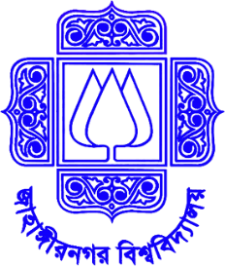
**Lab Report. 02**

*Course title: Computer Graphics Lab*

*Course code: CSE-304*

*3rd Year 1st Semester 2022*

**Date of Submission**: 04/06/2023



###### **Submitted to-**

###### Dr. Mohammad Shorif Uddin

###### Professor

*Department of Computer Science and Engineering*

*Jahangirnagar University*

*Savar, Dhaka-1342*

*And*

*Dr. Morium Akter*

###### Associate Professor

*Department of Computer Science and Engineering*

*Jahangirnagar University*

*Savar, Dhaka-1342*

| **Sl** | Class Roll | Exam Roll | Name |
| --- | --- | --- | --- |
| 01 | 360 |  | Snigdha Rahman |

**(i) Scan conversion of a circle using midpoint algorithm:**

Introduction:

The midpoint algorithm, also known as the midpoint circle algorithm, is a popular method for efficiently scan converting a circle. It allows for the smooth and accurate representation of circles on digital displays, making it an essential tool in various applications such as computer-aided design (CAD), image processing, and game development.

| #include <iostream>  #include <graphics.h>  void drawCircle(int radius, int xc, int yc) {  int x = 0;  int y = radius;  int d = 1 - radius;  int gd = DETECT, gm;  initgraph(&gd, &gm, "");  while (x <= y) {  putpixel(xc + x, yc + y, WHITE);  putpixel(xc - x, yc + y, WHITE);  putpixel(xc + x, yc - y, WHITE);  putpixel(xc - x, yc - y, WHITE);  putpixel(xc + y, yc + x, WHITE);  putpixel(xc - y, yc + x, WHITE);  putpixel(xc + y, yc - x, WHITE);  putpixel(xc - y, yc - x, WHITE);  if (d < 0) {  d += 2 \* x + 3;  } | else {  d += 2 \* (x - y) + 5;  y--;  }  x++;  }  getch();  closegraph();  }  int main() {  int radius;  std::cout << "Enter the radius of the circle: ";  std::cin >> radius;  int xc = 250;  int yc = 250;  drawCircle(radius, xc, yc);  return 0;  } |
| --- | --- |

Output:



Discussion:

The code demonstrates the implementation of the midpoint algorithm for scan converting a circle. It uses the putpixel function to mark pixels on or near the circumference of the circle with the color white. The result is a graphical output window showing the scan-converted circle with the specified radius and center coordinates.

