计算机图形学课程

小作业

实现画圆算法

何振邦

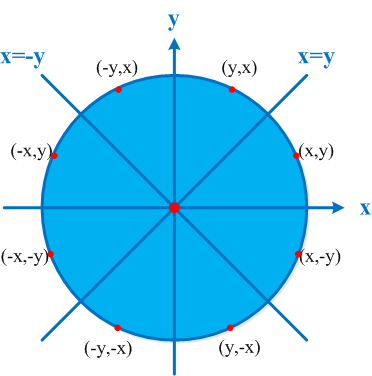
电计1503班

201585043

**算法简介**

1、中点画圆法

圆被定义为到给定中心位置（xc,yc）距离为r的点集。圆心位于原点的圆有四条对称轴x=0,y=0, x=y和x=-y。若已知圆弧上一点（x,y）,可以得到其关于四条对称轴的其它7个点，这种性质称为八分对称性。因此，只要扫描转换八分之一圆弧，就可以求出整个圆弧的象素集。



我们构造函数 F(x,y)=x^2+y^2-R^2，则对于圆上的点有F(x,y)=0，对于圆外的点有F(x,y)>0，对于圆内的点F(x,y)<0 。与中点画线法一样，构造判别式：

d=F(M)=F(xp+1,yp-0.5)=(xp+1) ^2+(yp-0.5) ^2-R^2

 若 d<0，则应取P1为下一象素，而且再下一象素的判别式为：

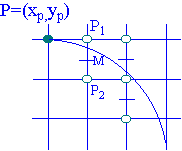
d=F(xp+2,yp-0.5)=(xp+2) ^2+(yp-0.5) ^2-R^2=d+2xp+3

 若d≥0，则应取P2为下一象素，而且下一象素的判别式为

d=F(xp+2,yp-1.5)=(xp+2) ^2+(yp-1.5) ^2-R^2=d+2(xp-yp)+5

我们这里讨论的第一个象素是（0,R），判别式d的初始值为：

d0=F(1,R-0.5)=1.25-R



2、中点画圆优化法

用 ∆d修正 d

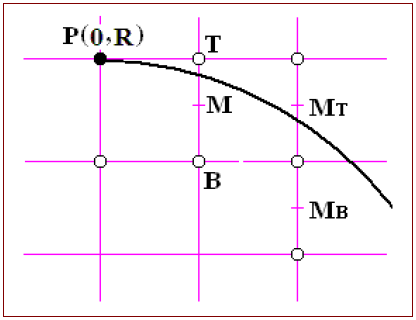
1) 选择 T点 ( xp← xp+1):

d的增量 ( 一次差分 ): ∆dt= 2xp+3

∆d的增量 ( 二次差分 ):

∆dt=2(xp+1)+3-(2xp+3)=2

∆db=2(xp+1)-2yp+5-(2xp-2yp+5)= 2



2) 选择 B点 ( xp← xp+1， yp← yp-1):

d的增量 ( 一次差分 ):

∆db= 2xp-2xp +5

∆d的增量 ( 二次差分 ):

∆dt=2(xp+1)+3-(2xp+3)=2

∆db=2(xp+1)-2(xp-1)+5-(2xp-2yp+5)=4

3、极坐标画法

利用圆的极坐标公式：

x=int(r\*sin(i))

y=int(r\*cos(i))

来画圆，令i的跳跃间隔足够密集，并让i在0-360之间递增，即可画出整个圆形。

**程序演示**

本程序使用Qt框架编写，程序在界面上先绘制棋盘网格来模拟计算机屏幕的像素，右上的滑动条可以用来设置画板棋盘格的大小，右边的圆半径滑动条可以用于调节要绘制的圆的半径，点击添加圆按钮便可在画板上添加一个绘制出来的圆形；“真实值对比”选项开启之后会为每个模拟绘制的圆上再绘制一个该模拟圆所逼近的真实数学意义上的圆。右下的单选框可以选择当前画圆的算法，依次有中点算法、极坐标算法和中点优化算法。算法性能测试则会使用当前选定的算法进行4000个圆的绘制，并显示绘制时间。

