

МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра САПР

ОТЧЕТ
по лабораторной работе №1
по дисциплине «Компьютерная графика»
Тема: Исследование математических методов представления и
преобразования графических объектов на плоскости и в пространстве

Студенты гр. 8362

Преподаватель

Ларионова Е.Е.

Матвеев Н.Д.

Матвеева И. В.

Санкт-Петербург

2021

ЗАДАНИЕ

Поворот плоского объекта относительно произвольной точки плоскости на заданный угол. Необходимо предусмотреть возможность редактирования положения точки

ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

- Рассмотрим треугольник ABC (рис. 4) и с помощью преобразования повернем его на 90° против часовой стрелки относительно начала координат

$$[T] = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

- Если использовать матрицу, состоящую из координат x и y вершин треугольника, то можно записать

$$\begin{bmatrix} 3 & -1 \\ 4 & 1 \\ 2 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 3 \\ -1 & 4 \\ -1 & 2 \end{bmatrix}$$

- координаты результирующего треугольника A*B*C*.

Поворот относительно начала координат

- на 180° $[T] = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$
- на 270° $[T] = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$
- на 0° или 360° $[T] = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

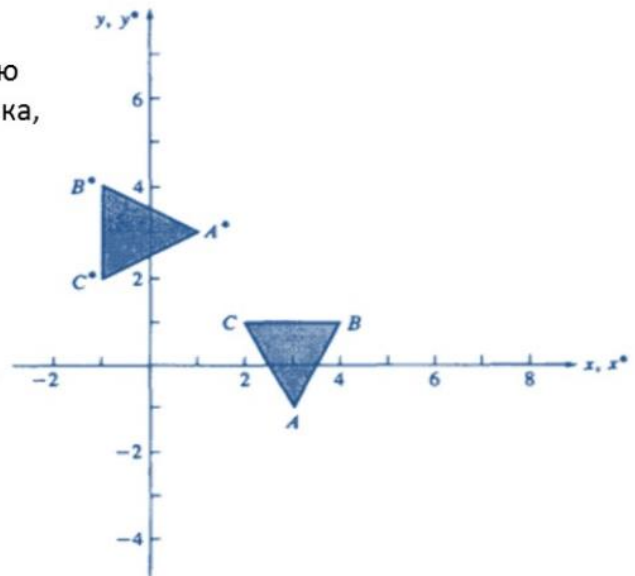


Рис. 4

- В этих примерах осуществляется преобразование в специальных случаях поворота вокруг начала координат на углы 0° , 90° , 180° и 270° .
- Поворот вокруг точки начала координат на произвольный угол θ .
- рассмотрим вектор положения от начала координат до точки P (рис. 5). Обозначим r — длину вектора, а ϕ — угол между вектором и осью x.
- Вектор положения поворачивается вокруг начала координат на угол θ и попадает в точку P*. Записав векторы положений для P и P*, получаем:
- $P = \begin{bmatrix} x & y \end{bmatrix} = \begin{bmatrix} r \cos \phi & r \sin \phi \end{bmatrix}$ $P^* = \begin{bmatrix} x^* & y^* \end{bmatrix} = \begin{bmatrix} r \cos(\phi + \theta) & r \sin(\phi + \theta) \end{bmatrix}$
- Используя формулу для cos суммы углов, перепишем выражение для P* следующим образом
- $P^* = \begin{bmatrix} x^* & y^* \end{bmatrix} = \begin{bmatrix} r(\cos \phi \cos \theta - \sin \phi \sin \theta) & r(\cos \phi \sin \theta + \sin \phi \cos \theta) \end{bmatrix}$
- Используя определения x и y, можно переписать P* как
- $P^* = \begin{bmatrix} x^* & y^* \end{bmatrix} = \begin{bmatrix} x \cos \theta - y \sin \theta & x \sin \theta + y \cos \theta \end{bmatrix}$
- Таким образом, преобразованная точка имеет координаты
- $x^* = x \cos \theta - y \sin \theta$ $y^* = x \sin \theta + y \cos \theta$
- или в матричном виде** $\begin{bmatrix} x^* \\ y^* \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} [T] = \begin{bmatrix} x^* & y^* \end{bmatrix} = \begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{bmatrix}$
- преобразование поворота вокруг точки начала координат на произвольный угол θ задается матрицей

$$[T] = \begin{bmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{bmatrix}$$

Повороты являются положительными, если они осуществляются против часовой стрелки относительно точки вращения (рис. 5).

