ГУАП

КАФЕДРА № 14

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

доцент., канд. тех. наук		А.В. Шахомиров
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №5

Закраска трехмерной фигуры с помощью алгоритма художника

по курсу: Компьютерная графика

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТЫ ГР. №	1842	12.12.2020	Г.А.Кольцов
		подпись, дата	инициалы, фамилия

Санкт-Петербург 2020

1. Цель работы

Реализовать программу построения стандартной функцией, интерактивного управления трехмерной фигуры и покраска - Пирамида (сдвиг, поворот, масштабирование, закраска). Программа реализована в среде разработки RAD Studio, как стандартное приложение windows с использованием стандартной библиотеки VCL.h. Для покраски использован алгоритм художника, который в некоторых случаях может выдавать ошибки, тк могут совпадать средние глубины двух граней.

Ссылка на GitHub:
https://github.com/KolcovBruin/Graphic/blob/main/Lab_5.cpp

Управление фигурой:

2. Листинг программы

```
#include <vcl.h>
#pragma hdrstop

#include "Lab_5.h"
//---------------------------------------------------------------
---------------
#pragma package(smart_init)
#pragma resource "*.dfm"
#include <stdlib.h>
#include <stdio.h>
#include <dos.h>
#define Row_Line 2 // number of matrix line rows
#define Colm_Line 3 // number of matrix line columns
#define Row_cnv 3 // number of conversion matrix rows
#define Colm_cnv 3 // number of conversion matrix columns
//#define PI 3.14159265    //   math.h
int enable_Br = 0;
int enable_std = 0;
float Brz_L[Row_Line][Colm_Line];
float Line_std[4][4];
float New_fig[4][4];
```

```
float z_med[4];
float MTX[Row_Line][Colm_Line];    //??
double Mod[4][4];
double Mod_z[4][4]={{1,0,0,0},{0,1,0,0},{sqrt(2)/2,sqrt(2)/
2,0,0},{0,0,0,1}};
//double Mod_z[4][4]={{sqrt(1/2),-sqrt(1/6),sqrt(1/3),0},
{0,sqrt(2/3),sqrt(1/3),0},{-sqrt(1/2),-
sqrt(1/6),sqrt(1/3),0},{0,0,0,1}};
float max_x[4];
float max_y[4];
float min_x[4];
float min_y[4];
float swop;
TForm1 *Form1;
//---------------------------------------------------------
---------------
__fastcall TForm1::TForm1(TComponent* Owner)
    : TForm(Owner)
{
}
//---------------------------------------------------------
---------------
void Start_Piramida()
{
    Line_std[0][0]=100.0;
    Line_std[0][1]=250.0;
    Line_std[0][2]=0.0;
    Line_std[0][3]=1.0;

    Line_std[1][0]=300.0;
    Line_std[1][1]=250.0;
    Line_std[1][2]=0.0;
    Line_std[1][3]=1.0;

    Line_std[2][0]=260.0;
    Line_std[2][1]=250.0;
    Line_std[2][2]=-100.0;
    Line_std[2][3]=1.0;

    Line_std[3][0]=225.0;
    Line_std[3][1]=100.0;
    Line_std[3][2]=-50.0;
    Line_std[3][3]=1.0;
}
```

```
void translate_z()
{

  for (int k = 0; k < 4; k++)
     {
          for (int i = 0; i < 4; i++)
          {
               New_fig[k][i]=0;
          }
     }
   for (int k = 0; k < 4; k++)
   {
        for (int i = 0; i < 4; i++) {
              for (int j = 0; j < 4; j++) {
                New_fig[k][i]+=Line_std[k][j]*Mod_z[j][i];
              }
        }
   }
}
 void mull_mtx(double MOD[4][4])
{
  for (int k = 0; k < 4; k++)
     {
          for (int i = 0; i < 4; i++)
