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 № 4 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. Привести описание и схему алгоритма Брезенхема для растрового представления окружности.

2. Разработать подпрограмму для рисования окружности (аналог процедуры circle из графической библиотеки). Аргументы подпрограммы – координаты центра и радиус окружности. При реализации подпрограммы использовать для рисования только процедуру putpixel. Для определения текущего цвета рисования использовать функцию getcolor.

3. Разработать подпрограмму для создания фигуры, показанной на рисунке. При создании контура фигуры использовать собственную подпрограмму рисования окружности. Для закраски деталей фигуры использовать процедуру floodfill.

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

- изменение цвета деталей и фона;
- сохранение результата в файл;
- выход из программы.

2 МАТЕМАТИЧЕСКАЯ МОДЕЛЬ

Пусть x_0, y_0, r – соответственно координаты центра окружности и её радиус. При рисовании окружности цвет задан изначально color через $COLOR_{MAX}$, а x и y изменения координат пикселей.

Рисование окружности:

$$x = 0 \quad y = r.$$

$$delta = 3 - 2 * r.$$

$$mark[2] = [1, -1].$$

Пока x $\leq$ y выполняем следующие действия:

Перебираем i от 0 до 2 не включительно:

Перебираем j от 0 до 2 не включительно:

Ставим пиксель в точке x_0+mark[i]*x, y_0+mark[j]*y цвета color.

Ставим пиксель в точке x_0+mark[i]*x, y_0+mark[j]*y цвета color.

После всех переборов если p>0 то:

$$p = p + 4 * (x - y) + 10.y = y - 1$$

Иначе:

$$p = p + 4 * x - +6.$$

После чего x=x+1.

3 ТЕКСТ ПРОГРАММЫ

Файл main.cpp

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

int main(){
   initwindow(800, 600);
   IMAGE *image;
   image = loadBMP("fon.jpg");
   putimage(0, 0, image, COPY_PUT);
   freeimage(image);
   create_control(CREATE_ROUNDS, 0, 550,"create.jpg");
   create_control(FILL, 200, 550,"fill.jpg");
   create_control(SAVE, 400, 550,"save.jpg");
   create_control(EXIT, 600, 550,"exit.jpg");
   while(true){
      while(mousebuttons() != 1);
      switch(select_control()){
         case NONE: break;
         case CREATE_ROUNDS: circles(500, 300, 100); break;
         case FILL: fill(500, 300); break;
         case SAVE: save(); break;
         case EXIT: closegraph(); return 0;
      }
   }
}
```

Файл task.h

```
#ifndef TASK_H
#define TASK_H

#define COLOR_MAX WHITE

void circ(int, int, int);
void circles(int, int, int);
void fill(int, int);
void save();

#endif
```

Файл task.cpp

```
#define _USE_MATH_DEFINES
#include "graphics.h"
#include "task.h"
#include "cstdlib"
#include "ctime"
#include <cmath>


bool is_drawed = false;

void circ(int x0, int y0, int r){
   int color = COLOR_MAX;
   int x = 0, y = r, p = 3 - 2 * r;
   int mark[2] = {1, -1};
   while(x <= y){
      for(int i = 0; i < 2; i++){
         for(int j=0; j < 2;j++){
            putpixel(x0 + mark[i] * x, y0 + mark[j] * y,color);
            putpixel(x0 + mark[i] * y, y0 + mark[j] * x,color);
         }
      }
      if(p>0){
         p+= 4 * (x - y) + 10;
         y--;
      }
      else{
         p += 4 * x + 6;
      }
      x++;
   }
}

void circles(int x0, int y0, int r){
   for(int i = 0; i<12; i++){
```

```
        circ(x0+r*cos((M_PI/6)*i), y0+r*sin((M_PI/6)*i),  r);
    }
    is_drawed = true;
}

void fill(int x0, int y0){
    if(is_drawed){
        while(mousebuttons()!=0);
        srand(time(0));
        int color[6];
        int r[6]= {25, 50, 80, 135, 160, 185};
        double d = M_PI / 6, d1 = d / 2;
        int x,y;
        for (int i=0; i < 6; i++){
            color[i]=rand()%WHITE;
        }
        for (int i=0; i<6; i++){
            setfillstyle(SOLID_FILL,color[i]);
            for (int j=0; j<12; j++){
                x=x0+r[i]*cos(j*d+d1*((i+1)%2));
                y=y0+r[i]*sin(j*d+d1*((i+1)%2));
                floodfill(x,y,COLOR_MAX);
            }
        }
    }
}

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.jpg", output);
    freeimage(output);
}
```

Файл control.h

