Design and Development of Gerber Data Editing Software

A dissertation submitted for the Degree of Master of Information Technology

A.M.D.S. Kularathna

University of Colombo School of Computing

2018

Declaration

The thesis is my original work and has not been submitted previously for a degree at this or any other university/institute.

To the best of my knowledge it does not contain any material published or written by another person, except as acknowledged in the text.

Student Name:	A.M.D.S. Kularathna
Registration Number:	2015/MIT/029
Index Number:	15550294

____

Signature: Date: 12/07/2018

This is to certify that this thesis is based on the work of

Mr. A.M.D.S. Kularathna

under my supervision. The thesis has been prepared according to the format stipulated and is of acceptable standard.

Certified by:

Supervisor Name: Dr. K.L. Jayarathne

____

Signature: Date:

Abstract

Solder paste application stencil design is a crucial task in Electronic Manufacturing and Service (EMS) industry due to the complexity of modern electronics devices. Proper solder joint must be built-up between electronics components and the printed circuit board (PCB) in order to achieve expected functionalities, performance and durability of such device. To obtain a quality solder joint, an optimal solder volume should be deposit by using a properly designed stencil on the component pads in the PCB layout.

The aim of this project is to provide a software solution for designing solder paste application stencils based on the respective Gerber data of the PCB. The Gerber is a 2-D monochrome vector image data representation format which extensively used in EMS industry. Gerber data file is an output of PCB designing applications. Gerber file consists of human readable series of commands that corresponds to series of graphics objects supper imposed on top of each and ultimately generates a single image.

Since Gerber file consists of set of graphic objects, object oriented programming concept was involved in the solution. All the basic objects as well as macro objects which are defined in the Gerber format specification [1] were developed according to the object oriented concept. Major functionalities required for stencil design process; reading, displaying, editing and saving were developed in the evolutionary prototype model.

Subsequent to the main problem, important algorithm for selecting complex 2-D polygon objects was invented during the project. The new algorithm can be used to solve common Point-In-Polygon problem in any 2-D vector graphics application.

ACKNOWLEDGEMENT

First of all I acknowledge with gratitude to my supervisor Dr. Lakshman Jayarathne who guided me in order to successfully complete my project throughout two semesters of study. During that period, when I was unable to fulfill some project milestones, he never blames at me and always encouraged me to plan the work and achieve next milestone on time. Even though he is extremely busy, his door was never closed when I go to meet him to discuss some issues I faced.

Secondly I would like to thank my friends who are working as an engineers in CCS Lanka (Pvt) Limited, for the given great support and valuable inputs from requirement gathering phase to various stages of my project. Without their support, I may not be able to successfully complete my project.

Finally, I must express my very profound gratitude to my parents and to my wife for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of development and writing this thesis. This accomplishment would not have been possible without them.

Table Contents

List of tables

List of figures

Definitions, Acronyms and Abbreviations

Term	**Definition**
Gerber	Two dimensional vector image data representation format
2-D	Two dimensional
EMS	Electronic manufacturing and service
Stencil	Sheet metal block consists of set of geometric shaped holes
The system	Gerber Data Editing Software System
FR	Functional requirement
Aperture	Geometric shape

Chapter 1: Introduction

1.1 Overview

Today electronic devices become essential elements in human's day to day life activities. Especially mobile phones become the most intimate device. All those electronic devices are manufactured by electronic manufacturing and service (EMS) industry. With the advancement of technology, electronic devices become smarter, more complex and compact in size.

Basic building block of any electronic device is an electronic circuit. It is composed of printed circuit board, passive components such as resistors, capacitors, inductors and active components such as diodes, transistors integrated circuits etc. Figure 1 shows the SAMSUNG GALAXY S 4G PCB board component layout.

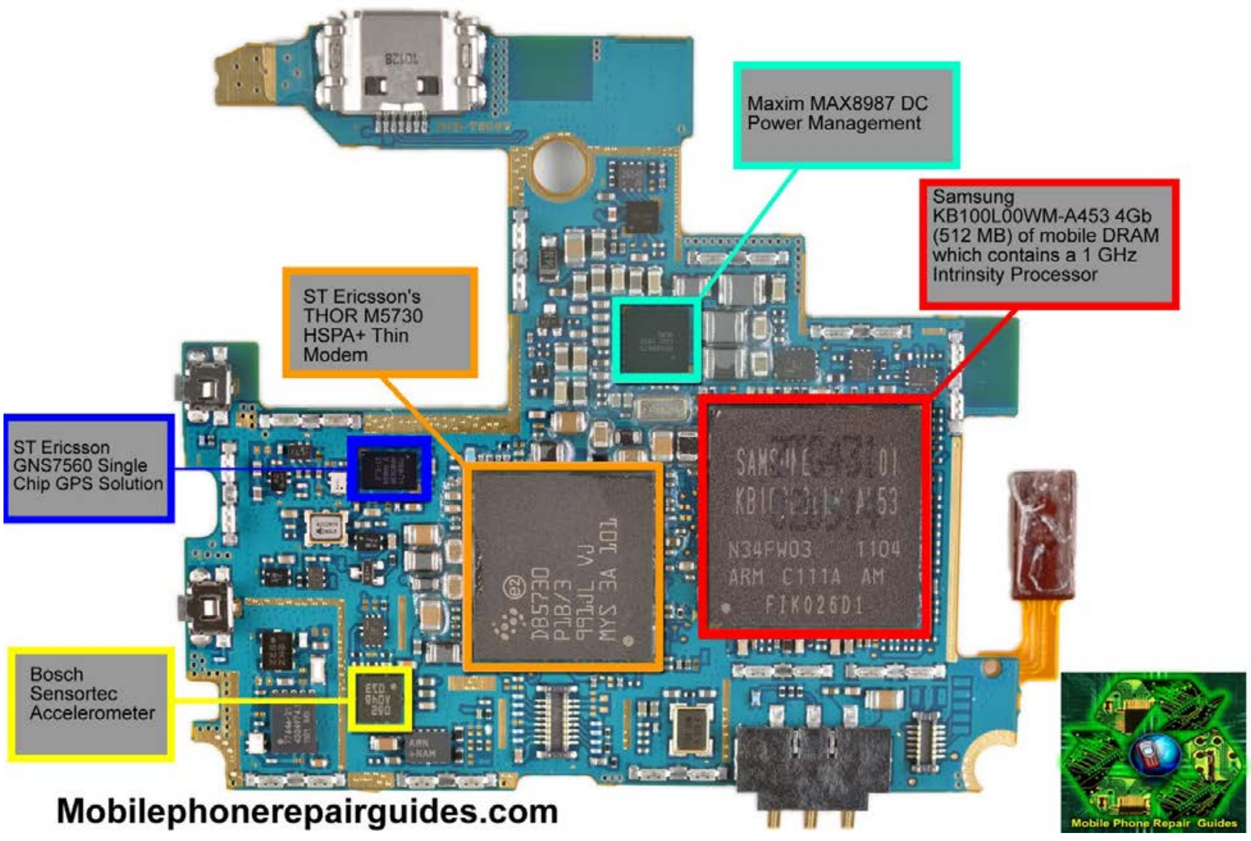

Maxim MAX8987 DC Power Management
Samsung KB100L00WM-A453 4Gb (512 MB) of mobile DRAM which contains a 1 GHz Intrinsity Processor
ST Ericsson's THOR M5730 HSPA+ Thin Modem
ST Ericsson GNS7560 Single Chip GPS Solution
Bosch Sensortec Accelerometer
Mobilephonerepairguides.com
Mobile Phone Repair Guides

Figure 1 – Mother board of a mobile phone

When the device becomes complex and compact, respective component layout is more complex and component density is high. Assembling such a device is really a challenge for EMS industry. Nowadays most of the component layouts come with surface mount devices

(SMD) to make the layout more compact. Bulky through-hole components were replaced by small SMD components with the technology advancement.

To assemble SMD components into a PCB layout, special technology called “Reflow Soldering” is used. Reflow soldering process can be divided into three main steps.

1. Solder paste application process

Solder paste is a mixture of tiny solder balls and a flux which formed into a paste. The solder paste is applied on component pads of a PCB layout by using a stencil. Figure 2 shows applied solder paste on a component pad by using a stencil.

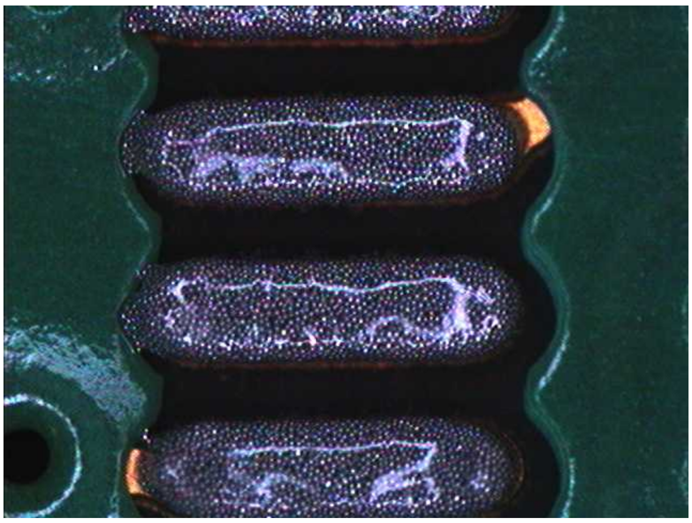


Figure 2 – Solder paste application on PCB pads

2. Pick and place process

Placing SMD components on the relevant position of the PCB layout by using automated machines are called the pick and place process. Normally SMD components are received as reels from suppliers. Such reels are loaded into pick and place machines and then the machine will pick the relevant component from specified reel and specified location according to the given assembly program to the machine.

3. Reflow process

Actual soldering process is occurred under reflow process. Special oven called “Reflow Oven” is involved in this process. PCB with assembled component is heated up according to a special profile in the reflow oven. When the temperature reaches the melting point of applied solder paste, it will create solder joint between PCB and the placed components.

Most challenging part of above mentioned processes is applying optimal solder paste volume on component pads. In other words, designing most appropriate solder paste application stencil for a given PCB layout is the challenging part. Stencil fabrication part is not so much challenging, because laser cutting technology is available to cut any arbitrary shape in a thin stainless steel sheet. Next section focuses on the design process of solder paste application stencil.

1.2 Problem

Most of the standard PCB design software has a capability of exporting layers of the PCB as two dimensional monochrome vector image format called “Gerber”. Since design file has all the intelligence about the circuit, design files are not interchanged between industries because the possibility of copying the design by another party. The Gerber Format has become de facto standard of image data transferring format among electronics industry. Basic resource available for stencil design process is the respective Gerber file of the PCB layout.

As mentioned earlier, Gerber is a two dimensional monochrome vector image data representation format. The format was originally developed by Gerber Systems Corp., a division of Gerber Scientific, founded by Joseph Gerber. Now it is owned by Ucamco. Latest version is 2017.11 in [1].

Gerber is a simple, compact and clear format which uses human readable 7 bit ASCII characters for image data representation. One Gerber file completely describes one single image. It consists of series of graphic objects super imposing on top of each other with polarity dark or clear. Graphic object with clear polarity will clear all the objects beneath it according to its contour whereas object with dark polarity creates mark on the image. Simple Gerber is mentioned below to understand its simplicity.

Sample Gerber file:

```
% FSLAX3.5Y3.5*%
%MOMM*%
%AD20C,0.75*%
D20*
X0Y0D03*
M02*
```

1.3 Aim

To develop a software solution for EMS industry to design solder paste application stencils efficiently and effectively using Gerber files.

1.4 Objective

- Gerber file format will be studied.
- Classes will be created based on the defined graphics objects in the Gerber format specification according to object oriented programming concept.
- A graphics API will be used to display graphics objects on the screen by creating user friendly user interface.
- Object selection method will be created for each graphic object class.
- A prototype software system will be created and tested using a sample Gerber file.

1.5 Scope

All the features defined in the current Gerber format specification will not be considered for the development of the initial prototype system. Essential graphics objects which are required to design solder paste stencil will be considered in the initial solution. Only one Gerber file will be possible to open at a time in the first prototype version. User will be given access to open, view, select, edit and save actions in the system. Rest of the features will be added iteratively into the prototype system.

Chapter 2: Background and Literature Review

2.1 About Gerber Format

As mentioned in previous chapter, Gerber format is a bi-level (two color) two dimensional vector image data representation format. Printable 7-bit ACSII characters are used in the representation. Single Gerber file consists of series of commands. All the commands are classified into two major groups called “Function Codes” and “Extended Codes” in [1].