          {
               New_fig[k][i]=0;
          }
     }
   for (int k = 0; k < 4; k++)
   {
        for (int i = 0; i < 4; i++) {
              for (int j = 0; j < 4; j++) {
                New_fig[k][i]+=Line_std[k][j]*MOD[j][i];
              }
        }
  }
  for (int k = 0; k < 4; k++)
     {
          for (int i = 0; i < 4; i++)
          {
               Line_std[k][i] = New_fig[k][i];
          }
     }
}
void max_min()
```

```
{

     for (int i = 0; i < 4; i++)
     {
     max_x[i]=0;
     max_y[i]=0;
     min_x[i]=10000;
     min_y[i]=10000;
     }

     for (int j=0; j < 4; j++)
     {



     for (int i = 0; i < 3; i++)
          {
               if (j==1)
               {
                if(i==2)
                {
                i=3;
                }
               }
               if (j==2)
               {
               i++;
               }
               if (j==3)
               {
                    if(i>0)
                    {
                         i++;
                    }
               }

               z_med[j]+=Line_std[i][2];

               if (New_fig[i][0]>max_x[j]) {
                    max_x[j]=New_fig[i][0];
               }
               if (New_fig[i][1]>max_y[j]) {
                    max_y[j]=New_fig[i][1];
               }
               if (New_fig[i][0]<min_x[j]) {
```

```
                    min_x[j]=New_fig[i][0];
                }
                if (New_fig[i][1]<min_y[j]) {
                    min_y[j]=New_fig[i][1];
                }
                if ((j==2) or ((j==3) and (i>1)))
                {
                i--;
                }

            }

        }

        for (int i = 0; i < 4; i++)
        {
            z_med[i]/=3; //
        }

}

void alg_Painter()
{
        int color[3];
         //
        float triangle[3][2];
        TColor col[4]={clYellow,clBlue,clGreen,clBlack};    //
        for (int i = 0; i < 4; i++)  //
        {
        float z_min[2]={1000,0};
          for (int j = 0; j < 4; j++)
          {
              if (z_med[j]<z_min[0])
              {
                z_min[0]=z_med[j];
                z_min[1]=j;
              }
          }
           z_med[(int)z_min[1]]=1000;
          for (int k = 0; k < 3; k++) //
             {
                  int ki=k;

                  if (z_min[1]==1)
                  {
```

```
 if(k==2)
 {
 k=3;
 }
}
if (z_min[1]==2)
{
k++;
}
if (z_min[1]==3)
{
     if(k>0)
     {
          k++;
     }
}

triangle[ki][0]=New_fig[k][0];
triangle[ki][1]=New_fig[k][1];
//
if ((z_min[1]==2) or ((z_min[1]==3) and (k>1)))
{
k--;
}




}

     if(z_min[1]==0)
     {
      color[0]=255;
      color[1]=254;
      color[2]=253;
     }
     if(z_min[1]==1)
     {
      color[0]=255;
      color[1]=251;
      color[2]=252;
     }
     if(z_min[1]==2)
     {
```

```
                color[0]=254;
                color[1]=250;
                color[2]=251;
               }
               if(z_min[1]==3)
               {
                color[0]=253;
                color[1]=250;
                color[2]=252;
               }
               Form1->Canvas->Pen->Color=(TColor) RGB
