Предварительные замечания:

1. Описание алгоритма состоит из описания семи нижеприведенных процедур (седьмая процедура — дополнительная, на которую нет прямой ссылки в алгоритме).

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. Стек AW — список окон, заданных пятерками $(U_{cx}, U_{cy}, U_x, U_y, APL)$, где первые четыре элемента определяют прямоугольник внутри окна наблюдения (аналог W_{cx}, W_{cy}, W_x, W_y), APL — список активных многоугольников для заданного окна. В APL для каждого многоугольника сохраняется пара (P, Sur), где P — идентификатор многоугольника, а Sur равно $True$ если многоугольник охватывающий для данного окна и $False$ — в противном случае.

5. Предполагается, что для каждого многоугольника P_i в списке $\mathscr{P}$ вычислены коэффициенты уравнения плоскости a_i, b_i, c_i, d_i (для этого можно использовать приведенный ниже седьмой алгоритм), а так же величины $x_{i\min}$, $x_{i\max}$, $y_{i\min}$, $y_{i\max}$ — минимальные и максимальные значения координат x и y вершин многоугольника.

6. В 6-м алгоритме используются операции над векторами (сложение, вычитание, умножение на скаляр) и псевдоскалярное произведение. Предполагается, что эти операции реализованы.

7. Для изображения многоугольника в программе на Visual C++ необходимо использовать процедуру `Graphics::FillPolygon(SolidBrush^, array<PointF>^)`.

Алгоритм 1: Алгоритм Варнока

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

начало алгоритма

· Присвоить $APL_0 = \varnothing$, $AW = \varnothing$;

· Для каждого многоугольника P_i в $\mathscr{P}$, у которого $c_i \neq 0$ поместить APL_0 пару $(P_i, False)$;

· Положить в AW пятерку $(W_{cx}, W_{cy}, W_x, W_y, APL_0)$;

цикл пока $AW \neq \varnothing$ **выполнять**

· Взять из AW пятерку $(U_{cx}, U_{cy}, U_x, U_y, APL)$;

· Выполнить процедуру ProcessWindow$(U_{cx}, U_{cy}, U_x, U_y, APL)$, получить в результате значения $polygonCount$ (количество многоугольников после обработки), $hasSurround$ ($True$, если в списке APL нашелся охватывающий многоугольник, закрывающий собой все остальные многоугольники), C (цвет охватывающего многоугольника, в случае истинности $hasSurround$), $insider$ ($True$, если в APL внутренний многоугольник в случае $polygonCount = 1$);

если $polygonCount = 0$ **то** закрасить окно $(U_{cx}, U_{cy}, U_x, U_y)$ цветом фона;

иначе если $hasSurround$ **то** закрасить окно $(U_{cx}, U_{cy}, U_x, U_y)$ цветом C;

иначе если $U_x \leqslant 1$ *и* $U_y \leqslant 1$ **то**

· Выполнить процедуру PointColor$(U_{cx} + U_x/2, U_{cy} + U_y/2, APL)$ для получения цвета C;

· Закрасить окно $(U_{cx}, U_{cy}, U_x, U_y)$ цветом C;

иначе если $polygonCount = 1$ **то**

· Взять пару (P, S) из APL;

· Закрасить окно $(U_{cx}, U_{cy}, U_x, U_y)$ цветом фона;

если $insider$ **то** Изобразить многоугольник P;

иначе

· Выполнить процедуру PolygonClip$(P, U_{cx}, U_{cy}, U_{cx} + U_x, U_{cy} + U_y)$ для получения P' — многоугольника — результата отсечения;

· Изобразить многоугольник P' цветом многоугольника P;

иначе

если $U_x > 1$ **то** присвоить $i_x = 2$ **иначе** $i_x = 1$;

если $U_y > 1$ **то** присвоить $i_y = 2$ **иначе** $i_y = 1$;

· Присвоить $U'_x = U_x/i_x$, $U'_y = U_y/i_y$;

