//program

//circle.cpp

#include "circle.h"

#include "mainwindow.h"

#include "ui\_mainwindow.h"

#include "pixel.h"

#include "QPainter"

circle::circle()

{

}

void circle::draw\_circle(QPainter \*painter, pixel c, int r)

{

int x=0,y=r,h=1-r,deltaE=3,deltaSE=(-2\*r)+5;

d\_c(painter,x,y,c);

while(y>x)

{

if(h<0)

{

h=h+deltaE;

deltaE=deltaE+2;

deltaSE=deltaSE+2;

}

else

{

h=h+deltaSE;

deltaE=deltaE+2;

deltaSE=deltaSE+4;

y=y-1;

}

x=x+1;

d\_c(painter,x,y,c);

}

}

void circle::d\_c(QPainter \*painter,int x, int y,pixel c)

{

painter->drawPoint(c.getx()+x+painter->window().width()/2,painter->window().height()/2-y-c.gety());

painter->drawPoint(c.getx()+x+painter->window().width()/2,painter->window().height()/2+y-c.gety());

painter->drawPoint(c.getx()-x+painter->window().width()/2,painter->window().height()/2-y-c.gety());

painter->drawPoint(c.getx()-x+painter->window().width()/2,painter->window().height()/2+y-c.gety());

painter->drawPoint(c.getx()+y+painter->window().width()/2,painter->window().height()/2-x-c.gety());

painter->drawPoint(c.getx()+y+painter->window().width()/2,painter->window().height()/2+x-c.gety());

painter->drawPoint(c.getx()-y+painter->window().width()/2,painter->window().height()/2-x-c.gety());

painter->drawPoint(c.getx()-y+painter->window().width()/2,painter->window().height()/2+x-c.gety());

}

//line.cpp

#include "line.h"

#include "mainwindow.h"

#include "pixel.h"

#include "math.h"

#include "QPainter"

#include "ui\_mainwindow.h"

void line::dda\_line(QPainter \*painter,pixel p1,pixel p2,float a)

{

QPen p(Qt::red,3,Qt::SolidLine);

painter->setPen(p);

int x1,y1,x2,y2,dx,i,dy,steps;float xinc,yinc;int cnt,cnt1;

float x,y;

x1=p1.getx();

x2=p2.getx();

y1=p1.gety();

y2=p2.gety();

dx=x2-x1;

dy=y2-y1;

if(abs(dx)>abs(dy))

{

steps=abs(dx);

}

else

{

steps=abs(dy);

}

xinc=dx/(float)steps;

yinc=dy/(float)steps;

x=x1;y=y1;

painter->drawPoint(x+painter->window().width()/2,painter->window().height()/2-y);

if(a==1)

{

for(i=0;i<steps;i++)

{

QPen p(Qt::blue,3,Qt::SolidLine);

painter->setPen(p);

x=x+xinc;

y=y+yinc;

if(i%4==0){

painter->drawPoint(x+painter->window().width()/2,painter->window().height()/2-y);}

}

}

else if(a==2)

{ cnt=0;cnt1=0;

for(i=0;i<steps;i++)

{ cnt++;

QPen p(Qt::black,2,Qt::SolidLine);

painter->setPen(p);

x=x+xinc;

y=y+yinc;

if(cnt<=7){

cnt1=0;

painter->drawPoint(x+painter->window().width()/2,painter->window().height()/2-y);

}

else

{

cnt1++;

}

if(cnt1==3)

{

cnt=0;

}

}

}

else if(a==3)

{

cnt=0;cnt1=0;

for(i=0;i<steps;i++)

{

cnt++;

QPen p(Qt::black,2,Qt::SolidLine);

painter->setPen(p);

x=x+xinc;

y=y+yinc;

if(cnt<=8)

{

cnt1=0;

painter->drawPoint(x+painter->window().width()/2,painter->window().height()/(2\*y));

}

else

{

cnt1++;

}

if(cnt1==3)

