Pattern Approach to Build Traditional Graphic Frameworks

Hari Ramakrishna, PhD.
Professor, Department of Computer Science and Engineering,
Chaitanya Bharathi Institute of Technology, Hyderabad INDIA -500 075

ABSTRACT
Architecture of a light weight three dimensional graphic framework model using traditional graphic methods is presented in this paper. The traditional graphics concepts are redesigned and remodeled for present requirements and development technologies. A pattern approach is adapted for presenting the model. Several pattern frames are suggested based on this model to suit new development methodologies. A pattern frame for porting traditional graphic models into new technologies is presented. A set of block diagrams and UML diagrams are used to describe the model. Lists of classified graphic functionalities to support graphic requirements of this model are listed. A sample client application of this model in VB is presented.

General Terms
Pattern, pattern-frames, pattern approach, traditional graphic model, display files, display file interpreter, graphic vector generation algorithms/functions, graphic Segments.

Keywords
Graphic frameworks, Object oriented graphic frameworks, traditional graphic frameworks, graphic pattern frame.

1. INTRODUCTION
Patterns for software development are one of the latest trends to emerge from the object oriented approach. Fundamental to any science or engineering discipline is a common vocabulary for expressing its concepts, and a language for expressing these interrelationships. The goal of patterns within the software community is to create a body of literature to help software developers resolve recurring problems encountered throughout all of software development. Patterns help create a shared language for communicating insight and experience about these problems and their solutions.

The patterns in the GOF book are Object Oriented Design Patterns. There are many other kinds of software patterns beside design patterns, analysis patterns published by Marin Fowler and other patterns like organizational patterns are also available.

Architectural patterns express a fundamental structural organization or schema for software systems. Design Patterns provide a schema for refining the subsystems or components of a software system or the relationships between them. They describe commonly recurring structure of communicating components that solve a general design problem within a particular context [1-6].

Integrative computer graphics and vector graphics are traditional domains of computer science [7, 8]. The usability of applications of this domain is vide and has proven impact in the computer science. Requirements of major software application demand graphic tools [9] [10] [11]. Several classes of graphic frameworks are available in the market. Though the graphic domain is enriched with several traditional methods for decades to fulfilling the requirements of applications still there is a need of remodeling the methods to port into new requirements and technologies.

The traditional graphic framework architecture presented in this paper presents a model to port and reconfigure the traditional graphic principles to modern requirements and into the new technologies. The pattern methodology is a proven way of documenting domain specific knowledge. The model in this thesis is named as "*traditional graphic pattern frame"* as it is domain specific and it presents a reusable light weight graphic framework useful for several applications.

The intent of Traditional graphic frameworks is to apply traditional graphic techniques for building frameworks. The guiding principle of extreme programming says that "adopt the first method that probably works to start with". Adopting traditional graphic methods to build generic modules are suggested as the first step to build the graphic frameworks.

The interactive computer graphics provide several techniques to build generic graphic applications. They allow building a graphic framework that can manage different applications. For example, display files represent a graph as a set of commands. Common operations can be performed on such graphs that are stored as a standard display file. Designing a generic display file and a set of instructions enables graphic modules work for more than one application [7, 8]. A model display file structure for building graphic frameworks is evolved and presented.

The Figure: 1 presents a model graphic framework using traditional graphic display file concepts. The model has the following major core components [7, 8].

i) Segment Table

ii) Graphic Instructions

iii) Vector Graphic Display file

iv) Display files Interpreter

These tools support design of application specific component library. Such frameworks depend on powerful graphic library supported by the compiler or vendor. C++ provides several graphic primitive libraries, while MFC provides powerful graphic environment for building such frameworks. JDK also provides graphic libraries for java for building java-based frameworks. These framework models can be implemented in any environment with mimimum graphic primitive library support.

