**Первая лабораторная: задания 1-7.**

**1. Работа с изображениями.**

Выбрать язык программирования и библиотеку для записи изображений в файл.

Создать матрицу размера H\*W, заполнить её элементы нулевыми значениями, сохранить в виде полутонового (одноканального) 8-битового изображения высотой H и шириной W, убедиться, что полученное изображение открывается средствами операционной системы и полностью чёрное.

Создать матрицу размера H\*W, заполнить её элементы значениями, равными 255, сохранить в виде полутонового (одноканального) 8-битового изображения высотой H и шириной W, убедиться, что полученное изображение открывается средствами операционной системы и полностью белое.

Создать матрицу размера H\*W\*3, заполнить её элементы значениями, равными (255, 0, 0), сохранить в виде цветного (трёхканального) 8-битового изображения высотой H и шириной W, убедиться, что полученное изображение открывается средствами операционной системы и полностью красное.

Создать матрицу размера H\*W\*3, заполнить её элементы произвольными значениями по выбранной схеме (например, значение элемента равно сумме его координат по модулю 256), сохранить в виде 8-битового изображения высотой H и шириной W, убедиться, что полученное изображение открывается средствами операционной системы (в предложенном примере должен получиться плавный градиент от чёрного цвета в верхнем левом углу изображения).

**2. Работа с изображениями (ООП)**

Создать класс для хранения изображения в памяти в виде массива.

Ниже приведён код для примера. Конкретная реализация и способ хранения изображения на ваше усмотрение.

class Color3 {

public:

float r, g, b;

<...>

}

class Image {

private:

int m\_width;

int m\_height;

std::vector<Color3> m\_data;

public:

Image(int width, int height):

m\_width(width), m\_height(height), m\_data(width\*height) {};

int width() const { return m\_width; }

int height() const { return m\_height; }

void set(int x, int y, const Color3& value) {

m\_data[x + y \* m\_width] = value;

}

const Color3& get(int x, int y) const {

return m\_data[x + y \* m\_width];

}

void save(const std::string& filename) const {

<...>

}

};

**3. Отрисовка прямых линий**

Реализовать все описанные в лекциях алгоритмы отрисовки прямых (до алгоритма Брезенхема включительно).

Для каждого алгоритма сохранить в файл изображение размера 200х200 с нарисованной на нём «звездой» (см. лекции).

(линии из точки в точки .

**4. Работа с трёхмерной моделью (вершины)**

Создать класс, содержащий трёхмерную модель в виде массив трёхмерных координат точек (для может потребоваться создать класс трёхмерных координат).

Считать из приложенного файла obj строки, содержащие информацию о вершинах модели в объект созданного класса:

v X1 Y1 Z1

v X2 Y2 Z2

<…>

**5. Отрисовка вершин трёхмерной модели**

Нарисовать вершины модели (игнорируя координату Z) на изображении размером (1000, 1000).

Преобразования координат точек для эксперимента:

[50 \* X + 500, 50 \* Y + 500]

[100 \* X + 500, 100 \* Y + 500]

[500 \* X + 500, 500 \* Y + 500]

[4000 \* X + 500, 4000 \* Y + 500]

**6. Работа с трёхмерной моделью (полигоны)**

Считать из приложенного файла строки, содержащие информацию о полигонах модели.

Сведения о полигонах в файле хранятся в формате:

f **v1**/vt1/vn1 **v2**/vt2/vn2 **v3**/vt3/vn3

В рамках лабораторной загрузить в память необходимо только первые значения в каждой тройке – номера вершин, загруженных ранее.

Обратите внимание, что вершины нумеруются, начиная с единицы.

**7. Отрисовка рёбер трёхмерной модели**

Отрисовать все рёбра всех полигонов модели с помощью алгоритма Брезенхема (координаты вершин округляем до ближайшего целого).