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 № 3 по дисциплине
«Математические основы компьютерной графики»

Автор работы,
студент группы ЕТ-211
_____ Новгородцев Н.В.
«____» _____ 2022 г.

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

Челябинск 2022

1 ЗАДАНИЕ

1. Привести описание и схему алгоритма By для растрового представления линии.

2. Разработать подпрограмму для рисования линии (аналог процедуры line из графической библиотеки). Аргументы подпрограммы – координаты начальной и конечной точек. При реализации подпрограммы использовать для рисования только процедуру putpixel. Для определения текущего цвета рисования использовать функцию getcolor.

3. Разработать подпрограмму для рисования правильной звезды. Аргументы подпрограммы – координаты центра, радиус описанной окружности и число вершин. При создании контура звезды использовать собственную подпрограмму рисования линии. Для закраски фигуры использовать процедуру floodfill.

4. Написать программу для тестирования разработанных подпрограмм. Интерфейс программы должен содержать следующие элементы управления:

- увеличение/уменьшение числа вершин;
- увеличение/уменьшение размера (радиуса описанной окружности);
- сохранение результата в файл;
- выход из программы.

2 МАТЕМАТИЧЕСКАЯ МОДЕЛЬ

Пусть x_0, y_0, x_1, y_1 – соответственно координаты первой и второй точки. При рисовании линии цвет задан изначально *color* в формате RGB и будем говорить что он равет col, а координаты простовляемого пикселя x и y.

Отрисовка линии:

$$deltax = |x_1 - x_0|.$$

$$deltay = |y_1 - y_0|.$$

$$error = 0.$$

$$deltaerr = 0.$$

Если deltay $>$ deltax, то:

Если deltay не равно 0 то:

$$deltaerr = dx/dy.$$

Если $y_0 > y_1$, то меняем точки местами.

$$x = x_0.$$

$$dir = x_1 - x_0.$$

$$dir = \begin{cases} 1 & \text{для } dir > 0 \\ -1 & \text{для } dir \leq 0 \end{cases}$$

Потом перебераем все значения y от y_0 до y_1:

Ставим пиксель в координах x y и с цветом color.

$$error = error + deltaerr.$$

Если error ≥ 1:

$$x = x + dir.$$

$$err = err - 1.$$

Если deltay $\geq$ deltaч, то делаем анологичные действия меняя в обработке x на y и y на x

Отрисовка звезды:

Пусть x и y центр звезды, r и n радиус внешней окружности и количество внешних углов.

$$kof = \frac{cos(3.14/n * 2)}{cos(3.14/n)}.$$

Перебираем каждый угол, где i это номер угла от 0 до 2n-1 включительно. Если i чётное то:

$$x_i = x + r * cos(i * 3.14/n).$$

$$y_i = y + r * sin(i * 3.14/n).$$

Иначе:

$$x_i = x + r * kof * cos(i * 3.14/n).$$

$$y_i = y + r * kof * sin(i * 3.14/n).$$

После чего проводятся последовательные линии между точками(углами).

3 ТЕКСТ ПРОГРАММЫ

Файл main.cpp

```
#include <iostream>
#include "graphics.h"
#include "control.h"
#include "task.h"

using namespace std;

int main()
{
   initwindow(600, 450);

   create_control(RUP, 0, 400, "rup.bmp");
   create_control(RDOWN, 75, 400, "rdown.bmp");
   create_control(NUP,   150, 400, "nup.bmp");
   create_control(NDOWN, 225, 400, "ndown.bmp");
   create_control(CLEAR, 300, 400, "clear.bmp");
   create_control(SAVE,   400, 400, "save.bmp");
   create_control(EXIT,   500, 400, "exit.bmp");

   IMAGE *image;
   image = loadBMP("fon.bmp");

   int left = 0, top = 0, width = 600, height = 400;
   int n = 5, r = 50;

   putimage(0, 0, image, COPY_PUT);
   while (true)
   {
      while (mousebuttons() != 1)
      {
         if(mousey()+r < 400 && mousebuttons())
         {
            setcolor(COLOR(255, 0, 0));
            star(mousex(), mousey(), n, r);
            while(mousebuttons());
         }
      }
      switch (select_control())
      {
         case NONE: break;
         case NUP: if(n < 20) n++; break;
         case NDOWN: if(n > 5) n--; break;
         case RUP: if(r < 100) r += 10; break;
         case RDOWN: if(r > 10) r -= 10; break;
         case CLEAR: putimage(0, 0, image, COPY_PUT); break;
         case SAVE: save(); break;
         case EXIT: closegraph(); return 0;
      }
```

```
        while (mousebuttons());
    }
    freeimage(image);
}
```

Файл task.h

```
#ifndef TASK_H
#define TASK_H

#define COLOR_MAX 255

struct Point
{
   int x, y;
};

void lineB(int, int, int, int);
void star(int, int, int, int);
void save();

#endif
```

