Машинная графика
Computer Graphics

Лекция 8

«Отсечение многоугольников»

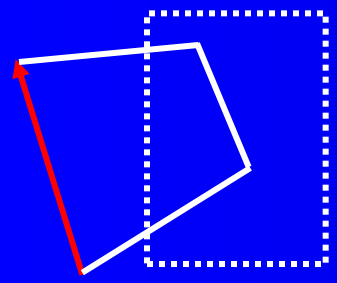
План лекции

- Упрощенный алгоритм отсечения многоугольников
- Алгоритм отсечения произвольных многоугольников произвольными окнами (взаимной заменой)
- Алгоритм Вейлера-Айзертона
- Отсечение литер (символов)
- Однородные координаты

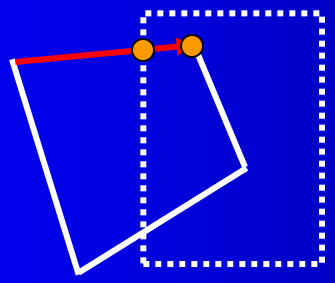
Упрощенный алгоритм отсечения многоугольников

Алгоритм построен по аналогии с алгоритмом Сазерленда-Хогмана. Порядок обхода многоугольника по часовой стрелке (но не принципиален).

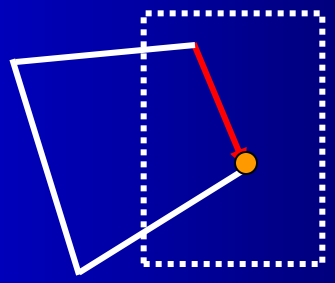
В отличие от алгоритма Сазерленда-Хогмана рассматриваются 9 частных случаев взаимного расположения ребра многоугольника и стороны окна. В алгоритме Сазерленда-Хогмана:



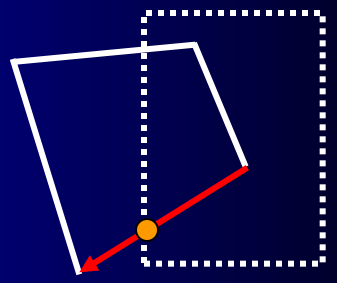
В список новых вершин не заносится



В список заносится 2 новые вершины



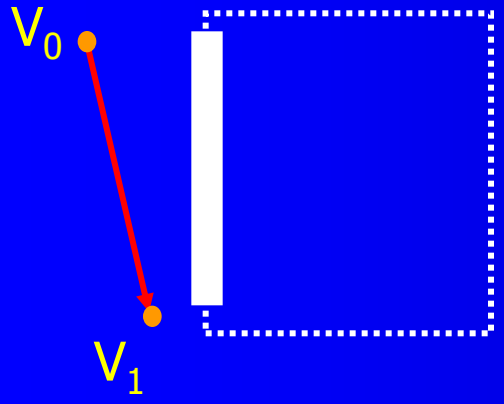
В список заносится только одна вершина



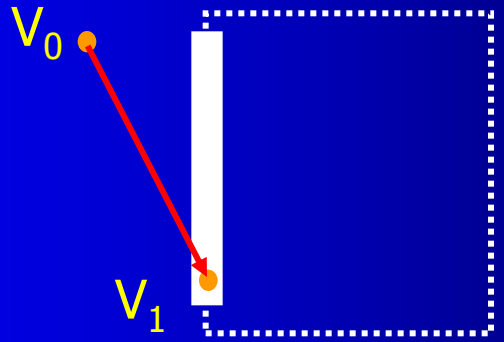
В список заносится одна граничная точка

Упрощенный алгоритм отсечения многоугольников

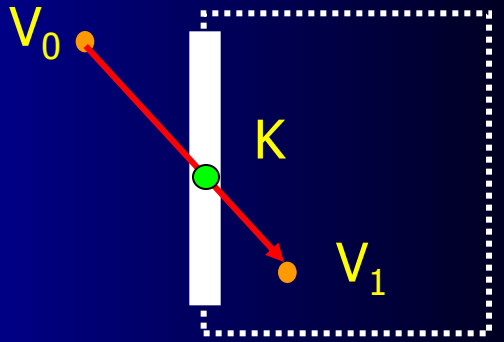
В упрощённом алгоритме:



1. V_0 лежит вне окна, V_1 лежит вне окна.



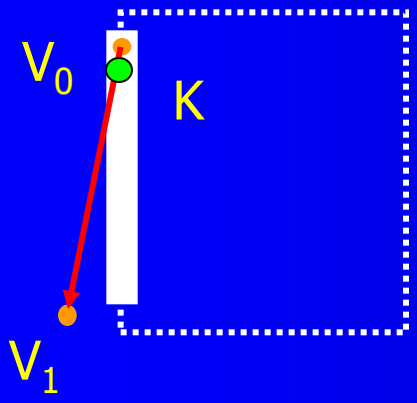
2. V_0 лежит вне окна, V_1 лежит на ребре окна.



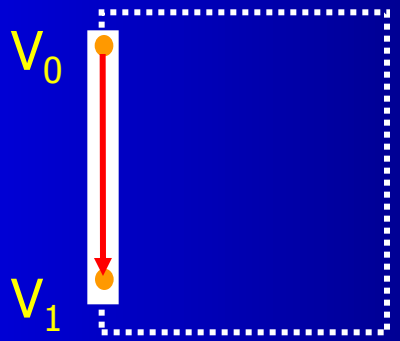
3. V_0 лежит вне окна, V_1 лежит внутри окна.

Упрощенный алгоритм отсечения многоугольников

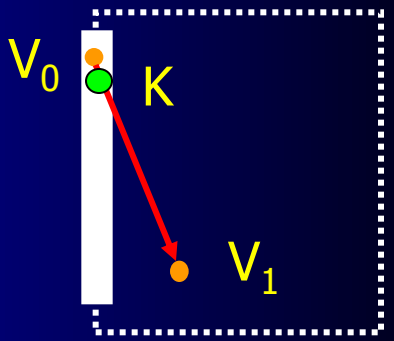
В упрощённом алгоритме:



4. V_0 лежит на ребре окна, V_1 лежит вне окна.



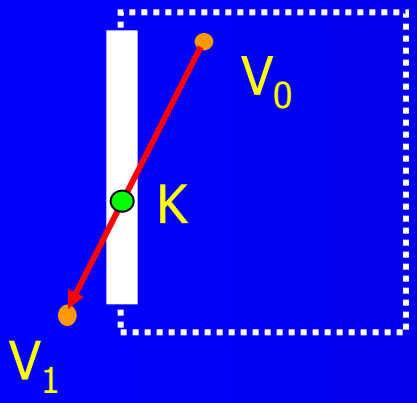
5. V_0 лежит на ребре окна, V_1 лежит на ребре окна.



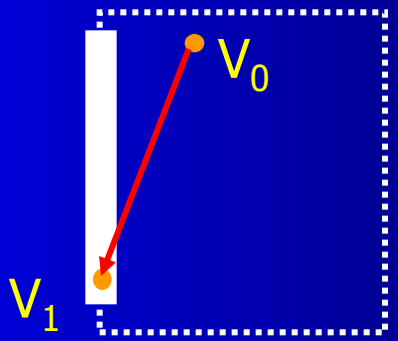
6. V_0 лежит на ребре окна, V_1 лежит внутри окна.