- Определитель общей матрицы поворота имеет следующий вид:
- $\det[T] = \cos^2\theta + \sin^2\theta = 1$.
- В общем случае преобразования по матрице с детерминантом, равным 1, приводят к полному повороту.
- Пусть, надо вернуть точку P^* в P , т. е. выполнить обратное преобразование. Очевидно, что требуемый угол поворота равен $-\theta$. Получим матрицу для обратного преобразования

$$[T]^{-1} = \begin{bmatrix} \cos(-\theta) & \sin(-\theta) \\ -\sin(-\theta) & \cos(-\theta) \end{bmatrix} = \begin{bmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{bmatrix}$$

- Можно показать, что матрица $[T]^{-1}$ является обратной к $[T]$, результат умножения матрицы на обратную дает единичную матрицу.

- Обратная матрица вращения является транспонированной

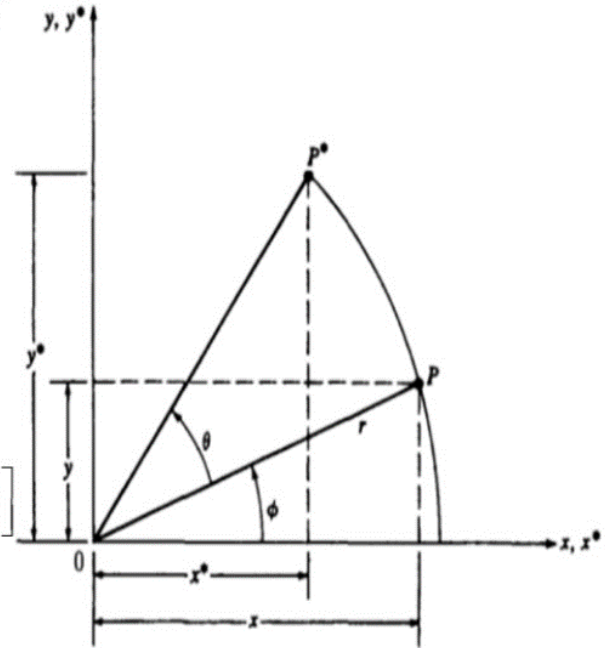
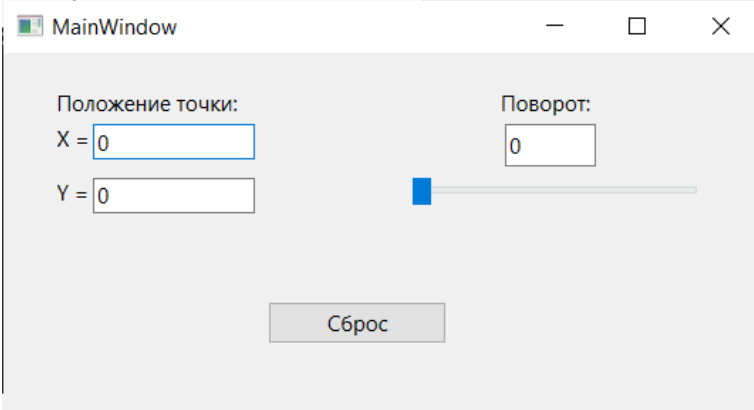


Рис . 5

РЕАЛИЗАЦИЯ ПРОГРАММЫ

Программа реализована на языке программирования C++ с помощью библиотек Qt. Программа работает в графическом режиме.

В начале работы программа открывает два окна «MainWindow» и «Form». Окно «MainWindow» запрашивает координаты точки и угол поворота квадрата относительно заданной точки (Рисунок 1). Окно «Form» служит для отрисовки квадрата, согласно заданным координатам в предыдущем окне (Рисунок 2).



The screenshot shows a window titled "MainWindow". It contains two input sections. The first section, labeled "Положение точки:" (Point Position), has two text boxes: "X =" with the value "0" and "Y =" with the value "0". The second section, labeled "Поворот:" (Rotation), has a text box with the value "0" and a horizontal slider below it. A blue square marker is positioned on the slider. At the bottom center of the window is a button labeled "Сброс" (Reset).

