Replica Technique for Geometric Modelling

HAMEED ULLAH KHAN
Department of Information Systems,
College of Computer & Information Sciences,
King Saud University,
Riyadh, Kingdom of Saudi Arabia
hukhanafridi@yahoo.com hkhan@ksu.edu.sa

Abstract:- Computer graphics incorporate almost everything represented on computers with the exception of text and sound. A graphic approach is developed for the construction of figures by using this replica technique. In this context, an intelligent algorithm is developed to construct geometric outlines. The algorithm is capable of performing other analysing functions such as filtering, prediction, recognition and translation.

Keywords:- Computer Graphics, Computational Geometry, Digital Graphics.

1 Introduction

This paper presents a way of controlling the un-ordered pixels on the screen. If this is intelligently [1] handled then the results will be maximized. In graphics, when the pixel activity is brought under control, it is possible to draw a figure, scale its dimensions and rotate its position (animation). Principally, this paper focuses on the idea of producing a figure by replicating and activating the distribution of pixels on a screen. The figures provide an accuracy of information with which to activate the pixels whereas an image is just the appearance or look of the figure [2, 3].

On the other hand, the proposed algorithm is intelligent because it calibrates and then activates its counter pixels on the screen based on earlier calculations. These are carried out only on one side of the *Central Point* (*Cp*).

A graphical and mathematical approach is also applied as the basis for the logical operations in the proposed algorithm, operating on the concept of ordering pixel activation to produce significant results.

The midpoint method is the de facto standard to generate a conic in a right-handed two dimensional Cartesian coordinate system. In the midpoint the sign of the decision variable selects the best point out of the two candidate points. Therefore, one uses an integer incremental algorithm [4].

The Digital Differential Analyzer (DDA) is a scan conversion line algorithm based on calculating either ∆Y or ∆X, as shown in equations:

$$y = mx + c \qquad (1)$$

and

$$m = Y2 – Y_1 / X_2 - X_1 \qquad (2)$$

Where m is the slope of the line and c is the y intercepts [5].

The DDA algorithm appears efficient, but it requires a floating-point addition for each pixel generated. The new approach to derive a line rasterization algorithm that avoids all the floating-point calculations and has become the standard algorithm used both in hardware and software rasterizers [6].

In this paper, section 2 provides the detailed explanation of the proposed algorithm with examples. In section 3 simulations are carried out along with results on all the three algorithms. Section 4 provides conclusion and in the next section future work is communicated.

2 Proposed Algorithm

A preliminary input to the algorithm is generated to start the production of a geometric figure. This technique allows the creation of uniform, high-resolution, consistently-drawn figures. This approach relieves the user of the burden of collecting large numbers of figures and increases the quality of user-driven figure-based modeling systems. The research described in this paper permits both the quantification of sampling effects in figure-based computer graphics systems as well as the correction of degradation in figure-based consistency.

The idea of generating simultaneous results in graphics is very important because the replica is used for drawing a figure. We know that figures mainly fall into two categories [7, 8]; moving and still. In the case of moving figures, there is movement either in the foreground or the background, or it may be in both, whereas in the case of still figures, both the foreground and the background are stationary [9]. In other words, the latter case is simply the activation of pixels based on different approaches [12-15]. In this paper we mainly consider the later case.

In this approach the computer screen is divided into four quadrants [10, 11], as shown in Figure 1.

