish to acknowledge the profound effort of my supervisor, Mal Y. U. Abubarkar for taking the pains to go through the project work. The Head of Department, Mr. L. N. Ezeako, the course Co-ordinator, Mal. Isah Audu. I equally appreciate the efforts my lectures like Prof. K. Adeboye, Mal Hakimi, Dr. Yomi Aiyesimi, Mr. Badmus and Mr. Victor.

The inspiration and relentless efforts of my Beloved Brother, Elijah Samuel and his friend Taiwo Obafemi is worthy of commendation.

My bosom friend Arch Adamu Jagaba and his family, Mr. Yaaba B. Nmadu and Sister Florence Tsado are also appreciated for their love, support and kindness.

Finally, My Darling wife, Mrs. Esther Samuel deserves a big commendation for her care, support and understanding.

I also wish to thank those who have contributed to this work whom time and space did not permit me to mention.

TABLE OF CONTENTS

CHAPTER ONE

1.1. GENERAL INTRODUCTION

Computer is an electronic device that accepts data in form of input, process the data and returns the data as output. The overall idea of computerisation is to assist in recording, filling and communicating data and information. This engenders efficiency, speeds up record handling quick response to the needs of user that might have been used to the manual system.

Introducing computer in the office can also be described as office automation. This term is derived from the application of computer and other electronic office equipments and processing techniques to tasks normally associated with office work. There is a variety of human dominated tasks in either a business or government environment which traditionally are done by specially trained staff. One such activity is typing; using a conventional typewriter that requires a high degree of hand and eyes co-ordination in order to achieve a reasonable rate of output. Error correction in typed text is time consuming and laborious.

In order to achieve a high degree of efficiency, the computer comes in.

In the field of communication, techniuques such as E-mail (Electric Mail) have made conventional system of inter office memoranda obsolete. Facsimile reproduction of documents is yet another advancement which has reduced the task of physical handling of documents. In using the computer in the office, or computerizing an office, some of the mode of operation that are normally changed

are word processing, communicating facilities, record keeping and some managerial tasks such as scheduling, project management etc. Succintly, computerisation of the office is the use of technology to increase productivity and achieve goals in the office environment.

1.2. OBJECTIVES OF THE STUDY

1. To find out the problems associated with the production of designs manually.
2. To analyze the processes in the production of designs. Consideration is given to the following areas to further elucidate this point.
3. To study the existing procedures of design, from conceptualization to finishing.
4. To outline the modern system of design, the configurations, cost analysis and benefits.
5. Recommend a suitable system that can engender time-saving, higher productivity, proper record keeping and referencing.

1.3. MINISTRY OF THE FEDERAL CAPITAL TERRITORY (AN OVERVIEW)

The ministry of the federal capital territory came into existence by Decree No 6 of 1979. It assumed the status of the Nation's capital on 12th December, 1989, when the seat of government from Lagos to Abuja.

1.4. SCOPE AND LIMITATIONS OF THE STUDY

The study covers the graphics section of the printing unit of the Federal Capital Development Authority. Abuja

It also covers the production of designs from conceptualization to finishing.

1.5. METHODOLOGY

The data for this study was collected principally through observation. In the drawing room, it was observed that the bulk of the designs are sketched and developed manually. The sketched designs are taken out to business centres, where other graphic artists are paid to develop and separate the designs into films.

From the interviews conducted among the staff, a lot of them complain of the time and effort it takes to produce designs, which invariably affect job orders.

Record inspection also indicated that most of the designs are not properly documented. Records of expenses for procurement of materials and other miscellaneous costs are grossly distorted and uncoordinated. In effect proper records of costs, designs etc are inadequate.

CHAPTER TWO

LITERATURE REVIEW

2.0. INTRODUCTION

Donald D. Spencer (1992) in his book "Information Processing (3rd Edition) noted that some organizations have different reasons for computerisation, which mostly depend on the nature of their jobs or activities.

The major reasons are:

(i) It is a working tool for management information system.

(ii) It handles personal activities and records as well as the organisation's services.

(iii) The cost per unit of output is less if we are to compare it with the manual system.

(iv) The computer's degree of speed and accuracy in the processing of data is beyond human capability.

(v) The computer can work out optimum solution to a problem.

Computerisation can also be considered as Manual Procedure, computerized procedure and mechanical procedure.

Edward (1981) in his write up " Computer and Society", defined manual procedure as a system where all organizational recordings are done by hand and stored in the database management (usually in cabinets). Computerised procedure is a system that makes use of electronic devices for collecting, analyzing, recording, storing and retrieval of data.

Robert H. Bilismer and Ronald H. Alder (1980) noted that computerisation of office proceedings, will eliminate the previous manual and mechanical system such as: -

- Misfiling or loss of records, personal turnover etc.
- Duplication of numbers.
- Misspelling of names.
- Calculation of numbers of applications on monthly basis.
- Entry of data in wrong places.

In the Printers Digest (Dec' 2002), the importance of computerisation is addressed under the topic "Printing in Nigeria Today". It reads "customers of printing firms do not care whether the print-run is short or long – they just want the job well done. They expect quality printed materials that can only be guaranteed by digital computer". From the foregoing, it is clear that computerisation ensures the production of sharp, clear and high quality jobs.

2.1. IMPACT OF COMPUTERISATION

Robert H. Bilismer and Ronald H. Aider (1980) in their book "Working with Computers" pointed out some advantages or benefits of computerisation. The main thrust of their points is to increase efficiency and effectiveness.

Every organisation succeeds as a result of proper record-keeping. The use of computer database system and Computer Aided Design (CAD) not only ensure proper and accurate record keeping, but also ensures the security of the data and it's integrity.

To keep abreast of time in organizational activities one needs a "helper" that can ensure prompt release of information when needed. This "helper" can only be found in a computer.

The computer has the ability to store large quantity of data in a large area. It can as well maintain very old records and quickly access

them. Computers are said to assist man in his business and many other walks of life. They reduce complicated problem (s) to simple level.

In addition, computers do not suffer human tiredness and loss of concentration as compared to manual method.

The aforementioned writers have clearly shown the impact of computerisation of organisation's operations; which include the improvement in quality and the acceleration of the speed of services rendered to people.

2.2. GEOGRAPHICAL LOCATION OF THE F. C. T.

The Federal Capital territory, Abuja is located at the centre of Nigeria. It's boarded in the North by Kaduna State, in the East and South East by Nasarawa state, in the West by Niger State and South West by Kogi State.

2.3. GRAPHICS - ART

Graphics art is the art of reproducing symbols, images and texts. It combines the basic design elements like lines, texture, shapes, contrast and texts to produce images that can be reproduced on paper, wall, board and several other background. The graphics art section undertakes all the design aspect of the printing unit.

2.4. OBJECTIVES OF THE PRINTING UNIT

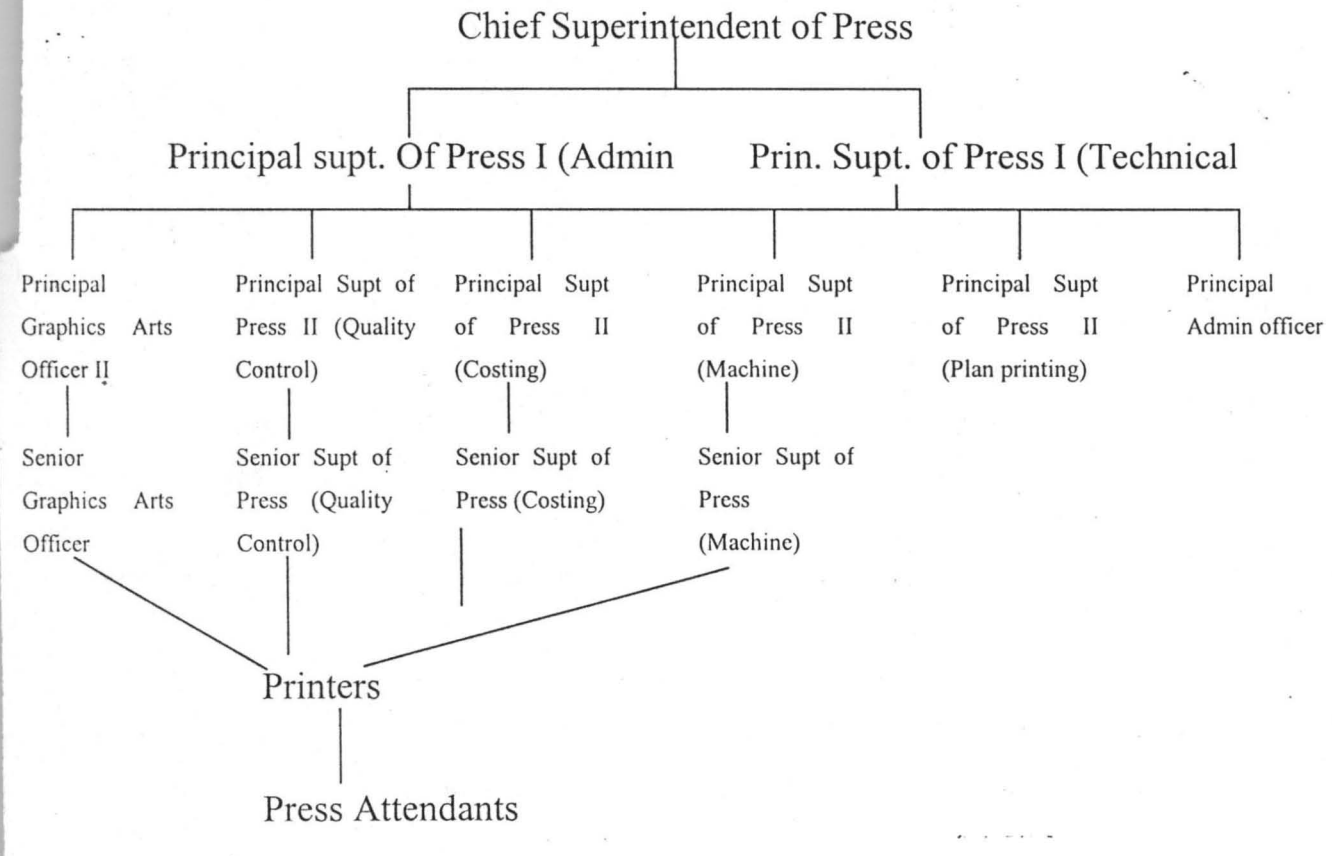
The FCDA Printing unit was primarily set up to undertake the printing of stationeries like file jackets, letter heading, office registers; departmental reports, seminar/workshop materials and security