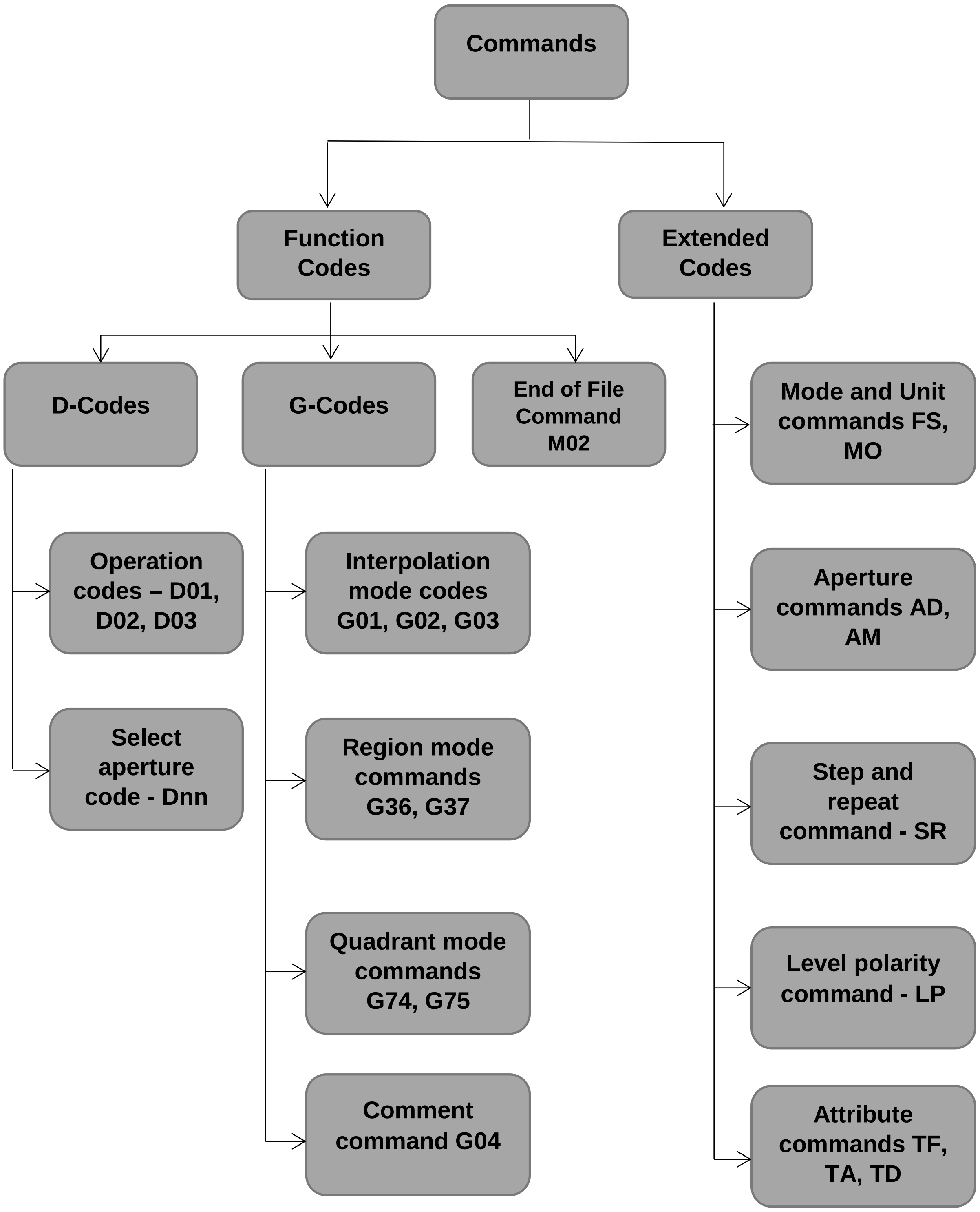

Commands
Function Codes
Extended Codes
D-Codes
G-Codes
End of File Command M02
Operation codes – D01, D02, D03
Select aperture code - Dnn
Interpolation mode codes G01, G02, G03
Region mode commands G36, G37
Quadrant mode commands G74, G75
Comment command G04
Mode and Unit commands FS, MO
Aperture commands AD, AM
Step and repeat command - SR
Level polarity command - LP
Attribute commands TF, TA, TD

Figure 3 – Gerber file structure

Single Gerber file completely describes single image. A Gerber file can be processed in a single pass. Like other programming languages, parameters and objects must be specified before they are used. As an example, simple Gerber file is illustrated below. It will create 3x1.5mm rectangle in the origin with 0.5mm hole at the center as shown in Figure 4.

```
%FSLAX23Y23*%
%MOMM*%
%AD10R,3X1.5X0.5*%
D10*
X0Y0D03*
M02*
```

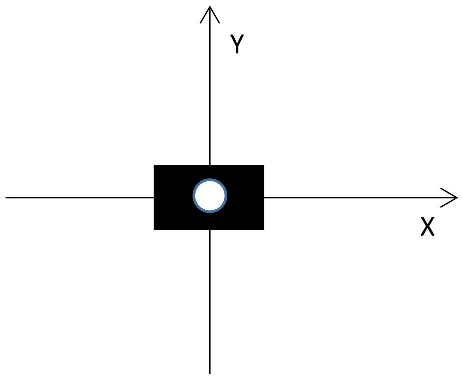


Figure 4 – Gerber image demonstration

2.2 Graphic Objects Defined in the Gerber Format

Special name called “Aperture” is used in the Gerber format to denote a graphic object. Two types of apertures are defined as standard apertures and macro apertures. Standard apertures consist of basic geometric shapes which are pre-defined. Circle (C), rectangle (R), obround (O) and polygon (P) are the standard apertures defined in Gerber format. Arbitrary geometric shapes can be defined using Macro Apertures. A hole can be defined with the standard apertures but cannot be done the same with macro apertures.

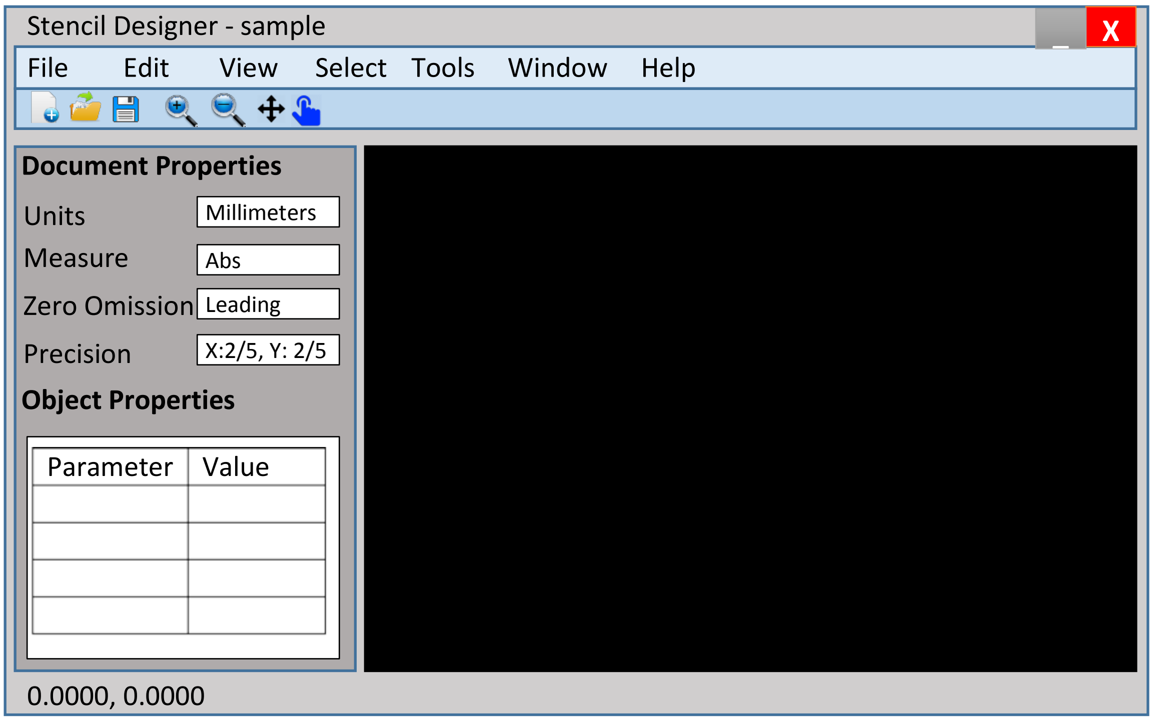
A Gerber file consists of series of graphic objects which can be a standard or macro aperture as a stream of commands mentioned in a Gerber file. Final image is generated by super imposing all the individual graphic objects. Important parameter of the state of any graphic object is its polarity. Polarity has two values such as Dark and Clear. Graphic object with dark polarity will mark the canvas whereas clear polarity object clear the canvas irrespective of underneath graphic objects. Single graphic object is defined only once at the beginning of the file. Defined object can be use any times in different locations. Lines and arcs can be generated by stroking circle (C) and rectangle (R) standard apertures.

To interpret actual size and location of graphic objects, functional attributes which are valid for entire file is defined at the start of Gerber file. Coordinate format and measurement units are one of such important attributes mentioned in Gerber files. As an example, such attributes are mentioned in the above illustrated simple Gerber file.

2.3 Requirements Analysis of the proposed System

Basic requirement of the proposed system is the ability of reading a Gerber file without missing any important information. System should be capable of reading and identification of Function codes, Operation codes and Graphic State Variables. While reading the file, system should be able to generate the aperture table which including all the defined apertures in the Gerber file. When executing operation codes, relevant stream of graphic objects should be created.

Graphical representation of Gerber file on the screen is another requirement. Interactive user interface also an essential part of the system. Appearance of such prototype interface for the proposed system is shown in Figure 5.



Units: Millimeters
Measure: Abs
Zero Omission: Leading
Precision: X:2/5, Y: 2/5

Parameter	Value

0.0000, 0.0000

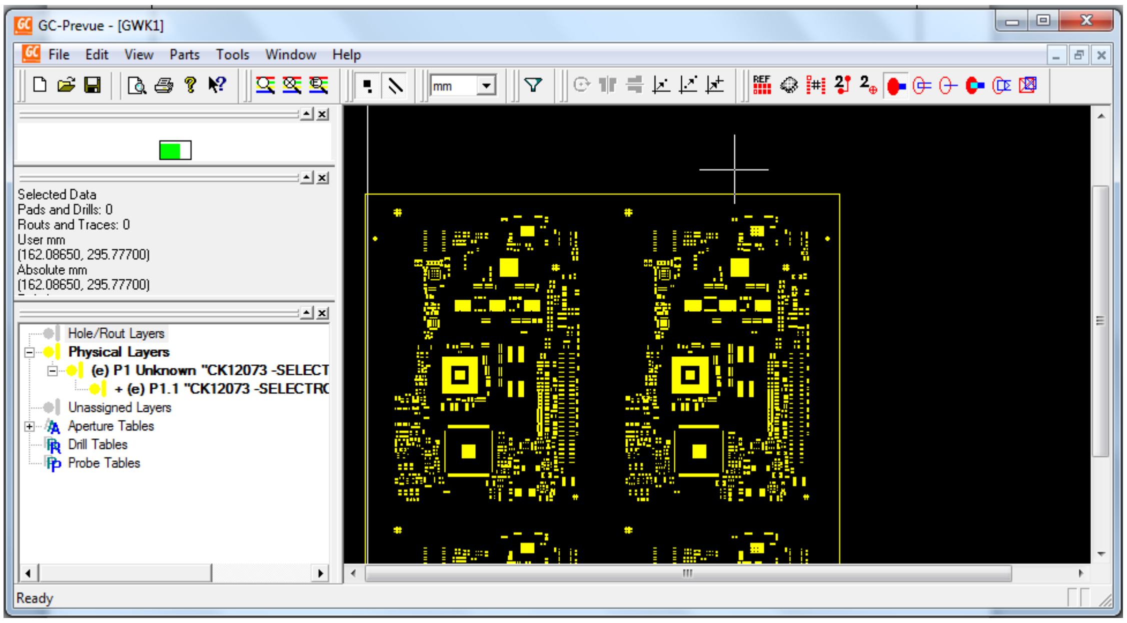
Figure 5 - Basic User Interface

User could be able to select any aperture in the Gerber file through the interface. Clear difference of selected object with other unselect objects should appear. Properties of selected object should be displayed on the left pane of above shown prototype interface. If the editing mode is enabled, text boxes which display the properties of selected object should be unlocked to enter new values by the user. When the values get changed, screen should update immediately with respond to the user input. System should respond not only for the keyboard input, but also to the mouse input as well. When mouse click on a graphic object, control handles should be appear on its nodes in order to facilitate editing by mouse input. Apart from selection and editing functionalities, basic Save and Print function should be available. Requirements are described in detail in the next chapter.