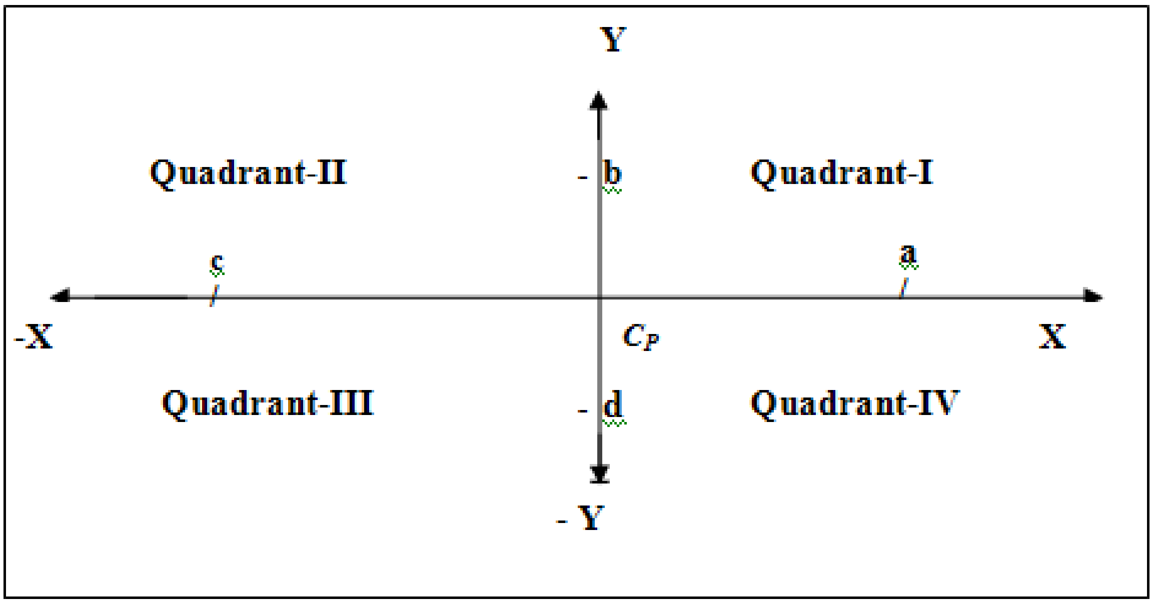


Figure 1: The relative positioning of four axes

In Figure 1, the *Central Point* (*Cp*) also called a *Seed Point* [6] for a figure and the pixel position will start on only one axis; let's say starting from Quadrant-I at point 'a' at X-axis. Then its replica is placed simultaneously in Quadrant–II at position 'b' at Y-axis. Similarly, a replica of Quadrant-II is placed in Quadrant-III at position 'c' at -X-axis and in return at position 'd' at –Y-axis in Quadrant-IV.

During the sketching of any figure, let's assume 'a' is the starting point on the X-axis. To activate the next pixel, 'a' will be incremented by 1 each time giving (a+1, a+2 ….a+n), where n is any positive integer and this will continue until it reaches point 'b' on the Y-axis. Furthermore, a+n = b indicates that the operations for the Quadrant-I have been completed. But it depends on the figures' shape. In case of square and similar shapes the position of coordinate (a_n, b_n) is considered, except in case of circle, see shown in Figure 2.

2.1 Steps for Central Point Selection

The proposed algorithm always first selects the *Central Point* (*Cp*) for a given figure on the following hypothesis:

- **Select the value for *Cp***
  (Explanation: a positive integer, the pixel position for the starting point on the computer screen)
- **Equation 1 = *Cp* + 1**
  (Explanation: Equation #1, adds 1 pixel each time to the starting position on the positive side)
- **Equation 2 = *Cp* - 1** (Explanation: Equation #2, subtracts 1 pixel each time from the starting position on the negative side)
- **Equation 3 = (*Cp* + 1) + (*Cp* - 1)**
  (Explanation: Equation #3, adds Equation 1 and Equation 2 to find the similarity on the axes)

- **Equation 4 = (*Cp* + 1) + (*Cp* - 1) / 2**
  (Explanation: Equation #4, divides Equation 3 by 2 to find the starting value of *Cp*)

In all the above mentioned equations, the position of the initial or first pixel from the *Central Point* on the starting axis is depicted. The next step is to choose the required figure. Because figures have different shapes, each figure has its own approaches/procedures/modules.

2.2 Steps for Figure Construction and selection

The proposed algorithm sub-routine would be adjusted according to the choice of figure to construct. The proposed algorithm has the following main steps:

- **Select the *Central Point***
  (Explanation: Central point is selected on the computer screen)

- **Choice for the figure to construct**
  (Explanation: select the required figure which is going to be drawn)

- **Choice for using the number of quadrants**
  (Explanation: In some figures we select only 2 quadrants instead of 4, for example in case of Triangle we use only 2 quadrants)

- **Activate the initial pixel at one axis**
  (Explanation: On the X-axis (or any axis) for Quadrant–I (or any quadrant) give a starting value to pixel)

- **The algorithm will repeat step 3 for Quadrant-II (Q-II)**
  (Explanation: If Quadrant-I is selected, the replica will be on the -X-axis.)

- **Activate pixels on the –X-axis by using step 4 for Quadrant-III (Q-III)**
  (Explanation: After Quadrant-II, the replica will be on the -X-axis.)

- **Repeat step 5 for Quadrant-IV (Q-IV) on the X-axis**
  (Explanation: After Quadrant-III, the replica will be on the -Y-axis.)

- **Construction of the figure starts on the axis from the originating Quadrant, usually Quadrant-I X-axis**
  (Explanation: No matter what axis is selected, the operation will complete one step before the axis where it originated)

- **Operations are repeated until on the Quadrant-I (Q-I) reaches the coordinate (a_n, b_n) for Y-axis.** Similarly, on the Quadrant-II reaches the coordinate (b_n, c_n) for Y-axis, and on the Quadrant-III reaches the coordinate (c_n, d_n) for –Y-axis and finally on the Quadrant-IV reaches the coordinate (d_n, a_n) for –Y-axis.

