МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В.Г.ШУХОВА»
(БГТУ им. В.Г.Шухова)

Лабораторная работа № 1
дисциплина «Компьютерная графика»
по теме «Графические примитивы»

Выполнил: студент группы ВТ-31 Макаров Д.С.
Проверил: Осипов О.В.

Белгород 2019

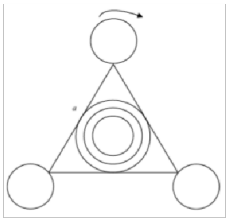
Лабораторная работа № 1

«Графические примитивы»

Цель работы: Изучение графических 2d-примитивов с использованием GDI в среде Qt.

Вариант 9

Задание: a – сторона треугольника, n – количество сторон многоугольника, вписанных в окружности. n вводится с клавиатуры. Реализовать вращение всей фигуры по часовой стрелке. Раскрасить все элементы по своему усмотрению.



Ход работы

Используемые формулы в построении фигуры.

Расчет шага α:

$$360/n$$

Получение n-угольника:

$$T_n = (r * cos(\alpha) + center_x; r * sin(\alpha) + center_y),$$

Для поворота фигур была использована матрица поворота.

$$x' = x * cos(\alpha) - y * sin(\alpha),$$
$$y' = x * sin(\alpha) + y * cos(\alpha).$$

Формула зависимости массива brushColors[5] от угла поворота.

$$(((i + 1) * 250 + \alpha/10)\%359; (i * 150)\%255; 150)_{HSL}$$

Формулы для радиусов вписанных в треугольник а окружностей.

$$r_1 = (height - 20|width - 20)/20$$
$$r_2 = r_1 + (height - 30|width - 30)$$
$$r_3 = r_2 + (height - 30|width - 30)$$

Сторона a вычисляется по радиусу наибольшей вписанной в него окружности.

$$a = 2 * \sqrt{3} * r_3$$

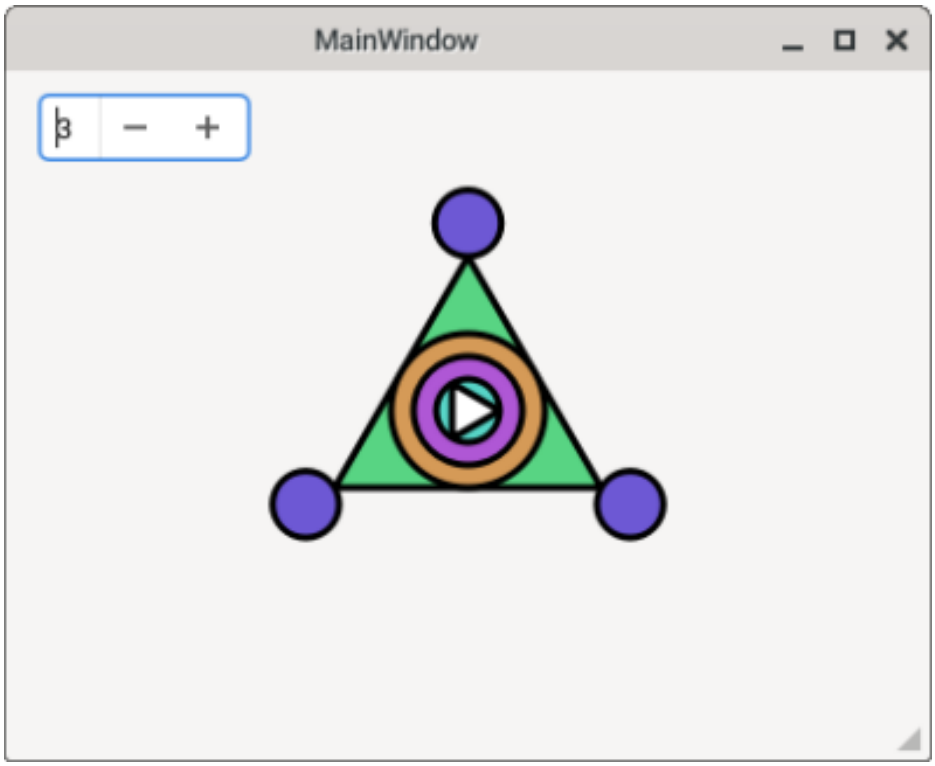
Радиус описанной окружности треугольника a.

$$R_a = \frac{\sqrt{3} * a}{3}$$

Окружности на ходящиеся на вершинах треугольника находятся на вершинах скрытого треугольника a' с радиусом описанной окружности на a/8 больше чем у треугольника a.

$$R_{a'} = R_a + \frac{a}{8}$$

Окно программы



Приложение

Содержимое файла mainwindow.cpp

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

MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow)
{
    ui->setupUi(this);
}

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

QRectF get_circle_area(qreal radius,QPointF center){
    QRectF area = QRectF(center.x() - radius, center.y() - radius, radius*
    return area;
}

qreal to_rad(qreal angle){
    return angle*3.14/180;
}

QPolygonF get_n_poly(int count_of_sides,QPointF center,qreal radius){
    QPolygonF poly;
    qreal angle_step = to_rad(360/count_of_sides);
    qreal current_angle = 0;
    for(int i=0;i<count_of_sides;i++){
        poly << QPointF(radius*qCos(current_angle)+center.x(),radius*qSin(
        current_angle+=angle_step;
    }
    return poly;
}


QPointF rotate(QPointF point,QPointF center,qreal angle){
    angle = to_rad(angle);
    qreal sin_A = qSin(angle);
    qreal cos_A = qCos(angle);
```