Упрощенный алгоритм отсечения многоугольников

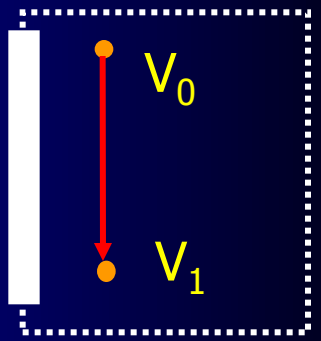
В упрощённом алгоритме:



7. V_0 лежит внутри окна, V_1 лежит вне окна.



8. V_0 лежит внутри окна, V_1 лежит на ребре окна.



9. V_0 лежит внутри окна, V_1 лежит внутри окна.

Упрощенный алгоритм отсечения многоугольников

При обходе многоугольника в список вершин заносится в:

1 случае – ничего не заносится,

2 случае – конечная точка ребра многоугольника V_1

3 случае – точка пересечения K и конечная точка ребра V_1

4 случае – ничего не заносится,

5 случае – заносится конечная точка ребра V_1

6 случае – конечная точка ребра многоугольника V_1

7 случае – точка пересечения K,

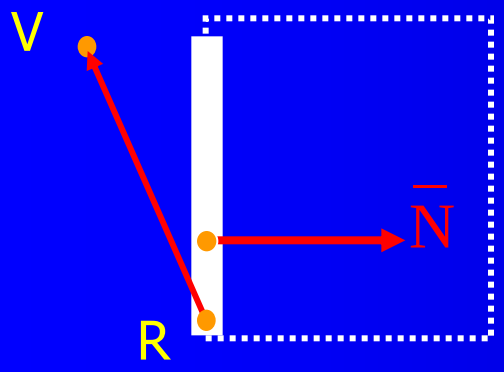
8 случае – заносится конечная точка ребра V_1

9 случае – конечная точка ребра многоугольника V_1

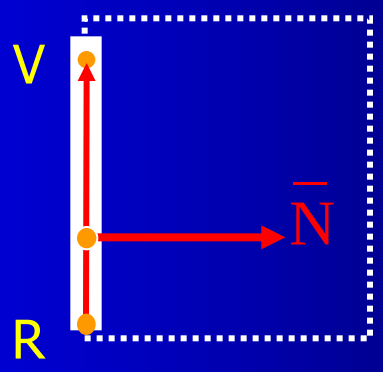
Упрощенный алгоритм отсечения многоугольников

Для определения факта видимости точки используются скалярные произведения векторов **RV** и **N**:

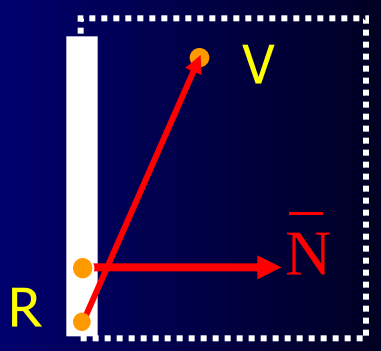
R- точка начала ребра многоугольника



1. V лежит вне окна



2. V лежит на ребре окна.



3. V лежит внутри окна.

Скалярное произведение $\mathbf{RV_0}$ и **N** - $\mathbf{Q_n = (V_0 - R) \cdot N}$

Скалярное произведение $\mathbf{RV_1}$ и **N** - $\mathbf{Q_k = (V_1 - R) \cdot N}$

Упрощенный алгоритм отсечения многоугольников

Для нахождения точки пересечения:

Скалярное произведение $\mathbf{RV_0}$ и $\mathbf{N}$ - $Q_n = (\mathbf{V_0}-\mathbf{R})\cdot\mathbf{N}$

Скалярное произведение $\mathbf{RV_1}$ и $\mathbf{N}$ - $Q_k = (\mathbf{V_1}-\mathbf{R})\cdot\mathbf{N}$

$\mathbf{V}(t) = \mathbf{V_0} + (\mathbf{V_1}-\mathbf{V_0})*t$ $\quad t = -Q_n/P_n$

$P_n = (\mathbf{V_1}-\mathbf{V_0})\cdot\mathbf{N}$

$P_n = (\mathbf{V_1}-\mathbf{V_0})\cdot\mathbf{N} = (\mathbf{V_1}-\mathbf{V_0}+\mathbf{R}-\mathbf{R})\cdot\mathbf{N} =$
$= (\mathbf{V_1}-\mathbf{R})\cdot\mathbf{N} - (\mathbf{V_0}-\mathbf{R})\cdot\mathbf{N} = Q_k - Q_n$

$t = -Q_n/(Q_k - Q_n) = Q_n/(Q_n - Q_k)$

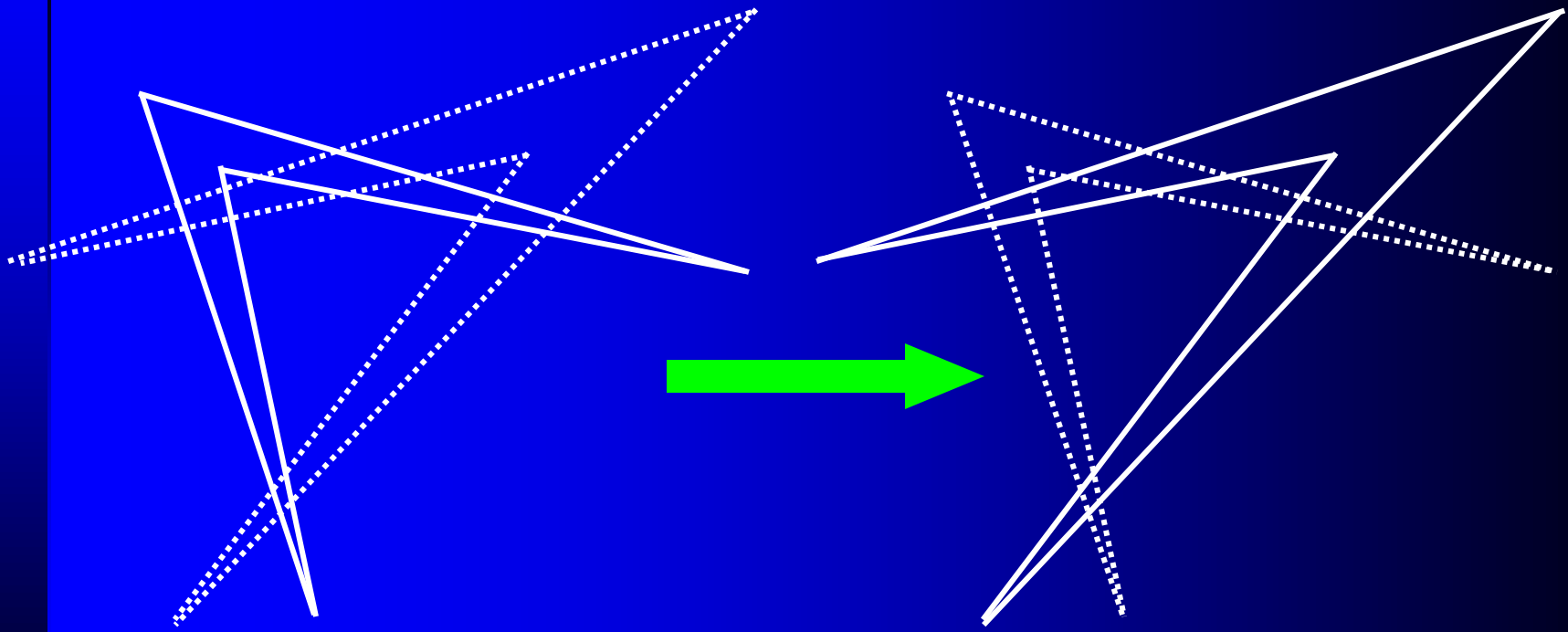
$x=x_0 +dx*t \quad y=y_0 +dy*t$

где $dx = x_1-x_0$; $dy = y_1-y_0$;