- **Operations are repeated until the line starting from 'a' reaches at coordinate (a_n, b_n) touches Y-axes at position 'b' and thus completes the Quadrant-I.** Similarly the line starting from point 'b', 'c' and 'd' also touch at their respective positions, such as, (b_n, c_n), (c_n, d_n) and (d_n, a_n) respectively.

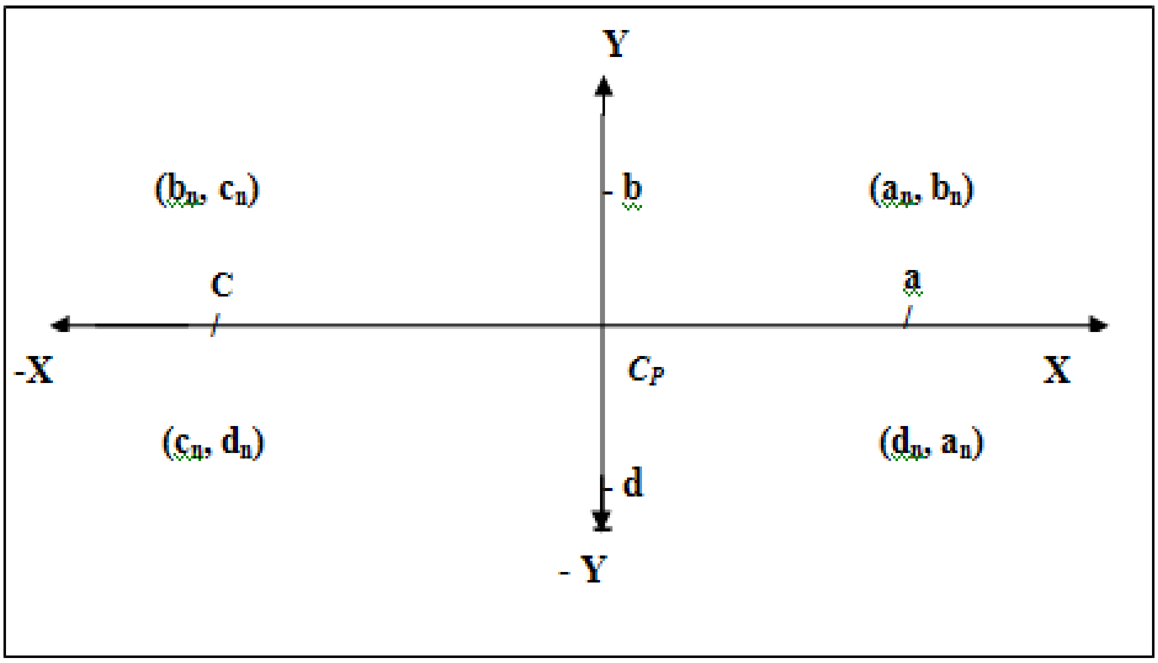


Figure 2: Positioning of four coordinates

All of the above mentioned steps are considered, depending on the kind of figure to draw, so the algorithm will be adjusted according to the choice of figure to be made and then the steps will be completed until they yield a complete figure.

2.3 Example 1: Construction of Figure by involving 4 Quadrants

Let's take an example to develop a square of size 6x6. Where 6 is an integer number starting from *Cp* in all directions as shown in Figure 3, which is assumed as the number of pixels.

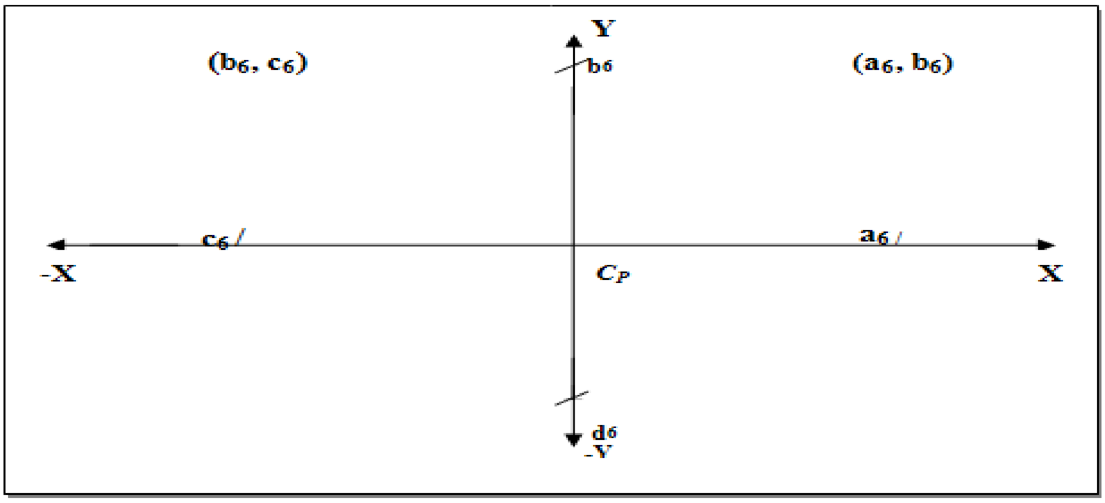


Figure 3: Pixel Activation on Quadrant-I

The data generated by the algorithm for the figure for the Quadrant-I is shown in Table1

.

