Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное
учреждение высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Институт естественных и точных наук
Кафедра прикладной математики и программирования

ОТЧЕТ

о выполнении лабораторной работы № 7 по дисциплине
«Математические основы компьютерной графики»

Автор работы,
студент группы ЕТ-212
_____ Алиев Э.Ф.
«____» _____ 2022 г.

Руководитель работы,
старший преподаватель
_____ Шелудько А.С.
«____» _____ 2022 г.

Челябинск 2022

1 ЗАДАНИЕ

1. Написать программу для построения гладкой кривой по четырем опорным точкам. При выборе опорных точек текущие координаты указателя мыши должны отображаться в графическом окне. Интерфейс программы должен содержать следующие элементы управления:

- выбор опорных точек;
- построение кубической кривой Безье;
- построение кривой по алгоритму Чайкина;
- сохранение результата в файл;
- выход из программы.

2 МАТЕМАТИЧЕСКАЯ МОДЕЛЬ

Класс Square с полями:

px[3] = (20, 0, 0)

py[3] = (0, 20, 0)

pz[3] = (0, 0, 20)

points[30] массив точек для отрисовки

Методы класса:

Square() конструктор, извлекает точки и полигоны

rote() вращает фигуру

move() двигает фигуру

draw() отрисовывает фигуру

Матрица точек:

$$\begin{pmatrix} 50 & 50 & 50 \\ 50 & -50 & 50 \\ -50 & -50 & 50 \\ -50 & 50 & 50 \\ 50 & 50 & -50 \\ 50 & -50 & -50 \\ -50 & -50 & -50 \\ -50 & 50 & -50 \end{pmatrix}$$

Матрица полигонов:

$$\begin{pmatrix} 0 & 1 & 2 & 3 \\ 4 & 5 & 6 & 7 \\ 0 & 1 & 5 & 4 \\ 1 & 2 & 6 & 5 \\ 2 & 3 & 7 & 6 \\ 0 & 3 & 7 & 4 \end{pmatrix}$$

Файл main.cpp

```
#include "graphics.h"
#include "control.h"
#include "task.h"

int main()
{
   initwindow(700, 600);

   IMAGE *image;

   image = loadBMP("fon.bmp");
   putimage(0, 0, image, COPY_PUT);

   create_control(SAVE, 550, 0, 650, 50);
   create_control(EXIT, 650, 0, 700, 50);
   create_control(LEFT, 0, 550, 100, 600);
   create_control(RIGHT, 100, 550, 200, 600);
   create_control(UP, 200, 550, 300, 600);
   create_control(DOWN, 300, 550, 400, 600);
   create_control(ROTEZ, 400, 550, 500, 600);
   create_control(ROTEY, 500, 550, 600, 600);
   create_control(ROTEX, 600, 550, 700, 600);
   Square square = Square();
   square.draw();

   while (true)
   {
      while (mousebuttons() != 1);
      switch (select_control())
      {
         case NONE: break;
         case RIGHT :
            square.move(0, 1);
            putimage(0, 0, image, COPY_PUT);
            square.draw();
            break;
         case LEFT :
            square.move(0, -1);
            putimage(0, 0, image, COPY_PUT);
            square.draw();
            break;
         case UP :
            square.move(1, 1);
            putimage(0, 0, image, COPY_PUT);
            square.draw();
            break;
         case DOWN:
            square.move(1, -1);
```

```
            putimage(0, 0, image, COPY_PUT);
            square.draw();
            break;
         case ROTEX:
            square.rote(0);
            putimage(0, 0, image, COPY_PUT);
            square.draw();
            break;
         case ROTEY:
            square.rote(1);
            putimage(0, 0, image, COPY_PUT);
            square.draw();
            break;
         case ROTEZ:
            square.rote(2);
            putimage(0, 0, image, COPY_PUT);
            square.draw();
            break;
         case SAVE: save(); break;
         case EXIT: closegraph(); return 0;
      }
      delay(50);
   }
   freeimage(image);
}
```

