#include "mainwindow.h"

#include <QPainter>

#include <QPoint>

MainWindow::MainWindow(QWidget \*parent)

: QMainWindow(parent)

{

setWindowTitle("Circle Drawing Algorithm");

resize(400, 400);

}

MainWindow::~MainWindow() {}

void MainWindow::paintEvent(QPaintEvent \*event)

{

QPainter painter(this);

painter.setRenderHint(QPainter::Antialiasing); // Enable antialiasing to smooth the edges

// Circle center and radius

int centerX = width() / 2;

int centerY = height() / 2;

int radius = 100;

// Draw the circle manually using Bresenham's algorithm

drawCircle(painter, centerX, centerY, radius);

// Draw two lines inside the circle

painter.drawLine(centerX - radius, centerY, centerX + radius, centerY);

painter.drawLine(centerX, centerY - radius, centerX, centerY + radius);

}

void MainWindow::drawCircle(QPainter &painter, int x0, int y0, int r)

{

int x = 0;

int y = r;

int d = 3 - 2 \* r; // Initial decision parameter

auto plotCirclePoints = [&](int x, int y) {

painter.drawPoint(x0 + x, y0 + y); // Octant 1

painter.drawPoint(x0 - x, y0 + y); // Octant 2

painter.drawPoint(x0 + x, y0 - y); // Octant 3

painter.drawPoint(x0 - x, y0 - y); // Octant 4

painter.drawPoint(x0 + y, y0 + x); // Octant 5

painter.drawPoint(x0 - y, y0 + x); // Octant 6

painter.drawPoint(x0 + y, y0 - x); // Octant 7

painter.drawPoint(x0 - y, y0 - x); // Octant 8

};

while (x <= y)

{

plotCirclePoints(x, y);

if (d < 0)

{

d = d + 4 \* x + 6;

}

else

{

d = d + 4 \* (x - y) + 10;

y--;

}

x++;

}

}