S. No	Quadrant-I	Figure	Comments
1.	(a_6, b_0)	Square	Starting Point Q-I
2.	(a_6, b_1)	- do -	
3.	(a_6, b_2)	- do -	
4.	(a_6, b_3)	- do -	
5.	(a_6, b_4)	- do -	
6.	(a_6, b_5)	- do -	
7	(a_6, b_6)	- do -	Coordinate Position (a_6, b_6) reaches
8.	(a_5, b_6)	- do -	
9.	(a_4, b_6)	- do -	
10.	(a_3, b_6)	- do -	
11.	(a_2, b_6)	- do -	
12.	(a_1, b_6)	- do -	
13.	(a_0, b_6)	- do -	Y-axis reached

Table 1: Data for pixel activation only for Quadrant-I

The operations will take place for other quadrants simultaneously to select the pixel, based on Quadrant-I values, as shown in Table 2.

S. No	Quadrant-II	Quadrant-III	Quadrant-IV	Figure	Comments
1.	(b_0, c_6)	(c_6, d_0)	(d_0, a_6)	Square	
2.	(b_1, c_6)	(c_6, d_1)	(d_1, a_6)	- do -	
3.	(b_2, c_6)	(c_6, d_2)	(d_2, a_6)	- do -	
4.	(b_3, c_6)	(c_6, d_3)	(d_3, a_6)	- do -	
5.	(b_4, c_6)	(c_6, d_4)	(d_4, a_6)	- do -	
6.	(b_5, c_6)	(c_6, d_5)	(d_5, a_6)	- do -	
7.	(b_6, c_6)	(c_6, d_6)	(d_6, a_6)	- do -	Coordinates Positions **Q-II (b_6, c_6), Q-III (c_6, d_6) &** **Q-IV (d_6, a_6)** reaches
8.	(b_6, c_5)	(c_5, d_6)	(d_6, a_5)	- do -	
9.	(b_6, c_4)	(c_4, d_6)	(d_6, a_4)	- do -	
10.	(b_6, c_3)	(c_3, d_6)	(d_6, a_3)	- do -	
11.	(b_6, c_2)	(c_2, d_6)	(d_6, a_2)	- do -	
12.	(b_6, c_1)	(c_1, d_6)	(d_6, a_1)	- do -	
13.	(b_6, c_0)	(c_0, d_6)	(d_6, a_0)	- do -	Y, -Y & -Y axes reached, respectively

Table 2: Data for pixel activation for Quadrants-II, III & IV

From Tables 1 & 2 it is obvious that the combinations of two variables take place for the pixel activation for a row and a column. Further the second variable among the combination is used for the next selection as a reference. For example, in Table 2; Serial No.: 1, Quadrant-II, Quadrant-III and Quadrant-IV the pixel activation values are **(b_0, c_6)**, **(c_6, d_0)** and **(d_0, a_6),** respectively**.** In first combination second variable $\boldsymbol{c_6}$ is repeated at the first variable position at second combination, similarly the second combination second variable $\boldsymbol{d_0}$ is repeated at the first variable position at third combination in the series. Similarly, the third combination second variable $\boldsymbol{a_6}$ is repeated at the first variable position; see Table 1, Serial No.: 1, **(a_6, b_0)**.

2.4 Example 2: Construction of Figure by involving 2 Quadrants

Let's take an example to develop a triangle of size 6x6. Where 6 is an integer number starting from *Cp* in two directions as shown in Figure 4, which is assumed as the number of pixels. For the selection of point on y-axis, the integer numbers are added, such as 12 in this case.