documents like Identification cards for staff, land forms, Certificate of Occupancy forms, Departmental Nominal rolls etc.

2.6. ORGANISATIONAL STRUCTURE OF THE PRINTING UNIT

The Printing Unit is headed by a Chief Superintendent of Press who is the Production Officer. He is assisted by two Principal Superintendent of Press I that assist in overseeing the running of the unit. Each of the six sub-units is headed by a Principal Superintendent of Press. Others are Printers and Press Attendants.

2.7. ORGANIZATIONAL CHART OF THE PRINTING UNIT.



CHAPTER THREE

SYSTEMS ANALYSIS AND DESIGN

3.0. REVIEW OF THE EXISTING SYSTEM

The FCDA Printing unit, designs, prints and publishes stationeries like land form, files, letter heading, staff ID cards, complimentary cards; and souvenirs like calendars, almanac, diary, seasonal cards e.t.c; and classified security documents like land C of O forms, secret file jackets.

The unit has 2 graphic artists that design the bulk of the artwork manually. Designs are produced in the following stages.

The 1st stage consist of conceptualization of the design. For instance for a sallah seasonal card, the graphic artist creates appropriate or related images for the cover of the card.

The second stage consists of creating 2-3 designs for considerations by the Senior Graphic Artist, who passes the designs to the head of printing for the next action.

The designs are then subjected to criticism by the head of unit and a few members of staff. When a particular design scales through, it is taken as the cover design for the card. The designs that were initially done in pencils can then be developed using drawing pens and tracing sheets. The design goes to the camera room for developing before, it goes to the plate makers.

The process of conceptualizing the design to the production of a plate, takes days and may extend to weeks due to the cumbersome nature of the processes.

3.1 DESIGNS OF THE NEW SYSTEM

All versions of Visual Basic are sophisticated and powerful, thus they require commensurate machines to match their capacities. Realistically, even the learning edition of Visual Basic requires around 50MB of free hard disk space.

This software was written using Visual Basic 6.0. Because it is a graphic programme that requires very high RAM. For good response, a minimum value of 64MB RAM, a fast Pentium class chip (200 MH_2) and at least 3 G3 of Hard disk is required.

The programme will run well below the mentioned 3GB, but for storage purpose, this large capacity is recommended. In addition, for good and beautiful graphics to be produced, it is inevitable to work in other packages like MS Word, Corel Draw and Page Maker.

Other graphics packages that are recommended along with the earlier mentioned ones are, Print Artist, Instant Artist and any of the numerous graphics packages available.

To get an excellent output on the screen, a good monitor of high resolution is hereby recommended. This is important in graphics designing.

Printers of varied capacities like 1120c, 1125c 5000 series are required for the print out of designs and for color separations.

In addition, scanners that has very resolution (between 300 – 600dpi) must be purchased, to add value to designs. Pictures, background, images and charts that are not in the computer can be scanned directly from other sources.

3.2. INPUT DESIGN

Graphics in Windows environment (of which V.B uses as opposed to DOS) requires the entering of some parameters. The input stage requires some property (at the property window) to be fixed in, in order to obtain desired result(s). Some of these properties are listed below.

To draw on the screen, Visual Basic tells windows what to display. Windows in turn tells the display adapter how to display the image. Thus what you can do with Visual Basic's graphics statement, depend on the driver programs which that Windows uses to control the screen and printer. However, using these driver programs is automatic. You do not have to worry about all the possible hardware combinations a user may have, as this is different from what Ms-Dos (Microsoft – Disk Operating System) programmers are used to.

When a program displays graphics under Dos, part of this program, must check what kind of graphic board is installed or whether any graphics board is installed, and the then the program must be adjusted accordingly.

However, nothing good comes free, because Windows has to do a lot to manage a graphics environment, and this forces trade-offs. For instance, unless you set Auto Redraw property to True so that visual Basic saves a copy of the object in memory, you will have to manage the redrawing of the graphics yourself. In other words, you could say that Auto Redraw controls the persistency or otherwise of graphics.

i. GRAPHICS VIA CODE

The properties Windows helps so much as it enables one to set any of its properties to the desired one. However, these sets of properties

cannot be changed. In order to manipulate graphics even at runtime, graphics via code is the way out.

ii. **COLORS**:

The first step is to decide what colors to use. If no colour specification, Visual Basic uses the foreground color of the object for all the graphics methods. There are (4) ways to specify colors.

The first way is the design time, using the palettes that show up in the properties Window.

The second way is to work directly from the hexadecimal coding at design time or run time.

The third way is to use RGB function. The syntax for this function is RGB (amount of Red, amount of Green, Amount of Blue), where the amount of color is an integer between 0 (do not blend in any of that color) to 255 (maximum amount of that color blended in)

The fourth way, the QB color function is the last means of calling color. It's syntax is given below QB color (color code).

Where color code is an interger between 0 and 15.

The colors thus function gives are summarized as follows

Code	Color	Code	Color
0	Black	8	Gray
1	Blue	9	Light Blue
2	Green	10	Light Green
3	Cyan	11	Light Cyan
4	Red	12	Light Red
5	Magenta	13	Light Magenta
6	Brown	14	Yellow
7	White	15	High Intensity
			White

Table 3:0

To set the Back color of a form at runtime, see below

Me.Back color = QB Color (Color Number)

iii. PIXEL CONTROL

Now that the color assignment is understood and the screen scale (output design) well explained, the next thing will be to turn a pixel on. The syntax for this method is

Pset (Col, Row) (Color Code)

This comes under Input device because as indicated vividly above, at least two values must be supplied i.e. column (Col) and row (Row). The third color code is an optional value to be entered if desired. After Visual Basic processes this statement, the pixel defined by that point lights up. Obviously, where that point is, depends on what scale one has chosen. For example using the default size for a form, the line Pset (3722, 2212) would turn on the center pixel on a standard 14 inch VGA screen.

After a scale mode = 3 command, however, this would cuse an overflow run time error. To use Pset, it is advisable to set Auto Redraw property to True

iv. LINES And BOXES

If you have drawn everything by plotting individual points, graphics programming could be too time – consuming Visual Basic comes with a rich supply of graphics tools, usually called Graphics Primitive. This allows one to plot such geometric figures as lines, boxes, circles, ellipse and wedges with single statements.

The line method takes the form

Line (StartCol, StartRow) – (EndCol, EndRow), Color code

The above line command gives a line connecting the two points with the given coordinates, using the color specified by color code. In addition, the Start Col, End Col, Start Row and End Row must be inputed.

v. CIRCLES ELLIPSE AND PIE CHARTS

In Visual Basic, a circle is described by its center and radius. The following fragment draws a circle of radius 0.5 units starting at the center of the screen.

```
Scale ( - 1, 1) – (1, – 1)
Circle (0,0), 0.5.
```

The circles drawing method can be converted option. This is shown below.

Circle (step) x center, Y center), radius ..... Aspect

Example: Circle (0,0) 0.5..... I x 0.1

3.3. OUTPUT DESIGN

In outputting the result of a designed graphics on the monitor screen, some features need to be set properly in the property Windows. Among these are;

(i) Auto Redraw Property

Suppose you change Auto Redraw to false while a program is running, then you clear the object by using the "clear screen" method. Whatever you might have drawn before you changed the Auto Redraw property will remain, but everything that was drawn after the switch will disappear.

