iJRASET
International Journal For Research in
Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 10 **Issue:** XII **Month of publication:** December 2022

DOI: https://doi.org/10.22214/ijraset.2022.47884

Call: ✆08813907089 | E-mail ID: ijraset@gmail.com

Reflection of 2D Objects in Computer Graphics

Mr. N. Krishna Khanth[1], Mr. Sharad Jain[2], Ms. Kanchana Bora[3]
[1, 2, 3]MCA, Jagan Institute of Management Studies, India

Abstract: The Research paper contains a brief introduction about 2D reflection or transformation along with the better or less complex algorithm in Computer Graphics. 2D reflection is used for manipulating, repositioning, changing size, rotating and also to get the mirror image of the real-world object which are stored in the form of images in a computer. Suggesting a less complex algorithm to make it easier to understand or for manipulating images more efficiently according to the user needs.

Keywords: Graphics, 2D Reflection, Repositioning

I. INTRODUCTION

Computer Graphics is the art of drawing, graphs, photographs, maps and more on the computer screen with the help of programming. It involves image creation, manipulation to create 2D and 3D images.

Reflection is a process of transforming an object which produces the mirror image. The mirror image can be in any of the axis either x -axis or y- axis. The image can also be rotated by 180°.

A. *Types of reflection*

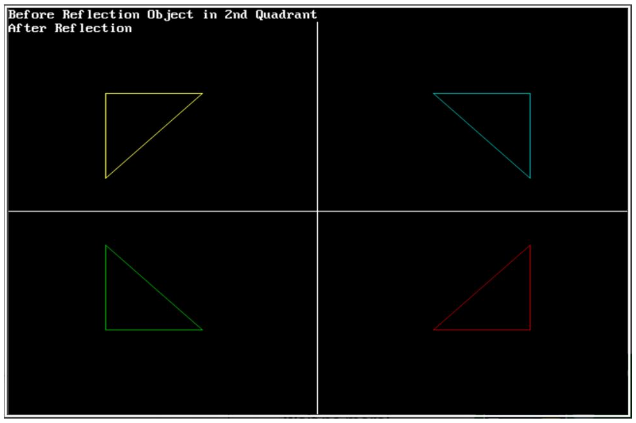
1) *Reflection about x -axis:* In this type, the Y coordinate will remain the same.
2) *Reflection about Y-axis:* X coordinate will remain the same.
3) *Reflection about Axis Perpendicular to the (x, y) Plane and Passing through The Origin:* Both X and Y coordinate will remain the same.

B. *Reflection*

Reflection is about the transformation that produces the mirror image of the object. The mirror image of the object is produced by rotating the object at 180 degrees with respect to the reflection axis. The reflection axis can be anywhere in the XY plane. [4]

II. ALGORITHM

Proposed algorithm is able to reduce space and time complexity. Algorithm performs reflection on X-axis, Y-axis, and origin as shown in image below. [3]



```
Before Reflection Object in 2nd Quadrant
After Reflection
```

A. *Existing Algorithm*

If the X-axis is the reflection axis, then after the reflection the new coordinates will be:

X’=X

Y’= -Y

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

If the Y-axis is the reflection axis, then after the reflection the new coordinates will be:
X’=-X
Y’=Y

$$\begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

If the reflection takes place simultaneously in both X-axis and Y-axis then after the reflection the new coordinates will be:
X’= -X
Y’= -Y