```
    QPointF temp_point = QPointF(point.x()-center.x(),point.y()-center.y()
    temp_point = QPointF(temp_point.x()*cos_A-temp_point.y()*sin_A,
                  temp_point.x()*sin_A+temp_point.y()*cos_A);
    return QPointF(temp_point.x()+center.x(),temp_point.y()+center.y());
}

QRectF rotate_QRectF(QRectF rect,QPointF center,qreal angle){
    rect.setBottomLeft(rotate(rect.bottomLeft(),center,angle));
    rect.setBottomRight(rotate(rect.bottomRight(),center,angle));
    rect.setTopLeft(rotate(rect.topLeft(),center,angle));
    rect.setTopRight(rotate(rect.topRight(),center,angle));
    return rect;
}

qreal hypotenuse (qreal side1,qreal side2){
    return qSqrt(qPow(side1,2)+qPow(side2,2));
}

void MainWindow::paintEvent(QPaintEvent* event)
{
    //отмена отрисовки при недостатке места.
    if (width() < 40 || height() < 40) return;

    QPainter painter(this);
    //включение НОРМАЛЬНОГО антиалиасинга.
    painter.setRenderHint(QPainter::HighQualityAntialiasing);
    //установка пера.
    QColor penColor(0,0,0);
    QPen borderPen(penColor,3);
    painter.setPen(borderPen);
    //установка цветов.
    int colorCount = 5;
    QColor brushColors[colorCount];
    for(int i=0;i<colorCount;i++){
        brushColors[i].setHsl(((i+1)*250+angle/10)%359,(i+1*150)%255,150);
    }
    //центр окна
    QPointF center = QPointF(width() / 2, height() / 2);

    //радиусы кругов находящихся в треугольнике. круг с inner_radius3 впис
    qreal inner_radius1,inner_radius2,inner_radius3;
    if (width() > height()){
        inner_radius1 = (height() - 20) /20;
```

```
    inner_radius2 = inner_radius1+(height()/30);
    inner_radius3 = inner_radius2+(height()/30);
}else{
    inner_radius1 = (width() - 20) /20;
    inner_radius2 = inner_radius1+(width()/30);
    inner_radius3 = inner_radius2+(width()/30);
}

qreal a = 2*qSqrt(3)*inner_radius3;
qreal outer_triangle_R = (qSqrt(3)/3)*a;

QPolygonF outer_triangle;
outer_triangle << QPointF(center.x(),center.y()-outer_triangle_R);
outer_triangle << QPointF(center.x()+a/2,center.y()+inner_radius3);
outer_triangle << QPointF(center.x()-a/2,center.y()+inner_radius3);
for(int i=0;i<3;i++) outer_triangle[i] = rotate(outer_triangle[i],cent

qreal end_circle_radius = a/8;
qreal a2_add = qSqrt((end_circle_radius*end_circle_radius)-(end_circle
qreal a2 = a+2*a2_add;
qreal circle_centers_R = qSqrt(6)/2*a2;
qreal circle_centers_r = qSqrt(3)/2*a2;

QPolygonF circle_centers;
circle_centers << QPointF(center.x(),center.y()-outer_triangle_R-end_c
circle_centers << QPointF(center.x()+(a/2)+a2_add,center.y()+inner_rad
circle_centers << QPointF(center.x()-(a/2)-a2_add,center.y()+inner_rad
for(int i=0;i<3;i++) circle_centers[i] = rotate(circle_centers[i],cent

QPolygonF inner_poly = get_n_poly(n,center,inner_radius1);
for(int i=0;i<n;i++) inner_poly[i] = rotate(inner_poly[i],center,angle

QRectF inner_circle_area1 = get_circle_area(inner_radius1,center);
QRectF inner_circle_area2 = get_circle_area(inner_radius2,center);
QRectF inner_circle_area3 = get_circle_area(inner_radius3,center);
QRectF end_circle_area1 = get_circle_area(end_circle_radius,circle_cen
QRectF end_circle_area2 = get_circle_area(end_circle_radius,circle_cen
QRectF end_circle_area3 = get_circle_area(end_circle_radius,circle_cen


painter.setBrush(brushColors[0]);
painter.drawEllipse(end_circle_area1);
painter.drawEllipse(end_circle_area2);
```

```
    painter.drawEllipse(end_circle_area3);
    painter.setBrush(brushColors[1]);
    painter.drawPolygon(outer_triangle);
    painter.setBrush(brushColors[2]);
    painter.drawEllipse(inner_circle_area3);
    painter.setBrush(brushColors[3]);
    painter.drawEllipse(inner_circle_area2);
    painter.setBrush(brushColors[4]);
    painter.drawEllipse(inner_circle_area1);
    painter.setBrush(Qt::white);
    painter.drawPolygon(inner_poly);
}

void MainWindow::wheelEvent(QWheelEvent* wheelevent)
{
    angle -= wheelevent->delta()/10;
    repaint();
}

void MainWindow::on_spinBox_valueChanged(int arg1)
{
    this->n=arg1;
    repaint();
}
```

Содержимое файла mainwindow.h

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
#include <QPainter>
#include <QWheelEvent>
#include <QtMath>
#include <QSpinBox>

namespace Ui {
class MainWindow;
}

class MainWindow : public QMainWindow
{
    Q_OBJECT
```

```
public:
    explicit MainWindow(QWidget *parent = 0);
    ~MainWindow();

private slots:
    void on_spinBox_valueChanged(int arg1);

private:
    Ui::MainWindow *ui;

    int angle,n=3;

    // Обработчик события перерисовки окна
    void paintEvent(QPaintEvent* event);

    // Обработчик события прокрутки колеса мыши
    void wheelEvent(QWheelEvent* );
};

#endif // MAINWINDOW_H
```