ISSN 2229-5518

An Efficient and Simplest Algorithm for Generating Diagrams

Tabinda Sarwar, Uzma Arif, Wajiha Habib Samana Zehra

Abstract—Whenever a system is to be designed and modeled, diagrams play a key role in the context. As diagrams enable us to understand, visualize and communicate concepts without ambiguity. Thus for the purpose many diagramming softwares are available. Different softwares uses different methodology for design and development but with the time the design is becoming complicated day by day and with the increased complexity also increases the learning time for the users. In contrast, software based on easy algorithm is easily understandable by the user. This paper presents a simple and efficient algorithm for designing "Diagram Generator" software (DG). The algorithm will describe the methodology for generating graphical shapes, which can be combined to produce diagrams like Flowchart, Block diagrams, Organizational charts, Data Flow diagrams, Entity-Relationship diagrams and many more.

Index Terms— Link Lists, Flow Charts, Organizational Chart, ER diagrams, Block diagrams, Click and Draw.

1 INTRODUCTION

DIAGRAMS, a two dimensional geometric symbolic representation of information according to some visualization technique [1], play a significant role in analysis and design of a system. Diagrams are an excellent means of communicating and clarifying customer requirements, to avoid misinterpretations and ambiguities.

Diagrams also help in designing and visualizing the target system to be developed (system architectures for instance). Examples of commonly used diagrams are flow charts, block diagrams, organizational chart, network diagrams, Entity-Relationship diagrams, bar chart, pie chart, graphs and UML (Unified Modeling Language) diagrams [2].

In today's age, different diagrams have different usability. For example, Entity Relationship diagrams is by far the most common way to express the analytical result of an early stage in the construction of a new database [3], Flowcharts are used in analyzing, designing, documenting or managing a program or process [4]. Because of the frequent use and significance of diagrams in various fields, different softwares are available and still more softwares are being developed but due to the complex design the usability of the software for the novice user is low.

A survey was conducted in University of Engineering and Technology Taxila, Pakistan according to which the most commonly used software for diagram generation process is "Paint". So in this paper, a simple and efficient algorithm is represented, from developer's point of view, which will generate Flow charts, Block Diagram, Organizational charts and ER diagrams. The algorithm can be extended to include more graphical shapes to accommodate more diagrams.

2 IMPLEMENTATION COSIDERATION

2.1 Link Lists

For representing the graphical object a very simple approach is used. As every programming language supports "Rectangle", so a "Rectangle" will be used to derive more graphical shapes (instead of line that will be used for connecting graphical objects). So at the back end each shape will be saved as rectangle instead of line, for which start and end point will be saved.

To keep track of all the graphical objects a link list is created. The link list stores the following attributes:

- The Graphical object / shape
  - For all graphical objects except line, rectangle is saved
  - For line its start and end points are saved
- Text related to the graphical shape
- Next element (graphical object / shape) of the list
- Preview element (graphical object / shape) of the list
- Other formatting information (e.g. color, font style, size)

2.2 Building Blocks of Diagrams, Graphical Objects

Following lists elaborate the graphical shapes which are derived from rectangle

- Rectangle will be used as it is (e.g. rectangle represents a "block" in Block diagrams)
- Parallelogram is generated by connecting the following points generated from a rectangle "rc"
  - Point1 = rc.X, rc.Y (where rc.X and rc.Y are the x and y-coordinates of the upper left corner of "rc")
  - Point2 = rc.X + rc.Width – (integer that

represents the slope for parallelogram sides), rc.Y (where rc.Width is the width of "rc")
    - Point3 = rc.X + rc.Width, rc.Y +rc.Height (where rc.Height is the height of "rc")
    - Point4 = rectangle.X + (integer that represents the slope for parallelogram sides), rectangle.Y + rectangle.Height
- Ellipse is easily generated by specifying the rectangle without any further computation, as ellipse is drawn within the rectangle
- Diamond is generted by connecting the points Point11, Point12, Point13, point14 and Point11 respectively.These points are derived from rectangle "rc"
    - Point1 = rc.X, rc.Y
    - Point2 =rc.Left (where Left is the x-coordinate of left edge of "rc"), rc.Bottom (where Bottom is the sum of rc.Y and rc.Height)
    - Point3 =rc.Right (where right is the sum of rc.X and rc.Width), rc.Bottom
    - Point4 =rc.Right, rc.Top (where Top is the Y-coordinate of top edge of "rc")
    - Point11.X = (Point1.X + Point4.X) / 2
    - Point11.Y = (Point1.Y + Point4.Y) / 2
    - Point12.X = (Point1.X + Point2.X) / 2
    - Point12.Y = (Point1.Y + point2.Y) / 2
    - Point13.X = (Point2.X + Point3.X) / 2
    - Point13.Y = (Point2.Y + Point3.Y) / 2
    - Point14.X = (Point3.X + Point4.X) / 2
    - Point14.Y = (Point3.Y + Point4.Y) / 2