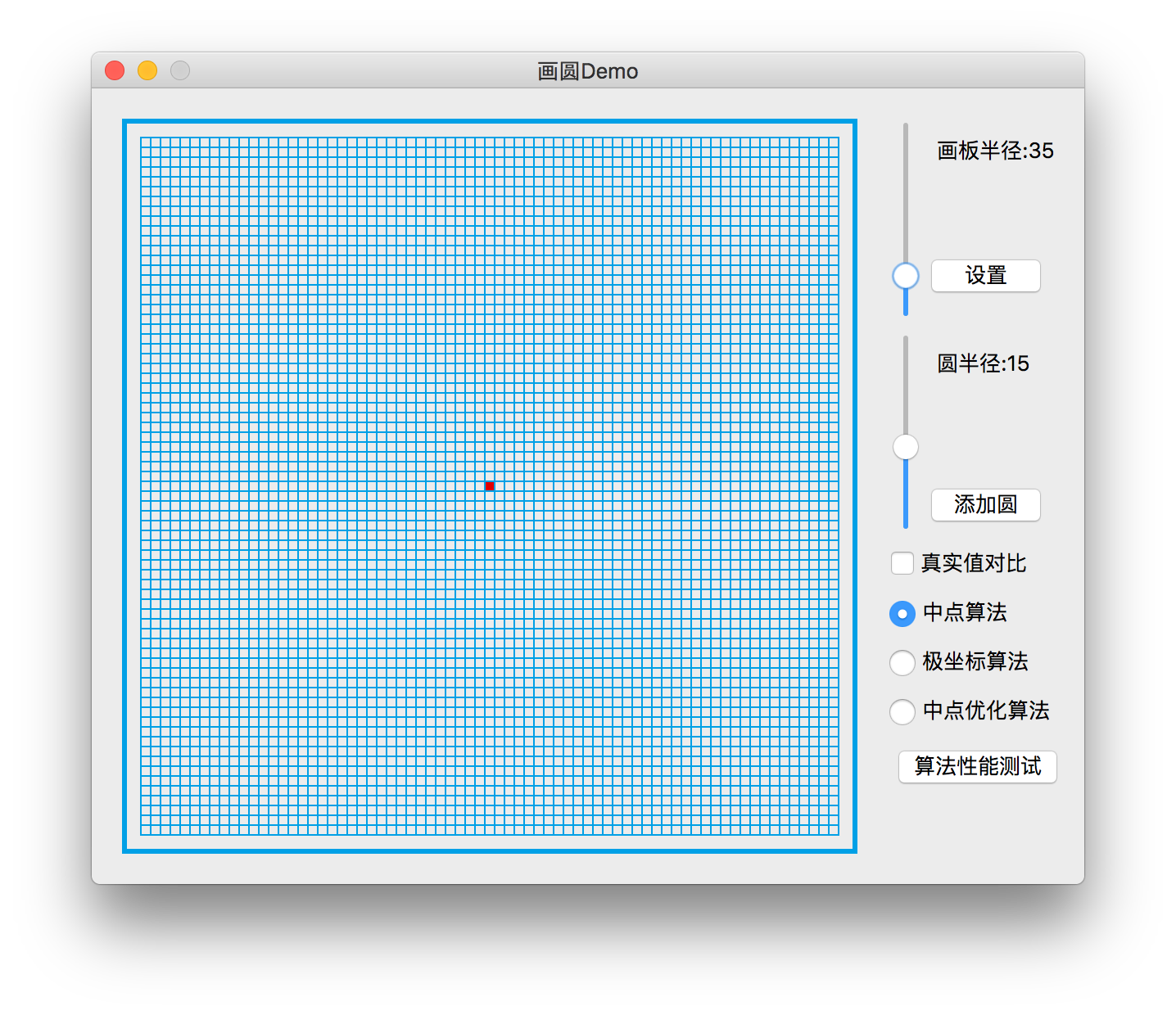


图1 程序的截图

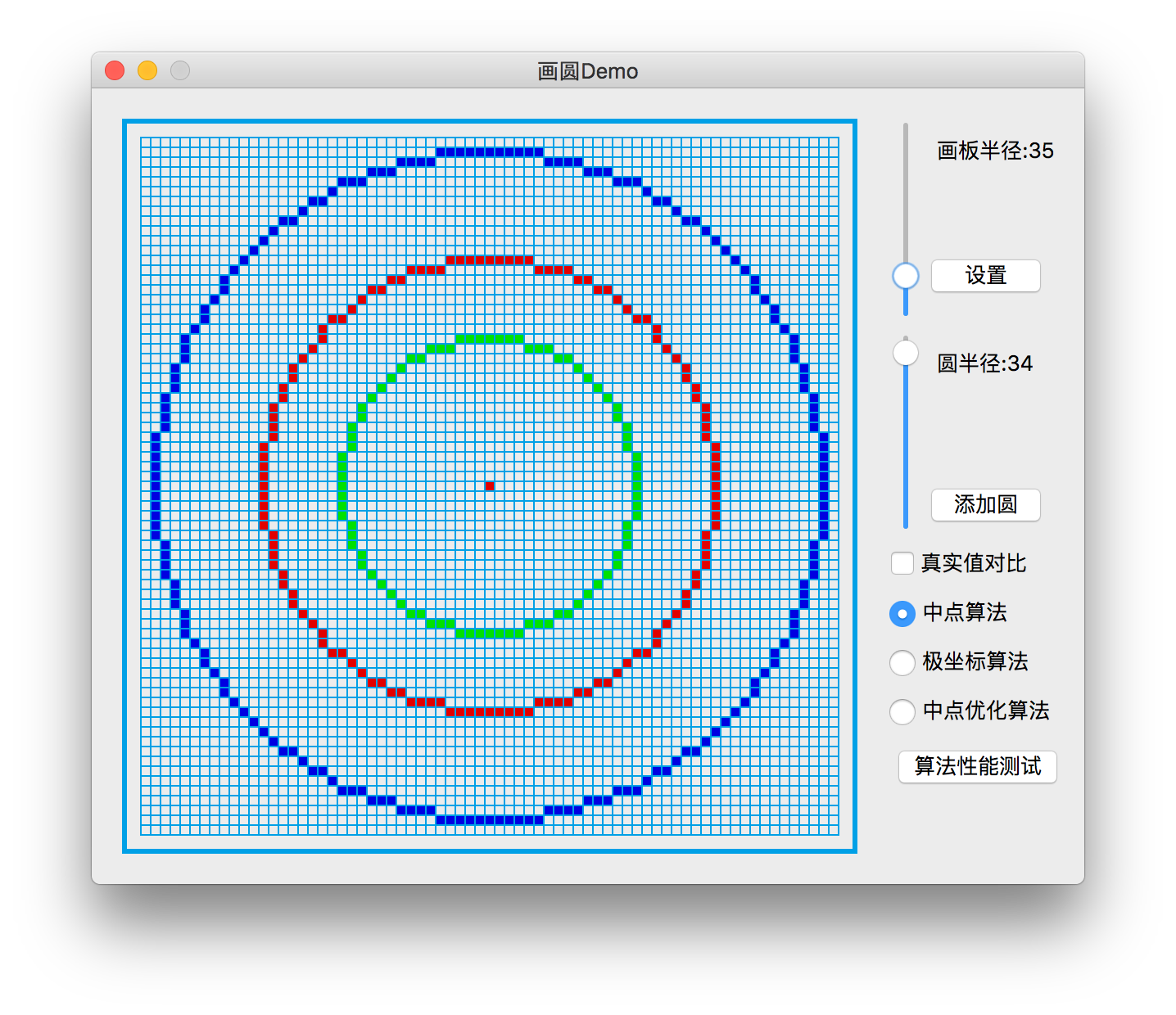


图2 程序使用中点算法绘制圆的效果