цикл для каждого $0 \leqslant j_x < i_x$ *и* $0 \leqslant j_y < i_y$ **выполнить**

Положить в стек AW пятерку $(U_{cx} + j_x U'_x, U_{cy} + j_y U'_y, U'_x, U'_y, APL)$

конец алгоритма

Алгоритм 2: ProcessWindow

Вход: U_{cx}, U_{cy}, U_x, U_y — параметры окна, APL — список активных многоугольников

Выход: $polygonCount$ — количество многоугольников после обработки; $hasSurround$ — $True$, если в списке APL нашелся охватывающий многоугольник, закрывающий собой все остальные многоугольники; C — цвет охватывающего многоугольника, в случае истинности $hasSurround$; $insider$ — $True$, если в APL находится внутренний многоугольник в случае $polygonCount = 1$; измененный список APL

начало алгоритма

- Присвоить $z_{lb} = 0$; $z_{lt} = 0$, $z_{rb} = 0$, $z_{rt} = 0$, $hasSurround = False$, $insider = False$, $polygonCount = 0$, $i = 1$;

цикл пока *не достигнут конец списка* APL **выполнять**

- Пусть (P_i, S_i, I_i) — очередной элемент APL;

если $!S_i$ *и* ($x_{i\max} < U_{cx}$ *или* $y_{i\max} < U_{cy}$ *или* $x_{i\min} > U_{cx} + U_x$ *или* $y_{i\min} > U_{cy} + U_y$) **то** удалить пару (P_i, S_i) из APL;

иначе

- $polygonCount = polygonCount + 1$;
- Вычислить
  $z_{i\,lb} = \text{Depth}(U_{cx}, U_{cy}, P_i)$;
  $z_{i\,lt} = \text{Depth}(U_{cx}, U_{cy} + U_y, P_i)$;
  $z_{i\,rb} = \text{Depth}(U_{cx} + U_x, U_{cy}, P_i)$;
  $z_{i\,rt} = \text{Depth}(U_{cx} + U_x, U_{cy} + U_y, P_i)$;
- $k = 0$;

если $z_{i\,lb} > z_{lb}$ **то** присвоить $z_{lb} = z_{i\,lb}$, $k = k + 1$;
если $z_{i\,lt} > z_{lt}$ **то** присвоить $z_{lt} = z_{i\,lt}$, $k = k + 1$;
если $z_{i\,rb} > z_{rb}$ **то** присвоить $z_{rb} = z_{i\,rb}$, $k = k + 1$;
если $z_{i\,rt} > z_{rt}$ **то** присвоить $z_{rt} = z_{i\,rt}$, $k = k + 1$;
если $0 < k < 4$ **то** присвоить $hasSurround = False$;
если $!S_i$ **то**

если SurroundTest(U_{cx}, U_{cy}, P_i) **то**

если SurroundTest$(U_{cx}, U_{cy} + U_y, P_i)$ **то**

если SurroundTest$(U_{cx} + U_x, U_{cy}, P_i)$ **то**

если SurroundTest$(U_{cx} + U_x, U_{cy} + U_y, P_i)$ **то**
присвоить $S_i = True$

иначе если $polygonCount = 1$ **то**
$insider = x_{i\min} > U_{cx}$ и $y_{i\min} > U_{cy}$ и $x_{i\max} < U_{cx} + U_x$ и $y_{i\max} < U_{cy} + U_y$

если S_i *и* $k = 4$ **то**
Присвоить цвет многоугольника P_i переменной C, $hasSurround = True$

- Присвоить $i = i + 1$;

конец алгоритма

Алгоритм 3: PointColor

Вход: x, y — координаты точки, APL — список многоугольников.

Выход: C — цвет ближайшего к наблюдателю многоугольника в точке, имеющей координаты x и y.