$$\begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

1) Implementation of Existing Algorithm in C Programming Language

```
void multiply(int mat1[][3], int mat2[][3], int res[][3]) {
        int i, j, k;
        for (i = 0; i < 3; i++)
                for (j = 0; j < 3; j++) {
                        res[i][j] = 0;
                        for (k = 0; k < 3; k++)
                                res[i][j] += mat1[i][k] * mat2[k][j];
                }
}
void refx(int mat[][3]) {
        int i, j, i1;
        int mx = getmaxx()/2, my = getmaxy()/2;
        int t[3][3] = {{1, 0, 0},
                      {0, -1, 0},
                      {0, 0, 1}};
        int res[3][3] = {{0, 0, 0},
                        {0, 0, 0},
                        {0, 0, 0}};
        multiply(mat, t, res);
        for (i = 0; i < 3; i++) {
                res[i][0] += mx;
                res[i][1] = (res[i][1] * -1) + my;
                mat[i][0] += mx;
                mat[i][1] = (mat[i][1] * -1) + my;
        }
        for (i = 0; i < 3; i++) {
                i1 = i + 1;
                if (i1 > 2)
                        i1 = 0;
                line(mat[i][0], mat[i][1], mat[i1][0], mat[i1][1]);
                line(res[i][0], res[i][1], res[i1][0], res[i1][1]);
        }
}
void refy(int mat[][3]) {
        int i, j, i1;
        int mx = getmaxx()/2, my = getmaxy()/2;
        int t[3][3] = {{-1, 0, 0},
                      {0, 1, 0},
                      {0, 0, 1}};
        int res[3][3] = {{0, 0, 0},
                        {0, 0, 0},
                        {0, 0, 0}};
        multiply(mat, t, res) ;
        for (i = 0; i < 3; i++) {
                res[i][0] += mx;
                res[i][1] = (res[i][1] * -1) + my;
                mat[i][0] += mx;
                mat[i][1] = (mat[i][1] * -1) + my;
        }
        for (i = 0; i < 3; i++) {
                i1 = i + 1;
                if (i1 > 2)
                        i1 = 0;
                line(mat[i][0], mat[i][1], mat[i1][0], mat[i1][1]);
                line(res[i][0], res[i][1], res[i1][0], res[i1][1]);
        }
}
void refo(int mat[][3]) {
        int i, j, i1;
        int mx = getmaxx()/2, my = getmaxy()/2;
        int t[3][3] = {{-1, 0, 0},
                      {0, -1, 0},
                      {0, 0, 1}};
        int res[3][3] = {{0, 0, 0},
                        {0, 0, 0},
                        {0, 0, 0}};
        multiply(mat, t, res);
        for (i = 0; i < 3; i++) {
                res[i][0] += mx;
                res[i][1] = (res[i][1] * -1) + my;
                mat[i][0] += mx;
                mat[i][1] = (mat[i][1] * -1) + my;
        }
        for (i = 0; i < 3; i++) {
                i1 = i + 1;
                if (i1 > 2)
                        i1 = 0;
                line(mat[i][0], mat[i][1], mat[i1][0], mat[i1][1]);
                line(res[i][0], res[i][1], res[i1][0], res[i1][1]);
        }
}
```

B. Proposed Algorithm

1) Approach

Step 1:
Draw a line which will act as a Y-axis and in the line function, pass 4 parameters as line(getmaxx()/2, 0 , getmaxx()/2, getmaxy).
Step 2:
Draw a line which will act as an X-axis and in the line function pass 4 parameters like line(0, getmaxy()/2, getmaxx(), getmaxy()/2).
Step 3:
Draw an object with the help of a line function with parameters from the set variables.
Step 4:
Now execute reflection and draw an object along the origin, make use of step3 and color it any color like Blue, so that it becomes distinguishable.
Step 5:
Execute reflection and draw an object along the X-axis, make use of step 1 and color it any different color so that it becomes distinguishable.
Step 6:
Execute reflection and draw an object along the Y-axis, make use of step 2 and color it any different color so that it becomes distinguishable.
[1]

2) Implementation of Algorithm in C Programming Language [1]

```
#include <conio.h>
#include <graphics.h>
#include <stdio.h>
  // Driver Code
void main()
{
  // Initialize the drivers
  int gm, gd = DETECT, ax, x1 = 100;
  int x2 = 100, x3 = 200, y1 = 100;
  int y2 = 200, y3 = 100;

  // Add in your BGI folder path
  // like below initgraph(&gd, &gm,
  // "C:\\TURBOC3\\BGI");
  initgraph(&gd, &gm, "");
  cleardevice();
    // Draw the graph
```

```
 line(getmaxx() / 2, 0, getmaxx() / 2, getmaxy());
 line(0, getmaxy() / 2, getmaxx(), getmaxy() / 2);
  // Object initially at 2nd quadrant
 printf("Before Reflection Object in 2nd Quadrant");
  // Set the color
 setcolor(14);
 line(x1, y1, x2, y2);
 line(x2, y2, x3, y3);
 line(x3, y3, x1, y1);
 getch();
  // After reflection
 printf("\nAfter Reflection");
  // Reflection along origin i.e.
 // in 4th quadrant
 setcolor(4);
 line(getmaxx() - x1, getmaxy() - y1,
 getmaxx() - x2, getmaxy() - y2);
  line(getmaxx() - x2, getmaxy() - y2,
 getmaxx() - x3, getmaxy() - y3);

 line(getmaxx() - x3, getmaxy() - y3,
 getmaxx() - x1, getmaxy() - y1);
  // Reflection along x-axis i.e.
 // in 1st quadrant
 setcolor(3);
 line(getmaxx() - x1, y1, getmaxx() - x2, y2);
 line(getmaxx() - x2, y2, getmaxx() - x3, y3);
 line(getmaxx() - x3, y3, getmaxx() - x1, y1);

 // Reflection along y-axis i.e.
 // in 3rd quadrant
 setcolor(2);
 line(x1, getmaxy() - y1, x2, getmaxy() - y2);
 line(x2, getmaxy() - y2, x3, getmaxy() - y3);
 line(x3, getmaxy() - y3, x1, getmaxy() - y1);
 getch();
  // Close the graphics
 closegraph();
}
```