2.4 Review of Similar Systems

There are commercial Gerber data editing tools available in the industry. Those are expensive and difficult to customize. Available features and limitations of those systems are described below.

GraphiCode [3] provides several software solutions for different industries. GC-Prevue is a free tool up to the end of year 2017 as shown in Figure 2.4 from GraphiCode which offer only viewing facility of Gerber data. GC Power Station is the commercially available tool from GraphiCode.



Selected Data
Pads and Drills: 0
Routs and Traces: 0
User mm
(162.08650, 295.77700)
Absolute mm
(162.08650, 295.77700)

Hole/Rout Layers
Physical Layers
(e) P1 Unknown "CK12073 -SELECT
+ (e) P1.1 "CK12073 -SELECTRC
Unassigned Layers
Aperture Tables
Drill Tables
Probe Tables

Figure 6 - GC Preview interface

Table 1 - Features Comparison of GC-Prevue

Features	Limitations
Import Gerber, PDF and DXF files	Unable to perform Cut, Copy and Paste operations
Open several files together with layer option	New data creation is not possible
Layer alignment facility	Feature editing is not supported
Printing and screen capture facility	Selection facility is very basic
Measurement capability	Centroid extraction is not possible
Customization of user interface	Gerber output format is not available

Viewplot [4] is also another CAD data editing and viewing tool provider. It supports industry standard formats such as ODB++, HPGL, DXF, Drill/NC etc. apart from Gerber. The viewer of the Viewplot is a free version. The interface of the Viewplot application which is displayed in its website is shown in Figure 7 below.

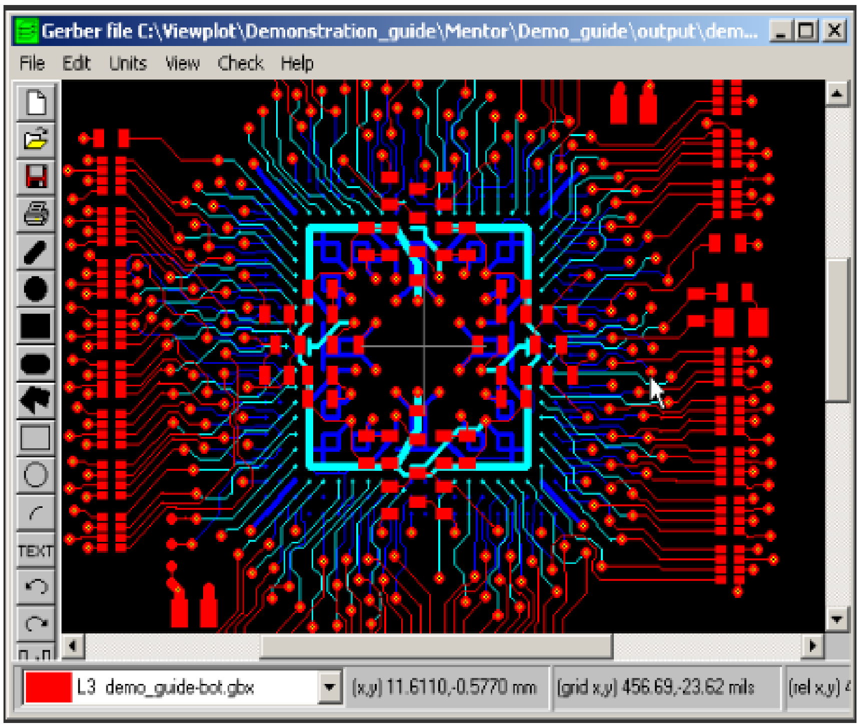

Gerber file C:\Viewplot\Demonstration_guide\Mentor\Demo_guide\output\dem...
File
Edit
Units
View
Check
Help
TEXT
L3 demo_guide-bot.gbx
(x,y) 11.6110,-0.5770 mm
(grid x,y) 456.69,-23.62 mils
(rel x,y)

Figure 7 - Viewplot Interface

Features of Viewplot application was identified as follows.

- Verify design data / design complexity
- Create high quality PDF documentation
- Basic design modification
- Conversion of RS-274D (obsolete Gerber format) to RS-274X (new format)
- Read various input formats

There are various similar kind of software solution providers can be found in the internet, but during the review of most of them found that providers who offer free Gerber data editing solution is very rare and most of them provides free tool for only viewing Gerber files. Two popular vendors of such were reviewed in details above.

Chapter 3: Analysis

3.1 Software Requirement Specification

3.1.1 Purpose

The purpose of this section is to give a detailed description of the requirements for the "Gerber Data Editor" software. The document contains the complete functional and non-functional requirements and design constraints to developers. It will further explain system interfaces and interactions with primary users. This document is primarily intended to be proposed to the customer for its approval and a reference for developing the first version of the system for the development team.

3.1.2 Scope

The "Gerber Data Editor" is a 2-D vector image data processor which intended to use for designing of solder paste application stencils in EMS industry. Application is a standalone PC software package with a license which can be installed in a Windows operation system.

Main input of the software is a Gerber file and system can output the same format file as Gerber. Within the application, user will be able to select and edit graphic objects which consist in the original input file. Details of those operations will be described in details in functional requirements section.

3.1.3 Overview

The rest of this chapter is focused on functional and non-functional requirements of the system. First part is consists of functional requirements which the system should offer. The second part consists of interactions and interfaces that the system should support to achieve the intended functional requirements. The last part is focused on non-functional requirements and specific constraints of the system.

3.2 Functional Requirements

Main functional requirement of the proposed system is to edit Gerber files graphically with the aid of user friendly interface. User and the system are the major actors of this system. User and the system should be able to perform below mentioned actions using the system.

Table 2 - Use cases

Actor	**Use Cases**
User	Installation
	Import Gerber file
	Select graphic object
	Edit selected graphic object
	Save edited file in Gerber format
System	Read the content of imported Gerber file
	Display the read file graphically in the screen
	Highlight graphic object which user select by mouse
	Display the properties of selected graphic object
	Prompt editing options available for selected graphic object
	Output edited Gerber file

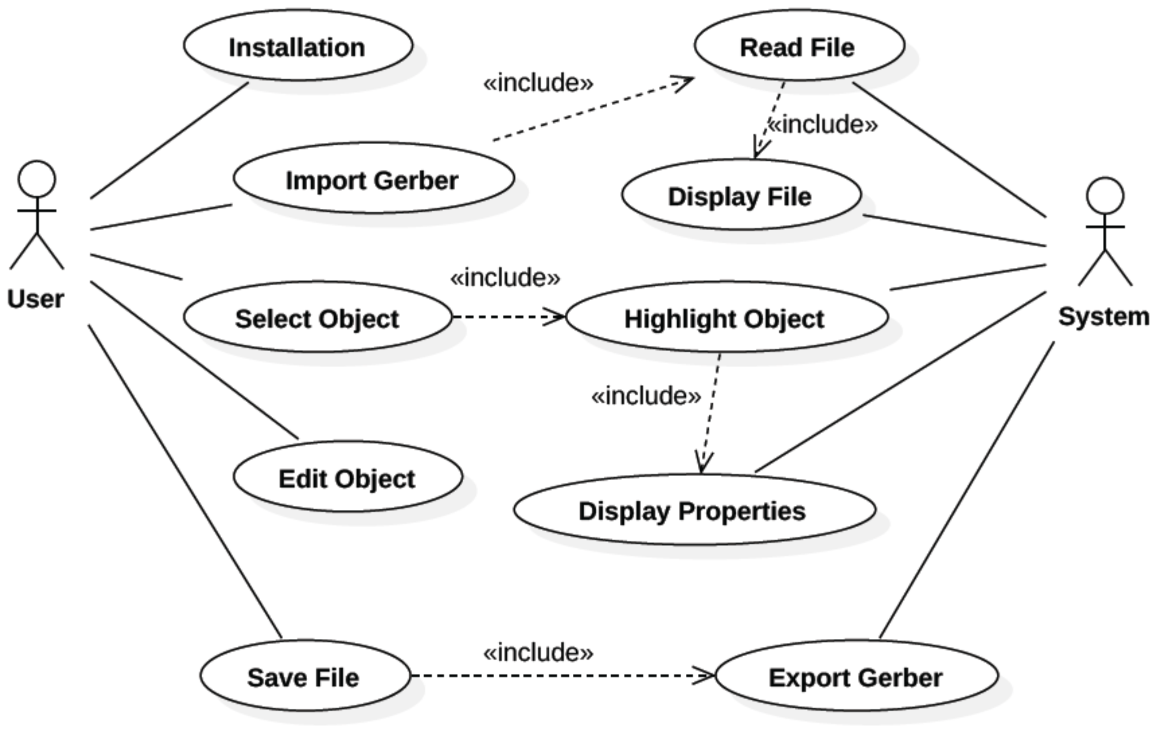


Figure 8 - Use Case Diagram

Detailed functional requirements.

Functional requirement 01:

ID: FR-01

Title: Installation of software application

Primary actor: User

Description: User should be able to install the application without facing any trouble like usual windows installation package. Unique activation key is given to the user in order to activate the software. User will able to try the application ten times as trial. After expiration of the trial period, application should not be open and display message to user to activate the product.

Dependency: None

Functional requirement 02:

ID: FR-02

Title: Registering the application

Primary actor: User

Description: User enters the given unique product key after installation of the system. System will verify the key and register the key for particular user.

Alternative path: User can use the system as trial until trial period expires.

Dependency: FR-01

Functional requirement 03:

ID: FR-03

Title: Launching the application

Primary actor: User

Description: User should be able to launch the application either double clicking on the desktop icon or selecting appropriate program name in the windows start menu.

Installer should create both desktop icon as well as program group.

Dependency: FR-01

Functional requirement 04:

ID: FR-04

Title: Open a Gerber file

Primary actor: User

Description: User should able to open a Gerber file using a menu icon / menu.

Dependency: FR-03

Functional requirement 05:

ID: FR-05

Title: Displaying file dialog

Primary actor: System

Description: When user click “Open” menu icon or select “Open file” option under “File” menu, system should display the file dialog pointing “My Documents” system folder with applying file filter “.gbr” and “all files”. User should able to select the desired Gerber file in the file dialog box.

Dependency: FR-04

Functional requirement 06:

ID: FR-06

Title: Read content of user selected Gerber file

Primary actor: System

Description: System should be able to read the content of the selected Gerber file according the [1] Gerber format specification. While reading the content of the file, system should able to construct the graphic object model which is described in design and methodology chapter.

Dependency: FR-04

Functional requirement 07:

ID: FR-07

Title: Display content of an opened Gerber file graphically on the screen

Primary actor: System

Description: Using proper color scheme which is described in details in “User Interface Requirements” section, graphic objects contain in the opened Gerber file should be displayed in the screen using appropriate scale factor. Final image should be initially center in the display area as well as whole image should be displayed.

Dependency: FR-06

Functional requirement 08:

ID: FR-08

Title: Select graphic elements in the file

Primary actor: User

Description: User should be able to select visually graphic objects which are displayed on the screen by using the mouse. Color scheme of selected graphic objects should be changed in order to distinguish with unselected objects. Properties of selected graphic object should be displayed on a property window. If more than one object is selected, only common properties are displayed in the property window.

Dependency: FR-07

Functional requirement 09:

ID: FR-09

Title: Identify selected graphic

Primary actor: System

Description: System should be able to track mouse pointer movements. If user perform left mouse click on an object while the mouse pointer lies within a boundary of a graphic object, resulting object is considered as selected. If the mouse pointer is in an intersecting area of several graphic objects, least distance from mouse pointer to the center of resulting graphic object is considered as selected.

Dependency: FR-07

<u>Functional requirement 10:</u>

ID: FR-10

Title: Display properties of a selected graphic

Primary actor: System

Description: System refers the graphic objects collection and identifies the selected object in the collection and displays the properties relevant to that object.

Dependency: FR-09

<u>Functional requirement 11:</u>

ID: FR-11

Title: Change properties of graphic objects

Primary actor: User

Description: User should able to change the parametric properties of selected object using the property window.

Dependency: FR-10

<u>Functional requirement 12:</u>

ID: FR-12

Title: Update object parameters

Primary actor: System

Description: When user changes any property, change should be updated in the relevant graphic object in the objects collection.