(ii) Clip Controls property and paint event

As long as Auto Redraw is off, Visual Basic activates the point event each time a part of the form is newly exposed. What actually happens within the paint event in this case depends on how the clip controls property is set at design time. If the clip controls is set to true (the default) and the Auto Redraw Property is false, then Visual Basic repaints the entire object(s). If clip controls are set to false, Visual Basic repaints only the newly exposed areas.

Setting clip controls to true also create what is known as clipping region around the non-graphical controls on the form or picture box and the controls on it in memory. Windows can use this outline to speed up how painting the form works by not having to paint some parts, such as background. However, creating the clipping region takes time, so you have to balance whether the extra time (and memory) is worth the effort.

(iii) The Refresh Method

When working with graphics, one will occasionally need to use the Refresh method. Calling this method forces an immediate refresh of the form or control. Whenever, Visual Basic processes an "object". Refresh statement, it ill redraw the object immediately and generate the paint event, if the object supports this feature.

(iv) Screen Scales

Forms and picture boxes – the two commonest controls used in graphics has a default scale called twips. It is a rather strange scale that is " 1/20 of a printers point" or "1/1440 of an inch". This is a

great scale for the printer printouts, it can be less than ideal for screen display. This property is very important for appropriate display of graphics.

The default size for a form on an ordinary 14 "VGA monitor running 640 x 480 is roughly 7,485 twips long by 4,425 twips wide. Since a twip is 1/1440 of an inch when printed, this default form size is roughly 5 inches by 3 inches – if you use the print form method on a screen of 640 x 480 resolution.

The following table gives the coordinates in one column and the location of the point in the other, for a form on a monitor running 640 x 480.

Coordinates	Location
(0,0)	Top left corner
(7485,0)	Top right corner
(0,4425)	Bottom left corner
(7485,4425)	Bottom roght corner
(3742,2212)	Roughly in the center

Table 3.1 Coordinates/Location Table

Regardless of the resolution you are running, if two points have the same first coordinate, they are on the same vertical; if they have the same second coordinate, they are on the same horizontal line.

(v) Other screen scales

There are six other possible scales besides the default scale, as well as a totally flexible user defined scale that will be examined very soon. These scales are set by changing the scale mode property of the form or picture box at design or run time.

SCALE MODE	UNITS
1.	Twips (the default)
2.	Points (72 per inch)
3.	Pixels (the number of dots as reported by windows)
4.	Characters (Units default as 12 points high and 20 points wide
5.	Inches
6.	Millimeters
7.	Centimeters

Table 3.2. Scale Mode – Units

(vi) Custom Scales

The screen is normally numbered with (0,0) at the top left corner, as shown in Table 3.1. This is obviously inconvenient for draining tables, charts, graphs and other mathematical shapes. In most of these situations, you want coordinates to decrease as you move from the top to bottom and to increase as you move from left to right.

The "scale method" sets up new coordinates for forms and picture boxes that one can use in any of the graphics method. For example, scale (-320, 240) – (320 – 240) sets up a new coordinate system with the coordinates of the top left corner being (-320, 240) and the bottom right corner being (320, - 240).

After this method, the four corners are described in a clockwise order, starting from the top left: (-320, 240), (320,240), (320, -240) and (-320, -240)

In general the scale method looks like this;

Scale (Left x, Top Y) – (Right x, Bottom Y).

(vii) Graphics controls

In Visual Basic programming language, graphics can be placed under three controls:

- Form
- Picture Box
- Image Box

The main difference in these three controls is that the Image Box control is designed specifically for displaying images and not for creating new images or manipulating them. The other two controls provide drawing methods that allow one to design graphics at run time.

3.4. LOADING GRAPHICS

The methods for loading graphics on the various controls are simpler than creating graphics from scratch.

Graphics can be placed on control either at design time or at run time. To load a graphic (bitmap or icon) on a control at design time, you assign its file name to the "picture" property of the control in the properties Window. However, this procedure can't change the image displayed at run time, instead, the Load Picture ( ) function is used. The syntax is given as

Form 1. Picture = Load Picture (FileName)

The Loading function is very useful. Beautiful pictures and images from internet, books, digital camera and from software such as Print Artist can be transferred and scanned into a folder in the computer from where it could be called forth using the Load Picture.

3.5. SAVING PICTURES

For any graphics to be called from time to time, it must be saved. If an application processes the displayed image during the course of its execution, such image can be saved by employing the Save Picture statements. Its syntax is as follows

Save Picture, Picture, Filename.

To save the contents of the Picture control to a file, the file name must reflect in the same function statements.

This is written as: - Save Picture, Picture. Picture, c:\ temp. Image bmp.

On the contrary, the above command cannot be used to save the contents of a Picture Box control if the Auto Redraw Property of the control is set to false.

3.6. FILES

Graphics works are saved as files.

Visual Basic has six commands that interact directly with the underlying operating system, mimicking the usual operating system commands that handles files and drives on the computers. Files are saved in folders (similar files) and group of folders are saved in drives.

COMMAND	FUNCTIONS
Ch Drive	Changes the logged drive for the underlying operating system.
Ch Dir	Changes the default directory
Mk Dir	Makes a new directory
Rm Dir	Removes a directory
Name	Changes the name of a file from one directory to another on the same drive
Kill	Deletes a file from a disk.

Table 3.3

A file is represented as

FileName. File Extension

Example: Motorable. Txt.

(i) File Handling Functions

In Visual Basic, four functions can be found in File Handling functions. These functions are

- File copy
- File date Time
- Get Attr
- Set Attr

(ii) File Copy Function

The File Copy function copies a file from the source path to another path. It does not use the shell command to activate the underlying operating system copy routine or call the File Manager. This function takes named arguments and its syntax is

File Copy source, destination.

(iii) File Date Time Function

The File Date Time function returns the date and time that a file was created or last modified. The syntax is: File Date Time (pathname)

(iv) Get Attr Function

The Get Attr Function returns an interger. Using masking techniques to get at the individual bits; you can determine how the various attributes are set. The syntax for – this function is

Get Attr (Pathname) As Interger

For example Get Attr (File Name) = vb Read Only + vb Hidden

This hides the file and make it Read – only.

(v) Set Attr Function

The Set Attr function sets attribute information for files. Using the same bit values given in the next table, the various attributes can be changed if so desired. The syntax for this function is

Set Attr Pathname, attributes

Similarly, the symbolic constants shown in the Table can also be used. The example below would hide the file and set it as read – only.

Set Attr File Name, Vb Hidden + 11b Read Only.

Attribute	Constant	Value
Normal	Vb Normal	0
Read Only	Vb Read Only	1
Hidden	Vb Hidden	2
System	Vb System	4
Volume	Vb Volume	8
Directory	Vb directory	16
Archive	Vb Archive	32

Table 3.4.

(vi) The LOF COMMAND

If a file is open, the Visual Basic "Length of File" command LOF ( ) could be used to know the capacity of the file.

(vii) File systems controls

Visual Basic File System Controls is a powerful tool. It allows the user to select a new drive; see the hierarchical directory structure of a disk or see the names of the files in a given directory. As with all

Visual Basic controls, there's the need to write the code to take full advantage of the file system controls.

(viii) File List Boxes

A file list box defaults to displaying the file in the current directory. As with any list box, one can control the position, size, colour and font characteristics at design time or via code. Most of the properties here are identical to those of ordinary list boxes. For example, as with all boxes, when the number of items can't fit the current size of the control, Visual Basic automatically adds vertical scroll bars. This allows the user to move through the list of files using the scroll bars. It's important to note that under Windows convention, a file can only be chosen by double clicking and not single clicking. In addition, by using an arrow key to move through a file list box, one could call any click procedure that has been written.

3.7. COST BENEFIT ANALYSIS

The change over to computerisation will really bring a lot of benefits. Apart from saving time, productivity is greatly enhanced. It is important to give a break down of this cost benefit analysis, to further illuminate this section.

Table 3.5. Development Cost

Equipments (Hardware & Accessories)	₦ 200,000.00
Software (Price & Installation)	₦ 160,000.00
Personnel Training	₦ 65,000.00
Total	**₦ 425,000.00**

Table 3.6.

System operating cost (1 YEAR)

Equipment Maintainance	₦ 95,000.00
Utilities i.e. Lights etc	₦ 20,000.00
Supplies	₦ 60,000.00
Miscellaneous	₦ 28,000.00
Total	**₦ 203,000.00**

Total cost, i.e. Development Cost + System Operating Cost = ₦ 628,000.00

Table 3.7. Visible system benefits

Saving from employing a copy artist	₦ 182,000.00
Saving from employing a lithographer	₦ 156,000.00
Operating savings	₦ 142,000.00
	₦ 480,000.00

From the above a total of ₦ 628,000.00 was spent on the Total Cost. Though the visible benefits is ₦ 480,000.00, there are other intrinsic benefits which will go a long way in improving the image, productivity and effectiveness of the unit.