```
#ifndef CONTROL_H
#define CONTROL_H

enum control_values { NONE = -1, EXIT, SAVE,
                      CREATE_ROUNDS, FILL,
                      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;
```

```
}
```

4 РЕЗУЛЬТАТ РАБОТЫ

Рисунок 4.1 – Результат выполнения программы (Пример 1)

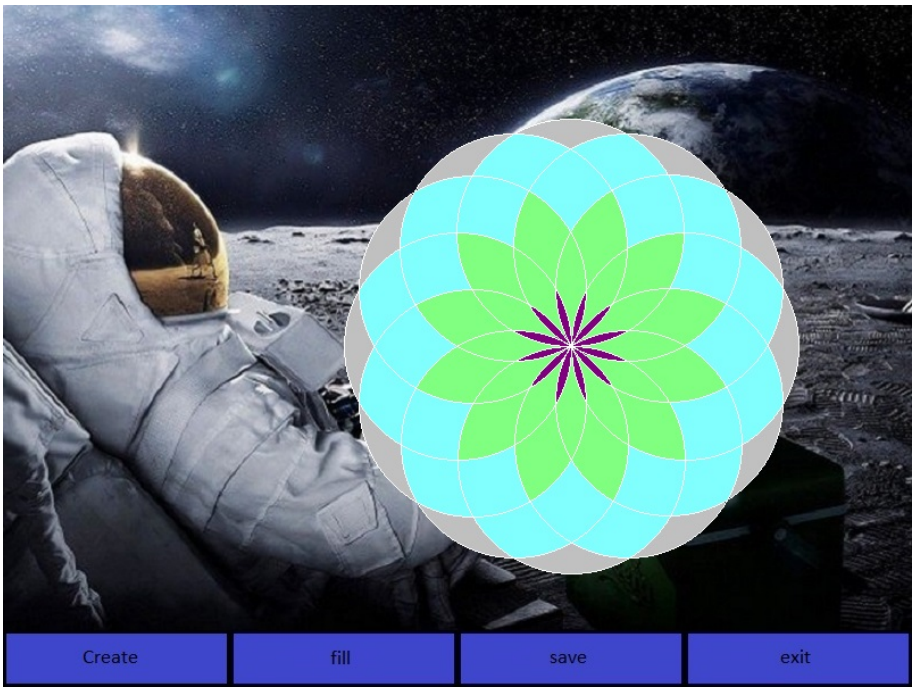

Рисунок 4.2 – Результат выполнения программы (Пример 2)

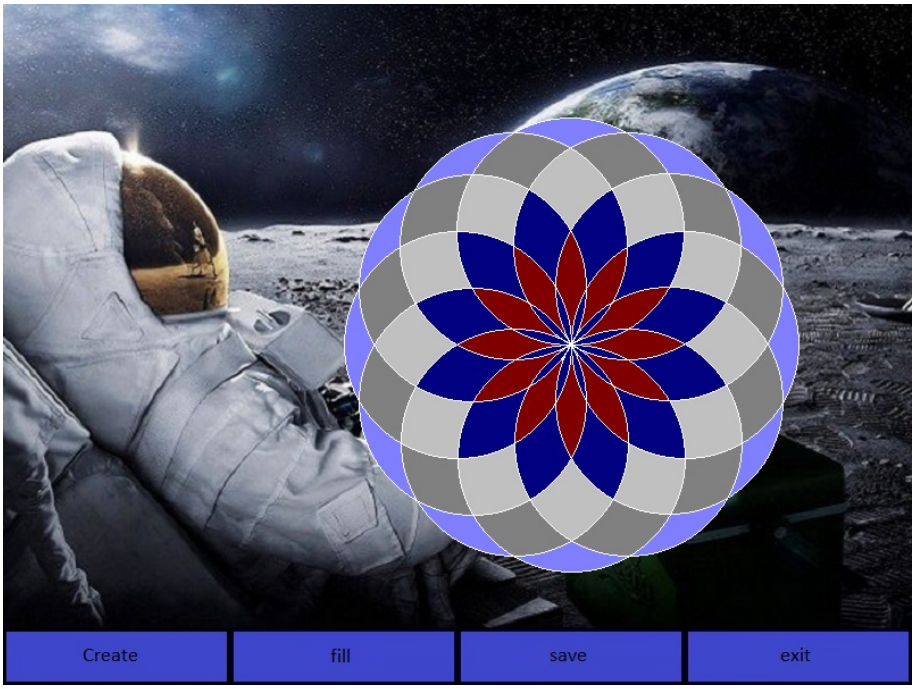

Рисунок 4.3 – Результат выполнения программы (Пример 3)