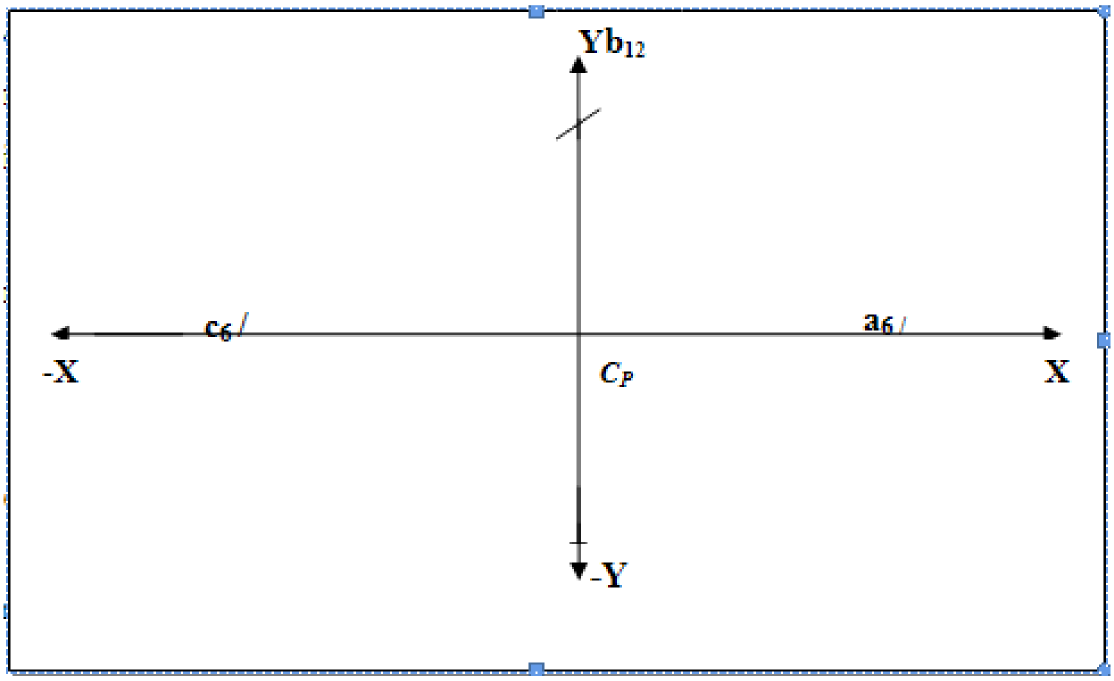
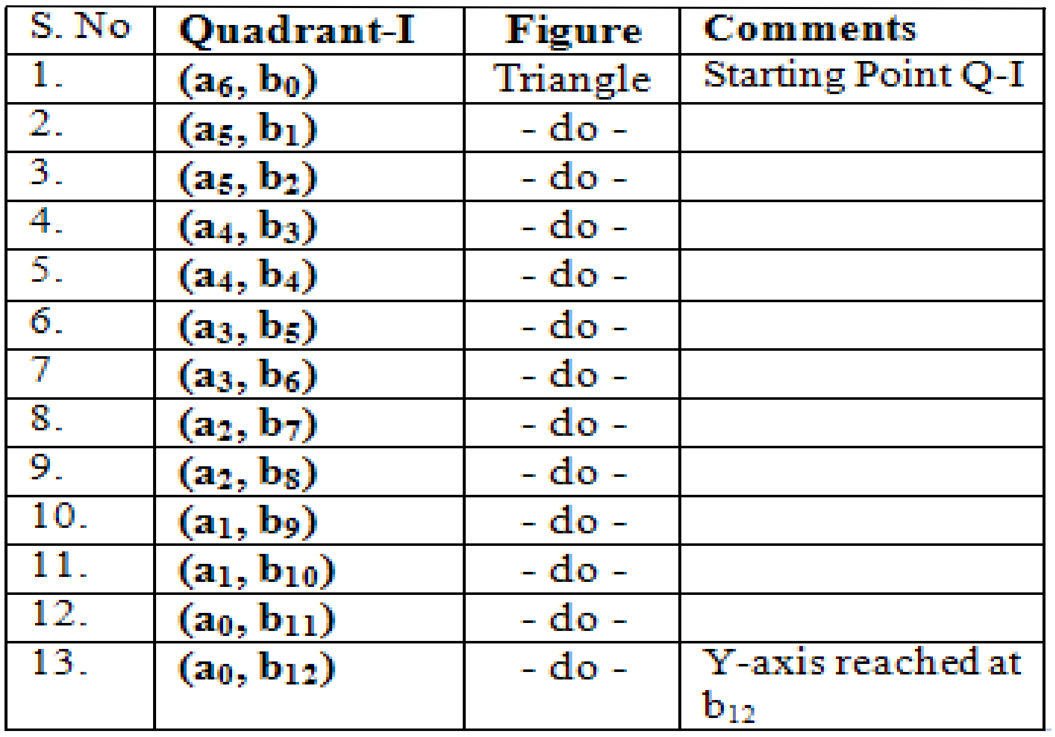


