Предварительные замечания:

1. Описание алгоритма состоит из описания четырех нижеприведенных процедур.

2. Предполагается, что к началу исполнения алгоритма был произведен переход к экранной системе координат и проведена операция кадрирования, в ходе которой координата z каждой точки (координата в экранной системе координат) не отбрасывается, а остается неизменной. Т. е. после перехода к экранной системе координат выполняется преобразование:
$$\begin{bmatrix} \chi' \\ \gamma' \\ \zeta' \\ \alpha' \end{bmatrix} = \begin{bmatrix} W_x/2 & 0 & 0 & W_{cx} + W_x/2 \\ 0 & -W_y/2 & 0 & W_{cy} - W_y/2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \chi \\ \gamma \\ \zeta \\ \alpha \end{bmatrix}$$

3. В описании алгоритма подразумевается, что область рисования имеет координаты по x от W_{cx} до $W_{cx} + W_x$, по y от W_{cy} до $W_{cy} + W_y$, где W_{cx}, W_x, W_{cy}, W_y — неотрицательны.

4. Список AEL — список четверок, первый элемент которых — идентификатор многоугольника, а остальные — вещественные числа.

5. При сравнении значений координаты y на равенство следует округлять эти значения до целого числа.

6. При рассмотрении ребер многоугольника $[(x_1, y_1, z_1), (x_2, y_2, z_2)]$ считаем, что $y_1 \leqslant y_2$ (начальная точка ребра находится не ниже конечной), а при равенстве $y_1 = y_2$ выполняется $x_1 \leqslant x_2$ (начальная точка ребра находится не левее конечной).

Алгоритм 1: Интервальный алгоритм построчного сканирования (Алгоритм Уоткинса)

Вход: $\mathscr{P}$ — список многоугольников трехмерной сцены

начало алгоритма

- Для каждого многоугольника в $\mathscr{P}$ вычислить $y_{P\min}$ и $y_{P\max}$ — значения минимальной и максимальной координаты y для вершин многоугольника;
- Удалить из $\mathscr{P}$ все многоугольники, у которых $y_{P\min} > W_{cy} + W_y$ или $y_{P\max} < W_{cy}$;
- Определить значение $y_{P\,next}$ — минимальное значение $y_{P\min}$ для многоугольников в $\mathscr{P}$;
- $y_t = y_{P\,next}$;
- **цикл пока** $y_t \leqslant W_{cy} + W_y$ **выполнять**
  - **если** $y_t = y_{P\,next}$, **то**
    - Занести в список APL все многоугольники из $\mathscr{P}$, для которых $y_{P\min} = y_t$, удалив их из $\mathscr{P}$;
    - Для каждого нового многоугольника P_j в APL установить значение $Active(P_j) = False$ и определить значения пятерки $(C_j, a_j, b_j, c_j, d_j)$, где C_j — цвет многоугольника, а a_j, b_j, c_j, d_j — коэффициенты уравнения несущей плоскости, полученные с помощью процедуры InitializePolygon(P_j). Если $c_j = 0$ — удалим многоугольник из APL;
    - $y_{APL\,next} = y_t$;
    - **если** *список* $\mathscr{P}$ *пуст,* **то** присвоить $y_{P\,next} = \infty$;
    - **иначе**
      - Определить значение $y_{P\,next}$ — минимальное значение $y_{P\min}$ для многоугольников в $\mathscr{P}$
  - **если** $y_t = y_{APL\,next}$, **то**
    - Добавить в AEL все четверки $\left(P, x_1, y_2, \dfrac{x_2 - x_1}{y_2 - y_1}\right)$, составленные для ребер многоугольников из APL, у которых $y_1 = y_t$ и $y_1 \neq y_2$;
    - Добавить в AEL все четверки $(P, x_1, y_2, 0)$, составленные для ребер многоугольников из APL, у которых $y_1 = y_t$ и $y_1 = y_2$;
    - Для ребер многоугольников в APL найти $y_{APL\,next}$ — минимальное значение y_1 такое, что $y_1 > y_t$. Если таких ребер нет в APL, то присвоить $y_{APL\,next} = \infty$;
    - Упорядочить AEL по значению 2-го элемента четверки;
    - Найти $y_{AEL\,next}$ — минимальное значение третьего элемента в четверках в AEL;
  - **если** $y_t \geqslant W_{cy}$ **то** выполнить процедуру ProcessLine ;
  - $y_t = y_t + 1$;
  - **если** $y_t \geqslant y_{AEL\,next}$, **то**
    - удалить из AEL четверки с третьим элементом меньшим либо равным y_t;
    - Обновить значение $y_{AEL\,next}$;
  - В каждой четверке $(P_j, x_j, y_j, \Delta_j x)$ в AEL заменить x_j на $x_j + \Delta_j x$;