III. PERFORMANCE ANALYSIS

Performance analysis is performed based on space and time complexity.
Comparison is done based on Apostrium Analysis.

A. Time Complexity

Following calculations are for single reflection.

1) Traditional Algorithm Analysis

Assignment operations = 2 + 2 + (3 * 3) = 13
Summary – 2 variables for loop iteration, 2 (3 * 3) matrices, and 9 assignments for matrix multiplication.

Arithmetic operations = 3 * 3 * 3 * 2 = 54
Summary – 54 arithmetic operations including multiplication and addition.
Asymptotic Time Complexity = O(1)
Apostrium Time Complexity = 67

2) Proposed Algorithm Analysis

Assignment operations = 7
Summary – 7 variables for triangle corners coordinates.
Arithmetic operations = 3 * 4 = 12
Summary – 3 corners of triangle and 4 arithmetic operations for each corner.
Asymptotic Time Complexity = O(1)
Apostrium Time Complexity = 19

3) Comparison

Proposed algorithm performs significantly better in terms of time complexity.
In large scale, proposed algorithm will save considerable amount of time. [2]

B. Space complexity

Following calculations are for single reflection.

1) Traditional Algorithm

Variables Space = (3 * 3) * 3 = 27
Summary – 3 matrices of (3 * 3) size are required.
Stack Space = O(1)
Asymptotic Space Complexity = O(1)
Total Space = 27

2) Proposed Algorithm

Variables Space = 7
Summary – 7 variables to store coordinates of corners of triangle.
Stack Space = O(1)
Asymptotic Space Complexity = O(1)
Total Space = 7

3) Comparison

Proposed algorithm performs significantly better in terms of space complexity.
In large scale, proposed algorithm will save considerable number of resources. [2]