Рисунок 1 – Запрос координат точки и угла поворота

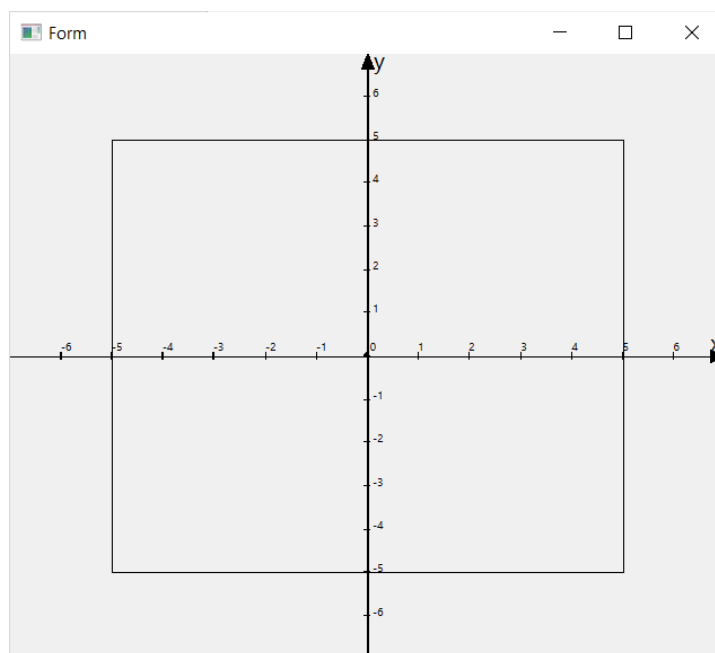


Рисунок 2 – Окно «Form»

После ввода координат и угла в окне «Form» появляется точка, с заданными параметрами. Также при изменении угла поворота, квадрат меняет свое положение (Рисунок 3 и Рисунок 4).

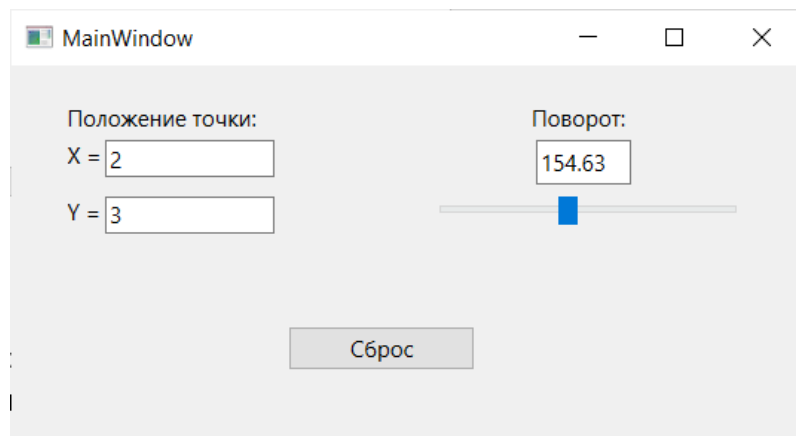


Рисунок 3 – Изменение координат точки и угла поворота

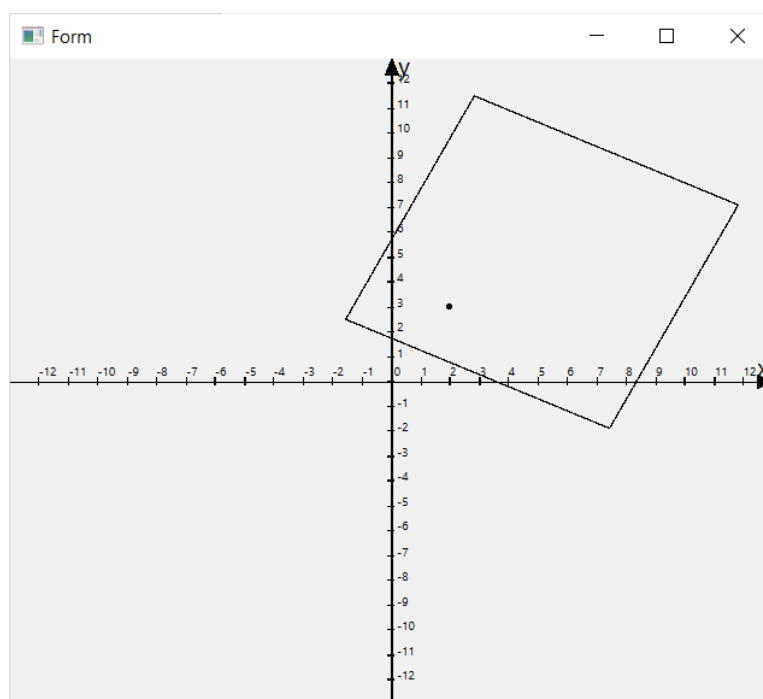


Рисунок 4 – Отображение повернутого квадрата

ПРИЛОЖЕНИЕ 1. КОД ПРОГРАММЫ

Файл Main.cpp

```
#include <application.h>

int main(int argc, char *argv[])
{
    Application a(argc, argv);
    return a.exec();
}
```