These benefits can be summarized thus: Accuracy of designs will be assured as sharper and clearer images are produced. The various stages of designing, from conceptualization to color separation is done in the new system.

Designs can also be generated and produced at the shortest time. Targets in production can be set and achieved.

Varieties of designs can also be produced easily, because the software are fully loaded with varied data. Hardwares like scanners will also assist in lifting images from external sources.

Proper records of jobs executed can be stored in files and can be retrieved easily when needed.

Designs will be properly stored to guide against plagiarism.

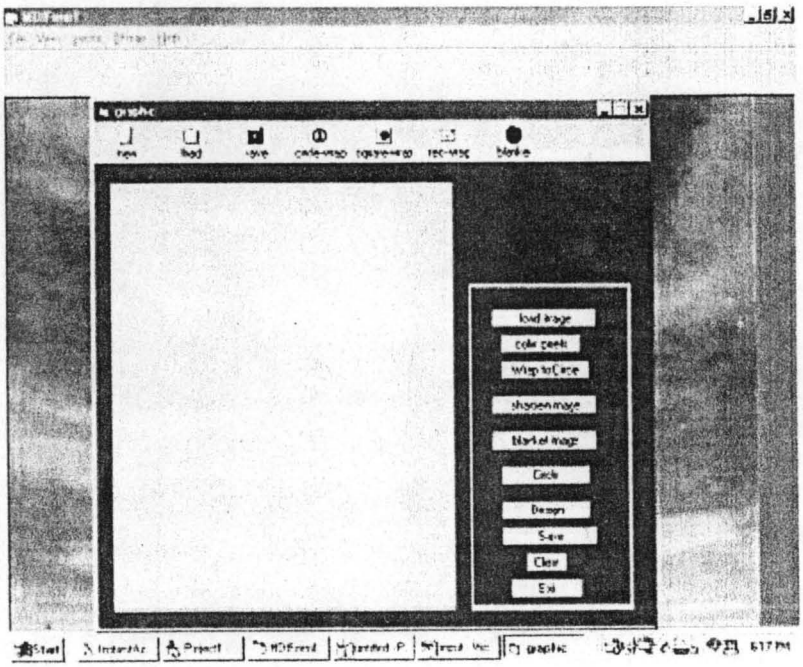
CHAPTER FOUR

4.0. SYSTEM IMPLEMENTATION

The concept of this project is to simulate a graphic display of images and drawings. The program is basically divided into four modules.

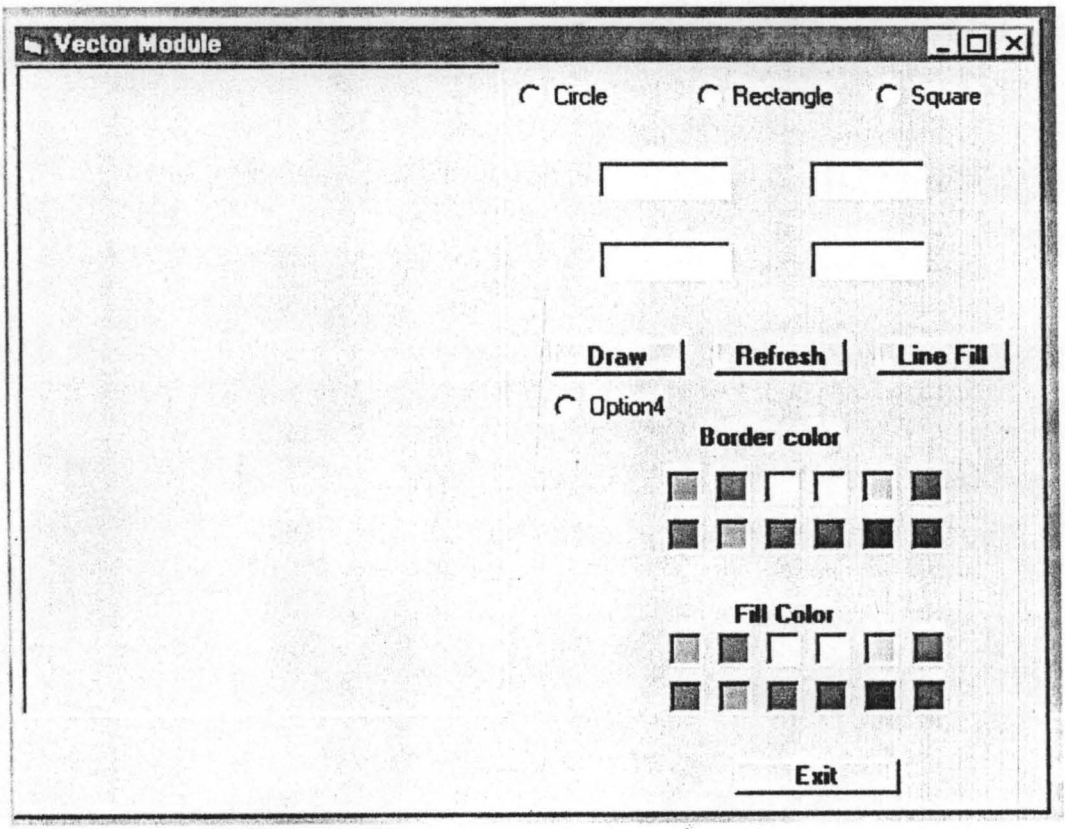
The first module is the Bitmap module, which is used to manipulate images. Images can be loaded into the Bitmap User Interface. The available tools include the load image tool that is used for loading images into the image box of the interface. The wrap tool is used to wrap images into a circular form. The sharpen image tool is used to sharpen blurred images. There's also the blanket image tool for covering an image with white ink. The circle and design tools can be used to draw circles on any image and the save tool enables the user to save the image in a folder inside the program.

BITMAP SCREEN



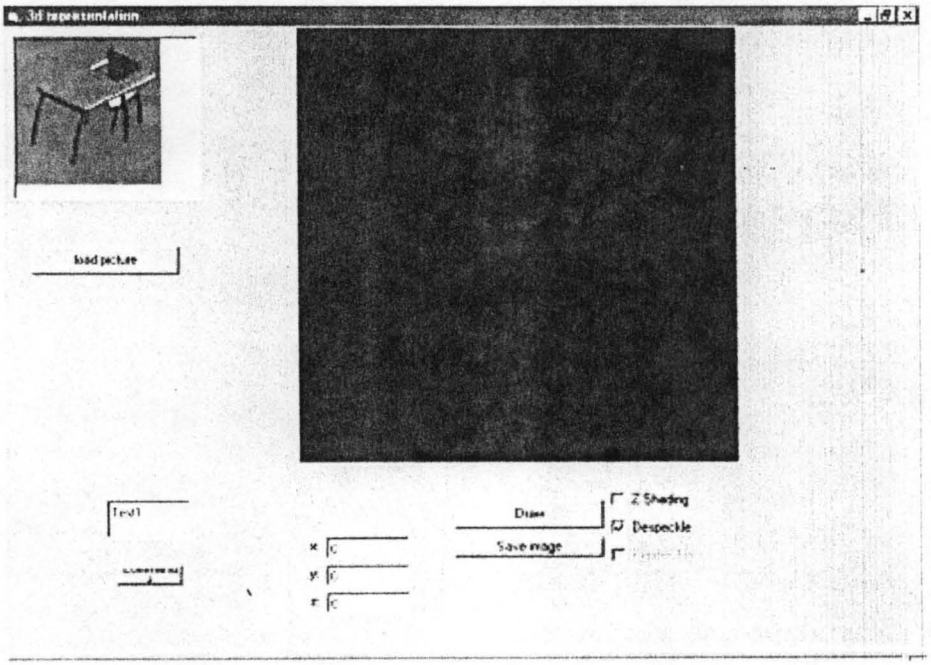
The vector module is used for drawings. It can be used to draw circles, rectangles, squares, arcs etc. these shapes can be drawn anywhere on the image box based on the user's specification. The user has to specify the position of the shape horizontally and vertically and the size of the shape before the required shape can be drawn. Drawn shapes can be filled with the desired colours.