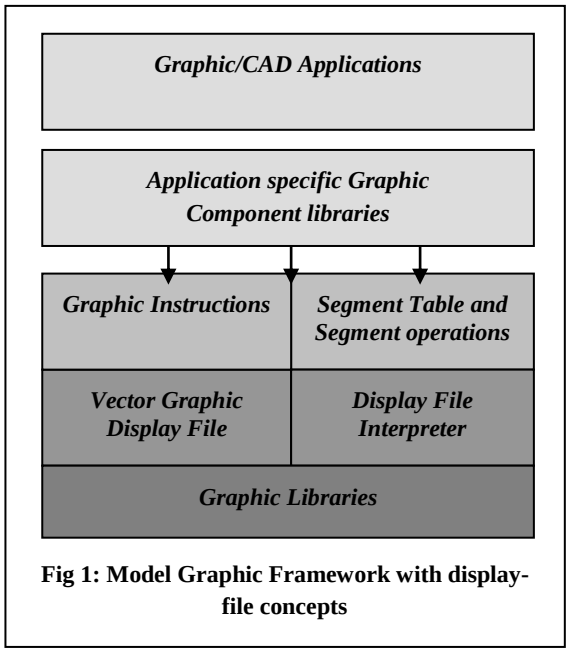

Fig 1: Model Graphic Framework with display-file concepts

2. SEGMENTATION TABLE

The display file for general-purpose interactive graphics software is divided into a set of segments such that each segment corresponds to a component of the overall display file. For example, in a building-graphics information system each civil engineering building element is treated as a segment. Windows, doors, racks etc, which are known as civil engineering building elements, are stored in the display file as graphic- segments. Sets of attributes are associated with each segment. All these attributes of segments are stored in segment-table.

Consider the information that must be associated with each segment and how the information might be organized. Each segment is given a unique name so that it can be referred with it. Perform operations on segments such as changing the visibility of segment; require some way to distinguish that segment from all other segments. When Display file segment must know which display file instructions belong to it. This may be determined by knowing where the display file instructions for that segment begin and how many of them are there in its specific display file. Each segment need some way of associating its display file position information and its attribute information with its name. The display file and its attributes can be organized in a tabular form as indicated below:

i) Segment name

ii) Segment starting address in the display file

iii) Segment size i.e. number of instructions in the display file

iv) Segment visibility i.e. on or off

v) Segment transformation parameters i.e. scaling, translation, rotation around x,y,z axes

vi) Segment reference point that is useful for transformations

vii)Segment transparency (on or off) useful for hidden line and surface elimination

Segmentation can be achieved through a set of procedures to create, open, close and transform a segment. The Table 1 presents various sample user-routines needed to handle segments:

Table 1: Graphic Segment interface operations

Create-segment (n)	Translate-segment (n,tx,ty,tz)
Close-segment (n)	Set-segment-reference-point (n,x,y,z)
Append-segment (n)	Scale-segment (n,sx,sy,sz)
Set-segment-visibility (n,I)	Show-segment (n)
Rotate-segment (n,ax,ay,az)	Delete-segment (n)

n: Segment Name	tx, ty, tz : Transformation parameters
x,y,z : coordinates	sx,sy,sz: Scaling parameters
I: visibility	ax,ay,az : Rotation parameters

3. GRAPHIC INSTRUCTIONS AND VECTOR GRAPHIC DISPLAY FILES

Graphic Instructions are used to define geometry of the graphic components in the form of a set of graphic commands. All these vector graphic commands are stored in display file. Display file interpreter will actually plot the drawing with the help of a set of graphic primitive algorithms. There are several advantages of storing drawings in the form of graphic instructions. This model allows performing operations on graphic elements such as scale, reflecting, rotating, moving etc. As all the drawings are stored in a uniform format it is easy to manage them. They will occupy less memory compared with image formats except in GIS applications. In GIS, image format will occupy less memory. This is discussed in Map object frameworks. Even in such cases display files concepts are used because this alone will allow operations on images in an effective way. This section will present a new model display file, which is useful for traditional graphic frameworks.

Considering the structure of the display file, each display file command contains two parts-operation code (*opcode*), and operands. *Opcode* indicates the type of command and operands are the required arguments such as the coordinates of the point (x, y, and z). The display file is made up of a series of these instructions. The display file must be large enough to hold all the commands needed to create the image. One must assign meaning to the possible operation code before proceeding to interpret them. Suitable geometrical elements should be provided for building graphical information system. For example graphical components of civil engineering building-graphical information system typical geometrical elements like point, line, circle, arc and polygon can be considered. Typical general attributes of a simple display file instruction, are type of the geometrical element, its color and x, y, z coordinates. The instruction is interpreted by invoking the required vector generator.