Figure 4: Pixel Activation on Quadrant-I

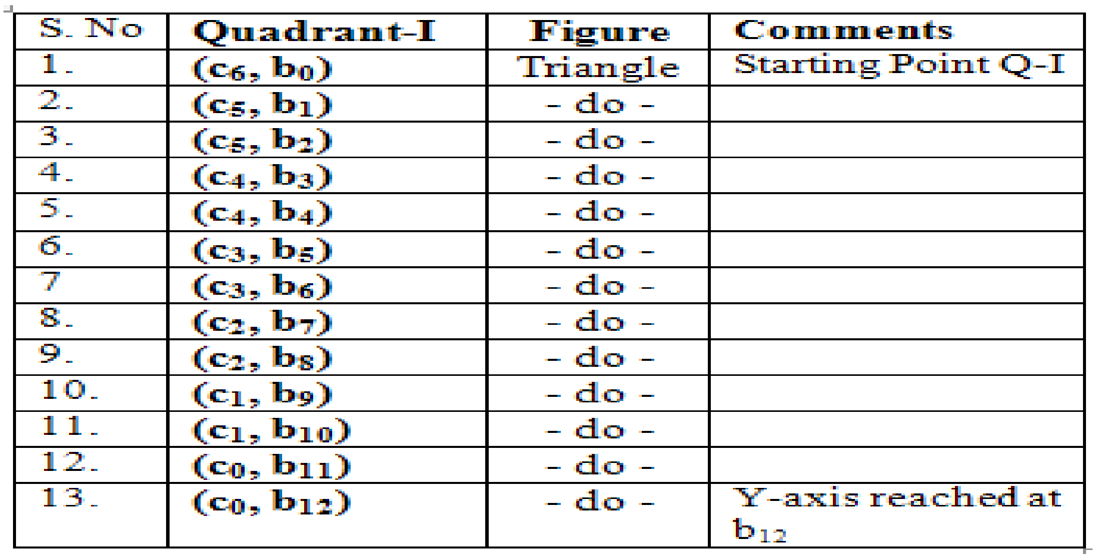
The data generated by the algorithm for the Figure 4 for the Quadrant-I is shown in Table 3.



S. No	Quadrant-I	Figure	Comments
1.	(a_6, b_0)	Triangle	Starting Point Q-I
2.	(a_5, b_1)	- do -	
3.	(a_5, b_2)	- do -	
4.	(a_4, b_3)	- do -	
5.	(a_4, b_4)	- do -	
6.	(a_3, b_5)	- do -	
7	(a_3, b_6)	- do -	
8.	(a_2, b_7)	- do -	
9.	(a_2, b_8)	- do -	
10.	(a_1, b_9)	- do -	
11.	(a_1, b_{10})	- do -	
12.	(a_0, b_{11})	- do -	
13.	(a_0, b_{12})	- do -	Y-axis reached at b_{12}

Table 3. Data for pixel activation for quadrant-1.

The operations will take place for other quadrants simultaneously to select the pixel, based on Quadrant-I values, as shown in Table 4.



S. No	Quadrant-I	Figure	Comments
1.	(c_6, b_0)	Triangle	Starting Point Q-I
2.	(c_5, b_1)	- do -	
3.	(c_5, b_2)	- do -	
4.	(c_4, b_3)	- do -	
5.	(c_4, b_4)	- do -	
6.	(c_3, b_5)	- do -	
7	(c_3, b_6)	- do -	
8.	(c_2, b_7)	- do -	
9.	(c_2, b_8)	- do -	
10.	(c_1, b_9)	- do -	
11.	(c_1, b_{10})	- do -	
12.	(c_0, b_{11})	- do -	
13.	(c_0, b_{12})	- do -	Y-axis reached at b_{12}

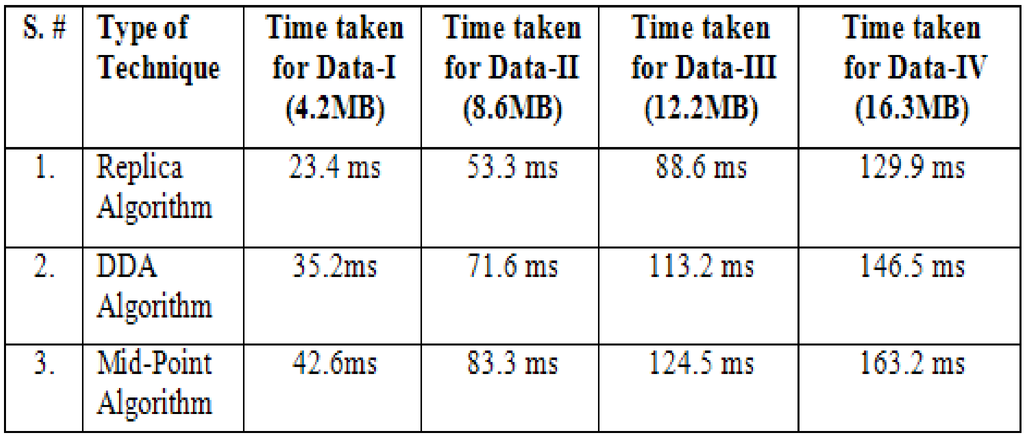
Table 4. Data for pixel activation for quadrant-2.

