

### МИНОБРНАУКИ РОССИИ

### федеральное государственное бюджетное образовательное учреждение высшего образования

### «Московский государственный технологический университет «СТАНКИН» (ФГБОУ ВО «МГТУ «СТАНКИН»)

### Институт информационных систем и технологий

### Кафедра инженерной графики

**Отчет по выполнению лабораторной работы №2**

## по дисциплине «Геометрическое моделирование и компьютерная графика»

## Направление подготовки: 09.03.03 «Прикладная информатика»

## Профиль: «Управление данными»

## Руководитель: Толок А.В.

Оценка Подпись Дата

## Студент:

## Баканов П.О.

## Группа ИДБ-20-11

## 

Подпись Дата

## Москва, 2023

# Задание

Общее задание для всех (**25 баллов**):

* Визуализировать М-образ сложной фигуры (более 5 функций)
* Сделать переключатель М-Образов. Чтобы можно было из GUI менять визуализируемый образ
* Добавить сохранение изображения на диск через GUI

# Код

float figure(const sf::Vector2f &point)

{

std::function<float(const sf::Vector2f &)> rFuncs[6];

rFuncs[0] = [](const sf::Vector2f &point) -> float { return std::pow(point.x - 0, 2) + std::pow(point.y + 4, 2) - 4; };

rFuncs[1] = [](const sf::Vector2f &point) -> float { return std::pow(point.x + 3, 2) + std::pow(point.y - 0, 2) - 4; };

rFuncs[2] = [](const sf::Vector2f &point) -> float { return std::pow(point.x - 8, 2) + std::pow(point.y - 0, 2) - 6; };

rFuncs[3] = [](const sf::Vector2f &point) -> float { return std::pow(point.x + 6, 2) + std::pow(point.y - 0, 2) - 1; };

rFuncs[4] = [](const sf::Vector2f &point) -> float { return std::pow(point.x - 10, 2) + std::pow(point.y - 0, 2) - 9; };

rFuncs[5] = [](const sf::Vector2f &point) -> float { return std::pow(point.x - 9, 2) + std::pow(point.y - 0, 2) - 9; };

float res1 = Functions::ROr(rFuncs[0](point), rFuncs[1](point));

float res2 = Functions::ROr(res1, rFuncs[2](point));

float res3 = Functions::ROr(res2, rFuncs[3](point));

float res4 = Functions::RAnd(rFuncs[4](point), rFuncs[5](point));

float res = Functions::ROr(res3, res4);

return res;

}

class RFuncSprite : public sf::Sprite

{

public:

void create(const sf::Vector2u &size, Shapes shapeType);

void drawRFunc(const std::function<float(const sf::Vector2f &)> &rfunc, const sf::FloatRect &subSpace);

const sf::Texture &getTexture() const { return \_texture; }

const sf::Image &getImage() const { return \_image; }

void saveToFile(const std::string &filename) { \_image.saveToFile(filename); }

private:

sf::Texture \_texture;

sf::Image \_image;

sf::Color \_firstColor = sf::Color::Black;

sf::Color \_secondColor = sf::Color::White;

Shapes \_shapeType;

float calculateDeterminant(const std::vector<std::vector<float>> &matrix);

sf::Color interpolateColors(const sf::Color &colorFirst, const sf::Color &colorSec, float k);

sf::Color getPixelColor(float Ai, float Bi, float Ci, float Di);

};

sf::Color RFuncSprite::getPixelColor(float Ai, float Bi, float Ci, float Di)

{

float nx, ny, nz, nw;

const float det = std::sqrt(Ai \* Ai + Bi \* Bi + Ci \* Ci + Di \* Di);

nx = Ai / det;

ny = Bi / det;

nz = Ci / det;

nw = Di / det;

float shapeValue;

switch (\_shapeType)

{

case Nx:

shapeValue = nx;

break;

case Ny:

shapeValue = ny;

break;

case Nz:

shapeValue = nz;

break;

case Nw:

shapeValue = nw;

break;

default:

throw std::runtime\_error("Wrong shape");

}

return interpolateColors(\_firstColor, \_secondColor, (1.f + shapeValue) / 2);

}

float RFuncSprite::calculateDeterminant(const std::vector<std::vector<float>> &matrix)

{

return matrix[0][0] \* (matrix[1][1] \* matrix[2][2] - matrix[1][2] \* matrix[2][1]) -

matrix[0][1] \* (matrix[1][0] \* matrix[2][2] - matrix[1][2] \* matrix[2][0]) +

matrix[0][2] \* (matrix[1][0] \* matrix[2][1] - matrix[1][1] \* matrix[2][0]);

}

void RFuncSprite::create(const sf::Vector2u &size, Shapes shapeType)

{

\_image.create(size.x, size.y, sf::Color::Cyan);

\_texture.loadFromImage(\_image);

setTexture(\_texture);

\_shapeType = shapeType;

}

sf::Color RFuncSprite::interpolateColors(const sf::Color &colorFirst, const sf::Color &colorSec, float k)

{

sf::Color val;

val.r = static\_cast<sf::Uint8>(colorFirst.r + (colorSec.r - colorFirst.r) \* k);

val.g = static\_cast<sf::Uint8>(colorFirst.g + (colorSec.g - colorFirst.g) \* k);

val.b = static\_cast<sf::Uint8>(colorFirst.b + (colorSec.b - colorFirst.b) \* k);

val.a = static\_cast<sf::Uint8>(colorFirst.a + (colorSec.a - colorFirst.a) \* k);

return val;

}

# Результаты работы

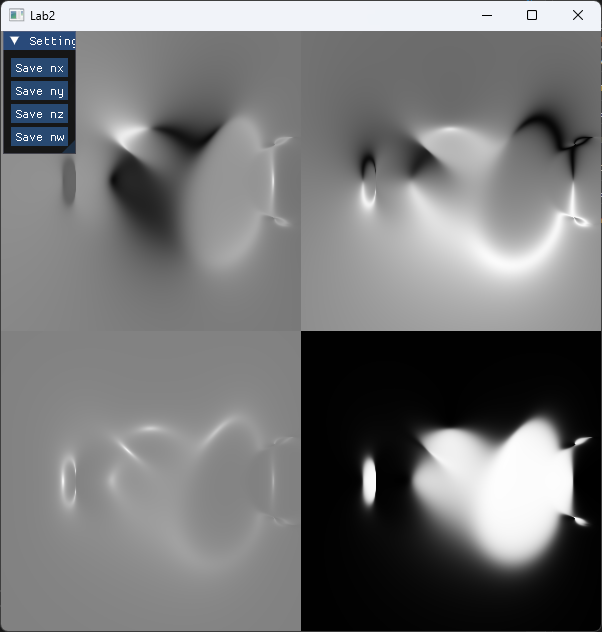


Рис1. Основное окно