Московский Авиационный Институт
(Национальный Исследовательский Университет)

Факультет: “Информационные технологии и прикладная математика”
Кафедра: 806 “Вычислительная математика и программирование”

Лабораторная работа №1

по курсу “Компьютерная графика”

Студент	Полей-Добронравова А.В.
Группа	М8О-307Б-18
Преподаватель	Г.С.Филиппов
Вариант	4
Дата	
Оценка	

Москва, 2020

Построение изображений 2D-кривых

Задача:
Написать и отладить программу, строящую изображение заданной замечательной кривой. Обеспечить автоматическое масштабирование и центрирование кривой при изменении размеров окна

Вариант 4
Кривая $\rho^2=a^2\cos(2\varphi)$ в полярных координатах, a - параметр, вводимый пользователем

Описание

Программа написана на языке C++ в QTCreator.
Создано три класса:

1) mainwindow - главное окно приложения;
2) mygraphicview - графические изображения, отрисовываемые в mainwindow;
3) demodialog - класс окна диалога для ввода параметра a.

В mainwindow подключение user interface, создаётся графика. Эта графика так же передаётся конструктору demodialog из main.cpp. В момент ввода в диалоговое окно значения a и нажатия на “Ок”, слот void DemoDialog::sl1() записывает в поле класса графики значение параметра, запускает перерисовку графики и закрывает диалоговое окно.

В mygraphicview создаётся сцена с двумя группами объектов - оси координат, которые будут отрисовываться черной кистью и сама функция красной кистью. В начале каждой перерисовки сцены запускается очистка групп графических объектов с помощью void MyGraphicView::deleteItemsFromGroup(QGraphicsItemGroup *group). Перерисовка графики задаётся таймером timer->start(50); в слоте void MyGraphicView::slotAlarmTimer(). Первая группа объектов рисуется в зависимости от текущей ширины и высоты экрана. Функцию для отрисовки я перевела в декартовы координаты:
x = +-a * cos(φ) * sqrt(cos(2φ));
y = +-a * sin(φ) * sqrt(cos(2φ));
С помощью цикла изменения угла φ от нуля до 2π, я создаю два полигона из точек, удовлетворяющих уравнению, и добавляю во вторую группу для отрисовки.
Кроме того существует void MyGraphicView::*resizeEvent*(QResizeEvent *event), который отлавливает изменение размера окна и перерисовывает графику.

Код

main.cpp

```
#include "mainwindow.h"
#include "demodialog.h"

#include <QApplication>
#include <QShortcut>
#include <QObject>

int main(int argc, char *argv[])
{
   QApplication a(argc, argv);

   MainWindow* w = new MainWindow(0);
   w->show();
   DemoDialog* d = new DemoDialog(0,w->myPicture);
   d->show();
   return a.exec();
}
```

mainwindow.h

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
#include "mygraphicview.h"

namespace Ui {
   class MainWindow;
}
class MainWindow : public QMainWindow
{
   Q_OBJECT

public:
   MainWindow(QWidget *parent = 0);
   ~MainWindow();
   MyGraphicView *myPicture; // Кастомный виджет

private:
   Ui::MainWindow *ui;
};

#endif // MAINWINDOW_H
```

mainwindow.cpp

```
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include "mygraphicview.h"

MainWindow::MainWindow(QWidget *parent)
   : QMainWindow(parent)
   , ui(new Ui::MainWindow)
{
   ui->setupUi(this);
   myPicture = new MyGraphicView(0);
   setCentralWidget(myPicture);
}

MainWindow::~MainWindow()
{
   delete ui;
}
```

mygraphicview.h

```
#ifndef MYGRAPHICVIEW_H
#define MYGRAPHICVIEW_H

#include <QWidget>
#include <QGraphicsView>
#include <QGraphicsScene>
#include <QGraphicsItemGroup>
#include <QGraphicsTextItem>
#include <QTimer>
#include <QInputDialog>
#include <QPoint>


class MyGraphicView : public QGraphicsView
{
   Q_OBJECT
public:
   MyGraphicView(QWidget *parent = 0);
   ~MyGraphicView();
   float a = 100;

signals:

public slots:
   void slotAlarmTimer();

private:
   QGraphicsScene      *scene;
   QGraphicsItemGroup  *group_1;
   QGraphicsItemGroup  *group_2;
   QTimer              *timer;

private:
   void resizeEvent(QResizeEvent *event);
    void deleteItemsFromGroup(QGraphicsItemGroup *group_1);
};