(x_0,y_0) и (x_1, y_1) – координаты точек V_0 и V_1, соответственно

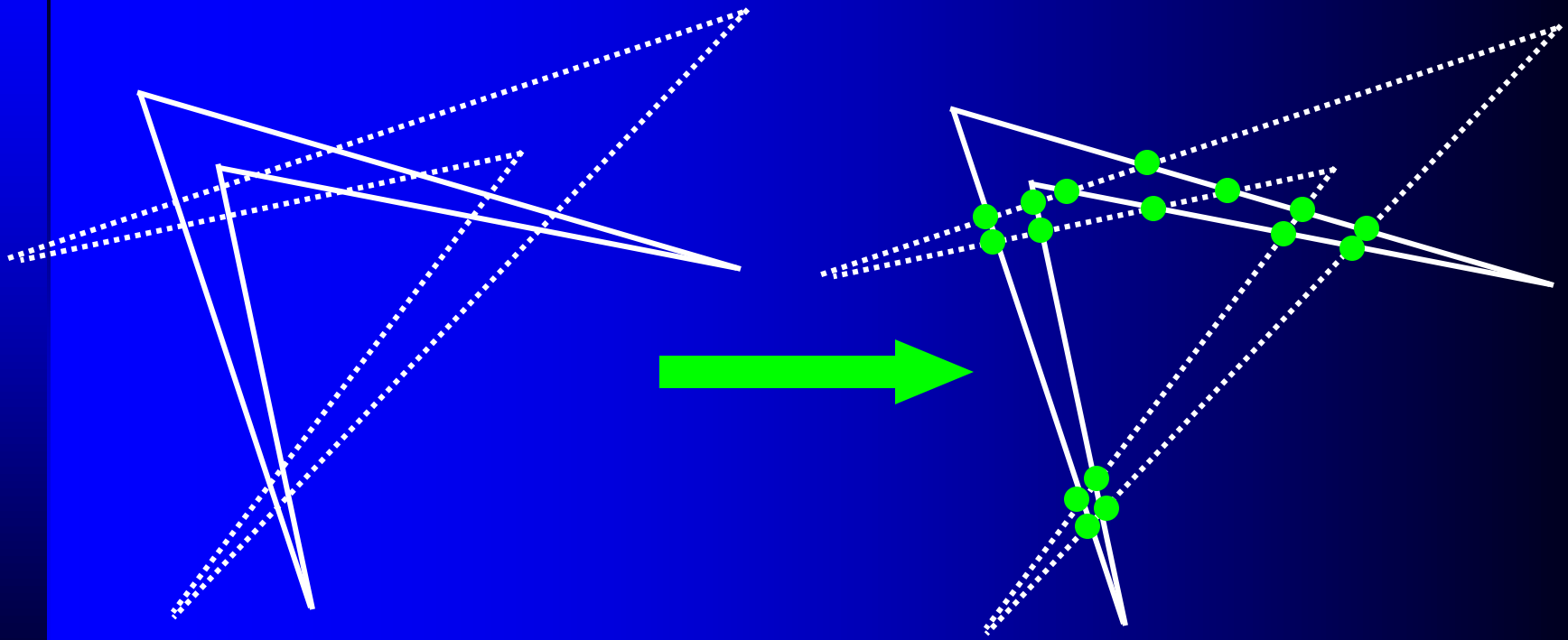
Алгоритм отсечения произвольных многоугольников произвольными окнами (взаимной заменой)

Алгоритм демонстрирует красивую идею взаимной замены (по функции) отсекаемого многоугольника и окна отсечения.



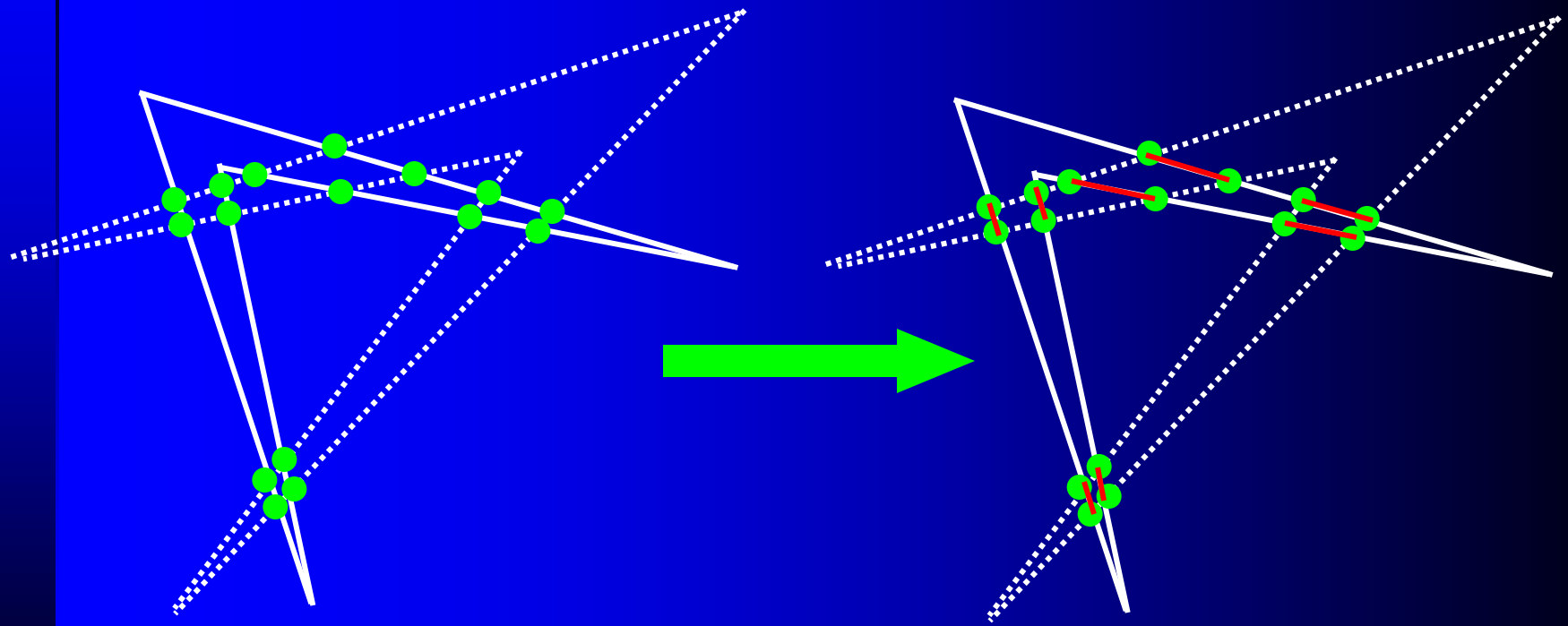
Алгоритм отсечения произвольных многоугольников произвольными окнами (взаимной заменой)

На первом этапе отсечения рассчитываются все точки пересечения рёбер многоугольника со сторонами окна.