Similarly other graphical shapes can also be generated by comuting points from rectangle.

- Line is generated by using two points, starting point "start" and ending point "end".

2.3 Figures Tracker of Graphical Object for Resizing/Editing

Editing and resizing the graphical shape is also simplified, as the basic rectangle will be used as a tracker for all graphical objects except line. Each graphical object is saved as rectangle "rc" using which the trackers are calculated.

- 8 rectangles are created, top_Right, top_Left,top_Center, bottom_Center, bottom_Left, bottom_Right, center_Right and center_Left with initial value=null
- The "rc" is retrieved from the link list
- The rectangles are created of size=5 and its position (x, y) are found as follows: top_Left = (rc.X,rc.Y) where "rc.X" and "rc.Y" is the X and Y-coordinates of node "rc" respectively
    - top_Center =(rc.X + (rc.Width / 2), rc.Y) where "rc.Width" is the width of "rc"
    - top_Right = (rc.X + rc.Width, rc.Y)
    - center_Left =(rc.X , rc.Y + (rc.Height / 2)) where "rc.Height" is the height of "rc"
    - center_Right =(rc.X + rc.Width, rc.Y + (rc.Height / 2))
    - bottom_Left = (rc.X , rc.Y + rc.Height)
    - bottom_Center =(rc.X + (rc.Width / 2), rc.Y + rc.Height)
    - bottom_Right =(rc.X + rc.Width, rc.Y + rc.Height)

These eight points will be used for resizing the respective sides.

The tracker for line is different from the rectangle. The line is saved as two points in link list, starting point "ls" and ending "le"

- 2 rectangles are created, Right and Left with initial value=null
- "ls" and "le" are retrieved from link list
- The rectangles are created of size=5 and its position (x, y) are found as follows:
    - Left = (ls.X, ls.Y) where "ls.X" and "ls.Y" is the X and Y-coordinates of starting point "ls" of line respectively
    - Right = (le.X, le.Y) where "le.X" and "le.Y" is the X and Y-coordinates of ending point "le" of line respectively

2.4 Copyright Form Generating Diagrams by using Click and Draw Methodology:

Three modes are used in diagram generation process; NONE when the software is idle, doing nothing, DRAW when the graphical object is to be drawn and SELECT when the drawn graphical object is to be edited. Figure 1 represents the algorithm when the graphical object is to be drawn in a bitmap image "DrawingImage".

2.5 Editing/Resizing Graphical Objects:

FindSelectNode() function used in Figure 2 works in a way that if the point belongs to the graphical object then it is selected.

For inserting text inside the graphical objects algorithm described in Figure 3 is followed.

