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-211
_____ Савонин М.В.
«____» _____ 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 Figure с полями:

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

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

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

figure[30] массив точек для отрисовки

Методы класса:

Figure() конструктор, извлекает точки и полигоны

rote() вращает фигуру

move() двигает фигуру

draw() отрисовывает фигуру

Матрица точек:

$$\begin{pmatrix} 0 & 0 & 70 \\ 114 & 37 & -50 \\ 70 & -97 & -50 \\ -70 & -97 & -50 \\ -114 & 37 & -50 \\ 0 & 120 & -50 \end{pmatrix}$$

Матрица полигонов:

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

Файл main.cpp

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

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

   create_control(SAVE, 0, 0, "save.bmp");
   create_control(EXIT, 600, 0, "exit.bmp");
   create_control(RX, 0, 550, "RX.bmp");
   create_control(RY, 100, 550, "RY.bmp");
   create_control(RZ, 200, 550, "RZ.bmp");
   create_control(RIGHT, 300, 550, "right.bmp");
   create_control(LEFT, 400,550, "left.bmp");
   create_control(UP, 500, 550, "up.bmp");
   create_control(DOWN, 600, 550, "down.bmp");
   Figure Piramida = Figure();
   Piramida.draw();

   while (true)
   {
      while (mousebuttons() != 1);
      switch (select_control())
      {
         case NONE: break;
         case RIGHT :
            Piramida.move(0, 1);
            setfillstyle(SOLID_FILL, BLACK);
            bar(100, 0, 600, 50);
            bar(0, 50, 700, 500);
            Piramida.draw();
            break;
         case LEFT :
            Piramida.move(0, -1);
            setfillstyle(SOLID_FILL, BLACK);
            bar(100, 0, 600, 50);
            bar(0, 50, 700, 500);
            Piramida.draw();
            break;
         case UP :
            Piramida.move(1, 1);
            setfillstyle(SOLID_FILL, BLACK);
            bar(100, 0, 600, 50);
            bar(0, 50, 700, 500);
            Piramida.draw();
            break;
         case DOWN:
```

```
            Piramida.move(1, -1);
            setfillstyle(SOLID_FILL, BLACK);
            bar(100, 0, 600, 50);
            bar(0, 50, 700, 500);
            Piramida.draw();
            break;
         case RX:
            Piramida.rote(0);
            setfillstyle(SOLID_FILL, BLACK);
            bar(100, 0, 600, 50);
            bar(0, 50, 700, 500);
            Piramida.draw();
            break;
         case RY:
            Piramida.rote(1);
            setfillstyle(SOLID_FILL, BLACK);
            bar(100, 0, 600, 50);
            bar(0, 50, 700, 500);
            Piramida.draw();
            break;
         case RZ:
            Piramida.rote(2);
            setfillstyle(SOLID_FILL, BLACK);
            bar(100, 0, 600, 50);
            bar(0, 50, 700, 500);
            Piramida.draw();
            break;
         case SAVE: save(); break;
         case EXIT: closegraph(); return 0;
      }
      delay(50);
   }
}
```

Файл task.h

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

using namespace std;

class Figure
{
private:
   int px[3] = {20, 0, 0};
   int py[3] = {0, 20, 0};
   int pz[3] = {0, 0, 20};
   int figure[30];
   int numPoint, numPol;
   int dir[3] = {0, 0, 0};
   double **Points;
   int **Pol;
public:
   Figure();
   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;

Figure::Figure()
{
   ifstream f("Piramida.txt");
   f >> numPoint >> numPol;
   Points = new double*[numPoint];
   for (int i = 0; i < numPoint; i++)
   {
      Points[i] = new double[3];
      for (int j = 0; j < 3; j++)
```

```
        {
            f >> Points[i][j];
        }
    }
    Pol = new int*[numPol];
    for (int i = 0; i < numPol; i++)
    {
        Pol[i] = new int[3];
        for (int j = 0; j < 3; j++)
        {
            f >> Pol[i][j];
        }
    }
    f.close();
}

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

    if(mode == 0)
    {
        for (int i = 0; i < numPoint; 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 < numPoint; 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);
        }
    }
    if(mode == 2)
    {
        for (int i = 0; i < numPoint; 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);
        }
    }
    for (int i = 0; i < numPoint; i++)
    {
        Points[i][0] += dir[0];
        Points[i][1] += dir[1];
        Points[i][2] += dir[2];
    }
}

void Figure::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 < numPoint; i++)
    {
        Points[i][0] += direction*px[mode];
        Points[i][1] += direction*py[mode];
        Points[i][2] += direction*pz[mode];
    }
}

void Figure::draw()
{
    int k;
    setcolor(WHITE);
    setlinestyle(SOLID_LINE, 0, 1);
    for(int i=0; i<numPol; i++)
    {
        k = 0;
        for(int j=0; j<3; j++)
        {
              figure[2*k] = WIDTH/2 + Points[Pol[i][j]][0];
              figure[2*k+1] = HEIGHT/2 - Points[Pol[i][j]][1];
              k++;
        }
        figure[2*k] = figure[0];
        figure[2*k+1] = figure[1];
        k++;
        drawpoly(k, figure);
    }
}

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,
    RX, RY, RZ,
    N_CONTROLS };

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

void create_control(int, int, int, const char*);
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,
                    const char *file_name)
{
    IMAGE *image;

    image = loadBMP(file_name);
    putimage(left, top, image, COPY_PUT);
```

```
   controls[i].left   = left;
   controls[i].top    = top;
   controls[i].right  = left + imagewidth(image) - 1;
   controls[i].bottom = top  + imageheight(image) - 1;

   freeimage(image);
}

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;
}
```