Dependency: FR-09

<u>Functional requirement 13:</u>

ID: FR-13

Title: Undo operation

Primary actor: User

Description: When user click on the “Undo” button in the interface, system should go to the previous state. User should be able to go back at least 5 stages.

Dependency: FR-06

Functional requirement 14:

ID: FR-14

Title: Save operation

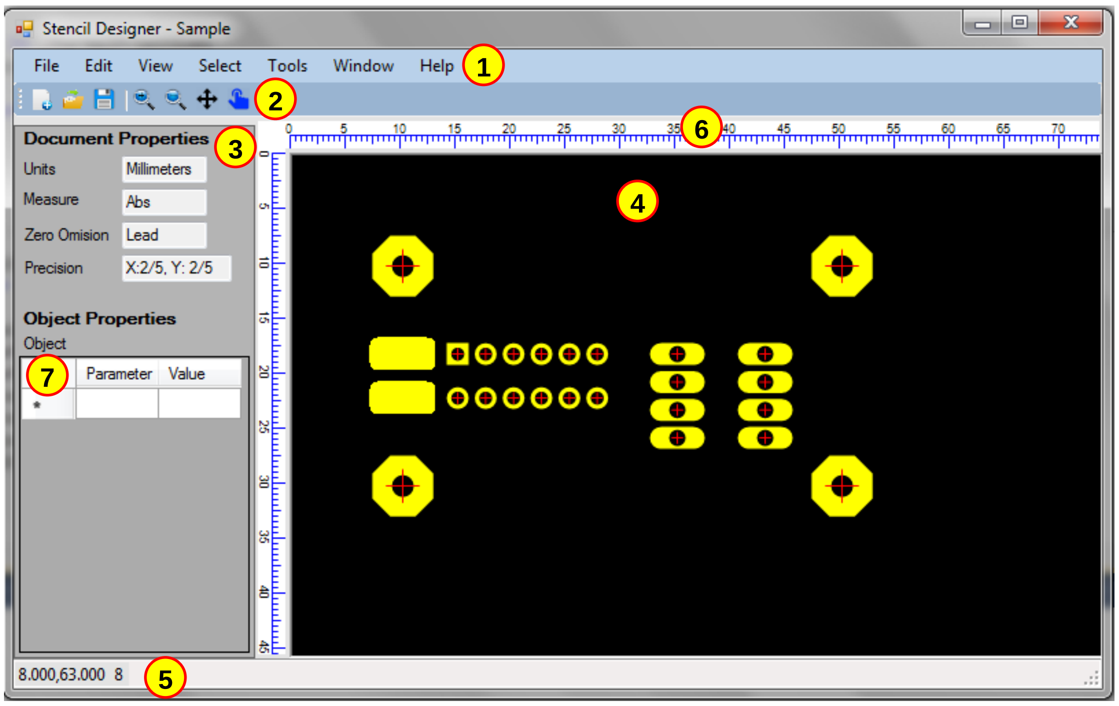
Primary actor: User

Description: System should offer both "Save" and "Save As" options. "Save" operation will overwrite the input file. "Save As" operation will create new file with Gerber file format. If user selects "Save As" option, file dialog should be displayed to user to create a folder and provide a file name for the output file.

Dependency: FR-06

3.3 User interface requirements

Software should be a windows form application. It should consist of title bar, menu bar, tool bars, property windows, graphic display area and status bar. Sample layout of proposed system is shown below.



Units: Millimeters
Measure: Abs
Zero Omision: Lead
Precision: X:2/5, Y: 2/5

8.000,63.000 8

Figure 9 - Basic user interface

Elements of the user interface

1. Menu bar
2. Tool bar
3. File information pane
4. Display area
5. Status bar
6. Ruler
7. Object property view

Self-descriptive icons should be used for the tool bar. When the user move the mouse move over any tool bar icon, respective action should be displayed as tool tip. Left side pane is allocated to display the properties of selected objects. Large black area is allocated for the canvas of the graphic display. Background color black is chosen to improve the visibility to the user. The default display color of graphic objects is yellow and default color of selected graphics is white as shown in Figure 10.

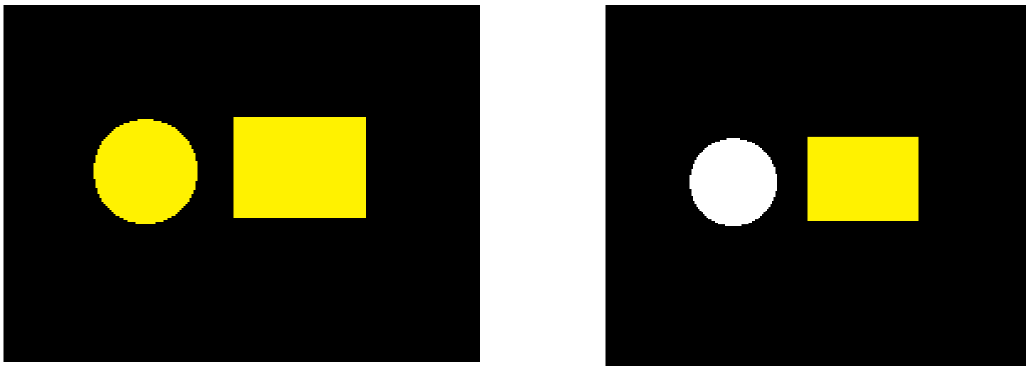


Figure 10 - Default color scheme

3.4 Other requirements

1. Computer with Windows operating system with latest .NET framework.
2. Minimum 1GB RAM
3. Minimum 2GB free hard disk space
4. Minimum 512MB VGA
5. Pointing device (mouse)
6. Screen with minimum resolution 1280 x 1024

Chapter 4: Design and Methodology

Some of the functional requirements mentioned in the previous chapter might be subjected to change. Gerber file format specification also getting updated when considers its revision history. Each revision it introduces new features into the format. So it is not possible to follow concrete design methodology for this project. Iterative approach is the best suitable design methodology for such problem.

4.1 Object model

Gerber file consists of two basic graphic object types.

1. Standard aperture
2. Macro aperture

4.1.1 Standard apertures

There are four different types of standard apertures defined in the Gerber format specification as mentioned in Table 3.

Table 3 - Standard apertures

Aperture name	Aperture code
Circle	C
Rectangle	R
Obround	O
Polygon	P

Circle, rectangle and polygon are common shapes, but obround is a special shape which is extensively used in PCB layouts. It consists of equal semi-circles connected with equal parallel lines as shown in Figure-11.

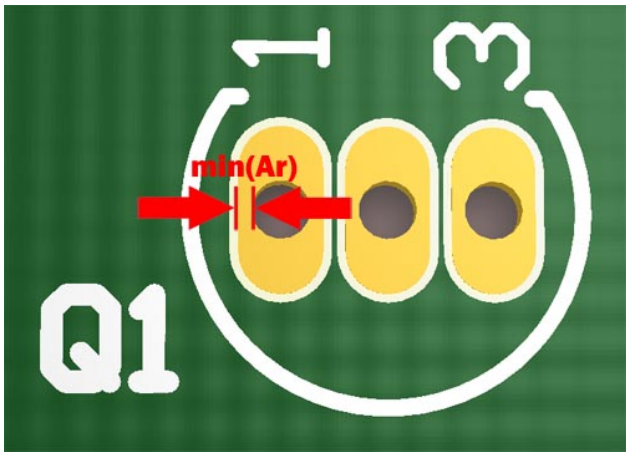
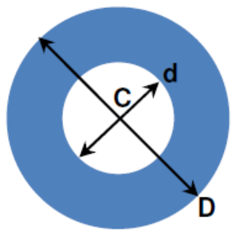


Figure 11 - Obround PCB pad

Standard apertures have below mentioned properties.

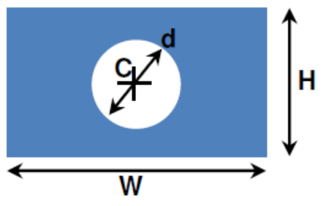
Circle

- D: Diameter (decimal, >= 0)
- d: Hole diameter (decimal, >0)



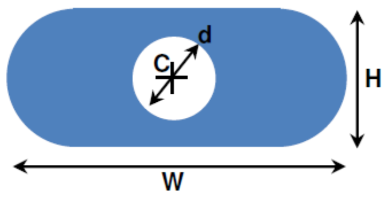
Rectangle

- W: X Size (decimal, >0)
- H: Y Size (decimal, >0)
- d: Hole diameter (decimal, >0)



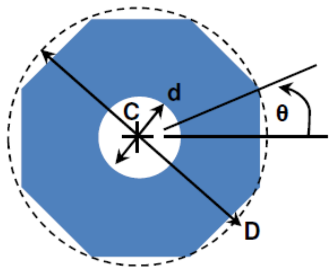
Obround

- W: X Size (decimal, >0)
- H: Y Size (decimal, >0)
- d: Hole diameter (decimal, >0)



Polygon

- D: Outer diameter of circumscribed circle (decimal, >0)
- Number of vertices (integer, 3≤n≤12)
- θ: Rotation angle (decimal): positive in counter clockwise direction with respect to
- d: Hole diameter (decimal, >0)



4.1.2 Macro apertures

Macro aperture is a customized shape which defined in the macro definition. Any complex shape can be defined by using macro. Macro definition consists of one or more basic shapes called "Primitives". There are eight primitives defined in the current Gerber format specification as mentioned below

Table 4 - Aperture macro primitives

Primitive	Primitive Code
Comment	0
Circle	1
Vector line	20
Center line	21
Outline	4
Polygon	5
Moire	6
Thermal	7

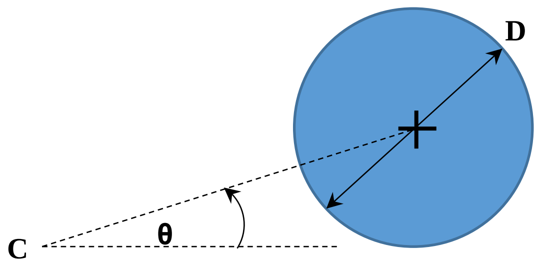
Primitives have below mentioned properties.

Comment

- Comment text (String)

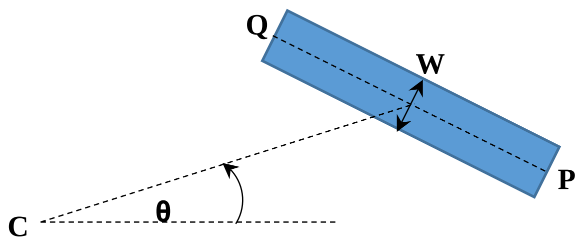
Circle

- Exposure (Boolean)
- Diameter (Decimal, >0)
- Center X (Decimal)
- Center Y (Decimal)
- Rotation angle (Decimal)



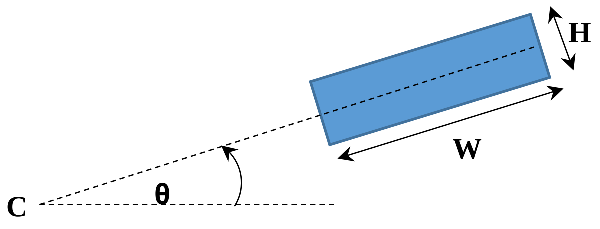
Vector line

- Exposure (Boolean)
- Line width (Decimal, ≥0)
- Start point X (Decimal)
- Start point Y (Decimal)
- End point X (Decimal)
- End point Y (Decimal)
- Rotation angle (Decimal)



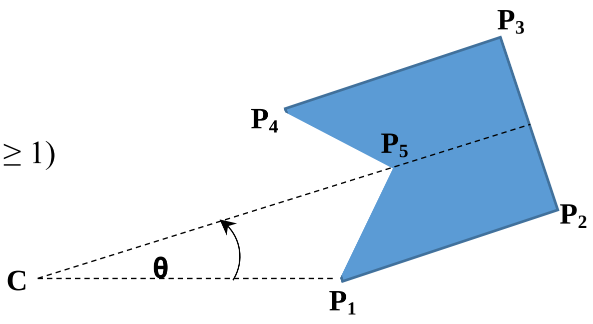
Center line

- Exposure (Boolean)
- Width (Decimal)
- Height (Decimal)
- Center point X (Decimal)
- Center point Y (Decimal)
- Rotation angle (Decimal)



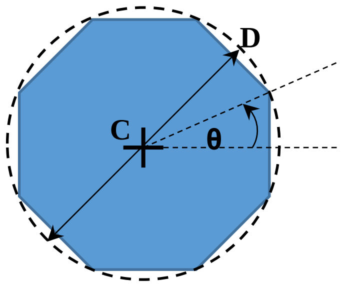
Outline

- Exposure (Boolean)
- Number of sub sequent points (Integer ≥ 1)
- Sub sequent points (List)
- Rotation angle (Decimal)