Файл task.h

```
#include <vector>
#include <fstream>
#ifndef TASK_H
#define TASK_H
#define WIDTH 700
#define HEIGHT 588

using namespace std;

class Square
{
private:
   int px[3] = {20, 0, 0};
   int py[3] = {0, 20, 0};
   int pz[3] = {0, 0, 20};
   int [30];
   int kolPoint, kolSide;
   int dir[3] = {0, 0, 0};
   double **Points;
   int **Sides;
public:
   Square();
   void rote(int);
   void move(int, int);
   void draw();
};

void save();

#endif
```

Файл task.cpp

```
#include "graphics.h"
#include "task.h"
#include <cmath>

using namespace std;

Square::Square()
{
   ifstream f("Square.txt");
   f >> kolPoint >> kolSide;
   Points = new double*[kolPoint];
   for (int i = 0; i < kolPoint; i++)
   {
      Points[i] = new double[3];
      for (int j = 0; j < 3; j++)
```

```
        {
            f >> Points[i][j];
        }
    }
    Sides = new int*[kolSide];
    for (int i = 0; i < kolSide; i++)
    {
        Sides[i] = new int[4];
        for (int j = 0; j < 4; j++)
        {
            f >> Sides[i][j];
        }
    }
    f.close();
}

void Square::rote(int mode)
{
    double temp, angle  = acos(-1) / 20;
    for (int i = 0; i < kolPoint; i++)
    {
        Points[i][0] -= dir[0];
        Points[i][1] -= dir[1];
        Points[i][2] -= dir[2];
    }

    if(mode == 0)
    {
        for (int i = 0; i < kolPoint; i++)
        {
            temp = Points[i][0];
            Points[i][0] = Points[i][0]*cos(angle)
                              + Points[i][1]*sin(angle);
            Points[i][1] = - temp * sin(angle)
                         + Points[i][1]*cos(angle);
        }
    }
    if(mode == 1)
    {
        for (int i = 0; i < kolPoint; i++)
        {
            temp = Points[i][1];
            Points[i][1] = Points[i][1] * cos(angle)
                           + Points[i][2] * sin(angle);
            Points[i][2] = - temp * sin(angle)
                         + Points[i][2]*cos(angle);
        }
    }
    if(mode == 2)
    {
        for (int i = 0; i < kolPoint; i++)
        {
```

```
            temp = Points[i][0];
            Points[i][0] = Points[i][0]*cos(angle)
                                 + Points[i][2]*sin(angle);
            Points[i][2] = - temp * sin(angle)
                           + Points[i][2]*cos(angle);
        }
    }
    for (int i = 0; i < kolPoint; i++)
    {
        Points[i][0] += dir[0];
        Points[i][1] += dir[1];
        Points[i][2] += dir[2];
    }
}

void Square::move(int mode, int direction)
{
    dir[0] += direction*px[mode];
    dir[1] += direction*py[mode];
    dir[2] += direction*pz[mode];
    for (int i = 0; i < kolPoint; i++)
    {
        Points[i][0] += direction*px[mode];
        Points[i][1] += direction*py[mode];
        Points[i][2] += direction*pz[mode];
    }
}

void Square::draw()
{
    int k;
    setcolor(WHITE);
    setlinestyle(SOLID_LINE, 0, 1);
    for(int i=0; i<kolSide; i++)
    {
        k = 0;
        for(int j=0; j<4; j++)
        {
              points[2*k] = WIDTH/2 + Points[Sides[i][j]][0];
              points[2*k+1] = HEIGHT/2 - Points[Sides[i][j]][1];
              k++;
        }
        points[2*k] = points[0];
        points[2*k+1] = points[1];
        k++;
        drawpoly(k, points);
    }
}

void save()
{
    int W, H;
```

```
    IMAGE *output;

    W = getmaxx() + 1;
    H = getmaxy() + 1;
    output = createimage(W, H);

    getimage(0, 0, W - 1, H - 1, output);
    getimage(0, 0, W - 1, H - 1, output);
    saveBMP("output.bmp", output);
    freeimage(output);
}
```

