Первая лабораторная: задания 1-7.

Вторая лабораторная: задания 8-11 + задания 12-15.

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 200x200 с нарисованной на нём «звездой» (см. лекции).

(линии из точки $(100,100)$ в точки $(100 + 95\ cos\ (\alpha)\ ,100 + 95\ sin\ (\alpha)\ ), \alpha = \frac{2\pi i}{13}, i = \overline{0,12}$.

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).

8. Барицентрические координаты

Написать функцию вычисления барицентрических координат для точки с экранными (целочисленными координатами) *(x,y)* относительно вещественных вершин треугольника *(x0, y0)*, *(x1, y1)* и *(x2, y2)*.

Они вычисляются по формулам:

```
lambda0 = ((x1 - x2)*(y - y2) - (y1 - y2)(x - x2)) / ((x1 - x2)*(y0 - y2) - (y1 - y2) (x0 - x2))

lambda1 = ((x2 - x0)*(y - y0) - (y2 - y0)(x - x0)) / ((x2 - x0)*(y1 - y0) - (y2 - y0) (x1 - x0))

lambda2 = ((x0 - x1)*(y - y1) - (y0 - y1)(x - x1)) / ((x0 - x1)*(y2 - y1) - (y0 - y1) (x2 - x1))
```

Убедиться, что сумма барицентрических координат

```
lambda0 + lambda1 + lambda2 = 1.0
```

Обратите внимание, что координаты *(x, y)* – экранные, и поэтому целочисленные. В то же время вершины треугольника *(x0, y0)*, *(x1, y1)* и *(x2, y2)* – вещественные, округлять их перед вычислениями **не надо**.

С точки зрения написания и выполнения программы это не оказывает большого влияния, но важно для понимания. Барицентрические координаты в рамках этого курса будут описывать положение **пикселя**, который вы собираетесь отрисовать, относительно **реального** треугольника.

9. Отрисовка треугольников

Написать функцию отрисовки треугольника с вершинами *(x0, y0)*, *(x1, y1)* и *(x2, y2)*. Для этого выполнить следующие шаги.

1. Определить ограничивающий прямоугольник: минимальные и максимальные возможные значения координат X и Y. Например:

```
xmin = min(x0, x1, x2)

ymin = min(y0, y1, y2)

xmax = max(x0, x1, x2)

ymax = max(y0, y1, y2)
```

2. Для ограничивающего прямоугольника учесть границы изображения, так, например:

```
if (xmin < 0): xmin = 0
```

Разумеется, вы можете сделать 1 и 2 пункты одновременно.

3. Для каждого пикселя внутри ограничивающего прямоугольника вычислить барицентрические координаты относительно вершин треугольника.

Если **все** барицентрические координаты пикселя больше нуля – пиксель рисуется, иначе – переходим к следующему.

Обратите внимание, что рёбра треугольника линиями (как в задании 7) рисовать **не надо**.

10. Тестирование функции

Протестировать функцию отрисовки треугольника для разных треугольников, в том числе, частично (или полностью) выходящих за пределы изображения.

11. Отрисовка полигонов трёхмерной модели

Нарисовать все полигоны модели разными цветами (для одного треугольника – один случайный цвет).

12. Вычисление нормали к поверхности треугольника

Для каждого треугольника вычислить нормаль к этому треугольнику по формуле:

$$\bar{n} = [X_1 - X_0 \quad Y_1 - Y_0 \quad Z_1 - Z_0] \times [X_1 - X_2 \quad Y_1 - Y_2 \quad Z_1 - Z_2],$$

где (X_0, Y_0, Z_0), (X_1, Y_1, Z_1) и (X_2, Y_2, Z_2) – **исходные** координаты вершин треугольника (до любых преобразований), а $\times$ – векторное произведение.

Координаты нормали могут быть вычислены через определитель:

$$\bar{n} = \begin{vmatrix} \bar{\imath} & \bar{\jmath} & \bar{k} \\ X_1 - X_0 & Y_1 - Y_0 & Z_1 - Z_0 \\ X_1 - X_2 & Y_1 - Y_2 & Z_1 - Z_2 \end{vmatrix}.$$

13. Отсечение нелицевых граней

Для каждого треугольника определить косинус угла падения направленного света (считать направление света равным $\bar{l} = [0, 0, 1]$) через нормализованное скалярное произведение $\frac{\langle \bar{n}, \bar{l} \rangle}{\|\bar{n}\| \cdot \|\bar{l}\|}$.

Изменить цикл отрисовки полигонов таким образом, чтобы отрисовывались только полигоны с $\frac{\langle \bar{n}, \bar{l} \rangle}{\|\bar{n}\| \cdot \|\bar{l}\|} < 0$.

14. Базовое освещение

Отрисовку полигонов выполнять не случайным цветом, а пропорциональным косинусу угла между $\bar{n}$ и $\bar{l}$, например $\left(255 * \frac{\langle \bar{n}, \bar{l} \rangle}{\|\bar{n}\| \cdot \|\bar{l}\|},\ 0,\ 0\right)$.

15. z-буфер

При отрисовке полигонов проверять перекрытие полигонов с использованием z-буфера.

z-буфер – это матрица из вещественных значений по размеру совпадающая с изображением. Все элементы z-буфера изначально инициализируются некоторым достаточно большим значением.

При отрисовке для каждой точки выполняется следующая проверка:

1. Вычисляются барицентрические координаты $(\lambda_0,\ \lambda_1,\ \lambda_2)$.

2. Если все барицентрические координаты больше нуля, вычисляем z-координату **исходного** полигона через **исходные** z-координаты вершин этого полигона:

$\hat{z} = \lambda_0 z_0 + \lambda_1 z_1 + \lambda_2 z_2$.

3. Если вычисленное значение координаты $\hat{z}$ больше координаты z-буфера для текущего пикселя, пропускаем точку.

4. Если вычисленное значение координаты $\hat{z}$ меньше координаты z-буфера для текущего пикселя, отрисовываем этот пиксель, а соответствующему элементу z-буфера присваиваем значение $\hat{z}$.