конец алгоритма

Алгоритм 2: ProcessLine

начало алгоритма

- $x_{left} = -\infty$; $x_{right} = W_{cx}$ $polygonCount = 0$; $i = 1$;
- **цикл пока** *не достигнут конец списка* AEL *и* $x_{left} < W_{cx} + W_x$ **выполнять**
  - Пусть $(P_i, x_i, y_i, \Delta_i x)$ — очередной элемент AEL;
  - $x_{right} = x_i$;
  - **если** $x_{right} > W_{cx}$ **то**
    - **если** $x_{left} < W_{cx}$ **то** присвоить $x_{left} = W_{cx}$;
    - **если** $x_{right} > W_{cx} + W_x$ **то** присвоить $x_{right} = W_{cx} + W_x$;
    - **если** $polygonCount = 0$ **то** присвоить цвет фона переменной C;
    - **иначе если** $polygonCount = 1$ **то** присвоить $C = C_i$ — цвет многоугольника P_i;
    - **иначе**
      - Присвоить $intersectionStack = \varnothing$;
      - **повторять вычисления**
        - **если** $intersectionStack \neq \varnothing$ **то**
          - Начертить отрезок $[(x_{left}, y_0), (x_{right}, y_0)]$ цветом C;
          - Присвоить $x_{left} = x_{right}$;
          - Извлечь значение из стека $intersectionStack$ и присвоить его переменной x_{right};
        - **цикл пока** HasIntersection$(x_{left}, x_{right}, x_{int})$ **выполнять**
          - Занести x_{right} в стек $intersectionStack$;
          - Присвоить $x_{right} = x_{int}$;
        - Для всех P, таких, что $Active(P)$, найти z_{mid}

$$z_{mid} = -\frac{a(x_{left} + x_{right}) + 2by_t + 2d}{2c};$$

        - Присвоить переменной C цвет многоугольника с максимальным z_{mid};
      - **пока** $intersectionStack \neq \varnothing$;
  - Изменить $Active(P_i) = !Active(P_i)$;
  - **если** $Active(P_i)$ **то** $polygonCount = polygonCount + 1$;
  - **иначе** $polygonCount = polygonCount - 1$;
  - Начертить отрезок $[(x_{left}, y_0), (x_{right}, y_0)]$ цветом C;
  - Присвоить $x_{left} = x_{right}$, $i = i + 1$;
- **если** $x_{left} < W_{cx} + W_x$ **то**
  - **если** $x_{left} < W_{cx}$ **то** присвоить $x_{left} = W_{cx}$;
  - Присвоить $x_{right} = W_{cx} + W_x$;
  - Начертить отрезок $[(x_{left}, y_0), (x_{right}, y_0)]$ цветом фона;

конец алгоритма

Алгоритм 3: InitializePolygon Вычисление коэффициентов уравнения плоскости

Вход: P — список вершин многоугольника в трехмерном пространстве

Выход: a, b, c, d — коэффициенты уравнения несущей плоскости

начало алгоритма

Пусть (x_1, y_1, z_1), (x_2, y_2, z_2), (x_3, y_3, z_3) — первые три вершины в списке P. Предполагаем, что эти вершины не лежат на одной прямой. Тогда

$$a = \begin{vmatrix} 1 & y_1 & z_1 \\ 1 & y_2 & z_2 \\ 1 & y_3 & z_3 \end{vmatrix}; \quad b = \begin{vmatrix} x_1 & 1 & z_1 \\ x_2 & 1 & z_2 \\ x_3 & 1 & z_3 \end{vmatrix}; \quad c = \begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix}; \quad d = -\begin{vmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \end{vmatrix}.$$

конец алгоритма

Алгоритм 4: HasIntersection Проверка наличия пересечений на интервале

Вход: x_{left}, x_{right}. Параметр x_{int} может использоваться для возвращения одного из результатов.

Выход: $False$ — если на отрезке от x_{left} до x_{right} отсутствуют пересечения активных многоугольников. $True$ — в противном случае. В случае возврата $True$ параметр x_{int} содержит значение координаты x для точки пересечения.

начало алгоритма

- **цикл для каждого** *многоугольника P_j такого, что $Active(P_j)$* **выполнить**
  - Вычислить
    $$z_{j\,left} = -\frac{a_j x_{left} + b_j y_t + d_j}{c_j}; \quad z_{j\,right} = -\frac{a_j x_{right} + b_j y_t + d_j}{c_j};$$
  - **цикл для каждого** *многоугольника P_k такого, что $Active(P_k)$* **выполнить**
    - Вычислить
      $$z_{k\,left} = -\frac{a_k x_{left} + b_k y_t + d_k}{c_k}; \quad z_{k\,right} = -\frac{a_k x_{right} + b_k y_t + d_k}{c_k};$$
    - Вычислить $\Delta z_{left} = z_{j\,left} - z_{k\,left}$; $\Delta z_{right} = z_{j\,right} - z_{k\,right}$;
    - **если** $sign(\Delta z_{left}) \neq sign(\Delta z_{right})$ **то**
      - Вычислить
        $$x_{int} = \frac{x_{right}\Delta z_{left} - x_{left}\Delta z_{right}}{\Delta z_{left} - \Delta z_{right}};$$
      - Выдать $True$ и **закончить алгоритм**;
- Выдать $False$;

конец алгоритма