**МИНИСТЕРСТВА 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

Дисциплина: Компьютерная графика

Тема: «Графические примитивы GDI»

Выполнила: ст. группы ПВ-31

Немшилова Анна Геннадьевна

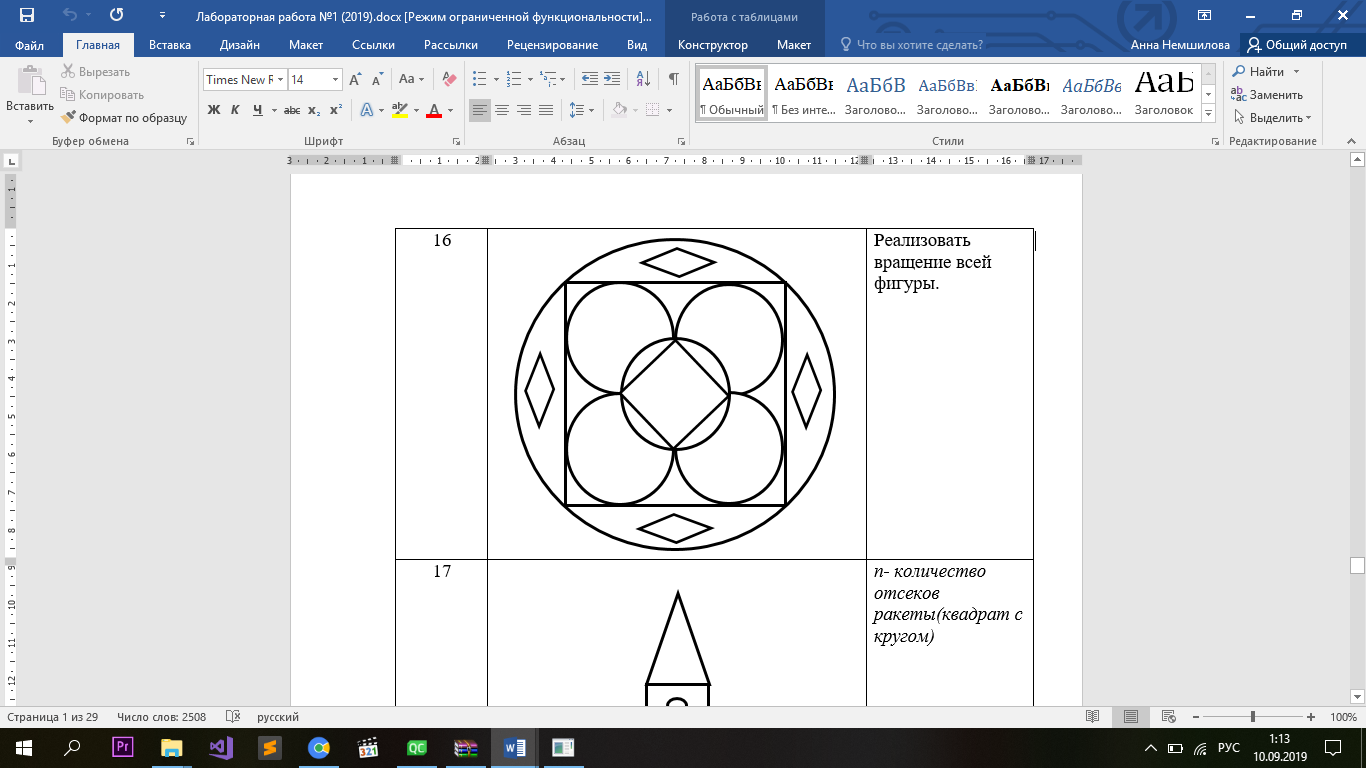
Проверил: Осипов О. В.

Белгород 2019

**Цель работы:** изучение графических 2D-примитивов с использованием GDI в среде Qt Creator.

**Порядок выполнения работы**

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.



**Код программы:**

**main.cpp**

#include "mainwindow.h"

#include <QApplication>

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

{

QApplication a(*argc*, argv);

MainWindow w;

w.show();

return a.exec();

}

**mainwindow.h**

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

#include <QPainter>

#include <QWheelEvent>

namespace **Ui** {

class **MainWindow**;

}

class **MainWindow** : public QMainWindow

{

Q\_OBJECT

public:

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

~***MainWindow***();

private:

Ui::MainWindow \*ui;

float angle;

// Обработчик события перерисовки окна

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

// Обработчик события прокрутки колеса мыши

void ***wheelEvent***(QWheelEvent\* wheelEvent);

};

#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);

}

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

{

delete ui;