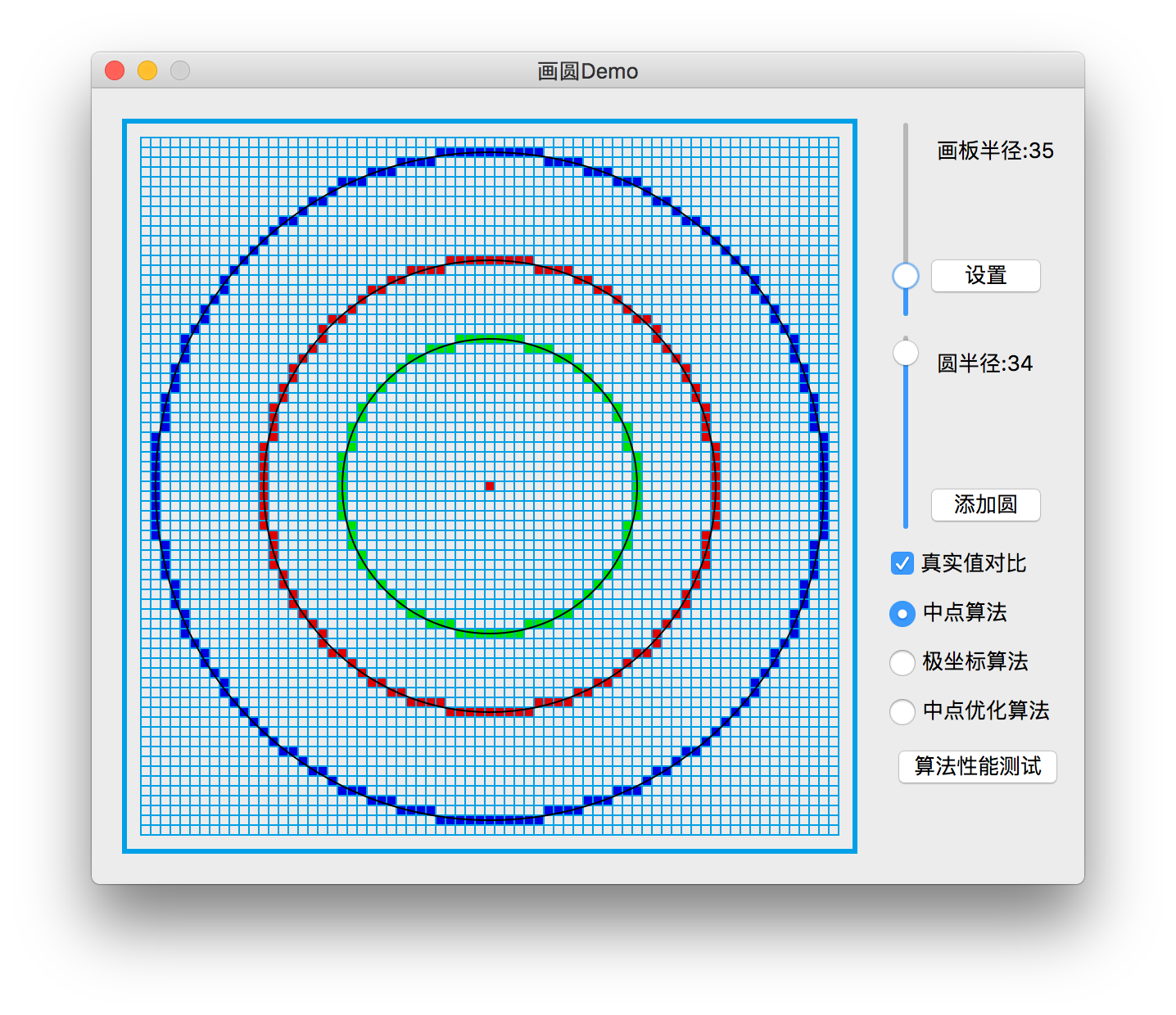


图3 显示真实值对比效果

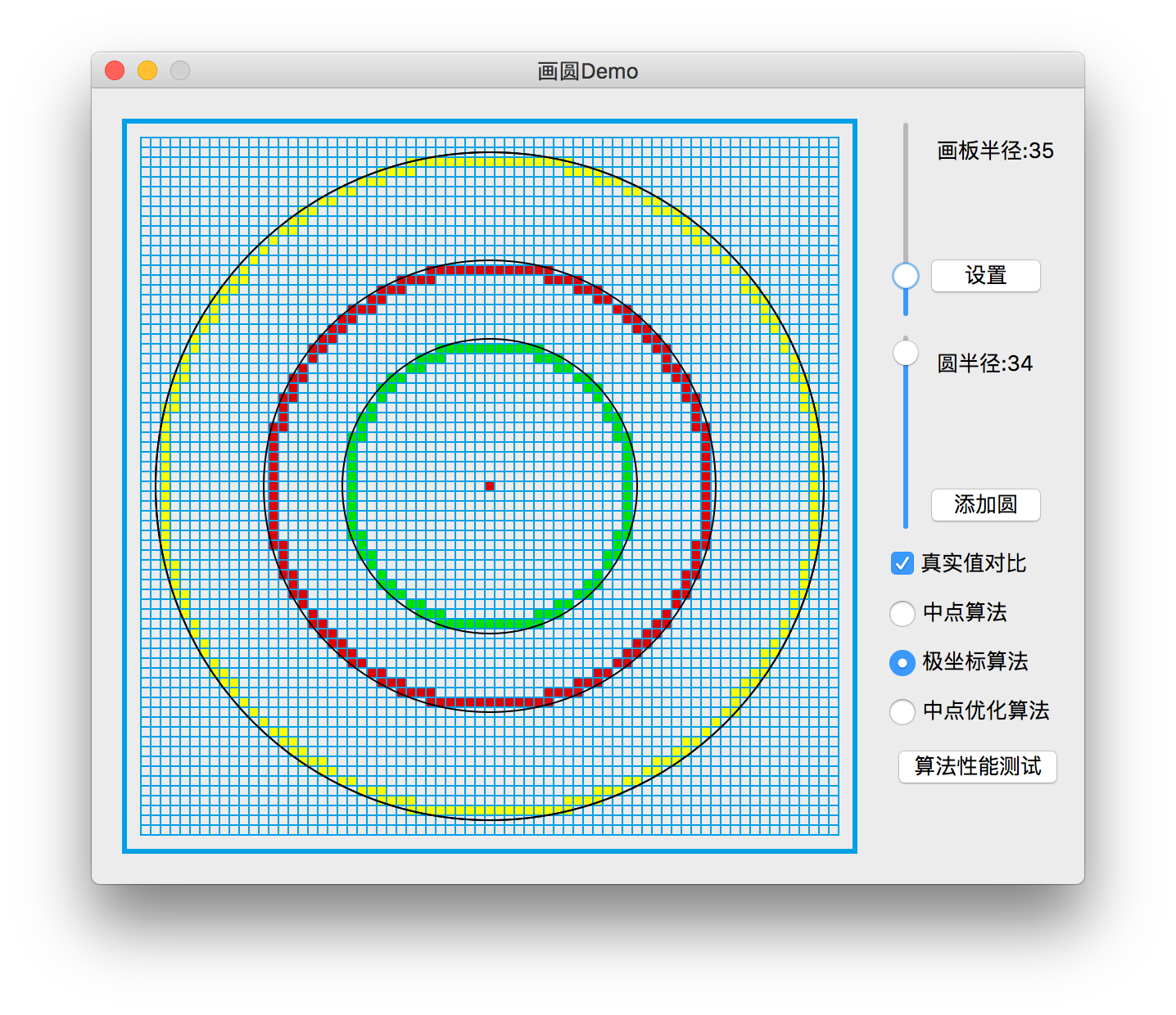


图4 程序使用极坐标算法绘制圆的效果