The vector generators of special geometrical elements may need more information than what is available in the main display file. This information is also in the form of graphic-commands, stored in a separate display file. For example all the instructions for plotting a polygon are in the polygon display file. Each vector generator of this type has its own interpreter for the interpretation of these commands. The starting-address and size of these instructions are the needed

attributes, which are stored in the main display file. The following Figure: 2 present a model display file structure :

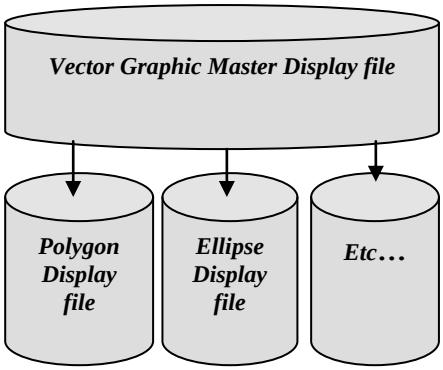

Fig 2: Model Vector Graphic display-file structure

4. DISPLAY FILE INTERPRETER

The information in the display file is useful to model the object and create the required image. The reason behind this is two-fold: some measure of device-independence is achieved, and it is easy to perform image transformation by changing the position and orientation of the required image. The display file contains the information necessary to construct the required image. The information can be in the form of instructions such as "move the pen", "draw a line", and "plot the required polygon". Saving instructions of this kind usually take much less storage than saving the picture itself. Each instruction of this kind usually takes much less storage than saving the picture itself. Each instruction indicates an action for the display device. A display file interpreter is used to convert these instructions into actual images. The display file interpreter serves as an interface between the graphics program and the display device. The display file instruction may be actually stored in a file either for a display layer or for transfer to another machine. Such files of imaging instruction are sometimes called "metafiles" The following are the sample vector-generating algorithms:

Table 2: Vector-generating algorithms

i	do-line3d (lc, bc, z, y, z),
ii	do-point3d(lc x, y, z),
iii	do-sphere(lc, cx, cy, cz, r)
iv	do-circle3d(lc, cx, cy, cz, r, ax, ay, az),
v	doarc3d(lc, cx, cy, cz, r, sa, ea, ax, ay, az),
vi	do-poly(lc, sadd, size)

In the above table : 2 lc is the line foreground color (the color type will be replaced with RGB values in windows based applications), cz,cy,cz are the coordinates, sa,ea are the starting and the ending angles, ax, ay, az are the angles of inclination along x, y, and z axes respectively, and r is the radius.

These functions are used by the display file interpreter while converting the display file instructions into the required picture on the display device. This process of generating image makes the graphics software independent of the nature of the display device and graphic application.

Whatever may be the way of storing and plotting the required images; it requires some tools for interaction with the graphics system. The following are the various sample user-routines (graphic instructions) for building-graphics information system:

Table 3: Graphic Instructions

i	Move3d (x, y, z)
ii	**Line3d(x,y,z)**
iii	Line3d(lc,x,y,z)
iv	Point3d (lc,x,y,z)
v	Arc3d(lc,x,y,z,r,sa,ea,ax,ay,az)
vi	Circle3d(lc,x,y,z,r,ax,ay,az)

Typical Functions of the model are listed in tables [4- 7].

Other frameworks namely function-class frameworks, Foundation class frameworks are used to manage the display file requirements in an efficient way using object oriented patterns[1].

5. GRAPHIC FRAMEWORKS WITH VECTOR GRAPHIC DISPLAY FILES

It is observed that the display files enable graphic developer to generate graphic structures that work for more than one application as the geometry of the domain specific graphic components can be described as set of display file instructions. Graphic user can build domain specific libraries over the existing graphic model. This will enable commencing with building graphic frameworks. Such systems can be reused in several computer applications where simulation and visualization are required. One can perform graphic operations on the defined segments to enable the graphic components participate in the simulation process of the application designed.

Such models also have requirements in some GIS and CAD based applications. Generating images for huge graphic data available in the display file is very common in GIS. The segment tables are known as named layers in GIS systems. The GIS graphic data can be divided as a set of layers. One can perform operations such as making visibility on/off on each named layer of GIS graphic data.

6. ARCHITECTURE OF TRADITIONAL GRAPHIC FRAMEWORKS

This section will discuss the architecture of a traditional graphic framework. This architecture is referred as traditional graphic pattern-frame. Pattern way of documentation is adopted for presentation of this architectural structure.

