A Mid-point Ellipse Drawing Algorithm on a Hexagonal Grid

M. Prabukumar and Bimal Kumar Ray

School of Information Technology & Engineering, VIT University
Vellore – 632 014 India
mprabukumar@vit.ac.in, bimalkumarray@vit.ac.in

Abstract

In this paper, the idea of Mid-point ellipse drawing algorithm on a hexagonal grid is proposed. The performance of the proposed algorithm is compared to that of the conventional ellipse drawing algorithm on a square grid. The qualitative and execution time analysis proves that the proposed algorithm performs better than the conventional ellipse drawing algorithm on a square grid.

Keywords: Hexagonal grids, rasterization, scan conversion, aliasing, computer graphics, frame buffer

1. Introduction

In addition to lines and circles another useful curve in graphics applications is the ellipse. A significant body of work in curve drawing algorithms on a square grid for raster display devices has already been published [1-11]. In this paper the mid-point approach for scan converting a non parametric equation of an ellipse into scan conversion algorithm on a hexagonal grid, that draws the ellipse into a bit-mapped frame buffer and that drives a raster display, is proposed. The proposed approach produces algorithms that are computationally efficient by means of reducing the number of multiplication operations required and also reduces the scan conversion error.

We summarize the related work to our proposed research as follows. Wuthrich and Stucki [12] provided a systematic proof for the similarity in the characteristics of digitized curves on square and hexagonal grids. Efficient hexagonal grid implementation of the algorithm was pointed out to give high stand for the possibility to build graphics devices on hexagonal grids. Liu Yong-Kui [13] used only integer arithmetic and developed an algorithm for the generation of straight line on hexagonal grid. Liu Yong-Kui [14] found the closest integer coordinates to the actual circular arc by using only integer arithmetic and designed two algorithms that can generate circular arc on hexagonal grids. Krzysztof T. Tytkowski [15] having discussed some advantages of hexagonal lattice over the conventional one and proposed the construction of hexagonal mesh hardware for a graphics display unit. He also presented a new approach for the representation of a pixel in raster graphics. The properties and advantages of grids based on rhombic truncated octahedral tilings were given in a study by Miller [16]. Using local counting algorithms, better perimeter estimates can be obtained using hexagonal or triangular grids. A method to trace lines in non-orthogonal grids in any dimension using only additions during the line tracing process was proposed by Luis Ibáñez et. al. [17] and achieved good performance. L. V. Pittway [18] presented an algorithm that could outline ellipses, circles, or any of the other conic sections on a hexagonal lattice. The basic algorithm requires just one test, three addition operations in the inner loop and an additional test to detect a change in the
overall direction between the two adjacent sections. It has geometric symmetries leading to better topological properties.

The paper is organized as follows, in Section 2 the hexagonal grid has been discussed. In Section 3 symmetric behavior of an ellipse is presented. In Section 4 implementation of the mid-point ellipse drawing algorithm on a hexagonal grid has been proposed. The performance analysis of the ellipse drawing algorithm on both square and hexagonal grid has been made in Section 5 and finally conclusion has been drawn in Section 6.

2. Square Grid versus Hexagonal Grid

Computer graphics, in general, represents an object in conventional square grid. In the conventional paradigm, hexagonal grid is an alternative that serves the same objective [19]. The representation of square and hexagonal grid is shown in Figure 1 and Figure 2 respectively. From the Figure 2 it is clear that hexagonal grid offers less distortion of distances than square grids, because each pixel has greater number of non-diagonal neighbors than that in a square grid [21]. A hexagonal grid has a pleasant appearance than a square grid. It is a standard result in multidimensional sampling that the hexagonal grid is optimum because it packs the most sample points into a given area [22]. Comparison of certain features between square and hexagonal grid is shown in Table 1.

![Figure 1. Square Grid](image1.png) ![Figure 2. Hexagonal Grid](image2.png)

### Table1. Square Grid versus Hexagonal Grid

<table>
<thead>
<tr>
<th>Features</th>
<th>Type of Grid</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Shape of the pixel</td>
<td>Square</td>
<td>Hexagonal</td>
</tr>
<tr>
<td>Number of neighboring pixels</td>
<td>8</td>
<td>6</td>
</tr>
<tr>
<td>Minimum distance of neighboring pixel distance from its center</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Maximum distance of neighboring pixel distance from it center</td>
<td>1.414</td>
<td>1</td>
</tr>
<tr>
<td>Number of privileged directions of the line display</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>Angle between the privileged directions(degree)</td>
<td>90</td>
<td>60</td>
</tr>
</tbody>
</table>

All the above comparisons support the fact that a hexagonal grid is a much better option over a square grid while drawing an object on a screen using computer graphics.
3. Ellipse Symmetry

The digital ellipse on a square grid is axially symmetric with respect to two lines, namely, \( x = 0 \) and \( y = 0 \) line. This is illustrated in Figure 3a. Whereas the digital ellipse on a hexagonal grid are axially symmetric with respect to three lines and those lines are \( x = y \), \( y = -2x \) and \( x = -2y \). This is illustrated in Figure 3b. Owing to the advantage of symmetry of an ellipse, we need not run the ellipse drawing algorithm to plot the entire ellipse. On a square grid only \( 1/4 \)th part of the entire ellipse needs to be plotted whereas in a hexagonal grid it is only \( 1/6 \)th. Due to this symmetry, the computation complexity of the ellipse drawing algorithm is reduced further in the case of a hexagonal grid when compared to that of the conventional square grid.