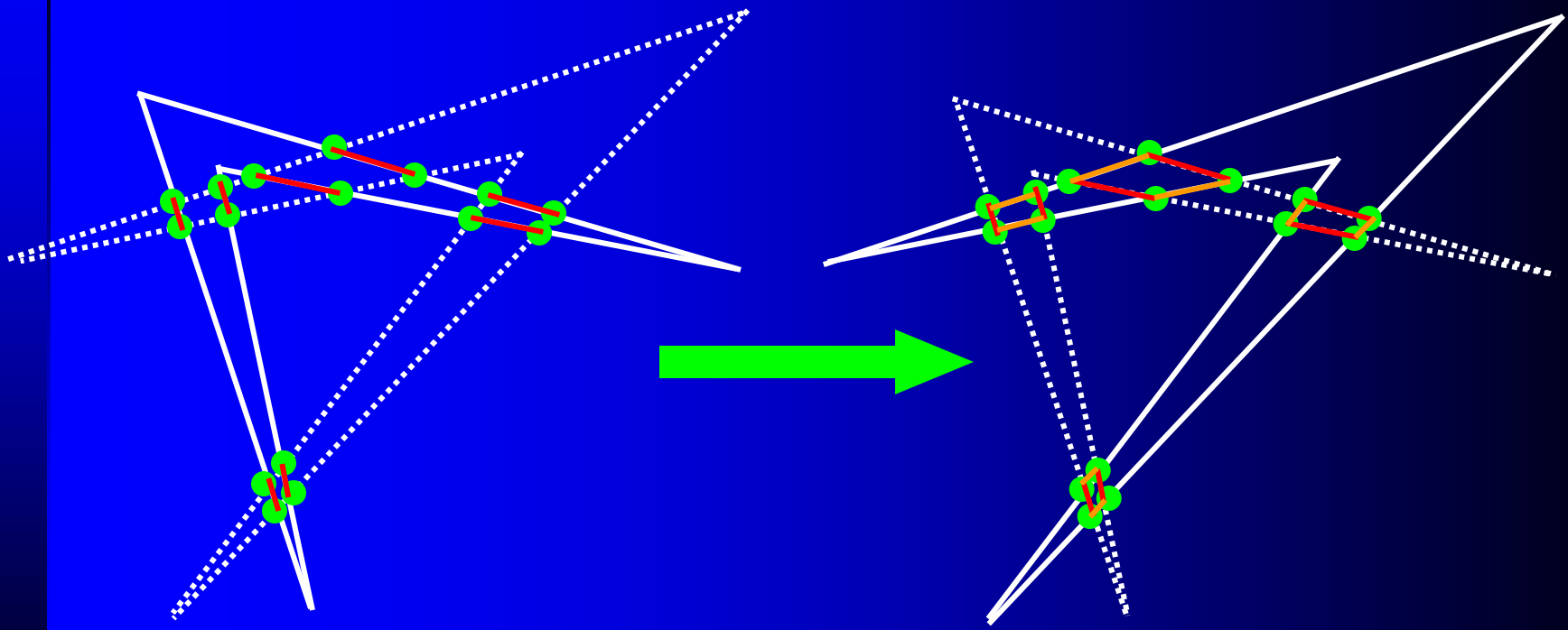
Алгоритм отсечения произвольных многоугольников произвольными окнами (взаимной заменой)

На втором этапе находятся отрезки – фрагменты рёбер отсекаемого многоугольника, попадающие в окно отсечения.



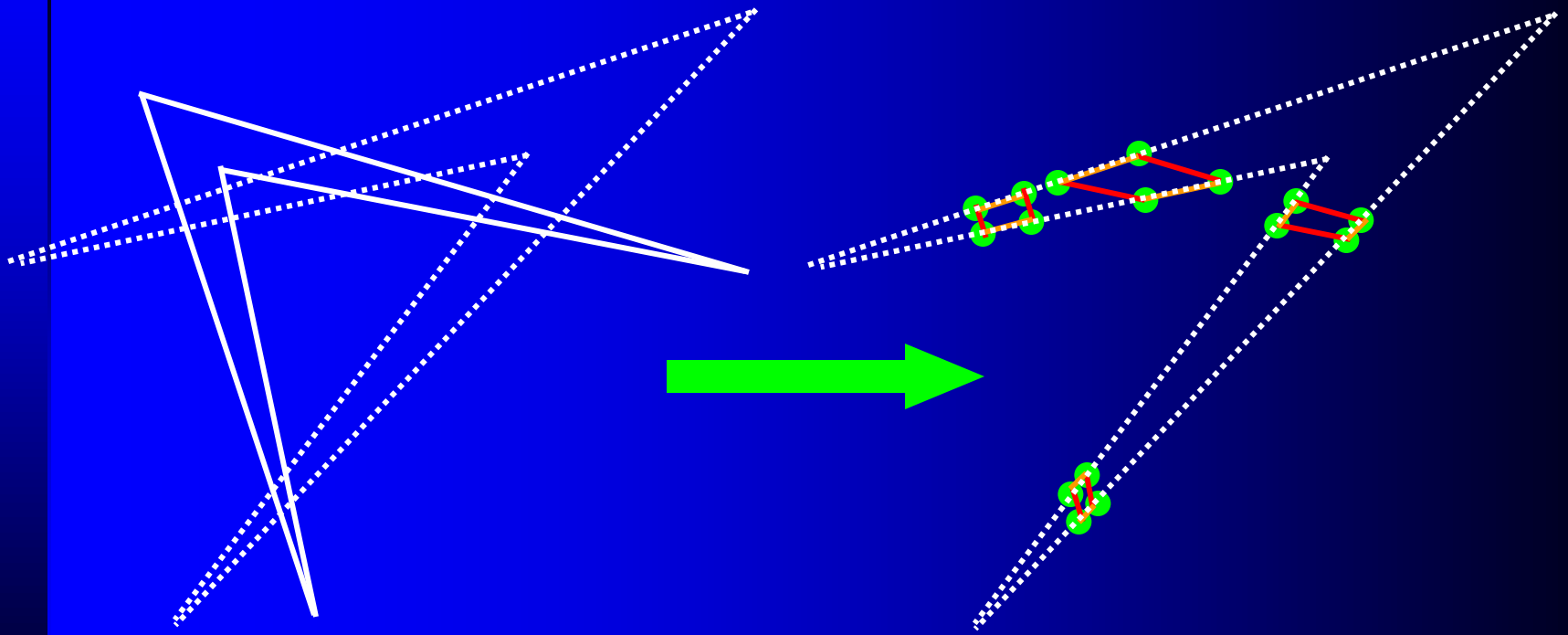
Алгоритм отсечения произвольных многоугольников произвольными окнами (взаимной заменой)

На третьем этапе производится «виртуальная» замена – окна отсечения и отсекаемого многоугольника и опять находятся отрезки – фрагменты рёбер «отсекаемого многоугольника», попадающие в «окно отсечения».



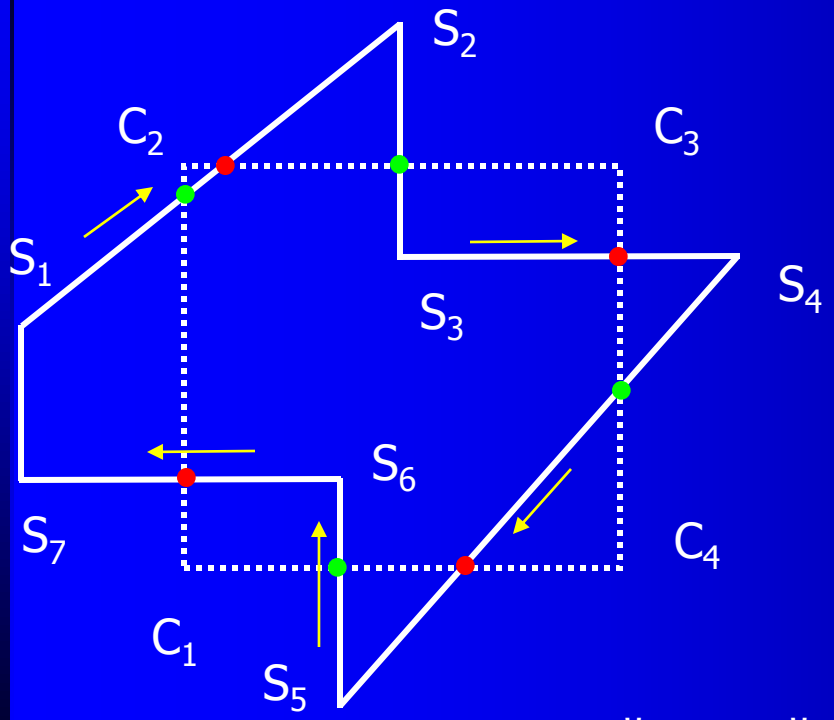
Алгоритм отсечения произвольных многоугольников произвольными окнами (взаимной заменой)

На заключительном этапе производится анализ получившихся точек пересечения и рёбер, а так же формирование результата в виде многоугольника (-ов).



