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-212
_____ Шафикова М.А.
«____» _____ 2022 г.

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

Челябинск 2022

1 ЗАДАНИЕ

1. Привести описание и схему алгоритма Брезенхема 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 $(x_0; y_0)$, $(x_n; y_n)$ – координаты начальной и конечной точки соответственно. Предварительно определяются горизонтальное и вертикальное расстояния Δ_x и Δ_y:

$$\Delta_x = |x_0 - x_n|\,,\ \Delta_y = |y_0 - y_n|\,.$$

Направление рисования определяется величинами s_x и s_y:

$$s_x = \begin{cases} 1,\ x_0 \le x_n; \\ -1,\ x_0 > x_n, \end{cases}$$

$$s_y = \begin{cases} 1,\ y_0 \le y_n; \\ -1,\ y_0 > y_n, \end{cases}$$

Если $\Delta_x \ge \Delta_y$, то число точек $n = \Delta_x + 1$. Начальное значение параметра $p_0 = 2\Delta_y - \Delta_x$. Координаты точек $(x_i; y_i)$, $i = 1,2,\ldots,n$ определяются рекуррентно по формулам:

$$x_i = x_{i-1} + s_x,$$

$$y_i = \begin{cases} y_{i-1} + s_y,\ p_{i-1} > 0; \\ y_{i-1},\ p_{i-1} \le 0, \end{cases}$$

$$p_i = \begin{cases} p_{i-1} + 2(\Delta_y - \Delta_x),\ p_{i-1} > 0; \\ p_{i-1} + 2\Delta_y,\ p_{i-1} \le 0. \end{cases}$$

Если $\Delta_x < \Delta_y$, то число точек $n = \Delta_y + 1$. Начальное значение параметра $p_0 = 2\Delta_x - \Delta_y$. Координаты точек $(x_i; y_i)$, $i = 1,2,\ldots,n$ определяются рекуррентно по формулам:

$$x_i = \begin{cases} x_{i-1} + s_x,\ p_{i-1} > 0; \\ x_{i-1},\ p_{i-1} \le 0, \end{cases}$$

$$y_i = y_{i-1} + s_y,$$

$$p_i = \begin{cases} p_{i-1} + 2(\Delta_x - \Delta_y),\ p_{i-1} > 0; \\ p_{i-1} + 2\Delta_x,\ p_{i-1} \le 0. \end{cases}$$

Файл main.cpp

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

using namespace std;

int main()
{
   initwindow(WIDTH,HEIGHT,"Stars");
   create_background("bg.jpg");

   create_control_mini(INCREASE_R,1,341);
   create_control_mini(DECREASE_R,80,420);
   create_control_mini(INCREASE_N,624,341);
   create_control_mini(DECREASE_N,548,418);

   create_control_maxi(EXIT,385,421);
   create_control_maxi(SAVE,184,423);

   canvas(CANVAS);

   int R=20, N=5;

   while(1)
   {
      while(mousebuttons()!=1);
      switch(select_control())
      {
         case CANVAS:
            draw_star(mousex(),mousey(),R,N);
            break;
         case INCREASE_R:
            R=R+5;
            show_number(R,0);
            delay(100);
            break;
         case DECREASE_R:
            if(R>5)
               R=R-5;
            show_number(R,0);
            delay(100);
            break;
         case INCREASE_N:
            N++;
            show_number(N,1);
            delay(100);
            break;
```

```
            case DECREASE_N:
               if(N>3)
                  N--;
               show_number(N,1);
               delay(100);
               break;
            case SAVE:
               save();
               break;
            case EXIT:
               return 0;

         }
      }
   }
```

Файл task.h

```
#ifndef TASK_H
#define TASK_H

void show_number(int,int);
void draw_star(int, int, int, int);
void save();

#endif
```

Файл task.cpp

```
#include <task.h>
#include <math.h>
#include <string>
#include <graphics.h>
#include <control.h>
#define M_PI acos(0.0)*2.0

using namespace std;

void create_line(int x1, int y1)
{
   int x0 = getx(), y0 = gety();
   int sx, sy, dx, dy, p;

   sx = x0 > x1 ? -1 : 1;
   sy = y0 > y1 ? -1 : 1;
   dx = abs(x1 - x0);
   dy = abs(y1 - y0);

   if (dx >= dy)
   {
      p = 2 * dy - dx;
      for (int i = x0; i != x1; i += sx)
      {
         putpixel(i, y0, getcolor());
         y0 = p > 0 ? y0 + sy : y0;
         p = p > 0 ? p + 2 * (dy - dx) : p + 2 * dy;
      }
   }
   else
   {
      p = 2 * dx - dy;
      for (int i= y0; i != y1; i += sy)
      {
         putpixel(x0, i, getcolor());
         x0 = p > 0 ? x0 + sx : x0;
         p = p > 0 ? p + 2 * (dx - dy) : p + 2 * dx;
```