![Figure 3. Ellipse Symmetry](image)

<table>
<thead>
<tr>
<th>a. Square Grid</th>
<th>b. Hexagonal Grid</th>
</tr>
</thead>
</table>

4. A Proposed Mid-point Ellipse Drawing Algorithm

Mid-point ellipse drawing algorithm is an incremental algorithm. The spatial relationship between an arbitrary point \((u, v)\) and a circle of radius \(r\) centered at the origin is computed on a hexagonal grid using the equation (1).

\[
f(u, v) = 0.75u^2b^2 + a^2v^2 + 0.25u^2a^2 + a^2uv - a^2b^2 = 0
\]  

(1)

If \( f(u, v) < 0 \) then the point \((u, v)\) lies inside an ellipse.

If \( f(u, v) = 0 \) then the point \((u, v)\) is on an ellipse.

If \( f(u, v) > 0 \) then the point \((u, v)\) lies outside an ellipse.

Now consider the coordinates of the midpoint \((u_i + 1, v_i - \frac{1}{2})\) which is in half way between the pixel \((u_i + 1, v_i)\) and the pixel \((u_i + 1, v_i - 1)\) (figure 4).
\[ d_i = f(u_i + 1, v_i - \frac{1}{2}) = 0.75u_ib^2 + 0.25a^2u_i^2 + a^2v_i^2 + a^2v_iu_i + 0.75b^2 \] (2)

If \( d_i \) is negative, then the mid-point is inside an ellipse, and we choose the pixel \((u_i + 1, v_i)\). On the other hand, if \( d_i \) is positive (or equal to zero), the midpoint is outside an ellipse (or on an ellipse), and we choose the pixel \((u_i + 1, v_i)\). Similarly the decision parameters for the next step is computed and shown in equation (5).

\[ d_{i+1} = f(u_{i+1} + 1, v_{i+1} - 1/2 ) = 0.75b^2u_{i+1} + 0.25a^2u_{i+1}^2 + a^2v_{i+1}^2 + a^2u_{i+1}v_{i+1} + 0.75b^2 \] (3)

\[ d_{i+1} - d_i = f(u_{i+1} + 1, v_{i+1} - 1/2 ) - f(u_i + 1, v_i - 1/2 ) = 0.75b^2(u_{i+1} - u_i) + 0.25a^2(u_{i+1}^2 - u_i^2) + a^2(v_{i+1}^2 - v_i^2) + a^2(u_{i+1}v_{i+1} - u_i v_i) \] (4)

\[ d_{i+1} = d_i + 0.75b^2(u_{i+1} - u_i) + 0.25a^2(u_{i+1}^2 - u_i^2) + a^2(v_{i+1}^2 - v_i^2) + a^2(u_{i+1}v_{i+1} - u_i v_i) \] (5)

When we choose the pixel \((u_i + 1, v_i)\), that is \( d_i < 0 \), we have \( v_{i+1} = v_i \) and \( u_{i+1} = u_i + 1 \). The updated decision parameter \( d_{i+1} \) for this case is given in equation (6).

\[ d_{i+1} = d_i + 0.75b^2 + 0.25a^2 + 0.5a^2u_i + a^2v_i \] (6)

Similarly when we choose the pixel \((u_i + 1, v_i - 1)\), that is \( d_i >= 0 \) we have \( v_{i+1} = v_i - 1 \) and \( u_{i+1} = u_i + 1 \). The updated decision parameter \( d_{i+1} \) for this case is given in equation (7).

\[ d_{i+1} = d_i + 0.75b^2 + 0.25a^2 - 0.5a^2u_i - a^2v_i \] (7)

Finally, the initial decision parameter is computed using equation (2) and initial point \((0, b)\). It is given by

\[ d_i = a^2b^2 + b^2 \] (8)
Steps that are required to scan-convert an ellipse using Mid-point ellipse drawing algorithm on a hexagonal Grid as follows.

1. Set the initial values of the variables: Circle center \((u_c, v_c)\), \(u = 0\), \(v = b\) initial decision parameter \(d_i = a^2b^2 + b^2\) where \(a\) is \(u\) axis radius and \(b\) is \(v\) axis radius of an ellipse.

2. Test to determine whether the entire ellipse has been scan-converted.
   If \(u \leq 0\), stop.

3. Plot the symmetric points with respect to the center \((u_c, v_c)\) and the current \((u, v)\) coordinates.

4. Compute the location of the next pixel.
   If \(d_i < 0\), then \(d = d + 0.75b^2 + 0.25a^2 - 0.5a^2u_i - a^2v_i\); and \(u = u + 1\)
   If \(d_i \geq 0\), then \(d = d + 0.75b^2 + 0.25a^2 - 0.5a^2u_i - a^2v_i\); \(u = u + 1\); and \(v = v - 1\).