Алгоритм отсечения многоугольников Вейлера-Азертона (Weiler-Atherton)

Алгоритм позволяет проводить отсечение невыпуклого многоугольника с внутренними отверстиями по другому невыпуклому многоугольнику с внутренними отверстиями.



Простой случай

Суть алгоритма:

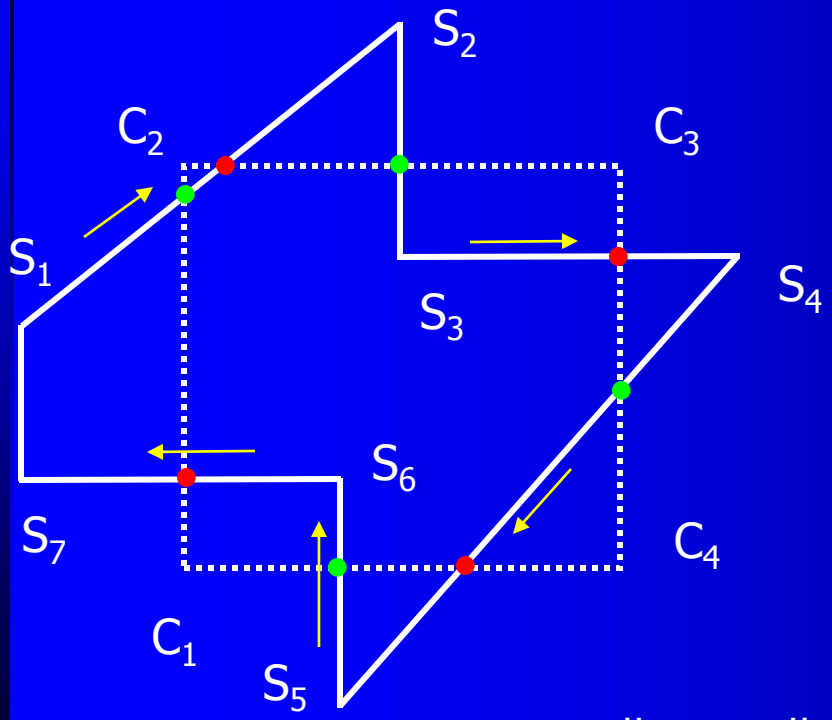
Многоугольники – как отсекаемый, так и окно отсечения описываются циклическими списками вершин.

Внешняя сторона каждого многоугольника обходится по часовой стрелке, внутренние границы (отверстия) – против.

Точки пересечения рёбер многоугольника и сторон окна могут быть сформированы по парам - точка входа • и точка выхода • .

Алгоритм отсечения многоугольников Вейлера-Азертона (Weiler-Atherton)

Суть алгоритма:



Простой случай

Основная идея алгоритма заключается в том, что начиная с некоторой точки пересечения входного типа, прослеживают внешнюю границу отсекаемого многоугольника по часовой стрелке до тех пор, пока не обнаружат очередное пересечение границы со стороной окна.

В точке обнаруженного пересечения производится «поворот направо» и следование границы многоугольника до очередной точки.

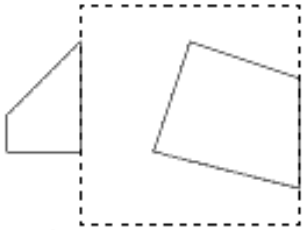
Цикл выполняется до тех пор пока не будет обнаружена точка начала алгоритма.

Внутренние границы обходят против часовой стрелки !!!!

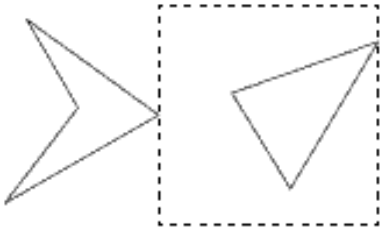
Алгоритм отсечения многоугольников Вейлера-Азертона (Weiler-Atherton)

Формальная запись алгоритма:

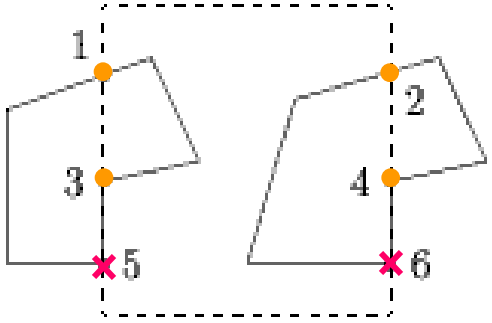
1. Строятся ЦИКЛИЧЕСКИЕ списки вершин для отсекаемого многоугольника и окна отсечения (S_1,S_2…S_7, S_1 / C_1,C_2,C_3,C_4,C_1).
2. Отыскиваются все точки пересечения. При этом не учитываются точки касания и случаи, при которых ребро инцидентно стороне окна либо совпадает со стороной окна.



Ребро отсекаемого многоугольника совпадает с окном.



Вершина отсекаемого многоугольника касается окна.



Точки 1–4 считаются пересечением.

Точки 5,6 не считаются пересечением.

Алгоритм отсечения многоугольников Вейлера-Азертона (Weiler-Atherton)

Формальная запись алгоритма:

3. Списки вершин отсекаемого многоугольника и окна отсечения дополняются новыми вершинами – точками пересечения. При этом, если точка пересечения P встречается на ребре между вершинами V_i и V_{i+1}, то она и вставляется в список между данными вершинами: …V_iPV_{i+1}…
4. Устанавливаются двухсторонние связи между одноимёнными точками списков.
5. Входные и выходные точки пересечения формируют собственные списки вершин.
6. Производится цикл отсечения.

Алгоритм отсечения многоугольников Вейлера-Азертона (Weiler-Atherton)

Цикл отсечения:

Если не исчерпан список входных вершин, берём очередную входную точку. Двигаемся по вершинам отсекаемого многоугольника, до тех пор пока не обнаружим следующую точку пересечения.

Все пройденные точки, за исключением прервавшей просмотр, заносим в результирующий список вершин.

Используя двухстороннюю связь между списками, переключаемся на просмотр списка вершин окна отсечения.

Двигаемся по вершинам окна отсечения до обнаружения след. точки пересечения. Все пройденные вершины (за исключением последней) заносим в список результата.

Используя двухстороннюю связь между списками, переключаемся на просмотр списка вершин многоугольника.

Цикл повторяем до тех пор, пока не достигнем входной вершины - очередная часть отсекаемого многоугольника, попавшая в окно, замкнулась.