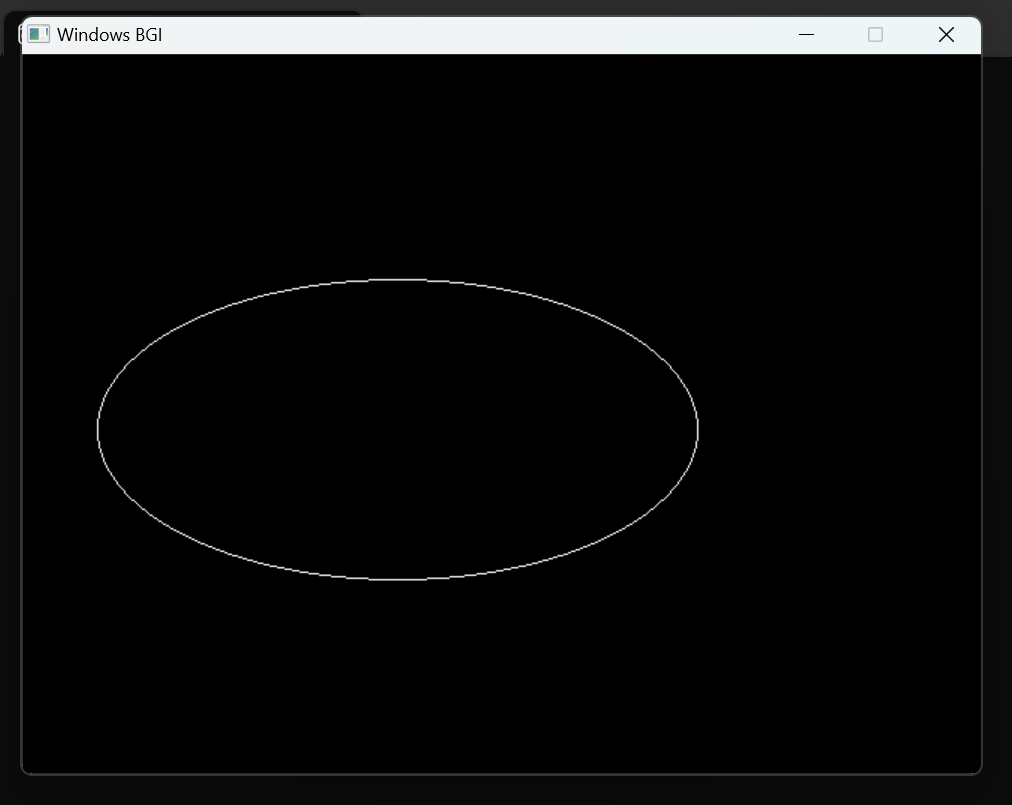
**(ii) Scan conversion of a ellipse:**

Introduction:

Scan converting an ellipse involves determining the set of pixels that lie on or near the boundary of the ellipse. The ellipse can be represented by its center coordinates, major and minor axes lengths, and rotation angle (if applicable). Various algorithms can be used to scan convert an ellipse, such as the Midpoint algorithm or the Polar algorithm.

| #include <iostream>  #include <graphics.h>  #include<math.h>  void drawEllipse(int a, int b, int xc, int yc) {  int x = 0;  int y = b;  int a\_squared = a \* a;  int b\_squared = b \* b;  int two\_a\_squared = 2 \* a\_squared;  int two\_b\_squared = 2 \* b\_squared;  int four\_a\_squared = 4 \* a\_squared;  int four\_b\_squared = 4 \* b\_squared;  int x\_end = b\_squared / sqrt(a\_squared + b\_squared);  int dx = 0;  int dy = two\_a\_squared \* y;  int gd = DETECT, gm;  initgraph(&gd, &gm, "");  putpixel(xc + x, yc - y, WHITE);  putpixel(xc + x, yc + y, WHITE);  int p1 = round(b\_squared - (a\_squared \* b) + (0.25 \* a\_squared));  while (dx < dy) {  x++;  if (p1 < 0) {  dx += two\_b\_squared;  p1 += dx + b\_squared;  } else {  y--;  dx += two\_b\_squared;  dy -= two\_a\_squared;  p1 += dx - dy + b\_squared;  }  putpixel(xc + x, yc - y, WHITE); | putpixel(xc + x, yc + y, WHITE);  putpixel(xc - x, yc + y, WHITE);  putpixel(xc - x, yc - y, WHITE);  }  int p2 = round(b\_squared \* (x + 0.5) \* (x + 0.5) + a\_squared \* (y - 1) \* (y - 1) - a\_squared \* b\_squared);  while (y > 0) {  y--;  if (p2 > 0) {  dy -= two\_a\_squared;  p2 += a\_squared - dy;  } else {  x++;  dx += two\_b\_squared;  dy -= two\_a\_squared;  p2 += dx - dy + a\_squared;  }  putpixel(xc + x, yc - y, WHITE);  putpixel(xc + x, yc + y, WHITE);  putpixel(xc - x, yc + y, WHITE);  putpixel(xc - x, yc - y, WHITE);  }  delay(5000);  closegraph();  }  int main() {  int a = 200;  int b = 100;  int xc = 250;  int yc = 250;  drawEllipse(a, b, xc, yc);  return 0;  } |
| --- | --- |

Output:



Discussion:

The code demonstrates the implementation of the midpoint algorithm for scan converting an ellipse. It uses the putpixel function to mark pixels on or near the boundary of the ellipse with the color white. The result is a graphical output window showing the scan-converted ellipse with the specified major and minor axes lengths and the center coordinates.