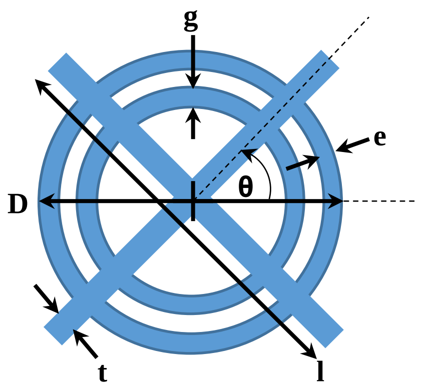
Polygon

- Exposure (Boolean)
- Number of vertices (Integer 3≤n≤12)
- Center point X (Decimal)
- Center point Y (Decimal)
- Diameter of circumscribed circle (Decimal, ≥0)
- Rotation angle (Decimal)



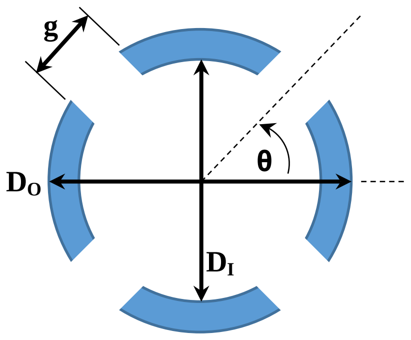
Moire

- Center point X (Decimal)
- Center point Y (Decimal)
- Outer diameter of outer concentric ring (Decimal, ≥0)
- Ring thickness (Decimal, ≥0)
- Gap between rings (Decimal, ≥0)
- Maximum number of rings (Integer, ≥0)
- Cross hair thickness (Decimal, ≥0)
- Cross hair length (Decimal, ≥0)
- Rotation angle (Decimal)



Thermal

- Center point X (Decimal)
- Center point Y (Decimal)
- Outer diameter (Decimal, outer diameter > inner diameter)
- Inner diameter (Decimal, ≥0)
- Gap thickness (Decimal)
- Rotation angle (Decimal)



4.2 Aperture Collection Object Structure

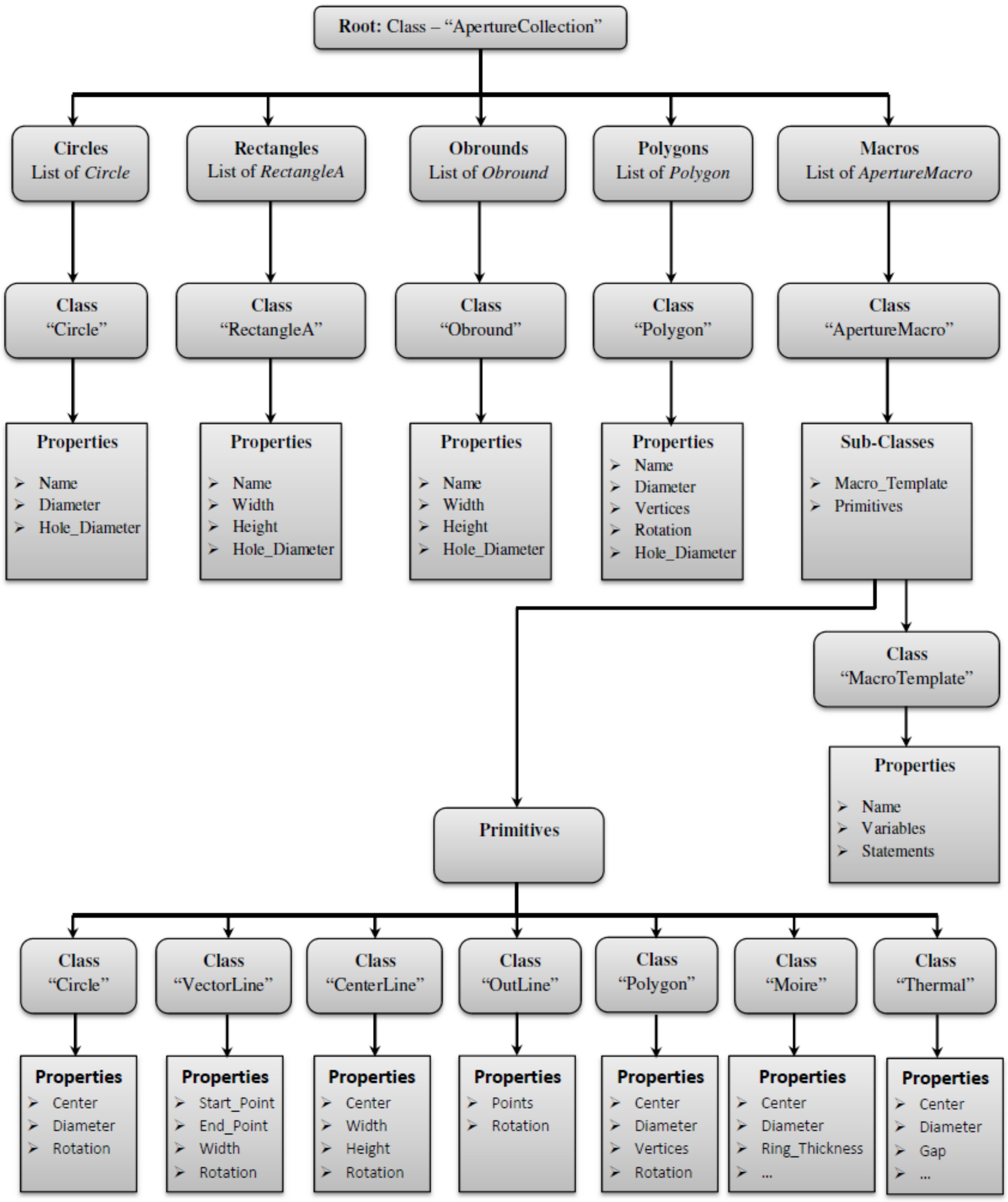
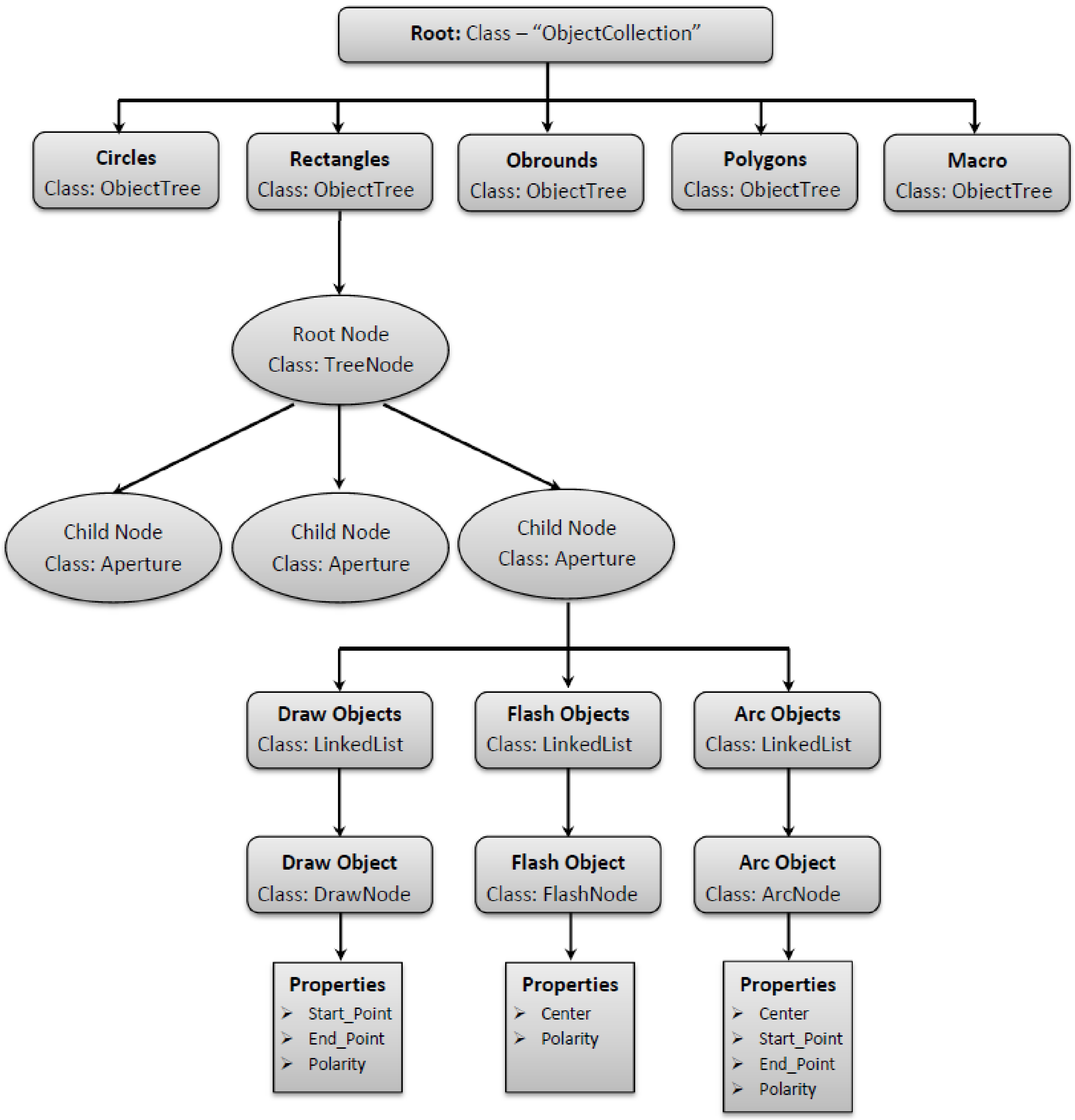

Root: Class – "ApertureCollection"
Circles List of Circle
Rectangles List of RectangleA
Obrounds List of Obround
Polygons List of Polygon
Macros List of ApertureMacro
Class "Circle"
Class "RectangleA"
Class "Obround"
Class "Polygon"
Class "ApertureMacro"
Properties: Name, Diameter, Hole_Diameter
Properties: Name, Width, Height, Hole_Diameter
Properties: Name, Width, Height, Hole_Diameter
Properties: Name, Diameter, Vertices, Rotation, Hole_Diameter
Sub-Classes: Macro_Template, Primitives
Class "MacroTemplate"
Properties: Name, Variables, Statements
Primitives
Class "Circle"
Class "VectorLine"
Class "CenterLine"
Class "OutLine"
Class "Polygon"
Class "Moire"
Class "Thermal"
Properties: Center, Diameter, Rotation
Properties: Start_Point, End_Point, Width, Rotation
Properties: Center, Width, Height, Rotation
Properties: Points, Rotation
Properties: Center, Diameter, Vertices, Rotation
Properties: Center, Diameter, Ring_Thickness, ...
Properties: Center, Diameter, Gap, ...

Figure 12: Object Structure of Aperture Collection

4.3 Graphics Object Collection Structure



Root: Class – "ObjectCollection"
Circles Class: ObjectTree
Rectangles Class: ObjectTree
Obrounds Class: ObjectTree
Polygons Class: ObjectTree
Macro Class: ObjectTree
Root Node Class: TreeNode
Child Node Class: Aperture
Child Node Class: Aperture
Child Node Class: Aperture
Draw Objects Class: LinkedList
Flash Objects Class: LinkedList
Arc Objects Class: LinkedList
Draw Object Class: DrawNode
Flash Object Class: FlashNode
Arc Object Class: ArcNode
Properties Start_Point End_Point Polarity
Properties Center Polarity
Properties Center Start_Point End_Point Polarity

Figure 13: Graphic Objects Collection Data Structure

4.4 Document object

Document object consists of all the graphic objects including standard apertures and macro apertures. Apart from that it consists of basic properties which are valid for the entire file.

Basic document properties

- Coordinate format
  - Integer precision (Integer, 0≤ N≤6)
  - Decimal precision (Integer, 4≤ M≤6)
  - Leading zeros omission (Boolean)
  - Absolute measurement (Boolean, False: incremental measurement)

- Unit (Boolean, True: millimeters, False: inches)
- Level polarity (Boolean)
- Operation mode (Integer, 1≤N≤3)
  - 1 – Interpolation
  - 2 – Move
  - 3 – Flash
- Interpolation mode (Boolean)
  - True – Linear
  - False - Circular
- Quadrant mode (Boolean)
  - True – Single quadrant mode
  - False – Multi-quadrant mode

Document object should support below mentioned methods.