Файл task.cpp

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



void lineB(int x0, int y0, int x1, int y1)
{
   int color = getcolor();
   int deltax = abs(x1-x0);
   int deltay = abs(y1-y0);
   double error = 0;
   double deltaerr = 0;
   if(deltay > deltax)
   {
      if(deltay != 0) deltaerr = 1.0*deltax/deltay;
      if(y0 > y1)
      {
         int a = x0;
         x0 = x1;
         x1 = a;
         a = y0;
         y0 = y1;
         y1 = a;
      }
      int x = x0;
      int dir = x1-x0;
      if(dir > 0)
      {
         dir = 1;
```

```
        }
        else
        {
            dir = -1;
        }
        for(int y = y0; y <= y1; y++)
        {
            putpixel(x, y, color);
            error += deltaerr;
            if(error >= 1)
            {
                x += dir;
                error--;
            }
        }
    }
    else
    {
        if(deltax != 0) deltaerr = 1.0*deltay/deltax;
        if(x0 > x1)
        {
            int a = x0;
            x0 = x1;
            x1 = a;
            a = y0;
            y0 = y1;
            y1 = a;
        }
        int y = y0;
        int dir = y1-y0;
        if(dir > 0)
        {
            dir = 1;
        }
        else
        {
            dir = -1;
        }
        for(int x = x0; x <= x1; x++)
        {
            putpixel(x, y, color);
            error += deltaerr;
            if(error >= 1)
            {
                y += dir;
                error--;
            }
        }
    }
}

void star(int x, int y, int n, int r)
```

```
{
   Point *points;
   points = new Point[2*n];
   for(int i = 0; i < n*2; i++)
   {
      Point point;
      if(i%2)
      {
         point.x = x+int(r*cos(i*3.14/n));
         point.y = y+int(r*sin(i*3.14/n));
      }
      else
      {
         point.x = x+int(r*(cos(3.14/n*2)/cos(3.14/n))*cos(i*3.14/n))
         point.y = y+int(r*(cos(3.14/n*2)/cos(3.14/n))*sin(i*3.14/n))
      }
      points[i].x = point.x;
      points[i].y = point.y;
   }

   for(int i = 1; i < n*2; i++) lineB(points[i-1].x, points[i-1].y, p

   lineB(points[2*n-1].x, points[2*n-1].y, points[0].x, points[0].y);

   setfillstyle(SOLID_FILL, getcolor());
   floodfill(x, y, getcolor());

   delete[] points;
}

void save()
{
   int width, height;
   IMAGE *output;

   width  = getmaxx() + 1;
   height = getmaxy() + 1;
   output = createimage(width, height);

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

Файл control.h

```
#ifndef CONTROL_H
#define CONTROL_H

enum control_values { NONE = -1, EXIT, SAVE, CLEAR,
                      NUP, NDOWN, RUP, RDOWN,
```

```
                    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;
}
```

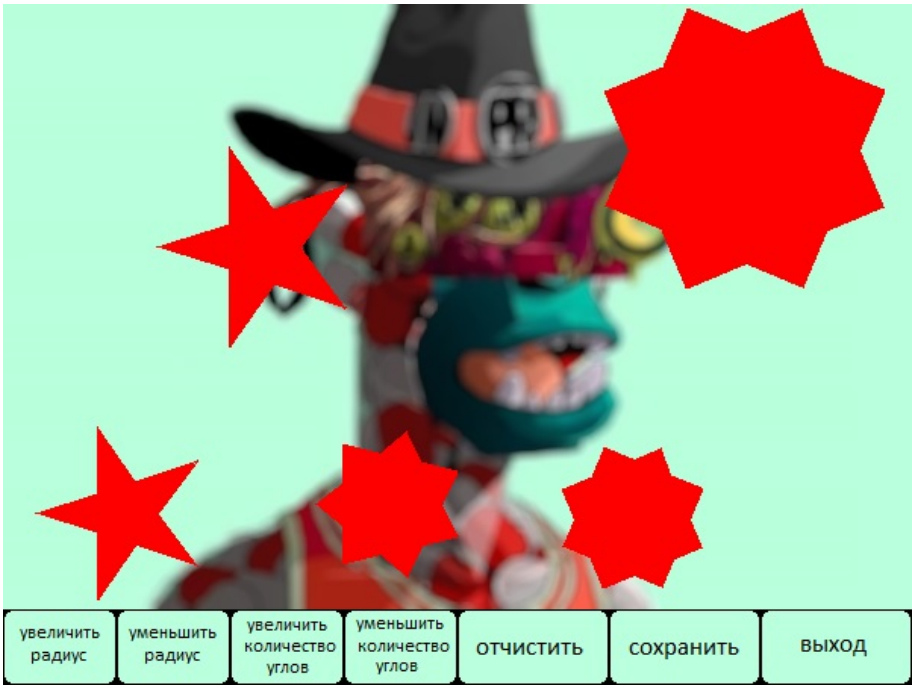

Рисунок 4.1 – Результат выполнения программы (Пример 1)