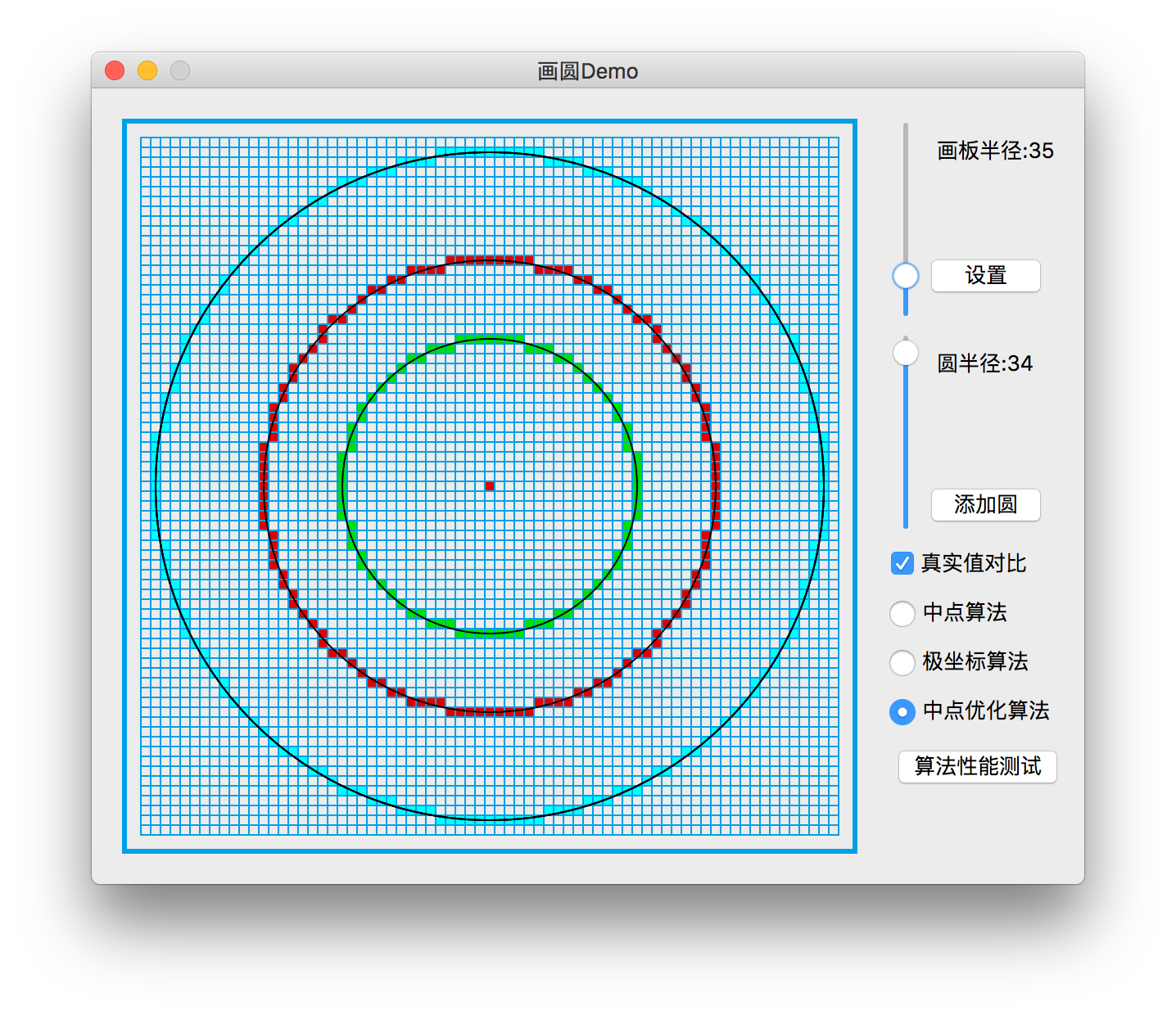


图5 程序使用中点优化算法绘制圆的效果

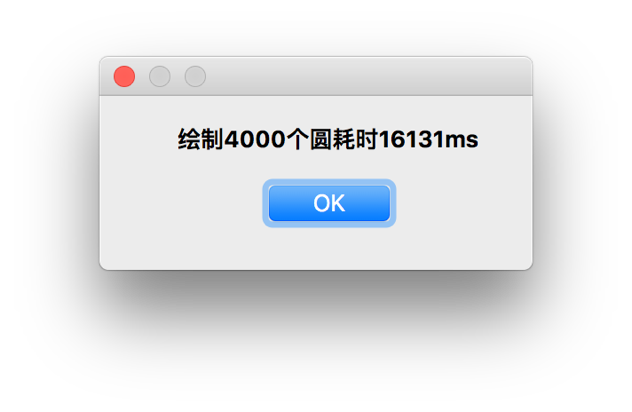


图6中点算法绘制4000个圆的耗时 图7极坐标算法绘制4000个圆的耗时



图8 使用中点优化算法绘制4000个圆耗时

|  |  |  |  |
| --- | --- | --- | --- |
|  | 中点算法 | 极坐标算法 | 中点优化算法 |
| 耗时（ms） | 2614 | 16131 | 2598 |