{

painter->drawPoint(x+painter->window().width()/2,painter->window().height()/2-y);

}

if(cnt1==5)

{

cnt=0;

}

}

}

else if(a==4)

{

cnt=0;cnt1=0;

for(i=0;i<steps;i++)

{

cnt++;

QPen p(Qt::black,2,Qt::SolidLine);

painter->setPen(p);

x=x+xinc;

y=y+yinc;

if(cnt<=12)

{

cnt1=0;

painter->drawPoint(x+painter->window().width()/2,painter->window().height()/2-y);

}

else

{

cnt1++;

}

if(cnt1==3 || cnt1==6)

{

painter->drawPoint(x+painter->window().width()/2,painter->window().height()/2-y);

}

if(cnt1==7)

{

cnt=0;

}

}

}

else

{

for(i=0;i<steps;i++)

{

QPen p(Qt::red,3,Qt::SolidLine);

painter->setPen(p);

x=x+xinc;

y=y+yinc;

painter->drawPoint(x+painter->window().width()/2,painter->window().height()/2-y);

}

}

}

void line::dda\_line(QPainter \*painter,pixel p1,pixel p2,int w)

{QPen p(Qt::red,1,Qt::SolidLine);

painter->setPen(p);

int x1,y1,x2,y2,dx,i,dy,s2,s1,intchange,temp,edx,edy,d;

float x,y;

intchange=0;

x1=p1.getx();

x2=p2.getx();

y1=p1.gety();

y2=p2.gety();

dx=abs(x2-x1);

dy=abs(y2-y1);

if(x1==x2)

{

s1=0;

}

else if(x1<x2)

{

s1=1;

}

else

{

s1=-1;

}

if(y1==y2)

{

s2=0;

}

else if(y1<y2)

{

s2=1;

}

else

{

s2=-1;

}

if(dy>dx)

{

intchange=1;

temp=dx;

dx=dy;

dy=temp;

w=(w-1)\*sqrt(pow(dx,2)+pow(dy,2))/(2\*dy);

}

else

{

w=(w-1)\*sqrt(pow(dx,2)+pow(dy,2))/(2\*dx);

}

edx=2\*dx;

edy=2\*dy;

d=edy-dx;

x=x1;

y=y1;

for(i=1;i<=dx;i++)

{

// painter->drawPoint(x+painter->window().width()/2,painter->window().height()/2-y);

dda\_line(painter,x,y,w,intchange);

while(d>0)

{

if(intchange==1)

{

x=x+s1;

}

else

{

y=y+s2;

}

d=d-edx;

}

if(intchange==1)

{

y=y+s2;

}

else

{

x=x+s1;

}

d=d+edy;

}

}

void line::dda\_line(QPainter \*painter, int x, int y, int w, int intchange)

{

int i=0;

if(intchange==0)

{

for(i=0;i<w;i++)

{

painter->drawPoint(x+painter->window().width()/2,painter->window().height()/2-y+i);

painter->drawPoint(x+painter->window().width()/2,painter->window().height()/2-y-i);

}

}

else

{

for(i=0;i<w;i++)

{

painter->drawPoint(x+i+painter->window().width()/2,painter->window().height()/2-y);

painter->drawPoint(x-i+painter->window().width()/2,painter->window().height()/2-y);

}

}

}

//main.cpp

#include <QtGui/QApplication>

#include "mainwindow.h"

int main(int argc, char \*argv[])

{

QApplication a(argc, argv);

MainWindow w;

w.show();

return a.exec();

}

//mainwindow.cpp

#include "mainwindow.h"

#include "ui\_mainwindow.h"

#include "line.h"

#include "pixel.h"

#include "QPainter"

#include "circle.h"

MainWindow::MainWindow(QWidget \*parent) :

QMainWindow(parent),

ui(new Ui::MainWindow)

{

ui->setupUi(this);

}

MainWindow::~*MainWindow*()

{

delete ui;

}

void MainWindow::*paintEvent*(QPaintEvent \*)