- Add aperture (both standard and macro type)
- Select aperture
- Edit aperture
- Delete aperture

4.5 System Class Diagram

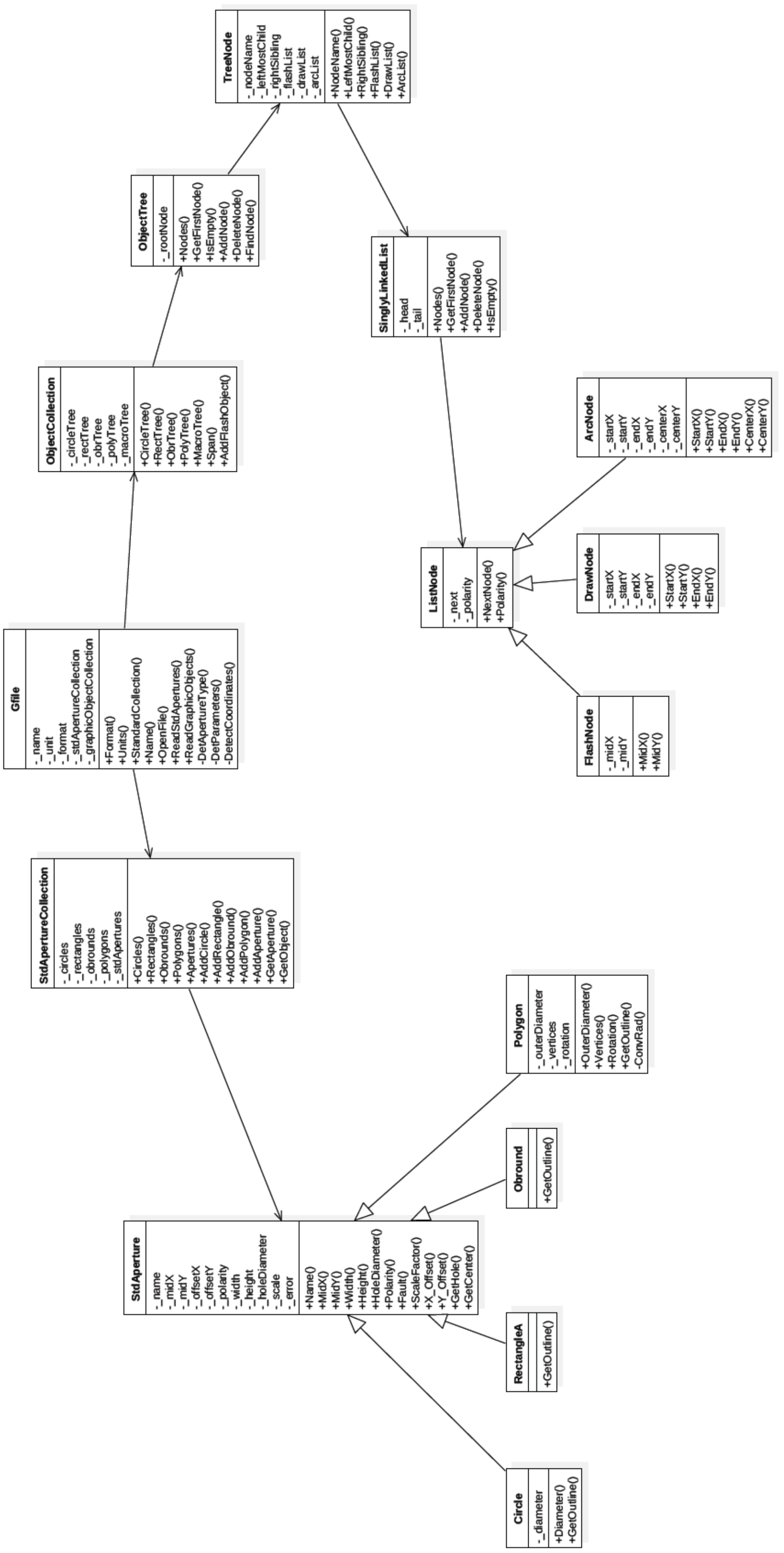


Figure 14 - Class diagram

Chapter 5: Implementation

5.1 Selection of the development plat form

Since the client of this application uses computers with Windows operating system, as it mentioned in the software requirement specification, it was decided to use Visual Basic 2010 as application development environment. The free version of Visual Basic Express 2010 was more than enough for the implementation of required components. Since the Visual Basic 2010 supports Object Oriented Programming concept, it is much easier to develop the designed objects without any issues. Furthermore Visual Studio comes with readymade user interface components such as forms, buttons, menu strips, tool strips and so, no need to add third part plugins or components to develop interactive user interfaces. At the same time most of the people are familiar with Windows based applications and it would be helpful when perform the user evaluation and testing task.

Since the main functionality of this application is 2-D vector graphics manipulation and programming, support of an API which has graphics programming capability is required for the implementation. Visual Basic is distributed with a powerful graphics programming API called GDI+. It has inbuilt very useful graphics objects such as rectangle, ellipse, line, graphics path, region and text etc. in order to develop powerful graphics programming. However some of the features available in GDI+ are not enough for some of the functional requirements mentioned earlier. For example, there are no direct graphic objects in GDI+ for obround and polygon objects which are defined in the Gerber Format Specification. So such objects have to be developed by extending the existing graphic path object.

5.2 Design Considerations

There are objects with similar properties were found in the design phase. For example each basic graphic object has similar properties such as MidX, MidY, HoleDiameter etc. So inheritance concept discussed in software engineering principles was implemented such classes. Generalized class is developed by considering similarities in the sub-classes. This is clearly demonstrated in the class diagram of the entire system.

Encapsulation is another important software engineering design principle which hides some of the internal functionalities of any object. Two objects can only be interacted with properly designed interfaces. Public methods with proper input parameters are created inside the objects in order to interact with other objects. For example when particular graphic object need to display on the screen, paint method of the form object will call the GetOutline() method inside the particular graphic object.

Polymorphism is also another important concept that should embedded into the objects to improve the functionalities and reduce the complexity of the objects. Same function or method could be developed to behave in different ways according to the situation. This concept was implemented in the GetHole() and GetCenter() method in graphic objects. Visual Basic provides “Overridable” and “Overloads” key words to implement such concept in object level.

5.3 Implementation of the User Interface

Common windows application appearance is selected for the interface. Useful features of similar applications also included into the interface especially for color scheme of the display area. Further popular mouse interaction, menu and tool bar based command execution features are built in into the interface. Apart from that, optional ruler object is included into the interface to satisfy the user convenience.

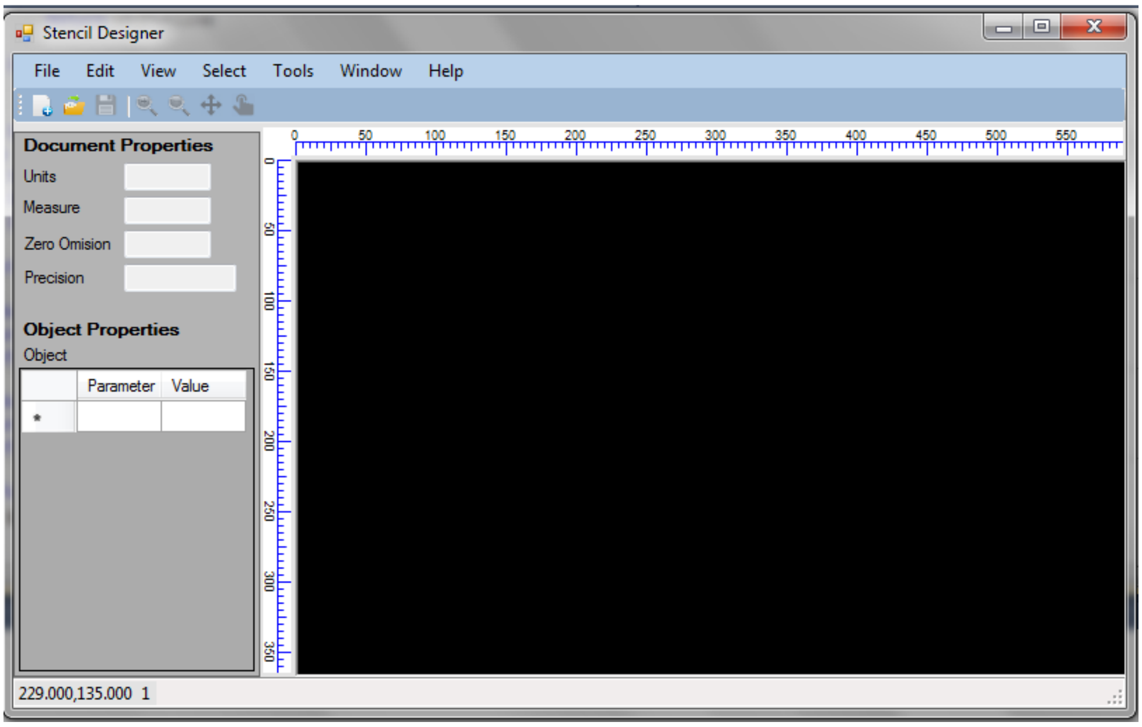
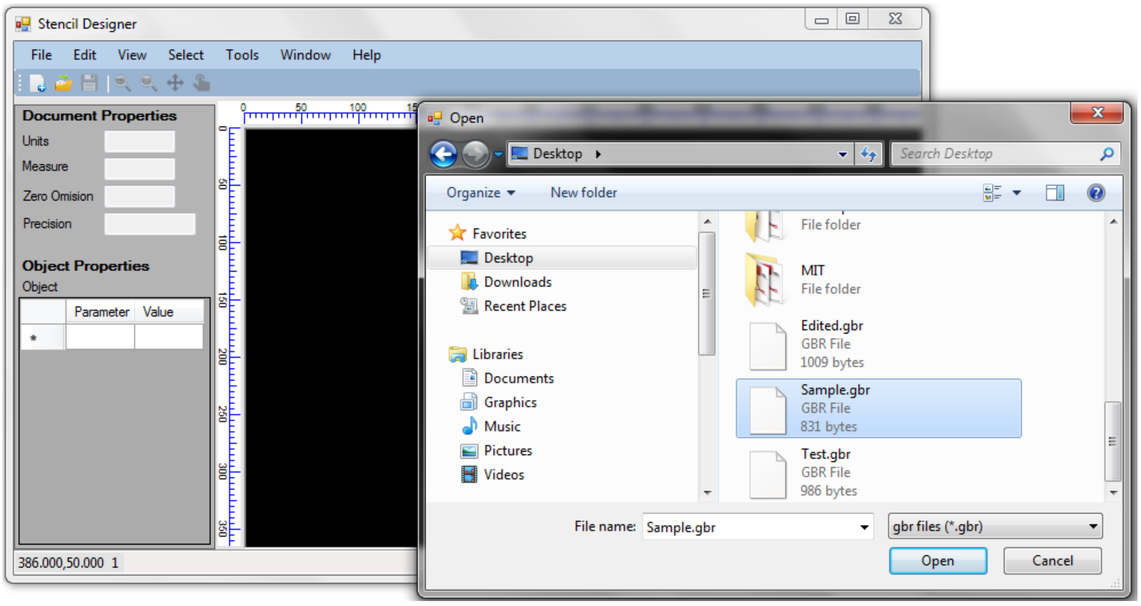


Figure 15: Initial User Interface

Windows File Dialog is used to easily fetch the required file to open.