Framework name: Traditional Graphic pattern-frame

Intent: Traditional graphic techniques are redesigned for building graphic frameworks that can manage domain independent graphic system.

Motivation: Graphic applications need to manage different graphic elements depending on the type of the domain such as flowchart symbols, UML blocks, Electronic circuits, Civil engineering elements etc. The requirement of the graphic framework is to provide a mechanism to allow the client to add its own symbol library for using same graphic subsystem in different graphic domains. Such applications are available

using traditional vector graphic techniques like display files concepts.

Applicability:

i) A graphic subsystem which will be used in more than one domain

ii) A graphic application where geometry of the graphics is added at run time

iii) A graphic application or subsystem for managing many graphic elements using single objects also can adopt this technique. Flyweight object frameworks presented in this thesis also use this pattern-frame

Structure:

The architecture of this traditional graphic framework is presented in Figures 4, 5 and 6. The Figure: 4 present the structure of display file. The Figure 5 presents the structure of vector graphic sub-system. This vector graphic sub-system in tern used the display file sub-system. The Figure: 6 present overall architecture of traditional graphic frameworks.

Participants:

Display files Instructions, Graphic Operations, Display file, Segment Operations, Graphic Functions, and Segment Table

Collaboration:

i). The display file stores the geometry of a graph as a set of graphic instructions.

ii). Display File Instructions are a set of instructions useful to define the geometry of a graphic element.

iii). Graphic operations perform operations on graphs stored in display file as a set of instructions.

iv). Segment table stores additional information of graphic components. This information is used for identifying the graphic component from display file contents.

v). Segment operations are a set of functions useful in creating and managing a graphic component.

vi). Graphic functions are typical functions used in defining geometry of a graphic component. Each Graphic function defines a geometric shape as a set of display file instructions.

Consequences:

i) Graphic framework allows client applications to define the geometry of a system at run time and perform operations on it.

ii) Different domain specific Graphic components can be evolved

iii) Such graphic frameworks are independent of graphic application.

Implementation: The implementation of this framework mainly requires the following

i) Identification of display files instructions suitable to define all types of graphic components belonging to different graphic applications domains.

ii) Identification of Graphic and Segment operations.

Sample set of functions are listed in tables [4, 5, 6, and 7]. The figure 3 presents a VB client using a sample traditional graphic framework.

Table 4: Three Dimensional Graphic Libraries for Plating Typical Shapes

	Procedure Name	Description
01	void do_move3d(int x,int y,int z);	Moves to the specified point
02	void do_line3d(CDC *pDC,int lc,int bc,int x, int y,int z);	Draws a line to the given point
03	void do_point3d(CDC *pDC,int lc,int bc,int x, int y,int z)	Plots point at x,y,z with color lc
04	void do_fill3d(CDC *pDC,int lc,int bc,int x, int y,int z);	Used for filling
05	void do_cir3d(int lc,int bc,int cx,int cy,int cz, int r);	Circle with radius r and center (cx, cy cz)
06	void do_arc3d(int lc,int bc,int cx,int cy,int cz, int r,int sa,int ea);	Arc with radius r and center (cx, cy cz)
07	void do_cir3dsl(CDC *pDC,int lc,int bc,int cx, int cy,int cz,int r,int ax,int ay,int az);	Circle with radius r, center (cx, cy cz) slanting at angles ax,ay, az
08	void do_cir3dsldf(CDC *pDC,int lc,int bc, int cx,int cy,int cz,int r,int ac,int ay,int az);	Filled circle with radius r, center (cx, cy cz) (rx, ry, rz) slanting at angles ax, ay, az
09	void do_arc3dsl(int lc,int bc,int cx,int cy,int cz, int r,int ax,int ay,int az,int sa,int ea);	Arc with radius r, center (cx,cy,cz) slanting at angles ax,ay,az
10	void do_fcir3d(int lc,int bc,int cx,int cy,int cz, int r);	Filled circle with radius r, centre(cx,cy,cz)
11	void do_farc3d(int lc,int bc,int cx,int cy,int cz, int r,int sa,int ea);	Filled arc with radius r , centre(cx, cy, cz)
12	void do_farc3dsl(int lc,int bc,int xc,int yc, int zc,int r,int ax,int ay,int az,int sa,int ea);	Filled arc with radius r, centre(cx, cy, cz) slanting at angles ax, ay, az
13	void do_sp3d(int lc,int bc,int sx,int sy, int sz,int r);	Sphere with radius r, center (sx, sy, sz)
14	void line3d(CDC *pDC,int lc,int bc,int x1, int y1,int z1,int x2,int y2,int z2);	Line from pt (x1, y1, z1) to pt (x2, y2, z2)
15	void rz(int lc,int x,int y,int z,int l,int b);	Rectangle on z=0 plane
16	void ry(int lc,int x,int y,int z,int l,int b);	Rectangle on y=0 plane
17	void rx(int lc,int x,int y,int z,int l,int b);	Rectangle on z=0 plane
18	void cxyz(int lc,int x,int y,int z,int l,int b,int d);	Cubical solid