#endif // MYGRAPHICVIEW_H
```

mygraphicview.cpp

```
#include "mygraphicview.h"
#include <QtMath>

MyGraphicView::MyGraphicView(QWidget *parent): QGraphicsView(parent)
{
   this->setHorizontalScrollBarPolicy(Qt::ScrollBarAlwaysOff); // Отключим
скроллбар по горизонтали
   this->setVerticalScrollBarPolicy(Qt::ScrollBarAlwaysOff);  // Отключим скроллбар
повертикали
   this->setAlignment(Qt::AlignCenter); // Делаем привязку содержимого к центру
   this->setSizePolicy(QSizePolicy::Expanding, QSizePolicy::Expanding); //
Растягиваем содержимое по виджету

   this->setMinimumHeight(100);
   this->setMinimumWidth(100);

   scene = new QGraphicsScene();
   this->setScene(scene);

   group_1 = new QGraphicsItemGroup(); //ось координат
   group_2 = new QGraphicsItemGroup(); //функция

   scene->addItem(group_1);
   scene->addItem(group_2);
```

```
    timer = new QTimer();
    timer->setSingleShot(true);
    connect(timer, SIGNAL(timeout()), this, SLOT(slotAlarmTimer()));
     timer->start(50);
}

MyGraphicView::~MyGraphicView() {

}
void MyGraphicView::slotAlarmTimer() {
    this->deleteItemsFromGroup(group_1);
    this->deleteItemsFromGroup(group_2);
    int width = this->width(); // определяем ширину нашего виджета
    int height = this->height();
    /* Устанавливаем размер сцены по размеру виджета
         * Первая координата - это левый верхний угол,
         * а Вторая - это правый нижний угол
         * */
    scene->setSceneRect(0,0,width,height);

    QPen penBlack(Qt::black); // Задаём чёрную кисть
    QPen penRed(Qt::red);   // Задаём красную кисть
    //рисуем черные оси и красный график
    group_1->addToGroup(scene->addLine(width / 2, 20, width / 2, height - 20,
penBlack));
    group_1->addToGroup(scene->addLine(20,height / 2, width - 20, height / 2,
penBlack));
    group_1->addToGroup(scene->addLine(width / 2, 20, width / 2 - 15, 35,
penBlack));
    group_1->addToGroup(scene->addLine(width / 2, 20, width / 2 + 15, 35,
penBlack));
    group_1->addToGroup(scene->addLine(width - 20, height / 2, width - 35, height /
2 + 15, penBlack));
    group_1->addToGroup(scene->addLine(width - 20, height / 2, width - 35, height /
2 - 15, penBlack));
    QGraphicsTextItem *textY = new QGraphicsTextItem("Y", 0);
    textY->setPos(width / 2 - 33, 20);
    QGraphicsTextItem *textX = new QGraphicsTextItem("X", 0);
    textX->setPos(width - 30, height / 2 + 15);
    group_1->addToGroup(textY);
    group_1->addToGroup(textX);
    int step = (width - 40) / 20;
    int istep = 0;
    for (int i = width / 2 - step + 13; i < width - 60; i = i + step) {
        QGraphicsTextItem *textS = new QGraphicsTextItem(QString::number(istep), 0);
        istep = istep + step;
        textS->setPos(20 + i, height / 2 -1);
        group_1->addToGroup(textS);
    }
    istep = -step;
    for (int i = width / 2 - step - step; i > step; i = i - step) {
        QGraphicsTextItem *textS = new QGraphicsTextItem(QString::number(istep), 0);
        istep = istep - step;
        textS->setPos(20 + i, height / 2 - 1);
        group_1->addToGroup(textS);
    }
    //step = (height - 20) / 20;
    istep = -step;
    for (int i = height / 2 + step - 13; i < height - step; i = i + step) {
        QGraphicsTextItem *textS = new QGraphicsTextItem(QString::number(istep), 0);
        istep = istep - step;
        textS->setPos(width / 2 + 7, i);
        group_1->addToGroup(textS);
    }
    istep = step;
    for (int i = height / 2 - step - 13; i > step * 2; i = i - step) {
        QGraphicsTextItem *textS = new QGraphicsTextItem(QString::number(istep), 0);
```