Файл control.h

```
#ifndef CONTROL_H
#define CONTROL_H

enum control_values{
    NONE = -1,
    EXIT, SAVE,
    RIGHT, LEFT,
    UP, DOWN,
    ROTEX, ROTEY, ROTEZ,
    N_CONTROLS };

struct Control {
    int left;
    int top;
    int right;
    int bottom;
};

void create_control(int, int, int, int, int);
int select_control();

#endif
```

Файл control.cpp

```
#include "graphics.h"
#include "control.h"
Control controls[N_CONTROLS];

void create_control(int i, int left, int top, int right, int bottom)
{
    controls[i].left   = left;
    controls[i].top    = top;
    controls[i].right  = right-1;
    controls[i].bottom = bottom-1;
}
```

```
int select_control()
{
   int x, y;

   x = mousex();
   y = mousey();

   for (int i = 0; i < N_CONTROLS; i++)
   {
      if (x > controls[i].left && x < controls[i].right &&
          y > controls[i].top  && y < controls[i].bottom)
      {
         return i;
      }
   }

   return NONE;
}
```

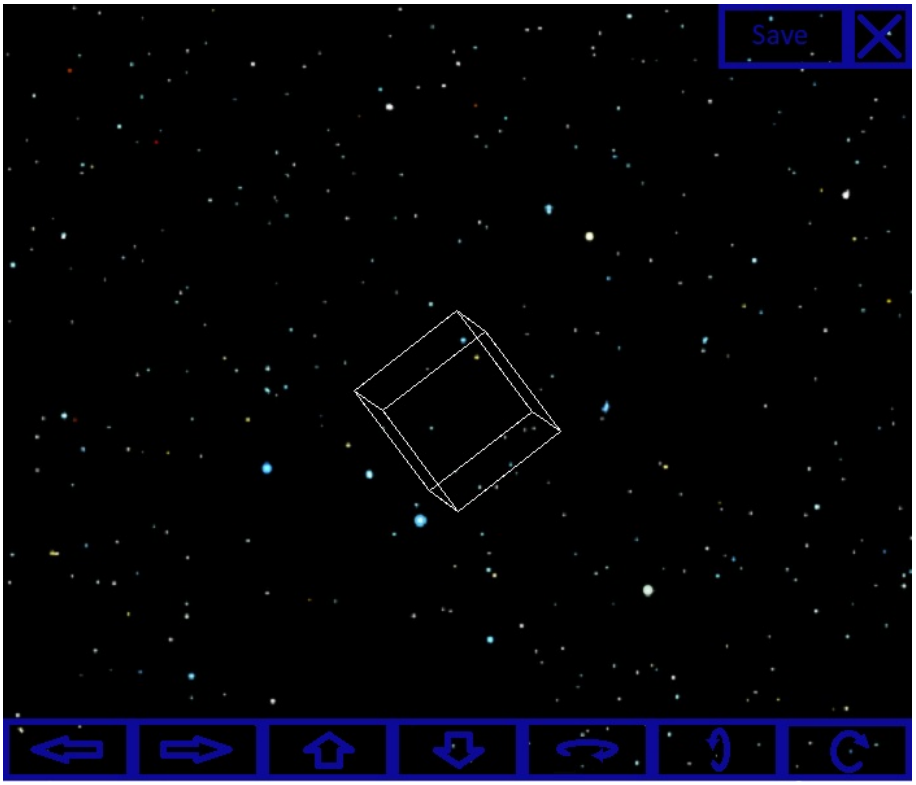

Рисунок 4.1 – Результат выполнения программы