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 ВТ-32

Воскобойников И. С.

Проверил: Гибкин Ю. С.

Белгород 2020 г.

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.

|  |  |
| --- | --- |
| *a* | *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. |

Код программы:

#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 slots:

void on\_verticalSlider\_valueChanged(int value);

private:

Ui::MainWindow \*ui;

int angle;

int n = 0;

// Обработчик события перерисовки окна

void *paintEvent*(QPaintEvent\* event);

// Обработчик события прокрутки колеса мыши

void *wheelEvent*(QWheelEvent\* );

// Отрисовка окружности с n-угольником внутри

void drawCircleWithNAngle(QPainter \*painter, QRectF rect, QBrush brush, QPointF c, int r=0, int n=0);

};

#endif // MAINWINDOW\_H

#include "mainwindow.h"

#include "ui\_mainwindow.h"

#include <QPolygon>

#include <QtMath>

MainWindow::MainWindow(QWidget \*parent) :

QMainWindow(parent),

ui(new Ui::MainWindow)

{

ui->setupUi(this);

}

MainWindow::~*MainWindow*()

{

delete ui;

}

QPoint getPointOnCircleByParams(QPointF c, int r, int i, int n)

{

return QPoint(c.x() + r \* cos(2 \* M\_PI \* i / n),

c.y() + r \* sin(2 \* M\_PI \* i / n));

}

void MainWindow::drawCircleWithNAngle(QPainter \*painter, QRectF rect, QBrush brush, QPointF c, int r, int n)

{

painter->setBrush(brush);

painter->drawEllipse(rect);

if (n < 3) return;

else {

for (int point\_i=0; point\_i < n; point\_i++){

QPoint p0 = QPoint(c.x(), c.y());

QPoint p1 = getPointOnCircleByParams(c, r, point\_i, n);

QPoint p2 = getPointOnCircleByParams(c, r, point\_i + 1, n);

QPolygon polygon({p0, p1, p2});

painter->setBrush(QBrush(Qt::blue));

painter->drawPolygon(polygon);

}

}

}

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

{

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

QColor orangeRed(240, 100, 0); // Оранжево-красный цвет

// Координаты центра окружности

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

int a = 10;

if (width() > height()) a = height() / 3;

else a = width() / 3;

if (a < 50) return;

painter.translate(center);

painter.rotate(angle);

painter.translate(-center);

int r = (float)a / (2 \* sqrt(3));

QPoint topTrianPoint = QPoint(center.x(), center.y() + 2\*r);

QPoint leftTrianPoint = QPoint(center.x() - a/2, center.y() - r);

QPoint rightTrianPoint = QPoint(center.x() + a/2, center.y() - r);

// Задаём область прорисовки Треугольника

QPolygon polygon({topTrianPoint, leftTrianPoint, rightTrianPoint});

painter.setBrush(QBrush(orangeRed));

painter.drawPolygon(polygon);

int out\_r = a / 4;

int toSide = cos(M\_PI/6)\*out\_r;

int toBottom = toSide \* tan(M\_PI/6);

QBrush green = QBrush(Qt::green);

QPoint new\_center = QPoint(topTrianPoint.x(), topTrianPoint.y() + out\_r);

QPoint centers[3] = {

QPoint(topTrianPoint.x(), topTrianPoint.y() + out\_r),

QPoint(leftTrianPoint.x() - toSide, leftTrianPoint.y() - toBottom),

QPoint(rightTrianPoint.x() + toSide, rightTrianPoint.y() - toBottom)

};

for (int i = 0; i < 3; i++) {

QRectF rect = QRectF(centers[i].x() - out\_r, centers[i].y() - out\_r, 2 \* out\_r, 2\* out\_r);

drawCircleWithNAngle(&painter, rect, green, centers[i], out\_r, n);

}

// Рисуем три круга меньшего диаметра и разными цветами

float radiuses[3] = { r, r\*0.75f, r\*0.5f };

// Инициализируем три кисти разного цвета

QBrush brushes[3] = { QBrush(Qt::black), QBrush(orangeRed), QBrush(Qt::black) };

for (int i = 0; i < 3; i++) {

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

drawCircleWithNAngle(&painter, rect, brushes[i], center, radiuses[i], n);

}

}

// Обработчик события прокрутки колеса мыши

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

{

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

repaint(); // Обновляем окно

}

void MainWindow::on\_verticalSlider\_valueChanged(int value)

{

n = value;

repaint(); // Обновляем окно

}