```
        istep = istep + step;
        textS->setPos(width / 2 + 7, i);
        group_1->addToGroup(textS);
    }
    step = (width - 40) / 20;
    for (int i = width / 2 - 1; i < width - step; i = i + step) {
        group_1->addToGroup(scene->addLine(i,height / 2, i, height / 2 + 6,
penBlack));
    }
    for (int i = width / 2 - 1; i > step; i = i - step) {
        group_1->addToGroup(scene->addLine(i,height / 2, i, height / 2 + 6,
penBlack));
    }
    //step = (height - 20) / 20;
    for (int i = height / 2 - 1; i < height - step; i = i + step) {
        group_1->addToGroup(scene->addLine(width / 2,i, width / 2 + 6, i,
penBlack));
    }
    for (int i = height / 2 - 1; i > step; i = i - step) {
        group_1->addToGroup(scene->addLine(width / 2,i, width / 2 + 6, i,
penBlack));
    }
//сам график
    QPolygonF polygon;
    QPolygonF polygon1;
    for (float f = 0; f <= 2 * M_PI; f = f + 0.000005) {
        polygon << QPointF(width/2  + a * cos(f)* sqrt(cos(2 * f)), height / 2 - a *
sin(f)* sqrt(cos(2 * f)));
        polygon1 << QPointF(width/2 - a * sin(f)* sqrt(cos(2 * f)), height / 2 + a *
cos(f)* sqrt(cos(2 * f)));
    }
    group_2->addToGroup(scene->addPolygon(polygon,penRed));
    group_2->addToGroup(scene->addPolygon(polygon1,penRed));
}

void MyGraphicView::resizeEvent(QResizeEvent *event) {
    timer->start(50);
    QGraphicsView::resizeEvent(event);
}


void MyGraphicView::deleteItemsFromGroup(QGraphicsItemGroup *group) {
    foreach( QGraphicsItem *item, scene->items(group->boundingRect())) {
        if(item->group() == group ) {
            delete item;
        }
    }
}
```

demodialog.h

```
#ifndef DEMODIALOG_H
#define DEMODIALOG_H

#include <QDialog>
#include <QLineEdit>
#include <QBoxLayout>
#include <QPushButton>
#include <QLabel>
#include <QFormLayout>
#include "mygraphicview.h"