对比上述数据可见，使用中点优化算法绘制圆形耗时最少，极坐标算法较慢是因为需要计算三角函数，并且每次需要进行double到int的类型转换，比较耗时，故最慢。

**代码清单：**

**MainWindow.h**

**#ifndef MAINWINDOW\_H**

**#define MAINWINDOW\_H**

**#include <QMainWindow>**

**#include <QPainter>**

**#include <QPen>**

**#include <vector>**

**#include <QTime>**

**#include <QMessageBox>**

**#include <QDebug>**

**#include <math.h>**

***namespace* Ui {**

***class* MainWindow;**

**}**

***class* MainWindow : *public* QMainWindow**

**{**

**Q\_OBJECT**

***public*:**

***explicit* MainWindow(QWidget \*parent = 0);**

**~*MainWindow*();**

**void *paintEvent*(QPaintEvent \*event);**

**void drawPointAt(int x,int y,QPainter& painter,QColor color=QColor(225, 0, 0));**

***private*:**

**std::vector<QColor> colorList;**

***const* int gap=6;**

**int r;**

**int tempRadius;int tempCircleRadius;**

**int lineNum=70;*//strongly* *recommend* *to* *set* *this* *value* *to* *an* *even* *value***

**int boxWidth=gap\*(lineNum-1);**

***const* int startX=30;**

***const* int startY=30;**

**std::vector<int> testCircles;**

**std::vector<int> circles;**

**bool isTesting;**

**int drawAlgorithm;**

**bool cmp;**

**void drawCircleWithRadius(int r,QPainter &painter,QColor color=QColor(225, 0, 0));**

**void drawCircleWithRadiusPolar(int r,QPainter &painter,QColor color=QColor(225, 0, 0));**

**void drawCircleWithRadiusInteger(int r,QPainter &painter,QColor color=QColor(225, 0, 0));**

**void drawCircleWithRadiusIntegerOptimized(int r,QPainter &painter,QColor color=QColor(225, 0, 0));**

***public* *slots*:**

**void setNewRadius();**

**void setNewCircleRadius();**

**void addCircle();**

**void resetPaintArea();**

**void setCmpValue();**

**void runPerformanceTest();**

***private* *slots*:**

**void on\_original\_clicked(bool checked);**

**void on\_integer\_clicked(bool checked);**

**void on\_optimized\_clicked(bool checked);**

***private*:**

**Ui::MainWindow \*ui;**

**};**

**#endif *//* *MAINWINDOW\_H***

**MainWindow.cpp**

**#include "mainwindow.h"**

**#include "ui\_mainwindow.h"**

**MainWindow::MainWindow(QWidget \*parent) :**

**QMainWindow(parent),**

**ui(*new* Ui::MainWindow)**

**{**

**ui->setupUi(*this*);**

**connect(ui->radiusSetter,*SIGNAL*(valueChanged(int)),*this*,*SLOT*(setNewRadius()));**

**connect(ui->circleRadiusSetter,*SIGNAL*(valueChanged(int)),*this*,*SLOT*(setNewCircleRadius()));**

**connect(ui->setRadius,*SIGNAL*(clicked(bool)),*this*,*SLOT*(resetPaintArea()));**

**connect(ui->addCircle,*SIGNAL*(clicked(bool)),*this*,*SLOT*(addCircle()));**

**connect(ui->cmpChecker,*SIGNAL*(clicked(bool)),*this*,*SLOT*(setCmpValue()));**

**connect(ui->performanceTest,*SIGNAL*(clicked(bool)),*this*,*SLOT*(runPerformanceTest()));**

**ui->radiusSetter->setValue(35);**

**ui->circleRadiusSetter->setValue(15);**

**resetPaintArea();**

**r=(lineNum/2-1);**

**colorList.push\_back(QColor(225,0,0));**

**colorList.push\_back(QColor(0,225,0));**

**colorList.push\_back(QColor(0,0,225));**

**colorList.push\_back(QColor(255,255,0));**

**colorList.push\_back(QColor(0,255,255));**

**colorList.push\_back(QColor(255,0,255));**

**drawAlgorithm=0;**

**}**

**MainWindow::~*MainWindow*(){**

***delete* ui;**

**}**

**void MainWindow::drawPointAt(int x, int y, QPainter &painter,QColor color){**

**QBrush tempBrush(color,Qt::*SolidPattern*);**

**painter.setBrush(tempBrush);**

**int realX=startX+gap\*(x+r);**