(color[0],0,0); //
               Form1->Canvas->MoveTo(round(triangle[0]
[0]),round(triangle[0][1]));
               Form1->Canvas->LineTo(round(triangle[1]
[0]),round(triangle[1][1]));
               Form1->Canvas->Pen->Color=(TColor) RGB
(color[1],0,0);
               Form1->Canvas->LineTo(round(triangle[2]
[0]),round(triangle[2][1]));
               Form1->Canvas->Pen->Color=(TColor) RGB
(color[2],0,0);
               Form1->Canvas->LineTo(round(triangle[0]
[0]),round(triangle[0][1]));

               float q1;
               float q2;
               float q3;
               for (int x = min_x[(int)z_min[1]]; x
<max_x[(int)z_min[1]] ; x++)
               {
                     for (int y = min_y[(int)z_min[1]]; y
<max_y[(int)z_min[1]]; y++)
                     {
                         q1=x*(triangle[1][1]-triangle[0][1])
+y*(triangle[0][0]-triangle[1][0])+triangle[0][1]*triangle[1]
[0]-triangle[0][0]*triangle[1][1];
                         q2=x*(triangle[2][1]-triangle[1][1])
+y*(triangle[1][0]-triangle[2][0])+triangle[1][1]*triangle[2]
[0]-triangle[1][0]*triangle[2][1];
                         q3=x*(triangle[0][1]-triangle[2][1])
+y*(triangle[2][0]-triangle[0][0])+triangle[2][1]*triangle[0]
[0]-triangle[2][0]*triangle[0][1];
//
```

```
                        if (((q1 >= 0) and (q2 >= 0) and (q3
>= 0)) or ((q1 < 0) and (q2 < 0) and (q3 < 0)))
                        {
//                          if((Form1->Canvas->Pixels[x][y] !
=(TColor) RGB (color[0],0,0))and(Form1->Canvas->Pixels[x]
[y] != (TColor) RGB (color[1],0,0))and(Form1->Canvas-
>Pixels[x][y] != (TColor) RGB (color[2],0,0)))
//                          {
//                               Form1->Canvas->Pixels[x][y] =
col[i];
//                          }
                            Form1->Canvas->Pixels[x][y] =
col[i];
                        }
                      // Form1->Canvas->Pixels[x][y] = col[i];
                    }
                }
     }


}
void refresh()

{
     Form1->Canvas->Brush->Color = clBtnFace;
     Form1->Canvas->FillRect(Form1->Canvas->ClipRect);
     translate_z();
if (enable_std)
     {
                                          ////(TColor) RGB
(255,0,0)
          Form1->Canvas->Pen->Color=(TColor) RGB (255,0,0); //
          Form1->Canvas->MoveTo(round(New_fig[0]
[0]),round(New_fig[0][1]));
          Form1->Canvas->LineTo(round(New_fig[1]
[0]),round(New_fig[1][1]));
          Form1->Canvas->Pen->Color=(TColor) RGB (254,0,0);
          Form1->Canvas->LineTo(round(New_fig[2]
[0]),round(New_fig[2][1]));
          Form1->Canvas->Pen->Color=(TColor) RGB (253,0,0);
          Form1->Canvas->LineTo(round(New_fig[0]
[0]),round(New_fig[0][1]));
          Form1->Canvas->Pen->Color=(TColor) RGB (252,0,0);
          Form1->Canvas->LineTo(round(New_fig[3]
[0]),round(New_fig[3][1]));
```

```
          Form1->Canvas->Pen->Color=(TColor) RGB (251,0,0);
          Form1->Canvas->LineTo(round(New_fig[1]
[0]),round(New_fig[1][1]));
          Form1->Canvas->Pen->Color=(TColor) RGB (250,0,0);
          Form1->Canvas->MoveTo(round(New_fig[3]
[0]),round(New_fig[3][1]));