class DemoDialog : public QDialog
{
    Q_OBJECT
public:
    DemoDialog( QWidget* parent = 0, MyGraphicView *myPicture = 0);
```

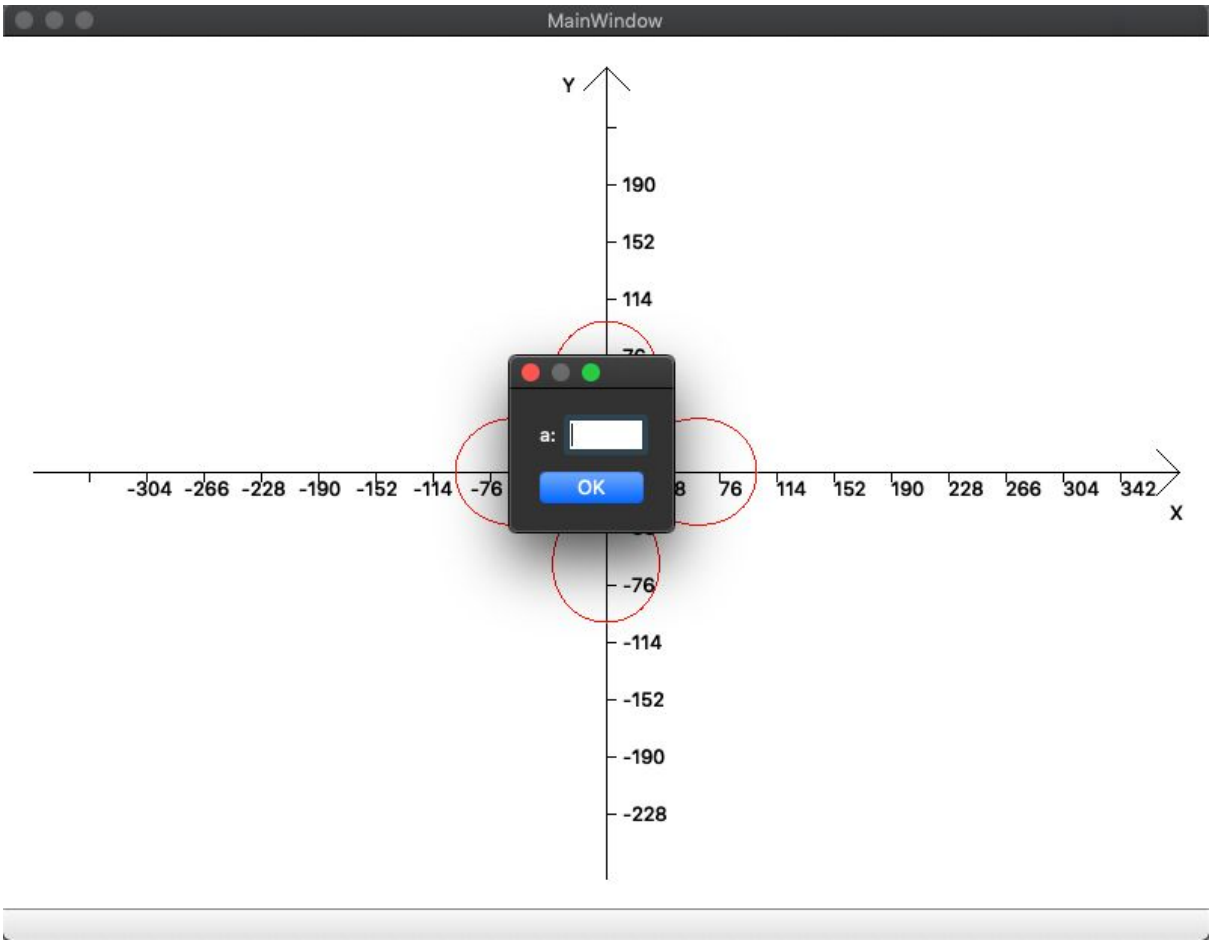
```
    ~DemoDialog();
    QString getInput() const;

    signals:
        void applied();
        void clicked();
    private slots:
        void sl1();
    private:
        MyGraphicView *my;
        QLabel* m_edit1;
        QLineEdit* l1;
        QFormLayout *layout;
        QPushButton* okBtn;
    };

#endif // DEMODIALOG_H
```

demodialog.cpp

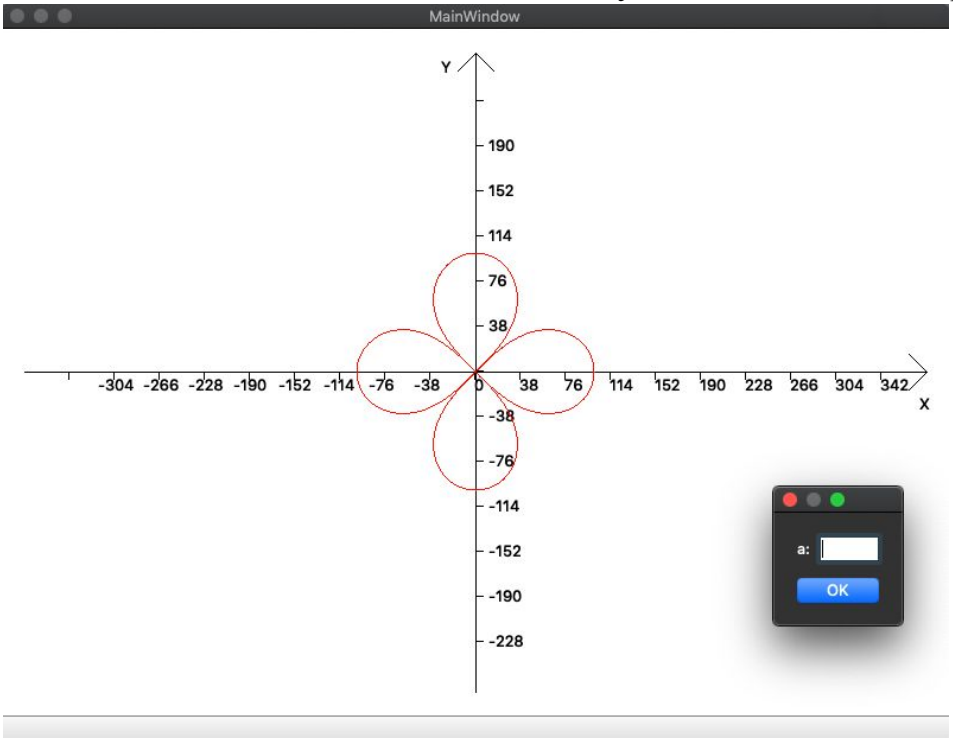
```
#include "demodialog.h"
#include<QDebug>