VECTOR SCREEN



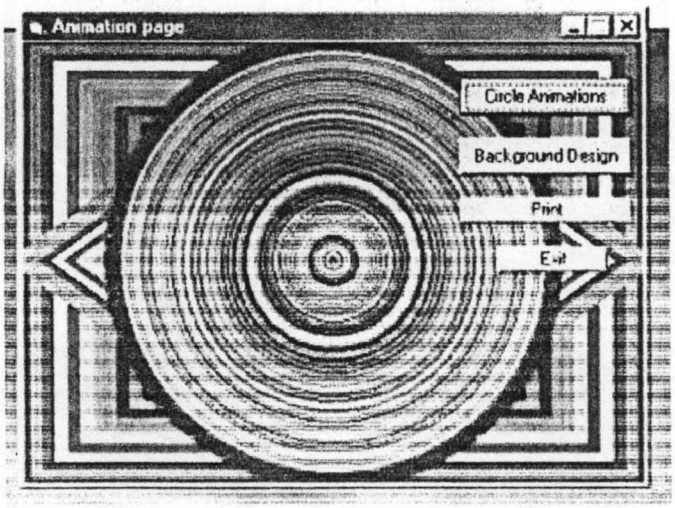
The 3D image representation module uses the concept of matrices to represent images in polar coordinates. The user specifies the X, Y and Z coordinates and the required image is represented in that format.

3D REPRESENTATION SCREEN



The Animation module displays simple background animations, which can be printed or saved.

ANIMATION SCREEN



4.1. INTRODUCTORY REMARKS ON THE PROGRAMMING LANGUAGE

Graphical user Interface, or GUI, have revolutionalized the micro computer industry. They demonstrate the substance of the saying that "a picture is worth a thousand words" to most computer users. Instead of the cryptic C:> prompt that Dos users have long seen, one is presented with a desktop filled with icons and with programs that uses mouse and menus.

One of the most important part of the programming is its graphics elements.

A program is defined as the step-by-step instruction that a computer uses to solve a problem. Among the programming languages around, Visual Basic is the most flexible; easy to understand and a user-friendly programming language.

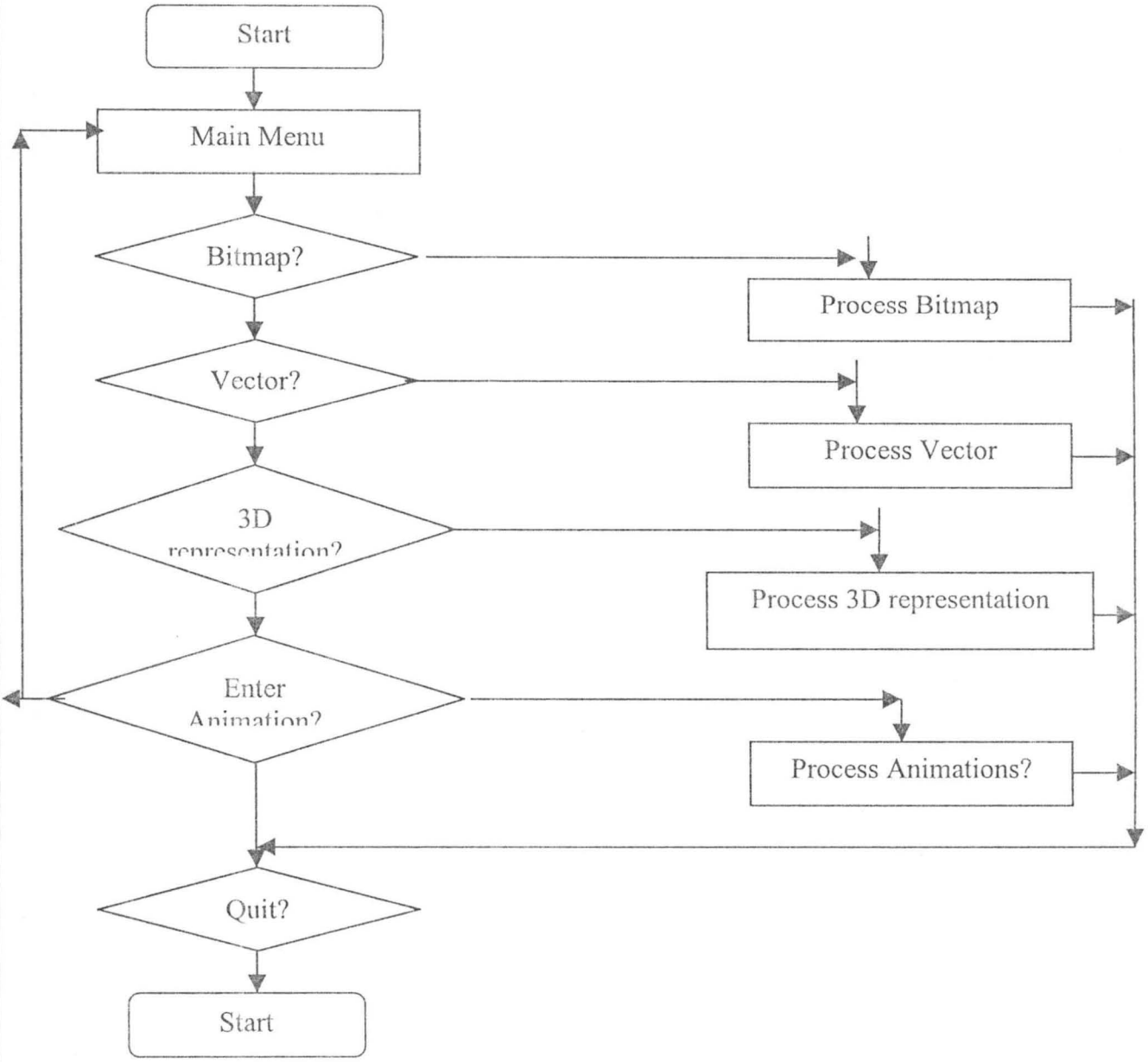
Visual Basic has been selected as the programming language for this project, because the Windows is a graphically based environment. Computer graphics (especially 3 Dimensional drawings) is a subject in which mathematics must inevitably rear its head, most especially trigonometry. The basis of all graphics are lines and shapes (such as circles and rectangles).

In general, graphics fall into two major categories; vector and bitmap. Vector graphics are images generated by graphics commands such as the line and circle commands. Bitmap graphics are images that can be displayed on various controls and processed on a pixel – by pixel basis.

4.2. ALGORITHM OF THE PROGRAM

1. Import the desired image (s)/graphics or
2. Decide which method of design to choose
3. If vector technique is chosen, enter the necessary parameters such as radius for a circle, start and end points for a line.
4. If Bitmap technique is chosen, adjust the pixel
5. Apply the desired colour, line, thickness etc.
6. If it's okay, check the output graphics.
7. If unsatisfactory, make necessary adjustment.
8. When satisfactory result is obtained, then save and/or print.

4.3. PROGRAM FLOW CHART



4.4. PROGRAM INTERFACE

To run the program, install into a computer. To install the written software, the following steps should be followed.

- Insert the CD labelled Graphic simulation into the CD-ROM drive.
- Double click on the folder labeled Graphic simulation.
- Next, double click on set-up

The last step will automatically install the program.

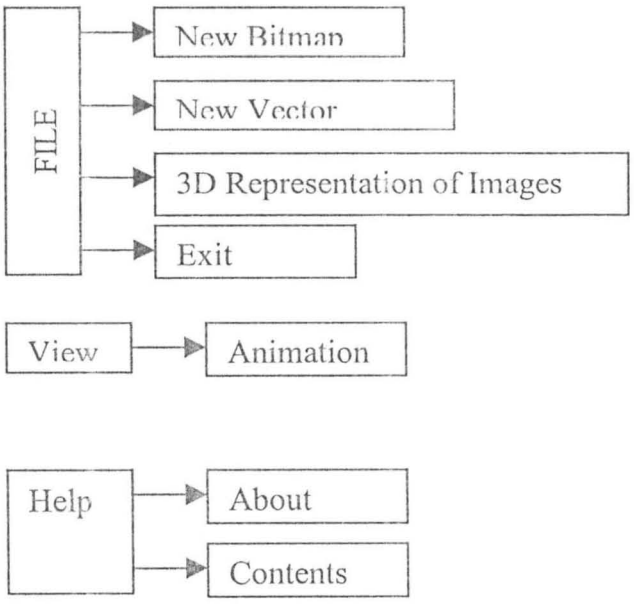
To run the program, it is important to follow these steps,

- Click on the "start" button
- Click on Program
- Locate and click on Graphic Simulation from the pop – up menu.

This last action will display a screen with the following at the top left hand corner of the screen.

File, View, Help.

The menus and their contents are illustrated below.



4.5. PROGRAM DOCUMENTATION

The aim of documentation is to guarantee the system's operational status and to also give a description of how the program works. The purpose is for easy reference in future. It could be internal and external.

The external documentation enables any other person apart from the writer of the program to understand the instruction to follow to be able to run the program successfully. It also comes handy in case the user encounters any problem in the application of the program.

Documentation is made up of various parts, the notable ones are:

- The program specification, which includes the description of the variables of the program in addition to the brief synopsis of the overall system that shows how the program fit into each other.
- Program listing, which contains the source codes (the source language)
- The operating instructions, which specify the series of operating instructions encoded.

It also includes other documentation that can be of use for the smooth running of the written program. The content of chapter 4 forms the bulk of the documentation.