5. Goto step 2.

6. End.

5. Performance Analysis

The performance of the proposed Mid-Point ellipse drawing algorithm on a hexagonal grid is compared qualitatively with the conventional ellipse drawing algorithm on a square grid. To provide a fair comparison between the two algorithms, the hexagonal grid area is made equal to the square grid area on the screen. The comparison is performed qualitatively by plotting ellipses on the screen (as shown in Figure 5a and Figure 5b). It can be observed that in Figure 5 the aliasing effect is less in the hexagonal grid algorithms than that on the square grid algorithms.

![Figure 5. Scan-converted Ellipse](image)

It is clear from Table 2 that the number of arithmetic and logical operations acting upon the object that is being displayed on the screen decrease with change in the grid alignment from square to a hexagonal fashion.
Table 2. Comparison of Number of Operations

<table>
<thead>
<tr>
<th>Radius</th>
<th>Square Grid</th>
<th>Proposed</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Total Number of Operations</td>
<td>Total Number of Operations</td>
</tr>
<tr>
<td>X-axis</td>
<td>Y-axis</td>
<td>+</td>
</tr>
<tr>
<td>125</td>
<td>50</td>
<td>537</td>
</tr>
<tr>
<td>100</td>
<td>40</td>
<td>441</td>
</tr>
<tr>
<td>75</td>
<td>30</td>
<td>319</td>
</tr>
<tr>
<td>50</td>
<td>20</td>
<td>223</td>
</tr>
<tr>
<td>25</td>
<td>15</td>
<td>122</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>63</td>
</tr>
<tr>
<td>5</td>
<td>05</td>
<td>36</td>
</tr>
</tbody>
</table>

The execution time is given in Table 3. The values of the execution time do not reflect the original running time as no pixels are written and the function call is made inactive since the operations are computationally expensive. It is observed that the average execution time on proposed algorithm is less as compared to that in square grid algorithm. The average execution time is calculated by executing the algorithm 500 times, measuring each running time individually and then averaging. The average execution time is calculated in each of the grids in order to reduce the impacts of multicasting and multiprogramming on execution time.

Table 3. Average Execution Time Analysis

<table>
<thead>
<tr>
<th>Radius</th>
<th>Mid-point Ellipse Drawing Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Square Grid</td>
</tr>
<tr>
<td></td>
<td>Average Execution Time(Seconds)</td>
</tr>
<tr>
<td>X-axis</td>
<td>Y-axis</td>
</tr>
<tr>
<td>125</td>
<td>50</td>
</tr>
<tr>
<td>100</td>
<td>40</td>
</tr>
<tr>
<td>75</td>
<td>30</td>
</tr>
<tr>
<td>50</td>
<td>20</td>
</tr>
<tr>
<td>25</td>
<td>15</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>5</td>
<td>05</td>
</tr>
</tbody>
</table>

From the experimental analysis we conclude that the proposed algorithm is better suited than the conventional square grid because of the following reasons. The aliasing effect is less in the case of proposed algorithm. The figures in the Table 3 give mathematical evidence that the proposed algorithm is faster than the conventional square grid algorithm. This optimal nature of the results is because of the fact that the number of symmetrical points generated by proposed algorithm is more than that of the existing algorithm. Moreover the proposed algorithm makes use of very less number of calculations to generate a single pixel of the ellipse than the conventional hexagonal grid algorithms. The proposed algorithm is conceptually simple to implement in terms of hardware. For implementation the pixel clock phase of the graphics generator must be shifted by half a period on alternate display fields.
6. Conclusion

Implementation of scan-conversion of the given two-dimensional ellipses on both square and hexagonal grids brings light on the fact that the rasterization or the scan-conversion error that usually occurs during the usage of a square grid has been significantly reduced in a hexagonal grid. The noticeable improvements in case of the proposed algorithm are that the computational time is considerably less and that the error rate has been greatly reduced. The number of arithmetic and logical operations acting upon the object being displayed on the screen is decreased in proposed algorithm. All the above instances drive home the point that the generated ellipse using mid-point algorithm of proposed approach is better than that of the conventional square grid algorithm. The wider range of benefits that are obtained due to the above in the field of real time systems is inevitably a strong element.

Acknowledgements

We thank VIT University, Vellore, India for supporting this research work. We also thank anonymous reviewers for their reviews and feedback on the paper.

References


Authors

M. Prabukumar currently pursuing his Ph.D. in VIT University, Vellore. He received his Master degree in Computer Science from Amrita University, Coimbatore. His research interests are in computer graphics, computer vision and image processing. He is life time member in CSI. He has published a number of research papers in peer reviewed journals.

Bimal Kumar Ray received his Ph.D. degree in Computer Science from Indian Statistical Institute, Kolkata, India. He received his Master degree in Applied Mathematics from Calcutta University and his Bachelor degree in Mathematics from St. Xavier’s College, Kolkata. His research interests are in computer graphics, computer vision and image processing. He has published a number of research papers in peer reviewed journals.