          Form1->Canvas->LineTo(round(New_fig[2]
[0]),round(New_fig[2][1]));



     }
}

void __fastcall TForm1::Button1Click(TObject *Sender)
{

 if (enable_std == 0)
{
     enable_std = 1;
     Start_Piramida();
}
else
{
     enable_std = 0;
}
refresh();
}

//---------------------------------------------------------
----------------

void __fastcall TForm1::Button2Click(TObject *Sender)
{
/////
     Mod[0][0]=1; Mod[0][1]=0; Mod[0][2]=0; Mod[0][3]=0;
     Mod[1][0]=0; Mod[1][1]=1; Mod[1][2]=0; Mod[1][3]=0;
     Mod[2][0]=0; Mod[2][1]=0; Mod[2][2]=1; Mod[2][3]=0;
//-2*z_n
     Mod[3][0]=-5; Mod[3][1]=0; Mod[3][2]=0; Mod[3][3]=1;
     mull_mtx(Mod);

refresh();
}
```

```
//---------------------------------------------------------
---------------


void __fastcall TForm1::Button4Click(TObject *Sender)
{
//////
     Mod[0][0]=1; Mod[0][1]=0; Mod[0][2]=0; Mod[0][3]=0;
     Mod[1][0]=0; Mod[1][1]=1; Mod[1][2]=0; Mod[1][3]=0;
     Mod[2][0]=0; Mod[2][1]=0; Mod[2][2]=1; Mod[2][3]=0;
//-2*z_n
     Mod[3][0]=0; Mod[3][1]=-5; Mod[3][2]=0; Mod[3][3]=1;
     mull_mtx(Mod);

refresh();
}
//---------------------------------------------------------
---------------

void __fastcall TForm1::Button5Click(TObject *Sender)
{
     Mod[0][0]=1; Mod[0][1]=0; Mod[0][2]=0; Mod[0][3]=0;
     Mod[1][0]=0; Mod[1][1]=1; Mod[1][2]=0; Mod[1][3]=0;
     Mod[2][0]=0; Mod[2][1]=0; Mod[2][2]=1; Mod[2][3]=0;
//-2*z_n
     Mod[3][0]=0; Mod[3][1]=5; Mod[3][2]=0; Mod[3][3]=1;
     mull_mtx(Mod);

refresh();
}
//---------------------------------------------------------
---------------

void __fastcall TForm1::Button20Click(TObject *Sender)
{
     Mod[0][0]=1; Mod[0][1]=0; Mod[0][2]=0; Mod[0][3]=0;
     Mod[1][0]=0; Mod[1][1]=1; Mod[1][2]=0; Mod[1][3]=0;
     Mod[2][0]=0; Mod[2][1]=0; Mod[2][2]=1; Mod[2][3]=0;
//-2*z_n
     Mod[3][0]=0; Mod[3][1]=0; Mod[3][2]=5; Mod[3][3]=1;
     mull_mtx(Mod);

refresh();
}
```

```
//---------------------------------------------------------
---------------

void __fastcall TForm1::Button6Click(TObject *Sender)
{
//////
double x_n=0, y_n=0, z_n=0;
     for (int i = 0; i < 4; i++) {
           x_n+=Line_std[i][0];
           y_n+=Line_std[i][1];
           z_n+=Line_std[i][2];
     }
     x_n=x_n/4;
     y_n=y_n/4;
     z_n=z_n/4;
     Mod[0][0]=cos(5*M_PI/180);  Mod[0][1]=sin(5*M_PI/180);
Mod[0][2]=0; Mod[0][3]=0;
     Mod[1][0]=-sin(5*M_PI/180); Mod[1][1]=cos(5*M_PI/180);
Mod[1][2]=0; Mod[1][3]=0;
     Mod[2][0]=0; Mod[2][1]=0; Mod[2][2]=1; Mod[2][3]=0;
//-2*z_n
     Mod[3][0]=x_n * (1 - cos(5*M_PI/180)) + y_n * sin(5*M_PI/
180); Mod[3][1]=y_n * (1 - cos(5*M_PI/180)) - x_n *
sin(5*M_PI/180); Mod[3][2]=0; Mod[3][3]=1;

mull_mtx(Mod);

refresh();
}
//---------------------------------------------------------
---------------

void __fastcall TForm1::Button7Click(TObject *Sender)
{
//////
double x_n=0, y_n=0, z_n=0;
     for (int i = 0; i < 4; i++) {
           x_n+=Line_std[i][0];
           y_n+=Line_std[i][1];
           z_n+=Line_std[i][2];
     }
     x_n=x_n/4;
     y_n=y_n/4;
     z_n=z_n/4;
```

```
     Mod[0][0]=cos(-5*M_PI/180);  Mod[0][1]=sin(-5*M_PI/180);
Mod[0][2]=0; Mod[0][3]=0;