DemoDialog::DemoDialog( QWidget* parent, MyGraphicView *myPicture) : QDialog(
parent ) {
    this->my = myPicture;
    layout = new QFormLayout;
    m_edit1 = new QLabel(this);
    m_edit1->setText("a:");
    m_edit1->setStyleSheet("color: rgb(255, 255, 255)");
    m_edit1->setObjectName("a:");
    m_edit1->setVisible(true);
    l1 = new QLineEdit(this);
    l1->setVisible(true);
    layout->addRow(m_edit1, l1);
    okBtn = new QPushButton( "OK" );
    connect( okBtn, SIGNAL( clicked() ), SLOT( sl1() ) );
    layout->addRow( okBtn );
    this->resize(this->sizeHint());
    setLayout( layout );
}

DemoDialog::~DemoDialog() {
}

QString DemoDialog::getInput() const {
    return m_edit1->text();
}

void DemoDialog::sl1() {
    my->a = this->l1->text().toFloat();
    my->slotAlarmTimer();
    accept();
}
```

Пример работы

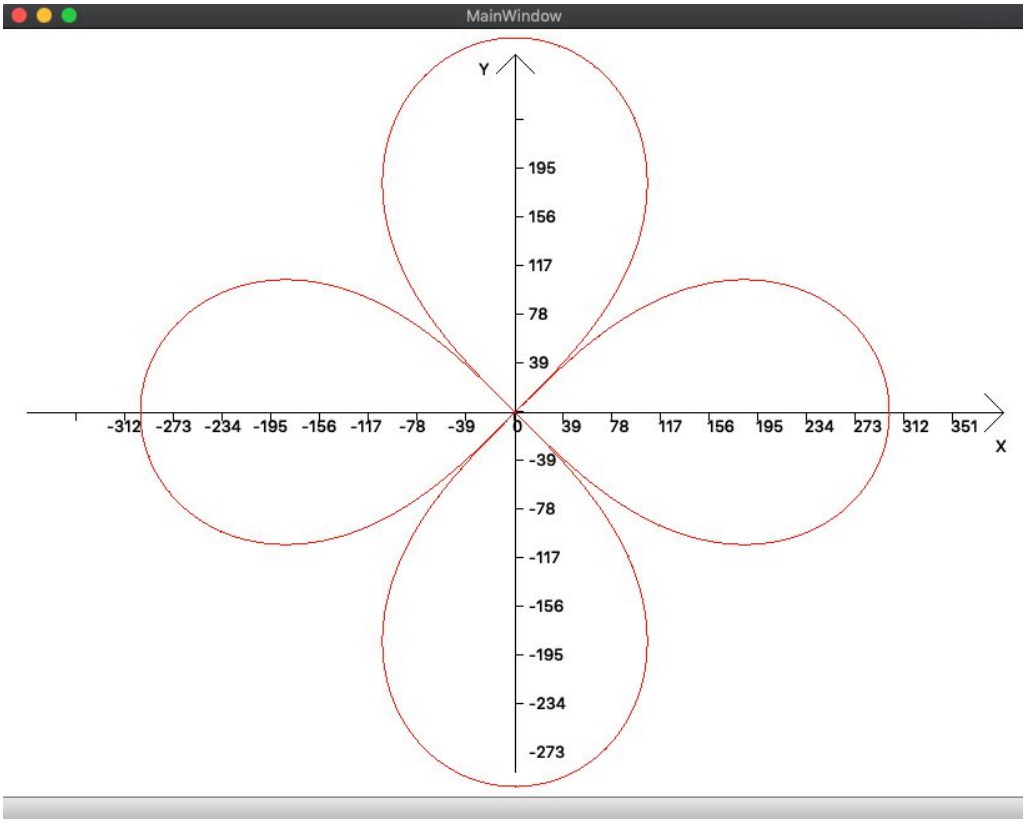