Файл MainWindow.cpp

```
#include "mainwindow.h"
#include "ui_mainwindow.h"

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

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

void MainWindow::res()
{
    on_pushButton_reset_clicked();
}
```

```
}
```

```
Control_State MainWindow::get_state()
```

```
{
```

```
    Control_State tmp;
```

```
    tmp.is_reset = false;
```

```
    tmp.point_changed = false;
```

```
    tmp.p_x = ui->lineEdit_x->text().toDouble();
```

```
    tmp.p_y = ui->lineEdit_y->text().toDouble();
```

```
    tmp.angle = ui->horizontalSlider->value()/100;
```

```
    return tmp;
```

```
}
```

```
void MainWindow::on_pushButton_reset_clicked()
```

```
{
```

```
    ui->lineEdit_x->setText(QString::number(0));
```

```
    ui->lineEdit_y->setText(QString::number(0));
```

```
    ui->lineEdit_angle->setText(QString::number(0));
```

```
    Control_State tmp = get_state();
```

```
    tmp.is_reset = true;
```

```
    emit update(tmp);
```

```
}
```

```
void MainWindow::on_lineEdit_x_textChanged(const
```

```
QString &arg1)
```

```
{
```

```
    ui->lineEdit_angle->setText(QString::number(0));
```

```
    Control_State tmp = get_state();
```

```
    tmp.point_changed = true;
```



```

        emit update(tmp);
    }

void MainWindow::on_lineEdit_y_textChanged(const
QString &arg1)
{
    ui->lineEdit_angle->setText(QString::number(0));
    Control_State tmp = get_state();
    tmp.point_changed = true;
    emit update(tmp);}

void MainWindow::on_horizontalSlider_valueChanged(int
value)
{
    ui->lineEdit_angle-
>setText(QString::number(value*1.0/100));
    emit update(get_state());
}

void MainWindow::on_lineEdit_angle_textEdited(const
QString &arg1)
{
    ui->horizontalSlider-
>setValue(arg1.toDouble()*100);
}

```

Файл Drawwindow.cpp

```

#include "drawwindow.h"
#include "ui_drawwindow.h"

```

```

DrawWindow::DrawWindow(QWidget *parent) :
    QWidget(parent),
    ui(new Ui::DrawWindow)
{
    ui->setupUi(this);
    this->setAttribute(Qt::WA_DeleteOnClose);
}

DrawWindow::~DrawWindow()
{
    delete ui;
}

void DrawWindow::receive_state(State new_state)
{
    draw_state = new_state;
    repaint();
}

void DrawWindow::paintEvent (QPaintEvent *event)
{
    Q_UNUSED(event);
    QPainter painter(this);
    QFont font;

    qreal cw = 0.5*rect().width();
    qreal ch = 0.5*rect().height();
    qreal cr = 0.015*(cw>ch?ch:cw);
    qreal ca = 0.05*(cw>ch?ch:cw);
    qreal caa = ca * 0.4;

```

```

qreal cf = 0.06 *(cw>ch?ch:cw);
int num = draw_state.max + 2;
qreal cnw = cw / num;
qreal cnh = ch / num;
qreal can = ca *0.2;
qreal cnf = cf * 0.5;

font.setPointSize(cf);
painter.setFont(font);

painter.setPen(QPen(Qt::black));
painter.drawLine(QLineF(0,ch,2*cw,ch));
painter.drawLine(QLineF(cw,0,cw,2*ch));
QPointF arr1[3],arr2[3];
arr1[0] = QPointF(cw,0);
arr1[1] = QPointF(cw-caa,ca);
arr1[2] = QPointF(cw+caa,ca);
arr2[0] = QPointF(2*cw,ch);
arr2[1] = QPointF(2*cw-ca,ch+caa);
arr2[2] = QPointF(2*cw-ca,ch-caa);
painter.setBrush(QBrush(Qt::black));
painter.drawPolygon(arr1,3);
painter.drawPolygon(arr2,3);
painter.drawText(arr1[2],"y");
painter.drawText(arr2[2],"x");

font.setPointSize(cnf);
painter.setFont(font);
for (int i = 0; i < num ; i++)