4.6. PROGRAM OUTPUT/RESULTS/DISCUSSION

This project is limited in certain ways. The program cannot handle speed issues effectively. Certain image manipulations takes longer time to be accomplished, this is because the language used is not very efficient in handling large graphic manipulations.

Secondly, free hard drawing is not implemented in the program. For instance in the vector module, when a shape is to be drawn, the user has to specify the position and size in numerical values before the image can be drawn as shown in fig. 4.1. It is not possible to draw a shape without a given numerical value for it's position and size. The advantage is that, it ensures accuracy of the shapes to be drawn.

In additional, it is not possible to move an already drawn image. Once an image is drawn, it remains unless you erase and redraw it. Thus the drawings are inflexible.

However, with the addition of other packages like Print Artist, beautiful cover designs for seasonal greetings can be produced, as shown in fig: 4.2 and 4.3.

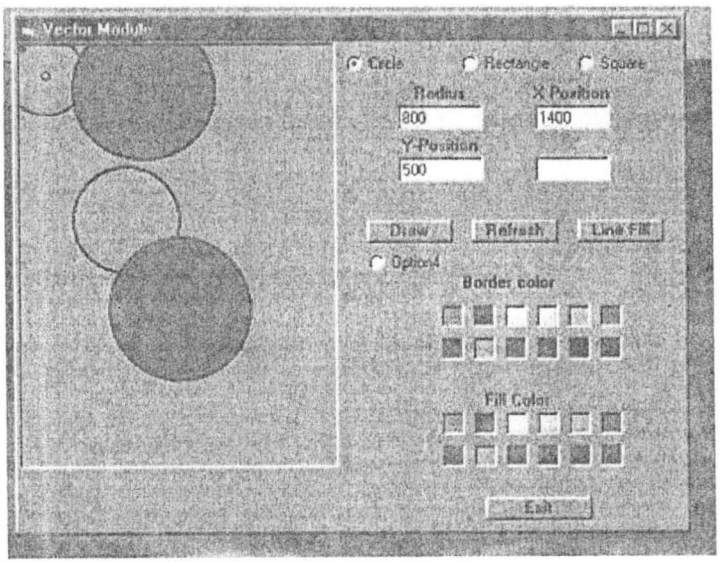


Fig. 4.1. Vector Output from graphics software

Fig. 4.2 Season's greetings- an output from the graphics software

Fig. 4.3. Season's greetings – an output from the graphics software

4.7. TRAINING

The implementation of the system must commence with having the required number of staff that will operate the new system. The system designer handles the coding of the data into the computer system before training is initiated. The training of the staff will be handled by the system designer or the organisation that has been contracted to handle the computerisation.

The advantage of this is that a uniform approach will be evolved and the system designer will be assured that the staff attain the level of proficiency required for the running of the new system. When the staff that will operate the new system have been properly trained, then a meeting can be arranged between the management and the system designer on the modalities for conversion/changeover.

4.8. CONVERSION

This is the conversion of the old file data into the form required by the new system. It is also regarded as part of the changeover. For this aspect to take place, there are some cogent things to consider;

1) The system must have been proved to the satisfaction of the system analyst to be okay and the other implementation activities must have been completed.
2) The user manager must be satisfied with the results of the systems tests, staff training and the reference manuals.
3) The target date for changeover is fixed when the above points have be actualized. The conversion can then be carried out in any of the following ways.

With extended system test, this method is not as disruptive as the parallel running.

4.9. MAINTENANCE

Whichever technique is selected for the changeover to the new system, a high priority must be given to establishing controls, by value or quantity inorder to maintain the qualitative integrity of the system. Users should keep overall control records incorporating both computer and clerical control figures to prove that the changeover has not corrupted this integrity. One of the reasons for changing into a new system is to promote efficiency

Thus the computer system must be maintained as at when due so that it can render optimum service. Most of the time, the maintenance of a system is handled by the system developer with other members of his team.

CHAPTER FIVE

5.0. SUMMARY

This chapter attempts to give a review of the preceeding chapters. Chapter one opens with the general introduction in which the concepts of computer and computerisation are outlined. The importance of computerisation, the objectives of the study, its scope and limitations and the methodology used for the study are clearly stated here.

The second chapter takes care of the reviews of related literature. The impact of computerisation, the geographical location of the FCT, the objectives of the printing unit and the organizational structure of the printing unit are given prominence in this chapter.

Chapter three is titled Systems analysis and design and it opens with the reviews of the existing system. This dovetails into the design of the new system, the input design, output design, files The cost benefit analysis is the last section of this chapter.

Chapter four gives an account of the System Implementation and introductory remarks on the programming language. The algorithm and program flowchart comes next. The program interface, documentation, output results and discussions are also discussed here. The various changeover procedures are clearly spelt out, for the implementation of the program.

Therefore, this chapter provides a clear picture of the entire study, outlining the various sections that can be found in the thesis.

5.1. CONCLUSION

In conclusion, the computerisation of any organisation is a function of; the volume of activities on ground for processing, the availability

of resources to effect and enhance computerisation, and the cost benefit analysis of the activities and resources available. The consideration of the above is of immense importance.

It is therefore pertinent that to achieve an effective, efficient and less time-consuming execution of jobs, graphic artists should be knowledgeable in graphical softwares like Adobe, CorelDraw, Ventura, Photoshop etc.

5.2. RECOMMENDATIONS

To underscore the relevance of computerisation, the department need to go the whole hug in embracing the concept, inorder to move the printing section into the modern age of technological advancement. Computer hardware and software should be acquired to meet up with these challenges.

Members of staff of the section should be encouraged to go for induction courses, workshops and seminars on computer education. Those that are qualified to attend certificate, diploma and post-graduate programmes should be sponsored.

Finally, substantial amount of money should be devoted to the computerisation exercise and subsequent upgrading of the systems. The head of the section should be in constant touch with happenings in the computer world. He should develop interest in journals, magazines, radio/TV programmes on computers.

References

(1) Donald D. Spencer "Information Processing" (3rd Edition) 1992

(2) Edward J. Laune "Computer and Society" (5th Edition) 1981

(3) Robert H. Billismer and Ronald H. Alder, "Working with Computers" D. P. Publishing London, 1980.

(4) Printers Digest (Dec 2002).

(5) Simpson, A. "Understanding Dbase III" Syber U. S. A. 1985.

