

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)**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

Лабораторная работа №1

по дисциплине: Компьютерная графика
тема: «Графические примитивы GDI»

Выполнил: ст. группы ПВ-201
Морозов Данила Александрович

Проверил:
Осипов Олег Васильевич

Белгород 2022 г.

Лабораторная работа №1
«Графические примитивы GDI»

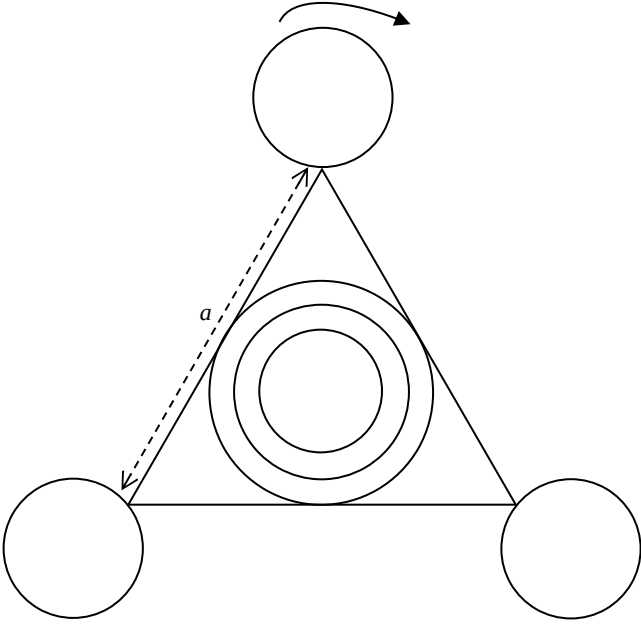
Цель работы:

Изучение графических 2D-примитивов с использованием GDI в среде Qt Creator.

Задания к работе:

1. Изучить графические примитивы библиотеки Qt (<http://doc.qt.io/qt-4.8/QPainter.html>).
2. Разработать алгоритм и составить программу для построения на экране изображения в соответствии с номером варианта. В качестве исходных данных взять указанные в таблицы №1.

Вариант №1:

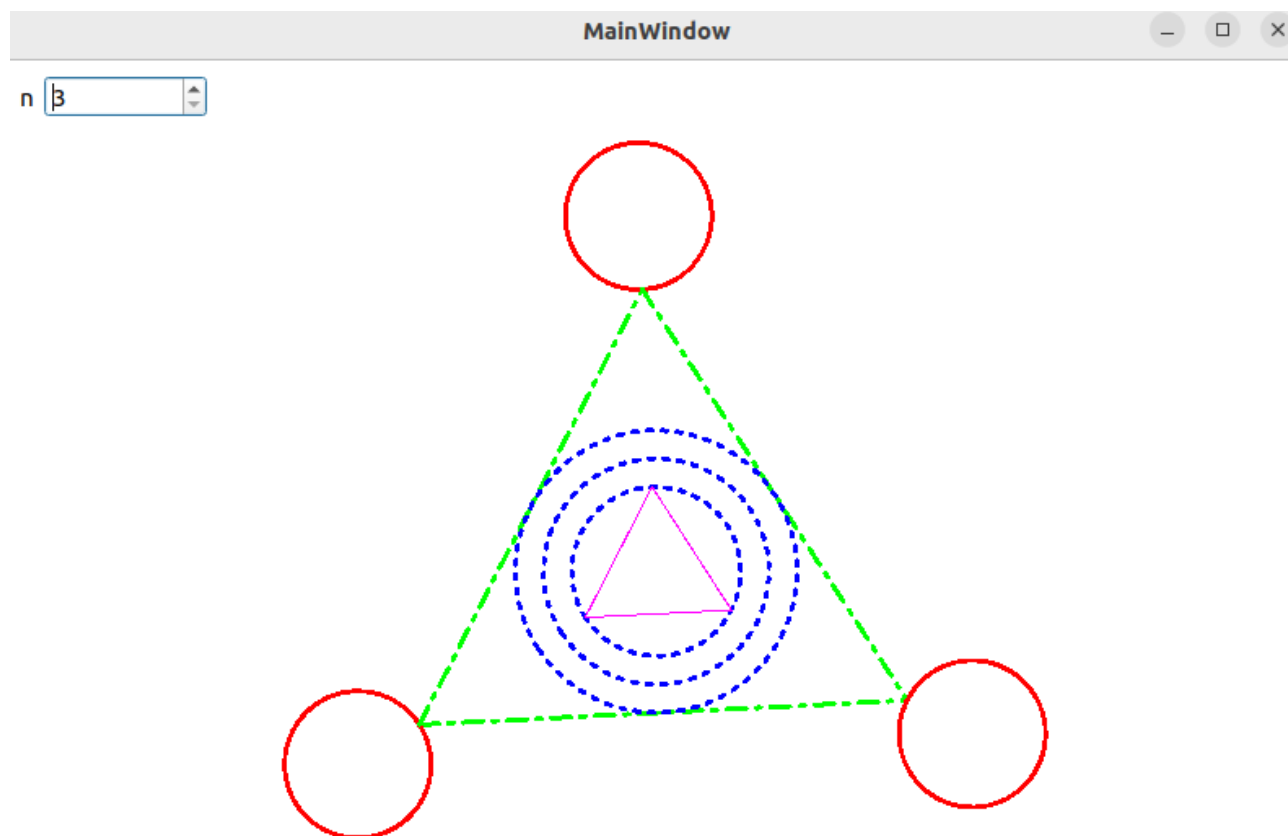
Вариант	Рисунок	Исходные данные
9		a – сторона треугольника, n – количество сторон многоугольника, вписанных в окружности. n вводится с клавиатуры. Реализовать вращение всей фигуры по часовой стрелке. Раскрасить все элементы по своему усмотрению.