     Mod[1][0]=-sin(-5*M_PI/180); Mod[1][1]=cos(-5*M_PI/180);
Mod[1][2]=0; Mod[1][3]=0;
     Mod[2][0]=0; Mod[2][1]=0; Mod[2][2]=1; Mod[2][3]=0;
//-2*z_n
     Mod[3][0]=x_n * (1 - cos(-5*M_PI/180)) + y_n *
sin(-5*M_PI/180); Mod[3][1]=y_n * (1 - cos(-5*M_PI/180)) -
x_n * sin(-5*M_PI/180); Mod[3][2]=0; Mod[3][3]=1;

mull_mtx(Mod);

refresh();
}
//---------------------------------------------------------
---------------

void __fastcall TForm1::Button8Click(TObject *Sender)
{
//////
double x_n=0, y_n=0, z_n=0;
     for (int i = 0; i < 4; i++) {
           x_n+=Line_std[i][0];
           y_n+=Line_std[i][1];
           z_n+=Line_std[i][2];
     }
     x_n=x_n/4;
     y_n=y_n/4;
     z_n=z_n/4;
     Mod[0][0]=cos(5*M_PI/180);  Mod[0][2]=-sin(5*M_PI/180);
Mod[0][1]=0; Mod[0][3]=0;
     Mod[1][0]=0; Mod[1][1]=1; Mod[1][2]=0; Mod[2][3]=0;
     Mod[2][0]=sin(5*M_PI/180); Mod[2][2]=cos(5*M_PI/180);
Mod[2][1]=0; Mod[1][3]=0;
     Mod[3][0]=x_n * (1 - cos(5*M_PI/180)) - z_n * sin(5*M_PI/
180); Mod[3][2]=z_n * (1 - cos(5*M_PI/180)) + x_n *
sin(5*M_PI/180); Mod[3][1]=0; Mod[3][3]=1;

mull_mtx(Mod);

refresh();
}
//---------------------------------------------------------
---------------
```

```
void __fastcall TForm1::Button9Click(TObject *Sender)
{
//////
double x_n=0, y_n=0, z_n=0;
     for (int i = 0; i < 4; i++) {
          x_n+=Line_std[i][0];
          y_n+=Line_std[i][1];
          z_n+=Line_std[i][2];
     }
     x_n=x_n/4;
     y_n=y_n/4;
     z_n=z_n/4;
     Mod[0][0]=cos(-5*M_PI/180);  Mod[0][2]=-sin(-5*M_PI/180);
Mod[0][1]=0; Mod[0][3]=0;
     Mod[1][0]=0; Mod[1][1]=1; Mod[1][2]=0; Mod[2][3]=0;
     Mod[2][0]=sin(-5*M_PI/180); Mod[2][2]=cos(-5*M_PI/180);
Mod[2][1]=0; Mod[1][3]=0;
     Mod[3][0]=x_n * (1 - cos(-5*M_PI/180)) - z_n *
sin(-5*M_PI/180); Mod[3][2]=z_n * (1 - cos(-5*M_PI/180)) +
x_n * sin(-5*M_PI/180); Mod[3][1]=0; Mod[3][3]=1;

mull_mtx(Mod);

refresh();

}
//---------------------------------------------------------
---------------

void __fastcall TForm1::Button10Click(TObject *Sender)
{
 Canvas->Brush->Color = clBtnFace;
Canvas->FillRect(Canvas->ClipRect);
}
//---------------------------------------------------------
---------------

void __fastcall TForm1::Button15Click(TObject *Sender)
{
//////
double x_n=0, y_n=0, z_n=0;
     for (int i = 0; i < 4; i++) {
          x_n+=Line_std[i][0];
          y_n+=Line_std[i][1];
          z_n+=Line_std[i][2];
```

```
     }
     x_n=x_n/4;
     y_n=y_n/4;
     z_n=z_n/4;
     Mod[0][0]=1; Mod[0][1]=0; Mod[0][2]=0; Mod[0][3]=0;
     Mod[1][0]=0; Mod[1][1]=cos(5*M_PI/180); Mod[1]
[2]=sin(5*M_PI/180); Mod[1][3]=0;
     Mod[2][0]=0; Mod[2][1]=-sin(5*M_PI/180); Mod[2]
[2]=cos(5*M_PI/180);  Mod[2][3]=0;
                                                                  //-
2*z_n
     Mod[3][0]=0; Mod[3][1]=y_n * (1 - cos(5*M_PI/180)) + z_n
* sin(5*M_PI/180); Mod[3][2]=z_n * (1 - cos(5*M_PI/180)) -
y_n * sin(5*M_PI/180);  Mod[3][3]=1;

mull_mtx(Mod);

refresh();
}
//---------------------------------------------------------
---------------

void __fastcall TForm1::Button16Click(TObject *Sender)
{
double x_n=0, y_n=0, z_n=0;
     for (int i = 0; i < 4; i++) {