**int realY=startY+gap\*(r-y);**

**painter.drawRect(realX,realY,gap,gap);**

**}**

**void MainWindow::setNewRadius(){**

**tempRadius=ui->radiusSetter->value();**

**ui->radius->setText(QString("画板半径:")+QString::number(tempRadius));**

**}**

**void MainWindow::setCmpValue(){**

**cmp=ui->cmpChecker->isChecked();**

***this*->repaint();**

**}**

**void MainWindow::setNewCircleRadius(){**

**tempCircleRadius=ui->circleRadiusSetter->value();**

**ui->circleRadius->setText(QString("圆半径:")+QString::number(tempCircleRadius));**

**}**

**void MainWindow::resetPaintArea(){**

**r=tempRadius;**

**lineNum=2\*(r+1);**

**boxWidth=gap\*(lineNum-1);**

***this*->setFixedHeight(30\*2+boxWidth);**

***this*->setFixedWidth(30+boxWidth+150);**

**ui->radiusSetter->setGeometry(30+boxWidth+30,20,22,120);**

**ui->setRadius->setGeometry(30+boxWidth+50,100,80,32);**

**ui->radius->setGeometry(30+boxWidth+60,30,75,16);**

**ui->circleRadiusSetter->setMaximum(r);**

**ui->circleRadiusSetter->setGeometry(30+boxWidth+30,150,22,120);**

**ui->addCircle->setGeometry(30+boxWidth+50,240,80,32);**

**ui->circleRadius->setGeometry(30+boxWidth+60,160,75,16);**

**ui->cmpChecker->setGeometry(30+boxWidth+30,280,91,20);**

**ui->original->setGeometry(30+boxWidth+30,310,100,20);**

**ui->integer->setGeometry(30+boxWidth+30,340,100,20);**

**ui->optimized->setGeometry(30+boxWidth+30,370,100,20);**

**ui->performanceTest->setGeometry(30+boxWidth+30,400,110,32);**

**}**

**void MainWindow::addCircle(){**

**circles.push\_back(tempCircleRadius);**

***this*->repaint();**

**}**

**void MainWindow::drawCircleWithRadius(int r,QPainter &painter,QColor color){**

***if*(r>*this*->r)*return*;**

**int x=0;**

**int y=r;**

**double delta=1.25-r;**

***while*(y>=x){**

***//draw***

**drawPointAt(x,y,painter,color);**

**drawPointAt(-x,y,painter,color);**

**drawPointAt(x,-y,painter,color);**

**drawPointAt(-x,-y,painter,color);**

**drawPointAt(y,x,painter,color);**

**drawPointAt(-y,x,painter,color);**

**drawPointAt(y,-x,painter,color);**

**drawPointAt(-y,-x,painter,color);**

***if*(delta<0){**

**delta+=2\*x+3.0;**

**}*else*{**

**delta+=2\*(x-y)+5.0;**

**--y;**

**}**

**++x;**

**}**

**}**

**void MainWindow::drawCircleWithRadiusPolar(int r, QPainter &painter, QColor color){**

***if*(r>*this*->r)*return*;**

***for*(float i=0.01;i<=6.28;i=i+0.01){**

**drawPointAt(r\*cos(i),r\*sin(i),painter,color);**

**}**

**}**

**void MainWindow::drawCircleWithRadiusInteger(int r, QPainter &painter, QColor color){**

***if*(r>*this*->r)*return*;**

**int x=0;**

**int y=r;**

**int delta=1-r;**

***while*(y>=x){**

***//draw***

**drawPointAt(x,y,painter,color);**

**drawPointAt(-x,y,painter,color);**

**drawPointAt(x,-y,painter,color);**

**drawPointAt(-x,-y,painter,color);**

**drawPointAt(y,x,painter,color);**

**drawPointAt(-y,x,painter,color);**

**drawPointAt(y,-x,painter,color);**

**drawPointAt(-y,-x,painter,color);**

***if*(delta<0){**

**delta+=2\*x+3;**

**}*else*{**

**delta+=2\*(x-y)+5;**

**--y;**

**}**

**++x;**

**}**

**}**

**void MainWindow::drawCircleWithRadiusIntegerOptimized(int r, QPainter &painter, QColor color){**

***if*(r>*this*->r)*return*;**

**int x=0;**

**int y=r;**

**int delta=1-r;**

**int delta\_t=3;**

**int delta\_b=-2\*r+5;**

***while*(y>=x){**

**drawPointAt(x,y,painter,color);**

**drawPointAt(-x,y,painter,color);**

**drawPointAt(x,-y,painter,color);**

**drawPointAt(-x,-y,painter,color);**

**drawPointAt(y,x,painter,color);**

**drawPointAt(-y,x,painter,color);**

**drawPointAt(y,-x,painter,color);**

**drawPointAt(-y,-x,painter,color);**

***if*(delta<0){**

**delta+=delta\_t;**

**delta\_t+=2;**

**delta\_b+=2;**

**}*else*{**

**delta+=delta\_b;**

**delta\_t+=2;**

**delta\_b+=4;**

**--y;**

**}**

**++x;**

**}**

**}**

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