Table 5: Graphic Libraries for Transformation routines

	Operation	Description
01	void rotate_ptx(int *x,int *y,int *z,int an);	Rotation around x-axis at angle an
02	void rotate_pty(int *x,int *y,int *z,int an);	Rotation around y-axis at angle an
03	void rotate_ptz(int *x,int *y,int *z,int an);	Rotation around z-axis at angle an
04	void translate_pt(int *x,int *y, int *z, int tx, int ty,int tz);	Translating a point with factors tx, ty, tz

05	void scale_pt(int *x,int *y,int *z,float sx, float sy,float sz)	Scaling a point with factors tx, ty, tz
06	void scale_rpt(int *x,int *y,int *z,float sx, float sy, float sz);	Scaling relative to a pt (x, y, z)
07	void rotate_rptxyz(int xv,int yv,int zc,int *rx, int *ry,int *rz,int anx,int any,int anz) ;	Rotation around x,y,z axes relative to a given pt at angles anx, any, anz
08	void rotate_rptx(int xc,int yc,int zc,int *rx, int *ry,int *rz,int an)	Rotation around x-axis relative to a given pt at an angle an
09	void rotate_rpty(int xc,int yc,int zc,int *rx, int *ry,int *rz,int an);	Rotation around y-axis relative to a given pt at an angle an
10	void rotate_rptz(int xc,int yc,int zc,int *rx, int *ry,int *rz,int an);	Rotation around z- axis relative to a given pt at an angle an

Table 6: Three Dimensional Display File Instructions

	Instruction	Description
01	void cir_abs3d(int lc,int x,int y,int z,int r);	Circle with radius r and center (x, y, z)
02	void cirf_abs3d(int lc,int x,int y,int z,int r);	Filled circle with radius r and center (x, y, z)
03	void cirslf_abs3d(int lc,int x,int y,int z,int r, int ax,int ay,int az);	Filled circle with radius r , centre(x, y, z) slanting at angles ax, ay, az
04	void cirsl_abs3d(int lc,int x,int y,int z,int r, int ax,int ay,int az);	Circle with radius r, centre(x, y, z) slanting at angles ax ay, az
05	void open_poly(int x,int y,int z);	To open a polygon
06	void close_poly();	To close polygon
07	void move_abs3(int x,int y,int z);	Move to a given point
08	void line_abs3(int lc,int bc,int x,int y,int z);	Draw line to a given point
09	void move_rel(int dx,int dy,int dz)	Moves from the current position to the relative specification
10	void point_abs3(int lc,int bc,int x,int y,int z);	Plots the given point
11	void line_rel3(int lc,int bc,int dx,int dy,int dz);	Draws line from the current position to the relative specification
12	void point_rel3(int lc,int bc,int dx,int dy,int dz);	Plots a point from the relative specification
13	void interpret1(CDC *pDC,int start,int count);	To interpret the segment from a given instructions starting address
14	void do_type3d(CDC *pDC, int typ,int sadd,int sz);	Executes a macro instruction
15	void show(CDC *pDC);	To display the display All segments
16	void showax(CDC *pDC);	To display the display axes