(6) Ayo, C. K. "Computer Literacy" (Operations and Appreciations) Allanukitan Commercial Press (Nig) Ltd, Kogi State, 1994.

```
Tmp& = StretchBlt(bitmap.hDC, Wid(a, 0), Wid(a, 1), Wid(a, 2), 2, Store.hDC, 0, a,
Store.ScaleWidth, 2, SRCCOPY)
Next
bitmap.Refresh
End Sub

Private Sub Command10_Click()
Dim H, W, N, D, Deg, P, pi
pi = 4 * Atn(1): pi = (pi / 180)
Store.ScaleMode = 3: Store.ForeColor = RGB(255, 255, 0): Form1.DrawWidth = 4
CX = Store.ScaleWidth / 2: CY = (Store.ScaleHeight / 2) + 10
W = 126: H = 120: D = 360: N = 0
CurrentX = W * Sin(1 * pi) + CX: CurrentY = H * Cos(1 * pi) + CY
For a = 0 To D
Store.Line -(W * Sin(a * pi) + CX, H * Cos(a * pi) + CY)
Cir(a, 0) = a: Cir(a, 1) = W * Sin(a * pi) + CX: Cir(a, 2) = H * Cos(a * pi) + CY
Next
End Sub

Private Sub Command11_Click()
CurrentX = Cir(1, 1): CurrentY = Cir(1, 2)
Form1.ForeColor = RGB(0, 0, 255): Store.DrawWidth = 4
For a = 1 To 360
Store.Line -(Cir(a, 1), Cir(a, 2)): Next

Form1.ForeColor = RGB(255, 0, 0): Form1.FillColor = RGB(0, 255, 0)
Store.Line (Cir(1, 1), Cir(1, 2))-(Cir(144, 1), Cir(144, 2))
Store.Line (Cir(144, 1), Cir(144, 2))-(Cir(288, 1), Cir(288, 2))
Store.Line (Cir(288, 1), Cir(288, 2))-(Cir(72, 1), Cir(72, 2))
Store.Line (Cir(72, 1), Cir(72, 2))-(Cir(216, 1), Cir(216, 2))
Store.Line (Cir(216, 1), Cir(216, 2))-(Cir(1, 1), Cir(1, 2))
Circle (CX, CY), 16, RGB(255, 255, 0): c = 0
End Sub

Private Sub Command2_Click()
Rem Clear the screen
Store.Cls
End Sub

Private Sub Command3_Click()
Command3.Caption = "please wait..."
Dim pixel As Long
Dim IMAGEPIXELS(3, 800, 800) As Integer
Dim I As Integer, J As Integer
Dim RED As Integer, GREEN As Integer, BLUE As Integer
x = Store.ScaleWidth
```

```
y = Store.ScaleHeight
For I = 2 To y - 1
For J = 2 To x - 1
pixel = Store.Point(J, I)
RED = pixel& Mod 256
GREEN = ((pixel And &HFF00) / 256&) Mod 256&
BLUE = (pixel And &HF0000) / 65536
Store.PSet (J, I), RGB(RED, GREEN, BLUE)
Command3.Caption = "Colour Pixels"
Next
Next
End Sub
Private Sub Command4_Click()
Dim I As Integer, J As Integer
Dim RED As Integer, GREEN As Integer, BLUE As Integer
Dim pixel As Long
Dim picturename As String
ReDim IMAGEPIXELS(3, 800, 800) As Integer

   CommonDialog1.InitDir = App.Path
   CommonDialog1.Filter = "images|*.bmp;*.gif;*.jpg|all files|*.*"
   CommonDialog1.Action = 1
   picturename = CommonDialog1.FileName
   If picturename <> "" Then
   Store.picture = LoadPicture(picturename)
   x = Store.ScaleWidth
   y = Store.ScaleHeight
   If picture.Height > y Or picture.Width > x Then
   MsgBox "PICTURE IS TOO LARGE"
   End If

 If x > 8000 Or y > 8000 Then
   MsgBox "image too large to process.pls load a smaller image"
   x = 0
   y = 0
   Exit Sub
  End If

   Form22.ProgressBar1.Value = I * 40 / (y - 1)
   For I = 0 To y - 1
   For J = 0 To x - 1
   pixel = Store.Point(J, I)
   RED = pixel& Mod 256
   GREEN = ((pixel And &HFF00) / 256&) Mod 256&
   BLUE = (pixel And &HF0000) / 65536
   IMAGEPIXELS(0, I, J) = RED
```

```
   IMAGEPIXELS(1, I, J) = GREEN
   IMAGEPIXELS(2, I, J) = BLUE
      Next
      Next
   Form22.Hide
   End If
End Sub

Private Sub Command5_Click()
Dim I As Integer, J As Integer
Dim RED As Integer, GREEN As Integer, BLUE As Integer
Dim pixel As Long
Dim picture As String
Dim IMAGEPIXELS(2, 800, 800) As Integer
x = Store.ScaleWidth
y = Store.ScaleHeight
dx = 1: dy = 1
T1 = Timer
Form22.Show
Form22.Refresh
Picture1.ScaleHeight)
For I = 2 To y - 1
For J = 2 To x - 1
RED = IMAGEPIXELS(0, I, J) + 0.5 * (IMAGEPIXELS(0, I, J) - IMAGEPIXELS(0, I -
dx, J - dy))
GREEN = IMAGEPIXELS(1, I, J) + 0.5 * (IMAGEPIXELS(1, I, J) - IMAGEPIXELS(1,
I - dx, J - dy))
BLUE = IMAGEPIXELS(2, I, J) + 0.5 * (IMAGEPIXELS(2, I, J) - IMAGEPIXELS(2, I
- dx, J - dy))
If RED > 255 Then RED = 255
If RED < 0 Then RED = 0
If GREEN > 255 Then GREEN = 255
If GREEN < 0 Then GREEN = 0
If BLUE > 255 Then BLUE = 255
If BLUE < 0 Then BLUE = 0
Next
Form22.ProgressBar1.Value = I * 40 / (y - 1)
DoEvents
Next
Form22.Hide
1, &HCC0020
Store.Refresh
MsgBox "processing completed in " & Format(Timer - T1, "##.000")
End Sub

Private Sub Command6_Click()
```

```
Dim I As Integer
Dim J As Integer
Dim RED As Integer, GREEN As Integer, BLUE As Integer
Dim IMAGEPIXELS(2, 800, 800) As Integer
x = Store.ScaleWidth
y = Store.ScaleHeight
For I = 1 To y - 2
For J = 1 To x - 2
RED = IMAGEPIXELS(0, I, J)
GREEN = IMAGEPIXELS(1, I, J)
BLUE = IMAGEPIXELS(2, I, J)
If ((RED > 120) Or (RED < 255)) Then
RED = 255 - RED
'End If
If ((GREEN > 120) Or (GREEN < 255)) Then
GREEN = 255 - GREEN
'End If
If ((BLUE > 120) Or (BLUE < 255)) Then
BLUE = 255 - BLUE
Store.PSet (J, I), RGB(RED, GREEN, BLUE)
End If
End If
End If
Next
DoEvents
Next
Store.Refresh
End Sub

Private Sub Command7_Click()
Unload Me
End Sub

Private Sub Command9_Click()
SavePicture Store.picture, App.Path & "\projectwork\picta.jpg"
End Sub

Private Sub Toolbar1_ButtonClick(ByVal Button As MSComctlLib.Button)
Select Case Button.Index
Case 1
Store.Cls
Case 2
Dim I As Integer, J As Integer
Dim RED As Integer, GREEN As Integer, BLUE As Integer
Dim pixel As Long
Dim picturename As String
```

```
ReDim IMAGEPIXELS(3, 800, 800) As Integer

    CommonDialog1.InitDir = App.Path
    CommonDialog1.Filter = "images|*.bmp;*.gif;*.jpg|all files|*.*"
    CommonDialog1.Action = 1
    picturename = CommonDialog1.FileName
    If picturename <> "" Then
    Store.picture = LoadPicture(picturename)
    'Form8.Refresh
    x = Store.ScaleWidth
    y = Store.ScaleHeight
    If picture.Height > y Or picture.Width > x Then
    MsgBox "PICTURE IS TOO LARGE"
    End If
    If x > 8000 Or y > 8000 Then
    MsgBox "image too large to process.pls load a smaller image"
    x = 0
    y = 0
    Exit Sub
    End If

    Form22.Show
    Form22.Refresh
    For I = 0 To y - 1
    For J = 0 To x - 1
    pixel = Store.Point(J, I)
    RED = pixel& Mod 256
    GREEN = ((pixel And &HFF00) / 256&) Mod 256&
    BLUE = (pixel And &HF0000) / 65536
    IMAGEPIXELS(0, I, J) = RED
    IMAGEPIXELS(1, I, J) = GREEN
    IMAGEPIXELS(2, I, J) = BLUE
      Next

      Next

    End If
  Case 3
  Case 4

  Case 5
  Case 6
  Case 7
  Dim Q As Integer
  Dim P As Integer
  'Dim RED As Integer, GREEN As Integer, BLUE As Integer
```

```
'Dim IMAGEPIXELS(2, 800, 800) As Integer
x = Store.ScaleWidth
y = Store.ScaleHeight
For Q = 1 To y - 2
For P = 1 To x - 2
  RED = IMAGEPIXELS(0, Q, P)
GREEN = IMAGEPIXELS(1, Q, P)
BLUE = IMAGEPIXELS(2, Q, P)
If ((RED > 120) Or (RED < 255)) Then
RED = 255 - RED
'End If
If ((GREEN > 120) Or (GREEN < 255)) Then
GREEN = 255 - GREEN
'End If
If ((BLUE > 120) Or (BLUE < 255)) Then
BLUE = 255 - BLUE
Store.PSet (P, Q), RGB(RED, GREEN, BLUE)
End If
End If
End If
Next
DoEvents
Next
Store.Refresh
End Select
End Sub

Private Type TDPoint 'this type holds a single point
x As Integer
y As Integer
z As Integer
End Type

'these are faster than pset and point
Private Declare Function GetPixel Lib "GDI32" (ByVal hDC As Long, ByVal x As
Long, ByVal y As Long) As Long
Private Declare Function SetPixel Lib "GDI32" (ByVal hDC As Long, ByVal x As Long,
