

Первая лабораторная: задания 1-7.

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

Третья лабораторная: задание 16 + задание 17.

Четвёртая лабораторная: задание 18.

1. Работа с изображениями.

Выбрать язык программирования и библиотеку для записи изображений в файл.

Создать матрицу размера $H \times W$, 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 $H \times W$, 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 $H \times W \times 3$, 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 $H \times W \times 3$, 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 $(100 + 95 \cos(\alpha), 100 + 95 \sin(\alpha))$, $\alpha = \frac{2\pi i}{13}$, $i = \underline{0,12}$).

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. Бариецентрические координаты

Написать функцию вычисления барицентрических координат для точки с экранными (целочисленными координатами) (x, y) относительно вещественных вершин треугольника (x_0, y_0) , (x_1, y_1) и (x_2, y_2) .

Они вычисляются по формулам:

$$\text{lambda0} = ((x_1 - x_2) * (y - y_2) - (y_1 - y_2) * (x - x_2)) / ((x_1 - x_2) * (y_0 - y_2) - (y_1 - y_2) * (x_0 - x_2))$$

$$\text{lambda1} = ((x_2 - x_0) * (y - y_0) - (y_2 - y_0) * (x - x_0)) / ((x_2 - x_0) * (y_1 - y_0) - (y_2 - y_0) * (x_1 - x_0))$$

$$\text{lambda2} = ((x_0 - x_1) * (y - y_1) - (y_0 - y_1) * (x - x_1)) / ((x_0 - x_1) * (y_2 - y_1) - (y_0 - y_1) * (x_2 - x_1))$$

Убедиться, что сумма барицентрических координат

$$\text{lambda0} + \text{lambda1} + \text{lambda2} = 1.0$$

Обратите внимание, что координаты (x, y) – экранные, и поэтому целочисленные. В то же время вершины треугольника (x_0, y_0) , (x_1, y_1) и (x_2, y_2) – 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 (x_0, y_0) , (x_1, y_1) и (x_2, y_2) . Для этого выполнить следующие шаги.

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

$$x_{\min} = \min(x_0, x_1, x_2)$$

$$y_{\min} = \min(y_0, y_1, y_2)$$

$$x_{\max} = \max(x_0, x_1, x_2)$$

$$y_{\max} = \max(y_0, y_1, y_2)$$

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:

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

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} .

16. Проективное преобразование

Заменить преобразование к экранным координатам

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

на проективное преобразование:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \sim \begin{bmatrix} a_x & 0 & u_0 \\ 0 & a_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

где (u, v) – экранные координаты, a_x, a_y – масштаб, (u_0, v_0) – центр изображения.

Чтобы проективное преобразование к координате Z всех точек модели необходимо добавить фиксированное значение (не менее 0.1).

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \sim \begin{bmatrix} a_x & 0 & u_0 \\ 0 & a_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \left(\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ t_z \end{bmatrix} \right)$$

Для теста можно воспользоваться следующими параметрами:

$$\mathbf{t} = \begin{bmatrix} 0.005 \\ -0.045 \\ 15.0 \end{bmatrix}, \mathbf{K} = \begin{bmatrix} 10000 & 0 & 500 \\ 0 & 10000 & 500 \\ 0 & 0 & 1 \end{bmatrix}$$

17. Поворот модели:

Реализовать поворот модели. Для этого исходные координаты вершин заменить на преобразованные:

$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = R \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

где

$$R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \alpha & \sin \alpha \\ 0 & -\sin \alpha & \cos \alpha \end{bmatrix} \cdot \begin{bmatrix} \cos \beta & 0 & \sin \beta \\ 0 & 1 & 0 \\ -\sin \beta & 0 & \cos \beta \end{bmatrix} \cdot \begin{bmatrix} \cos \gamma & \sin \gamma & 0 \\ -\sin \gamma & \cos \gamma & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

α, β, γ – углы поворота вокруг осей X, Y и Z , соответственно.

Для начала рекомендуется попробовать реализовать поворот вокруг оси Y . Такой поворот проще всего визуально отследить и проверить.

18. Затенение Гуро.

Реализовать продвинутое освещение посредством метода затенения Гуро.

Для этого необходимо будет использовать нормали ко всем вершинам модели (можно загрузить из модели либо вычислить, усреднив нормали ко всем полигонам, соседним с вершиной).

Для каждой вершины вычислить освещение этой вершины (как в задании 14).

Для каждой точки полигона вычислять освещение через барицентрические координаты относительно вершин этого полигона:

Пусть l_0, l_1, l_2 – соответственно равны

$$l_0 = \frac{\langle \bar{n}_0, \bar{l} \rangle}{\|\bar{n}_0\| \cdot \|\bar{l}\|}, \quad l_1 = \frac{\langle \bar{n}_1, \bar{l} \rangle}{\|\bar{n}_1\| \cdot \|\bar{l}\|}, \quad l_2 = \frac{\langle \bar{n}_2, \bar{l} \rangle}{\|\bar{n}_2\| \cdot \|\bar{l}\|},$$

где \bar{l} – направление света, а $\bar{n}_0, \bar{n}_1, \bar{n}_2$ – нормали соответствующих вершин треугольника.

Тогда значение яркости пиксела равно

$$255 * (\lambda_0 l_0 + \lambda_1 l_1 + \lambda_2 l_2)$$