          x_n+=Line_std[i][0];
          y_n+=Line_std[i][1];
          z_n+=Line_std[i][2];
     }
     x_n=x_n/4;
     y_n=y_n/4;
     z_n=z_n/4;
     Mod[0][0]=1; Mod[0][1]=0; Mod[0][2]=0; Mod[0][3]=0;
     Mod[1][0]=0; Mod[1][1]=cos(-5*M_PI/180); Mod[1]
[2]=sin(-5*M_PI/180); Mod[1][3]=0;
     Mod[2][0]=0; Mod[2][1]=-sin(-5*M_PI/180); Mod[2]
[2]=cos(-5*M_PI/180);  Mod[2][3]=0;
                                                                  //-
2*z_n
     Mod[3][1]=y_n * (1 - cos(-5*M_PI/180)) + z_n *
sin(-5*M_PI/180); Mod[3][2]=z_n * (1 - cos(-5*M_PI/180)) -
y_n * sin(-5*M_PI/180); Mod[3][0]=0; Mod[3][3]=1;

mull_mtx(Mod);
```

```
refresh();
}


//---------------------------------------------------------
---------------

void __fastcall TForm1::Button17Click(TObject *Sender)
{
 double x_n=0, y_n=0, z_n=0;
     for (int i = 0; i < 4; i++) {
           x_n+=Line_std[i][0];
           y_n+=Line_std[i][1];
           z_n+=Line_std[i][2];
     }
     x_n=x_n/4;
     y_n=y_n/4;
     z_n=z_n/4;
     Mod[0][0]=1.05; Mod[0][1]=0; Mod[0][2]=0; Mod[0][3]=0;
     Mod[1][0]=0; Mod[1][1]=1.05; Mod[1][2]=0; Mod[1][3]=0;
     Mod[2][0]=0; Mod[2][1]=0; Mod[2][2]=1.05;  Mod[2][3]=0;

                 //-2*z_n
     Mod[3][0]=x_n*(1-1.05); Mod[3][1]=y_n*(1-1.05); Mod[3]
[2]=z_n*(1-1.05); Mod[3][3]=1;

mull_mtx(Mod);

refresh();
}
//---------------------------------------------------------
---------------


void __fastcall TForm1::Button19Click(TObject *Sender)
{
     Mod[0][0]=1; Mod[0][1]=0; Mod[0][2]=0; Mod[0][3]=0;
     Mod[1][0]=0; Mod[1][1]=1; Mod[1][2]=0; Mod[1][3]=0;
     Mod[2][0]=0; Mod[2][1]=0; Mod[2][2]=1; Mod[2][3]=0;
//-2*z_n
     Mod[3][0]=0; Mod[3][1]=0; Mod[3][2]=-5; Mod[3][3]=1;
     mull_mtx(Mod);

refresh();
```

```
}
//---------------------------------------------------------------
---------------

void __fastcall TForm1::Button21Click(TObject *Sender)
{
//////
max_min();
alg_Painter();
}
//---------------------------------------------------------------
---------------



void __fastcall TForm1::Button3Click(TObject *Sender)
{

/////
     Mod[0][0]=1; Mod[0][1]=0; Mod[0][2]=0; Mod[0][3]=0;
     Mod[1][0]=0; Mod[1][1]=1; Mod[1][2]=0; Mod[1][3]=0;
     Mod[2][0]=0; Mod[2][1]=0; Mod[2][2]=1; Mod[2][3]=0;
//-2*z_n
     Mod[3][0]=5; Mod[3][1]=0; Mod[3][2]=0; Mod[3][3]=1;
     mull_mtx(Mod);

refresh();
}

//---------------------------------------------------------------
---------------

void __fastcall TForm1::Button18Click(TObject *Sender)
{
///
double x_n=0, y_n=0, z_n=0;
     for (int i = 0; i < 4; i++) {
           x_n+=Line_std[i][0];
           y_n+=Line_std[i][1];
           z_n+=Line_std[i][2];
     }
     x_n=x_n/4;
     y_n=y_n/4;
     z_n=z_n/4;
     Mod[0][0]=0.95; Mod[0][1]=0; Mod[0][2]=0; Mod[0][3]=0;
```

```
    Mod[1][0]=0; Mod[1][1]=0.95; Mod[1][2]=0; Mod[1][3]=0;
    Mod[2][0]=0; Mod[2][1]=0; Mod[2][2]=0.95;  Mod[2][3]=0;

                //-2*z_n
    Mod[3][0]=x_n*(1-0.95); Mod[3][1]=y_n*(1-0.95); Mod[3]
[2]=z_n*(1-0.95); Mod[3][3]=1;

mull_mtx(Mod);

refresh();
}
//-------------------------------------------------------------
---------------
```