Дефолтное значение параметра a для предварительного вывода равно 100, и при запуске программы появляется это:



Если отвести диалоговое окно, можно увидеть полностью график с a=100:



Введем другое значение a, например a = 300:



Лепестки расширились, максимальное значение координаты x становится большим 273, в прошлом примере оно было около 85.
Если не меняя значение а сжать окно, то:

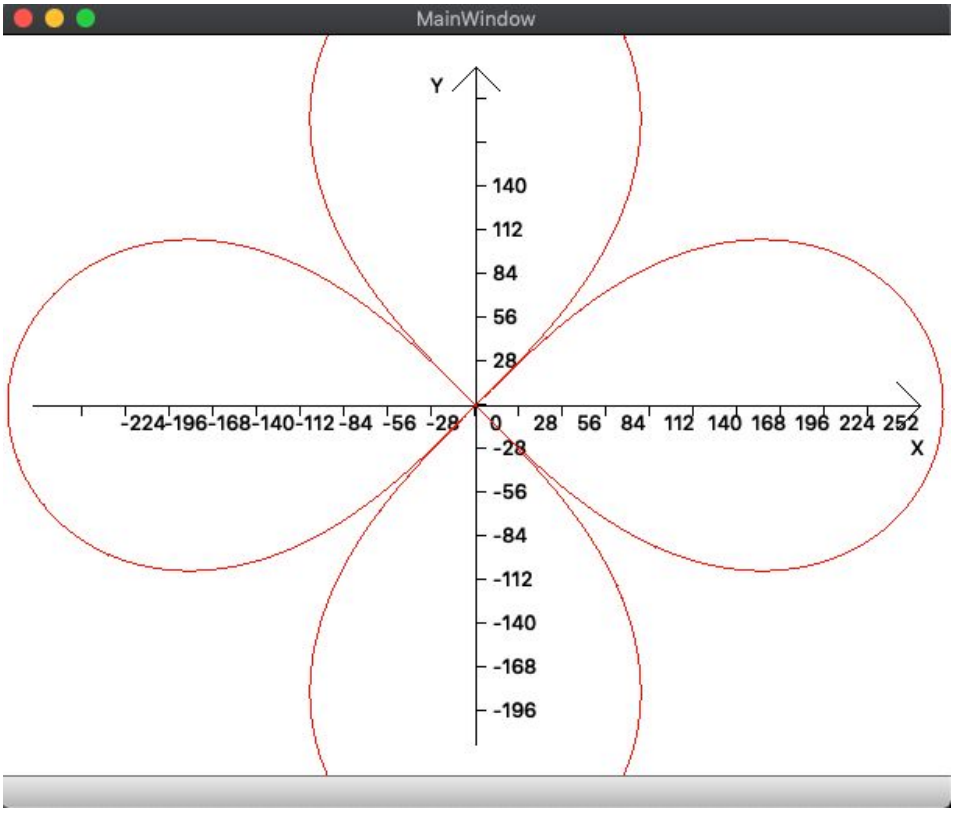


График функции имеет те же координаты, масштаб же осей и отображаемого шага изменился, лепестки функции теперь выходят за края осей, и шаг с 39 уменьшился до 28.

Вывод

Занимаясь компьютерной графикой, нужно всегда строить логику своего кода в зависимости от размеров окон приложения, взаимодействия пользователей с UI. Нужно понимать принцип координатной сетки и хорошо ориентироваться в функциях выбранной библиотеки.