Оглавление

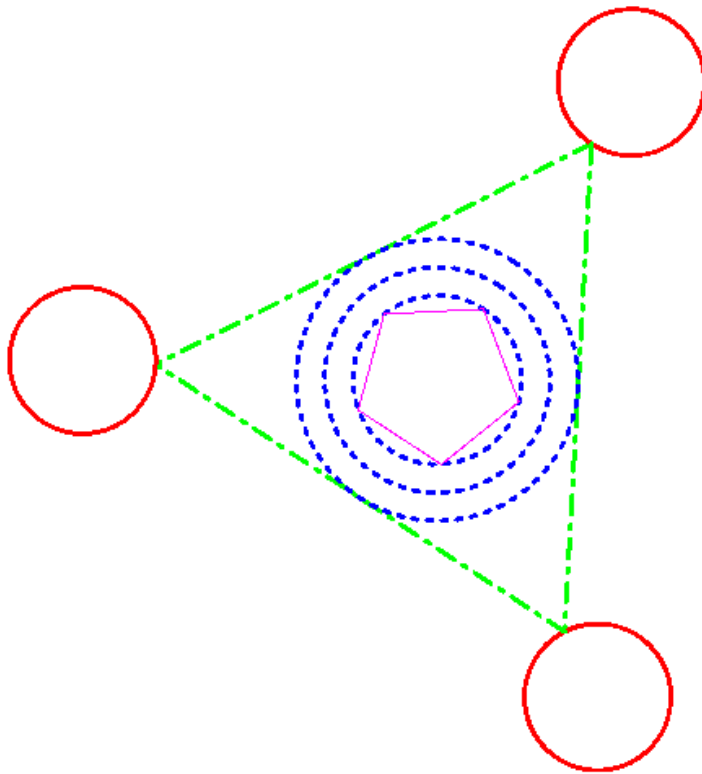
Цель работы:.....	2
Задания к работе:.....	2
Вариант №1:.....	2
Выполнение:.....	4
Снимки экрана:.....	4
Листинги кода:.....	4

Выполнение:

Снимки экрана:



n



Формулы:

Все радиусы внутренних окружностей и длина стороны многоугольника внутри окружностей вычисляются на основе длины стороны треугольника.

Обозначим длину стороны треугольника как «а».

$$\text{Радиус наибольшей внутренней окружности } R = \frac{a\sqrt{3}}{6}$$

Две другие внутренние окружности имеют радиусы: $R*0.8$, $R*0.6$

$$\text{Угол внутреннего многоугольника } \hat{\alpha} = 180 \frac{n-2}{n}$$

$$\text{Сторона внутреннего многоугольника } L = \left| R*0.6*2*\sin \frac{Pi}{n} \right|$$

Сама сторона треугольника «а» определяется величиной «s»

$$s = \frac{\min(\text{height}, \text{width})}{2}$$

$$a = \frac{s}{\cos(60^\circ)}$$

$$\text{Радиус окружностей по краям треугольника } c = s*0.15$$

Эти вычисления позволяют масштабировать изображение.

