**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ**

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ **«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ ИМ. В. Г. ШУХОВА»**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

**ЛАБОРАТОРНАЯ РАБОТА №1**

**Дисциплина: Компьютерная графика**

**Тема: Графические примитивы GDI**

Выполнил: ст. группы ВТ-31

Подкопаев Антон Валерьевич

Проверил: Осипов Олег Васильевич

Белгород 2019

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

**Порядок выполнения работы**

⦁ Изучить графические примитивы библиотеки Qt (<http://doc.qt.io/qt-4.8/QPainter.html>)

⦁ Разработать алгоритм и составить программу для построения на экране изображения в соответствии с номером варианта.

**Требования к программе**

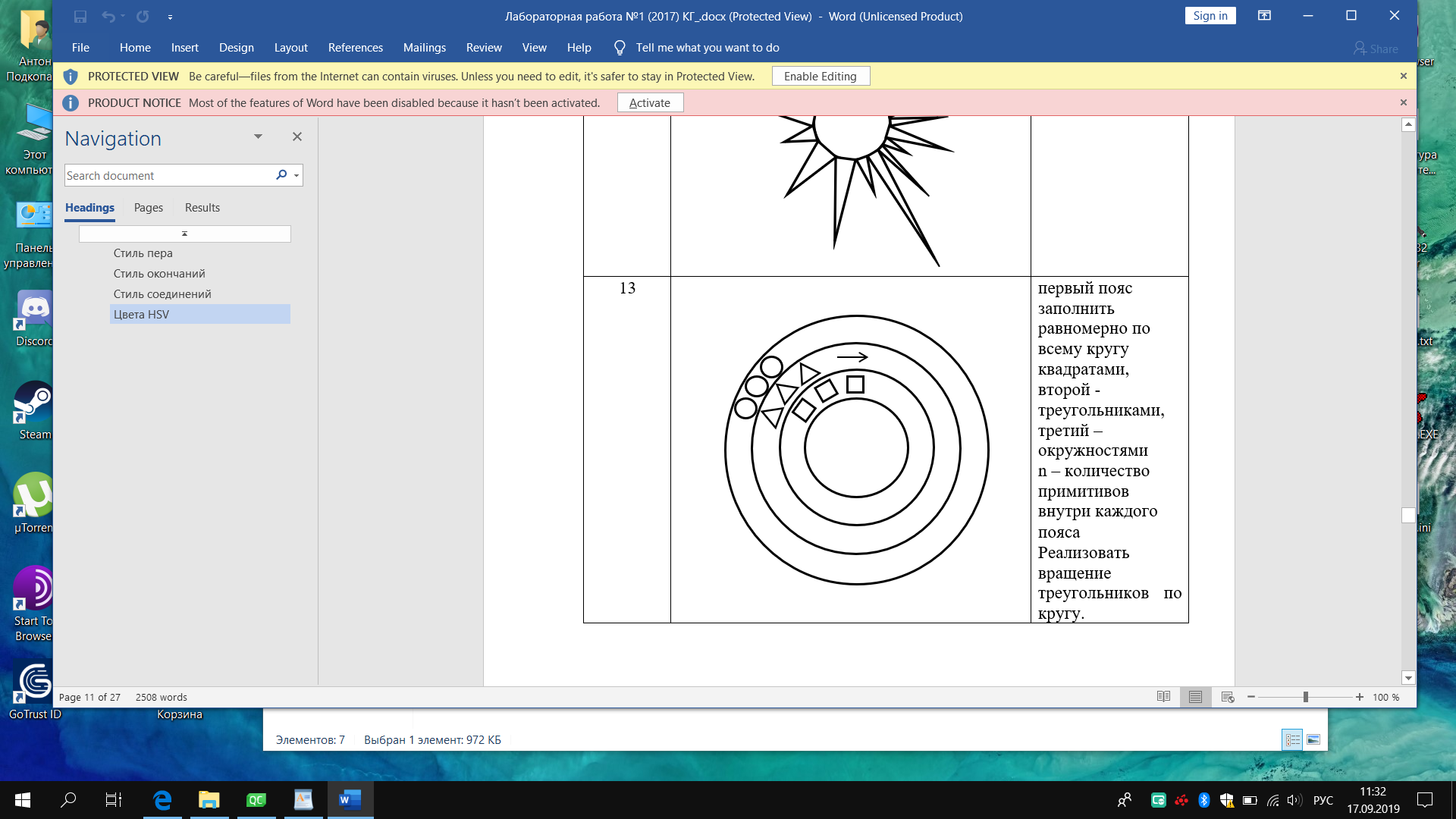
⦁ В программе должна быть предусмотрена возможность ввода пользователем исходных данных (из правой колонки таблицы №1).

⦁ Изображение должно масштабироваться строго по центру с отступом 10 пикселей от границ и реагировать на изменение размера окна

(см. пример проектов lab\_1\_qt\_cpp, lab\_1\_qt\_cpp\_2, lab\_1\_vcpp).

⦁ Раскрасить (залить) примитивы (круги, многоугольники и др.) по собственному усмотрению.