386.000,50.000 1

File folder

MIT
File folder

Edited.gbr
GBR File
1009 bytes

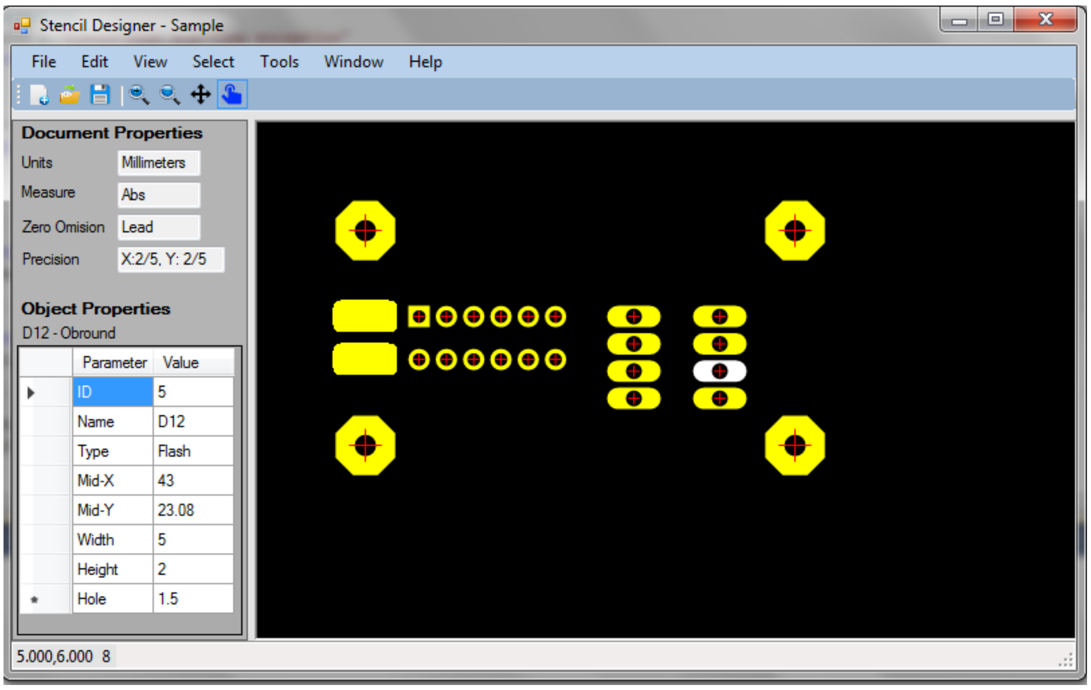
Sample.gbr
GBR File
831 bytes

Test.gbr
GBR File
986 bytes

File name: Sample.gbr
gbr files (*.gbr)

Figure 16: Open file User Interface

High contrast color scheme is used in the display area in order to visualize graphics objects very clearly. Background color is set to black in order to reduce eye irritation, because user will keep close eye contact with the screen for long time period. Default object display color is set to yellow because yellow is comfortable to the eye. The selected objects will displayed using white color from which maximum contrast can be obtained from black color background.



Document Properties

Units	Millimeters
Measure	Abs
Zero Omision	Lead
Precision	X:2/5, Y: 2/5

Object Properties
D12 - Obround

Parameter	Value
ID	5
Name	D12
Type	Flash
Mid-X	43
Mid-Y	23.08
Width	5
Height	2
Hole	1.5

5.000,6.000 8

Figure 17: Display File User Interface

Editing facility is provided through the graphical screen in such a way that, user is able to select any object in the screen using his pointing device of the computer. Then the corresponding object's properties are displayed in the interactive grid view exist in the left pane. Then user can adjust any property in the grid view and the result is updated in real time on the screen.

5.4 Special Implementation Requirements

Selection method of graphic objects required special techniques and algorithms, because standard methods available in GDI+ API do not offer that functionality for complex objects especially for polygons and outline objects.

5.4.1 Point inside a Triangle

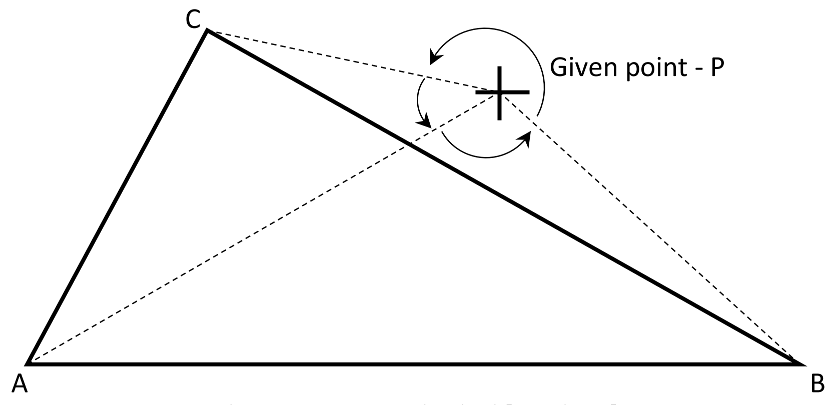


Figure 18: Test Point inside Triangle

In order to include the given point "P" inside the given triangle "ABC", marked three counter clockwise angles (APB, BPC and CPA) should be less than 180°.

5.4.2 Point inside an arbitrary Line Object

Gradual Triangular Scanning Algorithm.

Above mentioned special scanning algorithm is used to check whether a given point included in an arbitrary closed object consist of any number of nodes. According to the Gerber Format maximum number of nodes is limited to 5000. Internal structure of this new algorithm is described below.

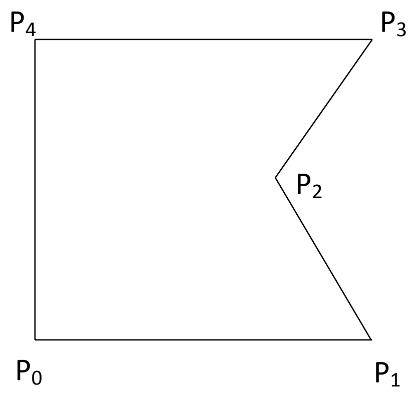


Figure 19: Simple Outline Object

Steps involved in the algorithm:

1. Divide the object into triangles by considering consecutive points starting from first point P_0.
2. Test the triangle is a valid one for the evaluation of given point inside it. In order to evaluate the triangle, two conditions are checked.
    a. Test whether the triangle is inside the object or outside the object.
    b. Ensure that, none of the rest of nodes are included in that triangle.
3. If the validity condition got passed, test whether the given point inside of that triangle.
4. If the point exist, algorithm stop and otherwise tested triangle is marked and proceed to the next triangle.
5. Algorithm is continues until all possible triangles are covered.

Chapter 6: User Evaluation and Testing

6.1 Scope

Scope of this document is to provide the way of selection of primary actor for Gerber Data Editing Software Package and the investigation of minimum capability requirements of intended user and analyze the performance of selected actors.

Second part of this document is focused on testing plan of the proposed system. Testing plan is carried out in three basic levels. In modular level each individual component is tested for possible range of input parameters. Integration testing is performed by interacting two or more components together with intended functionalities. Finally alpha testing is performed for the completed system by using selected group of users.

6.2 User evaluation

6.2.1 Primary actor

Primary actor is the user who is actively interacting with the system. Primary actor should be able to perform the intended functionalities which are mentioned in the Software Requirement Specification (SRS) without any problem. Further he should possess basic requirement which are described in following sections in order to access the system.

6.2.2 Actor selection criteria

Since the software solution is a specific one for an intended purpose, primary actors are specific group of people who involve in CAD / CAM industry. The intended actor should possess the required technical background knowledge to use the system effectively. Actors who do not possess the technical background knowledge also will be able to use the system by reading and understanding the technical documentation of the system.

For an effective interaction with the system and the user, below mentioned skills are expected from the primary actor.

- Basic knowledge of Mathematics, especially Cartesian geometry.
- Ability to interact with Windows applications.
- English knowledge
- Experience in 2D drafting packages or graphic design software
- Handling the pointing device smoothly

6.2.3 User knowledge evaluation

There are several ways of evaluating the knowledge of intended users. Most basic and difficult one is make a structured interview with the intended users. The result of that method is very accurate and fast. However the size of the target group is larger, interview method is difficult and time consuming.

The most easiest and faster method of evaluating primary actor is issuing a questionnaire. Preparing a questionnaire is bit tricky, because it should cover the above mentioned requirements while obtaining short answers from the user.

Two types of questions can be included into a questionnaire.

1. **Open-ended questions:** In this type of question, freedom is given to the audience to answer the question according to their opinion. To answer such question takes some time, so answering rate of such questions are lower.
2. **Closed-ended questions:** Respondent is restricted to select or rate the answer of closed-ended question. Such questions are frequent in questionnaires, because of better response rate and time factor.

Sample questions that can be used to evaluate user's skills:

- ❖ What is your result of G.C.E O/L Mathematics ?
  Ans: 1. A 2. B 3. C 4. S 5. W

- ❖ What is the short cut key used to perform "Cut" operation in MS Excel application?
  Ans: 1. Ctrl + C 2. Alt + X 3. Ctrl + X 4. Ctrl + V

- ❖ "Scale" command of a graphic design software is used …
  Ans: 1. To rotate an object 2. To erase an object 3. To change the size of an object

- ❖ Select the applications you already have experience. If you mark "Other", please mention the name in the given space.
  ☐ Auto CAD ☐ Solid Works ☐ Illustrator ☐ CorelDRAW ☐ Other ……………………….

- ❖ Which hand you operate the mouse?
  Ans: 1. Left hand 2. Right hand

6.2.4 Goal setting

After evaluating the eligibility of primary actor, tasks which are based on system functionalities are given to the actors in order to check whether particular user is capable of completing such tasks effectively. Outcome of such evaluation is used to improve the user interface of the application.

Task evaluation is conducted based on a format shown below.

Actor name:				
No	**Task**	**Result**	**Spent time**	**Comments**
1	Launch the application			
2	Open the given Gerber data file			
3	Zooming the graphic objects			
4	Move the given graphic object to the center of the screen			
5	Obtain center coordinates of given graphic object			
6	Change the size of given object			
7	Move a graphic object to given coordinate point			
8	Undo the previous operation			
9	Save changes as new Gerber file			
10	Exit the application			

Table 5: User evaluation template

Result field can be either “Success” or “Failure” depending upon how the actor achieves the goal.

Spent time field indicate the time in seconds which taken by the actor to perform a given task.

Comment field is allocated for any observations captured during respective task carrying out period.

6.2.5 User performance evaluation

Standard time is defined for each activity mentioned above as follows.

No	Task	Std time (Sec)
1	Launch the application	30
2	Open the given Gerber data file	45
3	Zooming the graphic objects	30
4	Move the given graphic object to the center of the screen	45
5	Obtain center coordinates of given graphic object	60
6	Change the size of given object	90
7	Move a graphic object to given coordinate point	60
8	Undo the previous operation	30
9	Save changes as new Gerber file	45
10	Exit the application	30

Table 6: Standard Task List

User performance is evaluated based on the scores obtained by all the users for the above mentioned activity evaluation score card.

For each “Success” score is given 10 marks whereas “Failure” is given 0 marks.

The score for an activity is calculated as follows.

Score = (10 * [Success | Failure]) * Performance Factor

Where, [Success | Failure] takes values either 0 (for Failure) or 1 (for Success).

Performance Factor is calculated based on the time taken to perform a particular task.

If the Spent time < Std time, performance factor = 1

Else, Performance Factor = 1 – (Spent time – Std time) / Std time

User can obtain maximum of 100 marks if all the activities are successfully completed. If the average score is more than 85marks, the selected groups of users have the fitness to use the system. If the average score is greater than 70 marks, system is fairly well and users need training about the system. If the average score is less than 55, system interface should be improved.

6.2.6 Conclusion

Group of eligible primary actors is selected based on the user knowledge evaluation criteria mentioned above. Then pre-determined list of goals are given to the selected actors as per the Standard activity list. Performance of each actor is monitored and recorded in the defined evaluation score card. Final grading is obtained based on the formula mentioned in the previous section. Final decision is made based upon the result of evaluation criteria.

6.3 Testing plan

6.3.1 Introduction

Testing is conducted in three basic levels.

1. **Unit testing:** Each individual component is tested by changing input parameters. Output is monitored with respect to each combination of input parameters.

2. **Integration testing:** Combine two or many components together and check whether components are responding to the message interactions under basic level functionalities.
3. **Usability / Acceptance testing:** Final or prototype version with proper user interface is tested by using selected group of primary actors.

6.3.2 Unit testing

Unit testing is performed for each and every individual component. This test ensures that control logic in all code paths are executed properly for wide range of input values. This is a kind of white box testing procedure. In practice it is not possible to test all possible input values. Therefore critical set of input values were selected and check the outcome is performed.

In order to perform above mentioned testing procedure, test cases and expected test result should be created. To make all the test cases consistence, standard format is used to create all the test cases.

Test Suite ID	The ID of the test suite which the test case is belongs to.
Test Case ID	Unique identifier of the test case in particular test suite
Test Case Summary	Goal of the test case in summarized form
Prerequisites	Any pre-conditions which should satisfy before execute the test
Test Procedure	Steps of carrying out the test
Test Data	Input parameters which requires to perform the test
Expected Result	The result that expecting from the component under given input
Actual Result	Result received after execution of the test
Status	Final decision of the test. Status could be “Pass” or “Fail”
Remarks	Any comment about test case or test result
Created By	Name of the author of the test case
Date of Creation	Created date of the test case
Executed By	Name of the person who execute the test
Date of Execution	Test case executed date
Test Environment	Hardware / Software / Network environment where the test is executed

Table 7: Test Case Template

To design test cases for individual components, it is required to identify all the individual components in the system and their methods which can be accessed externally. It is easily captured from the class diagram of the system.