Table 7: Typical Three Dimensional Display file transformation routines

	Operation	Description
01	void newtran3();	Creates an unit matrix for transformation matrix
02	void translate3(int tx,int ty,int tz);	Translates an object to the given translation factors tx, ty, tz
03	void rotatex3(int a);	Rotates around x-axis by an angle a
04	void rotatey3(int a);	Rotates around y-axis by an angle a
05	void rotatez3(int a);	Rotates around z-axis at angle a
06	void rotatexyz3(int ax,int ay,int az);	Rotates around x,y,z axes at angles ax,ay,az
07	void rotatexyz3rel(int x,int y,int z,int ax, int ay,int az);	Rotates around x,y,z axes relative to a fixed pt at angles ax,ay,az
08	void rotatex3rel(int x,int y,int z,int an);	Rotates around x-axis relative to a fixed pt at angle an
09	void rotatey3rel(int x,int y,int z,int an);	Rotates around y-axis relative to a fixed pt at angle an
10	void rotatez3rel(int x,int y,int z,int an);	Rotates around z-axis relative to a fixed pt at angle an
11	void scale3(float sx,float sy,float sz);	Scales an object given scaling factors as sx, sy, sz
12	void scale3rel(int x,int y,int z,float sx, float sy,float sz);	Scales with respect to a given fixed point x,y,z
13	void do_ transformation(int *x1,int *y1, int *z1);	Given point is transformed as per the transformation matrix.
14	void settrans();	Initializes the transformation system