**QPainter painter(*this*);**

**painter.setRenderHint(QPainter::*Antialiasing*, *true*);**

**QPen tempPen(QColor(0, 160, 230));**

**tempPen.setWidth(3);**

**painter.setPen(tempPen);**

***const* int boardOffset=10;**

**painter.drawLine(startX-boardOffset,startY-boardOffset,startX+boardOffset+boxWidth,startY-boardOffset);**

**painter.drawLine(startX-boardOffset,startY+boxWidth+boardOffset,startX+boardOffset+boxWidth,startY+boardOffset+boxWidth);**

**painter.drawLine(startX-boardOffset,startY-boardOffset,startX-boardOffset,startY+boardOffset+boxWidth);**

**painter.drawLine(startX+boardOffset+boxWidth,startY-boardOffset,startX+boxWidth+boardOffset,startY+boardOffset+boxWidth);**

**tempPen.setWidth(1);**

**painter.setPen(tempPen);**

***for*(int i=0;i<lineNum;++i){**

**painter.drawLine(startX,startY+i\*gap,startX+boxWidth,startY+i\*gap);**

**}**

***for*(int j=0;j<lineNum;++j){**

**painter.drawLine(startX+j\*gap,startY,startX+j\*gap,startY+boxWidth);**

**}**

**drawPointAt(0,0,painter);**

**int counter=0;**

**int colorListSize=colorList.size();**

**int zeroX=startX+gap\*r+gap/2;**

**int zeroY=startY+gap\*r+gap/2;**

**QPoint center(zeroX,zeroY);**

***if*(isTesting){**

**QTime startTime;**

**startTime.start();**

***for*(*auto* &c:testCircles){**

***if*(drawAlgorithm==0) *drawCircleWithRadius(c,painter,colorList.at(counter++%colorListSize));***

**drawCircleWithRadiusInteger(c,painter,colorList.at(counter++%colorListSize));**

***else* *if*(drawAlgorithm==1)**

**drawCircleWithRadiusPolar(c,painter,colorList.at(counter++%colorListSize));**

***else***

**drawCircleWithRadiusIntegerOptimized(c,painter,colorList.at(counter++%colorListSize));**

**}**

**isTesting=*false*;**

**QMessageBox::about(*this*,tr("绘图完成"),tr("绘制4000个圆耗时")+QString::number(startTime.elapsed())+tr("ms"));**

***return*;**

**}**

***for*(*auto* &c:circles){**

***if*(drawAlgorithm==0)**

**drawCircleWithRadiusInteger(c,painter,colorList.at(counter++%colorListSize));**

***else* *if*(drawAlgorithm==1)**

**drawCircleWithRadiusPolar(c,painter,colorList.at(counter++%colorListSize));**

***else***

**drawCircleWithRadiusIntegerOptimized(c,painter,colorList.at(counter++%colorListSize));**

**}**

***if*(cmp){**

**QPen blackPen(QColor(0,0,0));**

**painter.setPen(blackPen);**

**painter.setBrush(Qt::*NoBrush*);**

***for*(*auto* &c:circles){**

***if*(c>r)*continue*;**

**painter.drawEllipse(center,c\*gap,c\*gap);**

**}**

**}**

**}**

**void MainWindow::runPerformanceTest(){**

**isTesting=*true*;**

**testCircles.clear();**

***for*(int i=1;i<=4000;++i){**

**testCircles.push\_back(i%r);**

**}**

***this*->repaint();**

**}**

**void MainWindow::on\_original\_clicked(bool checked){**

**drawAlgorithm=0;**

**}**

**void MainWindow::on\_integer\_clicked(bool checked){**

**drawAlgorithm=1;**

**}**

**void MainWindow::on\_optimized\_clicked(bool checked){**

**drawAlgorithm=2;**

**}**