From Tables 3 & 4 it is notice that the combinations of two variables take place for the pixel activation for a row and a column to draw the slope towards the apex of y-axis. Further the first variable among the combination is used for the next selection as a reference. For example, in Table 3; the value of "a" is changing after two intervals where as the value of "b" is changing linearly. Similar changes also reported in the Table 4 for variable "c" and "b" activation. In this operation the value of the variable "a" is replaced by "c" as a replica, where as the values of variable "b" remains constant on both sides.

3 Simulation Results

A PC (Core 2 Duo) with a clock speed of 3GHz was used for processing different data. All three algorithms were programmed. While conducting a number of experiments on different shapes (triangle, circle, square, rectangle, etc) and different sizes of figures (Data-I to Data-IV), all three techniques were closely monitored. The results obtained during these experiments are shown in Table 5.

From Table 5, it is noticeable that in all four experiments from Data-I to Data-IV respectively, the proposed algorithm (Novel Alg.) took less time to perform the task as compared with DDA Algorithm and Mid-Point Algorithm. Figure 5 depicts the results graphically.



S. #	Type of Technique	Time taken for Data-I (4.2MB)	Time taken for Data-II (8.6MB)	Time taken for Data-III (12.2MB)	Time taken for Data-IV (16.3MB)
1.	Replica Algorithm	23.4 ms	53.3 ms	88.6 ms	129.9 ms
2.	DDA Algorithm	35.2ms	71.6 ms	113.2 ms	146.5 ms
3.	Mid-Point Algorithm	42.6ms	83.3 ms	124.5 ms	163.2 ms

Table 5: Time taken by each technique

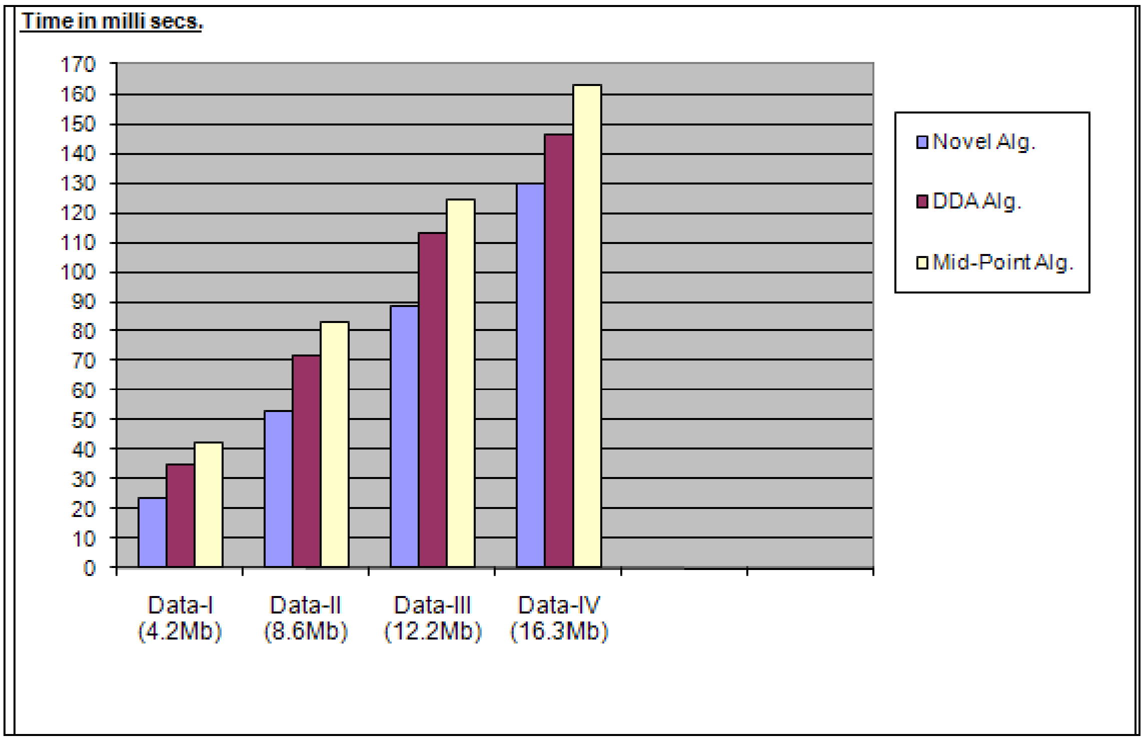


Figure 5: Comparison of Time verses Data for each approach

Figure 5 shows that the proposed algorithm required the least time. It can also be observed that the algorithms are working in linearity, that is, as the size of the figure increases the processing time increases in turn. In many instances the proposed algorithm required less than half the time taken by other algorithms to draw the figure. Secondly, the algorithm generates less data as compared to others, as it is restricted to calculations only at one position (quadrant) and the rest of the data (pixels activation) will be based on the earlier calculations.