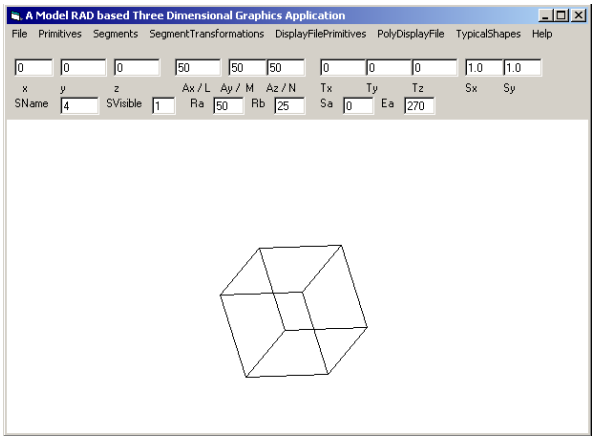

Fig 3: Model Simple VB client using the Vector Graphic display-file structure

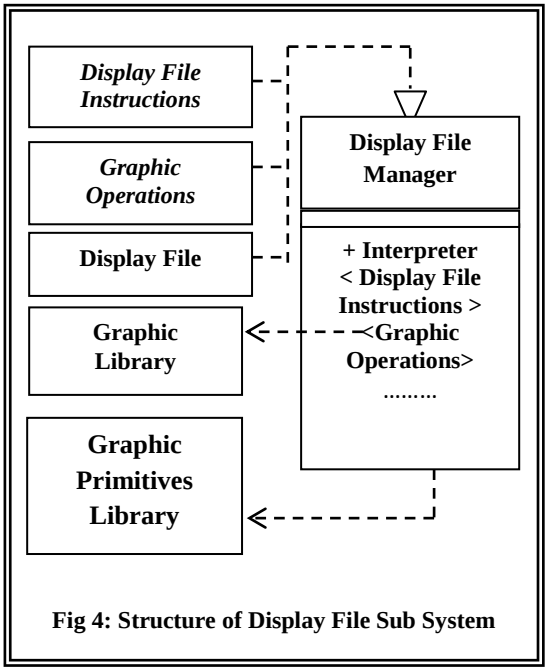

Fig 4: Structure of Display File Sub System

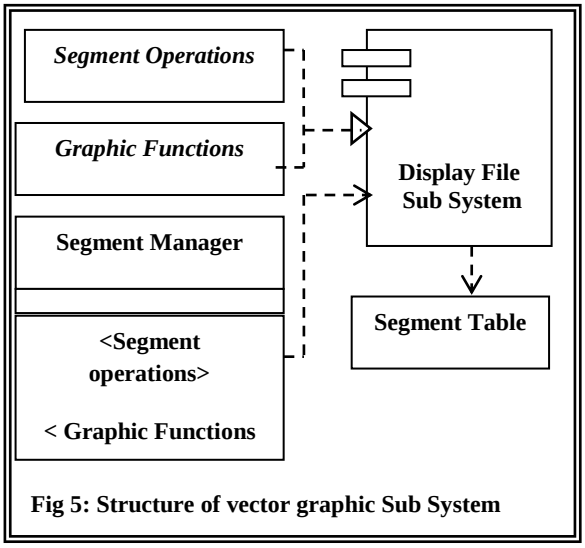

Fig 5: Structure of vector graphic Sub System

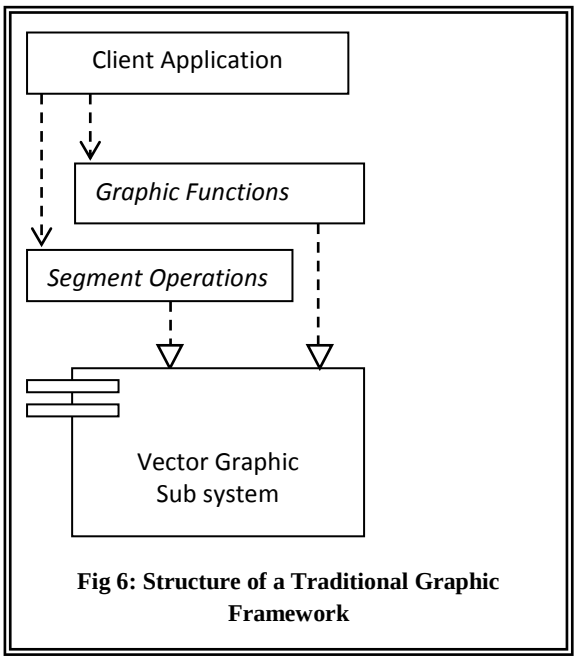

Fig 6: Structure of a Traditional Graphic Framework

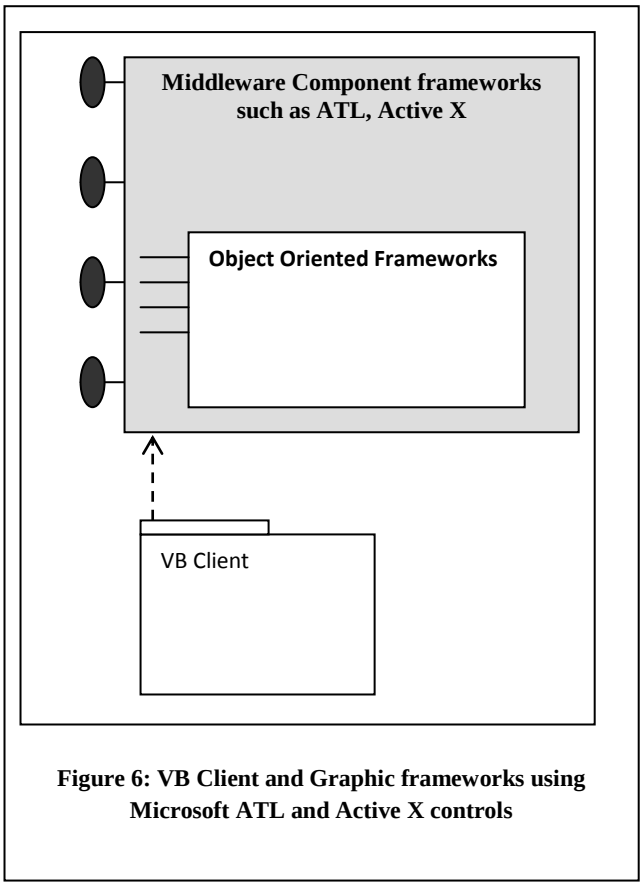

Figure 6: VB Client and Graphic frameworks using Microsoft ATL and Active X controls

Table 8: Sample VB statements using Graphic functions

For plotting a spear HGP3D1.Sp3d Text1.Text, Text2.Text, Text3.Text, Text4.Text
For plotting a cube HGP3D1.CXYZ Text1.Text, Text2.Text, Text3.Text, Text4.Text, Text5.Text, Text6.Text
For displaying all segments HGP3D1.ShowAll
For rotating a selected segment HGP3D1.RoteteSegmentAbs Text7.Text, Text4.Text, Text5.Text, Text6.Text HGP3D1.ShowAll
For resetting transformations matrix HGP3D1.ReSetTrans
For setting visibility of a sigment HGP3D1.SetSV Text7.Text, Text8.Text
For opening a segment HGP3D1.OpenSegment Text7.Text HGP3D1.CloseSegment Text7.Text
For performing rotation operation on a selected segment HGP3D1.RotateSegRel Text7.Text, Text4.Text, Text5.Text, Text6.Text HGP3D1.ShowAll
For plotting an Arc HGP3D1.Arc3D 1, Text1.Text, Text2.Text, Text3.Text, Text4.Text, Text5.Text, Text6.Text, Text14.Text, Text16.Text, Text17.Text