{

QPainter painter(this);

int x1,y1,x2,y2,x,y,w,r;

x1=ui->t1->toPlainText().toInt();

y1=ui->t2->toPlainText().toInt();

x2=ui->t3->toPlainText().toInt();

y2=ui->t4->toPlainText().toInt();

w=ui->t5->toPlainText().toInt();

x=ui->t6->toPlainText().toInt();

y=ui->t7->toPlainText().toInt();

r=ui->t8->toPlainText().toInt();

pixel p1,p2,p3 ,p4,p5,p6,p7;

p1.input(x1,y1);

p2.input(x2,y2);

p3.input((width()/2),0);

p4.input((width()/2),height());

p5.input(0,height()/2);

p6.input(width(),(height()/2));

p7.input(x,y);

circle obj2;

line obj1;

// obj1.dda\_line(&painter,p4,p3,0);

//obj1.dda\_line(&painter,p6,p5,0);

QPen p(Qt::red,1,Qt::SolidLine);

painter.setPen(p);

painter.drawLine(p3.getx(),p3.gety(),p4.getx(),p4.gety());

painter.drawLine(p5.getx(),p5.gety(),p6.getx(),p6.gety());

if(w==0){

obj1.dda\_line(&painter,p1,p2,a);}

else{

obj1.dda\_line(&painter,p1,p2,w);}

obj2.draw\_circle(&painter,p7,r);

}

void MainWindow::on\_actionDda\_line\_thin\_triggered()

{

a=0;

update();

}

void MainWindow::on\_actionThick\_line\_triggered()

{

update();

}

void MainWindow::on\_actionDashed\_line\_triggered()

{

a=2;

update();

}

void MainWindow::on\_actionDotted\_line\_triggered()

{

a=1;

update();

}

void MainWindow::on\_actionDash\_dot\_line\_triggered()

{

a=3;

update();

}

void MainWindow::on\_actionDash\_double\_dot\_triggered()

{

a=4;

update();

}

void MainWindow::on\_actionCircle\_triggered()

{

update();

}

//header files

//circle.h

#include "QPainter"

#include "mainwindow.h"

#include "pixel.h"

#ifndef CIRCLE\_H

#define CIRCLE\_H

class circle

{

public:

int radius;

pixel center;

void draw\_circle(QPainter \*painter,pixel c,int r );

void d\_c(QPainter \*painter,int x,int y,pixel c);

circle();

};

#endif // CIRCLE\_H

//line.h

#include "pixel.h"

#include "QPainter"

#include "mainwindow.h"

#ifndef LINE\_H

#define LINE\_H

class line

{

public:

pixel p1,p2;

void dda\_line(QPainter \*,pixel,pixel,float);

void dda\_line(QPainter \*,pixel,pixel,int);

void dda\_line(QPainter \*,int,int,int,int);

};

#endif // LINE\_H

//mainwindow.h

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include"ui\_mainwindow.h"

#include <QMainWindow>

namespace Ui {

class MainWindow;

}

class MainWindow : public QMainWindow

{

Q\_OBJECT

public:float a;

explicit MainWindow(QWidget \*parent = 0);

~*MainWindow*();

void *paintEvent*(QPaintEvent \*);

private slots:

void on\_actionDda\_line\_thin\_triggered();

void on\_actionThick\_line\_triggered();

void on\_actionDashed\_line\_triggered();

void on\_actionDotted\_line\_triggered();

void on\_actionDash\_dot\_line\_triggered();

void on\_actionDash\_double\_dot\_triggered();

void on\_actionCircle\_triggered();

private:

Ui::MainWindow \*ui;

};

#endif // MAINWINDOW\_H

//pixel.h

#ifndef PIXEL\_H

#define PIXEL\_H

class pixel

{

int x,y;

public:void input(int a,int b)

{

x=a;

y=b;

}

int getx()

{

return x;

}

int gety()

{

return y;

}

};

#endif // PIXEL\_H