**Вариант 13**



*Текст программы*

Мainwindow.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;

int angle;

// Обработчик события перерисовки окна

void paintEvent(QPaintEvent\* event);

// Обработчик события прокрутки колеса мыши

void wheelEvent(QWheelEvent\* );

};

#endif // MAINWINDOW\_H

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.cpp

#include "mainwindow.h"

#include "ui\_mainwindow.h"

#include "math.h"

#define M\_PI 3.14159265358979323846

// Возвращает точку point повернутую вокруг центра center на dgr градусов по чвсовой стрелке

QPointF rotatePoint(QPointF center, QPointF point, double dgr){

QPointF p;

double px = point.x() - center.x();

double py = point.y() - center.y();

double sn = sin((dgr \* M\_PI)/180),

cs = cos((dgr \* M\_PI)/180);

p.setX(center.x() + px \* cs - py \* sn);

p.setY(center.y() + py \* cs + px \* sn);

return p;

}

MainWindow::MainWindow(QWidget \*parent) :

QMainWindow(parent),

ui(new Ui::MainWindow)

{

ui->setupUi(this);

}

MainWindow::~MainWindow()

{

delete ui;

}

void MainWindow::paintEvent(QPaintEvent\* event)

{

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

QColor White(255, 255, 255); // Белый цвет

QColor Black(0, 0, 0); // Черный цвет

// Координаты центра окна

QPointF center = QPointF(width()\*0.5, height()\*0.5);

// Задаем основной радиус

double radius;

// Вычисляем радиус окружности

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

radius = (height() - 20) / 2;

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

// Если размеры окна маленькие, ничего не выводить

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

return;

// Задаем массив радиусов

double rad[4] = {radius, radius - radius\*0.25, radius - radius\*0.5, radius - radius\*0.75};

// Задаём область прорисовки окружности

QRectF rectForEllipse;

// Рисуем направляющие окружности

painter.setBrush(White);

painter.setPen(Black);

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

rectForEllipse = QRectF(center.x() - rad[i], center.y() - rad[i], rad[i] \* 2, rad[i] \* 2);

painter.drawEllipse(rectForEllipse);

}

int red, green, blue;

// Количество квадратов

int sqr = 5;

// Задаем параметры для квадратов

QRectF Square = QRectF(center.x() - rad[3]\*0.375, center.y() - rad[3]\*1.85, rad[3]\*0.75, rad[3]\*0.75);

QRectF SquareR = QRectF(-rad[3]\*0.375, -rad[3]\*0.375, rad[3]\*0.75, rad[3]\*0.75);

// Рисуем квадраты

green = 0;

double dgrS = 0;

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

QColor Green(0, green, 0);

painter.setBrush(Green);

painter.setPen(Green);

painter.save();

painter.translate(rotatePoint(center, Square.center(), dgrS));

painter.rotate(dgrS);

painter.drawRect(SquareR);

painter.restore();

dgrS += 45;

green += 45;

}

// Количество треугольников

int trg = 9;

// Задаем параметры треугольников

QRectF RectForT = QRectF(center.x() - rad[3]\*0.375, center.y() - rad[3]\*2.85, rad[3]\*0.75, rad[3]\*0.75);

QRectF RectForTR = QRectF(-rad[3]\*0.375, -rad[3]\*0.375, rad[3]\*0.75, rad[3]\*0.75);

QPolygonF Triangle;

Triangle << QPointF(RectForTR.left(), RectForTR.bottom())

<< QPointF(RectForTR.center().x(), RectForTR.top())

<< QPointF(RectForTR.right(), RectForTR.bottom());

// Рисуем треугольники

red;

double dgrT = angle;

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

QColor Red(red, 0, 0);

painter.setBrush(Red);

painter.setPen(Red);

painter.save();

painter.translate(rotatePoint(center, RectForT.center(), dgrT));

painter.rotate(dgrT);

painter.drawPolygon(Triangle);

painter.restore();

dgrT += 30;

red += 15;

}

// Количество окружностей

int rnd = 25;

// Задаем параметры для окружностей

QRectF Round = QRectF(center.x() - rad[3]\*0.375, center.y() - rad[3]\*3.85, rad[3]\*0.75, rad[3]\*0.75);

QRectF RoundR = QRectF(-rad[3]\*0.375, -rad[3]\*0.375, rad[3]\*0.75, rad[3]\*0.75);

// Рисуем окружности

/\*red = 0;

blue = 255;\*/

double clrDgr = 0;

double dgrR = 0;

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

QColor Color = QColor::fromHsv(clrDgr, 255 , 255);

Color.toHsv();

painter.setBrush(Color);

painter.setPen(Color);

painter.save();

painter.translate(rotatePoint(center, Round.center(), dgrR));

painter.rotate(dgrR);

painter.drawEllipse(RoundR);

painter.restore();

clrDgr += 14.5;

dgrR += 15;

/\*if(blue > 15)

blue -= 10;

if(red < 245)

red += 10;\*/

}

}

// Обработчик события прокрутки колеса мыши

void MainWindow::wheelEvent(QWheelEvent\* wheelevent)

{

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

repaint(); // Обновляем окно

}

**Результат работы программы**