```

```

    {
        painter.drawLine(cw+i*cnw,ch+can,cw+i*cnw,ch-
can);
        painter.drawLine(cw-i*cnw,ch+can,cw-i*cnw,ch-
can);
        painter.drawLine(cw-
can,ch+i*cnh,cw+can,ch+i*cnh);
        painter.drawLine(cw-can,ch-i*cnh,cw+can,ch-
i*cnh);
        if (i == 0)
        {
            painter.drawText(cw+i*cnw+can,ch-
2*can,QString::number(i));
        }
        else
        {
            painter.drawText(cw+i*cnw,ch-
2*can,QString::number(i));
            painter.drawText(cw-i*cnw,ch-
2*can,QString::number(-i));

painter.drawText(cw+2*can,ch+i*cnh,QString::number(-
i));

            painter.drawText(cw+2*can,ch-
i*cnh,QString::number(i));
        }
    }

```

```

        painter.drawEllipse(cw + draw_state.p.x()*cnw -
0.5*cr,ch - draw_state.p.y()*cnh - 0.5*cr,cr,cr);
        painter.setBrush(QBrush(Qt::transparent));
        QPointF tmp[4];
        for (int i = 0; i<4; i++)
        {
            tmp[i].setX(cw + draw_state.f[i].x()*cnw);
            tmp[i].setY(ch - draw_state.f[i].y()*cnh);
        }
        painter.drawPolygon(tmp,4);

    }

```

Файл Application.cpp

```

#include "drawwindow.h"
#include "ui_drawwindow.h"

DrawWindow::DrawWindow(QWidget *parent) :
    QWidget(parent),
    ui(new Ui::DrawWindow)
{
    ui->setupUi(this);
    this->setAttribute(Qt::WA_DeleteOnClose);
}

DrawWindow::~DrawWindow()
{
    delete ui;
}

```

```

void DrawWindow::recv_state(State new_state)
{
    draw_state = new_state;
    repaint();
}

void DrawWindow::paintEvent (QPaintEvent *event)
{
    Q_UNUSED(event);
    QPainter painter(this);
    QFont font;

    qreal cw = 0.5*rect().width();
    qreal ch = 0.5*rect().height();
    qreal cr = 0.015*(cw>ch?ch:cw);
    qreal ca = 0.05*(cw>ch?ch:cw);
    qreal caa = ca * 0.4;
    qreal cf = 0.06 *(cw>ch?ch:cw);
    int num = draw_state.max + 2;
    qreal cnw = cw / num;
    qreal cnh = ch / num;
    qreal can = ca *0.2;
    qreal cnf = cf * 0.5;

    font.setPointSize(cf);
    painter.setFont(font);

    painter.setPen(QPen(Qt::black));
    painter.drawLine(QLineF(0,ch,2*cw,ch));
    painter.drawLine(QLineF(cw,0,cw,2*ch));
}

```

```

QPointF arr1[3],arr2[3];
arr1[0] = QPointF(cw,0);
arr1[1] = QPointF(cw-caa,ca);
arr1[2] = QPointF(cw+caa,ca);
arr2[0] = QPointF(2*cw,ch);
arr2[1] = QPointF(2*cw-ca,ch+caa);
arr2[2] = QPointF(2*cw-ca,ch-caa);
painter.setBrush(QBrush(Qt::black));
painter.drawPolygon(arr1,3);
painter.drawPolygon(arr2,3);
painter.drawText(arr1[2],"y");
painter.drawText(arr2[2],"x");


font.setPointSize(cnf);
painter.setFont(font);
for (int i = 0; i < num ; i++)
{
    painter.drawLine(cw+i*cnw,ch+can,cw+i*cnw,ch-
can);
    painter.drawLine(cw-i*cnw,ch+can,cw-i*cnw,ch-
can);
    painter.drawLine(cw-
can,ch+i*cnh,cw+can,ch+i*cnh);
    painter.drawLine(cw-can,ch-i*cnh,cw+can,ch-
i*cnh);
    if (i == 0)
    {
        painter.drawText(cw+i*cnw+can,ch-
2*can,QString::number(i));

```

```

        }
        else
        {
            painter.drawText(cw+i*cnw,ch-
2*can,QString::number(i));
            painter.drawText(cw-i*cnw,ch-
2*can,QString::number(-i));

painter.drawText(cw+2*can,ch+i*cnh,QString::number(-
i));

            painter.drawText(cw+2*can,ch-
i*cnh,QString::number(i));
        }
    }

    painter.drawEllipse(cw + draw_state.p.x()*cnw -
0.5*cr,ch - draw_state.p.y()*cnh - 0.5*cr,cr,cr);
    painter.setBrush(QBrush(Qt::transparent));
    QPointF tmp[4];
    for (int i = 0; i<4; i++)
    {
        tmp[i].setX(cw + draw_state.f[i].x()*cnw);
        tmp[i].setY(ch - draw_state.f[i].y()*cnh);
    }
    painter.drawPolygon(tmp,4);

}

```