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-211

Чувилко Илья Романович

Проверил:

Осипов Олег Васильевич

Белгород 2023 г.

Вариант 20

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.

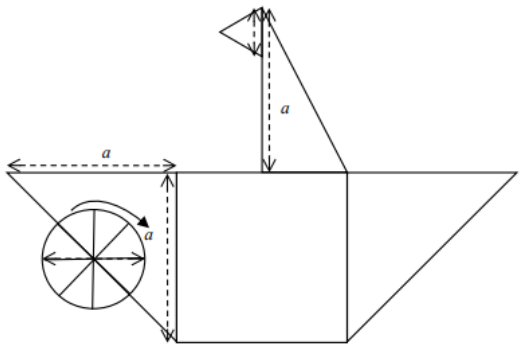
Требования к программе

1. В программе должна быть предусмотрена возможность ввода пользователем исходных данных (из правой колонки таблицы №1).
2. 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).
3. Раскрасить (залить) примитивы (круги, многоугольники и др.) по собственному усмотрению.

Содержание отчёта

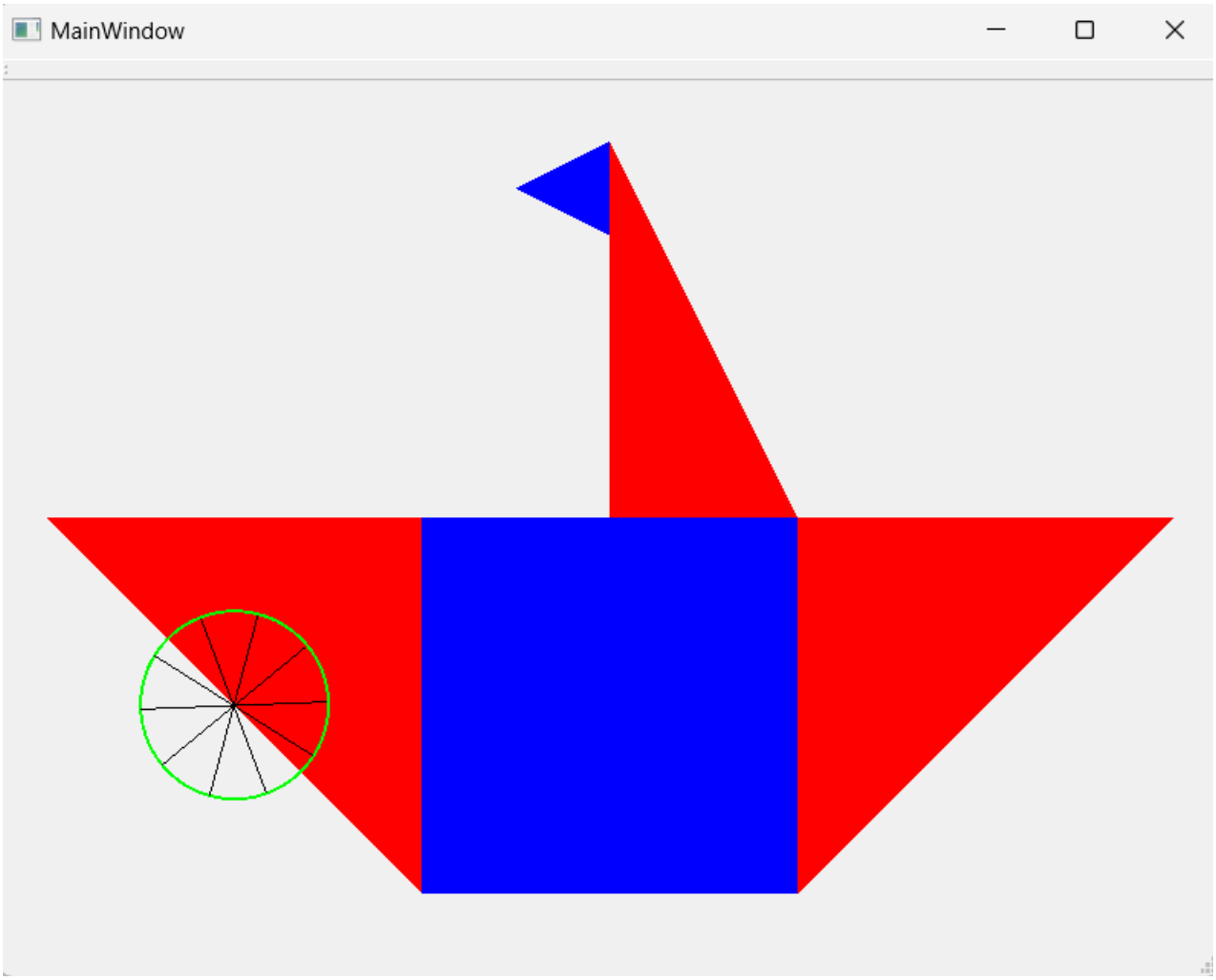
1. Название темы.
2. Цель работы.
3. Постановка задачи.
4. Вывод необходимых геометрических формул для построения изображения.
5. Текст программы.
6. Результат работы программы (снимки экрана).