```
      }
   }
    putpixel(x1, y1, getcolor());
}

void draw_star(int x0, int y0, int R, int n)
{
   setcolor(RED);
    double r = (R * cos(2 * M_PI / n)) / cos(M_PI / n);
    double da = M_PI / n;

    for (int k = 0; k < 2 * n + 1; k++)
    {
        int l = k % 2 == 0 ? R : r;
        double x = (x0 + l * cos(k * da));
        double y = (y0 + l * sin(k * da));
        if (k == 0) moveto(x, y);
        create_line(x, y);
        moveto(x, y);
    }
   delay(10);
   floodfill(x0, y0, getcolor());
}

void show_number(int n, int choose)
{
   setfillstyle(SOLID_FILL, BLACK);
   if(choose==0)
   {
      bar(20, 440, 74, 480);
      setcolor(WHITE);
      outtextxy(24, 444, to_string(n).c_str());
   }
   else
   {
      bar(634, 440, 700, 500);
      setcolor(WHITE);
      outtextxy(636, 444, to_string(n).c_str());
   }
}

void save()
{
   IMAGE *screen;
   screen=createimage(WIDTH+1, HEIGHT+1);
   getimage(0,0,WIDTH,HEIGHT,screen);
   saveBMP("screen.bmp", screen);
}
```

Файл control.h

```
#ifndef CONTROL_H
#define CONTROL_H
#define WIDTH 700
#define HEIGHT 500

enum control_values { NONE = -1, EXIT, SAVE, INCREASE_R, DECREASE_R,

struct Control
{
   int x0;
   int y0;
   int x1;
   int y1;
};


void create_control_mini(int,int,int);
void create_control_maxi(int,int,int);
void canvas(int);
void create_background(const char*);
int select_control();

#endif
```

Файл control.cpp

```
#include <graphics.h>
#include <control.h>

Control controls[N_CONTROLS];

void create_control_mini(int n,int x0,int y0)
{
   controls[n].x0=x0;
   controls[n].y0=y0;
   controls[n].x1=x0+75;
   controls[n].y1=y0+75;
}

void create_control_maxi(int n,int x0,int y0)
{
   controls[n].x0=x0;
   controls[n].y0=y0;
   controls[n].x1=x0+140;
   controls[n].y1=y0+75;
}

void canvas(int n)
{
   controls[n].x0=1;
   controls[n].y0=1;
```

```
    controls[n].x1=699;
    controls[n].y1=330;
}

void create_background(const char* image_name)
{
    IMAGE *image=loadBMP(image_name);
    putimage(0,0,image,COPY_PUT);
    freeimage(image);
}

int select_control()
{
    int x, y;

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

    for (int i = 0; i < N_CONTROLS; i++)
    {
        if (x > controls[i].x0 && x < controls[i].x1 &&
            y > controls[i].y0  && y < controls[i].y1)
        {
            return i;
        }
    }

    return NONE;
}
```

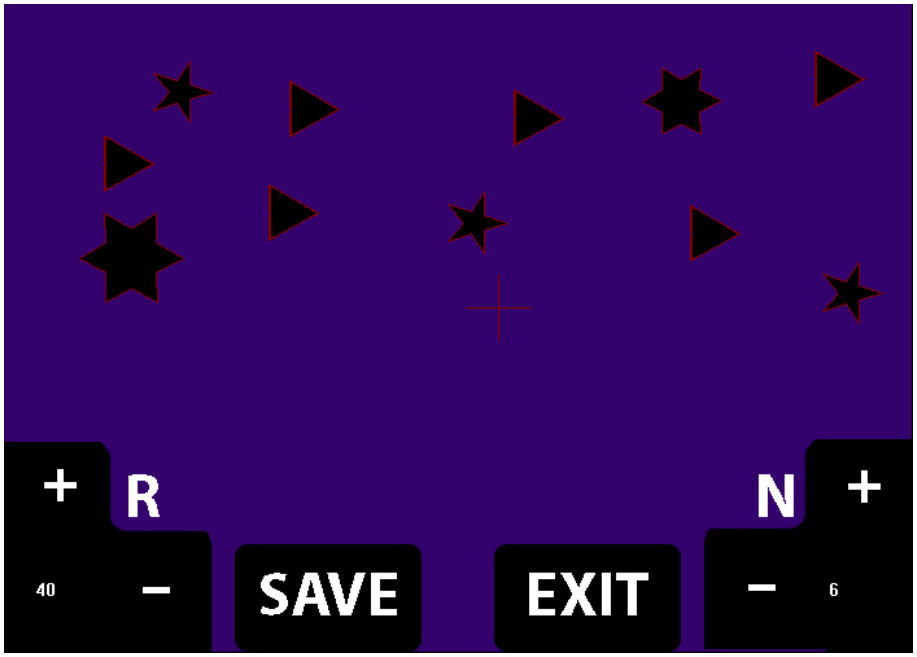

Рисунок 4.1 – Результат выполнения программы