Листинги кода:

Листинг №1: "main.cpp":

```
#include "mainwindow.h"
#include <QApplication>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    MainWindow w;
    w.show();
    return a.exec();
}
```

Листинг №2: "mainwindow.h":

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
#include <QTimer>
#include <QPainter>

QT_BEGIN_NAMESPACE
namespace Ui { class MainWindow; }
QT_END_NAMESPACE

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    MainWindow(QWidget *parent = nullptr);
    ~MainWindow();

private slots:
    void rotate();

private:
    Ui::MainWindow *ui;
    QTimer *timer;
    float angle;

    void paintEvent(QPaintEvent *event);
};
#endif // MAINWINDOW_H
```

Листинг №3: "mainwindow.cpp"

```
#include "mainwindow.h"
#include "../ui_mainwindow.h"
#include <cmath>

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

    timer = new QTimer;
```

```

    timer->setInterval(50);
    timer->start();
    connect(timer, &QTimer::timeout, this, &MainWindow::rotate);
}

void drawBG(QPainter *painter, int width, int height){
    auto oldBrush = painter->brush();
    auto oldPen = painter->pen();
    QColor white(255, 255, 255);
    painter->setBrush(white);
    painter->setPen(white);
    painter->drawRect(0, 0, width, height);
    painter->setBrush(oldBrush);
    painter->setPen(oldPen);
}

double distance(QPoint const &a, QPoint const &b){
    return std::sqrt(std::pow(a.x() - b.x(), 2) + pow(a.y() - b.y(), 2));
}

void MainWindow::paintEvent(QPaintEvent *event){
    int n = ui->spinBox->value();
    int a = 300;

    float s = std::min(height(), width()) / 2.0;
    a = s / std::cos(M_PI/6.0);

    float outerRadius = s * 0.15;
    float radius = a*std::sqrt(3)/6.0;
    float innerAngle = 180.0f*(n - 2)/n;
    float innerLength = std::abs(radius * 0.6f * 2.0f * std::sin(M_PI / n));

    // Setup
    QPainter painter(this);
    drawBG(&painter, width(), height());
    QPen trianglePen(Qt::green, 3, Qt::DashDotLine, Qt::RoundCap,
Qt::RoundJoin);
    QPen outerCirclePen(Qt::red, 3, Qt::SolidLine, Qt::RoundCap, Qt::RoundJoin);
    QPen innerCirclePen(Qt::blue, 3, Qt::DotLine, Qt::RoundCap, Qt::RoundJoin);
    QPen innerFigurePen(QColor(255, 0, 255));
    painter.setPen(trianglePen);

    // Rotate
    painter.translate(width()/2, height()/2);
    painter.rotate(angle);
    painter.translate(0, radius);

    //Triangles and circles
    auto outerCircleDraw = [&](int sig, int sigR = 1){
        painter.rotate(sig * 30.0);
        painter.setPen(outerCirclePen);
        painter.drawEllipse(QPointF(outerRadius * sigR, 0), outerRadius,
outerRadius);
        painter.rotate(-30.0 * sig);
        painter.setPen(trianglePen);
    };
    painter.drawLine(0 - a/2, 0, 0 + a/2, 0);
    painter.translate(0 - a/2, 0);
    painter.rotate(-60.0);
    outerCircleDraw(1, -1);
}

```

```

painter.drawLine(0, 0, a, 0);
painter.translate(a, 0);
outerCircleDraw(-1);

painter.rotate(120.0);
painter.drawLine(0, 0, a, 0);
painter.translate(a, 0);
outerCircleDraw(-1);
// Inner circles
painter.translate(-a/2, 0);
painter.rotate(90);
painter.translate(radius, 0);
painter.setPen(innerCirclePen);
painter.drawEllipse(QPointF(0, 0), radius, radius);
painter.drawEllipse(QPointF(0, 0), radius*0.8, radius*0.8);
painter.drawEllipse(QPointF(0, 0), radius*0.6, radius*0.6);
// Inner figure
painter.translate(radius*0.6, 0);
painter.rotate(180.0);
painter.rotate(innerAngle/2.0);
painter.setPen(innerFigurePen);
for(int i = 0; i < n; i++){
    painter.drawLine(0, 0, innerLength, 0);
    painter.translate(innerLength, 0);
    painter.rotate(180);
    painter.rotate(innerAngle);
}
}

void MainWindow::rotate() {
    angle += 1.5f;
    this->update();
}

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

```

Вывод:

В ходе выполнения лабораторной работы мы изучили графические 2D-примитивы в среду Qt Creator и научились работать с ними.