начало алгоритма

Присвоить переменной C цвет фона, $z = -\infty$, $i = 1$;

цикл пока *не достигнут конец списка* APL **выполнять**

Пусть (P_i, S_i, I_i) — очередной элемент APL;

если SurroundTest(x, y, P_i) **то**

Вычислить $z_i =$ Depth(x, y, P_i);

если $z_i > z$ **то** Присвоить $z = z_i$, $C = C_i$;

Присвоить $i = i + 1$;

конец алгоритма

Алгоритм 4: Depth Вычисление глубины точки на несущей плоскости

Вход: x, y — координаты точки, P — идентификатор многоугольника.

Выход: z — координата z точки на несущей плоскости многоугольника P, имеющая координаты x и y.

начало алгоритма

$$z = -\frac{ax + by + d}{c}$$

конец алгоритма

Алгоритм 5: SurroundTest Проверка на принадлежность точки многоугольнику

Вход: x, y — координаты точки, P — список вершин выпуклого многоугольника.

Выход: $True$ — Если точка с координатами x и y лежит внутри многоугольника.

начало алгоритма

Присвоить переменной n количество элементов в списке P, $i = 1$;

$(x_1, y_1, z_1) = P[n]$, $Sgn_1 = 0$;

цикл пока $Sgn_1 = 0$ **выполнять**

$(x_2, y_2, z_2) = P[i]$;

Вычислить $Sgn_1 = signum((x_2 - x_1)(y - y_1) - (y_2 - y_1)(x - x_1))$;

Присвоить $x_1 = x_2$, $y_1 = y_2$, $i = i + 1$;

цикл пока $i \leqslant n$ **выполнять**

$(x_2, y_2, z_2) = P[i]$;

Вычислить $Sgn_2 = signum((x_2 - x_1)(y - y_1) - (y_2 - y_1)(x - x_1))$;

если $Sgn_2 \neq 0$ *и* $Sgn_2 \neq Sgn_1$ **то** вернуть $False$; **закончить алгоритм**;

Присвоить $x_1 = x_2$, $y_1 = y_2$, $i = i + 1$;

Вернуть $True$;

конец алгоритма

Алгоритм 6: PolygonClip Алгоритм Сазерленда—Ходгмана отсечения многоугольника

Вход: P — список вершин выпуклого многоугольника, $x_{\min}$, $y_{\min}$, $x_{\max}$, y_{max} — параметры окна

Выход: P' — многоугольник после отсечения.

начало алгоритма

Присвоить $F = \{(x_{\min}, y_{\min}), (x_{\min}, y_{\max}), (x_{\max}, y_{\max}), (x_{\max}, y_{\min})\}$;

Присвоить переменной n_1 количество элементов в списке P, $P' = P$, $\bar{f}_0 = F_4$;

цикл для i *от* 1 *до* 4 **выполнять**

$Q(0) = (P'[n_1] - F_i) \times (\bar{f}_0 - F_i)$;

$\bar{p}_0 = P'[n_1]$;

$n_2 = 0$; $P'' = \{\}$;

цикл для k *от* 1 *до* n_1 **выполнять**

$Q(1) = (P'[k] - F_i) \times (\bar{f}_0 - F_i)$;

если $Q(0) \cdot Q(1) < 0$ **то**

$t = Q(0)/(Q(0) - Q(1))$;

$n_2 = n_2 + 1$; $P''[n_2] = \bar{p}_0 - (\bar{p}_0 - P'[k])t$

если $Q(1) \leqslant 0$ **то** $n_2 = n_2 + 1$; $P''[n_2] = P'[k]$;

$Q(0) = Q(1)$; $\bar{p}_0 = P'[k]$

$P' = P''$; $n_1 = n_2$; $\bar{f}_0 = F_i$;

если $n_1 = 0$ **то** Выдать P' (многоугольник с 0 вершинами); **закончить алгоритм**;

Выдать P' (многоугольник с n_1 вершинами);

конец алгоритма

Алгоритм 7: 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