In Gerber Data Editing Application, below mentioned individual components were captured from class diagram.

Components used in Standard Aperture set

Component Name	**Public Properties or Methods**
Circle	Diameter
	GetOutline()
RectangleA	GetOutline()
Obround	GetOutline()
Polygon	OuterDiameter
	Vertices
	Rotation
	GetOutline()
StdAperture	MidX
	MidY
	Name
	Width
	Height
	HoleDiameter
	Polarity
	Fault
	GetCenter()

Components used in Macro Aperture set

Component Name	**Public Properties or Methods**
PrimCircle	Diameter
	CenterX
	CenterY
	GetOutline()
PrimVline	StartX
	StartY
	EndX
	EndY
	LineWidth
	GetOutline()
PrimCline	CenterX
	CenterY

	Width
	Height
	GetOutline()
PrimOline	Vertices
	CodPoints
	GetOutline()
PrimPolygon	Vertices
	OuterDiameter
	CenterX
	CenterY
	GetOutline()
MacroPrimitive	Exposure
	Comment
	Rotation

Object nodes:

Component Name	**Public Properties or Methods**
ListNode	NextNode
	Polarity
FlashNode	MidX
	MidY
DrawNode	StartX
	StartY
	EndX
	EndY
ArcNode	StartX
	StartY
	EndX
	EndY
	CenterX
	CenterY

Since all test cases create this document too much lengthy, sample test case is described in details in this report. All remaining test cases should follow the similar procedure.

Test the “GetOutline” method in “Polygon” class

Test Suite ID	TS015
Test Case ID	TC008
Test Case Summary	To check whether the Polygon object return correct outline when “GetOutline” method is called.
Prerequisites	An instance of a polygon object should be created
Test Procedure	1. Create an instance of polygon object by providing input parameters to polygon class. 2. Inspect the output parameters by introducing program break points 3. Visualize the output on the screen
Test Data	Instance-1: MidX = 50, MidY = 50, OuterDiameter=60, Vertices = 6, Rotation= 0 Instance-2: MidX = 50, MidY = 50, OuterDiameter = 60, HoleDiameter = 20, Vertices = 6, Rotation = 30 Instance-3: MidX = 50, MidY = 50, OuterDiameter = 60, Vertices = 3, Rotation= 0 Instance-4: MidX = 50, MidY = 50, OuterDiameter = 60, Vertices = 2, Rotation=0 Instance-5: MidX = 50, MidY = 50, OuterDiameter = 60, Vertices = 13, Rotation=0
Expected Result	Instance-1: P0=(80,50), P1=(65,24), P2=(35,24), P3=(20,50), P4=(35,76), P5=(65,76)
Actual Result	Instance-1: P0=(80.0,50.0), P1=(65.0,24.02), P2=(35.0,24.02), P3=(20.0,50.0), P4=(35.0,75.98), P5=(65.0,75.98). Output is included in the appendix.
Status	Pass
Remarks	Only the first instance is evaluated.
Created By	Damith
Date of Creation	12.01.2018
Executed By	Damith
Date of Execution	12.01.2018
Test Environment	OS: Windows 7 Environment: Microsoft Visual Studio Express - 2010

Table 8: Test case template

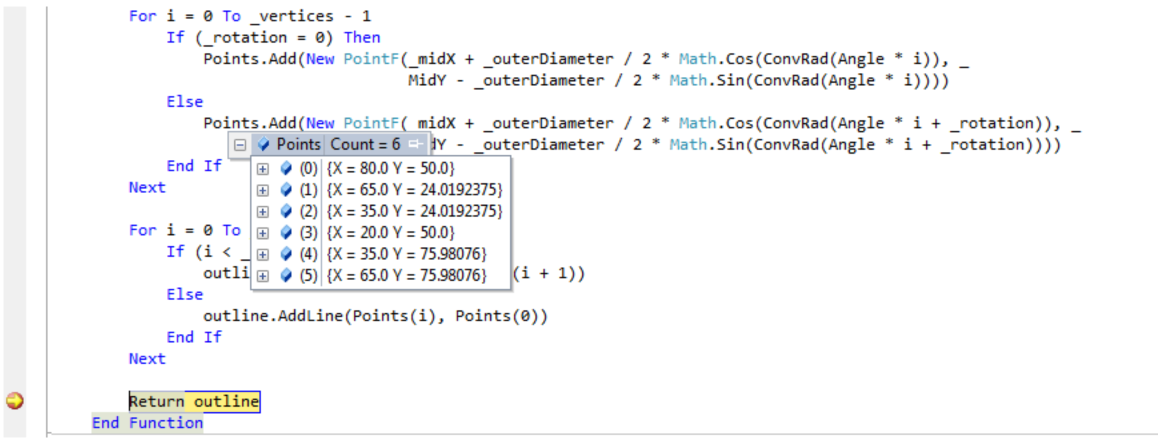
Appendix: Test results

Test Suite ID: TS015

Test Case ID: TC008

Instance-1:

Application branch output:



```
        For i = 0 To _vertices - 1
            If (_rotation = 0) Then
                Points.Add(New PointF(_midX + _outerDiameter / 2 * Math.Cos(ConvRad(Angle * i)), _
                                           MidY - _outerDiameter / 2 * Math.Sin(ConvRad(Angle * i))))
            Else
                Points.Add(New PointF( midX + _outerDiameter / 2 * Math.Cos(ConvRad(Angle * i + _rotation)), _
                                        dY - _outerDiameter / 2 * Math.Sin(ConvRad(Angle * i + _rotation))))
            End If
        Next

        For i = 0 To
            If (i < _
                outli                                  (i + 1))
            Else
                outline.AddLine(Points(i), Points(0))
            End If
        Next

        Return outline
    End Function
```

```
Points  Count = 6
  (0) {X = 80.0 Y = 50.0}
  (1) {X = 65.0 Y = 24.0192375}
  (2) {X = 35.0 Y = 24.0192375}
  (3) {X = 20.0 Y = 50.0}
  (4) {X = 35.0 Y = 75.98076}
  (5) {X = 65.0 Y = 75.98076}
```

Figure 20: TC008 Output

Screen output:

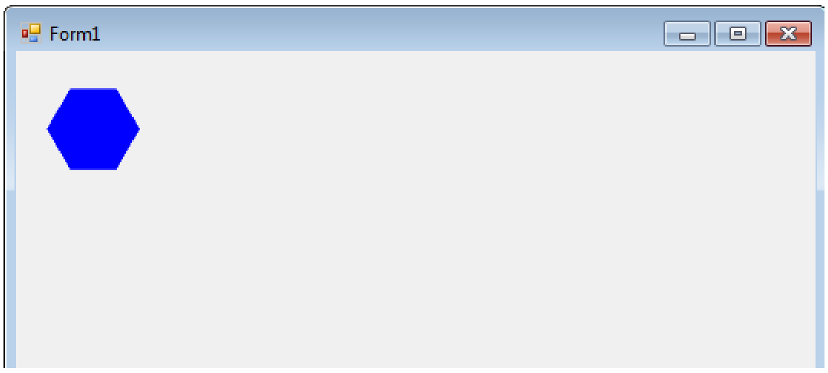


Figure 21: TC008 Screen Output

6.3.3 Integration testing

Integration testing is planned to perform for combined objects which are composed of two or more basic components. Same testing procedure which was described under the unit testing is extended for integration testing. In this section, more weight is given to the methods in which interaction of messages involved between objects rather than parameter testing.

As described in previous section, secondary level of object also derived from class diagram of the system. Interactions are captured from the activity diagram of the system. Then new set of test cases are developed for secondary level to perform the integration testing.

Secondary level of objects exist in the proposed system is listed as follows.

Component Name	**Public Properties or Methods**
ApertureCollection	AddCircle()
	AddRectangle()
	AddObround()
	AddPolygon()
	AddMacro()
	GetObject()
ObjectCollection	CircleTree
	RectTree
	ObrTree
	PolyTree
	MacroTree
ObjectTree	Nodes
	IsEmpty()
	GetFirstNode()
	AddNode()
	DeleteNode()
	FindNode()
TreeNode	NodeName
	LeftMostChild
	RightSibling
	FlashList
	DrawList
	ArcList
SinglyLinkedList	IsEmpty()
	GetFirstNode()
	FindNode()
	DeleteNode()
	EditNode()
Gfile	Name
	Unit
	Format
	ApertureCollection
	ObjectCollection
	OpenFile()
	ReadApertures()
	ReadObjects()

6.3.4 Alpha testing

Alpha testing is the last testing procedure planned for this software solution. Full functional system is given to a selected set of users who selected from user selection criteria mentioned in previous section. A task list is given to the selected users and observes the performance of each user during the completion of the tasks. Difficulties and errors which will occur during the testing are recorded for system improvement. This testing is mainly focused on usability related aspects. Improvement in the user interface is tolerated after alpha testing. Component level changes would be less frequent during this test.

Task list which is planned for the alpha test as follows:
All the users will be given a sample Gerber data file.
Users will have to do below mentioned activities.

1. Open the given Gerber data file.
2. Select the given graphic objects
3. Perform the given modification for the selected objects
4. Save the modified file as new Gerber file.

Evaluation is carried out during the test as well as after the test. During the test, any difficulty encountered by any user or any error occurred during the test is captured. The accuracy of the work is checked after the test by evaluating resulting Gerber files created by each user.

6.3.5 Summary

In practice, getting completely bug free solution is merely impossible. But proper control of the activities carried out during software development life cycle will lead to minimize the amount and severity of bug existence. If any bug is identified, it should be corrected before next stage. So most of the software development projects follow the iterative incremental approach, while minimizing bug and fulfilling expected requirements.

Chapter 7: Conclusion and Future Work

Unlike development of traditional software applications such as Access Control Systems, Payroll System, Library System which has plenty of examples of similar systems, this project was really hard at the beginning and every component had to be developed from scratch. But the enthusiasm of facing this challenge drove me to achieve the goal successfully at the end. Evolutionary prototyping software development approach was followed throughout this project to build the system in order to meet the expected requirements.

Gerber data file has a standard format and file structure is completely defined in the Gerber Format. So the application should be strictly adhere to the format and the Bible of this application is the Gerber Format Specification. So initially it was decided to follow the object oriented programming concept for the development lifecycle of this application. At the initial stage basic graphic objects which are defined in the Gerber Format was developed. Then the required properties and methods were embedded into the objects. Finally all the components were integrated together in order to meet the functional requirements.

The development process was stuck several times for few weeks due to the technical issues encountered. When developing macro objects, project was halted for two weeks. Macro objects had a complex data structure than normal standard objects. To generate the proper shape of macro objects on the screen was tedious task because single macro object is composed of several primitive shapes. Finally region object of the GDI+ framework was used to render such complex objects on the screen. The boundary of such region objects are not sharp as basic shapes in which Graphics Path object is used to render the shape.

Creating the selection method inside each and every graphic object also made big trouble during the development stage. Especially for macro objects consist of outline primitives. That problem was solved after investigation of new algorithm. Struggling with such stuff supported to gain massive experience and confidence about the learnt concepts and theories in the classroom.

Due to the limited number of hours that could be allocated for the project, each and every feature defined in the Gerber Format specification was unable to integrate into the application. However basic functional requirements such as Open and display a Gerber file, provision of editing facilities and save the edited file in Gerber format was included into the developed evolutionary prototype. There is a room kept in the design of the application in order to integrate rest of the features. Now the system is fully functional for the flash objects which are defined in the specification. In future, draw objects and arc objects need to be integrated into the application. Editing feature of macro objects also planned to introduce in future. Furthermore layer object will be introduced in order to manipulate two or more Gerber files together. Finally a user is given a capability to create new graphic objects and deleting existing graphics objects while complying with Gerber Format with the aid of toolbars.

Finally involving in this project not only causes to gain lots of experience and knowledge but also practice me to face challenges and solve problems logically. Some times when I was stuck with some technical issues, project was a headache but somehow a solution found, it was real amusement and generate a motivation to achieve the target even better.

References

[1] Ucamco Development Staff, The Gerber Format Specification, Ucamco NV, 2017.

[2] Mahesh Chand. *Graphics programming with GDI+*. 1st Edition. Boston, MA. Addison-Wesley. 2005

[3] GraphiCode, USA. [Online]. Available:

https://www.graphicode.com/GC-Prevue

[4] Viewplot. [Online]. Available:

http://www.viewplot.com/