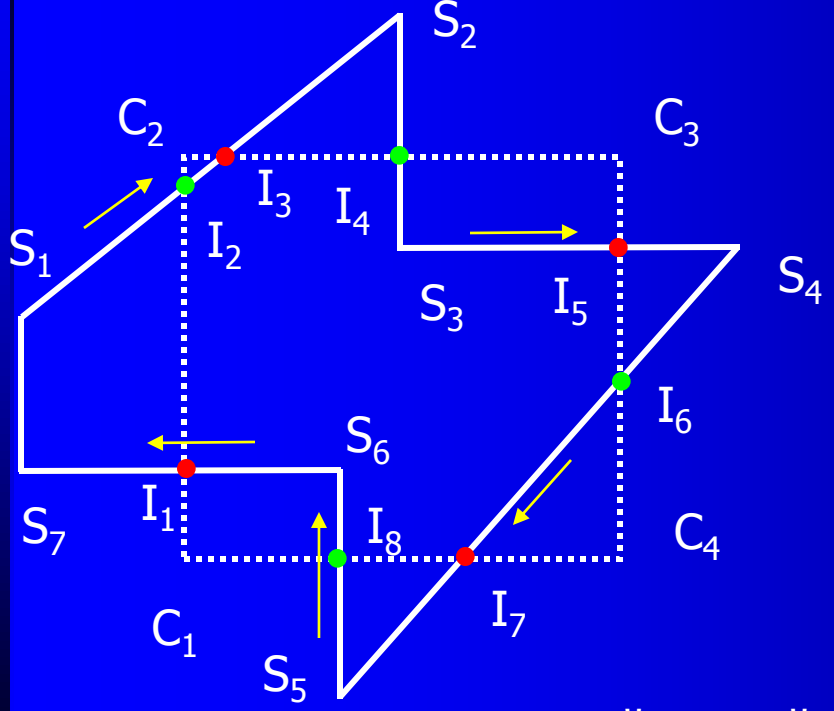
В случае наличия не пройденных вершин в списке входных точек, выбираем очередную входную точку и повторяем цикл, начав с неё.

Алгоритм отсечения многоугольников Вейлера-Азертона (Weiler-Atherton)

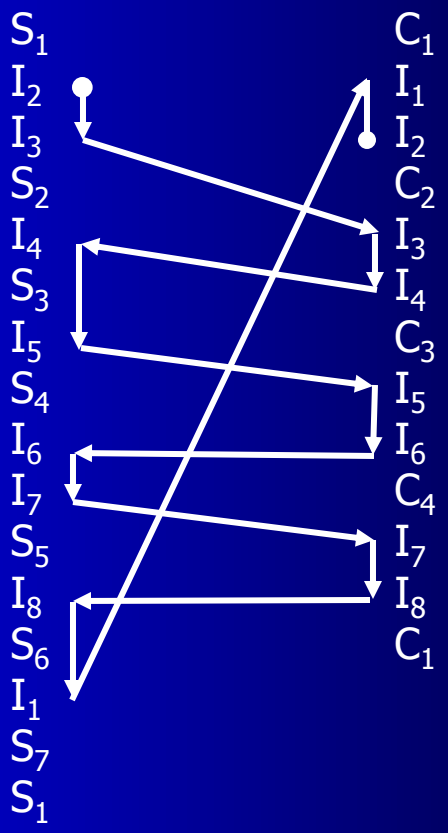
Пример:

точки входа: $I_2 I_4 I_6 I_8$ ●

точки выхода: $I_1 I_3 I_5 I_7$ ●



Простой случай



Вершины результата:

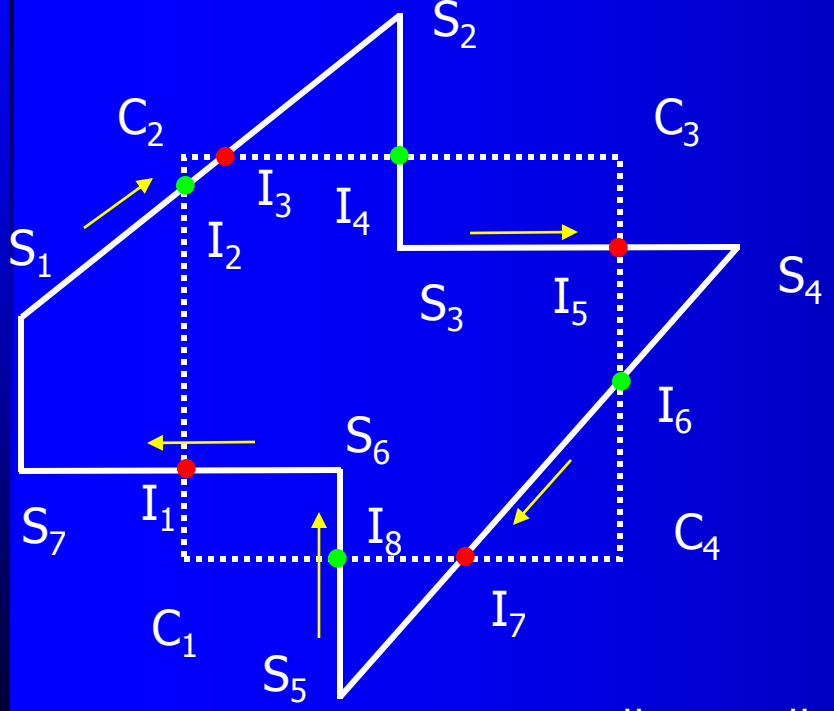
I_2
I_3
I_4
S_3
I_5
I_6
I_7
I_8
S_6
I_1
I_2

Алгоритм отсечения многоугольников Вейлера-Азертона (Weiler-Atherton)

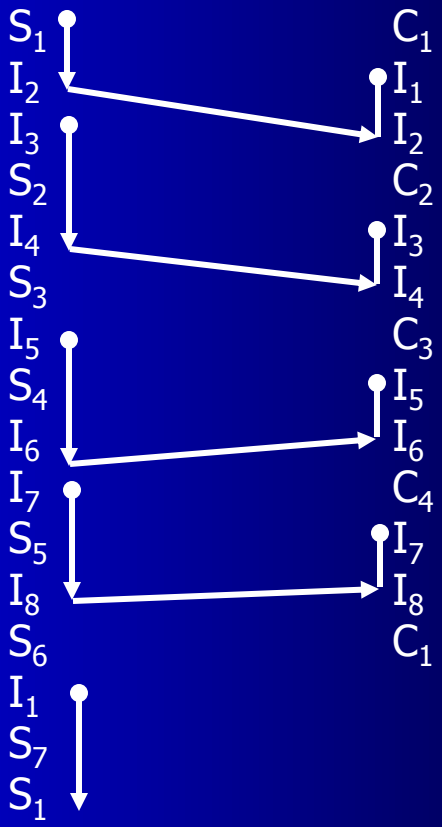
Пример: внешнее отсечение

точки входа: $I_2 I_4 I_6 I_8$

точки выхода: $I_1 I_3 I_5 I_7$



Простой случай



Вершины результата:

$I_1 S_7 S_1 I_2 I_1$

$I_3 S_2 I_4 I_3$

$I_5 S_4 I_6 I_5$

$I_7 S_5 I_8 I_7$

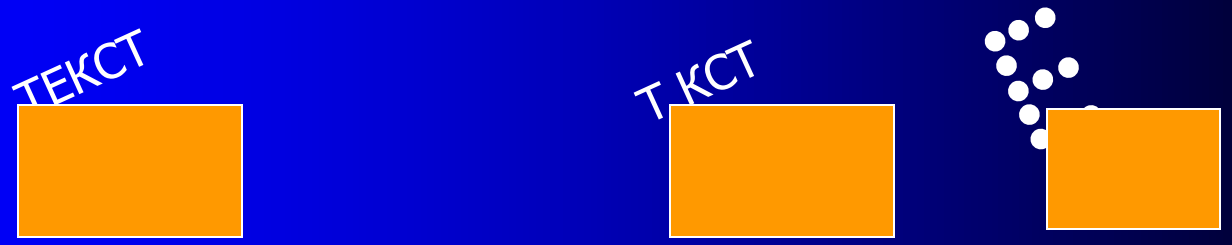
Отсечение литер

Символы (текст) могут быть сгенерированы программно, микропрограммно и аппаратно. Соответсвенно, литеры могут состоять из отдельных отрезков (штрихов, кривых - векторный вид) или быть заданными точечной матрицей (растровый вид).

Символы заданные в векторном виде можно обрабатывать как и любые другие отрезки: поворачивать, масштабировать, отсекать.

Программно сгенерированные символы в виде матрицы в принципе можно так же обрабатывать – но это более трудоёмкая задача – требуется проверить каждый пиксель.