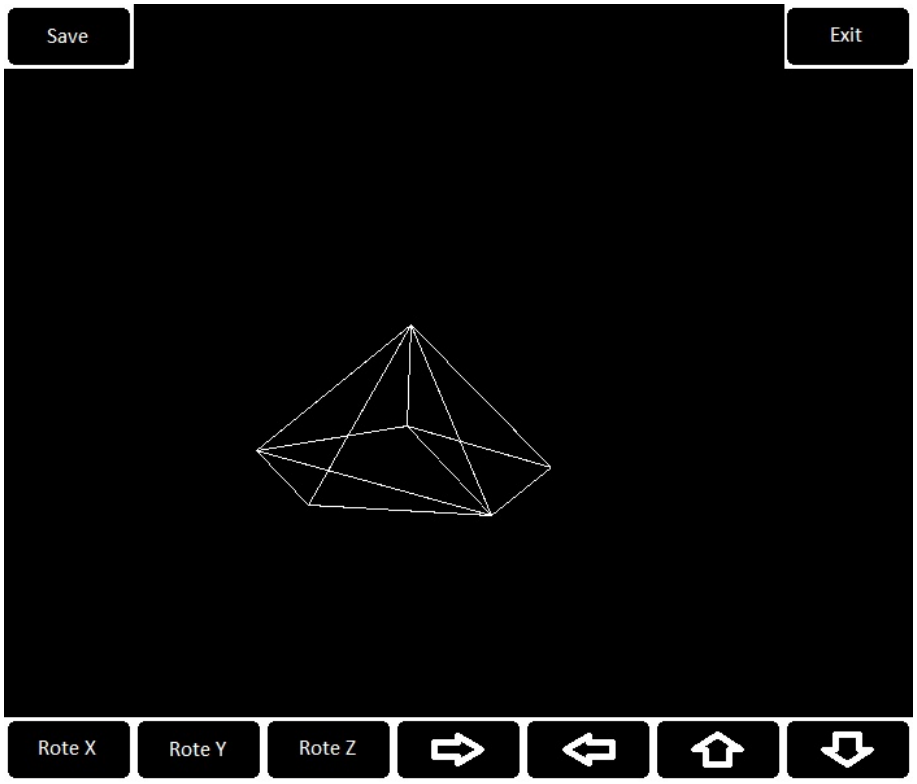

Рисунок 4.1 – Результат выполнения программы