4 Conclusion

This paper has presented the replica technique. As it is noticed from the examples given in the paper; has demonstrated the chaining/linking strategy (previous pixel value as a reference for the next pixel activation) as the main strength in the approach. Further, the difference between the results with two other approaches; it was observed that when the proposed technique was deployed for figure constructions it was not only faster but more intelligent as compared to the others. Therefore on the basis of these results, it can be reported that the *Replica algorithm* is intelligent because it calibrates and then activates its counter pixels on the screen, based on earlier calculations to obtain the one/three remaining side/sides of the figure depending on the figure to draw.

Future works

This concept can be further enhanced by using compression techniques which will assist the faster transmission of figures over the internet and through other media such as satellites. This technique also achieves unprecedented speed in calculating global illumination in synthetic scenes. At the same time, new techniques are being explored in digital photographic image/figure processing, image acquisition and image-based depiction. Furthermore, it plays a successful and important role in the areas of video games (cartoons) and in geometric design environments such as architectural design, target tracking system, etc.

Acknowledgements

I would like to acknowledge the assistance of all my colleagues who contributed in any form to make this approach successful. I would like to thank Dr. Rashid Jalal, Mr. Zahid Ullah & Mr. Maqsood Mahmud and last but not least Mr. Martin Barr & Mr. Abdul Basit for agreeing to read the manuscript of this paper willingly.

References:

[1] H. A. Fatmi, R. W. Young, A definition of Intelligence, *Nature*, Vol. 228, 1970, pp. 97.

[2] J. D. Foley, A. V. Dam, S. K., Feiner & J. F. Hughes, *Computer Graphics, Principles and Practice*, Edition 2, Addison Wesley, 1997.

[3] R. N. Aufmann, V. C. Baker, R. D. Nation, *College Algebra and Trigonometry*, Edition 5, Houghton Mifflin Company, 2002.

[4] V. Huypens, The sign corrected midpoint decision ....., *IEEE Computer Society*, 2008.

[5] *Computer Graphics*, Publishers by TATA The McGraw-Hill Companies, ace SERIES, 1979. ISBN:0-07-059376-0.

[6] A. Edward, *Interactive Computer Graphics*, Edition 5, Pearson Addison Wisely, 2009.

[7] C. Fangquan, D. Youdond, L. Jian, W. Daming, *A Novel non-stationary subdivision scheme for Geometric Modelling*, Cit-2004, pp. 748-752.

[8] R. Goldman, The ambient spaces of computer graphics and geometric modelling, *Int. J. IEEE Computer Graphics & App.*, March - April, 2000, Vol. 20, No. 2, pp. 76-84.

[9] W. Keith, R. W. Lance, Representing Interwoven Surfaces in 2-1/2D Drawings, *IEEE Computer Graphics and Applications*, July - August, 2007, Vol. 27, No. 4, pp. 70-83.

[10] A. Glassner, Crop Art. Part-1 Computer Graphics, *Int. J. IEEE Computer Graphics & App.*, September – October, 2004, Vol. 24, No. 5, pp. 86-99.

[11] H. Lilia, M. Brian, Graphics and the understanding of perceptual Mechanism: Analogies and Similarities, *GM&I*, 2006, pp. 107-112.

[12] P. V. Sender, Z. J. Wood, S. J. Gortler, et-al, Multi-Chart Geometry Images, *Proc. Symp. Geometry Processing*, 2003, pp. 146-155.

[13] C. Y. Yao, T. Y. Lee, Adaptive Geometry Image, *IEEE Transaction on Visual. And Computer Graphics*, July – August, 2008, Vol. 14, No. 4, pp. 948- 960.

[14] S. Coupland, R. John, A Fast Geometric Method for Defuzzification of Type-2 Fuzzy Sets, *IEEE Transaction on Fuzzy Systems*, August 2008, Vol. 16, No. 4, pp. 929 – 941.

[15] T. Y. Lee, S. W. Yen, I. C. Yeh, Texture Mapping with Hard Constraints using warping scheme, *IEEE Transaction on Visualization and Computer Graphics*, March – April, 2008, Vol. 14, No. 2, pp. 382 – 395.