Аппаратно сгенерированные символы в случае неполной видимости обычно удаляются полностью.



Однородные координаты

Преобразования 2D (Декартовы координаты)

Перенос:

$$\begin{cases} x' = x + dx \\ y' = y + dy \end{cases}$$

Поворот:

$$\begin{cases} x' = x \cos(\varphi) - y \sin(\varphi) \\ y' = x \sin(\varphi) + y \cos(\varphi) \end{cases}$$

Матрица поворота $R(\varphi) = \begin{bmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{bmatrix}$

и вектора $P = \begin{bmatrix} x \\ y \end{bmatrix}$, $P' = \begin{bmatrix} x' \\ y' \end{bmatrix}$

Запись преобразования в матричном виде будет следующей :

$$\mathbf{P' = R(\varphi)P}$$

Поворот относительно произвольной точки

Поворот относительно произвольной точки (x_0,y_0) можно рассматривать как последовательность нескольких действий:

- сдвиг системы координат,
- поворот относительно нового центра координат
- возврат к старой системе координат.

Соответственно можно обойтись следующими изменениями:

$x \to x-x_0$, $y \to y-y_0$, $x' \to x'-x_0$, $y' \to y'-y_0$

$$\begin{cases} x' - x_0 = (x-x_0)\cos(\varphi) - (y-y_0)\sin(\varphi) \\ y' - y_0 = (x-x_0)\sin(\varphi) + (y-y_0)\cos(\varphi) \end{cases} \qquad P_0 = \begin{bmatrix} x_0 \\ y_0 \end{bmatrix}$$

$$\mathbf{P' = P_0 + R(\varphi)(P - P_0)}$$

Масштабирование, отражение

2D Масштабирование: $x' = x \cdot s_x$, $y' = y \cdot s_y$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix}$$

$$\mathbf{P' = S \cdot P}$$

2D Отражение: $x' = (\ )1 \cdot x$, $y' = (\ )1 \cdot y$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} (\ )1 & 0 \\ 0 & (\ )1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix}$$

Сдвиг

2D Сдвиг в направлении оси OX: $x' = x,\ y' = sh_x \cdot y$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & sh_x \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix}$$

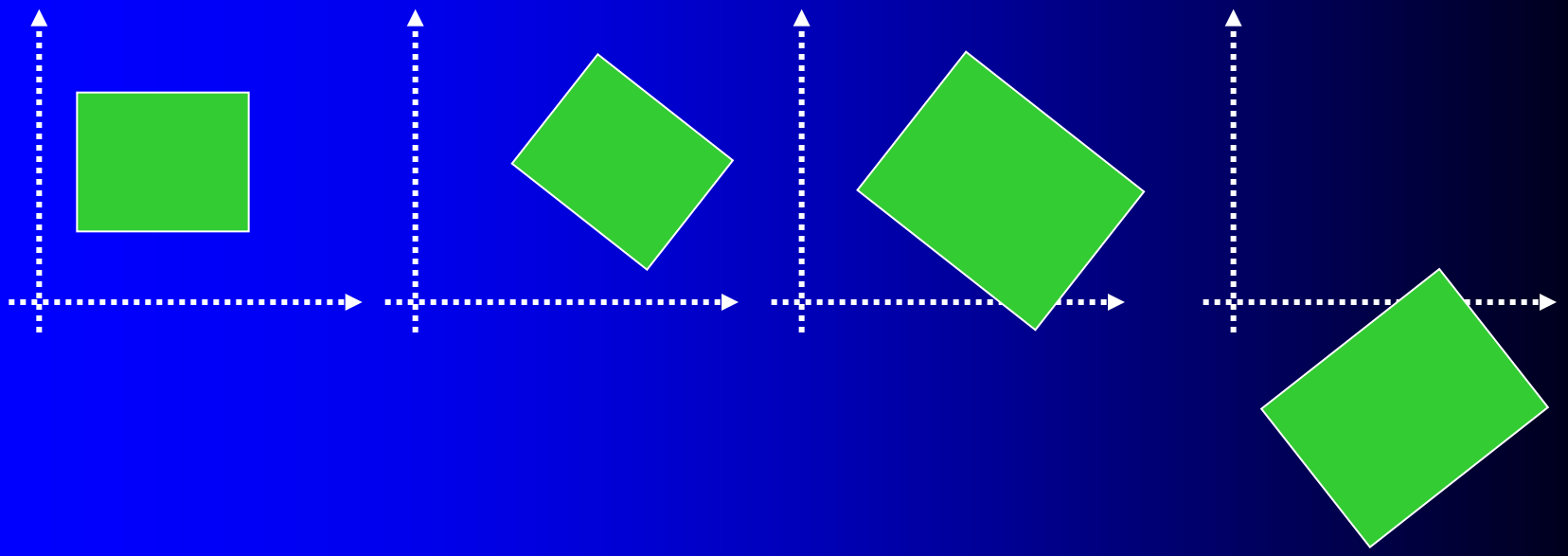
2D Сдвиг в направлении оси OY: $x' = sh_y \cdot x,\ y' = y$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ sh_y & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix}$$

Преобразования 2D (Декартовы координаты)

Что делать, если требуется произвести цепочку преобразований над некоторым объектом.

Например: сдвиг+поворот+масштабирование+отражение…



Однородные координаты

Однородными координатами служат тройки чисел (одновременно не равные нулю), связанные с обычными координатами точек плоскости соотношением:

$$\begin{pmatrix} \bar{x} \\ \bar{y} \\ w \end{pmatrix} = w \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$ так что $$x = \bar{x}/w,\ y = \bar{y}/w$$

Двумерный вектор (x,y) в однородных координатах записывается в виде (wx,wy,w) , где $w \neq 0$. Число w называется масштабным множителем. Для того, чтобы из вектора, записанного в однородных координатах получить вектор в обычных координатах необходимо разделить первые две координаты на третью: (wx/w,wy/w,w/w)->(x,y,1).

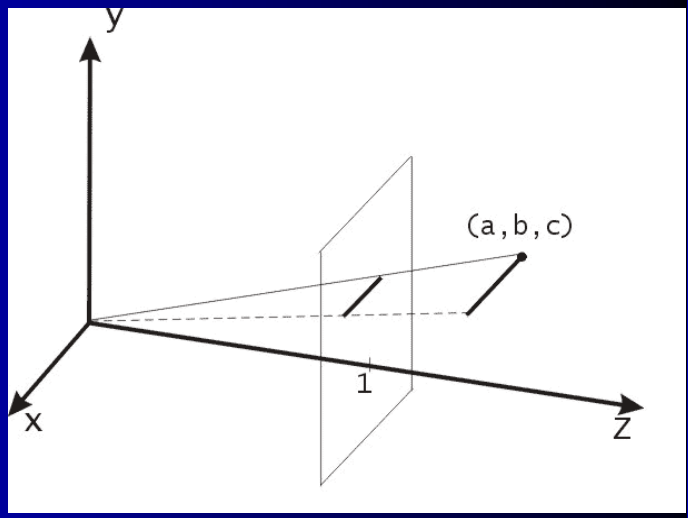
Однородные координаты для 2D случая можно представить как промасштабированные с коэффициентом w значения двумерных координат, расположенные в плоскости с $Z = w$.

Однородные координаты

В общем случае осуществляется переход от n-мерного пространства к (n+1)-мерному. Обратное преобразование называется проекцией однородных координат.

Некоторые свойства однородных координат.

Рассмотрим точку трехмерного пространства (a,b,c). Если представить эту точку как однородное представление точки двумерного пространства, то ее координаты будут (a/c,b/c). Сравнивая эти координаты со вторым видом формул, выведенных для центральной перспективной проекции, легко заметить, что двумерное представление точки с координатами (a,b,c) выглядит как ее проекция на плоскость z=1.