}

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

{

QPainter painter(this); // роздаём объект-живописец

// радиус большей окружности

float radius;

// вычисляем радиус окружности

if (width() > height()) // Если ширина окна больше высоты

radius = (height() - 20) / 2; // Отступ от краёв - 10 пикселей

else radius = (width() - 20) / 2;

// если размеры окна маленькие, ничего не выводить

if (width() < 30 || height() < 30)

return;

// координаты центра окружности

QPointF center = QPointF(width() / 2, height() / 2);

// вращение фигуры

painter.translate(center);

painter.rotate(angle);

painter.translate(-center);

// большая окружность

painter.setPen(QPen(Qt::black, 3, Qt::SolidLine, Qt::FlatCap));

QRectF rect = QRectF(center.x() - radius, center.y() - radius, radius \* 2, radius \* 2);

painter.setBrush(Qt::darkCyan);

painter.drawEllipse(rect);

// большой квадрат

float a = radius/(sqrt(2)); // половина стороны квадрата

QPolygonF sq;

painter.setBrush(Qt::blue);

sq.append(QPointF(center.x() - a, center.y() - a));

sq.append(QPointF(center.x() - a, center.y() + a));

sq.append(QPointF(center.x() + a, center.y() + a));

sq.append(QPointF(center.x() + a, center.y() - a));

painter.drawPolygon(sq);

// маленькие круги

float r\_m = a/2; // радиус маленького круга

QPointF center2\_1 = QPointF(center.x() + r\_m, center.y() + r\_m);

rect = QRectF(center2\_1.x() - r\_m, center2\_1.y() - r\_m, r\_m \* 2, r\_m \* 2);

painter.setBrush(Qt::darkMagenta);

painter.drawEllipse(rect);

center2\_1 = QPointF(center.x() + r\_m, center.y() - r\_m);

rect = QRectF(center2\_1.x() - r\_m, center2\_1.y() - r\_m, r\_m \* 2, r\_m \* 2);

painter.drawEllipse(rect);

center2\_1 = QPointF(center.x() - r\_m, center.y() - r\_m);

rect = QRectF(center2\_1.x() - r\_m, center2\_1.y() - r\_m, r\_m \* 2, r\_m \* 2);

painter.drawEllipse(rect);

center2\_1 = QPointF(center.x() - r\_m, center.y() + r\_m);

rect = QRectF(center2\_1.x() - r\_m, center2\_1.y() - r\_m, r\_m \* 2, r\_m \* 2);

painter.drawEllipse(rect);

// центральный круг

float radius2 = radius \* 0.4f;

QRectF rect2 = QRectF(center.x() - radius2, center.y() - radius2, radius2 \* 2, radius2 \* 2);

painter.setBrush(Qt::magenta);

painter.drawEllipse(rect2);

// центральный ромб

QPolygonF romb;

painter.setBrush(Qt::darkRed);

romb.append(QPointF(center.x() - radius2 - 1, center.y()));

romb.append(QPointF(center.x(), center.y() + radius2 - 1));

romb.append(QPointF(center.x() + radius2 - 1, center.y()));

romb.append(QPointF(center.x(), center.y() - radius2 - 1));

painter.drawPolygon(romb);

// маленькие ромбы по краям

QPolygonF rombs[4];

painter.setBrush(Qt::cyan);

float c\_r = a + (radius - a)/2;

QPointF centers[4] = {QPointF(center.x(), center.y() + c\_r), QPointF(center.x(), center.y() - c\_r), QPointF(center.x() + c\_r, center.y()), QPointF(center.x() - c\_r, center.y())};

for (int i = 0; i < 4; i++) {

if ((i == 0) || (i == 1)) {

rombs[i].append(QPointF(centers[i].x(), centers[i].y() + 15));

rombs[i].append(QPointF(centers[i].x() + 25, centers[i].y()));

rombs[i].append(QPointF(centers[i].x(), centers[i].y() - 15));

rombs[i].append(QPointF(centers[i].x() - 25, centers[i].y()));

}

else {

rombs[i].append(QPointF(centers[i].x(), centers[i].y() + 25));

rombs[i].append(QPointF(centers[i].x() + 15, centers[i].y()));

rombs[i].append(QPointF(centers[i].x(), centers[i].y() - 25));

rombs[i].append(QPointF(centers[i].x() - 15, centers[i].y()));

}

painter.drawPolygon(rombs[i]);

}

}

// Обработчик события прокрутки колеса мыши

void MainWindow::***wheelEvent***(QWheelEvent\* wheelevent)

{

angle -= wheelevent->delta() / 10;

repaint(); // обновляем окно

}

**Результат работы программы:**