7. CONCLUSION

Traditional graphic methodology of handling graphic requirements needs to be redesigned and ported to new technologies. The presented graphic patter frame model uses traditional graphic display file and segmentation concepts for building graphic frameworks to suit requirements of several graphic frameworks. The graphic functionality of the model is classified into several groups and sample list of functions are listed. The middleware frameworks of Microsoft are used to organize and export function libraries in the form of framework. MFC based ATL and application wizards of Microsoft are suggested for building graphic frameworks. The VB environment is suggested for client application for integrating graphic frameworks with required domain applications through automation layer. Sample block diagram of such model is presented in Figure: 6. Sample code segment in VB is presented in Table 8.

Evolving patterns and models for managing Functions of different types used in this system enable this system work more efficiently. The presented graphic models can be refined further by evolving objected oriented graphic frameworks. Component Technology can be adopted to make the same models more compatible to the present graphic requirements. The Complexity Component based graphic systems can be manages by evolving of few pattern-frames. The main focus of the present model is to adopt traditional graphic techniques for building graphic frameworks.

ACKNOWLEDGMENTS

The author acknowledges all the professional advisors who motivated for developing graphic framework patterns. The author conveys special regards to Dr.K.V. Chalapati Rao retired Professor of C.S.E., College of Engineering–OU Hyderabad and Dr.I.V. Ramana Retired Professor of College of Engineering JNTU Hyderabad for motivation and encouragement. The author acknowledges faculty of Department of Computer Science and Engineering, Chaitanya Bharathi Institute of Technology for their support.

REFERENCES

[1] Dr. Hari Ramakrishna, “Design Pattern for Graphic/CAD Frameworks”, Ph.D thesis submitted to Faculty of Engineering Osmania University March 2003,

[2] Dr. Hari Ramakrishna and Dr.K.V Chalapathi Rao, “Pattern Methodology of Documenting and Communicating Domain Specific Knowledge”, CVR Journal of Science and Technology Vol 2. June 2012 ISSN 2277-3916.

[3] Christopher Alexander, “An Introduction for Object-oriented Design”, A lecture Note at Alexander Personal web site ***www.patternlanguage.com***

[4] Pattern Languages of Program Design. Edited by James O. Coplien and Douglas C. Schmidt. Addison-Wesley, 1995

[5] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides, "Design Patterns: Elements of Reusable Software Architecture", Addison-Wesley, 1995

[6] LNCS Transactions on Pattern Languages of Programming http://www.springer.com/computer/lncs?SGWID=0-164-2-470309-0

[7] Harrington: 87Harrington, S. (1987), Computer graphics: programming approach, McGraw-Hill International, second edition.

[8] Newman,W.S and Sproul, R.S (1981), “Principles of interactive computer graphics”, McGraw-Hill International, Second edition.

[9] Hari RamaKrishna “Three-dimensional interactive computer graphics package for civil engineering application”, Proceedings of The National Conference on Civil Engineering Materials and Structures, Jan 19-21, 1995, Osmania University; Hyderabad ;India

[10] Hari RamaKrishna “Generation of flooring and wallpaper patterns using computer graphics” Proceedings of the First National Conference on Computer Aided Structural Analysis and Design, Jan 3-5,1996, Engineering Staff College of India and University College of Engineering, Osmania University, Hyderabad

[11] Hari RamaKrishna, “Application of computer graphics in interior design” Proceedings of Conference 1998 at Institutes of Engineers at Hyderabad.

AUTHOR’S PROFILE

Dr. Hari Ramakrishna was awarded B.E in Computer Science and Engineering in 1989 by Osmania University, Hyderabad, A.P., INDIA, M.S., in Computer Science by BITS PILANI, INDIA and Ph.D. in Computer Science and Engineering by the Faculty of Engineering Osmania University in ***“Pattern languages for graphic /CAD frameworks”***. He worked in Software Industry for several years developing Graphic, CAD /GIS products using Microsoft environment. He has about 15 years of teaching experience. Presently he is working as a Professor for last 7 years in the Department of Computer Science and Engineering at Chaitanya Bharathi Institutre of Technology, Hyderabad INDIA. He involved in the design and development of several graphic frameworks for various Engineering applications.