Проекция точки (a,b,c) на плоскость z=1

Однородные координаты

Таким образом в однородных координатах точки двумерного пространства описываются трехэлементными вектор-строками, что позволяет все матрицы преобразований, на которые будет умножаться вектор точки, привести к одному виду- размера 3×3. Матричное преобразование операции переноса для однородных координат:

$$[x', y', 1] = [x, y, 1] \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ D_x & D_y & 1 \end{bmatrix} \quad \text{или} \quad p' = p \cdot T(D_x, D_y)$$

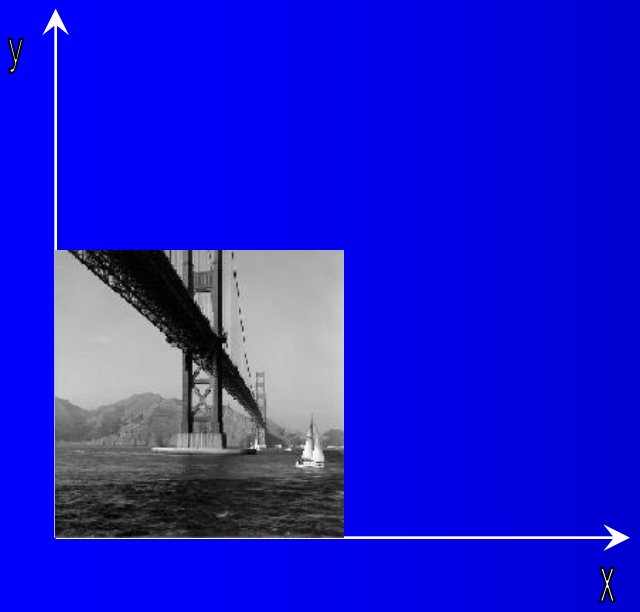
где $T(D_x, D_y) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ D_x & D_y & 1 \end{bmatrix}$ - матрица 2D переноса

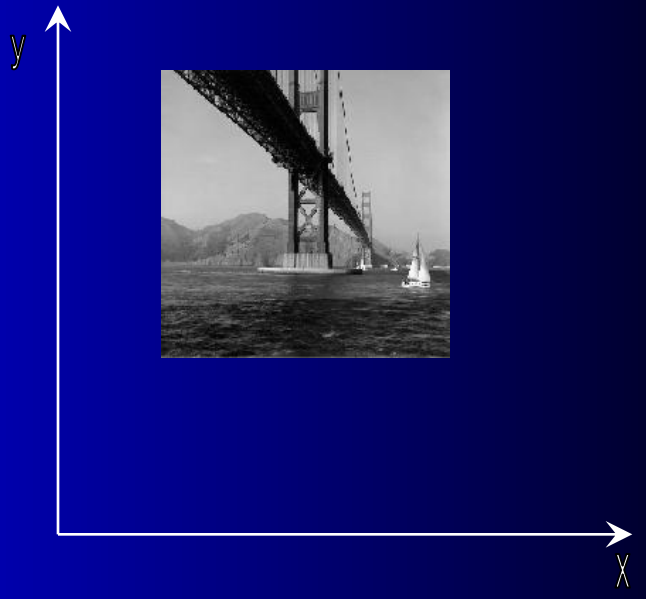
Однородные координаты

В некоторых источниках матрица переноса имеет вид:

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & D_x \\ 0 & 1 & D_y \\ 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & D_x \\ 0 & 1 & D_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$





Однородные координаты

Аналогично, рассматривая применение однородных координат для векторов трехмерного пространства, можно представить трехмерное пространство как проекцию четырехмерного пространства на гиперплоскость w=1 , если

$$(x, y, z) \rightarrow (wx, wy, wz, w) = (x, y, z, 1)$$

Двумерные преобразования в однородных координатах

В силу произвольности значения ***w*** в однородных координатах не существует единственного представления точки, заданной в декартовых координатах.

Преобразования сдвига, масштабирования и поворота в однородных координатах относительно центра координат все имеют одинаковую форму произведения вектора исходных координат на матрицу преобразования.

- Для сдвига

$$[xn \quad yn \quad wn] = [x \quad y \quad w] \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ Tx & Ty & 1 \end{bmatrix}$$

Растяжение *(сжатие)* вдоль координатных осей, задаваемое в виде:

$$x' = \alpha x, \alpha > 0,$$

$$y' = \beta y, \beta > 0.$$

Растяжению вдоль соответствующей оси соответствует значение масштабного множителя большего единицы. В однородных координатах матрица растяжения (сжатия) имеет вид

$$\mathbf{T}_{af} = \begin{bmatrix} \alpha & 0 & 0 \\ 0 & \beta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Поворот вокруг начальной точки на угол , описываемый формулой:

$$x' = x\cos\varphi - y\sin\varphi,$$

$$y' = x\sin\varphi + y\cos\varphi.$$

Матрица вращения (для однородных координат)

$$\mathbf{T}_{af} = \begin{bmatrix} \cos\varphi & -\sin\varphi & 0 \\ \sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Отражение (относительно какой либо из осей, например оси абсцисс) задается при помощи формулы:

$$x' = x,$$

$$y' = -y.$$

Матрица отражения, соответственно

$$\mathbf{T}_{af} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Для масштабирования $[xn \quad yn \quad wn] = [x \quad y \quad w] \cdot \begin{bmatrix} Sx & 0 & 0 \\ 0 & Sy & 0 \\ 0 & 0 & 1 \end{bmatrix}$

Для поворота $[xn \quad yn \quad wn] = [x \quad y \quad w] \cdot \begin{bmatrix} \cos\varphi & \sin\varphi & 0 \\ -\sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{bmatrix}$

матрица преобразования для двумерных однородных координат в **общем случае** имеет вид:

$$\begin{bmatrix} A & B & P \\ D & E & Q \\ L & M & S \end{bmatrix}$$

где элементы A, B, D и E определяют изменение масштаба, поворот и смещение, а L и M определяют сдвиг. Элемент S определяет общее изменение масштаба, а элементы P и Q определяют проецирование.

Композиция двумерных преобразований

Последовательное выполнение нескольких преобразований можно представить в виде единой матрицы суммарного преобразования. Умножение на единственную матрицу, естественно. выполняется быстрее, чем последовательное умножение на несколько матриц. Рассмотрим выполнение часто используемых преобразований:

- *Растяжение (сжатие)* вдоль координатных осей
- *Поворот* вокруг начальной точки на угол
- *Перенос*
- *Отражение* (относительно какой либо из осей)

Композиция двумерных преобразований

Последовательное выполнение нескольких преобразований можно представить в виде единой матрицы суммарного преобразования.

Последовательное выполнение сдвигов

Сдвинем точку P1 на расстояние (Tx1, Ty1) в точку P2:

$P2 = P1 \cdot T1$.

Затем сдвинем точку P2 на расстояние (Tx2, Ty2) в точку P3:

$P3 = P2 \cdot T2 = (P1 \cdot T1) \cdot T2) = P1 \cdot (T1 \cdot T2) = P1 \cdot T$.

Сдвиг аддитивен, т.е. последовательное выполнение T1 и T2 д.б. эквивалентно одному сдвигу на расстояние (Tx1+Tx2, Ty1+Ty2).

Для доказательства этого рассмотрим произведение матриц сдвига T1 и T2.

Композиция двумерных преобразований

Для доказательства этого рассмотрим произведение матриц сдвига T1 и T2.

Таким образом результирующий сдвиг есть (Tx1+Tx2, Ty1+Ty2), т.е. суммарный сдвиг, вычисленный как произведение матриц, как и ожидалось, аддитивен.