ByVal y As Long, ByVal color As Long) As Long
Dim RM(0 To 3, 0 To 3) As Double 'rotation matrix
Dim PPoint(0 To 16383) As TDPoint 'all the points

Private Sub Command1_Click()
Picture2.Cls
Dim GetPoint As TDPoint
Dim color As Long
Dim cred, cgreen, cblue
```

```
Dim x As Double
Dim y As Double
Dim z As Double
Const pi = 3.14159265358979
'PointArray
If Not IsNumeric(txtx.Text) Or Not IsNumeric(txty.Text) Or Not IsNumeric(txtz.Text)
Then
   MsgBox "You must enter numerical values for x, y, and z"
   Exit Sub
End If

x = txtx.Text * pi / 180 'convert to radians
y = txty.Text * pi / 180
z = txtz.Text * pi / 180
MatrixBuild x, y, z 'build the matrix
For y = -64 To 63 'for every single point
For x = -64 To 63
GetPoint = RotatePoint(x, y, 0) 'rotate point

color = GetPixel(Picture1.hDC, x + 64, y + 64) 'get the color of it
If Shade.Value = 1 Then 'shade depending on z distance
   cred = Int(color Mod 256)
   cblue = Int(color / 65536)
   cgreen = Int((color - (cblue * 65536) - cred) / 256)
   cred = cred + GetPoint.z * 3
   cgreen = cgreen + GetPoint.z * 3
   cblue = cblue + GetPoint.z * 3
   If cred > 255 Then cred = 255
   If cred < 0 Then cred = 0
   If cgreen > 255 Then cgreen = 255
   If cgreen < 0 Then cgreen = 0
   If cblue > 255 Then cblue = 255
   If cblue < 0 Then cblue = 0
   color = RGB(cred, cgreen, cblue)
End If

SetPixel Picture2.hDC, GetPoint.x + 120, GetPoint.y + 120, color 'set the new point
Next x
Next y

If Despeckle.Value = 1 Then
For y = 1 To 160 'despeckle
For x = 1 To 160
If (GetPixel(Picture2.hDC, x, y) = RGB(0, 0, 0)) Then
   fcolor = Picture2.Point(x - 1, y)
   cred = Int(fcolor Mod 256)
```

```
    cblue = Int(fcolor / 65536)
    cgreen = Int((fcolor - (cblue * 65536) - cred) / 256)

    fcolor = GetPixel(Picture2.hDC, x, y - 1)
    CRed2 = Int(fcolor Mod 256)
    CBlue2 = Int(fcolor / 65536)
    CGreen2 = Int((fcolor - (CBlue2 * 65536) - CRed2) / 256)

    fcolor = GetPixel(Picture2.hDC, x + 1, y)
    CRed3 = Int(fcolor Mod 256)
    CBlue3 = Int(fcolor / 65536)
    CGreen3 = Int((fcolor - (CBlue3 * 65536) - CRed3) / 256)

    fcolor = GetPixel(Picture2.hDC, x, y + 1)
    CRed4 = Int(fcolor Mod 256)
    CBlue4 = Int(fcolor / 65536)
    CGreen4 = Int((fcolor - (CBlue4 * 65536) - CRed4) / 256)

    SetPixel Picture2.hDC, x, y, RGB((cred + CRed2 + CRed3 + CRed4) / 4, _
       (cgreen + CGreen2 + CGreen3 + CGreen4) / 4, _
       (cblue + CBlue2 + CBlue3 + CBlue4) / 4)
End If

Next x
Next y
End If
End Sub


Private Sub MatrixBuild(ByVal x As Double, ByVal y As Double, ByVal z As Double)
Dim SinX, CosX, SinY, CosY, SinZ, CosZ, C1, C2

SinX = Sin(x)
CosX = Cos(x)
SinY = Sin(y)
CosY = Cos(y)
SinZ = Sin(z)
CosZ = Cos(z)

RM(0, 0) = (CosZ * CosY)
RM(0, 1) = (CosZ * -SinY * -SinX + SinZ * CosX)
RM(0, 2) = (CosZ * -SinY * CosX + SinZ * SinX)
RM(1, 0) = (-SinZ * CosY)
RM(1, 1) = (-SinZ * -SinY * -SinX + CosZ * CosX)
RM(1, 2) = (-SinZ * -SinY * CosX + CosZ * SinX)
```

```
RM(2, 0) = SinY
RM(2, 1) = CosY * -SinX
RM(2, 2) = CosY * CosX
End Sub

Private Function RotatePoint(ByVal x As Integer, ByVal y As Integer, ByVal z As
Integer) As TDPoint

Dim TempPoint As TDPoint
TempPoint.x = (x * RM(0, 0)) + (y * RM(0, 1)) + (z * RM(0, 2)) + RM(0, 3)
TempPoint.y = (x * RM(1, 0)) + (y * RM(1, 1)) + (z * RM(1, 2)) + RM(1, 3)
TempPoint.z = (x * RM(2, 0)) + (y * RM(2, 1)) + (z * RM(2, 2)) + RM(2, 3)

RotatePoint = TempPoint
End Function

Private Sub Command2_Click()
CommonDialog1.Filter = "allfiles|*.*"
CommonDialog1.ShowOpen
Picture1.picture = LoadPicture(CommonDialog1.FileName)
End Sub

Private Sub Command3_Click()
SavePicture Store.picture, App.Path & "\projectwork\picta.jpg"
End Sub

Private Sub Picture1_Click()
CommonDialog1.ShowOpen
If CommonDialog1.FileName = "" Then Exit Sub
Picture1.picture = LoadPicture(CommonDialog1.FileName)

End Sub

Private Sub Command1_Click()
Dim max, maxx, maxxx As Integer
max = Val(Text1.Text)
maxx = Val(Text2.Text)
maxxx = Val(Text3.Text)
Form4.Circle (maxx, maxxx), max
Form4.Refresh
Form4.Show
Unload Me
End Sub

Dim max, maxx, maxxx As Integer
```

```
Private Sub Command1_Click()

Unload Me
End Sub

Private Sub Command2_Click()
Picture1.FillStyle = 3
Picture1.Line (max, maxx)-(maxxx, maxx), , B
Picture1.Line -(maxxx, manx)
Picture1.Line -(max, manx)
Picture1.Line -(max, maxx)
End Sub

Private Sub Command4_Click()
max = Val(Text1.Text)
maxx = Val(Text2.Text)
maxxx = Val(Text3.Text)
manx = Val(Text4.Text)
Picture1.DrawWidth = 2
If Option1.Value = True Then
Picture1.Circle (maxx, maxxx), max
ElseIf Option2.Value = True Then

Picture1.Line (max, maxx)-(maxxx, maxx), , B
Picture1.Line -(maxxx, manx)
Picture1.Line -(max, manx)
Picture1.Line -(max, maxx)
ElseIf Option3.Value = True Then
Picture1.Line (max, maxx)-(maxxx, maxx), , B
'Picture1.Circle (maxx, maxxx), max, , -1, 2 * max
End If
End Sub

Private Sub Command5_Click()
Option1.Value = False
Option2.Value = False
Option3.Value = False
Text1.Text = ""
Text2.Text = ""
Text3.Text = ""
Text4.Text = ""
Text1.Enabled = False
Text2.Enabled = False
Text3.Enabled = False
Text4.Enabled = False
Picture1.FillStyle = 1
```

```
End Sub

Private Sub Option1_Click()
Label3.Caption = "Radius"
Label4.Caption = "X Position"
Label5.Caption = "Y-Position"
Text1.Enabled = True
Text2.Enabled = True
Text3.Enabled = True
End Sub

Private Sub Option2_Click()
Label3.Caption = "X1"
Label4.Caption = "Y1"
Label5.Caption = "X2"
Label6.Caption = "Y2"

Text1.Enabled = True
Text2.Enabled = True
Text3.Enabled = True
Text4.Enabled = True
End Sub

Private Sub Option3_Click()
Label3.Caption = "X1"
Label4.Caption = "Y1"
Label5.Caption = "X2"
Label6.Caption = "Y2"

Text1.Enabled = True
Text2.Enabled = True
Text3.Enabled = True
Text4.Enabled = True
End Sub

Private Sub Picture10_Click()
max = Val(Text1.Text)
maxx = Val(Text2.Text)
maxxx = Val(Text3.Text)
manx = Val(Text4.Text)
Picture1.DrawWidth = 2
If Option1.Value = True Then
Picture1.Circle (maxx, maxxx), max, QBColor(1)
ElseIf Option2.Value = True Then
Picture1.Line (max, maxx)-(maxxx, maxx), QBColor(1), B
Picture1.Line -(maxxx, manx), QBColor(1)
```

```
Picture1.Line -(max, manx), QBColor(1)
Picture1.Line -(max, maxx), QBColor(1)
End If
End Sub

Private Sub Picture12_Click()
Picture1.FillStyle = 0
Picture1.FillColor = &H8080FF
max = Val(Text1.Text)
maxx = Val(Text2.Text)
maxxx = Val(Text3.Text)
manx = Val(Text4.Text)
Picture1.DrawWidth = 2
If Option1.Value = True Then
Picture1.Circle (maxx, maxxx), max
ElseIf Option2.Value = True Then
Picture1.Line (max, maxx)-(maxxx, maxx)
Picture1.Line -(maxxx, manx)
Picture1.Line -(max, manx)
Picture1.Line -(max, maxx)
End If
End Sub

Private Sub Picture13_Click()
Picture1.FillStyle = 0
Picture1.FillColor = &HFF&
max = Val(Text1.Text)
maxx = Val(Text2.Text)
maxxx = Val(Text3.Text)
manx = Val(Text4.Text)
Picture1.DrawWidth = 2
If Option1.Value = True Then
Picture1.Circle (maxx, maxxx), max
ElseIf Option2.Value = True Then
Picture1.Line (max, maxx)-(maxxx, maxx)
Picture1.Line -(maxxx, manx)
Picture1.Line -(max, manx)
Picture1.Line -(max, maxx)
End If
End Sub
```