Задание:

	a – сторона треугольника, n – количество секторов в окружности. a и n вводятся с клавиатуры. Реализовать вращение окружности по часовой стрелке. Раскрасить каждый элемент сектора своим цветом.

Выполнение:

Снимки экрана:



Листинги кода:

Содержимое файла «main.cpp»

```
#include "mainwindow.h"

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

Содержимое файла «mainwindow.h»

```
#ifndef MAINWINDOW_H

#define MAINWINDOW_H
#include <QMainWindow>
#include <QTimer>
#include <QPainter>
#include <QSpinBox>
#include <QDebug>

QT_BEGIN_NAMESPACE
namespace Ui { class MainWindow; }
QT_END_NAMESPACE
class MainWindow : public QMainWindow
{
    Q_OBJECT
public:
    MainWindow(QWidget *parent = nullptr);
    ~MainWindow();
    void NValueChanged(int value);
    void AValueChanged(int value);
private slots:
    void rotate();
private:
    Ui::MainWindow *ui;
    QTimer *timer;
    float angle;
    int n = 10;
    void paintEvent(QPaintEvent *event);
};
#endif // MAINWINDOW_H
```

Содержимое файла «mainwindow.cpp»

```
#include "mainwindow.h"

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

MainWindow::MainWindow(QWidget *parent)
    : QMainWindow(parent)
    , ui(new Ui::MainWindow)
{
    ui->setupUi(this);
    resize(800, 600);
    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){
    float minSize = std::min(height(), width()) / 2.0; // Центр окна
    int a = std::min(width() / 6, height()/4) - 10;
    QPainter painter(this);

    // Создаем кисть для заливки
    QBrush blueBrush(Qt::blue);
    QBrush redBrush(Qt::red);

    // Rotate
    painter.translate(width()/2, height()/2);

    QPainterPath path;
    path.moveTo(a, 0);
    path.lineTo(-a, 0);
    path.lineTo(-a, 2*a);
    path.lineTo(a, 2*a);
    painter.fillPath(path, blueBrush);

    path = QPainterPath();
    path.moveTo(a, 0);
    path.lineTo(3*a, 0);
    path.lineTo(a, 2*a);
    painter.fillPath(path, redBrush);

    path = QPainterPath();
    path.moveTo(-a, 0);
    path.lineTo(-3*a, 0);
    path.lineTo(-a, 2*a);
    painter.fillPath(path, redBrush);

    path = QPainterPath();
    path.moveTo(0, 0);
    path.lineTo(0, -2*a);
    path.lineTo(a, 0);
    painter.fillPath(path, redBrush);

    path = QPainterPath();
    path.moveTo(0, -1.5*a);
    path.lineTo(0, -2*a);
    path.lineTo(-0.5*a, -1.75*a);
    painter.fillPath(path, blueBrush);

    QPoint wheelCenter(-2*a, a);
    painter.translate(wheelCenter);
    painter.rotate(angle);
    painter.setPen(QPen(Qt::green, 2));
    painter.drawEllipse(-a/2, -a/2, a, a);

    // Рисуем спицы
    painter.setPen(QPen(Qt::black, 1));

    for (int i = 0; i < n; ++i) {
        int deg = i * 360 / n;
        double radian = deg * M_PI / 180.0f;
```

```
        int x = a/2 * std::cos(radian);
        int y = a/2 * std::sin(radian);
        painter.drawLine(QPoint(0, 0), QPoint(x, y));
    }
}

void MainWindow::rotate() {
    angle += 5.0f;
    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