3. Тестовые примеры

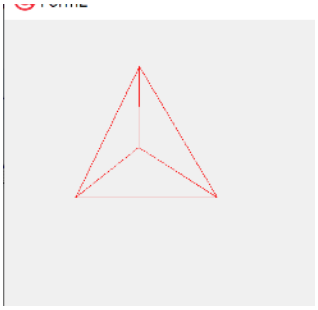

Рисунок 1 - отрисовка Пирамиды. Кнопка «Пирамида»

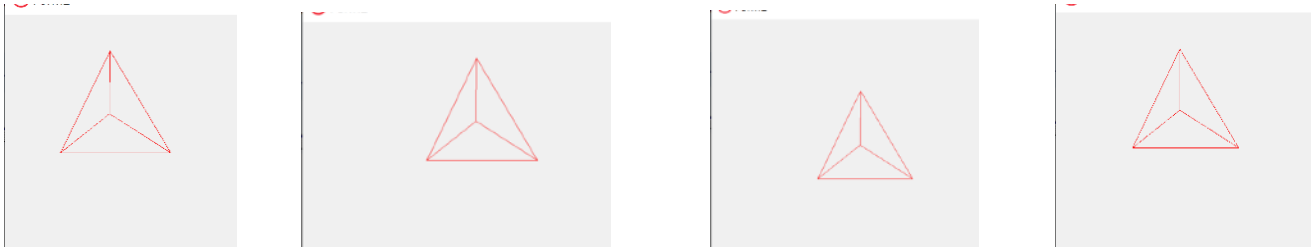

Рисунок 2 - Перемещение. Кнопки «Вправо» и «Влево», «Вверх» и «Вниз», «Назад» и «Вперед»

Рисунок 3 - Масштабирование. Кнопки «Увелич» и «Уменьш»

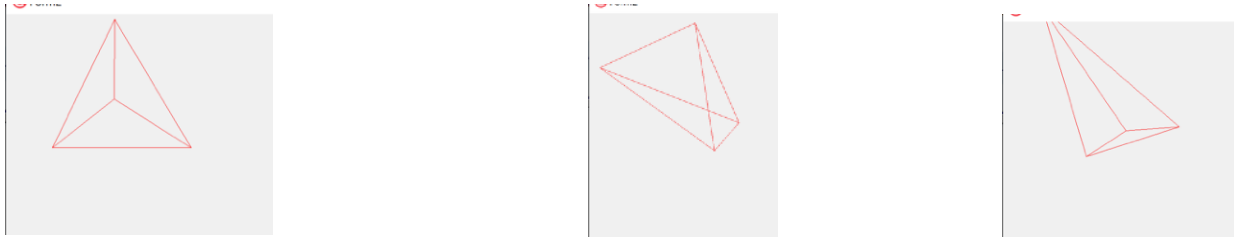

Рисунок 4 - Вращение по всем осям. Кнопки «По X+»(2) и «По X-», «По Y+» и «По Y-», «По Z+»(3) и «По Z-»

Рисунок 5 - Покраска. Кнопка «Закраска»

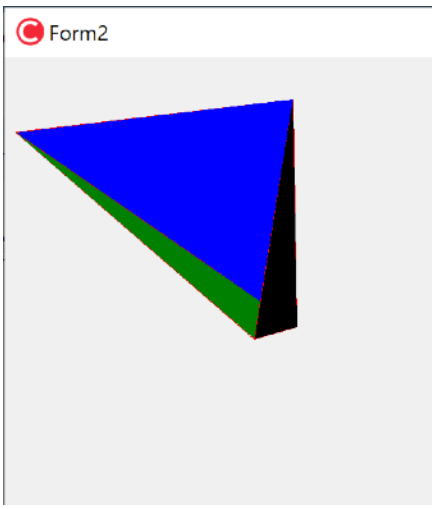

Рисунок 6 - Ошибка алгоритма художника