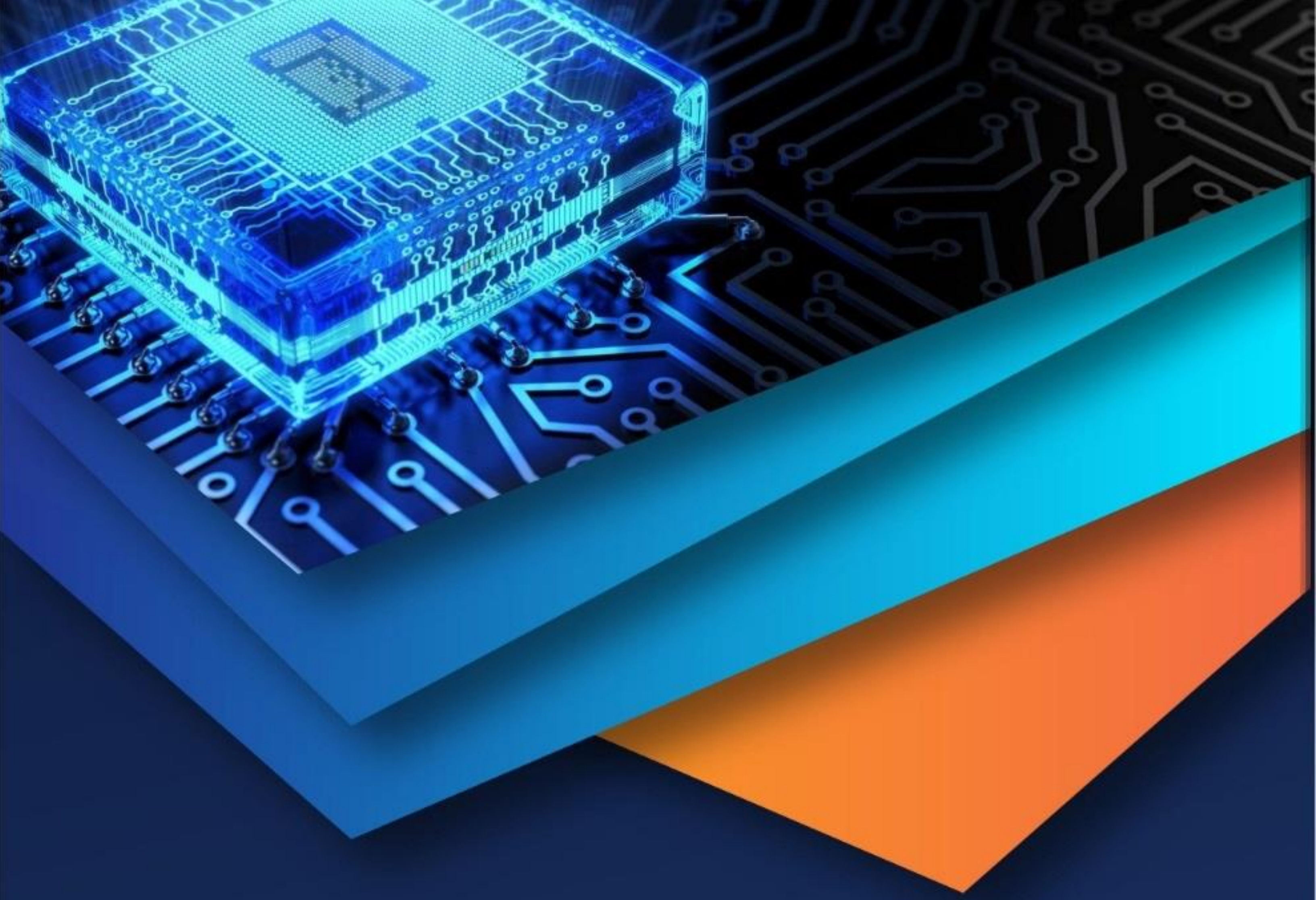
IV. CONCLUSION

Here in this paper, we have discussed reflection transformation. Among the various transformation we have discussed reflection. We proposed a new C program for 2d reflection in Computer reflection. It is much easier and less complex than the existing one. Better understanding in academics. More helpful in solving large scale problems.

REFERENCES

[1] https://www.geeksforgeeks.org/c-program-to-perform-reflection-of-the-given-2d-image-using-computer-graphics/
[2] Etsuji Tomita, Akira Tanaka, Haruhisa Takahashi, The worst-case time complexity for generating all maximal cliques and computational experiments, Theoretical Computer Science, Volume 363, Issue 1, 2006, Pages 28-42, ISSN 0304-3975.
[3] Pietro S. Oliveto, Carsten Witt, Improved time complexity analysis of the Simple Genetic Algorithm, Theoretical Computer Science, Volume 605, 2015, Pages 21-41, ISSN 0304-3975
[4] Ching L. Teo, Cornelia Fermuller, Yiannis Aloimonos; Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2015, pp. 1644-1652.
[5] Foley, Van Dam, Feiner, Hughes, Computer Graphics Principles & Practice, 2000, Pearson

[6] Chennakesava R. Alavla "Computer Graphics", PHI Learning Pvt. Limited
[7] D. Hearn & Baker: Computer Graphics with OpenGL, Pearson Education, Third Edition, 2009.
[8] Foley, J.D. & Van Dam, A: Fundamentals of Interactive Computer Graphics.
[9] Rogers & Adams, "Mathematical Elements for Computer Graphics", McGraw Hill, 1989
[10] John F. Hughes, Andries Van Dam, Morgan Mcguire, David F. Sklar James D. Foley, Steven Feiner, Kurt Akeley, "Computer Graphics Principles and Practice Third Edition".



10.22214/IJRASET

45.98

IMPACT FACTOR:
7.129

IMPACT FACTOR:
7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)