ISSN 2229-5518

2.6 Figures:

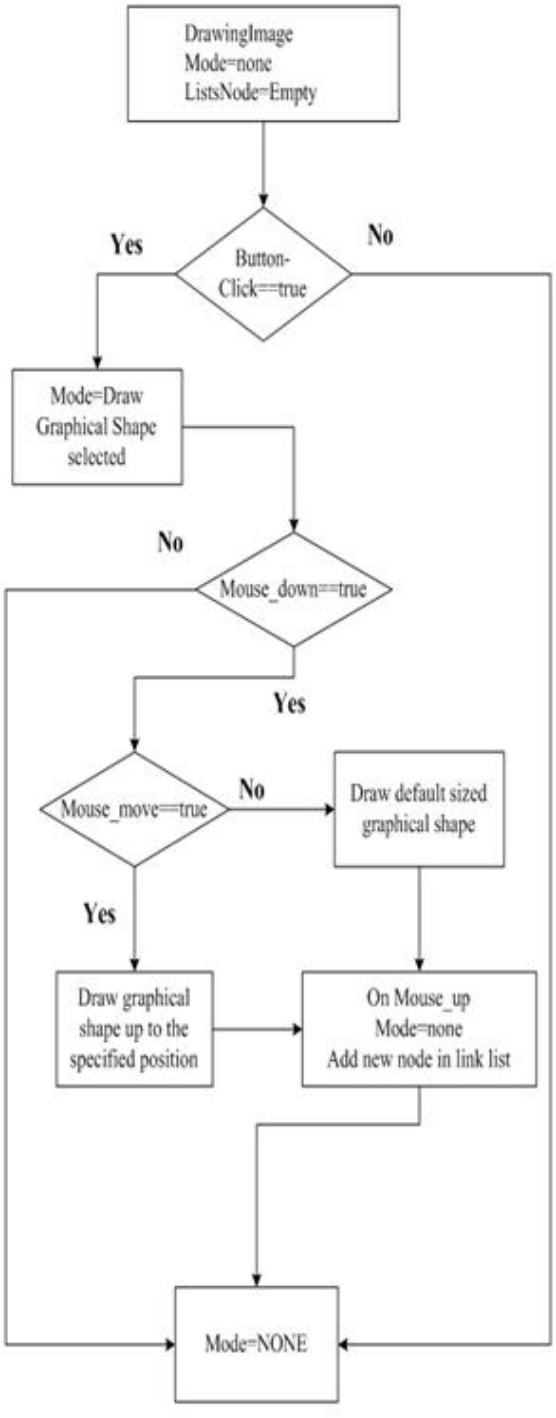

Figure 1: Algorithm for Graphical Object Creation

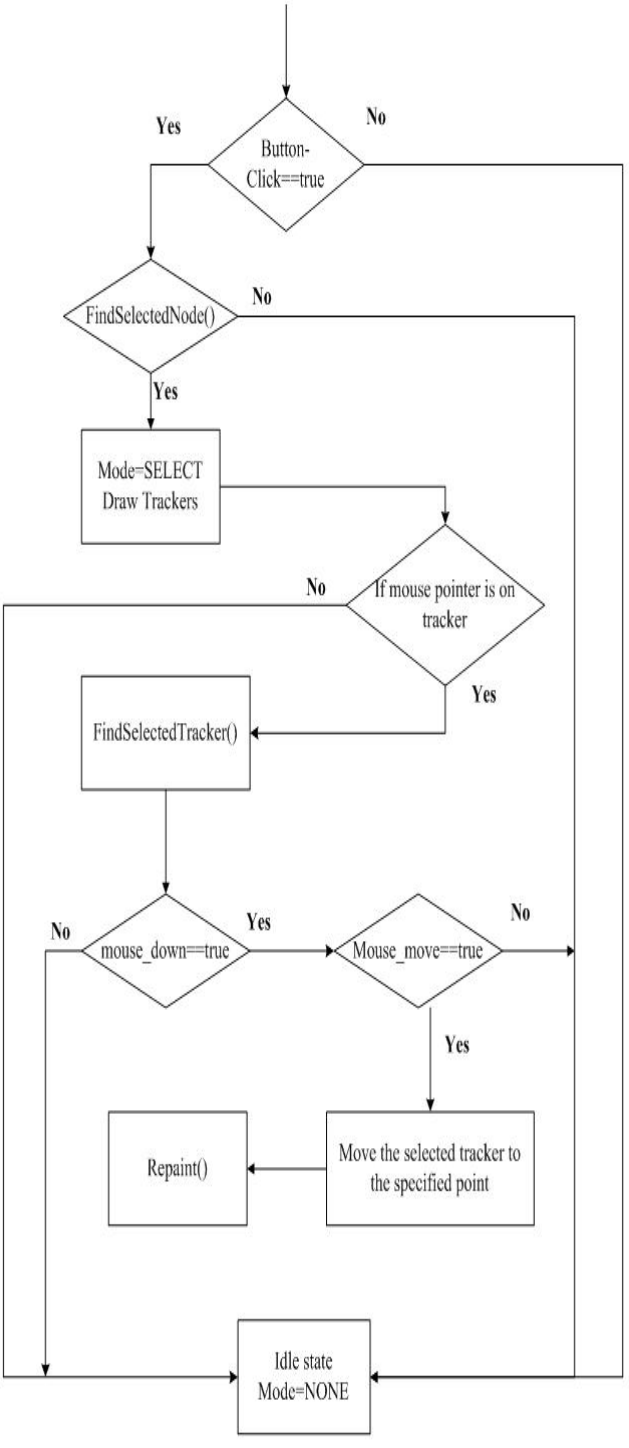

Figure 2: Algorithm for Resizing/Editing Graphical Objects

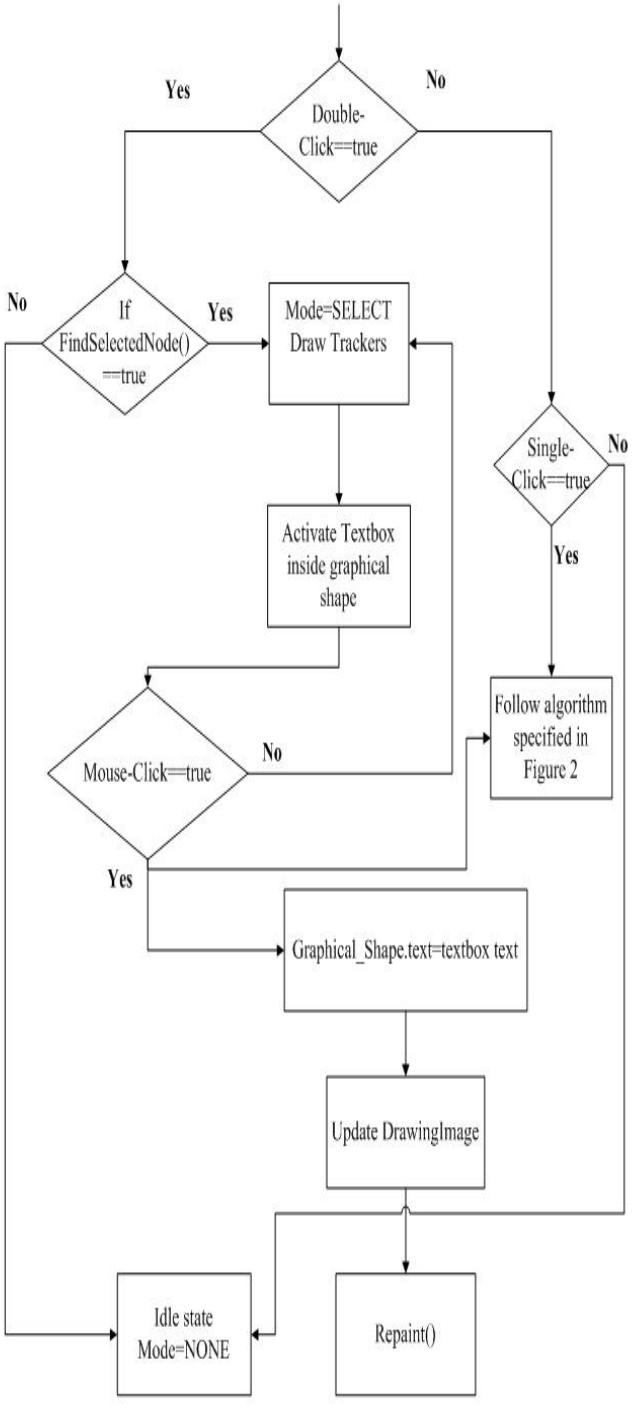

Figure 3: Algorithm for Editing Text of a Graphical Object

3 SUMMARY

Algorithm for DG software is the simplest and efficient algorithm that can be extended for any type of diagrams ranging from simple flowcharts to complex UML diagrams. The algorithm discusses the methodology for representing, generating, editing/resizing and maintaining the graphical objects.

REFERENCES

[1] "Stanford Encyclopedia of Philosophy," http://plato.stanford.edu/entries/diagrams/, April 2011.

[2] "Wikipedia, the free encyclopedia," http://en.wikipedia.org/wiki/Diagram, April 2011.

[3] SEVOCAB, "Software and Systems Engineering Vocabulary," http://pascal.computer.org/sev_display/search.action;jsessionid=989BD03B195021410505EFB99050FCFF, April 2011.

[4] Peter Chen, "Entity Relationship Modeling", http://www.devarticles.com/c/a/Development-Cycles/Entity-Relationship-Modeling/, April 2011.